EMBEDDED MACHINE LEARNING LECTURE 11 - ADVANCED NEURAL ARCHITECTURES

Holger Fröning
holger.froening@ziti.uni-heidelberg.de
HAWAII Group (formerly Computing Systems), Institute of Computer Engineering
Heidelberg University

WHY ALWAYS IMAGE CLASSIFICATION?

Image classification is a task that is very handy when it comes to feature extraction

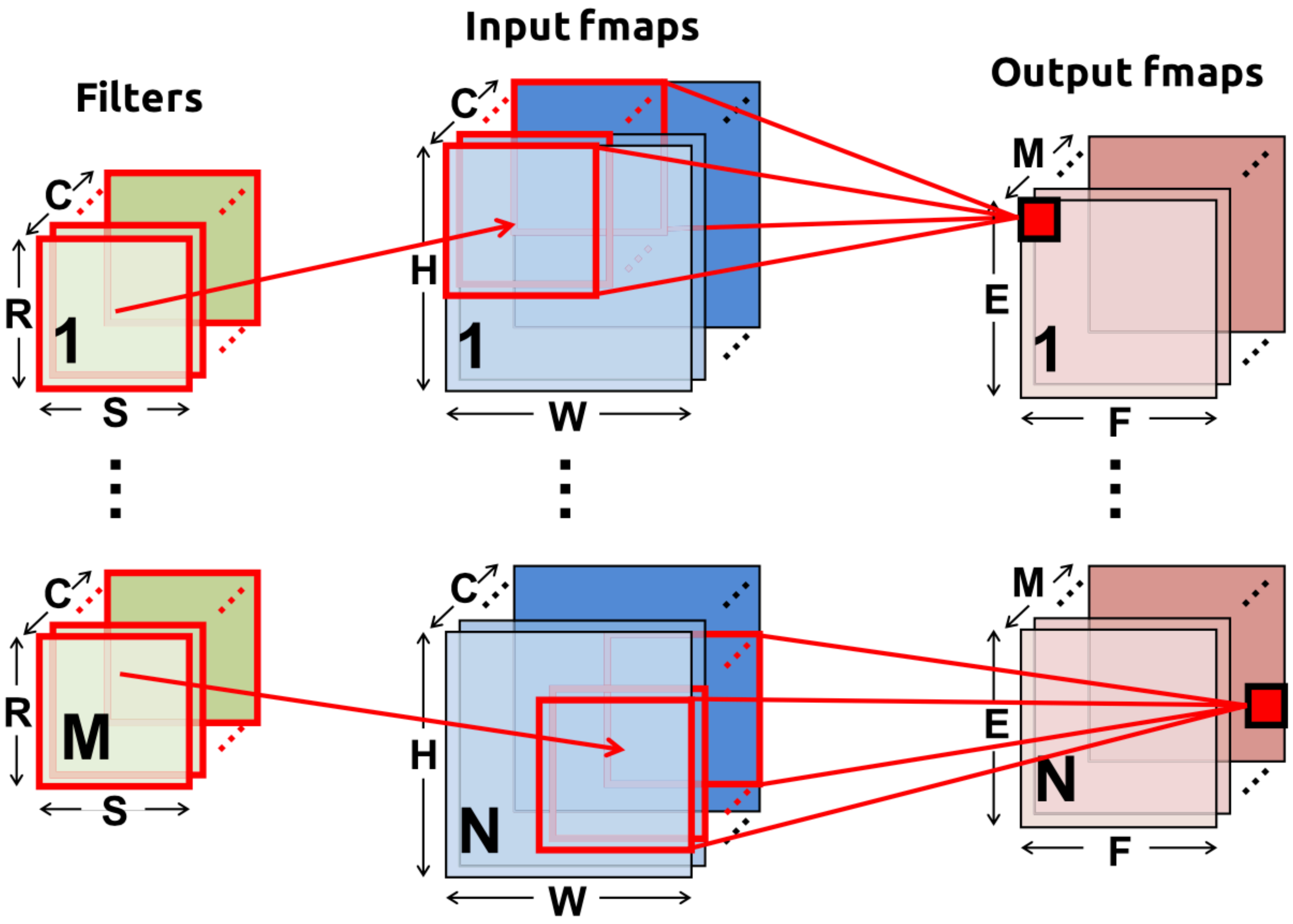
Feature extraction is foundational for many other tasks

- Object detection & object tracking
- Image segmentation - semantic (cannot distinguish between different instances of the same category) or instance (can distinguish)
- Pose estimation
- Image captioning

Transfer learning: knowledge from one task is re-used for a different task

1. Pre-train your model on image classification tasks
2. Use it as a feature extractor for more complex tasks
3. Transfer learning then uses this model as base and adds more layers for more complex tasks; new layers will be fine tuned based on task-specific data

RECAP: CONVOLUTION



RECAP: CONVOLUTION

$$\mathbf{O}[z][u][x][y] = \sum_{k=0}^{C-1} \sum_{i=0}^{S-1} \sum_{j=0}^{R-1} \mathbf{I}[z][k][Ux+i][Uy+j] \cdot \mathbf{W}[u][k][i][j] + \mathbf{B}[u]$$

ofmap $\mathbf{O}$, ifmap $\mathbf{I}$, filters (weights) $\mathbf{W}$, and biases $\mathbf{B}$

ofmap = output filter map (output activations)

ifmap = input filter map (input activations)

$$E = (H - R + U)/U$$
$$F = (W - S + U)/U$$

N	Batch size (3D fmaps)	0 <= z <= N
M	number of 3D filters / number of ofmaps	0 <= u <= M
C	number of ifmap/filter channels	k
H / W	ifmap plane height/width	x resp y
R / S	filter plane height/width	x resp y
E / F	ofmap plane height/width	0 <= x <= F, 0 <= y <= E
U	stride	

V. Sze, T.-J. Yang, Y.-H. Chen, J. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," Proceedings of the IEEE, vol. 105, no. 12, pp. 2295-2329, 2017.

RESOURCE-EFFICIENT MODEL ARCHITECTURES

Resource efficiency by model design

SQUEEZENET (2016)

Convolutional weights: $W_c = RSCM$

Three pillars

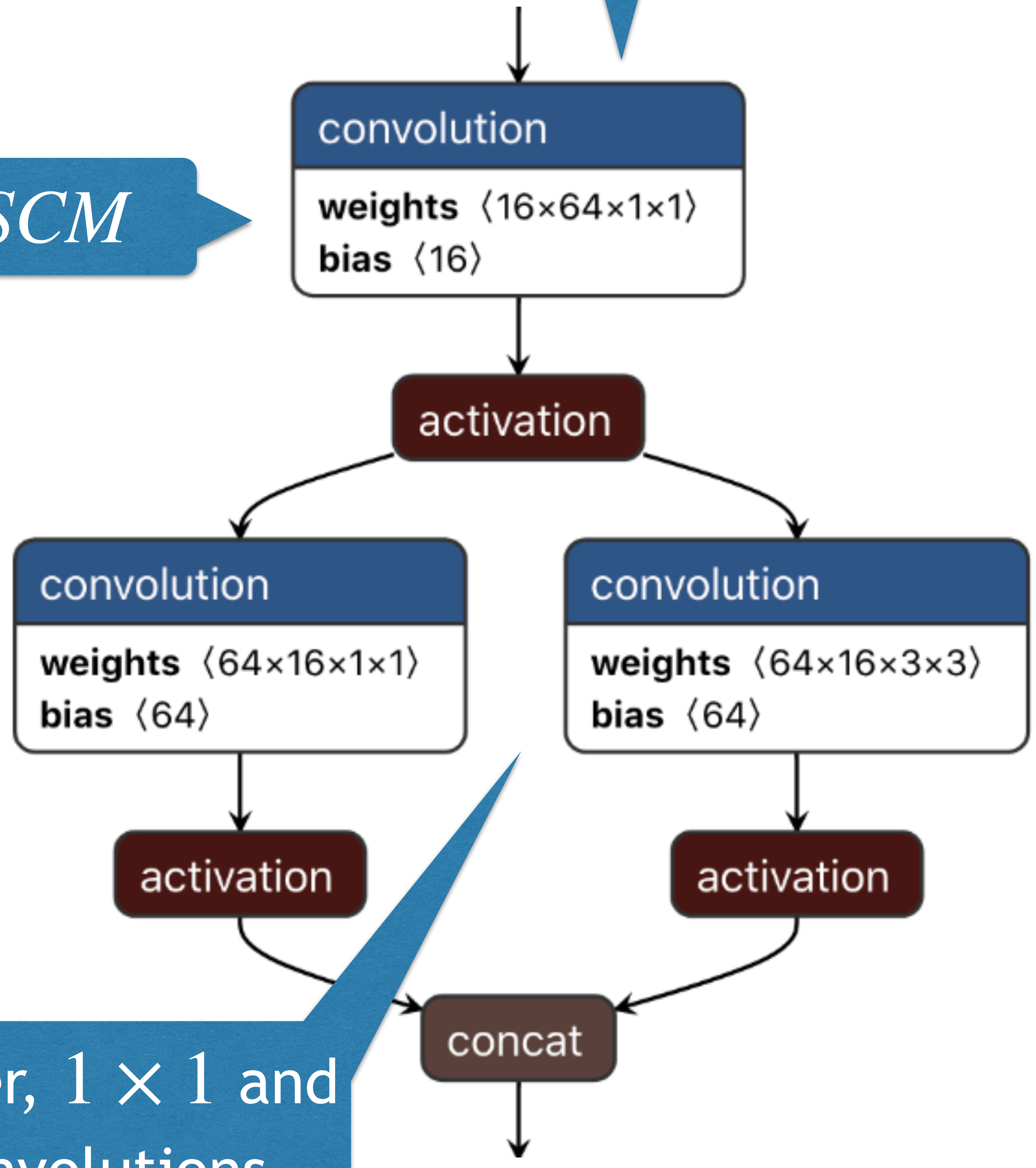
1. Replace 3x3 filters with 1x1 filters to reduce number of parameters (reduces R and S)
2. Decrease the number of input channels to 3x3 filters (reduces C)
3. Downsample late to maintain large activation maps (maintain accuracy)

Use 1. and 2. to implement a “fire module”

Example: C=64 -> 16 -> 64

Squeeze layer, only 1×1 convolutions

$W_c = RSCM$



Expand layer, 1×1 and 3×3 convolutions

SQUEEZENET ARCHITECTURE

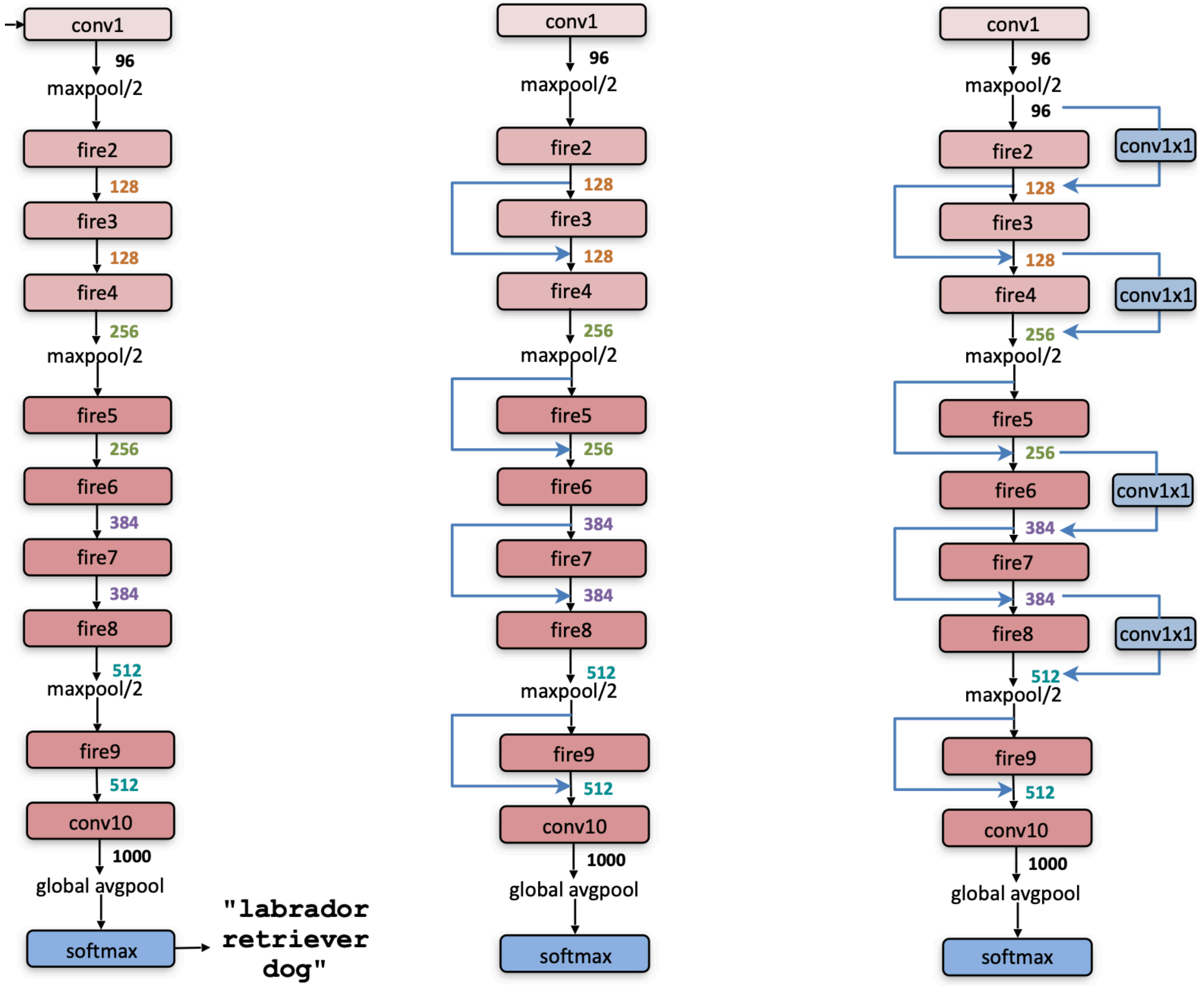
Different variants

- No bypass, simple bypass, complex bypass

No FC layers

Max pooling (stride 2) relatively late

- "Pooling layers reduce the size of the output map"



Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, https://arxiv.org/abs/1602.07360

SQUEEZENET RESULTS

CNN architecture	Compression Approach	Data Type	Original $\rightarrow$ Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB $\rightarrow$ 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB $\rightarrow$ 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB $\rightarrow$ 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB $\rightarrow$ 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB $\rightarrow$ 0.47MB	**510x**	57.5%	80.3%

AlexNet accuracy with 50x less parameters

~500x less parameters in combination with model compression

DeepCompression: pruning and quantization, but not inline with general-purpose processors

However: AlexNet performance is not state-of-the-art, SqueezeNet mainly important for its simplicity

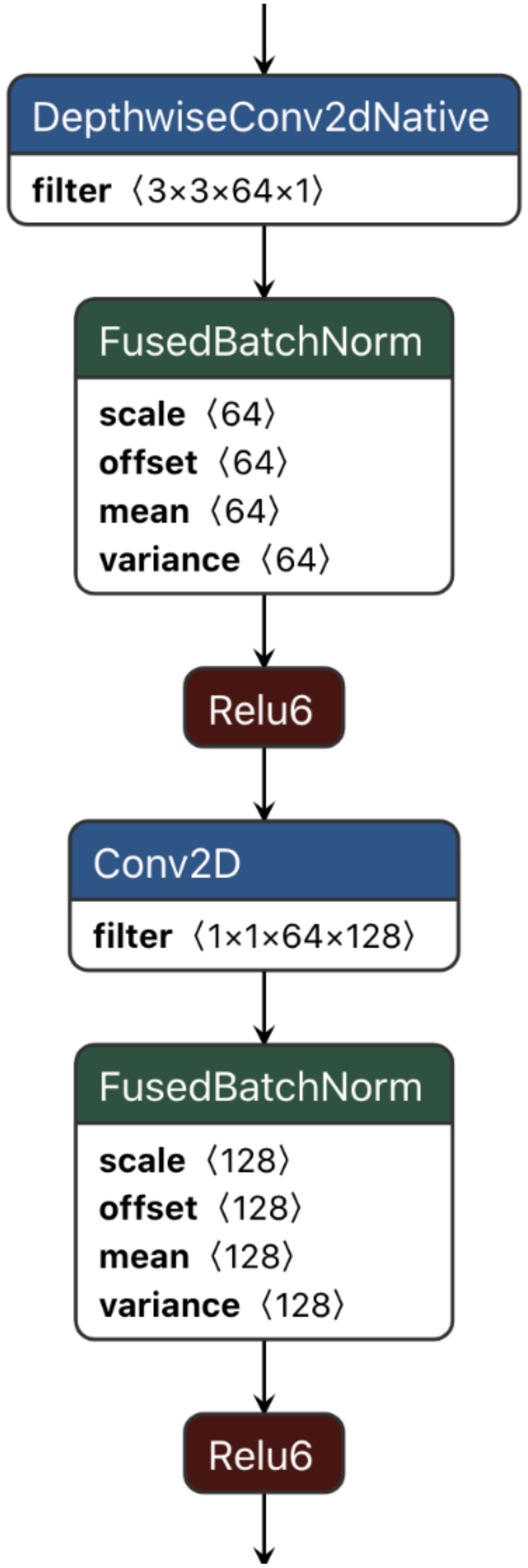
Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, https://arxiv.org/abs/1602.07360

MOBILENET V1 (2017)

Core idea: replace expensive convolutions with cheaper alternatives

- Depthwise separable convolution
- Less trainable parameters
- Less MACs

Separable convolutions?



Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, https://arxiv.org/abs/1704.04861

(SPATIALLY) SEPARABLE CONVOLUTIONS

An matrix $K^{(M \times N)}$ is separable if it can be decomposed into $(M \times 1)$ and $(1 \times N)$ vectors

$$\begin{bmatrix} ax & ay & az \\ bx & by & bz \\ cx & cy & cz \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \cdot \begin{bmatrix} x & y & z \end{bmatrix}, \text{ e.g., } \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

A convolution is then

$$o = \mathbf{I} * \mathbf{K} = \begin{bmatrix} i_{1,1} & i_{1,2} & i_{1,3} \\ i_{2,1} & i_{2,2} & i_{2,3} \\ i_{3,1} & i_{3,2} & i_{3,3} \end{bmatrix} * \begin{bmatrix} ax & ay & az \\ bx & by & bz \\ cx & cy & cz \end{bmatrix} = \begin{bmatrix} i_{1,1} & i_{1,2} & i_{1,3} \\ i_{2,1} & i_{2,2} & i_{2,3} \\ i_{3,1} & i_{3,2} & i_{3,3} \end{bmatrix} * \left(\begin{bmatrix} a \\ b \\ c \end{bmatrix} \cdot \begin{bmatrix} x & y & z \end{bmatrix} \right)$$

$$= \left(\begin{bmatrix} i_{1,1} & i_{1,2} & i_{1,3} \\ i_{2,1} & i_{2,2} & i_{2,3} \\ i_{3,1} & i_{3,2} & i_{3,3} \end{bmatrix} * \begin{bmatrix} a \\ b \\ c \end{bmatrix} \right) * \begin{bmatrix} x & y & z \end{bmatrix}$$

$*$ is a convolution operator, not a multiplication

Reduces computational complexity from MN down to $M + N$ multiplications

However, consider increase in space complexity - why?

More intermediate results

(SPATIALLY) SEPARABLE CONVOLUTIONS - EXAMPLE

$$\mathbf{I} * \mathbf{K} = \begin{bmatrix} 50 & 60 & 70 \\ 80 & 90 & 100 \\ 110 & 120 & 130 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 50 & 60 & 70 \\ 80 & 90 & 100 \\ 110 & 120 & 130 \end{bmatrix} * \left(\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \right)$$

$$= \left(\begin{bmatrix} 50 & 60 & 70 \\ 80 & 90 & 100 \\ 110 & 120 & 130 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \right) * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 50 \cdot 1 + 80 \cdot 2 + 110 \cdot 1 & 60 \cdot 1 + 90 \cdot 2 + 120 \cdot 1 & 70 \cdot 1 + 100 \cdot 2 + 130 \cdot 1 \end{bmatrix}$$
$$* \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 320 & 360 & 400 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 320 \cdot 1 + 360 \cdot 2 + 400 \cdot 1 \end{bmatrix} = 1440$$

$*$ is a convolution operator, not a multiplication

DEPTHWISE SEPARABLE CONVOLUTIONS

Spatially separable convolutions are not used often in ML as kernel search is limited

Instead, consider M $R \times S \times C$ convolutions

- E.g., transform a $7 \times 7 \times 3$ input shape into a $5 \times 5 \times 128$ output shape
- Using 128 kernels of size $3 \times 3 \times 3$

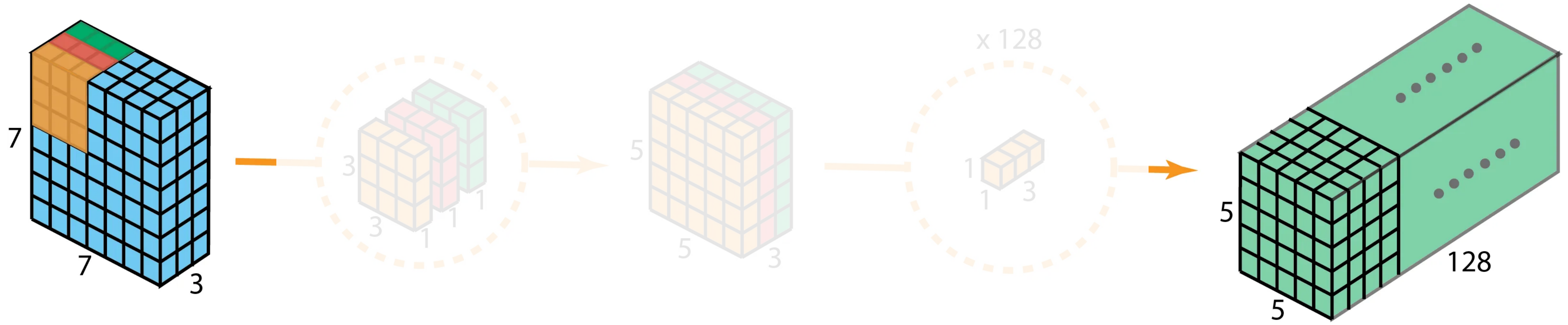
Step 1: replace a $R \times S \times C$ convolution with C $R \times S \times 1$ (separable) convolutions

- $3 \times 3 \times 3$ -> 3 kernels of size $3 \times 3 \times 1$
- Output shape is identical by stacking output fmaps

Step 2: as cross-channel correlation is lost, add afterwards M $1 \times 1 \times C$ convolutions

pointwise convolution

- One convolution then transforming the $7 \times 7 \times 3$ shape into a $7 \times 7 \times 1$ shape
- All M convolutions then result in a $7 \times 7 \times 128$ shape



https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

DEPTHWISE SEPARABLE CONVOLUTIONS - EFFICIENCY

Plain convolution: $\text{MAC}_c = (EF \cdot RSC) \cdot M$

For $E = (H - R + U)/U$ and $F = (W - S + U)/U$

For the following, assume a unit stride ($U = 1$) and zero padding

Then: $\text{MAC}_c = ((H - R + 1)(W - S + 1) \cdot RSC) \cdot M$

Step 1: replace M $R \times S \times C$ convolutions with C $R \times S \times 1$ convolutions

$\text{MAC}_{cds1} = ((H - R + 1)(W - S + 1) \cdot RS \cdot 1) \cdot C$

Step 2: followed by M $1 \times 1 \times C$ convolutions

$\text{MAC}_{cds1} = ((H - R + 1)(W - S + 1) \cdot 1 \cdot 1 \cdot C) \cdot M$

$\text{MAC}_{cds} = \text{MAC}_{cds1} + \text{MAC}_{cds2} = (H - R + 1)(W - S + 1) \cdot (RSC + CM)$

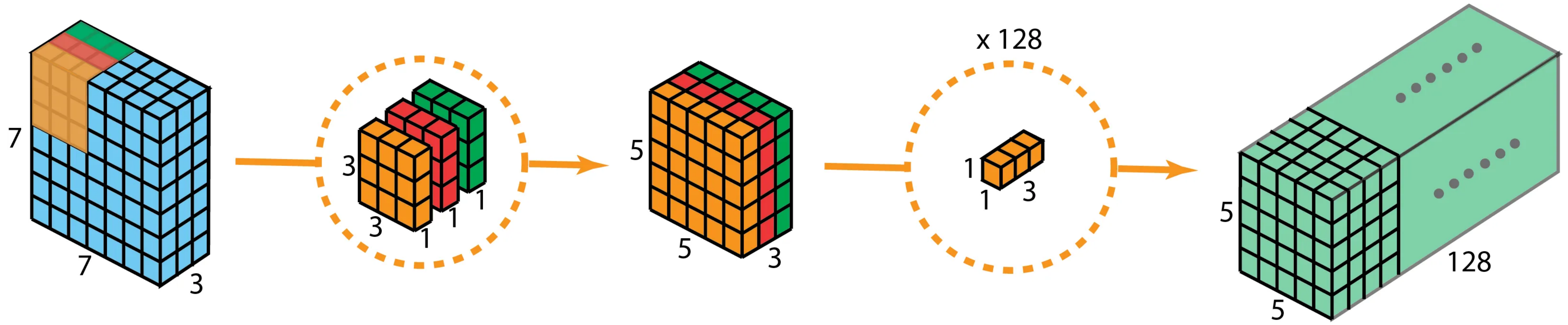
DEPTHWISE SEPARABLE CONVOLUTIONS - EFFICIENCY EXAMPLE

$$\text{MAC}_c = ((H - R + 1)(W - S + 1) \cdot RSC) \cdot M \qquad = 86400$$

$$\text{MAC}_{cds} = ((H - R + 1)(W - S + 1) \cdot (RSC + CM) \qquad = 10275$$

$$r = \frac{\text{MAC}_c}{\text{MAC}_{cds}} = \frac{RSM}{RSC + CM} = \frac{1}{M} + \frac{1}{RS}$$

Modern architectures: $M \gg R, S$ => $r \approx \dfrac{1}{RS}$



https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

MOBILENET V1 (2017)

Core idea: replace expensive convolutions with cheaper alternatives

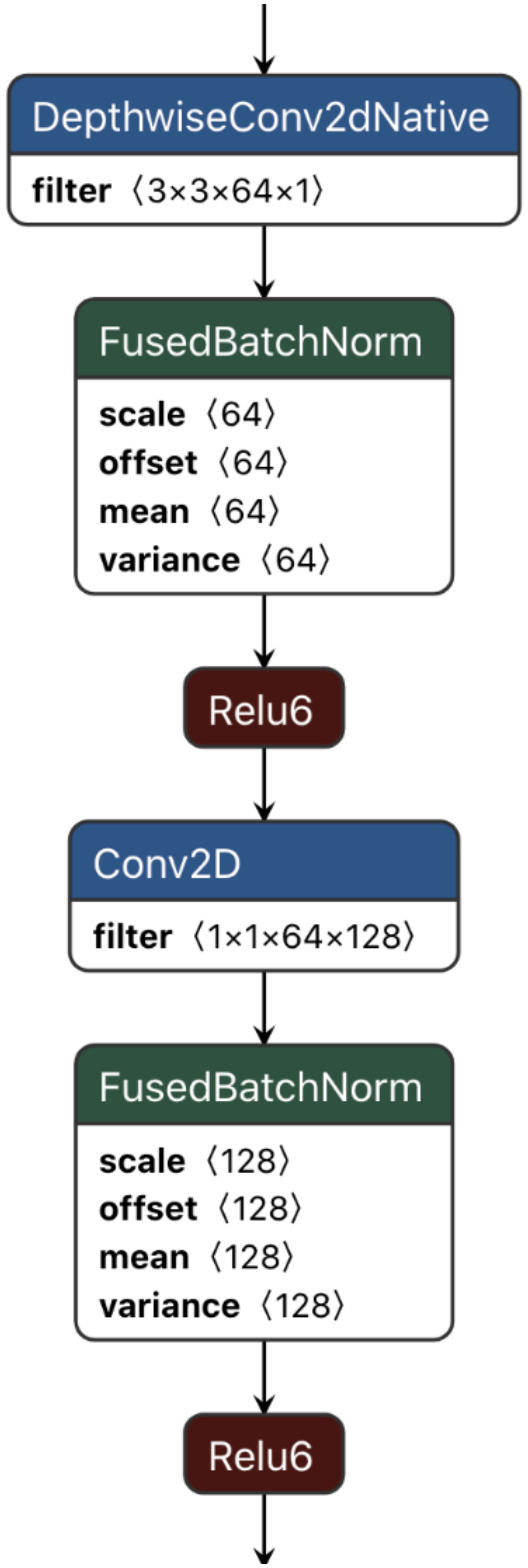
- Depthwise separable convolution
- Less trainable parameters
- Less MACs

Depthwise separable convolutions

- Save: MACs, save parameters
- Cost: parameters (less model capacity), activations

Width multiplier $\alpha \in (0,1]$ to control the input channel depth

- Thinner models based on typical settings of 1, 0.75, 0.5, and 0.25



Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, https://arxiv.org/abs/1704.04861

MOBILENET V1 (2017)

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

Table 6. MobileNet Width Multiplier

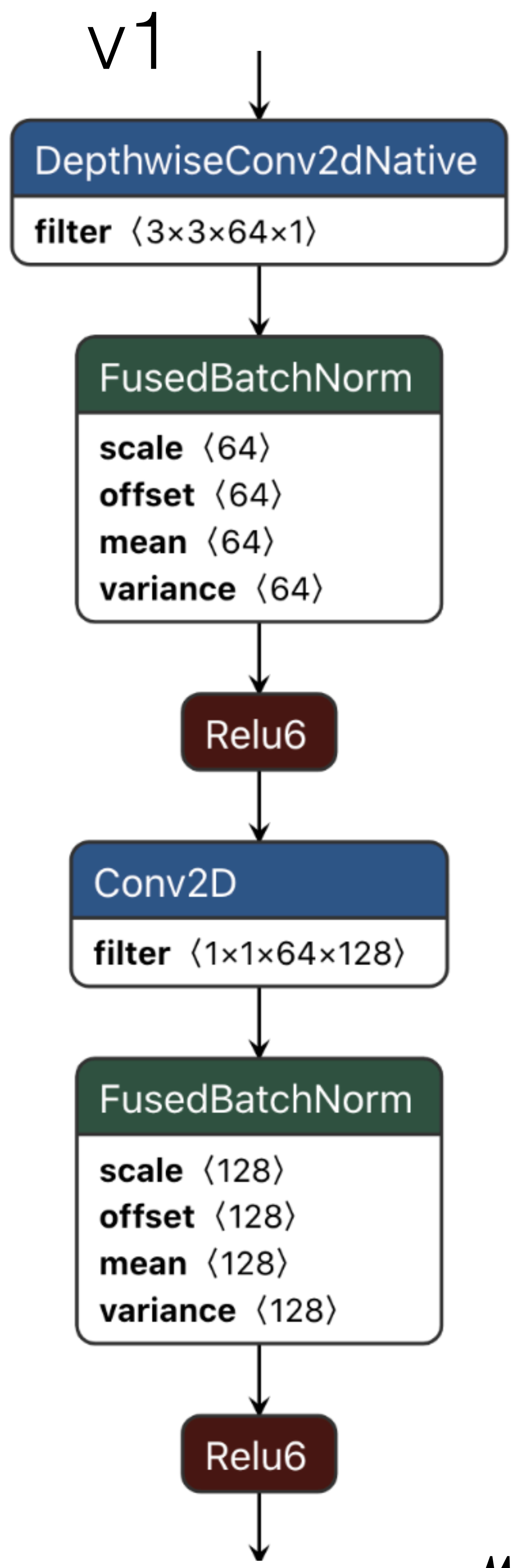
Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Outperforms various model alternatives based on accuracy, MACs, and parameter count

- GoogLeNet, VGG16, AlexNet, SqueezeNet ($\alpha = 0.5$), Inception v3 (“Stanford dogs” dataset)

MobileNet-SSD for object detection based on MS-COCO

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, https://arxiv.org/abs/1704.04861



MOBILENET V2 (2018)

Rearranged architecture based on building blocks

1. Expansion layer based on 1×1 convolution to increase channel depth
2. Depthwise convolutional layer
3. Bottleneck layer to reduce channel depth

Skip connection if input/output dimensions match

“Inverted residuals” as ResNet used skip connections based on many channels (reducing width inside a block)

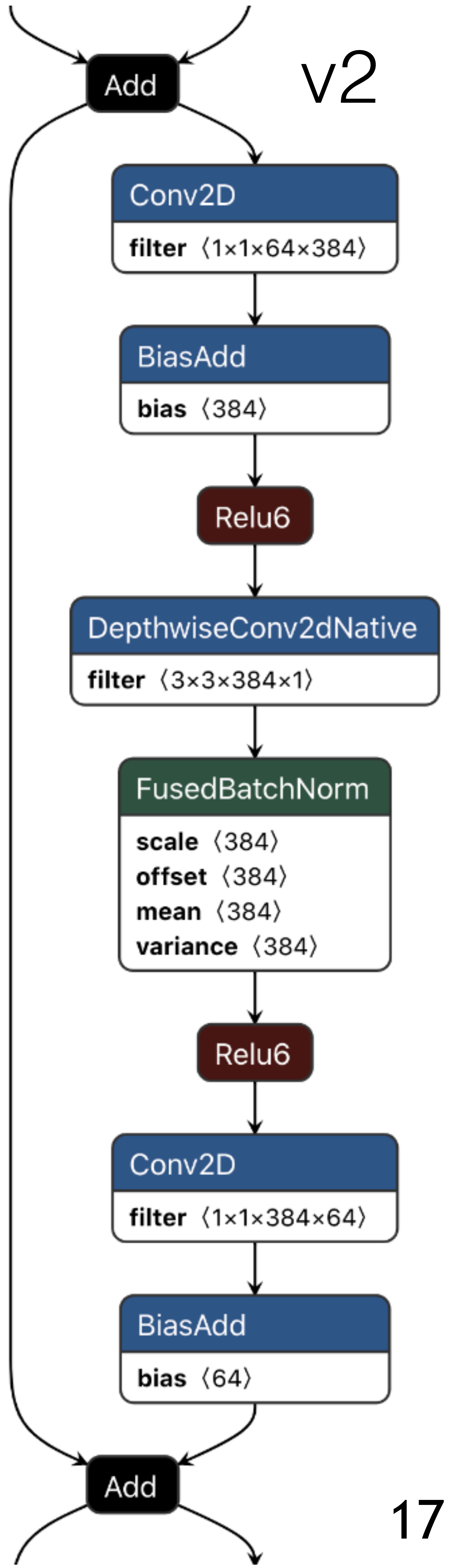
More memory efficient

No nonlinearity behind bottleneck layer

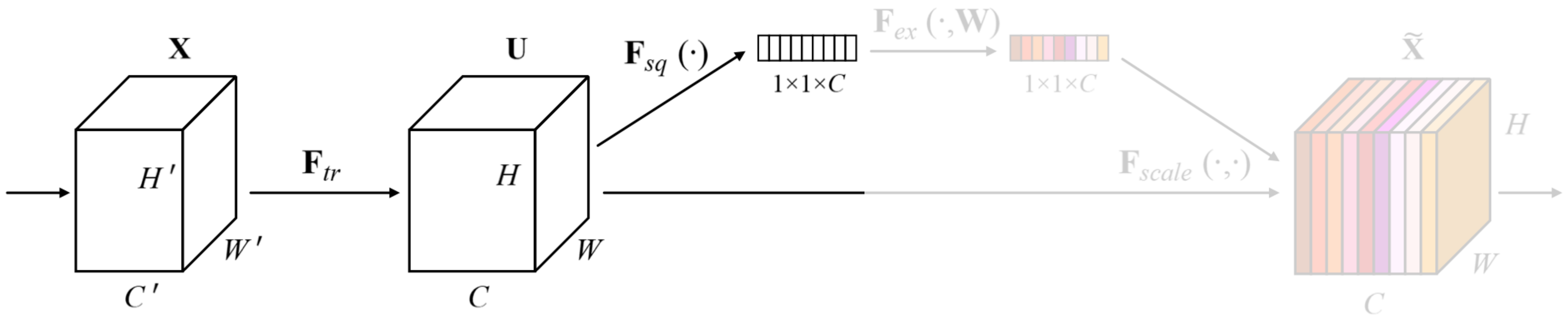
“Linear Bottlenecks”

Slightly less parameters than v1, same accuracy

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks, https://arxiv.org/abs/1801.04381



SQUEEZE-AND-EXCITATION NETWORKS (SENET)



Intuition: model shall learn where to attend. Here: which channel

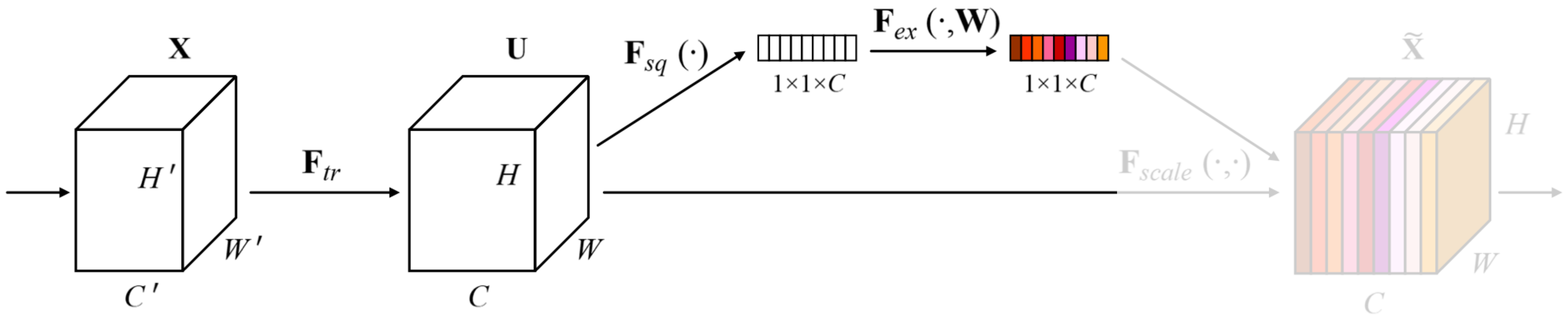
- Also: number of channels often becomes larger with an increasing depth of the network

Squeeze module $\mathbf{F}_{sq}$ to reduce information

- Feature descriptor that decomposes in information of each feature map into a single value
- Here by using global average pooling to transform an $H \times W \times C$ tensor into a $1 \times 1 \times C$ one
- Essentially a vector of length C that encodes the feature descriptor of each fmap

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu, Squeeze-and-Excitation Networks, https://arxiv.org/abs/1709.01507

SQUEEZE-AND-EXCITATION NETWORKS (SENET)



Excitation module $\mathbf{F}_{ex}$ for adaptive recalibration

Fully capture channel-wise dependencies by learning

Not mutually exclusive: multiple channels can be important

=> $\mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma\left(\mathbf{W}_1 \cdot \text{ReLU}(\mathbf{W}_2 \cdot \mathbf{z})\right)$

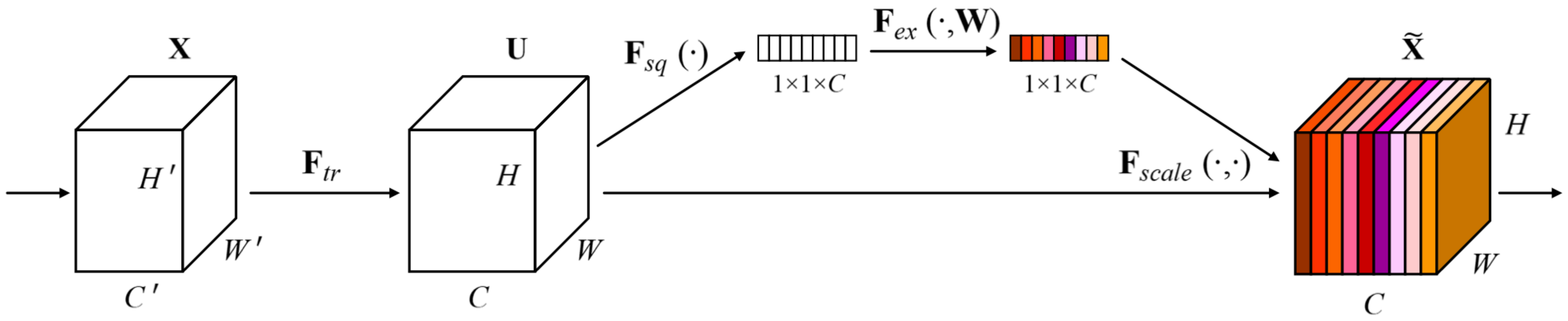
Dimensionality reduction layer ($\mathbf{W}_2$), followed by dimensionality-increasing layer ($\mathbf{W}_1$)

Shape is maintained, size of latent space can be controlled: C/r, for a reduction parameter r

Sigmoid activation σ scales the output values to a range $[0,1]$

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu, Squeeze-and-Excitation Networks, https://arxiv.org/abs/1709.01507

SQUEEZE-AND-EXCITATION NETWORKS (SENET)



Scale module $\mathbf{F}_{scale}$: channel-wise multiplication between the scalar output of $\mathbf{F}_{ex}$ and the corresponding fmap

- Simple, computationally cheap

Downside?

- Adds MACs and parameters
- Scale is computationally cheap but adds plenty of state

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu, Squeeze-and-Excitation Networks, https://arxiv.org/abs/1709.01507

MORE ARCHITECTURES

MNASNet (2018): use NAS to find efficient model architectures

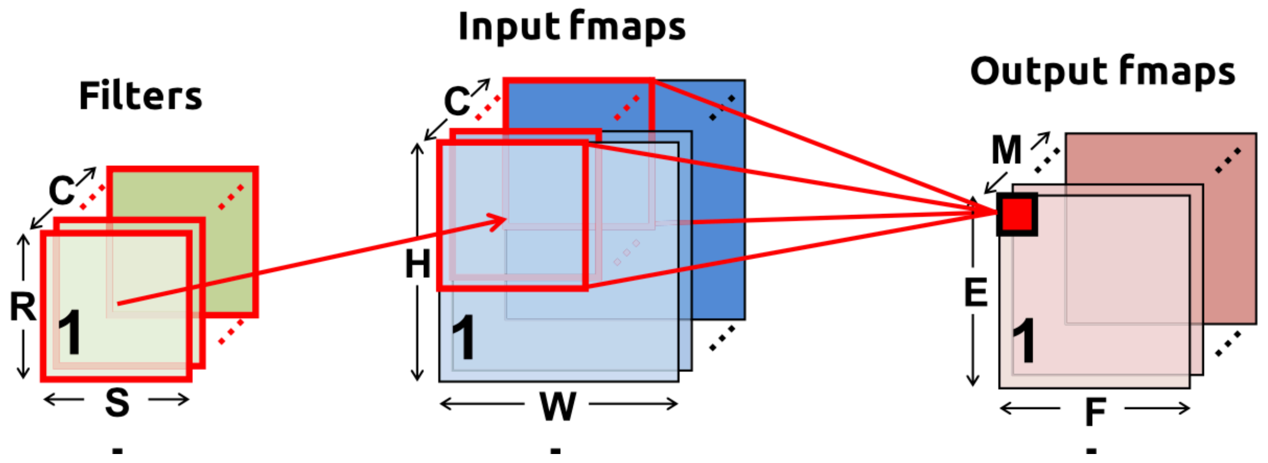
- Slightly outperforming MobileNet v2
- But based on SE blocks that were explicitly made part of the search

MobileNet v3 (2019): combining MNASNet and v2

EfficientNet (2019, 2020): combination of skip connections, depthwise-separable convolutions, SE blocks, and NAS

- Optimization goal: $\text{ACC}(m) \cdot (\text{FLOPs}(m)/T)^w$
- For test accuracy $\text{ACC}(m)$ and number of FLOPs $\text{FLOPs}(m)$ of a given model m, trade-off hyperparameter w that governs the relative importance, and target FLOP count T

Even more: SqueezeNext, LogicNets (FPGAs), ShuffleNet (2017, 2018), …



EFFICIENCY METRICS

	MACs	Parameters (weight state)	Units (activation state)
FC	$\mathrm{MAC}_f = WHCO$	$W_f = WHCO$	$U_f = O$
Convolution	$\mathrm{MAC}_c = (EF \cdot RSC) \cdot M$	$W_c = RSCM$	$U_c = EFM$
Grouped convolution	$\mathrm{MAC}_{cg} = \dfrac{\mathrm{MAC}_c}{g}$	$W_{cg} = \dfrac{W_c}{g}$	$U_{cg} = EFM$
Depthwise separable convolution	$\mathrm{MAC}_{cds} = EF \cdot (RSC + CM)$	$\mathrm{W}_{cds} = RSC + CM$	$U_{cds} = EFC + EFM$

BEYOND CONVOLUTIONS

With material from Roger Grosse (U. Toronto, CSC421/2516)

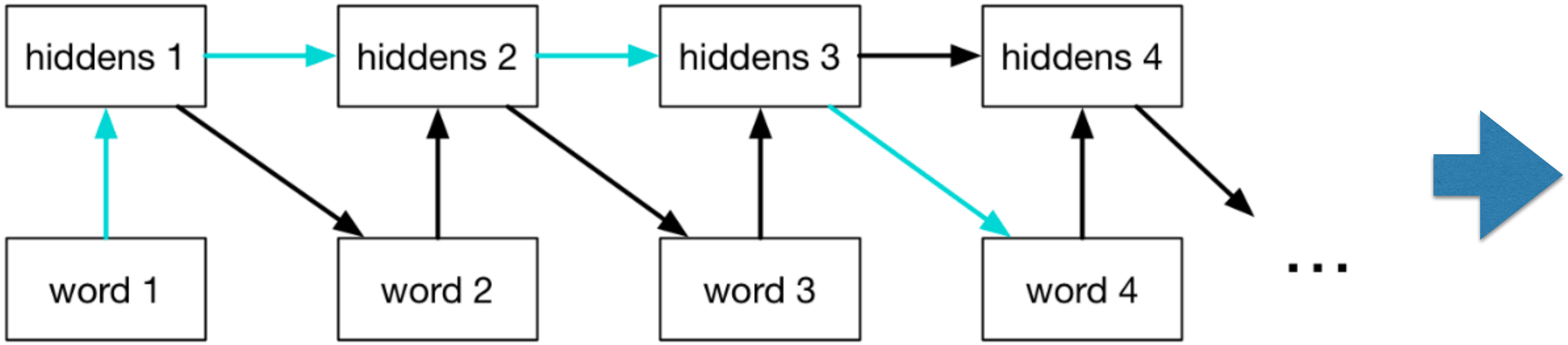
RECURRENT NEURAL NETWORKS

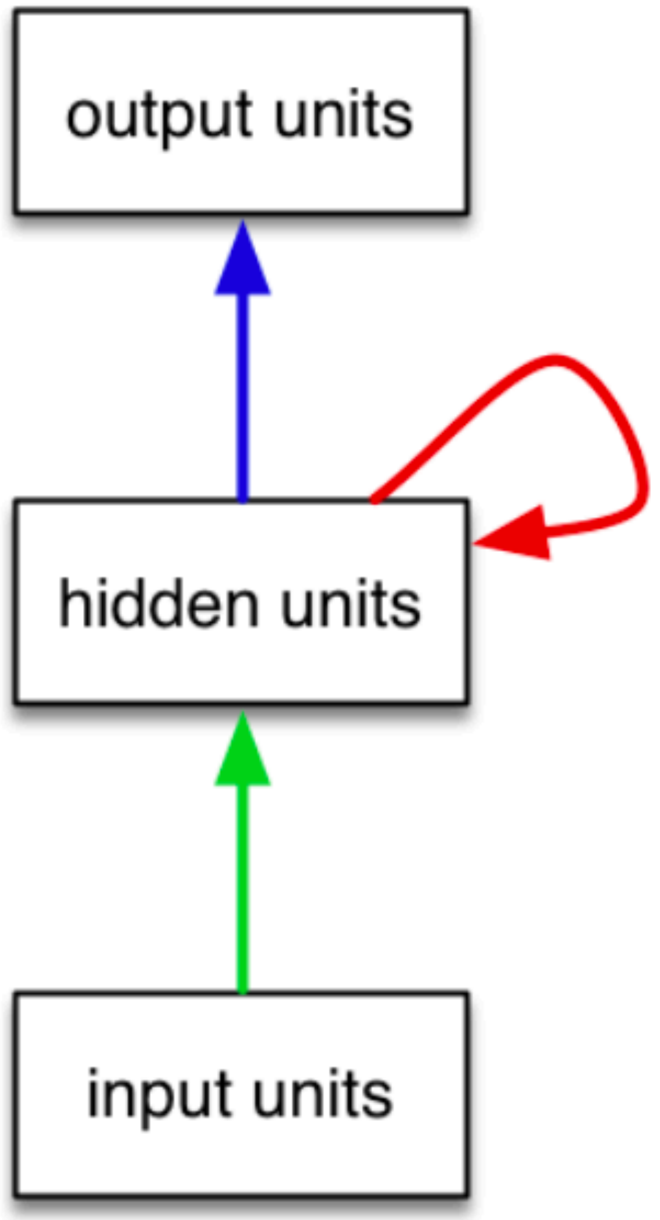
Predicting sequences: speech-to-text and text-to-speech, caption generation, machine translation, signal processing

- Sequence-to-sequence prediction: input is also a sequence
- Recurrent neural networks (RNNs) have memory which is very helpful for sequences
- For no memory, see neural language models and c.f. Markov assumption (model is memoryless)

=> RNNs are based on a hidden state/unit

- Recurrent NN vs. feedforward NN



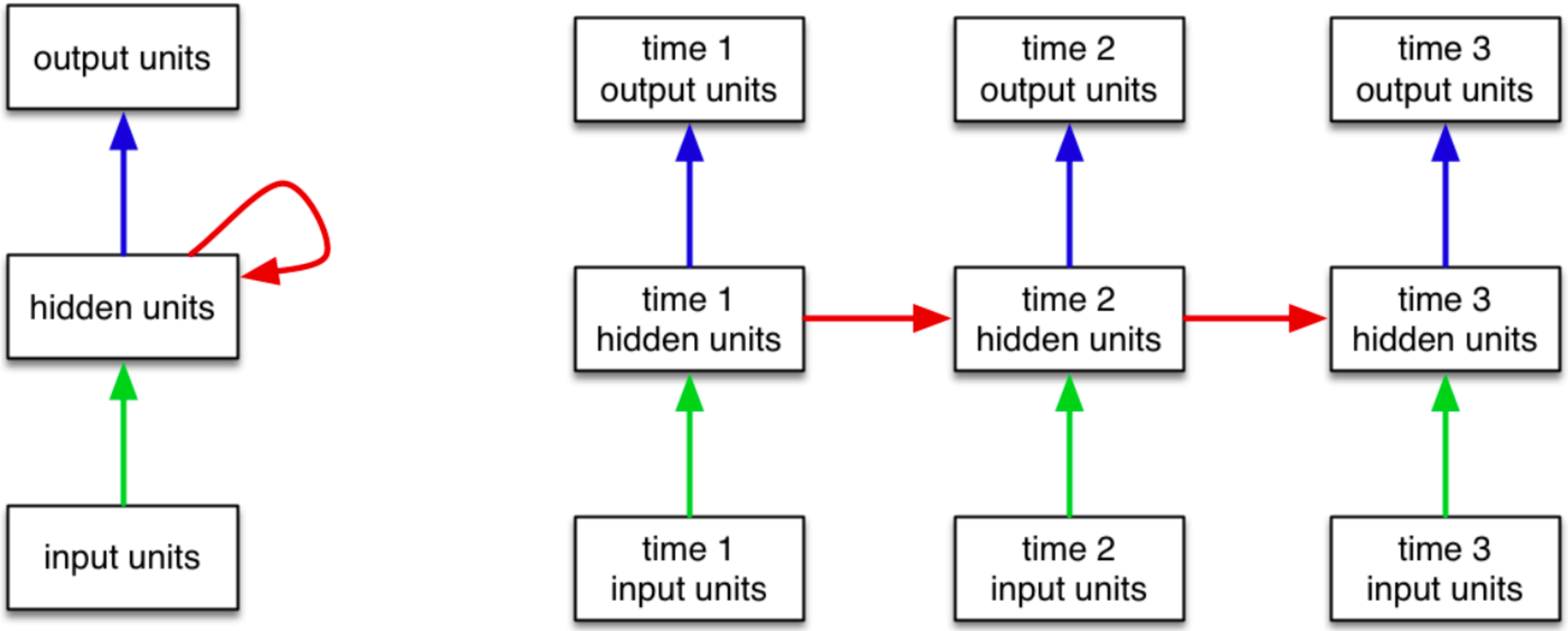


UNROLLING RNNS

RNN: dynamical system with one set of hidden units that feed into themselves

Unrolling: representing the units at all time steps

- Parameters are shared among all time steps
- Sometimes the biases for the first time step are treated differently



BACKPROP THROUGH TIME

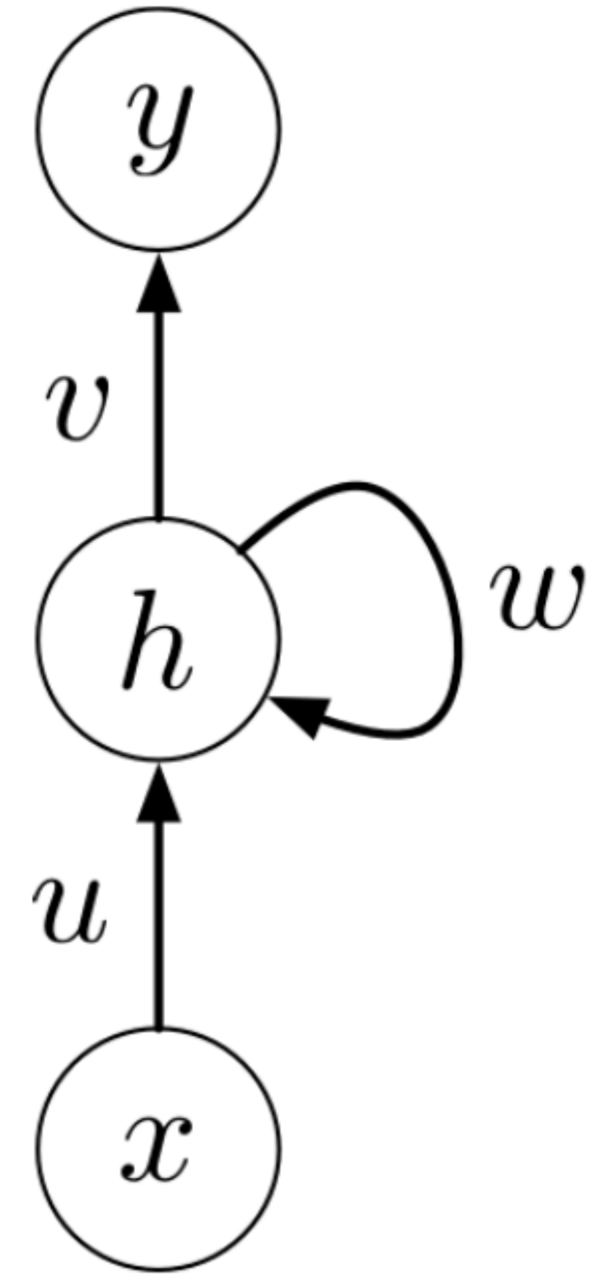
Unrolling allows to re-use backpropagation to train RNNs

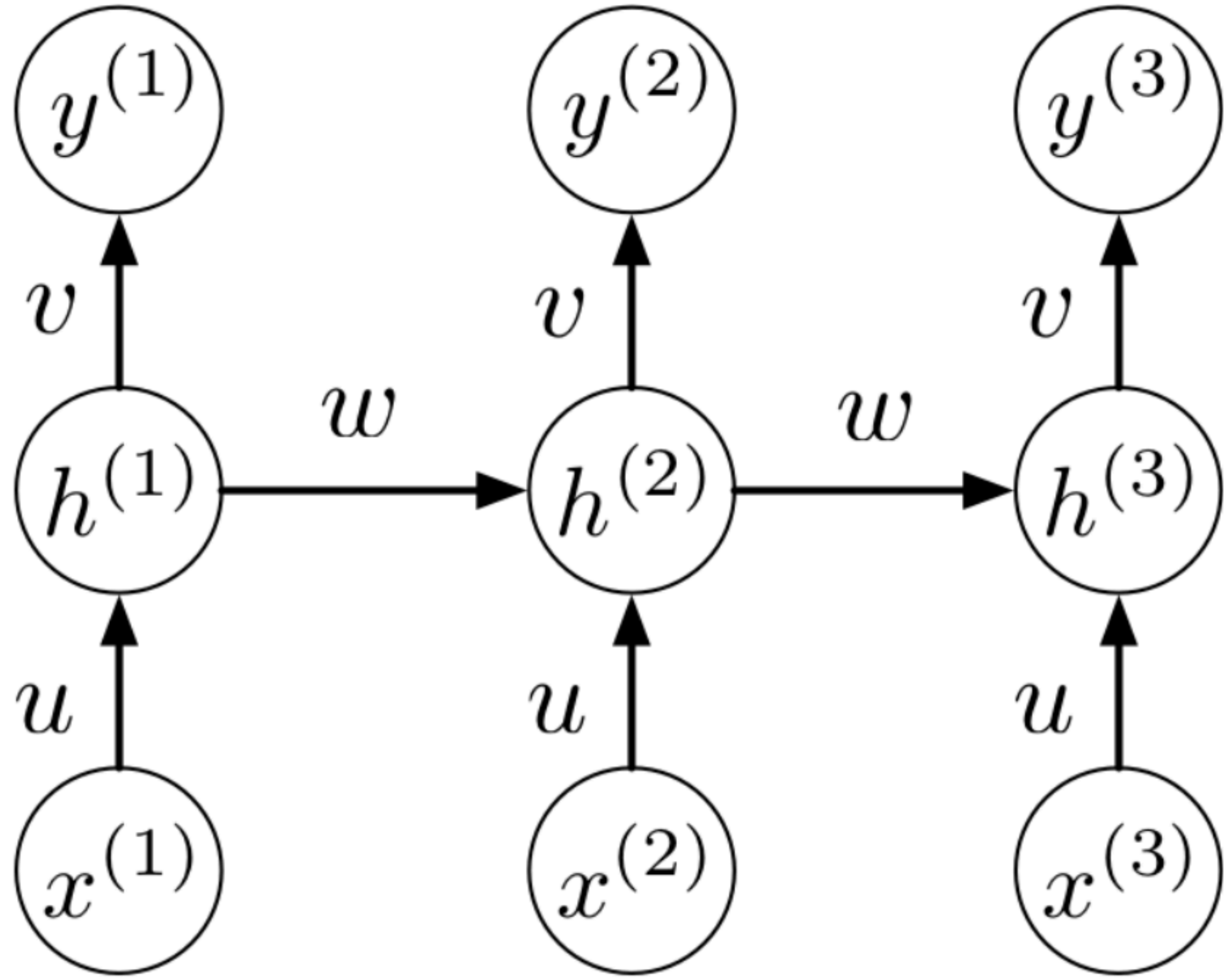
- Careful with weight sharing (u, w, v)
- Unrolling effectively increases the depth of the model => backprop is very sensitive to vanishing and exploding gradients

=> Essentially simple, very difficult in practice

- Usually RNNs are small
- Not a prime example for model compression
- Contrary, inherently resource-efficient

Other scalability issues, as difficult to predict long sequences

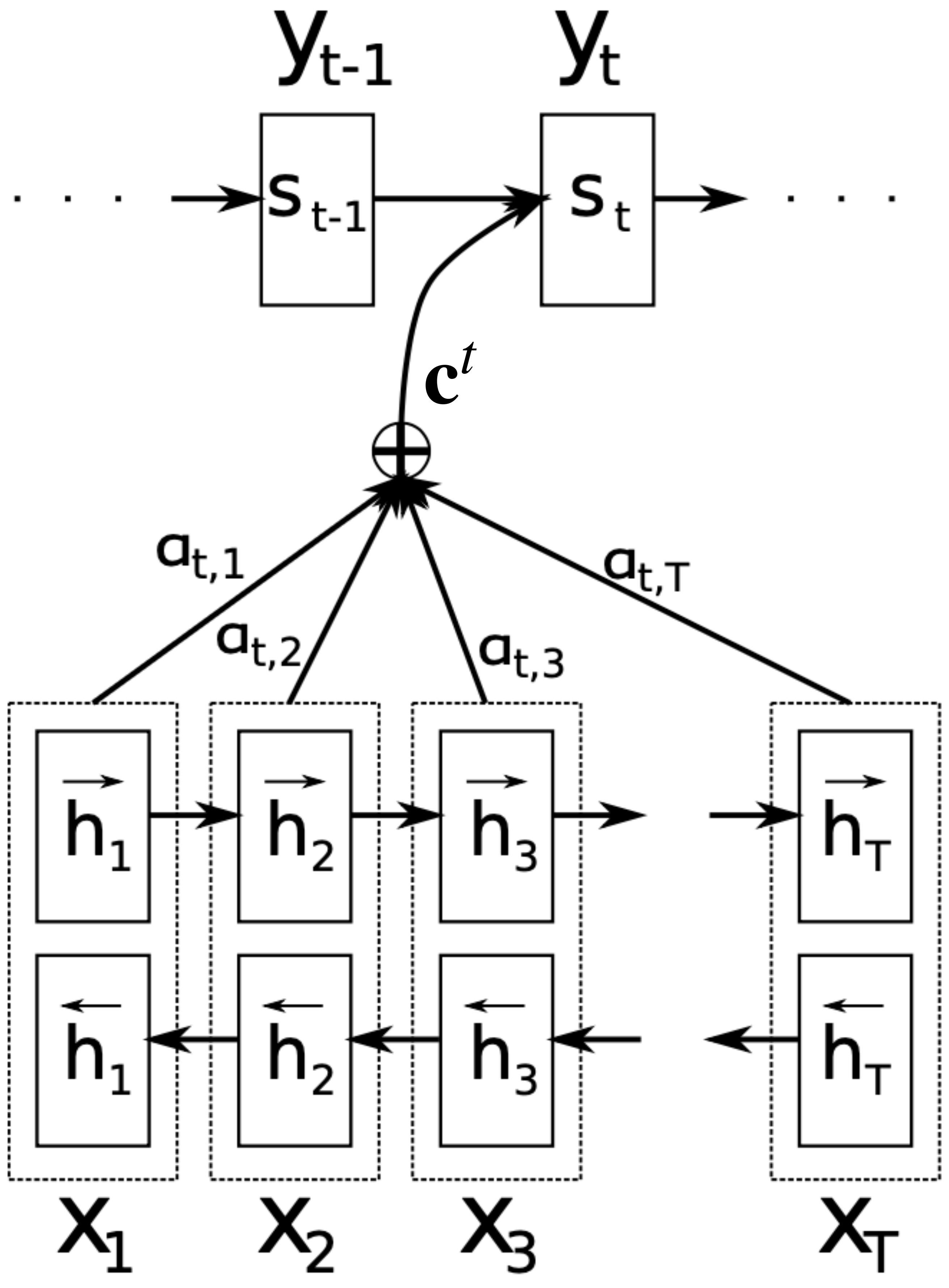




ATTENTION FOR LONG-RANGE SEQUENCE MODELING

Attention-based Neural Machine Translation (NMT) encoder-decoder

1. Encode the source sentence $\mathbf{x}$ (e.g., French) into a sequence of hidden states
2. Attend selectively to different parts of the source at each decoding step.
3. Decode by generating the target sentence $\mathbf{y}$ (e.g., English) one token at a time



Bidirectional RNN as encoder, normal RNN as decoder

Encoder: Forward hidden states $\overrightarrow{\mathbf{h}}$, and backward hidden states $\overleftarrow{\mathbf{h}}$

(annotation vector $\mathbf{h}$ by concatenating $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$)

Attention: compute context vector $\mathbf{c}^t$ for each time step t

Decoder: decides which part of the source to pay attention to, and combines with own hidden state $\mathbf{s}$

Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, 2014, https://arxiv.org/abs/1409.0473

ATTENTION

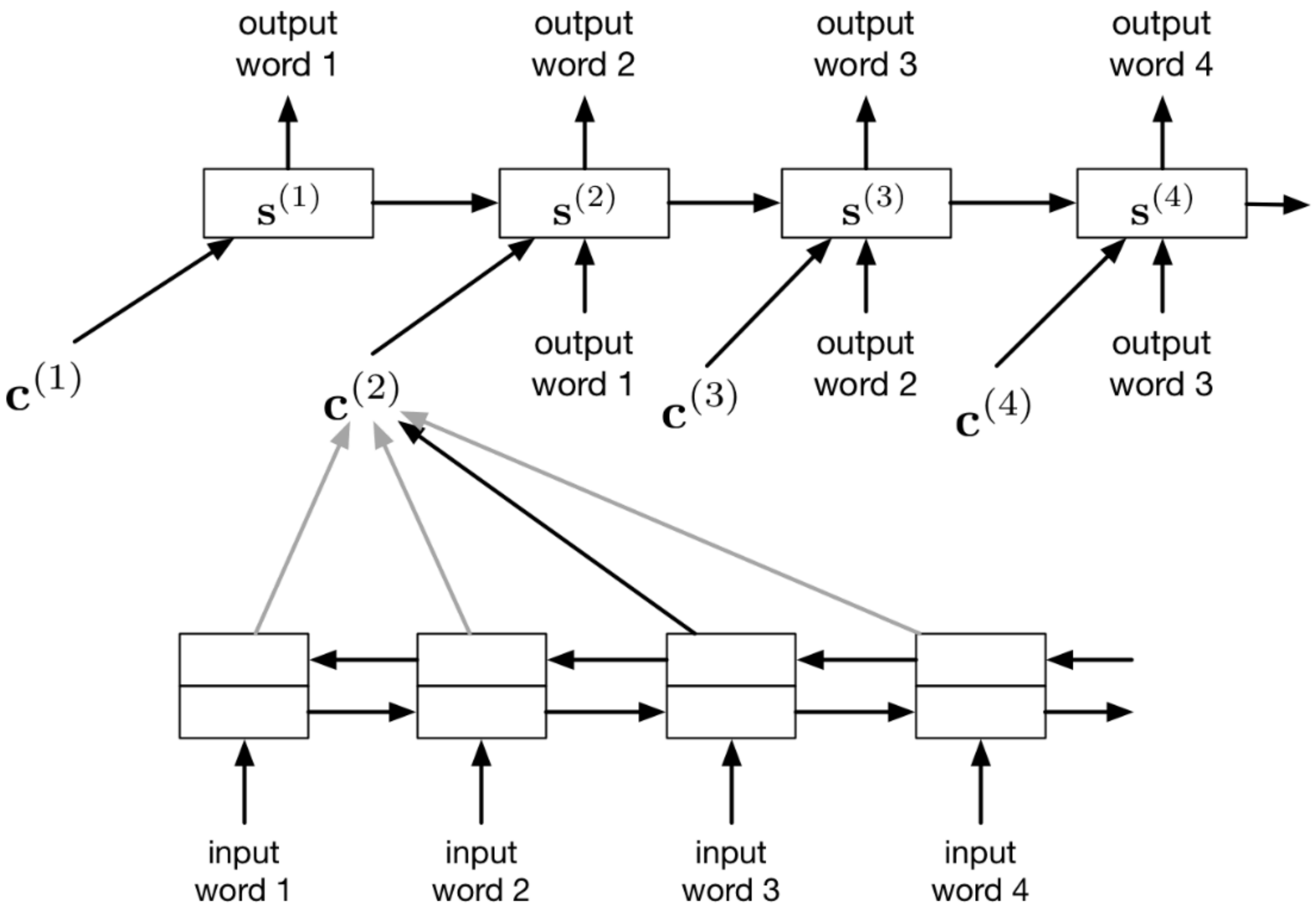
Context is a weighted sum of the encoder annotations: $c^i = \sum_j \alpha_{ij} \cdot h_j$

Context vector needs to be computed every time step

Weight α_{ij} of each annotation h_j

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}$$

, for an alignment model a that scores how well input at position j and output at position i match: $e_{ij} = a(s_{i-1}, h_j)$



ATTENTION

Content-based addressing: attention function depends on annotation vector rather than on position in the sentence

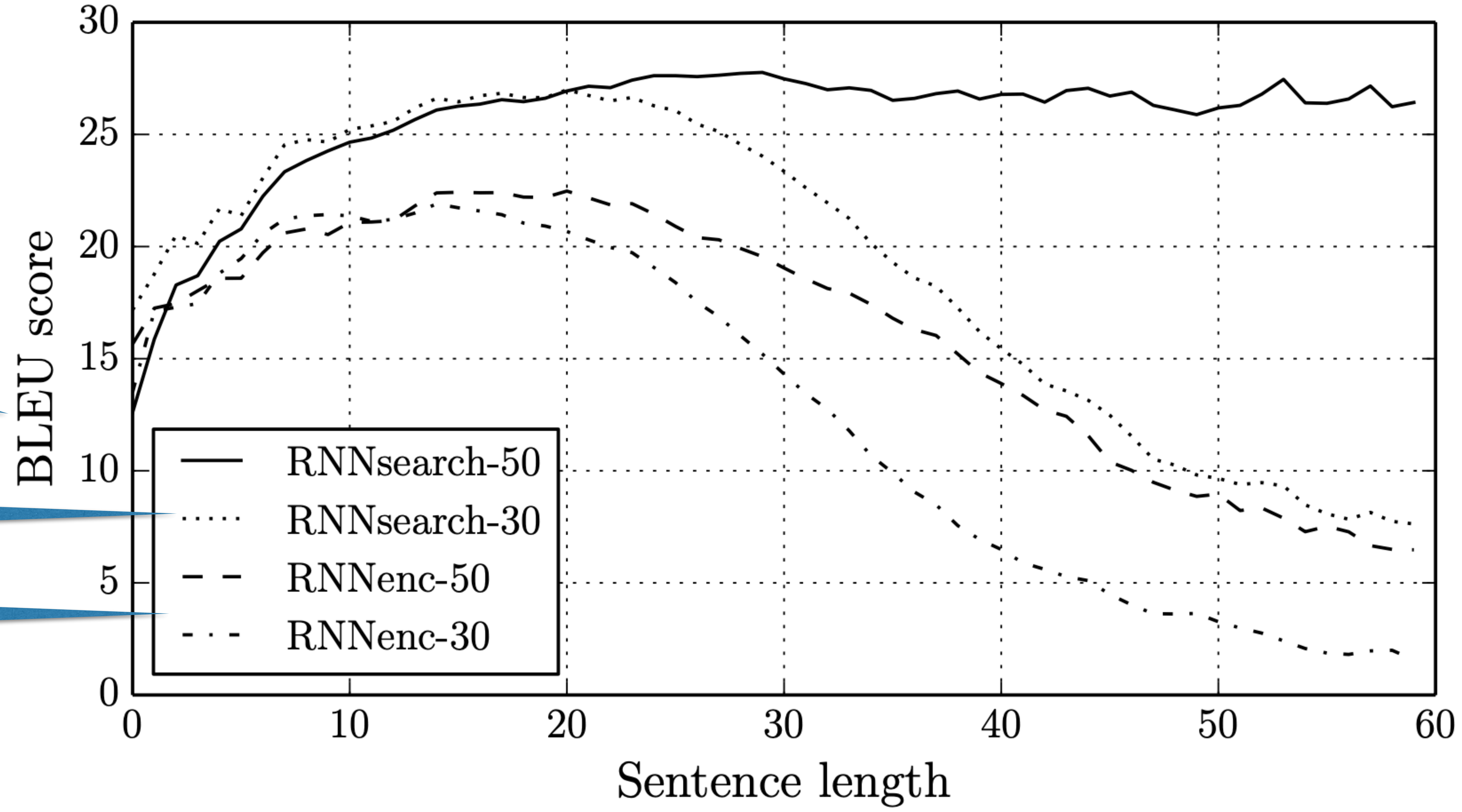
Inline with “biology”

C.f. convolutions

BLEU score: higher is better

Attention-based encoder-decoder

Plain RNN encoder-decoder



Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, 2014, https://arxiv.org/abs/1409.0473

COMPUTATIONAL COSTS

Sequence length t, number of layers d, number of neurons per layer k

Transformer is not covered here, but essentially it replaces RNNs with feedforward model architectures

	Training complexity	Training memory	Test complexity	Test memory
RNN	tk^2d	tkd	tk^2d	kd
RNN + attention	t^2k^2d	t^2kd	t^2k^2d	tkd
Transformer	t^2kd	tkd	t^2kd	tkd

WRAPPING UP

SUMMARY

Many advanced tricks to improve resource efficiency of convolutional neural networks

- Pay attention on “efficiency metrics” (FLOPs != latency, state is not only defined by number of parameters, …)
- There is more than MACs and parameters

RNNs extend feed-forward models by having memory

- But are difficult to train, effectively reducing the scalability of such architectures
- Attention mechanisms is much more effective than plain memory (all history)
- Transformers proven to be much more scalable, now also used in image processing, document processing, etc.

Image classification foundational for many image processing tasks

- Natural language processing continues this story, but at a different scale
- => Era of foundational models