|  |  |
| --- | --- |
| Lab 3: Image Classification | |
| Student ID: 24110164 | Student Name: 최태빈 |

|  |  |  |
| --- | --- | --- |
| Model | Validation Accuracy | The Number of Parameters |
| ConvNetV1 |  |  |
| ConvNetV2 |  |  |
| ModernConvNet |  |  |
| Vision Transformer |  |  |
| Your Model |  |  |

**Problem 1. Convolutional Neural Networks (1pt)**

1. Run “train\_convnet\_v1.py”, and then write the result to the Table.

**Problem 2. Convolutional Neural Networks with Modern Training Techniques (2pt)**

1. Run “train\_convnet\_v2.py”, and then write the result to the Table.
2. Compare ConvNetV1 and ConvNetV2. What makes them different?
3. Data augmentation 기법

v1에서는 기본적인 정규화만 적용했지만 v2에서는 RandomCrop, RandomHorizontalFlip, RandAugment, RandomErasing을 추가하였습니다.

1. 모델 구조

v2에서는 v1과 달리 dropout과 droppath을 이용하였습니다.

1. 학습 기법

v2에서는 v1과 달리 Mixup 또는 CutMix 기법을 사용하였고, label smoothing을 적용한 CrossEnropyLoss를 사용하였습니다.

1. List all regularization and data augmentation methods for ConvNetV2.
2. Regularization

* Dropout: 모델의 일부 뉴런을 무작위로 비활성화 하여 과적합을 방지합니다.
* DropPath: 중간 레이어의 출력을 x, x + f(x) 중 하나로 확률적으로 선택하여 과적합을 방지하고 모델의 robustness를 향상시킵니다.
* Weight Decay: 가중치의 크기를 제한하여 모델의 복잡도를 줄입니다.
* Label Smoothing: 학습 데이터의 label을 one-hot 벡터가 아닌 노이즈가 추가된 형태로 이용하여 학습 데이터를 과신뢰 하지 않도록 합니다.

1. Data augmentaion

* RandomCrop: 이미지의 일부분을 무작위로 잘라내어 사용합니다.
* RandomHorizontalFlip: 이미지를 수평으로 뒤집습니다.
* RandAugment: 여러 가지 증강 기법을 무작위로 적용합니다.
* RandomErasing: 이미지의 일부 영역을 무작위로 지웁니다.
* Mixup: 두 개의 이미지와 레이블을 선형 결합하여 새로운 샘플을 생성합니다.
* CutMix: 한 이미지의 일부를 잘라내어 다른 이미지에 붙여넣습니다.

1. Explain what the mixup and cutmix methods are, respectively.

Mixup

이는 data augmentation의 한 방법으로, 두개의 이미지와 각각의 해당하는 레이블을 선형적으로 결합하여 새로운 학습 샘플을 생성하는 방법입니다.

새로운 이미지: x̃ = λx\_i + (1-λ)x\_j

새로운 레이블: ỹ = λy\_i + (1-λ)y\_j

이 됩니다. 여기서 λ는 베타 분포에서 샘플링된 혼합 비율값으로 0과 1사이 값을 가집니다.

CutMix

이 또한 data augmentation기법 중 하나로, 두 개의 이미지를 조합하여 새로운 학습 샘플을 생성하는 방법입니다. 두 이미지 A, B를 뽑은 후 무작위로 이미지 A에서 직사각형 영역을 잘라내고 이를 이미지 B의 같은 위치에 붙여 넣습니다. 새로운 이미지의 레이블은 λy\_A + (1-λ)y\_B가 되고 여기서 λ는 새로운 이미지에서 A부분이 차지하는 면적의 비율입니다.

이 두 방법 모두 모델의 일반화 성능을 향상시키고 클래스 간의 관계를 학습하는 데 도움을 줄 수 있습니다.

1. Explain what the collate\_fn is in line 82 is~83 in the train\_convnet\_v2.py

collate\_fn은 데이터 로더에서 배치를 생성할 때 사용되는 함수입니다. train\_conv\_net\_v2.py에서는 Mixup, CutMix 중 한가지를 랜덤으로 선택한 후 배치내에서 적용하여 최종적인 배치를 생성하도록 했습니다.

**Problem 3. Modernize Convolutional Neural Networks (2pt)**

1. Implement “FeedForward” class in “models/modernconvnet.py”

*“FeedForward” class consists of two fully-connected layers with SwiGLU activation, which is a popular activation in modern large language models such as LLAMA 3. We can write the operation of “FeedForward” class as , where is a layer input, is a layer output, is an element-wise multiplication, is a sigmoid function, and , , and are weight matrices.*

1. Implement “ModernConvBlock” class in “models/modernconvnet.py”

*“ModernConvBlock” class is a residual block. Specifically, it first performs a depthwise convolution and the normalizes the tensor using a layer normalization method. It then performs two pointwise convolution with SwiGLU activation. This can be achieved by the “FeedForward” class. Next, it applies a droppath technique. Finally, it returns the output tensor by adding its input and the output of droppath.*

1. Implement “ModernConvNet” class in “models/modernconvnet.py”

*“ModernConvNet” class in “*models/modernconvnet.py*” is similar to the “ConvNet” class in “models/convnet.py” Be careful to the shape of tensors.*

1. Run “train\_modernconvnet.py”, and then write the result to Table.
2. Compare ConvNetV2 and ModernConvNet. Which architecture do you think is more efficient? Please explain why you think so.

**Problem 4. Vision Transformer (2pt)**

1. Implement “Attention” class in “models/vit.py”

*“*Attention*” class is a standard multi-head self-attention layer for the transformer block. It takes an input tensor of size (B, N, C), where B is a batch size, N is the number of tokens, and C is the dimension of each token.*

1. Implement “PatchEmbedding” class in “models/vit.py”

*“PatchEmbedding” class first separates an input image of size (B, 3, H, W) into non-overlapping patches (tokens) of size (B, H / P x W / P, P x P x 3), where H, W, and P are the height of image, width of image, and the patch size, and then encodes each patch into feature vector using a linear layer. Next, it adds the positional encoding for each patch, to feed the token’s spatial information to the transformer blocks.*

1. Run “train\_vit.py”, and then write the result to the Table
2. Compare ModernConvNet and VisionTransformer. Which model achieves higher performance? Please explain why this result is obtained.

**Problem 5. Network Design (1pt)**

1. Now, we want to replace some stages of “ModernConvNet” class with some transformer blocks. Feel free to design your model as long as the number of parameters of your model does not differ significantly from the number of parameters of the other models. Explain your design.
2. Train your model, and then write the result to Table