

# ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

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/1Wtcmi4grjVQK8G2KAdlNsqqb7Qm0VIGg>

## 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 περιπτώσεις 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.

```
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 fluctuation στις αναλογίες των κατανομών τους.

Ενδεικτικά αποτελέσματα από 5 runs του τελικού μας μοντέλου:

```

# run 1
# Best epoch = 14
# train_acc 0.8647 | train_loss 0.3721
# val_acc 0.8068 | val_loss 0.6143
# Weighted F1 (val) : 0.8050

```

```

# run 4
# Best epoch = 23
# train_acc 0.9025 | train_loss 0.2604
# val_acc 0.8213 | val_loss 0.5751
# Weighted F1 (val) : 0.8134

```

```

# run 2
# Best epoch = 22
# train_acc 0.9472 | train_loss 0.1642
# val_acc 0.8454 | val_loss 0.5026
# Weighted F1 (val) : 0.8448

```

```

# run 5
# Best epoch = 16
# train_acc 0.9071 | train_loss 0.2971
# val_acc 0.8696 | val_loss 0.5438
# Weighted F1 (val) : 0.8668

```

```

# run 3
# Best epoch = 14
# train_acc 0.8784 | train_loss 0.3704
# val_acc 0.8502 | val_loss 0.5496
# Weighted F1 (val) : 0.8482

```

```

Avg. Best Epoch 17.8
Avg. Val Loss 0.5571
Avg. Val Acc 0.8387
Avg. Val F1 0.8356
Avg. Test F1 0.7841

```

	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, αλλά το *val\_loss* έδειξε το ανάποδο με την άνοδό του.

Ενδεικτικά αποτελέσματα από 5 runs του μοντέλου με αυτήν την μορφή:



```
# run 1
# Best epoch = 19
# train_acc 0.8784 | train_loss 0.3446
# val_acc 0.7729 | val_loss 0.6463
# Weighted F1 (val) : 0.7778
```

```
# run 2
# Best epoch = 21
# train_acc 0.8911 | train_loss 0.3020
# val_acc 0.8019 | val_loss 0.6091
# Weighted F1 (val) : 0.8000
```

```
# run 3
# Best epoch = 31
# train_acc 0.9323 | train_loss 0.1961
# val_acc 0.7971 | val_loss 0.6636
# Weighted F1 (val) : 0.7954
```

```
# run 4
# Best epoch = 34
# train_acc 0.9656 | train_loss 0.1135
# val_acc 0.7874 | val_loss 0.6383
# Weighted F1 (val) : 0.7837
```

```
# run 5
# Best epoch = 17
# train_acc 0.8532 | train_loss 0.4371
# val_acc 0.7440 | val_loss 0.7363
# Weighted F1 (val) : 0.7460
```

```
Avg. Best Epoch 24.4
Avg. Val Loss 0.6587
Avg. Val Acc 0.7807
Avg. Val F1 0.7806
Avg. Test F1: 0.7714
```

	precision	recall	f1-score	support
bouncy	0.71	0.77	0.74	35
tekno	0.64	0.53	0.58	30
warzone	0.89	0.87	0.88	45
industrial	0.76	0.81	0.78	47
non-techno-drop	0.92	0.92	0.92	50
accuracy			0.80	207
macro avg	0.78	0.78	0.78	207
weighted avg	0.80	0.80	0.80	207

Confusion matrix (val):

```
[[27  4  0  4  0]
 [ 8 16  3  2  1]
 [ 0  0 39  4  2]
 [ 3  5  0 38  1]
 [ 0  0  2  2 46]]
```

(ενδεικτικό confusion matrix με val results)

Η πρώτη version του τελικού μας μοντέλου πάει ελάχιστα καλύτερα, άρα αυτήν διαλέξαμε σαν τελικό μας μοντέλο, και κάναμε evaluate στο τεστ 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]]
```

F1 από 5-repeat runs:

```
# Weighted F1 (test): 0.7621
# Weighted F1 (test): 0.8101
# Weighted F1 (test): 0.7641
# Weighted F1 (test): 0.7938
# Weighted F1 (test): 0.7904
Avg. Test F1: 0.7841
```

**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-classificat>

[ion](#)), 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]]
```