Deep Learning Project Proposal : Multi-instrument Classification using Partially Labeled Data and Weakly-supervised Learning

Clément Berger
clement.berger@ensta-paris.fr

Ilyes Er-Rammach
ilyes.er-rammach@ensta-paris.fr

Abstract

While single-instrument recognition has made tremendous progress, multi-instrument recognition is still considered as a hard task. A part of the difficulty comes from the lack of huge strongly labeled datasets. Recently, a large dataset of polyphonic audio clips called OpenMIC has been released. The drawback of the size of the dataset is that it is only weakly-labeled. Previous works have proposed to use an attention mechanism and a Recurrent Neural Network structure called Bidirectional Long-Short Term Memory (BiLSTM) to train on this dataset. Most works in this field use Log-Mel Spectrograms to treat the audio signal before giving it to the network. Here we explore the use of Analytic Wavelet Transform (AWT) to generate scalograms wich are then given to a Convolutional Neural Network (CNN). Such transformations are supposed to be more efficient in giving a representation of the whole signal. We also test different BiLSTM configurations and try some data augmentation techniques. While we do not improve state-of-the-art, our results are encouraging. With more computational power, pretraining our scalogram-CNN structure as feature extractor using a big dataset like YouTube should be able to achieve great results.

1. Introduction

1.1. Motivation

Multi-instrument recognition is a subfield of Music Information Retrieval (MIR) in which, given a list of instruments and an audio clip, one tries to tell if these instruments figure in the clip or not. Such a task is very useful for music providers to make recommandations based on affinity with some instrument, or to create filters for research and so on. An efficient model could also be used as a basis (like a feature extractor) for other MIR tasks such that source separation, or music transcription.

Such a task requires not only machine learning skills, but also signal processing expertise. Great results have been achieved in single-instrument recognition, see e.g. [15]. However, polyphonic sounds are the superposition of multiple instruments with different charasteristics and played differently therefore most of these techniques can not be applied for our task.

Another challenge is the difficulty to create datasets. There are roughly two types of datasets of annotated polyphonic sounds. First there are small datasets but very strongly annotated. We can for example cite MusicNet [14] containing 330 examples. Such datasets face big issues like overfitting. The other type of datasets is huge datasets (at least compared to the previous ones) but only partially labeled. In 2018 has been released a dataset called OpenMIC [7] which belongs to the second category. OpenMIC contains 20 000 audio clips of 10 seconds, sampled at 44,1 kHz. However, given an audio clip, we only know if an instrument is in the clip or not but the offset and onset times are not provided. Practically this means that an audio containing 1 second of violin will have the same label as one completely played with a violin. Moreover, there are some missing labels, meaning that given an audio clip, some instrument labels are not provided.

1.2. Problem definition

1.2.1 Signal processing : different transformations

We provide here a quick overview of two different transformations used in audio processing to create a visual representation of a signal.

The first one is the genration of a Log-Mel Spectrogram. This is the result of a transformation based on the Short-Time Fourier Transform. The raw signal is divided into a certain number of overlapping frames, before applying a Fourier Transform on each frame. This result in a time frequency representation of the signal, using color variations to represent the magnitude of the Fourier Transforms. However, human sensibility is not homogeneous in frequencies. To account for this problem, we convert the Hertz into what is called a Mel scale [12]. It is a non linear transformation of the frequencies to get a scale which better describe human hearing. The resulting visual representation is called a Log-Mel Spectrogram. Such representations have been successfully used for music recognition, see e.g. [11].

The second one is the AWT used to generate a scalogram. Given a signal $x(t)$ and a wavelet $\psi(t)$ satisfying $\psi(t) = 0$ for $t < 0$, a function acting as a filter on the signal, the AWT of the signal is $AWT_\psi(t,s) = (1/2\pi)\int_0^\infty \bar{\psi}(s\omega)x(\omega)e^{i\omega t}d\omega$. This is a variant of the Continuous Wavelet Transform comonly used in signal processing. We proceed as for the spectrogram to get a visual representation of this transform, which is then called a scalogram.

1.2.2 VGGish network

This section is devoted to briefly present a network created recently for multi-instrument recognition, called VGGish [5]. This name simply comes from the fact that it is derived from the classical VGG model [8] used for image recognition. Amongst other changes, it has been modified to receive a Log-Mel Spectrogram as entry and the end of the network acts now as a compact embedding layer. A precise definition can be found here.

1.2.3 F1-score

For instrument classification, the performances of an algorithm are generally not measured in terms of the usual accuracy. Indeed, as some instruments are easier than others to recognize, a difference of accuracy doesn't perfectly reflect an improvement in the classification of hard instruments. Therefore, another metric called the F1 score has been created specificaly for this purpose [10].

We talk about true positive (TP) when both the network and the ground truth set the instrument as present in the audio. When the prediction is negative but should be positive, it is called a false negative (FN). There are similarly the false positive (FP) when the network predicts the presence when it should not, and true negative when ground truth and network agree on the absence of the instrument. Then the precision (P) and recall (R) are introduced as :

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

Finally the F1-score is calculated as :

$$F1 = \frac{2P \times R}{P + R}$$

1.2.4 Training using OpenMIC

In this work we try to train a network on the OpenMIC dataset for multi-instrument recognition.

The OpenMIC dataset has been labeled with 20 instruments. For each of the 20 000 audio clips and each instruments, we have a binary value indicating if we have a label for this couple (audio, instrument) or not, and if we have so, we also have the probability that the instrument is effectively in the audio clip. In addition of the raw audios and labels, OpenMIC also provides features extracted using the VGGish. More precisely, a Log-Mel Spectrogram is computed for each raw audio, one frame corresponding to 1 second, and then given to the VGGish. After that, a Principal Component Analysis (PCA) is done on the extracted features for each frame. We end up with 10 vectors (one per second) containing 128 features, for each audio clip.

2. Related work

2.1. Attention mechanism

Multi-instrument recognition is not a new topic, and several works have been published. In 2018, Kong *et al.* [9] introduced an attention mechanism to train a CNN on another weakly-labeled dataset, AudioSet [3]. This mechanism has then been used by Gururani *et al.* [4] on the OpenMIC dataset. Starting from the VGGish features furnished by OpenMIC, an embedding layer is then followed by a prediction layer coupled with the attention mechanism to produce the desired predictions. Let's describe briefly the attention mechanism.

In Multi-Instance Label problems, a bag of labels is produced by a score function $S(X) = \mu(f(x)_{x\in X})$, where X is the input composed of multiple instances x, f gives a prediction for each of these instances and μ aggregates these predictions to give the final prediction. The score (the global score S as well as the instance-level scores f) represents in our case the probability of the label (here the probability for the instruments to effectively be part of the audio). Usually a max or an average is used for the aggregation function. However, previous works as Kong *et al.* [9] have shown that learning this function could lead to significant improvements. With this in mind, Gururani *et al.* parametrized this operator as :

$$S(X) = \sum_x w_x f(x)$$

with

$$w_x = \frac{\sigma(v^T h(x))}{\sum_{x'} \sigma(v^T h(x)')}$$

where $h(x)$ is the output of the embedding layer, v is a vector to be learned, and σ is the sigmoid function. Note that the division is here to enforce the sum of coefficients to be 1. The formula also means that the attention layer takes as input the results of the embedding layer and then use this information to aggregate the results of the prediction layer represented by f.

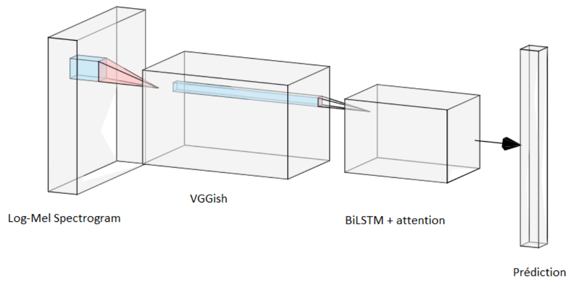

Figure 1. Structure using Log-Mel Spectrogram

2.2. Using a BiLSTM architecture

In 2020, Amir Kenarsari-Anhari [1] built upon these works to improve the efficiency of the network introduced in the previous section. His idea was to use Recurrent Neural Networks (RNN) to better exploit the fact that the different vectors of features correspond to different seconds of the audio clip (as explained in the Section 1.2.1). In particular, he used a BiLSTM structure. An LSTM is a particular structure of RNN built especially to improve the ability of the network to pass information through time, which has proved its efficiency in Natural Language Processing or to study Time Series (see [6] for a complete description). It is here set to be bidirectional in the sense that it treats the data by following its direction as well as reversing it.

Practically, the BiLSTM replaced the embedding and prediction layers used by Gurarani *et al.* to be coupled with the attention mechanism. The resulting architecture, presented in figure 1, improved the efficiency of the model and will be considered as the state-of-the-art for us.

2.3. Using scalograms

In 2020, a paper from Dutta *et al.* [2] investigated the use of scalograms based on AWT using a Morse wavelet to replace the spectrograms, for *single*-instrument recognition. The problem of the spectrograms using STFT is that as the signal is cut in pieces before processing each part independently, it is harder to get back the unity of the signal. That is a particular issue in our case as we do not have the onset and offset times of the instruments. This means that some pieces of the signal (after cutting) are treated as if they were containing an instrument while they may not. This procedure also significantly looses efficiency when the signal contains very low or very high frequencies.

On the contrary, the AWT is computed over the whole signal. Therefore is supposed to be able to extract features taking into account the entire audio clip, which should thus be better.

The generated scalograms have been given to a CNN to extract the extract the features, ending with a classification layer. Used to distinguish 14 different instruments, the model achieved a reasonnable accuracy and is to be followed by further works on the topic.

2.4. Data augmentation

As told before, a big issue in MIR can be the absence of a big dataset corresponding to a given task. Naturally researchers have been working on different data augmentation techniques, general or specified for a task. We can cite Schütler and Grill [13] who proposed to randomly add a Gaussian noise to slightly modify the data.

Zhang *et al.* [16] experimented interpolation between different samples. More precisely, given $\lambda \in [0, 1]$ and two signals x and y, labeled by l_x and l_y, we create a new signal z with the label l_z by computing :

$$z = \lambda x + (1 - \lambda) y$$

$$l_z = \lambda l_x + (1 - \lambda) l_y$$

3. Methodology

3.1. BiLSTM

As said earlier, we consider [1] as our state-of-the-art. It has to be noted that the article does not provide all the details necessaery to reproduce the exact same architecture, neither their code. Thus our first step is to try to reproduce their result, using an architecture corresponding to 1.

1. First, we get the features of the VGGish model furnished by the OpenMIC website.

2. Then our network is divided into two parts :

   - On one hand we set the attention mechanism as described in Section 2.1.
   - On the other hand we use a BiLSTM architecture with dropout. Here we try two versions with different number of layers to see the impact of the depth of the RNN component.

3. finally we agregate these two parts with a tanh non-linearity to get our predictions

The best of these networks (for 1 layer, as discussed later in Section 4.) gives us our benchmark before trying other techniques. We will from now on refer to this architecture as the Mel architecture.

3.2. Spectrograms

Next, we want to see if the AWT and spectrograms used by Dutta *et al.* can be transposed to the multi-instrument recognition. Indeed, the reasoning behind it is independent

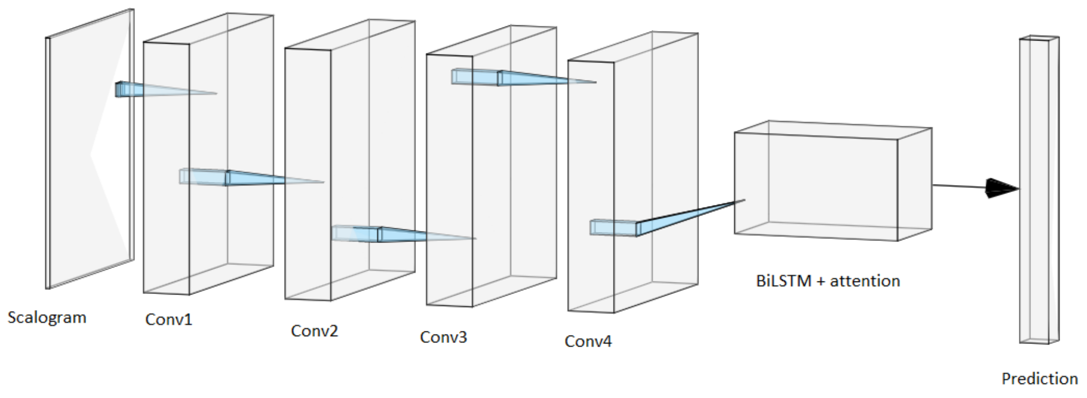

Figure 2. Convolutional network taking scalograms

of the number of predictions we want to make and thus is still relevant in our case.

Thus for this part, we take the raw audio sample of OpenMIC. We then compute their scalograms using a Morse wavelet. Here we are challenged by our lack of computational powerand memory. Indeed, to be able to train on our computers and even to be able to compute all scalograms in a reasonable amount of time, we had to reduce the resolution of the scalograms to 128x128. Hence we expect a loss of performance due to our imprecision (this has to be compared with the VGGish features which have already been calculated for us !).

The next issue was that the VGGish network had been trained with spectrograms, meaning that we can not use its pretrained version. With this constatation in mind, we deciced to replace the VGGish by the CNN used by Dutta *et al.*, as it has been proven to be quite efficient. The architecture is the following :

- **conv1** : an 8-filter convolutional layer taking the 128x128 scalograms
- **conv2** : a 16-filter convolutional layer
- **conv3** : a 32-filter convolutional layer
- **conv4** : a 32-filter convolutional layer followed by a dropout
- our **BiLSTM + attention** structure

All convolutional layers have a 3x3 kernel. This architecture is represented in figure 2. We will refer to it as the CNN architecture. As we had to reduce the resolution, we wondered if such a deep network was necessary. To test this, we also tried the same structure without the **conv3** and **conv4** layers (see 3). We will refer to it as the CNN-reduced architecture.

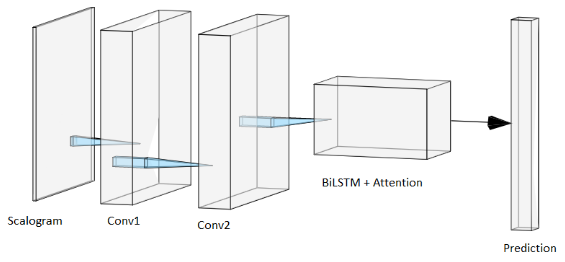

Figure 3. Convolutional network taking scalograms, reduced structure

3.3. Data augmentation

Our last step is to see the effect of data augmentation techniques, as suggested in [1].

A difficulty is that due to our limited computational power, we have to use precalculated scalograms/spectrograms (it already took about a day, so it is unrealistic to do it during the training). Thus the transformations we use have to be easily applied to the representations directly without too much calculations.

The first technique we use is the interpolation described earlier, as [1]. We also try to add a Gaussian noise as in [13]. Finally, as we have an image as representation of our signal, we thought we could use some techniques coming from the computer vision field. It appeared that due to the form of our representations, a horizontal flip of the image exactly corresponds to playing the audio backward, and thus we use

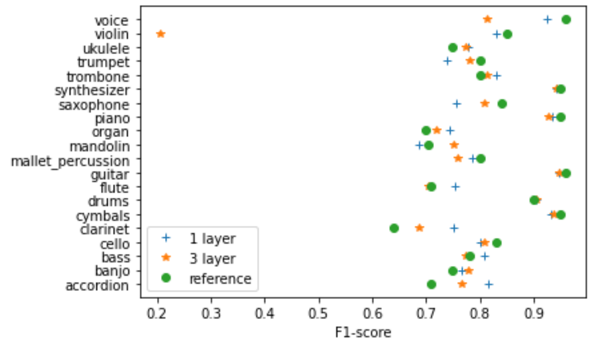

Figure 4. Comparison of the instrument-level F1-score between our different Mel architectures and state-of-the-art (denoted by *reference*), with no data augmentation.

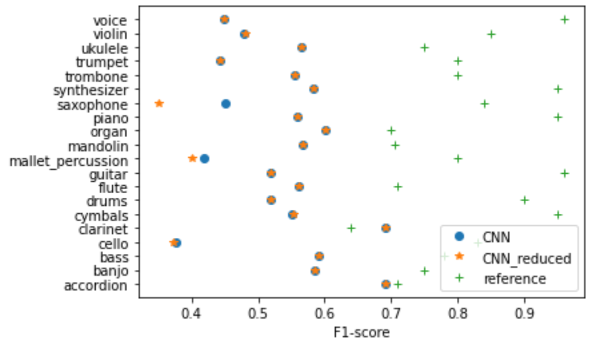

Figure 5. Comparison of the instrument-level F1-score between our Mel 1-layer architecture and our CNN and CNN-reduced architectures, with no data augmentation.

this as our last data augmentation technique.

We try to use them independently and then altogether. We make these tests both on the Mel architecture and the CNN architecture.

4. Evaluation

4.1. BiLSTM

Our complete code can be found here. Part of it has been taken from the code of Gururani *et al*. [4]. The implementation has been done in pytorch.

The dropout rate for the BiLSTM has been set to 0.2 as in [1]. We try to set the number of BiLSTM layers to 1 and 3. This part has be done without any form of data augmentation. To be close to the conditions of the paper, the batch-size has been set to 32 and we trained for 200 epochs. The results are sumed up in 4. We gave the F1-score of our model after 200 epochs, for each instrument. For the test set and train set split, we followed the split given by OpenMIC which ensures its relevance (in terms of representation of each instrument for example).

Our results are pretty closed to state-of-the-art. Depending on the instrument, the 1-layer version of the Mel architecture can be a bit better (clarinet, banjo, accordion, flute..) or a bit less good (trumpet, violin) than [1]. We deduce that this is probably the almost same structure that is used in this paper.

However we note a worse performance for the 3-layer version, especially for the violin. Taking into account the fact that the VGGish model already extracts high level features thanks to its pretrain and well designed structure, a deeper network after it may not be necessary and may create more confusion than help. So from now on all BiLSTM structures will be done using only 1 layer.

4.2. Spectrograms

We now put to the test the use of AWT and spectrograms. The precalculation of the spectrograms is the only part that has been done in MATLAB, as it provides tools to efficiently compute an AWT with Morse wavelet. The results have then been exported to be given to python.

As before, we take a batch-size of 32 and we train for 200 epochs. We use the same BiLSTM + attention structure as above, with only 1 layer as discussed. The results are compared to state-of-the-art. The results are represented in figure 5.

We can immediately see that the results are mostly worse than state-of-the-art (the clarinet is the exception). However we still achieve great results. We recall that this is compared to a network that has been pretrained on a big dataset, with a PCA. Thus the results seem to be promising. With more computational power, one could do the same for our CNN architecture, and great results would be expected.

We also note that the depth of the CNN has only a small influence on the performance. Most instruments do not differ between them. It still seems that the deeper network is more efficient, so is to be preferred.

4.3. Data augmentation

At last we try to see the influence of data augmentation on our models. Each data augmentation added for a training is applied to one tenth of the each batch. For example, if we only use interpolation, one tenth of each batch will be modified by interpolation. When we use all techniques, one tenth is modified by interpolation, one tenth by noise and one tenth by flip.

The variance of the centered gaussian noise is set to 0.2 as in [13]. For the interpolation, the coefficients are set by using a beta distribution of parameters 0.2 and 0.2, to follow [1]. We keep the batch-size and number of epochs to continue our comparisons.

We first perform each data augmentation technique individually, on the Mel architecture. Then we use them all at once. Results are compared with no data augmentation, see 6. We then repeat the procedure on the CNN architecture, see 6.

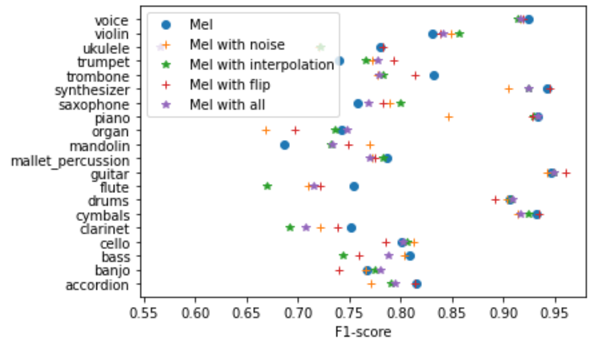

Figure 6. Comparison of the instrument-level F1-score between our different data augmentation strategies for the 1-layer Mel structure.

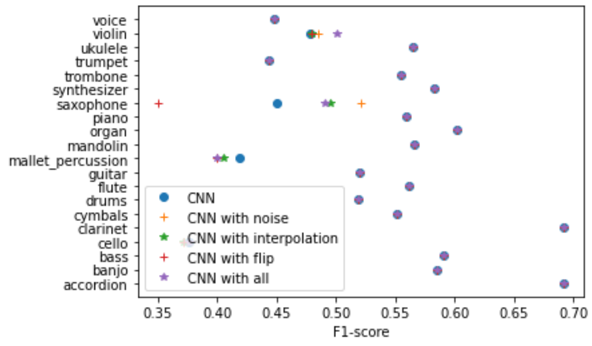

Figure 7. Comparison of the instrument-level F1-score between our different data augmentation strategies for the CNN structure.

We observe that for most instruments, no data augmentation gives better results, especially for the Mel architecture. This is a surprising result. We are not sure where it exactly comes from. Our best guess is that the number of epochs may be a bit too low to see real benefits from data augmentation, considering the size of the dataset. We still see that the CNN architecture responded better than the Mel architecture to these transformations, indicating that our techniques may be more compatible with the spectrograms.

The comparison between each technique is not very clear. A difficulty is that it depends on the instrument we look at. Some techniques seem to perform better on some instruments and worse on others. All in all, using all techniques seems to give more consistent results, as it combines all the benefits of each technique.

5. Conclusion

References

[1] Amir Anhari. Learning multi-instrument classification with partial labels. 01 2020. 3, 4, 5

[2] Arindam Dutta, Dibakar Sil, Aniruddha Chandra, and Sarbani Palit. Cnn based musical instrument identification using time-frequency localized features. pages 1–6, 05 2020. 3

[3] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017. 2

[4] Siddharth Gururani, Mohit Sharma, and Alexander Lerch. An attention mechanism for musical instrument recognition. *CoRR*, abs/1907.04294, 2019. 2, 5

[5] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson. Cnn architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135, 2017. 2

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. 3

[7] Eric J. Humphrey, S. Durand, and B. McFee. Openmic-2018: An open data-set for multiple instrument recognition. In *ISMIR*, 2018. 1

[8] A. Zisserman K. Simonyan. Very deep convolutional networks for large-scale image recognition. *ArXiv*, September 2014. 2

[9] Qiuqiang Kong, Yong Xu, and Mark Plumbley. Audio set classification with attention model: A probabilistic perspective. 04 2018. 2

[10] A. Mesaros, Toni Heittola, and T. Virtanen. Metrics for polyphonic sound event detection. *Applied Sciences*, 6:162, 2016. 2

[11] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. *CoRR*, abs/1703.06697, 2017. 2

[12] E. B. Newman S. S. Stevens, J. Volkmann. A scale for the measurement of the psychological magnitude pitch. *Acoust. Soc. Am.*, 8, 1937. 1

[13] Jan Schlüter and Thomas Grill. Exploring data augmentation for improved singing voice detection with neural networks. 01 2015. 3, 4, 5

[14] John Thickstun, Z. Harchaoui, and Sham M. Kakade. Learning features of music from scratch. *ArXiv*, abs/1611.09827, 2017. 1

[15] M. Lagrange V. Lostanlen, J. Andén. Extended playing techniques: The next mile-stone in musical instrument recognition. *5th International Conference on Digital Libraries for Musi-cology*, 09 2016. 1

[16] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017. 3