به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین امتیازی

پرسش	نام دانشجو	ثمین سلوکی
پرسش	شماره دانشجویی	[redacted]

فهرست

شکل‌ها

پرسش) کلمه بیدار باش

۳-۱) جمع آوری داده

هدف این پروژه حل مسئله ی طبقه بندی صدا به دو گروه کلمه ی بیدار باش و سایر اصوات است. کلمه ی بیدار باش در این تمرین، کلمه ی "ثمین" در نظر گرفته شده است. در مرحله ی جمع آوری داده، برای جمع آوری داده های کلاس ۱، کلاس کلمه ثمین، تابع record_wake_word نوشته شد. در این تابع با استفاده از کتابخانه ی sounddevice وscipy ، ۱۰۰ صوت ضبط شد و با پسوند wav دخیره شد. در طی این ۱۰۰ بار سعی شد که کلمه به انواع متفاوت با صدای کم و زیاد و متوسط تلفظ شود تا بتوان دیتاست خوبی برای آموزش مدل تهیه کرد. لازم به ذکر است که برای ضبط صدا، از jupyter notebook نصب شده بر روی لپتاپ استفاده شد و برای انجام پردازش داده و آموزش مدل، داده ها به google colab انتقال داده شدند.

```
def record_wake_word(save_path,n_times=100):
  input("To start audio recording press Enter: ")
  for i in range(n_times):
    fs= 44100 #sample rate (how many times per second a sound is sampled)
    seconds= 2 #length of the audio
    myrecording=sd.rec(int(seconds*fs),samplerate=fs,channels=2)
    sd.wait()
    write(save_path + str(i)+ ".wav" ,fs,myrecording)
    input(f"Press to record next or to stop press ctrl+c({i+1}/{n_times})")
```

Figure ۱: تابع ضبط کلمه ی بیدارباش- کلاس ۱

برای ضبط داده های کلاس صفر، صد بار کلمات دیگری که بعضا به کلمه ی بیدارباش (ثمین) نزدیکی داشتند ضبط شدند. این کلاس شامل اصوات تلفظ کلماتی چون ثمیر، امیر، امین، یاسمین، سمن، سمانه و.... ضبط شدند. همچنین برای ایجاد تنوع برای این کلاس تعدادی از داده ها، صوت ضبط شده ی محیط هستند. برای این ضبط از کد زیر استفاده شد:

```
def record_background_noise(save_path,n_times=100):
  input("To start audio recording press Enter: ")
  for i in range(n_times):
    fs= 44100
    seconds= 2
    myrecording=sd.rec(int(seconds*fs),samplerate=fs,channels=2)
    sd.wait()
    write(save_path + str(i)+ ".wav" ,fs,myrecording)
    input(f"Current;y on : {i+1}/{n_times}")
```

Figure ۲: تابع ضبط کلاس صفر

لازم به ذکر است که همه ی اصوات ضبط شده به طول دو ثانیه هستند.

۳-۲) پیش پردازش و استخراج ویژگی

مفاهیم

پردازش داده های صوتی شامل بکارگیری، تجزیه و تحلیل سیگنال های صوتی گرفته شده در قالب دیجیتال است. هدف اصلی، استخراج اطلاعات معنی‌دار از صوت است که می‌تواند برای تجزیه و تحلیل بیشتر استفاده شود. با تبدیل سیگنال‌های صوتی به داده‌های عددی، می‌توانیم از روش‌های ریاضی و آماری برای کشف الگوها و بینش‌هایی استفاده کنیم که مستقیماً از صوت خام برداشت نمی‌شوند.

پس از جمع آوری فایل های صوتی، اولین مرحله ی آماده سازی دیتا، تبدیل کردن صوت های ضبط شده با پسوند wav به قالب عددی است. در مرحله ی بعد به resampling اقدام می شود. resampling تبدیل یک سیگنال صوتی از یک نرخ نمونه به نرخ نمونه دیگر است. این تبدیل شامل درونیابی سیگنال صوتی برای مطابقت با نرخ نمونه مورد نظر است، که ممکن است شامل کاهش نمونه (کاهش نرخ نمونه- reducing the sample rate) یا افزایش نمونه (افزایش نرخ نمونه) باشد. در مرحله بعد به Normalization دامنه سیگنال صوتی می پردازیم. این مرحله برای اطمینان از اینکه صدای ضبط شده دارای حجم ثابتی است انجام می شود و به کاهش تنوع ناشی از تفاوت در شرایط ضبط کمک می کند. normalization می تواند شامل مقیاس بندی دامنه برای قرار گرفتن در یک محدوده خاص، مانند بین -۱ و ۱، یا تنظیم سیگنال برای دستیابی به سطح بلندی هدف باشد. قدم بعد انجام کاهش نویز است که برای حذف نویز ناخواسته پس زمینه از سیگنال های صوتی، بهبود وضوح و کیفیت صدا استفاده می شود. کاهش موثر نویز نسبت سیگنال به نویز (signal-to-noise ratio) را افزایش می‌دهد و تمرکز مدل‌های یادگیری را بر ویژگی‌های صوتی مرتبط آسان‌تر می‌کند.

سپس می توان Segmentation داد که شامل تقسیم یک صوتی طولانی به بخش های کوتاه تر است. این را می توان با تشخیص سکوت در صدا و تقسیم در آن نقاط، یا با استفاده از پنجره با طول ثابت برای ایجاد بخش های یکنواخت انجام داد. تقسیم بندی به ویژه در برنامه هایی مفید است که بخش های خاصی از صدا نیاز به تجزیه و تحلیل مستقل دارند، مانند تشخیص گفتار، جایی که جملات یا عبارات جداگانه پردازش می شوند.

مرحله ی مهم بعدی، استخراج ویژگی است. استخراج ویژگی، شکل موج های صوتی خام را به نمایشی تبدیل می کند که ویژگی های اساسی صدا را به تصویر می کشد و آنها را برای مدل های یادگیری مناسب می کند. ویژگی‌های معمول شامل spectrograms (نمایش بصری طیف فرکانس در طول زمان)،

Mel-Frequency Cepstral Coefficients (MFCCs)، که طیف کوتاه‌مدت spectrum را نشان می‌دهد)، chroma features (مربوط به ۱۲ کلاس pitch مختلف)، zero-crossing rate (سرعتی که سیگنال تغییر علامت می دهد)، و spectral centroid (مرکز جرم طیفspectrum). این ویژگی‌ها به شناسایی الگوها و ویژگی‌های مرتبط با وظایفی مانند تشخیص گفتار، تجزیه و تحلیل موسیقی و طبقه‌بندی صدا کمک می‌کنند. در ادامه به جزئیات بیشتر می پردازیم:

- **Spectrogram**

نمایش بصری طیف فرکانس ها در یک سیگنال صوتی است و با زمان تغییر می کند و نمای جامعی از نحوه تغییر محتوای فرکانس در طول زمان ارائه می دهد.

- **Mel-Frequency Cepstral Coefficients (MFCCs)**

MFCC ضرایبی هستند که مجموعاً طیف توان کوتاه مدت یک سیگنال صوتی را با استفاده از فاصله فرکانسی در مقیاس mel نشان می دهند و پاسخ گوش انسان را به فرکانس های مختلف، تقریب زده و نزدیک می کنند.

- **Chroma Features**

ویژگی های کروما نشان دهنده توزیع انرژی در ۱۲ کلاس pitch است که ویژگی های هارمونیک و ملودیک را به تصویر می کشد.این ویژگی در کارهای مرتبط با موسیقی مانند تشخیص آکورد و تشخیص کلید مفید است.

- **Zero-Crossing Rate**

این نرخ، نشان دهنده ی سرعت تغییر علامت سیگنال از مثبت به منفی یا بالعکس است و نشان دهنده نویز یا ناهمواری یک سیگنال است که در طبقه بندی ژانر گفتار و موسیقی مفید است.

- **Spectral Centroid**

مرکز جرم طیف، که نشان می دهد مرکز فرکانس در کجا قرار دارد، معیاری از روشنایی یک صدا را ارائه می دهد که می تواند برای تجزیه و تحلیل تامبر و طبقه بندی صدا مهم باشد.

- **Spectral Bandwidth**

پهنای باند طیفی، اندازه گیری گسترش فرکانس هاست و محدوده فرکانس های موجود در سیگنال را نشان می دهد که برای تشخیص انواع مختلف صداها مفید است.

- **Pitch**

فرکانس اساسی درک شده صداست و برای پردازش موسیقی و تجزیه و تحلیل صدای گفتار ضروری است.

برخی از ویژگی‌های صوت می‌توانند در برخی کاربردها مفید باشند و در بعضی مضر. به طور مثال نویز، در تشخیص گفتار و تشخیص رویدادهای صوتی، می تواند ویژگی های مهم را پنهان کند و شناسایی دقیق و پردازش سیگنال را برای مدل ها دشوار کند. از طرفی در تقویت داده ها برای آموزش مدل های قوی، می تواند با شبیه سازی شرایط مختلف دنیای واقعی به بهبود تعمیم مدل کمک کند. همین طور Pitch Variations برای تشخیص گوینده، تغییرات قابل توجه Pitchمی تواند دقت شناسایی افراد را بر اساس صدای آنها کاهش دهد ولی در سنتز و تجزیه و تحلیل موسیقی، تغییرات زیر و بمی برای ایجاد ملودی و هارمونی که از عناصر اساسی آهنگسازی هستند، بسیار مهم است.Volume Level Variations در برنامه های گفتار به متن، تغییرات زیاد در حجم می تواند منجر به رونویسی های متناقض و کاهش دقت شود. ولی در فشرده سازی محدوده دینامیکی برای تولید موسیقی، تغییرات در سطوح صدا را می توان برای دستیابی به کیفیت صدای مطلوب و تأثیر احساسی تنظیم کرد.

Data augmentation

برای تولید مصنوعی داده‌های صدا، می‌توانیم از موارد زیر استفاده کنیم:

- **noise injection**

یکی از روش های رایج اضافه کردن نویز است که در آن انواع تصادفی یا خاص نویز به سیگنال صوتی اصلی اضافه می شود. این فرآیند شرایط محیطی دنیای واقعی را شبیه‌سازی می‌کند و با قرار دادن آن در پس‌زمینه‌های صوتی مختلف، توانایی مدل را برای تعمیم بهبود می‌بخشد. نویز بسته به کاربرد می تواند نویز سفید، نویز صورتی یا انواع دیگر باشد و سطح نویز اضافه شده می تواند برای کنترل نسبت سیگنال به نویز متفاوت باشد.

- **shifting time**

زمان جابجایی شامل تغییر هم ترازی زمانی سیگنال های صوتی با کشش یا فشرده سازی آنها است. این تکنیک ، سرعت پخش صدا را تغییر می دهد و در عین حال زیر و بم آن را حفظ می کند. با اعمال تغییرات زمانی تصادفی در یک محدوده مشخص، مدل یاد می گیرد که الگوهای صوتی را در مقیاس های زمانی مختلف تشخیص دهد. این روش تقویت به ویژه برای کارهایی که پویایی زمانی صدا بسیار مهم است، مانند تشخیص گفتار یا تجزیه و تحلیل موسیقی مفید است.

- **changing speed**

تغییر سرعت به تغییر سرعت پخش سیگنال های صوتی بدون تغییر pitchآنها اشاره دارد. این تکنیک معمولاً برای شبیه سازی تغییرات در سرعت صحبت کردن یا آواز خواندن و همچنین تغییرات سرعت برای محتوای موسیقی استفاده می شود. با تنظیم سرعت، مدل یاد می‌گیرد که در سرعت های مختلف صحبت یا سرعت‌های موسیقی تعمیم بهتری داشته باشد، در نتیجه عملکرد خود را در کارهایی که نیاز به تشخیص تغییرناپذیر سرعت دارند، مانند رونویسی گفتار یا ردیابی ضربان در تجزیه و تحلیل موسیقی، بهبود می‌بخشد.

- **changing pitch**

تغییر گام یکی دیگر از رویکردهای تقویتی است که فرکانس اساسی سیگنال صوتی را تغییر می دهد و در عین حال ساختار زمانی آن را حفظ می کند. این تکنیک از طریق pitch shifting به دست می آید، که در آن کل طیف فرکانس های صوتی به بالا یا پایین منتقل می شود. با معرفی تغییرات در pitch، این مدل برای تغییرات زیر در سناریوهای دنیای واقعی، مانند بلندگوهای مختلف یا آلات موسیقی، قوی‌تر می‌شود.

پیاده سازی

آرایه سری زمانی-Time Series Array

آرایه سری زمانی که سیگنال صوتی را نشان می دهد دنباله ای از مقادیر عددی است که با دامنه موج صوتی در فواصل زمانی گسسته مطابقت دارد. در صدای دیجیتال، از امواج صوتی پیوسته در فواصل زمانی معین نمونه برداری می شود تا این آرایه ایجاد شود. هر مقدار در آرایه منعکس کننده فشار موج صوتی در یک نقطه خاص از زمان است. به عنوان مثال، اگر یک فایل صوتی با سرعت بالا نمونه برداری شود، آرایه سری زمانی مقادیر بیشتری داشته و جزئیات بیشتری از سیگنال صوتی را گزارش می دهد.

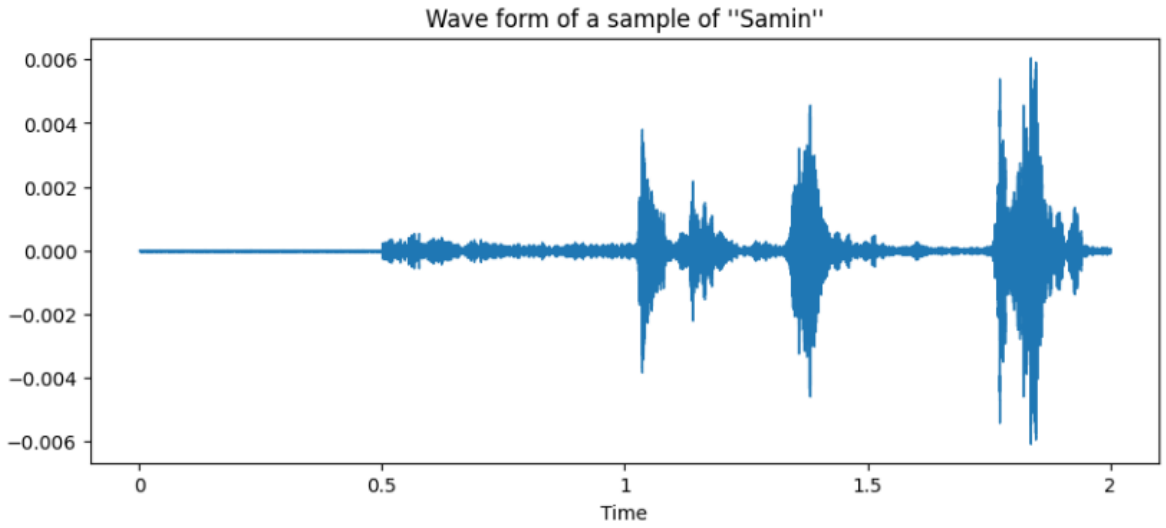
نرخ نمونه یک پارامتر مهم در پردازش صوت دیجیتال است. این نرخ تعداد نمونه های گرفته شده در هر ثانیه از یک سیگنال صوتی پیوسته را مشخص می کند تا آن را به سیگنال دیجیتال تبدیل کند. نرخ نمونه معمولی ۴۴,۱۰۰ هرتز (۴۴.۱ کیلوهرتز) است که به این معنی است که صدا ۴۴,۱۰۰ بار در ثانیه نمونه برداری می شود. نرخ نمونه بالاتر می تواند جزئیات بیشتری را ثبت کند، اما همچنین به فضای ذخیره سازی و قدرت محاسباتی بیشتری نیاز دارد. تنظیم و انتخاب این نرخ بر مسائل متعددی تاثیر گذار است چون: تأثیر بر وضوح زمانی سیگنال صوتی (بر محاسبات مربوط به زمان، فرکانس و سایر ویژگی های صوتی تأثیر می گذارد.)

بدین منظور از دستور زیر استفاده کردیم:

```
data_ , sample_rate = librosa.load(single_file)
```

هنگامی که فایل صوتی بارگذاری شد و به یک آرایه سری زمانی با نرخ نمونه مشخص تبدیل شد، می توان تحلیل ها و دستکاری های مختلفی را انجام داد. به عنوان مثال، ویژگی هایی مانند فرکانس Mel (MFCCs) را می توان استخراج کرد تا صدا را به گونه ای نشان دهد که برای وظایف یادگیری ماشین مناسب باشد. درباره ی MFCCs در ادامه بیشتر توضیح خواهیم داد. با تبدیل داده‌های صوتی خام به یک قالب ساختاریافته از طریق «librosa.load»، می‌توانیم از قدرت پردازش سیگنال دیجیتال و یادگیری عمیق برای تجزیه و تحلیل و تفسیر سیگنال‌های صوتی به طور مؤثر استفاده کنیم.

در تصویر زیر نمونه ای از آرایه ی پلات شده از کلمه ی بیدار باش (در این پروژه کلمه‌ی "ثمین") را مشاهده می کنید:



wave form :۳ Figure

اسپکتروگرام - Spectrogram

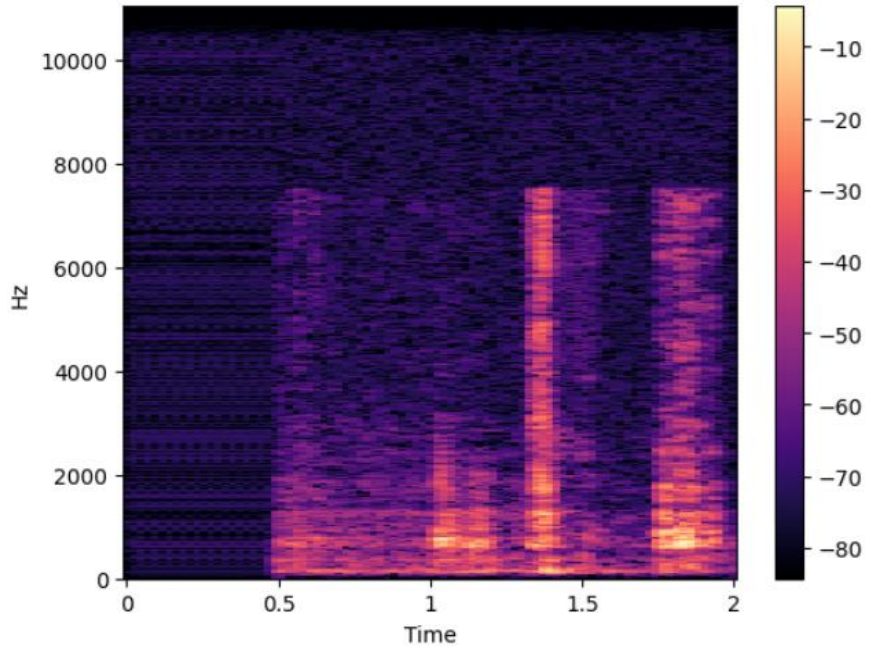
برای نمایش بصری سطوح انرژی یک سیگنال در فرکانس های مختلف در طول زمان، از اسپکتروگرام استفاده می شود. اسپکتروگرام ، تغییرات سطوح انرژی در طول زمان و همچنین قدرت نسبی فرکانس های موجود در یک شکل موج را نشان می دهد. اسپکتروگرام یک نمایش بصری از قدرت سیگنال یا بلندی(loudness) سیگنال در طول زمان در فرکانس های مختلف موجود در یک شکل موج خاص است.

اسپکتروگرام معمولاً به صورت نقشه حرارتی نمایش داده می‌شود که از رنگ یا روشنایی برای نشان دادن قدرت سیگنال استفاده می‌کنند.

در این بخش از کد زیر استفاده کردیم:

```
data= librosa.stft(data)
Xdb = librosa.amplitude_to_db(abs(data))
librosa.display.specshow(Xdb,sr=sample_rate, x_axis='time', y_axis='hz')
plt.colorbar()
```

تابع .stft داده ها را به یک تبدیل فوریه کوتاه مدت تبدیل می کند که به ما امکان می دهد دامنه یک فرکانس معین را در یک زمان خاص تعیین کنیم. با استفاده از STFT، می‌توانیم دامنه فرکانس‌های متعددی را که در یک سیگنال صوتی در یک زمان خاص وجود دارند، شناسایی کنیم. محور فرکانس در محور عمودی نشان داده شده از ۰ تا ۱۰ کیلوهرتز است و زمان کلیپ صوتی در محور افقی نشان داده شده است. نتیجه را در تصویر زیر مشاهده می کنید:

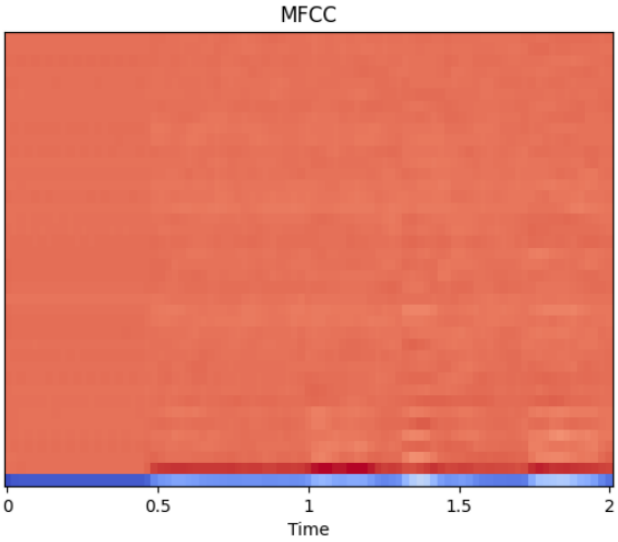


STFT :۴ Figure

در ادامه، از MFCCs استفاده کردیم. توضیحات مربوط به MFCC و کاربرد و عملکردشان را پیش از این توضیح دادیم. برای MFCC از کد زیر استفاده کردیم.

```
mfccs = librosa.feature.mfcc(y=data, sr=sample_rate, n_mfcc=40)
```

نتیجه ی به تصویر کشیده شده را در زیر مشاهده می کنید:



MFCC :۵ Figure

برای تقویت داده های موجود از data augmentation استفاده کردیم. این داده افزایی شامل اضافه کردن نویز، shift time ،تغییرpitch ، تغییر speed بود:

```
def noise(d):
  noise_factor=0.005
  noise = np.random.randn(len(d))
  augmented_data = d + noise_factor * noise
  augmented_data = augmented_data.astype(type(d[0]))
  return augmented_data

def time_shift(d, sampling_rate=16000, shift_max=0.5, shift_direction='both'):
  shift= np.random.randint(-sampling_rate * shift_max, sampling_rate * shift_max)
  if shift_direction == 'right':
    shift= abs(shift)
  elif shift_direction == 'both':
    direction= np.random.randint(0, 2)
    if direction == 1:
      shift= -shift
  augmented_data = np.roll(d, shift)
  if shift>0:augmented_data[:shift] = 0
  else: augmented_data[shift:] = 0
  return augmented_data

def augment_pitch(d):
  pitch_factor=2
  n_steps = int(pitch_factor * 12)  #convert pitch_factor to semitones
  return librosa.effects.pitch_shift(d,sr=16000,n_steps=n_steps)

def augment_speed(d, speed_factor=1.2):
  return librosa.effects.time_stretch(d,rate=speed_factor)


augmentation_functions = [noise,time_shift,augment_pitch,augment_speed]
def apply_random_augmentation(d):
  augmentation_function = random.choice(augmentation_functions)
  augmented_audio = augmentation_function(d)
  return augmented_audio
```

data augmentation :۶ Figure

برای حفظ توزیع داده های اولیه از ترکیب روش های متفاوت data augmentation بر روی یک داده اجتناب کردیم. اعمال داده افزایی بدین صورت بود که به صورت رندوم بر هر داده، یک روش خاص داده افزایی اعمال می شد و دیتای آگمنت شده به همراه دیتای اورجینال به لیست داده اصلی که بعدا برای آموزش و تست استفاده می شد، اضافه شد.

بعد از اعمال داده افزایی، به استخراج ویژگی پرداختیم. ویژگی‌های استخراج‌شده از فایل‌های صوتی، MFCCs است. به طور خاص، برای هر فایل صوتی، کد ۴۰ MFCC را با استفاده از تابع librosa.feature.mfcc محاسبه کردیم، که سیگنال صوتی خام را به نمایشی تبدیل می‌کند که طیف قدرت spectrum را به گونه‌ای که شنوایی انسان را تقریب میزدند ضبط می‌کند. سپس MFCCها با میانگین‌گیری آن‌ها در طول زمان پردازش می‌شوند. داده‌های داده حاصل، بردارهای ویژگی MFCC و برچسب‌های کلاس مربوط به آنهاست. لازم به ذکر است که مرحله ی استخراج ویژگی بعد از مرحله ی داده افزایی انجام شد.

۳-۳) طراحی شبکه عصبی

در این مرحله به مدلسازی می پردازیم.

برای شناسایی کلمه ی بیدارباش در طبقه بندی صدا با صداهای با طول ثابت، یک شبکه عصبی کانولوشن (CNN) انتخاب ایده‌آل ماست. CNN ها در ثبت الگوهای محلی در داده ها عملکرد عالی ای دارند ، که برای تشخیص ویژگی های صوتی کوتاه و متمایز بسیار مهم است. این local feature extraction در نمایش فرکانس زمانی و برای شناسایی کلمات کلیدی مفید است. همچنین در مقایسه با شبکه‌های fully connected ، CNNها از نظر پارامتر کارآمدتر هستند و برای اهداف real-time مناسب هستند. معماری‌های CNN را می‌توان با تنظیم تعداد لایه‌ها و فیلترها کوچک‌تر کرد و بسته به پیچیدگی و محدودیت‌های منابع، امکان انعطاف‌پذیری را فراهم کرد. شبکه کانولوشنی شامل تبدیل صدای با طول ثابت به MFCCs است(به عنوان ورودی). با استفاده از لایه‌های کانولوشنال دوبعدی، شبکه می‌تواند سلسله مراتب و الگوهای فضایی، مانند واج‌ها یا کلمات کلیدی خاص را در spectrogram شناسایی کند. لایه های Pooling ، ابعاد و پیچیدگی محاسباتی را کاهش می دهند و لایه های fully connected به دنبال لایه خروجی softmax طبقه بندی نهایی را انجام می دهند. این معماری به دلیل قابلیت های استخراج ویژگی محلی، کارایی پارامتر و مقیاس پذیری سودمند است. CNN ها در استخراج الگوهای فرکانس زمانی ماهر هستند و آنها را برای شناسایی کلمات کلیدی مناسب می کند. به طور کلی خلاصه می کنیم که یک رویکرد مبتنی بر CNN، با پیش‌پردازش مناسب، راه‌حلی قوی برای شناسایی کلمات کلیدی در وظایف طبقه‌بندی صوتی با طول ثابت ارائه می‌دهد.

برای شناسایی کلمات کلیدی در طبقه بندی صدا با ورودی های صوتی با طول متغیر، ترکیبی از شبکه RNN و CNN می تواند موثر باشد به طوری که استخراج ویژگی با CNN انجام شود و صوت را به نمایش

فرکانس زمانی، مانند spectrogram یا MFCC تبدیل کند. لایه‌های CNN الگوها و ویژگی‌های مهم را در فریم‌های کوچک صدا شناسایی می‌کند و ورودی با ابعاد بالا را با ابعاد پایین‌تر و قابل کنترل‌تر کاهش می‌دهند. سپس خروجی CNN به RNN به عنوان یک شبکه LSTM یا GRU وارد شود. RNN ها در مدیریت داده های sequential ماهر هستند و می توانند وابستگی ها و الگوهای زمانی را در طول زمان ثبت کنند. به عبارت دیگر، لایه‌های RNN دینامیک و وابستگی‌های زمانی را در دنباله صوتی ضبط می‌کنند، که برای تشخیص کلمات کلیدی در ورودی‌های با طول متغیر مهم است. این رویکرد ترکیبی صدای با طول متغیر را به طور موثر مدیریت می‌کند و آن را برای سناریوهای دنیای واقعی که مدت زمان صدا ممکن است متفاوت باشد، مناسب می‌کند.این معماری ترکیبی، از نقاط قوت معماری‌های CNN و RNN استفاده می‌کند.

برای انجام این پروژه دو مدل شبکه عصبی اجرا شد:

مدل ۱) شبکه متشکل از لایه های dense و dropout که پیچیدگی و افزونگی کمتری نسبت به معماری های کانولوشنی دارد.

مدل ۲) شبکه متشکل از لایه های Conv2D و MaxPooling2D و BatchNormalization و Dense و dropout

مدل ۱)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               10496

 activation (Activation)     (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 activation_1 (Activation)   (None, 256)               0

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 76802 (300.01 KB)
Trainable params: 76802 (300.01 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure ۷: معماری مدل ۱

مدل یک، با معماری روبرو، با ۷۶,۸۰۲ پارامتر در طی ۱۱۰ ایپاک و با batch_size=۳۲ با ۸۰ درصد داده ی تست و تقسیم ۹۰-۱۰ داده ی آموزش و اعتبارسنجی آموزش با بهینه ساز آدام ، آموزش دید.

نتیجه ی دقت این مدل بر داده ی آموزش ۹۰٪ بود.

گزارش هزینه و دقت مدل را در زیر مشاهده میکنید:

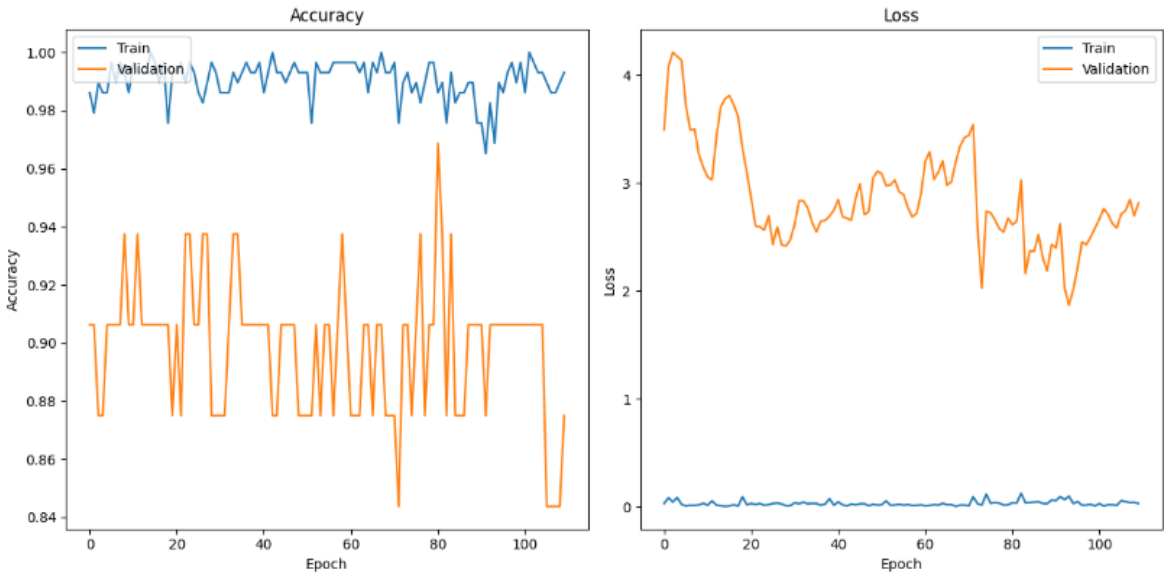


Figure ۸: گزارش هزینه و دقت در داده آموزش و اعتبارسنجی - مدل ۱

نتیجه عملکرد بر داده ی تست را در زیر مشاهده می کنید:

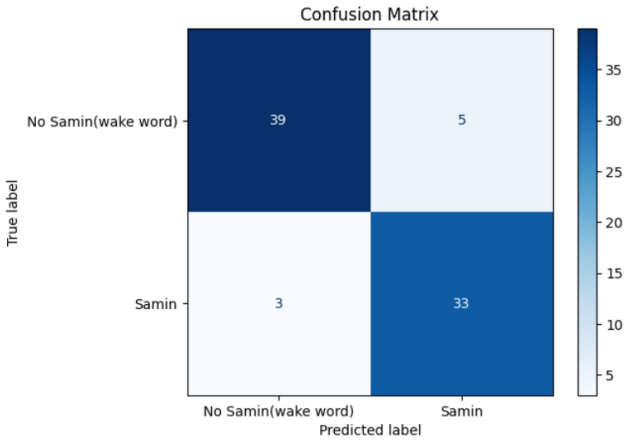


Figure ۹: مدل ۱- **confusion matrix**

در این مدل، یک تفکیک دقیق از عملکرد مدل بر اساس گزارش طبقه بندی آمده است. دقت کلی مدل ۹۰ درصد است، یعنی مدل توانسته ۹۰ درصد نمونه های صوتی را به درستی طبقه بندی کند. recall به چند مورد از موارد مثبت واقعی اشاره دارد که مدل به درستی شناسایی کرده است (جلوگیری از منفی کاذب). در مورد این مدل، recall و precision هر دو در حدود ۰.۹۰ برای دو کلمه کلیدی است، به این معنی که مدل در هر دو معیار عملکرد خوبی داشته است.

```
Model Classification Report:

3/3 [==============================] - 0s 5ms/step
              precision    recall  f1-score   support

           0       0.93      0.89      0.91        44
           1       0.87      0.92      0.89        36

    accuracy                           0.90        80
   macro avg       0.90      0.90      0.90        80
weighted avg       0.90      0.90      0.90        80
```

Figure ۱۰: گزارش عملکرد مدل ۱

به طور کلی، این نتایج نشان می‌دهد که شبکه عصبی fully connected در طبقه‌بندی کلمات کلیدی در نمونه‌های صوتی از دقت بالایی برخوردار است و قادر است از مثبت کاذب و منفی کاذب با نرخ مشابهی جلوگیری کند.

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 126, 398, 32)      320

 max_pooling2d_15 (MaxPooli  (None, 63, 199, 32)       0
 ng2D)

 batch_normalization_15 (Ba  (None, 63, 199, 32)       128
 tchNormalization)

 conv2d_16 (Conv2D)          (None, 61, 197, 64)       18496

 max_pooling2d_16 (MaxPooli  (None, 30, 98, 64)        0
 ng2D)

 batch_normalization_16 (Ba  (None, 30, 98, 64)        256
 tchNormalization)

 conv2d_17 (Conv2D)          (None, 28, 96, 128)       73856

 max_pooling2d_17 (MaxPooli  (None, 14, 48, 128)       0
 ng2D)

 batch_normalization_17 (Ba  (None, 14, 48, 128)       512
 tchNormalization)

 flatten_5 (Flatten)         (None, 86016)             0

 dense_10 (Dense)            (None, 128)               11010176

 dropout_5 (Dropout)         (None, 128)               0

 dense_11 (Dense)            (None, 2)                 258

=================================================================
Total params: 11104002 (42.36 MB)
Trainable params: 11103554 (42.36 MB)
Non-trainable params: 448 (1.75 KB)
_________________________________________________________________
```

Figure ۱۱: معماری مدل ۲

مدل ۲)

مدل ۲ با معماری روبر با ۱۱,۱۰۳,۵۵۴ پارامتر در طی ۱۰۰ ایپاک به بهینه ساز آدام با batch_size=32 آموزش دید.

نتیجه ی دقت این مدل بر داده ی آموزش ۸۵٪ بود.

گزارش هزینه و دقت مدل را در زیر مشاهده می کنید:

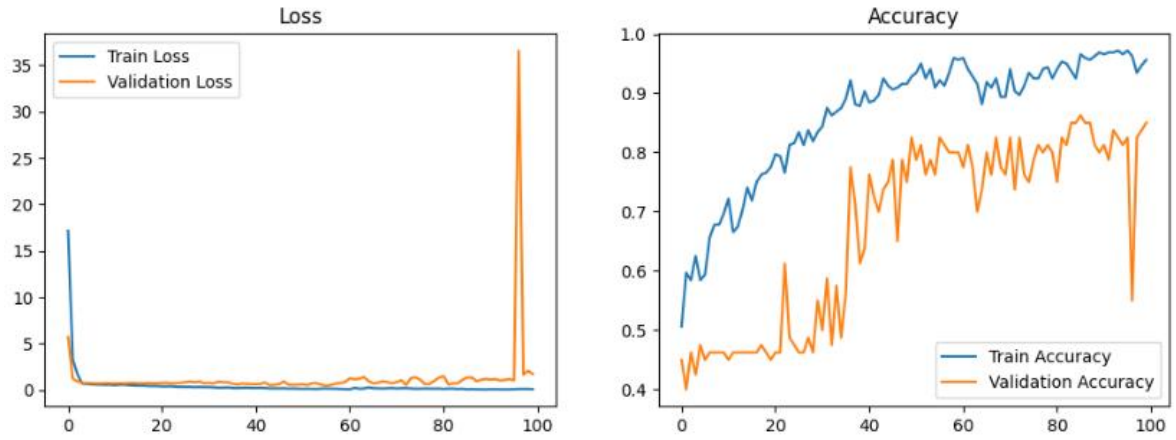


Figure ۱۲: گزارش هزینه و دقت در داده آموزش و اعتبارسنجی - مدل ۲

به نظر می رسد دقت تمرین در طول دوره ها به طور پیوسته در حال افزایش است.این نشان می‌دهد که مدل به تدریج در حال یادگیری طبقه‌بندی صحیح نمونه‌های صوتی برای تشخیص کلمه کلیدی است. همچنین روند کاهشی هزینه نشان می دهد که مدل به طور موثر خطاهای خود را به حداقل می رساند و پیش بینی های خود را بهبود می دهد. با توجه به نمودار مربوط به دقت، احتمال مقداری بیش برازش می دهیم.

نتیجه عملکرد بر داده ی تست را در زیر مشاهده می کنید:

```
Model Classification Report:

3/3 [==============================] - 0s 15ms/step
              precision    recall  f1-score   support

           0       0.94      0.77      0.85        44
           1       0.77      0.94      0.85        36

    accuracy                           0.85        80
   macro avg       0.86      0.86      0.85        80
weighted avg       0.87      0.85      0.85        80
```

Figure **۱۳: گزارش عملکرد مدل ۲**

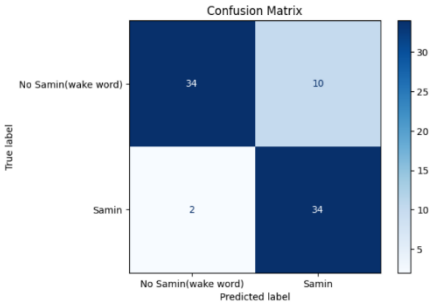


Figure ۱۴: مدل ۲- **confusion matrix**

بر اساس گزارش طبقه بندی، مدل CNN در کل عملکرد خوبی داشته است. دقت کلی مدل ۸۶ درصد است که به این معنی است که مدل توانسته ۸۶ درصد از نمونه های صوتی را به درستی طبقه بندی کند.

Precision به نسبت مثبت های واقعی به تعداد کل پیش بینی های مثبت اشاره دارد. recall به نسبت موارد مثبت واقعی به تعداد کل موارد مثبت واقعی اشاره دارد. در مورد این مدل، امتیاز Precision و recall هر دو در حدود ۰.۸۵ برای کلمه کلیدی است، به این معنی که مدل در اجتناب از مثبت کاذب (طبقه بندی نمونه های منفی به عنوان مثبت) و منفی کاذب (طبقه بندی نمونه های مثبت به عنوان منفی) خوب بود. به طور کلی، این نتایج نشان می‌دهد که مدل CNN در طبقه‌بندی صوتی تشخیص کلمات کلیدی از دقت کلی بالایی برخوردار است و قادر است از مثبت کاذب و منفی کاذب با نرخ مشابهی جلوگیری کند.

مقایسه عملکرد دو مدل

در نهایت به مقایسه عملکرد ۲ مدل میپردازیم. مدل ۱ که معماری ساده تری داشته و از لایه ی کانولوشنی استفاده نکرده بود، عمکرد دقت ۹۰ درصد داشته و مدل ۲ با لایه های کانولوشنی دقت ۸۵٪ داشته است. اگر معیار مقایسه تنها دقت باشد، مدل اول عملکرد بهتری داشته. این عملکرد بهتر مدل ۱ (تعداد پارامتر یادگرفته شده: ۱۱,۱۰۳,۵۵۴) در حالی است که پارامتر های بسیار کمتری نسبت به مدل ۲(تعداد پارامتر یادگرفته شده: ۷۶,۸۰۲) داشته است. همچنین به این اشاره می کنیم که مدل ۱ با لایه های fully connected در زمان کوتاهتری آموزش دید اما مدل ۲ در زمان طولانی تر. در این پروژه مشخص شد که همواره افزایش پیچیدگی مدل و افزونگی مدل، منجر به بهبود عملکرد مدل نمی شود. مدل ۱ دقت بهتر با تعداد پارامتر کمتر، پیچیدگی و هزینه محاسبه ی کمتری داشته و همچنین توانسته در تشخیص کلمه ی بیدار باش، عملکرد بهتری نسبت به مدل با لایه های کانولوشنی داشته باشد.