CSC413 Group45 Final Project

Vanessa Yu
Department of Computer Science
University of Toronto
vanessahuiting.yu@mail.utoronto.ca

Zuoyu Wang
Department of Computer Science
University of Toronto
zuoyu.wang@mail.utoronto.ca

Yixuan Chen
Department of Computer Science
University of Toronto
yixuanch.chen@mail.utoronto.ca

Abstract

Aiming to study two limitations of Convolutional neural networks (CNNs), we replicate and analyze the results from two different research papers. MobileNets are a class of computationally efficient models, achieved by replacing the standard convolution with two steps: depth-wise convolution and point-wise convolution. In addition, two new hyperparameters (width multiplier and resolution multiplier) are introduced to balance efficiency and accuracy. The second technique we look at is Deformable CNNs, which try to strengthen the CNNs' ability to model transformations. Deformable CNNs add an additional step before the standard convolution and pooling: modify the sampling locations by adding augmented offsets, where the offsets are learned from the input feature map.

1 Introduction

After the success of LeNet [8] and AlexNet [7], convolutional networks (CNNs) have been a popular approach when dealing with image-processing tasks. With the wider usage of CNNs, we hope to see CNNs' usage on vision applications in mobile and embedded devices and make CNNs more robust to a variety of complicated usage. Even though CNNs are indeed powerful tools, the standard convolutional networks introduced in LeNet and AlexNet do have limitations. In this project, we study two of these limitations: the relatively large computational cost and the limited transformation-invariant ability of CNNs. In particular, we replicate and examine two specific approaches presented by two research papers, each trying to tackle one of these two problems. MobileNet [5] aims to solve the computation problem with depth-wise separable convolutions and two new hyperparameters. Deformable CNN [2] tries to strengthen CNNs' ability to model transformations by replacing the standard convolution and pooling.

2 Related work

2.1 MobileNet

MobileNet [5] is the first method our project aims to re-implement and perform analysis. MobileNet, first built in 2017, was a convolutional neural network architecture that was designed based on depth-wise separable convolutions. Convolutional neural networks, such as AlexNet [7] and ResNets[4], have grown to become the most popular algorithms in computer vision over the past decade. Depth-wise separable convolutions were initially introduced by Laurent Sifre in his Ph.D. thesis [10] in 2014. It was later utilized in the initial layers of Inception models [6, 12] for improving efficiency.

Preprint. Under review.

2.2 Deformable Convolution

Standard convolution involves convolving over a regular grid of locations on the input feature map. Although pooling layers or RoI operations can be used to accommodate spatial transformations, these methods typically reduce resolution and have fixed receptive fields. Deformable convolution networks [2], published in 2017, can learn 2D offsets without supervision to generalize various spatial variations. Experiments in [2] have shown that the receptive field in deformable convolution can adaptively adjust to an object's scale and shape, resulting in improved accuracy compared to dilated convolutions. Previous methods, such as affine transformations which use global parametric transformations or those hand-crafted algorithms, are not suitable for learning large and unknown transformations. In contrast, offset learning in deformable convolution can be trained end-to-end through backpropagation, allowing it to learn dynamic sampling locations based on image locations. Despite adding only a small overhead in model parameters and computation, deformable convolutional networks have been shown to significantly improve performance in object detection and semantic segmentation.

3 Method / algorithm

3.1 MobileNet and Depth-wise separable Convolution

We first want to re-implement MobileNet based on the structure proposed in the paper [5]. The key algorithm for MobileNet is to split a standard convolution into two steps: a depth-wise convolution on each input channel and a 1×1 point-wise convolution to sum up outputs from the previous step. Assume the inputs have a dimension of $K \times K \times M$, where $K \times K$ is Height $\times$ Width, and M is the number of channels. The depth-wise convolution layer will have M number of $K \times K \times 1$ filters, each one performing convolution on each input channel. Therefore, the output dimension for the depth-wise convolution layer is still $K \times K \times M$. Then, $1 \times 1 \times M \times N$ point-wise convolution is applied to convert M to N new out features. Therefore it helps make reasonable trade-offs between model efficiency and accuracy.

3.2 Deformable Convolution

In the original paper [2], deformable convolution replaced its conventional counterparts in the backbone CNN architectures ResNet-101 [4] and a modified version of Inception-ResNet [11], and the model was trained for both semantic segmentation and object detection tasks.
Due to computation regulation, we will implement a smaller backbone architecture that consists of four usual convolution layers and three deformable convolution layers(DeformConv). The key algorithm for DeformConv is described below. A structure of DeformConv is attached in Appendix DeformConv Structure.

In the proposed module mentioned in [2], a deformable convolution layer first utilizes a usual convolution layer to learn the 2D offsets of dimension $B \times 2N \times H \times W$, where B represents the batch size, N represents the kernel size multiplied by the kernel size and H and W represent the height and width of the feature map, respectively. By adding these learned offsets to the positions of the input feature map pixels, the output feature map pixel locations are obtained. Since the additional offset is usually fractional, bilinear interpolation is used to obtain the final pixel value. The resulting feature map has a size of $B \times C \times H \times W \times N$ (C represents the channel size), with each entry representing a pixel value in the input feature map. By reshaping it to $B \times C \times (H \times K) \times (W \times K)$, ($K = 3$ represents the kernel size), and convolving it with a kernel of size $K \times K$, the output feature map of size $B \times C \times H \times W$ is obtained, which has the same dimension as the input feature map.

4 Experiments

4.1 Dataset

Due to computation constraints, we analyze the performance of depth-wise separable convolution and deformable convolution on a smaller task of dog breed classification instead of the entire ImageNet. Two dog breed datasets, Stanford Dogs Dataset and Dog Breed Images, are used for our project. We combine the images of the 7 dog breeds that appear in both datasets: Bernese mountain dog, Border collie, Chihuahua, Golden retriever, Labrador, Pug, and Siberian husky, into one dataset for our purpose (Task idea from CSC420 assignment 2 [13]). After separating the dataset into training, validation, and test set, we perform random deformation on the validation and test set to amplify the possible difference in performances of deformable convolutions and usual convolutions.

4.2 Network Structure

In total, we introduce three network structures, each specified below.

4.2.1 DogBreedClassificationCNN

We first construct a basic neuron network structure by simplifying the MobileNet (without depth-wise separable convolution layers), namely DogBreedClassificationCNN. The simplified structure consists of seven convolution layers, an average pool, and a fully connected layer to convert the feature into seven classes. Each convolution layer uses 3×3 kernel size and one padding to keep dimensions. For odd(1,3,5,7) convolution layers, we apply two strides to apply as subsampling.

4.2.2 DogBreedClassificationDSC

DogBreedClassificationDSC adds depth-wise separable convolutions into the model. To do so, we replace the middle six convolution layers with depth-wise separable layers.

4.2.3 DogBreedClassificationDeformableCNN

Again we are trying to incorporate deformable convolution into the Simplified CNN. According to the ablation study in the paper[2], the accuracy improvement saturates when using 3 deformable layers using DeepLab with ResNet-101 backbone. Thus, we replace 3 convolution layers in simplified CNN. We apply DeformConv at every other layer.

Detailed structures of the three networks are shown in Appendix.

4.3 Results & Discussions

The tables below show the result of the three different CNN networks. We trained each network with 3 stages: a combination of 3 numbers of epochs (separated by "/") with 3 decreasing max learning rates (also separated by "/"). By setting the hyperparameters as grad_clip = 0.1, weight_decay = 1e-4, max_lr = 0.01/0.001/0.0001, we get the following results (we've also included some accuracy curves in the appendix for reference):

4.3.1 Evaluation of DogBreedClassificationCNN

DogBreedClassificationCNN Results			
epoch	20/14/6	50/7/3	50/14/6
train_loss	0.9680	0.5488	0.5086
train_acc	0.7050	0.8925	0.9050
val_loss	1.3285	1.4217	1.2668
val_acc	0.5333	0.5333	0.5646
test_loss	1.3299	1.2469	1.2487
test_acc	0.5101	0.5901	0.5923

4.3.2 Evaluation of DogBreedClassificationDSC

DogBreedClassificationDSC Results			
epoch	20/14/6	50/7/3	50/14/6
train_loss	1.1875	0.5744	0.6265
train_acc	0.6133	0.8508	0.8592
val_loss	1.4953	1.1783	1.1861
val_acc	0.4542	0.5708	0.5521
test_loss	1.3417	1.3159	1.3388
test_acc	0.5148	0.5622	0.5471

Compared with DogBreedClassificationCNN, introducing depth-wise separable convolutions help significantly reduce the model size, while remaining relatively high accuracy. From the network structures(Appendix), DogBreedClassificationDSC reduced the number of trainable parameters by 87.2% (585383 vs. 74663). While the best test accuracy remains around 56%, which does not imply a notable decrease.

4.3.3 Evaluation of DogBreedClassificationDeformableCNN

DogBreedClassificationDeformableCNN Results			
epoch	20/14/6	50/7/3	50/14/6
train_loss	0.8231	0.2878	0.2178
train_acc	0.7700	0.9467	0.9883
val_loss	1.1414	1.2395	1.3552
val_acc	0.6562	0.5583	0.6250
test_loss	1.4600	1.5843	1.3622
test_acc	0.4649	0.5044	0.5682

Looking at the performance of DogBreedClassificationCNN and DogBreedClassificationDeformableCNN on the training and the validation datasets, we find that DogBreedClassificationDeformableCNN achieves a higher training accuracy and validation accuracy than DogBreedClassificationCNN for all three different epoch number setups. This implies that, with the same number of training steps, the model with the deformable convolution layer can pick up more information from the training set and possibly has a better generalization than the model without it. Such an effect can be further tested with a larger dataset. However, we can't test it for this project due to computation and time constraints. We also notice that for both neural networks, even though training accuracy increases as epoch number increases, the validation accuracy is stabilizing (not improving or decreasing). This might be because both models are overfitting due to data scarcity.

By comparing the two models' performance on the test set, we can see that the performance of both models is roughly equivalent, with DogBreedClassificationDeformableCNN performing slightly worse. On the other hand, as shown above, the DogBreedClassificationDeformableCNN has a higher validation accuracy. Possible reasons could be that our dataset is too small such that misclassification of only a few images will cause a large fluctuation in the accuracy.

5 Conclusions

Both depth-wise separable convolution and deformable convolution modify a standard convolution operation to enhance the model efficiency and ability of generalization respectively. In this report, we design a simplified version of MobileNet with standard convolution layers and compare it with two variants: one with depth-wise separable convolution layers, and the other with deformable convolution layers. Experiments have shown that depth-wise separable convolution is able to make a smaller model without losing too much accuracy. Also, deformable convolution can pick up more useful information from the image by learning different offsets and adding them to pixel locations. This adds flexibility to the model during convolution.

References

[1] Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. Advances in neural information processing systems, 29.

[2] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., & Wei, Y. (2017). Deformable Convolutional Networks. 2017 IEEE International Conference on Computer Vision (ICCV), 764-773.

[3] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2009). The pascal visual object classes (voc) challenge. International journal of computer vision, 88, 303-308.

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

[5] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv, abs/1704.04861.

[6] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. ArXiv, abs/1502.03167.

[7] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 6 (June 2017), 84–90. https://doi.org/10.1145/3065386

[8] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[9] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

[10] Sifre, L. Rigid-motion scattering for image classification, 2014. Ph.D. thesis.

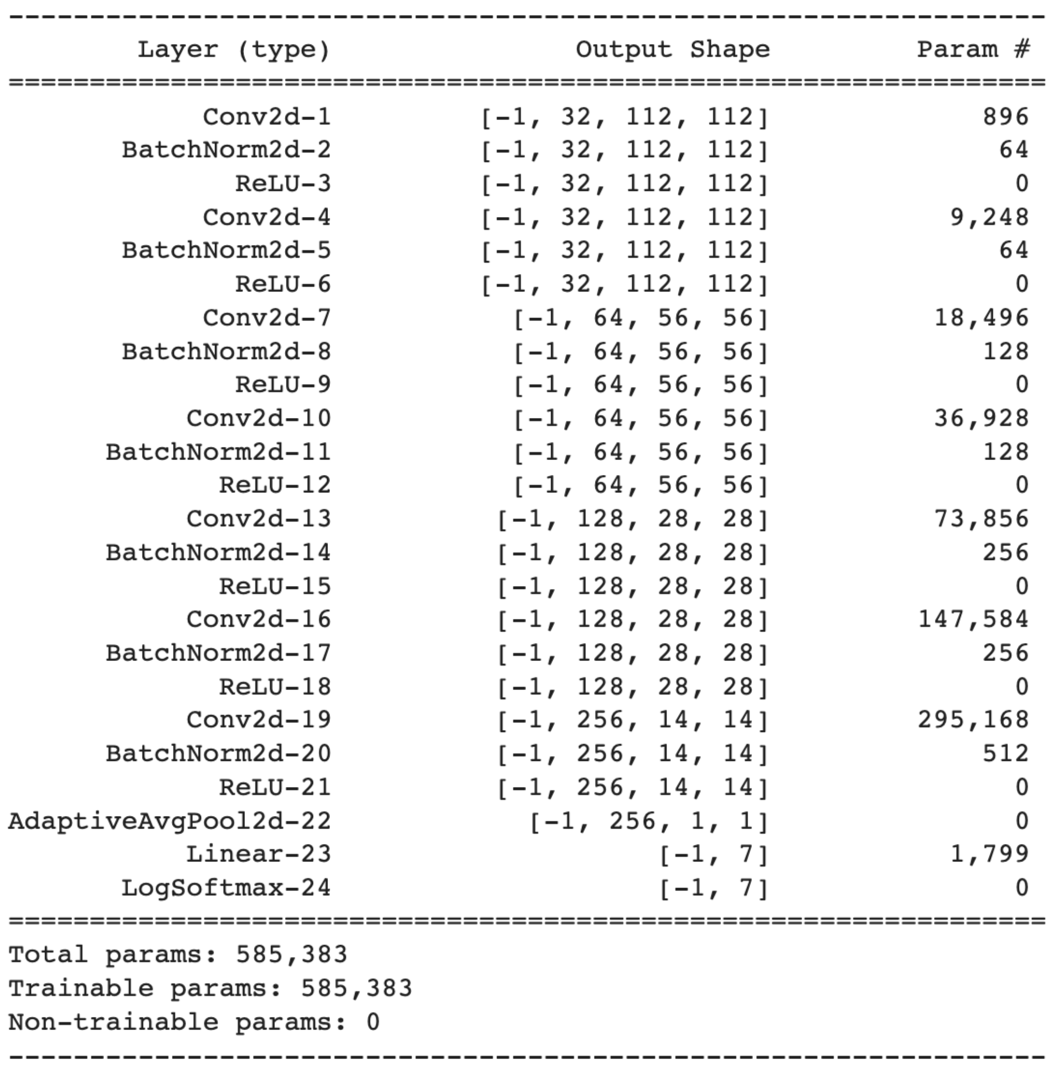
[11] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 31, No. 1).

[12] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1-9.

[13] Taati, B. (2022) CSC420 assignment 2 [Class handout]. University of Toronto, CSC420.

Appendix

DogBreedClassificationCNN Structure



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 112, 112]             896
       BatchNorm2d-2         [-1, 32, 112, 112]              64
              ReLU-3         [-1, 32, 112, 112]               0
            Conv2d-4         [-1, 32, 112, 112]           9,248
       BatchNorm2d-5         [-1, 32, 112, 112]              64
              ReLU-6         [-1, 32, 112, 112]               0
            Conv2d-7           [-1, 64, 56, 56]          18,496
       BatchNorm2d-8           [-1, 64, 56, 56]             128
              ReLU-9           [-1, 64, 56, 56]               0
           Conv2d-10           [-1, 64, 56, 56]          36,928
      BatchNorm2d-11           [-1, 64, 56, 56]             128
             ReLU-12           [-1, 64, 56, 56]               0
           Conv2d-13          [-1, 128, 28, 28]          73,856
      BatchNorm2d-14          [-1, 128, 28, 28]             256
             ReLU-15          [-1, 128, 28, 28]               0
           Conv2d-16          [-1, 128, 28, 28]         147,584
      BatchNorm2d-17          [-1, 128, 28, 28]             256
             ReLU-18          [-1, 128, 28, 28]               0
           Conv2d-19          [-1, 256, 14, 14]         295,168
      BatchNorm2d-20          [-1, 256, 14, 14]             512
             ReLU-21          [-1, 256, 14, 14]               0
AdaptiveAvgPool2d-22            [-1, 256, 1, 1]               0
           Linear-23                    [-1, 7]           1,799
       LogSoftmax-24                    [-1, 7]               0
================================================================
Total params: 585,383
Trainable params: 585,383
Non-trainable params: 0
----------------------------------------------------------------
```

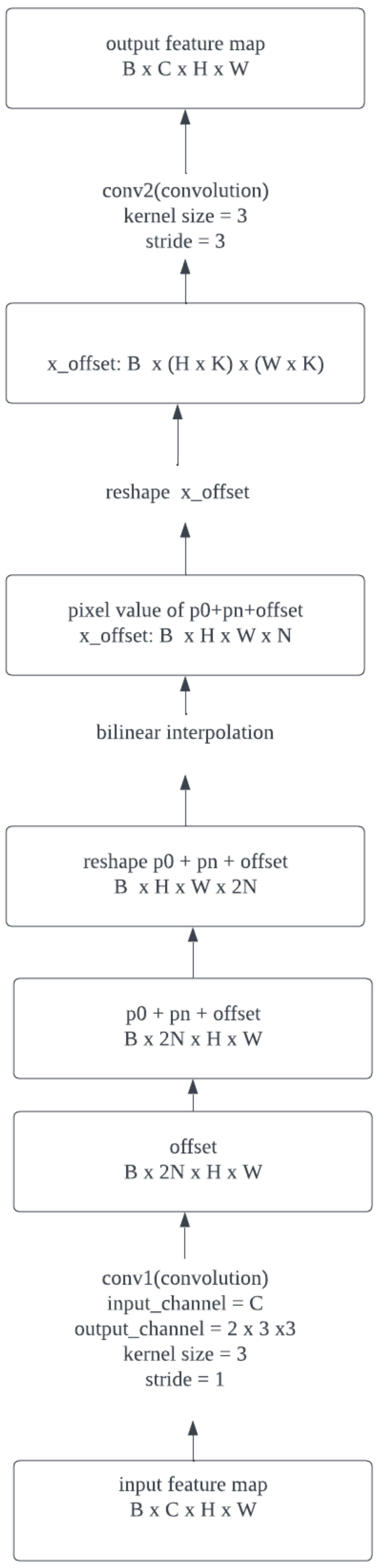
DogBreedClassificationDSC Structure

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 112, 112]             896
       BatchNorm2d-2         [-1, 32, 112, 112]              64
              ReLU-3         [-1, 32, 112, 112]               0
            Conv2d-4         [-1, 32, 112, 112]             320
       BatchNorm2d-5         [-1, 32, 112, 112]              64
              ReLU-6         [-1, 32, 112, 112]               0
     DepthWiseConv-7         [-1, 32, 112, 112]               0
            Conv2d-8         [-1, 32, 112, 112]           1,056
       BatchNorm2d-9         [-1, 32, 112, 112]              64
             ReLU-10         [-1, 32, 112, 112]               0
    PointWiseConv-11         [-1, 32, 112, 112]               0
     DepthSepConv-12         [-1, 32, 112, 112]               0
           Conv2d-13           [-1, 32, 56, 56]             320
      BatchNorm2d-14           [-1, 32, 56, 56]              64
             ReLU-15           [-1, 32, 56, 56]               0
    DepthWiseConv-16           [-1, 32, 56, 56]               0
           Conv2d-17           [-1, 64, 56, 56]           2,112
      BatchNorm2d-18           [-1, 64, 56, 56]             128
             ReLU-19           [-1, 64, 56, 56]               0
    PointWiseConv-20           [-1, 64, 56, 56]               0
     DepthSepConv-21           [-1, 64, 56, 56]               0
           Conv2d-22           [-1, 64, 56, 56]             640
      BatchNorm2d-23           [-1, 64, 56, 56]             128
             ReLU-24           [-1, 64, 56, 56]               0
    DepthWiseConv-25           [-1, 64, 56, 56]               0
           Conv2d-26           [-1, 64, 56, 56]           4,160
      BatchNorm2d-27           [-1, 64, 56, 56]             128
             ReLU-28           [-1, 64, 56, 56]               0
    PointWiseConv-29           [-1, 64, 56, 56]               0
     DepthSepConv-30           [-1, 64, 56, 56]               0
           Conv2d-31           [-1, 64, 28, 28]             640
      BatchNorm2d-32           [-1, 64, 28, 28]             128
             ReLU-33           [-1, 64, 28, 28]               0
    DepthWiseConv-34           [-1, 64, 28, 28]               0
           Conv2d-35          [-1, 128, 28, 28]           8,320
      BatchNorm2d-36          [-1, 128, 28, 28]             256
             ReLU-37          [-1, 128, 28, 28]               0
    PointWiseConv-38          [-1, 128, 28, 28]               0
     DepthSepConv-39          [-1, 128, 28, 28]               0
           Conv2d-40          [-1, 128, 28, 28]           1,280
      BatchNorm2d-41          [-1, 128, 28, 28]             256
             ReLU-42          [-1, 128, 28, 28]               0
    DepthWiseConv-43          [-1, 128, 28, 28]               0
           Conv2d-44          [-1, 128, 28, 28]          16,512
      BatchNorm2d-45          [-1, 128, 28, 28]             256
             ReLU-46          [-1, 128, 28, 28]               0
    PointWiseConv-47          [-1, 128, 28, 28]               0
     DepthSepConv-48          [-1, 128, 28, 28]               0
           Conv2d-49          [-1, 128, 14, 14]           1,280
      BatchNorm2d-50          [-1, 128, 14, 14]             256
             ReLU-51          [-1, 128, 14, 14]               0
    DepthWiseConv-52          [-1, 128, 14, 14]               0
           Conv2d-53          [-1, 256, 14, 14]          33,024
      BatchNorm2d-54          [-1, 256, 14, 14]             512
             ReLU-55          [-1, 256, 14, 14]               0
    PointWiseConv-56          [-1, 256, 14, 14]               0
     DepthSepConv-57          [-1, 256, 14, 14]               0
AdaptiveAvgPool2d-58            [-1, 256, 1, 1]               0
           Linear-59                    [-1, 7]           1,799
       LogSoftmax-60                    [-1, 7]               0
================================================================
Total params: 74,663
Trainable params: 74,663
Non-trainable params: 0
```

DogBreedClassificationDeformableCNN Structure

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 112, 112]             896
       BatchNorm2d-2         [-1, 32, 112, 112]              64
              ReLU-3         [-1, 32, 112, 112]               0
            Conv2d-4         [-1, 18, 112, 112]           5,202
            Conv2d-5         [-1, 32, 112, 112]           9,248
        DeformConv-6         [-1, 32, 112, 112]               0
       BatchNorm2d-7         [-1, 32, 112, 112]              64
              ReLU-8         [-1, 32, 112, 112]               0
            Conv2d-9           [-1, 64, 56, 56]          18,496
      BatchNorm2d-10           [-1, 64, 56, 56]             128
             ReLU-11           [-1, 64, 56, 56]               0
           Conv2d-12           [-1, 18, 56, 56]          10,386
           Conv2d-13           [-1, 64, 56, 56]          36,928
       DeformConv-14           [-1, 64, 56, 56]               0
      BatchNorm2d-15           [-1, 64, 56, 56]             128
             ReLU-16           [-1, 64, 56, 56]               0
           Conv2d-17          [-1, 128, 28, 28]          73,856
      BatchNorm2d-18          [-1, 128, 28, 28]             256
             ReLU-19          [-1, 128, 28, 28]               0
           Conv2d-20           [-1, 18, 28, 28]          20,754
           Conv2d-21          [-1, 128, 28, 28]         147,584
       DeformConv-22          [-1, 128, 28, 28]               0
      BatchNorm2d-23          [-1, 128, 28, 28]             256
             ReLU-24          [-1, 128, 28, 28]               0
           Conv2d-25          [-1, 256, 14, 14]         295,168
      BatchNorm2d-26          [-1, 256, 14, 14]             512
             ReLU-27          [-1, 256, 14, 14]               0
AdaptiveAvgPool2d-28            [-1, 256, 1, 1]               0
           Linear-29                    [-1, 7]           1,799
       LogSoftmax-30                    [-1, 7]               0
================================================================
Total params: 621,725
Trainable params: 621,725
Non-trainable params: 0
----------------------------------------------------------------
```

DeformConv Structure



Accuracy curves

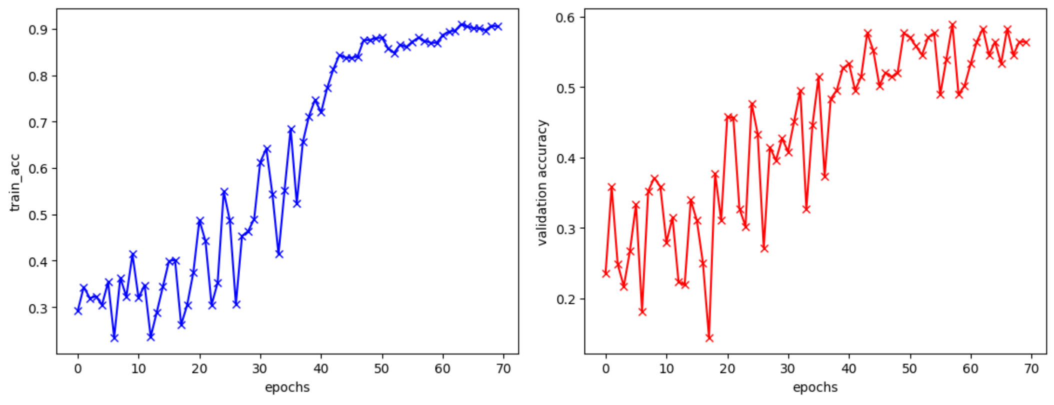


Figure 1: Accuracy curve of epochs_num = 50/14/6 for DogBreedClassificationCNN

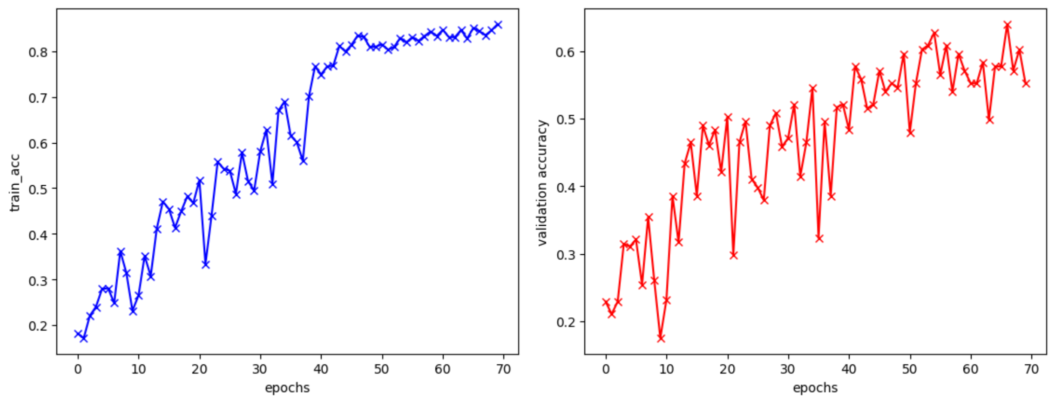


Figure 2: Accuracy curve of epochs_num = 50/14/6 for DogBreedClassificationDSC

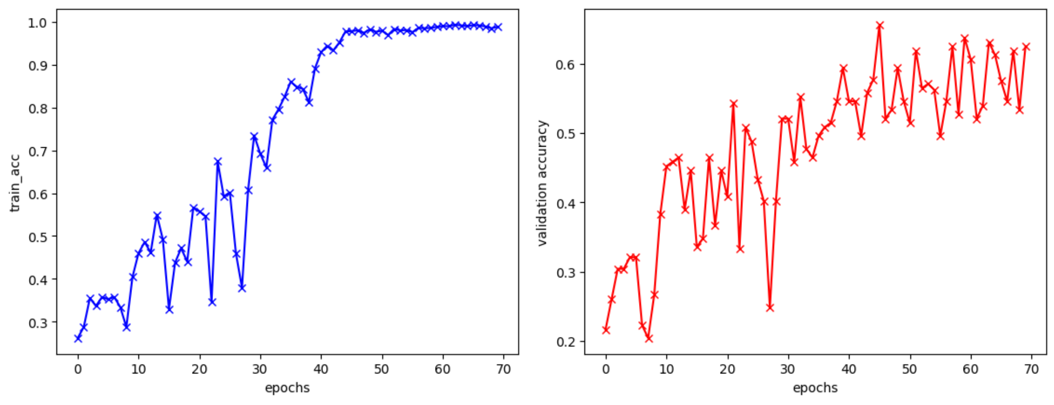


Figure 3: Accuracy curve of epochs_num = 50/14/6 for DogBreedClassificationDeformableCNN