CNN - Based Music Genre Recognition

Georgopoulos Panagiotis 2407
Verykokidis Apostolos 2415

The “Problem”

- Different subgenres of techno music
- 8 second clips
- From more than 250 songs
- Single - label case

1st Approach - Let’s try some Machine Learning!

Incorporated some machine learning to start with, specifically SVM, XGBoost and LightGBM.

- Feature Extraction from the clips with both Librosa & pyAudioAnalysis
- 80-20 split
- Training the models

Best ML Results

All approaches were practically equal, but the best results were with the use of Soft Vote Stacking (*tuned XGBoost & tuned LightGBM*):

	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

2nd Approach: Moving on to the CNN - Basic Guidelines

Before setting up and training the model, some adjustments needed to be made.

- Creating the mel-spectrograms with the use of Librosa and Matplotlib
- Splitting the spectrograms into the Train-Validation-Test sets in accordance to the **track** they came from -> *all clips* from the ***same track*** go to the same split to avoid overfitting
- Using ***grayscale*** for melgrams, in order to save on computing power & time, and since mel-spectrograms represent intensity/power of frequency components over time, and each pixel encodes a scalar value; therefore RGB would be quite an overkill for our task

CNN First Attempts

The first implementations we did of CNN architectures were pretty simple, and done over a part of the data, since the annotation was a difficult and timely process.

- Training accuracies of no more than 50-60% for training and validation…
- …Which was the best case scenario, since the other option was extreme overfitting (80 train, 45 validation)
- Small architectural tweaks and different splits didn't help much

The solution? More data + class weighting!

Finally enough data (?) — Weighting classes (?)

After collecting more data (1772 mel-spectrograms of 128x128 input size), we got the final splits

- Train clips: 872
- Val clips: 207
- Test clips: 226

And we are ready to test more architectures!

Class weighting during training significantly improved generalization & more consistency in the results; and battled *imbalances* caused by uneven class distributions, especially in the validation & test sets, that are obviously smaller

Best Architecture

```
model = Sequential([
 Conv2D(32, (3,3), activation='relu', input_shape=(128, 128, 1), 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')
])
```

Results - Train/Validation

```
# 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 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 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

# 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 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

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

Final Evaluation on the test set

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

	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

Avg. Test F1-score **~0.78** *after 5 runs*

Transfer Learning case

GTZAN Music Genre Classification Dataset

- 10 Classes
- Spectrograms of 10 second snippets

Average test-set F1-score after 5 CNN runs ~0.626

The results…

```
# === 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
```

🎧 Thank you for your time! 🎵