Transfer Learning Tests for Kaggle Fingers Dataset Using AWS Sagemaker

Context

- The goal of the project is to build a model able to count fingers as well as distinguish between left and right hand.

Content

- 21600 images of left and right hands fingers.
- All images are 128 by 128 pixels.
- Training set: 18000 images
- Test set: 3600 images
- Images are centered by the center of mass
- Noise pattern on the background

Labels

- Labels are in 2 last characters of a file name. L/R indicates left/right hand; 0,1,2,3,4,5 indicates number of fingers.

Note

- Images of a left hand were generated by flipping images of right hand.
- Later, the *training images* where divided in a *validation set* of 3600 images giving a training set of 14400 images

[Kaggle Fingers Dataset (https://www.kaggle.com/datasets/koryakinp/fingers)]

PreInstall libraries that are functional with sagemaker debug and profiling

NOTE:

- This function will be deprecated on sagemaker>2.0
- WE WILL NOT TRAIN FOR THE WHOLE DATASET BECAUSE IS NOT NECCESARY AT ALL, THE MODEL PERFORMS WELL WITH 20% OF THE TOTAL EPOCH

In [1]:
```
!pip install smdebug
!pip install --upgrade wandb==0.12.17
!pip install protobuf==3.19.0
!pip install --upgrade bokeh==2.4.3
```

```
Downloading sentry_sdk-1.32.0-py2.py3-none-any.whl (240 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 241.0/241.0 kB 1.2 MB/s eta 0:00:00:00:01
Downloading setproctitle-1.3.3-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (30 kB)
Downloading gitdb-4.0.11-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 517.1 kB/s eta 0:00:0000:01
Using cached smmap-5.0.1-py3-none-any.whl (24 kB)
Building wheels for collected packages: promise, pathtools
  Building wheel for promise (setup.py) ... done
  Created wheel for promise: filename=promise-2.3-py3-none-any.whl size=21484 sha256=99dbb620771c26d64093e7274a830d940cd36dfdc95dac24b322c268db885345
  Stored in directory: /home/ec2-user/.cache/pip/wheels/54/4e/28/3ed0e1c8a752867445bab994d2340724928aa3ab059c57c8db
  Building wheel for pathtools (setup.py) ... done
  Created wheel for pathtools: filename=pathtools-0.1.2-py3-none-any.whl size=8791 sha256=ba7bd371eb7c058f26560e77d7f261cce41b778896f9a333af70665838af914b
  Stored in directory: /home/ec2-user/.cache/pip/wheels/e7/f3/22/152153d6eb222ee7a56ff8617d80ee5207207a8c00a7aab794
Successfully built promise pathtools
Installing collected packages: pathtools, smmap, shortuuid, setproctitle, sentry-sdk, promise, docker-pycreds, gitdb, GitPython, wandb
Successfully installed GitPython-3.1.40 docker-pycreds-0.4.0 gitdb-4.0.11 pathtools-0.1.2 promise-2.3 sentry-sdk-1.32.0 setpro
```

Resets the notebook for the pre-installed libraries to work

In [ ]:
```
from IPython.core.display import HTML
HTML("<script>Jupyter.notebook.kernel.restart()</script>") # reset the kernel
```

Out[2]:

Load Important Libraries for HyperParameter Optimization and Training

In [1]:
```
import sagemaker
import boto3
from sagemaker.pytorch import PyTorch
from io import BytesIO
import zipfile
from sagemaker.tuner import (
    IntegerParameter,
    CategoricalParameter,
    ContinuousParameter,
    HyperparameterTuner,
)
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
```

In [2]:
```
bucket = 'sagemaker-instance-bucket' # bucket where the fingers_sorted.zip data is located
role = sagemaker.get_execution_role() # the default role
sagemaker_session = sagemaker.Session() # the default session
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
```

Dataset

For a better understanding of the dataset you can look at the **Data Exploration** notebook on the root

In [20]:
```
# Download the dataset from this instance bucket
!aws s3 cp s3://sagemaker-instance-bucket/fingers_sorted.zip fingers_sorted.zip
```

```
download: s3://sagemaker-instance-bucket/fingers_sorted.zip to ./fingers_sorted.zip
```

In [21]:
```
# unzip the dataset locally
!unzip fingers_sorted.zip
```

...

In [3]:
```
sagemaker_session = sagemaker.Session() # the default session
bucket = sagemaker_session.default_bucket() # the default bucket where will be located the data
prefix = "fingers_sorted" # local folder where the data is

role = sagemaker.get_execution_role() # sagemaker default role
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
```

Data Upload

We will demonstrate (for general purpose only) we can upload data, unzip it and load the the bucket that will hold the results of training.

In [4]:
```
# upload again to the same bucket (just to demonstrate that i could upload data)
s3_client = boto3.client('s3')
Filename = 'fingers_sorted.zip'
Bucket = 'sagemaker-instance-bucket'
s3_client.upload_file(Filename=Filename, Bucket=Bucket, Key=Filename) # upload the fingers_sorted.zip file to the default bucket
```

In [26]:
```
# upload to the default bucket in fingers_sorted folder
s3_resource = boto3.resource('s3')
zip_obj = s3_resource.Object(bucket_name=Bucket, key=Filename)
buffer = BytesIO(zip_obj.get()["Body"].read())

# buffer each data to the memory and upload to the default bucket in a folder
z = zipfile.ZipFile(buffer)
for filename in z.namelist():
    file_info = z.getinfo(filename)
    s3_resource.meta.client.upload_fileobj(
        z.open(filename),
        Bucket=bucket,
        Key=f'{filename}'
    )
```

In [5]:
```
# inputs used to train the finger, will be on the default bucket and the fingers_sorted folder
inputs = f"s3://{bucket}/fingers_sorted"
inputs
```

Out[5]:
```
's3://sagemaker-us-east-1-254050731868/fingers_sorted'
```

Hyperparameter Tuning

This is the part where we will finetune a pretrained model with hyperparameter tuning. Our Neural Network used here is the VGG16 model. We will be tunning 3 hyperparameters:

- **learning-rate**: This parameter is crucial for the performance of the model, usually a good value for hyperparameter tunning will start between 0.001 (1e-3) or below (sometimes above like this case).
- **batch-size**: The lower the batch size the grained the model will learn but will delay more.
- **epochs**: One epoch corresponds to the whole dataset run. This will lead the neural network to learn better the relationships.

In [6]:
```
# First we will define the metrics, in this case we will monitor the validation loss
objective_metric_name = "valid_loss"
objective_type = "Minimize"
metric_definitions = [{"Name": objective_metric_name, "Regex": "valid loss: ([0-9\\.]+)"}]
```

In [10]:
```
# Those are the hyperparameters to vary, ensuring correct training
hyperparameter_ranges = {
    "learning-rate": ContinuousParameter(0.001, 0.01),
    "batch-size": CategoricalParameter([32, 64, 128]),
    "epochs": IntegerParameter(2, 5)
}
```

Training (Fine Tunning with HyperParameter Optimization)

Estimators are the machine learning / deep learning models that we will use. In this case we will use PyTorch.

In [11]:
```
# The estimator ta
estimator = PyTorch(
    source_dir="code", # where the codes of the estimator reside
    entry_point="hpo.py", # the file that has the algorithm
    framework_version="1.8", # framework used of the container
    py_version='py36', # which python version is used
    instance_count=1, # number of machines for distributed training
    instance_type="ml.m5.large", # type of model to train
    role=role, # aws role
    sagemaker_session=sagemaker_session, # aws session
    base_job_name="est-fingers", # base job name of the estimator
    hyperparameters = {'epochs': 1, 'batch-size':32, 'learning-rate': 0.01}, # hyperparamers only act if the estimator is trained
#    output_path=f's3://{bucket}', # the bucket that will have the results/model
#    max_run=300, # time in seconds this machine will execute
#    dependencies=['code/requirements.txt'] # if you want to ensure some dependencies installed
)

tuner = HyperparameterTuner(estimator, # our previous estimator
                            objective_metric_name, # the metric name (validation loss)
                            hyperparameter_ranges,  # values that change
                            metric_definitions, # the definition
                            max_jobs=4, # number of maximun jobs
                            max_parallel_jobs=2,  # jobs at the same time
                            objective_type=objective_type, # objective name
                             base_tuning_job_name='tun-fingers', # base name
#                              max_runtime_in_seconds=1800, # total running time of all estimators in the HPO task
                           )
```

In [12]:
```
tuner.fit({"training": inputs}, wait=True) # tune the model and wait
```

```
No finished training job found associated with this estimator. Please make sure this estimator is only used for building workflo
w config

Using provided s3_resource
.....................................................................................................................
.....................................................................................................................
.....................................................................................................................
.....................................................................................................................
.....................................................................................................................
.....................................................................................................................
.....................................................................................................................
.....................................................................................!
```

In [13]:
```
best_estimator = tuner.best_estimator() # get the results
best_estimator
```

```
2023-10-26 03:50:52 Starting - Preparing the instances for training
2023-10-26 03:50:52 Downloading - Downloading input data
2023-10-26 03:50:52 Training - Training image download completed. Training in progress.
2023-10-26 03:50:52 Uploading - Uploading generated training model
2023-10-26 03:50:52 Completed - Resource reused by training job: tun-fingers-231026-0306-004-dd9b1726
```

Out[13]:
```
<sagemaker.pytorch.estimator.PyTorch at 0x7f92894953c0>
```

In [14]:
```
best_estimator.hyperparameters() # see the hyperparameters
```

Out[14]:
```
{'_tuning_objective_metric': '"valid_loss"',
 'batch-size': '"32"',
 'epochs': '4',
 'learning-rate': '0.0031425732251105973',
 'sagemaker_container_log_level': '20',
 'sagemaker_estimator_class_name': '"PyTorch"',
 'sagemaker_estimator_module': '"sagemaker.pytorch.estimator"',
 'sagemaker_job_name': '"est-fingers-2023-10-26-03-06-40-237"',
 'sagemaker_program': '"hpo.py"',
 'sagemaker_region': '"us-east-1"',
 'sagemaker_submit_directory': '"s3://sagemaker-us-east-1-254050731868/est-fingers-2023-10-26-03-06-40-237/source/sourcedir.tar.
gz"'}
```

In [15]:
```
# reload the best hyperparameters
hyperparameters = {
    'batch-size': int(best_estimator.hyperparameters()['batch-size'][1:-1]),
    'learning-rate': float(best_estimator.hyperparameters()['learning-rate'][1:-1]),
    'epochs': int(best_estimator.hyperparameters()['epochs'])
}
hyperparameters
```

Out[15]:
```
{'batch-size': 32, 'learning-rate': 0.003142573225110597, 'epochs': 4}
```

Model Profiling and Debugging

Here we will be using the best hyperparameters, create and finetuned new model

In [16]:
```
# setting debug and profiling hooks
from sagemaker.debugger import Rule, ProfilerRule, rule_configs

rules = [
    Rule.sagemaker(rule_configs.loss_not_decreasing()),
    ProfilerRule.sagemaker(rule_configs.LowGPUUtilization()),
    ProfilerRule.sagemaker(rule_configs.ProfilerReport()),
    Rule.sagemaker(rule_configs.vanishing_gradient()),
    Rule.sagemaker(rule_configs.overfit()),
    Rule.sagemaker(rule_configs.overtraining()),
    Rule.sagemaker(rule_configs.poor_weight_initialization()),
]
```

In [17]:
```
# get the profile configuration
from sagemaker.debugger import ProfilerConfig, FrameworkProfile

profiler_config=ProfilerConfig(
    framework_profile_params=FrameworkProfile()
)
```

```
Framework profiling will be deprecated from tensorflow 2.12 and pytorch 2.0 in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html (https://sagemaker.readthedocs.io/en/stable/v2.html) for details.
```

In [18]:
```
from sagemaker.debugger import DebuggerHookConfig, ProfilerConfig, FrameworkProfile, CollectionConfig

profiler_config = ProfilerConfig(
    system_monitor_interval_millis=500, framework_profile_params=FrameworkProfile(num_steps=10)
)

debugger_config = DebuggerHookConfig(
    hook_parameters={"train.save_interval": "5", "eval.save_interval": "2"},
)
```

```
Framework profiling will be deprecated from tensorflow 2.12 and pytorch 2.0 in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html (https://sagemaker.readthedocs.io/en/stable/v2.html) for details.
```

In [20]:
```
# Create an estimator with debug
estimator_with_debug = PyTorch(
    source_dir="code",
    entry_point="train_model.py",
    framework_version="1.8",
    py_version='py36',
    instance_count=1,
    instance_type="ml.m5.large",
    role=role,
    sagemaker_session=sagemaker_session,
    base_job_name="sm-fingers",
    hyperparameters=hyperparameters,
    profiler_config=profiler_config,
    debugger_hook_config=debugger_config,
    rules=rules,
#    output_path=f's3://{bucket}',
#    max_run=300,
#    dependencies=['code/requirements.txt']
)
```

In [21]:
```
estimator_with_debug.fit({'training': inputs}, wait=True) # train
```

```
92.2MB/s]#015 50%|████▉     | 264M/528M [00:03<00:02, 92.8MB/s]#015 52%|█████     | 273M/528M [00:03<00:02, 93.6MB/s]#015 5
3%|█████▍    | 282M/528M [00:03<00:02, 94.1MB/s]#015 55%|█████▌    | 291M/528M [00:03<00:02, 94.8MB/s]#015 57%|█████▋    | 3
00M/528M [00:03<00:02, 95.0MB/s]#015 59%|█████▊    | 310M/528M [00:03<00:02, 95.3MB/s]#015 60%|█████▉    | 319M/528M [00:03<0
0:02, 95.1MB/s]#015 62%|██████    | 328M/528M [00:03<00:02, 93.3MB/s]#015 64%|██████▏   | 337M/528M [00:03<00:02, 94.0MB/s]#
015 66%|██████▍   | 346M/528M [00:03<00:02, 94.1MB/s]#015 67%|██████▌   | 355M/528M [00:04<00:01, 93.5MB/s]#015 69%|██████▋
| 364M/528M [00:04<00:01, 93.4MB/s]#015 71%|██████▉   | 373M/528M [00:04<00:01, 92.8MB/s]#015 72%|███████   | 381M/528M [00:0
4<00:01, 90.7MB/s]#015 74%|███████▏  | 390M/528M [00:04<00:01, 91.6MB/s]#015 76%|███████▍  | 399M/528M [00:04<00:01, 92.3MB/
s]#015 77%|███████▌  | 408M/528M [00:04<00:01, 92.5MB/s]#015 79%|███████▋  | 417M/528M [00:04<00:01, 90.8MB/s]#015 81%|████
██  | 426M/528M [00:04<00:01, 91.0MB/s]#015 82%|████████  | 435M/528M [00:04<00:01, 91.6MB/s]#015 84%|████████▏ | 444M/528M
[00:05<00:00, 91.7MB/s]#015 86%|████████▍ | 452M/528M [00:05<00:00, 91.8MB/s]#015 87%|████████▌ | 461M/528M [00:05<00:00, 91.
6MB/s]#015 89%|████████▋ | 470M/528M [00:05<00:00, 91.9MB/s]#015 91%|████████▊ | 479M/528M [00:05<00:00, 91.8MB/s]#015 92%|██
███▉  | 487M/528M [00:05<00:00, 91.4MB/s]#015 94%|█████████ | 496M/528M [00:05<00:00, 91.5MB/s]#015 96%|█████████▎| 505M/5
28M [00:05<00:00, 90.0MB/s]#015 97%|█████████▍| 514M/528M [00:05<00:00, 90.4MB/s]#015 99%|█████████▋| 522M/528M [00:05<00:0
0, 91.1MB/s]#015100%|██████████| 528M/528M [00:06<00:00, 91.4MB/s]
2023-10-26 05:14:07,280 sagemaker-training-toolkit INFO     Reporting training SUCCESS


2023-10-26 05:14:44 Completed - Training job completed
LowGPUUtilization: NoIssuesFound
```

In [22]:
```
# Look for the training job name of the estimator/debugger
training_job_name = estimator_with_debug.latest_training_job.name
print(f"Training jobname: {training_job_name}")
print(f"Region: {sagemaker_session.boto_region_name}")
```

```
Training jobname: sm-fingers-2023-10-26-04-34-05-974
Region: us-east-1
```

In [23]:
```
# see the results of the trial
from smdebug.trials import create_trial
from smdebug.core.modes import ModeKeys

trial = create_trial(estimator_with_debug.latest_job_debugger_artifacts_path())
```

```
[2023-10-26 05:15:44.656 ip-172-16-156-252.ec2.internal:14651 INFO utils.py:28] RULE_JOB_STOP_SIGNAL_FILENAME: None
[2023-10-26 05:15:44.668 ip-172-16-156-252.ec2.internal:14651 INFO s3_trial.py:42] Loading trial debug-output at path s3://sagem
aker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/debug-output
```

In [24]:
```
# Bring the tensor names
print(trial.tensor_names())
```

```
[2023-10-26 05:15:48.172 ip-172-16-156-252.ec2.internal:14651 INFO trial.py:197] Training has ended, will refresh one final time
in 1 sec.
[2023-10-26 05:15:49.190 ip-172-16-156-252.ec2.internal:14651 INFO trial.py:210] Loaded all steps
['CrossEntropyLoss_output_0', 'gradient/Net_model.classifier.0.bias', 'gradient/Net_model.classifier.0.weight', 'gradient/Net_mo
del.classifier.3.bias', 'gradient/Net_model.classifier.3.weight', 'gradient/Net_model.classifier.5.bias', 'gradient/Net_model.cl
assifier.5.weight']
```

In [25]:
```
# TODO: numbers of datapoints processed of those tensors (train and eval)
print(len(trial.tensor("CrossEntropyLoss_output_0").steps(mode=ModeKeys.TRAIN)))
print(len(trial.tensor("CrossEntropyLoss_output_0").steps(mode=ModeKeys.EVAL)))
```

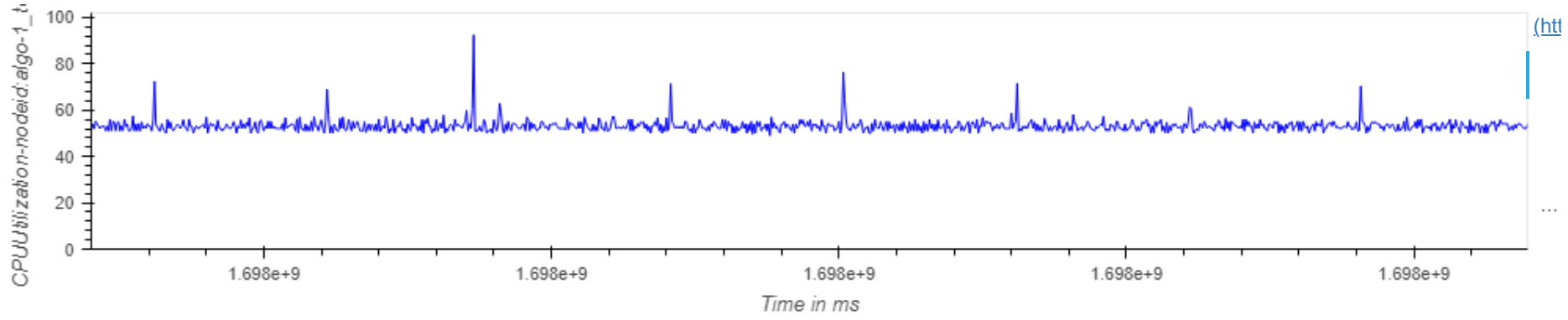
```
1
1
```

In [26]:
```
# check the profiler
from smdebug.profiler.analysis.notebook_utils.training_job import TrainingJob

tj = TrainingJob(training_job_name, region=sagemaker_session.boto_region_name)
tj.wait_for_sys_profiling_data_to_be_available()
```

```
ProfilerConfig:{'S3OutputPath': 's3://sagemaker-us-east-1-254050731868/', 'ProfilingIntervalInMilliseconds': 500, 'ProfilingPara
meters': {'DataloaderProfilingConfig': '{"StartStep": 0, "NumSteps": 10, "MetricsRegex": ".*", }', 'DetailedProfilingConfig':
'{"StartStep": 0, "NumSteps": 10, }', 'FileOpenFailThreshold': '50', 'HorovodProfilingConfig': '{"StartStep": 0, "NumSteps": 10,
}', 'LocalPath': '/opt/ml/output/profiler', 'PythonProfilingConfig': '{"StartStep": 0, "NumSteps": 10, "ProfilerName": "cprofil
e", "cProfileTimer": "total_time", }', 'RotateFileCloseIntervalInSeconds': '60', 'RotateMaxFileSizeInBytes': '10485760', 'SMData
ParallelProfilingConfig': '{"StartStep": 0, "NumSteps": 10, }'}, 'DisableProfiler': False}
s3 path:s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/profiler-output


Profiler data from system is available
```

In [27]:
```
# load analysis
from smdebug.profiler.analysis.notebook_utils.timeline_charts import TimelineCharts

system_metrics_reader = tj.get_systems_metrics_reader()
system_metrics_reader.refresh_event_file_list()

view_timeline_charts = TimelineCharts(
    system_metrics_reader,
    framework_metrics_reader=None,
    select_dimensions=["CPU", "GPU"],
    select_events=["total"],
)
```

```
[2023-10-26 05:15:58.805 ip-172-16-156-252.ec2.internal:14651 INFO metrics_reader_base.py:134] Getting 40 event files
select events:['total']
select dimensions:['CPU', 'GPU']
filtered_events:{'total'}
filtered_dimensions:{'CPUUtilization-nodeid:algo-1'}
```



In [55]:
```
# print where will be the profiler output
rule_output_path = estimator_with_debug.output_path + estimator_with_debug.latest_training_job.job_name + "/rule-output"
print(f"You will find the profiler report in {rule_output_path}")
```

```
You will find the profiler report in s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output
```

In [56]:
```
!aws s3 ls {rule_output_path} --recursive
```

```
2023-10-26 05:14:28     386024 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-report.html
2023-10-26 05:14:28     236043 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-report.ipynb
2023-10-26 05:14:23        192 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/BatchSize.json
2023-10-26 05:14:23        200 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/CPUBottleneck.json
2023-10-26 05:14:23       1934 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/Dataloader.json
2023-10-26 05:14:23        127 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/GPUMemoryIncrease.json
2023-10-26 05:14:23        199 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/IOBottleneck.json
2023-10-26 05:14:23        119 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/LoadBalancing.json
2023-10-26 05:14:23        151 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/LowGPUUtilization.json
2023-10-26 05:14:23        231 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/MaxInitializationTime.json
2023-10-26 05:14:23        879 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/OverallFrameworkMetrics.json
2023-10-26 05:14:23        473 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/OverallSystemUsage.json
2023-10-26 05:14:23       2199 sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/StepOutlier.json
```

In [57]:
```
!aws s3 cp {rule_output_path} ./ --recursive
```

```
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-report.html to ProfilerReport/profiler-output/profiler-report.html
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/BatchSize.json to ProfilerReport/profiler-output/profiler-reports/BatchSize.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/LoadBalancing.json to ProfilerReport/profiler-output/profiler-reports/LoadBalancing.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/CPUBottleneck.json to ProfilerReport/profiler-output/profiler-reports/CPUBottleneck.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/LowGPUUtilization.json to ProfilerReport/profiler-output/profiler-reports/LowGPUUtilization.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/MaxInitializationTime.json to ProfilerReport/profiler-output/profiler-reports/MaxInitializationTime.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/GPUMemoryIncrease.json to ProfilerReport/profiler-output/profiler-reports/GPUMemoryIncrease.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/Dataloader.json to ProfilerReport/profiler-output/profiler-reports/Dataloader.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/OverallFrameworkMetrics.json to ProfilerReport/profiler-output/profiler-reports/OverallFrameworkMetrics.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/OverallSystemUsage.json to ProfilerReport/profiler-output/profiler-reports/OverallSystemUsage.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/IOBottleneck.json to ProfilerReport/profiler-output/profiler-reports/IOBottleneck.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-reports/StepOutlier.json to ProfilerReport/profiler-output/profiler-reports/StepOutlier.json
download: s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/rule-output/ProfilerReport/profiler-output/profiler-report.ipynb to ProfilerReport/profiler-output/profiler-report.ipynb
```

In [58]:
```
# get the autogenerated folder name of profiler report
profiler_report_name = [
    rule["RuleConfigurationName"]
    for rule in estimator_with_debug.latest_training_job.rule_job_summary()
    if "Profiler" in rule["RuleConfigurationName"]
][0]
```

In [60]:
```
profiler_report_name
```

Out[60]:
```
'ProfilerReport'
```

In [62]:
```
# display the profile report
import IPython

IPython.display.HTML(filename=profiler_report_name + "/profiler-output/profiler-report.html")
```

During your training job, the StepOutlier rule was the most frequently triggered. It processed 603 datapoints and was
l times.

	Description	Recommendation	Number of times rule triggered	Number of datapoints	Rule parameters
tepOutlier	Detects outliers in step duration. The step duration for forward and backward pass should be roughly the same throughout the training. If there are significant outliers, it may indicate a system stall or bottleneck issues.	Check if there are any bottlenecks (CPU, I/O) correlated to the step outliers.	14	603	threshold:3 mode:None n_outliers:10 stddev:3
lataloader	Checks how many data loaders are running in parallel and whether the total number is equal the number of available CPU cores. The rule triggers if number is much smaller or larger than the number of available cores. If too small, it might lead to low GPU utilization. If too large, it might impact other compute intensive	Change the number of data loader processes.	0	10	min_threshold:70 max_threshold:200

There are some anomalous (warnings) behaviours that could be resolved if take more care (bottleneck). Probaly increasing the size of dataloaders we will have a lower training time.

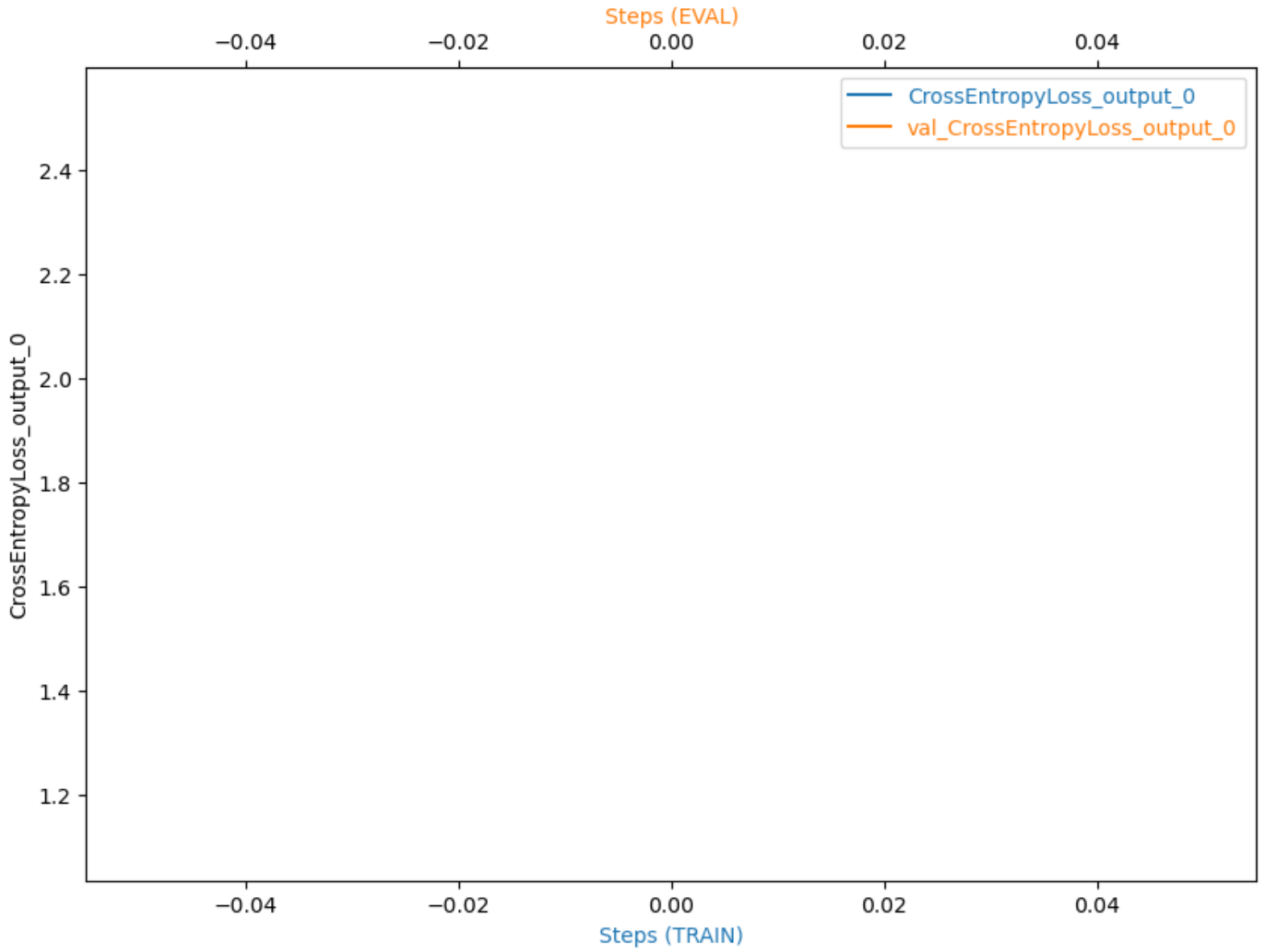
In [135]:
```
# see if we can plot the results
def get_data(trial, tname, mode):
    tensor = trial.tensor(tname)
    steps = tensor.steps(mode=mode)
    vals = []
    for s in steps:
        vals.append(tensor.value(s, mode=mode))
    return steps, vals

print(get_data(trial, "CrossEntropyLoss_output_0", mode=ModeKeys.TRAIN))
print(get_data(trial, "CrossEntropyLoss_output_0", mode=ModeKeys.EVAL))
```

```
([0], [array(2.5270367, dtype=float32)])
([0], [array(1.105053, dtype=float32)])
```

In [64]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import host_subplot


def plot_tensor(trial, tensor_name):

    steps_train, vals_train = get_data(trial, tensor_name, mode=ModeKeys.TRAIN)
    steps_eval, vals_eval = get_data(trial, tensor_name, mode=ModeKeys.EVAL)

    fig = plt.figure(figsize=(10, 7))
    host = host_subplot(111)

    par = host.twiny()

    host.set_xlabel("Steps (TRAIN)")
    par.set_xlabel("Steps (EVAL)")
    host.set_ylabel(tensor_name)

    (p1,) = host.plot(steps_train, vals_train, label=tensor_name)
    (p2,) = par.plot(steps_eval, vals_eval, label="val_" + tensor_name)
    leg = plt.legend()

    host.xaxis.get_label().set_color(p1.get_color())
    leg.texts[0].set_color(p1.get_color())

    par.xaxis.get_label().set_color(p2.get_color())
    leg.texts[1].set_color(p2.get_color())

    plt.ylabel(tensor_name)

    plt.show()
```

```
WARNING:matplotlib.font_manager:Matplotlib is building the font cache; this may take a moment.
INFO:matplotlib.font_manager:generated new fontManager
```

In [65]:
```
# because is one point we will not see all the loss and accuracy to flow on the plot
plot_tensor(trial, "CrossEntropyLoss_output_0")
```



Model Deploying

We will now deploy our model using the same train_model.py file, but we could also load the model using the *PyTorchModel* class if you run the training previously and want to load the data.

In [170]:
```
%%time
predictor = estimator_with_debug.deploy(initial_instance_count=1, instance_type="ml.t2.medium")
```

```
INFO:sagemaker:Repacking model artifact (s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/output/model.tar.gz), script artifact (s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-04-34-05-974/source/sourcedir.tar.gz), and dependencies ([]) into single tar.gz file located at s3://sagemaker-us-east-1-254050731868/sm-fingers-2023-10-26-08-43-50-271/model.tar.gz. This may take some time depending on model size...
INFO:sagemaker:Creating model with name: sm-fingers-2023-10-26-08-43-50-271
INFO:sagemaker:Creating endpoint-config with name sm-fingers-2023-10-26-08-43-50-271
INFO:sagemaker:Creating endpoint with name sm-fingers-2023-10-26-08-43-50-271

-------!CPU times: user 8.53 s, sys: 1.13 s, total: 9.67 s
Wall time: 4min 12s
```

In [196]:
```
from PIL import Image
import os
import numpy as np
from io import BytesIO

# selectin a random test image
test_folder = os.path.join('fingers_sorted', 'test')
random_folder = np.random.choice(['0L', '0R', '1L', '1R', '2L', '2R', '3L', '3R', '4L', '4R', '5L', '5R'])
random_test_folder = os.path.join(test_folder, random_folder)
random_test_image = np.random.choice(os.listdir(random_test_folder))

# load the image (payload)
with open(os.path.join(random_test_folder, random_test_image), "rb") as f:
    image_data = f.read()
```

In [197]:
```
output_object = predictor.predict(data=image_data, initial_args={'ContentType':'application/x-image', 'Accept':'application/json'})
output_object
```

Out[197]:
```
array({'classes': [['2L', '2R', '3L', '1L', '3R', '4L', '1R', '5L', '5R', '4R', '0L', '0R']], 'predictions': [[5.466394901275635, 4.69625186920166, 3.871279716491699, 1.1568315029144287, 1.1320208311080933, 0.7645937204360962, 0.4788486659526825, -0.5899500846862793, -1.4304615259170532, -1.6674001216888428, -2.602926015853882, -2.7487614154815674]]},
      dtype=object)
```

In [199]:
```
output_object.ravel()[0]
```

Out[199]:
```
{'classes': [['2L',
   '2R',
   '3L',
   '1L',
   '3R',
   '4L',
   '1R',
   '5L',
   '5R',
   '4R',
   '0L',
   '0R']],
 'predictions': [[5.466394901275635,
   4.69625186920166,
   3.871279716491699,
   1.1568315029144287,
   1.1320208311080933,
   0.7645937204360962,
   0.4788486659526825,
   -0.5899500846862793,
   -1.4304615259170532,
   -1.6674001216888428,
   -2.602926015853882,
   -2.7487614154815674]]}
```

In [198]:
```
# plot to see if we have the same result of the labeled dataset
plt.imshow(Image.open(BytesIO(image_data)), cmap='gray');
gt = random_test_image.split('.')[0][-2:]
label = output_object.ravel()[0]['classes'][0][0]
plt.title(f'Ground Thruth: {gt}\nPrediction: {label}')
plt.axis(False)
plt.show()
```

In [169]:
```
predictor.delete_endpoint()
```

```
INFO:sagemaker:Deleting endpoint configuration with name: sm-fingers-2023-10-26-08-37-25-910
INFO:sagemaker:Deleting endpoint with name: sm-fingers-2023-10-26-08-37-25-910
```