

Dog/Cat Breed Image Classification/Localization/Segmentation with Transfer Learning

Haenara Shin
Materials Science and Engineering
University of California San Diego
has163@ucsd.edu (A53233226)

Abstract

Fine-grained image recognition is an interesting topic to deep learning learners because it has not only small inter-class variations, but also large intra-class variations. I investigate the fine-grained object categorization problem of determining the breed of dogs and cats from images. To do this, there are the Oxford-IIIT Pet dataset[1], which has 37 different breeds of dogs and cats.

I made a number of contributions: first, I build a model to classify a pet breed automatically from an image using convolutional neural networks (CNN). From a vanilla CNN to pre-trained models, I try to enhance the performance (especially, the accuracy) of models. Here, I observe that DenseNet121 shows the best accuracy, and MobileNetV2 is the best cost-efficient model. In addition, I investigate the localization of dogs and cats, focused on their head. While studying the localization, I implement the class activation map and its result to the multi-task learning to do the localization. This multi-task learning with pre-trained models shows the improved performance (especially, the intersection of union). Finally, I have experimented image segmentation of U-Net with VGG-16 and VGG-19.

1. Introduction

Research on object recognition has largely concentrated on the discrimination of distinguished objects, which means to the generic image recognition. Even in the larger ImageNet database[2], categories are defined based on a high-level hierarchy and, as such, any visual similarity between them is more random than systematic. This ImageNet work focuses instead on the discriminating different breeds of cats and dogs, a challenging example of fine-grained object classification like that of previous work classification[3]. The difficulty is in fact that breeds have very small variations, as opposite to the species classification, which it has large variations.

Beyond this technical challenging of fine-grained classification[4], the classification of breeds in dog or cats is hard to non-expert people. Also, some breeds are not familiar with people. Therefore, there is a need for automated classification method to fine grained image recognition to help non-experts in the daily life. In addition

to this, this classification model can be good candidate to object detection, including products recognition, or another bio-diversities classification.

2. Method

2.1. Dataset

This paper is the introduction of a large annotated collection of images of 37 different breeds of cats and dogs. It includes 12 cat breeds and 25 dog breeds. Roughly 200 images per each breed with (a) associated ground truth annotations of species and breed names, (b) a tight bounding box ROI around the head, and (c) a pixel level foreground-background segmentation(Figure 1)[1]. This dataset, which is publicly available, has the images that have large variations in scale, pose and lighting. After downloading the dataset, the first thing to do is deleting non-3 channel images. Labeling with each class is necessary step for classification, and then split the dataset as 75% of training set and 25% of validation set. To easy access with smaller memory usage, TensorFlow record files (TFRecord) for training and validation set are used.

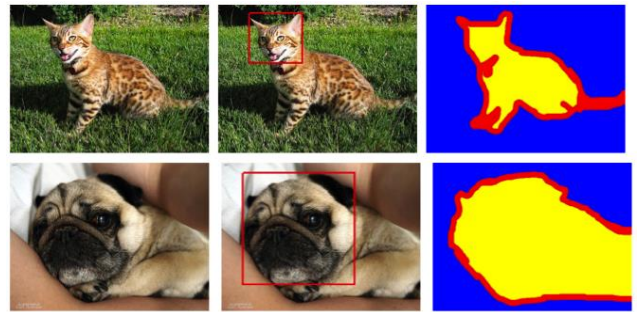


Figure 1. Annotations in the Oxford-IIIT Pet data examples. From left to right: pet image, head bounding box, and tri-map segmentation (blue: background, red: ambiguous region, yellow: foreground region).

2.2. Classification

The first goal of this project is the broad evaluation of pre-trained models provided by TensorFlow framework, which means that users can easily import and use them, as the classification models. In order to enhance the performance of models, non-constant learning rate and data

augmentation method are implemented. The important evaluation standard is the accuracy for validation set, but the time-cost is also very essential to be considered.

2.3. Localization

The second contribution of this project is to localize the object. Thanks to the dataset that has a large annotated collection of images, the ground truth of representing the head region (or, face region) can be useful data to the localization. From the previous classification results, the top accuracy models are used after the class activation map (CAM)[5] to show that the number of class in the classification models will be appropriate with the localization.

2.4. Segmentation

Lastly, the image segmentation is also implemented with *VGG-16* and *VGG-19* pretrained models. *U-Net* model is the basic structure for the image segmentation, thus the vanilla *U-Net* like *VGG-16* is tested as the preliminary test. From the results, the transfer learning using pretrained models shows that it is the most power method to enhance the performance of image classification, localization, and segmentation processes.

2.5. Evaluation protocol

Three tasks are defined: (i) classification (a 37-class problem), (ii) localization, and (iii) segmentation. In classification task, the performance is measured as the classification accuracy and the average step-time, which is to index for the time-cost. Both metrics is very familiar with people who studies the machine learning. In localization and segmentation tasks, intersection of union (IoU) is used to evaluate the models. IoU is an evaluation metric used to measure the accuracy of an object detector on a particular dataset[6]. We often see this evaluation metric used in object detect challenges such as the popular PASCAL VOC challenge[7]. The ground truth bounding box is here the red in the middle of Figure 1, and predicted bounding box will be the blue rectangular one. Computing IoU can therefore be determined like in Figure 2. In the numerator we compute the area of overlap between the predicted bounding box and the ground truth bounding box. The denominator is the area of union, or more simply, the area encompasses by both the predicted bounding box and the ground truth bounding box. Diving area of overlap by the area of union yields our final score, the intersection over union (IoU). As we can see, predicted bounding boxes that heavily overlap with the ground truth bounding boxes have higher scores than those with less overlap. This makes IoU an excellent metric for evaluating custom object detectors, to localization and segmentation here.

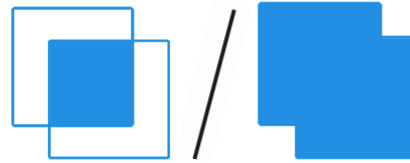


Figure 2. Computing the Intersection over Union (IoU). Diving the area of overlap between the bounding boxes by the area of union.

3. Experiment

3.1. Classification

Before jumping into the SOTA models of transfer learning, the preliminary testing is done first to check the performance or feasibilities of models, dataset and hyperparameters. The best condition is applied with various pre-trained models imported from TensorFlow library.

3.1.1 Preliminary test

The first step of preliminary test is that, without any data augmentation and hyperparameter tuning, the training set is just dumped into the Vanilla CNN in Figure 3. The accuracy performance is very poor of around 27% and it suffers from the severe overfitting, shown in Figure 4 (Gray). To address the overfitting issue and improve the performance, the Vanilla CNN structure has batch normalization layer after each convolutional layer, and the exponential decay learning rate among learning rate schedule methods is implemented[8]. Learning rate schedules seek to adjust the learning rate during training by reducing the learning rate according to a pre-defined schedule. We can implement this by defining the exponential decay function like in Figure 5 and pass it to the *LearningRateScheduler* in *keras*, then the learning rate during training shows the behavior of Figure 6. The performance with those variations is improved (~31%), but it is still lower than expectation. In order to increase the accuracy, “CutMix” data augmentation technique is implemented (Figure 7). Cut mix method is one of the regional dropout strategies that have been proposed to enhance the performance of CNN classifiers. Patches are cut and pasted among training images where the ground truth labels are also mixed proportionally to the area of the patches. By making efficient use of training pixels and retaining the regularization effect of regional dropout, CutMix consistently outperforms the SOTA augmentation strategies on CIFAR and ImageNet classification tasks[9]. Sadly, the worse performance is recorded (~29%), which means that Vanilla CNN structure is impossible to have the good performance anymore. Therefore, the model is changed to one of the pretrained models, *MobileNetV2*, and it shows the significantly improved result (~93.76%) and diminishes overfitting, only with exponential learning rate.

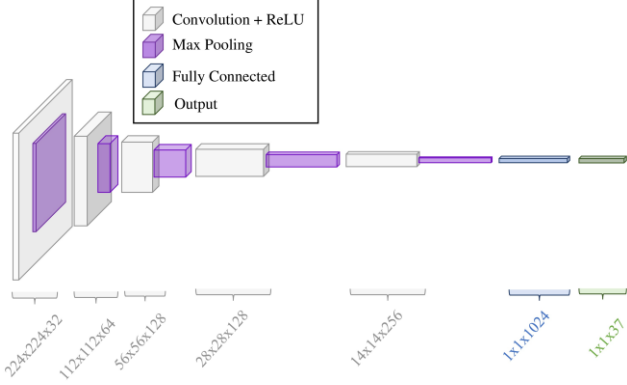


Figure 3. Vanilla CNN structure. 5 Convolutional layers with ReLU activation and Max Pooling layers with 1 Fully Connected layer and 1 Output layer.

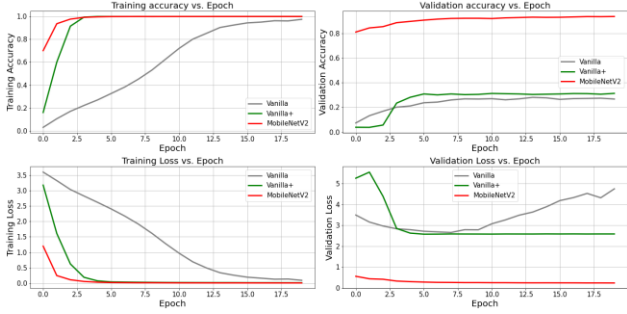


Figure 4. Training/Validation accuracy and loss vs. Epoch plot. Gray: Vanilla CNN, Green Vanilla CNN with batch normalization, and Red: MobileNetV2 structure, respectively.

```
def lr_schedule_fn(epoch):
    if epoch < RAMPUP_EPOCH:
        lr = (LR_MAX - LR_MIN) / RAMPUP_EPOCH * epoch + LR_INIT
    else:
        lr = (LR_MAX - LR_MIN) * EXP_DECAY**(epoch - RAMPUP_EPOCH)
    return lr
```

Figure 5. Code for exponential learning rate function.

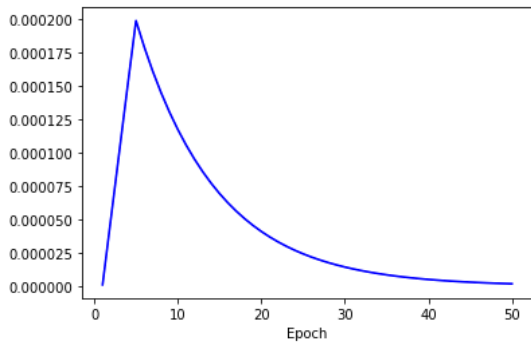


Figure 6. Plot for exponential decay learning rate in 50 epochs. The initial learning rate is 0.000001, the max learning rate is 0.0002, ramp-up epoch is 4, exp_decay constant is 0.9.

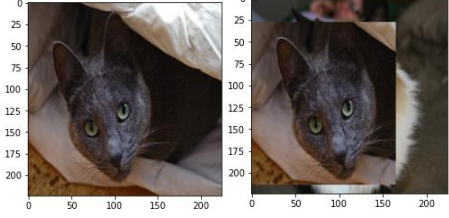


Figure 7. CutMixed data augmentation. Left: original image, Right: CutMixed image

3.1.2 Transfer learning to Classification

From the preliminary tests, we can say that (i) pretrained model, (ii) exponential decay learning rate, (iii) CutMix data augmentation will be helpful to the classification model. The contribution of this section is to introduce the 10 pre-trained models are implemented which are imported from TensorFlow. The only thing to be modified is the bottle-neck layer replaced with one Global Average Pooling layer (GAP), one Dense layer with the batch normalization and ReLU activation, and one Dense layer with softmax activation to output (Figure 8).

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Model)	(None, 7, 7, 1280)	2257984
global_average_pooling2d (G1	(None, 1280)	0
dense (Dense)	(None, 256)	327936
batch_normalization (BatchNo	(None, 256)	1024
re_lu (ReLU)	(None, 256)	0
dense_1 (Dense)	(None, 37)	9509
Total params: 2,596,453		
Trainable params: 2,561,829		
Non-trainable params: 34,624		

Figure 8. Bottle-neck layer with MobileNetV2. All bottle-neck layer structure is the same, but pretrained models.

From VGG-16 and VGG-19, to sophisticated models like InceptionResNetV2, results are shown in Table 1 and we can observe the DenseNet121 shows the best accuracy of 94.51%, but the best cost-efficient model is MobileNetV2 which has the accuracy of 93.28% and the average process time/step of 29 sec.

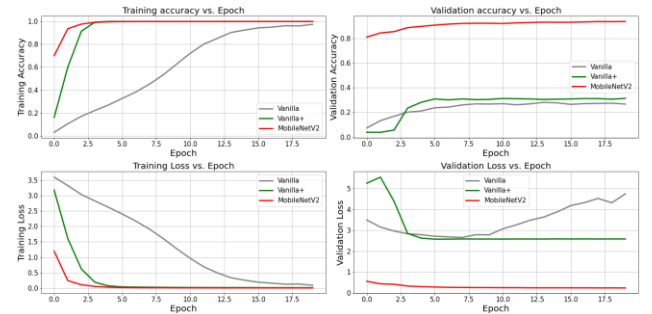


Figure 9. Plot for transfer learning models to classification. (Top-left: Training Acc., Top-right: Validation Acc., Bottom-left: Training loss, Bottom-right: Validation loss vs. epoch)

Table 1. Performance results of 10 pre-trained weight transfer learning models. 50 epochs. *T is the average time per training step. (*MbNV2*: MobileNetV2, *IncepV3*: InceptionV3, *Xcep*: Xception, *DN121*: DenseNet121, *RN50*: ResNet50, *R152V2*: ResNet152V2, *NASMo*: NASNetMobile, *InRNV2*: InceptionResNetV2)

	Models									
	<i>VGG-16</i>	<i>VGG-19</i>	<i>MbNV2</i>	<i>IncpV3</i>	<i>Xcep</i>	<i>DN121</i>	<i>RN50</i>	<i>R152V2</i>	<i>NASMo</i>	<i>InRNV2</i>
Acc. [%]	89.85	91.22	93.28	93.62	<u>94.17</u>	<u>94.51</u>	92.11	91.02	92.87	93.48
*T [s]	50	58	<u>29</u>	<u>30</u>	68	42	38	80	54	68

Table 2. Preliminary test of Localization. 40 epochs. (*Red*: Ground truth bounding box, *Blue*: Predicted bounding box) (*Xcep*: Xception, *MbNV2*: MobileNetV2, *InRNV2*: InceptionResNetV2, *IncpV3*: InceptionV3, *DN121*: DenseNet121)

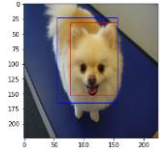
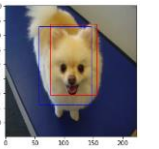
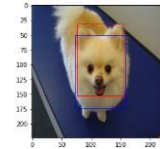
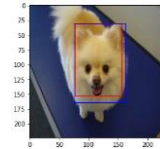
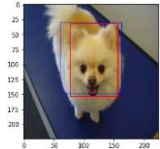
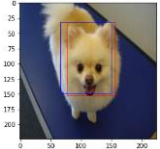
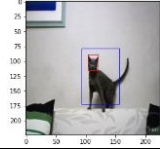
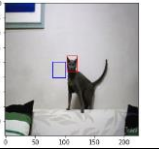
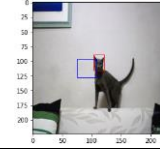
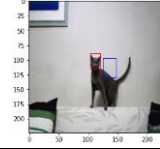
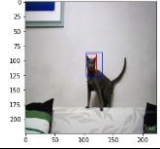
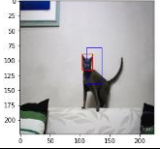
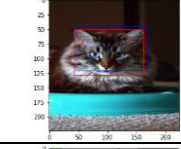
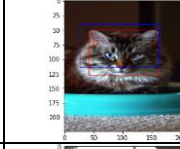
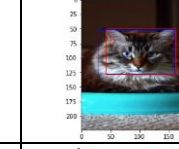
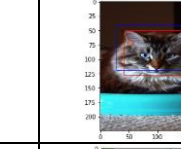
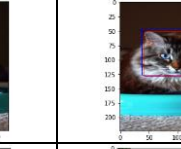
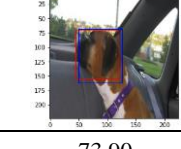
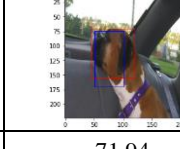
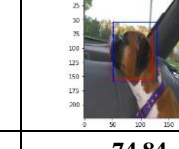
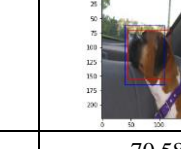
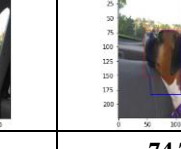
	<i>Vanilla CNN</i>	<i>Xcep</i>	<i>MbNV2</i>	<i>InRNV2</i>	<i>IncpV3</i>	<i>DN121</i>
Sample #1						
Sample #2						
IoU [%]	49.65	62.05	65.24	66.85	69.32	70.69

Table 3. Localization results with multi-task learning and transfer learning. (*Red*: Ground truth bounding box, *Blue*: Predicted bounding box) (*Xcep*: Xception, *MbNV2*: MobileNetV2, *InRNV2*: InceptionResNetV2, *IncpV3*: InceptionV3, *DN121*: DenseNet121)

	<i>Xcep</i>	<i>MbNV2</i>	<i>InRNV2</i>	<i>IncpV3</i>	<i>DN121</i>
Sample #1					
Sample #2					
IoU [%]	73.90	71.94	<u>74.84</u>	70.58	<u>74.78</u>
*T [s]	35	<u>15</u>	34	<u>17</u>	22

3.2. Localization

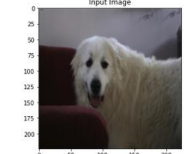
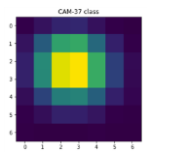
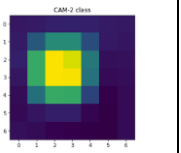
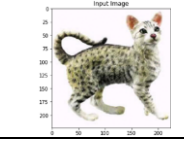
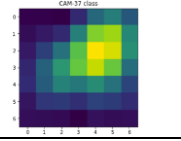
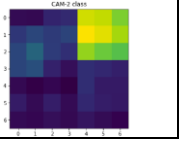
This section evaluates the models on the task of localization the head region in the image given their 3685 annotated information. This is done by 2 steps. At the preliminary test, Vanilla CNN and top 5 accuracy pre-trained models are checked as the baseline. In order to improve the performance, class activation maps for 2-class and 37-class classification model are investigated. Then, the confirmed condition and its pre-trained classification models are concatenated with the localization model to upgrade the performance of localization.

3.2.1 Preliminary test

The purpose of this step is to see the baseline performance and unveil the latent problem of models. 3000 annotated training images are input to the Vanilla CNN, *MobileNetV2*, *Xception*, *InceptionV3*, *InceptionResNetV2*, and *DenseNet121*, which they are selected from the based on the performance except the Vanilla CNN. As we can see in Table 2, the pre-trained models show the better performance than the Vanilla CNN. *DenseNet121*(*DN121*) shows the best IoU of around 70%, but the predicted bounding box is still a little away from the ground truth bounding box.

We can assume that the usual classification model is concentrated on the entire object in the image, which means that it may not be helpful to implement the localization. To verify this, the class activation map (CAM) is implemented between 37-class (breeds) and 2-class (species; dog/cat). A class activation map for a particular category indicates the discriminative image regions used by the CNN to identify that category[5]. In this annotated dataset, we can say that the head ROI is the most important information for the localization, and it is possible to confirm in Table 4 that 2-class classification model (i.e., *DenseNet121*) based on species has been more concentrated on the ‘local’ region, which is the head.

Table 4. Class Activation Mapping (CAM) for the classification of *DenseNet121* model having 37-class vs. 2-class.

Input image	37-class (breed)	2-class(species)
		
		

3.2.2 Multi-task learning via Transfer learning

Based on the previous results, the multi-task learning is implemented to increase the IoU performance. Multi-task learning is method that by sharing representations between related tasks, we can enable our model to generalize better on our original task[10][11]. In Figure 10, the multi-task learning structure is shown. The pre-trained model is connected to the global average pooling layer (GAP), and 2 tasks layers are parallel-connected to it as 2-branch, and then they are concatenated. As we can see in Table 3, all IoU is enhanced, and *DenseNet121* is still showing good performance (74.78%) among them, and *MobileNetV2* is also still the best cost-efficient model (IoU of 71.94% and average time per step of 15 seconds).



Figure 10. Multi-task learning structure with transfer learning.

3.3. Segmentation

This section evaluates the object segmentation which is the last goal of the project. Usually, the object segmentation uses the *U-Net* structure[12] (Figure 11), which is made of the encoder and decoder sections. The *U-Net* includes a contracting path (the encoder) with several layers of

convolution and pooling for down-sampling. The second half of the network includes an expansion path (the decoder) that uses up-sampling and convolution layers sequentially to generate an output with a similar size as the input image[13]. Fully convolutional networks (FCN) are extensively used for semantic segmentation tasks. The first FCN architecture that I use in this work is based on the FCN-8s net that uses the *VGG-16* layer net[14][15]. The *VGG-16* net is converted into an FCN by decapitating the final classification layer and converting fully connected layers into convolution. Deconvolution layers are then used to upsample the coarse outputs to pixel-dense outputs. Skip connections are used to merge output from previous pooling layers in the network which was shown to improve the segmentation quality[14].

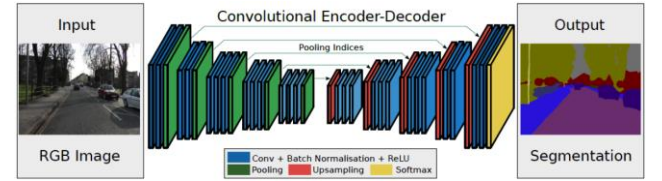


Figure 11. *U-Net* structure of encoder-decoder architecture from *SegNet*[16][17].

The preliminary test in the segmentation thus *U-Net* like *VGG-16* model training through the random initialization to set the baseline. And then, *VGG-16* and *VGG-19* are used as the transfer learning methods.

3.3.1 Preliminary test

The pristine *U-Net* like *VGG-16* structure is shown in Figure 12. It has the IoU of 83.39% during 20 epochs. This is our baseline to the segmentation evaluation. As we can see in Table 5, there are mis-segmented region at the corner or boundary between foreground and background.

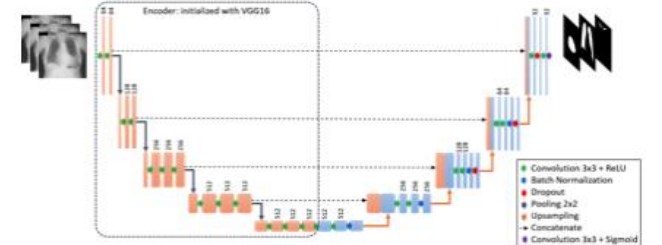
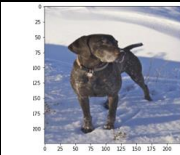
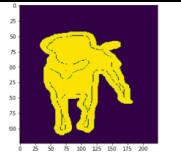
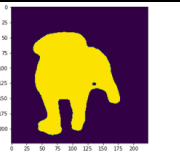
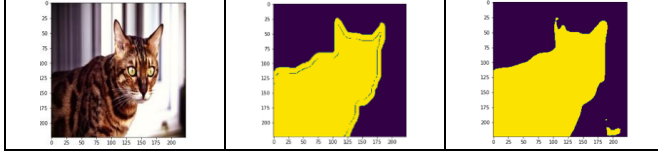


Figure 12. The used *U-Net* architecture with a *VGG-16* based encoder and their corresponding decoder[13].

Table 5. Segmented images of *U-Net* based on *VGG-16* encoder and their corresponding decoder structure.

Input image	Ground Truth	Segmented
		



3.3.2 Transfer learning with VGG-16 and VGG-19

This transfer learning experiment is simpler than previous classification and localization. This is because the segmentation what I target is to binary segmentation, which means deleting the layer between foreground and background, to see the transfer learning's positive effect on the image processing. Also, the encoder and decoder section are getting to be more complex once adopting branch-like structures (i.e., *InceptionV3*, *Xception*). The encoder codes of *VGG-16*[18] and *VGG-19*[19] are from the official github of TensorFlow. The weights of *VGG-16* or *VGG-19*[20] is added after the encoder block. The results, which is shown in Table 6, are not surprising, because the transfer learning keep showing its ability for enhancing the model's performance. Both U-Net structures based on *VGG-16* or *VGG-19* is showing the IoU of over 90% during 20 epochs without mis-segmented regions. The segmentation is not perfect at the corner of sharp region, but we can verify that U-Net with pre-trained model's weight successfully helps to improve the segmentation model's performance.

Table 6. Segmentation of U-Net like VGG-16 and VGG-19.

Ground Truth	U-Net like VGG-16	U-Net like VGG-19
IoU [%]	90.02	90.09

4. Conclusion

This project has introduced the Oxford-IIIT Pet dataset for the fine-grained categorization problem of identifying the family and breed of dogs and cats. Three different but related tasks and corresponding baseline models and best conditions have been proposed and investigated to obtain encouraging classification, localization, and segmentation results on the dataset. In conclusion, the transfer learning which means using pre-trained models of ImageNet is highly recommended to do the image recognition tasks. Also, the schedule learning rate method is showing better

performance than the constant learning rate, as alleviating the overfitting. In addition to this, data augmentation such as CutMix is helpful to enhance the classification accuracy.

For the future work, it can be a good candidate that implementing adaptive learning rate[8] to the classification models to upgrade the performance. In the data augmentation section, Mixup[21] or Cutout[22] can be tested even though they did not overwhelm the CutMix. The deeper or more sophisticated pre-trained models should be evaluated in the classification, localization, and especially, segmentation.

References

- [1] <http://www.robots.ox.ac.uk/~vgg/data/pets/>
- [2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In Proc. CVPR, 2009
- [3] Y. Chai, V. Lempitsky, and A. Zisserman. Bicos: A bi-level co-segmentation method for image classification. In Proc. ICCV, 2011
- [4] Wei X S, Wu J, Cui Q. *Deep learning for fine-grained image analysis: A survey*. arXiv preprint arXiv:1907.03069, 2019
- [5] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba, Learning Deep Features for Discriminative Localization, CVPR, p. 2921-2929, 2016
- [6] <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>
- [7] <http://host.robots.ox.ac.uk/pascal/VOC/>
- [8] <https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1>
- [9] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*, arXiv:1905.04899v2, 2019
- [10] Girshick, R, *Fast R-CNN*, In Proc. ICCV, pages 1440–1448, 2015
- [11] Sebastian Ruder, *An Overview of Multi-Task Learning in Deep Neural Networks*, arXiv:1706.05098v1, 2017
- [12] Ronneberger, O., Fischer, P. and Brox, T., *U-net: Convolutional networks for biomedical image segmentation*, International Conference on Medical image computing and computer-assisted intervention, pp. 234-241, Springer, Cham, 2015
- [13] <https://www.groundai.com/project/improving-the-segmentation-of-anatomical-structures-in-chest-radiographs-using-u-net-with-an-imagenet-pre-trained-encoder/1#bib.bib8>
- [14] Long, J., Shelhamer, E., and Darrell, T., *Fully convolutional networks for semantic segmentation*. Proc. CVPR, pp. 3431-3440, 2015
- [15] Simonyan, K. and Zisserman, A., *Very deep convolutional networks for large-scale image recognition*. arXiv preprint arXiv:1409.1556, 2014
- [16] <http://mi.eng.cam.ac.uk/projects/segnet/demo.php>
- [17] <https://towardsdatascience.com/review-segnet-semantic-segmentation-e66f2e30fb96>
- [18] <https://github.com/tensorflow/tensorflow/blob/v2.2.0/tensorflow/python/keras/applications/vgg16.py#L39-L216>
- [19] <https://github.com/tensorflow/tensorflow/blob/v2.2.0/tensorflow/python/keras/applications/vgg19.py#L44-L226>

- [20] <https://github.com/fchollet/deep-learning-models/releases>
- [21] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz, *Mixup: Beyond empirical risk minimization*, arXiv preprint arXiv:1710.09412, 2017
- [22] Terrance DeVries and Graham W Taylor, *Improved regularization of convolutional neural networks with cutout*, arXiv preprint arXiv:1708.04552, 2017