دانشگاه شهید بهشتی

دانشکده برق

پایان‌نامه کارشناسی مهندسی برق

پردازش سیگنال نوار قلبی (ECG) بوسیله شبکه های عصبی در پایتون

استاد راهنما:

دکتر علیرضا یزدی زاده

نگارش:

سجاد رضوانی خالدی

زمستان ۱۴۰۲


بسم الله الرحمن الرحيم

تشکر و قدردانی

از استاد گرانقدر جناب آقای دکتر علیرضا یزدی‌زاده که زحمت راهنمایی این پایان نامه را بر عهده داشتند، کمال تشکر را دارم. همچنین از پدر و مادر دلسوزم بابت حمایت های بی دریغشان، سپاسگزارم.

سجاد رضوانی خالدی

آیین‌نامه حق مالکیت مادی و معنوی در مورد نتایج پژوهشهای علمی دانشگاه شهید بهشتی

مقدمه: با عنایت به سیاست‌های پژوهشی و فناوری دانشگاه در راستای تحقق عدالت و کرامت انسان‌ها که لازمه شکوفایی علمی و فنی است و رعایت حقوق مادی و معنوی دانشگاه و پژوهشگران، لازم است اعضای هیأت علمی، دانشجویان، دانش‌آموختگان و دیگر همکاران طرح، در مورد نتایج پژوهشهای علمی که تحت عناوین پایان‌نامه، رساله و طرحهای تحقیقاتی با هماهنگی دانشگاه انجام شده‌است، موارد زیر را رعایت نمایند:

ماده ۱- حق نشر و تکثیر پایان نامه/ رساله و درآمدهای حاصل از آنها متعلق به دانشگاه می‌باشد ولی حقوق معنوی پدید آورندگان محفوظ خواهد بود.

ماده ۲- انتشار مقاله یا مقالات مستخرج از پایان‌نامه/ رساله به صورت چاپ در نشریات علمی و یا ارائه در مجامع علمی باید به نام دانشگاه بوده و با تایید استاد راهنمای اصلی، یکی از اساتید راهنما، مشاور و یا دانشجو مسئول مکاتبات مقاله باشد. ولی مسئولیت علمی مقاله مستخرج از پایان نامه و رساله به عهده اساتید راهنما و دانشجو می‌باشد.

تبصره: در مقالاتی که پس از دانش‌آموختگی بصورت ترکیبی از اطلاعات جدید و نتایج حاصل از پایان‌نامه/ رساله نیز منتشر می‌شود نیز باید نام دانشگاه درج شود.

ماده ۳- انتشار کتاب، نرم افزار و یا آثار ویژه (اثری هنری مانند فیلم، عکس، نقاشی و نمایشنامه) حاصل از نتایج پلیان‌نامه/ رساله و تمامی طرحهای تحقیقاتی کلیه واحدهای دانشگاه اعم از دانشکده‌ها، مراکز تحقیقاتی، پژوهشکده‌ها، پارک علم و فناوری و دیگر واحدها باید با مجوز کتبی صادره از معاونت پژوهشی دانشگاه و براساس آئین نامه های مصوب انجام شود.

ماده ۴- ثبت اختراع و تدوین دانش فنی و یا ارائه یافته ها در جشنواره‌های ملی، منطقه‌ای و بین‌المللی که حاصل نتایج مستخرج از پایان‌نامه/ رساله و تمامی طرح‌های تحقیقاتی دانشگاه باید با هماهنگی استاد راهنما یا مجری طرح از طریق معاونت پژوهشی دانشگاه انجام گیرد.

ماده ۵- این آیین‌نامه در ۵ ماده و یک تبصره در تاریخ ..... در شورای پژوهشی و در تاریخ ..... در هیأت رئیسه دانشگاه به تایید رسید و در جلسه مورخ ..... شورای دانشگاه به تصویب رسیده و از تاریخ تصویب در شورای دانشگاه لازم‌الاجرا است.

"اینجانب **سجاد رضوانی خالدی** دانشجوی رشته **مهندسی برق** ورودی سال تحصیلی **۱۳۹۸** مقطع **کارشناسی** دانشکده **مهندسی برق** متعهد می‌شوم کلیه نکات مندرج در آئین نامه حق مالکیت مادی و معنوی در مورد نتایج پژوهش‌های علمی دانشگاه **شهید بهشتی** را در انتشار یافته‌های علمی مستخرج از پایان نامه/ رساله تحصیلی خود رعایت نمایم. در صورت تخلف از مفاد آئین نامه فوق الاشعار به دانشگاه وکالت و نمایندگی می‌دهم که از طرف اینجانب نسبت به لغو امتیاز اختراع به‌نام بنده و یا هر گونه امتیاز دیگر و تغییر آن به نام دانشگاه اقدام نماید. ضمناً نسبت به جبران فوری ضرر و زیان حاصله بر اساس برآورد دانشگاه اقدام خواهم نمود و بدینوسیله حق هر گونه اعتراض را از خود سلب نمودم."

امضا:

تاریخ:

چکیده

بیماری های قلبی یکی از علل اصلی مرگ و میر در میان انسان ها در سراسر دنیا هستند که هرساله جان بسیاری را می‌گیرند. تحلیل سیگنال های الکتریکی قلب که به الکتروکاردیوگرام یا نوار قلب معروف هستند، از اصلی ترین روش های تشخیص این بیماری های قلبی هستند که توسط پزشکان صورت می‌گیرد تا از بیماری های قلبی پیشگیری کنند یا پس از دچار شدن بیمار به آنها، به درمان بیماری بپردازند. حال از آنجایی که تشخیص بیماری های قلبی توسط سیستم های کامپیوتری ازجمله هوش مصنوعی دارای دقت و سرعت بالاتری نسبت به تشخیص انسانی دارد، مدل های مختلفی در این حوزه ارائه شده است که همگی سعی دارند تا دقت یا سرعت پردازش سیگنال و تشخیص بیماری را افزایش بدهند یا رویکردی جدید را برای این تشخیص ارائه دهند ولی در این بین چالش هایی از جمله پیچیدگی سیگنال ها یا کمبود داده در نتیجه اورفیت شدن مدل ها مطرح است که باید بر آنها غلبه کرد. پژوهش ما با تست و بررسی مدل هایی بر پایه شبکه های LSTM و CNN بروی دیتاست PTB-XL و مقایسه دقت نهایی این مدل ها به مدلی بر پایه CNN یک بعدی با دقت مناسب دست پیدا کرد که با دقت ۷۹.۲۴ درصد به پیش‌بینی بیماری سابجکت در ۵ کلاس می‌پردازد و با توجه به بررسی های انجام شده در قسمت پیشینه پژوهش که بهترین دقت مقاله اول ۷۶درصد و بهترین دقت مقاله دوم ۷۴.۹-۷۹.۱ درصد بود، دقت بدست آماده در این پژوهش (۷۹.۲۴) بهتر از این مدل ها عمل می‌کند و به بهترین دقت مقاله سوم که ۷۷.۹-۸۰.۲ درصد بود، بسیار نزدیک است در میان بازه آن قرار دارد.

واژه های کلیدی- نوار قلب، دسته‌بندی سیگنال قلبی، شبکه های عصبی، یادگیری عمیق.

فهرست

فهرست شکل ها

فهرست جداول

فصل ۱– مقدمه ای بر پژوهش

۱-۱- پیشگفتار

امروزه تحقیقات مبتنی بر هوش مصنوعی و شبکه‌های عصبی در حوزه پردازش و دسته‌بندی سیگنال‌های قلبی از اهمیت بسیار زیادی برخوردار است زیرا می‌تواند تاثیر به سزایی در زندگی و سلامتی انسان ها داشته باشد. پروژه حاضر به منظور بررسی روش‌های تحلیل سیگنال‌های قلبی با استفاده از روش های هوش مصنوعی و شبکه های عصبی برای رسیدن به دقت بالا برای تشخیص بیماری‌های قلبی طراحی شده است.

این تحقیق به منظور مقابله با چالش‌های موجود در تحلیل سیگنال‌های قلبی، از جمله پیچیدگی‌های الگوهای قلبی، به روش‌های شبکه عصبی و یادگیری عمیق متکی است. امیدواریم که این تحقیق عملکرد سیستم‌های پزشکی را در تشخیص و پیش‌بینی بیماری‌های قلبی ارتقا داده و به بهبود بیماران کمک نمایند.

۱-۲- بیان مسئله

انسان همواره سعی داشته در زمینه های مختلف که یکی از مهم ترین آن ها پزشکی است از تکنولوژی کمک بگیرد تا جای ممکن به دقت، سرعت و راحتی بالاتری در امور مربوط به پزشکی و سلامتی خود دست پیدا کند. در این پژوهش مسئله اصلی این است که بررسی سیگنال های قلبی که در حال حاضر در مراکز درمانی توسط پزشکان صورت می‌گیرد، می‌تواند با کمک هوش مصنوعی، با دقت، سرعت و دسترسی بیشتری صورت گیرد. برای پی بردن به جزئیات این مسئله به این مثال توجه کنید. زمانی که افراد برای هرنوع بیماری جدیدی که به تازگی علائم آن را احساس کرده اند، به بیمارستان مراجعه می‌کنند، یکی از اولین روش های تشخیص اولیه بیماری ها در اورژانس، گرفتن نوار قلب و بررسی آن توسط پزشک متخصص است. زیرا نوار قلب می‌تواند یکی از روش های جامع تشخیص بیماری ها به صورت اولیه باشد. ولی مشکل اصلی این است که تشخیص آن به دانش و تخصص بسیار توسط پزشک نیاز دارد بنابراین مسئله اول این است که همواره پزشکانی با این حد از دانش و تخصص در دسترس نیستند. علاوه بر این بررسی تعداد زیادی نوار قلب برای پزشک متخصص در زمان محدودی که دارد، زمان بر است. بنابراین استفاده پیوسته از پزشک برای چکاپ و پیشگیری بیماری ها توسط عموم مردم می‌تواند بسیار هزینه‌بر و غیر بهینه باشد. در آخر مهم‌تر از همه این ها، به هرحال پزشکان دارای خطای انسانی هستند و باتوجه به اهمیت و حساسیت تشخیص یا پیشگیری بیماری ها، هرچه قدر بتوان دقت و سرعت پردازش این سیگنال های حیاتی را به وسیله سیستم های هوش مصنوعی افزایش داد، بسیار ارزشمند خواهد بود. ناگفته نماند که عدم تخصص بسیاری از پزشکان مخصوصا در افراد کم تجربه‌تر و تازه‌کار باعث کاهش دقت و بروز خطای دوچندان در تشخیص بیماری می‌شود که میتواند باعث ضرر های جبران ناپذیر شود.

۱–۳– اهداف کلی پروژه

هدف از این پروژه طراحی مدلی بر پایه شبکه عصبی است که در ورودی ۱۲ سیگنال نوار قلبی سابجکت را می‌گیرد و با دقت خوبی در خروجی شبکه، کلاس بیماری قلبی را در یکی از ۵ کلاس طبقه بندی می‌کند.

برای حل مسئله موجود با این روش باید توجه کرد که سیستم های کامپیوتری به خصوص پژوهش ما، روش هایی بر پایه هوش مصنوعی و شبکه عصبی، می‌توانند عملکرد خوبی در سیستم هایی که داده زیادی از آن ها در دست است، از جمله سیستم های پزشکی داشته باشد. بنابراین با یاری گرفتن از این سیستم ها در حوزه پزشکی می‌توان بر همه ی نواقص سیستم های تمام انسانی از جمله موارد زیر غلبه کرد:

اول، سیستم کامپیوتری می‌تواند همواره در دسترس باشد و مانند سیستم های انسانی خسته نمی‌شود و به استراحت نیاز ندارد. همچنین با پیشرفت چشم‌گیر و همه گیری سخت افزار های کامپیوتری، امکان دسترسی به سیستم های محاسباتی در لحظه برای بسیاری آسان تر شده است.

دوم، به همان دلیل مورد اول، برخلاف سیستم های انسانی، با استفاده از سیستم های کامپیوتری می‌توان به صورت مرتب و در زمان های مشخص داده را برای پردازش توسط این الگوریتم ها فرستاد و از این روش برای چکاپ منظم افراد با هزینه بسیار کمتر و دسترسی آسان بجای پزشک متخصص استفاده کرد.

سوم، سیستم های کامپیوتری بخصوص هوش مصنوعی همواره ثابت کرده اند که می‌توانند در تسک های مختلف ریاضیاتی و پردازش سیگنالی سریع‌تر و دقیق‌تر از انسان ها باشند.

۱–۴– پرسش‌های پژوهش

آیا هوش مصنوعی می‌تواند تحلیل سیگنال های حیاتی بدن را مانند سیگنال قلبی ECG که به نوار قلب معروف است با دقت قابل قبولی انجام دهد؟

از چه الگوریتم های هوش مصنوعی برای پردازش سیگنال های حیاتی قلب می‌توان استفاده کرد و دقت آنها چه قدر است؟

۱–۵– اهمیت و ضرورت پژوهش

بیماری های قلبی یکی از موارد شایع مرگ و میر در میان افراد مخصوصا در سنین بالاتر هستند. این بیماری علت اصلی مرگ و میر در بین مردان، زنان و افراد زیادی از گروه‌های نژادی و اقوامی مختلف در ایالات

متحده اســـت. هر ۳۳ ثانیه یک نفر در ایالات متحده به علت بیماری قلبی عروقی جان خود را از دســـت می‌دهد. حدود ۶۹۵۰۰۰ نفر در ایالات متحده در سال ۲۰۲۱ از بیماری قلبی جان خود را از دست دادند که این به معنی یک نفر در هر پنج نفر اســت.[۱] همچنین در ایالات متحده، هر ســاله حدود ۸۰۵۰۰۰ نفر در ایالات متحده سکته قلبی می‌کنند و به صورت میانگین هر ۴۰ ثانیه یک نفر سکته قلبی می‌کند که نشان دهنده آمار بسیار قابل توجهی است. [۱] همه‌ی این آمار ها نشان دهنده اهمیت و ضرورت پیشگیری و درمان این عارضه است زیرا می‌تواند باعث نجات جان بسیاری از انسان ها شود. تشخیص بیماری‌های قلبی معمولاً بر اساس تاریخچه بالینی بیمار، تست های فیزیکی و تست‌های تشخیصی صورت می‌گیرد. ECG یکی از معروف‌ترین و پرکاربردترین تســت‌های قلبی اســت. این تســت برای ثبت فعالیت الکتریکی قلب به کمک الکترودها انجام می‌شود. الکترودها بر روی پوست بدن قرار می‌گیرند و الگوی نوسانات الکتریکی قلب را ثبت می‌کنند. الگوهای نوسانی ثبت شده توسط ECG می‌توانند نشانگر نقاط ضعف، اختلالات ریتمی، یا حملات قلبی باشند. پردازش سیگنال‌های قلبی به عنوان یکی از حوزه‌های مهم در پزشکی و علوم پزشکی، اهمیت و ضرورت بسیاری دارد. در زیر به برخی از این اهمیت‌ها و ضرورت‌ها اشاره می‌کنم: [۲]

۱-۵-۱- تشخیص بیماری‌های قلبی

پردازش سیگنال‌های قلبی به پزشکان امکان می‌دهد تا الگوهای ناهماهنگی و اختلالات در فعالیت الکتریکی قلب را تشخیص دهند. این الگوها می‌توانند نشانگر آریتمی‌ها، انسداد عروق و بیماری‌های قلبی دیگر باشند.

۲-۵-۱- پیش‌بینی خطرات قلبی

با استفاده از پردازش سیگنال‌های قلبی، می‌توان به تحلیل علائمی مانند تغییرات در فاصله بین نوسانات قلبی، شدت موج‌ها و تغییرات در نوسانات الکتریکی پرداخت تا خطرات مربوط به بیماری‌های قلبی را پیش‌بینی کرد.

۳-۵-۱- مانیتورینگ مرتب بیماران

پردازش سیگنال‌های قلبی می‌تواند در مانیتورینگ بیماران با بیماری‌های قلبی مفید باشد. این فرآیند می‌تواند به پزشکان اطلاعات در لحظه از فعالیت قلب بیماران را ارائه دهد و در صورت لزوم، اقدامات درمانی فوری را آغاز کند.

با توجه به این نکات، می‌توان نتیجه گرفت که پردازش سیگنال‌های قلبی نقش بسیار مهمی در تشخیص، پیش‌بینی، درمان و تحقیقات مرتبط با بیماری‌های قلبی دارد و از اهمیت و ضرورت بالایی برخوردار است.

۱–۶– پیشینه پژوهش

هوش مصنوعی و یادگیری ماشین در زمینه پردازش سیگنال‌های نوار قلب [۱]پیشرفت‌های قابل توجهی داشته‌اند و ابزارهای ارزشمندی را برای تشخیص و پایش بیماری‌های قلبی ارائه می‌دهند. تاریخچه توسعه هوش مصنوعی در تحلیل نوار قلب را می‌توان به صورت زیر بررسی کرد: [۲] [۳]

طبقه بندی سیگنال ECG بوسیله روش های یادگیری عمیق بر دیتاست PTB-XL: در این مقاله با استفاده از سه روش شبکه کانولوشنی، مدل SinceNet، ترکیب شبکه کانولوشنی با خصوصیات انتروپی به دسته بندی سیگنال ها در دو، پنج و بیست کلاس می‌پردازد که دقت نهایی بدست آمده این مدل ها در کلاس‌بندی پنج کلاسه در هر یک از این سه روش به ترتیب مساوی ۰.۷۲ ، ۰.۷۳ و ۰.۷۶۵ شده است.[۴]

تکنیک های یادگیری عمیق در طبقه بندی سیگنال های ECG با تشخیص پیک های R: این مطالعه کاربرد های یادگیری عمیق را در کلاس بندی سیگنال های ECG داده ست PTB-XL با تمرکز به استفاده از ویژگی های ظاهری سیگنال از جمله فاصله پیک های R بررسی می‌کند. بهترین دقت بدست آمده در مدل های مختلف تست شده توسط این مقاله در بازه ۷۴.۹-۷۹.۱ بوده است. [۵]

بررسی Few Shot learning برای طبقه بندی سیگنال های ECG در دیتاست PTB-XL: این پژوهش به دنبال یادگیری با نمونه های کم است که با استفاده از تعداد محدودی داده دقت قابل قبولی برای کلاس بندی سیگنال های نوار قلبی دیتاست PTB-XL بدست آورد و بدین منظور از ترکیب الگوریتم های کلاس بندی SVM و KNN با تکنیک های few shot learning به دقت های خوبی دست پیدا کرده است. از جمله در میان مدل ها مختلف تست شده بهترین دقت بدست آمده آنها ۷۷.۹-۸۰.۲ است. [۶]

۱–۷– فعالیت های انجام شده در این پایان‌نامه

در این پژوهش قصد داریم شبکه عصبی‌ای طراحی کنیم که سیگنال های نوار قلبی را در پنج کلاس طبقه بندی می‌کند. در فصل اول به مباحث آغازین و مقدمه بحث می‌پردازیم تا با مسئله اصلی که این پژوهش به آن می‌پردازد و سپس با اهداف و اهمیت این پژوهش آشنا شویم و در آخر به پیشینه این پژوهش اشاره می‌کنیم. در فصل دوم به توضیحات اولیه و بیان تعاریف و مفاهیم این حوزه پرداخته ایم و شبکه عصبی و طرز کار مدل سازی‌ها و اصطلاحات آنها را مطرح کردیم. در فصل سوم به دیتاست و بررسی های آماری آن

[۱] ECG (Electrocardiography)

می‌پردازیم. سپس در فصل چهارم به با تست و بررسی مدل هایی برپایه شبکه های LSTM و CNN بروی دیتاست XL_PTB و مقایسه دقت نهایی این مدل ها به مدلی بر پایه CNN یک بعدی با دقت مناسب دست پیدا کردیم که با دقت ۷۹.۲۴ به پیش‌بینی بیماری سابجکت در ۵ کلاس می‌پردازد. در نهایت در فصل آخر نتیجه گیری نهایی را بیان کرده و مطالب این پژوهش را جمع بندی می‌کنیم.

فصل ۲– مفاهیم وتعاریف

۲–۱– تعریف هوش مصنوعی وکاربرد آن در علم پزشکی

هوش مصنوعی [۱]یک حوزه چند رشته‌ای است که علوم کامپیوتر و داده هایی که در هر سیستمی وجود دارد را ترکیب می‌کند تا امکان حل مسائل فراهم شود. این شامل ایجاد سیستم‌هایی است که می‌توانند فرآیندهای فکری انسان را تقلید کرده و وظایفی را انجام دهند که به طور معمول نیاز به هوش انسانی دارند. هوش مصنوعی شامل زیر حوزه‌هایی مانند یادگیری ماشین است و الگوریتم‌هایی را شامل می‌شود که می‌توانند از داده‌ها یاد بگیرند و پیش‌بینی یا طبقه‌بندی‌هایی انجام دهند.

هوش مصنوعی می‌تواند به دو نوع تقسیم شود: [۷]

۱) هوش مصنوعی ضعیف، که به عنوان هوش مصنوعی محدود شناخته می‌شود و طراحی شده است تا وظایف خاصی را انجام دهد، مانند تشخیص بیماری، شناسایی صدا یا سیستم‌های پیشنهادی. این نوعی از هوش مصنوعی است که بیشتر مردم به طور روزانه با آن تعامل دارند، مانند Siri یا Alexa.

۲) هوش مصنوعی قوی یا AGI، که همان هوش مصنوعی عمومی است و به سیستم‌هایی اشاره دارد که دارای هوش و خردمندی معادل یا فراتر از هوش انسانی هستند که توانایی تفکر و حل مسئله در همه موضوعات را دارند.

در دهه اخیر هوش مصنوعی و شبکه های عصبی تحول زیادی کردند و کاربرد آنها در بسیاری از زمینه ها از جمله پزشکی افزایش یافته است. یادگیری ماشین یکی از شاخه های هوش مصنوعی است که به روش هایی گفته می‌شود که بدون نیاز به برنامه‌نویسی مستقیم برای هر وظیفه از داده‌ها یاد بگیرند و الگوهایی را شناسایی کنند. این الگوریتم‌ها الگوها را از داده‌های ورودی استخراج می‌کنند و با تجزیه و تحلیل دقیق این داده‌ها، مدل‌هایی را ایجاد می‌کنند که به ارائه پیش‌بینی‌ها یا تصمیم‌های خروجی کمک می‌کنند. دسته ای از الگوریتم های هوش مصنوعی را شبکه عصبی می‌نامند. این شبکه‌ها قادرند الگوهای پیچیده را از داده‌ها استخراج کنند و با تجزیه و تحلیل دقیق آنها، تصمیم‌گیری را انجام دهند.

هوش مصنوعی یکی از فناوری‌های بسیار چالش‌برانگیز در علم پزشکی بوده و تاثیرات گسترده‌ای در تشخیص، پیش‌بینی و درمان بیماری‌ها داشته است. برای مثال هوش مصنوعی می‌تواند از داده‌های پزشکی مانند نتایج آزمایش‌های خون، فشار خون و داده‌های الکترونیکی پرونده‌های بیماران استفاده کند تا به پیش‌بینی بیماری‌ها و تغییرات در وضعیت سلامتی بپردازد. یا با تجزیه و تحلیل داده‌های پزشکی بوسیله هوش مصنوعی، می‌توان برنامه‌های درمانی شخصی‌سازی شده از جمله درمان‌های دارویی، رژیم غذایی و

[۱] Artificial Intelligence (AI)

برنامه تمرین فیزیکی برای بیماران تهیه کرد. اگرچه این تنها چند نمونه از کاربردهای هوش مصـــنوعی در علم پزشـــکی هســـتند. با رشـــد و پیشـــرفت روزافزون در این حوزه، انتظار می‌رود که کاربردهای بیشـــتر و نوآورانه‌تری از این فناوری در بهبود سلامتی انسان‌ها به وجود آید. [۸]

۲-۲- نقاط ضعف هوش مصنوعی در پزشکی

هوش مصنوعی قابلیت انقلابی در بهبود دقت و سرعت تشخیص بیماری ها و بهبود آن ها دارد، اما همچنین با چندین چالش و نقطه ضعف همراه است. در ادامه به برخی از آنها اشاره شده است:

۲-۲-۱- تصمیم‌گیری با جعبه سیاه

زمانی که از الگوریتم های هوش مصـنوعی در مسـائل اسـتفاده می‌شـود، پیش‌بینی‌ها قابل مشـاهده و تفسـیر پذیر نیسـتند. به عنوان مثال، در همین تحقیق درآخر به مدلی نهایی دسـت پیدا می‌کنیم که با دقتی بالا به دسـت بندی بیماری بر اسـاس الگو های سـیگنال می‌پردازد ولی زمانی که سـیگنال را در یکی از دسـته‌بندی بیماری ها قرار می‌دهد نمی‌توان فهمید که برچه اساس این تصمیم گیری را انجام داده است. تصمیم‌گیری با جعبه سـیاه به معضـلی اشـاره دارد که در فناوری هوش مصـنوعی به وجود می‌آید، زمانی که پیش‌بینی‌ها یا تصـمیمات توسـط سـیسـتم‌های هوش مصـنوعی انجام می‌شـوند و فرآیند یا مکانیزم دقیقی برای توضـیح دلیل این تصمیمات در دسترس نیست. در مواردی که تصمیم‌گیری با جعبه سیاه صورت می‌گیرد، افراد نمی‌توانند به طور کامل بفهمند چرا یک سیسـتم هوش مصـنوعی یک تصـمیم خاص را اتخاذ کرده است. به عنوان مثال، یک الگوریتم پردازش تصـویر ممکن اسـت یک تصـویر از بیمار را بررسـی کرده و یک تشـخیص ارائه دهد، اما دلیل دقیقی که منجر به آن تصـمیم شـده اسـت، در دسـترس نیسـت. این می‌تواند منجر به ناامنی و عدم اطمینان در مورد صـحت و قابلیت اعتماد تصـمیمات سـیسـتم هوش مصـنوعی شـود، زیرا کاربران و ارائه دهندگان خدمات نمی‌توانند دقیقاً درک کنند که سیسـتم چگونه به تصمیمات خود رسیده است. این مشکل می‌تواند از موانع اصـلی در مقبولیت و گسـترش اسـتفاده از هوش مصـنوعی در برخی از زمینه‌ها باشـد ، به خصـوص زمینه‌هایی که امنیت، اعتماد و شـفافیت موارد حیاتی هسـتند. برای حل این چالش، نیاز به توسـعه روش‌هایی برای توضـیح و تفسـیر تصـمیمات گرفته شـده توسـط سـیسـتم‌های هوش مصـنوعی وجود دارد تا اعتماد و قابلیت اطمینان در این سیستم‌ها تقویت شود. [۹]

۲-۲-۲- پیچیدگی‌های آموزش

الگوریتم های هوش مصـنوعی به داده‌ی زیاد برای آموزش نیاز دارند بنابرین در مسـائلی که داده‌ی زیادی از مسئله موجود نیست یا جمع آوری داده سخت است، ممکن است لزوما کارآمد نباشند. همچین پردازش این

داده‌ی زیاد در مرحله یادگیری الگوریتم نیاز به سیستم های محاسباتی قوی دارد که باید مد نظر قرار گیرد. [۱۰]

۳-۲-۲- غیرقابل لمس و اعتماد بودن

هوش مصنوعی هنوز به طور کامل و به طور ایمن قادر به جایگزینی پزشکان انسانی در تصمیم‌گیری نیست. در برخی مواقع، یک پزشک ممکن است با دقت به ایمنی و راحتی بیمار توجه کرده و سپس تصمیم‌گیری کند ولی هوش مصنوعی های فعلی منطق بیشتری در نظر می‌گیرند که در بعضی مواقع ممکن است لزوما مفید نباشد. به عبارتی دیگر هنوز مراوده‌ی انسان با انسان مخصوصا در حوزه پزشکی، راحتتر و قابل اعتمادتر از انسان با هوش مصنوعی است. [۹]

۴-۲-۲- تغییر در توزیع داده و پیامدهای غیرمترقبه اش

الگوریتم های هوش مصنوعی می‌توانند به توزیع داده آموزش وابسته باشند. یعنی ممکن است اگر داده ای در مرحله تست به آن ها داده شود که حتی مشابه آن را در مرحله آموزش ندیده باشند، خروجی ناصحیح بدهند. به عنوان مثال، الگوهای بیماری ممکن است در طول زمان تغییر کنند، که منجر به اختلاف بین داده‌های آموزش و داده‌های تست شود که موجب پیامد های غیر قابل انتظار در خروجی این الگوریتم ها می‌شود. [۳]

این چالش‌ها نیاز به اجرای دقیق و نظارت مداوم بر سیستم‌های هوش مصنوعی در حوزه پزشکی را برای اطمینان از ایمنی بیمار و نتایج درمان مؤثر برجسته می‌کنند ولی ناگفته نماند که تحقیقات و بررسی های فراوانی در تمام جهان در این حوزه در حال انجام است و هیچ کدام از نقاط ضعف ذکر شده در بالا نتوانسته مانع غیرقابل حلی برای پیشرفت و همه گیری هوش مصنوعی در علم پزشکی شود.

یادگیری عمیق پس از یک دوره کاهش علاقه به تحقیقات هوش مصنوعی به دلیل محدودیت ها و دشواری های آن در دهه ۱۹۹۰ و اوایل دهه ۲۰۰۰، به عنوان یک پیشرفت قابل توجه در این زمینه ظهور کرد و به عنوان زیرمجموعه ای از یادگیری ماشین، نقش محوری در بازگشت هوش مصنوعی داشت که به توضیحات دقیق تر آن خواهیم پرداخت. [۱۱]

۲-۳- پردازش سیگنال پزشکی و اهمیت و کاربرد آن در بیماری های قلبی

پردازش سیگنال به معنای تبدیل یا تغییر داده ها به روشی است که به ما امکان می دهد چیزهایی را در آن ببینیم که با مشاهده مستقیم قابل مشاهده نیستند .پردازش سیگنال به مهندسان و دانشمندان این امکان را

می دهد که سیگنال ها، از جمله داده های علمی، داده های صوتی، تصاویر و ویدیوها را تجزیه و تحلیل و اصلاح کنند. [۱۲] [۱۳]

پردازش سیگنال در حوزه پزشکی یکی از بخش‌های بسیار مهم و حیاتی است که در تشخیص، پیش‌بینی و درمان بیماری‌ها اهمیت زیادی دارد. در این زمینه، سیگنال‌هایی مانند سیگنال‌های الکتروفیزیولوژیکی (مانند سیگنال‌های قلبی، مغزی و عضلانی)، تصویربرداری پزشکی (مانند سیگنال‌های MRI و CT) و دیگر انواع سیگنال‌های پزشکی مورد بررسی قرار می‌گیرند. پردازش سیگنال‌های قلبی به عنوان یکی از حوزه‌های مهم در پزشکی و علوم پزشکی، اهمیت و ضرورت بسیاری دارد.

از کاربرد های پردازش سیگنال پزشکی به خصوص در زمینه قلبی میتوان به موارد زیر اشاره کرد: [۱۴]

۱-۳-۲- تشخیص بیماری‌ها:

سیگنال‌های قلبی مانند ECG: در اینجا، الگوریتم‌ها و مدل‌های پردازش سیگنال برای تشخیص نوارهای قلبی استفاده می‌شوند تا نقاط ناهنجاری مانند آریتمی‌ها و حملات قلبی شناسایی شوند.

سیگنال‌های مغزی مانند EEG: در اینجا، الگوریتم‌ها به دنبال الگوهایی از سیگنال‌های مغزی می‌گردند که ممکن است به نشانه‌هایی از بیماری‌هایی مانند صرع، اختلالات خواب یا حتی بیماری‌های نورولوژیکی اشاره کنند.

سیگنال‌های عضلانی مانند EMG: در اینجا، الگوریتم‌ها و مدل‌های پردازش سیگنال برای تحلیل فعالیت عضلات و شناسایی اختلالات عضلانی مانند بیماری‌های عضلانی از قبیل ALS استفاده می‌شود.

۲-۳-۲- پیش‌بینی بیماری‌ها:

این بخش از پردازش سیگنال بر پایه تحلیل الگوها و سری های زمانی سیگنال‌های بیماری‌زا انجام می‌شود. مثلاً، با استفاده از الگوریتم‌های یادگیری ماشین، می‌توان الگوهایی از سیگنال‌های قلبی که پیش از حمله قلبی رخ می‌دهند را تشخیص داد و به پیش‌بینی حمله قلبی کمک کرد. همچنین با استفاده از پردازش سیگنال‌های قلبی، می‌توان به تحلیل علائمی مانند تغییرات در فاصله بین نوسانات قلبی، شدت موج‌ها و تغییرات در نوسانات الکتریکی پرداخت تا خطرات مربوط به بیماری‌های قلبی را پیش‌بینی کرد.

۳-۳-۲- رصد و نظارت بر وضعیت بیمار توسط سیستم های کامپیوتری:

در اینجا، سیگنال‌های مرتبط با وضعیت فیزیولوژیکی بیمار دائما در سیستم های کامپیوترتوسط الگوریتم های مختلف ازجمله هوش مصنوعی نظارت می‌شود. مثلاً در مواردی که بیمار نیاز به مانیتورینگ مداوم قلبی دارد، سیستم‌های پردازش سیگنال به صورت مداوم الگوهای نامطلوب را تشخیص داده و در صورت لزوم به پزشک اطلاع می‌دهند.

۴-۳-۲- طراحی دستگاه‌های پزشکی:

در این بخش، پردازش سیگنال به عنوان یکی از عناصر اساسی در طراحی دستگاه‌های پزشکی استفاده می‌شود. مثلاً در دستگاه‌های تصویربرداری مانند MRI و CT، سیگنال‌های دریافتی از بیمار باید به صورت دقیق و کامل پردازش شوند تا تصویری دقیق از وضعیت بدن بیمار ارائه شود.

۵-۳-۲- تصمیم‌گیری درمانی:

اطلاعات به دست آمده از پردازش سیگنال‌های قلبی می‌تواند به پزشکان کمک کند تا تصمیم‌های بهتری در مورد درمان بیماری‌های قلبی بگیرند. برای مثال، انتخاب نوع درمان مناسب برای آریتمی‌های قلبی بر اساس شدت و نوع آریتمی ممکن است توسط این اطلاعات تسهیل شود.

پس باتوجه با این کاربرد ها می‌توان فهمید به صورت کلی پردازش سیگنال در حوزه پزشکی نقش بسیار حیاتی و گسترده‌ای دارد و پردازش سیگنال‌های قلبی نقش مهمی در تشخیص، پیش‌بینی، درمان و تحقیقات مرتبط با بیماری‌های قلبی داراست و از اهمیت و ضرورت بالایی برخوردار است. حال به بررسی دقیق‌تر ساختار قلب و سیگنال های آن می‌پردازیم.

۲-۴- قلب و ساختار الکتریکی آن

قلب انسان عضوی چهار حفره‌ای است که وظیفه پمپاژ خون در سراسر بدن را بر عهده دارد. ساختار آن به گونه ای طراحی شده است که خون و اکسیژن را به طور مؤثر به بافت ها و اندام ها برساند. در اینجا اجزای اصلی ساختار قلب انسان آمده است: [۱۵] [۱۶]

حفره ها[۱]: قلب از چهار حفره تشکیل شده است که دو دهلیز (حفره های بالایی) و دو بطن (حفره های پایین) هستند. دهلیز راست خون بدون اکسیژن را از سیاهرگ های ریوی دریافت می کند و دهلیز چپ خون بدون اکسیژن را از سیاهرگ های بالایی و تحتانی دریافت می کند.

[۱] Chambers

دریچه ها[1]: در قلب چهار دریچه وجود دارد که جریان خون را تنظیم می کنند. دو مورد در خروجی هر حفره یافت می شود - دریچه سه لختی در بطن راست و دریچه ریوی در بطن راست. دریچه میترال در دهلیز چپ و دریچه آئورت در بطن چپ از برگشت جریان خون جلوگیری می کنند.

دیواره ها[2]: دیواره قلب از سه لایه تشکیل شده است: لایه داخلی (اندوکارد)، لایه میانی (میوکارد) و لایه بیرونی (اپیکارد). اندوکارد داخل حفره ها و دریچه های قلب را پوشانده است، میوکارد حاوی سلول های عضلانی است که برای پمپاژ خون منقبض می شوند و اپیکارد خارج قلب را می پوشاند.

سیستم اسکلتی[3]: قلب توسط پریکارد، یک کیسه دو لایه که قلب را احاطه و محافظت می‌کند، حمایت می شود. لایه بیرونی، به نام پریکارد فیبری، به دنده ها و جناغ سینه متصل می شود، در حالی که لایه داخلی، پریکارد احشایی، خود قلب را پوشانده است.

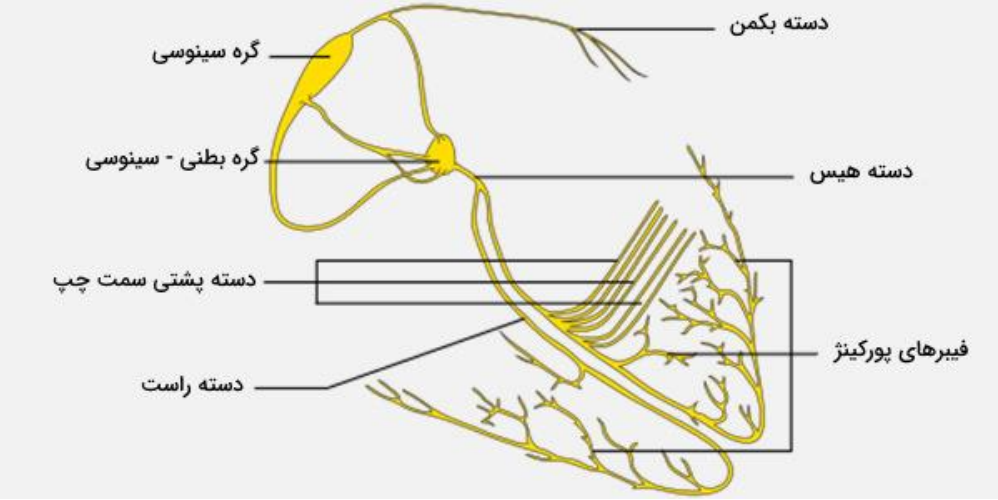
رگ های خونی[4]: قلب از طریق عروق کرونری که از آئورت منشعب می شوند، از بدن خون دریافت می کند. این عروق، خون غنی از اکسیژن را به عضله قلب می رسانند. گردش خون کرونری برای عملکرد قلب حیاتی است زیرا تضمین می کند که عضله قلب به اندازه کافی اکسیژن دریافت می کند.

موقعیت[5]: قلب بین دو ریه و کمی به سمت چپ مرکز، پشت جناغ سینه، روی دیافراگم که پرده بین قفسه سینه و حفره شکمی است، قرار دارد.

---

[1] Valves
[2] Walls
[3] Skeletal System
[4] Vessels
[5] Location

همچنین قلب برای کارکرد صحیح به سیستم الکتریکی منظم نیاز دارد. این سیستم هماهنگی انقباضات ماهیچه ای را ایجاد می کند که نیروی لازم برای حرکت خون را فراهم می کنند. دیاگرام این سیستم را در شکل زیر مشاهده می‌کنید:



شکل ۱-۲ اجزای سیستم هدایت الکتریکی قلب [۱۷]

اجزای اصلی سیستم هدایت الکتریکی قلب عبارتند از: [۱۸]

۱) **گره سینوسی**[۱]: این گره در دهلیز راست قرار دارد و مانند ضربان ساز طبیعی قلب عمل می کند. تولید سیگنال الکتریکی از این گره آغاز می‌شود که باعث انقباض حفره های بالایی قلب می‌شود.

۲) **گره دهلیزی بطنی**[۲]: این گره بین حفره های بالایی قرار دارد و سرعت انتقال سیگنال الکتریکی را کاهش می دهد تا بطن ها فرصت داشته باشند شل شوند و خون را پر کنند.

۳) **بسته هیس**[۳]: این مجموعه ای از سلول های هدایت کننده است که از گره AV به سمت مرکز قلب امتداد می یابد. بسته هیس سیگنال الکتریکی را به سمت پایین هدایت می کند و مانند رسانای الکتریکی، جریان الکتریکی را به عمق قلب هدایت می کند.

---

[۱] SA node
[۲] AV node
[۳] Bundle of His

۴) **شاخه های دسته هیس**[۱]: در قلب، دسته هیس به دو شاخه تقسیم می شود که سیگنال الکتریکی را به بطن ها منتقل می کنند.

۵) **الیاف پورکنژ**[۲]: این شبکه ای از سلول های هدایت کننده است که از شاخه های دسته هیس پخش می شود و ساختاری شبکه ای شبیه به چتر واژگون تشکیل می دهد. این الیاف سیگنال الکتریکی را در سراسر بطن ها پخش می کنند و باعث انقباض و خروج خون از قلب می شوند.

سیستم الکتریکی قلب بسیار هماهنگ است. سیگنال های الکتریکی از گره SA به گره AV و سپس به الیاف پورکنژ منتقل می شوند، که تضمین می کند انقباضات قلب به صورت متوالی رخ می دهد و به حرکت منظم خون کمک می کند.

۲–۵– سیگنال قلبی ECG (نوار قلب)

بیماری‌های قلبی یک طیف گسترده از اختلالاتی هستند که قلب و عروق را درگیر می‌کنند. تشخیص بیماری‌های قلبی معمولاً بر اساس تاریخچه بالینی بیمار، اعمال جسمانی، تست های فیزیکی و تست‌های تشخیصی صورت می‌گیرد. ECG یکی از معروف‌ترین و پرکاربردترین تست‌های قلبی است. این تست برای ثبت فعالیت الکتریکی قلب به کمک الکترودها انجام می‌شود. از آنجایی که منبع الکتریکی اصلی در بالاتنه انسان، سیستم الکتریکی درون قلب است که شرح داده شد، الکترودهایی که بر روی پوست بدن قرار می‌گیرند، الگوی نوسانات الکتریکی قلب را ثبت می‌کنند. الگوهای نوسانی ثبت شده توسط ECG می‌توانند نشانگر اختلالات ریتمی یا حملات قلبی باشند. [۱۹]

ثبت سیگنال قلبی ECG به منظور ضبط الکتریکی فعالیت قلب و الگوی نوسانات الکتریکی قلب به کمک الکترودها انجام می‌شود. زمانی که قلب انقباض می‌کند، جریان الکتریکی از یک نقطه به دیگری در قلب حرکت می‌کند، که این الگوهای الکتریکی در ECG ثبت می‌شوند و به عنوان نوار قلب قابل مشاهده هستند.

نوار قلب معمولاً از ۱۲ کانال ضبط می‌شود که هر یک نمایانگر فعالیت الکتریکی در یک منطقه خاص از قلب هستند. این کانال‌ها به صورت استاندارد در نوارهای ECG به شرح زیر هستند: [۱۹]

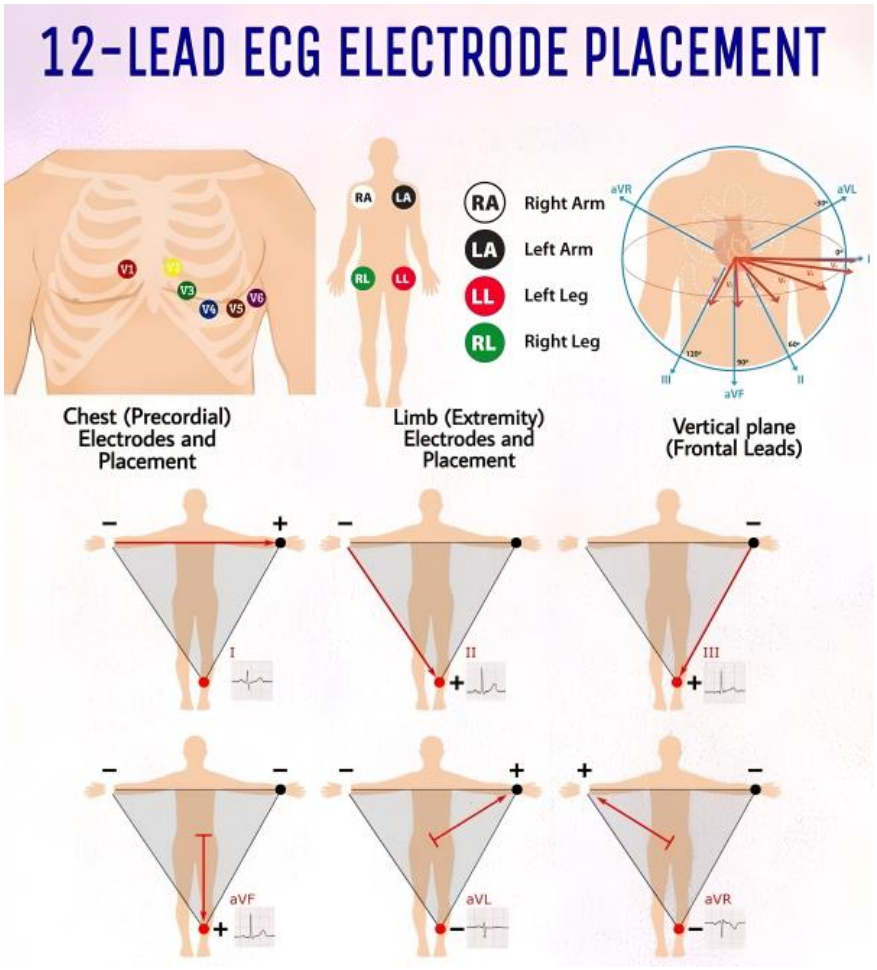
---

[۱] Bundle Branches

[۲] Purkinje Fibers

۱) **Lead I, II, III, aVR, aVL, aVF**: شش کانال اصلی که از الکترودها که به لید های اندامی[۱] شناخته می‌شوند.

۲) **V۱ تا V۶**: شش کانال دیگر که الکترودها روی قفسه [۲]سینه قرار دارند.



شکل ۲-۲ دوازده لید اصلی سیگنال نوار قلب [۲۰]

با تحلیل الگوها، فاصله بین موج‌ها و اندازه و شکل موج‌ها، پزشکان می‌توانند اطلاعات مهمی درباره عملکرد قلب، اختلالات ریتمی، آسیب‌های قلبی و اختلالات سایر اندام‌ها مانند الکترولیت‌ها و داروها را بدست آورند. این تست معمولاً در اورژانس‌ها، بخش‌های قلبی، مطب پزشکان و مراکز درمانی استفاده می‌شود و یکی از ابزارهای حیاتی برای تشخیص و درمان بیماری‌های قلبی است.

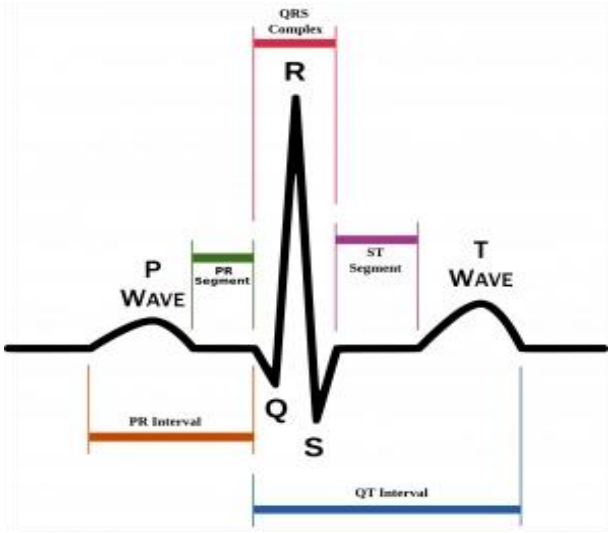
---

[۱] Limb leads

[۲] Chest leads

انقباضات قلب بوسیله الکتریسیته ناشی از سیستم الکتریکی قلب که توضیح داده شد، صورت می‌گیرد. این فعالیت الکتریکی همراه با انقباضات قلب، که باعث پمپاژ خون به سایر بخش‌های بدن می‌شود، اتفاق می‌افتد. ECG یکی از روش‌های مهم برای ثبت این فعالیت الکتریکی اســت. در این روش، الکترودها به دســتگاه بدن اتصال داده می‌شوند و الکتریسیته که توسط قلب تولید می‌شود، توسط این الکترودها ضبط می‌شود.

نوار قلب یک نمودار دوبعدی است که شدت الکتریسیته را برحسب زمان نشان میدهد. هر نوار ECG شامل چندین بخش اصلی است که در شکل زیر می‌بینید:



شکل۳-۲ پنج موج اصلی نوار قلب [۲۱]

موجP : موج P نشان دهنده دپولاریزاسیون دهلیزی است، یعنی زمانی که دهلیزها برای پمپاژ خون به داخل بطن ها منقبض می شوند.

کمپلکسQRS : این مجموعه ای از سه انحراف است که در یک نوار قلب معمولی دیده می شود. کمپلکس QRS با دپولاریزاسیون بطن های راست و چپ که همان انقباض بطن ها است، مطابقت دارد.

موجT : موج T نشان دهنده رپلاریزاسیون بطنی است، یعنی زمانی که بطن ها شل می شوند.

فاصلهPR : فاصله PR از ابتدای موج P شروع شده و تا انتهای کمپلکس QRS ادامه می یابد. این فاصله، زمانی را نشان می دهد که سیگنال الکتریکی که از گره سینوسی از طریق گره AV وارد بطن ها می شود، طی می‌کند.

قطعهST : قطعه ST از پایان کمپلکس QRS شروع شده و تا ابتدای موج T ادامه می یابد. این دوره زمانی را نشان می دهد که بطن ها دپولاریزه شده اند.

با تحلیل این نوار، پزشــکان می‌توانند اطلاعات مهمی راجع به فعالیت الکتریکی قلب، آریتمی‌ها، اختلالات ریتمی، عوارض قلبی و اختلالات مختلف دیگری که ممکن است بر روی سلامتی قلب تأثیر بگذارند، به دست

آورند. این اطلاعات برای تشخیص و درمان بیماری‌های قلبی بسیار ارزشمند هستند و ECG یکی از ابزارهای حیاتی برای مدیریت بیماران قلبی است.

۲-۶- تعریف شبکه های عصبی عمیق

شبکه‌های عصبی عمیق [۱]یک نوع از شبکه‌های عصبی مصنوعی هستند که از چندین لایه از نورون‌ها یا واحدهای پردازشی برای استخراج ویژگی‌های پیچیده و سطح بالا از داده‌های ورودی استفاده می‌کنند. این شبکه‌ها معمولاً شامل چندین لایه پنهان [۲]هستند که هر لایه به تدریج ویژگی‌های عمیق‌تر و جامع‌تری را از داده‌های ورودی استخراج می‌کند.

یک شبکه عصبی عمیق معمولاً شامل سه نوع لایه است:

۱) لایه ورودی[۳]:

این لایه وظیفه دریافت داده‌های ورودی را دارد. معمولاً هر نورون در این لایه به یک ویژگی یا ویژگی‌های خاص از داده‌های ورودی اختصاص دارد.

۲) لایه‌های پنهان [۴]:

این لایه‌ها بین لایه ورودی و خروجی قرار دارند و ویژگی‌های پیچیده‌تری را از داده‌های ورودی استخراج می‌کنند. هر لایه پنهان می‌تواند از چندین نورون تشکیل شده باشد و این نورون‌ها با استفاده از توابع فعال‌سازی مختلف و وزن‌های مختلف با یکدیگر ارتباط برقرار می‌کنند.

۳) لایه خروجی[۵]:

این لایه وظیفه تولید خروجی نهایی را بر اساس ویژگی های استخراج شده از لایه‌های پنهان دارد. معمولاً هر نورون در این لایه با یک کلاس مرتبط است که مورد پیش‌بینی یا تشخیص قرار می‌گیرد.

---

[۱] Deep Neural Networks یا DNNs

[۲] Hidden Layers

[۳] Input Layer

[۴] Hidden Layers

[۵] Output Layer

شبکه‌های عصبی عمیق به عنوان یکی از روش های کلیدی در حوزه هوش مصنوعی شناخته می‌شوند، زیرا این شـــبکه‌ها از توانایی یادگیری ویژگی ها و الگوهای داده‌های بزرگ و پیچیده بهره می‌برند. آن‌ها می‌توانند الگوهای پیچیده را در داده‌های ورودی شـــناســـایی کرده و ویژگی‌های اســـاســـی را اســـتخراج کنند، که این ویژگی‌ها می‌توانند برای انواع کاربرد ها از جمله پیش‌بینی، دسته‌بندی، تشخیص الگو، ترجمه و تولید متن و دیگر وظایف هوش مصنوعی بخصوص در حوزه پزشکی مورد استفاده قرار گیرند.

یکی از ویژگی‌های مهم شـــبکه‌های عصـــبی عمیق این اســـت که این شـــبکه‌ها در لایه های پنهان خود، کار اســـتخراج ویژگی ها را از داده ورودی انجام می‌دهند. این ویژگی‌ها به صـــورت خودکار از داده‌ها اســـتخراج می‌شـوند و معمولاً نیازی به تعریف دسـتی ویژگی‌ها نیسـت. به این ترتیب، شـبکه‌های عصـبی عمیق قادرند الگوهای پیچیده‌تر و ارتباطات عمیق‌تر در داده‌ها را تشـــخیص دهند که این امکان را به آن‌ها می‌دهد تا عملکرد بهتری در وظایف مختلفی از جمله پردازش و پیش‌بینی سری‌های زمانی ارائه دهند.

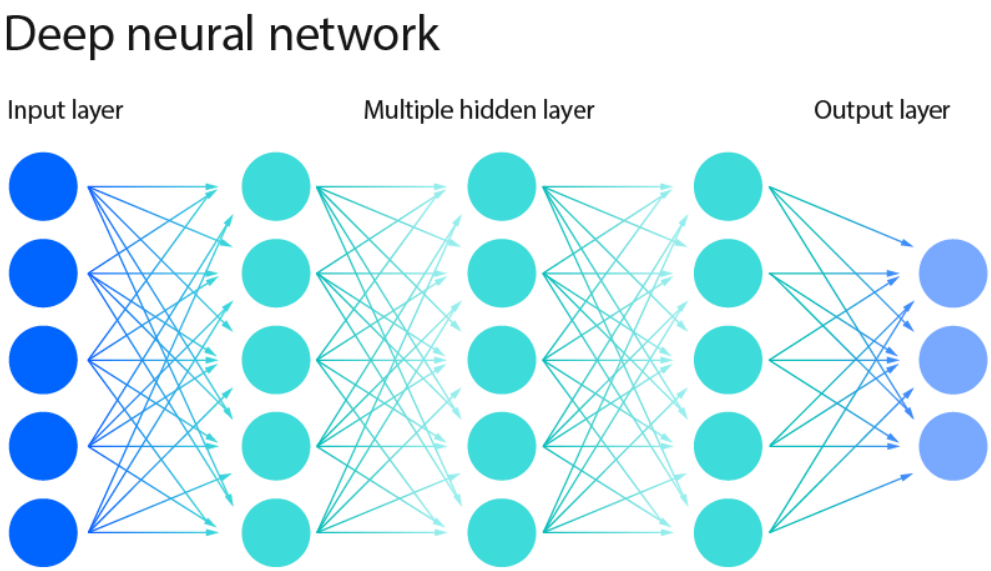
۲-۷- ساختار شبکه عصبی

۱-۷-۲- نرون ها[1]:

نرون‌های شبکه‌های عصبی، واحدهای اصلی ساختاری این شبکه‌ها هستند که از آنها برای پردازش اطلاعات اسـتفاده می‌شـود. علت نامگذاری اسـم آنها این اسـت که این نرون‌ها بر اسـاس مدل سـاختاری مغز انسـان طراحی شـده‌اند و از عملکرد سـلول‌های عصـبی بیولوژیکی الهام گرفته‌اند، اما با سـطحی از سـادگی برای استفاده در محاسبات رایانه‌ای بکار می‌روند.

[1] neurons

در شکل زیر ساختار یک شبکه عصبی را می‌بینید:



شکل ۴-۲ ساختار یک شبکه عصبی

هر نرون عصبی از چند قسمت اصلی تشکیل شده است:

۲-۷-۲- ورودی‌ها[۱]:

این بخش نرون برای دریافت اطلاعات ورودی از نرون‌ها یا سایر منابع استفاده می‌شود. این اطلاعات معمولاً به صورت عددی یا برداری است که وزن‌های متناظر با هر ورودی دارد.

۳-۷-۲- وزن‌ها[۲]:

وزن‌ها به اهمیت و تأثیر ورودی‌ها بر خروجی نرون اشاره دارند. هر وزن نشان دهنده ارتباطی است که نشان می‌دهد که چقدر ورودی مربوطه باید در تصمیم‌گیری نرون مورد توجه قرار گیرد. این وزن‌ها قبل از آموزش شبکه مقادیر تصادفی دارند و سپس در طی فرآیند آموزش تنظیم می‌شوند تا شبکه به درستی کار کند.

---

[۱] Inputs
[۲] Weights

۴-۷-۲- تابع فعال‌سازی:

این تابع به عنوان عاملی که تعیین می‌کند که آیا نرون باید فعال شـــود یا خیر، عمل می‌کند. این تابع معمولاً یک تابع غیرخطی اســـت که ارزش ورودی را تبدیل به مقدار خروجی مورد نظر می‌کند. به عبارت دیگر، اگر ورودی نرون به ازای یک مقدار خاصی از حدی بالاتر رود، نرون فعال می‌شود و خروجی غیر صفر تولید می‌کند. [۱]

تابع فعال‌ســازی یکی از عناصــر کلیدی در شــبکه‌های عصــبی اســت که وظیفه تبدیل ورودی‌های نورون به خروجی‌هایی با ویژگی‌های خاص دارد. این تابع به عنوان یک فیلتر غیرخطی عمل می‌کند که ارزش‌های ورودی را تبدیل به مقادیر خروجی مورد نظر می‌کند. از این تابع برای اضافه کردن پیچیدگی به مدل استفاده می‌شود و امکان یادگیری الگوهای پیچیده‌تر را ایجاد می‌کند.

حال توضیحی بیشتر در مورد تابع فعال‌سازی (ReLU) ارائه می‌دهم که در این تحقیق از آن استفاده شده: [۲]

۱) تابع فعال‌سازی ReLU:

ReLU یکی از محبوب‌ترین توابع فعال‌سازی در شبکه‌های عصبی عمیق است. ReLU مخفف " Rectified Linear Unit" اســت و توابع فعال‌ســازی خطی نامیده می‌شــوند که اگر مقدار ورودی به نرون مثبت باشــد، خروجی همان مقدار ورودی است؛ اما اگر مقدار ورودی منفی باشد، خروجی صفر است. به طور گسترده‌ای از این تابع به عنوان تابع فعال‌سازی در لایه‌های مختلف شبکه‌های عصبی استفاده می‌شود.

۲) ویژگی‌های تابع ReLU:

سادگی: ReLU یک تابع ساده در مقایسه با توابع فعال‌سازی غیرخطی دیگر است.

انتقال غیرخطی: با اینکه ReLU خطی در بخش مثبت اســت، اما در بخش منفی آن غیرخطی اســت، که به شبکه اجازه می‌دهد الگوهای پیچیده‌تری را یاد بگیرد.

جلوگیری از مشـــکل محو شـــدن گرادیان: از آنجا که ReLU برای مقادیر مثبت گرادیان یک اســـت، مشـــکل محو شـــدن گرادیان در زمان آموزش شـــبکه را کاهش می‌دهد و از ایجاد مشـــکل "مرگ نرون[۳]" جلوگیری می‌کند.

---

[۱] Activation Function
[۲] Activation Function
[۳] Dead Neuron

۳) محدودیت‌ها و مسائل مرتبط:

یکی از مشکلات ReLU ممکن است از فرآیندی به نام "مرگ نرون" باشد که به معنای این است که اگر وزن‌ها به گونه‌ای تنظیم شوند که همه ورودی‌ها منفی باشند، نرون دیگر خروجی مثبتی تولید نخواهد کرد و از این جایی که گرادیان منفی برای آموزش استفاده نمی‌شود، به سرعت غیرفعال می‌شود.

یک نسخه از ReLU با نام Leaky ReLU وجود دارد که برای رفع مشکل مرگ نرون ارائه شده است. این تابع فعال‌سازی یک شیب خطی بسیار کوچک به بخش منفی اضافه می‌کند تا این مشکل را حل کند.

با اینکه ReLU در اکثر موارد با موفقیت استفاده می‌شود، اما باید توجه داشت که در برخی موارد ممکن است بهتر باشد از توابع فعال‌سازی دیگری مانند sigmoid یا tanh استفاده کنید، به ویژه در مسائلی که مقادیر خروجی بین -۱ و ۱ یا ۰ و ۱ نیاز دارند.

۵-۷-۲- خروجی[۱]:

این بخش نرون مسئول تولید خروجی است. این خروجی ممکن است به عنوان خروجی نهایی شبکه یا به عنوان ورودی به نرون‌های لایه‌های بعدی در معماری چند لایه شبکه عمل کند.

نرون‌ها معمولاً به صورت لایه‌های متعدد در شبکه‌های عصبی عمیق سازماندهی می‌شوند. هر لایه شامل تعدادی نرون است که اطلاعات را از لایه قبلی دریافت کرده و آن را پردازش می‌کنند. این فرآیند ادامه دارد تا خروجی شبکه تولید شود. با استفاده از تعداد زیادی از این نرون‌ها و لایه‌ها، شبکه‌های عصبی عمیق قادر به یادگیری و تفسیر الگوهای پیچیده در داده‌ها هستند و در بسیاری از حوزه‌های کاربردی از جمله تصویربرداری، صوت‌شناسی و پردازش زبان طبیعی کارآیی بالایی دارند.

۶-۷-۲- تابع هزینه[۲]:

تابع هزینه یا تابع خطا معیاری است که برای ارزیابی عملکرد شبکه‌های عصبی استفاده می‌شود. این تابع میزان اختلاف بین خروجی‌های تولید شده توسط شبکه و مقادیر واقعی مورد انتظار را اندازه‌گیری می‌کند.

---

[۱] Output

[۲] Cost function

هدف اصـــلی این اســـت که با کمینه کردن مقدار تابع هزینه، شـــبکه به تولید خروجی‌هایی که به داده‌های واقعی نزدیک‌تر باشـند، هدایت شـود. هر تابع هزینه متناسـب با نوع مسـئله‌ای که داریم و داده‌هایی که با آن کار می‌کنیم، انتخاب می‌شود. انتخاب صحیح تابع هزینه می‌تواند به بهبود کارایی و دقت شبکه کمک زیادی کند. در این پژوهش از تابع هزینه Binary Cross Entropy استفاده کرده ایم. این تابع معیاری برای سنجش عدم تطابق بین دو توزیع احتمال اســـت. این معیار به طور گســـترده در یادگیری ماشـــین، به ویژه در طبقه بندی دودویی، برای ارزیابی عملکرد مدل و به عنوان cost Function در آموزش مدل ها استفاده می شود.

۷-۷-۲- بهینه سازی[۱]:

بهینه‌ساز [۲]در شبکه‌های عصبی مسئول بهبود وزن‌های شبکه و کمینه کردن تابع هزینه است. هدف اصلی بهینه‌ساز، یافتن مقادیر بهینه برای وزن‌ها به نحوی است که تابع هزینه کمینه شود و عملکرد شبکه بهبود یابد. بهینه‌ســازها از مفهوم گرادیان نزولی (Gradient Descent) برای به روزرســانی وزن‌ها با توجه به مقدار گرادیان تابع هزینه استفاده می‌کنند. گرادیان نزولی یکی از ساده‌ترین الگوریتم‌های بهینه‌سازی است که بر اساس مفهوم گرادیان تابع هزینه نسبت به وزن‌ها عمل می‌کند و پایه بسیاری از الگوریتم های بهینه‌سازی پیچیده تر است.در این الگوریتم، در هر مرحله، مقدار گرادیان تابع هزینه نسبت به وزن‌ها محاسبه شده و وزن‌ها به سمت مخالف جهت گرادیان (جهت کاهش تابع هزینه) به روزرسانی می‌شوند.

هر بهینه‌ساز ممکن است برای مسائل و داده‌های مختلف بهترین عملکرد را داشته باشد. انتخاب صحیح بهینه‌ساز و تنظیم پارامترهای آن می‌تواند به بهبود عملکرد و دقت شبکه عصبی کمک زیادی کند. در این پژوهش از بهینه ساز Adam استفاده کردم که پرامتر های آن را برحسب نیاز مسئله تنظیم نمودم. Adam از بهینه سازهای مبتنی بر مومنتوم است. این بهینه‌ســازها از مفهوم مومنتوم برای افزایش سرعت آموزش و جلوگیری از گیر کردن در نقاط مینیمم محلی استفاده می‌کنند.

۸-۷-۲- بازگشت به عقب:

Backpropagation یا به فارسی "پس‌انتشار"، یک الگوریتم مهم در آموزش شبکه‌های عصبی است که برای به روزرسانی وزن‌های شبکه با استفاده از گرادیان تابع هزینه نسبت به وزن‌ها استفاده می‌شود. این الگوریتم

---

[۱] Optimization

[۲] Optimizer

به شبکه‌ها کمک می‌کند تا خطا را از طریق لایه‌ها به عقب منتقل کنند و سپس گرادیان‌های ضروری برای به‌روزرسانی وزن‌های هر نورون محاسبه کنند.

مراحل آموزش شبکه که Backpropagation ختم می‌شود به شرح زیر است:

۱) انتشار جلو [۱]:

در این مرحله، ورودی شبکه از طریق لایه‌های مختلف به جلو منتقل می‌شود و خروجی تولید می‌شود. این عملیات شامل محاسبه خروجی‌های هر لایه بر اساس وزن‌ها و توابع فعال‌سازی مربوطه است.

۲) محاسبه خطا [۲]:

در این مرحله، خطای شبکه بر اساس خروجی تولید شده و مقادیر واقعی داده‌ها محاسبه می‌شود. این تفاوت بین خروجی واقعی و پیش‌بینی شده توسط شبکه استفاده می‌شود.

۳) پس‌انتشار خطا [۳]:

این مرحله مهمترین قسمت الگوریتم Backpropagation است. در این مرحله، خطا از لایه خروجی به سمت عقب به لایه‌های قبلی انتقال می‌یابد و گرادیان‌های مربوط به تابع هزینه نسبت به وزن‌ها محاسبه می‌شود.

۴) به‌روزرسانی وزن‌ها [۴]:

در این مرحله، وزن‌های شبکه بر اساس گرادیان‌های محاسبه شده به‌روزرسانی می‌شوند. این به‌روزرسانی معمولاً با استفاده از یک الگوریتم بهینه‌سازی مانند گرادیان نزولی انجام می‌شود.

---

۱ Forward Propagation
۲ Compute Loss
۳ Backpropagation
۴ Update Weights

الگوریتم Backpropagation مبتنی بر قانون زنجیره‌ای در محاسبات گرادیان است که امکان محاسبه گرادیان تابع هزینه نسبت به وزن‌ها را فراهم می‌کند. این الگوریتم به شبکه‌های عصبی امکان می‌دهد تا از طریق به‌روزرسانی وزن‌ها به طور تدریجی و با توجه به اطلاعات بهتری که از داده‌ها به دست می‌آورند، بهبود یابند و یاد بگیرند.

۲-۸- پیش پردازش داده ها

پیش‌پردازش سیگنال ECG (نوار قلب) یک مرحله اساسی و حیاتی در تحلیل سیگنال‌های قلبی با استفاده از هوش مصنوعی و شبکه‌های عصبی است. پیش‌پردازش سیگنال ECG یک فرآیند کلی است که به منظور بهبود کیفیت و قابلیت تحلیل سیگنال‌های قلبی انجام می‌شود. سیگنال ECG یک سیگنال الکتریکی است که فعالیت‌های الکتریکی قلب را نشان می‌دهد. در حین تحلیل ECG، مسائلی مانند نویز و تغییرات بیولوژیکی در طول زمان می‌توانند تحلیل دقیق سیگنال را مشکل کنند. برای مقابله با این چالش‌ها، انجام پیش‌پردازش می‌تواند اطلاعات مفیدی را از سیگنال استخراج کند.

در ابتدا، سیگنال ECG از لایه‌های پوستی بدن با استفاده از الکترودها گرفته می‌شود. سپس، مراحل پیش‌پردازش شامل چندین مرحله می‌شود:

۱) فیلترهای پایین‌گذر

برای حذف فرکانس‌های بالا و نویزهای مختلف، اعمال می‌شوند. این فیلترها کمک می‌کنند تا سیگنال با فرکانس مناسب برای تحلیل باقی بماند.

۲) تقویت سیگنال

ممکن است نسبت سیگنال به نویز ضعیف باشد، بنابراین از تقویت کننده‌ها برای تقویت سیگنال استفاده می‌شود تا اطلاعات قابل تحلیل تر فراهم شود.

۳) حذف نویز

فرآیند حذف نویز با استفاده از تکنیک‌های متنوعی انجام می‌شود، از جمله فیلترهای متوسطه، حذف افتراق قلبی و مهندسی ویژگی‌ها برای حذف نویزهای مختلف.

۴) نرمال‌سازی

اطلاعات سیگنال ECG باید به‌طور متناسب و قابل مقایسه با یکدیگر باشند. نرمال‌سازی باعث می‌شود که واحدهای مختلف سیگنال یکسان باشند و برای آموزش مدل‌های مبتنی بر هوش مصنوعی ایده‌آل باشد.

۵) استخراج ویژگی

در این مرحله اطلاعات مهمی مانند فرکانس‌ها، زمان‌های بروز ویژگی هایی مهم و الگوهای خاص از سیگنال استخراج می‌شود. این ویژگی‌ها به عنوان ورودی‌های مهم برای مدل‌های هوش مصنوعی در مراحل مدل سازی و اجرای الگوریتم استفاده می‌شوند.

دراین پژوهش، برروی داده‌ی دیتاست PTB-XL ، بسیاری از پیش پردازش ها از جمله فیتلر های نویز اعمال شده بود ولی برخی دیگر مانند نرمال سازی لازم بود که در کدنویسی مدل اعمال شده است که توضیحات دقیق آن را در فصول بعد خواهید دید.

فصل ۳– دیتاست پژوهش و بررسی های آماری آن

۳–۱– دیتاست مورد استفاده

مجموعه داده ECG PTB-XL یک مجموعه داده بزرگ از ۲۱۷۹۹ از ۱۲ کلنال ECG بالینی از ۱۸۸۶۹ بیمار با طول سیگنال ۱۰ ثانیه است. داده‌های مجموعه داده ECG PTB-XL با دستگاه‌های شرکت Schiller AG در طی حدود هفت سال از اکتبر ۱۹۸۹ تا ژوئن ۱۹۹۶ جمع‌آوری شد. با خرید پایگاه داده اصلی از شرکت Schiller AG، کلیه حقوق استفاده به PTB انتقال یافت. این پایگاه داده در تعدادی از مقالات استفاده شد، اما دسترسی تا زمان انتشار عمومی در سال ۲۰۱۹ محدود باقی ماند. برای دسترسی به فایل های این دیتاست از آدرس زیر می‌توانید استفاده کنید:

https://physionet.org/content/ptb-xl/۱.۰.۳

ECGها و بیماران با شناسه‌های منحصر به فرد (ecg_id و patient_id) شناسایی می‌شوند. اطلاعات شخصی در جدول فراداده‌ها، مانند نام‌های کاردیولوژیست‌های اعتبارسنج، پرستاران و محل ضبط (بیمارستان و غیره) ضبط شده‌اند. لیبل‌های ECG استفاده شده برای برچسب‌گذاری رکوردها به استاندارد SCP-ECG پیروی می‌کنند.

ساختار داده در پوشه ها به شکل زیر است:

```
ptbxl
├── ptbxl_database.csv
├── scp_statements.csv
├── records100
│   ├── 00000
│   │   ├── 00001_lr.dat
│   │   ├── 00001_lr.hea
│   │   ├── ...
│   │   ├── 00999_lr.dat
│   │   └── 00999_lr.hea
│   ├── ...
│   └── 21000
│       ├── 21001_lr.dat
│       ├── 21001_lr.hea
│       ├── ...
│       ├── 21837_lr.dat
│       └── 21837_lr.hea
└── records500
    ├── 00000
    │   ├── 00001_hr.dat
    │   ├── 00001_hr.hea
    │   ├── ...
    │   ├── 00999_hr.dat
    │   └── 00999_hr.hea
    ├── ...
    └── 21000
```

```
├── 21001_hr.dat
├── 21001_hr.hea
├── ...
├── 21837_hr.dat
└── 21837_hr.hea
```

این مجموعه داده شامل ۲۱۷۹۹ سابقه ECG بالینی با ۱۲-کانال با طول ۱۰ ثانیه است، که ۵۲٪ مرد و ۴۸٪ زن هستند و سن آن‌ها کل محدوده از ۰ تا ۹۵ سال (میانگین ۶۲) را پوشش می‌دهد. ارزش این مجموعه داده از مجموعه جامع بسیاری از پاتولوژی‌های مختلف هم‌زمان است، اما همچنین از نمونه‌های سالم نیز به نسبت بزرگی برخوردار است. در جدول زیر می‌توانید نام ۵ کلاس تخصیص یافته سیگنال ها را مشاهده کنید که بعد از آن توضیحات هرکدام آمده است:

جدول۱-۳ کلاس بندی لیبل های داده

#Records	Superclass	Description
۹۵۱۴	NORM	Normal ECG
۵۴۶۹	MI	Myocardial Infarction
۵۲۳۵	STTC	ST/T Change
۴۸۹۸	CD	Conduction Disturbance
۲۶۴۹	HYP	Hypertrophy

فایل‌های سیگنال نوار قلب به فرمت پایگاه داده (WFDB) که فرمت استاندارد ذخیره داده ECG است، با دقت ۱۶ بیت با وضوح ۱ میکروولت و فرکانس نمونه‌برداری ۵۰۰ Hz در پوشه records۵۰۰ ذخیره می‌شوند. برای راحتی کاربران، همچنین نسخه‌های کاهش‌یافته از داده‌های موج‌شکل با فرکانس نمونه‌برداری ۱۰۰ Hz در پوشه records۱۰۰ منتشر شده است.

جدول فراداده‌ها که شامل اطلاعات جدولی سابجکت ها مانند سن، جنسیت، قد و وزن آنها است در ptbxl_database.csv ذخیره شده که یک ستون برای هر رکورد با ecg_id شناسه بخصوص هر سابجکت را مشخص می‌کند. همچنین دو ردیف از این جدول با نام های filename_hr و filename_lr حاوی نام فایل های هر سیگنال هستند و ستون scp_codes لیبل بیماری هر سابجکت را مشخص می‌کند. جلوتر در بخش مدل سازی از داده‌ی این جدول با نام داده جدولی نام می‌بریم.

۳-۲- توضیح کلاس بندی لیبل های دیتاست

۳-۲-۱- ECG نرمال[۱]:

یک ECG نرمال فعالیت الکتریکی قلب را نشـــان می‌دهد هنگامی که قلب در محدوده‌های ســـلامتی فعالیت می‌کند. ویژگی های این سیگنال در بخش ها قبل توضیح داده شده است.

۳-۲-۲- سکته قلبی[۲]: [۲۲]

جریان خون به یک بخش از عضله قلب به مدت طولانی مسدود می‌شود و باعث آسیب یا مرگ آن قسمت از قلب می‌شود که به عنوان حمله قلبی معروف است. در یک ECG، نشانه‌های حمله قلبی عبارتند از:

افزایش ST-سگمنت: ST سگمنت‌های بلند نشانه آسیب حاد به عضله قلب هستند، که به طور معمول از یک حمله قلبی است.

موج Q غیرطبیعی: این نوع موج به عنوان Q نامطلوب، طولانی تر و بلند تر از حالت عادی ظاهر می‌شـــوند و نشان دهنده منطقه بافت مرده است.

۳-۲-۳- تغییرات ST/T[۳]: [۲۲]

تغییرات ST-T در یک ECG می‌توانند نشـان دهنده نقص‌های قلبی مختلف باشـند، از جمله ایسـکمی قلبی و آسیب بازسازی[۴]. این تغییرات ممکن است شامل موارد زیر باشد:

افزایش ST-سگمنت: این به نشانه آسیب حاد عضله قلبی است، احتمالاً ناشی از حمله قلبی.

کاهش ST-سگمنت: اغلب نشانه ایسکمی قلبی است، جایی که جریان خون به عضله قلب کاهش یافته است.

برعکس شدن موج T: ممکن است نشانه ایسکمی قلبی باشد، اما همچنین می‌تواند در آسیب عضله قلبی یا نقص‌های بازسازی دیده شود.

---

۱ Normal ECG
۲ Myocardial Infarction
۳ ST/T Change
۴ Cardiac regeneration defects

۴-۲-۳- اختلال هادی[1]: [۲۲]

اختلالات هادی در قلب می‌تواند توالی و زمان‌بندی نرمال انتقال سیگنال‌های الکتریکی را تحت تأثیر قرار دهد و منجر به یافته‌های ECG غیرطبیعی شود. برخی از اختلالات هادی شایع شامل موارد زیر هستند:

مسدودیت شاخه باندل: تأخیر یا مسدود شدن در انتقال سیگنال‌ها از طریق شاخه‌های چپ یا راست.

مسدودیت هادی دهلیزی-بطنی: اختلال در انتقال بین دهلیزها و بطن‌ها.

سندرم وولف-پارکینسون-وایت: یک مسیر الکتریکی اضافی بین دهلیزها و بطن‌ها که منجر به موج دلتا مشخص در ECG می‌شود.

۵-۲-۳- هیپرتروفی[2]: [۲۲]

هیپرتروفی قلبی به بزرگ شدن عضله قلب اشاره دارد، اغلب در پاسخ به فشار کاری افزایش یافته رخ می‌دهد. در یک ECG، نشانه‌های هیپرتروفی شامل موارد زیر هستند:

ولتاژ افزایش یافته: بالاتر از طبیعی بودن اندازه کمپلکس QRS، نشان دهنده ضخامت بیشتر عضله قلب است.

هیپرتروفی بطن چپ (LVH): اغلب به عنوان افزایش ولتاژ در لیدهای سمت چپ (I، aVL، V۵، V۶) مشخص می‌شود.

هیپرتروفی بطن راست (RVH): با Rهای بلند در لیدهای سمت راست (V۱، V۲) مشخص می‌شود.

۳-۳- پیاده سازی اولیه و بررسی های آماری

برای مشاهده و دسترسی به کد پایتون پیاده سازی و سایر داکیومنت ها میتوانید به گیت هاب من به آدرس https://github.com/sajjadrezvani/ECG_Ai مراجعه فرمایید. همانطور که اشاره شد برای پیاده سازی این مسئله از کتابخانه Keras در پایتون استفاده شده است.

حال در این بخش به توضیح بعضی از قسمت های مهم کد و نکات عملی می‌پردازیم تا در قسمت بعد وارد اصل جزئیات مدل ها شویم. دقت کنید که در این قسمت هدف مدل سازی و طراحی مدل شبکه عصبی

---

[1] Conduction Disturbance

[2] Hypertrophy

نیست. بلکه با رسم داده ها و بررسی های آماری آن دیدی بهتر از داده بدست می‌آوریم که در بخش بعد از آن استفاده می‌کنیم.

ابتدا برای اجرای صحیح کد باید تمام کتابخانه های ذیل را نصب داشته باشید:

```
import tensorflow as tf
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import wfdb # WaveForm DataBase software package, provides tools for working with biomedical time-series data.
import os
import ast
from ecgdetectors import Detectors
import numpy as np
import matplotlib.pyplot as plt
import pywt  # Import the PyWavelets library
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, LSTM, Dropout, BatchNormalization
from tensorflow.keras.models import Sequential
from tensorflow.keras import regularizers
from tensorflow.keras import optimizers
from sklearn.preprocessing import StandardScaler
import joblib
```

سپس در اینجا مسیر پوشه اصلی را مشخص و دیتاست ptbxl_database.csv را که حاوی داده های جدولی است می‌خوانیم. توجه کنید که ستون های مورد نیاز خود برا ورودی دادن به شبکه عصبی اینجا تعیین می‌کنیم. سپس طول داده که تعداد سابجکت هاست را خروجی که می‌گیریم که عدد ۲۱۷۹۹ نمایش داده می‌شود. در آخر برای افزایش سرعت مدل و محدودیت های رم سیستم ۵۰۰۰ داده را جدا می‌کنیم تا به در ادامه از آن استفاده کنیم:

```
root_path = './ptb-xl-dataset-1.0.3/'
ecg_data = pd.read_csv('./ptb-xl-dataset-1.0.3/ptbxl_database.csv'\
    , usecols= ['sex', 'height', 'weight', 'strat_fold','scp_codes', 'infarction_stadium1' \
    , 'infarction_stadium2', 'heart_axis' , 'pacemaker' ,'filename_lr', 'filename_hr'])
print( len(ecg_data) )
# ecg_data = ecg_data.sort_values(by='strat_fold')
ecg_data = ecg_data[:5000]
```

نمایی از چند سطر اول داده‌ی جدولی دیتاست ecg_data را در اینجا می‌بینید:

جدول ۲_۳ نمایی از چند سطر اول دیتاست *ecg_data*

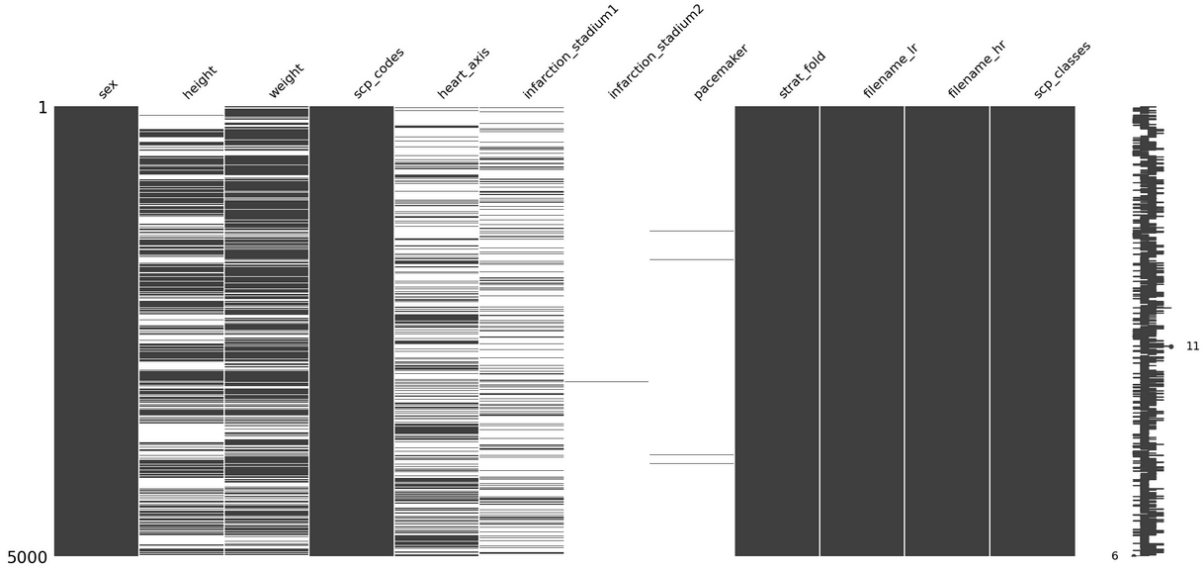
	sex	height	weight	scp_codes	heart_axis	infarction_stadium1	infarction_stadium2	pacemaker	strat_fold	filename_lr	filename_hr
0	1	NaN	63.0	{'NORM': 100.0, 'LVOLT': 0.0, 'SR': 0.0}	NaN	NaN	NaN	NaN	3	records100/00000/00001_lr	records500/00000/00001_hr
1	0	NaN	70.0	{'NORM': 80.0, 'SBRAD': 0.0}	NaN	NaN	NaN	NaN	2	records100/00000/00002_lr	records500/00000/00002_hr
2	1	NaN	69.0	{'NORM': 100.0, 'SR': 0.0}	NaN	NaN	NaN	NaN	5	records100/00000/00003_lr	records500/00000/00003_hr
3	0	NaN	82.0	{'NORM': 100.0, 'SR': 0.0}	NaN	NaN	NaN	NaN	3	records100/00000/00004_lr	records500/00000/00004_hr
4	1	NaN	70.0	{'NORM': 100.0, 'SR': 0.0}	NaN	NaN	NaN	NaN	4	records100/00000/00005_lr	records500/00000/00005_hr
...	...	...	...	...	...	...	...	...	...	...	...

حال با این دو خط زیر ابتدا دیتاست لیبل هاست را می‌خوانیم و سپس با توجه به ستون نام فایل های سیگنال که از دیتاست اولیه به دست آوردیم با استفاده از فانکشن wfdb.rdsamp فایل ها سیگنال های ECG را می‌خوانیم و در ecg_signal می‌ریزیم:

```
scp_data = pd.read_csv(root_path + 'scp_statements.csv' , index_col= 0)#, usecols= ['diagnostic','diagnostic_class']  )
ecg_signal = np.array([wfdb.rdsamp(os.path.join(root_path, file))[0] for file in ecg_data.filename_lr])
```

با استفاده از Msno میتوانیم به زیبایی نمودار وجود یا عدم وجود داده‌ی جدولی خود به دست آوریم:

```
1 import missingno as msno
2 msno.matrix(ecg_data)
3 plt.show()
```

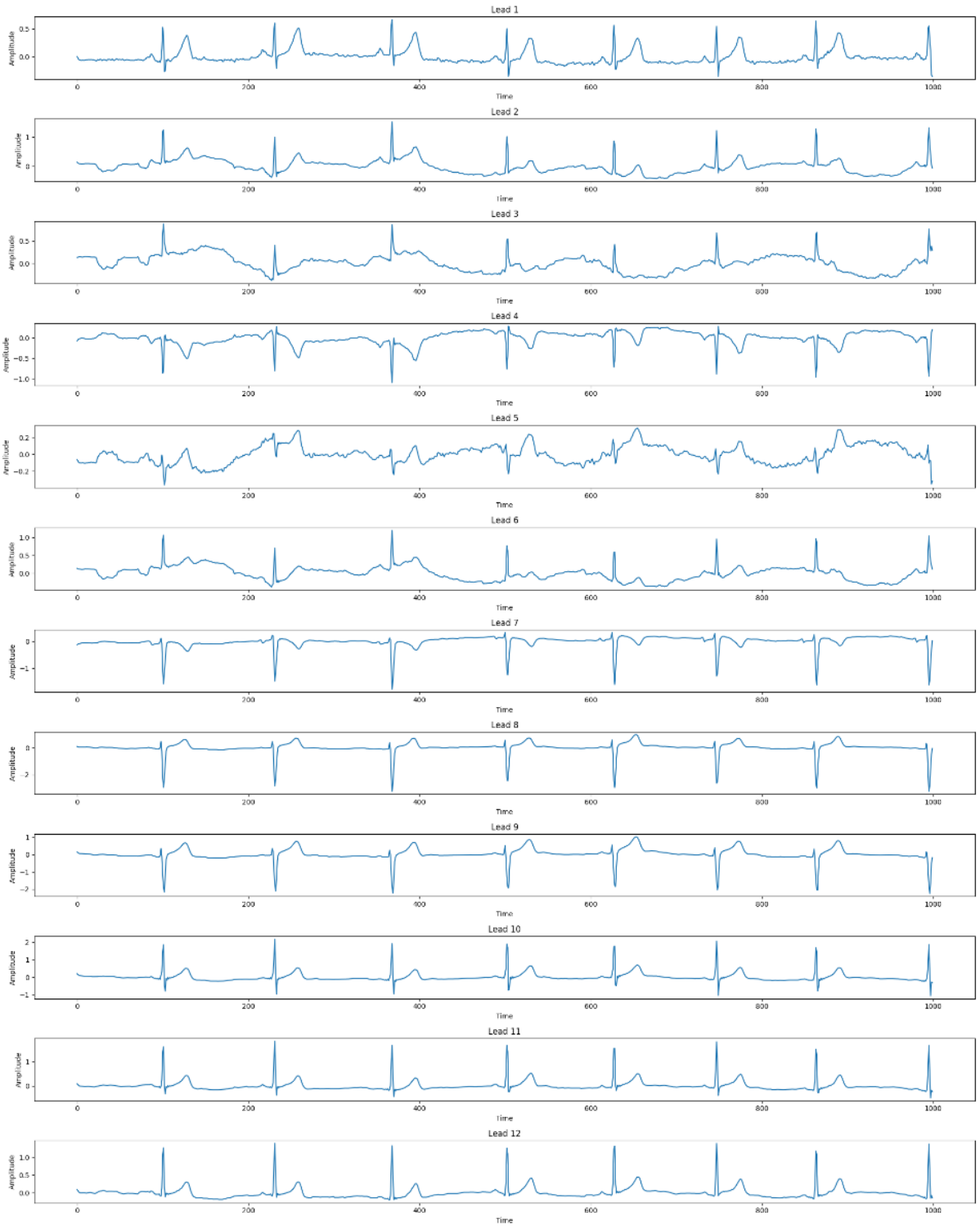


شکل ۵-۳ شکل وجود یا عدم وجود داده‌ی جدولی

حال با استفاده از کد زیر برای مشاهده کلی سیگنال نوار قلبی سابجکت اول را با ۱۲ کانال آن رسم می‌کنیم:

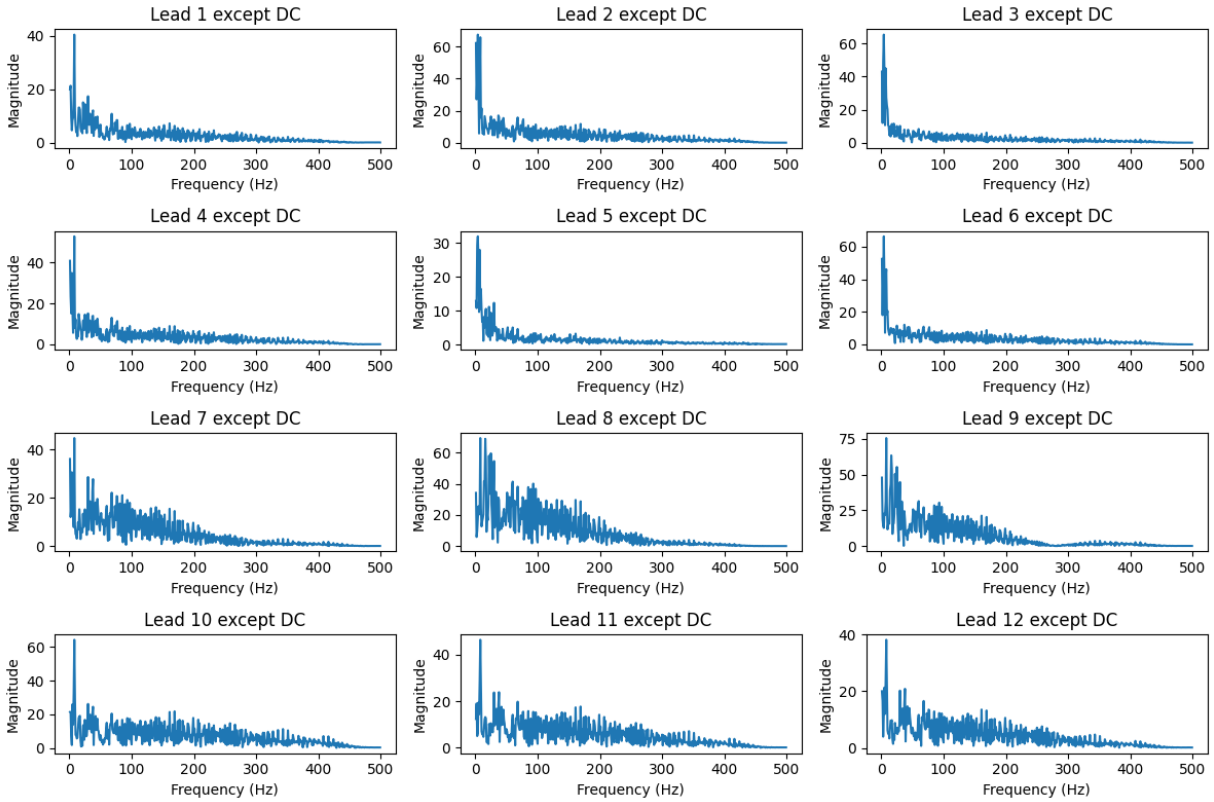
```
# Generate a time array (assuming a sampling rate of 1 Hz for simplicity)
time = np.arange(0, len(sample), 1)

# Plot each lead in a single column
plt.figure(figsize=(20, 25))

for lead in range(12):
    plt.subplot(12, 1, lead + 1)
    plt.plot(time, sample[:, lead])
    plt.title(f'Lead {lead + 1}')
    plt.xlabel('Time')
    plt.ylabel('Amplitude')

plt.tight_layout()
plt.show()
```

همانطور که می‌بینید پیک های pqrst که در قسمت تعاریف و مفاهیم سیگنال های قلب نشان و توضیح داده شدند در اینجا قابل مشاهده هستند:



شکل ۶-۳ سیگنال نوار قلبی سابجکت اول را با ۱۲ کانال آن

همچنین برای کسب اطلاعات بیشتر از سیگنال، تبدیل فوریه ۱۲ کانال سابجکت اول را نیز رسم کردیم:



شکل۷-۳ تبدیل فوریه ۱۲ کانال سابجکت اول

سپس ۵ فرکانس غالب هر ۱۲ کانال را که بیشترین مقدار را دارند بدست آوردیم:

Top frequencies in Lead ۱: [۸. ۲. ۱.۳۰. ۷.] Hz

Top frequencies in Lead ۲: [۴. ۸. ۱. ۳. ۷.] Hz

Top frequencies in Lead ۳: [۴. ۳. ۷. ۵. ۱.] Hz

Top frequencies in Lead ۴: [۸. ۱. ۴. ۲. ۷.] Hz

Top frequencies in Lead ۵: [۴. ۳. ۷. ۵. ۹.] Hz

Top frequencies in Lead ۶: [۴. ۱. ۸. ۳. ۵.] Hz

Top frequencies in Lead ۷: [۸. ۱. ۴.۳۰. ۳۸.] Hz

Top frequencies in Lead ۸: [۸. ۱۶. ۲۵. ۲۲. ۱۷.] Hz

Top frequencies in Lead ۹: [۸. ۱۶. ۲۵. ۱۷. ۲۲.] Hz

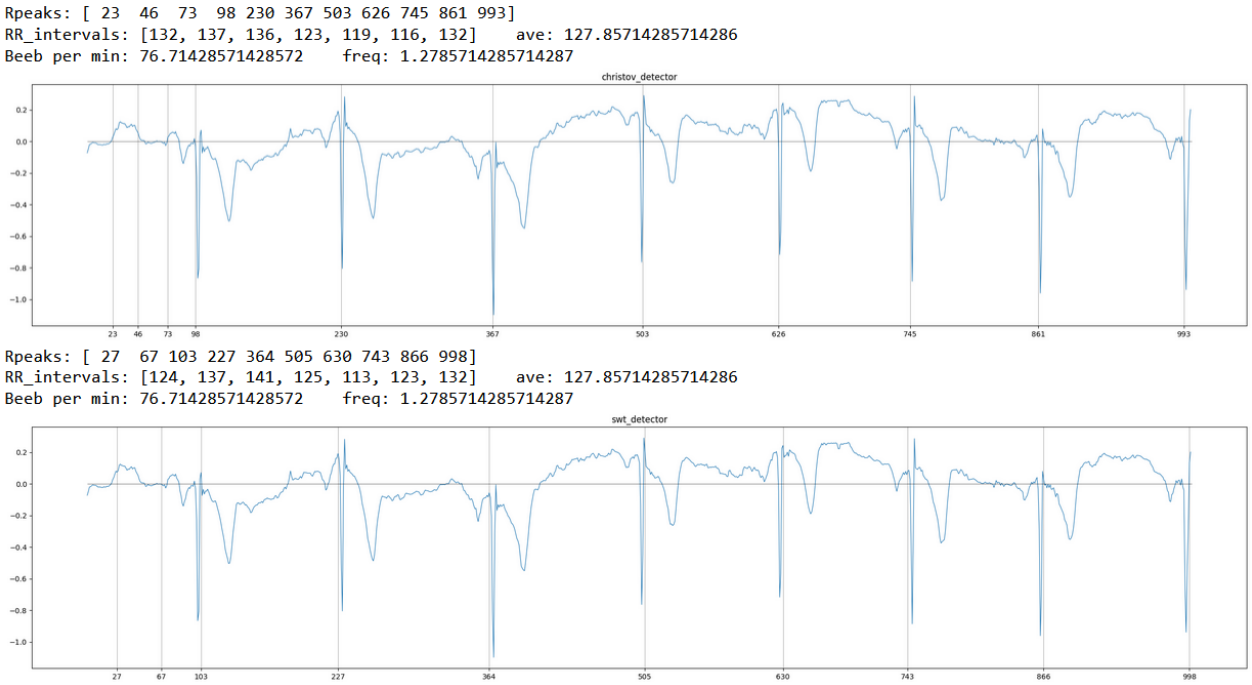
Top frequencies in Lead ۱۰: [۸. ۷. ۹.۳۰. ۵.] Hz

Top frequencies in Lead ۱۱: [۸. ۷. ۹. ۳۸. ۳۰.] Hz

Top frequencies in Lead ۱۲: [۸. ۷. ۹. ۵.۳۸.] Hz

جالب اســت که بعضــی فرکانس ها مانند ۸ هرتز و ۴ هرتز در بســیاری از کانال ها جزو فرکانس های اصــلی سیگنال هستند.

ســپس با اســتفاده از ecgdetectors با اســتفاده از دو الگوریتم christov_detector و swt_detector که با آزمون و خطا متوجه بهترین عملکرد از آن ها شدیم پیک های سیگنال را بدست می‌آوریم:



شکل ۸-۳ پیک های R سیگنال

که این مقدار ۱.۲۷ فرکانس قلب این ســابجکت را نشــان می‌دهد که بعدا در تایین batch size می‌تواند مورد استفاده قرار گیرد.

سپس باید مقادیر اسمی در دیتاست را به مقادیر عددی تبدیل کنیم و لیبل دسته بیماری به فرمت one hot انکد کنیم که به شکل زیر درمی‌آید:

دقت کنید که ۲۱ هزار داده‌ی دیتاست در این مثال فقط ۵۰۰۰ دیتا بیرون آورده و استفاده شده اند.

جدول ۳۲- فرمت *one hot* انکد لیبل ها

	NORM	MI	STTC	CD	HYP
0	1	0	0	0	0
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	0	0	0	0
...	...	...	...	...	...
4995	1	0	0	0	0
4996	0	0	0	1	0
4997	0	1	1	0	0
4998	0	0	1	0	1
4999	1	0	0	0	0

5000 rows × 5 columns

یکی از نکاتی که همواره در پیاده سازی مدل ها باید در نظر گرفت، بالانس بودن داده در کلاس های مختلف است. این به این معناست که اوردر تعداد سمپل ها در کلاس های مخلتف داده باهم متفاوت نباشد و اختلاف فاحش نداشته باشند.

Is Balanced?

```
class_counts = np.sum(Z_train, axis=0)

# Print the count of data in each class
for i, count in enumerate(class_counts):
    print(f"Class {i+1}: {count} data points")
```

```
Class 1: 4665 data points
Class 2: 2252 data points
Class 3: 2399 data points
Class 4: 2221 data points
Class 5: 1185 data points
```

مثلا همانطور که در بالا مشاهده می‌کنید برای بدست آمدن مدل نهایی مان به این نکته توجه کردیم و کلاس ها حدودا بالانس هستند.

یکی از مراحل مهم همه الگوریتم های ماشین لرنینگ تبدیل داده های توصیفی به عددی است. در کد زیر این کار را به وسیله تابع replace انجام می‌دهیم:

```
metaDatas['infarction_stadium1'] = ecg_data['infarction_stadium1'].replace({
    'unknown': 0,
    'Stadium I': 1,
    'Stadium I-II': 2,
    'Stadium II': 3,
    'Stadium II-III': 4,
    'Stadium III': 5
}).fillna(0)

metaDatas['infarction_stadium2'] = ecg_data['infarction_stadium2'].replace({
    'unknown': 0,
    'Stadium I': 1,
    'Stadium II': 2,
    'Stadium III': 3
}).fillna(0)

# 0 represents unknown
metaDatas['heart_axis'] = ecg_data.heart_axis.replace({
    'LAD': 1,
    'ALAD': 2,
    'RAD': 3,
    'AXR': 4,
    'MID': 5,
    'ARAD': 6,
    'AXL': 7,
    'SAG': 8
}).fillna(0)
metaDatas['pacemaker'] = (ecg_data.pacemaker == 'ja, pacemaker').astype(float)
```

سپس قبل از ورود به مرحله مدل سازی، بوسیله کد بالا داده را نرمال میکنیم که این کار را در این کد بوسیله standardScaler انجام می‌دهیم:

```
# Scaling X
X_scaler = StandardScaler()
X_scaler.fit(X_train)
X_train_scaled = pd.DataFrame(X_scaler.transform(X_train), columns=X_train.columns, index=X_train.index)
X_val_scaled = pd.DataFrame(X_scaler.transform(X_val), columns=X_val.columns, index=X_val.index)
X_test_scaled = pd.DataFrame(X_scaler.transform(X_test), columns=X_test.columns, index=X_test.index)

# Scaling Y
Y_scaler = StandardScaler()
Y_scaler.fit(Y_train.reshape(-1, Y_train.shape[-1]))
Y_train_scaled = Y_scaler.transform(Y_train.reshape(-1, Y_train.shape[-1])).reshape(Y_train.shape)
Y_val_scaled = Y_scaler.transform(Y_val.reshape(-1, Y_val.shape[-1])).reshape(Y_val.shape)
Y_test_scaled = Y_scaler.transform(Y_test.reshape(-1, Y_test.shape[-1])).reshape(Y_test.shape)
```

بعد از این داده‌ی ورودی را به داده آموزش، ارزیابی و تست تقسیم می‌کنیم که ابعاد داده قبل از ورود به مدل ها را مشاهده میکنید:

```
X_train: (3769, 7) Y_train: (3769, 1000, 12) Z_train: (3769, 5)
X_val: (608, 7) Y_val: (608, 1000, 12) Z_val: (608, 5)
X_test: (623, 7) Y_test: (623, 1000, 12) Z_test: (623, 5)
```

در آخر، پیش از ورود به مرحله‌ی مدل سازی خوب است که داده‌ی پراسس شده را ذخیره کرده تا هربار نیاز نباشد که دوباره این عملیات های ذکر شده را رو داده‌ی ورودی اعمال کنیم. این کار را بوسیله joblib.dump انجام می‌دهیم و توسط کد زیر داده را پس از ذخیره بلافاصله لود می‌کنیم:

```
joblib.dump(X_train_scaled, root_path + 'X_train_scaled.joblib')
joblib.dump(Y_train_scaled, root_path + 'Y_train_scaled.joblib')
joblib.dump(Z_train, root_path + 'Z_train.joblib')
joblib.dump(X_test_scaled, root_path + 'X_test_scaled.joblib')
joblib.dump(Y_test_scaled, root_path + 'Y_test_scaled.joblib')
joblib.dump(Z_test, root_path + 'Z_test.joblib')
joblib.dump(X_val_scaled, root_path + 'X_val_scaled.joblib')
joblib.dump(Y_val_scaled, root_path + 'Y_val_scaled.joblib')
joblib.dump(Z_val, root_path + 'Z_val.joblib')

X_train_scaled = joblib.load(root_path + 'X_train_scaled.joblib')
Y_train_scaled = joblib.load(root_path + 'Y_train_scaled.joblib')
Z_train = joblib.load(root_path + 'Z_train.joblib')
X_val_scaled = joblib.load(root_path + 'X_val_scaled.joblib')
Y_val_scaled = joblib.load(root_path + 'Y_val_scaled.joblib')
Z_val = joblib.load(root_path + 'Z_val.joblib')
X_test_scaled = joblib.load(root_path + 'X_test_scaled.joblib')
Y_test_scaled = joblib.load(root_path + 'Y_test_scaled.joblib')
Z_test = joblib.load(root_path + 'Z_test.joblib')
```

فصل ۴– پیاده سازی مدل و بررسی نتایج

۴–۱– پیاده سازی شبکه های عمیق در پایتون

زبان‌های برنامه‌نویسی و کتابخانه‌های پیاده‌سازی شبکه‌های عصبی عمیق یکی از مهم‌ترین ابزارهایی هستند که برای توسعه و آموزش مدل‌های عمیق مورد استفاده قرار می‌گیرند. هر یک از این زبان‌ها و کتابخانه‌ها ویژگی‌ها و مزایای منحصر به فردی دارند که بسته به نیازها و ترجیحات پروژه، ممکن است انتخاب مناسبی برای پروژه باشند. این ابزارها همچنین معمولاً از اجتماع‌های فعال کاربران و منابع آموزشی بسیاری برخوردارند که می‌توانند در فرآیند یادگیری و استفاده از آنها به شما کمک کنند. در کنار این زبان‌ها و کتابخانه‌ها، تعداد زیادی از ابزارها و فریمورک‌های دیگر نیز برای پیاده‌سازی شبکه‌های عصبی عمیق مورد استفاده قرار می‌گیرند که هرکدام مزایا و محدودیت‌های خاص خود را دارند. انتخاب زبان و کتابخانه مناسب برای پروژه خاص خود وابسته به نیازها و ترجیحات شماست.

زبان برنامه‌نویسی Python یکی از محبوب‌ترین زبان‌های برنامه‌نویسی برای توسعه شبکه‌های عصبی عمیق است. این زبان به دلیل سادگی و قابلیت‌های بسیاری که ارائه می‌دهد، انتخاب اول بسیاری از پژوهشگران و توسعه‌دهندگان است.

کتابخانه TensorFlow یکی از پرکاربردترین کتابخانه‌های پیاده‌سازی شبکه‌های عصبی عمیق است که توسط Google توسعه داده شده است. این کتابخانه امکانات فراوانی برای طراحی، آموزش و ارزیابی مدل‌های عمیق ارائه می‌دهد. در زیر مجموعه آن، کتابخانه Keras یکی از کتابخانه‌های محبوب و ساده برای توسعه شبکه‌های عصبی عمیق در Python است. از طرف دیگر کتابخانه PyTorch یکی دیگر از محبوب‌ترین کتابخانه‌های برنامه‌نویسی برای شبکه‌های عصبی عمیق است. این کتابخانه توسط Facebook توسعه داده شده است و از طراحی ساده و ماژولار برای توسعه مدل‌های پیچیده پشتیبانی می‌کند.

در این پروژه من از کتابخانه Keras برای پیاده سازی شبکه عصبی استفاده کرده‌ام.

۴–۲– مدل LSTM چیست؟

شبکه‌های LSTM [۱] یک نوع ویژه از شبکه‌های عصبی بازگشتی [۲] هستند که برای پردازش داده‌های دنباله‌ای مانند متون، صوت‌ها و سایر داده‌های زمانی استفاده می‌شوند. طراحی LSTM برای مدیریت مشکل گسسته

---

[۱] Long Short-Term Memory

[۲] RNNs

شدن گرادیان در زمان آموزش شبکه‌های عصبی بازگشتی است که باعث می‌شود آنها نتوانند اطلاعات بلند مدت را در حین زمان مورد نظر نگه دارند.

معمولاً، یک شبکه LSTM شامل تعدادی واحد LSTM است که هر کدام دارای سه دروازه [۱]اصلی هستند: دروازه فراموشی[۲]، دروازه ورودی [۳]و دروازه خروجی[۴]. این دروازه‌ها به شبکه اجازه می‌دهند تا اطلاعاتی را که در زمان گذشته به آن ها رسیده، فراموش کند، اطلاعات جدید را به داخل سلول حافظه وارد کند و اطلاعات مناسب را از سلول حافظه برای خروجی انتخاب کند.

این دروازه‌ها بر اساس ورودی فعلی شبکه و وضعیت قبلی خود، تصمیم می‌گیرند که چه اطلاعاتی را نگه دارند، فراموش کنند یا به حافظه اضافه کنند. با استفاده از این دروازه‌ها، LSTM می‌تواند اطلاعات مربوط به زمان‌های گذشته را بیشتر نگه دارد و از آنها در تصمیم‌گیری‌های آینده استفاده کند، که این ویژگی باعث می‌شود که به خوبی برای وظایف پیش‌بینی دنباله‌های زمانی مانند ترجمه ماشینی، تشخیص گفتار، تحلیل متن و غیره مناسب باشند.

۴-۳- مدل CNN چیست؟

شبکه‌های عصبی CNNs ابزارهای قدرتمندی هستند که برای پردازش تصاویر و شناسایی الگوها در داده‌های بصری استفاده می‌شوند. این مدل‌ها از تکنیک‌های متعددی تشکیل شده‌اند که به آنها کمک می‌کند الگوهای سطح بالا را از داده‌های ورودی استخراج کنند که انواع آنرا توضیح می‌دهیم:

لایه کانولوشنی[۵]: این لایه مسئول اعمال فیلترهای کانولوشنی بر روی تصویر است. این فیلترها به صورت معمول الگوهای کوچک، مانند لبه‌ها یا گوشه‌ها را شناسایی می‌کنند و ویژگی‌های مهم را از تصویر استخراج می‌کنند.

لایه ادغام[۶]: این لایه‌ها برای کاهش ابعاد فضایی تصویر استفاده می‌شوند. با اعمال تکنیک‌هایی مانند حذف اطلاعات غیرضروری و تقریبی، ابعاد تصویر کاهش می‌یابد و پردازش سریع‌تر و کارآمدتر می‌شود.

---

[۱] Gate
[۲] Forget Gate
[۳] Input Gate
[۴] Output Gate
[۵] Convolutional Layer
[۶] Pooling Layer

لایه کاملاً متصل[1]: پس از لایه‌های کانولوشنی و ادغام، اطلاعات استخراج شده به لایه‌های کاملاً متصل منتقل می‌شوند. در این لایه‌ها، اطلاعات استخراج شده از تصویر به فضای ویژگی‌های نهایی که به طور مثال در مسئله‌ی ما ۵کلاس بیماری قلبی است، تبدیل می‌شوند و برای تصمیم‌گیری نهایی استفاده می‌شوند.

درکل CNNها توانایی زیادی برای تشخیص الگوها و ویژگی‌های پیچیده در تصاویر دارند و به دلیل توانایی‌های عمیق، در بسیاری از وظایف پردازش تصویر از جمله تشخیص اشیاء، تشخیص چهره، تصحیح تصاویر و حتی در حوزه‌ی پزشکی برای تشخیص بیماری‌ها از طریق سیگنال ها و تصاویر پزشکی موفق بوده‌اند.

۴-۴- پیاده سازی معماری های بر پایه LSTM:

۴-۴-۱- معماری پایه LSTM:

ساده ترین معماری که با استفاده از شبکه های LSTM می‌توان برای این تسک متصور شد به شکل زیر است:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_5 (LSTM)               (None, 64)                19712

 dense_8 (Dense)             (None, 5)                 325

=================================================================
Total params: 20037 (78.27 KB)
Trainable params: 20037 (78.27 KB)
Non-trainable params: 0 (0.00 Byte)
```

شکل ۴-۱ ساختار معماری شبکه پایه LSTM

[1] Fully Connected Layer

ولی همانطور که مشاهده می‌کنید این مدل اصلا قابلیت یادگیری ویژگی ها را ندارد و از دقت بسیار پایینی برخوردار است:

```
Epoch 20/50
54/54 [==============================] - 34s 631ms/step - loss: 1.9021 - accuracy: 0.4990 - val_loss: 2.0159 - val_accuracy: 0.4422
Epoch 21/50
54/54 [==============================] - 34s 632ms/step - loss: 1.9024 - accuracy: 0.4993 - val_loss: 1.9980 - val_accuracy: 0.4422
Epoch 22/50
54/54 [==============================] - 35s 646ms/step - loss: 1.9035 - accuracy: 0.4991 - val_loss: 1.9924 - val_accuracy: 0.4422
Epoch 23/50
54/54 [==============================] - 39s 726ms/step - loss: 1.9020 - accuracy: 0.4991 - val_loss: 2.0084 - val_accuracy: 0.4422
Epoch 24/50
54/54 [==============================] - 41s 766ms/step - loss: 1.9010 - accuracy: 0.4991 - val_loss: 2.0113 - val_accuracy: 0.4422
Epoch 25/50
54/54 [==============================] - 37s 687ms/step - loss: 1.9011 - accuracy: 0.4990 - val_loss: 2.0090 - val_accuracy: 0.4422
Epoch 26/50
54/54 [==============================] - 35s 647ms/step - loss: 1.9034 - accuracy: 0.4990 - val_loss: 2.0021 - val_accuracy: 0.4416
Epoch 27/50
54/54 [==============================] - 39s 712ms/step - loss: 1.9017 - accuracy: 0.4991 - val_loss: 1.9909 - val_accuracy: 0.4416
Epoch 28/50
54/54 [==============================] - 35s 656ms/step - loss: 1.9014 - accuracy: 0.4991 - val_loss: 2.0090 - val_accuracy: 0.4422
```

پس معماری مدل را پیچیده تر می‌کنیم.

۲-۴-۴- افزایش پیچیدگی شبکه LSTM:

بعد از افزایش پیچیدگی مدل مشاهده می‌کنیم که مدل دچار اورفیت می‌شود. پس باید Regularizaion مدل را افزایش دهیم. این کار را با اضافه کردن لایه dropout و batch normalization انجام می‌دهیم و به مدل زیر میرسیم:

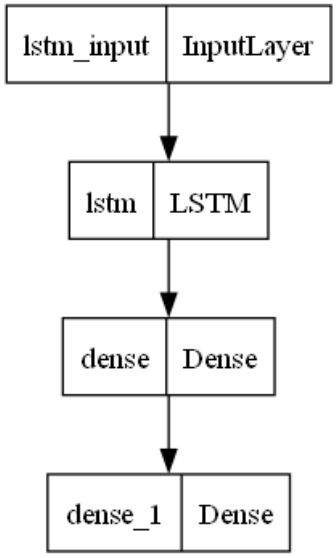
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 64)                19712

 dropout_4 (Dropout)         (None, 64)                0

 batch_normalization_4 (Bat  (None, 64)                256
 chNormalization)

 dense_4 (Dense)             (None, 64)                4160

 dropout_5 (Dropout)         (None, 64)                0

 batch_normalization_5 (Bat  (None, 64)                256
 chNormalization)

 dense_5 (Dense)             (None, 5)                 325

=================================================================
Total params: 24709 (96.52 KB)
Trainable params: 24453 (95.52 KB)
Non-trainable params: 256 (1.00 KB)
```

شکل ۲-۴ ساختار معماری شبکه افزایش یافته LSTM

ولی همانطور که از نتایج زیر مشـاهده می‌شـود باز هم دقت مدل بسـیار پایین اسـت و قابل قبول نیسـت:

```
54/54 [==============================] - 28s 515ms/step - loss: 2.0768 - accuracy: 0.4517 - val_loss: 2.1057 - val_accuracy: 0.4428
Epoch 19/50
54/54 [==============================] - 33s 615ms/step - loss: 2.0821 - accuracy: 0.4452 - val_loss: 2.1134 - val_accuracy: 0.4422
Epoch 20/50
54/54 [==============================] - 33s 607ms/step - loss: 2.0933 - accuracy: 0.4420 - val_loss: 2.1253 - val_accuracy: 0.4422
Epoch 21/50
54/54 [==============================] - 28s 523ms/step - loss: 2.1061 - accuracy: 0.4416 - val_loss: 2.1261 - val_accuracy: 0.4416
Epoch 22/50
54/54 [==============================] - 26s 477ms/step - loss: 2.1204 - accuracy: 0.4258 - val_loss: 2.1177 - val_accuracy: 0.4422
Epoch 23/50
54/54 [==============================] - 34s 631ms/step - loss: 2.1270 - accuracy: 0.4362 - val_loss: 2.1108 - val_accuracy: 0.4439
Epoch 24/50
54/54 [==============================] - 35s 646ms/step - loss: 2.1324 - accuracy: 0.4116 - val loss: 2.1220 - val accuracy: 0.4393
```

پس معماری مدل را ساده تر میکنیم و هایپرپارامتر ها را تغییر می‌دهیم تا به بهترین مدل ممکن برسیم.



شکل ۳-۴ ساختار معماری شبکه مدل ساده شده LSTM

۳-۴-۴- کاهش رگولاریزیشن LSTM:

در اینجا مدل را ساده کردیم و با تغییر هایپرپارامتر های شبکه در بهینه ساز شبکه به مدلی با نتایج بهتر دست پیدا کردیم که نتیجه آن را در پایین میبینید که به دقت ۵۹ درصد رسیده است. هرچند این دقت کافی نیست و به دنبال معماری و متد بهتری می‌رویم.

```
100/100 [==============================] - 17s 171ms/step - loss: 0.5141 - accuracy: 0.5494 - val_loss: 0.5037 - val_accuracy: 0.5925
Epoch 10/40
100/100 [==============================] - 17s 173ms/step - loss: 0.5113 - accuracy: 0.5500 - val_loss: 0.5078 - val_accuracy: 0.5925
Epoch 11/40
100/100 [==============================] - 18s 176ms/step - loss: 0.5132 - accuracy: 0.5500 - val_loss: 0.5037 - val_accuracy: 0.5925
Epoch 12/40
100/100 [==============================] - 17s 172ms/step - loss: 0.5097 - accuracy: 0.5512 - val_loss: 0.5027 - val_accuracy: 0.5900
Epoch 13/40
100/100 [==============================] - 17s 173ms/step - loss: 0.5093 - accuracy: 0.5500 - val_loss: 0.5031 - val_accuracy: 0.5925
Epoch 14/40
100/100 [==============================] - 17s 171ms/step - loss: 0.5072 - accuracy: 0.5500 - val_loss: 0.5088 - val_accuracy: 0.5850
Epoch 15/40
100/100 [==============================] - 17s 172ms/step - loss: 0.5097 - accuracy: 0.5506 - val_loss: 0.5035 - val_accuracy: 0.5925
Epoch 16/40
100/100 [==============================] - 17s 171ms/step - loss: 0.5097 - accuracy: 0.5494 - val_loss: 0.5055 - val_accuracy: 0.5850
Epoch 17/40
100/100 [==============================] - 17s 172ms/step - loss: 0.5071 - accuracy: 0.5487 - val_loss: 0.5055 - val_accuracy: 0.5925
Epoch 18/40
100/100 [==============================] - 17s 172ms/step - loss: 0.5081 - accuracy: 0.5500 - val_loss: 0.4987 - val_accuracy: 0.5925
```

در زیر می‌توانید پارامتر های شبکه را مشاهده کنید:

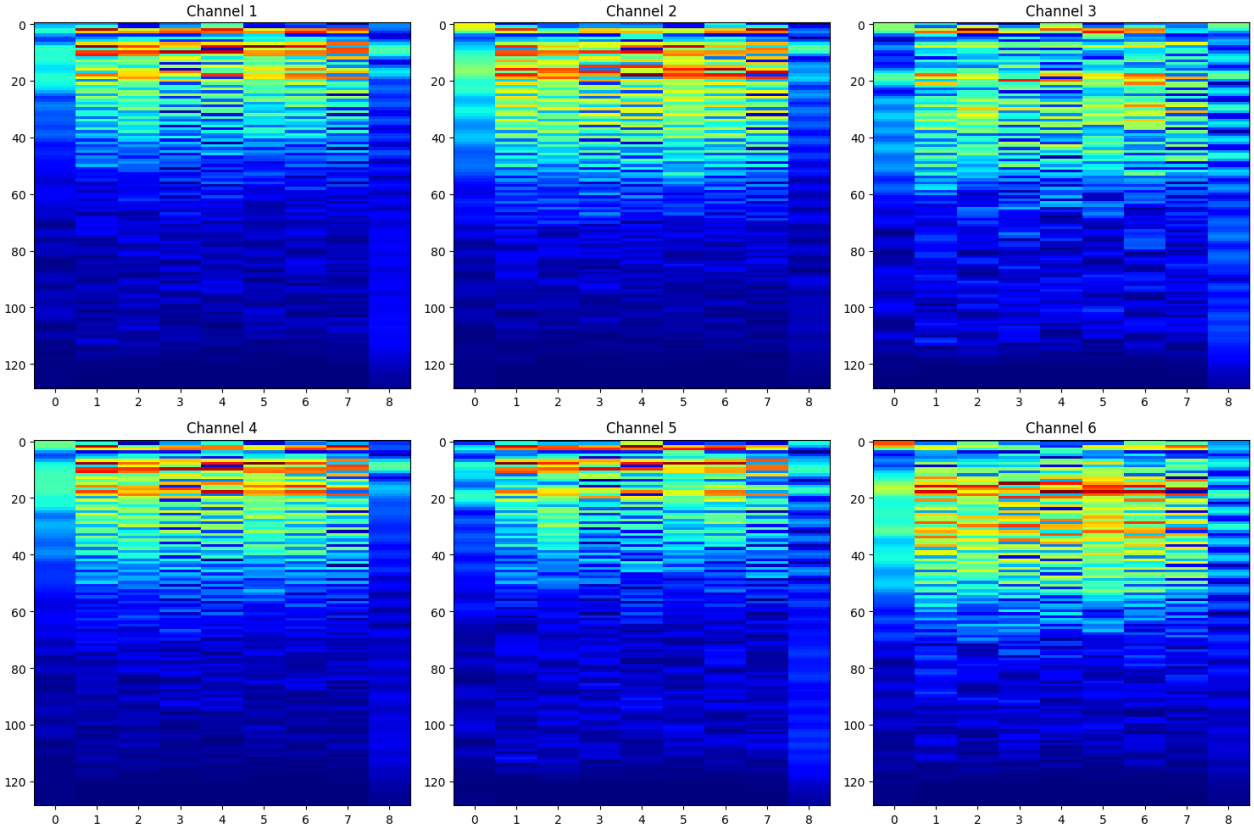
```
# Define model parameters
input_shape = (1000, num_channel)
num_classes = 5

# Define LSTM model
model = Sequential()
model.add(LSTM(64, input_shape=input_shape))
# model.add(Dropout(0.2))
# model.add(BatchNormalization())

model.add(Dense(64, activation='relu', kernel_regularizer=regularizers.l2(l=0.001)))
# model.add(Dropout(0.2))
# model.add(BatchNormalization())
model.add(Dense(num_classes, activation='softmax'))

# Compile model
adam = optimizers.legacy.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
model.compile(loss='binary_crossentropy', optimizer=adam , metrics=['accuracy'])

# Print model summary
model.summary()

# Train model
model.fit(Y_train_scaled, Z_train, epochs=40, batch_size=16 , validation_split=0.2)
```

۴–۵– پیاده سازی معماری های برپایه CNN:

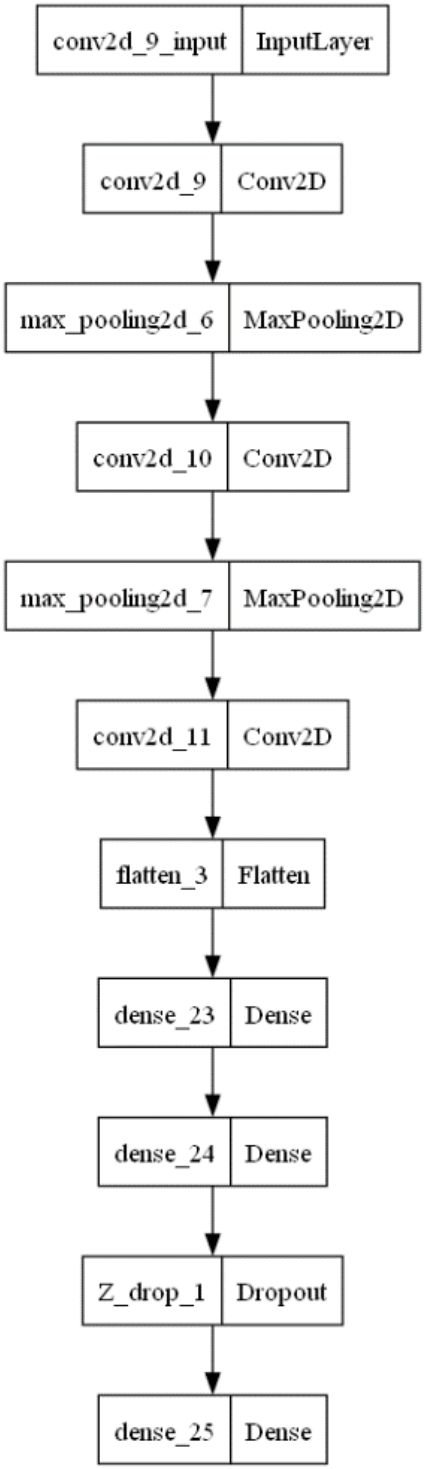
۴–۵–۱ CNN در حوزه فرکانسی با استفاده از STFT, CWT

برای پیاده سازی این روش از هر ۶ کلاس اول از ۱۲ کلاس داده هر سابجکت (که تعیین بیماری افراد نقش اساسی تری دارند) تبدیل شورت تایم فوریه یا ویولت میگیریم.

خروجی تبدیل شورت تایم فوریه روی ۶ کانال سابجکت اول را مشاهده می‌کنید:



شکل ۱۲-۴ خروجی تبدیل شورت تایم فوریه روی ۶ کانال سابجکت اول



شکل ۱۳-۴ ساختار شبکه در مدل باورودی فرکانسی

حال این ۶ تصویر را در بعد سوم به هم متصل کرده و به شبکه عصبی کانولوشنی به ساختار روبه رو می‌دهیم:

در ابتدا مدل از افزایش گرادیان فاحش رنج می‌برد و با طول زمان دقت مدل به شدت کاهش پیدا می‌کرد. ولی **با کاهش نرخ یادگیری[1] و افزایش رگولاریزیشن این مشکل حل شد.**

رگولاریزیشن در شبکه های عصبی یک تکنیک مهم برای جلوگیری از **Overfitting** و **Underfitting** مدل است .به طور خلاصه، رگولاریزیشن با جریمه کردن مدل به دلیل پیچیدگی بیش از حد، آن را به سمت تعمیم بهتر به داده های جدید سوق می دهد.

انواع مختلفی از رگولاریزیشن وجود دارد:

رگولاریزیشن L۱, L۲: این روش با اضافه کردن جریمه ای به مجموع مربعات خطاها از بزرگ شدن ضرایب نرون ها و در نتیجه وابستگی بیش از حد مدل به یک ویژگی جلوگیری می‌کند.

Dropout: این روش به طور تصادفی برخی از نرون‌های شبکه عصبی را در طول آموزش غیرفعال می کند، که به مدل کمک می کند تا به ویژگی های مختلف داده ها وابستگی کمتری داشته باشد و مدل اورفیت نشود.

Early stopping: این روش، آموزش مدل را زمانی که عملکرد آن در مجموعه داده های اعتبارسنجی شروع به کاهش می کند یا افزایش چندانی در مرحله آموزش ندارد، متوقف می کند.

[1] learning rate

در عکس زیر حل مشکل افزایش گرادیان را مشاهده می‌کنید:

```
13/13 [==============================] - 1s 61ms/step - loss: 52.0195 - accuracy: 0.3331 - val_loss: 78.2931 - val_accuracy: 0.4225
Epoch 6/10
13/13 [==============================] - 1s 62ms/step - loss: 313.6123 - accuracy: 0.3388 - val_loss: 343.1458 - val_accuracy: 0.5500
Epoch 7/10
13/13 [==============================] - 1s 61ms/step - loss: 1392.1661 - accuracy: 0.3100 - val_loss: 1641.2991 - val_accuracy: 0.5500
Epoch 8/10
13/13 [==============================] - 1s 67ms/step - loss: 5442.3721 - accuracy: 0.3613 - val_loss: 4791.3901 - val_accuracy: 0.5500
Epoch 9/10
13/13 [==============================] - 1s 60ms/step - loss: 19718.0391 - accuracy: 0.3000 - val_loss: 16921.5586 - val_accuracy: 0.5500
Epoch 10/10
13/13 [==============================] - 1s 63ms/step - loss: 53637.8984 - accuracy: 0.3113 - val_loss: 35996.9414 - val_accuracy: 0.5500
13/13 [==============================] - 0s 10ms/step - loss: 35996.9375 - accuracy: 0.5500
Validation loss: 35996.9375
Validation accuracy: 0.550000011920929
```

تا اینجا بهبود چشم گیری داشتیم ولی همانطور که در عکس زیر مشاهده می‌کنید با وجود تغییرات فراوان در معماری مدل و هایپر پارامتر های آن، دقت مدل در ۵۵ درصد گیر میکند و مدل آندر فیت است و به دقت لازم نمی‌رسد.

```
Epoch 6/10
13/13 [==============================] - 1s 62ms/step - loss: 2.2399 - accuracy: 0.5213 - val_loss: 2.1160 - val_accuracy: 0.5500
Epoch 7/10
13/13 [==============================] - 1s 62ms/step - loss: 2.4586 - accuracy: 0.4594 - val_loss: 2.2346 - val_accuracy: 0.5500
Epoch 8/10
13/13 [==============================] - 1s 62ms/step - loss: 3.1595 - accuracy: 0.4106 - val_loss: 2.4282 - val_accuracy: 0.5500
Epoch 9/10
13/13 [==============================] - 1s 59ms/step - loss: 4.6180 - accuracy: 0.3537 - val_loss: 3.2974 - val_accuracy: 0.5500
Epoch 10/10
13/13 [==============================] - 1s 62ms/step - loss: 6.6532 - accuracy: 0.3481 - val_loss: 4.2043 - val_accuracy: 0.5500
13/13 [==============================] - 0s 10ms/step - loss: 4.2043 - accuracy: 0.5500
Validation loss: 4.204294204711914
Validation accuracy: 0.550000011920929
```

علت این امر را در دو موضوع می‌توان دانست:

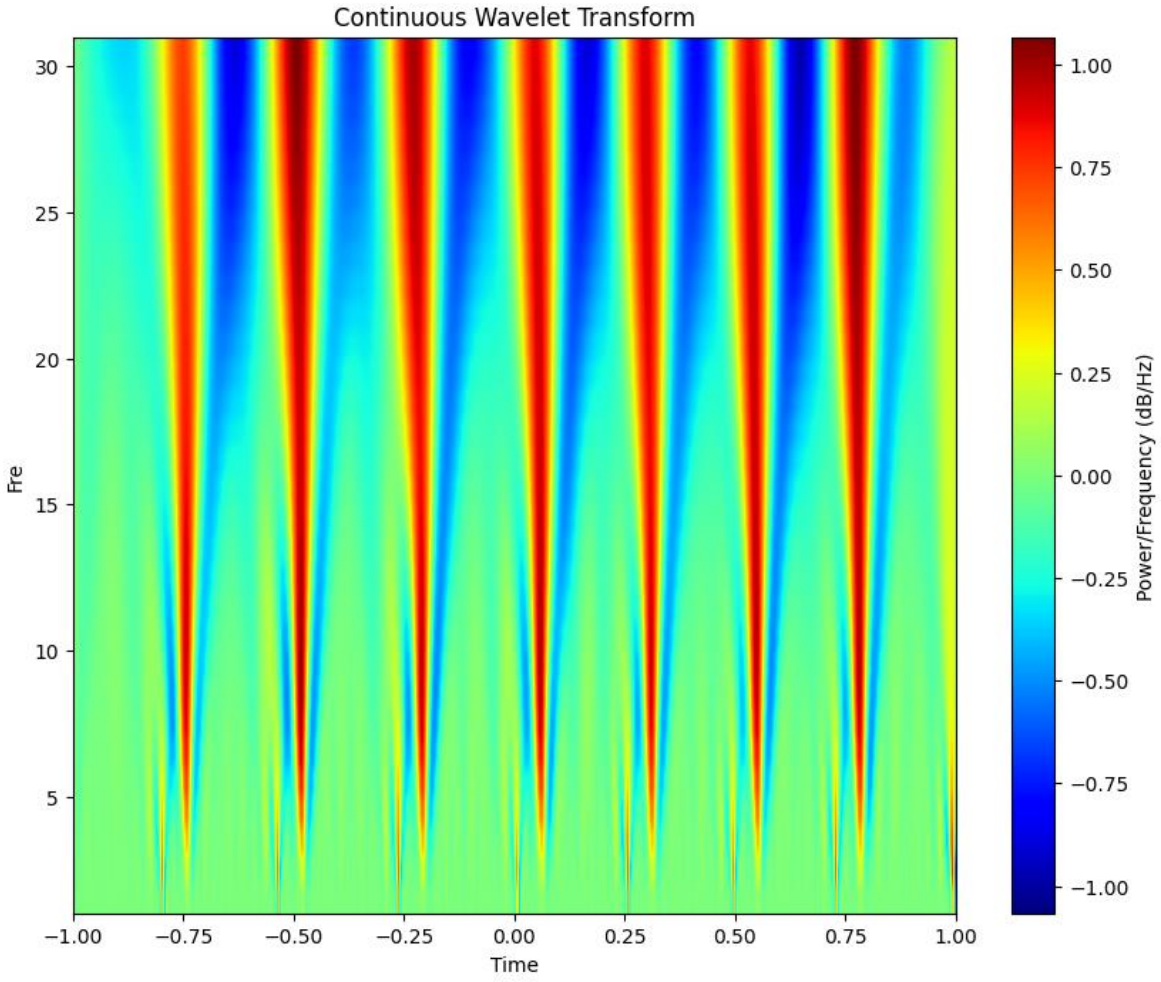
۱. در بسیاری از سیگنال ها لزوما اطلاعات سیگنال در حوزه فرکانسی آن ها ذخیره نشده است. لذا با تمرکز مدل بر روی فرکانس ممکن است به وضوح ارتباطی میان لیبل ها و فیچر ها نتوان پیدا کرد.

۲. ممکن است با توجه به پیچیدگی داده و تعداد کانال های داده، پیچیدگی مدل یا تعداد سمپل برای پیدا کردن ارتباط میان فیچرها و لیبل ها کافی نباشد.

همچنین از تبدیل ویولت هم استفاده کردیم که برای خروجی آن روی سیگنال به شکل زیر نمایش داده می‌شود. همانطور که در کد زیر می‌بینید با استفاده از کتابخانه scipy تبدیل ویولت سیگنال را می‌گیریم و نمودار آنرا می‌کشیم:

```python
from scipy import signal
cwtmatr = signal.cwt(sig, signal.ricker, widths)
fig = plt.figure(figsize=(10, 8))
cwtmatr_yflip = np.flipud(cwtmatr)
plt.imshow(cwtmatr_yflip, extent=[-1, 1, 1, 31], cmap='jet', aspect='auto',
           vmax=abs(cwtmatr).max(), vmin=-abs(cwtmatr).max())
```

که خروجی تبدیل، به شکل زیر است:



شکل ۴-۱۴ خروجی تبدیل CWT از کانال اول سابجکت اول

ولی همانطور که در عکس زیر مشاهده می‌کنید به دلیل حجم محاسباتی زیاد این تبدیل و تعداد چنل های زیاد سیگنال، با وجود تست روش های مختلف از جمله کاهش حداکثری داده، با مشکل کمبود منابع محاسباتی مواجه شدیم:

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
Cell In[22], line 37
     35     subject_wavelet = compute_concat_wavelet_transform(Y_train[i])
     36     wavelet_images.append(subject_wavelet)
---> 37 wavelet_images = np.stack(wavelet_images, axis=0)
     39 # Plot Wavelet Transform images for the first subject
     40 fig, axs = plt.subplots(2, 3, figsize=(15, 10))

File <__array_function__ internals>:200, in stack(*args, **kwargs)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site
e-packages\numpy\core\shape_base.py:471, in stack(arrays, axis, out, dtype, casting)
    469 sl = (slice(None),) * axis + (_nx.newaxis,)
    470 expanded_arrays = [arr[sl] for arr in arrays]
--> 471 return _nx.concatenate(expanded_arrays, axis=axis, out=out,
    472                       dtype=dtype, casting=casting)

File <__array_function__ internals>:200, in concatenate(*args, **kwargs)

MemoryError: Unable to allocate 11.4 GiB for an array with shape (1000, 1000, 128, 12) and data type float64
```

شکل ۴-۱۵ مشکل کمبود منابع محاسباتی در تبدیل CWT

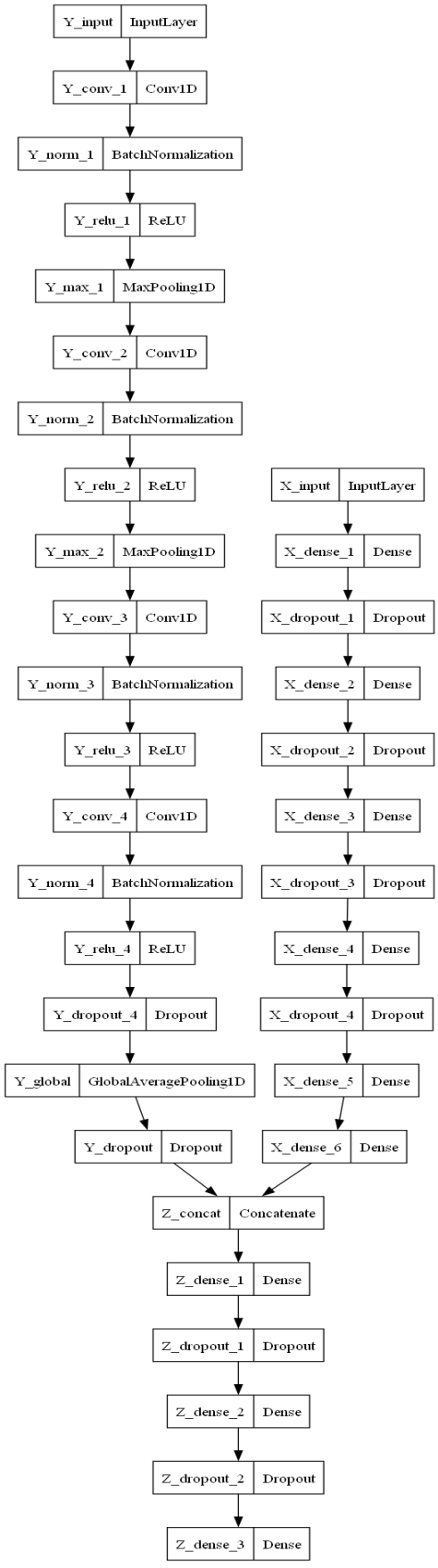
حتی این الگوریتم را با معماری Le_Net دیگری در کتابخانه pytorch امتحان کردیم که نتایج مشابه بود!

```
LeNet5(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (relu1): ReLU()
  (maxpool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (relu2): ReLU()
  (maxpool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv2d(16, 120, kernel_size=(5, 5), stride=(1, 1))
  (relu3): ReLU()
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=959760, out_features=84, bias=True)
  (relu4): ReLU()
  (fc2): Linear(in_features=84, out_features=5, bias=True)
)
Number of parameters: 80671341

Epoch [5/35], Train Loss: 1.8364, Train Acc: 55.30%, Test Loss: 1.8734, Test Acc: 53.00%
Epoch [6/35], Train Loss: 1.8193, Train Acc: 55.30%, Test Loss: 1.8690, Test Acc: 53.00%
Epoch [7/35], Train Loss: 1.8356, Train Acc: 55.30%, Test Loss: 1.8860, Test Acc: 53.00%
Epoch [8/35], Train Loss: 1.8269, Train Acc: 55.30%, Test Loss: 1.8852, Test Acc: 53.00%
Epoch [9/35], Train Loss: 1.7973, Train Acc: 55.30%, Test Loss: 1.8452, Test Acc: 53.00%
```

با وجود عدم موفقیت این دو متد برای پژوهش ما، برای کارهای آینده این دو مدل می‌توانند از پتانسیل خوبی برخوردار باشند زیرا از هردو خصوصیت زمانی و فرکانسی سیگنال ها حمایت می‌کنند.

۲–۵–۴– CNN یک بعدی در حوزه زمانی (مدل موفق نهایی)

برای این مدل از داده‌ی ۵۰۰۰ سابجکت استفاده کردیم. در نهایت شبکه کانولوشنی روی خود سیگنال زمانی را امتحان کردیم که موفق ترین مدل بدست آمد. ساختار این شبکه کانولوشنی را در زیر می‌بینید:



شکل ۱۶-۴ ساختار مدل CNN یک بعدی

همانطور که ساختار این شبکه پیداست از دو شاخه تشکیل شده است.

شاخه سمت راست اطلاع جدولی هر سابجکت را می‌گیرد مانند:

Sex, height, weight, scp_codes, heart_axis

شاخه سمت چپ سیگنال قلبی فرد در ۱۲ کانال را بوسیله شبکه CNN می‌گیرد و بررسی می‌کند و این دوشاخه در نهایت به هم می‌رسند جایی که توسط سه لایه Fully connected تصمیم گیری نهایی خروجی مدل، هم براساس سیگنال که از شاخه سمت چپ آمده و هم براساس داده جدولی که از شاخه سمت راست آمده انجام می‌شود.

حال به بررسی کد این مدل می‌پردازیم.

در این قسمت از کد ساختار شاخه سمت راست که داده جدولی سابجکت های را می‌گیرد مشاهده می‌کنید. همانطور که مشاهده می‌کنید ابتدا لایه ها شبکه CNN را تعریف کرده ایم.

```
# X model
X_input = tf.keras.layers.Input(X_train_scaled.shape[1:], name="X_input")
X = Dense(64, activation='relu', name="X_dense_1")(X_input)
# X = Dropout(0.3, name="X_dropout_1")(X)
X = Dense(128, activation='relu', name="X_dense_2")(X)
# X = Dropout(0.3, name="X_dropout_2")(X)
X = Dense(64, activation='relu', name="X_dense_3")(X)
X = Dropout(0.3, name="X_dropout_3")(X)
X = Dense(64, activation='relu', name="X_dense_4")(X)
X = Dropout(0.3, name="X_dropout_4")(X)
X = Dense(32, activation='relu', name="X_dense_5")(X)
outputX = Dense(Z_train.shape[-1], activation='sigmoid', name="X_dense_6")(X)

modelX = tf.keras.Model( inputs= X_input , outputs= outputX)
adam = optimizers.legacy.Adam(learning_rate=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.001)

modelX.compile(loss='binary_crossentropy',
               optimizer=adam,
               metrics=['binary_accuracy', 'Precision', 'Recall'])


modelX.fit( X_train_scaled, Z_train, epochs = 120, batch_size=16, validation_data= (X_val_scaled, Z_val) )
```

حال این شـــاخه را Train می‌کنیم و همانطور که مشـــاهده می‌کنید به دقت ۷۹.۵۷ برای این شـــاخه که با داده‌ی جدولی هر سابجکت کار می‌کند، بدست می‌آید:

```
Epoch 106/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4056 - binary_accuracy: 0.8240 - precision: 0.7197 - r
ecall: 0.4910 - val_loss: 0.4845 - val_binary_accuracy: 0.7947 - val_precision: 0.6639 - val_recall: 0.4092
Epoch 107/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4075 - binary_accuracy: 0.8231 - precision: 0.7203 - r
ecall: 0.4843 - val_loss: 0.4849 - val_binary_accuracy: 0.7951 - val_precision: 0.6632 - val_recall: 0.4130
Epoch 108/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4063 - binary_accuracy: 0.8233 - precision: 0.7221 - r
ecall: 0.4832 - val_loss: 0.4844 - val_binary_accuracy: 0.7941 - val_precision: 0.6592 - val_recall: 0.4130
Epoch 109/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4049 - binary_accuracy: 0.8256 - precision: 0.7277 - r
ecall: 0.4891 - val_loss: 0.4895 - val_binary_accuracy: 0.7905 - val_precision: 0.6436 - val_recall: 0.4156
Epoch 110/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4071 - binary_accuracy: 0.8242 - precision: 0.7219 - r
ecall: 0.4889 - val_loss: 0.4862 - val_binary_accuracy: 0.7937 - val_precision: 0.6585 - val_recall: 0.4118
Epoch 111/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4054 - binary_accuracy: 0.8248 - precision: 0.7235 - r
ecall: 0.4904 - val_loss: 0.4872 - val_binary_accuracy: 0.7944 - val_precision: 0.6612 - val_recall: 0.4118
Epoch 112/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4060 - binary_accuracy: 0.8247 - precision: 0.7300 - r
ecall: 0.4805 - val_loss: 0.4877 - val_binary_accuracy: 0.7944 - val_precision: 0.6612 - val_recall: 0.4118
Epoch 113/120
236/236 [==============================] - 1s 2ms/step - loss: 0.4051 - binary_accuracy: 0.8257 - precision: 0.7254 - r
ecall: 0.4936 - val_loss: 0.4838 - val_binary_accuracy: 0.7944 - val_precision: 0.6612 - val_recall: 0.4118
Epoch 114/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4056 - binary_accuracy: 0.8238 - precision: 0.7212 - r
ecall: 0.4872 - val_loss: 0.4859 - val_binary_accuracy: 0.7957 - val_precision: 0.6633 - val_recall: 0.4182
Epoch 115/120
236/236 [==============================] - 1s 2ms/step - loss: 0.4032 - binary_accuracy: 0.8249 - precision: 0.7299 - r
ecall: 0.4815 - val_loss: 0.4905 - val_binary_accuracy: 0.7957 - val_precision: 0.6646 - val_recall: 0.4156
Epoch 116/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4051 - binary_accuracy: 0.8240 - precision: 0.7191 - r
ecall: 0.4919 - val_loss: 0.4840 - val_binary_accuracy: 0.7944 - val_precision: 0.6718 - val_recall: 0.3926
Epoch 117/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4052 - binary_accuracy: 0.8253 - precision: 0.7329 - r
ecall: 0.4796 - val_loss: 0.4915 - val_binary_accuracy: 0.7918 - val_precision: 0.6475 - val_recall: 0.4182
Epoch 118/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4027 - binary_accuracy: 0.8254 - precision: 0.7299 - r
ecall: 0.4849 - val_loss: 0.4901 - val_binary_accuracy: 0.7911 - val_precision: 0.6455 - val_recall: 0.4169
Epoch 119/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4050 - binary_accuracy: 0.8242 - precision: 0.7281 - r
ecall: 0.4798 - val_loss: 0.4894 - val_binary_accuracy: 0.7905 - val_precision: 0.6441 - val_recall: 0.4143
Epoch 120/120
236/236 [==============================] - 1s 3ms/step - loss: 0.4034 - binary_accuracy: 0.8253 - precision: 0.7273 - r
ecall: 0.4881 - val_loss: 0.4865 - val_binary_accuracy: 0.7951 - val_precision: 0.6646 - val_recall: 0.4105
```

حال ساختار شبکه عصبی شاخه سمت چپ را تعریف می‌کنیم:

```
Y_input = tf.keras.layers.Input(Y_train_scaled.shape[1:], name="Y_input")

Y = Conv1D(32, 3, padding='same', name="Y_conv_1")(Y_input)
Y = BatchNormalization(name="Y_norm_1")(Y)
Y = ReLU(name="Y_relu_1")(Y)
Y = MaxPooling1D(2, name="Y_max_1")(Y)

Y = Conv1D(64, 3, padding='same', name="Y_conv_2")(Y)
Y = BatchNormalization(name="Y_norm_2")(Y)
Y = ReLU(name="Y_relu_2")(Y)
Y = MaxPooling1D(2, name="Y_max_2")(Y)

Y = Conv1D(128, 3, padding='same', name="Y_conv_3")(Y)
Y = BatchNormalization(name="Y_norm_3")(Y)
Y = ReLU(name="Y_relu_3")(Y)

Y = Conv1D(64, 3, padding='same', name="Y_conv_4")(Y)
Y = BatchNormalization(name="Y_norm_4")(Y)
Y = ReLU(name="Y_relu_4")(Y)
Y = Dropout(0.3, name="Y_dropout_4")(Y)

Y = GlobalAveragePooling1D(name="Y_global")(Y)

outputY = Dropout(0.4, name="Y_dropout")(Y)
```

در اینجا شاخه اصلی که دو شاخه سمت راست و چپ را به هم متصل می‌کند با نام modleZ تعریف می‌کنیم و در ورودی آن دو مدل شاخه چپ و راست را می‌دهیم:

```
Z = Concatenate(name="Z_concat")([outputX, outputY])
Z = Dense(64, activation='relu', name="Z_dense_1")(Z)
Z = Dropout(0.4, name="Z_dropout_1")(Z)
Z = Dense(64, activation='relu', name="Z_dense_2")(Z)
Z = Dropout(0.4, name="Z_dropout_2")(Z)
output = Dense(Z_train.shape[-1], activation="sigmoid", name="Z_dense_3")(Z)
from plot_model import plot_model
from tensorflow.keras.utils import plot_model

modelZ = tf.keras.Model(inputs=[X_input, Y_input], outputs=output)
```

حال optimizer مدل نهایی‌مان Adam تعریف می‌کنیم و پارامتر های آن را مشـــخص می‌کنیم. پارامتر learning_rate اندازه گام بهینه ســـاز در همگرایی و پارامتر beta میزان مومنتوم تغییرات این گام ها را تعیین می‌کنند که ســـرعت همگرایی مدل به نقطه بهینه را مشـــخص می‌کنند که در این تابع مقدار دهی کردیم. همچنین پارامتر decay میزان کاهش leaning_rate را تعیین می‌کند که باعث می‌شـــود زمانی که بهینه ساز حدود محدوده بهینه را پیدا کرد با گام هایی کوچکتر با دقت بالاتری به نقاط بهینه همگرا شود.

```
adam = optimizers.legacy.Adam(learning_rate=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.001)
modelZ.compile( loss= 'binary_crossentropy', optimizer= adam, metrics=['binary_accuracy', 'Precision', 'Recall'])

callback = [
    callbacks.ModelCheckpoint( monitor= 'val_binary_accuracy', save_best_only= True, filepath= root_path+'model'),
    callbacks.EarlyStopping( monitor= 'val_binary_accuracy', restore_best_weights= True, patience=10, min_delta= 0.001),
    callbacks.TensorBoard(log_dir= root_path+'/logs')
]

fitting = modelZ.fit( [X_train_scaled, Y_train_scaled], Z_train, epochs=20, batch_size=128,
                     callbacks= callback, validation_data=([X_val_scaled, Y_val_scaled] , Z_val))
```

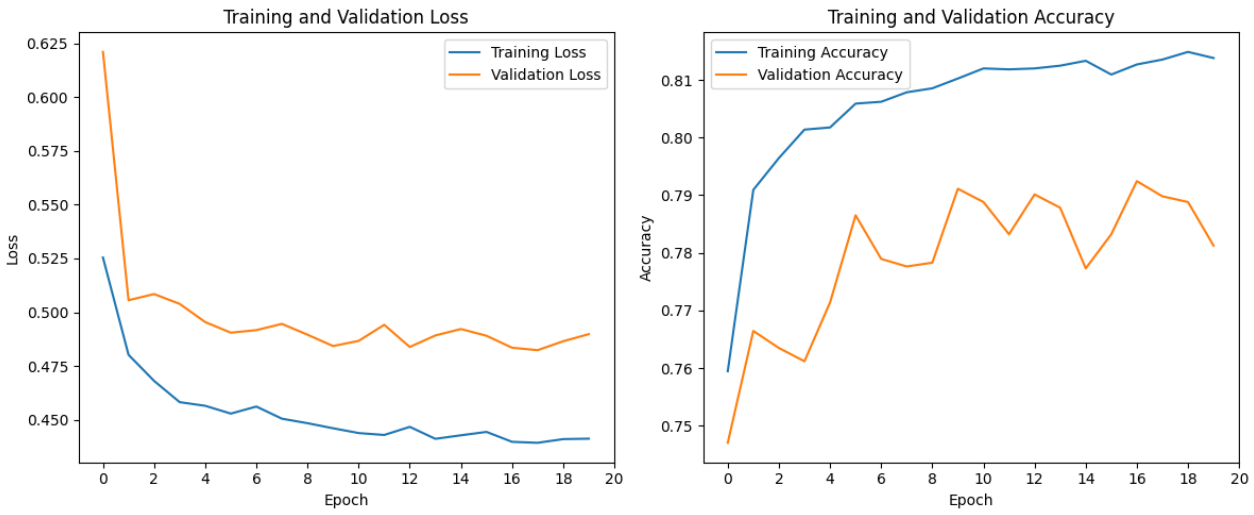
تابع callback که در زمان آموزش، فراخوانی می‌شـود در زمان آموزش هر بار که مدل به بهترین دقت رسـید از مدل یه خروجی بگیرد تا اگر در epoch های بعدی دچار افت کوچکی شـد، همواره بهترین مدل را ذخیره داشته باشیم. عکس زیر زمانی که خطی با زمینه قرمز نمایان شده است این تابع به دلیل بالا فراخوانی شده اســـت. همچنین کاربرد دیگر این تابع به نام early stopping زمانی به کار می‌آید که مدل در چند epoch

پشت هم افزایش دقت خاصی را تجربه نمی‌کند که این نشان از رسیدن به بهترین نقطه ممکن است و در این مواقع این تابع فرآیند آموزش را متوقف می‌کند.

در آخر، مدل کلی را فیت کردیم که در اینجا فرآیند آموزش آن را مشاهده می‌کنید که مدل بر داده تست به دقت ۷۹.۲۴ درصد رسیده است که دقت خوبی به شمار می‌رود:

```
Epoch 10/20
30/30 [==============================] - ETA: 0s - loss: 0.4460 - binary_accuracy: 0.8102 - precision: 0.6694 - recall:
0.4843INFO:tensorflow:Assets written to: ./ptb-xl-dataset-1.0.3\model\assets

INFO:tensorflow:Assets written to: ./ptb-xl-dataset-1.0.3\model\assets

30/30 [==============================] - 8s 267ms/step - loss: 0.4460 - binary_accuracy: 0.8102 - precision: 0.6694 - r
ecall: 0.4843 - val_loss: 0.4843 - val_binary_accuracy: 0.7911 - val_precision: 0.6554 - val_recall: 0.3964
Epoch 11/20
30/30 [==============================] - 5s 181ms/step - loss: 0.4438 - binary_accuracy: 0.8120 - precision: 0.6783 - r
ecall: 0.4794 - val_loss: 0.4867 - val_binary_accuracy: 0.7888 - val_precision: 0.6417 - val_recall: 0.4054
Epoch 12/20
30/30 [==============================] - 6s 183ms/step - loss: 0.4429 - binary_accuracy: 0.8118 - precision: 0.6774 - r
ecall: 0.4801 - val_loss: 0.4941 - val_binary_accuracy: 0.7832 - val_precision: 0.6194 - val_recall: 0.4079
Epoch 13/20
30/30 [==============================] - 5s 180ms/step - loss: 0.4467 - binary_accuracy: 0.8120 - precision: 0.6804 - r
ecall: 0.4754 - val_loss: 0.4838 - val_binary_accuracy: 0.7901 - val_precision: 0.6545 - val_recall: 0.3900
Epoch 14/20
30/30 [==============================] - 5s 182ms/step - loss: 0.4411 - binary_accuracy: 0.8125 - precision: 0.6779 - r
ecall: 0.4839 - val_loss: 0.4892 - val_binary_accuracy: 0.7878 - val_precision: 0.6367 - val_recall: 0.4079
Epoch 15/20
30/30 [==============================] - 5s 180ms/step - loss: 0.4428 - binary_accuracy: 0.8133 - precision: 0.6860 - r
ecall: 0.4746 - val_loss: 0.4922 - val_binary_accuracy: 0.7773 - val_precision: 0.5949 - val_recall: 0.4207
Epoch 16/20
30/30 [==============================] - 5s 180ms/step - loss: 0.4443 - binary_accuracy: 0.8109 - precision: 0.6724 - r
ecall: 0.4832 - val_loss: 0.4890 - val_binary_accuracy: 0.7832 - val_precision: 0.6194 - val_recall: 0.4079
Epoch 17/20
30/30 [==============================] - ETA: 0s - loss: 0.4397 - binary_accuracy: 0.8127 - precision: 0.6860 - recall:
0.4699INFO:tensorflow:Assets written to: ./ptb-xl-dataset-1.0.3\model\assets

INFO:tensorflow:Assets written to: ./ptb-xl-dataset-1.0.3\model\assets

30/30 [==============================] - 8s 277ms/step - loss: 0.4397 - binary_accuracy: 0.8127 - precision: 0.6860 - r
ecall: 0.4699 - val_loss: 0.4835 - val_binary_accuracy: 0.7924 - val_precision: 0.6736 - val_recall: 0.3747
Epoch 18/20
30/30 [==============================] - 5s 181ms/step - loss: 0.4393 - binary_accuracy: 0.8135 - precision: 0.6936 - r
ecall: 0.4625 - val_loss: 0.4823 - val_binary_accuracy: 0.7898 - val_precision: 0.6462 - val_recall: 0.4041
Epoch 19/20
30/30 [==============================] - 5s 179ms/step - loss: 0.4410 - binary_accuracy: 0.8149 - precision: 0.6944 - r
ecall: 0.4706 - val_loss: 0.4865 - val_binary_accuracy: 0.7888 - val_precision: 0.6452 - val_recall: 0.3977
Epoch 20/20
30/30 [==============================] - 5s 178ms/step - loss: 0.4412 - binary_accuracy: 0.8138 - precision: 0.6921 - r
ecall: 0.4670 - val_loss: 0.4898 - val_binary_accuracy: 0.7812 - val_precision: 0.6154 - val_recall: 0.3990
```

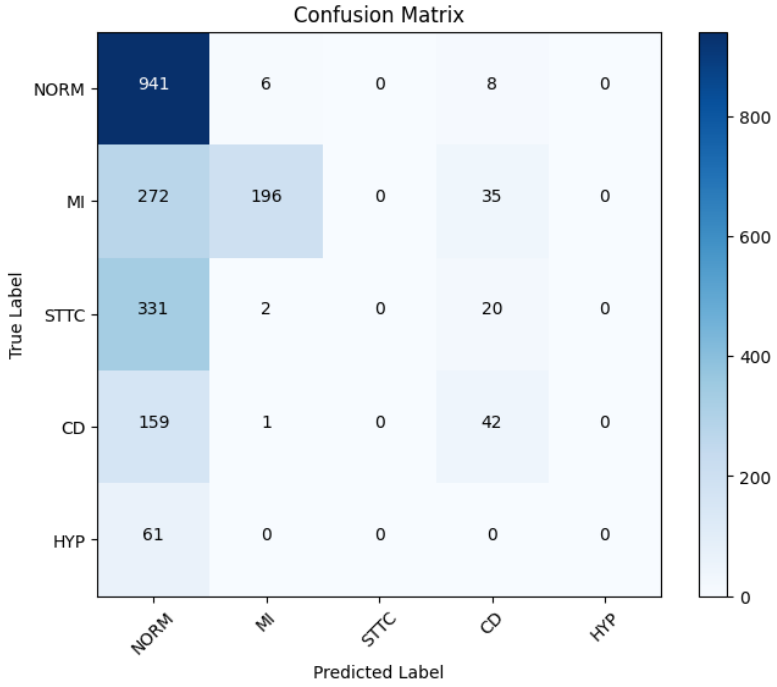
در اینجا میتوانید نمودار دقت و هزینه مدل در طی روند آموزش، هم بر داده آموزش و هم بر داده تست را مشاهده کنید:



شکل ۱۷-۴ روند کاهش هزینه و افزایش دقت مدل در طی فرآیند یادگیری مدل

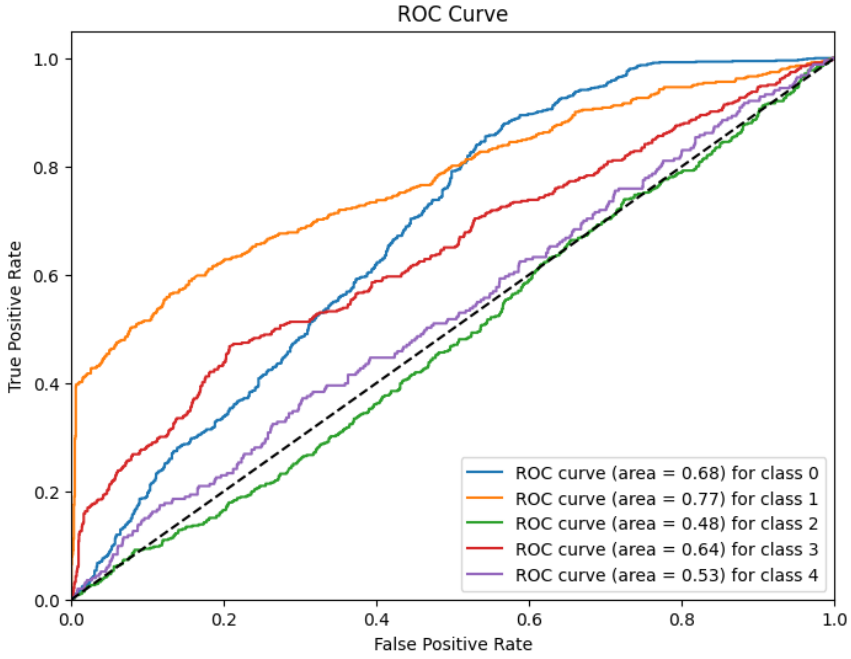
شـکل زیر نمودار confusion matrix مدل را نشـان می‌دهد که همانطور که می‌بینید بیشـتر داده ها روی قطر این نمودار قرار دارد. این موضـوع نشـان دهنده این اسـت که مدل به خوبی کار می‌کند. همانطور که مشـاهده کردید از دقت بالای مدل معلوم بود. همچنین از نقاط دیگر نمودار می‌توان به کلاس هایی که مدل در آنها ضـعف دارد پی برد. همانطور که می‌بینید مقادیر ۲۷۲ و ۳۳۱ و ۱۵۹ از مواردی بودند که مدل سـابجکت را سـالم پیش‌بینی کرده ولی در واقع دچار یکی از لیبل های MI یا STTC یا CD بودند که این موضوع قدری باعث کاهش Recall مدل شده که در نتایج بالا مشهود بود.

جدول ۴-۴ *confusion matrix* مدل *CNN* یک بعدی



همچنین در شکل زیر نمودار ROC [۱]در کلاس های مختلف را مشاهده می‌کنید:

[۱] Receiver operating characteristic



شکل ۱۸-۴ نمودار ROC در کلاس های مختلف در مدل CNN یک بعدی

این نمودار Trade off میان precision و recall را که تابع ای از threshold اســت به خوبی نشــان می‌دهد. مساحت زیر این نمودار نشان دهنده AUC است که معیاری برای عملکرد مدل به شمار می‌رود و همانطور که مشاهده می‌کنید مدل در کلاس های ۰ و ۱ و ۳ عملکرد بهتری دارد.

فصل ۵– نتیجه‌گیری

۵–۱– تحقیقات بیشتر در آینده

در آینده، تحقیقات و توسعه‌های بیشتر در حوزه پردازش و دسته‌بندی سیگنال‌های قلبی با استفاده از هوش مصنوعی و شبکه‌های عصبی می‌تواند به رشد و پیشرفت بیشتری در این زمینه منجر شود. در زیر، برخی از ایده‌هایی که در آینده می‌تواند مورد توجه قرار گیرند را بیان کرده‌ام:

۵-۱-۱- بهبود عملکرد شبکه‌های عصبی عمیق

بوسیله بهبود هرروزه‌ی الگوریتم های یادگیری عمیق و شبکه های عصبی می‌توان به دقت های بالاتر برای روش های پیشبینی و تشخیص انواع بیماری رسید.

۵-۱-۲- یادگیری ماشینی تقویت شده

یادگیری تقویت شده به ماشین‌ها امکان می‌دهد تا از تجربیات خود یاد بگیرند و تصمیم‌هایی را در مورد تحلیل سیگنال‌های قلبی بگیرند. با استفاده از این روش، ماشین‌ها می‌توانند اطلاعات بیشتری را از رویکردهای مختلف تحلیل سیگنال‌های قلبی به دست آورند و عملکرد خود را بهبود بخشند. اگرچه این روش ها به پلنت سیستم یا محیط شبیه سازی نیازمندند ولی میتوانند بسیار Robust و Data-efficient باشند.

۵-۱-۳- ترکیب داده‌های چند منبع

ترکیب داده‌های از منابع مختلف مانند سیگنال‌های ECG با داده‌های دیگر پزشکی می‌تولند اطلاعات بیشتری را ارائه دهد. به این ترتیب، تشخیص و تحلیل بیماری‌های قلبی می‌تواند دقیق‌تر و کارآمدتر باشد. برای مثال ترکیب داده های قلبی بیمار، فشار و اکسیژن خون.

۵-۱-۴- استفاده از شبکه‌های مختلط

ترکیب شبکه‌های عصبی با روش‌های سنتی می‌تواند عملکرد و دقت تحلیل سیگنال‌های قلبی را بهبود بخشد. به این ترتیب، می‌توان از مزایای هر دو روش استفاده کرد و به نتایج بهتری دست پیدا کرد.

۵-۱-۶- حل مسائل امنیتی و حریم خصوصی:

توسعه روش‌هایی برای حفظ حریم خصوصی بیماران و اطمینان از امنیت داده‌های پزشکی از اهمیت بالایی برخوردار است. این اقدامات می‌تواند اعتماد مردم به سیستم‌های تحلیل سیگنال‌های قلبی را بالا ببرد و پذیرش بیشتری داشته باشد.

۵-۲- نتیجه‌گیری نهایی

در این پژوهش قصد داشتیم که مدل شبکه عصبی طراحی کنیم که با دقت مناسب به کلاس بندی سیگنال های نوار قلب در دیتاست XL_PTB بپردازد. بدین منظور ابتدا در فصل اول به مباحث آغازین و مقدمه بحث با تاکید به طرح مسئله و بیان اهمیت و ضرورت این پژوهش پرداختیم. سپس در فصل دوم مفاهیم اولیه و کاربرد های پردازش سیگنال و اصول شبکه های عصبی را مطرح کردیم. پس از آن در فصل سوم و چهارم به بررسی دیتاست و مدل سازی شبکه های عصبی با معماری های مختلف پرداختیم و با تغییر معماری مدل ها و هایپرپارامترهای شبکه به بهترین مدل خود با معماری CNN یک بعدی رسیدیم. همانطور که مشاهده کردیم با افزایش پیچیدگی مدل ها یا افزایش regularization مدل ها لزوما نمی‌توان به مدل بهتری دست پیدا کرد و با توجه به مشکل بوجود آمده در خروجی یک مدل باید از این روش ها استفاده کرد تا به دقت بالاتر دست یافت. پژوهش ما با تست و بررسی مدل هایی بر پایه شبکه های LSTM و CNN بر روی دیتاست PTB-XL و مقایسه دقت نهایی این مدل ها به مدلی بر پایه CNN یک بعدی، با دقت مناسب دست پیدا کردیم که با دقت ۷۹.۲۴ درصد به پیش‌بینی بیماری سابجکت در ۵ کلاس می‌پردازد.

با توجه به بررسی های انجام شده در قسمت پیشینه پژوهش که بهترین دقت مقاله اول در بین هر سه مدل بررسی شده‌ی آنها ۷۶درصد و بهترین دقت مقاله دوم در بین همه مدل های بررسی شده‌ی آنها ۷۴.۹-۷۹.۱ درصد بود، دقت بدست آماده در این پژوهش (۷۹.۲۴) با اختلاف اندکی بهتر از این مدل ها عمل می‌کند و به بهترین دقت مقاله سوم در بین همه مدل های بررسی شده‌ی آنها که ۷۷.۹-۸۰.۲ درصد بود، بسیار نزدیک است در میان بازه آن قرار دارد، در حالی که با روشی کاملا متفاوت نسبت به این مقاله بر پایه شبکه های عصبی به این دقت رسیده است.

در نتیجه می‌توان گفت مدل های شبکه عصبی بخصوص CNNها می‌توانند به دقت قابل قبولی برای تشخیص و کلاس بندی سیگنال های نوار قلب دست پیدا کنند و تا زمانی که به دقت های بالایی برسند و درصد خطای آنان به صفر میل کند می‌توان از آنها کنار پزشکان و با نظارت انسانی استفاده کرد تا سرعت و دقت تشخیص و پیش‌بینی بیماری ها را افزایش دهند.

در آخر امیدواریم که نتایج حاصله بتواند به بهبود دقت و صحت روش های تشخیص بیماری‌های قلبی کمک نماید.

منابع

١. Centers for Disease Control and Prevention. "Heart Disease Facts." Accessed . Available: https://www.cdc.gov/heartdisease/facts.htm.

٢. Boulif, Abir, et al. "A Literature Review: ECG-Based Models for Arrhythmia Diagnosis Using Artificial Intelligence Techniques." PubMed, .٢٠٢٣

٣. Powell, A. "AI revolution in medicine." The Harvard Gazette, ٢٠٢٠. Accessed . Available: https://news.harvard.edu/gazette/story/١١/٢٠٢٠/risks-and-benefits-of-an-ai-revolution-in-medicine./

٤. Śmigiel S, Pałczyński K, Ledziński D. ECG Signal Classification Using Deep Learning Techniques Based on the PTB-XL Dataset. Entropy (Basel, Switzerland). ٢٠٢١ Aug;٢٣(٩):١١٢١. DOI: ١٠,٣٣٩٠/e٢٣٠٩١١٢١. PMID: ٣٤٥٧٣٧٤٦; PMCID: PMC٨٤٦٩٤٢٤.

٥. Śmigiel, Sandra, Krzysztof Pałczyński, and Damian Ledziński. ٢٠٢١. "Deep Learning Techniques in the Classification of ECG Signals Using R-Peak Detection Based on the PTB-XL Dataset" *Sensors* ٢١, no. ٢٤: ٨١٧٤. https://doi.org/١٠,٣٣٩٠/s٢١٢٤٨١٧٤

٦. Pałczyński, Krzysztof, Sandra Śmigiel, Damian Ledziński, and Sławomir Bujnowski. ٢٠٢٢. "Study of the Few-Shot Learning for ECG Classification Based on the PTB-XL Dataset" *Sensors* ٢٢, no. ٣: ٩٠٤. https://doi.org/١٠,٣٣٩٠/s٢٢٠٣٠٩٠٤

٧. BJ-Copeland. "artificial intelligence." ibmbritannica, ٢٠٢٤. Accessed . Available: https://www.britannica.com/technology/artificial-intelligence.

٨. Amisha, P. M., et al. "Overview of artificial intelligence in medicine." PubMed, .٢٠١٩

٩. Naoum, K. D. "The Dangers of AI in the Healthcare Industry." Thomas, ٢٠٢١. Accessed . Available: https://www.thomasnet.com/insights/the-challenges-and-dangers-of-ai-in-the-health-care-industry-report./

١٠. CHG-MERIDIAN. "ADVANTAGES AND DISADVANTAGES OF ARTIFICIAL INTELLIGENCE IN HEALTHCARE." Accessed . Available: https://www.chg-meridian.co.uk/resource-centre/blog/advantages-and-disadvantages-of-artificial-intelligence-in-healthcare.html.

١١. jadagi, Samyuktha. "AI Winter and Resurgence: Understanding the Cycles of Artificial Intelligence." Medium, ٢٠٢٣. Accessed . Available:

https://medium.com/@samyukthajadagi٠/ai-winter-and-resurgence-understanding-the-cycles-of-artificial-intelligence-b٠٣c٦٨c٦٦٢f.٨
١٢. Smith, G. M. "What is Signal Processing?" dewesoft, ٢٠٢٣. Accessed . Available: https://dewesoft.com/blog/what-is-signal-processing.
١٣. "Complete Guide to Understanding Signal Processing." electronicsforu, ٢٠٢٣. Accessed . Available: https://www.electronicsforu.com/technology-trends/learn-electronics/signal-processing.
١٤. Bajaj, Varun, et al. "Biomedical Signal Processing for Healthcare Applications." .٢٠٢٢
١٥. kenhub. "Heart." kenhub, ٢٠٢٣. Accessed . Available: https://www.kenhub.com/en/library/anatomy/heart.
١٦. Britannica. "heart anatomy." britannica, Accessed . Available: https://www.britannica.com/science/heart.
١٧. Basiri, M.٢٠٢٣ شماره, ."قلب چیست؟ | وظیفه، ساختار، جایگاه و عملکرد" .
١٨. cleveland clinic editors. "Electrical Conduction System of the Heart." cleveland clinic, ٢٠٢٢. Accessed . Available: https://my.clevelandclinic.org/health/body/٢٢٥٦٢-electrical-system-of-the-heart.
١٩. Potter, D. L. "Understanding an ECG." ٢٠٢٣. Accessed . Available: https://geekymedics.com/understanding-an-ecg./
٢٠. Tech, W. "A Complete Guide to Einthoven's Triangle & ECG Testing ". .٢٠٢٣ شماره,Accessed . Available: https://www.linkedin.com/pulse/complete-guide-einthovens-triangle-ecg-testing-wellnestinc./
٢١. Ryan, S. S. "Understanding the EKG Signal." Accessed . Available: https://a-fib.com/treatments-for-atrial-fibrillation/diagnostic-tests-٢/the-ekg-signal./
٢٢. Rawshani, A. Clinical ECG Interpretation.
٢٣. Duprez, P. D. "Early detection of cardiovascular disease - the future of cardiology?" . .٢٠٠٦Accessed February ١٩, ٢٠٢٤. https://www.escardio.org/Journals/E-Journal-of-Cardiology-Practice/Volume-٤/vol٤n١٩-Title-Early-detection-of-cardiovascular-disease-the-future-of-cardi.
٢٤. Y. Y. Q. Y. & T. H. Xiaoyu Sun. "Artificial intelligence in cardiovascular diseases: diagnostic and therapeutic perspectives." European Journal of Medical Research, .٢٠٢٣
٢٥. S. S. & M. Kumar. "A Systematic Review on Artificial Intelligence-Based Techniques for Diagnosis of Cardiovascular Arrhythmia Diseases: Challenges and Opportunities." SpringerLink, .٢٠٢٣

۲۶. American Heart Association. "AI may accurately detect heart valve disease and predict cardiovascular risk." Newsroom, . .۲۰۲۳Accessed February ۱۹, ۲۰۲٤. https://newsroom.heart.org/news/ai-may-accurately-detect-heart-valve-disease-and-predict-cardiovascular-risk.
۲۷. Henderson, E. "Using AI for early diagnosis of cardiovascular disease." News-medical, . .۲۰۲۳Accessed February ۱۹, ۲۰۲٤. https://www.news-medical.net/news/۲۰۲۳۰۲۲۷/Using-AI-for-early-diagnosis-of-cardiovascular-disease.aspx.

(برای مشاهده و دسترسی به کد پایتون برنامه و سایر داکیومنت ها و اطلاعات بیشتر میتوانید به گیت هاب من به آدرس https://github.com/sajjadrezvani/ECG_Ai مراجعه فرمایید.)

Abstract

Heart diseases are a major cause of death worldwide that take many lives each year. Doctors use ECGs to analyze the heart's electrical signals and diagnose these diseases, either to prevent them or treat them after they occur. In comparison with humans, computer systems, including artificial intelligence, offer more accurate and faster diagnosis of heart diseases. Many models have been developed to improve the accuracy and speed of diagnosing heart diseases or suggest new approaches; but there are some challenges like complex signals or lack of data that cause model overfitting. In our research, we tested LSTM and CNN models on the PTB-XL dataset and found a one-dimensional CNN-based model with an accuracy of ۷۹٫۲٤٪ in predicting disease across five categories. This accuracy performs better previous models and is comparable to the best accuracy in similar dataset.

Keywords: Electrocardiogram, Cardiac signal classification, Neural networks, Deep learning