# ECE371 Neural Networks and Deep Learning Assignment 1

**Runbin He 22308046**
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
herb5@mail2.sysu.edu.cn

**Abstract:** This paper details the training of a model to classify flower photographs by employing two prevalent deep learning strategies: data augmentation and transfer learning. Data augmentation techniques—including random cropping, scaling, rotation, and normalization applied to original images—improve the model's generalization performance by preventing the neural network from learning spurious features. For the transfer learning component, a ResNet-18 pre-trained model is utilized; its final fully connected layer is fine-tuned to adapt the model for the required classification tasks. This application of transfer learning significantly curtails training expenditure .

**Keywords:** Data augmentation,Transfer learning

## 1 Introduction

The dataset for this study consists of flower photographs, specifically 588 daisies, 556 dandelions, 583 roses, 536 sunflowers, and 585 tulips. This dataset is characterized by a limited number of distinct categories and a relatively small sample size per category. To address the limitation of data scarcity, data augmentation techniques are employed to generate additional training samples. Moreover, in scenarios with constrained computational resources, transfer learning is adopted to substantially decrease the computational requirements for model training.

The model and associated processes were implemented in Python, leveraging the open-source PyTorch library.

## 2 Related Work

For data augmentation, as early as 1998, the classic paper Lecun et al. [1], which introduced LeNet-5, explicitly mentioned various operations performed on the MNIST handwritten digit dataset, such as horizontal and vertical translations; scaling; squeezing (simultaneous horizontal compression and vertical elongation, or the reverse); and horizontal shearing. Although such operations were initially a validation of the authors' hypothesis that 'increasing the amount of data can improve accuracy,' in practice, their more critical impact was a significant enhancement in the model's generalization ability. Notably, while the term 'data augmentation' was not explicitly used in Lecun et al. [1], this methodology profoundly influenced subsequent research. Later, in the context of the modern deep learning surge, the influential work Krizhevsky et al. [2] formally adopted the term 'data augmentation.' This paper described corresponding operations such as image translation, horizontal flipping, and altering the intensity of RGB channels. This pivotal paper was instrumental in popularizing the systematic use of data augmentation throughout the modern deep learning landscape.

Regarding transfer learning, the 1991 paper Pratt et al. [3] clearly studied how to transfer weights (knowledge) learned by one neural network for a certain task to another neural network for a related task, in order to accelerate learning or improve performance. Pratt et al. [3] also highlighted that

this approach could increase model training speeds by an order of magnitude. Subsequently, the field was comprehensively surveyed and structured in Pan and Yang [4]. This paper systematically defined and categorized various types, methods, and applications of transfer learning. Furthermore, it provided detailed empirical comparisons of performance with and without transfer learning across domains such as sensor network-based localization, text classification, and image classification. A major achievement of this paper was its instrumental role in fostering the widespread acceptance and standardization of the term 'Transfer Learning'.

# 3 Method

## 3.1 Data augmentation

To enhance model generalization and reduce overfitting, our data augmentation strategy involved several sequential operations.

First, images underwent Random Resized Crop to 224×224 pixels, a standard input size facilitating the use of pre-trained models. Subsequently, Random Horizontal Flip was applied with a 0.1 probability to introduce mirrored versions while largely preserving original information. Random Rotation within a range of ±30° was used to simulate common minor pose variations. These initial geometric transformations helped the model learn features from diverse perspectives.

Next, Color Jitter was applied to improve robustness to color variations. This involved randomly adjusting brightness, contrast, and saturation with factors from $[0.8, 1.2]$, and hue with an offset from $[-0.1, 0.1]$, simulating various real-world lighting and camera conditions. Additionally, Random Affine Transformation introduced translations up to $10\%$ of image dimensions (with rotation fixed at 0° as it was handled separately), aiding in learning positional invariance.

Following these augmentation steps, images were converted to Tensor format, and their pixel values were normalized to the $[0.0, 1.0]$ range for training stability. Finally, the image tensors were standardized using the ImageNet dataset's mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$. This step aligns the input data distribution with that of pre-trained models, aiding weight utilization and convergence.

The combined application of these data augmentation techniques significantly increased training data diversity, thereby helping the model learn more invariant feature representations.

## 3.2 Transfer Learning

To maximize model performance with limited computational resources, transfer learning was employed. We utilized the ResNet-18 model, pre-trained on the ImageNet dataset.

For our classification task, characterized by a small dataset and few categories, we retained the ResNet-18's feature extraction backbone. The final fully connected layer was replaced with a new linear layer tailored to our target classes, where num_ftrs represents the input features to the original classifier. All parameters of this modified model were then fine-tuned.

The optimization strategy involved Stochastic Gradient Descent (SGD) with a learning rate of 0.001 and momentum of 0.9. A StepLR learning rate scheduler was also used, reducing the learning rate by a factor of 0.1 every 7 epochs to aid stable convergence. Standard practices were followed by using model.train() during training and model.eval() during validation/testing to manage layers like Dropout and Batch Normalization.

In summary, this study adopted a transfer learning approach using an ImageNet pre-trained ResNet-18, modifying its classifier, adhering to consistent data preprocessing, and employing a defined training strategy. This method is well-suited for limited data scenarios, leveraging prior knowledge for accelerated convergence and enhanced classification performance.

## 4 Experiments

In the first training session, the ResNet-18 model with its default settings was used as the pre-trained model. The training was configured for 20 epochs, with a batch size of 32 samples. The per-epoch training set loss, validation set loss, training set accuracy, and validation set accuracy are presented in Table 1:

| **Metric** | **Epoch 0** | **Epoch 1** | **Epoch 2** | **Epoch 3** | **Epoch 4** | **Epoch 5** | **Epoch 6** | **Epoch 7** | **Epoch 8** | **Epoch 9** |
|---|---|---|---|---|---|---|---|---|---|---|
| Train Loss | 0.959 | 0.433 | 0.368 | 0.324 | 0.310 | 0.271 | 0.290 | 0.240 | 0.260 | 0.237 |
| Train Acc | 0.640 | 0.854 | 0.868 | 0.885 | 0.892 | 0.898 | 0.896 | 0.912 | 0.903 | 0.918 |
| Val Loss | 0.462 | 0.341 | 0.289 | 0.342 | 0.251 | 0.272 | 0.296 | 0.245 | 0.256 | 0.220 |
| Val Acc | 0.844 | 0.883 | 0.881 | 0.884 | 0.914 | 0.912 | 0.895 | 0.912 | 0.914 | 0.914 |
| **Metric** | **Epoch 10** | **Epoch 11** | **Epoch 12** | **Epoch 13** | **Epoch 14** | **Epoch 15** | **Epoch 16** | **Epoch 17** | **Epoch 18** | **Epoch 19** |
| Train Loss | 0.225 | 0.216 | 0.231 | 0.214 | 0.247 | 0.227 | 0.235 | 0.234 | 0.222 | 0.229 |
| Train Acc | 0.923 | 0.927 | 0.921 | 0.927 | 0.908 | 0.921 | 0.914 | 0.916 | 0.922 | 0.919 |
| Val Loss | 0.216 | 0.290 | 0.260 | 0.257 | 0.224 | 0.247 | 0.240 | 0.213 | 0.220 | 0.272 |
| Val Acc | 0.932 | 0.895 | 0.905 | 0.909 | 0.930 | 0.907 | 0.919 | 0.921 | 0.932 | 0.911 |

Table 1: Batch Size = 32, Resnet18 Model Training Data

The training and validation metrics from the first training session are plotted as line graphs in Fig. 1. As observed from Figure 1, the model appears to converge after the 10th epoch. In the subsequent epochs, the validation loss and validation accuracy exhibit small-scale fluctuations, which are considered acceptable.

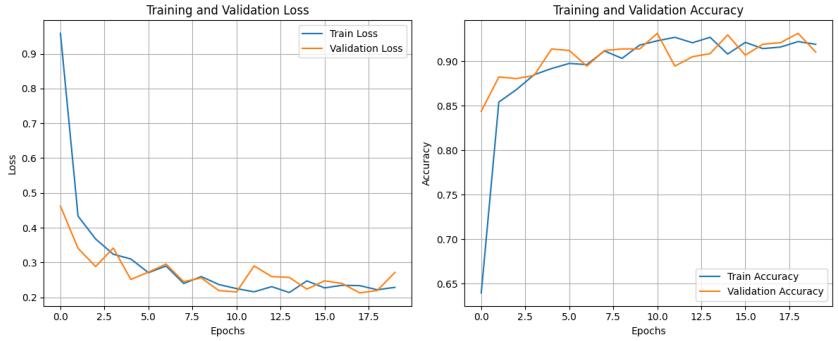


Figure 1: Batch Size = 32, Resnet18

In the second training session, building upon the first, only the batch size parameter was changed to 64. The corresponding training results are presented in Table 2: The results from the second training

| **Metric** | **Epoch 0** | **Epoch 1** | **Epoch 2** | **Epoch 3** | **Epoch 4** | **Epoch 5** | **Epoch 6** | **Epoch 7** | **Epoch 8** | **Epoch 9** |
|---|---|---|---|---|---|---|---|---|---|---|
| Train Loss | 1.115 | 0.557 | 0.435 | 0.365 | 0.326 | 0.313 | 0.299 | 0.284 | 0.285 | 0.281 |
| Train Acc | 0.606 | 0.823 | 0.857 | 0.875 | 0.885 | 0.888 | 0.892 | 0.897 | 0.897 | 0.901 |
| Val Loss | 0.671 | 0.450 | 0.379 | 0.362 | 0.384 | 0.326 | 0.316 | 0.329 | 0.315 | 0.292 |
| Val Acc | 0.805 | 0.851 | 0.870 | 0.856 | 0.858 | 0.879 | 0.893 | 0.890 | 0.891 | 0.890 |
| **Metric** | **Epoch 10** | **Epoch 11** | **Epoch 12** | **Epoch 13** | **Epoch 14** | **Epoch 15** | **Epoch 16** | **Epoch 17** | **Epoch 18** | **Epoch 19** |
| Train Loss | 0.283 | 0.255 | 0.272 | 0.275 | 0.271 | 0.285 | 0.245 | 0.260 | 0.257 | 0.265 |
| Train Acc | 0.900 | 0.905 | 0.902 | 0.905 | 0.904 | 0.905 | 0.913 | 0.911 | 0.912 | 0.913 |
| Val Loss | 0.299 | 0.300 | 0.300 | 0.321 | 0.293 | 0.309 | 0.303 | 0.303 | 0.319 | 0.291 |
| Val Acc | 0.893 | 0.904 | 0.900 | 0.888 | 0.895 | 0.893 | 0.881 | 0.898 | 0.875 | 0.895 |

Table 2: Batch Size = 64, Resnet18 Model Training Data

session were plotted as line graphs in Figure 2 and compared with those from the first training session in Fig. This comparison reveals that with a batch size of 64, the model converges faster, and the trends for the various performance metrics are smoother compared to when a batch size of 32 was used. However, batch size of 64 also results in a slight decrease in accuracy.

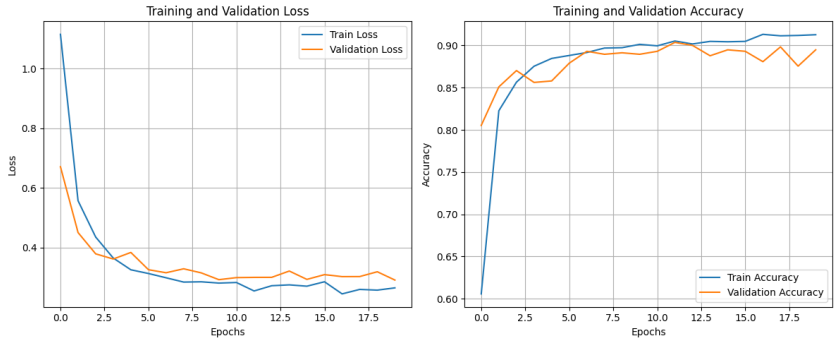


Figure 2: Batch Size = 64, Resnet18

In the third training session, also building upon the setup of the first, the only modification was changing the pre-trained model from the default ResNet-18 to the default ResNet-34. The training results are presented in Table 3:

| **Metric** | **Epoch 0** | **Epoch 1** | **Epoch 2** | **Epoch 3** | **Epoch 4** | **Epoch 5** | **Epoch 6** | **Epoch 7** | **Epoch 8** | **Epoch 9** |
|---|---|---|---|---|---|---|---|---|---|---|
| Train Loss | 0.818 | 0.378 | 0.321 | 0.297 | 0.273 | 0.255 | 0.232 | 0.208 | 0.209 | 0.193 |
| Train Acc | 0.700 | 0.866 | 0.887 | 0.894 | 0.906 | 0.901 | 0.914 | 0.923 | 0.928 | 0.931 |
| Val Loss | 0.391 | 0.327 | 0.329 | 0.274 | 0.252 | 0.281 | 0.254 | 0.271 | 0.297 | 0.257 |
| Val Acc | 0.865 | 0.900 | 0.879 | 0.909 | 0.923 | 0.890 | 0.911 | 0.902 | 0.893 | 0.919 |
| **Metric** | **Epoch 10** | **Epoch 11** | **Epoch 12** | **Epoch 13** | **Epoch 14** | **Epoch 15** | **Epoch 16** | **Epoch 17** | **Epoch 18** | **Epoch 19** |
| Train Loss | 0.223 | 0.206 | 0.197 | 0.193 | 0.186 | 0.185 | 0.189 | 0.191 | 0.193 | 0.190 |
| Train Acc | 0.921 | 0.928 | 0.931 | 0.932 | 0.938 | 0.940 | 0.936 | 0.931 | 0.936 | 0.932 |
| Val Loss | 0.215 | 0.234 | 0.231 | 0.277 | 0.264 | 0.225 | 0.263 | 0.248 | 0.235 | 0.220 |
| Val Acc | 0.914 | 0.916 | 0.905 | 0.905 | 0.909 | 0.925 | 0.911 | 0.914 | 0.921 | 0.916 |

Table 3: Batch Size = 32, Resnet34 Model Training Data)

Figure 3 comparing the results from the third training session (ResNet-34) with those from the first session (ResNet-18) clearly illustrate that the ResNet-34 model converges faster and achieves higher accuracy than the ResNet-18 model. This observation aligns with the material presented on page 59 of the Lecture 3 slides. However, the training duration for the ResNet-34 model was approximately double that of the ResNet-18 model

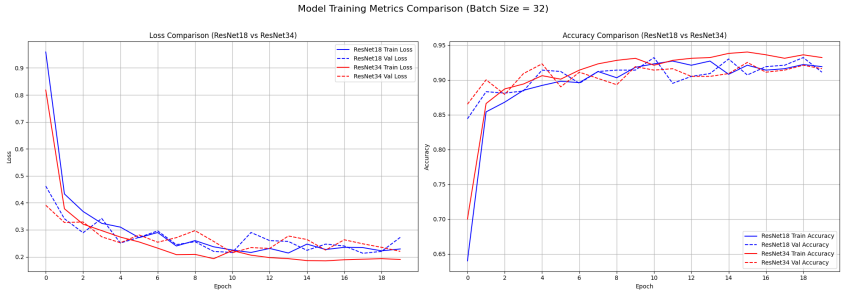


Figure 3: Resnet18 vs Resnet34

## References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[3] L. Y. Pratt, J. Mostow, and C. A. Kamm. Direct transfer of learned information among neural networks. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, volume 2, pages 584–589. AAAI Press, 1991.

[4] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi:10.1109/TKDE.2009.191.