Technische Universität Berlin

# Grundlagen der automatischen Spracherkennung

## *Aufgabe 5 – PyTorch-Einführung*

13.12.2023

Wentao Yu

Technische Universität Berlin

# VoxCeleb-Gender-Korpus

| Female | Male |
|---|---|
| 2311 | 3682 |

- Ziel: das Geschlecht des Sprechers durch die entsprechende Sprachaufnahme zu identifizieren.
- Herunterladen: https://tubcloud.tu-berlin.de/s/Banx6DmqNMArGbq

Technische Universität Berlin

# Vorbereitung

Laden die Datei Aufgabe5.zip aus dem ISIS-Kurs herunter.
Entpacken Sie den Inhalt des Archivs in folgender Struktur:

./my-repository/torch_intro/dataset/train.json
./my-repository/torch_intro/dataset/dev.json
./my-repository/torch_intro/dataset/test.json
./my-repository/torch_intro/local/train.py
./my-repository/torch_intro/local/utils.py
./my-repository/torch_intro/local/model.py
./my-repository/uebung5.py

Technische Universität Berlin

# Vorbereitung

Der VoxCeleb-Gender-Korpus soll **NICHT** im ./my-repository Ordner gespeichert werden.

Konfigurieren Sie den Pfad zum VoxCeleb-Gender-Korpus in der Funktion get args() in der Datei ./my-repository/uebung5.py mit der Variable --sourcedatadir.

Kopieren Sie die Dateien feature_extraction.py und tools.py aus Ihrem Repository in die Verzeichnisse ./my-repository/torch_intro/local/.

Technische Universität Berlin

# Aufgaben

1. „Dataloader“ konstruieren.

2. Einfache neuronale Netze entwerfen.

3. Klassifikator trainieren und evaluieren

Technische Universität Berlin

# Datenlader

In ./my-repository/ torch_intro/ local/utils.py:

- get_data() Funktion: Metainformationen in Python als Dictionary einlesen
- __getitem__(self, index) Funktion in Dataloader() Klasse: lädt die Metainformationen einer Sprachaufnahme und extrahiert die Merkmale. Alle Ausgangsdateien sollen in Tensoren umgewandelt werden.
- padding() Funktion: um Zero-Padding anzuwenden und die akustischen Merkmale eines Batches auf die gleiche Länge (maximale Batchgröße) zu bringen.

Technische Universität Berlin

# Einfaches Feedforward-Netzwerk mit PyTorch

In ./my-repository/ torch_intro/ local/model.py:

```
class Classification(torch.nn.Module):
        def __init__(self, idim, odim, hidden_dim):
            ...
        def forward(self, audio_feat):
            ...
```

Ergänzen Sie Classification() Klasse:

1. Implementieren Sie bitte ein Feedforward-Netzwerk mit drei vollständig verbundenen Schichten, auf jede vollständig verbundene Schicht sollte eine ReLU-Aktivierungsfunktion folgen.

2. Implementieren Sie zusätzlich eine vollständig verbundene Klassifizierungsschicht, gefolgt von Sigmoid-Aktivierungsfunktion.

Technische Universität Berlin

# Training und Evaluierung eines Klassifikators

In ./my-repository/ torch_intro/local/train.py:

```
def train(dataset, model, optimizer=None, criterion=None):

def evaluation(dataset, model):
    ...
```

Ergänzen Sie die train() und evaluation() Funktionen, um den Klassifikator zu trainieren und evaluieren.

train():

- Modell -> Trainingsmodus
- Trainingsdatensatz
- Posterior-Wahrscheinlichkeiten berechnen
- Loss-Wert berechnen
- Gradienten berechnen und Modellparameter aktualisieren
- Genauigkeit berechnen

evaluation():

- Modell -> Evaluierungsmodus
- Evaluierungsdatensatz
- Posterior-Wahrscheinlichkeiten berechnen
- Ergebnisse speichern
- Genauigkeit berechnen

Technische Universität Berlin

# Melden Sie sich gerne bei Fragen