به نام خدا

دانیال فرهنگی

Speech recognition

شرکت هوش مصنوعی پارت

مسئله تشخیص گفتار، با استفاده از هوش مصنوعی و پردازش صوت برای تشخیص کلمات و تبدیل آن ها به متن به عنوان خروجی، انجام میشود.

برای انجام این مسئله، دو مدل آموزش داده شد. در مدل اول از کانولوشن یک بعدی (بدون آگمنتیشن) و در مدل دوم از کانولوشن دو بعدی (با آگمنتیشن) استفاده شد که به نتایج زیر رسیدیم.

	Train Accuracy	Test Accuracy	Train Loss	Test Loss
A model with Conv 1d	97	94	0.1	0.4
A model with Conv 2d	92	89	0.3	0.4

در ادامه به بررسی کامل طراحی و معماری مدل دوم که از کانولوشن دو بعدی استفاده شده است، میپردازیم.

مرحله pre-processing:

در این مرحله فایل های صوتی را خواندیم و با استفاده از کتابخانه librosa، سیگنال و MFCC هر صوت را بدست می‌آوریم. از آنجایی که ابعاد هر صوت متفاوت است پس یک padding در نظر میگیریم و ابعاد تمام داده‌ها را با هم یکسان میکنیم و در نهایت MFCC و label هر صوت را در یک فایل json ذخیره کردیم.

برای اینکه به دقت بهتری دست پیدا کنیم و مدل بهتر آموزش ببیند، به داده های بیشتری نیاز داشتیم؛ از این رو با استفاده از تکنیک‌های audio data augmentation داده های بیشتری ایجاد کردیم و از آنها در آموزش مدل استفاده کردیم.

آماده‌سازی داده ها برای آموزش:

بعد از مرحله پیش‌پردازش و آماده‌سازی دیتاست، ۷۰ درصد داده‌ها برای train مدل و ۳۰ درصد برای test درنظر گرفته شد.

داده ها را کاملا shuffle میکنیم و سعی میکنیم داده‌ها به طور یکنواخت برای تست و ترین تقسیم شوند به گونه‌ای که از هر label در تست وجود داشته باشد.

داده‌های ورودی همان MFCC مربوط به صوت‌ها هستند و داده‌های خروجی کتگوری مربوط به هر صوت هست. (داده‌های خروجی را به صورت one-hot شده به شبکه میدهیم.

معماری شبکه عصبی:

از چهار لایه کانولوشن دو بعدی و چهار لایه MaxPooling و یک لایه Flatten و دو لایه fully connected و در لایه آخر هم یک لایه Dense با تابع فعالساز softmax برای پیش بینی category استفاده کردیم. (تصویر ۱)

لایه اول یک کانولوشن دو بعدی با kernel_size= (2,2) است و input_shape آن هم به اندازه ابعاد داده‌هاست.

یک فیلتر 2*2 بر روی ماتریس ورودی convolut میشود و هدف پیدا کردن بهترین فیلتر خواهد بود.

در لایه بعدی یک لایه MaxPooling قرار داده شد که ماکزیمم را نگه میدارد و بقیه را دور میریزد که باعث کوچکتر شدن ابعاد ماتریس میشود.

همچنین به دلیل جلوگیری از overfitting از regularization و لایه Dropout استفاده کردیم.

بعد از لایه‌های convolution و MaxPooling یک لایه Flatten قرار میدهیم تا تمام ابعاد ماتریس را flat کند تا بتوانیم در ادامه از لایه fully connected استفاده کنیم.

در لایه آخر هم به تعداد کنگوری‌ها نورون قرار میدهیم و از تابع فعالساز softmax استفاده میکنیم. تابع softamx یک توزیع احتمال میدهد؛

پس مدلی که آموزش داده شد یک ورودی میگیرد که MFCC صوت‌ها است و طبق توضیحاتی که داده شد بر روی آنها پردازش انجام میدهد و در نهایت یک توزیع احتمال به عنوان خروجی برمیگرداند که نشان میدهد هر کتگوری چقدر احتمال دارد مربوط به صوت ورودی باشد. پس بیشترین احتمال را به عنوان کتگوری مربوطه برمیگردانیم.

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(64, kernel_size=(2, 2), activation='relu', input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3])))
model.add(tf.keras.layers.MaxPooling2D((2, 2), padding='same'))
model.add(tf.keras.layers.Conv2D(32, kernel_size=(2, 2), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.MaxPooling2D((2, 2), padding='same'))
model.add(tf.keras.layers.Conv2D(32, kernel_size=(2, 2), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.MaxPooling2D((2, 2), padding='same'))
model.add(tf.keras.layers.Conv2D(128, kernel_size=(2, 2), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.MaxPooling2D((2, 2), padding='same'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(64, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(len(categories), activation='softmax'))
```

تصویر ۱- معماری شبکه عصبی

با ۲۰۳ ایپاک به نتایج زیر رسیدیم:

```
# evaluate network on test set
test_loss, test_acc = model.evaluate(X_test, y_test)
print("\ntest loss: {}, test accuracy: {}".format(test_loss, 100*test_acc))
```

```
27/27 [==============================] - 0s 6ms/step - loss: 0.4871 - accuracy: 0.8877

test loss: 0.4871078431606293, test accuracy: 88.77314925193787
```

تصویر ۲- دقت و لاس مدل

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.90	0.76	0.83	72
1	0.86	0.85	0.85	72
2	0.92	0.99	0.95	72
3	0.85	0.79	0.82	72
4	0.93	0.92	0.92	72
5	0.93	0.89	0.91	72
6	0.78	0.86	0.82	72
7	0.90	0.85	0.87	72
8	0.87	0.96	0.91	72
9	0.93	0.93	0.93	72
10	0.86	0.90	0.88	72
11	0.95	0.96	0.95	72
accuracy			0.89	864
macro avg	0.89	0.89	0.89	864
weighted avg	0.89	0.89	0.89	864

تصویر ۳- گزارش

تصویر ٤ confusion matrix