Deep Learning – HW2

劉仁忠 RE6125015

1. **Task: Designing a Convolution Module for Variable Input Channels**

In this task focused on designing dynamic convolution layers, I have developed four variations (v1, v2, v3, v4) to explore the concept of dynamic convolution, using ResNet18 as the foundational model. Each of these layers is engineered to adjust its weights according to the number of input channels, thereby enabling the selection of channels and generation of convolutional weights tailored to the incoming data. This customization enhances the model's ability to process relevant image features effectively. Each of these layers modifies ResNet18's initial convolution layer into a dynamic one. Below is an overview of each layer:

* v1: Implements masking to exclude non-essential channels, selecting only relevant channels for which it dynamically generates weights.
* v2: Instead of creating weights for all possible channels, this version produces weights for each specific subset size. It uses a channel mask to select relevant channels, applying this mask to dynamically generate weights.
* v3: Incorporates the dynamic convolution layer described in “Dynamic Convolution: Attention over Convolution Kernels” by Wu et al., CVPR 2020, leveraging Attention2D and Dynamic2D for a dynamic weight-generation network.
* v4: Similar to v2 but includes a mechanism to randomly drop channels with a controlled probability. This approach encourages the model to learn more robust features by not consistently relying on specific channels or features, thus fostering the development of redundant pathways and enhancing generalization.

Training and validation phases are governed by an early stopping criterion, meaning the number of epochs may vary across models. **For detailed metrics such as precision, recall, F1-score, and class support, as well as confusion matrices and plots of training and validation accuracy and loss, please refer to the accompanying “.ipynb” file.**

Batch size: 64, input image size: (84, 84, 3)

Computational costs:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | #PARAMS (M) | FLOPS (GFLOPS) | Memory Usage (Mb) |
| ResNET18 | 11.194 | 135.887 | 2026 |
| v1 | 11.199 | 502.252 | 9150 |
| v2 | 11.204 | 407.667 | 1790 |
| v3 | 11.192 | - | 3016 |
| v4 | 11.204 | 271.778 | 1710 |

#PARAMS and FLOPS were count using ***calflops*** packages provided in [*https://github.com/MrYxJ/calculate-flops.pytorch/tree/main*](https://github.com/MrYxJ/calculate-flops.pytorch/tree/main)

Experiment results (chosen as best model from early stopping):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | # of epochs | Training Acc (%) | Training Loss | Validation Acc (%) | Validation Loss | Testing Acc on RGB (%) |
| ResNET18 | 15 | 63.32 | 1.1510 | 59.11 | 1.4622 | 54.89 |
| v1 | 16 | 72.89 | 0.8495 | 64.00 | 1.2063 | 65.56 |
| v2 | 12 | 59.98 | 1.2793 | 48.89 | 1.7110 | 53.11 |
| v3 | 12 | 61.63 | 1.2195 | 54.44 | 1.5170 | 54.89 |
| v4 | 18 | 61.74 | 1.2377 | 48.67 | 2.0200 | 46.44 |

As shown in the table, version 1 (v1) of the dynamic convolution layers already improves performance significantly, achieving a 65.56% accuracy, which is a 10% increase over the baseline ResNet18 model at 54.89%. However, v1 has a limitation: it can only infer using the same channel configuration it was trained on. For example, if it is trained with RGB images, it is restricted to inferencing with RGB as well. It cannot process different channel combinations (such as RG, GB, R, G, B) without being retrained for each specific configuration. This limitation prompted the development of additional versions (v2, v3, and v4), allowing for the exploration of how different channel combinations affect performance without needing retraining. The testing accuracies for models v2, v3, and v4, which utilize various channel combinations, are detailed below.

Testing accuracy for v2, v3, and v4 model:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | RGB | RG | RB | GB | R | G | B |
| v2 | 53.11 | 2.89 | 2.89 | 3.78 | 2.67 | 3.78 | 4.44 |
| v3 | 54.89 | 5.11 | 4.67 | 4.22 | 3.11 | 2.89 | 2.22 |
| v4 | 46.44 | 26.89 | 31.78 | 28.89 | 15.11 | 17.78 | 19.11 |

Ablation Studies and Analysis

In our analysis, both versions 2 (v2) and 3 (v3) demonstrated similar performances, which were notably weaker when utilizing different channel combinations during inference. Specifically, accuracy dropped significantly in v2 from 53% to around 2-3% and in v3 from 55% to approximately 3-4% with different channel combinations. Although these versions are capable of inferencing with various channel configurations without the need for retraining, the resultant accuracy is considerably low, highlighting their limitations.

This observation led to the development of version 4 (v4), which incorporates random channel dropping during training to enhance the model's robustness across diverse channel combinations. Despite v4 achieving a lower testing accuracy in RGB at 46.44%, it shows relatively better performance in handling 2-channel and 1-channel inferences, with accuracies of about 29% and 18%, respectively. This outcome is consistent with the logic behind random channel dropping; by occasionally omitting channels during training, the model does not overly depend on any specific channels, thereby learning to extract useful features from whatever channels are available. This method promotes a more versatile and resilient approach, making v4 an intriguing option for scenarios requiring flexibility in channel usage.

References

1. **Designing a Two-Layer Network for Image Classification**

Batch size: 16 input image size: (84, 84, 3)