cnn_v5

October 13, 2022

[ ]:

```
%env LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/
```

```
env: LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/
```

[ ]:

```
import os
print(os.environ["LD_LIBRARY_PATH"])
```

```
:/home/nkspartan/miniconda3/envs/tf-gpu/lib/:/home/nkspartan/miniconda3/envs/tf-
gpu/lib/
```

[ ]:

```
import tensorflow as tf
import numpy as np
import pandas as pd
import os
import keras

from keras import Sequential, models, Input
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout,␣
 ↪LeakyReLU
from keras.optimizers import SGD, Adam
```

```
2022-10-13 22:32:01.785635: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2022-10-13 22:32:02.288671: E tensorflow/stream_executor/cuda/cuda_blas.cc:2981]
Unable to register cuBLAS factory: Attempting to register factory for plugin
cuBLAS when one has already been registered
2022-10-13 22:32:03.388483: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot open shared
object file: No such file or directory; LD_LIBRARY_PATH:
:/home/nkspartan/miniconda3/envs/tf-gpu/lib/:/home/nkspartan/miniconda3/envs/tf-
gpu/lib/
2022-10-13 22:32:03.388691: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
```

```
dynamic library 'libnvinfer_plugin.so.7'; dlerror: libnvinfer_plugin.so.7:
cannot open shared object file: No such file or directory; LD_LIBRARY_PATH:
:/home/nkspartan/miniconda3/envs/tf-gpu/lib/:/home/nkspartan/miniconda3/envs/tf-
gpu/lib/
2022-10-13 22:32:03.388696: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot
dlopen some TensorRT libraries. If you would like to use Nvidia GPU with
TensorRT, please make sure the missing libraries mentioned above are installed
properly.
```

[ ]:

```python
from tensorflow.python.client import device_lib

#print(device_lib.list_local_devices())
print('Default GPU Device: {}'.format(tf.test.gpu_device_name()))
```

```
Default GPU Device: /device:GPU:0
```

```
2022-10-13 22:32:06.425925: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2022-10-13 22:32:06.448950: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:32:06.496166: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:32:06.496370: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:32:07.290819: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:32:07.291450: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:32:07.291606: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:32:07.291738: I
```

```
tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device
/device:GPU:0 with 4078 MB memory:  -> device: 0, name: NVIDIA GeForce RTX 2060,
pci bus id: 0000:08:00.0, compute capability: 7.5
```

0.1 Read the csv dataset to get the values for stage and discharge of the images

[ ]:

```
df = pd.read_csv("../../dataset/2012_2019_PlatteRiverWeir_features_merged_all.
 ↪csv")
df.head()
```

[ ]:

```
   Unnamed: 0           SensorTime          CaptureTime  \
0           0  2012-06-09 13:15:00  2012-06-09T13:09:07
1           1  2012-06-09 13:15:00  2012-06-09T13:10:29
2           2  2012-06-09 13:45:00  2012-06-09T13:44:01
3           3  2012-06-09 14:45:00  2012-06-09T14:44:30
4           4  2012-06-09 15:45:00  2012-06-09T15:44:59

                                  Filename Agency  SiteNumber TimeZone  Stage  \
0  StateLineWeir_20120609_Farrell_001.jpg   USGS     6674500      MDT   2.99
1  StateLineWeir_20120609_Farrell_002.jpg   USGS     6674500      MDT   2.99
2  StateLineWeir_20120609_Farrell_003.jpg   USGS     6674500      MDT   2.96
3  StateLineWeir_20120609_Farrell_004.jpg   USGS     6674500      MDT   2.94
4  StateLineWeir_20120609_Farrell_005.jpg   USGS     6674500      MDT   2.94

   Discharge          CalcTimestamp  …  WeirPt2X  WeirPt2Y  WwRawLineMin  \
0      916.0  2020-03-11T16:58:28  …        -1        -1           0.0
1      916.0  2020-03-11T16:58:33  …        -1        -1           0.0
2      873.0  2020-03-11T16:58:40  …        -1        -1           0.0
3      846.0  2020-03-11T16:58:47  …        -1        -1           0.0
4      846.0  2020-03-11T16:58:55  …        -1        -1           0.0

   WwRawLineMax  WwRawLineMean  WwRawLineSigma  WwCurveLineMin  \
0           0.0            0.0             0.0             0.0
1           0.0            0.0             0.0             0.0
2           0.0            0.0             0.0             0.0
3           0.0            0.0             0.0             0.0
4           0.0            0.0             0.0             0.0

   WwCurveLineMax  WwCurveLineMean  WwCurveLineSigma
0             0.0              0.0               0.0
1             0.0              0.0               0.0
2             0.0              0.0               0.0
3             0.0              0.0               0.0
4             0.0              0.0               0.0

[5 rows x 60 columns]
```

[ ]:

```
df = df[["Filename", "Stage", "Discharge"]]
```

0.1.1 Scale the data

[ ]:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
```

[ ]:

```
df[["Stage", "Discharge"]] = scaler.fit_transform(df[["Stage", "Discharge"]])
df
```

[ ]:

```
                                      Filename     Stage  Discharge
0      StateLineWeir_20120609_Farrell_001.jpg  0.138117  -0.046094
1      StateLineWeir_20120609_Farrell_002.jpg  0.138117  -0.046094
2      StateLineWeir_20120609_Farrell_003.jpg  0.100875  -0.082160
3      StateLineWeir_20120609_Farrell_004.jpg  0.076046  -0.104807
4      StateLineWeir_20120609_Farrell_005.jpg  0.076046  -0.104807
…                                           …        …          …
42054  StateLineWeir_20191011_Farrell_409.jpg -0.420526  -0.450369
42055  StateLineWeir_20191011_Farrell_410.jpg -0.420526  -0.450369
42056  StateLineWeir_20191011_Farrell_411.jpg -0.420526  -0.450369
42057  StateLineWeir_20191011_Farrell_412.jpg -0.420526  -0.450369
42058  StateLineWeir_20191011_Farrell_413.jpg -0.420526  -0.450369

[42059 rows x 3 columns]
```

[ ]:

```
from joblib import dump, load
dump(scaler, 'std_scaler.joblib', compress=True)
```

[ ]:

```
['std_scaler.joblib']
```

0.2 Create the dataset pipeline

[ ]:

```
IMG_SIZE = 512
BATCH_SIZE = 32
```

[ ]:

```
from glob import glob

def make_dataset(path, batch_size, df, seed=None):
  np.random.seed(seed)

  def parse_image(filename):
    image = tf.io.read_file(filename)
    image = tf.image.decode_jpeg(image, channels=3)
    #image = tf.image.resize(image, [IMG_SIZE, IMG_SIZE])
    image = tf.cast(image, tf.float32)
    image /= 255
    return image

def configure_for_performance(ds):
    ds = ds.shuffle(buffer_size=100)
    ds = ds.batch(batch_size)
    ds = ds.repeat()
    ds = ds.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
    return ds

filenames = glob(path + '/*')

# make train, val and test splits of the dataset (70%, 10%, 20% split)
split1 = int(0.7 * len(filenames))
split2 = int(0.8 * len(filenames))

np.random.shuffle(filenames)
train_files = filenames[:split1] # up to split 1 (ex 70%)
val_files = filenames[split1:split2] # from ex. 70% to 80%
test_files = filenames[split2:] # from ex. 80% until the end

# create stage values
stage_train_values = [df[df.Filename == file.split('/')[-1]].Stage.values for␣
↪file in train_files]
stage_val_values = [df[df.Filename == file.split('/')[-1]].Stage.values for␣
↪file in val_files]
stage_test_values = [df[df.Filename == file.split('/')[-1]].Stage.values for␣
↪file in test_files]

# create discharge values
discharge_train_values = [df[df.Filename == file.split(
    '/')[-1]].Discharge.values for file in train_files]
discharge_val_values = [df[df.Filename == file.split(
    '/')[-1]].Discharge.values for file in val_files]
discharge_test_values = [df[df.Filename == file.split(
    '/')[-1]].Discharge.values for file in test_files]

# join stage and discharge values
stage_discharge_train_values = [[np.squeeze(s), np.squeeze(d)] for s, d in␣
↪zip(stage_train_values, discharge_train_values)]
stage_discharge_val_values = [[np.squeeze(s), np.squeeze(d)] for s, d in␣
↪zip(stage_val_values, discharge_val_values)]
stage_discharge_test_values = [[np.squeeze(s), np.squeeze(
    d)] for s, d in zip(stage_test_values, discharge_test_values)]

# create images dataset (train, val, test)
filenames_train_ds = tf.data.Dataset.from_tensor_slices(train_files)
```

```
    filenames_val_ds = tf.data.Dataset.from_tensor_slices(val_files)
    filenames_test_ds = tf.data.Dataset.from_tensor_slices(test_files)

    images_train_ds = filenames_train_ds.map(parse_image, num_parallel_calls=5)
    images_val_ds = filenames_val_ds.map(parse_image, num_parallel_calls=5)
    images_test_ds = filenames_test_ds.map(parse_image, num_parallel_calls=5)

    # create stage and discharge dataset (train, val, test)
    stage_discharge_train_ds = tf.data.Dataset.
↪from_tensor_slices(stage_discharge_train_values)
    stage_discharge_val_ds = tf.data.Dataset.
↪from_tensor_slices(stage_discharge_val_values)
    stage_discharge_test_ds = tf.data.Dataset.from_tensor_slices(
        stage_discharge_test_values)

    # create tensorflow dataset of images and values (train, val, test)
    train_ds = tf.data.Dataset.zip((images_train_ds, stage_discharge_train_ds))
    train_ds = configure_for_performance(train_ds)
    val_ds = tf.data.Dataset.zip((images_val_ds, stage_discharge_val_ds))
    val_ds = configure_for_performance(val_ds)
    test_ds = tf.data.Dataset.zip((images_test_ds, stage_discharge_test_ds))
    test_ds = configure_for_performance(test_ds)

    return train_ds, len(train_files), val_ds, len(val_files), test_ds,␣
↪len(test_files)
```

[ ]:

```
path = "../../dataset/images_tmp"

train_ds, train_size, val_ds, val_size, test_ds, test_size = make_dataset(path,␣
↪BATCH_SIZE, df, 0)
```

```
2022-10-13 22:33:36.393434: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:33:36.393646: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:33:36.393786: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:33:36.394135: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
```

```
2022-10-13 22:33:36.394284: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:33:36.394423: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:33:36.394685: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:33:36.394835: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:980] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-10-13 22:33:36.394952: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 4078 MB memory:  -> device: 0,
name: NVIDIA GeForce RTX 2060, pci bus id: 0000:08:00.0, compute capability: 7.5
```

[ ]:

```python
input_shape = 0
output_shape = 0

for image, stage_discharge in train_ds.take(1):
    print(image.numpy().shape)
    print(stage_discharge.numpy().shape)

    input_shape = image.numpy().shape[1:]
    output_shape = stage_discharge.numpy().shape[1:]
```

```
(32, 512, 512, 3)
(32, 2)
```

[ ]:

```python
print(input_shape)
print(output_shape)
```

```
(512, 512, 3)
(2,)
```

0.3 Create model

[ ]:

```python
def create_model(input_shape, output_shape):
    model = Sequential()

    model.add(Input(shape=input_shape))
    model.add(Conv2D(64, kernel_size=(4, 4), strides=(2, 2), padding='same',␣
↪activation=LeakyReLU()))
    model.add(MaxPooling2D(pool_size=(4, 4)))
    model.add(Conv2D(64, kernel_size=(4, 4), activation=LeakyReLU(),␣
↪padding='same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(3, 3)))
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))


    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(32, activation='sigmoid'))
    model.add(Dense(output_shape, activation='linear')) # linear regression␣
↪output layer

    return model
```

[ ]:
```
model = create_model(input_shape, output_shape[0])
```

[ ]:
```
model.summary()
```

```
Model: "sequential_6"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_24 (Conv2D)          (None, 256, 256, 64)      3136

 max_pooling2d_24 (MaxPoolin  (None, 64, 64, 64)       0
 g2D)

 conv2d_25 (Conv2D)          (None, 64, 64, 64)        65600

 max_pooling2d_25 (MaxPoolin  (None, 32, 32, 64)       0
 g2D)

 conv2d_26 (Conv2D)          (None, 32, 32, 32)        18464

 max_pooling2d_26 (MaxPoolin  (None, 10, 10, 32)       0
 g2D)

 conv2d_27 (Conv2D)          (None, 8, 8, 32)          9248
 max_pooling2d_27 (MaxPoolin  (None, 4, 4, 32)         0
 g2D)

 flatten_6 (Flatten)         (None, 512)               0

 dense_30 (Dense)            (None, 128)               65664

 dense_31 (Dense)            (None, 32)                4128

 dense_32 (Dense)            (None, 32)                1056

 dense_33 (Dense)            (None, 32)                1056

 dense_34 (Dense)            (None, 2)                 66

=================================================================
Total params: 168,418
Trainable params: 168,418
Non-trainable params: 0
_________________________________________________________________
```

[ ]:

```
def compile_model(loss_func, optimizer, metrics=["accuracy"]):
    model.compile(loss=loss_func, optimizer=optimizer, metrics=metrics)
```

[ ]:

```
sgd = SGD(learning_rate=0.01, decay=1e-4, momentum=0.9, nesterov=True)
adam = Adam(learning_rate=1e-3, decay=1e-3 / 100)

compile_model('mse', adam,  [
              'mse', tf.keras.metrics.RootMeanSquaredError(name='rmse'), 'mae',␣
 ↪'mape'])
```

[ ]:

```
def fit_model(training_values, validation_values=None, batch_size=32,␣
 ↪epochs=10, steps=32, val_steps=32, callbacks=[]):
    return model.fit(training_values, validation_data=validation_values,␣
 ↪batch_size=batch_size, epochs=epochs, steps_per_epoch=steps,␣
 ↪validation_steps=val_steps, callbacks=callbacks)
```

[ ]:

```
import datetime

date_actual = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
log_dir = "logs/fit/" + date_actual
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)

checkpoint_callback = tf.keras.callbacks.
 ↪ModelCheckpoint(filepath=f"model_weights/{date_actual}_cnn_best_weights.
 ↪hdf5",
                               monitor='val_mse',
                               verbose=1,
                               save_best_only=True)
```

[ ]:
```
# batch_size = 0 because we already have batch size in tf dataset
history = fit_model(train_ds, val_ds, batch_size=0, epochs=20, steps=np.
 ↪ceil(train_size / BATCH_SIZE), val_steps=np.ceil(val_size / BATCH_SIZE),␣
 ↪callbacks=[tensorboard_callback, checkpoint_callback])
```

```
Epoch 1/20
347/348 [============================>.] - ETA: 0s - loss: 0.3567 - mse: 0.3567
- rmse: 0.5972 - mae: 0.3288 - mape: 162.6058
Epoch 1: val_mse improved from inf to 0.10324, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 39s 111ms/step - loss: 0.3565 - mse:
0.3565 - rmse: 0.5971 - mae: 0.3287 - mape: 162.5373 - val_loss: 0.1032 -
val_mse: 0.1032 - val_rmse: 0.3213 - val_mae: 0.1697 - val_mape: 87.4317
Epoch 2/20
347/348 [============================>.] - ETA: 0s - loss: 0.0707 - mse: 0.0707
- rmse: 0.2659 - mae: 0.1487 - mape: 55.4671
Epoch 2: val_mse improved from 0.10324 to 0.03743, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 34s 98ms/step - loss: 0.0707 - mse:
0.0707 - rmse: 0.2658 - mae: 0.1487 - mape: 55.4547 - val_loss: 0.0374 -
val_mse: 0.0374 - val_rmse: 0.1935 - val_mae: 0.1163 - val_mape: 57.8829
Epoch 3/20
347/348 [============================>.] - ETA: 0s - loss: 0.0309 - mse: 0.0309
- rmse: 0.1759 - mae: 0.1046 - mape: 39.8158
Epoch 3: val_mse improved from 0.03743 to 0.02378, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 33s 95ms/step - loss: 0.0309 - mse:
0.0309 - rmse: 0.1758 - mae: 0.1046 - mape: 39.8099 - val_loss: 0.0238 -
val_mse: 0.0238 - val_rmse: 0.1542 - val_mae: 0.1036 - val_mape: 49.5065
Epoch 4/20
347/348 [============================>.] - ETA: 0s - loss: 0.0204 - mse: 0.0204
- rmse: 0.1430 - mae: 0.0896 - mape: 44.9386
Epoch 4: val_mse improved from 0.02378 to 0.01643, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 34s 99ms/step - loss: 0.0204 - mse:
0.0204 - rmse: 0.1430 - mae: 0.0896 - mape: 44.9378 - val_loss: 0.0164 -
val_mse: 0.0164 - val_rmse: 0.1282 - val_mae: 0.0843 - val_mape: 40.7541
Epoch 5/20
347/348 [============================>.] - ETA: 0s - loss: 0.0174 - mse: 0.0174
- rmse: 0.1321 - mae: 0.0827 - mape: 52.0188
Epoch 5: val_mse improved from 0.01643 to 0.01280, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 35s 101ms/step - loss: 0.0174 - mse:
```

```
0.0174 - rmse: 0.1320 - mae: 0.0827 - mape: 51.9978 - val_loss: 0.0128 -
val_mse: 0.0128 - val_rmse: 0.1131 - val_mae: 0.0729 - val_mape: 40.1540
Epoch 6/20
347/348 [============================>.] - ETA: 0s - loss: 0.0149 - mse: 0.0149
- rmse: 0.1219 - mae: 0.0759 - mape: 40.7168
Epoch 6: val_mse improved from 0.01280 to 0.01113, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 36s 104ms/step - loss: 0.0148 - mse:
0.0148 - rmse: 0.1219 - mae: 0.0759 - mape: 40.7016 - val_loss: 0.0111 -
val_mse: 0.0111 - val_rmse: 0.1055 - val_mae: 0.0702 - val_mape: 33.6337
Epoch 7/20
347/348 [============================>.] - ETA: 0s - loss: 0.0118 - mse: 0.0118
- rmse: 0.1084 - mae: 0.0680 - mape: 32.6481
Epoch 7: val_mse improved from 0.01113 to 0.01086, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 36s 104ms/step - loss: 0.0118 - mse:
0.0118 - rmse: 0.1084 - mae: 0.0680 - mape: 32.6601 - val_loss: 0.0109 -
val_mse: 0.0109 - val_rmse: 0.1042 - val_mae: 0.0691 - val_mape: 33.7990
Epoch 8/20
347/348 [============================>.] - ETA: 0s - loss: 0.0109 - mse: 0.0109
- rmse: 0.1044 - mae: 0.0654 - mape: 36.7081
Epoch 8: val_mse did not improve from 0.01086
348/348 [==============================] - 34s 97ms/step - loss: 0.0109 - mse:
0.0109 - rmse: 0.1044 - mae: 0.0654 - mape: 36.6934 - val_loss: 0.0109 -
val_mse: 0.0109 - val_rmse: 0.1043 - val_mae: 0.0693 - val_mape: 31.4723
Epoch 9/20
347/348 [============================>.] - ETA: 0s - loss: 0.0097 - mse: 0.0097
- rmse: 0.0984 - mae: 0.0605 - mape: 31.5169
Epoch 9: val_mse did not improve from 0.01086
348/348 [==============================] - 37s 105ms/step - loss: 0.0097 - mse:
0.0097 - rmse: 0.0984 - mae: 0.0605 - mape: 31.5105 - val_loss: 0.0116 -
val_mse: 0.0116 - val_rmse: 0.1077 - val_mae: 0.0677 - val_mape: 34.1491
Epoch 10/20
347/348 [============================>.] - ETA: 0s - loss: 0.0094 - mse: 0.0094
- rmse: 0.0970 - mae: 0.0597 - mape: 29.1988
Epoch 10: val_mse did not improve from 0.01086
348/348 [==============================] - 38s 108ms/step - loss: 0.0094 - mse:
0.0094 - rmse: 0.0970 - mae: 0.0597 - mape: 29.1928 - val_loss: 0.0143 -
val_mse: 0.0143 - val_rmse: 0.1195 - val_mae: 0.0790 - val_mape: 36.7539
Epoch 11/20
347/348 [============================>.] - ETA: 0s - loss: 0.0089 - mse: 0.0089
- rmse: 0.0941 - mae: 0.0577 - mape: 28.1576
Epoch 11: val_mse improved from 0.01086 to 0.00949, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 39s 112ms/step - loss: 0.0089 - mse:
0.0089 - rmse: 0.0941 - mae: 0.0577 - mape: 28.1647 - val_loss: 0.0095 -
val_mse: 0.0095 - val_rmse: 0.0974 - val_mae: 0.0650 - val_mape: 30.9810
Epoch 12/20
```

```
347/348 [============================>.] - ETA: 0s - loss: 0.0077 - mse: 0.0077
- rmse: 0.0877 - mae: 0.0540 - mape: 27.0972
Epoch 12: val_mse did not improve from 0.00949
348/348 [==============================] - 39s 113ms/step - loss: 0.0077 - mse:
0.0077 - rmse: 0.0877 - mae: 0.0540 - mape: 27.0872 - val_loss: 0.0130 -
val_mse: 0.0130 - val_rmse: 0.1141 - val_mae: 0.0751 - val_mape: 30.6593
Epoch 13/20
347/348 [============================>.] - ETA: 0s - loss: 0.0082 - mse: 0.0082
- rmse: 0.0907 - mae: 0.0557 - mape: 35.7169
Epoch 13: val_mse did not improve from 0.00949
348/348 [==============================] - 39s 111ms/step - loss: 0.0082 - mse:
0.0082 - rmse: 0.0907 - mae: 0.0557 - mape: 35.7048 - val_loss: 0.0128 -
val_mse: 0.0128 - val_rmse: 0.1130 - val_mae: 0.0713 - val_mape: 37.3328
Epoch 14/20
347/348 [============================>.] - ETA: 0s - loss: 0.0073 - mse: 0.0073
- rmse: 0.0855 - mae: 0.0525 - mape: 31.8779
Epoch 14: val_mse improved from 0.00949 to 0.00895, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 41s 117ms/step - loss: 0.0073 - mse:
0.0073 - rmse: 0.0855 - mae: 0.0524 - mape: 31.8660 - val_loss: 0.0090 -
val_mse: 0.0090 - val_rmse: 0.0946 - val_mae: 0.0684 - val_mape: 25.2155
Epoch 15/20
347/348 [============================>.] - ETA: 0s - loss: 0.0110 - mse: 0.0110
- rmse: 0.1049 - mae: 0.0642 - mape: 47.2699
Epoch 15: val_mse improved from 0.00895 to 0.00804, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 38s 110ms/step - loss: 0.0110 - mse:
0.0110 - rmse: 0.1049 - mae: 0.0642 - mape: 47.2535 - val_loss: 0.0080 -
val_mse: 0.0080 - val_rmse: 0.0897 - val_mae: 0.0605 - val_mape: 28.6601
Epoch 16/20
347/348 [============================>.] - ETA: 0s - loss: 0.0062 - mse: 0.0062
- rmse: 0.0786 - mae: 0.0471 - mape: 22.1928
Epoch 16: val_mse improved from 0.00804 to 0.00688, saving model to
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 39s 112ms/step - loss: 0.0062 - mse:
0.0062 - rmse: 0.0786 - mae: 0.0471 - mape: 22.1864 - val_loss: 0.0069 -
val_mse: 0.0069 - val_rmse: 0.0830 - val_mae: 0.0567 - val_mape: 28.5051
Epoch 17/20
347/348 [============================>.] - ETA: 0s - loss: 0.0073 - mse: 0.0073
- rmse: 0.0852 - mae: 0.0521 - mape: 24.7771
Epoch 17: val_mse did not improve from 0.00688
348/348 [==============================] - 39s 112ms/step - loss: 0.0073 - mse:
0.0073 - rmse: 0.0852 - mae: 0.0521 - mape: 24.7728 - val_loss: 0.0070 -
val_mse: 0.0070 - val_rmse: 0.0835 - val_mae: 0.0518 - val_mape: 32.8219
Epoch 18/20
347/348 [============================>.] - ETA: 0s - loss: 0.0059 - mse: 0.0059
- rmse: 0.0765 - mae: 0.0465 - mape: 22.7724
Epoch 18: val_mse improved from 0.00688 to 0.00521, saving model to
```

```
model_weights/20221013-221702_cnn_best_weights.hdf5
348/348 [==============================] - 39s 112ms/step - loss: 0.0059 - mse:
0.0059 - rmse: 0.0765 - mae: 0.0465 - mape: 22.7665 - val_loss: 0.0052 -
val_mse: 0.0052 - val_rmse: 0.0722 - val_mae: 0.0467 - val_mape: 24.6541
Epoch 19/20
347/348 [============================>.] - ETA: 0s - loss: 0.0053 - mse: 0.0053
- rmse: 0.0728 - mae: 0.0447 - mape: 24.0311
Epoch 19: val_mse did not improve from 0.00521
348/348 [==============================] - 38s 109ms/step - loss: 0.0053 - mse:
0.0053 - rmse: 0.0728 - mae: 0.0447 - mape: 24.0230 - val_loss: 0.0070 -
val_mse: 0.0070 - val_rmse: 0.0834 - val_mae: 0.0549 - val_mape: 28.3918
Epoch 20/20
347/348 [============================>.] - ETA: 0s - loss: 0.0053 - mse: 0.0053
- rmse: 0.0726 - mae: 0.0437 - mape: 21.7055
Epoch 20: val_mse did not improve from 0.00521
348/348 [==============================] - 35s 100ms/step - loss: 0.0053 - mse:
0.0053 - rmse: 0.0726 - mae: 0.0437 - mape: 21.6978 - val_loss: 0.0063 -
val_mse: 0.0063 - val_rmse: 0.0796 - val_mae: 0.0527 - val_mape: 30.9943
```

0.4 Evaluate model

[ ]:

```python
best_model = models.load_model(f'model_weights/20221013-221702_cnn_best_weights.
 ↪hdf5')
```

[ ]:

```python
def evaluate_model(model, test_values, steps):
  score = model.evaluate(test_values, steps=steps)
  return score
```

[ ]:

```python
test_loss, test_mse, test_rmse, test_mae, test_mape =␣
 ↪evaluate_model(best_model, test_ds, steps=np.ceil(test_size / BATCH_SIZE))
```

```
2022-10-13 22:34:16.889444: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384]
Loaded cuDNN version 8100
2022-10-13 22:34:18.077149: I
tensorflow/core/platform/default/subprocess.cc:304] Start cannot spawn child
process: No such file or directory
2022-10-13 22:34:18.077988: I
tensorflow/core/platform/default/subprocess.cc:304] Start cannot spawn child
process: No such file or directory
2022-10-13 22:34:18.078006: W tensorflow/stream_executor/gpu/asm_compiler.cc:80]
Couldn't get ptxas version string: INTERNAL: Couldn't invoke ptxas --version
2022-10-13 22:34:18.078825: I
tensorflow/core/platform/default/subprocess.cc:304] Start cannot spawn child
process: No such file or directory
2022-10-13 22:34:18.078873: W
tensorflow/stream_executor/gpu/redzone_allocator.cc:314] INTERNAL: Failed to
launch ptxas
Relying on driver to perform ptx compilation.
```

```
Modify $PATH to customize ptxas location.
This message will be only logged once.
```

```
100/100 [==============================] - 9s 59ms/step - loss: 0.0052 - mse:
0.0052 - rmse: 0.0719 - mae: 0.0469 - mape: 15.1341
```

[ ]:

```python
predictions = best_model.predict(test_ds, steps=np.ceil(test_size / BATCH_SIZE))
```

```
100/100 [==============================] - 6s 60ms/step
```

[ ]:

```python
#small_test_ds = next(iter(test_ds))
```

[ ]:

```python
for image, stage_discharge in test_ds.take(1):
        predictions = best_model.predict(x=image)

        stage_discharge_test_values = stage_discharge[:2].numpy()
        predictions_values = predictions[:2]

        diff = predictions_values.flatten() - stage_discharge_test_values.
 ↪flatten()
        percentDiff = (diff / stage_discharge_test_values.flatten()) * 100
        absPercentDiff = np.abs(percentDiff)
        # compute the mean and standard deviation of the absolute percentage
        # difference
        mean = np.mean(absPercentDiff)
        std = np.std(absPercentDiff)
        # finally, show some statistics on our model
        print(mean)
        print(std)

        for i in range(len(stage_discharge_test_values)):
                print(f"pred stage: {scaler.
 ↪inverse_transform(predictions_values)[i][0]}, actual stage: {scaler.
 ↪inverse_transform(stage_discharge_test_values)[i][0]}")
                print(f"pred discharge: {scaler.
 ↪inverse_transform(predictions_values)[i][1]}, actual discharge: {scaler.
 ↪inverse_transform(stage_discharge_test_values)[i][1]}")
```

```
1/1 [==============================] - 0s 64ms/step
3.6887279750450386
3.6212939608575954
pred stage: 2.1561872959136963, actual stage: 2.15
pred discharge: 154.75941467285156, actual discharge: 155.0
pred stage: 4.178528308868408, actual stage: 4.12
pred discharge: 2891.14892578125, actual discharge: 2730.0
```

[ ]:

```python

```

0.5 Visualize layers

[ ]:

```
layer_outputs = [layer.output for layer in best_model.layers[:12]]
# Extracts the outputs of the top 12 layers
activation_model = models.Model(inputs=best_model.input, outputs=layer_outputs) # Creates a model that will return these outputs, given the model input
```

[ ]:

```
activations = activation_model.predict(test_ds.take(1))
```

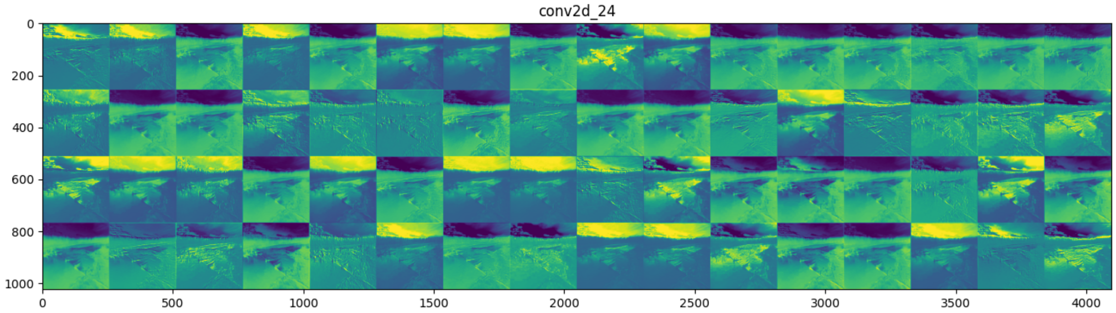
```
1/1 [==============================] - 0s 202ms/step
```

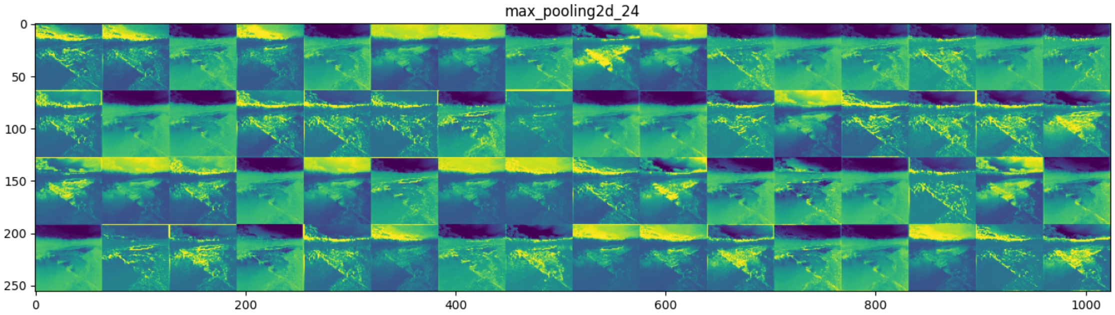
[ ]:

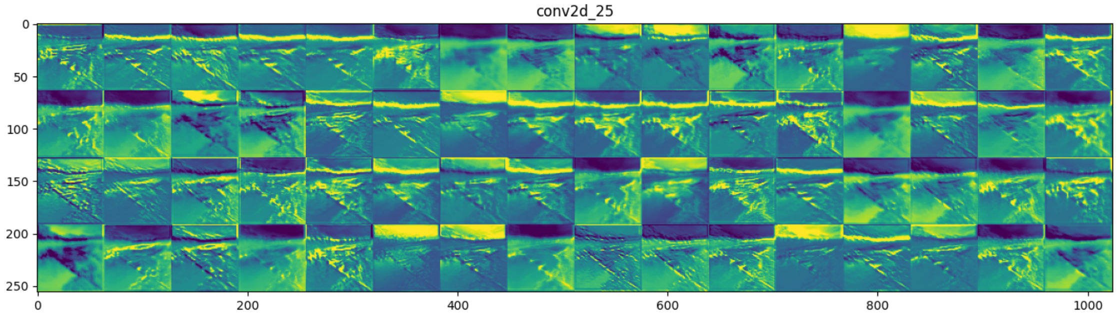
```
import matplotlib.pyplot as plt

layer_names = []
for layer in best_model.layers[:12]:
    layer_names.append(layer.name) # Names of the layers, so you can have them as part of your plot

images_per_row = 16

for layer_name, layer_activation in zip(layer_names, activations): # Displays the feature maps
    n_features = layer_activation.shape[-1] # Number of features in the feature map
    size = layer_activation.shape[1] #The feature map has shape (1, size, size, n_features).
    n_cols = n_features // images_per_row # Tiles the activation channels in this matrix
    display_grid = np.zeros((size * n_cols, images_per_row * size))

    print(layer_name)
    if ("flatten" in layer_name): break

    for col in range(n_cols): # Tiles each filter into a big horizontal grid
        for row in range(images_per_row):
            channel_image = layer_activation[0,
                                             :, :,
                                             col * images_per_row + row]
            channel_image -= channel_image.mean() # Post-processes the feature to make it visually palatable
            channel_image /= channel_image.std()
            channel_image *= 64
            channel_image += 128
            channel_image = np.clip(channel_image, 0, 255).astype('uint8')
            display_grid[col * size : (col + 1) * size, # Displays the grid
                         row * size : (row + 1) * size] = channel_image
    scale = 1. / size
    plt.figure(figsize=(scale * display_grid.shape[1],
                        scale * display_grid.shape[0]))
    plt.title(layer_name)
    plt.grid(False)
    plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

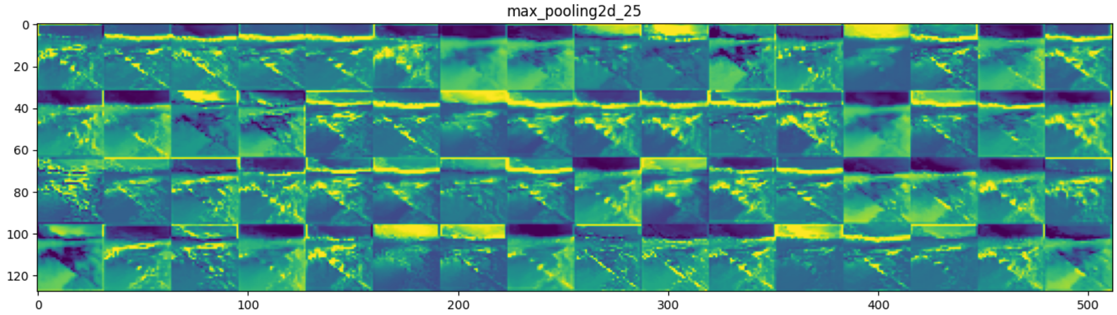
```
conv2d_24
max_pooling2d_24
conv2d_25
max_pooling2d_25
conv2d_26
max_pooling2d_26
conv2d_27
max_pooling2d_27
flatten_6
```

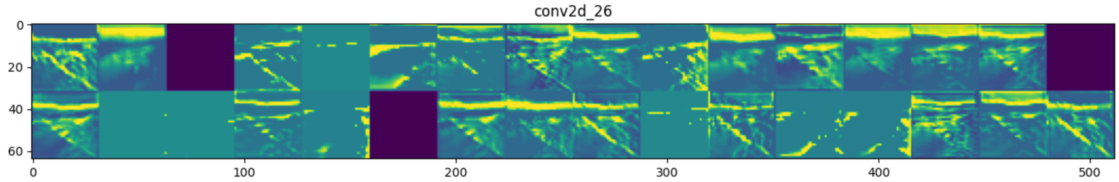
```
/tmp/ipykernel_35667/2269795348.py:24: RuntimeWarning: invalid value encountered
in divide
  channel_image /= channel_image.std()
```

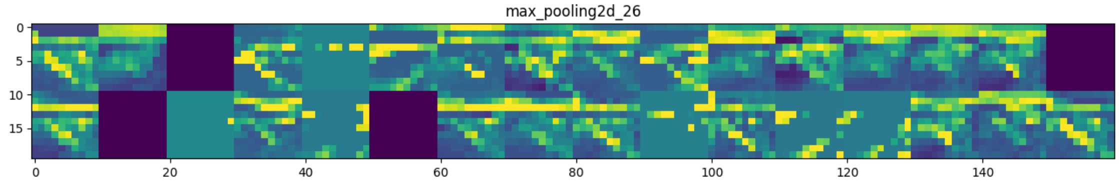


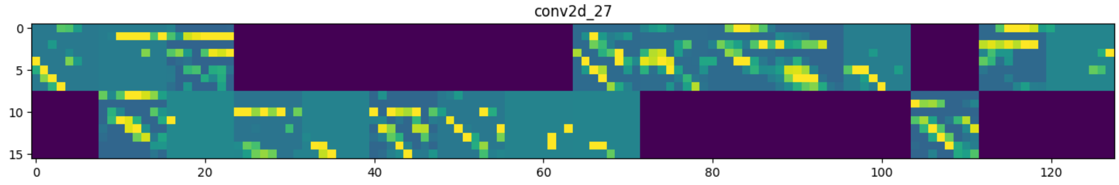


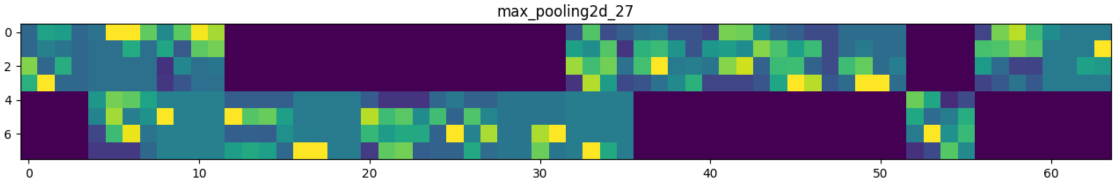

conv2d_25


max_pooling2d_25


conv2d_26


max_pooling2d_26


conv2d_27


max_pooling2d_27
0
2
4
6
0
10
20
30
40
50
60