Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Ερευνητική εργασία για το μάθημα της Ψηφιακής Επεξεργασίας Σήματος

Speech Emotion Recognition - Classification [1]

Χριστίνα Σαρτζετάκη - 03116193

Σεπτέμβριος 2019

[1] https://github.com/SergeantChris/tfSER

Περίληψη

Η αναγνώριση συναισθήματος είναι ένα υπερσύγχρονο αντικείμενο έρευνας στο χώρο της αλληλεπίδρασης ανθρώπου υπολογιστή (HCI) με πολλές ενδιαφέρουσες εφαρμογές. Η εργασία αυτή επικεντρώνεται στο κομμάτι της ταξινόμησης δειγμάτων φωνής από τη Ryerson Audio-Visual Database of Emotional Speech and Song σε ένα από τα οκτώ συναισθήματα (ουδέτερο, ηρεμία, χαρά, λύπη, θυμός, φόβος, έκπληξη, αηδία), με χρήση βαθειών Νευρωνικών Δικτύων (DNNs). Μετά από εφαρμογή τεχνικών βελτιστοποίησης του μοντέλου και δοκιμή διαφορετικών συνόλων χαρακτηριστικών επιτεύχθηκε μέγιστο ποσοστό αναγνώρισης 54%.

1 Εισαγωγή

Το συναίσθημα είναι ένα πολύ σημαντικό μέρος της πληροφορίας που μεταδίδεται κατά την επικοινωνία των ανθρώπων. Οι καθημερινές συναναστροφές μας διέπονται από την έκφραση συναισθημάτων στην ομιλία μας, και βασίζονται στην αναγνώριση αυτών των συναισθημάτων από την άλλη πλευρά. Ιδιαίτερη χρισιμότητα έχει η αναγνώριση συναισθήματος όταν ο ομιλιτής βρίσκεται σε κάποια δυσάρεστη ψυχολογικά κατάσταση, όπως στεναχώρια, άγχος ή φόβος, όπου η όποια δράση του ακροατή θα καθορίσει την επίλυση ή όχι του προβλήματος. Η αναγνώριση όμως δεν γίνεται πάντα με επιτυχία καθώς ο χαρακτήρας και η ανατροφή κάποιου μπορεί να παίζει ρόλο στην ενσυναίσθηση που διαθέτει, με αποτέλεσμα να μην μπορέσει να κατανοήσει το συναίσθημα του άλλου, και να λάβει τη λάθος απόφαση αντιμετώπισης. Αν όμως αυτή η αναγνώριση γινόταν από μια τέλεια εκπαιδευμένη μηχανή, θα υπήρχε μικρότερο περιθώριο λάθους, σημαντικό σε συστήματα επικοινωνίας μεταξύ αγνώστων (call centers, customer service) και σε συστήματα προσωπικού βοηθού στο σπίτι ή στο αυτοκίνητο, όπου χρειάζεται να διαχειριστούν με επιτυχία άτομα με κατάθλιψη, διαταραχές άγχους ή άτομα που βρίσκονται σε κίνδυνο.

Από αυτά, και λόγω της γενικότερης προσπάθειας για πιο φυσική επικοινωνία μεταξύ ανθρώπου και μηχανής, το ενδιαφέρον για speech emotion recognition φαίνεται μεγάλο και μάλιστα η έρευνα έχει προοδεύσει πολύ σε αυτό το χώρο τα τελευταία χρόνια. Έχουν φτιαχτεί πολλές βάσεις δεδομένων φωνής για αυτό το σκοπό, όπως η Berlin Database of Emotional Speech [2] και η Ryerson Audio-Visual Database of Emotional Speech and Song [1], έχουν μελετηθεί εκτενώς διάφοροι συνδυασμοί χαρακτηρηστικών όπως energy και pitch frequency [3], formant frequency [4], Linear Prediction Coefficients (LPC), Linear Prediction Cepstrum Coefficients (LPCC), Mel-Frequency Cepstrum Coefficients (MFCC) [5], ενώ έχουν δοκιμαστεί πολλές μέθοδοι ταξινόμησης όπως Neural Networks (NN) [6], Gaussian Mixture Model (GMM), Hidden Markov model (HMM) [7], Maximum Likelihood Bayesian classifier (MLC), Kernel Regression και K-nearest Neighbors (KNN) και Support vector machines (SVM) [8, 21].

Σε αυτήν την εργασία χρησιμοποιήθηκε η Ryerson Audio-Visual Database of Emotional Speech and Song, ένας συνδυασμός από prosodic και spectral features (energy, ZCR & MFCC) και ως μέθοδος ταξινόμησης τα βαθειά Νευρωνικά

Δίκτυα (DNNs) με χρήση του εργαλείου TensorFlow, στα οποία μελετήθηκαν και συγκρίθηκαν πολλά διαφορετικά μοντέλα.

2 Μεθοδολογία

2.1 Βάση Δεδομένων Φωνής

Η Βάση Δεδομένων Φωνής που χρησιμοποιήθηκε σε αυτήν την εργασία είναι η open-source βάση RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) [1], συγκεκριμένα το μέρος της που περιέχει ομιλία σε αρχεία ήχου, με 1440 αρχεία = 24 ηθοποιοί (12 άντρες και 12 γυναίκες) από 60 δοκιμές ο καθένας στην ανάγνωση 2 προτάσεων στα αγγλικά σε Αμερικάνικη προφορά. Περιέχονται 8 συναισθήματα: ουδέτερο, ηρεμία, χαρά, λύπη, θυμός, φόβος, έκπληξη και αηδία, το καθένα από αυτά (εκτός του ουδέτερου) σε 2 εντάσεις, ουδέτερη και συναισθηματικά έντονη. Η κάθε πρόταση, στο κάθε συναίσθημα, στην κάθε ένταση, από τον κάθε ηθοποιό, επαναλαμβάνεται δύο φορές, με την κάθε επανάληψη να διαρκεί 3 δευτερόλεπτα (2 δευτερόλεπτα ήχου και 1 ησυχίας στο τέλος). Στην ονομασία των αρχείων ακολουθείται η σύμβαση 1-2-3-4-5-6-7.wav :
1. Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
2. Vocal channel (01 = speech, 02 = song).
3. Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
4. Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.
5. Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
6. Repetition (01 = 1st repetition, 02 = 2nd repetition).
7. Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).
(π.χ. 03-01-06-01-02-01-12.wav)
Επιπλέον, μετέπειτα προσθέθηκε ξανά ο αριθμός του ηθοποιού μπροστά στα ονόματα των αρχείων για διευκόλυνση στο διαχωρισμό train και test sets.
(π.χ. 12_03-01-06-01-02-01-12.wav)
Συγκεκριμένα, επιλέχθηκε το $1200/1440 \approx 83.3\%$ των αρχείων για training (10 άντρες και 10 γυναίκες) και τα υπόλοιπα $240/1440 \approx 16.7\%$ των αρχείων για testing (2 άντρες και 2 γυναίκες), για λόγους συμμετρίας στο φύλο και λόγους που αφορούν το μικρό μέγεθος της βάσης.

2.2 Εξαγωγή χαρακτηριστικών

Για την εξαγωγή χαρακτηριστικών χρησιμοποιήθηκε η open-source βιβλιοθήκη σε Python pyAudioAnalysis [9], συγκεκριμένα η συνάρτηση dirWavFeatureExtraction του αρχείου audioFeatureExtraction.py, που με παράμετρο το φάκελο με τα WAVE files, δημιουργεί μια λίστα από διανύσματα χαρακτηριστικών, ένα διάνυσμα για κάθε αρχείο. Τα διανύσματα αυτά αποτελούνται απο 34 μέσες τιμές και 34

αποκλίσεις χαρακτηριστικών για όλη τη διάρκεια του αρχείου ήχου (συνολικά 68 τιμές), με τα 34 αυτά χαρακτηριστικά να είναι:
1. Zero-Crossing Rate, 2. Energy, 3. Energy Entropy, 4-5. Spectral Centroid, Spectral Spread, 6. Spectral Entropy, 7. Spectral Flux, 8. Spectral Rolloff, 9-21: Mel Frequency Cepstrum Coefficients (MFCCs), 22-34: Chroma Features.
Η συνάρτηση αυτή επίσης δέχεται παραμέτρους το παράθυρο και το βήμα σε δευτερόλεπτα για mid-term και short-term υπολογισμούς. Για τους mid-term υπολογισμούς, φάνηκε βέλτιστη μια επιλογή που διατηρεί τον ελάχιστο αριθμό παραθύρων στη συνολική διάρκεια των αρχείων (3 s με 1 s ησυχίας στο τέλος), δηλαδή μεγάλο μέγεθος παραθύρου (2 - 2.5 s) και μικρό μέγεθος βήματος (0.25 - 0.5 s). Για τους short-term υπολογισμούς χρησιμοποιήθηκε μέγεθος παραθύρου 10 ms, με κριτήριο τη μέση διάρκεια φωνημάτων (80 ms) που καθορίζει την τάξη μεγέθους του μέγιστου διαστήματος κατά το οποίο το σήμα είναι στάσιμο [10], ενώ βέλτιστος φάνηκε να είναι ο λόγος βήματος προς παράθυρο 1:4 (αρκετά μικρός). Μετά την εξαγωγή των διανυσμάτων χαρακτηριστικών, και αφού εφαρμόστηκε min max normalization [11], αυτά μεταγράφηκαν σε 2 αρχεία Comma Seperated Values, train_data.csv και test_data.csv (αρχείο script.py), ενώ τυχαιοποιήθηκε η σειρά των διανυσμάτων για να εξαλειφθεί η πιθανότητα μάθησης πάνω σε ανύπαρκτα μοτίβα στα αρχεία train_dataSFL.csv και test_dataSFL.csv (αρχείο shuffle.py).

2.3 Αρχικό Νευρωνικό Δίκτυο

Για την κατασκευή και την εκμάθηση νευρωνικού δικτύου σε αυτό το n-way classification task, χρησιμοποιήθηκε το εργαλείο TensorFlow της Google, με χρήση υπολογιστικών πόρων που προσφέρει η browser-based πλατφόρμα ανάπτυξης Google Colaboratory. Την είσοδο στο αρχικό νευρωνικό δίκτυο που κατασκευάστηκε αποτελούν τα 4 αρχεία csv, 2 αρχεία δεδομένων (train_dataSFL.csv, test_dataSFL.csv) μεγέθους 1200x68 και 240x68 αντίστοιχα, και 2 αρχεία labels (train_labelsSFL.csv, test_labelsSFL.csv) μεγέθους 1200x1 και 240x1 αντίστοιχα. Τα αρχεία labels κατασκευάστηκαν από τα ονόματα των αρχείων, που περιέχουν τον αριθμό του συναισθήματος (αρχείο mklabels.py). Ως αφετηρία για την κατασκευή του νευρωνικού δικτύου πάρθηκε το πλήθος των layers και ο αριθμός/το είδος των κόμβων σε αυτά παρόμοια με το σχήμα που προτείνεται σε tutorial [16], οι επιλογές ειδικές για n-way classification (multi-class, single-label) ήταν ο συνδυασμός softmax ως τελευταίο στρώμα και sparse categorical crossentropy ως loss function [12, 15, 17], ενώ ως optimizer επιλέχθηκε ο adam [12, 18].

```
[10] import tensorflow as tf
     from tensorflow import keras
     import pandas as pd
     from google.colab import files
     import matplotlib.pyplot as plt
     import numpy as np
```

```
[4]  upl = files.upload()
     Train_Data = pd.read_csv('train_dataSFL.csv', header = None)
     Train_Labels = pd.read_csv('train_labelsSFL.csv', header = None)
     Test_Data = pd.read_csv('test_dataSFL.csv', header = None)
     Test_Labels = pd.read_csv('test_labelsSFL.csv', header = None)
```

Choose Files | 4 files

- **test_dataSFL.csv**(application/vnd.ms-excel) - 316751 bytes, last modified: 10/9/2019 - 100% done
- **test_labelsSFL.csv**(application/vnd.ms-excel) - 720 bytes, last modified: 10/9/2019 - 100% done
- **train_dataSFL.csv**(application/vnd.ms-excel) - 1584319 bytes, last modified: 10/9/2019 - 100% done
- **train_labelsSFL.csv**(application/vnd.ms-excel) - 3600 bytes, last modified: 10/9/2019 - 100% done

```
Saving test_dataSFL.csv to test_dataSFL.csv
Saving test_labelsSFL.csv to test_labelsSFL.csv
Saving train_dataSFL.csv to train_dataSFL.csv
Saving train_labelsSFL.csv to train_labelsSFL.csv
```

Figure 1: Φόρτωση αρχείων εισόδου

```
[ ]  def plot_history(histories, key='acc'):
       plt.figure(figsize=(16,10))

       for name, history in histories:
         val = plt.plot(history.epoch, history.history['val_'+key],
                        '--', label=name.title()+' Validate')
         plt.plot(history.epoch, history.history[key], color=val[0].get_color(),
                  label=name.title()+' Train')

       plt.xlabel('Epochs')
       plt.ylabel(key.title()+'uracy')
       plt.legend()

       plt.xlim([0,max(history.epoch)])
```

Figure 2: Συνάρτηση σχεδίασης γραφικής παράστασης απόδοσης [13]

```
[ ]  model_ini = keras.Sequential()

     model_ini.add(keras.layers.Dense(units = 64, activation = tf.nn.relu, input_dim = 68))
     model_ini.add(keras.layers.Dense(units = 32, activation = tf.nn.relu))
     model_ini.add(keras.layers.Dense(units = 16, activation = tf.nn.relu))
     model_ini.add(keras.layers.Dense(units = 8, activation = tf.nn.softmax))

     model_ini.compile(optimizer = 'adam',
                   loss = 'sparse_categorical_crossentropy',
                   metrics = ['accuracy'])

     ini_history = model_ini.fit(Train_Data, Train_Labels, epochs = 100, validation_data = (Test_Data, Test_Labels))

     plot_history([('Initial', ini_history)])
```

Figure 3: Αρχικό μοντέλο νευρωνικού

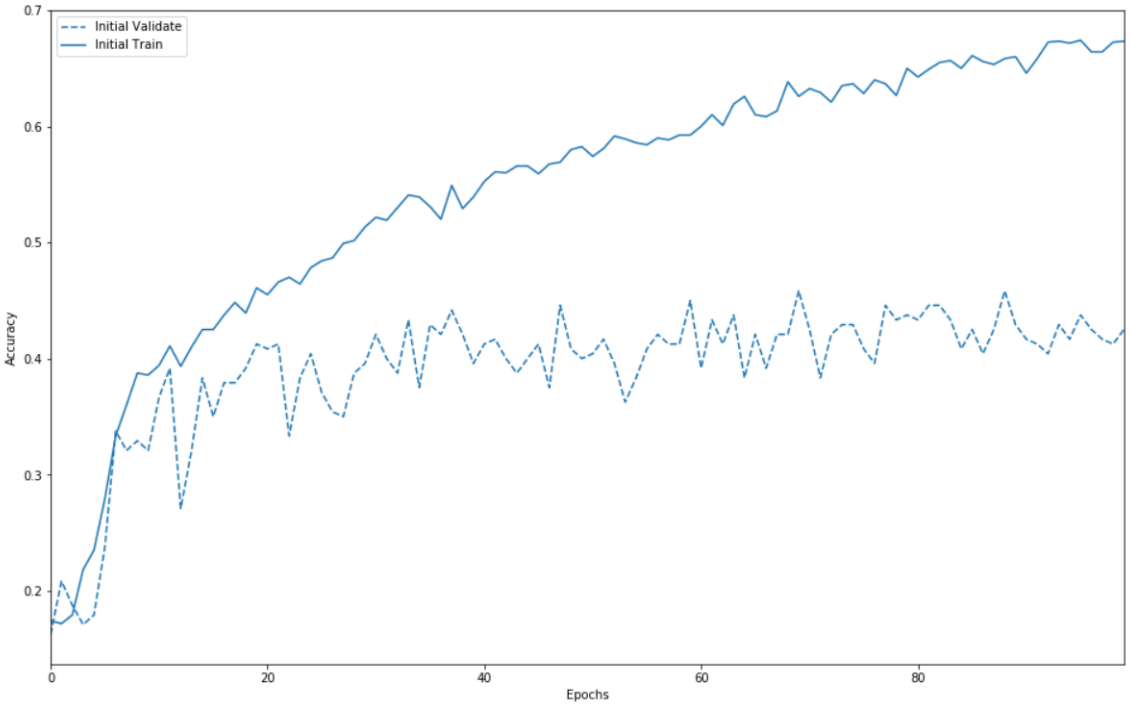

Figure 4: Γραφική παράσταση απόδοσης εκμάθησης και επιβεβαίωσης σε νέα δεδομένα για το αρχικό νευρωνικό

2.4 Τεχνικές ελαχιστοποίησης overfitting

Όπως εύκολα παρατηρείται, η καμπύλη απόδοσης της εκμάθησης έχει μεγάλη απόκλιση από αυτή στα καινούργια δεδομένα, και μάλιστα αυξανόμενη απόκλιση καθώς εκπαιδεύεται το μοντέλο, συγκεκριμένα είναι πολύ καλύτερη και συνεχώς βελτιούμενη. Αυτό συμβαίνει καθώς από ένα σημείο και μετά το μοντέλο μαθαίνει πάνω σε μοτίβα στα δεδομένα εκμάθησης τα οποία δεν χαρακτηρίζουν το σύνολο και δεν γενικεύονται, κάτι που ονομάζεται overfitting [14]. Για την αντιμετώπιση αυτού του φαινομένου εφαρμόστηκαν κάποιες βασικές τεχνικές [13]. Αρχικά, μειώθηκε το μέγεθος του μοντέλου, ενώ στη συνέχεια προσθέθηκε dropout και άλλοι regularizers [19].

```
[ ]  model_smaller = keras.Sequential()

     model_smaller.add(keras.layers.Dense(units = 32, activation = tf.nn.relu, input_dim = 68))
     model_smaller.add(keras.layers.Dense(units = 16, activation = tf.nn.relu))
     model_smaller.add(keras.layers.Dense(units = 8, activation = tf.nn.softmax))

     model_smaller.compile(optimizer = 'adam',
                   loss = 'sparse_categorical_crossentropy',
                   metrics = ['accuracy'])

     smaller_history = model_smaller.fit(Train_Data, Train_Labels, epochs = 100, validation_data = (Test_Data, Test_Labels))

     plot_history([('Initial', ini_history), ('Smaller', smaller_history)])
```

Figure 5: Μικρότερο μοντέλο

```
[ ]  model_overmin = keras.Sequential()

     model_overmin.add(keras.layers.Dense(units = 32, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.selu, input_dim = 68))
     model_overmin.add(keras.layers.Dropout(0.2))
     model_overmin.add(keras.layers.Dense(units = 16, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.selu))
     model_overmin.add(keras.layers.Dropout(0.2))
     model_overmin.add(keras.layers.Dense(units = 8, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.softmax))

     model_overmin.compile(optimizer = 'adam',
                   loss = 'sparse_categorical_crossentropy',
                   metrics = ['accuracy'])

     overmin_history = model_overmin.fit(Train_Data, Train_Labels, epochs = 100, validation_data = (Test_Data, Test_Labels))

     plot_history([('Initial', ini_history), ('Smaller', smaller_history), ('Regularizers and Dropout', overmin_history)])
```

Figure 6: Μοντέλο με regularizers και dropout (πάνω στο μικρότερο μοντέλο)

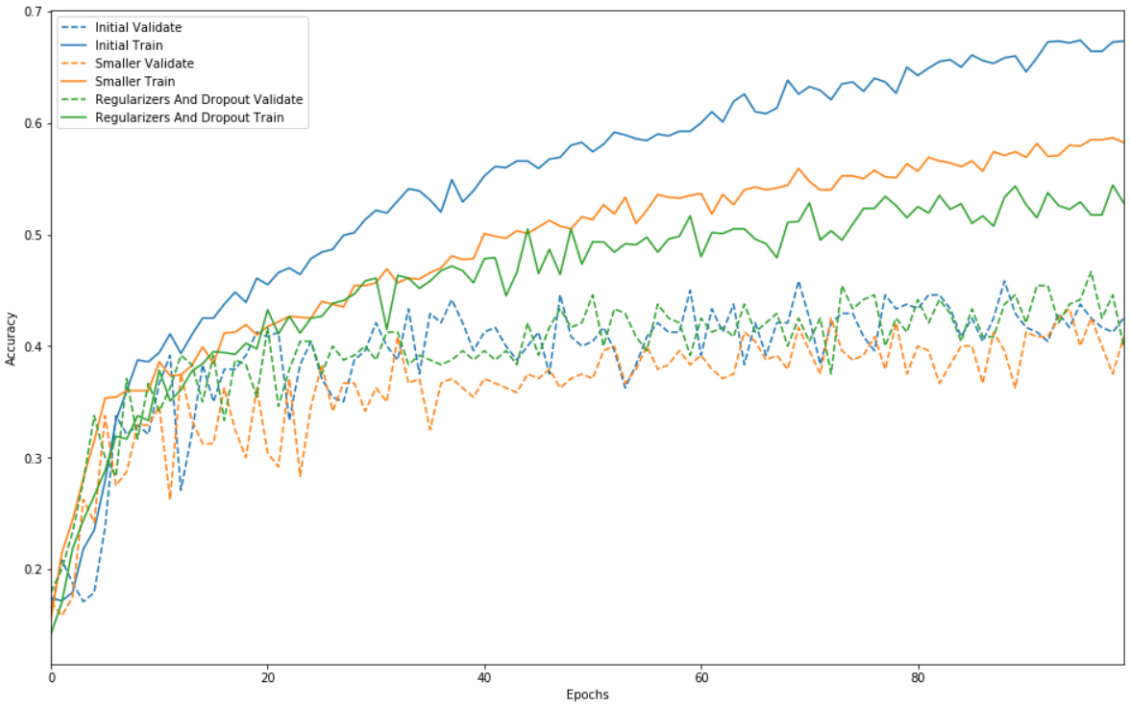

Figure 7: Σύγκριση καμπυλών απόδοσης

Παρατηρείται ότι οι καμπύλες εκμάθησης και νέων δεδομένων παρουσιάζουν σημαντικά μικρότερη απόκλιση μετά την εφαρμογή των παραπάνω τεχνικών, και μάλιστα απόκλιση με μικρότερο ρυθμό αύξησης καθώς το μοντέλο εκπαιδεύεται. Όμως η ακρίβεια στις προβλέψεις παραμένει σταθερή κοντά στο 40%, οπότε κρατώντας το μοντέλο με τους regularizers, αναζητήθηκε ένα υποσύνολο των χαρακτηριστικών εισόδου που θα βελτιώσει αυτό το ποσοστό.

2.5 Επιλογή καλύτερου υποσυνόλου χαρακτηριστικών και βελτίωση απόδοσης σε άγνωστα δεδομένα

Μετά από μελέτη των δημοσιεύσεων [20, 21], και κρίνοντας ότι τα Chroma Features είναι ασύνδετα με το συναίσθημα στην ομιλία (θα ήταν, πιθανώς, χρήσιμα για το συναίσθημα στη μουσική) [22], αποφασίσθηκε το εξής υποσύνολο χαρακτηριστικών (συμμετρικά σε μέση τιμή και σε τυπική απόκλιση):
1. Zero-Crossing Rate, 2. Energy, 3. Energy Entropy, 9-21: Mel Frequency Cepstrum Coefficients (MFCCs).
δηλαδή συνολικά 16 χαρακτηριστικά (32 τιμές). Αυτά αποκτήθηκαν από τα αρχικά csv files μέσω του cutfeats.py. Στη συνέχεια δοκιμάστικαν οι εναλλακτικές relu/selu στο μοντέλο της προηγούμενης υποενότητας. Σημειώνεται ότι το μοντέλο τώρα εκπαιδεύτηκε σε 1000 αντί για 100 εποχές για να αποκτηθεί μια

καλύτερη εικόνα της οριακής κατάστασης.

```
upl = files.upload()
Train_Data = pd.read_csv('train_dataSFLCUT2.csv', header = None)
Train_Labels = pd.read_csv('train_labelsSFL.csv', header = None)
Test_Data = pd.read_csv('test_dataSFLCUT2.csv', header = None)
Test_Labels = pd.read_csv('test_labelsSFL.csv', header = None)
```

Choose Files 4 files

- **test_dataSFLCUT2.csv**(application/vnd.ms-excel) - 148170 bytes, last modified: 9/27/2019 - 100% done
- **test_labelsSFL.csv**(application/vnd.ms-excel) - 720 bytes, last modified: 9/26/2019 - 100% done
- **train_dataSFLCUT2.csv**(application/vnd.ms-excel) - 740246 bytes, last modified: 9/27/2019 - 100% done
- **train_labelsSFL.csv**(application/vnd.ms-excel) - 3600 bytes, last modified: 9/26/2019 - 100% done

```
Saving test_dataSFLCUT2.csv to test_dataSFLCUT2.csv
Saving test_labelsSFL.csv to test_labelsSFL.csv
Saving train_dataSFLCUT2.csv to train_dataSFLCUT2.csv
Saving train_labelsSFL.csv to train_labelsSFL.csv
```

Figure 8: Φόρτωση νέων αρχείων εισόδου

```
[ ] model_relu = keras.Sequential()

    model_relu.add(keras.layers.Dense(units = 32, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.relu, input_dim = 32))
    model_relu.add(keras.layers.Dropout(0.2))
    model_relu.add(keras.layers.Dense(units = 16, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.relu))
    model_relu.add(keras.layers.Dropout(0.2))
    model_relu.add(keras.layers.Dense(units = 8, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.softmax))

    model_relu.compile(optimizer = 'adam',
                  loss = 'sparse_categorical_crossentropy',
                  metrics = ['accuracy'])

    relu_history = model_relu.fit(Train_Data, Train_Labels, epochs = 1000, validation_data = (Test_Data, Test_Labels))

    plot_history([('Subset feature vector', relu_history)])
```

Figure 9: Μοντέλο με relu

```
model_selu = keras.Sequential()

model_selu.add(keras.layers.Dense(units = 32, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.selu, input_dim = 32))
model_selu.add(keras.layers.Dropout(0.2))
model_selu.add(keras.layers.Dense(units = 16, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.selu))
model_selu.add(keras.layers.Dropout(0.2))
model_selu.add(keras.layers.Dense(units = 8, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.softmax))

model_selu.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])

selu_history = model_selu.fit(Train_Data, Train_Labels, epochs = 1000, validation_data = (Test_Data, Test_Labels))

plot_history([('Subset feature vector', relu_history), ('Subset feature vector selu', selu_history)])
```

Figure 10: Μοντέλο με selu

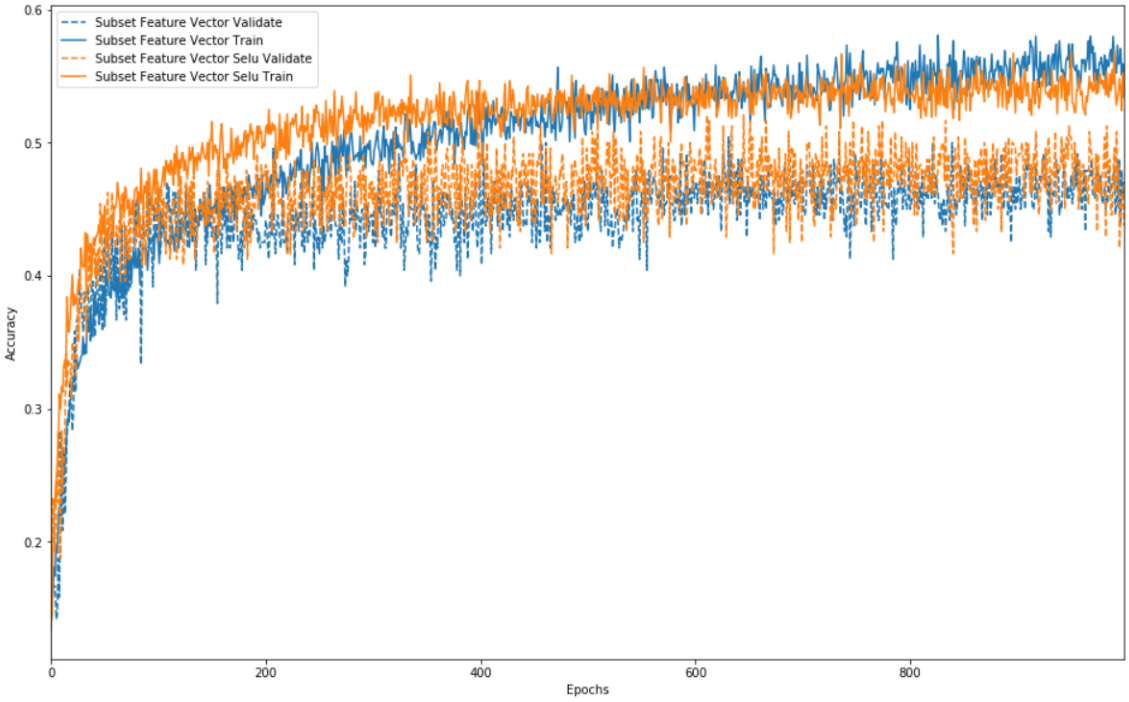

Figure 11: Σύγκριση καμπυλών απόδοσης relu-selu

Όπως φαίνεται παραπάνω, η απόδοση και των δύο μοντέλων ειναι αρκετά καλύτερη από ότι με το πλήρες σύνολο χαρακτηριστικών, καθώς έχει ανέβει 4-8 ποσοστιαίες μονάδες. Συγκεκριμένα, στο μοντέλο με relu η απόδοση στα νέα δεδομένα είναι 43%-46% ενώ στο μοντέλο με selu 45%-49%. Μια άλλη διαφορά ανάμεσα στα δύο μοντέλα, είναι ότι το selu φαίνεται να κάνει overfit πιο γρήγορα, αλλά οριακά να διατηρεί καμπύλη εκμάθησης μηδενικής κλίσης και άρα να παρουσιάζει σταθερή απόκλιση, σε αντίθεση με το relu που η απόκλιση αυξάνεται πιο σταδιακά αλλά αυξάνεται σταθερά με μη μηδενική κλίση. Σύμφωνα με τα παραπάνω, αλλα και για λόγους λειτουργικότητας [23], προτιμάται το selu ως το τελικό μοντέλο.

2.6 Cross-validation

Στη συνέχεια επιλέχθηκε να εφαρμοστεί άλλη μια δημοφιλής τεχνική, το cross-validation, στοχεύοντας στην ελαχιστοποίηση του overfitting αλλά και στη γενικότερη βελτιστοποίηση του μοντέλου και καλύτερη εκμετάλλευση ενός μικρού dataset [24, 25]. Δοκιμάστηκαν δύο είδη cross-validation, πρώτα το Leave One Out Cross Validation, και στη συνέχεια το KFold, και τα δύο με χρήση μεθόδων της βιβλιοθήκης της Python scikit learn. Πρώτα χρειάστηκε να εφαρμοστεί concatenation των αρχείων train και test, κάτι που επιτεύχθηκε με το mkfilesforloocv.py, ώστε οι μέθοδοι που καλούνται να κάνουν τον δικό τους χωρισμό.

```
upl = files.upload()
Data = pd.read_csv('dataSFLCUT2.csv', header = None)
Labels = pd.read_csv('labelsSFL.csv', header = None)
```

Choose Files | 2 files

- **dataSFLCUT2.csv**(application/vnd.ms-excel) - 888416 bytes, last modified: 10/12/2019 - 100% done
- **labelsSFL.csv**(application/vnd.ms-excel) - 4320 bytes, last modified: 10/12/2019 - 100% done

```
Saving dataSFLCUT2.csv to dataSFLCUT2.csv
Saving labelsSFL.csv to labelsSFL.csv
```

Figure 12: Φόρτωση νέων αρχείων εισόδου

```
[ ]  import tensorflow as tf
     from tensorflow import keras
     import pandas as pd
     from google.colab import files
     import matplotlib.pyplot as plt
     import numpy as np
     import sklearn.model_selection as ms
```

Figure 13: Συμπεριλήφθηκε η βιβλιοθήκη sklearn.model_selection

```
[ ]  loo = LeaveOneOut()

     cvscores = []
     for train_index, test_index in loo.split(Data):
       model_selu.fit(Data.to_numpy()[train_index], Labels.to_numpy()[train_index], epochs = 1)
       scores = model_selu.evaluate(Data.to_numpy()[test_index], Labels.to_numpy()[test_index])
       cvscores.append(scores[1] * 100)
     print("%.2f%% (+/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))
```

Figure 14: Εκπαίδευση μοντέλου με LOOCV

```
Train on 1439 samples
1439/1439 [==============================] - 0s 53us/sample - loss: 1.3894 - acc: 0.5469
1/1 [==============================] - 0s 810us/sample - loss: 1.4451 - acc: 1.0000
Train on 1439 samples
1439/1439 [==============================] - 0s 51us/sample - loss: 1.3933 - acc: 0.5448
1/1 [==============================] - 0s 1ms/sample - loss: 0.4375 - acc: 1.0000
Train on 1439 samples
1439/1439 [==============================] - 0s 56us/sample - loss: 1.3865 - acc: 0.5372
1/1 [==============================] - 0s 1ms/sample - loss: 0.6414 - acc: 1.0000
Train on 1439 samples
1439/1439 [==============================] - 0s 47us/sample - loss: 1.4066 - acc: 0.5281
1/1 [==============================] - 0s 1ms/sample - loss: 2.9282 - acc: 0.0000e+00
53.96% (+/- 49.84%)
```

Figure 15: Αποτέλεσμα LOOCV

```
kf = ms.KFold(5)
cvscores = []
for train_index, test_index in kf.split(Data):
  model_selu = keras.Sequential()

  model_selu.add(keras.layers.Dense(units = 32, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.selu, input_dim = 32))
  model_selu.add(keras.layers.Dropout(0.2))
  model_selu.add(keras.layers.Dense(units = 16, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.selu))
  model_selu.add(keras.layers.Dropout(0.2))
  model_selu.add(keras.layers.Dense(units = 8, kernel_regularizer=keras.regularizers.l2(0.001), activation = tf.nn.softmax))

  model_selu.compile(optimizer = 'adam',
             loss = 'sparse_categorical_crossentropy',
             metrics = ['accuracy'])
  model_selu.fit(Data.to_numpy()[train_index], Labels.to_numpy()[train_index], epochs = 200)
  scores = model_selu.evaluate(Data.to_numpy()[test_index], Labels.to_numpy()[test_index])
  cvscores.append(scores[1] * 100)
print("%.2f%% (+/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))
```

Figure 16: Εκπαίδευση μοντέλου με KFold των 5 folds [26]

```
Epoch 196/200
1152/1152 [==============================] - 0s 51us/sample - loss: 1.4451 - acc: 0.5139
Epoch 197/200
1152/1152 [==============================] - 0s 61us/sample - loss: 1.4606 - acc: 0.5113
Epoch 198/200
1152/1152 [==============================] - 0s 55us/sample - loss: 1.4582 - acc: 0.4931
Epoch 199/200
1152/1152 [==============================] - 0s 57us/sample - loss: 1.4619 - acc: 0.5026
Epoch 200/200
1152/1152 [==============================] - 0s 58us/sample - loss: 1.4561 - acc: 0.5017
288/288 [==============================] - 0s 365us/sample - loss: 1.5405 - acc: 0.4792
48.75% (+/- 1.72%)
```

Figure 17: Αποτέλεσμα KFold

Όπως φαίνεται, το LOOCV (το οποίο είναι εφαρμόσιμο μόνο επειδή έχουμε μικρό dataset), εκπαιδεύει καλύτερα το μοντέλο, καθώς εκμεταλλεύεται ολόκληρη τη βάση (εκπαίδευση σε 1439 δείγματα κάθε φορά, αντί για 1200 που είχαμε πριν), οπότε επιτυγχάνεται καλύτερο ποσοστό ακριβείας (μέση τιμή $\approx 54\%$) - με παρόμοιο ποσοστό στα δεδομένα εκμάθησης. Από την άλλη, λόγω του ότι τεστάρεται κάθε φορά μόνο σε ένα δείγμα, έχει υψηλή μεταβλητότητα (μεγάλη τυπική απόκλιση). Παρατηρείται επίσης ότι το KFold με 5 folds πετυχαίνει μικρότερη ακρίβεια (μέση τιμή $= 48.75\%$), αλλά επίσης και μικρότερο σφάλμα μεταβλητότητας. Επίσης να σημειωθεί ότι η παράμετρος 5 πετυχαίνει ένα διαχωρισμό (1152-288) παρόμοιο με τον manual που είχε γίνει νωρίτερα (1200-240), οπότε μπορούμε να παρατηρήσουμε μία ακριβή εκτίμηση της απόδοσης του μοντέλου που κατασκευάστηκε με τον αρχικό διαχωρισμό. Το καθαρό αποτέλεσμα του LOOCV είναι, εμφανώς, το προτιμότερο.

3 Αποτελέσματα

Συνοψίζοντας, το τελικό μοντέλο (small size, regularizers, selu), με το τελικό feature vector (16 features) και με χρήση leave one out cross-validation, δηλαδή με μέγιστη εκμετάλλευση του dataset, πετυχαίνει κατά μέσο όρο 54% ακρίβεια, αποτέλεσμα που κρίνεται ικανοποιητικό για το μέγεθος και το είδος της βάσης καθώς και το πλήθος των συναισθημάτων προς αναγνώριση. Επιπλέον, επιτεύχθηκε

από ελαχιστοποίηση του overfitting με τις πρώτες μεθόδους που εφαρμόστηκαν, έως και εξάλειψη του με το leave one out cross-validation.

4 Συμπεράσματα και μελοντική δουλειά

Από την παραπάνω δουλειά βγαίνει το συμπέρασμα ότι τα μεγαλύτερα ζητήματα μελέτης στην αναγνώριση συναισθήματος με βαθειά Νευρωνικά Δίκτυα αφορούν την επιλογή της κατάλληλης βάσης και των κατάλληλων χαρακτηριστικών, αλλά και την αντιμετώπιση του overfitting του μοντέλου. Με όσα και στο βαθμό που μελετήθηκαν, το ποσοστό το οποίο επιτεύχθει είναι μια καλή αρχή, ενώ είναι πολύ πάνω του τυχαίου και λειτουργικό στο βαθμό που δεν βασίζονται άλλες αποφάσεις πάνω στην αναγνώριση. Αυτή η εργασία που έγινε στο πλαίσιο του μαθήματος της Ψηφιακής Επεξεργασίας Σήματος αφήνει πολλά σημεία να χρήζουν παραπάνω μελέτης, όπως το θέμα της επιλογής βέλτιστου συνόλου χαρακτηριστικών, της δοκιμής άλλων βάσεων δεδομένων φωνής (π.χ. Emo-DB [2]) και της εξαγωγής συμπερασμάτων σχετικά με τα πιο εύκολα συγχεόμενα συναισθήματα μέσω confusion matrices, σημεία που στοχεύεται να μελετηθούν σε μελοντική εργασία.

Αναφορές

[1] Livingstone, S.R., Russo, F.A. (2018). The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5): e0196391. https://doi.org/10.1371/journal.pone.0196391

[2] Burkhardt, F., Paeschke, A., Rolfes, M., Sendlmeier, W. & Weiss, B. (2005). A database of German emotional speech. 9th European Conference on Speech Communication and Technology. 5. 1517-1520.

[3] Ververidis, D., Kotropoulos, C., Pitas, I. "Automatic emotional speech classification", in Proc. 2004 IEEE Int. Conf. Acoustics, Speech and Signal Processing, vol. 1, pp. 593-596, Montreal, May 2004.

[4] Xiao, Z., Dellandrea, E., Dou W., Chen, L. "Features extraction and selection for emotional speech classification". 2005 IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS), pp.411- 416, Sept 2005.

[5] T.-L. Pao, Y.-T. Chen, J.-H. Yeh, P.-J. Li, "Mandarin emotional speech recognition based on SVM and NN", Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), vol. 1, pp. 1096-1100,September 2006.

[6] Xia Mao, Lijiang Chen, Liqin Fu, "Multi-level Speech Emotion Recognition Based on HMM and ANN", 2009 WRI World Congress, Computer Science and Information Engineering, pp.225-229, March 2009.

[7] B. Schuller, G. Rigoll, M. Lang, "Hidden Markov model-based speech emotion recognition", Proceedings of the IEEE ICASSP Conference on Acoustics,Speech and Signal Processing, vol.2, pp. 1-4, April 2003.

[8] Yashpalsing Chavhan, M. L. Dhore, Pallavi Yesaware, "Speech Emotion Recognition Using Support Vector Machine", International Journal of Computer Applications, vol.1, pp.6-9,February 2010

[9] Giannakopoulos, T. (2015). pyAudioAnalysis: An Open-Source Python Library for Audio Signal Analysis. PLoS ONE 10(12): e0144610. https://doi.org/10.1371/journal.pone.0144610

[10] Deng, L., O'Shaughnessy, D.D. (2003). Speech processing: a dynamic and optimization-oriented approach, 232.

[11] Jason Brownlee. (2016, 2019). How to Scale Machine Learning Data From Scratch With Python. Retrieved from https://machinelearningmastery.com/scale-machine-learning-data-scratch-python/

[12] Basic classification: Classify images of clothing. (n.d.). Retrieved from https://www.tensorflow.org/tutorials/keras/classification

[13] Explore overfit and underfit. (n.d.). Retrieved from https://www.tensorflow.org/tutorials/keras/overfit_and_underfit

[14] Jesus Rodriguez. (2019). Three Simple Theories to Help Us Understand Overfitting and Underfitting in Machine Learning Models. Retrieved from https://towardsdatascience.com/three-simple-theories-to-help-us-understand-overfitting-and-underfitting-in-machine-learning-models-54a4905a2222

[15] Chengwei. (2017). How to choose Last-layer activation and loss function. Retrieved from https://www.dlology.com/blog/how-to-choose-last-layer-activation-and-loss-function/

[16] TensorFlow. (2019, February 12). Build a deep neural network in 4 mins with TensorFlow in Colab [Video File]. Retrieved from https://www.youtube.com/watch?v=_VTtrSDHPwU

[17] Jovian Lin. (2018, February 17). categorical_crossentropy VS. sparse_categorical_crossentropy. Retrieved from https://jovianlin.io/cat-crossentropy-vs-sparse-cat-crossentropy/

[18] David Mack. (2018, April 9). How to pick the best learning rate for your machine learning project. Retrieved from https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2

[19] Shubham Jain. (2018, April 19). An Overview of Regularization Techniques in Deep Learning (with Python code). Retrieved from https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

[20] El Ayadi, M., Kamel, M.S., Karray, F. (2011). Survey on speech emotion recognition: Features, classification schemes, and databases. Pattern Recognition, 44(3), 572-587.

[21] Shen, P., Changjun, Z., Chen, X. (2011, August). Automatic speech emotion recognition using support vector machine. In Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology (Vol. 2, pp. 621-625). IEEE.

[22] Wikipedia contributors. (2019, February 21). Chroma feature. In Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/w/index.php?title=Chroma_feature&oldid=884367502

[23] Timo Böhm. (2018, August 28). A first Introduction to SELUs and why you should start using them as your Activation Functions. Retrieved from https://towardsdatascience.com/gentle-introduction-to-selus-b19943068cd9

[24] Adi Bronshtein. (2017, May 17). Train/Test Split and Cross Validation in Python. Retrieved from https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

[25] Renu Khandelwal. (2018, November 3). K fold and other cross-validation techniques. Retrieved from https://medium.com/datadriveninvestor/k-fold-and-other-cross-validation-techniques-6c03a2563f1e

[26] Jason Brownlee. (2016, May 26). Evaluate the Performance Of Deep Learning Models in Keras. Retrieved from https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/