Amrit Parimi

ap4142

Problem 1

In [ ]:
```
!git clone https://github.com/qfgaohao/pytorch-ssd
```

```
Cloning into 'pytorch-ssd'...
remote: Enumerating objects: 812, done.
remote: Total 812 (delta 0), reused 0 (delta 0), pack-reused 812
Receiving objects: 100% (812/812), 1.05 MiB | 25.60 MiB/s, done.
Resolving deltas: 100% (544/544), done.
```

In [ ]:
```
# 1.1
!wget -P data http://pjreddie.com/media/files/VOCtrainval_06-Nov-2007.tar
!wget -P data http://pjreddie.com/media/files/VOCtest_06-Nov-2007.tar

!tar -xvf "data/VOCtrainval_06-Nov-2007.tar" -C "data"
!tar -xvf "data/VOCtest_06-Nov-2007.tar" -C "data"
```

```
--2021-12-02 08:03:30--  http://pjreddie.com/media/files/VOCtrainval_06-Nov-2007.tar (http://pjreddie.com/media/files/VOCtrainval_06-Nov-2007.tar)
Resolving pjreddie.com (pjreddie.com)... 128.208.4.108
Connecting to pjreddie.com (pjreddie.com)|128.208.4.108|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://pjreddie.com/media/files/VOCtrainval_06-Nov-2007.tar (https://pjreddie.com/media/files/VOCtrainval_06-Nov-2007.tar) [following]
--2021-12-02 08:03:30--  https://pjreddie.com/media/files/VOCtrainval_06-Nov-2007.tar (https://pjreddie.com/media/files/VOCtrainval_06-Nov-2007.tar)
Connecting to pjreddie.com (pjreddie.com)|128.208.4.108|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 460032000 (439M) [application/octet-stream]
Saving to: ‘data/VOCtrainval_06-Nov-2007.tar’

VOCtrainval_06-Nov- 100%[===================>] 438.72M  28.8MB/s    in 12s

2021-12-02 08:03:43 (35.2 MB/s) - ‘data/VOCtrainval_06-Nov-2007.tar’ saved [460032000/460032000]

URL transformed to HTTPS due to an HSTS policy
```

In [ ]:

```
!wget -P models https://storage.googleapis.com/models-hao/mobilenet-v1-ssd-m
!wget -P models https://storage.googleapis.com/models-hao/voc-model-labels.t
```

```
--2021-12-02 08:04:03--  https://storage.googleapis.com/models-hao/mobilenet-v1
-ssd-mp-0_675.pth (https://storage.googleapis.com/models-hao/mobilenet-v1-ssd-m
p-0_675.pth)
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.135.128, 7
4.125.195.128, 74.125.142.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.135.128|:4
43... connected.
HTTP request sent, awaiting response... 200 OK
Length: 37995286 (36M) [application/octet-stream]
Saving to: ‘models/mobilenet-v1-ssd-mp-0_675.pth’

mobilenet-v1-ssd-mp 100%[===================>]  36.23M  16.9MB/s    in 2.1s

2021-12-02 08:04:06 (16.9 MB/s) - ‘models/mobilenet-v1-ssd-mp-0_675.pth’ saved
 [37995286/37995286]

--2021-12-02 08:04:08--  https://storage.googleapis.com/models-hao/voc-model-la
bels.txt (https://storage.googleapis.com/models-hao/voc-model-labels.txt)
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.135.128, 7
4.125.195.128, 74.125.142.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.135.128|:4
43... connected.
HTTP request sent, awaiting response... 200 OK
Length: 145 [text/plain]
Saving to: ‘models/voc-model-labels.txt’

voc-model-labels.tx 100%[===================>]     145  --.-KB/s    in 0s

2021-12-02 08:04:08 (210 MB/s) - ‘models/voc-model-labels.txt’ saved [145/145]
```

In [ ]:
```
!python pytorch-ssd/eval_ssd.py --net mb1-ssd \
                                --trained_model models/mobilen
                                --dataset_type voc \
                                --dataset /content/data/VOCdev
                                --label_file /content/models/v
```

```
boat: 0.5612220055063379
bottle: 0.34852207089954895
bus: 0.7677814849265677
car: 0.7280986468467315
cat: 0.8369208203985581
chair: 0.5169138632991064
cow: 0.6238697603075337
diningtable: 0.7062172972736019
dog: 0.7872868219961326
horse: 0.819446325939355
motorbike: 0.7918539457195842
person: 0.7023619139901851
pottedplant: 0.39852951468542563
sheep: 0.6066678298227772
sofa: 0.7573083661544429
train: 0.8262441264750008
tvmonitor: 0.6461898726506375

Average Precision Across All Classes:0.6759028864961053
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:

```
# 1.2
# !wget -P models https://storage.googleapis.com/models-hao/gun_model_2.21.p
!wget -P models https://storage.googleapis.com/models-hao/open-images-model-
```

```
--2021-12-02 09:38:23--  https://storage.googleapis.com/models-hao/gun_model_2.
21.pth (https://storage.googleapis.com/models-hao/gun_model_2.21.pth)
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.135.128, 7
4.125.195.128, 74.125.142.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.135.128|:4
43... connected.
HTTP request sent, awaiting response... 200 OK
Length: 27044080 (26M) [application/octet-stream]
Saving to: ‘models/gun_model_2.21.pth.1’

gun_model_2.21.pth. 100%[===================>]  25.79M   170MB/s    in 0.2s

2021-12-02 09:38:24 (170 MB/s) - ‘models/gun_model_2.21.pth.1’ saved [27044080/
27044080]

--2021-12-02 09:38:24--  https://storage.googleapis.com/models-hao/open-images-
model-labels.txt (https://storage.googleapis.com/models-hao/open-images-model-l
abels.txt)
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.135.128, 7
4.125.195.128, 74.125.142.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.135.128|:4
43... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26 [text/plain]
Saving to: ‘models/open-images-model-labels.txt.1’

open-images-model-l 100%[===================>]      26  --.-KB/s    in 0s

2021-12-02 09:38:24 (30.0 MB/s) - ‘models/open-images-model-labels.txt.1’ saved
[26/26]
```

In [ ]:
```
1 !pip install boto3
```

```
Collecting boto3
  Downloading boto3-1.20.18-py3-none-any.whl (131 kB)
     |████████████████████████████████| 131 kB 8.6 MB/s
Collecting botocore<1.24.0,>=1.23.18
  Downloading botocore-1.23.18-py3-none-any.whl (8.4 MB)
     |████████████████████████████████| 8.4 MB 33.1 MB/s
Collecting s3transfer<0.6.0,>=0.5.0
  Downloading s3transfer-0.5.0-py3-none-any.whl (79 kB)
     |████████████████████████████████| 79 kB 10.6 MB/s
Collecting jmespath<1.0.0,>=0.7.1
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/lib/py
thon3.7/site-packages (from botocore<1.24.0,>=1.23.18->boto3) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /opt/conda/lib/python3.
7/site-packages (from botocore<1.24.0,>=1.23.18->boto3) (1.26.7)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packag
es (from python-dateutil<3.0.0,>=2.1->botocore<1.24.0,>=1.23.18->boto3) (1.16.
0)
Installing collected packages: jmespath, botocore, s3transfer, boto3
Successfully installed boto3-1.20.18 botocore-1.23.18 jmespath-0.10.0 s3transfe
r-0.5.0
```

In [ ]:
```
1 !python pytorch-ssd/open_images_downloader.py --root ~/data/open_images --cl
```

```
2021-12-02 10:57:58,075 - root - Download https://storage.googleapis.com/openim
ages/2018_04/class-descriptions-boxable.csv. (https://storage.googleapis.com/op
enimages/2018_04/class-descriptions-boxable.csv.)
2021-12-02 10:57:58,119 - root - Download https://storage.googleapis.com/openim
ages/2018_04/train/train-annotations-bbox.csv. (https://storage.googleapis.com/
openimages/2018_04/train/train-annotations-bbox.csv.)
2021-12-02 10:58:08,626 - root - Read annotation file /home/parim/data/open_ima
ges/train-annotations-bbox.csv
```

In [ ]:
```
!python pytorch-ssd/train_ssd.py --dataset_type open_images --datasets ~/dat
```

```
r-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is
 deprecated. If you meant to do this, you must specify 'dtype=object' when c
reating the ndarray
  mode = random.choice(self.sample_options)
/content/pytorch-ssd/vision/transforms/transforms.py:247: VisibleDeprecation
Warning: Creating an ndarray from ragged nested sequences (which is a list-o
r-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is
 deprecated. If you meant to do this, you must specify 'dtype=object' when c
reating the ndarray
  mode = random.choice(self.sample_options)
/content/pytorch-ssd/vision/transforms/transforms.py:247: VisibleDeprecation
Warning: Creating an ndarray from ragged nested sequences (which is a list-o
r-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is
 deprecated. If you meant to do this, you must specify 'dtype=object' when c
reating the ndarray
  mode = random.choice(self.sample_options)
2021-12-01 20:54:53,058 - root - INFO - Epoch: 88, Step: 100, Average Loss:
 1.8540, Average Regression Loss 0.5043, Average Classification Loss: 1.3497
/content/pytorch-ssd/vision/transforms/transforms.py:247: VisibleDeprecation
Warning: Creating an ndarray from ragged nested sequences (which is a list-o
```

In [ ]:
```
# Before finetuning
!python pytorch-ssd/eval_ssd.py --net mb1-ssd \
                                --trained_model models/mb1-ssd
                                --dataset_type open_images \
                                --dataset /root/data/open_imag
                                --label_file /content/models/o
```

```
process image 120
Load Image: 0.030503 seconds.
Inference time:  0.010792016983032227
Prediction: 0.044940 seconds.
process image 121
Load Image: 0.035109 seconds.
Inference time:  0.011407136917114258
Prediction: 0.043935 seconds.
process image 122
Load Image: 0.020850 seconds.
Inference time:  0.011580944061279297
Prediction: 0.049960 seconds.


Average Precision Per-class:
Handgun: 0.7526829425387441
Shotgun: 0.3529092048726733

Average Precision Across All Classes:0.5527960737057087
```

In [ ]:
```
# After finetuning
!python pytorch-ssd/eval_ssd.py --net mb1-ssd \
                                --trained_model models/mb1-ssd
                                --dataset_type open_images \
                                --dataset /root/data/open_imag
                                --label_file /content/models/o
```

```
Prediction: 0.029038 seconds.
process image 120
Load Image: 0.036938 seconds.
Inference time:  0.01143193244934082
Prediction: 0.041948 seconds.
process image 121
Load Image: 0.031048 seconds.
Inference time:  0.01518869400024414
Prediction: 0.038679 seconds.
process image 122
Load Image: 0.017745 seconds.
Inference time:  0.011113643646240234
Prediction: 0.029559 seconds.


Average Precision Per-class:
Handgun: 0.8130237277657372
Shotgun: 0.5787321045226184

Average Precision Across All Classes:0.6958779161441777
```

In [ ]:
```

```

In [1]:
```
!pip install onnx
```

```
Collecting onnx
  Downloading onnx-1.10.2-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_6
4.whl (12.7 MB)
     |████████████████████████████████| 12.7 MB 6.7 MB/s
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (f
rom onnx) (1.15.0)
Requirement already satisfied: typing-extensions>=3.6.2.1 in /usr/local/lib/pyt
hon3.7/dist-packages (from onnx) (3.10.0.2)
Requirement already satisfied: protobuf in /usr/local/lib/python3.7/dist-packag
es (from onnx) (3.17.3)
Requirement already satisfied: numpy>=1.16.6 in /usr/local/lib/python3.7/dist-p
ackages (from onnx) (1.19.5)
Installing collected packages: onnx
Successfully installed onnx-1.10.2
```

In [ ]:
```
# Converting to ONNX
# 1.3
!python pytorch-ssd/convert_to_caffe2_models.py mb1-ssd /content/models/mb1-
```

```
/usr/local/lib/python3.7/dist-packages/caffe2/python/onnx/backend.py:878: UserW
arning: ShapeInferenceWarning: utils module not found in ONNX version 1.10.2
  warnings.warn("ShapeInferenceWarning: utils module not found in ONNX version
{}".format(onnx.__version__))
/usr/local/lib/python3.7/dist-packages/caffe2/python/onnx/backend.py:885: UserW
arning: OptimizerWarning: optimizer module not found in ONNX version 1.10.2
  warnings.warn("OptimizerWarning: optimizer module not found in ONNX version
{}".format(onnx.__version__))
Save the model in binary format to the files models/mb1-ssd_init_net.pb and mod
els/mb1-ssd_predict_net.pb.
Save the model in txt format to the files models/mb1-ssd_init_net.pbtxt and mod
els/mb1-ssd_predict_net.pbtxt.
```

In [2]:
```
!git clone https://github.com/onnx/onnx.git
```

```
Cloning into 'onnx'...
remote: Enumerating objects: 29055, done.
remote: Counting objects: 100% (868/868), done.
remote: Compressing objects: 100% (492/492), done.
remote: Total 29055 (delta 504), reused 605 (delta 361), pack-reused 28187
Receiving objects: 100% (29055/29055), 20.78 MiB | 21.15 MiB/s, done.
Resolving deltas: 100% (16131/16131), done.
```

In [ ]:
```
!git clone https://gitlab.com/graphviz/graphviz.git
```

```
Cloning into 'graphviz'...
remote: Enumerating objects: 106337, done.
remote: Counting objects: 100% (1818/1818), done.
remote: Compressing objects: 100% (108/108), done.
remote: Total 106337 (delta 1744), reused 1753 (delta 1710), pack-reused 104519
Receiving objects: 100% (106337/106337), 176.20 MiB | 25.44 MiB/s, done.
Resolving deltas: 100% (80825/80825), done.
```

In [3]:
```
!pip install onnx
!pip install pydot
```

```
Requirement already satisfied: onnx in /usr/local/lib/python3.7/dist-packages
(1.10.2)
Requirement already satisfied: typing-extensions>=3.6.2.1 in /usr/local/lib/pyt
hon3.7/dist-packages (from onnx) (3.10.0.2)
Requirement already satisfied: protobuf in /usr/local/lib/python3.7/dist-packag
es (from onnx) (3.17.3)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (f
rom onnx) (1.15.0)
Requirement already satisfied: numpy>=1.16.6 in /usr/local/lib/python3.7/dist-p
ackages (from onnx) (1.19.5)
Requirement already satisfied: pydot in /usr/local/lib/python3.7/dist-packages
(1.3.0)
Requirement already satisfied: pyparsing>=2.1.4 in /usr/local/lib/python3.7/dis
t-packages (from pydot) (3.0.6)
```

In [ ]:
```
!git clone https://github.com/onnx/tutorials.git
```

```
Cloning into 'tutorials'...
remote: Enumerating objects: 654, done.
remote: Counting objects: 100% (65/65), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 654 (delta 17), reused 17 (delta 5), pack-reused 589
Receiving objects: 100% (654/654), 37.26 MiB | 32.09 MiB/s, done.
Resolving deltas: 100% (289/289), done.
```

In [6]:
```
import os
!python onnx/onnx/tools/net_drawer.py --input "mb1-ssd.onnx" --output squeez
```

In [7]:
```
!dot -Tsvg squeezenet.dot -o squeezenet.svg
```

In [ ]:
```
!pip install pillow
```

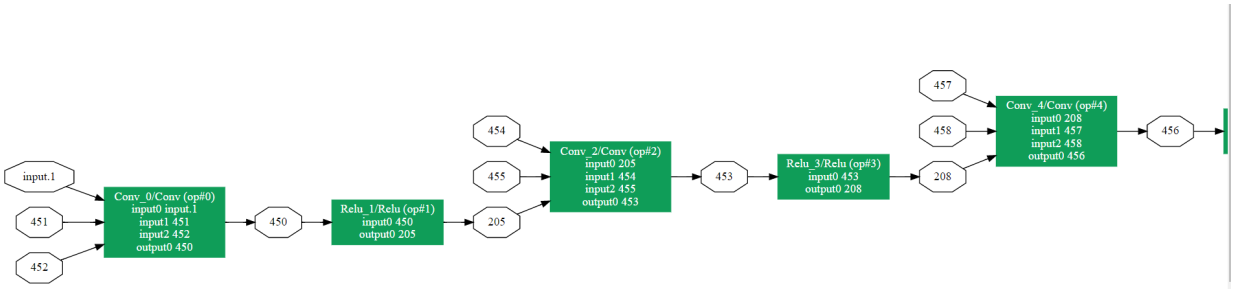
```
Requirement already satisfied: pillow in /opt/conda/lib/python3.7/site-packages
(8.4.0)
```

In [8]:
```
from IPython.display import SVG, display
display(SVG('squeezenet.svg'))
```

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

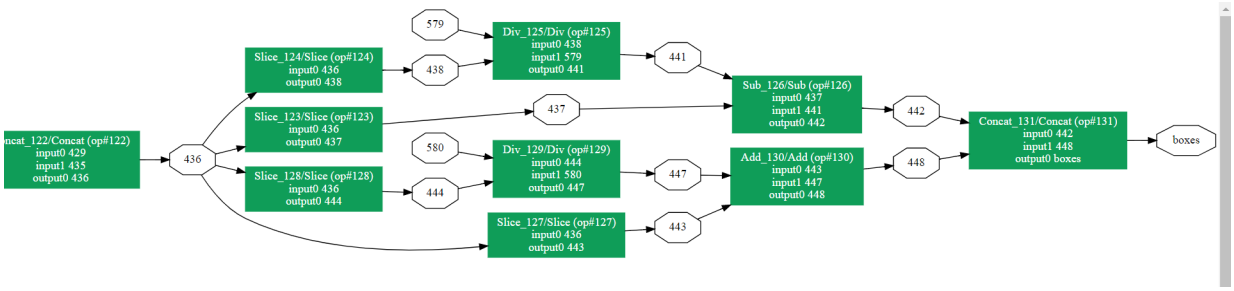
In [2]:
```
# As I cannot view the output of net_drawer in Jupyter notebook I have inclu
from PIL import Image
Image.open('C:/Users/parim/Desktop/Columbia Class/DL/hw5/net-start.PNG')
```

Out[2]:



In [3]:
```
Image.open('C:/Users/parim/Desktop/Columbia Class/DL/hw5/net-end.PNG')
```

Out[3]:



In [ ]:
```

```

In [ ]:
```
# 1.5,1.6
!pip install onnxruntime
```

In [ ]:
```

!sudo docker pull mcr.microsoft.com/onnxruntime/server
```

```
Using default tag: latest
latest: Pulling from onnxruntime/server
Digest: sha256:a23da0977bbc4aca4d3de56ad648ebde86031e61d7a3b7cbe1daaebbc7f6ad3d
Status: Image is up to date for mcr.microsoft.com/onnxruntime/server:latest
mcr.microsoft.com/onnxruntime/server:latest
```

In [ ]:
```
!sudo docker run -it -v $(pwd):$(pwd) -p 9005:8001 mcr.microsoft.com/onnxrun
```

```
tializer.cc:345 SaveInputOutputNamesToNodeMapping]: Graph input with name ba
ckbone.model.layer2.0.4.bn2.num_batches_tracked is not used by any node.
[2021-12-02 11:59:52.246] [ServerApp] [info] [ onnxruntime session_state_ini
tializer.cc:345 SaveInputOutputNamesToNodeMapping]: Graph input with name ba
ckbone.model.layer2.0.5.bn1.num_batches_tracked is not used by any node.
[2021-12-02 11:59:52.246] [ServerApp] [info] [ onnxruntime session_state_ini
tializer.cc:345 SaveInputOutputNamesToNodeMapping]: Graph input with name ba
ckbone.model.layer2.0.5.bn2.num_batches_tracked is not used by any node.
[2021-12-02 11:59:52.246] [ServerApp] [info] [ onnxruntime inference_sessio
n.cc:620 Initialize]: Session successfully initialized.
[2021-12-02 11:59:52.248] [ServerApp] [info] GRPC Listening at: 0.0.0.0:5005
1
[2021-12-02 11:59:52.248] [ServerApp] [info] Listening at: http://0.0.0.0:80
01 (http://0.0.0.0:8001)
[2021-12-02 12:08:33.415] [ac257471-add5-47a6-9d53-7ff6d99a95c2] [info] Mode
l Name: default, Version: 1, Action: predict
[2021-12-02 12:09:05.223] [cd2768d4-f409-438a-b2cf-a0bca1199302] [info] Mode
l Name: default, Version: 1, Action: predict
^C
```

In [ ]:
```
import os

import tutorials.tutorials.assets.onnx_ml_pb2 as onnx_ml_pb2
import tutorials.tutorials.assets.predict_pb2 as predict_pb2
```

In [ ]:
```
import tensorflow as tf
x = tf.cast(img2, tf.float32)
img2 = tf.keras.applications.mobilenet.preprocess_input(x)
```

In [ ]:
```
!pip install tensorflow
```

...

In [ ]:
```
import numpy as np
import requests
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.patches as patches
img1 = Image.open("ims/831f04c7b16ad888.jpg")
img1 = img1.resize((300, 300), Image.BILINEAR)
img1

```

Out[39]:

In [ ]:
```
img_data = np.array(img1)
img_data = np.transpose(img_data, [2, 0, 1])
img_data = np.expand_dims(img_data, 0)
mean_vec = np.array([0.5093532301981356, 0.47697768609685176, 0.443410724237
stddev_vec = np.array([0.17204669910914244, 0.1762167087185571, 0.1910801545
norm_img_data = np.zeros(img_data.shape).astype('float32')
for i in range(img_data.shape[1]):
    norm_img_data[:,i,:,:] = (img_data[:,i,:,:]/255 - mean_vec[i]) / stddev_
```

In [ ]:
```
input_tensor = onnx_ml_pb2.TensorProto()
input_tensor.dims.extend(norm_img_data.shape)
input_tensor.data_type = 1
input_tensor.raw_data = norm_img_data.tobytes()

request_message = predict_pb2.PredictRequest()

print(request_message.inputs)
# For your model, the inputs name should be something else customized by you
request_message.inputs["input.1"].data_type = input_tensor.data_type
request_message.inputs["input.1"].dims.extend(input_tensor.dims)
request_message.inputs["input.1"].raw_data = input_tensor.raw_data

content_type_headers = ['application/x-protobuf', 'application/octet-stream'

for h in content_type_headers:
    request_headers = {
        'Content-Type': h,
        'Accept': 'application/x-protobuf'
    }
```

```
{}
```

In [ ]:
```
PORT_NUMBER = 9005 # Change appropriately if needed based on any changes whe
inference_url = "http://127.0.0.1:" + str(PORT_NUMBER) + "/v1/models/default
response = requests.post(inference_url, headers=request_headers, data=reques
```

In [ ]:
```
# print(response.content)
response_message = predict_pb2.PredictResponse()
response_message.ParseFromString(response.content)

# print(response_message.outputs)
# For your model, the outputs names should be something else customized by y
bboxes = np.frombuffer(response_message.outputs['boxes'].raw_data, dtype=np.
# labels = np.frombuffer(response_message.outputs['labels'].raw_data, dtype=
scores = np.frombuffer(response_message.outputs['scores'].raw_data, dtype=np

print(scores)
print('Boxes shape:', response_message.outputs['boxes'].dims)
# print('Labels shape:', response_message.outputs['labels'].dims)
print('Scores shape:', response_message.outputs['scores'].dims)
```

```
[0.9897613  0.00536984 0.00486883 ... 0.7378382  0.25728315 0.00487863]
Boxes shape: [1, 3000, 4]
Scores shape: [1, 3000, 3]
```

In [ ]:

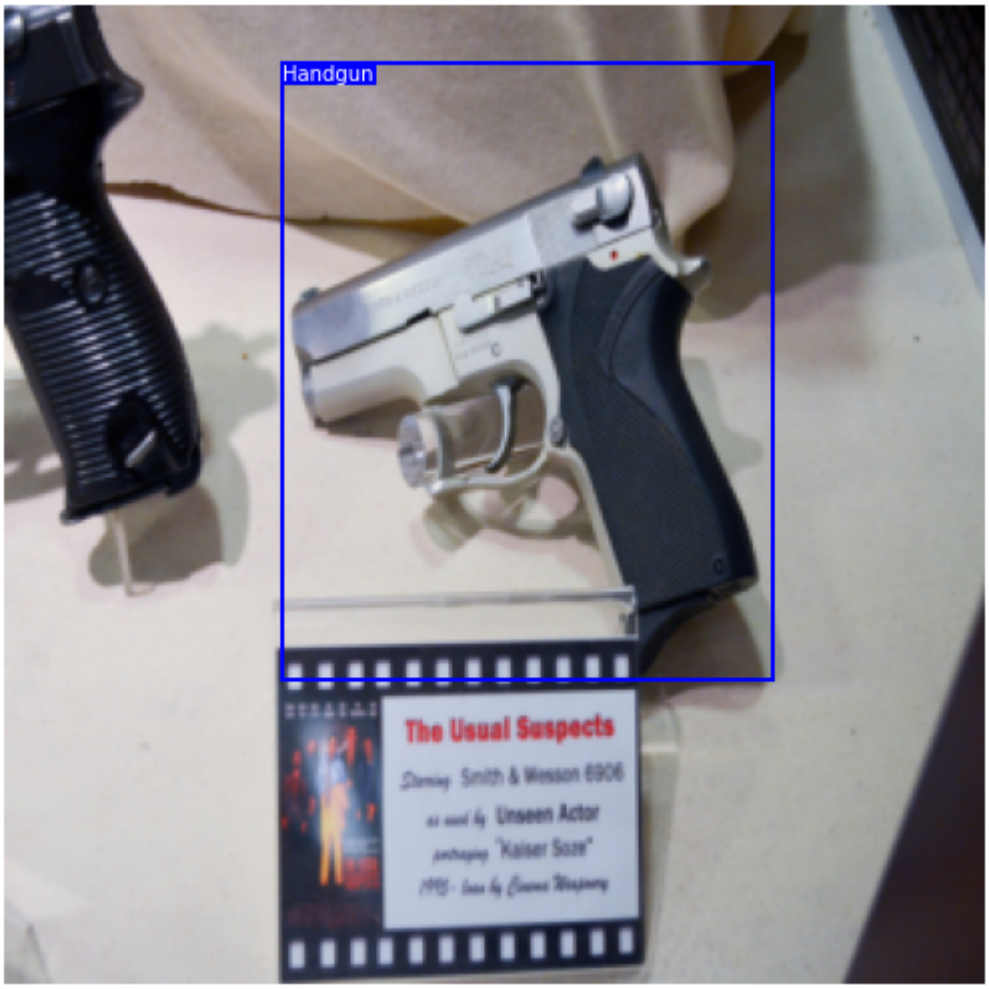
```
print(bboxes[0],bboxes[1],bboxes[2],bboxes[3])
print(len(bboxes),len(scores))
print(sum(scores>0.999),max(scores))
b = []
for i in range(len(scores)//3):
    if max([scores[3*i+1]])>0.99:
        b.append([bboxes[4*i],bboxes[4*i+1],bboxes[4*i+2],bboxes[4*i+3]])
        print([scores[3*i],scores[3*i+1],scores[3*i+2]])
print(b)
print(len(b))
print([[300*i for i in j] for j in b])
```

```
-0.020249806 -0.011044905 0.10776336 0.107537776
12000 9000
600 0.99998856
[0.008273488, 0.99049395, 0.0012325967]
[[0.2823299, 0.059160978, 0.7833177, 0.68726707]]
1
[[84.69896614551544, 17.7482932806015, 234.9953055381775, 206.18011951446533]]
```

In [ ]:

```
# Parse the list of class labels
classes = [line.rstrip('\n') for line in open('models/open-images-model-labe
# classes = ['Handgun','Shotgun']

# Plot the bounding boxes on the image
plt.figure()
fig, ax = plt.subplots(1, figsize=(12,9))
ax.imshow(img1)

resized_width = 300  # we resized the original image, remember ?
resized_height = 300
num_boxes = len(b) # we limit displaying to just 10 boxes to avoid clogging
               # The results are already sorted based on box confidences, so

for c in range(num_boxes):
    y1, x1, y2, x2 = b[c][0] * resized_height, b[c][1] * resized_width, b[c]
    color = 'blue'
    box_h = (y2 - y1)
    box_w = (x2 - x1)
    bbox = patches.Rectangle((y1, x1), box_h, box_w, linewidth=2, edgecolor=
    ax.add_patch(bbox)
#     print(labels[c] - 1)
    plt.text(y1, x1, s=classes[1], color='white', verticalalignment='top', b
plt.axis('off')

# Save image
# plt.savefig("output/ssd_result.jpg", bbox_inches='tight', pad_inches=0.0)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```


Handgun
The Usual Suspects
Smith & Wesson 6906
Unseen Actor
"Kaiser Soze"

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
img2 = Image.open("ims/bf7b28e33c453f4a.jpg")
img2 = img2.resize((300, 300), Image.BILINEAR)
img2
```

Out[64]:

In [ ]:
```
img_data = np.array(img2)
img_data = np.transpose(img_data, [2, 0, 1])
img_data = np.expand_dims(img_data, 0)
mean_vec = np.array([0.5093532301981356, 0.47697768609685176, 0.443410724237
stddev_vec = np.array([0.17204669910914244, 0.1762167087185571, 0.1910801545
norm_img_data = np.zeros(img_data.shape).astype('float32')
for i in range(img_data.shape[1]):
    norm_img_data[:,i,:,:] = (img_data[:,i,:,:]/255 - mean_vec[i]) / stddev_
```

In [ ]:
```
input_tensor = onnx_ml_pb2.TensorProto()
input_tensor.dims.extend(norm_img_data.shape)
input_tensor.data_type = 1
input_tensor.raw_data = norm_img_data.tobytes()

request_message = predict_pb2.PredictRequest()

print(request_message.inputs)
# For your model, the inputs name should be something else customized by you
request_message.inputs["input.1"].data_type = input_tensor.data_type
request_message.inputs["input.1"].dims.extend(input_tensor.dims)
request_message.inputs["input.1"].raw_data = input_tensor.raw_data

content_type_headers = ['application/x-protobuf', 'application/octet-stream'

for h in content_type_headers:
    request_headers = {
        'Content-Type': h,
        'Accept': 'application/x-protobuf'
    }
```

```
{}
```

In [ ]:
```
PORT_NUMBER = 9005 # Change appropriately if needed based on any changes whe
inference_url = "http://127.0.0.1:" + str(PORT_NUMBER) + "/v1/models/default
response = requests.post(inference_url, headers=request_headers, data=reques
```

In [ ]:
```
# print(response.content)
response_message = predict_pb2.PredictResponse()
response_message.ParseFromString(response.content)

# print(response_message.outputs)
# For your model, the outputs names should be something else customized by y
bboxes = np.frombuffer(response_message.outputs['boxes'].raw_data, dtype=np.
# labels = np.frombuffer(response_message.outputs['labels'].raw_data, dtype=
scores = np.frombuffer(response_message.outputs['scores'].raw_data, dtype=np

print(scores)
print('Boxes shape:', response_message.outputs['boxes'].dims)
# print('Labels shape:', response_message.outputs['labels'].dims)
print('Scores shape:', response_message.outputs['scores'].dims)
```

```
[0.9965726  0.00137203 0.00205533 ... 0.9824255  0.00197902 0.01559554]
Boxes shape: [1, 3000, 4]
Scores shape: [1, 3000, 3]
```

In [ ]:
```
print(bboxes[0],bboxes[1],bboxes[2],bboxes[3])
print(len(bboxes),len(scores))
print(sum(scores>0.999),max(scores))
b = []
for i in range(len(scores)//3):
    if max([scores[3*i+2]])>0.96:
        b.append([bboxes[4*i],bboxes[4*i+1],bboxes[4*i+2],bboxes[4*i+3]])
        print([scores[3*i],scores[3*i+1],scores[3*i+2]])
print(b)
print(len(b))
print([[300*i for i in j] for j in b])
```

```
-0.020073365 -0.010506172 0.1116215 0.09724927
12000 9000
910 1.0
[0.024588618, 0.008374293, 0.96703714]
[[0.14970711, 0.27332062, 0.7277031, 0.707513]]
1
[[44.91213262081146, 81.99618458747864, 218.31092834472656, 212.2538924217224]]
```

In [ ]:

```
# Parse the list of class labels
classes = [line.rstrip('\n') for line in open('models/open-images-model-labe
# classes = ['Handgun','Shotgun']

# Plot the bounding boxes on the image
plt.figure()
fig, ax = plt.subplots(1, figsize=(12,9))
ax.imshow(img2)

resized_width = 300  # we resized the original image, remember ?
resized_height = 300
num_boxes = len(b) # we limit displaying to just 10 boxes to avoid clogging
               # The results are already sorted based on box confidences, so

for c in range(num_boxes):
    y1, x1, y2, x2 = b[c][0] * resized_height, b[c][1] * resized_width, b[c]
    color = 'blue'
    box_h = (y2 - y1)
    box_w = (x2 - x1)
    bbox = patches.Rectangle((y1, x1), box_h, box_w, linewidth=2, edgecolor=
    ax.add_patch(bbox)
#     print(labels[c] - 1)
    plt.text(y1, x1, s=classes[2], color='white', verticalalignment='top', b
plt.axis('off')

# Save image
# plt.savefig("output/ssd_result.jpg", bbox_inches='tight', pad_inches=0.0)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

In [ ]:
```
1
```

In [ ]:
```
1
```

Problem 2

```
1. DL Frameworks supported and their versions:

IBM, Google, Microsoft, and Amazon

a) IBM :- IBM Cloud Paks 4.0:<br>
i. Pytorch version - 1.7<br>
ii. Tensorflow version - 2.4<br>
Other useful ML frameworks: XGBoost version(1.3), Scikit-learn(0.23),
Spark version(3.0)

b) Google :- GCP:<br>
i. Tensorflow version - upto 2.7 (Long term vevrsion support for 2.6)<br>
ii. Pytorch version - 1.10<br>
iii. MXNet version - 1.5.1<br>
iv. XGBoost version - 1.4.2<br>
v. Caffe version - 1.0<br>
Other frameworks include - Pytorch-XLA, RAPIDS, Chainer, CNTK, etc.
```

c) Microsoft :- Azure:
i. Pytorch version - 1.9.0
ii. Tensorflow version - 2.5
iii. Chainer version - 7.0.0
Other frameworks include - Horovod(0.21.3), CUDA/cudNN(11), CNTK, scikit-learn

d) Amazon :- AWS:
i. Pytorch version - 1.6
ii. Tensorflow version - 2.1
iii. Keras version - 2.2.4
iv. ONNX version - 1.7.0
v. MXNet version - 1.8.0
vi. Caffe version - 1.0
ther frameworks include - XGBoost(1.3.3), Gluon toolkit, CNTK(2.4), Theano(1.0)

2. Compute Units (GPUs):

a) IBM :- Nvidia K80, V100, P100, T4
b) Google :- Nvidia K80, P4, V100, A100, T4, P100
c) Microsoft :- Nvidia V100, P40, M60, K80, T4, P100
d) Amazon :- Nvidia A100, V100, K80, T4, M60, A10G, AMD Radeon Pro V520

3. Model lifecycle management:

a) IBM :- IBM Cloud Pak
b) Google :- Google cloud interface with KubeFlow
c) Microsoft :- MLOps
d) Amazon :- Amazon SageMaker

4. Monitoring:

a) IBM :- IBM Cloud Application Performance Management (APM), LogDNA, Sysdig
b) Google :- Google Cloud's operations suite
c) Microsoft :- Azure Monitor
d) Amazon :- Amazon CloudWatch

5. Visualization:

a) IBM :- Watson OpenScale
b) Google :- Tensorboard while using Tensorflow
c) Microsoft :- Azure Monitor Metrics
d) Amazon :- Amazon SageMaker has built-in support for metrics

6. Elastic Scaling:

a) IBM :- AutoScale can be used for elastic scaling
b) Google :- Google Cloud offers server-side load balancing and autoscaling for groups of instances so we can distribute incoming traffic across multiple virtual machine instances

c) Microsoft :- Azure virtual machine scale set can automatically increase or decrease the number of VM instances that run your application. Autoscale is a built-in feature of Cloud Services, Mobile Services, Virtual Machines, and Websites that helps applications perform their best when demand changes. For example, you could have a web app that handles millions of requests during the day and none at night. Autoscale can scale your service by any of these—or by a custom metric you define.
d) Amazon :- AWS Auto Scaling lets you build scaling plans that automate how groups of different resources respond to changes in demand. You can optimize availability, costs, or a balance of both.

7. Training job description:

a) IBM :-

```
 model_definition:
  framework:
    name: tensorflow
    version: "1.15"
    runtimes:
      name: python
      version: "3.5"
  name: tf-mnist
  description: Simple MNIST model implemented in TF
  execution:
    command: python3 convolutional_network.py --trainImagesFile
${DATA_DIR}/train-images-idx3-ubyte.gz
      --trainLabelsFile ${DATA_DIR}/train-labels-idx1-ubyte.gz --
testImagesFile ${DATA_DIR}/t10k-images-idx3-ubyte.gz
      --testLabelsFile ${DATA_DIR}/t10k-labels-idx1-ubyte.gz --
learningRate 0.001
      --trainingIters 2000000
    compute_configuration:
      name: v100
training_data_reference:
  name: MNIST image data files
  connection:
    endpoint_url: <auth-url>
    access_key_id: <username>
    secret_access_key: <password>
  source:
    bucket: mnist-training-data
  type: s3
training_results_reference:
  name: DL Model Storage
  connection:
    endpoint_url: <auth-url>
    access_key_id: <username>
    secret_access_key: <password>
  target:
    bucket: mnist-training-models
  type: s3
```

b) Google :-

```
PACKAGE_PATH="/path/to/your/application/sources"
```

```
now=(date +"%Y%m%d_%H%M%S")
JOB_NAME="your_name_now"
MODULE_NAME="trainer.task"
JOB_DIR="gs://your/chosen/job/output/path"
REGION="us-east1"
RUNTIME_VERSION="2.6"

gcloud ai-platform jobs submit training JOB_NAME \
        --scale-tier basic \
        --package-path PACKAGE_PATH \
        --module-name MODULE_NAME \
        --job-dir JOB_DIR \
        --region REGION \
        -- \
        --user_first_arg=first_arg_value \
        --user_second_arg=second_arg_value




c) Microsoft :- <br>
 from azureml.core import ScriptRunConfig, Experiment

   experiment = Experiment(workspace, 'MyExperiment')
   cluster = workspace.compute_targets['MyCluster']
   env = Environment.get(ws, name='MyEnvironment')
   config = ScriptRunConfig(source_directory='.',
                            script='train.py',
                            arguments=['--arg1', arg1_val, '--arg2',
arg2_val],
                            compute_target=cluster,
                            environment=env)
   script_run = experiment.submit(config)



d) Amazon :-
 create-training-job
--training-job-name <value>
[--hyper-parameters <value>]
--algorithm-specification <value>
--role-arn <value>
[--input-data-config <value>]
--output-data-config <value>
--resource-config <value>
[--vpc-config <value>]
--stopping-condition <value>
[--tags <value>]
[--enable-network-isolation | --no-enable-network-isolation]
[--enable-inter-container-traffic-encryption | --no-enable-inter-
container-traffic-encryption]
[--enable-managed-spot-training | --no-enable-managed-spot-training]
[--checkpoint-config <value>]
[--debug-hook-config <value>]
[--debug-rule-configurations <value>]
[--tensor-board-output-config <value>]
[--experiment-config <value>]
```

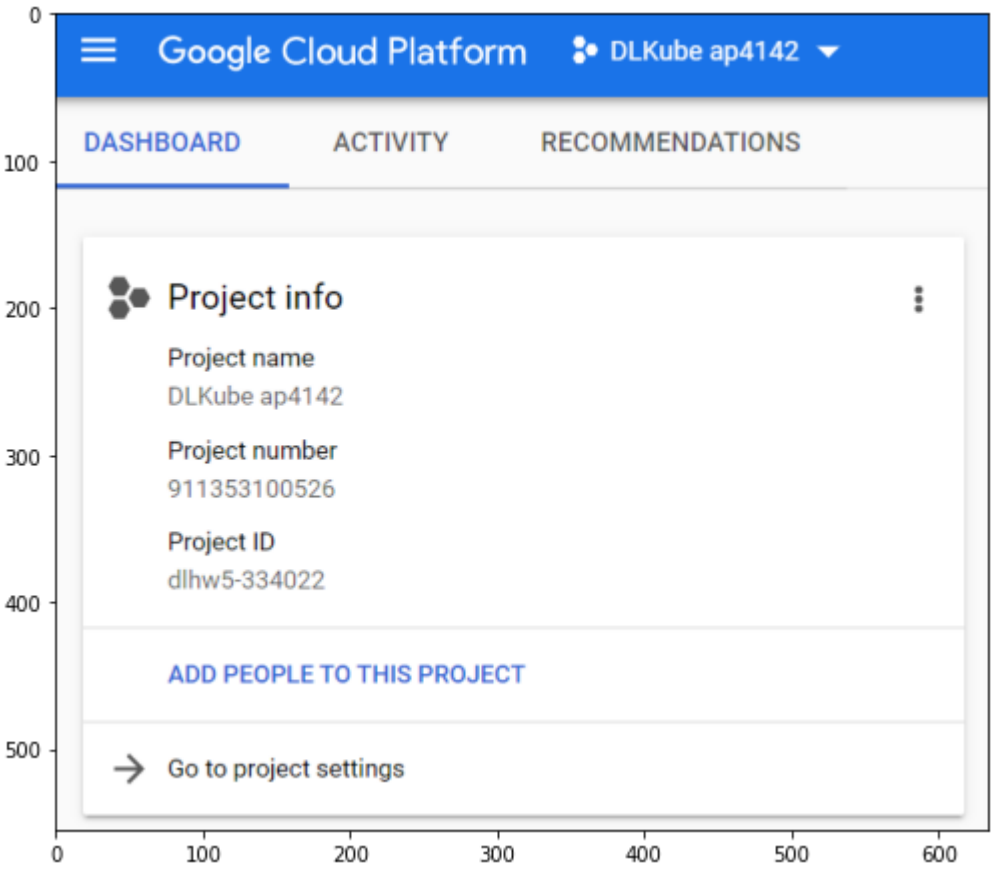
```
[--profiler-config <value>]
[--profiler-rule-configurations <value>]
[--environment <value>]
[--retry-strategy <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton <value>]


As we can see the different cloud platforms require similar data like the framaework to be used, the pythhon file to be run, hyperparameter values, locatioin of thhe file, environment or workspace to be used, package path, stopoping condition or number of iterations or time to be run.

```

Problem 3

In [ ]:

```
from PIL import Image
import glob
import matplotlib.pyplot as plt

print('(a)')
files = glob.glob('3.1.1*')
for f in files:
    fig = plt.gcf()
    fig.set_size_inches(10, 7.5)
    img = Image.open(f)
    plt.imshow(img)
    plt.show()

```

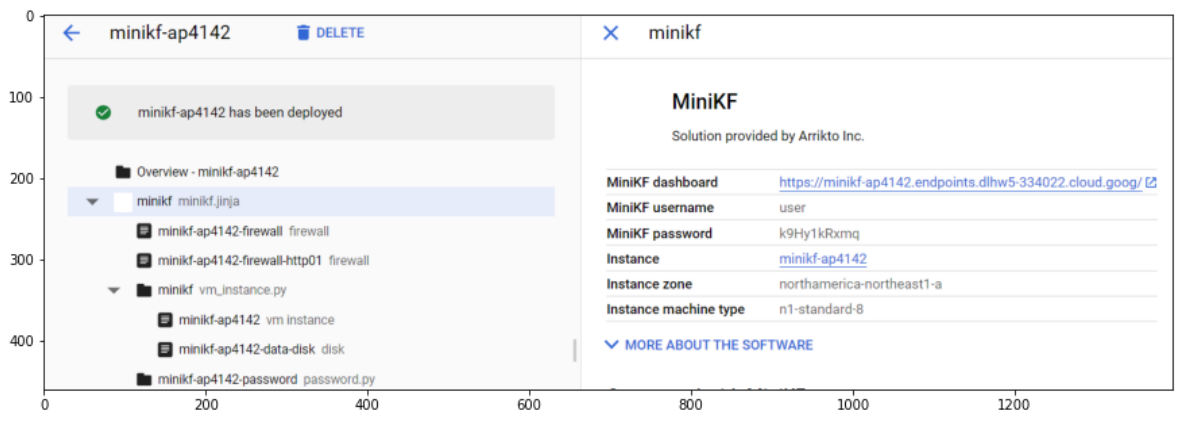
(a)

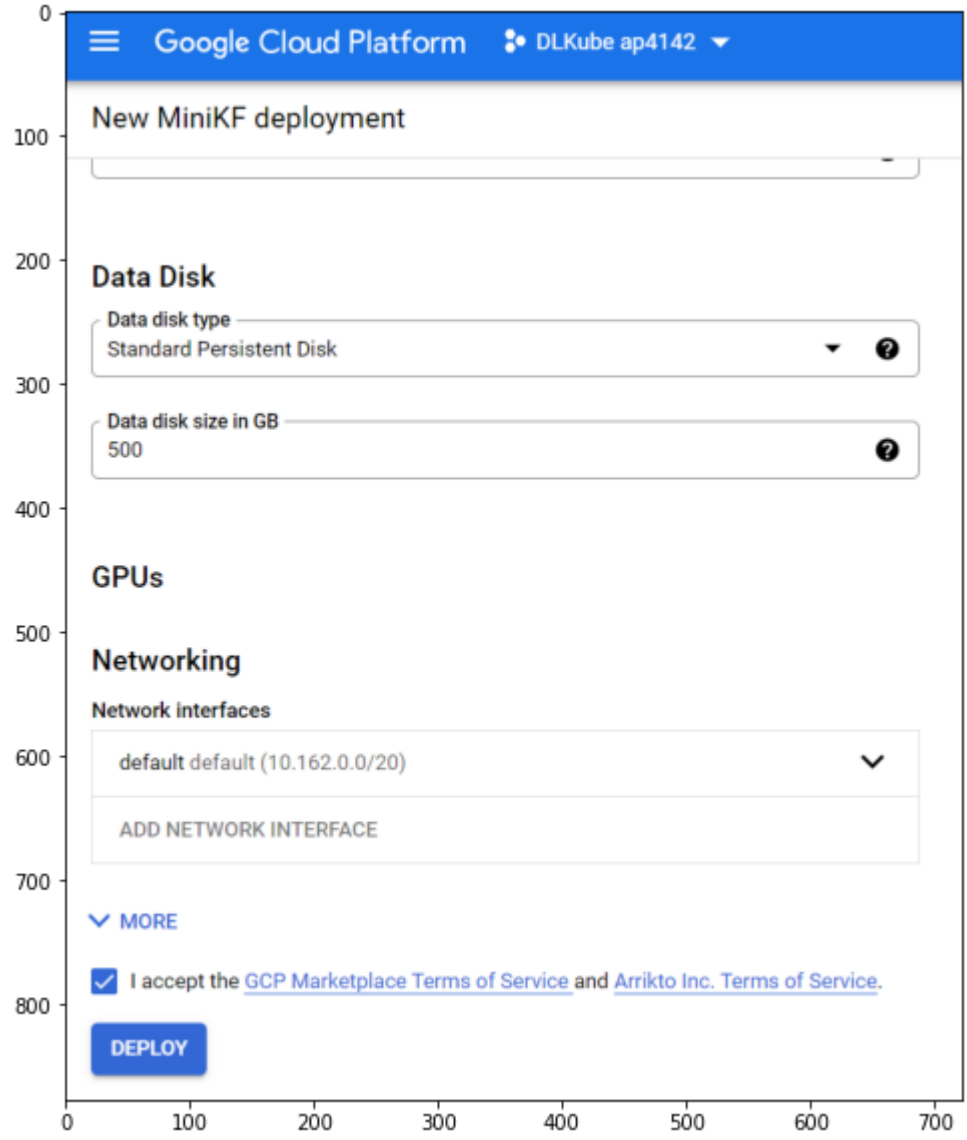


In [ ]:

```
print('(b)')
files = glob.glob('3.1.2*')
for f in files:
    fig = plt.gcf()
    fig.set_size_inches(15,10)
    img = Image.open(f)
    plt.imshow(img)
    plt.show()
```

(b)

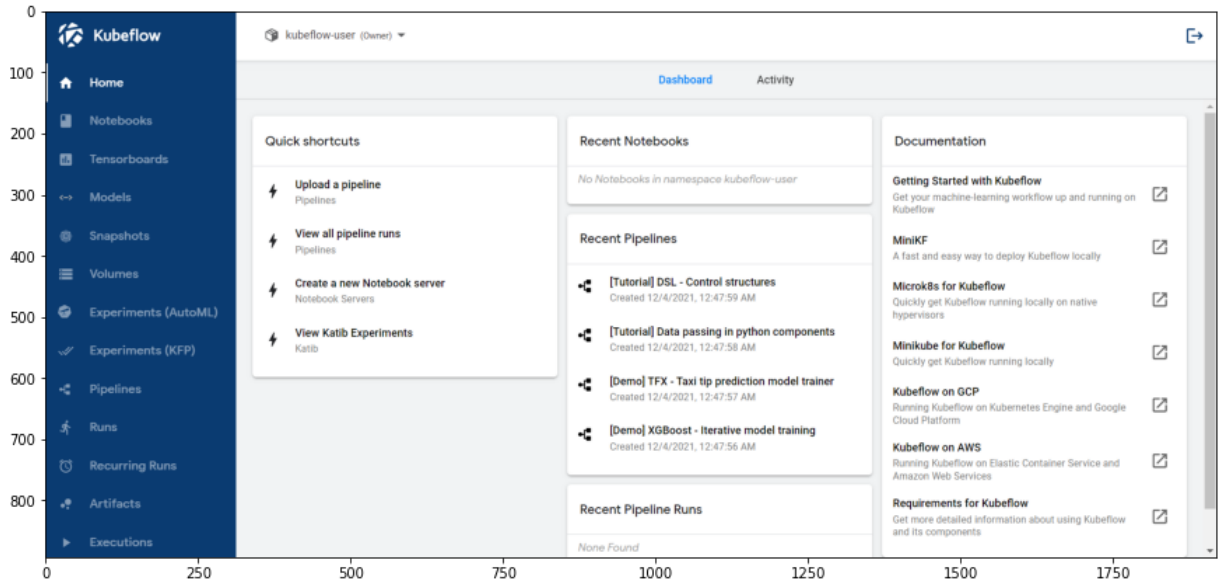

minikf-ap4142
DELETE
minikf
minikf-ap4142 has been deployed
Overview - minikf-ap4142
minikf minikf.jinja
minikf-ap4142-firewall firewall
minikf-ap4142-firewall-http01 firewall
minikf vm_instance.py
minikf-ap4142 vm instance
minikf-ap4142-data-disk disk
minikf-ap4142-password password.py
MiniKF
Solution provided by Arrikto Inc.
MiniKF dashboard
https://minikf-ap4142.endpoints.dlhw5-334022.cloud.goog/
MiniKF username
user
MiniKF password
k9Hy1kRxmq
Instance
minikf-ap4142
Instance zone
northamerica-northeast1-a
Instance machine type
n1-standard-8
MORE ABOUT THE SOFTWARE

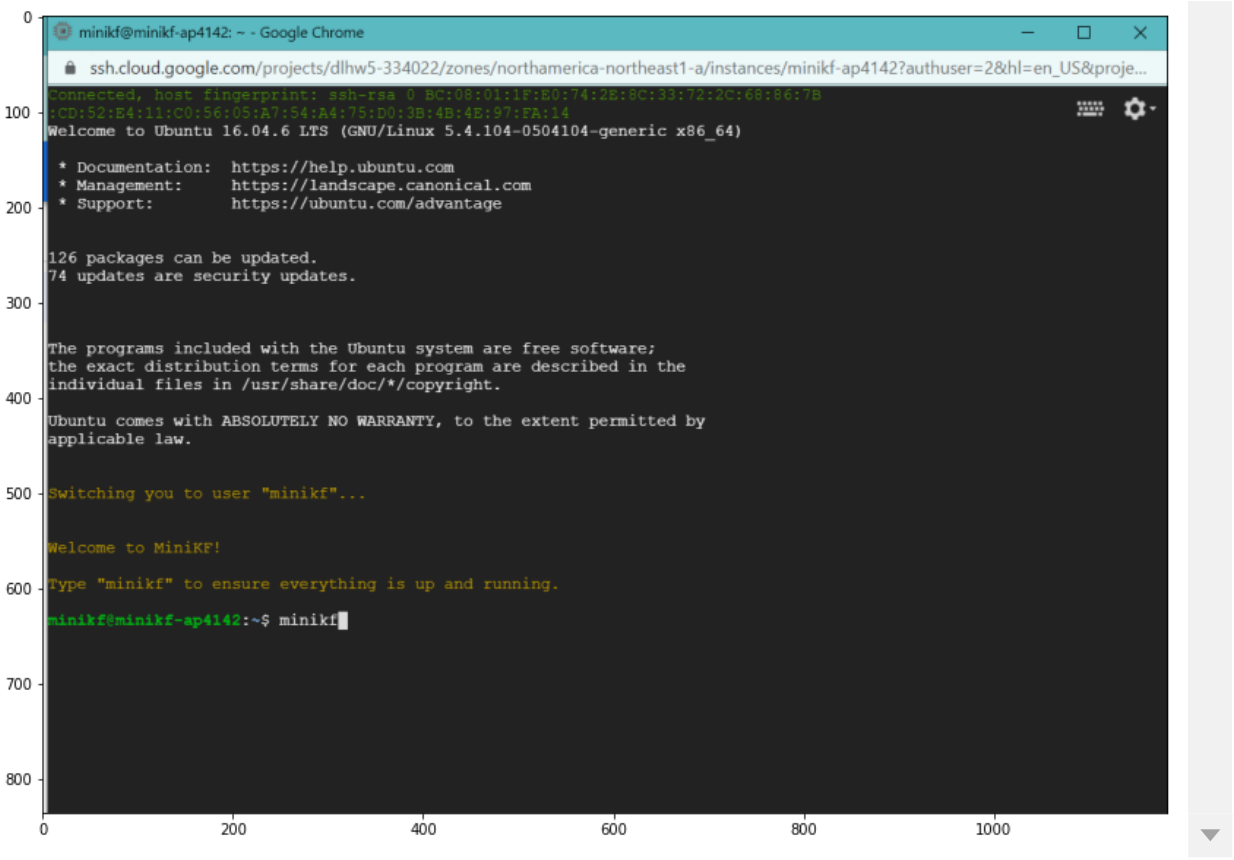

Google Cloud Platform
DLKube ap4142
New MiniKF deployment
Data Disk
Data disk type
Standard Persistent Disk
Data disk size in GB
500
GPUs
Networking
Network interfaces
default default (10.162.0.0/20)
ADD NETWORK INTERFACE
MORE
I accept the GCP Marketplace Terms of Service and Arrikto Inc. Terms of Service.
DEPLOY

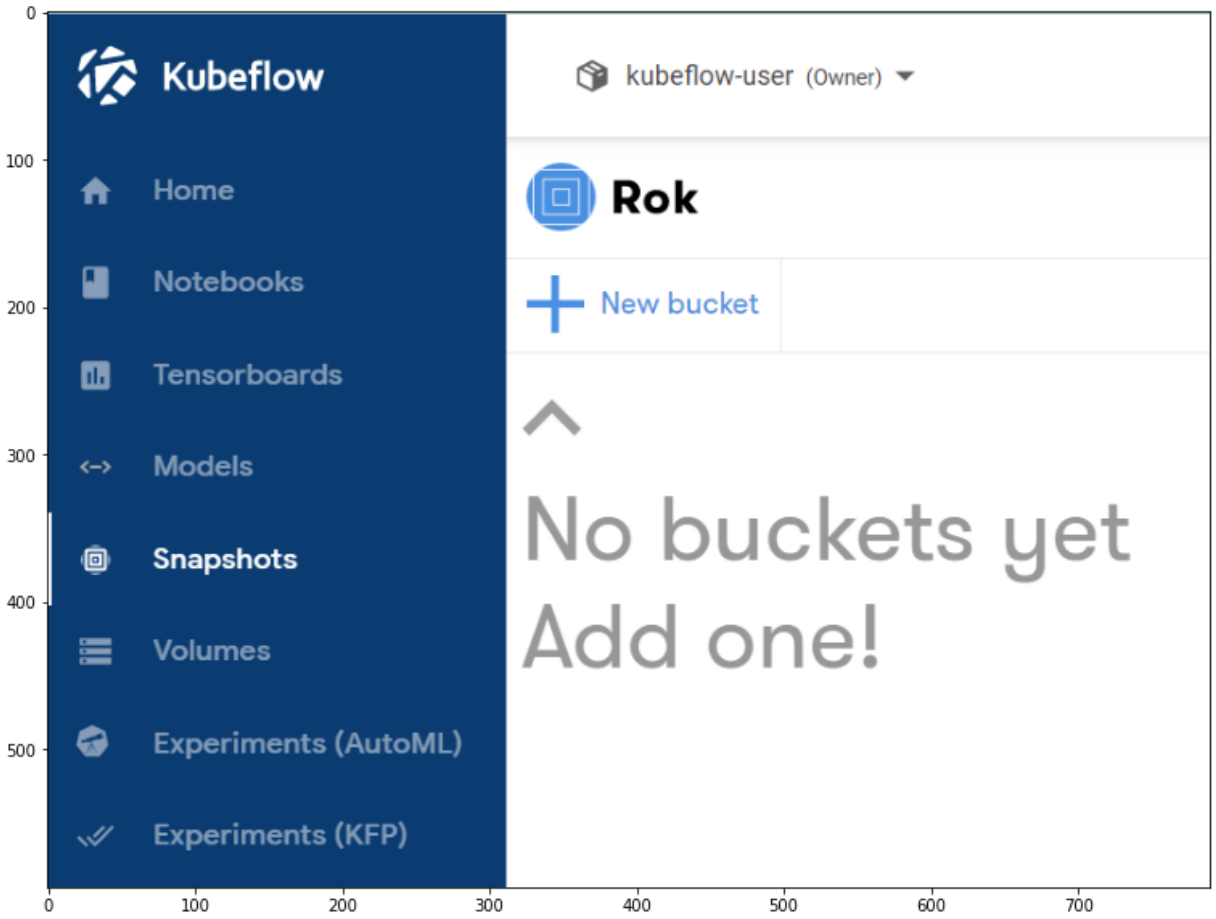
In [ ]:

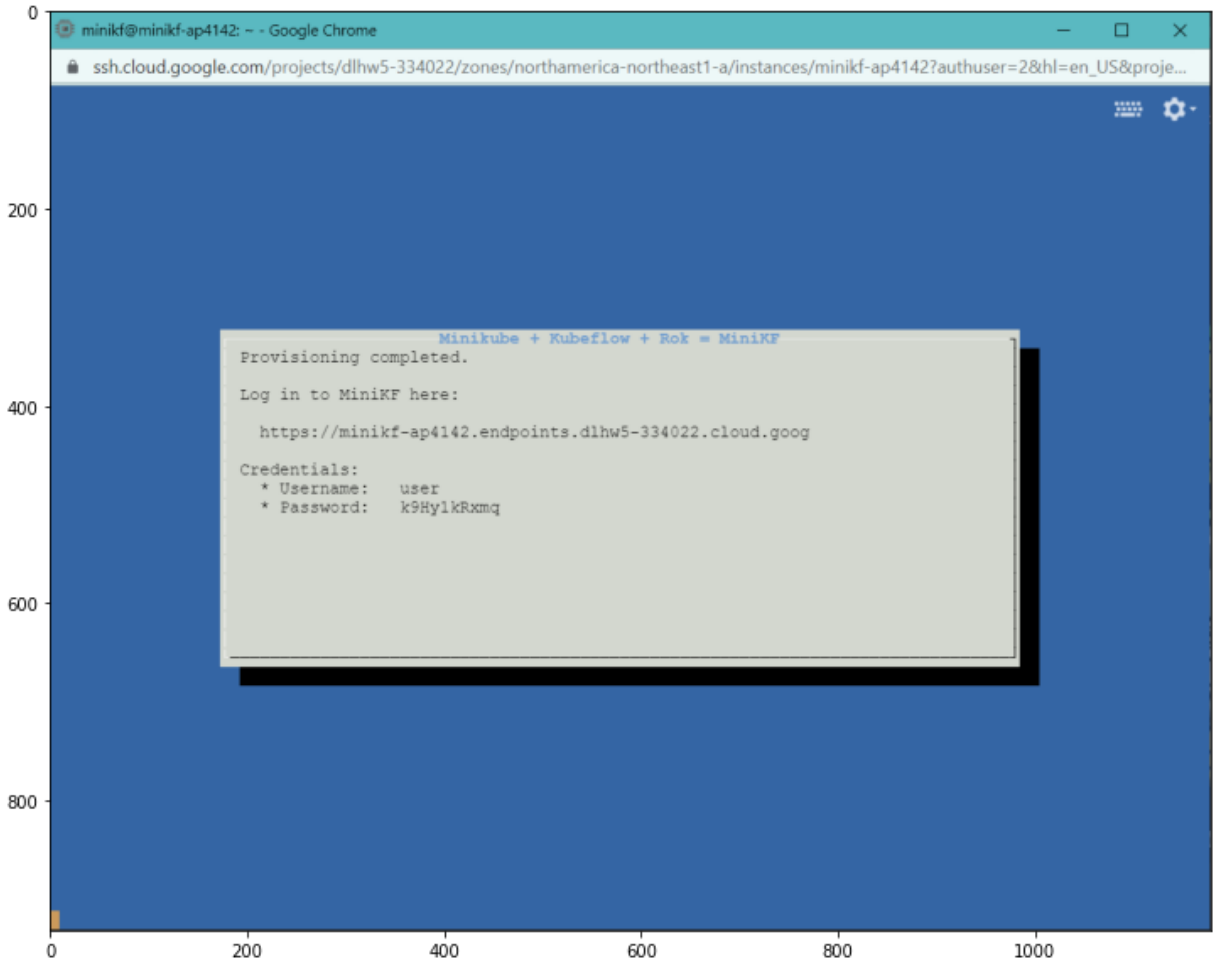
```
print('(c)')
files = glob.glob('3.1.3*')
for f in files:
    fig = plt.gcf()
    fig.set_size_inches(15,10)
    img = Image.open(f)
    plt.imshow(img)
    plt.show()
```

(c)




minikf@minikf-ap4142: ~ - Google Chrome
ssh.cloud.google.com/projects/dlhw5-334022/zones/northamerica-northeast1-a/instances/minikf-ap4142?authuser=2&hl=en_US&proje...
Connected, host fingerprint: ssh-rsa 0 BC:08:01:1F:E0:74:2B:8C:33:72:2C:68:86:7B
:CD:52:E4:11:C0:56:05:A7:54:A4:75:D0:3B:4B:4E:97:FA:14
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 5.4.104-0504104-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
126 packages can be updated.
74 updates are security updates.
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
Switching you to user "minikf"...
Welcome to MiniKF!
Type "minikf" to ensure everything is up and running.
minikf@minikf-ap4142:~$ minikf

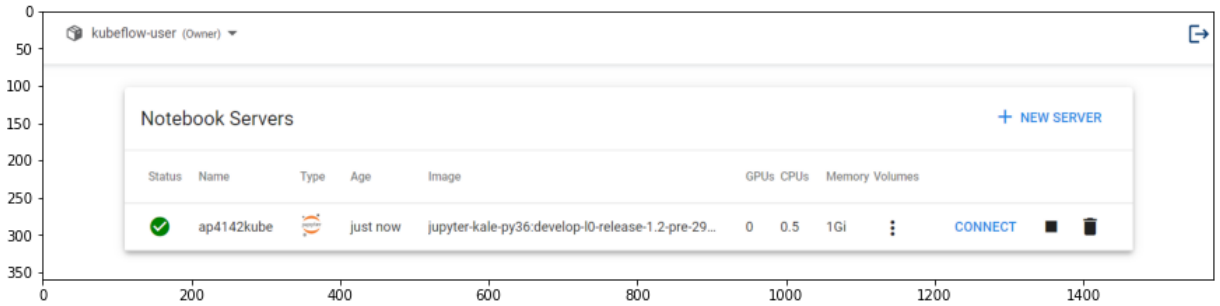

Kubeflow
kubeflow-user (Owner)
Home
Notebooks
Tensorboards
Models
Snapshots
Volumes
Experiments (AutoML)
Experiments (KFP)
Rok
New bucket
No buckets yet
Add one!

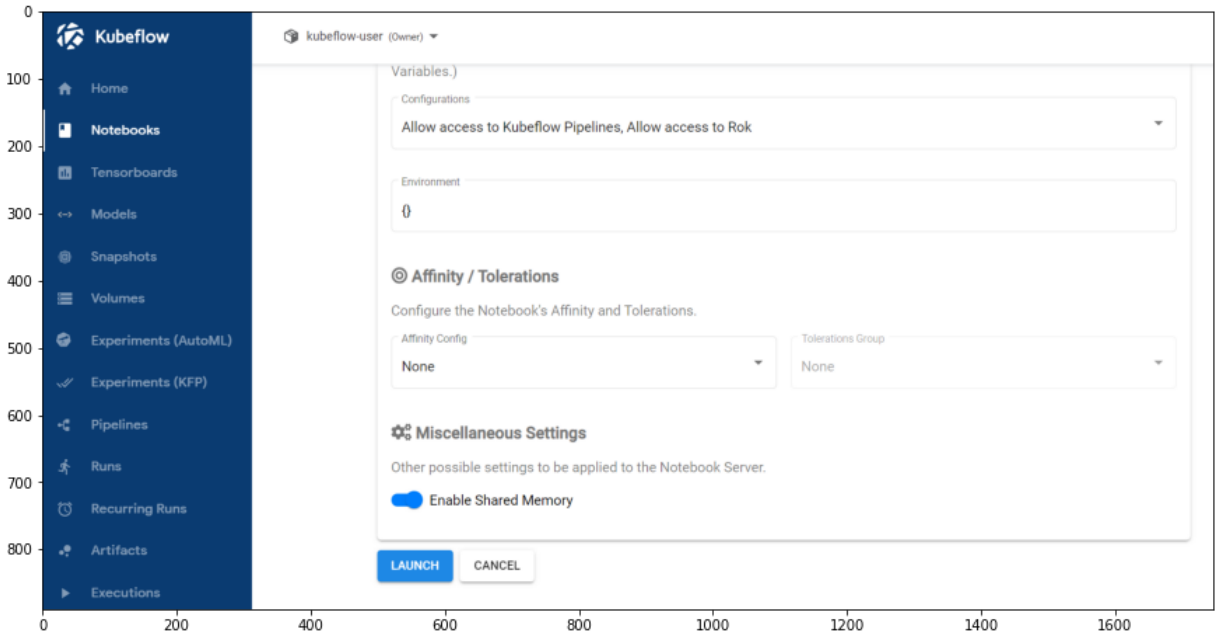

minikf@minikf-ap4142: ~ - Google Chrome
ssh.cloud.google.com/projects/dlhw5-334022/zones/northamerica-northeast1-a/instances/minikf-ap4142?authuser=2&hl=en_US&proje...
Minikube + Kubeflow + Rok = MiniKF
Provisioning completed.
Log in to MiniKF here:
https://minikf-ap4142.endpoints.dlhw5-334022.cloud.goog
Credentials:
* Username: user
* Password: k9HylkRxmq
0
200
400
600
800
0
200
400
600
800
1000

In [ ]:

```
print('(a)')
files = glob.glob('3.2.1*')
for f in files:
    fig = plt.gcf()
    fig.set_size_inches(15,10)
    img = Image.open(f)
    plt.imshow(img)
    plt.show()
```

(a)



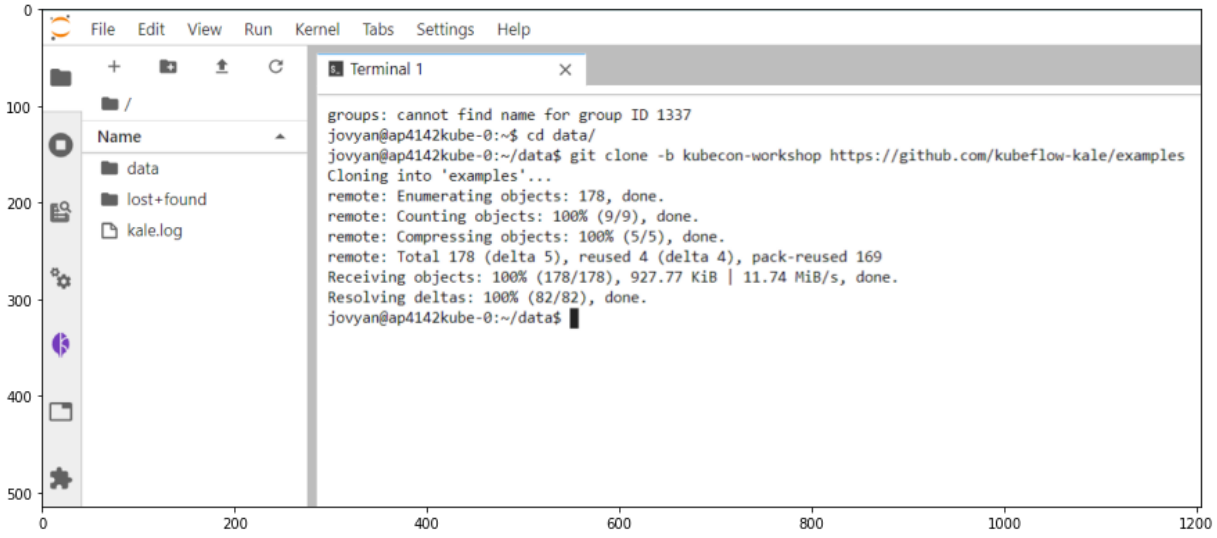


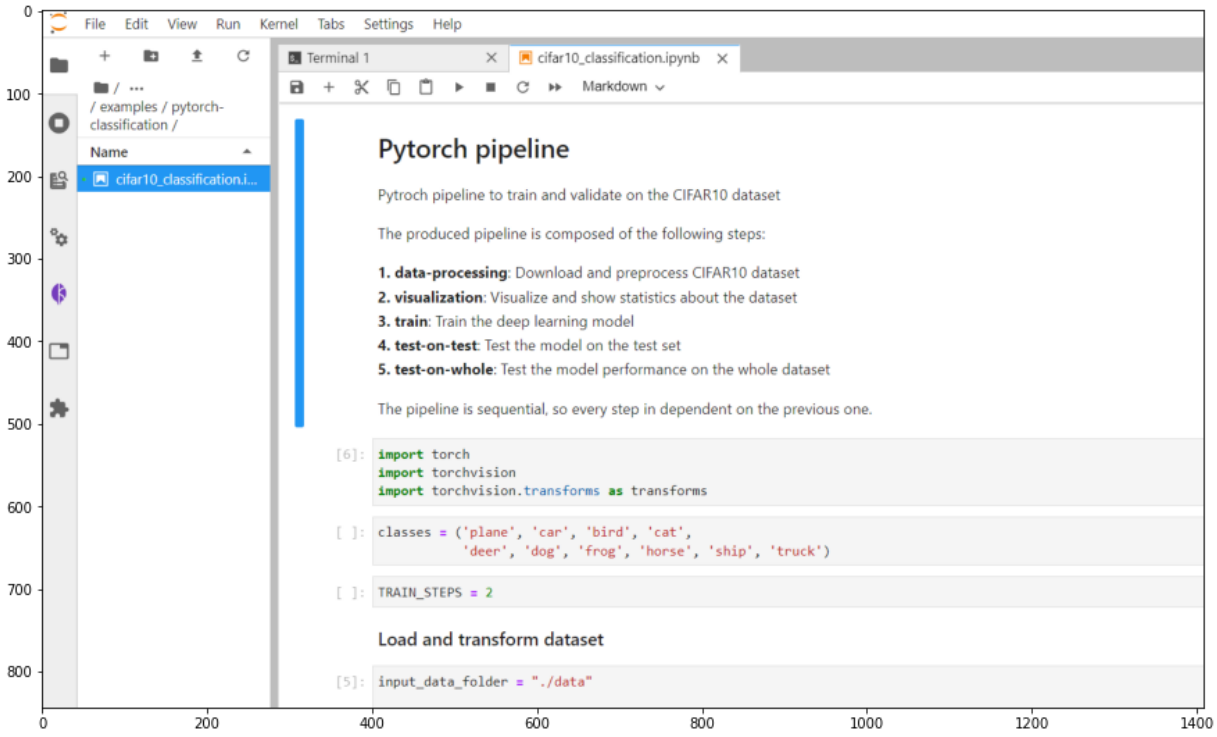

Kubeflow
kubeflow-user (Owner)
Home
Notebooks
Tensorboards
Models
Snapshots
Volumes
Experiments (AutoML)
Experiments (KFP)
Pipelines
Runs
Recurring Runs
Artifacts
Executions
Rok JupyterLab URL
Load an existing Jupyter Lab by providing a valid Rok URL.
Rok URL
Name
Specify the name of the Notebook Server and the Namespace it will belong to.
Name
ap4142kube
Namespace
kubeflow-user
Image
A started Jupyter Docker Image with a baseline deployment and typical ML packages
Custom Image
jupyterlab
Visual Studio Code
RStudio
Image
gcr.io/arrikto/jupyter-kale-py36:develop-l0-release-1.2-pre-295-g622fe91aca

In [ ]:

```
print('(b)')
files = glob.glob('3.2.2*')
for f in files:
    fig = plt.gcf()
    fig.set_size_inches(15,10)
    img = Image.open(f)
    plt.imshow(img)
    plt.show()
```

(b)

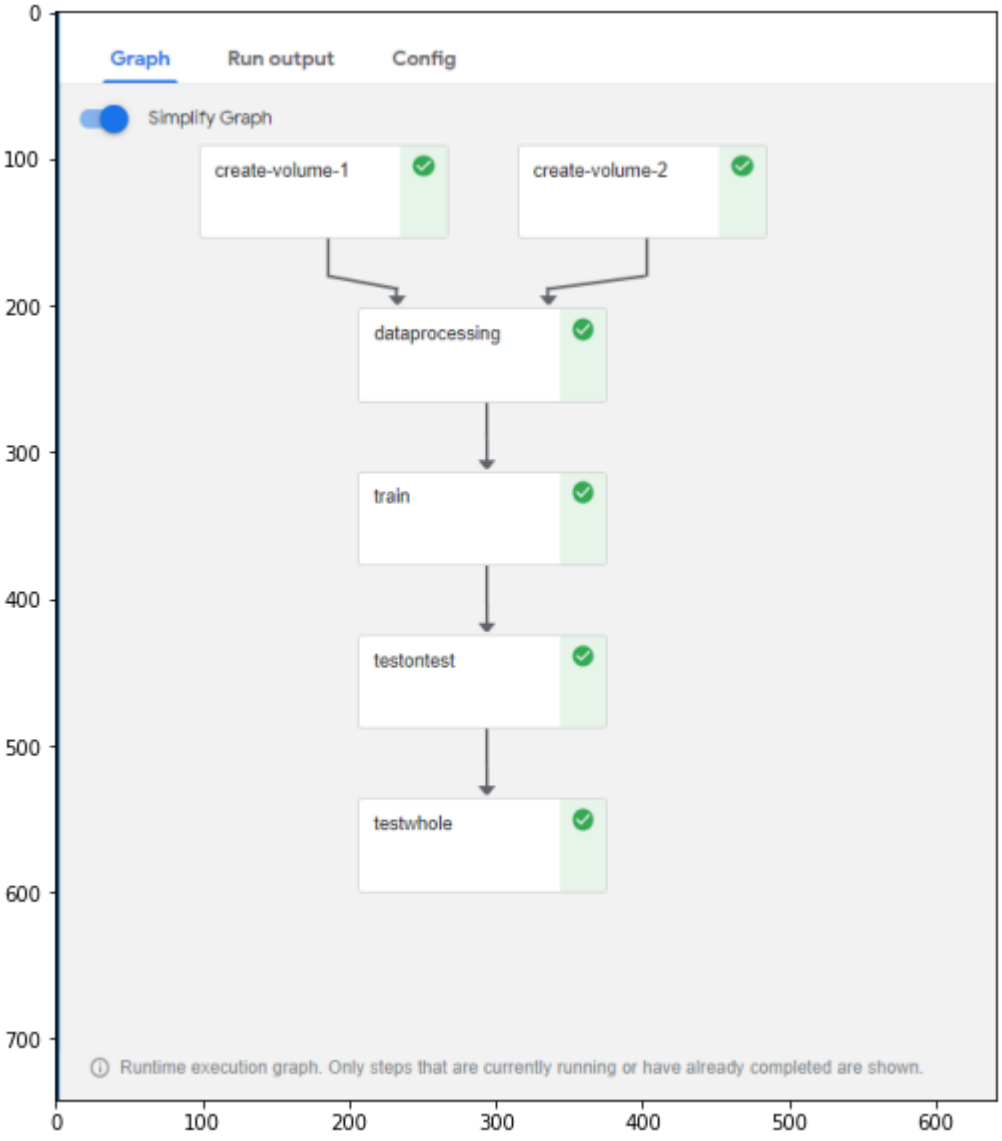


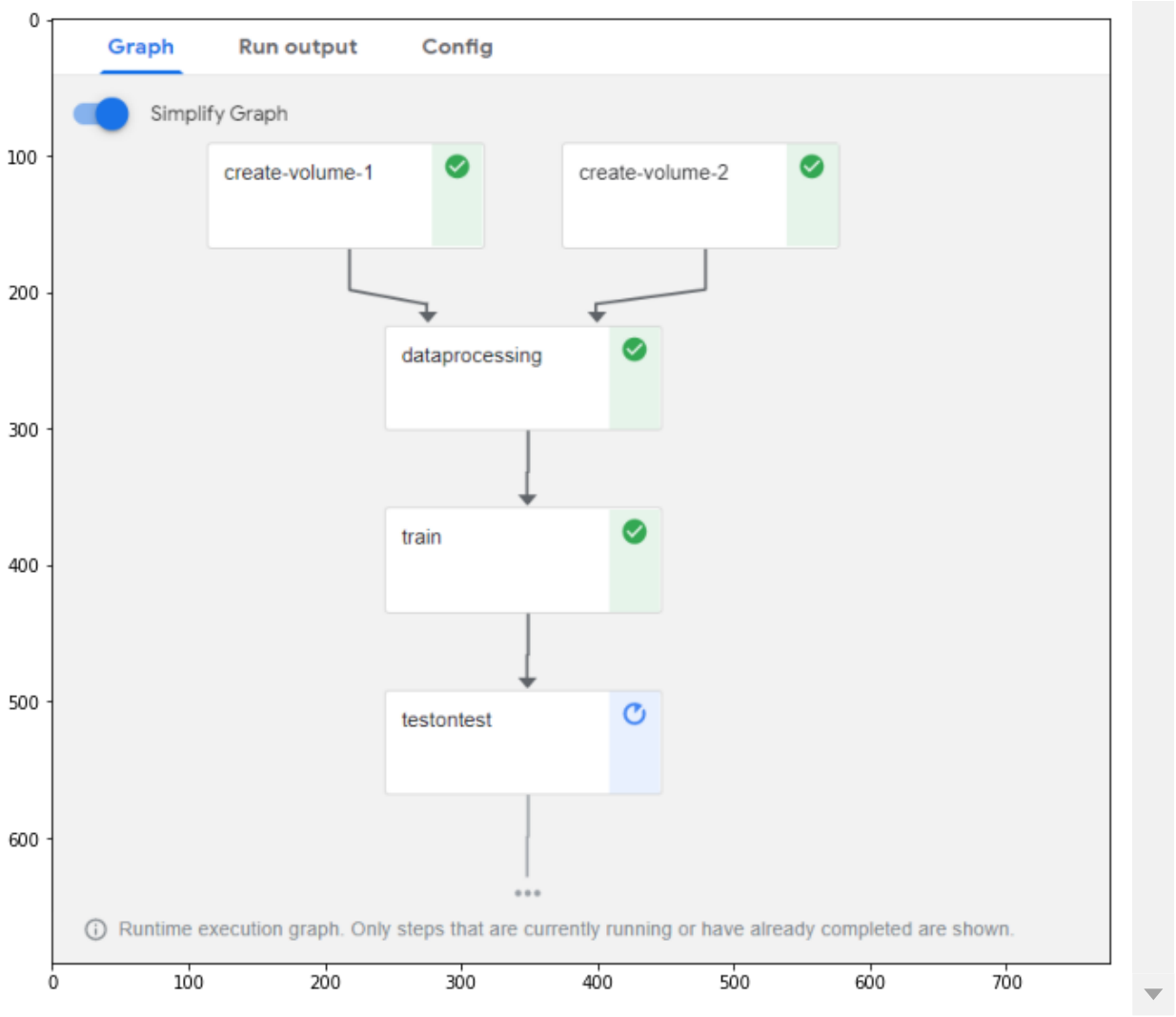


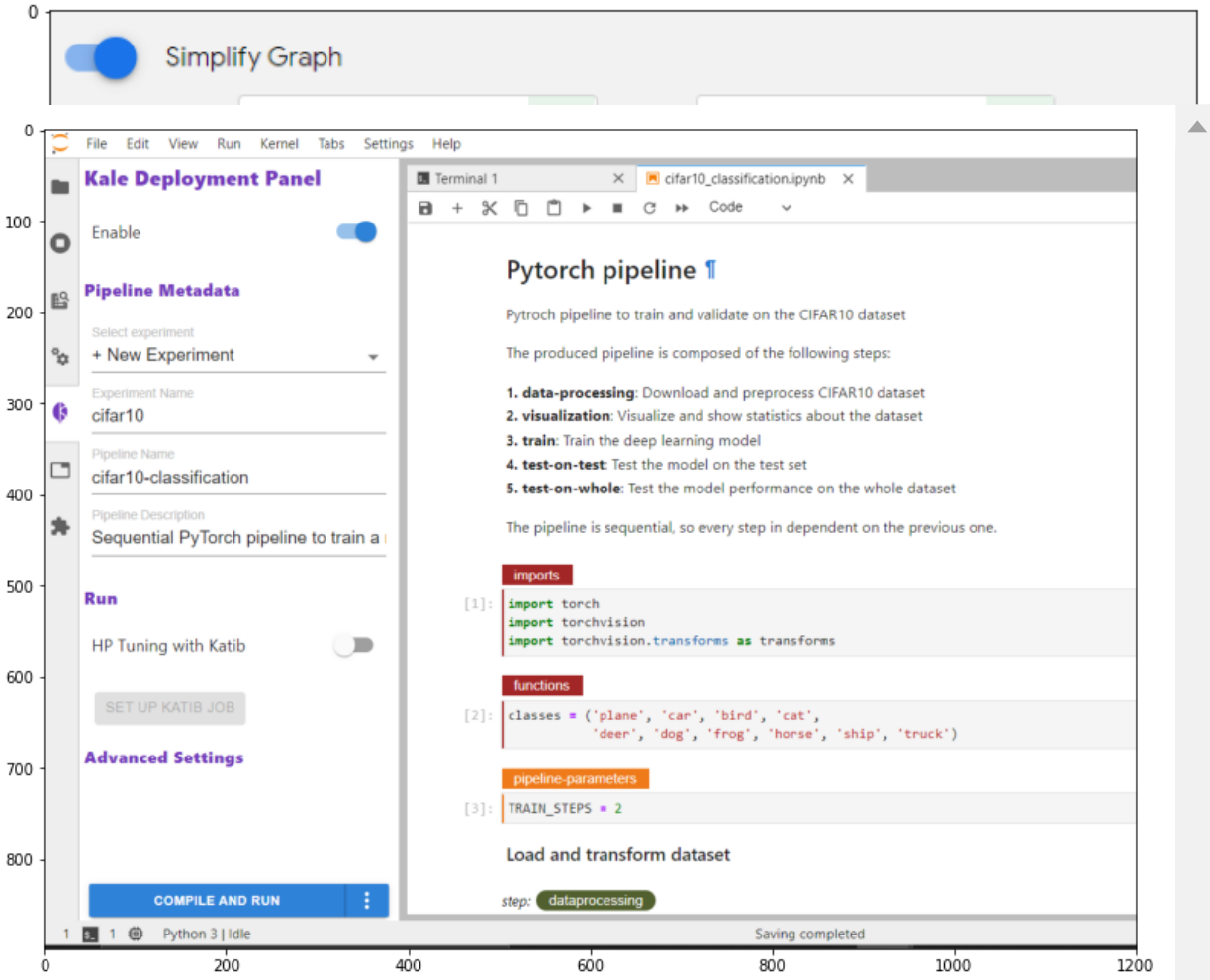
In [ ]:

```
print('(c)')
files = glob.glob('3.2.3*')
c=0
for f in files:
    if(c<4):
        fig = plt.gcf()
        fig.set_size_inches(15,10)
        img = Image.open(f)
        plt.imshow(img)
        plt.show()
    c+=1
```

(c)



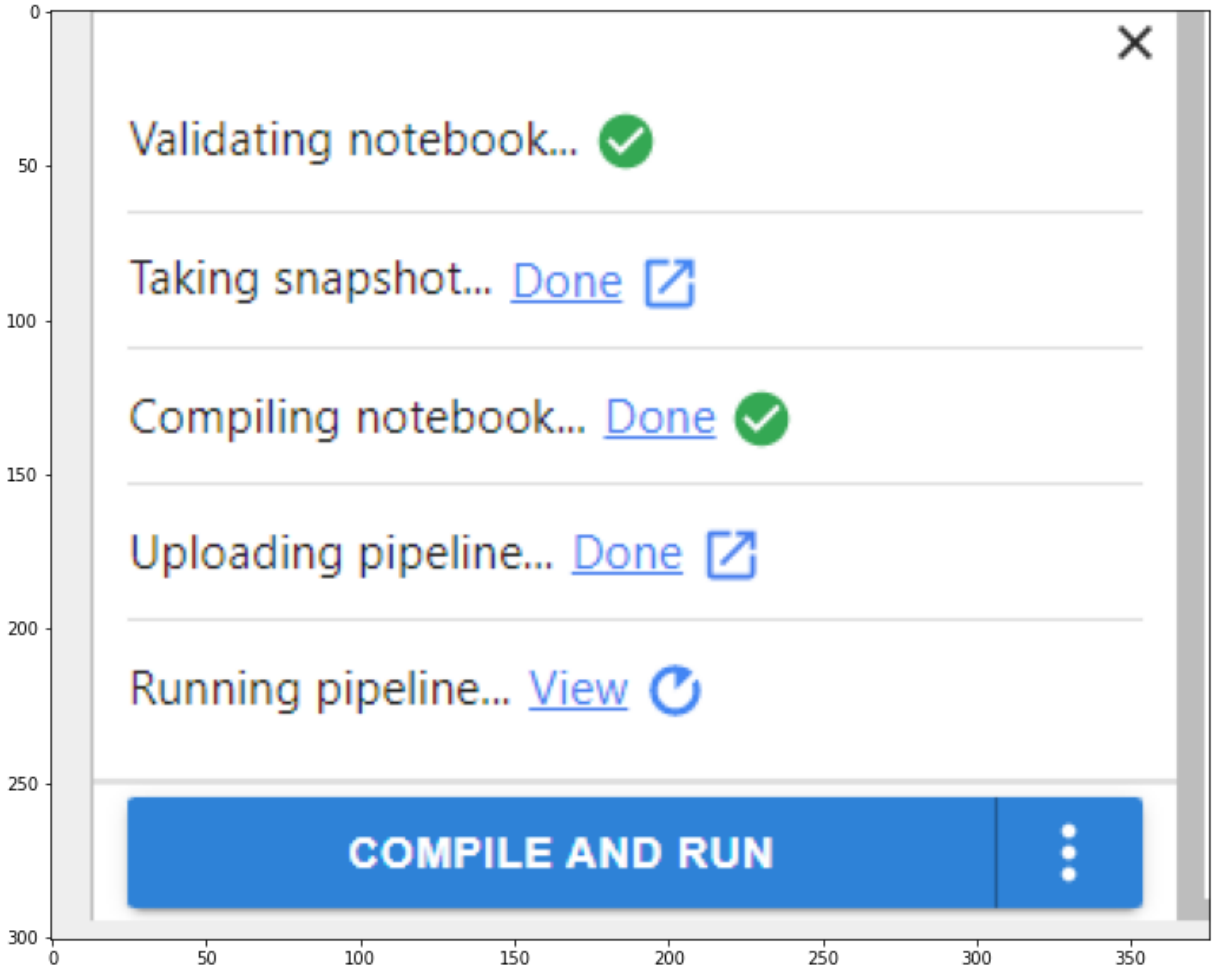

Graph
Run output
Config
Simplify Graph
create-volume-1
create-volume-2
dataprocessing
train
testontest
Runtime execution graph. Only steps that are currently running or have already completed are shown.
0
100
200
300
400
500
600
700

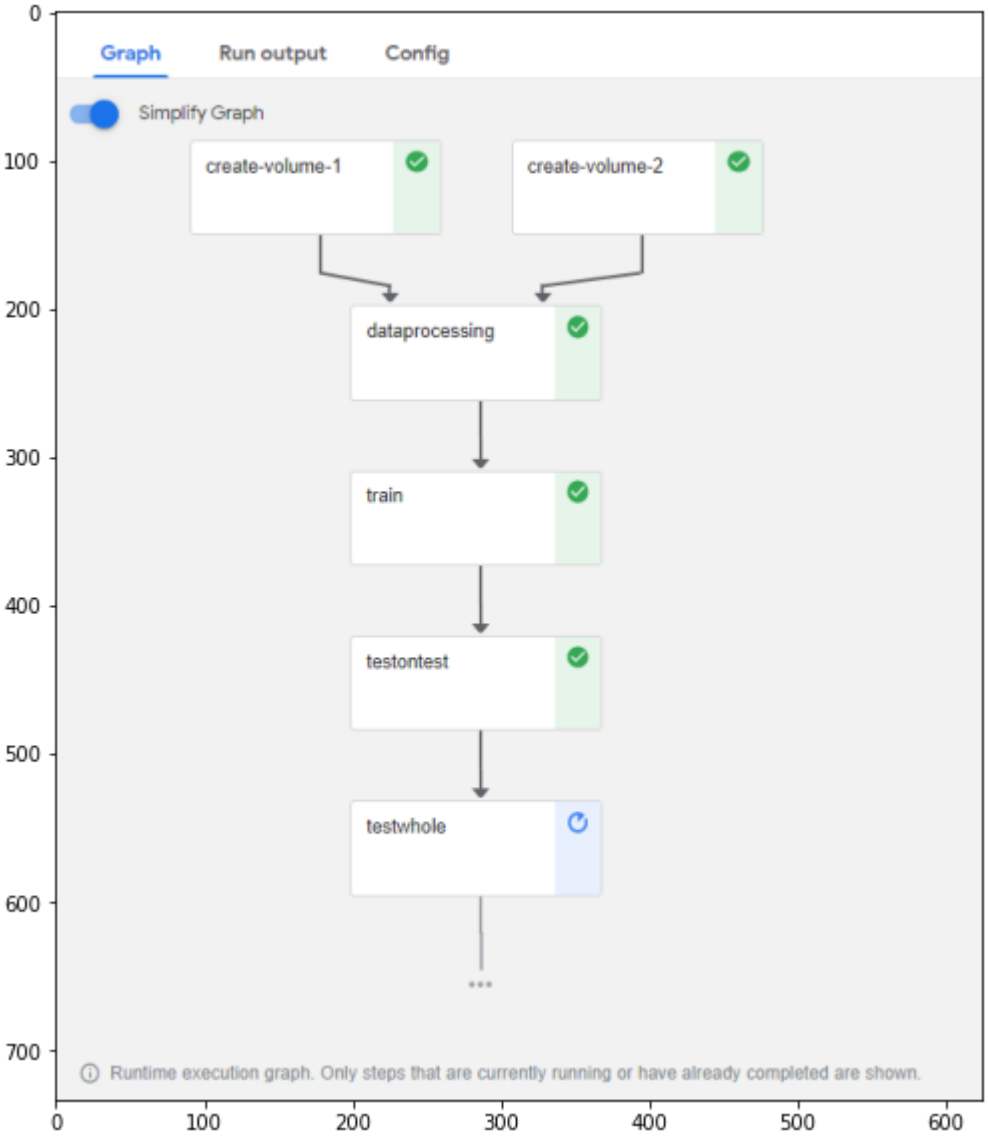

Simplify Graph
File Edit View Run Kernel Tabs Settings Help
Kale Deployment Panel
Enable
Pipeline Metadata
Select experiment
+ New Experiment
Experiment Name
cifar10
Pipeline Name
cifar10-classification
Pipeline Description
Sequential PyTorch pipeline to train a
Run
HP Tuning with Katib
SET UP KATIB JOB
Advanced Settings
COMPILE AND RUN
Terminal 1
cifar10_classification.ipynb
Code
Pytorch pipeline ¶
Pytroch pipeline to train and validate on the CIFAR10 dataset
The produced pipeline is composed of the following steps:
1. data-processing: Download and preprocess CIFAR10 dataset
2. visualization: Visualize and show statistics about the dataset
3. train: Train the deep learning model
4. test-on-test: Test the model on the test set
5. test-on-whole: Test the model performance on the whole dataset
The pipeline is sequential, so every step in dependent on the previous one.
imports
[1]: import torch
import torchvision
import torchvision.transforms as transforms
functions
[2]: classes = ('plane', 'car', 'bird', 'cat',
'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
pipeline-parameters
[3]: TRAIN_STEPS = 2
Load and transform dataset
step: dataprocessing
1 1 Python 3 | Idle
Saving completed
0 200 400 600 800 1000 1200
0 100 200 300 400 500 600 700 800

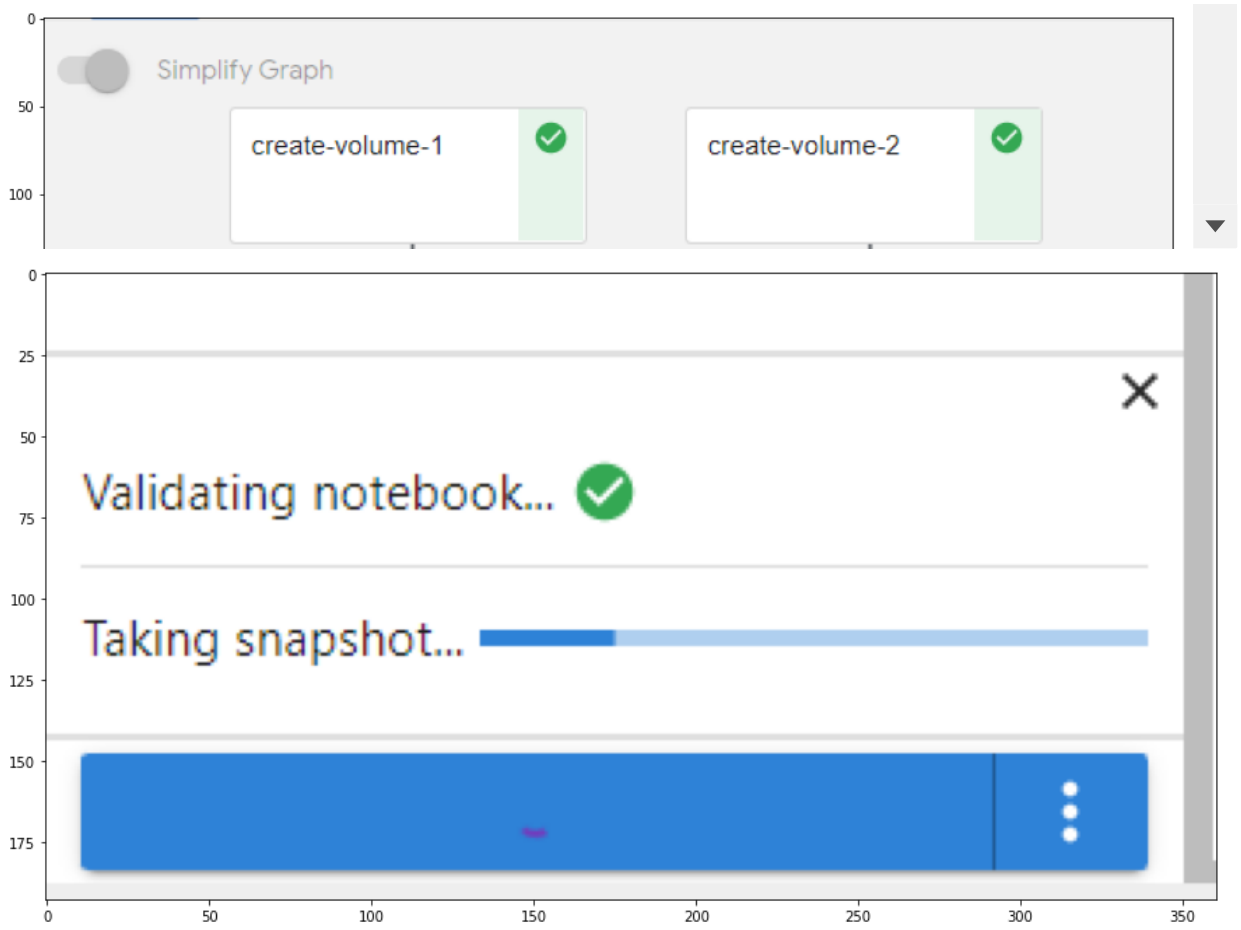
In [ ]:

```
print('(c)')
files = glob.glob('3.2.3*')
c=0
for f in files:
    if(c>=4):
        fig = plt.gcf()
        fig.set_size_inches(15,10)
        img = Image.open(f)
        plt.imshow(img)
        plt.show()
    c+=1
```

(c)



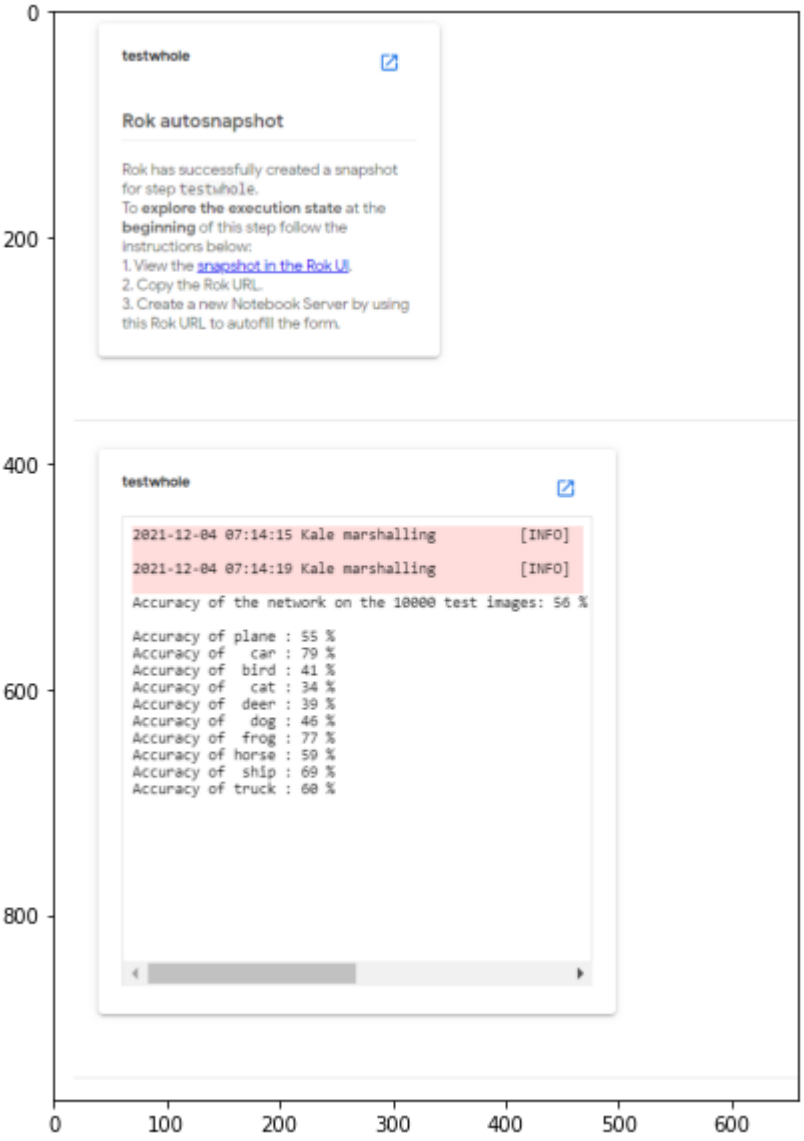

Graph
Run output
Config
Simplify Graph
create-volume-1
create-volume-2
dataprocessing
train
testontest
testwhole
Runtime execution graph. Only steps that are currently running or have already completed are shown.
0
100
200
300
400
500
600
700

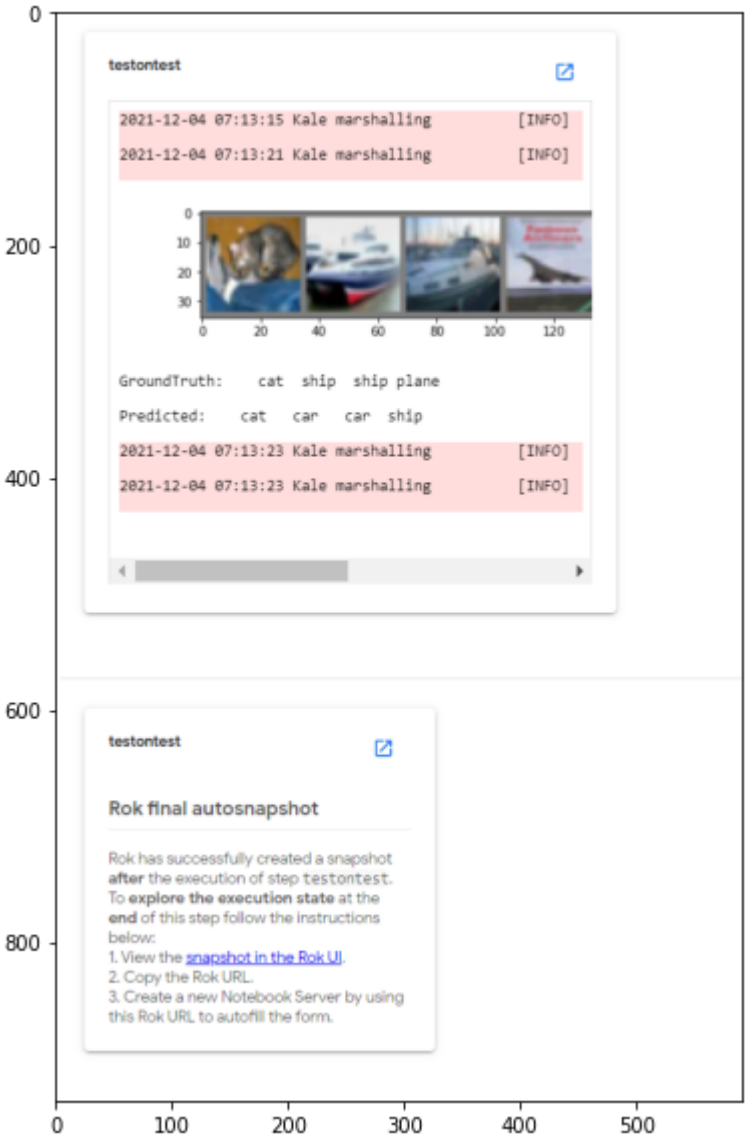

Simplify Graph
create-volume-1
create-volume-2
Validating notebook...
Taking snapshot...

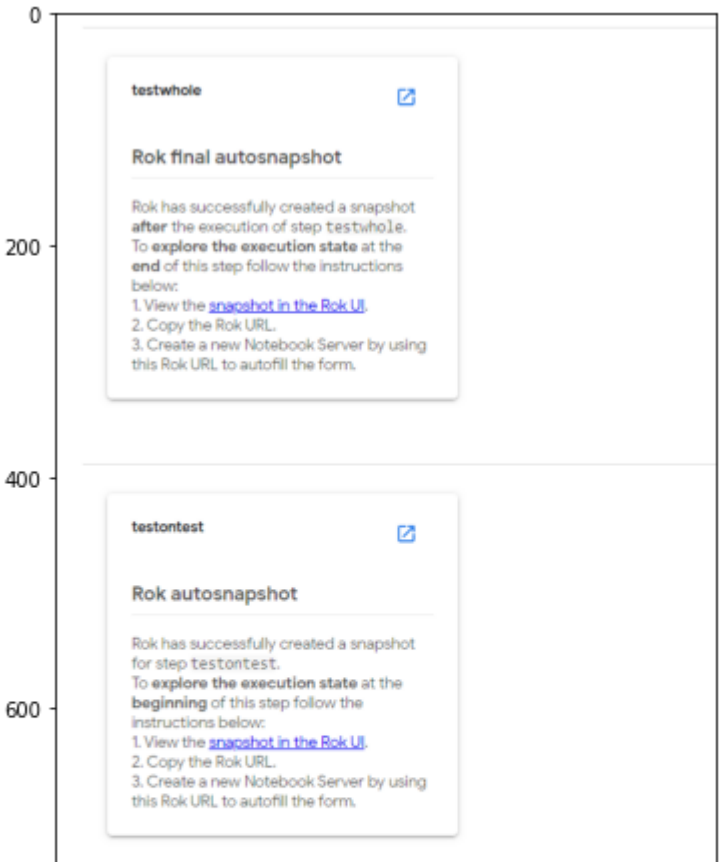
In [ ]:

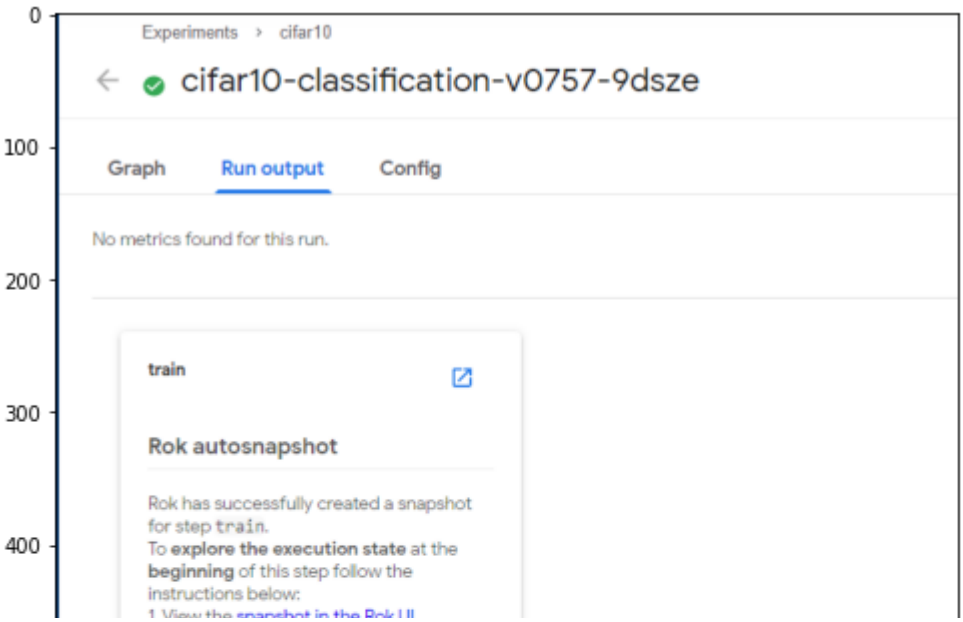
```
print('(d)')
files = glob.glob('3.2.4*')
c=0
for f in files:
    if(c<4):
        fig = plt.gcf()
        fig.set_size_inches(15, 10)
        img = Image.open(f)
        plt.imshow(img)
        plt.show()
    c+=1
```

(d)




testontest
2021-12-04 07:13:15 Kale marshalling [INFO]
2021-12-04 07:13:21 Kale marshalling [INFO]
GroundTruth: cat ship ship plane
Predicted: cat car car ship
2021-12-04 07:13:23 Kale marshalling [INFO]
2021-12-04 07:13:23 Kale marshalling [INFO]
testontest
Rok final autosnapshot
Rok has successfully created a snapshot after the execution of step testontest.
To explore the execution state at the end of this step follow the instructions below:
1. View the snapshot in the Rok UI.
2. Copy the Rok URL.
3. Create a new Notebook Server by using this Rok URL to autofill the form.

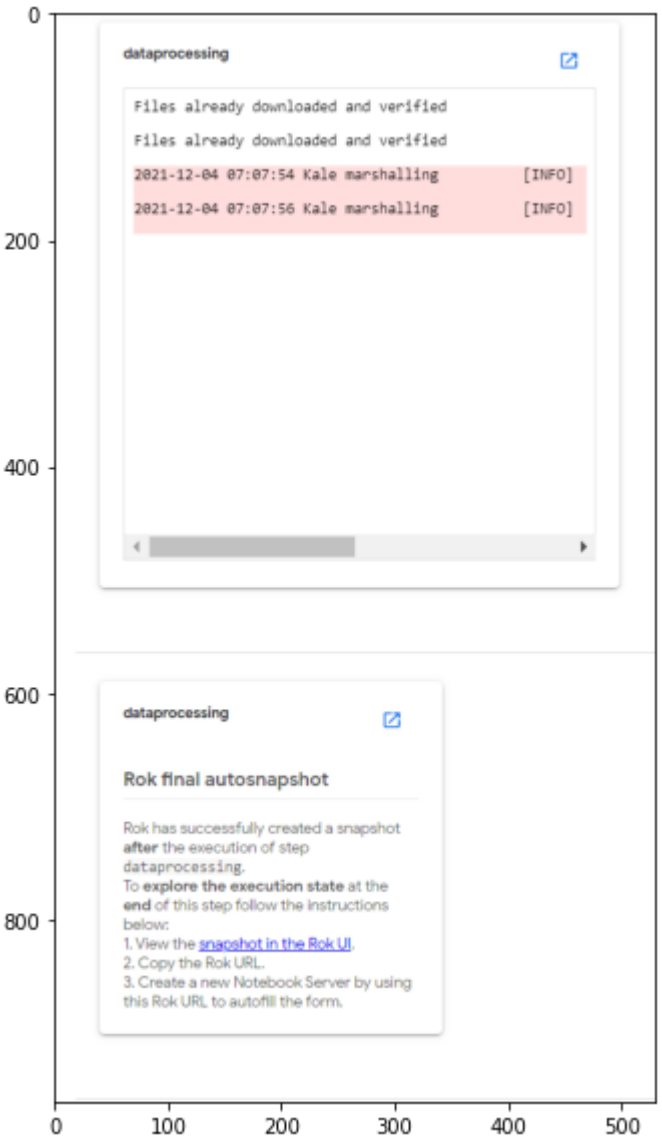

0
200
400
600
testwhole
Rok final autosnapshot
Rok has successfully created a snapshot after the execution of step testwhole.
To explore the execution state at the end of this step follow the instructions below:
1. View the snapshot in the Rok UI.
2. Copy the Rok URL.
3. Create a new Notebook Server by using this Rok URL to autofill the form.
testontest
Rok autosnapshot
Rok has successfully created a snapshot for step testontest.
To explore the execution state at the beginning of this step follow the instructions below:
1. View the snapshot in the Rok UI.
2. Copy the Rok URL.
3. Create a new Notebook Server by using this Rok URL to autofill the form.

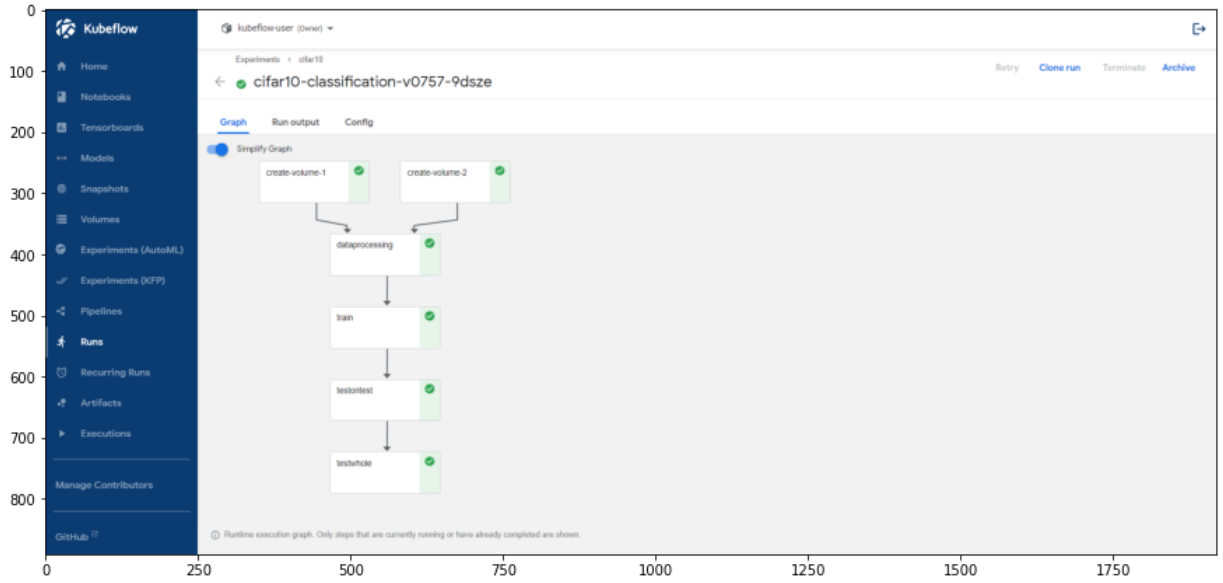

0
100
200
300
400
Experiments › cifar10
cifar10-classification-v0757-9dsze
Graph
Run output
Config
No metrics found for this run.
train
Rok autosnapshot
Rok has successfully created a snapshot for step train.
To explore the execution state at the beginning of this step follow the instructions below:

In [ ]:

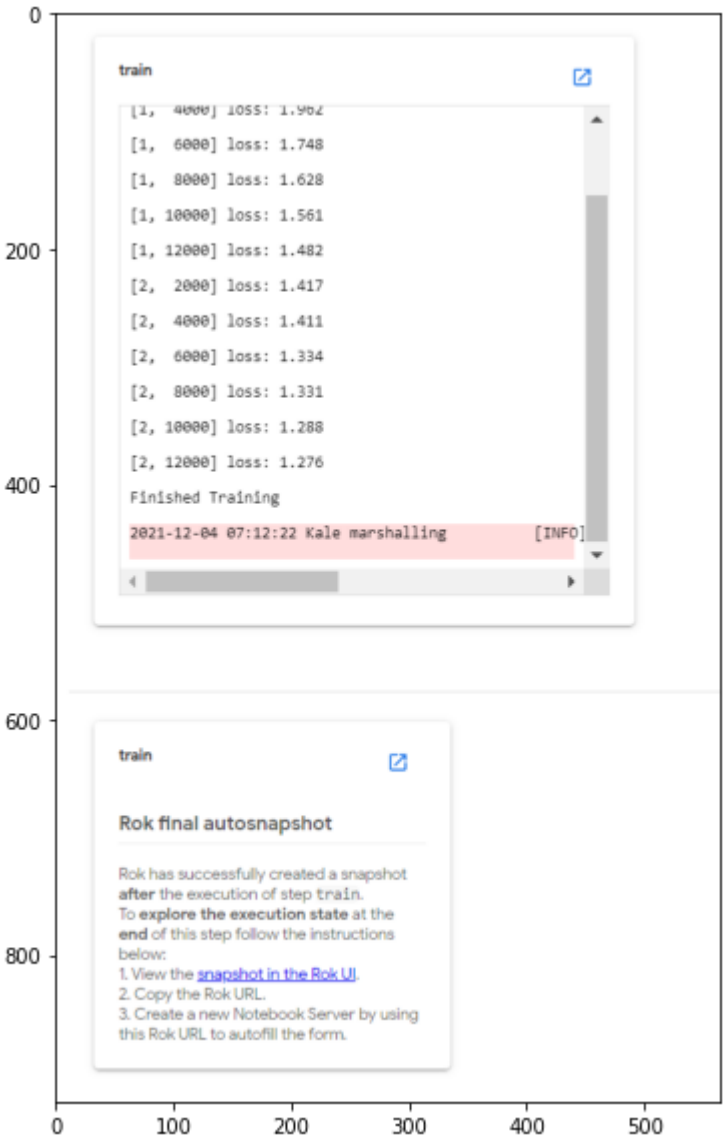
```
print('(d)')
files = glob.glob('3.2.4*')
c=0
for f in files:
    if(c>=4):
        fig = plt.gcf()
        fig.set_size_inches(15,10)
        img = Image.open(f)
        plt.imshow(img)
        plt.show()
    c+=1
```

(d)



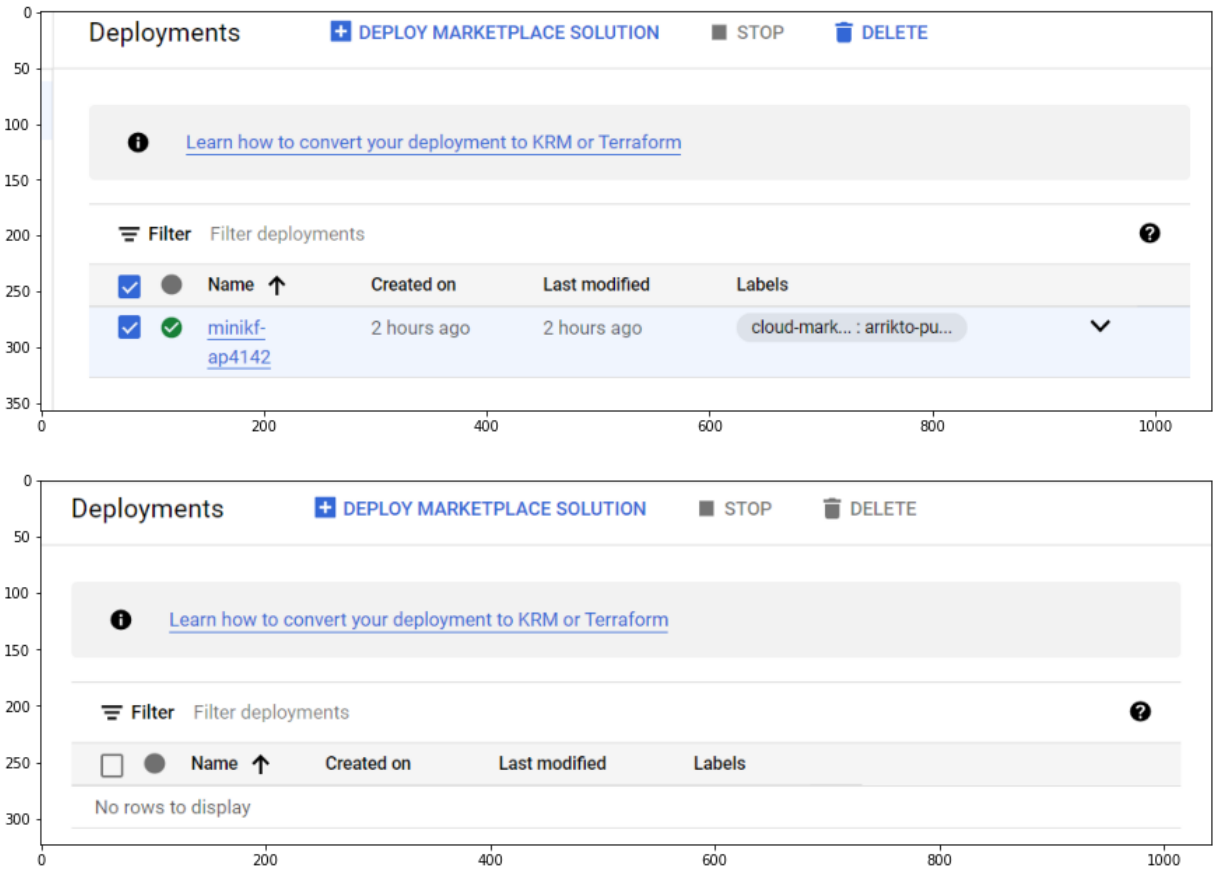

Kubeflow
kubeflow-user (Owner)
Home
Notebooks
Tensorboards
Models
Snapshots
Volumes
Experiments (AutoML)
Experiments (KFP)
Pipelines
Runs
Recurring Runs
Artifacts
Executions
Manage Contributors
GitHub
Experiments › cifar10
cifar10-classification-v0757-9dsze
Retry
Clone run
Terminate
Archive
Graph
Run output
Config
Simplify Graph
create-volume-1
create-volume-2
dataprocessing
train
testontest
testwhole
Runtime execution graph. Only steps that are currently running or have already completed are shown.
0
100
200
300
400
500
600
700
800
0
250
500
750
1000
1250
1500
1750


dataprocessing
Rok autosnapshot
Rok has successfully created a snapshot for step dataprocessing.
To explore the execution state at the beginning of this step follow the instructions below:
1. View the snapshot in the Rok UI.
2. Copy the Rok URL.
3. Create a new Notebook Server by using this Rok URL to autofill the form.
dataprocessing
Files already downloaded and verified
Files already downloaded and verified
2021-12-04 07:07:54 Kale marshalling [INFO]
2021-12-04 07:07:56 Kale marshalling [INFO]
0
200
400
600
800
0
100
200
300
400
500


0
200
400
600
800
0
100
200
300
400
500
train
[1, 4000] loss: 1.962
[1, 6000] loss: 1.748
[1, 8000] loss: 1.628
[1, 10000] loss: 1.561
[1, 12000] loss: 1.482
[2, 2000] loss: 1.417
[2, 4000] loss: 1.411
[2, 6000] loss: 1.334
[2, 8000] loss: 1.331
[2, 10000] loss: 1.288
[2, 12000] loss: 1.276
Finished Training
2021-12-04 07:12:22 Kale marshalling [INFO]
train
Rok final autosnapshot
Rok has successfully created a snapshot after the execution of step train.
To explore the execution state at the end of this step follow the instructions below:
1. View the snapshot in the Rok UI.
2. Copy the Rok URL.
3. Create a new Notebook Server by using this Rok URL to autofill the form.

In [ ]:
```
print('(e)')
files = glob.glob('3.2.5*')
c=0
for f in files[:-1]:
    fig = plt.gcf()
    fig.set_size_inches(15,10)
    img = Image.open(f)
    plt.imshow(img)
    plt.show()
```

(e)



In [ ]:
```
1
```

Problem 4

1. An episodic task is a task which has an end state or a goal after which the episode terminates that is it has at least one finite state. An example of episodic task is a game like tic-tac-toe in which an episode is finised after thhe game terminates and a new one starts.
   A continuous task is a task which does not terminate. For example prediction of stock market for trading. This task has no end to it therefore it is a continuous task.
2. In reinnforcement learning Exploration corresponds to the case of choosing an action randomly(explored or unexplored) which may not maximize the reward from the information we have at that point. Whereas, exploitation corresponds to the case of greedily choosing the action which will maximize the reward even if there are unexplored actions.

ϵ-greed method is used to make sure that the model does not always make the greed choice and explores all te possible actions in the action space instead of always choosing only one path.
A decaying ϵ will perform better than a fixed ϵ. So following a schedule is more preferrable than having a fixed ϵ. This is because in the first few iterations, we will be able to cover the action space after which choosing actions more greedily is preferred.
Having $\epsilon = 0$ will ensure that actions are picked greedily so certain explored actions will be chosen over unexplored actions which could give better results. So choosing a random action with a non-zero probability of ϵ will make sure that the unexplored actions are also selected and are explored. The higher the ϵ more the exploration. So a decaying ϵ will make sure we do enough exploration in thhe beginning and follow exploitation towards the end.

3. In Q-learning the values of each state-action pairs is stored in a table format. But in deep Q-learning we use a neural network to learn the Q function.
   Deep Q-learning uses experience relay to store the experience tuple after each time step.
   Deep Q-learning has 2 separate networks, a Q-network and a target Q-network.
   After initializing the network we first choose the action using an ϵ-greed policy. We then apply the action chosen to observe the reward froom thhe environment and the new state, and generate an experience tuple which is stored in the experience buffer D. Then we select a random minibatch of the tuples stored in D to train the Q-network. The loss function is given as $(y_j - Q(\phi_j, a_j; \theta))^2$ where $y_j = r_j$ if thhe new state is terminal and
   $y_j = r_j + max_{a'} Q(\phi_{j+1}, a'; \theta)$ otherwise.
4. In the case of Deep Q-learning, Q values are generated using a neural network. So a change in the network changes te Q value for all the actions. A target network is required to handle the moving target problem as the update equation for the Q value depends on the Q value of another action as well. It helps the network achieve more stability as updating the network at each time step may affect the stability and convergence of the model. It makes training more stable because we use a minibatch of data to update the network instead of doing it at every time step.
5. Updating the network after each iteration can cause a lot of instability in the model. Using a minibatch of experience tuples from the experience buffer D helps training to be more stable. Another advantage is that as we randomly choose a minibatch from D the tuples will be trained multiple times resulting in more efficient learning. As some of the eperience tuples come from previous versions of the Q network, there is also less overfitting.
6. Prioritized experience replay is used to improve data efficiency. In deep RL it is used in replay buffers which allows for more efficient learning by replaying important transitions more frequently. This is because not all experiences are equal and some experiences contribute more to than agent's learning than others. Generally more importance is given to the most 'surprising' experiences.
7. Similarities:
   1)Both GORILA and Ape-X architectures follow 2 parts - acting & learning.
   2)Both GORILA and Ape-X run with no high level asynchronization.
   3)In both GORILA and Ape-X acting and learning run concurrently.
   4)For both, acting & learning could be distributed across multiple workers.
   Differences:
   1)Ape-X uses a shared, centralized replay memory unlike Gorila
   2)Gorila uses uniform sampling, whereas Ape-X prioritizes the most useful data.
   3)Since priorities are shared, high priority data discovered by any actor can benefit the whole system which does not hhappen in Gorila.

In [ ]:
```
1
```