1. **Parameter explanation**

Num\_conv: a parameter in class block, refers to the layers of convolution.

num\_block: a parameter in class backbone, represents Num\_conv and output channel respectively.

Output: output channel.

1. **Basic model. 🡪test accuracy 0.343**

* batch size=256, in class block, kernel size=5, padding=2, AdaptiveAvgPool2d(1) , Linear(3,num\_conv)---3 is the channel from dataset. nn.Sigmoid().
* in backbone, AdaptiveAvgPool2d(1) , nn.Linear(outputc,256), nn.Linear(256,40),

nn.Linear(40,10), nn.Softmax(dim=1)

* init\_weights: init.xavier\_uniform\_(m.weight), num\_block =((2,64),(2,128),(1,128)), learning rate = 0.01, optimizer = torch.optim.SGD(net.parameters(), lr=lr,momentum=0.9), num\_epochs=24

Parameters are set as above and got loss 2.111, train acc 0.340, test acc 0.343.

1. **Basic model+ batchnorm + relu**🡪**test accuracy 0.593**

Only difference from basic model is batchnorm and relu added in class block within convolution layer as self.add\_module('conv{0}'.format(i),nn.Sequential(nn.Conv2d(inputc, outputc, kernel\_size=5,padding=2),nn.BatchNorm2d(outputc),nn.ReLU(inplace=True))). All other parameters are not changed, results: loss 1.764, train acc 0.700, test acc 0.593.

1. **On 4.3’s model +Maxpool+ dropout+ more linear layers in classifier+ learning rate decrease + Adam optimizer + more blocks(differences are highlighted in yellow)**🡪 **test accuracy 0.664**

* batch size=256, in class block, self.add\_module('conv{0}'.format(i),nn.Sequential(nn.Conv2d(inputc, outputc, kernel\_size=5,padding=2),nn.BatchNorm2d(outputc),nn.ReLU(inplace=True),**nn.MaxPool2d(kernel\_size=2, stride=2)**)), AdaptiveAvgPool2d(1) , Linear(3,num\_conv)---3 is the channel from dataset. nn.Sigmoid().
* in backbone, AdaptiveAvgPool2d(1) , **self.drop=nn.Dropout(0.5,inplace=True),** ,nn.Linear(outputc,256),**nn.ReLU(),nn.Linear(256,128),nn.ReLU(),nn.Linear(128,64),nn.ReLU(),nn.Linear(64,32),nn.ReLU(),nn.Linear(32,10))**, nn.Softmax(dim=1)
* init\_weights: init.xavier\_uniform\_(m.weight), **num\_block** **((2,64),(2,64),(2,128),(2,128),(2,256)),**learning rate = **0.001**, **optimizer = torch.optim.Adam(net.parameters(),lr=lr),** num\_epochs=30.

With parameters changed as above, results are loss 1.730, train acc 0.730, test acc 0.664.

1. **on 4.4 model+ adjust num\_block🡪test accuracy** **0.729**

Based on 4.4 model, the only change made was **num\_block = ((2,64),(2,64),(2,64),(2,128))**,and it turns out higher accuracy.but if I use more (2,64) blocks the model would meet error. Running this got results: loss 1.652, train acc 0.808, test acc 0.729.

1. **on 4.5 model+ more linear layers in class block🡪test accuracy 0.740**

In class block, linear layer added more layers as self.linear=nn.Sequential(nn.Linear(inputc,32),nn.ReLU(),nn.Linear(32,16),nn.ReLU(),nn.Linear(16,num\_conv)),other parameters are not changed ,results came out with higher accuracy, loss 1.641, train acc 0.820, test acc 0.740.

1. **On 4.6 model + more epochs🡪 test accuracy 0.770**

Based on 4.6 model, epochs changed into 100, the accuracy improved. Results: loss 1.590, train acc 0.871, test acc 0.770.

1. **On 4.6 model + changed transforms + learning rate decreased to 0.0005 + 302 epochs🡪 test accuracy 0.825**

Models among 4.2-4.7 all with transforms ([transforms.ToTensor(),transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]), but the accuracy keeps staying under 0.8, refer to codes[3], transforms changed into transforms.RandomCrop(32, padding=4), transforms.RandomHorizontalFlip(), transforms.ToTensor(),transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),learning rate decreased to 0.0005 and 302 epochs. Due to the accuracy at 300 epoch is lower than the accuracy at 230, 2 more epochs were run, the final model result loss 1.581, train acc 0.880, test acc 0.825.

|  |  |
| --- | --- |
| Figure 1- epoch from 0-30Chart, line chart  Description automatically generated | Figure 2 epoch from 31-130  Chart  Description automatically generated |
| Chart  Description automatically generated  Figure 3 epoch from 131 to 230 | Chart, line chart  Description automatically generated with medium confidence  Figure 4 epoch from 231 to 300 |

Chart

Description automatically generated Figure 5 final 2 epochs

# 5. Final model accuracy on CIFAR-10 Validation Set

Final model accuracy I chose to use the function evaluate\_accuracy\_gpu in my\_utils file, the final test accuracy is 0.8258.

Graphical user interface, text, application

Description automatically generated

# Reference

1. TensorFlow, "tf.keras.datasets.cifar10.load\_data", TensorFlow API Documentation, 2021. [Online]. Available: <https://www.tensorflow.org/api_docs/python/tf/keras/datasets/cifar10/load_data>.
2. J. Brownlee, "How to Develop a Convolutional Neural Network From Scratch for CIFAR-10 Photo Classification", Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/.
3. K. Liu, "kuangliu/pytorch-cifar", GitHub, 2021. [Online]. Available: https://github.com/kuangliu/pytorch-cifar.