CompVis_Assignment2_MarcoFuchs

January 14, 2022

1 MSE Computer Vision - Assignment 2

1.1 Read image paths

[1]:
```
import os

TRAIN_PATH = 'data/train2/'
VAL_PATH = 'data/val/'
TST_PATH = 'data/test/'

def path_names_list(path, name_snipped):
    return sorted([os.path.join(path, fname)
                   for fname in os.listdir(path)
                   if fname.endswith(".png") and name_snipped in fname])

train_img_paths = path_names_list(TRAIN_PATH, "train_img")
train_lbl_paths = path_names_list(TRAIN_PATH, "train_lbl")
val_img_paths = path_names_list(VAL_PATH, "val_img")
val_lbl_paths = path_names_list(VAL_PATH, "val_lbl")
tst_img_paths = path_names_list(TST_PATH, "test_img")
tst_lbl_paths = path_names_list(TST_PATH, "test_lbl")


print("Length training data:", len(train_img_paths), len(train_lbl_paths))
for n, (img_path, lbl_path) in enumerate(zip(train_img_paths, train_lbl_paths)):
    print(f"{img_path} -> {lbl_path}")
    if n == 3:
        print()
        break

print("Length validation data:", len(val_img_paths), len(val_lbl_paths))
for n, (img_path, lbl_path) in enumerate(zip(val_img_paths, val_lbl_paths)):
    print(f"{img_path} -> {lbl_path}")
    if n == 3:
        print()
        break

print("Length test data:", len(tst_img_paths), len(tst_lbl_paths))
```

```
for n, (img_path, lbl_path) in enumerate(zip(tst_img_paths, tst_lbl_paths)):
    print(f"{img_path} -> {lbl_path}")
    if n == 3:
        break
```

```
Length training data: 23520 23520
data/train2/train_img_000_00.png -> data/train2/train_lbl_000_00.png
data/train2/train_img_000_01.png -> data/train2/train_lbl_000_01.png
data/train2/train_img_000_02.png -> data/train2/train_lbl_000_02.png
data/train2/train_img_000_03.png -> data/train2/train_lbl_000_03.png

Length validation data: 8544 8544
data/val/val_img_000_00.png -> data/val/val_lbl_000_00.png
data/val/val_img_000_01.png -> data/val/val_lbl_000_01.png
data/val/val_img_000_02.png -> data/val/val_lbl_000_02.png
data/val/val_img_000_03.png -> data/val/val_lbl_000_03.png

Length test data: 1888 1888
data/test/test_img_000_00.png -> data/test/test_lbl_000_00.png
data/test/test_img_000_01.png -> data/test/test_lbl_000_01.png
data/test/test_img_000_02.png -> data/test/test_lbl_000_02.png
data/test/test_img_000_03.png -> data/test/test_lbl_000_03.png
```

1.2 Dataloader

[27]:
```
import numpy as np
import tensorflow as tf
from sklearn.utils import shuffle

IMG_SHAPE = 128

def normalize(image, mask):
    image = tf.cast(image, tf.float32) / 255.0
    mask += 1 # to avoid negative labels
    return image, mask

def load_and_preprocess(img_filepath, mask_filepath):
    img = tf.io.read_file(img_filepath)
    img = tf.io.decode_jpeg(img, channels=3)
    img = tf.image.resize(img, [IMG_SHAPE, IMG_SHAPE])

    mask = tf.io.read_file(mask_filepath)
    mask = tf.io.decode_png(mask, channels=1)
    mask = tf.image.resize(mask, [IMG_SHAPE, IMG_SHAPE])

    img, mask = normalize(img, mask)

    return img, mask
```

```
AUTO = tf.data.experimental.AUTOTUNE
BATCH_SIZE = 32

# prepare data loaders
train_img_paths, train_lbl_paths = shuffle(train_img_paths, train_lbl_paths,␣
↪random_state=42)
val_img_paths, val_lbl_paths = shuffle(val_img_paths, val_lbl_paths,␣
↪random_state=42)
tst_img_paths, tst_lbl_paths = shuffle(tst_img_paths, tst_lbl_paths,␣
↪random_state=42)

trainloader = tf.data.Dataset.from_tensor_slices((train_img_paths,␣
↪train_lbl_paths))
valloader = tf.data.Dataset.from_tensor_slices((val_img_paths, val_lbl_paths))
tstloader = tf.data.Dataset.from_tensor_slices((tst_img_paths, tst_lbl_paths))


trainloader = (
    trainloader
    .shuffle(1024)
    .map(load_and_preprocess, num_parallel_calls=AUTO)
    .batch(BATCH_SIZE)
    .prefetch(AUTO) # prefetch next batch in a thread
)

valloader = (
    valloader
    .map(load_and_preprocess, num_parallel_calls=AUTO)
    .batch(BATCH_SIZE)
    .prefetch(AUTO)
)

tstloader = (
    tstloader
    .map(load_and_preprocess, num_parallel_calls=AUTO)
    .batch(BATCH_SIZE)
    .prefetch(AUTO)
)
```

[3]:
```
labels = [
    #  name                     id    trainId   category            catId     ␣
↪hasInstances   ignoreInEval   color
    [  'unlabeled'            ,  0 ,      255 , 'void'            , 0       ,␣
↪False        , True         , (  0,  0,  0) ],
```

```
    [  'ego vehicle'            ,  1 ,      255 , 'void'            , 0       ,␣
↪False        , True         , (  0,  0,  0) ],
    [  'rectification border' ,  2 ,      255 , 'void'            , 0       ,␣
↪False        , True         , (  0,  0,  0) ],
    [  'out of roi'           ,  3 ,      255 , 'void'            , 0       ,␣
↪False        , True         , (  0,  0,  0) ],
    [  'static'               ,  4 ,      255 , 'void'            , 0       ,␣
↪False        , True         , (  0,  0,  0) ],
    [  'dynamic'              ,  5 ,      255 , 'void'            , 0       ,␣
↪False        , True         , (111, 74,  0) ],
    [  'ground'               ,  6 ,      255 , 'void'            , 0       ,␣
↪False        , True         , ( 81,  0, 81) ],
    [  'road'                 ,  7 ,        0 , 'flat'            , 1       ,␣
↪False        , False        , (128, 64,128) ],
    [  'sidewalk'             ,  8 ,        1 , 'flat'            , 1       ,␣
↪False        , False        , (244, 35,232) ],
    [  'parking'              ,  9 ,      255 , 'flat'            , 1       ,␣
↪False        , True         , (250,170,160) ],
    [  'rail track'           , 10 ,      255 , 'flat'            , 1       ,␣
↪False        , True         , (230,150,140) ],
    [  'building'             , 11 ,        2 , 'construction'    , 2       ,␣
↪False        , False        , ( 70, 70, 70) ],
    [  'wall'                 , 12 ,        3 , 'construction'    , 2       ,␣
↪False        , False        , (102,102,156) ],
    [  'fence'                , 13 ,        4 , 'construction'    , 2       ,␣
↪False        , False        , (190,153,153) ],
    [  'guard rail'           , 14 ,      255 , 'construction'    , 2       ,␣
↪False        , True         , (180,165,180) ],
    [  'bridge'               , 15 ,      255 , 'construction'    , 2       ,␣
↪False        , True         , (150,100,100) ],
    [  'tunnel'               , 16 ,      255 , 'construction'    , 2       ,␣
↪False        , True         , (150,120, 90) ],
    [  'pole'                 , 17 ,        5 , 'object'          , 3       ,␣
↪False        , False        , (153,153,153) ],
    [  'polegroup'            , 18 ,      255 , 'object'          , 3       ,␣
↪False        , True         , (153,153,153) ],
    [  'traffic light'        , 19 ,        6 , 'object'          , 3       ,␣
↪False        , False        , (250,170, 30) ],
    [  'traffic sign'         , 20 ,        7 , 'object'          , 3       ,␣
↪False        , False        , (220,220,  0) ],
    [  'vegetation'           , 21 ,        8 , 'nature'          , 4       ,␣
↪False        , False        , (107,142, 35) ],
    [  'terrain'              , 22 ,        9 , 'nature'          , 4       ,␣
↪False        , False        , (152,251,152) ],
```

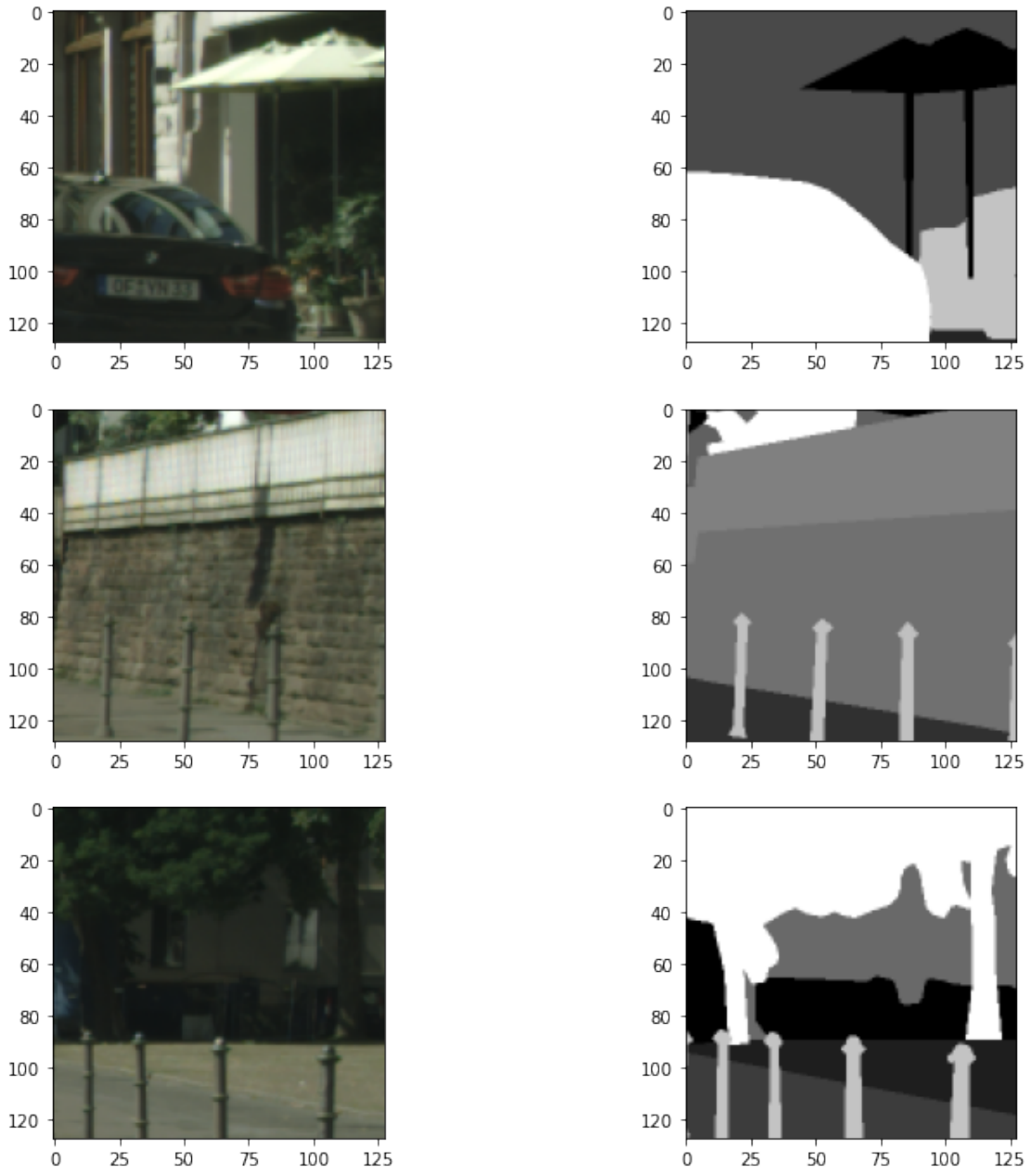
```
    [  'sky'                , 23 ,       10 , 'sky'            , 5       ,␣
 ↪False        , False        , ( 70,130,180) ],
    [  'person'             , 24 ,       11 , 'human'          , 6       ,␣
 ↪True         , False        , (220, 20, 60) ],
    [  'rider'              , 25 ,       12 , 'human'          , 6       ,␣
 ↪True         , False        , (255,  0,  0) ],
    [  'car'                , 26 ,       13 , 'vehicle'        , 7       ,␣
 ↪True         , False        , (  0,  0,142) ],
    [  'truck'              , 27 ,       14 , 'vehicle'        , 7       ,␣
 ↪True         , False        , (  0,  0, 70) ],
    [  'bus'                , 28 ,       15 , 'vehicle'        , 7       ,␣
 ↪True         , False        , (  0, 60,100) ],
    [  'caravan'            , 29 ,      255 , 'vehicle'        , 7       ,␣
 ↪True         , True         , (  0,  0, 90) ],
    [  'trailer'            , 30 ,      255 , 'vehicle'        , 7       ,␣
 ↪True         , True         , (  0,  0,110) ],
    [  'train'              , 31 ,       16 , 'vehicle'        , 7       ,␣
 ↪True         , False        , (  0, 80,100) ],
    [  'motorcycle'         , 32 ,       17 , 'vehicle'        , 7       ,␣
 ↪True         , False        , (  0,  0,230) ],
    [  'bicycle'            , 33 ,       18 , 'vehicle'        , 7       ,␣
 ↪True         , False        , (119, 11, 32) ],
    [  'license plate'      , -1 ,       -1 , 'vehicle'        , 7       ,␣
 ↪False        , True         , (  0,  0,142) ],
]

# segmentation_classes = [label[0] for label in labels]
SEGMENTATION_CLASSES_DICT = {n: label[0] for n, label in enumerate(labels) if␣
 ↪label[0] != "license plate"}
SEGMENTATION_CLASSES_DICT[-1] = "license plate"

OUTPUT_CHANNEL = len(SEGMENTATION_CLASSES_DICT)
print(f"There are {OUTPUT_CHANNEL} segmentatin classes.")
print(SEGMENTATION_CLASSES_DICT)
```

```
There are 35 segmentatin classes.
{0: 'unlabeled', 1: 'ego vehicle', 2: 'rectification border', 3: 'out of roi',
4: 'static', 5: 'dynamic', 6: 'ground', 7: 'road', 8: 'sidewalk', 9: 'parking',
10: 'rail track', 11: 'building', 12: 'wall', 13: 'fence', 14: 'guard rail', 15:
'bridge', 16: 'tunnel', 17: 'pole', 18: 'polegroup', 19: 'traffic light', 20:
'traffic sign', 21: 'vegetation', 22: 'terrain', 23: 'sky', 24: 'person', 25:
'rider', 26: 'car', 27: 'truck', 28: 'bus', 29: 'caravan', 30: 'trailer', 31:
'train', 32: 'motorcycle', 33: 'bicycle', -1: 'license plate'}
```

1.3 Visualize

[4]:
```
import matplotlib.pyplot as plt

num_example_to_display = 3
img, mask = next(iter(valloader))
fig, axs = plt.subplots(nrows=num_example_to_display, ncols=2,␣
 ↪figsize=(12,4*num_example_to_display))

for i in range(num_example_to_display):
    axs[i][0].imshow(img[i])
    axs[i][1].imshow(np.squeeze(mask[i],-1), cmap='gray');
```

```
2022-01-13 09:42:10.813714: I
tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR
Optimization Passes are enabled (registered 2)
```



1.4 Model

[5]:
```
from tensorflow.keras.layers import *
from tensorflow.keras.models import *
import keras
```

[6]:
```
def fcn_simple_no_border(input_height:int, input_width:int) -> keras.Model:
    """
```

```
    Create a simple fcn model for semantic segmentation with 2 classes
    """
    model = keras.Sequential()

    # we use grayscale (1-channel input)

    # (used to define input shape on the first layers)
    model.add(keras.layers.Layer(input_shape=(input_height , input_width, 3)))

    # add 3 convolutional layers with 3x3 filters
    model.add(keras.layers.Convolution2D(filters=4, kernel_size=3, strides=(2,␣
↪2), padding='same', activation='relu'))
    model.add(keras.layers.Conv2DTranspose(filters=4, kernel_size=3,␣
↪strides=(2, 2), padding='same', activation='relu'))
    model.add(keras.layers.Convolution2D(filters=4, kernel_size=3,␣
↪padding='same', activation='relu'))

    # go to logits which is the number of classes and add sigmoid layer for␣
↪activation
    model.add(keras.layers.Convolution2D(filters=1, kernel_size=1,␣
↪activation=None,
                                         kernel_initializer=keras.initializers.
↪TruncatedNormal(mean=0.0, stddev=0.001, seed=None)))
    model.add(keras.layers.Activation('sigmoid'))

    # reshape so that we have a sample for each pixel
    model.add(keras.layers.Reshape(target_shape=(input_height, input_width, 1)))

    return model
```

[7]:
```
def model_u(img_shape, output_channels):
    model = keras.Sequential()
    model.add(keras.layers.Layer(input_shape=(img_shape , img_shape, 3)))

    # down
    model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',␣
↪padding='same'))
    model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',␣
↪padding='same'))
    model.add(MaxPooling2D((2, 2)))

    model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu',␣
↪padding='same'))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu',␣
↪padding='same'))
    model.add(MaxPooling2D((2, 2)))
```

```
    model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
    model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
    model.add(MaxPooling2D((2, 2)))

#     model.add(Conv2D(filters=256, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
#     model.add(Conv2D(filters=256, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
#     model.add(MaxPooling2D((2, 2)))

    # up
#     model.add(Conv2D(filters=512, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
#     model.add(Conv2D(filters=512, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
#     model.add(Conv2DTranspose(filters=256, kernel_size=(2,2), strides=(2,2),␣
 ↪padding='same'))

    model.add(Conv2D(filters=256, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
    model.add(Conv2D(filters=256, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
    model.add(Conv2DTranspose(filters=128, kernel_size=(2,2), strides=(2,2),␣
 ↪padding='same'))

    model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
    model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
    model.add(Conv2DTranspose(filters=64, kernel_size=(2,2), strides=(2,2),␣
 ↪padding='same'))

    model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu',␣
 ↪padding='same'))
    model.add(Conv2DTranspose(filters=32, kernel_size=(2,2), strides=(2,2),␣
 ↪padding='same'))

    model.add(Conv2D(output_channels, (3, 3), activation='softmax',␣
 ↪padding='same'))

    return model
```

1.5 Compile Model

[8]:
```
tf.keras.backend.clear_session()
# model = SegmentationModel().prepare_model(OUTPUT_CHANNEL)
model = model_u(IMG_SHAPE, OUTPUT_CHANNEL)
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy")

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer (Layer)                (None, 128, 128, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 128, 128, 32)      896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 32)      9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 128)       73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 32, 32, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 16, 16, 256)       295168
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 16, 16, 256)       590080
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 32, 32, 128)       131200
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 32, 32, 128)       147584
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 32, 32, 128)       147584
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 64, 64, 64)        32832
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 64, 64, 64)        36928
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 64, 64, 64)        36928
```

```
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 128, 128, 32)      8224
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 128, 128, 35)      10115
=================================================================
Total params: 1,723,651
Trainable params: 1,723,651
Non-trainable params: 0
_________________________________________________________________
```

1.6 Callbacks

[9]:
```
from datetime import date

today = date.today()
d1 = today.strftime("%d/%m/%Y") # dd/mm/YY

# return dictionary with segmentation classes (key->number, value->name)
def labels():
    return SEGMENTATION_CLASSES_DICT

# util function for generating interactive image mask from components
def wandb_mask(bg_img, pred_mask, true_mask):
  return wandb.Image(bg_img, masks={
      "prediction" : {
          "mask_data" : pred_mask,
          "class_labels" : labels()
      },
      "ground truth" : {
          "mask_data" : true_mask,
          "class_labels" : labels()
      }
    }
  )

# early stopping callback
early_stopping_callback = tf.keras.callbacks.EarlyStopping(monitor='loss',␣
 ↪patience=3)
# get epochs outputs
output_epoch_callback = tf.keras.callbacks.ModelCheckpoint(filepath='model.
 ↪{epoch:02d}-{val_loss:.2f}.h5')
# always safe best model
model_checkpoint_callback = keras.callbacks.ModelCheckpoint(f"models/
 ↪2022-01-05_best_model.h5",save_best_only=True)
```

1.7 Train

[10]:
```
import wandb
from wandb.keras import WandbCallback

!wandb login
```

```
wandb: W&B API key is configured (use `wandb login --relogin` to
force relogin)
```

[11]:
```
wandb.init(project='image-segmentation', reinit=True)

EPOCHS = 25

history = model.fit(trainloader,
                epochs=EPOCHS,
                validation_data=valloader,
                callbacks=[early_stopping_callback, WandbCallback(),␣
 ↪output_epoch_callback, model_checkpoint_callback])

model.save(f'models/')
wandb.finish()
```

```
wandb: W&B API key is configured (use `wandb login --relogin` to
force relogin)

<IPython.core.display.HTML object>

Epoch 1/25
735/735 [==============================] - 2003s 3s/step - loss: 1.8858 -
val_loss: 1.5484
Epoch 2/25
735/735 [==============================] - 2095s 3s/step - loss: 1.3117 -
val_loss: 1.1279
Epoch 3/25
735/735 [==============================] - 1932s 3s/step - loss: 1.0578 -
val_loss: 0.9751
Epoch 4/25
735/735 [==============================] - 1923s 3s/step - loss: 0.9475 -
val_loss: 0.9847
Epoch 5/25
735/735 [==============================] - 1704s 2s/step - loss: 0.8785 -
val_loss: 0.9123
Epoch 6/25
735/735 [==============================] - 5324s 7s/step - loss: 0.8299 -
val_loss: 0.8680
Epoch 7/25
735/735 [==============================] - 3293s 4s/step - loss: 0.7928 -
val_loss: 0.8034
Epoch 8/25
```

```
735/735 [==============================] - 1838s 3s/step - loss: 0.7466 -
val_loss: 0.8009
Epoch 9/25
735/735 [==============================] - 1884s 3s/step - loss: 0.7298 -
val_loss: 0.7682
Epoch 10/25
735/735 [==============================] - 1851s 3s/step - loss: 0.7017 -
val_loss: 0.8202
Epoch 11/25
735/735 [==============================] - 1881s 3s/step - loss: 0.6793 -
val_loss: 0.7879
Epoch 12/25
735/735 [==============================] - 1886s 3s/step - loss: 0.6585 -
val_loss: 0.7472
Epoch 13/25
735/735 [==============================] - 1902s 3s/step - loss: 0.6311 -
val_loss: 0.7160
Epoch 14/25
735/735 [==============================] - 1874s 3s/step - loss: 0.6161 -
val_loss: 0.7415
Epoch 15/25
735/735 [==============================] - 1738s 2s/step - loss: 0.5949 -
val_loss: 0.7524
Epoch 16/25
735/735 [==============================] - 6412s 9s/step - loss: 0.5871 -
val_loss: 0.7277
Epoch 17/25
735/735 [==============================] - 5746s 8s/step - loss: 0.5685 -
val_loss: 0.7800
Epoch 18/25
735/735 [==============================] - 4961s 7s/step - loss: 0.5516 -
val_loss: 0.7244
Epoch 19/25
735/735 [==============================] - 9950s 14s/step - loss: 0.5448 -
val_loss: 0.7226
Epoch 20/25
735/735 [==============================] - 11476s 16s/step - loss: 0.5220 -
val_loss: 0.7511
Epoch 21/25
593/735 [======================>...] - ETA: 20:30 - loss: 0.5193

wandb: Network error (ConnectionError), entering retry loop.

735/735 [==============================] - 7164s 10s/step - loss: 0.5160 -
val_loss: 0.6998
Epoch 22/25
 72/735 [=>...] - ETA: 23:10 - loss: 0.4829
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/var/folders/kh/8twngknj0j9130x9cc9xlqdm0000gn/T/ipykernel_69564/2536146075.py␣
 ↪in <module>
      3 EPOCHS = 25
      4
----> 5 history = model.fit(trainloader,
      6                 epochs=EPOCHS,
      7                 validation_data=valloader,

/opt/anaconda3/envs/compvis/lib/python3.8/site-packages/wandb/integration/keras
 ↪keras.py in new_v2(*args, **kwargs)
    165             for cbk in cbks:
    166                 set_wandb_attrs(cbk, val_data)
--> 167         return old_v2(*args, **kwargs)
    168
    169     training_arrays.orig_fit_loop = old_arrays

/opt/anaconda3/envs/compvis/lib/python3.8/site-packages/keras/engine/training.p␣
 ↪in fit(self, x, y, batch_size, epochs, verbose, callbacks, validation_split,␣
 ↪validation_data, shuffle, class_weight, sample_weight, initial_epoch,␣
 ↪steps_per_epoch, validation_steps, validation_batch_size, validation_freq,␣
 ↪max_queue_size, workers, use_multiprocessing)
   1182                 _r=1):
   1183               callbacks.on_train_batch_begin(step)
-> 1184               tmp_logs = self.train_function(iterator)
   1185               if data_handler.should_sync:
   1186                 context.async_wait()

/opt/anaconda3/envs/compvis/lib/python3.8/site-packages/tensorflow/python/eager
 ↪def_function.py in __call__(self, *args, **kwds)
    883
    884       with OptionalXlaContext(self._jit_compile):
--> 885         result = self._call(*args, **kwds)
    886
    887       new_tracing_count = self.experimental_get_tracing_count()

/opt/anaconda3/envs/compvis/lib/python3.8/site-packages/tensorflow/python/eager
 ↪def_function.py in _call(self, *args, **kwds)
    915       # In this case we have created variables on the first call, so we
 ↪run the
    916       # defunned version which is guaranteed to never create variables.
--> 917       return self._stateless_fn(*args, **kwds)  # pylint:␣
 ↪disable=not-callable
    918     elif self._stateful_fn is not None:
    919       # Release the lock early so that multiple threads can perform the
 ↪call
```

```
/opt/anaconda3/envs/compvis/lib/python3.8/site-packages/tensorflow/python/eager
↪function.py in __call__(self, *args, **kwargs)
   3037       (graph_function,
   3038        filtered_flat_args) = self._maybe_define_function(args, kwargs)
-> 3039     return graph_function._call_flat(
   3040         filtered_flat_args, captured_inputs=graph_function.
↪captured_inputs)  # pylint: disable=protected-access
   3041

/opt/anaconda3/envs/compvis/lib/python3.8/site-packages/tensorflow/python/eager
↪function.py in _call_flat(self, args, captured_inputs, cancellation_manager)
   1961         and executing_eagerly):
   1962       # No tape is watching; skip to running the function.
-> 1963       return self._build_call_outputs(self._inference_function.call(
   1964           ctx, args, cancellation_manager=cancellation_manager))
   1965     forward_backward = self._select_forward_and_backward_functions(

/opt/anaconda3/envs/compvis/lib/python3.8/site-packages/tensorflow/python/eager
↪function.py in call(self, ctx, args, cancellation_manager)
    589       with _InterpolateFunctionError(self):
    590         if cancellation_manager is None:
--> 591           outputs = execute.execute(
    592               str(self.signature.name),
    593               num_outputs=self._num_outputs,

/opt/anaconda3/envs/compvis/lib/python3.8/site-packages/tensorflow/python/eager
↪execute.py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     57   try:
     58     ctx.ensure_initialized()
---> 59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,␣
↪op_name,
     60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

1.8 Prediction

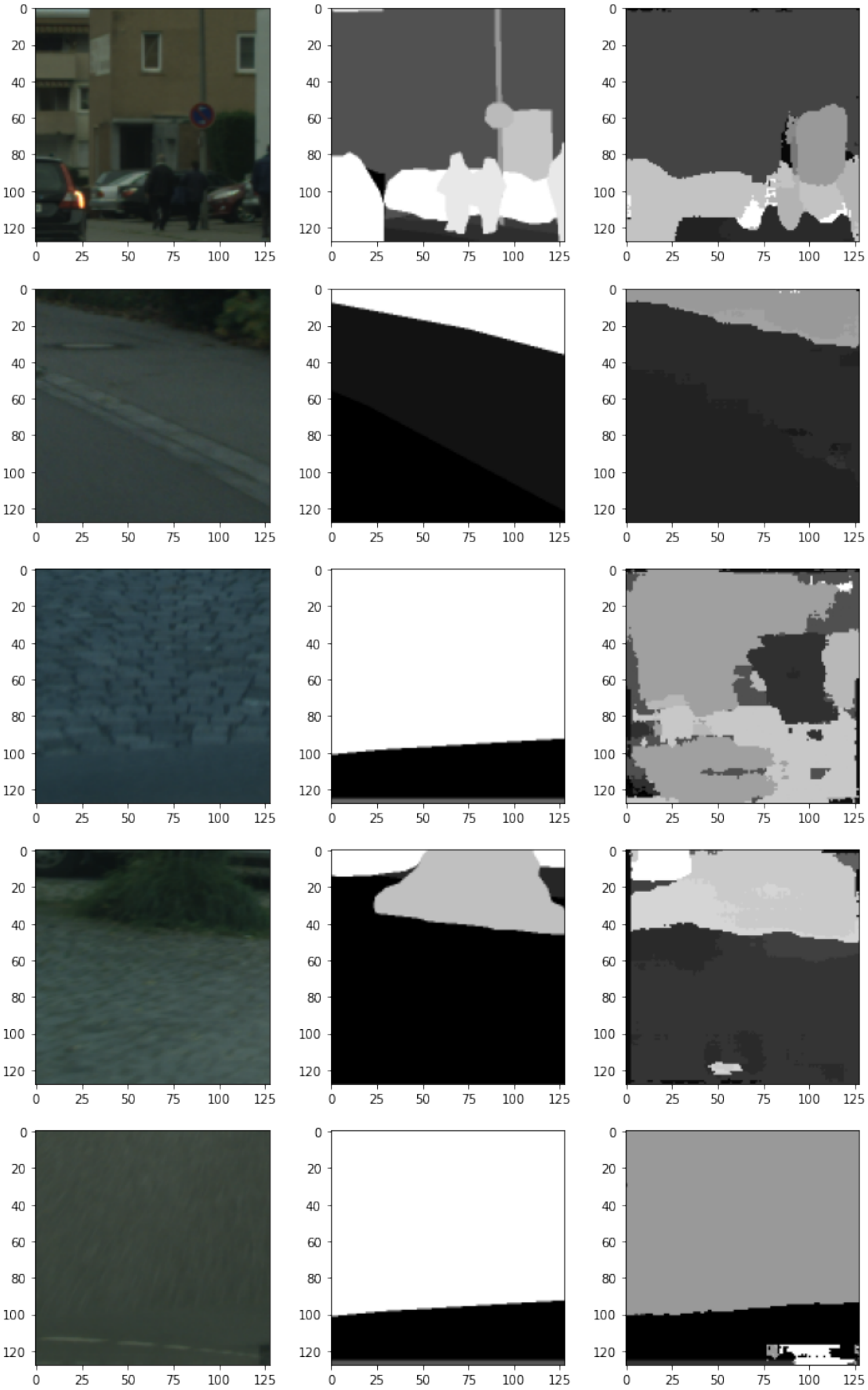
[13]:
```
from pycm import *

def confusion_matrix(pred_mask, true_mask):

    # 2D -> 1D Array
    pred = pred_mask.ravel()
```

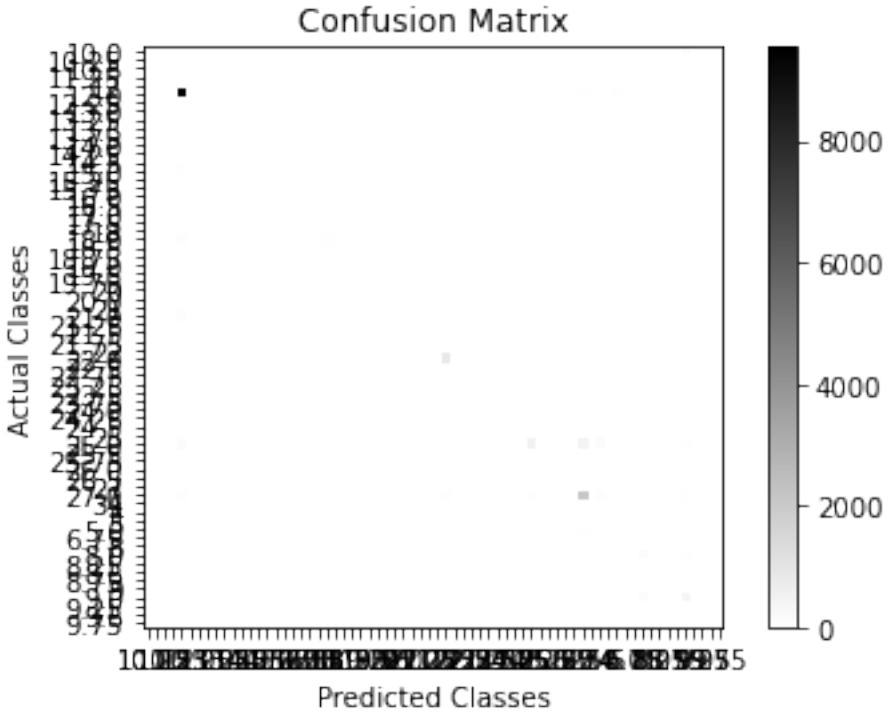
```
    true = true_mask.ravel()

    # Create Classification metrics and plot confusion matrix
    cm = ConfusionMatrix(actual_vector=true, predict_vector=pred)
    cm.plot()

def calc_accuracy(pred_mask, true_mask):
    # Count correct and wrong prediction
    true_prediction = np.count_nonzero((pred_mask == true_mask))
    false_prediction = np.count_nonzero((pred_mask != true_mask))
    accuracy_percentage = true_prediction*100/(false_prediction +␣
 ↪true_prediction)
#     print(f"Accuracy: {true_prediction}/{false_prediction+true_prediction} =␣
 ↪{accuracy_percentage:.1f}%")
    return accuracy_percentage
```

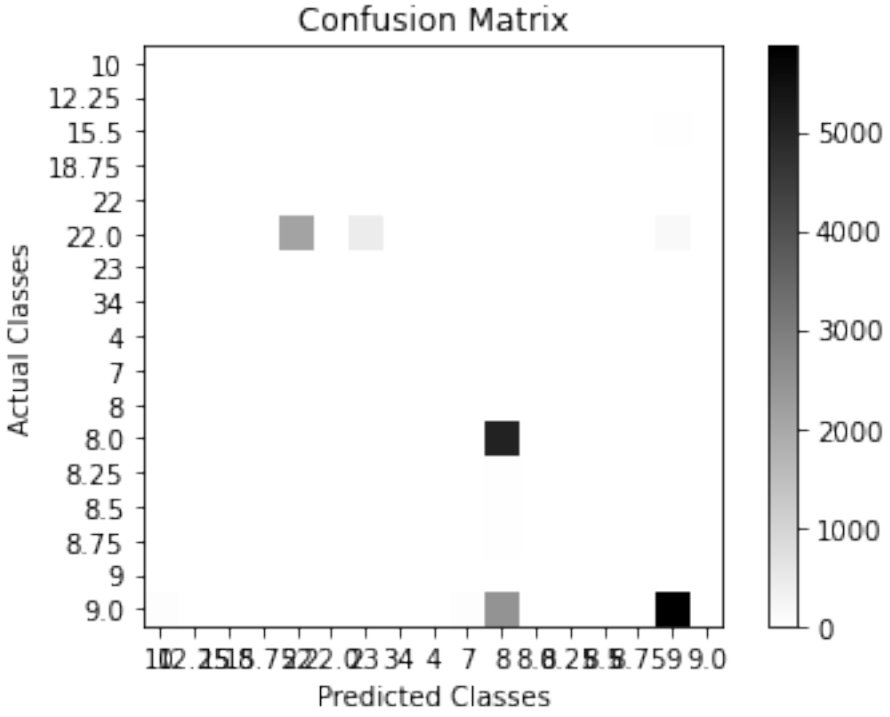
[14]:
```
# Load model, if already trained
from keras.models import load_model

# model_ = load_model('model.15-0.76.h5')
model = load_model('models/2022-01-05_best_model.h5')
```

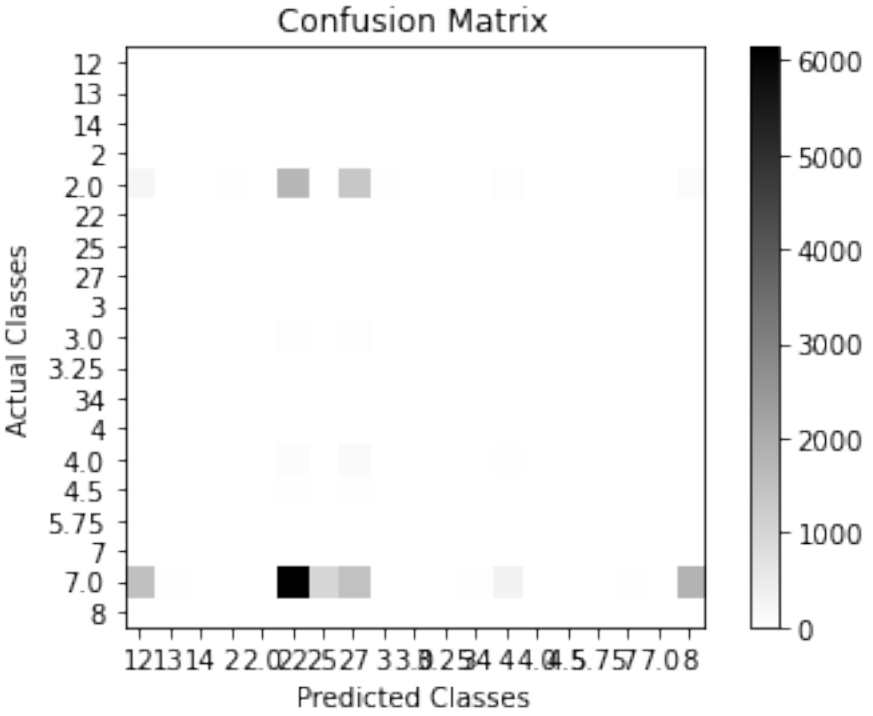
[28]:
```
val_img, val_mask = next(iter(tstloader))

pred_mask = model.predict(val_img)
pred_mask = np.argmax(pred_mask, axis=-1)
pred_mask = np.expand_dims(pred_mask, axis=-1)

num_accuracy = 29
accuracy = np.mean([calc_accuracy(np.squeeze(pred_mask[i],-1), np.
 ↪squeeze(val_mask[i],-1)) for i in range(num_accuracy)])
print(f"Accuracy for {num_accuracy} test images: {accuracy:.1f} %")

num_example_to_display = 5
fig, axs = plt.subplots(nrows=num_example_to_display, ncols=3,␣
 ↪figsize=(12,4*num_example_to_display))
for i in range(num_example_to_display):
    confusion_matrix(np.squeeze(pred_mask[i],-1), np.squeeze(val_mask[i],-1))
    axs[i][0].imshow(val_img[i]);
    axs[i][1].imshow(np.squeeze(val_mask[i],-1), cmap='gray')
    axs[i][2].imshow(np.squeeze(pred_mask[i],-1), cmap='gray')
```

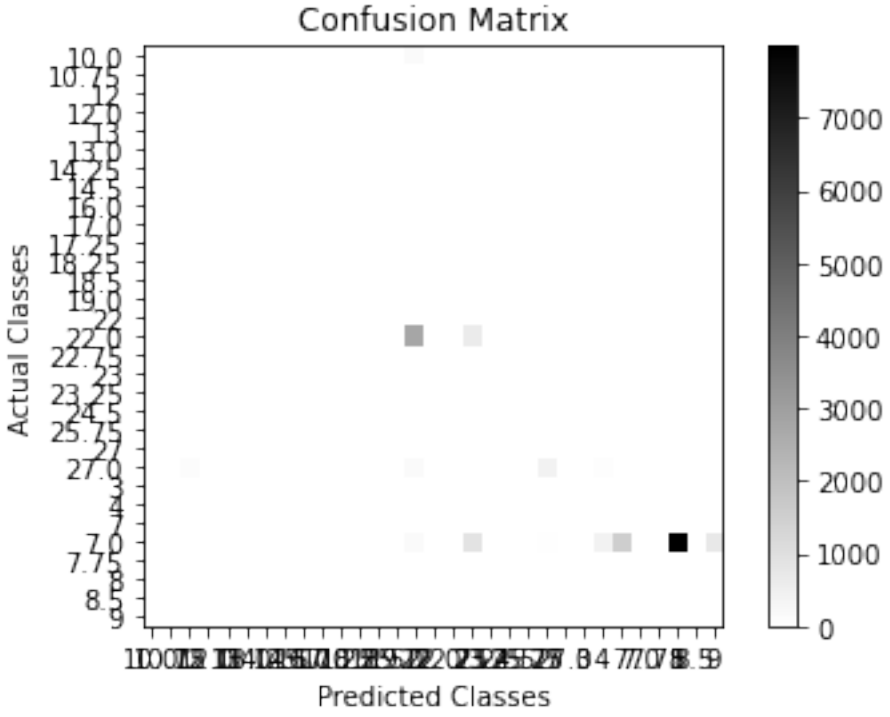
```
Accuracy for 29 test images: 60.8 %
```

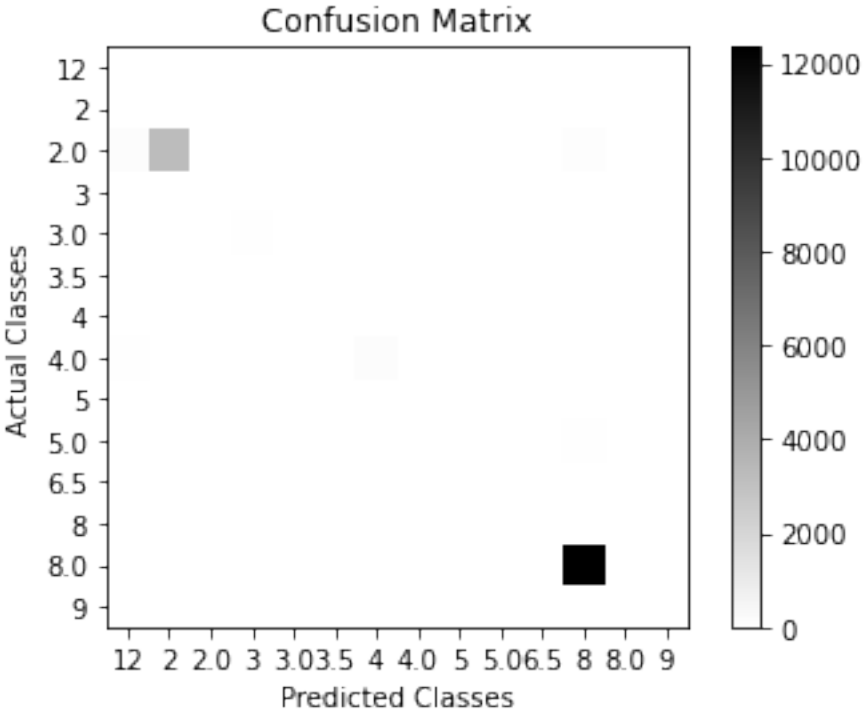



Confusion Matrix
Actual Classes
Predicted Classes
8000
6000
4000
2000
0


Confusion Matrix
Actual Classes
10
12.25
15.5
18.75
22
22.0
23
34
4
7
8
8.0
8.25
8.5
8.75
9
9.0
Predicted Classes
5000
4000
3000
2000
1000
0


Confusion Matrix
Actual Classes
Predicted Classes
6000
5000
4000
3000
2000
1000
0


Confusion Matrix
Actual Classes
Predicted Classes
7000
6000
5000
4000
3000
2000
1000
0


Confusion Matrix
Actual Classes
Predicted Classes
12
2
2.0
3
3.0
3.5
4
4.0
5
5.0
6.5
8
8.0
9
12000
10000
8000
6000
4000
2000
0