محمد جواد رنجبر

۸۱۰۱۰۱۱۷۳

تمرین چهارم درس پردازش گفتار

دکتر ویسی

بهار ۱۴۰۳

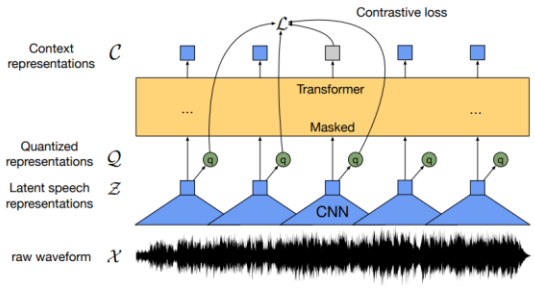
سوال ۱: پژوهش

بازشناسی گفتار

Wav2vec 2.0

معماری مدل

Wav2vec 2.0 از یک معماری ترانسفورمر استفاده می‌کند. در این مدل، یک شبکه عصبی convolutional (کانولوشن) برای استخراج ویژگی‌های اولیه از سیگنال صوتی خام استفاده می‌شود.



شکل ۱معماری Wav2vec 2.0

پس از استخراج ویژگی‌ها، این ویژگی‌ها به یک شبکه ترانسفورمر تغذیه می‌شوند که وظیفه یادگیری بازنمایی‌های گفتار را دارد.

یادگیری خودنظارتی:

مدل Wav2vec 2.0 به جای استفاده از برچسب‌های دستی برای آموزش، از یادگیری خودنظارتی استفاده می‌کند. در این روش، مدل با استفاده از بخشی از داده‌های بدون برچسب به صورت پیش‌بینی‌های مبتنی بر خود آموزی می‌کند.

این مدل ابتدا سیگنال صوتی را به بخش‌های کوچک تقسیم کرده و سپس این بخش‌ها را به صورت تصادفی ماسک می‌کند. مدل باید ماسک‌ها را پیش‌بینی کند که این فرآیند باعث می‌شود مدل بازنمایی‌های گفتاری قدرتمندی یاد بگیرد.

فازهای آموزشی:

Pretraining

مدل در این مرحله از داده‌های خام صوتی برای یادگیری بازنمایی‌های کلی صوتی استفاده می‌کند.

Fine-tuning

در این مرحله مدل با استفاده از داده‌های برچسب‌دار بهینه‌سازی می‌شود تا بتواند وظایف خاصی مانند تشخیص گفتار را به خوبی انجام دهد.

الگوریتم‌های یادگیری ماشین:

یادگیری خودنظارتی(Self-Supervised Learning)

این الگوریتم در مرحله پیش‌آموزش (pretraining) استفاده می‌شود و مدل را قادر می‌سازد تا بدون نیاز به برچسب‌های دستی، از داده‌های خام بازنمایی‌های مفیدی یاد بگیرد.

یادگیری نظارت‌شده (Supervised Learning)

در مرحله تنظیم دقیق (fine-tuning) از یادگیری نظارت‌شده استفاده می‌شود که در آن مدل با داده‌های برچسب‌دار بهینه‌سازی می‌شود.

مقایسه با پروژه‌های مشابه:

DeepSpeech

معماری DeepSpeech

از یک معماری شبکه عصبی عمیق (RNN) استفاده می‌کند.

روش آموزش

بیشتر متکی به داده‌های برچسب‌دار است و از یادگیری نظارت‌شده استفاده می‌کند.

دقت

دقت DeepSpeech به اندازه Wav2vec 2.0 بالا نیست زیرا نمی‌تواند به اندازه Wav2vec 2.0 از داده‌های بدون برچسب بهره ببرد.

BERT for Speech

معماری BERT for Speech

از معماری مشابه BERT که برای پردازش زبان طبیعی طراحی شده، استفاده می‌کند.

روش آموزش

ترکیبی از یادگیری نظارت‌شده و خودنظارتی.

دقت

این مدل نیز دقت بالایی دارد، اما پیچیدگی پردازش آن بیشتر است و نیاز به منابع بیشتری دارد.

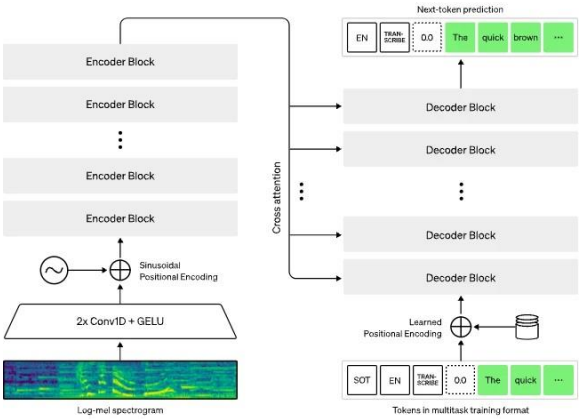
Whisper

یک سیستم شناسایی خودکار گفتار (ASR) است که بر روی ۶۸۰,۰۰۰ ساعت داده نظارت‌شده چندزبانه و چندوظیفه‌ای که از وب جمع‌آوری شده، آموزش دیده است.

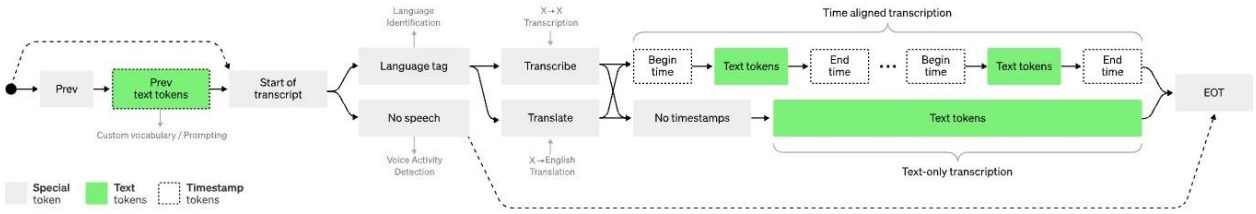
معماری مدل:

Whisperاز یک مدل ترانسفورمر (Transformer) بهره می‌برد. ترانسفورمرها به دلیل قابلیت موازی‌سازی و توانایی درک توالی‌های طولانی، برای وظایف پردازش زبان طبیعی و گفتار بسیار مناسب هستند .این مدل با استفاده از لایه‌های Encoder-Decoder ساختار یافته است که بازنمایی‌های پیچیده‌ای از سیگنال‌های صوتی ایجاد می‌کند. معماری Whisper یک رویکرد ساده انتها به انتها است که به عنوان یک ترانسفورمر رمزگذار-رمزگشا پیاده‌سازی شده است. صدای ورودی به بخش‌های ۳۰ ثانیه‌ای تقسیم می‌شود، به طیف‌نگار لاگ-مل تبدیل

می‌شود و سپس به یک رمزگذار منتقل می‌شود. یک رمزگشا برای پیش‌بینی متن متناظر آموزش داده می‌شود، که با نشانه‌های خاصی ترکیب شده که مدل واحد را به انجام وظایفی مانند شناسایی زبان، نشانه‌گذاری زمانی در سطح عبارت، رونویسی چندزبانه گفتار و ترجمه گفتار به انگلیسی هدایت می‌کنند.



شکل ۲ معماری Whisper



شکل ۳ نحوه کارکرد Whisper

رویکردهای دیگر معمولاً از مجموعه داده‌های آموزشی صوت-متن در کنار یکدیگر با حجم کم استفاده می‌کنند، یا از پیش‌آموزش صوتی گسترده ولی بدون نظارت استفاده می‌کنند. از آنجا که Whisper بر روی یک مجموعه داده بزرگ و متنوع آموزش دیده و به هیچ مجموعه خاصی بهینه‌سازی نشده است، در عملکرد بر روی مجموعه داده LibriSpeech که یک معیار رقابتی مشهور در شناسایی گفتار است، از مدل‌های تخصصی بهتر عمل نمی‌کند. با این حال، وقتی عملکرد بدون‌نیاز به تنظیم Whisper (zero-shot) را در مجموعه داده‌های متنوع اندازه‌گیری می‌کنیم، می‌بینیم که بسیار مقاوم‌تر است و ۵۰٪ کمتر از آن مدل‌ها خطا می‌کند.

حدود یک‌سوم مجموعه داده صوتی Whisper غیرانگلیسی است و به صورت متناوب وظیفه رونویسی در زبان اصلی یا ترجمه به انگلیسی را به عهده دارد.

یادگیری خودنظارتی و نیمه‌نظارتی:

مانندWav2vec 2.0، Whisper نیز از یادگیری خودنظارتی استفاده می‌کند. با استفاده از داده‌های بدون برچسب، مدل یاد می‌گیرد که بازنمایی‌های صوتی موثری تولید کند. همچنین، از یادگیری نیمه‌نظارتی بهره می‌برد که ترکیبی از داده‌های برچسب‌دار و بدون برچسب را برای بهینه‌سازی مدل استفاده می‌کند.

فازهای آموزشی:

Pretraining

مدل ابتدا با داده‌های بزرگ و بدون برچسب آموزش می‌بیند تا بتواند بازنمایی‌های عمومی و گسترده‌ای از صوت یاد بگیرد.

Fine-tuning

در این مرحله، مدل با داده‌های برچسب‌دار دقیق‌تر بهینه‌سازی می‌شود تا بتواند وظایف خاص مانند تشخیص گفتار را با دقت بالاتری انجام دهد.

الگوریتم‌های یادگیری ماشین:

یادگیری خودنظارتی(Self-Supervised Learning)

در مرحله پیش‌آموزش، مدل با استفاده از داده‌های بدون برچسب، الگوهای عمومی گفتاری را یاد می‌گیرد.

یادگیری نیمه‌نظارتی(Semi-Supervised Learning)

در این روش، داده‌های برچسب‌دار و بدون برچسب به طور ترکیبی استفاده می‌شوند تا مدل بهینه‌سازی شود.

یادگیری نظارت‌شده(Supervised Learning)

در مرحله تنظیم دقیق، داده‌های برچسب‌دار برای بهبود دقت مدل در وظایف خاص به کار می‌روند.

مقایسه با پروژه‌های مشابه:

Wav۲vec ۲.۰ (Facebook AI Research)

معماری

هر دو مدل از معماری ترانسفورمر استفاده می‌کنند، اما ساختارهای دقیق و فرآیندهای آموزش ممکن است متفاوت باشند.

روش آموزش

هر دو مدل از یادگیری خودنظارتی برای استفاده از داده‌های بدون برچسب بهره می‌برند.

دقت

هر دو مدل دقت بالایی دارند، اما Whisper ممکن است در برخی موارد به دلیل استفاده از داده‌های گسترده‌تر و تکنیک‌های نیمه‌نظارتی عملکرد بهتری داشته باشد.

DeepSpeech (Mozilla)

معماری DeepSpeech

از شبکه‌های عصبی عمیق (RNN) استفاده می‌کند که ممکن است در مقایسه با ترانسفورمرها عملکرد کمتری در پردازش داده‌های طولانی‌تر داشته باشند.

روش آموزش

بیشتر متکی بر یادگیری نظارت‌شده است و نیاز به داده‌های برچسب‌دار بیشتری دارد.

دقت

عملکرد DeepSpeech به دلیل معماری و روش‌های آموزشی کمتر از Wav2vec 2.0 و Whisper است.

نتیجه‌گیری:

Whisper با استفاده از یادگیری خودنظارتی و نیمه‌نظارتی، توانایی بهره‌برداری از داده‌های گسترده و متنوع صوتی را دارد و می‌تواند بازنمایی‌های صوتی قدرتمندی ایجاد کند که به دقت بالایی در تشخیص گفتار منجر می‌شود.

Wav2vec 2.0 نیز یک مدل پیشرفته است که از یادگیری خودنظارتی بهره می‌برد، اما Whisper ممکن است در برخی موارد به دلیل استفاده از داده‌های بیشتر و روش‌های پیشرفته‌تر نیمه‌نظارتی عملکرد بهتری داشته باشد.

DeepSpeech با معماری قدیمی‌تر RNN و تکیه بیشتر بر داده‌های برچسب‌دار، نسبت به دو مدل دیگر عملکرد کمتری دارد.

تبدیل متن به گفتار

Tacotron 2

معماری مدل

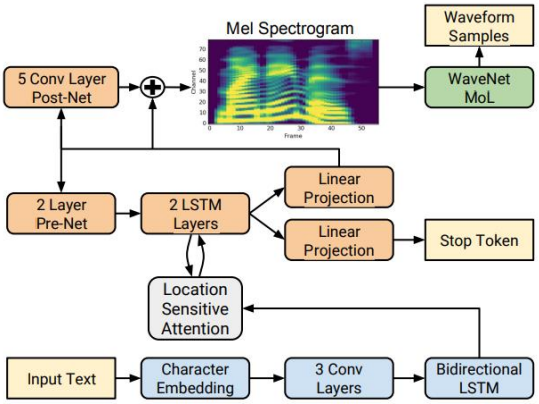
Tacotron 2 از یک سیستم دو مرحله‌ای استفاده می‌کند. اولین مرحله شامل یک مدل پردازش متن به طیف (spectrogram) است و دومین مرحله یک شبکه WaveNet vocoder است که طیف‌ها را به سیگنال‌های صوتی تبدیل می‌کند. Tacotron 2 قادر است با استفاده از داده‌های صوتی و متن‌های همگام‌سازی شده، مدل‌های صوتی با کیفیت بالا تولید کند. این سیستم قابلیت تولید صدای با کیفیت انسانی، شامل تنوعات صوتی را داراست.

مرحله اول

مدل Tacotron 2 شامل یک Encoder-Decoder مبتنی بر ترانسفورمر (Transformer) است. این مدل ابتدا ورودی متنی را به یک بازنمایی طیفی (mel-spectrogram) تبدیل می‌کند.

مرحله دوم WaveNet vocoder

که یک شبکه عصبی کانولوشنی است، این بازنمایی طیفی را به سیگنال صوتی تبدیل می‌کند.



شکل ۴ معماری Tacotron 2

الگوریتم‌های یادگیری ماشین

یادگیری نظارت‌شده (Supervised Learning)

از یادگیری نظارت‌شده استفاده می‌کند، جایی که مدل با استفاده از داده‌های جفت‌شده متن-صوت آموزش می‌بینند.

مدل‌های:Sequence-to-Sequence

از مدل‌های sequence-to-sequence استفاده می‌کند که شامل Encoder-Decoder است. این مدل‌ها برای تبدیل ورودی متنی به بازنمایی طیفی به کار می‌روند.

شبکه عصبی کانولوشنی (Convolutional Neural Networks - CNNs): WaveNet vocoder از شبکه‌های عصبی کانولوشنی برای تبدیل بازنمایی طیفی به سیگنال صوتی استفاده می‌کند.

مقایسه با پروژه‌های مشابه

DeepVoice

معماری

از یک سیستم pipeline چند مرحله‌ای مشابه Tacotron 2 استفاده می‌کند، اما معماری داخلی آن متفاوت است.

روش آموزش

نیز از یادگیری نظارت‌شده با داده‌های متن-صوت جفت‌شده استفاده می‌کند.

عملکرد

هر دو مدل به دقت بالایی در تولید گفتار طبیعی دست یافته‌اند، اما Tacotron 2 با استفاده از WaveNet vocoder صدای با کیفیت‌تری تولید می‌کند.

WaveNet (Google DeepMind)

معماری

به طور مستقیم برای تولید گفتار از سیگنال‌های صوتی استفاده می‌شود، در حالی که Tacotron 2 از WaveNet به عنوان مرحله دوم برای تبدیل بازنمایی‌های طیفی به سیگنال‌های صوتی استفاده می‌کند.

روش آموزش

از شبکه عصبی کانولوشنی برای تولید گفتار استفاده می‌کند.

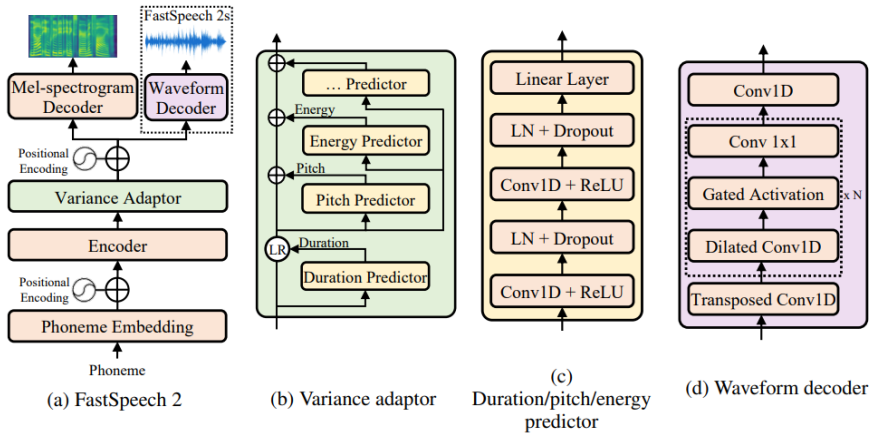
عملکرد

به خودی خود صدای با کیفیت بالایی تولید می‌کند، اما Tacotron 2 با استفاده از آن در مرحله دوم، کیفیت صدای حتی بالاتری را فراهم می‌کند.

FastSpeech 2

معماری مدل

FastSpeech 2 از یک معماری مبتنی بر ترانسفورمر (Transformer) استفاده می‌کند که برای تولید سریع و باکیفیت گفتار طراحی شده است. این مدل شامل دو بخش اصلی است: یک مدل پردازش متن به بازنمایی طیفی (mel-spectrogram) و یک vocoder که بازنمایی طیفی را به سیگنال صوتی تبدیل می‌کند. برخلاف Tacotron 2 ، FastSpeech 2 از روش پیش‌بینی طول(length prediction) برای همگام‌سازی ورودی و خروجی استفاده می‌کند، که موجب بهبود پایداری و کیفیت گفتار تولیدی می‌شود.



شکل ۵ معماری FastSpeech 2

الگوریتم‌های یادگیری ماشین

یادگیری نظارت‌شده

از یادگیری نظارت‌شده استفاده می‌کند، جایی که مدل با استفاده از داده‌های جفت‌شده متن-صوت آموزش می‌بیند.

مقایسه با پروژه‌های مشابه

Tacotron ۲

معماری

از یک مدل Encoder-Decoder مبتنی بر ترانسفورمر برای تولید mel-spectrogram استفاده می‌کند، اما فاقد پیش‌بینی طول است که در FastSpeech 2 وجود دارد.

روش آموزش

هر دو مدل از یادگیری نظارت‌شده استفاده می‌کنند، اما FastSpeech 2 با پیش‌بینی طول و تولید سریع‌تر، بهبود یافته است.

عملکرد

هر دو مدل برای تبدیل متن به گفتار استفاده می‌شوند، اما FastSpeech 2 به دلیل سرعت بالاتر و پایداری بیشتر، مناسب‌تر است.

WaveGlow

معماری

از یک معماری مبتنی بر شبکه‌های عصبی برای تولید مستقیم سیگنال‌های صوتی استفاده می‌کند.

روش آموزش

نیاز به داده‌های جفت‌شده متن-صوت دارد و از یادگیری نظارت‌شده استفاده می‌کند.

عملکرد

کیفیت صدای تولید شده توسط WaveGlow بالاست، اما سرعت تولید آن ممکن است کمتر از FastSpeech 2 باشد.

DeepVoice

معماری

از یک سیستم چند مرحله‌ای شامل مدل‌های مختلف برای تولید گفتار استفاده می‌کند.

روش آموزش

بیشتر بر یادگیری نظارت‌شده متکی است.

عملکرد

کیفیت صدای تولید شده توسط DeepVoice بالاست، اما پیچیدگی و زمان آموزش آن ممکن است بیشتر باشد.

نتیجه‌گیری

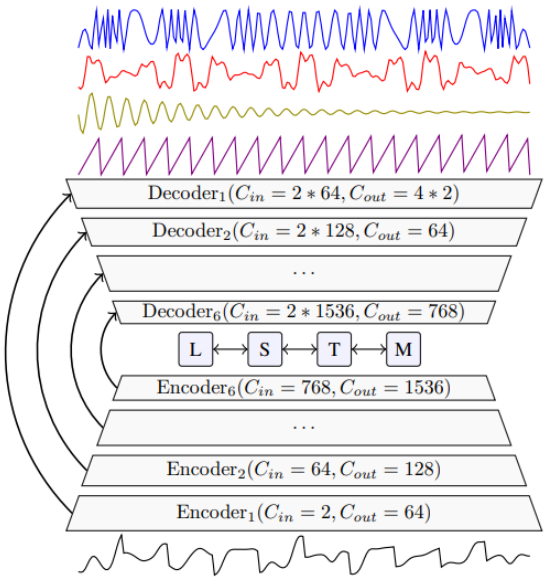
FastSpeech 2 با استفاده از معماری ترانسفورمر و الگوریتم پیش‌بینی طول، قادر به تولید گفتار با سرعت بالا و کیفیت بسیار خوب است. این مدل از یادگیری نظارت‌شده بهره می‌برد و با کاهش زمان تولید و افزایش پایداری، بهبودهایی نسبت به مدل‌های قبلی مانند Tacotron 2و WaveGlow دارد. Tacotron 2 یک مدل قدرتمند برای تبدیل متن به گفتار است، اما FastSpeech 2 با پیش‌بینی طول و تولید سریع‌تر، عملکرد بهتری دارد. WaveGlow و DeepVoice نیز مدل‌های قدرتمندی برای تولید گفتار هستند، اما FastSpeech 2 با توجه به سرعت و کارایی بالا، بهبودهایی در این زمینه ارائه می‌دهد.

بهسازی گفتار

Demucs

معماری مدل

Demucs از یک شبکه عصبی مبتنی بر کانولوشن و LSTM بهره می‌برد که به طور خاص برای جداسازی منابع صوتی موسیقی طراحی شده است. مدل شامل چندین لایه کانولوشنی (Convolutional Layers) و LSTM برای یادگیری ویژگی‌های زمانی و فرکانسی پیچیده موسیقی است. مدل Demucs از دو مسیر موازی استفاده می‌کند: یکی برای پردازش سیگنال در دامنه زمان (time domain) و دیگری در دامنه فرکانس(frequency domain) . این دو مسیر به مدل کمک می‌کند تا به طور موثرتری منابع موسیقی را جدا کند.



شکل ۶ معماری Demucs

الگوریتم‌های یادگیری ماشین

یادگیری نظارت‌شده

از یادگیری نظارت‌شده استفاده می‌کند، جایی که مدل با استفاده از داده‌های برچسب‌دار آموزش می‌بیند. این داده‌ها شامل آهنگ‌های موسیقی و منابع جداگانه آنها (مانند وکال و درام) هستند.

مقایسه با پروژه‌های مشابه

Open-Unmix

معماری

از یک مدل مبتنی بر LSTM و کانولوشن برای جداسازی منابع موسیقی استفاده می‌کند.

روش آموزش

مانندDemucs ، Open-Unmix از یادگیری نظارت‌شده با داده‌های موسیقی و منابع جداگانه استفاده می‌کند.

عملکرد

Open-Unmix کیفیت خوبی دارد، اما Demucs به دلیل استفاده از دو مسیر موازی برای پردازش سیگنال در دامنه زمان و فرکانس، عملکرد بهتری در جداسازی منابع دارد.

Spleeter

معماری

از مدل‌های مبتنی بر کانولوشن برای جداسازی منابع موسیقی استفاده می‌کند.

روش آموزش

از یادگیری نظارت‌شده با استفاده از داده‌های برچسب‌دار بهره می‌برد.

عملکرد

به دلیل سادگی و کارایی بالا محبوب است، اما Demucs با استفاده از معماری پیچیده‌تر و یادگیری عمیق، دقت و کیفیت بهتری ارائه می‌دهد.

Wave-U-Net

معماری

از یک مدل U-Net در دامنه زمان برای جداسازی منابع موسیقی استفاده می‌کند.

روش آموزش

یادگیری نظارت‌شده با داده‌های برچسب‌دار.

عملکرد

این مدل نیز کیفیت خوبی دارد، اما Demucs با استفاده از ترکیب CNN و LSTM و پردازش در دو دامنه زمانی و فرکانسی، عملکرد بهتری دارد.

نتیجه‌گیری:

Demucsبا استفاده از یک معماری ترکیبی شامل CNN و LSTM و پردازش سیگنال در دو دامنه زمانی و فرکانسی، به عنوان یکی از پیشرفته‌ترین مدل‌های جداسازی منابع موسیقی شناخته می‌شود. این مدل با استفاده از یادگیری نظارت‌شده و داده‌های برچسب‌دار، قادر به استخراج دقیق و با کیفیت منابع مختلف موسیقی است.

Open-Unmixو Spleeter نیز مدل‌های قوی و محبوبی هستند، اما Demucs به دلیل استفاده از معماری پیچیده‌تر و بهینه‌سازی بهتر، عملکرد بهتری در بسیاری از موارد دارد.

Wave-U-Net با استفاده از U-Net در دامنه زمان نیز عملکرد خوبی دارد، اما Demucs با استفاده از روش‌های پیشرفته‌تر و ترکیب لایه‌های CNN وLSTM ، کیفیت بالاتری در جداسازی منابع موسیقی ارائه می‌دهد.

Demucs به عنوان یک ابزار قدرتمند برای جداسازی منابع موسیقی، کاربردهای گسترده‌ای در تولید موسیقی، میکس و مسترینگ، تحلیل موسیقی و حتی در کاربردهای آموزشی دارد. با توجه به کیفیت بالای جداسازی و کارایی اجرای مدل، Demucs یکی از مدل‌های پیشرو در این حوزه است.

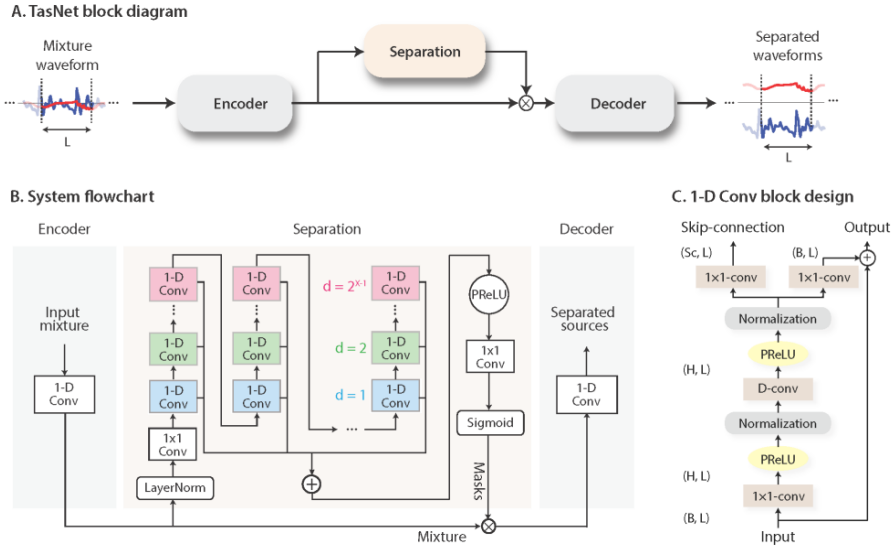
Conv-TasNet

معماری مدل

Conv-TasNet از یک معماری مبتنی بر کانولوشن (Convolutional Neural Network) برای پردازش مستقیم سیگنال‌های صوتی در دامنه زمان استفاده می‌کند. مدل شامل سه بخش اصلی است:

- Encoder: سیگنال ورودی را به یک بازنمایی زمانی-فرکانسی با دقت بالا تبدیل می‌کند.
- Separator: با استفاده از چندین لایه کانولوشنی با دروازه‌گیری و نرمال‌سازی، اجزای مختلف سیگنال را جدا می‌کند.
- Decoder: بازنمایی‌های جدا شده را به سیگنال‌های صوتی در دامنه زمان تبدیل می‌کند.

این مدل با استفاده از لایه‌های کانولوشنی به جای شبکه‌های recurrent مانند LSTM یا GRU ، بهبودهای قابل توجهی در سرعت و کارایی دارد.



شکل ۷ معماری Conv-TasNet

الگوریتم‌های یادگیری ماشین:

یادگیری نظارت‌شده

از یادگیری نظارت‌شده استفاده می‌کند، جایی که مدل با استفاده از داده‌های برچسب‌دار شامل سیگنال‌های صوتی مخلوط و اجزای جدا شده آموزش می‌بیند.

مقایسه با پروژه‌های مشابه

Demucs

معماری

از یک مدل ترکیبی شامل کانولوشن و LSTM برای جداسازی منابع موسیقی استفاده می‌کند، در حالی که Conv-TasNet از کانولوشن در دامنه زمان برای جداسازی گفتار استفاده می‌کند.

روش آموزش

هر دو مدل از یادگیری نظارت‌شده استفاده می‌کنند، اما داده‌های آموزشی و وظایف آنها متفاوت است.

Wave-U-Net

معماری

از یک مدل U-Net در دامنه زمان برای جداسازی منابع صوتی استفاده می‌کند، که مشابه به Conv-TasNet است اما با تفاوت‌هایی در جزئیات معماری.

روش آموزش

نیز از یادگیری نظارت‌شده با داده‌های برچسب‌دار استفاده می‌کند.

عملکرد

هر دو مدل کیفیت بالایی دارند، اما Conv-TasNet به دلیل استفاده از لایه‌های کانولوشنی با دروازه‌گیری و نرمال‌سازی ممکن است در برخی موارد عملکرد بهتری داشته باشد.

TasNet

معماری

مدل پایه Conv-TasNet است که از کانولوشن برای جداسازی منابع در دامنه زمان استفاده می‌کند.

روش آموزش

مشابه Conv-TasNet ، از یادگیری نظارت‌شده استفاده می‌کند.

عملکرد

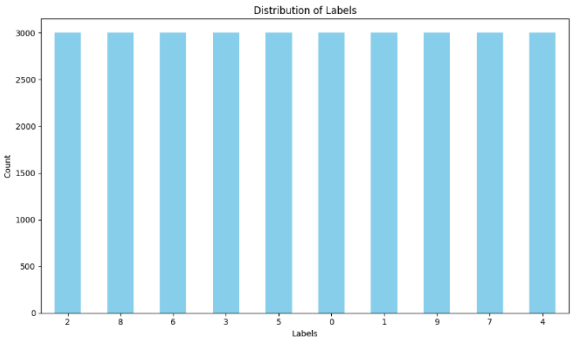
نسخه پیشرفته‌تر TasNet است و با بهبودهایی در معماری و الگوریتم‌ها، دقت و کارایی بالاتری دارد.

نتیجه‌گیری

Conv-TasNet با استفاده از یک معماری مبتنی بر کانولوشن و پردازش مستقیم در دامنه زمان، به عنوان یکی از پیشرفته‌ترین مدل‌های جداسازی منابع گفتار شناخته می‌شود. این مدل از یادگیری نظارت‌شده و داده‌های برچسب‌دار استفاده می‌کند و قادر به جداسازی دقیق و سریع منابع صوتی مختلط است. Demucs و Wave-U-Net نیز مدل‌های قوی در حوزه جداسازی منابع صوتی هستند، اما Conv-TasNet به دلیل استفاده از معماری ساده‌تر و بهینه‌تر برای جداسازی گفتار، عملکرد بهتری دارد. TasNet نسخه پایهConv-TasNet است وConv-TasNetبا بهبودهای بیشتر در معماری و الگوریتم‌ها، کیفیت و کارایی بالاتری ارائه می‌دهد.

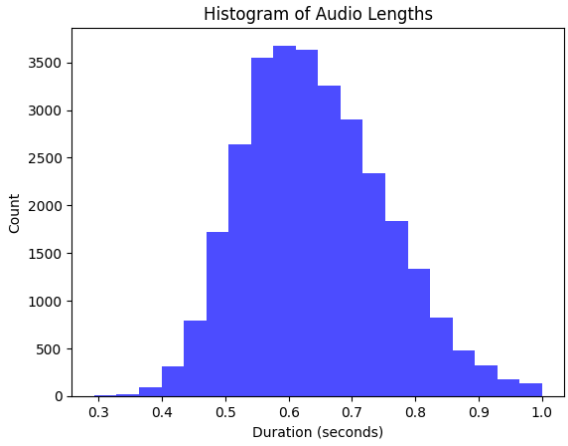
سوال ۲: تشخیص اعداد با شبکه عصبی پرسپترون چندلایه

ابتدا مجموعه داده را از سایت کگل دانلود می‌کنیم و به بررسی‌های اولیه می‌پردازیم. توزیع داده‌ها در همه‌ی کلاس‌ها یکسان است، بنابراین نیاز به پیش‌پردازش اولیه‌ای در این بخش نداریم.



شکل ۸ توزیع داده‌های مجموعه‌داده‌ی AudioMNIST

با این وجود طول همه‌ی فایل‌های صوتی برابر نبوده و در صورت استفاده از ویژگی‌هایی مانند MFCC و مشتقات آن نیاز است که طول همه‌ی این داده‌ها برابر شود. بنابراین یکی از گام‌های پیش پردازش ما استفاده از Zero Padding خواهد بود.



شکل ۹ توزیع طول داده‌های مجموعه‌داده‌ی AudioMNIST

بنابراین ابتدا تابع Preprocess ابتدا فایل‌های صوتی را بارگذاری می‌کند و سپس اقدام به پیش‌پردازش آن‌ها می‌کند. پیش‌پردازش شامل حذف سکوت‌های ابتدایی و انتهایی فایل صوتی و پر کردن صداهای باقی‌مانده تا به طول ۸۰۰۰ برسند می‌شود. در نهایت، داده‌های پیش‌پردازش شده و برچسب‌های اولیه به عنوان خروجی تابع بازگردانده می‌شوند.

- ## گام ۱: فراخوانی داده‌ها و بخش‌بندی

داده‌ها را به نسبت ۸۰ و ۲۰ به دو دسته‌ی آموزش و آزمون تقسیم می‌کنیم. برای اینکار از کد زیر استفاده می‌کنیم:

```
X_train, X_test, Y_train, Y_test = train_test_split(features, Y,
test_size=0.2, random_state=42)
```

البته بدون استفاده از کتابخانه هم این کار ممکن است که تابع آن نیز در کد تعریف شده است.

- ## گام ۲: پیش‌پردازش‌های لازم روی داده‌ها

برای این کار از تابع feature_extractor استفاده می‌کنیم. این تابع ورودی‌های X و Y را به عنوان لیستی از داده‌های صوتی و برچسب‌های مربوط به آن‌ها می‌گیرد. سپس برای هر داده صوتی، با توجه به خواسته‌ی سوال از روش MFCC با طول فریم ۲۵ میلی ثانیه، ۲۴ فلیتر مل و ۱۲ ویژگی با مشتق های مرتبه یک و دو استفاده می‌شود. ویژگی‌های MFCC (Mel-frequency cepstral coefficients) و مشتقات آن‌ها (delta و delta-delta) را استخراج می‌کند. همچنین از میانگین‌گیری برای هم اندازه و کمتر کردن ویژگی‌ها استفاده می‌شود. سپس این ویژگی‌ها را به لیست features اضافه می‌کند و برچسب مربوط به هر داده صوتی را به لیست Y_lable اضافه می‌کند.

حال ابتدا ویژگی‌ها رو نورمالیز می‌کنیم تا مدل بهتر آموزش ببیند، علاوه بر این از آنجا که کلاس‌ها به صورت اعداد ۰ تا ۹ تعریف شده‌اند از OneHotEncoder نیز برای تبدیل آن‌ها به یک بردار OneHot استفاده می‌کنیم.

- ## گام ۳: ساخت مدل MLP

حال یک کلاس MLP تعریف می‌کنیم که شرح آن به صورت زیر است:

__init__: در این متد، مشخصات شبکه مانند تعداد لایه‌ها، اندازه وزن‌ها و بایاس‌ها و توابع فعال‌سازی مقداردهی اولیه می‌شوند.

Forward: این متد برای انجام عملیات فوروارد پس از هر لایه در شبکه استفاده می‌شود. ابتدا خروجی‌های لایه‌ها با استفاده از وزن‌ها و بایاس‌ها محاسبه می‌شوند و سپس تابع فعال‌سازی مربوطه بر روی خروجی لایه اعمال می‌شود.

compute_loss: این متد برای محاسبه تابع خطا (Loss function) برای خروجی تولید شده توسط شبکه استفاده می‌شود. در اینجا از میانگین مربعات خطا (Mean Squared Error) استفاده شده است.

Backward: این متد برای انجام عملیات بک‌پروپگیشن در شبکه به منظور به‌روزرسانی وزن‌ها و بایاس‌ها استفاده می‌شود.

update_parameters: این متد برای به‌روزرسانی وزن‌ها و بایاس‌ها با استفاده از نرخ یادگیری (learning rate) و گرادیان‌ها استفاده می‌شود.

Train: این متد برای آموزش شبکه استفاده می‌شود. این شامل تقسیم داده‌ها به دسته‌های آموزش و اعتبارسنجی، ایجاد دسته‌های کوچک (Mini-batches)، انجام فوروارد و بک‌پروپگیشن برای هر دسته، به‌روزرسانی وزن‌ها و بایاس‌ها و محاسبه دقت و خطاها است.

Predict: این متد برای پیش‌بینی خروجی برای داده‌های ورودی استفاده می‌شود.

Evaluate: این متد برای ارزیابی عملکرد شبکه با استفاده از داده‌های ورودی و برچسب‌های مربوطه استفاده می‌شود.

در این کد همچنین شامل متدهای دیگری مانند relu، sigmoid، Bipolar_sigmoid و softmax برای انجام عملیات فعال‌سازی و مشتقات آن‌ها می‌باشد.

- **گام ۴- آموزش و آزمون مدل**

الف- **MLP با یک لایه‌ی مخفی**

Single-Layer MLPs with Sigmoid Activation Functions

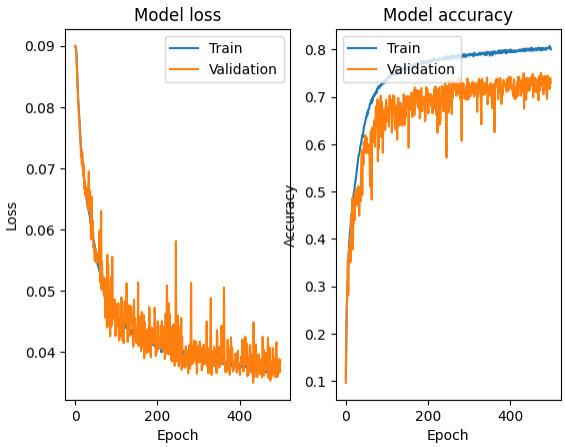
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	Bipolar Sigmoid	MSE	10	36	$\frac{(36+10)}{2}=23$	0.001	SGD

جدول 1 مشخصات MLP تک لایه با تابع فعال‌سازی Sigmoid

برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش



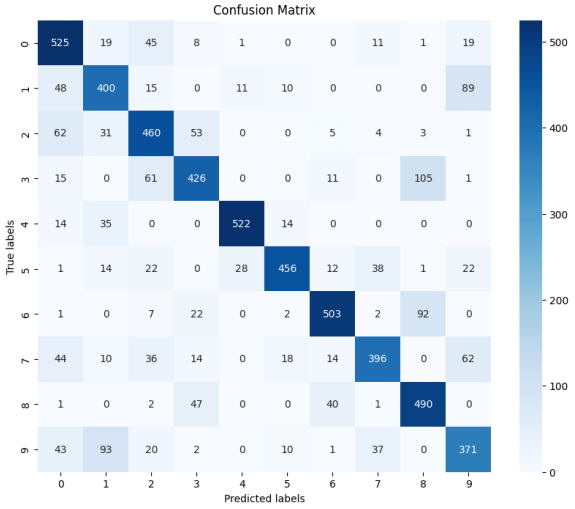
شکل ۱۰ MLP تک لایه با تابع فعال‌سازی Sigmoid در طول آموزش

با توجه به نمودار بالا متوجه می‌شویم که مدل در حال یادگیری الگوهای مورد نظر بوده و عملکرد مناسبی داشته است.

ماتریس درهمریختگی و Classification report

Classification Report:

	precision	recall	f1-score	support
0	0.70	0.83	0.76	629
1	0.66	0.70	0.68	573
2	0.69	0.74	0.71	619
3	0.74	0.69	0.72	619
4	0.93	0.89	0.91	585
5	0.89	0.77	0.83	594
6	0.86	0.80	0.83	629
7	0.81	0.67	0.73	594
8	0.71	0.84	0.77	581
9	0.66	0.64	0.65	577
accuracy			0.76	6000
macro avg	0.76	0.76	0.76	6000
weighted avg	0.77	0.76	0.76	6000



شکل 11 عملکرد MLP تک لایه با تابع فعال‌سازی Sigmoid

مدل دقت ۷۶٪ روی داده‌های آزمون نشان می‌دهد که برای یک مدل تک لایه عملکرد بسیار مناسبی است.

استفاده نکردن از میانگین‌گیری در ویژگی‌ها

در بخش الف، ما ابتدا داده‌ها را هم‌طول می‌کنیم و از میانگین ویژگی‌های MFCC و در کنار میانگین ویژگی مشتق‌های MFCC را به عنوان ویژگی برای آموزش مدل استفاده می‌کنیم که در نهایت یک بردار ویژگی با طول ۳۶ به ما می‌دهد.

در صورتی که این کار را انجام ندهیم و از میانگین‌گیری استفاده نکنیم، ویژگی برای هر صوت دو بعدی خواهد بود و به اندازه‌ی (۳۶, ۸۱) خواهد بود. و برای اینکه بتوانیم به شبکه این ماتریس را ورودی بدهیم مجبوریم از flatten کردن ویژگی‌ها استفاده کنیم که یک بردار ۲۹۱۶ تایی برای هر صوت خواهیم داشت. شبکه با استفاده از نوع ویژگی نیز آموزش خواهد دید، اما چون تعداد نورون‌های ورودی به شدت زیاد می‌شود (و البته در این سوال نورون‌های لایه مخفی نیز به تعداد نورون‌های ورودی وابسته است.) بنابراین سرعت آموزش شبکه به شدت پایین‌تر خواهد آمد و همچنین شبکه به داده‌های آموزش بیشتر overfit خواهد شد. به عبارت دیگر با افزایش ابعاد ورودی و داشتن ویژگی‌های بیشتر، مدل عملکرد بهتری روی داده‌های ما خواهد داشت ولی لزوما به داده‌هایی که توزیع دیگری دارند خوب fit نخواهد شد.

علاوه بر این اگر داده‌ها را در بخش الف هم طول نکنیم، در این روش نیز مجبوریم به بردار ویژگی‌های padding صفر اضافه کنیم تا بردارها هم طول شوند و باقی توضیحات مثل قبل می‌باشد. استفاده از این نوع ویژگی در کد با عنوان Second features موجود می‌باشد، اما به واسطه طول کشیدن آموزش آن، همه‌ی شبکه‌ها با این نوع ویژگی‌ها آموزش ندیده است. همچنین بخشی از نتایج در انتهای این سوال ضمیمه شده است.

ب- MLP با دو لایه‌ی مخفی

- **Two-Layer MLPs with Sigmoid Activation Functions (H1=23, H2=23)**

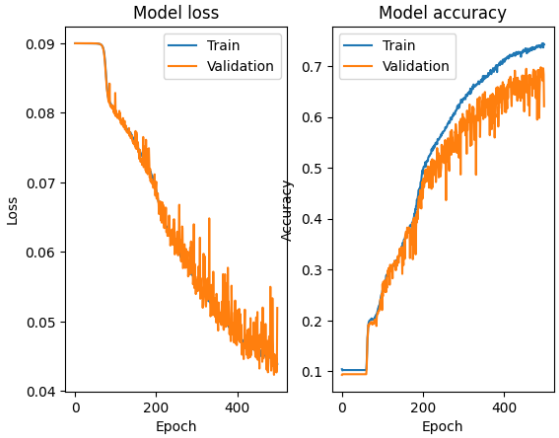
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی اول	تابع فعال‌سازی لایه مخفی دوم	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	Bipolar Sigmoid	Bipolar Sigmoid	MSE	10	36	$\frac{(36+10)}{2}=23$	$\frac{(36+10)}{2}=23$	0.001	SGD

جدول 2 MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی Sigmoid

برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

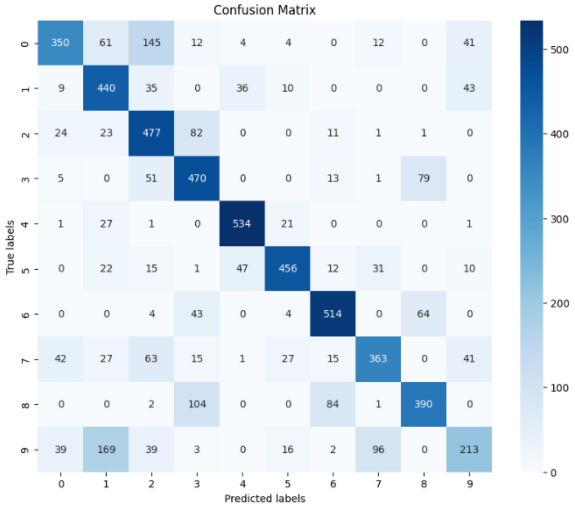
نمودار خطا و دقت در طول آموزش



شکل ۱۲ MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی Sigmoid در طول آموزش

این مدل نیز کمی دیرتر همگرا شده است که احتمالا به خاطر vanishing gradient باشد.

ماتریس درهمریختگی و Classification report




```
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.56      0.64       629
           1       0.57      0.77      0.66       573
           2       0.57      0.77      0.66       619
           3       0.64      0.76      0.70       619
           4       0.86      0.91      0.88       585
           5       0.85      0.77      0.81       594
           6       0.79      0.82      0.80       629
           7       0.72      0.61      0.66       594
           8       0.73      0.67      0.70       581
           9       0.61      0.37      0.46       577

    accuracy                           0.70      6000
   macro avg       0.71      0.70      0.70      6000
weighted avg       0.71      0.70      0.70      6000
```

شکل ۱۳عملکرد MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی Sigmoid

- **Two-Layer MLPs with Sigmoid Activation Functions (H1=24, H2=18)**

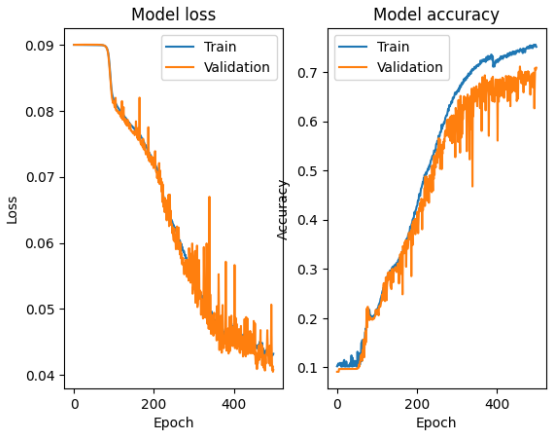
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی اول	تابع فعال‌سازی لایه مخفی دوم	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	Bipolar Sigmoid	Bipolar Sigmoid	MSE	10	36	$\frac{2*(36)}{3}=24$	$\frac{(36)}{2}=18$	0.001	SGD

جدول ۳ MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی Sigmoid

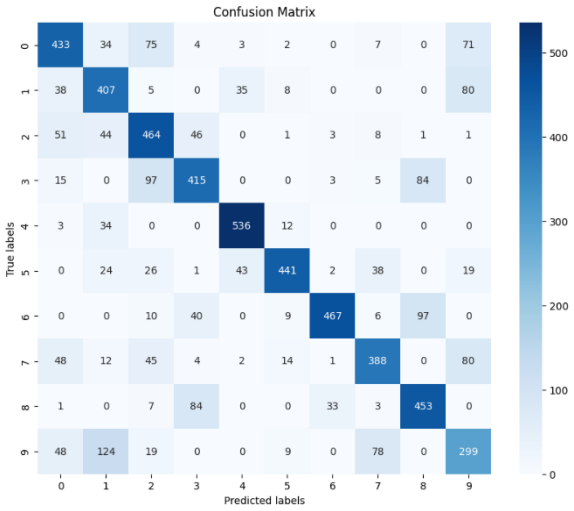
برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش



شکل ۱۴ MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی Sigmoid در طول آموزش

ماتریس درهمریختگی و Classification report




```
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.69      0.68       629
           1       0.60      0.71      0.65       573
           2       0.62      0.75      0.68       619
           3       0.70      0.67      0.68       619
           4       0.87      0.92      0.89       585
           5       0.89      0.74      0.81       594
           6       0.92      0.74      0.82       629
           7       0.73      0.65      0.69       594
           8       0.71      0.78      0.75       581
           9       0.54      0.52      0.53       577

    accuracy                           0.72      6000
   macro avg       0.73      0.72      0.72      6000
weighted avg       0.73      0.72      0.72      6000
```

شکل ۱۵عملکرد MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی Sigmoid

- **Two-Layer MLPs with Sigmoid Activation Functions (H1=18, H2=12)**

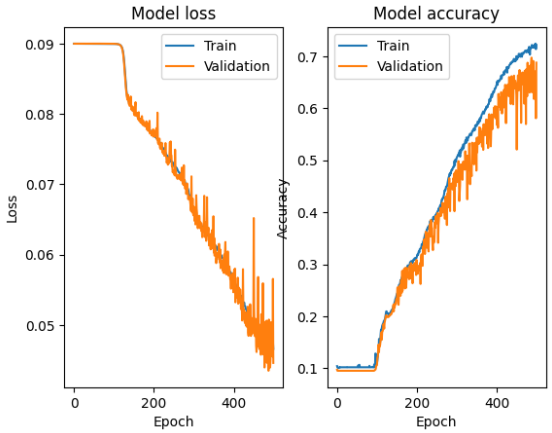
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی اول	تابع فعال‌سازی لایه مخفی دوم	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	Bipolar Sigmoid	Bipolar Sigmoid	MSE	10	36	$\frac{(36)}{2} = 18$	$\frac{(36)}{3} = 12$	0.001	SGD

جدول ۴ MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی Sigmoid

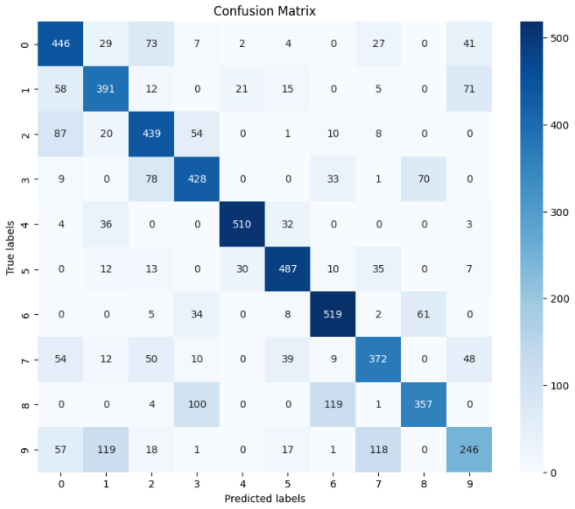
برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش



شکل ۱۶MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی Sigmoid در طول آموزش

ماتریس درهمریختگی و Classification report




```
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.71      0.66       629
           1       0.63      0.68      0.66       573
           2       0.63      0.71      0.67       619
           3       0.68      0.69      0.68       619
           4       0.91      0.87      0.89       585
           5       0.81      0.82      0.81       594
           6       0.74      0.83      0.78       629
           7       0.65      0.63      0.64       594
           8       0.73      0.61      0.67       581
           9       0.59      0.43      0.50       577

    accuracy                           0.70      6000
   macro avg       0.70      0.70      0.70      6000
weighted avg       0.70      0.70      0.70      6000
```

شکل ۱۷ عملکرد MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی Sigmoid

پ- MLP با یک لایه و دو لایه‌ی مخفی و تابع ReLU

Single-Layer MLPs with ReLU Activation Functions

با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

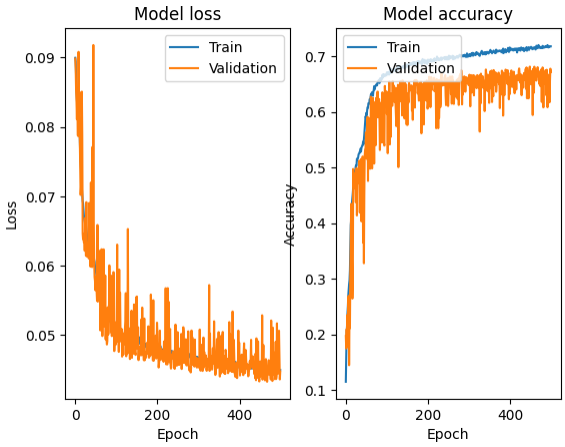
تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	نرخ یادگیری	رویکرد آموزش

۵۰۰ دوره	SoftMax	ReLU	MSE	10	36	$\frac{(36+10)}{2}=23$	0.001	SGD

جدول5 مشخصات MLP تک لایه با تابع فعال‌سازی ReLU

برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش

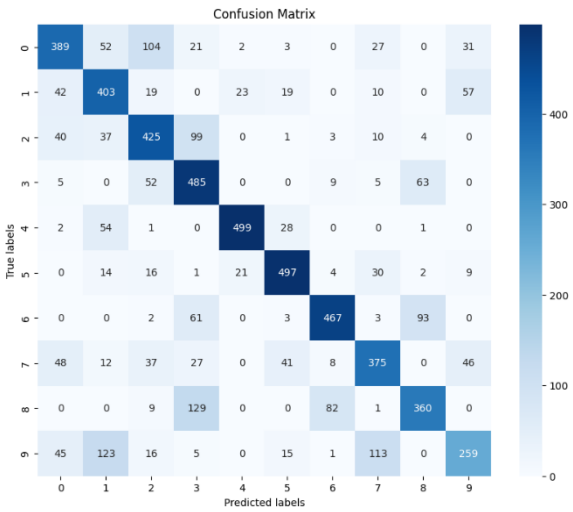


شکل ۱۸ MLP تک لایه با تابع فعال‌سازی ReLU در طول آموزش

با توجه به نمودار بالا متوجه می‌شویم که مدل نتوانسته الگوی مورد نظر را یاد بگیرد و احتمالا به خاطر این است که یک لایه ReIU غیر خطی بودن لازم را فراهم نمی‌کند.

ماتریس درهمریختگی و Classification report

Classification Report:				
	precision	recall	f1-score	support
0	0.68	0.62	0.65	629
1	0.58	0.70	0.64	573
2	0.62	0.69	0.65	619
3	0.59	0.78	0.67	619
4	0.92	0.85	0.88	585
5	0.82	0.84	0.83	594
6	0.81	0.74	0.78	629
7	0.65	0.63	0.64	594
8	0.69	0.62	0.65	581
9	0.64	0.45	0.53	577
accuracy			0.69	6000
macro avg	0.70	0.69	0.69	6000
weighted avg	0.70	0.69	0.69	6000



شکل 19 عملکرد MLP تک لایه با تابع فعال‌سازی ReLU

- **Two-Layer MLPs with ReLU Activation Functions (H1=23, H2=23)**

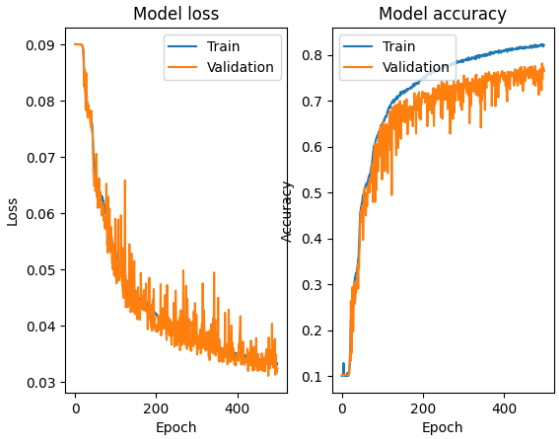
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی اول	تابع فعال‌سازی لایه مخفی دوم	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	ReLU	ReLU	MSE	10	36	$\frac{(36+10)}{2} = 23$	$\frac{(36+10)}{2} = 23$	0.001	SGD

جدول 6 MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی ReLU

برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

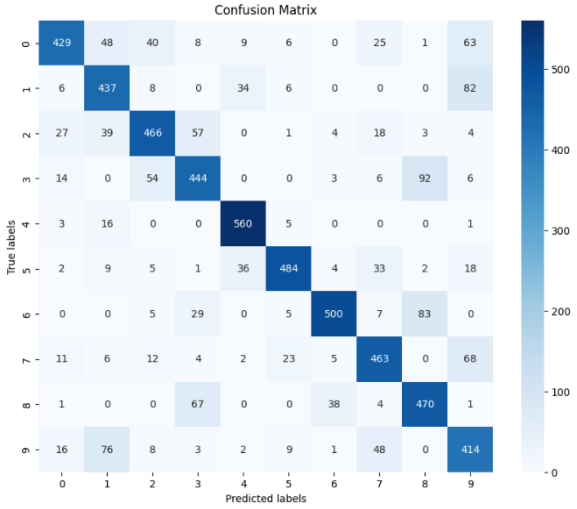
نمودار خطا و دقت در طول آموزش



شکل ۲۰ MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی ReLU در طول آموزش

این مدل با زیاد شدن لایه، عملکرد بهتری داشته و در حال یادگیری الگوی داده‌ها است.

ماتریس درهمریختگی و Classification report




```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.68      0.75       629
           1       0.69      0.76      0.73       573
           2       0.78      0.75      0.77       619
           3       0.72      0.72      0.72       619
           4       0.87      0.96      0.91       585
           5       0.90      0.81      0.85       594
           6       0.90      0.79      0.84       629
           7       0.77      0.78      0.77       594
           8       0.72      0.81      0.76       581
           9       0.63      0.72      0.67       577

    accuracy                           0.78      6000
   macro avg       0.78      0.78      0.78      6000
weighted avg       0.78      0.78      0.78      6000
```

شکل ۲۱عملکرد MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی ReLU

- **Two-Layer MLPs with ReLU Activation Functions (H1=24, H2=18)**

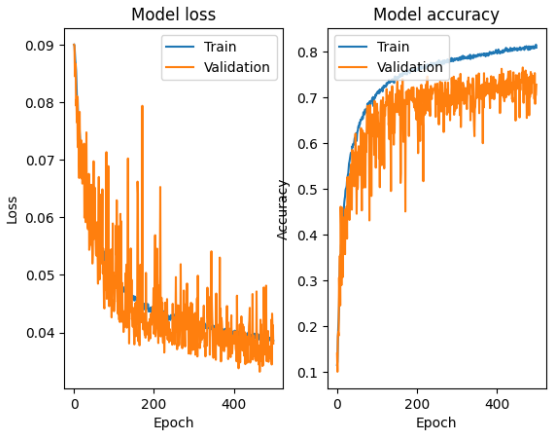
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی اول	تابع فعال‌سازی لایه مخفی دوم	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	ReLU	ReLU	MSE	10	36	$\frac{2*(36)}{3}=24$	$\frac{(36)}{2}=18$	0.001	SGD

جدول ۷ MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی ReLU

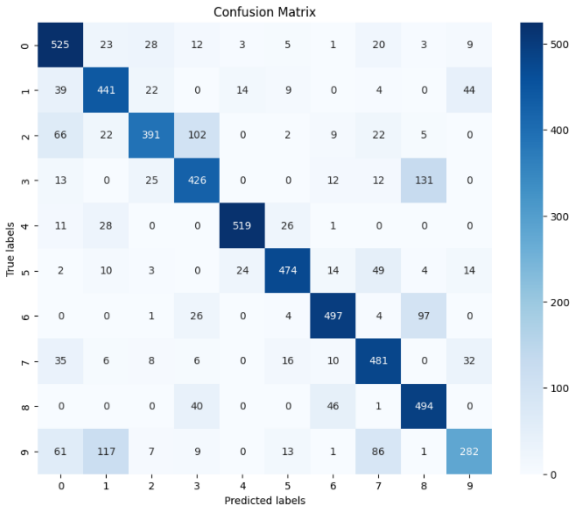
برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش



شکل ۲۲MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی ReLU در طول آموزش

ماتریس درهمریختگی و Classification report




```
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.83      0.76       629
           1       0.68      0.77      0.72       573
           2       0.81      0.63      0.71       619
           3       0.69      0.69      0.69       619
           4       0.93      0.89      0.91       585
           5       0.86      0.80      0.83       594
           6       0.84      0.79      0.81       629
           7       0.71      0.81      0.76       594
           8       0.67      0.85      0.75       581
           9       0.74      0.49      0.59       577

    accuracy                           0.76      6000
   macro avg       0.76      0.75      0.75      6000
weighted avg       0.76      0.76      0.75      6000
```

شکل ۲۳عملکرد MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی ReLU

- **Two-Layer MLPs with ReLU Activation Functions (H1=18, H2=12)**

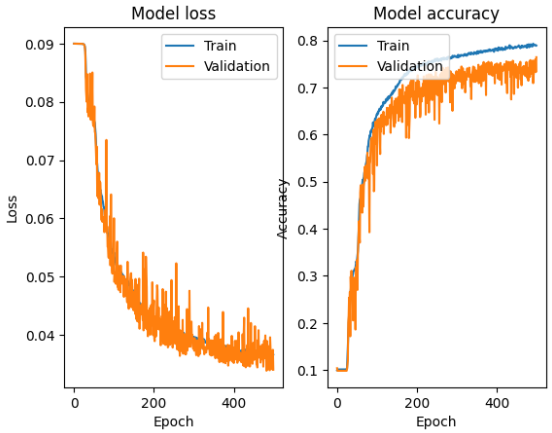
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی اول	تابع فعال‌سازی لایه مخفی دوم	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	ReLU	ReLU	MSE	10	36	$\frac{(36)}{2} = 18$	$\frac{(36)}{3} = 12$	0.001	SGD

جدول ۸ MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی ReLU

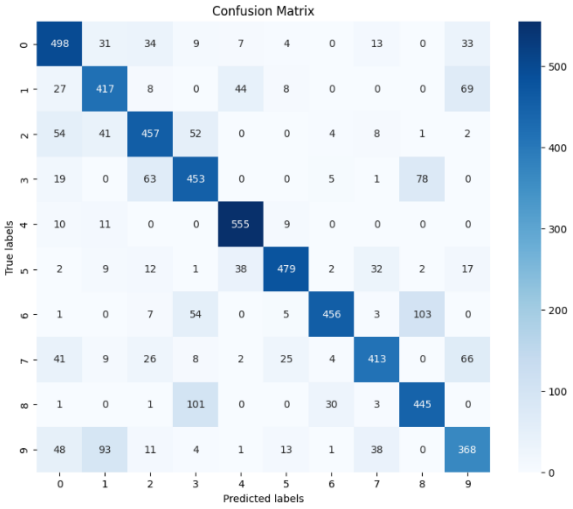
برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش



شکل MLP۲۴ دو لایه (۱۸و۱۲) با تابع فعال‌سازی ReLU در طول آموزش

ماتریس درهمریختگی و Classification report




```
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.79      0.75       629
           1       0.68      0.73      0.70       573
           2       0.74      0.74      0.74       619
           3       0.66      0.73      0.70       619
           4       0.86      0.95      0.90       585
           5       0.88      0.81      0.84       594
           6       0.91      0.72      0.81       629
           7       0.81      0.70      0.75       594
           8       0.71      0.77      0.74       581
           9       0.66      0.64      0.65       577

    accuracy                           0.76      6000
   macro avg       0.76      0.76      0.76      6000
weighted avg       0.76      0.76      0.76      6000
```

شکل ۲۵ عملکرد MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی ReLU

مشخصا با کاهش تعداد نورون‌های لایه‌ها نسبت به مدل‌هایی که نورون‌های بیشتری داشتند بدتر عمل کرده است.

نتیجه‌گیری و مقایسه

- بهترین نتیجه برای مدل دو لایه با ۲۳ نورون در هر دو لایه مخفی و تابع ReLU به عنوان تابع فعال‌ساز بود.
- تابع sigmoid خوب عمل می‌کند اما با افزایش لایه‌ها احتمالا به دلیل vanishing gradient سرعت همگرایی کمتری نسبت به ReLU دارد. علاوه بر این ReLU دارای سرعت همگرایی بیشتری می‌باشد.
- همچنین با افزایش نورون‌های لایه مخفی مدل بهتر عمل می‌کند و مشخصا با کاهش آن مدل ضعیف‌تر می‌شود.

به صورت کلی افزایش لایه، نورون و استفاده از تابع ReLU بهترین نتیجه را می‌دهد.

- **ضمیمه**

الف- **MLP با یک لایه‌ی مخفی**

Single-Layer MLPs with Sigmoid Activation Functions

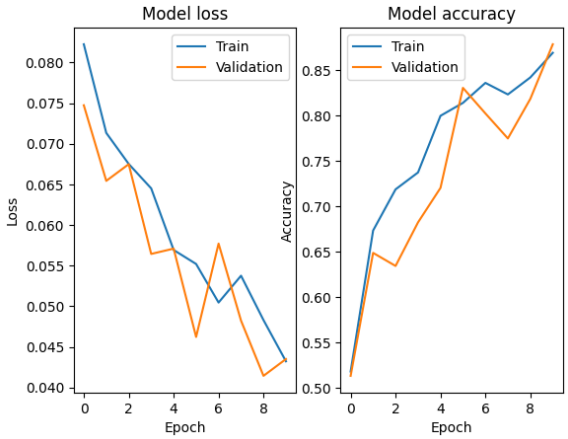
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	نرخ یادگیری	رویکرد آموزش
۵ دوره	SoftMax	Bipolar Sigmoid	MSE	10	36	$\frac{(36+10)}{2}=23$	0.001	SGD

جدول 9 مشخصات MLP تک لایه با تابع فعال‌سازی Sigmoid

برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

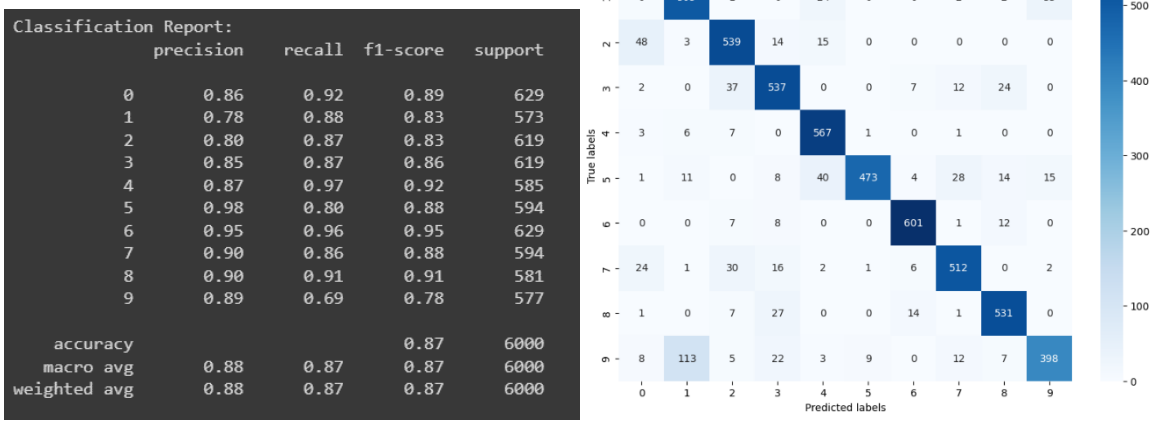
نمودار خطا و دقت در طول آموزش



شکل ۲۶ MLP تک لایه با تابع فعال‌سازی Sigmoid در طول آموزش

با توجه به نمودار بالا متوجه می‌شویم که این مدل عملکرد بهتری نسبت به مدل با ویژگی‌های کمتر دارد و همچنان جا برای آموزش دارد، اما به واسطه طولانی بودن فرایند آن فقط ۵ دوره آموزش دیده است.

ماتریس درهمریختگی و Classification report



شکل 27 عملکرد MLP تک لایه با تابع فعال‌سازی Sigmoid

مدل دقت ۸۷٪ روی داده‌های آزمون نشان می‌دهد که برای یک مدل تک لایه عملکرد بسیار مناسبی است.

ب- MLP با دو لایه‌ی مخفی

- **Two-Layer MLPs with Sigmoid Activation Functions (H1=23, H2=23)**

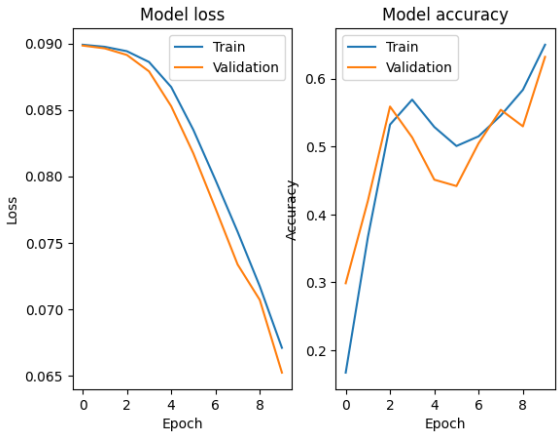
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی اول	تابع فعال‌سازی لایه مخفی دوم	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش
۵ دوره	SoftMax	Bipolar Sigmoid	Bipolar Sigmoid	MSE	10	36	$\frac{(36+10)}{2}=23$	$\frac{(36+10)}{2}=23$	0.001	SGD

جدول 10 MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی Sigmoid

برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

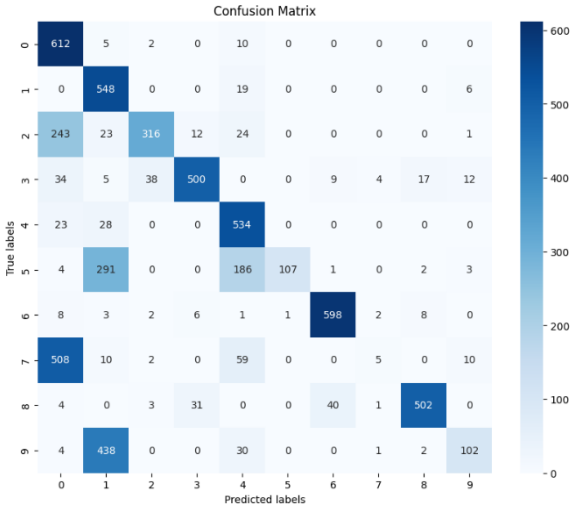
نمودار خطا و دقت در طول آموزش



شکل ۲۸ MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی Sigmoid در طول آموزش

این مدل نیز کمی دیرتر همگرا شده است که احتمالا به خاطر vanishing gradient باشد.

ماتریس درهمریختگی و Classification report




```
Classification Report:
              precision    recall  f1-score   support

           0       0.42      0.97      0.59       629
           1       0.41      0.96      0.57       573
           2       0.87      0.51      0.64       619
           3       0.91      0.81      0.86       619
           4       0.62      0.91      0.74       585
           5       0.99      0.18      0.30       594
           6       0.92      0.95      0.94       629
           7       0.38      0.01      0.02       594
           8       0.95      0.86      0.90       581
           9       0.76      0.18      0.29       577

    accuracy                           0.64      6000
   macro avg       0.72      0.63      0.58      6000
weighted avg       0.73      0.64      0.59      6000
```

شکل ۲۹ عملکرد MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی Sigmoid

- **Two-Layer MLPs with Sigmoid Activation Functions (H1=24, H2=18)**

با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

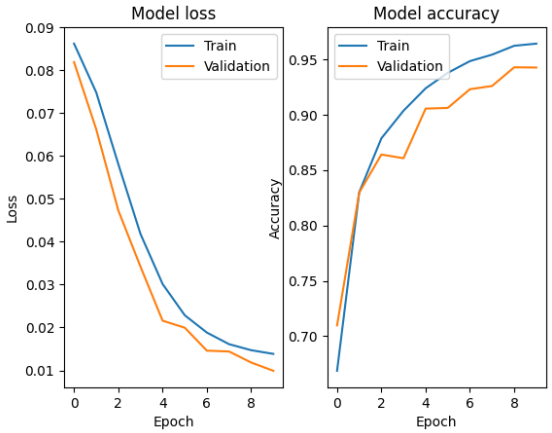
تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش

							لایه مخفی دوم	لایه مخفی اول		
SGD	0.001	$\frac{(36)}{2} = 18$	$\frac{2 * (36)}{3} = 24$	36	10	MSE	Bipolar Sigmoid	Bipolar Sigmoid	SoftMax	۵۰۰ دوره

جدول ۱۱ MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی Sigmoid

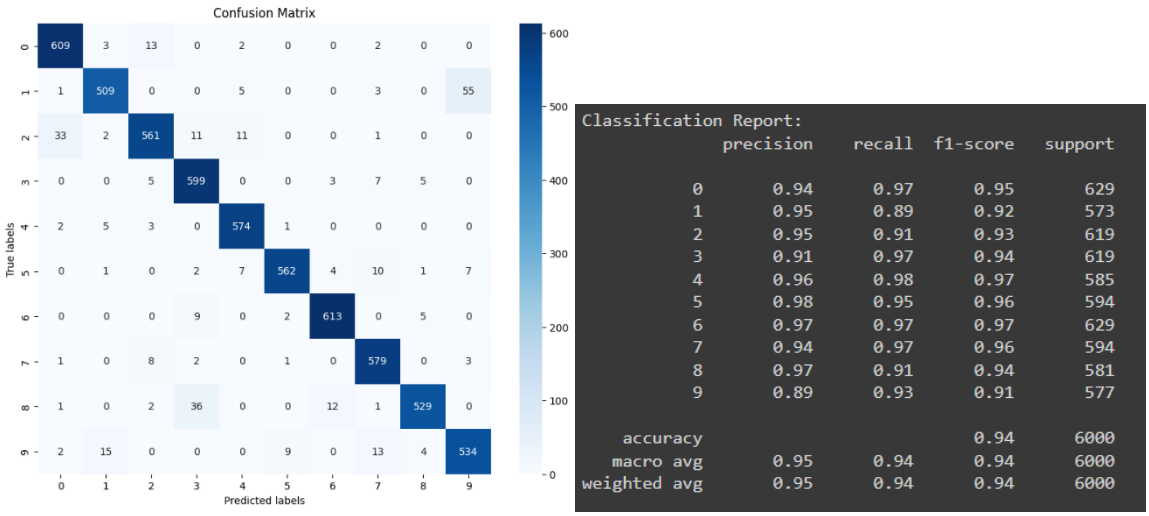
برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش



شکل ۳۰ MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی Sigmoid در طول آموزش

ماتریس درهمریختگی و Classification report



شکل ۳۱ عملکرد MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی Sigmoid

- **Two-Layer MLPs with Sigmoid Activation Functions (H1=18, H2=12)**

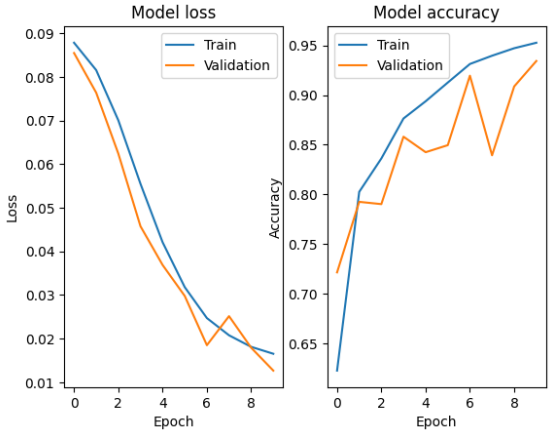
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی اول	تابع فعال‌سازی لایه مخفی دوم	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	Bipolar Sigmoid	Bipolar Sigmoid	MSE	10	36	$\frac{(36)}{2} = 18$	$\frac{(36)}{3} = 12$	0.001	SGD

جدول ۱۲ MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی Sigmoid

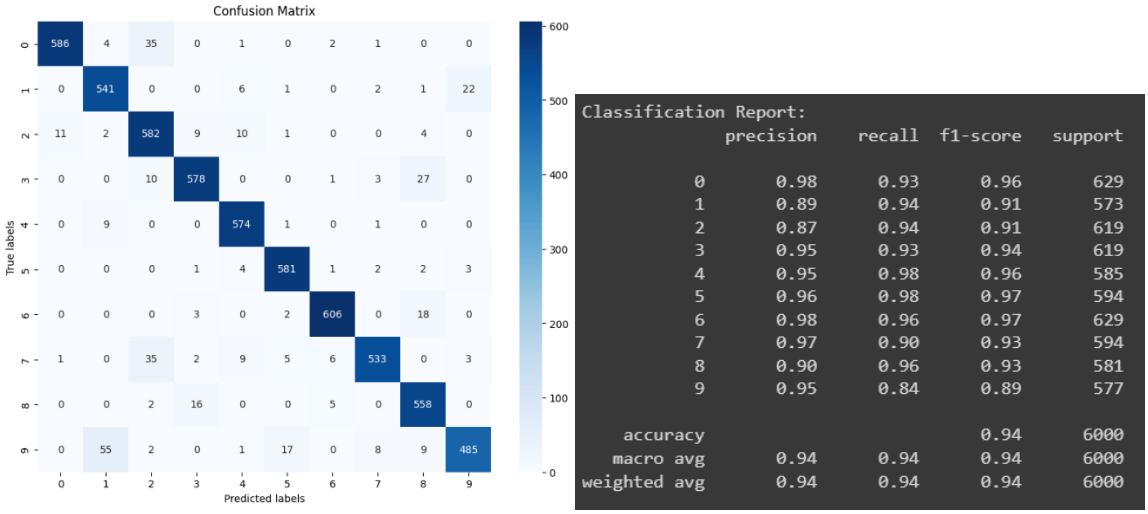
برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش



شکل ۳۲ MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی Sigmoid در طول آموزش

ماتریس درهمریختگی و Classification report



شکل ۳۳ عملکرد MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی Sigmoid

پ- MLP با یک لایه و دو لایه‌ی مخفی و تابع ReLU

Single-Layer MLPs with ReLU Activation Functions

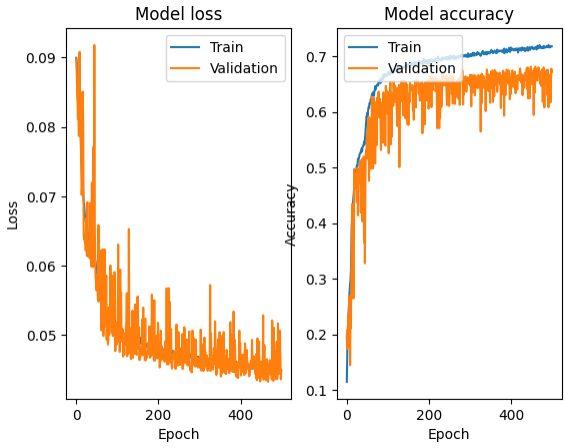
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	ReLU	MSE	10	36	$\frac{(36+10)}{2}=23$	0.001	SGD

جدول 13 مشخصات MLP تک لایه با تابع فعال‌سازی ReLU

برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش

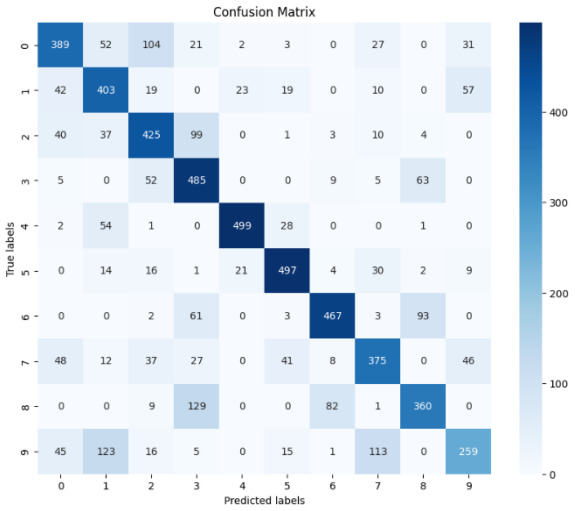


شکل ۳۴ MLP تک لایه با تابع فعال‌سازی ReLU در طول آموزش

با توجه به نمودار بالا متوجه می‌شویم که مدل نتوانسته الگوی مورد نظر را یاد بگیرد و احتمالا به خاطر این است که یک لایه RelU غیر خطی بودن لازم را فراهم نمی‌کند.

ماتریس درهمریختگی و Classification report

```
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.62      0.65       629
           1       0.58      0.70      0.64       573
           2       0.62      0.69      0.65       619
           3       0.59      0.78      0.67       619
           4       0.92      0.85      0.88       585
           5       0.82      0.84      0.83       594
           6       0.81      0.74      0.78       629
           7       0.65      0.63      0.64       594
           8       0.69      0.62      0.65       581
           9       0.64      0.45      0.53       577

    accuracy                           0.69      6000
   macro avg       0.70      0.69      0.69      6000
weighted avg       0.70      0.69      0.69      6000
```



شکل 35 عملکرد MLP تک لایه با تابع فعال‌سازی ReLU

- **Two-Layer MLPs with ReLU Activation Functions (H1=23, H2=23)**

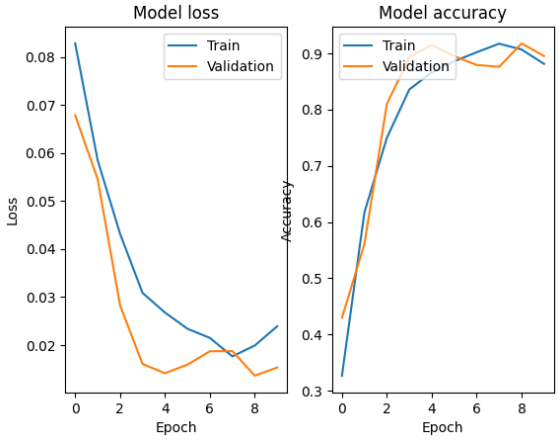
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی اول	تابع فعال‌سازی لایه مخفی دوم	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	ReLU	ReLU	MSE	10	36	$\frac{(36+10)}{2}=23$	$\frac{(36+10)}{2}=23$	0.001	SGD

جدول 14 MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی ReLU

برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

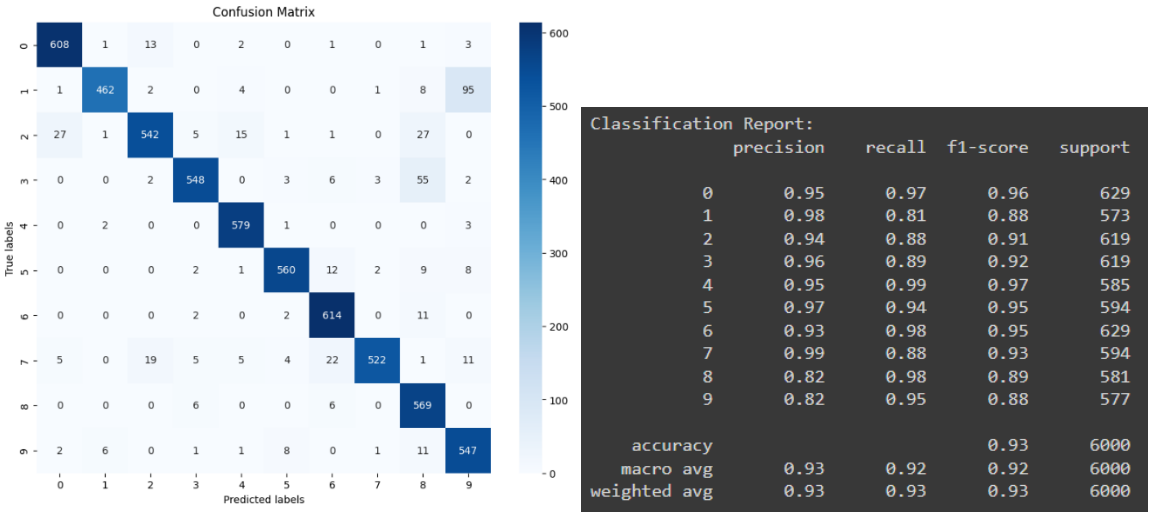
نمودار خطا و دقت در طول آموزش



شکل ۳۶ MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی ReLU در طول آموزش

این مدل با زیاد شدن لایه، عملکرد بهتری داشته و در حال یادگیری الگوی داده‌ها است.

ماتریس درهمریختگی و Classification report



شکل ۳۷ عملکرد MLP دو لایه (۲۳و۲۳) با تابع فعال‌سازی ReLU

- **Two-Layer MLPs with ReLU Activation Functions (H1=24, H2=18)**

با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

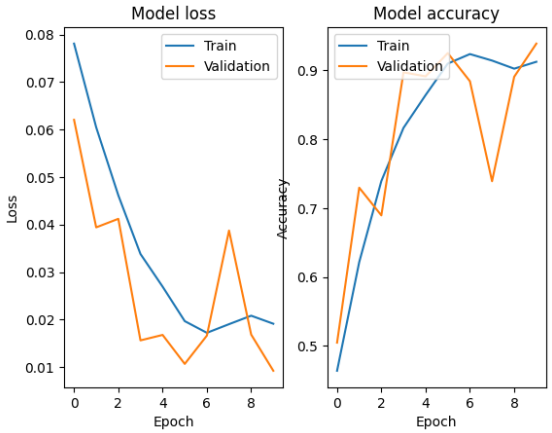
تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی	تابع فعال‌سازی	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش

							لایه مخفی دوم	لایه مخفی اول		
SGD	0.001	$\frac{(36)}{2} = 18$	$\frac{2 * (36)}{3} = 24$	36	10	MSE	ReLU	ReLU	SoftMax	۵۰۰ دوره

جدول ۱۵ MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی ReLU

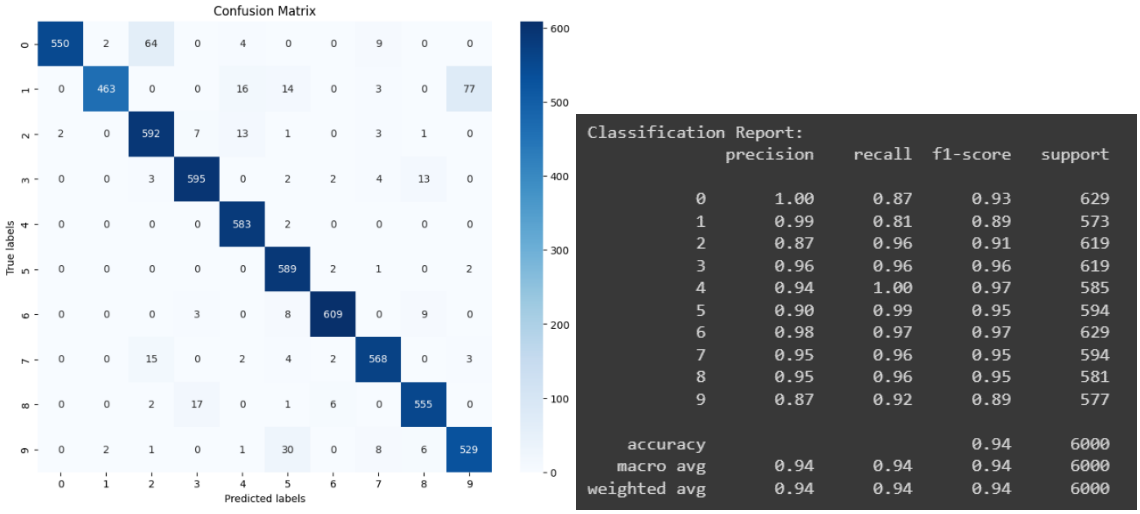
برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش



شکل ۳۸ MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی ReLU در طول آموزش

ماتریس درهمریختگی و Classification report



شکل ۳۹ عملکرد MLP دو لایه (۲۴و۱۸) با تابع فعال‌سازی ReLU

- **Two-Layer MLPs with ReLU Activation Functions (H1=18, H2=12)**

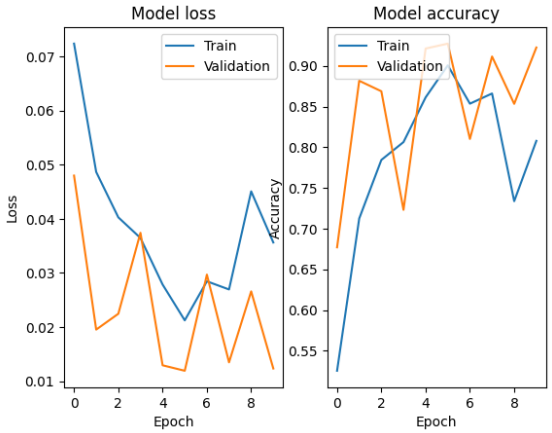
با توجه به خواسته‌های سوال که شامل جدول شماره ۱ می‌باشد، مدلی آموزش داده‌ایم.

تعداد دفعات آموزش	تابع فعال‌سازی خروجی	تابع فعال‌سازی لایه مخفی اول	تابع فعال‌سازی لایه مخفی دوم	تابع خطا	تعداد نورون خروجی	تعداد نورون ورودی	تعداد نورون لایه مخفی	تعداد نورون لایه مخفی دوم	نرخ یادگیری	رویکرد آموزش
۵۰۰ دوره	SoftMax	ReLU	ReLU	MSE	10	36	$\frac{(36)}{2} = 18$	$\frac{(36)}{3} = 12$	0.001	SGD

جدول ۱۶ MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی ReLU

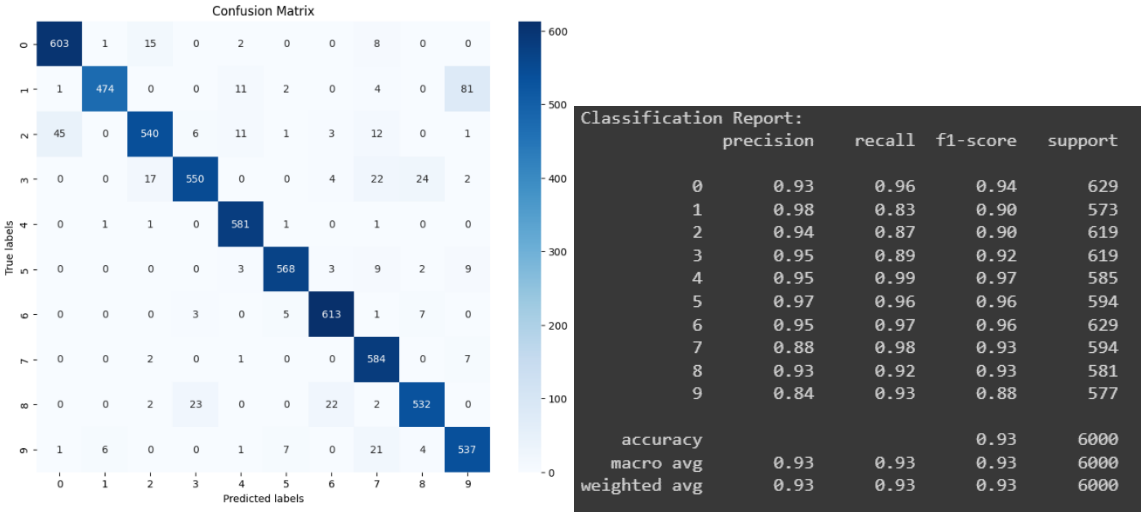
برای این مدل ۰.۲ از داده‌های آموزش را برای ارزیابی در طول آموزش جدا می‌کنیم. نتایج این مدل به شرح زیر می‌باشد:

نمودار خطا و دقت در طول آموزش



شکل ۴۰ MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی ReLU در طول آموزش

ماتریس درهمریختگی و Classification report



شکل ۴۱ عملکرد MLP دو لایه (۱۸و۱۲) با تابع فعال‌سازی ReLU

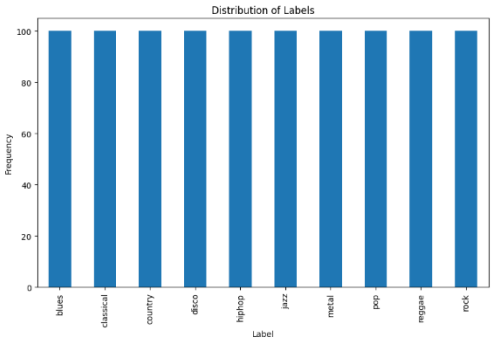
مشخصا با کاهش تعداد نورون‌های لایه‌ها نسبت به مدل‌هایی که نورون‌های بیشتری داشتند بدتر عمل کرده است.

- **مقایسه و نتیجه‌گیری کلی**

ویژگی‌های استفاده شده در این که بدون میانگین‌گیری می‌باشد، به صورت کلی دقت بالاتری روی داده‌های می‌دهند که دلیل آن هم وجود ویژگی‌ها بیشتر است و شبکه‌های بزرگتر و قوی تر است. سایر نتایج و تحلیل‌ها در مورد تعداد نورون‌ها، توابع فعال‌سازی مانند قسمت قبل می‌باشد.

سوال ۳: تشخیص ژانر موسیقی با شبکه عصبی پیچشی

ابتدا مجموعه داده را از سایت کگل دانلود می‌کنیم و به بررسی‌های اولیه می‌پردازیم. توزیع داده‌ها در همه‌ی کلاس‌ها یکسان است، بنابراین نیاز به پیش‌پردازش اولیه‌ای در این بخش نداریم.



شکل ۴۲ توزیع داده‌های مجموعه‌داده‌ی gtzan

از آنجا که می‌خواهیم از ویژگی mel-spectogram استفاده کنیم نیز نیازی به یکسان کردن طول همه‌ی داده‌ها وجود ندارد.

- **گام ۱- فرخوانی دادگان و بخش‌بندی آن**

داده‌ها را به نسبت ۸۰ و ۲۰ به دو دسته‌ی آموزش و آزمون تقسیم می‌کنیم. برای اینکار از کد زیر استفاده می‌کنیم:

```
X_train, X_test, Y_train, Y_test = train_test_split(features, Y,
test_size=0.2, random_state=42)
```

البته بدون استفاده از کتابخانه هم این کار ممکن است که تابع آن نیز در کد تعریف شده است.

- **گام ۲- ورودی شبکه**

برای همه‌ی داده‌ها با استفاده از extract_and_resize_spectrogram طیف نگار را استخراج می‌کنیم، هر طیف نگار یک ماتریس است و از آنجا که قصد ورودی دادن به شبکه CNN را داریم سایز همه‌ی این ماتریس‌ها را یکسان می‌کنیم. حال ابتدا ویژگی‌ها رو نورمالیز می‌کنیم تا مدل بهتر آموزش ببیند، علاوه بر این از آنجا که کلاس‌ها به صورت اعداد ۰ تا ۹ تعریف شده‌اند از OneHotEncoder نیز برای تبدیل آن‌ها به یک بردار OneHot استفاده می‌کنیم.

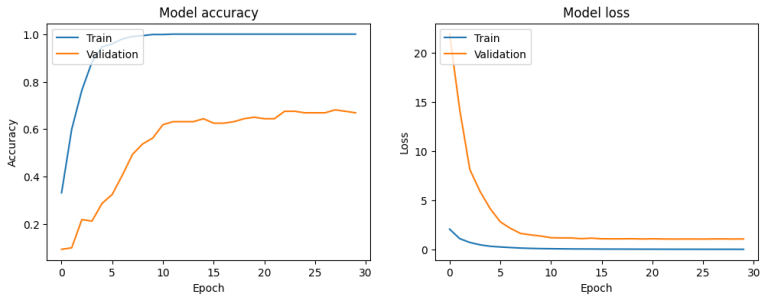
• گام ۳- ساخت و آموزش مدل

مدل را با توجه به خواسته‌های سوال آموزش می‌دهیم که معماری آن در شکل قابل رویت می‌باشد:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 126, 126, 32)      320

 max_pooling2d_2 (MaxPoolin  (None, 63, 63, 32)        0
 g2D)

 batch_normalization_2 (Bat  (None, 63, 63, 32)        128
 chNormalization)

 conv2d_3 (Conv2D)           (None, 61, 61, 32)        9248

 max_pooling2d_3 (MaxPoolin  (None, 31, 31, 32)        0
 g2D)

 batch_normalization_3 (Bat  (None, 31, 31, 32)        128
 chNormalization)

 flatten_1 (Flatten)         (None, 30752)             0

 dense_1 (Dense)             (None, 10)                307530

=================================================================
Total params: 317354 (1.21 MB)
Trainable params: 317226 (1.21 MB)
Non-trainable params: 128 (512.00 Byte)
_________________________________________________________________
```

شکل ۴۳ معماری مدل CNN

حال مدل را با بهینه‌سازی Adam، نرخ یادگیری ۰.۰۰۰۱ با batch ۳۲ برای ۳۰ دوره آموزش می‌دهیم و نتایج آن به شرح زیر می‌باشد:



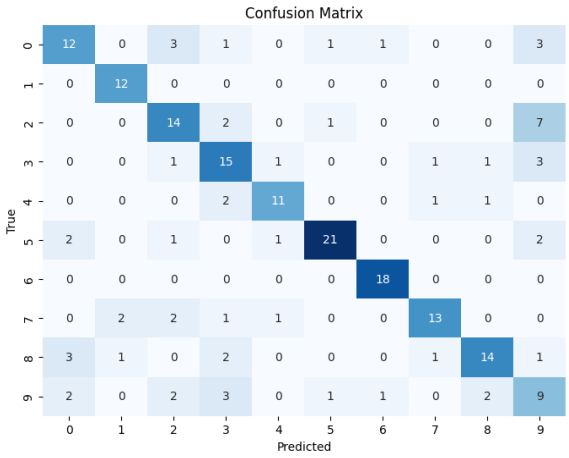
شکل ۴۴ عملکرد مدل CNN در طول آموزش

با توجه به شکل مشخص است که مدل به داده‌های آموزش overfit شده است و از جایی بیشتر نمی‌تواند، روی داده‌های ارزیابی خوب عمل کند، برای حل این مشکل باید از مدل عمیق‌تر یا معماری بهتری استفاده کنیم.

- **گام ۴- آزمون مدل**

Classification Report:

	precision	recall	f1-score	support
0	0.63	0.57	0.60	21
1	0.80	1.00	0.89	12
2	0.61	0.58	0.60	24
3	0.58	0.68	0.62	22
4	0.79	0.73	0.76	15
5	0.88	0.78	0.82	27
6	0.90	1.00	0.95	18
7	0.81	0.68	0.74	19
8	0.78	0.64	0.70	22
9	0.36	0.45	0.40	20
accuracy			0.69	200
macro avg	0.71	0.71	0.71	200
weighted avg	0.71	0.69	0.70	200



شکل ۴۵ عملکرد مدل CNN روی داده‌های آزمون

مدل تا حد مناسبی قابلیت تشخیص داده‌های آزمون را دارد. با این حال با توجه به داده‌های ارزیابی متوجه شدیم که مدل overfit شده است و جا برای بهتر کردن این مدل وجود دارد.

سوال ۴: تشخیص گوینده

بخش الف)

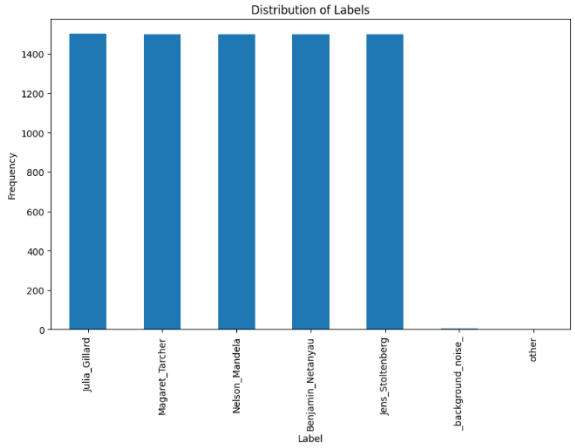
برای آموزش یک مدل تشخیص گوینده کارآمد، داده های آموزشی باید دارای ویژگی های زیر باشند:

- غنای آوایی: داده ها باید شامل طیف گسترده ای از واج ها، صامت ها و واکه ها باشد تا مدل بتواند ویژگی های آوایی مختلف را یاد بگیرد.
- کیفیت ضبط صدا: ضبط صدا باید با کیفیت بالا و بدون نویز زمینه یا اعوجاج باشد تا جزئیات صدای گوینده به درستی ضبط شود.
- حجم کافی داده: برای آموزش یک مدل قوی، به حجم زیادی داده نیاز است تا تنوع صداهای گوینده پوشش داده شود. اما میزان دقیق داده مورد نیاز بسته به پیچیدگی مسئله و معماری مدل متفاوت است. معمولا برای هر کلاس (گوینده)، حداقل ۳ تا ۵ دقیقه داده آموزشی با کیفیت بالا توصیه می شود. . اما هرچه حجم داده بیشتر باشد، عملکرد مدل بهتر خواهد بود. البته در بعضی منابع ۱ تا ۲ ساعت، حجم داده‌ی مورد نیاز ذکر شده است. در دیتاست مربوط به این سوال ۲۵ دقیقه، برای هر گوینده، داده داشتیم.
- تغییرپذیری گوینده: از هر گوینده باید چندین نمونه صدا در شرایط مختلف محیطی، زمانی و احساسی ضبط شود تا مدل بتواند گوینده را در موقعیت های گوناگون تشخیص دهد

همچنین در صورتی که قصد ساخت سیستمی داریم که برای هر گوینده، بردار ویژگی دربیارید که بتوان آن‌ها را باهم مقایسه کرد، به چند صد ساعت داده، به همراه تعداد زیادی گوینده نیاز با تفاوت زیاد نیاز دارم.

- **گام ۱- خواندن دادگان**

ابتدا مجموعه داده را از سایت کگل دانلود می‌کنیم و به بررسی‌های اولیه می‌پردازیم. توزیع داده‌ها در همه‌ی کلاس‌های اشخاص یکسان است، بنابراین نیاز به پیش‌پردازش اولیه‌ای در این بخش نداریم.



- **گام ۲- استخراج ویژگی**

با استفاده از تابع feature_extractor برای هر صوت ویژگی‌های خواسته شده در سوال شامل rms، zcr و spectral_centroid را استخراج می‌کنیم.

- **گام ۳- موازی‌سازی**

حال از آنجا که ویژگی‌ها و صوت‌ها به یکدیگر مرتبط نیستند، کافیست به صورت موازی هربار این تابع را صدا بزنیم، برای مقایسه تفاوت زمانی این عملکرد دو تابع به صورت زیر تعریف کرده‌ایم:

```
def extract_features_sequential(X):
    start_time = time.time()
    features = [feature_extractor(audio_file) for audio_file in X]
    end_time = time.time()
    execution_time = end_time - start_time
    return features, execution_time

# Function to extract features in parallel
def extract_features_parallel(X):
    start_time = time.time()
    with ThreadPoolExecutor(max_workers=4) as executor:
        # Map the feature extraction function to each audio file in
X
        features = list(executor.map(feature_extractor, X))
    end_time = time.time()
    execution_time = end_time - start_time
import joblib
start_time = time.time()
```

```
n_jobs=32
verbose=1
jobs = [ joblib.delayed(feature_extractor)(i) for i in X ]
features_parallel = joblib.Parallel(n_jobs=n_jobs, verbose=verbose)(jobs)
end_time = time.time()
time_parallel= end_time - start_time
print("Parallel execution time:", time_parallel, "seconds")
```

این دو تابع برای هر صوت تابع feature_extractor را صدا می‌کنند، ولی یکی از توابع منتظر استخراج ویژگی نمی‌ماند و به صورت موازی عمل می‌کند. مدت زمان اجرای هرکدام از این دو تابع به صورت زیر می‌باشد:

```
Sequential execution time: 105.72517657279968 seconds
Parallel execution time: 32.111114740371704 seconds
[Parallel(n_jobs=32)]: Using backend LokyBackend with 32 concurrent workers.
[Parallel(n_jobs=32)]: Done 208 tasks      | elapsed:    0.7s
[Parallel(n_jobs=32)]: Done 708 tasks      | elapsed:    1.9s
[Parallel(n_jobs=32)]: Done 1408 tasks      | elapsed:    3.5s
[Parallel(n_jobs=32)]: Done 2308 tasks      | elapsed:    5.7s
[Parallel(n_jobs=32)]: Done 3408 tasks      | elapsed:    8.2s
[Parallel(n_jobs=32)]: Done 4708 tasks      | elapsed:   11.3s
[Parallel(n_jobs=32)]: Done 6208 tasks      | elapsed:   14.9s
Parallel execution time: 17.855644464492798 seconds
[Parallel(n_jobs=32)]: Done 7438 out of 7501 | elapsed:   17.7s remaining:    0.1s
[Parallel(n_jobs=32)]: Done 7501 out of 7501 | elapsed:   17.7s finished
```

شکل ٤٦ مدت زمان استخراج ویژگی‌های صوت

زمان اجرا تقریبا با استفاده از متد معرفی شده در تمرین یک سوم می‌شود و متد دوم زمان را تقریبا یک ششم می‌کند که سرعت بالاتری است و بهتر است.

- **گام ۴- تقسیم‌بندی داده‌ها**

داده‌ها را به نسبت ۸۰ و ۲۰ به دو دسته‌ی آموزش و آزمون تقسیم می‌کنیم. برای اینکار از کد زیر استفاده می‌کنیم:

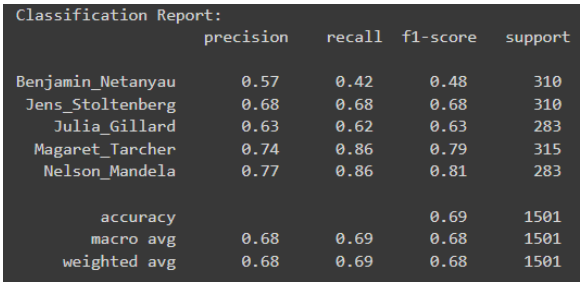
```
X_train, X_test, Y_train, Y_test = train_test_split(features, Y,
test_size=0.2, random_state=42)
```

البته بدون استفاده از کتابخانه هم این کار ممکن است که تابع آن نیز در کد تعریف شده است.

همچنین داده‌ها را با استفاده از StandardScaler نورمالیز می‌کنیم و برای اموزش مدل آماده می‌کنیم. (کد کتابخانه و بدون کتابخانه هردو موجود است)

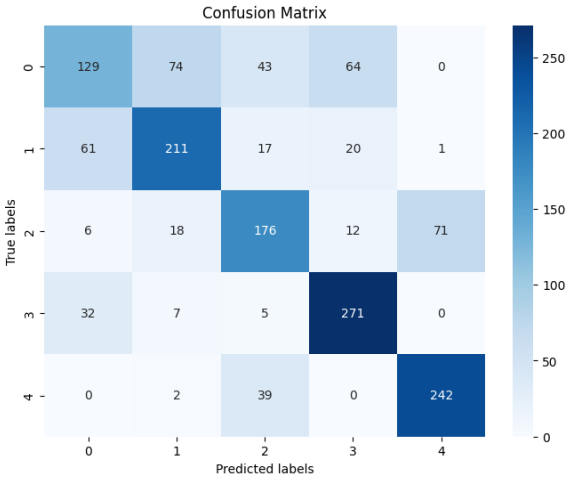
- **گام ۵- آموزش مدل**

مدل logistic regression را با استفاده از روش‌های مختلف شامل Sklearn، SGDClassifer و torch پیاده‌سازی شده است (چون مدل دیفالت Sklearn از SGD استفاده نمی‌کرد و learning rate هم نداشت. بنابراین از این دو روش دیگر نیز استفاده کرده‌ایم). و طبق خواسته‌های سوال آن آموزش می‌دهیم و نتایج هر سه روش در کد موجود می‌باشد، نتایج برای به شرح زیر می‌باشد.



Classification Report:

	precision	recall	f1-score	support
Benjamin_Netanyau	0.57	0.42	0.48	310
Jens_Stoltenberg	0.68	0.68	0.68	310
Julia_Gillard	0.63	0.62	0.63	283
Magaret_Tarcher	0.74	0.86	0.79	315
Nelson_Mandela	0.77	0.86	0.81	283
accuracy			0.69	1501
macro avg	0.68	0.69	0.68	1501
weighted avg	0.68	0.69	0.68	1501



همچنین عملکرد مدل با متریک‌های خواسته شده در سوال به شرح زیر می‌باشد:

```
Accuracy: 0.6855429713524317
Precision: 0.6759526418017952
Recall: 0.6868246911971977
F1 Score: 0.677734839096743
```

بخش ب)

- **گام ۱- خواندن دادگان**

این بخش با قسمت قبل مشترک است و تغییری نکرده است.

- **گام ۲- استخراج ویژگی**

با استفاده از تابع mfcc_feature_extractor برای هر صوت ویژگی‌های خواسته شده در سوال را استخراج می‌کنیم.

- **گام ۳- موازی‌سازی**

حال از آنجا که ویژگی‌ها و صوت‌ها به یکدیگر مرتبط نیستند، کافیست به صورت موازی هربار این تابع را صدا بزنیم، برای مقایسه تفاوت زمانی این عملکرد دو متد زیر را به کار می‌گیریم.

```
import joblib
start_time = time.time()
n_jobs=32
verbose=1
jobs = [ joblib.delayed(mfcc_feature_extractor)(i) for i in X ]
out = joblib.Parallel(n_jobs=n_jobs, verbose=verbose)(jobs)
end_time = time.time()
time_parallel= end_time - start_time
print("Parallel execution time:", time_parallel, "seconds")
```

```
def mfcc_extract_features_sequential(X):
    start_time = time.time()
    features = [mfcc_feature_extractor(audio_file) for audio_file
in X]
    end_time = time.time()
    execution_time = end_time - start_time
    return features, execution_time
```

این دو برای هر صوت تابع mfcc_feature_extractor را صدا می‌کنند، ولی یکی از توابع منتظر استخراج ویژگی نمی‌ماند و به صورت موازی عمل می‌کند. (برای اجرای موازی mfcc، امکان استفاده از متد قبلی نبود و برای موازی سازی در اینترنت این روش پیشنهاد شده بود) مدت زمان اجرای هرکدام از این دو تابع به صورت زیر می‌باشد:

```
Sequential execution time: 77.46701884269714 seconds
[Parallel(n_jobs=32)]: Using backend LokyBackend with 32 concurrent workers.
[Parallel(n_jobs=32)]: Done 136 tasks      | elapsed:   13.4s
[Parallel(n_jobs=32)]: Done 633 tasks      | elapsed:   15.7s
[Parallel(n_jobs=32)]: Done 2021 tasks      | elapsed:   20.6s
[Parallel(n_jobs=32)]: Done 3806 tasks      | elapsed:   25.3s
[Parallel(n_jobs=32)]: Done 5997 tasks      | elapsed:   30.4s
Parallel execution time: 42.93901968002319 seconds
```

شکل ٤٧ مدت زمان اجرای برای استخراج ویژگی MFCC

زمان اجرا تقریبا نصف شده است.

- **گام ۴- تقسیم‌بندی داده‌ها**

داده‌ها را به نسبت ۸۰ و ۲۰ به دو دسته‌ی آموزش و آزمون تقسیم می‌کنیم. برای اینکار از کد زیر استفاده می‌کنیم:

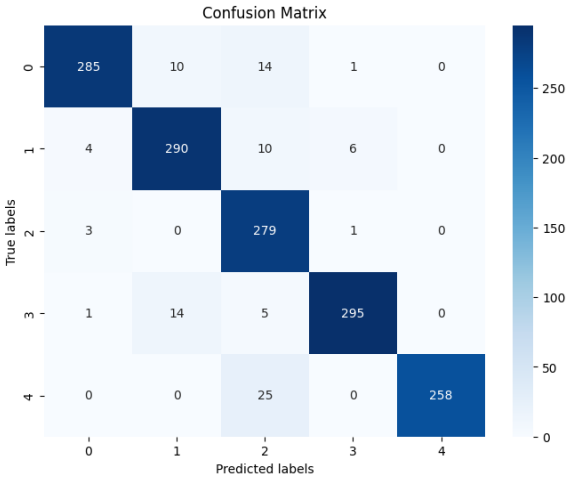
```
X_train, X_test, Y_train, Y_test = train_test_split(features, Y,
test_size=0.2, random_state=42)
```

البته بدون استفاده از کتابخانه هم این کار ممکن است که تابع آن نیز در کد تعریف شده است.

همچنین داده‌ها را با استفاده از StandardScaler نورمالیز می‌کنیم و برای اموزش مدل آماده می‌کنیم. (کد کتابخانه و بدون کتابخانه هردو موجود است)

- **گام ۵- آموزش مدل**

مدل logistic regression را با استفاده از روش‌های مختلف شامل Sklearn، SGDClassifer و torch پیاده‌سازی شده است (چون مدل دیفالت Sklearn از SGD استفاده نمی‌کرد و learning rate هم نداشت. بنابراین از این دو روش دیگر نیز استفاده کرده‌ایم). و طبق خواسته‌های سوال آن آموزش می‌دهیم و نتایج هر سه روش در کد موجود می‌باشد، نتایج برای به شرح زیر می‌باشد.




```
Classification Report:
                    precision    recall  f1-score   support

Benjamin_Netanyau       0.97      0.92      0.95       310
 Jens_Stoltenberg       0.92      0.94      0.93       310
    Julia_Gillard       0.84      0.99      0.91       283
 Magaret_Tarcher        0.97      0.94      0.95       315
   Nelson_Mandela       1.00      0.91      0.95       283

         accuracy                           0.94      1501
        macro avg       0.94      0.94      0.94      1501
     weighted avg       0.94      0.94      0.94      1501
```

همچنین عملکرد مدل با متریک‌های خواسته شده در سوال به شرح زیر می‌باشد:

```
Accuracy: 0.9373750832778148
Precision: 0.9415396644576983
Recall: 0.9377746295904281
F1 Score: 0.9378173605838379
```

بخش پ)

چندین راه برای حل این مشکل پیشنهاد می‌شود:

- اضافه کردن کلاس برای داده‌ی out of domain: علاوه بر کلاس‌هایی مربوط به افراد مورد نظر خود که داریم، کلاس دیگری شامل همه‌ی صوت‌های دیگری که این اشخاص نیستند داشته باشیم، که مدل هم صدای این افراد را یاد بگیرد و هم صدایی که جز این افراد نیست را بتواند Other تشخیص بدهد.
- استفاده از آستانه مدل: یک راه‌حل ساده، می‌تواند استفاده از تعیین یک آستانه احتمال برای مدل باشد به این معنی که احتمالی که مدل برای آن کلاس می‌دهد اگر از حدی پایین تر بودن، برنامه‌ی ما آن صدا را ناشناس تشخیص بدهد.
- استفاده از یک مدل دسته‌بندی دو کلاسه: این مدل فقط تشخیص می‌دهد که صوت ورودی عضوی از دیتاست ما هست یا خیر و بعد برای دسته‌بندی به مدل اصلی می‌دهد.