Learning from noisy labels in an audio-based deep learning application.

G027 (s1456085, s1884786, s1870525)

Abstract

Nowadays, internet can provide us with considerable amount of data. However, due to the prominently unknown sources of this data, it usually needs manual verification before being processed by a machine learning algorithm. This constraint results in a tremendous use of time and resources. In previous work, it was proven that a large amount of unverified labels in a dataset can be beneficial to tasks such as classification. Our main contribution is a further analysis of the impact of such unverified labels on an original sound dataset called FSDnoisy18k comprising of around 18,000 sounds gathered from Freesound.

1. Introduction

With the continuously growing amount of data available online, an infinite number of possibilities arose in the field of machine learning. However, if the recurrent problem is not the acquisition of data anymore, it is certainly defined by the lack of consistent and reliable labeling processes.

Throughout the project, we decided to focus on a specific task where unverified labels are frequent: Automatic tagging of acoustic events. Automatic tagging of acoustic events refers to the task of identifying sounds from an audio file. To do so, the research community takes inspiration in the human auditory system and usually couples perceptual parameters of the signal with machine learning techniques.

The motivation to study acoustic detection goes beyond just imitating human cognition abilities. Some examples of possible applications can be found in many fields. For instance, environmental noise monitoring (Maijala et al., 2018) can greatly help understanding the impact of human-made elements in natural environments (e.g. planes and highways). Another possible application of acoustic detection is related to surveillance monitoring of roads. (Li et al., 2018) shows a sound detection system that outperforms a video-based monitoring system in speed under certain conditions.

Most of the research conducted in the field of machine learning is carried out using a carefully labelled database (supervised learning applications). Nevertheless, this is a limitation when it comes to data coming from different user sources as it is not always possible to ensure the accuracy of the labels. With the aim of understanding the impact of unreliable labels, the DCASE Challenge was proposed in 2018 (Fonseca et al., 2018). The database provided by the authors consisted of a set of audio files, containing 60% of unverified labels. The different participants demonstrated the possibility for convolutional neural network (short: **CNN**) structures to classify unseen audio files while having a training set containing around half of unreliable data labels.

In 2019, *Fonseca et. al.* extended further their findings and showed that it is feasible to improve the performance accuracy of a simple classifier by training on a vast majority of noisy labels (~90%). To verify their experiments, they presented FSDnoisy18k, a dataset comprising of sound files coming from different user sources on internet. The dataset is defined by about 90% of noisy labels alongside the clean labels and will be used for our experiments here. More information about the dataset is given in section 2. The experiments carried out in their publication were focused on CNNs with modified loss functions. Although it proved to be fairly efficient, they do not investigate further the reasons why noisy labels affect the overall performance of the classifier.

This project is based on their original paper (Fonseca et al., 2019), and will be divided as follows: After presenting some brief overview of related works, we introduce a full description of the dataset. Then, the methodology section will familiarize the reader with the main concepts on which we based our experiments. Following, a full description of each experiment will be clearly shown with their corresponding results. Finally, a conclusion and further work section will bring formal answers summarizing the questions formulated in the experimental section.

2. Related work

Sometimes, assigning a ground truth label to a signal is impossible. Usually, labels are assigned by experts. However, even experts can have different opinion regarding the classification of an instance. In tasks such as image recognition of digits, there are little to no uncertainty, but when the signals are much more complex, discrepancies arise. To a certain extend, we can think of all labels are somewhat noisy because each person might perceive signals in a different way. (Guan et al., 2018) address this issue by training systems independently for each expert that has labelled part of a database (doctors labelling a Diabetic Retinopathy database). They stated that finding a weight for each expert can improve classification accuracy. A fundamental work in noisy labels, (Reed et al., 2015), manages noise in a probabilistic modelling approach. In the same line of work, in the image classification domain as well, (Xiao et al., 2015) shapes the relationship between images in a

probabilistic graphical model approach. They observed that previous semi-supervised approaches where noisy labels are discarded are not viable for large-scale dataset (as the complexity of the system increases). (Xiao et al., 2015) proposed an improvement of the work carried out by (Reed et al., 2015). They observe that the assumption that noisy labels are conditionally independent of the input images given the clean labels does not hold in reality. A more general work worth mentioning, (Nettleton et al., 2010), estimates the effect of noisy labels in different machine learning algorithms. (Zhang et al., 2017) show the property of deep neural networks to memorize completely random inputs. This suggests that a not too big amount of incorrect labels might not be dangerous for a Deep Learning architecture. This hypothesis is supported by the previous publications mentioned.

In the audio processing field, the research is mainly concentrated on the DCASE Challenge (Fonseca et al., 2018). However, some exciting research is carried out in many different applications. For instance, in (Salamon & Bello, 2017), they assessed different data augmentation techniques over a sound environmental database. Another relevant piece of work is AudioSet (Gemmeke et al., 2017), made by Google and quite similar to DCASE but on a larger scale.

3. Data set and task

3.1. Overview

Throughout this project, we will use the FSDnoisy18k dataset. This data was collected from Freesound [1], a free webpage promoting the sharing of sounds. Overall, it includes 18, 532 audio files summing up to more than 42 hours. Each audio file is assigned to one and only one of the 20 existing classes (see figure 1). As the data was collected from different sources, it does not come all in the same format. Although they all are wav files, some are 300 ms long and some other ones are up to 30 seconds long.

In order to fully understand the remaining part of the report, some definitions are needed:

- **Clean labels** refer to audio files that were labelled by unknown users on internet and manually verified and approved by the authors of the FSDnoisy18k database.
- **Noisy labels** do **not** refer to audio files containing physical noise but rather data that was not manually inspected by the authors of the database. This means that the labels might be *correct*, *incorrect* or *partially correct* (see section 3.3 for further information).

3.2. Dataset organisation

The FSDnoisy18k dataset is divided into training and testing sets. The training set comprises of 17, 585 files (slightly more than 41 hours) and represent about 95% of the data number (though 96.7% in terms of duration). As we can see in figure 1, the classes in the training set are unequally distributed and mainly include noisy labels. In total, the training set has a clean/noisy label ratio of 0.1.
The testing set comprises of the remaining 947 clips (about 5% - 1.4 hours) and does not contain any noisy labels.

[1] https://freesound.org/

Label noise type	Amount	Label noise type	Amount
Overall	60%	Incorrect/I-V	6%
Incorrect/O-O-V	38%	Incomplete/I-V	5%
Incomplete/O-O-V	10%	Ambiguous labels	1%

Table 1. Distribution of noise in 15% of non-manually verified labels. Source: (Fonseca et al., 2019).

However, the original dataset does not include any validation set. To allow formal comparison, we decided to create a validation set, which will be used to evaluate all the experiments. This leaves an unseen test set to report the final results. To create the validation set, *Fonseca et al.* take 15% of the training set to create a validation set. This results in a validation set that has both clean and noisy data. The problem of this procedure is that the validation set is not representative of the test set because the former includes noisy labels. So, we decided to create a validation set that consists only of clean labels. To do so, we first created a validation set by taking 15% of the clean training set. This results in a training set of size 17,310, a validation set of size 275 and a testing set of size 947. Nevertheless, we realised that this was not enough to truly assess the quality of our models. Therefore, we decided to concatenate the 275 clean audio files taken from the training set with the testing set, and then divide the resulting set in half. This has been decided considering that the validation set **must** be similar to the testing set in terms of clean labels. The resulting divisions of the dataset are now the following: 17,310 training audios (including 15,813 noisy labels and 1,497 clean labels), 611 validation audios (all clean) and 611 testing audios (all clean).

3.3. Label analysis

The original paper performed a further analysis of the noisy labels present in the dataset. They took a random 15% of non-verified data (noisy labels) and manually examined the audio files and their corresponding label. They discovered the different types of labelling that are present:

As shown in table 1, the distinction is made between labels based on their correctness and whether they belong to the vocabulary list (shown in figure 1). Out of the 15% of noisy data analysed, about 40% were correctly labelled and the remaining 60% of the data was encoded with some incorrect or incomplete labels. Incorrect labels can be either In-Vocabulary (e.g. *"Rain"* when it should be *"Piano"*) or Out-Of-Vocabulary (e.g. *"Rain"* when it should be *"Laughter"*, which is not in the vocabulary considered in this task). Incomplete labels can be either In-Vocabulary (*"Rain"* when

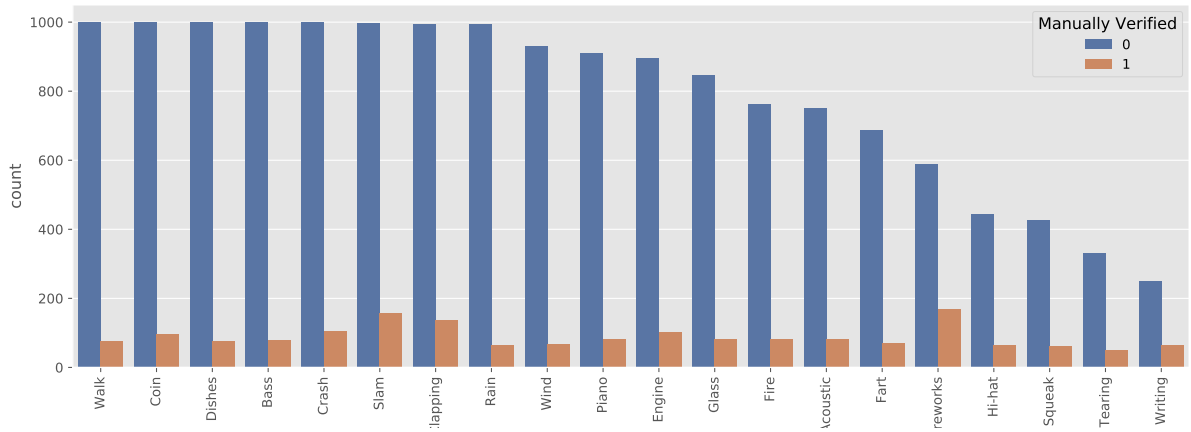

Figure 1. Distribution of the 20 classes in the training set. The blue bins represent noisy data, and the orange bins the manuall verified data.

it should be *"Rain" and "Piano"*, which was not encoded by the model since it can only accept one target class) or Out-Of-Vocabulary (*"Engine"* when it should be *"Car engine"*).

These distinctions will be useful later on, when experimenting on data-based model improvements.

3.4. Pre-processing

As mentioned before, the data files come in wav audio format and in various lengths. In order to be processed by a machine learning algorithm, the clips need to be modified. The database was recorded originally in 44.1 kHz, which is the standard sampling rate for audio recording. However, in order to reduce the amount of data stored, we have applied downsampling to 32kHz and normalized the amplitude of each on each one[2]. For the rest of the pre-processing, the decision to apply all the following techniques was based on the baseline presented in the original paper. Figure 3 presents a chart flow of the pipeline, and 2 the parameters used for the pre-processing.

The next step of the pre-processing pipeline is the standardization of the length of the audio files to two seconds (115 frames). The reason behind this is the need for inputs of the same size in order to be used by a CNN. We have chosen a length of 2 seconds, as in *Fonseca et. al.*. For audios of length less than 2 seconds, we repeated them until the desired length was reached. For audio files of longer length, we discarded the information after 2 seconds.

To extract meaningful features, we parameterized the raw wav files using log-mel filterbank features (Davis & Mermelstein, 1980). A log-mel filterbank (FBANK) is an auditory system-based processing which aims to smooth the Fourier spectrum. FBANK is based on short-time processing of the signal, which assumes stationarity over a window frame to apply the Discrete Fourier Transform (short: **DFT**) (Rabiner & Schafer, 2011). We will use the power spectrum, discarding the phase information of the DFT, as studies have shown its irrelevance.

Then, the spectrum is converted to a mel-scale, which has a logaritmic behaviour. Mel-scale is based on the human-being resolution to differentiate frequencies. Next, we apply a triangular filterbank which emulates the behaviour of the basilar membrane in the auditory system (Robles & Ruggero, 2001). We finalize by compressing the dynamic range. This is done by applying a logarithmic function to the energy-based features (FBANKs).

FBANK features have been successfully applied to several deep learning based speech recognition tasks and has been widely adopted as baseline for audio tasks (see (Hinton et al., 2012) and (Yu et al., 2018)). At the end of this transformation, each file is represented as a matrix of shape 96x115. This matrix can be visualised in figure 2.

A follow-up feature from FBANK is the well-known MFCC, which stands for Mel Frequency Cepstral Coefficients (Davis & Mermelstein, 1980). The difference with FBANK is that we perform a Discrete Fourier Transform (DCT) to the FBANK. This is done because of the property of the DCT to decorrelate information. To test the efficiency of such technique, we experimented with MFCC as well as with FBANK. Nonetheless, the results obtained with MFCC were not as promising as with FBANK. The explanation arises from the uncorrelated property of MFCC, which is not ideal for neural network architectures, where it is preferable to let the network learn or extract meaningful representations by itself (Mohamed et al., 2012).

4. Methodology

4.1. Convolutional neural networks

CNNs are a type of neural networks characterized by their ability to encompass contextual information. From their inputs, they apply matrix convolution with filters (also called

[2]This can result in some accuracy problems, especially with high frequency sounds

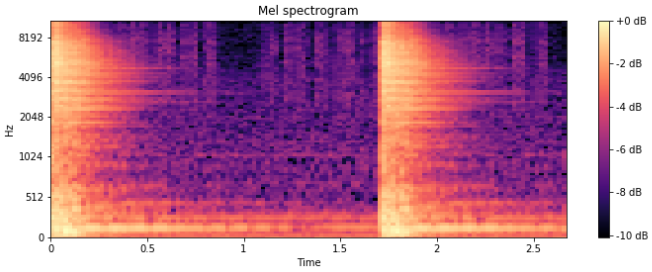

Figure 2. Visual representation of the mel spectrogram of an audio file after pre-processing. We can see how the sequence is repeated to reach the required number of frames.

Parameters for the Preprocessing	
Sampling Frequency	32.000 Hz
audio duration	2 s
Windows Size	35 ms
Window type	Hamming
Overlapping time	17.5 s
FFT resolution	2048
number of mels	96

Table 2. Parameters used for pre-processing the audio.

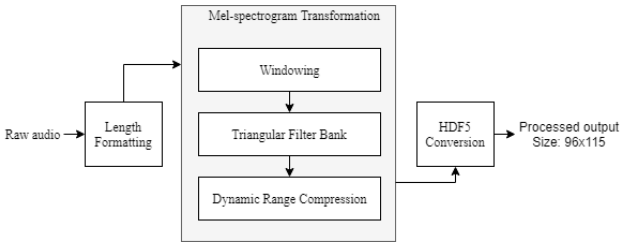

Figure 3. Control chart of the data pre-processing.

kernels) in order to extract relevant information. By defining a proper loss function, CNNs correct their estimation at each iteration to get closer to the desired output (this action is often referred to as *backward pass*). Their ability to learn relevant features with translation invariant makes them appropriate for audio file classification.

4.1.1. Baseline architecture

A first serie of experiments has been carried out in order to choose the CNN architecture that will serve as baseline throughout the project, based on its results over the original dataset. Table 3 summarizes all the experiments carried out and the different parameters with their corresponding results. Following these, it has been decided to use a 5-layer CNN with {48, 48, 64, 64, 128} filters of size 5x5 and dropout of 0.3. At each convolutional block, we apply in the following order: pre-activation (He et al., 2016) (which comprises of batch normalization (Ioffe & Szegedy, 2015) and ReLU), convolution with a kernel size of 5x5, batch normalization, ReLU, max pooling with a 2x2 kernel and dropout (Srivastava et al., 2014). The last convolutional block is connected to a fully-connected layer (short: **FCL**) with dropout and a softmax layer. The loss function used to

Num. layers	Dropout	Val. acc.	
		k = 3	k = 5
3	0.3	64.1%	68.0%
3	0.6	38.3%	46.5%
3	0.8	21.9%	25.8%
4	0.3	61.8%	71.9%
4	0.6	31.5%	46.8%
4	0.8	14.7%	21.2%
5	**0.3**	**66.0%**	**72.8%**
5	0.6	30.7%	52.5%
5	0.8	7.00%	24.3%

Table 3. Summary of the results of the different baseline structures. Best results are highlighted in bold.

train this baseline is categorical cross entropy (short: **CCE**), and it is optimized using Adam (Kingma & Ba, 2014) with weight decay (Loshchilov & Hutter, 2018) because it was proven to be very fast at converging. The optimization uses a learning rate of $1e-3$, betas equal to 0.9 and 0.999, and a weight decay of $1-5$, as suggested by (Loshchilov & Hutter, 2018). The batch size was fixed to 64, as in *Fonseca et al.* and each experiment was run for 100 epochs. The data is shuffled before each epoch to alleviate any pattern that might be present in the ordering of the data. Initially, we experimented with mean absolute error (short: **MAE**) but as the results were not as conclusive, we decided to perform our analyses with CCE. MAE did not work well for *Fonseca et al. 2019* either.

On a side note, we chose to work with a simple CNN architecture rather than a more complex one because we are not interested in having the highest accuracy possible. Instead, our objective is to be able to quantify the improvement we gain using different techniques over a common baseline architecture.

4.2. Label smoothing

When confronted to a certain amount of noisy labels, an interesting approach to solve the problem of target unreliability is label smoothing (Szegedy et al., 2016). Since we cannot be fully certain that the labels given in the training are correct, we assign some probability to the assigned target class and some smaller ones to all the other classes (equally distributed):

$$P(class) = \begin{cases} \frac{\epsilon}{|class|} & \text{if } class \neq target\_class \\ 1-\epsilon & \text{otherwise} \end{cases}$$

Where ϵ is a hyper-parameter that controls the uncertainty, and $|class|$ is the number of classes. The motivation behind this approach is that instead of modifying the loss function, we modify what goes into it.

Because we know which audio files have been manually verified, we decided to extend the label smoothing technique and define a hyper-parameter that allows us to decide whether to apply label smoothing on the whole dataset or

only on the noisy labels. To reference it in future sections, we call this hyper-parameter δ. When it is set to false, label smoothing is applied to **all** the data, whereas when it is set to `true`, label smoothing is just applied to the noisy labels.

4.3. Data augmentation

Data augmentation is a common technique mainly used when the training data size is too small. It consists in the generation of similar data to the training by applying some kind of transformation (e.g. scale and rotation for images). It allows models to find good data representation without overfitting. In order to further experiment with noisy labels, we implemented three types of data augmentation. The first one uses speed perturbation, the second one uses pitch perturbation (Ko et al., 2015) and the third one uses Mix-up training (Zhang et al., 2018).

4.3.1. Speed Perturbation - WSOLA

We first start by changing the speed of the audio signal without affecting its pitch and timbre. We achieve this by using the *tempo* function provided by *sox*[3].This algorithm is based on the WSOLA methodology (Verhelst & Roelands, 1993). WSOLA is a technique arising from extensive research on speech synthesis which cuts the original signal and overlap it in such a way that does not affect the pitch of the signal.

(Ko et al., 2015) states that speed perturbation skews the log-mel spectrogram towards lower frequencies, which means that coefficients corresponding to energies in high frequencies will have very little influence. Despite this fact, the paper claims that this is not prejudicial to the final accuracy of the system. Nevertheless, this project is a speech recognition based research and it is known that most relevant information is usually present in lower frequencies (Rabiner & Schafer, 2011). This means that we can have some undesirable effects in audio processing applications. For that reason, we will assess the isolated effect of speed perturbation on our baseline architecture using 0.7, 0.8, 0.9 and 1.2 speed values [4]. The exact procedure is explained in section 5.

4.3.2. Pitch Perturbation

To perform pitch perturbation, we make a standard resampling of the signal. Modern resampling algorithms make sure to not cause any pitch perturbation effect (e.g. librosa library) to the audio. For that reason, we used the *speed* function in *sox*, which changes the pitch when the signal is resampled. For this data augmentation technique, we decided to use values corresponding to 0.7 and 1.2. We will observe their effect by proceeding in a similar way as for speed perturbation in the former section.

For both speed and pitch perturbations, we will have some changes in the audio duration. For values lower than 1, we will end up with longer audio files (more than 2 seconds). Equivalently, values bigger than 1 will result in smaller audio files. To address this, we just take the first 2 seconds of larger audios and use zero padding (with the data centered) for smaller audios. For instance, if the data is v = [1,2,3] and we perform zero padding to obtain a length of 9, the resulting file will be $v_{transformed}$ = [0, 0, 0, 1, 2, 3, 0, 0, 0].

[3] http://sox.sourceforge.net/

[4] A value of 0.7 is to be taken as a speed value of 70% of the original speed, and so on.

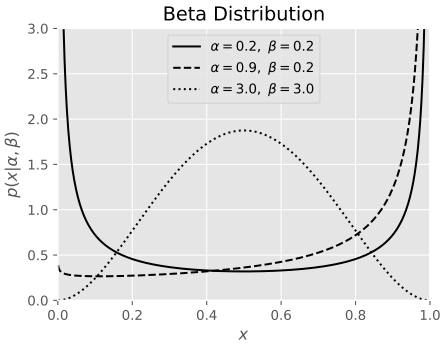

Figure 4. Comparison of Beta distribution for different values of α. Notice that when β=α, the resulting function is symmetrical. This plot shows that the lower the value of α, the higher the probability to get a sample at the edges of the range. The attribution of the code to generate this plot goes to [6]

4.3.3. Mix-up

Mix-up (Zhang et al., 2018) is a data augmentation technique that was proposed to combat memorization in large deep neural networks (short: **DNNs**). Mix-up generates a new sample from a linear combination of two samples chosen at random. Thus, the new resulting sample is:

$$\begin{aligned} \widetilde{x} &= \lambda x_i + (1-\lambda)x_j \\ \widetilde{y} &= \lambda y_i + (1-\lambda)y_j \end{aligned} \tag{1}$$

where x_i and x_j are the two samples chosen at random, and y_i and y_j are the one-hot encoding labels, respectively. λ is a random variable in the range [0-1] drawn from a Beta distribution, $\lambda \sim \text{Beta}(\alpha, \beta)$. In this case, $\beta = \alpha$, and α is a hyper-parameter to tune. The reason to set $\beta = \alpha$ is to force the probability density function to be symmetrical to the expected value, $\mathbf{E}[\lambda] = 0.5$ (see figure 4). Another constraint that we set is $\alpha < 1$. There are several reasons for this. The first one is that the higher is the value of α, the more regularization effect is applied. The other reason, very linked to the previous, is that if $\alpha < 1$, most of the sampled λ's will take either high or low values (as seen on figure 4), and this will result in a low perturbation of the one-hot encoded labels. We have chosen an$\alpha = 0.3$ as it is a similar value to the one used in the original paper.

Mix-up has been successfully applied to speech (Meden-

[6] http://www.astroml.org/book_figures/chapter3/fig_beta_distribution.html

Model	Val. Acc.	Test Acc.
Baseline	**72.8%**	**66.9%**
Clean labels only	50.9%	44.7%

Table 4. Result of experiment 1. These results show that including noisy labels is indeed beneficial to the CNN.

nikov et al., 2018) and audio (Jeong & Lim, 2018) tasks. To perform Mix-up we implemented the formula above in Pytorch, following closely the method that the authors provide.

5. Experiments

Our experiments are divided in four main parts. Each part is related to the management of noise in a dataset. However, they vary drastically in their functioning. The first experiment will demonstrate the effect of a large amount of noisy labels in a big dataset compared to a smaller dataset with only clean labels. The second experiment will test the efficiency of smooth labelling. Then, we explore further the field of data augmentation by trying out different ways to increase the data number and finally, we observe the combined effect of all techniques previously mentioned.

From now on, when presenting a modified version of the dataset, the notation (*X/Y/Z*) will represent the number of audio files in the training, validation and testing sets respectively.

All the experiments were carried out using the baseline architecture and parameters, unless specifically stated differently. Refer to the baseline architecture section for further information.

Our experiments were carried out on three different devices: (1) the cluster provided by the School of Informatics at the University of Edinburgh, (2) a local GTX-1070Ti GPU and (3) the Google Cloud Platform (One Tesla P100 GPU).

5.1. Experiment 1: Noisy labels

The first experiment is directly based on the original paper. Our incentive was to demonstrate the beneficial effect of noisy labels on a classification task. To do so, we compared the result of our baseline architecture when training with: **full original training set** (*17,310/611/611*) comprising of 15,813 noisy labels and 1,772 clean labels and when training with **only clean training data** (*1772/611/611*). Results can be seen in table 4. We can see that using noisy labels is indeed beneficial to the CNN. The results show a drastic improvement of 22% which confirms the results obtained in (Fonseca et al., 2019).

5.2. Experiment 2: Label smoothing

Our incentive to carry out this experiment on label smoothing was based on the taxonomy provided about noisy labels

Smoothing	δ	**Val. acc.**	**Test. acc**
None (baseline)	×	72.8%	66.9%
0.01	×	72.9%	68.8%
0.05	×	**73.9%**	**68.9%**
0.1	×	73.6%	68.1%
0.01	√	63.7%	67.7%
0.05	√	72.8%	69.4%
0.1	√	73.1%	69.0%

Table 5. Result of experiment 2. √ is the case that δ is set to True, and False otherwise.

in (Fonseca et al., 2019) (shown in table 1). In the paper, it is shown that there is approximately 60% of data with incorrect or incomplete labels. Such labels induce fundamentals errors during the training of the models. Overall, we observed that among the noisy labels, there are three types of misclassification that we can improve with smooth labelling: incorrect labels/I-V, incomplete/I-V labels and ambiguous labels. These three account for ~12% of the whole data. Therefore, we started with a smoothing value of 0.1. (which represents approximately the ~12%). Then, we decided to try different ϵ that are smaller and greater than 0.1. For ϵ>0.1, we tried 0.2, 0.3 and 0.5, but did not find any improvement. This might be because if a lot of uncertainty is introduced in a network, the task of generalisation becomes harder. The results for $\epsilon \leq 0.1$ are reported in Table 5. Additionally, we make a distinction between label smoothing over the entire training set and label smoothing over the data with noisy label only. This is controlled by δ, as explained in Section 4.2.

The best results on the validation set were obtained with $\epsilon = 0.05$ (without affecting the uncertainty of the manually verified labels). Unsurprisingly, an espilon value of 0.05 is also the recommended choice in the original paper, (Szegedy et al., 2016) (which corresponds to 1 over the total number of classes). We can see from table 5 that the improvement of our best model compared to the baseline is characterised by an increase of 1.1% in the validation set and 2.0% in the testing set. We therefore argue that label smoothing is a useful technique for machine learning system where uncertainty in the labels is present.

5.3. Experiment 3: Data augmentation

The experiments of this subsection have been carried out to assess the effect of different levels of data augmentation. To evaluate the benefits of data augmentation on its own, we will not use label smoothing. Each data augmentation level adds 17,310 transformed instances. Therefore, the size of the training data for each subexperiment is a multiple of the original one, as shown in table 6.

TECHNIQUE	VALUES	TRAINING SIZE	VAL. ACC.	TEST ACC.	DIFF. VAL.	DIFF. TEST
BASELINE	NONE	1x	72.80%	66.90%	-	-
SPEED PERTURB.	[0.7]	2x	73.99%	72.00%	+1.19%	+5.1%
	[0.7, 0.8]	3x	75.13%	69.25%	+ 2.23%	+2.35%
	[0.7, 0.8, 0.9]	4x	74.63%	67.40%	+1.83%	+0.50%
	[0.7, 0.8, 0.9, 1.2]	5x	75.88%	70.50%	+3.08%	+3.60%
PITCH PERTURB.	[0.9]	2x	75.44%	71.16%	+2.64%	+4.26%
	[0.9, 1.2]	3x	77.89%	73.80%	+5.09%	+6.90%
SPEED & PITCH	[0.7] - [0.9]	3x	75.26%	72.20%	+2.46%	+5.30%
	[0.7, 0.8] - [0.9, 1.2]	**5x**	**76.79%**	**73.47%**	**+3.99%**	**+6.57%**

Table 6. Data augmentation analysis compared to the baseline system. We compare validation and testing accuracies. "*Diff. Val.*" and "*Diff. Test*" represent the difference in accuracy between the corresponding model and the baseline.

Here are the different subexperiments carried out:

Speed Perturbation

1. Original data plus **0.7** speed perturbation data (*34,620/611/611*).
2. Original data plus **0.7** and **0.8** speed perturbation data (51,930/611/611).
3. Original data plus **0.7**, **0.8** and **0.9** speed perturbation data (*69,240/611/611*).
4. Original data plus **0.7**, **0.8**, **0.9** and **1.2** speed perturbation data (*86,550/611/611*).

Pitch Perturbation

1. Original data plus **0.9** pitch perturbation data (*34,620/611/611*).
2. Original data plus **0.9** and **1.2** pitch perturbation data (*51,930/611/611*).

Speed Perturbation and Pitch Perturbation combined

1. Original plus **0.7** speed perturbation and **0.9** pitch perturbation (*51,930/611/611*).
2. Original plus **0.7**, **0.8** speed perturbation and **0.9**, **1.2** pitch perturbation (*86,550,930/611/611*).

As we can see from table 6, the application of data augmentation on the data results in a considerable increase in performance. Our best performing model, comprising of a mix between speed perturbation and pitch perturbation techniques (see bold) proves the beneficial effect of a tremendously larger amount (x5 the original size) of noisy labels on a dataset. However, further analysis of the results show some interesting trends. For instance, a combination of 0.7 and 0.8 speed perturbations tends to give worst results than a simple 0.7 speed perturbation. Our hypothesis is that it might be happening because we are biasing the system towards lower speed values. This argument is reinforced by the even poorer results of the experiment combining speed values of 0.7, 0.8 and 0.9. We can observe that simply adding a speed perturbation of 1.2 to the previously mentioned combination makes the accuracy increase again. The last results combining both speed and pitch perturbations also supports our hypothesis.

This is interesting in the sense that it constitutes a good guideline regarding how to perform data augmentation. In any case, data augmentation seems to be beneficial to the model. The performance increases according to the data size as well as with the type of augmentation performed. An important note to take into account is that data augmentation helps even when compressing the energy towards lower frequencies, as stated in (Ko et al., 2015).

Mix-up

We assessed the performance of Mix-up by applying it directly on our original data (and therefore "replacing" our original data by the newly-generated one) and by adding the newly-generated data to the existing data ("on top" of the original dataset). We perform Mix-up data augmentation in the log-mel spectrogram level, given that in (Zhang et al., 2018), it is stated that it works properly as well in the raw time domain as in the log-mel spectrogram domain. Our incentive is to test this technique and compare it with the baseline as well as with the data augmentation methods mentioned above. The performance of applying Mix-up on our baseline can be seen in table 7.

We observe that applying Mix-up directly to our original data without preserving the original for training is not beneficial. Therefore, we conclude that it is best to add Mix-up-generated data on top of the original one (increasing the data size by a 2x factor). This last result, which we referred to as "Mix-up with stack", obtained a lower performance in the validation set. Nevertheless, it managed to increase the test performance by 0.62%. This result supports the original paper's hypothesis, which stated that Mix-up is useful because it can improve the generalization error (testing set).

5.4. Experiment 4: Combining all experiments

Given the success of our previous experiments, we extended our research and decided to explore the idea of combining the different methods introduced in this report. We will observe the combined effects of label smoothing, Mix-up, speed perturbation and pitch perturbation. The hyperparameters chosen were the ones that gave the best results in each subsection:

- **Label smoothing** of 0.05 and δ set to false.

Model	Stack	Val. Acc.	Test Acc.
Baseline	×	72.80%	66.90%
Mix-up	×	69.00%	65.00%
Mix-up	√	71.38%	67.52%

Table 7. Results of the effect of Mix-up on our baseline. Mix-up without stack means that the method is applied in place, replacing the original data. Mix-up with stack means that we maintain our original data and augment it to twice its original size using Mix-up.

Smooth.	Perturb.	Mix-up	Val. Acc.	Test. Acc.
×	×	×	72.80%	66.90%
×	√	√	77.29%	72.20%
√	×	√	71.98%	68.00%
√	√	×	**77.95%**	**72.67%**
√	√	√	74.66%	70.00%

Table 8. Experiment 4: results. The first row corresponds to the baseline structure.

- **Speed perturbation** of 0.7 and 0.8.
- **Pitch perturbation** of 0.9 and 1.2.
- **Mix-up** with stack.

Results can be seen in table 8. As we can see, the combination of label smoothing and data perturbations without Mix-up yields the best results. On the other hand, even though Mix-up improves the overall accuracy of the model on its own (see section 5.3), it decreases the accuracy of the models when coupled with data perturbation and label smoothing at the same time.

6. Further work

There could definitely be an improvement in the hyper-parameter choices that we made. However, hyper-parameter optimization is a time-consuming task. Therefore, an extension of our work could include Bayesian hyper-parameter optimization (Snoek et al., 2012), which has been shown to be an efficient way to select the best hyper-parameters configuration. Another interesting extension would be Neural Architecture Search (Zoph & Le, 2017) (with data augmentation, weight decay, dropout, Mix-up and label smoothing as the search space).

7. Conclusions

Throughout the experimental section, we analysed 3 different techniques based on noisy label management. Our research proved the beneficial impact of noisy labels on audio-based deep learning applications. The first three experiments showed that, independently, each technique improves on the overall performance, and that data augmentation is the most relevant, followed by label smoothing and then Mix-up. Data augmentation on its own is the most relevant technique to improve classification performance, followed by label smoothing. Given the good results of the isolated experiments, we decided to combine them and observe their joint effect. We realised that the coupling of certain techniques yielded better results than just stacking the three on top of each other.

We therefore conclude that noisy labels can drastically improve the performance of a good CNN architecture and should not automatically be discarded during classification tasks. However, noise needs to be handled in a proper way. Our research shows that the best way to manage an audio dataset with noisy labels is to use data augmentation (combining speed and pitch perturbation) with label smoothing.

References

Davis, S. and Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, pp. 357–366, 1980.

Fonseca, Eduardo, Plakal, Manoj, Font, Frederic, Ellis, Daniel P. W., Favory, Xavier, Pons, Jordi, and Serra, Xavier. General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline. Submitted to DCASE2018 Workshop, 2018. URL https://arxiv.org/abs/1807.09902.

Fonseca, Eduardo, Plakal, Manoj, Ellis, Daniel P. W., Font, Frederic, Favory, Xavier, and Serra, Xavier. Learning sound event classifiers from web audio with noisy labels. *CoRR*, abs/1901.01189, 2019. URL http://arxiv.org/abs/1901.01189.

Gemmeke, Jort F., Ellis, Daniel P. W., Freedman, Dylan, Jansen, Aren, Lawrence, Wade, Moore, R. Channing, Plakal, Manoj, and Ritter, Marvin. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

Guan, Melody, Gulshan, Varun, Dai, Andrew, and Hinton, Geoffrey. Who said what: Modeling individual labelers improves classification. 2018. URL https://arxiv.org/pdf/1703.08774.pdf.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. URL http://arxiv.org/abs/1603.05027.

Hinton, Geoffrey, Deng, Li, Yu, Dong, Dahl, George, Mohamed, Abdel-Rahman, Jaitly, Navdeep, Senior, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Sainath, Tara, and Kingsbury, Brian. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal

covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Jeong, Il-Young and Lim, Hyungui. Audio tagging system for dcase 2018: Focusing on label noise, data augmentation and its efficient learning. Technical report, Tech. Rep., DCASE2018 Challenge, 2018.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Ko, Tom, Peddinti, Vijayaditya, Povey, Daniel, and Khudanpur, Sanjeev. Audio augmentation for speech recognition. In *INTERSPEECH*, 2015.

Li, Yanxiong, Li, Xianku, Zhang, Yuhan, Liu, Man, and Wang, Wucheng. Anomalous sound detection using deep audio representation and a blstm network for audio surveillance of roads. *IEEE Access*, 6:58043–58055, 2018.

Loshchilov, Ilya and Hutter, Frank. Fixing weight decay regularization in adam, 2018. URL https://openreview.net/forum?id=rk6qdGgCZ.

Maijala, Panu, Shuyang, Zhao, Heittola, Toni, and Virtanen, Tuomas. Environmental noise monitoring using source classification in sensors. *Applied Acoustics*, 129:258–267, 2018.

Medennikov, Ivan, Khokhlov, Yuri Y., Romanenko, Aleksei, Popov, Dmitry, Tomashenko, Natalia A., Sorokin, Ivan, and Zatvornitskiy, Alexander. An investigation of mixup training strategies for acoustic models in ASR. In *Interspeech*, pp. 2903–2907. ISCA, 2018.

Mohamed, Abdel-Rahman, Hinton, Geoffrey E., and Penn, Gerald. Understanding how deep belief networks perform acoustic modelling. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4273–4276, 2012.

Nettleton, David F., Orriols-Puig, Albert, and Fornells, Albert. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial Intelligence Review*, 33(4):275–306, Apr 2010. ISSN 1573-7462. doi: 10.1007/s10462-010-9156-z. URL https://doi.org/10.1007/s10462-010-9156-z.

Rabiner, L. R. and Schafer, R. W. *Theory and applications of digital speech processing*, chapter 6. Prentice Hall, Upper Saddle River, 2011.

Reed, Scott E., Lee, Honglak, Anguelov, Dragomir, Szegedy, Christian, Erhan, Dumitru, and Rabinovich, Andrew. Training deep neural networks on noisy labels with bootstrapping. In *ICLR 2015*, 2015. URL http://arxiv.org/abs/1412.6596.

Robles, L. and Ruggero, M. Mechanics of the mammalian cochlea. *Physiological Reviews*, 81:1305–1352, 2001.

Salamon, Justin and Bello, Juan Pablo. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 3 2017. ISSN 1070-9908. doi: 10.1109/LSP.2017.2657381.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical bayesian optimization of machine learning algorithms. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 2951–2959. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jonathon, and Wojna, Zbigniew. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

Verhelst, Werner and Roelands, Marc. An overlap-add technique based on waveform similarity (wsola) for high quality time-scale modification of speech. In *proceedings of ICASSP-93*, pp. 554–557, 1993.

Xiao, Tong, Xia, Tian, Yang, Yi, Huang, Chang, and Wang, Xiaogang. Learning from massive noisy labeled data for image classification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2691–2699, 2015.

Yu, Changsong, Barsim, Karim Said, Kong, Qiuqiang, and Yang, Bin. Multi-level attention model for weakly supervised audio classification. *arXiv preprint arXiv:1803.02353*, 2018.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. 2017. URL https://arxiv.org/abs/1611.03530.

Zhang, Hongyi, Cissé, Moustapha, Dauphin, Yann, and Lopez-Paz, David. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2018.

Zoph, Barret and Le, Quoc V. Neural architecture search with reinforcement learning. 2017. URL https://arxiv.org/abs/1611.01578.