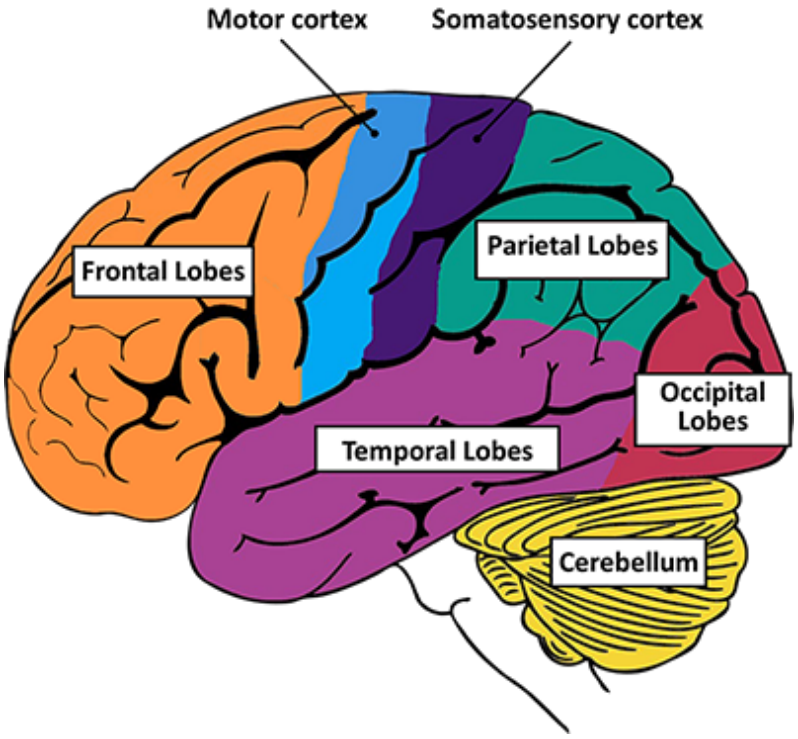
Bilag 5 - Baggrund

Dette bilag dokumenterer projektets teori mere dybdegående end rapporten og uddyber flere relaterede aspekter. Her introduceres relevant teori, matematik samt nogle af de anvendte AI-modeller i projektet forklares.

1. Kort om hjernen[1]

Hjernen er et af de mest komplekse organer i kroppen og er ansvarlig for at regulere både bevidste og ubevidste funktioner. Den består af forskellige strukturer og områder, der hver især har specifikke funktioner. Den motoriske cortex, som ligger i den frontallappen spiller en central rolle i at styre frivillige bevægelser. Den sender elektriske signaler til musklerne via rygmarven, hvilket muliggør præcise og koordinerede bevægelser. På figuren forneden ses et overblik af hjernen:



Figur 1: Overblik af hjernens struktur[2]

[1] Baseret på (Hansen, 2002, s. 1 - 45), (Andrusca, 2023) og (Hunt & Sugano, 2020)

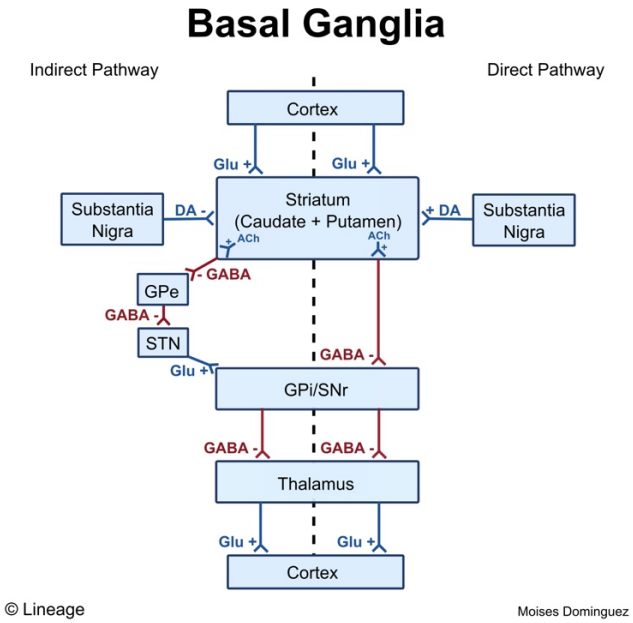
[2] Lånt af https://www.ninds.nih.gov/health-information/public-education/brain-basics/brain-basics-know-your-brain

Basal ganglierne[3] er centrale for motorisk kontrol og fungerer som et kompleks kredsløb, der integrerer signaler fra cortex og sender bearbejdet output tilbage via thalamus. Systemet består primært af: striatum (putamen og nucleus caudatus), globus pallidus (GPe og GPi), substantia nigra (pars compacta (SNc) og pars reticulata (SNr)) samt nucleus subthalamicus (STN).

For at kontrollere, justere og modulere bevægelser benytter basalganglierne tre hovedveje: Basalganglierne regulerer bevægelser gennem tre hovedveje:

- **Den direkte vej**: fremmer bevægelse ved at øge excitation af motorisk cortex (dopamin → D1-receptorer).
- **Den indirekte vej**: hæmmer bevægelse ved at øge inhibition af motorisk cortex (dopamin → hæmning via D2-receptorer i striatum).
- **Den hyperdirekte vej**: hurtigt hæmmer bevægelse via cortex → STN → GPi → thalamus (ikke dopaminmoduleret[4])

Ved enten at anvende exictoriske (Glutamat) og inhiboriske (GABA) signalstoffer, kan den overordnede effekt moduleres. For bedre at forstå disse veje, så lad os forestille en person der gerne vil bevæge sine ben, i form af en gåtur. På figuren forneden er den direkte og indirekte vist.



Figur 2: Den direkte og indirekte vej i de basal ganglier.

[3] Også kaldet basal nuclei

Direkte vej
Den direkte vej faciliterer bevægelse: Cortex sender glutamat signaler til putamen, som via GABA hæmmer de inhiberende outputneuroner i GPi og SNr. Da disse normalt hæmmer thalamus, fører denne hæmning til disinhibition af thalamus, som nu kan excitere motorisk cortex igen – og dermed starte bevægelse.

Dopamin fra SNc modulerer denne vej ved at aktivere D1-receptorer i striatum, hvilket forstærker den excitatoriske effekt på bevægelse. Uden tilstrækkelig dopamin (som ved Parkinsons-sygdom) reduceres aktiviteten i den direkte vej, hvilket resulterer i motorisk hypokinesis og bradykinesis.

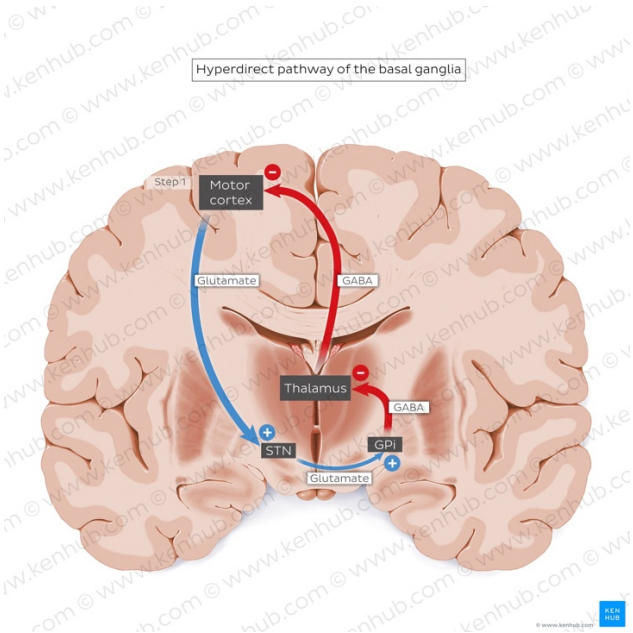
I den direkte vej sker der altså en positiv feedback, fordi de to inhibitoriske synapser er forbundet direkte. Det betyder GABA fra putamen undertrykker GABA fra GPi, og dermed reduceres den inhiberende effekt af GPi på thalamus, som kaldes disinhibition. Et neuron hæmmer et hæmmende neuron, som slutvist giver en aktiverende effekt.

Indirekte vej
Samtidig med den direkte vej aktiveres, kan også den indirekte vej i basalganglierne være aktiv. Her sender den motoriske cortex først glutamat til putamen (ligesom i den direkte vej). Putamen sender derefter inhibitoriske GABA-signaler til GPe.

Dette hæmmer GPe, som normalt hæmmer STN. Når GPe hæmmes, sker en disinhibition af STN og kan derfor nu sende excitoriske glutamat-signaler til GPi og SNr. Disse hæmmer så thalamus via GABA. Når thalamus hæmmes, reduceres excitatorisk output til motorisk cortex, hvilket hæmmer bevægelse.

Dopamin fra SNc modulerer denne vej ved at binde til D2-receptorer i striatum, hvilket hæmmer aktiviteten i den indirekte vej. Dvs. dopamin svækker bevægelseshæmning og dermed fremmer bevægelse. Ved dopaminmangel (som i Parkinsons-sygdom) bliver den indirekte vej overaktiv, hvilket bidrager til bradykinesis og rigiditet.

Hyperdirekte vej
Glutamat sendes direkte fra motor cortex til STN, som giver stærke exitoriske signaler til GPi og SNr. Da GPi sender inhiberende GABA til thalamus, sker en netto hæmning af thalamus, som reducerer det excitatoriske output til motorisk cortex, hvilket hæmmer bevægelse. Den hyperdirekte vej spiller en vigtig rolle i hurtig bevægelsesinhibering, f.eks. når man skal undertrykke en igangsat bevægelse eller impulshandling – en slags "emergency brake". På figuren forneden, vises den hyperdirekte vej:



Figur 3: Den hyperdirekte vej[5]

Relateret til Parkinsons-sygdom

En central årsag til de motoriske symptomer ved Parkinsons-sygdom er tabet af dopamin-producerende neuroner i substantia nigra pars compacta. Dette dopamintab svækker den direkte vej i basalganglierne (som normalt faciliterer bevægelse via D1-receptorer) og forstærker den indirekte vej (som hæmmer bevægelse via D2-receptorer). Resultatet er en overinhibering af thalamus, hvilket reducerer excitatorisk output til motorisk cortex og dermed hæmmer igangsættelsen af bevægelser. Bevægelse bliver dermed langsom, stiv eller helt fraværende.

Denne dysfunktion i de basale ganglier afspejles også i hjernens elektriske aktivitet. Hos Parkinsons-patienter ser man typisk en reduktion i højfrekvente oscillationer (alfa, beta og især gamma), som normalt er forbundet med bevægelsesforberedelse og fleksibel netværkskommunikation. Samtidig ses en stigning af lavfrekvente bånd (især delta og theta), som indikerer mere rigid og mindre responsiv netværksaktivitet.

Højfrekvente bølger kræver en vis grad af desynkronisering mellem neuronpopulationer for at muliggøre fleksibel signalbehandling. Denne desynkronisering forudsætter både tilstrækkelig excitation (bl.a. fra thalamus) og energitilgængelighed - begge dele svækkes ved dopaminmangel. Resultatet er, at neurale netværk forbliver i synkroniseret hvileaktivitet, hvilket hæmmer opbygning af målrettet motorisk aktivitet.

[5] Lånt af https://www.kenhub.com/en/library/anatomy/direct-and-indirect-pathways-of-the-basal-ganglia

Kort sagt: Dopaminmangel fører til ubalance i basalgangliernes kredsløb, reducerer excitatorisk feedback til motorisk cortex, hæmmer kortikal desynkronisering og medfører et skift mod lavfrekvent oscillation – hvilket samlet bidrager til de motoriske symptomer ved Parkinsons-sygdom.

Men i de tidlige stadier af Parkinsons sygdom kan gamma-konnektivitet være øget som en kompensatorisk mekanisme, selvom dopaminniveauerne begynder at falde[6]. Dette skyldes, at kortikale netværk stadig har tilstrækkelig excitation og fleksibilitet til at opretholde højfrekvent kommunikation. Men når dopaminmanglen bliver størrre, svækkes den excitatoriske input til cortex gennem thalamus, og gamma-aktiviteten kollapser. Dette medfører reduceret netværksfleksibilitet og øget dominans af lavfrekvente og patologiske bånd – hvilket korrelerer med de motoriske symptomer i avanceret Parkinson.

2. Signalbehandling[7]

2,1) Fouriertransformation

Nogle gange er det mere relevant at beskrive en funktion i frekvensdomænet fremfor tidsdomænet. Fouriertransformationen giver en metode til at transformere en tidsafhængig funktion, $f(t)$, til en frekvensafhængig funktion $F(\omega)$.

Idéen bag Fouriertransformationen er, at alle signaler kan opdeles i en sum af sinus- og cosinusfunktioner med forskellige frekvenser og amplituder. Hvis vi har funktionen $f(t)$ i kompleks form, gælder:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega) \cdot e^{\omega t \cdot i} d\omega$$

Her er i det imaginære tal $\sqrt{-1}$ og ω er vinkelfrekvsen $\omega = 2\pi f$, hvor f er frekvensen i Hz.

Fouriertransformationen defineres som,

[6] Som vist i (Conti & et al., 2022)
[7] Primære kilder i dette afsnit er: (wikipedia.org, 2024), (Jensen & Jensen, 2019), (Google colab, u.d.) (Google colab, u.d.) (geeksforgeeks, 2022)

$$\mathcal{F}\{f(t)\} = F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-\omega t \cdot i} dt$$

Ligeledes kan man gå den anden vej, fra frekvensdomænet over til det tidslige domæne, ved at tage den inverse Fouriertransformation.

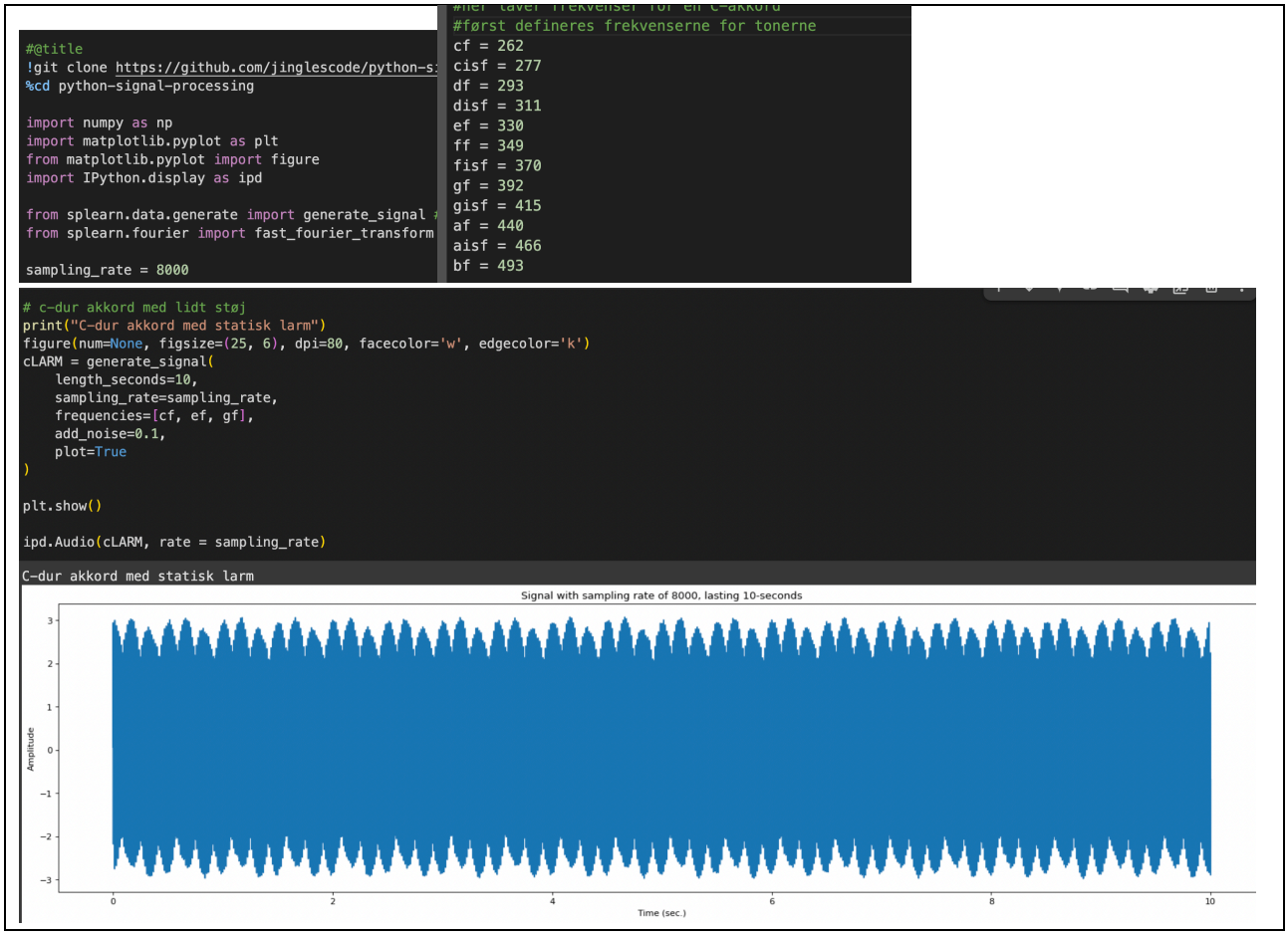
$$\mathcal{F}^{-1}\{F(\omega)\} = f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) \cdot e^{\omega t \cdot i} d\omega$$

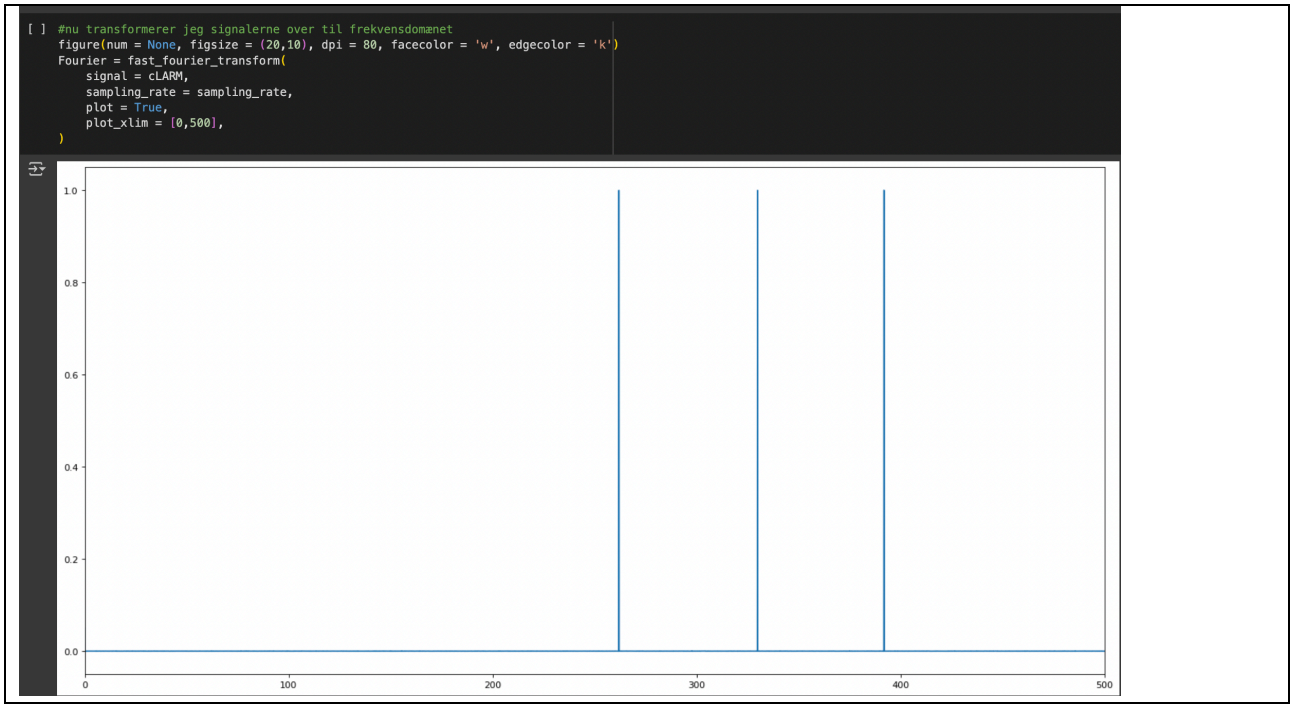
Fouriertransformationen er en bijektiv transformation, hvilket betyder, at der findes en entydig omvendt transformation (den inverse Fouriertransformation), som genskaber originalsignalet uden tab.

2,1,1) Fast Fourier Transform (FFT)

I praksis anvender man ofte diskrete data, f.eks. fra en EEG-måling, og her bruges den numeriske algoritme FFT. FFT er en effektiv metode til at beregne Fouriertransformationen for diskrete datapunkter og gør det muligt at analysere signaler hurtigt og præcist.

Forneden vises et eksempel på FFT anvendt på akkorden C-dur med ekstra tilføjet støj:





Således kan FFT anvendes til at undersøge signaler og finde frekvenserne i disse.

2,1,2) Praktisk anvendelse

Ved hjælp af Fouriertransformationer kan signaler som EEG analyseres. For eksempel kan man identificere bestemte frekvenser relateret til hjernens Bereitschaftspotentialer eller filtrere støj. FFT gør det muligt at analysere på store datasæt hurtigt og præcist. Fouriertransformationer og FFT er derfor essentielle værktøjer i dette projekt.

1,2) Filtrering af støj

Filtrering er en essentiel del af signalbehandling, især når man arbejder med biologiske signaler som EEG og EMG, der ofte indeholder betydelige mængder støj. Formålet med filtrering er at forbedre signalets kvalitet ved at undertrykke støj uden at miste de relevante karakteristika i datasættet. Flere metoder bruges afhængigt af støjkilden og det ønskede resultat.

2,2,2) Båndpas-filtrering,

Båndpas-filtrering anvendes til at isolere bestemte frekvensbånd ved at tillade frekvenser i et givent interval at passere, mens resten dæmpes. Dette kan være nyttigt til at isolere de relevante frekvenser for EEG eller EMG signaler - f.eks. μ-bølger og betabølger. Højpas- og lavpasfiltre bruges ofte til at fokusere på bestemte frekvenser i datasættet.

2,2,3) Baseline-korrektion

Baseline-korrektion bruges til at normalisere signalet ved at justere det til en baselineperiode, som typisk defineres som en periode før en stimulus (fra -0,2 til 0 sekunder). Dette sikrer, at signalet er justeret for eventuelle systematiske driftsændringer.

2,2,4) Notch-filter

Et Notch-filter er et smalbåndsfilter, der dæmper en specifik frekvens uden at påvirke nærliggende frekvenser signifikant. Det bruges ofte til at fjerne elektrisk støj på 50/60 Hz fra netspænding

2,2,5) Savitzky-Golay-filter[8]

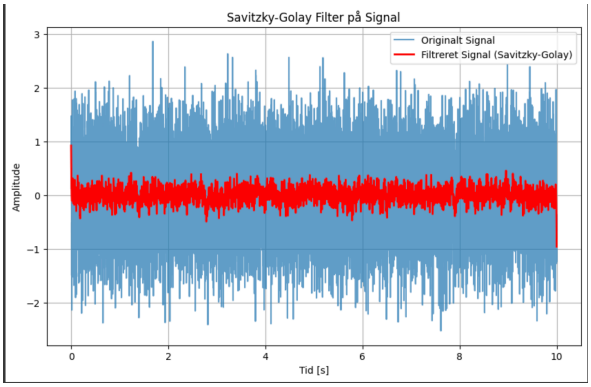
Savitzky-Golay-filteret er en metode til at glatte et ud signal og fjerne støj, ved at tilpasse et polynomium til signalet over et glidende vindue. Dette filter er effektivt til at fjerne højfrekvent støj, mens det bevarer signalets hovedstrukturer. På figuren forneden ses et eksempel hvor Savitzkt-Golay-filteret anvendes:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import savgol_filter

# Simuler et signal: sinusbølge med støj
fs = 1000  # Sampling frekvens (Hz)
t = np.arange(0, 10, 1/fs)  # Tidsvektor på 10 sekunder
signal = np.sin(2 * np.pi * 50 * t) + 0.5 * np.random.normal(size=len(t))  # Sinusbølge med støj

# Anvend Savitzky-Golay filter på signalet
# Vinduesstørrelse på 51 og polynomiets grad på 3
filtered_signal = savgol_filter(signal, window_length=51, polyorder=3)

# Visualiser originalt og filtreret signal
plt.figure(figsize=(10, 6))
plt.plot(t, signal, label='Originalt Signal', alpha=0.7)
plt.plot(t, filtered_signal, label='Filtreret Signal (Savitzky-Golay)', color='red', linewidth=2)
plt.xlabel('Tid [s]')
plt.ylabel('Amplitude')
plt.title('Savitzky-Golay Filter på Signal')
plt.legend()
plt.grid(True)
plt.show()
```



Figur 4: Eksempel på anvendelsen af Savitzky-Golay-filter

[8] (wikipedia.org, 2024)

2,2,6) Spektrogram (Tids-frekvensanalyse)

Spektrogrammet er en metode til at analysere signaler i både tid og frekvens. Det opdeler signalet i korte tidsvinduer og beregner FFT for hver sektion, hvilket gør det muligt at visualisere, hvordan frekvenskomponenterne af signalet ændrer sig over tid. Dette er især nyttigt for at identificere tidsafhængige frekvensmønstre i EEG eller EMG data. På figur 11 ses koden og et eksempel på et spektrogram:

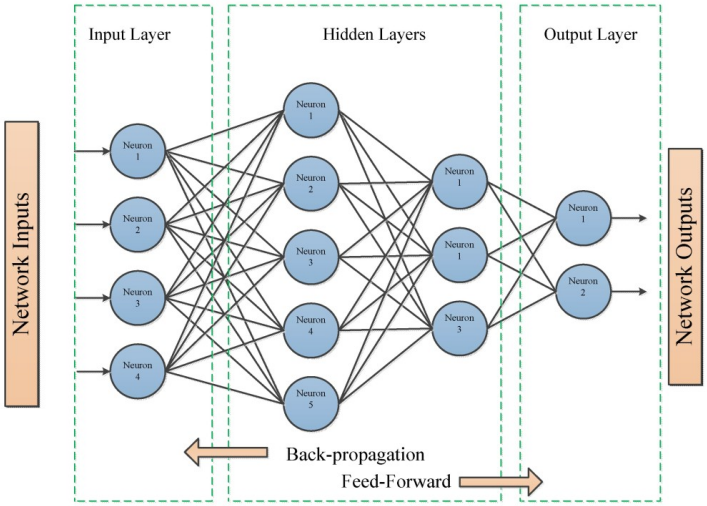
3. Neuralt netværk (NN)[9]

Et neuralt netværk er en machine learnings-model, der er inspireret af den menneskelige hjernes struktur og funktion. Modellen består af forbundne *neuroner*, der tager data ind for at lære, genkende mønstre og klassificere data.

De vigtigste komponenter i et NN er:

- **Neuroner**: enheder der modtager data og aktiveres af en aktiveringsfunktion
- **Forbindelser/synapser**: kæder mellem neuroner, styret af vægte og biaser
- **Vægte og biaser**: parametre der bestemmer forbindelsers styrke og indflydelse
- **Aktiveringsfunktioner**: funktioner der bearbejder data og overfører dem gennem lagene
- **Læringsregel**: metode, der justerer vægte og biaser under træning for at forbedre nøjagtigheden

Et neuralt netværk består af **inputlag**, **skjulte** lag og **outputlag**, som arbejder sammen for at lære mønstre i data. Inputlaget modtager rådata, hvor hvert neuron repræsenterer en feature, fx en pixelværdi eller en variabel. Data sendes videre til **skjulte lag**, der bearbejder det ved hjælp af vægte, biases og aktiveringsfunktioner som ReLU eller sigmoid. Disse lag identificerer komplekse mønstre i dataen, såsom kanter i billeder eller trends i tidsserier.
På figuren forneden kan strukturen af et NN ses:



Figur 5: Grafisk forklaring af et neuralt netværk[10]

Outputlaget genererer det endelige resultat. Antallet af neuroner her afhænger af opgaven: klassifikation kræver fx én neuron pr. klasse, mens regression typisk bruger én neuron til en kontinuerlig værdi.

---

[9] Afsnittet er baseret på kilderne: (3Blue1Brown, 2024), (IBM, u.d.), (Geeksforgeeks, 2024)
[10] Lånt af https://www.linkedin.com/pulse/feedforward-vs-backpropagation-ann-saffronedge1/

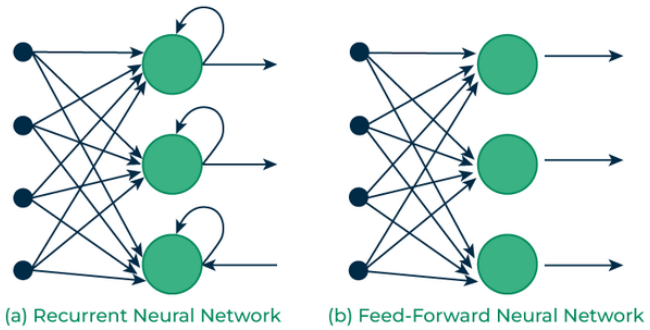
Netværket lærer gennem en iterativ proces. Ved **fremadpropagering** sendes data én vej gennem lagene for at producere en forudsigelse. Fejlen mellem forudsigelsen og det ønskede resultat beregnes, og ved **bagudpropagering** justeres vægte og biases for at minimere denne fejl. Over mange iterationer forbedrer netværket gradvist sin nøjagtighed og evne til at genkende mønstre.

Man har et **inputlag,** hvor der modtages data og hvert input neuron er tilkoblet en feature i inputdataet. Et **skjult lag** der bebrejder data gennem transformationer for at lære komplekse mønstre i data og et **outputlag** som genererer det endelige resultat. Data hopper tit frem og tilbage mellem inputlag og gemte lag, for konstant at ittere sin bearbejdelse og mønstergenkendelse i dataet. Evaluering af NN's output sker vha. en *tabfunktion*, som måler forskellen mellem NN's output og det korrekte resultat

3,1) Recurrent Neural Network (RNN) og Long Short-Term Memory (LSTM)[11]

Et RNN er et NN, der er designet til at håndtere sekventielle data, hvor rækkefølgen af input er vigtig. I modsætning til normale feedforward-netværk, som kun tager én input ad gangen, indeholder RNN'er feedback-løkker, der gør det muligt for netværket at "huske" information fra tidligere tidspunkter i sekvensen. Dette sker gennem en skjult tilstand, kaldet hukommelsestilstanden, som opdateres ved hver tidsstep og bevarer vigtig information fra tidligere inputs.

På figuren herunder viser forskellen mellem et feedforward-netværk og et RNN:



Figur 6: Forskellen mellem feedforward netværk og RNN'er[12]

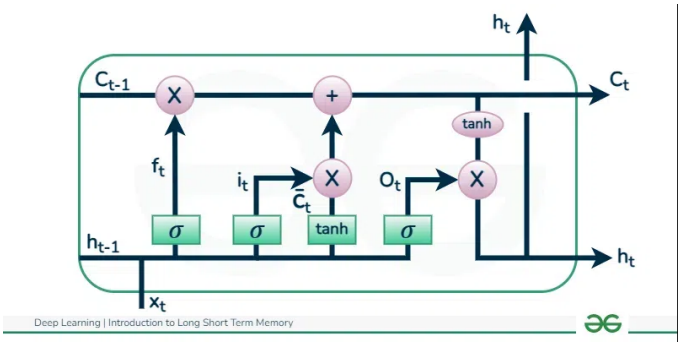
[11] Afsnittet er baseret på kilderne: (Hamad, 2023), (geeksforgeeks, 2024), (geeksforgeeks, 2023), (Wikipedia, 2025). Matematikken bag LSTM nævnes kort i bilag 5.
[12] Lånt af https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

RNN'er har dog en tendens til at være ineffektive til at håndtere lange sekvenser, da hukommelsestilstanden gradvist mister vigtig information. Dette skyldes det såkaldte **forsvindende gradient-problem**, hvor gradienterne, der bruges til at opdatere vægtene under træning, bliver meget små. Dette forhindrer netværket i at lære langtidshukommelse og afhængigheder i sekvenser.

For at løse dette problem blev LSTM (Long Short-Term Memory) introduceret. LSTM-netværk indeholder en **hukommelsescelle**, som gør det muligt for netværket at bevare informationer over længere tid. I LSTM-cellen reguleres hukommelsen af tre porte, der styrer, hvilke informationer der skal opbevares og glemmes:

1. **Input port (i_t, $\widehat{C}_t$, C_t)**: kontrol af hvilken information der tilføjes hukommelsescellen
2. **Forget port (f_t)**: bestemmer hvilke informationer i hukommelsescellen der skal bibeholdes eller slettes, baseret på tidligere input
3. **Output port (o_t)** : kontrol af hvilken information der skal være output i hukommelsescellen

LSTM'er er især nyttige til opgaver som sprogmodellering (f.eks. ChatGPT) eller behandling af tidsseriedata med lange intervaller, som er relevant for dette projekt. På figuren herunder ses arkitekturen af et LSTM

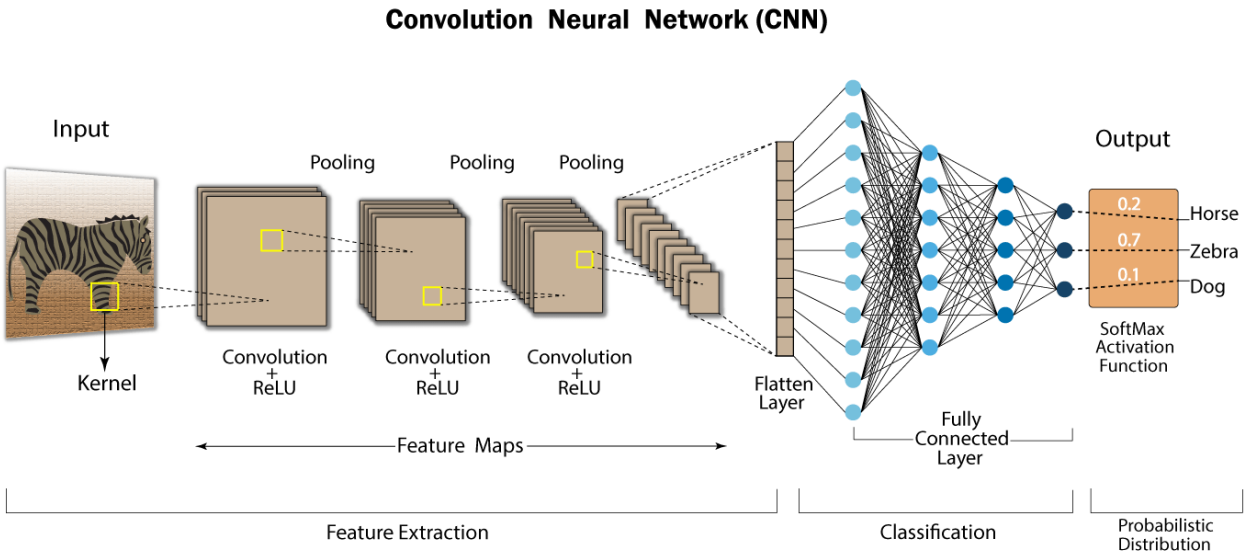


Figur 7: Arkitekturen bag LSTM[13]

3,2) Convolutional Neural Network (CNN)[14]

Et Convolutional Neural Network (CNN) er en type neuralt netværk, der er specialiseret i behandling af data med en gitterstruktur, f.eks. billeder eller signaler fra sensorer. CNN'er er opbygget til at efterligne menneskets visuelle system, hvor neuroner reagerer på specifikke regioner, kaldet receptive felter. I CNN identificeres simple mønstre som linjer og kanter og mere komplekse elementer som former og objekter gennem flere lag.
På figuren herunder ses arkitekturen bag et CNN:

[13] Lånt af https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/
[14] Afsnittet er baseret på kilderne: (Keita, 2023), (Jorgecardete, 2024), (ibm, u.d.), (Mishra, 2020)



Figur 8 : Arkitekturen bag CNN[15]

Et CNN består typisk af tre hovedlag:

- **Convolution**-lag: Her udføres en konvolutionsoperation, hvor små, lærebare filtre (kerner) scanner dataens gitterstruktur for at fremhæve mønstre. Resultatet af processen er et aktiveringskort, der repræsenterer de fundne mønstre.
-
- **Pooling-lag**: Pooling reducerer størrelsen på aktiveringskortene ved at opsummere nærliggende værdier, ofte ved hjælp af maksimering (max-pooling). Dette reducerer datamængden og skaber mere robuste repræsentationer.

- **Fuldt forbundet lag**: Dette lag kombinerer de udtrukne funktioner og skaber forbindelser mellem input og output, hvilket typisk bruges til klassifikation eller regression.

[15] Lånt af https://ai.plainenglish.io/deep-dive-into-the-world-of-cnns-8cf22cd84e7

4. Transformer-modellen[16]

En transformer er et sofistikeret dybt neuralt netværk, der arbejder parallelt med data, fremfor sekventielt som i en LSTM. Den gør dette ved hjælp af self-attention, som hjælper modellen med at fokusere på de vigtigste dele af dataen – uanset hvor de befinder sig. Det betyder, at den kan forstå sammenhænge i lange sekvenser meget effektivt.

De forskellige datagrupper, modellen modtager, kaldes for modaliteter. Haves f.eks. data som EEG, EMG og IMU, så er hver af disse sin egen modalitet, og modellen består af 3 modaliteter. Vi har altså en multimodal-model. Vigtigt er, at modaliteter ofte kræver forskellige præprocessering trin, før de kan fusioneres i en transformer. Modaliteternes signaler skal altså filtreres og de mest relevante features udtrækkes

- EEG: Lavpasfiltrering, Notch (50/60 Hz støj), ICA (for at fjerne øjenblink-artefakter) er relevante.
- EMG: Envelope-ekstraktion er en almindelig teknik, men ICA bruges sjældent her.
- IMU: Har normalt ikke brug for ICA eller envelope-ekstraktion, men snarere signalderivering for hastighed og acceleration.

En transformer arbejder med vektorer, og derfor transformeres modaliteterne til de nødvendige vektorielle repræsentationer. Man har nu en masse vektorer, som vi gerne vil normalisere og projicere til et fælles vektorrum vha. en lineær transformation. På den måde repræsenteres alle signaler nu ved en d-vektor, hvor d angiver en såkaldt ”embedding-dimmension”. En embedding er en transformation fra en rå input-repræsentation (fx sensor-data) til en mere semantisk meningsfuld vektor i et højdimensionelt rum, hvor lignende inputs ligger tættere på hinanden. Embedding hjælper modellen med at lære en kompakt og semantisk meningsfuld repræsentation af signalerne.

Da vores inputsignaler nu har mistet deres tidlige tidslige dimension, og transformeren ingen fornemmelse har af den sekventielle rækkefølge længere, introduceres positional encoding. Positional encoding er en metode til at give en relativ position for hver token i en transformer, hvor en token er de projicerefr data i vores d-vektor. En almindelig teknik er cosinus- og sinusfunktioner eller learnable position embeddings (hvor modellen lærer dem under træning).

Disse tokens skal nu gennem transformer-modellens encoder, hvor de kommer gennem forskellige antal encoding-blokke, som hver især består af:

[16] Kilder: (3Blue1Brown, 2024), (Wikipedia, 2025), (Geeksforgeeks, 2024), (Sachinsoni, 2024)

1. **Multi-Head Self-Attention (MHSA)**
   MHSA fungerer ved at transformere inputvektorer til Query, Key, og Value, hvor opmærksomhedsscoren beregnes som en vægtning mellem Query og Key, og derefter anvendes på Value. Dette betyder, at modellen kan fokusere på de mest relevante dele af sekvensen på tværs af modaliteter.

2. **Feedforward Neural Network (FNN)**
   Efter MHSA sendes outputtet gennem et FNN, som skaber en mere kompleks repræsentation af signalet. Dette anvendes pr. token, og transformerer embeddings til mere abstrakte repræsentationer.

3. **Residual Connections & Layer Normalization**
   Residual connections sikrer, at information fra tidligere lag bevares, hvilket hjælper med gradient flow i dybe netværk. Layer normalization stabiliserer aktiveringer og forbedrer træningskonvergens

Disse encodingblokke samles i lag på 4-6 blokke ovenpå hinanden, for at lære en dybere repræsentation af tokens.

Efter encoding kan modaliteter sammenføjes gennem concatenation, hvor deres embeddings kombineres direkte, eller via attention-baserede mekanismer såsom cross-modal attention, hvor transformeren lærer vægtningen af hver modalitet dynamisk. Når transformer-encodingens sidste lag er beregnet, kan en repræsentation af hele tidsvinduet udtrækkes enten via CLS-token, gennemsnits-pooling eller en vægtet sum af token-repræsentationerne

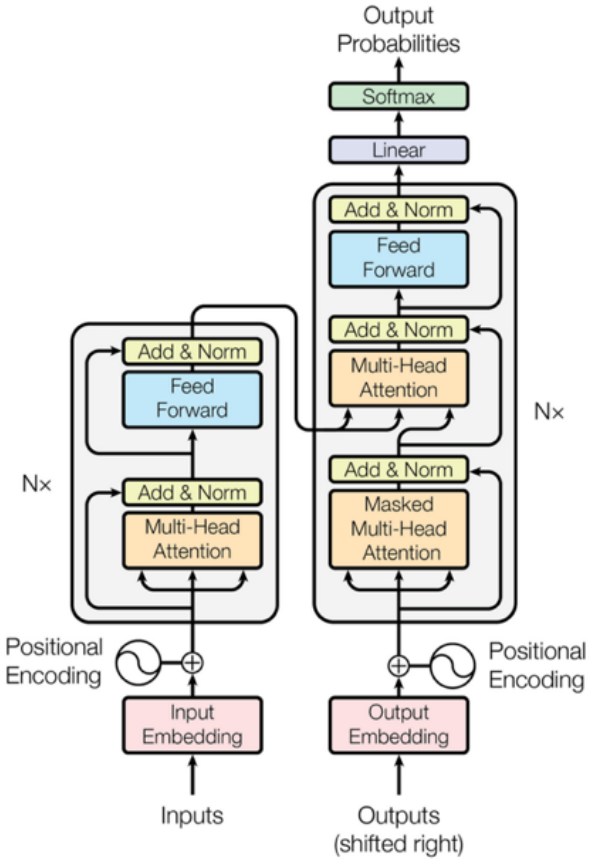
1. **Flatten & Concatenation**[17]
   Transformerens output fra de 3 modaliteter sammenkædes til en samlet vektor

2. **Dense Layers & Softmax**
   Et helt forbundet neuralt netværk bruges til at træffe en beslutning baseret på den behandlede information. Altså man forudsiger output baseret på input.

Output kan f.eks være:
- Klassifikation (hvilken type af bevægelse)
- Regression (vinkel på en bevægelse)

På figuren forneden ses pipelinen af en transformer:

[17] Et andet ord for sammenkædning, hvor man får et nyt string bestående de to originale.



Figur 9: Overblik af flowet i en transformer-model[18]

Og således fungerer en transformer, der i dette projekt skal klassificere et EEG-signal som en konkret bevægelse, hvor output bliver en binær-talværdi repræsentation for hver af disse bevægelser. Modellen modaliteter er: EEG, EMG og IMU.

5. Matematikken bag projektet

Fremad propagering

Matematikken bag fremadpropagering:

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b$$

Her er:

- z er det samlede aktiveringsinput til en neuron, som er den lineære kombination af vægtede input plus bias

[18] Lånt af https://pub.towardsai.net/transformer-architecture-part-1-d157b54315e6

- w_i er vægten tilknyttet input x_i. Vægtene bestemmer, hvor meget hvert input bidrager til det samlede output.
- x_i er den i'te inputværdi. Dette er den rå data, der sendes ind i neuronet (f.eks. en pixelværdi eller en sensoraflæsning)
- b er et bias term. Bias er en konstant, som justerer aktiveringsniveauet, så neuronet kan lære forskydninger i data

Når z er beregnet sendes den tit gennem en aktiveringsfunktion, så neuronet kan lære ikke-lineære mønstre

Aktiveringsfunktioner
Aktiveringsfunktioner introducerer ikke-linearitet i dybe neurale netværk og spiller en afgørende rolle i modellens evne til at lære komplekse mønstre. En aktiveringsfunktion bestemmer outputtet for en neuron, givet dens input.

Et simpelt eksempel kan være et kredsløb, hvor aktiveringsfunktionen afgør, om kredsløbet er tændt (ON) eller slukket (OFF) baseret på de inputsignaler, det modtager.

ReLU (Rectified Linear Unit)
Den mest anvendte aktiveringsfunktion i dybe neurale netværk er Rectified Linear Unit (ReLU), som er defineret ved:

$$f(z) = \max(0, z)$$

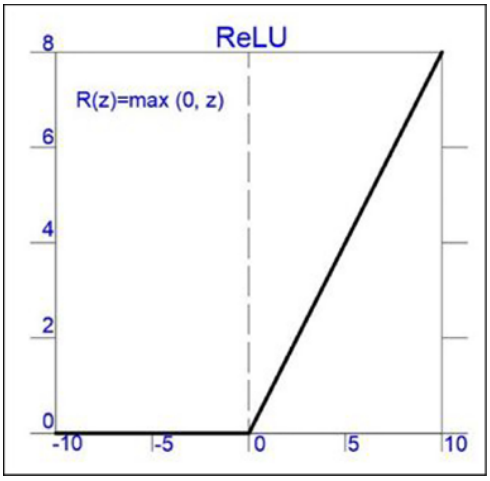
Hvilket betyder

$$f(z) = \begin{cases} 0, & \text{hvis } z \leq 0 \\ z, & \text{hvis } z > 0 \end{cases}$$

Den afledte funktion (gradienten) er givet ved,

$$f'(z) = \begin{cases} 0, & \text{hvis } z < 0 \\ 1, & \text{hvis } z > 0 \end{cases}$$

hvilket viser, at ReLU ikke er differentiabel ved $z = 0$. I praksis sættes gradienten ved dette punkt ofte til 0 eller 1 afhængigt af implementeringen. På figuren nedenfor, ses grafen for ReLU



Figur 10: Graf for ReLU[19]

Egenskaber ved ReLU

Værdimængden:

$$f(z) \in [0, \infty)$$

hvilket betyder, at alle negative input sættes til 0, mens positive værdier forbliver uændrede.

Effektiv beregning:

ReLU er hurtig at evaluere sammenlignet med sigmoide- og tanh-funktioner, da den kun kræver en simpel maksimumsoperation.

Bagudpropagering:

Ved træning af et neuralt netværk skal aktiveringsfunktionen differentieres i bagudpropageringen. For ReLU gælder:

$$\text{Hvis } z > 0, \text{er gradienten } 1$$
$$\text{Hvis } z < 0, \text{er gradienten } 0$$
$$\text{Hvis } z = 0, \text{er gradienten ikke defineret, men sættes typisk til } 0 \text{ eller } 1$$

Denne egenskab gør ReLU mindre følsom over for gradientproblemer, men kan føre til "dying ReLU", hvor neuroner kan blive permanent inaktive, hvis deres vægte fører til negative input. For at afhjælpe 'dying ReLU'-problemet kan man anvende Leaky ReLU, hvor negative værdier får en lille hældning i stedet for at blive sat til nul.

Sigmoid:

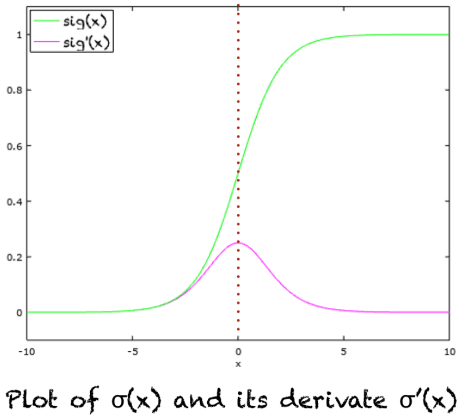
[19] Lånt af https://www.researchgate.net/figure/Activation-function-ReLu-ReLu-Rectified-Linear-Activation_fig2_370465617

En anden aktiveringsfunktion er Sigmoid-funktionen. Denne er glat, kontinuert og differentiabel i hele dens domæne. Sigmoid er en matematisk funktion med en S-formet kurve, der komprimerer inputværdier til et interval mellem 0 og 1. Funktionen er givet ved

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Her er:
z er inputtet (som er den vægtede sum af neuronens inputværdier
e er eulers tal.

På figuren forneden ses Sigmoidfunktionen, og den afledte:



Domain: (−∞, +∞)
Range: (0, +1)
σ(0) = 0.5

Other properties

$$\sigma(x) = 1 - \sigma(-x)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^{x}}{e^{x} + 1}$$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Figur 11: Graf for Sigmoidfunktion og den afledte

Egenskaber ved Sigmoid Funktion
Outputtet ligger altid mellem 0 og 1, hvilket gør den velegnet til problemer, hvor man ønsker en sandsynlighed som output, f.eks. ved klassificering af data eller billeder.

Da $\sigma(0) = 0{,}5$, er funktionen symmetrisk omkring $z = 0$. Det betyder, at positive værdier skubbes mod 1, mens negative skubbes mod 0.

Som aktiveringsfunktion bruges Sigmoid til at introducere ikke-linearitet, blandt andet i binær klassifikation i output-laget af et neuralt netværk. Her anvendes Sigmoid i den sidste neuron i netværket, der klassificerer mellem to klasser. Da outputtet altid er mellem 0 og 1, kan det tolkes som en sandsynlighed for, at en prøve tilhører en bestemt klasse. Sigmoid anvendes også i logistisk regression, hvor den estimerer sandsynligheden for en binær hændelse.

Et problem ved Sigmoid opstår, når z enten bliver meget stor eller meget lille. I disse tilfælde opstår "The vanishing gradient problem", hvor gradienterne i bagudpropageringen bliver ekstremt små. Dette hæmmer læring i dybe neurale netværk, da vægtjusteringerne bliver minimale. På grund af dette problem anvendes ReLU oftest i stedet, ligesom det gøres i dette projekt.

Tab funktioner[20]

Evaluering af et neuralt netværks output sker ved hjælp af en tabfunktion, som måler forskellen mellem netværkets forudsagte output og de faktiske værdier. I dette projekt er følgende tabfunktioner anvendt:

Mean Absolute Error (MAE),

MAE[21] er en tabfunktion, der anvendes i regressionsmodeller til at måle forskellen mellem forudsagte værdier $\hat{y}$ og faktiske værdier y MAE beregnes som gennemsnittet af de absolutte fejl mellem de forudsagte og faktiske værdier:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_1 - \hat{y}_i|$$

Her er:

- n er antal datapunkter
- y_i er den faktiske værdi
- $\hat{y}_i$ er den forudsagte værdi
- $|\hat{y}_i - y_i|$ er den absolutte fejl

MAE bruges i neurale netværk til at minimere fejlene mellem forudsagte og faktiske outputværdier ved at optimere netværkets vægte, så MAE reduceres over tid.

En høj MAE indikerer, at der er stor forskel mellem det forudsagte og det faktiske output, mens en lav MAE indikerer en mere præcis model. Derfor bør MAE være så lav som muligt for at sikre en brugbar og præcis model.

MAE har dog nogle ulemper. Den straffer store fejl mindre end MSE, hvilket kan være en ulempe, hvis store fejl er særligt problematiske. Derudover er MAE ikke differentiabel i $z = 0$, hvilket kan gøre gradientbaserede optimeringsalgoritmer mindre effektive.

Mean Squared Error (MSE)

Ligesom MAE er Mean Squared Error (MSE) en tabfunktion, der ofte anvendes i neurale netværk og regressionsmodeller til at måle forskellen mellem forudsagte værdier $\hat{y}$ og faktiske værdier y. MSE beregnes som gennemsnittet af de kvadrerede fejl mellem de forudsagte og faktiske værdier:

[20] (IBM, 2024)

[21] (Wikipedia, 2025),

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Her er:

- n antal datapunkter
- y faktiske værdi
- $\hat{y}$ forudsagte værdi
- $(y_i - \hat{y}_i)^2$ den kvadrerede fejl

MSE måler den gennemsnitlige kvadrerede afvigelse mellem de forudsagte og faktiske værdier.

I neurale netværk bruges MSE til at minimere fejlene ved at optimere vægtene, så den gennemsnitlige kvadrerede fejl reduceres over tid. En lav MSE indikerer en præcis model, mens en høj MSE indikerer, at modellen laver store fejl.

En ulempe ved MSE er, at den straffer store fejl kraftigere end MAE, fordi fejlene kvadreres. Hvis et dataset indeholder outliers, kan disse dominere tabet og føre til ustabil læring. Desuden har MSE enheden kvadreret i forhold til inputdataene, hvilket kan gøre den vanskeligere at fortolke.

Huber loss

Huber Loss[22] er en kombination af MSE og MAE, hvor den skifter mellem de to afhængigt af en forudbestemt tærskel δ. Den bruges ofte i regressionsmodeller, hvor man ønsker at straffe store fejl kraftigere end MAE, men samtidig ikke lade outliers dominere læringsprocessen, som det kan ske med MSE.

Huber Loss er defineret som,

$$L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & \text{hvis } |y - \hat{y}| \leq \delta \\ \delta(|y - \hat{y}|) - \frac{1}{2}\delta, & \text{hvis } |y - \hat{y}| > \delta \end{cases}$$

Her er,

- y den faktiske værdi
- $\hat{y}$ den forudsagte værdi
- δ en tærskel, der bestemmer hvornår tabet skifter fra MSE til MAE

Når **fejlen** er **lille** $|y - \hat{y}| \leq \delta$, fungerer Huber loss som MSE

[22] (Wikipedia, 2024)

$$L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2, \qquad \text{lille fejl}$$

Det betyder, at små fejl straffes kvadreret, hvilket sikrer en glat gradient og præcis læring.

Når **fejlen** er **stor** $|y - \hat{y}| > \delta$ fungerer Huber loss som MAE

$$L(y, \hat{y}) = \delta(|y - \hat{y}|) - \frac{1}{2}\delta, \qquad \text{stor fejl}$$

Her overgår tabet til en lineær straf, hvilket gør den mindre følsom over for outliers.
Huber Loss er en meget effektiv tabfunktion, der balancerer præcision med robusthed, fordi den kombinerer MSE og MAE i én funktion med en glidende overgang mellem dem.
Derfor er Huber Loss også anvendt i dette projekt, da den tager det bedste fra begge verdener og sikrer stabil læring, selv hvis der er outliers i dataene.

Optimeringsalgoritmer:
Adam (Adaptive Moment Estimation) er en adaptiv optimeringsalgoritme, der kombinerer momentumbaseret gradientopdatering (β_1) med RMSProp-lignende tilpasning af læringsraten ($\beta_2 2$), hvilket resulterer i stabil og hurtig konvergens
Beregner adaptiv læringsrate for hver parameter.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla\theta_t$$
$$v_t = \beta_2 m_{t-1} + (1 - \beta_2)(\nabla\theta_t)^2$$

Hvor,
m_t og v_t er estimaterne af første og andet moment for gradienten
β_1 og β_2 er eksponentielle glatningsfaktorer
$\nabla\theta_t$ er gradienten af loss-funktionen mht. parametrene

Læringsraten for parameteropdateringen θ justeres som,

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{v_t} + \epsilon} m_t$$

Hvor,
α er den første læringsrate, og ϵ er en lille værdi (så ingen division med 0).

Samlet set danner disse metoder fundamentet for projektet, hvor signalbehandling bruges til at optimere EEG-input, neurale netværk klassificerer bevægelsesmønstre, og transformer-

modellen muliggør multimodal analyse. Kombinationen af disse teknikker sikrer en robust og præcis detektion af intentionelle bevægelser hos personer med motoriske udfordringe