NYCU Introduction to Machine Learning, Final

- ## Environment details (5%)
  - ### Python version
    Python 3.10.12
  - ### Framework
    PyTorch
  - ### Hardware
    GPU (Google Colab)

- ## Implementation details (15%)
  - ### Model architecture
    ResNet-50 with the number of output features in the last layer modified to 200 (corresponding to the total number of classes in this case).
  - ### Hyperparameters

1. re_size = 256
size of the input image to ensure consistency

2. crop_size = 224
size of the cropped image

3. batch_size = 64
number of images trained as a group in each iteration

4. lr_begin = (batch_size / 256) * 0.1
learning rate at the beginning

5. epochs = 60
total number of training iterations

6. seed = 0
random seed to ensure result consistency

- Training strategy

Fine-tune a ResNet50 model with pre-trained weights on ImageNet.

The training strategy involves a gradual reduction of the learning rate using "Cosine Learning Rate Decay" in each epoch, promoting a refined learning pattern.

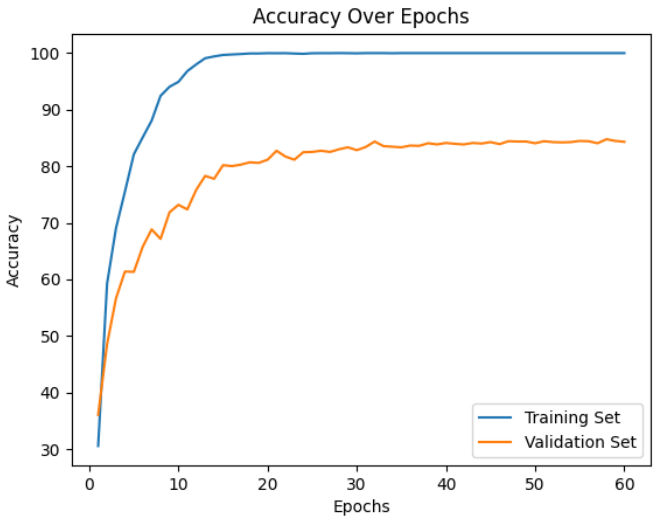
The optimization is performed with "Stochastic Gradient Descent" (SGD), allowing for faster updates by updating parameters based on a subset of the training data.

Additionally, the training incorporates "Label smoothing" in the loss calculation to encourage better generalization. This smoothing mechanism prevents the model from being overly confident in its predictions, mitigating overfitting and enhancing generalization.

Experimental results (15%)

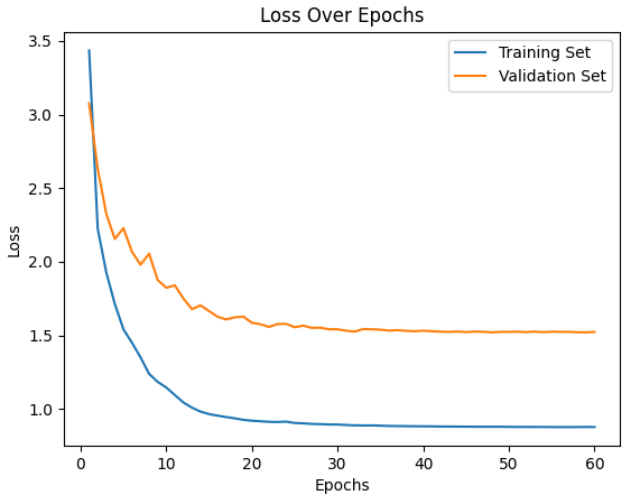
- Evaluation metrics

Split the entire dataset into training and validation sets in an 8:2 ratio, and evaluate the model using “Accuracy” on both the training and validation sets.



- ## Learning curve

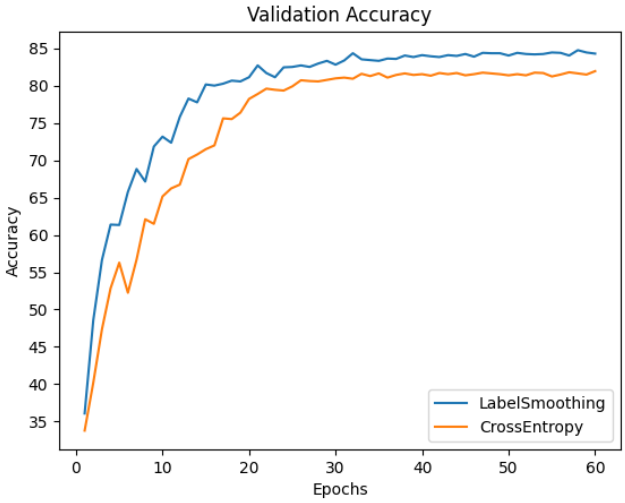
Calculate the Loss while incorporating Label Smoothing trick for both the training and validation sets.



- ## Ablation Study

I wondered whether the "Label Smoothing" technique makes a difference, so I compared the model's performance with and without Label Smoothing.

Modify the loss criterion from "LabelSmoothingLoss()" to "CrossEntropyLoss()," and keep all other architectural aspects unchanged. Here is the result:



There is a difference of 3% in accuracy, suggesting that the Label Smoothing technique does have a meaningful impact on the model’s performance.

- # Bonus (5%)
  - ## papers review

The paper I reference is “Bag of Tricks for Image Classification with Convolutional Neural Networks” [1]. This paper discusses several existing "tricks" to improve model accuracy and experimentally evaluates their impact on multiple network architectures and datasets.

The tricks I apply include:

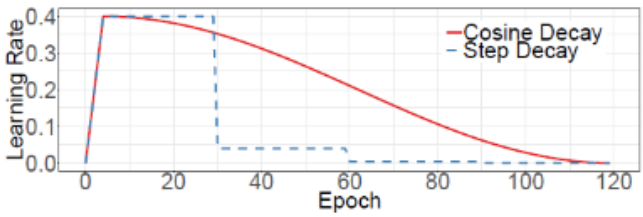
1. Linear scaling learning rate

   The paper suggests a formula for the initial learning rate in SGD. Increasing the batch size in SGD reduces gradient noise, allowing for a higher learning rate. The proposed formula linearly increases the initial learning rate with the batch size, facilitating larger progress along the opposite direction of the gradient.

2. Cosine Learning Rate Decay

   We need to adjust the learning rate along the way for better performance, convergence, and addressing issues like vanishing gradients or exploding gradients. Instead of widely used strategies like “step decay,” the paper proposes a cosine annealing strategy. The idea is to decrease the learning rate from the initial value to 0 by following the cosine function.

   Compared to step decay, cosine decay decreases the learning rate slowly at the beginning, becomes almost linear in the middle, and slows down again at the end. This pattern potentially improves the training progress.



(a) Learning Rate Schedule

3. Label Smoothing

   The paper points out a drawback in the traditional approach of

updating model parameters by minimizing the negative cross-entropy loss between the probability distribution and true labels. This method tends to encourage output scores that are significantly distinctive, potentially leading to overfitting. To mitigate this issue, the paper introduces a trick called "Label Smoothing."

It redefines the probability distribution (q) as:

$$q_i = \begin{cases} 1 - \varepsilon & \text{if } i = y, \\ \varepsilon/(K-1) & \text{otherwise,} \end{cases}$$

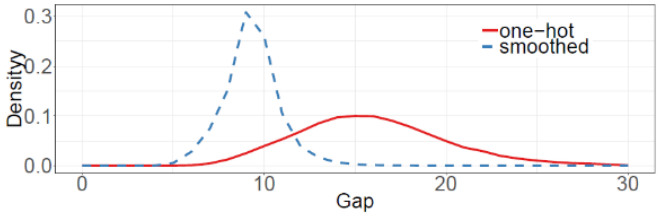
, where ε is a small constant (they use 0.1 in their experiment), and K is the number of classes.

So the optimal solution becomes:

$$z_i^* = \begin{cases} \log((K-1)(1-\varepsilon)/\varepsilon) + \alpha & \text{if } i = y, \\ \alpha & \text{otherwise,} \end{cases}$$

, where α is an arbitrary real number.

By computing the gap between the maximum prediction value and the average of the rest (log((K-1)(1-ε)/ ε) in label smoothing), we see that label smoothing centers the distribution at a lower value and reduces the occurrence of extreme values. This indicates that the resulting model is more stable and exhibits enhanced generalization.



(b) Empirical gap from ImageNet validation set

Reference

[1] He, Tong, et al. "Bag of tricks for image classification with convolutional neural networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.