MOSCOW INSTITUTE OF PHYSICS AND TECHNOLOGY

Speech Technology 2022

Lecture #5
STFT & Keyword Spotting

@georgygospodinov

Plan

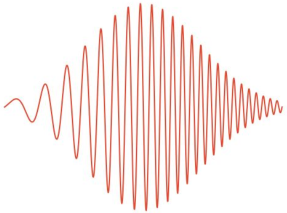
- From Fourier Transform to Short Time Fourier Transform
  - empirical examples
  - windowing
  - scaling
- Keyword Spotting
  - problem formulation
  - challenges
  - metrics
  - architectures
  - benchmarks
- Homework

Why do we need spectrograms?

$$e^{-\frac{(t-T/2)^2}{\beta}} \sin(2\pi\nu t + \alpha t^2)$$

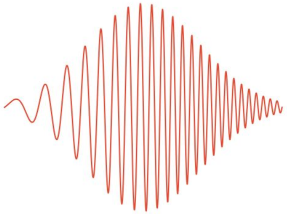
Why do we need spectrograms?

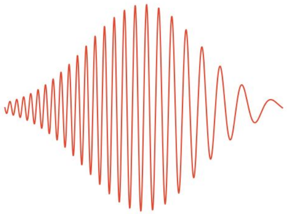
$$e^{-\frac{(t-T/2)^2}{\beta}} \sin(2\pi\nu t + \alpha t^2)$$



Why do we need spectrograms?

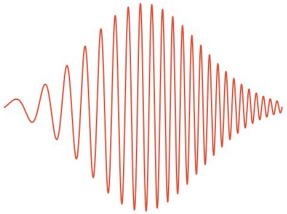
$$e^{-\frac{(t-T/2)^2}{\beta}} \sin(2\pi\nu t + \alpha t^2)$$

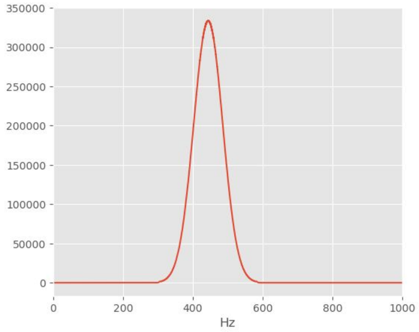


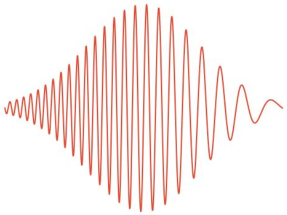


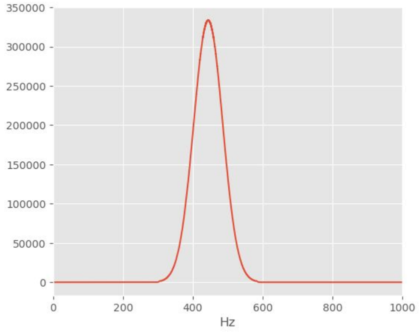
Why do we need spectrograms?

$$e^{-\frac{(t-T/2)^2}{\beta}} \sin(2\pi\nu t + \alpha t^2)$$



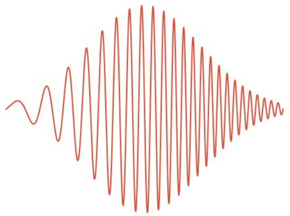


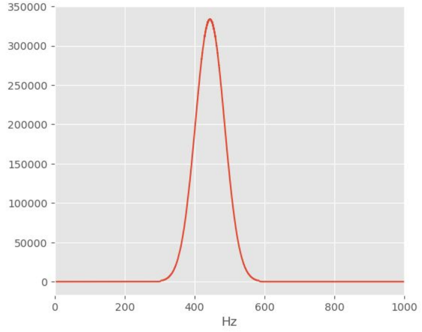


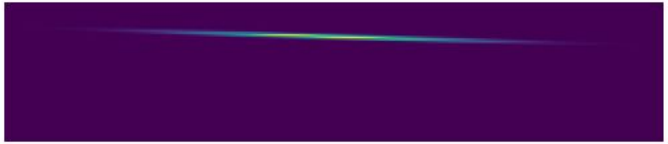


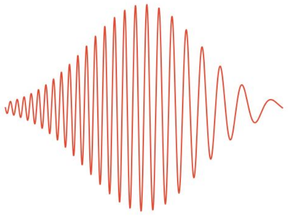
Why do we need spectrograms?

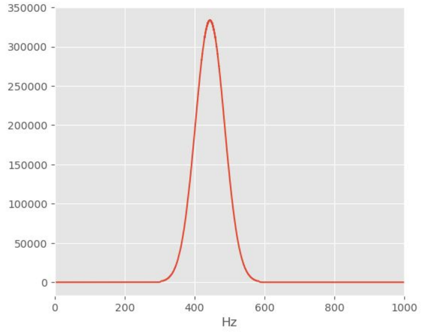
$$e^{-\frac{(t-T/2)^2}{\beta}} \sin(2\pi\nu t + \alpha t^2)$$



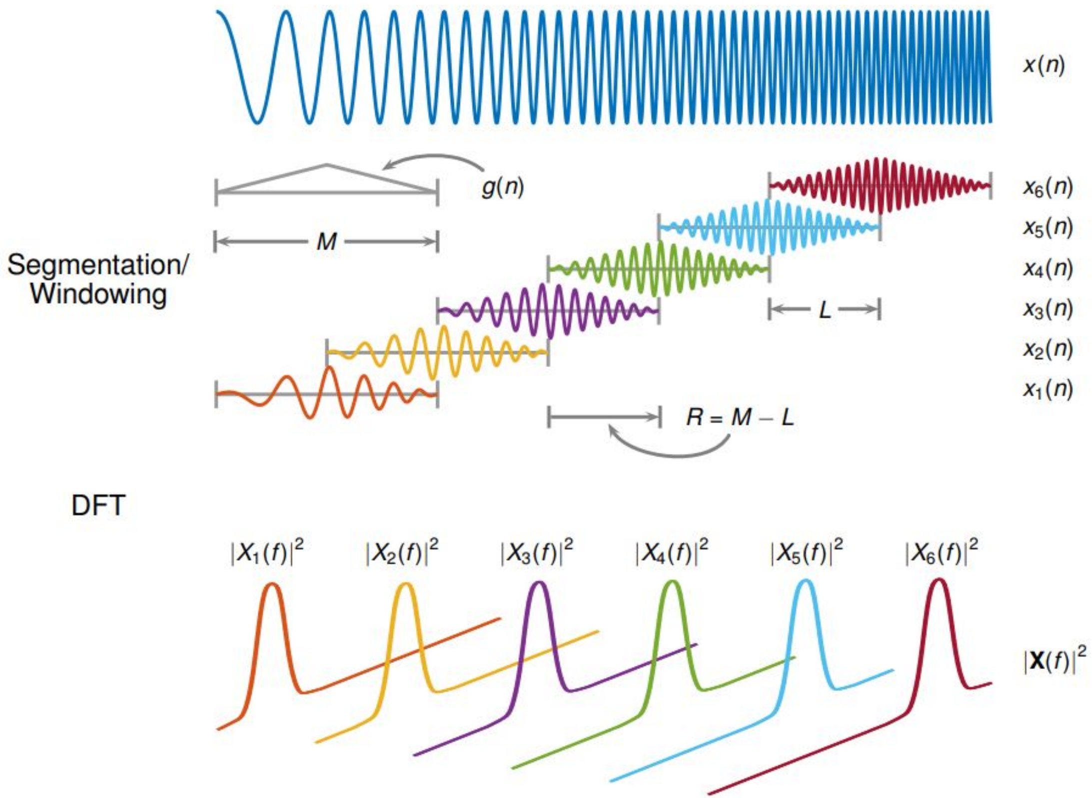






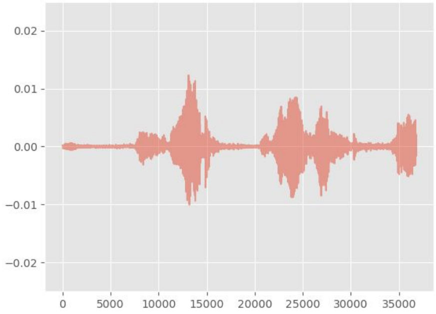


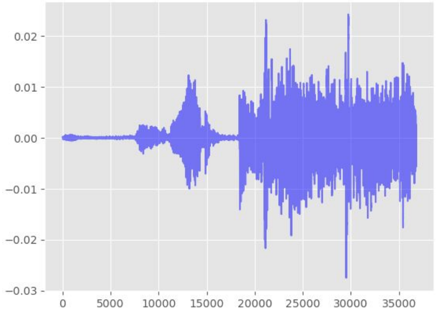
Spectrogram: Short Time Fourier Transform



Why do we need spectrograms?

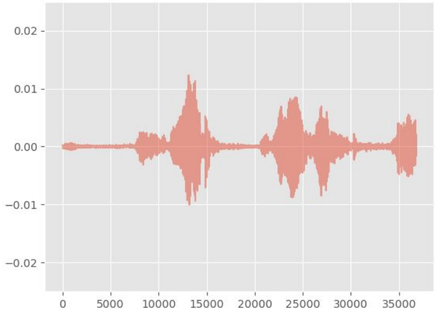
Time Domain

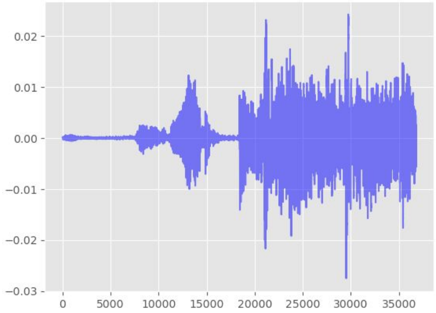




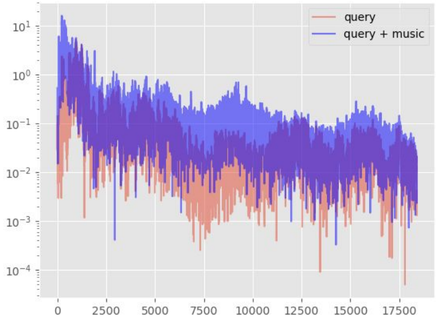
Why do we need spectrograms?

Time Domain



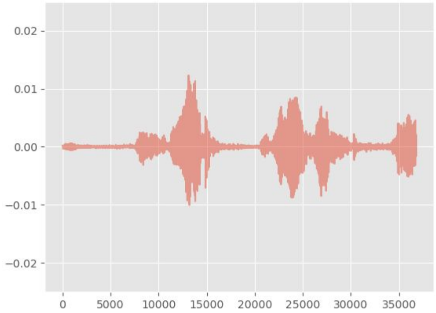


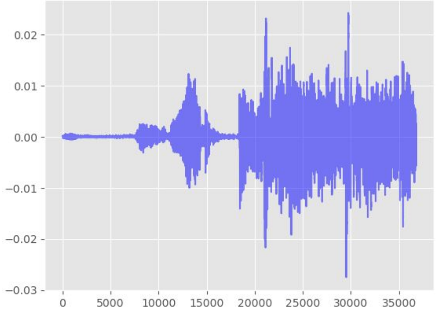
FFT



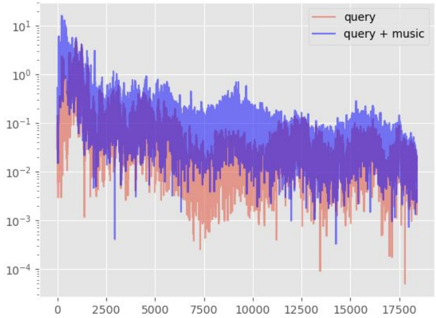
Why do we need spectrograms?

Time Domain

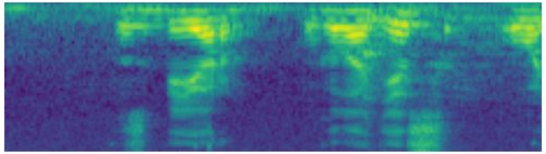


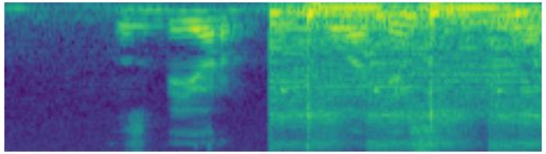


FFT

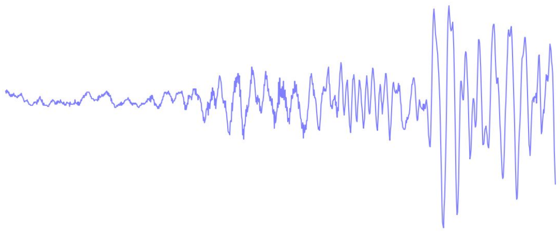


Log Mel STFT

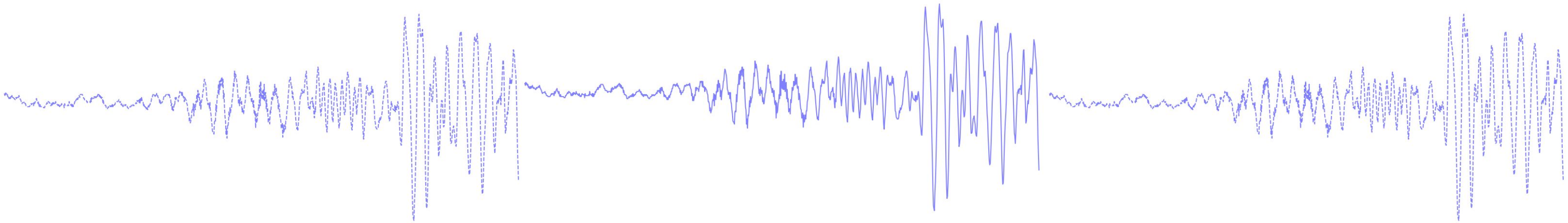




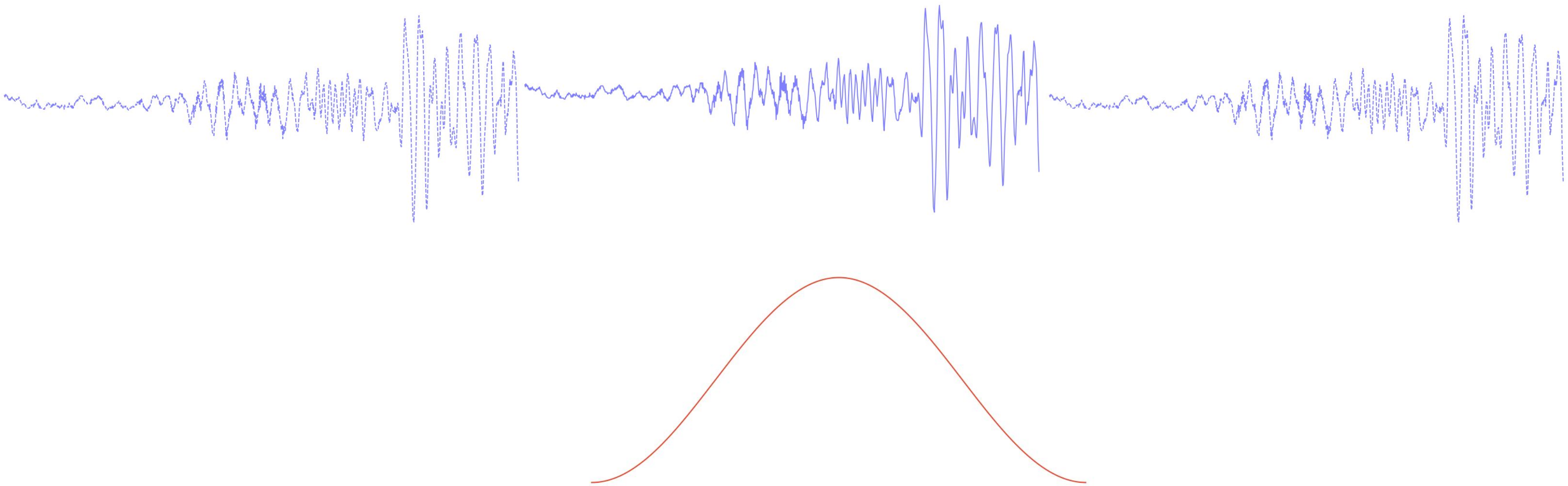
Spectrogram: Windowing



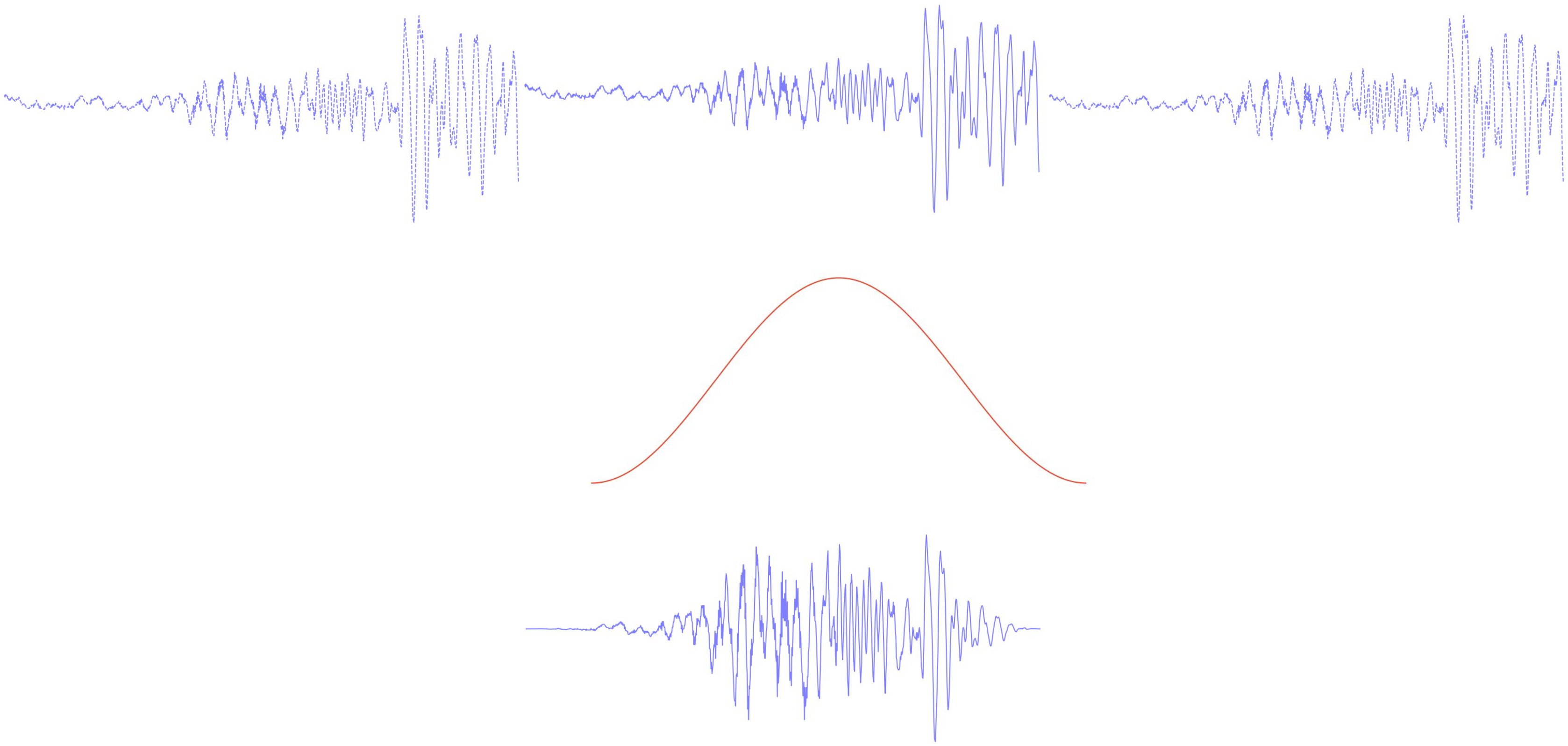
Spectrogram: Windowing



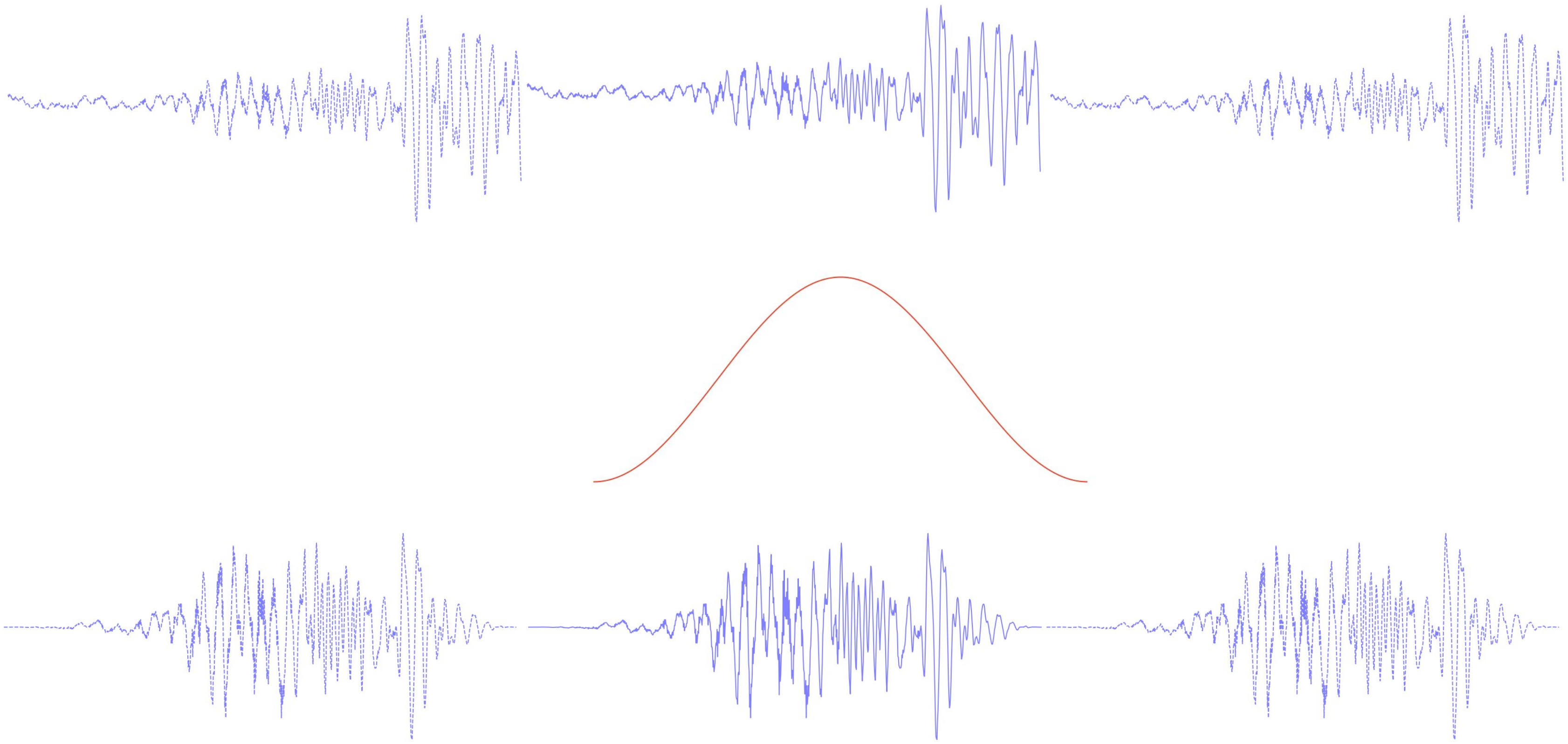
Spectrogram: Windowing



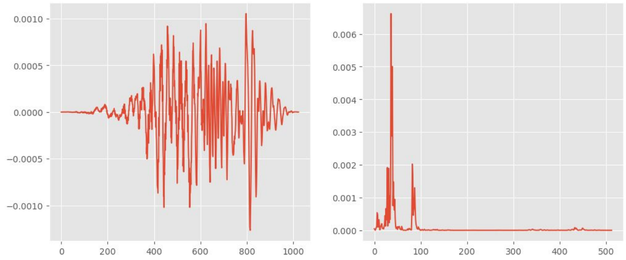
Spectrogram: Windowing



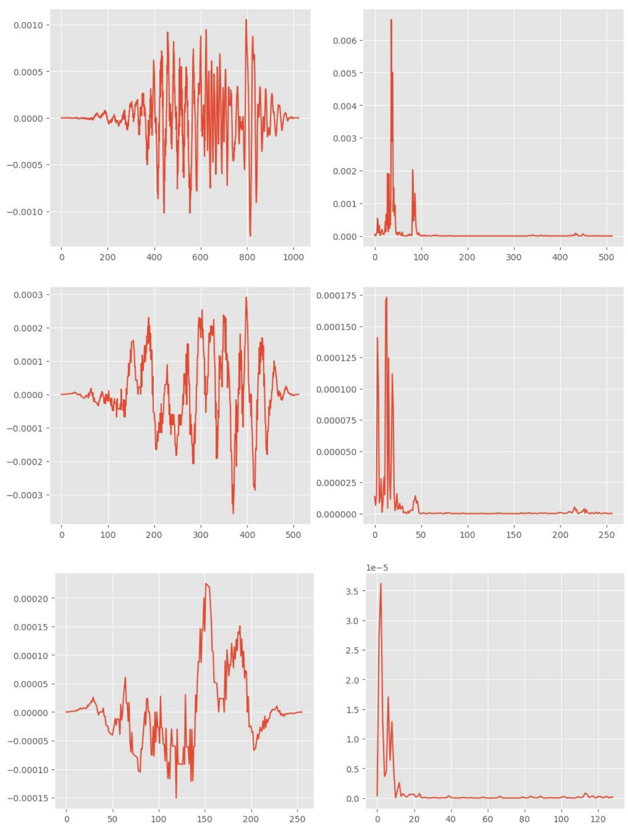
Spectrogram: Windowing



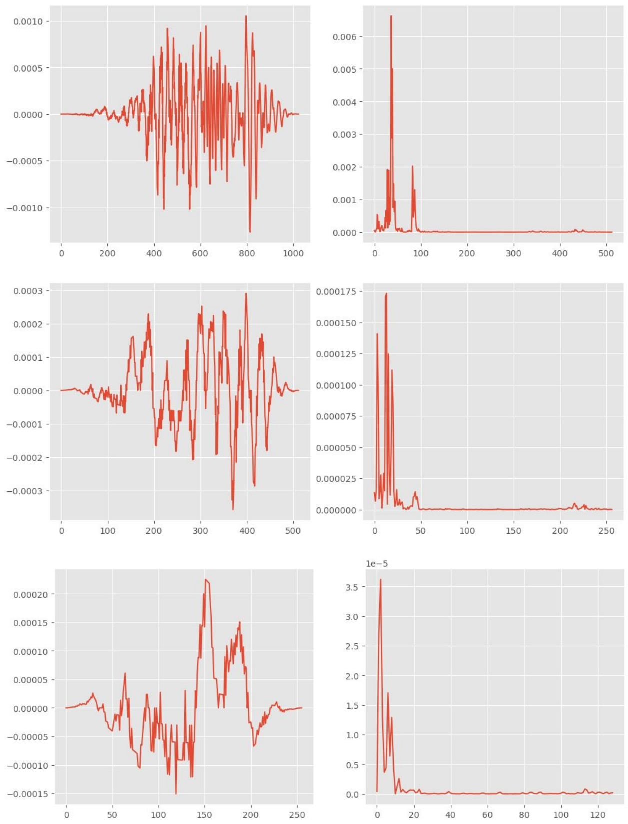
Spectrogram: window length?



Spectrogram: window length?

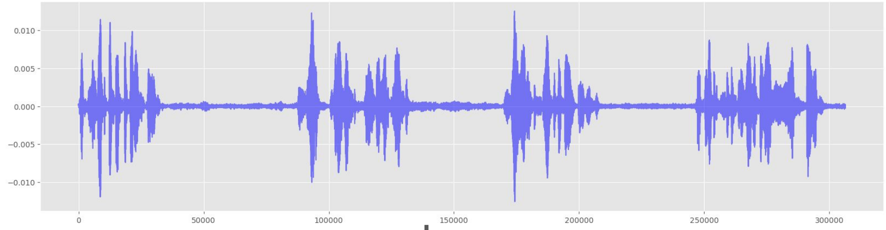


Spectrogram: window length?

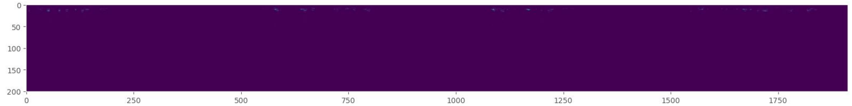


- tradeoff between time-domain and frequency-domain resolution
- typical values:
  - window_length = 25ms
  - hop_length = 10ms

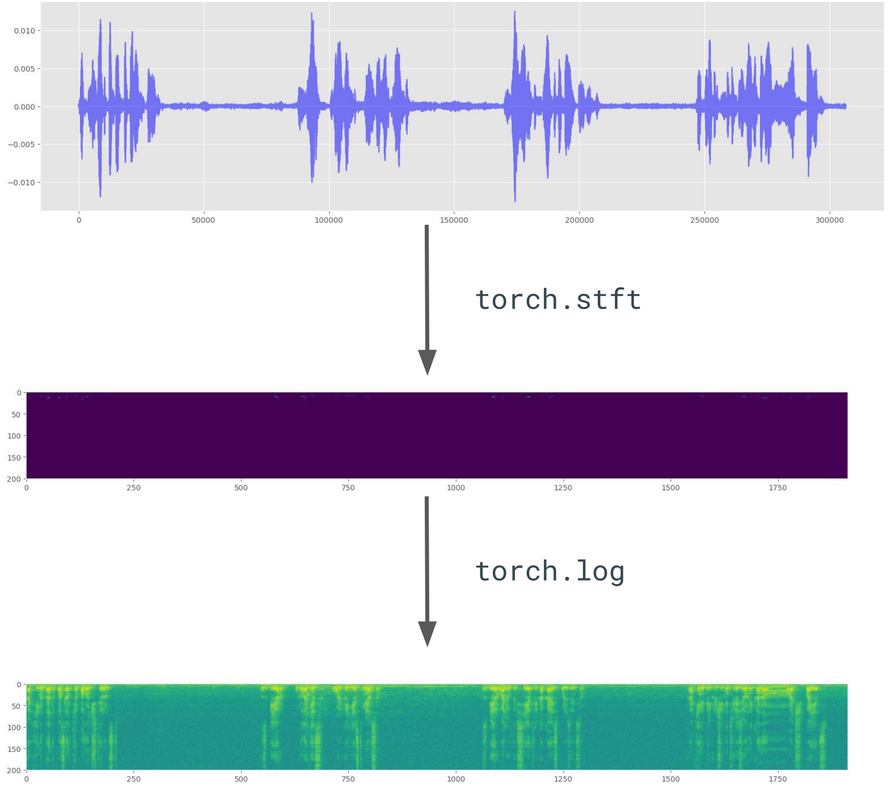
Spectrogram: Logarithmic Scale



torch.stft



Spectrogram: Logarithmic Scale



- inspired by human ear sensitivity: we don’t perceive loudness on a linear scale but rather on a logarithmic scale

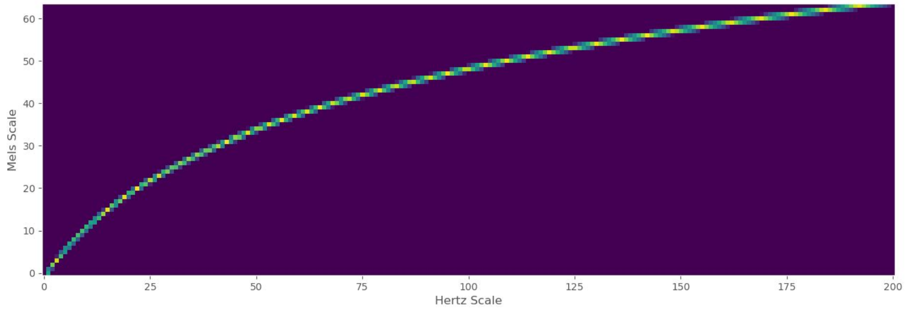
Spectrogram: Mel Scale

$$m[mel] = 1127 \ln(1 + \frac{f[Hz]}{700})$$

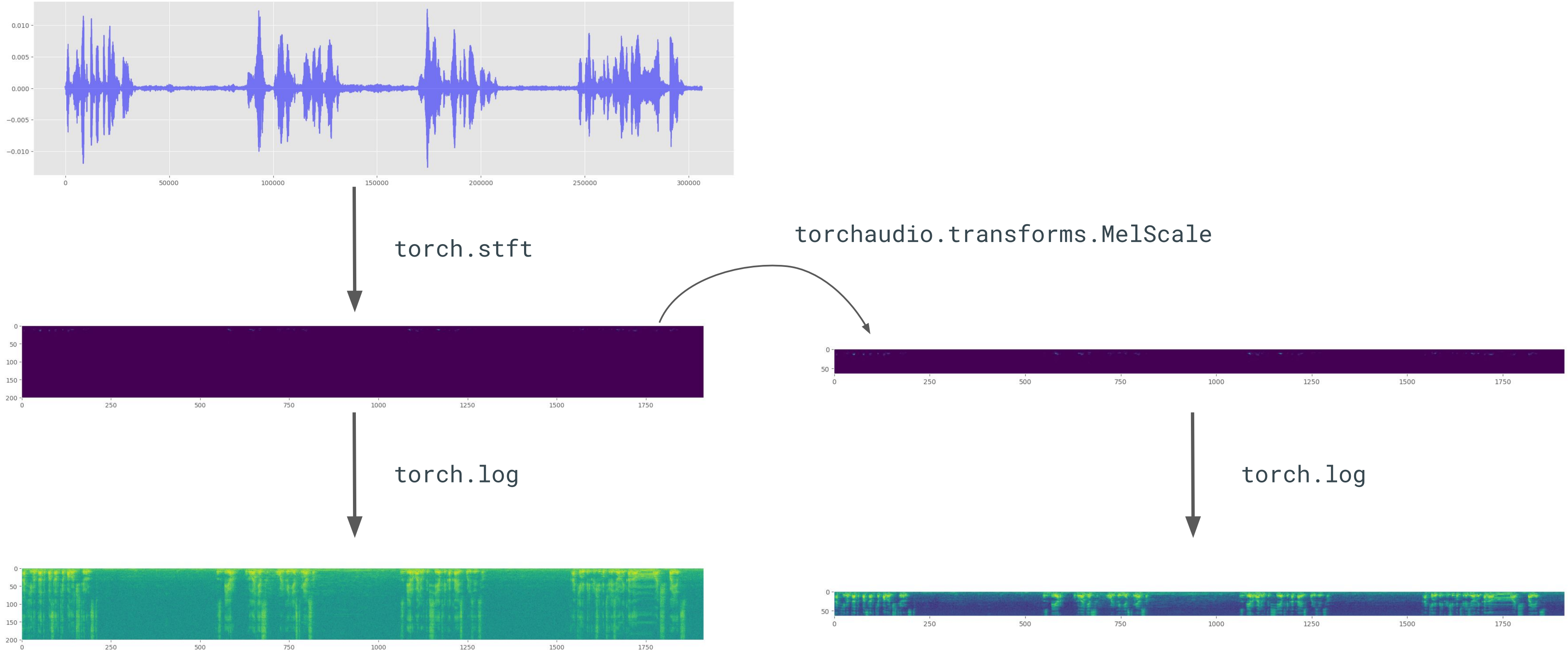
- inspired by human ear sensitivity

```
mel_scaler = torchaudio.transforms.MelScale(
    n_mels=n_mels,
    sample_rate=sr,
    n_stft=n_fft // 2 + 1
)

plt.figure(figsize=(20, 5))
plt.imshow(mel_scaler.fb.T)
plt.xlabel('Hertz Scale')
plt.ylabel('Mels Scale')
plt.grid()
plt.gca().invert_yaxis()
plt.show()
```



Spectrogram: Logarithmic Scale



Spectrogram: summary

- STFT + Mel Scaling + Log Scaling
- Parameters:
  - window_length
  - hop_size
  - window_function
  - n_mels
- Example
  - 1 second wav, sr=16kHz
  - window_length=400, hop_size=160, n_mels=64
  - mel-spectrogram 64x101
- Applications:
  - KWS
  - ASR
  - TTS
  - …

Keyword Spotting

Keyword Spotting

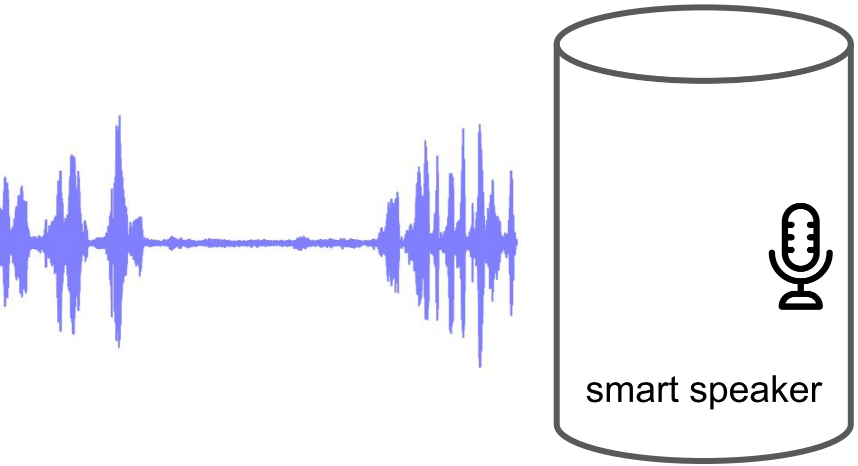
- When user is initiating interaction with device?
- Searching through audio stream by keyword

Keyword Spotting

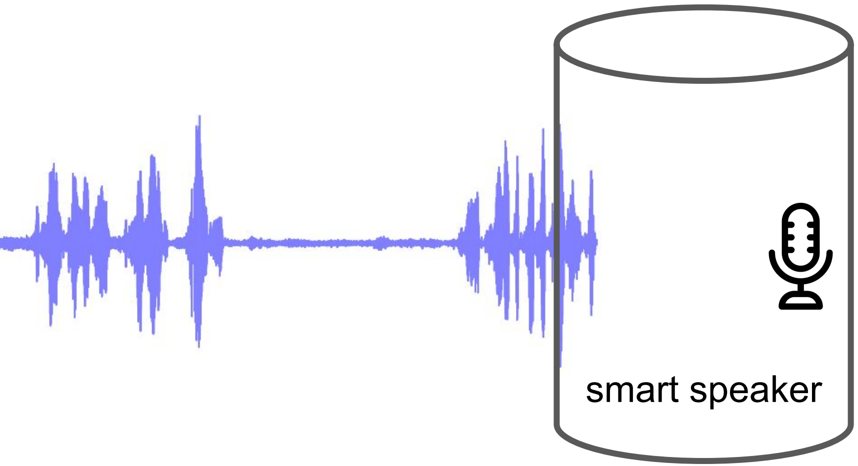
- When user is initiating interaction with device?
- Searching through audio stream by keyword

- Phrase Spotting
- Wake Word Detection
- Wake-Up Word Detection
- Hotword Detection
- Trigger Word Detection

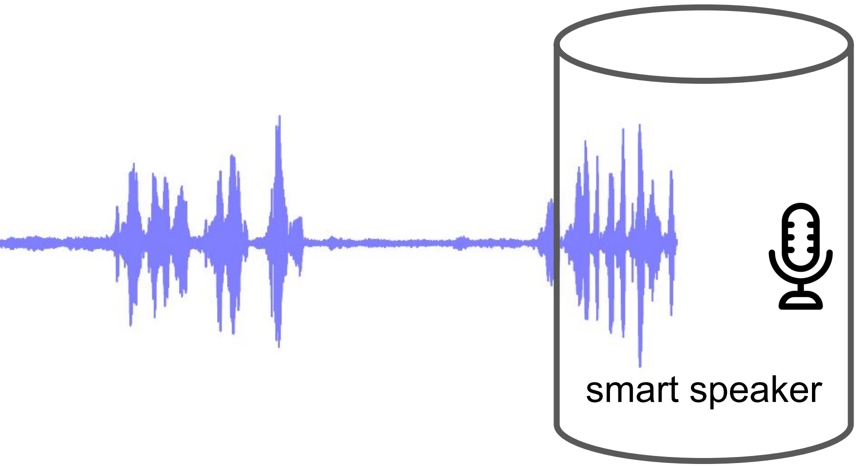
Keyword Spotting



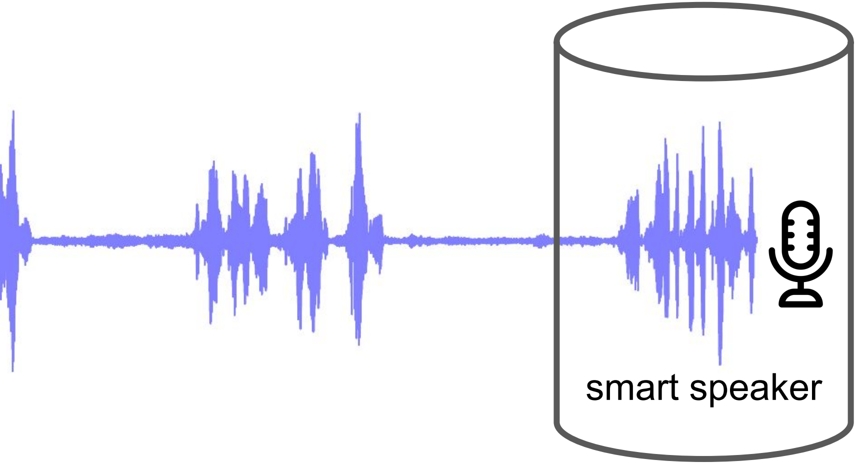
Keyword Spotting



Keyword Spotting

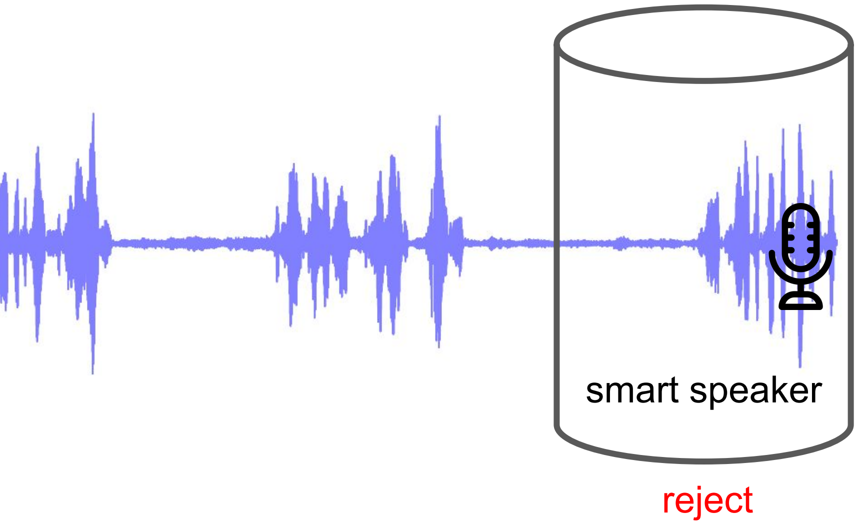


Keyword Spotting

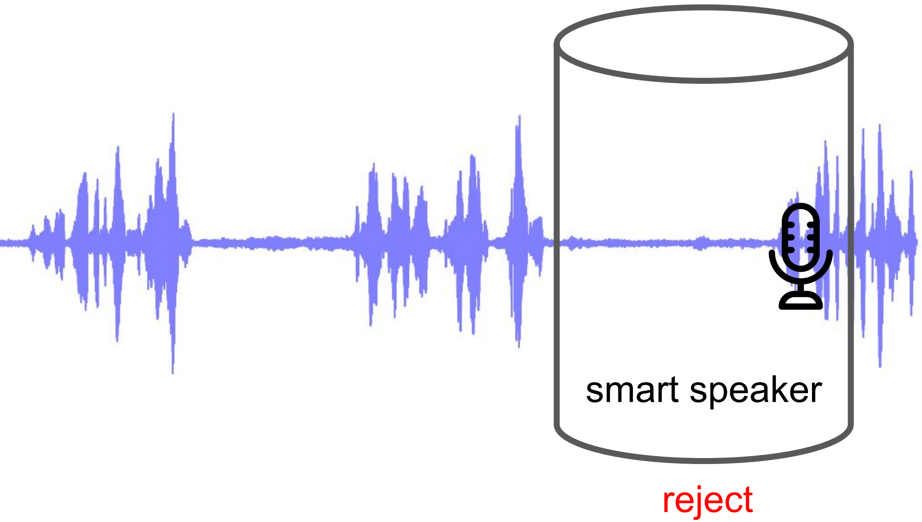


reject

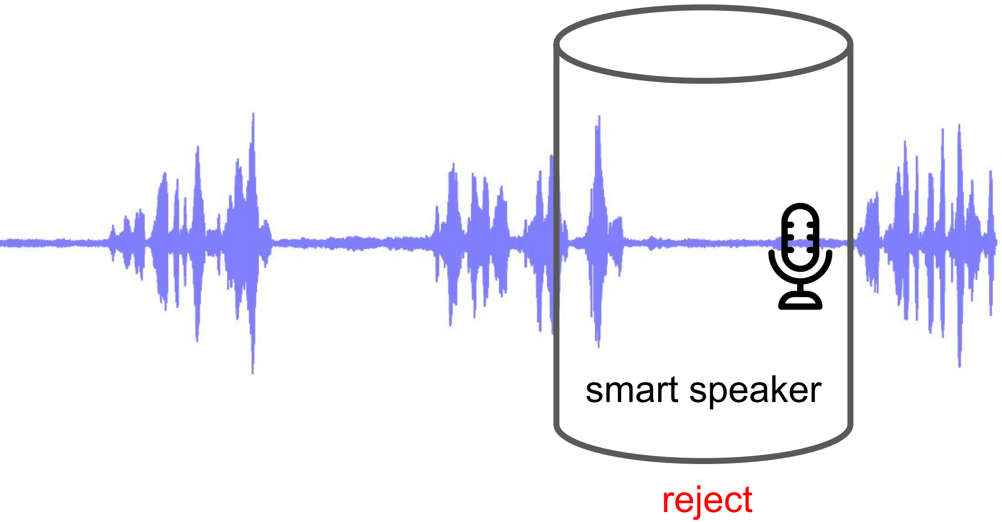
Keyword Spotting



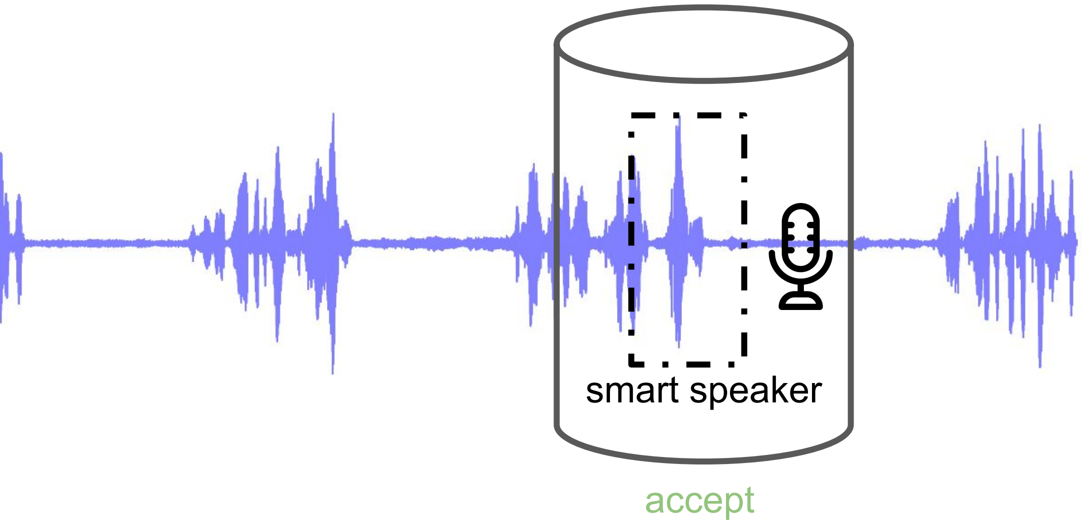
Keyword Spotting



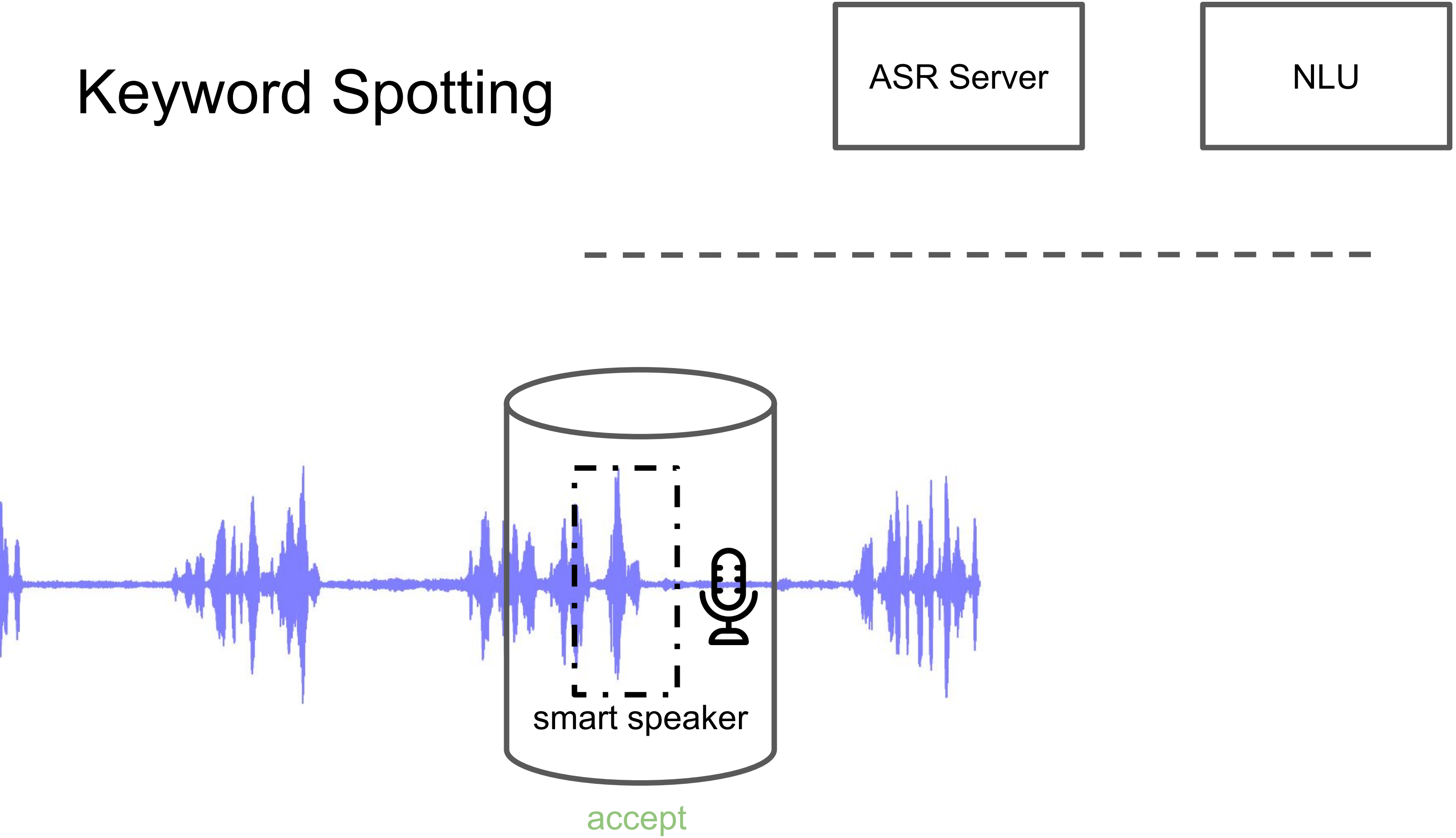
Keyword Spotting



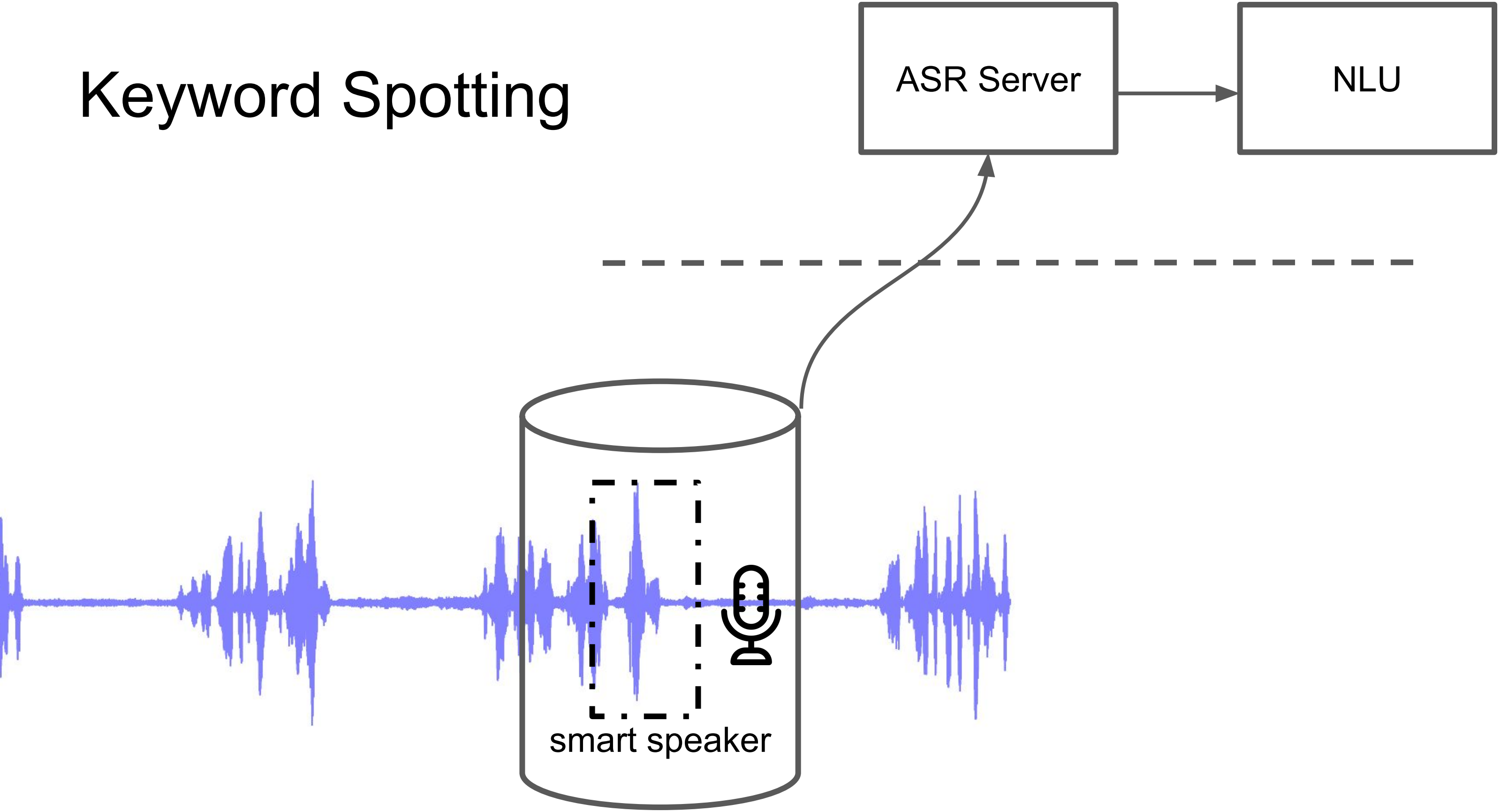
Keyword Spotting



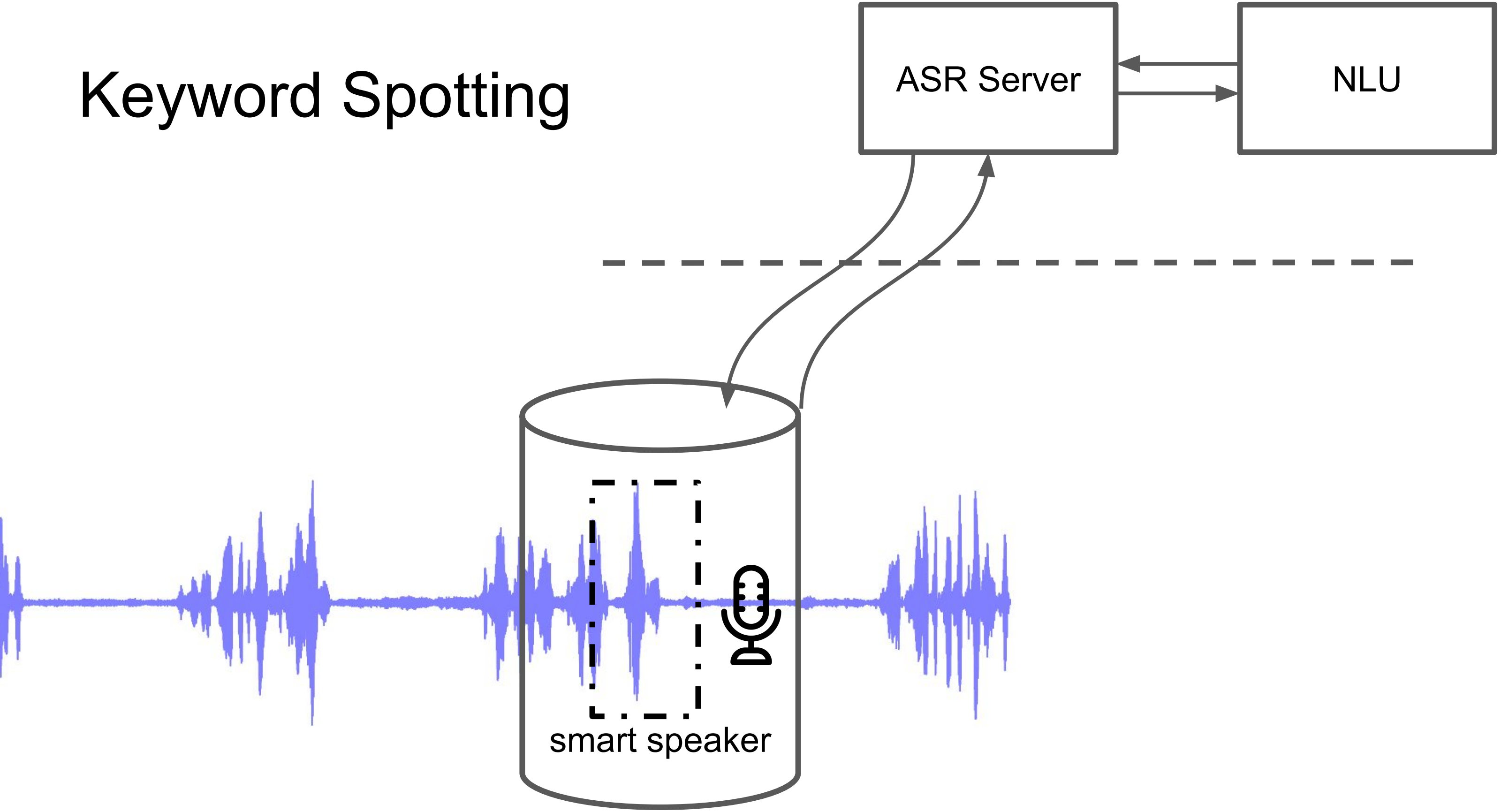
Keyword Spotting



Keyword Spotting



Keyword Spotting

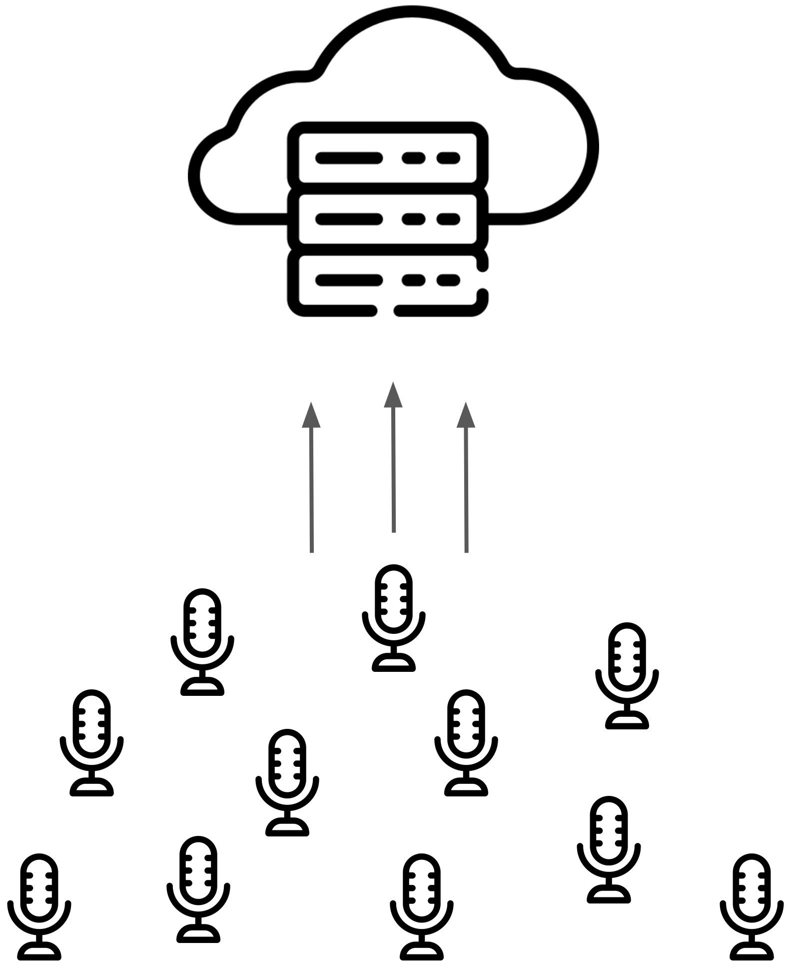


Why on-device?

- Privacy

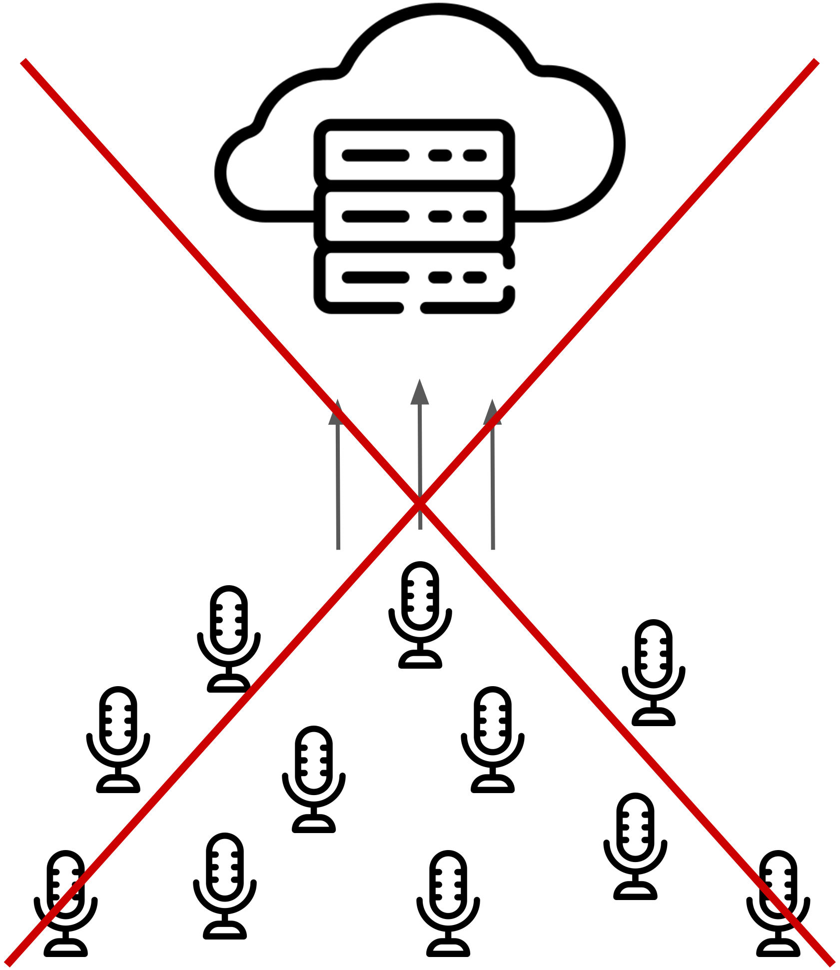
Why on-device?

- Privacy
- Computational Costs

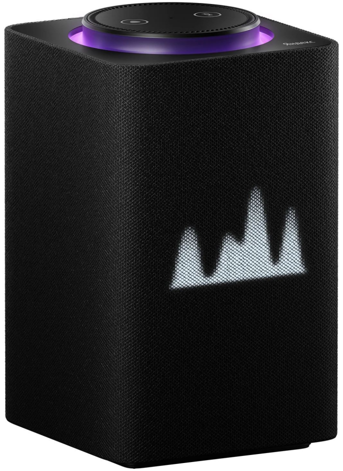


Why on-device?

- Privacy
- Computational Costs

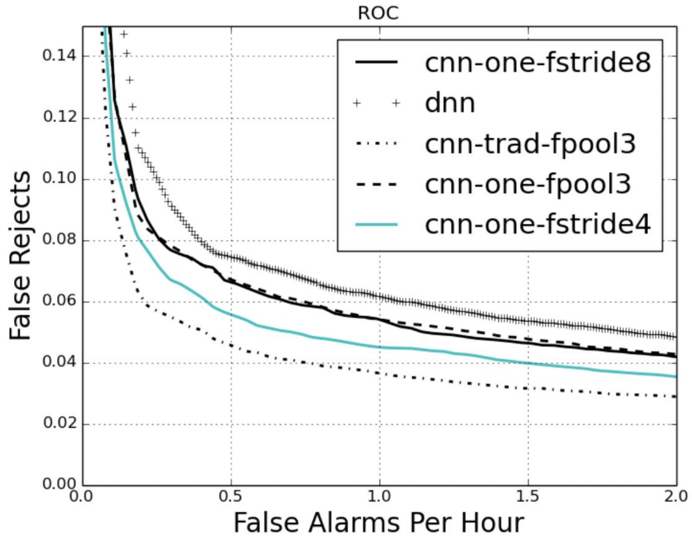


Keyword Spotting: Better, Faster, Smaller



Metrics

- FAR:
  - **F**alse (**A**ccepts / **A**larm / **A**ctivations) **R**ate
  - FAR = FP / (FP + TN)
- FRR
  - **F**alse **R**ejects **R**ate
  - FN / (FN + TP)
- FAh:
  - False Accepts per hour
- lower AUC is better
- model threshold => operation point

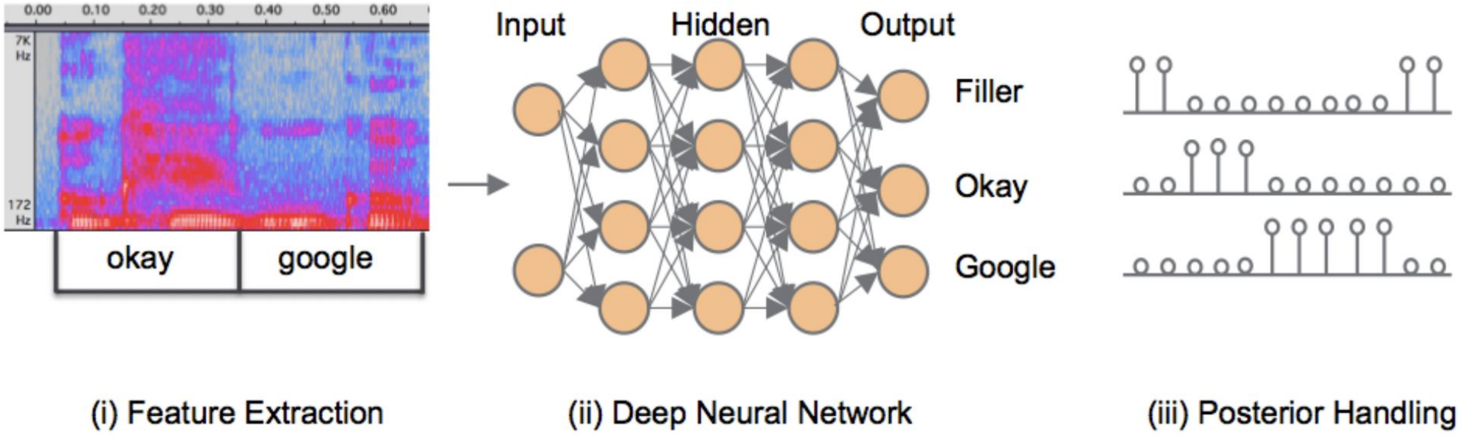


Challenges

- Accuracy: listen continuously, but only trigger at the right moment
  - False Rejects => User Unsatisfaction
  - False Accepts => Privacy
- Latency: Light up immediately
- Robust: low SNR, TV background, music playback, …
- Memory footprint, computational constraints, battery power

SMALL-FOOTPRINT KEYWORD SPOTTING USING DEEP NEURAL NETWORKS

- Google (2014). Cited by 532



[SMALL-FOOTPRINT KEYWORD SPOTTING USING DEEP NEURAL NETWORKS]

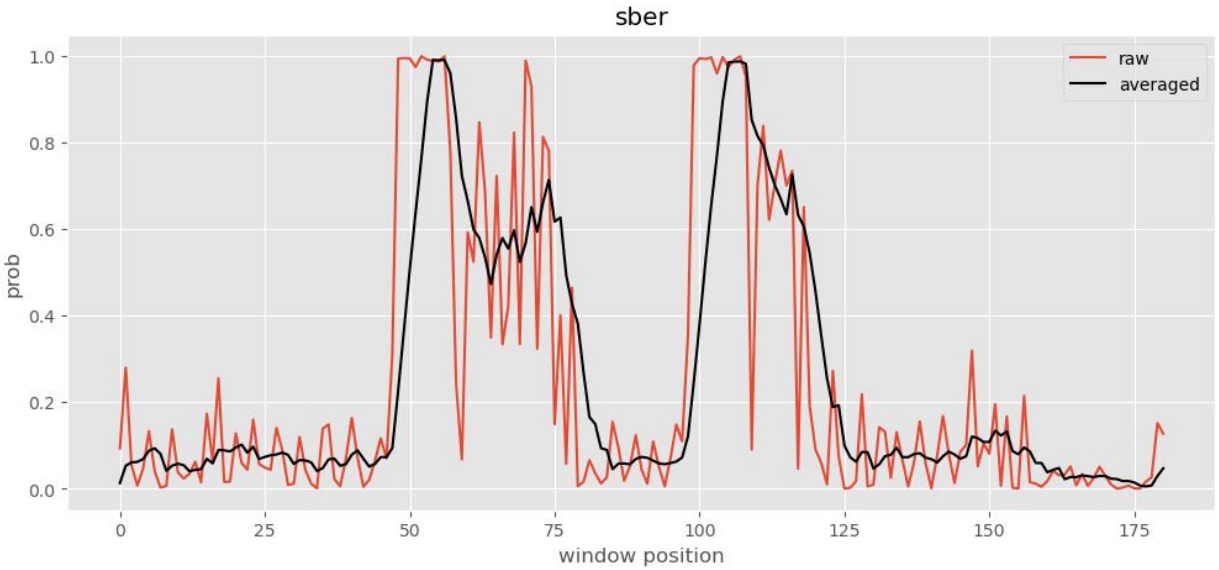
Posterior Handling

$$p'_{ij} = \frac{1}{j - h_{smooth} + 1} \sum_{k=h_{smooth}}^{j} p_{ik}$$

SMALL-FOOTPRINT KEYWORD SPOTTING USING DEEP NEURAL NETWORKS

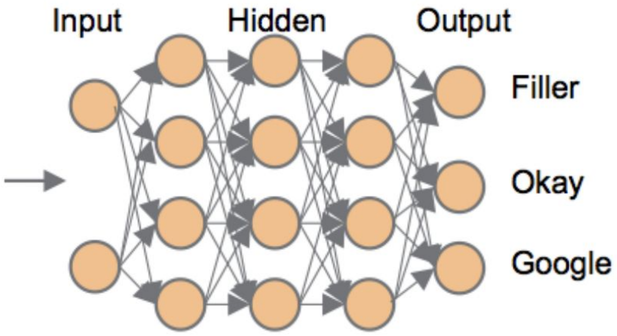
Posterior Handling

$$p'_{ij} = \frac{1}{j - h_{smooth} + 1} \sum_{k=h_{smooth}}^{j} p_{ik}$$



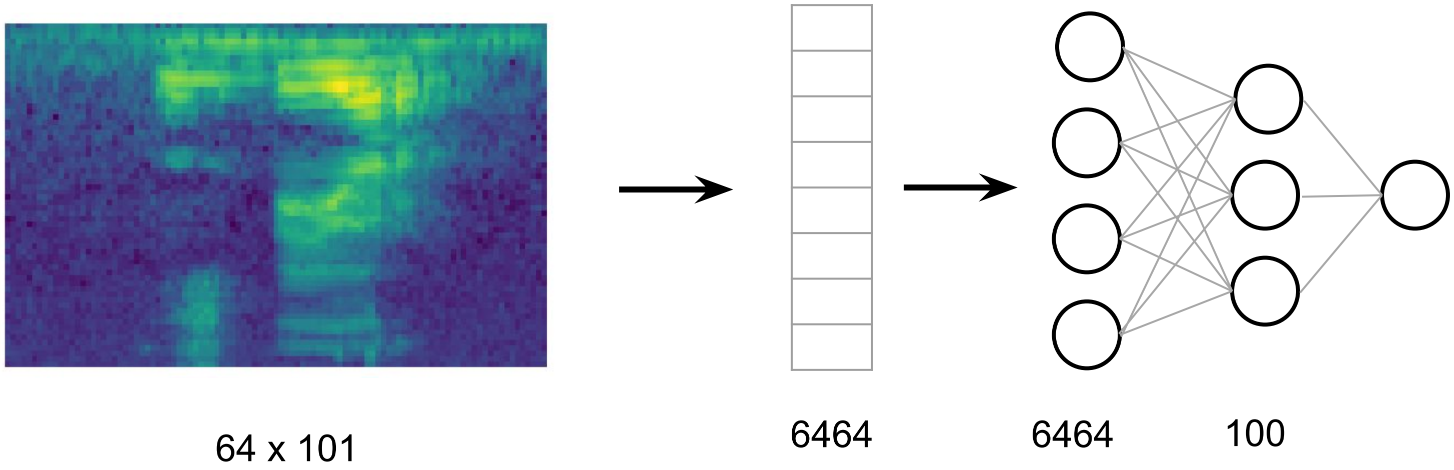
SMALL-FOOTPRINT KEYWORD SPOTTING USING DEEP NEURAL NETWORKS

- Google (2014)
- Problems?

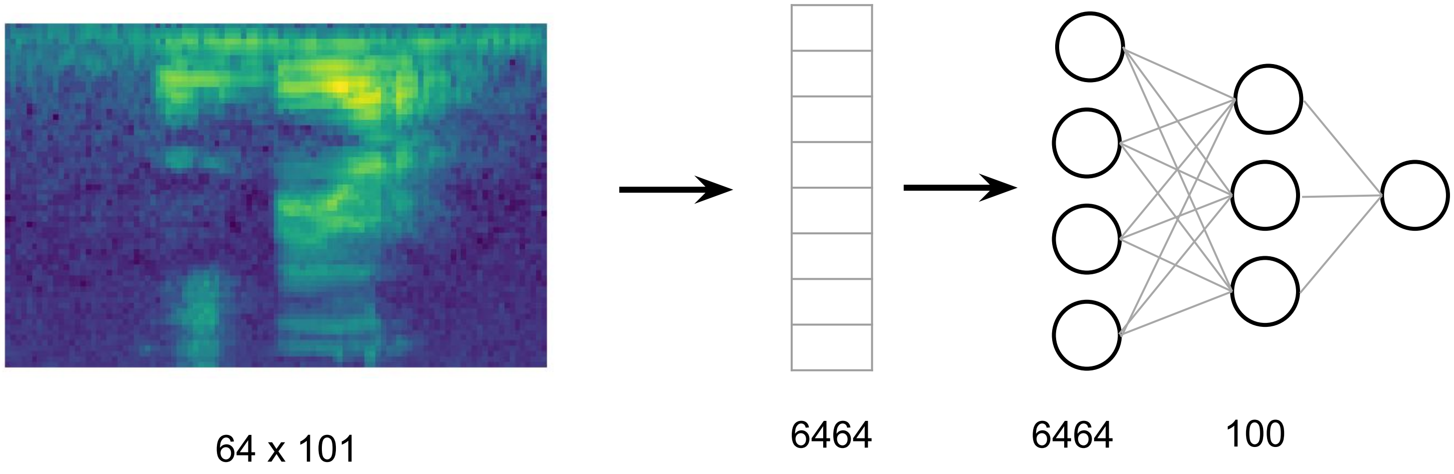


(ii) Deep Neural Network

DNN Problems



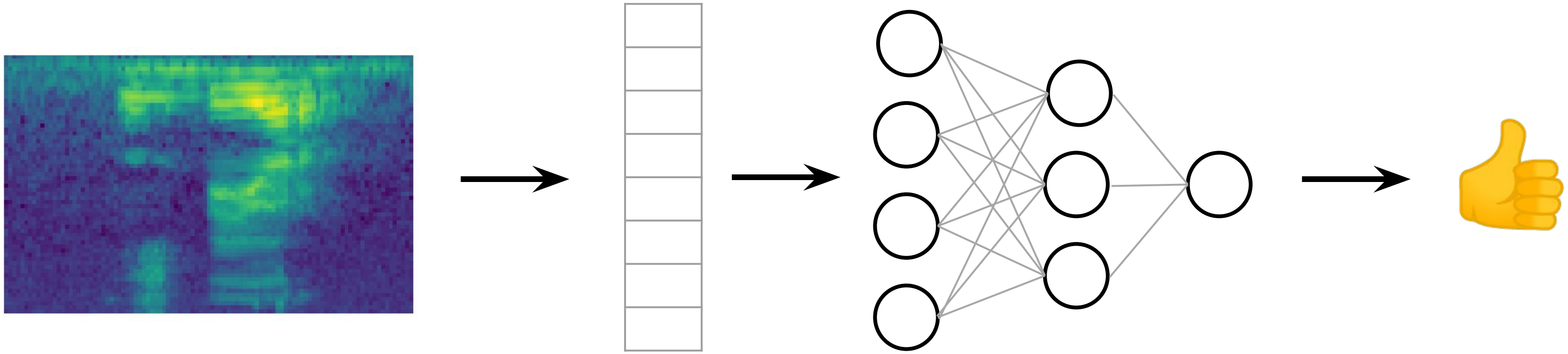
DNN Problems




```
>>> import torch
>>> sum(p.numel() for p in torch.nn.Linear(6464, 100).parameters())
646500
```

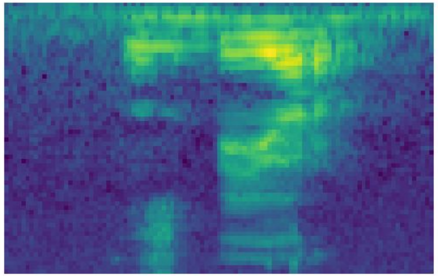
DNN Problems

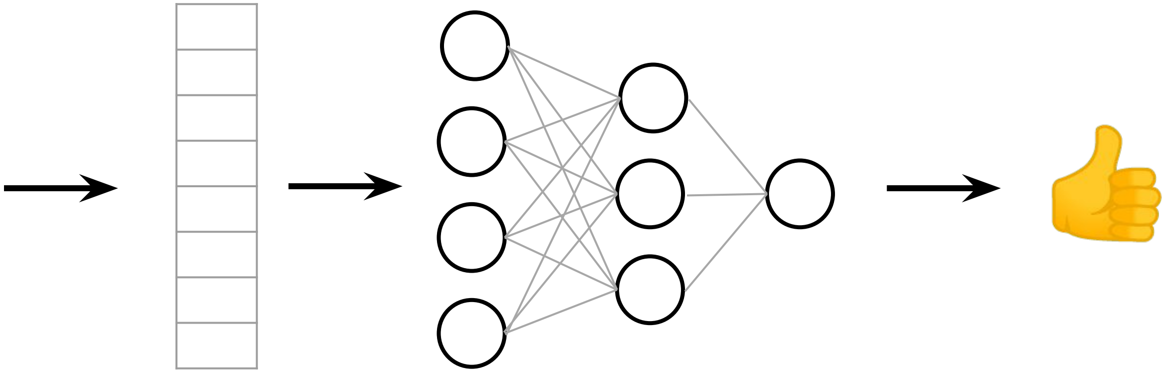
Train



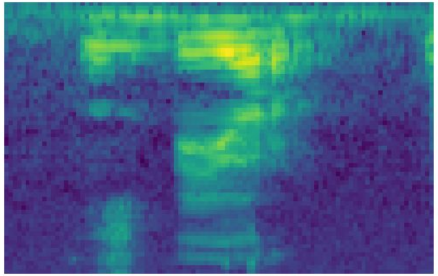
DNN Problems

Train

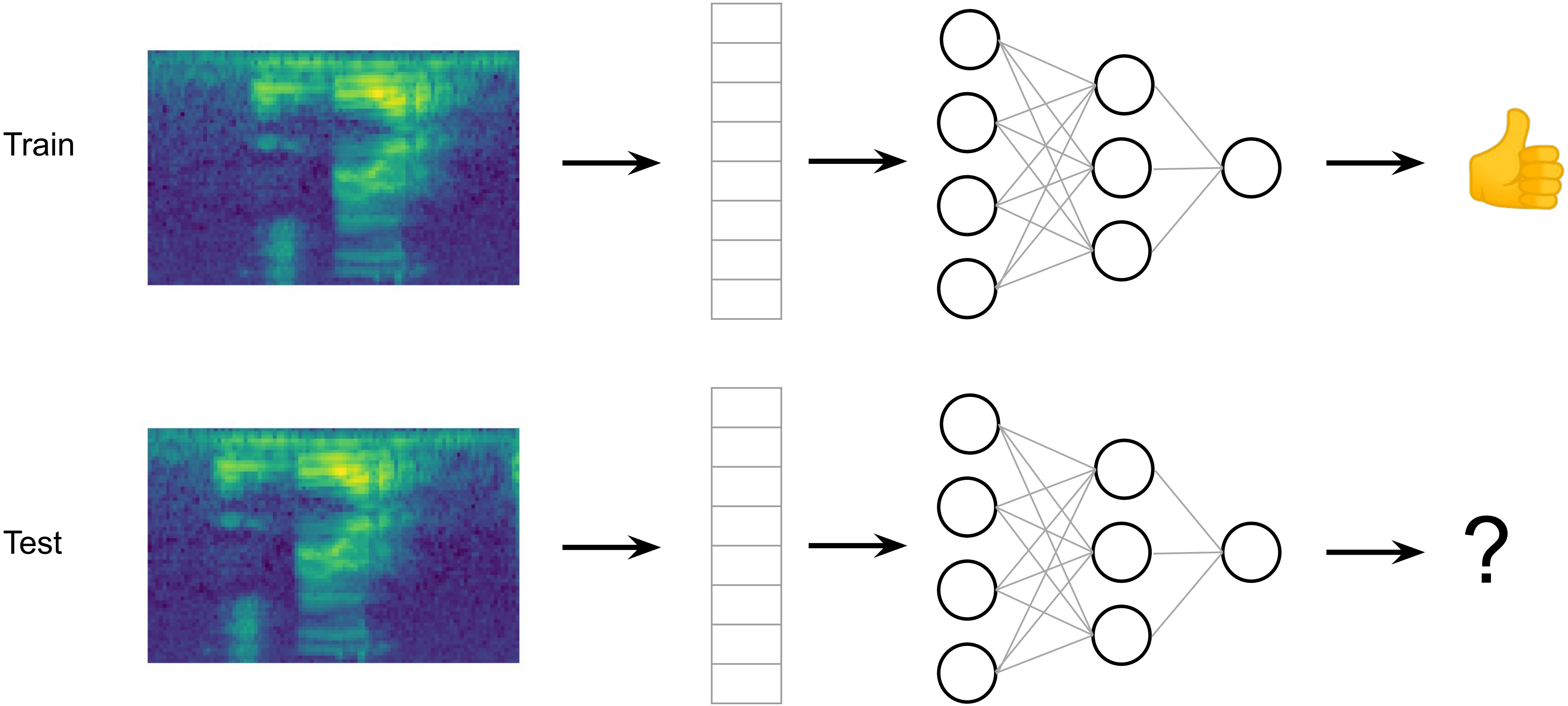




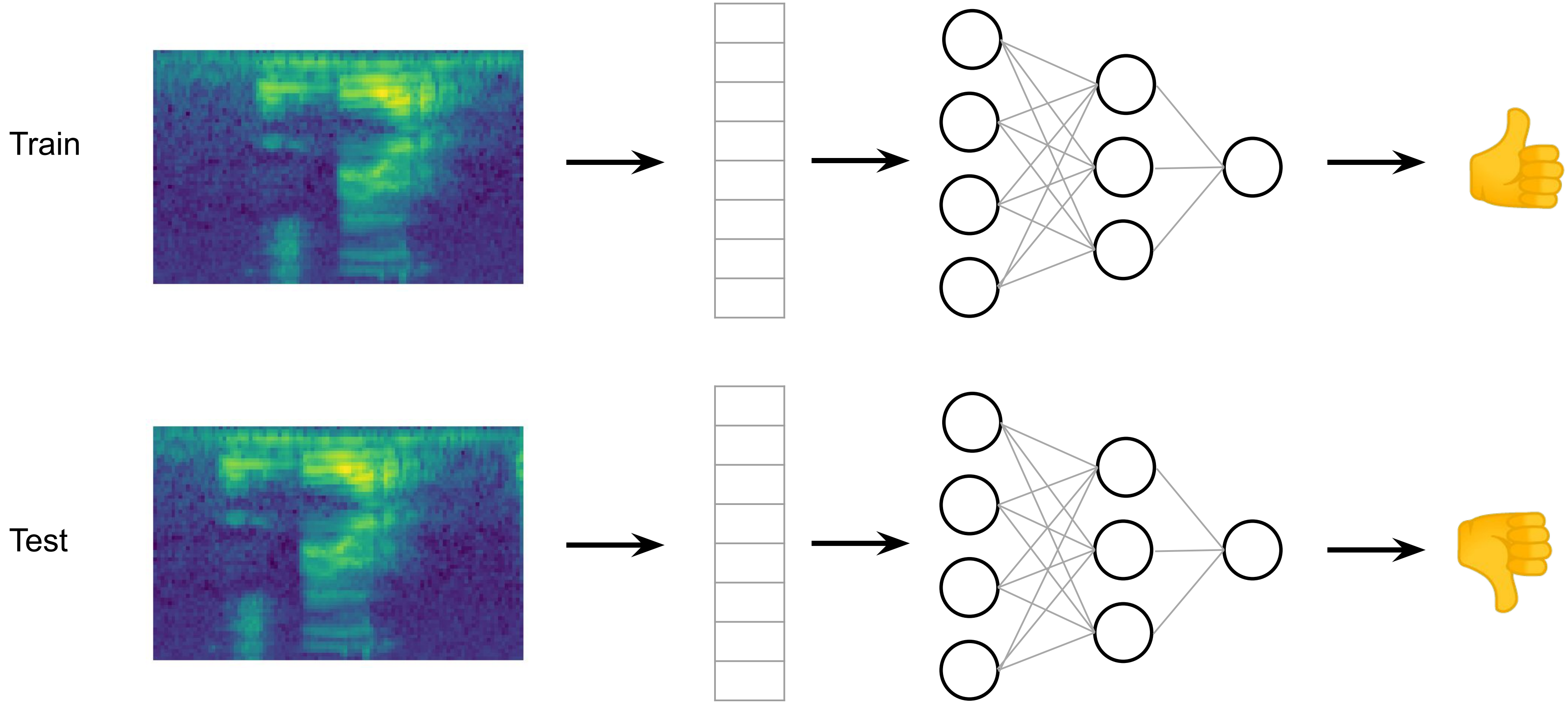
Test



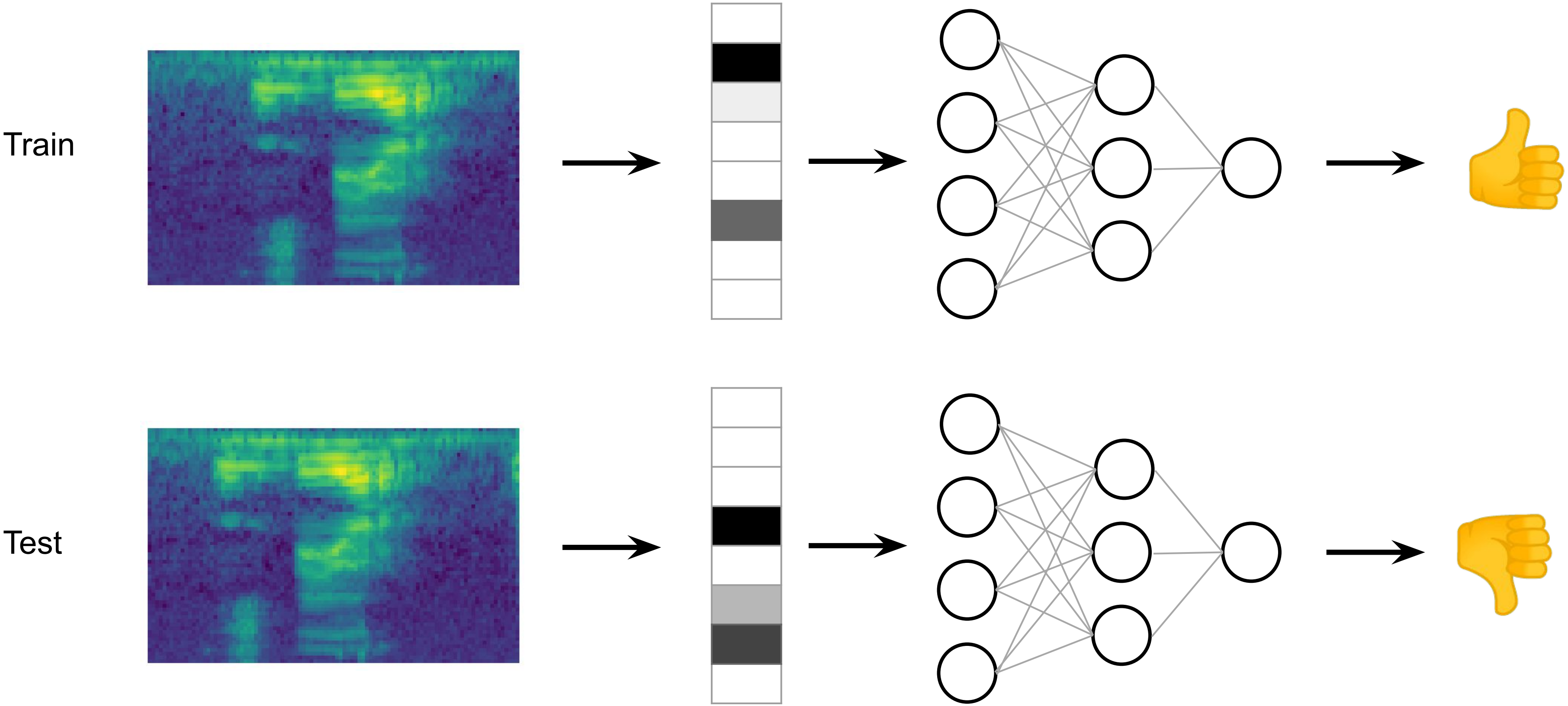
DNN Problems



DNN Problems



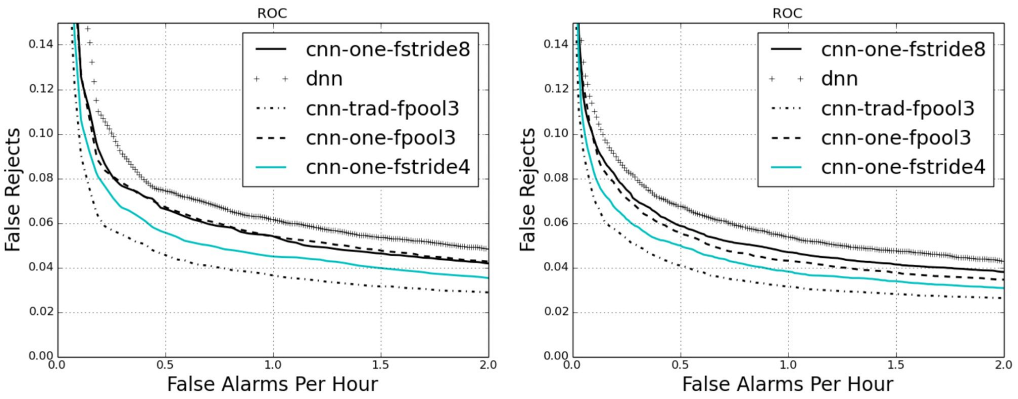
DNN Problems: no translational equivariance



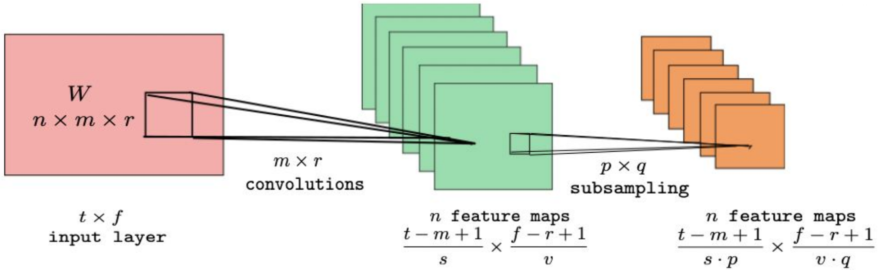
Convolutional Neural Networks for Small-footprint Keyword Spotting

- Google (2015), cited by 522

type	m	r	n	p	q	Params	Mult
conv	32	8	54	1	3	13.8K	456.2K
linear	-	-	32	-	-	19.8K	19.8K
dnn	-	-	128	-	-	4.1K	4.1K
dnn	-	-	128	-	-	16.4K	16.4K
softmax	-	-	4	-	-	0.5K	0.5K
Total	-	-	4	-	-	53.8K	495.6K



(a) Results on Clean (b) Results on Noisy



Convolution

$$(A * w)_{ij} = \sum_{i'=0}^{K-1} \sum_{j'=0}^{K-1} A(i+i', j+j')w(i', j')$$

$A =$

3	1	4
1	5	9
2	6	5

$w =$

1	0
0	-1

$(A * w) =$

Convolution

$$(A * w)_{ij} = \sum_{i'=0}^{K-1} \sum_{j'=0}^{K-1} A(i + i', j + j')w(i', j')$$

$$A = \begin{array}{|c|c|c|} \hline 3 & 1 & 4 \\ \hline 1 & 5 & 9 \\ \hline 2 & 6 & 5 \\ \hline \end{array} \qquad w = \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \qquad (A * w) = \begin{array}{|c|c|} \hline \boxed{-2} & \\ \hline & \\ \hline \end{array}$$

Convolution

$$(A * w)_{ij} = \sum_{i'=0}^{K-1} \sum_{j'=0}^{K-1} A(i+i', j+j')w(i', j')$$

$A =$

3	1	4
1	5	9
2	6	5

$w =$

1	0
0	-1

$(A * w) =$

-2	-8

Convolution

$$(A * w)_{ij} = \sum_{i'=0}^{K-1} \sum_{j'=0}^{K-1} A(i+i', j+j')w(i', j')$$

$$A = \begin{bmatrix} 3 & 1 & 4 \\ 1 & 5 & 9 \\ 2 & 6 & 5 \end{bmatrix} \quad w = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (A * w) = \begin{bmatrix} -2 & -8 \\ \boxed{-1} & \end{bmatrix}$$

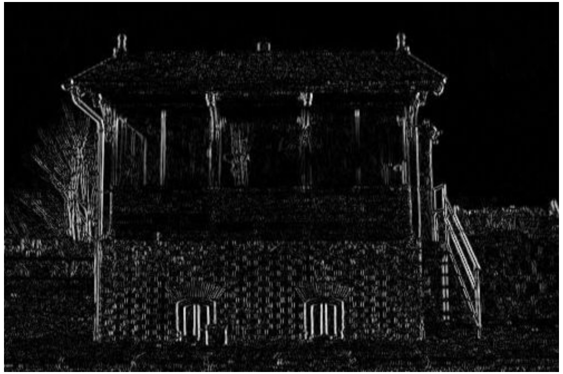
Convolution

$$(A * w)_{ij} = \sum_{i'=0}^{K-1} \sum_{j'=0}^{K-1} A(i+i', j+j') w(i', j')$$

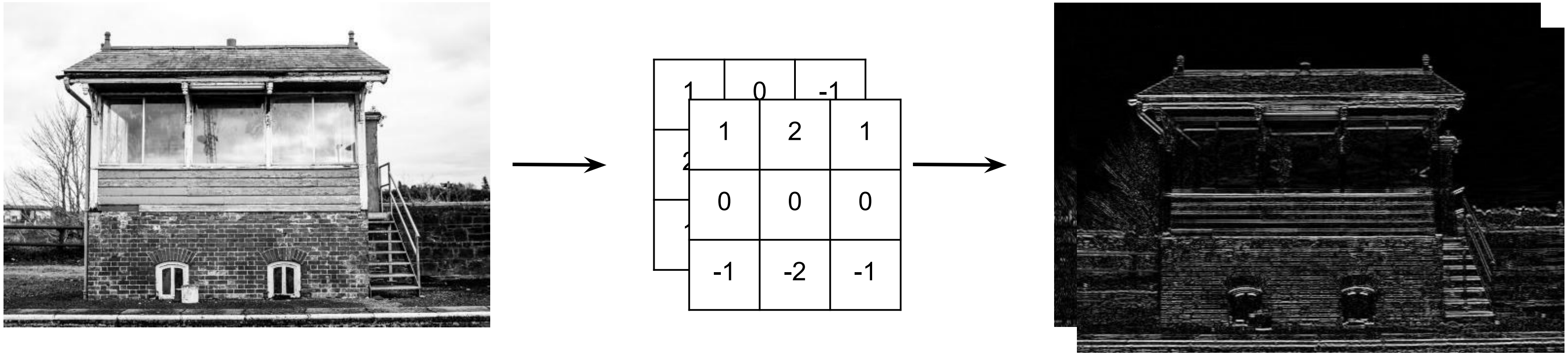
$$A = \begin{bmatrix} 3 & 1 & 4 \\ 1 & 5 & 9 \\ 2 & 6 & 5 \end{bmatrix} \quad w = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (A * w) = \begin{bmatrix} -2 & -8 \\ -1 & \boxed{0} \end{bmatrix}$$

Convolution: Feature Transformation

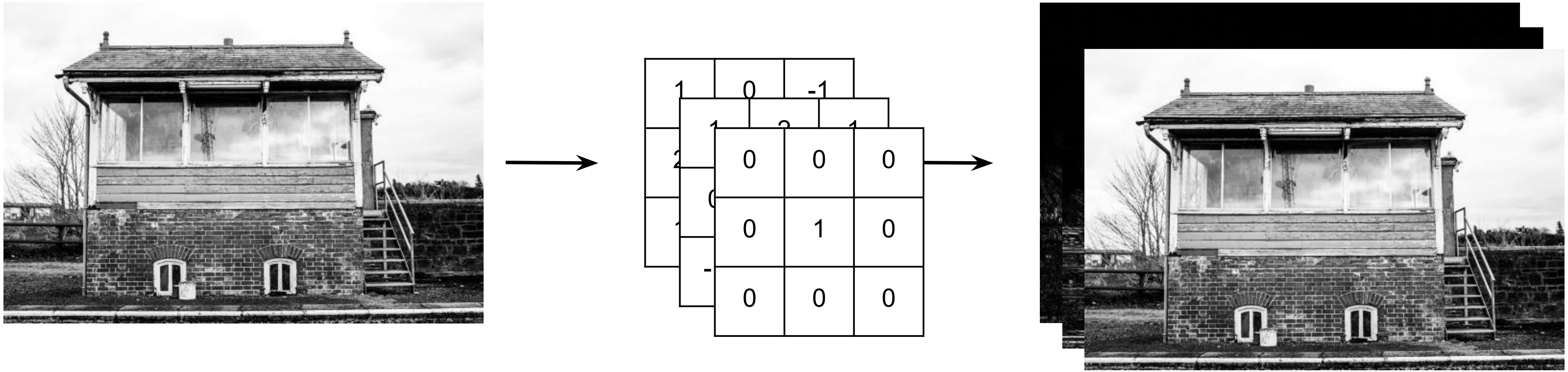
1	0	-1
2	0	-2
1	0	-1



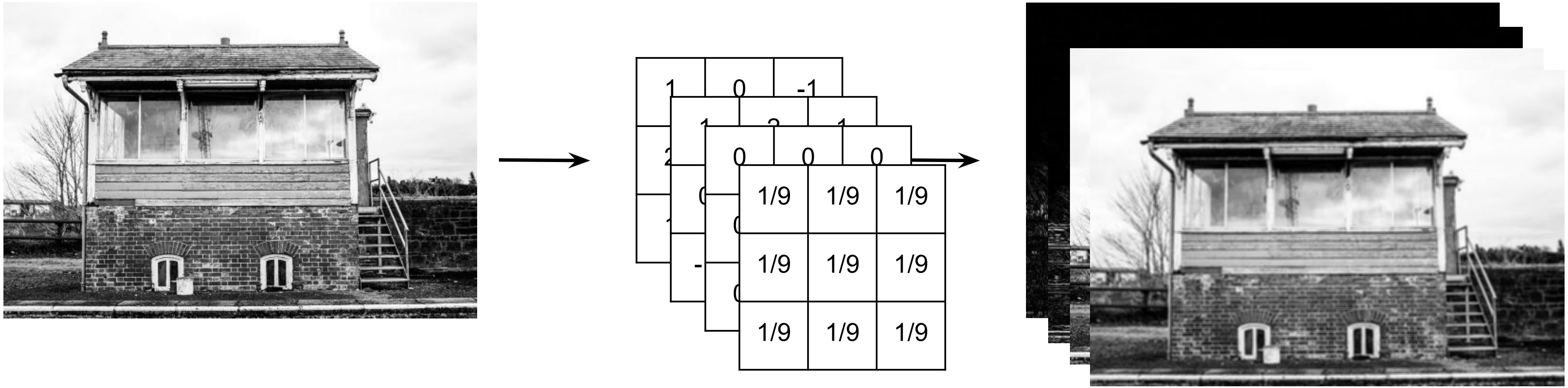
Convolution: Feature Transformation



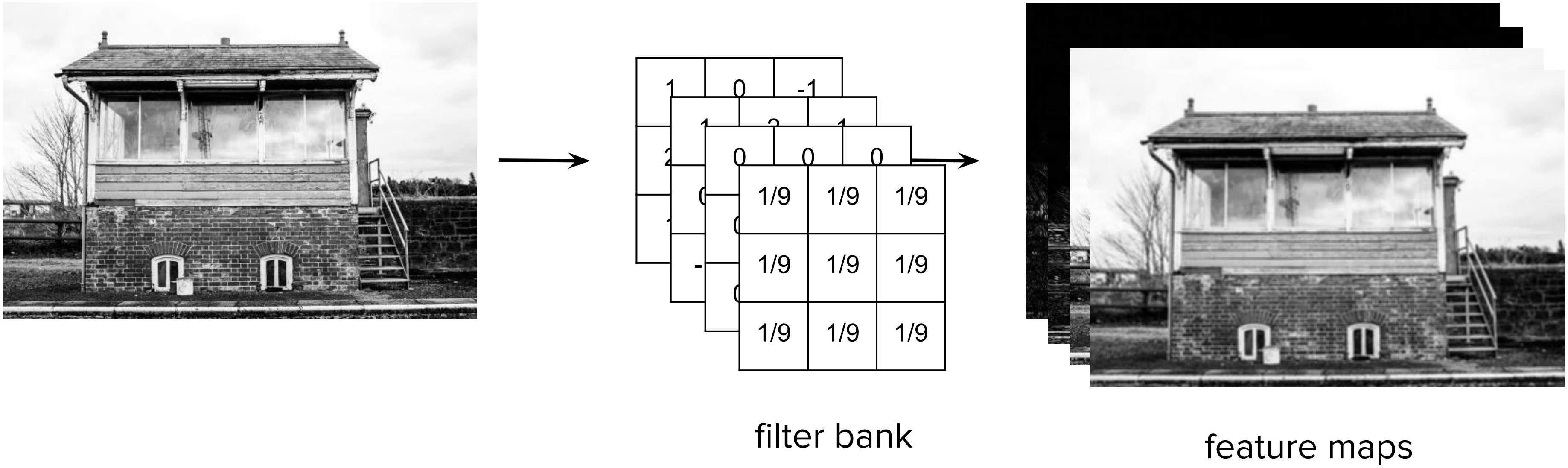
Convolution: Feature Transformation



Convolution: Feature Transformation



Convolution: Feature Transformation



Convolution: Pattern Finding

$$(A * w)_{ij} = \sum_{i'=0}^{K-1} \sum_{j'=0}^{K-1} A(i+i', j+j')\, w(i', j')$$

$A =$

1	2	-1	0	1
-1	1	3	4	4
2	-3	4	3	-4
-3	2	0	-1	0
5	2	-4	0	1

$w =$

1	0
0	-1

$(A * w) =$

0	-1	-5	-4
2	-3	0	8
0	-3	5	3
-5	6	0	-1

Convolution: Pattern Finding

$$(A * w)_{ij} = \sum_{i'=0}^{K-1} \sum_{j'=0}^{K-1} A(i+i', j+j') w(i', j')$$

$$A = \begin{array}{|c|c|c|c|c|} \hline 1 & 2 & -1 & 0 & 1 \\ \hline -1 & 1 & 3 & 4 & 4 \\ \hline 2 & -3 & 4 & 3 & -4 \\ \hline -3 & 2 & 0 & -1 & 0 \\ \hline 5 & 2 & -4 & 0 & 1 \\ \hline \end{array} \qquad w = \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \qquad (A * w) = \begin{array}{|c|c|c|c|} \hline 0 & -1 & -5 & -4 \\ \hline 2 & -3 & 0 & \boxed{8} \\ \hline 0 & -3 & 5 & 3 \\ \hline -5 & \boxed{6} & 0 & -1 \\ \hline \end{array}$$

Convolution: Pattern Finding

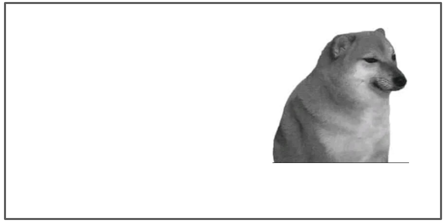
0	1	0
0	1	0
0	1	0

0	0	0
1	1	1
0	0	0

Convolution: summary

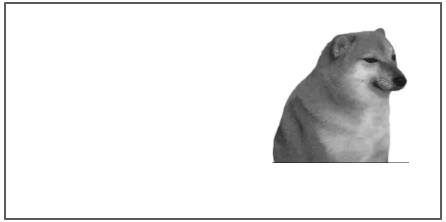
- weights sharing
- pattern finder
- feature transformation
- translational equivariance

Convolution: images vs log mel-spectrograms

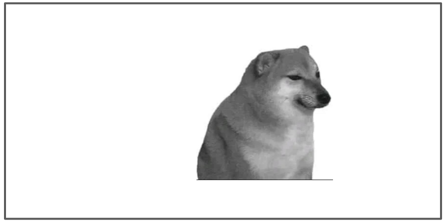


dog

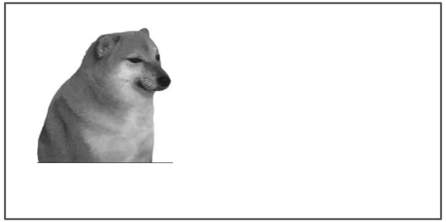
Convolution: images vs log mel-spectrograms



dog

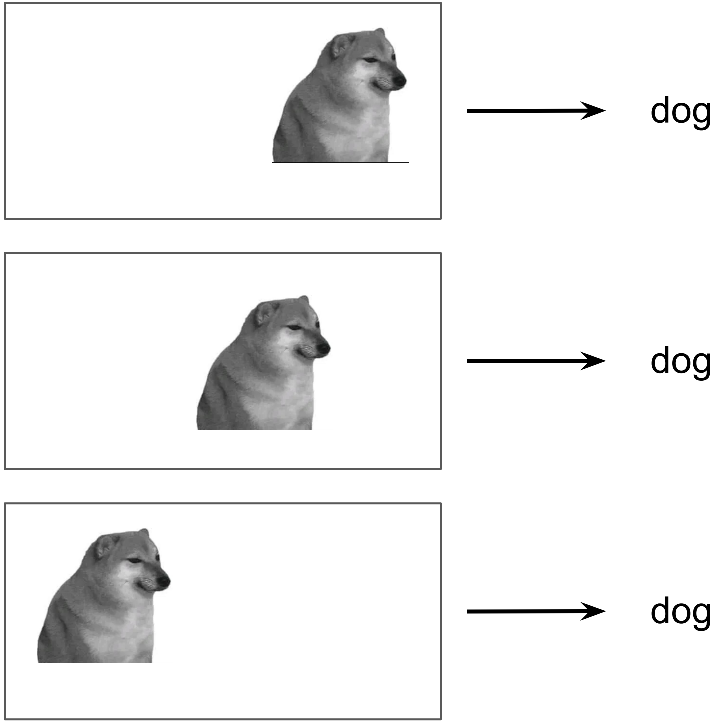


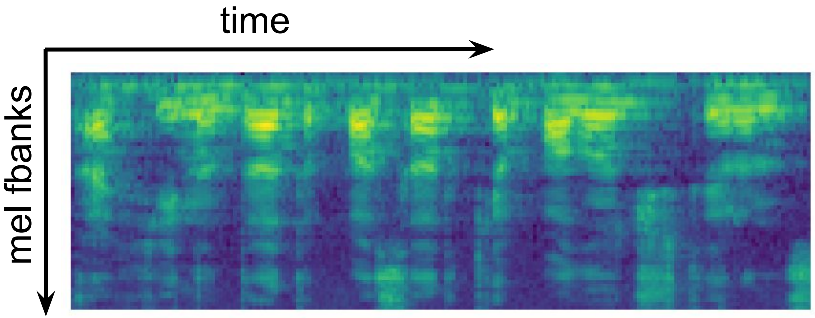
dog



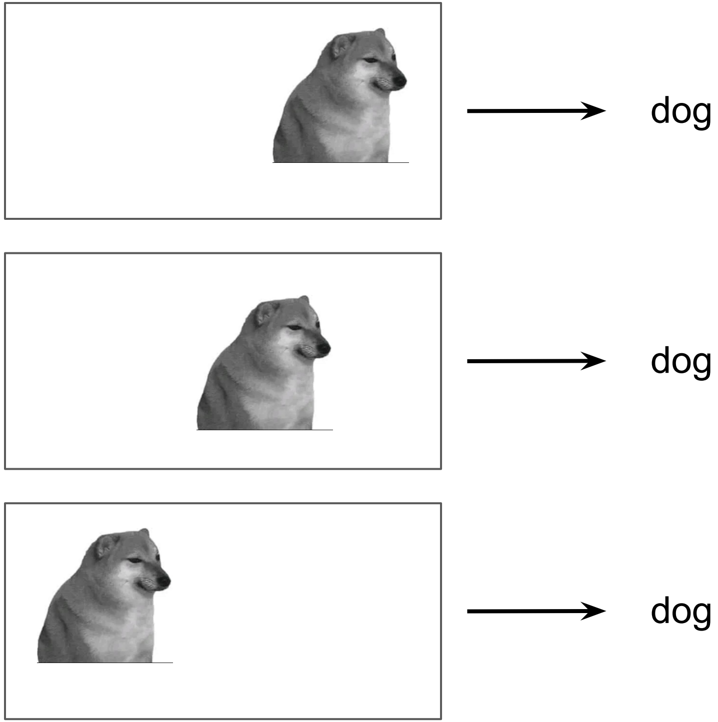
dog

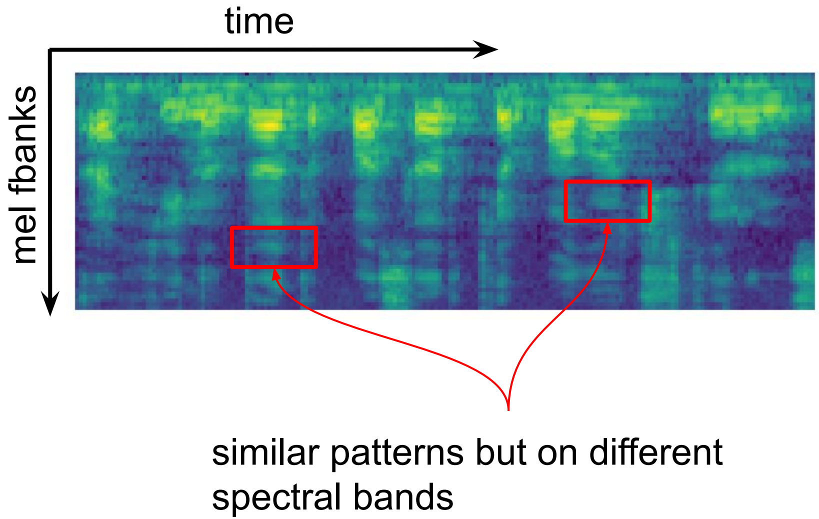
Convolution: images vs log mel-spectrograms



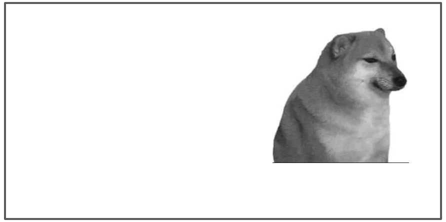


Convolution: images vs log mel-spectrograms

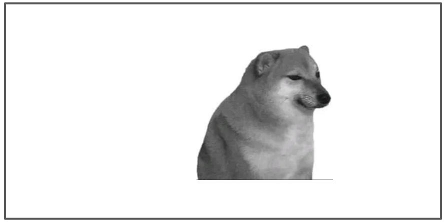




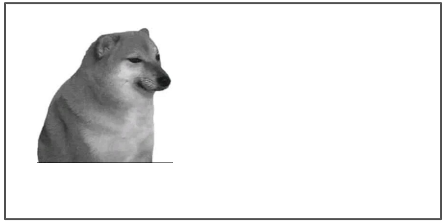
Convolution: images vs log mel-spectrograms



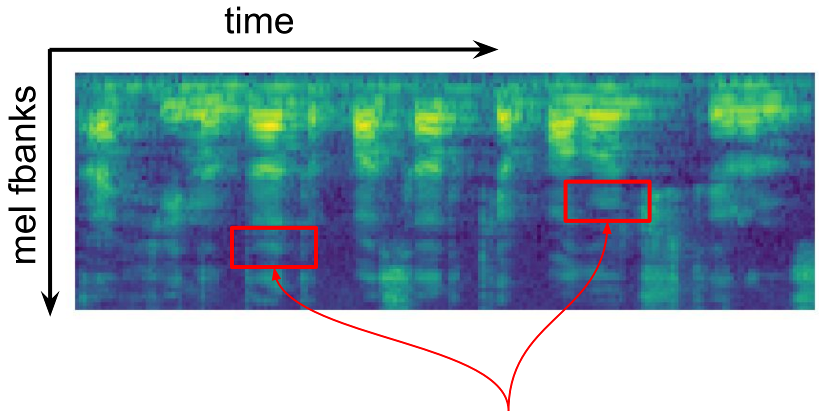
dog



dog



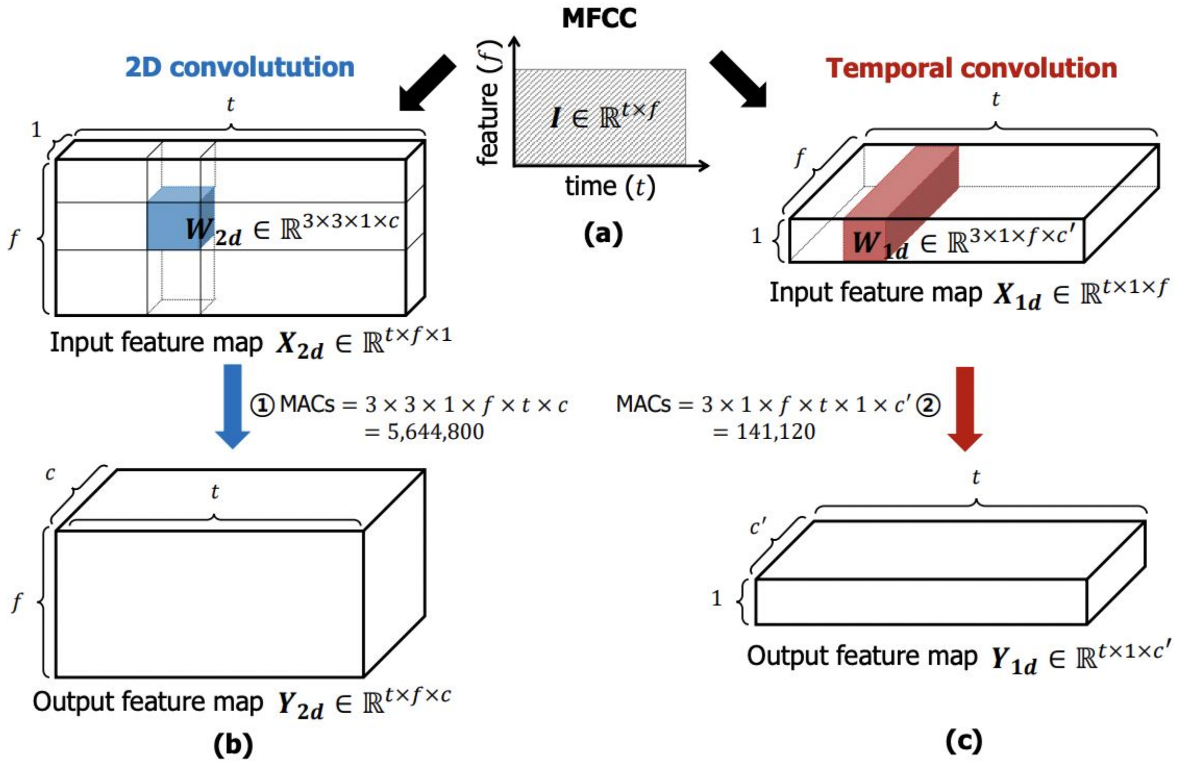
dog



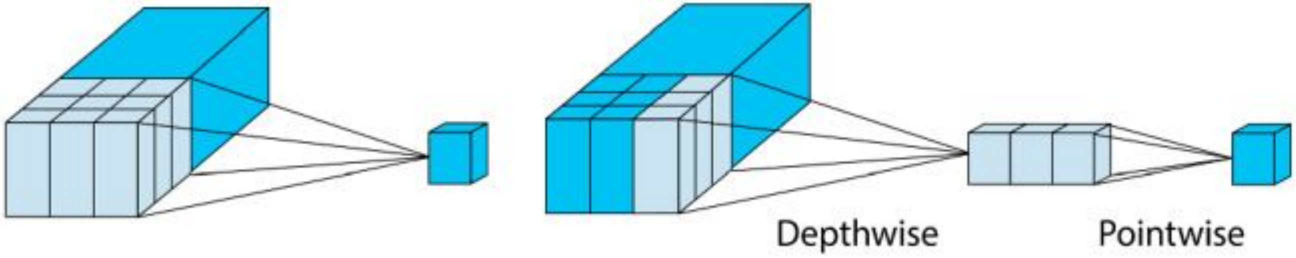
similar patterns but on different spectral bands

do we really need translational equivariance along the frequency axis ?

conv2d vs conv1d

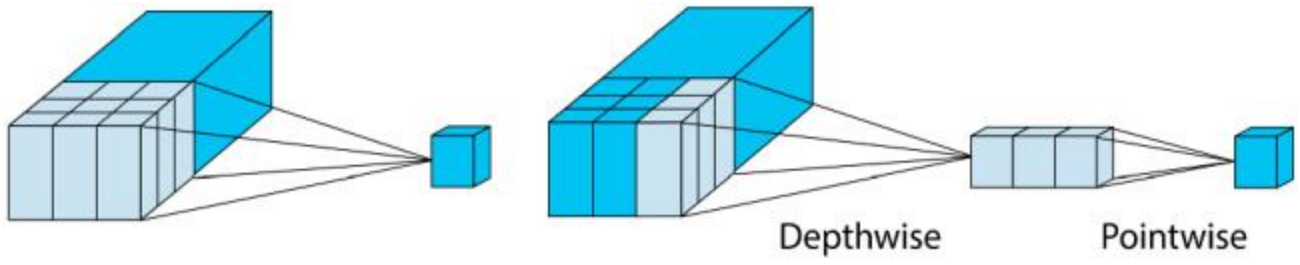


Depthwise Separable Convolutions



(a) Standard 1-D Convolution (b) Depth wise separable 1-D Convolution

Depthwise Separable Convolutions



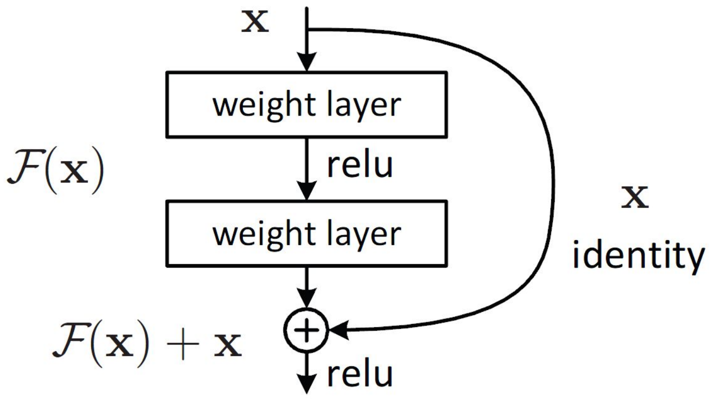
(a) Standard 1-D Convolution (b) Depth wise separable 1-D Convolution

- Standard Convolution:
    - ch_in x kernel_size x ch_out
- Depthwise Separable Convolution
    - ch_in x kernel_size + ch_in x ch_out

Depthwise Separable Convolutions

```
>>> import torch
>>> import thop
>>> conv1d = torch.nn.Sequential(
...     torch.nn.Conv1d(in_channels=64, out_channels=32, kernel_size=3)
... )
>>> depth_wise_conv1d = torch.nn.Sequential(
...     torch.nn.Conv1d(in_channels=64, out_channels=64, kernel_size=3, groups=64),
...     torch.nn.ReLU(),
...     torch.nn.Conv1d(in_channels=64, out_channels=32, kernel_size=1)
... )
>>> inputs = (torch.randn((1, 64, 101)),)
>>> thop.profile(conv1d, inputs=inputs)
[INFO] Register count_convNd() for <class 'torch.nn.modules.conv.Conv1d'>.
[INFO] Register zero_ops() for <class 'torch.nn.modules.container.Sequential'>.
(608256.0, 6176.0)
>>> thop.profile(depth_wise_conv1d, inputs=inputs)
[INFO] Register count_convNd() for <class 'torch.nn.modules.conv.Conv1d'>.
[INFO] Register zero_ops() for <class 'torch.nn.modules.activation.ReLU'>.
[INFO] Register zero_ops() for <class 'torch.nn.modules.container.Sequential'>.
(221760.0, 2336.0)
>>>
```

Residual Block (Skip Connections)



[Deep Residual Learning for Image Recognition](2015)

[DEEP RESIDUAL LEARNING FOR SMALL-FOOTPRINT KEYWORD SPOTTING] (2018)

[MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition]

- NVIDIA, 2020

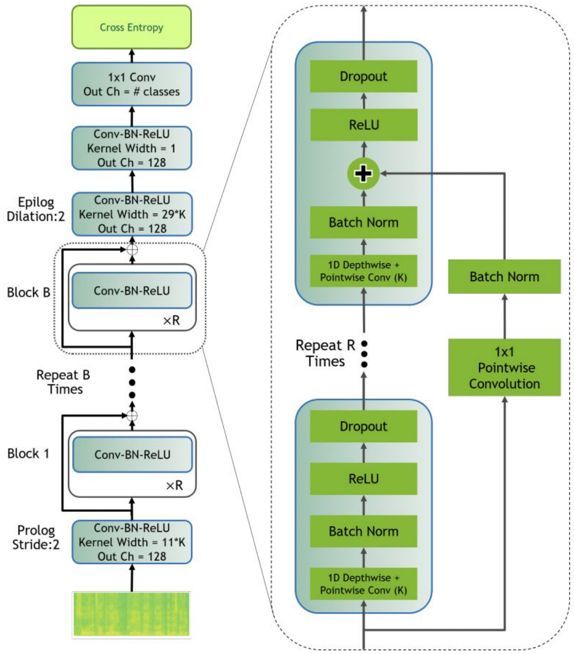
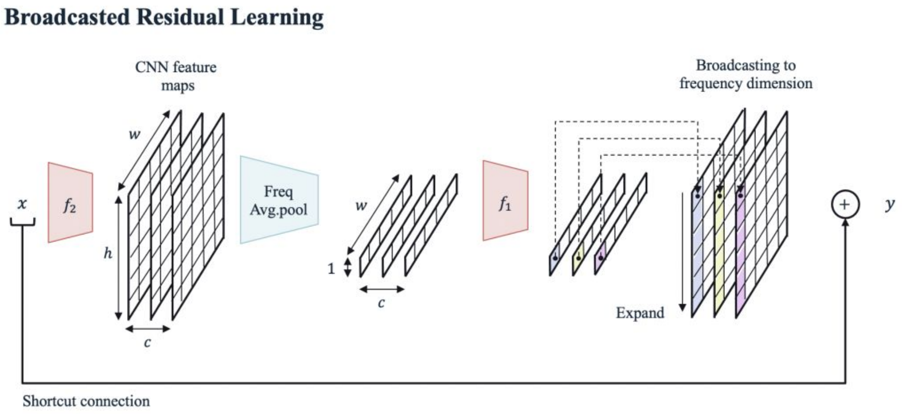
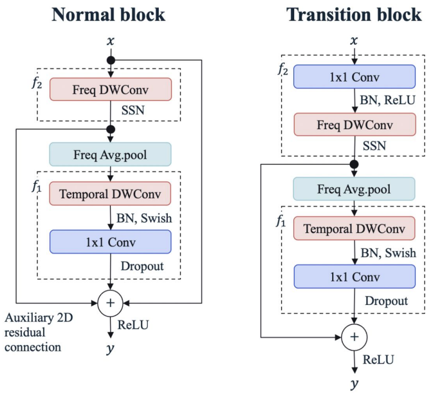


Figure 1: *MatchboxNet BxRxC model: B - number of blocks, R - number of sub-blocks, C - the number of channels.*

Broadcasted Residual Learning for Efficient Keyword Spotting

- Qualcomm AI Research, 2021

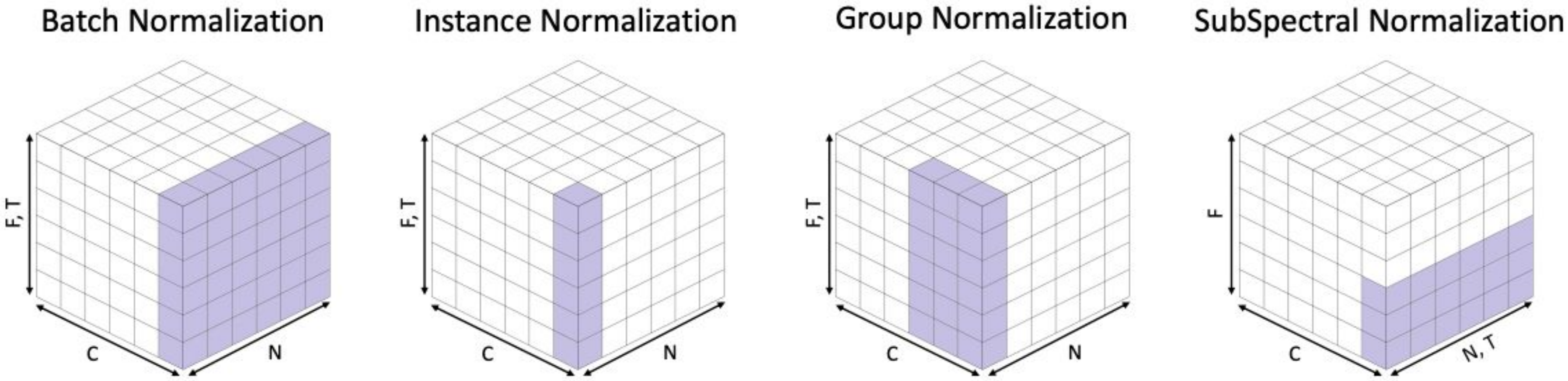




Input	Operator	n	c	s	d
$1 \times 40 \times W$	conv2d 5x5	-	16	(2,1)	1
$16 \times 20 \times W$	BC-ResBlock	2	8	1	1
$8 \times 20 \times W$	BC-ResBlock	2	12	(2,1)	(1,2)
$12 \times 10 \times W$	BC-ResBlock	4	16	(2,1)	(1,4)
$16 \times 5 \times W$	BC-ResBlock	4	20	1	(1,8)
$20 \times 5 \times W$	DWconv 5x5	-	20	1	1
$20 \times 1 \times W$	conv2d 1x1	-	32	1	1
$32 \times 1 \times W$	avgpool	-	-	-	-
$32 \times 1 \times 1$	conv2d 1x1	-	12	-	-

Broadcasted Residual Learning for Efficient Keyword Spotting
SubSpectral Normalization

- Qualcomm AI Research, 2021



Broadcasted Residual Learning for Efficient Keyword Spotting

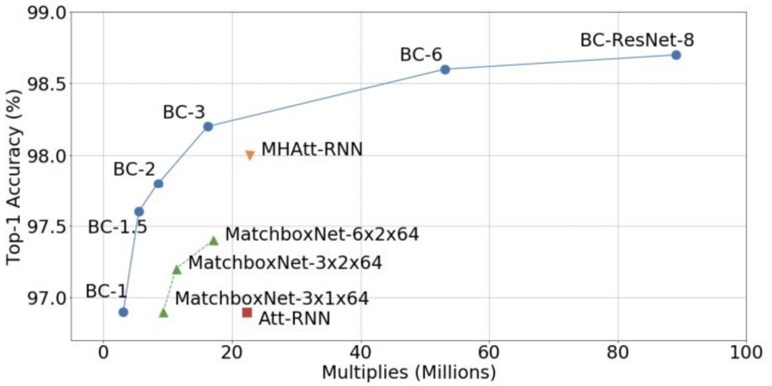


Figure 3: ***MACs vs. Google speech command dataset v2 Accuracy.*** *Details are in Table 3.*

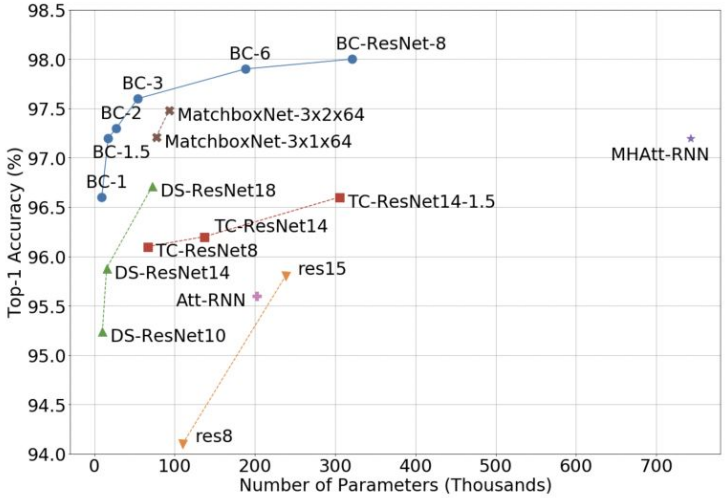
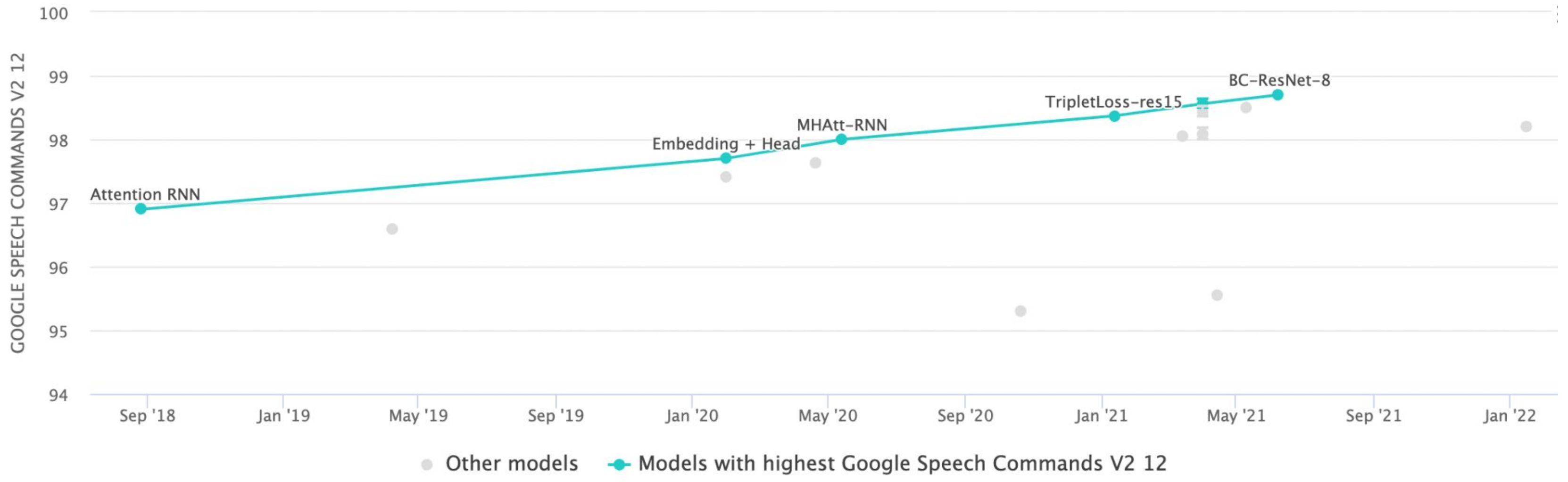


Figure 1: ***Model Size vs. Google speech command dataset v1 Test Accuracy.*** *The proposed BC-ResNets significantly outperform other KWS approaches. The smallest BC-ResNet-1 achieves 96.6% accruacy with less than 10k parameters. We scale the BC-ResNet-1 by channel width with a factor of 8, and BC-ResNet-8 achieves the state-of-the-art 98.0%. The details are in Table 3.*

KWS Benchmark

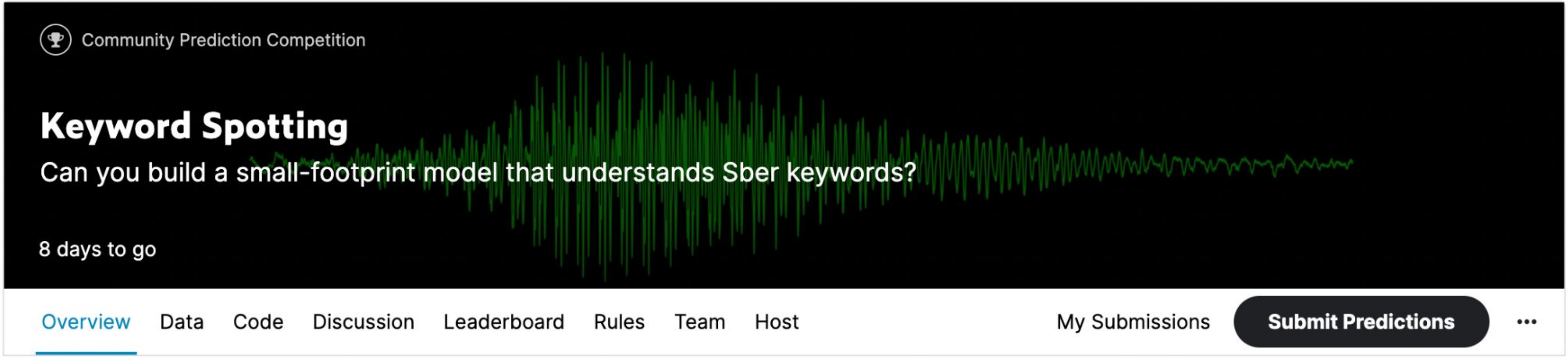
- [Google Speech Commands]
    - v1: 65,000 one-second long utterances of 30 short words, by thousands of different people
    - v2: 105,829 utterances of 35 words



Homework

Keyword Spotting

- Kaggle In-Class Competition
- 100k train, 2k test
- model: <= **1e4** params, <= **1e6** MACs
- report + model-checkpoint + leaderboard submits
- deadline: 2022-10-11 17:59



Thank you for your attention!

@georgygospodinov