Comparing Black Box and White Box Classification. A Case Study on GTZAN Music Genres Dataset.

Numerical Analysis for Machine Learning Project
Edie Miglio

Academic Year 2021/2022
Mathematical Engineering, Politecnico di Milano

Jacopo Ghirri, Marta Mastropietro

July 27, 2022

Contents

1 Introduction

Supervised classification is one of the core tasks solved by Machine Learning, and as such it can be approached both via a Black Box or a White Box approach.
In this report we intend to show the potentialities and issues of two of the most common instruments, with the ultimate goal of comparing their performances.
As a prime example of Black Box classification we decided to exploit Neural Network theory, as it is one of the most powerful instruments and embodies the core essence of a Black Box approach; indeed with the *Universal Approximation Theorem* there is theoretical foundation for its applicability over any problem, regardless of the specific domain knowledge that one could have, while on the other hand being of difficult interpetability and of stochastic fitting.
For a White Box approach we fit Generalized Linear Models, in particular for the case of binary classification we used Logistic Regression, while we chose Multinomial Logistic Models in the case of multiple classes. Such methods allow for a deterministic fitting (even if approximation methods are usually applied, since no closed form solution exists), having explicit coefficients for each variable, leading to possible new insight on the phenomenon itself, while on the other hand they result much less flexible and the results are highly dependent on the feature selected, hence strong domain knowledge is required.
Such procedures are tested in classifying data belonging to *GTZAN* dataset, containing .wav files divided into ten genres; more information on the data and how we treated it can be found in Section 2.
The analysis is developed in the following steps:

- We build and train Convolutional Neural Networks (*CNNs*) for classifying between genres, in particular we propose a sample of problems aimed at classifying 2, 3, 5 genres and all 10 genres at once. In this section we also explore Transfer Learning as a way to ease the build of such *CNNs* and Data Augmentation as a way to boost performances.

- We test the performances of binary classificators when the training dataset includes outlying observations which have not been correctly labeled.

- We test the stability of the *CNN* built for global classification with respect to different realizations of the Stochastic Gradient Descent procedure and we assert whether Data Augmentation can boost stability as well as performances.

- We build Generalized Linear Models (*GLMs*) for solving the problems previously approached with Black Box methods. In this section we also introduce Quadratic Discriminant Analysis and Fisher's Linear Discriminant Analysis as ways to verify the separability of classes in the feature spaces.

- We test the performances of binary classificators when the training set includes outlying observations; for this purpose we exploit residual-outliers identification techniques.

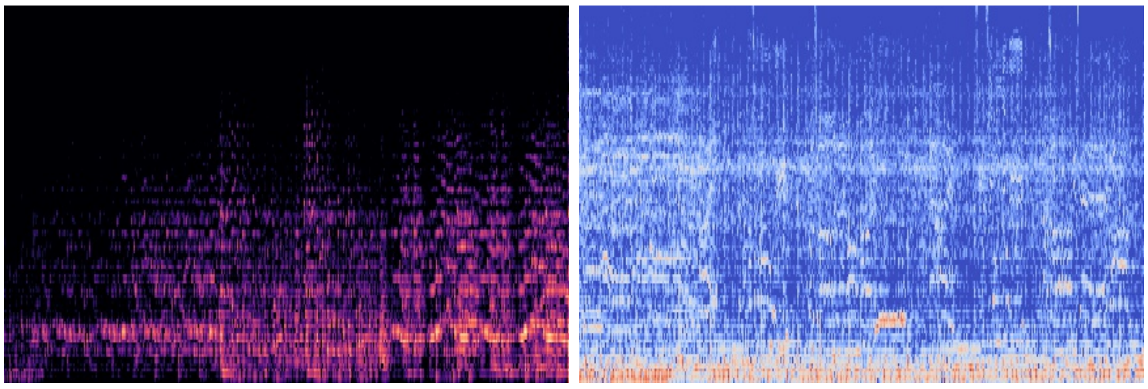

(a) Classical music (b) Metal music

Figure 1: Mel Spectrogram examples

2 Data

The *GTZAN* dataset consists of a collection of 10 genres (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock) with 100 audio files each, all having a length of 30 seconds.
The data extraction and preprocessing section of this analysis was conducted via the use of the *Librosa* package ([5]), available in *Python*.

For the Black Box section we decided to build *CNNs* working on the Mel Spectrograms of the audio files.
An example of what such data representation looks like is reported in Figure 1.
In summary each pixel correspond to a specific pitch in the Mel Scale (y-axis) and time instance (x-axis), and its colour represents the intensity (in decibels) of the signal in that pitch in that time instance, with the colour scale being from dark-blue (down to black) to light-orange (up to white).
This is indeed one of the most general representation of an audio recording, hence we can hope that the minimum amount of information has been lost in the procedure, while at the same time having the datum in an easy-to-manage form.
Even more importantly we hoped that the ability of *CNNs* to tune for the search of local or global features, which can be done by modifying the kernel of each convolution layer, could be useful for the extraction of meaningful and useful features.
Our choice for the Mel Spectrogram in particular, and our understanding of it, has been guided by [4].
For the White Box section we had to extract scalar features ourselves; looking at the features already computed by Librosa package, we settled for the following:

- Zero Crossing Rate: the rate at which the signal changes sign, widely considered to be a key feature to classify percussive sounds.
- Root Mean Square: the square root of the average intensity of the signal, it an energy indicator, roughly representing the overall loudness of the audio recording.

- Mean of the Chromagram: another energy indicator, averaging the intensity of the signal across all pitchess. We hope that such an indicator will be able not only to capture the overall loudness, much like RMS, but also the complexity of the signal, being the mean across pitches.
- Spectral Flatness: a measure to quantify how much noise-like a sound is, as opposed to being tone-like. A high spectral flatness (closer to 1.0) indicates the spectrum is similar to white noise.
- Spectral Contrast: we use three bins (high, mid and low frequencies) over which we compute the mean spectral contrast, hence comparing the energy's top quantile (peak energy) and bottom quantile (valley energy). High contrast values generally correspond to clear, narrow-band signals, while low contrast values correspond to broad-band noise.

This procedure is particularly delicate, indeed White Box models in particular are highly susceptible to the "garbage-in garbage-out" problem. In order to select meaningful features one should have solid domain knowledge, and the lack of it could be really detrimental for the analysis.
This issue however is a main characteristic of any White Box approach, and as such should be included in the comparison. We acknowledge our lack of expertise in this field but proceed anyway.

We evaluate all classification models' performances through the use of four metrics:

- Accuracy: ratio between correctly classified samples and misclassified samples.
- Precision: number of samples correctly identified as being of a given class over the total number of samples assigned to that class, averaged over all groups in the classification problem.
- Recall: number of samples correctly identified as being of a class over the total number of samples truly belonging to that class, averaged over all groups in the classification problem.
- F1 Score: combination of precision and recall, giving an overall metric for the classification of a given class, averaged over all groups in the classification problem.

$$\text{F1 Score} = 2 * \frac{\text{Precision*Recall}}{\text{Precision+Recall}}$$

3 Black Box Approach

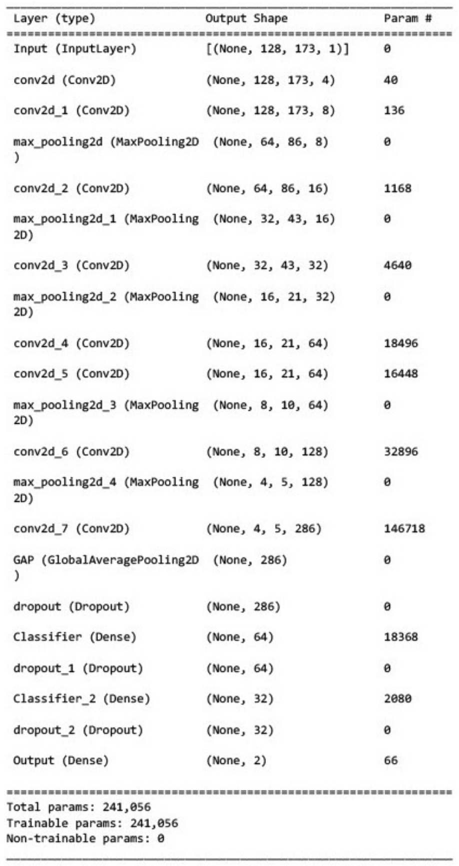
We now explore Black Box techniques and methodologies to try and solve the classification task.

3.1 Convolutional Neural Networks

As stated in Section 1, we intend to solve the classification task by the use of Convolutional Neural Networks, in particular we developed networks according to the following structure:

- Feature extraction section: composed of a stacking of convolution layer and pooling layer. This section ends with a Global Average Pooling (*GAP*) layer.
- Classification section: working on the output of the *GAP* layer, we use a fully connected section to classify the datum. This section ends with a softmax layer and uses dropout as a mean of preventing overfitting.

After many trials, the best template for a model we produced works according to the scheme reported in Figure 2.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input (InputLayer)          [(None, 128, 173, 1)]     0

 conv2d (Conv2D)             (None, 128, 173, 4)       40

 conv2d_1 (Conv2D)           (None, 128, 173, 8)       136

 max_pooling2d (MaxPooling2D  (None, 64, 86, 8)        0
 )

 conv2d_2 (Conv2D)           (None, 64, 86, 16)        1168

 max_pooling2d_1 (MaxPooling  (None, 32, 43, 16)       0
 2D)

 conv2d_3 (Conv2D)           (None, 32, 43, 32)        4640

 max_pooling2d_2 (MaxPooling  (None, 16, 21, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 16, 21, 64)        18496

 conv2d_5 (Conv2D)           (None, 16, 21, 64)        16448

 max_pooling2d_3 (MaxPooling  (None, 8, 10, 64)        0
 2D)

 conv2d_6 (Conv2D)           (None, 8, 10, 128)        32896

 max_pooling2d_4 (MaxPooling  (None, 4, 5, 128)        0
 2D)

 conv2d_7 (Conv2D)           (None, 4, 5, 286)         146718

 GAP (GlobalAveragePooling2D  (None, 286)              0
 )

 dropout (Dropout)           (None, 286)               0

 Classifier (Dense)          (None, 64)                18368

 dropout_1 (Dropout)         (None, 64)                0

 Classifier_2 (Dense)        (None, 32)                2080

 dropout_2 (Dropout)         (None, 32)                0

 Output (Dense)              (None, 2)                 66

=================================================================
Total params: 241,056
Trainable params: 241,056
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Binary Classification Template model summary

Its main features are:

- Stacking of two consecutive convolution layers in the first section, this idea comes from the structure of *VGG16*, as a way of increasing the receptive field in a more controlled manner.
- Maxpooling layers between convolutions, allowing to reduce the size of each slice as we go deeper into the network.
- *GAP* layer, averaging the feature maps obtained in the feature extraction section, obtaining for each one of them a single scalar to be used in the classification section.
- Two fully connected layer and a final fully connected layer with softmax activation, each preceeded by a dropout layer, masking 30% of their inputs during training, forcing redundancy in the structure and helping in preventing overfitting.

Specific tasks however required a specific tuning of the *CNN* structure, on top of modifying the number of layers and neurons, we also tried to use rectangular filters (trying to force time dependence in the features we extracted), with various degrees of success.
In all cases the activation function of choice is *ReLU*, as it is widely used and recognised as one of the most useful in preventing the vanishing gradient effect.

For developing these network we adopted a split of 70% training, 20% validation and 10% test set, maintaining proportionality between different genres.

The loss function of choice is cross entropy, we used early stopping monitoring the validation loss to prevent overfitting during training (patience parameter is set to 20), and an adaptive learning rate, which is halved with a patience of 5 iteration, up to a minimum of 10^{-4}.

Results are reported in the following tables, for binary classification models were heavily based on the template provided, slight modifications usually involved changing the number of neurons in the dense layer to prevent overfitting and underfitting.

Genres	Accuracy	Precision	Recall	F1 Score
Blues, Country	0.9	0.9167	0.9	0.899
Classical, Hip Hop	0.95	0.9545	0.95	0.9499
Classical, Metal	1	1	1	1
Disco, Pop	0.9	0.9167	0.9	0.899
Jazz, Reggae	0.95	0.9545	0.95	0.9499
Metal, Rock	0.9	0.9167	0.9	0.899

Table 1: Binary classifiers: hand-made CNNs

Three genres classification is a slighlty more challenging task, that required tuning of the architecture of the network. Our most successful attempts often involved rectangular filter sizes, to better force pattern exploration along the y axis (time), but here the structure may vary significantly from a classificator to the other.

Genres	Accuracy	Precision	Recall	F1 Score
Classical, Jazz, Rock	0.9333	0.9444	0.9333	0.9346
Blues, Jazz, Reggae	0.9333	0.9333	0.9333	0.9333
Country, Disco, Hip Hop	0.8667	0.8897	0.8667	0.8644

Table 2: 3 genres classifiers: hand-made CNNs

Again, as the task becomes more complex, there is the need to tune each specific model for the specific problem it has to solve.

Genres	Accuracy	Precision	Recall	F1 Score
Country, Disco, Hip Hop, Metal, Rock	0.84	0.8886	0.84	0.8176
Blues, Reggae, Jazz, Country, Pop	0.82	0.8327	0.82	0.8235

Table 3: 5 genres classifiers: hand-made CNNs

Finally, we try and classify all 10 genres simultaneously:

Genres	Accuracy	Precision	Recall	F1 Score
All ten genres	0.77	0.7741	0.77	0.7682

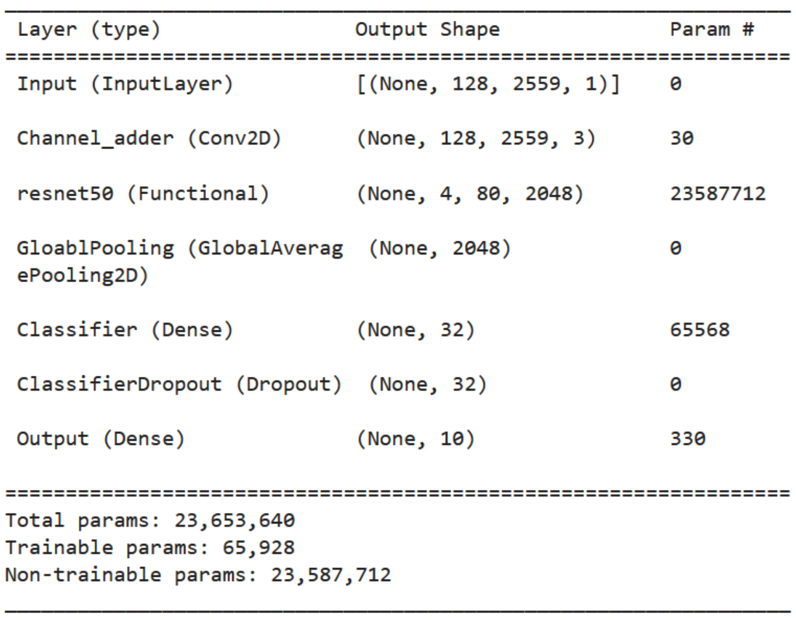
Table 4: Complete classifier: hand-made CNNs

The specific structures of each network are reported in the Github Repository ([2]) under directory *Neural_Networks/Hand-made CNNs.*

3.2 Borrowing power: Transfer Learning

One of the main difficulties of exploiting a *CNN* is designing it, indeed it is not straightforward to properly tune all the different hyperparameters (from number of layers, number of neurons, filter size, strides, order of convolutions and pooling layers...), and without lots of experience trial and error is the only way forward.
For Convolutional Neural Networks in particular, however, such a task can be rendered much more approachable through the exploitation of already designed and pre-trained networks, whose structure has been meticulously developed, and have been trained over massive datasets. Such an idea is the core of Transfer Learning.
In our case we selected *ResNet50*, as it is a network with exceptional performances

that also provides a lot of different feature maps, amongst which we hoped to find some useful ones for our task. The actual network we use for classifying amongst all 10 genres is then of the structure reported in Figure 3.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input (InputLayer)          [(None, 128, 2559, 1)]    0

 Channel_adder (Conv2D)      (None, 128, 2559, 3)      30

 resnet50 (Functional)       (None, 4, 80, 2048)       23587712

 GloablPooling (GlobalAverag  (None, 2048)             0
 ePooling2D)

 Classifier (Dense)          (None, 32)                65568

 ClassifierDropout (Dropout)  (None, 32)               0

 Output (Dense)              (None, 10)                330

=================================================================
Total params: 23,653,640
Trainable params: 65,928
Non-trainable params: 23,587,712
_________________________________________________________________
```

Figure 3: Transfer Learning model summary

We use the feature extraction section of *ResNet50*, using again GAP to connect it to a fully connected section acting as classifier. We needed to add an extra convolution layer at the beginning of the network, since *ResNet50* has been trained on RGB images, we needed to input it with 3 channels and decided to leave the actual channel expansion as a trainable mechanism. Such a layer indeed has no activation function and each channel consist of pure convolutions.
One could also notice that the amount of parameters has drastically increased with respect to our hand-made networks, and training such a network would have been increadibly difficult on our laptops and with the limited amount of data at our disposal, but keeping the *ResNet50* parameters fixed, i.e. non trainable, allows the training to affect only a much smaller amount of free parameters, resulting in a very agile training and much less tuning requirements. In this case indeed we only had one hidden layer in the classification section.
Such a procedure allows not only to alleviate the task of Network Design, but also resulted in better performances compared to our handmade network (Table 4).
Results are reported in Table 5.

3.3 Need for more data: Augmentation

Another main difficulty of Black Box approaches is that, given the particularly high amount of parameters, there is the need for lots of data. This is indeed a problem

	Accuracy	Precision	Recall	F1 Score
TL model	0.8	0.8188	0.8	0.8023
Variation *wrt* hand-made	+0.03	+0.0447	+0.03	+0.0341

Table 5: Complete classifier: Transfer Learning approach summary

given that for each genre we only have 100 songs, to be split amongst training, validation and test set.
Data augmentation techniques consists in ficticiously augmenting the amount of data at our disposal, by either modifying the data itself (in image classification it is standard practice to mirror images, flip them, apply shifts and rotations or even add white noise) or segmenting it (again in image classification one could extract multiple crops of the same image).
The result is a much larger dataset where observations are not truly independent, as many of them are basically replicas, but that nevertheless helps in more effectively training the network.
Such a procedure is also regularly applied to train the network not to be susceptible to specific transformations of the input datum, again in image classification one could for instance want the network not to be influenced by different lighting conditions and so would fictitiously change the brightness of the images during training.
In our case we decided to apply such a technique on the *CNN* designed to classify all 10 genres, as it is the one with the worst performance. We then split each audio file into three consecutive section, and treat each section independently, extracting its own Mel Spectrogram and feeding it to the network, effectively increasing the dimension of the dataset by 3x.
Particular care had to be taken into making sure that audio segments coming from the same original song ended up in the same dataset during the split, as to keep the estimates of validation and test error as unbiased as possible.
The results (Table 6) where exceptionally better, achieving an almost perfect classification at test time.

	Accuracy	Precision	Recall	F1 Score
DA model	0.99	0.9883	0.9881	0.988
Variation *wrt* non augmented	+0.22	+0.2142	+0.2181	+0.2198

Table 6: Complete classifier: Data Augmentation approach summary

It should be noted that such a split of the original data does not impact the applicability of the network, as one could either just use it on shorter audio files, or apply the network over all three segments and then either use majority voting or average the results of the softmax to determine the overall class, the latter option is also a quite common practice, as it consists in an augmentation of the test set.

3.4 Stability results over outliers

In this section we try and assess how much the presence of outliers in the training set impacts the classification power of a Neural Network. The way we test for this is, for each binary classifier we designed, we design a dataset with a strong presence of outliers, use it for training and test its performance.
A pseudocode of this procedure is reported in Algorithm 1.

Algorithm 1: Adding outlier to Black Box binary classification between genres g_1, g_2

```
Input: 10 datasets of different genres $G$
Output: A dataset for $g_1, g_2$, containing outliers
Initialize Training set, Validation set and Test set as empty
for $i \in \{1, 2\}$ do
    for $j \in \{1, ..., 100\}$ do
        Extract song $s_j$ from genre $g_i$
        Set label($s_j$) to $i$
    end
    Shuffle $\{s_j\}$ randomly, obtaining $\{s_{j^\star}\}$
    Assign $j^\star \in \{1, ..., 70\}$ to Training set, $j^\star \in \{71, ..., 90\}$ to Validation set,
      $j^\star \in \{91, ..., 100\}$ to Test set
    for $g_{out} \in G \setminus \{g_1, g_2\}$ do
        for $k \in \{1 + 6 * i, 6 + 6 * i\}$ do
            Extract song $s_k$ from genre $g_{out}$
            Set label($s_k$) to $i$
        end
        Assign $k \in \{1 + 6 * i, 5 + 6 * i\}$ to Training set, $k = 6 + 6 * i$ to
          Validation set
    end
end
return Training set, Validation set, Test set
```

In summary the composition of the dataset is:

- Training set: 220 songs, 110 songs labeled as each genre. 70 of such 110 correctly labeled, 40 are outliers coming from other genres (5 from each outlier genre).
- Validation set: 56 songs, 28 songs labeled as each genre. 20 of such 28 correctly labeled, 8 are outliers coming from other genres (1 each).
- Test set: 20 songs, 10 songs labeled as each genre, all correctly labeled.

As one can notice the outlier presence is important ($\sim 36\%$ in Training, $\sim 29\%$ in Validation), but no label swapping between the two genres to be classified has been applied, hence sticking to the definition of outlier as a piece of data originated from another distribution.
We stress the fact that the test set over which we need to evaluate performances has

been kept intact during this procedure, hence providing unbiased estimates of the performances for the actual classification.

Genres	Accuracy	Precision	Recall	F1 Score
Blues, Country	0.85	0.8535	0.85	0.8496
Variation *wrt* clean data	-0.05	-0.0632	-0.05	-0.0494
Classical, Hip Hop	0.95	0.9545	0.95	0.9499
Variation *wrt* clean data	0	0	0	0
Classical, Metal	0.9	0.9	0.9	0.9
Variation *wrt* clean data	-0.1	-0.1	-0.1	-0.1
Disco, Pop	0.85	0.8846	0.85	0.8465
Variation *wrt* clean data	-0.05	-0.0321	-0.05	-0.0525
Jazz, Reggae	0.95	0.9545	0.95	0.9499
Variation *wrt* clean data	0	0	0	0
Metal, Rock	0.8	0.8571	0.8	0.7917
Variation *wrt* clean data	-0.1	-0.0596	-0.1	-0.1073

Table 7: Binary classifiers: hand-made CNNs with dirty dataset

As one could see from Table 7 the results were surprisingly good and solid, despite the massive amount of outliers.
This leads as to believe that, as long as no label swapping over the classes occurs, the Neural Networks are still able to effectively determine the differences between the two classes and apply them to solve the classification problem.
This could be the case since Neural Networks are able to find arbitrary separation boundaries in the feature space, hence if classes are fairly separable, outliers would have a very limited impact in the region where the classification is most sensible.
It should be noted that indeed the classifiers between Classical music and Hip Hop, as well as Jazz and Reggae, have not been affected at all, and indeed it is reasonable to assume that they are amongst the most easily separable genres; while the worst drop in performance is experienced in classifying Metal and Rock, where the separation might be blurry and outlier could have a bigger impact.

3.5 Dependency on randomness

The training of a Neural Network is usually performed through the use of Stochastic Gradient Descent or SDG-inspired methods (in our case we used Adam's optimization, [3], which also includes a momentum component), which consequently leads to randomness in both fit and performances.
We now aim at evaluating the dependence of such randomness during training, and we do this by training the classifier over all 10 genres a total of 10 times, collecting

the performances of the 10 different models, whose summary is reported in Table 8

Metric	Mean	Standard deviation	Range	Minimum value
Accuracy	0.734	0.036	0.130	0.680
Precision	0.744	0.044	0.153	0.676
Recall	0.734	0.036	0.129	0.680
F1 Score	0.727	0.039	0.145	0.664

Table 8: 10 genres classifier, results over 10 fittings

Such results have two main implications:

- Model variance plays a huge role in Neural Networks, this effect has probably been amplified by the small amount of data at our disposal.
- Big performances on the proposed models may have been inflated by a selection bias. We indeed used a training-validation-test split, using validation data to perform early stopping and test data to estimate the error committed by our model, such an estimate is unbiased, but due to the huge variance in said estimation we may have just experienced a lucky estimate on classifiers that may underperform on larger test sets.

It is our argument that data augmentation helps not only in boosting performance, but in boosting model stability as well. Indeed the bias-variance decomposition formula is:

$$MSE = bias^2 + Variance$$

and Data augmentation can hopefully operate on both terms of the equation, hence using it might not be an actual trade off as well as an overall improvement.
For this reason we perform the same experiment displayed in Table 8, but using models that exploit the data augmentation proposed in Section 3.3. Results are reported in Table 9

Metric	Mean	Standard deviation	Range	Minimum value
Accuracy	0.982	0.013	0.050	0.946
Precision	0.983	0.014	0.051	0.944
Recall	0.981	0.012	0.047	0.949
F1 Score	0.981	0.013	0.049	0.946

Table 9: 10 genres classifier with Data Augmentation, results over 10 fittings

As expected, not only the average performances increase, but dispersion metrics decrease as well.

4 White Box Approach

We now try to tackle the same challenge previously approached with *CNNs* by using White Box approaches.

4.1 General indicators of the quality of features: QDA and Fisher-LDA

Arguably the most critical aspect of any White Box procedure is feature selection. Indeed a classification procedure can work at most as well as the embedding of data allows it to do.
For this reason we need a procedure to, at least qualitatively, assess the separability of different classes in the feature space defined in Section 2.
We propose two approaches:

- Quadratic Discriminant Analysis: Such a procedure is based on the assumption that data of each class are distributed according to a Multivariate Gaussian.
  We use the training set to learn each distribution parameters (mean vector and variance matrix), and assign a new observation to the class which correspond to the maximum likelihood.
  This leads to separation boundaries which are piece-wise quadratic, hence the name.

- Fisher's Linear Discriminant Analysis: Such a procedure is based on the assumption that classes have a similar covariance structure.
  We use the training set to learn the center (mean vector) of each class, and assign a new observation to the class whose center is closer; if each class were additionally Gaussianly distributed, this procedure coincides with a maximum likelihood approach, under homoscedasticity assumptions.
  This leads to a piece-wise linear separation boundary, hence the name.

The assumptions were often not satisfied, hence we do not propose them as actual classifiers; nevertheless we use them to gain insights into the quality of the feature space we selected.
It should moreover be noted that such procedures just select a separation boundary, but do not provide an explicit model or any other instruments that would help get insights into the phenomenon.

4.2 Generalized Linear Models

The White Box classificators we build and propose are based on Generalized Linear Models (*GLMs*); in particular Logistic Regression for binary classification problems and Multinomial Logistic Regression when we consider more than two classes.
Both methods are fitted via minimizing cross-entropy, that under Gaussianity assumptions of the Logit link-function correspond to a maximum likelihood approach.
Both aim at finding linear relationships between features and resulting classification

and even though this could be an oversimplification of the problem, leading to poorer performances, one could potentially extract useful information out of the parameters of the learned models.
We fit a GLM over each task we designed for the Neural Networks, and we perform feature selection by iteratively removing the statistically least significant features (Backward Feature Selection by using coeffiecient's *p-values*), while keeping under control the McFadden Pseudo-R^2, for a quick performance analysis.
Each model ended up with its own particular set of meaningful variables, as a result of feature selection. For the sake of simplicity we just report the performance metrics of the test set of the final model of each problem, togheter with the performances of LDA and QDA. The full model selection procedure, togheter with training set evaluation metrics of each model presented can be found in the Github Repository ([2]) under directory *White_box*.
Binary classification results are reported in Table 10.

Problem	Accuracy	Precision	Recall	F1 Score
Blues, Country				
QDA	0.675	0.803	0.675	0.6366
Fisher's LDA	0.4	0.2222	0.4	NaN
Logistic Regression	0.475	0.4641	0.475	0.432
Classical, Hip Hop				
QDA	0.975	0.9762	0.975	0.9749
Fisher's LDA	0.975	0.9762	0.975	0.9749
Logistic Regression	0.95	0.9545	0.95	0.9499
Classical, Metal				
QDA	1	1	1	1
Fisher's LDA	1	1	1	1
Logistic Regression	0.975	0.9762	0.975	0.9749
Disco, Pop				
QDA	1	1	1	1
Fisher's LDA	0.975	0.9761	0.975	0.9749
Logistic Regression	0.95	0.9545	0.95	0.9498
Jazz, Reggae				
QDA	0.675	0.6754	0.675	0.6747
Fisher's LDA	0.775	0.7756	0.775	0.7748
Logistic Regression	0.75	0.7525	0.75	0.7493
Metal, Rock				
QDA	0.85	0.8535	0.85	0.8496
Fisher's LDA	0.725	0.7255	0.725	0.7248
Logistic Regression	0.775	0.7756	0.775	0.7748

Table 10: Binary classifiers: Logistic Regression

For three genre classification results are shown on Table 11.

Problem	Accuracy	Precision	Recall	F1 Score
Classical, Jazz, Rock				
QDA	0.7166	0.7475	0.7166	0.6674
Fisher's LDA	0.7166	0.7469	0.7166	0.6672
Multinomial Regression	0.6833	0.7246	0.6833	0.6397
Blues, Jazz, Reggae				
QDA	0.75	0.7884	0.75	0.7563
Fisher's LDA	0.7	0.7368	0.7	0.7051
Multinomial Regression	0.7	0.7244	0.7	0.7024
Country, Disco, Hip Hop				
QDA	0.7333	0.8055	0.7333	0.7377
Fisher's LDA	0.7166	0.7376	0.7166	0.7193
Multinomial Regression	0.75	0.75	0.75	0.7464

Table 11: 3 genres classifiers: Multinomial Logistic Regression

Results for five genres classification are reported in Table 12.

Problem	Accuracy	Precision	Recall	F1 Score
Country, Disco, Hip Hop, Metal, Rock				
QDA	0.43	0.4895	0.43	0.3955
Fisher's LDA	0.41	0.4493	0.41	0.3994
Multinomial Regression	0.44	0.4948	0.44	0.4403
Blues, Reggae, Jazz, Country, Pop				
QDA	0.26	0.2437	0.26	NaN
Fisher's LDA	0.26	0.2437	0.26	NaN
Multinomial Regression	0.39	0.3746	0.39	NaN

Table 12: 5 genres classifiers: Multinomial Logistic Regression

Finally, the performances of the model for the classification of all ten genres are reported in Table 13.
It is easy to see how performances can be only as good as the separation in the feature space is clear, and how this easily becomes an issue as the number of genres to be classified increases. The presence of NaN in the F1 Score indicates that at least a whole genre has been completely misclassified, having both precision and recall equal to zero.
Even though the performances are as expected worse than those of Neural Networks, there are multiple instances where they have outstandingly good performances. It should be noted that in such scenarios QDA and LDA both retain exceptional per-

Technique	Accuracy	Precision	Recall	F1 Score
QDA	0.37	0.3168	0.37	NaN
Fisher's LDA	0.335	0.2886	0.335	NaN
Multinomial Regression	0.345	0.3307	0.345	NaN

Table 13: Complete classifiers: Multinomial Logistic Regression

formances, indicating a clear separation of the classes in the feature space.

Criticisms could be raised from the fact that this performance analysis has been heavily influenced by our ability to determine good features for this problem, and such criticisms would be correct.
Nevertheless since feature selection is a critical task of any White Box procedure, its difficulty should be accounted for and a fair comparison should showcase how lack of field knowledge could render a White Box approach nearly infeasible.

4.3 Insights on the phenomenon

As stated in the introduction of White Box models, the major advantage of such techniques is the ability to gain insights on the phenomenon from the trained model. This is made particularly easy by the linear structure of the proposed *GLMs*, indeed by looking at each variable parameter it is possible to assess their impact in the classification, and the higher the parameter the more determinant the impact.
In doing this interpretative considerations one should however be mindful of possible differences in scale of the input variables, as they have not been standardized.
As the goal of this project is not to investigate differences in music genres, but to showcase the pros and cons of different Machine Learning techniques, we illustrate the interpretation of a single model: the classifier between classical and metal songs.
After feature selection, the model formula is as follows:

$$\mathbb{P}(s_i \in g_{metal} | features(s_i), s_i \in \{g_{metal}, g_{classical}\}) \cong \sigma(19.007 + \\ 24.5849 * RMS_{energy}(s_i) - 2.366 * MF_{contrast} + 0.3791 * LF_{contrast})$$

With:

- RMS_{energy} being the Root Mean Square of song s_i, the higher the value the more energetic the song is overall.
- $MF_{contrast}$ and $LF_{contrast}$ being the mean spectral contrasts of mid and low frequencies, respectively. The higher the values the clearer the sound is on such frequencies, as opposed of being more noise-like.
- $\sigma() : \mathbb{R} \to (0, 1)$ being the sigmoid logistic function:

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

The interpretation of this model is straightforward, we can expect a metal track to be on average more energetic than a classical one, while having a more noise-like behaviour on mid frequencies and being slightly clearer and more defined on low frequencies.
All other features do not have a significant impact in the classification, albeit due to collinearities between features it is possible to obtain alternative models based on other features if one were to follow a different feature selection procedure.

4.4 Outliers: issues and remedies

We now try and evaluate the performance of binary classification models (Logistic Regression) in the scenario of contaminated training set, as done for Neural Networks. Since such models did not need a validation set, we propose a different split that allows to better estimate model performances while not penalizing training.
The procedure is illustrated in Algorithm 2.

Algorithm 2: Adding outlier to White Box binary classification between genres g_1, g_2

```
Input: 10 datasets of different genres $G$
Output: A dataset for $g_1, g_2$, containing outliers
Initialize Training set and Test set as empty
for $i \in \{1, 2\}$ do
    for $j \in \{1, ..., 100\}$ do
        Extract song features $s_j$ from genre $g_i$
        Set label($s_j$) to $i$
    end
    Assign $j \in \{1, ..., 80\}$ to Training set, $j \in \{81, ..., 100\}$ to Test set
    for $g_{out} \in G \setminus \{g_1, g_2\}$ do
        for $k \in \{1 + 5 * (i - 1), 5 + 5 * (i - 1)\}$ do
            Extract song $s_k$ from genre $g_{out}$
            Set label($s_k$) to $i$
            Assign $k$ to Training set
        end
    end
end
return Training set, Test set
```

Following this procedure we reach a Training Set with a cardinality of 240, with a total of 80 misclassified songs, coming from all 8 other genres.
When Logistic Regression models are trained using such datasets results may vary; just like in the Neural Networks scenario some models still retain good performances on a test set, while others end up being severely affected.
A great advantage of directly fitting a *GLM*, however, is the straightforward procedure to identify outlying observations. Such observations indeed are not just the training data which are misclassified by the model, but those for which we have unusually large residuals. We use Pearson's residual, as it is standard practice in dealing with logistic regression; further information on their use can be found in [6].

Since the estimation procedure is not robust with respect to outlier, effects like residual masking are not uncommon, meaning that the presence of some outliers skews the estimates so much that other outlying observations are not detectable as such; for this reason we iterate an outlier detection step and a fitting on the clean dataset, until we are satisfied with the results and no worrying (large) outliers are detected.
Test set metrics for the final models produced are reported in Table 14.

Problem	Accuracy	Precision	Recall	F1 Score
Blues, Country				
Logistic Regression	0.425	0.3285	0.425	0.3309
Variation *wrt* clean data	-0.05	-0.1356	-0.05	-0.1011
Classical, Hip Hop				
Logistic Regression	0.9750	0.9761	0.975	0.9749
Variation *wrt* clean data	+0.025	+0.0205	+0.025	+0.025
Classical, Metal				
Logistic Regression	0.9750	0.9761	0.975	0.9749
Variation *wrt* clean data	0	-0.0001	0	0
Disco, Pop				
Logistic Regression	0.95	0.9545	0.95	0.9498
Variation *wrt* clean data	0	0	0	0
Jazz, Reggae				
Logistic Regression	0.675	0.679	0.675	0.6731
Variation *wrt* clean data	-0.075	-0.0775	-0.075	-0.0765
Metal, Rock				
Logistic Regression	0.575	0.5854	0.575	0.5615
Variation *wrt* clean data	-0.2	-0.2002	-0.2	-0.2133

Table 14: Binary classifiers: Logistic Regression with dirty dataset

As one can see, this procedure allows to train models that reach performances which are often comparable to, if not even exceeding, those trained on clean datasets. It is indeed standard practice to treat outliers when training *GLMs*, even in scenarios where one would be sure on the quality of data.
As in the Black Box case, the problem suffering the most is the one which one could imagine be the least separable: rock vs metal comparison.

5 Comparison and conclusions

Comparing the results obtained in the two steps of our work, one can see what the main advantages and disadvantages of each procedure is.
We indeed showed the flexibility of Black Box approaches, and how they shift the modeling burden from feature selection to hyperparameters tuning, allowing to effectively solve problem without needing a strongly informed backgroung. The flexibility of Neural Networks in particular allows a data scientist to work on more complex, but complete, representations of the datum; representations that would not be manageable in a White Box setting.
On the other hand White Box approaches allow for easily interpretable and straightforward models, at the expense of a much more challenging pre-processing of the data, under the shape of feature extraction. Albeit it is not as crucial and challenging as in Neural Networks, White Box models too might require hyperparameters tuning, under the form of penalization factors (like in Ridge Regression) or even feature selection (e.g. number of degrees in polynomial Regression).
Surprisingly enough the effect of outliers is similar between the two approaches, however it should be stressed that we explored the case of true outliers, without data mislabeling between the two classes of interest. It is possible that a more aggressive approach would show significant differences.
Significant differences can be found in the way that the two approaches can overcome their specific challenges, a clear example is Transfer Learning, with which it is possible to efficiently solve any *CNN*-based task without the need of excessive tuning, significantly easing the model selection.

It is our argument that the two approaches are not as parallel as it is commonly thought, there are significant benefits that one can obtain by exploiting techniques generally adopted by the other.
Data Augmentation techniques are rarely used in a Generalized Linear Model setting, as they increase coodependence between data and can sometimes lead to the violation of assumption of said model, nevertheless they can be incredibly useful in dealing with complex problems with few amounts of data. Similar principles are used in the shape of bootstrap approaches, but to the best of our knowledge proper augmentations techniques are far from common in White Box settings.
A similar argument can be made for outlier detection, which is standard practice for *GLMs*, but is still under study on Neural Networks. The most research to the best of our knowledge is done via Autoencoders, hence building Networks specifically for this task, although research is being made on how to embed this practice into a generic network([1]). As it was shown in Section 4.4, such techniques can have a huge benefit in the quality of the fitted model, and should be further investigated in a Black Box setting too.

References

[1] Gleb Beliakov, Andrei Kelarev, and John Yearwood. Derivative-free optimization and neural networks for robust regression. *Optimization*, 61(12):1467–1490, dec 2012.

[2] Jacopo Ghirri and Marta Mastropietro. Naml_project, 2022. [GitHub repository, https://github.com/JacopoGhirri/NAML_project].

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations, San Diego*, 2015. [conference paper, arXiv:1412.6980].

[4] Roberts Leland. Understanding the mel spectrogram, March 2020. [Online; posted 6-March-2020, https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53].

[5] Brian McFee, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, Ayoub Malek, Dana, Kyungyun Lee, Oriol Nieto, Dan Ellis, Jack Mason, Eric Battenberg, Scott Seyfarth, Ryuichi Yamamoto, viktorandreevichmorozov, Keunwoo Choi, Josh Moore, Rachel Bittner, Shunsuke Hidaka, Ziyao Wei, nullmightybofo, Adam Weiss, Darío Hereñú, Fabian-Robert Stöter, Lorenz Nickel, Pius Friesch, Matt Vollrath, and Taewoon Kim. librosa/librosa: 0.9.2, June 2022.

[6] Zhongheng Zhang. Residuals and regression diagnostics: focusing on logistic regression. *Annals of Translational Medicine*, 4(10):195, May 2016.