cnn_v11

October 30, 2022

[ ]:
```
%env LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/
#%env TF_GPU_ALLOCATOR=cuda_malloc_async
```

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

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

```
$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/
```

[ ]:
```
import tensorflow as tf
import numpy as np
import pandas as pd
import os
import keras
import matplotlib.pyplot as plt

from tensorflow.keras import Sequential, models, Input
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D,␣
↪Dropout, LeakyReLU, AveragePooling2D, GlobalAveragePooling2D,␣
↪BatchNormalization, TimeDistributed, LSTM, SpatialDropout2D
from tensorflow.keras.optimizers import SGD, Adam
```

```
2022-10-30 19:17:16.242430: 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.
2022-10-30 19:17:16.678262: 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
2022-10-30 19:17:17.727708: 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/
```

```
2022-10-30 19:17:17.727788: 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/
2022-10-30 19:17:17.727793: 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.
```

[ ]:
```
from tensorflow.python.client import device_lib

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

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

```
2022-10-30 19:17:19.248427: 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.
2022-10-30 19:17:19.281426: 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
2022-10-30 19:17:19.322587: 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
2022-10-30 19:17:19.322783: 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
2022-10-30 19:17:20.170499: 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
2022-10-30 19:17:20.170698: 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
2022-10-30 19:17:20.170848: 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
```

```
2022-10-30 19:17:20.170982: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device
/device:GPU:0 with 4016 MB memory:  -> device: 0, name: NVIDIA GeForce RTX 2060,
pci bus id: 0000:08:00.0, compute capability: 7.5
```

[ ]:
```
physical_devices = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(physical_devices[0], True)
```

```
2022-10-30 19:17:20.209508: 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
2022-10-30 19:17:20.209776: 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
2022-10-30 19:17:20.209945: 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
```

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	Filename	Agency	SiteNumber	TimeZone	Stage	Discharge	CalcTimestamp	…	WeirPt2X	WeirPt2Y	WwRawLineMin
0	0	2012-06-09 13:15:00	2012-06-09T13:09:07	StateLineWeir_20120609_Farrell_001.jpg	USGS	6674500	MDT	2.99	916.0	2020-03-11T16:58:28	…	-1	-1	0.0
1	1	2012-06-09 13:15:00	2012-06-09T13:10:29	StateLineWeir_20120609_Farrell_002.jpg	USGS	6674500	MDT	2.99	916.0	2020-03-11T16:58:33	…	-1	-1	0.0
2	2	2012-06-09 13:45:00	2012-06-09T13:44:01	StateLineWeir_20120609_Farrell_003.jpg	USGS	6674500	MDT	2.96	873.0	2020-03-11T16:58:40	…	-1	-1	0.0
3	3	2012-06-09 14:45:00	2012-06-09T14:44:30	StateLineWeir_20120609_Farrell_004.jpg	USGS	6674500	MDT	2.94	846.0	2020-03-11T16:58:47	…	-1	-1	0.0
4	4	2012-06-09 15:45:00	2012-06-09T15:44:59	StateLineWeir_20120609_Farrell_005.jpg	USGS	6674500	MDT	2.94	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", 'SensorTime']]
```

[ ]:
```
df['SensorTime'] = pd.to_datetime(df['SensorTime'])
df['Year'] = df['SensorTime'].dt.year
df.head()
```

[ ]:
```
                              Filename  Stage  Discharge  \
0  StateLineWeir_20120609_Farrell_001.jpg   2.99      916.0
1  StateLineWeir_20120609_Farrell_002.jpg   2.99      916.0
2  StateLineWeir_20120609_Farrell_003.jpg   2.96      873.0
3  StateLineWeir_20120609_Farrell_004.jpg   2.94      846.0
4  StateLineWeir_20120609_Farrell_005.jpg   2.94      846.0

           SensorTime  Year
0 2012-06-09 13:15:00  2012
1 2012-06-09 13:15:00  2012
2 2012-06-09 13:45:00  2012
3 2012-06-09 14:45:00  2012
4 2012-06-09 15:45:00  2012
```

0.1.1 Remove outliers

[ ]:
```
df = df[df.Stage > 0]
df = df[df.Discharge > 0]
```

We consider values equal to 0 as outliers because from the photos it doesn't seem that it would be possible that at this time we would have a value of 0 for stage or discharge

[ ]:
```
df.shape
```

[ ]: (40148, 5)

0.1.2 Scale the data

[ ]:
```
from sklearn.preprocessing import StandardScaler
from joblib import load

scaler = StandardScaler()
#scaler = load('std_scaler.joblib') # scaler with all the 42059 observations
```

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

[ ]:
```
                                    Filename     Stage  Discharge  \
0      StateLineWeir_20120609_Farrell_001.jpg  0.106063  -0.084154
1      StateLineWeir_20120609_Farrell_002.jpg  0.106063  -0.084154
2      StateLineWeir_20120609_Farrell_003.jpg  0.069235  -0.119960
3      StateLineWeir_20120609_Farrell_004.jpg  0.044683  -0.142442
4      StateLineWeir_20120609_Farrell_005.jpg  0.044683  -0.142442
…                                        …         …          …
42054  StateLineWeir_20191011_Farrell_409.jpg -0.446354  -0.485510
42055  StateLineWeir_20191011_Farrell_410.jpg -0.446354  -0.485510
42056  StateLineWeir_20191011_Farrell_411.jpg -0.446354  -0.485510
42057  StateLineWeir_20191011_Farrell_412.jpg -0.446354  -0.485510
42058  StateLineWeir_20191011_Farrell_413.jpg -0.446354  -0.485510

               SensorTime  Year
0     2012-06-09 13:15:00  2012
1     2012-06-09 13:15:00  2012
2     2012-06-09 13:45:00  2012
3     2012-06-09 14:45:00  2012
4     2012-06-09 15:45:00  2012
…                     …     …
42054 2019-10-11 09:00:00  2019
42055 2019-10-11 10:00:00  2019
42056 2019-10-11 11:00:00  2019
42057 2019-10-11 12:00:00  2019
42058 2019-10-11 12:45:00  2019

[40148 rows x 5 columns]
```

[ ]:
```
from joblib import dump
#dump(scaler, 'std_scaler.joblib')
```

0.2 Create the dataset pipeline

[ ]:
```
IMG_SIZE = 224
#IMG_SIZE = 512
BATCH_SIZE = 32
FRAMES = 5
```

[ ]:
```
from dataset_transformer import make_dataset
```

[ ]:
```
path = "../../dataset/images_tmp_draw"

train_ds, train_size, val_ds, val_size, test_ds, test_size = make_dataset(path,␣
 ↪BATCH_SIZE, IMG_SIZE, FRAMES, df, 10, True, "cnn")
```

```
20304
7117
12727
```

```
2022-10-30 19:20:08.734097: 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
2022-10-30 19:20:08.734291: 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
2022-10-30 19:20:08.734431: 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
2022-10-30 19:20:08.734614: 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
2022-10-30 19:20:08.734757: 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
2022-10-30 19:20:08.734871: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 4016 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:]
```

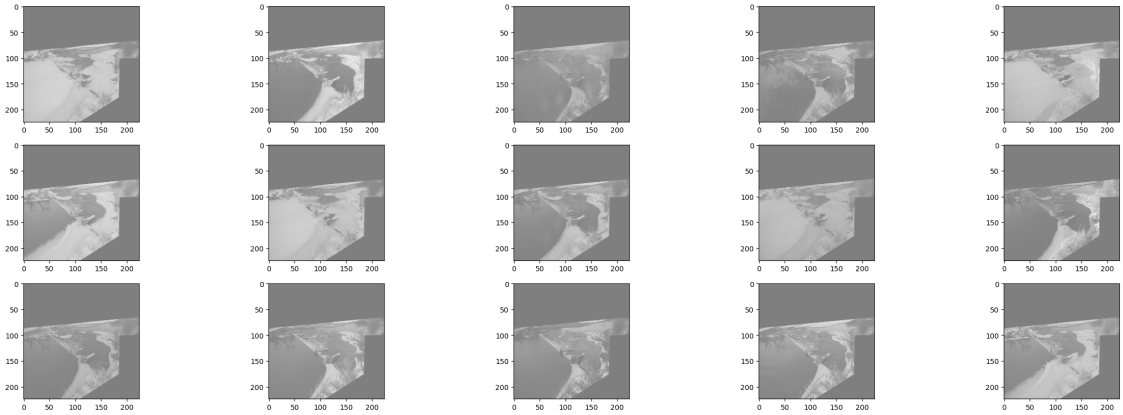
```
(32, 224, 224, 3)
(32, 2)
```

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

```
(224, 224, 3)
(2,)
```

0.3 Check images

[ ]:
```
fig, ax = plt.subplots(nrows=3, ncols=5, figsize=(30, 10))

for image, stage_discharge in test_ds.take(1):
    images = image[:15]
    for img, ax in zip(images, ax.flatten()):
        #img = img.numpy()[0]
        img = img.numpy()
        img = img / 2 + 0.5     # unnormalize
        ax.imshow(img)

plt.show()
```



0.4 Create model

[ ]:
```
def create_model(input_shape, output_shape, option="normal"):
    model = Sequential()

    if option == "transfer":
        base_model = tf.keras.applications.ResNet50V2(include_top=False,
                                                       weights='imagenet',
                                                       input_shape=input_shape)
        base_model.trainable = False
        base_model._name = 'base_model_ResNet50'

        model.add(base_model)
        model.add(Dropout(0.3))
        model.add(GlobalAveragePooling2D())

        model.add(Dense(512, activation='elu'))
        model.add(Dense(512, activation='elu'))
        model.add(Dense(256, activation='elu'))
        model.add(Dense(128, activation='elu'))
    elif option == "normal":
        model.add(Input(shape=input_shape))

        """model.add(Conv2D(16, kernel_size=(3, 3), activation="elu",␣
↪padding='same', kernel_initializer='he_uniform'))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
        model.add(BatchNormalization())

        model.add(Conv2D(32, kernel_size=(3, 3), activation="elu",␣
↪padding='same', kernel_initializer='he_uniform'))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
        model.add(BatchNormalization())

        model.add(Conv2D(32, kernel_size=(3, 3), activation="elu",␣
↪padding='same', kernel_initializer='he_uniform'))
        model.add(Conv2D(32, kernel_size=(3, 3), activation="elu",␣
↪padding='same', kernel_initializer='he_uniform'))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
        model.add(BatchNormalization())

        model.add(Conv2D(64, kernel_size=(4, 4), activation="elu",␣
↪padding='same', kernel_initializer='he_uniform'))
        model.add(Conv2D(64, kernel_size=(4, 4), activation="elu",␣
↪padding='same', kernel_initializer='he_uniform'))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
        model.add(BatchNormalization())
```

```
        model.add(Conv2D(64, kernel_size=(4, 4), activation="elu",␣
↪padding='same', kernel_initializer='he_uniform'))
        model.add(Conv2D(64, kernel_size=(4, 4), activation="elu",␣
↪padding='same', kernel_initializer='he_uniform'))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
        model.add(BatchNormalization())

        model.add(Conv2D(64, kernel_size=(3, 3), activation="elu",␣
↪padding='same', kernel_initializer='he_uniform'))
        model.add(Conv2D(64, kernel_size=(3, 3), activation="elu",␣
↪padding='same', kernel_initializer='he_uniform'))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
        model.add(BatchNormalization())

        model.add(GlobalAveragePooling2D())

        model.add(Dense(512, activation='elu'))
        model.add(Dropout(0.3))
        model.add(Dense(512, activation='elu'))
        model.add(Dropout(0.3))
        model.add(Dense(256, activation='elu'))
        model.add(Dense(64, activation='elu'))"""

        model.add(Conv2D(32, kernel_size=(4, 4), strides=(2, 2),␣
↪padding='same', activation="elu"))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Conv2D(32, kernel_size=(4, 4), strides=(2, 2),␣
↪activation="elu", padding='same'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Conv2D(64, kernel_size=(3, 3), activation="elu",␣
↪padding='same'))
        #model.add(AveragePooling2D(pool_size=(2, 2)))

        model.add(Conv2D(64, kernel_size=(3, 3), activation='elu'))

        model.add(Conv2D(64, kernel_size=(2, 2), activation='elu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Conv2D(64, kernel_size=(3, 3), activation='elu'))

        model.add(Conv2D(64, kernel_size=(2, 2), activation='elu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Flatten())
```

```
        model.add(Dense(256, activation='tanh'))
        model.add(Dropout(0.3))
        model.add(Dense(128, activation='tanh'))
        model.add(Dense(64, activation='tanh'))
        model.add(Dense(32, activation='tanh'))
    elif option == "cnn/lstm":
        model.add(Input(shape=input_shape))

        model.add(TimeDistributed(Conv2D(32, kernel_size=(4, 4), strides=(2,␣
↪2), padding='same', activation=LeakyReLU())))
        model.add(TimeDistributed(MaxPooling2D(pool_size=(4, 4))))

        model.add(TimeDistributed(Conv2D(32, kernel_size=(4, 4), strides=(2,␣
↪2), activation=LeakyReLU(), padding='same')))
        model.add(TimeDistributed(MaxPooling2D(pool_size=(2, 2))))

        model.add(TimeDistributed(Conv2D(32, kernel_size=(3, 3),␣
↪activation=LeakyReLU(0.2), padding='same')))
        model.add(TimeDistributed(AveragePooling2D(pool_size=(2, 2))))


        model.add(TimeDistributed(GlobalAveragePooling2D()))

        model.add(LSTM(10))

        model.add(Dense(64, 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], "normal")
#model = create_model(input_shape, output_shape[1], "cnn/lstm")
```

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

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_32 (Conv2D)          (None, 112, 112, 32)      1568

 max_pooling2d_14 (MaxPoolin  (None, 56, 56, 32)       0
 g2D)
```

```
conv2d_33 (Conv2D)           (None, 28, 28, 32)        16416

max_pooling2d_15 (MaxPoolin  (None, 14, 14, 32)        0
g2D)

conv2d_34 (Conv2D)           (None, 14, 14, 64)        18496

conv2d_35 (Conv2D)           (None, 12, 12, 64)        36928

conv2d_36 (Conv2D)           (None, 11, 11, 64)        16448

max_pooling2d_16 (MaxPoolin  (None, 5, 5, 64)          0
g2D)

conv2d_37 (Conv2D)           (None, 3, 3, 64)          36928

conv2d_38 (Conv2D)           (None, 2, 2, 64)          16448

max_pooling2d_17 (MaxPoolin  (None, 1, 1, 64)          0
g2D)

flatten_4 (Flatten)          (None, 64)                0

dense_20 (Dense)             (None, 256)               16640

dropout_4 (Dropout)          (None, 256)               0

dense_21 (Dense)             (None, 128)               32896

dense_22 (Dense)             (None, 64)                8256

dense_23 (Dense)             (None, 32)                2080

dense_24 (Dense)             (None, 2)                 66

=================================================================
Total params: 203,170
Trainable params: 203,170
Non-trainable params: 0

_________________________________________________________________
```

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

[ ]:
```python
sgd = SGD(learning_rate=0.01, decay=1e-3, momentum=0.9, nesterov=True)
adam = Adam(learning_rate=1e-3, decay=1e-3 / 200)
```

```
compile_model('mse', sgd,  [
              '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)

es_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min',␣
↪verbose=1, patience=5)

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

[ ]:
```
# batch_size = 0 because we already have batch size in tf dataset
model_h = fit_model(train_ds, val_ds, epochs=60, steps=np.ceil(train_size /␣
↪BATCH_SIZE), val_steps=np.ceil(val_size / BATCH_SIZE),␣
↪callbacks=[tensorboard_callback, checkpoint_callback, es_callback])
```

```
Epoch 1/60
635/635 [==============================] - ETA: 0s - loss: 0.1610 - mse: 0.1610
- rmse: 0.4013 - mae: 0.2118 - mape: 68.5414
Epoch 1: val_loss improved from inf to 0.04874, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 31s 48ms/step - loss: 0.1610 - mse:
0.1610 - rmse: 0.4013 - mae: 0.2118 - mape: 68.5414 - val_loss: 0.0487 -
val_mse: 0.0487 - val_rmse: 0.2208 - val_mae: 0.1577 - val_mape: 41.4325
Epoch 2/60
635/635 [==============================] - ETA: 0s - loss: 0.0190 - mse: 0.0190
- rmse: 0.1379 - mae: 0.0948 - mape: 38.0835
Epoch 2: val_loss improved from 0.04874 to 0.03504, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 27s 43ms/step - loss: 0.0190 - mse:
0.0190 - rmse: 0.1379 - mae: 0.0948 - mape: 38.0835 - val_loss: 0.0350 -
```

```
val_mse: 0.0350 - val_rmse: 0.1872 - val_mae: 0.1340 - val_mape: 39.6013
Epoch 3/60
634/635 [============================>.] - ETA: 0s - loss: 0.0135 - mse: 0.0135
- rmse: 0.1162 - mae: 0.0801 - mape: 31.9182
Epoch 3: val_loss improved from 0.03504 to 0.03017, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 41ms/step - loss: 0.0135 - mse:
0.0135 - rmse: 0.1162 - mae: 0.0800 - mape: 31.9021 - val_loss: 0.0302 -
val_mse: 0.0302 - val_rmse: 0.1737 - val_mae: 0.1270 - val_mape: 39.2433
Epoch 4/60
635/635 [==============================] - ETA: 0s - loss: 0.0115 - mse: 0.0115
- rmse: 0.1071 - mae: 0.0739 - mape: 29.4948
Epoch 4: val_loss improved from 0.03017 to 0.02752, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 41ms/step - loss: 0.0115 - mse:
0.0115 - rmse: 0.1071 - mae: 0.0739 - mape: 29.4948 - val_loss: 0.0275 -
val_mse: 0.0275 - val_rmse: 0.1659 - val_mae: 0.1206 - val_mape: 37.2144
Epoch 5/60
635/635 [==============================] - ETA: 0s - loss: 0.0101 - mse: 0.0101
- rmse: 0.1005 - mae: 0.0693 - mape: 27.3959
Epoch 5: val_loss did not improve from 0.02752
635/635 [==============================] - 26s 40ms/step - loss: 0.0101 - mse:
0.0101 - rmse: 0.1005 - mae: 0.0693 - mape: 27.3959 - val_loss: 0.0276 -
val_mse: 0.0276 - val_rmse: 0.1662 - val_mae: 0.1144 - val_mape: 32.7270
Epoch 6/60
635/635 [==============================] - ETA: 0s - loss: 0.0092 - mse: 0.0092
- rmse: 0.0959 - mae: 0.0664 - mape: 26.6044
Epoch 6: val_loss improved from 0.02752 to 0.02501, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 40ms/step - loss: 0.0092 - mse:
0.0092 - rmse: 0.0959 - mae: 0.0664 - mape: 26.6044 - val_loss: 0.0250 -
val_mse: 0.0250 - val_rmse: 0.1581 - val_mae: 0.1112 - val_mape: 33.5459
Epoch 7/60
635/635 [==============================] - ETA: 0s - loss: 0.0085 - mse: 0.0085
- rmse: 0.0921 - mae: 0.0642 - mape: 26.2814
Epoch 7: val_loss improved from 0.02501 to 0.02380, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 41ms/step - loss: 0.0085 - mse:
0.0085 - rmse: 0.0921 - mae: 0.0642 - mape: 26.2814 - val_loss: 0.0238 -
val_mse: 0.0238 - val_rmse: 0.1543 - val_mae: 0.1069 - val_mape: 31.6758
Epoch 8/60
635/635 [==============================] - ETA: 0s - loss: 0.0083 - mse: 0.0083
- rmse: 0.0912 - mae: 0.0634 - mape: 25.5699
Epoch 8: val_loss improved from 0.02380 to 0.02371, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 41ms/step - loss: 0.0083 - mse:
0.0083 - rmse: 0.0912 - mae: 0.0634 - mape: 25.5699 - val_loss: 0.0237 -
val_mse: 0.0237 - val_rmse: 0.1540 - val_mae: 0.1072 - val_mape: 31.8466
```

```
Epoch 9/60
633/635 [============================>.] - ETA: 0s - loss: 0.0077 - mse: 0.0077
- rmse: 0.0879 - mae: 0.0610 - mape: 25.2393
Epoch 9: val_loss improved from 0.02371 to 0.02269, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 27s 42ms/step - loss: 0.0077 - mse:
0.0077 - rmse: 0.0879 - mae: 0.0610 - mape: 25.2183 - val_loss: 0.0227 -
val_mse: 0.0227 - val_rmse: 0.1506 - val_mae: 0.1030 - val_mape: 29.6224
Epoch 10/60
635/635 [==============================] - ETA: 0s - loss: 0.0077 - mse: 0.0077
- rmse: 0.0878 - mae: 0.0608 - mape: 24.0997
Epoch 10: val_loss improved from 0.02269 to 0.02234, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 41ms/step - loss: 0.0077 - mse:
0.0077 - rmse: 0.0878 - mae: 0.0608 - mape: 24.0997 - val_loss: 0.0223 -
val_mse: 0.0223 - val_rmse: 0.1495 - val_mae: 0.1036 - val_mape: 30.8130
Epoch 11/60
634/635 [============================>.] - ETA: 0s - loss: 0.0070 - mse: 0.0070
- rmse: 0.0839 - mae: 0.0585 - mape: 23.9554
Epoch 11: val_loss improved from 0.02234 to 0.02147, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 41ms/step - loss: 0.0070 - mse:
0.0070 - rmse: 0.0839 - mae: 0.0585 - mape: 23.9445 - val_loss: 0.0215 -
val_mse: 0.0215 - val_rmse: 0.1465 - val_mae: 0.1007 - val_mape: 29.5933
Epoch 12/60
635/635 [==============================] - ETA: 0s - loss: 0.0070 - mse: 0.0070
- rmse: 0.0837 - mae: 0.0579 - mape: 24.0591
Epoch 12: val_loss did not improve from 0.02147
635/635 [==============================] - 26s 41ms/step - loss: 0.0070 - mse:
0.0070 - rmse: 0.0837 - mae: 0.0579 - mape: 24.0591 - val_loss: 0.0217 -
val_mse: 0.0217 - val_rmse: 0.1474 - val_mae: 0.1014 - val_mape: 29.3682
Epoch 13/60
635/635 [==============================] - ETA: 0s - loss: 0.0068 - mse: 0.0068
- rmse: 0.0823 - mae: 0.0573 - mape: 24.0166
Epoch 13: val_loss did not improve from 0.02147
635/635 [==============================] - 26s 41ms/step - loss: 0.0068 - mse:
0.0068 - rmse: 0.0823 - mae: 0.0573 - mape: 24.0166 - val_loss: 0.0220 -
val_mse: 0.0220 - val_rmse: 0.1484 - val_mae: 0.1023 - val_mape: 29.7553
Epoch 14/60
633/635 [============================>.] - ETA: 0s - loss: 0.0067 - mse: 0.0067
- rmse: 0.0818 - mae: 0.0569 - mape: 23.8327
Epoch 14: val_loss improved from 0.02147 to 0.02129, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 41ms/step - loss: 0.0067 - mse:
0.0067 - rmse: 0.0819 - mae: 0.0570 - mape: 23.8016 - val_loss: 0.0213 -
val_mse: 0.0213 - val_rmse: 0.1459 - val_mae: 0.1006 - val_mape: 29.6195
Epoch 15/60
633/635 [============================>.] - ETA: 0s - loss: 0.0064 - mse: 0.0064
```

```
- rmse: 0.0801 - mae: 0.0559 - mape: 23.0364
Epoch 15: val_loss improved from 0.02129 to 0.02126, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 27s 42ms/step - loss: 0.0064 - mse:
0.0064 - rmse: 0.0801 - mae: 0.0559 - mape: 23.0177 - val_loss: 0.0213 -
val_mse: 0.0213 - val_rmse: 0.1458 - val_mae: 0.1004 - val_mape: 28.8095
Epoch 16/60
635/635 [==============================] - ETA: 0s - loss: 0.0064 - mse: 0.0064
- rmse: 0.0803 - mae: 0.0557 - mape: 22.5464
Epoch 16: val_loss did not improve from 0.02126
635/635 [==============================] - 26s 41ms/step - loss: 0.0064 - mse:
0.0064 - rmse: 0.0803 - mae: 0.0557 - mape: 22.5464 - val_loss: 0.0214 -
val_mse: 0.0214 - val_rmse: 0.1464 - val_mae: 0.0997 - val_mape: 27.7094
Epoch 17/60
633/635 [============================>.] - ETA: 0s - loss: 0.0063 - mse: 0.0063
- rmse: 0.0793 - mae: 0.0552 - mape: 23.4435
Epoch 17: val_loss did not improve from 0.02126
635/635 [==============================] - 26s 40ms/step - loss: 0.0063 - mse:
0.0063 - rmse: 0.0794 - mae: 0.0552 - mape: 23.4109 - val_loss: 0.0216 -
val_mse: 0.0216 - val_rmse: 0.1471 - val_mae: 0.1006 - val_mape: 28.6674
Epoch 18/60
633/635 [============================>.] - ETA: 0s - loss: 0.0063 - mse: 0.0063
- rmse: 0.0791 - mae: 0.0547 - mape: 22.6994
Epoch 18: val_loss did not improve from 0.02126
635/635 [==============================] - 26s 40ms/step - loss: 0.0063 - mse:
0.0063 - rmse: 0.0791 - mae: 0.0547 - mape: 22.6669 - val_loss: 0.0221 -
val_mse: 0.0221 - val_rmse: 0.1486 - val_mae: 0.1013 - val_mape: 27.8553
Epoch 19/60
635/635 [==============================] - ETA: 0s - loss: 0.0061 - mse: 0.0061
- rmse: 0.0783 - mae: 0.0546 - mape: 22.7656
Epoch 19: val_loss improved from 0.02126 to 0.02073, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 42ms/step - loss: 0.0061 - mse:
0.0061 - rmse: 0.0783 - mae: 0.0546 - mape: 22.7656 - val_loss: 0.0207 -
val_mse: 0.0207 - val_rmse: 0.1440 - val_mae: 0.0990 - val_mape: 28.4503
Epoch 20/60
632/635 [============================>.] - ETA: 0s - loss: 0.0059 - mse: 0.0059
- rmse: 0.0770 - mae: 0.0535 - mape: 22.1829
Epoch 20: val_loss improved from 0.02073 to 0.02018, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 41ms/step - loss: 0.0059 - mse:
0.0059 - rmse: 0.0770 - mae: 0.0535 - mape: 22.1841 - val_loss: 0.0202 -
val_mse: 0.0202 - val_rmse: 0.1421 - val_mae: 0.0969 - val_mape: 27.7550
Epoch 21/60
635/635 [==============================] - ETA: 0s - loss: 0.0060 - mse: 0.0060
- rmse: 0.0773 - mae: 0.0537 - mape: 22.4613
Epoch 21: val_loss did not improve from 0.02018
635/635 [==============================] - 26s 41ms/step - loss: 0.0060 - mse:
```

```
0.0060 - rmse: 0.0773 - mae: 0.0537 - mape: 22.4613 - val_loss: 0.0209 -
val_mse: 0.0209 - val_rmse: 0.1446 - val_mae: 0.0990 - val_mape: 27.7635
Epoch 22/60
635/635 [==============================] - ETA: 0s - loss: 0.0059 - mse: 0.0059
- rmse: 0.0766 - mae: 0.0533 - mape: 22.1828
Epoch 22: val_loss did not improve from 0.02018
635/635 [==============================] - 26s 41ms/step - loss: 0.0059 - mse:
0.0059 - rmse: 0.0766 - mae: 0.0533 - mape: 22.1828 - val_loss: 0.0206 -
val_mse: 0.0206 - val_rmse: 0.1437 - val_mae: 0.0978 - val_mape: 27.7665
Epoch 23/60
635/635 [==============================] - ETA: 0s - loss: 0.0058 - mse: 0.0058
- rmse: 0.0764 - mae: 0.0530 - mape: 22.2552
Epoch 23: val_loss did not improve from 0.02018
635/635 [==============================] - 26s 41ms/step - loss: 0.0058 - mse:
0.0058 - rmse: 0.0764 - mae: 0.0530 - mape: 22.2552 - val_loss: 0.0207 -
val_mse: 0.0207 - val_rmse: 0.1440 - val_mae: 0.0983 - val_mape: 27.4993
Epoch 24/60
635/635 [==============================] - ETA: 0s - loss: 0.0058 - mse: 0.0058
- rmse: 0.0760 - mae: 0.0528 - mape: 21.7594
Epoch 24: val_loss improved from 0.02018 to 0.01967, saving model to
model_weights/20221030-202000_cnn_best_weights.hdf5
635/635 [==============================] - 26s 41ms/step - loss: 0.0058 - mse:
0.0058 - rmse: 0.0760 - mae: 0.0528 - mape: 21.7594 - val_loss: 0.0197 -
val_mse: 0.0197 - val_rmse: 0.1403 - val_mae: 0.0958 - val_mape: 27.2156
Epoch 25/60
634/635 [==============================>.] - ETA: 0s - loss: 0.0058 - mse: 0.0058
- rmse: 0.0763 - mae: 0.0526 - mape: 21.9381
Epoch 25: val_loss did not improve from 0.01967
635/635 [==============================] - 25s 40ms/step - loss: 0.0058 - mse:
0.0058 - rmse: 0.0763 - mae: 0.0526 - mape: 21.9236 - val_loss: 0.0201 -
val_mse: 0.0201 - val_rmse: 0.1417 - val_mae: 0.0969 - val_mape: 27.2769
Epoch 26/60
632/635 [==============================>.] - ETA: 0s - loss: 0.0056 - mse: 0.0056
- rmse: 0.0747 - mae: 0.0520 - mape: 22.2876
Epoch 26: val_loss did not improve from 0.01967
635/635 [==============================] - 26s 41ms/step - loss: 0.0056 - mse:
0.0056 - rmse: 0.0747 - mae: 0.0520 - mape: 22.2408 - val_loss: 0.0213 -
val_mse: 0.0213 - val_rmse: 0.1461 - val_mae: 0.0996 - val_mape: 27.1532
Epoch 27/60
635/635 [==============================] - ETA: 0s - loss: 0.0056 - mse: 0.0056
- rmse: 0.0749 - mae: 0.0520 - mape: 22.2381
Epoch 27: val_loss did not improve from 0.01967
635/635 [==============================] - 29s 45ms/step - loss: 0.0056 - mse:
0.0056 - rmse: 0.0749 - mae: 0.0520 - mape: 22.2381 - val_loss: 0.0201 -
val_mse: 0.0201 - val_rmse: 0.1419 - val_mae: 0.0970 - val_mape: 27.4242
Epoch 28/60
635/635 [==============================] - ETA: 0s - loss: 0.0054 - mse: 0.0054
- rmse: 0.0737 - mae: 0.0515 - mape: 21.9319
```

```
Epoch 28: val_loss did not improve from 0.01967
635/635 [==============================] - 27s 42ms/step - loss: 0.0054 - mse:
0.0054 - rmse: 0.0737 - mae: 0.0515 - mape: 21.9319 - val_loss: 0.0201 -
val_mse: 0.0201 - val_rmse: 0.1419 - val_mae: 0.0971 - val_mape: 27.1583
Epoch 29/60
635/635 [==============================] - ETA: 0s - loss: 0.0055 - mse: 0.0055
- rmse: 0.0743 - mae: 0.0516 - mape: 21.7428
Epoch 29: val_loss did not improve from 0.01967
635/635 [==============================] - 26s 41ms/step - loss: 0.0055 - mse:
0.0055 - rmse: 0.0743 - mae: 0.0516 - mape: 21.7428 - val_loss: 0.0203 -
val_mse: 0.0203 - val_rmse: 0.1424 - val_mae: 0.0973 - val_mape: 27.1245
Epoch 29: early stopping
```

0.5 Evaluate model

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

```
20221030-202000
```

[ ]:
```
best_model = models.load_model(f'model_weights/{date_actual}_cnn_best_weights.
↪hdf5')
#best_model = models.load_model(f'best_models_weights/cnn_best_weights_v9.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))
```

```
398/398 [==============================] - 12s 30ms/step - loss: 0.0505 - mse:
0.0505 - rmse: 0.2246 - mae: 0.1389 - mape: 72.7527
```

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

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

        stage_discharge_test_values = stage_discharge.numpy()
        predictions_values = predictions

        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 88ms/step
441.19266182980726
3243.6085036341224
pred stage: 2.074333429336548, actual stage: 2.42
pred discharge: 267.8599853515625, actual discharge: 210.0
pred stage: 2.1716480255126953, actual stage: 2.24
pred discharge: 227.58538818359375, actual discharge: 193.9999999999999
pred stage: 2.097628116607666, actual stage: 2.24
pred discharge: 134.73583984375, actual discharge: 193.9999999999999
pred stage: 1.9582111835479736, actual stage: 2.46
pred discharge: 116.59221649169922, actual discharge: 209.0
pred stage: 2.164323091506958, actual stage: 2.24
pred discharge: 195.36407470703125, actual discharge: 193.9999999999999
pred stage: 2.0977611541748047, actual stage: 2.25
pred discharge: 154.81903076171875, actual discharge: 198.0
pred stage: 2.106117010116577, actual stage: 2.25
pred discharge: 164.81689453125, actual discharge: 199.0
pred stage: 2.0872802734375, actual stage: 2.25
pred discharge: 146.6630859375, actual discharge: 198.0
pred stage: 2.1385679244995117, actual stage: 2.26
pred discharge: 154.3438720703125, actual discharge: 204.0
pred stage: 2.1561810970306396, actual stage: 2.26
pred discharge: 170.2928466796875, actual discharge: 204.0
```

0.5.1 Residual analysis

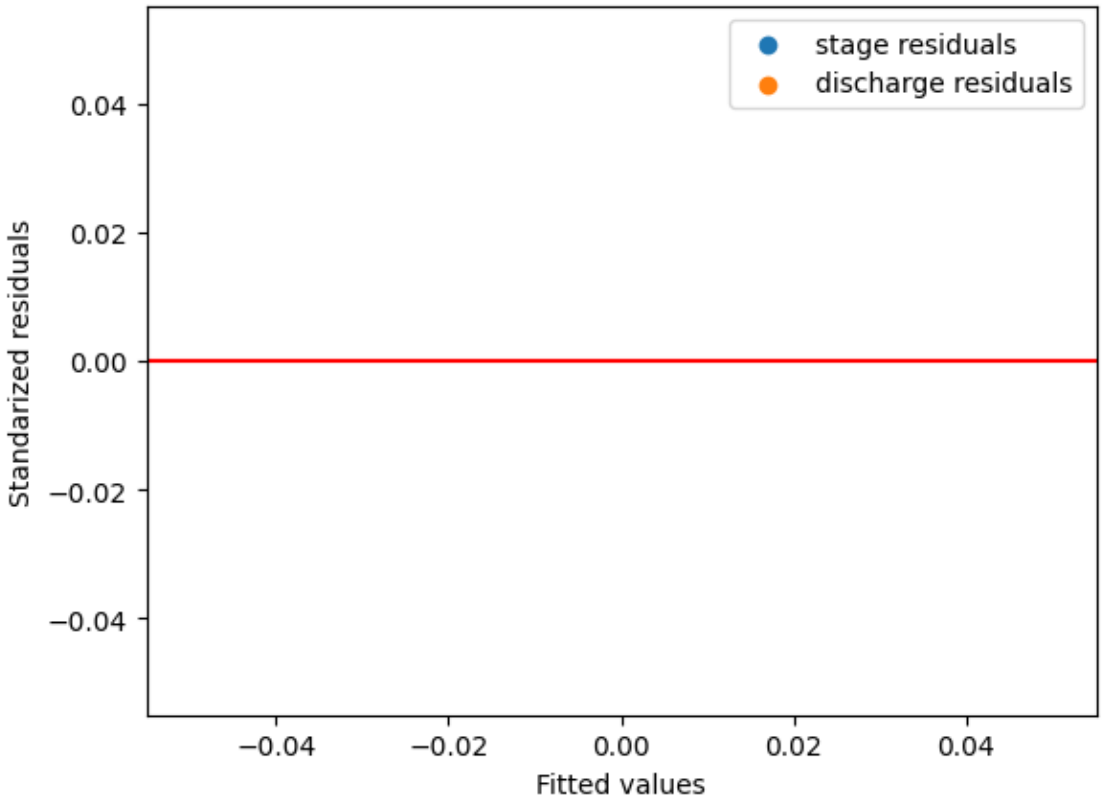
[ ]:
```
y_predictions = np.empty(shape=(1, 2))
y_real = np.empty(shape=(1, 2))

"""for image, stage_discharge in test_ds.take(100):
```

```
    y_predictions = np.concatenate((y_predictions, best_model.predict(x=image)))
    y_real = np.concatenate((y_real, stage_discharge.numpy()))"""
```

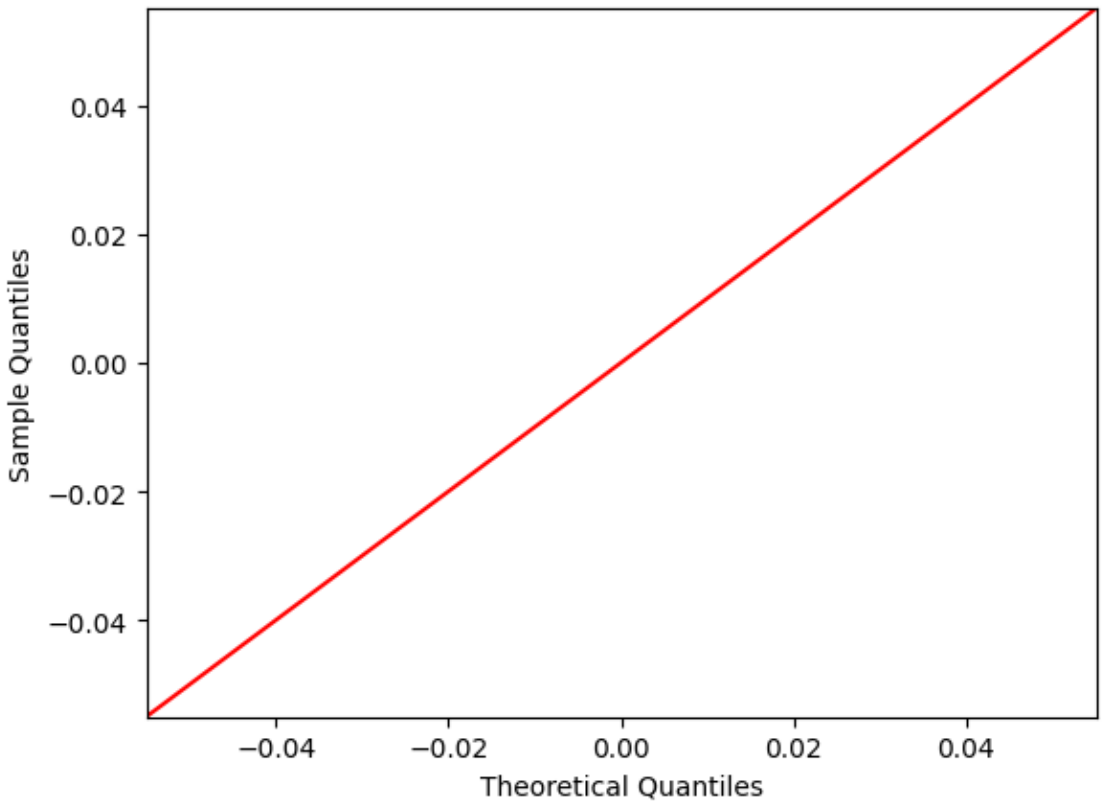
[ ]:
```
'for image, stage_discharge in test_ds.take(100):\n    y_predictions =
np.concatenate((y_predictions, best_model.predict(x=image)))\n    y_real =
np.concatenate((y_real, stage_discharge.numpy()))'
```

[ ]:
```
residuals = y_real - y_predictions
residuals_std = residuals/residuals.std()

y_real_stage = np.array([i[0] for i in y_real])
residual_stage = np.array([i[0] for i in residuals])

y_real_discharge = np.array([i[1] for i in y_real])
residual_discharge = np.array([i[1] for i in residuals])

plt.scatter(y_real_stage, residual_stage / residual_stage.std(), label="stage␣
 ↪residuals")
plt.scatter(y_real_discharge, residual_discharge / residual_discharge.std(),␣
 ↪label="discharge residuals")
plt.axhline(y=0.0, color='r', linestyle='-')
plt.xlabel("Fitted values")
plt.ylabel("Standarized residuals")

plt.legend()
plt.show()
```

```
/tmp/ipykernel_23436/3264406076.py:11: RuntimeWarning: divide by zero
encountered in divide
  plt.scatter(y_real_discharge, residual_discharge / residual_discharge.std(),
label="discharge residuals")
```

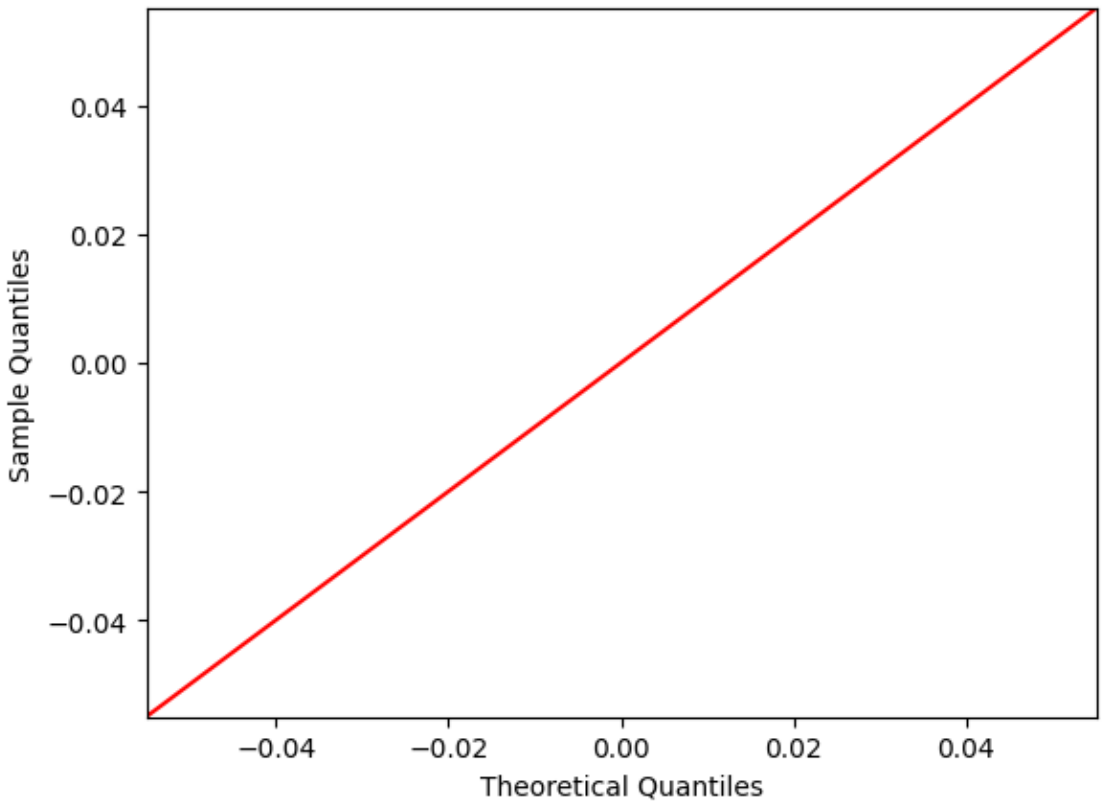


[ ]:
```
import statsmodels.api as sm
from statsmodels.stats.diagnostic import normal_ad

figure = sm.qqplot(residual_stage / residual_stage.std(), line ='45',␣
 ↪label='stage')
plt.show()
```



[ ]:
```
figure = sm.qqplot(residual_discharge / residual_discharge.std(), line='45',␣
↪label='discharge')
plt.show()
```

```
/tmp/ipykernel_23436/1192211388.py:1: RuntimeWarning: divide by zero encountered
in divide
  figure = sm.qqplot(residual_discharge / residual_discharge.std(), line='45',
label='discharge')
```



[ ]:
```
import seaborn as sns

#sns.histplot(residuals, kde=True, bins = 10)
```

[ ]:
```
stat, pval = normal_ad(residual_stage)
print("p-value:", pval)

if pval<0.05:
    print("Hay evidencia de que los residuos no provienen de una distribución␣
 ↪normal.")
else:
    print("No hay evidencia para rechazar la hipótesis de que los residuos␣
 ↪vienen de una distribución normal.")
```

```
p-value: 0.0
Hay evidencia de que los residuos no provienen de una distribución normal.

/home/nkspartan/miniconda3/envs/tf-gpu/lib/python3.10/site-
packages/numpy/core/_methods.py:265: RuntimeWarning: Degrees of freedom <= 0 for
slice
  ret = _var(a, axis=axis, dtype=dtype, out=out, ddof=ddof,
```

[ ]:
```
stat, pval = normal_ad(residual_discharge)
print("p-value:", pval)

if pval < 0.05:
    print("Hay evidencia de que los residuos no provienen de una distribución␣
 ↪normal.")
else:
    print("No hay evidencia para rechazar la hipótesis de que los residuos␣
 ↪vienen de una distribución normal.")
```

```
p-value: 0.0
Hay evidencia de que los residuos no provienen de una distribución normal.
```

```
/home/nkspartan/miniconda3/envs/tf-gpu/lib/python3.10/site-
packages/numpy/core/_methods.py:257: RuntimeWarning: invalid value encountered
in double_scalars
  ret = ret.dtype.type(ret / rcount)
```

0.6 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))
```

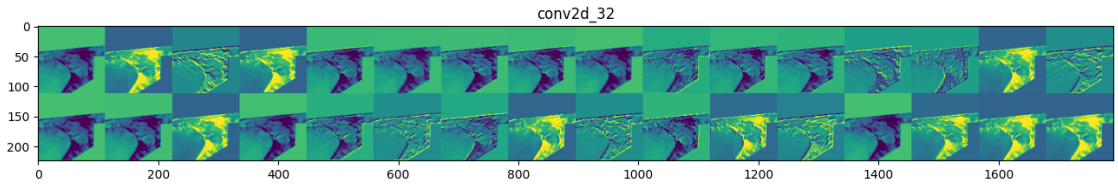
```
1/1 [==============================] - 0s 212ms/step
```

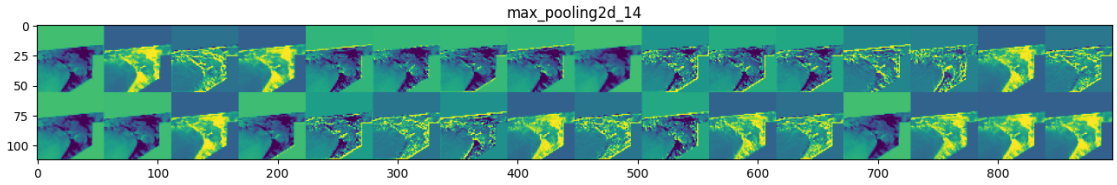
[ ]:
```
import matplotlib.pyplot as plt

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))
```

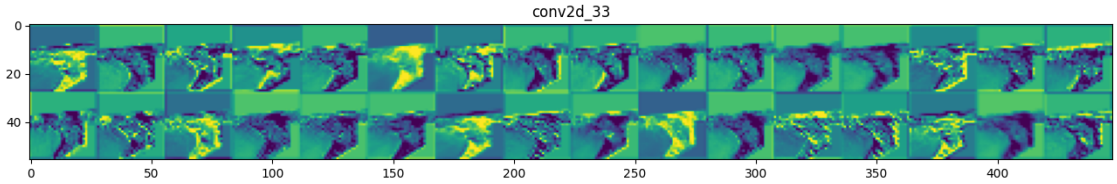
```
    print(layer_name)
    if "flatten" in layer_name or "dense" in layer_name: break

    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')
```

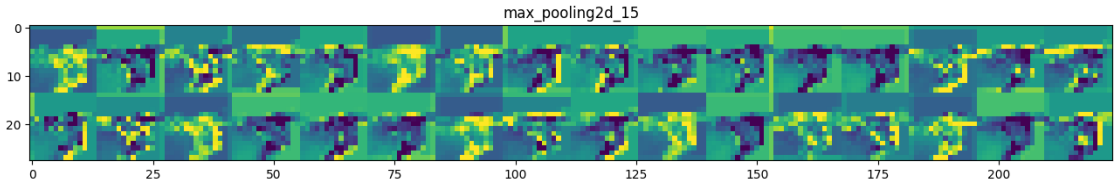
```
conv2d_32
max_pooling2d_14
conv2d_33
max_pooling2d_15
conv2d_34
conv2d_35
conv2d_36
max_pooling2d_16
conv2d_37
conv2d_38
max_pooling2d_17
flatten_4
```

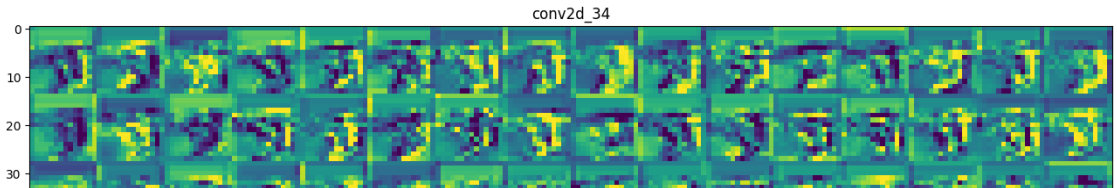
```
/tmp/ipykernel_23436/1031377702.py:24: RuntimeWarning: invalid value encountered
in divide
  channel_image /= channel_image.std()
```

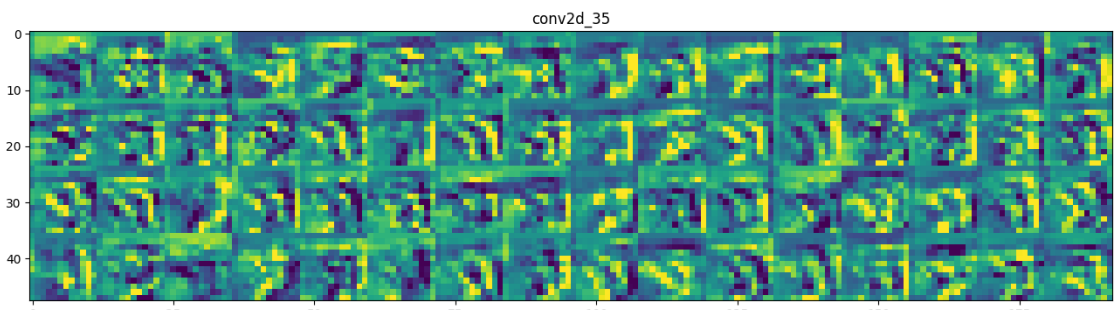


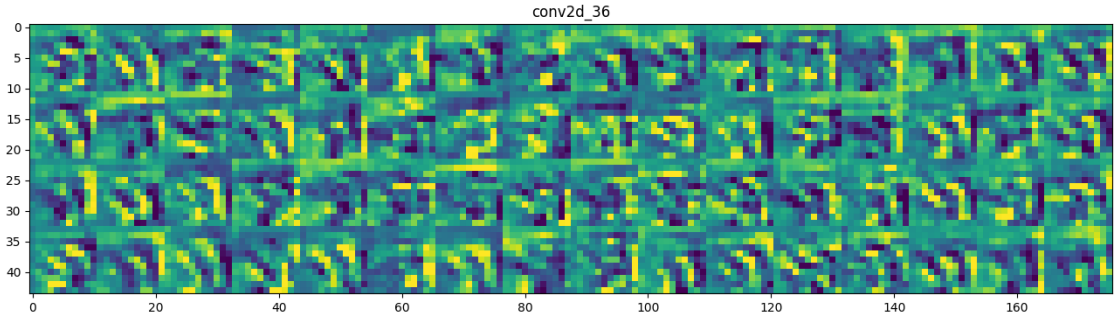

max_pooling2d_14

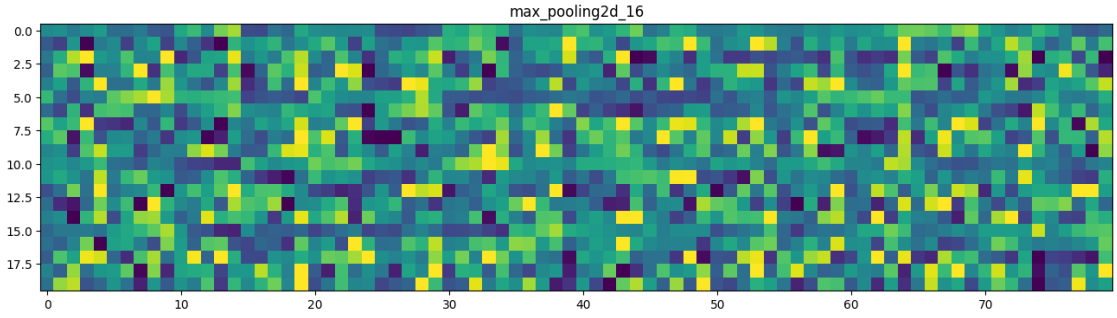

conv2d_33

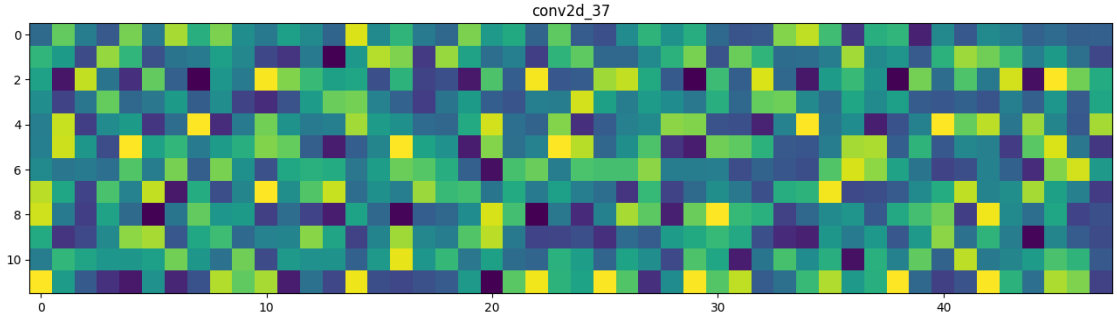

max_pooling2d_15

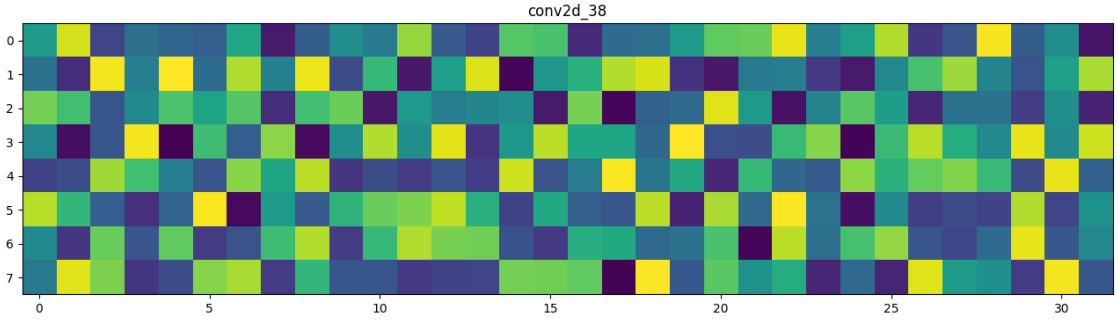

conv2d_34

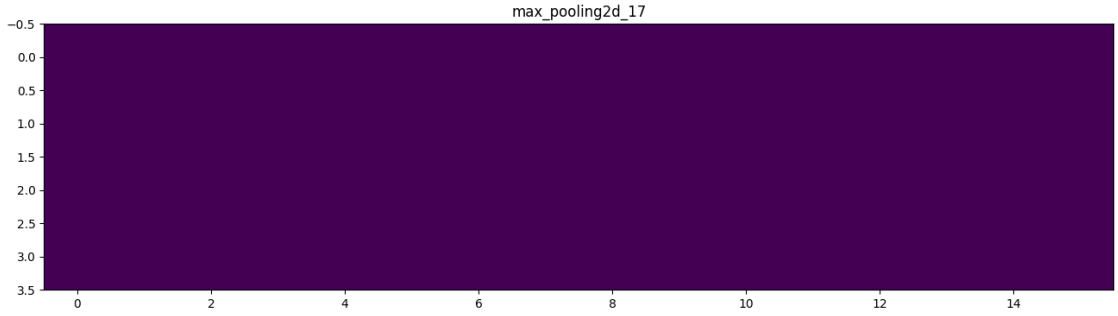

conv2d_35


conv2d_36
0
5
10
15
20
25
30
35
40
0
20
40
60
80
100
120
140
160


max_pooling2d_16
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
10
20
30
40
50
60
70


conv2d_37
0
2
4
6
8
10
0
10
20
30
40


conv2d_38
0
1
2
3
4
5
6
7
0
5
10
15
20
25
30


max_pooling2d_17
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
2
4
6
8
10
12
14