Hello Edge: Keyword Spotting on Microcontrollers

Yundong Zhang, Naveen Suda, Liangzhen Lai and Vikas Chandra
ARM Research, Stanford University
arXiv.org, 2017

Presented by **Mohammad Mofrad**
University of Pittsburgh
March 20, 2018

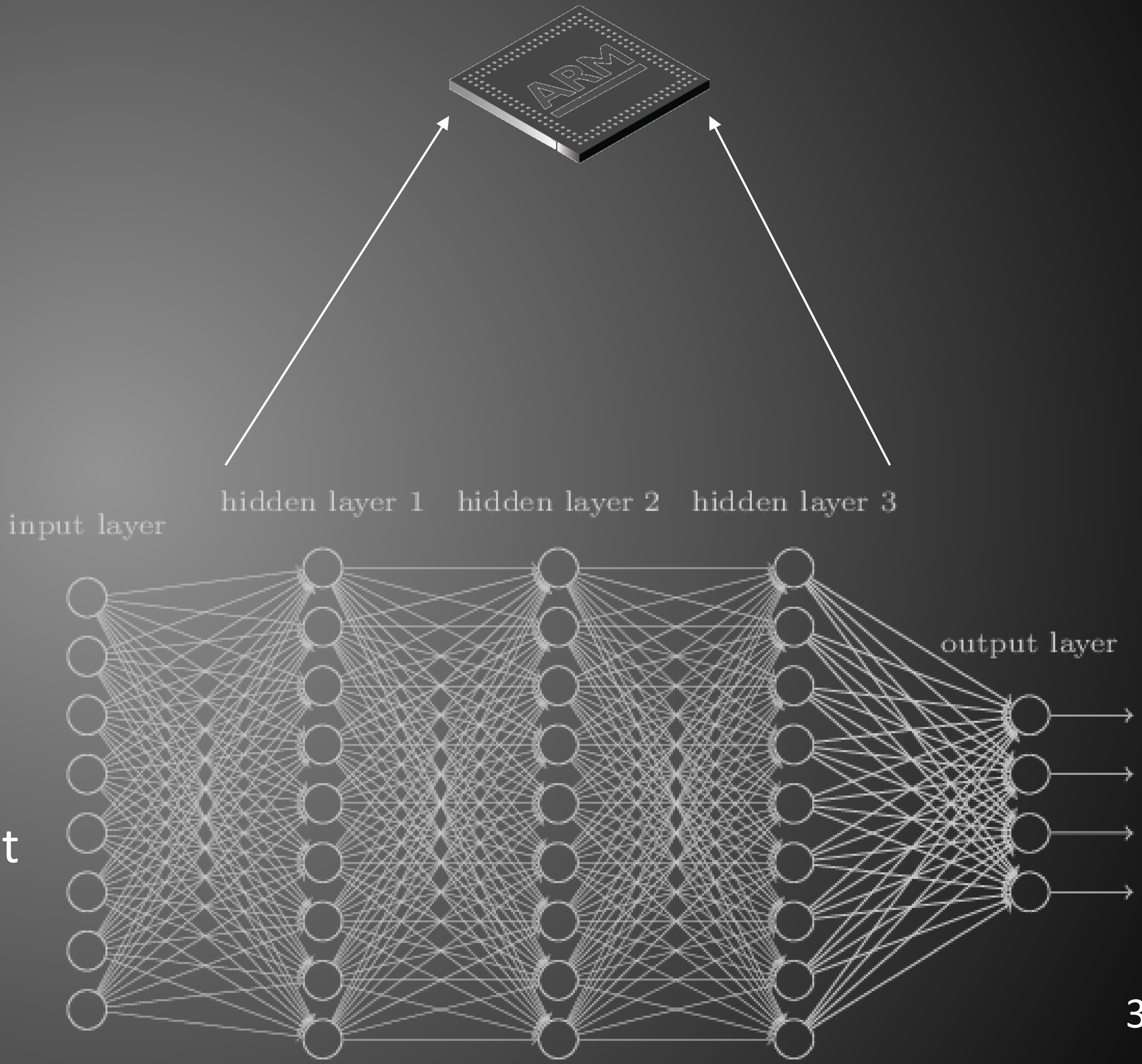
Problem Statement

- Speech in increasingly becoming a natural way to interact with electronic devices:
    - Amazon echo
    - Google home
    - Smart homes


- Keyword Spotting (KWS) is the process of detecting commonly known keywords. Example of KWS is “Alexa”, “Ok Google”, and “Hey Siri”
- In smart speakers Keyword Spotting (KWS) is used to:
    - Save energy
    - Avoid Cloud latency

Contribution

- Train a neural network for running **Keyword Spotting** on resource-constrained microcontrollers
  - 1st constraint: Limited memory footprint
  - 2nd constraint: Limited processing power
  - Optimize the neural network for these constraints without sacrificing accuracy
  - And meet low latency requirement



Keyword Spotting (KWC)

$T = (L - l\ /\ s) + 1$ frames

Signal of length L

Overlapping frames of length l

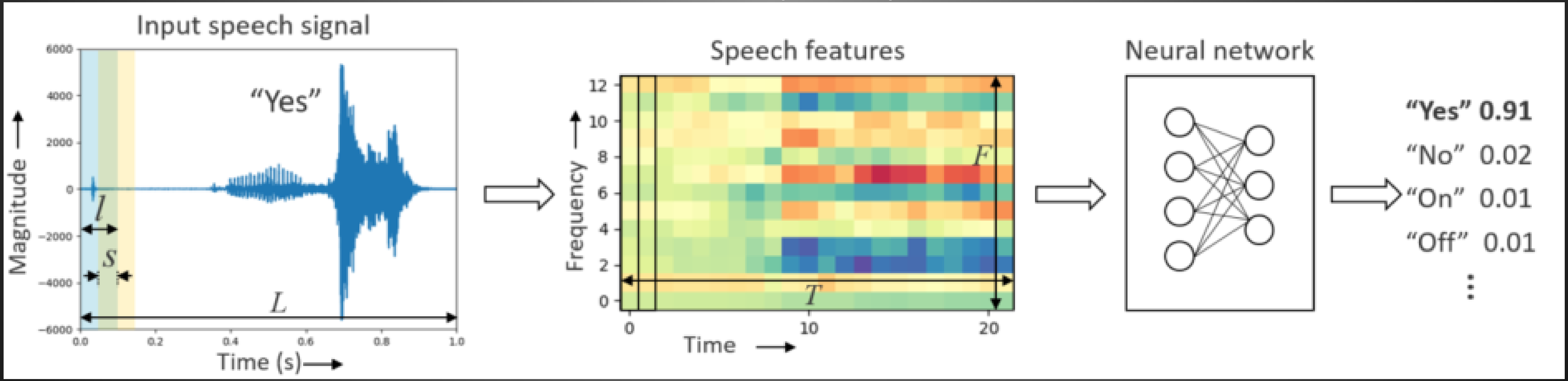
Stride s

$T \times F$ features

using Log-mel filter bank energies (LFBE) and Mel-frequency Cepstral Coefficients (MFCC)

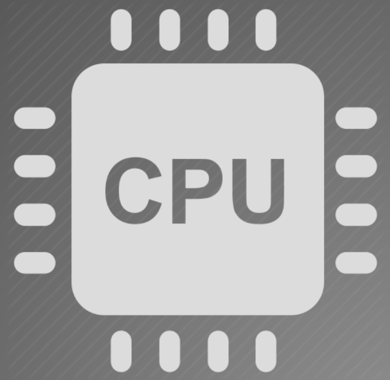
Speech feature matrix

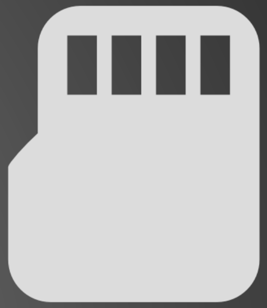
is fed to a classifier which generates the probability of output classes



Microcontroller Systems

- Microcontroller
  - Processor core
  - On-chip SRAM
  - On-chip embedded flash





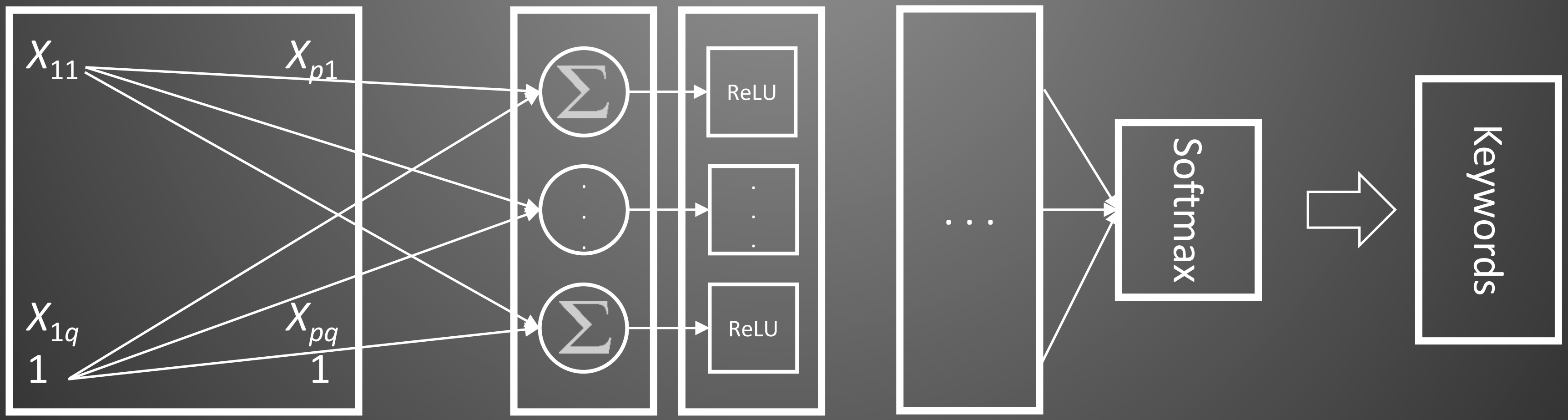
- ARM MbedTM platform
  - Processor: Cortex-M0 → Cortex M7
  - Frequency: 48 MHz → 216 MHz
  - SRAM: 8 KB → 320 KB
  - Flash: 32 KB → 1MB
- Mostly, microprocessors are designed for low cost and energy efficient applications.
- Integrated DSPs and SIMD and MAC instructions can accelerate neural network computations

Neural Networks for Keyword Spotting

1. Deep Neural Network (DNN)
2. Convolutional Neural Network (CNN)
3. Recurrent Neural Network (RNN) for KWS
4. Convolutional Recurrent Neural Network (CRNN) for KWS
5. Depthwise Separable Convolutional Neural Network (DS-CNN) for KWS

Deep Neural Network (DNN)

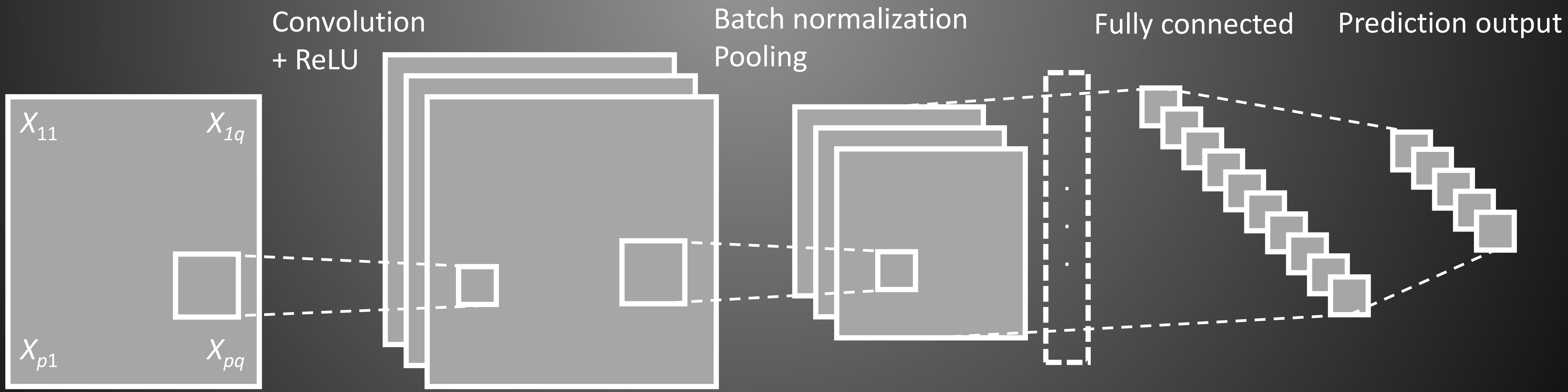
- Input: Flattened feature matrix
- *d x n* layers (*d* layers each having *n* neurons)
- Each layer is followed by a rectified linear unit (ReLU) activations
- Output layer is a softmax generating probabilities for *k* keywords



$X_{pxq} = X'_{qxp}$ i.e. X in $x_1 \ldots x_q$ *d layers of n neurons* *k keywords*

Convolutional Neural Network (CNN)

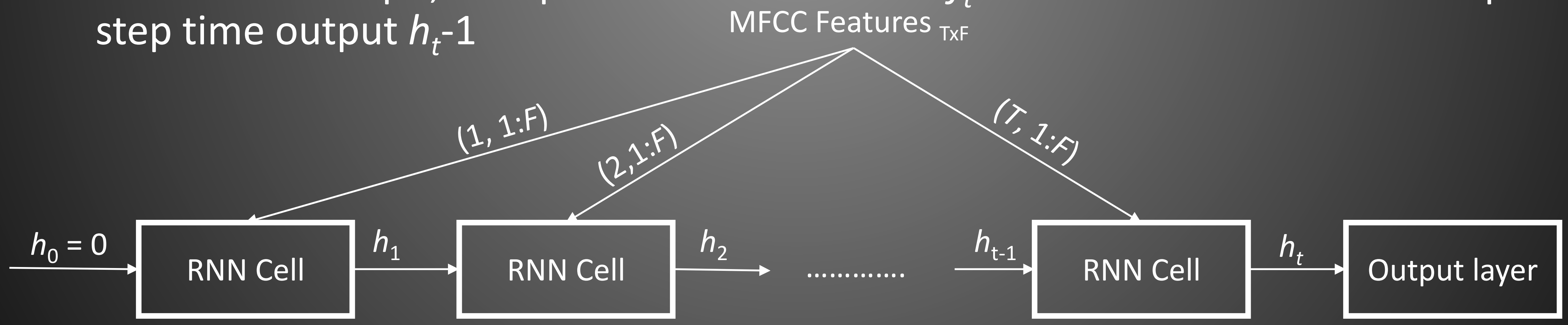
- DNN fails to efficiently model
  - Local temporal and spectral correlation in the speech features
- CNNs exploit this correlation by
  - treating the input time-domain and spectral-domain features as an image
  - performing 2-D convolution operations over it



X^i_{pxq} i.e. X^i in x_{11} … x_{1q}, …, x_{p1},… x_{pq}

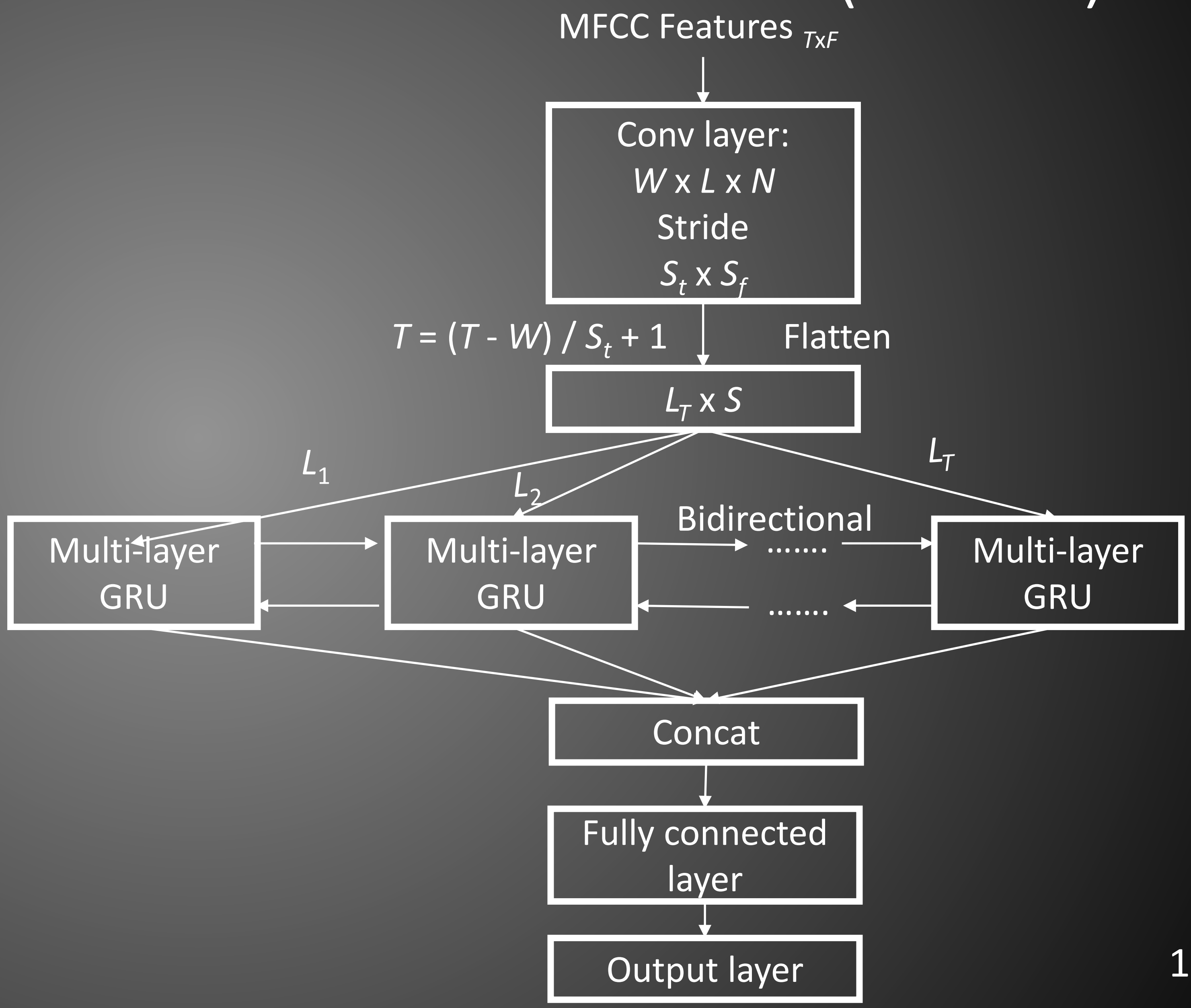
Recurrent Neural Network (RNN)

- RNNs exploits temporal relation between signals
- RNNs captures long term dependencies using “gating” mechanism
  - RNN cells can be of type Long short-term memory (LSTM) or Gated Recurrent Unit (GRU)
    - Input, output, and forget gate
- RNNs operate for T time steps
  - In each time step *t*, the spectral feature vector $f_t \in R^F$ concatenated with the previous step time output h_t-1



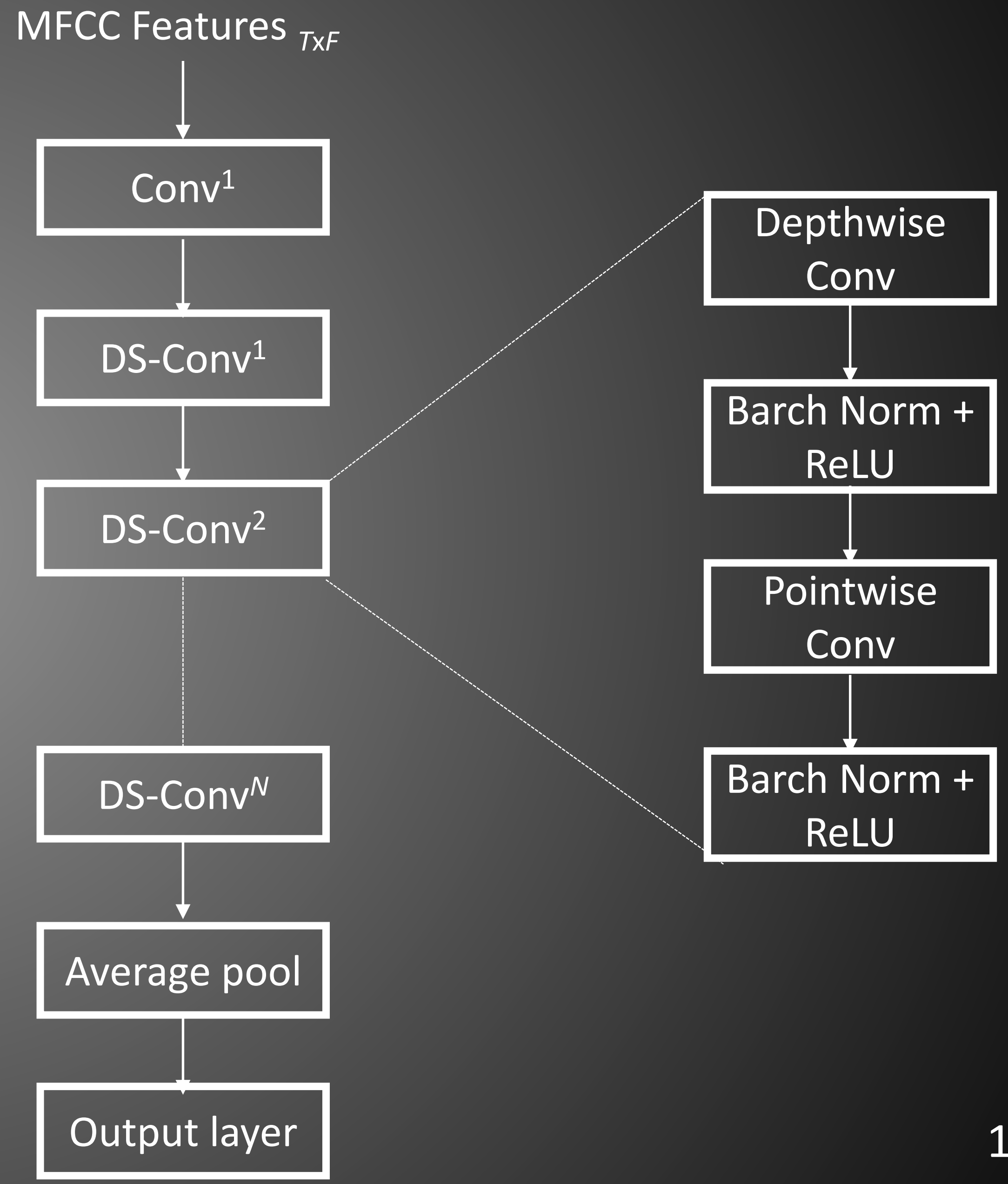
Convolutional Recurrent Neural Network (CRNN)

- CNN + RNN
  - Exploit temporal / spatial correlations using
    - Convolutional layers
    - Global temporal dependencies in the speech features using recurrent network
  - Network is bidirectional
    - More learning capacity
  - GRUs VS LSTM
    - Fewer parameters
    - Better convergence



Depthwise Separable Convolutional Neural Network (DS-CNN)

- DS-CNN replace the 3D convolutional operation of CNN into 2D convolutions followed by 1D convolutions
  - A 2D filter is used to convolve each channel in the input feature
  - A 1D filter is used to convolve the outputs in the depth dimension
- Compared to CNN, DS-CNN is more efficient in terms of
  - Number of parameters
  - Number of operations
- Thus, we can have deeper and wider architectures



Experimental Setup

- Google speech commands dataset
  - 65K 1 second audio clips of 30 keywords including:
    - "Yes", "No", "Up", "Down“, "Left", "Right", "On", "Off", "Stop", "Go", along with "silence" (i.e. no word spoken) and "unknown“ word, which is the remaining 20 keywords from the dataset.
    - 80:10:10 → Training: validation: test
- All neural networks are trained in Google Tensorflow framework
  - Cross entropy loss
  - Adam optimizer
  - Batch size of 100
  - Initial training of 10K iterations with learning rate 5 x 10^{-4}
  - Then, training of 10K iterations with learning rate 1 x 10^{-4}
  - Background noise and random time shift up to 100ms is added to training data

Training results

- Memory for activations is reused across different layers
- Operations include multiplications and additions in the matrix multilpication operations in each layer in the network

NN Architecture	Accuracy	Memory	Operations
DNN	84.3 %	288 KB	0.57 Mops
CNN-1	**90.7 %**	**556 KB**	**76.02 Mops**
CNN-2	84.6 %	149 KB	1.46 Mops
LSTM	**88.8 %**	**26 KB**	**2.06 Mops**
CRNN	87.8 %	298 KB	5.85 MOps

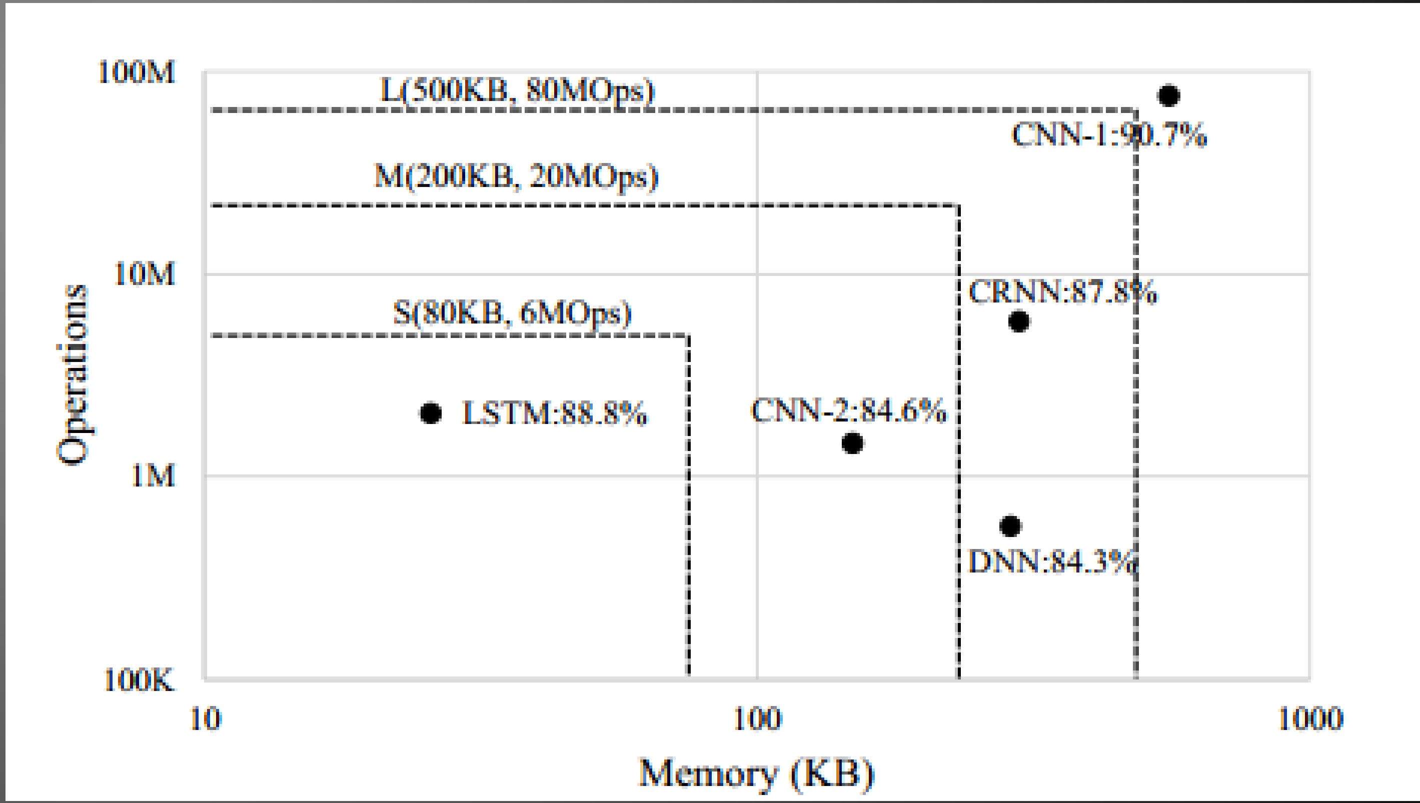
Resource constrained Neural Networks

- Keyword spotting considerations on microcontrollers
  - Memory footprint
  - Execution time

Neural network size	Neural network limit	Operations / inference limit
Small (S)	80 KB	6 MOps
Medium (M)	200 KB	20 MOps
Large (L)	500 KB	80 MOps

Resource constrained Neural Network Architecture Exploration

- Ideal model would have
  - High accuracy
  - Small memory footprint
  - Lower number of computations
- Figure belonged to prior works trained on speech commands dataset

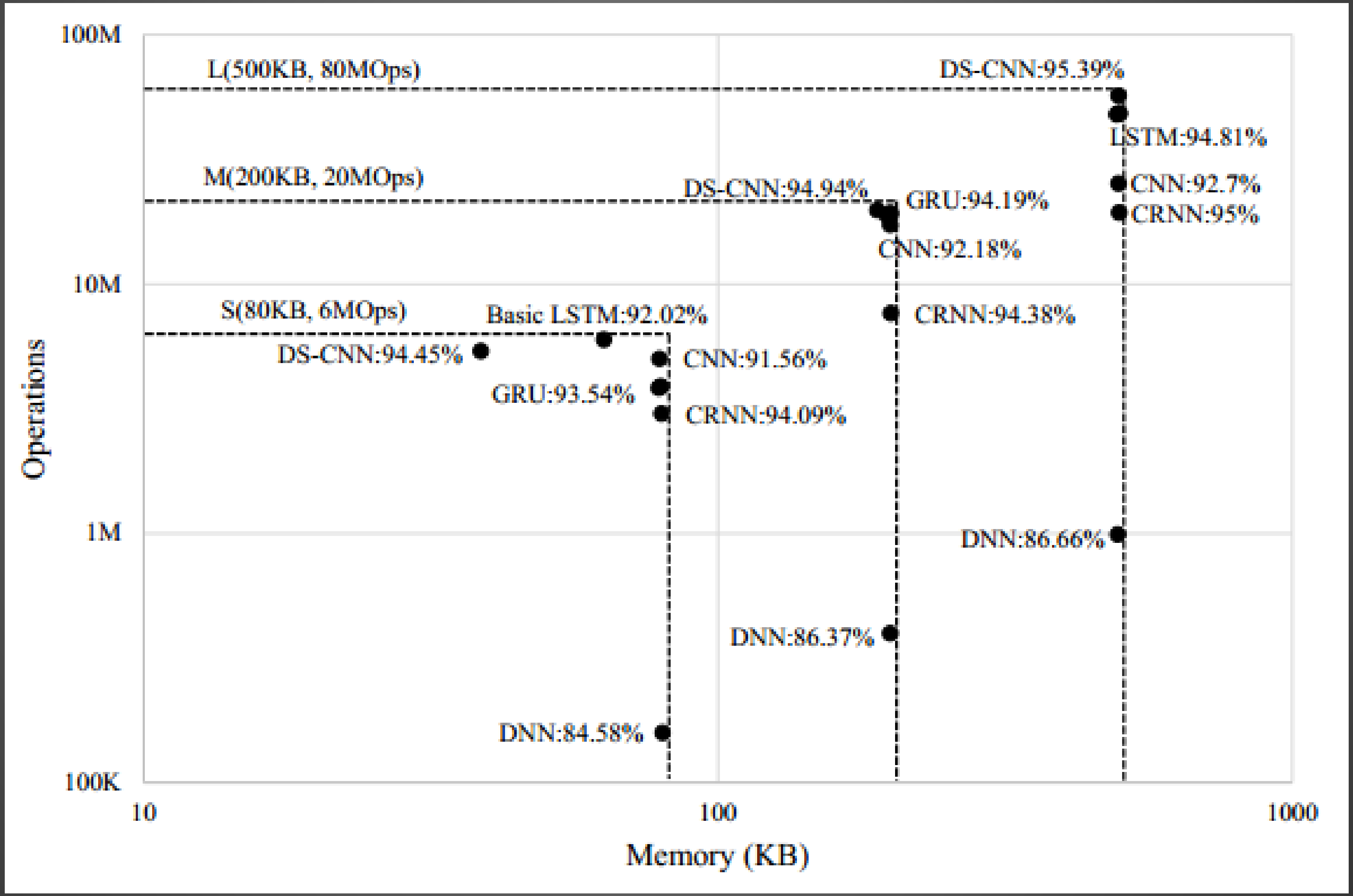


Summery of Best Neural Networks results

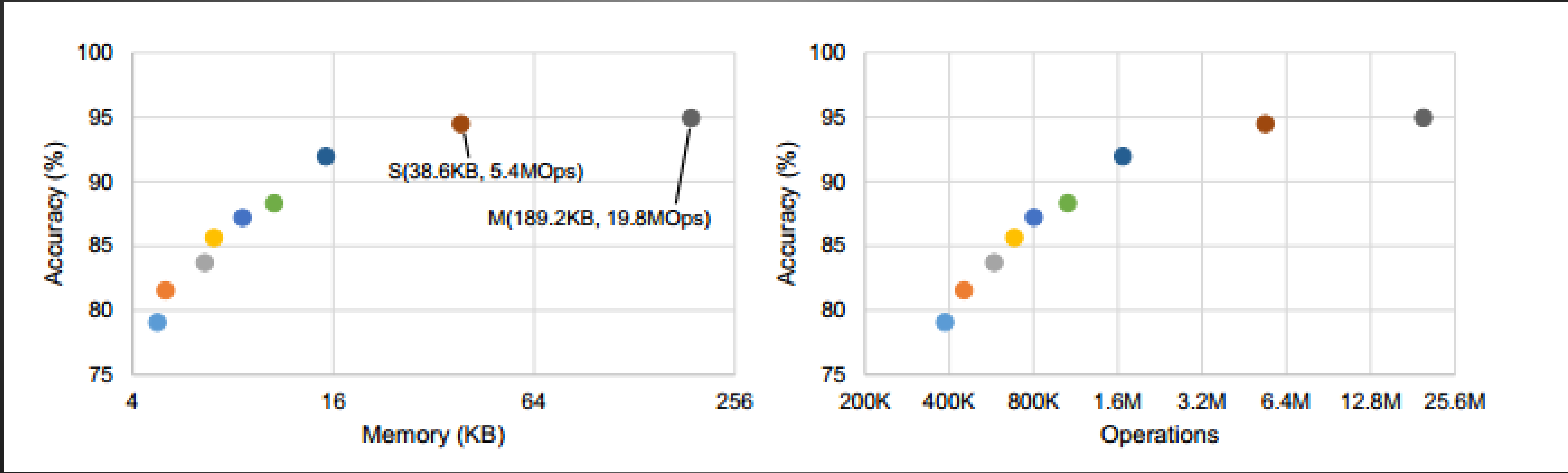
- An exhaustive search of feature extraction of hyperparameters followed by a manual selection to narrow down the search space.
- DNNs are memory bound

NN model	S(80KB, 6MOps)			M(200KB, 20MOps)			L(500KB, 80MOps)		
	Acc.	Mem.	Ops	Acc.	Mem.	Ops	Acc.	Mem.	Ops
DNN	84.6%	80.0KB	158.8K	86.4%	199.4KB	397.0K	86.7%	496.6KB	990.2K
CNN	91.6%	79.0KB	5.0M	92.2%	199.4KB	17.3M	92.7%	497.8KB	25.3M
Basic LSTM	92.0%	63.3KB	5.9M	93.0%	196.5KB	18.9M	93.4%	494.5KB	47.9M
LSTM	92.9%	79.5KB	3.9M	93.9%	198.6KB	19.2M	94.8%	498.8KB	48.4M
GRU	93.5%	78.8KB	3.8M	94.2%	200.0KB	19.2M	94.7%	499.7KB	48.4M
CRNN	94.0%	79.7KB	3.0M	94.4%	199.8KB	7.6M	95.0%	499.5KB	19.3M
DS-CNN	94.4%	38.6KB	5.4M	94.9%	189.2KB	19.8M	95.4%	497.6KB	56.9M

Summery of Memory vs Operations vs Accuracy

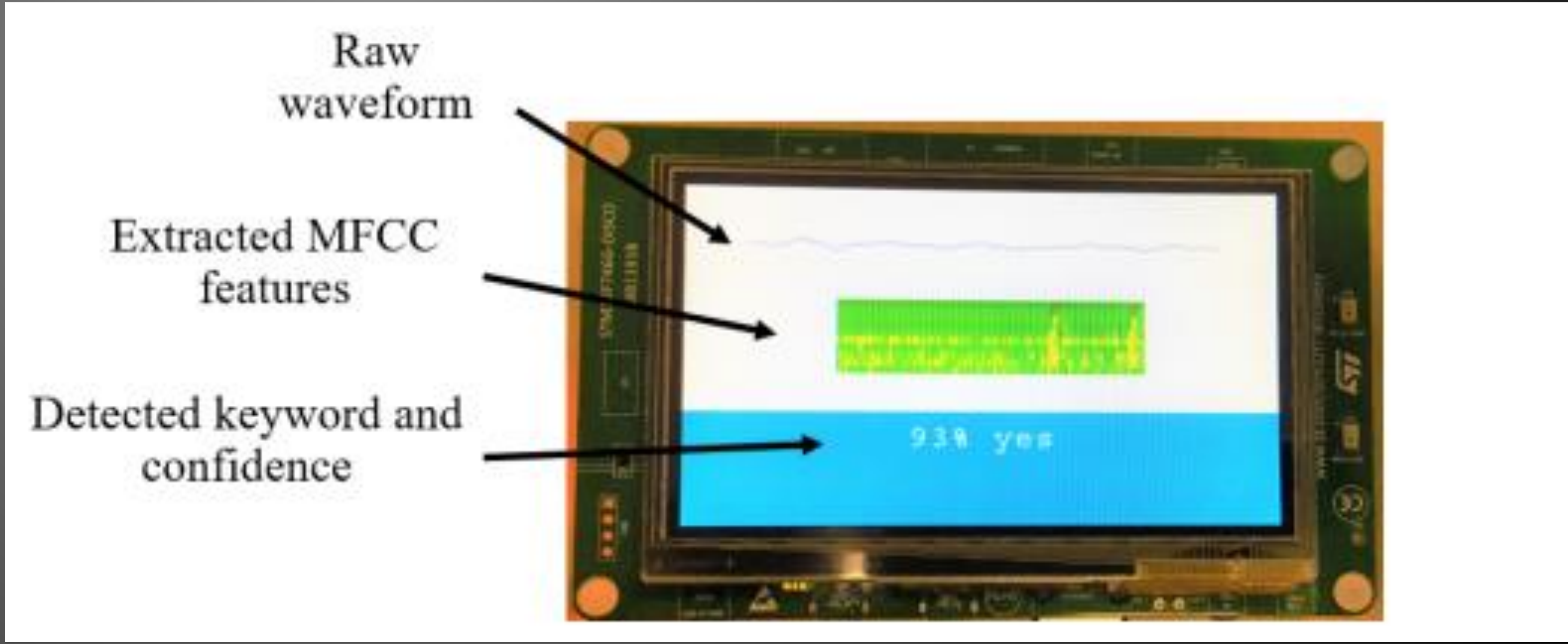


Accuracy vs Memory and Operations of different DS-CNN



KWS Deployement on Microcontoller

- STM32F746G-DISCO board
  - Cortex –M7
  - CMSIS-NN kernels
  - 8-bit weights
  - 8-bit activations
  - 10 inference per second
  - MFCC feature extraction plus DNN execution takes about 12 ms
- Application
  - ~70 KB memory
    - ~66 KB weights
    - ~1 KB activations
    - ~2 KB audio I/O and MFCC features



Conclusion

- They design a hardware optimized neural network for microcontrollers which is memory and compute efficient
- They carry out the task of keyword spotting
- They explore the hyperparameter search space and suggest parameter settings for memory/compute constrained neural networks