model

March 31, 2022

[1]:
```
import sys
sys.path.append('G:/wenet_location/wenet/')
import paddle
from read_fbank import init_data_list
from paddle.io import Dataset, DataLoader
import numpy as np
```

```
D:\Miniconda3\envs\wenet\lib\site-packages\numpy\_distributor_init.py:30:
UserWarning: loaded more than 1 DLL from .libs:
D:\Miniconda3\envs\wenet\lib\site-
packages\numpy\.libs\libopenblas.EL2C6PLE4ZYW3ECEVIV3OXXGRN2NRFM2.gfortran-
win_amd64.dll
D:\Miniconda3\envs\wenet\lib\site-
packages\numpy\.libs\libopenblas.QVLO2T66WEPI7JZ63PS3HMOHFEY472BC.gfortran-
win_amd64.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
D:\Miniconda3\envs\wenet\lib\site-packages\torchaudio\backend\utils.py:67:
UserWarning: No audio backend is available.
  warnings.warn('No audio backend is available.')
```

[2]:
```
data_list = init_data_list()
```

[3]:
```
print("  :{}".format(len(data_list)))
for index, data in enumerate(data_list):
    if index > 5 :
        break
    print("idx={}, shape={}, label={}".format(index, data[0].shape, data[1]))
```

```
  :2041
idx=0, shape=(16, 80), label=1
idx=1, shape=(13, 80), label=1
idx=2, shape=(16, 80), label=1
idx=3, shape=(17, 80), label=1
idx=4, shape=(16, 80), label=1
idx=5, shape=(25, 80), label=1
```

[4]:
```
print(min(data[0].shape[0] for data in data_list))
print(max(data[0].shape[0] for data in data_list))
```

```
10
223
```

[5]:
```python
NUM_SAMPLES=len(data_list)
BATCH_SIZE = 64
BATCH_NUM = NUM_SAMPLES // BATCH_SIZE

train_offset = int(NUM_SAMPLES * 0.6)
val_offset = int(NUM_SAMPLES * 0.8)
print(train_offset, val_offset)

class MyDataset(Dataset):
    """
       paddle.io.Dataset
    """
    def __init__(self, mode='train'):
        """

        """
        super(MyDataset, self).__init__()

        np.random.shuffle(data_list)
        if mode == 'train':
            self.data_list = data_list[0: train_offset]
            pass
        elif mode == 'val':
            self.data_list = data_list[train_offset: val_offset]
            pass
        elif mode == 'test':
            self.data_list = data_list[val_offset:]
            pass
        else:
            print("mode should be in ['train', 'test', 'val']")
        self.num_samples = len(self.data_list)

    def __getitem__(self, index):
        """
           __getitem__    index
        """
        data = self.data_list[index][0]
        padlen = 223 - data.shape[0]
        data = np.pad(data, ((0,padlen),(0,0)))
        label = np.array(self.data_list[index][1], dtype=np.int64)

        return data, label

    def __len__(self):
```

```
        """
            __len__
        """
        return self.num_samples

train_dataset = MyDataset(mode='train')
test_dataset = MyDataset(mode='test')
val_dataset = MyDataset(mode='val')

print('=============train_dataset len is {} ============='.
 ↪format(len(train_dataset)))
for data, label in train_dataset:
    print(data.shape, label)
    break
print('=============test_dataset len is {} ============='.
 ↪format(len(test_dataset)))
for data, label in test_dataset:
    print(data.shape, label)
    break
print('=============val_dataset len is {} ============='.
 ↪format(len(val_dataset)))
for data, label in val_dataset:
    print(data.shape, label)
    break
```

```
1224 1632
=============train_dataset len is 1224 =============
(223, 80) 1
=============test_dataset len is 409 =============
(223, 80) 1
=============val_dataset len is 408 =============
(223, 80) 1
```

[6]:
```
train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True,␣
 ↪drop_last=True)
test_loader = DataLoader(test_dataset, batch_size=BATCH_SIZE, shuffle=True,␣
 ↪drop_last=True)
val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE, shuffle=True,␣
 ↪drop_last=True)
```

[7]:
```
from paddle.nn import Layer, Linear, AdaptiveAvgPool1D, Softmax,␣
 ↪CrossEntropyLoss
import paddle.nn.functional as F
class MyNet(Layer):
    def __init__(self):
        super().__init__()
        self.linear1 = Linear(80, 128)
```

```
        self.avgpool1 = AdaptiveAvgPool1D(output_size=1)
        self.linear2 = Linear(128, 64)
        self.linear3 = Linear(64, 2)
    def forward(self, inputs):
        # inputs.shape = (B, T, L) B   T    L  fbank
        # y.shape = (B, L2) B    L2

        # (1, 223, 80)
        y = paddle.zeros((inputs.shape[0],inputs.shape[1],128), dtype=paddle.
↪float32)
        for idx in range(inputs.shape[1]):
            y[:,idx,:] = self.linear1(inputs[:,idx,:])
        # (1, 223, 128)
        y = paddle.transpose(y, [0, 2, 1])
        # (1, 128, 223)
        y = self.avgpool1(y)
        y = y[:, :, 0]
        # (1, 128)
        y = self.linear2(y)
        y = F.relu(y)
        # (1, 64)
        y = self.linear3(y)
        # (1, 2)
        return y
```

[8]:
```
paddle.summary(MyNet(), (1, 10, 80))
```

```
---------------------------------------------------------------------------
   Layer (type)         Input Shape          Output Shape         Param #
===========================================================================
     Linear-1            [[1, 80]]             [1, 128]           10,368
AdaptiveAvgPool1D-1    [[1, 128, 10]]        [1, 128, 1]             0
     Linear-2            [[1, 128]]            [1, 64]             8,256
     Linear-3            [[1, 64]]              [1, 2]              130
===========================================================================
Total params: 18,754
Trainable params: 18,754
Non-trainable params: 0
---------------------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.07
Estimated Total Size (MB): 0.08
---------------------------------------------------------------------------
```

[8]:
```
{'total_params': 18754, 'trainable_params': 18754}
```

[9]:
```
class PrecisionSoft(paddle.metric.Metric):
    """
    1.  paddle.metric.Metric
    """
    def __init__(self, name='PrecisionSoft'):
        """
        2.
        """
        super(PrecisionSoft, self).__init__()
        self.tp = 0
        self.fp = 0
        self._name = name

    def name(self):
        """
        3.  name
        """
        return self._name

    def update(self, preds, labels):
        """
        5.  update    batch
        -  `compute`                `update`
        -  `compute`      compute    `update`
        """
        sample_num = labels.shape[0]

        preds = paddle.to_tensor(preds, dtype=paddle.float32)
        # print("preds={}".format(preds))
        preds = paddle.argsort(preds, descending=True)
        preds = paddle.slice(
            preds, axes=[len(preds.shape) - 1], starts=[0], ends=[1])
        # print(preds)
        # print(len(preds),sample_num)
        for i in range(sample_num):
            pred = preds[i, 0].numpy()[0]
            label = labels[i]

            if pred == 1:
                #print("VALUE = {}:{}, DTYPE={}:{}".format(pred, label, pred.
↪dtype, label.dtype))
                if pred == label:
                    self.tp += 1
                else:
                    self.fp += 1

    def accumulate(self):
```

```
        """
        6.  accumulate    batch
         `update`      `accumulate`
           `fit`
        """
        #  update
        ap = self.tp + self.fp
        return float(self.tp) / ap if ap != 0 else .0

    def reset(self):
        """
        7.  reset   Epoch          Epoch
        """
        # do reset action
        self.tp = 0
        self.fp = 0
```

[10]:
```
class RecallSoft(paddle.metric.Metric):
    """
    1.  paddle.metric.Metric
    """
    def __init__(self, name='RecallSoft'):
        """
        2.
        """
        super(RecallSoft, self).__init__()
        self.tp = 0
        self.fn = 0
        self._name = name

    def name(self):
        """
        3.  name
        """
        return self._name

    def update(self, preds, labels):
        """
        5.  update    batch
        -  `compute`                `update`
        -  `compute`      compute    `update`
        """
        sample_num = labels.shape[0]

        preds = paddle.to_tensor(preds, dtype=paddle.float32)
        # print("preds={}".format(preds))
        preds = paddle.argsort(preds, descending=True)
```

```
        preds = paddle.slice(
            preds, axes=[len(preds.shape) - 1], starts=[0], ends=[1])
        # print(preds)
        # print(len(preds),sample_num)

        for i in range(sample_num):
            pred = preds[i, 0].numpy()[0]
            label = labels[i]

            if label == 1:
                #print("VALUE = {}:{}, DTYPE={}:{}".format(pred, label, pred.
↪dtype, label.dtype))
                if pred == label:
                    self.tp += 1
                else:
                    self.fn += 1

    def accumulate(self):
        """
        6.  accumulate    batch
          `update`      `accumulate`
            `fit`
        """
        #  update
        recall = self.tp + self.fn
        return float(self.tp) / recall if recall != 0 else .0

    def reset(self):
        """
        7.  reset   Epoch          Epoch
        """
        # do reset action
        self.tp = 0
        self.fn = 0
```

[11]:
```
class F1soft(paddle.metric.Metric):
    """
    1.  paddle.metric.Metric
    """
    def __init__(self, name='F1soft'):
        """
        2.
        """
        super(F1soft, self).__init__()
        self.tp1 = 0
        self.fn = 0
```

```
        self.tp2 = 0
        self.fp = 0
        self._name = name

    def name(self):
        """
        3.  name
        """
        return self._name

    def update(self, preds, labels):
        """
        5.  update    batch
        -  `compute`                `update`
        -  `compute`      compute    `update`
        """
        sample_num = labels.shape[0]

        preds = paddle.to_tensor(preds, dtype=paddle.float32)
        # print("preds={}".format(preds))
        preds = paddle.argsort(preds, descending=True)
        preds = paddle.slice(
            preds, axes=[len(preds.shape) - 1], starts=[0], ends=[1])
        # print(preds)
        # print(len(preds),sample_num)

        for i in range(sample_num):
            pred = preds[i, 0].numpy()[0]
            label = labels[i]

            if label == 1:
                #print("VALUE = {}:{}, DTYPE={}:{}".format(pred, label, pred.
↪dtype, label.dtype))
                if pred == label:
                    self.tp1 += 1
                else:
                    self.fn += 1

            if pred == 1:
                if pred == label:
                    self.tp2 += 1
                else:
                    self.fp += 1
```

```
    def accumulate(self):
        """
        6.  accumulate    batch
          `update`      `accumulate`
            `fit`
        """
        #  update
        ap = self.tp2 + self.fp
        recall = self.tp1 + self.fn

        ap = float(self.tp2) / ap if ap != 0 else .0
        recall = float(self.tp1) / recall if recall != 0 else .0

        return 2 * (ap * recall) / (ap + recall) if (ap + recall) != 0 else .0

    def reset(self):
        """
        7.  reset   Epoch         Epoch
        """
        # do reset action
        self.tp1 = 0
        self.fn = 0
        self.tp2 = 0
        self.fp = 0
```

[12]:
```
from paddle import Model
from paddle.optimizer import Adam
from paddle.metric import Accuracy, Precision, Recall

model = Model(MyNet())

model.prepare(Adam(learning_rate=0.001, parameters= model.parameters()),
              CrossEntropyLoss(),
               [Accuracy(), PrecisionSoft(), RecallSoft(), F1soft()]
              )

model.fit(train_loader,val_loader,epochs=10, verbose=2)
```

```
The loss value printed in the log is the current step, and the metric is the
average value of previous steps.
Epoch 1/10

D:\Miniconda3\envs\wenet\lib\site-packages\paddle\fluid\layers\utils.py:77:
DeprecationWarning: Using or importing the ABCs from 'collections' instead of
from 'collections.abc' is deprecated since Python 3.3, and in 3.10 it will stop
working
  return (isinstance(seq, collections.Sequence) and

step 10/19 - loss: 0.5410 - acc: 0.8281 - PrecisionSoft: 0.9167 - RecallSoft:
```

```
0.8949 - F1soft: 0.9057 - 214ms/step
step 19/19 - loss: 0.4673 - acc: 0.8808 - PrecisionSoft: 0.9280 - RecallSoft:
0.9452 - F1soft: 0.9365 - 214ms/step
Eval begin…
step 6/6 - loss: 0.4824 - acc: 0.9453 - PrecisionSoft: 0.9453 - RecallSoft:
1.0000 - F1soft: 0.9719 - 141ms/step
Eval samples: 384
Epoch 2/10
step 10/19 - loss: 0.2402 - acc: 0.9328 - PrecisionSoft: 0.9328 - RecallSoft:
1.0000 - F1soft: 0.9652 - 197ms/step
step 19/19 - loss: 0.3511 - acc: 0.9301 - PrecisionSoft: 0.9301 - RecallSoft:
1.0000 - F1soft: 0.9638 - 196ms/step
Eval begin…
step 6/6 - loss: 0.3569 - acc: 0.9453 - PrecisionSoft: 0.9453 - RecallSoft:
1.0000 - F1soft: 0.9719 - 139ms/step
Eval samples: 384
Epoch 3/10
step 10/19 - loss: 0.3060 - acc: 0.9313 - PrecisionSoft: 0.9313 - RecallSoft:
1.0000 - F1soft: 0.9644 - 198ms/step
step 19/19 - loss: 0.4810 - acc: 0.9301 - PrecisionSoft: 0.9301 - RecallSoft:
1.0000 - F1soft: 0.9638 - 196ms/step
Eval begin…
step 6/6 - loss: 0.3675 - acc: 0.9479 - PrecisionSoft: 0.9504 - RecallSoft:
0.9973 - F1soft: 0.9733 - 171ms/step
Eval samples: 384
Epoch 4/10
step 10/19 - loss: 0.4731 - acc: 0.9203 - PrecisionSoft: 0.9218 - RecallSoft:
0.9983 - F1soft: 0.9585 - 213ms/step
step 19/19 - loss: 0.2288 - acc: 0.9293 - PrecisionSoft: 0.9300 - RecallSoft:
0.9991 - F1soft: 0.9633 - 202ms/step
Eval begin…
step 6/6 - loss: 0.2231 - acc: 0.9479 - PrecisionSoft: 0.9479 - RecallSoft:
1.0000 - F1soft: 0.9733 - 140ms/step
Eval samples: 384
Epoch 5/10
step 10/19 - loss: 0.3428 - acc: 0.9266 - PrecisionSoft: 0.9279 - RecallSoft:
0.9983 - F1soft: 0.9618 - 197ms/step
step 19/19 - loss: 0.3204 - acc: 0.9293 - PrecisionSoft: 0.9315 - RecallSoft:
0.9973 - F1soft: 0.9633 - 193ms/step
Eval begin…
step 6/6 - loss: 0.2424 - acc: 0.9453 - PrecisionSoft: 0.9478 - RecallSoft:
0.9973 - F1soft: 0.9719 - 136ms/step
Eval samples: 384
Epoch 6/10
step 10/19 - loss: 0.3199 - acc: 0.9281 - PrecisionSoft: 0.9323 - RecallSoft:
0.9950 - F1soft: 0.9626 - 210ms/step
step 19/19 - loss: 0.4314 - acc: 0.9301 - PrecisionSoft: 0.9337 - RecallSoft:
0.9956 - F1soft: 0.9636 - 219ms/step
```

```
Eval begin…
step 6/6 - loss: 0.1798 - acc: 0.9531 - PrecisionSoft: 0.9555 - RecallSoft:
0.9973 - F1soft: 0.9759 - 150ms/step
Eval samples: 384
Epoch 7/10
step 10/19 - loss: 0.2322 - acc: 0.9375 - PrecisionSoft: 0.9415 - RecallSoft:
0.9950 - F1soft: 0.9675 - 196ms/step
step 19/19 - loss: 0.2832 - acc: 0.9301 - PrecisionSoft: 0.9343 - RecallSoft:
0.9947 - F1soft: 0.9636 - 198ms/step
Eval begin…
step 6/6 - loss: 0.3099 - acc: 0.9505 - PrecisionSoft: 0.9676 - RecallSoft:
0.9808 - F1soft: 0.9741 - 163ms/step
Eval samples: 384
Epoch 8/10
step 10/19 - loss: 0.2156 - acc: 0.9328 - PrecisionSoft: 0.9441 - RecallSoft:
0.9866 - F1soft: 0.9649 - 234ms/step
step 19/19 - loss: 0.2494 - acc: 0.9285 - PrecisionSoft: 0.9365 - RecallSoft:
0.9903 - F1soft: 0.9626 - 231ms/step
Eval begin…
step 6/6 - loss: 0.1944 - acc: 0.9635 - PrecisionSoft: 0.9656 - RecallSoft:
0.9973 - F1soft: 0.9812 - 142ms/step
Eval samples: 384
Epoch 9/10
step 10/19 - loss: 0.4138 - acc: 0.9094 - PrecisionSoft: 0.9303 - RecallSoft:
0.9745 - F1soft: 0.9519 - 226ms/step
step 19/19 - loss: 0.4826 - acc: 0.9252 - PrecisionSoft: 0.9377 - RecallSoft:
0.9850 - F1soft: 0.9607 - 214ms/step
Eval begin…
step 6/6 - loss: 0.5841 - acc: 0.9479 - PrecisionSoft: 0.9479 - RecallSoft:
1.0000 - F1soft: 0.9733 - 140ms/step
Eval samples: 384
Epoch 10/10
step 10/19 - loss: 0.2610 - acc: 0.9422 - PrecisionSoft: 0.9489 - RecallSoft:
0.9917 - F1soft: 0.9698 - 204ms/step
step 19/19 - loss: 0.2051 - acc: 0.9326 - PrecisionSoft: 0.9374 - RecallSoft:
0.9938 - F1soft: 0.9648 - 198ms/step
Eval begin…
step 6/6 - loss: 0.2304 - acc: 0.9583 - PrecisionSoft: 0.9629 - RecallSoft:
0.9945 - F1soft: 0.9784 - 135ms/step
Eval samples: 384
```

[13]:
```
result = model.predict(test_loader)
print(len(result), len(result[0]), result[0][0].shape)
result = paddle.argsort(paddle.to_tensor(result), descending=True)
result = paddle.slice(result, axes=[len(result.shape) - 1], starts=[0],␣
 ↪ends=[1])
print(result.shape)
```

```
Predict begin…
step 6/6 [==============================] - 96ms/step
Predict samples: 384
1 6 (64, 2)
[1, 6, 64, 1]
```

[14]:
```
result = result[0,:,:,0]
result
```

[14]:
```
Tensor(shape=[6, 64], dtype=int64, place=CUDAPlace(0), stop_gradient=True,
       [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
         1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])
```

64 0 12 0 64-12 = 52 1

[15]:
```
t_r = paddle.where(result == 0, paddle.ones(result.shape), paddle.zeros(result.
 ↪shape))
print(paddle.sum(t_r))
```

```
Tensor(shape=[1], dtype=float32, place=CUDAPlace(0), stop_gradient=True,
       [12.])
```

[17]:
```
!pip install pandoc
```

```
Collecting pandoc
  Downloading pandoc-2.1.tar.gz (29 kB)
Collecting plumbum
  Downloading plumbum-1.7.2-py2.py3-none-any.whl (117 kB)
Collecting ply
  Using cached ply-3.11-py2.py3-none-any.whl (49 kB)
Requirement already satisfied: pywin32 in d:\miniconda3\envs\wenet\lib\site-
packages (from plumbum->pandoc) (303)
Building wheels for collected packages: pandoc
```

```
  Building wheel for pandoc (setup.py): started
  Building wheel for pandoc (setup.py): finished with status 'done'
  Created wheel for pandoc: filename=pandoc-2.1-py3-none-any.whl size=29531
sha256=7aa1171e074b7a1e5214bb7289ceb0f18fb7c8837091e0f229749d1e7ffd885f
  Stored in directory: c:\users\11347\appdata\local\pip\cache\wheels\ce\41\63\bf
7cb60c03dc7f93180e91e0972c12345b40bf59212d307157
Successfully built pandoc
Installing collected packages: ply, plumbum, pandoc
Successfully installed pandoc-2.1 plumbum-1.7.2 ply-3.11

WARNING: Error parsing requirements for numpy: [Errno 2] No such file or
directory: 'd:\\miniconda3\\envs\\wenet\\lib\\site-packages\\numpy-1.22.2.dist-
info\\METADATA'
```

[ ]:

[ ]:

[ ]: