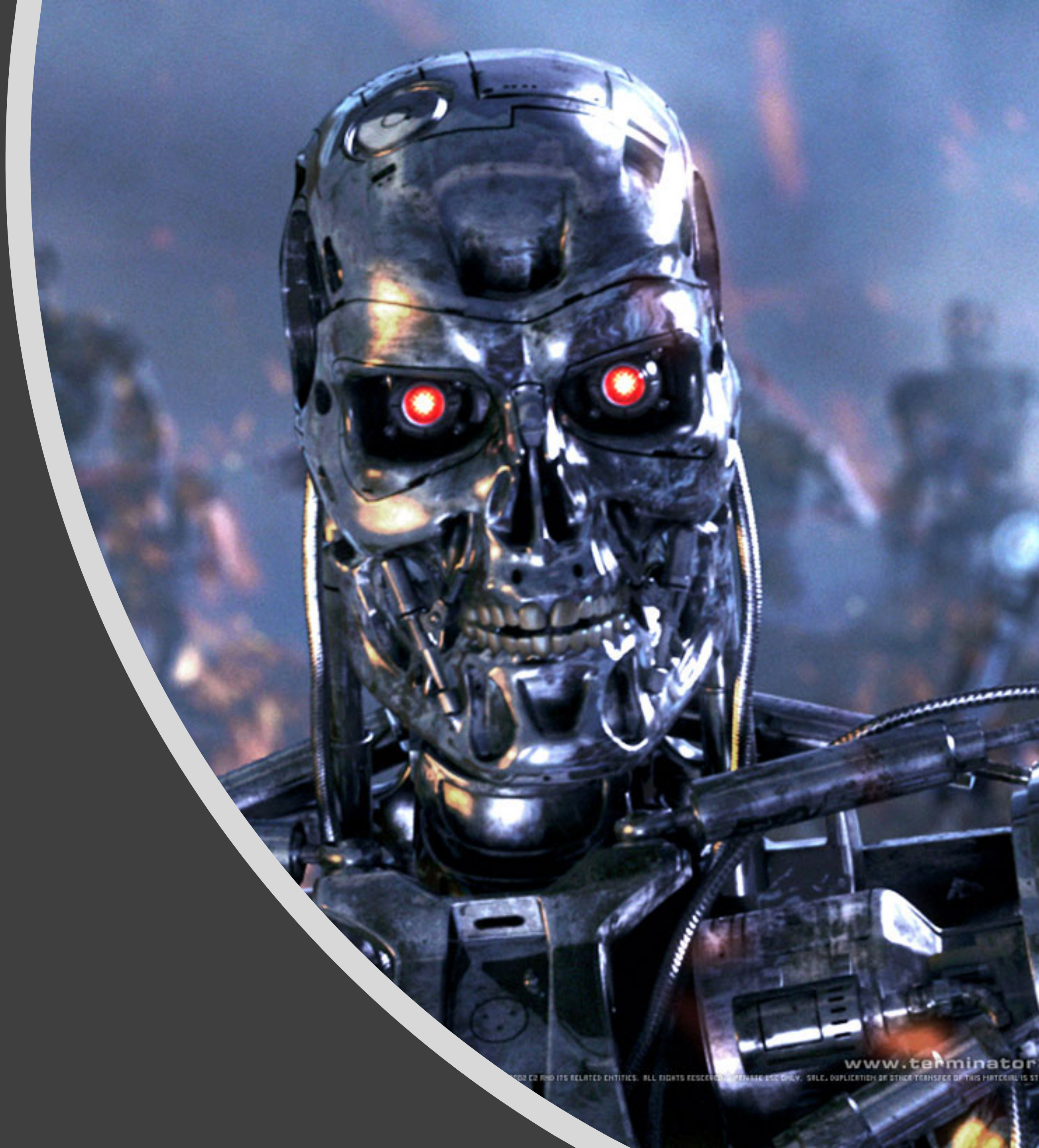
An introduction to neural network compression

Elliot J. Crowley

School of Informatics, University of Edinburgh

The appeal of small networks

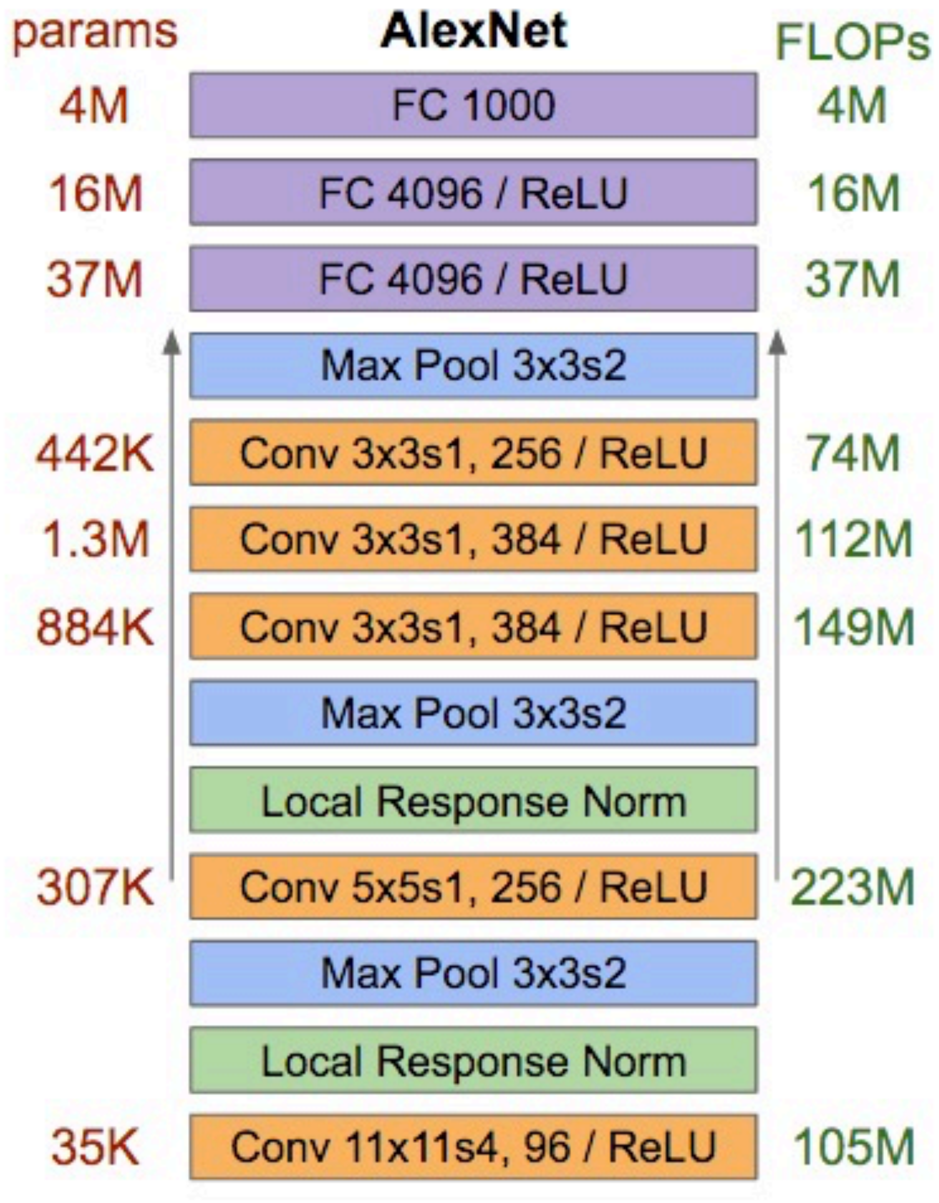
- Convolutional Neural Networks are pretty good (and definitely not overhyped, see robot →)
- The winner of the 2017 ImageNet Challenge was an ensemble of 115-million parameter Squeeze-Excite nets
- Good luck fitting this on your smartwatch
- Smaller networks allow for faster inferences for real-time applications



How do we do make our networks smaller?

- **Architecture design**
- Neural network pruning
- Network distillation

Don’t use big fully connected layers!



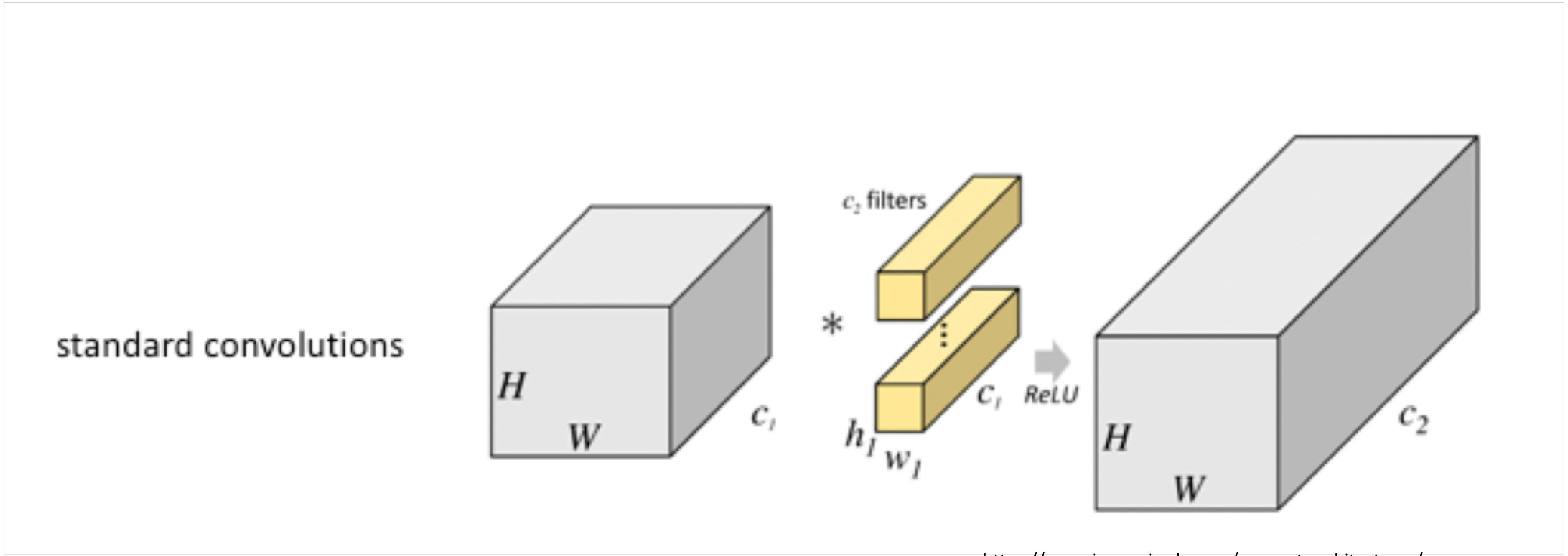
params	AlexNet	FLOPs
4M	FC 1000	4M
16M	FC 4096 / ReLU	16M
37M	FC 4096 / ReLU	37M
	Max Pool 3x3s2	
442K	Conv 3x3s1, 256 / ReLU	74M
1.3M	Conv 3x3s1, 384 / ReLU	112M
884K	Conv 3x3s1, 384 / ReLU	149M
	Max Pool 3x3s2	
	Local Response Norm	
307K	Conv 5x5s1, 256 / ReLU	223M
	Max Pool 3x3s2	
	Local Response Norm	
35K	Conv 11x11s4, 96 / ReLU	105M

https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html

Avoid AlexNet and VGG nets

``I can’t believe people still use VGG nets’’

- Karen Simonyan, author of the VGG nets paper



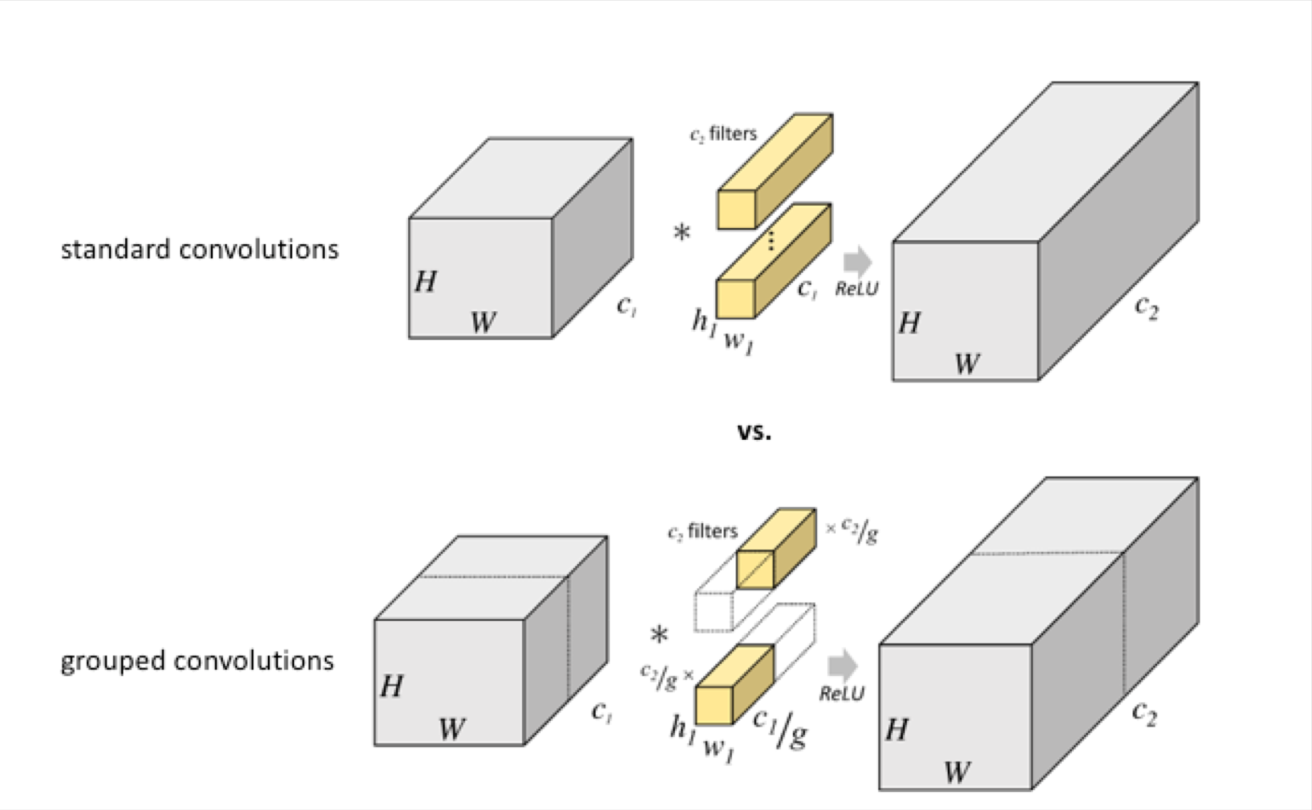
https://www.jeremyjordan.me/convnet-architectures/

Convolution Reminder

- A convolutional layer takes a C1 channel input and spits out a C2 channel output
- It consists of C2 filters each of size h1*w1*C1
- This uses C2*C1*h1*w1 parameters
- Channel size can get quite big (usually up to 512)

Grouped Convolutions

- In a grouped convolution, we split the input along the channel dimension
- Picture this as a bunch of smaller convolutions, each going from C1/g channels to C2/g channels
- Each of these uses (C2/g)*(C1/g)*h1*w1 parameters
- **But there are g of them** so the total cost is (C2/g)*(C1/g)*h1*w1*g
- (C2/g)*(C1/g)*h1*w1*g = C2*C1*h1*w1*(1/g) = original_cost /g



What’s the catch?

``So this means I can split all my convolutions into loads of groups and enjoy a massive parameter reduction without any consequences.’’

– Geoff Hinton*

Q: How is Geoff mistaken?

* Geoff Hinton probably didn’t say this

Channel mixing builds up concepts

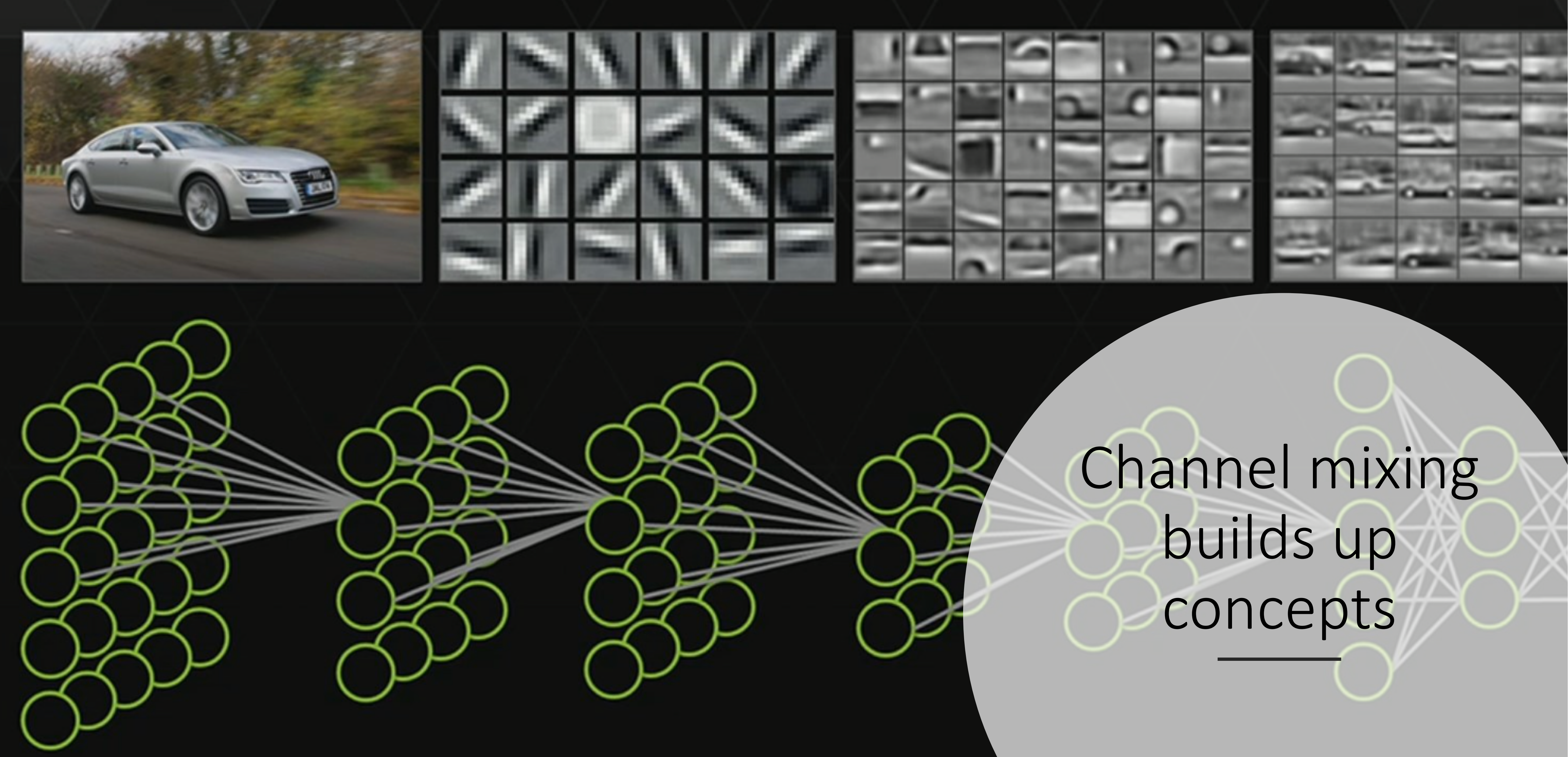
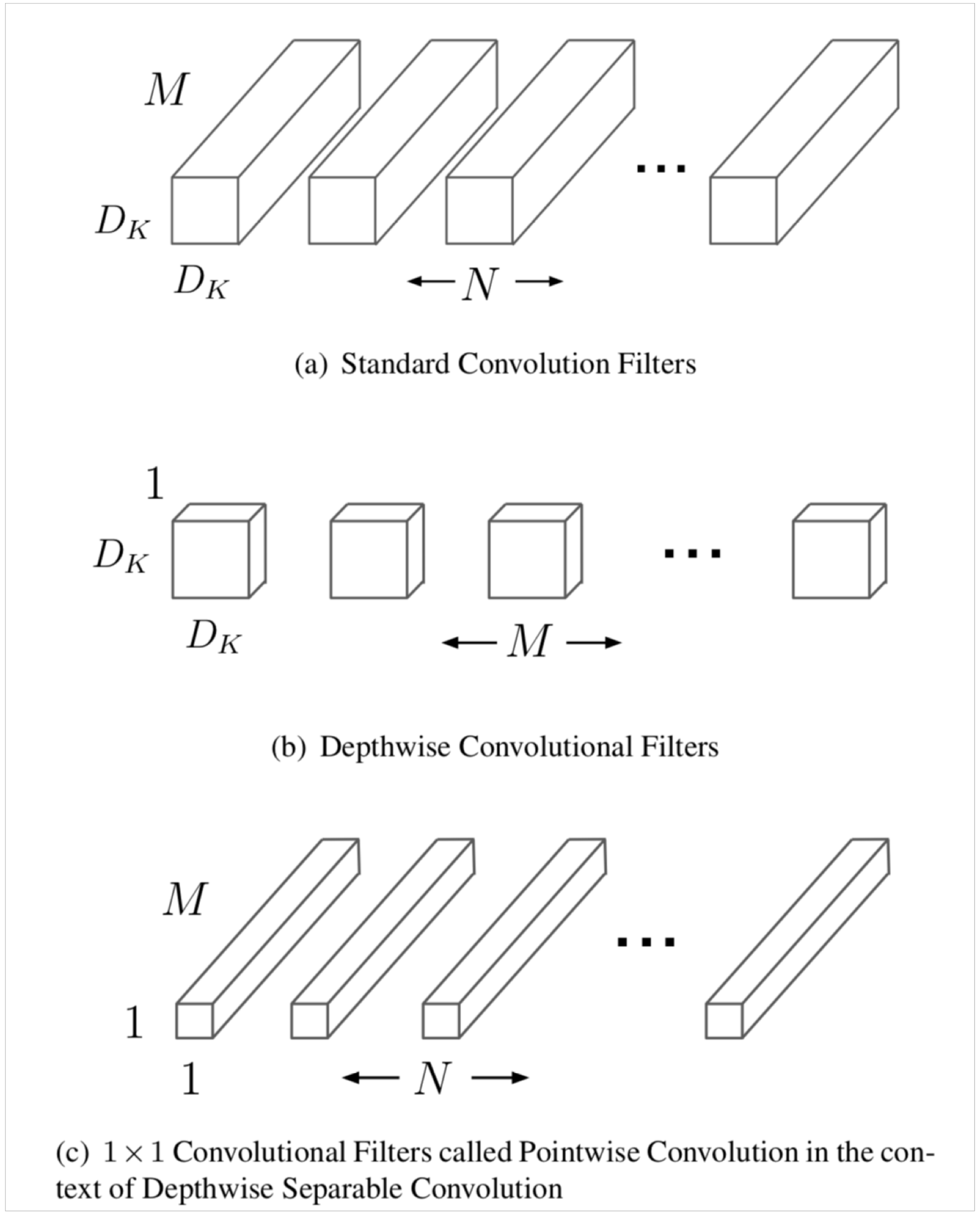


Image source: "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks" ICM
Honglak Lee, Roger Grosse, Rajesh

Channel mixing!

- In MobileNet (https://arxiv.org/pdf/1704.04861.pdf) the authors split all their convolutions into the maximum number of groups (g equal to the number of input channels)
- Channels are then mixed by using a **pointwise convolution**
- This uses a 1x1 kernel so uses 1 * 1 * C1 * C2 parameters

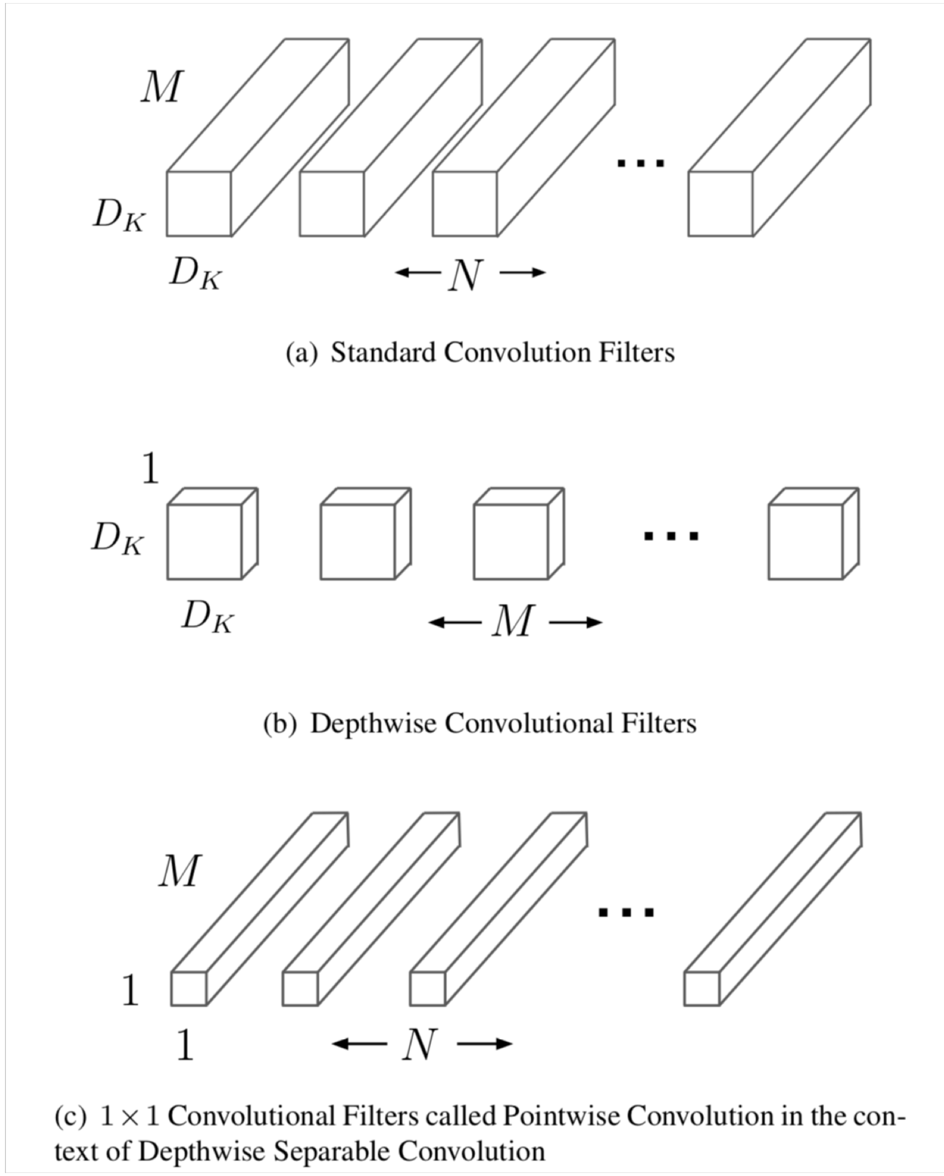

M
D_K
D_K
$\leftarrow N \rightarrow$
1
D_K
D_K
$\leftarrow M \rightarrow$
M
1
1
$\leftarrow N \rightarrow$

(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

M = 256
N = 512
Dk = 3


M
D_K
D_K
$\longleftarrow N \longrightarrow$
1
D_K
D_K
$\longleftarrow M \longrightarrow$
M
1
1
$\longleftarrow N \longrightarrow$

(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

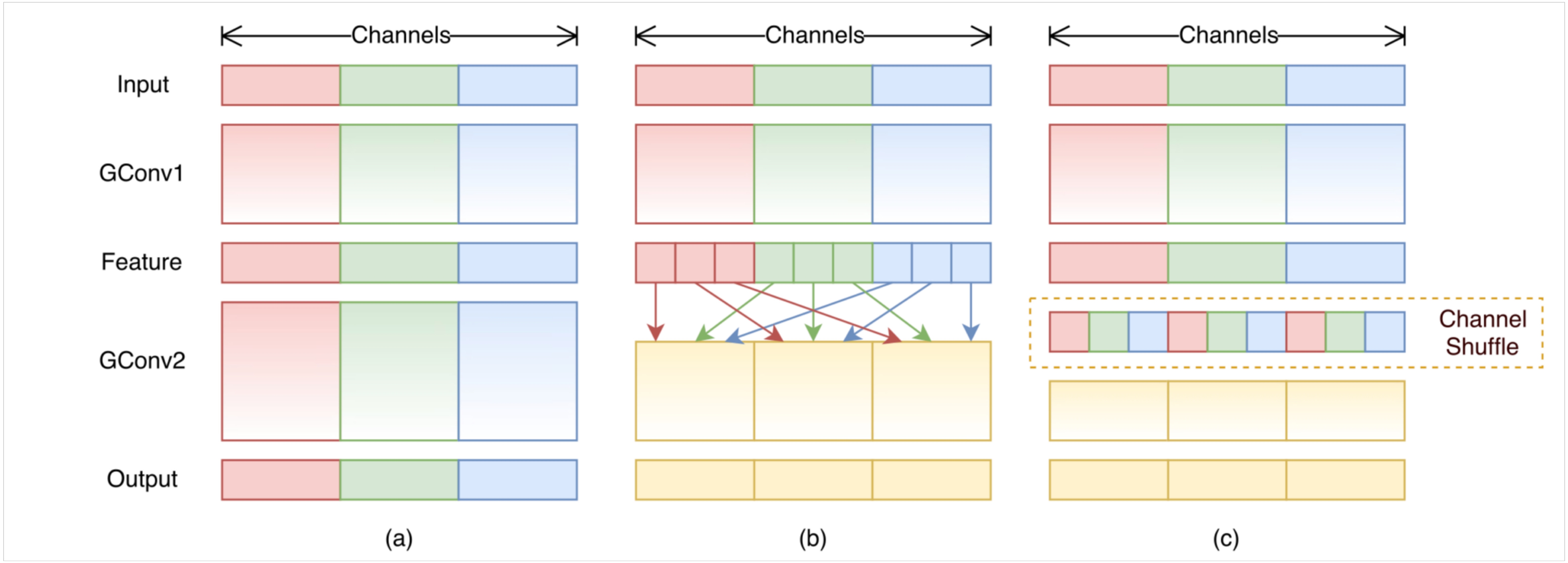
(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

1,179,648 parameters

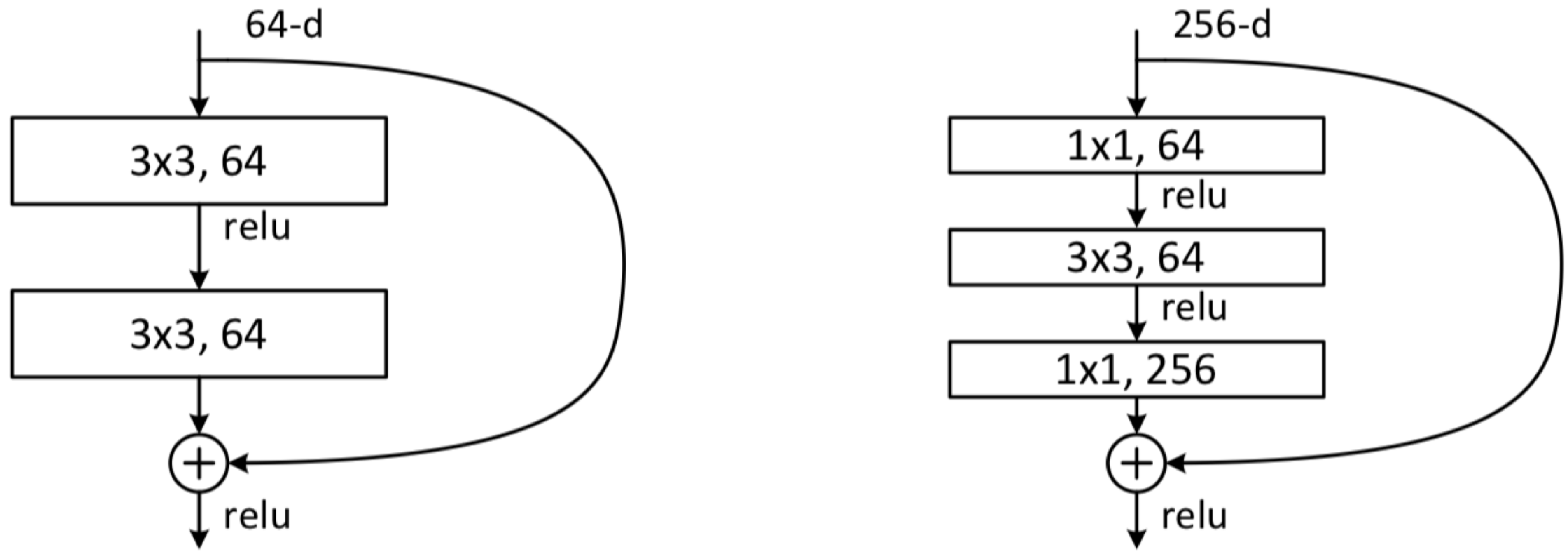
2,304 parameters

131,072 parameters

Enter ShuffleNet (https://arxiv.org/pdf/1707.01083.pdf)

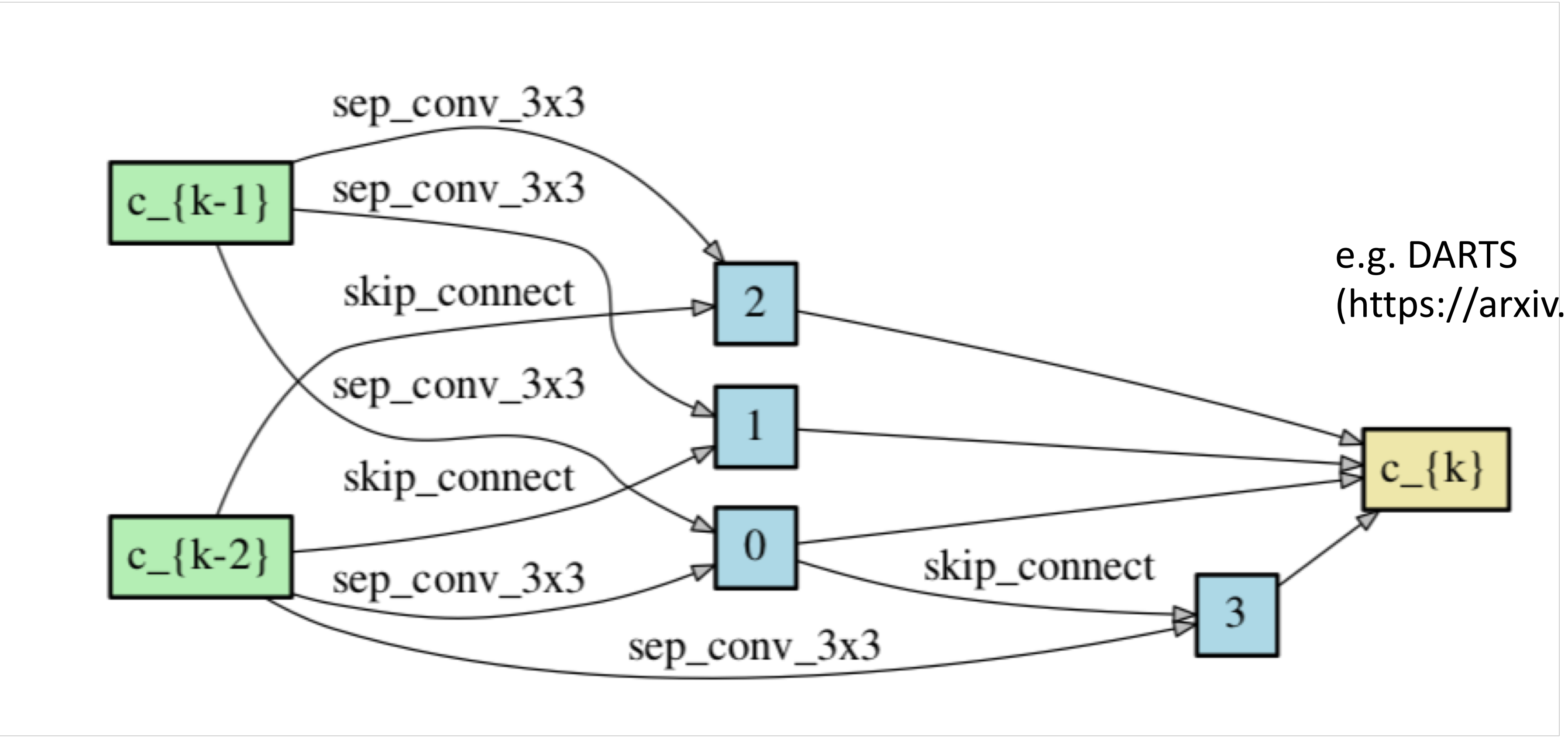


Bottlenecks



https://arxiv.org/pdf/1512.03385.pdf

Neural Architecture Search



e.g. DARTS
(https://arxiv.org/pdf/1806.09055.pdf)

How do we do make our networks smaller?

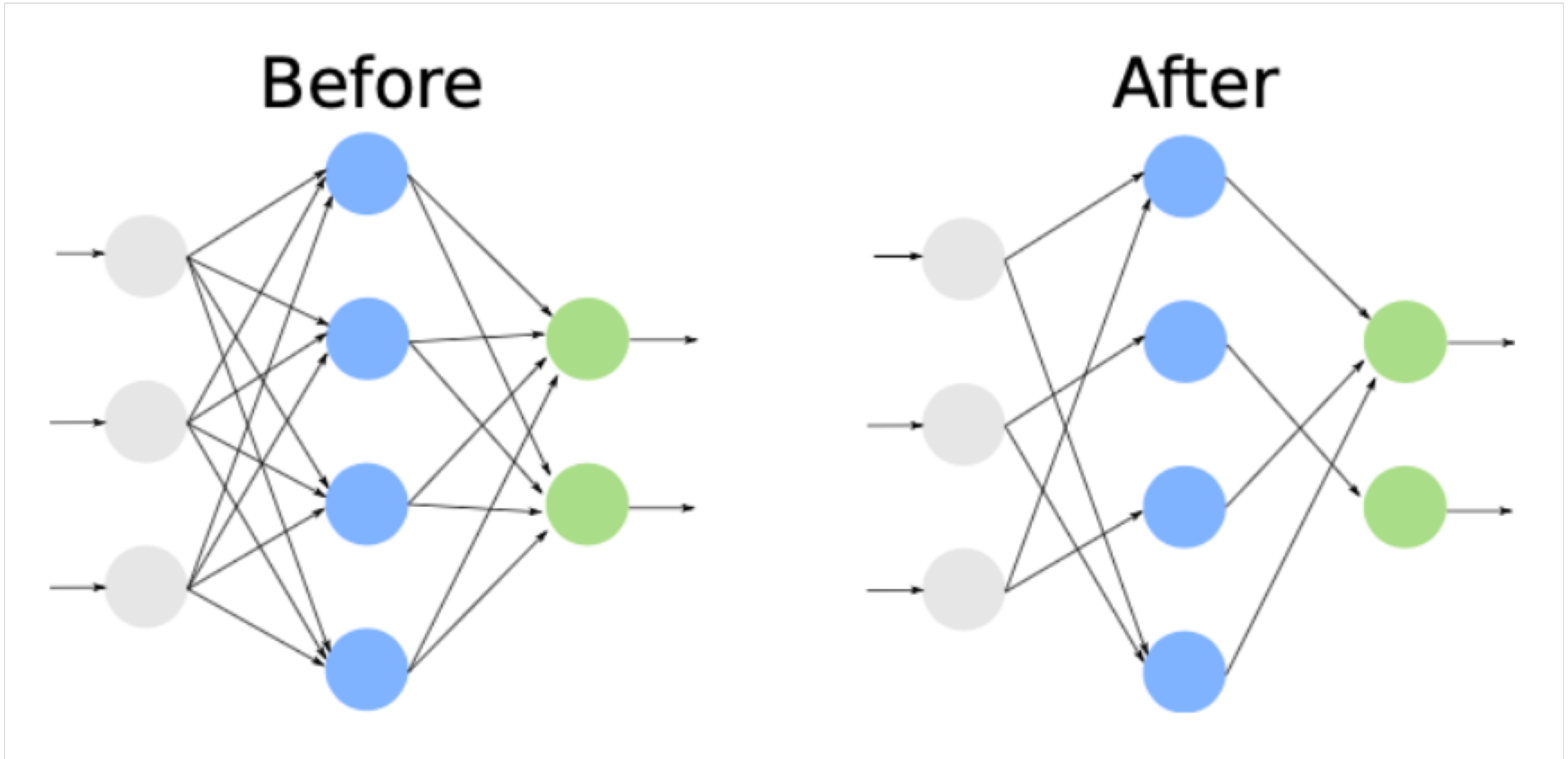
- Architecture design
- **Neural network pruning**
- Network distillation

Pruning neural networks: alternative facts

- Neural network pruning, like most of deep learning, was invented in 1997 by Jurgen Schmidhuber
- He hypothesised that neural networks are basically plants and should therefore be watered and pruned
- Because neural networks live inside computers, Schmidhuber realised that watering would be difficult, and settled on pruning

Exhibit A: Jurgen Schmidhuber posing for the camera with a sad fanboy

Weight pruning



Deep Compression – a weight pruning technique

https://arxiv.org/pdf/1510.00149.pdf

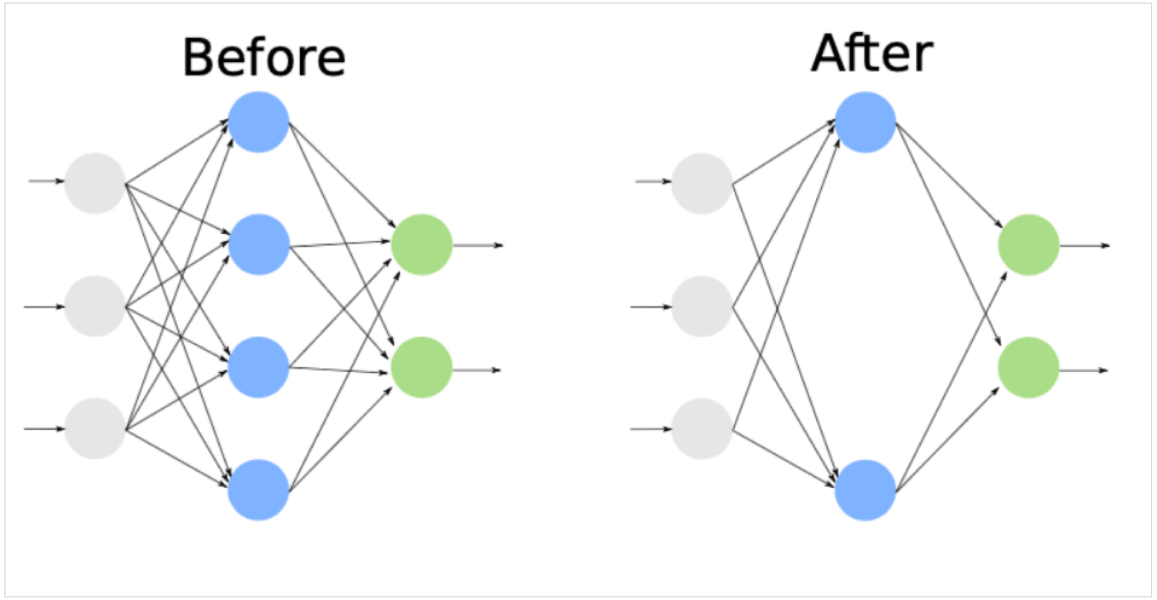
1. Start with a trained network, and set threshold T
2. Rank all non-zero weights by their magnitude
3. Set the T% lowest ranked weights to zero
4. Fine-tune the network and increase T. Return to step 2

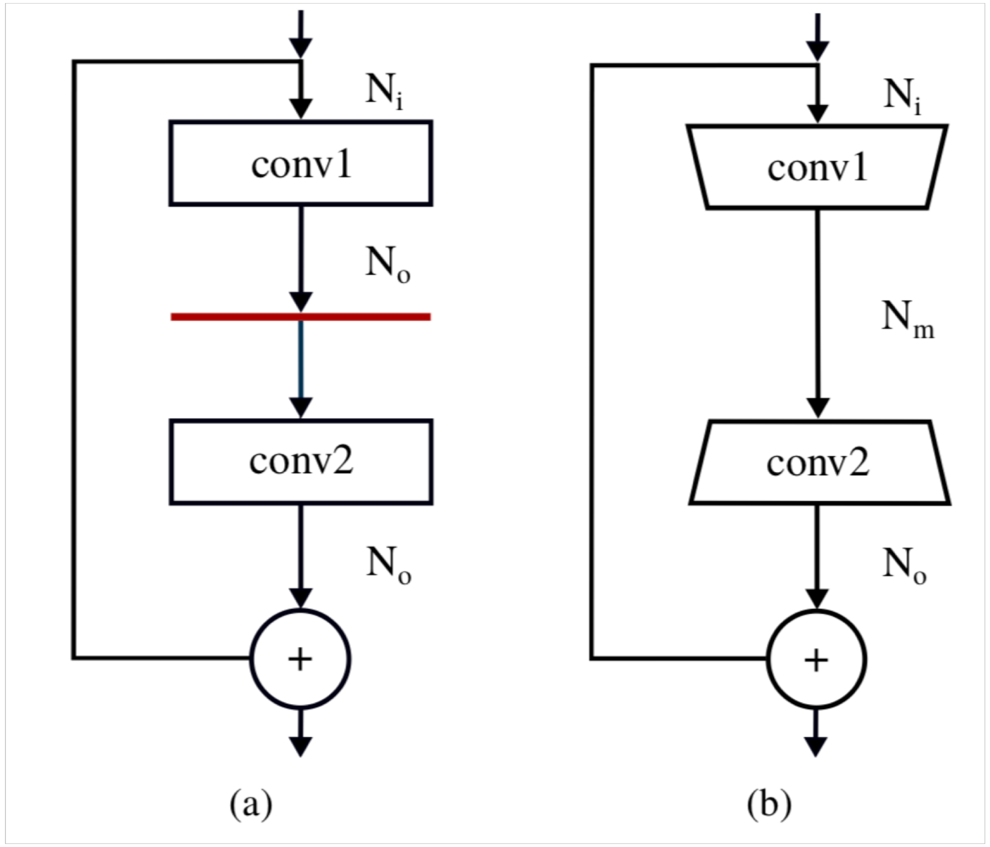
This massively compresses VGG nets with very little loss of accuracy. Why is this?

The problem with weight pruning

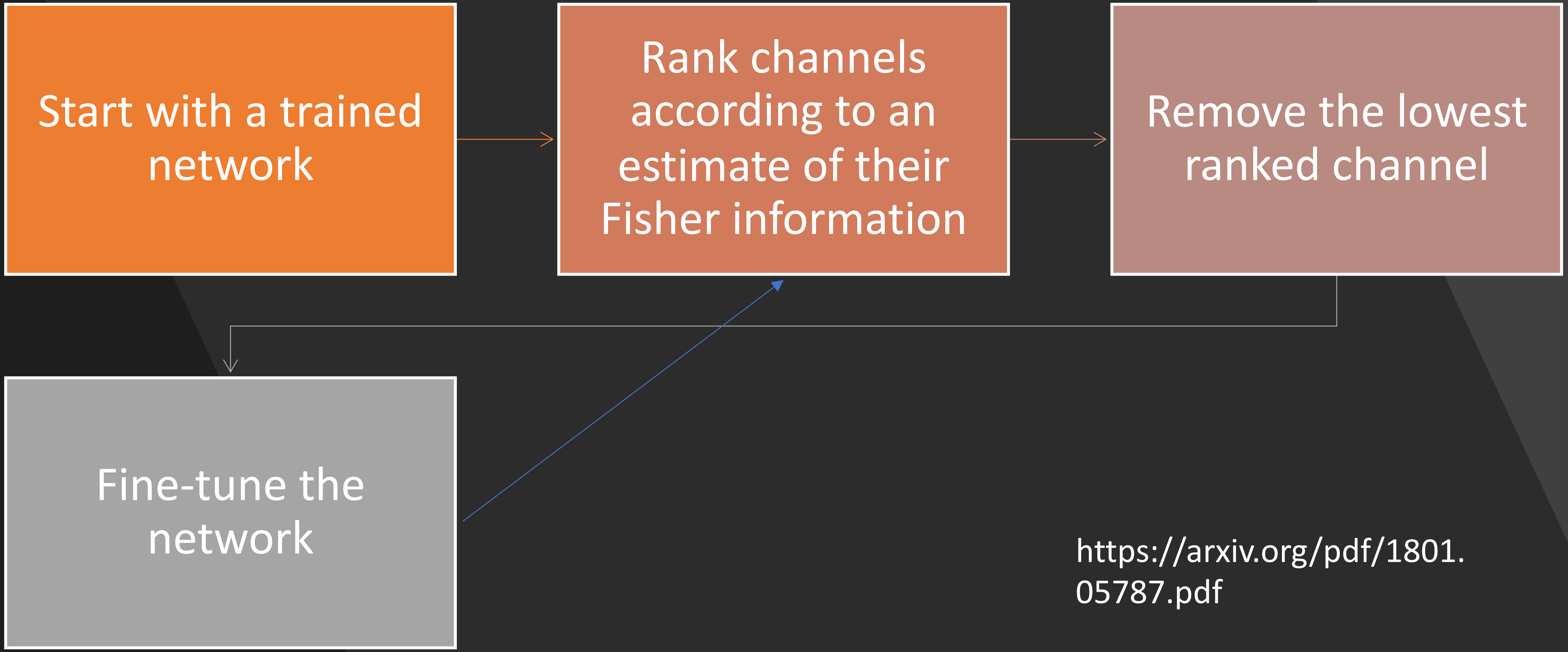
- It results in sparse weight matrices, which are difficult to leverage into actual performance improvements on general purpose hardware ☹ (see our work: https://arxiv.org/pdf/1809.07196.pdf)
- Not even TPUs have in-built support for sparsity ☹
- It targets the giant fully connected layers that don’t need to be there in the first place ☹

Channel pruning





Fisher pruning – a channel pruning technique



Start with a trained network

Rank channels according to an estimate of their Fisher information

Remove the lowest ranked channel

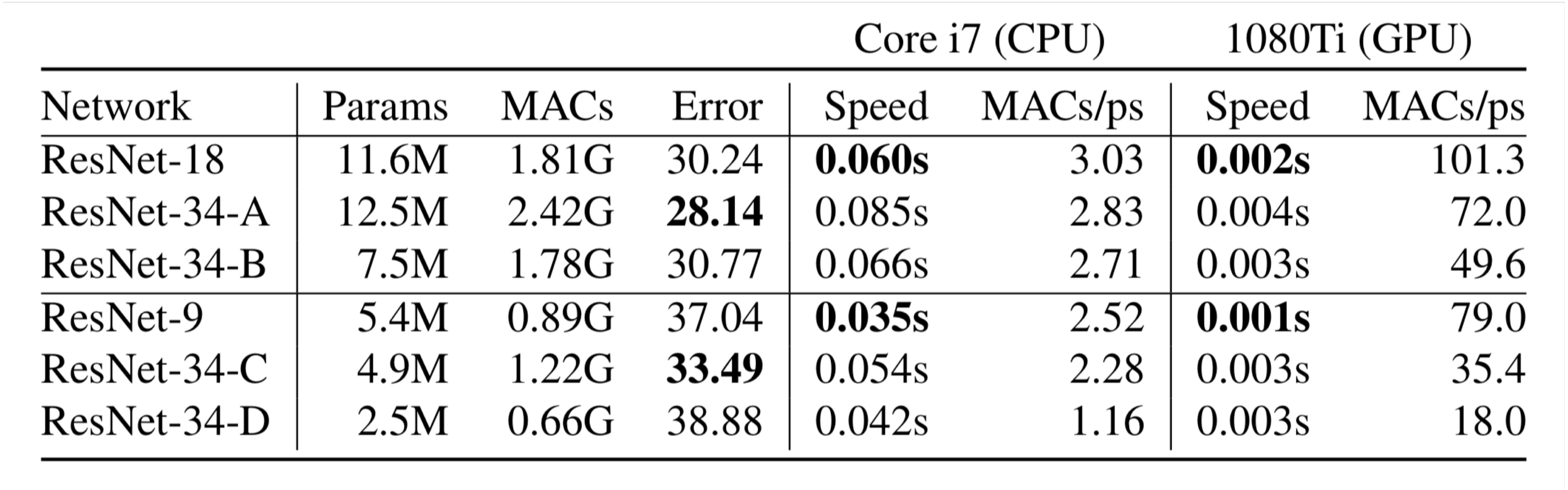
Fine-tune the network

https://arxiv.org/pdf/1801.05787.pdf

The problem with pruning

Smaller versions of the original network trained from scratch outperform pruned networks

However, if we take a pruned network, reset its weights and train from scratch these are powerful



				Core i7 (CPU)		1080Ti (GPU)	
Network	Params	MACs	Error	Speed	MACs/ps	Speed	MACs/ps
ResNet-18	11.6M	1.81G	30.24	**0.060s**	3.03	**0.002s**	101.3
ResNet-34-A	12.5M	2.42G	**28.14**	0.085s	2.83	0.004s	72.0
ResNet-34-B	7.5M	1.78G	30.77	0.066s	2.71	0.003s	49.6
ResNet-9	5.4M	0.89G	37.04	**0.035s**	2.52	**0.001s**	79.0
ResNet-34-C	4.9M	1.22G	**33.49**	0.054s	2.28	0.003s	35.4
ResNet-34-D	2.5M	0.66G	38.88	0.042s	1.16	0.003s	18.0

Pruned networks are also slow ☹

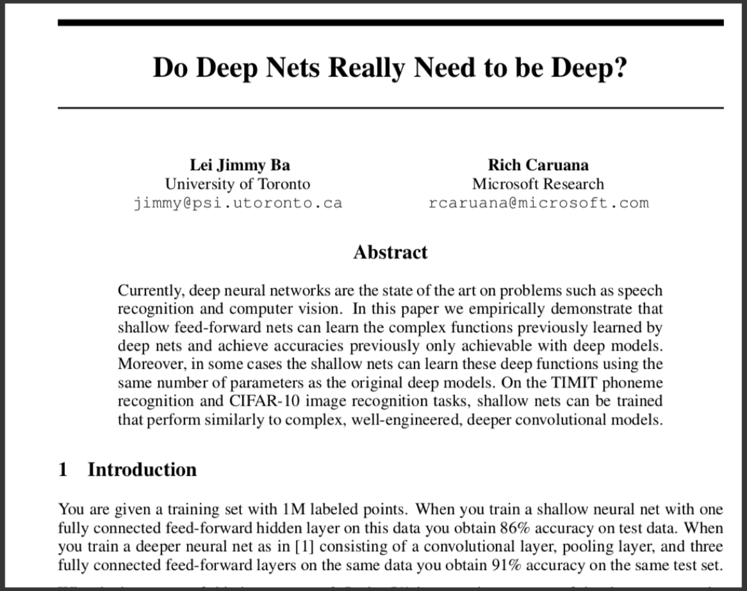
How do we make our networks smaller?

- Architecture design
- Neural network pruning
- **Network distillation**

A cautionary tale

- Network distillation can be attributed back to “Do Deep Nets Really Need to be Deep?” (2013)
- It has ~500 citations
- ”Distilling the Knowledge in a Neural Network” (1 year later) takes this work and adds a single hyperparameter T, which is always set to 4
- ~1000 citations

- What can we learn from this?



Do Deep Nets Really Need to be Deep?

Lei Jimmy Ba
University of Toronto
jimmy@psi.utoronto.ca

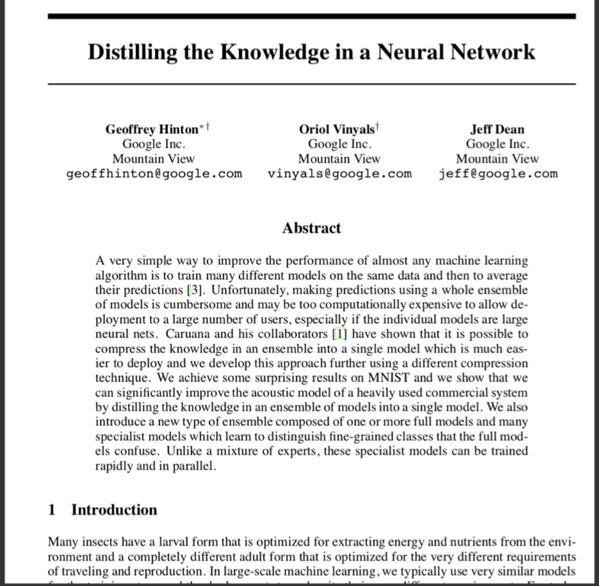
Rich Caruana
Microsoft Research
rcaruana@microsoft.com

Abstract

Currently, deep neural networks are the state of the art on problems such as speech recognition and computer vision. In this paper we empirically demonstrate that shallow feed-forward nets can learn the complex functions previously learned by deep nets and achieve accuracies previously only achievable with deep models. Moreover, in some cases the shallow nets can learn these deep functions using the same number of parameters as the original deep models. On the TIMIT phoneme recognition and CIFAR-10 image recognition tasks, shallow nets can be trained that perform similarly to complex, well-engineered, deeper convolutional models.

1 Introduction

You are given a training set with 1M labeled points. When you train a shallow neural net with one fully connected feed-forward hidden layer on this data you obtain 86% accuracy on test data. When you train a deeper neural net as in [1] consisting of a convolutional layer, pooling layer, and three fully connected feed-forward layers on the same data you obtain 91% accuracy on the same test set.



Distilling the Knowledge in a Neural Network

Geoffrey Hinton[*†]
Google Inc.
Mountain View
geoffhinton@google.com

Oriol Vinyals[†]
Google Inc.
Mountain View
vinyals@google.com

Jeff Dean
Google Inc.
Mountain View
jeff@google.com

Abstract

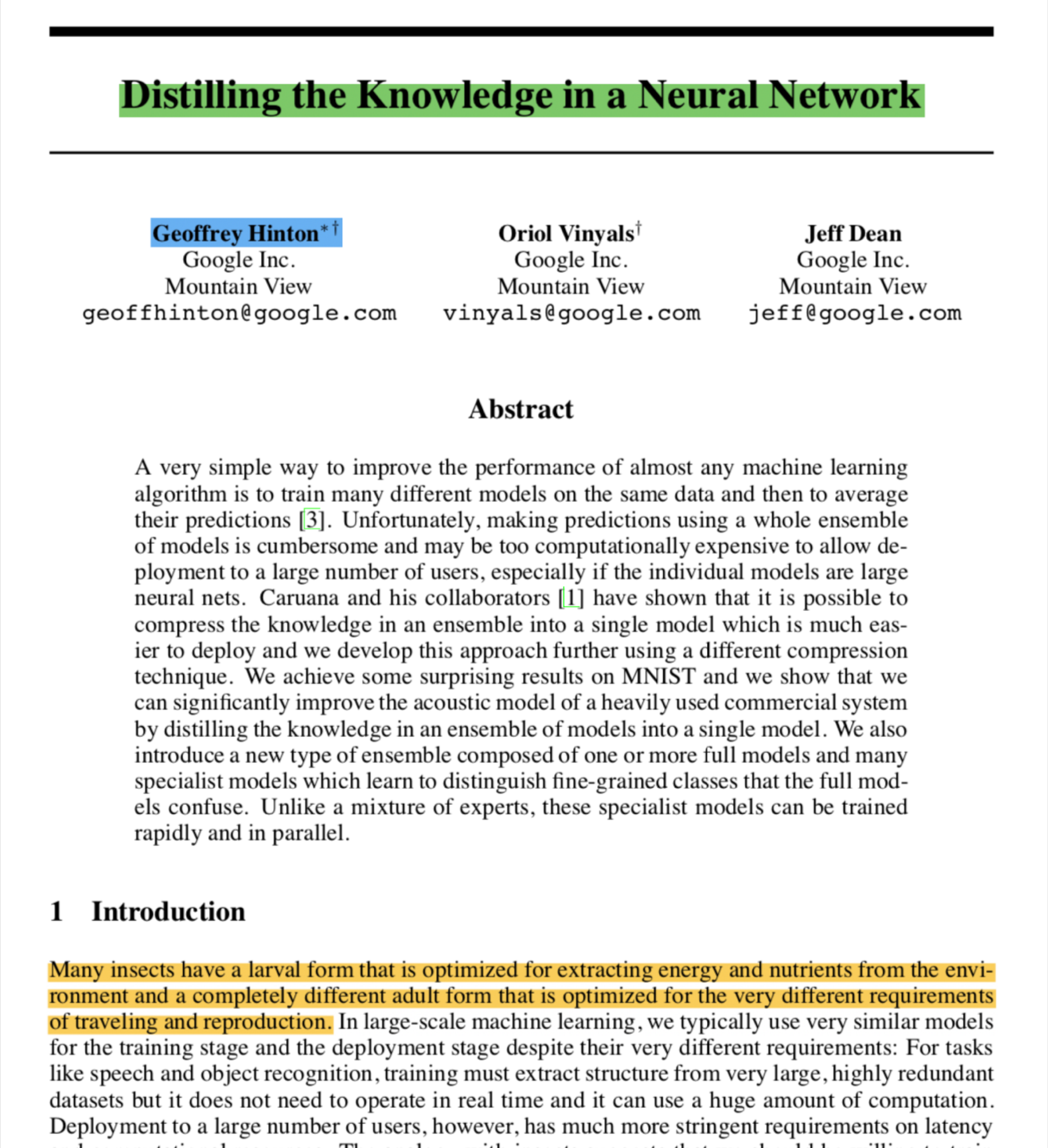
A very simple way to improve the performance of almost any machine learning algorithm is to train many different models on the same data and then to average their predictions [3]. Unfortunately, making predictions using a whole ensemble of models is cumbersome and may be too computationally expensive to allow deployment to a large number of users, especially if the individual models are large neural nets. Caruana and his collaborators [1] have shown that it is possible to compress the knowledge in an ensemble into a single model which is much easier to deploy and we develop this approach further using a different compression technique. We achieve some surprising results on MNIST and we show that we can significantly improve the acoustic model of a heavily used commercial system by distilling the knowledge in an ensemble of models into a single model. We also introduce a new type of ensemble composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. Unlike a mixture of experts, these specialist models can be trained rapidly and in parallel.

1 Introduction

Many insects have a larval form that is optimized for extracting energy and nutrients from the environment and a completely different adult form that is optimized for the very different requirements of traveling and reproduction. In large-scale machine learning, we typically use very similar models

How to get all the citations

- Give your paper a cool title
- Begin with a tenuous link to biology
- Be Geoff Hinton



Distilling the Knowledge in a Neural Network

Geoffrey Hinton[∗†]
Google Inc.
Mountain View
geoffhinton@google.com

Oriol Vinyals[†]
Google Inc.
Mountain View
vinyals@google.com

Jeff Dean
Google Inc.
Mountain View
jeff@google.com

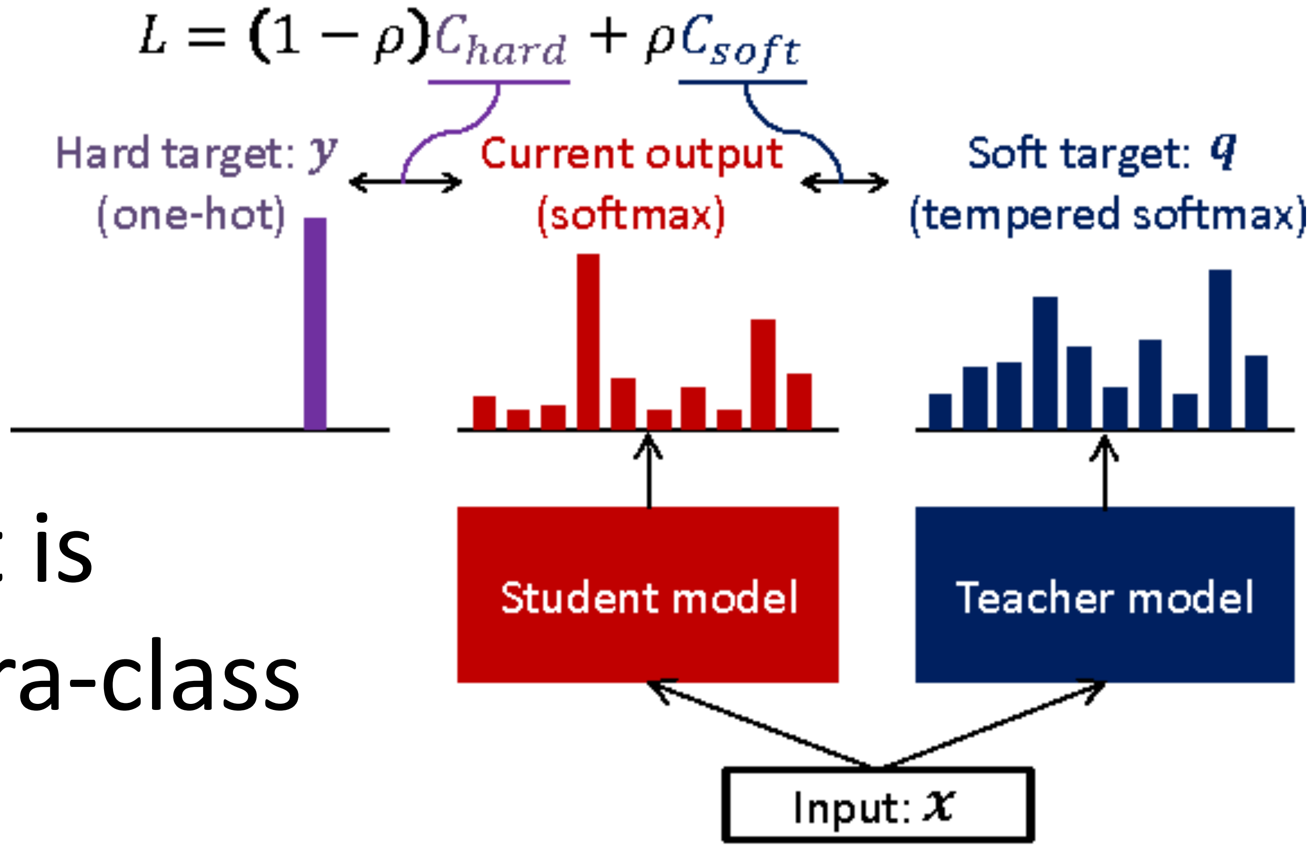
Abstract

A very simple way to improve the performance of almost any machine learning algorithm is to train many different models on the same data and then to average their predictions [3]. Unfortunately, making predictions using a whole ensemble of models is cumbersome and may be too computationally expensive to allow deployment to a large number of users, especially if the individual models are large neural nets. Caruana and his collaborators [1] have shown that it is possible to compress the knowledge in an ensemble into a single model which is much easier to deploy and we develop this approach further using a different compression technique. We achieve some surprising results on MNIST and we show that we can significantly improve the acoustic model of a heavily used commercial system by distilling the knowledge in an ensemble of models into a single model. We also introduce a new type of ensemble composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. Unlike a mixture of experts, these specialist models can be trained rapidly and in parallel.

1 Introduction

Many insects have a larval form that is optimized for extracting energy and nutrients from the environment and a completely different adult form that is optimized for the very different requirements of traveling and reproduction. In large-scale machine learning, we typically use very similar models for the training stage and the deployment stage despite their very different requirements: For tasks like speech and object recognition, training must extract structure from very large, highly redundant datasets but it does not need to operate in real time and it can use a huge amount of computation. Deployment to a large number of users, however, has much more stringent requirements on latency

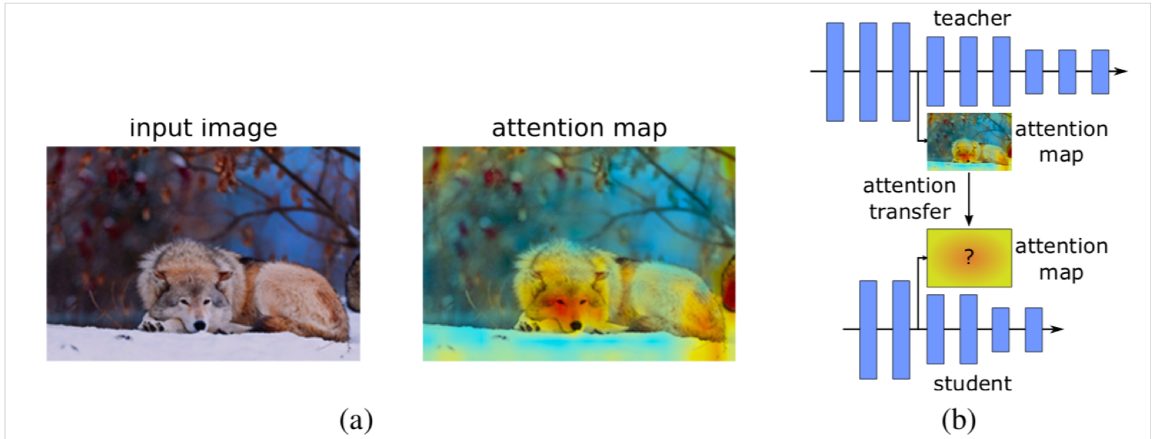
Knowledge Distillation



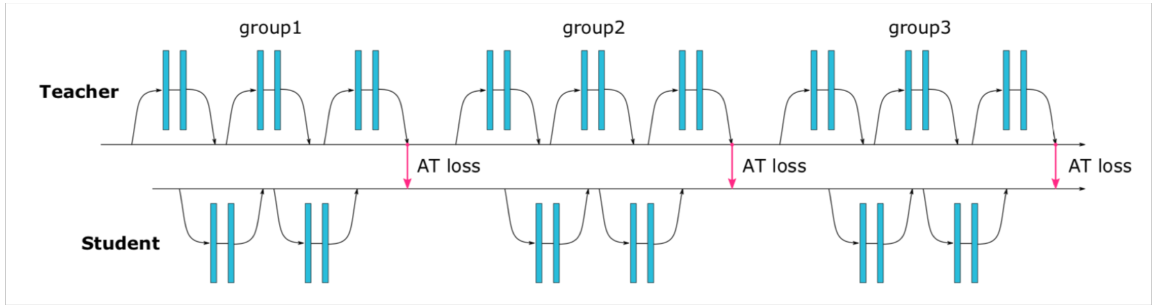
The student is learning intra-class information

https://www.semanticscholar.org/paper/Domain-adaptation-of-DNN-acoustic-models-using-Asami-Masumura/76697e29b72b1980beb54784226ab7235ce298aa

Attention Transfer

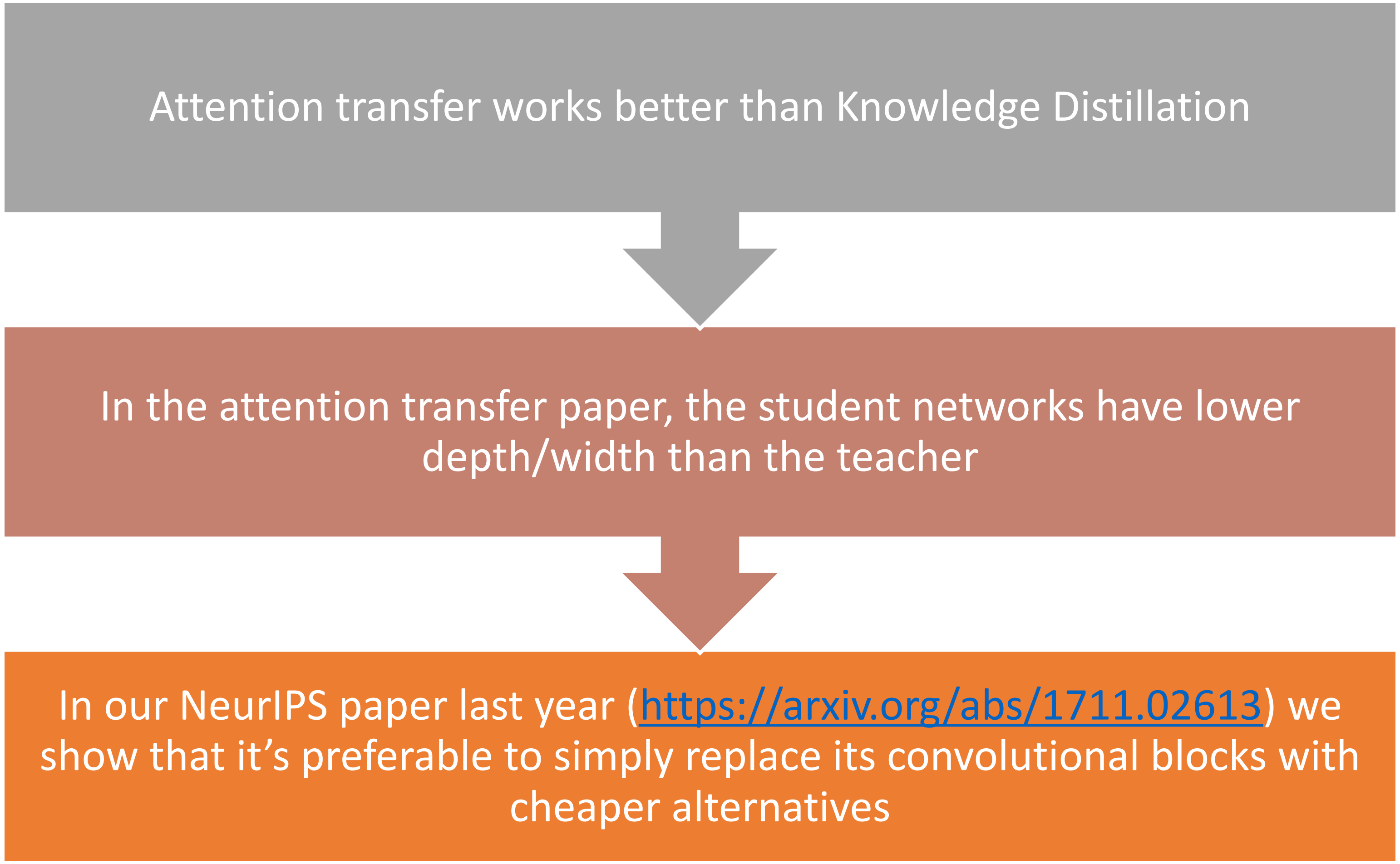


The student is learning where to look



https://arxiv.org/pdf/1612.03928.pdf

Shameless Self-Promotion Slide



Attention transfer works better than Knowledge Distillation

In the attention transfer paper, the student networks have lower depth/width than the teacher

In our NeurIPS paper last year (https://arxiv.org/abs/1711.02613) we show that it's preferable to simply replace its convolutional blocks with cheaper alternatives

Summary

- Sensible design choices allow for substantial compression
- Avoid pruning ☺
- Attention Transfer works well for network distillation. Use replacement blocks
- Be Geoff Hinton

Thanks!

Any questions?