

model.mfcc

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 MFCC (Mel-frequency cepstral coefficients) features extracted from audio.

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

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

      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}")

LOADED: 2400

```
[ ]: class Reshape(nn.Module):
      def __init__(self):
          super(Reshape, self).__init__()
          # self.shape = shape

      def forward(self, x):
          # return x.view((x.shape[0], *self.shape))
          return x.view(x.shape[0], 1, x.shape[1])

      def __call__(self, x):
          return self.forward(x)

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=0.5,
    ↪dropout=0.1):
        super(SpeechEmotionRecognitionModel, self).__init__()
        self.lr = lr

        self.layers = nn.Sequential(
```

```

        Reshape(),
        nn.Conv1d(in_channels=1, kernel_size=40, out_channels=7),
        nn.ReLU(),
        nn.Dropout(p=0.2),
        nn.Flatten(start_dim=1, end_dim=2),
        nn.Linear(7, 7),
        nn.Softmax(dim=1),
        # nn.Sigmoid()
    )

    # initialize the metrics
    self.loss_function = nn.CrossEntropyLoss()
    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):
        output = self.layers(src)
        return output

    def training_step(self, batch, batch_idx):
        src, tgt = batch

        output = self(src)

        loss = self.loss_function(output, torch.argmax(tgt, axis=1))
        self.log('cross entropy loss_step', 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, torch.argmax(tgt, axis=1))
        # print(f"VALIDATION LOSS: {loss}")

    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_step', # 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}")

```

```

[ ]: 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)
# callbacks=[TQDMProgressBar(refresh_rate=10)]
trainer = pl.Trainer(default_root_dir='checkpoints', accelerator="auto",
    ↪ max_epochs=50, min_epochs=2, 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

```

[ ]: for batch in test:
    X, y = batch
    X = X.cuda(0)
    y = y.cuda(0)
    print(f"X: {X.device}")
    print(f"y: {y.device}")
    print(f"model: {model.device}")

    model = model.cuda(0)
    output = model(X)
    # print out put device
    print(f"output: {X.device}")
    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} >>>
↳{torch.max(output[i]):.2f}")
break
```

```
X: cuda:0
y: cuda:0
model: cpu
output: cuda:0
CE:      2.068202495574951
PRECISION: 0.42825111746788025
RECALL:   0.4973958432674408
F1:      0.4602409601211548
```

```
[ ]: torch.set_float32_matmul_precision('medium')
trainer.fit(model, train_dataloaders=train, val_dataloaders=test)
```

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

	Name	Type	Params
0	layers	Sequential	343
1	loss_function	CrossEntropyLoss	0
2	precision	MulticlassPrecision	0
3	recall	MulticlassRecall	0
4	F1	MulticlassF1Score	0

```
-----
343      Trainable params
0        Non-trainable params
343      Total params
0.001    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]
```

```
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 5.0000e-02.
```

```
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 5.0000e-03.

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 5.0000e-04.

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 5.0000e-05.

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 5.0000e-06.

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

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

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

Epoch 00023: reducing learning rate of group 0 to 5.0000e-07.

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

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

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

Epoch 00026: reducing learning rate of group 0 to 5.0000e-08.

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 00031: reducing learning rate of group 0 to 5.0000e-09.

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]
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]
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]
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]
`Trainer.fit` stopped: `max_epochs=50` reached.

```

```

[ ]: for batch in test:
    X, y = batch
    X = X.cuda(0)
    y = y.cuda(0)
    print(f"X: {X.device}")
    print(f"y: {y.device}")
    print(f"model: {model.device}")

    model = model.cuda(0)
    output = model(X)
    # print out put device
    print(f"output: {X.device}")
    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} >>>␣
    ↪{torch.max(output[i]):.2f}")
    break

```

```

X: cuda:0
y: cuda:0
model: cuda:0

```

```
output: cuda:0
CE:      1.8528821468353271
PRECISION: 0.6099656224250793
RECALL:   0.6041666269302368
F1:      0.6068861484527588
```

```
[ ]: # create 'models' directory if nonexistent
      # import os
      # if not os.path.exists("models"):
      #     os.makedirs("models")
      # # save model weights
      # torch.save(model.state_dict(), "./models/model_weights.pth")
```