**Abschnitt: Aufbau und Funktionsweise des 1D-CNN für Klatsch-Erkennung**

1. **Vorverarbeitung**  
   Vor dem eigentlichen Netz-Durchlauf wird jedes Audiosignal auf eine einheitliche Länge gebracht. Enthält eine Aufnahme weniger als 4410 Samples (entspricht etwa 100 ms bei 44,1 kHz), wird sie **aufgefüllt** (Padding), ist sie länger, wird sie **gekürzt** (Trimming). Auf diese Weise liegt das Netz jedesmal ein Datenpaket der Größe (1, 4410) als Eingabe vor. Außerdem findet eine **Normalisierung** statt, sodass das Signal im Wertebereich [−1, 1] liegt.
2. **Convolutionale Layers (Feature-Extraktion)**  
   Im Anschluss folgt ein **1D-CNN**, bestehend aus zwei **Faltungs-Blöcken**:
   * **Block 1**:
     + Eine **1D-Convolution** mit in\_channels=1, out\_channels=16, kernel\_size=9, stride=1, padding=4.
     + Eine **ReLU-Aktivierung**, die negative Werte auf 0 setzt und dem Modell Nichtlinearität verleiht.
     + Ein **MaxPooling** über 4 Samples, um die Zeitachse herunterzuskalieren (z. B. von 4410 auf 1102 Samples).
   * **Block 2**:
     + Eine weitere **1D-Convolution**, diesmal in\_channels=16, out\_channels=32.
     + Eine erneute **ReLU** und ein zweites **MaxPooling**, was zu einer weiteren Verkleinerung der Zeitachse auf 275 Samples führt.

Jeder **out\_channel** definiert einen eigenen „Kanal“ bzw. Filter. So erzeugt Block 2 insgesamt **32 Kanäle**, die unterschiedliche akustische Merkmale (z. B. Peaks, Frequenzanteile(nur spekulation, was die KI hier extrahiert ist unbekannt)) extrahieren.

1. **Kanäle und Muster**  
   Die Idee hinter den **32 Kanälen** ist, dass jede Faltungsschicht verschiedene **Filter** lernt. Einige können sich auf hohe Frequenzbereiche oder impulsartige Einschwingvorgänge spezialisieren, andere auf tiefe Frequenzen oder bestimmte Signalformen. Auf diese Weise entstehen unterschiedliche **Feature Maps**, welche die relevanten Charakteristika eines Klatschens (versus Nicht-Klatsch) erfassen.
2. **Fully-Connected Layers (Klassifizierung)**  
   Nach dem zweiten Faltungsblock wird der Ausgabetensor (Batch, 32, 275) **geflacht** zu einem Vektor der Länge 32 × 275 = 8800. Dieser fließt in zwei **Linear-Schichten**:

fc1: nn.Linear(8800, 128), gefolgt von einer ReLU. => wir hatten nach der convolution layer mehr datenpunkte als wenn wir es roh verfüttert hätten.

fc2: nn.Linear(128, 2), um zwischen den **zwei Klassen** „Clap“ und „NoClap“ zu unterscheiden.

Im Training bewertet eine **Loss-Funktion** (Cross-Entropy) die Vorhersage (z. B. [LogitClap, LogitNoClap]) gegen das reale Label. Daraus lernt das Netzwerk, die Filtergewichte so anzupassen, dass Klatschenbestimmung zuverlässig gelingt.

**Abschnitt: Test nr.2 mit nur einer simplen convolution layer und weniger übergebenen datenpunkte**

**Zusätzliche Dokumentation: Unterschiede zum vorherigen Mehr-Convolution-Modell**

Im vorherigen Modell wurden **mehr Convolution-Layer** (z. B. zwei Faltungsblöcke mit je Conv+ReLU+Pooling) verwendet und im Anschluss **wenige Fully-Connected-Schichten** (typischerweise ein oder zwei). Das Rohsignal wurde dabei mehrfach gefaltet und stark heruntergesampelt, sodass am Ende ein großer Vektor (z. B. ~8800 Merkmale) in eine relativ kompakte FC-Struktur floss (etwa 128 → 2).

Im **neuen Modell** hingegen wird:

1. **Nur eine** Convolution-Schicht (mit 32 Kanälen) genutzt, worauf ein **Global Average Pooling** folgt.
2. Die danach verbleibenden 32 Merkmalswerte pro Sample werden nun durch **mehrere, deutlich größere FC-Schichten** verarbeitet:
   * **fc1**: 32 → 512
   * **fc2**: 512 → 512
   * **fc3**: 512 → 2

**Hauptunterschiede**

1. **Verlagerung der Modellkapazität**
   * Zuvor war der Fokus auf den **Convolution-Layern**: Mehrere Faltungen lernten Zeit-/Frequenzmuster, und ein kleiner FC-Teil klassifizierte die extrahierten Merkmale.
   * Jetzt findet nur **eine** (recht einfache) Faltung statt, während der größte Teil der „Lernfähigkeit“ in den **drei großen** (groß ist hier relativ weil bei dem ersten durchgang wir eine deutlich größere eingagnslayer hatten) **FC-Layern** liegt.
2. **Merkmalsextraktion vs. abstrakte Repräsentation**
   * Bei mehreren Conv-Layern werden Merkmale schon stark hierarchisch extrahiert (von groben bis zu feineren Mustern). Dadurch landet das Signal vergleichsweise klar separiert im FC-Teil.
   * In der neuen Variante („eine Conv + große FC-Blöcke“) erzeugt die einzelne Faltung nur eine grobe Vorverarbeitung. Die eigentliche **Komplexität** wird dann in den FC-Layern gelernt, die aus den 32 Kanälen (Global Average) eine leistungsfähige Klassifikationsfunktion bilden.
3. **Datenfluss & Dimensionsreduktion**
   * Vorher: 2× Conv + Pooling → starke Reduktion auf (Batch, 32, ~275), dann Flatten (~8800 Features) → 2 FC-Schichten.
   * Jetzt: 1× Conv (32 Kanäle) → Global Average Pooling → (Batch, 32) → **3** aufeinanderfolgende FC-Layer.
4. **Performance & Genauigkeit**
   * In vielen Szenarien können tiefere Convs robuster gegen Rauschen und Variation sein.
   * Hier hat sich jedoch gezeigt, dass **weniger Convs** + **mehr große FC-Layer** teils ebenfalls **hohe Genauigkeit** erzielen – vorausgesetzt, der Datensatz und die Rechenressourcen reichen aus. Eine Accuracy von 100 % wurde erreicht.
5. **Rechenaufwand**
   * Mehr Convolution-Layer erhöht oft die Rechenlast auf dem Audio-Teil, kann aber effizient auf GPUs laufen (dieser Vorteil kann auf dem Raspi nicht ausgenutzt werden weil keine zusätzliche GPU verwendet wird).
   * Wenige Convs + viele FC-Neuronen bedeuten hohe Parameterzahlen in den FC-Gewichten. Das kann für eine CPU-Umgebung mehr Rechen- oder Speicherbedarf erzeugen, muss aber nicht generell langsamer sein. Es hängt stark vom Framework und der Hardware ab.

**Fazit**:  
Während das alte System stark auf **mehrere Convolution-Stufen** setzte, verschiebt das neue System die Komplexität in **dicke FC-Schichten**. Die **eine** Faltungsschicht plus **Global Average Pooling** liefert lediglich 32 grobe Merkmale, die dann in **mehreren großen Dense-Layern** verarbeitet werden. Dieses andere „Architektur-Layout“ kann ebenfalls sehr leistungsfähig sein, hat aber eine andere **Balance** zwischen Feature-Extraktion im Faltungsbereich und Klassifikation im Fully-Connected-Bereich.

**Dokumentation: Einfacheres 1D-CNN mit FFT-Vorverarbeitung und weniger Convolution**

1. **Überblick**  
   Im aktuellen Netz wird das Audiosignal zunächst per **Fast Fourier Transformation (FFT)** in ein Frequenzspektrum umgewandelt. Anschließend kommen wie bei dem ersten versuch 2 Convolution-Schichten zum Einsatz, bevor die Daten in 2 Fully-Connected-Layer fließen. Das Training läuft hier **schneller** als bei tiefen Convolution-Netzen, weil weniger Faltungs- und Pooling-Operationen ausgeführt werden müssen.
2. **Erste Ergebnisse**
   * Das Netz erreicht in wenigen Epochen schon akzeptable Vorhersagewerte.
   * Allerdings zeigen die anfänglichen Ausgaben mehr **False Positives** (falsch als „Klatsch“ eingestufte Geräusche) als bei den komplexeren Modellen.
   * **False Negatives** (echte Klatscher, die fälschlich nicht erkannt werden) treten dagegen seltener auf. Durch kontinuierliches Training (mehr Epochen) können die FP-Fehler weiter reduziert werden.
3. **Vorteile**
   * **Trainingsgeschwindigkeit**: Durch die verringerte Konvolutions-Architektur und die direkte Frequenzdarstellung via FFT ist der Datendurchsatz höher, und das Netz konvergiert schnell.
   * **Geringerer Rechenaufwand**: Weniger Faltungen bedeuten weniger Parameter im Convolution-Teil, was vor allem auf Geräten wie dem Raspberry Pi 5 spürbar ist.
4. **Nachteile**
   * **Erhöhte Fehlalarme (False Positives)**: Das Modell neigt anfangs dazu, mehr Alltagsgeräusche als „Klatschen“ zu klassifizieren und braucht zusätzliche Epochen (oder andere Maßnahmen), um diese Fehler zu reduzieren.
   * Um es anzuwenden muss das gerät die erfassten daten immer zuerst umwandeln bevor es ausgewertet wenden kann
5. **Fazit**
   * Das Netz ist **rascher trainiert** und liefert zügig eine brauchbare Erkennungsleistung.
   * Wer **größtmögliche Genauigkeit** und weniger Fehlalarme benötigt, wird womöglich doch auf eine **tiefere Convolution-Struktur** (oder mehr Epochentraining + hyperparameter tuning) setzen.

Btw ich habe hier mit GPT alles wichtige ausschreiben lassen und es selber nochmal überprüft das alles soweit stimmen sollte. Alle drei tests sind fertig und bereit dokumentiert zu werden. Warscheinlich müssen wir noch schauen was wir wie Visualisieren aber soweit sollte es genug für die Doku sein. Mein größter takeaway ist das 120 Datensätze absolut nicht genug sind. Daduch haben wir das problem das eine abweichung von einem false positive oder negative im test sofort dazu führt das wir nurnoch eine präzision von 96% haben was giga müll ist.