به نام خدا

گزارش پروژه دسته‌بندی داده‌های صوتی

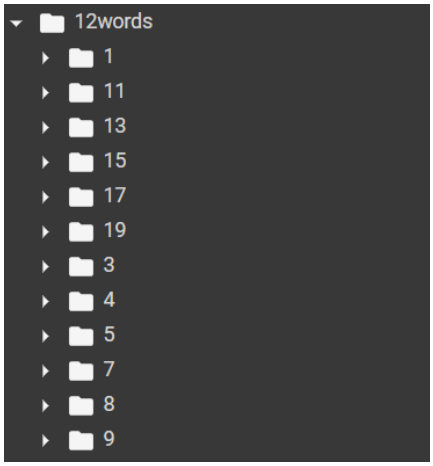
محدثه سادات اطیابی

تعریف مسئله: در این پروژه می‌خواهیم یک دستیار صوتی توسعه دهیم تا داده‌ها را در ۱۲ کلاس، دسته‌بندی کند. کلاس‌های مورد نظر عبارتند از: اوراق، ارز، سکه، بانک، طلا، نفت، مشتقات، فلزات، صندوق سهامی، صندوق درآمد ثابت، صندوق مختلط و صندوق قابل معامله. برای هر یک از ۱۲ کلاس، ۴۰ نمونه فایل صوتی موجود است که به عنوان دیتاست استفاده خواهد شد. مدل پیشنهادی باید حجمی کمتر از ۱۰ مگابایت داشته باشد و با استفاده از زبان پایتون توسعه داده خواهد شد.

توضیح کد و راه حل: کدهای این پروژه در فایل speech_command_classification.ipynb موجود می‌باشد که در قالب نوت بوک نوشته شده‌است. در ابتدا به کتابخانه‌های مورد نیاز برای انجام این پروژه پرداخته می‌شود. سپس پردازش داده در قالب دیتاست مورد بررسی قرار خواهد گرفت. در گام بعد، مدل توسعه یافته توضیح داده خواهد شد و در نهایت، به آموزش مدل و بررسی نتایج پرداخته خواهد شد.

- **کتابخانه‌ها:** برای آموزش مدل به GPU نیاز است و به همین دلیل کد پروژه در google colab اجرا شده است. بسیاری از کتابخانه‌ها بصورت پیش‌فرض در کولب نصب می‌باشند و می‌توان از آنها استفاده کرد. در این پروژه، برای توسعه مدل از کتابخانه pytorch و برای کار با داده‌های صوتی از کتابخانه torchaudio استفاده شده است. البته کتابخانه‌های دیگری مانند sklearn (برای رسم ماتریس confusion)، IPython (برای پخش فایل صوتی)، tqdm (برای ترسیم میزان پیشرفت آموزش مدل) و .... نیز استفاده شده است.

- **ساخت دیتاست:** برای تبدیل دیتاهای موجود به فرمت دیتاست، کلاسی بنام SpeechCommandDataset توسعه داده شده است که از کلاس Dataset در کتابخانه torch.utils ارث بری می‌کند. در این کلاس لیستی به عنوان ورودی دریافت می‌شود و مجموعه داده مربوطه تشکیل می‌گردد. در هنگام فراخوانی هر عضو از دیتاست، ویس مربوطه خوانده می‌شود و مقادیر شکل موج، نرخ نمونه برداری و کلاس مربوطه به عنوان خروجی بازگردانده می‌شود. نمونه‌های موجود از هر کلاس در پوشه جداگانه‌ای قرار گرفته است و همگی در پوشه اصلی 12words ذخیره شده‌اند. نحوه قرار گرفتن پوشه‌ها در تصویر زیر قابل مشاهده است. برای بدست آوردن لیست تمام فایل‌های موجود، ابتدا متد getAudioList پیاده سازی شده است که به ازای تمام پوشه‌های موجود در پوشه 12words، فایل‌های موجود در پوشه را در لیستی ذخیره کرده و در نهایت آن را برمی‌گرداند. پس از یافتن لیست تمام نمونه‌های موجود، آن را به نسبت ۸۰٪ ۱۰٪ ۱۰٪ برای داده‌های آموزش، تست و اعتبارسنجی تقسیم کرده و برای هر کدام، یک شئ از کلاس SpeechCommandData ساخته شده است. به این ترتیب دیتاست مربوطه تشکیل شده است.



- **قالب‌بندی داده‌ها:** در گام بعدی نیاز است تا تبدیل‌هایی برای داده‌ها تعریف شود. برای اینکه فرایند دسته بندی داده با سرعت بیشتری انجام شود اما دقت و عملکرد مدل دچار کاهش نشود، از downsample استفاده می‌شود. در برخی از مجموعه داده‌ها معمول است که تعداد کانال‌های را کاهش دهند (مثلا از استریو به مونو) که این کار با گرفتن میانگین در امتداد بُعد کانال یا صرفا نگه داشتن یکی از کانال‌ها انجام می‌شود. از آنجایی که مجموعه داده ما از یک کانال واحد برای صدا استفاده می‌کند، این مورد در اینجا لازم نیست و تنها یک مورد تغییر اعمال شده است.

در گام بعدی، نیاز است تا برچسب داده‌ها با اندیس مناسب، متناظر شود تا برای آموزش مدل از اندیس‌ها استفاده گردد. برای اعمال تناظر بین برچسب و اندیس، دو تابع label_to_index و index_to_label پیاده سازی شده است.

برای تبدیل فهرستی از داده‌های ساخته شده از فایل‌های ضبط شده به تانسور دسته‌ای (batched tensors) برای مدل، یک تابع دسته‌بندی پیاده‌سازی شده است که توسط دیتالودر PyTorch استفاده می‌شود و این امکان را می‌دهد تا روی یک مجموعه داده به صورت دسته‌ای حرکت کرد. درواقع تابع collate_fn این امکان را می‌دهد تا به تعداد اندازه دسته، داده‌ها را از دیتاست دریافت کرده و آنها را پردازش کرد. در این تابع تمام تنسورهای داخل یک دسته pad می‌شوند تا همگی طول یکسانی داشته باشند. پس از این گام، دیتالودرهای مربوط به داده‌ای آموزش، تست و اعتبارسنجی ساخته می‌شوند.

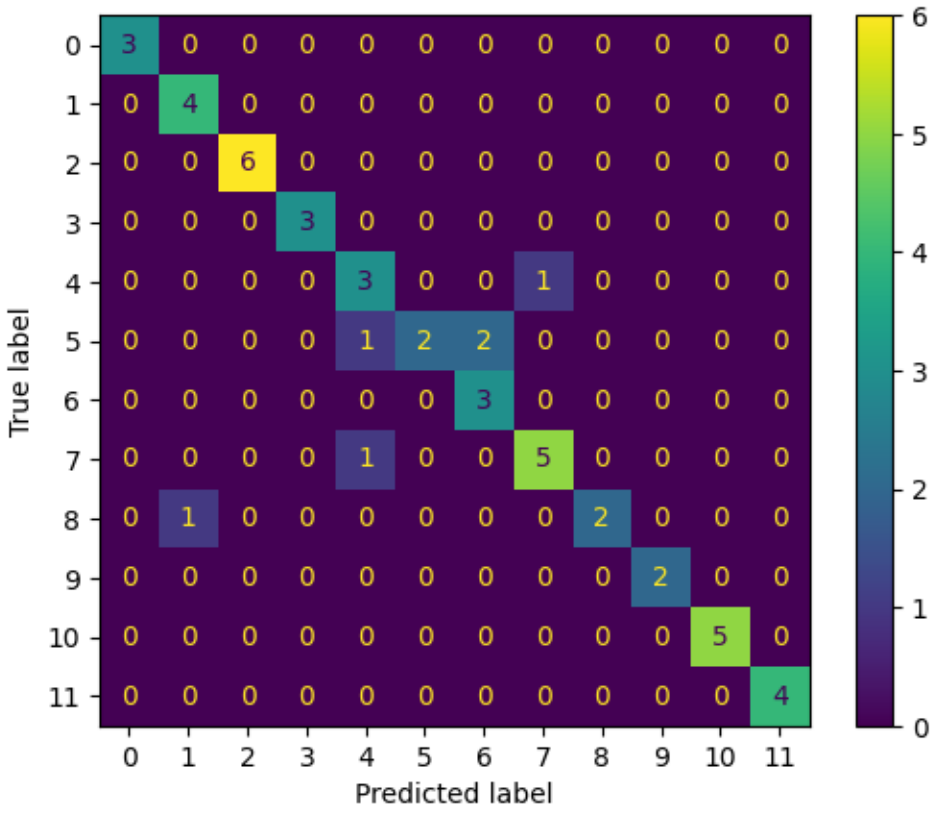
- **تعریف شبکه:** برای این پروژه از یک شبکه عصبی کانولوشن برای پردازش داده‌های صوتی خام استفاده شده است. معمولاً تبدیل‌های پیشرفته‌تری برای داده‌های صوتی اعمال می‌شود، با این حال CNNها می‌توانند برای پردازش دقیق داده‌های خام استفاده شوند. معماری خاص بر اساس معماری شبکه M5 که در این مقاله توضیح داده شده مدل شده است. یکی از جنبه‌های مهم مدل‌هایی که داده‌های صوتی خام را پردازش می‌کنند، میدان دریافتی فیلترهای لایه اول آنها است. اولین فیلتر مدل ما ۸۰ طول دارد، بنابراین هنگام پردازش صدای نمونه برداری شده در ۸ کیلوهرتز، میدان دریافتی حدود ۱۰ میلی ثانیه است. معماری این شبکه و اندازه لایه‌ها و تعداد کل پارامترهای مربوطه در تصویر زیر قابل مشاهده است.

```
M5(
  (conv1): Conv1d(1, 32, kernel_size=(80,), stride=(16,))
  (bn1): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool1): MaxPool1d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv1d(32, 32, kernel_size=(3,), stride=(1,))
  (bn2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool2): MaxPool1d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv1d(32, 64, kernel_size=(3,), stride=(1,))
  (bn3): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool3): MaxPool1d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (conv4): Conv1d(64, 64, kernel_size=(3,), stride=(1,))
  (bn4): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool4): MaxPool1d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=64, out_features=12, bias=True)
)
Number of parameters: 25420
```

تابع بهینه‌ساز در این پروژه Adam می‌باشد که با نرخ یادگیری ۰/۰۱ تعریف شده است و با استفاده از تابع scheduler بعد از گذشت ۲۰ دوره، مقدار نرخ یادگیری برابر با ۰/۰۰۱ خواهد شد.

- **آموزش مدل:** در تابع train، داده‌های آموزشی به تعداد اندازه دسته برداشته شده و پس از اعمال transform روی آنها، به مدل داده می‌شوند. سپس خروجی بدست آمده وارد تابع هزینه می‌شود که در این پروژه از کراس آنتروپی به عنوان تابع هزینه استفاده شده است. در نهایت عملیات backpropagation روی تابع هزینه اجرا شده و سپس step روی تابع بهینه ساز اعمال می‌شود تا وزن‌ها آپدیت شوند. برای توابع تست و اعتبارسنجی نیز به روش مشابه، به تعداد اندازه دسته داده از دیتالودر دریافت می‌شود و پس از اعمال transform، خروجی مدل دریافت شده و با برچسب اصلی داده مقایسه می‌شودتا تعداد پیش بینی‌های درست مدل بدست آید و در نهایت تحت عنوان accuracy گزارش شود. برای آموزش از اندازه دسته ۶ و تعدا دوره ۶۰ استفاده شده است که خروجی‌های بدست آمده در فایل speech_command_classification.ipynb قابل مشاهده می‌باشد. نمودار loss برای داده‌های آموزشی در شکل زیر گزارش شده است.

همانطور که در تصویر مشاهده می‌شود، سیر کلی loss بصورت نزولی است و با گذشت دوره‌های مختلف، در iterationهای متفاوت می‌توان سیر نزولی را شاهد بود که بیانگر عملکرد مناسب مدل می‌باشد. مقدار دقت مدل برای داده‌های تست برابر با ۸۸ درصد گزارش شده است که دقت بالای مدل را در مواجهه با داده‌های جدید نشان می‌دهد. معیار دیگری که برای ارزیابی مدل استفاده شده است، confusion matrix می‌باشد که در محور عمودی، برچسب واقعی داده تست و در محور افقی، برچسب پیش‌بینی شده توسط مدل گزارش شده است که در تصویر زیر قابل مشاهده می‌باشد. همانطور که انتظار می‌رود بیشتر داده‌ها روی قطر اصلی قرار گرفته‌اند که یعنی اکثر برچسب‌ها درست تشخیص داده شده است و تعداد کمی از پیش‌بینی‌ها نادرست هستند.



در پایان مدل آموزش دیده ذخیره شده است و حجم آن کمتر از ۱۰ مگابایت است.

برای اینکه در ادامه تنها از مدل آماده استفاده شود و همواره نیازی به آموزش مدل نباشد، کلاس Recognizer توسعه داده شده است که با تابع load_model مدل آماده در حافظه لود می‌شود و با استفاده تابع predict، آدرس فایل صوتی مربوطه دریافت می‌شود و خروجی بدست آمده از مدل بازگردانده می‌شود.