**شرح نحوه انجام پروژه Artifact detection and reconstruction**

فازهای انجام:

1. تحقیق
2. جمع آوری دیتا
3. توسعه و ایده پردازی
4. استقرار روش توسعه داده شده و تست آن

شرح مراحل:

**تحقیق:**

در این مرحله به جستجو در حوزه شناسایی نویز blink و روش های مختلف مقابله با آن پرداخته شد و مقاله های مرتبط در این بخش انتخاب شدند. لیست مقاله های انتخاب شده و مطالعه شده در ادامه آمده است:

* EEG artifacts removal using machine learning algorithms  
  and independent component analysis
* Blink: A Fully Automated Unsupervised Algorithm for Eye-Blink  
  Detection in EEG Signals
* A Review on Machine Learning Algorithms in Handling EEG Artifacts
* Automatic ocular artifacts removal in EEG using deep learning
* Automated eye blink artefact removal from EEG using support vector machine an autoencoder
* Automatic EEG eyeblink artefact identification and removal technique using independent component analysis in combination with support vector machines and denoising autoencoder

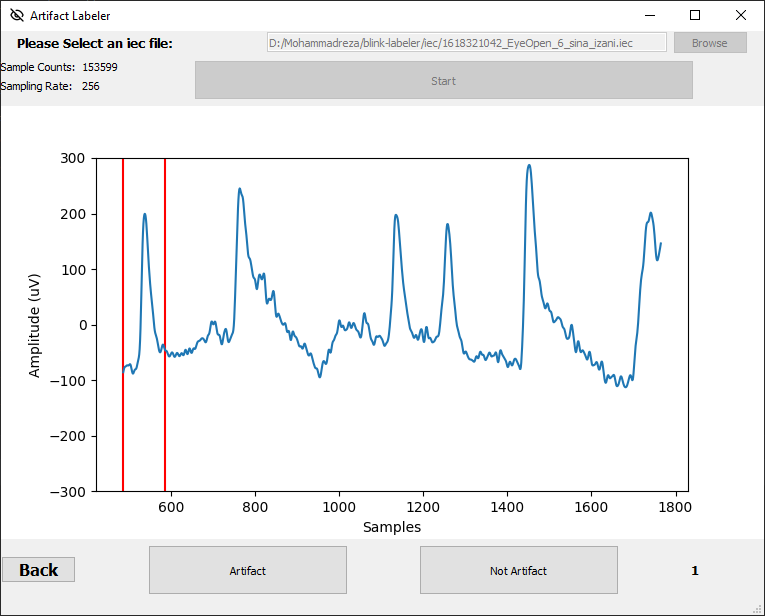
با توجه به روشهایی که قبل از انجام پروژه مد نظر داشتیم و همچنین ابزارهای موجود و نتیجه مقالات مختلف، مقاله اصلی انتخاب شد که با هایلایت مشخص شده است.

در این روش ابتدا دیتابیسی شامل بخش های دارای نویز پلک و بخش های سالم ساخته می شود و در ادامه با استفاده از ویژگی های واریانس، دامنه peak to peak و kurtosis این دو کلاس داده توسط الگوریتم طبقه بندی SVM از هم جدا می شوند. خلاصه ای از مقاله مربوطه در ادامه آمده است:

* ارائه یک روش برای شناسایی و بازسازی blink artifact
* یک پنجره 0.45 ثانیه ای با همپوشانی 50 درصد حرکت می کند
* یک SVM با ویژگی های های واریانس، دامنه peak to peak و kurtosis آموزش داده شده است
* به نظر می رسد فقط پنجره هایی که کل blink را در بر می گیرند در هنگام برچسب زدن به عنوان artifact در نظر گرفته می شود نه پنجره هایی که بخشی از آن را در بر می گیرند
* بعد از predict طبقه بند SVM، اگر پنجره نویز تشخیص داده نشود، پنجره رو به جلو حرکت می کند در غیر این صورت پنجره به سک شبکه AutoEncoder (AE) فرستاده می شود
* در طبقه بند SVM برای هر کلاس 1000 نمونه در نظر گرفته شده است
* شبکه AE با هزار سیگنال خراب به عنوان ورودی و هزار سیگنال ساالم به عنوان خروجی و 50 نورون در لایه hidden آموزش داده شده است
* این روش فقط بخش های دارای آرتیفکت را تغییر می دهد و بخش های دیگر سیگنال بدون تغییر می مانند
* این روش فقط نویز blink را تشخیص می دهد

**جمع آوری دیتا:**

برای این منظور با استفاده از PyQt5 یک App به نام blink-labeler جهت نمونه برداری توسعه داده شد و کانال های Fp1 و Fp2 از سیگنال های eye open بعد از crop نویزهای بزرگ بخش های دارای نویز آنها برچسب گذاری شد. فرکانس نمونه برداری سیگنال ها 256 هرتز است و اندازه پنجره برای برچسب گذاری 100 سمپل و overlap برابر با 75 سمپل در نظر گرفته شده است.



در ادامه نیز روی سیگنال های crop شده ICA که در نرم افزار razin توسعه داده شده (kurtosis ICA) زده می شود. بخش های خراب به عنوان ورودی AE و بخش های متناظر بخش های خراب از خروجی ICA به عنوان خروجی AE در نظر گرفته می شوند.

**توسعه و ایده پردازی:**

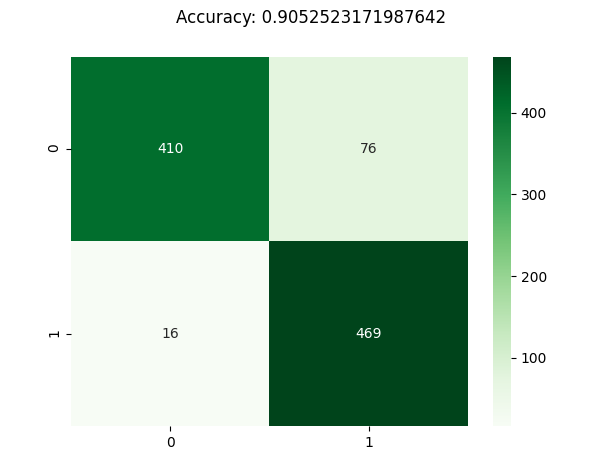
در بخش SVM:

ابتدا پنجره سیگنال تقسیم بر gain می شود در ادامه سیگنال detrend شده و سپس فیلترهای حذف نویز برق شهر و میان گذر 0.53 و 32 هرتز اعمال می شود.

در این بخش علاوه بر ویژگی های گفته شده، ویژگی های جدیدی نیز اضافه شده است:

* میانگین
* انحراف معیار
* rms
* skewness
* area under curve
* تعداد zero crossing
* مجموع لگاریتمی فرکانس های 1 تا 3.5 هرتز

با ویژگی های انتخاب شده که با هایلایت مشخص شده اند، دقت تشخیص blink بین 12 subject مختلف نزدیک به 91 درصد به دست آمد.



در بخش AE:

دو شبکه Dense و 1D convolutional آموزش داده شدند که در مقاله اصلی از شبکه Dense استفاده شده بود و در پیاده سازی ما شبکه convolution نتیجه بهتری را ارائه کرد.

در این بخش داده ها با استفاده از min max normalization نرمال می شوند و هایپر پارامترهای به دست آمده و معماری شبکه در ادامه مشخص شده است.

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv1d (Conv1D) (None, 98, 128) 512

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_1 (Conv1D) (None, 96, 32) 12320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_transpose (Conv1DTran (None, 98, 32) 3104

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_transpose\_1 (Conv1DTr (None, 100, 128) 12416

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_2 (Conv1D) (None, 100, 1) 385

=================================================================

Total params: 28,737

Trainable params: 28,737

Non-trainable params: 0

مقادیر هایپر پارامترها:

input\_shape = (100, 1)  
batch\_size = 128  
no\_epochs = 100  
validation\_split = 0.2  
max\_norm\_value = 2.0

optimizer=Adam(learning\_rate=0.001), loss='binary\_crossentropy'

**استقرار روش توسعه داده شده و تست آن:**

مدل های ذخیره شده در نرم افزار Neuro Studio لود می شوند. به ازای هر پنجره (که مقدار آن 512 سمپل است)، پنجره ای به اندازه 100 سمپل از ابتدا در نظر گرفته می شود و بعد از هر بار این پنجره 40 سمپل به جلو حرکت می کند. مانند حالتی که در زمان train شبکه داشتیم ابتدا سیگنال تقسیم بر gain سیگنال می شود، سپس سیگنال detrend شده و فیلترهای برق شهر و میان گذر بر آن اعمال می شود. اگر هر کدام از این پنجره ها به عنوان blink انتخاب شوند، آن پنجره به شبکه AE برای denoise شدن داده می شود.

در این بخش با سه چالش روبرو هستیم:

* برای ورود داده ها به شبکه AE داده ها باید نرمال شوند در نتیجه باید scale آن ها نگهداری شوند تا بعد از denoise شدن بتوان آن ها را rescale کرد
* اگر چند پنجره پشت هم به عنوان blink انتخاب شوند و آن ها از ابتدا و پشت سر هم به AE داده شوند، چون شبکه یاد گرفته است که سیگنال blink با فرکانس پایین را به فرکانس با سرعت بالاتر تبدیل کند، پس از denoise کردن پنجره اول و دادن پنجره بعدی (که خود با پنجره اول همپوشانی دارد) با توجه به اینکه پنجره اول خود denoise شده است سبب تولید خروجی نهایی فرکانسی بالا خواهد شد؛ برای جلوگیری از این اتفاق، هر کدام از پنجره ها به صورت جدا denoise می شوند و در ادامه چک می شود که آیا پنجره ها در دنباله یکدیگر هستند (آیا کمتر از 100 سمپل فاصله دارند) در این صورت فقط بخش انتهایی از پنجره denoise شده دوم به تعداد سمپل مورد نیاز به ادامه پنجره denoise شده اول اضافه می شود.
* در برخی از موارد طول blink بیش از 100 سمپل است یا قبل و بعد از وقوع blink صعود یا نزولی در سیگنال داشته باشیم، مقدار dc بخش denoise شده بیشتر از مقدار dc باقی سیگنال می شود. در این حالت سعی شد با یکی کردن میانگین دو بخش مقادیر dc را یکسان کرد اما به دلیل پایین آمدن همبستگی فرکانسی دو سیگنال قبل و بعد از artifact rejection از انجام این روش صرف نظر شد.

**کارهای آتی:**

* تست کردن تغییر اندازه طول پنجره blink برای بهینه کردن آن(احتمالا با افزایش اندازه پنجره تعداد blink تشخیص داده شده توسط SVM کاهش می یابد و dc بخش denoise شده و باقی سیگنال به هم نزدیک تر می شود)
* اضافه کردن ویژگی برای بالابردن دقت تشخیص بخش blink و non blink با افزودن ویژگی های دیگر سری های زمانی
* در نظر گرفتن اندازه امپدانس کانال در حالتهایی که blink تشخیص داده شده اند (زمانی که امپدانس کانال پایین تر است، دامنه نویز blink افزایش می یابد)
* افزایش تعداد داده ها با افزودن subject های متفاوت و نمونه های blink جدید (افزایش قدرت پیش بینی AE)
* آموزش شبکه های AE با معماری های متفاوت

در شکل هایی که در ادامه آمده است، سیگنال و psd آن قبل و بعد از artifact rejection نمایش داده شده است.

