40 lines - 37 Removals

```
import torch

from binaryLoader import BinaryDataset, pad_collate
_func


def load_dataset(batch_size, max_seq_len=256, shuff
le=True, num_workers=0):
    train_set = BinaryDataset(good_dir='/data1/embe
r2018/train/benign/',
                              bad_dir='/data1/ember
2018/train/malware/',
                              max_len=max_seq_len)


    train_loader = torch.utils.data.DataLoader(trai
n_set,
                                               batc
h_size=batch_size,
                                               shuf
fle=shuffle,
                                               num_
workers=num_workers,
                                               coll
ate_fn=pad_collate_func,
                                               drop
_last=True)

    test_set = BinaryDataset(good_dir='/data1/ember
2018/test/benign/',
                             bad_dir='/data1/ember2
018/test/malware/',
                             max_len=max_seq_len)

    test_loader = torch.utils.data.DataLoader(test_
set,
                                              batch
_size=batch_size,
                                              shuff
le=False,
                                              num_w
orkers=num_workers,
                                              colla
te_fn=pad_collate_func,
                                              drop_
last=True)

    return train_loader, test_loader


```

124 lines + 121 Additions

```
import datetime

import torch

from initializations import *
from binaryLoader import BinaryDataset, pad_collate
_func


def printf(message):
    timestamp = datetime.datetime.now()
    timestamp_str = timestamp.strftime("%Y-%m-%d %
H:%M:%S")
    print(f"[{timestamp_str}] {message}")


def load_dataset(batch_size, max_seq_len=256, shuff
le=True, num_workers=0):
    if TRAIN:

        train_set = BinaryDataset(

            good_dir=TRAIN_DATA_PATH_GOOD,
            bad_dir=TRAIN_DATA_PATH_BAD,
            max_len=max_seq_len)

        train_loader = torch.utils.data.DataLoader
(train_set,
                                                batch_size=batch_size,
                                                shuffle=shuffle,
                                                num_workers=num_workers,
                                                collate_fn=pad_collate_func,
                                                drop_last=True)

        validation_set = BinaryDataset(

            good_dir=VALIDATION_DATA_PATH_GOOD,

            bad_dir=VALIDATION_DATA_PATH_BAD,
            max_len=max_seq_len)

        validation_loader = torch.utils.data.DataLo
ader(validation_set,
                                                   batch_size=batch_size,
                                                   shuffle=shuffle,
                                                   num_workers=num_workers,
                                                   collate_fn=pad_collate_func,
                                                   drop_last=True)

        return train_loader, validation_loader, len
(train_set), len(validation_set)

    elif TEST:
```

```
30
31  if __name__ == '__main__':
32      train, test = load_dataset(batch_size=32, max_seq_len=20, shuffle=True, num_workers=0)
33
34      for x, y in train:
35          print(type(x))
36          print(x.shape)
37          print(y.shape)
38          print(x)
39          print(y)
40          break
```

```
44          # Testing: load test data set only
45          test_set = BinaryDataset(
46              good_dir=TEST_DATA_PATH_GOOD,
47              bad_dir=TEST_DATA_PATH_BAD,
48              max_len=max_seq_len)
49
50          test_loader = torch.utils.data.DataLoader(test_set,
51                                                    batch_size=batch_size,
52                                                    shuffle=shuffle,
53                                                    num_workers=num_workers,
54                                                    collate_fn=pad_collate_func,
55                                                    drop_last=True)
56
57          return test_loader, len(test_set)
58
59      elif PREDICT:
60          test_set = BinaryDataset(
61              good_dir=TEST_DATA_PATH_GOOD + "temp",
62              bad_dir=TEST_DATA_PATH_BAD + "temp",
63              max_len=max_seq_len)
64
65          test_loader = torch.utils.data.DataLoader(test_set,
66                                                    batch_size=batch_size,
67                                                    shuffle=shuffle,
68                                                    num_workers=num_workers,
69                                                    collate_fn=pad_collate_func,
70                                                    drop_last=True)
71
72          return test_loader, len(test_set)
73
74
75  try:
76      if TRAIN:
77          printf(f"Parameter selected: TRAIN")
78          train, validation, num_train_samples, num_validation_samples = load_dataset(batch_size=256, max_seq_len=256,
79                                                                                      shuffle=True, num_workers=0)
80
81          printf(f"Number of training samples: {num_train_samples}")
82          printf(f"Number of validation samples: {num_validation_samples}\n")
83
84          printf("Printing training set data...")
85          for x, y in train:
86              printf(type(x))
87              printf(x.shape)
88              printf(y.shape)
89              printf(x)
90              printf(y)
91              break
92
93          print()
```

```
        printf("Printing validation set data...")
        for x, y in validation:
            printf(type(x))
            printf(x.shape)
            printf(y.shape)
            printf(x)
            printf(y)
            break

    elif TEST:

        printf(f"Parameter selected: TEST")
        test, num_test_samples = load_dataset(batch_size=256, max_seq_len=256, shuffle=True,
                                              num_workers=0)

        printf("Printing test set data...")
        for x, y in test:
            printf(type(x))
            printf(x.shape)
            printf(y.shape)
            printf(x)
            printf(y)
            break

except Exception as ex:
    if ex.args[0] == 'num_samples should be a positive integer value, but got num_samples=0':
        printf("No file or folder found for one or all of the specified dataset paths")
        raise Exception(FileNotFoundError)
    else:
        raise ex

```