AUDIO BASED MULTIMEDIA EVENT CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS AND BAYESIAN LEARNING

Nishant Gurunath, Soham Kelkar, Kevin Chon
{ngurunat, sohamsak, khchon}@andrew.cmu.edu

ABSTRACT

Audio content present in web videos contains partial but valuable information about the event occurring in the video. The availability of only partial information poses a big challenge in event classification based on audio analysis. However, audio analysis has a key advantage in terms of the analysis rate and required data size needed in comparison to the full multimedia analysis. This work proposes convolutional neural network architectures for event classification based on audio content for YLI-MED dataset. We compare the performance to baseline i-vector and deep neural network (DNN) based methods. We adopt data augmentation and Bayesian learning methodologies to address the issue of lack of training data. We propose a training strategy to better handle negative event samples, events that do not belong to any class. From our results we discover that Bayesian networks clearly outperform regular convolutional neural networks and the importance of data augmentation for performance improvement given limited training data.

1 INTRODUCTION

Multimedia analysis has become increasingly popular in today's world, having applications in fields ranging from emotion recognition, home automation, real time captioning to telematics, robotics and aerospace. In this project, we focus on one of its applications - event classification. Event classification follows an algorithm of first isolating the requisite sounds from real-world noisy conditions and then classifying them into one of such possible events.

Speech recognition poses a lot of challenges. Our problem focuses on the audio aspect of the multimedia, which makes the event classification task particularly challenging. This is because the entire video represents the event, yet we are classifying it only based on its audio. This may lead to supply of inadequate information to our network. For our problem we use the YLI-MED dataset (`https://multimediacommons.wordpress.com/features/`), which is the subset of the YFCC100M corpus. The training data comprises of 5,609 utterances and their corresponding events. There are 11 categories of events out which 10 are labeled positive and one category which denotes 'None of these' is labeled negative. There are 952 positive samples and the remaining are labeled negative. Table-1 shows the distribution of samples across events for train and test dataset.

As a large chunk of training data consists of negative labels, it adds a challenge of how to classify these events. Another major challenge is the background noise. Since the dataset is not a pure audio dataset but an audio extracted from a video, the background noise is more prominent. Due to the lack of positive samples for our training, our model needs more information on the features of the samples of each event.

2 PREVIOUS WORK

Deep Neural Networks have proven to be very effective in speech recognition problems. Several approaches have been tried to solve this problem such as vanilla feedforward networks (1) and CNNs (2) (7). There has been a boost to the performance of image classification problems through

Event	**Training Samples**	**Test Samples**
Birthday Party	99	142
Flash Mob	91	85
Getting a Vehicle Unstuck	89	65
Parade	95	134
Person Attempting a Board Trick	99	112
Person Grooming an Animal	97	49
Person Hand-Feeding an Animal	95	118
Person Landing a Fish	99	63
Wedding Ceremony	90	114
Working on a Woodworking Project	98	62
None of the Above	4657	39935
Total	5609	40879

Table 1: Distribution of Samples Across Events

different CNN architectures like AlexNet (3), VGG (4), Inception (5) and ResNet (6). Some work has also been done in this field using the i-vector system. All these provide a helpful baseline for us to compare our work. The usual metric used for estimating the performance of these networks is the mean average precision (mAP). This is calculated by taking the mean of the average precision, which is calculated by obtaining the cumulative precision for each sample of the test data. The i-vector system (9) gave a mAP estimate of 0.22% when trained on both train and test data, while the DNN gave a mAP estimate of 1.18% (1). This clearly shows the superiority of neural networks in this problem. As convolutional architectures have been very effective in dealing with speech recognition problems, we are curious to explore whether they will perform better than dense networks in this problem.

3 OUR EXPERIMENTS

To address the limitations in our dataset, we performed data augmentation on the training dataset and added temporal perturbation to generate more new data. This improved the performance of the CNN significantly. We then applied the concept of Bayesian learning to train the model. The idea was to learn a distribution on the parameters instead of a single deterministic value. This would make the model robust to uncertainty with regard to unknown data and help the model to generalize better when there is very limited training data. Consequently, the performance of the Bayesian network on the YLI-MED dataset was significantly better than a non-Bayesian network.

We also trained our CNN on the DCASE General Purpose Audio Tagging dataset(`http://dcase.community/challenge2018/task-general-purpose-audio-tagging`). This dataset contains videos where each video labeled by one action e.g scissors, fireworks etc. We then use the model trained on these actions and map it to our original YLI-MED dataset. Thus, we obtained a model which can predict an event not only based on the random raw features of the videos. This model followed a more logical approach of combining different actions learned from the DCASE dataset and then predicted one of the 10 events in our original data. However, the results obtained from this approach didn't match our expectations.

We implemented a CNN with a ResNet architecture for classification (6). We used 2D convolution layers in our network as they capture both the temporal and the frequency components, which intuitively makes more sense. Our Residual Block consisted of 2 convolution layers with a kernel size of 3x3 and a padding of 1x1. Since we have extracted the audio data from videos, each audio sample is comparatively different from the other. We applied batch normalization after each layer in the residual block. Each residual block was followed by another convolution layer with a kernel size of 3x3, a stride of 2. ELU activations were applied to each of these convolution layers, as ELU has shown to perform better specifically for speech progressing tasks as opposed to other standard activation functions. The CNN was followed by 2 dense layers to downsample it to the output size.

Layer No.	Layer	Channels
1	Input (2000x20)	1
2	Conv2D(3x3) (Stride=2)	32
3	ResNet (3x3)	32
4	Conv2D (3x3) (Stride=2)	64
5	ResNet (3x3)	64
6	Conv2D (3x3) (Stride=2)	128
7	ResNet (3x3)	128
8	AvgPool (249)	128

Table 2: Model Architecture

The CNN was used as a feature extractor on the YouTube audios. We tried different methods and algorithms to improve this model.

3.1 Boosting

As mentioned previously, the training data present in the YLI-MED dataset consists of a large chunk of negative samples. With vanilla sampling, the model tends to overfit to this large chunk while reducing the loss. However, we wanted the model to focus on the relevant events that were misclassified. Hence, we adopted the approach of boosting by increasing the weights of the misclassified samples. The weight update for the samples was performed after each epoch. The loss function was computed between the weighted predictions and true labels,

$$\epsilon_t = \sum_i D_i(t)\delta(h_i(t) \neq y_i) \tag{1}$$

$$\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t}) \tag{2}$$

where, ϵ_t and α_t are the weighted error and learning rates for weights respectively at epoch t. The weight update rule for misclassified points is

$$D_i(t+1) = D_i(t)\exp(0.1\alpha_t) \tag{3}$$

The weight update rule for correctly classified points is

$$D_i(t+1) = D_i(t)\exp(-0.1\alpha_t) \tag{4}$$

where, $D_i(t)$ represents weight for the i^{th} sample at epoch t. The loss function is computed as

$$loss(t) = Xent(xD(t), y)$$

3.2 Transfer Learning

The lack of data presented a problem for the network that it could not extract all the features which represented a particular event. For example, 100 samples are not enough for the network to learn all the possible combinations of features which represent a birthday party. So building on this logic, we wanted our network to learn these features through another dataset. We use the DCASE Audio Dataset which contains 41 basic actions and trained our model on that data. We then used this trained model to extract features from the YouTube audios and then used a dense layer to map the extracted features to the events. So basically, the network was learning how to extract basic actions and then correlating them to the events. As shown in Figure 1, the model first learns the basic actions on the Action Dataset and then learns the mapping between the actions and the events on the YouTube audios.

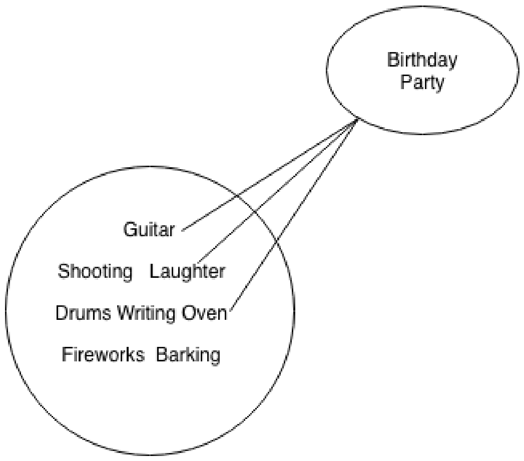

Figure 1: Transfer Learning

3.3 Data Augmentation

Another way to address the issue of lack of training data is data augmentation. We employ ways to transform or perturb data temporally such that the newly generated data will represent the corresponding event while appearing distinct to the model. The way we achieve this is to first add small Gaussian random noise to the data and then temporally rotate the data.

$$x(t) = x_i(t) + 0.01\mathcal{N}(0,1) \tag{5}$$

$$x_o(t) = x(t - t') \tag{6}$$

This is effective for event classification because irrespective of when the important features that represent the event occur they still describe the event. For example, it doesn't matter when the cake cutting celebration sounds occur in a Birthday Party event. Figure 2 depicts the data before and after the transformation.

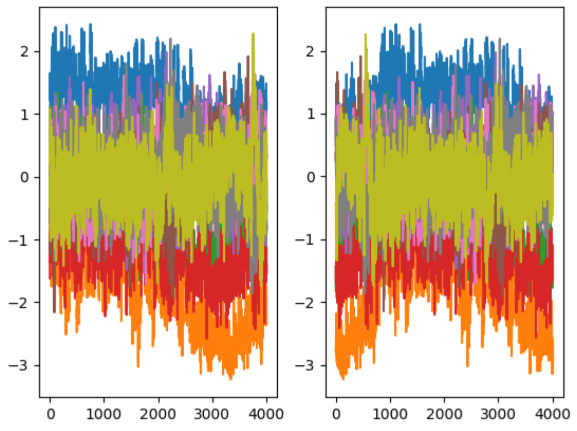

Figure 2: Transformation for Data Augmentation

As expected, we observe almost a 3 percent point improvement in the CNN performance after data augmentation. We also observe that performance doesn't improve much as we keep augmenting more and more data as it doesn't change the data variance after a point.

3.4 BAYESIAN NETWORKS

While neural networks learn a maximum likelihood (MLE) value of the network parameters, the Bayesian neural networks learn a posterior by learning the distribution over each parameter. This makes the model robust to the uncertainties with regard to the unknown data.

Neural networks MLE:

$$W^* = argmax_W P(Y/X, W) \tag{7}$$

Bayesian neural networks with gaussian prior on parameters:

$$\mu^*, \sigma^* = argmax_{\mu,\sigma} P(Y/X, W)P(W; \mu, \sigma) \tag{8}$$

This is essentially a re-parameterization problem with an assumption over a prior on parameters. We adopted the Pytorch model on Bayesian neural networks from (11). We replaced our regular convolutional layers in the network with the Bayesian version of it. This method produced the best result, among the methods we have tried, with a 4 percent point improvement over the vanilla CNN. We came across one key observation while using Bayesian neural networks with augmented data. Bayesian neural network performance deteriorated when trained on the augmented training dataset. We think this could be due to the parameter uncertainty applied over the data uncertainty, resulting in the model poorly estimating the network parameters.

4 RESULTS

We experimented with four methods to improve the baseline model. The baseline model was a dense network pre-trained on TRECVID MED dataset with a similar set of events. We could not train on the TRECVID MED dataset as it is not publicly available. The first method we used was a Vanilla CNN with Boosting. This CNN was trained on 10% of the data and tested on the rest, in the same fashion observed in the baseline model. The second method was transfer learning which gave results worse than the vanilla CNN. The data augmentation improved the model above the vanilla CNN, and the Bayesian Networks gave the best results with an accuracy of 34.2%. The Bayesian Networks were trained both with and without data augmentation. However as discussed earlier, the data without the perturbation gave better results which are mentioned in Table 3.

Model	Baseline	Vanilla CNN	Data Augmentation	Bayesian Network
Accuracy	37.4%	30.3%	33.1%	34.2%

Table 3: Results

As individual events are unique with respect to the other, the results were computed on every event individually as shown in Table 4.

Event	**Vanilla CNN**	**Data Augmentation**	**Bayesian Networks**
Birthday Party	76.06	64.79	64.79
Flash Mob	15.29	28.24	55.29
Getting a Vehicle Unstuck	12.31	18.46	16.92
Parade	38.31	44.78	47.76
Person Attempting a Board Trick	11.61	24.11	21.43
Person Grooming an Animal	14.29	18.37	16.33
Person Hand-Feeding an Animal	21.19	18.64	19.49
Person Landing a Fish	20.63	30.16	28.57
Wedding Ceremony	32.46	28.07	26.32
Working on a Woodworking Project	16.13	24.19	9.68

Table 4: Accuracy of Individual Events

5 Discussion

While the previous state-of-the-art approach involved dense neural networks pre-trained on a private dataset, we expected our results to still be able to meet the baseline result of 37.4% accuracy. While the intuition behind our transfer learning approach (to learn the mapping of basic actions to events) seemed valid, we realized that our approach was ultimately limited by the lack of coverage in both datasets. 41 different sounds could not be translated as sufficient differentiating features corresponding to our specific ten diverse events. Had the audio dataset been significantly larger (both in terms of different range of corresponding actions and more samples), we believe our approach would have been successful. One other thing that we realized is that boosting does not seem to work well with neural networks. This could possibly because of the assumptions we had to make to reduce computation complexity while implementing boosting with neural networks.

The two methods that did work well given a small training dataset were data augmentation and application of Bayesian inference on neural networks. Although we could not match the baseline model that we were targeting, we consider these methods as our key contributions (as they have not been tried on this dataset to the best of our knowledge). There are ways to extend and potentially improve the results. We could input our data into a LSTM and take its output into our CNN architecture. The motivation behind this being that since our event detection can occur anywhere in the audio file, a LSTM structure would help preserve time-invariance. The other way we could possibly improve our model is to pretrain our network on dataset with similar events. We also thought about using GANs or VAEs to generate more data from the input distribution. However, after some literature review and initial experimentation, we came to the conclusion that generative models may not work effectively with such a small dataset.

References

[1] Khalid Ashraf, Benjamin Elizalde, Forrest Iandola1, Matthew Moskewicz1, Julia Bernd2, Gerald Friedland2, Kurt Keutzer1. Audio-Based Multimedia Event Detection with DNNs and Sparse Sampling

[2] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, Kevin Wilson, Google, Inc., New York, NY, and Mountain View, CA, USA. CNN Architectures for Large-Scale Audio Classification

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, Rethinking the inception architecture for computer vision, arXiv preprint arXiv:1512.00567, 2015.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in Advances in neural information processing systems, 2012, pp. 10971105.

[5] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition.

[7] Rohan Badlani, Ankit Shah, Benjamin Elizalde, Anurag Kumar, Bhiksha Raj. FRAMEWORK FOR EVALUATION OF SOUND EVENT DETECTION IN WEB VIDEOS.

[8] Tom Mikolov, Martin Karafit, Luk Burget, Jan ernock, Sanjeev Khudanpur. Recurrent Neural Network Based Language Model

[9] B. Elizalde, H. Lei, and G. Friedland. An i-vector representation of acoustic environments for audio-based video event detection on user generated content. In ISM, 2013.

[10] Neal, R. M. (2012). Bayesian learning for neural networks (Vol. 118). Springer Science Business Media.

[11] https://github.com/kumar-shridhar/PyTorch-BayesianCNN/blob/master/utils/BBBlayers.py