Deep Learning for Medical Imaging 2024 Spring Midterm Project

廖致豪 R11625015[a,b]

[a] National Taiwan University, School of Forestry and Resource Conservation, Taipei, Taiwan
[b] National Taiwan University, Graduate Institute of Biomedical Electronics and Bioinformatics, Taipei, Taiwan

Abstract

This report is Prof. Ruey-Feng Chang's deep learning for the medical imaging spring course's midterm project. I was given the assignment of devising a midterm project focusing on Deep Learning in Medical Imaging. A classification system that can predict different types of breast ultrasound images (US) will be implemented via various state-of-the-art (SOTA) CNN models, such as AlexNet, VGG16, GoogLeNet, and ResNet18. In this project, the images will be preprocessed by performing different transformation processes, and normalized by pre-trained ImageNet parameters. All selected SOTA models will be built with best validation accuracy, to build a baseline model, followed by applying advanced techniques, including data augmentation, and architecture modification, to further improve the performance of SOTA models. The result showed that our work is pretty decent, and the self-modified CNN model has the best accuracy.

Keywords: Breast Ultrasound Images, Deep Learning, State-of-the-Art

1. Introduction

1.1. Background

Medical ultrasound (US) imaging has emerged as a prominent tool for breast cancer detection, owing to its simplicity, affordability, and safety. The interpretation of breast ultrasound images has long been crucial in clinical settings. Therefore, employing machine learning classification techniques to swiftly assess breast ultrasound images can greatly aid in clinical decision-making and treatment planning. In this project, we will we will utilize several kinds of machine learning techniques to classify breast US images.

1.2. Frame The Problem

The problem in this project is a classic multi-class classification problem:

Given a training set of input-output pairs $\mathcal{D}_{train} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, and find f such that $f(\mathbf{x}) = \hat{y} \sim y$, $y \in \{0,1,2\}$ *or* $y \in \{0,2\}$, $\mathbf{x} \in \mathbb{R}^d$. Then, minimize $\mathrm{err}(\hat{y}, y)$, defined as Mean Absolute Error (MAE) in this project, for a learning algorithm to select f out.

1.3. Performance Measurement

This project evaluates the performance of our model by Mean Absolute Error (MAE), we can formulate the equation below (1):

$$MAE = \frac{1}{n}\sum_{i}^{n}|\hat{y}_i - y_i| \tag{1}$$

Where n is the number of labels, $\hat{y}_i$ is the i-th predicted label, and y_i is the i-th true label.

2. Method

2.1. Dataset

The dataset published online by Al-Dhabyani *et al* [1] was used in this project, which comprised 780 images (referred to as BUSI) obtained from 600 women patients, with an average image size of 500 x 500 pixels pixels. The BUSI dataset are categorized into three classes, and there are 133 normal images, 437 malignant masses, and 210 benign tumors. All the training images will be resized to 224 x 224. Figure 1 illustrates examples of breast US imags. To assess the performance of the classification models, common metrics used in medical image studies are employed, such as classification accuracy.

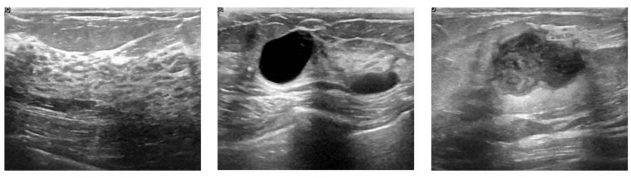

Figure 1: The samples of breast US images with three different categories.

2.2. Preprocess

2.2.1. Data Augmentation

Since CNN models are powerful and widely used for image classification tasks, inherently lack rotational invariance. This means that if an image is rotated, even slightly, the network may fail to recognize the object in the image correctly.

To address the data variability and the rotational invariance, randomly horizontal flip images with a fifty-fifty possibility, followed by randomly rotating the images with default expansion, which means the image size won't be changed, but this helps the network learn to recognize objects from different orientations. By augmenting the data, we introduce variability into the training set, which helps the model learn to generalize better and become more robust to changes in the input data.

2.2.2. Normalization

Normalization improves model performance by scaling input features to a similar range, aiding convergence, and preventing gradient explosion or vanishing. It enhances stability, accelerates training, and enables networks to learn efficiently from data, leading to faster convergence and better generalization. The normalization equation can be depicted below (2):

$$\hat{x}_{i,d} = \frac{x_{i,d}}{max(|x_{1,d}|, |x_{2,d}|, \ldots, |x_{n,d}|)} \tag{2}$$

Where $\hat{x}_{i,d}$ is the i-th row, d-th dimension's new feature, $x_{i,d}$ is the i-th row, d-th dimension's original feature. However, researches suggest [4] [5] that employing pre-trained weights from ImageNet provides better performance than randomly initialized weights. Additionally, studies show that architectures trained on ImageNet tend to generalize well. As a result, normalization methods utilize pre-trained weights from ImageNet, integrating mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225) for feature normalization in this project.

2.3. Models

To ensure uniformity in input image dimensions and better accuracy for classification tasks, some popular SOTA CNN models such as AlexNet, VGG16, GoogLeNet, and ResNet18 were selected for training in this project.

2.3.1. AlexNet

In our project methodology, we include AlexNet [6] as the baseline model for our image classification experiments. It's architecture is depicted on Figure 2. AlexNet consists of five convolutional layers followed by three fully connected layers, with ReLU activation functions applied after each convolutional and fully connected layer. The architecture is characterized by incorporating max-pooling layers to downsample features and dropout layers to prevent overfitting, and the layers enable the model to automatically learn hierarchical features from raw image data.

During fine-tuning, we typically modify the final fully connected layers of AlexNet to match the number of classes in our dataset. By adopting AlexNet as our baseline model, we establish a benchmark for comparison with more recent and sophisticated architectures. Its relatively simple architecture and well-established training procedures make it a suitable starting point for our experiments.

Additionally, since AlexNet has been extensively studied and widely implemented, it serves as a reference point for evaluating the advancements and performance improvements achieved by newer models.

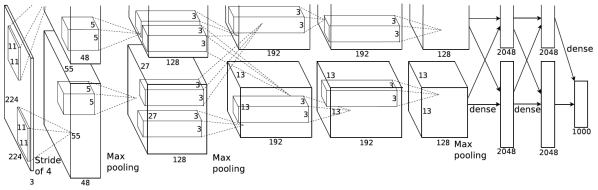

Figure 2: The architecture of AlexNet.

2.3.2. VGG16

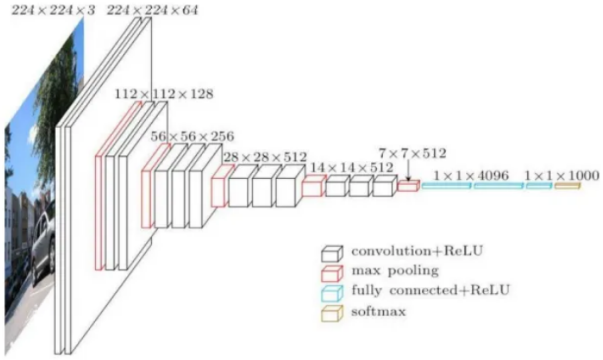

Figure 3: The architecture of VGG16.

The architecture of VGG16 [7] is depicted on Figure 3, and is characterized by its simplicity and effectiveness. It consists of 16 layers, including 13 convolutional layers and 3 fully connected layers, all layers are showd on Figure 4. The convolutional layers are composed of 3x3 filters with a stride of 1 and a padding of 1, followed by max-pooling layers of 2x2 with a stride of 2. This design allows VGG16 to effectively capture hierarchical features from input images while maintaining spatial information.

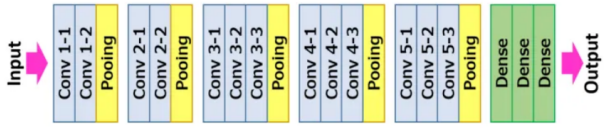

Figure 4: The overall VGG16 layers from input to output.

2.3.3. GoogLeNet

GoogLeNet [8], also known as Inception-v1, is notable for its inception modules, which allow for efficient and parallel processing of information at multiple scales. The architecture is characterized by its depth and computational efficiency, achieved through the innovative use of inception modules, which allow for efficient and parallel processing of information at multiple scales. The architecture is showed on Figure 5.

These modules consist of multiple parallel convolutional layers with different receptive field sizes, including 1x1, 3x3, and 5x5 convolutions, along with max-pooling operations. By incorporating these diverse operations within each inception module, GoogLeNet can capture rich spatial hierarchies of features across different scales.

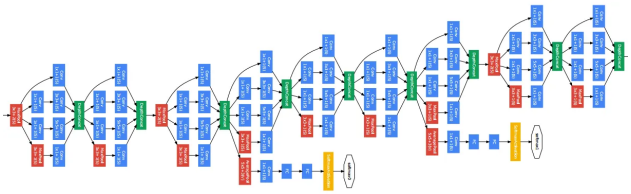

Figure 5: The architecture of GoogLeNet.

2.3.4. ResNet18

The core innovation of ResNet18 lies in the introduction of residual connections, or skip connections, which enable the network to learn residual mappings rather than attempting to directly learn the desired underlying mapping. These skip connections facilitate the flow of gradients during training, mitigating the vanishing gradient problem and allowing for the successful training of very deep networks. ResNet18 specifically consists of 18 layers showed on Figure [3], with a basic building block structure comprising convolutional layers, batch normalization, ReLU activation functions, and residual connections. The network architecture is designed to gradually downsample feature maps while increasing the number of filters, enabling the model to capture increasingly abstract and complex features from input images.

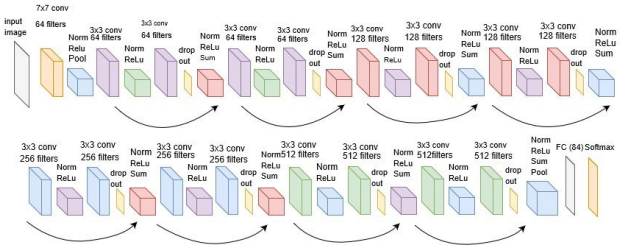

Figure 6: The architecture of ResNet18.

2.4. Experimental Procedure

Before initiating the project, we perform manual processing on the raw data, ensuring it conforms to the required format and integrating it into the dataset for subsequent processing stages. Subsequently, we engage in preprocessing the dataset, which involves three primary phases. Initially, we assign true labels to the images based on the structured file names. Then, employing the train_test_split function, we segment the data into training, testing, and validation subsets,maintaining proportions of 0.6, 0.2, and 0.2, respectively. Finally, we preserve the filename and label information within the annotation.json file for reference and downstream task processing.

2.4.1. Build Baseline Models

After preprocessing, I aim to swiftly establish a rudimentary model to serve as a baseline model for refining and referencing subsequent models. To achieve this, I opt for AlexNet as the initial baseline model for this purpose. Throughout the model establishment process, no adjustments were made to any parameters until the model was fully constructed. Only then are slight adjustments made to the hyperparameters. Subsequently, subsequent models are then trained using these same hyperparameters, facilitating the establishment of baseline models across various CNN architectures, which can be used as benchmarks for future optimization endeavors.

3. Result and Discussion

After applying various data augmentation techniques and modifying the model architectures, the research findings reveal a noticeable improvement in accuracy for most models after optimization. Both during the training and validation phases, it's apparent that all models perform better than AlexNet, achieving accuracies exceeding 70%. Notably, GoogleNet stands out with an accuracy of 82.05%.

However, when tested on a private dataset, nearly all models exhibit a significant decline in classification accuracy, indicating a strong association with evident overfitting. Furthermore, the limited size of the training dataset, comprising fewer than 500 samples, increases the likelihood of encountering issues like early gradient saturation during training.

Table 1: Model accuracies on public and private datasets.

Models	Train/Valid(%)	Test (%)
AlexNet	67.95	64.10
VGG16	76.92	68.59
GoogLeNet	82.05	73.08
ResNet18	78.85	71.79

Analyzing the data in the appendix reveals distinct patterns across different models. In the training process of AlexNet, there are performance plateaus around epoch 40, followed by a potential decline starts to appear. Delve into the data for VGG16, it exhibits promising training curve performance; however, there is a notable rapid increase in validation accuracy early on, suggesting data scarcity issues.

GoogleNet, boasting the highest accuracy, demonstrates relatively stable training but experiences notable validation fluctuations due to limited data, this can be proved from other researches [2]. Contrastingly, ResNet18 fails to converge, with significant validation loss fluctuations, indicating ineffective learning updates possibly due to inappropriate learning rates, resulting in lower final accuracy.

4. Conclusion

In the initial stage, we established baseline models by adjusting hyperparameters. Subsequently, we enhanced data variability through various data augmentation techniques. During training, we experimented with modifying model architectures to improve accuracy and feature extraction capabilities. Despite these efforts, there are still many areas for improvement. For instance, leveraging masked images provided by the original data to extract feature regions for training, or employing a feature extraction approach where classification precedes training, are avenues for further research and development.

Looking ahead, we aim to explore novel approaches to feature extraction and representation learning. This includes investigating methods such as self-supervised learning, where the model learns to predict certain properties of the input data without explicit supervision, and then fine-tuning the model on the downstream task of interest.

Overall, our journey towards improving model performance is ongoing and multi-faceted. By continuously exploring new techniques and refining existing methodologies, we strive to push the boundaries of what is possible in the field of machine learning and pattern recognition.

References

[1] Al-Dhabyani, W., Gomaa, M., Khaled, H., Fahmy, A., 2020. Dataset of breast ultrasound images. Data in Brief 28, 104863. URL: https://www.sciencedirect.com/science/article/pii/S2352340919312181, doi:https://doi.org/10.1016/j.dib.2019.104863.

[2] Gheflati, B., Rivaz, H., 2022. Vision transformer for classification of breast ultrasound images. arXiv:2110.14731.

[3] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. doi:10.1109/CVPR.2016.90.

[4] Ke, A., Ellsworth, W., Banerjee, O., Ng, A.Y., Rajpurkar, P., 2021. Chextransfer: performance and parameter efficiency of imagenet models for chest x-ray interpretation, in: Proceedings of the Conference on Health, Inference, and Learning, Association for Computing Machinery, New York, NY, USA. p. 116–124. URL: https://doi.org/10.1145/3450439.3451867, doi:10.1145/3450439.3451867.

[5] Kornblith, S., Shlens, J., Le, Q.V., 2019. Do better imagenet models transfer better? arXiv:1805.08974.

[6] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[7] Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556.

[8] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9. doi:10.1109/CVPR.2015.7298594.

Appendices

A. AlexNet

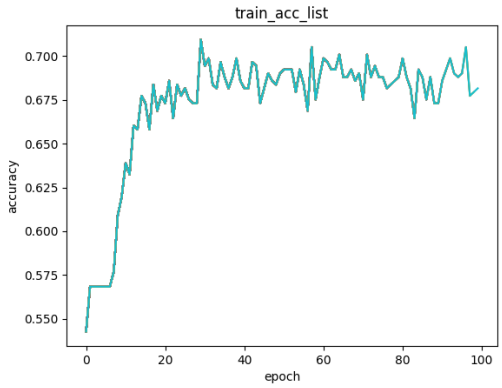

Figure 7: Accuracy curve of AlexNet training process.

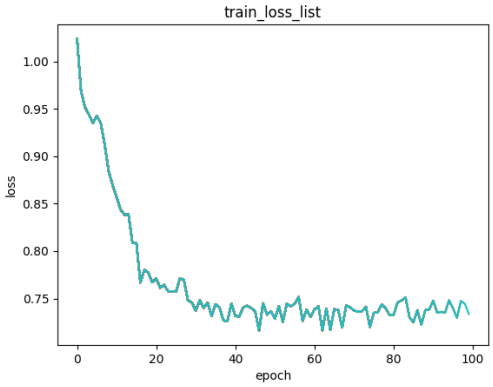

Figure 8: Loss curve of AlexNet training process.

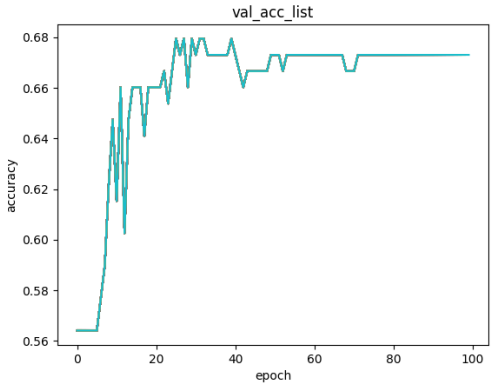

Figure 9: Accuracy curve of AlexNet validation process.

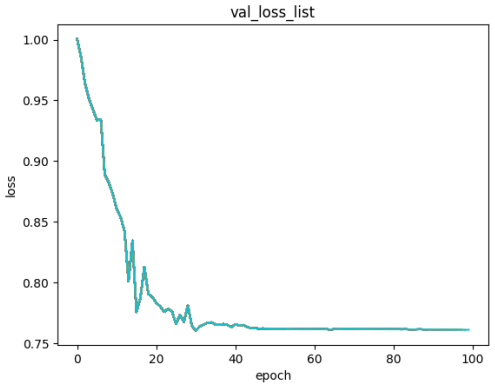

Figure 10: Loss curve of AlexNet validation process.

B. VGG16

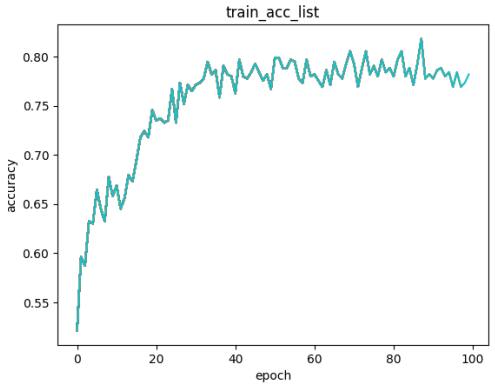

Figure 11: Accuracy curve of VGG16 training process.

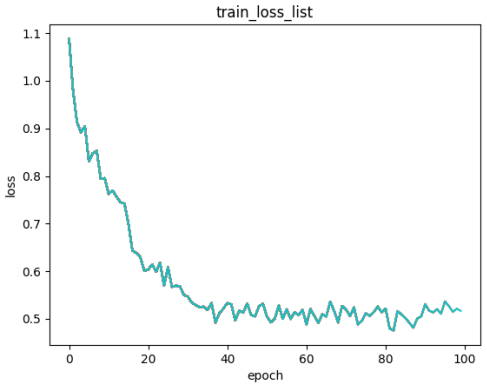

Figure 12: Loss curve of VGG16 training process.

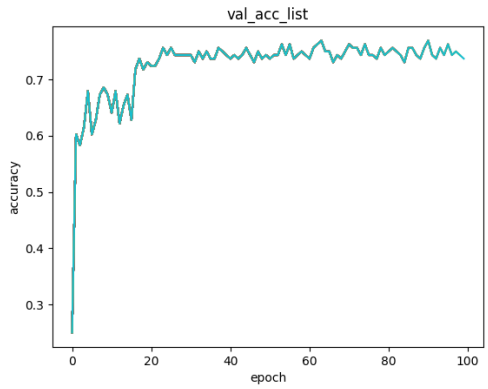

Figure 13: Accuracy curve of VGG16 validation process.

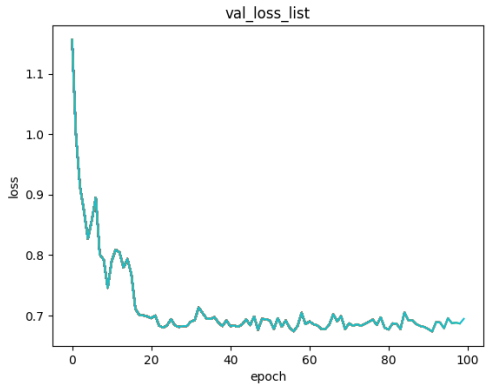

Figure 14: Loss curve of VGG16 validation process.

C. GoogLeNet

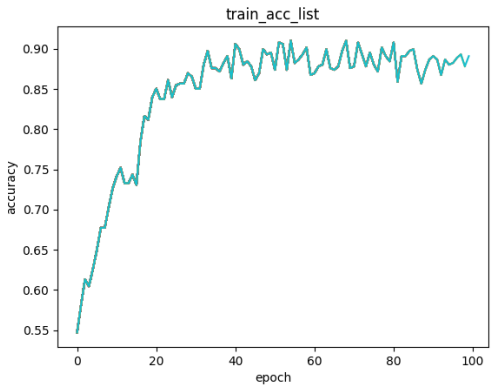

Figure 15: Accuracy curve of GoogLeNet training process.

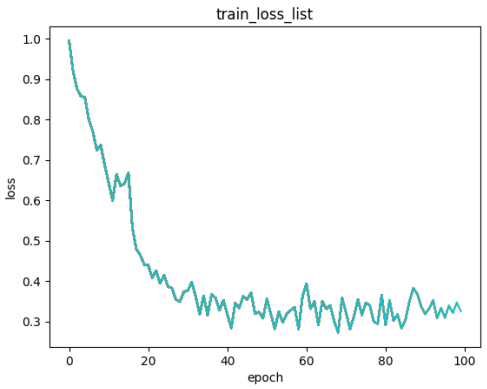

Figure 16: Loss curve of GoogLeNet training process.

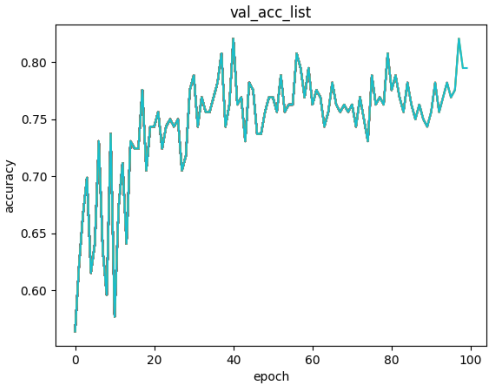

Figure 17: Accuracy curve of GoogLeNet validation process.

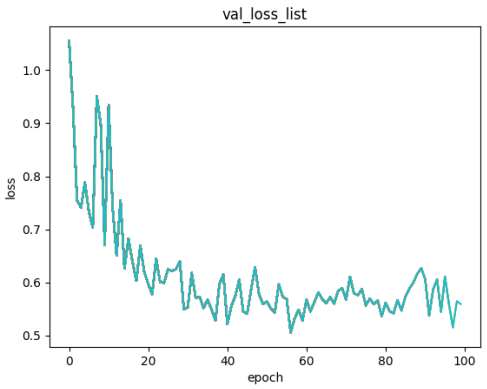

Figure 18: Loss curve of GoogLeNet validation process.

D. ResNet18

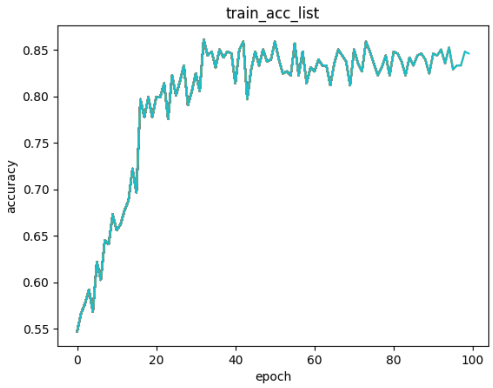

Figure 19: Accuracy curve of ResNet18 training process.

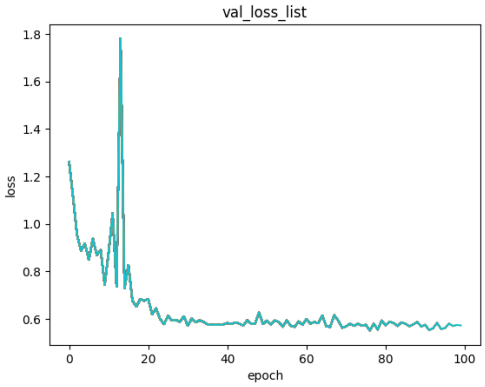

Figure 22: Loss curve of ResNet18 validation process.

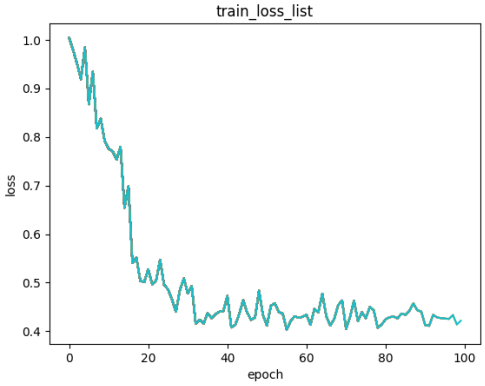

Figure 20: Loss curve of ResNet18 training process.

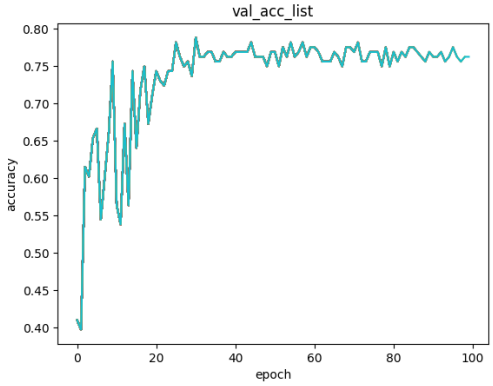

Figure 21: Accuracy curve of ResNet18 validation process.