End-to-End Sound Recognition using Temporal Convolutional Networks

Eric Schölzel
Chair of Media Design
Technical University Dresden

Abstract—In this work, an extensible and efficient end to end pipeline for sound classification based on state-of-the-art techniques in audio recognition using artificial neural networks is constructed. The pipeline operates using only mel spectrograms and includes data augmentation through implementation of the SpecAugment algorithm. Additionally, the pipeline can handle input sequences of arbitrary length as well as mapping them to an output sequence of the same size, which can be used for tasks like audio segmentation.

I. Introduction

Since the success of AlexNet[1], deep learning has been adapted to a wide range of distinctive tasks. While implementations of recurrent neural networks like the Long Short-Term Memory (LSTM)[2] seem to be a natural choice when evaluating time series like audio data, recent research has shown that purely image based methods operating on spectrograms can reach state of the art performance or even outperform recurrent models while being fully parallizable[3]. Additionally, they allow for simple and very effective data augmentation by manipulating the spectogramm itself instead of the source audio as shown in the SpecAugment paper[4].

For this work, three key components get introduced within the next sections and then combined to form an flexible end-to-end pipeline. First, it is shown why spectogram images are used as features. Second, the choice of the neural network architecture is explained. The third component will be an highly efficient data augmentation technique called SpecAugment. After that, these components will get combined to form the pipeline.

II. Image Recognition on Spectrograms

Neural networks are able to learn on raw audio due to deep architectures performing feature extraction directly from waveform audio. Despite being a challenging task, it has been shown that deep architectures are even able to be on par some the earlier spectogram based models[5].

However, with this approach, a neural net has to learn feature representations first which comes at increased network complexity and computational cost.

For traditional machine learning algorithms like k-nearest-neigbours or random forests, it is necessary to compute features that represent the data in a more abstract way. A common choice is to compute Mel-Scale Frequence Cepstral Coefficients (MFCC)[6]. In MFCC, magnitude spectra are projected into a smaller set of frequency bands which are converted to logarithmic magnitudes. Finally, these magnitudes get approximately whitened and compressed through discrete cosine tranform (DCT).

It has been shown that the last step is not necessary when using deep learning models and can even lead to worse results due to information removal[7]. Without this it leads to the log-mel spectrum which is a commonly used feature representation in audio deep learning.

With the usage of Spectograms as input features, audio recognition tasks can be modeled as time-dependent image recognition tasks. Therefore, it allows taking advantage of state-of-the-art image recognition methods.

III. Network Architectures

Deep models for image recognition tasks have been heavily evolved in the past years. Researching various approaches like the inception block[8] or residual connections[9] led to constantly improving models as well as improved algorithms like better weight initialization[10] or optimizers like Adam[11].

Using spectrograms, Image recognition models like ResNet or Inception have been proven to yield promising results on audio classification tasks[12]. However, those models are mainly optimized for training tasks like ImageNet[13] including millions of images with many different class labels. Therefore, this types of models have millions of parameters which comes at high computational cost at training time. Inference is much cheaper, but saving the trained weights of such models alone needs hundreds of megabytes of disk space.

This is one of the reasons why for example mobile apps featuring deep learning image recognition often need a permanent online connection and don't perform the inference themselves.

Additionally, these models need a fixed input and output size, which can be a problem if there are audio sequences with great difference in length. For example, they are not suited for dealing with sequence-to-sequence tasks such as speech recognition (where input can be audio of arbitrary length and output can be a text of arbitrary length, completely depending on the input) or segmentation of audio clips.

For learning on such temporal data, mostly recurrent architectures have been state-of-the-art over many years. However, recurrent networks have some drawbacks which specific non-recurrent networks called Temporal Convolutional Networks (TCN) overcome while achieving even better results on temporal data. By analyzing the function of both recurrent approaches and the TCN architecture, it is shown why TCN units can outperform recurrent architectures in many use cases. Additionally, further optimization of the network architecture by using feature extraction is explained.

A. Recurrent Architectures

In recurrent layers, the output of a unit at timestep t is fed back as input at timestep $t+1$. This implies that the activation function of the neuron is applied over and over again:

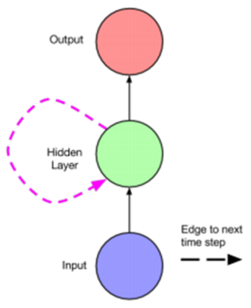

Figure 1. A basic recurrent unit. Source: [14]

Trying to backpropagate errors through repeated usage of activation functions like the *sigmoid* function causes the Vanishing Gradient problem which makes recurrent networks hard to train[14]:

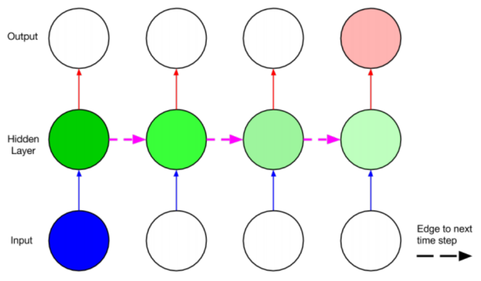

Figure 2. The Vanishing Gradient Problem on recurrent neural networks makes learning long-term dependencies difficult. Source: [14]

This lead to the design of the LSTM[2] and Gated Recurrent Units (GRU) [15]. Though details will not be discussed in this work, it is important to note that these architectures are quite similar and enable learning despite the Vanishing Gradient through the introduction of Gates, but are still recurrent units and convergence can still be very slow. Even a fast GPU still has to wait for the calculations of timesteps $t_0 ... t_{i-1}$ in order to calculate the results of timestep t_i. Therefore, both training and inference cannot simply be parallized as it is common with other network architectures, making them slow and inefficient.

Because of that, instead of (still widely used) recurrent architectures, another type of architecture called Temporal Convolutional Network will be used in this work.

B. Temporal Convolutional Networks (TCN)

A novel architecture for learning on temporal data was introduced in WaveNet[16] and lead to the introduction of a general TCN architecture in 2018[3]. Temporal Convolutional Networks uses dilated 1D convolutions instead of recurrent connections.
Like a RNN, a TCN is able to map any arbitrary long input sequence to an output sequence of the same length and for convolutions at timestep t_i, only information of timesteps $t_0, t_1, ..., t_i$ is used (causal convolutions)[1]. For time series, it should use information from a history - or receptive field - that is as long as possible. The usage of dilated convolutions leads to an exponential growing receptive field:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) * x^{(s-d*i)}$$

x - input vector $x \in \mathbb{R}^n$
f - filter $f : \{0, ..., k-1\} \to \mathbb{R}$
s - element s of the input sequence
d - dilation factor
k - filter size

The history size of such layers is equal to $(k-1)*d$. Using a exponentially increasing d per layer of the network a TCN reaches a great increase of the receptive field per layer, while every point of data inside this history is still reached by the filters. The usage of residual connections[9] further speeds up convergence. Instead of having to wait for the calculations of previous timesteps, a TCN can be parallelized just like normal CNN architectures. The flat structure and the usage of *ReLu* activations prevents vanishing gradients and speed up convergence. It has been shown that TCN architectures train substantially faster than comparable recurrent units like LSTM and lead to better results on a variety of tasks[3]. Due to their advantages over recurrent architectures, TCN units are used as main component for the network architecture of the pipeline.

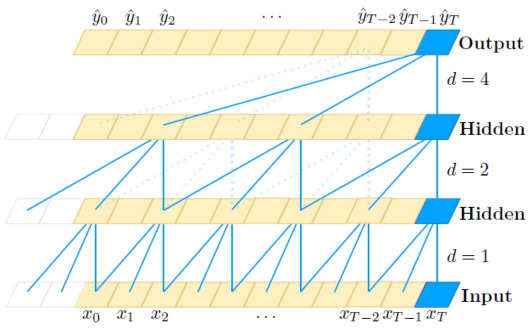
$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 ... x_{T-2} x_{T-1} x_T

Figure 3. A Dilated Causal Convolution with Dilation Factors 1, 2, 4 and filter size 3. Source: [3]

[1]Because the semantic is the same, LSTM/GRU and TCN units often can be simply swapped in network implemenations

C. Extracting features using Layers

Applying some Convolutional Layers before reccurent units can extract additional features and/or reduce dimensionality of the input sequence, which leads to increased performance[17].

Since the shape of input and output data of a TCN unit is the same as it is for LSTM units, simple replacing them by a TCN is possible. Therefore, the same approach can be adapted for any TCN based architecture as well.

IV. Data Augmentation: SpecAugment

Data augmentation techniques are an important strategy to combat overfitting and greatly improve general performance of a given model. These techniques alter the training data for example by random rotating, zooming or shifting. This way, data scientists are able to artificially create more samples for training without having to collect more labeled data. Additionally, the model can get robust against the altered parameters. For audio, traditional approaches include adding noise, pitch shifting or stretching of the audio signal before calculating the features.

However, researchers of Google Brain recently proposed a novel method that directly modifies spectrograms instead of raw audio. They have shown that by this method, the performance of their speech recognition model drastically improved. Even without a language model, their network was able to outperform previous networks that did have a language model[4].

In SpecAugment, a log mel spectogram is viewed as an image with τ time steps with a horizontal time axis and a vertical frequency axis.

The algorithm performs three main steps:

- Time Warping: At the horizontal line that passes through the image center within the time steps $W(\tau - W)$ gets warped by a distance w to the left or to the right.
    - w is chosen from a uniform distribution from 0 to time warp parameter W along the line.
- Frequency Masking: f consecutive mel frequency channels from interval $[f_0, f_0 + f)$ get masked.
    - f is chosen from a uniform distribution from 0 to frequency mask parameter F
    - f_0 is chosen from interval $[0, -f]$ where is the number of mel frequency channels
- Time Masking: Consecutive time steps $[t_0, t_0 + t]$ are masked.
    - t is the numbers of time steps to be masked and is chosen from a uniform distribution from 0 to time masking parameter T.
    - t_0 is chosen from interval $[0, \tau - t)$
    - Upper Bound p: a time mask cannot be wider than p times the number of time steps.

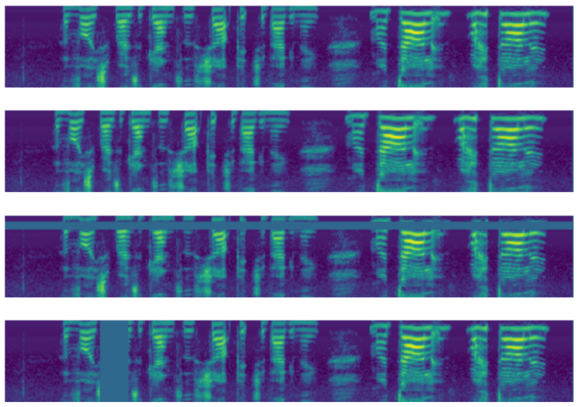

Figure 4. From top to bottom: No augmentation, time warping, frequency masking, time masking. Source: [4]

Furthermore, the SpecAugment paper introduced hand-crafted policies depending on specific datasets. In those policies, multiple frequency and time masks can be applied which can overlap.

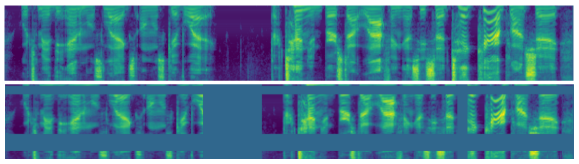

Figure 5. Top: No augmentation. Bottom: 'Libri Speech double' policy with parameters $W = 80, F = 27, T = 100, p = 1.0, m_F = 2, m_T = 2$ where m_F is the number of frequency masks and m_T is the number of time masks. Source: [4]

V. Pipeline

The previously introduced components can now be tied together to an end-to-end pipeline. The first step is the calculation of mel spectrograms (see II) for the entire dataset, which is only neccessary once. After that, the data is split into train and test sets.

The network architecture (see III) consists of an Input Layer, some Convolutional Layers for additional feature extraction and a single or stacked TCN components. The number of layers, convolutional filters and other parameters of the previous layers can easily be varied depending on task and dataset. If the same dimensionality should be maintained, zero padding is performed at the layers before the TCN. For classification tasks, a fully connected output layer can be added where the output dimensionality is the number of classes.

The training is performed via backpropagation over n epochs, where an epoch is one iteration over the entire training dataset. For each batch of data during training, data augmentation (see IV) can be applied in real time. After each epoch, the current state is then evaluated on the test dataset.

A. Implementation

An example implementation of the proposed pipeline is available[2]. This implemantation features the training of the pipeline on the UrbanSound8K dataset[18] which contains 8732 samples of sound clips ($<=$ 4 seconds) in 10 mutually exclusive classes.

VI. Conclusion

The introduced approach can not only be used for classification, but be modified for different tasks like audio segmentation or speech recognition. Additionally, it allows for small network sizes that can be utilized even on mobile devices due to low memory requirements. The pipeline can be further enhanced depending on the specific task - for example adding a language model for speech recognition.

This work provides an extensible and flexible approach for audio recognition which can be adapted for many use-cases as well as an overview about some state-of-the-art methods.

Acknowledgment

Special thanks goes to Lars Engeln for giving some writing advices in order to polish this work.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. NIPS’12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105. url: http://dl.acm.org/citation.cfm?id=2999134.2999257.

[2] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: Neural Comput. 9.8 (Nov. 1997), pp. 1735–1780. issn: 0899-7667. doi: 10.1162/neco.1997.9.8.1735. url: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling”. In: (2018). arXiv: 1803.01271 [cs.LG].

[4] Daniel S Park et al. “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition”. In: arXiv preprint arXiv:1904.08779 (2019).

[5] Wei Dai et al. “Very deep convolutional neural networks for raw waveforms”. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Mar. 2017). doi: 10.1109/icassp.2017.7952190. url: http://dx.doi.org/10.1109/ICASSP.2017.7952190.

[6] Siwat Suksri and Thaweesak Yingthawornsuk. “Speech Recognition using MFCC”. In: 2012.

[7] Hendrik Purwins et al. “Deep Learning for Audio Signal Processing”. In: IEEE Journal of Selected Topics in Signal Processing (2019), pp. 1–1. issn: 1941-0484. doi: 10.1109/jstsp.2019.2908700. url: http://dx.doi.org/10.1109/JSTSP.2019.2908700.

[8] Christian Szegedy et al. “Going deeper with convolutions”. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015). doi: 10.1109/cvpr.2015.7298594. url: http://dx.doi.org/10.1109/CVPR.2015.7298594.

[9] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016). doi: 10.1109/cvpr.2016.90. url: http://dx.doi.org/10.1109/CVPR.2016.90.

[10] Kaiming He et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: 2015 IEEE International Conference on Computer Vision (ICCV) (Dec. 2015). doi: 10.1109/iccv.2015.123. url: http://dx.doi.org/10.1109/ICCV.2015.123.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 2014. arXiv: 1412.6980 [cs.LG].

[12] Shawn Hershey et al. “CNN architectures for large-scale audio classification”. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Mar. 2017). doi: 10.1109/icassp.2017.7952132. url: http://dx.doi.org/10.1109/ICASSP.2017.7952132.

[13] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: CVPR09. 2009.

[14] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning. 2015. arXiv: 1506.00019 [cs.LG].

[15] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014). doi: 10.3115/v1/d14-1179. url: http://dx.doi.org/10.3115/v1/D14-1179.

[16] Aaron van den Oord et al. “WaveNet: A Generative Model for Raw Audio”. In: (2016). arXiv: 1609.03499 [cs.SD].

[17] T. N. Sainath et al. “Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks”. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Apr. 2015, pp. 4580–4584. doi: 10.1109/ICASSP.2015.7178838.

[18] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. “A Dataset and Taxonomy for Urban Sound Research”. In: Proceedings of the 22Nd ACM International Conference on Multimedia. MM ’14. Orlando, Florida, USA: ACM, 2014, pp. 1041–1044. isbn: 978-1-4503-3063-3. doi: 10.1145/2647868.2655045. url: http://doi.acm.org/10.1145/2647868.2655045.

[2]https://colab.research.google.com/drive/1wflMtZzj3wXiXYrH7Y3vR1m4fdf5TJyD