ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Sviluppo di un framework per l'analisi di segnali elettrofisiologici

Tesi di laurea in
Data Mining

Relatore
Prof. Matteo Golfarelli

Candidato
Luca Rossi

Quarta Sessione di Laurea
Anno Accademico 2022-2023

Indice

Elenco delle figure

Elenco dei listati

Capitolo 1

Introduzione

La tesi di laurea presentata propone lo sviluppo di un framework innovativo per l'analisi di dati di elettrofisiologia, con particolare attenzione ai file ABF. L'obiettivo principale della tesi è di fornire uno strumento in grado di semplificare e velocizzare l'analisi dei dati di elettrofisiologia, riducendo il tempo e lo sforzo richiesti per ottenere informazioni utili dai dati.

Tra le principali funzionalità del framework vi sono la visualizzazione, l'implementazione di filtri per la manipolazione delle tracce, la possibilità di eseguire analisi spettrali, la generazione di istogrammi per l'analisi delle distribuzioni di frequenza e la possibilità di restringere la regione di interesse per eseguire analisi mirate su porzioni specifiche dei dati.

Inoltre, il framework include una funzionalità di fitting di curve, che consente di modellare i dati di elettrofisiologia utilizzando una varietà di funzioni matematiche, tra cui le curve di Boltzmann e le funzioni esponenziali. Questa funzionalità permette di ottenere informazioni dettagliate sulle proprietà dei canali ionici e dei recettori presenti nelle cellule biologiche. Il framework è stato implementato in Python, sfruttando diverse librerie tra cui NumPy, SciPy e Matplotlib, per l'analisi dei dati e la visualizzazione dei risultati. Per dare maggior robustezza al software prodotto sono stati usati unit test e tecniche di Continuous Integration and Continuous Deployment (CI/CD). Inoltre, il framework è stato progettato per funzionare su diverse piattaforme, compresi sistemi Windows, Linux e macOS.

Nel capitolo capitolo 2 viene fornito un background sull'elettrofisiologia. Vengono presentati i principali concetti e le tecniche di registrazione dei segnali elettrofisiologici, con un focus sulle problematiche e le sfide dell'analisi di tali segnali.

Nel capitolo capitolo 3 viene descritta la raccolta dei requisiti per lo sviluppo del framework. In questa fase sono stati individuati i bisogni degli utenti finali e le funzionalità chiave del software, attraverso interviste, sondaggi e analisi di casi d'uso.

Nel capitolo capitolo 4 viene presentato lo sviluppo del sistema, con una panoramica dell'architettura del framework e delle tecnologie utilizzate per la sua implementazione. Vengono descritte le funzionalità principali del software, come l'acquisizione dei segnali, la loro elaborazione, la visualizzazione e l'analisi.

Nel capitolo capitolo 5 viene approfondita l'analisi dati avanzata, con una presentazione di alcune tecniche di elaborazione dei segnali elettrofisiologici utilizzate all'interno del framework. In particolare, vengono descritte tecniche di filtraggio, di analisi della coerenza, di decomposizione in componenti indipendenti e di analisi di connettività.

Nel capitolo capitolo 6 vengono presentate le conclusioni del lavoro di tesi, con un riassunto dei risultati ottenuti e delle prospettive future di sviluppo del framework.

In sintesi, la tesi si propone di presentare un framework innovativo e completo per l'analisi di segnali elettrofisiologici, che si pone come un'importante risorsa per la comunità scientifica interessata allo studio del sistema nervoso, del sequenziamento del DNA e delle sue funzioni.

Capitolo 2

Background

L'analisi dei dati di elettrofisiologia ha visto un notevole sviluppo negli ultimi anni grazie alla rapida evoluzione delle tecnologie di registrazione dei segnali, dei metodi di analisi e della strumentazione utilizzata. Attualmente, esistono diverse tecniche di registrazione e analisi dei dati di elettrofisiologia, ognuna con i propri vantaggi e limiti.

Una delle aziende più conosciute e famose in questo ambito è la "Molecular Devices", è attualmente considerata lo standard de facto nel suo ambito e produce strumenti e software per la ricerca biologica e farmaceutica.

2.1 Elettrofisiologia

L'elettrofisiologia [6] è una disciplina che si occupa di studiare l'attività elettrica delle cellule e dei tessuti biologici, in particolare delle cellule neuronali. L'elettrofisiologia è una delle principali tecniche sperimentali utilizzate in neuroscienze e fisiologia, ed è basata sulla registrazione di segnali elettrici prodotti dalle cellule, attraverso l'utilizzo di elettrodi.

Gli elettrodi possono essere inseriti all'interno di cellule singole o tessuti, oppure posizionati sulla superficie esterna dei tessuti biologici, per registrare l'attività elettrica. I segnali registrati possono essere utilizzati per studiare una vasta gamma di fenomeni biologici, tra cui il potenziale di membrana, l'attività ionica, la liberazione di neurotrasmettitori, la generazione di potenziali d'azione, l'attività sinaptica, e molte altre.

L'elettrofisiologia è utilizzata in molti campi di ricerca biomedica, tra cui neuroscienze, cardiologia, endocrinologia e oncologia. Le tecniche elettrofisiologiche sono anche utilizzate in medicina clinica per la diagnosi e il trattamento di diverse patologie, come l'epilessia, la malattia di Parkinson, la depressione e molte altre.

2.2 Tecniche

L'elettrofisiologia sta riscontrando una fase di forte sviluppo grazie all'avvento di tecniche nuove e rivoluzionarie che hanno permesso l'analisi di fenomeni prima invisibili. Tra le varie tecniche, le più famose, utilizzate, studiate e promettenti sono:

- Patch clamp [7]
- Lipid Bilayer
- Nanopori allo stato solido [4]

2.2.1 Patch clamp

Il patch clamp è una tecnica di elettrofisiologia utilizzata per misurare l'attività ionica di singole cellule, in particolare delle cellule neuronali. La tecnica del patch clamp è stata sviluppata negli anni '70 dal fisico tedesco Erwin Neher e dal biologo Bert Sakmann, e ha rappresentato una vera e propria rivoluzione nell'elettrofisiologia cellulare.

Il patch clamp prevede l'utilizzo di un elettrodo di vetro molto sottile, con una punta di circa 1 micron, che viene appoggiato sulla membrana cellulare. Questa punta forma una sorta di "sigillo" sulla membrana cellulare, creando uno spazio virtuale, o patch, all'interno del quale si può misurare l'attività ionica.

Il patch clamp permette di registrare l'attività elettrica di singole cellule, misurando la corrente elettrica che attraversa la membrana cellulare. In questo modo, è possibile studiare i meccanismi che regolano l'attività ionica, la generazione di potenziali d'azione, e molti altri fenomeni cellulari.

Il patch clamp è una tecnica molto sensibile e precisa, che consente di misurare correnti elettriche dell'ordine dei picoampere o, nei sistemi più avanzati anche di femtoampere. Il patch clamp viene utilizzato in molti campi di ricerca biomedica, tra cui neuroscienze, farmacologia, e fisiologia, e ha permesso di fare importanti scoperte sulla funzione e il funzionamento delle cellule neuronali.

2.2.2 Lipid Bilayer

La tecnica della "Lipid Bilayer" è una tecnica di elettrofisiologia che viene utilizzata per studiare la funzione dei canali ionici e delle proteine di membrana. In questa tecnica, viene creato un doppio strato di fosfolipidi, o "lipid bilayer", che riproduce le proprietà strutturali della membrana cellulare.

La tecnica della "Lipid Bilayer" prevede l'utilizzo di un'apposita camera, divisa in due compartimenti, che sono separati da un sottile film di fosfolipidi. All'interno

di ciascun compartimento viene inserito un elettrodo, che permette di applicare una differenza di potenziale tra i due lati della membrana.

La tecnica della "Lipid Bilayer" è una tecnica molto sensibile e precisa, che consente di studiare le proprietà delle proteine di membrana con una risoluzione molto elevata. Tuttavia, la tecnica richiede una certa esperienza e abilità da parte dell'utilizzatore, in quanto la creazione del doppio strato di fosfolipidi può essere un processo delicato e complesso.

2.2.3 Nanopori allo stato solido

I nanopori allo stato solido sono una tecnologia emergente nell'ambito dell'elettrofisiologia, che consente di analizzare singole molecole e ioni a livello di singola cellula. In particolare, i nanopori allo stato solido sono utilizzati per creare una sorta di "foro" attraverso una membrana solida, che permette il passaggio di singole molecole o ioni.

La tecnologia dei nanopori allo stato solido prevede l'utilizzo di un materiale solido, come il silicio o l'ossido di alluminio, che viene sottoposto a un processo di litografia per creare un canale nanostrutturato. Questo canale viene poi inserito all'interno di una membrana lipidica, creando un nanoporo che permette il passaggio di singole molecole o ioni.

In questo modo, i nanopori allo stato solido consentono di analizzare il comportamento di singole molecole o ioni in una soluzione elettrolitica, misurando la corrente elettrica che lo attraversa. Ad esempio, è possibile utilizzare questa tecnologia per studiare la dinamica delle proteine, l'interazione tra proteine e farmaci, o per fare analisi su DNA.

I nanopori allo stato solido presentano numerosi vantaggi rispetto alle altre tecniche di elettrofisiologia. In particolare, consentono di analizzare singole molecole o ioni, il che consente di ottenere informazioni più dettagliate e precise sulla loro funzione. Inoltre, i nanopori allo stato solido sono molto sensibili e consentono di analizzare molecole e ioni a bassa concentrazione.

2.2.4 Analisi

In elettrofisiologia possono essere fatti diversi tipi di analisi, tra cui:

- Analisi Gap-Free
- Analisi Episodiche
- Analisi Online
- Analisi Batch

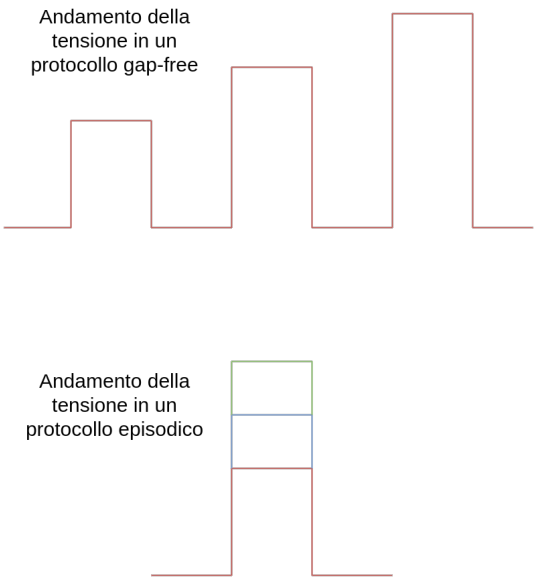

Figura 2.1: Differenze tra analisi episodiche e gap-free

Gap-Free L'analisi gap-free si basa sulla registrazione continua del segnale elettrofisiologico, senza interruzioni. Questo tipo di analisi è utile per studiare la dinamica temporale delle attività elettrofisiologiche, come ad esempio la registrazione di potenziali d'azione, sinapsi, attività di campo locale e attività di oscillazione. Inoltre, l'analisi gap-free permette di analizzare le correlazioni temporali tra le varie attività, per esempio per studiare le interazioni tra le cellule nervose.

Analisi Episodiche L'analisi episodica si basa sulla registrazione di brevi episodi di attività elettrofisiologica, ad esempio stimoli elettrici o potenziali di azione, registrati in modo sincrono con la presentazione degli stimoli. Questo tipo di analisi è utile per studiare la risposta dell'organismo a stimoli specifici e per valutare le proprietà dell'attività elettrofisiologica, come ad esempio la soglia di attivazione neuronale, la durata e la latenza del potenziale d'azione.

Questi approcci sono fondamentali per l'analisi dei dati elettrofisiologici nell'ambito della neuroscienza, che possono essere utilizzati in modo complementare per studiare le proprietà e la dinamica dell'attività elettrofisiologica in vivo e in vitro.

In figura 2.1 può essere esplorato un esempio di protocollo di analisi in modalità gap-free o episodica. La differenza sostanziale sta nella visualizzazione dei dati e nella fruibilità da parte degli scienziati.

Analisi Online L'analisi online (o in tempo reale) si riferisce all'elaborazione dei dati elettrofisiologici in tempo reale, durante l'acquisizione stessa. Questo approccio consente di visualizzare i dati elettrofisiologici in tempo reale e di effettuare una prima analisi dei dati mentre l'acquisizione è in corso. Ciò può essere particolarmente utile per la selezione dei dati, per identificare eventuali problemi durante l'acquisizione e per adattare i parametri di acquisizione in tempo reale.

Analisi Batch L'analisi batch (o offline) si riferisce all'elaborazione dei dati dopo che l'acquisizione è stata completata. In questo caso, l'intera registrazione viene elaborata successivamente utilizzando specifici software di analisi dei dati. Questo approccio consente di effettuare un'analisi più completa e accurata dei dati, utilizzando algoritmi più complessi e tempi di calcolo più lunghi.

L'analisi batch viene spesso utilizzata per l'elaborazione di dati complessi e per la generazione di modelli matematici che possano descrivere l'attività neuronale in modo preciso. Ad esempio, l'analisi batch può essere utilizzata per identificare i pattern di attività neuronale associati a specifiche funzioni cognitive o per studiare le modifiche dell'attività neuronale in condizioni patologiche.

In ogni caso, sia l'analisi online che quella batch sono importanti per l'elettrofisiologia e devono essere utilizzate in modo complementare per una comprensione completa dell'attività elettrica delle cellule e dei tessuti biologici.

2.3 Presentazione del progetto

L'azienda che mi ha accompagnato nello sviluppo di questo framework e che ha proposto questo progetto è la **Elements srl.**, un'azienda italiana specializzata nello sviluppo di soluzioni hardware e software per l'analisi di dati di elettrofisiologia. Elements è diventata rapidamente un'azienda leader nel campo dell'elettrofisiologia, fornendo soluzioni innovative e personalizzate per la ricerca accademica e industriale, insieme a servizi di consulenza e formazione per aiutare i ricercatori a sfruttare al massimo le potenzialità dei propri dispositivi e delle tecniche di elettrofisiologia in generale. L'azienda ha anche sviluppato una forte collaborazione con numerosi istituti di ricerca in tutto il mondo, lavorando a stretto contatto con i ricercatori per comprendere le loro esigenze specifiche e sviluppare soluzioni personalizzate per soddisfarle.

Uno dei software più usati e importanti all'interno dell'azienda che permette di fare analisi batch è L'EDR4 (Elements Data Reader), mentre EDA (Elements Data Analyzer), viene usato per fare analisi batch.

Il software che prenderemo in considerazione è EDA e tra le funzionalità offerte ci sono:

- Visualizzazione delle tracce attuali di corrente e tensione (con pan, zoom e ritaglio di sezioni della traccia)
- Possibilità di usare filtri digitali per modificare segnali
- Esportazione dei segnali in altri formati
- Creazione di istogrammi
- Analisi spettrale

Il software tuttavia iniziava ad essere datato e difficile da mantenere, inoltre si voleva cercare di creare una community attiva di scienziati che facessero crescere il progetto. Per questi motivi si voleva creare una nuova versione, aggiornata e più utilizzabile di quella attuale. Nel capitolo 3 andremo ad analizzare i requisiti del nuovo sistema.

Capitolo 3

Raccolta dei requisiti

La raccolta dei requisiti è avvenuta tramite intervista con il responsabile del dipartimento IT, con cui sono stati schematizzati ad alto livello i requisiti funzionali e alcuni dei requisiti non funzionali.

3.1 Requisiti Funzionali

Come già accennato l'applicazione da costruire doveva permettere di eseguire *analisi batch* su dati già registrati. Tra le funzionalità di base da implementare c'erano:

- Visualizzazione dei dati
- Filtri digitali
- Istogrammi
- Analisi spettrale
- Storico delle operazioni eseguite sui dati con eventuale possibilità di passare da una versione a quella precedente o successiva
- Esportazione del segnale modificato in altri formati (esempio csv)
- Lettura di file di tipo abf (Axon Binary Format) nella versione 2.0
- Creazione di regioni di interesse specifiche (a partire dalla traccia intera)
- Possibilità di generare la funzione che più si avvicina a una porzione di dati (**fitting**)

Il requisito di lettura di dati di tipo abf nasce da motivi legacy e di interoperabilità con i dispositivi Axon. Essendo questa compagnia il leader storico per la strumentazione elettrofisiologica la clientela vuole poter analizzare i dati raccolti in precedenza o registrarne di nuovi su cui indagare con software terzi. Purtroppo però l'abf è una tipologia di file proprietario, la cui struttura è nota ma il supporto agli sviluppatori per leggere file di questo tipo è scarso, infatti, l'unico strumento ufficiale (fornito dall'azienda) per fare operazioni su questo dato è una dll [1].

3.2 Requisiti Non funzionali

Uno dei requisiti non funzionali dell'applicazione era l'**open source**. Per facilitare l'accesso a nuovi utenti una delle tattiche scelte è stata quella della produzione di un programma totalmente open e gratuito, al quale chiunque potesse lavorare e dare un contributo. Questa scelta avrebbe anche potuto aiutare con:

- L'espandibilità delle funzionalità dell'applicazione (sia con contributi concreti da parte del pubblico, che con richieste specifiche che sarebbero potute emergere)
- La correzione di bug
- La distribuzione del software

Il software dovrà tenere in considerazione anche l'**esperienza utente**, cercando di renderla gradevole tramite un'interfaccia utente moderna e facendo in modo di consentire analisi in tempi ragionevolmente brevi, ad esempio cercando di non avere tempi di attesa superiori ai secondi (per il completamento di operazioni comuni come caricamento di file e/o filtraggio di dati) su pc/laptop di fascia media prodotti negli ultimi 5 anni.

Un processo molto utile che permette di tenere il codice manutenibile e accessibile è quello conosciuto col nome di **CI/CD**. Per lo sviluppo di questo progetto è stato sfruttato questo approccio per tenere il codice sempre online assieme ad una sua versione installabile su desktop. Le tecniche di CI/CD raggiungono però l'apice della loro utilità solo se affiancate ad un utilizzo estensivo di test. Per questo motivo, e per dare più robustezza al programma finale, si è deciso di corredare il codice sorgente con quanti più **test** possibile, che sarebbero serviti in qualche modo sia da documentazione, sia da strumento di debug. A questa pratica è stato

[1] DLL è l'acronimo di "Dynamic Link Library", ovvero una libreria di collegamento dinamico. Le DLL sono molto utilizzate in ambiente Windows, dove sono spesso utilizzate per fornire funzionalità aggiuntive a programmi esistenti, come ad esempio i driver di periferiche hardware, librerie grafiche e audio, e altri tipi di plug-in.

ovviamente affiancato un processo di testing automatico per garantire che il codice scritto funzionasse in modo corretto (o quantomeno per escludere errori ovvi o banali) e soprattutto che continuasse a funzionare in maniera corretta con l'agiunta di codice e funzioni.

Un altro requisito da tenere a mente per la creazione di una community attiva è il pubblico a cui ci si rivolge, in questo caso non si sta facendo un framework web o un pacchetto da importare in java, bensì uno strumento che verrà utilizzato principalmente da scienziati che spesso non hanno familiarità con linguaggi compilati, con programmi lunghi e complessi, e prediligono linguaggi di scripting come python e matlab. Per questo motivo, pur non avendo alcun vincolo sul linguaggio da utilizzare e sulle tecnologie da sfruttare, sarebbe stato opportuno tenere in considerazione questi aspetti quando sarebbe iniziata l'implementazione.

Essendo il tempo da dedicare al progetto limitato e dovendo lavorare da solo al programma, si è deciso di utilizzare un approccio incrementale allo sviluppo software, per cui di settimana in settimana ci saremmo dati nuovi obiettivi da raggiungere come l'implementazione di nuove feature o la correzione/modifica di elementi già sviluppati, avendo come obiettivo finale e ideale, lo sviluppo di tutte quante le funzionalità sopra descritte.

Capitolo 4

Sviluppo del sistema

In questo capitolo, esploreremo in dettaglio le varie fasi del processo di sviluppo del sistema software, evidenziando le sfide e le best practice che sono state utilizzate per la realizzazione del progetto. Per la precisione verranno trattati i seguenti argomenti:

- Scelte tecnologiche
- Design del sistema
- Implementazione
- Risultato finale

4.1 Scelte tecnologiche

Tra le scelte tecnologiche degne di nota vanno menzionate quelle riguardo i seguenti aspetti:

- Linguaggio di programmazione
- Sistema di versionamento
- CI/CD

4.1.1 Linguaggio di programmazione

La prima scelta importante che è stata presa all'inizio del progetto è stata quella del linguaggio di programmazione. Pur non avendo vincoli sul linguaggio da utilizzare bisognava comunque tener conto di molteplici aspetti. Come già detto nel capitolo 3 uno dei principali obiettivi a lungo termine di questo progetto è la

creazione di una community forte che facesse crescere e arricchisse questo progetto tramite l'aggiunta di feature, bug fix e ottimizzazioni. Un altro requisito da tenere a mente era la quantità di tempo a disposizione, abbastanza limitata e per questo sarebbe stato utile un linguaggio di programmazione di alto livello che permettesse di implementare codice testabile e funzionante senza dover perdere tempo con costrutti di basso livello. Sarebbe stato utile avere un sistema con supporto nativo al multipiattaforma (write once run everywhere) anche se in ambito scientifico il sistema operativo principalmente utilizzato è **Windows**, e probabilmente il supporto per **Linux** e **MacOS** sarebbe stato quasi inutilizzato. Un altro aspetto di interesse per lo sviluppo dell'aplicazione sarebbe stata la disponibilità di pacchetti e librerie, che avrebbero permesso uno sviluppo molto più rapido e "sicuro" (si spera che i pacchetti disponibili per un certo linguaggio di programmazione siano stati testati e che all'occorrenza di bug vengano effettuate correzioni dal team di sviluppo del pacchetto).

Fatte queste premesse il linguaggio su cui è ricaduta la scelta è stato **Python**.

Python è un linguaggio di programmazione di alto livello, interpretato, dinamicamente tipizzato e orientato agli oggetti. È ampiamente utilizzato in diverse applicazioni, tra cui lo sviluppo web, l'intelligenza artificiale, la data science e l'automazione.

Alcuni dei vantaggi che offre sono:

- Sintassi semplice e leggibile: ha una sintassi semplice e leggibile che lo rende facile da imparare e comprendere.
- Ampia gamma di librerie: offre una vasta gamma di librerie di terze parti che semplificano il lavoro di sviluppo di applicazioni e la gestione dei dati.
- Multi-piattaforma: è compatibile con diverse piattaforme, tra cui Windows, Linux e macOS.
- Grande comunità di sviluppatori: ha una grande comunità di sviluppatori che contribuiscono alla sua crescita e alla creazione di nuove librerie e strumenti.
- Adatto per l'intelligenza artificiale e la data science: è molto popolare nell'ambito dell'intelligenza artificiale e della data science grazie alle librerie come NumPy, Pandas e Scikit-learn. Queste sue peculiarità lo rendono particolarmente interessante per lo sviluppo di un framework per l'analisi dati.

Tuttavia questo linguaggio non è esente da svantaggi:

- Velocità di esecuzione: è un linguaggio interpretato, il che significa che le prestazioni del codice possono essere inferiori rispetto ai linguaggi compilati come C o C++.

- Gestione della memoria: utilizza un sistema di gestione della memoria automatico, il che può portare ad un consumo eccessivo di memoria.
- Non adatto per alcune applicazioni: potrebbe non essere adatto per alcune applicazioni che richiedono una velocità di esecuzione particolarmente elevata, come ad esempio i videogiochi.
- Tipizzazione dinamica: la tipizzazione dinamica di Python è un'arma a doppio taglio, da una parte fornisce estrema flessibiltià al programmatore, dall'altra potrebbe introdurre bug difficili da trovare e risolvere, anche se questo problema può essere mitigato con del testing mirato.

4.2 Design del sistema

Il software è stato strutturato come un'applicazione Desktop, cercando di tenere separate logica e visualizzazione.

4.2.1 Gestione dei dati

Una delle classi che stanno alla base dell'applicazione è quella denominata **MetaData**, che contiene dei **DataGroup** composti a loro volta da **BasicData**. Il diagramma in figura 4.1 ne mostra la struttura.

Questa serie di strutture dati è tutto ciò che serve per immagazzinare i dati letti da file. In particolare questa struttura ad albero ci permette di applicare trasformazioni e navigare tutti i nodi già esplorati.

Si passi ora ad un'analisi più dettagliata.

MetaData È la classe che ha accesso ai dati interni dell'applicazione. C'è una sola istanza di questa classe, ed è l'entità dalla quale si deve passare per fare operazioni CRUD sui dati. In particolare essa ha un'istanza del DataGroup correntemente visualizzato e un Set orinato (OrderedSet) di DataGroup. In particolare il set ordinato di DataGroup è la struttura dati che permette di aprire più file contemporaneamente.

DataGroup È la struttura utilizzata per modellare dei dati che hanno qualcosa in comune, ovvero, tempo, unità di misura e altro. Spesso una misurazione consiste nella registrazione di uno o più canali di tensione e/o uno o più canali di corrente. Queste misurazioni ovviamente sono tutte state fatte nel medesimo intervallo di tempo, tutti i canali di corrente avranno la stessa unità di misura e lo stesso vale per tensione e tempo. Per questo motivo è stata creata questa classe che ha come

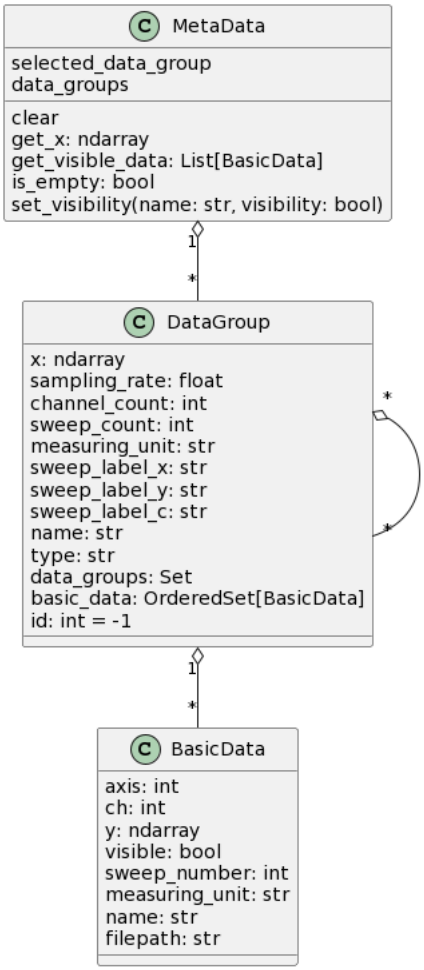

Figura 4.1: Diagramma delle classi: MetaData

compito principale quello di immagazzinare tutti i BasicData con informazioni comuni e detiene anche un set ordinato di altri Datagroup. Questo rende possibile la modifica e l'applicazione di trasformazioni sui dati (come ad esempio filtri, istogrammi, ecc).

BasicData È la rappresentazione di un singolo segnale. Questo è il tipo di dato che viene utilizzato per generare i grafici e andrà a racchiudere le versioni modificate dei dati originali.

Tramite queste tre classi si riescono a rappresentare tutti i dati di interesse.

4.2.2 Logica

La logica del sistema è tutta incapsulata all'interno della classe **Logics**, della quale si può avere una schematizzazione tramite il diagramma delle classi in figura 4.2.

Tramite questa classe possono essere eseguite operazioni come:

- Aperire/esportare file
- Creare filtri digitali
- Fare l'analisi spettrale
- Creare istogrammi
- Modificare regioni di interesse specifiche
- Trovare la curva che più si avvicina a una porzione di dati ed esportare i parametri delle curve trovate

Questa classe è divisa in moduli composti principalmente da funzioni, questo per modellare il problema in modo il più possibile funzionale e tenere tutte le varie aree di interesse ben separate.

4.2.3 GUI

La View ha un solo riferimento alla logica e non fa altro che richiamare le sue funzioni per ottenere i cambiamenti di stato da mostrare all'utente.

PyQt L'interfaccia utente è stata implementata tramite PyQt, un insieme di librerie di binding Python per il toolkit di interfacce grafiche Qt. Qt è un framework per lo sviluppo di applicazioni cross-platform, utilizzabile in diversi linguaggi di programmazione, tra cui C++, Python, Java e altri, è uno dei binding Python più popolari per Qt, insieme a PySide.

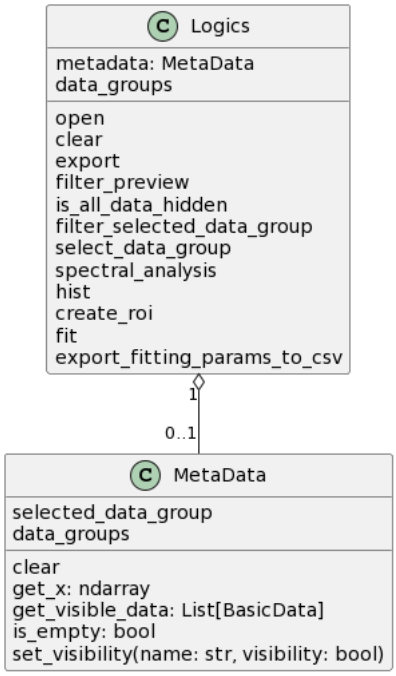

Figura 4.2: Diagramma delle classi: Logics

Con PyQt si possono creare applicazioni desktop cross-platform con un'interfaccia utente moderna e professionale. PyQt fornisce un'interfaccia Python per l'accesso alle funzionalità di Qt, compresi widget, layout, segnali e slot, e molti altri strumenti per la creazione di interfacce grafiche interattive e responsive.

PyQt supporta diverse piattaforme, tra cui Windows, Linux, macOS e alcune piattaforme mobile. Inoltre, PyQt è una libreria open source distribuita sotto la licenza GPL, il che significa che è gratuita e può essere utilizzata liberamente anche per lo sviluppo di applicazioni commerciali.

4.3 Implementazione

Come detto in precedenza l'implementazione del software è avvenuta in maniera incrementale e iterativa. Dopo aver concordato l'attività da svolgere con il responsabile del reparto IT si procedeva all'implementazione della feature. Ogni feature sviluppata era corredata da vari test che ne mostravano il corretto funzionamento e fornivano anche una linea guida di utilizzo. Al completamento della feature il responsabile dava l'approvazione o chiedeva di applicare delle modifiche correttive, se la feature risultava completa, si passava a quella successiva.

4.3.1 Manipolazione Dei Dati

NumPy [3] Tra le librerie utilizzate per l'implementazione del progetto ovviamente non poteva mancare NumPy. NumPy è una delle librerie più popolari in Python per il calcolo scientifico, fornisce supporto per array multidimensionali, matrici e funzioni matematiche.

NumPy fornisce un'interfaccia semplice ed efficiente per la manipolazione di grandi quantità di dati, fornendo funzioni per la creazione, l'accesso e la manipolazione di array multidimensionali in modo efficiente. La libreria fornisce anche funzioni per l'algebra lineare, la trasformata di Fourier e molte altre operazioni matematiche.

In NumPy, gli array sono oggetti di tipo ndarray, che rappresentano array multidimensionali di elementi omogenei.

NumPy è una libreria open source distribuita sotto la licenza BSD, quindi è gratuita e può essere utilizzata per lo sviluppo di applicazioni commerciali.

PyAbf [2] PyABF è una libreria Python open-source progettata per lavorare con i dati acquisiti da strumenti di acquisizione elettrica di segnali, in particolare i file in formato ABF generati da dispositivi di acquisizione dati della Axon Instruments.

PyABF permette di aprire, leggere, analizzare e visualizzare i dati acquisiti da file ABF. Ciò significa che, una volta importato un file, è possibile accedere a tutte le informazioni in esso contenute, come i dati di traccia, i parametri di acquisizione, i dati del protocollo sperimentale e così via.

PyABF è progettato per essere facile da usare, anche per chi ha poca esperienza di programmazione. La libreria è ben documentata e include numerosi esempi di codice, che illustrano come utilizzare le diverse funzioni. Inoltre, PyABF è attivamente sviluppato e mantenuto da una comunità di sviluppatori open-source, il che significa che la libreria è costantemente migliorata e aggiornata [1].

scipy [10] SciPy è una libreria open-source molto potente e flessibile, che fornisce una vasta gamma di funzioni e strumenti per la computazione scientifica e l'analisi dei dati. È una delle librerie Python più utilizzate in campo scientifico e ingegneristico.

Si basa su NumPy e fornisce funzionalità aggiuntive per l'elaborazione dei dati, l'analisi statistica, l'ottimizzazione, l'interpolazione, la risoluzione di equazioni differenziali, l'analisi dei segnali e molto altro.

SciPy è anche estremamente personalizzabile e configurabile, grazie alla disponibilità di numerose opzioni di configurazione e delle funzioni.

[1] Durante lo sviluppo dell'applicazione ci si è imbattuti in un bug della libreria, con il responsabile del progetto è stato studiato il problema e sono state fatte alcune modifiche. Successivamente è stata fatta una pull-request e ora il codice aggiornato e funzionante è online e disponibile.

Il suo utilizzo è stato fondamentale per l'implementazione di filtri personalizzati, analisi spettrale, per la creazione di istogrammi e per la funzionalità di fitting delle curve sui dati.

4.3.2 Visualizzazione dei plot

La visualizzazione dei plot è uno degli aspetti core del software. La rappresentazione delle varie linee deve essere chiara e l'esplorazione dei dati deve avvenire in maniera veloce ed intuitiva. Il grafico deve poter rappresentare in maniera veritiera tutti i punti della traccia registrata, il che, come vedremo nella sezione relativa al **plot simplifier** 4.3.3 non è un compito banale come potrebbe sembrare visto che spesso ci si ritrova a lavorare con grosse moli di dati ed essendo Python un linguaggio notoriamente lento, a volte si potrebbe far fatica a disegnare tracce con migliaia di punti. Ad esempio uno degli accorgimenti che ha aiutato nella visualizzazione è stata la larghezza della linea con cui vengono disegnati i grafici. Già soltanto disegnando linee più sottili si è riscontrato un notevole miglioramento delle performance a livello di pan/zoom e caricamento di nuovi grafici, soprattutto quando si trattava di disegnare grafici con molti punti. Tuttavia questo accorgimento non è risultato sufficiente per una navigazione agevole di qualsiasi grafico e per questo è stato creato il *plot simplifier*, una classe che gestisce il comportamento da utilizzare con tracce eccessivamente lunghe.

matplotlib [5] Matplotlib è una delle librerie di visualizzazione di dati più popolari in Python, progettata per creare grafici e plot di alta qualità in modo semplice ed efficiente.

Matplotlib è altamente personalizzabile, il che significa che gli utenti possono controllare ogni aspetto del loro grafico, compresa la dimensione, il tipo di linea, la legenda e i colori. La libreria supporta diversi tipi di grafici, tra cui: istogrammi, grafici a dispersione, grafici a barre, grafici a torta, grafici a linea e molti altri. Inoltre, Matplotlib è altamente compatibile con altri strumenti di analisi dei dati in Python, come NumPy e Pandas.

Matplotlib è una libreria open source distribuita sotto la licenza BSD, e può essere utilizzato per lo sviluppo di applicazioni open source e commerciali.

4.3.3 Visualizzazione di plot densi

Un esperimento solitamente consiste nella lettura a una determinata frequenza (che va dalle centinaia di kHz alla decina di MHz) di dati collezionati mediante un dispositivo collegato tramite seriale. Considerando che un dispositivo ha più di un canale di corrente e potenzialmente più di un canale di tensione. Il throughput in

breve tempo diventa abbastanza elevato da permettere la scrittura di GB di dati in pochi minuti.

Solitamente questi dati vengono registrati come abf e gli abf non possono essere più grandi di 2 GB.

Solitamente gli esperimenti durano abbastanza poco da permettere una rappresentazione completa del dataset ma può capitare di avere migliaia o decine di migliaia di punti da disegnare. Questo aspetto potrebbe non creare particolari problemi in alcuni ambienti di sviluppo, tuttavia Python è famoso per essere drammaticamente lento e disegnare così tanti punti porta a rallentamenti non tollerabili nell'utilizzo di tutti i giorni. A questo punto ci sono diversi modi per mitigare questo problema, quelli presi in analisi sono la rasterizzazione e una sorta di compressione dei dati disegnati.

Rasterizzazione La rasterizzazione è il processo di conversione di immagini o oggetti vettoriali in una griglia di pixel (o raster) per renderli visualizzabili su un dispositivo di visualizzazione, come un monitor o una stampante. Durante la rasterizzazione, l'immagine o l'oggetto viene suddiviso in una griglia di pixel, ognuno dei quali viene colorato in base alle informazioni di colore dell'originale.

La rasterizzazione è spesso utilizzata per convertire immagini o oggetti vettoriali in formati che possono essere utilizzati per la stampa, l'editing di immagini o la visualizzazione su schermo. Tuttavia, poiché la rasterizzazione può portare a una perdita di dettagli e qualità dell'immagine, è importante scegliere la risoluzione appropriata per il dispositivo di destinazione. Inoltre, alcune immagini o oggetti possono non essere adatti per la rasterizzazione se contengono dettagli molto complessi o linee sottili che potrebbero essere sfocate o persi nel processo di rasterizzazione.

Questo metodo compare anche direttamente nella documentazione di matplotlib, tuttavia questa tecnica non sembrava fare al caso nostro, dato che spesso quello che un utente vuole fare non è aprire un immagine e fissarla, quanto invece zoomare, spostarsi, tornare indietro cercare di nuovo e così via. Per questo motivo temo che la rasterizzazione fosse inadeguata per il nostro caso d'uso. Inoltre la documentazione in materia non era molto facilmente reperibile e la mancanza di esempi avrebbe comportato un dispendio di tempo non indifferente.

Compressione senza perdita Il metodo che si è scelto di utilizzare invece consiste in una sorta di compressione di dati senza perdita. Questa tecnica si basa su un concetto banale che però se mal gestito comporta un dispendio elevato di potenza di calcolo. La parte pesante dal punto di vista computazionale è il rendering dell'immagine da visualizzare su schermo, questa complessità è data dal fatto che per ogni punto da disegnare bisogna calcolare i pixel da accendere su schermo,

quindi, più punti bisogna disegnare più i tempi si dilatano. La **soluzione naïve** è quella di prendere un punto ogni n per disegnare un sottoinsieme dei punti di partenza, scegliendo arbitrariamente un sistema di campionamento si può decidere quanta informazione perdere per favorire la velocità del sistema. Tuttavia, questo tipo di rappresentazione comporterebbe una snaturazione piuttosto evidente del grafico di partenza, rendendo difficile da parte degli utilizzatori la ricerca di eventi specifici che potrebbero essere notati a colpo d'occhio in una traccia "piena". La domanda che sorge spontanea è: si potrebbe utilizzare questo trucchetto per avere un grafico più facile da navigare senza compromettere la visualizzazione dei dati? La **soluzione più intelligente** sfrutta il fatto che la visualizzazione di dati su uno schermo è strettamente vincolata dalla risoluzione dello schermo. Se si vuole visualizzare una traccia da 100.000 (centomila) punti nella sua interezza dovremmo avere a nostra disposizione 100.000 punti diversi dove poterli disegnare, tuttavia, gli attuali monitor di fascia alta hanno una risoluzione di 4 o 8k (8192x4320 pixel), che si traducono in al più 8192 valori distinti nell'asse delle ascisse. Questo significa che i nostri 100.000 punti verranno disegnati tutti quanti in un sottoinsieme di questi 8192 pixel, dato che il nostro gragico non occuperà la totalità dello schermo, infatti sarà presente anche l'interfaccia utente con al suo interno vari controlli, widget e form).Comunque, anche se usassimo tutti quanti i punti dello schermo disegneremmo all'incirca

$$\frac{100.000}{8.000} \simeq 13$$

punti per ogni pixel, quindi, per ogni "colonna di pixel" verranno disegnati 13 punti, che poi il motore grafico disegnerà con n linee che li collegano, ma che saranno percepite come un'unica linea che va dal valore più piccolo a quello più grande. In figura 4.3 si ha una rappresentazione grafica di quello che succede.

Per alleviare questo problema, quindi, ci basta trasformare i dati originali eliminando tutti i valori non necessari per una corretta visualizzazione. Riprendendo i dati di prima, si supponga di avere un monitor 8k (res: 7680 X 4320) e una traccia con 100.000 valori. Per ottenere il risultato voluto dovremo dividere la traccia in chunk da $\sim$13 valori l'uno e di questi $\sim$13 valori prendere il massimo e il minimo. In questo modo la libreria dovrà renderizzare molti meno valori e l'esperienza utente risulterà molto più fluida e gradevole, tuttavia bisogna comunque ricordarsi di ricampionare i valori da rappresentare ad ogni zoom/pan sul plot.

4.3.4 Filtri digitali

I filtri digitali sono strumenti utilizzati nell'elaborazione digitale del segnale per rimuovere determinate componenti di frequenza da un segnale (tipicamente audio o video). I filtri digitali sono spesso usati nell'ambito dell'elaborazione audio, in

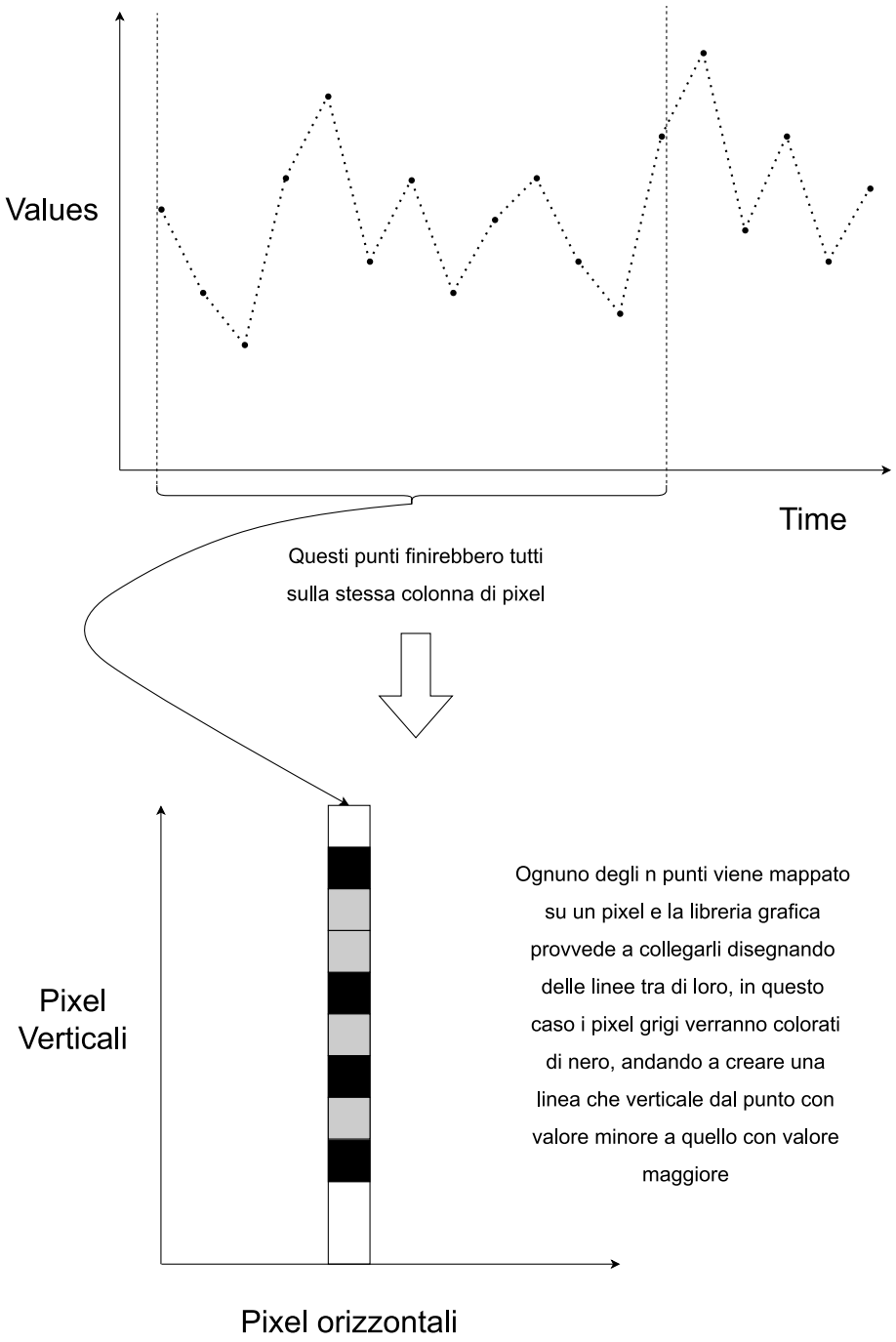

Figura 4.3: Immagine che mostra come n punti che in seguito alla renderizzazione avranno tutti la stessa x

applicazioni come la cancellazione del rumore, la separazione delle tracce audio, l'equalizzazione e la modifica delle caratteristiche tonali.

Ci sono due tipi principali di filtri digitali: i filtri FIR e i filtri IIR. I filtri FIR (Finite Impulse Response) utilizzano un insieme di coefficienti finiti per determinare la risposta del filtro, mentre i filtri IIR (Infinite Impulse Response) utilizzano feedback per determinare la risposta del filtro. I filtri FIR sono generalmente più facili da progettare e garantiscono una risposta in frequenza lineare, ma richiedono più memoria e potenza di elaborazione. I filtri IIR possono essere più efficienti in termini di memoria e potenza di elaborazione, ma possono essere più difficili da progettare e possono avere problemi di stabilità. In generale le due tipologie di filtri hanno le seguenti differenze:

- Risposta all'impulso: la risposta all'impulso di un filtro FIR è finita nel tempo, mentre la risposta all'impulso di un filtro IIR è infinita nel tempo. Questo significa che un filtro FIR ha un ritardo fisso e un tempo di stabilizzazione definito, mentre un filtro IIR può avere un ritardo variabile e può richiedere un tempo di stabilizzazione più lungo.

- Linearità di fase: i filtri FIR hanno una risposta in fase lineare in tutta la banda di passaggio, mentre i filtri IIR non hanno una risposta in fase lineare. Ciò significa che l'utilizzo di un filtro IIR può alterare la forma d'onda del segnale in ingresso.

- Stabilità: i filtri FIR sono sempre stabili, mentre i filtri IIR possono essere instabili se i coefficienti del filtro sono scelti in modo improprio. La stabilità del filtro IIR dipende anche dalla frequenza di campionamento del segnale di ingresso.

- Progettazione: la progettazione di un filtro FIR è relativamente semplice e prevedibile, poiché la risposta in frequenza del filtro è determinata esclusivamente dai suoi coefficienti. La progettazione di un filtro IIR è più complessa, poiché la risposta in frequenza del filtro dipende sia dai suoi coefficienti che dalla loro posizione polare nel piano complesso.

- Implementazione: i filtri FIR possono essere implementati in modo efficiente utilizzando tecniche di convoluzione lineare, mentre i filtri IIR richiedono solitamente l'utilizzo di tecniche di convoluzione circolare o di fattorizzazione del polinomio per garantire una maggiore efficienza.

I filtri digitali possono essere caratterizzati in base alla loro risposta in frequenza. I filtri passa-basso eliminano le componenti ad alta frequenza di un segnale, i filtri passa-alto eliminano le componenti a bassa frequenza, i filtri passa-banda eliminano le componenti al di fuori di un determinato intervallo di frequenza, e i filtri

elimina-banda eliminano le componenti all'interno di un determinato intervallo di frequenza.

La progettazione di un filtro digitale può essere realizzata tramite diversi metodi, tra cui il metodo del polinomio di Chebyshev, il metodo del polinomio di Butterworth, o attraverso l'uso di software dedicati per la progettazione dei filtri digitali.

Per l'applicazione che andremo a sviluppare si è deciso di utilizzare filtri IIR dato che consentono di avere un ordine più basso per raggiungere le stesse specifiche di attenuazione e di risposta in frequenza della controparte implementata con un filtro FIR, inoltre i filtri IIR sono più flessibili e modificabili e si adattano meglio al segnale in ingresso. Si è deciso di fornire all'utente completa libertà per la creazione di filtri, di tre diverse categorie: butterworth, bessel e cheby1. Quindi l'utente potrà personalizzare il filtro da applicare, guardarne la preview e conseguentemente applicare il filtro al segnale di partenza per trasformarlo e raffinarlo.

4.3.5 Istogramma

Un istogramma è una rappresentazione grafica della distribuzione di frequenza di un insieme di dati. In pratica, l'istogramma divide l'intervallo di valori possibili dei dati in un certo numero di intervalli e conta il numero di dati che ricadono in ciascuno di essi. Questo conteggio viene quindi rappresentato su un grafico a barre in cui l'asse orizzontale rappresenta gli intervalli e l'asse verticale rappresenta il numero di dati che ricadono in ciascuno di essi.

L'istogramma può essere utilizzato per analizzare diversi tipi di segnali. Ad esempio, in un segnale audio, l'istogramma può essere utilizzato per visualizzare la distribuzione di ampiezza delle forme d'onda del segnale. In un'immagine digitale, l'istogramma può essere utilizzato per visualizzare la distribuzione dei livelli di luminosità o di colore dei pixel dell'immagine. Ovviamente vale altrettanto per segnali di corrente.

L'analisi dell'istogramma può fornire informazioni utili sulla distribuzione del segnale. Ad esempio, può essere utilizzato per identificare i picchi di frequenza o i valori di ampiezza più comuni del segnale. Inoltre, può essere utilizzato per valutare l'omogeneità o l'eterogeneità del segnale. Ad esempio, in un'immagine digitale, un istogramma con una distribuzione uniforme dei livelli di luminosità può indicare una presenza di rumore o una mancanza di dettaglio dell'immagine, mentre un istogramma con una distribuzione concentrata può indicare la presenza di una forte caratteristica visiva dell'immagine, come una zona di ombra o una zona di luce intensa.

4.3.6 Analisi spettrale

L'analisi spettrale è una tecnica di analisi dei segnali che permette di scomporre un segnale in componenti di frequenza e di visualizzarle attraverso uno spettro.

Per eseguire l'analisi spettrale di un segnale, si utilizza la trasformata di Fourier, che consente di passare dal dominio del tempo al dominio delle frequenze. La trasformata di Fourier scompone il segnale in una serie di componenti sinusoidali di diverse frequenze e ampiezze. Ogni componente sinusoidale è caratterizzata da una frequenza (che rappresenta il numero di oscillazioni al secondo) e da un'ampiezza (che rappresenta l'intensità dell'oscillazione).

In generale, la trasformata di Fourier produce un insieme di valori complessi, ciascuno dei quali rappresenta l'ampiezza e la fase di una determinata componente sinusoidale. Tuttavia, quando si visualizza lo spettro del segnale, si considera solo l'ampiezza di ogni componente, poiché la fase non ha alcuna rilevanza per l'analisi spettrale.

Lo spettro del segnale può essere visualizzato in diversi modi. Il modo più comune è attraverso un grafico a barre, in cui l'asse orizzontale rappresenta le diverse frequenze presenti nel segnale (in Hz) e l'asse verticale rappresenta l'ampiezza di ogni componente sinusoidale (in dB o in scala lineare). In genere, le componenti di frequenza più basse si trovano sulla sinistra del grafico, mentre quelle più alte si trovano sulla destra.

Per rendere più omogenea la rappresentazione di questo tipo di analisi rispetto alle altre già menzionate e per permettere di generalizzare alcune funzionalità (possibilità di eseguire qualsiasi tipo di analisi su qualsiasi tipo di dato, anche se già stato trasformato), nell'implementazione attuale si è deciso di non disegnare la trasformata di Fourier con un grafico a barre, ma di utilizzare il solito grafico lineare. Nulla vieta di modificarne la visualizzazione in un'implementazione futura.

Questo tipo di analisi può fornire informazioni molto utili sulla natura del segnale stesso. Ad esempio, può mostrare la presenza di componenti di rumore o di segnali indesiderati, oppure può evidenziare la presenza di picchi di frequenza che corrispondono a specifici eventi o fenomeni.

Inoltre, il grafico dello spettro può essere utilizzato per filtrare il segnale, ovvero per eliminare le componenti di frequenza indesiderate o per attenuarle, utilizzando filtri digitali appositamente progettati.

Esistono diverse varianti della trasformata di Fourier, tra cui la trasformata discreta di Fourier (DFT) e la trasformata di Fourier a corto termine (STFT). La DFT viene utilizzata per analizzare segnali discreti, come quelli generati da un computer, mentre la STFT viene utilizzata per analizzare segnali non stazionari, ovvero segnali che cambiano nel tempo.

Attualmente il sistema usa l'algoritmo di Welch, che permette di stimare lo spettro di potenza di un segnale diviso in segmenti sovrapposti, in modo da ridurre

l'effetto della variazione temporale del segnale sulla stima spettrale. In questo modo, l'algoritmo di Welch è in grado di fornire una stima più accurata dello spettro di potenza del segnale, soprattutto in presenza di segnali non stazionari o di rumore.

4.3.7 Fitting

In analisi del segnale, il fitting è una tecnica utilizzata per stimare i parametri di un modello matematico che meglio si adattano ai dati del segnale osservato. In altre parole, si tratta di una tecnica di regressione che permette di trovare la curva o la funzione che meglio approssima i dati del segnale.

Il processo di fitting prevede l'utilizzo di un modello matematico che descrive il comportamento del segnale, ad esempio una funzione polinomiale, esponenziale, trigonometrica o di altro tipo. Il modello viene quindi adattato ai dati del segnale utilizzando un algoritmo di minimizzazione dell'errore, ovvero cercando i parametri del modello che minimizzano la differenza tra i dati osservati e quelli predetti dal modello.

Il fitting può essere utilizzato in diverse applicazioni dell'analisi del segnale, come ad esempio per:

Identificare la frequenza di picchi nel segnale, ad esempio nel caso di segnali periodici come quelli prodotti da un oscillatore. Stimare la relazione tra due variabili, ad esempio la relazione tra la temperatura e l'umidità nell'ambiente. Separare le diverse componenti di un segnale composto da più fonti, ad esempio per isolare la componente di rumore o quella del segnale di interesse. Rilevare eventuali deviazioni dal modello previsto, ad esempio per individuare anomalie nei dati del segnale, come nel caso di un'onda sismica che mostra un comportamento anomalo rispetto al modello di propagazione previsto. Tuttavia, il fitting può anche comportare alcune limitazioni e difficoltà, come ad esempio la presenza di rumore nel segnale, la necessità di scegliere un modello adeguato, e la possibile presenza di dati aberranti o di outlier che possono influenzare la stima dei parametri del modello. In questi casi, possono essere necessarie tecniche avanzate di analisi statistica, come la robust regression o la regolarizzazione, per ottenere stime più accurate dei parametri del modello.

Attualmente il sistema supporta le seguenti funzioni di fitting:

- Lineare, approssima i dati con una retta
- Quadratica, approssima i dati con una parabola
- Esponenziale, approssima i dati con una funzione esponenziale
- Power Law, approssima i dati con la legge di potenza

- Gaussiana, approssima i dati con una gaussiana
- Boltzmann, approssima i dati con una sigmoide di boltzmann

4.3.8 Testing

Il testing in un progetto software ha diversi vantaggi, tra cui:

- Identificazione di bug e problemi: il testing permette di individuare errori, bug e problemi nel software, consentendo di correggerli prima che il software venga rilasciato.
- Miglioramento della qualità del software: l'identificazione e la correzione dei problemi consentono di migliorare la qualità del software, rendendolo più affidabile e stabile, soprattutto quando si ha a che fare con un linguaggio tipato dinamicamente come Pyhthon.
- Riduzione dei costi di manutenzione: correggere i problemi prima del rilascio del software riduce il rischio di dover effettuare interventi di manutenzione costosi in futuro.
- Aumento della soddisfazione del cliente: il testing consente di consegnare ad eventuali clienti un software di qualità superiore, riducendo il rischio di malfunzionamenti e migliorando la soddisfazione del cliente.
- Aumento della produttività del team di sviluppo: il testing automatizzato consente di ridurre il tempo necessario per effettuare i test manuali, consentendo ai membri del team di concentrarsi su altre attività di sviluppo.
- Conformità alle normative: il testing può essere utilizzato per garantire la conformità del software alle normative e alle regolamentazioni di un determinato settore.

Per il testing del progetto è stato utilizzato pytest, un framework di testing per Python, che consente di scrivere unit test in modo semplice ed efficace.

Test della logica del framework In questo paragrago verranno esplorati alcuni test che verificano il comportamento della classe di logica. Nel listato Listing 4.1 viene dichiarata la classe di test che conterrà tutti i dati che verranno utilizzati per la verifica del corretto funzionamento della logica. All'interno della repository sono stati aggiunti alcuni file abf di risorse da utilizzare assieme ai test di logica o di immagazzinamento dei dati, visto che la maggior parte delle funzionalità è legata all'apertura di tali file.

Listato 4.1: Classe del file di test

```
import unittest
from src.logics.logics import Logics
.
.


class LogicsTest(unittest.TestCase):
    path_to_abf1 = "res/Data/Data_CH001_000.abf"
    path_to_abf2 = "res/Data/Data_CH002_000.abf"
  .
  .
  .
```

Listato 4.2: Apertura di un abf con 2 canali

```
def test_open_first_abf(self):
  logics_test = Logics()
  logics_test.open(self.path_to_abf1)
  self.assertTrue(logics_test.metadata.
     selected_data_group.channel_count == 2)
  self.assertTrue(len(logics_test.metadata.
     selected_data_group.basic_data) == 2)
```

All'interno delle classi di test vengono definite funzioni ti test similmente al listato Listing 4.2. In particolare in questa porzione di codice viene aperto un abf e si verifica che il file contenga effettivamente il numero di canali attesi. Inoltre, questi file concisi e strutturati servono anche da documentazione e da esempio di utilizzo delle varie funzioni implementate.

Non andremo a discutere nel dettaglio tutti quanti i test, tuttavia chi fosse interessato potrebbe trovarli e analizzarli al seguente link: `https://github.com/Elements-SRL/EDA_python/tree/main/tests`.

4.3.9 CI/CD

CI/CD è l'acronimo di Continuous Integration/Continuous Deployment (o Delivery). Si tratta di un processo di sviluppo del software che si concentra sulla consegna di codice di alta qualità in modo rapido e continuo, attraverso l'automazione e la

collaborazione continua tra gli sviluppatori e gli altri membri del team. In questo caso in particolare il team di sviluppo era composto da una sola persona, tuttavia l'utilizzo di questa pratica dovrebbe consentire una migliore gestione del progetto anche con un'eventuale partecipazione di più sviluppatori.

La Continuous Integration (CI) prevede l'integrazione frequente e continua del codice in un repository condiviso, accompagnata da una serie di test automatizzati per verificare l'integrità del codice e la corretta esecuzione delle funzionalità del software. In questo modo, gli sviluppatori sono in grado di individuare e risolvere i problemi di integrazione in modo tempestivo, riducendo i rischi di conflitti ed errori.

La Continuous Deployment (CD) (o Continuous Delivery, a seconda dei casi) è la pratica di rilasciare in modo continuo e automatico il software già testato e pronto per l'utilizzo, attraverso l'automazione dei processi di distribuzione. In questo modo, gli sviluppatori sono in grado di ridurre i tempi di rilascio del software, migliorando la rapidità di risposta alle esigenze degli utenti e la qualità del prodotto.

La combinazione di CI/CD consente di automatizzare l'intero processo di sviluppo del software, dalla scrittura del codice alla distribuzione finale, riducendo il rischio di errori, migliorando la rapidità di rilascio e aumentando la qualità del software.

Sistema di versionamento

Il sistema di versionamento usato è stato ovviamente **git**. Lo standard de facto per i sistemi di versionamento in qualsiasi progetto software, di qualsiasi dimensione.

Piattaforma

La piattaforma utilizzata per l'hosting della repository online è **GitHub**. L'automazione del testing e del deploy è stato fatto tramite le **GitHub actions**, inoltre, per garantire l'integrità e l'autenticità del codice all'interno del repository Git, i commit sono stati tutti quanti firmati.

4.3.10 Action

Per ora il progetto è connesso ad un solo Workflow, ma in futuro ne verranno sicuramente aggiunti altri per consentire una buona scalabilità col numero di persone che ci lavoreranno.

Nel listato Listing 4.3 può essere osservato il comportamento generale del workflow.

Si analizzino ora questi passi più nel dettaglio:

Listato 4.3: Jobs del workflow

```
name: Python package

on: [ push ]

jobs:
  build:
  .
  .

  create_bundles:
    .
    .

  pre-release:
    .
    .

  clean-up:
    .
    .
```

- build: è il primo job del workflow, ha il compito di tirar su l'ambiente necessario al corretto funzionamento di Python e mandare i test, nel caso i test fallissero il workflow verrebbe interrotto.

- create-bundles: se i test passano viene ritirato su l'ambiente Python, e dopo aver installato le varie dipendenze (su un insieme di sistemi operativi) tramite il pacchetto **PyInstaller**, viene generato l'eseguibile che potrà poi essere scaricato dagli utenti veri e propri.

- pre-release: si procede alla creazione di una pre-release tramite GitHub con al suo interno i codici sorgente e gli eseguibili precedentemente generati.

- clean-up: passo finale in cui tutti gli artefatti precedentemente generati e ancora online su GitHub, ma che non hanno utilità e, anzi, occupano spazio, vengono cancellati liberando memoria [2].

4.4 Risultato finale

In questa sezione verrà mostrata l'applicazione sviluppata in uno scenario giocattolo in cui verrà importato un file su cui poi verranno effettuate alcune trasformazioni.

In figura 4.4 viene effettuata l'apertura di un file edh (un file proprietario dell'azienda), si nota che vengono creati 2 plot, uno per la corrente e uno per la tensione. Questi plot hanno diverse linee, per la precisione quattro nel canale di corrente e una in quello di tensione, questo perché il dispositivo che ha prodotto questi dati era provvisto di quattro canali indipendenti di corrente ma di un solo canale di tensione. Su questo grafico possono essere effettuate operazioni di zoom/pan ecc. Sulla sinistra invece c'è una struttura ad albero per esplorare i dati e le trasformazioni fatte su di essi.

In figura 4.5 viene mostrata la schermata che permette la creazione di un filtro digitale, controllando l'ordine del filtro, la/e frequenza/e da filtrare e il tipo di filtro da applicare. Una volta soddisfatti del filtro creato, questa trasformazione può essere applicata ai dati di partenza ottenendo il grafico in figura 4.6, in questo caso è stato applicato un filtro passa basso con frequenza di taglio di 1 Hz, questo filtro di è praticamente inutile al fine di possibili analisi visto che vado ad eliminare praticamente tutte le frequenze del segnale. Tuttavia questo filtro molto spinto mostra bene come i dati vengano modificati in seguito alla sua applicazione.

Se viene effettuato l'istogramma sui dati grezzi, ovvero quelli presi direttamente dal file, otterremo il grafico in figura 4.7. L'istogramma è l'unica trasformazione

[2] Agli inizi del progetto questo aspetto non era stato considerato ma bastano poche attivazioni del workflow per andare a saturare lo spazio che mette a disposizione il piano free di GitHub

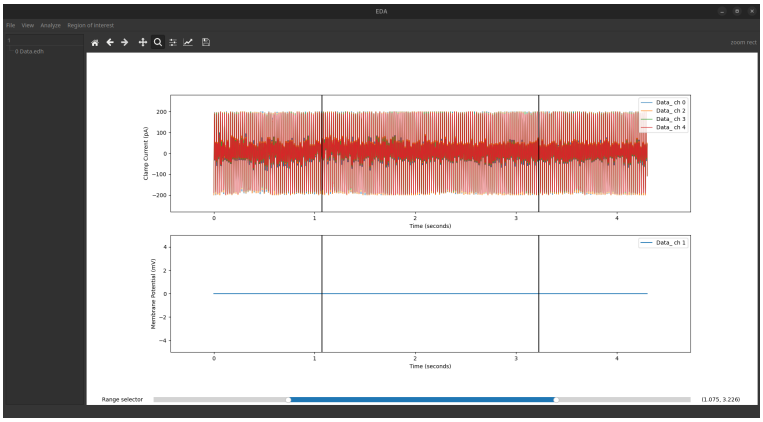

Figura 4.4: Apertura di un file edh

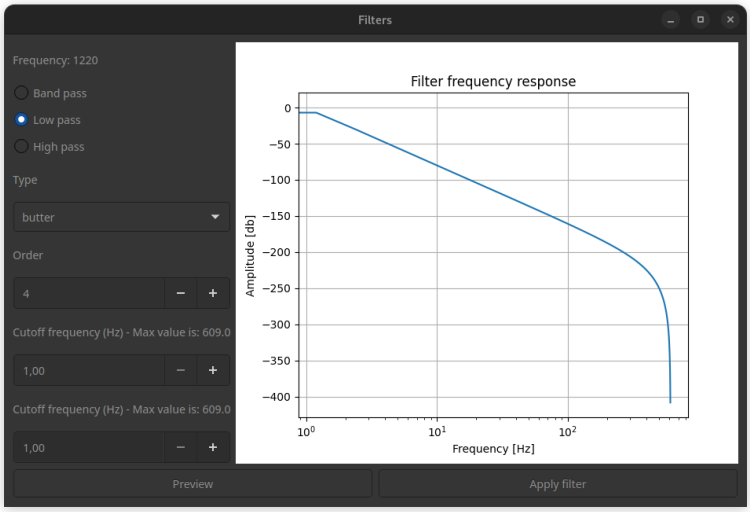

Figura 4.5: Creazione di un filtro

che, per comodità di visualizzazione, crea due DataGroup diversi a differenza della semplice trasformazione che viene normalmente effettuata, questo perché il numero di bins in cui andremo a suddividere le tracce di corrente e tensione sarà diverso (posso avere n bins di tensione e m di corrente con $n \neq m$). Questo si traduce con un cambio di interfaccia, ora si ha un solo grafico (di default viene visualizzato quello di corrente ma si può visualizzare anche quello di tensione). Dal grafico di tensione di cui si è appena discusso è possibile estrarre una sottoporzione, ad esempio quella centrale, tramite gli strumenti forniti dal framework per ottenere un'area di interesse, per esempio ci si può spostare coi cursori che disegnano le linee e usarli per tagliare la sezione interessata o usare la finestra per la creazione avanzata in cui su può decidere di escludere dei canali. Ci si accorge che questa

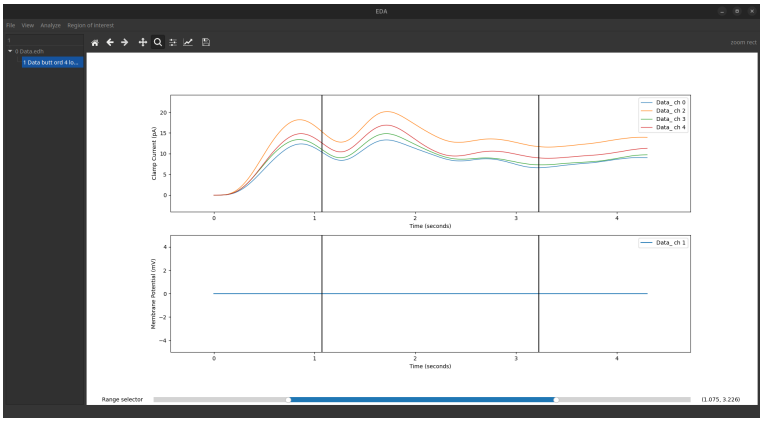

Figura 4.6: Dati filtrati

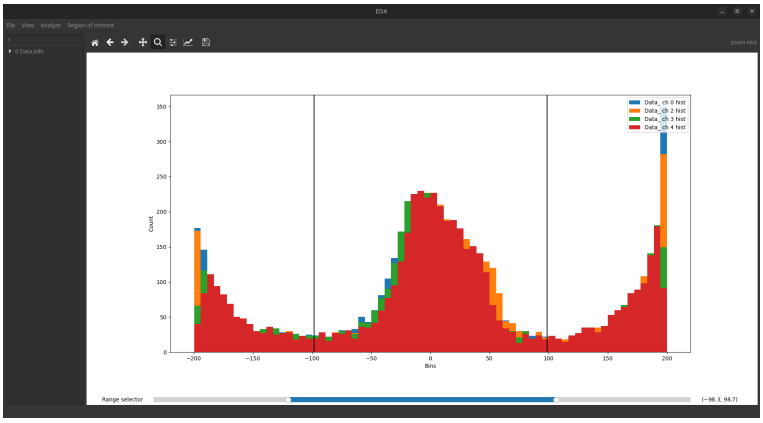

Figura 4.7: Istogramma

porzione di dati sembra assumere la forma di una gaussiana. Sarebbe interessante controllare che tipo di gaussiana ci sia. Si applica quindi la funzione di **fitting** su questa parte di traccia, scegliendo la funzione gaussiana, ottenendo il risultato in figura 4.8. Per ogni canale di corrente viene mostrata la funzione che meglio la approssima e la schermata di riepilogo in figura 4.9, che permette di esportare i parametri di fitting che danno origine alle curve trovate. Ultima, ma non per importanza l'analisi spettrale in figura 4.10. L'analisi spettrale è stata effettuata sui dati grezzi e si può notare come ci sia una componente dominante sui 100 Hz e un picco notevole attorno ai 50 Hz. Starà poi agli esperti del settore capire come mai si hanno queste conformazioni e cosa esse vogliano dire. Auspicabilmente questo tool dovrebbe aiutarli nell'indagine e portarli alla formazione di nuove tesi.

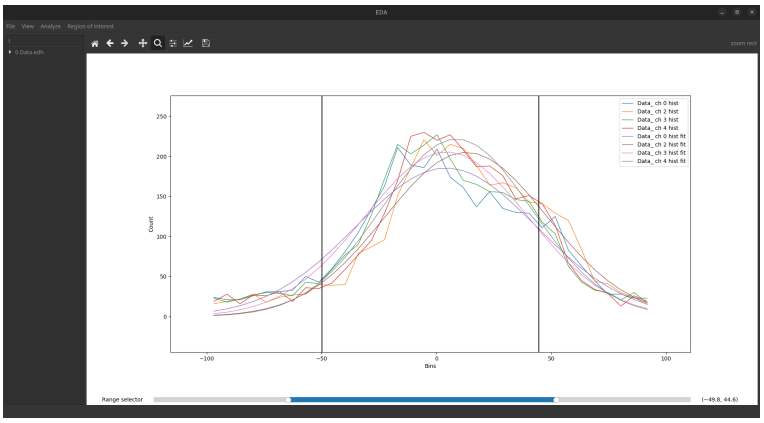

Figura 4.8: Fitting

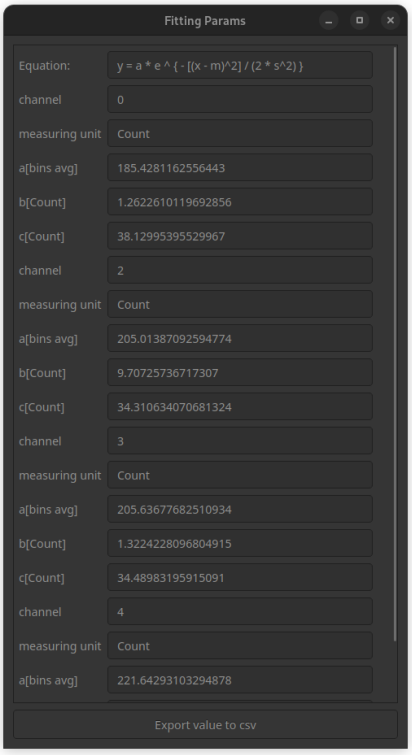

Figura 4.9: Parametri trovati per la funzione desiderata

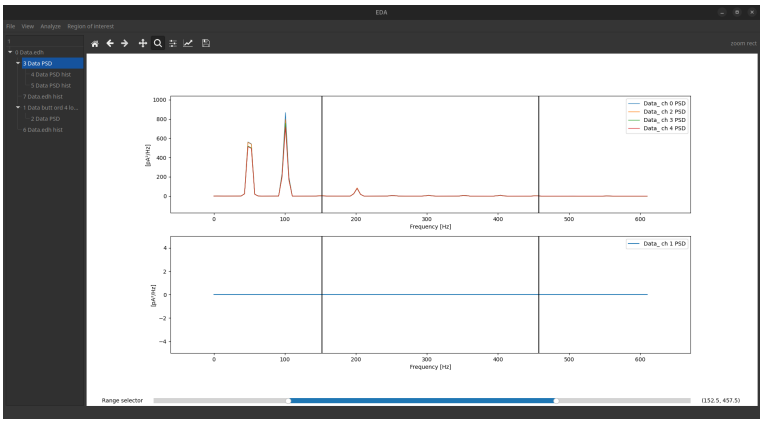

Figura 4.10: Power spectral density

Capitolo 5

Analisi Dati Avanzata

In questo capitolo verranno analizzate alcune feature avanzate, ancora in fase di studio, che sarebbero utili in un framework di analisi di segnali. Nello specifico si parlerà di **Eventi**, della loro estrazione dal segnale e di possibili utilizzi.

5.1 Eventi

Nell'analisi dei segnali, un evento è un cambiamento significativo o una variazione nel segnale che rappresenta una determinata grandezza fisica (ad esempio la tensione, la corrente elettrica, la pressione, la temperatura ecc.) rispetto al tempo o allo spazio.

Gli eventi possono essere di varia natura, come ad esempio la comparsa di un impulso, l'insorgenza di un disturbo o di un rumore, una variazione brusca della frequenza o dell'ampiezza del segnale, una variazione nel regime transitorio del segnale, la comparsa di una armonica o di un'interferenza, una transizione di fase, un'interruzione o una discontinuità del segnale, ecc.

L'estrazione di eventi sarà inoltre fondamentale nella fase di replicazione di esperimenti e in generale per utilizzare tecniche di ML sui dati estratti.

Sempre in questo capitolo infatti si andranno ad esplorare dati provenienti da un dataset pubblico da cui tenteremo di estrarre eventi che poi verranno utilizzati per addestrare un classificatore di virus.

5.1.1 Cos'è per noi un evento

Ora ci si focalizzerà su quelli che possono essere considerati eventi rilevanti. I dispositivi con cui si opera sono per lo più sistemi con dei nanopori, dei quali è stata data una panoramica nel capitolo 2, verranno ora ripresi quei concetti andando più nel dettaglio sul loro funzionamento e su ciò che ci si aspetta di vedere/trovare.

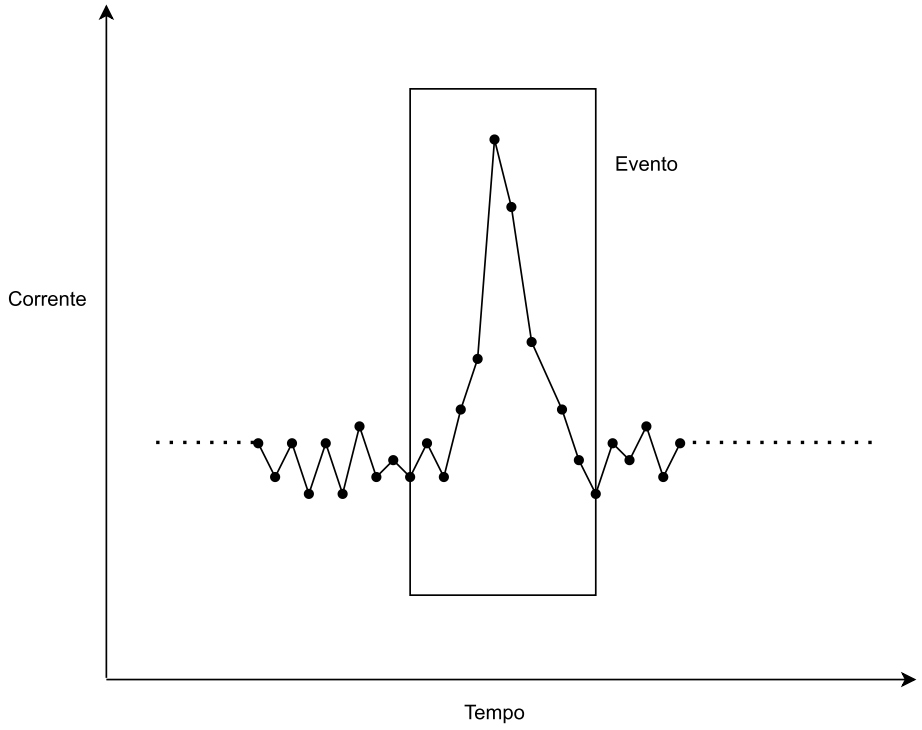

Figura 5.1: Esempio di evento, per convenzione gli eventi vanno verso l'alto

Un sistema con nanoporo è tipicamente formato da un contenitore, diviso in due stanze, ogni stanza è provvista di un elettrodo e riempita di un liquido. Fin qui non c'è nulla di complesso, ora, anziché avere due stanze perfettamente isolate, si immagini che la parete che le divide abbia un buco, con diametro delle dimensioni di ~ 20 nm. Il foro sarà talmente piccolo che il liquido tenderà a rimanere isolato, è a questo punto che entrano in gioco gli elettrodi, grazie ad essi, si può applicare una corrente, tale corrente farà fluire e spostare gli ioni presenti all'interno della soluzione presente nelle camere. Monitorando la corrente che fluisce ci si aspetterà di avere un andamento pressoché lineare (senza considerare il rumore di fondo), tuttavia, se la soluzione contenesse proteine, virus, o in generale molecole di grandi dimensioni (rispetto al nanoporo), si noterebbe che l'andamento della corrente potrebbe avere variazioni considerevoli.

Questo perché la molecola che passa all'interno del nanoporo non è di dimensioni trascurabili rispetto al poro stesso e nel momento in cui la corrente spinge questa molecola ad attraversarlo si ha un abbassamento molto evidente della corrente (**Evento Resistivo**). L'oggetto dell'analisi, sono appunto questi spike di cui si ha un esempio in figura 5.1, queste variazioni improvvise possono essere di tre tipi:

- Eventi Resistivi: abbassamento repentino di corrente
- Eventi Capacitivi: aumento repentino di corrente
- Eventi Bifasici: concatenazione di un evento resistivo e uno capacitivo.

Ovviamente più la frequenza di campionamento è alta, più si riuscirà a dare una forma precisa agli spike e meno si rischierà di perdere eventi. I dispositivi più avanzati attualmente in commercio raggiungono frequenze di campionamento di 10 MHz, frequenze più che sufficienti per un'analisi accurata di segnali. Questi tre tipi di eventi sono attualmente oggetto di studio, e non si hanno ancora molte informazioni sulla loro origine e sul loro significato, per questo motivo il primo step per permettere uno studio accurato di questi fenomeni è una strumentazione adeguata che supporti gli studiosi del settore.

5.1.2 Esplorazione dei Dati

I dati disponibili per fare ricerca lato software non sono abbondanti a causa della difficoltà intrinseca degli esperimenti condotti. Si lavora sempre con correnti molto molto piccole, e attrezzature molto difficili da utilizzare e, nel caso di problemi può essere difficile risalire alla loro radice e trovare una soluzione. Inoltre a parte il formato abf, di cui si è parlato, non c'è uniformità dei dati, o un formato standard, il che complica la vita degli sviluppatori e tende sfavorire uno sviluppo open. Il lavoro di estrazione di eventi è quindi iniziato con l'esplorazione dei dati. Come già accennato, per questa parte di progetto è stato utilizzato un dataset open, disponibile al seguente link: `https://zenodo.org/record/4971313#.Y_ol0dLMKXI`. Questi dati sono stati usati per la scrittura del paper *Combining machine learning and nanopore construction creates an artificial intelligence nanopore for coronavirus detection* [9] che si tenterà di riprodurre più avanti, nella sezione 5.2. Per ora basti sapere che questi sono i dati migliori e più abbondanti che sono stati trovati e su cui si è potuto sperimentare.

Tuttavia questi dati non sono ben descritti, né facilmente esplorabili o navigabili.

Dopo aver scaricato cartella per cartella, è iniziato un processo iterativo di esplorazione (in cui si è dovuto fare anche reverse engeneering sui dati vista la mancanza di documentazione), estrazione e raffinamento dei dati.

Dopo uno studio accurato si è scoperto che i dati erano caratterizzati da 0,1 o più inversione di fase, il che vuol dire che gli eventi erano a volte positivi e a volte negativi.

Per far fronte a questo problema è stato prodotto uno script che correggesse la fase, invertendola dove necessario, questo avrebbe permesso di ragionare in maniera più semplice sui dati permettendo di cercare eventi in maniera coerente.

Inoltre le tracce spesso erano divise su più file e spesso sarebbe stato scomodo lavorare su file multipli, per questo motivo i file appartenenti alla stessa misurazione/esperimento sono stati concatenati in un'unico file.

Un altro problema non indifferente era la visualizzazione dei dati. Il framework costruito infatti è pensato per l'apertura di file di tipo abf, che sono i più comuni, tuttavia i file in questione erano stati registrati in binario (.dat) e questi file non hanno un modo univoco per essere letti, potrebbero contenere uno stream di interi, float, o qualsiasi tipo di dato. Facendo qualche analisi e andando per tentativi è stato possibile ad aprirli e visualizzarli, per comodità è stato anche creato un branch che implementasse questa feature sull'applicazione. Tuttavia per essere rilasciata dovrebbe essere molto raffinata e più personalizzabile, infatti essendo i file in binario, la loro codifica e decodifica non è univoca, potrebbero esserci dispositivi a 4, 8, n canali, con sampling rate variabili e l'estrazione dei dati sarebbe sempre diversa.

Ora che i dati sono tutti "dritti", "attaccati" e visualizzabili, si può iniziare a lavorare ad algoritmi di estrazione di eventi che verranno discussi nella sottosezione 5.1.3.

5.1.3 Algoritmo di estrazione

Anche questa fase è stata portata avanti in maniera iterativa come nel diagramma in figura 5.2. Questi algoritmi di estrazione sono infatti supervisionati e in futuro, se i dati disponibili fossero sufficientemente abbondanti, sarebbe interessante utilizzare tecniche di ML e Deep Learning per l'estrazione automatica di eventi.

Il principio di funzionamento alla base di questo algoritmo è semplice:

- si identifica una *baseline*, ovvero una linea di base in cui il sistema è stabile, a riposo
- si decidono due soglie di durata, minima e massima, per evento
- si decide la soglia di ampiezza, superata la quale si classifica lo spike come evento e magari una soglia massima per ogni evento

Decisi questi parametri si fa andare l'algoritmo e si controllano i risultati. Tuttavia, nonostante l'idea alla base dell'algoritmo sia semplice, non è facile ottenere risultati soddisfacenti a causa di problemi intrinseci degli esperimenti, primo tra tutti il rumore. Un problema non banale ad esempio è proprio il primo punto del nostro elenco, la baseline infatti può variare nel tempo, spostandosi diventa anche molto difficile ricavare gli altri parametri necessari al corretto funzionamento dell'algoritmo. Per fare un esempio banale, se si decide che la soglia minima di ampiezza per classificare un evento come tale è 3 volte il valore della baseline, che

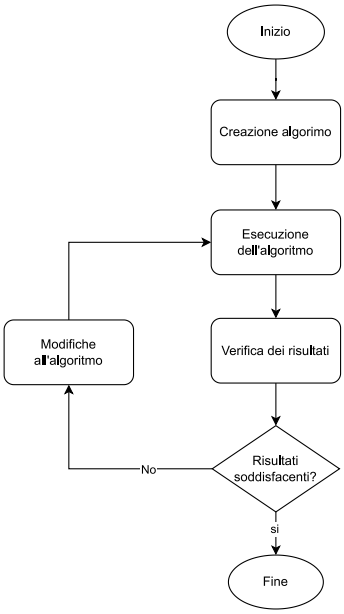

Figura 5.2: Produzione dell'algoritmo di estrazione degli eventi

inizialmente vale 5, ma dopo un tempo x vale 15, dal tempo x in poi classificheremo come evento la baseline stessa (o del rumore).

Questo comportamento ovviamente è qualcosa da evitare e la tecnica che è stata scelta è la seguente. La baseline viene calcolata come media mobile su una finestra di dati, finestra con al centro un "buco" che dovrebbe ospitare l'evento. Questo accorgimento serve perché altrimenti gli eventi entrerebbero a far parte della baseline e siccome la soglia che delimita l'ampiezza minima di un evento è calcolata in funzione della media mobile, se si considerasse tutta quanta la finestra di dati, senza escludere la porzione in cui si troverà l'evento, aumenteremmo la soglia minima e di fatto l'estrazione dell'evento sarebbe meno accurata.

Per chiarire l'algoritmo si può osservare l'immagine in figura 5.3. Quello sarebbe lo stato dell'estrattore di eventi al tempo t,$\mathbf{m}$, $\mathbf{x}$ ed $\mathbf{e}$ sono i parametri che vengono decisi a priori e da loro dipende la corretta estrazione di eventi.

Tuttavia, il calcolo delle soglie avviene dinamicamente e questo permette all'algoritmo di adattarsi a cambiamenti di corrente e al drifting della baseline rendendolo sufficientemente robusto per un'estrazione sensata di eventi su tracce diverse.

Inoltre, solitamente quando si studiano gli eventi due sono i fattori di interesse maggiore:

1. L'ampiezza: ovvero l'altezza massima (o minima) che viene raggiunta da un evento

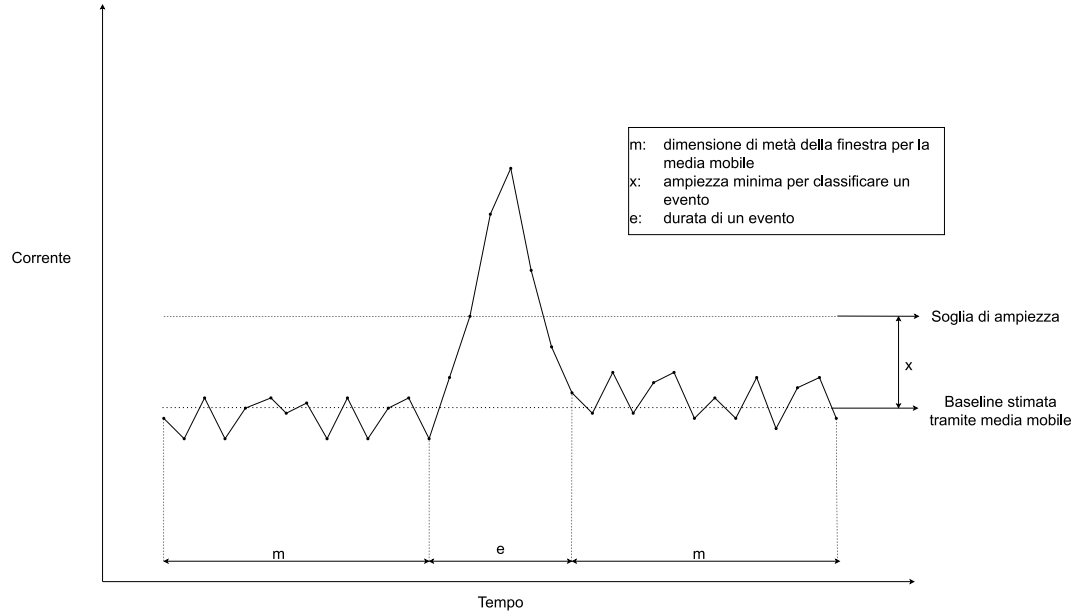

Figura 5.3: Algoritmo di estrazione degli eventi

2. La durata: ovvero il tempo impiegato dal segnale per tornare al livello di baseline dal picco * 2 (andata e ritorno). Questa misura in particolare potrebbe essere molto soggetta al rumore, diventa infatti molto difficile capire quando un evento inizia, di conseguenza, come misura di durata, anziché scegliere esattamente il tempo in cui il segnale supera la baseline (o una cerata soglia), viene usato un altro tipo di misura. In letteratura sono state proposte altre misurazioni che conferiscono maggior robustezza ai dati. In questo studio è stata utilizzata la D50 ovvero la durata dell'evento al 50% dell'ampiezza.

In figura 5.4 può essere osservato un esempio di durata e ampiezza di un evento in un mondo perfetto in cui la baseline è costante e priva di rumore.

Dopo aver spiegato il funzionamento dell'algoritmo si può passare all'implementazione.

5.1.4 Implementazione

L'intera implementazione di questa parte sperimentale può essere trovata al seguente link: `https://github.com/lucarossi147/py-scripts`, in particolare verrà preso in analisi il file `https://github.com/lucarossi147/py-scripts/blob/main/detector2.py`.

I dati originali erano strutturati in cartelle, una per ogni misurazione, ogni cartella conteneva al proprio interno zero, uno o più file, alcuni di descrizione del

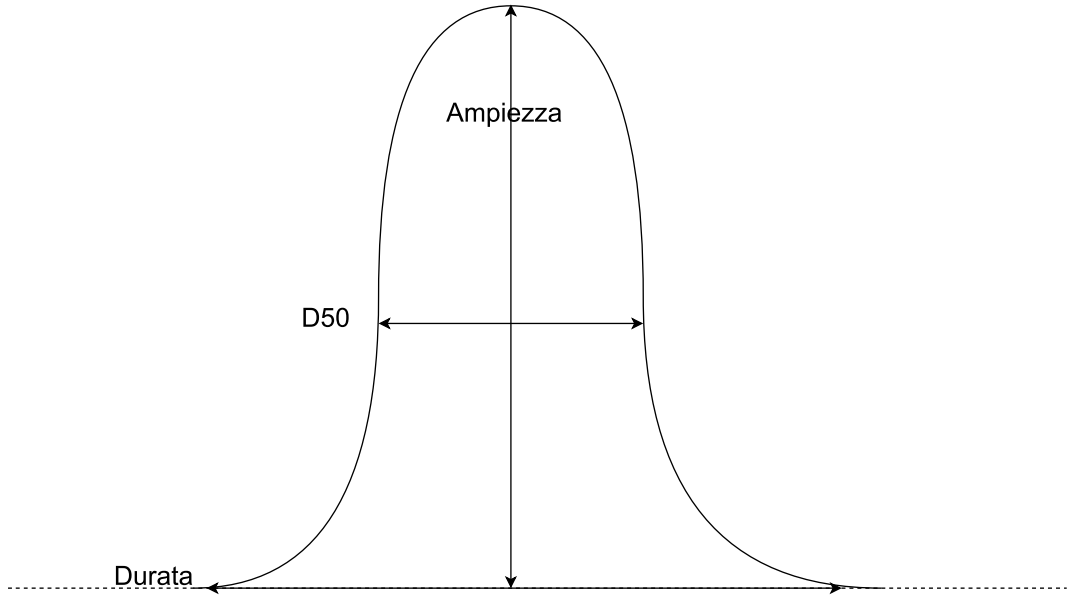

Figura 5.4: Ampiezza e durata di un evento

file stesso, alcuni di monitoraggio (che non sono stati utilizzati perché reputati inutili per l'analisi di eventi in seguito alla fase esplorativa) e zero, uno o più file di misurazione (.dat).

Queste cartelle potevano essere a loro volta raggruppate per macro area (ad esempio per misurazioni che riguardano un certo tipo di malattia).

Goal Si vuole che, data una cartella, l'algoritmo restituisca in output una cartella con la medesima struttura di quella presa in input e che per ogni misurazione al proprio interno venga generata una cartella con due file:

- Un file binario: dat con la concatenazione di tutti gli eventi trovati
- Un file in formato csv contenente:
  - Gli indici di inizio e fine degli eventi estratti, legato al file costruito
  - Il percorso del file originale da cui sono stati estratti gli eventi

Scartando cartelle vuote o inutilizzabili in caso di necessità.

Risultato Ora si passerà all'analisi delle parti più interessanti del detector di eventi, chiamato anche "Event detector".

Non ci si concentrerà sulla parte di creazione delle cartelle che conterranno i risultati ma soltanto sui risultati stessi.

Listato 5.1: calcolo dei parametri da utilizzare

```
    mov_avg_length_mono = 5000
    mov_avg_length = mov_avg_length_mono * 2 + 1
    min_event_l = 10
    max_event_length_mono = 250
    max_event_length = max_event_length_mono * 2 + 1
    max_std = 0.2e-9
    mov_avg_den = mov_avg_length - max_event_length
    min_noise_ratio = 2
    b = [1 / 3, 1 / 3, 1 / 3]
    a = 1
    sr = 250000
    blf, alf = signal.butter(4, 1000 / (sr / 2), 'low')
    smoothed = signal.filtfilt(b, a, raw)
    smoothed = smoothed[mov_avg_length_mono + 1:len(smoothed) - mov_avg_length_mono]
    low_freqs_data = signal.filtfilt(blf, alf, raw)
    cs = np.cumsum(raw)
    cs2 = np.cumsum(np.power(raw, 2))
    # cumsum low freqs
    cslf = np.cumsum(low_freqs_data)
    cs2lf = np.cumsum(np.power(low_freqs_data, 2))
    center = np.array(range(mov_avg_length_mono + 1, len(raw) - mov_avg_length_mono))
    m = (cs[center + mov_avg_length_mono] - cs[center + max_event_length_mono] + cs[
        center - 1 - max_event_length_mono] - cs[center - 1 - mov_avg_length_mono]) /
            mov_avg_den
    s = np.sqrt((cs2[center + mov_avg_length_mono] - cs2[center + max_event_length_mono] + cs2[
        center - 1 - max_event_length_mono] - cs2[center - 1 - mov_avg_length_mono]) /
            mov_avg_den - np.power(m, 2))

    mlf = (cslf[center + mov_avg_length_mono] - cslf[center + max_event_length_mono] + cslf[
        center - 1 - max_event_length_mono] - cslf[center - 1 - mov_avg_length_mono]) /
            mov_avg_den
    slf = np.sqrt((cs2lf[center + mov_avg_length_mono] - cs2lf[center + max_event_length_mono]
        + cs2lf[
        center - 1 - max_event_length_mono] - cs2lf[center - 1 - mov_avg_length_mono]) /
            mov_avg_den - np.power(mlf, 2))
    noise_ratio = s / slf
    th = m + 3 * s
```

La prima parte della fase di detezione degli eventi è sempre quella di import dei dati raw dai .dat.

Nel listato Listing 5.1, dopo aver elencato i parametri che verranno utilizzati di fatto sono delle costanti, il segnale originale viene smussato, viene calcolato il rumore alle basse frequenze applicando un filtro passa basso ai dati originali ed anche questi dati vengono smussati. Vengono poi calcolati la baseline e la varianza del segnale con il metodo descritto precedentemente. Dato che questa fase era ancora prototipale, e il linguaggio utilizzato era Python, si è cercato di utilizzare il più possibile funzioni derivanti dal modulo **numpy**, questo perché i dati da analizzare, seppur derivanti da relativamente pochi esperimenti, avevano un peso complessivo di parecchi GB e analizzarli tutti quanti utilizzando codice Python puro, sarebbe stato eccessivamente dispendioso in termini di tempo. Ad esempio, per il calcolo della media e della varianza è stata utilizzata la funzione **cumsum**. Al termine di tutte le trasformazioni, il risultato finale fosse una lista di valori direttamente confrontabili con le soglie definite e utilizzabili per l'analisi, questo sempre per cercare di utilizzare funzioni derivanti da numpy.

Il listato Listing 5.2 invece illustra la macchina a stati finiti utilizzata per l'estrazione di eventi veri e propri. Seppur questa implementazione lasci ampio margine di miglioramento, sia in fatto di leggibilità che di performance, una mo-

Listato 5.2: calcolo dei parametri da utilizzare

```
NO_EVENT = 0
COUNTING = 1
EVENT = 2
NOISE = 3
status = NO_EVENT
count = 0
noise_count = 0
events = []
begin_of_event = 0
end_of_event = 0
print("analyzing")
for i in range(len(center)):
    if status == NO_EVENT:
        if noise_ratio[i] < min_noise_ratio:
            noise_count += 1
        else:
            noise_count = 0
        if noise_count > mov_avg_length_mono:
            status = NOISE
            continue
        if s[i] > max_std:
            continue
        if smoothed[i] > th[i]:
            begin_of_event = i
            count = 1
            status = COUNTING
    elif status == COUNTING:
        if smoothed[i] > th[i]:
            count += 1
            end_of_event = i
            if count >= min_event_l:
                status = EVENT
        else:
            status = NO_EVENT
    elif status == EVENT:
        if count > max_event_length:
            status = NO_EVENT
            continue
        count += 1
        if smoothed[i] > th[i]:
            end_of_event = i
        if smoothed[i] < m[i] and end_of_event > begin_of_event and count > min_event_l:
            events.append([begin_of_event, end_of_event])
            status = NO_EVENT
    elif status == NOISE:
        if noise_ratio[i] > min_noise_ratio:
            status = NO_EVENT
            noise_count = 0
```

dellazione di questo tipo risultava più facile da pensare inizialmente. Se questa funzionalità si dimostrasse sufficientemente buona da poter essere rilasciata verrebbe sicuramente sviluppata una versione più dichiarativa in un linguaggio che permetta performance migliori, come ad esempio **Rust**.

Dopo questa fase manca solo l'encoding in csv e dat, che però non verranno discussi.

5.2 Ricerca e replicazione

L'estrazione automatica di eventi non è utile soltanto agli scienziati e ingegneri biomedici, è anche utile a possibili sviluppatori software per sperimentare e ricercare nuovi modi per automatizzare dei task ripetitivi o trovare soluzioni nuove a problemi già risolti.

Il paper *"Combining machine learning and nanopore construction creates an artificial intelligence nanopore for coronavirus detection"*, che è stato introdotto nella sezione precedente, sfrutta un approccio innovativo per classificare virus (in particolare si parla di coronavirus) tramite l'utilizzo di nanopori e machine learning.

5.2.1 Riassunto del paper

Attualmente la tecnologia più utilizzata per diagnosticare la presenza di coronavirus è la *reverse transcription-polymerase chain reaction*(RT-PCR). Tuttavia questa tecnica richiede l'estrazione di RNA virale da un campione clinico per ottenere sensività elevate. Il team di scienziati propone un metodo innovativo che sfrutta i nanopori e un sistema di intelligenza artificiale per ottenere alte sensività senza il bisogno di estrazione dell'RNA virale. La piattaforma risultante, che loro chiamano "The artificial intelligent nanopore" consiste in un server che ospita un sistema di machine learning, un sistema di misurazione portatile, altamente veloce e preciso e moduli di nanopori economici. Hanno dimostrato di riuscire a riconoscere con successo quattro tipi diversi di coronavirus di grandezze simili:

- HCoV-229E
- SARS-CoV
- MERS-CoV
- SARS-CoV-2

La detezione di SARS-CoV-2 in campioni salivari raggiunge una sensitività del 90% e una specificità del 96% con una misurazione di 5 minuti.

5.2.2 Disclaimer

Essendo l'azienda che ha proposto il progetto una produttrice di sistemi con nanopori, l'implementazione di un sistema come quello descritto nel paper sarebbe stato senza dubbio molto interessante. Tuttavia ci sono alcune limitazioni, prima su tutte, la mancanza di dati da parte dell'azienda produttrice. Alla base di un sistema accurato di machine learning c'è una buona mole di dati che permette di addestrare un classificatore a riconoscere scenari differenti.

La mancanza di dati proprietari non sarebbe di per sé limitante, sempre che in rete si riescano a trovare dati dello stesso tipo su cui poter provare algoritmi di machine learning. Purtroppo però, essendo questa una tecnologia all'avanguardia e molto innovativa, sono pochissimi i dati in circolazione, infatti, gli unici disponibili sono quelli rilasciati dai ricercatori.

È ottimo avere un dataset pubblico e sempre disponibile, tuttavia il dataset in questione non è ben documentato (oltre che scomodo da scaricare).

Come anticipato nel capitolo precedente, sono state necessarie diverse ore e molte iterazioni di verifica dei risultati per arrivare ad una buona comprensione dei dati.

L'altro porblema è l'abbondanza dei dati. I dati prodotti e messi online sono veramente tanti, scaricati e decompressi sfiorano i 150 GB, ma allora come mai non sono sufficienti per ottenere risultati notevoli in tempi ragionevolmente brevi?

Le ragioni sono molteplici:

- Numero di ricercatori: Il team di ricercatori che ha lavorato al paper contava più di 20 persone altamente specializzate, mentre il gruppo che si è occupato della replicazione di questo esperimento era composto soltanto da due persone.

- Errori dovuti all'interpretazione dei dati: A causa della mancanza di documentazione, è possibile che siano stati presi in esame dati non pertinenti, errati o che siano stati scartati dati buoni.

- Know how: Gli autori del paper hanno sicuramente sviluppato un Know How elevato e tecniche efficaci, che però non vengono spiegate nel dettaglio rendendo il paper non riproducibile se non tramite la loro apparecchiatura.

- Il problema è difficilmente supervisionabile: Ad ogni misurazione è associata una classe, ma ogni classe potrebbe avere al suo interno eventi molto diversi tra loro, questo è dovuto al fatto che nel liquido che viene inserito all'interno del dispositivo con nanoporo non si hanno certezze assolute, tantomeno se si tratta di un campione salivare. Il sistema con nanoporo potrebbe intercettare moltissimi eventi che però in realtà non ci danno alcuna informazione sul virus.

- I dati non sono densi di informazione: un file può arrivare a pesare quasi un GB, ma in quel GB di dati potremmo avere soltanto rumore, non si hanno garanzie sulla qualità dei dati.

- Mancanza di dati per misurazioni singole: nel paper si parla di capacità di distinguere 4 tipologie diverse di CoronaVirus da colture sviluppate in laboratorio, tuttavia queste 4 categorie di virus hanno meno di ~ 30 misurazioni l'una, il che rende difficile fare stime statisticamente rilevanti.

I risultati ottenuti non sono al livello di quelli dei ricercatori giapponesi. In futuro, magari con tecnologie disponibili in-house is riuscirà ad arrivare ad ottenere risultati accurati come quelli del paper.

Premessa Nonostante un problema di classificazione multiclasse sia notoriamente più complesso rispetto ad un problema di classificazione binario si è scelto di concentrarsi su questo tipo di problema per due motivazioni principali:

- Dati teoricamente più "puliti": I dati che i ricercatori hanno messo a disposizione per il problema di classificazione multiclasse sono derivati da coulture di laboratorio, quindi ci si aspetta che gli eventi generati da un tipo di virus siano simili tra loro, auspicabilmente più simili tra loro rispetto ad eventi generati da virus diversi. Gli altri dati invece venivano estrapolati da campioni salivari, questo implica che un campione contenente un tipo di virus diverso da quello oggetto di studio potrebbe generare degli eventi x, magari il campione contiene più di un tipo di virus ecc. In generale lo studio sui dati coltivati sembrava la scelta più saggia, anche perché il sistema di detezione degli eventi non era stabile né definitivo, e avere dei dati più "puliti" avrebbe contribuito alla buona riuscita dell'esperimento.

- Trasferimento del know how: Una volta capito quali feature estrarre dal campione in input, quali operazioni fare sui dati e quali siano i classificatori più efficaci per questo tipo di problema, portare questa conoscenza al problema binario diventava banale.

5.2.3 Approcci e risultati

Principalmente sono state provate due strade per l'analisi e classificazione dei file di misurazioni. Entrambe hanno come passo comune l'estrazione di eventi dai dati raw, che è stato trattato nella sezione 5.1.3. I risultati di questa classificazione sono fortemente dipendenti dall'algoritmo di estrazione, se gli eventi estratti hanno la forma che ci si aspetta, può essere progettata una toolchain che trasformi i dati e li classifichi, ma se l'algoritmo di estrazione tira fuori i dati che secondo gli esperti del settore non hanno alcun valore o significato, praticamente ci si sarebbe ritrovati a popolare il dataset con dati "dannosi" e controproducenti.

Altri termini che verranno utilizzati spesso sono:

- Evento: un picco di forma simile a quelli discussi nella sezione precedente

- Misurazione: File contenente da zero a N eventi, consiste nella traccia registrata dal team di scienziati che ha lavorato al paper, modificata per una manipolazione più agevole [1].

- Classe: è associata con rapporto 1 a 1 alla misurazione e corrisponde alla tipologia di virus della misurazione

[1] Gli sviluppatori avevano i file spezzettati, con cambiamenti di fase e racchiusi dentro cartelle, i nostri dati sono tutti in fase e in unico file

Quello che si vuole andare a fare non è realmente classificazione di eventi, quanto classificazione di **misurazioni**. Si vuole capire tramite gli eventi contenuti in una misurazione di corrente di durata x (in questo caso 5 minuti) se e quale tipologia di virus sia stata rilevata. Per farlo si potrebbe dover classificare anche gli eventi, ma come già accennato precedentemente, non è detto che **tutti** gli eventi relativi alla misurazione siano eventi di quella classe, alcuni eventi potrebbero non essere associati a nessuna classe o a virus/batteri che non sono oggetto di ricerca.

Prima di pensare alla classificazione bisogna però definire delle possibili feature da utilizzare per l'addestramento di classificatori. Per questo motivo verranno discussi due approcci, uno basato sulla costruzione di prototipi di eventi, mentre l'altro utilizza delle misure che sintetizzano la forma degli eventi.

Prototipi di eventi

L'idea alla base di questo approccio è legata all'assunzione (che sembra emergere anche dal paper) che eventi legati alla stessa classe abbiano una forma tra loro più simile rispetto ad eventi di classi diverse.

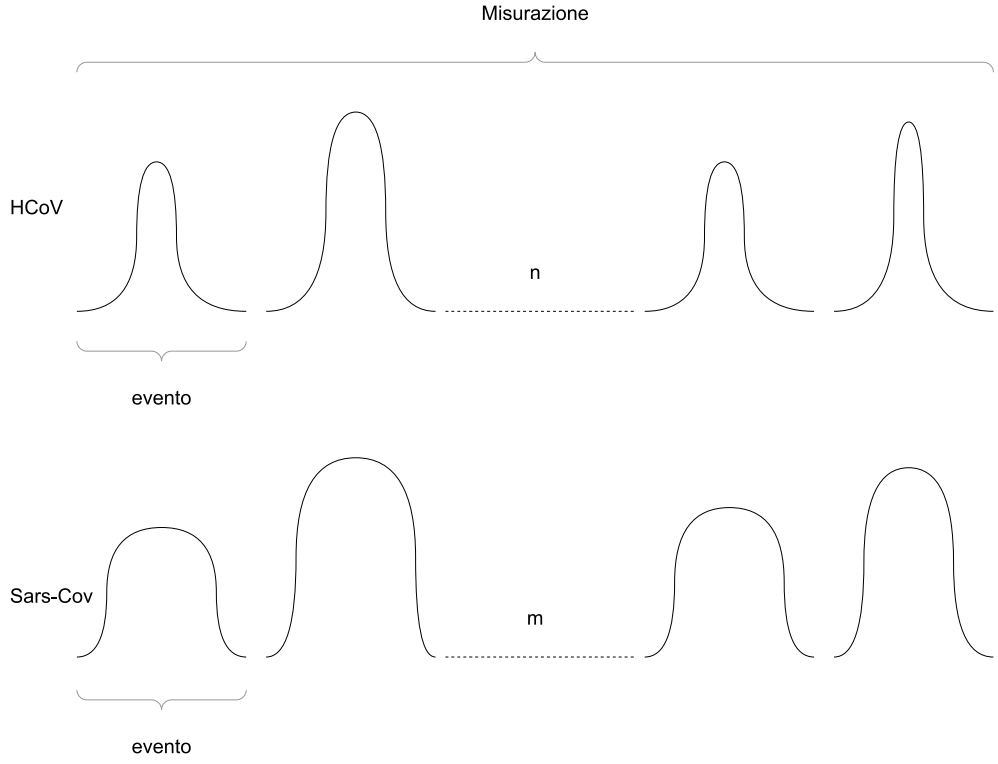

Figura 5.5: Immagine che accentua come ci si immagina eventi in classi diverse

In figura 5.5 si ha una rappresentazione di una misurazione, essa è composta da un certo numero di eventi di durata e ampiezza diversa, tuttavia misurazioni diverse potrebbero avere un numero diverso di eventi, anche misurare due campioni provenienti dalla stessa coltura potrebbe portare ad eventi diversi.

Fatta questa premessa, si vuole capire il prototipo di evento di ogni classe ovvero la forma che secondo qualche metrica meglio approssima gli eventi di una stessa categoria.

Una volta ottenuti questi dati per classificare nuove misurazioni dovremmo estrarre eventi dalla nuova misurazione in input, classificare gli eventi ottenuti e combinare questi risultati secondo una qualche regola, ad esempio assegnando la label alla misurazione con la classe più frequente.

L'algoritmo di estrazione degli eventi con questo approccio non era ancora molto preciso e affidabile ma questo non ha pregiudicato l'inizio del nostro lavoro. Per fare analisi sui dati è stato utilizzato **Jupyter Notebook**, un ambiente di sviluppo interattivo che consente di creare e condividere documenti che contengono codice eseguibile, equazioni, visualizzazioni e testo narrativo. È ampiamente utilizzato in ambito scientifico e tecnico per la prototipazione, l'analisi dei dati, la visualizzazione e la condivisione dei risultati di ricerca.

Durante la parte di esplorazione dati, per prima cosa si è cercato di capire la forma, degli eventi, per questo motivo si è pensato di fare 2 operazioni per ogni evento:

1. Una normalizzazione sull'ampiezza: ovvero fare in modo che tutti gli eventi avessero ampiezza 1, quindi ogni valore contenuto in un evento è stato diviso per l'ampiezza massima

2. Una normalizzazione di durata: siccome gli eventi non hanno una durata fissa, per capirne la forma e in generale per portarli ad una rappresentazione comune, viene fatta un'interpolazione sugli eventi, in modo tale da portarli tutti quanti ad una lunghezza precedentemente concordata, di 35 campioni.

Dopo aver sperimentato un po' con Framework di machine learning come **Weka**[1] e **SciKit learn**[8] ci si è accorti di essere finiti in un vicolo cieco. In particolare gli scarsi risultati ottenuti erano probabilmente da spiegarsi con:

- Un algoritmo di estrazione degli eventi immaturo

- Utilizzo di fetature poco rappresentative: ad esempio tra le feature utilizzate per la classificazione c'erano i punti disegnati dalla curva (35) che ne davano un'informazione sulla forma, inoltre queste feature erano ottenute tramite interpolazione che comunque aggiunge errori.

- Curse of dimensionality: Addestrando a riconoscere eventi su così tante feature i classificatori avrebbero potuto fare fatica a classificare record correttamente.

Per questo motivo dopo aver ragionato su possibili modifiche applicabili, si è deciso di tentare con un nuovo approccio.

Sintesi della forma degli eventi

Questa metodologia di lavoro punta ad estrarre i dati direttamente dagli eventi, senza costruire rappresentazioni intermedie o interpolazioni di feature.

In particolare le misure che vengono utilizzate per descrivere gli eventi, sono la durata (ad una certa ampiezza, come può essere d50, d30, d60) e il centro dell'evento per quella durata.

Questo accorgimento ci ha permesso di snellire di molto la dimensionalità dei campioni, portandoli ad avere solamente sei attributi nelle prime versioni del classificatore, una misura di centro e una di durata ogni terzo di evento.

In seguito per vedere come venisse influenzata l'accuratezza si è deciso di utilizzare un maggior dettaglio, fino ad arrivare a 20 attributi. Tuttavia questi attributi sono più che altro da intendere come punto di partenza per esplorare dati e fare prototyping. Ad esempio con software come Weka diventa banale rimuovere degli attributi, studiare come cambia la classificazione in base alle modifiche effettuate e capire quali attributi hanno maggior contenuto informativo.

Weka ha avuto un ruolo importante anche per studiare quale algoritmo di classificazione portasse ai risultati migliori.

Dopo una sessione iniziale di studio con Weka si è passati ad un'implementazione in Python con scikit-learn, che garantisce più flessibilità su operazioni possibili e metodologie di lavoro, in particolare i risultati sono migliorati notevolmende andando ad addestrare i classificatori su training set bilanciati.

Tramite scikit-learn sono state valutate le performance di classificazione facendo leave-one-out. Per il problema che è stato studiato le misurazioni utilizzabili arrivavano più o meno a un centinaio (per tutte e 4 le classi sommate assieme).

L'algoritmo completo è visionabile al seguente link: `https://github.com/lucarossi147/py-scripts/blob/main/classifier2.ipynb` e gli ultimi risultati prima dello stop ai lavori erano rassicuranti.

Tuttavia i risultati ottenuti non sono considerabili statisticamente rilevanti, come già detto le misurazioni con cui è stato possibile lavorare erano all'incirca una cinquantina in tutto, 30 dei quali venivano da una sola categoria di virus. Per questo motivo, pur avendo ottenuto accuratezze nell'intorno dell'80%, non si può dire che queste siano statisticamente rilevanti, soprattutto quando altri tipi di virus avevano circa cinque campioni per classe.

Resta ancora ampio margine di miglioramento in generale, ma si spera di poter replicare questi esperimenti con più dati e minor rumore per risolvere al meglio il problema di classificazione di eventi e/o misurazioni.

Capitolo 6

Conclusioni

Gli obiettivi di questa tesi erano di sviluppare un framework per l'analisi di dati di elettrofisiologia che ponesse la base per un software estendibile e facilmente adattabile ai bisogni specifici di scienziati ed elettrofisiologi.

Il framework sviluppato si è dimostrato estremamente utile per l'analisi dei dati di elettrofisiologia, consentendo di lavorare in modo più efficiente e accurato. Inoltre, il framework ha dimostrato di essere altamente flessibile e personalizzabile, il che lo rende adatto per una vasta gamma di applicazioni nella ricerca di elettrofisiologia.

Tra le funzionalità raggiunte dal software si hanno la visualizzazione di diverse tipologie dati analizzati (gapfree, episodic) tramite algoritmi ottimizzati per velocizzare la navigazione, la possibilità di esportare tracciati o parti di tracciati in csv, filtraggio tramite filtri digitali, creazione di istogrammi, analisi spettrale e fitting di curve su porzioni di dati con conseguente possibilità di esportazione dei parametri trovati. Tra le feature sperimentali ci sono inoltre l'estrazione di eventi a partire da un tracciato e si sta valutando l'integrazione di classificatori personalizzati.

In sintesi, questa tesi ha dimostrato che il framework sviluppato può essere un valido strumento per la comunità di ricerca in elettrofisiologia, offrendo un modo efficiente e preciso per analizzare i dati e migliorare la comprensione dell'attività neuronale. Uno dei possibili partner/utilizzatori di questo sistema è infatti la **NASA** che potrebbe far uso del software per missioni di esplorazione spaziale.

6.1 Lavori futuri

Release con tag e Semantic versioning

Il workflow attuale pubblica una nuova prerelease andando a sovrascrivere quella esistente ogni volta che viene fatto un push su main e tutti i controlli passano.

Questo comportamento non è ottimale, per questo motivo una delle prime modifiche che verranno effettuate su questo progetto sarà un controllo più accurato delle versini e dei tag che utilizzano il Semantic Versioning (SemVer), uno standard di numerazione delle versioni del software, che viene utilizzato per indicare in modo chiaro e univoco le modifiche apportate a un prodotto software. Il SemVer specifica che una versione del software deve essere composta da tre numeri separati da un punto: X.Y.Z. Il significato dei numeri è il seguente:

- X (Major): indica la versione principale del software. Viene incrementato quando vengono introdotte modifiche incompatibili con le versioni precedenti.
- Y (Minor): indica la versione di sviluppo. Viene incrementato quando vengono aggiunte funzionalità compatibili con le versioni precedenti.
- Z (Patch): indica la versione di correzione. Viene incrementato quando vengono apportate correzioni di bug o miglioramenti minori compatibili con le versioni precedenti.

Firma degli eseguibili

Gli eseguibili che vengono compilati non sono firmati, questo fa si che, una volta scaricati ed eseguiti un qualsiasi sistema operativo potrebbe sconsigliarne l'apertura dicendo che il software non è affidabile e potenzialmente pericoloso.

Supporto a macos

Github offre un budget mensile agli sviluppatori per automatizzare i propri processi, tuttavia, se si iniziano a fare test su più sistemi operativi, con n versini diverse di Python i minuti a disposizione calano rapidamente, oltre ciò i runner hanno un costo al minuto diverso basato sul sistema che si utilizza. In particolare i sistemi macos hanno un costo 10 volte superiore a quelli ubuntu. Dato che il progetto era ancora in una fase prototipale e che gli utilizzatori di sistemi macos in questa industria sono irrilevanti rispetto a quelli che utilizzano Windows, si è deciso di non utilizzare runner con macos. In futuro però sarebbe opportuno fornire un supporto adeguato anche su questa piattaforma.

Integrazione della ricerca di eventi

La prima cosa da fare sarebbe senza dubbio aggiungere l'algoritmo di ricerca degli eventi e integrarla con i moduli già presenti dell'applicazione. Ora come ora infatti questa funzionalità è stata implementata in maniera esterna al framework, anche se potrebbe essere molto utile in fase di analisi.

Librerie customizzate

Uno dei problemi più limitanti di Python sta nella sua lentezza. La stessa funzione scritta in Python e in C++/Rust è diversi ordini di granzezza più veloce negli altri linguaggi. Questo vuol dire che, potenzialmente, la sola riscrittura delle porzioni di codice più gravose in un altro linguaggio potrebbe comportare un miglioramento sostanziale nell'utilizzo di tutti i giorni. Con linguaggi come C/C++ o Rust scrivere pacchetti capaci di interoperare con Python è anche molto facile, per questo motivo è molto probabile che questa funzionalità verrà sfruttata in versioni future dell'applicazione, questo ci porta anche al prossimo punto.

Framework backend e Forntend

Sotto l'ottica di maggior robustezza e migliori performance si potrebbe anche pensare di aggiungere funzionalità in maniera incrementale alla libreria precedentemente menzionata, in modo da ottenere pian piano un vero e proprio framework (magari rilasciato pubblicamente), che permetta di creare la propria versione di forntend.

Aggiunta di un DB interno

Registrando qualche benchmark si è visto che con file di dimensioni molto grandi (più di 2 Gb) pc poco prestanti a livello di ram potrebbero faticare, soprattutto se si tratta di più file che poi arrivano a saturare la memoria volatile. Per questo motivo sarebbe estremamente utile istanziare un db in cui mantenere una versione intermedia dei dati non visualizzati. Questo svincolerebbe la ram dal dover riservare dello spazio per centinaia di Mb di dati, garantendo maggior stabilità al sistema.

Refactor dell'applicazione in termini di trasformazioni sui dati

Dato che la maggior parte delle trasformazioni sono quasi istantanee da calcolare e/o visualizzare, un refactor molto radicale del sistema che però mi piacerebbe attuare è il seguente. Anziché registrare tutti gli stati del dato, andando spesso a replicare molti dati, sarebbe molto bello ricordare la sequenza di trasformazioni applicate sui dati. Ad esempio se volessimo filtrare un file di 2 Gb, per 3 volte avremmo all'incirca un'occupazione di memoria di 2 + (2*3) = 8 Gb. Attuando il refactor discusso dovremmo solamente ricordare 2 Gb del dato originale + 2 Gb del dato filtrato. Utilizzando anche la strategia discussa precedentemente di salvataggio dei dati intermedi in un DB interno, potenzialmente arriveremmo ad un'occupazione di soli 2 Gb (quelli del file filtrato). Tuttavia questo refactor potrebbe essere molto costoso da portare a termine e il risparmio di memoria

dipenderebbe enormemente dalla lunghezza media delle sequenze di operazioni che vengono fatte sui dati. Se sono stati aperti cinque file ma non è stata fatta nessuna operazione su di essi, non si avrebbe nessun risparmio di memoria. Anche per questo motivo il primo cambiamento da realizzare sarebbe quello del DB, che ha un utilizzo più versatile rispetto a quello delle trasformazioni.

Bibliografia

[1] E. Frank, M. A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, and I. H. Witten. *Weka: A machine learning workbench for data mining.*, pages 1305–1314. Springer, Berlin, 2005.

[2] Harden. pyabf 2.3.5, 2022.

[3] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[4] Ken Healy, Birgitta Schiedt, and Alan P Morrison. Solid-state nanopore technologies for nanopore-based dna analysis. *Nanomedicine*, 2(6):875–897, 2007. PMID: 18095852.

[5] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[6] Hans Machemer. *Electrophysiology*, pages 185–215. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988.

[7] Thomas Nowotny and Vincenzo Marra. *Patch Clamp Technique*, pages 1–4. Springer New York, New York, NY, 2013.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] Masateru Taniguchi, Shohei Minami, Chikako Ono, Rina Hamajima, Ayumi Morimura, Shigeto Hamaguchi, Yukihiro Akeda, Yuta Kanai, Takeshi Kobayashi, Wataru Kamitani, Yutaka Terada, Koichiro Suzuki, Nobuaki Hatori, Yoshiaki Yamagishi, Nobuei Washizu, Hiroyasu Takei, Osamu Sakamoto, Norihiko Naono, Kenji Tatematsu, Takashi Washio, Yoshiharu Matsuura, and Kazunori Tomono. Combining machine learning and nanopore construction creates an artificial intelligence nanopore for coronavirus detection. *Nature Communications*, 12(1):3726, Jun 2021.

[10] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.