EasyChair Preprint

№ 336

Human action recognition using fusion of modern deep convolutional and recurrent neural networks

Dmytro Tkachenko

July 10, 2018

Human action recognition using fusion of modern deep convolutional and recurrent neural networks

Dmytro Tkachenko
Institute for Applied System Analysis
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Kyiv, Ukraine
me@dmitriytkachenko.com

***Abstract*—This paper studies the application of modern deep convolutional and recurrent neural networks to video classification, specifically human action recognition. Multi-stream architecture, which uses the ideas of representation learning to extract embeddings of multimodal features, is proposed. It is based on 2D convolutional and recurrent neural networks, and the fusion model receives a video embedding as input. Thus, the classification is performed based on this compact representation of spatial, temporal and audio information. The proposed architecture achieves 93.1% accuracy on UCF101, which is better than the results obtained with the models that have a similar architecture, and also produces representations which can be used by other models as features; anomaly detection using autoencoders is proposed as an example of this.**

Keywords—convolutional neural networks, human action recognition, recurrent neural networks, representation learning, video classification.

I. Introduction

Video classification is an important problem which has many applications in robotics, security, user-generated content moderation, etc. In addition to the classification having a value on its own, it is also a subproblem in more complex models or control systems.

Videos are intrinsically multimodal so that one can use motion and auditory clues in addition to a sequence of frames. This gave rise to multi-stream architectures [1]-[2] which model spatial (still video frames), temporal (motion), and audio streams. The temporal stream is usually represented [1] by the dense optical flows between frames, which serve as the hand-crafted features for a model since they are computed by a different model or algorithm. The audio stream can be transformed into a spectrogram instead of using a raw waveform, in line with established approaches to audio classification [3].

More complex problems which include video classification as a subproblem may also require modeling other information such as text, and consequently, the complete model output might need to be based on all of the data sources. Such fusion model may receive feature representations as input. Because of this, it is valuable to be able to learn representations (embeddings) for videos. If a model can extract a compact representation of a video, it can be fused with, e.g., an embedding for the relevant text (obtained using techniques similar to word2vec [4]).

Datasets, which contain clips that depict various human actions (like sports, playing musical instruments, etc.), are often used as a benchmark for video classification models. Examples of the popular human action recognition datasets are UCF101 [5] and Kinetics [6].

II. Related Works

One of the first multi-stream architectures described in [1] is based on 2D convolutional neural networks (CNN). This method takes motion into account by modeling a temporal stream using a separate model, and the classification result is computed by fusing class scores returned by the two CNNs. The temporal stream CNN receives stacked optical flows, estimated over a window of several frames, as input. Although the motion is taken into account this way, this architecture might not be able to model longer videos with more complex actions. This is because the optical flow stacks are computed over relatively short windows. Fusing the class scores which are obtained by classifying short video segments represented by optical flows (in a sliding window manner) thus cannot model long sequences, because fusion model does not take the ordering of segments into account.

The improved multi-stream model has been proposed in [2]. This model uses recurrent neural networks (specifically, LSTM [7]) which receive video segment embeddings as input: for the spatial stream it is a sequence of frame embeddings; the temporal stream is represented by a sequence of stacked optical flow embeddings. The audio clues are also used: a separate CNN classifies spectrograms. The final class scores are computed by fusing the predictions of the individual streams. Overall, the multi-stream framework proposed in [2] uses all of the available information and can classify videos which contain long and complex actions because of the use of RNNs. However, one downside of this method is that every stream is processed independently; as a result, the fusion model only looks at the class probabilities and thus cannot consider all of the information at the same time. video2vec [8] explores using embedding vectors of spatial and temporal streams for classification; however, the prediction accuracy on UCF101 is lower compared to [2].

Recently, several architectures based on 3D convolutional neural networks have been developed. 3D convolution and pooling operations are performed spatio-temporally, meaning that they model a volume of multiple 2D images with the third dimension being time, and produce another 3D volume. C3D [9] uses relatively shallow CNN architecture, trained from scratch on large video datasets, that is applied to non-overlapping frame clips with the classification result computed by averaging the scores predicted for all clips. I3D [10] proposes inflating 2D CNNs into 3D and bootstrapping 3D filters from 2D filters, which provides parameter initialization from 2D models trained on ImageNet. [11] explores even deeper ResNet-based architectures similar to those that worked well for image recognition. These 3D CNNs achieved very good results; however, it is worth noting that they were trained on very large datasets like Kinetics [6] and were only able to produce decent accuracy on smaller datasets

like UCF101 after pre-training on larger ones. Models based on 3D convolutional networks have a bigger number of parameters and thus require larger datasets to train on, and more computational resources, compared to architectures based on 2D CNNs and RNNs. As a result, it seems that the architectures based on 3D CNNs are less sample-efficient compared to 2D CNN + RNN solutions, meaning that they require more data to achieve the same results.

Conceptually, it seems that using recurrent neural networks for modeling sequences is better, because CNNs suffer from fundamental limitations, i.e., not modeling the relative positions or spatial hierarchies between objects – the very same problem capsule networks [12] are trying to address. Consequently, convolving the time dimension can lead to losing important information. Another drawback of using 3D CNNs stems from applying them to relatively short clips and fusing (e.g., averaging) the predictions; it is impractical for the input volume to span an entire video since it would be too big to process; however, there are cases where this approach would fail, for example, when a video contains only a short segment which depicts an action of interest (for classification); in this case averaging would most likely lose the information about some action detected in only a small number of segments, so the result would be incorrect. Given these shortcomings, although 3D CNNs work well in practice (on existing datasets), it is unclear whether they would perform as well on more complex data (long and complex videos, such as movies).

III. Methodology

The proposed architecture is based on 2D convolutional neural networks and recurrent neural networks. It is illustrated in the fig. 1. The submodels for all streams have the similar architecture: segments of raw data → CNN → sequence of segment embeddings → RNN → video-level embedding.

A. *Spatial stream*

The raw 2D images (RGB) are fed into the convolutional neural network to extract embedding vector which is taken from the last layer before the dense classification layer.

Then the sequence of frame embeddings (of size *ExL*, where *E* is the length of the embedding vector, and *L* is the sequence length) is processed by the recurrent neural network, which returns the embedding vector with the length that depends on the number of hidden units in the last RNN cell. This embedding vector describes the entire video.

B. *Temporal stream*

The optical flows are computed between pairs of frames (e.g., between the first and the second frame, then between the second and the third frame). Then they are stacked in a way described in [1]. This results in stacks of size *WxHx2L*, where *W* is the width of a frame, *H* is its height, and *L* is the size of the stack (multiplied by 2 since there are *x* and *y* components of optical flow displacement vectors).

Another 2D CNN is then used to extract embeddings for stacked optical flows. These embeddings are then processed by RNN, which again returns the video embedding.

C. *Audio stream*

The audio track is split into the non-overlapping examples. For each of them, the log-mel spectrogram is computed, which is then fed into the convolutional neural network to extract an

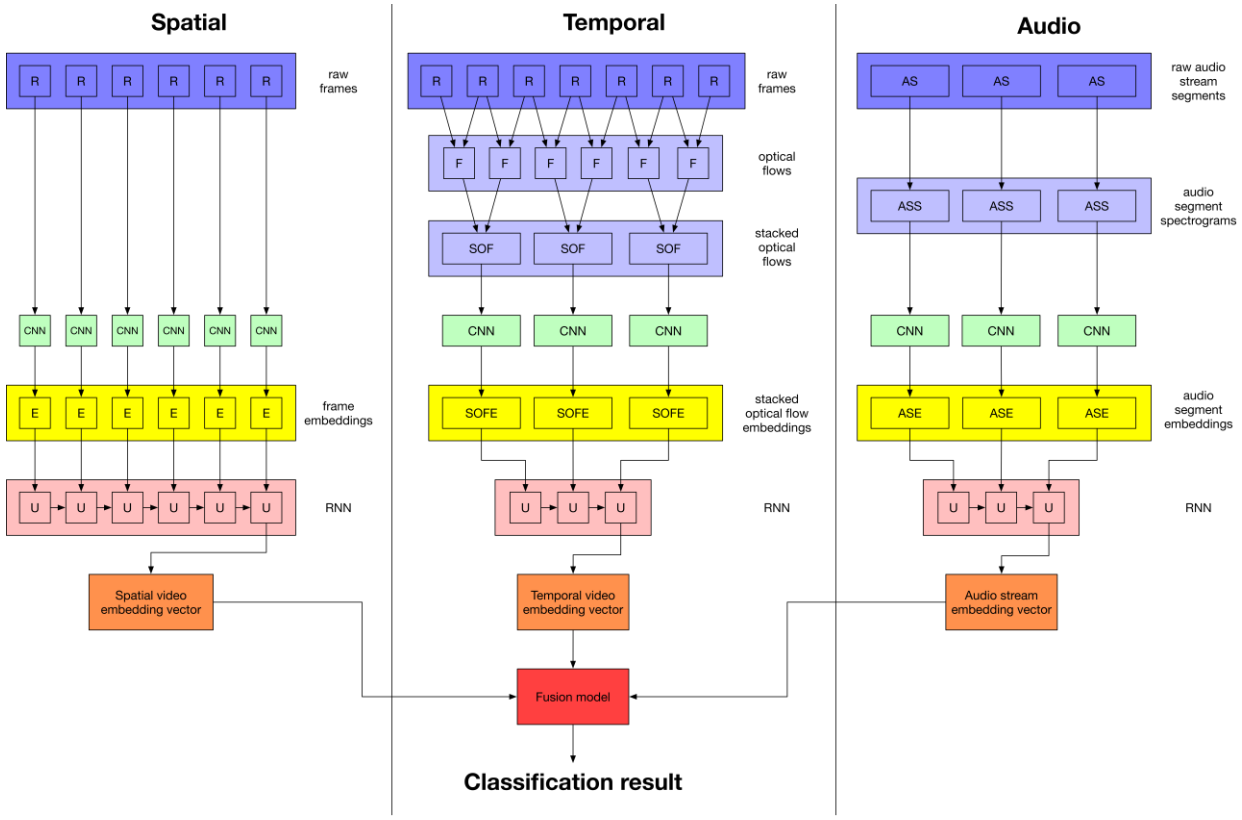

Fig. 1. Architecture of the proposed model

embedding vector. This approach follows the one described in [3]. Video-level embedding vector is then extracted by an RNN.

D. Fusion model

Video-level embeddings for the three streams are then concatenated, and the resulting vector completely describes the features of a video. The fusion model then computes the classification result (class scores).

E. Training

The models for every stream are trained separately. Joint training of the entire model is very computationally demanding, and [13] shows that it only yields a tiny improvement in accuracy. The separate training of streams makes the model more flexible, since replacing some component of the model does not require the complete re-training. So, the training process includes training (or fine-tuning) the CNNs, then RNNs, and then the fusion model.

IV. Implementation Details

A. Image feature extraction

As a CNN for extracting embedding vectors from the images, InceptionResNetV2 [14] architecture is used. This model achieves 80.4% accuracy on the test set of the ImageNet classification (CLS) challenge and 95.3% top-5 accuracy, which makes it one of the best models as of now. To retrieve frame embedding, the upper dense layer (used for classification) is removed and the output of the top global average pooling layer, which produces a 1536-dimensional vector, is used.

B. Optical flow computation

Recently, it has been shown that the methods for estimating optical flow between frames, which use convolutional neural networks, outperform classical computer vision methods. Particularly, FlowNet 2.0 [15] currently has the best accuracy, so it is used in the proposed method. FlowNet 2.0, despite being somewhat slow, produces very accurate results with smooth flow fields and crisp motion boundaries.

C. Optical flow feature extraction

For extracting embeddings from optical flow stacks, the convolutional neural network similar to the one described in [1], is used. This motion CNN looks as follows:

conv1 (7x7x96; stride 2; pooling 3x3; norm) → conv2 (5x5x256; stride 2; pooling 3x3) → conv3 (3x3x512; stride 1) → conv4 (3x3x512; stride 1) → conv5 (3x3x512; stride 1; pooling 3x3) → full6 (4096; dropout) → full7 (2048; dropout) → softmax

Here *FxFxN* for convolutional layers denotes *N* filters of size *FxF*. Max pooling is done over 3x3 spatial windows with stride 2. Local response normalization used in [1] is replaced by batch normalization [16]. The number of neurons is specified in brackets for fully-connected (*full*) layers.

With this architecture, extracting embedding from the last fully-connected layer results in a 2048-dimensional vector.

D. Audio feature extraction

VGGish [3] model is used for extracting a 128-D embedding for every audio segment.

E. Recurrent neural network architecture

It has been found experimentally (see "Experiments" section) that stacked (two layers) bidirectional GRU with 512 hidden units performs best. This model produces 1024-dimensional (outputs from the forward and backward RNNs are concatenated) video-level embedding vectors.

F. Adaptive sampling

It is often not practical to use all frames as input to RNN since it is harder to train them with long sequences. As a result, videos are usually subsampled to reduce dimensionality. This is done by selecting frames with a constant step, for example, taking every 5^{th} frame.

This paper proposes the adaptive sampling method which uses variable step depending on the amount of information a specific video segment contains.

Specifically, computing optical flow between two frames produces displacement vectors for every pixel. The speed of a movement can be estimated by looking at the lengths of these vectors. It might be advantageous to select more frames from the video segments which contain faster movement. As a result, the number of samples selected from a segment is suggested to be made proportionate to average displacement vectors lengths between frames in this segment.

G. Fusion model

Linear support vector machine (SVM; i.e., hinge loss and L_2 regularization) is used as a fusion model.

V. Experiments

A. Dataset

Experiments were performed on UCF101 [5], which is a widely used human action recognition dataset that contains 13320 video clips annotated into 101 classes. The video clips have a resolution of 320x240 and a fixed frame rate of 25 frames per second. UCF101 provides three train/test splits; this paper reports the results on the first split.

B. Training

Fine-tuning upper layers (3 Inception blocks) of spatial CNN (InceptionResNetV2) on UCF101 didn't yield any improvement, so pre-trained (on ImageNet) weights are used.

UCF101 doesn't have enough audio data for training or fine-tuning a sufficiently deep convolutional neural network. Thus, VGGish [3] model is used with the weights pre-trained on AudioSet [17].

Motion CNN is trained from scratch, as well as RNNs for all streams.

C. Preprocessing

Before feeding frames to CNN, they are resized to 299x299x3 which is the default input size for InceptionResNetV2.

Videos are limited to the first 36 seconds (only one clip in UCF101 is longer). Sequence length is limited to 180, and

frames are sampled using the adaptive sampling method described above.

For motion CNN, the native video resolution of 320x240 is used.

D. Training setup and parameters

RNN models were trained using stochastic gradient descent (SGD) with Nesterov momentum (also known as Nesterov Accelerated Gradient or NAG). Learning rate has been reduced by a factor of 10 if validation loss hadn't decreased for 5 epochs. Initial learning rate for SGD was 10^{-2}, and the lowest possible value (after reductions) was 10^{-6}.

Adaptive gradient methods (like Adam, RMSProp, or Adadelta) didn't yield significantly faster convergence when training RNNs, and NAG usually produced better results, which is consistent with published observations [18]. Adam was only used for motion CNN which benefits from faster convergence since it was trained from scratch. Default parameters provided in the original paper [19] were used.

Batch size was 32. Training has been run for some number of epochs until validation loss had stopped improving: if it hadn't decreased for 10 epochs, the training had been ceased.

In RNN models, dropout (with the rate of 0.5) was applied to the input layer, between RNN model and top dense layer, and between LSTM/GRU layers.

L2 regularization parameter for linear SVM (fusion model) was set to 10^{-2}.

E. Selecting RNN architecture

Different RNN model architectures (based on LSTM and GRU) with a varying number of hidden units were trained on the spatial stream data. Results are shown in the Table I. Loss function used was the cross-entropy loss (top dense layer used softmax activation function). The best model (marked with (*)) was also evaluated with linear SVM, and with attention layer [20] applied after the input layer.

"S." denotes two stacked layers, and "Bi" prefix means bidirectional layer was used; the specified number of units is per single LSTM/GRU cell in a model.

TABLE I. SPATIAL STREAM CLASSIFICATION RESULTS WITH DIFFERENT RNN ARCHITECTURES

Model	Hidden units	Top-1, %	Top-5, %
LSTM	256	76.24	93.49
LSTM	512	77.09	93.09
LSTM	1024	77.07	93.62
LSTM	2048	76.22	93.62
GRU	512	77.83	93.88
GRU	1024	77.81	93.17
BiLSTM	512	77.36	93.41
BiLSTM	1024	75.93	93.49
BiGRU	512	77.33	93.46
BiGRU	1024	77.52	93.41
S. LSTM	512	74.71	92.80
S. LSTM	1024	76.80	**94.09**
S. GRU	512	77.09	93.03
S. GRU	1024	77.22	93.25
S. BiGRU (*)	512	**78.34**	93.51
S. BiGRU	1024	77.15	93.17
S. BiLSTM	512	74.50	92.77
S. BiGRU + attention	512	75.03	92.89
S. BiGRU + SVM	512	**79.40**	90.84

Results show that the models with RNN cells that contain 512 or 1024 units generally perform best on this task. Stacked GRU layers provide better accuracy than stacked LSTM layers, likely because they are less prone to overfitting since they have fewer parameters. Using attention layer significantly decreased the accuracy. The best top-1 classification accuracy (78.34%) has been obtained with two stacked bidirectional GRU layers with 512 hidden units in every cell. This was further improved by using SVM instead of softmax, which increased the accuracy to 79.40%.

F. Model evaluation

Finally, the complete model was evaluated after training motion CNN and RNNs for all streams. Table II shows the results and the comparison with other models.

VI. DISCUSSION

The proposed model achieved better accuracy than the models which have the similar architecture [1]-[2] (2D CNN + RNN), and the result is significantly better compared to [8] which also used embeddings for classification. It also outperforms early 3D CNN architecture [9]. [10] and [11] obtain better results, however, these are much more complicated models which were pre-trained on much bigger dataset (Kinetics); consequently, their results are not directly comparable. The proposed model achieves good results without pre-training on bigger datasets, and thus it is better suited for use cases where the limited amount of training data is available.

VII. ANOMALY DETECTION

It has been stated that the video embeddings extracted by the proposed model can be used as features input to another model. Anomaly detection using autoencoders can be an example of this. Detecting anomalies in a video stream is useful for a variety of applications; for example, traffic camera feed can be analyzed to detect accidents.

TABLE II. MODEL EVALUATION RESULTS AND COMPARISON

	UCF101 top-1, %	Type	Streams	Pre-training
Proposed model	93.1	2D CNN + RNN (GRU)	Spatial, temporal, audio	Spatial 2D CNN on ImageNet
Two-stream fusion [1]	88.0	2D CNN	Spatial, temporal	Spatial 2D CNN on ImageNet
Multi-stream [2]	92.6	2D CNN + RNN (LSTM)	Spatial, temporal, audio	Spatial 2D CNN on ImageNet
video2vec [8]	87.5	2D CNN + RNN (GRU)	Spatial, temporal	Spatial 2D CNN on ImageNet
C3D [9]	90.4	3D CNN	Spatial, temporal	I380K (private) + Sports-1M (public)
I3D [10]	98.0	3D CNN	Spatial, temporal	ImageNet + Kinetics
ResNeXt-101 (64f) [11]	94.5	3D CNN	Spatial	Kinetics

Autoencoder model consists of an encoder, a bottleneck, and a decoder. The idea of anomaly detection using autoencoders is that because the hidden layer (bottleneck) has much lower dimensionality than the input, it only learns the most important features; putting it another way, it learns the most frequent features; so, in the context of anomaly detection problem, it would learn the features of normal examples because there is a majority of them. The autoencoder loss function measures the difference between the input and the reconstructed output and can be, for example, a mean squared error. So, after training an autoencoder, anomalous examples would have a higher reconstruction error than the normal ones.

The video embedding vectors can be used as an input to autoencoder. The bottleneck layer would need to have lower dimensionality (than embeddings), and autoencoder would learn to reconstruct the input embeddings. This would have an effect of losing information related to anomalous examples so they would have a higher reconstruction error. Theoretically, the proposed model can be used as an encoder part of an autoencoder. However, such autoencoder would be extremely hard to train before of a large number of parameters; using embeddings serves as an easy way to perform anomaly detection with autoencoder that can have a simple architecture.

VIII. Conclusions

This paper presented a multi-stream model for learning video representations, which takes spatial, temporal and audio streams into account. It is based on two-dimensional convolutional and recurrent neural networks.

It has been shown that the individual components of this model can be trained separately. The resulting embeddings can be fused to perform classification, which has been shown to outperform the similar architectures, achieving 93.1% accuracy on UCF101.

Compared to the previous works, the proposed model uses more modern architectures and techniques, e.g., the optical flow estimation method based on convolutional neural networks. The adaptive sampling method has also been proposed; it allows to sample frames from a video at a variable rate depending on the amount of information in a video segment so that the segments with faster movement are represented by a higher number of training examples.

The video representations, extracted by this model, can be used as features input to another model; using them in anomaly detection with autoencoders has been described as an example.

It is argued that having the proposed model use recurrent neural networks is beneficial because such architectures tend to be more sample-efficient, i.e., can be trained with fewer data. It also avoids the pitfalls of applying convolutional neural networks to the time dimension, which results in the loss of long-term context and order. Consequently, this model can also be more successful in modeling longer videos with more complex actions.

References

[1] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” *Advances in Neural Information Processing Systems (NIPS)*, vol. 27, pp. 568-576, 2014.

[2] Z. Wu, Y.-G. Jiang, H. Ye, X. Wang, and X. Xue, “Multi-stream multi-class fusion of deep networks for video classification,” *Proceedings of the 2016 ACM on Multimedia Conference*, pp. 791-800, 2016.

[3] S. Hershey et al., “CNN architectures for large-scale audio classification,” *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131-135.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *CoRR*, abs/1301.3781, 2013.

[5] K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A dataset of 101 human actions classes from videos in the wild,” *CoRR*, abs/1212.0402, 2012.

[6] W. Kay et al., “The kinetics human action video dataset,” *CoRR*, abs/1705.06950, 2017.

[7] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9 (8), pp. 1735-1780, 1997.

[8] S. Hu, Y. Li, and B. Li, “Learning semantic spatio-temporal embeddings for video representation,” *International Conference on Pattern Recognition (ICPR)*, vol. 23, pp. 811-816, 2016.

[9] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3D convolutional networks,” *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 4489–4497, 2015.

[10] J. Carreira and A. Zisserman, “Quo vadis, action recognition? A new model and the kinetics dataset,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4733, 2017.

[11] K. Hara, H. Kataoka, and Y. Satoh, “Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet?”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” *Advances in Neural Information Processing Systems (NIPS)*, pp. 3859–3869, 2017.

[13] J. Donahue et al., “Long-term recurrent convolutional networks for visual recognition and description,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2625–2634, 2015.

[14] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, Inception-ResNet and the impact of residual connections on learning,” *AAAI*, vol. 4, 2017.

[15] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “FlowNet 2.0: Evolution of optical flow estimation with deep networks,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017.

[16] C. Szegedy and S. Ioffe, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, pp. 448–456, 2015.

[17] J. F. Gemmeke et al., “Audio Set: An ontology and human-labeled dataset for audio events,” *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776-780, 2017.

[18] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, “The marginal value of adaptive gradient methods in machine learning,” *Advances in Neural Information Processing Systems (NIPS)*, pp. 4151-4161, 2017.

[19] D. P. Kingma, “Adam: A method for stochastic optimization,” *CoRR*, abs/1412.6980, 2014.

[20] A. Vaswani et al., “Attention is all you need,” *Advances in Neural Information Processing Systems (NIPS)*, pp. 6000-6010, 2017.