Acoustic Scene Recognition with Deep Learning

Wei Dai

Machine Learning Department Carnegie Mellon University

Abstract

Background. Sound complements visual inputs, and is an important modality for perceiving the environment. Increasingly, machines in various environments have the ability to hear, such as smartphones, autonomous robots, or security systems. This work applies state-of-the-art Deep Learning models that have revolutionized speech recognition to understanding general environmental sounds.
Aim. This work aims to classify 15 common indoor and outdoor locations using environmental sounds. We compare both conventional and Deep Learning models for this task.
Data. We use a dataset from the ongoing IEEE challenge on Detection and Classification of Acoustic Scenes and Events (DCASE). The dataset contains 15 diverse indoor and outdoor locations, such as buses, cafes, cars, city centers, forest paths, libraries, and trains, totaling 9.75 hours of audio recording.
Methods. We extract features using signal processing techniques, such as mel-frequency cepstral coefficients (MFCC), various statistical functionals, and spectrograms. We extract 4 feature sets: MFCCs (60-dimensional), Smile983 (983-dimensional), Smile6k (6573-dimensional), and spectrograms (only for CNN-based models). On these features we apply 5 models: Gaussian Mixture Models (GMMs), Support Vector Machines (SVMs), Deep Neural Networks (DNNs), Recurrent Neural Networks (RNNs), Recurrent Deep Neural Networks (RDNNs), Convolutional Neural Networks (CNNs), and Recurrent Convolutional Neural Networks (RCNNs). Among them GMMs and SVMs are popular conventional models for this task, while RDNNs, CNNs, and RCNNs are, to our knowledge, the first application of these models in the context of environmental sound.
Results. Our experiments show that model performance varies with features. With a small set of features (MFCCs and Smile983) temporal models (RNNs, RDNNs) outperform non-temporal models (GMMs, SVMs, DNNs). However, with large feature sets (Smile6k) DNNs outperform temporal models (RNNs and RDNNs) and achieve the best performance among all studied methods. The GMM with MFCC features, the baseline model provided by the DCASE contest, achieves 67.6% test accuracy, while the best performing model (a DNN with the Smile6k feature) reaches 80% test accuracy. RNNs and RDNNs generally have performance in the range of 68∼77%, while SVMs vary between 56∼73%. CNNs and RCNNs with spectrogram features lag in performance compared with other Deep Learning models, reaching 63∼64% accuracy.
Conclusions. We find that Deep Learning models compare favorably to conventional models (GMMs and SVMs). No single model outperforms all other models across all feature sets, showing that model performance varies significantly with the feature representation. The fact that the best performing model is a non-temporal DNN is evidence that environmental sounds do not exhibit strong temporal dynamics. This is consistent with our day-to-day experience that environmental sounds tend to be random and unpredictable.

1 Introduction

Recent developments in Deep Learning have brought significant improvements to automatic speech recognition (ASR) (Hannun et al. (2014)) and music characterization (Van den Oord et al. (2013)). However, speech is only one of many types of sounds, and humans often rely on a broad range of environmental sounds to detect danger and enhance scene understanding, such as when one crosses a busy street or navigates in a bustling office. More broadly, sound is a useful modality complementing visual information such as videos and images, with the advantage that audio can be more easily collected and stored.

Increasingly, machines in various environments can hear, such as smartphones, security systems, and autonomous robots. The prospect of human-like sound understanding could open up a range of applications, including intelligent monitoring system of equipment using acoustic information, acoustic surveillance, cataloging, and search in audio archives (Ranft (2004)).

This broad range of environmental sounds also poses different challenges than speech recognition. Compared with speech, environmental sounds are more diverse and span a wide range of frequencyies Moreover, they are often less well-defined. For example, there is no standard dictionary for environmental sound events analogous to sub-word dictionary phonemes in speech, and the duration of environmental sounds could vary widely. While sound analysis traditionally falls within the signal processing domain, recent advances in machine learning and Deep Learning hold the promise to improve upon existing signal processing methods.

In this work we focus on the task of acoustic scene identification, which aims to characterize the acoustic environment of an audio stream by selecting a semantic label for it. Existing work for this task largely uses conventional classifiers such as GMMs and SVMs, which do not have the feature abstraction found in deeper models. Furthermore, conventional models do not model temporal dynamics, but rely on feature extraction pipelines to capture local temporal dynamics. For example, the winning solution by Roma et al. (2013) for the last IEEE challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) challenge, extracts MFCC and temporal features using Recurrence Quantification Analysis over a short time window. The actual SVM classifier does not explicitly model temporal dynamics.

We apply state-of-the-art Deep Learning (DL) models to various feature representations generated from signal processing methods. Specifically, we use the following architectures: (1) Deep Neural Networks (DNNs); (2) Recurrent Neural Networks (RNNs); (3) Recurrent Deep Neural Networks (RDNNs); (4) Convolutional Neural Networks (CNNs); (5) Recurrent Convolutional Neural Networks (RCNNs). Additionally, we compare DL models with Gaussian Mixture Models (GMMs), and Support Vector Machines (SVMs). We also use several feature representations based on signal processing techniques: mel-frequency cepstral coefficients (MFCC), spectrograms, other conventional features such as the pitch, the energy, the zero-crossing rate, the mean crossing rate, etc. To our knowledge this is the first comprehensive study of a diverse set of deep architectures on acoustic scene recognition task, borrowing ideas from signal processing as well as recent advancements in automatic speech recognition.

We use a dataset from the currently ongoing IEEE challenge on Detection and Classification of Acoustic Scenes and Events (DCASE). The dataset contains 15 diverse indoor and outdoor locations, such as buses, cafes, cars, city centers, forest paths, libraries, trains, totaling 9.75 hours of audio recording (see section 3.1 for detail). Our system has entered the DCASE 2016 contest, which runs from February 8 to June 30, 2016, at which point the final ranking of the contestants will be announced.

2 Background

2.1 Signal Processing Pipeline

Finding an appropriate representation of the audio signal is a crucial step in acoustic scene recognition. We consider several feature extractions based on digital signal processing methods. Some of those, such as statistical functionals, provide more succinct representation than others, such as spectrograms, but could throw away more information. In our experiments we consider some combinations of these features.

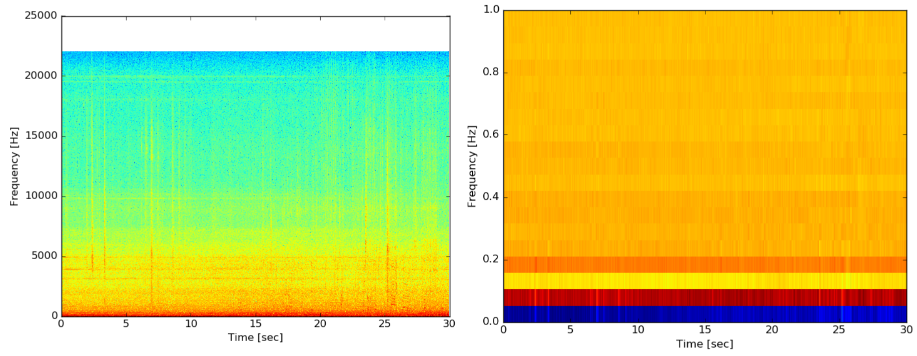

Figure 1: **Left**: A visualization of the spectrogram of a 30-second audio clip recorded in a grocery store. Red represents high intensity while blue is low. **Right**: a visualization of 20 mel-frequency cepstral coefficients (MFCC) from the same audio clip.

MFCCs: Mel-frequency cepstral coefficients uses the mel-frequency, which is a frequency scale based on human perception experiments. As pitch becomes higher, larger frequency gaps produce a perceptually similar difference in pitch, thus squashing the difference at high frequency (Hasan et al. (2004)). MFCCs are the coefficients obtained from the cepstral analysis of signals in mel-frequency.

Spectrogram: A spectrogram is a visual representation of discretized frequencies (spectrum) at each time interval. Figure 1 (left) is a spectrogram of a 30 second long recording in a grocery store. One has the option to tune the window shape of how to incorporate neighboring time intervals in calculating spectrum, as well as how fine-grained the spectra are.

Statistical Functionals: We employ OpenSmile by Eyben et al. (2010) to compute various conventional statistical features, such as: **Zero crossing rate**: Zero crossings are the points at which amplitude is 0. Zero crossing rate is the rate at which a signal switches between negative and positive. **FFT magnitude phase**: This is a magnitude-phase pairs representation of complex coefficients from Fourier transforms. **Energy**: The energy is the logarithm of energy in each frame. **Pitch**: The pitch computes the probability of human voicing and pitch using auto-correlation functions.

Temporal Dynamics: Based on previous features, we can compute velocity and acceleration features. One way is to simply take the first and second order differences between consecutive frames. Another way is to perform local smoothing, which is a common practice in conventional speech recognition systems such as used in Young et al. (1997). Given a 1-dimensional time series $\{x_1, ..., x_T\}$, we can compute d_t the smoothed first order difference at time t following

$$d_t = \frac{\sum_{i=1}^{W} i(x_{t+i} - x_{t-1})}{2\sum_{i=1}^{W} i^2} \tag{1}$$

where W is half of the window size used in smoothing. Unless otherwise stated, we use $W = 2$ throughout. The second order difference could be computed using Eq. 1, with input $\{d_1, ..., d_T\}$, the first order difference.

2.2 Gaussian Mixture Models (GMMs)

Gaussian Mixture Models (GMMs) have long been the model of choice for acoustic and speech modeling, as they are flexible (GMMs can approximate complex density by increasing the number of mixture components) and computationally efficient to train. GMMs have been used in speaker verification (Reynolds et al. (2000)), and in the form of GMM-based Hidden Markov Models (HMMs) to model the latent speaker state in speech recognition (Gales and Young (2008)).

A GMM models the distribution as a linear superposition of Gaussians:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where $\boldsymbol{x}, \boldsymbol{\mu}_k \in \mathbb{R}^d$, d is the input feature dimension, K is the number of mixtures, and $\{\pi_k\}$ lies on the $(K-1)$-simplex $(0 \leq \pi_k \leq 1, \sum_k \pi_k = 1)$, and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a Gaussian distribution with associated mean and covariance. GMMs can be efficiently trained using the Expectation-Maximization (EM) algorithm (Bishop (2006)).

Here we use GMMs as a baseline system for acoustic scene recognition. Each audio clip is represented as a bag of acoustic features extracted from audio segments, and for each class label (scene location) j, a GMM $p_j(\boldsymbol{x})$ is trained on this bag of acoustic features using only audio clips from class j. At the test time the test audio clips are processed to the same bag of n segments $i = 1...n$ as during the training time. We compute the class likelihood $p_j^i, j = 1...J$ for each class using the trained GMMs. $\hat{p}_j := \sum_i p_j^i$ gives the likelihood of class j for the test audio clip, and the class with the highest likelihood is the predicted label.

2.3 Support Vector Machines (SVMs)

SVMs are commonly used in acoustic modeling. In fact in the previous run of DCASE challenge in 2013, 8 out of 11 solutions involves SVM. We use Liblinear (Fan et al. (2008)), which implements binary linear SVM. Given an i.i.d. dataset $(\boldsymbol{x}_i, y_i)$, $y_i \in \{-1, 1\}$, $i = 1, ..., N$, it solves the following $L2$-SVM optimization objective:

$$\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N} \max(1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i, 0)^2 \tag{2}$$

where $C > 0$ is a tuning parameter, which we tune by cross validation. For a multi-class problem Liblinear uses the one-vs-rest scheme. Liblinear solves Eq. 2 using the primal trust region Newton method (Fan et al. (2008)).

2.4 Deep Neural Networks (DNNs)

Deep Neural Networks (DNNs), or the multi-layer perceptron, have recently been successfully applied to speech recognition and audio analysis and show superior performance than GMMs (Graves et al. (2013)). Whereas GMMs are trained with generative objective, DNNs are trained discriminatively and thus can be statistically more efficient. By building a deeper architecture, DNNs can also parameterize diverse nonlinearity more efficiently than GMMs which can only model nonlinearity by adding mixture components (Hinton et al. (2012)).

Figure 3 left shows an example feed-forward DNN. Each layer consists of a number of "neurons", represented by a circle, and is connected densely to the previous layer and the next layer. Except the input layer, all neurons are endowed with some non-linear transforms, which gives rise to the non-linearity. Common forms of transforms include sigmoid, tanh, rectified linear units (ReLU), and softmax. Here for intermediate layers we use rectified linear units (ReLU) (Zeiler et al. (2013)). We also tried parametric rectified linear unit (He et al. (2015)) and exponential linear units (ELU) (Clevert et al. (2015)) but did not find an improvement over ReLU. We optimize DNNs using stochastic gradient descent (SGD) with momentum and Nesterov momentum (Sutskever et al. (2013)), using cross entropy as the loss function.

2.4.1 Batch Normalization

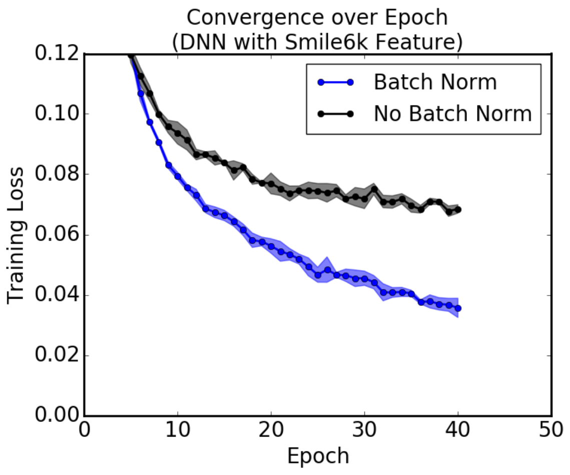

Figure 2: Convergence over epoch with and without Batch Normalization. With all other parameters fixed, training with Batch Normalization converges much faster and to a lower loss given the studied horizon.

Deeper architectures often suffer from vanishing gradients and are sensitive to the learning rate. This is due to the so-called the "internal covariate shift" (Ioffe and Szegedy (2015)). To understand this, we can view each layer of the DNN as some transform function, whose output becomes the input of the next layer. Mathematically,

$$\ell = F_2(F_1(u, \Theta_1), \Theta_2)$$

where F_1, F_2 represent, respectively, the first and second layer transform, and Θ_1, Θ_2 are the weights parameters associated with layer 1 and 2, respectively. As we perform gradient descent, both Θ_1 and Θ_2 are updated. Now, because Θ_1 is changing, the distribution of $F_1(u, \Theta_1)$, *i.e.* the input to F_2 (the second layer), is also changing, which is called *internal covariate shift.* Overall this slows down the convergence. The deeper the network is, the more manifest the internal covariate shift. Batch Normalization introduces a simple linear transform for each parameter to compensate for the covariate shift and thus reduces the distribution shift of the output of each layer. Empirically we observed that Batch Normalization significantly speeds up convergence per iteration for DNN training, reducing the number of iteration needed by a factor of 5$\sim$10. We apply Batch Normalization in all our Neural Network models.

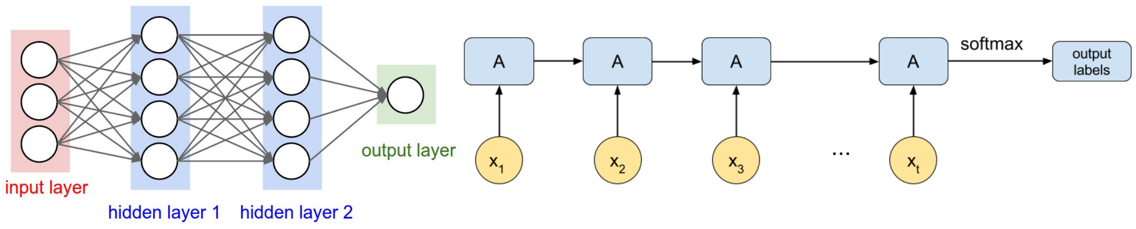

Figure 3: **Left**: A Deep Neural Network. It is a feed-forward network of 2 hidden layers. **Right**: An illustration of the Recurrent Neural Network.

2.5 Recurrent Neural Networks (RNNs)

The DNNs take as input a single i.i.d. instance, but cannot handle sequence data where the length of a training instance varies with linear (temporal) dependency. The Recurrent Neural Networks (RNNs) are suitable models as they explicitly account for the temporal dependencies with latent states. The RNNs have been shown to be excellent in tasks such as speech recognition (Hannun et al. (2014)), image captioning (Vinyals et al. (2015)), machine translation (Bahdanau et al. (2014)), and language modeling (Mikolov et al. (2011)). Here we use RNNs to model the temporal dynamics of adjacent audio segments extracted as described in section 2.1. Figure 3 (right) is a high-level representation of an RNN. At each

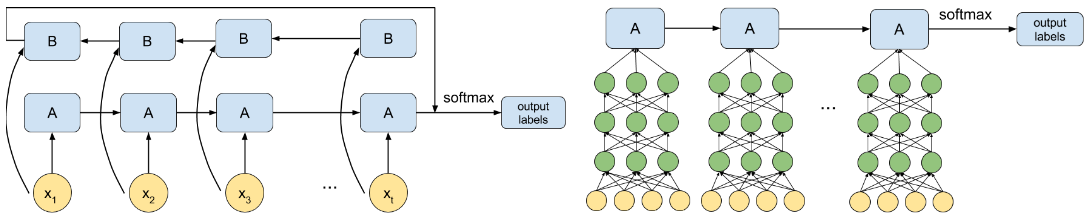

Figure 4: **Left**: Bidirectional Recurrent Neural Networks. The two layers are only connected at the softmax layer. **Right**: Recurrent Deep Neural Networks. To avoid clutter we only show one layer RNN, but our experiments use bi-directional RNN as shown in the left panel.

time point t, the recurrent block A takes as input latent activations from the previous time step h_{t-1}, the observation from the current time step x_t, and emits the next latent state h_t after some nonlinear transforms. Thus at a high level A is just a function that performs a (nonlinear) transform

$$f(x_t, h_{t-1}; \Theta) = h_t \tag{3}$$

where Θ denotes parameters associated with A.

However, even though in theory it should be able to represent arbitrarily complex temporal dependencies given sufficient number of neurons in Θ, it has been observed that this straightforward instantiation of RNNs in practice is unable to capture long-range dependencies (Bengio et al. (1994)). We therefore opt for a gated unit within recurrent block A proposed by Cho et al. (2014), called Gated Recurrent Unit (GRU), which is a variant of the popular "Long Short Term Memory" (LSTM) recurrent unit (Hochreiter and Schmidhuber (1997)). Both units are designed to maintain cell of long term memories (though GRU does not maintain those cells explicitly). Mathematically, Eq. (3) consists of the following operations within a GRU unit

$$\begin{aligned}
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \\
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) \\
\tilde{h}_t &= \tanh(W \cdot [r_t \circ h_{t-1}, x_t]) \\
h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t
\end{aligned}$$

where $\sigma(\cdot)$ is the sigmoid function, $\circ$ denotes element-wise product, and $[a, b]$ is the concatenation of vectors, and W, W_z, W_r are parameters to be learned. We use RMSProp (Hinton (2012)), a variant of gradient descent to learn the weights of GRU-RNN.

In our experiments we use bi-directional RNNs (Figure 4 left). Bi-directional RNNs provide the long-range context in both input directions, and is generally the preferred architecture in speech recognition (Hannun et al. (2014)). Through cross validation we also found that it performs as well or better than uni-directional RNNs.

2.6 Recurrent Deep Neural Networks (RDNNs)

The Recurrent Deep Neural Networks combines an RNN and a DNN (Figure 4). The idea is that the DNN can potentially extract useful feature representations more effectively than stacking RNNs, as stacked RNNs output only one hidden state to both the next time step as well as next layer, thus entangling the temporal embedding with the layer-wise embedding. From the RNN's perspective the DNN performs feature dimension reduction (or expansion if there are more neurons than the input dimension), while from the DNN's perspective it gains a temporal dependency. This architecture is inspired by speech recognition pipelines which connect a Convolutional Neural Network (CNN) with an RNN (Hannun et al. (2014)), and to our knowledge this is the first time this architecture has been explored on acoustic data. Previous works achieved depth in architecture only with stacked RNNs (Graves et al. (2013)). We train RNN+DNN jointly using RMSProp.

2.7 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have achieved extraordinary performance in the visual domain, sometimes even surpassing human-level performance. Examples include image classification (He et al. (2015)), face recognition (Taigman et al. (2014)), handwritten digit recognition, and recognizing traffic signs (Ciresan et al. (2012)). Lately, CNNs have been applied to speech recognition using spectrogram features (Hannun et al. (2014)) and achieve state-of-the-art speech recognition performance. We therefore apply CNNs to see if CNNs can extract useful acoustic features for our task.

Recent state-of-the-art CNN architecture suggests that it is advantageous to use small receptive fields to reduce the number of parameters at each layer and increase the number of layers. Our architecture design follows the VGG network (Simonyan and Zisserman (2014)) and uses 3×3 2D receptive fields. We found that, unlike the case of natural images, the pooling layer does not improve the performance on spectrogram features and is therefore not employed in our architecture. We train CNN using stochastic gradient descent with momentum and Nesterov momentum (Sutskever et al. (2013)).

2.8 Recurrent Convolutional Neural Networks (RCNNs)

Recurrent Convolution Neural Networks (RCNNs) have been used in video understanding such as scene labeling (Pinheiro and Collobert (2013)). Here instead of natural images we use an audio spectrogram. RCNNs have similar overall architecture as RDNNs (Figure 4 left), but with the DNN replaced by the CNN. Similar to RDNNs, RCNNs rely on the CNN to extract features from spectrogram, and uses RNN to model the temporal dynamics. We train RCNNs using stochastic gradient descent with momentum and Nesterov momentum.

3 Experiments

3.1 Dataset

We use a dataset from the currently ongoing IEEE challenge on Detection and Classification of Acoustic Scenes and Events (Mesaros (2016)). The dataset contains 15 diverse indoor and outdoor locations (labels), totaling 9.75 hours of recording and 8.7GB in wav format. The recording is 24-bit audio in 2 channels, with sampling rate at 44100 Hz, and each recording is 30 seconds long. The classes are:

- Bus: traveling by bus in the city (vehicle)
- Cafe or Restaurant: small cafe or restaurant (indoor)
- Car: driving or traveling as a passenger, in the city (vehicle)
- City center (outdoor)
- Forest path (outdoor)
- Grocery store: medium size grocery store (indoor)
- Home (indoor)
- Lakeside beach (outdoor)
- Library (indoor)
- Metro station (indoor)
- Office: multiple people, typical work day (indoor)
- Residential area (outdoor)
- Train (traveling, vehicle)
- Tram (traveling, vehicle)
- Urban park (outdoor)

There are 1170 30 second long audio clips. We use the evaluation split from the contest and withhold 290 audio clips for testing (25%), leaving 880 audio clips for training. Since there is more than one audio clip from each location, we make sure no one location appears in both training and testing sets. Note that this test set we use is part of the development set from the contest, not the final test set (which has not yet been released). Therefore our results might be different from the final contest result.

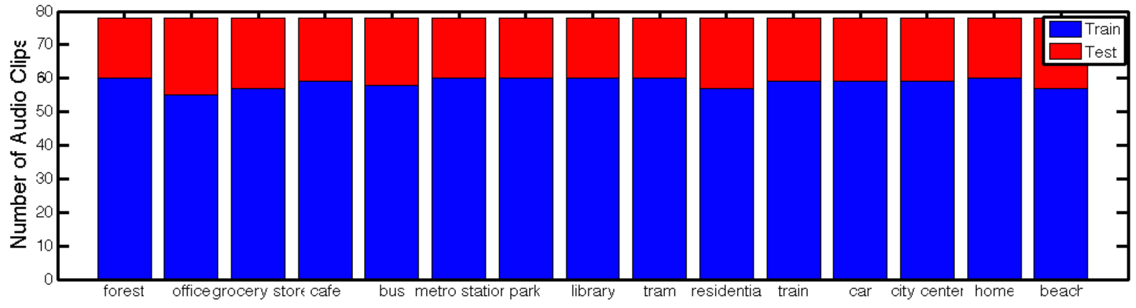

Figure 5: Class distribution of training and test data, which are fairly uniform across all classes. Each class has 18–21 test samples.

Within the 880 audio clip training set we perform 8-fold cross validation for model selection and parameter tuning, and again we made sure that no one location appears in both training set and validation set to have better generalization accuracy estimates.

3.2 Features

Using signal processing techniques (Section 2.1), we create 4 sets of features:

- **MFCCs**: We take 20 mel-frequency cepstral coefficients over windows of 0.04 second. We augment the feature with first and second order differences, resulting in a 60-dimensional vector.
- **Smile983**: We use OpenSmile (Eyben et al. (2010)) to generate MFCCs, Fourier transforms, the zero crossing rate, the energy, and the pitch, among others. We also compute first and second order dynamics features. After removing features with all zeros, this results in 6573 features. We select 983 features recommended by domain experts to create Smile983 feature (983 dimensional). Note this is much larger feature set than MFCC features and each feature represents a time windows of 0.1 second.
- **Smile6k**: This is the full 6573-dimensional feature set extracted from OpenSmile as described above.
- **Spectrograms**: We compute spectrograms (section 2.1) and truncate at frequency 750Hz (the recordings were recorded with up to 44100Hz). This is to speed up CNN computation by reducing the input size, and also most audible patterns have frequencies below 750Hz.

All features are standardized to have zero mean and unit standard deviation on the training set. The same standardization is applied at the validation and the test time.

For each audio clip (training or testing), our processing pipeline consists of the following: 1) Segment the audio clips into windows of 4$\sim$100ms segments. 2) Apply transforms to each audio clips to extract feature representation. 3) For non-temporal models such as GMMs, we treat each feature as a training example. For temporal models such as RNNs, we consider a sequence of features as one training example; 4) At the test time we apply the same pipeline as during training. The likelihood of a class label for a test audio clip is the sum of predicted class likelihood for each segment of a test example. The class with the highest predicted likelihood is the predicted label for the test audio clip.

3.3 Hyperparameter Tuning

Hyperparameter tuning is one of the most time-consuming aspects of using Deep Learning. Within each model class there are numerous important tuning knobs: the network architecture (the number of layers, the number of neurons in each layer, activation functions, dropout units), and optimization algorithms such as stochastic gradient descent, RMSProp (Hinton (2012)), Adam (Kingma and Ba (2014)) as well as numerical hyperparameters like the number of feature maps for the CNN, the L2 and L1 regularization parameters, the dropout rate, and the learning rate, among others. Generally, architectural changes are the most fundamental and could potentially have the largest impact on the results. Sometimes architectural changes could also speed up training, such as employing Batch Normalization layers (section 2.4.1). Overall we conducted over 500 cross validation experiments. Here we

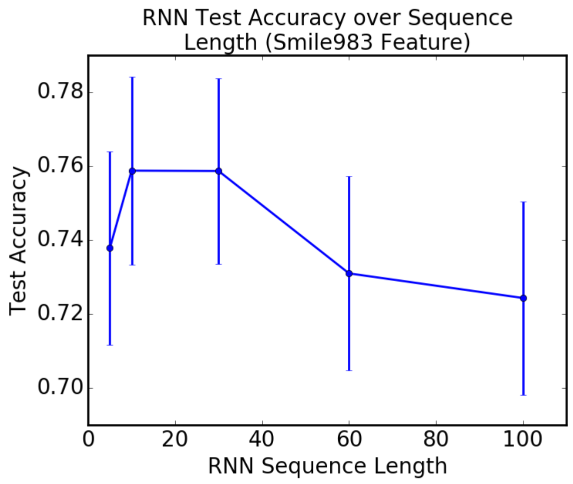

Figure 6: The effect of sequence length on RNNs using the Smile983 feature. There is a sweet spot around length 10–30, which corresponds to 1–3 seconds of audio segment. Error bars are standard deviation from bootstrap on the test set.

briefly discuss the important parameters for each model, and present the models we select from cross validation to use on the test data in Table 1 and 2.

DNN: The number of layers and the number of neurons are the most significant hyper-parameters. Dropout layers and the dropout rate also play a role. We found that adding a dropout layer after each layer works better than adding a dropout layer right before the output layer. In terms of activation functions, parametric rectified linear (PReLU) units do not perform as well as rectified linear units (ReLU), and exponential linear units (ELU) perform similarly to ReLU with Batch Normalization.

RNN: The bi-directional RNNs generally performs better than uni-directional counter-parts. We did not use LSTM (section 2.5) as LSTM empirically has similar model characteristics as GRU but is slower to train due to a more complex gating mechanism. We use only 2 layers (one direction per layer), as cross validation experiments show that deeper RNN networks beyond 2 layers are difficult to train to good accuracy. Figure 6 shows the effect of sequence length on the RNN performance, with other parameters fixed. It is interesting to note that the RNN performance actually degrades with too small or large sequence length, indicating that long-range temporal dynamics in the dataset is limited.

RDNN: We start with optimal architectures from the DNN and RNN experiments, and explore ways to reduce model capacity (since combining a DNN and an RNN increases the number of model parameters). This can be done by reducing the number of DNN layers or increasing L2 regularization strength, or by reducing the number of neurons.

CNN: We employ architectures similar to the VGG net (Simonyan and Zisserman (2014)) to keep the number of model parameters small. We use rectified linear units (ReLU) to model non-linearity. We also found that pooling layers do not help but only slow down computation, so we do not include them in most experiments. The dropout layer significantly improves performance, which is consistent with the CNN behaviors on natural images. Overall CNN takes a lot longer to train than RNNs, DNNs, and RDNNs due to the convolutional layers. We did not try 1D convolution layers as we believe 2D layers are more general and introduce very few additional parameters.

RCNN: We start with the well-tuned CNN model and bi-directional RNN. Due to the expensive training procedure for RCNN, we only tune the number of neurons in RNN.

Experiences from RDNN suggest that further tuning could have only limited benefit given a well tuned CNN component.

System Configuration: We train our Deep Learning models with the Keras library (Chollet (2015)) built on Theano (Bastien et al. (2012)), using 4 Titan X GPU on a 64GB memory, Intel Core i7 node.

3.4 Results

Figure 7 shows the test accuracy for 5 classifiers over 3 features. The model parameters are selected via cross validation and are detailed in Table 1. We perform 10000 bootstrap repetitions on the test set to estimate the standard deviations, and the difference between the test errors are statistically significant within each feature group using the unpaired T-test. We point out that the GMM with MFCC features is the official baseline provided in IEEE challenge on Detection and Classification of Acoustic Scenes and Events (DCASE), which achieves a mean accuracy of 67.6%, while our best performing model (DNN with the Smile6k feature) achieves a mean accuracy of 80%.

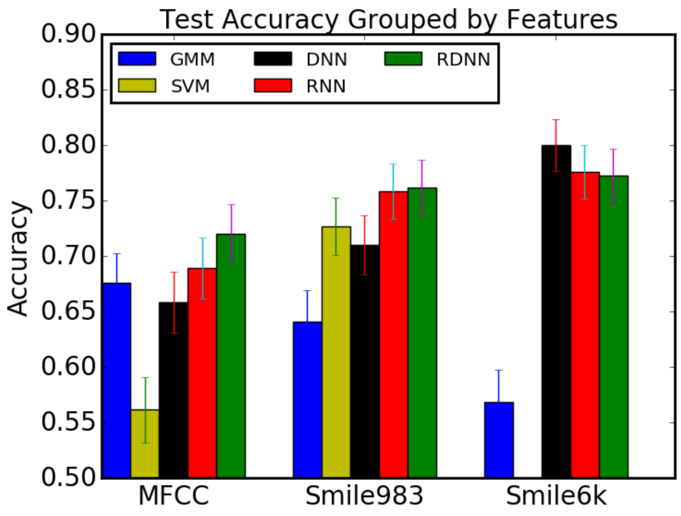

Figure 7: The test accuracies of GMMs, SVMs, RNNs, DNNs, and RDNNs on three features: MFCCs, Smile983, and Smile6k feature. The model parameter details are in Table 1. Except GMMs, all other models exhibit higher accuracy with increasing feature dimensions (dimensions of MFCC, Smile983, Smile6k features are, 60, 983, 6573, respectively). Also the DNNs outperform temporal models (RNNs, RDNNs) on the Smile6k feature, but on other features temporal models yield higher accuracies. Generative model GMMs suffer the curse of dimensionality when using higher dimensional features. All models perform above the chance performance (6.7%). Note that for the Smile6k feature Liblinear SVM could not finish computation in a reasonable amount of time ($\sim$12 hours) and thus was not included.

In addition to the 5 models in Figure 7, CNN using spectrogram features achieves test accuracy of 0.6377. RCNN has test accuracy of 0.641.

Figure 8 is the confusion matrix of the test result between 15 classes using the DNN with the Smile6k feature, which is the best performing setting from Figure 7.

3.5 Discussion

Figure 7 shows that feature representation is critical for classifier performance. For each Neural Networks models (RNNs, DNNs, RDNNs) higher dimensional features extracted

Feature	Model	# of layers	model size	Description
MFCC	GMM		2.5k	20 mixture components, diagonal covariance matrix
	SVM		61	$C = 1$
	RNN	4	50k	2 layers of GRU in opposite directions (bi-directional) with 64 units (neurons) each, no regularization, 1 Batch Normalization layer, 1 softmax layer. RNN sequence length is 10 (10 frames in a sequence).
	DNN	16	1.1M	5 hidden dense layers (512 units, regularized with $L2 = 0.1$), 5 dropout layers (0.2 dropout rate), 5 Batch Normalization layers, and 1 softmax layer.
	RDNN	19	2.7M	5 hidden dense layers (256 units, regularized with $L2 = 0.1$), 5 dropout layers (0.25 dropout rate), 5 Batch Normalization layers, and 2 RNN layers in opposite directions (bi-directional GRU) with 512 units each, regularized by $L2 = 0.1$, 1 Batch Normalization layer, and 1 softmax layer. RNN sequence length is 10.
Smile983	GMM		40k	20 mixture components, diagonal covariance matrix
	SVM		984	$C = 0.1$
	RNN	4	4.6M	2 layers of GRU in opposite directions (bi-directional) with 512 units (neurons) each, $L2 = 0.01$, 1 Batch Normalization layer, and 1 softmax layer. RNN length is 30.
	DNN	10	1M	3 hidden dense layers (512 units, regularized with $L2 = 0.01$), 3 dropout layers (0.2 dropout rate), 3 Batch Normalization layers, and 1 softmax layer.
	RDNN	13	4.2M	3 hidden dense layers (512 units, regularized with $L2 = 0.01$), 3 dropout layers (0.25 dropout rate), 3 Batch Normalization layers, and 2 RNN layers in opposite directions (bi-directional GRU) with 512 units each, regularized by $L2 = 0.01$, 1 Batch Normalization layer, and 1 softmax layer.
Smile6k	GMM		262k	25 mixture components, diagonal covariance matrix
	DNN	16	4.4M	5 hidden dense layers (256 units, regularized with $L2 = 0.1$), 5 dropout layers (0.2 dropout rate), 5 Batch Normalization layers, and 1 softmax layer.
	RNN	4	2.6M	bi-directional GRU with 64 units (neurons), regularized with $L2 = 0.01$. 1 Batch Normalization Layer, and 1 softmax layer. RNN length is 10 (10 frames in a sequence).
	RDNN	13	2.6M	3 hidden dense layers (256 units, regularized with $L2 = 0.2$), 3 dropout layers (0.25 dropout rate), 3 Batch Normalization layers, and 2 RNN layers in opposite directions (bi-directional GRU) with 256 units each, regularized by $L2 = 0.2$, 1 Batch Normalization, 1 softmax layer. RNN sequence length is 10.

Table 1: Models selected from cross validation for each model-feature combination. Model size denotes the number of model parameters.

Feature	Model	# of layers	model size	Description
Spectrogram	CNN	12	218k	4 convolutional layers (32 feature maps each), 4 Batch Normalization layers, 1 fully connected layer, 1 Batch Normalization layer, 1 dropout layers (0.5 dropout rate), and 1 softmax layer. (Not counting padding layers)
	RCNN	14	1.67M	4 convolutional layers (32 feature maps each), 4 Batch Normalization layer, 2 RNN layers (64 neurons, in opposite directions), 1 fully connected layer, 1 Batch Normalization layer, 1 dropout layer (dropout rate 0.2), and 1 softmax layer.

Table 2: CNN and RCN selected from cross validation using spectrogram feature.

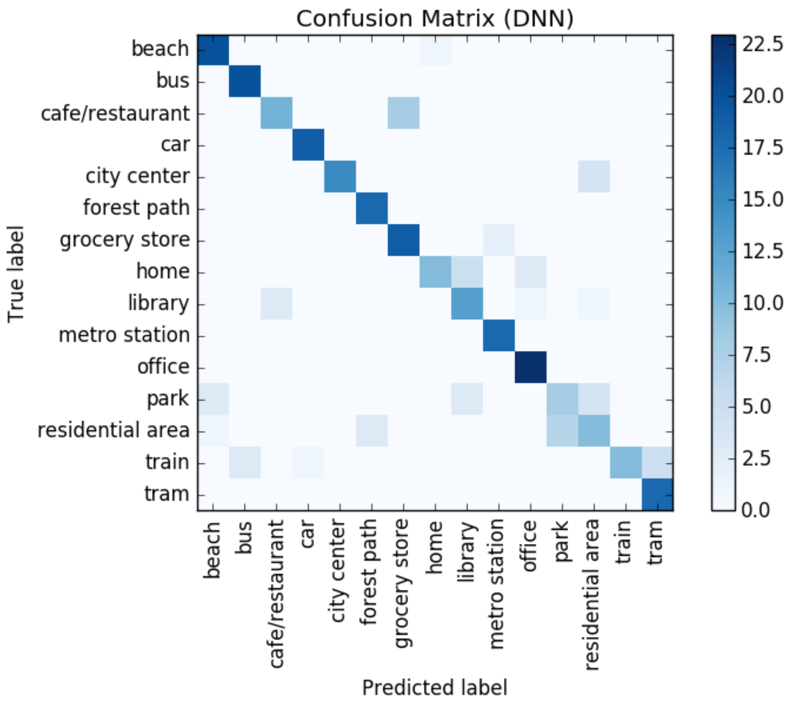

Figure 8: Confusion Matrix of the test results using the DNN with the Smile6k feature (the top performing setting). A perfect classification would result in only diagonal entries, in which the predicted labels (x-axis) agree with the ground truths (y-axis). The darker the off-diagonal entries, the more challenging it is for the classifier to distinguish the two classes. Note that each class has 18–21 test samples, and the color gradient spans this (unnormalized) range.

from signal processing pipeline improves performance. Among the Neural Network models, it is interesting to note that temporal models (RNNs and RDNNs) outperform DNNs using MFCC and Smile983 features, but DNNs outperform RNNs and RDNNs on Smile6k features and achieve the best accuracy among all models. It is possible that with the limited feature representation (e.g., MFCC and Smile983 features), modeling temporally adjacent pieces enhances local feature representation and thus improves performance in those cases. However, with sufficiently expressive features (e.g., Smile6k), temporal modeling becomes less important. It becomes more effective to model local dynamics rather than long-range dependencies.

This observation is somewhat surprising as we originally expected temporal models to outperform static models (e.g. DNNs) because sound is time-series data. A more careful consideration reveals that, unlike speech, which has long-range dependencies (a sentence utterance could span 6∼20 seconds), environmental sounds generally lack a coherent context, as events in the environment occur more or less randomly from the listener's perspective. To put it another way, a human listener of environmental noise is unlikely able to predict what sound will occur next. (Even though there are speeches and chatters in the environment, it is the presence of the speech rather than the content of the speech that is instrumental for our task.) This weak global dependency property in a time series data is not unique to this problem setting. Kim et al. (2015) made a similar observation in the context of facial expression synthesis based on speech. They find that even though the facial motion is temporal, it is more beneficial to simply model the local dynamics with the decision tree, which outperforms HMM, LSTM, and other temporal models. Another example is edge detection in the image. While different parts of an image could be related to each other, Dollár and Zitnick (2013) shows that it is more beneficial to model local patch dynamics than to consider the picture as a whole in performing edge detection.

The performance of CNNs and RCNNs using spectrogram features is lower than all other Neural Network models. This is somewhat unexpected as the state-of-the-art speech pipeline's acoustic model uses spectrogram directly as input (Hannun et al. (2014)). We hypothesize that the main reason for the poor performance of CNN-based models is due to the small data size. In Hannun et al. (2014) they train on thousands of hours of labeled data, which is further augmented with background noise. It is likely that there is simply not enough data for the CNN to learn good features for it to be competitive with well-tuned hand-crafted signal processing features.

Regarding the non-Neural Network models, the performance of GMMs decreases with increasing feature dimensions, which is expected due to the "curse of dimensionality". That is, in high dimensional space the volume grows exponentially while the number of available data stays constant, leading to highly sparse sample. SVM's performance is very poor with MFCC features, as the linear SVM has a limited model capacity with low dimensional features. By increasing the feature dimension and using the Smile983 features SVM performance improves.

Finally, the confusion matrix in Figure 8 shows that many locations are relatively easy to identify, such as the beach, the bus, and the car. However, some locations are fairly difficult to distinguish, such as parks and the residential areas, or home and libraries. These are consistent with our intuition that these less distinguishable locations would sound like each other (parks could be close to residential areas; both home and library could be rather quiet). This is evidence that the classifiers we train indeed learn the characteristics of the environmental sounds.

4 Limitations

Since our results could be due to the limited data, data augmentation is expected to be very helpful. However, data augmentation in the context of environmental sound recognition is trickier than in speech recognition, because noise is often part of the environmental sound, and simply adding noise could change the label. One possible way to avoid that is to change the playback speed without changing the frequency using Phase Vocoder. Another possibility to enhance data is to use other environmental sound data to perform joint training on the two datasets. For example we can let two tasks share the same feature extraction and DNN pipeline, but use separate classifiers or softmax layers at the end for each task. While we can't use external data for the purpose of DCASE contest, it would still be an interesting direction to improve performance with the (labeled) data outside of the competition.

5 Conclusion

In this work we apply 7 models to acoustic scene recognition: Gaussian Mixture Models (GMMs), Support Vector Machines (SVMs), Deep Neural Networks (DNNs), Recurrent Neural Networks (RNNs), Recurrent Deep Neural Networks (RDNNs), Convolutional Neural Networks (CNNs), and Recurrent Convolutional Neural Networks (RCNNs). We use 4 sets of features extracted using signal processing techniques: MFCCs (60-dimension), Smile983 (983-dimension), Smile6k (6573-dimension), and spectrograms.

Our contributions are three-folds. 1) We show that Deep Learning models compare favorably with conventional models. A well-tuned DNN achieves 80% accuracy, compared with SVMs and GMMs with accuracies below 75%. 2) We show that with a sufficient feature representation non-temporal models outperform temporal models. In our experiments DNNs outperform RNNs and RDNNs on the Smile6k feature, which is an extensive feature set with 6573 dimensions. 3) We show that for the environmental sound recognition task, signal processing features outperform spectrogram features with visual models like CNNs. Overall our work suggests that variants of DNNs and signal processing features are preferred tools for modeling environmental sound and helps narrow down the model-feature search space substantially for future works in this domain.

6 Comments on the IEEE Contest

Just like most machine learning contests, feature engineering has large effects on accuracy. Better features give larger boosts than better models, though both feature and model choices are important for achieving good results. In terms of the features, supervised features extraction through CNNs and RCNNs do not outperform hand-crafted signal processing features, likely due to the limited data size. More extensive trials on various signal processing features could be beneficial. In terms of the models, the DNNs are the model of choice, with the caveat that careful tuning is needed for them to work well.

Another thing to note is that the labels in the contest dataset are rather coarse-grained. The dataset provides only the class label for each 30-second audio clip, whereas each audio clip could contain many events. Without resorting to external data (which is disallowed by the contest) and working within limited data size, it is unlikely to build more intelligent models to have levels of understanding akin to how humans understand the sound. Human listeners generally have strong prior knowledge of sound events (like bar code scanner beeping sound in the grocery store) to guide their classification. Enabling machines to understand these higher level features (e.g., from a large amount of unlabeled environmental sound recordings) could be a step closer to the human-level understanding of the environmental sounds.

7 Acknowledgement

We thank Bosch Research and Technology Center for valuable discussions and providing the computation resources to conduct the experiments.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012). Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chollet, F. (2015). keras. `https://github.com/fchollet/keras`.

Ciresan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012*, pages 3642–3649. IEEE.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Dollár, P. and Zitnick, C. (2013). Structured forests for fast edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1841–1848.

Eyben, F., Wöllmer, M., and Schuller, B. (2010). Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1459–1462. ACM.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874.

Gales, M. and Young, S. (2008). The application of hidden markov models in speech recognition. *Foundations and trends in signal processing*, 1(3):195–304.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Hasan, M. R., Jamil, M., and Rahman, M. G. R. M. S. (2004). Speaker identification using mel frequency cepstral coefficients. *variations*, 1:4.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

Hinton, G. (2012). Neural networks for machine learning. `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kim, T., Yue, Y., Taylor, S., and Matthews, I. (2015). A decision tree framework for spatiotemporal sequence prediction. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 577–586. ACM.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mesaros, A. (2016). 2016 dcase challenge. http://www.cs.tut.fi/sgn/arg/dcase2016/.

Mikolov, T., Kombrink, S., Deoras, A., Burget, L., and Cernocky, J. (2011). Rnnlm-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop*, pages 196–201.

Pinheiro, P. H. and Collobert, R. (2013). Recurrent convolutional neural networks for scene parsing. *arXiv preprint arXiv:1306.2795*.

Ranft, R. (2004). Natural sound archives: past, present and future. *Anais da Academia Brasileira de Ciências*, 76(2):456–460.

Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. (2000). Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1):19–41.

Roma, G., Nogueira, W., Herrera, P., and de Boronat, R. (2013). Recurrence quantification analysis features for auditory scene classification. *IEEE AASP Challenge: Detection and Classification of Acoustic Scenes and Events, Tech. Rep.*

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147.

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708.

Van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al. (1997). *The HTK book*, volume 2. Entropic Cambridge Research Laboratory Cambridge.

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., et al. (2013). On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE.