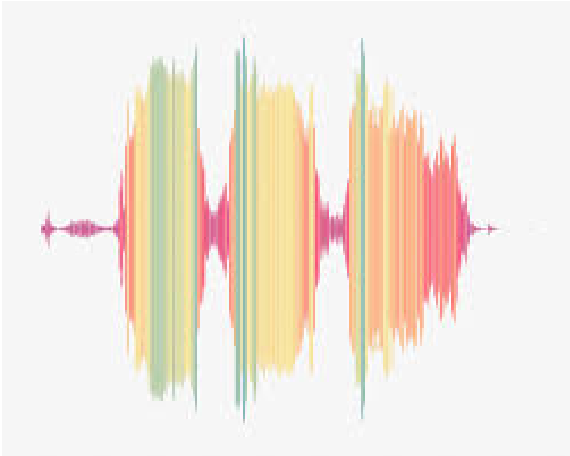
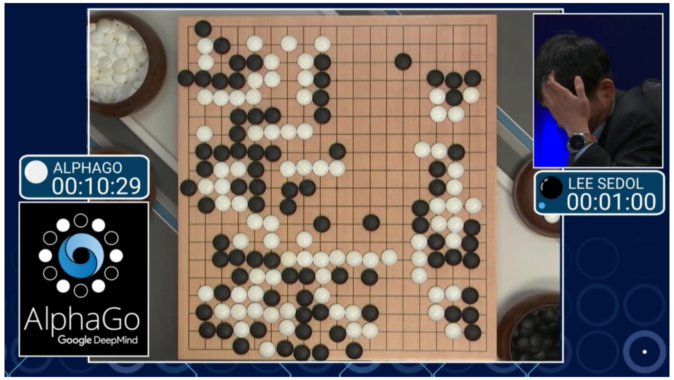
Institut Mines-Télécom

Processing and Training Deep Neural Networks on Chip

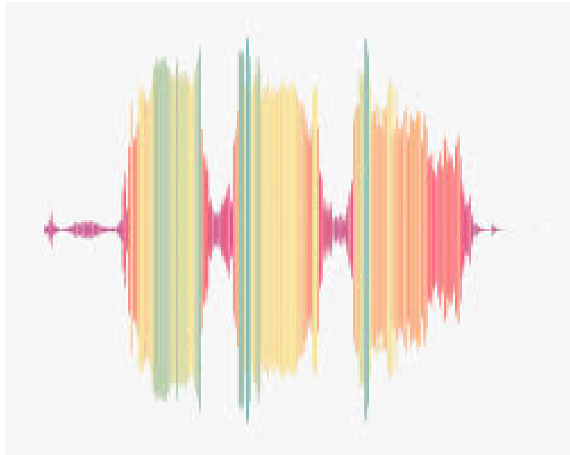
Ghouthi BOUKLI HACENE, Vincent GRIPON, Nicolas FARRUGIA, Matthieu ARZEL, Michel JEZEQUEL, Yoshua BENGIO




Phone
Mail
Messages
What can I help you with?

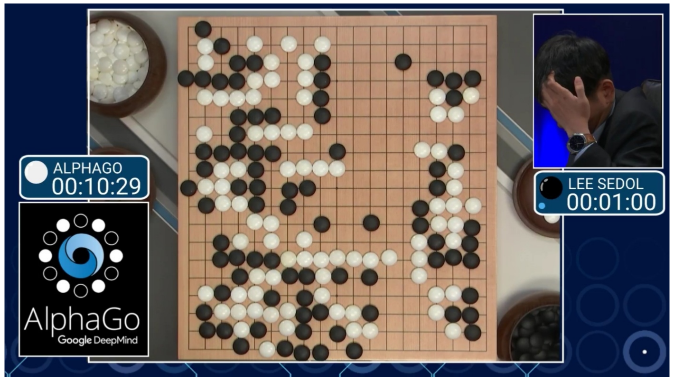

ALPHAGO
00:10:29
AlphaGo
Google DeepMind
LEE SEDOL
00:01:00

1T FLOPs for one decision



100M parameters to learn

1024 V100 during 1 day for training



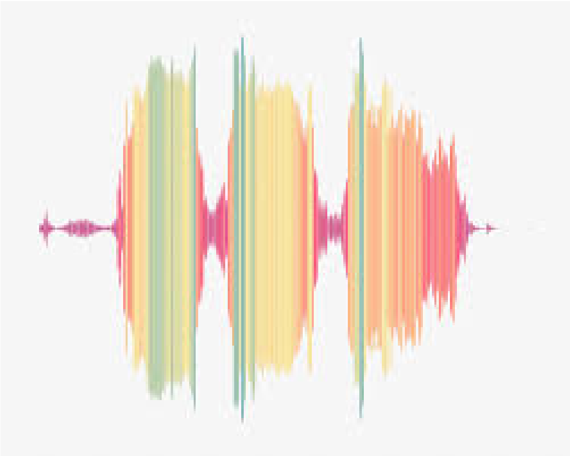
4 TPUs during 1 month for training

Challenges

Technical Challenges

- Real time applications.
- Running deep learning on limited resources embedded systems.

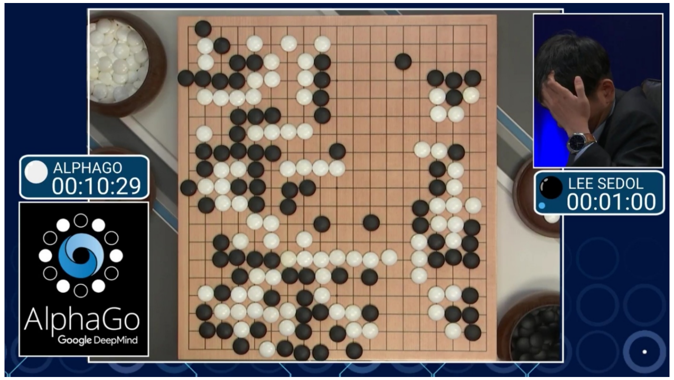
1T FLOPs for one decision



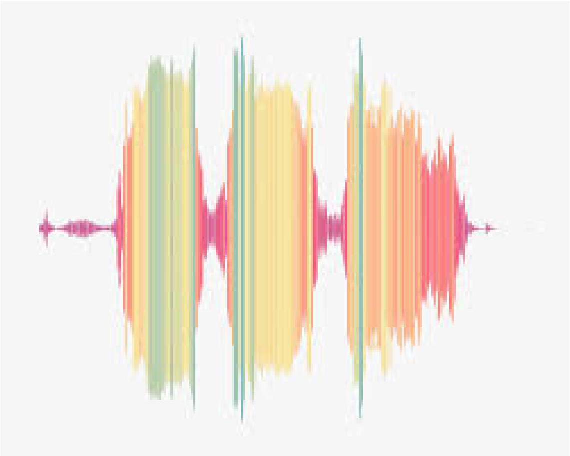
100M parameters to learn

Scientific Challenges

- Large architectures harden visualization and interpretation.
- Simulation time limits the progress of the field.



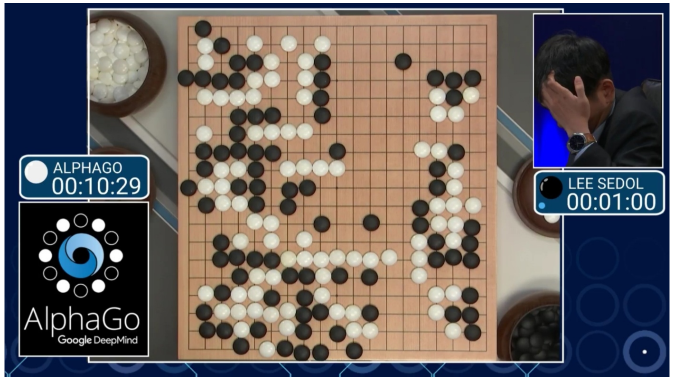
4 TPUs during 1 month for training



100M parameters to learn

Societal Challenges

- Large energy consumption.
- Accessibility of deep learning to everyone.



4 TPUs during 1 month for training

1024 V100 during 1 day for training

Outline

1. **Deep Learning**
   - 1.1 Deep Learning
   - 1.2 Some DNN Architectures
   - 1.3 Importance of the Architecture
2. **Efficient Inference**
   - 2.1 Reducing DNNs Size
   - 2.2 Compression Methods
   - 2.3 Quantization
   - 2.4 Pruning
3. **Conclusion**

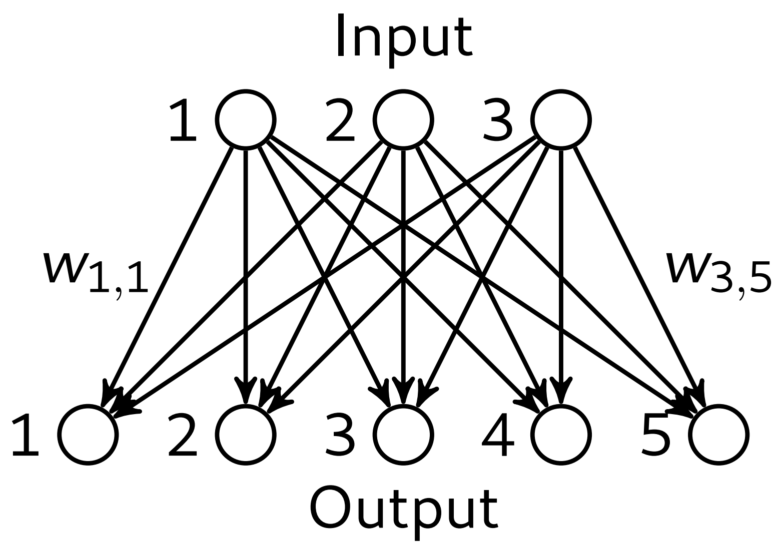
Deep Learning

- A deep learning architecture is basically an assembly of functions.
- Each function can be represented by an entity called layer.
- Two most important layers:

- A deep learning architecture is basically an assembly of functions.
- Each function can be represented by an entity called layer.
- Two most important layers:

- A deep learning architecture is basically an assembly of functions.
- Each function can be represented by an entity called layer.
- Two most important layers:

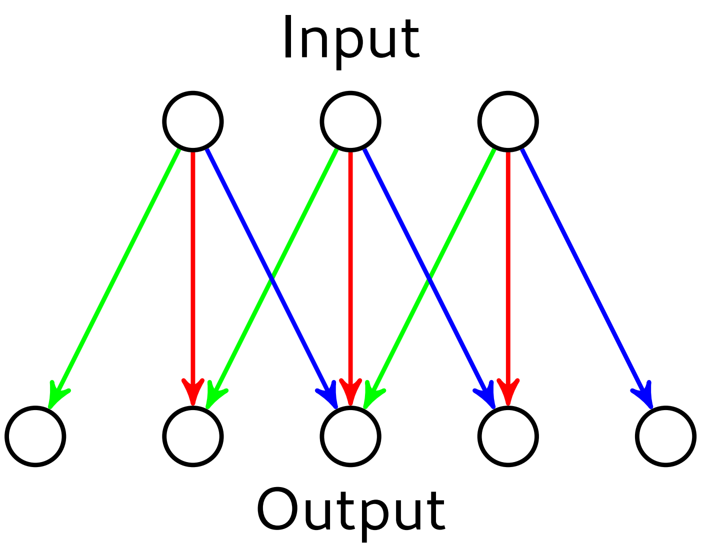
Fully connected layer



$$\begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} \end{pmatrix}$$

- A deep learning architecture is basically an assembly of functions.
- Each function can be represented by an entity called layer.
- Two most important layers:

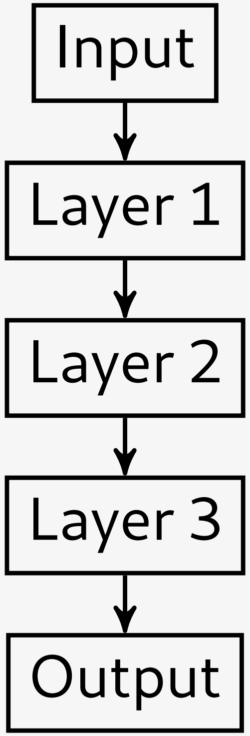
Convolutional layer



$$\begin{pmatrix} w_1 & w_2 & w_3 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 \end{pmatrix}$$

Some DNN Architectures

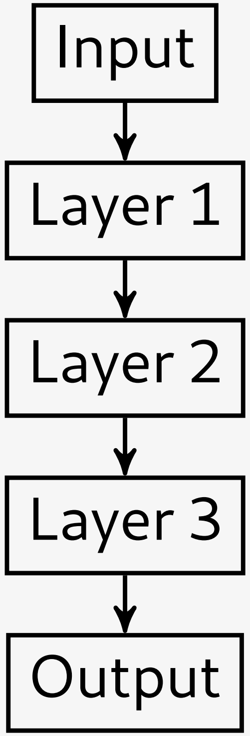
AlexNet-like Network



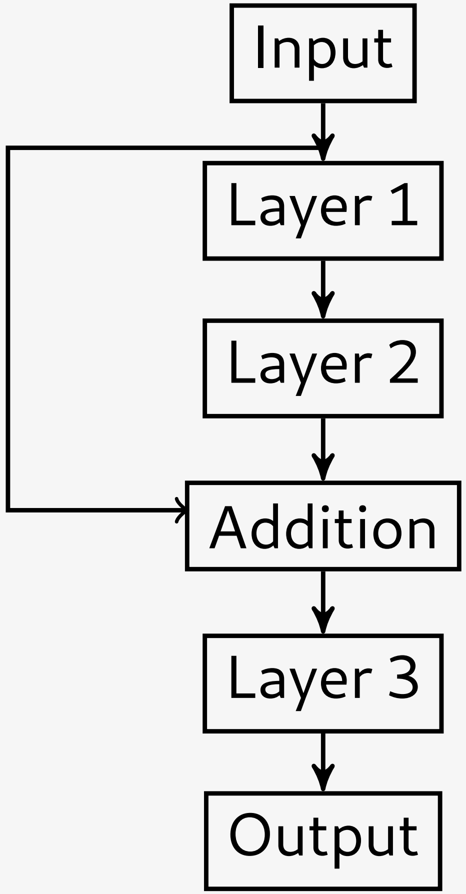
Residual-like Network

Some DNN Architectures

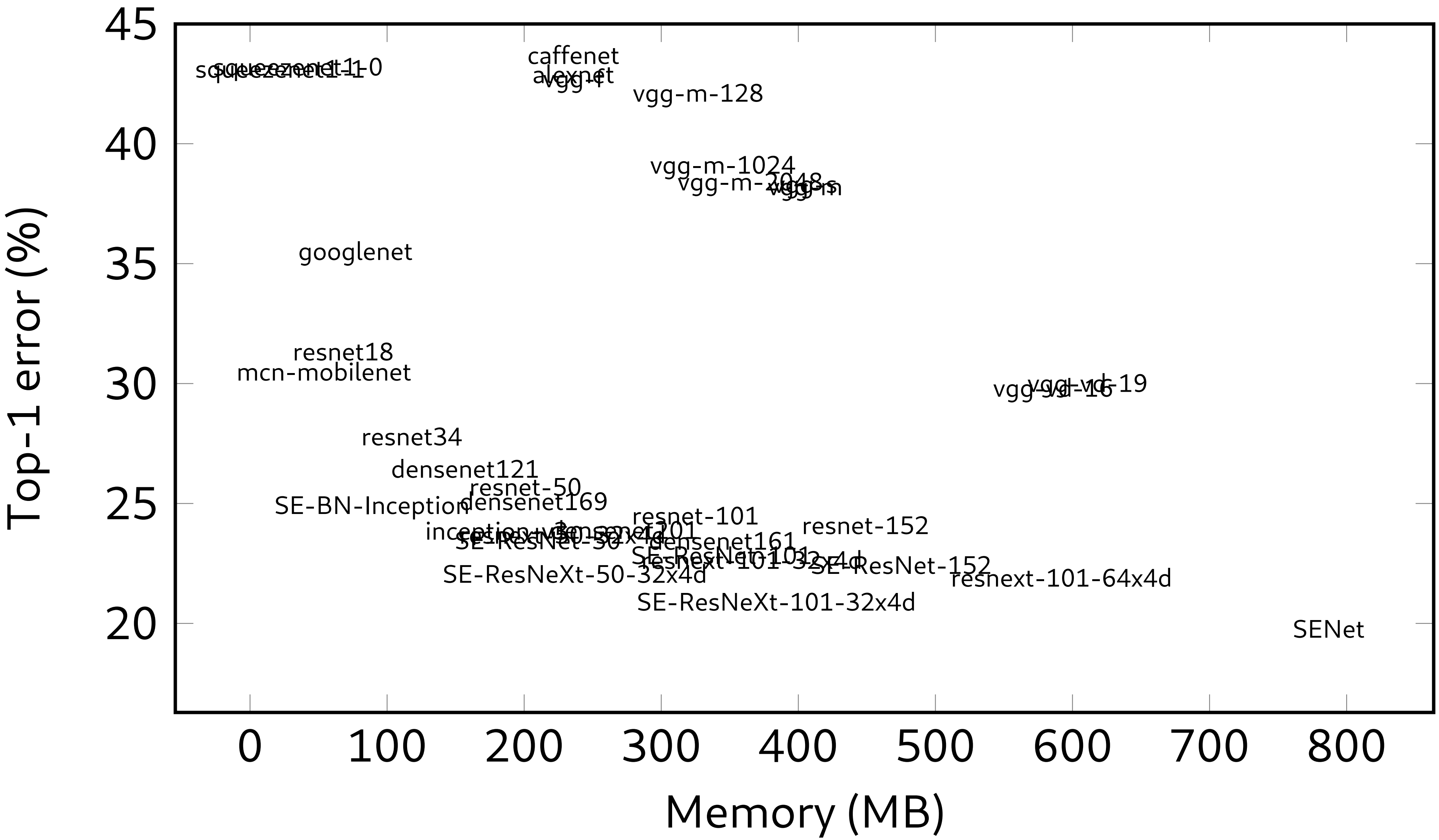
AlexNet-like Network



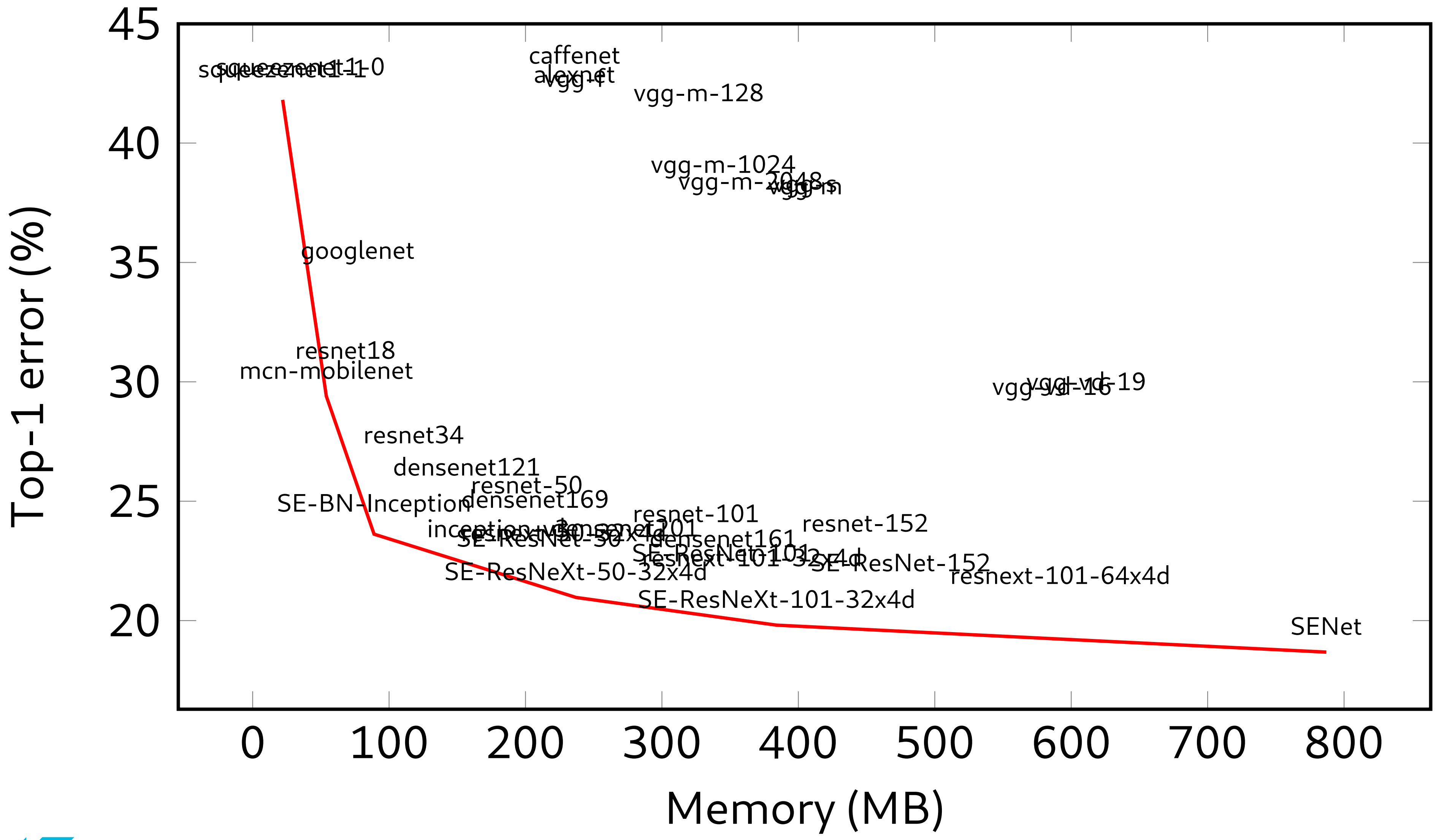
Residual-like Network



Memory vs. Top-1 error for models trained on 2012 ILSVRC

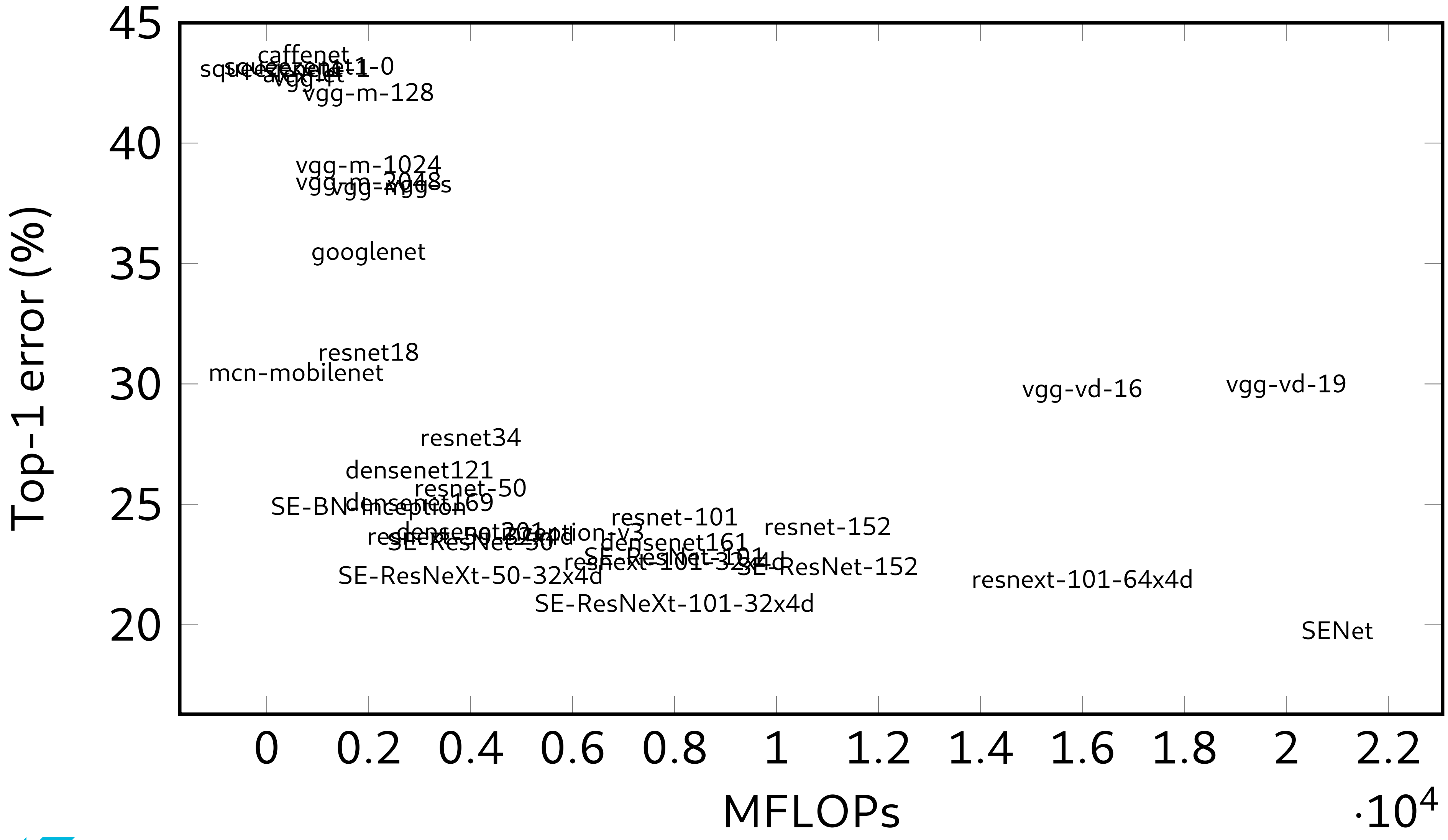


Memory vs. Top-1 error for models trained on 2012 ILSVRC



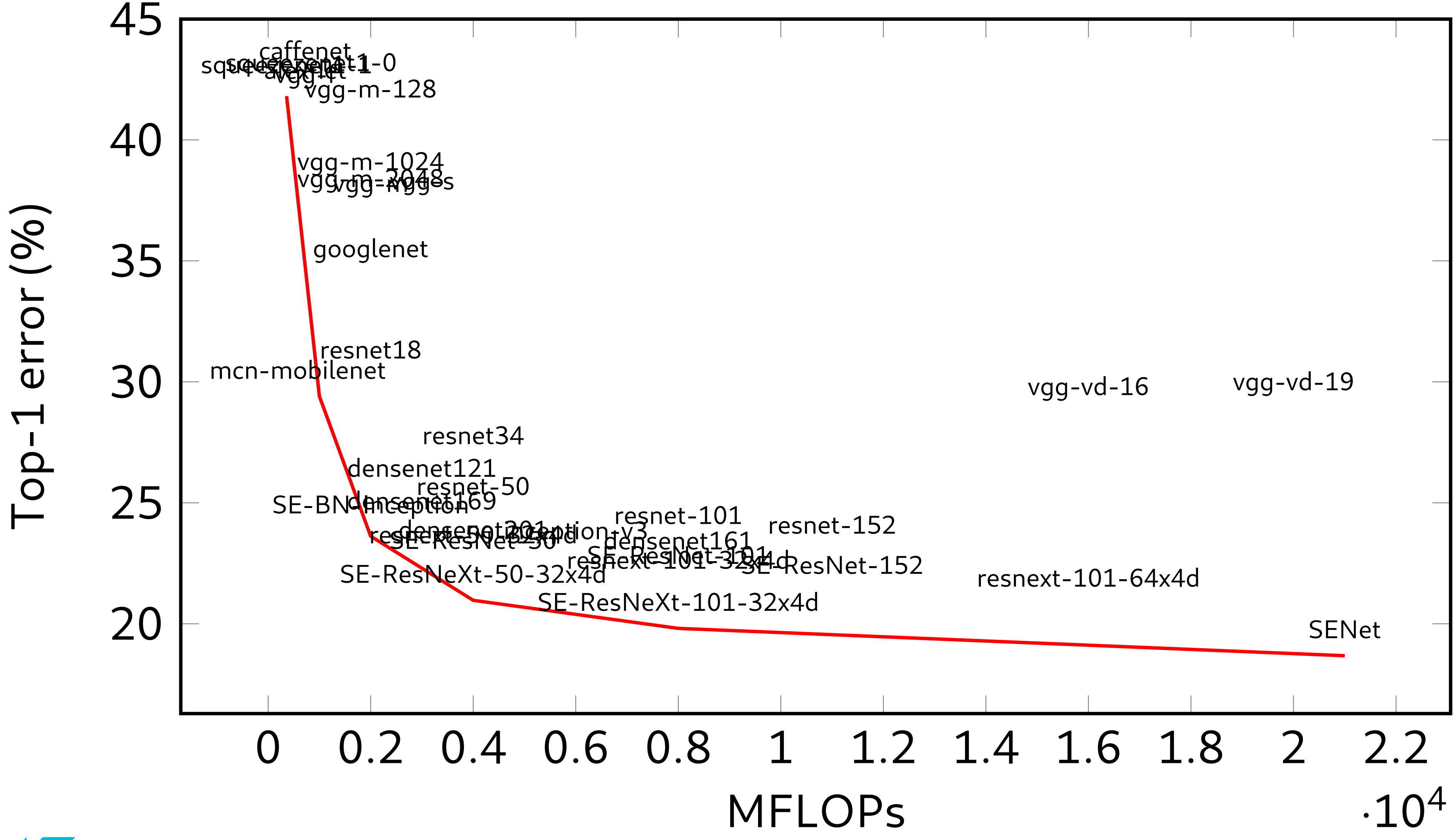
Importance of the architecture: FLOPs

FLOPs vs. Top-1 error for models trained on 2012 ILSVRC

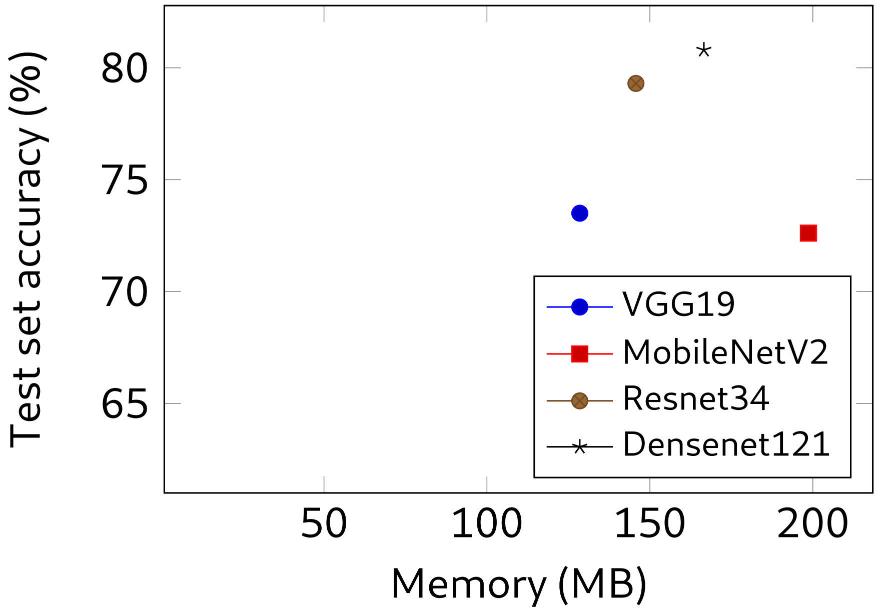


Institut Mines-Télécom

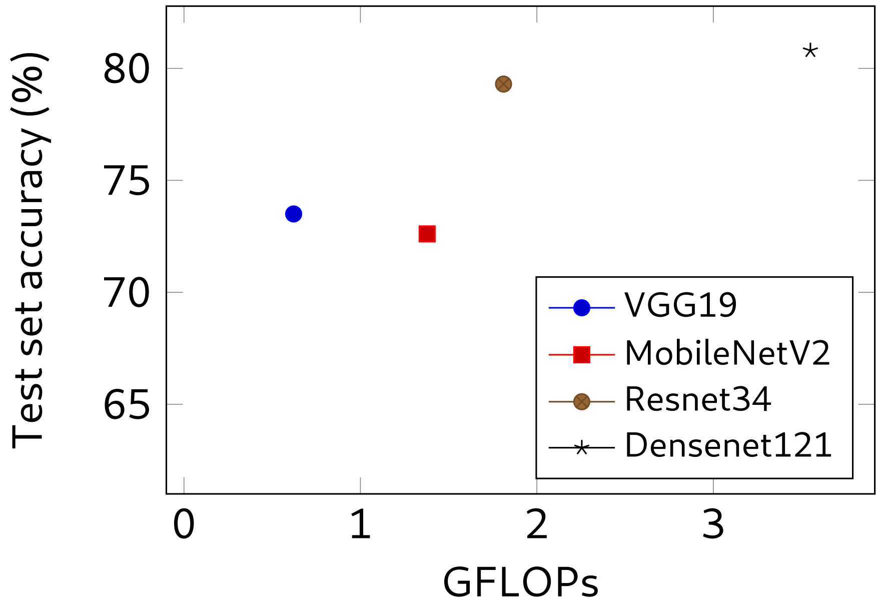
FLOPs vs. Top-1 error for models trained on 2012 ILSVRC



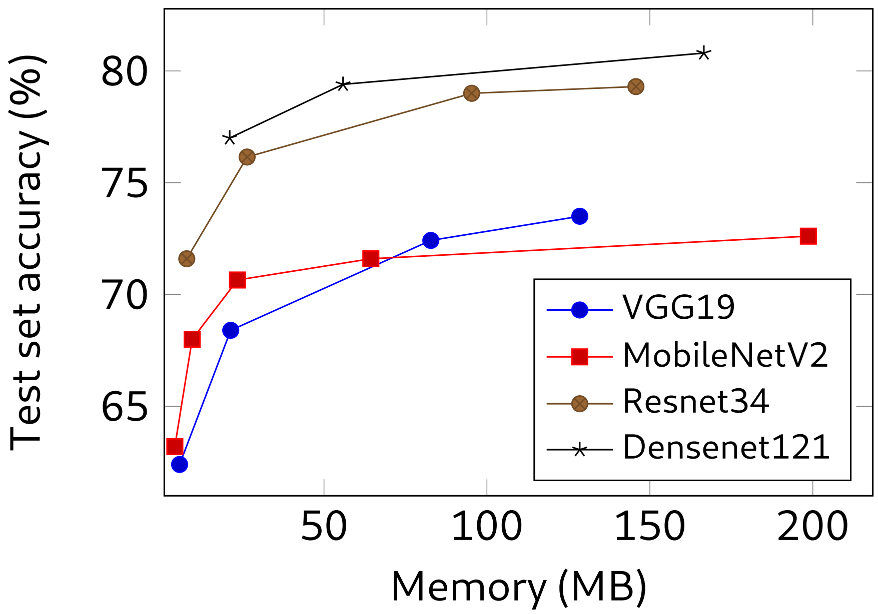
Efficient Inference



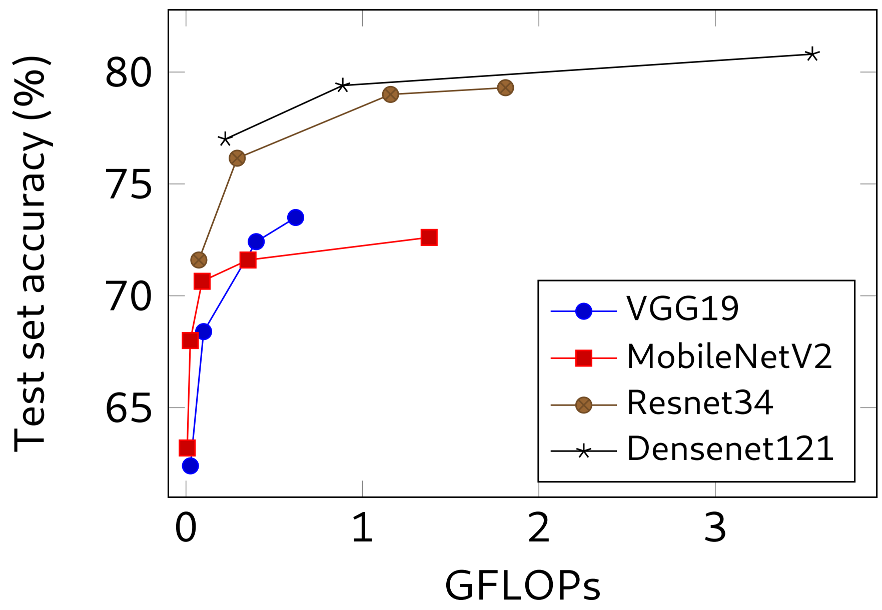
Memory vs accuracy.



FLOPs vs accuracy.



Memory vs accuracy.



FLOPs vs accuracy.

Scaling down proportionally the number of feature maps of each layer.

Layer 1

0.478	0.314	0.231	1.231	−0.423
−1.987	1.332	0.977	−0.541	1.230
0.322	0.431	0.221	0.112	−0.445
−0.718	0.891	−0.231	−1.231	−0.331
−1.412	0.490	0.791	0.901	−1.002

$\longrightarrow$

Layer 2

0.528	0.710	0.730	0.231	−1.423
−1.087	0.132	1.797	−1.041	1.131
1.220	0.321	0.341	1.912	−1.445
−0.798	1.291	1.481	−0.871	−0.821
1.772	−0.484	0.179	−0.121	−1.921

Baseline

Layer 1

0.478	0.314	0.231	1.231	−0.423
−1.987	1.332	0.977	−0.541	1.230
0.322	0.431	0.221	0.112	−0.445
−0.718	0.891	−0.231	−1.231	−0.331
−1.412	0.490	0.791	0.901	−1.002

→

Layer 2

0.528	0.710	0.730	0.231	−1.423
−1.087	0.132	1.797	−1.041	1.131
1.220	0.321	0.341	1.912	−1.445
−0.798	1.291	1.481	−0.871	−0.821
1.772	−0.484	0.179	−0.121	−1.921

Baseline

Layer 1

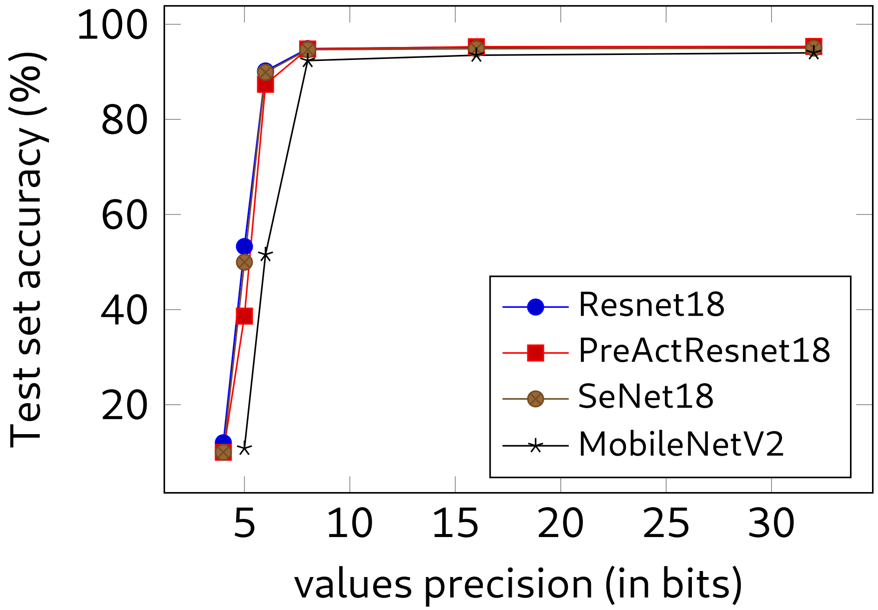
0.5	0.3	0.2	1.2	−0.4
−2.0	1.3	1.0	−0.5	1.2
0.3	0.4	0.2	0.1	−0.4
−0.7	0.9	−0.2	−1.2	−0.3
−1.4	0.5	0.8	0.9	−1.0

$\longrightarrow$

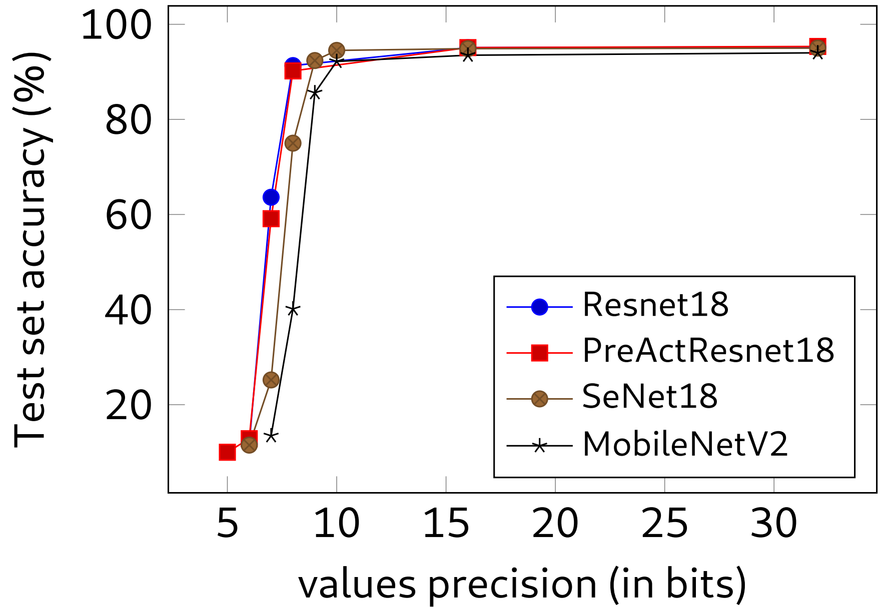
Layer 2

0.5	0.7	0.7	0.2	−1.4
−1.1	0.1	1.8	−1.0	1.1
1.2	0.3	0.3	1.9	−1.4
−0.8	1.3	1.5	−0.9	−0.8
1.8	−0.5	0.2	−0.1	−1.9

Quantization



Weights only (CIFAR10).



Weights and activations (CIFAR10).

- BC straight through principle:

Binary Connect (BC)

- BC straight through principle:
  1. Map weight values to their signs (1 or -1).

Binary Connect (BC)

- BC straight through principle:
  1. Map weight values to their signs (1 or -1).
  2. Compute feed forward and back propagation.

Binary Connect (BC)

- BC straight through principle:
  1. Map weight values to their signs (1 or -1).
  2. Compute feed forward and back propagation.
  3. Update initial floating point values.

Binary Connect (BC)

- BC straight through principle:
  1. Map weight values to their signs (1 or -1).
  2. Compute feed forward and back propagation.
  3. Update initial floating point values.
- Binary Weight Network (BWN) outperforms BC by adding a scaling factor ($-\alpha$ or α).

Binary Connect (BC)

- BC straight through principle:
  1. Map weight values to their signs (1 or -1).
  2. Compute feed forward and back propagation.
  3. Update initial floating point values.
- Binary Weight Network (BWN) outperforms BC by adding a scaling factor ($-\alpha$ or α).

Comparison of accuracy between baseline, BC and BWN on CIFAR10.

	Resnet34	Densenet121	MobilenetV2
Full-precision	95.0%	95.0%	93.8%
BC	93.6%	94.5%	93.0%
BWN	94.3%	94.7%	93.4%

Compression Methods

Layer 1

w_1^1	w_2^1	w_3^1	w_4^1	w_5^1
w_6^1	w_7^1	w_8^1	w_9^1	w_{10}^1
w_{11}^1	w_{12}^1	w_{13}^1	w_{14}^1	w_{15}^1
w_{16}^1	w_{17}^1	w_{18}^1	w_{19}^1	w_{20}^1
w_{21}^1	w_{22}^1	w_{23}^1	w_{24}^1	w_{25}^1

$\longrightarrow$

Layer 2

w_1^2	w_2^2	w_3^2	w_4^2	w_5^2
w_6^2	w_7^2	w_8^2	w_9^2	w_{10}^2
w_{11}^2	w_{12}^2	w_{13}^2	w_{14}^2	w_{15}^2
w_{16}^2	w_{17}^2	w_{18}^2	w_{19}^2	w_{20}^2
w_{21}^2	w_{22}^2	w_{23}^2	w_{24}^2	w_{25}^2

Baseline

Layer 1

0	w^1_2	0	w^1_4	w^1_5
0	0	w^1_8	w^1_9	w^1_{10}
w^1_{11}	0	w^1_{13}	w^1_{14}	0
w^1_{16}	w^1_{17}	0	w^1_{19}	0
w^1_{21}	w^1_{22}	0	w^1_{24}	w^1_{25}

Non structured

$\longrightarrow$

Layer 2

0	0	0	0	0
w^2_6	w^2_7	w^2_8	w^2_9	w^2_{10}
w^2_{11}	w^2_{12}	w^2_{13}	w^2_{14}	w^2_{15}
0	0	0	0	0
w^2_{21}	w^2_{22}	w^2_{23}	w^2_{24}	w^2_{25}

Structured

Pruning

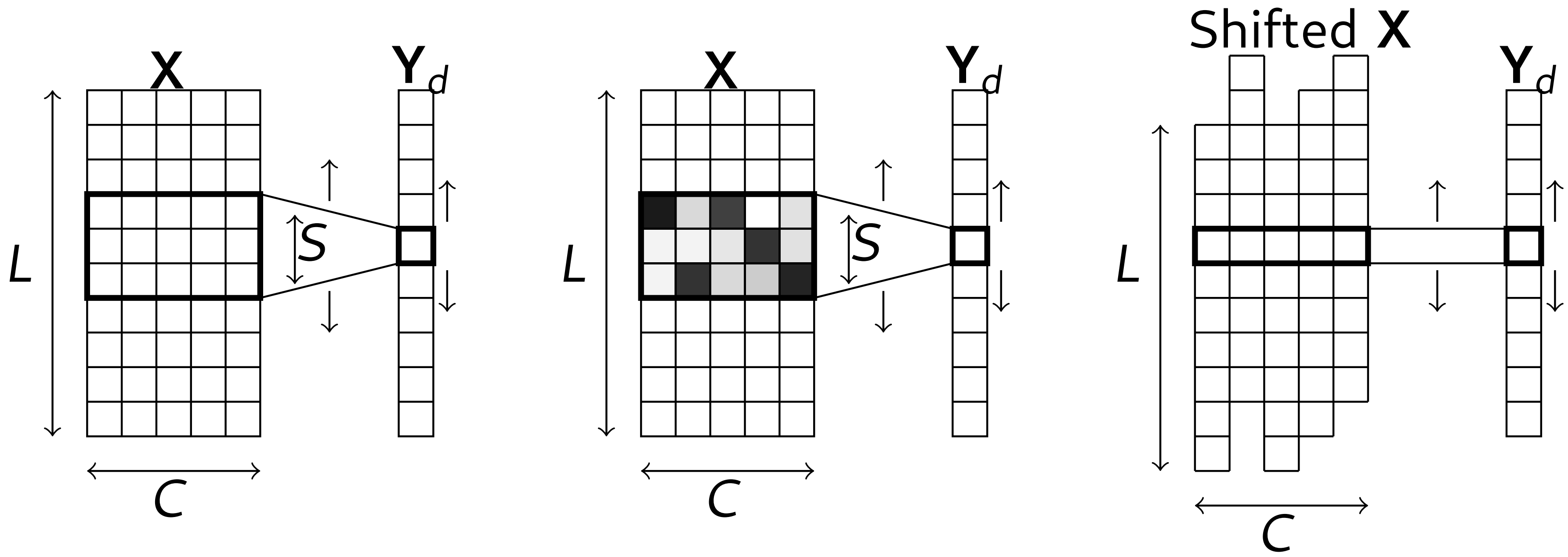
Pruning

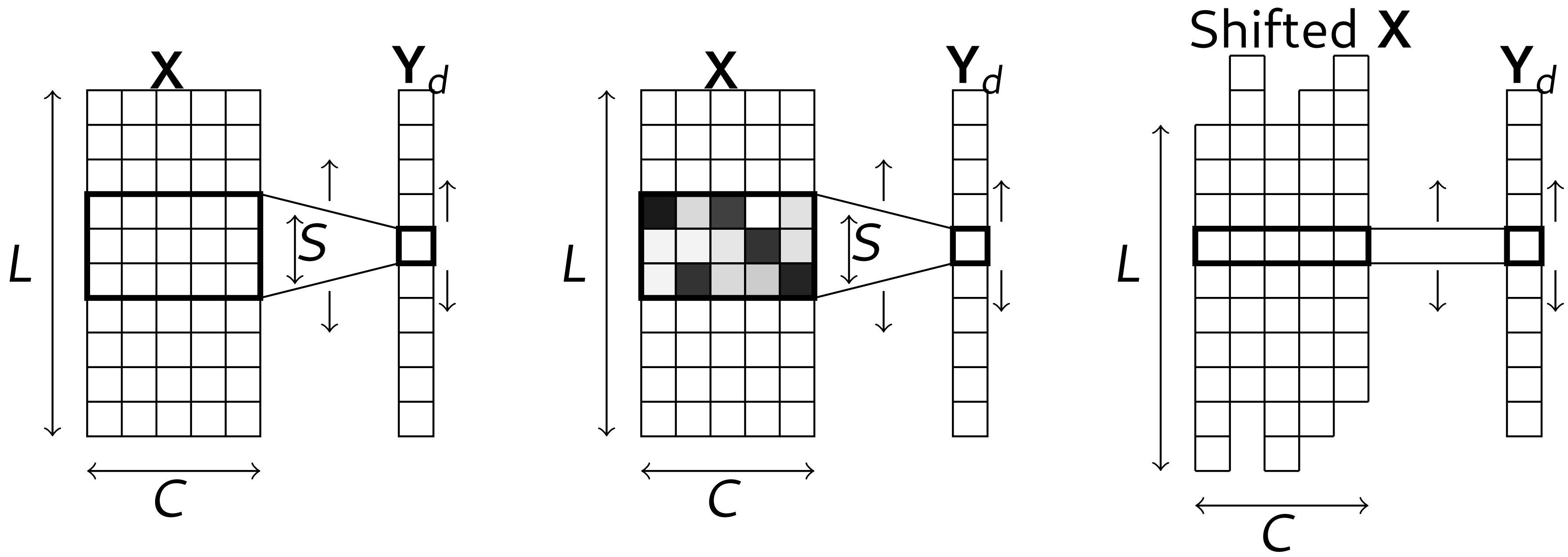
- Evaluate the importance of neurons and eliminate the least important ones to reduce neural network size.

- Evaluate the importance of neurons and eliminate the least important ones to reduce neural network size.
- Non structured pruning: eliminate neurons independently, only exploitable for very large levels of sparsity.

Pruning

- Evaluate the importance of neurons and eliminate the least important ones to reduce neural network size.
- Non structured pruning: eliminate neurons independently, only exploitable for very large levels of sparsity.
- Structured pruning: eliminate kernels, filters or even layers, exploitable for even low levels of sparsity.


X
$\mathbf{Y}_d$
L
S
C
X
$\mathbf{Y}_d$
L
S
C
Shifted X
$\mathbf{Y}_d$
L
C



Shift Attention Layer (SAL)

- Simplified operations,
- Reduced number of parameters,
- Fully exploitable technique.

Comparison of accuracy, number of parameters and number of floating point operations (FLOPs) using ResNet architectures.

Method	CIFAR10			CIFAR100		
	Accuracy	NP (M)	MFLOPs	Accuracy	NP (M)	MFLOPs
Pruned-B	93.06%	0.73	91	73.6%	7.83	616
NISP	93.01%	0.49	71	–	–	–
PCAS	93.58%	0.39	56	73.84%	4.02	475
SAL	**93.6%**	**0.36**	**42**	**77.6%**	**3.9**	**251**

Quantized Shift Layers

Shift Layers + BC or BWN

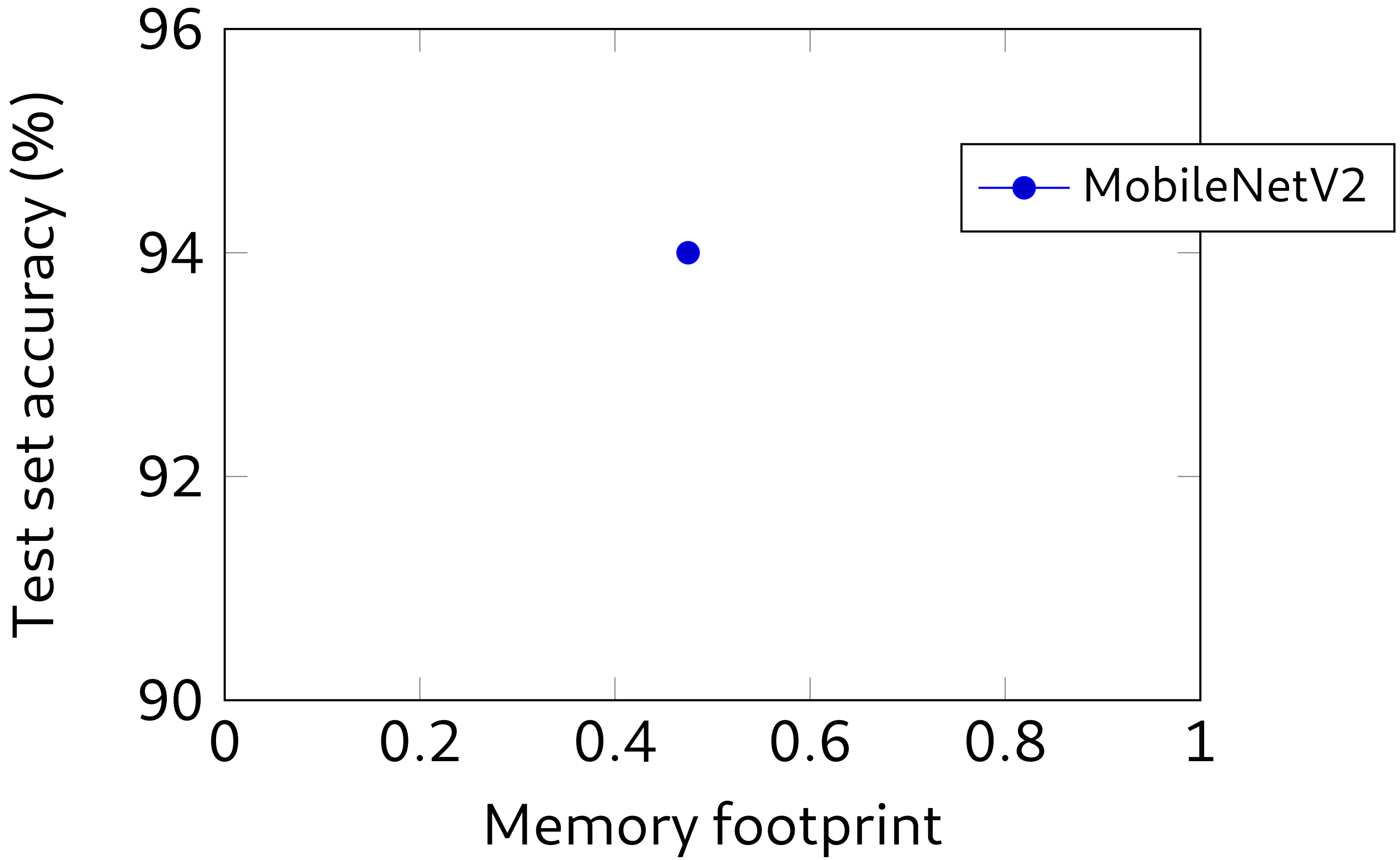
- **Shift layer:** replace a convolution by a multiplication.
- **BC/BWN**: replace a multiplication by a low-cost multiplexer.
- **Shift layer +BC/BWN**: replace a convolution by a low-cost multiplexer.

Shift Layers + BC or BWN

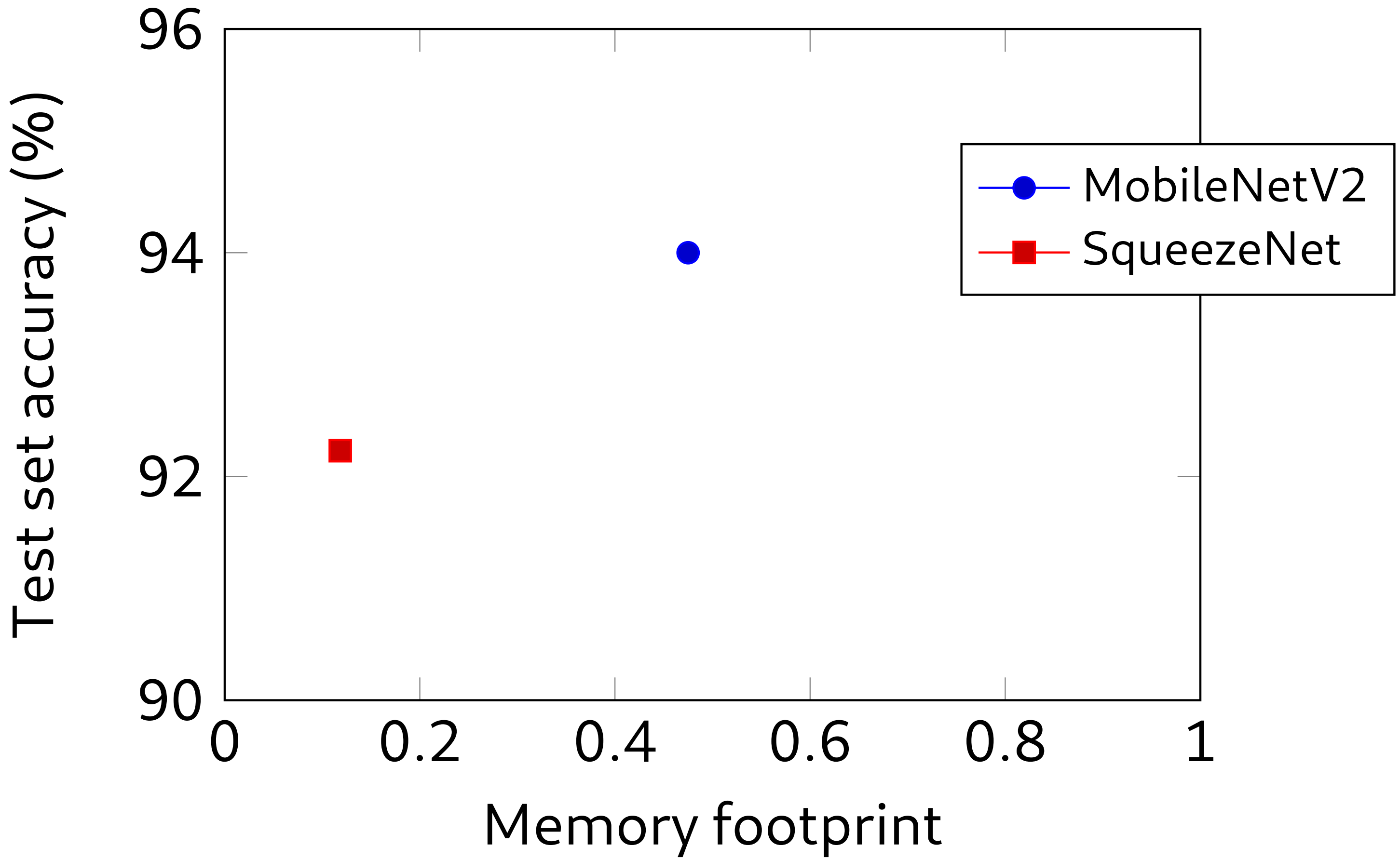
- **Shift layer**: replace a convolution by a multiplication.
- **BC/BWN**: replace a multiplication by a low-cost multiplexer.
- **Shift layer +BC/BWN**: replace a convolution by a low-cost multiplexer.

Comparison of accuracy and memory usage between Resnet-20 baseline, SAL, SAL with BC and SAL with BWN on CIFAR10.

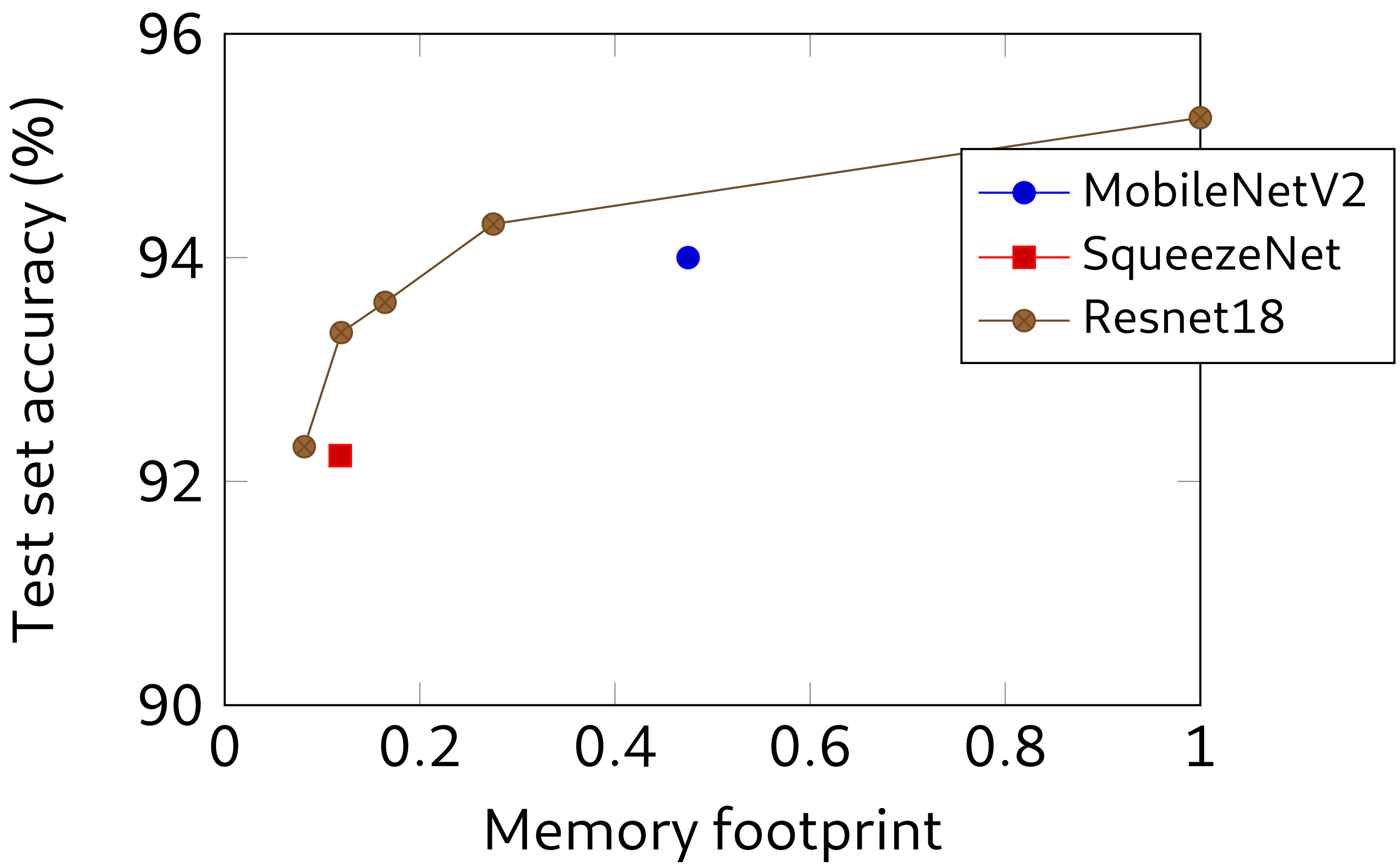
	Accuracy(%)	Memory(Mb)
Baseline	94.66	39.04
SAL	95.52	31.36
SAL + BC	93.20	6.87
SAL + BWN	94.00	6.87



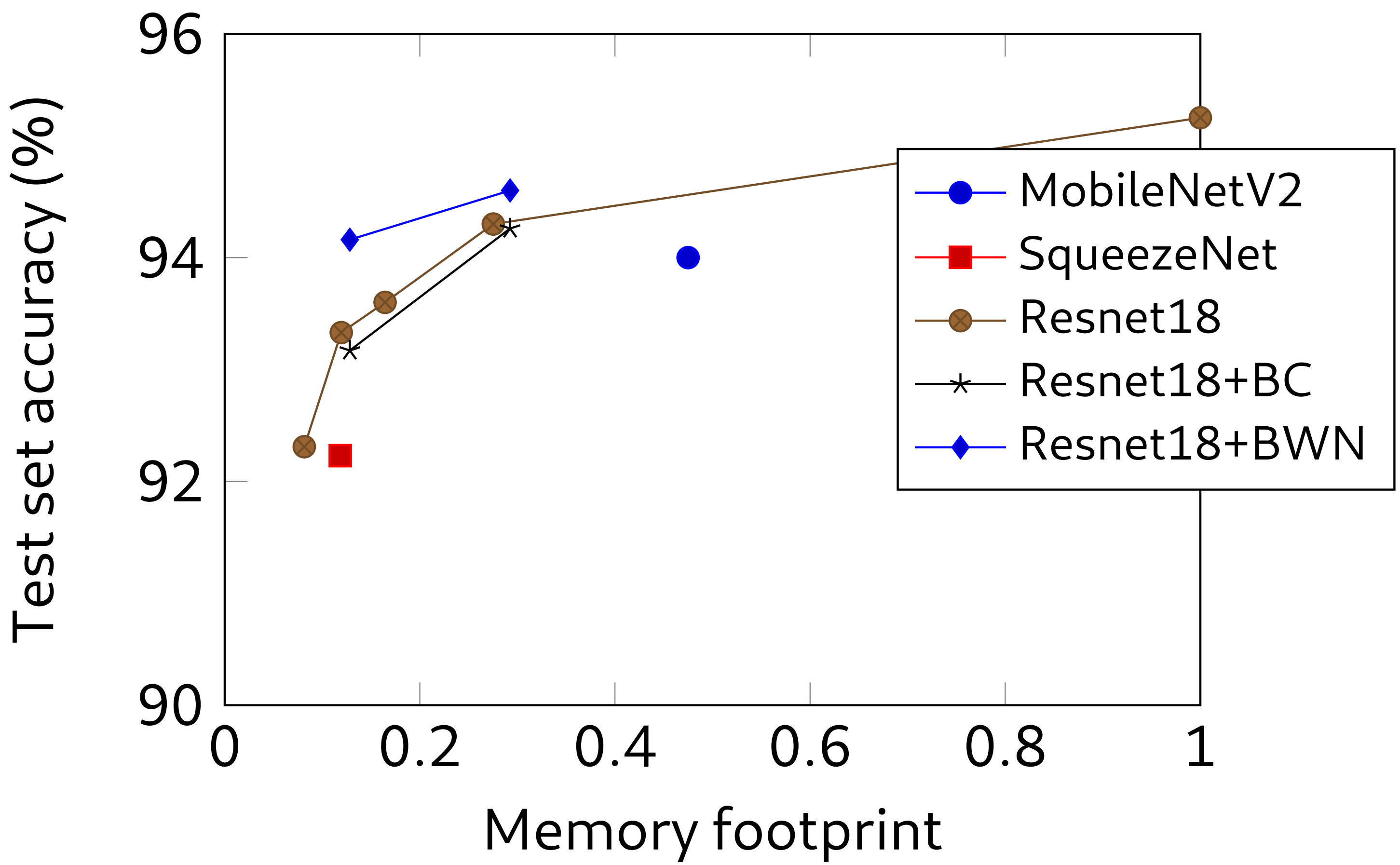
Evolution of accuracy when applying compression methods on different DNN architectures for the CIFAR10 dataset.



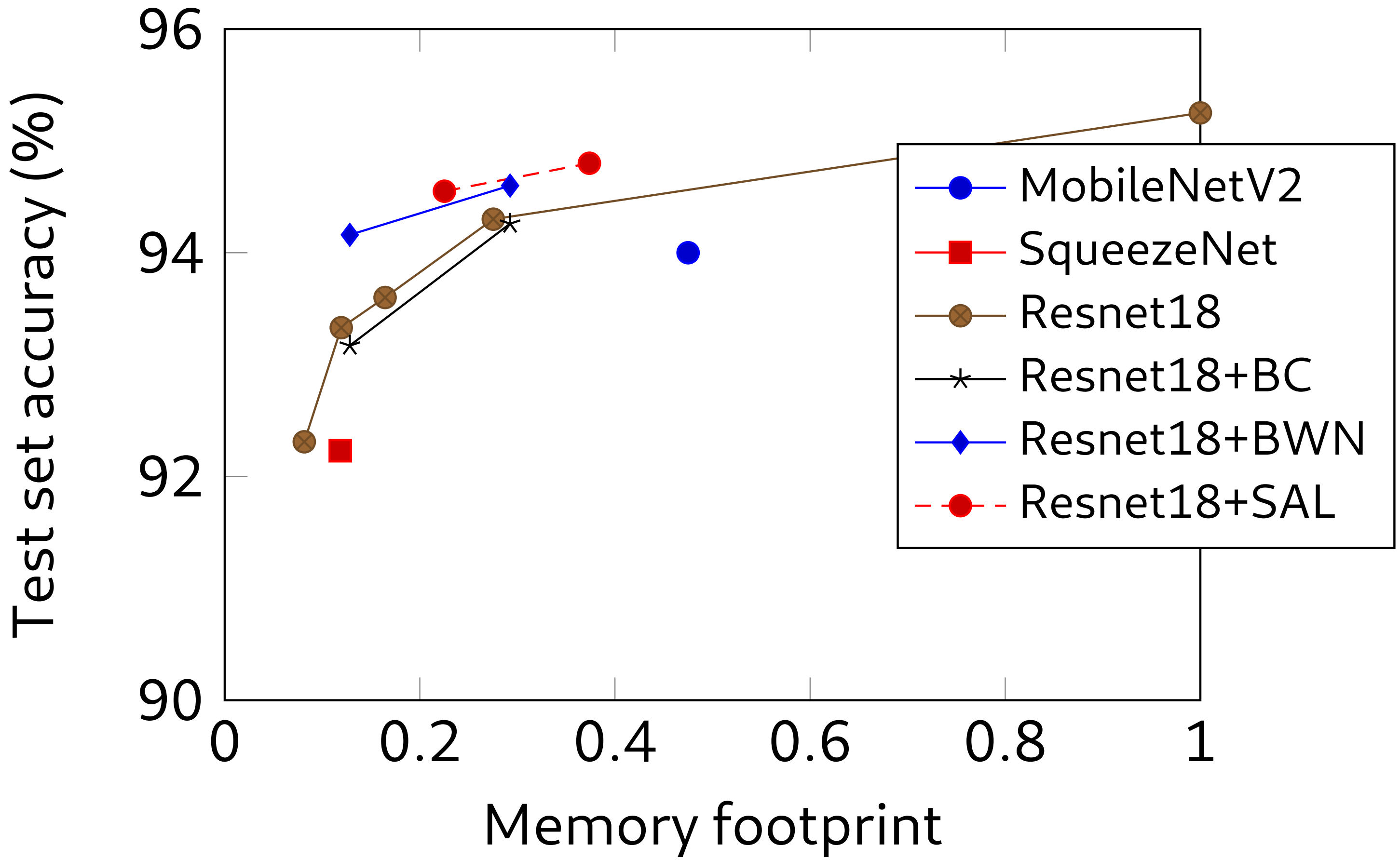
Evolution of accuracy when applying compression methods on different DNN architectures for the CIFAR10 dataset.



Evolution of accuracy when applying compression methods on different DNN architectures for the CIFAR10 dataset.



Evolution of accuracy when applying compression methods on different DNN architectures for the CIFAR10 dataset.



Evolution of accuracy when applying compression methods on different DNN architectures for the CIFAR10 dataset.

Conclusion

Compression Methods

- Different ways to reduce DNNs size, complexity and thus energy consumption.
- Compression methods are only applicable to the inference part, and not the learning part.

Compression Methods

- Different ways to reduce DNNs size, complexity and thus energy consumption.
- Compression methods are only applicable to the inference part, and not the learning part.

Directions

- Which combinations of quantization methods are efficient?
- Can training process decide the most efficient number of bits to quantize values?
- Can SAL perform well in other domains than classification?
- Can compression methods be reconsidered to reduce training complexity?