بسم الله الرحمن الرحیم

محمد مهدی مسعودپور
۹۸۱۰۲۳۳۵

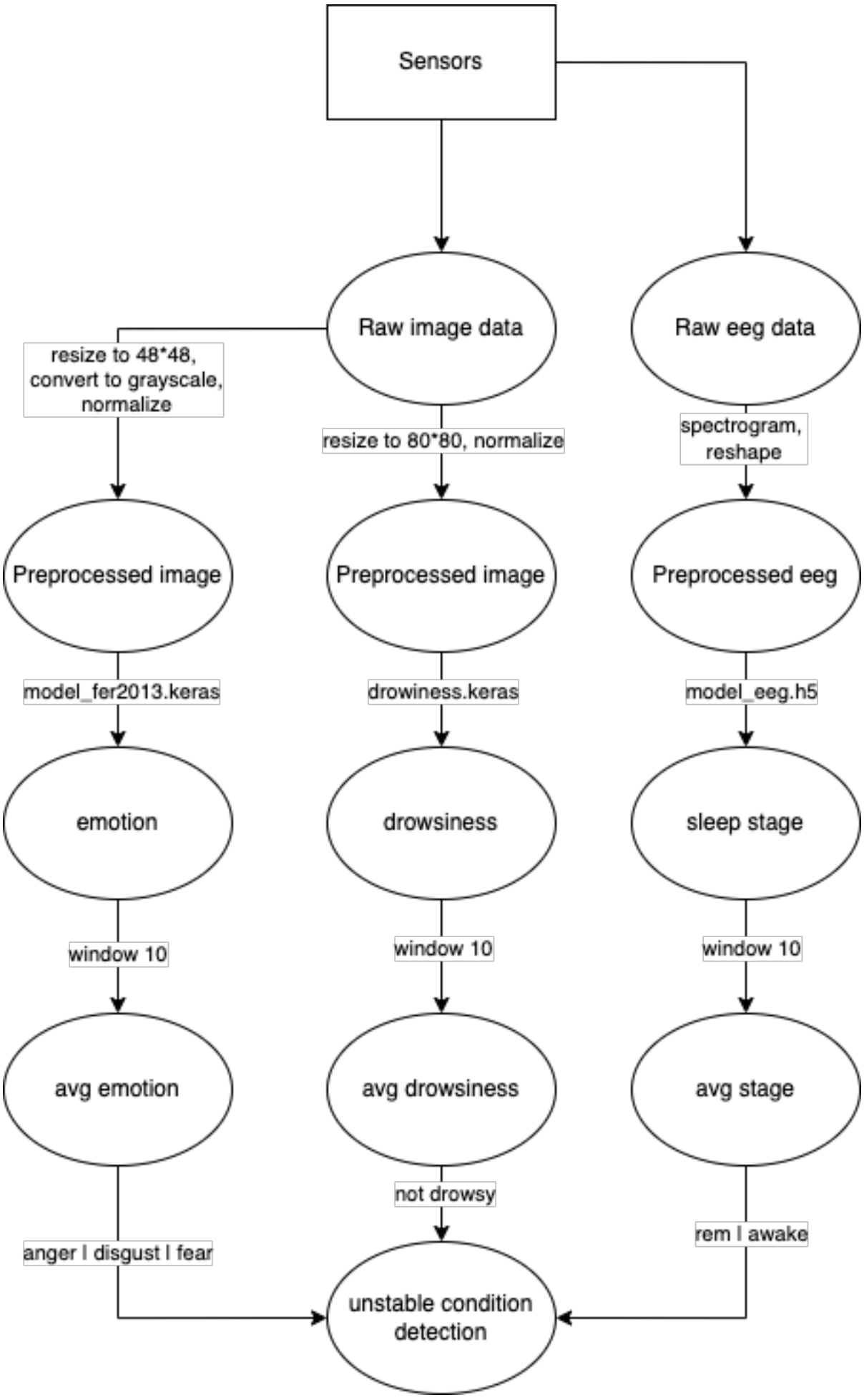
## تعریف

در این پروژه قصد داریم وضعیت روحی بیمار را با استفاده از حسگرهای eeg و دوربین به صورت زنده تشخیص داده و در صورت بروز وضعیت‌های پایدار نامطلوب، به سرپرست بیمار اطلاع دهیم.

## شرح پایپ‌لاین

ابتدا داده‌های خام eeg و تصویر فرد پس از دریافت‌شدن از حسگرها در تاپیک sensor-data ذخیره میشود.
سپس داده‌ها به سه صورت جلو میروند:
۱- ابعاد تصویر به 48*48 تغییر پیدا میکند. تصویر سیاه و سفید میشود و مقادیر آن به بازه [0,1] میرود. پس از آن تصویر حاصل به مدل model_fer2013.keras داده میشود. بیشترین حالت شخص در ۱۰ دقیقه به خروجی انتقال پیدا میکند.
۲- ابعاد تصویر به 80*80 تغییر پیدا میکند. همچنین مقادیر آن به بازه [0,1] میرود. پس از آن تصویر حاصل به مدل drowiness.keras داده میشود. بیشترین حالت شخص در ۱۰ دقیقه به خروجی انتقال پیدا میکند.
۳- spectrogram بازه از داده eeg محاسبه میشود. سپس شکل خروجی به ( اندازه spectrogram ورودی، بازه فرکانس(۶۴)، اندازه spectrogram(۴۷) و ۲) تبدیل میشود. پس از آن خروجی به مدل model_eeg.h5 ورودی داده میشود و نتیجه آن در ۱۰ دقیقه به صورت میانگین گرفته میشود و بیشترین حالت فرد بازگردانده میشود.

در نهایت اگر فرد دارای وضعیتی منفی مانند خشم، نفرت، ترس بود، (خسته نبود و بیدار بود یا در وضعیت rem بود)، هشدار در تاپیک result قرار داده میشود(۱) در غیر این صورت صفر در تاپیک result قرار داده میشود.


Sensors
Raw image data
Raw eeg data
resize to 48*48, convert to grayscale, normalize
resize to 80*80, normalize
spectrogram, reshape
Preprocessed image
Preprocessed image
Preprocessed eeg
model_fer2013.keras
drowiness.keras
model_eeg.h5
emotion
drowsiness
sleep stage
window 10
window 10
window 10
avg emotion
avg drowsiness
avg stage
not drowsy
anger | disgust | fear
rem | awake
unstable condition detection

## کد

پایپ‌لاین اصلی با استفاده از زبان برنامه‌نویسی جاوا پیاده‌سازی شده. به علت وجود مدل‌های tensorflow، از پایتون نیز استفاده شده.
کتابخانه‌های استفاده شده شامل kafka apache، flink apache، pyflink، tensorflow و opencv است.

۱- فایل producer.py همان دستگاه بیمار است. به این صورت که اطلاعات دوربین و eeg را از طریق تاپیک sensor-data در group-id: main-server به consumer.java میفرستد و منتظر پاسخ آن میماند.
۲- به علت وجود پردازش‌های tensorflow، سرور داده را از طریق تاپیک processing_raw و group-id: data-processing-group به consumer.py میفرستد. consumer.py داده‌های مربوط به دوربین و eeg را پردازش میکند و نتیجه را از طریق تاپیک processing_result و group-id: result-consumer به consumer.java میفرستد.
۳- consumer.java نیز روی آن یک میانگین متحرک گرفته، نتیجه پردازش دوربین و eeg را ترکیب کرده و نتیجه نهایی را از طریق تاپیک results و group-id: main-server به producer.py میفرستد.
۴- producer.py نیز اگر نتیجه مثبت باشد، تابع تماس را صدا میزند و در غیر این صورت به کار خود ادامه میدهد.

## راه‌اندازی سیستم

- zookeeper-server-start /usr/local/etc/kafka/zookeeper.properties
- kafka-server-start /usr/local/etc/kafka/server.properties
- flink-1.20.0/bin/start-cluster.sh

همچنین برای ساخت تاپیک‌ها:

- kafka-topics --create --topic sensor-data --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
- kafka-topics --create --topic results --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
- kafka-topics --create --topic processing_raw --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
- kafka-topics --create --topic processing_result --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

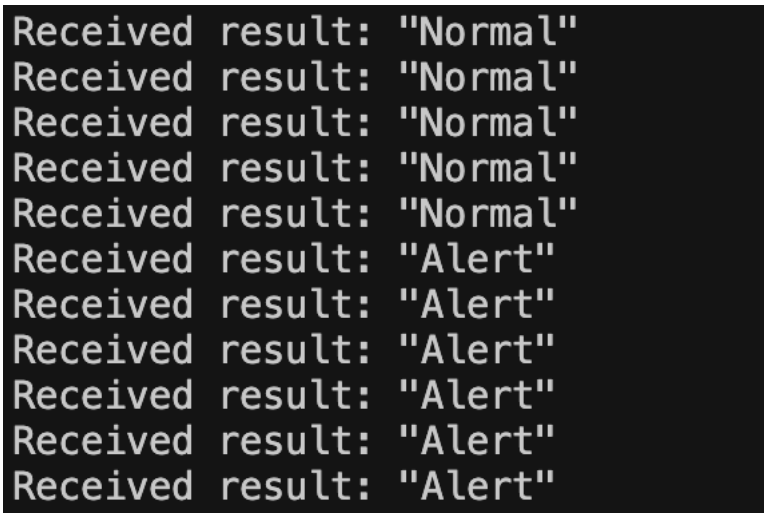
## اجرای کدها

---

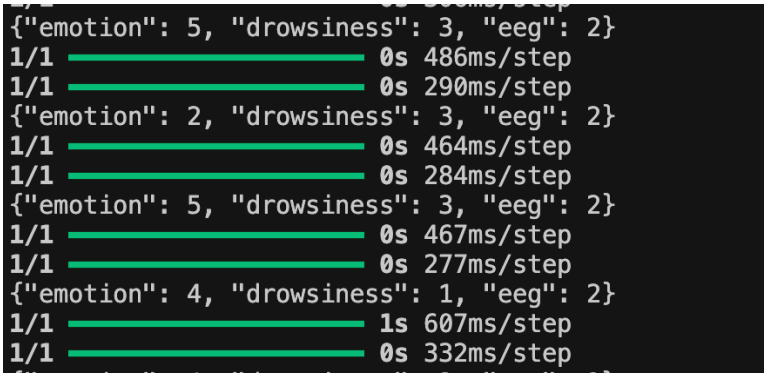
- python producer.py
- python consumer.py
- bin/flink run ./phase2_gradle/build/libs/Consumer.jar

**دقت شود نسخه flink 1.20 باشد.**

خروجی producer:




خروجی consumer:



اجرای job:


flink-1.20.0 bin/flink run /Users/mmm/IdeaProjects/phase2_gradle/build/libs/Consumer.jar
Job has been submitted with JobID f4f79003220e5dc16e441bc498772ae9