University at Buffalo
School of Engineering and Applied Sciences

CSE 676 – Deep Learning
Spring 2023

Assignment #1

Prepared by

Sakshamdeep Singh - 50475857

April 7, 2022

TABLE OF CONTENTS

Submission Directory Structure

```
└── ssingh86_assignment1/
        ├── report.pdf
        ├── task_1/
                ├── train.py
                ├── test.py
                ├── graphs.py
                ├── results.py
                ├── util.py
                ├── model.py
                ├── task_1_new_model.pth            (will be updated when train.py runs)
                ├── task_1_my_model.pth             (submitted model with best performance)
                ├── task_1_my_run/                  (tensorboard log folder for my run)
                        └── events.out.tfevents…
                ├── task_1_new_run/                 (will be created when train.py runs)
                        └── events.out.tfevents…
                └── kernels_and_feature_maps/
                        ├── sample_img.png
                        ├── kernels_img.png
                        ├── feature_map_0.png
                        ├── feature_map_1.png
                              …
                        └── feature_map_15.png
        ├── task_2/
                ├── train.py
                ├── test.py
                ├── graphs.py
                ├── results.py
                ├── util.py
                ├── model.py
                ├── task_2_new_model.pth            (will be updated when train.py runs)
                ├── task_2_my_model.pth             (submitted model with underfitting)
                ├── task_2_my_run/                  (tensorboard log folder for my run)
                        └── events.out.tfevents…
                └── task_2_new_run/                 (will be created when train.py runs)
                        └── events.out.tfevents…
        └── task_3/
                ├── train.py
                ├── test.py
                ├── graphs.py
                ├── results.py
                ├── util.py
                ├── model.py
                ├── task_3_new_model.pth            (will be updated when train.py runs)
                ├── task_3_my_model.pth             (submitted model with overfitting)
                ├── task_3_my_run/                  (tensorboard log folder for my run)
                        └── events.out.tfevents…
                └── task_3_new_run/                 (will be created when train.py runs)
                        └── events.out.tfevents…
```

Testing Guidelines

Task1:

1. Run train.py to generate a new CNN model which will be saved as 'task_1_new_model.pth'. Tensorboard logs will be created under the directory 'task_1_new_run'.
2. Run test.py file to test the new model saved as 'task_1_new_model.pth' on the test set.
3. Run graphs.py file to check the tensorboard logs for originally trained model. It picks up the logs from the directory named 'task_1_my_run'. A browser will open showing the tensorboard graphs. Kernels and feature maps can be viewed under 'images' tab. Please close the process at the port before moving on to the next task.
4. Run results.py file to verify the performance of the originally trained model. The original model is submitted as 'task_1_my_model.pth'.

Task2:

1. Run train.py to generate a new CNN model which will be saved as 'task_2_new_model.pth'. Tensorboard logs will be created under the directory 'task_2_new_run'.
2. Run test.py file to test the new model saved as 'task_2_new_model.pth' on the test set.
3. Run graphs.py file to check the tensorboard logs for originally trained model. It picks up the logs from the directory named 'task_2_my_run'. A browser will open showing the tensorboard graphs. Please close the process at the port before moving on to the next task.
4. Run results.py file to verify the performance of the originally trained model. The original model is submitted as 'task_2_my_model.pth'.

Task3:

1. Run train.py to generate a new CNN model which will be saved as 'task_3_new_model.pth'. Tensorboard logs will be created under the directory 'task_3_new_run'.
2. Run test.py file to test the new model saved as 'task_3_new_model.pth' on the test set.
3. Run graphs.py file to check the tensorboard logs for originally trained model. It picks up the logs from the directory named 'task_3_my_run'.
4. Run results.py file to verify the performance of the originally trained model. The original model is submitted as 'task_3_my_model.pth'.

TASK 1

For the texture classification task, a CNN model was built and trained to classify 2D images into 47 distinct texture classes.

Input Image size: During the training and validation phase, all images were randomly cropped and resized to a size of 256 x 256 x 3. However, during the testing phase, all images were resized to the same dimensions without any cropping.

Data Split: Standard data augmentation was done to increase the size of training samples from 1800 to 11280.

Set	Number of Samples
Training	11280
Validation	1880
Test	1880

Hyperparameters:

Parameter	Value
Learning rate	0.001
Epochs	50
Optimizer	Adam
Loss	Crossentropy
Batch Size	128

Model Architecture: A CNN model was built using 5 CNN layers and 1 Linear layer.

```
Layer (type:depth-idx)          Output Shape            Param #
===============================================================
├─Conv2d: 1-1                   [-1, 16, 256, 256]      1,216
├─MaxPool2d: 1-2                [-1, 16, 128, 128]      --
├─Conv2d: 1-3                   [-1, 32, 128, 128]      12,832
├─MaxPool2d: 1-4                [-1, 32, 64, 64]        --
├─Conv2d: 1-5                   [-1, 64, 64, 64]        51,264
├─MaxPool2d: 1-6                [-1, 64, 32, 32]        --
├─Conv2d: 1-7                   [-1, 128, 32, 32]       204,928
├─MaxPool2d: 1-8                [-1, 128, 16, 16]       --
├─Conv2d: 1-9                   [-1, 256, 16, 16]       295,168
├─MaxPool2d: 1-10               [-1, 256, 8, 8]         --
├─Linear: 1-11                  [-1, 256]               4,194,560
├─Linear: 1-12                  [-1, 47]                12,079
===============================================================
Total params: 4,772,047
Trainable params: 4,772,047
```

Non-trainable params: 0
Total mult-adds (M): 787.49

Input size (MB): 0.75
Forward/backward pass size (MB): 15.50
Params size (MB): 18.20
Estimated Total Size (MB): 34.46

Evaluation:

	Loss (cross entropy)	Accuracy (%)
Training	1.421	59.82
Validation	2.735	39.52
Test	2.735	39.57

Graphs: Following 4 graphs were creating to illustrate the accuracies and losses for both the training and validation data.

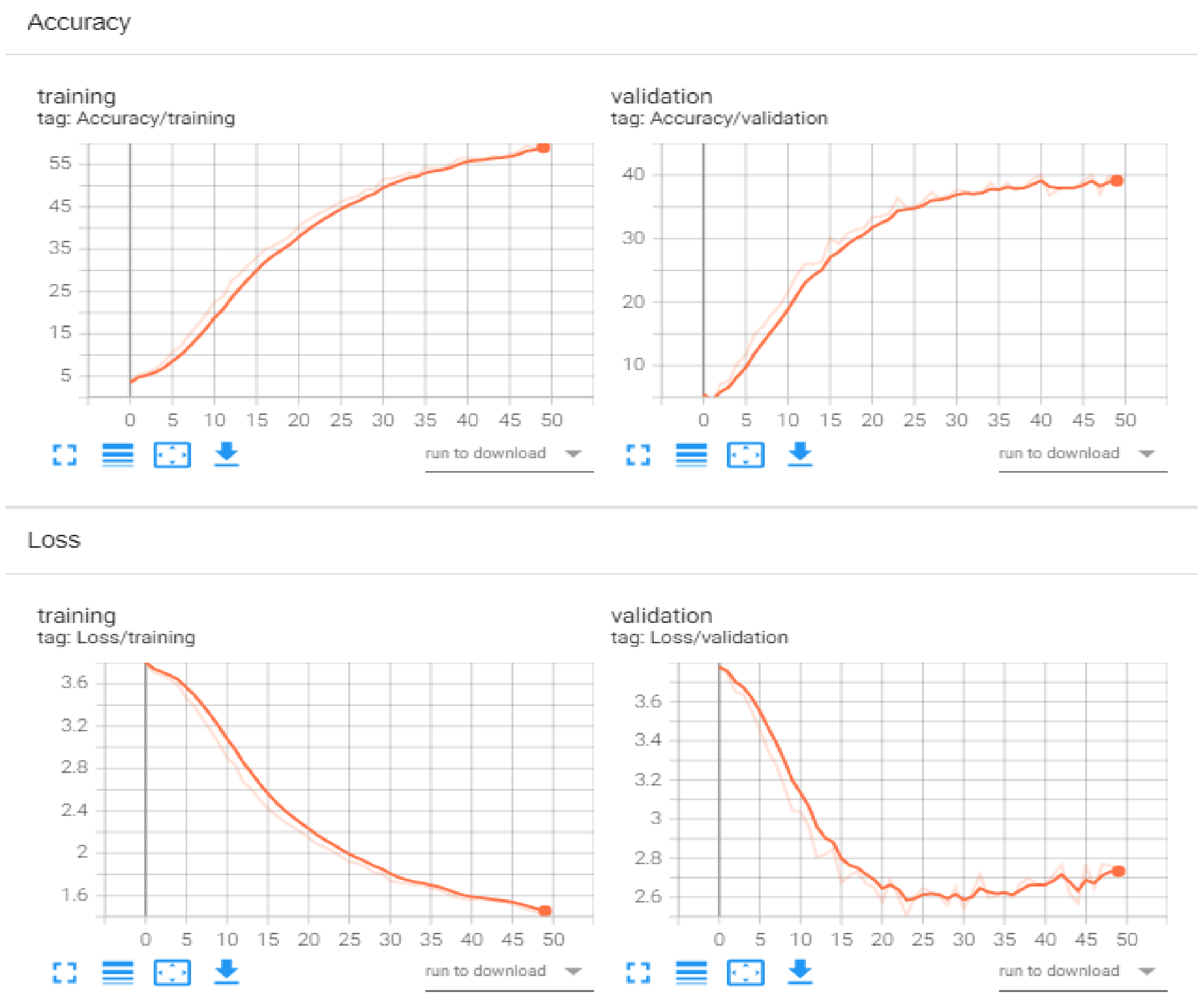


Figure 1. Training accuracy (top-left), Validation Accuracy (top-right), Training Loss (bottom-left) and Validation Loss (bottom-right)

Conv1 Kernels: The following 16 kernels were created from the training dataset at the first training layer.

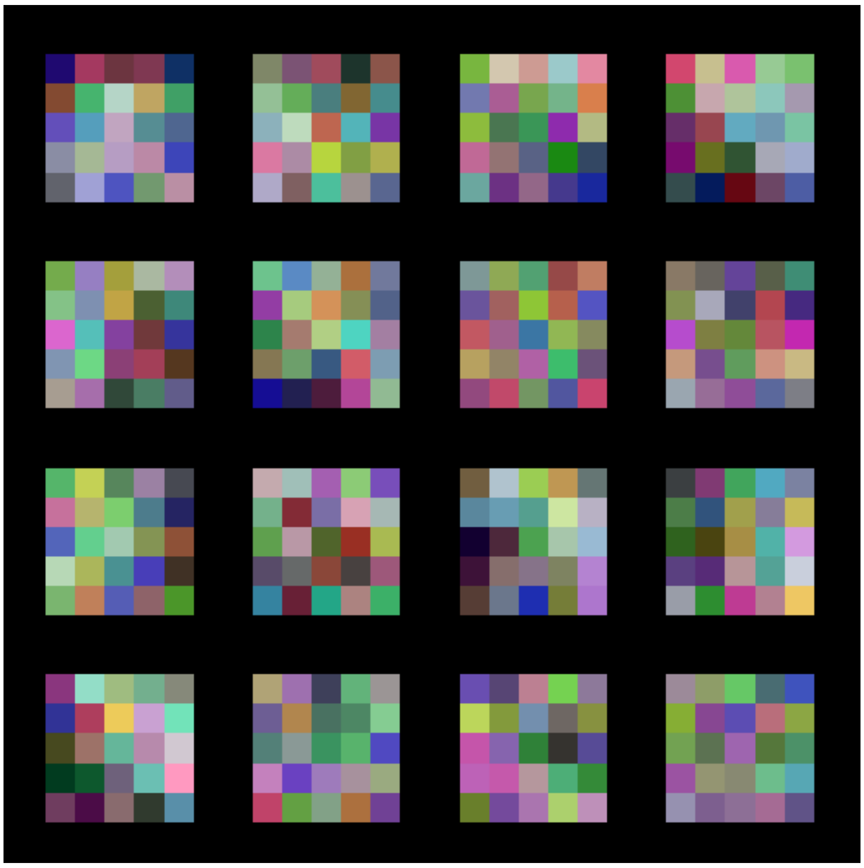
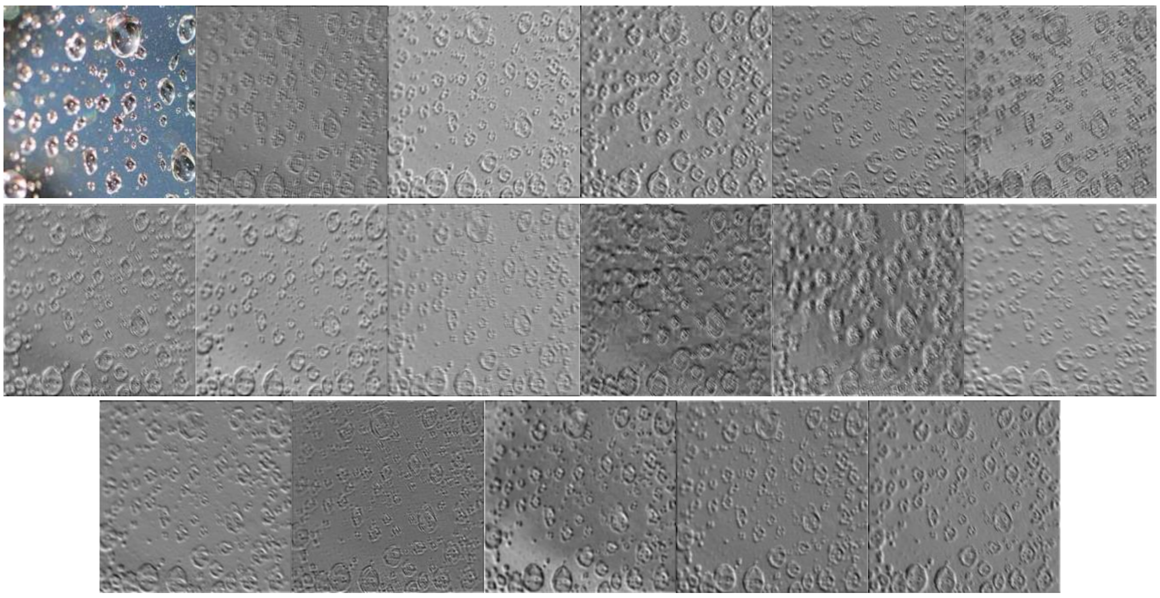


Figure 2. Visualized 16 kernels.

Conv1 feature maps: Applying the above 16 filters on a sample image the following 16 feature maps were generated.



Observations:

- The model was trained for a total of 50 epochs with a total training time of 2 hours and 31 minutes.
- Based on the generated graphs, the training error continued to decrease over the course of the 50 epochs.
- However, the validation/generalization error decreased only during the first 30 epochs before gradually increasing, indicating a slight degree of overfitting.
- Various hyperparameters were tuned, and different models were experimented with to achieve good performance.
- Data augmentation was done using the standard ImageNet, CIFAR10 and SVHN policies available in PyTorch. Data augmentation helps in training large models which require a substantial amount of data.

Task 2 - Underfitting

When a model is too simple to learn from the training data, we refer to it as an underfitting model. In such cases, both the training error and the generalization/validation error tend to be excessively high.

Input Image size: During the training and validation phase, all images were randomly cropped and resized to a size of 256 x 256 x 3. However, during the testing phase, all images were resized to the same dimensions without any cropping.

Data Split:

Set	Number of Samples
Training	1880
Validation	1880
Test	1880

Hyperparameters:

Parameter	Value
Learning rate	0.001
Epochs	20
Optimizer	Adam
Loss	Crossentropy
Batch Size	32

Model Architecture: A simple CNN model was built using 1 CNN layer and 1 Linear layer.

```
Layer (type:depth-idx)              Output Shape              Param #
======================================================================
├─Conv2d: 1-1                       [-1, 1, 256, 256]         76
├─MaxPool2d: 1-2                    [-1, 1, 8, 8]             --
├─Linear: 1-3                       [-1, 47]                  3,055
======================================================================
Total params: 3,131
Trainable params: 3,131
Non-trainable params: 0
Total mult-adds (M): 4.92
======================================================================
Input size (MB): 0.75
Forward/backward pass size (MB): 0.50
Params size (MB): 0.05
Estimated Total Size (MB): 1.26
======================================================================
```

Evaluation:

	Loss (cross entropy)	Accuracy (%)
Training	3.726	4.57
Validation	3.755	4.46
Test	3.764	4.14

Graphs:

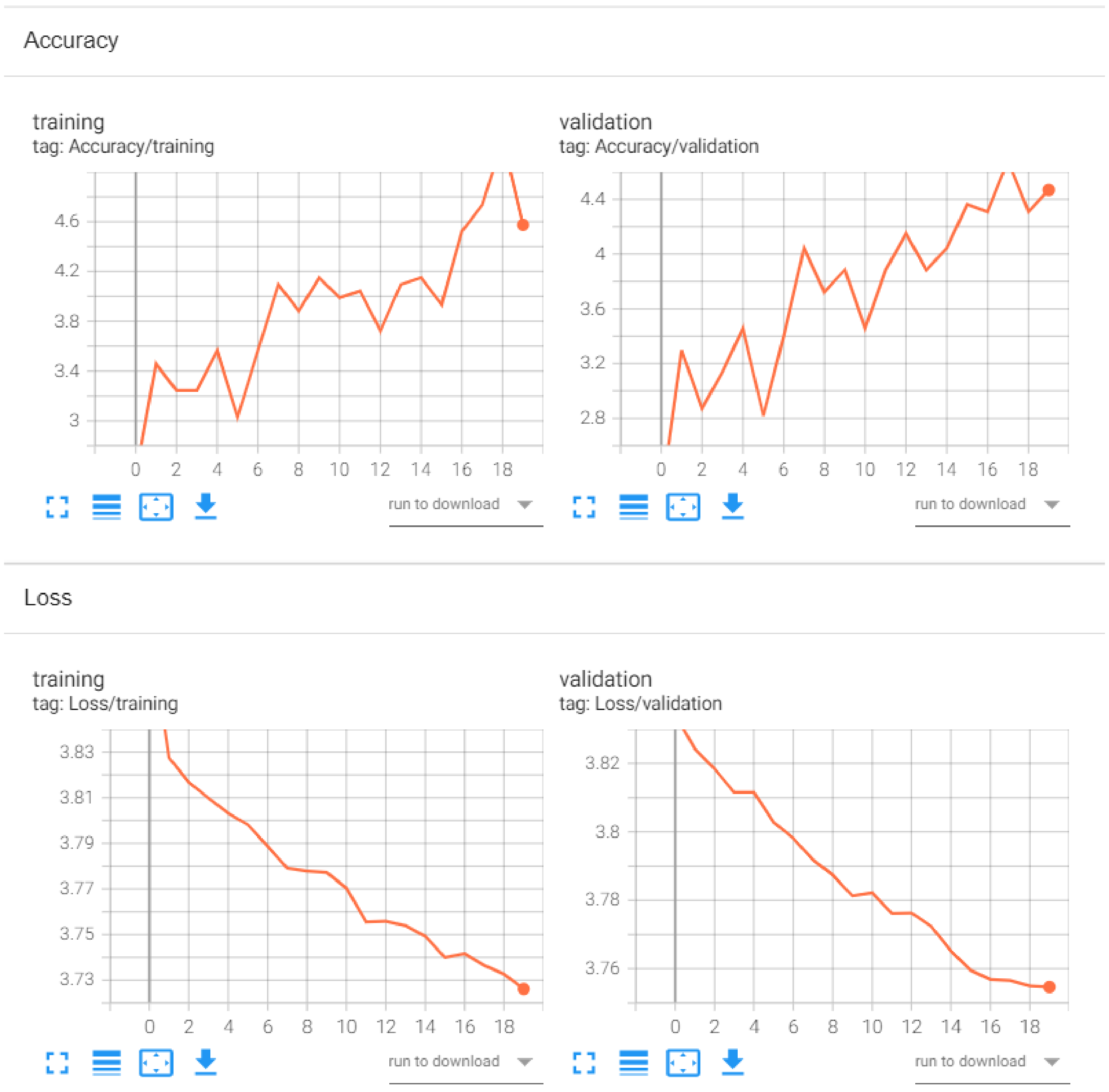


Figure 3.Training accuracy (top-left), Validation Accuracy (top-right), Training Loss (bottom-left) and Validation Loss (bottom-right)

Observations:

- The model was trained for a total of 20 epochs.
- According to the generated graphs, there is minimal fluctuation in the training and generalization error throughout the 20 epochs of training, suggesting that the model is too simplistic to learn effectively thereby implying underfitting.

Task 3 - Overfitting

An overfitting model occurs when a model is excessively trained on the training dataset, leading to a low training error but high generalization error.

Input Image size: During the training and validation phase, all images were randomly cropped and resized to a size of 256 x 256 x 3. However, during the testing phase, all images were resized to the same dimensions without any cropping.

Data Split:

Set	Number of Samples
Training	1880
Validation	1880
Test	1880

Hyperparameters:

Parameter	Value
Learning rate	0.001
Epochs	100
Optimizer	Adam
Loss	Crossentropy
Batch Size	64

Model Architecture: A CNN model was built using 3 CNN layers and 1 Linear layer.

```
Layer (type:depth-idx)            Output Shape              Param #
==========================================================================
├─Conv2d: 1-1                     [-1, 16, 256, 256]        448
├─MaxPool2d: 1-2                  [-1, 16, 128, 128]        --
├─Conv2d: 1-3                     [-1, 32, 128, 128]        4,640
├─MaxPool2d: 1-4                  [-1, 32, 64, 64]          --
├─Conv2d: 1-5                     [-1, 64, 64, 64]          18,496
├─MaxPool2d: 1-6                  [-1, 64, 32, 32]          --
├─Linear: 1-7                     [-1, 47]                  3,080,239
==========================================================================
Total params: 3,103,823
Trainable params: 3,103,823
Non-trainable params: 0
Total mult-adds (M): 182.39
==========================================================================
Input size (MB): 0.75
Forward/backward pass size (MB): 14.00
Params size (MB): 11.84
Estimated Total Size (MB): 26.59
==========================================================================
```

Evaluation:

	Loss (cross entropy)	Accuracy (%)
Training	1.116	70.00
Validation	5.547	20.63
Test	5.251	21.96

Graphs:

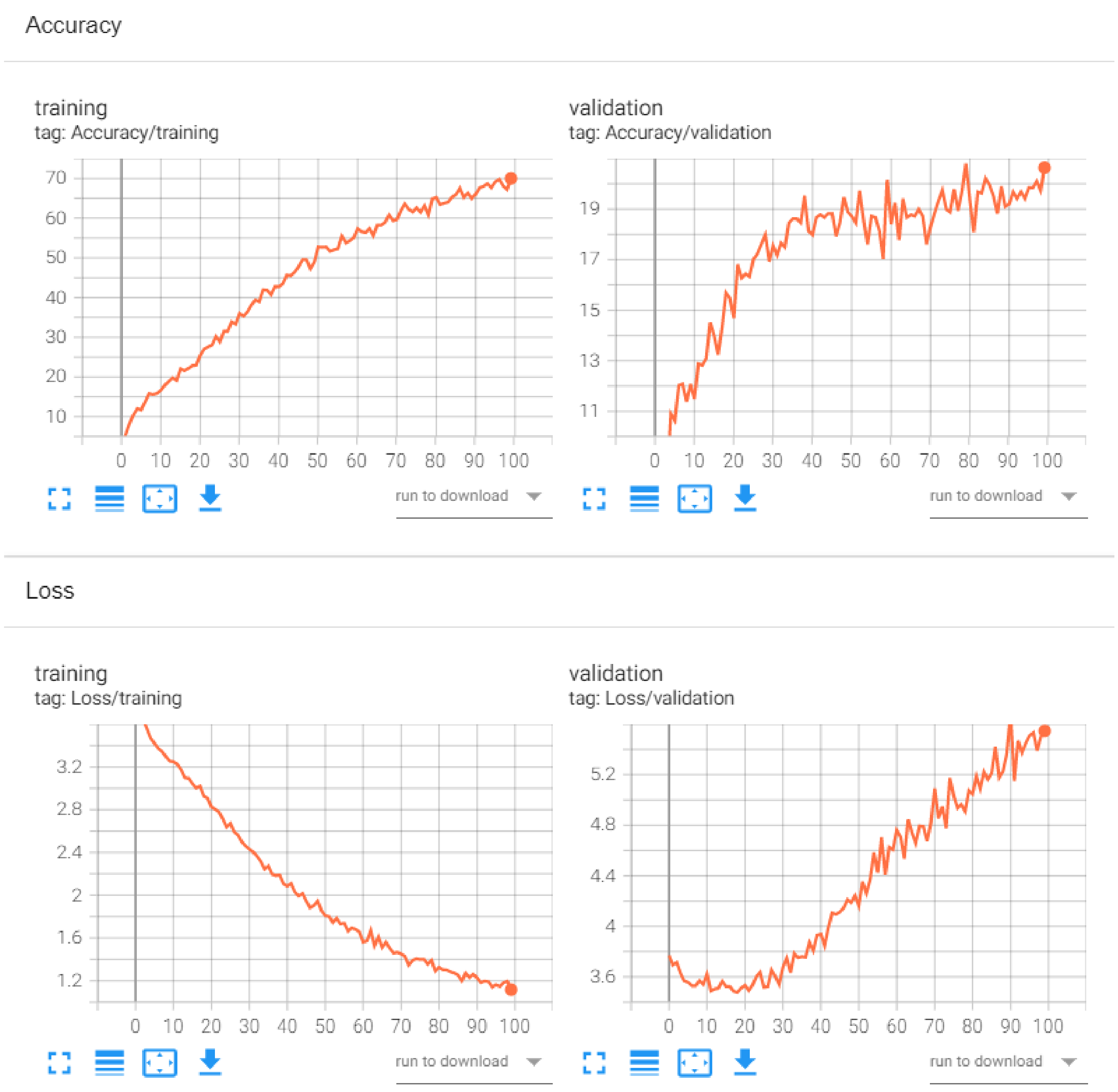


Figure 4.Training accuracy (top-left), Validation Accuracy (top-right), Training Loss (bottom-left) and Validation Loss (bottom-right)

Observations:

- The model was trained for a total of 100 epochs with a total training time of 31 minutes.
- During the 100 epochs of training, the training loss consistently decreased. However, the generalization loss exhibited a decreasing trend only during the first 15 epochs and then began to increase, indicating overfitting.

References

- https://pytorch.org/tutorials/beginner/basics/data_tutorial.html
- https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html
- https://pytorch.org/vision/stable/transforms.html
- https://pytorch.org/vision/stable/auto_examples/plot_transforms.html#sphx-glr-auto-examples-plot-transforms-py
- https://pytorch.org/docs/stable/tensorboard.html
- https://pytorch.org/vision/stable/index.html
- https://pytorch.org/vision/stable/generated/torchvision.datasets.DTD.html#torchvision.datasets.DTD
- https://stackoverflow.com/questions/58151507/why-pytorch-officially-use-mean-0-485-0-456-0-406-and-std-0-229-0-224-0-2
- https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html