model.wav2vec

March 12, 2024

1 CS-72: Accelerated Computational Linguistics

1.1 Final Project Code

1.1.1 Emotion Detection in Audio

1.1.2 Team Members:

- Aiwei Zhang
- Amittai Siavava
- Carlos Guerrero Alvarez

This is a variant of our model that uses Wav2Vec features extracted from audio.

[ ]:
```python
# %pip install pytorch_lightning
# %pip install transformers
# %pip install torchmetrics
# %pip install soundfile
# %pip install librosa
# %pip install ipywidgets
```

[ ]:
```python
# imports
import torch
import torch.nn as nn
import torch.optim as optim
import math
import pytorch_lightning as pl
# import torchaudio
import torchmetrics
from torch.optim.lr_scheduler import ReduceLROnPlateau
```

```
/usr/lib/python3/dist-packages/pkg_resources/__init__.py:116:
PkgResourcesDeprecationWarning: 1.12.1-git20200711.33e2d80-dfsg1-0.6 is an
invalid version and will not be supported in a future release
  warnings.warn(
/usr/lib/python3/dist-packages/pkg_resources/__init__.py:116:
PkgResourcesDeprecationWarning: 1.12.1-git20200711.33e2d80-dfsg1-0.6 is an
invalid version and will not be supported in a future release
  warnings.warn(
```

[ ]:
```
torch.cuda.is_available()
```

[ ]:
```
True
```

[ ]:
```
from local_dataset import AudioEmotionsDataset
# import TQDMProgressBar
from pytorch_lightning.callbacks import TQDMProgressBar
```

[ ]:
```
BATCH_SIZE = 32
# dataset = AudioEmotionsDataset("data/audio-emotions", batch_size=BATCH_SIZE)
dataset = AudioEmotionsDataset("/home/ubuntu/siavava-west-1/test/data/
 ↪audio-emotions", batch_size=BATCH_SIZE, max_size=200, feature_type="wav2vec")

train = dataset.train_dataloader
test = dataset.test_dataloader
```

```
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at
facebook/wav2vec2-base-960h and are newly initialized:
['wav2vec2.masked_spec_embed']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
/usr/lib/python3/dist-packages/scipy/__init__.py:146: UserWarning: A NumPy
version >=1.17.3 and <1.25.0 is required for this version of SciPy (detected
version 1.25.2
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
2024-03-12 18:25:01.408103: I tensorflow/core/util/port.cc:110] oneDNN custom
operations are on. You may see slightly different numerical results due to
floating-point round-off errors from different computation orders. To turn them
off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-03-12 18:25:01.451361: I tensorflow/core/platform/cpu_feature_guard.cc:182]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX512F AVX512_VNNI, in other operations,
rebuild TensorFlow with the appropriate compiler flags.
```

[ ]:
```
class SpeechEmotionRecognitionModel(pl.LightningModule):
    def __init__(self, input_size, num_classes, dim_feedforward=2048,␣
 ↪dim_model=1024, nhead=8, num_encoder_layers=6, num_decoder_layers=6,␣
 ↪lr=1e-2, dropout=0.1):
        super(SpeechEmotionRecognitionModel, self).__init__()
        self.lr = lr

        encoder_layers = nn.TransformerEncoderLayer(dim_model, nhead,␣
 ↪dim_feedforward, dropout)
        self.transformer_encoder = nn.TransformerEncoder(encoder_layers,␣
 ↪num_encoder_layers)
        self.encoder = nn.Linear(input_size, dim_model)
        self.decoder = nn.Sequential(
            nn.Linear(dim_model, num_classes),
            nn.Softmax(dim=1)
        )

        self.loss_function = nn.CrossEntropyLoss()

        # initialize the metrics
        self.precision = torchmetrics.Precision(task='multiclass',␣
↪num_classes=num_classes, average="macro")
        self.recall = torchmetrics.Recall(task='multiclass',␣
↪num_classes=num_classes, average="macro")
        self.F1 = torchmetrics.F1Score(task='multiclass',␣
↪num_classes=num_classes, average="macro")

    def forward(self, src):
        src = self.encoder(src)
        # print(f"{src.shape = }")
        # src = src.unsqueeze(1)  # Add batch dimension
        output = self.transformer_encoder(src)
        output = output.squeeze(1)  # Remove the batch dimension
        output = self.decoder(output)
        return output

    def training_step(self, batch, batch_idx):
        src, tgt = batch[0], batch[1]
        output = self(src)
        loss = self.loss_function(output, tgt)
        self.log('cross entropy loss', loss, on_step=True, on_epoch=True,␣
↪prog_bar=True)
        return loss

    def validation_step(self, batch, batch_idx):
        src, tgt = batch
        output = self(src)
        loss = self.loss_function(output, tgt.float())
        self.log('cross entropy loss', loss, on_epoch=True, prog_bar=True)

    def configure_optimizers(self):
        optimizer = torch.optim.Adam(self.parameters(), lr=self.lr)
        scheduler = {
            'scheduler': ReduceLROnPlateau(optimizer, mode='min', factor=0.1,␣
↪patience=2, verbose=True),
            'monitor': 'cross entropy loss_epoch',  # Name of the metric to␣
↪monitor
            'interval': 'epoch',
            'frequency': 1,
        }
        return {'optimizer': optimizer, 'lr_scheduler': scheduler}

    # function for evaluating the quality of output and target
    def evaluation(self, output, target, loss):

        precision = self.precision(output, target)
        recall = self.recall(output, target)
        f1 = self.F1(output, target)

        print(f"CE:        {loss}")
        print(f"PRECISION: {precision}")
        print(f"RECALL:    {recall}")
        print(f"F1:        {f1}")
```

Create Model and Trainer

[ ]:

```
model = SpeechEmotionRecognitionModel(input_size=dataset.feature_count,␣
  ↪num_classes=dataset.class_count)

for p in model.parameters():
    if p.dim() > 1:
        nn.init.xavier_uniform_(p)

trainer = pl.
  ↪Trainer(default_root_dir='checkpoints',callbacks=[TQDMProgressBar(refresh_rate=10)],␣
  ↪accelerator="auto", max_epochs=50, min_epochs=10, log_every_n_steps=1)
```

```
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
```

Sample Evaluation

Train The Model

[ ]:

```
trainer.fit(model, train_dataloaders=train, val_dataloaders=test)
```

```
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name                | Type                | Params
------------------------------------------------------------
0 | transformer_encoder | TransformerEncoder  | 50.4 M
1 | encoder             | Linear              | 102 M
2 | decoder             | Sequential          | 7.2 K
3 | loss_function       | CrossEntropyLoss    | 0
4 | precision           | MulticlassPrecision | 0
5 | recall              | MulticlassRecall    | 0
```

```
6 | F1                  | MulticlassF1Score   | 0
------------------------------------------------------------
152 M     Trainable params
0         Non-trainable params
152 M     Total params
610.718   Total estimated model params size (MB)

Sanity Checking: |          | 0/? [00:00<?, ?it/s]

Training: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Epoch 00005: reducing learning rate of group 0 to 1.0000e-03.

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Epoch 00008: reducing learning rate of group 0 to 1.0000e-04.

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Epoch 00011: reducing learning rate of group 0 to 1.0000e-05.

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Epoch 00014: reducing learning rate of group 0 to 1.0000e-06.

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Epoch 00017: reducing learning rate of group 0 to 1.0000e-07.

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]

Epoch 00020: reducing learning rate of group 0 to 1.0000e-08.
```

```
Validation: |          | 0/? [00:00<?, ?it/s]

Validation: |          | 0/? [00:00<?, ?it/s]
```

Sample Evaluation

[ ]:

```python
for batch in test:
    X, y = batch
    X = X.cuda(0)
    y = y.cuda(0)
    model = model.cuda(0)
    print(f"X: {X.shape}")
    print(f"y: {y.shape}")
    # print(f"model: {model.device}")

    output = model(X)
    print(f"output: {output.shape}")
    model.evaluation(output, y, model.loss_function(output, y))


    # for i in range(32):
    #     print(f"{torch.argmax(output[i]):2d} | {torch.argmax(y[i]):2d} with␣
 ↪{torch.max(output[i]):.2f}")
    break
```

```
X: torch.Size([32, 99875])
y: torch.Size([32, 7])
output: torch.Size([32, 7])
CE:         1.9527029991149902
PRECISION:  0.4285714328289032
RECALL:     0.5
F1:         0.4615384638309479
```