deepLID

Sprachidentifikation mit Deep Learning

Joel André

Maturaarbeit
Betreuungsperson:
Beni Keller

Kantonsschule Zug
2019

Inhaltsverzeichnis

1 Einleitung

2 Deep learning

Die Begriffe *Deep Learning*, *maschinelles Lernen* und, *künstliche Intelligenz*, werden oft fälschlicherweise auswechselbar verwendet, und in eine Schublade geräumt. Es gibt allerdings eine ganz klare, und für das Verständnis wichtige Hierarchie zwischen den Wörtern. Um Klarheit zu verschaffen werden darum alle Gebiete aufgeführt.

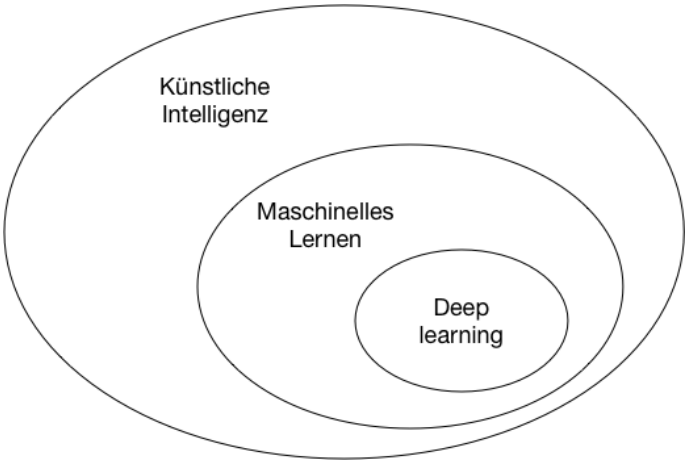


Abbildung 1: Künstliche Intelligenz, Maschinelles Lernen und Deep Learning

Das Gebiet der künstlichen Intelligenz gibt es schon so lange wie den Computer selbst. Die Frage wie schlau ein Computer werden kann, beschäftigt uns bis heute. Al anerkannte Definition für KI gilt, das Bestreben intellektuelle Aufgaben, die normalerweise von Menschen gelöst werden zu automatisieren.

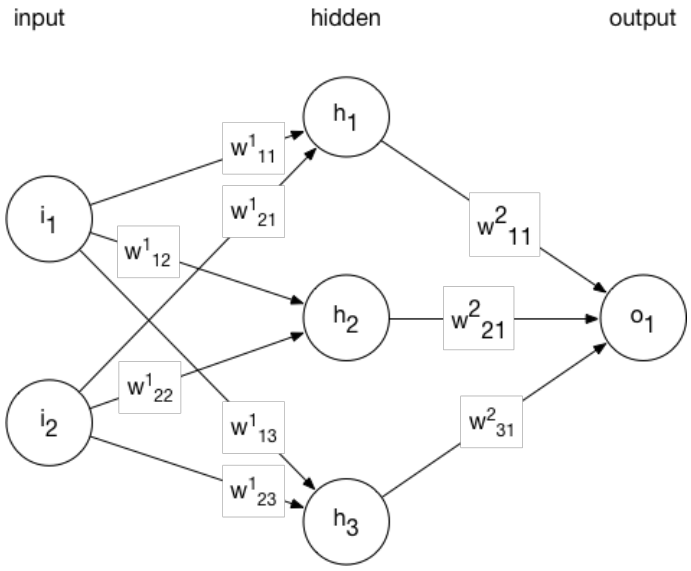
Erste Erfolge erreichte man zum Beispiel mit Schachcomputern, die handgeschriebene Regeln befolgten. Diese Form von künstlicher Intelligenz hatte aber schnell Grenzen, da viele Prozesse schlicht zu komplex waren um sie unter angemessenem Aufwand mit Regeln zu beschreiben. An diesem Punkt wurde man auf maschinelles Lernen aufmerksam.

Der Ablauf von maschinellem Lernen ist grundlegend anders als konventionelles Programmieren. Der Entwickler muss keinen Programmcode mit festen Regeln schreiben, im Gegenteil. Man liefert dem Computer Eingabe und Ausgabe, und der Computer lernt selber die Regeln. Maschinelles Lernen entstand erst so richtig um 1990, wurde aber schnell zum grössten Teilgebiet der künstlichen Intelligenz.

Beim maschinellem Lernen, lernt die Software im Grunde eine nützlichere Darstellungsweise der Daten bzw. der Eingabe. Anhand dieser anderen Darstellungsweise kann der Computer die Antwort einfach erkennen. Wenn der Computer stufenweise nützlichere Repräsentationen bestimmt kann er zunehmend komplexe Probleme in einfachere Zwischenschritten lösen. Genau das ist deep learning. Es beschreibt also mehr das Konzept von stufenweisem Lernen als eine Methode selber. Ein weit verbreitete Mehode sind allerdings *tiefe künstliche neuronalen Netzen*[vgl. 1].

2.1 Künstliche Neuronale Netze

Künstliche neuronale Netze hat man sich, wie der Name schon preisgibt, von der Natur abgesehen. Ähnlich wie in unserem Gehirn gibt es Neuronen (Knoten, z.b o_1) und dazwischenliegende Verbindungen. Die Verbindungen haben ein Gewicht



$$o_1 = \sigma\Big(\sum_i h_i \cdot w^2_{i1}\Big)$$

Abbildung 2: Grafische Darstellung eines künstlichen neuronalen Netzwerks

w. Der Wert eines Knotens ist eine Funktion der Summe all er seiner eingehenden Verbindungen. Diese Funktion wird Aktivierungsfunktion σ genannt. Eine bekannte Aktivierungsfunktion ist zum Beispiel die Sigmoid Funktion: $exp(x) = \frac{1}{1+e^{-x}}$[6]. Die Aktivierungsfunktion ist wichtig damit das Netwerk auch nicht lineare Repräsentationen lernen kann. Es ist bewiesen dass solche neuronale Netze jede Funktion abbilden können[4, Kap. 4].

Die mathematischen Operationen in einem neuronalen Netwerk lassen sich alle als Matrizen-Operationen berechnen. Mit Matrizen kann der Computer auf einer Grafikkarte und mit einem BLAS (*Basic linear algebra system*) sehr effizient rechnen.

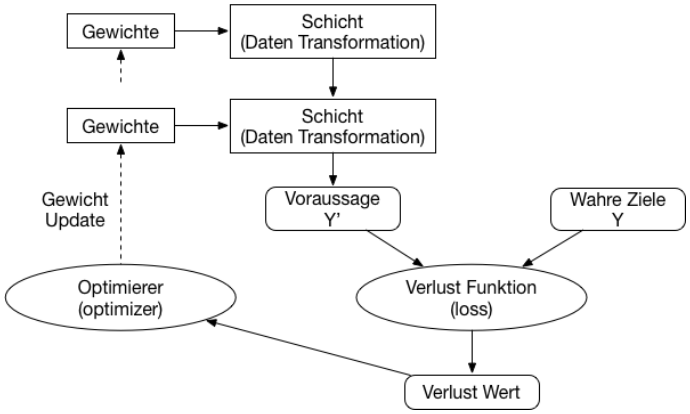


Abbildung 3: Grafische Darstellung eines künstlichen neuronalen Netzwerks

Die Gewichte sind die Parameter. Um diese zu erlernen braucht es eine *Verlust Funktion* die uns angiebt, wie weit die Ausgabe von den korrekten Zielen entfernt ist. Anhand des Verlustwerts passt das Netzwerk die Gewichte schrittweise an. Die-

sen Teil übernimmt der *Optimierer*. Ein einfacher Optimierer ist zum Beispiel das Gradientenverfahren. Da bei diesem Typ von neuronalen Netzwerken alle Knoten miteinander verbunden sind, wird es oft *dense neural network* genannt.

2.2 Convolutional Neural Networks

Convolutional Neural Networks sind eine sehr weit verbreitete Methode im Feld von *Computer Vision*. Der fundamentale Unterschied zwischen dem oben besprochenen *dense network* und einem *CNN* ist, dass ein *CNN* lokale Muster erkennen kann, wo hingegen das vorherige Netzwerk nur globale Muster erkennen konnte. Das heisst, dass ein Muster an einer bestimmten Stelle angetroffen wird, an jeder anderen Stelle ebenfalls erkannt wird. [1]

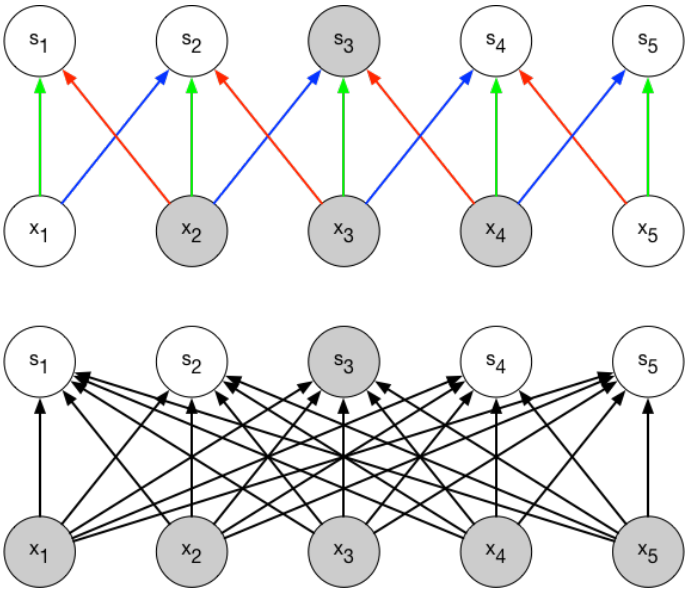


Abbildung 4: (*Oben*)1D Convolution mit *kernel* der Grösse 3. s_3 wird durch 3 inputs beeinflusst. (*Unten*) *Dense Network*. s_3 wird durch alle inputs beeinflusst.[2]

Ein weiterer Vorteil von *Convolutional neural networks* ist, dass sie eine räumliche hyrarchie von Mustern erlernen können. Wenn die Eingabe das Bild einer Katze ist, wird zum Beispiel die erste Schicht unterschiedliche Kanten erkennen, die zweite Schicht dann einzelne Merkmale (z.b Augen), und so weiter.

Damit das gilt, muss aber der analysierte Bereich eines Knotens, von Schicht zu Schicht grösser werden. Deshalb wird meisten nach jedem *convolution layer* ein *pooling layer* gesetzt. Das *pooling layer* fasst mehrere Datenpunkte zusammen um dem nächsten Netzwerk eine grösseren Analysebereich zu verschaffen.

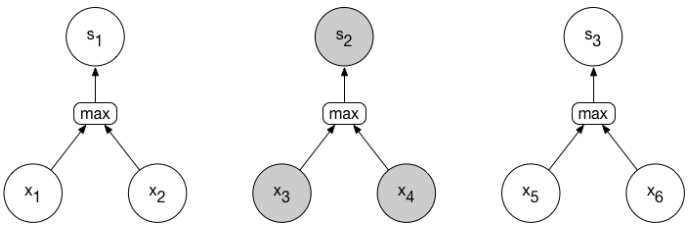


Abbildung 5: Abbildung eines 1D Max-Pooling layer. s_2 ist $\max(x3, x4)$

2.3 [andere Tricks]

3 Daten

3.1 Daten Auswahl

Es gibt keine frei zugänglichen umfassenden Datasets für Sprachidentifikations-Aufgaben. Datensets wie das *NIST Language Recognition Evaluation*[5] sind nur unter teuren Gebühren zugänglich. Wie verwandte Arbeiten empfehlen [8], wird darum ein eigenes Datenset zusammengestellt. Es werden gleichmässig Daten zu den Sprachen Deutsch, Englisch, und Französisch gesammelt. Die Daten stammen einerseits vom *Voxforge*[9] Datenset und von *Youtube*[10].

Voxforge ist ein open-source Datenset für die Spracherkennung. Es besteh aus vielen kurzen (1-10s), von Benutzern hochgeladenen, Audiodateien. Die englische Sprache dominiert mit rund 120h Audio das Datenset. Insgesamt kommen 190h Aufnahme zusammen. Die Audioqualität variiert je nach Benutzer.

Die Sprache ist langsam und deutlich verständlich. Sie hört sich eher künstlich an, im Vergleich zu einem natürlichem Gespräch.

Youtube dient als Quelle für natürlichere Sprache. Es werden Nachrichtenkanäle wie CNN, ARD, etc. verwendet. Die Sendungen haben den Vorteil, dass oft fremde Gäste eingeladen werden, was zu einer grossen Variation der Sprecher führt. Ausserdem kommen oft natürliche Hintergrundgeräusche dazu. Zum Schluss ist die Datenmenge die hier erhältlich ist praktisch unbeschränkt gross.

Sprache	Voxforge	Youtube
Französisch	20h	50h
Deutsch	22h	60h
Englisch	23h	200h

(a) Verteilung der Datenmenge

Sprache	Kanäle
Französisch	France24, FranceInfo
Deutsch	ARD, ZDF
Englisch	CNN, BBC

(b) Youtube Kanäle

Tabelle 1: Daten Auswahl

3.2 Daten Split

Zu grosse Datenmengen sind für das Training problematisch. Sie sind technisch aufwendig und sehr langsam in der Verarbeitung. Zudem ist es immer das Ziel eines Algorithmus, mit möglichst wenigen Daten auszukommen. Darum wird das Datenset auf 100'000 Stücke reduziert, was ungefähr 139h insgesamt ausmacht. Die Daten stammen zufällig zu gleichen Teilen sowohl aus Youtube wie von Voxforge. Die einzelnen Sprachen sind ebenfalls genau gleichrangig repräsentiert.

Das Datenset wird wiederum in *Trainingset*, *Validationset*, und *Testset* aufgeteilt zu den Anteilen 80%, 10% und 10%. Das *Trainingset* wird, wie der Name preisgibt, für das Training verwendet. Das *Validationset* wird verwendet um zu beobachten, wie das Modell auf neue Daten reagiert. Die *Hyperparamter* (Parameter die das Netzwerk nicht selber lernen kann, z.B die Anzahl Knoten) werden manuell so angepasst dass das Netzwerk möglichst gut auf dem *Validationset* abschneidet. Zuletzt wird das *Testset* in Betracht gezogen, um die Leistung auf gänzlich neue Daten zu erforschen.

3.3 Preprocessing

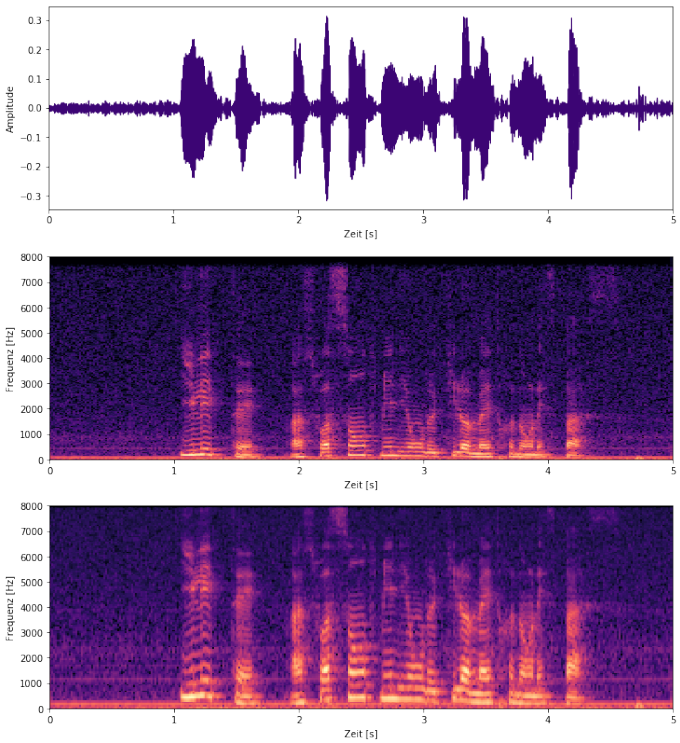


Abbildung 6: Audio-Preprocessing: *(oben)* Rohe Schallwelle, *(mitte)* Log-power Spectrogramm, *(unten)* Mel Spectrogramm

4 Web Interface

Es war seit Anfang an klar, dass ein Web-Interface gestaltet werden sollte. Das Web-Interface ist ideal um die Leistung der Software hautnah zu erleben. Man kann mit jedem internetfähigen Gerät (mit Mikrofon) darauf zugreifen.

Die Idee ist einfach. Man nimmt direkt im Browser eine kurze Audioaufnahme auf, lädt diese hoch zum Server, und dieser antwortet mit der geschätzten Sprache. Bevor man die Aufnahme hochlädt kann man sie optional selbst abspielen.
Aktuell läuft ein Web-Server unter: `jo.guru.ksz.ch/deeplid`

4.1 Frontend

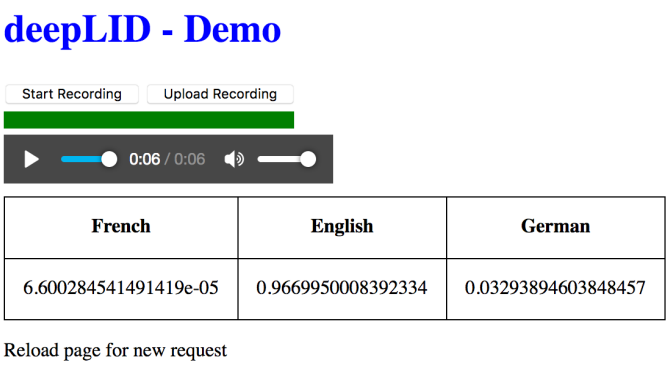


Abbildung 7: Web-Interface Screenshot

Das Design ist in einfachem *HTML* und *CSS* geschrieben, die Logik mit *Javascript*. Um im Browser Audioaufnahmen nehmen zu können, wird *RecordRTC*[3] verwendet. *RecordRTC* ist eine unter der *MIT* Lizenz veröffentlichte *Javascript* Bibliothek für Medienaufnahmen. Sie ist kompatibel mit vielen modernen Browsern und Geräten.

Nach der Aufnahme wird die Datei über AJAX POST an den Server verschickt. Der Server antwortet mit einem XML-Snippet der Resultate. Dieses wird wiederum direkt in die Webseite eingebaut.

4.2 Backend

Der Webserver läuft in der Programmiersprache *Python*. Als Framework wird mit *Flask*[7] verwendet. Da sowohl der *Flask* wie auch *Keras* mit Python laufen, kann der Webserver das trainierte Modell ohne zusätzliche Hürden verwenden. Der Code findet sich unter 7.1

5 Resultate

6 Diskussion

6.1 Ausbaumöglichkeiten

7 Code

7.1 Webserver

```
@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        request_file = request.files['file']
        if request_file:
            # Virtuelle Datei generieren
            tmp_file = io.BytesIO(request_file.stream.read())
            # Abfrage an Modell
            pred = predict(tmp_file)
            # HTML als Antowrt senden
            return render_template('prediction.html',
                                   french_prob=str(pred[0]),
                                   english_prob=str(pred[1]),
                                   german_prob=str(pred[2]))
    else:
        # Web-Interface Standard Seite verschicken
        return render_template('site.html')
```

7.2 Beispiel-Modell

```
# Importieren von Bibliotheken
from keras import models, layers
from kapre.time_frequency import Melspectrogram

# Definieren des Modells
architecture = [
    Melspectrogram(n_dft=512, input_shape=(1, 5 * 16000,),
                   padding='same', sr=16000, n_mels=28,
                   fmin=0.0, fmax=10000, power_melgram=1.0,
                   return_decibel_melgram=False, trainable_fb=
                       False,
                   trainable_kernel=False),
    layers.Conv2D(64, (3, 6), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 6), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 6), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(1024, activation='relu'),
    layers.Dense(3, activation='softmax')
]

model = models.Sequential(architecture)

# Zusammenbauen des Modells
model.compile(optimizer='Rmsprop',
              metrics=['accuracy'],
```

```
                loss='categorical_crossentropy')

# Trainieren des Modells
model.fit(train_data, train_labels,
          batch_size=64,
          epochs=9,
          validation_data=(val_data, val_labels))

# OUTPUT
# Train on 7500 samples, validate on 3750 samples
# Epoch 1/9
# 7500/7500 [==============================] - 17s 2ms/step -
    loss: 1.0368 - acc: 0.4888 - val_loss: 1.0159 - val_acc:
    0.5152
# Epoch 2/9
# 7500/7500 [==============================] - 13s 2ms/step -
    loss: 0.8200 - acc: 0.6319 - val_loss: 1.1726 - val_acc:
    0.4509
# ...
```

Literatur

[1] François Chollet. *Deep learning with Python*. Shelter Island, NY: Manning Publications Co., 2018.

[2] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[3] Muaz Khan. *WebRTC*. URL: `https://github.com/muaz-khan/RecordRTC` (besucht am 23. 07. 2018).

[4] Michal Nilson. *Neural Networks and Deep Learning*. Dez. 2017. URL: `http://neuralnetworksanddeeplearning.com/index.html` (besucht am 21. 07. 2018).

[5] *NIST 2017 Language Recognition Evaluation*. URL: `https://www.nist.gov/itl/iad/mig/nist-2017-language-recognition-evaluation` (besucht am 24. 07. 2018).

[6] Tariq Rashid. *Neuronale Netze selbst programmieren*. Heidelberg: O'Reilly, 2017.

[7] Armin Ronacher. *Flask*. URL: `http://flask.pocoo.org/` (besucht am 23. 07. 2018).

[8] Thomas Werkmeister Tom Herold. "Practical Applications of Multimedia Retrieval. Language Identification in Audio Files". In: (2016). URL: `https://github.com/twerkmeister/iLID/blob/master/Deep%5C%20Audio%5C%20Paper%5C%20Thomas%5C%20Werkmeister%5C%2C%5C%20Tom%5C%20Herold.pdf`.

[9] *Voxforge*. URL: `http://www.voxforge.org/` (besucht am 24. 07. 2018).

[10] *Youtube*. URL: `https://www.youtube.com/` (besucht am 24. 07. 2018).

Abbildungsverzeichnis