ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

Σχολή Οικονομίας και Τεχνολογίας

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Deep Learning Assignment

Γεωργόπουλος Παναγιώτης 2407

Βερυκοκίδης Απόστολος 2415

Διδάσκων:

Θοδωρής Γιαννακόπουλος

Ιούνιος 2025

Εισαγωγικά - Τεχνικές Λεπτομέρειες

Για την εργασία αυτή του μαθήματος του Deep Learning, διαλέξαμε να επικεντρωθούμε στην κατεύθυνση του ήχου, και κατ επέκταση δηλαδή των CNNs, επιλέγοντας ως πρόβλημα την ανάπτυξη μοντέλου σχεδιασμένου για ικανοποιητικές αποδόσεις στην κατηγοριοποίηση ειδών μουσικής, και συγκεκριμένα κατηγοριοποίηση σε clips από διαφορετικά υποείδη της hard-techno.

Για την εκτέλεση της εργασίας δουλέψαμε με την Python 3.10, σε notebooks τα οποία τρέχαμε τοπικά και σε Google Colab directories. Αναφορικά με το dataset το οποίο χρησιμοποιήσαμε, τα δεδομένα του προέρχονται από τραγούδια της μουσικής αυτής, από τα οποία εξάγαμε clips διάρκειας 8 δευτερολέπτων το καθένα μέσω της βιβλιοθήκης Librosa της python.

Για την σωστή εξαγωγή των clips, η *librosa* “έκοβε” τα κομμάτια σύμφωνα με το *tracks.csv* αρχείο, το οποίο περιέχει το **filename** του κάθε κομματιού, και το **start-end** time για κάθε clip. Το πρόβλημα μας ήταν *single-label* classification, καθώς κάθε clip ανήκει σε ένα μόνο genre. Συνολικά, εξήχθησαν 1772 clips.

Δεδομένης της υποκειμενικότητας και το πόσο “σχετικά” είναι τα ήδη μουσικής, πόσο μάλλον εδώ που είναι ακόμα πιο “loosely defined” καθώς πρόκειται για υποκατηγοριοποίηση υποκατηγορίας (techno > hard-techno > our classes), κάνει όλο το πρόβλημα πιο challenging, οπότε έχουμε πιθανές αμφιβολίες ή ακόμα και βελτιώσεις για το data annotation.

Όλα τα data & content files είναι σε αυτό το public Google Drive link:

https://drive.google.com/drive/u/2/folders/1Wtcmi4grjVQK8G2KAdINsqbb7Qm0VIGg

Machine Learning Προσέγγιση

Προτού προχωρήσουμε στο να δημιουργήσουμε melgrams και CNN αντιμετώπιση του προβλήματος, θεωρήσαμε πως μια καλή πρώτη προσέγγιση θα ήταν να πειραματιστούμε με τη χρήση αλγορίθμων μηχανικής μάθησης. Για να το πετύχουμε αυτό, θα έπρεπε για κάθε clip να έχουμε εκτός του genre (ουσιαστικά label) στο οποίο ανήκει και το οποίο εξάγεται από το tracks.csv, και μια σειρά χαρακτηριστικών, τα οποία θα χρησιμοποιούνταν για το train και τα predictions από τους αλγορίθμους μηχανικής μάθησης τους οποίους θα χρησιμοποιούσαμε. Για να το πετύχουμε αυτό, κάναμε χρήση feature extraction από τα clips με τη βοήθεια της Librosa, και για κάθε clip αποκτήσαμε τα means των εξής χαρακτηριστικών: chroma shift, spectral centroid/bandwidth/rolloff, zero crossing rate και mel-frequency cepstral coefficients 0 έως 6 (για demonstration purposes έχουμε κρατήσει και κώδικα ο οποίος εξάγει τα features μέσω της Pyaudioanalysis βιβλιοθήκης). Τα χαρακτηριστικά μαζί με το label του κάθε clip αποθηκεύονται στο csv αρχείο "clip_features_and_labels_2.csv".

Ξεκινήσαμε τις machine learning μεθόδους με την χρήση **SVM**, έχοντας κάνει split τα features και labels των clips από το αντίστοιχο csv (X και Y αντίστοιχα) σε train και test sets, με αναλογία 80-20. Αφότου κάνουμε έλεγχο για missing values, εκτελέσαμε έναν απλό SVM χωρίς κάποιο tuning, και είχαμε τα εξής αποτελέσματα ανά genre:

	precision	recall	f1-score	support
bouncy	0.57	0.83	0.68	48
tekno	0.55	0.79	0.65	53
warzone	0.56	0.85	0.68	52
industrial	0.53	0.84	0.65	61
non-techno-drop	0.90	1.00	0.95	47
micro avg	0.60	0.86	0.71	261
macro avg	0.62	0.86	0.72	261
weighted avg	0.62	0.86	0.71	261
samples avg	0.68	0.86	0.74	261

Το precision βρισκόταν σε χαμηλά επίπεδα, ενώ το recall ήταν αρκετά υψηλό σχεδόν σε όλες τις κλάσεις. Η υψηλότερη δυσκολία εμφανιζόταν όπως φαίνεται στα είδη και *industrial* & *tekno*.

Με την χρήση **XGBoost** πήραμε καλύτερα αποτελέσματα γενικά όσον αφορά τα f1-scores καθώς αυξήθηκε το συνολικό precision, αλλά είχαμε *αξιοσημείωτη* πτώση στο recall:

	precision	recall	f1-score	support
bouncy	0.63	0.69	0.66	48
tekno	0.68	0.68	0.68	53
warzone	0.73	0.79	0.76	52
industrial	0.76	0.64	0.70	61
non-techno-drop	0.92	0.96	0.94	47
accuracy			0.74	261
macro avg	0.75	0.75	0.75	261
weighted avg	0.74	0.74	0.74	261

Το tuning του **XGBoost** μας βοήθησε σε έναν βαθμό, κυρίως όσον αφορά το recall:

for *TUNED* XGBoost

	precision	recall	f1-score	support
bouncy	0.66	0.65	0.65	48
tekno	0.67	0.72	0.69	53
warzone	0.75	0.83	0.79	52
industrial	0.82	0.67	0.74	61
non-techno-drop	0.92	0.98	0.95	47
accuracy			0.76	261
macro avg	0.76	0.77	0.76	261
weighted avg	0.76	0.76	0.76	261

Ο tuned **LightGBM** μας έδωσε λίγο ακόμα πιο *βελτιωμένα* αποτελέσματα:

for *TUNED* LightGBM

	precision	recall	f1-score	support
bouncy	0.70	0.69	0.69	48
tekno	0.70	0.70	0.70	53
warzone	0.74	0.81	0.77	52
industrial	0.76	0.67	0.71	61
non-techno-drop	0.94	1.00	0.97	47
accuracy			0.77	261
macro avg	0.77	0.77	0.77	261
weighted avg	0.76	0.77	0.76	261

Τέλος, προχωρήσαμε σε **voting** (soft vote stacking) με τους *XGBoost* και *LightGBM*, και σε αυτήν την περίπτωση είδαμε μια σχετική άνοδο στα metrics, με τα καλύτερα αποτελέσματα τα οποία μπορέσαμε να εξάγουμε από machine learning τεχνικές:

	precision	recall	f1-score	support
bouncy	0.71	0.67	0.69	48
tekno	0.66	0.70	0.68	53
warzone	0.77	0.83	0.80	52
industrial	0.78	0.69	0.73	61
non-techno-drop	0.92	0.98	0.95	47
accuracy			0.77	261
macro avg	0.77	0.77	0.77	261
weighted avg	0.77	0.77	0.76	261

Αυτό που ξεχωρίσαμε από τα αποτελέσματα συνολικά ήταν η *υψηλή απόδοση* στα **non-techno-drop** clips, την οποία *αναμέναμε*, καθώς πρόκειται για κομμάτια τα οποία θα μπορούσαν να χαρακτηριστούν και σαν “other”/misc.

Deep Learning - CNNs

Για το κομμάτι του Deep Learning, ξέραμε ότι θα χρειαζόταν να μετατρέψουμε τα clips σε εικόνες, και έτσι μέσω της librosa εξάγαμε mel-spectrograms, ένα για κάθε clip. Συνολικά, και λόγω κάποιων σφαλμάτων σε κάποια instances, συνολικά αποκτήσαμε 1772 mel-spectrograms. Στην συνέχεια, ακολούθησε mapping των spectrograms με τα αντίστοιχα label μέσω του csv και split τους στα 3 set (train, val, test) με αναλογία 70-15-15, και για τις 5 κλάσεις με τις οποίες ασχολείται το πρόβλημα, μας έμειναν τα εξής spectrograms:

Train clips : 872
Valid clips : 207
Test clips : 226

Είναι πολύ σημαντικό να σημειώσουμε ότι το split έγινε έτσι ώστε να βρίσκονται όλα τα clips τα οποία προέρχονταν από το ίδιο τραγούδι, στο ίδιο split. Αυτός ο έλεγχος γινόταν μέσω του spot_id στο tracks.csv, το οποίο είναι και το αναγνωριστικό του κάθε τραγουδιού.

Στην συνέχεια, μπορούσαμε να προχωρήσουμε με τις CNN αρχιτεκτονικές. Ξεκινήσαμε με απλά μοντέλα τα οποία είχαμε δοκιμάσει αρχικά σε μέρος των δεδομένων ενώ το annotation δεν ήταν ακόμα έτοιμο, και είχαμε score γύρω στο 50-60 σε train και validation accuracies, οπότε δεν προχωρήσαμε με αυτές.

Προχωρήσαμε σε νέες αρχιτεκτονικές, και ανεβάσαμε το *patience* στα 10 από το 5 στο οποίο ξεκινήσαμε καθώς είδαμε αρκετές περιπτώσει όπου το μοντέλο κατέληγε να ρίξει το *val_loss* του μετά από 4 ή και 6 epochs, ενώ το ανεβήκαμε στα 50 *Epochs*, καθώς αρκετές φορές το χαμηλότερο val_loss βρέθηκε μεταξύ των *epochs* 30-40.

Επίσης δοκιμάσαμε **δύο ειδών melgrams,** ένα τυπικό melgram με cutoff στα 8k hz, και ένα σαν *low pass filter* στα 2k hz, κόβοντας τις ψηλές συχνότητες ώστε το μοντέλο να μαθαίνει αποκλειστικά από τις χαμηλές.

Καθώς επίσης και **δύο μεγέθη για input**, 128x128 και 256x256, για να δούμε κατα πόσο η περισσότερη πληροφορία βοηθάει τα μοντέλα μας.

Όλα τα αποτελέσματα αφορούν μέσο όρο από 5 διαδοχικά runs, το κάθε run με ***διαφορετικό seed για το split****.*

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=input_shape,
padding='same'),
    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation='relu', padding='same'),
    MaxPooling2D(2,2),

    Conv2D(128, (3,3), activation='relu', padding='same'),
    MaxPooling2D(2,2),

    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(len(labels_list), activation='softmax')  # sigmoid since we got
multilabel
])
```

Ενώ χρησιμοποιήσαμε και **class weights**, για ακόμα καλύτερο balancing, ιδιαίτερα στα validation & test sets, που όντας μικρότερα είχαν περισσότερο fluxation στις αναλογίες των *κατανομών* τους.

Ενδεικτικά αποτελέσματα από 5 runs του αρχικού μας μοντέλου, για δύο ειδών melgrams και δύο διαφορετικά input sizes:

```
=== SUMMARY [melgrams | 128x128] ===
all F1s: ['0.8084', '0.7680', '0.7652', '0.7154', '0.7886']
mean +/- std: 0.7692 +/- 0.0311     << winner so far


=== SUMMARY [melgrams | 256x256] ===
all F1s: ['0.7427', '0.8115', '0.8277', '0.6881', '0.7459']
mean +/- std: 0.7632 +/- 0.0507
```

```
=== SUMMARY [low_melgrams | 128x128] ===
all F1s: ['0.7379', '0.7683', '0.7228', '0.7558', '0.7932']
mean +/- std: 0.7556 +/- 0.0244

=== SUMMARY [low_melgrams | 256x256] ===
all F1s: ['0.7726', '0.7796', '0.7869', '0.6714', '0.7426']
mean +/- std: 0.7506 +/- 0.0424
```

```
                precision    recall  f1-score   support

        bouncy       0.77      0.86      0.81        35
         tekno       0.81      0.70      0.75        30
       warzone       0.88      0.84      0.86        45
    industrial       0.79      0.81      0.80        47
non-techno-drop      0.94      0.96      0.95        50

      accuracy                           0.85       207
     macro avg       0.84      0.83      0.83       207
  weighted avg       0.85      0.85      0.84       207

Confusion matrix (val):
 [[30  1  0  4  0]
 [ 4 21  2  2  1]
 [ 0  3 38  3  1]
 [ 5  0  3 38  1]
 [ 0  1  0  1 48]]
```

Στην συνέχεια, δοκιμάσαμε και αλλάζοντας ελαφρώς το μοντέλο, απλώς για να δούμε αν θα πήγαινε καλύτερα:

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=input_shape),
    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Dropout(0.5),
    Flatten(),
    Dense(256, activation='relu'),
    Dense(len(CLASSES), activation='softmax')
```

```
])
```

Ουσιαστικά αλλάξαμε την σειρά με την οποία κλείνει το μοντέλο, με dropout -> flatten -> dense ροή, μήπως και βελτιωθεί το generalization/overfitting, και πήραμε καλύτερο f1-score σε test set από το 1ο μοντέλο.

```
=== SUMMARY [melgrams | 128x128] ===
all F1s: ['0.7688', '0.8140', '0.7768', '0.7237', '0.8008']
mean +/- std: 0.7768 +/- 0.0311          << new winner !

=== SUMMARY [melgrams | 256x256] ===
all F1s: ['0.7576', '0.7711', '0.8116', '0.5639', '0.8129']
mean +/- std: 0.7434 +/- 0.0924

=== SUMMARY [low_melgrams | 128x128] ===
all F1s: ['0.7711', '0.6985', '0.7081', '0.6500', '0.7336']
mean +/- std: 0.7123 +/- 0.0400

=== SUMMARY [low_melgrams | 256x256] ===
all F1s: ['0.7358', '0.7367', '0.7489', '0.7512', '0.6282']
mean +/- std: 0.7202 +/- 0.0464
```

Όλα τα αποτελέσματα αφορούν μέσο όρο από 5 διαδοχικά runs, το κάθε run με ***διαφορετικό seed για το split****.*

Επιπλέον 2 μοντέλα που δοκιμάστηκαν, και ενδεικτικά results:

```
def cnn_1_2(input_shape):
    return Sequential([
        Input(input_shape),
        Conv2D(32, (3, 3), activation='relu', padding='same'),
        MaxPooling2D(1, 2),

        Conv2D(64, (3, 3), activation='relu', padding='same'),
        MaxPooling2D(2, 2),

        Conv2D(128, (3, 3), activation='relu', padding='same'),
        MaxPooling2D(2, 2),

        Dropout(0.5),
        Flatten(),
```

```
        Dense(256, activation='relu'),
        Dense(len(CLASSES), activation='softmax')
    ])

full_test(cnn_1_2)
```

```
=== SUMMARY [melgrams | 128x128] ===
all F1s: ['0.8212', '0.8025', '0.7107', '0.6536', '0.8040']
mean +/- std: 0.7584 +/- 0.0652
```
*(το συγκεκριμένο σε **προηγούμενα** 5-run-tests έπιασα μέσο όρο f1 και **0.80**)*

```
=== SUMMARY [melgrams | 256x256] ===
all F1s: ['0.7098', '0.7429', '0.8223', '0.6135', '0.7860']
mean +/- std: 0.7349 +/- 0.0717

=== SUMMARY [low_melgrams | 128x128] ===
all F1s: ['0.7990', '0.7839', '0.7726', '0.7354', '0.7413']
mean +/- std: 0.7664 +/- 0.0245

=== SUMMARY [low_melgrams | 256x256] ===
all F1s: ['0.7183', '0.7373', '0.7962', '0.7102', '0.6940']
mean +/- std: 0.7312 +/- 0.0354
```

```
def cnn_2_1(input_shape):
    return Sequential([
        Input(input_shape),
        Conv2D(32, (3, 3), activation='relu', padding='same'),
        MaxPooling2D(2, 1),

        Conv2D(64, (3, 3), activation='relu', padding='same'),
        MaxPooling2D(2, 2),

        Conv2D(128, (3, 3), activation='relu', padding='same'),
        MaxPooling2D(2, 2),

        Dropout(0.5),
        Flatten(),
        Dense(256, activation='relu'),
        Dense(len(CLASSES), activation='softmax')
```

```
    ])

full_test(cnn_2_1)
```

```
=== SUMMARY [melgrams | 128x128] ===
all F1s: ['0.6957', '0.6642', '0.6419', '0.6904', '0.7201']
mean +/- std: 0.6825 +/- 0.0270
```

-από εκεί και πέρα και locally μας crash-άρει συνέχεια στα 256x256-

Ενδεικτικό confusion matrix + metrics του τελικού μας μοντέλου, πάντα με βάση το f1-score στο unseen test-set:

```
=== TEST-SET RESULTS ===
Weighted F1 (test): 0.7904
                precision    recall  f1-score   support

         bouncy       0.89      0.79      0.84        42
          tekno       0.67      0.67      0.67        46
        warzone       0.87      0.77      0.82        44
     industrial       0.55      0.74      0.63        42
non-techno-drop       1.00      0.92      0.96        52

       accuracy                           0.78       226
      macro avg       0.80      0.78      0.78       226
   weighted avg       0.81      0.78      0.79       226

Confusion matrix (test):
 [[33  0  0  9  0]
 [ 2 31  2 11  0]
 [ 0  6 34  4  0]
 [ 0  9  2 31  0]
 [ 2  0  1  1 48]]
```

Το μοντέλο εμφανίζει αρκετά καλή απόδοση (ειδικά σε σχέση με το Machine Learning κομμάτι), κάτι στο οποίο προφανώς ρόλο έπαιξε και η μείωση των κλάσεων λόγω αμφιβολιών για το annotation όπως αναφέραμε πριν**. Η καλύτερη επίδοση εμφανίζεται όπως περιμέναμε στην *non-techno-drop*** κλάση, με τις ***bouncy*** και ***warzone*** να ακολουθούν, ενώ οι ***tekno*** και ***industrial*** δυσκόλεψαν το μοντέλο, σε όλες τις φάσεις της ανάπτυξης του.

Transfer Learning

Αφότου εξάγαμε το μοντέλο, θελήσαμε να το χρησιμοποιήσουμε για transfer learning, για να δούμε την απόδοση του σε ένα τελείως διαφορετικό dataset. Αφότου αποθηκεύσαμε το τελικό μας μοντέλο σε μορφή .h5, κάναμε λήψη από το Kaggle το γνωστό music genre classification dataset GTZAN (https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification), το οποίο περιέχει spectrograms από clips που ανήκουν σε 10 διαφορετικά είδη μουσικής.

Μέσος όρος του f1-score στο τεστ σετ, μετά από 5 runs για Transfer Learning: ~0.626

Ενδεικτικά αποτελέσματα του μέσου-run εκ των 5, για το f1-score στο 15% *unseen* test-set του Transfer Learning.

```
# === TEST-SET RESULTS ===
# Weighted F1 (test): 0.6247
#               precision    recall  f1-score   support

#        blues       0.59      0.67      0.62        15
#    classical       0.81      0.87      0.84        15
#      country       0.44      0.53      0.48        15
#        disco       0.40      0.53      0.46        15
#       hiphop       0.56      0.60      0.58        15
#         jazz       0.85      0.73      0.79        15
#        metal       0.79      0.73      0.76        15
#          pop       0.64      0.60      0.62        15
#       reggae       0.89      0.53      0.67        15
#         rock       0.46      0.40      0.43        15

#     accuracy                           0.62       150
#    macro avg       0.64      0.62      0.62       150
# weighted avg       0.64      0.62      0.62       150

# Confusion matrix (test):
#  [[10  0  3  2  0  0  0  0  0  0]
#  [ 0 13  1  0  0  0  0  1  0  0]
```

```
#  [ 0  1  8  0  0  2  0  0  0  4]
#  [ 0  1  1  8  4  0  0  1  0  0]
#  [ 0  0  0  3  9  0  1  2  0  0]
#  [ 3  0  1  0  0 11  0  0  0  0]
#  [ 1  0  0  0  1  0 11  0  0  2]
#  [ 0  0  1  4  0  0  0  9  0  1]
#  [ 1  0  0  2  2  0  1  1  8  0]
#  [ 2  1  3  1  0  0  1  0  1  6]]
```