cnn_v8

October 14, 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, AveragePooling2D
from keras.optimizers import SGD, Adam
```

[ ]:
```
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
```

```
2022-10-14 11:39:11.186822: 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.
2022-10-14 11:39:11.209616: 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
2022-10-14 11:39:11.261664: 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
2022-10-14 11:39:11.261850: 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
2022-10-14 11:39:12.100458: 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
2022-10-14 11:39:12.101067: 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
2022-10-14 11:39:12.101224: 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
2022-10-14 11:39:12.101367: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device
/device:GPU:0 with 4023 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"

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

```
2022-10-14 11:47:25.673029: 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
2022-10-14 11:47:25.673317: 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
2022-10-14 11:47:25.673566: 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
2022-10-14 11:47:25.674117: 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
2022-10-14 11:47:25.674268: 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
2022-10-14 11:47:25.674407: 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
2022-10-14 11:47:25.674587: 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
2022-10-14 11:47:25.674729: 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
2022-10-14 11:47:25.674844: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 4023 MB memory:  -> device: 0,
name: NVIDIA GeForce RTX 2060, pci bus id: 0000:08:00.0, compute capability: 7.5
```

[ ]:
```
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)
```

[ ]:
```
print(input_shape)
print(output_shape)
```

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

0.3 Create model

[ ]:
```
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(AveragePooling2D(pool_size=(2, 2)))

    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
    model.add(AveragePooling2D(pool_size=(2, 2)))



    model.add(Flatten())
    model.add(Dense(128, activation='tanh'))
    model.add(Dense(64, activation='tanh'))
    model.add(Dense(32, activation='tanh'))
    model.add(Dense(32, activation='tanh'))
    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_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)            (None, 256, 256, 64)      3136

 max_pooling2d_2 (MaxPooling  (None, 64, 64, 64)       0
 2D)

 conv2d_5 (Conv2D)            (None, 64, 64, 64)        65600

 max_pooling2d_3 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 conv2d_6 (Conv2D)            (None, 32, 32, 32)        18464

 conv2d_7 (Conv2D)            (None, 30, 30, 32)        9248

 average_pooling2d_2 (Averag  (None, 15, 15, 32)       0
 ePooling2D)

 flatten_1 (Flatten)          (None, 7200)              0

 dense_5 (Dense)              (None, 128)               921728

 dense_6 (Dense)              (None, 64)                8256

 dense_7 (Dense)              (None, 32)                2080

 dense_8 (Dense)              (None, 32)                1056

 dense_9 (Dense)              (None, 2)                 66

=================================================================
Total params: 1,029,634
Trainable params: 1,029,634
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, epochs=10, steps=32,␣
↪val_steps=32, callbacks=[]):
    return model.fit(training_values, validation_data=validation_values,␣
↪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, epochs=25, steps=np.ceil(train_size /␣
↪BATCH_SIZE), val_steps=np.ceil(val_size / BATCH_SIZE),␣
↪callbacks=[tensorboard_callback, checkpoint_callback])
```

```
Epoch 1/25
921/921 [==============================] - ETA: 0s - loss: 0.0987 - mse: 0.0987
- rmse: 0.3142 - mae: 0.1597 - mape: 66.8773
Epoch 1: val_mse improved from inf to 0.01878, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 111s 120ms/step - loss: 0.0987 - mse:
0.0987 - rmse: 0.3142 - mae: 0.1597 - mape: 66.8773 - val_loss: 0.0188 -
val_mse: 0.0188 - val_rmse: 0.1370 - val_mae: 0.0888 - val_mape: 32.6099
Epoch 2/25
920/921 [==============================>.] - ETA: 0s - loss: 0.0129 - mse: 0.0129
- rmse: 0.1134 - mae: 0.0691 - mape: 35.6959
Epoch 2: val_mse improved from 0.01878 to 0.00667, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 105s 114ms/step - loss: 0.0129 - mse:
0.0129 - rmse: 0.1134 - mae: 0.0691 - mape: 35.6947 - val_loss: 0.0067 -
val_mse: 0.0067 - val_rmse: 0.0817 - val_mae: 0.0544 - val_mape: 21.2948
Epoch 3/25
920/921 [==============================>.] - ETA: 0s - loss: 0.0056 - mse: 0.0056
```

```
- rmse: 0.0747 - mae: 0.0505 - mape: 23.9313
Epoch 3: val_mse improved from 0.00667 to 0.00385, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 107s 117ms/step - loss: 0.0056 - mse:
0.0056 - rmse: 0.0747 - mae: 0.0505 - mape: 23.9305 - val_loss: 0.0039 -
val_mse: 0.0039 - val_rmse: 0.0621 - val_mae: 0.0414 - val_mape: 14.9443
Epoch 4/25
920/921 [============================>.] - ETA: 0s - loss: 0.0044 - mse: 0.0044
- rmse: 0.0665 - mae: 0.0452 - mape: 21.1921
Epoch 4: val_mse did not improve from 0.00385
921/921 [==============================] - 108s 118ms/step - loss: 0.0044 - mse:
0.0044 - rmse: 0.0665 - mae: 0.0452 - mape: 21.1915 - val_loss: 0.0044 -
val_mse: 0.0044 - val_rmse: 0.0664 - val_mae: 0.0528 - val_mape: 15.3347
Epoch 5/25
920/921 [============================>.] - ETA: 0s - loss: 0.0035 - mse: 0.0035
- rmse: 0.0591 - mae: 0.0401 - mape: 18.6906
Epoch 5: val_mse did not improve from 0.00385
921/921 [==============================] - 110s 119ms/step - loss: 0.0035 - mse:
0.0035 - rmse: 0.0591 - mae: 0.0401 - mape: 18.6901 - val_loss: 0.0065 -
val_mse: 0.0065 - val_rmse: 0.0804 - val_mae: 0.0568 - val_mape: 16.2642
Epoch 6/25
920/921 [============================>.] - ETA: 0s - loss: 0.0032 - mse: 0.0032
- rmse: 0.0568 - mae: 0.0377 - mape: 16.6648
Epoch 6: val_mse did not improve from 0.00385
921/921 [==============================] - 107s 116ms/step - loss: 0.0032 - mse:
0.0032 - rmse: 0.0568 - mae: 0.0377 - mape: 16.6654 - val_loss: 0.0048 -
val_mse: 0.0048 - val_rmse: 0.0696 - val_mae: 0.0557 - val_mape: 22.7504
Epoch 7/25
920/921 [============================>.] - ETA: 0s - loss: 0.0029 - mse: 0.0029
- rmse: 0.0536 - mae: 0.0355 - mape: 17.7932
Epoch 7: val_mse did not improve from 0.00385
921/921 [==============================] - 98s 106ms/step - loss: 0.0029 - mse:
0.0029 - rmse: 0.0536 - mae: 0.0355 - mape: 17.7930 - val_loss: 0.0071 -
val_mse: 0.0071 - val_rmse: 0.0843 - val_mae: 0.0613 - val_mape: 21.6124
Epoch 8/25
920/921 [============================>.] - ETA: 0s - loss: 0.0030 - mse: 0.0030
- rmse: 0.0547 - mae: 0.0360 - mape: 19.0689
Epoch 8: val_mse improved from 0.00385 to 0.00231, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 102s 111ms/step - loss: 0.0030 - mse:
0.0030 - rmse: 0.0547 - mae: 0.0360 - mape: 19.0684 - val_loss: 0.0023 -
val_mse: 0.0023 - val_rmse: 0.0480 - val_mae: 0.0321 - val_mape: 11.6860
Epoch 9/25
920/921 [============================>.] - ETA: 0s - loss: 0.0023 - mse: 0.0023
- rmse: 0.0479 - mae: 0.0313 - mape: 15.5731
Epoch 9: val_mse improved from 0.00231 to 0.00201, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 110s 119ms/step - loss: 0.0023 - mse:
```

```
0.0023 - rmse: 0.0479 - mae: 0.0313 - mape: 15.5728 - val_loss: 0.0020 -
val_mse: 0.0020 - val_rmse: 0.0449 - val_mae: 0.0316 - val_mape: 14.6471
Epoch 10/25
920/921 [============================>.] - ETA: 0s - loss: 0.0021 - mse: 0.0021
- rmse: 0.0462 - mae: 0.0303 - mape: 16.4243
Epoch 10: val_mse improved from 0.00201 to 0.00165, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 105s 114ms/step - loss: 0.0021 - mse:
0.0021 - rmse: 0.0462 - mae: 0.0303 - mape: 16.4241 - val_loss: 0.0017 -
val_mse: 0.0017 - val_rmse: 0.0407 - val_mae: 0.0263 - val_mape: 10.4725
Epoch 11/25
920/921 [============================>.] - ETA: 0s - loss: 0.0021 - mse: 0.0021
- rmse: 0.0457 - mae: 0.0297 - mape: 16.0292
Epoch 11: val_mse did not improve from 0.00165
921/921 [==============================] - 108s 118ms/step - loss: 0.0021 - mse:
0.0021 - rmse: 0.0457 - mae: 0.0297 - mape: 16.0287 - val_loss: 0.0021 -
val_mse: 0.0021 - val_rmse: 0.0457 - val_mae: 0.0323 - val_mape: 16.1571
Epoch 12/25
920/921 [============================>.] - ETA: 0s - loss: 0.0020 - mse: 0.0020
- rmse: 0.0446 - mae: 0.0284 - mape: 13.2536
Epoch 12: val_mse did not improve from 0.00165
921/921 [==============================] - 110s 119ms/step - loss: 0.0020 - mse:
0.0020 - rmse: 0.0446 - mae: 0.0284 - mape: 13.2532 - val_loss: 0.0017 -
val_mse: 0.0017 - val_rmse: 0.0407 - val_mae: 0.0265 - val_mape: 10.0987
Epoch 13/25
920/921 [============================>.] - ETA: 0s - loss: 0.0019 - mse: 0.0019
- rmse: 0.0433 - mae: 0.0276 - mape: 13.7047
Epoch 13: val_mse improved from 0.00165 to 0.00164, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 111s 121ms/step - loss: 0.0019 - mse:
0.0019 - rmse: 0.0433 - mae: 0.0276 - mape: 13.7043 - val_loss: 0.0016 -
val_mse: 0.0016 - val_rmse: 0.0405 - val_mae: 0.0261 - val_mape: 10.0526
Epoch 14/25
920/921 [============================>.] - ETA: 0s - loss: 0.0016 - mse: 0.0016
- rmse: 0.0394 - mae: 0.0252 - mape: 14.1183
Epoch 14: val_mse improved from 0.00164 to 0.00158, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 111s 121ms/step - loss: 0.0016 - mse:
0.0016 - rmse: 0.0394 - mae: 0.0252 - mape: 14.1179 - val_loss: 0.0016 -
val_mse: 0.0016 - val_rmse: 0.0398 - val_mae: 0.0264 - val_mape: 9.7027
Epoch 15/25
920/921 [============================>.] - ETA: 0s - loss: 0.0054 - mse: 0.0054
- rmse: 0.0737 - mae: 0.0429 - mape: 20.7148
Epoch 15: val_mse did not improve from 0.00158
921/921 [==============================] - 111s 121ms/step - loss: 0.0054 - mse:
0.0054 - rmse: 0.0737 - mae: 0.0429 - mape: 20.7143 - val_loss: 0.0037 -
val_mse: 0.0037 - val_rmse: 0.0612 - val_mae: 0.0451 - val_mape: 15.5806
Epoch 16/25
```

```
920/921 [============================>.] - ETA: 0s - loss: 0.0022 - mse: 0.0022
- rmse: 0.0466 - mae: 0.0301 - mape: 17.8235
Epoch 16: val_mse did not improve from 0.00158
921/921 [==============================] - 112s 122ms/step - loss: 0.0022 - mse:
0.0022 - rmse: 0.0466 - mae: 0.0301 - mape: 17.8233 - val_loss: 0.0017 -
val_mse: 0.0017 - val_rmse: 0.0411 - val_mae: 0.0280 - val_mape: 9.7793
Epoch 17/25
920/921 [============================>.] - ETA: 0s - loss: 0.0014 - mse: 0.0014
- rmse: 0.0373 - mae: 0.0233 - mape: 12.1485
Epoch 17: val_mse improved from 0.00158 to 0.00146, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 109s 118ms/step - loss: 0.0014 - mse:
0.0014 - rmse: 0.0373 - mae: 0.0233 - mape: 12.1733 - val_loss: 0.0015 -
val_mse: 0.0015 - val_rmse: 0.0382 - val_mae: 0.0271 - val_mape: 10.0689
Epoch 18/25
920/921 [============================>.] - ETA: 0s - loss: 0.0012 - mse: 0.0012
- rmse: 0.0344 - mae: 0.0215 - mape: 12.4680
Epoch 18: val_mse improved from 0.00146 to 0.00134, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 105s 114ms/step - loss: 0.0012 - mse:
0.0012 - rmse: 0.0344 - mae: 0.0215 - mape: 12.4682 - val_loss: 0.0013 -
val_mse: 0.0013 - val_rmse: 0.0365 - val_mae: 0.0239 - val_mape: 8.8601
Epoch 19/25
920/921 [============================>.] - ETA: 0s - loss: 0.0012 - mse: 0.0012
- rmse: 0.0342 - mae: 0.0211 - mape: 10.3338
Epoch 19: val_mse did not improve from 0.00134
921/921 [==============================] - 107s 116ms/step - loss: 0.0012 - mse:
0.0012 - rmse: 0.0342 - mae: 0.0211 - mape: 10.3336 - val_loss: 0.0018 -
val_mse: 0.0018 - val_rmse: 0.0419 - val_mae: 0.0292 - val_mape: 13.8838
Epoch 20/25
920/921 [============================>.] - ETA: 0s - loss: 0.0012 - mse: 0.0012
- rmse: 0.0347 - mae: 0.0215 - mape: 12.0982
Epoch 20: val_mse did not improve from 0.00134
921/921 [==============================] - 107s 117ms/step - loss: 0.0012 - mse:
0.0012 - rmse: 0.0347 - mae: 0.0215 - mape: 12.0980 - val_loss: 0.0024 -
val_mse: 0.0024 - val_rmse: 0.0489 - val_mae: 0.0341 - val_mape: 13.9538
Epoch 21/25
920/921 [============================>.] - ETA: 0s - loss: 0.0013 - mse: 0.0013
- rmse: 0.0366 - mae: 0.0225 - mape: 12.6725
Epoch 21: val_mse did not improve from 0.00134
921/921 [==============================] - 107s 116ms/step - loss: 0.0013 - mse:
0.0013 - rmse: 0.0366 - mae: 0.0225 - mape: 12.6723 - val_loss: 0.0049 -
val_mse: 0.0049 - val_rmse: 0.0700 - val_mae: 0.0554 - val_mape: 20.3059
Epoch 22/25
920/921 [============================>.] - ETA: 0s - loss: 0.0032 - mse: 0.0032
- rmse: 0.0565 - mae: 0.0338 - mape: 17.4778
Epoch 22: val_mse did not improve from 0.00134
921/921 [==============================] - 108s 117ms/step - loss: 0.0032 - mse:
```

```
0.0032 - rmse: 0.0565 - mae: 0.0338 - mape: 17.4773 - val_loss: 0.0014 -
val_mse: 0.0014 - val_rmse: 0.0376 - val_mae: 0.0244 - val_mape: 9.6925
Epoch 23/25
920/921 [============================>.] - ETA: 0s - loss: 0.0011 - mse: 0.0011
- rmse: 0.0331 - mae: 0.0201 - mape: 10.4921
Epoch 23: val_mse improved from 0.00134 to 0.00129, saving model to
model_weights/20221014-123925_cnn_best_weights.hdf5
921/921 [==============================] - 108s 117ms/step - loss: 0.0011 - mse:
0.0011 - rmse: 0.0331 - mae: 0.0201 - mape: 10.4920 - val_loss: 0.0013 -
val_mse: 0.0013 - val_rmse: 0.0359 - val_mae: 0.0228 - val_mape: 8.9852
Epoch 24/25
920/921 [============================>.] - ETA: 0s - loss: 9.6015e-04 - mse:
9.6015e-04 - rmse: 0.0310 - mae: 0.0186 - mape: 9.2455
Epoch 24: val_mse did not improve from 0.00129
921/921 [==============================] - 105s 114ms/step - loss: 9.6012e-04 -
mse: 9.6012e-04 - rmse: 0.0310 - mae: 0.0186 - mape: 9.2453 - val_loss: 0.0014 -
val_mse: 0.0014 - val_rmse: 0.0370 - val_mae: 0.0239 - val_mape: 10.2687
Epoch 25/25
920/921 [============================>.] - ETA: 0s - loss: 9.8383e-04 - mse:
9.8383e-04 - rmse: 0.0314 - mae: 0.0191 - mape: 9.8359
Epoch 25: val_mse did not improve from 0.00129
921/921 [==============================] - 107s 116ms/step - loss: 9.8380e-04 -
mse: 9.8380e-04 - rmse: 0.0314 - mae: 0.0191 - mape: 9.8356 - val_loss: 0.0014 -
val_mse: 0.0014 - val_rmse: 0.0372 - val_mae: 0.0243 - val_mape: 10.0188
```

0.4 Evaluate model

[ ]:
```
print(date_actual)
```

```
20221014-123925
```

[ ]:
```
best_model = models.load_model(f'model_weights/{date_actual}_cnn_best_weights.
↪hdf5')
```

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

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

```
263/263 [==============================] - 17s 64ms/step - loss: 0.0042 - mse:
0.0042 - rmse: 0.0649 - mae: 0.0239 - mape: 18.2376
```

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

```
263/263 [==============================] - 17s 64ms/step
```

[ ]:
```
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)

        stage_discharge_test_values = stage_discharge[:10]
        predictions_values = predictions[:10]

        for i in range(len(stage_discharge_test_values.numpy())):
                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 144ms/step
76.13851851998916
70.14508968849664
pred stage: 3.2092437744140625, actual stage: 3.24
pred discharge: 1432.246337890625, actual discharge: 1450.0
pred stage: 3.1332528591156006, actual stage: 2.98
pred discharge: 1035.6925048828125, actual discharge: 813.0
pred stage: 2.30507493019104, actual stage: 2.31
pred discharge: 250.9044952392578, actual discharge: 261.0
pred stage: 3.960446357727051, actual stage: 4.0
pred discharge: 2270.589111328125, actual discharge: 2320.0
pred stage: 1.5294480323791504, actual stage: 1.54
pred discharge: 12.187579154968262, actual discharge: 14.899999999999977
pred stage: 5.178676128387451, actual stage: 5.22
pred discharge: 4734.4560546875, actual discharge: 4760.0
pred stage: 2.1457924842834473, actual stage: 2.15
pred discharge: 114.64216613769531, actual discharge: 151.0
pred stage: 2.6019279956817627, actual stage: 2.58
```

```
pred discharge: 484.6307678222656, actual discharge: 460.0
pred stage: 2.4353623390197754, actual stage: 2.46
pred discharge: 334.6253967285156, actual discharge: 343.0
pred stage: 4.1649675369262695, actual stage: 4.18
pred discharge: 2854.3115234375, actual discharge: 2850.0
```

[ ]:

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))
```

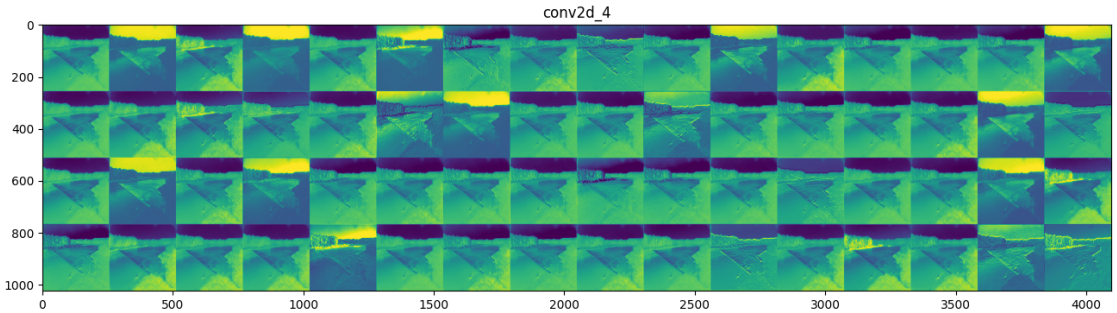
```
1/1 [==============================] - 0s 140ms/step
```

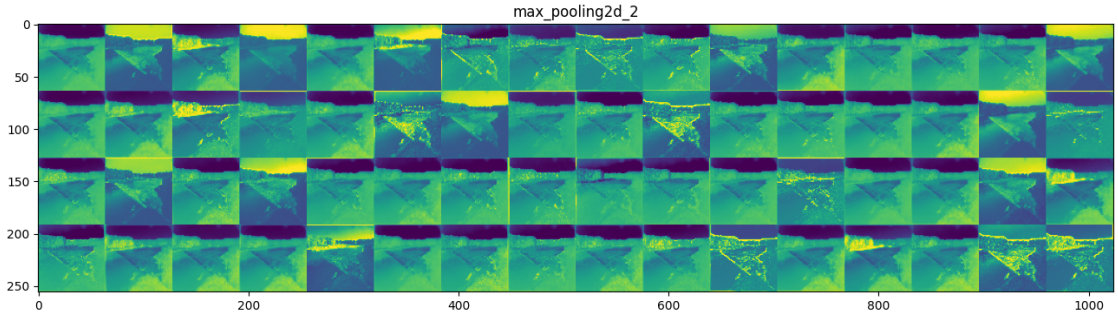
[ ]:
```
import matplotlib.pyplot as plt

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]
```

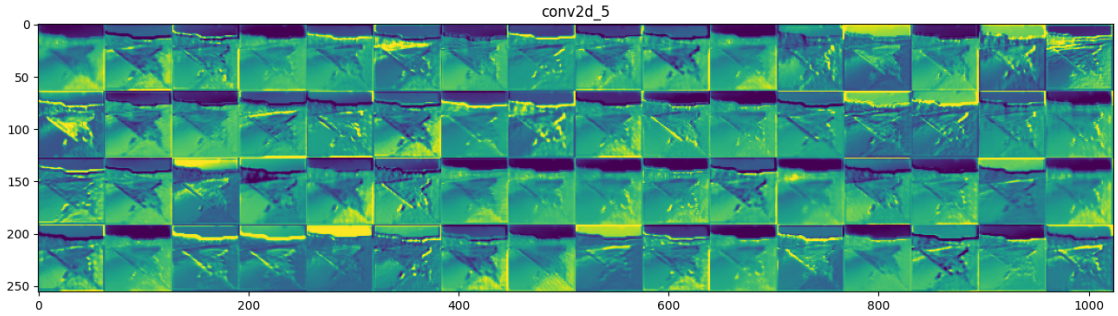
```
            channel_image -= channel_image.mean() # Post-processes the feature␣
↪to make it visually palatable
            channel_image /= channel_image.std()
            channel_image *= 64
            channel_image += 128
            channel_image = np.clip(channel_image, 0, 255).astype('uint8')
            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')
```

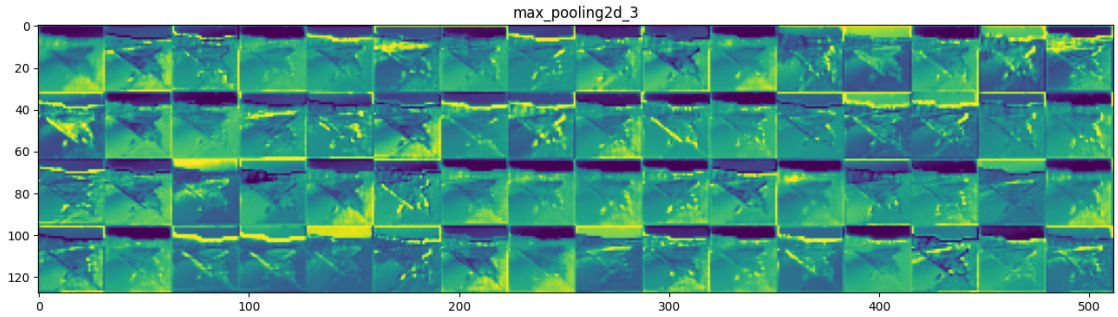
```
conv2d_4
max_pooling2d_2
conv2d_5
max_pooling2d_3
conv2d_6
conv2d_7
average_pooling2d_2
```

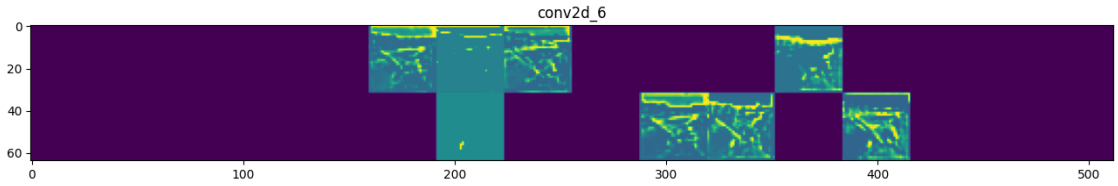
```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
Cell In [60], line 13
     11 size = layer_activation.shape[1] #The feature map has shape (1, size,␣
 ↪size, n_features).
     12 n_cols = n_features // images_per_row # Tiles the activation channels i␣
 ↪this matrix
---> 13 display_grid = np.zeros((size * n_cols, images_per_row * size))
     15 print(layer_name)
     16 if ("flatten" in layer_name): break

MemoryError: Unable to allocate 2.72 TiB for an array with shape (3240000,␣
 ↪115200) and data type float64
```

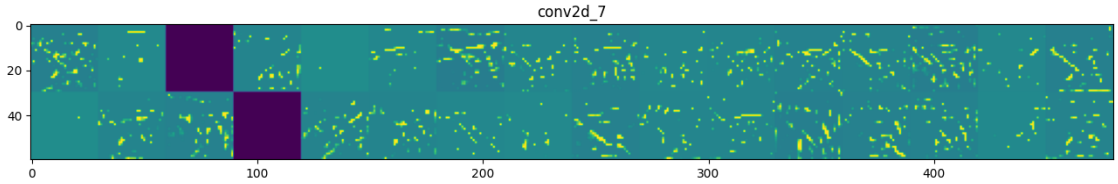


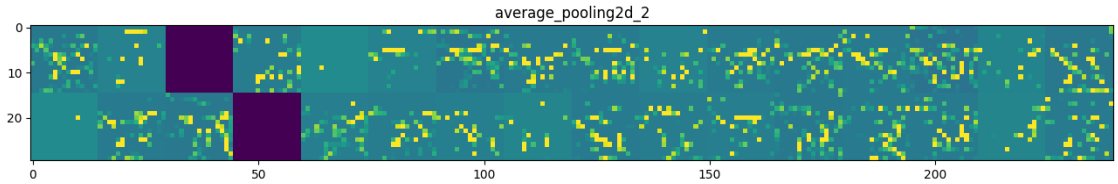

max_pooling2d_2
0
50
100
150
200
250
0
200
400
600
800
1000


conv2d_5
0
50
100
150
200
250
0
200
400
600
800
1000


max_pooling2d_3
0
20
40
60
80
100
120
0
100
200
300
400
500


conv2d_6
0
20
40
60
0
100
200
300
400
500


conv2d_7
0
20
40
0
100
200
300
400


average_pooling2d_2
0
10
20
0
50
100
150
200