New recognition architectures

The VGG pattern

- Every convolution is 3x3, padded by 1
- Every convolution followed by ReLU
- ConvNet is divided into “stages”
  - Layers within a stage: no subsampling
  - Subsampling by 2 at the end of each stage
- Layers within stage have same number of channels
- Every subsampling → double the number of channels

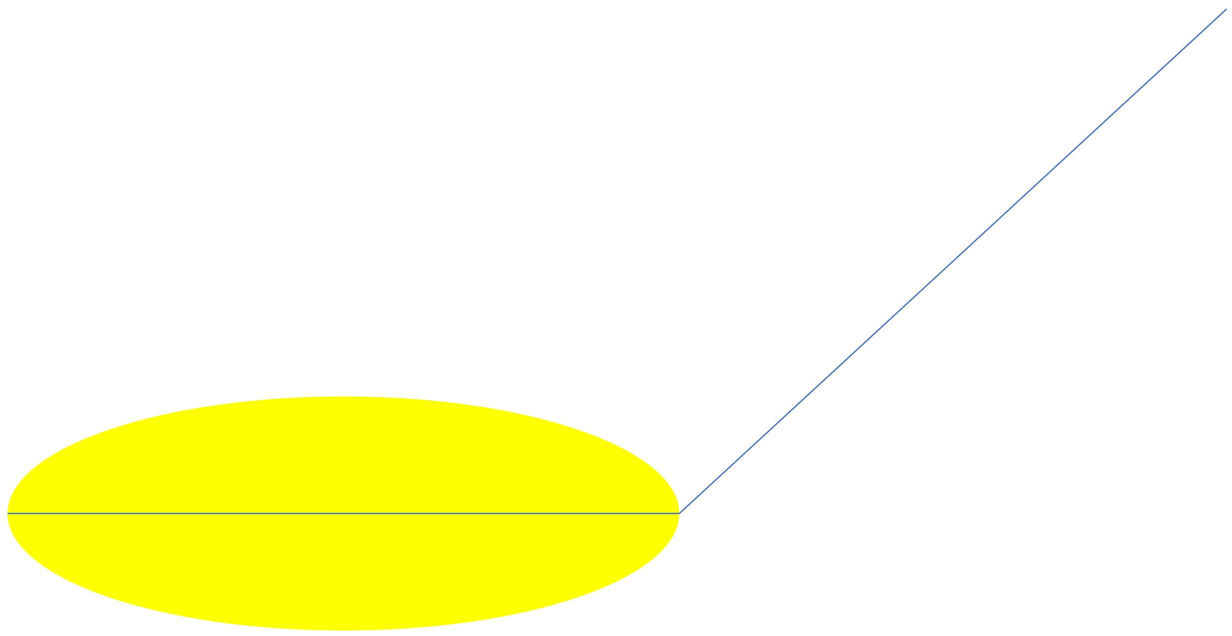
Challenges in training: exploding / vanishing gradients

- Vanishing / exploding gradients

$$\frac{\partial z}{\partial z_i} = \frac{\partial z}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_{i+1}}{\partial z_i}$$

  - If each term is (much) greater than 1 → *explosion of gradients*
  - If each term is (much) less than 1 → *vanishing gradients*

Challenges in training: dependence on init



Solutions

- Careful init
- Batch normalization
- Residual connections

Careful initialization

- Key idea: want variance to remain approx. constant
  - Variance increases in backward pass => exploding gradient
  - Variance decreases in backward pass => vanishing gradient
- “MSRA initialization”
  - weights = Gaussian with 0 mean and variance = 2/(k*k*d)

Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. K. He, X. Zhang, S. Ren, J. Sun

How do we make better neural network architectures?

- Better optimization
- Cheaper compute requirements
- New primitives

Residual connections

- In general, gradients tend to vanish
- Key idea: allow gradients to flow unimpeded

$$z_{i+1} = f_{i+1}(z_i, w_{i+1})$$

$$\frac{\partial z_{i+1}}{\partial z_i} = \frac{\partial f_{i+1}(z_i, w_{i+1})}{\partial z_i}$$

$$\frac{\partial z}{\partial z_i} = \frac{\partial z}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_{i+1}}{\partial z_i}$$

He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Residual connections

- In general, gradients tend to vanish
- Key idea: allow gradients to flow unimpeded

$$z_{i+1} = g_{i+1}(z_i, w_{i+1}) + z_i \qquad \frac{\partial z_{i+1}}{\partial z_i} = \frac{\partial g_{i+1}(z_i, w_{i+1})}{\partial z_i} + I$$

$$\frac{\partial z}{\partial z_i} = \frac{\partial z}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_{i+1}}{\partial z_i}$$

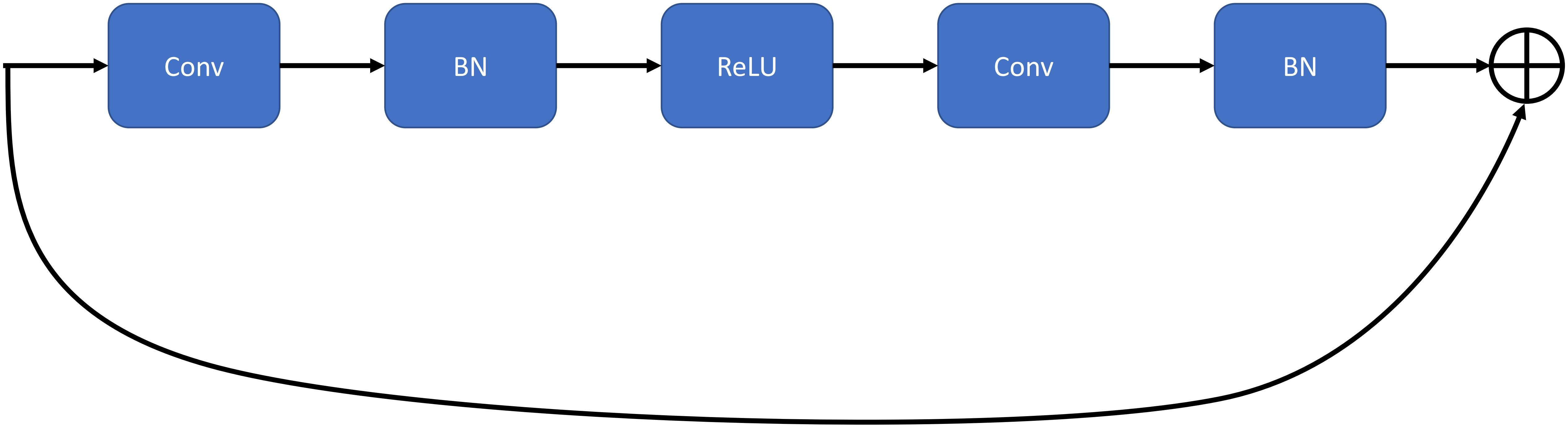
Residual connections

- Assumes all z_i have the same size
- True within a stage
- Across stages?
  - Doubling of feature channels
  - Subsampling
- Increase channels by 1x1 convolution
- Decrease spatial resolution by subsampling

$$z_{i+1} = g_{i+1}(z_i, w_{i+1}) + \mathrm{subsample}(W z_i)$$

A residual block

- Instead of single layers, have residual connections over block



Better optimization - Batch normalization

- Key idea: normalize so that each layer output has zero mean and unit variance
  - Compute mean and variance for each channel
  - Aggregate over batch
  - Subtract mean, divide by std
- Generally makes optimization landscape smoother
- Need to reconcile train and test
  - No ”batches” during test
  - After training, compute means and variances on train set and store

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. S. Ioffe, C. Szegedy. In *ICML,* 2015.

Other forms of normalization

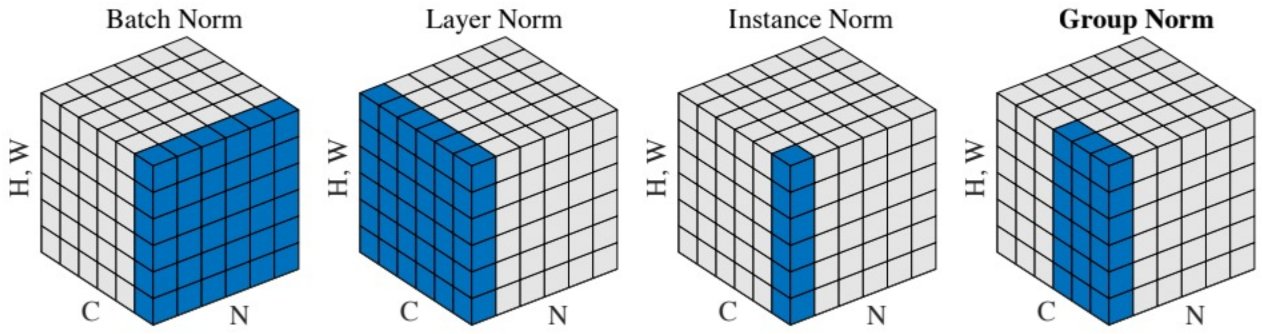


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." *arXiv preprint arXiv:1607.06450* (2016).

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Instance normalization: The missing ingredient for fast stylization." *arXiv preprint arXiv:1607.08022* (2016).

Wu, Yuxin, and Kaiming He. "Group normalization." *Proceedings of the European conference on computer vision (ECCV)*. 2018.

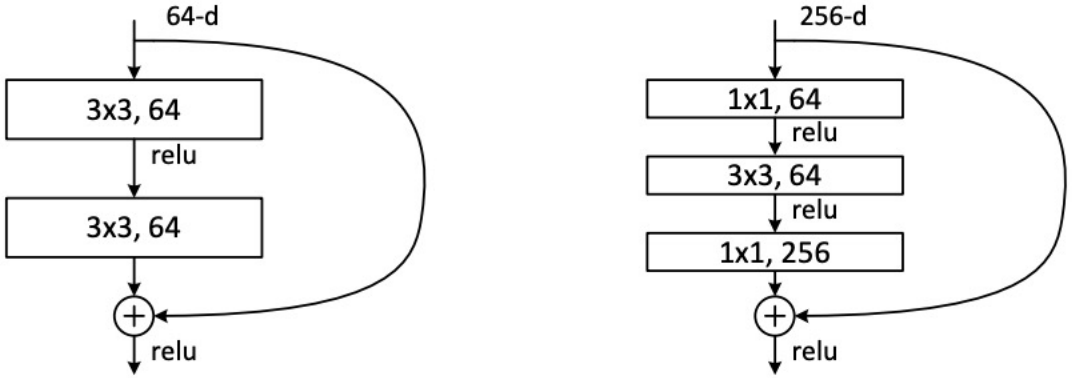
Analyzing computational complexity

- What is the computational complexity of a single convolutional layer?
  - $h \times w \times c$ input and output
  - $k \times k$ kernel
- Space:
  - Input/output: hwc
  - Filters: k^2c^2
- Time (Flops): hwk^2c^2

Reducing computational complexity

- …while maintaining accuracy?
- Multiple ways:
  - Make architecture *a priori* cheaper
  - Make *weights* and *operations* cheaper
  - Make inference adaptive

Better compute-cost tradeoffs: bottleneck layers

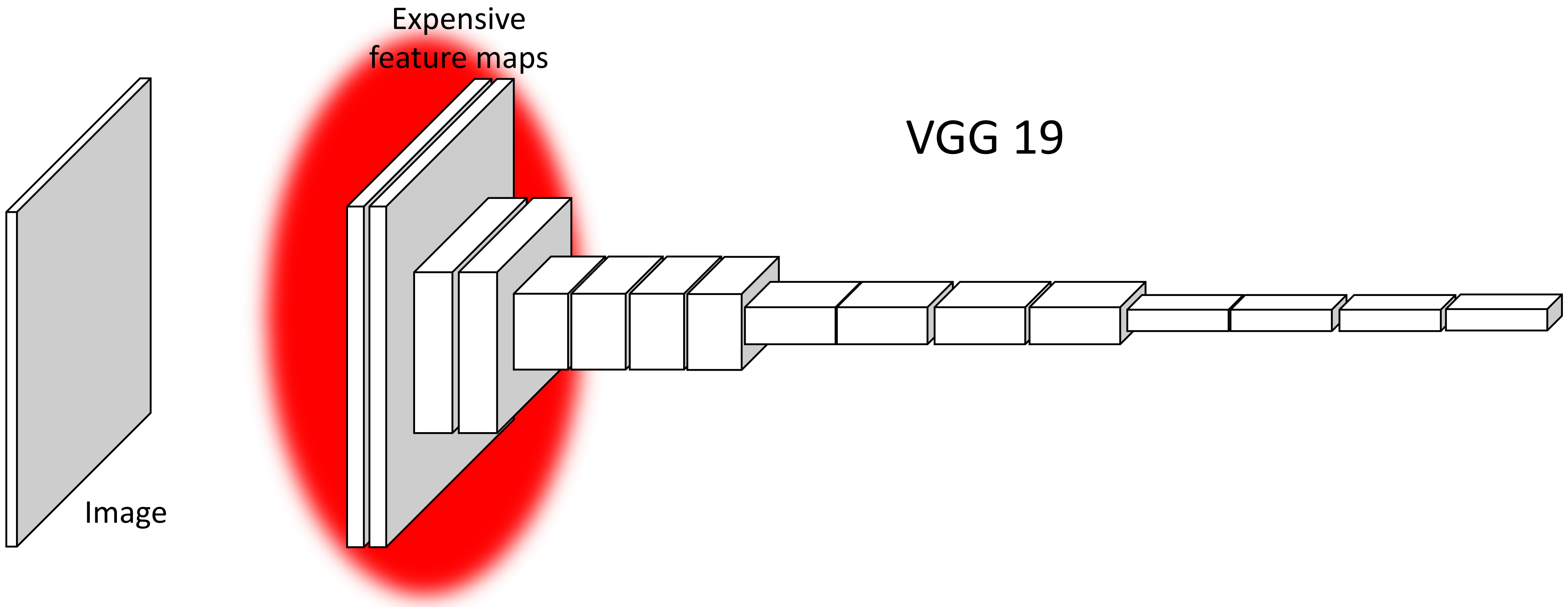


Bottleneck blocks

- Problem: When channels increases, 3x3 convolutions introduce many parameters
  - $3 \times 3 \times c^2$
- Key idea: use 1x1 to project to lower dimensionality, do convolution, then come back
  - $c \times d \ + \ 3 \times 3 \times d^2 \ + \ d \times c$

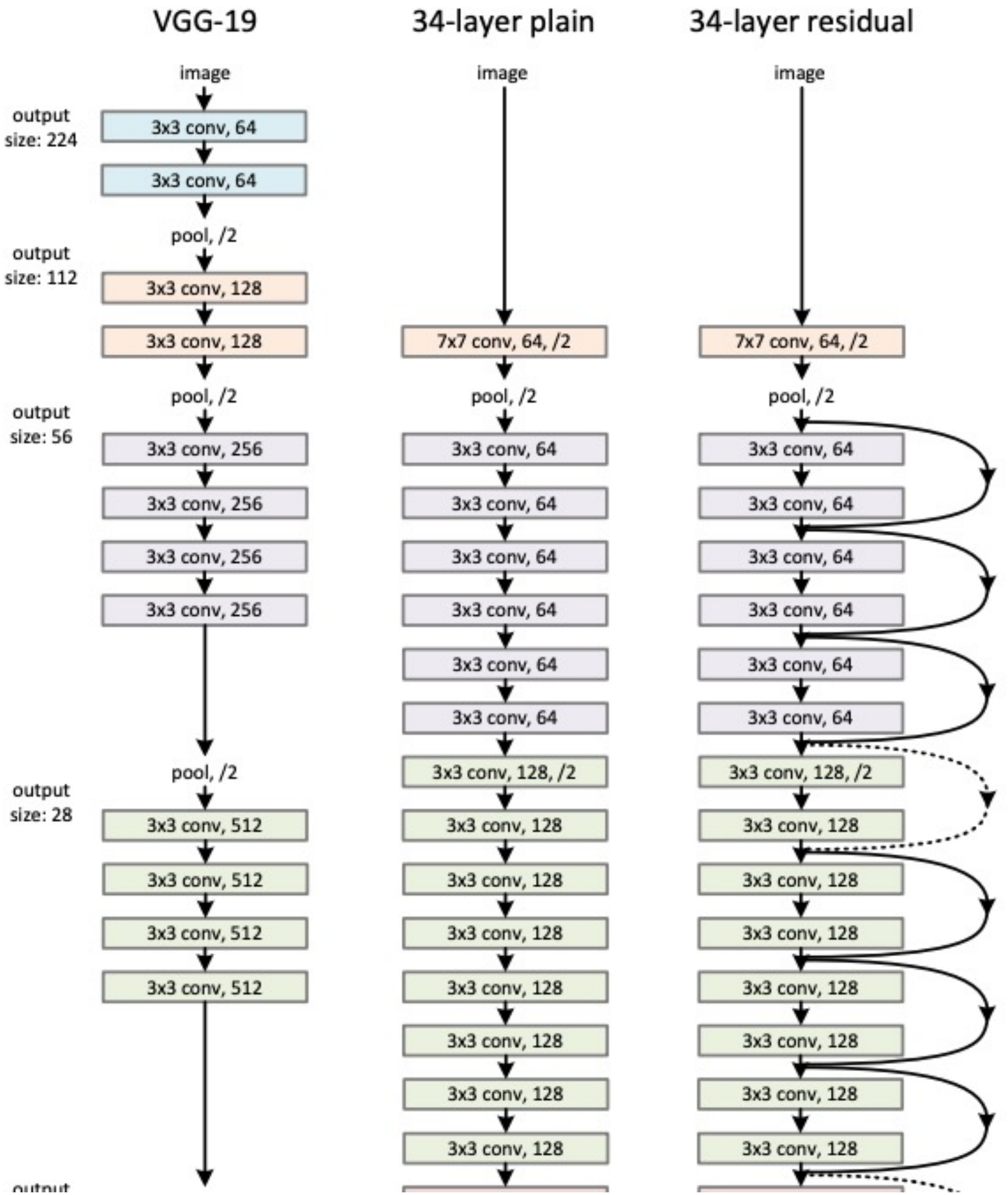
Other architectural changes

- Biggest memory consumption: large feature maps



Other architectural changes

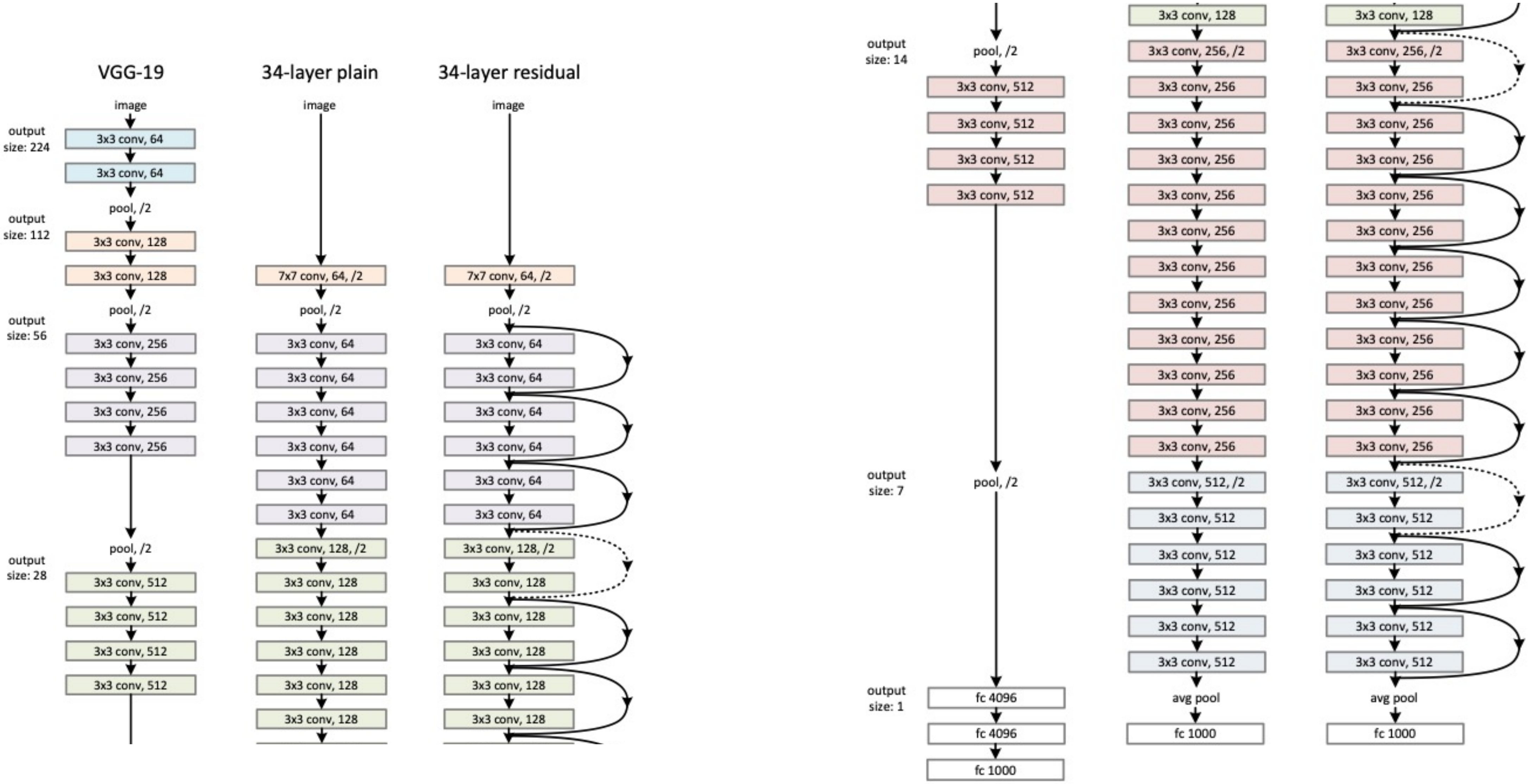
- Biggest memory consumption: large feature maps
- Simple solution: reduce resolution early



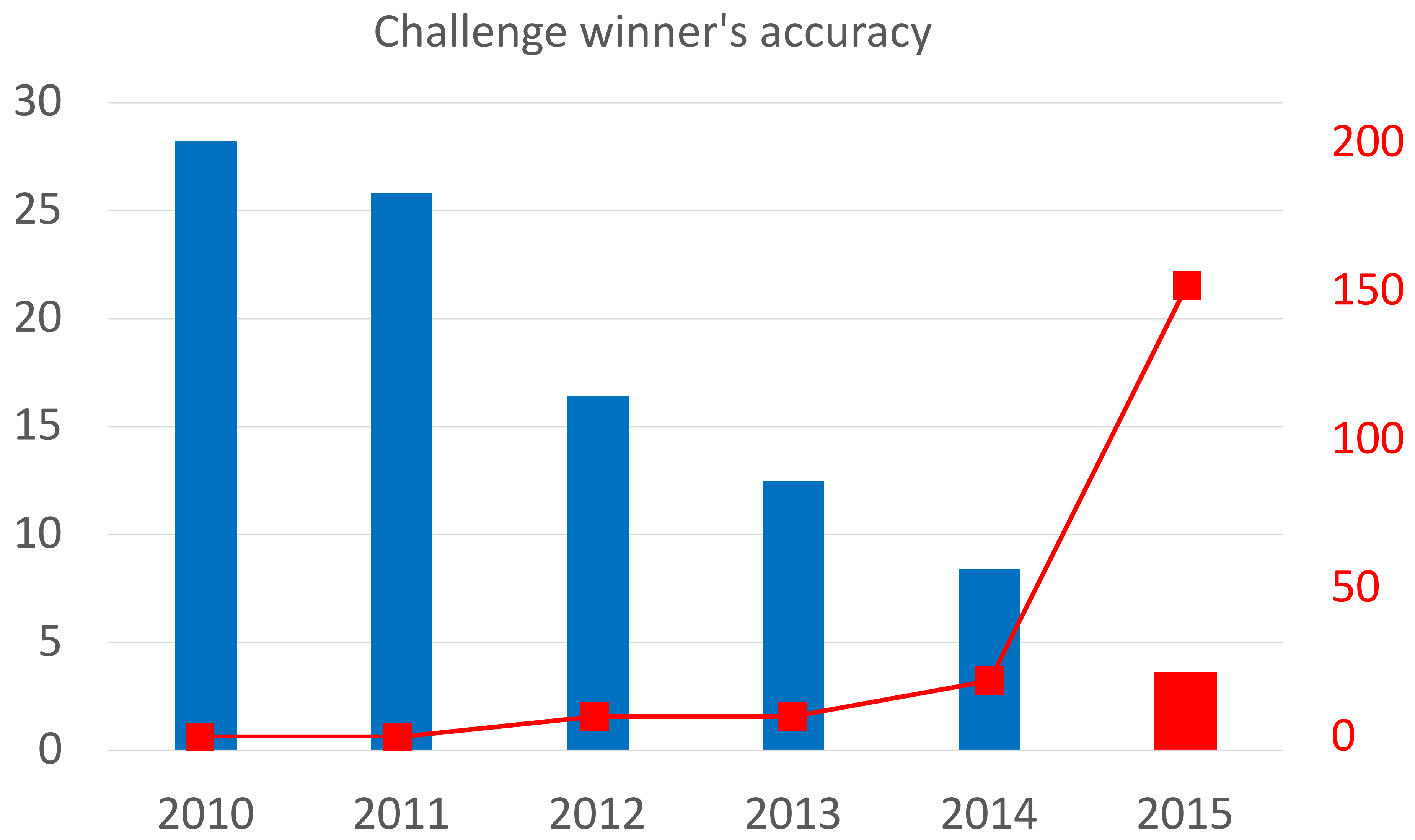
The ResNet pattern

- Decrease resolution substantially in first layer
  - Reduces memory consumption due to intermediate outputs
- Divide into stages
  - maintain resolution, channels in each stage
  - halve resolution, double channels between stages
- Divide each stage into residual blocks
- At the end, compute average value of each channel to feed linear classifier

The ResNet Pattern

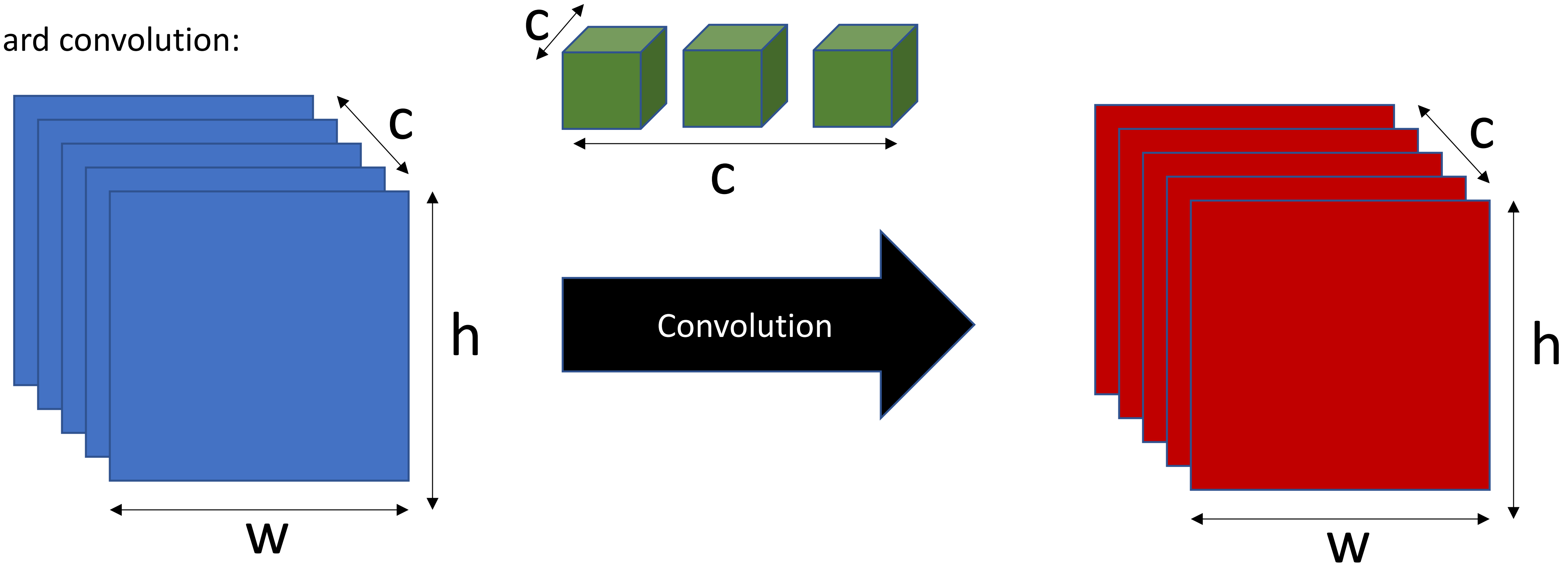


Putting it all together - Residual networks



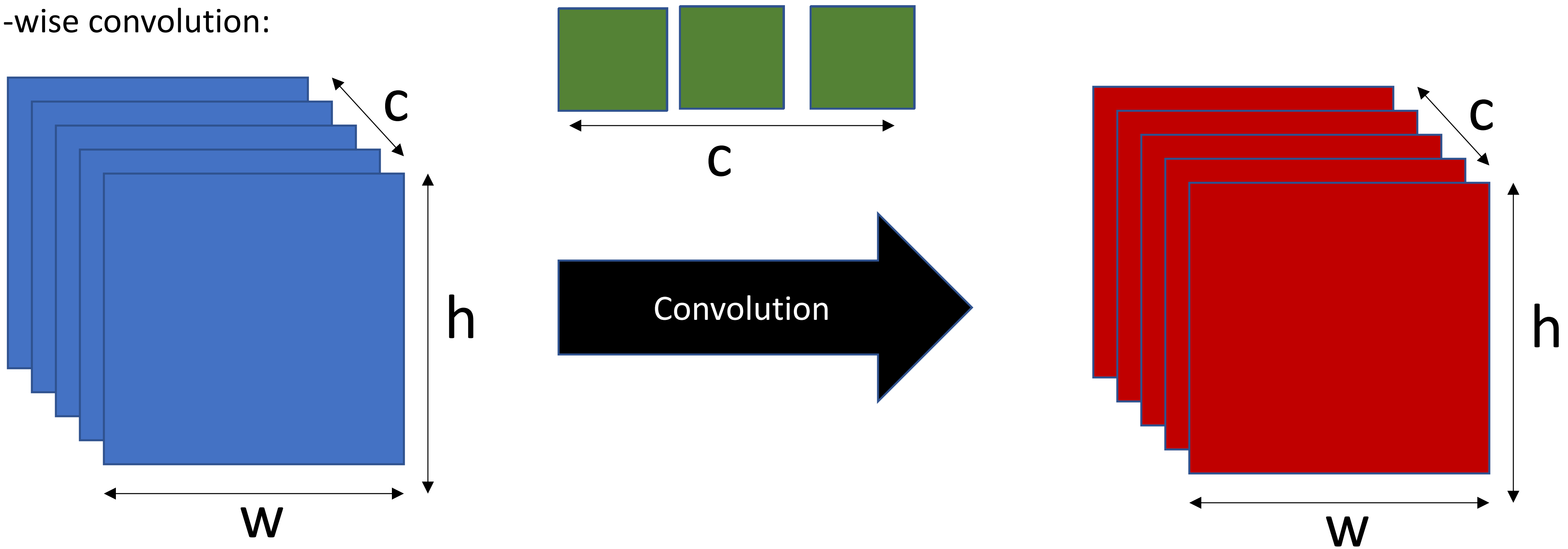
Better compute-cost tradeoffs: Other kinds of convolution

Standard convolution:



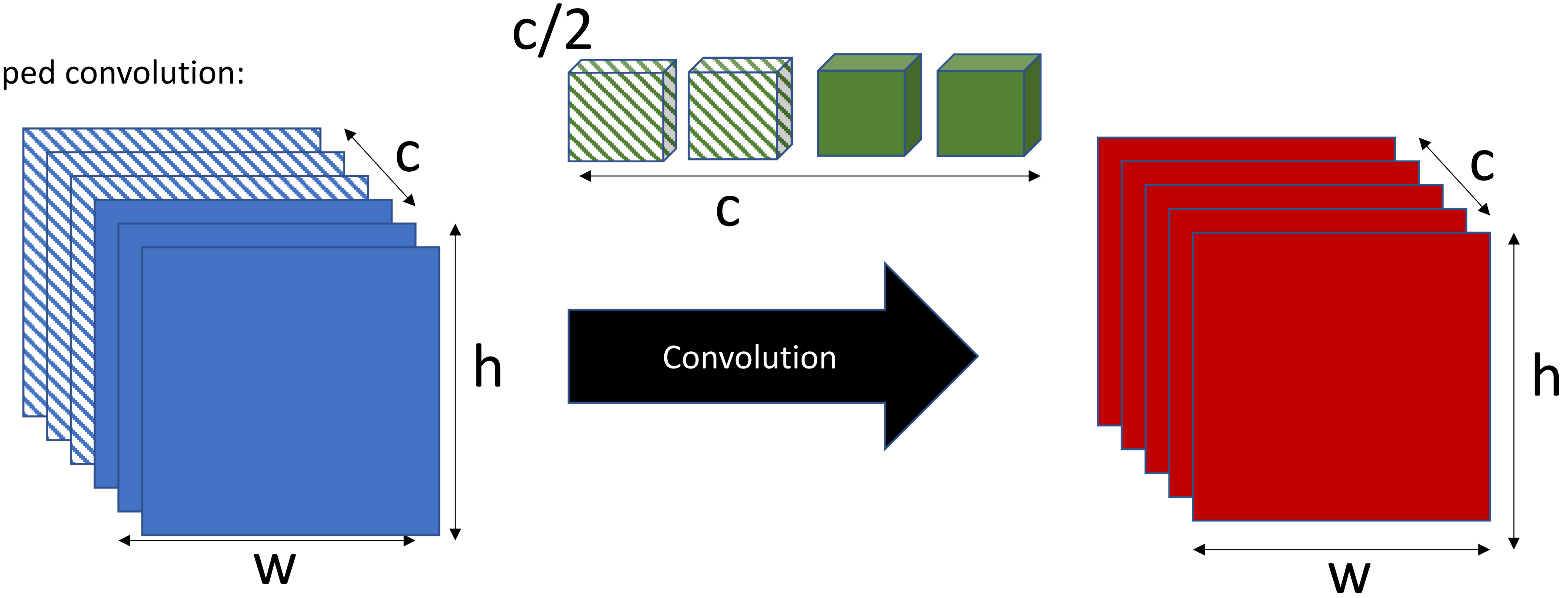
Better compute-cost tradeoffs: Other kinds of convolution

Depth-wise convolution:



Better compute-cost tradeoffs: Other kinds of convolution

Grouped convolution:



Cheaper convolutional blocks

- Standard convolution:
  - Each filter operates on all channels
  - Single $k \times k$ filter operating on c channels producing one output channel: $k^2 c$ parameters, cost
  - c such filters: $k^2 c^2$ parameters, cost
- *Depthwise separable* convolution
  - Each filter operates on a single channel
  - c filters operating on c channels: $k^2 c$ parameters, cost
  - But each channel is independently processed
  - Add a 1x1 convolution at the end with cost c^2 : $k^2 c + c^2$ parameters

Cheaper convolutional blocks

- Depthwise separable convolutions are specific instance of more general idea: *grouped convolutions*
- Grouped convolutions in original AlexNet network
- Grouped convolution:
    - Divide input channels into g groups
    - Apply convolutional layers on each group independently
    - Concatenate

Grouped and depth-wise convolutions

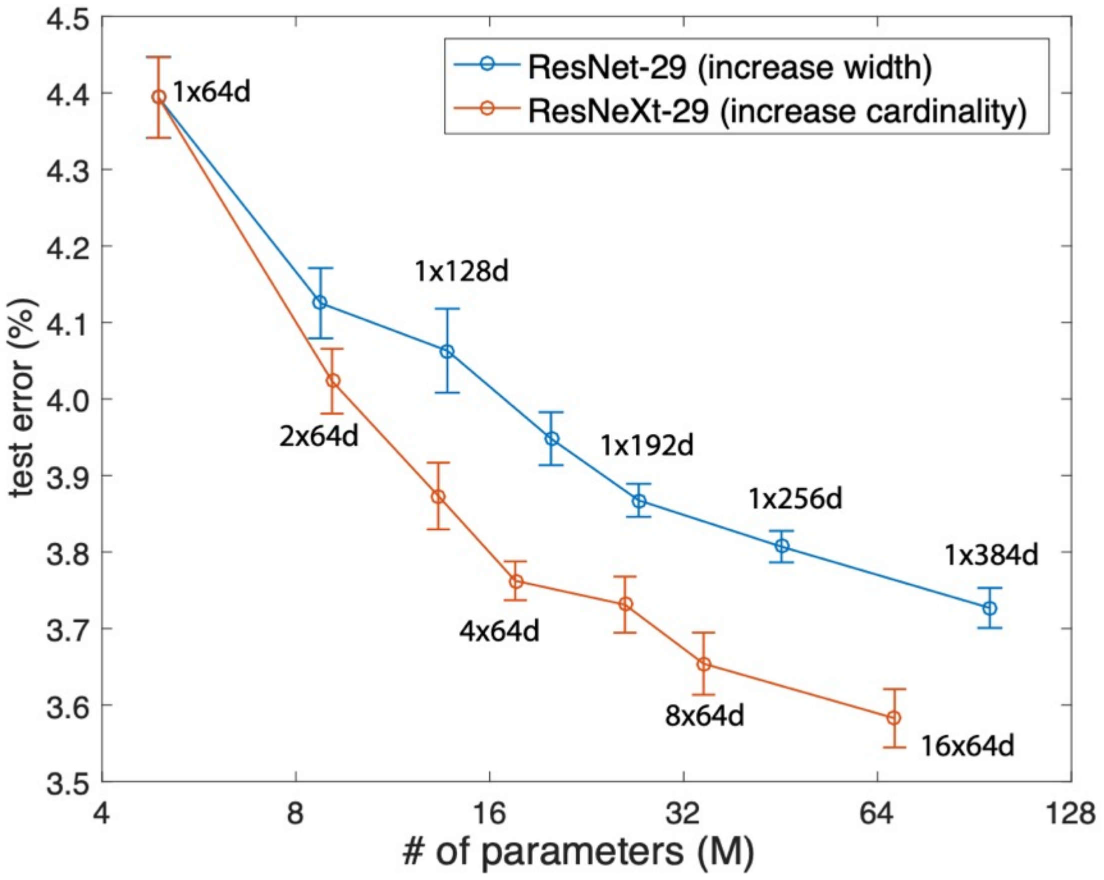


Table 4. Depthwise Separable vs Full Convolution MobileNet

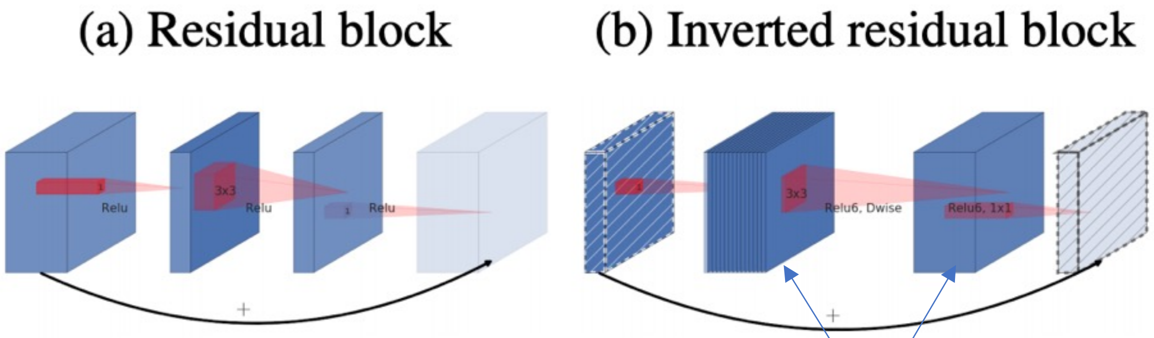
Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

Xie, Saining, et al. "Aggregated residual transformations for deep neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).

Other architectural changes

- Biggest memory consumption: large feature maps
- Simple solution (ResNet):
  - Reduce resolution drastically (/4) early
- More sophisticated changes: Inverted residuals (MobileNet v2)



Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

Dispose of these

Other kinds of connections

- DenseNets
  - Replace addition of residuals with concatenation
  - Alternative to solving vanishing gradient problem
  - Should *increase* number of parameters, but *decreases* them
  - Better re-use of features

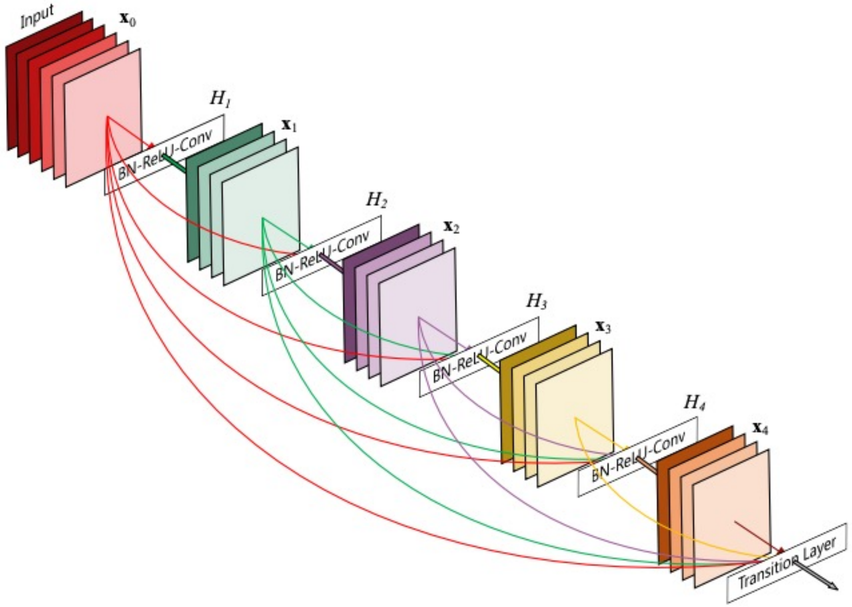


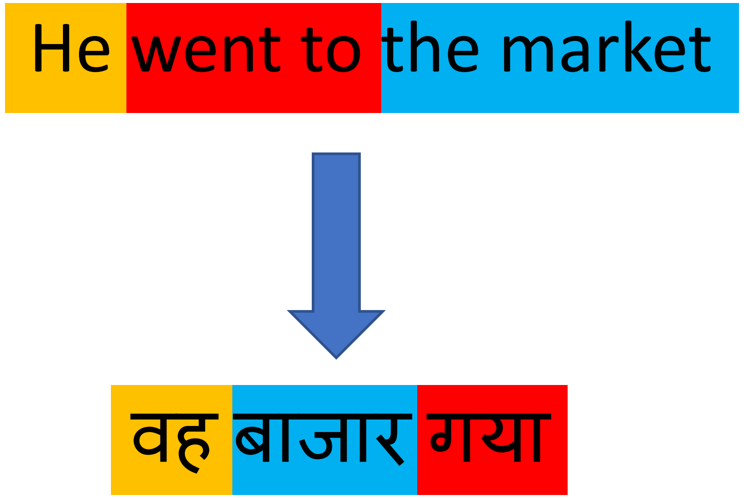
Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Huang, Gao, et al. "Densely connected convolutional networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

Transformers

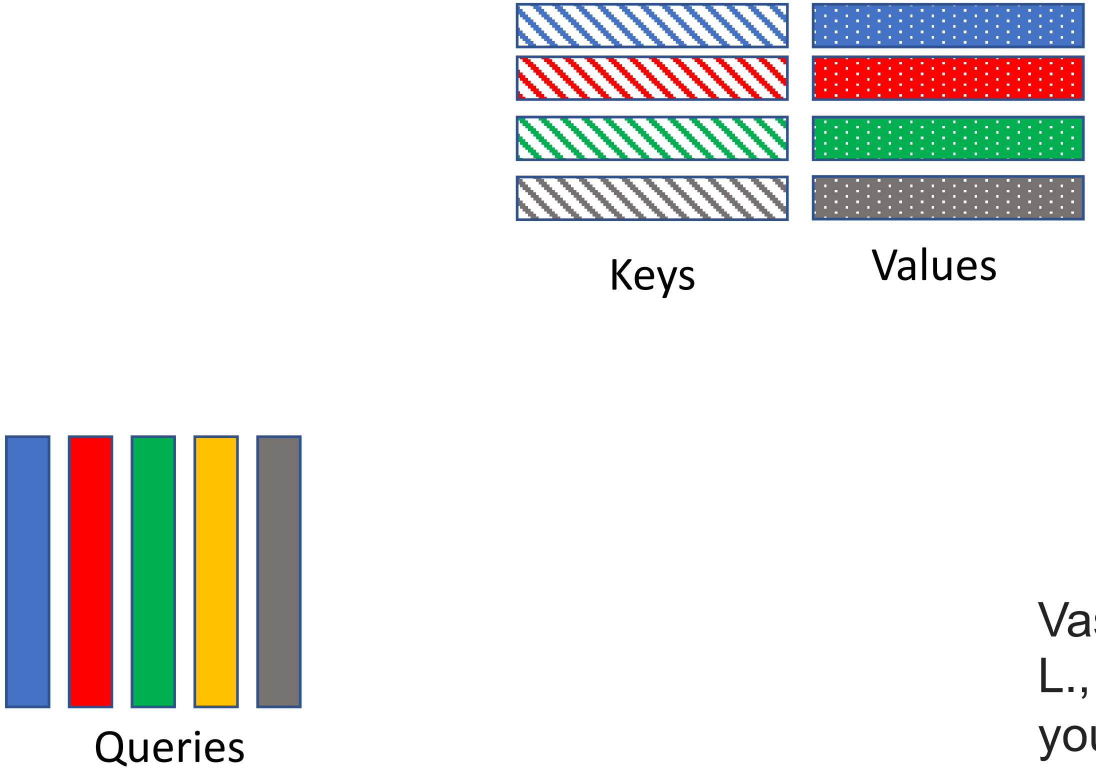
A brief dive into language generation

- Consider the machine translation problem



Attention (Transformers)

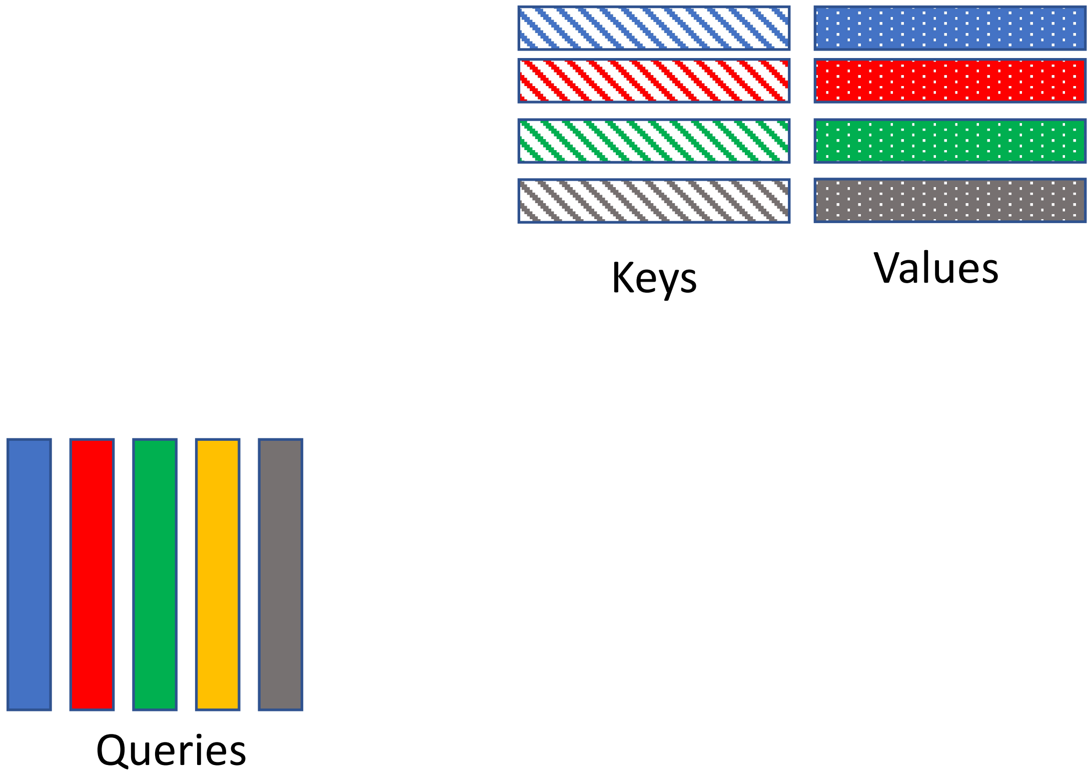
- Comes from the NLP community
- Is an approach for processing sets



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*(pp. 5998-6008).

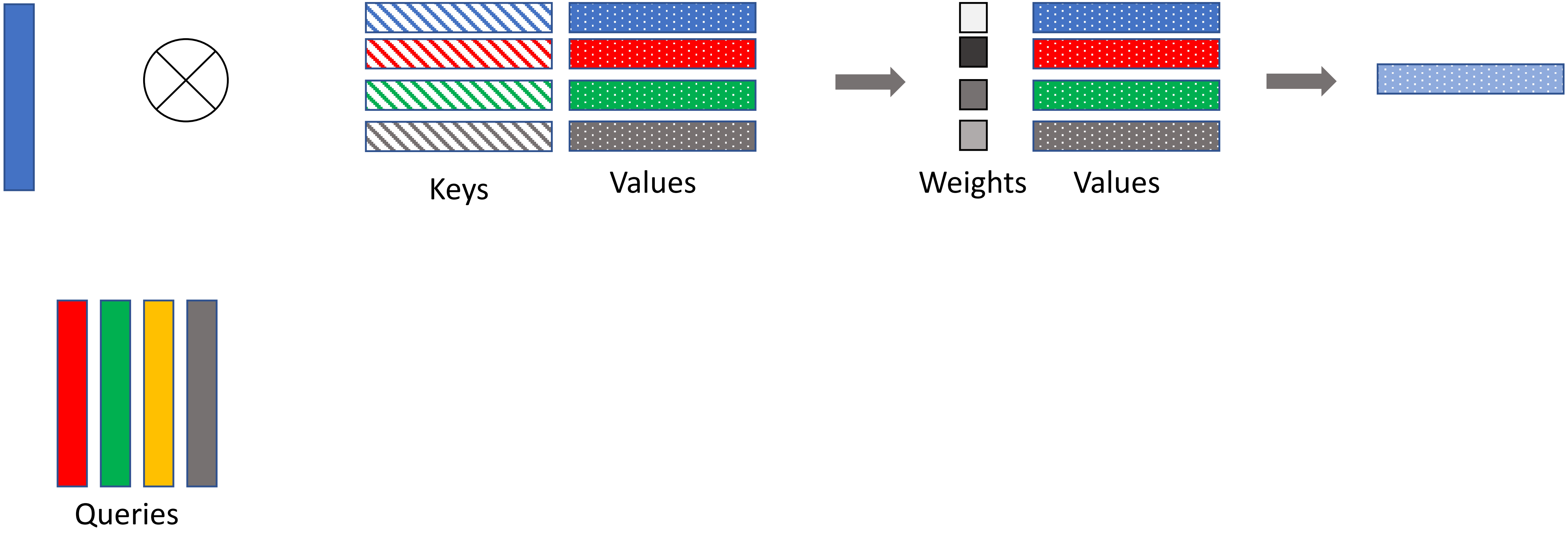
Attention (Transformers)

- Comes from the NLP community
- Is an approach for processing sets

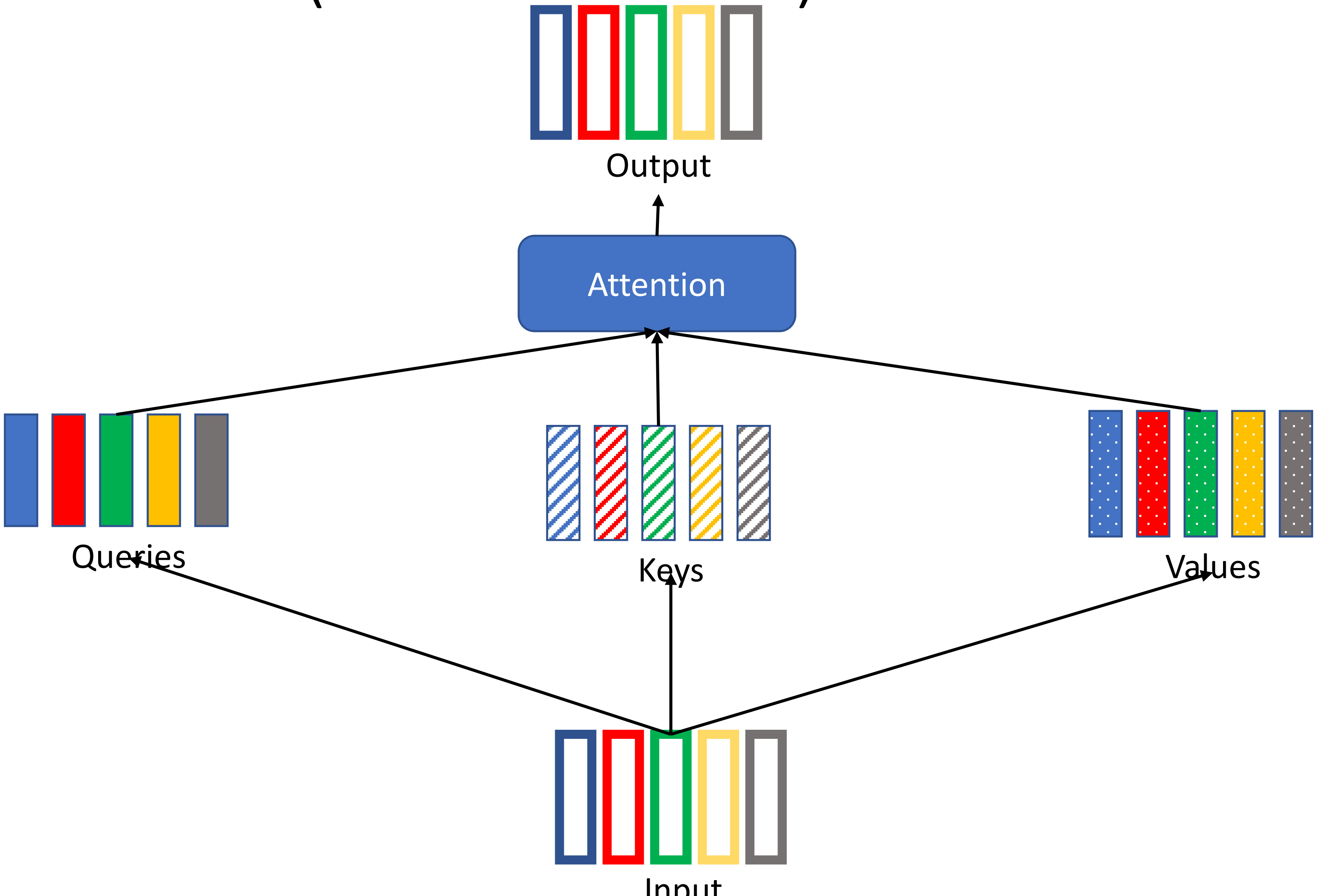


Attention (Transformers)

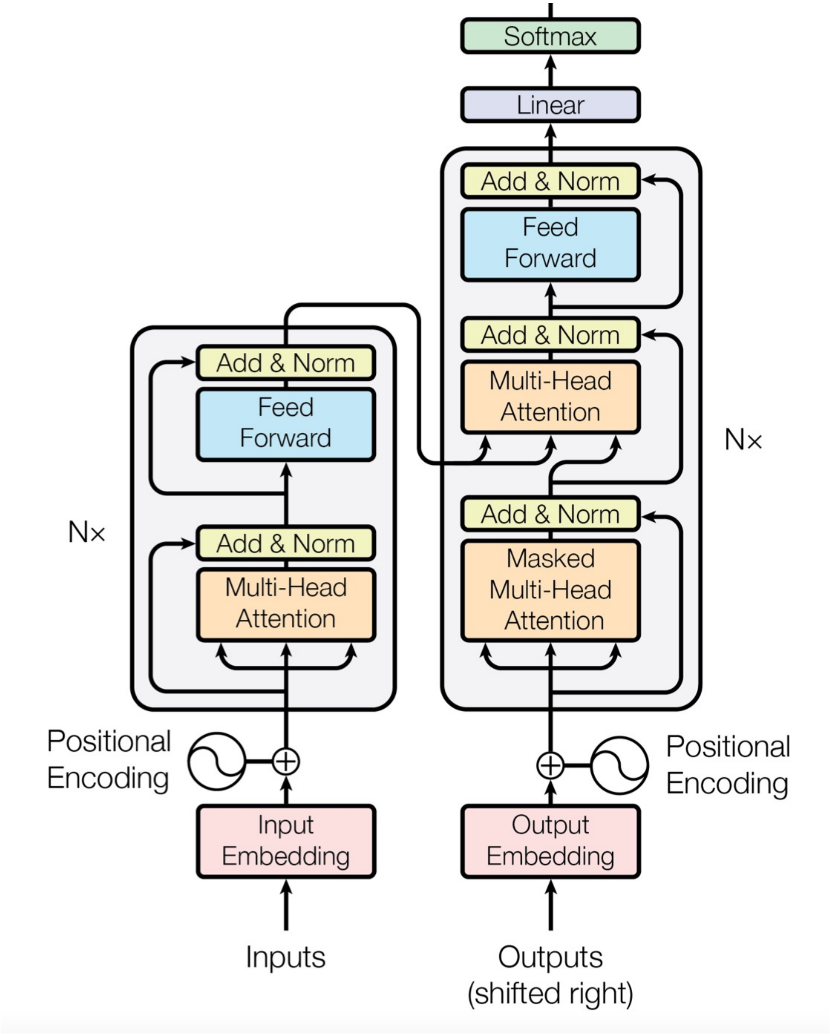
- Comes from the NLP community
- Is an approach for processing sets



Attention (Transformers)

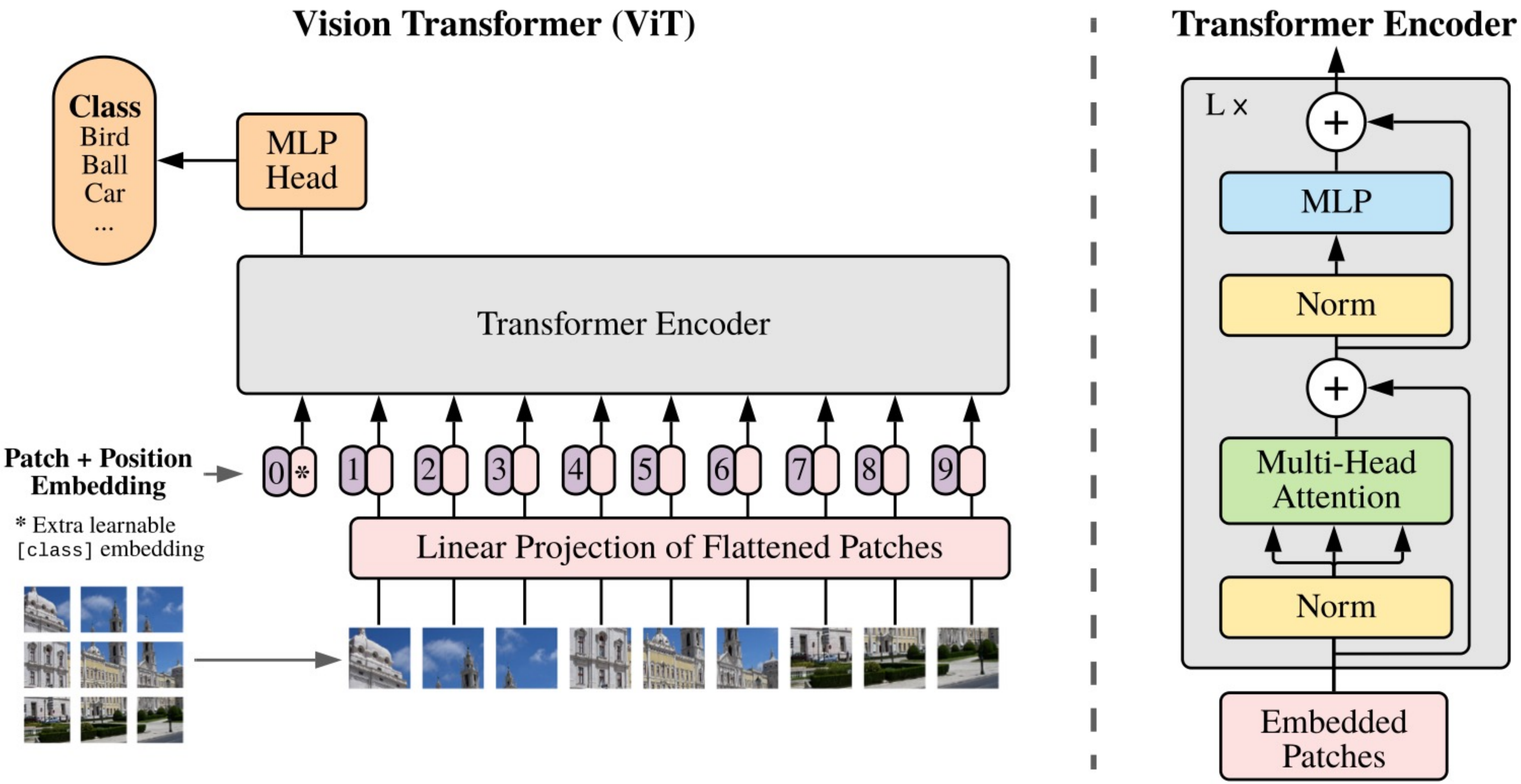


Attention (Transformers)

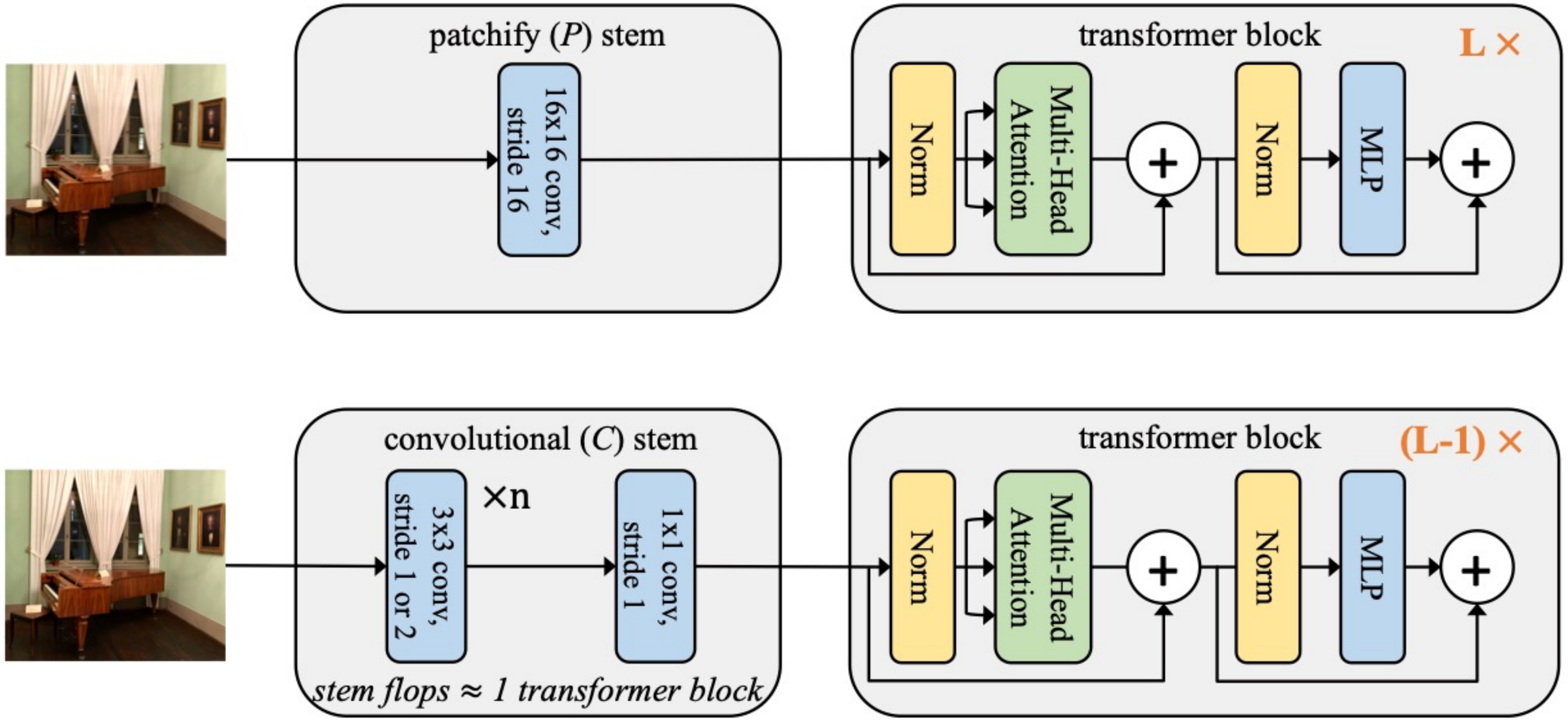


Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Transformers on images



Transformers + Convolution



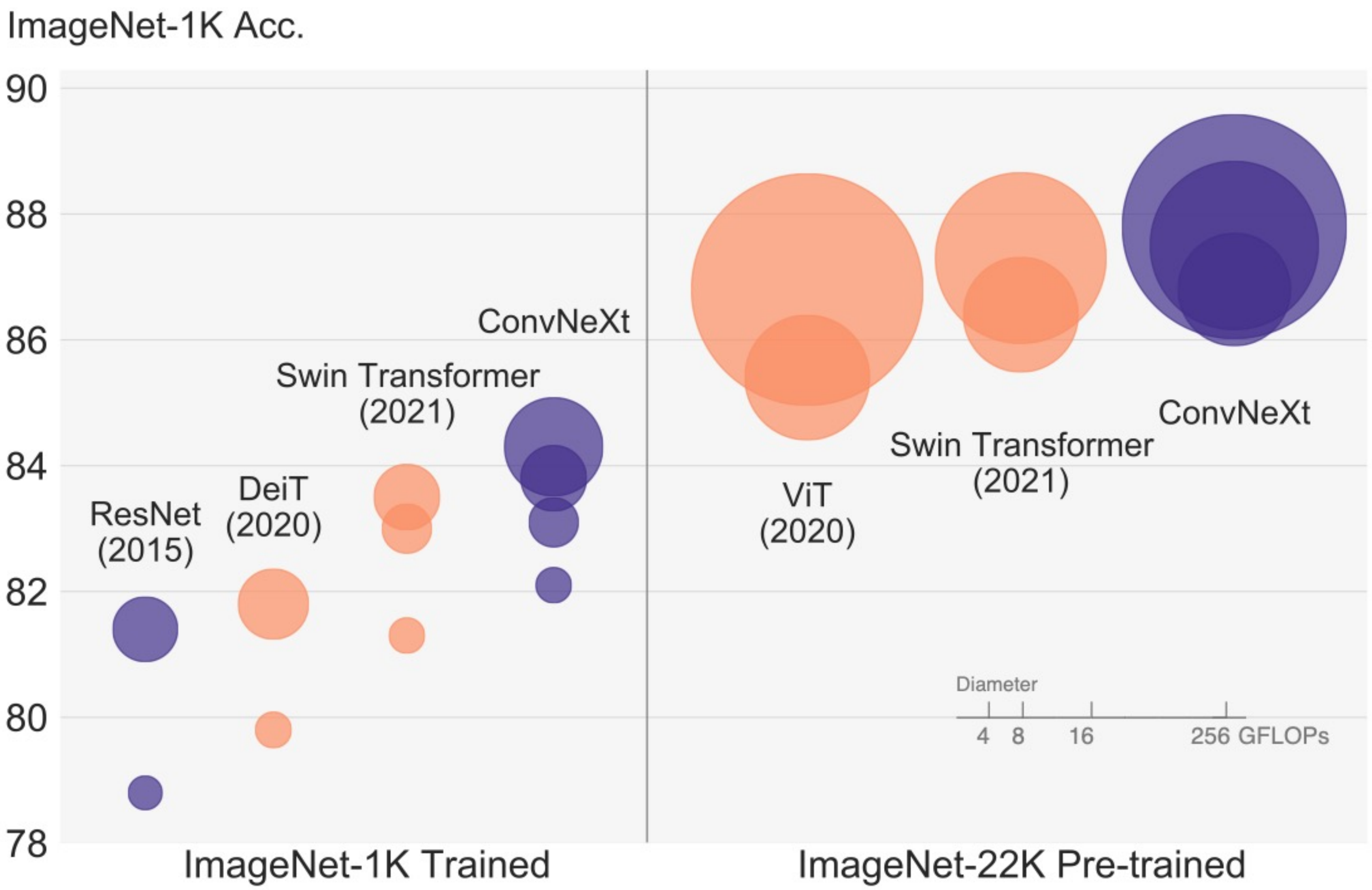
Original ViT (baseline, termed ViT$_P$):

- *Sensitive to lr and wd choice*
- *Converges slowly*
- *Works with AdamW, but not SGD*
- *Underperforms sota CNNs on ImageNet*

Ours (termed ViT$_C$, same runtime):

- ✓ *Robust to lr and wd choice*
- ✓ *Converges quickly*
- ✓ *Works with AdamW, and also SGD*
- ✓ *Outperforms sota CNNs on ImageNet*

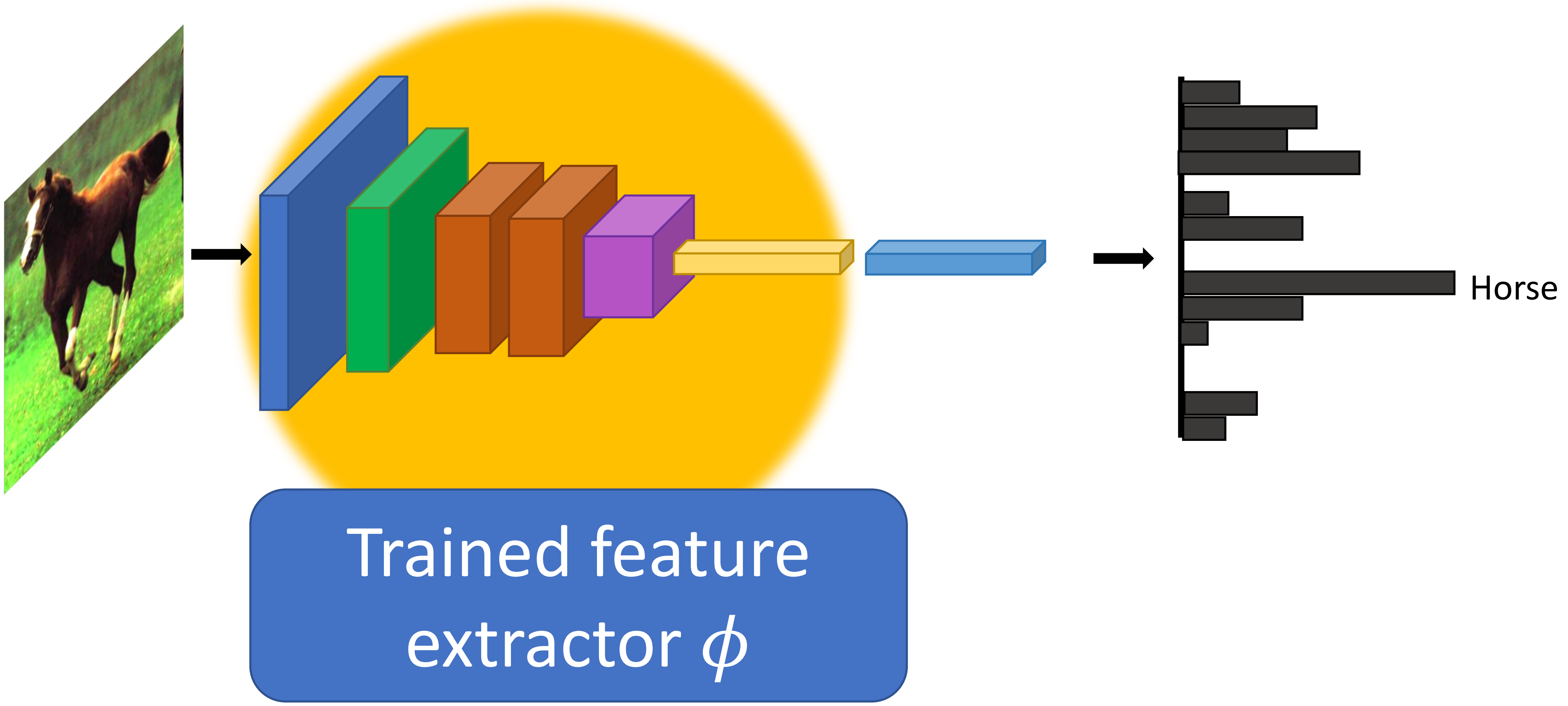
Transformers or convolution?



Liu, Zhuang, et al. "A convnet for the 2020s." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

Transfer Learning

Transfer learning with neural networks



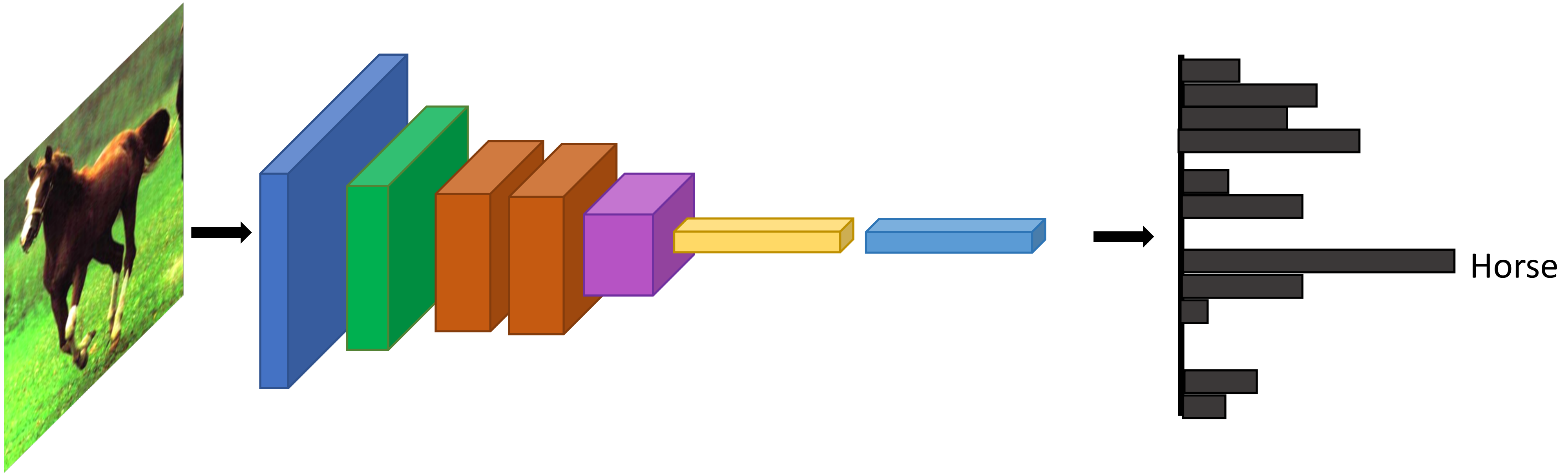
Transfer learning with convolutional networks

Dataset	Non-Convnet Method	Non-Convnet perf	Pretrained convnet + classifier	Improvement
Caltech 101	MKL	84.3	87.7	+3.4
VOC 2007	SIFT+FK	61.7	79.7	+18
CUB 200	SIFT+FK	18.8	61.0	+42.2
Aircraft	SIFT+FK	61.0	45.0	-16
Cars	SIFT+FK	59.2	36.5	-22.7

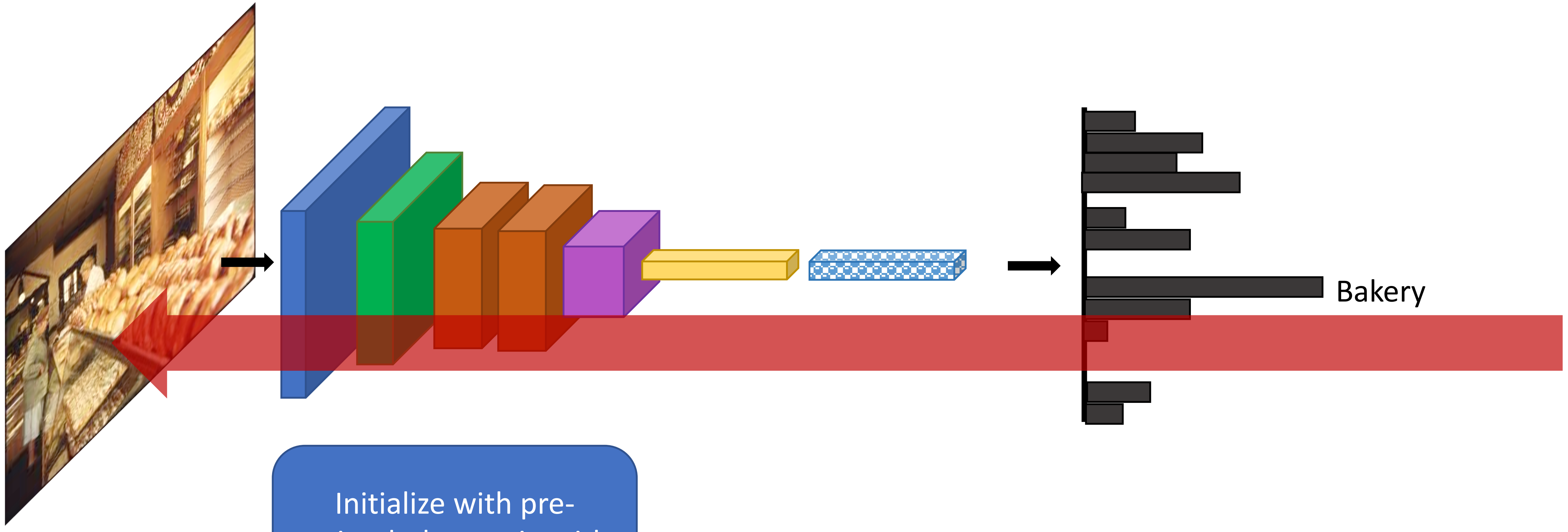
Why transfer learning?

- Availability of training data
- Computational cost
- Ability to pre-compute feature vectors and use for multiple tasks
- *Con: NO end-to-end learning*

Finetuning



Finetuning



Initialize with pre-trained, then train with low learning rate

Finetuning

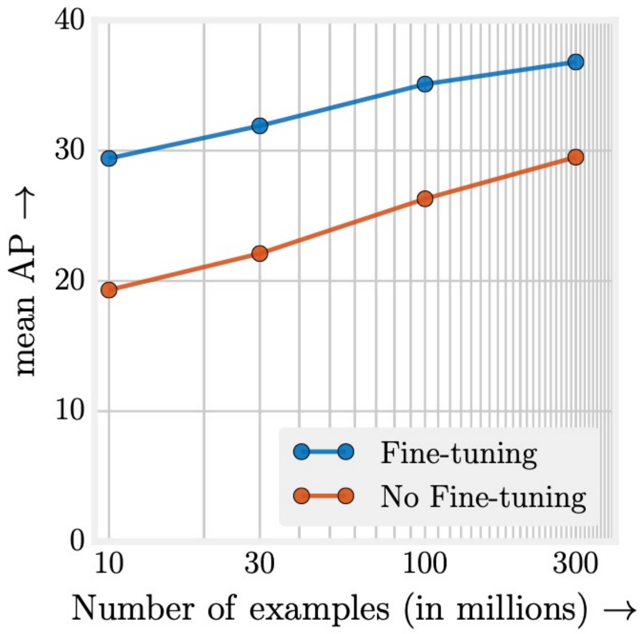
Dataset	Non-Convnet Method	Non-Convnet perf	Pretrained convnet + classifier	Finetuned convnet	Improvement
Caltech 101	MKL	84.3	87.7	88.4	+4.1
VOC 2007	SIFT+FK	61.7	79.7	82.4	+20.7
CUB 200	SIFT+FK	18.8	61.0	70.4	+51.6
Aircraft	SIFT+FK	61.0	45.0	74.1	+13.1
Cars	SIFT+FK	59.2	36.5	79.8	+20.6

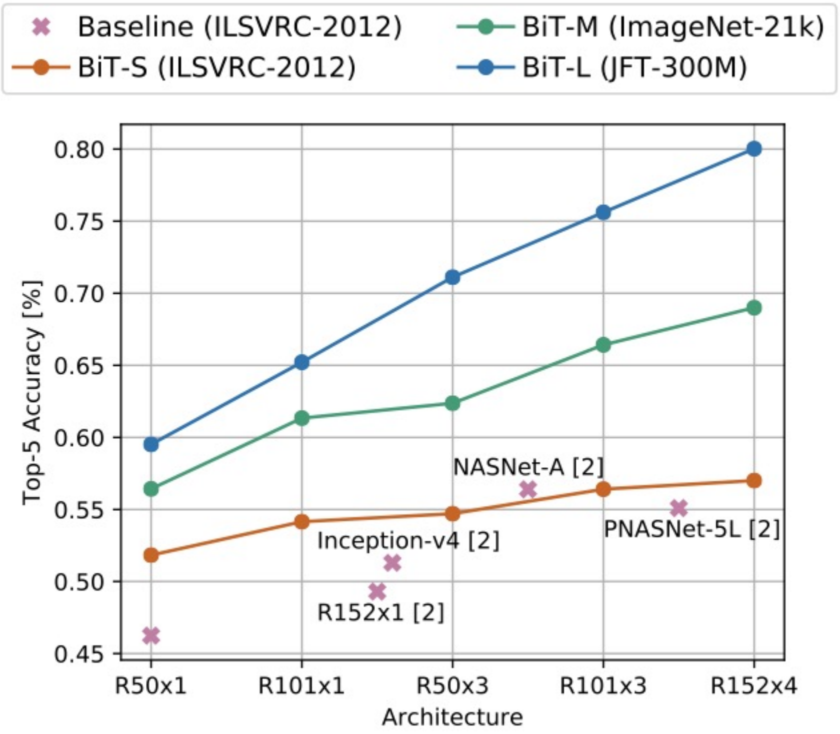
What impacts transfer accuracies?

- Relationship between pre-training and target task?
- Unclear: sometimes transfer works across very different domains
  - E.g., ImageNet -> Satellite images
- Very limited work on understanding this

What impacts transfer accuracy?

- Size of the pre-training dataset
- Size of the model
- Bigger is better





1. Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., & Houlsby, N. (2020). Big transfer (bit): General visual representation learning. In *Computer Vision–ECCV 2020: 16th European Conferenc* *Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16* (pp. 491-507). Springer International Publishing.
2. Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision* (pp. 843-852).

Concerns about big transfer

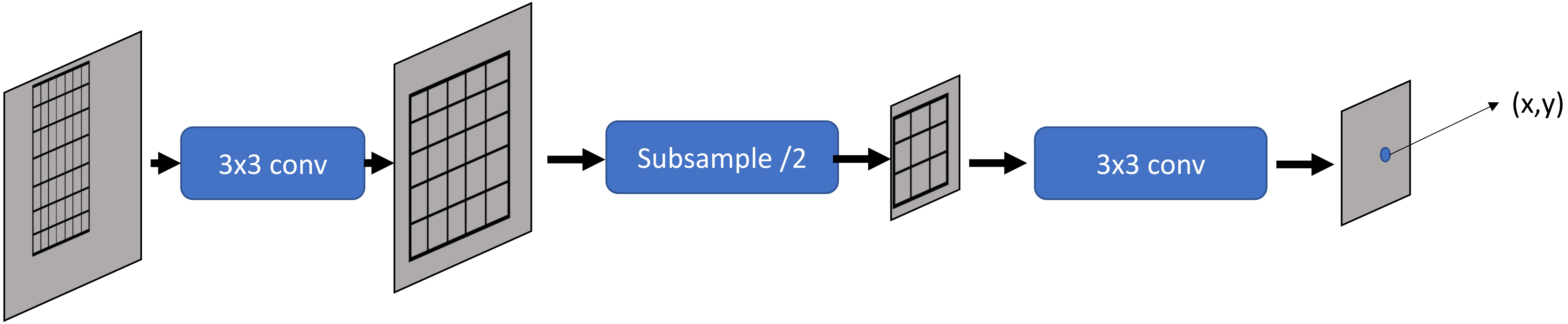
- Opaque datasets?
- Uncurated datasets?
- Bias in the datasets?
  - Do biased datasets affect transfer? Turns out yes.

Neural network visualization

Why?

- “Interpretability”:
    - Want to know what the neural network is basing its decision on.
- In general, any way of visualization approximate

Receptive field



(2x-3,2y-3) …
(2x+3, 2y+3)

(2x-2,2y-2) …
(2x+2, 2y+2)

(x-1,y-1) …
(x+1, y+1)

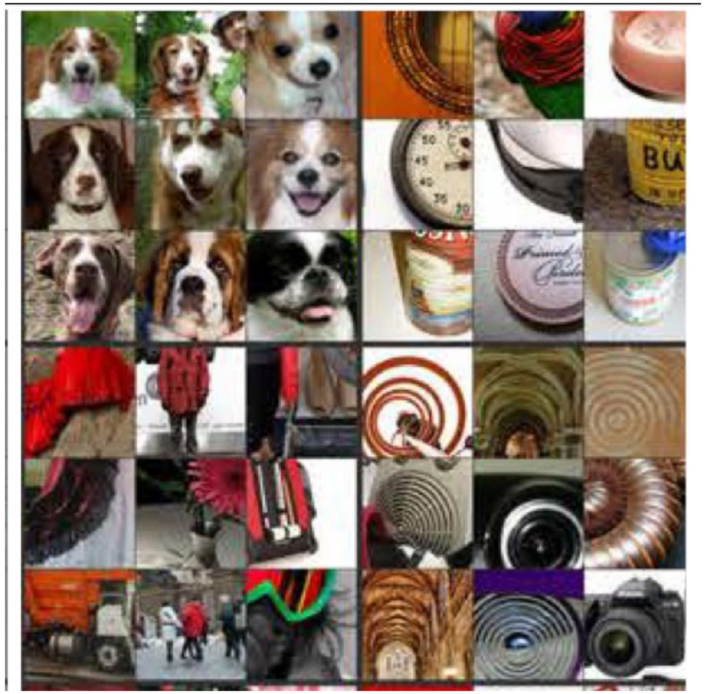
Visualizing the receptive field

- Pick a layer
- Pick a channel
- Pick a particular location in feature map
- Draw out the corresponding receptive field in the image

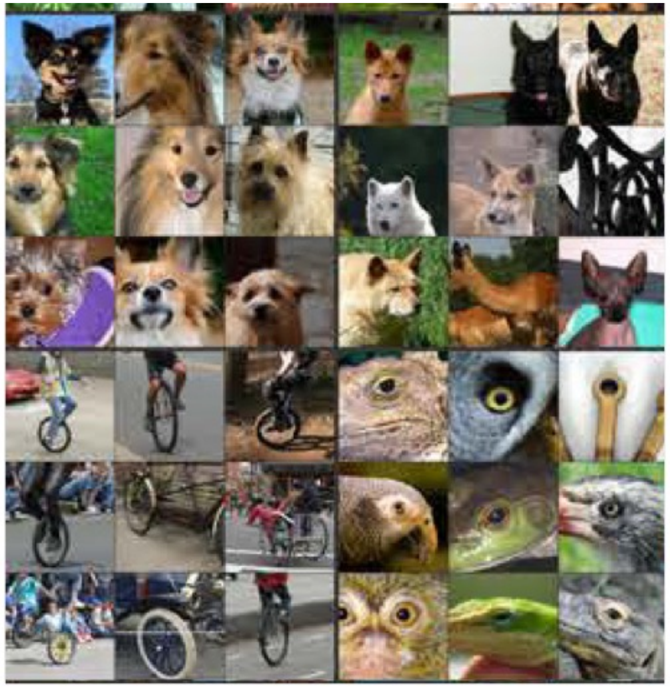
Visualizing what activates channels

- Pick a layer
- Pick a channel
- Identify images and locations that give the highest value

Layer 1



Layer 4

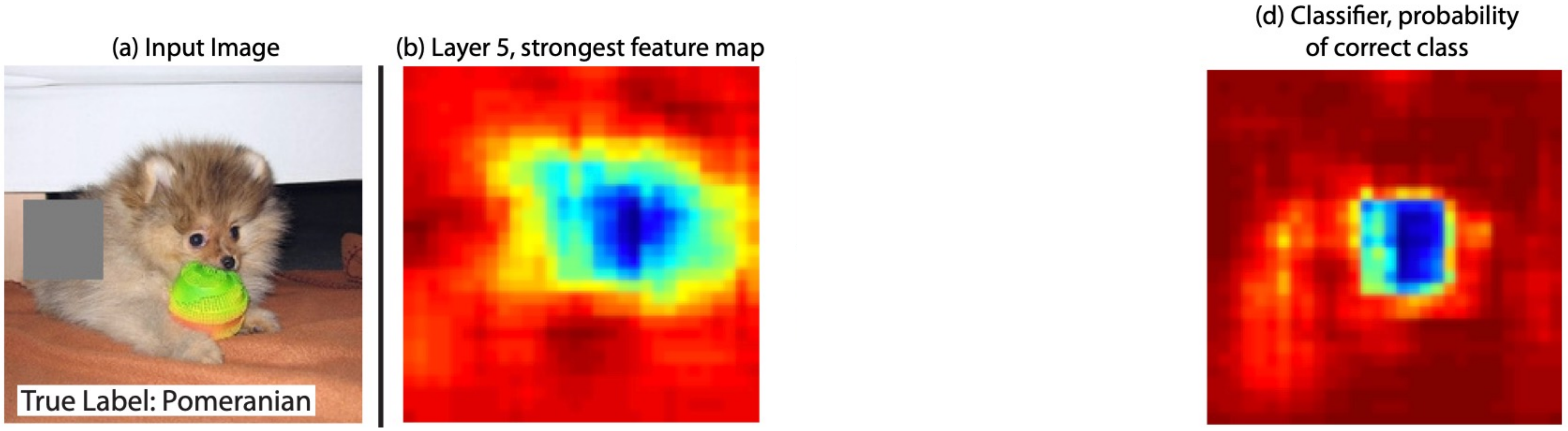


Layer 5

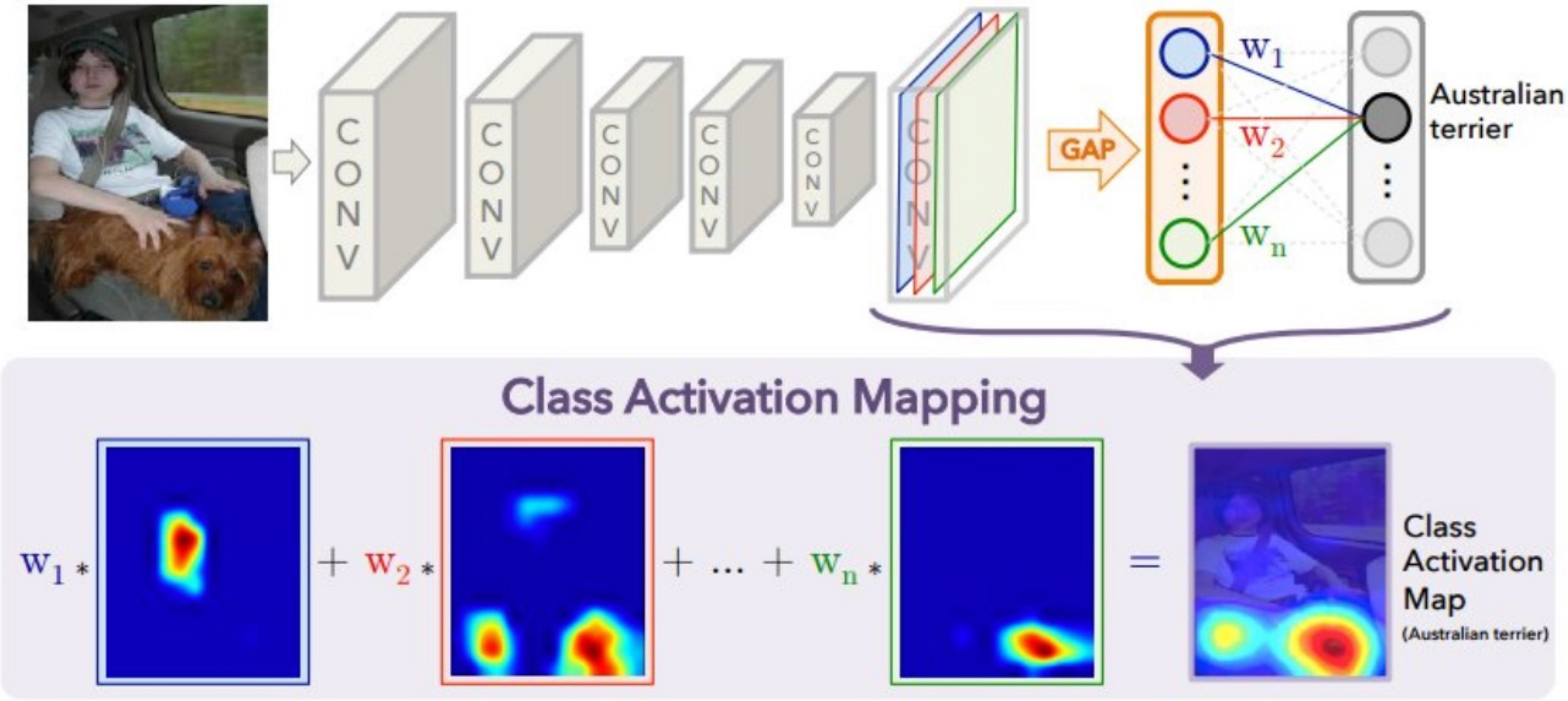
Matthew Zeiler, D., and Fergus Rob. "Visualizing and understanding convolutional neural networks." ECCV, 2014.

Visualizing what patches are important

- Block part of image with a grey square and record class score
- Move gray square over image to get 2D array of scores
- Result is heatmap with low score for important patches



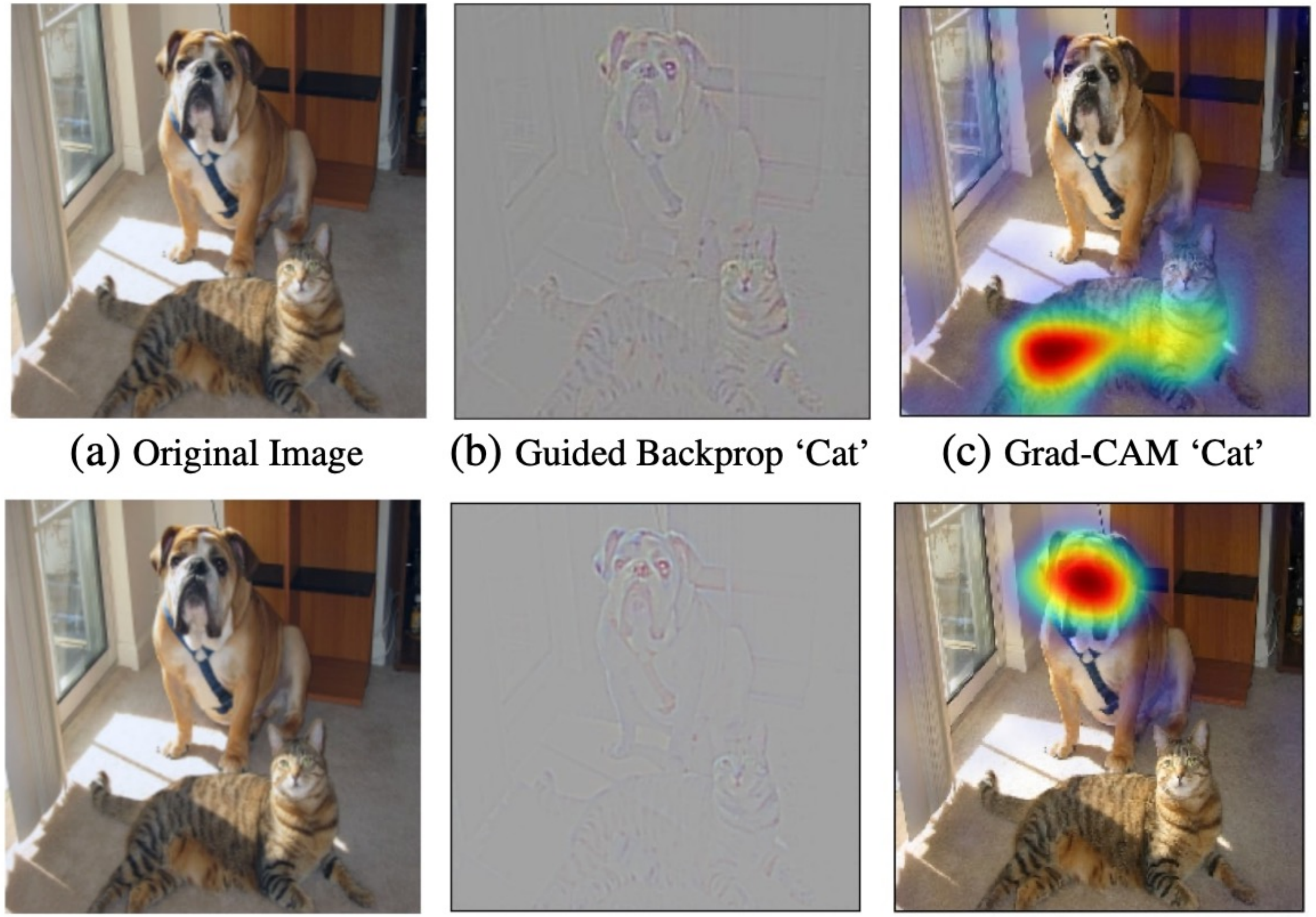
Class activation maps



Zhou, Bolei, et al. "Learning deep features for discriminative localization." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Grad-CAM

- Can also look at gradient of class scores w.r.t. image pixels
- Indicates which pixels will cause highest change in scores
- By itself not useful because it tends to highlight edges or corners
- But can be combined with CAM



(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat'

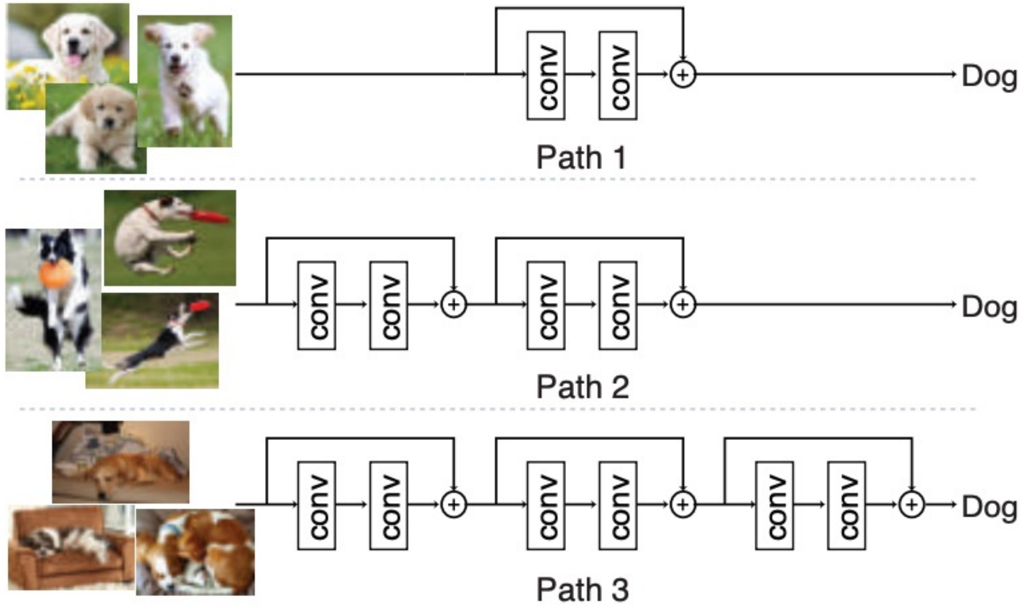
(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog'

Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." *Proceedings of the IEEE international conference on computer vision*. 2017.

Other ways of reducing computation

Adaptive inference

- Some examples are harder than others
- Should be able to use different amounts of computation for different examples
- Version 1: skip some residuals

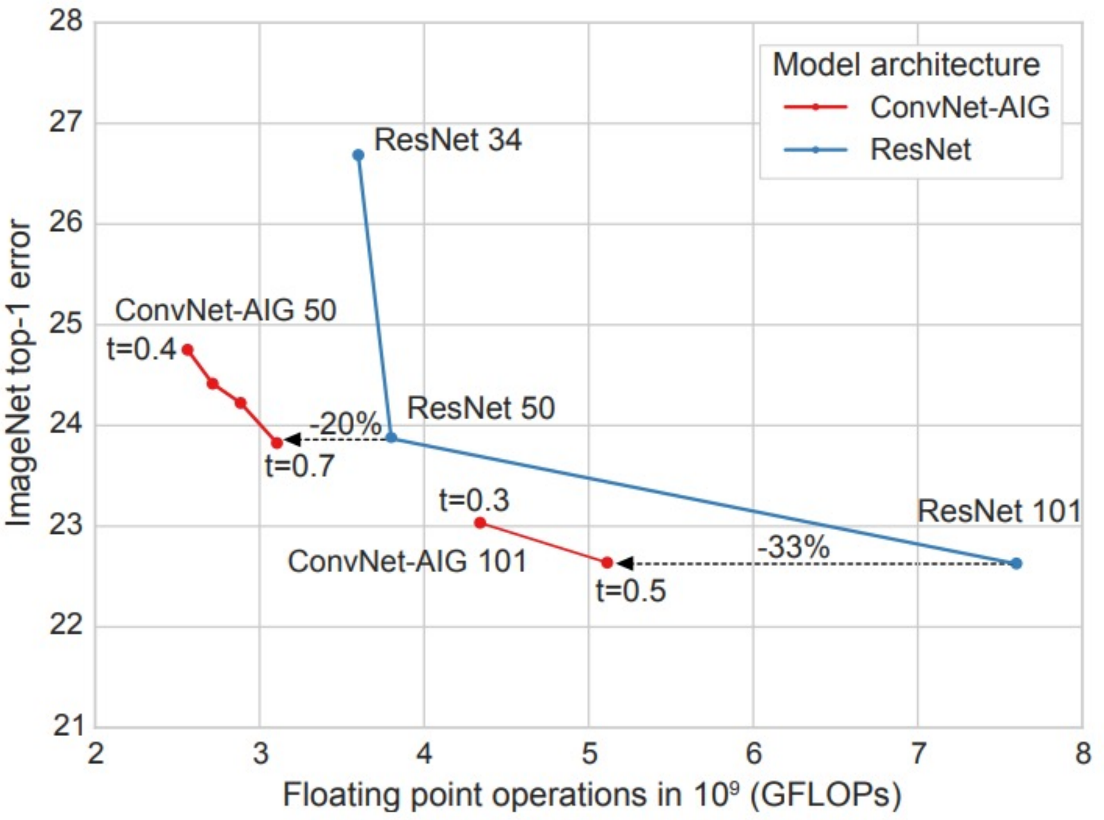


Veit, Andreas, and Serge Belongie. "Convolutional networks with adaptive inference graphs." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Wu, Zuxuan, et al. "Blockdrop: Dynamic inference paths in residual networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

Adaptive inference

- Some examples are harder than others
- Should be able to use different amounts of computation for different examples
- Version 1: skip some residuals

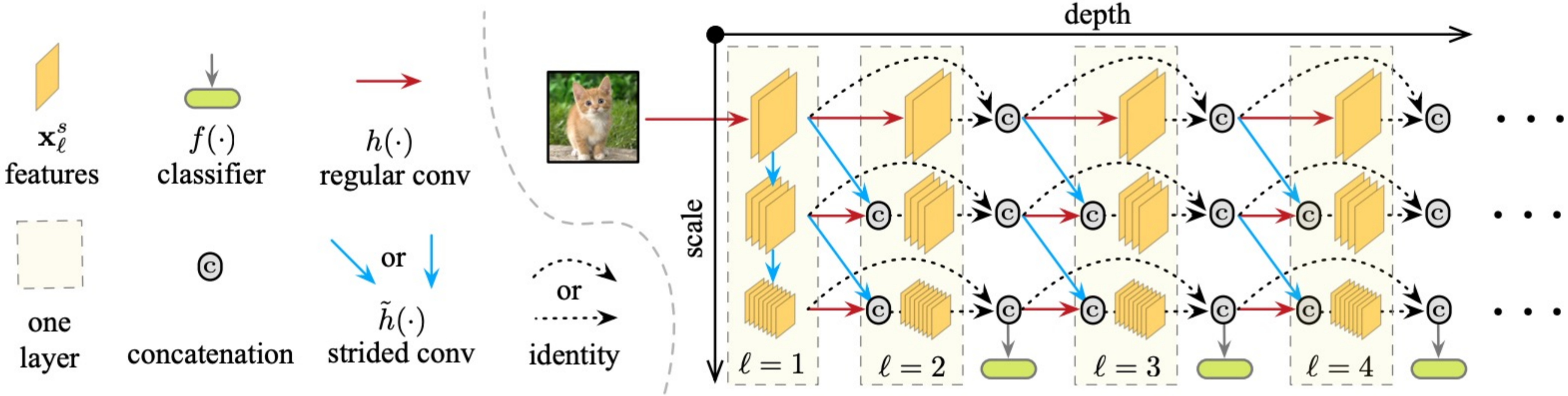


Veit, Andreas, and Serge Belongie. "Convolutional networks with adaptive inference graphs." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

Wu, Zuxuan, et al. "Blockdrop: Dynamic inference paths in residual networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

Adaptive inference

- Some examples are harder than others
- Should be able to use different amounts of computation for different examples
- Version 2: reduce resolution at different rates



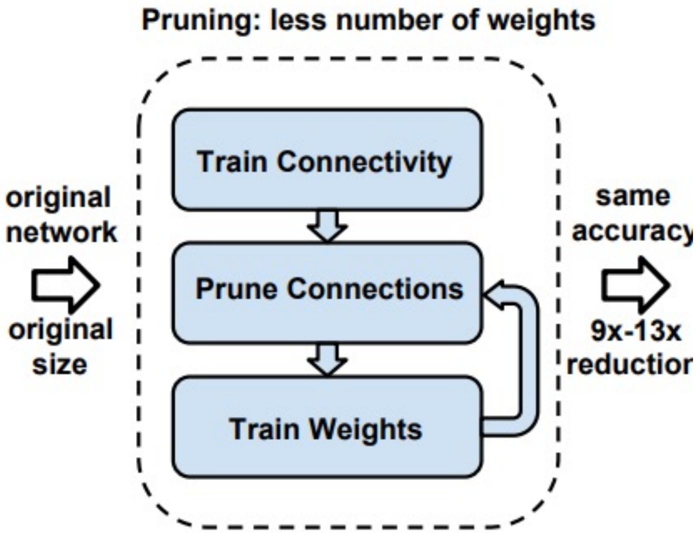
Huang, Gao, et al. "Multi-scale dense networks for resource efficient image classification." *ICLR 2018*

Compressing model weights

- All of model storage: filters
- Flops also scale with non-zero entries in filters (in principle)
- Compress filters
  - Sparsify them
  - Represent them with fewer bits

Pruning network connections

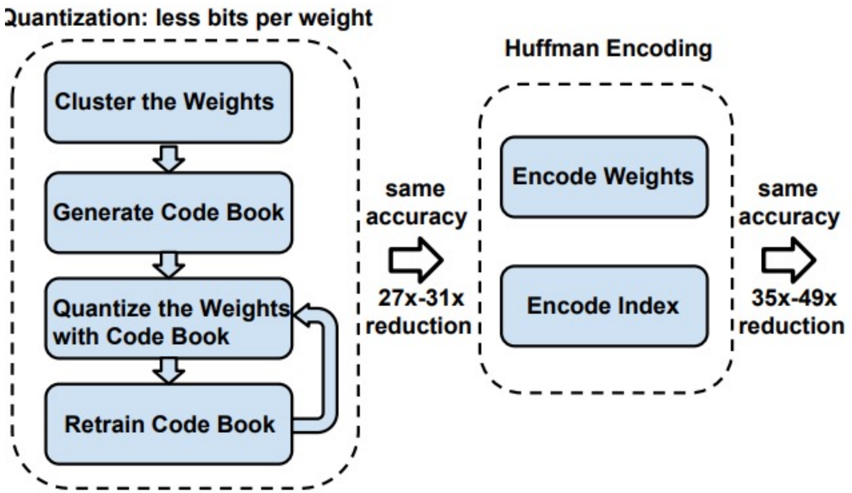
- Simple approach: prune weights that are below a threshold
- Retrain rest of the weights
- Repeat



Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *ICLR, 2016*

Filter quantization

- Two questions:
  - How do we quantize?
  - Quantization → discrete values. How do we optimize?
- Example 1: *cluster*
  - Weights → indices into dictionary
  - Update dictionary elements as parameters.



Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *ICLR, 2016*

Filter quantization

- Two questions:
  - How do we quantize?
  - Quantization → discrete values. How do we optimize?
- Example 2: *binarize/ternarize*
  - Weights → binary/ternary, + real-valued scale
  - Parameter updates happen in real space

Zhu, Chenzhuo, et al. "Trained ternary quantization." *ICLR, 2017*.

Rastegari, Mohammad, et al. "Xnor-net: Imagenet classification using binary convolutional neural networks." *European Conference on Computer Vision*. Springer, Cham, 2016.