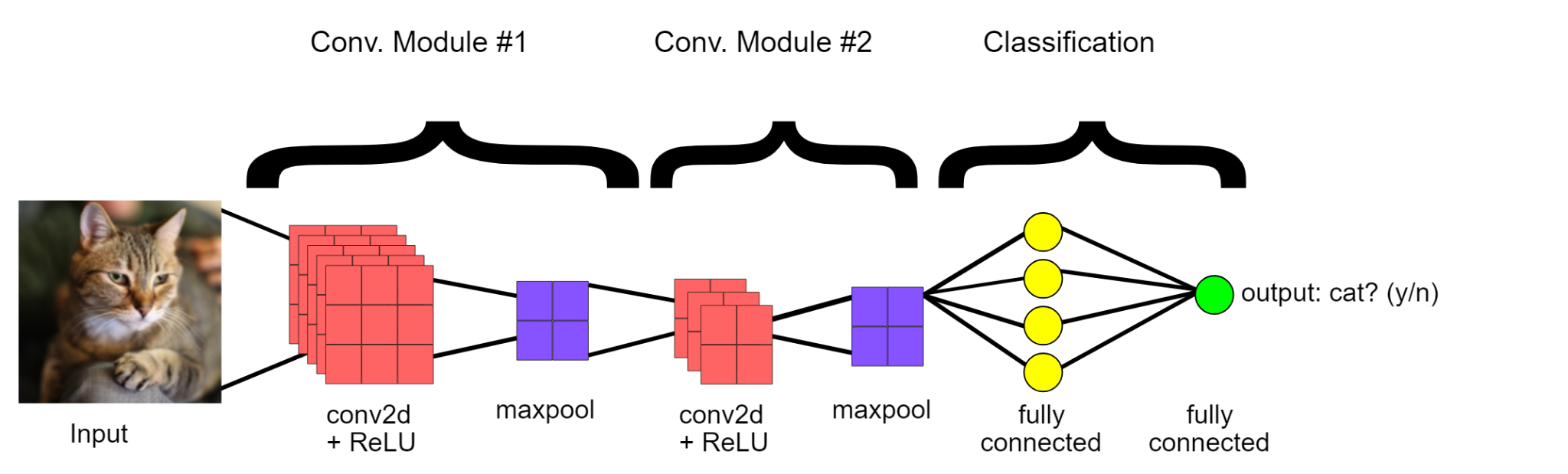
Report:

CNN Model analysis and classification



By: Anthony Barrera

Code:

[**https://github.com/AnthonyBarrera116/CIFAR10-with-96-by-96**](https://github.com/AnthonyBarrera116/CIFAR10-with-96-by-96)

Video Pres:

<https://www.youtube.com/watch?v=D5jLHew9954>

Demo:

<https://youtu.be/a_xK2F-Rhd4>

**Plan:**

* Re run all models using 36 by 36
* Modify code to use 96 by 96
* Make modifications to see at least on photo before running training for 36 by 36
* Make modifications to see at least on photo before running training for 96 by 96
* Study accuracies of 36 by 36 and see best model
* Study accuracies of 96 by 96 and see best model
* Understand how each model works
* Test hypothesis

**Goal:**

Our goals are to get 96 by 96 to run on all models created on the github project, Test 96 by 96 and see if higher pixel count can help improve the model faster. Study how each model works and how they differ from others.

**Hypothesis:**

Normally for humans, the higher quality image helps us determine what we see. If you give a training model more information about a picture, could it help improve the model and converge faster. We know in practice more information can be worse for the model. But with 4k resolutions out we assume with DLSS upscaling etc it would use a resolution of 1080 or 1440 to scale the picture.

**Schedule of tasks done:**

**Sep 4th:**

* Simple DLA 96 x 96
  + Overfit at Epoch: between 150 to 199 → Train: 99% Test: 75%
  + Best at Epoch: 150ish → Train: 98% Test: 73%
* Regnet 96 x 96
  + Overfit at Epoch: between 120 to 132 → Train: 99% Test: 80%
  + Best at Epoch: 120ish → Train: 99% Test: 80%
* Efficient Net 96 x 96
  + Overfit at Epoch: 150 to 199 → Train: 99% Test: 78%
  + Best at Epoch: 150ish → Train:98% Test: 77%
* Shufflenetv2 96 x 96
  + Overfit at Epoch: None (Didn’t work)
  + Best at Epoch: None (Didn’t work)

**Summary**: The best model for 96 by 96 was Regnet But all three models seem to have overfitting issues. They performed great on training but with Test data it seemed to fail. ShuffleNetV2 didn’t work. I never ran and would break.

**Sep 6th:**

* SENet18 96 x 96
  + Overfit at Epoch: None
  + Best at Epoch: 11 → Train: 10% Test: 10%
* DPN92 96 x 96
  + Overfit at Epoch: None (VRAM ran out)
  + Best at Epoch: None (VRAM ran out)
* MobileNetV2 96 x 96
  + Overfit at Epoch: None (VRAM ran out)
  + Best at Epoch: None (VRAM ran out)
* MobileNet 96 x 96
  + Overfit at Epoch: 150 to 199 → Train: 99% Test: 78%
  + Best at Epoch: 150ish → Train:98% Test: 77%

**Summary**: The best model for 96 by 96 was MobileNet. Both DPN92 and MobileNetV2 ran into VRAM usage errors. Mainly due to the size of the image.SENet18 seemed to have a problem to be stuck at 10 percent. I think this is due to the size and the model needed more attention.

**Sep 8th:**

* ResNetrXt29 96 x 96
  + Overfit at Epoch: None (VRAM ran out)
  + Best at Epoch: None (VRAM ran out)
* Dense Net 96 x 96
  + Overfit at Epoch: None (VRAM ran out)
  + Best at Epoch: None (VRAM ran out)
* GoogLeNet
  + Overfit at Epoch: None (VRAM ran out)
  + Best at Epoch: None (VRAM ran out)
* PreActResNet 96 x 96
  + Overfit at Epoch: 5→ Train: 10% Test: 10%
  + Best at Epoch: 0 → Train:10% Test: 10%

**Summary**: There was no good model PreActResNet had the same problem as SENet18. Due to the creation of the model either more parameters needed to change or is not good with higher resolution. ResNetrXT29, Dense and GoogLeNeet all ran into VRAM issues of running out of memory.

**Sep 9th:**

* ResNet18 96 x 96
  + Overfit at Epoch: 150 to 199 → Train: 100% Test: 71%
  + Best at Epoch: 150ish → Train:98% Test: 67%
* VGG 96 x 96
  + Overfit at Epoch: 150 to 199 → Train: 99% Test: 71%
  + Best at Epoch: 150 to 170 → Train:83% Test: 68%

**Summary**: Best model is VGG since its accuracy was higher for tests when accuracy of training was lower. This means when slight improvement the other improved whereas with ResNet18 we had 96 to 67 percent which is where our training was grasping the data as well as 83 to 68 percent.

**Sep 10th:**

* Simple DLA 35 x 35
  + Overfit at Epoch:
  + Best at Epoch: 199 → Train: 99% Test: 94%
* Regnet 35 x 35
  + Overfit at Epoch:
  + Best at Epoch: 199 → Train: 100% Test: 94%
* Efficient Net 35 x 35
  + Overfit at Epoch: 170 - 199 → Train: 99% Test: 85%
  + Best at Epoch: 150ish → Train:92% Test: 81%
* SENet18 35 x 35
  + Overfit at Epoch:
  + Best at Epoch: 199 → Train: 99% Test: 95%
* Shufflenetv2 35 x 35
  + Overfit at Epoch:
  + Best at Epoch: 199 → Train: 99% Test: 91%

**Summary**: The best model was Simple DLA and Regnet having the highest Training and Test accuracy. SENet18 and shufflenetV2 did well too but not as good as simple DLA and Regnet. Efficient Net did okay but could have lead to overfitting.

**Sep 11th:**

* DPN92 35 x 35
  + Overfit at Epoch: None (Test 115 epoch)
  + Best at Epoch: 115→ Train:96% Test: 88%
* MobileNetV2 35 x 35
  + Overfit at Epoch: 1
  + Best at Epoch: 199→ Train: 99% Test: 91%
* MobileNet 35 x 35
  + Overfit at Epoch: Train:98% Test:75%
  + Best at Epoch: None

**Summary**: The best model is MobileNetV2 having a 99 and 91 percent accuracy. DPN did well for just 115 epochs and could have led to better results than MobileNetV2 or similar. MobileNet seemed to have an issue over overfitting where the result of training did not show in Test.

**Sep 12th:**

* ResNetrXt29 35 x 35
  + Overfit at Epoch: None
  + Best at Epoch: 199 → Train: 100% Test: 95%

**Summary**: Only one Model for the day this took very long amount of time unlike others and showed very good performance in accuracy.

**Sep 13th**:

* Dense Net 35 x 35
  + Overfit at Epoch:
  + Best at Epoch: 199→ Train:99% Test: 95%
* PreActResNet 35 x 35
  + Overfit at Epoch:
  + Best at Epoch: 10→ Train:10% Test: 10%
* ResNet18 35 x 35
  + Overfit at Epoch:
  + Best at Epoch: 199→ Train:100% Test: 95%
* VGG 35 x 35
  + Overfit at Epoch:
  + Best at Epoch: 199→ Train:99% Test: 93%

**Summary**: Best models ResNet18 and Dense with best accuracy of 95 percent on test set. PreActResNet had an issue of evening improvement and stuck at 10 percent. VGG did well too but not as well as Dense and ResNet18

**OVERALL:**

The best models for 36 by 36 with the best training and testing accuracies are ResNet18, Dense, ResNetrXt29 and SENet. These models have accuracy of 100 percent for training and 95 percent accuracy for testing. The worst 36 by 36 model is PreActResNet with train accuracy at 10 percent and test accuracy at 10 percent. For the 96 by 96, there wasn’t any good model. All models seemed to train well and have high accuracy but they didn’t show in the test accuracy. THis can be due to the size of resolution and it’s not capturing the more pixels of the image. THe question that a rose was can more pixels help a model capture the understanding of the model and the answer is no. But this doesn’t mean that it can’t be done. There can be several reasons why. It can be due to the complexity where the model needs more layers and more complexity to grasp the 96 by 96. In standard practice the mode pixels and more complex, you’ll need a much more complex model with layers. Another thing can be due to PC resources. When I used 96 by 96 it used all 24 gbs of VRAM and some of my normal Memory.

**GITHUB REPO OF PYTORCH EXAMPLE PROJECT:**

[**https://github.com/kuangliu/pytorch-cifar**](https://github.com/kuangliu/pytorch-cifar)

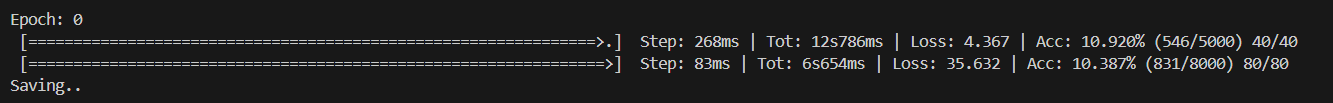
**Modified Code with my comments:**

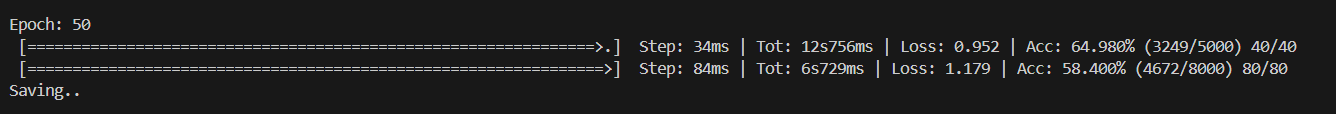
[**https://github.com/AnthonyBarrera116/CIFAR10-with-96-by-96**](https://github.com/AnthonyBarrera116/CIFAR10-with-96-by-96)

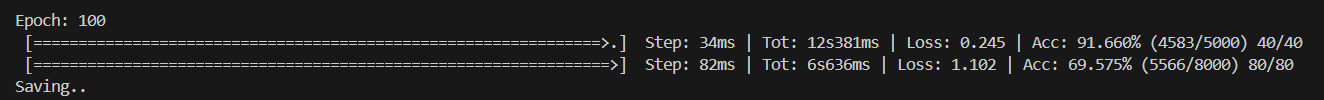
**96 X 96**

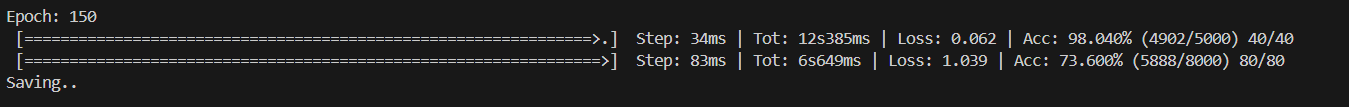
**Sep 4th**

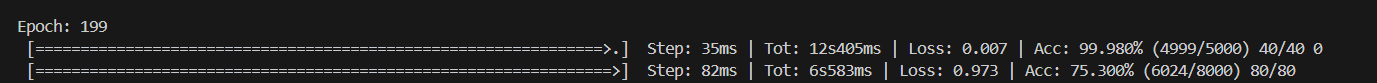
Simple DLA line: self.linear = nn.Linear(4608 , num\_classes)

****

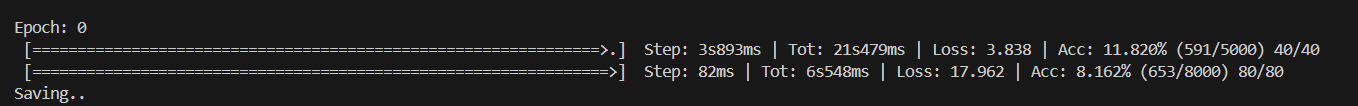


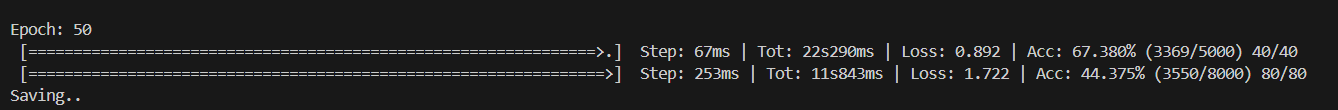


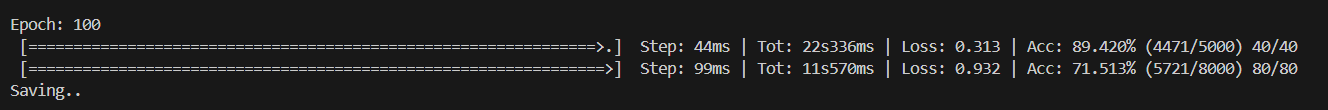


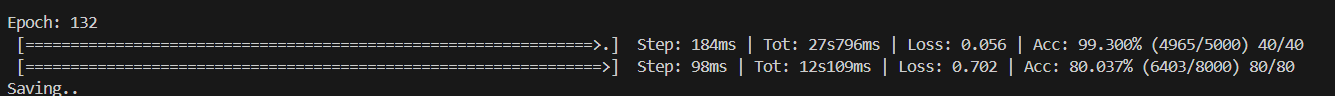


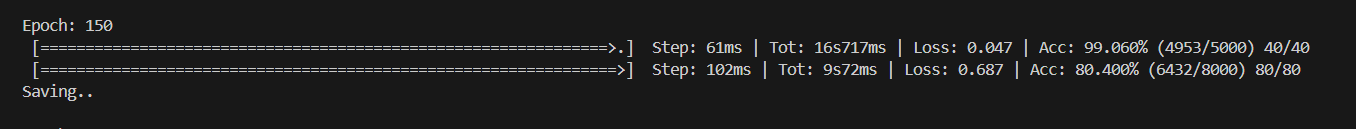
Regnet

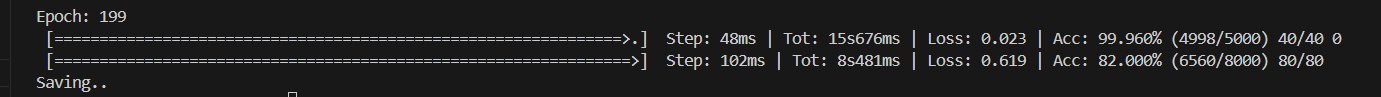




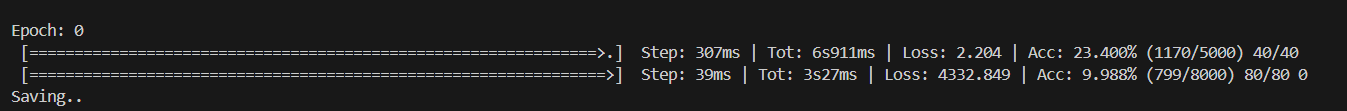


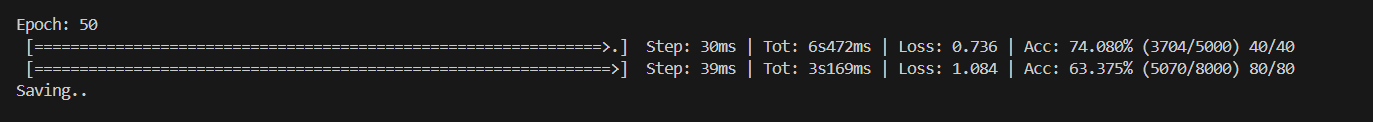


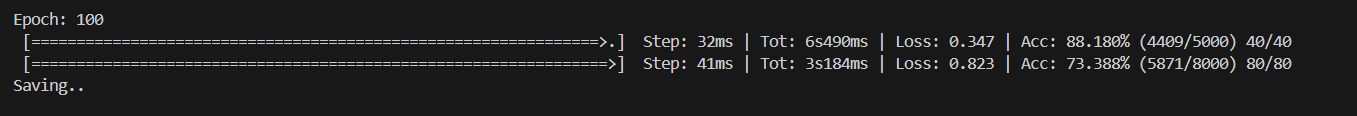


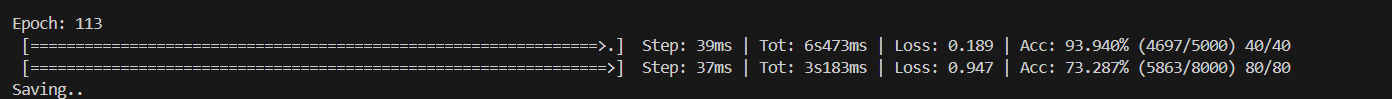


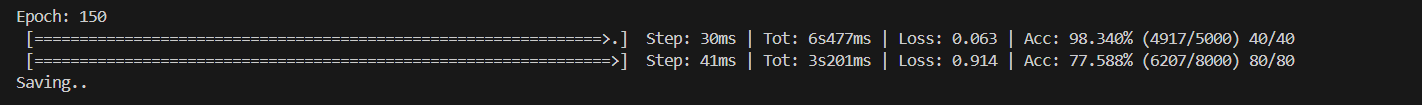
Efficient net

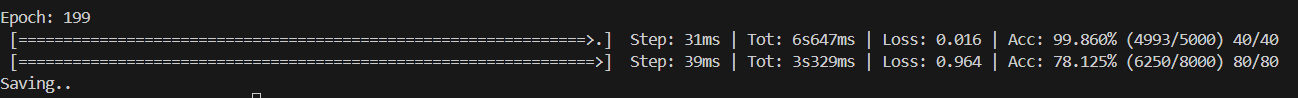










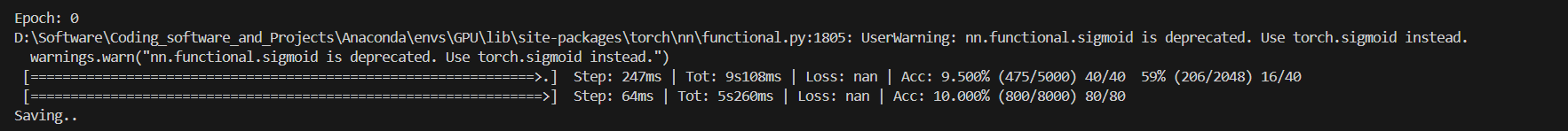


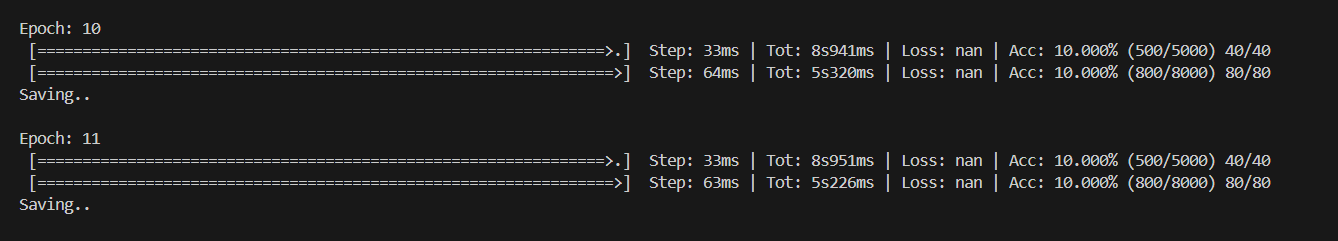
ShuffleNetV2

Didn’t work

**Sep 6th**

SENet18 line: self.linear = nn.Linear(4608, num\_classes)





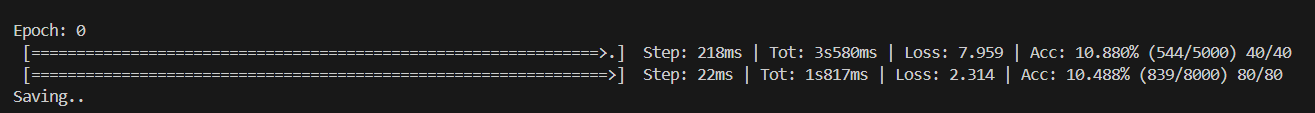
DPN92

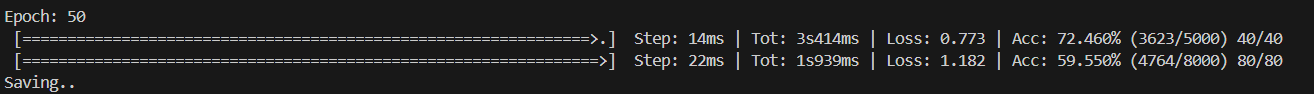
Used 24 gbs VRAM and of NOrmal RAM and ran out of space can’t run

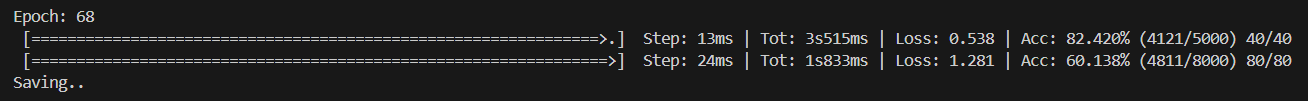
MobileNetv2

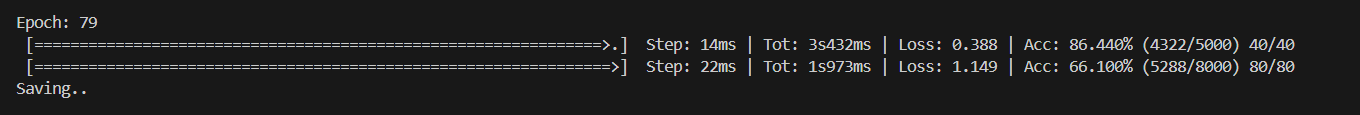
Used 24 gbs VRAM and of NOrmal RAM and ran out of space can’t run

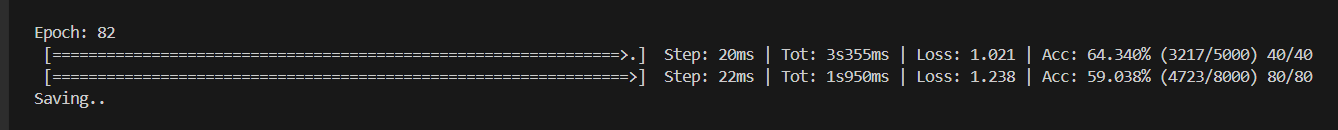
MobileNet line: self.linear = nn.Linear(9216, num\_classes)

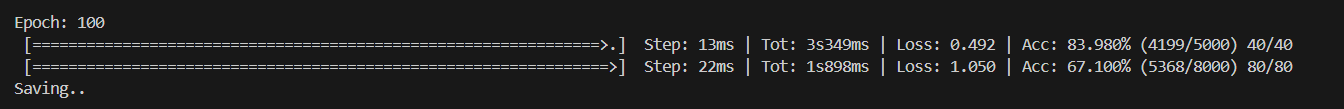


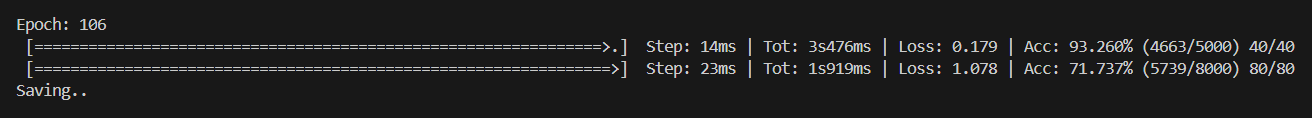


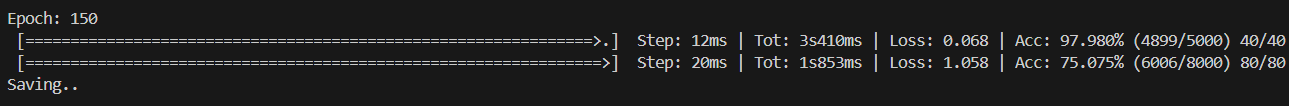


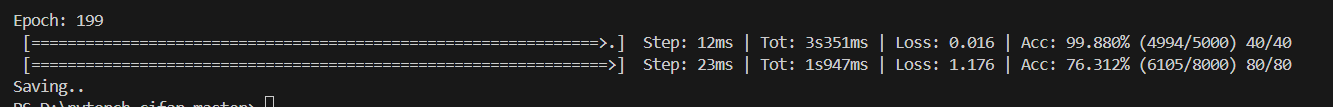












**Sep 8th**

ResNeXt29

Used 24 gbs VRAM and of NOrmal RAM and ran out of space can’t run

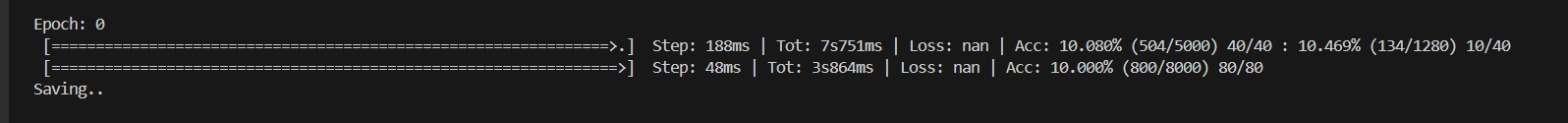
DenseNet

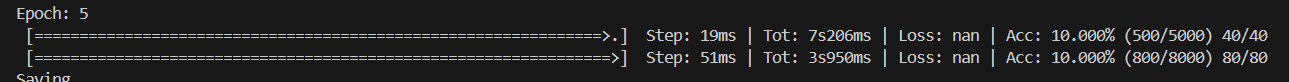
Used 24 gbs VRAM and of NOrmal RAM and ran out of space can’t run

GoogLeNet

Used 24 gbs VRAM and of NOrmal RAM and ran out of space can’t run

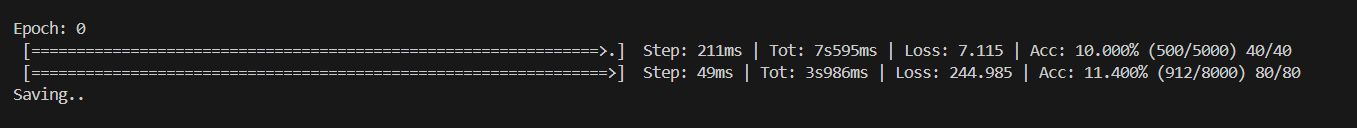
PreActResNet line: self.linear = nn.Linear(4608\*block.expansion, num\_classes)

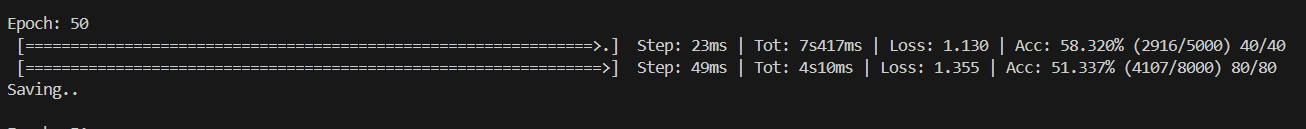


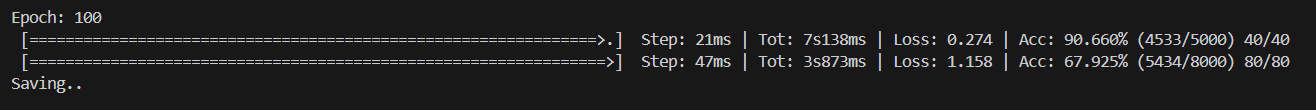


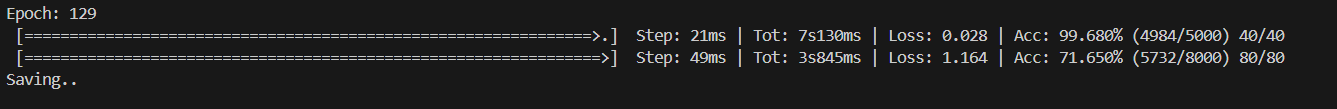
**Sep 9th**

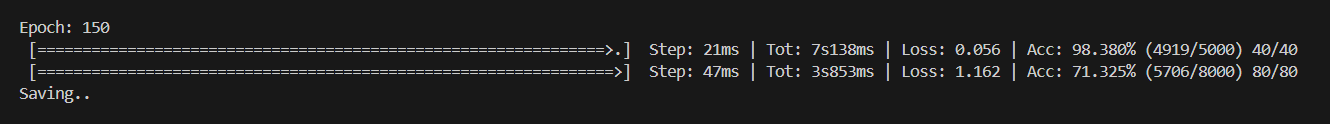
ResNet18 line: self.linear = nn.Linear(4608\*block.expansion, num\_classes)

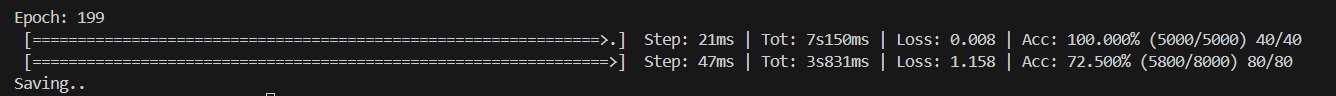




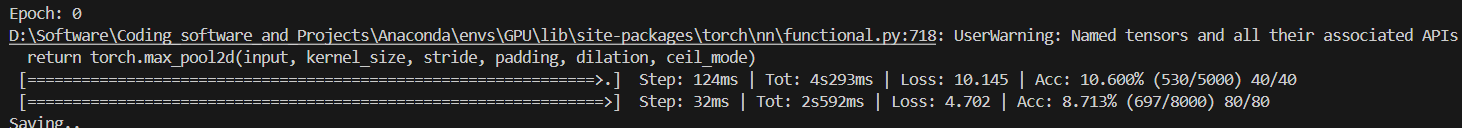


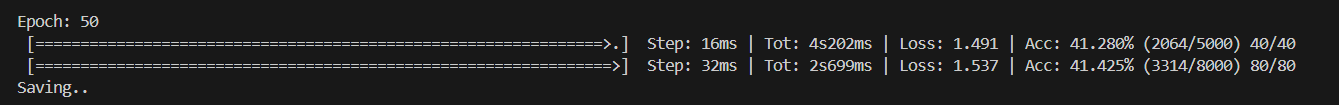


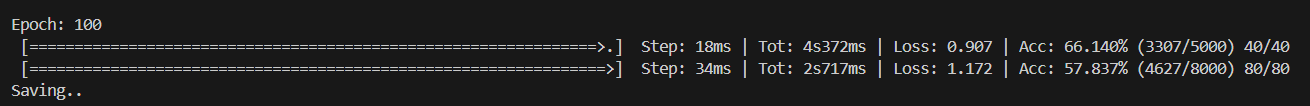


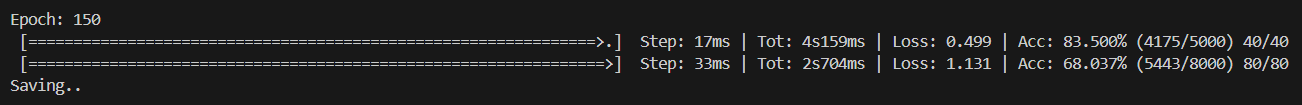


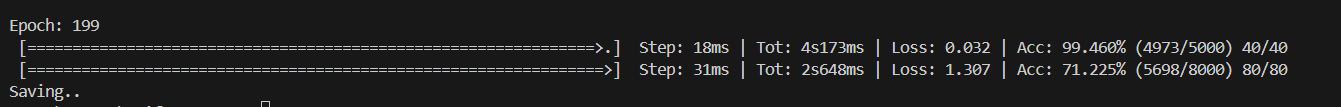
VGG line: self.classifier = nn.Linear(4608, 10)







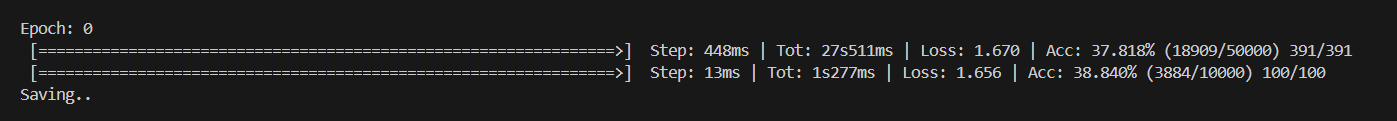


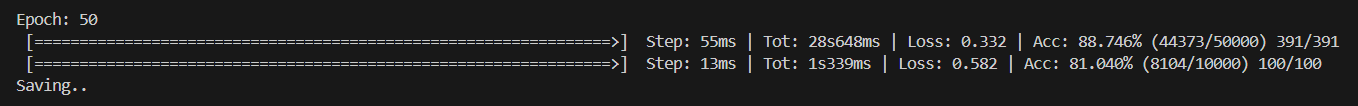


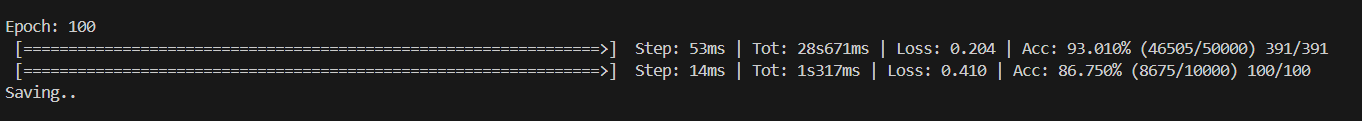
**35 x 35**

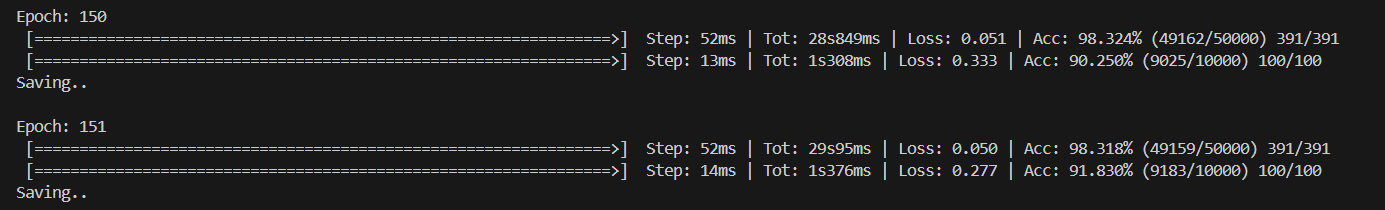
**Sep 10th**

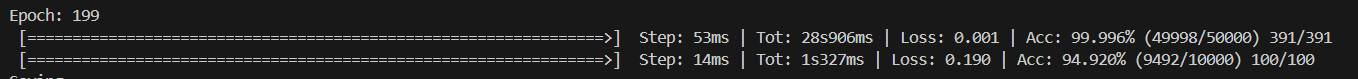
Simple DLA line: self.linear = nn.Linear(512 , num\_classes)





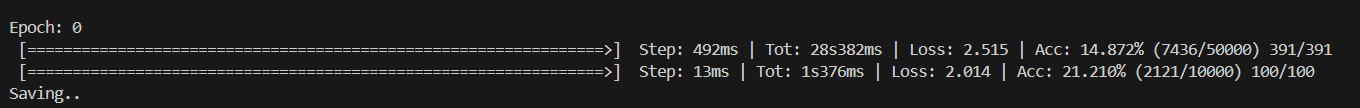


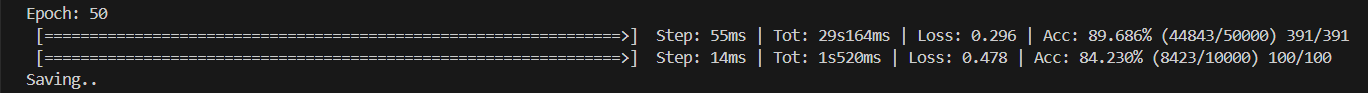


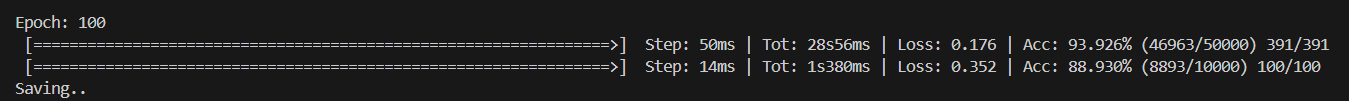


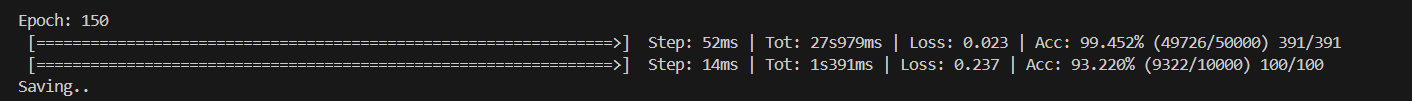
**Sep 10th**

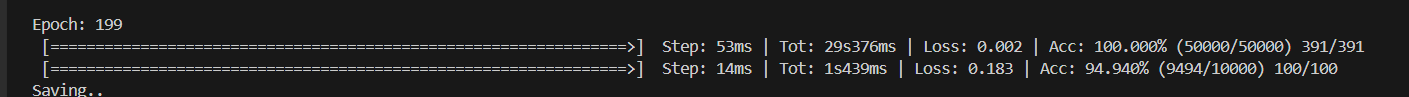
Regnet



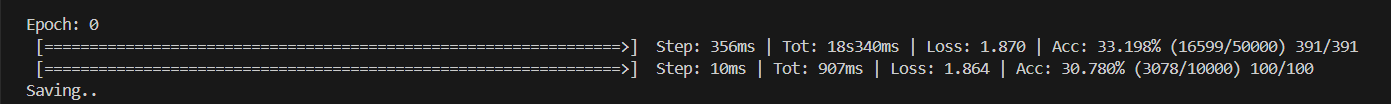


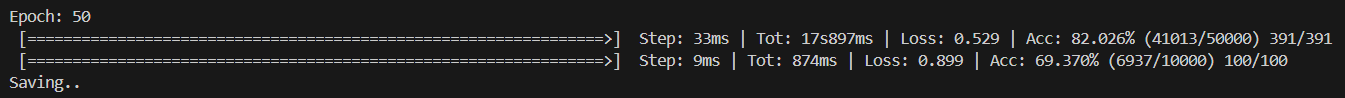


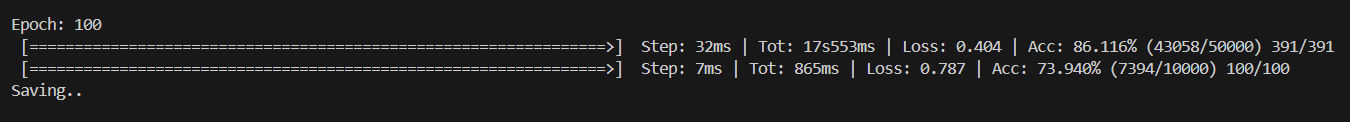


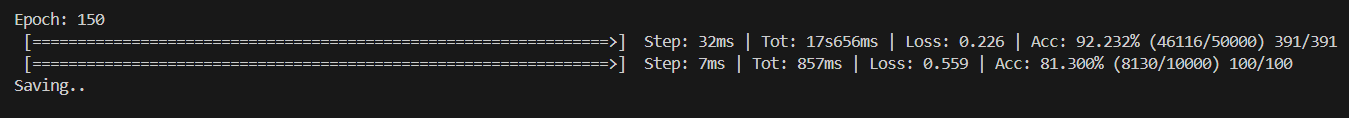


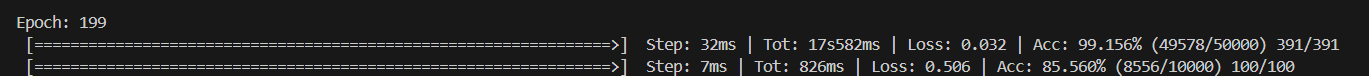
Efficient net



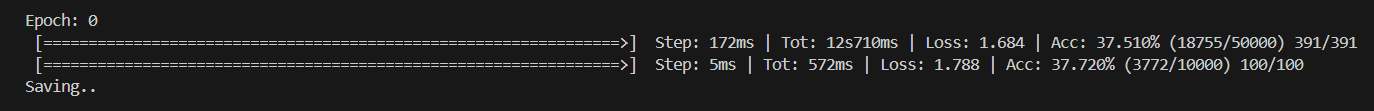


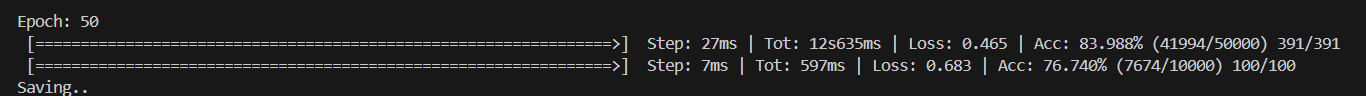


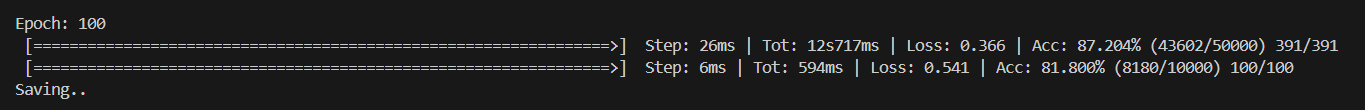


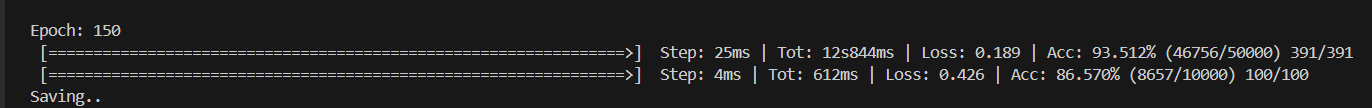


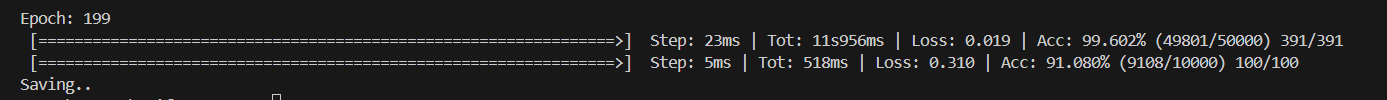
ShuffleNETV2



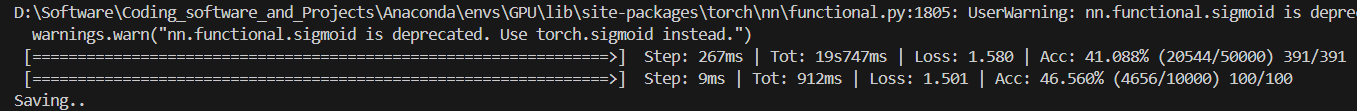


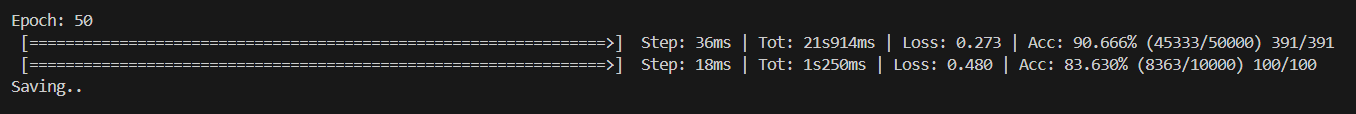


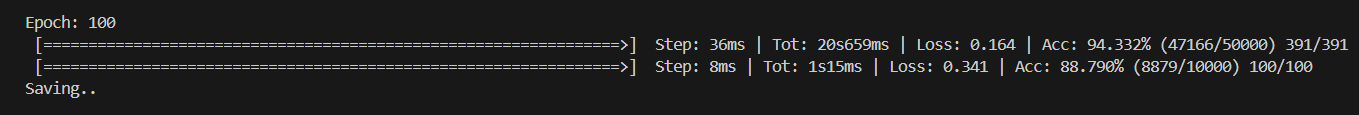


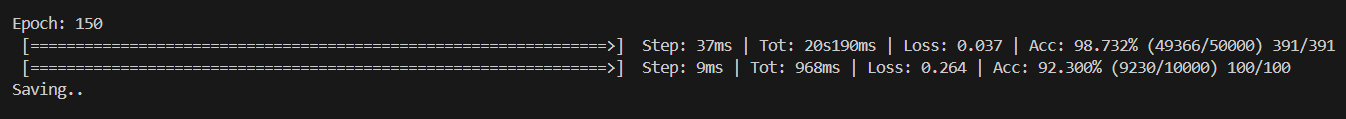


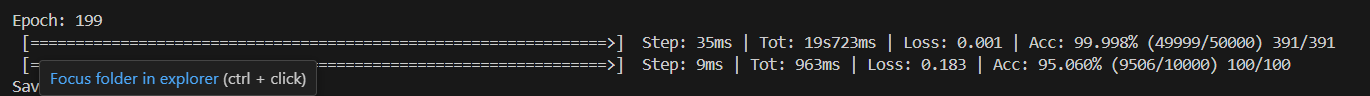
SENet18 line: self.linear = nn.Linear(512, num\_classes)





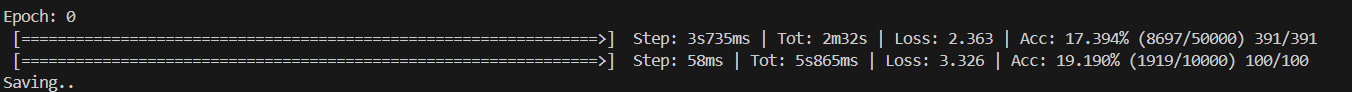


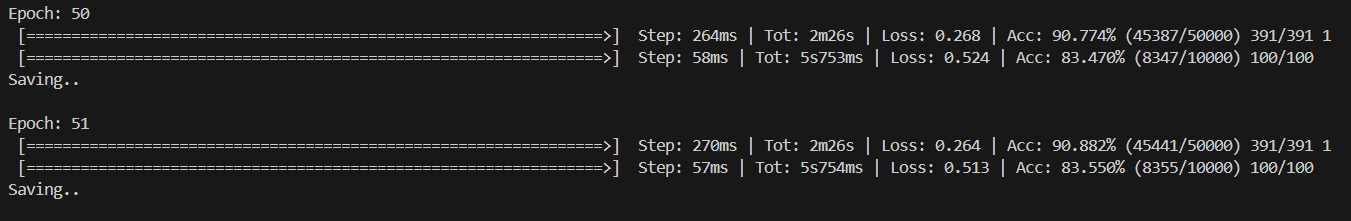


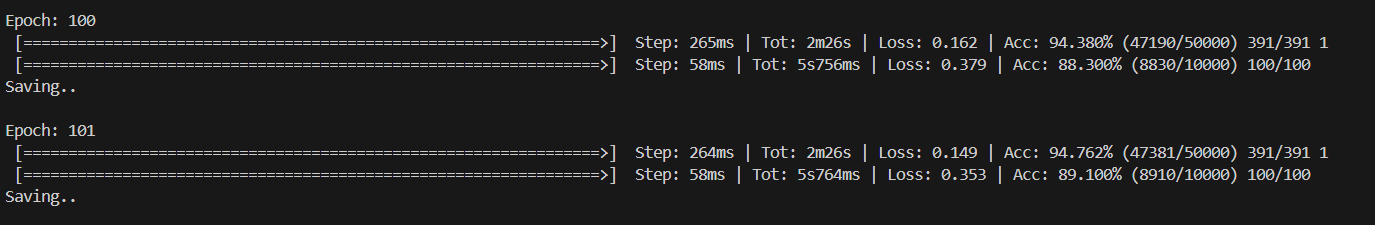


**Sep 11th**

DPN92





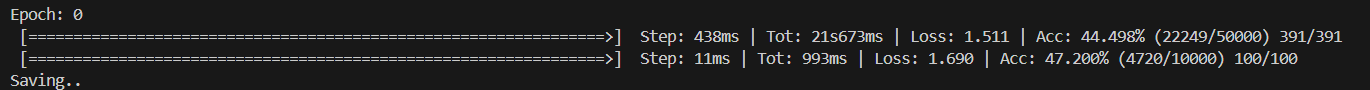


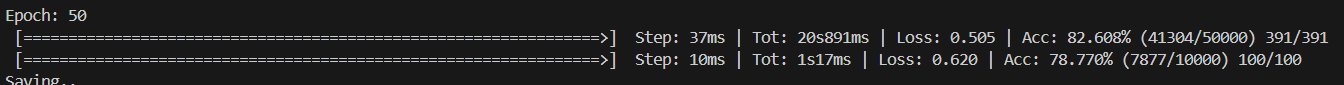
****

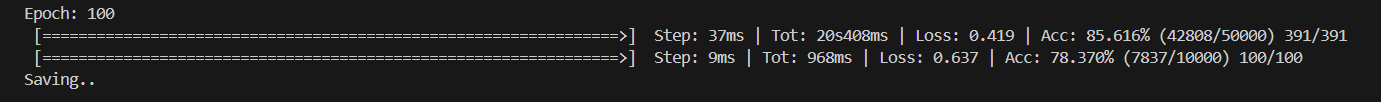


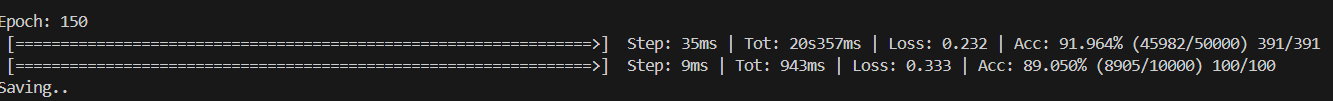


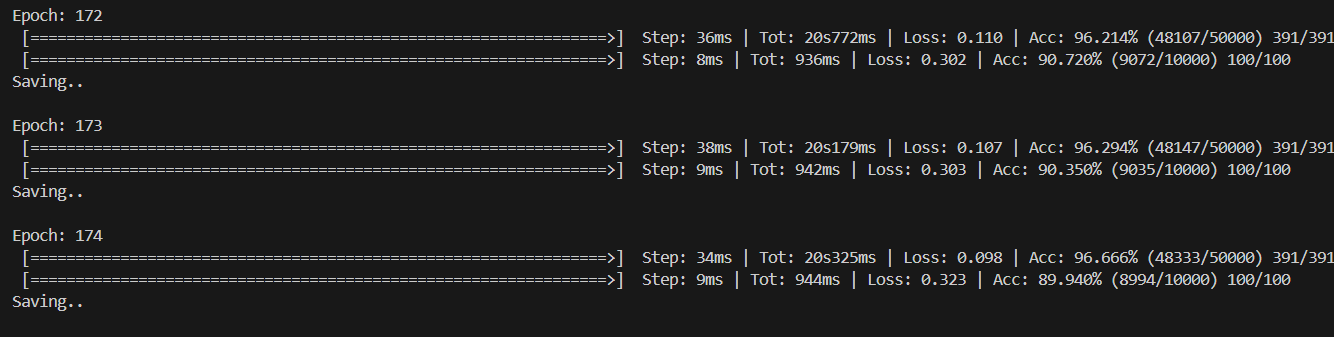
MobileNetv2

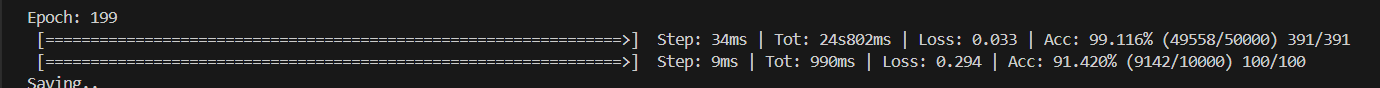




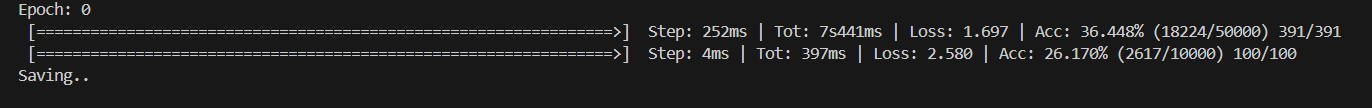


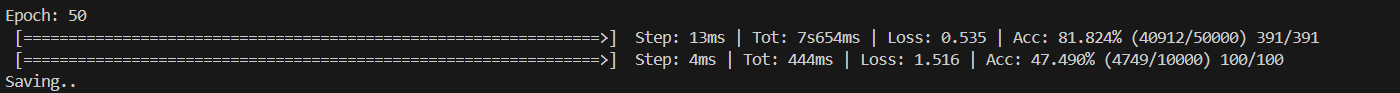


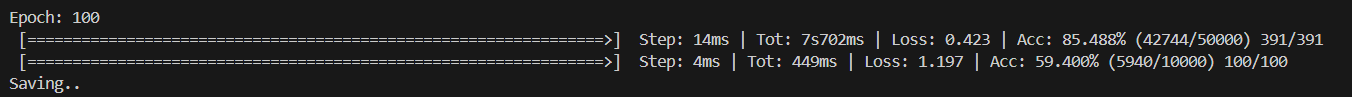


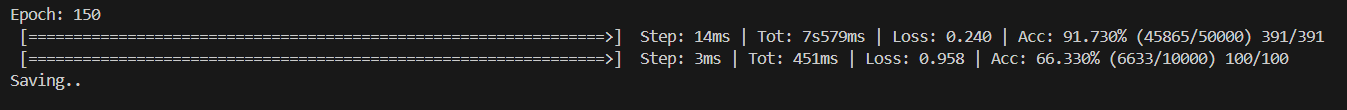


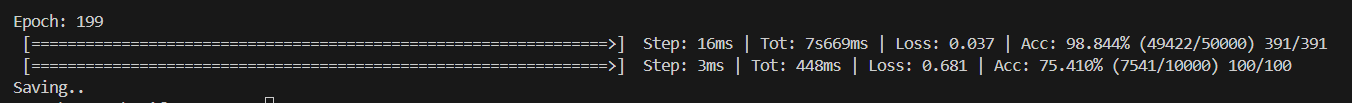
MobileNet line: self.linear = nn.Linear(1024, num\_classes)





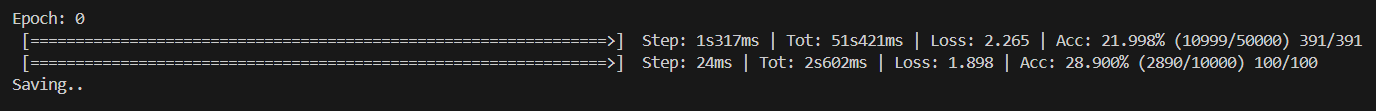


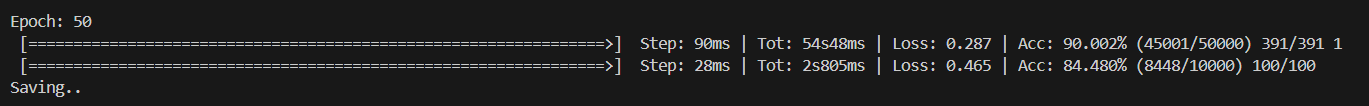


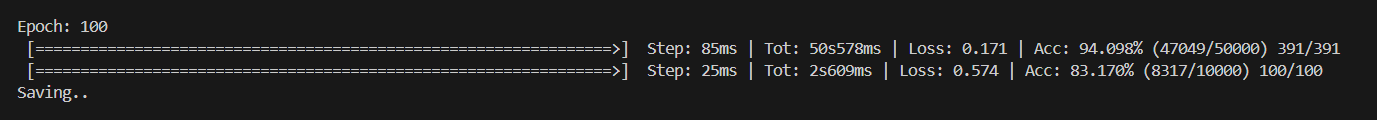


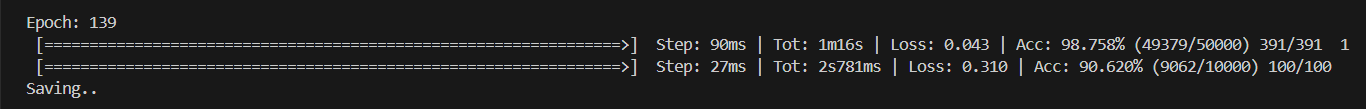
**Sep 12th**

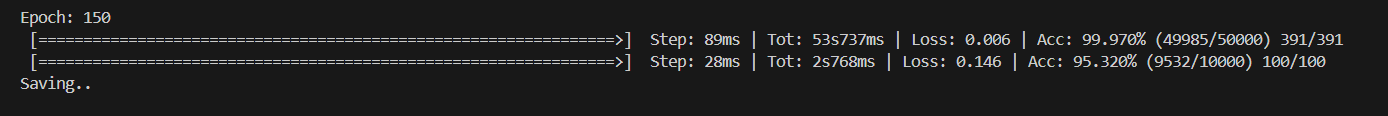
ResNeXt29

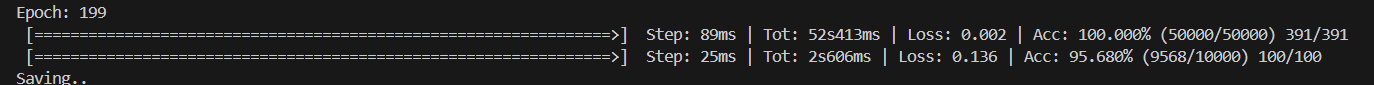




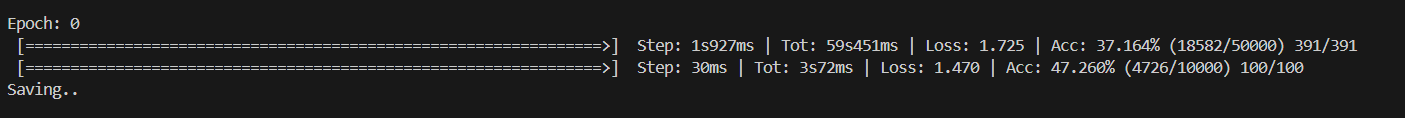


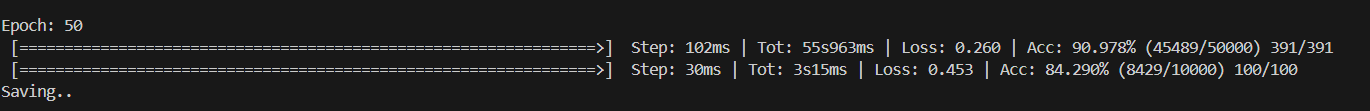


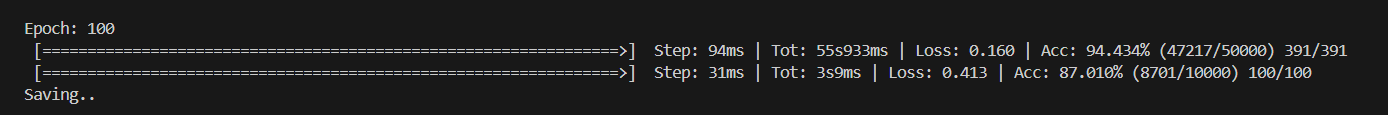


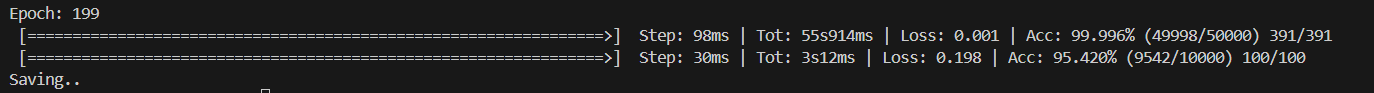
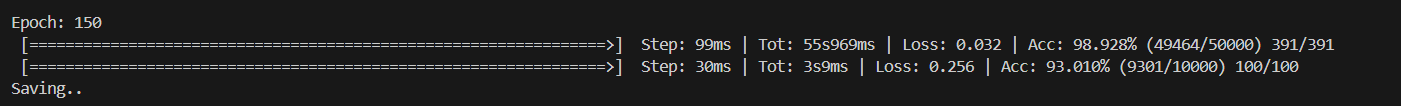


DenseNet







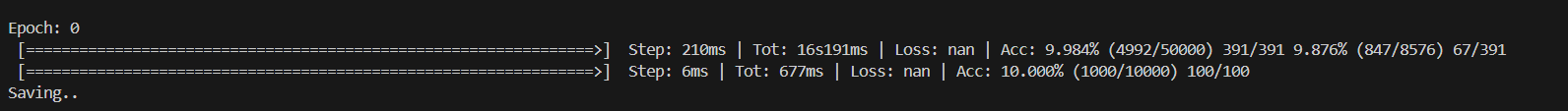


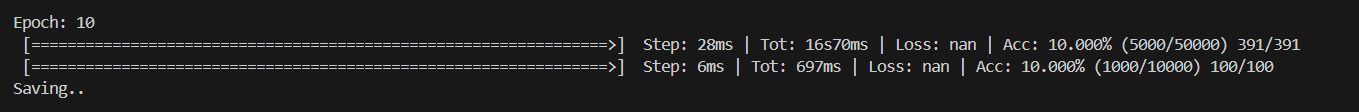
**Sep 13th:**

GoogLeNet

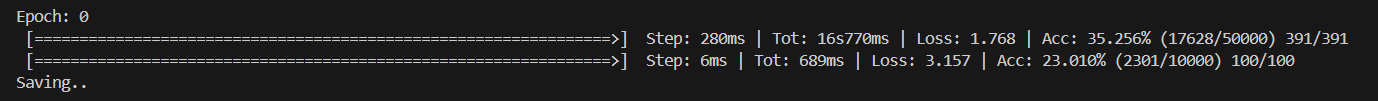
RuntimeError: CUDA error: CUBLAS\_STATUS\_INVALID\_VALUE when calling `cublasSgemm( handle, opa, opb, m, n, k, &alpha, a, lda, b, ldb, &beta, c, ldc)`

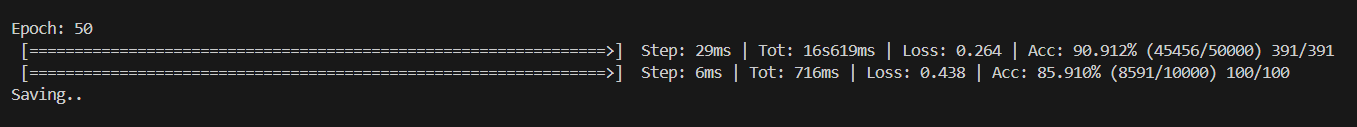
PreActResNet line: self.linear = nn.Linear(512\*block.expansion, num\_classes)

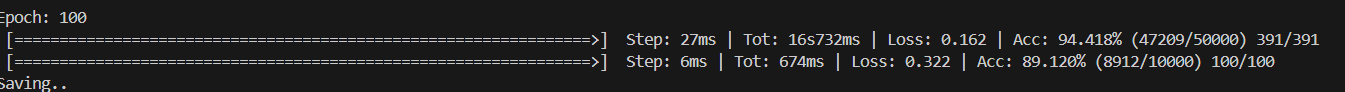




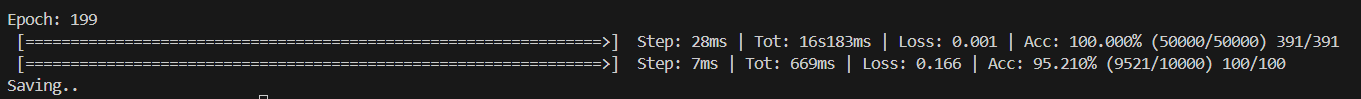
ResNet18 line: self.linear = nn.Linear(512\*block.expansion, num\_classes)



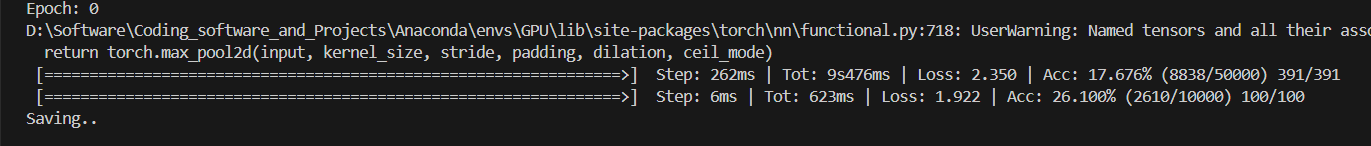


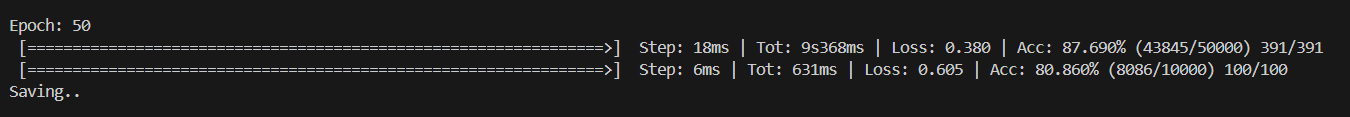


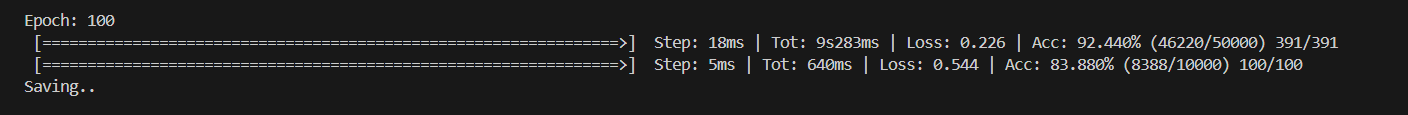


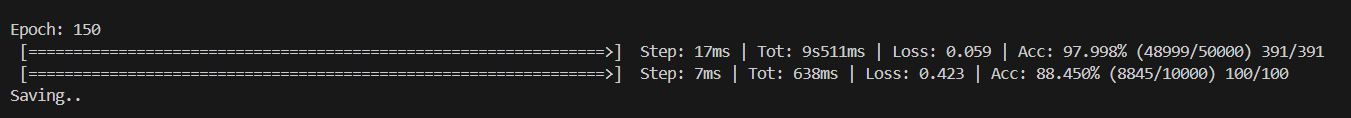


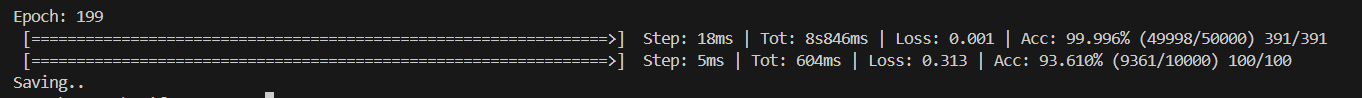
VGG line: self.classifier = nn.Linear(512, 10)



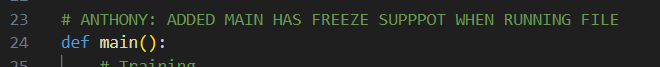




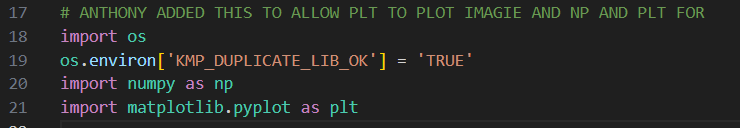




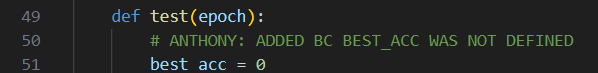
**Some Changes to Code:**

Added Def main because of main stop freeze  


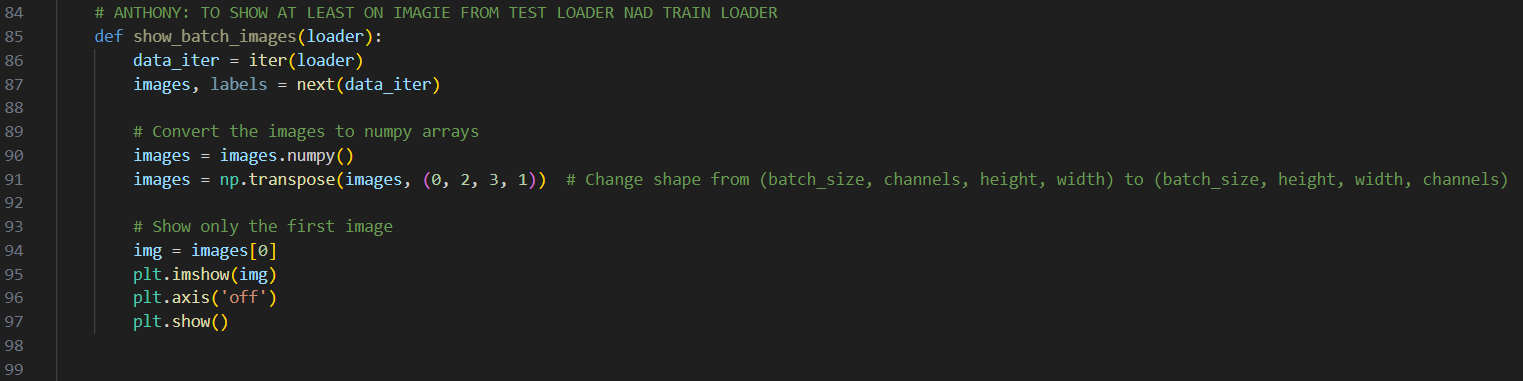
Import environment variable and add libraries to see the image pixelated



Best Acc not defined



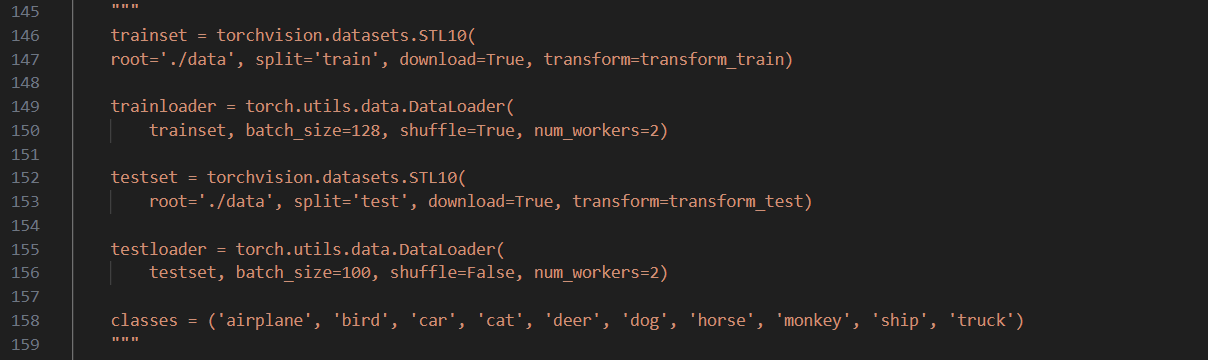
Imagine Showing



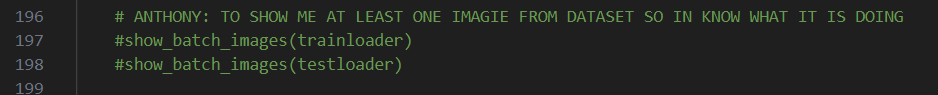
Global Best Acc variable



Other data set



Show images functions



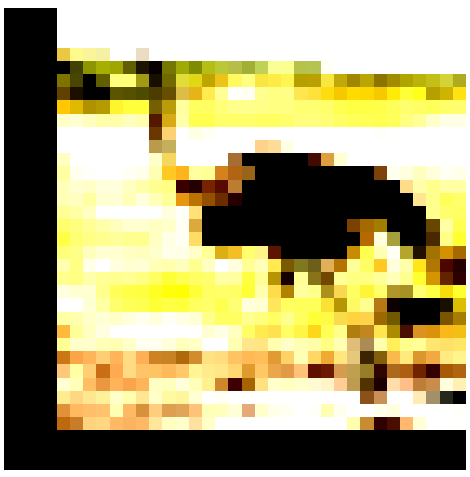
**Images 36 by 36**

**classes = ('plane', 'car', 'bird', 'cat', 'deer',**

**'dog', 'frog', 'horse', 'ship', 'truck')**

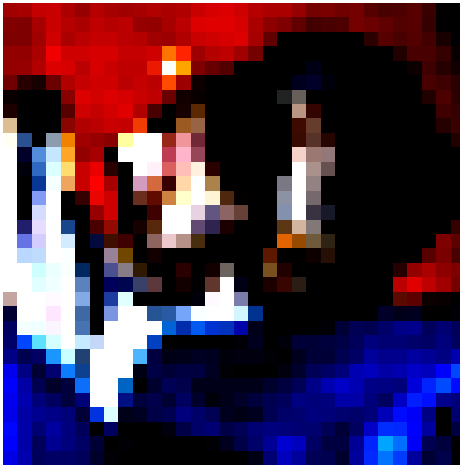
Train:

2: Bird



Test:

3: Cat



**Images 96 by 96**

classes = ('airplane', 'bird', 'car', 'cat', 'deer', 'dog', 'horse', 'monkey', 'ship', 'truck')

Train:

2: Car



Test:

6: horse

