Lab 4 – Dynamic Convolutional Filters for CIFAR-10 Classification

ENGN8536, 2019

20th September 2019

1. Introduction

The goal of this lab is to introduce you a state-of-the-art idea called the Dynamic Filters, which is closely related to the concept of Conditional Probability and the Attention Mechanism, it has been widely applied in many machine learning networks.

You might never come across any of the terms above, but with all the knowledge that you have learnt in this course so far, you should be able to complete this lab. However, if you want to have a deeper understanding of the knowledge behind the lab tasks, please refer to the papers in the reference list.

Let's first take a glance at the dynamic filter with a simple example: Consider a system which takes two signals as input, one is fixed (and assume it is the main signal) while the other one changes with respect to time, the system predicts an output based on the two input signals. Clearly, the output will be different if the second signal varies. An easy way to address this problem is to simply concatenate the two signals and pass to the network for prediction, however, this requires the network to have a strong capability to learn about the variation of the second signal and its relationship to the first signal at the same time, which could be very difficult, especially when the variation is large. What could happen is either the network only learns about the first signal because it is fixed and therefore easy for learning features, or the network only cares about the second signal since it varies all the time and contains more information. Of course, you can initialize numbers of submodules each dealing with a specific type of the second signal. However, in the case that if the signal type is unknown, you will need to do clustering which means lots more work, or you can deal with it implicitly (e.g. Multihead Attention [4]) which makes your network much larger.

This is where the idea of Dynamic Filters comes in – what if we can train a network that can learn to generate different filters each suitable for an/some input itself?

For this lab, we are going to apply dynamic filters on one of the convolutional layers to achieve 'label-guided feature extraction', please read and follow the instruction carefully, and don't forget to answer the questions. Make good use of the PyTorch official documentation and you may find some tricks for coding in the PyTorch discussion forum.

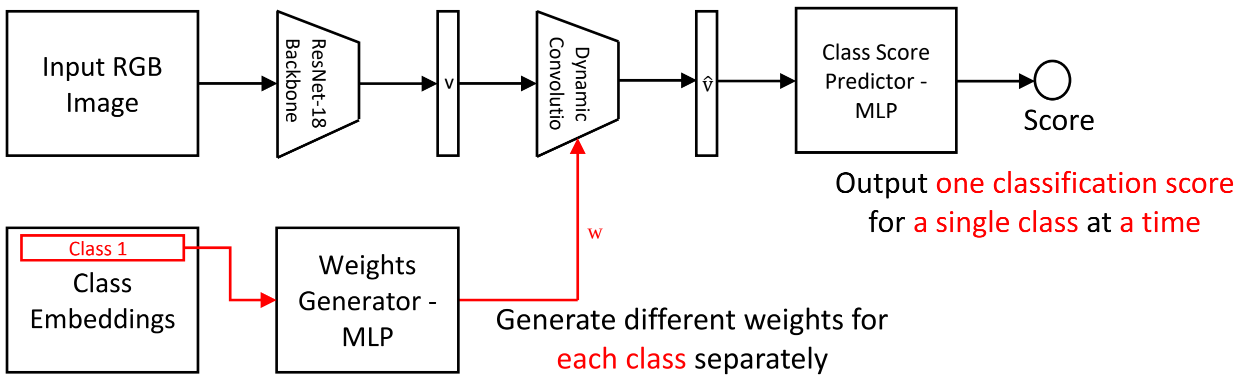
Input RGB Image
ResNet-18 Backbone
v
Dynamic Convolutio
v̂
Class Score Predictor - MLP
Score
Output one classification score for a single class at a time
Class 1
Class Embeddings
Weights Generator - MLP
w
Generate different weights for each class separately

Figure 1. Network schematic.

The network that you are going to build in this lab is shown in Figure 1. In brief, the network takes an image from the CIFAR-10 Dataset as input, encode by the first several layers of a pre-trained ResNet-18 (backbone) and a 1-by-1 dynamic convolutional layer (DC). The encoded image features will pass to a fully-connected layer (FC) to **produce a single score for a class at a time**. The weights for the dynamic convolutional layer are generated by a multi-layer perceptron (MLP), which takes one class embedding at a time as input. By repeating this process for 10 times, each one for a single class and the same input image, you will get 10 prediction scores and now you can use argmax() to assign a label to the input image.

2. Task 1 – Implement the Dynamic Convolutional Network

2.1 Build the network by filling in the code for the BaseModel() class in *model.py*

For the BaseModel initialization: (2 marks)

- Initialize an Dim128 representation for each class label randomly using nn.Embedding().
- Initialize the Weights Generator MLP as FC1 – ReLU – FC2 – Tanh, where FC1 maps input Dim128 to hidden Dim256, FC2 maps hidden Dim256 to output Dim64.
- Initialize a pretrained ResNet-18 model from torchvision.models, however, only use the model up to (layer1) as your Backbone. You can print out the ResNet-18 model from torchvision.models to see how each layer is named. Be careful, do not freeze the weights for the Backbone (i.e. do NOT set params.requires_grad=False).
- Initialize the Dynamic Convolutional (DC) layer as a 1-by-1 2D convolutional layer which maps 64 input channels to 1 output channels with stride 1 and no padding.
- Initialize the Class Score Predictor MLP as FC1 – ReLU – FC2, where FC1 maps input Dim64 to hidden Dim64, FC2 maps hidden Dim64 to output Dim1.

For the forward path: (2 marks)

- Pass the input image to the backbone convolutional network to get feature map v. Just for sanity check, v should be dimension of batch_size $\times 64 \times 8 \times 8$.
- Take out one vector representation for a class (e.g. Car) from the Class Embeddings, pass it to the Weights Generator to get a tensor w of Dim64, which is the weight for the DC layer.
- Apply L2-normalization on w.
- Set the weights of the DC layer as the normalized w.
- Pass the feature map v to the DC layer to get the response map $\hat{v}$. For numerical stability reasons, rescale $\hat{v}$ by $\sqrt{64}$, where 64 is the size of the dynamic filter.
- Reshape the scaled response map $\hat{v}$ and pass it to the Class Score Predictor to get a classification score for a single class (e.g. Car), corresponding to the class-specific DC filter.
- Repeat the above for 10 times for each class to obtain the classification score for each class.

2.2 Fill in the missing code in *train.py* as indicated by the comments

- Save the checkpoints. Basically, for each evaluation(), you need to save the states of the model and the optimizer whenever you trained a better network (e.g. whenever the new testing accuracy is higher than the previous best testing accuracy). (0.5 mark)
- Fill in the code for resume() to load the saved model and optimizer, so that you can resume your network to a state with the highest performance anytime and continue training or run testing. (0.5 mark)

2.3 Train the network and compare to the network without dynamic filter

- Train the network which you have built in 2.1, record the training and testing statistics (any information that you think can best reflect the training process and the result). (1 mark)
- Compare the training stats to the same network without the dynamic mechanism, i.e. stop using the Weights Generator MLP and make the 1-by-1 convolutional layer trainable. (1 mark)

3. Task 2 – Understand the Dynamic Convolutional Network

3.1 Visualize the response map for an arbitrary input

As shown in Figure 2, the prediction score for class Plane is the highest among all for the input image. We can see that (although it is very coarse), the shape of the object in the input image is most similar to the shape of region with higher response in the map for Plane.

- Choose a correctly-labelled image from the train or the test set (and different from the image in Figure 2) and visualize its response maps, one for each class. You need to rescale each response map to the same size as the input image. (optional task, you will 1 extra mark if you do the visualization nicely)
- Briefly talk about what do you think the possible usage of the response map could be. (1 mark)

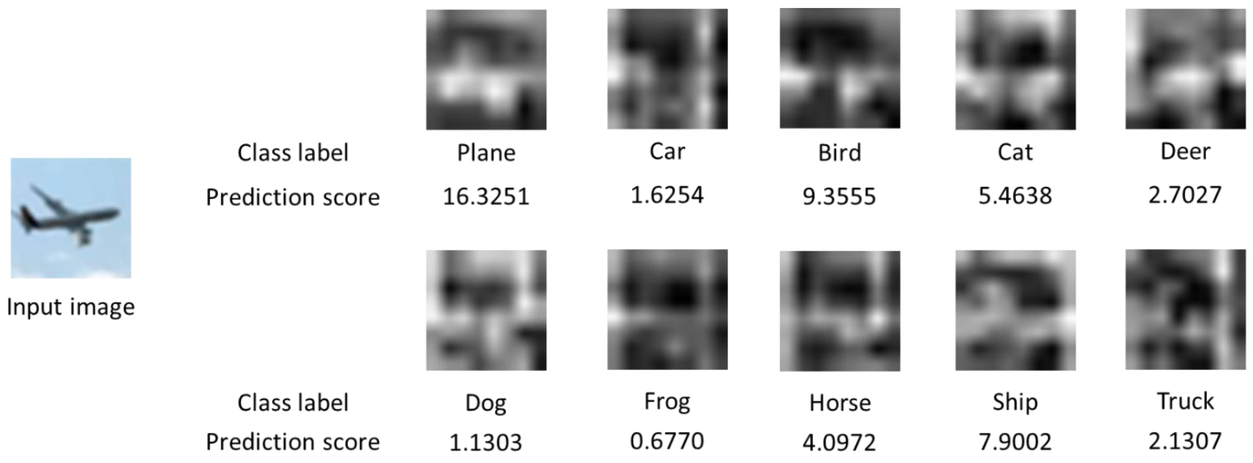

Figure 2. Response map visualization.

3.2 Answer the following questions

- What will happen if you freeze the weights for the Backbone and why is that? If you have to freeze the weights, what could be a possible way to address the issue? (1 mark)
- List at least two advantages and two possible downsides of using the Dynamic Filter. (1 mark)

Notes

The lab report will be due on Sunday, 13 October 2019, 6:00pm. You have two weeks to complete the lab. You are required to complete all tasks and answer all questions. Please submit a single zip file containing a report in pdf format and all the code. Name your pdf as Lab_4_uxxxxxxx.pdf, replacing uxxxxxxx with your uni-ID.

This Lab is worth 5% of the total course assessment.

References

[1] Jia, X., De Brabandere, B., Tuytelaars, T. and Gool, L.V., 2016. Dynamic filter networks. In *Advances in Neural Information Processing Systems* (pp. 667-675).

[2] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. and Bengio, Y., 2015, June. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048-2057).

[3] Landi, F., Baraldi, L., Corsini, M. and Cucchiara, R., 2019. Embodied Vision-and-Language Navigation with Dynamic Convolutional Filters. *arXiv preprint arXiv:1907.02985*.

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, and Polosukhin, I., 2017. Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

[5] Gavrilyuk, K., Ghodrati, A., Li, Z. and Snoek, C.G., 2018. Actor and action video segmentation from a sentence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5958-5966).