1. Introduction

Using ResNet18, ResNet50, ResNet152 to analysis acute lymphoblastic leukemia’s classification problem, which has two class labels: 0, 1(Normal, Leukemia blast)

1. Implementation Details
2. The details of your model:

ResNet18:

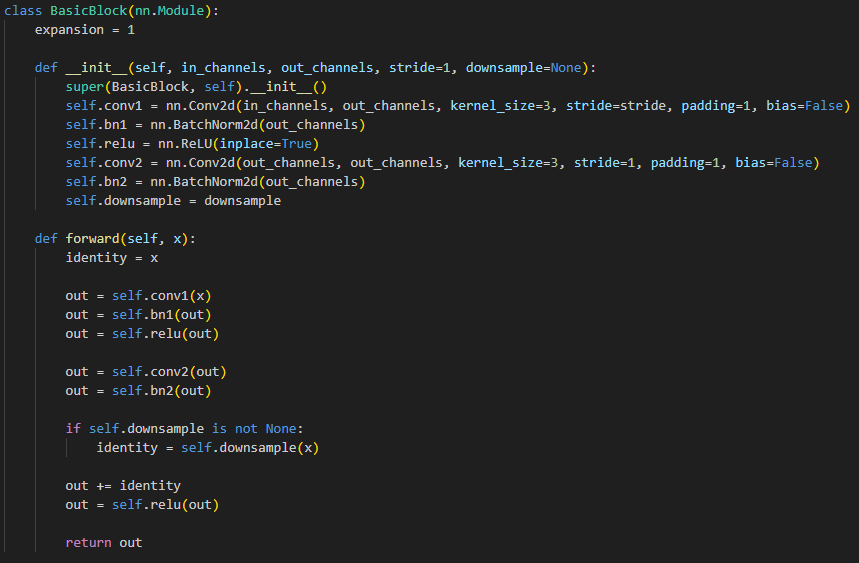
**Input Layer**: Accepts input images, typically three-channel color images with dimensions of 224x224 pixels.

**Convolutional and Pooling Layers**: The initial layers of ResNet-18 consist of convolutional and pooling layers for feature extraction.

First Layer: Convolutional layer with 64 filters of size 7x7, a stride of 2, and padding of 3, producing 64 feature maps.

Pooling Layer: Max-pooling layer with a pool size of 3x3 and a stride of 2.

**Basic Blocks**: ResNet-18 is primarily composed of 4 residual blocks, each containing multiple convolutional layers. Each residual block consists of two 3x3 convolutional layers, each followed by batch normalization and a ReLU activation function. The first residual block includes an additional 1x1 convolutional layer to adjust the channel dimensions to match the output of subsequent residual blocks. Residual connections are used to add the input of each residual block to its output, creating skip connections that facilitate gradient propagation and model training.

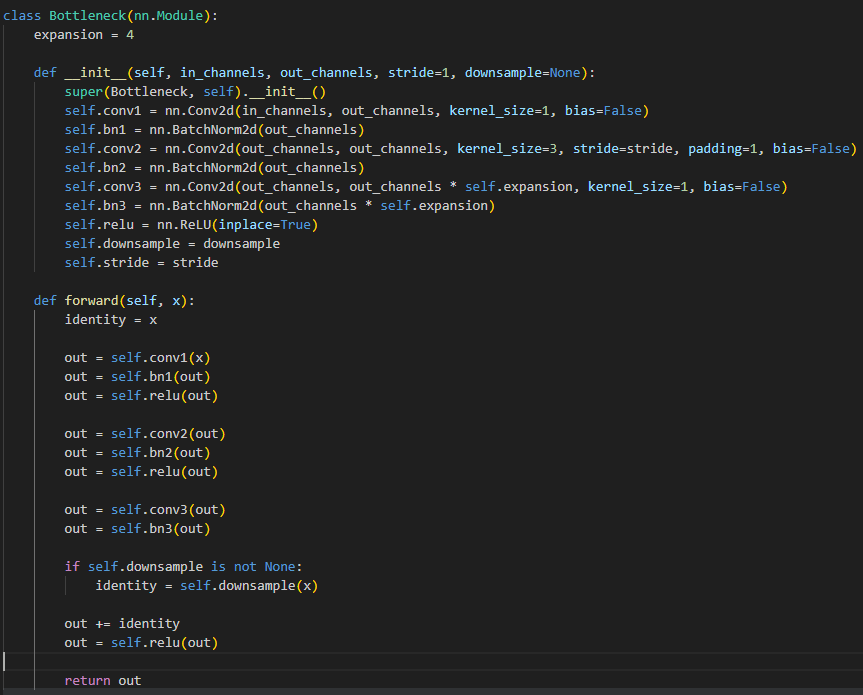


**Global Average Pooling Layer**: After the last residual block, ResNet-18 employs global average pooling to reduce spatial dimensions and transform feature maps into vectors.

**Fully Connected Layer**: The architecture concludes with a fully connected layer that maps the final feature vector to class label predictions.

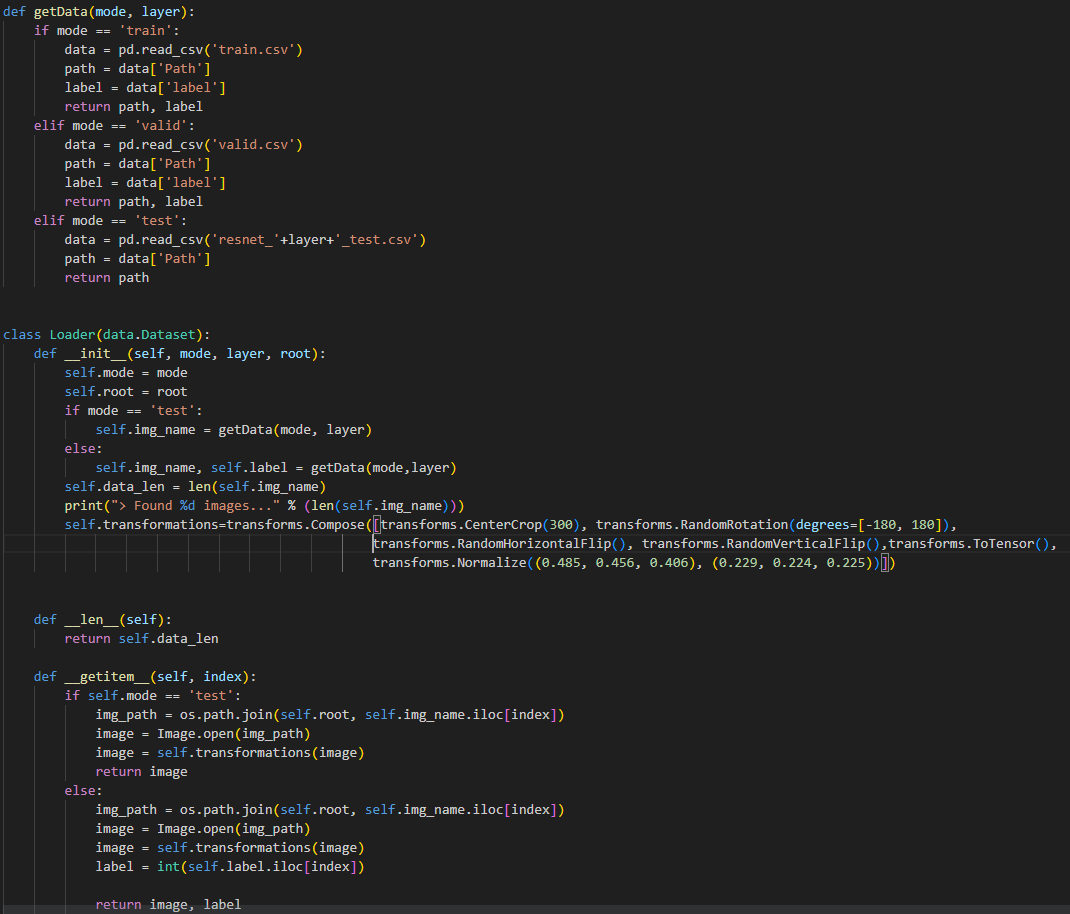
ResNet50, ResNet152:

**Bottleneck Blocks**: ResNet-50 is composed of multiple residual blocks, each containing convolutional layers and skip connections. Each residual block includes multiple 3x3 convolutional layers, batch normalization, and ReLU activation functions. The first convolutional layer in each block may have a stride other than 1 to reduce spatial dimensions. The number of convolutional layers in each block increases as the network becomes deeper. Residual connections add the input of each block to its output, allowing for gradient flow and training of deep networks.



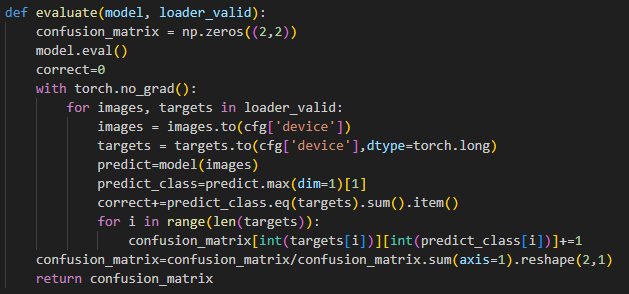
1. The details of your Dataloader

To create dataset and use it with a DataLoader in PyTorch, we have to define a custom dataset class that inherits from torch.utils.data.Dataset. Within this class, we define the \_\_init\_\_, \_\_len\_\_, and \_\_getitem\_\_ methods to load and process data. We can also apply data augmentation using transforms from the torchvision.transforms.

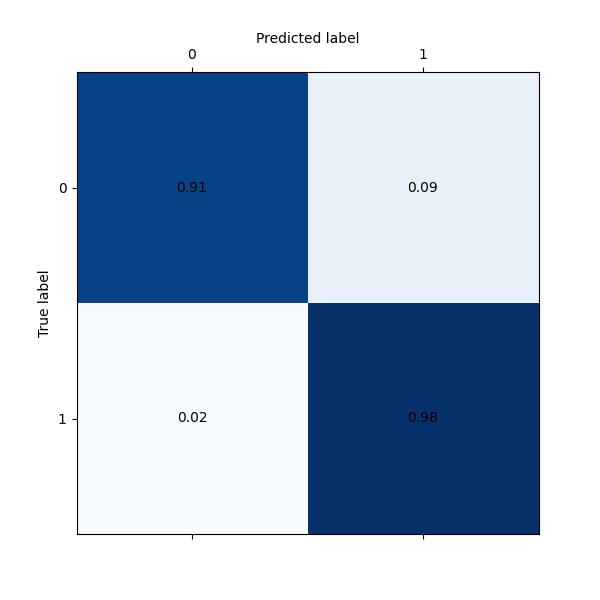
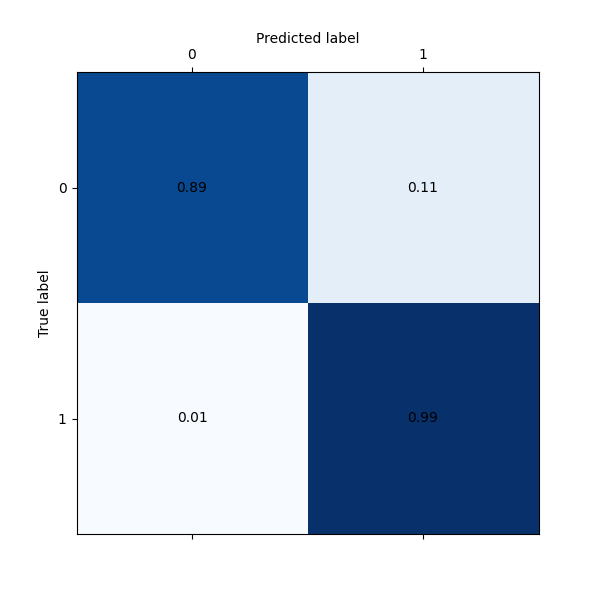


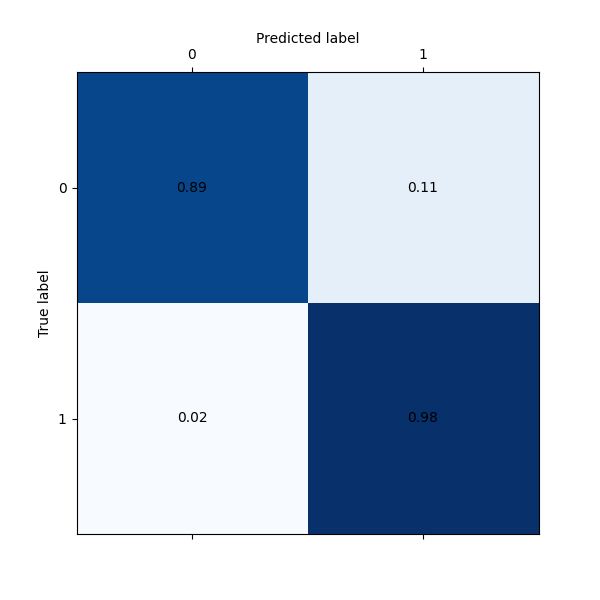
1. Describing your evaluation through the confusion matrix

In validating data, utilize a 2x2 matrix to aggregate counts and subsequently normalize along each row.



Due to the observation of a high False Positive (FP) value in the results, I increased the number of epochs from 80 to 105 to extend the training duration. Additionally, I incorporated data augmentation to generate similar yet distinct training samples. As a result of these changes, the FP value decreased from approximately 0.2 to around 0.1.

ResNet18: ResNet50:

ResNet152:

1. Data Preprocessing
2. How you preprocessed your data

Data augmentation transform refers to a series of operations applied to input data in order to create variations of the original data. These variations help enhance the model's ability to generalize and improve its performance.

**CenterCrop**: Crops the input image to a specified size while keeping the center of the image intact.

**Random Rotation**: Apply a random rotation to the image to introduce variability in object angles.

**RandomHorizontalFlip**/ **RandomVerticalFlip**: Flips the input image horizontally/ vertically with a certain probability (usually 0.5). This can help the model learn from both original and horizontally/ vertically mirrored versions of the same object.

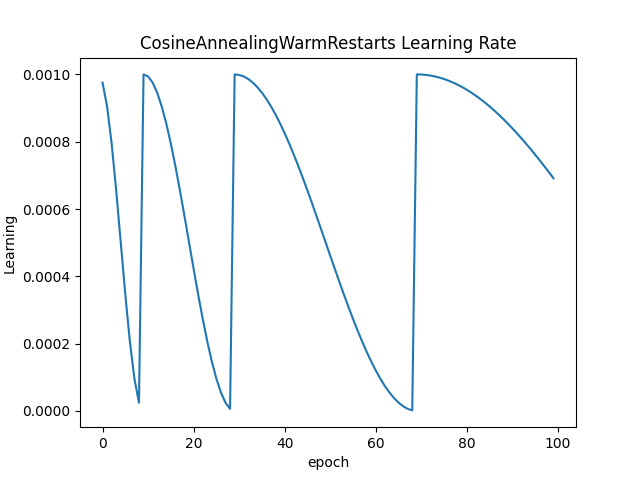
**ToTensor**: Converts the input image (in the form of a NumPy array or a PIL image) into a PyTorch tensor. This transform also scales the pixel values to the range [0, 1].

**Normalize**: Adjusts the pixel values of the input image tensor by subtracting the mean and dividing by the standard deviation. This helps ensure that the pixel values have a standardized distribution, which can aid in training convergence and optimization.

1. What makes your method special

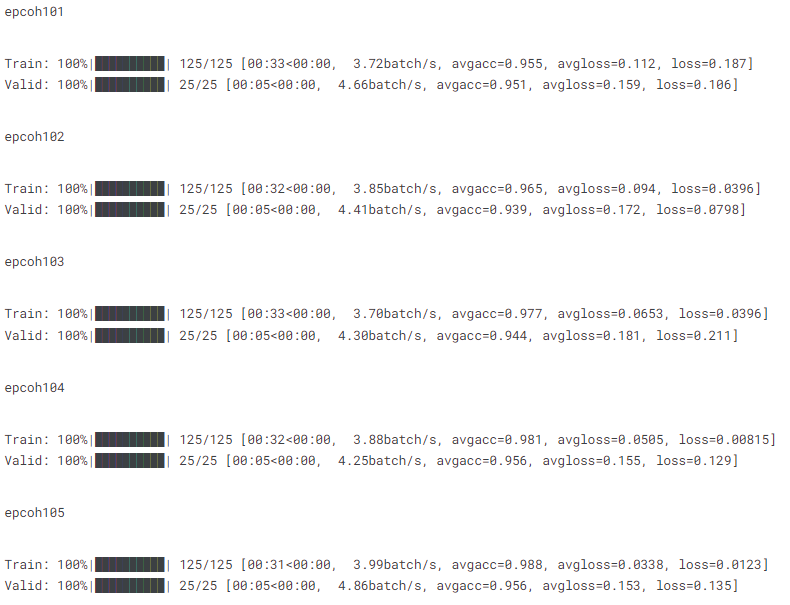
**AdamW**: an extension of Adam that includes a weight decay term in the update step, which helps prevent the model's weights from growing too large during training and can lead to better generalization. Weight decay is a regularization technique used to prevent overfitting by adding a penalty term to the loss function based on the magnitudes of the model's weights. It encourages the model to have smaller weights, which can help prevent the model from fitting the noise in the training data.

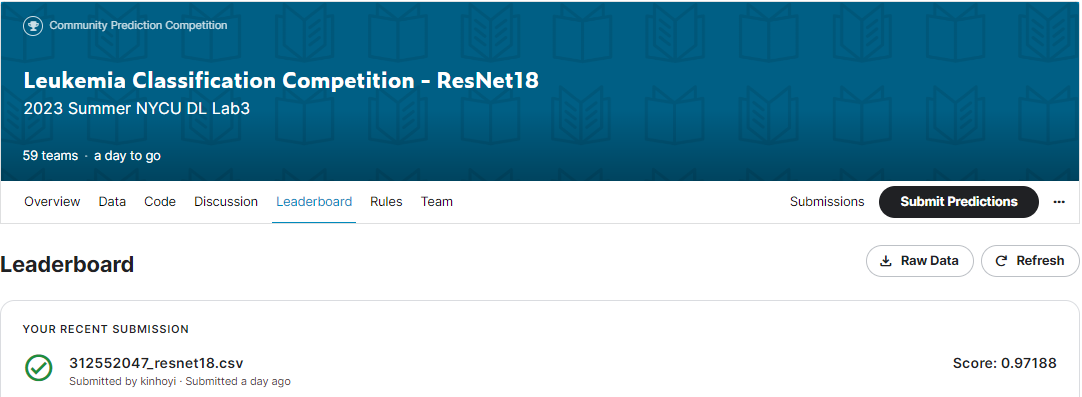
**CosineAnnealingWarmRestarts**: a kind of learning rate scheduling technique commonly used during the training of neural networks. It's an extension of the cosine annealing learning rate scheduler that incorporates warm restarts. This technique can help improve training convergence and generalization by adjusting the learning rate in a cyclical manner.



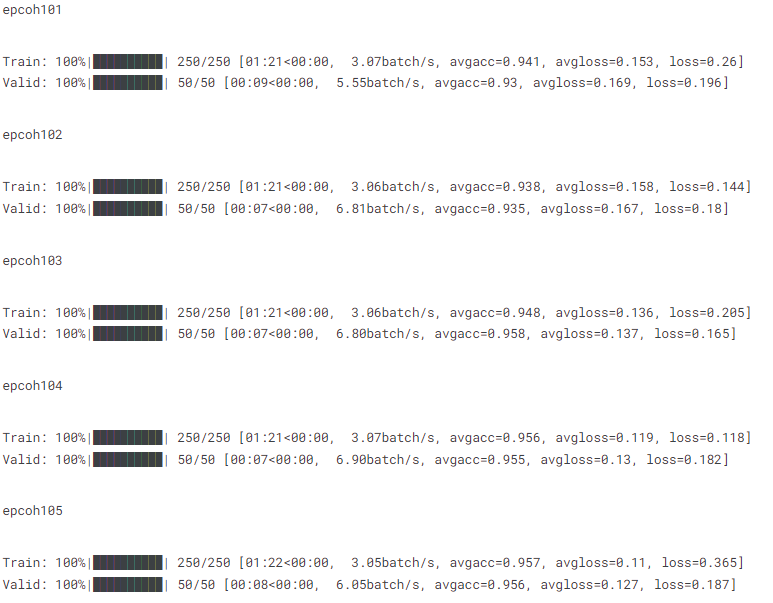
1. Experimental results
2. The highest testing accuracy

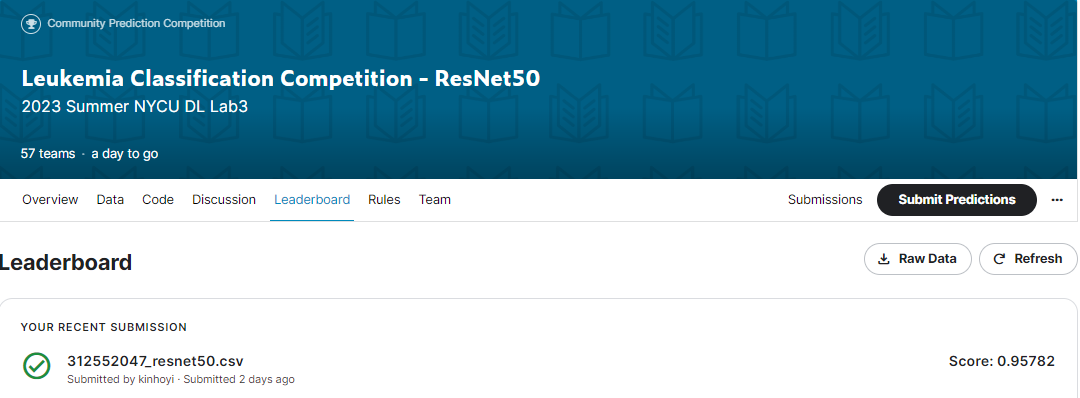
ResNet18:



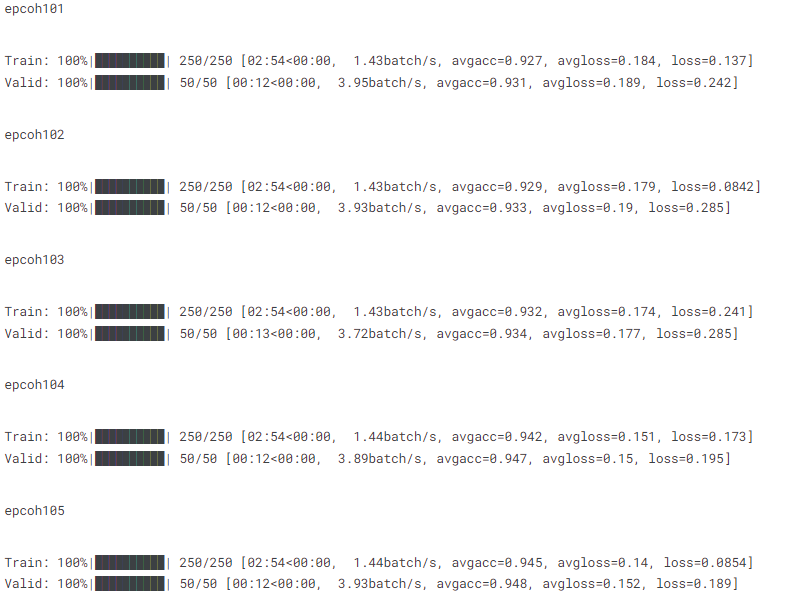


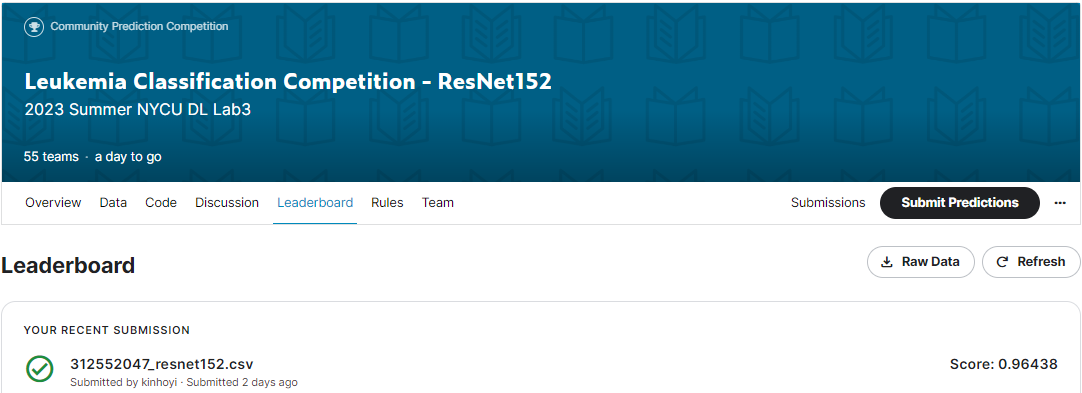
ResNet50:



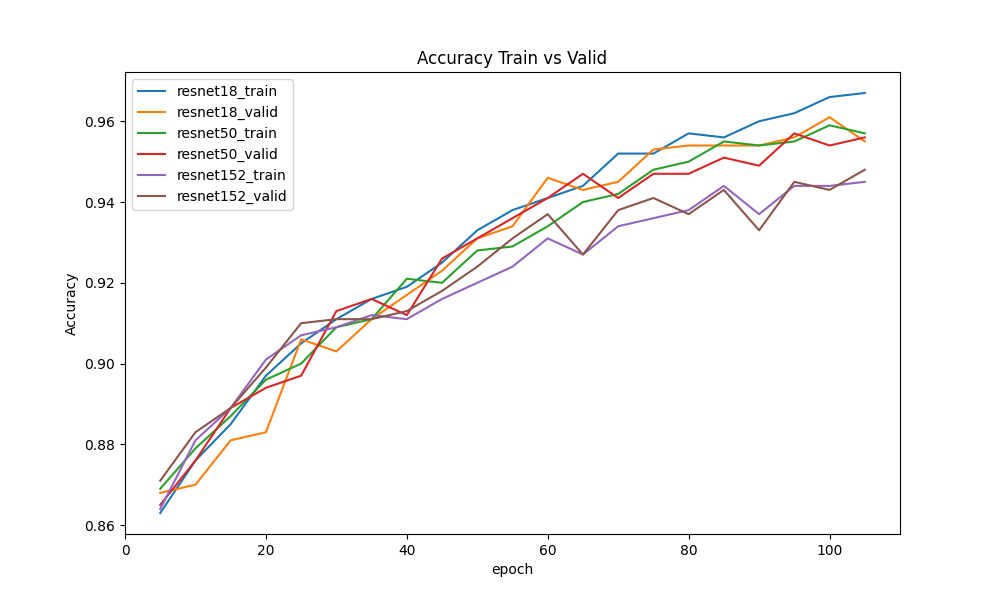


ResNet152:

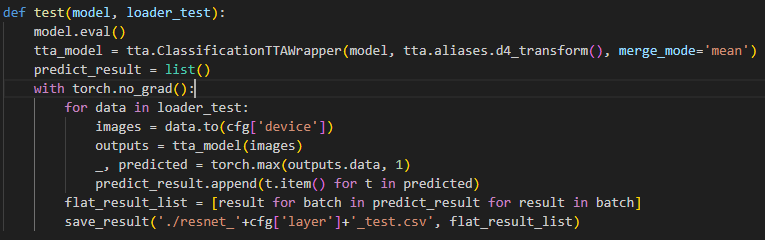




1. Comparison figures



1. Discussion
2. During the training process, it has been observed that certain training methods within the data augmentation not only fail to improve accuracy but sometimes even lead to a reduction in accuracy. Additionally, these methods contribute to increased training time. These methods include GaussianBlur, RandomGrayscale, and others.
3. When training with a fixed learning rate, you may observe a noticeable slowdown in convergence as the training progresses, and there's a possibility of getting stuck in a local minimum. To address this issue, using a learning rate scheduler becomes essential. After implementing a scheduler, it becomes evident that the optimal accuracy is achieved when the learning rate reaches its lowest point.
4. tta.ClassificationTTAWrapper: Test Time Augmentation is a technique used to enhance the performance of trained models. When given an input image during inference, it applies various data augmentation transformations (such as flipping, cropping, scaling, etc.) to the image to create multiple augmented versions. After using Test Time Augmentation, the accuracy has a significant increase of up to 1%.



1. Why not using a softmax layer as the last layer in the ResNet?

The reason for not using a softmax layer as the last layer in the ResNet architecture is that the design philosophy of ResNet is focused on learning richer feature representations, not just the final class probabilities. Placing a softmax layer at the end could restrict the model's feature representation to specific class divisions and overlook deeper-level features.