Deep Learning on FPGA for Keyword Spotting

Duong Van Hai

Supervisor: **Assoc.Prof. Nguyen Quoc Cuong**

Hanoi University of Science and Technology

Agenda

Agenda

Introduction

Keyword Spotting definition

Keyword Spotting - process of identifying pre-defined keywords, in speech recorded in real-time.

Keyword Spotting in real world

Wakeup Word - common way to begin an interaction by the voice interface.

Example

- Amazon - "Alexa"
- Google - "OK Google"
- Apple - "Hey Siri"

Figure 1: Keyword spotting on production.

Introduction

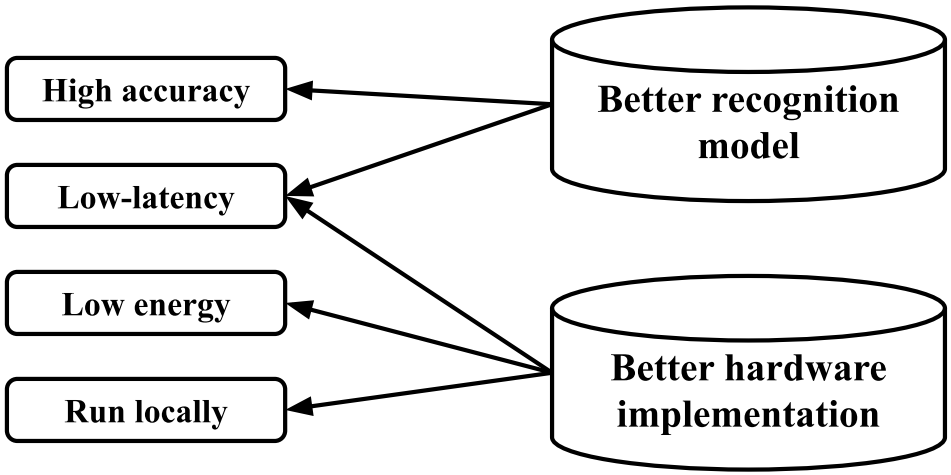


Figure 2: KWS requirements.

Better recognition model

- Apply lastest model on pattern recognition.
- Design model with less parameters.

Better hardware implementation

- Optimize the inference implementation.
- Deploy on other platform (FPGA).

Agenda

Keyword Spotting Overview

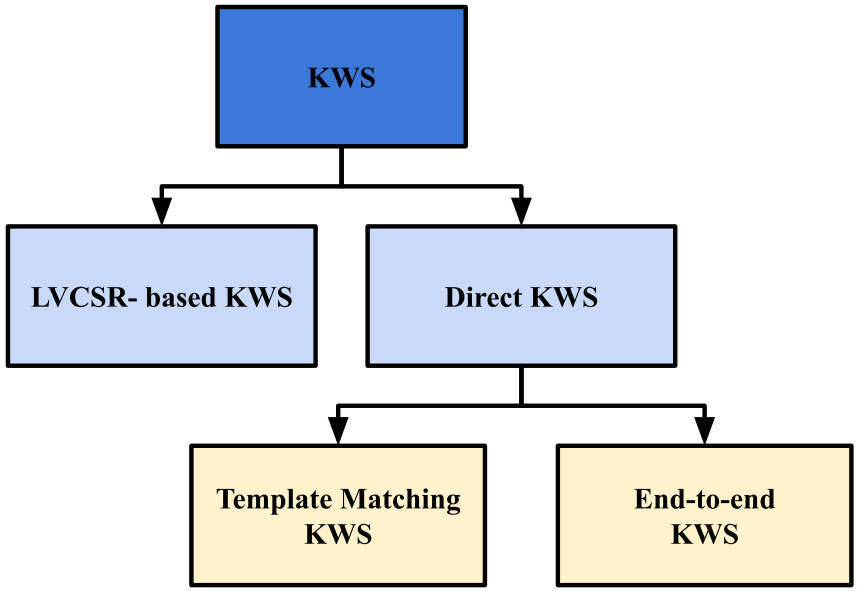


Figure 3: KWS approaches

Keyword spotting approaches

- **LVCSR-based KWS** Large-Vocabulary Continuous Speech Recognition (LVCSR) converts speech utterance to text, then a text search algorithm determines the keyword occurrence.
- **Direct KWS**:
  - **Template Matching**: search over input utterances that sound like the template.
  - **End-to-end**: directly detects the keyword.

Keyword Spotting Evaluation Metrics

Real Time Factor (RTF)

If it takes time P to process an input of duration I, RTF is computed using the calculation:

$$RTF = \frac{P}{I} \quad (1)$$

False rejection rate

False rejection rate refers to the detection of a keyword as a non-keyword.

$$\text{FRR} = \frac{\text{Total false rejection}}{\text{Total keyword occurrences}} \quad (2)$$

False Alarm Rate

False alarm rate refers to detect a non-keyword as a keyword.

$$FAR = \frac{\text{Total false detection}}{\text{Total non-kw occurrences}} \quad (3)$$

False alarm per hour per keyword

$$FA/k/hour = \frac{FAR}{\text{Total Kw} \times \text{Duration}} \quad (4)$$

Agenda

DNN-based KWS

Deep KWS

- In 2014, a DNN-based KWS system was proposed by Chen et al.[1] to solve the problem of real-time KWS on mobile devices.
- System includes 3 modules: the **Feature Extraction**, the **Deep Neural Network**, and the **Posterior Handling** module (Figure 4).

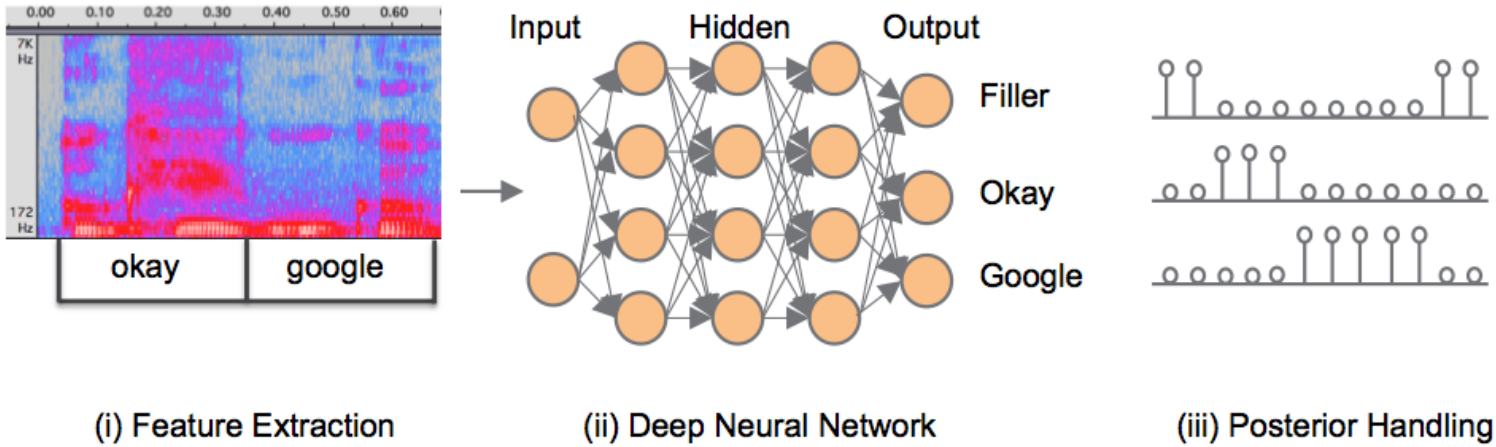


Figure 4: Framework of Deep KWS system [1].

[1] Guoguo Chen, Carolina Parada, and Georg Heigold. "Small-footprint keyword spotting using deep neural networks". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 4087–4091

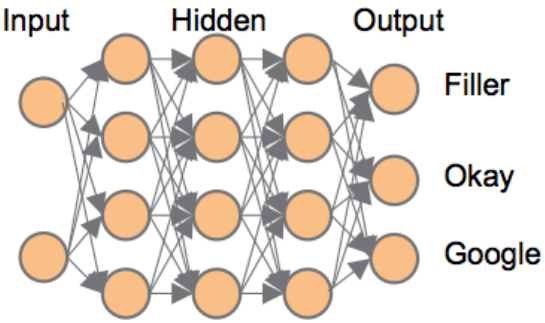
DNN-based KWS / Model

Log-Filterbank Energies (LFE)

- **Input**: $x \in \mathbb{Z}^N$, where N is number of samples.
- **Processing**: 40-dimensional LFE computed every 10ms over a window of 25ms.
- **Output**: $X_{LFE} \in \mathbb{R}^{T \times F}$, where T is number of frames, F is number of frequency bins.

Deep Fully-connected Neural Network

- **Input**: $X_{LFE} \in \mathbb{R}^{T \times F}$
- **Ouput**: $p_{ij} = P(y_j = Y_i | X_{LFE}; \theta)$, where
  - y_j is a prediction of j^{th} frame.
  - Y_i is a i^{th} label, $i \in \{0, 1, ..., n-1\}$.
  - n the number of total labels.
  - θ is the set of parameters of the DNN.
- **Training**: Optimize θ to maximize the likelihood of predicted label and ground truth.



(ii) Deep Neural Network

Figure 5: Neural Network module.

Attention-based KWS

Attention-based KWS

- The architecture consists of 2 major sub-modules: the **encoder** and the **attention mechanism** (Figure 6).
- End-to-end system:
  - a simple model directly outputs keyword detection.
  - no complicated searching involved.
  - no alignments needed beforehand to train the model.

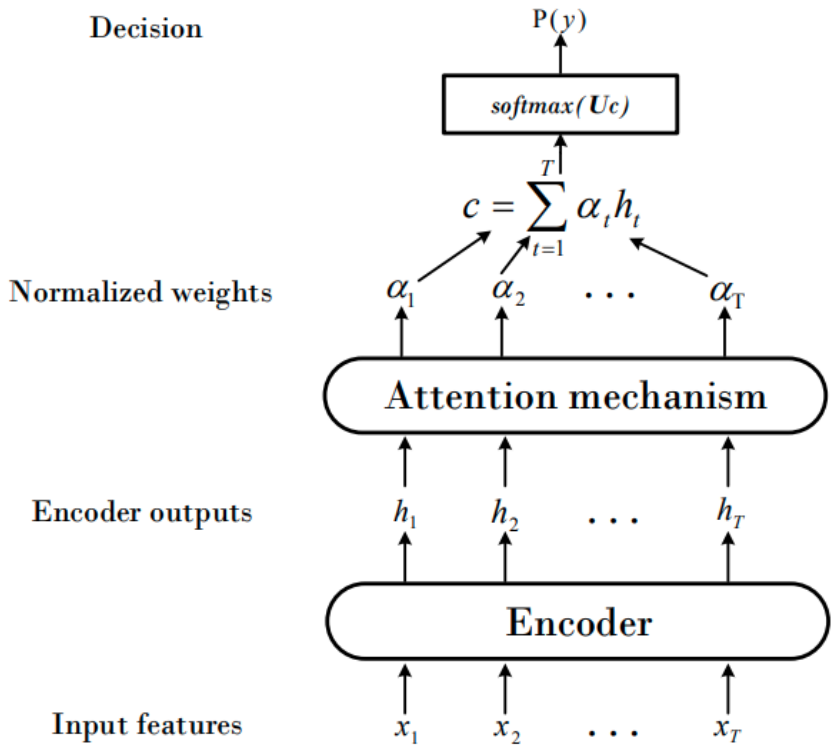


Figure 6: Attention-based end-to-end model[2].

[2] Changhao Shan, Junbo Zhang, Yujun Wang, and Lei Xie. "Attention-based End-to-End Models for Small-Footprint Keyword Spotting". In: *Proc. Interspeech 2018* (2018), pp. 2037–2041

Attention-based KWS / Model

End-to-end Architecture

- **Input**: Per-Channel Energy Normalization[3] (PCEN) $X_{PCEN} \in \mathbb{R}^{T \times F}$
- **Encoder**: h $= Encoder(X_{PCEN})$, where
  - h $= \{h_1, ..., h_T\}$, $h_t \in \mathbb{R}^{F_1}$, with F_1 is number of hidden nodes of Encoder.
  - *Encoder* is usually a RNN to make use of speech contextual information
- **Attention**:

$$\alpha_t = softmax\left[W_2 tanh(W_1 h_t + b_1)\right]$$
$$c = \sum_{t=1}^{T} \alpha_t h_t \tag{5}$$

  where $\alpha = \{\alpha_1, ..., \alpha_T\}$, $\alpha_t \in \mathbb{R}^{F_1}$, is normalized weights of the attention mechanism; $c \in \mathbb{R}^{F_1}$ is weighted average of the Encoder outputs h.
- **Output**: $p(y) = softmax(W_3 c)$, y indicates whether a keyword detected.
- Note that, the system is triggered when the $p(y = 1)$ exceeds a preset threshold.

[3] Yuxuan Wang, Pascal Getreuer, Thad Hughes, Richard F Lyon, and Rif A Saurous. "Trainable frontend for robust and far-field keyword spotting". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 5670–5674

Wavenet-based KWS

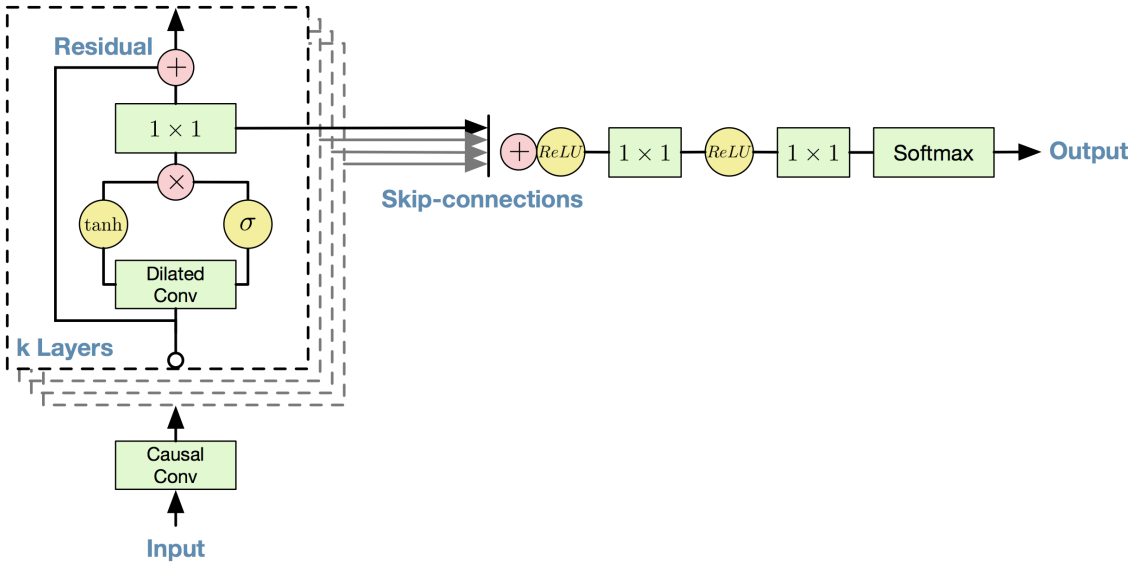


Figure 7: Wavenet-based KWS architecture[4].

Wavenet Architecture

- Wavenet[5] was initially proposed as a generative model for speech synthesis.
- The system includes stacked **dilated convolution** layers with **gated activation** units, wrapped in a **residual block** (Figure 7).

[4] Alice Coucke, Mohammed Chlieh, Thibault Gisselbrecht, David Leroy, Mathieu Poumeyrol, and Thibaut Lavril. "Efficient keyword spotting using dilated convolutions and gating". In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 6351–6355

[5] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016)

Wavenet-based KWS / Model

Dilated Convolution

- Non-autoregressive model
- Dilated convolution captures long temporal patterns

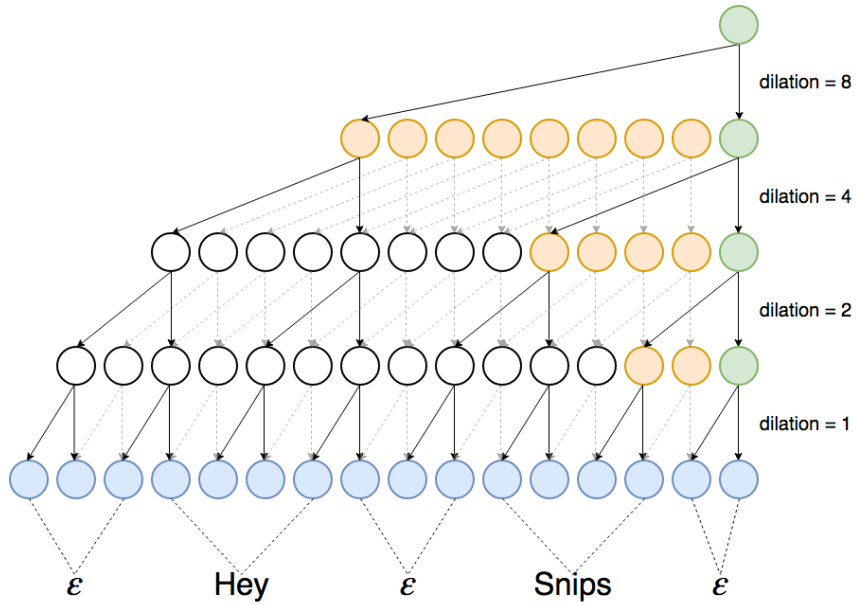


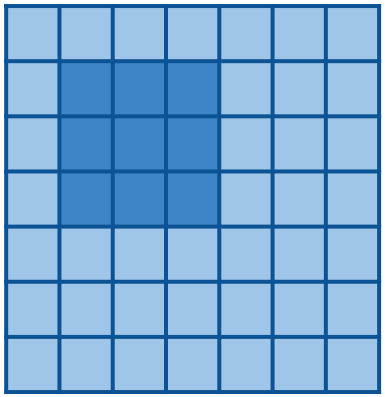
Figure 8: Dilated convolution layers[4] with an exponential dilation rate of 1, 2, 4, 8 and filter size of 2.

[4] Alice Coucke, Mohammed Chlieh, Thibault Gisselbrecht, David Leroy, Mathieu Poumeyrol, and Thibaut Lavril. "Efficient keyword spotting using dilated convolutions and gating". In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 6351–6355

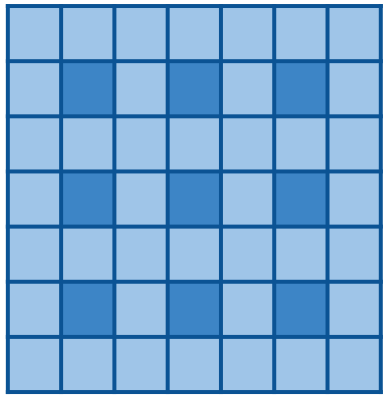
Agenda

Proposed System

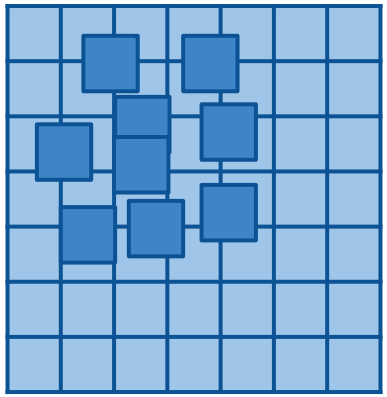
CNN Types



(a) Conventional Convolution



(b) Dilated Convolution



(c) Deformable Convolution

Figure 9: Different types of CNN.

Proposed System / System Description

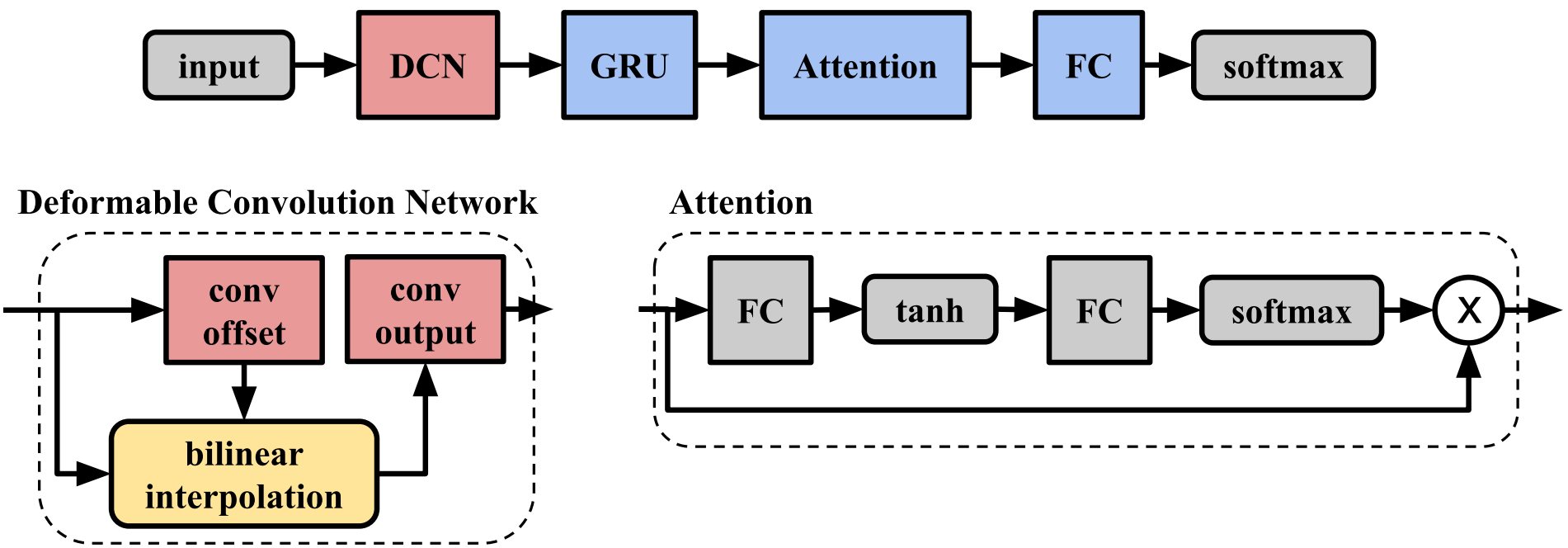


Figure 10: End-to-end architecture of the proposed KWS system.

Proposed System / Deformable Convolution Network

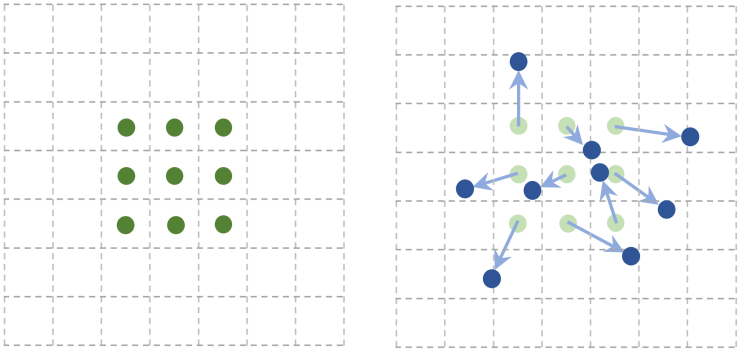


Figure 11: Deformable convolution[6].

Deformable convolution consists:

- Generate offsets
- Interpolate
- Deform features

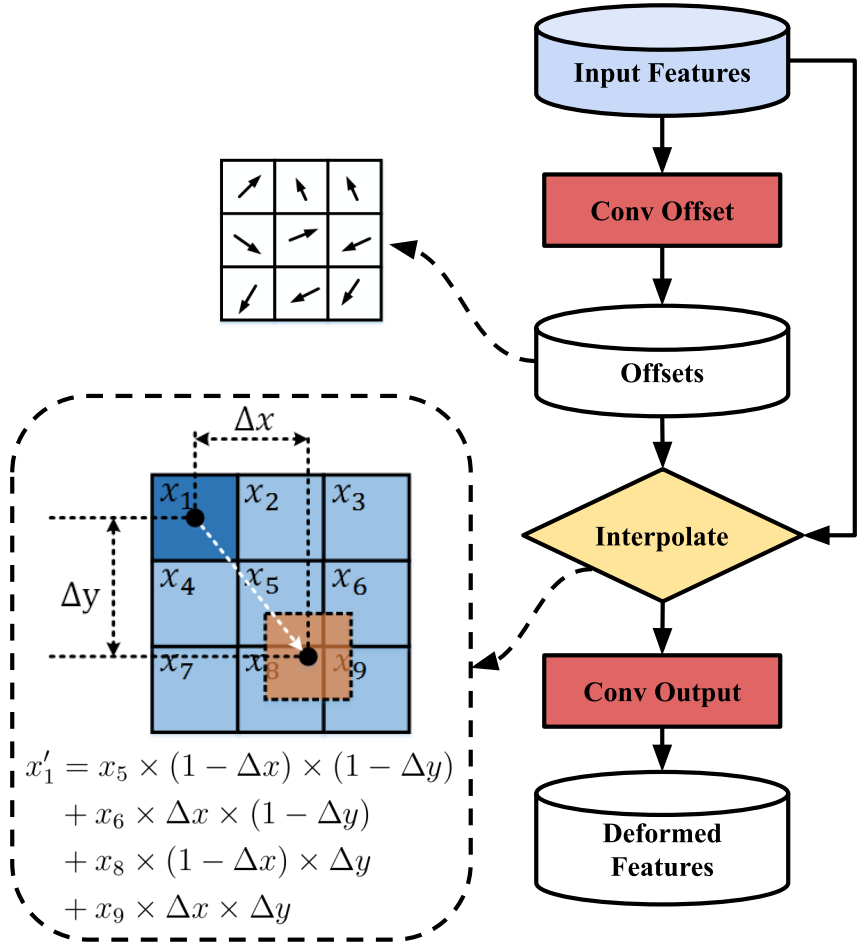


Figure 12: Deformable convolution architecture.

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. "Deformable convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 764–773

Experimental Result / Dataset

"Hey Snips" dataset[4]

- Real-world wake-up data collected from crowdsourced close-talk dataset.
  - 11K wake-up word *"Hey Snips"*.
  - 86K negative utterances.
- Noisy augmenting with **noise** and **music** background from Musan[7].

		Train	Dev	Test
Hey Snips	utterances	5876	2504	2588
	speakers	1179	516	520
	max/speakers	10	10	10
Negative	speakers	3330	1474	1469
	utterances	45344	20321	20821
	max/speakers	30	30	30

Table 1: Dataset statistics.

[4] Alice Coucke, Mohammed Chlieh, Thibault Gisselbrecht, David Leroy, Mathieu Poumeyrol, and Thibaut Lavril. "Efficient keyword spotting using dilated convolutions and gating". In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 6351–6355

[7] David Snyder, Guoguo Chen, and Daniel Povey. "Musan: A music, speech, and noise corpus". In: *arXiv preprint arXiv:1510.08484* (2015)

Experimental Result / Setup

Feature Extraction

40-dimensional Mel-Frequency Cepstral Coefficients (MFCCs) extracted every 10ms over a window of 25ms.

Model Configurations

Deformable Convolution Network

- The number of filters: $\{8, 16\}$,
- Filter size: $\{(3, 3), (5, 5), (10, 3), (10, 5), (20, 3), (20, 5)\}$
- Stride: $\{(1, 1), (1, 2)\}$

Gated Recurrent Unit

- Number of layers: $\{1, 2\}$
- Hidden nodes: $\{16, 32, 64\}$

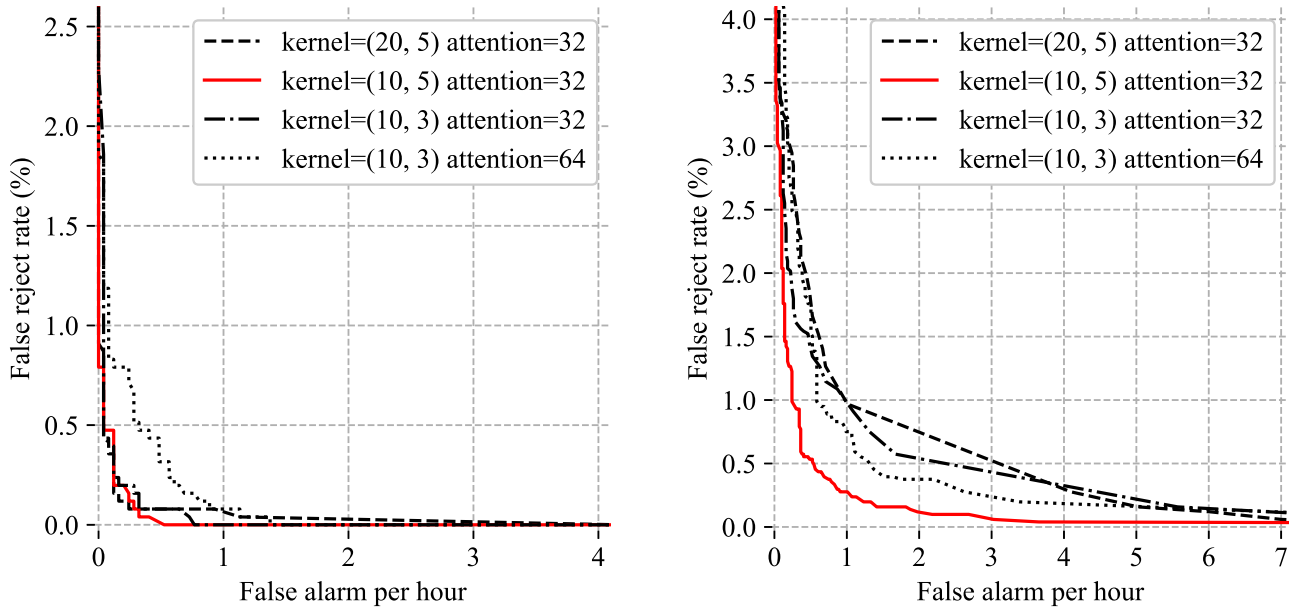
Attention Module

- Attention size: $\{16, 32, 64\}$

Experimental Result / Proposed Model

Detection Error Tradeoff (DET)

- Clean environment: the improvement is marginal.
- Noise environment: best parameter set (**1 DCN** layer with **16 filters** of size (**10,5**), **1 GRU** layer with **64 hidden nodes**, **attention size 32**) outperforms the other set.



(a) clean **(b)** noisy (SNR=5dB)

Figure 13: DET curve for different parameters of the proposed model.

Experimental Result / Proposed Model vs Attention-based

Attention-based Model

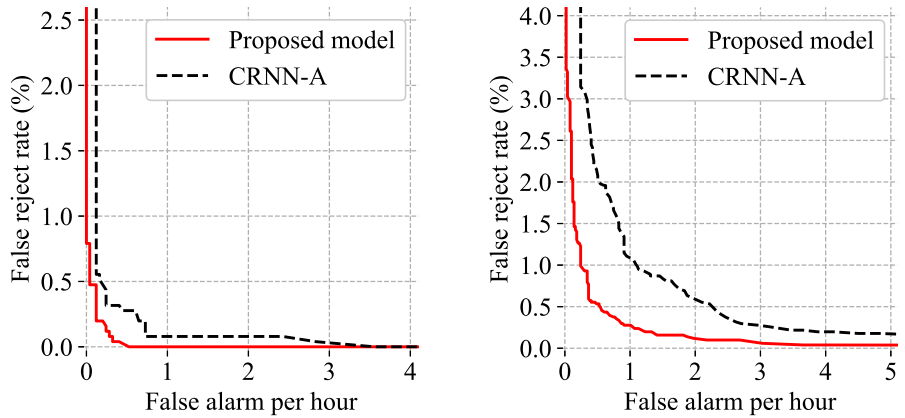
- Interspeech 2018.
- Reimplement and train on dataset *"Hey Snips"* with all configurations.
- Best performance: **1 CNN** layer, **16 filters** of size **(20, 5)**; **1 GRU** layer **64 hidden nodes**, and **attention size 64**.

Comparison

- Proposed model yields a lower FRR with a 98% on clean and 74% in noisy environments.
- With similar architecture, DCN improves performance significantly compared to regular CNN.

Model	Parameters	FRR clean	FRR noisy
CRNN-A	78K	0.27	2.05
Proposed model	**78K**	**0.005**	**0.531**

Table 2: Performances of proposed model and Attention-based (CRNN-A) model.



(a) clean **(b)** noisy (SNR=5dB)

Figure 14: DET curve for the proposed model compared to the CRNN-A baselines in clean (a) and noisy (b) conditions.

Experimental Result / Proposed Model vs Wavenet-based

Wavnet-based Model

- The results of Wavenet-based model reported by Coucke et al.[4] is used to compare.

Comparison

- Proposed model parameters is significantly less than the baseline.
- Proposed model yields a lower FRR with a 95% on clean and 66% in noisy environments.

Model	Parameters	FRR clean	FRR noisy
Wavenet[4]	222K	0.12	1.60
Proposed model	**78K**	**0.005**	**0.531**

Table 3: Performances of proposed model and Wavenet-based model.

[4] Alice Coucke, Mohammed Chlieh, Thibault Gisselbrecht, David Leroy, Mathieu Poumeyrol, and Thibaut Lavril. "Efficient keyword spotting using dilated convolutions and gating". In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 6351–6355

Agenda

FPGA Implementation / Overview

System Overall

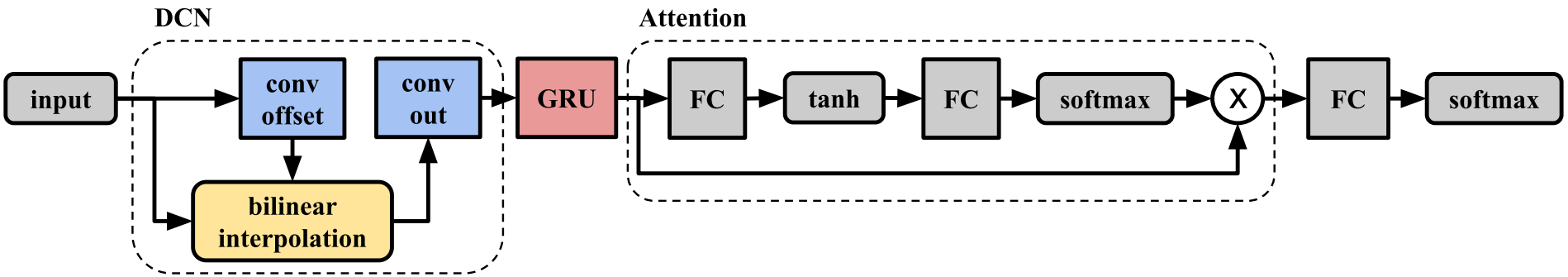


Figure 15: Block diagram of the proposed KWS system.

Processing Time

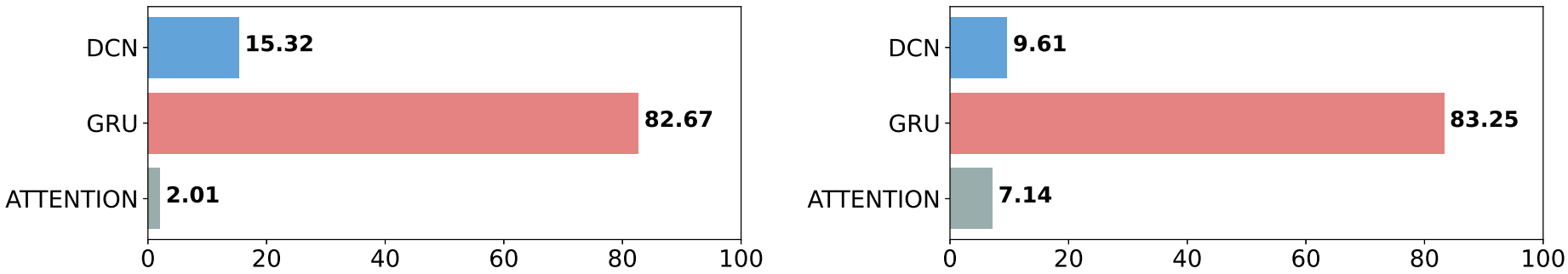


Figure 16: Time consuming (%) on CPU and GPU of each components in proposed system.

FPGA Implementation / Deformable Convolution Network

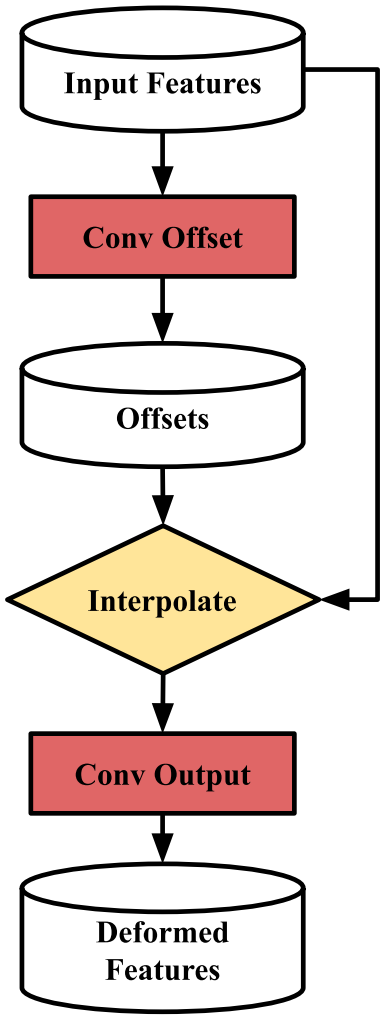


Figure 17: DCN

Deformable convolution consists:

- Regular convolution modules
  - **Conv Offset**
  - **Conv Output**
- Interpolate module

FPGA Implementation / Deformable Convolution Network

Regular convolution:

- A straightforward implementation would work on a CPU.
- Have to store input values for multiple accesses.
- The amount I/O accesses per clock cycle creates a bottleneck.

Pseudocode 1 Regular convolution layer

```
1:  for h in 0 → H − 1 do
2:      for w in 0 → W − 1 do
3:          for o in 0 → O − 1 do
4:              for i in 0 → I − 1 do
5:                  for k1 in 0 → K − 1 do
6:                      for k2 in 0 → K − 1 do
7:                          out[o][h][w] += w[o][i][k1][k2] ∗ in[i][h + k1][w + k2];
8:                      end for
9:                  end for
10:             end for
11:         end for
12:     end for
13: end for
```

Figure 18: Pseudocode of a regular convolution

FPGA Implementation / Deformable Convolution Network

Line buffer architecture

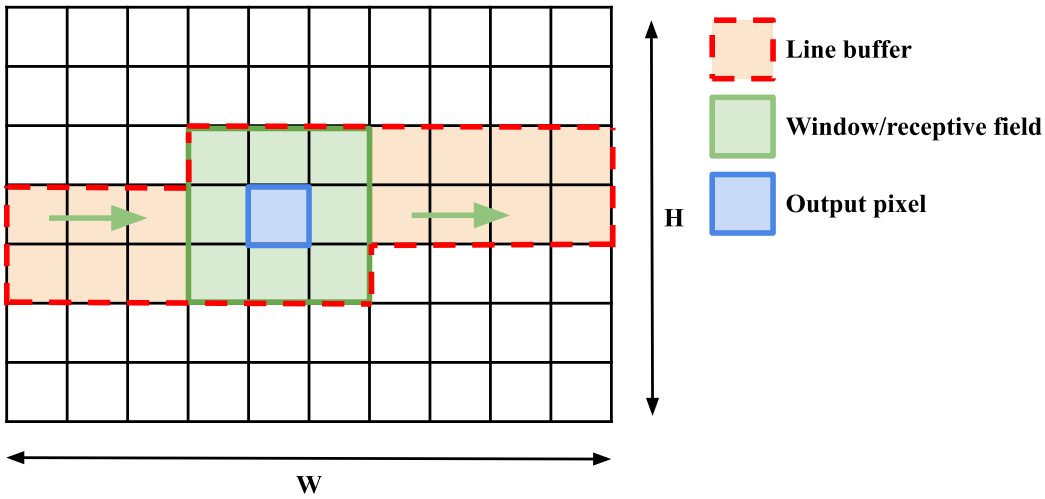


Figure 19: Operation of the line buffer and window for 2D filtering.

Line buffer operation

- The line buffer caches just enough elements to fill the rightmost column of the window
- Every clock cycle
  - One new input is read, and write into the buffer.
  - Oldest entry of the buffer is discarded.
  - One output is written to the external data stream.

FPGA Implementation / Gated Recurrent Unit

GRU operations

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr})$$

$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz})$$

$$\tilde{h}_t = \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{(t-1)} + b_{hn}))$$

$$h_t = (1 - z_t) \odot \tilde{h}_t + z_t \odot h_{(t-1)}$$

Analysis

- Naive implementation.
- **Problem 1**: Access x_t and h_{t-1} mulitple times.
- **Problem 2**: Execute multiple operations concurrently.

Problem 1: Single access

Concatenate weight matrices

$$W_i = \begin{bmatrix} W_{ir} \\ W_{iz} \\ W_{in} \end{bmatrix}, W_h = \begin{bmatrix} W_{hr} \\ W_{hz} \\ W_{hn} \end{bmatrix}$$

Problem 2: Operation dividing

$$y = \sigma(Wx + b) \rightarrow \begin{cases} z = Wx + b \\ y = \sigma(z) \end{cases}$$

FPGA Implementation / Gated Recurrent Unit

MODULE	LATENCY	BRAM	DSP48E	FF	LUT
gru	1,184,679	103	252	49,435	69,352
Total	**1,428,651**	**281 (45%)**	**374 (21%)**	**125,706 (27%)**	**122,239 (53%)**

Table 4: Detail performance of the Design 1 (100 MHz)

MODULE	LATENCY	BRAM	DSP48E	FF	LUT
gru	561,963	231	501	125,206	112,527
Total	**808,989**	**409 (65%)**	**623 (36%)**	**201,477 (43%)**	**165,414 (71%)**

Table 5: Detail performance of the Design 2 (100 MHz)

MODULE	LATENCY	BRAM	DSP48E	FF	LUT
gru	476,501	91	481	134,727	107,321
Total	**728,895**	**297 (47%)**	**757 (43%)**	**357,597 (77%)**	**210,133 (91%)**

Table 6: Detail performance of the Design 3 (200 MHz)

Implementation Result

FPGA Platform

Xilinx Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit.

FPGA Detail Performance (200 MHz)

MODULE	**LATENCY** (cycle)
gru	476,501
conv_offset	241,091
conv_output	240,899
fc_1	13,008
fc_2	12,942
Total	**728,895**

Table 7: Implementation Result

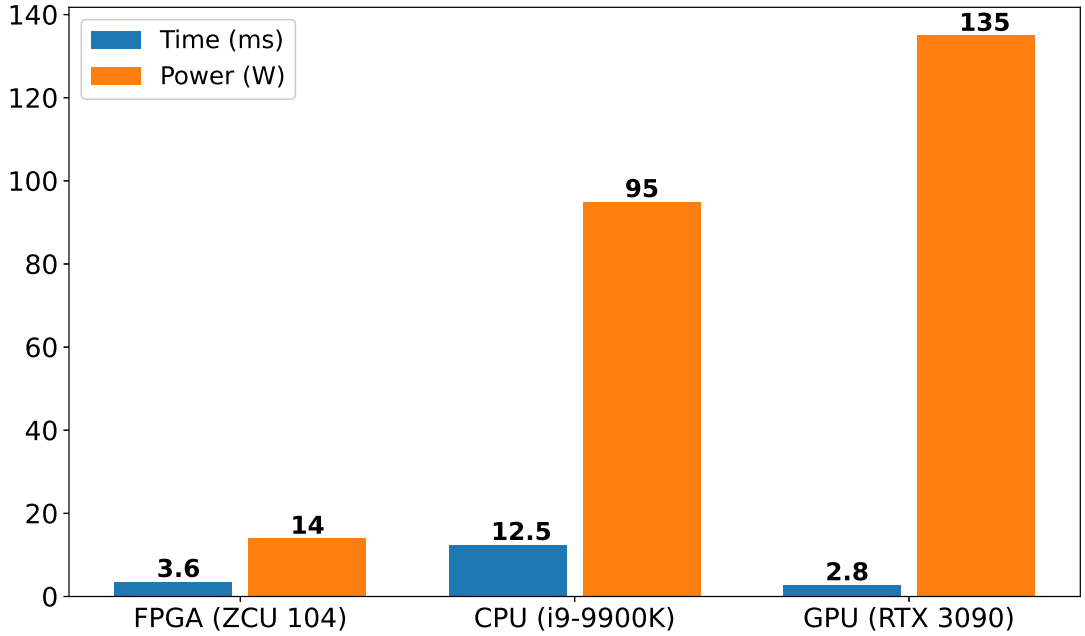


Figure 20: FPGA design performance compares with CPU and GPU.

Summary

KWS

- A new deep neural network model has been proposed to improve the accuracy of the KWS system.

FPGA

- The implementation of KWS system on FPGA has been presented.

Thank you for your attention!

Publication

[1] Duong Van Hai, Nguyen Thi Diem Cai, Tran Nguyen Duc Tho, Pham Hoang, Nguyen Huu Binh, Tran Thi Anh Xuan, and Nguyen Quoc Cuong. "Hệ thống nhận dạng từ khoá tiếng nói trên nền tảng hệ thống nhúng". In: *Hội nghị Khoa học Kỹ thuật Đo lường Toàn quốc lần thứ VII*. 2020

[2] Huu Binh Nguyen, Van Hai Duong, Anh Xuan Tran Thi, and Quoc Cuong Nguyen. "Efficient Keyword Spotting System using Deformable Convolutional Network". In: *IETE Journal of Research* (2020) (**submitted** on December 9, 2020) (**minor revision** on April 14, 2021) (ISI/Scopus)

[3] Binh Nguyen Huu, Hai Duong Van, Thanh Le Tien, Xuan Tran Thi Anh, and Cuong Nguyen Quoc. "EQ-MVDR: Scalable multi-channel speech enhancement using QRNN and ensemble learning". In: *Proc. Interspeech 2021*. 2021 (**submitted** on March 27, 2021)

References

[1] Guoguo Chen, Carolina Parada, and Georg Heigold. "Small-footprint keyword spotting using deep neural networks". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 4087–4091.

[2] Changhao Shan, Junbo Zhang, Yujun Wang, and Lei Xie. "Attention-based End-to-End Models for Small-Footprint Keyword Spotting". In: *Proc. Interspeech 2018* (2018), pp. 2037–2041.

[3] Yuxuan Wang, Pascal Getreuer, Thad Hughes, Richard F Lyon, and Rif A Saurous. "Trainable frontend for robust and far-field keyword spotting". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 5670–5674.

[4] Alice Coucke, Mohammed Chlieh, Thibault Gisselbrecht, David Leroy, Mathieu Poumeyrol, and Thibaut Lavril. "Efficient keyword spotting using dilated convolutions and gating". In: *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 6351–6355.

[5] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, et al. "Deformable convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 764–773.

[7] David Snyder, Guoguo Chen, and Daniel Povey. "Musan: A music, speech, and noise corpus". In: *arXiv preprint arXiv:1510.08484* (2015).

Gradient Descent

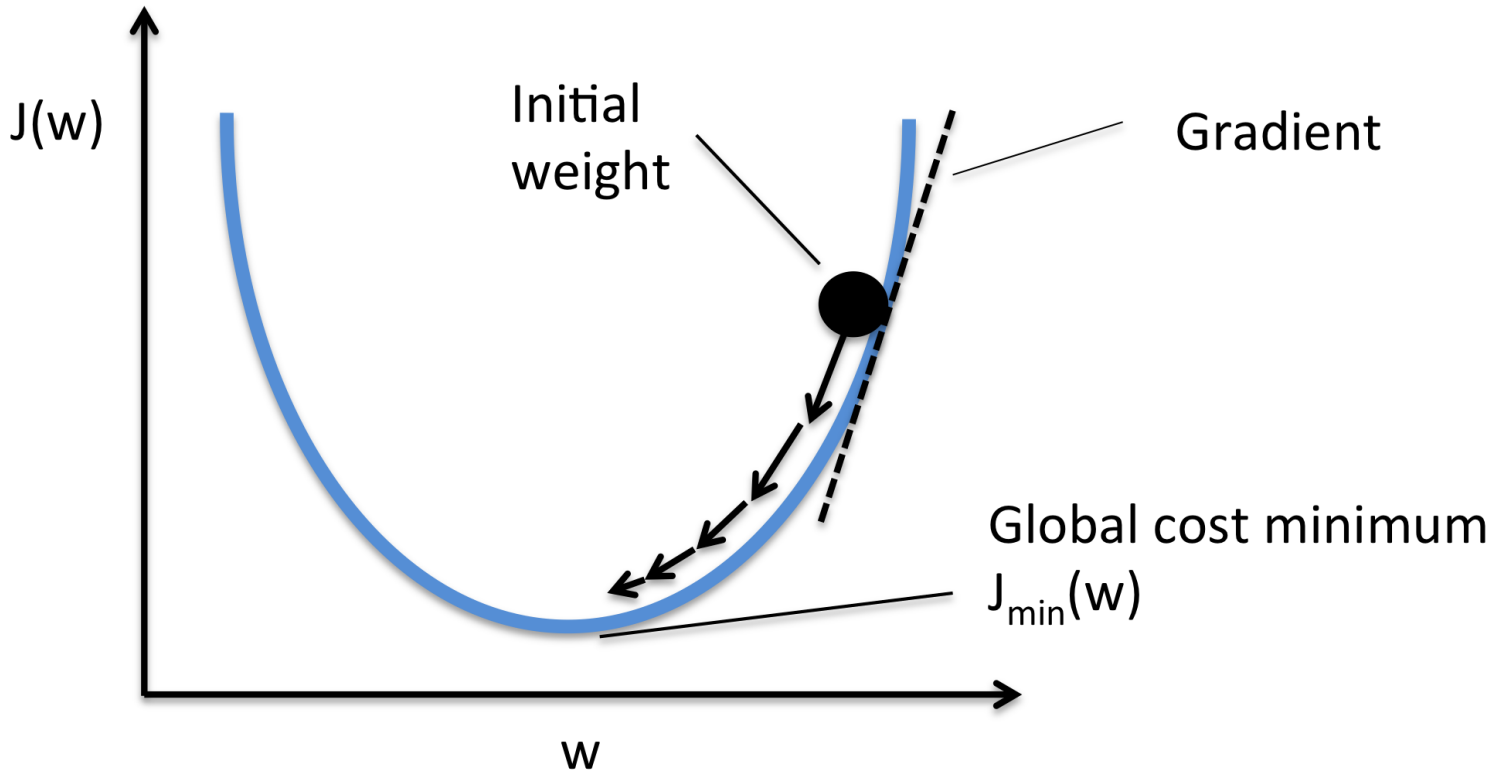


Figure 21: Gradient Descent

Mel-Spectrogram

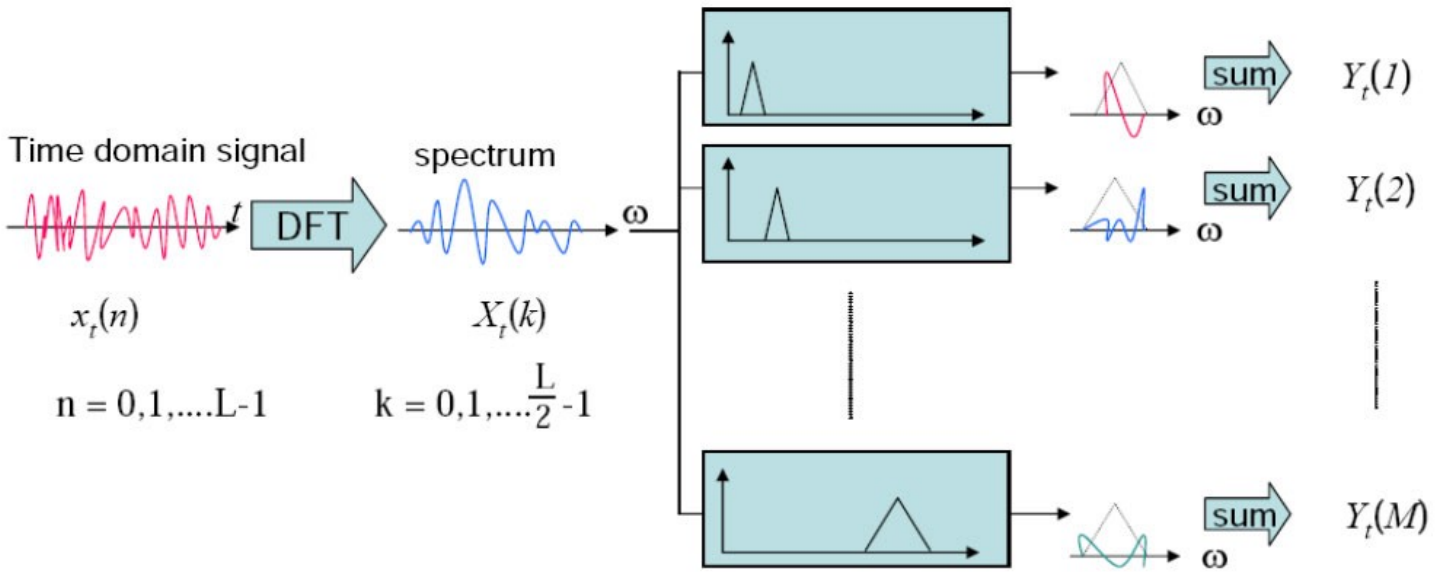


Figure 22: Mel-Spectrogram

HLS Loop Optimization

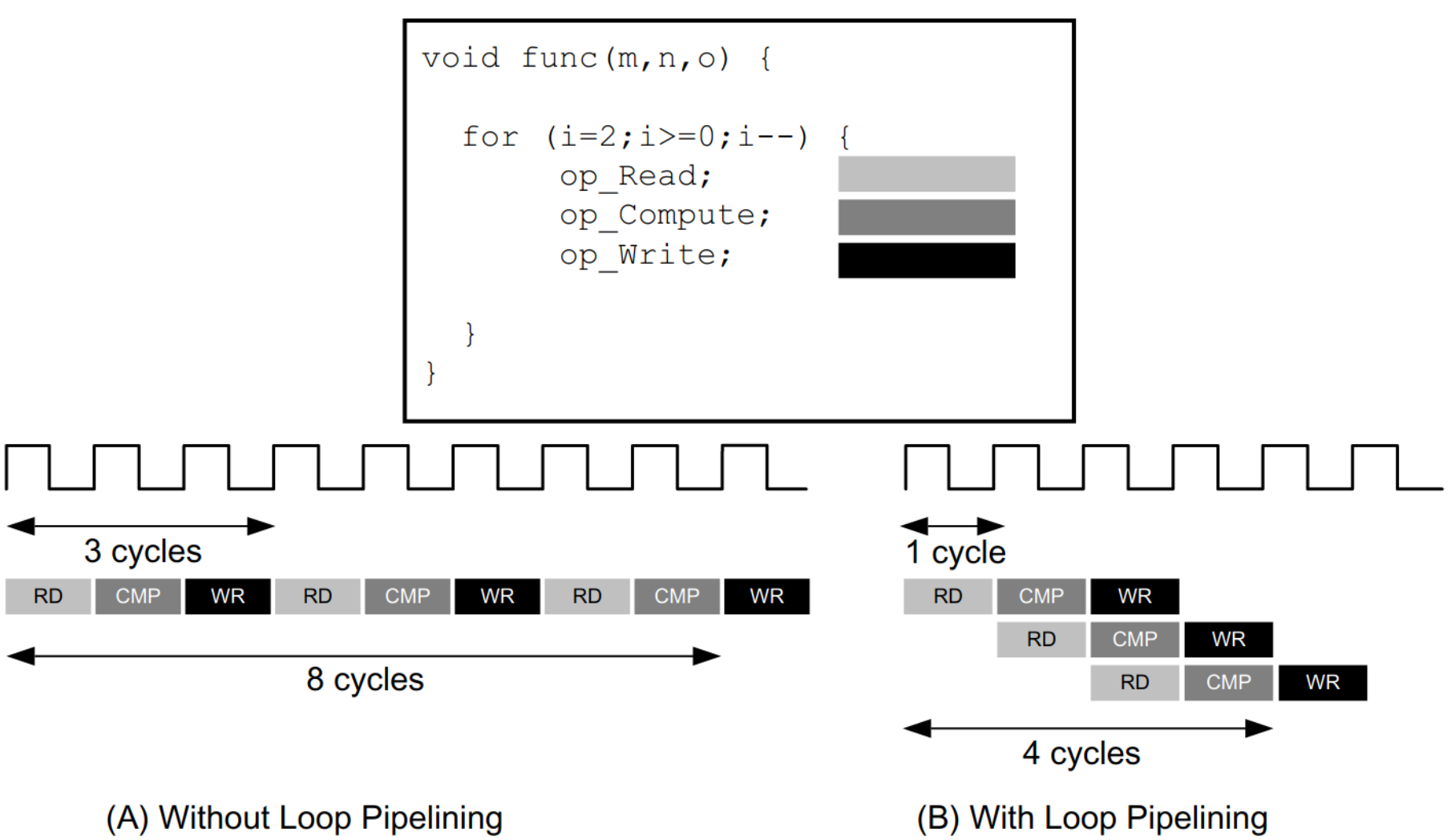
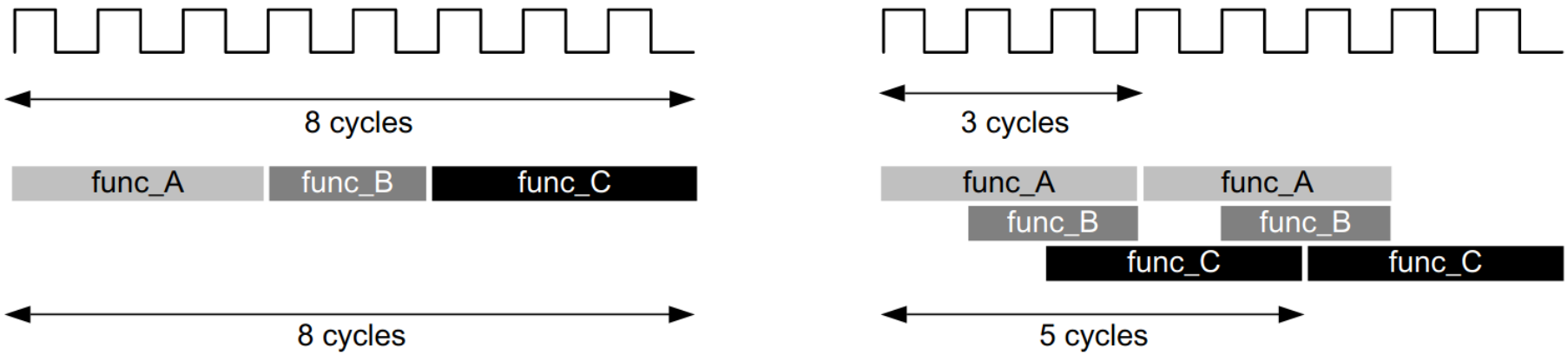


Figure 23: Loop Pipeline

HLS Dataflow Optimization

```
void top (a,b,c,d) {
  ...
  func_A(a,b,i1);
  func_B(c,i1,i2);
  func_C(i2,d)

  return d;
}
```



(A) Without Dataflow Pipelining

(B) With Dataflow Pipelining

Figure 24: HLS Dataflow

DNN-based KWS / Posterior Handling (1)

Posterior smoothing

The smoothed state posterior $\bar{p}_{ij}$ of p_{ij} is computed as:

$$\bar{p}_{ij} = \frac{1}{j - h_{smooth} + 1} \sum_{k=h_{smooth}}^{j} p_{ik} \tag{6}$$

- i is the index of label
- j is the index of frame.
- $h_{smooth} = max\{1, j - w_{smooth} + 1\}$ is the index of the first frame of smoothing window.
- w_{smooth} is the size of smoothing window.

DNN-based KWS / Posterior Handling (2)

Confidence

The confidence score at j^{th} frame c_j is computed as:

$$c_j = \sqrt[n-1]{\prod_{i=0}^{n-1} \max_{h_{max} \leq k \leq j} \bar{p}_{ik}} \tag{7}$$

where

- j is the index of frame.
- n the number of total labels.
- $h_{max} = max\{1, j - w_{max} + 1\}$ is the index of the first frame within the sliding window.
- w_{max} is the size of sliding window.

Gated Recurrent Unit

GRU operation

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \tag{8}$$

$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \tag{9}$$

$$\tilde{h}_t = \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{(t-1)} + b_{hn})) \tag{10}$$

$$h_t = (1 - z_t) \odot \tilde{h}_t + z_t \odot h_{(t-1)} \tag{11}$$

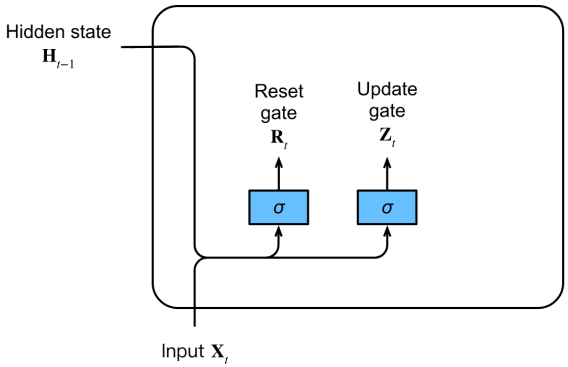


Figure 25: Equation 8-9

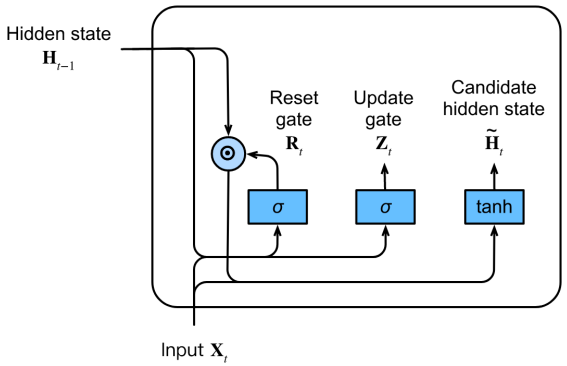


Figure 26: Equation 10

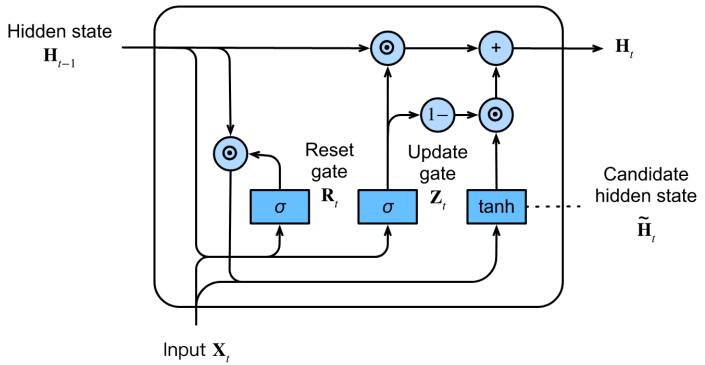


Figure 27: Equation 11

FPGA Implementation Result

MODULE	LATENCY	BRAM	DSP48E	FF	LUT
gru	476,501	91	481	134,727	107,321
dcn_conv_offset	241,091	19	30	35,466	12,384
dcn_conv_output	240,899	9	25	16,394	13,716
attention_fc_1	13,008	9	120	83,851	36,078
attention_fc_2	12,942	0	80	71,626	26,910
Total	**728,895**	**297 (47%)**	**757 (43%)**	**357,597 (77%)**	**210,133 (91%)**

Table 8: Detail performance of the Design 3 (200 MHz)