312540021 / Kezia Nathania (林紀霞)

HW1 Report

Introduction

The objective of the task is to classify images into 100 distinct classes using ResNet as the model backbone. This experiment explores various techniques to enhance image classification performance, including fine-tuning ResNet and ResNeXt models, incorporating the Convolutional Block Attention Module (CBAM) for better feature focus, and using label smoothing to improve generalization. Additionally, data augmentation and an averaging ensemble method are employed to boost model robustness and accuracy.

Method

1. ResNet

It is designed with residual learning, which helps mitigate the problem of gradients vanishing. It is highly capable of learning complex features from large-scale image datasets. Leveraging pre-trained weights on large ImageNet datasets, ResNet50 and ResNet152 are fine-tuned for classifying the given dataset.

- *Final Block Fine-Tuning*
  Fine-tuning just layer4 is computationally efficient and prevents overfitting. The final block (layer4) learns more task-specific features (e.g., higher-level patterns).

2. ResNext

It extends the ResNet (Residual Network) design by incorporating the concept of grouped convolutions to improve efficiency and performance. ResNeXt follows a modular, cardinality-based approach, where multiple parallel convolutional paths (called cardinality groups) enhance feature extraction while maintaining computational efficiency. This design allows ResNeXt to achieve higher accuracy with fewer parameters compared to traditional ResNets, making it a powerful model for image classification and recognition tasks.

3. Data Augmentation

Improve the generalization ability of deep learning models, especially on relatively smaller datasets. The images are randomly cropped and resized to help the model become invariant to object scale and location. They are also randomly flipped horizontally with a 50% probability, which further augments the data and makes the model invariant to left-right orientation.

```python
# Data augmentations and normalization
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
}
```

Applying the same normalization used during pre training ensures that the input data aligns with the distribution the model was trained on, resulting in improved performance and more accurate predictions.

4. Convolutional Block Attention Module (CBAM)

CBAM is a simple yet effective attention module for feed-forward convolutional neural networks. Its lightweight and general characteristics allows it to be integrated into any CNN architecture and is end-to-end trainable along with base CNNs. CBAM can enhance a CNN's accuracy by allowing the network to focus on the most important features in both the channel and spatial dimensions, effectively improving its ability to learn and recognize relevant patterns in the data.

5. Label Smoothing

Label smoothing is a regularization technique that turns “hard” class label assignments to “soft” label assignments. It can enhance the model generalization by reducing the model’s confidence in its predictions, which avoids overfitting.

6. Averaging Ensemble

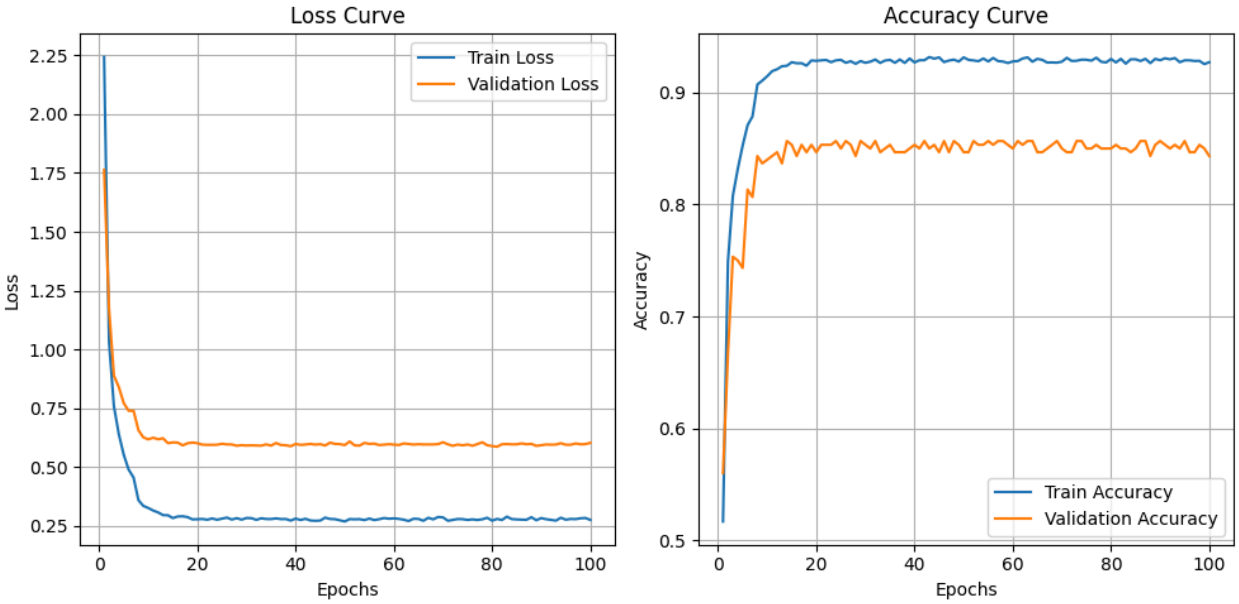
The predictions from multiple models are combined by averaging their output probabilities. This approach can help to reduce model variance and improve overall robustness by leveraging the strengths of different architectures.

7. Experiment

Loss function used is Cross Entropy Loss. It measures the difference between the predicted probabilities and the true class labels, encouraging the model to make more accurate predictions by penalizing incorrect classifications based on the confidence level.

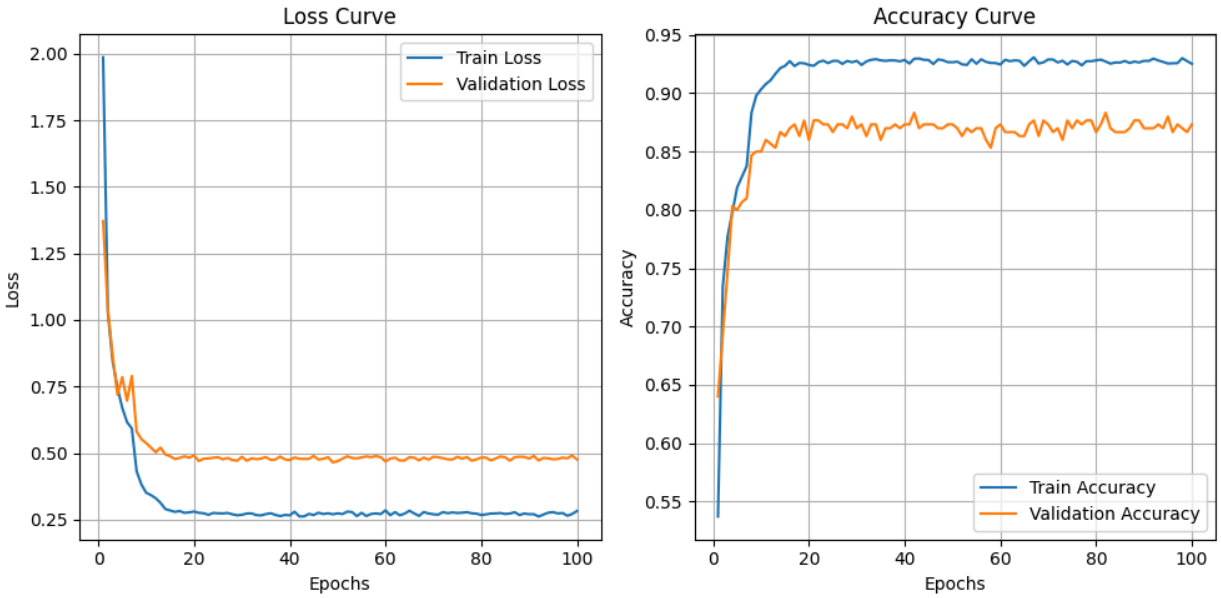
a. ResNet50 (final block)

- Optimizer : Adam
- Epoch : 100
- Learning Rate : 0.0001 (with scheduler)



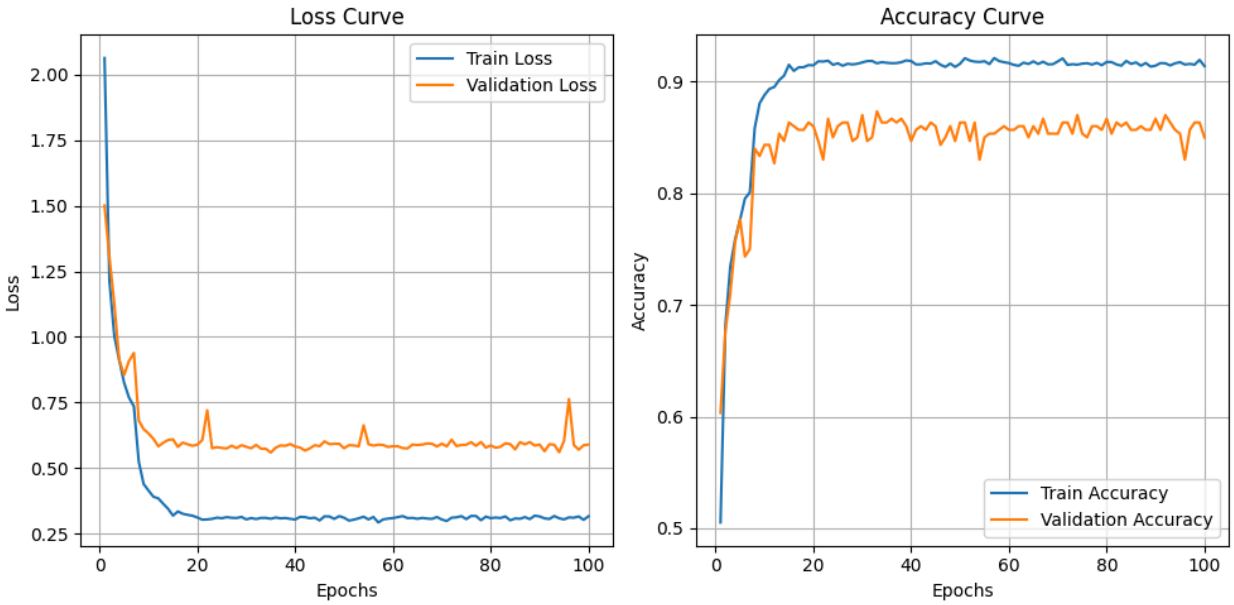
b. ResNet50 + CBAM

- Optimizer : Adam
- Epoch : 100
- Learning Rate : 0.0001 (with scheduler)
- Convolutional Block Attention Module (CBAM) is applied after each layer group (layer1, layer2, layer3, and layer4) to refine the feature maps before they are passed to the next layer or final fully connected layer for classification.



c. ResNet152 + CBAM

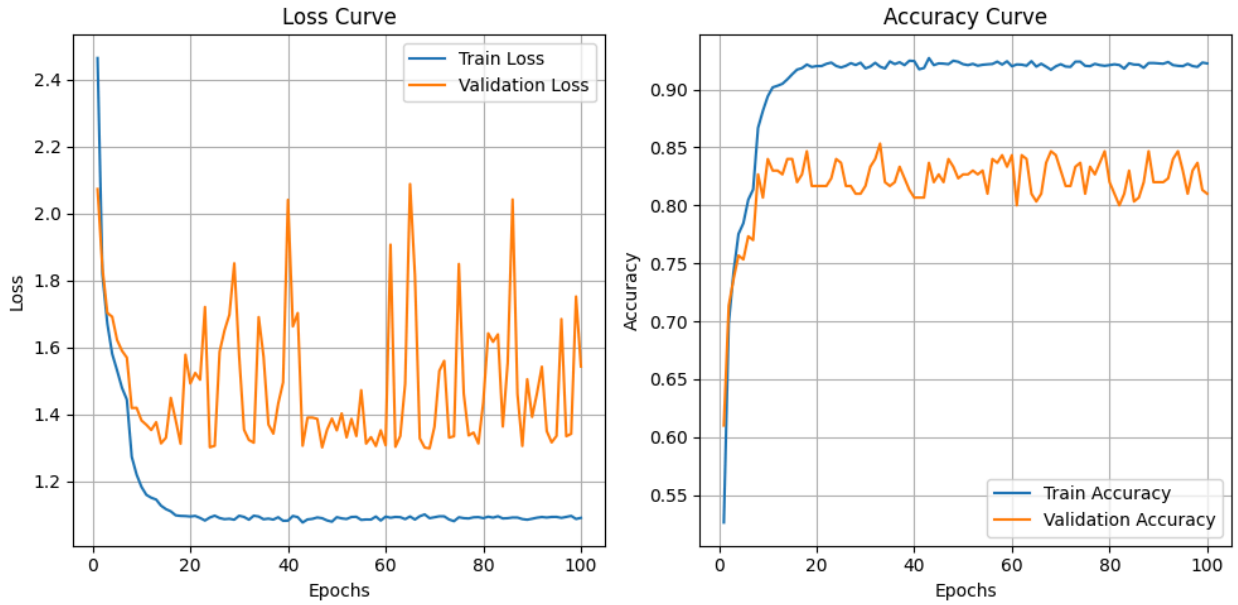
- Optimizer : Adam
- Epoch : 100
- Learning Rate : 0.0001 (with scheduler)
- Convolutional Block Attention Module (CBAM) is applied after each layer group (layer1, layer2, layer3, and layer4) to refine the feature maps before they are passed to the next layer or final fully connected layer for classification.



d. ResNet152 + CBAM + label smoothing

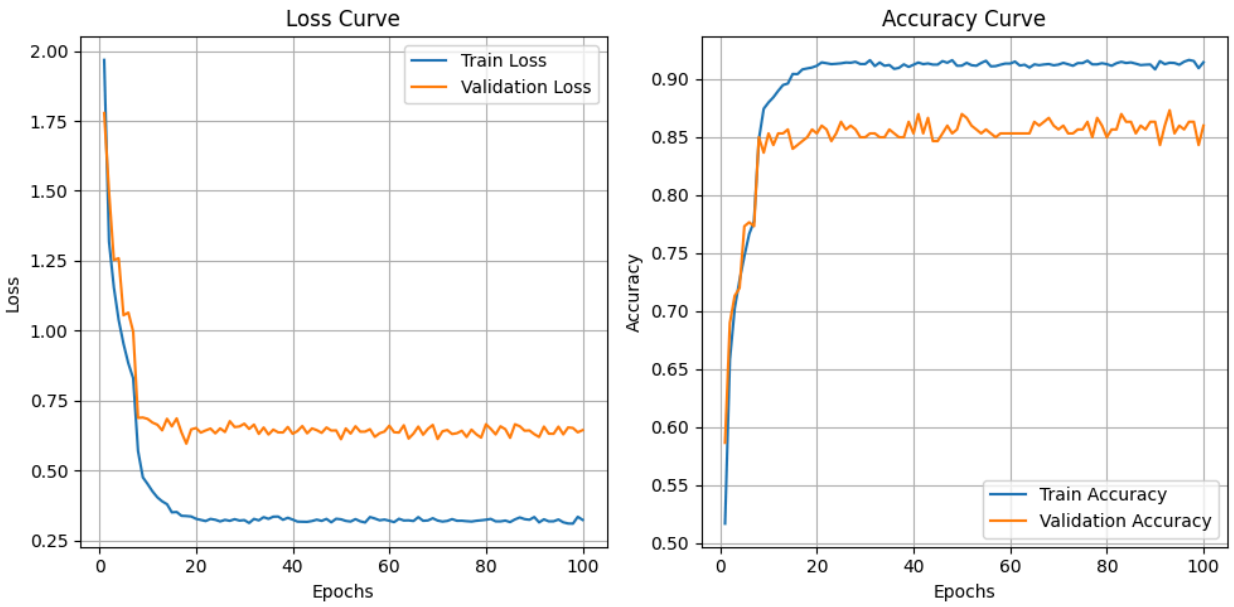
- Optimizer : Adam

- Epoch : 100
- Learning Rate : 0.0001 (with scheduler)
- Label smoothing : 0.1
- Convolutional Block Attention Module (CBAM) is applied after each layer group (layer1, layer2, layer3, and layer4) to refine the feature maps before they are passed to the next layer or final fully connected layer for classification.



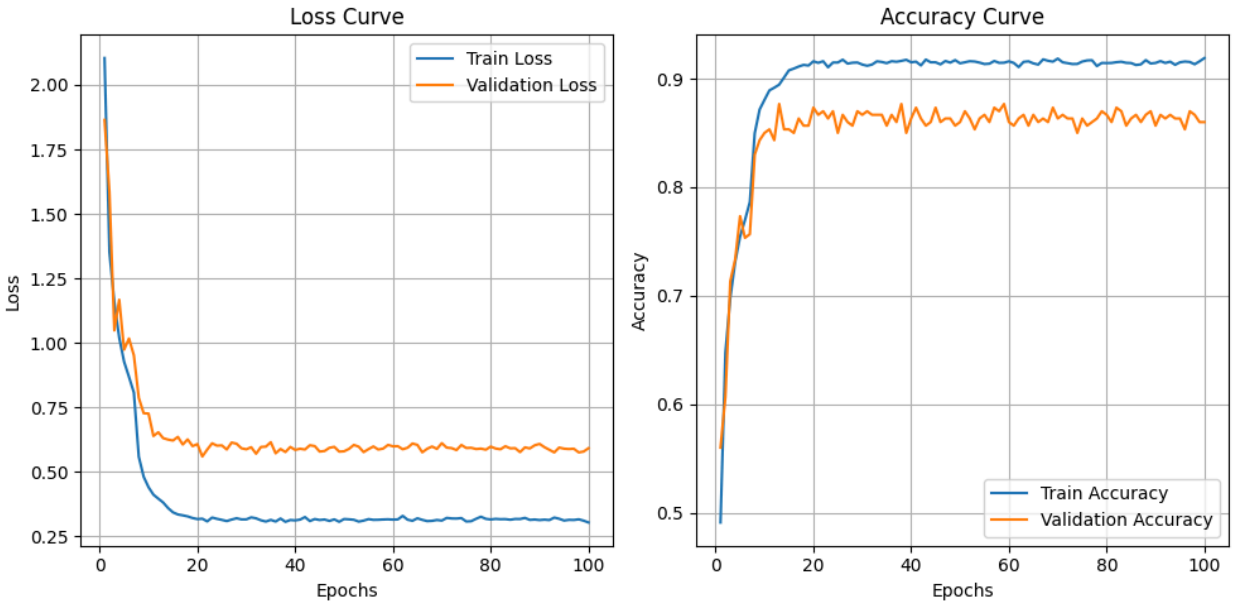
e. ResNext101

- Optimizer : Adam
- Epoch : 100
- Learning Rate : 0.0001 (with scheduler)

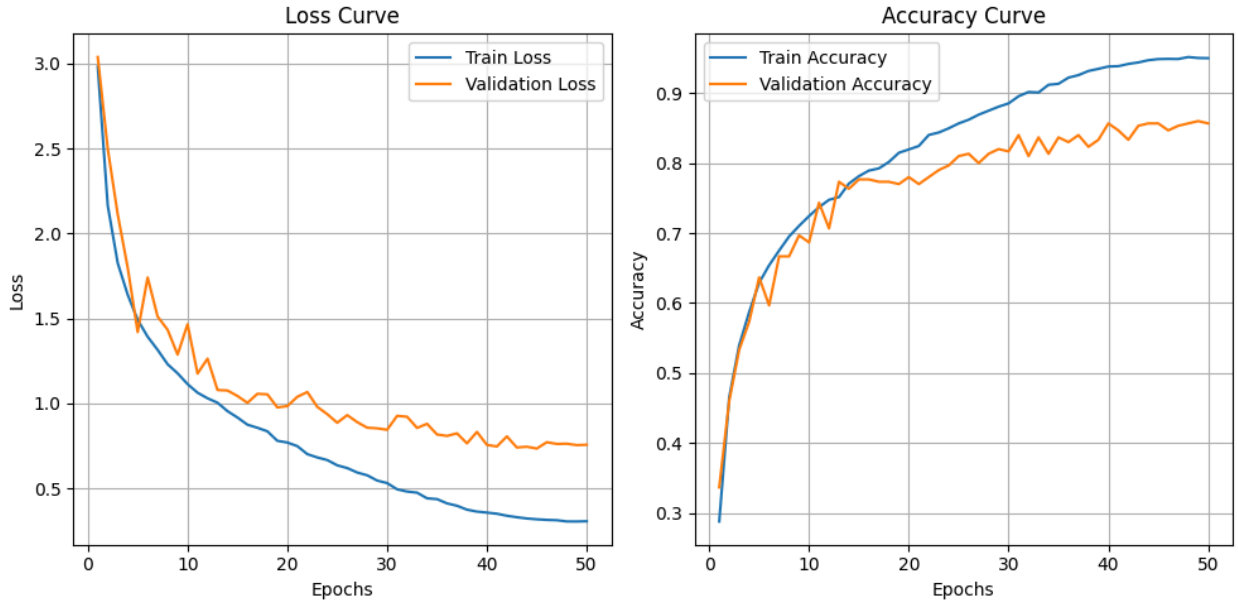


f. ResNext101 + CBAM

- Optimizer : Adam
- Epoch : 100
- Learning Rate : 0.0001 (with scheduler)
- Convolutional Block Attention Module (CBAM) is applied after each layer group (layer1, layer2, layer3, and layer4) to refine the feature maps before they are passed to the next layer or final fully connected layer for classification.



g. ResNext101 + CBAM + label smoothing
   - Optimizer : SGD
   - Epoch : 50
   - Learning Rate : 0.0001 (with scheduler)
   - Label smoothing : 0.01
   - Convolutional Block Attention Module (CBAM) is applied after each layer group (layer1, layer2, layer3, and layer4) to refine the feature maps before they are passed to the next layer or final fully connected layer for classification.



h. Averaging Ensemble: ResNext101 & ResNet152 + CBAM
   - Prediction : Max of averaged output probability from the ResNext101 and ResNet152 + CBAM.

Result

Model	**Num. of Parameters**	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
ResNet50 (final block)	23,712,932	93%	85.67%	90%
ResNet50 + CBAM	24,409,644	92%	88.33%	92%
ResNet152 + CBAM	59,045,420	91%	87.33%	92%
ResNet152 + CBAM + label smoothing	59,045,420	92%	85.33%	92%

ResNext101	86,947,236	91%	87.33%	93%
ResNext101 + CBAM	87,643,948	91%	87.67%	92%
ResNext101 + CBAM + label smoothing	87,643,948	95%	86.00%	92%
Averaging Ensemble: ResNext101 & ResNet152 + CBAM	86,947,236 & 59,045,420			94%

The averaging ensemble method yields the highest accuracy.

The model performance shows that the addition of CBAM generally improves validation accuracy compared to the base ResNet models. For example, ResNet50 with CBAM achieves a validation accuracy of 88.33%, compared to 85.67% for the base ResNet50. Similarly, ResNext101 with CBAM improves validation accuracy to 87.67%.

Label smoothing did not show a clear improvement in validation accuracy for most models. For example, ResNet152 with CBAM and label smoothing achieved a validation accuracy of 87.33%, slightly lower than ResNet152 with CBAM alone (88.33%).

The ensemble model, which combines ResNext101 and ResNet152 with CBAM, achieved the highest test accuracy of 94%. This suggests that combining models with different architectures can lead to improved performance.

Overall, CBAM has a positive effect on model performance, and combining models through ensemble methods can further boost test accuracy.

References

- Woo, S., Park, J., Lee, J. Y., & Kweon, I. S. (2018). *CBAM: Convolutional Block Attention Module*. In the European Conference on Computer Vision (ECCV) 2018. Retrieved from https://openaccess.thecvf.com/content_ECCV_2018/html/Sanghyun_Woo_Convolutional_Block_Attention_ECCV_2018_paper.html
- Chollet, F. (2019, December 30). *Label smoothing with Keras, TensorFlow, and deep learning*. PyImageSearch. Retrieved from https://pyimagesearch.com/2019/12/30/label-smoothing-with-keras-tensorflow-and-deep-learning/
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 770-778. https://doi.org/10.1109/CVPR.2016.90
- Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). *Aggregated residual transformations for deep neural networks*. Proceedings of the IEEE conference on

computer vision and pattern recognition (CVPR), 1492-1500. https://doi.org/10.1109/CVPR.2017.634

GitHub Link

https://github.com/Kezia-Nathania/NYCU-Visual-Recognition-2025-Spring-HW1.git