VERY DEEP CONVOLUTIONAL NEURAL NETWORKS FOR RAW WAVEFORMS

Wei Dai, Chia Dai*, Shuhui Qu, Juncheng Li, Samarjit Das*

{wdai,chiad}@cs.cmu.edu, shuhuiq@stanford.edu, {billy.li,samarjit.das}@us.bosch.com

ABSTRACT

Learning acoustic models directly from the raw waveform data with minimal processing is challenging. Current waveform-based models have generally used very few (∼2) convolutional layers, which might be insufficient for building high-level discriminative features. In this work, we propose very deep convolutional neural networks (CNNs) that directly use time-domain waveforms as inputs. Our CNNs, with up to 34 weight layers, are efficient to optimize over very long sequences (e.g., vector of size 32000), necessary for processing acoustic waveforms. This is achieved through batch normalization, residual learning, and a careful design of down-sampling in the initial layers. Our networks are fully convolutional, without the use of fully connected layers and dropout, to maximize representation learning. We use a large receptive field in the first convolutional layer to mimic bandpass filters, but very small receptive fields subsequently to control the model capacity. We demonstrate the performance gains with the deeper models. Our evaluation shows that the CNN with 18 weight layers outperform the CNN with 3 weight layers by over 15% in absolute accuracy for an environmental sound recognition task and matches the performance of models using log-mel features.

Index Terms— Convolutional Neural Networks, Raw Waveform, Acoustic Modeling, Neural Networks, Environmental Sound

1. INTRODUCTION

Acoustic modeling is traditionally divided into two parts: (1) designing a feature representation of the audio data, and (2) building a suitable predictive model based on the representation. However, it is often challenging and time-intensive to find the right representation in the so-called "feature-engineering" process, and the often heuristically designed features might not be optimal for the predictive task. Deep neural networks, which have achieved state-of-the-art performances in acoustic scene recognition [1] and speech recognition [2], have increasingly blurred the line between representation learning and predictive modeling. Instead of using the hand-tuned Gaussian Mixture Model features and Mel-frequency cepstrum coefficients, neural network models can directly take as input features such as spectrograms [2] and even raw waveforms [3]. By using simpler features, deep neural networks can be viewed as extracting feature representation *jointly* with classification, rather than separately [4]. This joint optimization is highly effective in speech recognition [2] and image classification [5], among others.

A fundamental building block of these models is the convolutional neural networks (CNNs), which can learn spatially or temporally invariant features from pixels or time-domain waveforms. CNNs have famously achieved performance competitive or even surpassing human-level performance in the visual domains, such as object recognition [6] and face recognition [7, 8]. A common theme among these powerful CNN models is that they are usually very deep, with the number of layers ranging from tens to even over a hundred. Nonetheless, designing and training a deep network suitable for a new application domain remain challenging.

Recent works have applied CNNs to audio tasks such as environmental sound recognition and speech recognition and found that CNNs perform well with just the raw waveforms [9, 4, 10]. In one case, CNNs with time-domain waveforms can match the performance of models using conventional features like log-mel features [4]. These works, however, have mostly considered only less deep networks, such as two convolutional layers [4, 11].

In this work, we propose and study very deep architectures with up to 34 weight layers, directly using time-series waveforms as the input. Our deep networks are efficient to optimize over long sequences (e.g., vector of length 32000), necessary for processing raw audio waveforms. Our architectures use a very small receptive field in the convolutional layers, but a large receptive field in the first layer chosen based on the audio sampling rate to mimic bandpass filter. Our models are fully convolutional, without fully connected layers and dropout, in order to maximize the representation learning in the convolutional layers and can be applied to audio of varying lengths. By applying batch normalization [12], residual learning [6], and a careful design of down-sampling layers, we overcome the difficulties in training very deep models while keeping the computation cost low.

On an environmental sound recognition task [13], we show that deep networks improve the performance of networks with 2 convolutional layers by over 15% in absolute accuracy. We further demonstrate that the performance of deep models using just the raw signal is competitive with models using log-mel features [11]. To our knowledge, this

*These authors contributed equally.

is the first report of a parity performance between log-mel features and raw time signal for environmental sound recognition.

2. VERY DEEP CONVOLUTIONAL NETWORKS

Table 1 outlines the 5 architectures we consider. Our architectures take as input time-series waveforms, represented as a long 1D vector, instead of hand-tuned features or specially designed spectrograms. Key design elements are:

Deep architectures. To build very deep networks, we use very small receptive field 3 for all but the first 1D convolutional layers[1]. This reduces the number of parameters in each layer and control the model sizes and computation cost as we go deeper. Furthermore, we aggressively reduce the temporal resolution in the first two layers by 16x with large convolutional and max pooling strides to limit the computation cost in the rest of the network [16]. After the first two layers, the reduction of resolution is complemented by a doubling in the number of feature maps[2]. We use rectified linear units (ReLU) for lower computation cost, following [17, 15].

Fully convolutional networks. Most deep convolutional networks for classification use 2 or more fully connected (FC) layers of high dimensions (e.g., 4096 in [15, 5]) for discriminative modeling, leading to a very high number of parameters. We hypothesize that most of the learning occurs in the convolutional layers, and with a sufficiently expressive representation from convolutional layers, no FC layer is necessary. We therefore adopt a *fully convolutional* design for our network construction [6, 18]. Instead of FC layers, we use a single global average pooling layer which reduces each feature map into one float by averaging the activation across the temporal dimension. By removing FC layers, the network is forced to learn good representation in the convolutional layers, potentially leading to better generalization. We support this design decision in our evaluation and demonstrate that fully convolutional networks perform comparably or better compared with their counterparts endowed with FC layers.

First layer receptive field. Time-domain waveforms at a reasonable sampling rate (e.g. 8000Hz) over a few seconds could have very large number of samples along a single dimension. If we exclusively use small receptive field for all convolutional layers such as in [15], which uses 3x3 in pixel for all layers, our model would need many layers in order to abstract high level features, which could be computationally expensive. Furthermore, audio sampling rate could affect the receptive field size in the first layer, since a field size of 80 at 8kHz sampling rate is at a different length scale than at 16kHz sampling rate. We thus choose our first layer receptive field to cover a 10-millisecond duration, which is similar to the window size for many MFCC computation. In Section 3 we show that a much smaller or larger receptive field gives poor performance.

[1] Small receptive fields were first popularized by [15] for 2D images.

[2] In the visual domain this change in resolution and the number of features maps leads to more specialized filters at the higher layers (e.g., feature maps responding to faces) and more basic filters at the bottom (e.g., feature maps responding diagonal lines).

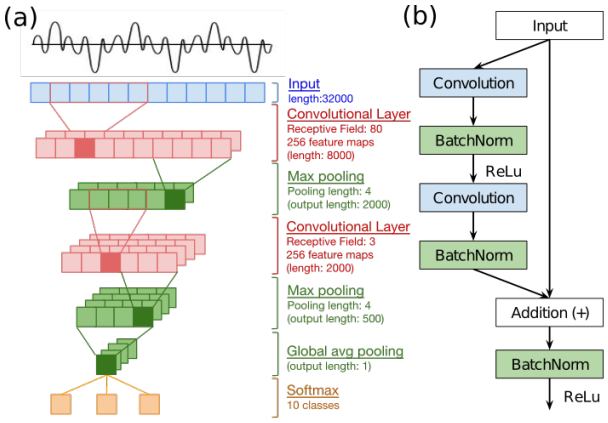

Fig. 1: (a) The model architecture of M3 (Table 1). The input audio is represented by a single feature map (or channel). In each convolutional layer a feature map encodes activity level of the associated convolutional kernel. Note that the number of feature maps doubles as temporal resolution decreases by factor of 4 in the max pooling layers, capped by a global average pooling. Note that a reduction by factor of 4 in our max pooling layers is equal to a 2D max pooling with stride (2x2) used in many vision networks. (b) Residual block (res-block) used in M34-res. A resblock consists of two convolution layers.

Batch Normalization. We adopt auxiliary layers called batch normalization (BN) [12] that alleviates the problem of exploding and vanishing gradients, a common problem in optimizing deep architectures. BN normalizes the output of the previous layer so the gradients are well-behaved. This makes possible training very deep networks (M18, M34-res) that were not studied previously [19]. Following [12], we apply BN on the output of each convolutional layer before applying ReLU non-linearity.

Residual Learning. Residual learning [6] is a recently proposed learning framework to ease the training of very deep networks. Normally we train a block of neural network layers to fit a desired mapping $\mathcal{H}(\boldsymbol{x})$ of $\boldsymbol{x}$ ($\boldsymbol{x}$ being the the input to the layers). In the residual framework, we instead let the block of layers approximate $\mathcal{F}(\boldsymbol{x}) = \mathcal{H}(\boldsymbol{x}) - \boldsymbol{x}$, the residual mapping. Residual learning is achieved through a skip connection in the residual block ("res-block", Figure 1b). We apply residual learning in M34-res (Table 1).

3. EXPERIMENT DETAILS

We use UrbanSound8k dataset which contains 10 environmental sounds in urban areas, such as drilling, car horn, and children playing [13]. The dataset consists of 8732 audio clips of 4 seconds or less, totalling 9.7 hours. We use the official fold 10 to be our test set, and the rest for training and validation. For computational speed, the audio waveforms are down-sampled to 8kHz and standardized to 0 mean and variance 1. We shuffle the training data but do not perform data augmentation.

We train the CNN models using Adam [20], a variant of

M3 (0.2M)	M5 (0.5M)	M11 (1.8M)	M18 (3.7M)	M34-res (4M)
Input: 32000x1 time-domain waveform				
[80/4, 256]	[80/4, 128]	[80/4, 64]	[80/4, 64]	[80/4, 48]
Maxpool: 4x1 (output: 2000 × n)				
[3, 256]	[3, 128]	[3, 64] × 2	[3, 64] × 4	$\begin{bmatrix} 3, 48 \\ 3, 48 \end{bmatrix} \times 3$
Maxpool: 4x1 (output: 500×n)				
	[3, 256]	[3, 128] × 2	[3, 128] × 4	$\begin{bmatrix} 3, 96 \\ 3, 96 \end{bmatrix} \times 4$
	Maxpool: 4x1 (output: 125 × n)			
	[3, 512]	[3, 256] × 3	[3, 256] × 4	$\begin{bmatrix} 3, 192 \\ 3, 192 \end{bmatrix} \times 6$
	Maxpool: 4x1 (output: 32 × n)			
		[3, 512] × 2	[3, 512] × 4	$\begin{bmatrix} 3, 384 \\ 3, 384 \end{bmatrix} \times 3$
Global average pooling (output: 1 × n)				
Softmax				

Table 1: Architectures of proposed fully convolutional network for time-domain waveform inputs. M3 (0.2M) denotes 3 weight layers and 0.2M parameters. [80/4, 256] denotes a convolutional layer with receptive field 80 and 256 filters, with stride 4. Stride is omitted for stride 1 (e.g., [3, 256] has stride 1). [...] $\times k$ denotes k stacked layers. Double layers in a bracket denotes a residual block and only occur in M34-res. Output size after each pooling is written as $m \times n$ where m is the size in time-domain and n is the number of feature maps and can vary across architectures. All convolutional layers are followed by batch normalization layers, which are omitted to avoid clutter. Without fully connected layers, we do not use dropout [14] in these architectures.

stochastic gradient descent that adaptively tunes the step size for each dimension. We run each model for 100-400 epochs (defined as a pass over the training set) until convergence. The weights in each model are initialized from scratch without any pretrained model. We use glorot initialization [21] to avoid exploding or vanishing gradients. All weight parameters are subjected to ℓ_2 regularization with coefficient 0.0001. Our models are implemented in Tensorflow [22] and trained on machines equipped with a Titan X GPU.

Additional Models. To aid analysis, we train variants of models in Table 1. The "fc" models replace global average pooling layer with 2 fully connected (FC) layers of dimension 1000 (Table 5), since many conventional deep convolutional networks use 2 FC layers of dimension in the thousands [5, 15, 11]. Following these works we also use a dropout layer between each fully connected layers for regularization, with a dropout rate of 0.3. We insert a batch normalization layer after each fully connected layers to aid training. These models have substantially more parameters than the original models due to the FC layers (Table 5). Additionally, M3-big and M5-big (Table 4) are variants of M3 and M5, respectively, with 50% and 100% more filters (e.g., 384/256 filters in the first convolutional layer in M3-big/M5-big).

4. RESULTS AND ANALYSES

Table 2 shows the test accuracies and training time for models in Table 2. We first note that M3 perform very poorly compared with the other models, indicating that 2-layered CNNs are insufficient to extract discriminative features from raw waveforms for sound recognition. This is in contrast with models using the spectrogram as input, which achieve good performance with just 2 convolutional layers [11], and shows that applying CNN directly on time-series data is challenging. M3-big, a variant of M3 with 50% more filters and 2.5x more parameters, does not significantly improve the performance

Model	Test	Time
M3	56.12%	77s
M5	63.42%	63s
M11	69.07%	71s
M18	71.68%	98s
M34-res	63.47%	124s

Table 2: Test accuracies and training time per epoch (a sweep over the training set) for models in Table 1 on UrbanSound8k dataset using a Titan X GPU.

(Table 4), showing that shallow models have limited capacity to capture time-series inputs even with a larger model.

Deeper networks (M5, M11, M18, M34-res) substantially improve the performance. The test accuracy improves with increasing network depth for M5, M11, and M18. Our best model M18 reaches 71.68% accuracy that is competitive with the reported test accuracy of CNNs on spectrogram input using the same dataset [11][3]. The performance increases cannot be simply attributed to the larger number of parameters in the deep models. For example, M5-big has 2.2M parameters (Table 4) but only achieves 63.30% accuracy, compared with the 69.07% by M11 (1.8M parameters). By using a very deep architecture, M18 outperforms M3 by as much as 15.56% in absolute accuracy, which shows that deeper architectures substantially improve acoustic modeling using waveforms. Furthermore, by using an aggressive down-sampling in the initial layers, very deep networks can be economical to train (Table 2 Time column). When we use stride 1 instead of 4 in the first convolutional layer for M11, we observe a 3.5x increase in training time but a lower test accuracy (67.37%) af-

[3]Figure 4 in [11] reports ∼68% accuracy using a baseline CNN model. We point out that we have a different evaluation scheme: we use the 10-th fold as test set, while [11] performs 10-fold evaluation. Also we use sound at 8kHz sampling rate while they use the original 44.1kHz.

Model	Test
M11-srf	64.78%
M18-srf	65.55%
M11-lrf	65.67%
M18-lrf	65.08%

Table 3: Test accuracies for M11 and M18 variants different receptive field in the first convolutional layer. M11-srf and M18-srf have receptive field 8; M11-lrf and M18-lrf have 320.

Model	Test	# Parameters
M3-big	57.55%	0.5M
M5-big	63.30%	2.2M

Table 4: Test accuracies for M3, M5 variants with more filters in the convolutional layers. M3-big, M5-big have 50% and 100% more filters (384 and 256 filters in the first layers, respectively).

ter 10 hours of training, compared with 68.42% test accuracy reached in 2 hours by M18.

Interestingly, the performance improves with depth up to M18, at 71.68% test accuracy. M34-res only achieves 63.47% test accuracy. This is due to overfitting. We observe that with residual learning we have no problem optimizing deep networks like M34-res, and M34-res reaches an extremely high training accuracy of 99.21%, compared with 96.72% training accuracy by M18. We also observe overfitting in a residual variant of M11 network (not shown here) which reaches higher training accuracy but a lower test accuracy (by 0.17%). Overfitting caused by very deep networks is well documented [6]. We believe that our dataset is too small to train M34-res without further regularization. Nonetheless, M34-res still outperforms M3 and M5.

We compare our fully convolutional network with conventional networks that use large fully connected layers (FC) for classification. Table 5 shows that FC layers can increase number of parameters significantly and increase training time by 2∼95%. However, FC layers do not improve test accuracy, and in the cases of M3-fc and M11-fc the additional FC layers lead to lower test accuracy (i.e., poorer generalization). We believe that the lack of FC layers in our network design pushes learning down to convolutional layers, leading to better representation and generalization.

To understand the effect of the receptive field (RF) size in the first convolutional layer, we train M11-srf and M18-srf, variants of M11 and M18 with RF 8, and M11-lrf and M18-lrf with RF 320. Table 3 shows that the performance degrades significantly by up to 6.6% compared with M11 and M18 with RF 80. Previous works have shown that the first convolutional layer, when trained on raw waveforms, mimics wavelet transforms [9, 4]. Our results suggest that a small RF popularized by vision models is insufficient to capture the necessary bandpass filter characteristics in the first convolutional layer, while a large RF smooths out local structures and cannot effectively detect local impulse patterns.

We study the effect of batch normalization (BN) in optimizing very deep networks (Table 6). Without BN, both M11-no-bn and M18-no-bn can be optimized to high training

Model	Test	# Parameters	Time
M3-fc	46.82%	129M	150s
M5-fc	62.76%	18M	66s
M11-fc	68.29%	1.8M	73s
M18-fc	64.93%	8.7M	100s

Table 5: Test accuracy for models in Table 1 endowed with fully connected (FC) layers. Time is training time per epoch.

Model	Train	Test
M11-no-bn	98.58%	69.38%
M18-no-bn	99.33%	62.48%
M34-no-bn	10.96%	11.45%

Table 6: Test accuracies for models variants without batch normalization.

accuracy. Note that M18-no-bn results in lower test accuracy, indicating that BN has a regularization effect [12]. M34-no-bn could not be optimized without BN and performs close to random guess (10%) after 159 epochs of training.

Fig. 2 shows the learned kernels for M18 variants with different RF sizes in the first convolution layer. All of them learn a filter bank of bandpass filter. M18 (Fig. 2 left) has well-distributed filters. In contrast, the small RF model (Fig. 2 middle) has much more dispersed bands, and thus lower frequency resolution for subsequent layers. Conversely, large RF model (Fig. 2 right) has fine-grained filters, but does not have sufficient filters in the high frequency range, showing that it cannot effectively respond to local high frequency impulses.

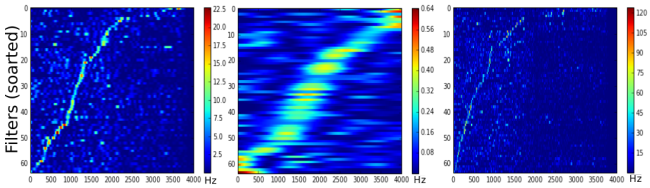

Fig. 2: Kernels of the first convolutional layer after Fourier transformation, sorted by activation frequencies. Left: M18. Middle: M18-srf (small receptive field). Right: M18-lrf (large receptive field).

5. CONCLUSION

In this work, we propose very deep convolutional neural networks that operate directly on acoustic waveform inputs. Our networks, up to 34 weight layers, are efficient to optimize, thanks to the combination of batch normalization, residual learning, and down-sampling. We use a broad receptive field (RF) in the first convolutional layer and narrow RFs in the rest of the network. Our results show that a deep network with 18 weight layers outperforms networks with 2 convolutional layers by 15.56% accuracy absolutely and achieves 71.8% accuracy, competitive with CNNs using log-mel spectrogram inputs [11]. Our fully convolutional networks compare favorably with those with fully connected layers. Our proposed deep architectures hold the promise to improve CNNs for speech recognition and other time-series modeling.

6. ACKNOWLEDGEMENT

This work is supported by contract FA8702-15-D-0002 with Software Engineering Institute, a center sponsored by the United States Department of Defense.

7. REFERENCES

[1] Hamid Eghbal-Zadeh, Bernhard Lehner, Matthias Dorfer, and Gerhard Widmer, "CP-JKU submissions for DCASE-2016: a hybrid approach using binaural i-vectors and deep convolutional neural networks," Tech. Rep., DCASE2016 Challenge, September 2016.

[2] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al., "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[3] Yedid Hoshen, Ron J Weiss, and Kevin W Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *ICASSP*. IEEE, 2015, pp. 4624–4628.

[4] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Proc. Interspeech*, 2015.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, Eds., pp. 1106–1114. 2012.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[7] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[8] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015, pp. 815–823.

[9] Zoltán Tüske, Pavel Golik, Ralf Schlüter, and Hermann Ney, "Acoustic modeling with deep neural networks using raw time signal for lvcsr.," in *INTERSPEECH*, 2014, pp. 890–894.

[10] Pavel Golik, Zoltán Tüske, Ralf Schlüter, and Hermann Ney, "Convolutional neural networks for acoustic modeling of raw time signal in lvcsr," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[11] Karol J Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2015, pp. 1–6.

[12] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[13] 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*. ACM, 2014, pp. 1041–1044.

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, 2014.

[15] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] Christian et al Szegedy, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[17] Matthew D et al. Zeiler, "On rectified linear units for speech processing," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3517–3521.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[19] Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun, "Very deep multilingual convolutional neural networks for lvcsr," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4955–4959.

[20] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks.," in *Aistats*, 2010, vol. 9, pp. 249–256.

[22] Martın Abadi, Ashish Agarwal, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.