Semi-supervised Leaf Pathology Classification

Singh, Manmeet
Charles W. Davidson College of Engineering
San Jose State University
San Jose, California
manmeet.singh01@sjsu.edu

Vadgama, Chinmay
Charles W. Davidson College of Engineering
San Jose State University
San Jose, California
chinmay.vadgama@sjsu.edu

Asha Balshiram, Aher
Charles W. Davidson College of Engineering
San Jose State University
San Jose, California
ashabalshiram.aher@sjsu.edu

Ronak Mehta
Charles W. Davidson College of Engineering
San Jose State University
San Jose, California
ronak.mehta@sjsu.edu

***Abstract*—This e**

***Keywords*—semi-supervised, simclr, contrastive, learning, augmentation, wide, deep**

I. Introduction

Misdiagnosis of the many diseases impacting agricultural crops can lead to misuse of chemicals leading to the emergence of resistant pathogen strains, increased input costs, and more outbreaks with significant economic loss and environmental impacts. Current disease diagnosis based on human scouting is time-consuming and expensive, and although computer-vision based models have the promise to increase efficiency, the great variance in symptoms due to age of infected tissues, genetic variations, and light conditions within trees decreases the accuracy of detection.

We explore plant disease detection by following computer vision based deep learning architectures on a large plant image dataset. We explore traditional supervised architectures such as ResNet-50, InceptionResNetV2 etc. and compare it with new self-supervised learning approaches such as SimCLR architectures. Generally there is very little labeled dataset available for plants and specific diseases. Using self-supervised techniques such as SimCLR could provide ground for using large unlabeled dataset on top of learning from small labeled dataset. New data can be collected with help of drone based cameras without any manual help and that data could be used as an unlabeled source for SimCLR network. This could get us more generalization and model adaptation can be increased even if we don't have higher accuracy than supervised techniques.

II. Related Work

Plant disease detection using deep learning and computer vision is an ongoing research topic and a lot of research has been done in the past. Lots of different approaches have been implemented including traditional computer vision algorithms like image segmentation, Support Vector Machines(SVM),surf using Artificial Neural Networks(ANN). Variety of CNN models such as AlexNet, GoogleNet, VGGNet etc have been used in Deep Learning based plant disease classification. It often comes as a challenge to find perfect dataset size, lots of classes in multi class classification need careful tuning of hyperparameter in order to avoid overfitting.

In our study, we use a deep learning method for plant disease recognition, we have simclrv2 which builds on top of contrastive learning based simclr. This approach uses heavy data augmentation followed by learning of the non linear transformation between representation and the contrastive loss which substantially improves the quality of learned representations. In conclusion, the contrastive learning benefits from larger batch sizes compared to supervised learning. By amalgamating of these methodologies, we are able to achieve high performance with small amounts of labeled data.

III. Data

Initially,We were exploring the medical domain and chose Alzheimer disease detection. The Alzheimer classification was carried out on the OASIS-1 Alzheimer as well as Alzheimer's severity dataset. OASIS-1 is a cross-sectional MRI data in young,middle aged, non demented and demented older adults. OASIS 1 has 416 subjects with 434 MRI scans. We carried out data engineering and data augmentation on this dataset and implemented SimCLR. There was a scope for accuracy improvement.

We carried out our classification on Plant disease detection. Two datasets were used in order to carry out the experiment.
a) Plant village b) Plant pathology.
Plant pathology dataset has images of apple leaves and those are distinguished between healthy , scab & rust.Plant pathology has a total 3642 samples of above combination leaves.Plant pathology dataset contains different plant leaves including tomato,potato and bell paper.

IV. Methods

For the semi-supervised learning approach, we used simclrv2 which builds on top of contrastive learning based simclr. The contrastive approaches use heavy data augmentation followed by learning of the non linear transformation between representation and the contrastive loss which substantially improves the quality of learned representations. Lastly, the contrastive learning benefits from larger batch sizes compared to supervised learning. By combining these methodologies, we are able to achieve high performance with small amounts of labeled data.

Specific to simclrv2, a large amount of unlabeled data is used for unsupervised pre-training followed by supervised fine-tuning. This paradigm uses unlabeled data in a task-agnostic way initially. The key ingredient of this approach are the deep and wide networks during pretraining and fine-tuning. The fewer the labels in this approach, the more the pre-training benefits from this approach.

In our first approach, we used simclrv2 to perform unsupervised pre training on the OASIS-1 dataset. Next, we fine-tuned using the Alzheimer's severity dataset from Kaggle. After exhausting the possibilities for this approach, we pivoted to pre-training with plant village dataset and fine-tuned on the plant pathology dataset.

We trained our models on a combination of using the P100 GPU on the HPC machine, as well as the Google Colab environment.

V. Experiments

Initially, we used SimCLRv2 to pretrain on the OASIS-1 dataset. The idea was to build an embedding space that could then be used to finetune on the Alzheimer's severity dataset. This dataset included images from 4 classes ranging from normal to high severity.

After pre-training, we did not receive great accuracy during this methodology, likely related to the fact that we need a huge pre-training dataset to gain real benefits of the contrastive approach.

Next, we tried to pre-train on plant village dataset which is a multiclass classification dataset with 38 classes. Each class contains the leaf type as well as the disease/condition being experienced by the leaf. We achieved better accuracy after fine-tuning the plant village dataset on plant pathology dataset. We followed the standard procedure for fine tuning where we removed the first projection head, attached a fine-tuning layer to the pretrained model and backpropagation of the loss for this dataset. We were about to achieve 85% accuracy with this approach.

Detailed experiment results are shown in Appendix section.

VI. Conclusion

In this paper, we study how to develop contrastive learning of visual representations methods to accurately detect plant disease from plant leaf images.For the ease of open research in this area, we build SimCLR model using plant pathology dataset which contains leaf images with different combination of the leaf situations which includes healthy, scab & rust. This dataset is publicly available to the date. In Deep learning models, overfitting is high risk for data-hungry deep learning models. In order to tackle this problem we develop a model which is data efficient and is able to alleviate data deficiency. In this paper we propose an approach of a simple framework for contrastive learning of visual representations that learns expressive and unbiased visual feature representations that are robust to overfitting.In order to show effectiveness of our methods, we carry out extensive experiments.

References

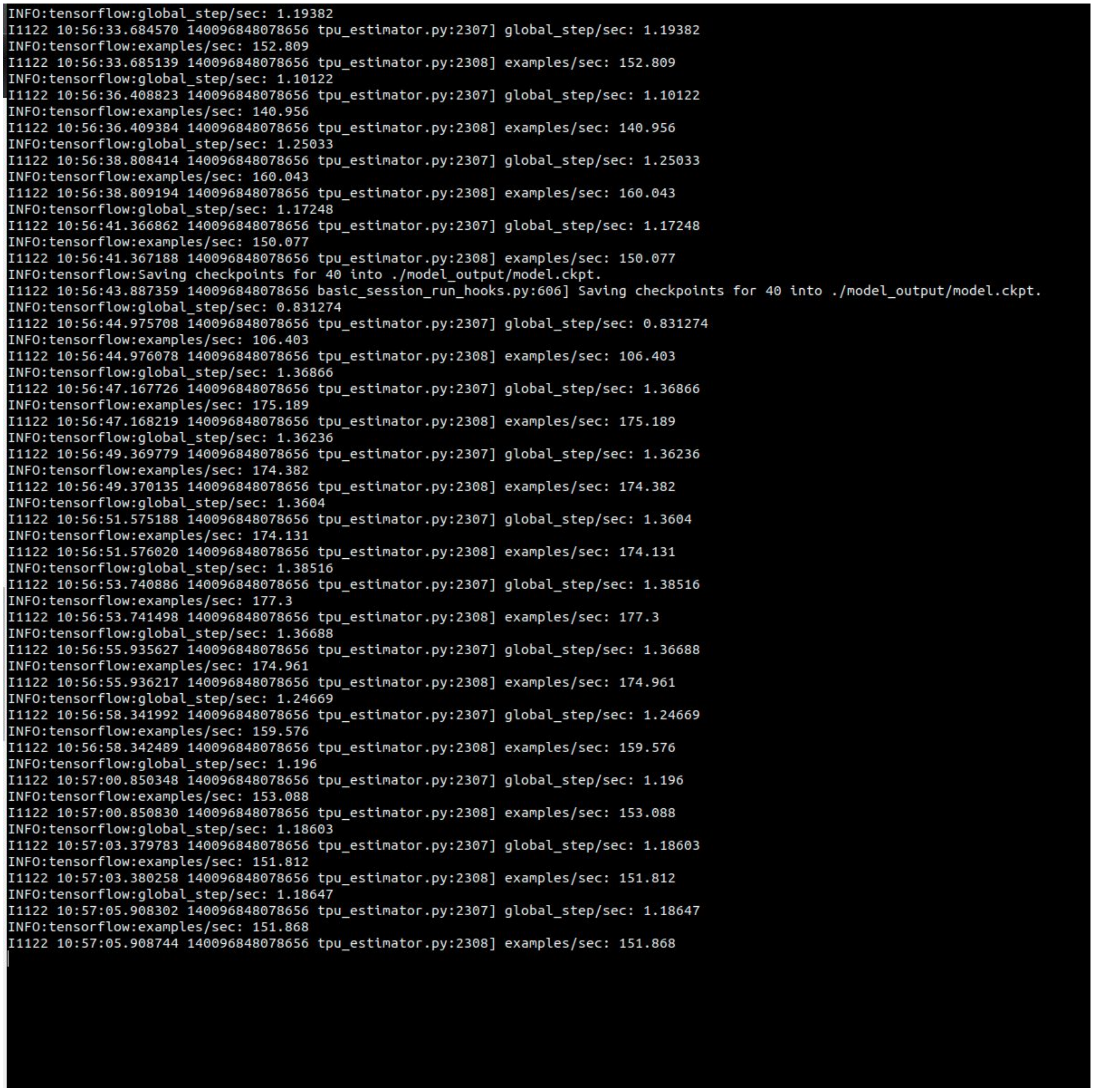
[1] Ranjita Thapa1 , Noah Snavely2 , Serge Belongie2 , Awais Khan*1,"The Plant Pathology 2020 challenge dataset to classify foliar disease of apples" arXiv:2004.11958

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, & Geoffrey Hinton. (2020). A Simple Framework for Contrastive Learning of Visual Representations.

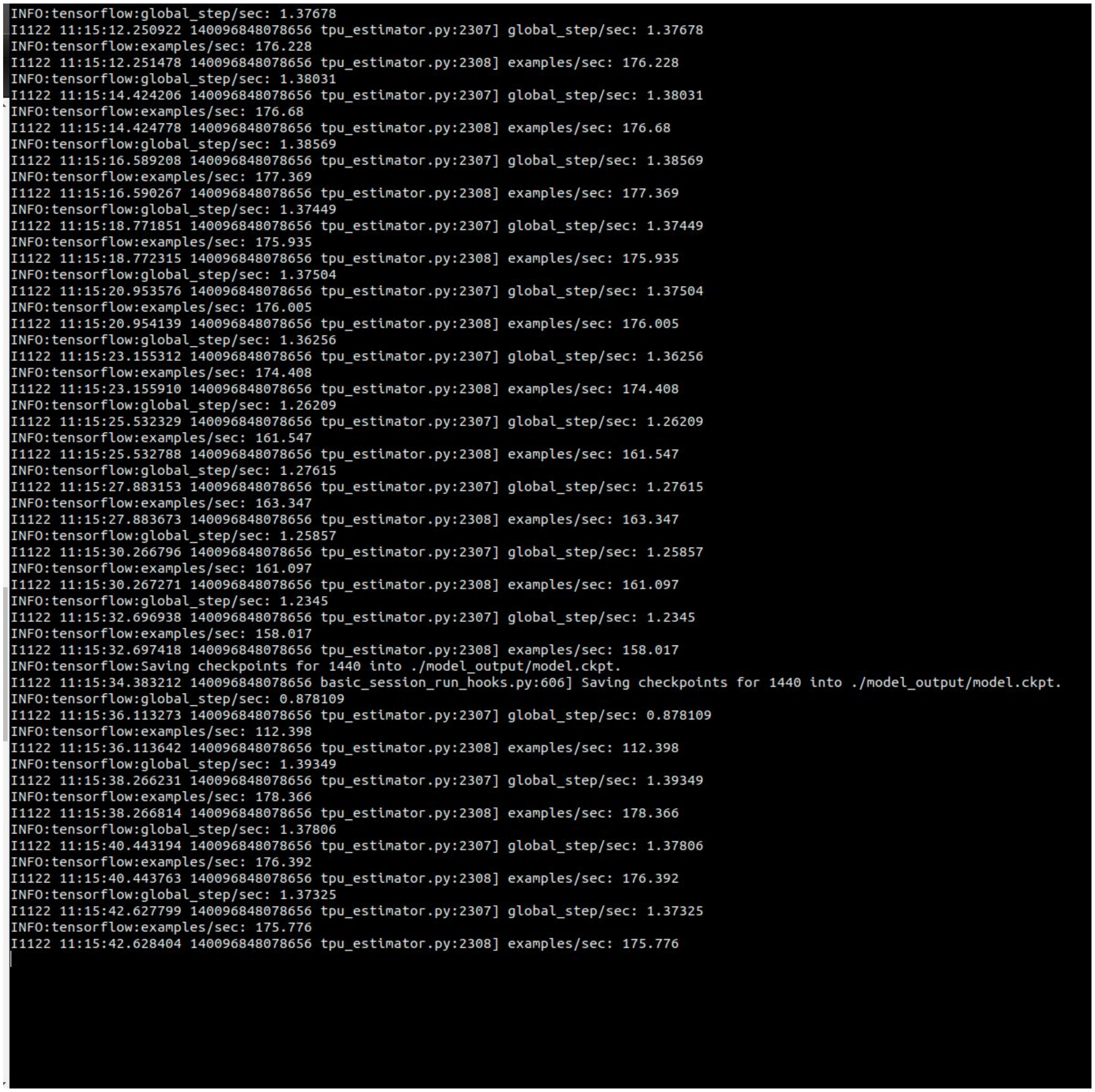
[3] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, & Geoffrey Hinton. (2020). Big Self-Supervised Models are Strong Semi-Supervised Learners.

APPENDIX

Alzheimers Dataset pretraining with OASIS-1 (on SJSU HPC)

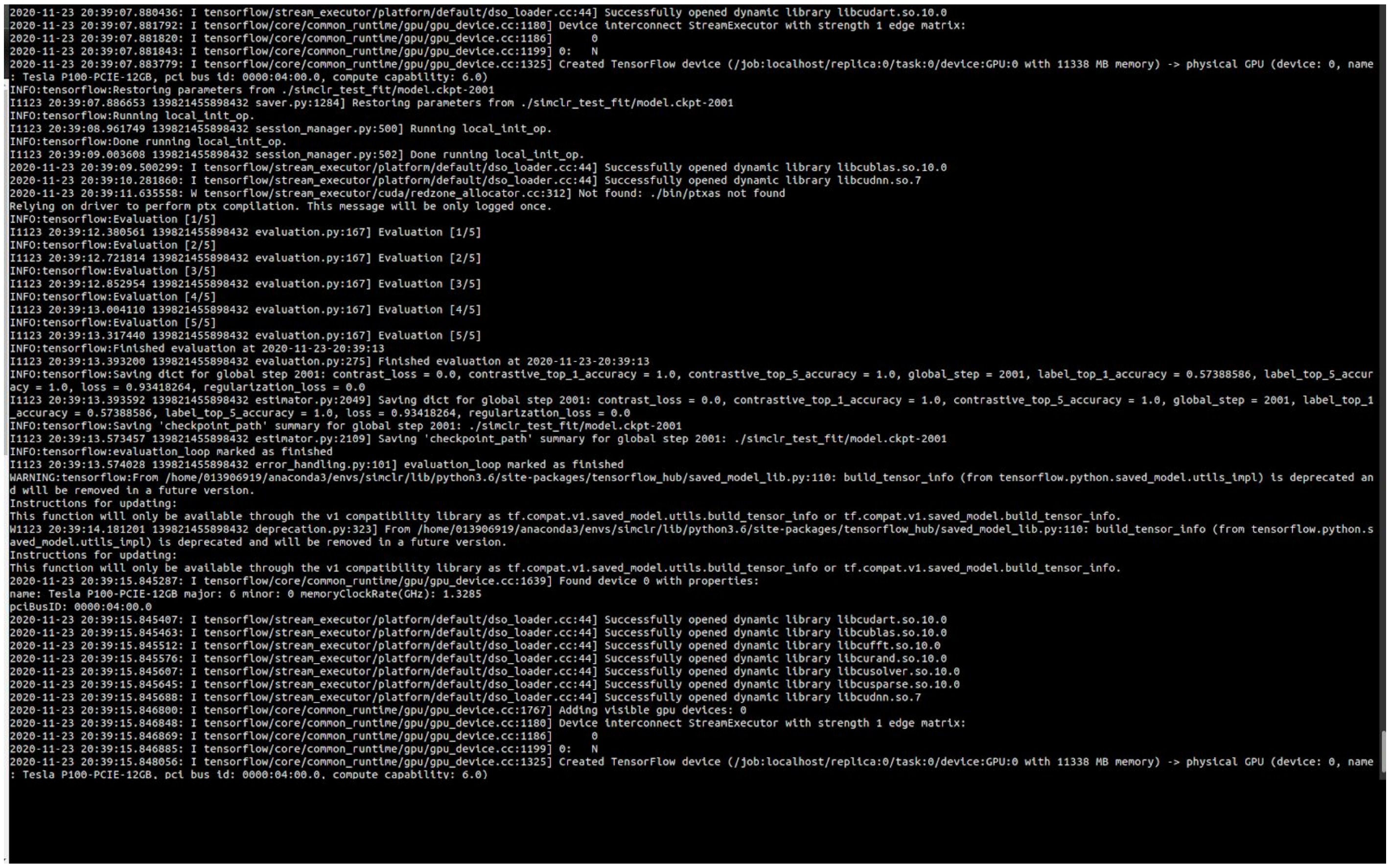


```
INFO:tensorflow:global_step/sec: 1.19382
I1122 10:56:33.684570 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.19382
INFO:tensorflow:examples/sec: 152.809
I1122 10:56:33.685139 140096848078656 tpu_estimator.py:2308] examples/sec: 152.809
INFO:tensorflow:global_step/sec: 1.10122
I1122 10:56:36.408823 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.10122
INFO:tensorflow:examples/sec: 140.956
I1122 10:56:36.409384 140096848078656 tpu_estimator.py:2308] examples/sec: 140.956
INFO:tensorflow:global_step/sec: 1.25033
I1122 10:56:38.808414 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.25033
INFO:tensorflow:examples/sec: 160.043
I1122 10:56:38.809194 140096848078656 tpu_estimator.py:2308] examples/sec: 160.043
INFO:tensorflow:global_step/sec: 1.17248
I1122 10:56:41.366862 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.17248
INFO:tensorflow:examples/sec: 150.077
I1122 10:56:41.367188 140096848078656 tpu_estimator.py:2308] examples/sec: 150.077
INFO:tensorflow:Saving checkpoints for 40 into ./model_output/model.ckpt.
I1122 10:56:43.887359 140096848078656 basic_session_run_hooks.py:606] Saving checkpoints for 40 into ./model_output/model.ckpt.
INFO:tensorflow:global_step/sec: 0.831274
I1122 10:56:44.975708 140096848078656 tpu_estimator.py:2307] global_step/sec: 0.831274
INFO:tensorflow:examples/sec: 106.403
I1122 10:56:44.976078 140096848078656 tpu_estimator.py:2308] examples/sec: 106.403
INFO:tensorflow:global_step/sec: 1.36866
I1122 10:56:47.167726 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.36866
INFO:tensorflow:examples/sec: 175.189
I1122 10:56:47.168219 140096848078656 tpu_estimator.py:2308] examples/sec: 175.189
INFO:tensorflow:global_step/sec: 1.36236
I1122 10:56:49.369779 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.36236
INFO:tensorflow:examples/sec: 174.382
I1122 10:56:49.370135 140096848078656 tpu_estimator.py:2308] examples/sec: 174.382
INFO:tensorflow:global_step/sec: 1.3604
I1122 10:56:51.575188 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.3604
INFO:tensorflow:examples/sec: 174.131
I1122 10:56:51.576020 140096848078656 tpu_estimator.py:2308] examples/sec: 174.131
INFO:tensorflow:global_step/sec: 1.38516
I1122 10:56:53.740886 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.38516
INFO:tensorflow:examples/sec: 177.3
I1122 10:56:53.741498 140096848078656 tpu_estimator.py:2308] examples/sec: 177.3
INFO:tensorflow:global_step/sec: 1.36688
I1122 10:56:55.935627 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.36688
INFO:tensorflow:examples/sec: 174.961
I1122 10:56:55.936217 140096848078656 tpu_estimator.py:2308] examples/sec: 174.961
INFO:tensorflow:global_step/sec: 1.24669
I1122 10:56:58.341992 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.24669
INFO:tensorflow:examples/sec: 159.576
I1122 10:56:58.342489 140096848078656 tpu_estimator.py:2308] examples/sec: 159.576
INFO:tensorflow:global_step/sec: 1.196
I1122 10:57:00.850348 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.196
INFO:tensorflow:examples/sec: 153.088
I1122 10:57:00.850830 140096848078656 tpu_estimator.py:2308] examples/sec: 153.088
INFO:tensorflow:global_step/sec: 1.18603
I1122 10:57:03.379783 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.18603
INFO:tensorflow:examples/sec: 151.812
I1122 10:57:03.380258 140096848078656 tpu_estimator.py:2308] examples/sec: 151.812
INFO:tensorflow:global_step/sec: 1.18647
I1122 10:57:05.908302 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.18647
INFO:tensorflow:examples/sec: 151.868
I1122 10:57:05.908744 140096848078656 tpu_estimator.py:2308] examples/sec: 151.868
```



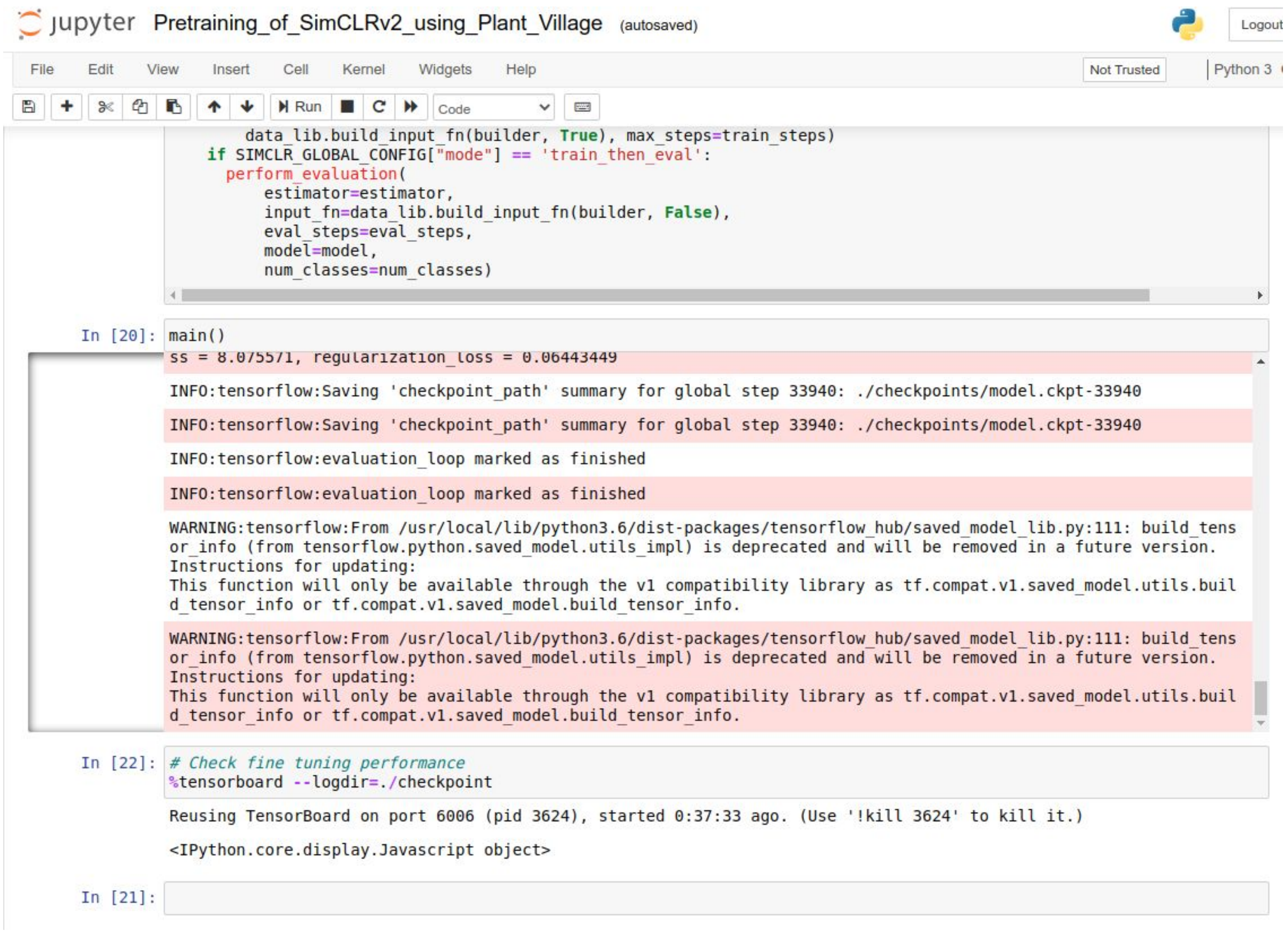
```
INFO:tensorflow:global_step/sec: 1.37678
I1122 11:15:12.250922 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.37678
INFO:tensorflow:examples/sec: 176.228
I1122 11:15:12.251478 140096848078656 tpu_estimator.py:2308] examples/sec: 176.228
INFO:tensorflow:global_step/sec: 1.38031
I1122 11:15:14.424206 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.38031
INFO:tensorflow:examples/sec: 176.68
I1122 11:15:14.424778 140096848078656 tpu_estimator.py:2308] examples/sec: 176.68
INFO:tensorflow:global_step/sec: 1.38569
I1122 11:15:16.589208 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.38569
INFO:tensorflow:examples/sec: 177.369
I1122 11:15:16.590267 140096848078656 tpu_estimator.py:2308] examples/sec: 177.369
INFO:tensorflow:global_step/sec: 1.37449
I1122 11:15:18.771851 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.37449
INFO:tensorflow:examples/sec: 175.935
I1122 11:15:18.772315 140096848078656 tpu_estimator.py:2308] examples/sec: 175.935
INFO:tensorflow:global_step/sec: 1.37504
I1122 11:15:20.953576 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.37504
INFO:tensorflow:examples/sec: 176.005
I1122 11:15:20.954139 140096848078656 tpu_estimator.py:2308] examples/sec: 176.005
INFO:tensorflow:global_step/sec: 1.36256
I1122 11:15:23.155312 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.36256
INFO:tensorflow:examples/sec: 174.408
I1122 11:15:23.155910 140096848078656 tpu_estimator.py:2308] examples/sec: 174.408
INFO:tensorflow:global_step/sec: 1.26209
I1122 11:15:25.532329 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.26209
INFO:tensorflow:examples/sec: 161.547
I1122 11:15:25.532788 140096848078656 tpu_estimator.py:2308] examples/sec: 161.547
INFO:tensorflow:global_step/sec: 1.27615
I1122 11:15:27.883153 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.27615
INFO:tensorflow:examples/sec: 163.347
I1122 11:15:27.883673 140096848078656 tpu_estimator.py:2308] examples/sec: 163.347
INFO:tensorflow:global_step/sec: 1.25857
I1122 11:15:30.266796 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.25857
INFO:tensorflow:examples/sec: 161.097
I1122 11:15:30.267271 140096848078656 tpu_estimator.py:2308] examples/sec: 161.097
INFO:tensorflow:global_step/sec: 1.2345
I1122 11:15:32.696938 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.2345
INFO:tensorflow:examples/sec: 158.017
I1122 11:15:32.697418 140096848078656 tpu_estimator.py:2308] examples/sec: 158.017
INFO:tensorflow:Saving checkpoints for 1440 into ./model_output/model.ckpt.
I1122 11:15:34.383212 140096848078656 basic_session_run_hooks.py:606] Saving checkpoints for 1440 into ./model_output/model.ckpt.
INFO:tensorflow:global_step/sec: 0.878109
I1122 11:15:36.113273 140096848078656 tpu_estimator.py:2307] global_step/sec: 0.878109
INFO:tensorflow:examples/sec: 112.398
I1122 11:15:36.113642 140096848078656 tpu_estimator.py:2308] examples/sec: 112.398
INFO:tensorflow:global_step/sec: 1.39349
I1122 11:15:38.266231 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.39349
INFO:tensorflow:examples/sec: 178.366
I1122 11:15:38.266814 140096848078656 tpu_estimator.py:2308] examples/sec: 178.366
INFO:tensorflow:global_step/sec: 1.37806
I1122 11:15:40.443194 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.37806
INFO:tensorflow:examples/sec: 176.392
I1122 11:15:40.443763 140096848078656 tpu_estimator.py:2308] examples/sec: 176.392
INFO:tensorflow:global_step/sec: 1.37325
I1122 11:15:42.627799 140096848078656 tpu_estimator.py:2307] global_step/sec: 1.37325
INFO:tensorflow:examples/sec: 175.776
I1122 11:15:42.628404 140096848078656 tpu_estimator.py:2308] examples/sec: 175.776
```

Alzheimers Severity dataset fine-tuning



```
2020-11-23 20:39:07.880436: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudart.so.10.0
2020-11-23 20:39:07.881792: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1180] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-11-23 20:39:07.881820: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1186]      0
2020-11-23 20:39:07.881843: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1199] 0:   N
2020-11-23 20:39:07.883779: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1325] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 11338 MB memory) -> physical GPU (device: 0, name
: Tesla P100-PCIE-12GB, pci bus id: 0000:04:00.0, compute capability: 6.0)
INFO:tensorflow:Restoring parameters from ./simclr_test_fit/model.ckpt-2001
I1123 20:39:07.886653 139821455898432 saver.py:1284] Restoring parameters from ./simclr_test_fit/model.ckpt-2001
INFO:tensorflow:Running local_init_op.
I1123 20:39:08.961749 139821455898432 session_manager.py:500] Running local_init_op.
INFO:tensorflow:Done running local_init_op.
I1123 20:39:09.003608 139821455898432 session_manager.py:502] Done running local_init_op.
2020-11-23 20:39:09.500299: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcublas.so.10.0
2020-11-23 20:39:10.281860: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudnn.so.7
2020-11-23 20:39:11.635558: W tensorflow/stream_executor/cuda/redzone_allocator.cc:312] Not found: ./bin/ptxas not found
Relying on driver to perform ptx compilation. This message will be only logged once.
INFO:tensorflow:Evaluation [1/5]
I1123 20:39:12.380561 139821455898432 evaluation.py:167] Evaluation [1/5]
INFO:tensorflow:Evaluation [2/5]
I1123 20:39:12.721814 139821455898432 evaluation.py:167] Evaluation [2/5]
INFO:tensorflow:Evaluation [3/5]
I1123 20:39:12.852954 139821455898432 evaluation.py:167] Evaluation [3/5]
INFO:tensorflow:Evaluation [4/5]
I1123 20:39:13.004110 139821455898432 evaluation.py:167] Evaluation [4/5]
INFO:tensorflow:Evaluation [5/5]
I1123 20:39:13.317440 139821455898432 evaluation.py:167] Evaluation [5/5]
INFO:tensorflow:Finished evaluation at 2020-11-23-20:39:13
I1123 20:39:13.393200 139821455898432 evaluation.py:275] Finished evaluation at 2020-11-23-20:39:13
INFO:tensorflow:Saving dict for global step 2001: contrast_loss = 0.0, contrastive_top_1_accuracy = 1.0, contrastive_top_5_accuracy = 1.0, global_step = 2001, label_top_1_accuracy = 0.57388586, label_top_5_accur
acy = 1.0, loss = 0.93418264, regularization_loss = 0.0
I1123 20:39:13.393592 139821455898432 estimator.py:2049] Saving dict for global step 2001: contrast_loss = 0.0, contrastive_top_1_accuracy = 1.0, contrastive_top_5_accuracy = 1.0, global_step = 2001, label_top_1
_accuracy = 0.57388586, label_top_5_accuracy = 1.0, loss = 0.93418264, regularization_loss = 0.0
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 2001: ./simclr_test_fit/model.ckpt-2001
I1123 20:39:13.573457 139821455898432 estimator.py:2109] Saving 'checkpoint_path' summary for global step 2001: ./simclr_test_fit/model.ckpt-2001
INFO:tensorflow:evaluation_loop marked as finished
I1123 20:39:13.574028 139821455898432 error_handling.py:101] evaluation_loop marked as finished
WARNING:tensorflow:From /home/013906919/anaconda3/envs/simclr/lib/python3.6/site-packages/tensorflow_hub/saved_model_lib.py:110: build_tensor_info (from tensorflow.python.saved_model.utils_impl) is deprecated an
d will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.build_tensor_info or tf.compat.v1.saved_model.build_tensor_info.
W1123 20:39:14.181201 139821455898432 deprecation.py:323] From /home/013906919/anaconda3/envs/simclr/lib/python3.6/site-packages/tensorflow_hub/saved_model_lib.py:110: build_tensor_info (from tensorflow.python.s
aved_model.utils_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.build_tensor_info or tf.compat.v1.saved_model.build_tensor_info.
2020-11-23 20:39:15.845287: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1639] Found device 0 with properties:
name: Tesla P100-PCIE-12GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:04:00.0
2020-11-23 20:39:15.845407: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudart.so.10.0
2020-11-23 20:39:15.845463: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcublas.so.10.0
2020-11-23 20:39:15.845512: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcufft.so.10.0
2020-11-23 20:39:15.845576: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcurand.so.10.0
2020-11-23 20:39:15.845607: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcusolver.so.10.0
2020-11-23 20:39:15.845645: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcusparse.so.10.0
2020-11-23 20:39:15.845688: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudnn.so.7
2020-11-23 20:39:15.846800: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1767] Adding visible gpu devices: 0
2020-11-23 20:39:15.846848: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1180] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-11-23 20:39:15.846869: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1186]      0
2020-11-23 20:39:15.846885: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1199] 0:   N
2020-11-23 20:39:15.848056: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1325] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 11338 MB memory) -> physical GPU (device: 0, name
: Tesla P100-PCIE-12GB, pci bus id: 0000:04:00.0, compute capability: 6.0)
```

Plant Village Dataset pre-training


jupyter Pretraining_of_SimCLRv2_using_Plant_Village (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
        data_lib.build_input_fn(builder, True), max_steps=train_steps)
    if SIMCLR_GLOBAL_CONFIG["mode"] == 'train_then_eval':
      perform_evaluation(
          estimator=estimator,
          input_fn=data_lib.build_input_fn(builder, False),
          eval_steps=eval_steps,
          model=model,
          num_classes=num_classes)
```

```
In [20]: main()
```

```
ss = 8.075571, regularization_loss = 0.06443449

INFO:tensorflow:Saving 'checkpoint_path' summary for global step 33940: ./checkpoints/model.ckpt-33940

INFO:tensorflow:Saving 'checkpoint_path' summary for global step 33940: ./checkpoints/model.ckpt-33940

INFO:tensorflow:evaluation_loop marked as finished

INFO:tensorflow:evaluation_loop marked as finished

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_hub/saved_model_lib.py:111: build_tens
or_info (from tensorflow.python.saved_model.utils_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.buil
d_tensor_info or tf.compat.v1.saved_model.build_tensor_info.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_hub/saved_model_lib.py:111: build_tens
or_info (from tensorflow.python.saved_model.utils_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.buil
d_tensor_info or tf.compat.v1.saved_model.build_tensor_info.
```

```
In [22]: # Check fine tuning performance
         %tensorboard --logdir=./checkpoint
```

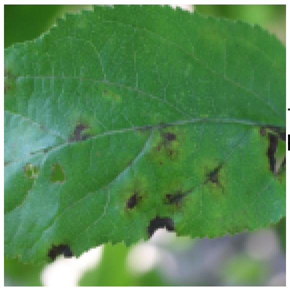
```
Reusing TensorBoard on port 6006 (pid 3624), started 0:37:33 ago. (Use '!kill 3624' to kill it.)

<IPython.core.display.Javascript object>
```

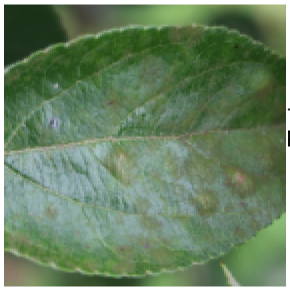
```
In [21]:
```

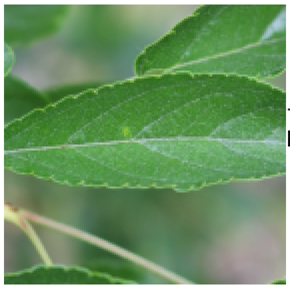
Plant Pathology Dataset Finetuning (from Finuetuning_and_Distillation_of_SimCLRv2_using_Plant_Pathology.ipynb)

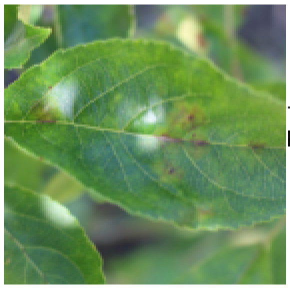
```
[Iter 1] Loss: 1.7947008609771729 Top1 Accuracy: 0.328125
[Iter 2] Loss: 6.353813171386719 Top1 Accuracy: 0.3125
[Iter 3] Loss: 11.957975387573242 Top1 Accuracy: 0.359375
[Iter 4] Loss: 10.061290740966797 Top1 Accuracy: 0.34375
[Iter 5] Loss: 2.5968565940856934 Top1 Accuracy: 0.625
[Iter 6] Loss: 2.488285779953003 Top1 Accuracy: 0.640625
[Iter 7] Loss: 3.117185115814209 Top1 Accuracy: 0.46875
[Iter 8] Loss: 4.863947868347168 Top1 Accuracy: 0.546875
[Iter 9] Loss: 6.003618240356445 Top1 Accuracy: 0.578125
[Iter 10] Loss: 5.8602190017700195 Top1 Accuracy: 0.59375
[Iter 11] Loss: 1.8799350261688232 Top1 Accuracy: 0.671875
[Iter 12] Loss: 6.011013031005859 Top1 Accuracy: 0.34375
[Iter 13] Loss: 1.7753324508666992 Top1 Accuracy: 0.78125
[Iter 14] Loss: 3.5107946395874023 Top1 Accuracy: 0.640625
[Iter 15] Loss: 2.1833629608154297 Top1 Accuracy: 0.765625
[Iter 16] Loss: 2.1224470138549805 Top1 Accuracy: 0.6875
[Iter 17] Loss: 0.8193728923797607 Top1 Accuracy: 0.828125
[Iter 18] Loss: 1.5057377815246582 Top1 Accuracy: 0.796875
[Iter 19] Loss: 1.2158958911895752 Top1 Accuracy: 0.828125
```

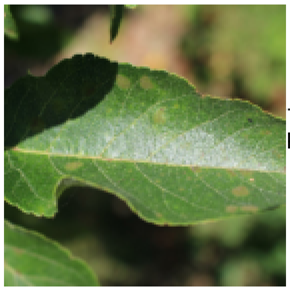
```
[Iter 20] Loss: 1.0019350051879883 Top1 Accuracy: 0.828125
----------Evaluate the model----------
[Iter 1] test_loss: 1.6662030220031738 test_top1_accuracy: 0.78125
[Iter 2] test_loss: 1.8175444602966309 test_top1_accuracy: 0.75
[Iter 3] test_loss: 0.9973149299621582 test_top1_accuracy: 0.828125
[Iter 4] test_loss: 0.7481752634048462 test_top1_accuracy: 0.859375
[Iter 5] test_loss: 0.8586392402648926 test_top1_accuracy: 0.796875
{'block_group1': <tf.Tensor 'model/StatefulPartitionedCall:0' shape=(34, 64, 64, 256) dtype=float32>,
'block_group2': <tf.Tensor 'model/StatefulPartitionedCall:1' shape=(34, 32, 32, 512) dtype=float32>,
'block_group3': <tf.Tensor 'model/StatefulPartitionedCall:2' shape=(34, 16, 16, 1024) dtype=float32>,
'block_group4': <tf.Tensor 'model/StatefulPartitionedCall:3' shape=(34, 8, 8, 2048) dtype=float32>,
'initial_max_pool': <tf.Tensor 'model/StatefulPartitionedCall:6' shape=(34, 64, 64, 64) dtype=float32>,
'final_avg_pool': <tf.Tensor 'model/StatefulPartitionedCall:4' shape=(34, 2048) dtype=float32>, 'logits_sup':
<tf.Tensor 'model/StatefulPartitionedCall:7' shape=(34, 1000) dtype=float32>, 'initial_conv': <tf.Tensor
'model/StatefulPartitionedCall:5' shape=(34, 128, 128, 64) dtype=float32>}
Variables to train: [<tf.Variable 'model/head_supervised_new/kernel:0' shape=(2048, 4) dtype=float32>,
<tf.Variable 'model/head_supervised_new/bias:0' shape=(4,) dtype=float32>]
[Iter 6] test_loss: 0.9046571254730225 test_top1_accuracy: 0.8529411764705882
----------Results----------
                  precision    recall  f1-score   support

         healthy       0.54      0.50      0.52       363
multiple_diseases       0.11      0.07      0.08        59
            rust       0.65      0.66      0.65       429
            scab       0.64      0.69      0.66       429

        accuracy                           0.60      1280
       macro avg       0.48      0.48      0.48      1280
    weighted avg       0.59      0.60      0.59      1280
```


Truth: scab
Pred: multiple_diseases


Truth: scab
Pred: scab


Truth: healthy
Pred: healthy


Truth: scab
Pred: scab


Truth: scab
Pred: healthy