Neural Networks and Deep Learning project

Project:

Music Genre Classification: enhancing the baseline architectures with DenseNet

Authors:

Eugenio Fella, Matteo Pedrazzi

Outline

1. Introduction
2. Dataset analysis
3. Pre-processing & data augmentation
4. Proposed architectures
5. Results
6. Comments and conclusions

Introduction

Music streaming platforms are constantly growing and the task of **genre classification**, useful to deliver finely tuned music recommendations, is not something that can be done manually anymore

Due to the subtlety of genre boundaries this is not a trivial task, however artificial intelligence and deep learning proved have proven to be efficient

FMA dataset

Free Music Archive (FMA) *small* dataset [1] https://github.com/mdeff/fma

- *fma_small*: 8000 mp3 tracks 30-seconds each, from 8 different genres (8.0 GB)
- *fma_metadata*: informations and features associated with tracks (1.5 GB)

The dataset is splitted in 80% training set, 10% validation and test set, properly removing corrupted audio files

	track_id	genre_top	top_genre_ind
0	122077	Rock	4
1	41568	Pop	1
2	114392	Electronic	6
3	47662	Folk	2
4	56692	International	5

Audio files

Audio files can be loaded and inspected using the **librosa** library [2] with the possibility of fixing the sampling rate

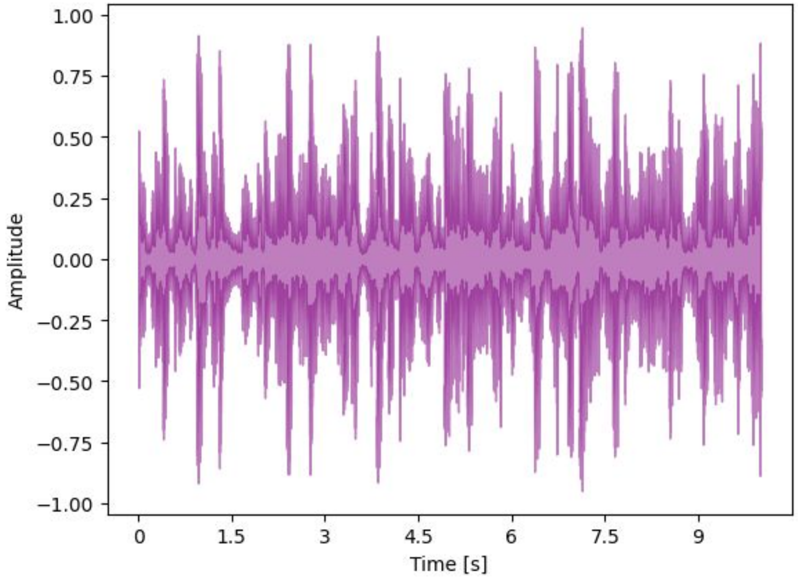
sampling rate = 16000

Track_id: 36959

Genre: Pop

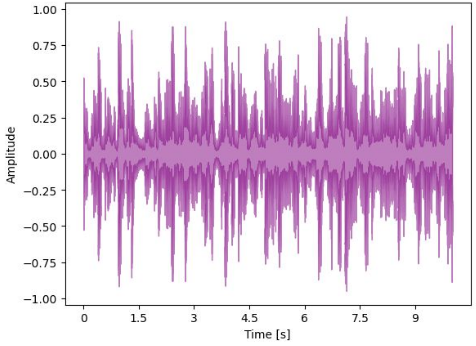
10 sec.

```
File: /content/drive/MyDrive/project/fma_small/036/036959.mp3
Duration: 29.98s, 479626 samples
[ 5.5964757e-09  6.8681225e-09  8.3595832e-09 ... -9.0769986e-03
  1.1497736e-03  0.0000000e+00]
```

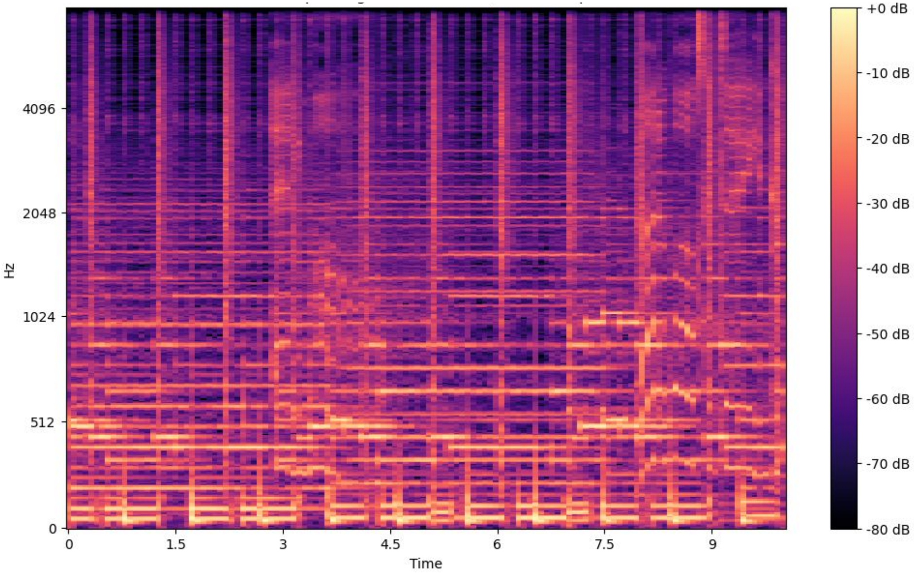


Mel-spectrograms

Using `librosa` to compute the **STFT**, setting the window parameters in order to have desired output shape and then converting it to **mel-scale** and to dB



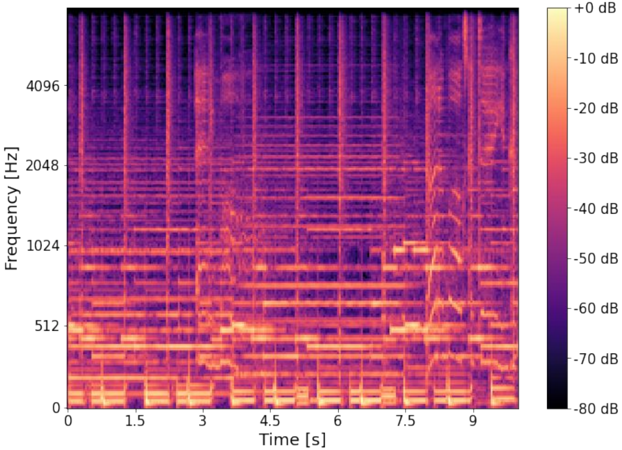
1D raw waveform signal



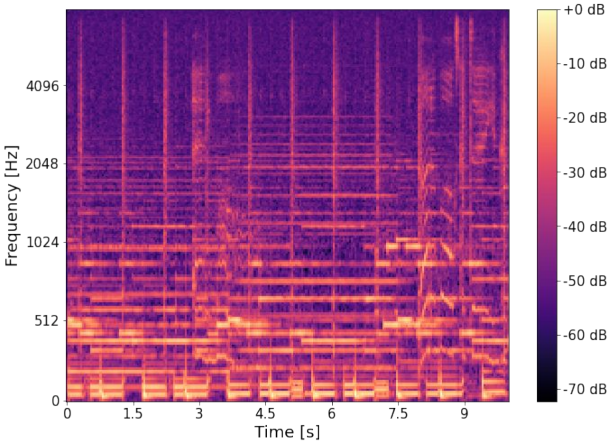
2D mel-spectrogram matrix

Data augmentation

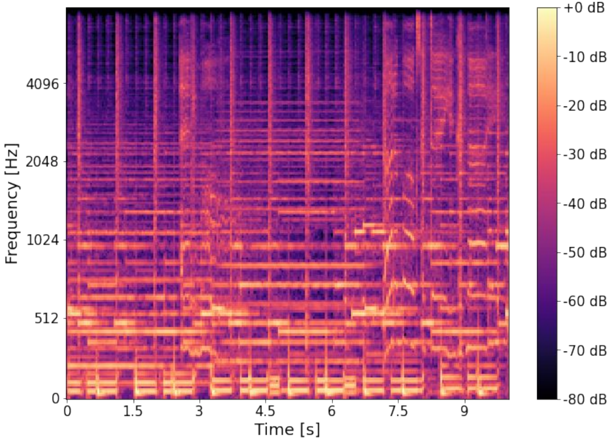
Original mel-spectrogram



white noise



time-stretching



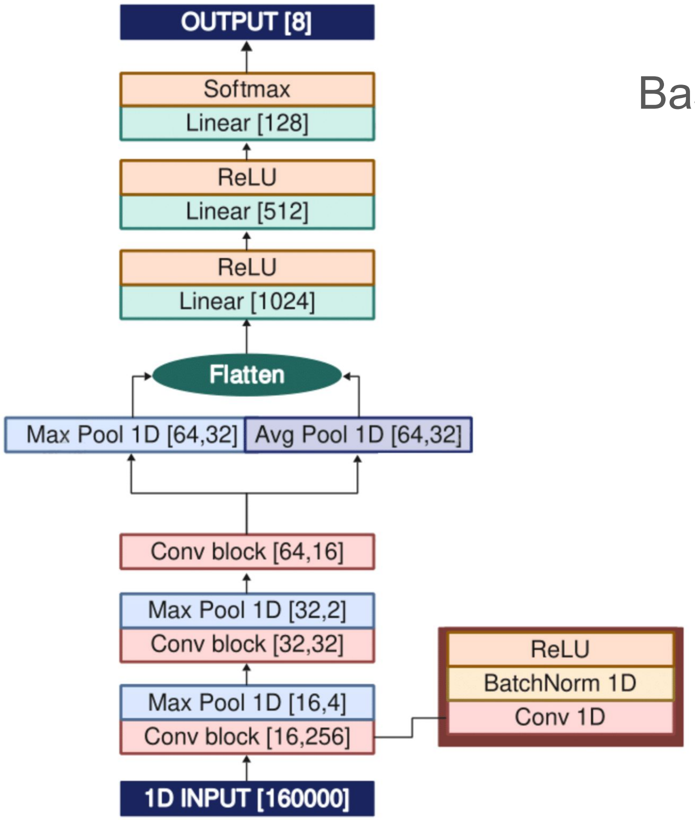
Architectures (1)

- **Baseline A** (1D&2D) ➪ Batch size: 16 & 32
- **Baseline B** (2D) ➪ Batch size: 32
- **DenseNet** (2D) ➪ Batch size: 32
- **FusionNet** (1D+2D) ➪ Batch size: 8

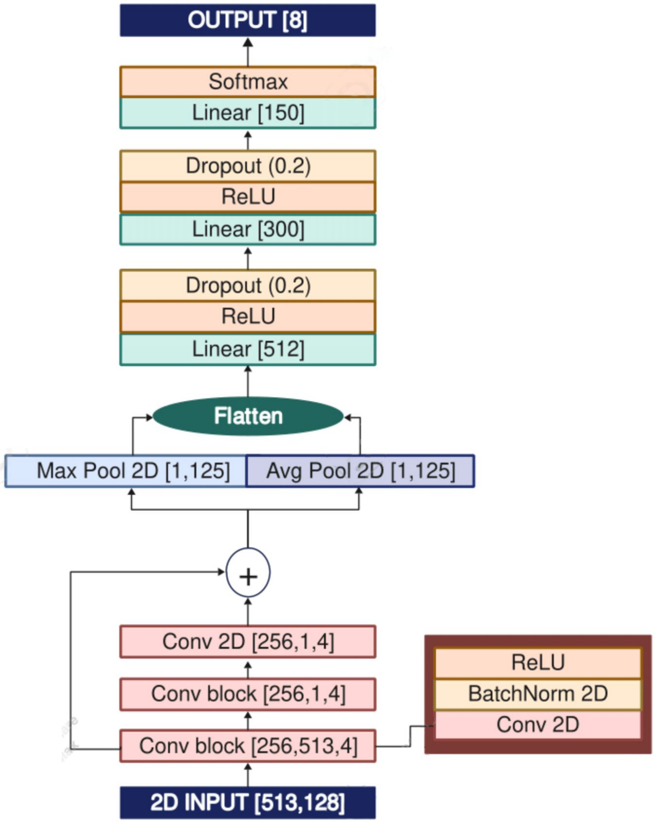
Runs with Adam optimizer, with learning rate and weight decay 10^{-4}, and considering cross entropy loss.

Architectures (2) - Baseline A and Baseline B

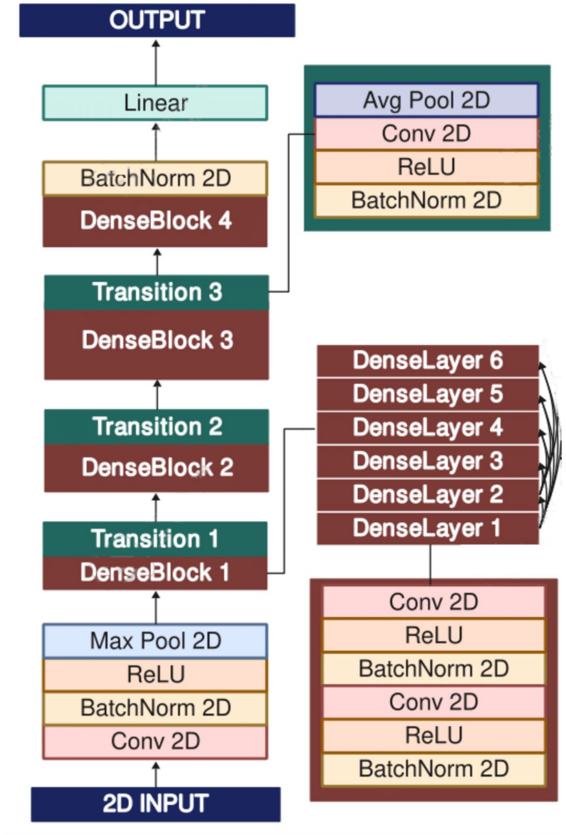
Baseline A:



Baseline B:



Architectures (3) - DenseNet



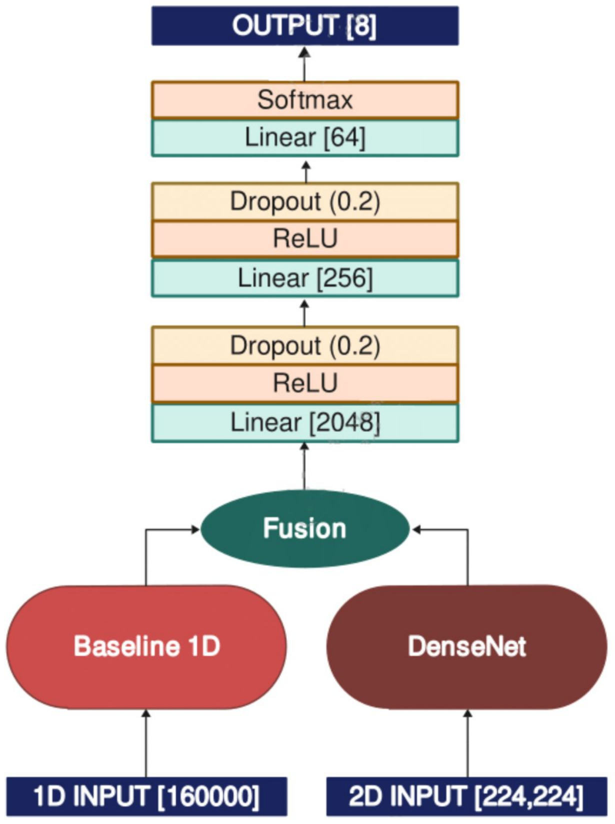
Architecture from computer vision image recognition tasks revealed to be efficient also on MGC [4].
We modified the `densenet121` PyTorch architecture to handle correctly sized inputs and outputs

- One convolutional block
- Alternation of **dense blocks** (containing densely connected dense layers) and **transition blocks**
- One layer linear classifier

Architectures (4) - FusionNet

Infeasibility of *early fusion* forces to adopt ***late fusion*** technique, concatenating the output of the Baseline A architecture for 1D data together with the one of the DenseNet, along the time axis

- Baseline A // DenseNet
- Late fusion
- Linear classifier



Results (1)

We tested the performances of the networks considering 4 different metrics:

$$\text{precision} = \frac{1}{k}\sum_{i=1}^{k}\frac{TP_i}{TP_i + FP_i}$$

$$\text{recall} = \frac{1}{k}\sum_{i=1}^{k}\frac{TP_i}{TP_i + FN_i}$$

$$\text{accuracy} = \frac{1}{k}\sum_{i=1}^{k}\frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}$$

$$\text{F1-score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

where *k* indicates the total number of classes

Results (2)

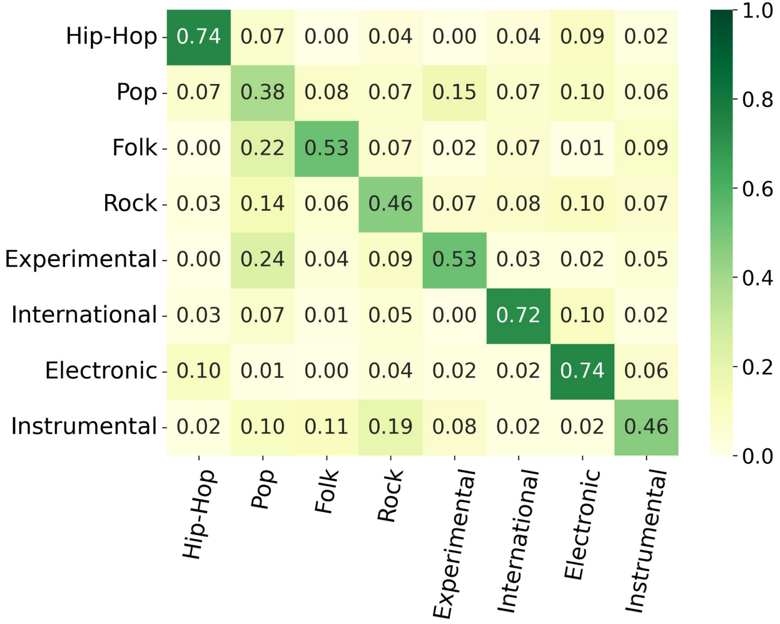
All architectures run for 20 epochs, keeping until the lower validation error

	F1-score	Accuracy	Precision	Recall
1D Baseline A	0.448	0.864	0.438	0.457
2D Baseline A	0.558	0.889	0.556	0.559
Baseline B	0.534	0.883	0.537	0.531
DenseNet	**0.574**	**0.892**	**0.578**	**0.571**
FusionNet	0.553	0.888	0.548	0.557

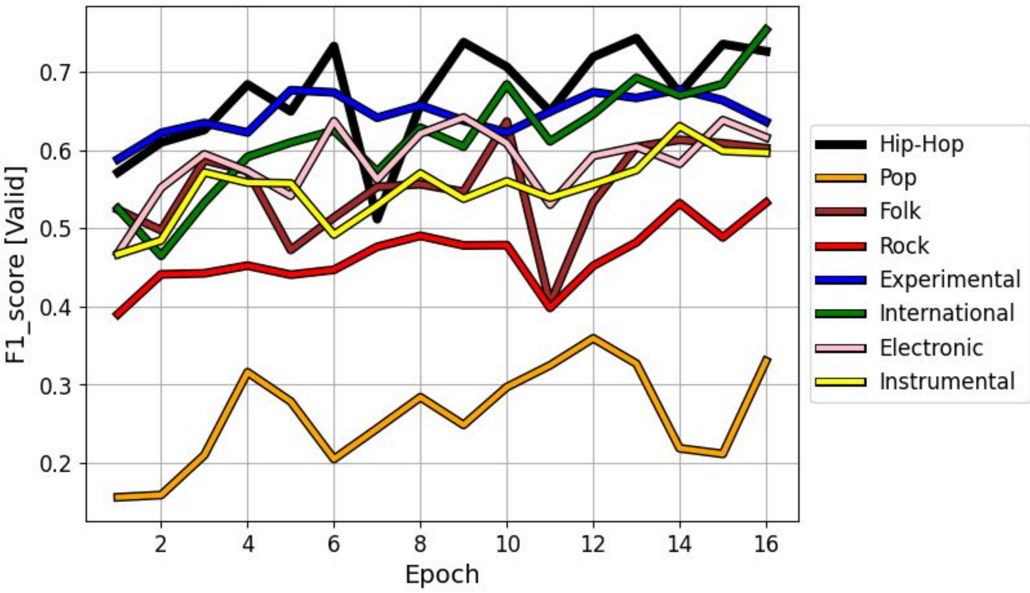
- DenseNet is achieving better performances in all the considered metrics
- 2D input proves to be more informative than the 1D input
- FusionNet is not able to outperform the DenseNet F1-score and accuracy

Results (3)

Confusion matrix (Densenet):



F1 score on the validation set:



Comments and conclusions

Although the results of **DenseNet** do not significantly surpass those of simpler architectures, the use of even deeper convolutional neural networks, that have been successful in various computer vision tasks, on mel-spectrograms emerges as promising for further exploration. This is especially true when larger data sets can be exploited, to effectively counteract overfitting problems.

Notably, we also explored the **fusion** of both types of input data, albeit with less encouraging results. This suggests the need for further exploration, potentially involving advanced network architectures for the 1D signal and refined fusion strategies that make better use of temporal information.

References

[1] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, “FMA: A dataset for music analysis,” in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

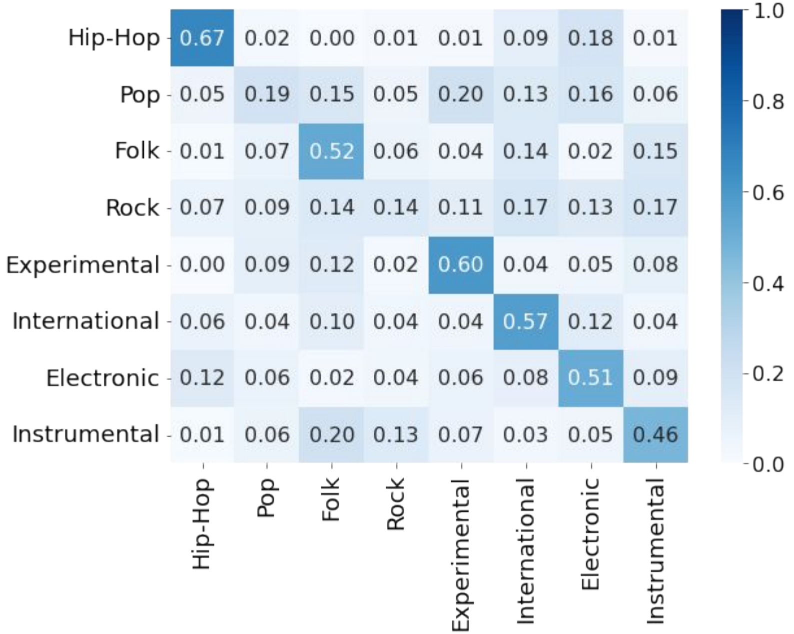
[2] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and music signal analysis in python,” in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[3] W. Zhang, W. Lei, X. Xu, and X. Xing, “Improved Music Genre Classification with Convolutional Neural Networks,” in *Proc. Interspeech 2016*, pp. 3304–3308, Sept. 2016.

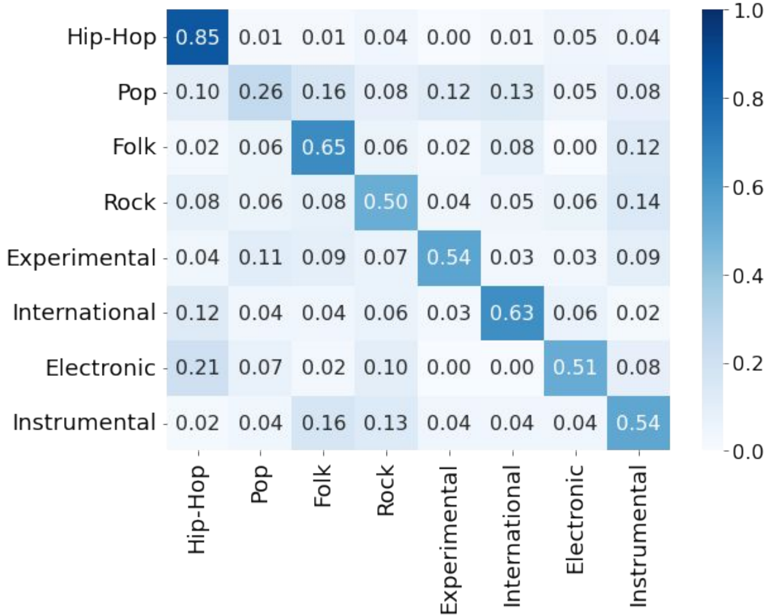
[4] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” 2018.

Backup slides:

Baseline A (1D):

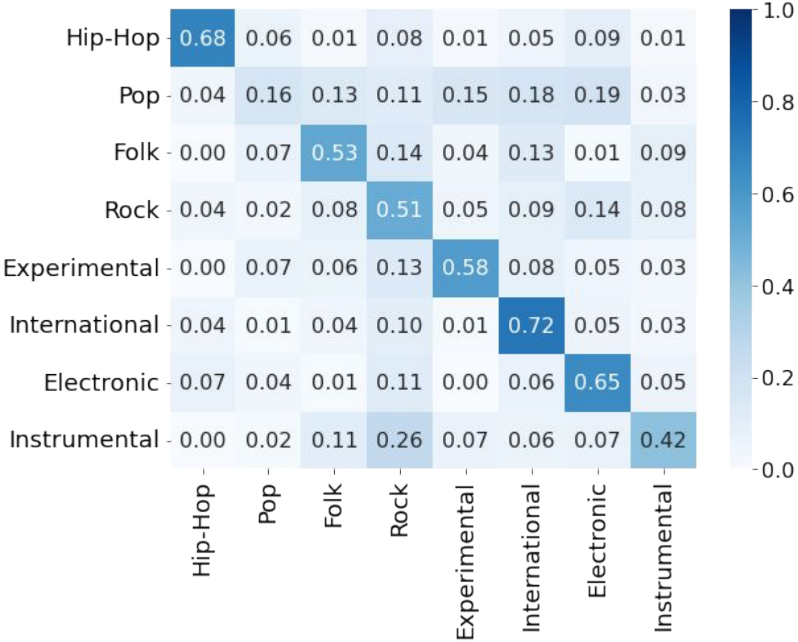


Baseline A (2D):

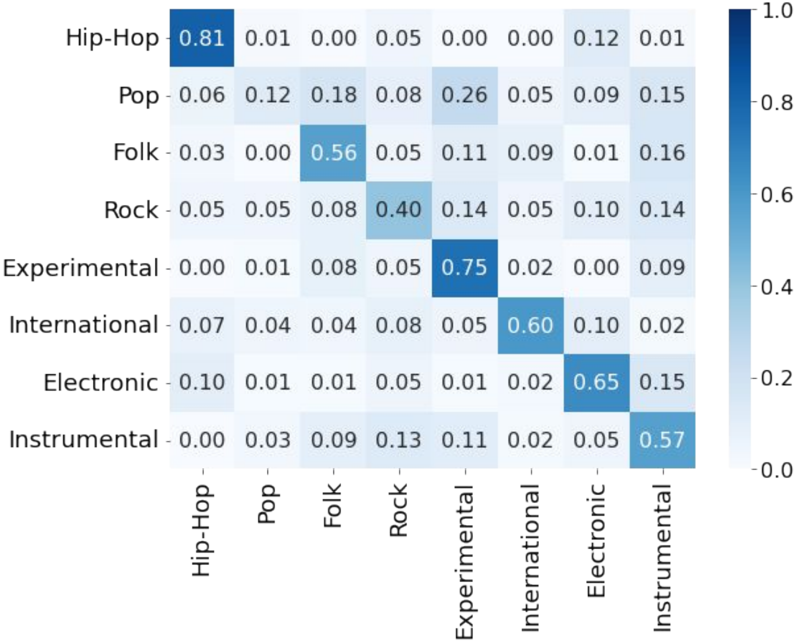


Backup slides:

Baseline B:



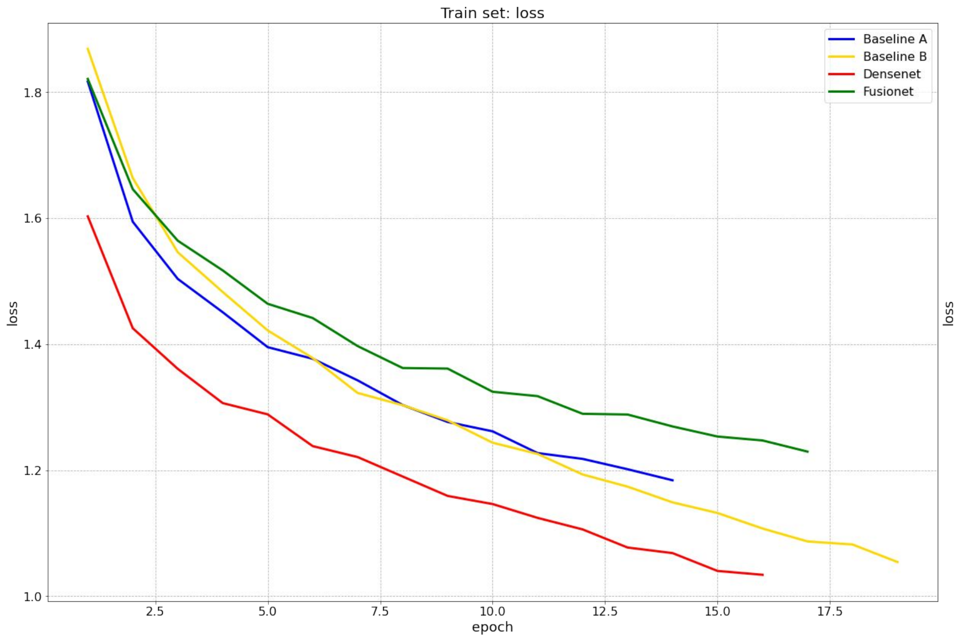
Fusionet:

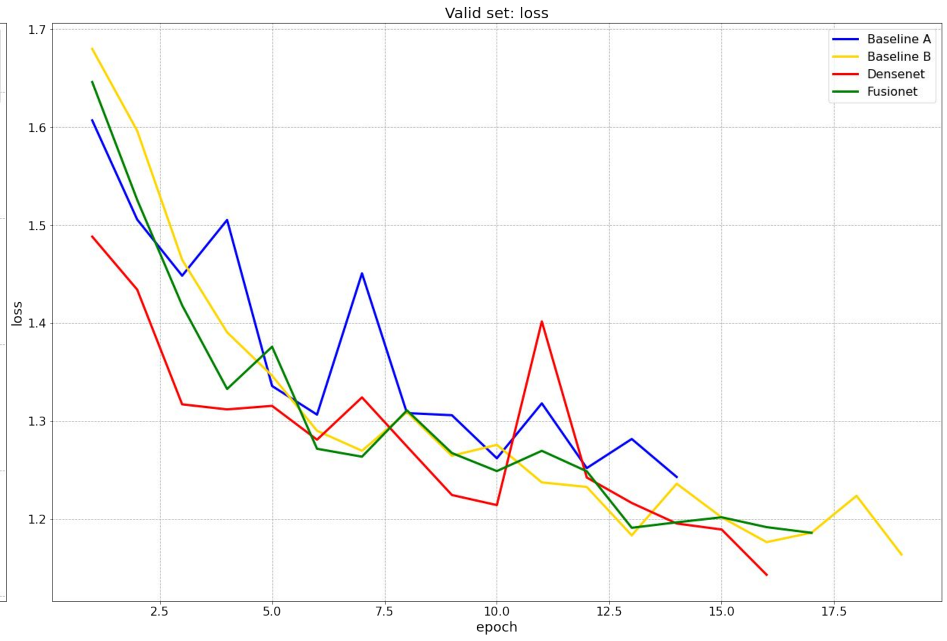


Backup slides:

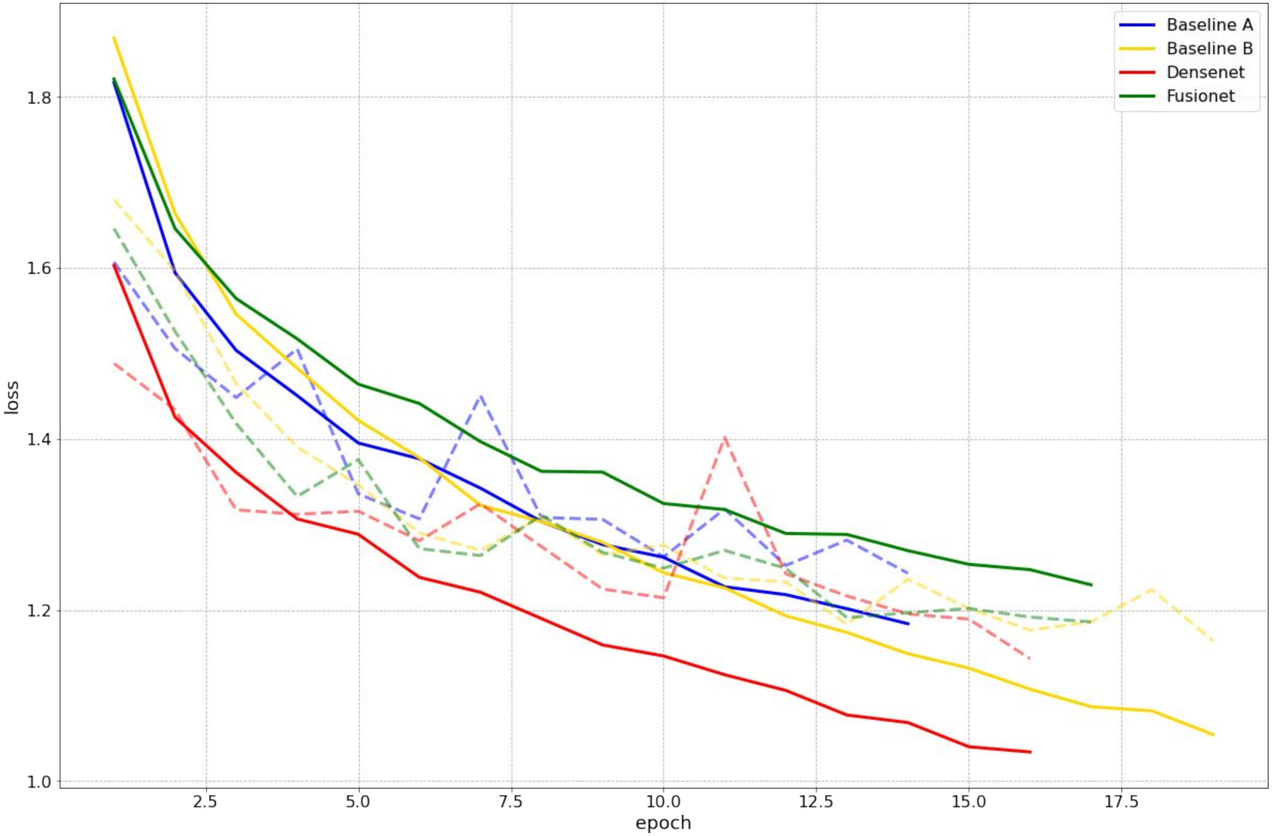
	F1-score	Accuracy	Precision	Recall	# params
Baseline A (1D)	0.448	0.864	0.438	0.457	645,080
Baseline A (2D)	0.558	0.889	0.556	0.559	661,058
Baseline B	0.534	0.883	0.537	0.531	1,252,162
DenseNet	**0.574**	**0.892**	**0.578**	**0.571**	6,955,784
FusionNet	0.553	0.888	0.548	0.557	**7,542,552**

Backup slides:





Backup slides:



Backup slides: