DLOps Assignment: 3

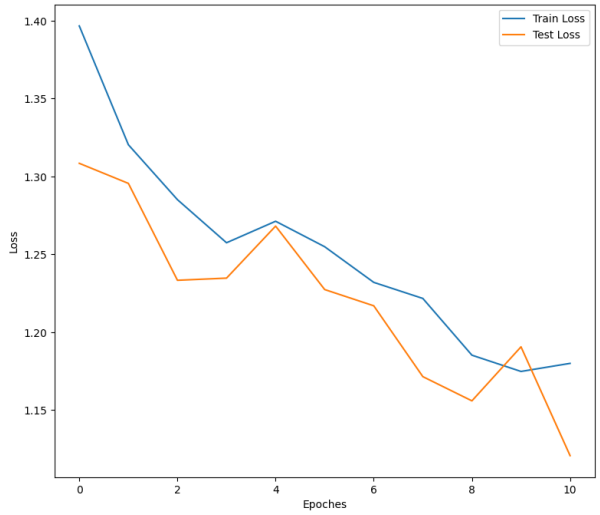
Question 1:

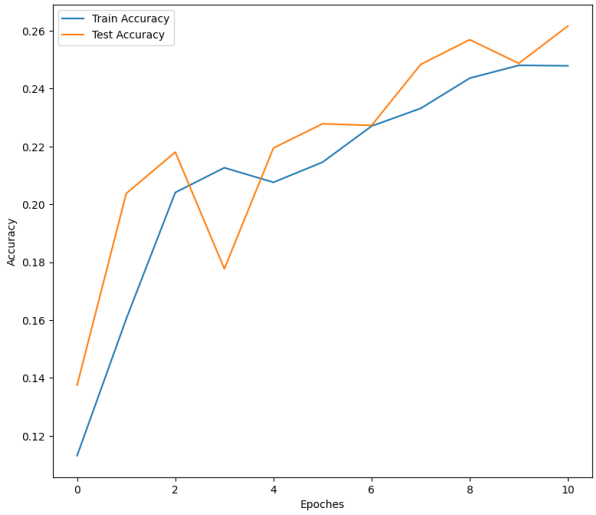
- Import required libraries and write device agnostic code
- Gat data set and get only even classes and make data loader
- Write train and test loop and take cross entropy as loss function
- Build model from various reference mention at last in report
- Now train mode with cousin positional encoder and learnable positional encode with relu an tanh respectively

Result:

COsin with ReLU

Epoch: 1 Train Loss: 1.3966 | Test Loss: 1.3084 | Train Accuray: 0.1132 | Test Accuracy: 0.1375
Epoch: 2 Train Loss: 1.3203 | Test Loss: 1.2955 | Train Accuray: 0.1604 | Test Accuracy: 0.2037
Epoch: 3 Train Loss: 1.2851 | Test Loss: 1.2333 | Train Accuray: 0.2041 | Test Accuracy: 0.2180
Epoch: 4 Train Loss: 1.2574 | Test Loss: 1.2347 | Train Accuray: 0.2126 | Test Accuracy: 0.1777
Epoch: 5 Train Loss: 1.2712 | Test Loss: 1.2681 | Train Accuray: 0.2076 | Test Accuracy: 0.2195
Epoch: 6 Train Loss: 1.2548 | Test Loss: 1.2273 | Train Accuray: 0.2146 | Test Accuracy: 0.2278
Epoch: 7 Train Loss: 1.2320 | Test Loss: 1.2169 | Train Accuray: 0.2270 | Test Accuracy: 0.2273
Epoch: 8 Train Loss: 1.2216 | Test Loss: 1.1714 | Train Accuray: 0.2332 | Test Accuracy: 0.2483
Epoch: 9 Train Loss: 1.1852 | Test Loss: 1.1559 | Train Accuray: 0.2436 | Test Accuracy: 0.2569
Epoch: 10 Train Loss: 1.1748 | Test Loss: 1.1906 | Train Accuray: 0.2480 | Test Accuracy: 0.2487
Epoch: 11 Train Loss: 1.1799 | Test Loss: 1.1207 | Train Accuray: 0.2479 | Test Accuracy: 0.2616



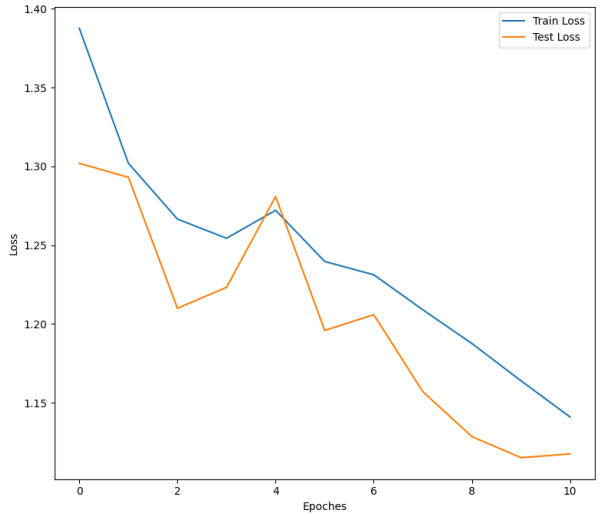


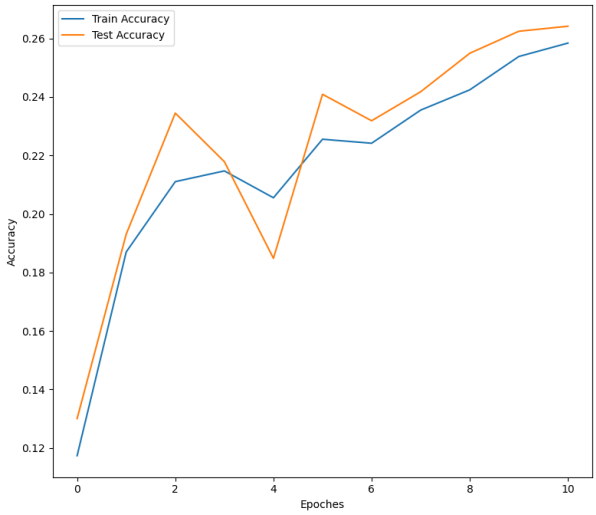
train ing time: 637.9368668800003

Learnable with ReLU

Epoch: 1 Train Loss: 1.3877 | Test Loss: 1.3020 | Train Accuray: 0.1174 | Test Accuracy: 0.1300
Epoch: 2 Train Loss: 1.3020 | Test Loss: 1.2931 | Train Accuray: 0.1870 | Test Accuracy: 0.1931
Epoch: 3 Train Loss: 1.2666 | Test Loss: 1.2100 | Train Accuray: 0.2110 | Test Accuracy: 0.2345
Epoch: 4 Train Loss: 1.2544 | Test Loss: 1.2233 | Train Accuray: 0.2147 | Test Accuracy: 0.2179

Epoch: 5 Train Loss: 1.2722 | Test Loss: 1.2809 | Train Accuray: 0.2055 | Test Accuracy: 0.1848
Epoch: 6 Train Loss: 1.2397 | Test Loss: 1.1959 | Train Accuray: 0.2255 | Test Accuracy: 0.2409
Epoch: 7 Train Loss: 1.2312 | Test Loss: 1.2059 | Train Accuray: 0.2242 | Test Accuracy: 0.2319
Epoch: 8 Train Loss: 1.2090 | Test Loss: 1.1571 | Train Accuray: 0.2355 | Test Accuracy: 0.2418
Epoch: 9 Train Loss: 1.1876 | Test Loss: 1.1285 | Train Accuray: 0.2424 | Test Accuracy: 0.2549
Epoch: 10 Train Loss: 1.1640 | Test Loss: 1.1152 | Train Accuray: 0.2538 | Test Accuracy: 0.2625
Epoch: 11 Train Loss: 1.1410 | Test Loss: 1.1176 | Train Accuray: 0.2584 | Test Accuracy: 0.2642

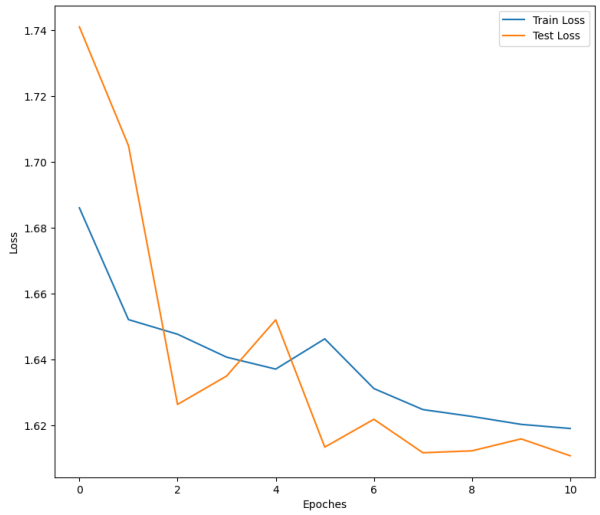


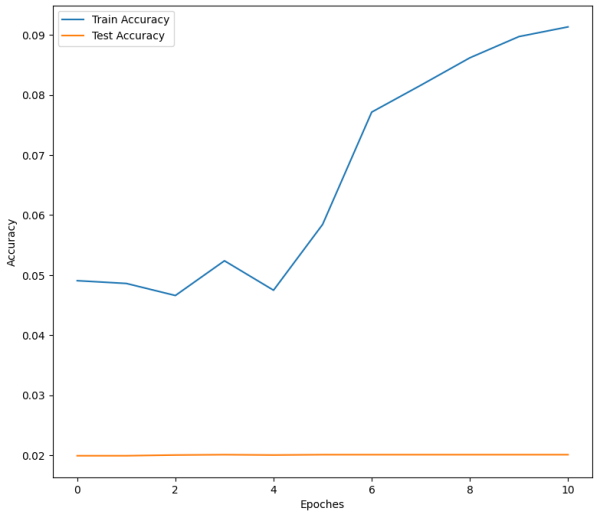


train ing time: 587.6857188160002

Cosin with Tanh

Epoch: 1 Train Loss: 1.6861 | Test Loss: 1.7410 | Train Accuray: 0.0491 | Test Accuracy: 0.0199
Epoch: 2 Train Loss: 1.6522 | Test Loss: 1.7050 | Train Accuray: 0.0486 | Test Accuracy: 0.0199
Epoch: 3 Train Loss: 1.6477 | Test Loss: 1.6264 | Train Accuray: 0.0466 | Test Accuracy: 0.0200
Epoch: 4 Train Loss: 1.6408 | Test Loss: 1.6351 | Train Accuray: 0.0524 | Test Accuracy: 0.0201
Epoch: 5 Train Loss: 1.6371 | Test Loss: 1.6521 | Train Accuray: 0.0475 | Test Accuracy: 0.0200
Epoch: 6 Train Loss: 1.6463 | Test Loss: 1.6135 | Train Accuray: 0.0584 | Test Accuracy: 0.0201
Epoch: 7 Train Loss: 1.6312 | Test Loss: 1.6219 | Train Accuray: 0.0771 | Test Accuracy: 0.0201
Epoch: 8 Train Loss: 1.6249 | Test Loss: 1.6118 | Train Accuray: 0.0816 | Test Accuracy: 0.0201
Epoch: 9 Train Loss: 1.6228 | Test Loss: 1.6123 | Train Accuray: 0.0862 | Test Accuracy: 0.0201
Epoch: 10 Train Loss: 1.6204 | Test Loss: 1.6160 | Train Accuray: 0.0897 | Test Accuracy: 0.0201
Epoch: 11 Train Loss: 1.6191 | Test Loss: 1.6109 | Train Accuray: 0.0913 | Test Accuracy: 0.0201

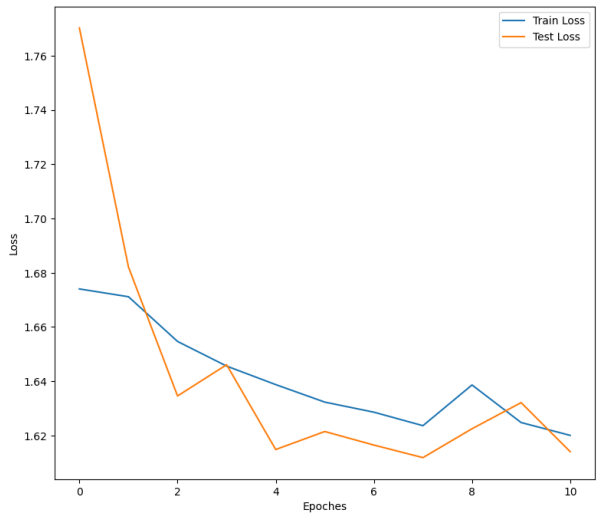


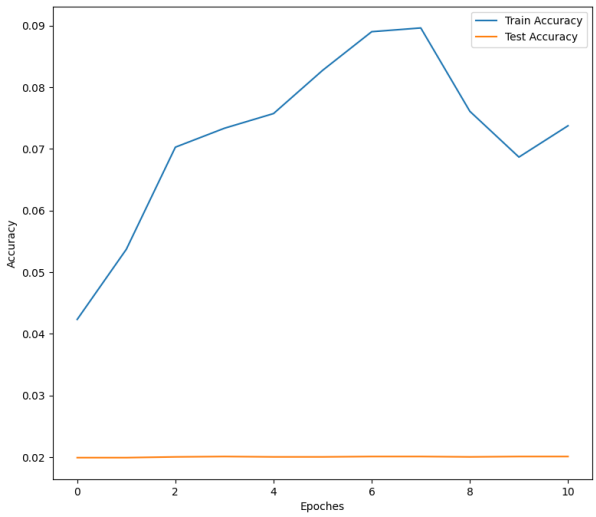


train ing time: 582.4674055289997

Learnable with Tanh

Epoch: 1 Train Loss: 1.6740 | Test Loss: 1.7703 | Train Accuray: 0.0423 | Test Accuracy: 0.0199
Epoch: 2 Train Loss: 1.6712 | Test Loss: 1.6821 | Train Accuray: 0.0537 | Test Accuracy: 0.0199
Epoch: 3 Train Loss: 1.6547 | Test Loss: 1.6346 | Train Accuray: 0.0703 | Test Accuracy: 0.0200
Epoch: 4 Train Loss: 1.6456 | Test Loss: 1.6461 | Train Accuray: 0.0734 | Test Accuracy: 0.0201
Epoch: 5 Train Loss: 1.6388 | Test Loss: 1.6148 | Train Accuray: 0.0757 | Test Accuracy: 0.0200
Epoch: 6 Train Loss: 1.6323 | Test Loss: 1.6215 | Train Accuray: 0.0827 | Test Accuracy: 0.0200
Epoch: 7 Train Loss: 1.6286 | Test Loss: 1.6165 | Train Accuray: 0.0890 | Test Accuracy: 0.0201
Epoch: 8 Train Loss: 1.6236 | Test Loss: 1.6118 | Train Accuray: 0.0896 | Test Accuracy: 0.0201
Epoch: 9 Train Loss: 1.6386 | Test Loss: 1.6225 | Train Accuray: 0.0761 | Test Accuracy: 0.0200
Epoch: 10 Train Loss: 1.6248 | Test Loss: 1.6321 | Train Accuray: 0.0687 | Test Accuracy: 0.0201
Epoch: 11 Train Loss: 1.6200 | Test Loss: 1.6140 | Train Accuray: 0.0738 | Test Accuracy: 0.0201





train ing time: 581.4097971070005

Observation:

- With ReLU Cosin and Learnable positional encoder train properly and Learnable trains faster than Cosin and also give better performance than Cosin.
- However with Tanh both do overfitting because Tanh does not converge to minima in both positional encoder models.
- However VIT takes less time to train than other VGG models.

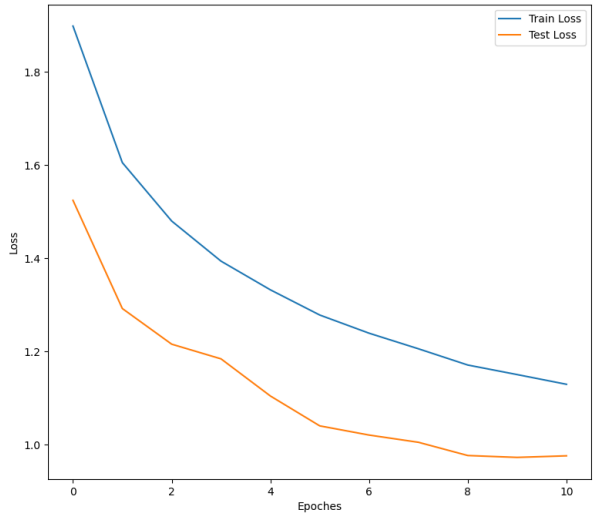
Question 2:

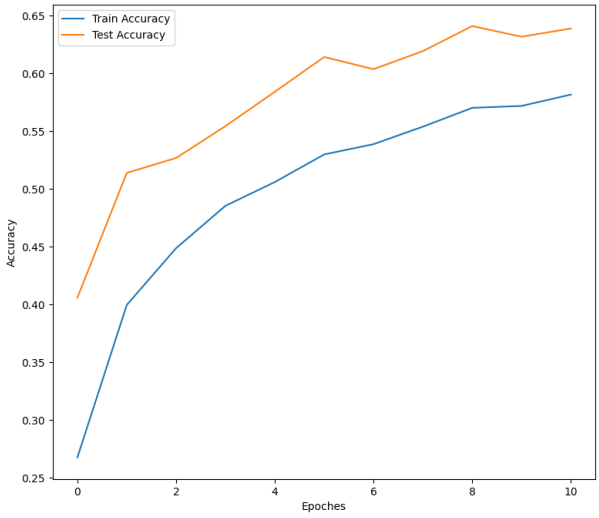
- Import required libraries and write device agnostic code
- Now lode data and augment and preprocess it
- Visualize the augment data
- Write train and test function and plot function for Loss/Accuracy vs Epoch
- Take cross entropy as loss function and multiclass accuracy as accuracy function from torch metrics
- Define default class and train it
- Same for VGG 16 take default from pytorch and train it

Result:

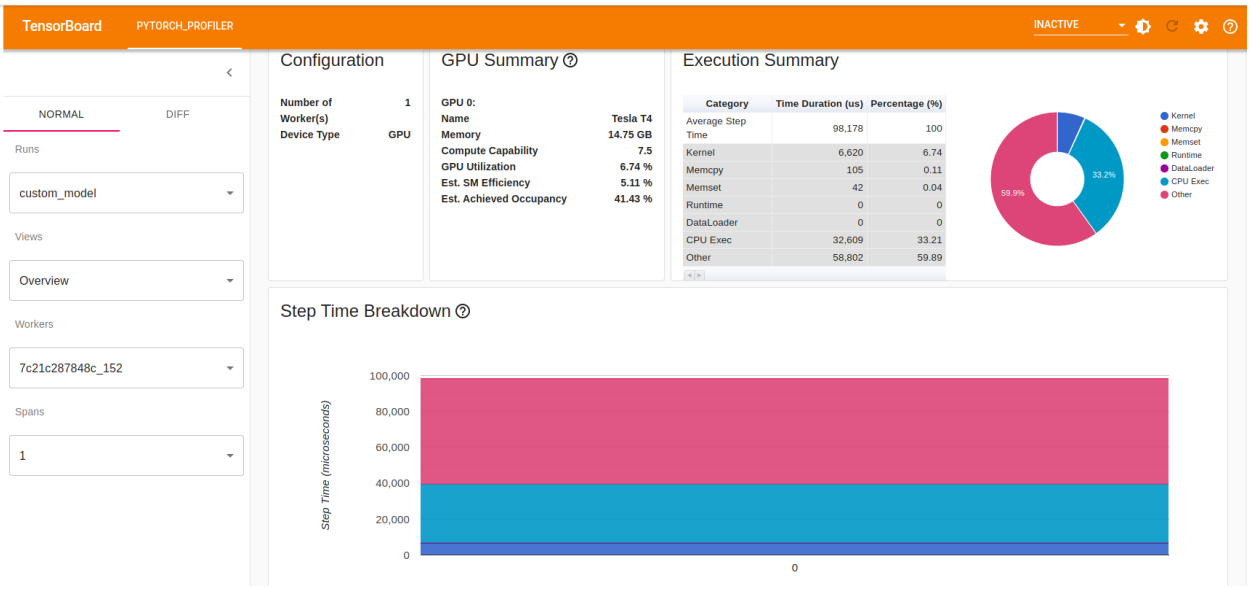
CUSTOM MODEL

Epoch: 1 Train Loss: 1.8983 | Test Loss: 1.5242 | Train Accuray: 0.2677 | Test Accuracy: 0.4059
Epoch: 2 Train Loss: 1.6052 | Test Loss: 1.2919 | Train Accuray: 0.3996 | Test Accuracy: 0.5138
Epoch: 3 Train Loss: 1.4797 | Test Loss: 1.2154 | Train Accuray: 0.4487 | Test Accuracy: 0.5267
Epoch: 4 Train Loss: 1.3936 | Test Loss: 1.1839 | Train Accuray: 0.4853 | Test Accuracy: 0.5543
Epoch: 5 Train Loss: 1.3321 | Test Loss: 1.1042 | Train Accuray: 0.5059 | Test Accuracy: 0.5839
Epoch: 6 Train Loss: 1.2780 | Test Loss: 1.0401 | Train Accuray: 0.5297 | Test Accuracy: 0.6140
Epoch: 7 Train Loss: 1.2390 | Test Loss: 1.0204 | Train Accuray: 0.5386 | Test Accuracy: 0.6035
Epoch: 8 Train Loss: 1.2055 | Test Loss: 1.0048 | Train Accuray: 0.5538 | Test Accuracy: 0.6191
Epoch: 9 Train Loss: 1.1707 | Test Loss: 0.9764 | Train Accuray: 0.5700 | Test Accuracy: 0.6408
Epoch: 10 Train Loss: 1.1502 | Test Loss: 0.9724 | Train Accuray: 0.5717 | Test Accuracy: 0.6315
Epoch: 11 Train Loss: 1.1293 | Test Loss: 0.9758 | Train Accuray: 0.5815 | Test Accuracy: 0.6387



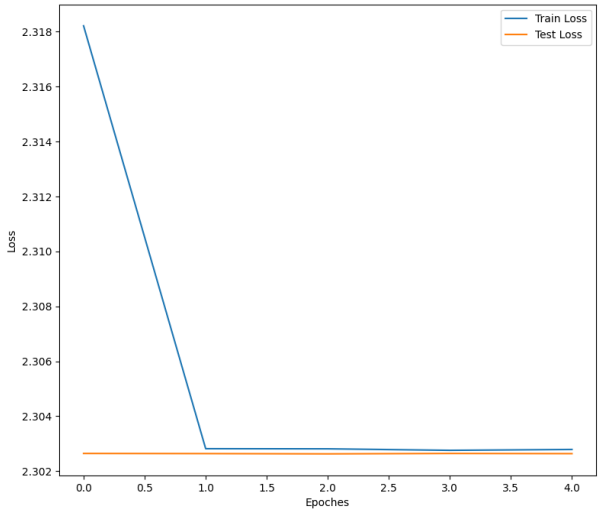


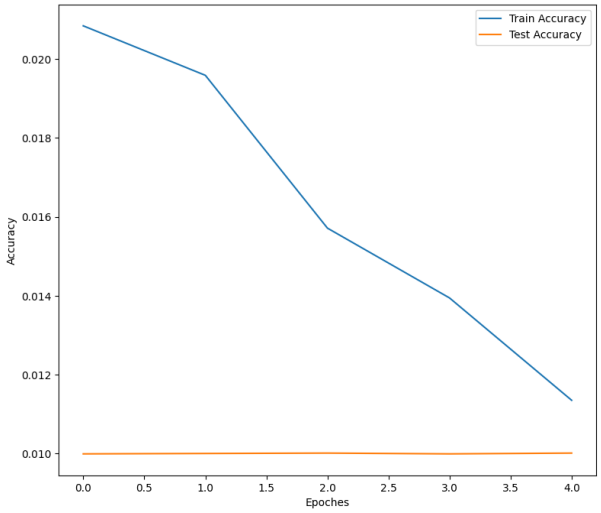
train ing time: 466.17159349300005



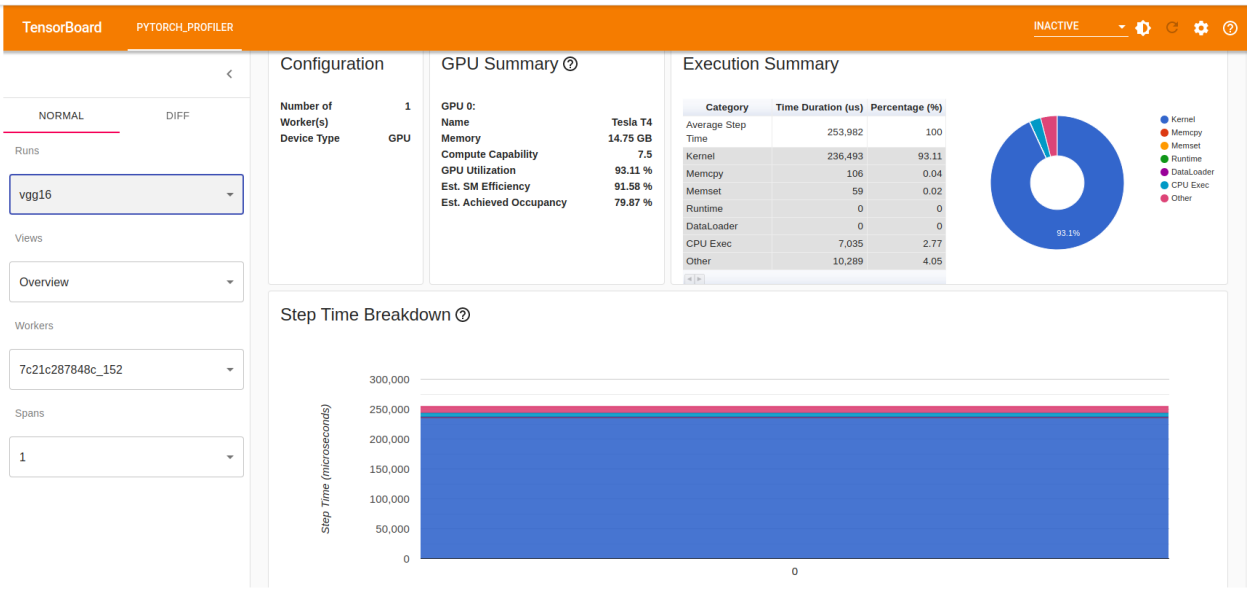
VGG16

Epoch: 1 Train Loss: 2.3182 | Test Loss: 2.3026 | Train Accuray: 0.0208 | Test Accuracy: 0.0100
Epoch: 2 Train Loss: 2.3028 | Test Loss: 2.3026 | Train Accuray: 0.0196 | Test Accuracy: 0.0100
Epoch: 3 Train Loss: 2.3028 | Test Loss: 2.3026 | Train Accuray: 0.0157 | Test Accuracy: 0.0100
Epoch: 4 Train Loss: 2.3028 | Test Loss: 2.3026 | Train Accuray: 0.0139 | Test Accuracy: 0.0100
Epoch: 5 Train Loss: 2.3028 | Test Loss: 2.3026 | Train Accuray: 0.0114 | Test Accuracy: 0.0100



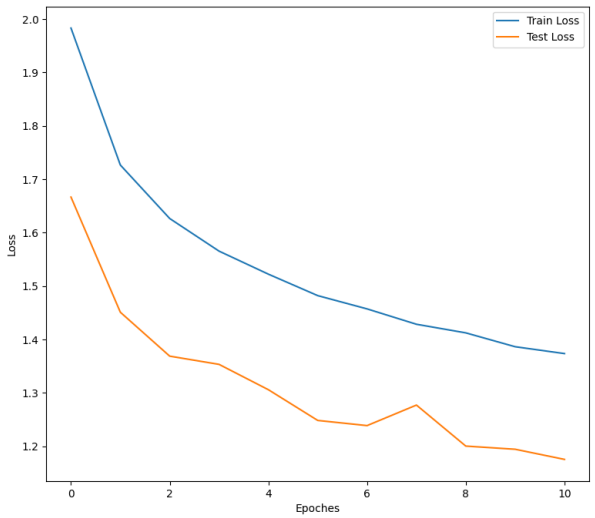


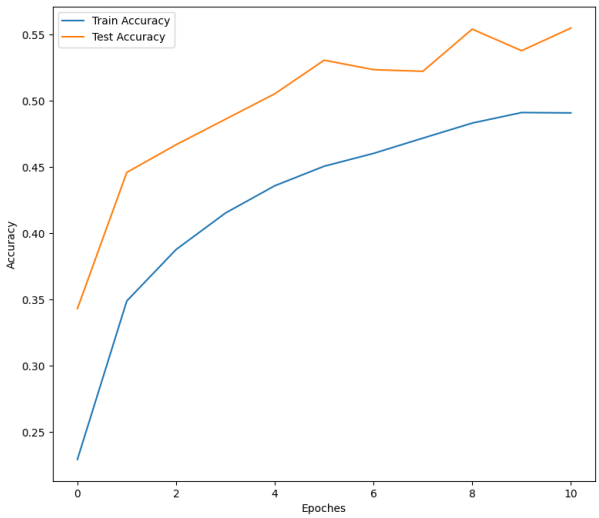
train ing time: 709.2815415939999



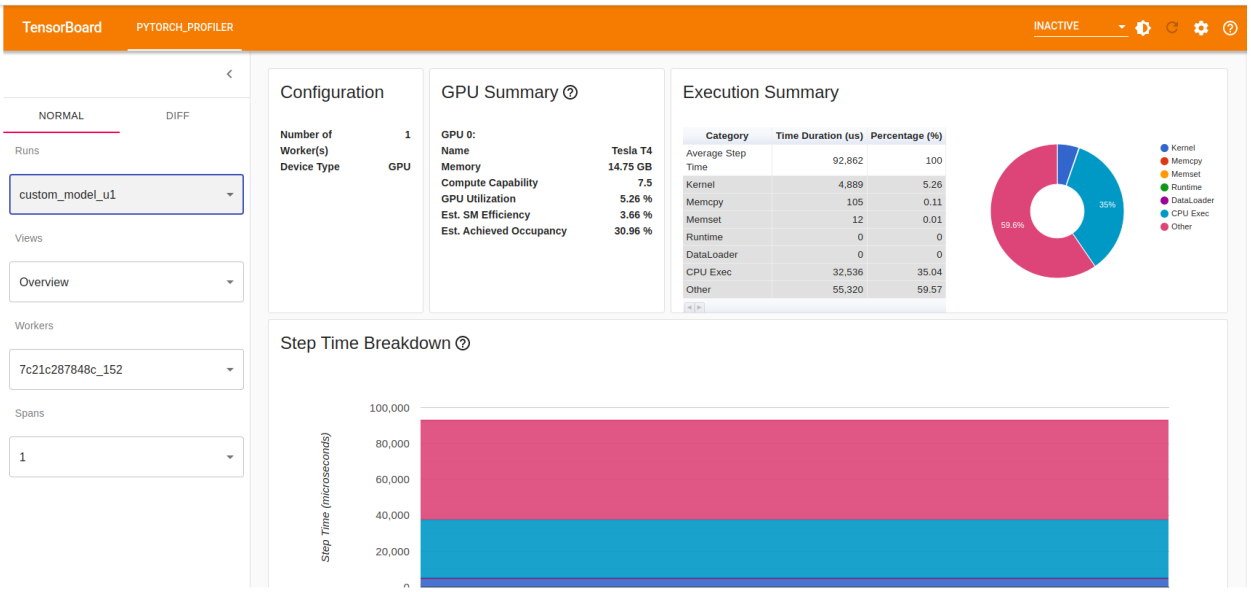
CUSTOM MODEL U1

Epoch: 1 Train Loss: 1.9829 | Test Loss: 1.6665 | Train Accuray: 0.2295 | Test Accuracy: 0.3433
Epoch: 2 Train Loss: 1.7265 | Test Loss: 1.4509 | Train Accuray: 0.3491 | Test Accuracy: 0.4461
Epoch: 3 Train Loss: 1.6264 | Test Loss: 1.3687 | Train Accuray: 0.3879 | Test Accuracy: 0.4671
Epoch: 4 Train Loss: 1.5653 | Test Loss: 1.3533 | Train Accuray: 0.4155 | Test Accuracy: 0.4863
Epoch: 5 Train Loss: 1.5222 | Test Loss: 1.3058 | Train Accuray: 0.4361 | Test Accuracy: 0.5055
Epoch: 6 Train Loss: 1.4820 | Test Loss: 1.2483 | Train Accuray: 0.4508 | Test Accuracy: 0.5308
Epoch: 7 Train Loss: 1.4571 | Test Loss: 1.2386 | Train Accuray: 0.4605 | Test Accuracy: 0.5237
Epoch: 8 Train Loss: 1.4284 | Test Loss: 1.2771 | Train Accuray: 0.4720 | Test Accuracy: 0.5225
Epoch: 9 Train Loss: 1.4123 | Test Loss: 1.2003 | Train Accuray: 0.4834 | Test Accuracy: 0.5543
Epoch: 10 Train Loss: 1.3863 | Test Loss: 1.1943 | Train Accuray: 0.4913 | Test Accuracy: 0.5380
Epoch: 11 Train Loss: 1.3735 | Test Loss: 1.1754 | Train Accuray: 0.4910 | Test Accuracy: 0.5551



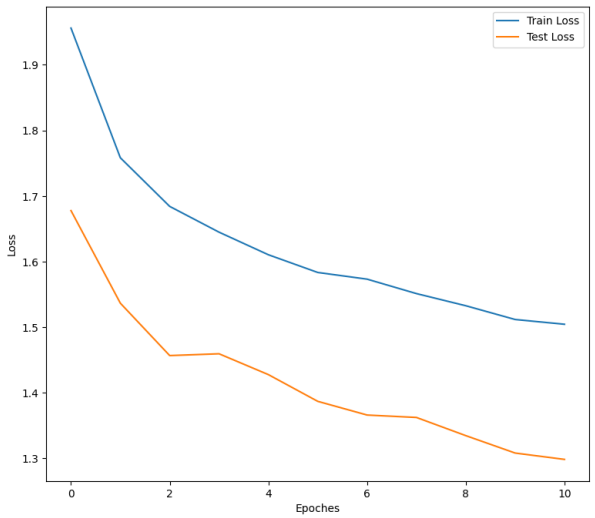


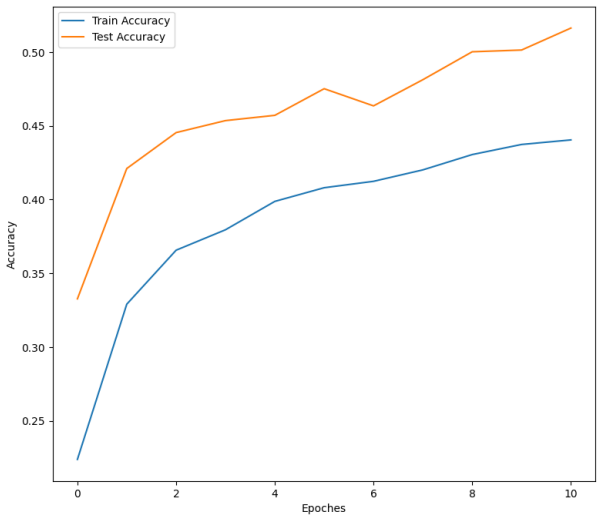
train ing time: 467.91294503899985



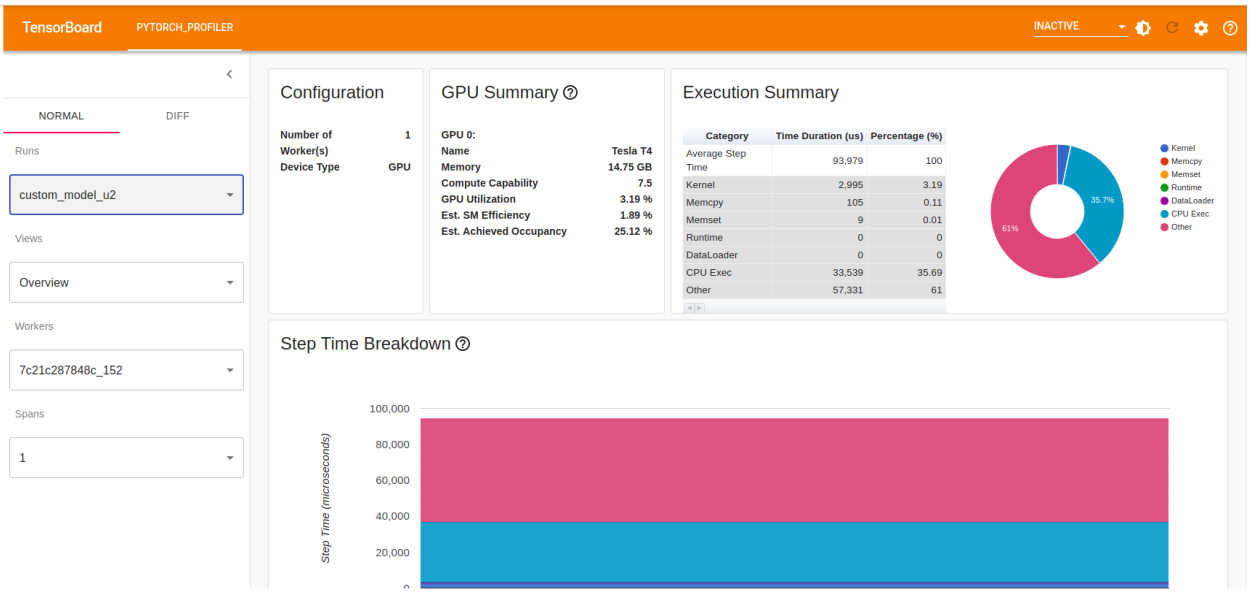
CUSTOM MODEL U2

Epoch: 1 Train Loss: 1.9561 | Test Loss: 1.6778 | Train Accuray: 0.2239 | Test Accuracy: 0.3327
Epoch: 2 Train Loss: 1.7583 | Test Loss: 1.5367 | Train Accuray: 0.3291 | Test Accuracy: 0.4210
Epoch: 3 Train Loss: 1.6842 | Test Loss: 1.4568 | Train Accuray: 0.3657 | Test Accuracy: 0.4454
Epoch: 4 Train Loss: 1.6449 | Test Loss: 1.4595 | Train Accuray: 0.3795 | Test Accuracy: 0.4535
Epoch: 5 Train Loss: 1.6105 | Test Loss: 1.4278 | Train Accuray: 0.3987 | Test Accuracy: 0.4571
Epoch: 6 Train Loss: 1.5835 | Test Loss: 1.3870 | Train Accuray: 0.4080 | Test Accuracy: 0.4751
Epoch: 7 Train Loss: 1.5733 | Test Loss: 1.3662 | Train Accuray: 0.4123 | Test Accuracy: 0.4635
Epoch: 8 Train Loss: 1.5513 | Test Loss: 1.3625 | Train Accuray: 0.4201 | Test Accuracy: 0.4812
Epoch: 9 Train Loss: 1.5328 | Test Loss: 1.3348 | Train Accuray: 0.4305 | Test Accuracy: 0.5001
Epoch: 10 Train Loss: 1.5118 | Test Loss: 1.3082 | Train Accuray: 0.4373 | Test Accuracy: 0.5013
Epoch: 11 Train Loss: 1.5045 | Test Loss: 1.2985 | Train Accuray: 0.4404 | Test Accuracy: 0.5162





train ing time: 407.1790703509996



Observation:

- Custom architecture model perform very well on CIFAR10. Meanwhile, VGG16 shows overfitting. Because CIFAR10 is less complex than the ImageNet dataset.
- By reducing in custom model CNN channels we reduce the performance but train time remains the same. However, by removing some CNN from architecture we reduce the training time but performance also goes down.

SOME APPROACH TO INCREASE THE PERFORMANCE AND REDUCE THE TIME:

- Use batch normalization
- If possible use MAXpool or AVGpool after each conv layer.
- Reduce the con layer from architecture

REFERENCE:

https://towardsdatascience.com/how-to-reduce-training-parameters-in-cnns-while-keeping-accuracy-99-a213034a9777

https://medium.com/@dipti.rohan.pawar/improving-performance-of-convolutional-neural-network-2ecfe0207de7

https://www.analyticsvidhya.com/blog/2019/11/4-tricks-improve-deep-learning-model-performance/

https://www.youtube.com/watch?v=U0s0f995w14

https://www.youtube.com/watch?v=qU7wO02urYU

https://www.youtube.com/watch?v=tjpW_BY8y3g