گزارش توسعه‌ی Task ورودی شرکت Part AI

دستیار صوتی برای اپلیکیشن سیگنال

(فرزاد بلورچی فرد – ۱۴۰۲.۰۲.۰۳)

۱) فایل partai_nn.ipynb:

- در توسعه‌ی این فایل از این tutorial الگو برداری شده است.
- در ابتدا حتما تمام دیتاست باید در یک فولدر اصلی باشد که در آن به تعداد کلاس‌ها زیرفولدر (شامل فایل‌های صوتی) وجود دارد.

پیش پردازش‌ها:

- اینستنس کلاس AudioDataset با گرفتن آدرس فولدر اصلی ایجاد می‌شود و با رفتن به هر زیر فولدر، audiofileها و labelهای متناظرشان را تشکیل می‌دهد. اسمی که برای زیرفولدرها گذاشته شده به عنوان label انتخاب می‌شود.
- در ادامه کل دیتاها با انتخاب یک ضریب (در اینجا ۰.۰۵) به train_set و test_set بطور رندوم تقسیم می‌شود.
- تنها یک Transform اعمال می‌گردد: تغییر Sample rate فایل‌های صوتی به 8000
- با توجه به برابر نبودن طول فایل‌های صوتی، برای ایجاد یک batch در loaderها، با zero padding تنسورها هم طول می‌شوند.
- اندازه‌ی هر batch برای loaderها ۳۲ در نظر گرفته شد.

معماری شبکه‌ی عصبی:

- مدل استفاده شده عینا مدل داخل لینک tutorial می‌باشد که از این مقاله گرفته شده است.
- **لایه‌ی convolutional اول: (self.conv1)**
  - دارای ۱ چنل ورودی و ۳۲ چنل خروجی
  - اندازه‌ی فیلتر (kernel): ۸۰ - جابجایی فیلتر (stride): ۱۶
  - خروجی‌ها با روش batchNormalization نرمالایز می‌شوند. (مقاله: to avoid clutter)
  - تابع اکتیویشن: ReLU
- **لایه‌ی maxpooling اول: (self.pool1)**
  - اندازه‌ی pooling: ۴
  - (تنسور ورودی ۴ واحد ۴ واحد بررسی می‌شود و بیشینه مقدار بجای آن ۴ مقدار جایگزین می‌شود.)
- **لایه‌ی convolutional دوم: (self.conv2)**
  - دارای ۳۲ چنل ورودی و ۳۲ چنل خروجی
  - اندازه‌ی فیلتر (kernel): ۳ - جابجایی فیلتر (stride): ۱ (مقدار پیش‌فرض)
  - خروجی‌ها با روش batchNormalization نرمالایز می‌شوند. (مقاله: to avoid clutter)
  - تابع اکتیویشن: ReLU
- **لایه‌ی maxpooling دوم: (self.pool2)**

- اندازه‌ی pooling: ۴

- **لایه‌ی convolutional سوم: (self.conv3)**
  - دارای ۳۲ چنل ورودی و ۶۴ = ۳۲ × ۲ چنل خروجی
  - اندازه‌ی فیلتر (kernel): ۳ - جابجایی فیلتر (stride): ۱ (مقدار پیش‌فرض)
  - خروجی‌ها با روش batchNormalization نرمالایز می‌شوند. (مقاله: to avoid clutter)
  - تابع اکتیویشن: ReLU
- **لایه‌ی maxpooling سوم: (self.pool3)**
  - اندازه‌ی pooling: ۴
- **لایه‌ی convolutional چهارم: (self.conv4)**
  - دارای ۶۴ = ۳۲ × ۲ چنل ورودی و ۶۴ = ۳۲ × ۲ چنل خروجی
  - اندازه‌ی فیلتر (kernel): ۳ - جابجایی فیلتر (stride): ۱ (مقدار پیش‌فرض)
  - خروجی‌ها با روش batchNormalization نرمالایز می‌شوند. (مقاله: to avoid clutter)
  - تابع اکتیویشن: ReLU
- **لایه‌ی maxpooling چهارم: (self.pool4)**
  - اندازه‌ی pooling: ۴
- **اعمال global average pooling: (F.avg_pool1d())**
  - بکارگیری avgpool با پنجره‌ای (kernel size) به طول کل ورودی (کاهش بعد)
- **جابجایی بعد دوم و سوم با استفاده از permute():**
  - تنسوری که از مرحله‌ی قبلی (avgpool) می‌آید دارای چنین ابعادی است:
    - [batch_size, channels, sequence_length]
  - یک لایه‌ی fully connected خطی (nn.Linear) تنسوری به فرم زیر ورودی می‌گیرد:
    - [batch_size, sequence_length, channels]
  - برای همین باید پیش از دادن تنسور به fc1 جای بعد دوم و سوم را عوض کرد.
- **لایه‌ی خطی (fully connected) انتهایی: (self.fc1)**
  - دارای ۱۲ نورون (تعداد کلاس‌ها = ۱۲ کلمه)
  - به هر نورون، ۶۴ خروجی از لایه‌ی قبلی وارد می‌شود.
  - ابعاد خروجی fc1: [batch_size, sequence_length, n_output]
  - تابع اکتیویشن: log_softmax (روی dim=2 یعنی n_output)

نحوه‌ی Trainکردن مدل و نتایج:

- مقدار learning_rate = ۰.۰۱
- همچنین بعد از انجام هر ۲۰تا epoch مقدار lr یک‌دهم می‌شود. (scheduler)
- تعداد کل epochها = ۱۰۰
- نتیجه‌ی کسب شده روی test_set: ۹۲ درصد صحیح (۲۲ از ۲۴)

بعد از اتمام Train، مدل بدست آمده در فایل ‘model.pt’ ذخیره می‌گردد.

۲) فایل Recognizer.py:

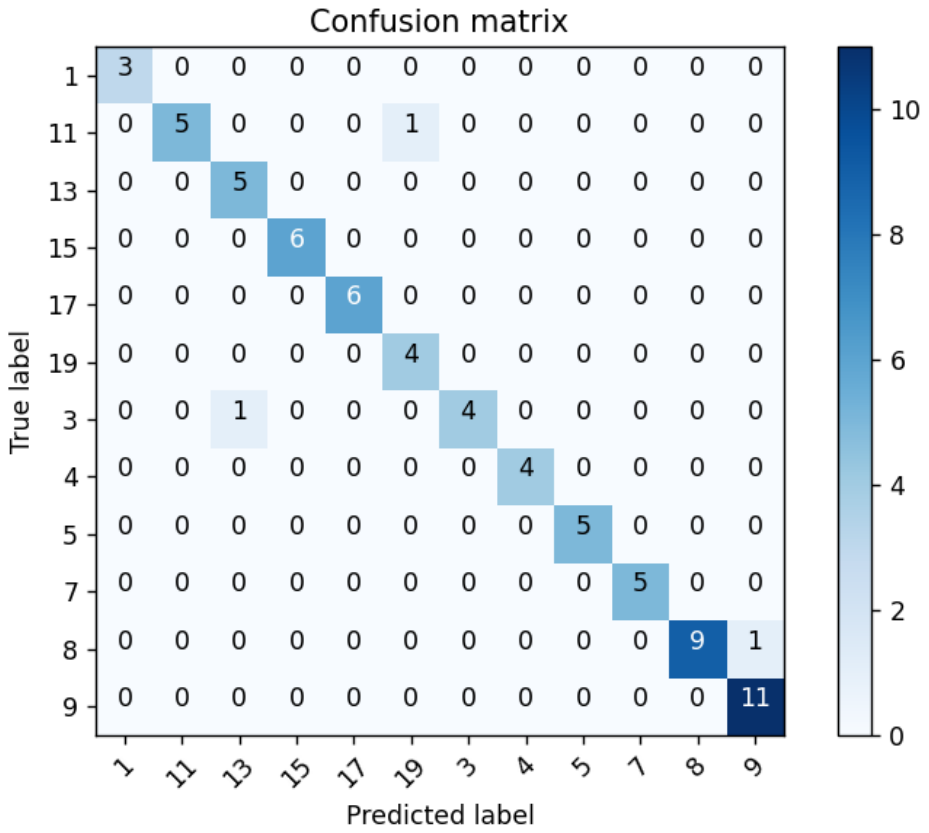
- شامل کلاس خواسته‌شده در تسک (class Recognizer) می‌باشد. (همچنین کلاس CNN جهت اجرای مدل مجدد آورده شده)

توضیحات مربوط به کلاس:

- برای ساخت یک اینستنس از این کلاس باید آدرس مدل Train شده از قسمت قبلی ('model.pt') به constructor داده شود.
- با فراخوانی load_model()، پارامترها و مدل بدست‌آمده در attribute مورد نظر (recognizer.model) ذخیره می‌شود.
- سپس با فراخوانی recognizer.predict(AUDIO_PATH) کلاس پیشبینی‌شده برای فایل (یک فایل) نمایش داده می‌شود.
- جهت تست راحتر، متود recognizer.folder_predict(FOLDER_PATH) هم ایجاد شده که با گرفتن آدرس یک فولدر دو لیست برمی‌گرداند. یک لیست برای labelهای صحیح (actuals) و یک لیست برای labelهای پیشبینی‌شده (predictions).
  - **توجه:** نام فایل‌های صوتی در این فولدر **باید** به این فرمت باشد: name-(label_int).mp3

ترسیم Confusion Matrix:

- جهت ایجاد ماتریس confusion تعدادی (۷۰تا) از فایل‌های صوتی داده شده را بطور رندوم داخل فولدر 'test' قرار می‌دهیم.
  - **توجه:** می‌دانیم این تست دقیقی نیست زیرا مدل تقریبا با همین فایل‌ها train شده است.
- با فراخوانی recognizer.folder_predict('test') دو لیست actuals و predictions بدست می‌آید.
- از این دو لیست جهت ترسیم ماتریس استفاده شد و نتیجه‌ی زیر بدست‌آمد:



شکل ۱ – ماتریس confusion برای ۷۰ فایل صوتی رندوم