DL-Ops

Lab Assignment 8- Report

Debonil Ghosh
Roll No: M21AIE225
Executive MTech
Artificial Intelligence
Indian Institute of Technology, Jodhpur

Problem Statement:

Question 1 [100 marks]

Train a ResNet18 model for classification on even classes of CIFAR-10 (for students with even roll no)/ odd classes of FashionMNIST (for students with odd roll no) for 30 epochs. Analyze the training of model using wandb library and "weights & biases" tool covered in the class.

1) Show comparison of the performance of the hyperparameter tuning using plots using the "weights & biases" tool. Hyperparameters: choice of activation function, optimizer

2) Write the observations about GPU (like GPU utilization, temperature etc).

3) Show all the logs about accuracy, loss using plots using "weights & biases" tool

4) Generate the report of the entire experiment using the "weights & biases" tool. Attach this report to your assignment report.

5) Add a visual result analysis on the "weights & biases" tool taught by to Dr Anush, where he shows the prediction for different images. Include this in the assignment video in addition to the other demonstrations.

Note: The comparison should be done at least between 3-4 models (resulting from the choice of hyperparameters)

Reference:

https://wandb.ai/site

Solution Google Colab Link:

https://colab.research.google.com/drive/1VLG8ipcuIq4utJUc0sbYIwJByvrE8E14?usp=sharing

Solution Weights & Biases Report Link:

https://api.wandb.ai/links/ghosh-11/c7dy4c2n

Model Allotted:

ResNet18 on FashionMNIST dataset

[Roll number M21AIE225]

ResNet18

ResNet-18 is a convolutional neural network architecture developed by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun of Microsoft Research in 2015. It is a variant of the original ResNet architecture, which stands for "Residual Network," and was developed to address the problem of vanishing gradients in very deep neural networks. ResNet-18 is a relatively shallow network, consisting of only 18 layers, but it is still powerful enough to achieve state-of-the-art performance on a variety of computer vision tasks.

The ResNet-18 architecture is based on a series of residual blocks, each of which contains two convolutional layers with batch normalization and a shortcut connection that skips over the second convolutional layer. The shortcut connection allows information to flow directly from the input of the block to the output, bypassing the convolutional layers. This helps to avoid the vanishing gradient problem and allows the network to be trained more effectively.

The first layer of the network is a standard convolutional layer that takes in the input image. This is followed by a max pooling layer that reduces the spatial dimensions of the feature maps. The network then consists of four stages, each of which contains a varying number of residual blocks.

The first stage consists of two residual blocks, each with two convolutional layers with 64 filters. The second stage contains two residual blocks, each with two convolutional layers with 128 filters. The third stage contains two residual blocks, each with two convolutional layers with 256 filters. Finally, the fourth stage contains two residual blocks, each with two convolutional layers with 512 filters.

The output of the final stage is passed through a global average pooling layer, which averages the feature maps over the spatial dimensions. This is followed by a fully connected layer with 1000 units, corresponding to the 1000 classes in the ImageNet dataset. The output of this layer is passed through a softmax activation function to produce the final probability distribution over the classes.

ResNet-18 is a powerful convolutional neural network architecture that addresses the problem of vanishing gradients in deep networks. It achieves state-of-the-art performance on the ImageNet dataset while being relatively small and easy to train. Its success has inspired the development of many other ResNet variants, including ResNet-34, ResNet-50, and ResNet-101, which have achieved even better performance on a variety of computer vision tasks.

Architecture:

```
ResNet(
 (conv1): Conv2d(1, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
 (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
 (relu): ReLU(inplace=True)
 (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
 (layer1): Sequential(
  (0): BasicBlock(
   (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
   (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   (relu): ReLU(inplace=True)
   (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
   (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

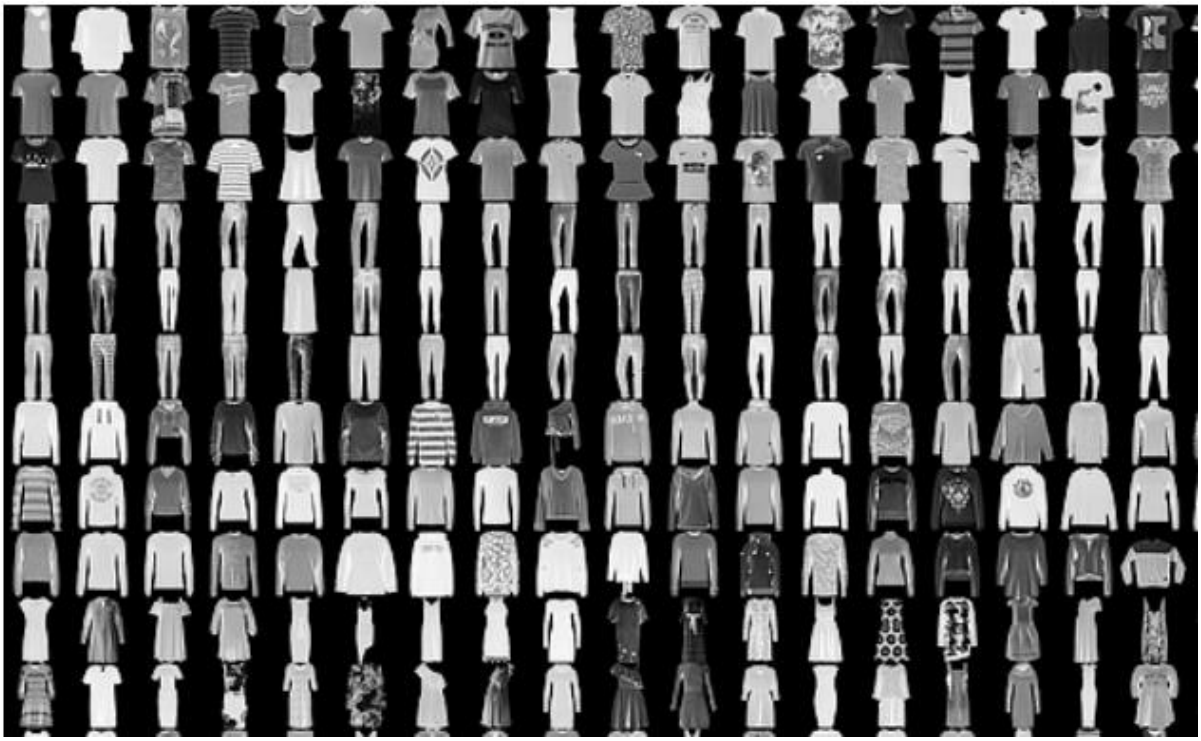
```
  )
  (1): BasicBlock(
    (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(layer2): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(layer3): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(layer4): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
```

```
      (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     )
    )
    (1): BasicBlock(
     (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
     (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (relu): ReLU(inplace=True)
     (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
     (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
   )
   (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
   (fc): Linear(in_features=512, out_features=5, bias=True)
)
```

[Above model is modified for 1 channel image input and 5 class output from original ResNet18]

FashionMNIST Dataset:

Fashion-MNIST is a dataset of Zalando's article images consisting of **a training set of 60,000** examples and a **test set of 10,000** examples. Each example is a **28x28 grayscale image**, associated with a label from **10 classes**.



Torch Transformations Applied:

- Normalize to (0.5,), (0.5)
- Transform to tensor

Using default PyTorch data-loader function with the dataset path ‘./data’ and used above mentioned transforms for pre-processing. Selected even classes 0,2,4,6,8 as roll number is odd (M21AIE225).

Filtering other than ODD classes 1,3,5,7,9 as roll number is odd (M21AIE225)

```
def filter_dataset(dataset_full):
    # Selecting ODD classes 1,3,5,7,9 as roll number is odd (M21AIE225)
    targets = np.array(dataset_full.targets)
    idx = (targets == 1) | (targets == 3) | (
        targets == 5) | (targets == 7) | (targets == 9)
    dataset_full.targets = np.floor(targets[idx]/2).astype(int)
    dataset_full.data = dataset_full.data[idx]
    dataset_full.classes = [dataset_full.classes[c] for c in [1, 3, 5, 7, 9]]
    return dataset_full
```

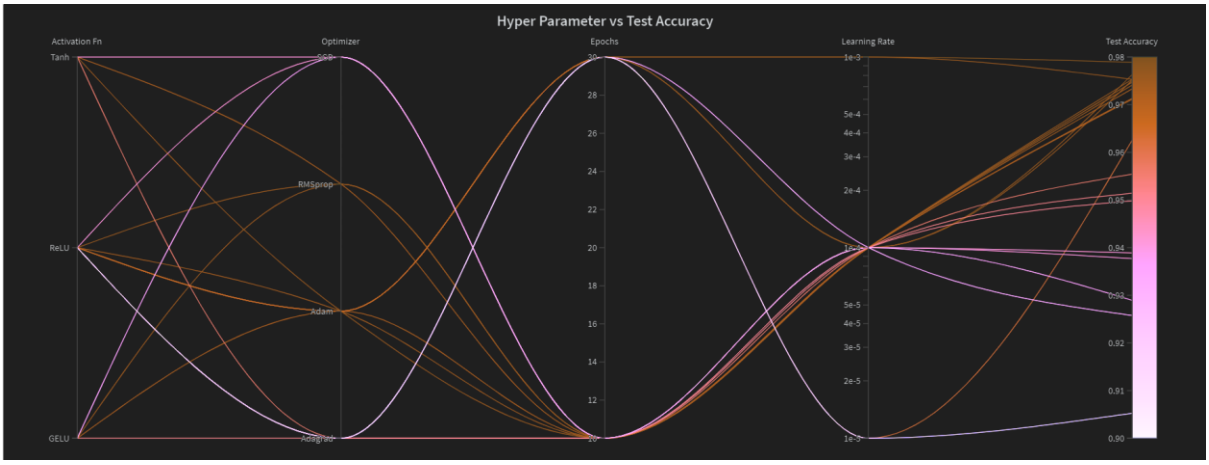
Dataset is **FashionMNIST** as roll number is odd

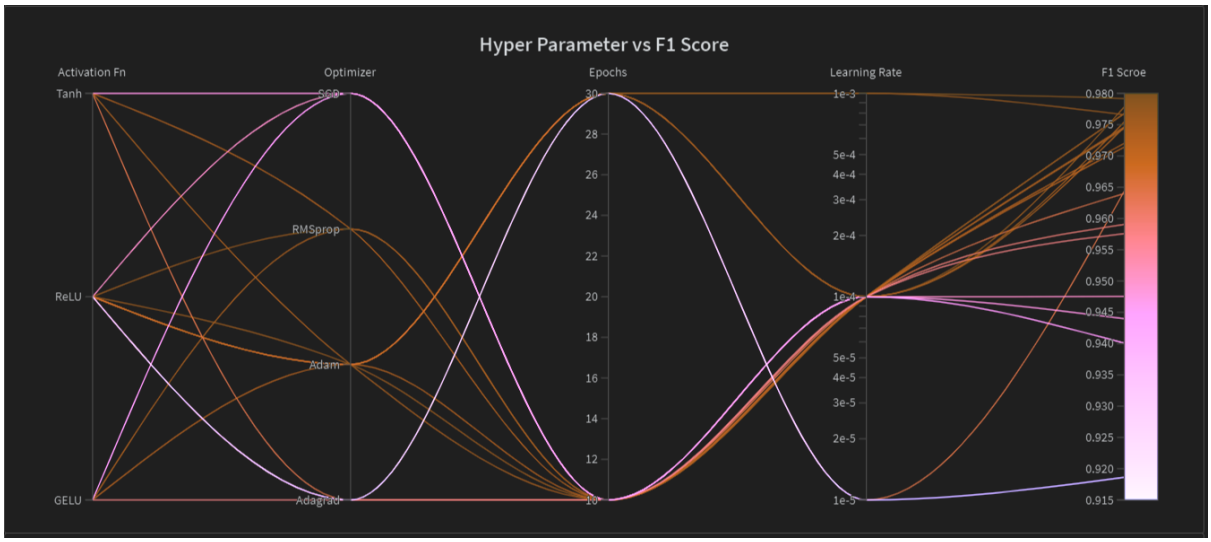
```
[6] transform = T.Compose(
        [T.ToTensor(),
            T.Normalize((0.5,), (0.5,))])
    train_set = torchvision.datasets.FashionMNIST(
        root='./data', train=True, download=True, transform=transform)
    train_set = filter_dataset(train_set)
    trainloader = torch.utils.data.DataLoader(
        train_set, batch_size=32, shuffle=True)
    test_set = torchvision.datasets.FashionMNIST(
        root='./data', train=False, download=True, transform=transform)
    test_set = filter_dataset(test_set)
    testloader = torch.utils.data.DataLoader(test_set, batch_size=32, shuffle=True)
    print(train_set.data.shape)
    print(train_set.classes)
```

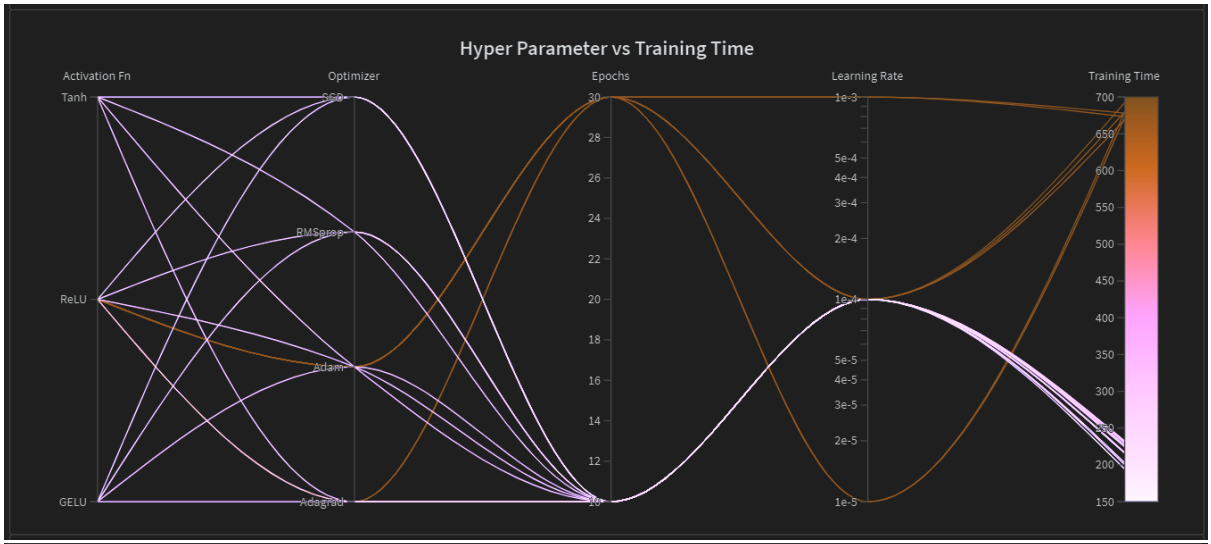
Hyper Parameters Used:

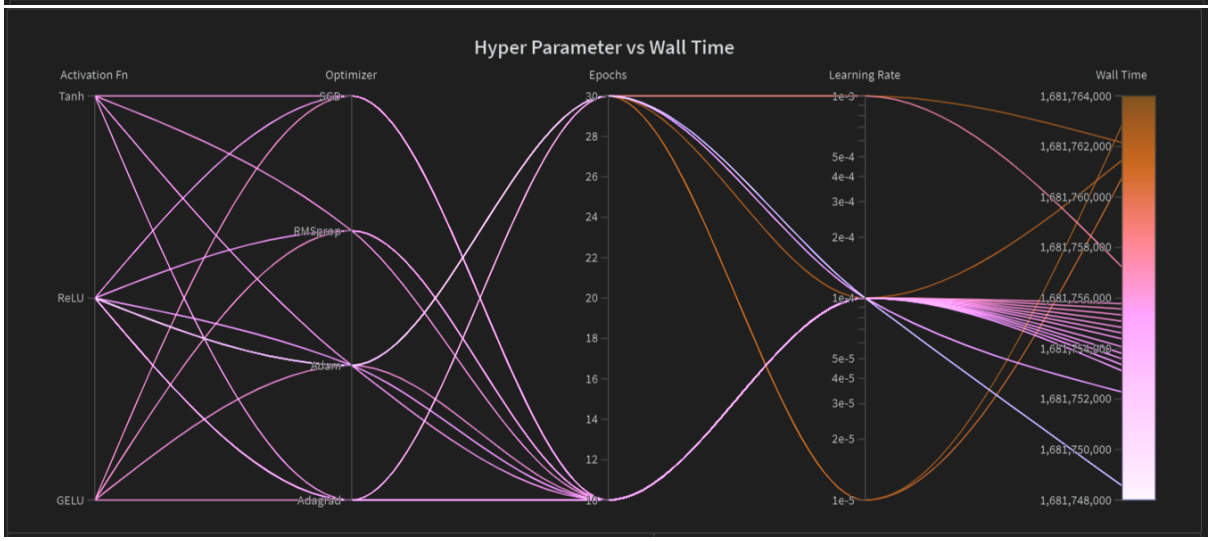
Hyper Parameter	Value
Learning Rate	1e-3, 1e-4, 1e-5
Number of Epochs	10, 15
Batch Size	32
Loss Function	CrossEntropyLoss
Optimizer	'Adam','Adagrad','RMSprop','SGD'
Activation Functions	'ReLU','Tanh','GELU'
Number of Epochs	10,30

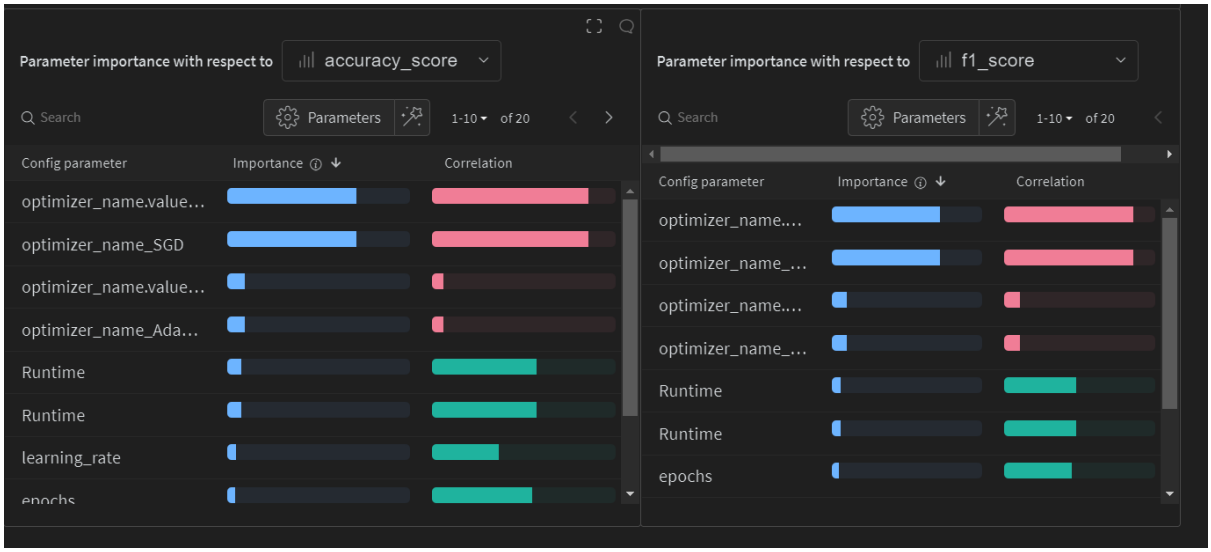
Comparison of the performance of the of different Hyperparameters:



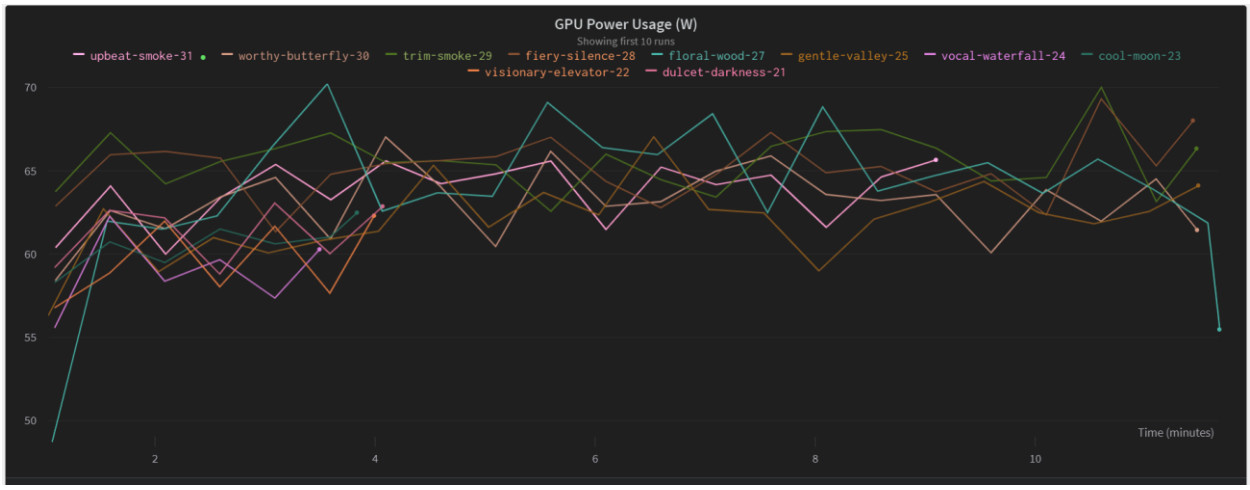


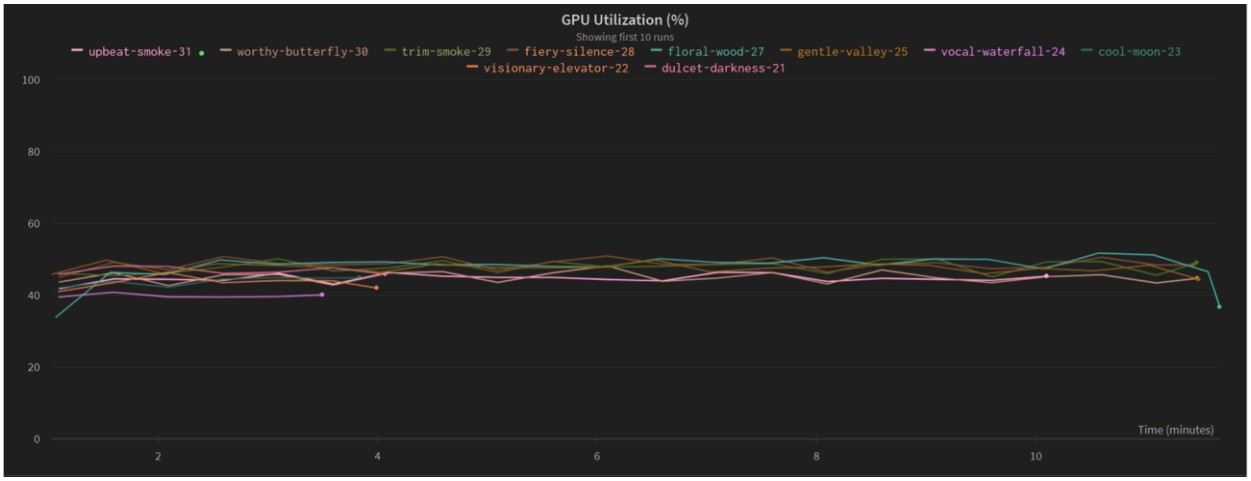

Hyper Parameter vs Training Time
Activation Fn
Optimizer
Epochs
Learning Rate
Training Time
Tanh
ReLU
GELU
SGD
RMSprop
Adam
Adagrad

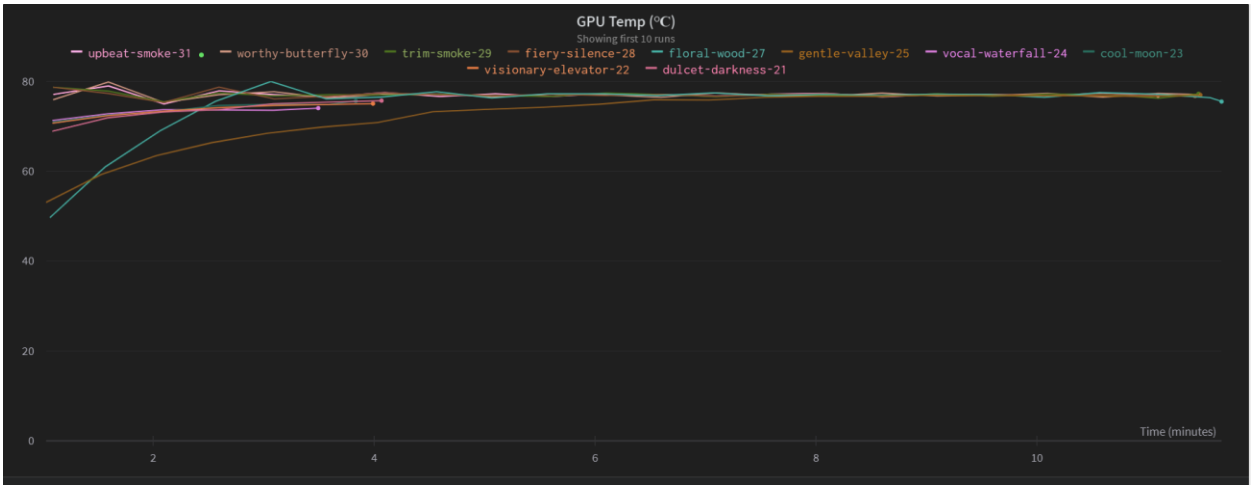

Hyper Parameter vs Wall Time
Activation Fn
Optimizer
Epochs
Learning Rate
Wall Time
Tanh
ReLU
GELU
SGD
RMSprop
Adam
Adagrad

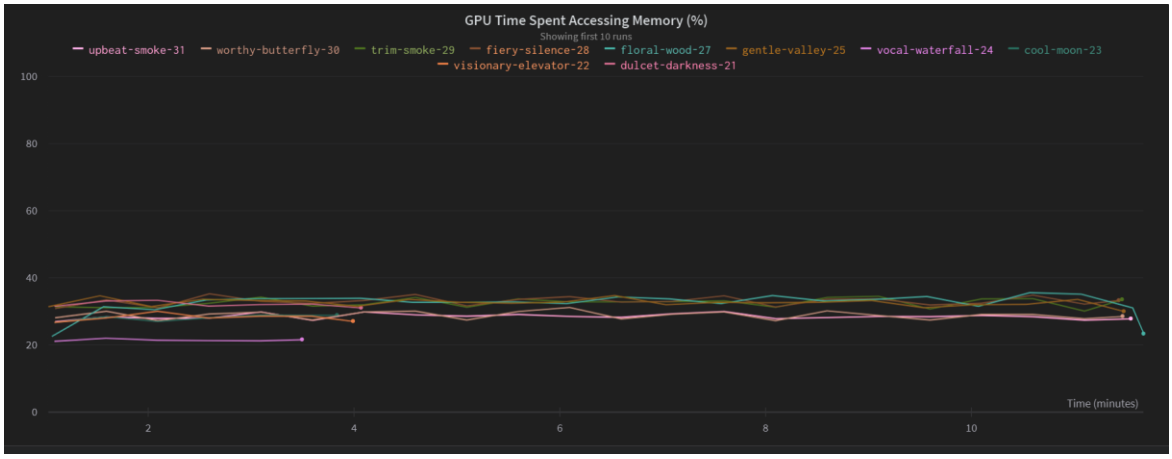

Parameter importance with respect to accuracy_score
Search
Parameters
1-10 of 20
Config parameter
Importance
Correlation
optimizer_name.value...
optimizer_name_SGD
optimizer_name.value...
optimizer_name_Ada...
Runtime
Runtime
learning_rate
epochs
Parameter importance with respect to f1_score
Search
Parameters
1-10 of 20
Config parameter
Importance
Correlation
optimizer_name....
optimizer_name_...
optimizer_name....
optimizer_name_...
Runtime
Runtime
epochs

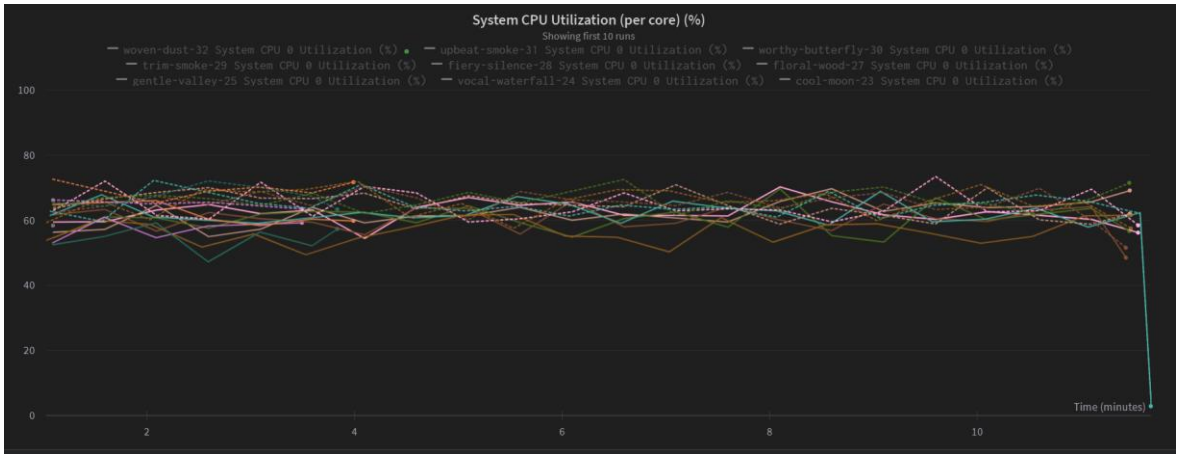
Observations about GPU and Systems

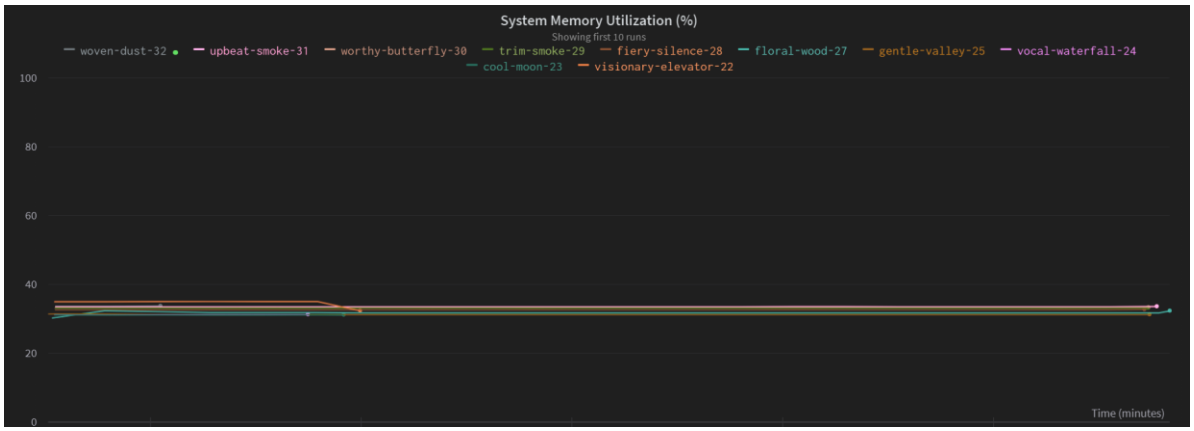


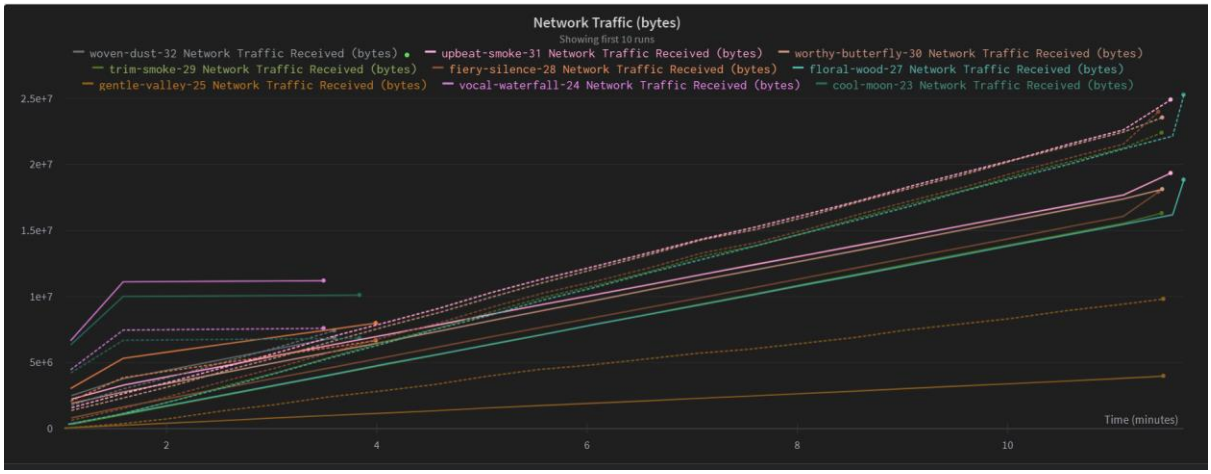




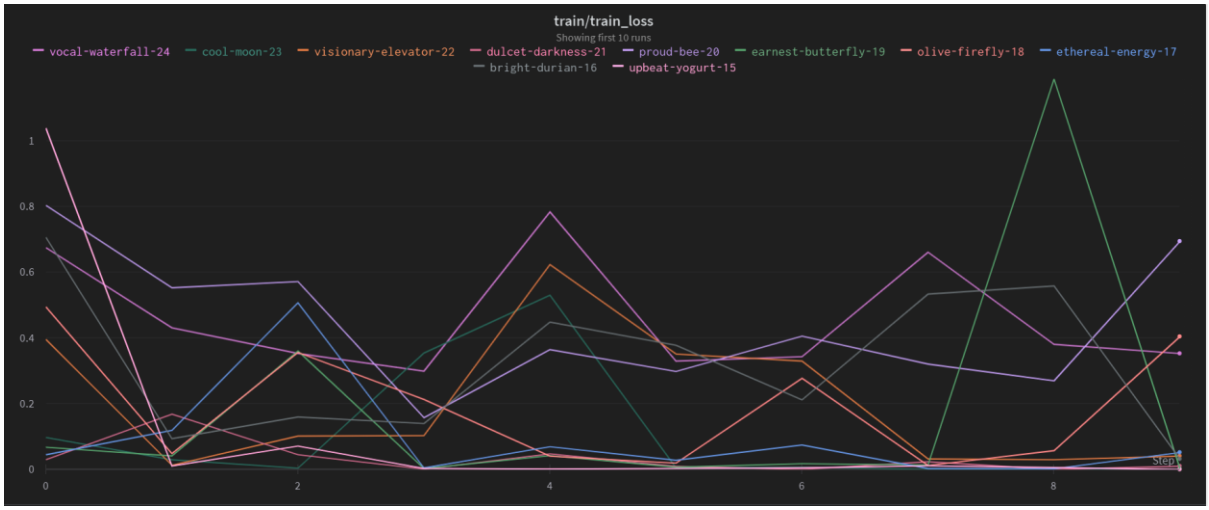

GPU Time Spent Accessing Memory (%)
Showing first 10 runs
upbeat-smoke-31
worthy-butterfly-30
trim-smoke-29
fiery-silence-28
floral-wood-27
gentle-valley-25
vocal-waterfall-24
cool-moon-23
visionary-elevator-22
dulcet-darkness-21
100
80
60
40
20
0
2
4
6
8
10
Time (minutes)

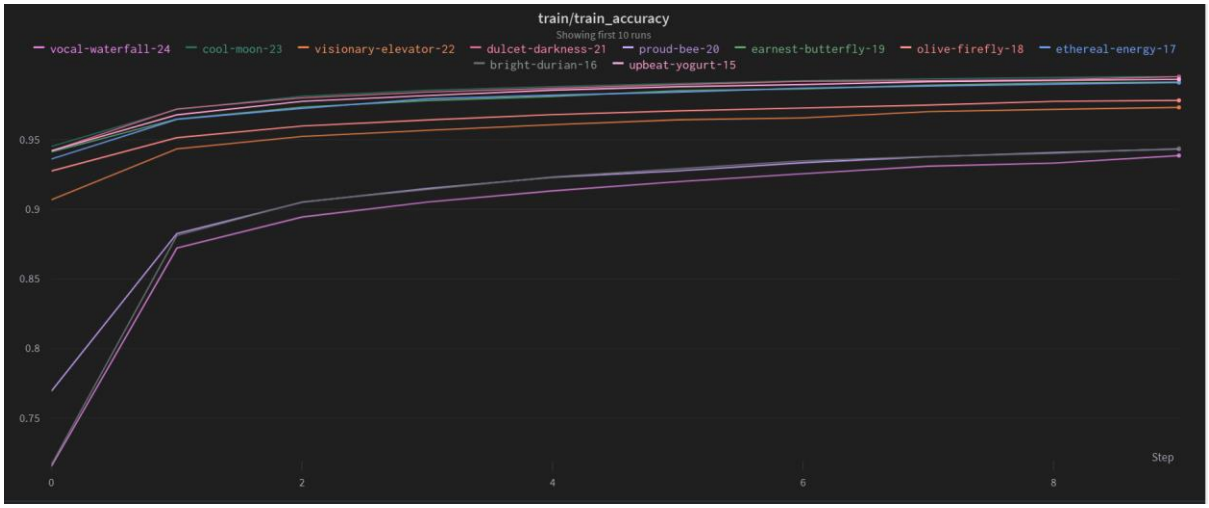

System CPU Utilization (per core) (%)
Showing first 10 runs
woven-dust-32 System CPU 0 Utilization (%)
upbeat-smoke-31 System CPU 0 Utilization (%)
worthy-butterfly-30 System CPU 0 Utilization (%)
trim-smoke-29 System CPU 0 Utilization (%)
fiery-silence-28 System CPU 0 Utilization (%)
floral-wood-27 System CPU 0 Utilization (%)
gentle-valley-25 System CPU 0 Utilization (%)
vocal-waterfall-24 System CPU 0 Utilization (%)
cool-moon-23 System CPU 0 Utilization (%)
100
80
60
40
20
0
2
4
6
8
10
Time (minutes)

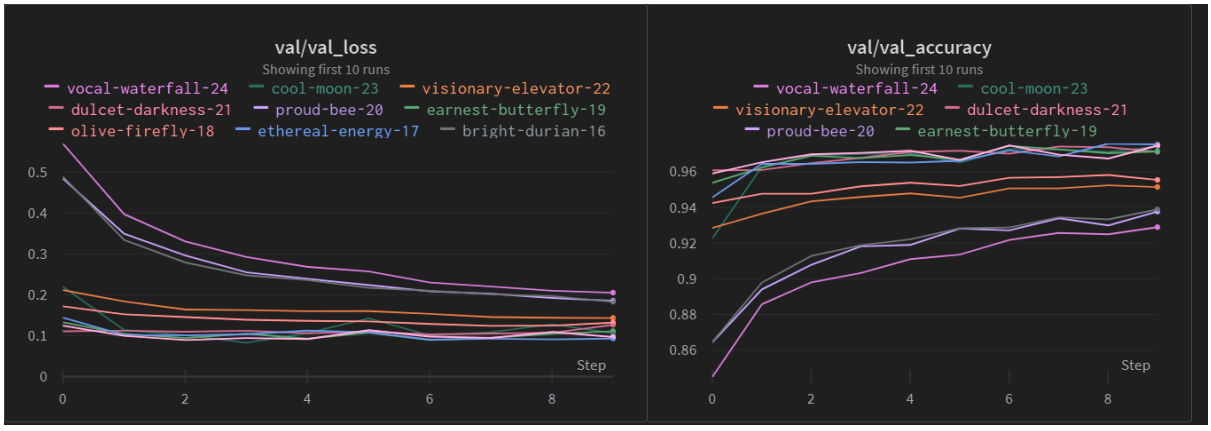

System Memory Utilization (%)
Showing first 10 runs
woven-dust-32
upbeat-smoke-31
worthy-butterfly-30
trim-smoke-29
fiery-silence-28
floral-wood-27
gentle-valley-25
vocal-waterfall-24
cool-moon-23
visionary-elevator-22
100
80
60
40
20
0
Time (minutes)


Network Traffic (bytes)
Showing first 10 runs
woven-dust-32 Network Traffic Received (bytes)
upbeat-smoke-31 Network Traffic Received (bytes)
worthy-butterfly-30 Network Traffic Received (bytes)
trim-smoke-29 Network Traffic Received (bytes)
fiery-silence-28 Network Traffic Received (bytes)
floral-wood-27 Network Traffic Received (bytes)
gentle-valley-25 Network Traffic Received (bytes)
vocal-waterfall-24 Network Traffic Received (bytes)
cool-moon-23 Network Traffic Received (bytes)
2.5e+7
2e+7
1.5e+7
1e+7
5e+6
0
2
4
6
8
10
Time (minutes)

Logs about training accuracy and loss







Visual Result Analysis

	image	pred	target	Trouser	Dress	Sandal	Sneaker	Ankle boot
1		Dress	Dress	3.541e-8	1.	4.626e-8	0.000003319	2.333e-8
2		Sandal	Sandal	0.00007175	0.000007038	0.9994	0.0004611	0.00007008
3		Dress	Dress	0.000002521	1.	0.00001083	8.017e-9	1.188e-8
4		Trouser	Trouser	1.	0.00001023	3.437e-8	0.00000868	5.965e-7
5		Ankle boot	Ankle boot	2.569e-7	1.989e-8	0.00003296	2.501e-9	1.
6		Sneaker	Sneaker	0.000022	4.002e-7	6.602e-7	1.	0.000006513
7		Sneaker	Sneaker	5.892e-8	8.206e-9	1.862e-10	1.	1.158e-8
8		Dress	Dress	3.935e-8	1.	4.669e-10	1.915e-7	9.324e-8

1 - 8 of 384 Export as CSV Columns... Reset Table

	image	pred	target	Trouser	Dress	Sandal	Sneaker	Ankle boot
9		Dress	Dress	6.328e-8	1.	3.462e-7	2.717e-8	5.454e-8
10		Dress	Dress	0.000009537	1.	5.466e-7	3.541e-8	7.186e-8
11		Ankle boot	Ankle boot	0.000003306	6.561e-9	2.163e-8	9.142e-9	1.
12		Dress	Dress	0.00001178	0.9999	0.000004091	0.000001276	0.00003757
13		Trouser	Trouser	1.	0.000005187	0.000003085	0.000002153	5.766e-7
14		Ankle boot	Ankle boot	3.513e-9	7.118e-8	3.964e-9	9.847e-10	1.
15		Trouser	Trouser	1.	9.351e-7	9.194e-8	0.00000143	0.000002493
16		Sneaker	Ankle boot	2.149e-7	0.000006007	0.00000258	1.	0.00001376

9 - 16 of 384 Export as CSV Columns... Reset Table

	image	pred	target	Trouser	Dress	Sandal	Sneaker	Ankle boot
17		Sneaker	Sneaker	0.00005446	0.00003148	0.00001112	0.9941	0.005793
18		Trouser	Trouser	1.	8.619e-7	0.000001735	0.000001186	8.343e-7
19		Trouser	Trouser	1.	4.664e-7	0.000001262	4.051e-7	3.270e-7
20		Sneaker	Sneaker	0.00002636	0.0004817	0.01321	0.9863	0.0000177
21		Sandal	Sandal	1.708e-7	0.00001862	1.	0.000003028	1.245e-7
22		Sandal	Sandal	0.00004185	5.032e-7	0.9999	0.00008542	0.00000186
23		Trouser	Trouser	1.	6.315e-7	1.091e-8	1.318e-7	6.688e-9
24		Sandal	Sandal	1.224e-7	7.552e-9	1.	1.233e-8	1.469e-9

17 - 24 of 384 Export as CSV Columns... Reset Table

References:

1. Lectures of DL-Ops class by Professors
2. Implementation Session by Teaching Assistants