CZ4042 Neural Network & Deep Learning
Car Model Recognition

Jun Liang Ang, Nigel Ang Wei Jun, and Wei Min Yee

Nanyang Technological University, Singapore

Abstract. Computer Vision (CV) is one of integral branches of Artificial Intelligence and widely applied to understand and analyze visual imagery. One of the research areas of CV is Fine-Grained Image Classification (FGIC), which has been a fundamental and longstanding problem. The challenge in FGIC comes from the small inter-class variations and the large intra-class variations due to the fine-grained nature of the dataset. In recent years, in part due to the advent of Deep Learning, we have seen remarkable progress in FGIC. In this report, we focus on the Compcars dataset [1], with the goal of fine-grained car model categorization. We combined various methods from MMAL-Net [2] and RandAugment [3] and Multitask Learning to perform the classification. Specifically, the attention object location module (AOLM) can predict the position of the object and the attention part proposal module (APPM) can propose informative part regions without the need of bounding-box or part annotations. Together with the original image, three kinds of training images (object image, part image and original image) are used by the multi-branch network for supervised learning. Multitask Learning was applied to learn classifiers for both car model and car make simultaneously. Additionally, image augmentation with RandAugment was used to further improve the performance of the model. An accuracy of 98.3% for car model was obtained with our approach, achieving state-of-the-art over the previous 95.9% by Tanveer et al. [4].

Keywords: Image Classification · Attention · Location · Scale · Augmentation · Multitask Learning.

1 Introduction

1.1 Car Model Analysis

Recent years have seen a surge in research and innovation of automated car model analysis. Motivations include creating intelligent transportation systems which can perform regulation, description and indexing of cars. Additionally, the ability to classify cars according to car models allow many possibilities such as improving the performance of traffic control systems, speed and red light cameras, automating toll fees according to car types, and many more.

1.2 Convolutional Neural Networks

Following the initial success of Convolutional Neural Network (CNN) in ImageNet, it is widely agreed that CNNs are the best tool for most computer vision tasks including image classification. Compared to its predecessors, they work extremely well because they can automatically detect important features rather than use laboriously engineered features. Additionally, the convolution and pooling operations are computationally efficient, making CNNs an even more attractive option.

1.3 Fine Grained Image Classification

However, FGIC is a much more difficult task than normal image classification due to the small inter-class variations and the large intra-class variations. FGIC is an important topic because it is key for performing a more detailed subcategory division for coarse-grained large categories. Some domains where FGIC is useful include bird species, aircraft models and car models. Previous studies [5] [6] demonstrated that identifying informative regions in an image is crucial for FGIC. Hence, annotations of bounding box and part type are provided to localize regions of interest. However, providing such annotations is a labor-intensive task. While there are other methods proposed [7] [8] to automatically localize important regions with unsupervised learning, their performance is not ideal as they lack the ability to focus on the correct areas.

2 Related Works

In recent years, various approaches to tackle FGIC have been proposed and refined upon. There are three main categories of FGIC approaches: 1) Human-in-the-loop; 2) End-to-end encoding; 3) Localization based.

2.1 Human-in-the-loop Approaches For FGIC

These approaches require a human to interactively identify the important and discriminating regions of interest during FGIC. Branson et al. [9] introduced a general framework that incorporates question answering by experts while reducing the amount of human interaction required. Deng et al. [10] introduced a game and algorithm that reveals discriminating features humans use. Wah et al. [11] introduced a framework where non-expert users are incorporated during inference stage to judge the relative similarity between a query image and various sets of images. While these approaches are novel and useful, they are not scalable to large scale image classification due to the need for human interaction.

2.2 End-to-end Encoding Approaches For FGIC

These approaches directly learn discriminating features from deep CNNs. The most representative method is a Bilinear-CNN [12] where bilinear architecture

that consists of 2 sub-CNNs is proposed. The image is represented as a pooled outer product of the features from the 2 sub-CNNs. The benefit of doing so is that the model can learn pairwise feature interactions that are translationally invariant, which is particularly useful for FGIC. Additionally, the model is able to capture higher order statistics of convolutional activations such as the Fisher vector [13], VLAD [14] and O2P [15] which enhances the learning ability of the model. Additional improvements include the CNN Tree [16] which is able to progressively learn fine-grained features specific to a subset of classes. This is motivated by the observation that the model makes errors for similar looking classes. Similarly, Selective Regularized Subspace Learning [17] conducts learning only on a subset of classes which look similar rather than the entire set of classes.

2.3 Localization Based Approaches For FGIC

These approaches usually involve localizing salient regions of interest which are either manually defined or automatically learnt. Liu et al. [18] used part localization and correspondence to perform FGIC for different dog breeds. Features from manually defined parts such as the eyes and face from different images are extracted and used to improve model accuracy. Berg et al. [19] introduced Part-based One-vs-One Features (POOFs) which is a large and distinctive set of highly discriminating features where each feature specializes in differentiating two different classes based on the appearance of the same part. Additionally, these features work well even if training data is scarce. The problem of earlier works is that they require more than image-level annotations. They also require part annotation which is expensive. For example, HSnet [6] is a novel technique that uses a LSTM to generate proposals of important image regions and merges them to create an overall image representation for FGIC. However, it still requires object part annotation which can be costly. As a result, attention-based approaches are proposed to avoid this problem. These approaches only require image-level annotations and do not require bounding box or part annotations. For example, Zheng et al. [7] proposed that part localization and fine-grained feature learning are mutually correlated and both can be learned together using attention. Yang et al. [8] proposed a self-supervision based framework to learn salient image regions with only image-level annotations. The model is based on multi-agent cooperation between a Navigator agent, a Teacher agent and a Scrutinizer agent. The Teacher helps the Navigator choose important image regions and the Scrutinizer makes predictions based on the proposed regions from the Navigator agent.

3 Methodology

In this section, We go through the main ideas behind the methods used in this project.

3.1 Overview

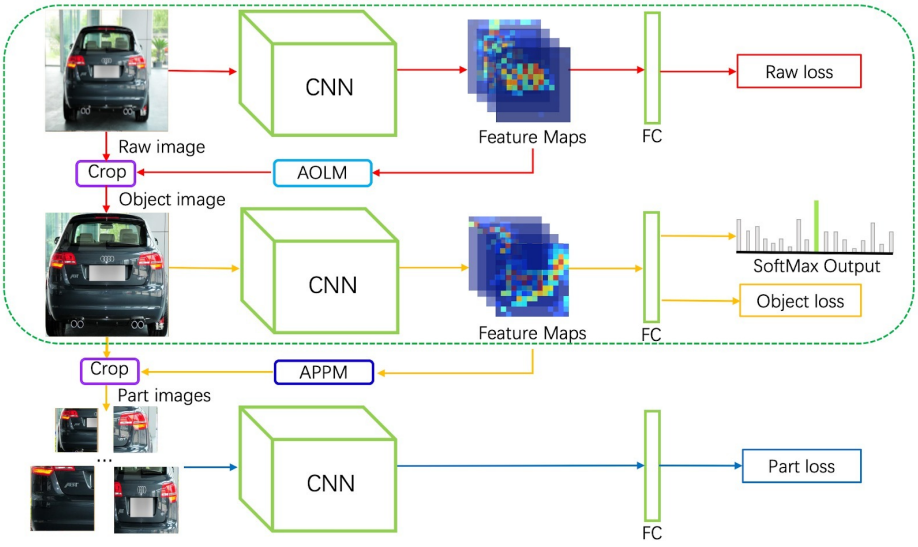

Fig. 1. Overview of MMAL-Net in the training phase. The red branch corresponds to original raw image, orange branch corresponds to object image, and blue branch corresponds to part image. In the test phase, only the dotted green box is involved, i.e. object image is directly used for classification without being cropped into part images. The convolutional and fully connected layers have the same color, meaning they share the same parameters. All branches use cross entropy loss.

MMAL-Net with ResNet-50 backbone was used as our FGIC model. An overview of MMAL-Net is shown in Fig. 1. In the training phase, three branches are involved. The raw branch that takes the original image learns the overall features in the object. AOLM obtains the object's bounding box from the original image feature maps produced by the raw branch. The cropped object image based on this bounding box is useful as it contains both the structural and fine-grained features of the object. APPM learns to propose several part regions from the object image feature maps produced by the object branch. These part regions proposals are chosen to maximize discriminative power and minimize redundancy. They are then used by the part branch to crop the object image into part images and sent to the network for training. These part images allow

the network to learn fine-grained features of different parts in different scales. The parameters of the CNN and FC in the three branches are intentionally shared so that the trained network has good performance in both different parts and different scales of object. Lastly, to improve inference time, classification is performed on the object branch during testing phase instead of the concatenation of feature vectors from the multiple part images.
Some of the reasons why we chose MMAL-Net over other models include:

- It can be trained end-to-end.
- It has faster training time compared to other part proposal models that do not share parameters.
- It only requires image-level annotations and can automatically learn to propose discriminative part images.
- It achieved state-of-the-art performances for 3 datasets, CUB-200-2011, FGVC-Aircraft and Stanford Cars.

3.2 Attention Object Location Module (AOLM)

This module is a non-parameterized technique to generate the object's bounding box position. This process is described in Fig. 2. The feature maps produced by the CNN have with C channels and are represented by $F \in R^{C \times H \times W}$ where $H \times W$ is the spatial output size from the last convolutional layer of an input image X and f_i is the feature map of the i-th channel. An activation map A can

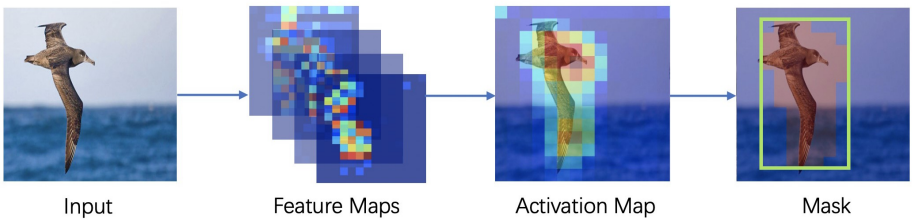

Fig. 2. The feature maps produced by the CNN are aggregated to produce an activation map where the bounding box is derived. Taken from MMAL-Net paper [2].

be obtained by summing the feature maps F as shown in Eqn. 1

$$A = \sum_{i=0}^{C-1} f_i \tag{1}$$

and the mean activation $\bar{a}$ can be obtained as shown in Eqn. 2

$$\bar{a} = \frac{\sum_{x=0}^{W-1} \sum_{y=0}^{H-1} A(x, y)}{H \times W} \tag{2}$$

The mean activation $\bar{a}$ serves as a threshold to determine if a point belongs to the object. A mask $\widetilde{M}_{conv\_5c}$ for the last convolutional layer *Conv_5c* of ResNet-50 is produced as shown in Eqn. 3. A point (x, y) is included if its activation is greater than the mean activation.

$$\widetilde{M}_{(x,y)} = \begin{cases} 1 \; if A_{(x,y)} > \bar{a} \\ 0 \; otherwise \end{cases} \tag{3}$$

A similar mask $\widetilde{M}_{conv\_5b}$ is produced for the second last convolutional layer *Conv_5b*. A final and more accurate mask M is obtained from the intersection of the 2 masks as shown in Eqn. 4. The corresponding bounding box for the object can then be obtained by taking the smallest bounding box containing the largest connected area in M.

$$M = \widetilde{M}_{conv\_5c} \cap \widetilde{M}_{conv\_5b} \tag{4}$$

3.3 Attention Part Proposal Module (APPM)

While AOLM can achieve high object localization accuracy, the model can be further improved by focusing on important feature regions. A sliding window approach is used to get each window's activation map A_w and its activation mean value $\bar{a}_w$. Next, the different windows are sorted based on the value of $\bar{a}_w$. The larger the value of $\bar{a}_w$, the more informative the part region is. However, adjacent windows approximate the same part and have similar $\bar{a}_w$. To reduce redundancy, non-maximum suppression (reject if intersection over union is exceeds threshold) is used to select a fixed number of windows with different scales as part images.

3.4 Architecture of MMAL-Net

In addition to the main raw branch for the original image, an object branch and part branch are needed to fully and effectively learn the object images from AOLM and part images from APPM, as shown in fig. 1. All three branches share the same CNN for feature extraction and the same FC layer for classification. All three branches use cross entropy loss for classification as shown in Eqn. 5, Eqn. 6 and Eqn. 7.

$$L_{raw} = -\log(P_r(c)) \tag{5}$$

$$L_{object} = -\log(P_o(c)) \tag{6}$$

$$L_{parts} = -\sum_{n=0}^{N-1} \log(P_{p_{(n)}}(c)) \tag{7}$$

where c is the ground truth label of the input image, P_r, P_o, $P_{p_{(n)}}$ are the categorical probabilities of the softmax output of the raw branch, softmax output of the object branch and softmax output of the part branch of the nth part image respectively. N is the number of part images. The total loss is defined as the sum of all three losses. The total loss is used to train the shared CNN and FC layers during backpropagation so that the final model not only learns the

overall structural characteristics of the object but also the fine-grained features of a part. In addition to high classification accuracy, MMAL-Net is also efficient when performing inference as the part branch is removed during the test phase.

3.5 Multitask Learning

Multitask Leaning (MTL) is a learning paradigm in machine learning whose aim is leverage useful information contained in multiple related tasks to help improve the generalization performance of all the tasks. For this project, we used a soft parameter sharing approach where the output layer has m output units with for m different tasks as shown in Fig. 3 compared to m different models for single task learning. Two different tasks are performed - classification of car model and classification of car make, with car model being the more fine-grained task. The total loss used for backpropagation to train the model is the sum of loss for each task. We tried Multitask Learning with the goal of improving the generalization accuracy for the classification of car models. We hypothesized that using both car make and car model labels will encourage the model to learn more generalized features useful for both tasks, improving generalization accuracy for classification.

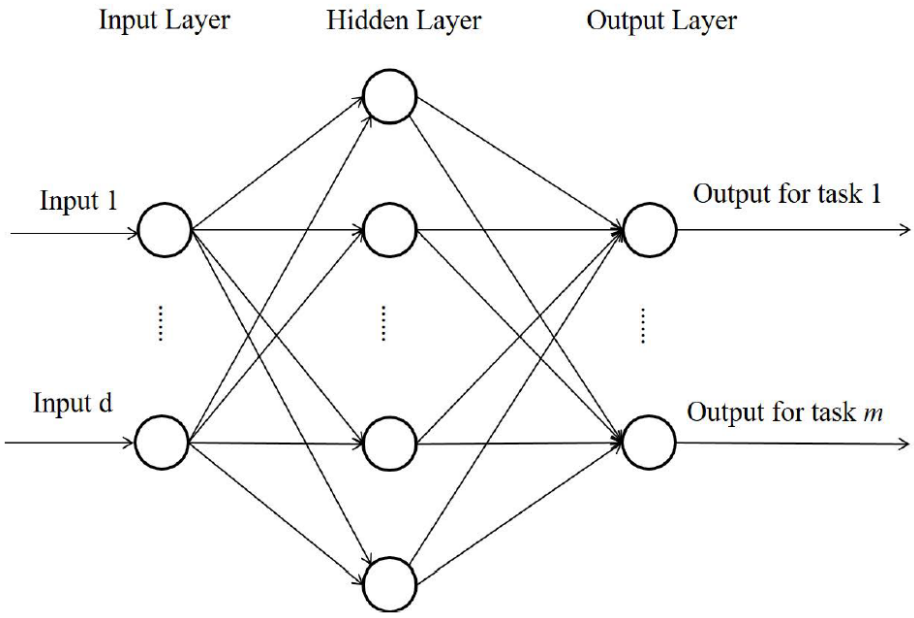

Fig. 3. The hidden layer represents all hidden layers between input and output.

3.6 Data Augmentation

While data augmentation can inject additional knowledge to train vision models, it is time consuming to manually design augmentation strategies for each new application. Hence, methods to automatically learn data augmentation strategies from data have been proposed to address the weaknesses of traditional data augmentation methods. However, many of these methods require a separate and expensive search space. Hence, we decided to use RandAugment [3] as our augmentation strategy. Experiments performed by Cubuk et al. [3] empirically showed two important observations.

Firstly, probabilities for applying each transformation is not crucial and can be fixed to $1/K$ for K different transformations. Secondly, the learned magnitudes of transformations tend to follow a similar learning schedule and their experiments show that a single global distortion M is sufficient for parameterizing all the magnitudes of transformations. As a result, only two parameters are needed to be optimized - the number of augmentation transformations to apply sequentially N and the magnitude of all the transformations M.

It is important to note that the original authors intended for RandAugment to replace the need for a proxy task due to its much smaller search space. However, due to time constraints, the grid search suggested by the authors was still computationally expensive, considering we also used cross-validation during grid search. Hence, we elected to use only 10% of the training dataset to select the best parameters of N and M.

4 Experiments & Results

4.1 Dataset & Preprocessing

The dataset used in this project is the CompCars dataset, which is much larger in scale and diversity than previous car image datasets. Specifically, we focused on entire car images for fine-grained classification. In total, there are 30955 images consisting of 16016 training images and 14939 test images across 431 car model and 75 car make (car manufacturers) categories. We elected to strictly follow the train and test splits originally provided, so we can compare our results to previous work.

The labels, bounding boxes, and car information (such as the car model, its maximum speed, number of seats etc.) were extracted from the various files in the CompCars dataset. We then performed data cleaning including One-Hot Encoding on car model and car make. This information is combined into a dataframe which contains the necessary information for input of training and test images for model training.

4.2 Experimental Procedures

Changes were made to the original implementation of MMAL-Net in PyTorch, to include support for the CompCars dataset, which was previously unsupported. This included loading and processing the CompCars dataset into training and test datasets and dataloaders.

We also implemented Stratified K-Fold Cross Validation using the scikit-learn package [20], which partitions the training dataset into 5 different combinations of 80-20 train-val splits. Each fold is stratified, made by preserving the percentage of samples for each class, allowing each fold to be a better representation of the entire dataset. This would be the workflow used when performing hyperparameter tuning or model selection. We also implemented EarlyStopping so that training will cease once the model has converged, signalled by no improvement in model metrics for a given number of epochs.

Multitask Learning was also implemented to test if it can improve generalization of car model classification. If the target number of classes were set as a tuple to indicate two target attributes (car make and car model), the datasets will be loaded with the appropriate target columns, and the model will be modified to facilitate training on the attributes via a separate branch with soft shared parameters.

RandAugment was also introduced in training if specified via a config parameter. The RandAugment transforms will be performed in addition to the specified image transforms by MMAL-Net. Note that we only apply augmentation during training.

4.3 Implementation Details

The details of MMAL-Net follows that of the original PyTorch implementation.

The images will be preprocessed into images of size 448 x 448 before inputting to the raw branch (red branch in Fig.1). After retrieving the object images from AOLM, the object images will again be scaled into 448 x 448 before inputting to the object branch (orange branch in Fig.1). The part images retrieved from APPM will also resized to 224 x 224 before inputting to the part branch (blue branch in Fig.1). For APPM, we considered windows with three categories of scales W1=[4 x 4, 3 x 5], W2=[6 x 6, 5 x 7], W3=[8 x 8, 6 x 10, 7 x 9, 7 x 10] to perform 2D convolution on the activation map of size 14 x 14.

N=7 part images will be retrieved from APPM, 2 images from W1, 3 images from W2, and 2 images from W3. Note also that this parameter N is not the same as the hyperparameter N used in our experiments with RandAugment, however we will elect to retain the original nomenclature of the parameters for both APPM and RandAugment.

A pretrained ResNet-50 [21] model is used as the backbone of our network structure. For all training and testing, only image-level labels are used. The optimizer used is the SGD (Stochastic Gradient Descent) optimiser, with the parameter settings of momentum=0.9, weight decay=0.0001 and batch size=6. Learning Rate is specified to be 0.001 and is multiplied by a factor of 0.1 on epoch 60 and multiplied again by 0.1 on epoch 100 via a Learning Rate Scheduler.

4.4 Multitask Learning Results

To determine whether Multitask Learning can improve the task of fine-grained car model classification, we devised two sets of experiments.

Firstly, Stratified 5-fold Cross Validation on the training dataset was performed with only one task (car model classification). After which, another separate model is trained on car make classification in the same way.

For our second experiment, Stratified 5-fold Cross Validation on the training dataset was performed with Multitask Learning (on both car model and car make classification).

In sum, we have 3 models: one trained on the classification of car models, one trained on classification of car make, and one trained on both attributes via Multitask Learning.

Fig. 4. Depiction of the 80:20 Train/Validation dataset splits. The training dataset of CompCars is split into 5 folds. Each 20% subset is used as the validation set once during each fold's run.

The above Cross Validation results are reported in Table 1.

Multitask Learning?	CV Accuracy for car model	CV Accuracy for car make
No	96.78% ± 0.25%	99.04% ± 0.12%
Yes	96.12%± 0.25%	99.28% ± 0.13%

Table 1: Results with and without Multitask Learning

4.5 RandAugment Results

We observed the models' cross validation losses across different values of N and M via grid search to determine the optimal N and M. Note that due to time constraints, experiments are performed on a reduced training dataset containing only 10% of the original training dataset, where CV is then performed on.

No Multitask Learning was performed with RandAugment as we have previously determined that it did not improve car model classification.

Fig. 5. Depiction of the the reduced 10% dataset used in Cross Validation during hyperparameter tuning of N and M. 10% of the original CompCars Training dataset is taken, then the training/validation splits are performed and Cross Validation is performed.

The results of CV on the reduced training and validation datasets are reported in Table. 2.

N	M	CV Loss	CV Accuracy
0	0	10.7587 ± 0.0472	95.57% ± 0.84%
1	5	10.7551 ± 0.0834	96.07% ± 1.28%
1	7	10.7499 ± 0.0463	96.19% ± 0.77%
1	9	10.7224 ± 0.0653	96.69% ± 0.85%
1	11	10.7308 ± 0.0512	96.94% ± 1.11%
1	13	10.7263 ± 0.0727	96.00% ± 1.18%
1	15	10.6626 ± 0.0298	96.94% ± 0.57%
2	5	10.7311 ± 0.0663	96.38% ± 0.75%
2	7	10.7427 ± 0.0451	95.94% ± 1.00%
2	9	10.7362 ± 0.0599	96.32% ± 1.32%
2	11	10.6967 ± 0.0390	96.57% ± 0.98%
2	13	10.7339 ± 0.0398	96.38% ± 0.61%
2	15	10.6814 ± 0.0442	96.75% ± 0.87%
3	5	10.7554 ± 0.0962	96.13% ± 1.46%
3	7	10.7147 ± 0.0752	96.19% ± 1.38%
3	9	10.7414 ± 0.0665	95.88% ± 0.66%
3	11	10.7380 ± 0.0965	96.44% ± 1.73%
3	13	10.7408 ± 0.0263	95.82% ± 0.61%
3	15	10.6668 ± 0.0613	96.88% ± 0.86%

Table 2: Results for different N and M for RandAugment

Note that for (N=0, M=0), the validation accuracy of 95.57% comes from training on the reduced 10% training dataset, as opposed to the 96.78% obtained from Cross Validation on the full training dataset (No Multitask Learning).

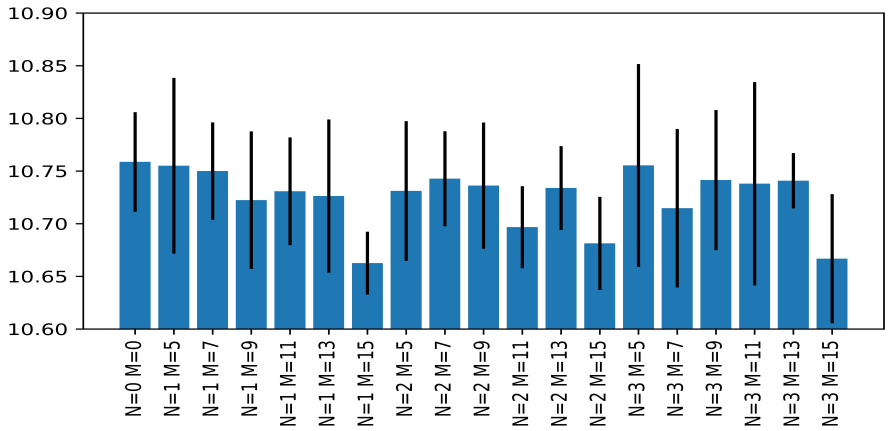

Fig. 6. Results of Cross Validation on the reduced 10% dataset. The mean validation losses and standard deviation are shown. (N=1, M=15) has the lowest mean validation loss.

4.6 Final Model

Based on the results above, we elected to use RandAugment but not Multitask Learning for the final model. The final model was trained with the optimal parameters of N=1 and M=15 for RandAugment, this time on the full training dataset. The model was then tested on the test dataset according to the original dataset splits of the CompCars dataset and a test accuracy of 98.3% was achieved. This was the one and only time we used the test dataset to ensure our results are not biased. All previous experiments and models were evaluated with cross validation.

5 Discussion & Conclusion

5.1 Analysis of Multitask Learning Experiments

We found that the use of Multitask Learning actually reduced model performance for car model classification. However, Multitask Learning can still be useful in reality as we can have a single model that performs different tasks, which is convenient and efficient in production systems. Multitask Learning works when the tasks are related by means of a common low dimensional representation, which is learned jointly with the tasks' parameters. We hypothesize that for our case, the tasks of classifying car model and classifying car make are too similar, so there were no new or different features learnt compared to the model without Multitask Learning.

5.2 Analysis of RandAugment Experiments

We found that RandAugment significantly improved the model based on cross validation accuracy. With RandAugment, the cross validation accuracy on the reduced 10% CV train and validation datasets improved from 95.57% (No RandAugment: N=0, M=0) to 96.94% (Optimal RandAugment hyperparameters: N=1, M=15). Additionally, We noted that for N>0, the average number of epochs elapsed before training ceases due to EarlyStopping is larger than when N=0. This is to be expected as when N=0, no augmentation is performed so the model can converge more quickly. When augmentation is performed, the model takes more epochs to learn the general features of augmented images.

5.3 Visualization of Object and Part Regions

To visualize the output of AOLM and APPM, we draw the object's bounding box proposed by AOLM and the part regions proposed by APPM in Fig.7. For the left image, the red rectangle represents the object bounding box ground truth and green rectangle represents the object bounding box proposed by AOLM. From the image, we can see that AOLM is able to do a good job. For the two right images, red/orange/green rectangles denote the proposed part regions, with

red being the one with the highest activation. The red rectangle, which is the window with the highest activation, contains the logo of the car, which highly suggests the car brand strongly determines the car model.

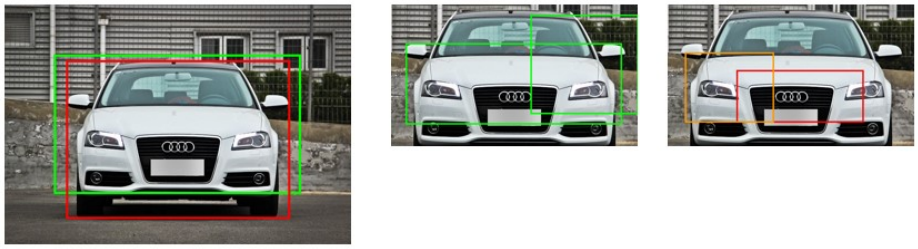

Fig. 7. Proposed AOLM object and APPM part regions from model.

5.4 Conclusion

In this project, we used MMAL-Net and RandAugment to achieve state-of-the-art classification accuracy on the CompCars dataset. Additionally, we found that the simple Multitask Learning approach via soft parameter sharing for 2 tasks (car model classification and car make classification) did not improve car model classification. We ensured that the test dataset was only used once so there was no bias in model selection and hyperparameter tuning.

From this project, we gained knowledge in topics related to Deep Learning including fine-grained image classification and Multitask Learning. The comprehensive literature review exposed us to the history of improvements in various fields to state-of-the-art techniques of today like RandAugment.

References

1. Yang, L., Luo, P., Change Loy, C., & Tang, X. (2015). A large-scale car dataset for fine-grained categorization and verification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3973-3981)
2. Zhang, F., Li, M., Zhai, G., & Liu, Y. (2020). Multi-branch and Multi-scale Attention Learning for Fine-Grained Visual CategorizationarXiv e-prints, arXiv:2003.09150
3. Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 702-703)
4. Tanveer, M. S., Khan, M. U. K., & Kyung, C. M. (2020). Fine-Tuning DARTS for Image Classification. arXiv preprint arXiv:2006.09042

5. Lin, T. Y., RoyChowdhury, A., & Maji, S. (2015). Bilinear cnn models for fine-grained visual recognition. In Proceedings of the IEEE international conference on computer vision (pp. 1449-1457)
6. Lam, M., Mahasseni, B., & Todorovic, S. (2017). Fine-grained recognition as hsnet search for informative image parts. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2520-2529)
7. Zheng, H., Fu, J., Mei, T., & Luo, J. (2017). Learning multi-attention convolutional neural network for fine-grained image recognition. In Proceedings of the IEEE international conference on computer vision (pp. 5209-5217)
8. Yang, Z., Luo, T., Wang, D., Hu, Z., Gao, J., & Wang, L. (2018). Learning to navigate for fine-grained classification. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 420-435)
9. Branson, S., Wah, C., Schroff, F., Babenko, B., Welinder, P., Perona, P., & Belongie, S. (2010, September). Visual recognition with humans in the loop. In European Conference on Computer Vision (pp. 438-451). Springer, Berlin, Heidelberg
10. Deng, J., Krause, J., & Fei-Fei, L. (2013). Fine-grained crowdsourcing for fine-grained recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587)
11. Wah, C., Van Horn, G., Branson, S., Maji, S., Perona, P., & Belongie, S. (2014). Similarity comparisons for interactive fine-grained categorization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 859-866)
12. Lin, T. Y., RoyChowdhury, A., & Maji, S. (2015). Bilinear cnn models for fine-grained visual recognition. In Proceedings of the IEEE international conference on computer vision (pp. 1449-1457)
13. Zheng, H., Fu, J., Mei, T., & Luo, J. (2017). Learning multi-attention convolutional neural network for fine-grained image recognition. In Proceedings of the IEEE international conference on computer vision (pp. 5209-5217)
14. Jégou, H., Douze, M., Schmid, C., & Pérez, P. (2010, June). Aggregating local descriptors into a compact image representation. In 2010 IEEE computer society conference on computer vision and pattern recognition (pp. 3304-3311). IEEE
15. Carreira, J., Caseiro, R., Batista, J., & Sminchisescu, C. (2012, October). Semantic segmentation with second-order pooling. In European Conference on Computer Vision (pp. 430-443). Springer, Berlin, Heidelberg
16. Wang, Z., Wang, X., & Wang, G. (2018). Learning fine-grained features via a CNN tree for large-scale classification. Neurocomputing, 275, 1231-1240
17. Luo, C., Ni, B., Yan, S., & Wang, M. (2015). Image classification by selective regularized subspace learning. IEEE Transactions on Multimedia, 18(1), 40-50
18. Liu, J., Kanazawa, A., Jacobs, D., & Belhumeur, P. (2012, October). Dog breed classification using part localization. In European conference on computer vision (pp. 172-185). Springer, Berlin, Heidelberg
19. Berg, T., & Belhumeur, P. N. (2013). Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 955-962)
20. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python In Journal of Machine Learning Research pp. 2825-2830 (2011)
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770-778 (2016)