Neural report

Part 1

Firstly, I called my load_data function with arguments batch size and number of workers which returns the train_loader and test_loader. The training pipeline included data augmentation, this was used to help with generalisation using a method of regularisation to help decrease overfitting. Looking at [1] Showed me how to correctly load and also to include normalize where I researched the best values and came across these which are the ones I use [2] . The augmentations I used where found from here [3] .

Part 2 (more detailed explanation found in the code comments)

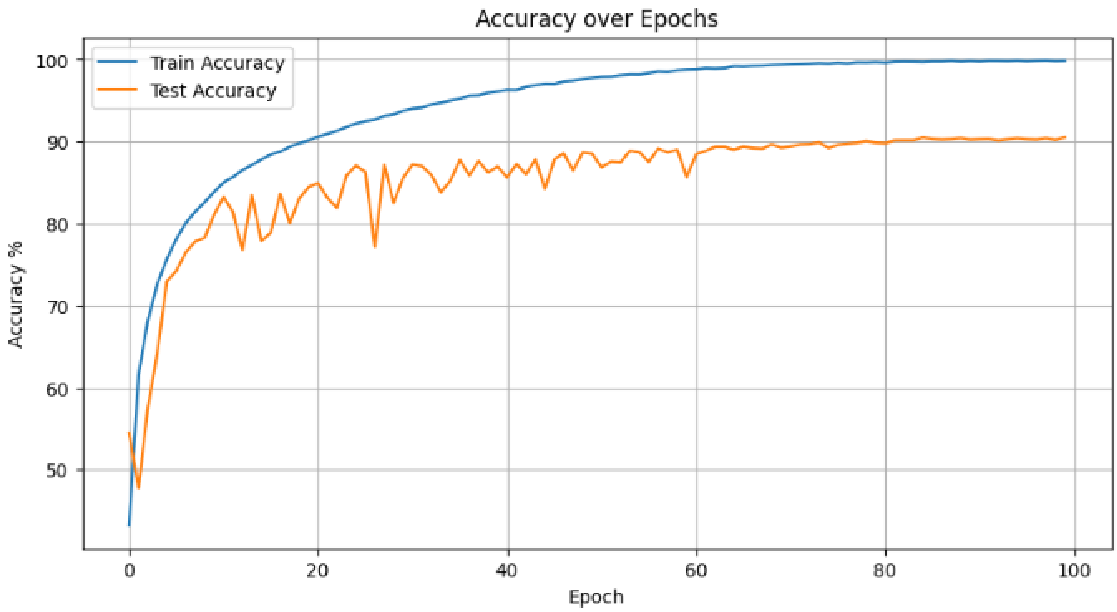
For creating the model, I firstly started by breaking each section down into its own class. The model itself is a class, this contains the initialisation of the stem, backbone and classifier, it returns the output of the classifier after going through the other components. I then create a class for the Stem, the Stem is a convolutional layer, that has BatchNorm to make the output of conv easier to learn from, next adding Relu for non-linearity to make features more complex (inspiration for the stem as a conv layer came from ECS659U TUTORIAL SHEET "conv_example.ipynb"). Then I have a class Backbone which initialises a list of Block modules for the desired amount [4]. Block class is where the calculation for the output of each block happens, it also initialises the ExpertBranch and ConvLayers. ExpertBranch was implemented by following the CW guidelines (inspired by "conv_example.ipynb"). Next is the conv layers also implemented as CW guidelines say, but I found that adding BatchNorm and relu produced higher accuracies for me due to reasons stated earlier. Lastly the output is passed to the classifier where the required implementation is done, however SoftMax is done in the train_one_epoch method.

Part 3

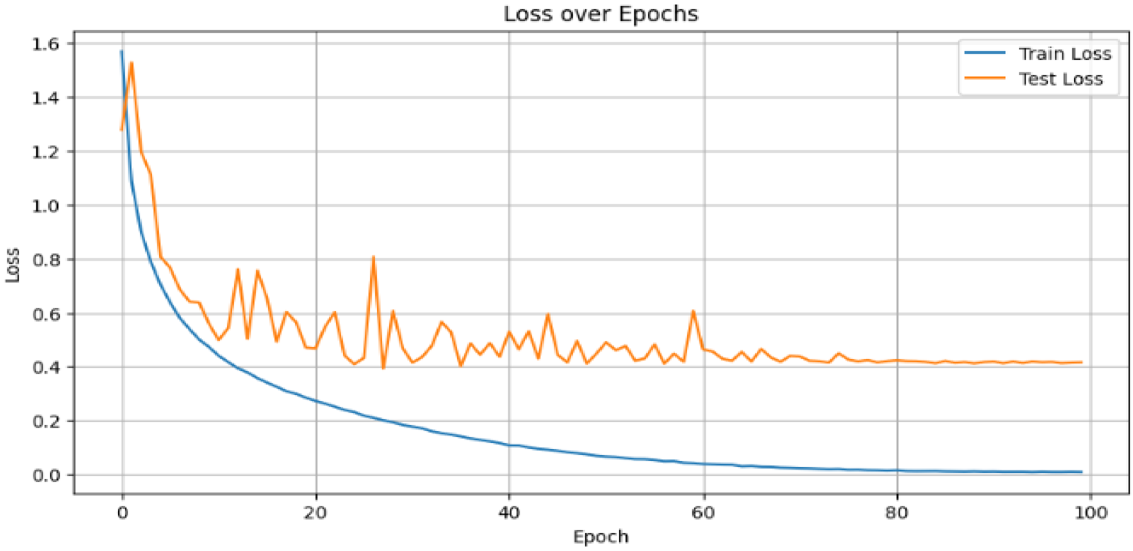
For my loss I used Cross Entropy and for my optimiser I used AdamW these were implemented in my get_lossfunc_optimiser_scheduler.

Part 4

The curves for the evolution of training and validation



The curves for the evolution of loss



All Training details including hyper-parameters used

<u>num_blocks = 8</u>
This refers to the number of blocks within my backbone, each block produces an output / feature map this is used as the next blocks input. This controls the depth of my network going beyond the initial stem, increasing the extraction of high-level features from an input image in the batch. I found that decreasing the number of blocks to 3-5 range makes my time per epoch much higher, however making accuracy much lower. Compared to increasing the number of blocks I found I get a higher accuracy on both training and testing, however it takes much longer per epoch.

<u>output_channels or channels = 64</u>
This hyper parameter was used as the amount of output channels for the stem, each conv layer in the convolution branch, and used as input for the classifier, conv branch and expert branch. This defines the number of channels a feature map has, increasing it means there is more capacity to learn more features. I found that having this too high meant epoch times increased by a large margin. Increasing the number of channels to 128 gave me lower accuracy peaks for my test data, I believe this is because this produced more features making the model memorise instead of generalising, leading to overfitting.

<u>Reduction = 8</u>
This controls how much the channels are reduced by in the expert branch. I initially played around with this value and found that having it too low meant there were more channels which lead to overfitting. I didn't really tune this parameter as much and kept it at 8 the majority of my time trying to improve test accuracy via other parameters. Looking back, I should have increased this reduction as I still have an aspect of overfitting.

<u>K = 3</u>
This determines how many convolutional branches there are in each block, and the output dimensions of the expert branch. I experimented with changing the k value between the ranges 2-5 and found that as I increased the k the initial accuracy also increased, reducing k decreased accuracy but made the time per epoch faster. With a small number of K I also observed that each jump in training per epoch initially was much lower compared to if I increased K to a specific point which was 3. When I set k=2 my model was stuck at a test accuracy of 89%, increasing it to 3 allowed me to achieve 90%.

<u>learning rate = 0.0003</u>

I found that increasing my learning rate also made my initial and final test and train accuracies much lower. Slowly decreasing it after many attempts, I landed on 0.0003 as the right balance to help me achieve 90%, however this may be too lower as my model gets stuck in the 89% test accuracy for a while .

<u>weight_decay = 0.001</u>

I added a small weight decay to help counteract overfitting I was getting into the mid 50 epochs, this slightly solved the issue.

<u>AdamW, Cross Entropy Loss, Scheduler CosineAnnealingLR</u>

Here I Used AdamW as my optimiser as I heard this pairs well with changing learning rates and decay, whereas Adam doesn't work as well with weight decay [5]. I used CrossEntropyLoss as it's the standard for classification models. I also used Cosine to help adjust the learning rate over time to.

<u>Data augmentations (referenced in part 1)</u>

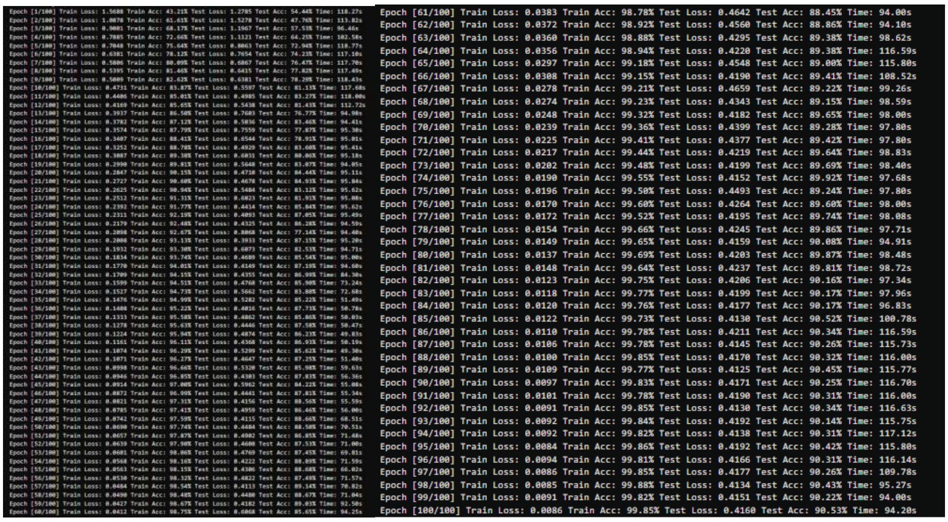
Implementing these allowed to massively counteract memorisation and allow to model to generalise.

<u>Overall</u>

The classification process begins in the Stem class, where low level features from the input images are extracted. This output is then passed into the BackBone class which holds the 'n' number of blocks. Each block has a ExpertBranch and a Conv branch with 'k' conv layers. The expert branch outputs a vector 'a' of length 'k' for each input image. The output of the conv branch is then multiplied by the corresponding 'a_K' and summed. Next this output feature map is passed to the classifier which produces logits for the 10 classes. Calling cross entropy loss applies SoftMax and calculates the probabilities for the logits that sum to 1.

Part 5

Final model accuracy was 90.53% over 100 epochs



```
Epoch [1/100] Train Loss: 1.5688 Train Acc: 43.21% Test Loss: 1.2785 Test Acc: 54.44% Time: 118.27s
Epoch [2/100] Train Loss: 1.0878 Train Acc: 61.61% Test Loss: 1.5278 Test Acc: 47.76% Time: 113.82s
Epoch [3/100] Train Loss: 0.9081 Train Acc: 68.17% Test Loss: 1.1967 Test Acc: 57.53% Time: 96.46s
Epoch [4/100] Train Loss: 0.7885 Train Acc: 72.66% Test Loss: 1.1121 Test Acc: 64.25% Time: 102.58s
Epoch [5/100] Train Loss: 0.7048 Train Acc: 75.64% Test Loss: 0.8063 Test Acc: 72.94% Time: 118.77s
Epoch [6/100] Train Loss: 0.6381 Train Acc: 78.12% Test Loss: 0.7654 Test Acc: 74.23% Time: 117.10s
Epoch [7/100] Train Loss: 0.5806 Train Acc: 80.09% Test Loss: 0.6867 Test Acc: 76.47% Time: 117.70s
Epoch [8/100] Train Loss: 0.5395 Train Acc: 81.46% Test Loss: 0.6415 Test Acc: 77.82% Time: 117.49s
Epoch [9/100] Train Loss: 0.5009 Train Acc: 82.62% Test Loss: 0.6381 Test Acc: 78.29% Time: 118.43s
Epoch [10/100] Train Loss: 0.4731 Train Acc: 83.87% Test Loss: 0.5597 Test Acc: 81.13% Time: 117.68s
Epoch [11/100] Train Loss: 0.4406 Train Acc: 85.01% Test Loss: 0.4985 Test Acc: 83.27% Time: 118.00s
Epoch [12/100] Train Loss: 0.4169 Train Acc: 85.65% Test Loss: 0.5438 Test Acc: 81.43% Time: 112.72s
Epoch [13/100] Train Loss: 0.3937 Train Acc: 86.50% Test Loss: 0.7603 Test Acc: 76.77% Time: 94.98s
Epoch [14/100] Train Loss: 0.3782 Train Acc: 87.12% Test Loss: 0.5036 Test Acc: 83.46% Time: 94.41s
Epoch [15/100] Train Loss: 0.3574 Train Acc: 87.79% Test Loss: 0.7559 Test Acc: 77.87% Time: 95.30s
Epoch [16/100] Train Loss: 0.3407 Train Acc: 88.41% Test Loss: 0.6544 Test Acc: 78.91% Time: 95.01s
Epoch [17/100] Train Loss: 0.3252 Train Acc: 88.78% Test Loss: 0.4929 Test Acc: 83.60% Time: 95.41s
Epoch [18/100] Train Loss: 0.3087 Train Acc: 89.38% Test Loss: 0.6031 Test Acc: 80.06% Time: 95.18s
Epoch [19/100] Train Loss: 0.2990 Train Acc: 89.81% Test Loss: 0.5648 Test Acc: 83.07% Time: 94.05s
Epoch [20/100] Train Loss: 0.2847 Train Acc: 90.15% Test Loss: 0.4710 Test Acc: 84.44% Time: 95.11s
Epoch [21/100] Train Loss: 0.2727 Train Acc: 90.60% Test Loss: 0.4678 Test Acc: 84.93% Time: 95.84s
Epoch [22/100] Train Loss: 0.2625 Train Acc: 90.94% Test Loss: 0.5484 Test Acc: 83.12% Time: 95.62s
Epoch [23/100] Train Loss: 0.2512 Train Acc: 91.31% Test Loss: 0.6023 Test Acc: 81.91% Time: 95.08s
Epoch [24/100] Train Loss: 0.2392 Train Acc: 91.77% Test Loss: 0.4414 Test Acc: 85.84% Time: 95.62s
Epoch [25/100] Train Loss: 0.2311 Train Acc: 92.19% Test Loss: 0.4093 Test Acc: 87.05% Time: 95.49s
Epoch [26/100] Train Loss: 0.2179 Train Acc: 92.48% Test Loss: 0.4325 Test Acc: 86.28% Time: 94.59s
Epoch [27/100] Train Loss: 0.2098 Train Acc: 92.67% Test Loss: 0.8068 Test Acc: 77.14% Time: 94.40s
Epoch [28/100] Train Loss: 0.2008 Train Acc: 93.13% Test Loss: 0.3933 Test Acc: 87.15% Time: 95.20s
Epoch [29/100] Train Loss: 0.1932 Train Acc: 93.30% Test Loss: 0.6073 Test Acc: 82.53% Time: 94.71s
Epoch [30/100] Train Loss: 0.1834 Train Acc: 93.74% Test Loss: 0.4689 Test Acc: 85.54% Time: 95.00s
Epoch [31/100] Train Loss: 0.1770 Train Acc: 94.01% Test Loss: 0.4149 Test Acc: 87.19% Time: 94.60s
Epoch [32/100] Train Loss: 0.1709 Train Acc: 94.15% Test Loss: 0.4355 Test Acc: 86.99% Time: 84.30s
Epoch [33/100] Train Loss: 0.1599 Train Acc: 94.51% Test Loss: 0.4768 Test Acc: 85.90% Time: 73.24s
Epoch [34/100] Train Loss: 0.1527 Train Acc: 94.73% Test Loss: 0.5662 Test Acc: 83.80% Time: 72.68s
Epoch [35/100] Train Loss: 0.1476 Train Acc: 94.99% Test Loss: 0.5282 Test Acc: 85.22% Time: 51.49s
Epoch [36/100] Train Loss: 0.1408 Train Acc: 95.22% Test Loss: 0.4016 Test Acc: 87.73% Time: 50.78s
Epoch [37/100] Train Loss: 0.1333 Train Acc: 95.58% Test Loss: 0.4862 Test Acc: 85.86% Time: 50.03s
Epoch [38/100] Train Loss: 0.1278 Train Acc: 95.63% Test Loss: 0.4446 Test Acc: 87.58% Time: 50.47s
Epoch [39/100] Train Loss: 0.1224 Train Acc: 95.94% Test Loss: 0.4874 Test Acc: 86.23% Time: 49.83s
Epoch [40/100] Train Loss: 0.1161 Train Acc: 96.11% Test Loss: 0.4368 Test Acc: 86.93% Time: 50.19s
Epoch [41/100] Train Loss: 0.1074 Train Acc: 96.29% Test Loss: 0.5299 Test Acc: 85.62% Time: 49.30s
Epoch [42/100] Train Loss: 0.1071 Train Acc: 96.27% Test Loss: 0.4647 Test Acc: 87.25% Time: 51.40s
Epoch [43/100] Train Loss: 0.0998 Train Acc: 96.66% Test Loss: 0.5320 Test Acc: 85.98% Time: 59.63s
Epoch [44/100] Train Loss: 0.0946 Train Acc: 96.85% Test Loss: 0.4303 Test Acc: 87.83% Time: 56.36s
Epoch [45/100] Train Loss: 0.0914 Train Acc: 97.00% Test Loss: 0.5962 Test Acc: 84.22% Time: 55.08s
Epoch [46/100] Train Loss: 0.0872 Train Acc: 96.99% Test Loss: 0.4441 Test Acc: 87.81% Time: 55.34s
Epoch [47/100] Train Loss: 0.0821 Train Acc: 97.31% Test Loss: 0.4156 Test Acc: 88.56% Time: 55.59s
Epoch [48/100] Train Loss: 0.0785 Train Acc: 97.41% Test Loss: 0.4959 Test Acc: 86.46% Time: 56.00s
Epoch [49/100] Train Loss: 0.0742 Train Acc: 97.59% Test Loss: 0.4115 Test Acc: 88.66% Time: 68.51s
Epoch [50/100] Train Loss: 0.0690 Train Acc: 97.74% Test Loss: 0.4484 Test Acc: 88.50% Time: 70.51s
Epoch [51/100] Train Loss: 0.0657 Train Acc: 97.87% Test Loss: 0.4902 Test Acc: 86.85% Time: 71.48s
Epoch [52/100] Train Loss: 0.0639 Train Acc: 97.98% Test Loss: 0.4608 Test Acc: 87.53% Time: 71.00s
Epoch [53/100] Train Loss: 0.0601 Train Acc: 98.06% Test Loss: 0.4769 Test Acc: 87.45% Time: 69.81s
Epoch [54/100] Train Loss: 0.0568 Train Acc: 98.16% Test Loss: 0.4222 Test Acc: 88.89% Time: 71.59s
Epoch [55/100] Train Loss: 0.0563 Train Acc: 98.15% Test Loss: 0.4306 Test Acc: 88.68% Time: 66.02s
Epoch [56/100] Train Loss: 0.0530 Train Acc: 98.32% Test Loss: 0.4822 Test Acc: 87.49% Time: 71.57s
Epoch [57/100] Train Loss: 0.0484 Train Acc: 98.54% Test Loss: 0.4113 Test Acc: 89.14% Time: 70.82s
Epoch [58/100] Train Loss: 0.0490 Train Acc: 98.48% Test Loss: 0.4480 Test Acc: 88.67% Time: 71.04s
Epoch [59/100] Train Loss: 0.0427 Train Acc: 98.67% Test Loss: 0.4182 Test Acc: 89.03% Time: 92.50s
Epoch [60/100] Train Loss: 0.0412 Train Acc: 98.75% Test Loss: 0.6068 Test Acc: 85.65% Time: 94.25s
Epoch [61/100] Train Loss: 0.0383 Train Acc: 98.78% Test Loss: 0.4642 Test Acc: 88.45% Time: 94.00s
Epoch [62/100] Train Loss: 0.0372 Train Acc: 98.92% Test Loss: 0.4560 Test Acc: 88.86% Time: 94.10s
Epoch [63/100] Train Loss: 0.0360 Train Acc: 98.88% Test Loss: 0.4295 Test Acc: 89.38% Time: 98.62s
Epoch [64/100] Train Loss: 0.0356 Train Acc: 98.94% Test Loss: 0.4220 Test Acc: 89.38% Time: 116.59s
Epoch [65/100] Train Loss: 0.0297 Train Acc: 99.18% Test Loss: 0.4548 Test Acc: 89.00% Time: 115.80s
Epoch [66/100] Train Loss: 0.0308 Train Acc: 99.15% Test Loss: 0.4190 Test Acc: 89.41% Time: 108.52s
Epoch [67/100] Train Loss: 0.0278 Train Acc: 99.21% Test Loss: 0.4659 Test Acc: 89.22% Time: 99.26s
Epoch [68/100] Train Loss: 0.0274 Train Acc: 99.23% Test Loss: 0.4343 Test Acc: 89.15% Time: 98.59s
Epoch [69/100] Train Loss: 0.0248 Train Acc: 99.32% Test Loss: 0.4182 Test Acc: 89.65% Time: 98.00s
Epoch [70/100] Train Loss: 0.0239 Train Acc: 99.36% Test Loss: 0.4399 Test Acc: 89.28% Time: 97.80s
Epoch [71/100] Train Loss: 0.0225 Train Acc: 99.41% Test Loss: 0.4377 Test Acc: 89.42% Time: 97.80s
Epoch [72/100] Train Loss: 0.0217 Train Acc: 99.44% Test Loss: 0.4219 Test Acc: 89.64% Time: 98.83s
Epoch [73/100] Train Loss: 0.0202 Train Acc: 99.48% Test Loss: 0.4199 Test Acc: 89.69% Time: 98.40s
Epoch [74/100] Train Loss: 0.0190 Train Acc: 99.55% Test Loss: 0.4152 Test Acc: 89.92% Time: 97.68s
Epoch [75/100] Train Loss: 0.0196 Train Acc: 99.50% Test Loss: 0.4493 Test Acc: 89.24% Time: 97.80s
Epoch [76/100] Train Loss: 0.0170 Train Acc: 99.60% Test Loss: 0.4264 Test Acc: 89.60% Time: 98.00s
Epoch [77/100] Train Loss: 0.0172 Train Acc: 99.52% Test Loss: 0.4195 Test Acc: 89.74% Time: 98.08s
Epoch [78/100] Train Loss: 0.0154 Train Acc: 99.66% Test Loss: 0.4245 Test Acc: 89.86% Time: 97.71s
Epoch [79/100] Train Loss: 0.0149 Train Acc: 99.65% Test Loss: 0.4159 Test Acc: 90.08% Time: 94.91s
Epoch [80/100] Train Loss: 0.0137 Train Acc: 99.69% Test Loss: 0.4203 Test Acc: 89.87% Time: 98.48s
Epoch [81/100] Train Loss: 0.0148 Train Acc: 99.64% Test Loss: 0.4237 Test Acc: 89.81% Time: 98.72s
Epoch [82/100] Train Loss: 0.0123 Train Acc: 99.75% Test Loss: 0.4206 Test Acc: 90.16% Time: 97.34s
Epoch [83/100] Train Loss: 0.0118 Train Acc: 99.77% Test Loss: 0.4199 Test Acc: 90.17% Time: 97.96s
Epoch [84/100] Train Loss: 0.0120 Train Acc: 99.76% Test Loss: 0.4177 Test Acc: 90.17% Time: 96.83s
Epoch [85/100] Train Loss: 0.0122 Train Acc: 99.73% Test Loss: 0.4130 Test Acc: 90.52% Time: 100.78s
Epoch [86/100] Train Loss: 0.0110 Train Acc: 99.78% Test Loss: 0.4211 Test Acc: 90.34% Time: 116.59s
Epoch [87/100] Train Loss: 0.0106 Train Acc: 99.78% Test Loss: 0.4145 Test Acc: 90.26% Time: 115.73s
Epoch [88/100] Train Loss: 0.0100 Train Acc: 99.85% Test Loss: 0.4170 Test Acc: 90.32% Time: 116.00s
Epoch [89/100] Train Loss: 0.0109 Train Acc: 99.77% Test Loss: 0.4125 Test Acc: 90.45% Time: 115.77s
Epoch [90/100] Train Loss: 0.0097 Train Acc: 99.83% Test Loss: 0.4171 Test Acc: 90.25% Time: 116.70s
Epoch [91/100] Train Loss: 0.0101 Train Acc: 99.78% Test Loss: 0.4190 Test Acc: 90.31% Time: 116.00s
Epoch [92/100] Train Loss: 0.0091 Train Acc: 99.85% Test Loss: 0.4130 Test Acc: 90.34% Time: 116.63s
Epoch [93/100] Train Loss: 0.0092 Train Acc: 99.84% Test Loss: 0.4192 Test Acc: 90.14% Time: 115.75s
Epoch [94/100] Train Loss: 0.0092 Train Acc: 99.82% Test Loss: 0.4138 Test Acc: 90.31% Time: 117.12s
Epoch [95/100] Train Loss: 0.0084 Train Acc: 99.86% Test Loss: 0.4192 Test Acc: 90.42% Time: 115.80s
Epoch [96/100] Train Loss: 0.0094 Train Acc: 99.81% Test Loss: 0.4166 Test Acc: 90.31% Time: 116.14s
Epoch [97/100] Train Loss: 0.0086 Train Acc: 99.85% Test Loss: 0.4177 Test Acc: 90.26% Time: 109.78s
Epoch [98/100] Train Loss: 0.0085 Train Acc: 99.88% Test Loss: 0.4134 Test Acc: 90.43% Time: 95.27s
Epoch [99/100] Train Loss: 0.0091 Train Acc: 99.82% Test Loss: 0.4151 Test Acc: 90.22% Time: 94.00s
Epoch [100/100] Train Loss: 0.0086 Train Acc: 99.85% Test Loss: 0.4160 Test Acc: 90.53% Time: 94.20s
```

References (corresponding code can be found by looking at the code comments)

[1] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
[2] https://github.com/kuangliu/pytorch-cifar/issues/19
[3] https://juliusruseckas.github.io/ml/lightning.html
[4] https://pytorch.org/docs/stable/generated/torch.nn.ModuleList.html
[5] https://www.datacamp.com/tutorial/adamw-optimizer-in-pytorch
[6] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
[7] https://pytorch.org/tutorials/beginner/introyt/trainingyt.html
[8] https://discuss.pytorch.org/t/on-running-loss-and-average-loss/107890