Deep-Car: Fine-grained Detection of Vehicle Model

Tong Qiu, Jerry Sun, Charlie Wu
Department of Computer Science, University of Virginia, Charlottesville, VA 22904
[tq7bw, ys7va, jw7jb]@virginia.edu

Abstract

This class project explores the deep learning based fine-grained object detection specifically on the vehicle make/model categories. A fine-tuning based approach is performed on several pre-trained CNN models, achieving over 90% *accuracy on a subset of 115 vehicle model classes. An RA-CNN based model is implemented to achieve better accuracy in recognizing more classes without any object part label compared to those Part-based RN-CNNs. This model will crop and up-sample the attended region in a coarse image to allow the model to recognize more features in details. The training strategy is used to iteratively and alternatively optimize weights in the classifier and APN (Attention Proposal Network) with pre-trained VGG-16 CNN features extractor.*

1. Introduction

While classification between different categories of objects requires proper deep learning techniques, differentiating between species of the same type of object is even more challenging. However, the latter is usually the goal we desire. Therefore, fine-grained training in neural networks is needed to achieve this result. A common technique for fine-grained training is fine-tuning an existing and pre-trained neural network model, such as VGG or Resnet. During this process, one needs to care more about details of the image, and focus on extracting local important features. More Recently, lots of new approaches have been suggested in exploiting the effectiveness of the local features for fine-grained classification [1, 4] Among which, we are particularly interested in the one called "Look Closer to See Better"[2]. The general idea of the paper is to extract the region in the original image which has larger effectiveness[5] in classification when globally classified, and do further classification on finer regions.

In this project, our goal is to classify the make and model of cars in a given set of images. Many existing datasets (see Section 2) already contain a large number of labelled car images, with hundreds of different categories of car makes and models. So our main tasks are choosing an appropriate model to fine-tune, and optimizing the model accuracy using various techniques proposed in previous works.

Figure 1. Detecting that an image contains a car is probably easy, but classifying the make/model of the car requires more effort.

2. Related Work

2.1. Fine-Grained Detection

2.1.1 Part-based R-CNN

The work, by Ning Zhang et al. [10], proposed a model for fine-grained categorization leveraging deep convolution features computed on bottom-up region proposals illustrated in Figure.2. The basic approach is to first learn the object parts, then based on regional proposal algorithm it detects the object parts and learns the object from the parts and the geometric constraints between them.

2.1.2 Recurrent Attention CNN

The recent paper, also known as Look Closer to See Better [2], introduced a novel recurrent attention convolutional neural network (RA-CNN) which recursively learns discriminative region attention and region-based feature representation at multiple scales in a mutually reinforced way. The learning at each scale consists of a classification sub-network and an attention proposal sub-network (APN). The APN starts from full images, and iteratively generates region attention from coarse to fine by taking previous predictions as a reference, while a finer scale network takes as input an amplified attention region from previous scales in a recurrent way. The paper achieved the state of art results better than most existing models in different fine-grained

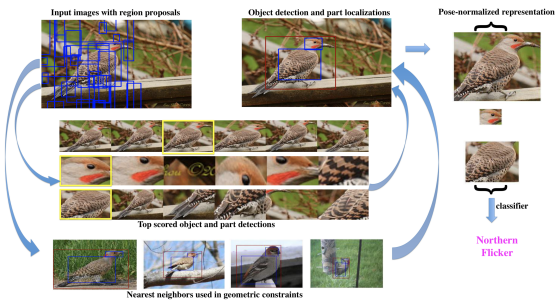

Figure 2. Overview of the part localization in Part-based R-CNN by [10] Starting from bottom-up region proposals (top-left), they train both object and part detectors based on deep convolution features. During test time, all the windows are scored by all detectors (middle), and we apply non-parametric geometric constraints (bottom) to re-score the windows and choose the best object and part detections (top-right). The final step is to extract features on the localized semantic parts for fine-grained recognition for a pose-normalized representation and then train a classifier for the final categorization.

dataset, Stanford Dogs(120 classes)[3], Stanford Cars (196 classes)[6] and CUB-200-2011(200 classes)[8] with a recurrent structure of three scales. Moreover, the attention region in the last scale level indeed focus on the car make label, bog face, bird head respectively as the look-closer human perception suggests.

3. Model Overview

3.1. Fine-tune

The first model we propose is based on the fine-tuning of several existing network including VGG16, ResNet152, ResNet18 on the car brand & make using different paramter settings. There are several different existing dataset that we are using, so first thing we need to do is to normalize all the dataset available. Another potential idea to implement is orientation normalization [7]. The idea is to first extract the orientation of the vehicle to be detected, so that this piece of information can also be passed into the network.

3.2. RA-CNN

The second model we are using is a simplified version of RA-CNN we mentioned above. The network is essentially divided into 2 parts. The first part is the Attention Proposal Network, which is almost similar to a bounding box proposal. And the second part is multiple fine-tuned VGG networks. For each layer there is a APN network and a CNN network using pre-trained fine-tune classifier we trained in the previous section. Then for each layer, the APN will propose a bounding box and do the cropping so that a smaller region will be proposed to the next layer and recurrently run through the same task again. Finally the output will be concatenated together and fed into another linear layer to produce the final predictions.

3.2.1 APN

The Attention Proposal Network is used for finding the correct/interesting location for the subsequent layer for to further investigate using CNN and fine-tuned classifier. More specifically, the input of the APN will be the flattened feature output of VGG network on the current layer which is of size 25088. Then it will run through a simple classifier including a linear layer, a drop-out layer and a $\tanh$ layer, so that the final output will be 3 values including the 2D position of the center location of the next region as well as the diameter of the new region.

4. Implementation Detail

4.1. Fine-tune CNN

We tested the Fine-tune CNN using Resnet-18, VGG-16 and Resnet-152. The output of each network will be connected to a sequential module of linear layers, followed by a softmax, so the output will be the probability distribution of the classification of the car model. We then calculate both top-1 and top-5 loss with regard to the correct label.

4.2. RA-CNN

The forward path of the RA-CNN model is overall clear: the first part is the APN, which uses the feature vector retrieved from the pre-trained CNN network to produce a prediction of interested region. This can be done using a sequential module with 2 fc-layers to reduce the dimension from 25088 to 3 which correspondingly stands for the 2D location of the center of the region as t_x, t_y and the radius of the region as t_l. The related formulae are:

$$p(\mathbf{X}) = f(\mathbf{W}_c * \mathbf{X}) \tag{1}$$

$$[t_x, t_y, t_l] = g(\mathbf{W}_c * \mathbf{X}) \tag{2}$$

where $\mathbf{X}$ is the input image, $\mathbf{W}_c$ denotes the convolution layer operations, f is the classifier, and g is the APN.

The second part of the APN is an attention crop layer, whose forward and backward paths require a customized implementation. Therefore, we will need to manually handle the forward and backward pass by writing a class that inherits from `torch.autograd.Function`.

4.2.1 Forward

The forward paths involves two parts, the first part handles the cropping and the second part is the up-sample layer which is provided by `PyTorch`. In the first cropping procedure, the range of the t_x, t_y from the APN is bounded by

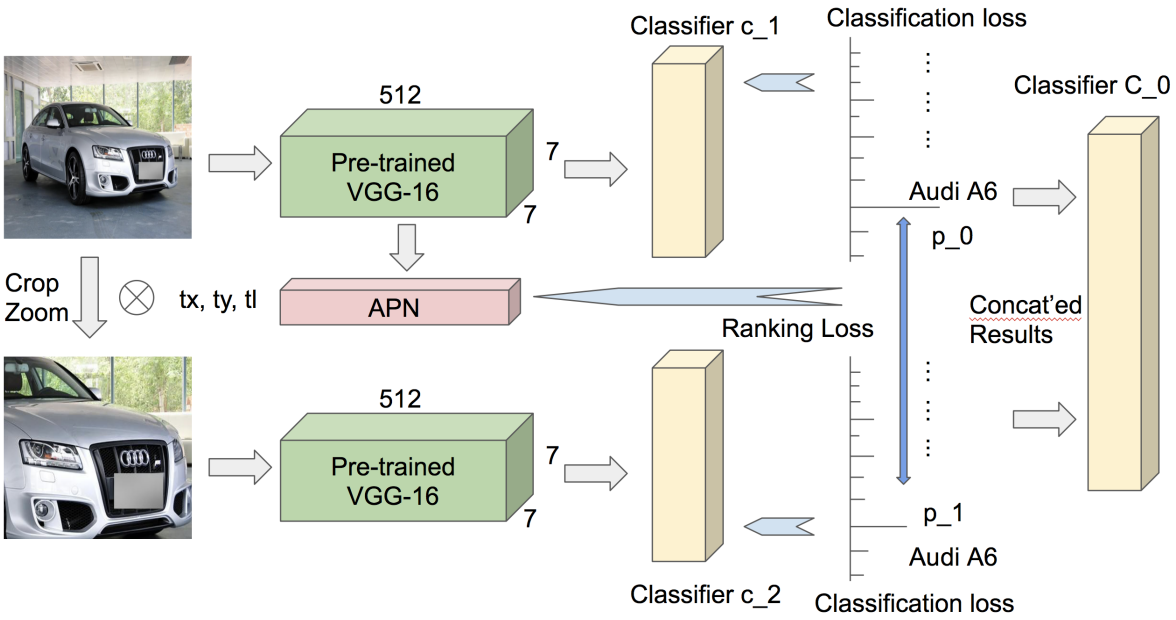

Figure 3. The network architecture we implement: The inputs are from coarse full-size images to finer region attention. Different network modules for classification (marked in yellow) and attention proposal (marked in red) are alternatively optimized by classification losses between label prediction Y(s) and ground truth Y$\star$ at each scale, and pairwise ranking losses between p(0) and p(1) from the two scales, where p(0) and p(1) denote the probabilities on the correct category, and s denotes the scale. APN is the attention proposal network with two fully-connected layer, the classifier is three fully connected layer with softmax layer matches to category entries. The "crop" and "zoom in" operation is used from convert coarse image to finer region attention with the $\{t_x, t_y, t_l\}$ predicted from the APN model.

-1 and 1, since the final layer is the $\tanh$ activation function. So we amplify them 56 times and add 56 to it, so they result in x, y coordinates between 56 and 168 (the center region of the image). Similarly, we set t_l to range from 28 to 56. In this way, we've made sure that the cropped image won't go out of bound.

The crop procedure is implemented by an element-wise multiplication between the original image and an attention mask, which can be computed from t_x, t_y, t_l we generated by a logistic function, according to [2]. Specifically, let $\mathbf{X}$ denote the image, $\mathbf{M}$ the attention mask, and h the logistic function. The first step is to generate an attention region, which has the formula:

$$\mathbf{X}^{att} = \mathbf{X} \cdot \mathbf{M}(t_x, t_y, t_l) \tag{3}$$

where $\mathbf{M}$ is determined by t_x, t_y and t_l as:

$$\begin{aligned} tx(tl) &= t_x - tl, \quad ty(tl) = t_y - tl \\ tx(br) &= t_x + tl, \quad ty(br) = t_y + tl \end{aligned} \tag{4}$$

$$\begin{aligned} \mathbf{M}(\cdot) = &[h(x - tx(tl)) - h(x - tx(br))] \\ &\cdot [h(y - ty(tl)) - h(y - ty(br))] \end{aligned} \tag{5}$$

The generated image is then upsampled to size 224×224 using `nn.upsampler`, to feed into the next level of CNN.

4.2.2 Backward

The gradient of the attention crop layer is completely unknown to the auto gradient mechanism, so we need to specify the entire backward path. In other words, we need to implement the gradient for the crop and the bilinear up-sample operation. From [2], the gradient from the ranking loss is proportional to the element-wise multiplication of the gradient output from the second CNN layer, and the gradient of the APN, which can be characterized as:

$$\frac{\partial L_{rank}}{\partial t_x} \propto \mathbf{D}_{top} \odot \frac{\partial \mathbf{M}(t_x, t_y, t_l)}{\partial t_x} \tag{6}$$

Furthermore, the paper specifies that the gradient of $\mathbf{M}$, or the APN, should have qualitative results as follows:

$$\mathbf{M}'(t_x) = \begin{cases} < 0 & x \rightarrow t_{x_{tl}} \\ > 0 & x \rightarrow t_{x_{br}} \\ = 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\mathbf{M}'(t_y) = \begin{cases} < 0 & y \rightarrow t_{y_{tl}} \\ > 0 & y \rightarrow t_{y_{br}} \\ = 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\mathbf{M}'(t_l) = \begin{cases} > 0 & \{x, y\} \rightarrow t_{\{x,y\}_{tl}} \text{or } t_{\{x,y\}_{br}} \\ < 0 & \text{otherwise} \end{cases} \tag{9}$$

In the above expression, "$\rightarrow$" means "approaching to". Since the images have size 224×224, we can set a metric of 56 pixels to evaluate this condition: a pixel is considered "approaching" the specific border if it is within 56 pixels away from that border.

To put everything together, we first calculate the negative norm of the gradient output, which can be visualized as an "energy map" of the cropped image. Then we incorporate equations (7)-(9) by applying a pre-calculated mask with values 1, 0 or -1 to the norm. Finally, since t_x, t_y and t_l are scalars, the masked values for each pixel is summed up to propagate the gradient into the APN. The whole process can be visualized in the following figure:

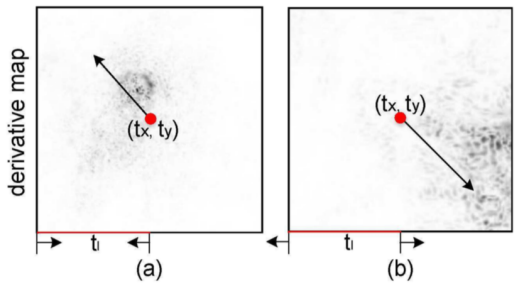

Figure 4. Visualize how (t_x, t_y, t_l) move based on derivative map

4.3. Training Detail

First, we initialize convolution layers with the same pre-trained VGG-16 network from ImageNet, the classifier layers from Fine-tune VGG-16 we obtained previously.

Second, we consider a square (represented by t_x , t_y , t_l) with the half length of the side of original image. The square is selected by searching regions in the original image, with the highest response value in the last convolution layer. These selected squares are used to pre-train APN to obtain parameters by learning the transformation from convolution feature maps to $\{t_x, t_y, t_l\}$.

Third, we optimize the parameters in the above two steps in an alternative way. We keep APN parameters unchanged, and optimize the classification losses at three scales to converge. Then we fix parameters in convolution and classification layers, and use ranking loss to optimize the two APNs. The learning process for the two parts is iterative, until the two types of losses no longer change. Besides, t_l at each scale is constrained to be no less than one-third of the previous t_l at coarse scale, since presumably the result will usually go bad when t_l is predicted too small.

The pairwise ranking loss we used to enforces $p^{(1)} > p^{(0)} + margin$ in training is:

$$L_{rank}(p_t^{(0)}, p_t^{(1)}) = max(0, p_t^{(0)} - p_t^{(1)} + margin) \quad (10)$$

where $p_t^{(0)}$ and $p_t^{(1)}$ is the probability predicted for the label class in first and second scale.

Based on the above derivation, the total loss of this model is given by:

$$L(X) = \sum_{s=1}^{2} L_{cls}(Y^{(s)}, Y^*) + L_{rank}(p_t^{(0)}, p_t^{(1)}) \quad (11)$$

where the classification loss L_{cls} is implemented with cross entropy loss.

5. Experiments and Result

5.1. Dataset & Data Augmentation

The first part of our work is to pre-process the large dataset we mentioned above and organize them into a organized fashion, since the both car dataset below have thousands of models/makes and with different poses or external noises including light, etc. Also we want to pre-process the image to make sure the cost of memory during training is under control.

5.1.1 CompCars Dataset by CUHK

The Comprehensive Cars (CompCars) Dataset by CUHK [9] contains data from two scenarios, including images from web-nature and surveillance-nature. The web-nature data contains 163 car makes with 1,716 car models. There are a total of 136,726 images capturing the entire cars and 27,618 images capturing the car parts. The full car images are labeled with bounding boxes and viewpoints. Each car model is labeled with five attributes, including maximum speed, displacement, number of doors, number of seats, and type of car. The surveillance-nature data contains 50,000 car images captured in the front view. Please refer to our paper for the details.

5.1.2 Data Augmentation

We have augmented the data obtained from the CompCars dataset with the following techniques, in order to provide more data samples to train our models:

1. *Color Perturbations*: To make our model able to predict cars of all colors and under all lighting conditions, etc, we augment brightness, contrast, saturation and the color in the HSV color space to spice up the training data.

2. *Rotations and Projective Transformations*: Slight rotations and horizontal flip of car images from different axes and angles are made to increase the robustness of our model.

3. *Random Crop*: Randomized crop operation to the raw image when it is scaled to the size of model input

5.2. Training

The validation percentage accuracy and loss after 200 epochs for each model we tested are included in the following table:

Model	Top-1	Top-5	Loss
VGG-16	76.5	93.2	0.914
ResNet-18	32.8	62.3	3.19
ResNet-152	36	67.6	3.25
RA-CNN	78.1	93.7	0.897

Table 1. Performance comparison of different pre-trained deep CNN model in 200 epochs based on the CUHK Comprehensive Cars Dataset[1]

We can find that RA-CNN and VGG-16 stand out in this case and RA-CNN is marginally better then the fine-tune model. The detailed loss and accuracy plots for each model can be found in the appendix section. The `PyTorch` implementation for all the models we trained is made publicly available at `https://github.com/Charleo85/DeepCar`

5.3. Inference

Attached are several sample inference results we got from evaluation set. First, the below are two of the validation images for the classifier; the prediction results are all correct with relatively high confidence:

Figure 5. Fine-tuned VGG-16 Model predict image: BWM Z4 with confidence 0.99; Benz E Class with confidence 0.95

For the RACNN, the classifiers are similar to the fine-tuned models. To avoid redundancy, we just include the results of the APN. Below are one original image from the validation set, the result after logistic crop, and the amplified region to feed the next CNN, respectively:

Figure 6. Result of attention crop: the model successfully focused on the region with the Audi logo

[1]With the same learning rate, the ResNet-18 and ResNet-152 fine-tuning results are unreasonably bad than VGG-16. In future work, we shall try more learning rate for ResNet fine-tuning result

6. Conclusion

This class project explores the deep learning based fine-grained object detection specifically on the vehicle make/model categories. The fine-tune result on pre-trained CNN model is reasonably good on a small subset of class, while with more class, the accuracy performance will go down significantly. The RA-CNN based model is implemented intentionally achieve better accuracy in recognizing more classes without any object part label compared to those Part-based RN-CNNs. However, the improvement is rather limited, compared to the training effort underwent. Specifically, it is a more complicated model to implements with more hyper-parameters to adjust.

The reason that the accuracy improvement is limited is probably the attention layer classifier accuracy is more or less bounded by that of first scale classifier. Meanwhile, in many cases, it might be better to have several attentions in a single image. Indeed, if we have more time, we can potentially try to implement multi-attention network by H. Zheng et al.[11] and potentially integrate the multiple attention mechanism with the current RA-CNN model.

Acknowledgments

Thanks to Prof. Vicente Ordóñez for instructions and feedback on this class project. All of the experiments are conducted on the Google Cloud Nvidia P100 GPU compute instance. Our model implementation with `PyTorch`.

A. APPENDIX

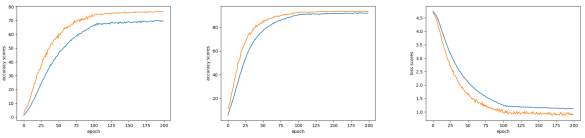

Figure 7. VGG-16 top5 accuracy, top1 accuracy, and loss trend in 200 epoches; blue = train, orange = validate

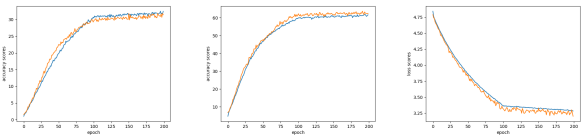

Figure 8. Resnet-18 top5 accuracy, top1 accuracy, and loss trend in 200 epoches; blue = train, orange = validate

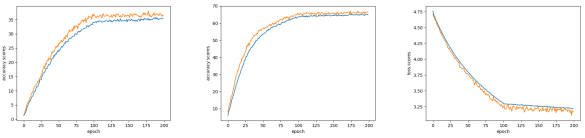

Figure 9. Resnet-156 top5 accuracy, top1 accuracy, and loss trend in 200 epoches; blue = train, orange = validate

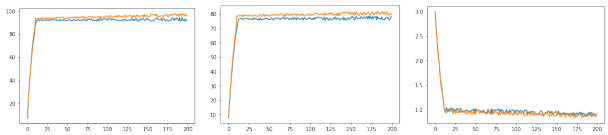

Figure 10. RACNN (final concatenated layer) top5 accuracy, top1 accuracy, and loss trend in 200 epoches; blue = train, orange = validate

References

[1] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2013.

[2] J. Fu, H. Zheng, and T. Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition.

[3] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, volume 2, page 1, 2011.

[4] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5546–5555, 2015.

[5] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. *The Unreasonable Effectiveness of Noisy Data for Fine-Grained Recognition*, pages 301–320. Springer International Publishing, Cham, 2016.

[6] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[7] J. Sochor, J. Spanhel, and A. Herout. Boxcars: Improving vehicle fine-grained recognition using 3d bounding boxes in traffic surveillance. *CoRR*, abs/1703.00686, 2017.

[8] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[9] L. Yang, P. Luo, C. C. Loy, and X. Tang. A large-scale car dataset for fine-grained categorization and verification. *CoRR*, abs/1506.08959, 2015.

[10] N. Zhang, J. Donahue, R. B. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. *CoRR*, abs/1407.3867, 2014.

[11] H. Zheng, J. Fu, T. Mei, and J. Luo. Learning multi-attention convolutional neural network for fine-grained image recognition (iccv 2017 oral). October 2017.