**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 ungefähr 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 etwa 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) 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.

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.