Name: Tianxiu Ma
Student number: 230277302
module code: ECS659P/ECS7026P

My neural network architecture is as follows:

1. **IntermediateBlock:**

    a. **Repeatedly applies a sequence composed of a convolutional layer, batch normalization, and ReLU activation function** to extract features from images, where the ReLU activation function is applied to the output of the convolutional layer to increase the network's nonlinearity. Without nonlinearity, no matter how many layers the network has, it essentially behaves like a linear model. Batch normalization helps to prevent the vanishing gradient problem, allowing deep networks to train faster and introducing noise to reduce overfitting on the training data.
    b. Uses a **fully connected layer combined with the softmax function** to dynamically adjust the weighted sum of outputs from different convolutional layers, facilitating the model's ability to capture important features in images.

2. **OutputBlock:**

    The extracted features are transformed into class predictions through adaptive **average pooling and two fully connected layers**, where Dropout is introduced to reduce overfitting.

3. **CIFAR10Model：**

    a. The architecture includes **four IntermediateBlocks, each followed by a max-pooling layer** to reduce the spatial dimensions of the feature maps. The inclusion of max-pooling layers allows the network to focus more on the most significant features, such as capturing textures and edge information in images. It enhances the model's robustness to variations in the input data.
    b. **An OutputBlock is used after the last IntermediateBlock** to generate the final class predictions.

The hyperparameters and techniques employed for training:

1. **Batchsize: 128**, this value is moderately sized, allowing for more efficient memory usage as it neither leads to memory overflow nor underutilizes the available memory, thus enhancing computational efficiency.
2. By chaining operations like horizontal **flipping, cropping, and image rotation with PyTorch's transforms.Compose,** the approach improves the model's resilience to

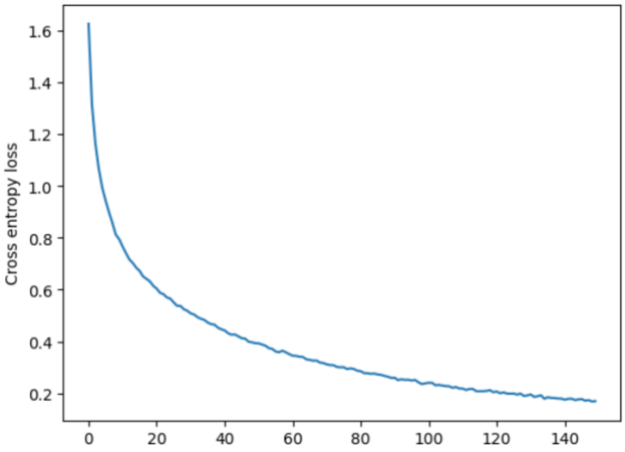
variations in image orientation and broadens the dataset's diversity.

3. In the IntermediateBlock, **the kernel size for each convolutional layer is set to 9, with padding set to 4.** Before this configuration, kernel sizes of 3, 5, and 7 were tested, and it was observed that increasing the kernel size led to a significant improvement in test accuracy. This is because using a larger kernel size can capture more contextual information and makes the model more sensitive to information at the edges of the input image. Conversely, using a smaller kernel size might lead to the loss of information after several convolutional layers.
4. **Learning rate: The default learning rate of 0.001 that comes with the Adam method**. After experimenting with 0.1, 0.01, and 0.0001, it was found that the Adam method's default learning rate of 0.001 performed better.
5. **Optimizer: Adam.** Compared to SGD, Adam is able to converge more quickly.
6. **Number of epochs: 100.** Initially set to 20, it was observed that the accuracy continued to improve steadily, so later trials were conducted with 50 and 80, before finally setting it to 100.

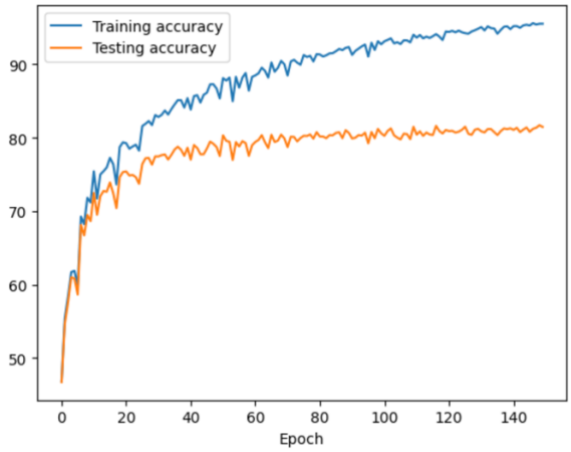
Training and testing accuracy:

The **highest accuracy obtained in the testing dataset is 81.7%.** When training t he model, I set it for 150 epochs. At the 149th epoch, the testing accuracy reache d its highest at 81.7%.

Here is a plot of the loss for each batch:



Here is a plot of the training accuracy and testing accuracy for each epoch:



brief history of how I tried to improve the results of my initial implementation:

1. Within the IntermediateBlock layer, when only containing convolutional layers, the accuracy did not exceed 50%. However, after adding batch normalization and the ReLU function, accuracy increased to over 70%.
2. The problem of model overfitting was mitigated after applying data augmentation.
3. Increasing the number of convolutional layers in the IntermediateBlock from 1 to 6 led to an improvement in accuracy, but increasing from 10 to 20 did not result in a significant change.

Summary:

A good model requires consideration of many factors, such as the number of convolutional layers, the number of IntermediateBlocks, and the choice of optimizer method. It necessitates continuous adjustment of parameters to improve accuracy. Sometimes, hyperparameters that are effective in other models may not necessarily apply to the current model, thus requiring time to be spent on continual training. The current model has a test accuracy of 81.7%, indicating there is still room for improvement.