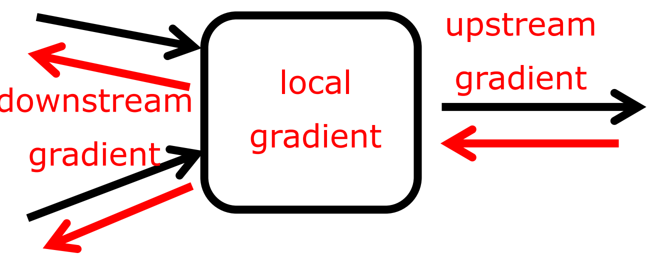
Photogrammetry & Robotics Lab

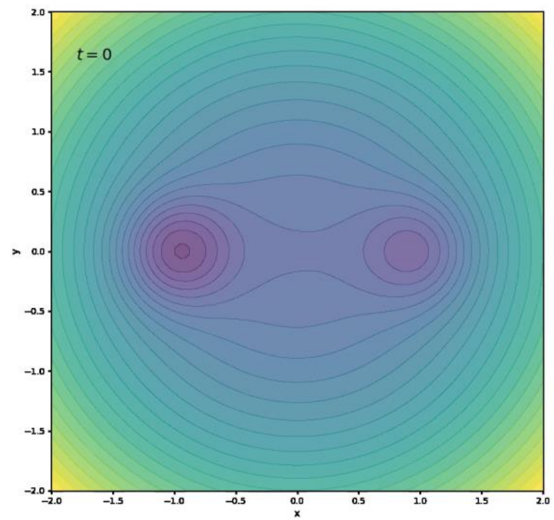
Machine Learning for Robotics and Computer Vision

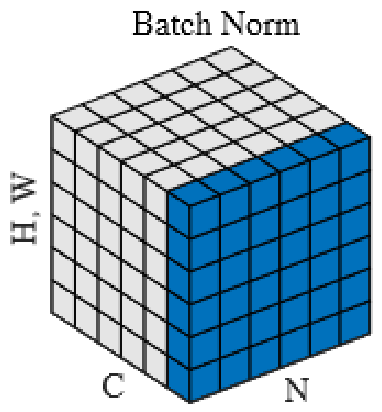
Building CNNs

Jens Behley

Last Lecture







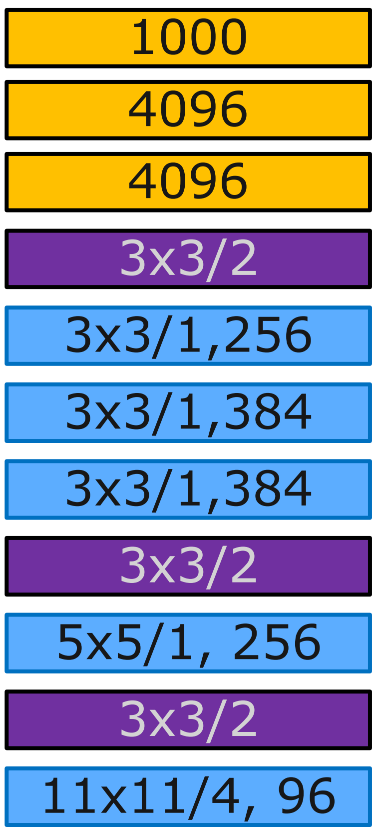
- Backpropagation for gradient computation
- Beyond gradient descent
- Good start for learning: Initialization
- Keep learning: Layer normalization

Designing CNNs

- Learning is one part, but more important how do we spend the **parameter budget**?
- How to **combine building blocks** to get a good classifier?

- We will look at **successful approaches** of the ImageNet competition
- Architectures that perform well on ImageNet usually also perform well in other tasks

Recap: AlexNet (2012)

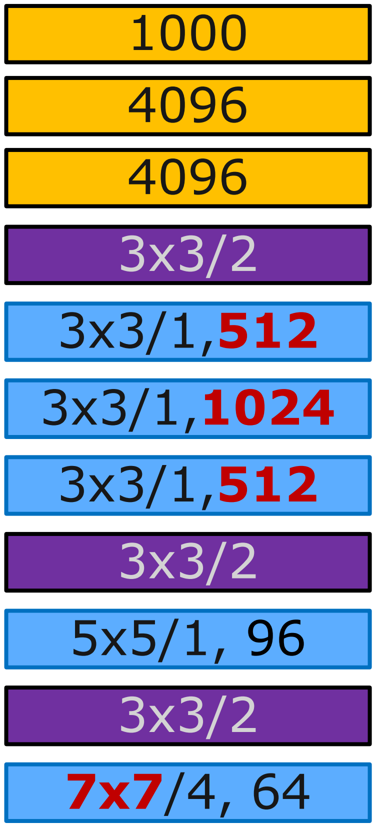


- 8 learnable layers
- ReLU after each convolutional layer
- 60M parameters, mostly in fully-connected layers

- Ensemble of 5 CNNs reaches 16.4% Top-5 error on test

[Krizhevsky, 2012]

ZFNet (2013)



- 8 learnable layers and some refinement on channels
- Ensemble of 6 CNNs reaches 11.7% Top-5 error on test set

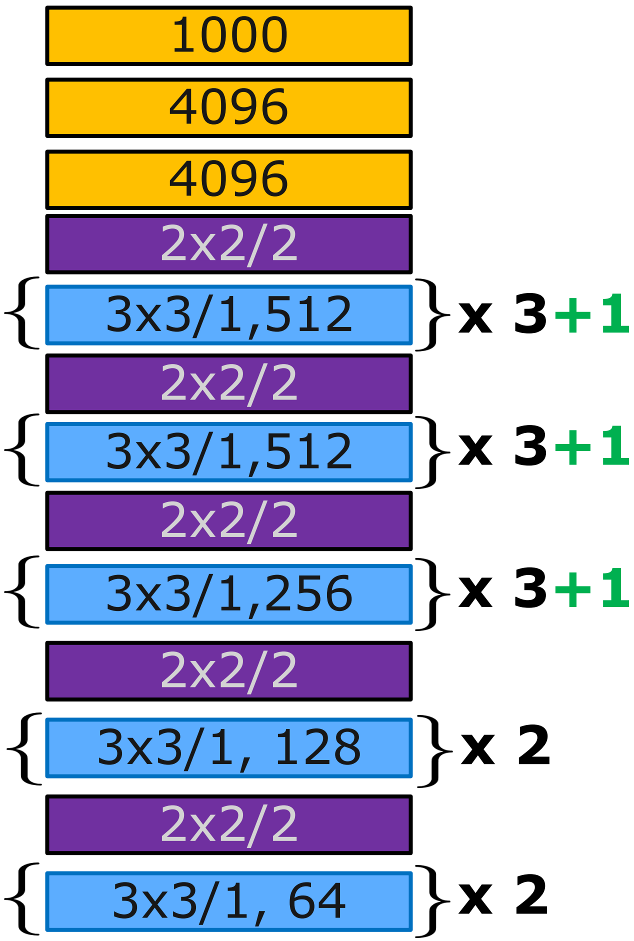
[Zeiler, 2013]

VGG Net: Design Principles

- Exploring deep networks with 16/19 learnable layers

- **Design Principles**
  - Only **3x3 convolutions**, **2x2/2 max pooling**
  - Same output feature map size → layers have same number of filters
  - If feature maps are halved → the number of filters is doubled

VGG-16/VGG-19 (2014)



VGG-16/19

- 16/19 learnable layers
- 138M/144M parameters again most in the FC layers
- **Pre-training** needed to make learning work: Smaller version are learned and weights copied → better initialization
- Reaches 7.3% top 5-error

[Simonyan, 2015]

Convolution + ReLU   Max Pooling   FC layer

GoogLeNet (2014)

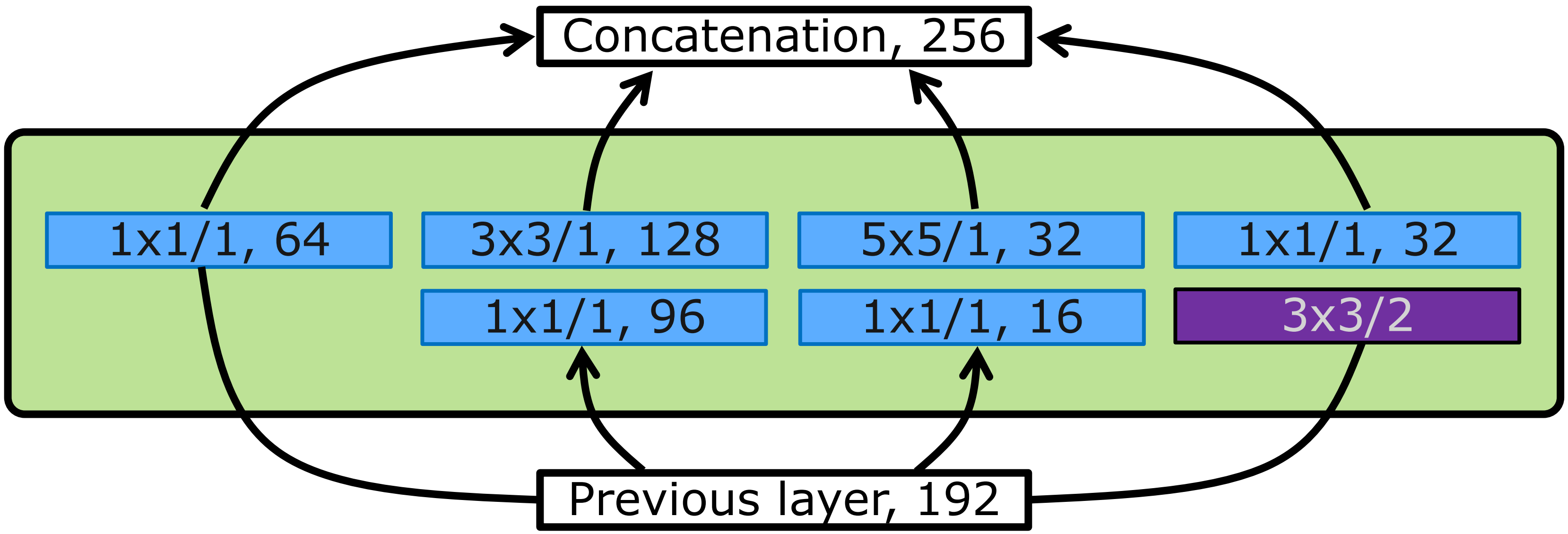
- Also uses much deeper networks and specific architecture with multiple pathways

- Main innovations:
  1. **Inception Layer** with multiple pathways
  2. **Global average pooling** to replace FC layers
  3. 1x1 convolutions as bottleneck (reducing channels)
  4. Intermediate auxiliary losses to enable training

- Note: With batch norm pre-training and auxiliary losses are not needed anymore!

[Szegedy, 2015]

Inception Layer

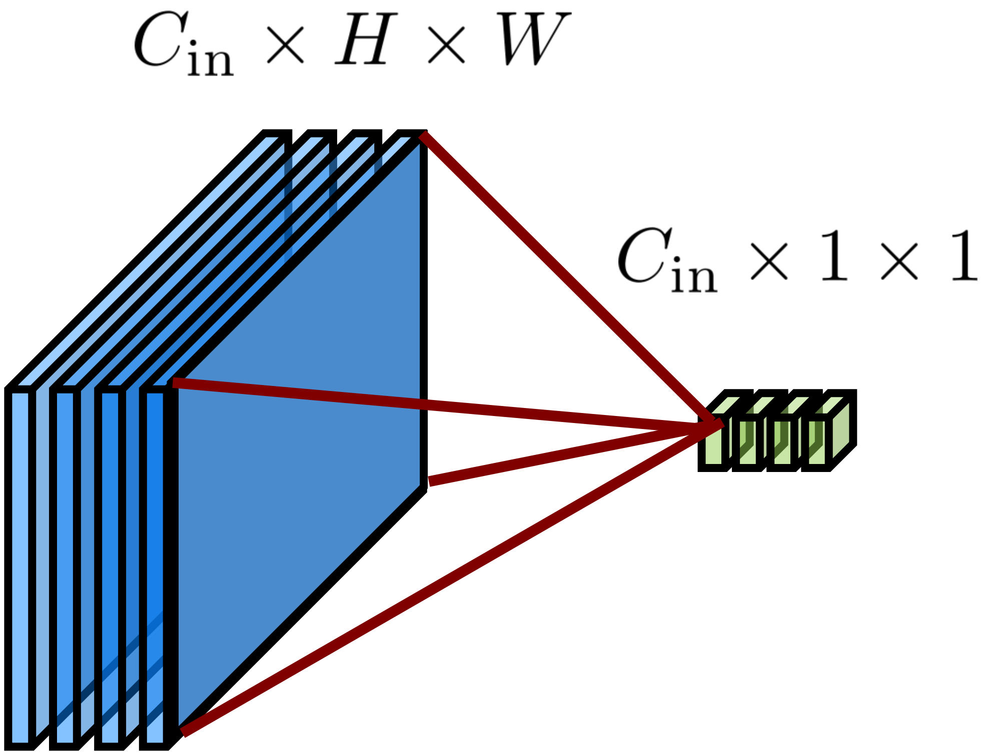


Inception (3a)

- Multiple pathways with different kernel sizes
- Concatenation of convolutional layers
- 1 x 1 convolutions reduce channels and makes following convolutions more efficient

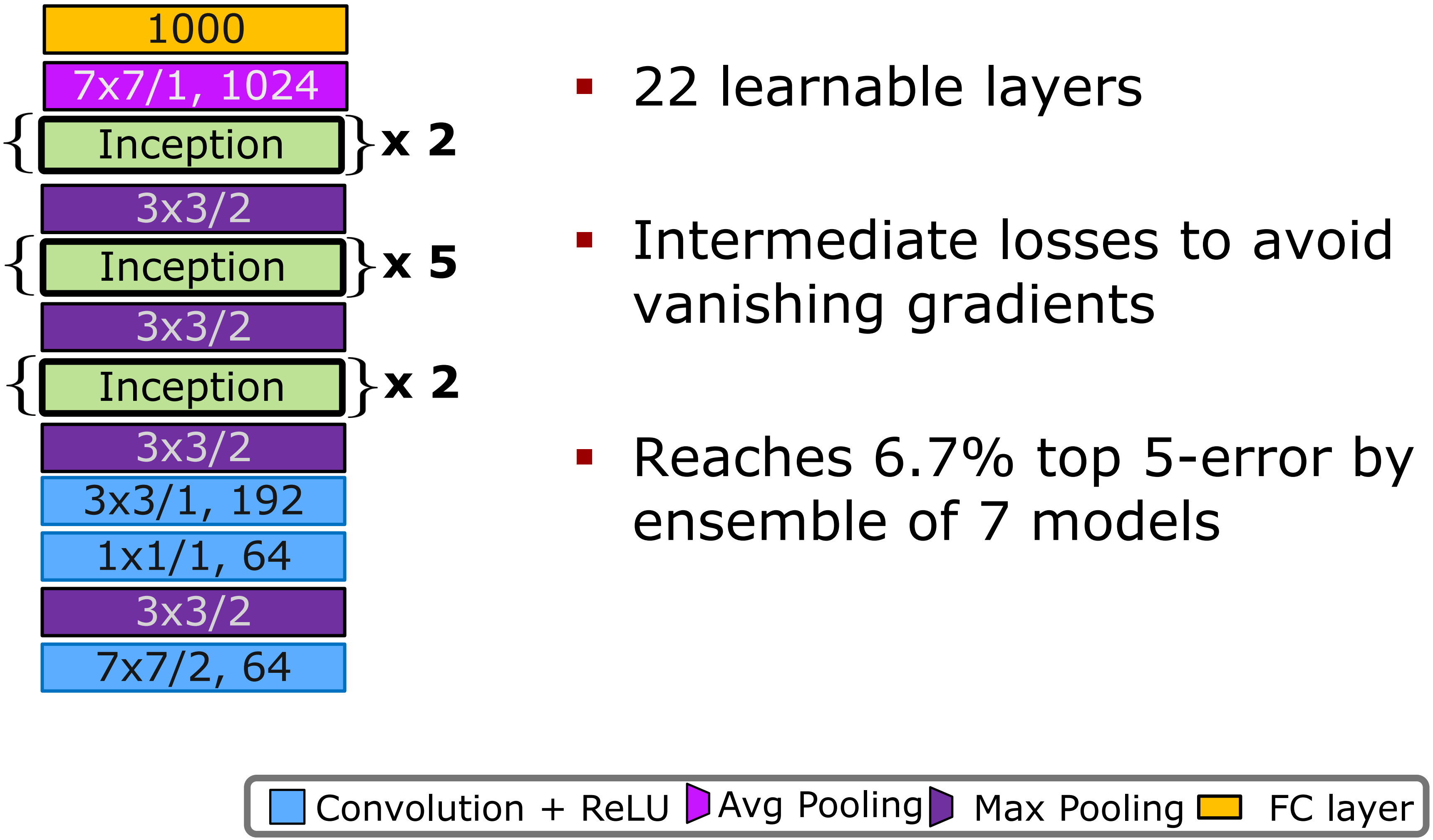
[Szegedy, 2015]

Global Average Pooling



- Average pooling with kernel HxW of feature map
  → average over each feature map
- Parameter-free reduction instead of FC layers

GoogLeNet (2014)

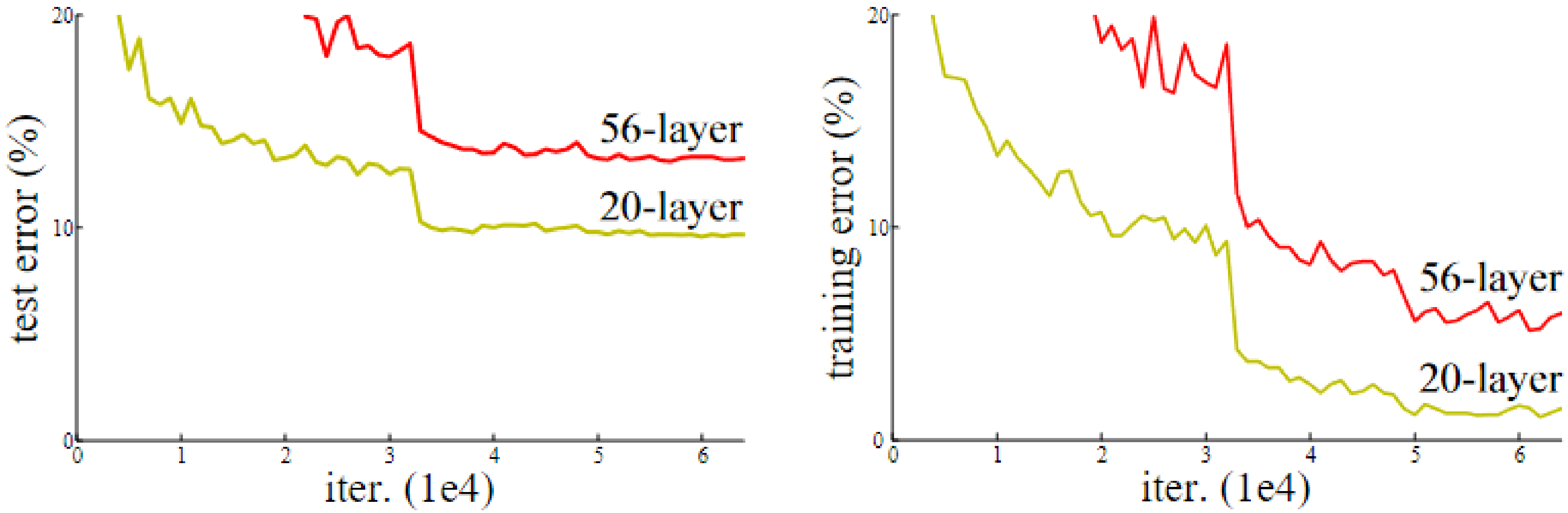


- 22 learnable layers
- Intermediate losses to avoid vanishing gradients
- Reaches 6.7% top 5-error by ensemble of 7 models

[Simonyan, 2015]

Deeper = Better Networks?

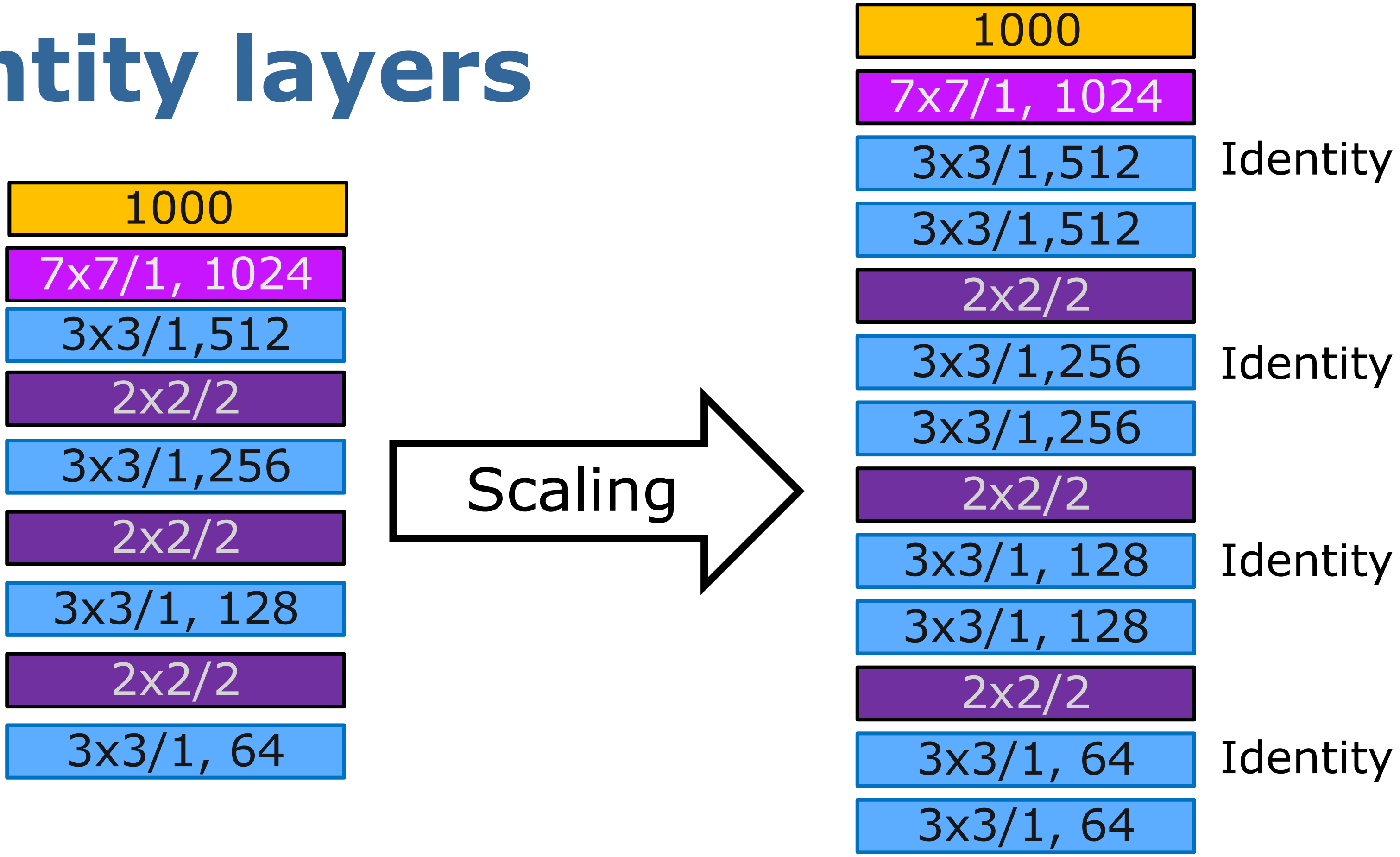
Performance on CIFAR-10



- Training deeper networks with proper initialization and batch norm (easily) possible
- **But:** With larger depth performance degrades
- Overfitting seems not to be the reason as training error also increases!

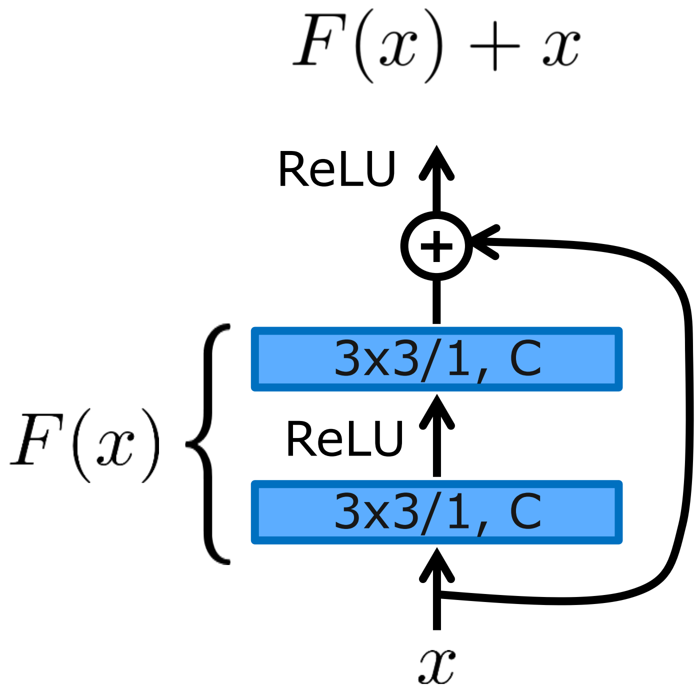
Figure from [He, 2015]

Identity layers



- Deeper networks should be able to learn weights of smaller networks by setting additional layers to identity
- How do we achieve this?

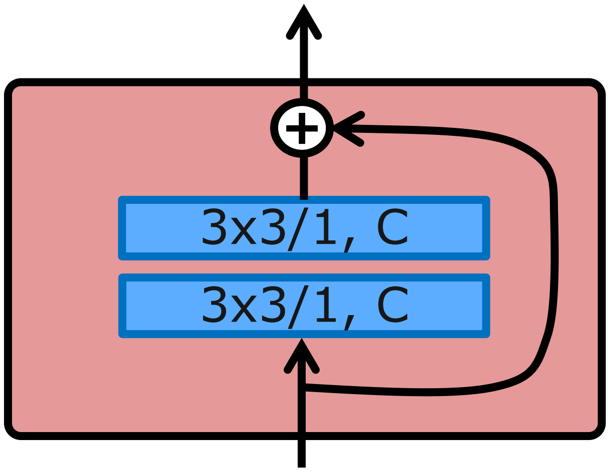
Skip Connections



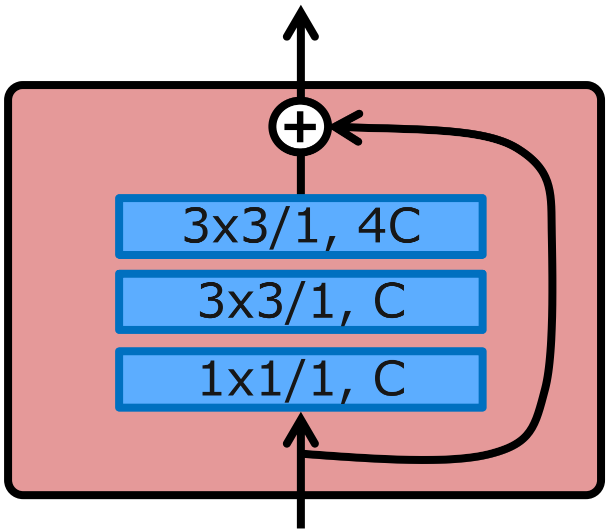
- Add skip connections that directly copy outputs from earlier layers

[He, 2015]

ResNet Blocks



Basic Block
(ResNet-18/34)

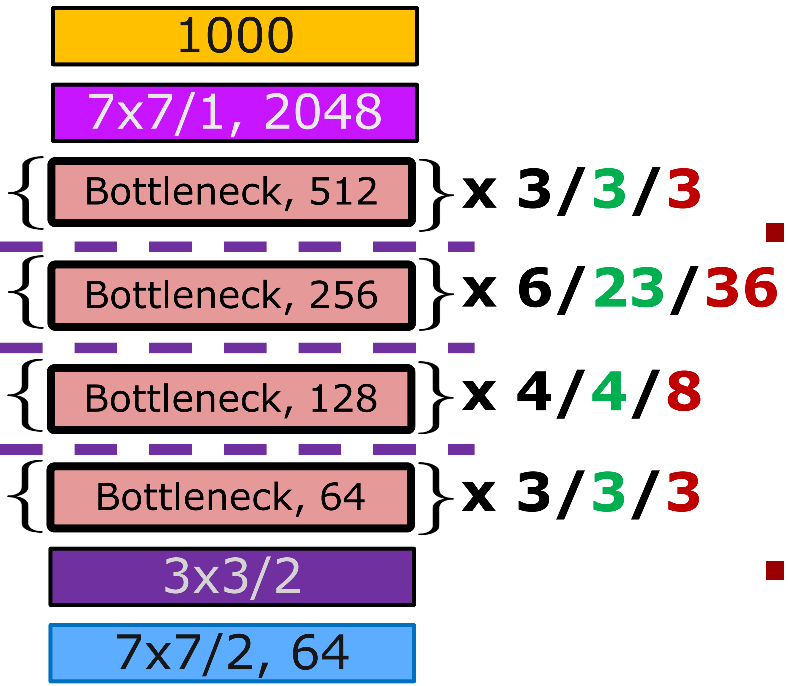


Bottleneck Block
(ResNet-50/101/152)

- Bottleneck block reduces channels for efficiency
- Batch Normalization after each convolution layer

[He, 2015]

ResNet (2015)



ResNet-50/101/152

- 18/34/50/101/152 learnable layers
- Downsampling (– –) via strided (s=2) convolution in first convolution of 2nd, 3rd, 4th stage
- ResNet-152 reaches 3.6% Top 5-error by ensemble of 6 models

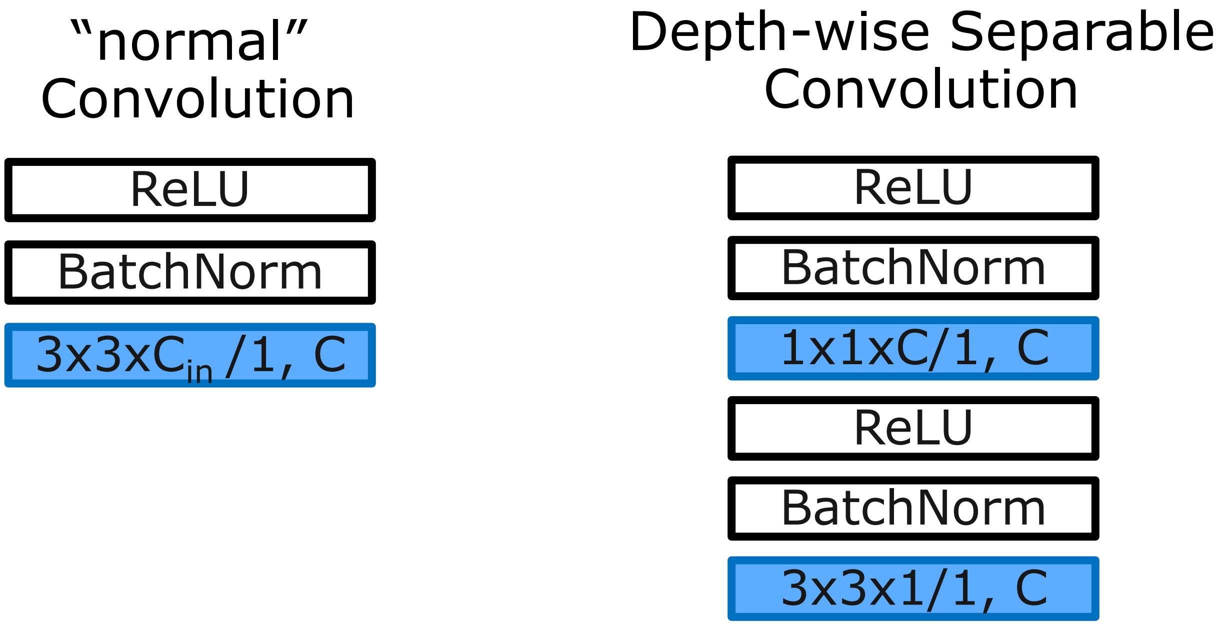
[He, 2015]

Progress on ImageNet

Architecture	Layers	Top-5 Error
▪ AlexNet (2012)	8	16.4%
▪ ZFNet (2013)	8	11.7%
▪ VGG-19 (2014)	19	7.3%
▪ GoogLeNet (2014)	22	6.7%
▪ ResNet-152 (2015)	152	3.6%
▪ ResNeXt-152-SE (2017)	152	2.3%

- No ImageNet competition after 2017
- Still progress on ImageNet
  - Now: validation set errors ~ test set errors, minival from training data for hyper parameter search)
- Top-1 accuracy commonly reported

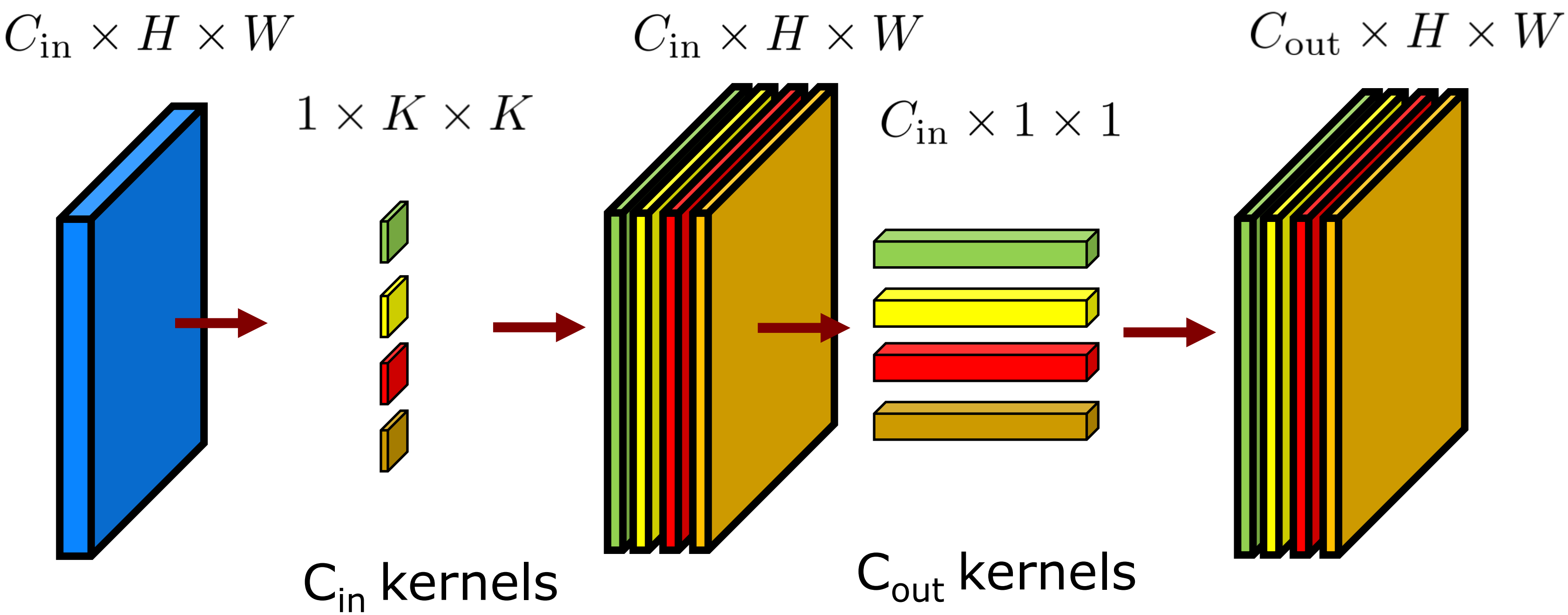
MobileNet (2017)



- On robots & mobile phones: efficiency needed in terms of time & memory
- Depth-wise separable 3x3 convolutions reduce computation by factor 8-9
- Small loss in accuracy compared to "normal" conv

[Howard, 2017]

Depth-wise Separable Convolution



- Separate channel-wise application and "mixing" of channels

Comparison

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 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

- Small drop in accuracy by using depth-wise separable convolution
- Comparable performance to “earlier” ImageNet models

[Howard, 2017]

Tables from [Howard et al., 2017]

Scaling MobileNet

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

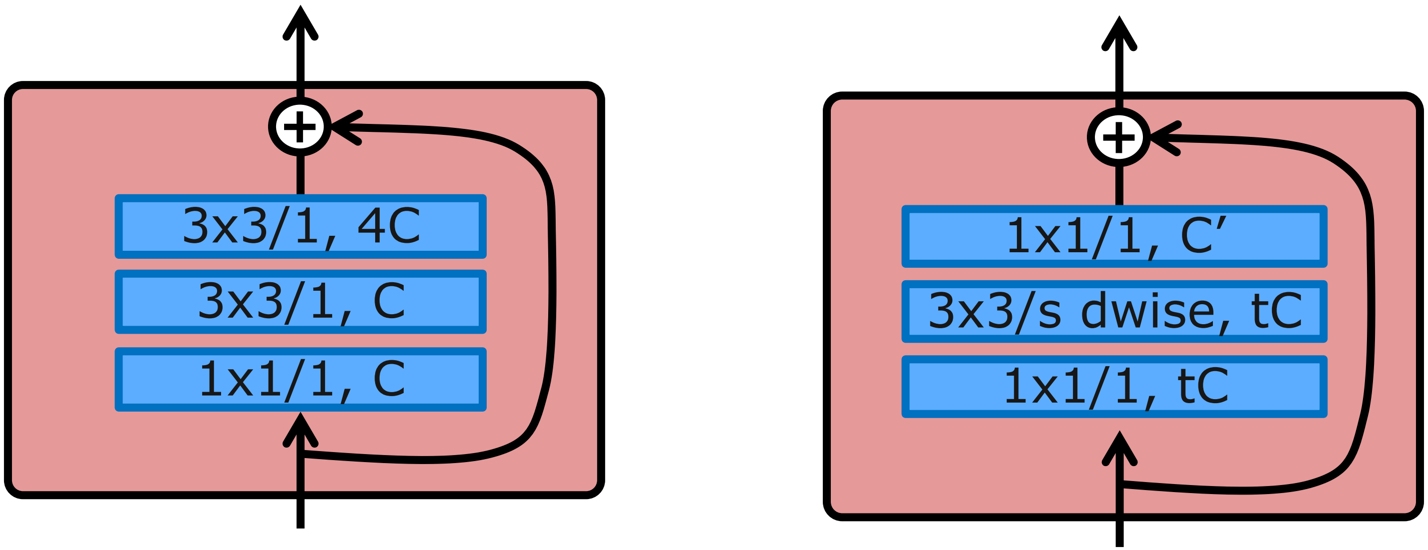
Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

- Width & resolution scaling enables to further reduce the model size
- Performance degrades gracefully with less parameters (width) and smaller images

[Howard, 2017]

Tables from [Howard et al., 2017]

Inverted Bottleneck



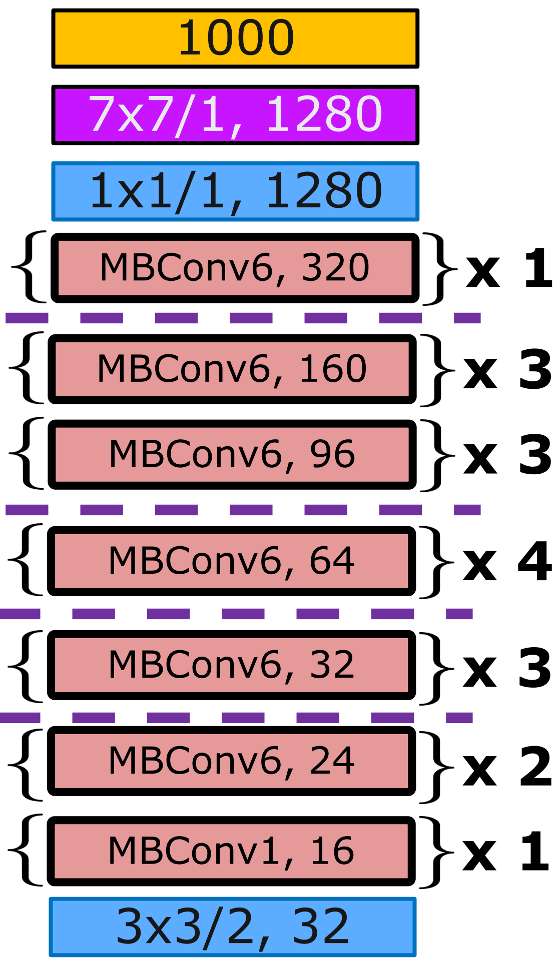
Bottleneck Block (larger ResNets)

Inverted Bottleneck MBConv

- ResNet Bottleneck Block: reduces number of channels
- Inverted Bottleneck for more memory efficiency

[Sandler, 2018]

MobileNetV2 (2018)



MobileNetV2

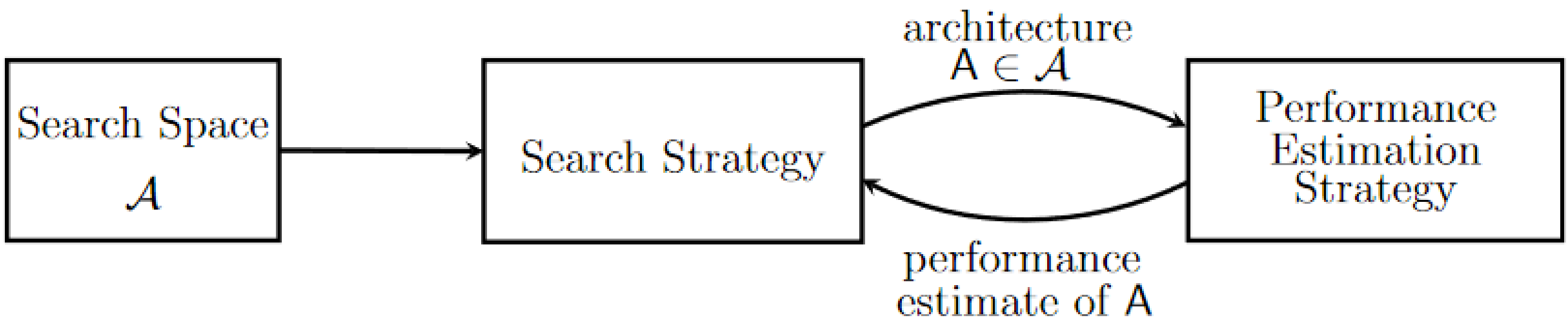
[Sandler, 2018]

- 20 learnable layers
- All bottleneck layers are 3x3 depth-wise separable convolutions
- MBConv1 is t=1 expansion factor
  MBConv6 is t=6 expansion factor

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Table from [Sandler et al., 2018]

Neural Architecture Search (NAS)

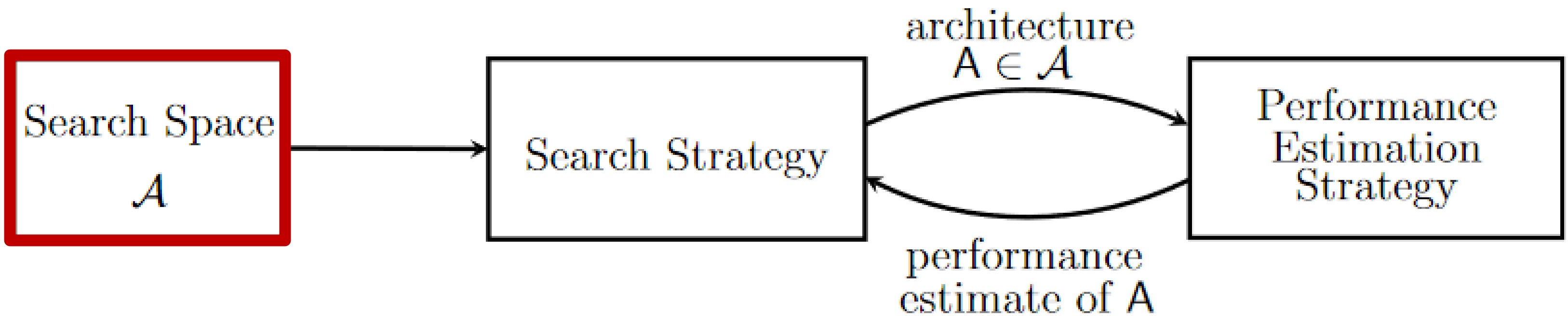


- Instead of handcrafted design: Automatically search good architectures → NAS

[Elsken, 2019]

Figure from [Elsken et al., 2019]

Neural Architecture Search (NAS)

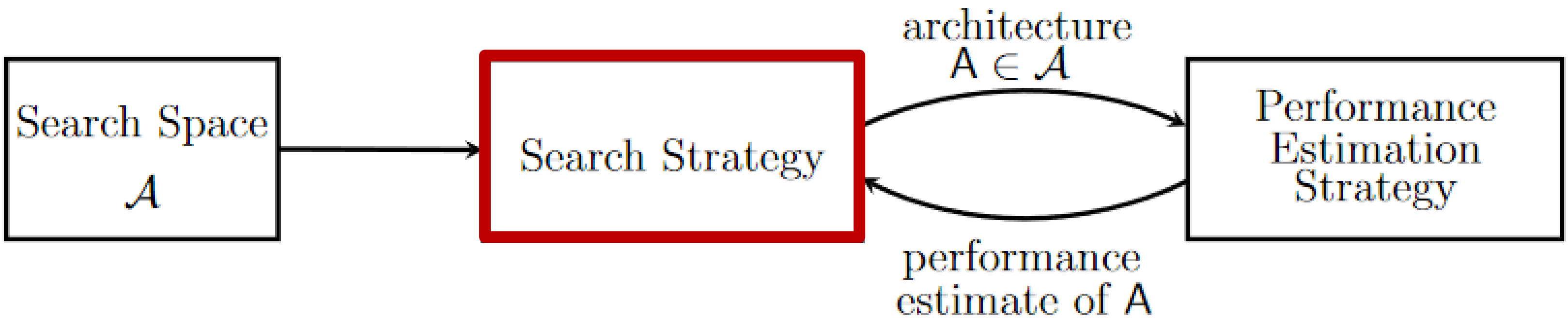


- **Search Space**: Structure and building blocks
  - Structure: Linear vs. parallel pathways
  - Operations: Convolutions, Depth-separable Convolutions, Activations, …
  - Blocks: ResNet Blocks, Inception, …

[Elsken et al., 2019]

Figure from [Elsken et al., 2019]

Neural Architecture Search (NAS)

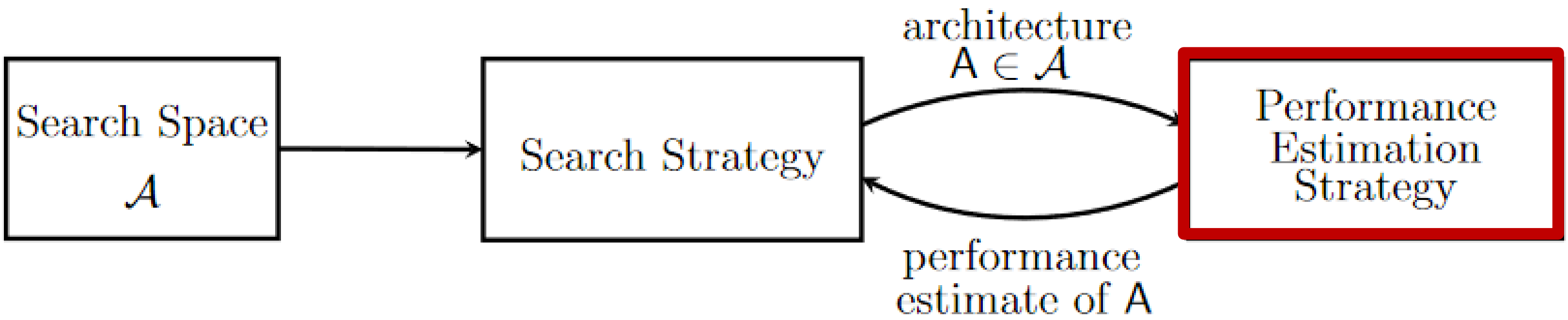


- **Search Strategy:** Selection of instances from Search Space
  - Reinforcement Learning: Select instances based on reward from performance
  - Genetic Algorithms: Mutate current strategy based on performance

[Elsken et al., 2019]

Figure from [Elsken et al., 2019]

Neural Architecture Search (NAS)

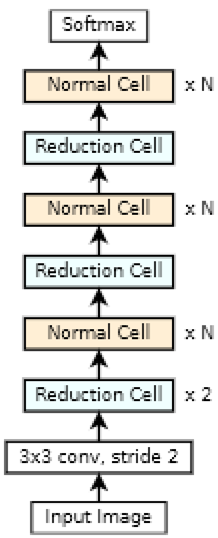


- **Performance Estimation:** How well does the selected architecture perform?
- Main objective: Get quick estimate of performance
  - Short training cycle with fewer epochs
  - Extrapolate performance from a few examples
  - Initialize weights from previous structure

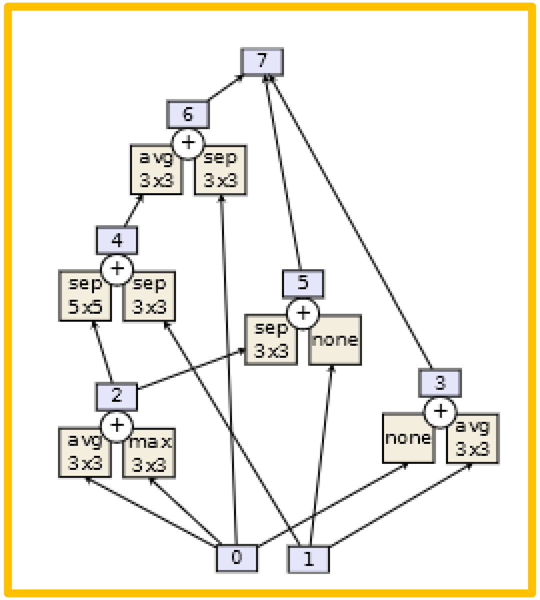
[Elsken et al., 2019]

Figure from [Elsken et al., 2019]

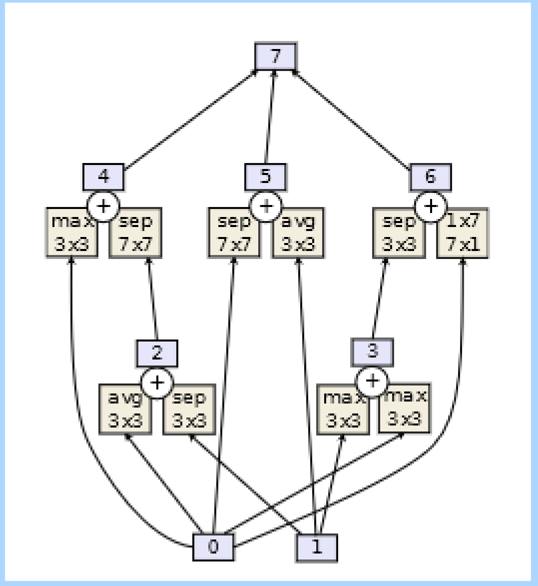
Neural Architecture Search (NAS)



AmoebaNet-A



Normal Cell



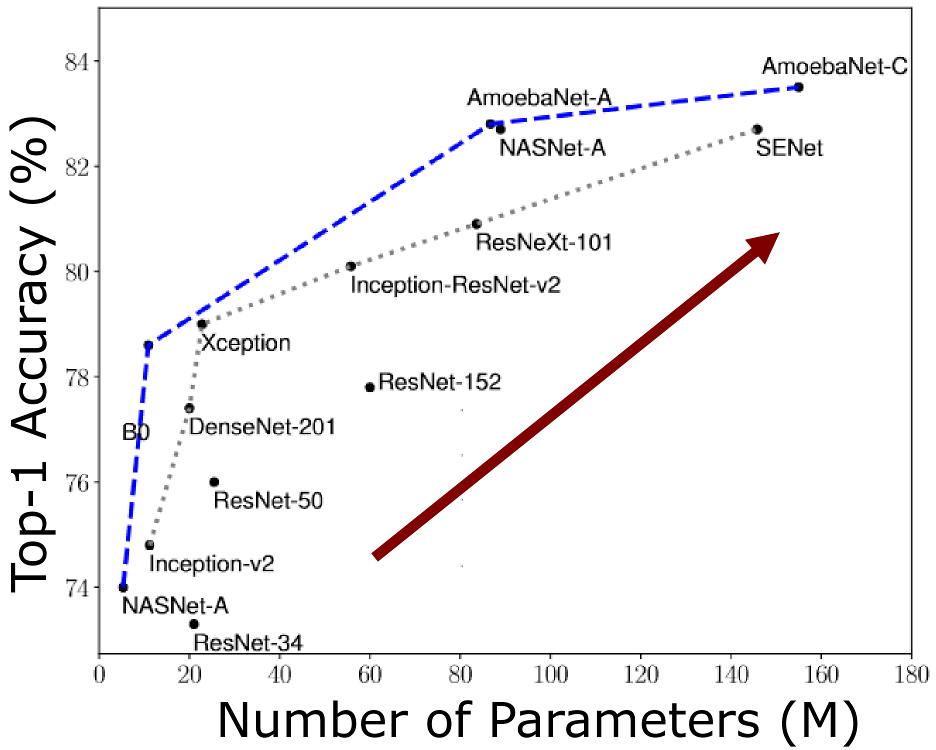
Reduction Cell

- Still computational expensive search
- Solutions sometimes not very intuitive
- Often used with objective (number of parameters, FLOPS, targeted architecture, e.g., mobile phones)

[Real et al., 2019]

Figure from [Real et al., 2019]

Parameter Trade-off

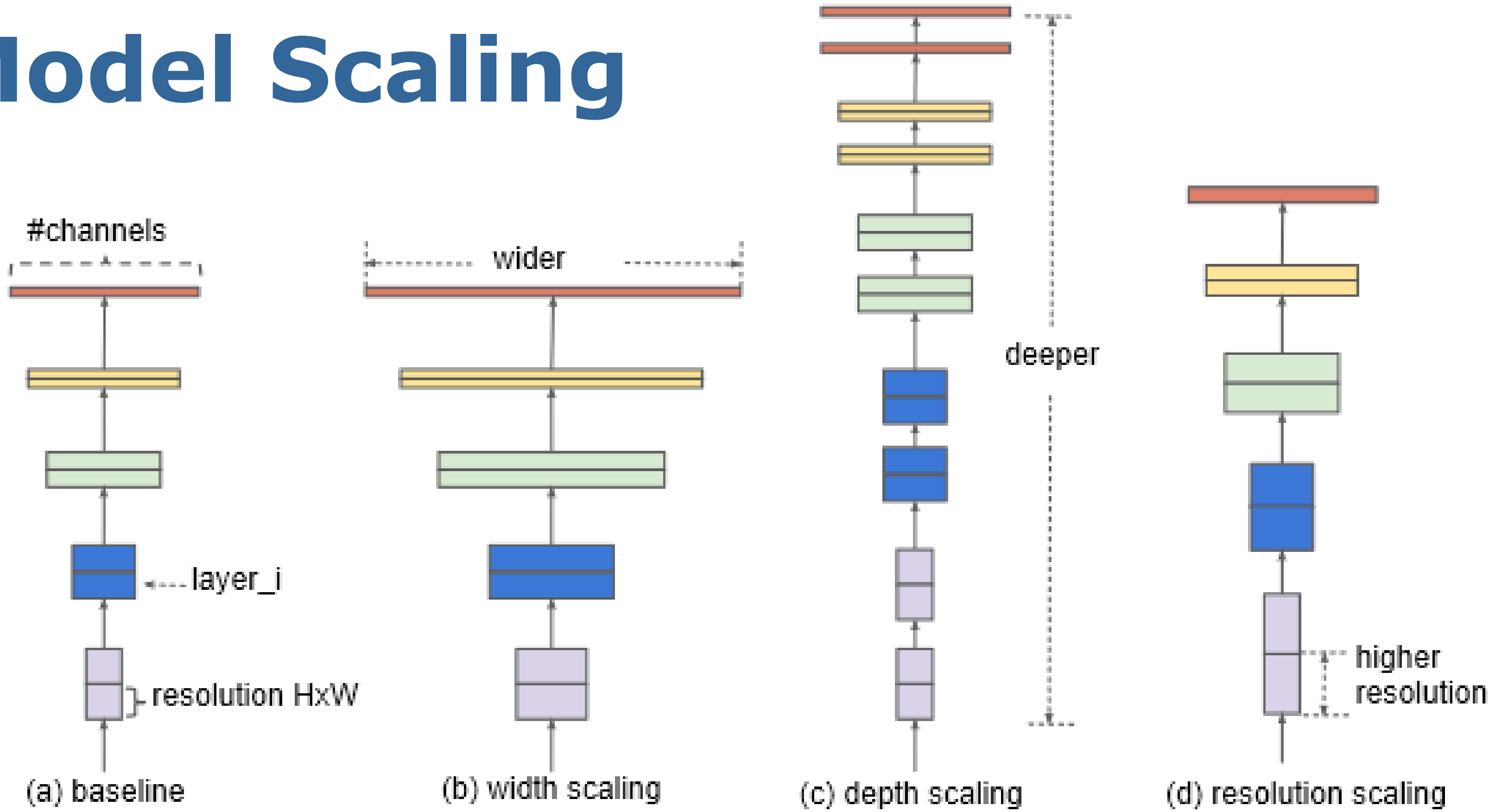


- Trend: more parameters → better performance?
- Can we get away with less parameters?
- How to efficiently scale a network?

[Tan et al., 2019]

Figure from [Tan et al., 2019]

Model Scaling

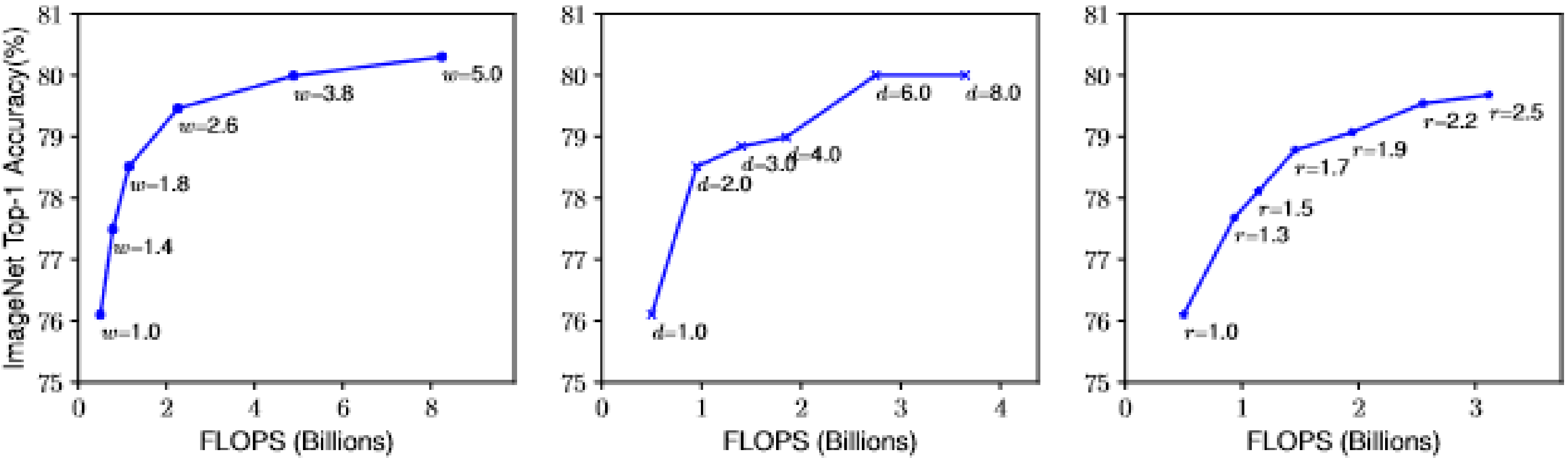


- Common scaling patterns (e.g., MobileNet)
  - (b) more channels
  - (c) more layers
  - (d) higher input resolution

[Tan et al., 2019]

Figure from [Tan et al., 2019]

Model Scaling



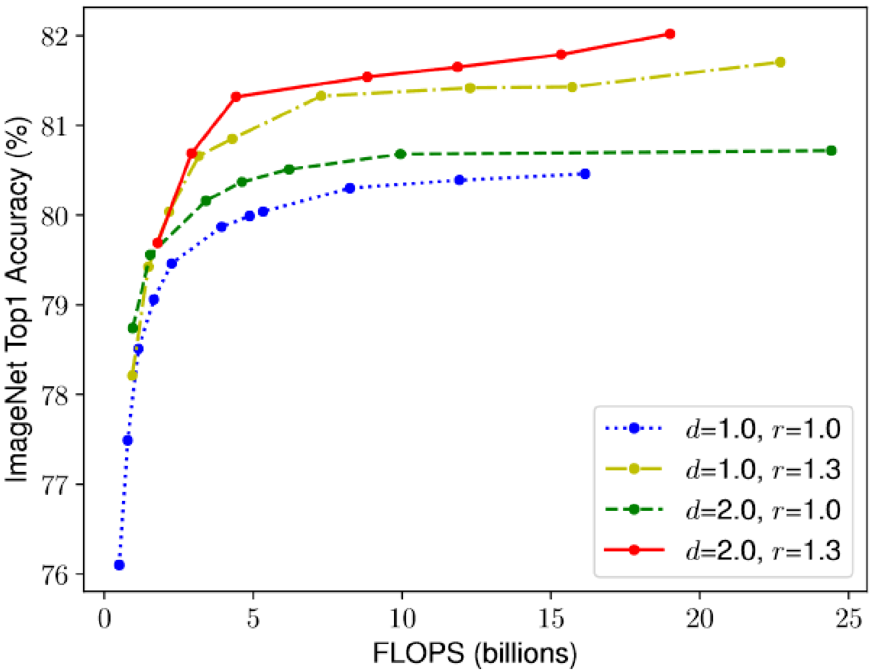
- Scaling individual dimensions reaches higher accuracies
- But: Diminishing returns with larger scaling

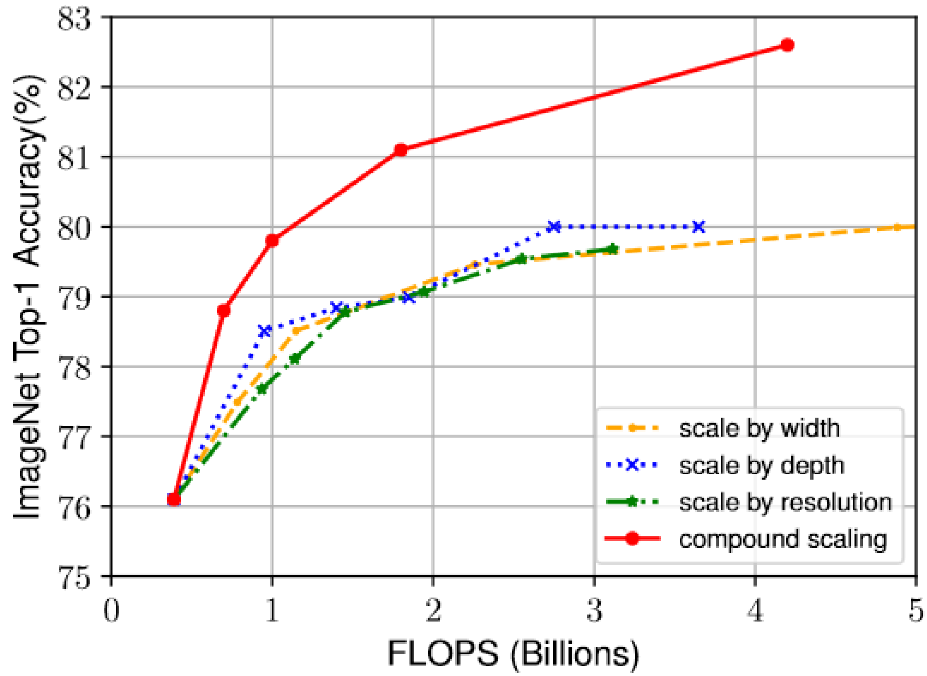
- Can we do better?

[Tan et al., 2019]

Figure from [Tan et al., 2019]

Compound Scaling



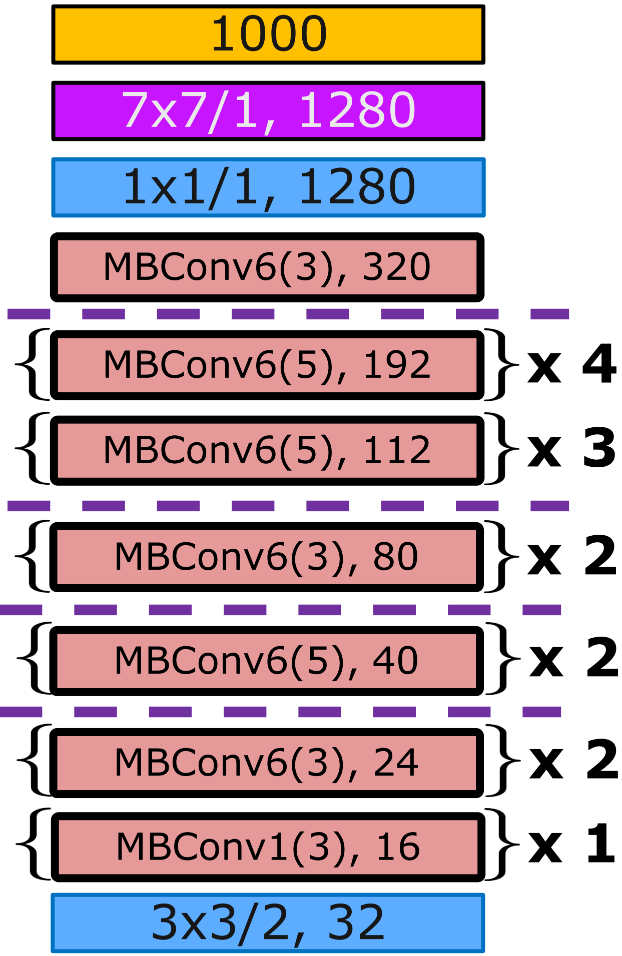


- Scaling multiple dimensions at the same time (**compound scaling**) reaches higher performance!

[Tan, 2019]

Figure from [Tan, 2019]

EfficientNet (2019)



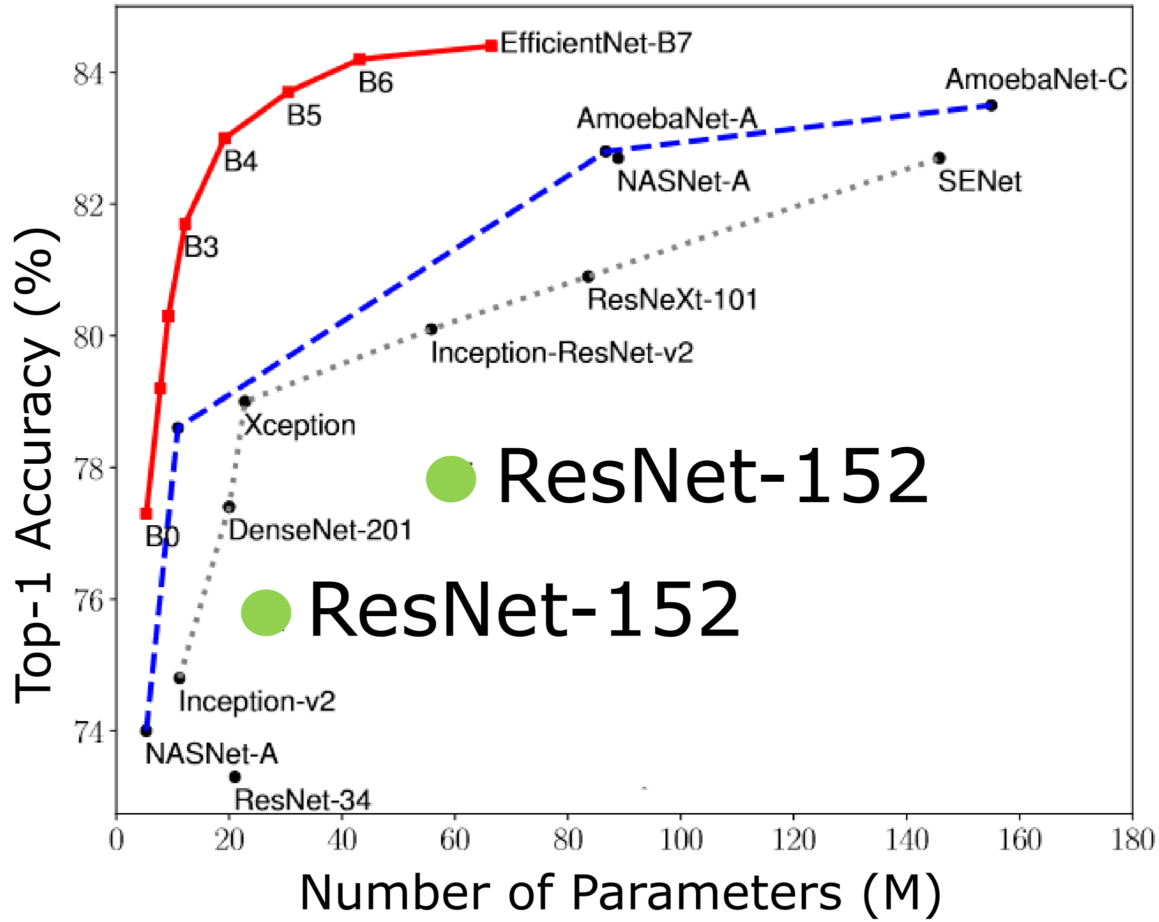
EfficientNet-B0

[Tan, 2019]

- NAS with objective of accuracy and computational efficiency (FLOPS)
- Structural very similar to MobileNetV2
  - Different number of channels
  - Convs with 3x3 and 5x5 kernels
- MBConv6(k) is inverted bottleneck with t=6 and k x k depth-wise separable convolutions

Model	Top-1	#Params
EfficientNet-B0	**77.1**	5.3M
ResNet-50	76.0	26 M
MobileNetV2	72.0	**3.4M**

Scaling EfficientNet



- Based on EfficientNet-B0, compound scaling (B1-B7) shows superior performance with smaller number of parameters

[Tan, 2019]

Figure from [Tan., 2019]

Current CNN Trends

- Normalization-free Networks [Brock,2021]
  - Replace normalization by weight adaptive gradient updates
  - Improved training efficiency, but very large models

- ResNet-RS [Bello, 2021]
  - Small changes to ResNet architecture
  - Improved training and scaling strategies
  - Faster to train then EfficientNet

- EfficientNetV2 [Tan, 2021]
  - Replace MBConv with FusedMBConvs in earlier layers
    → again normal 3 x 3 convolutions in inverted bottleneck
  - Improved training efficiency

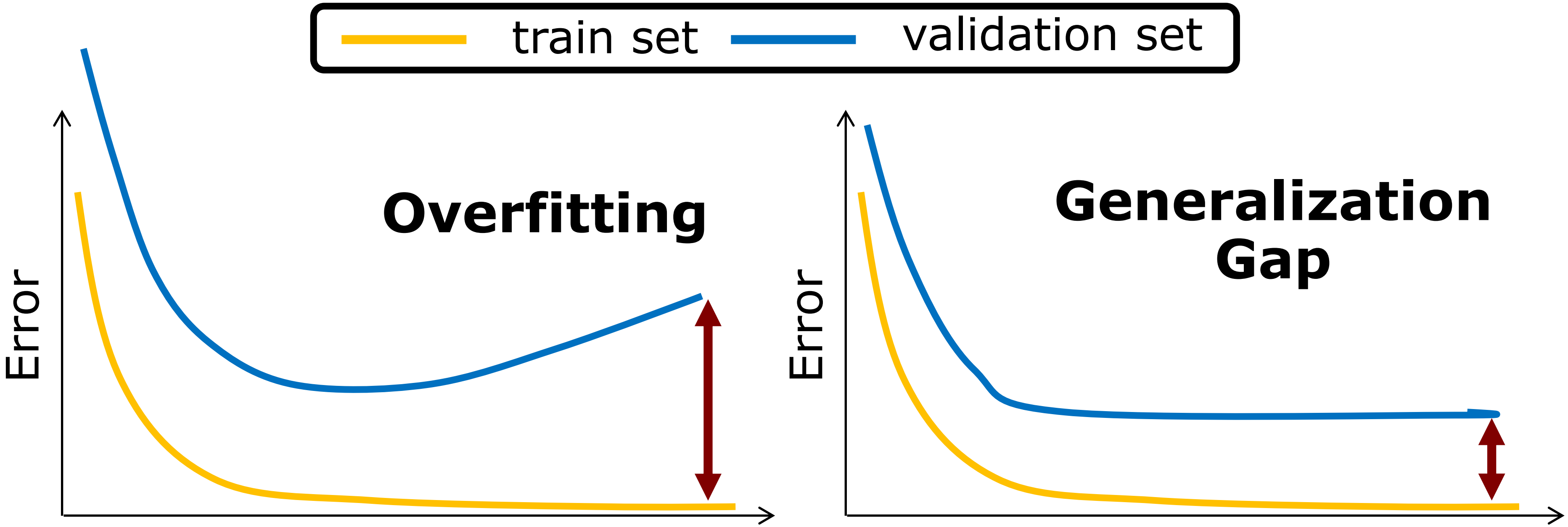
Which architecture to choose?

- **ResNet-50** is a good baseline model that is often used and shows good performance

- Nowadays, **EfficientNets** are efficient alternative and current research focuses on more on training efficiency

- Vibrant research field → novel insights
- Improved training strategies shows promising improvement even for “old” ResNets

Overfitting & Generalization



- With more parameters comes the risk of overfitting
- But even if overfitting does not occur: How do we ensure that learned model also performs well on unseen data?

Common solutions

1. **Network-based adaptions** that modify the way we update the parameters
   - Early stopping, L2-Regularization, Dropout, Stochastic Depth
2. **Data-based adaptions** that modify the inputs
   - Data augmentation, RandAugment, mixup, cutout, cutmix

- General idea: Make it harder to overfit!
- Often combination of network-based and data-based adaptions

L2-Regularization/weight decay

- L2-regularization can reduce risk of overfitting

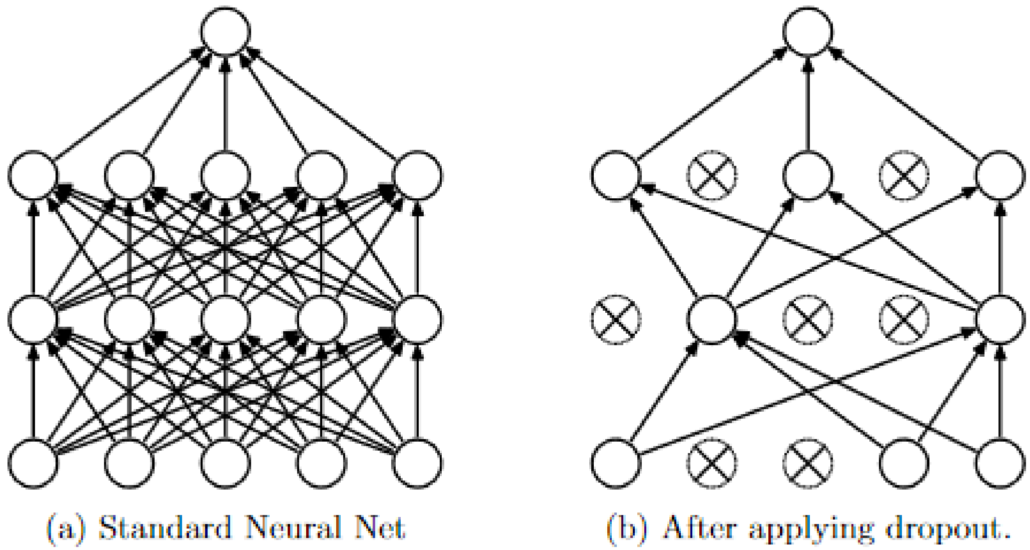
$$L(\theta) = \frac{1}{N}\sum_i \ell(y_i, f(\mathbf{x}_i;\theta)) \boxed{+ \lambda||\theta||^2}$$

- L2-Regularization commonly applied in gradient update, leads to so-called **weight decay**

$$\theta_{k+1} = \theta_k - \eta\frac{\partial L}{\partial \theta} \boxed{- \lambda\theta_k}$$

- Only apply to weights of convolutional and fully connected layers (biases and BN params are unregularized)
- Beware with Adam: L2-regularization != weight decay
  - See AdamW of [Loshchilov et al., 2019]

Dropout

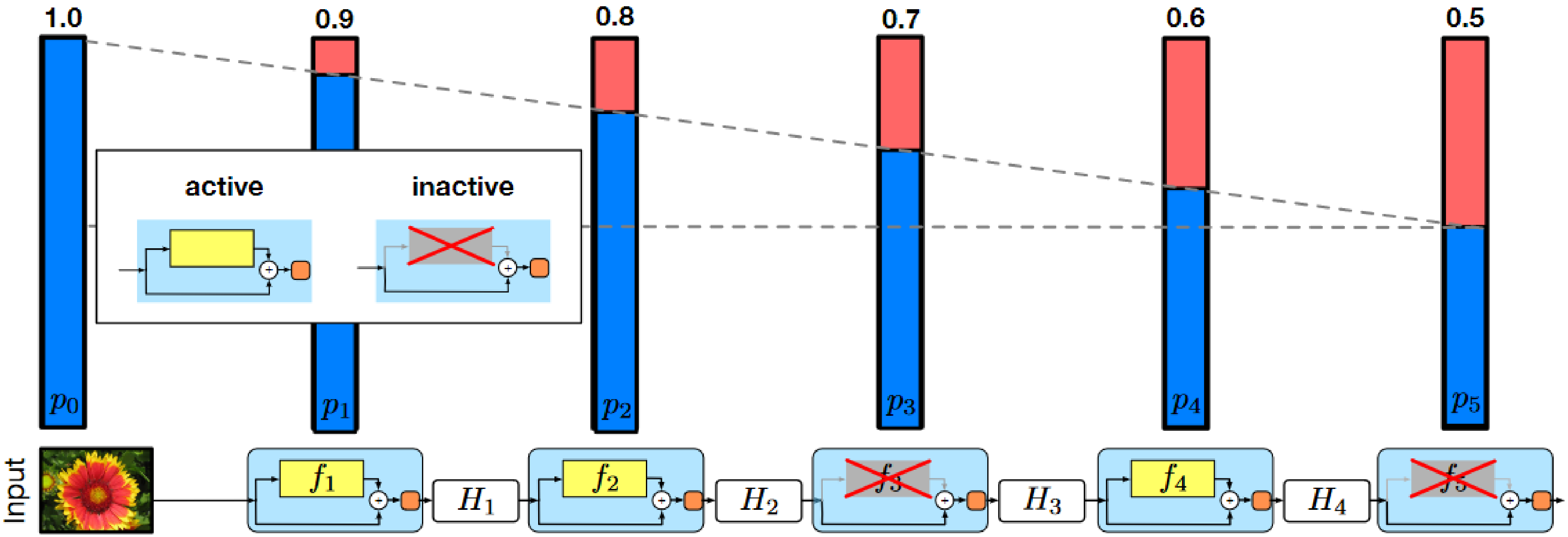


- Remove part of weights from network (usually applied in FC layers) randomly with probability p
- Helps to prevent co-adaption of weights
- At test time, we need to scale the activations by p

[Srivastava, 2014]

Figure from [Srivastava, 2014]

Stochastic Depth



- With increasing depth of the network only use skip connection
- Similar to dropout: implicit ensembles and scale functions according to probability

[Huang, 2016]

Figure from [Huang, 2016]

Data Augmentation


original

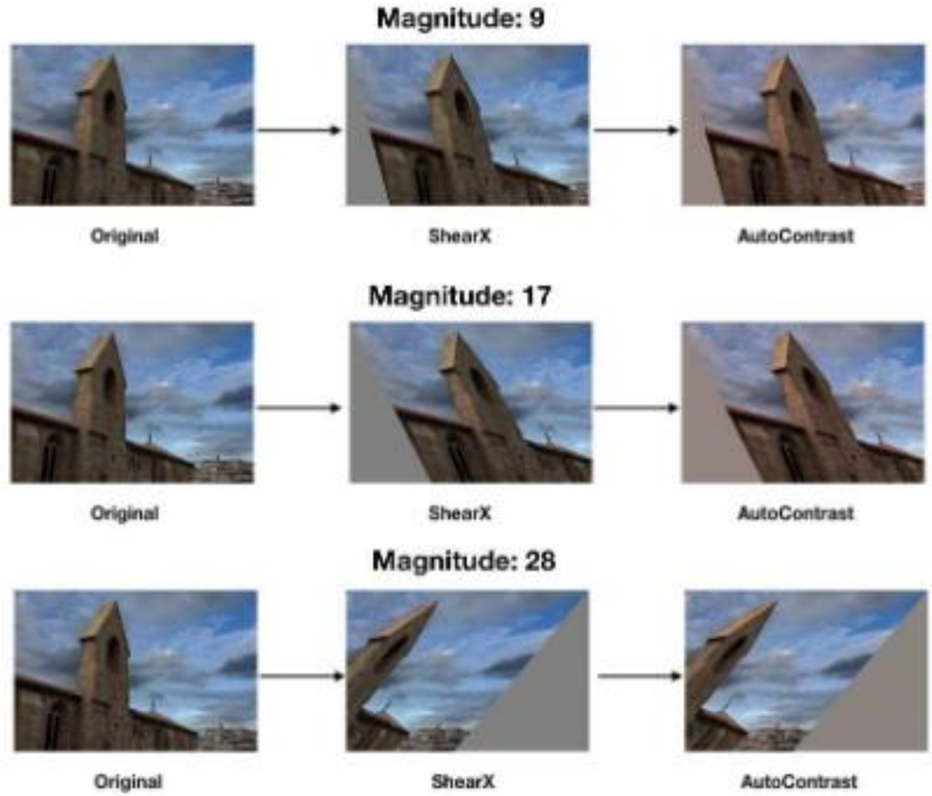

Horizonal mirror


Crop+resize


Color Distort (Saturation)

- Data Augmentation effectively increases training set size
- Augmented examples provide cases that network has to deal with
- Common augmentations: horizontal mirroring, color distortions, shearing, cropping, …

RandAugment



- Combine N different augmentations and scales effect by factor M (magnitude)
- N and M are hyperparameters that should be determined using grid search on validation set

[Cubuk, 2020]

Figure from [Cubuk, 2020]

RandAugment Results

Image Classification on ImageNet

	baseline	Fast AA	AA	RA
ResNet-50	76.3 / 93.1	**77.6** / 93.7	**77.6 / 93.8**	**77.6 / 93.8**
EfficientNet-B5	83.2 / 96.7	-	83.3 / 96.7	**83.9 / 96.8**
EfficientNet-B7	84.0 / 96.9	-	84.4 / 97.1	**85.0 / 97.2**

Object Detection on MS COCO

augmentation	search space	ResNet-101	ResNet-200
Baseline	0	38.8	39.9
AutoAugment	10^{34}	**40.4**	**42.1**
RandAugment	10^{2}	40.1	41.9

- Better or on-par with other augmentation techniques for different tasks

[Cubuk, 2020]

mixup

Image

Label

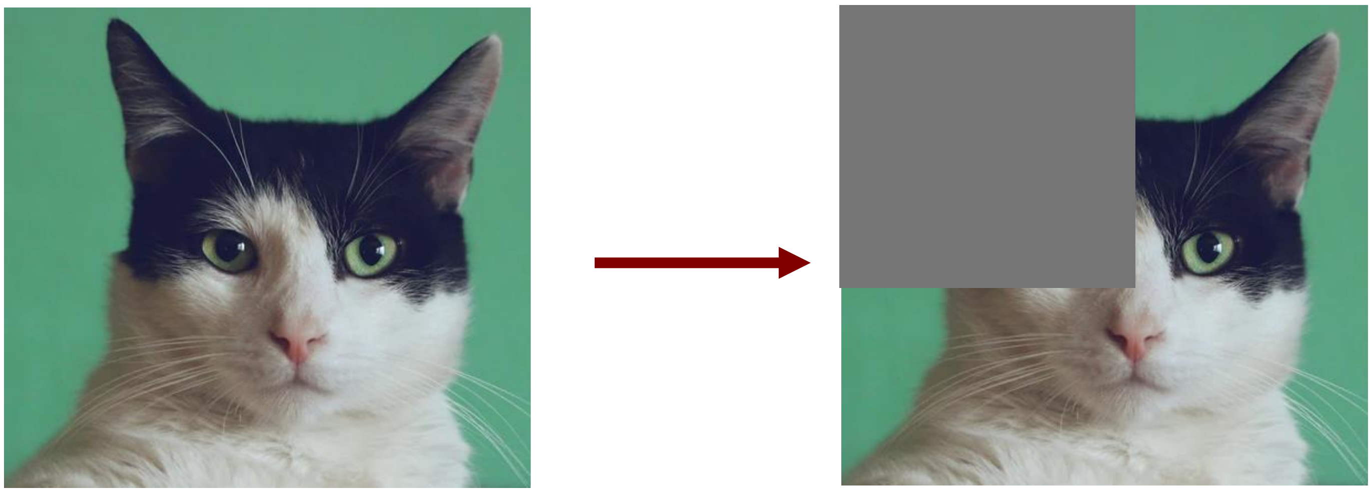
Cat 0.6

Dog 0.4

- Linear combination of images and target values
- Assumption: linear combinations of inputs should lead to linear interpolations of associated targets

[Zhang, 2018] Cat photo by Manja Vitolic and Dog photo by Pauline Loroy on Unsplash

Cutout



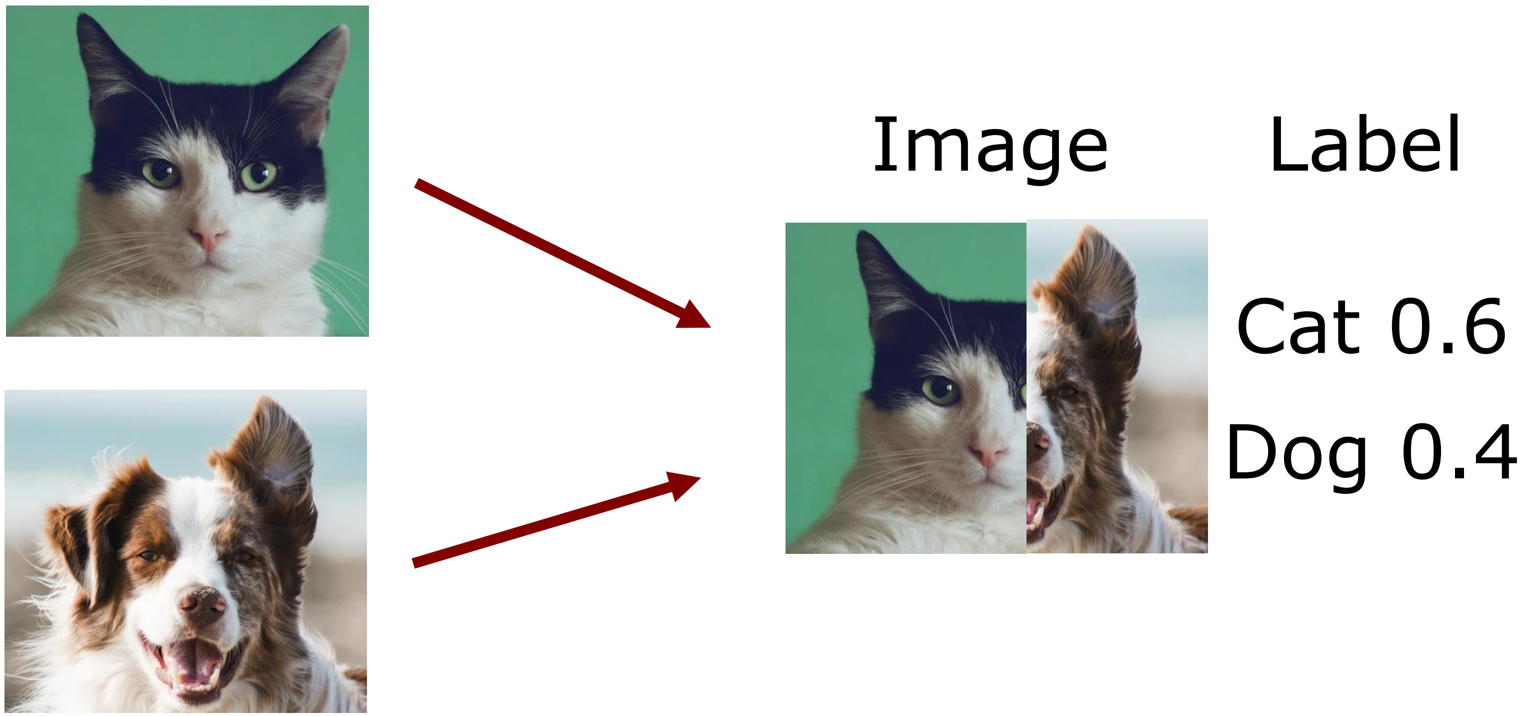
Cat 1.0

- Similar to Dropout but on the input level
- Randomly cut parts of the image and replace with mean

[DeVries, 2017]

Cat photo by [Manja Vitolic](#) on [Unsplash](#)

Cutmix



- Crop part of images and combine images into new image
- Mix labels according to the proportion (cf. mixup)

[Yun, 2019] Cat photo by Manja Vitolic and Dog photo by Pauline Loroy on Unsplash

Effect of Data-dependent Augmentations

	ResNet-50	Mixup	Cutout	CutMix
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.6 (+2.3)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3 (+1.0)**
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	**76.7 (+1.1)**

- All method improve classification performance
- Cutmix has advantages for object detection

[Yun, 2019]

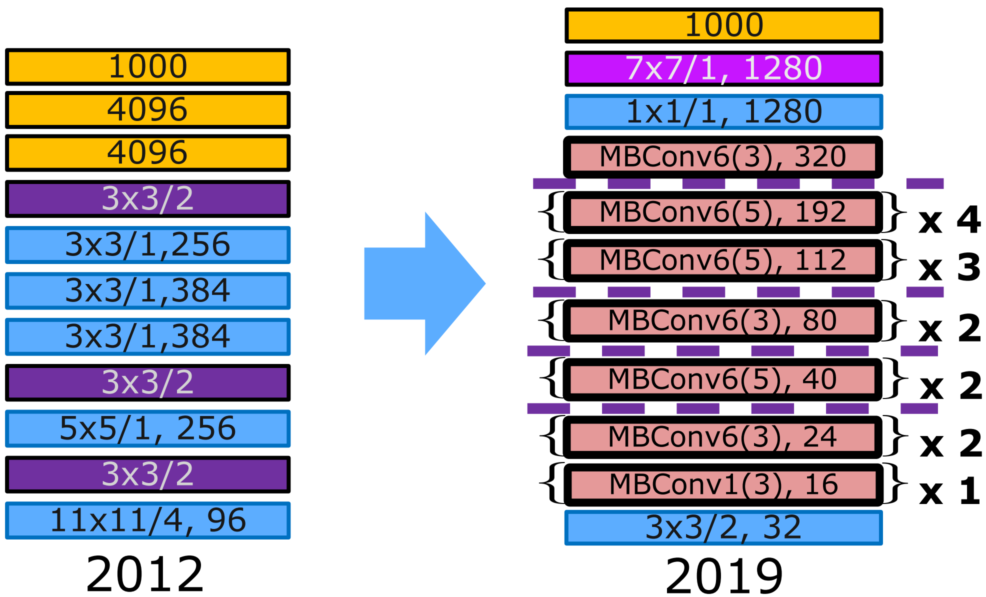
Common Combinations

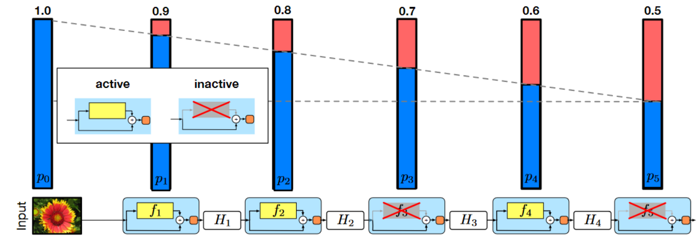
- Often multiple methods used in combination
  - EfficientNet: AutoAugment, StochasticDepth, Dropout
  - EfficientNetV2: RandAugment -> Mixup
  - NFNets: Mixup (0.2) + CutMix → RandAugment (N=4)

- But similar to hyper parameters, these are usually highly data-dependent ("no free lunch")

[Tan, 2019] [Tan, 2021] [Brock, 2021]

Summary





- Popular and significant architectures and changes
  - global avg pooling → skip connection → efficiency
  - VGG → GoogleLeNet → ResNet → MobileNetV2 → EfficientNet

- Looked at common ways to close the gap between training and test performance (generalization gap)

See you next week!

References

Bello et al. *Revisiting ResNets: Improved Training and Scaling Strategies*, Arxiv, 2021.

Brock et al. *High-Performance Large-Scale Image Recognition Without Normalization*, arxiv, 2021.

Cubuk et al. *Randaugment: Practical automated data augmentation with a reduced search space* , CVPR, 2020.

DeVries et al. Improved Regularization of Convolutional Neural Networks with Cutout, arxiv, 2017.

Elsken et al. *Neural Architecture Search: A Survey*, JMLR 20, pp. 1-21, 2019.

He et al. *Deep Residual Learning for Image Recognition*, CVPR, 2016.

Huang et al. Deep Networks with Stochastic Depth. ECCV, 2016.

Loshchilov et al. *Decoupled Weight Decay Regularization.* ICLR, 2019.

Real et al. *Regularized Evolution for Image Classifier Architecture Search*, AAAI, 2019.

Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. CVPR, 2018.

References

Simonyan et al. *Very Deep Convolutional Networks for Large-Scale Image Recognition*, ICLR, 2015.

Srivastava et al. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting, JMLR 15(56): 1929-1958*

Szegedy et al. *Going deeper with convolutions*. CVPR, 2015.

Tan et al. *EfficientNet: Rethinking model scaling for convolutional neural networks*, ICML, 2019.

Tan et al. *EfficientNetV2: Smaller Models and Faster Training*, ICML, 2021.

Yun et al*. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*, ICCV, 2019.

Zhang et al. *mixup: Beyond empirical risk minimization*, ICLR, 2018.