Homework 1:
Image Classification with ResNet-50

Marie Picquet
Student ID: 313551818
National Yang Ming Chiao Tung University
Selected Topics in Visual Recognition
using Deep Learning (535521)

March 26, 2025

1 Introduction

The goal of this assignment is to develop an image classification model using a convolutional neural network (CNN) capable of accurately categorizing a given dataset. The ResNet-50 architecture selected for this task is known for its effectiveness in training very deep networks using residual connections, which help mitigate the vanishing gradient problem [1].

Since the base ResNet-50 model is pretrained on the ImageNet dataset, several adaptations were necessary to tailor it to the characteristics of the target dataset. To improve generalization and reduce overfitting, data augmentation techniques were applied during training. Given the unbalanced class distribution in the dataset, a weighted cross-entropy loss function was employed to place greater emphasis on minority classes. Furthermore, layers of the network were frozen, and only the final classification layer and the fourth residual block (`layer4`) were unfrozen to allow fine-tuning on the target dataset.

Model training was conducted on Google Colab using GPU acceleration. Early stopping and learning rate scheduling were also incorporated to improve training efficiency and performance stability. Furthermore, checkpoints were saved for the best-performing model.

The complete code for this assignment is available at the following GitHub repository: `https://github.com/mariep00/535521-HW1.git`.

2 Method

Dataset

The provided dataset consists of RGB images categorized into 100 different classes. The data distribution is not uniform across classes, which can introduce challenges during training.

Image type	RGB
Number of classes	100
Class distribution	Unbalanced across classes
Training/Validation set	21,024 labeled images
Test set	2,344 unlabeled images

Table 1: Dataset summary

Data Preprocessing

All input images were resized to 224×224 pixels and normalized using the mean and standard deviation of the ImageNet dataset to ensure compatibility with ResNet-50.

Data Augmentation

To improve model generalization and prevent overfitting, the following transformations were applied to the training data:

- RandomHorizontalFlip
- ColorJitter (brightness, contrast, saturation, hue)
- RandomRotation ($\pm 15°$)

Model Backbone

The ResNet-50 architecture was used as the base model, as shown in Figure 1. The final fully connected layer was replaced with a new linear layer to accommodate 100 output classes. The implementation followed a reference from a ResNet example on CIFAR-10 [6].

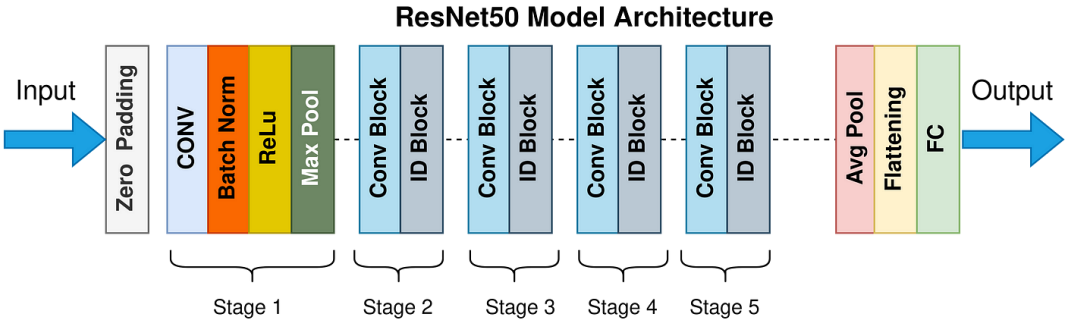

Figure 1: ResNet-50 architecture used as the base model.

Freezing Layers

During training, the model's parameters were selectively frozen to preserve pretrained feature representations. Specifically, the final classification layer (`fc`) and the fourth residual block (`layer4`) were unfrozen, allowing them to be fine-tuned on the target dataset. This decision was made due to limited computational resources. Unfreezing more layers would have increased training time and added more hyperparameters to adjust. By keeping the earlier layers frozen, the model trained more efficiently, adapted to the new dataset through the unfrozen layers, and reduced the risk of overfitting.

Training Hyperparameters

Optimizer

The model was trained using the AdamW optimizer. This variant decouples weight decay from the gradient update, enabling more effective regularization [3]. While the original ResNet example on CIFAR-10 [6] used a more simple Stochastic Gradient Descent (SGD) optimizer, AdamW showed better results in this setting and was therefore used for the final training pipeline.

Learning Rate

A learning rate of 1×e-4 was initially used during training. However, it was later reduced to 3×e-5 to ensure more stable updates during fine-tuning, as the higher learning rate led to unstable validation performance. A cosine annealing scheduler was applied to gradually decrease the learning rate throughout training, promoting smoother convergence [2].

Batch Size

A batch size of 128 was used for training, validation, and testing to balance training stability and GPU memory usage.

Epochs

The model was trained for 50 epochs, constrained by the computational resources available on Google Colab. Early stopping was used to stop training when validation accuracy converged.

Dropout Rate

A dropout rate of 0.5 was initially applied to the final classification head. The rate was then reduced to 0.2, which provided a better balance between regularization and model capacity. Dropout helps reduce overfitting by randomly deactivating neurons during training [4].

Loss Function

Cross-entropy loss was initially used to train the model. However, due to significant class imbalance in the dataset, class weights were later incorporated to penalize misclassification of under-represented classes more heavily. In addition, label smoothing with a factor of 0.05 was introduced to prevent the model from becoming overly confident in its predictions, thereby improving generalization [5].

The weighted cross-entropy loss is defined as:

$$\mathcal{L}_{\text{WCE}} = -\sum_{i=1}^{C} w_i \cdot y_i \cdot \log(\hat{y}_i)$$

where C is the number of classes, y_i is the ground-truth label (one-hot encoded), $\hat{y}_i$ is the predicted probability for class i, and w_i is the class-specific weight.

This formulation allows the model to focus more on minority classes while maintaining stable optimization [7].

3 Results

The model was trained for 50 epochs, and performance was evaluated using both accuracy and loss on the training and validation sets. As shown in Figure 2 and Figure 3, both training and validation loss decreased, indicating stable convergence. Accuracy curves showed consistent improvement across epochs.

Interestingly, the validation accuracy was slightly higher than the training accuracy for a significant portion of the training process. This uncommon pattern can be attributed to the use of regularization techniques such as dropout, label smoothing, and data augmentation, which are applied during training but not validation. These techniques tend to reduce training accuracy while improving generalization.

Another possible explanation is the use of a weighted cross-entropy loss function. Since class weights were derived from the training set's imbalance, the model was penalized more heavily for missclassifying minority classes during training. In contrast, validation performance—measured using standard accuracy without reweighting—did not apply these penalties. As a result, the validation metric may appear better than the training metric, even if the model performs similarly on both.

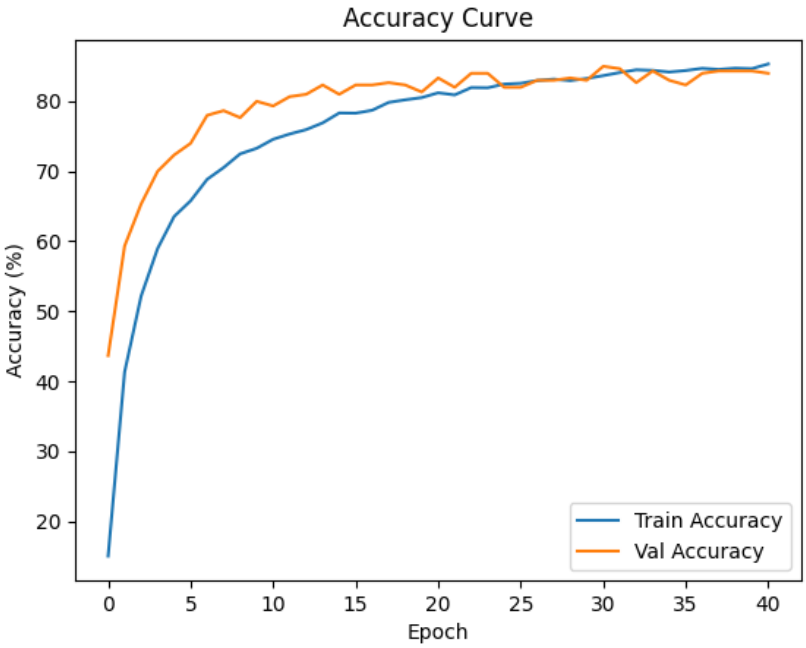

Figure 2: Training and validation accuracy over 50 epochs.

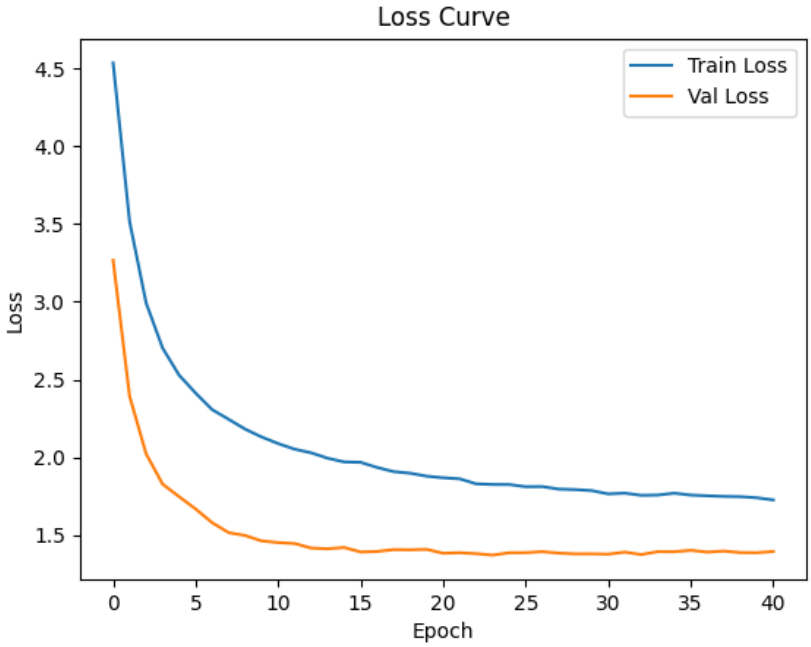

Figure 3: Training and validation loss over 50 epochs.

To evaluate class-wise performance, a normalized confusion matrix was computed on the validation set (Figure 4). The matrix shows that most predictions align closely with ground truth labels, particularly for majority classes. Class weighting in the loss function and the application of label smoothing helped improve predictions for under-represented classes, resulting in more balanced classification outcomes.

Finally, the model achieved a competition score of **89%** on the test set.

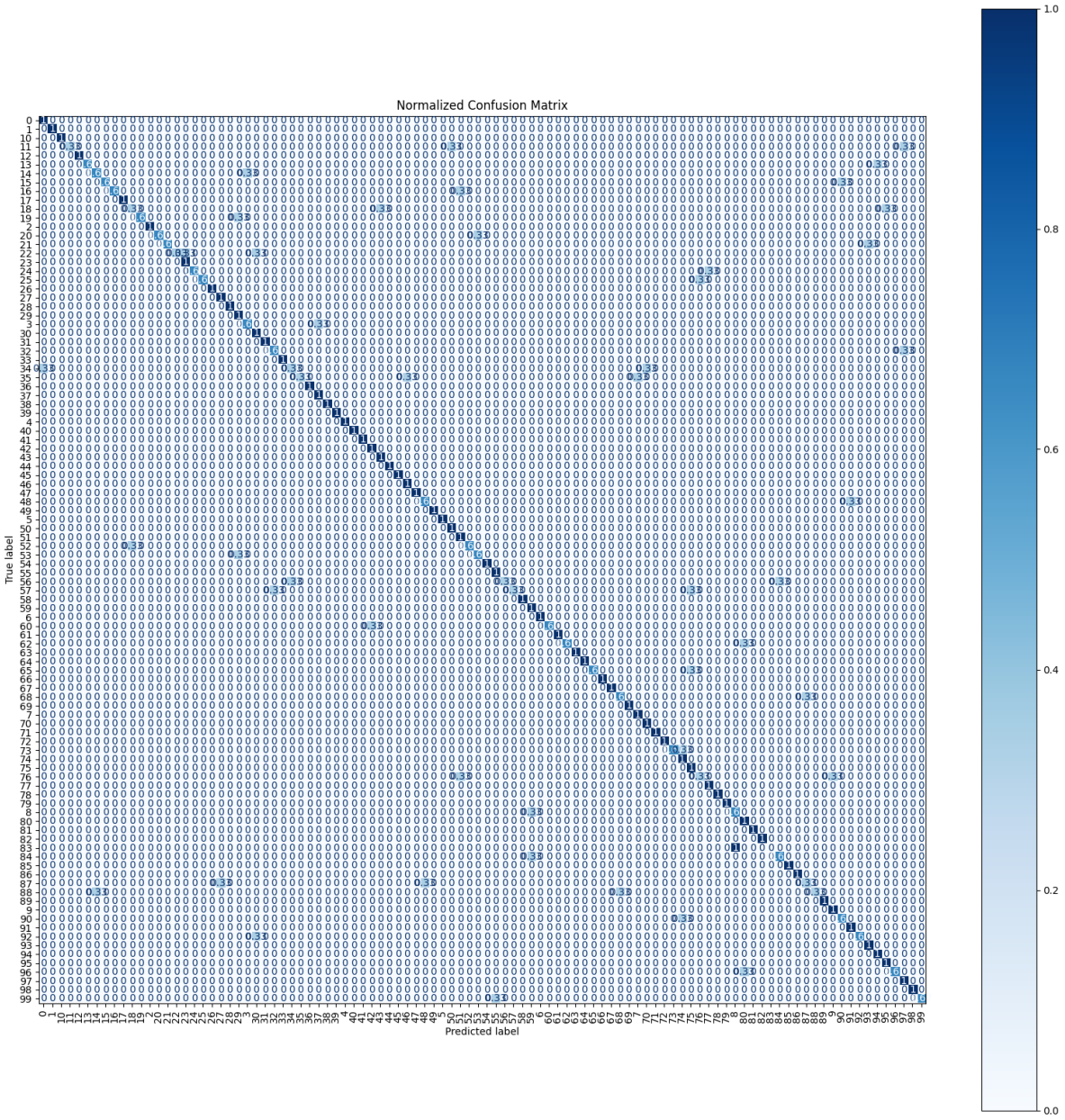

Figure 4: Normalized confusion matrix for the validation set.

References

[1] He, K., Zhang, X., Ren, S., and Sun, J. (2015). *Deep Residual Learning for Image Recognition.* arXiv:1512.03385.

[2] Loshchilov, I., and Hutter, F. (2016). *SGDR: Stochastic Gradient Descent with Warm Restarts.* arXiv:1608.03983.

[3] Loshchilov, I., and Hutter, F. (2019). *Decoupled Weight Decay Regularization.* arXiv:1711.05101.

[4] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* JMLR, 15(1), 1929–1958.

[5] Müller, R., Kornblith, S., and Hinton, G. (2019). *When Does Label Smoothing Help?* arXiv:1906.02629.

[6] GitHub – ResNet on CIFAR10. `https://github.com/JayPatwardhan/ResNet-PyTorch`

[7] King, G., & Zeng, L. (2001). *Logistic Regression in Rare Events Data.* Political Analysis, 9(2), 137–163.