EdgeL^3: Compressing L^3-Net for Mote-Scale Urban Noise Monitoring

Sangeeta Kumari
Ohio State University

Dhrubojyoti Roy
Ohio State University

Mark Cartwright
New York University

Juan Pablo Bello
New York University

Anish Arora
Ohio State University

May 24, 2019

Outline

Outline

Credit: *Getty Images*

- 70 million people across USA were exposed to noise levels beyond what the EPA considers harmful (2014)

Urban Noise Monitoring

Credit: *Getty Images*

- 70 million people across USA were exposed to noise levels beyond what the EPA considers harmful (2014)
- In 2016, NYC's 311 service line received an average of 48 noise complaints per hour

Urban Noise Monitoring

Credit: *Getty Images*

- 70 million people across USA were exposed to noise levels beyond what the EPA considers harmful (2014)
- In 2016, NYC's 311 service line received an average of 48 noise complaints per hour
- Limitations with 311 reporting
  - Inaccurate information on all sources of disruptive noise
  - Verification of authentic noise complaints

SONYC

- Sounds of New York City (SONYC) aims at continuous monitoring, analysing, and mitigating urban noise pollution

Figure 1: Acoustic sensing unit deployed on a New York City street

Machine Listening Goals

- Low-cost and battery/solar powered sensing

Machine Listening Goals

- Low-cost and battery/solar powered sensing
- Real-time multi-label noise classification
  - Noise: traffic, sirens, construction, unnecessary honking, social noise etc.

Machine Listening Goals

- Low-cost and battery/solar powered sensing
- Real-time multi-label noise classification
  - Noise: traffic, sirens, construction, unnecessary honking, social noise etc.
- Address lack of annotated data

Machine Listening Goals

- Low-cost and battery/solar powered sensing
- Real-time multi-label noise classification
  - Noise: traffic, sirens, construction, unnecessary honking, social noise etc.
- Address lack of annotated data
- Limited Flash (2 MB) and RAM (1 MB) on edge devices (ARM Cortex-M7)
  - '*mote-scale*' devices

Outline

Look, Listen, and Learn (L^3-Net)

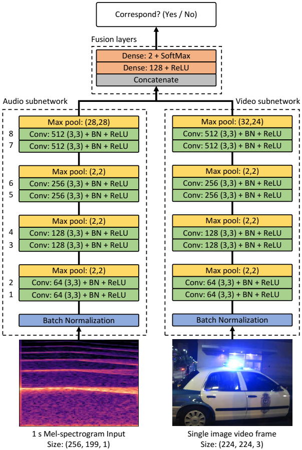

Figure 2: Architecture of the L^3-Net embedding models

- L^3-Net trains audio embedding by learning associations between audio snippets and video frames [1]
    - Audio-Visual Correspondence (AVC) task

[1] Arandjelovic, Relja and Zisserman, Andrew. "Look, Listen and Learn". IEEE ICCV. 2017.

Look, Listen, and Learn (L^3-Net)

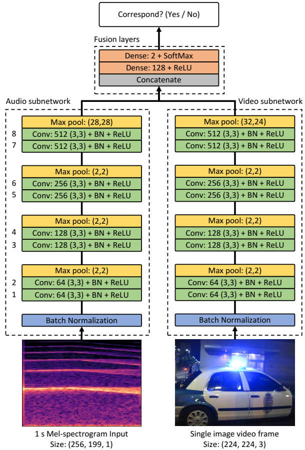

Figure 2: Architecture of the L^3-Net embedding models

- L^3-Net trains audio embedding by learning associations between audio snippets and video frames [1]
  - Audio-Visual Correspondence (AVC) task
- Use audio embedding to train downstream task (classifier for limited data)

[1] Arandjelovic, Relja and Zisserman, Andrew. "Look, Listen and Learn". IEEE ICCV. 2017.

Look, Listen, and Learn (L^3-Net)

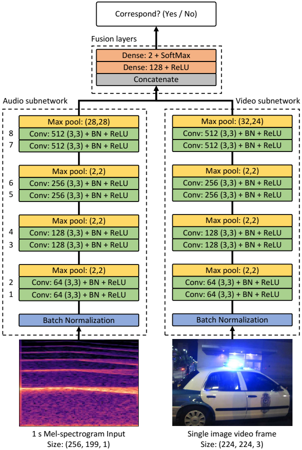

Figure 2: Architecture of the L^3-Net embedding models

- L^3-Net trains audio embedding by learning associations between audio snippets and video frames [1]
  - Audio-Visual Correspondence (AVC) task
- Use audio embedding to train downstream task (classifier for limited data)
- Downstream datasets:
  - *US8K*: 8732 audio clips divided into 10 cross-validation folds
  - *ESC-50*: 2000 clips divided into 5 folds
- Downstream Accuracy
  - US8K: 75.91% | ESC-50: 73.65%

[1] Arandjelovic, Relja and Zisserman, Andrew. "Look, Listen and Learn". IEEE ICCV. 2017.

Look, Listen, and Learn (L^3-Net)

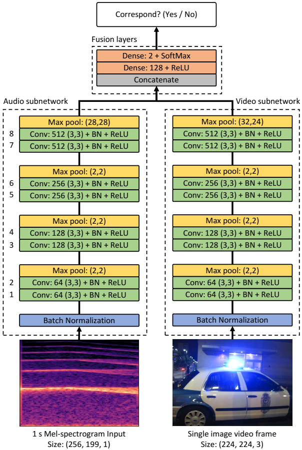

Figure 2: Architecture of the L^3-Net embedding models

- L^3-Net trains audio embedding by learning associations between audio snippets and video frames [1]
  - Audio-Visual Correspondence (AVC) task
- Use audio embedding to train downstream task (classifier for limited data)
- Downstream datasets:
  - *US8K*: 8732 audio clips divided into 10 cross-validation folds
  - *ESC-50*: 2000 clips divided into 5 folds
- Downstream Accuracy
  - US8K: 75.91% | ESC-50: 73.65%
- L^3-Net audio has 4,688,066 parameters and is 18 MB

[1] Arandjelovic, Relja and Zisserman, Andrew. "Look, Listen and Learn". IEEE ICCV. 2017.

Outline

Non-sparse Audio Model

- **Depth Reduction**:
  - *conv8* has 2,359,808 params (50% of total)

[2] Li, Hao et al. "Pruning Filters for Efficient ConvNets." ICLR. 2017.

Non-sparse Audio Model

- **Depth Reduction**:
  - *conv8* has 2,359,808 params (50% of total)
  - Embedding could be generated from penultimate layer or before

[2] Li, Hao et al. "Pruning Filters for Efficient ConvNets." ICLR. 2017.

Non-sparse Audio Model

- **Depth Reduction**:
  - *conv8* has 2,359,808 params (50% of total)
  - Embedding could be generated from penultimate layer or before
- **Width Reduction**:
  - Filters with smaller kernel weights produce feature maps with weaker activations [2]

[2] Li, Hao et al. "Pruning Filters for Efficient ConvNets." ICLR. 2017.

Non-sparse Audio Model

- **Depth Reduction**:
  - *conv8* has 2,359,808 params (50% of total)
  - Embedding could be generated from penultimate layer or before
- **Width Reduction**:
  - Filters with smaller kernel weights produce feature maps with weaker activations [2]
  - Drop kernels whose absolute weight sum is less than a threshold value.

[2] Li, Hao et al. "Pruning Filters for Efficient ConvNets." ICLR. 2017.

Sparse Audio Model

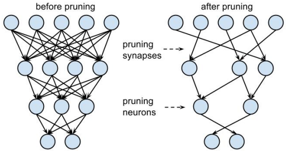

Figure 3: Pruning Weights [3]

- Prune potentially unimportant connections [3]
  - Zero out the weights whose absolute magnitude < threshold

[3] Han, Song, et al. "Learning both weights and connections for efficient neural network." NIPS. 2015.

Sparse Audio Model

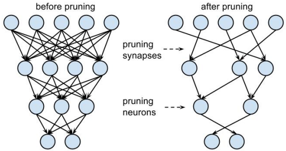

Figure 3: Pruning Weights [3]

- Prune potentially unimportant connections [3]
  - Zero out the weights whose absolute magnitude < threshold

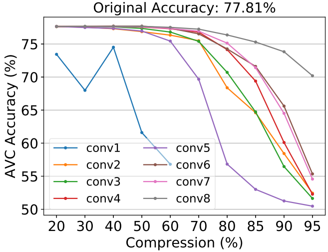

Figure 4: Impact of pruning individual layers on AVC accuracy

- Prune each layer independently with a range of sparsity values to determine sensitivity of each
  - *conv1* most sensitive
  - *conv8* least sensitive

[3] Han, Song, et al. "Learning both weights and connections for efficient neural network." NIPS. 2015.

Re-Training Methodology

- **Fine-tuning**: Retrain the L^3-Net for AVC task with the pruned audio model while freezing the video model

Re-Training Methodology

- **Fine-tuning**: Retrain the L^3-Net for AVC task with the pruned audio model while freezing the video model
- **Knowledge Distillation**: Minimize the Mean Square Error loss between embedding from original L^3-Net and sparse L^3-Net audio

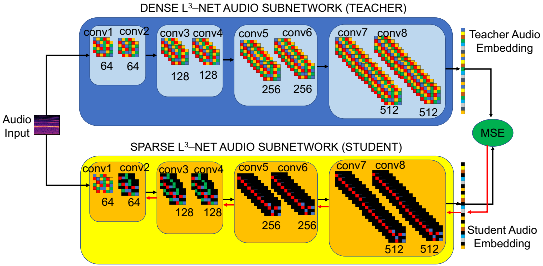

Figure 5: Knowledge Distillation setup with original L^3 audio as teacher and pruned audio model as student for audio embedding approximation.

Outline

Depth Reduction

Given a trained L^3-Net audio, generate embedding out of 7^{th}, 6^{th} or 5^{th} layer

Reduction In	**Num. Filters in Audio Convolution Layers**								**Reduction in Weights (%)**	**Accuracy (%)**	
	conv 1	**conv 2**	**conv 3**	**conv 4**	**conv 5**	**conv 6**	**conv 7**	**conv 8**		**US8K**	**ESC-50**
Original	64	64	128	128	256	256	512	512	NA	75.91	73.65
Depth	64	64	128	128	256	256	512		50.42	74.38	74
	64	64	128	128	256	256			72.34	71.74	68.7
	64	64	128	128	256				86.66	68.77	66.6

Table 1: Downstream accuracy of L^3-Net depth reduction experiments.

Width Reduction

Reduction In	Num. Filters in Audio Convolution Layers								Reduction in Weights (%)	US8K		ESC-50	
	conv 1	conv 2	conv 3	conv 4	conv 5	conv 6	conv 7	conv 8		Before FT (%)	After FT (%)	Before FT (%)	After FT (%)
Original	64	64	128	128	256	256	512	512	NA	75.91	NA	73.65	NA
Width	64	48	64	64	128	128	256	256	43.14	51.39	74	34	71.25
	64	48	64	64	128	128	128	128	64.54	52.16	71.45	33.3	64.7
	64	48	64	64	64	64	128	128	69.89	48.87	72.46	32.6	67.2

Table 2: Downstream accuracy of L^3-Net before and after fine-tuning

Sparse Models

Sparsities in Audio Convolution Layers (%)								Reduction in Weights (%)	Memory (MB)
conv1	**conv2**	**conv3**	**conv4**	**conv5**	**conv6**	**conv7**	**conv8**		
0	0	0	0	0	0	0	0	NA	18
0	30	40	50	30	50	50	60	53.49	8.317
0	40	50	60	40	60	60	70	63.48	6.530
0	40	50	60	40	70	70	80	72.29	4.955
0	60	60	70	50	70	70	80	73.55	4.730
0	70	70	75	60	80	80	85	80.87	3.421
0	80	80	85	40	85	85	95	87.08	2.310
30	85	85	90	60	90	90	95	90.51	1.697
0	85	85	85	75	95	98	98	95.45	0.814
0	93	94	96	97	95.97	98	97	97.00	0.536
0	95	96	97	97	98	98.65	98	98.00	0.357
0	94	96	99	99	99	99	99.2	99.00	0.179

Table 3: Decomposition plan for L^3-Net audio subnetwork layerwise pruning. The first row corresponds to the original model with 18MB weights.

Re-training Performance on AVC

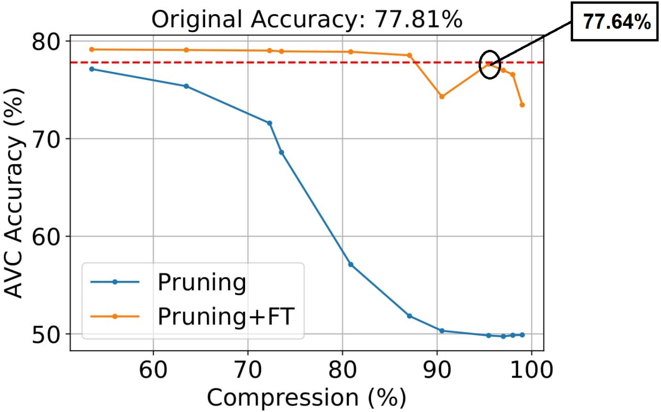

Figure 6: Improvement in L^3-Net AVC through fine-tuning (FT). The red dotted line corresponds to the baseline model performance.

EdgeL^3

Sparsities in Audio Convolution Layers (%)								Reduction in Weights (%)	Memory (MB)
conv1	**conv2**	**conv3**	**conv4**	**conv5**	**conv6**	**conv7**	**conv8**		
0	0	0	0	0	0	0	0	NA	18
0	30	40	50	30	50	50	60	53.49	8.317
0	40	50	60	40	60	60	70	63.48	6.530
0	40	50	60	40	70	70	80	72.29	4.955
0	60	60	70	50	70	70	80	73.55	4.730
0	70	70	75	60	80	80	85	80.87	3.421
0	80	80	85	40	85	85	95	87.08	2.310
30	85	85	90	60	90	90	95	90.51	1.697
0	85	85	85	75	95	98	98	95.45	0.814
0	93	94	96	97	95.97	98	97	97.00	0.536
0	95	96	97	97	98	98.65	98	98.00	0.357
0	94	96	99	99	99	99	99.2	99.00	0.179

Table 4: EdgeL^3 audio model is only 0.8MB and meets our flash memory constraint

Re-training Performance on Downstream Task

Improvements on downstream tasks through fine-tuning (FT) and knowledge distillation (KD)

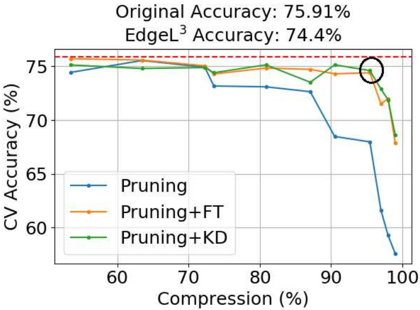

Figure 7: UrbanSound8K

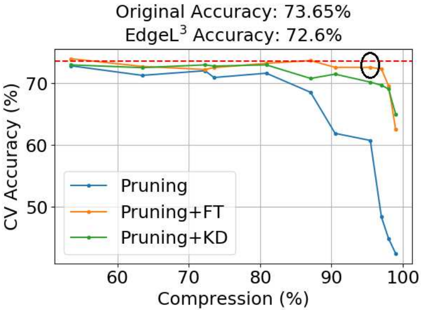

Figure 8: ESC-50

Outline

Mote-scale Implementation

- Fixed-point quantization

Mote-scale Implementation

- Fixed-point quantization
- Change in input representation

Mote-scale Implementation

- Fixed-point quantization
- Change in input representation
- Incremental computation of activations

Outline

edgel3 Python Package

- Reference model for generating audio embedding for Edge computing
- *pip install edgel3*

```python
import edgel3
import soundfile as sf

audio, sr = sf.read('/path/to/file.wav')

# Get embedding out of EdgeL3 (95.45% sparse fine-tuned L3)
emb, ts = edgel3.get_embedding(audio, sr, retrain_type='ft',
                                sparsity=95.45)

#Get embedding out of 81.0% sparse knowledge distilled L3 audio
emb, ts = edgel3.get_embedding(audio, sr, retrain_type='kd',
                                sparsity=81.0)
```

Outline

Conclusion

- EdgeL^3 has 213,491 parameters and is only 0.814 MB

Conclusion

- EdgeL^3 has 213,491 parameters and is only 0.814 MB
- Negligible loss of 0.22% in AVC performance and 1.4% (1.9%) drop in the ESC-50 (US8K)

Conclusion

- EdgeL^3 has 213,491 parameters and is only 0.814 MB
- Negligible loss of 0.22% in AVC performance and 1.4% (1.9%) drop in the ESC-50 (US8K)
- Pruned models made available in *edgel3* package

Conclusion

- EdgeL^3 has 213,491 parameters and is only 0.814 MB
- Negligible loss of 0.22% in AVC performance and 1.4% (1.9%) drop in the ESC-50 (US8K)
- Pruned models made available in *edgel3* package
- Ongoing work for a realistic mote scale realization of EdgeL^3

Questions?