cnn_v12

November 4, 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 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
```

```
2022-11-03 22:13:21.588312: 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.
2022-11-03 22:13:22.142775: 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
2022-11-03 22:13:23.284245: 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/
2022-11-03 22:13:23.284487: 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/
2022-11-03 22:13:23.284497: 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

2022-11-03 22:13:24.971054: 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.
2022-11-03 22:13:25.006076: 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
2022-11-03 22:13:25.050066: 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
2022-11-03 22:13:25.050275: 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
2022-11-03 22:13:25.934775: 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
2022-11-03 22:13:25.935857: 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
2022-11-03 22:13:25.936081: 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
2022-11-03 22:13:25.936259: I
```

```
tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device
/device:GPU:0 with 4008 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)
```

```
2022-11-03 22:13:25.991236: 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
2022-11-03 22:13:25.992002: 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
2022-11-03 22:13:25.992181: 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
```

[ ]:
```
df = df.sort_values(by="SensorTime", ascending=True)
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
```

Scale the data based only on the training dataset (in this case the training dataset is from 2012 to 2016)

[ ]:
```
data_to_scale_fit = df[(df["Year"] >= 2012) & (df["Year"] <= 2016)][["Stage",␣
↪"Discharge"]]
data_to_scale_fit
```

[ ]:
```
       Stage  Discharge
0       2.99      916.0
1       2.99      916.0
2       2.96      873.0
3       2.94      846.0
4       2.94      846.0
…       …         …
21416   2.38      279.0
21417   2.38      279.0
21418   2.38      279.0
21419   2.38      279.0
21420   2.38      279.0

[20304 rows x 2 columns]
```

[ ]:
```
scaler.fit(data_to_scale_fit)
```

[ ]:
```
StandardScaler()
```

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

[ ]:
```
                                    Filename     Stage  Discharge  \
0      StateLineWeir_20120609_Farrell_001.jpg  0.077964  -0.136077
1      StateLineWeir_20120609_Farrell_002.jpg  0.077964  -0.136077
2      StateLineWeir_20120609_Farrell_003.jpg  0.045759  -0.165451
3      StateLineWeir_20120609_Farrell_004.jpg  0.024290  -0.183894
4      StateLineWeir_20120609_Farrell_005.jpg  0.024290  -0.183894
…                                        …         …          …
42054  StateLineWeir_20191011_Farrell_409.jpg -0.405103  -0.465332
42055  StateLineWeir_20191011_Farrell_410.jpg -0.405103  -0.465332
42056  StateLineWeir_20191011_Farrell_411.jpg -0.405103  -0.465332
42057  StateLineWeir_20191011_Farrell_412.jpg -0.405103  -0.465332
42058  StateLineWeir_20191011_Farrell_413.jpg -0.405103  -0.465332

                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_train_value_0_outliers.joblib')
```

0.2 Create the dataset pipeline

[ ]:
```
IMG_SIZE = 224
#IMG_SIZE = 512
BATCH_SIZE = 32
FRAMES = 10
```

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

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

with tf.device("/gpu:0"):
    train_ds, train_size, val_ds, val_size, test_ds, test_size =␣
 ↪make_dataset(path, BATCH_SIZE, IMG_SIZE, FRAMES, df, 10, True, "cnn")
```

```
2022-11-03 22:13:27.304760: 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
2022-11-03 22:13:27.304964: 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
2022-11-03 22:13:27.305105: 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
2022-11-03 22:13:27.305466: 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
2022-11-03 22:13:27.305620: 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
2022-11-03 22:13:27.305737: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1616] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 4008 MB memory:  -> device: 0,
name: NVIDIA GeForce RTX 2060, pci bus id: 0000:08:00.0, compute capability: 7.5
```

```
20304
7117
12727
```

[ ]:
```
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:]
```

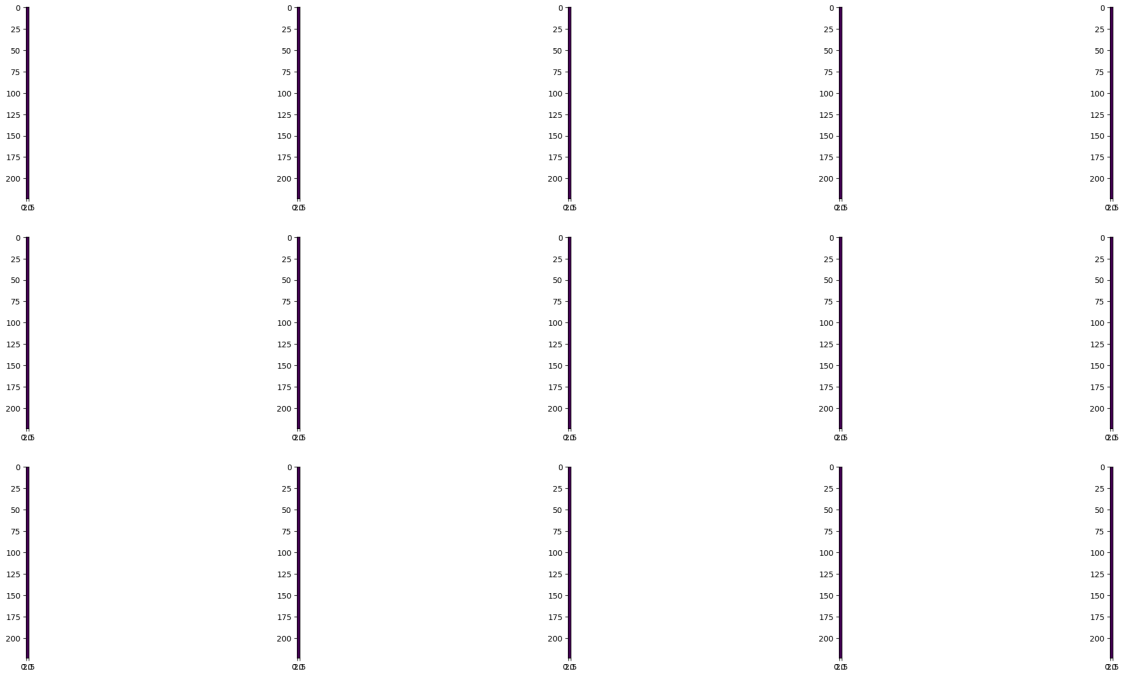
```
(32, 224, 224, 3)
(32, 2)
```

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

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

0.3 Check images

[ ]:
```
fig, ax = plt.subplots(nrows=3, ncols=5, figsize=(30, 15))

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)

        for layer in base_model.layers:
            layer.trainable = False

        base_model._name = 'base_model_ResNet50'

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

        model.add(Dense(1024, 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(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":
        base_model = tf.keras.applications.ResNet50V2(include_top=False,
                                                       weights='imagenet',
                                                       input_shape=(224, 224, 3))
        base_model.trainable = False
        base_model._name = 'base_model_ResNet50'
        model.add(Input(shape=input_shape))
        model.add(TimeDistributed(base_model))
        model.add(TimeDistributed(Dropout(0.3)))
        model.add(TimeDistributed(GlobalAveragePooling2D()))

        """model.add(Input(shape=input_shape))

        model.add(TimeDistributed(Conv2D(16, kernel_size=(4, 4), strides=(2,␣
↪2), padding='same', activation='elu')))
        model.add(TimeDistributed(MaxPooling2D(pool_size=(3, 3))))

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

        model.add(TimeDistributed(Conv2D(32, kernel_size=(3, 3),␣
↪activation='elu', padding='same')))
        model.add(TimeDistributed(MaxPooling2D(pool_size=(2, 2))))

        model.add(TimeDistributed(Conv2D(32, kernel_size=(3, 3),␣
↪activation='elu', padding='same')))
        model.add(TimeDistributed(MaxPooling2D(pool_size=(2, 2))))


        model.add(TimeDistributed(Flatten()))"""

        model.add(LSTM(10, return_sequences=True))
        model.add(LSTM(15))

        model.add(Dense(512, activation='elu'))
        model.add(Dense(256, activation='elu'))
```

```
        model.add(Dense(128, activation='elu'))
        model.add(Dense(128, activation='elu'))

    model.add(Dense(output_shape, activation='linear')) # linear regression␣
↪output layer

    return model
```

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

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

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 base_model_ResNet50 (Functi  (None, 7, 7, 2048)       23564800
 onal)

 dropout_6 (Dropout)         (None, 7, 7, 2048)        0

 global_average_pooling2d_2   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_10 (Dense)            (None, 1024)              2098176

 dropout_7 (Dropout)         (None, 1024)              0

 dense_11 (Dense)            (None, 512)               524800

 dropout_8 (Dropout)         (None, 512)               0

 dense_12 (Dense)            (None, 256)               131328

 dense_13 (Dense)            (None, 128)               32896

 dense_14 (Dense)            (None, 2)                 258

=================================================================
Total params: 26,352,258
Trainable params: 2,787,458
Non-trainable params: 23,564,800
_________________________________________________________________
```

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

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)

es_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min',␣
↪verbose=1, patience=8)

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
with tf.device("/gpu:0"):
    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
634/635 [============================>.] - ETA: 0s - loss: 0.2345 - mse: 0.2345
- rmse: 0.4842 - mae: 0.2784 - mape: 140.3885
Epoch 1: val_loss improved from inf to 0.08667, saving model to
model_weights/20221103-230002_cnn_best_weights.hdf5
635/635 [==============================] - 80s 122ms/step - loss: 0.2343 - mse:
0.2343 - rmse: 0.4841 - mae: 0.2784 - mape: 140.3092 - val_loss: 0.0867 -
val_mse: 0.0867 - val_rmse: 0.2944 - val_mae: 0.2105 - val_mape: 65.5157
Epoch 2/60
634/635 [============================>.] - ETA: 0s - loss: 0.0419 - mse: 0.0419
```

```
- rmse: 0.2046 - mae: 0.1448 - mape: 78.3303
Epoch 2: val_loss improved from 0.08667 to 0.07906, saving model to
model_weights/20221103-230002_cnn_best_weights.hdf5
635/635 [==============================] - 76s 120ms/step - loss: 0.0419 - mse:
0.0419 - rmse: 0.2046 - mae: 0.1448 - mape: 78.2950 - val_loss: 0.0791 -
val_mse: 0.0791 - val_rmse: 0.2812 - val_mae: 0.1927 - val_mape: 64.4429
Epoch 3/60
634/635 [============================>.] - ETA: 0s - loss: 0.0305 - mse: 0.0305
- rmse: 0.1748 - mae: 0.1226 - mape: 70.2164
Epoch 3: val_loss improved from 0.07906 to 0.07415, saving model to
model_weights/20221103-230002_cnn_best_weights.hdf5
635/635 [==============================] - 76s 119ms/step - loss: 0.0305 - mse:
0.0305 - rmse: 0.1747 - mae: 0.1225 - mape: 70.1723 - val_loss: 0.0742 -
val_mse: 0.0742 - val_rmse: 0.2723 - val_mae: 0.1817 - val_mape: 60.1087
Epoch 4/60
634/635 [============================>.] - ETA: 0s - loss: 0.0270 - mse: 0.0270
- rmse: 0.1642 - mae: 0.1144 - mape: 63.5036
Epoch 4: val_loss improved from 0.07415 to 0.06344, saving model to
model_weights/20221103-230002_cnn_best_weights.hdf5
635/635 [==============================] - 76s 119ms/step - loss: 0.0269 - mse:
0.0269 - rmse: 0.1641 - mae: 0.1144 - mape: 63.4689 - val_loss: 0.0634 -
val_mse: 0.0634 - val_rmse: 0.2519 - val_mae: 0.1574 - val_mape: 49.7525
Epoch 5/60
634/635 [============================>.] - ETA: 0s - loss: 0.0224 - mse: 0.0224
- rmse: 0.1496 - mae: 0.1036 - mape: 57.7936
Epoch 5: val_loss did not improve from 0.06344
635/635 [==============================] - 75s 118ms/step - loss: 0.0224 - mse:
0.0224 - rmse: 0.1496 - mae: 0.1036 - mape: 57.7635 - val_loss: 0.0652 -
val_mse: 0.0652 - val_rmse: 0.2553 - val_mae: 0.1577 - val_mape: 49.2678
Epoch 6/60
634/635 [============================>.] - ETA: 0s - loss: 0.0219 - mse: 0.0219
- rmse: 0.1479 - mae: 0.1024 - mape: 59.2665
Epoch 6: val_loss did not improve from 0.06344
635/635 [==============================] - 75s 118ms/step - loss: 0.0219 - mse:
0.0219 - rmse: 0.1479 - mae: 0.1024 - mape: 59.2348 - val_loss: 0.0666 -
val_mse: 0.0666 - val_rmse: 0.2582 - val_mae: 0.1632 - val_mape: 50.4255
Epoch 7/60
634/635 [============================>.] - ETA: 0s - loss: 0.0192 - mse: 0.0192
- rmse: 0.1387 - mae: 0.0957 - mape: 55.0968
Epoch 7: val_loss did not improve from 0.06344
635/635 [==============================] - 75s 119ms/step - loss: 0.0192 - mse:
0.0192 - rmse: 0.1387 - mae: 0.0957 - mape: 55.0644 - val_loss: 0.0690 -
val_mse: 0.0690 - val_rmse: 0.2627 - val_mae: 0.1642 - val_mape: 59.2706
Epoch 8/60
634/635 [============================>.] - ETA: 0s - loss: 0.0247 - mse: 0.0247
- rmse: 0.1572 - mae: 0.1047 - mape: 58.8439
Epoch 8: val_loss did not improve from 0.06344
635/635 [==============================] - 75s 119ms/step - loss: 0.0247 - mse:
```

```
0.0247 - rmse: 0.1571 - mae: 0.1047 - mape: 58.8113 - val_loss: 0.0781 -
val_mse: 0.0781 - val_rmse: 0.2794 - val_mae: 0.1724 - val_mape: 55.7035
Epoch 9/60
634/635 [============================>.] - ETA: 0s - loss: 0.0246 - mse: 0.0246
- rmse: 0.1568 - mae: 0.1041 - mape: 59.2239
Epoch 9: val_loss did not improve from 0.06344
635/635 [==============================] - 75s 118ms/step - loss: 0.0246 - mse:
0.0246 - rmse: 0.1568 - mae: 0.1041 - mape: 59.1929 - val_loss: 0.0802 -
val_mse: 0.0802 - val_rmse: 0.2831 - val_mae: 0.1682 - val_mape: 48.9895
Epoch 10/60
634/635 [============================>.] - ETA: 0s - loss: 0.0170 - mse: 0.0170
- rmse: 0.1302 - mae: 0.0882 - mape: 51.5287
Epoch 10: val_loss improved from 0.06344 to 0.06236, saving model to
model_weights/20221103-230002_cnn_best_weights.hdf5
635/635 [==============================] - 76s 119ms/step - loss: 0.0170 - mse:
0.0170 - rmse: 0.1302 - mae: 0.0881 - mape: 51.4980 - val_loss: 0.0624 -
val_mse: 0.0624 - val_rmse: 0.2497 - val_mae: 0.1509 - val_mape: 58.8778
Epoch 11/60
634/635 [============================>.] - ETA: 0s - loss: 0.0145 - mse: 0.0145
- rmse: 0.1205 - mae: 0.0821 - mape: 50.1194
Epoch 11: val_loss improved from 0.06236 to 0.05849, saving model to
model_weights/20221103-230002_cnn_best_weights.hdf5
635/635 [==============================] - 76s 119ms/step - loss: 0.0145 - mse:
0.0145 - rmse: 0.1205 - mae: 0.0821 - mape: 50.0889 - val_loss: 0.0585 -
val_mse: 0.0585 - val_rmse: 0.2419 - val_mae: 0.1437 - val_mape: 44.7324
Epoch 12/60
634/635 [============================>.] - ETA: 0s - loss: 0.0147 - mse: 0.0147
- rmse: 0.1211 - mae: 0.0813 - mape: 47.9710
Epoch 12: val_loss did not improve from 0.05849
635/635 [==============================] - 75s 119ms/step - loss: 0.0147 - mse:
0.0147 - rmse: 0.1211 - mae: 0.0813 - mape: 47.9434 - val_loss: 0.1153 -
val_mse: 0.1153 - val_rmse: 0.3395 - val_mae: 0.2035 - val_mape: 49.4566
Epoch 13/60
634/635 [============================>.] - ETA: 0s - loss: 0.0145 - mse: 0.0145
- rmse: 0.1203 - mae: 0.0804 - mape: 48.8194
Epoch 13: val_loss did not improve from 0.05849
635/635 [==============================] - 72s 113ms/step - loss: 0.0145 - mse:
0.0145 - rmse: 0.1202 - mae: 0.0804 - mape: 48.7864 - val_loss: 0.0618 -
val_mse: 0.0618 - val_rmse: 0.2487 - val_mae: 0.1561 - val_mape: 63.2799
Epoch 14/60
634/635 [============================>.] - ETA: 0s - loss: 0.0129 - mse: 0.0129
- rmse: 0.1137 - mae: 0.0762 - mape: 43.7422
Epoch 14: val_loss did not improve from 0.05849
635/635 [==============================] - 72s 113ms/step - loss: 0.0129 - mse:
0.0129 - rmse: 0.1137 - mae: 0.0762 - mape: 43.7214 - val_loss: 0.0601 -
val_mse: 0.0601 - val_rmse: 0.2451 - val_mae: 0.1567 - val_mape: 58.1463
Epoch 15/60
634/635 [============================>.] - ETA: 0s - loss: 0.0118 - mse: 0.0118
```

```
- rmse: 0.1088 - mae: 0.0739 - mape: 43.2454
Epoch 15: val_loss did not improve from 0.05849
635/635 [==============================] - 72s 113ms/step - loss: 0.0118 - mse:
0.0118 - rmse: 0.1088 - mae: 0.0739 - mape: 43.2178 - val_loss: 0.0681 -
val_mse: 0.0681 - val_rmse: 0.2610 - val_mae: 0.1638 - val_mape: 59.8237
Epoch 16/60
634/635 [============================>.] - ETA: 0s - loss: 0.0209 - mse: 0.0209
- rmse: 0.1445 - mae: 0.0915 - mape: 52.1147
Epoch 16: val_loss did not improve from 0.05849
635/635 [==============================] - 72s 113ms/step - loss: 0.0209 - mse:
0.0209 - rmse: 0.1445 - mae: 0.0915 - mape: 52.0831 - val_loss: 0.0629 -
val_mse: 0.0629 - val_rmse: 0.2509 - val_mae: 0.1527 - val_mape: 51.7619
Epoch 17/60
634/635 [============================>.] - ETA: 0s - loss: 0.0119 - mse: 0.0119
- rmse: 0.1091 - mae: 0.0733 - mape: 43.4485
Epoch 17: val_loss did not improve from 0.05849
635/635 [==============================] - 71s 112ms/step - loss: 0.0119 - mse:
0.0119 - rmse: 0.1090 - mae: 0.0733 - mape: 43.4247 - val_loss: 0.0628 -
val_mse: 0.0628 - val_rmse: 0.2505 - val_mae: 0.1556 - val_mape: 52.8115
Epoch 18/60
634/635 [============================>.] - ETA: 0s - loss: 0.0134 - mse: 0.0134
- rmse: 0.1158 - mae: 0.0766 - mape: 43.6014
Epoch 18: val_loss did not improve from 0.05849
635/635 [==============================] - 72s 113ms/step - loss: 0.0134 - mse:
0.0134 - rmse: 0.1160 - mae: 0.0767 - mape: 43.5898 - val_loss: 0.0890 -
val_mse: 0.0890 - val_rmse: 0.2984 - val_mae: 0.1766 - val_mape: 57.7655
Epoch 19/60
634/635 [============================>.] - ETA: 0s - loss: 0.0213 - mse: 0.0213
- rmse: 0.1460 - mae: 0.0934 - mape: 55.5282
Epoch 19: val_loss did not improve from 0.05849
635/635 [==============================] - 72s 113ms/step - loss: 0.0213 - mse:
0.0213 - rmse: 0.1460 - mae: 0.0934 - mape: 55.4964 - val_loss: 0.0698 -
val_mse: 0.0698 - val_rmse: 0.2643 - val_mae: 0.1614 - val_mape: 55.8318
Epoch 19: early stopping
```

0.5 Evaluate model

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

```
20221103-230002
```

[ ]:
```
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 [==============================] - 32s 78ms/step - loss: 0.0540 - mse:
0.0540 - rmse: 0.2324 - mae: 0.1440 - mape: 68.1599
```

[ ]:
```
#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]
        stage_discharge_test_values = stage_discharge_test_values.numpy().
↪reshape(10, 2)
        predictions_values = predictions[:10]

        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 495ms/step
79.2600525051517
218.17216149864743
pred stage: 2.4097249507904053, actual stage: 2.31
pred discharge: 284.9043884277344, actual discharge: 210.0000000000001
```

```
pred stage: 2.463397979736328, actual stage: 2.41
pred discharge: 332.1559753417969, actual discharge: 210.0000000000001
pred stage: 3.3087246417999268, actual stage: 2.67
pred discharge: 1338.3272705078125, actual discharge: 210.0000000000001
pred stage: 2.3510382175445557, actual stage: 2.43
pred discharge: 257.7505187988281, actual discharge: 210.0000000000001
pred stage: 2.4264814853668213, actual stage: 2.42
pred discharge: 329.0776672363281, actual discharge: 210.0000000000001
pred stage: 2.7900302410125732, actual stage: 2.71
pred discharge: 804.0298461914062, actual discharge: 210.0000000000001
pred stage: 2.40978741645813, actual stage: 2.71
pred discharge: 290.5561828613281, actual discharge: 210.0000000000001
pred stage: 2.2910003662109375, actual stage: 2.76
pred discharge: 218.1156768798828, actual discharge: 210.0000000000001
pred stage: 2.5312607288360596, actual stage: 2.89
pred discharge: 354.0725402832031, actual discharge: 210.0000000000001
pred stage: 2.907449245452881, actual stage: 2.82
pred discharge: 1025.2611083984375, actual discharge: 210.0000000000001
```

0.5.1 Residual analysis

[ ]:
```
y_predictions = np.empty(shape=(1, 2)).astype('float32')
y_real = np.empty(shape=(1, 2)).astype('float32')

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()))
```

```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
```

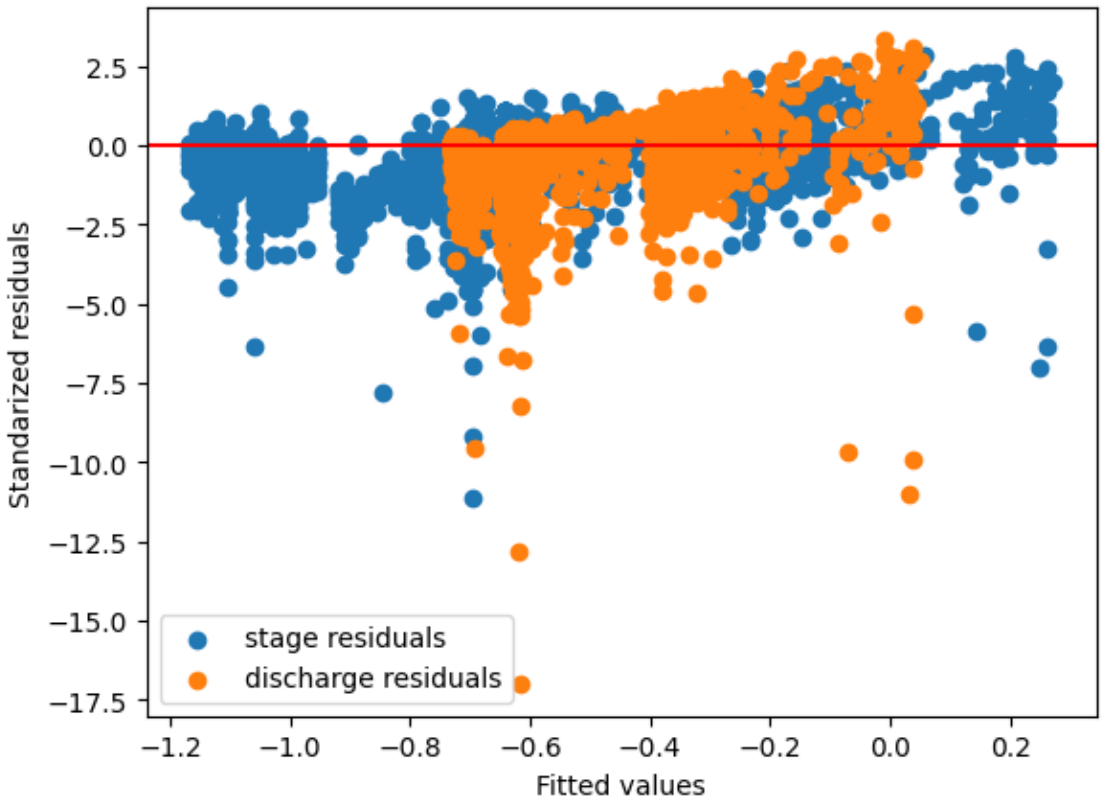
```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
```

```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
```

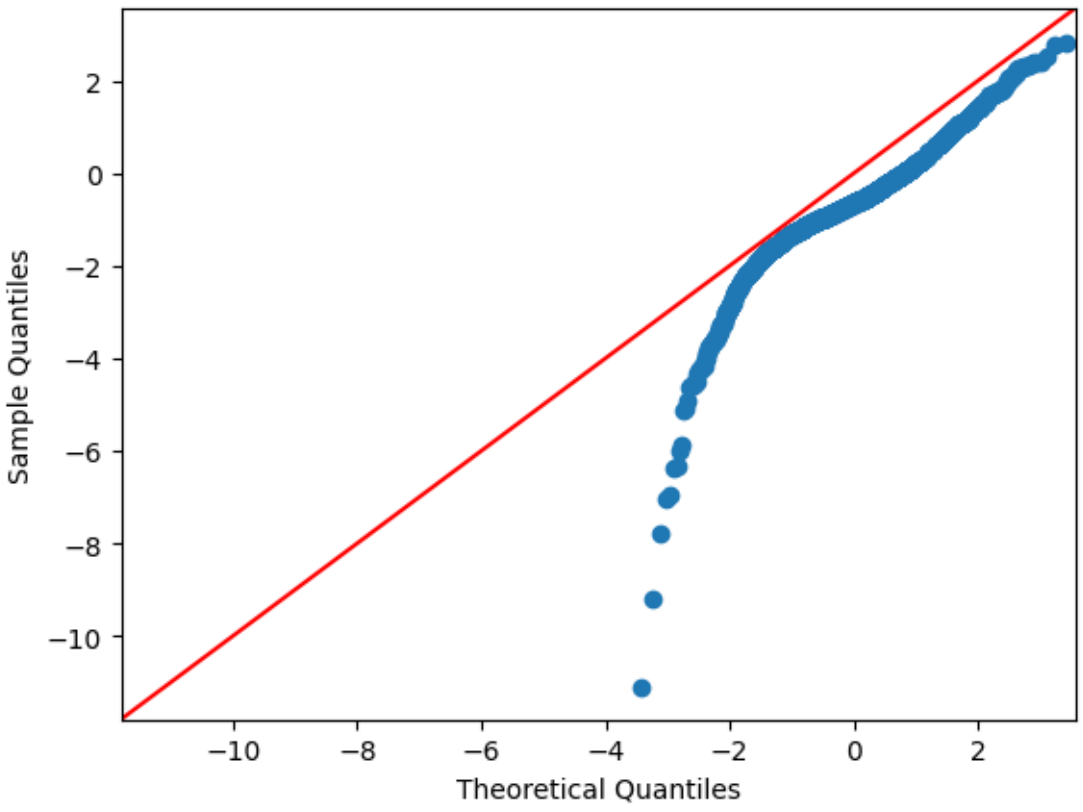
[ ]:
```
residuals = y_real - y_predictions
residuals_std = residuals/residuals.std()

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])

print(residual_stage.std())

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()
```

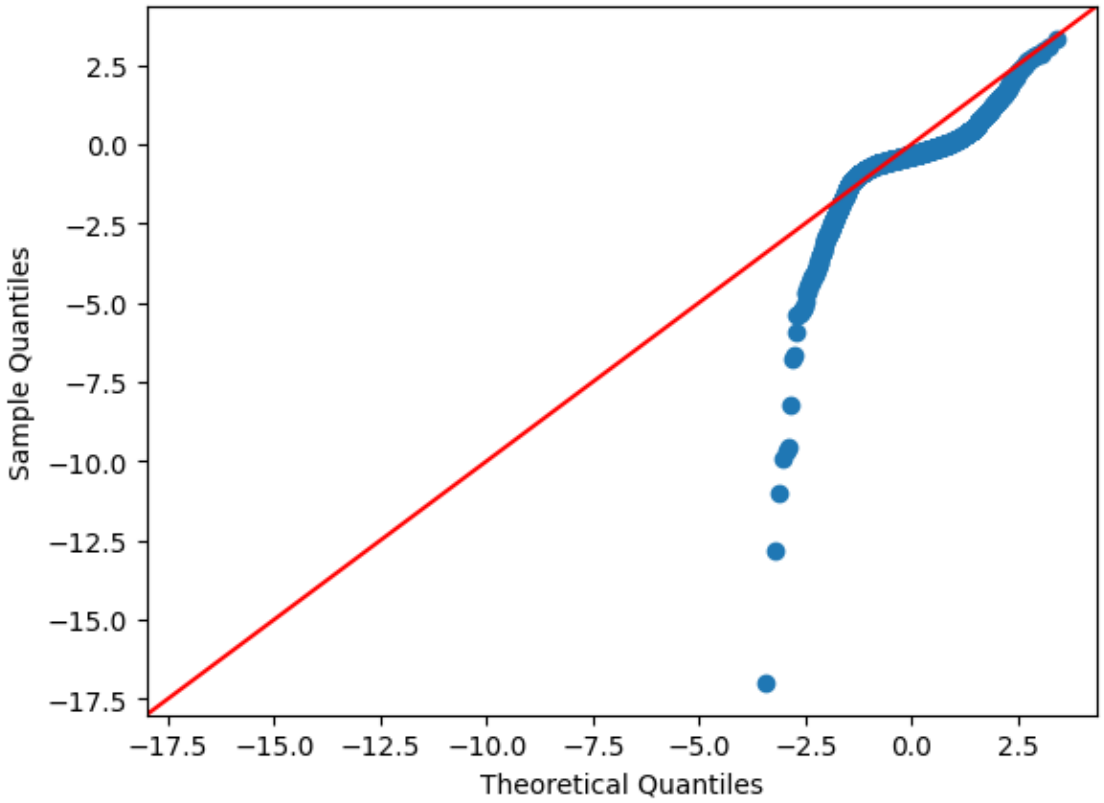
```
0.21593582108796294
```



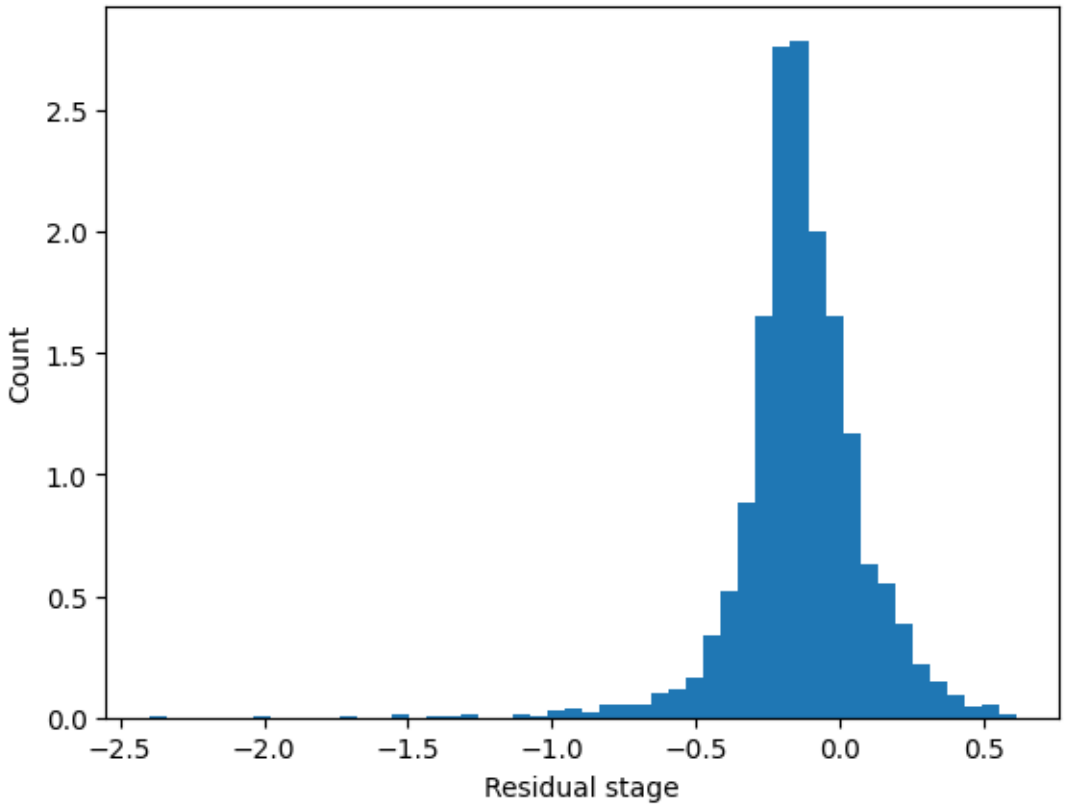
[ ]:
```
import statsmodels.api as sm
from statsmodels.stats.diagnostic import normal_ad

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



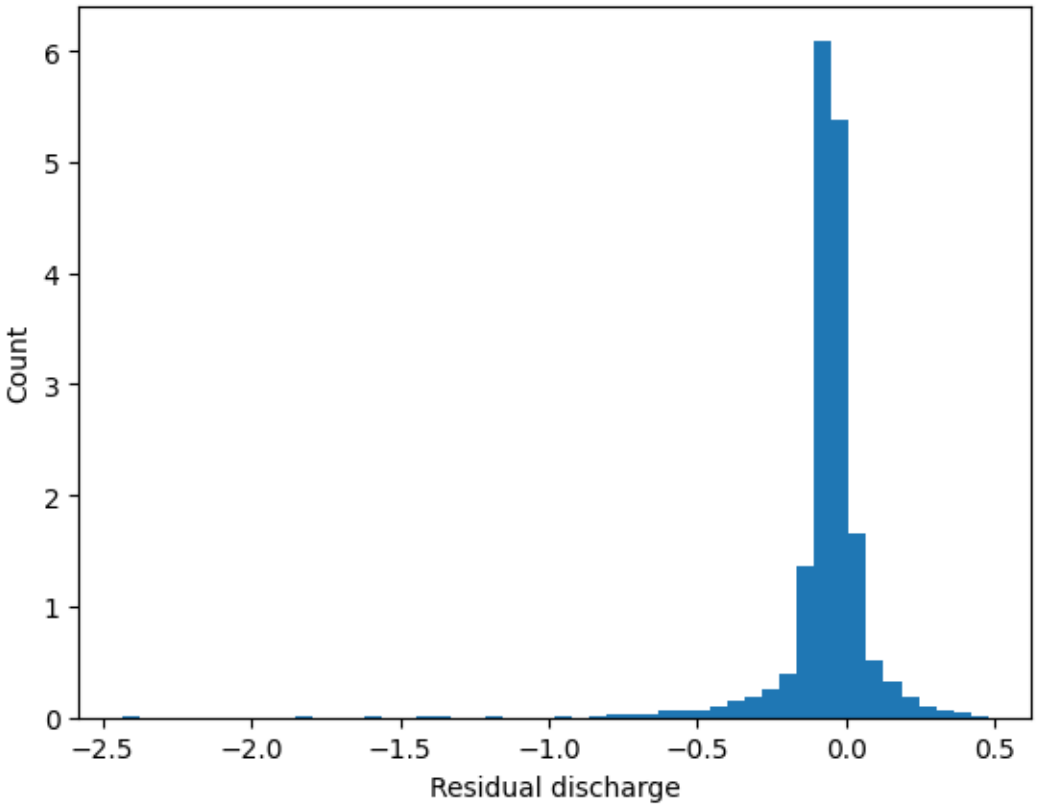
[ ]:
```
figure = sm.qqplot(residual_discharge / residual_discharge.std(), line='45',␣
↪label='discharge')
plt.show()
```



[ ]:
```
plt.hist(residual_stage, density=True, bins = 50)
plt.ylabel('Count')
plt.xlabel('Residual stage');
plt.show()
```



[ ]:
```
plt.hist(residual_discharge, density=True, bins = 50)
plt.ylabel('Count')
plt.xlabel('Residual discharge');
plt.show()
```



[ ]:
```
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.
```

[ ]:
```
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.
```

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
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [95], line 3
      1 layer_outputs = [layer.output for layer in best_model.layers[:12]]
      2 # Extracts the outputs of the top 12 layers
----> 3 activation_model = models.Model(inputs=best_model.input,␣
↪outputs=layer_outputs)

File ~/miniconda3/envs/tf-gpu/lib/python3.10/site-packages/tensorflow/python/
↪trackable/base.py:205, in no_automatic_dependency_tracking.<locals>.
↪_method_wrapper(self, *args, **kwargs)
    203 self._self_setattr_tracking = False  # pylint: disable=protected-access
    204 try:
--> 205   result = method(self, *args, **kwargs)
    206 finally:
    207   self._self_setattr_tracking = previous_value  # pylint:␣
↪disable=protected-access

File ~/miniconda3/envs/tf-gpu/lib/python3.10/site-packages/keras/engine/
↪functional.py:165, in Functional.__init__(self, inputs, outputs, name,␣
↪trainable, **kwargs)
    156     if not all(
    157         [
    158             functional_utils.is_input_keras_tensor(t)
    159             for t in tf.nest.flatten(inputs)
    160         ]
    161     ):
    162         inputs, outputs = functional_utils.clone_graph_nodes(
    163             inputs, outputs
    164         )
--> 165 self._init_graph_network(inputs, outputs)

File ~/miniconda3/envs/tf-gpu/lib/python3.10/site-packages/tensorflow/python/
↪trackable/base.py:205, in no_automatic_dependency_tracking.<locals>.
↪_method_wrapper(self, *args, **kwargs)
```

```
    203 self._self_setattr_tracking = False  # pylint: disable=protected-access
    204 try:
--> 205   result = method(self, *args, **kwargs)
    206 finally:
    207   self._self_setattr_tracking = previous_value  # pylint:␣
 ↪disable=protected-access

File ~/miniconda3/envs/tf-gpu/lib/python3.10/site-packages/keras/engine/
 ↪functional.py:264, in Functional._init_graph_network(self, inputs, outputs)
    261     self._input_coordinates.append((layer, node_index, tensor_index))
    263 # Keep track of the network's nodes and layers.
--> 264 nodes, nodes_by_depth, layers, _ = _map_graph_network(
    265     self.inputs, self.outputs
    266 )
    267 self._network_nodes = nodes
    268 self._nodes_by_depth = nodes_by_depth

File ~/miniconda3/envs/tf-gpu/lib/python3.10/site-packages/keras/engine/
 ↪functional.py:1128, in _map_graph_network(inputs, outputs)
   1126 for x in tf.nest.flatten(node.keras_inputs):
   1127     if id(x) not in computable_tensors:
-> 1128         raise ValueError(
   1129             f"Graph disconnected: cannot obtain value for "
   1130             f'tensor {x} at layer "{layer.name}". '
   1131             "The following previous layers were accessed "
   1132             f"without issue: {layers_with_complete_input}"
   1133         )
   1134 for x in tf.nest.flatten(node.outputs):
   1135     computable_tensors.add(id(x))

ValueError: Graph disconnected: cannot obtain value for tensor␣
 ↪KerasTensor(type_spec=TensorSpec(shape=(None, 224, 224, 3), dtype=tf.float32,
 ↪name='input_4'), name='input_4', description="created by layer 'input_4'") at
 ↪layer "conv1_pad". The following previous layers were accessed without issue:
 ↪[]
```

[ ]:
```
activations = activation_model.predict(test_ds.take(1))
```

```
Running cells with 'Python 3.9.13 ('tf-metal')' requires ipykernel package.

Run the following command to install 'ipykernel' into the Python environment.

Command: 'conda install -n tf-metal ipykernel --update-deps --force-reinstall'
```

[ ]:
```
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')
```

```
Running cells with 'Python 3.9.13 ('tf-metal')' requires ipykernel package.

Run the following command to install 'ipykernel' into the Python environment.
```

```
Command: 'conda install -n tf-metal ipykernel --update-deps --force-reinstall'
```