Neural Networks and Deep Learning

Architectures for Image Classification

Slides credit: Pau Rodriguez, Ariadna Romero

What´s in a scene

Type of environment	Indoor
Scene categories	martial_arts_gym (0.204)
Scene attributes	no horizon, enclosed area, man-made, cloth, natural light, competing, exercise, sports, glossy
Informative región for predicting the top category	

Try it yourself: http://places2.csail.mit.edu/demo.html

What´s in a scene

Type of environment	Indoor	Outdoor
Scene categories	martial_arts_gym (0.204)	apartment_building/outdoor (0.195), office_building (0.140), hospital (0.131), parking_garage/outdoor (0.126)
Scene attributes	no horizon, enclosed area, man-made, cloth, natural light, competing, exercise, sports, glossy	natural light, man-made, open area, no horizon, sunny, glass, vegetation, vertical components, leaves
Informative región for predicting the top category		

Try it yourself: [*http://places2.csail.mit.edu/demo.html*](http://places2.csail.mit.edu/demo.html)

What´s in a scene

Binary classification

Type of environment	Indoor	Outdoor
Scene categories	martial_arts_gym (0.204)	apartment_building/outdoor (0.195), office_building (0.140), hospital (0.131), parking_garage/outdoor (0.126)
Scene attributes	no horizon, enclosed area, man-made, cloth, natural light, competing, exercise, sports, glossy	natural light, man-made, open area, no horizon, sunny, glass, vegetation, vertical components, leaves
Informative región for predicting the top category		

Try it yourself: [*http://places2.csail.mit.edu/demo.html*](http://places2.csail.mit.edu/demo.html)

What´s in a scene

Multi-class classification

		
Type of environment	Indoor	Outdoor
Scene categories	martial_arts_gym (0.204)	apartment_building/outdoor (0.195), office_building (0.140), hospital (0.131), parking_garage/outdoor (0.126)
Scene attributes	no horizon, enclosed area, man-made, cloth, natural light, competing, exercise, sports, glossy	natural light, man-made, open area, no horizon, sunny, glass, vegetation, vertical components, leaves
Informative región for predicting the top category		

Try it yourself: http://places2.csail.mit.edu/demo.html

What´s in a scene

Multi-label classification

		
Type of environment	Indoor	Outdoor
Scene categories	martial_arts_gym (0.204)	apartment_building/outdoor (0.195), office_building (0.140), hospital (0.131), parking_garage/outdoor (0.126)
Scene attributes	no horizon, enclosed area, man-made, cloth, natural light, competing, exercise, sports, glossy	natural light, man-made, open area, no horizon, sunny, glass, vegetation, vertical components, leaves
Informative región for predicting the top category		

Try it yourself: http://places2.csail.mit.edu/demo.html

What´s in a scene

Understanding / visualising CNNs

Type of environment	Indoor	Outdoor
Scene categories	martial_arts_gym (0.204)	apartment_building/outdoor (0.195), office_building (0.140), hospital (0.131), parking_garage/outdoor (0.126)
Scene attributes	no horizon, enclosed area, man-made, cloth, natural light, competing, exercise, sports, glossy	natural light, man-made, open area, no horizon, sunny, glass, vegetation, vertical components, leaves
Informative región for predicting the top category		

Try it yourself: http://places2.csail.mit.edu/demo.html

Image Classification

PROBLEM METRICS AND DATASETS

Evaluation Metrics

$$\textbf{Accuracy} = \frac{\#\ correct\ predictions}{\#\ total\ predictions}$$

$$\textbf{Misclassification Error} = 1 - Accuracy$$

Scene categories	**apartment_building/outdoor** (0.195), office_building (0.140), hospital (0.131), parking_garage/outdoor (0.126)

Evaluation Metrics

$$\textbf{Accuracy} = \frac{\#\ correct\ predictions}{\#\ total\ predictions}$$

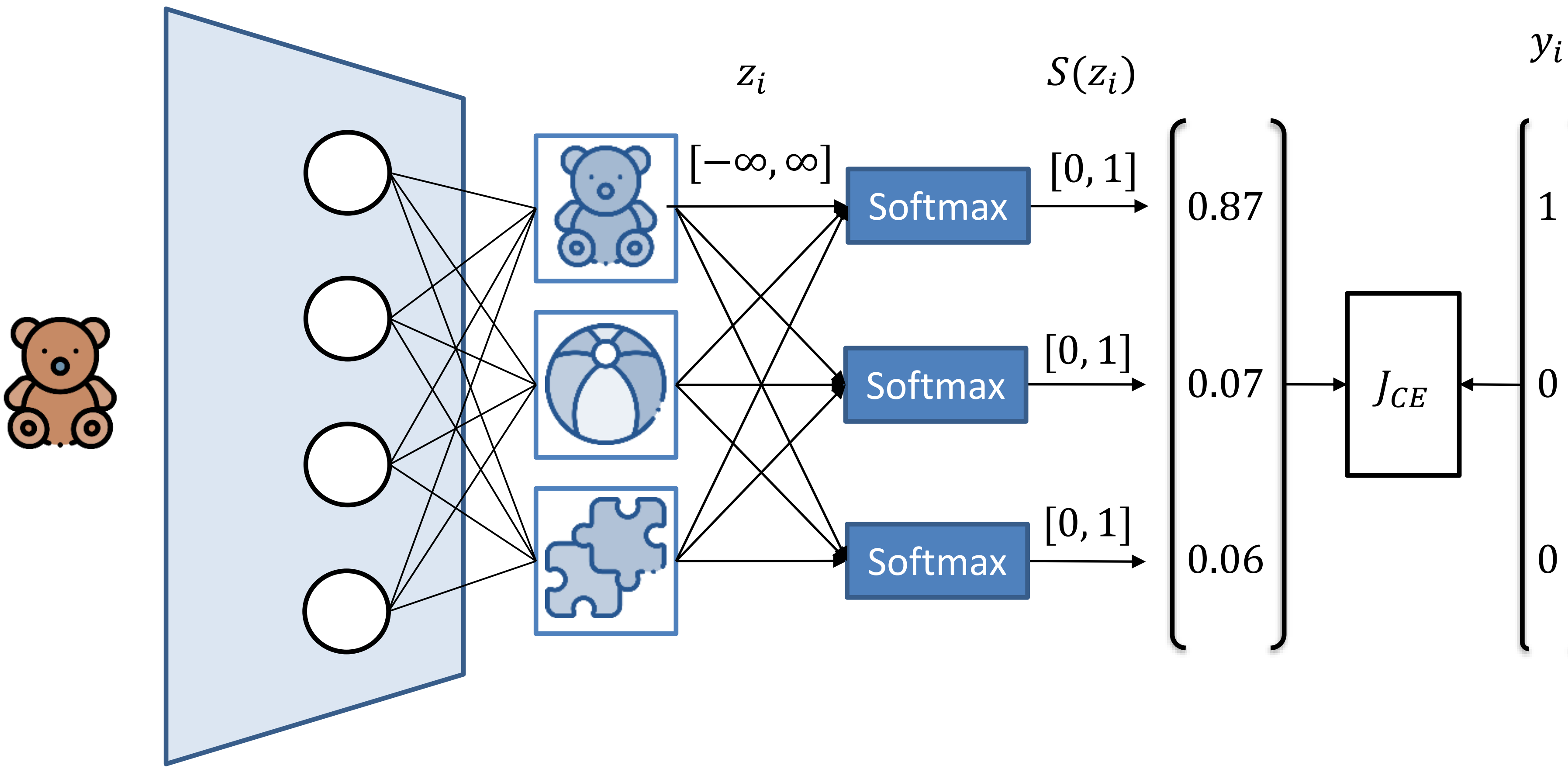
Scene categories	**apartment_building/outdoor** (0.195), **office_building** (0.140), **hospital** (0.131), **parking_garage/outdoor** (0.126)

$$\textbf{Misclassification Error} = 1 - Accuracy$$

$$\textbf{Top−k Accuracy} = \frac{\#\ correct\ prediction\ in\ the\ top\ k}{\#\ total\ predictions}$$

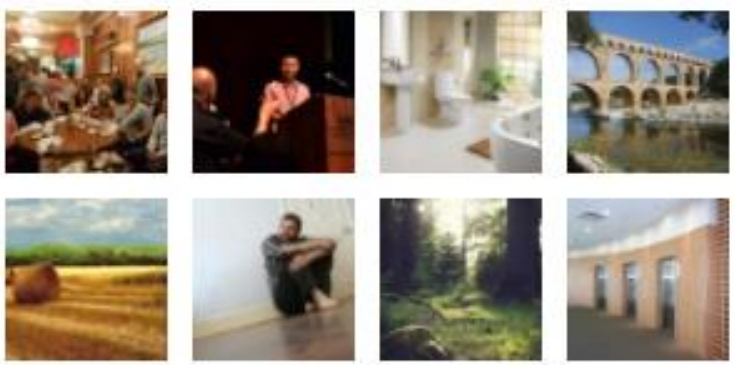
$$\textbf{Top−k Classification Error} = 1 - \text{Top−k Accuracy}$$

Multi-class classification



Datasets

- Scene photographs from image search engines, non-uniform distribution of images per category.
- 8M train, 36K val, 328K test, 1000 classes.



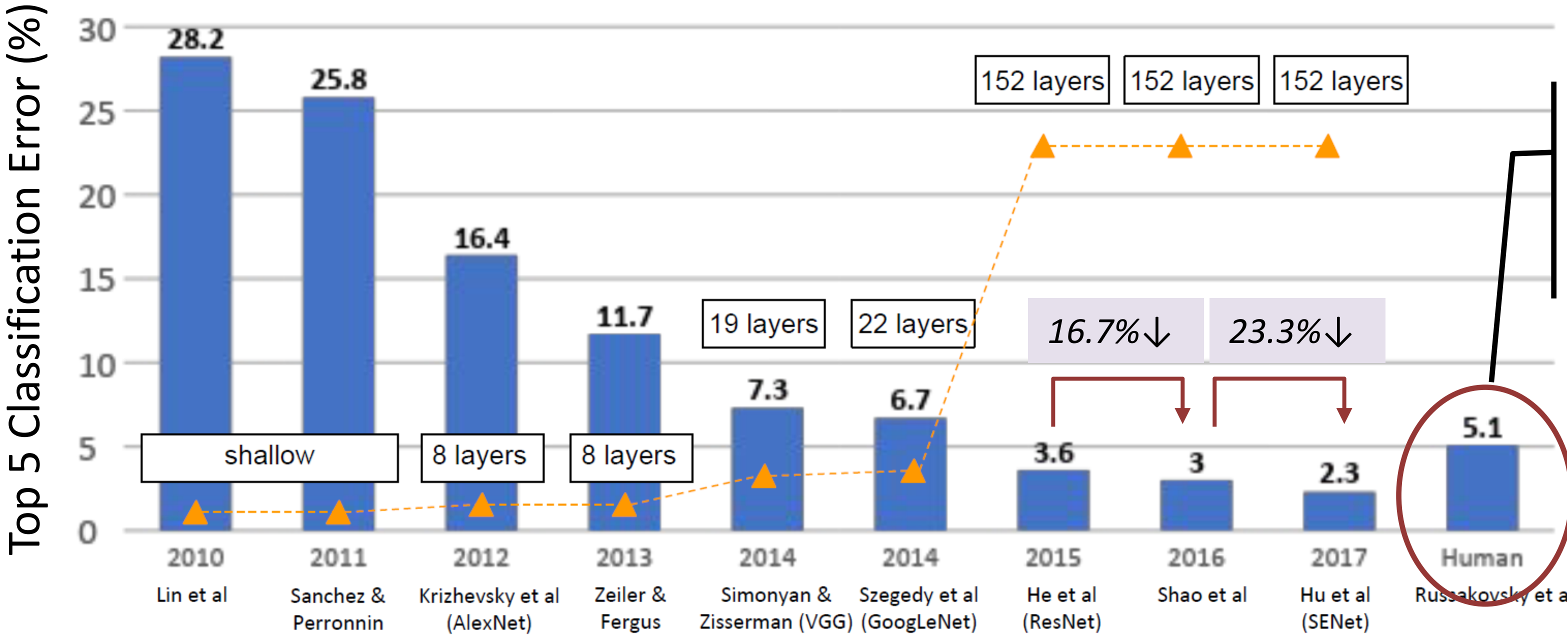
from ILSVRC 2016, Places Dataset

- Photos of 91 object types in the context of the broader question of scene understanding that would be easily recognizable by a 4 year old.
- 2.5 million labeled instances in 328k images.

from the COCO Dataset

ImageNet Large Scale Visual Recognition Challenge

- Standard benchmark for image classification
- Standardised evaluation (top-k classification error)
- Evolution over time



[Try it](#) out yourself

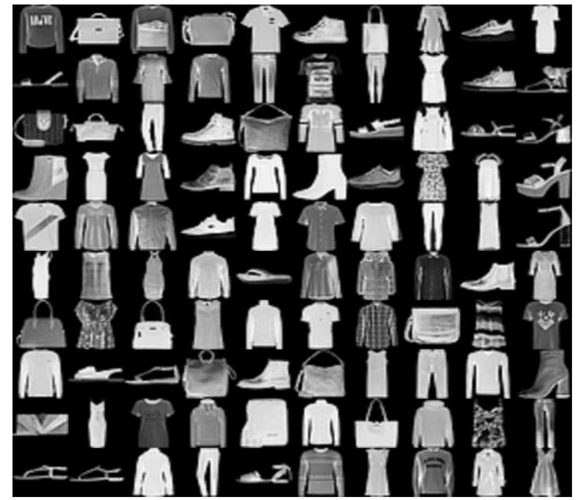
ILSVRC: https://image-net.org/challenges/LSVRC/2016/#scene

Prototype Datasets

MNIST

- 60000 images of 10 handwritten digits of 28x28 pixels
- Accuracy: 99.9 %
- Usage: test if your model runs correctly

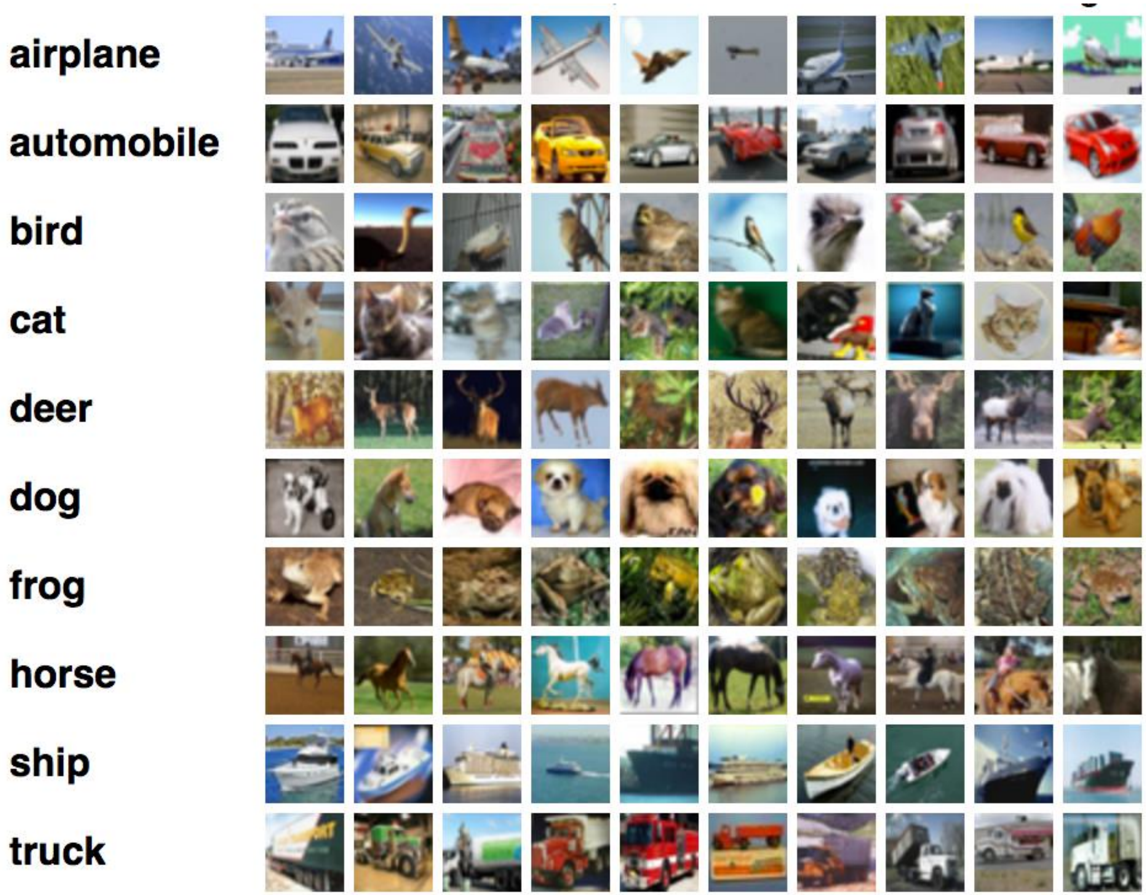
Fashion-Mnist



- 50000 images of 10 different clothes from Zalando of 28x28 pixels
- Accuracy: 96 %
- Usage: test if your model runs correctly

Prototype Datasets

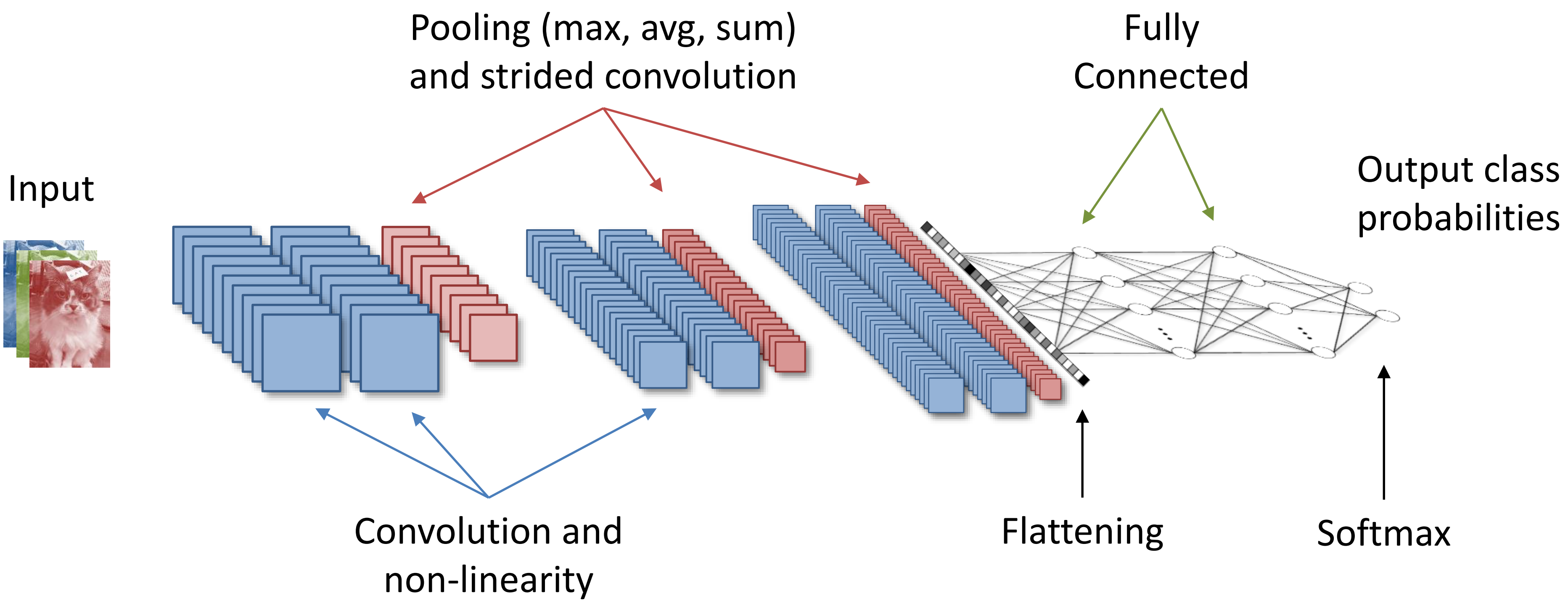
CIFAR



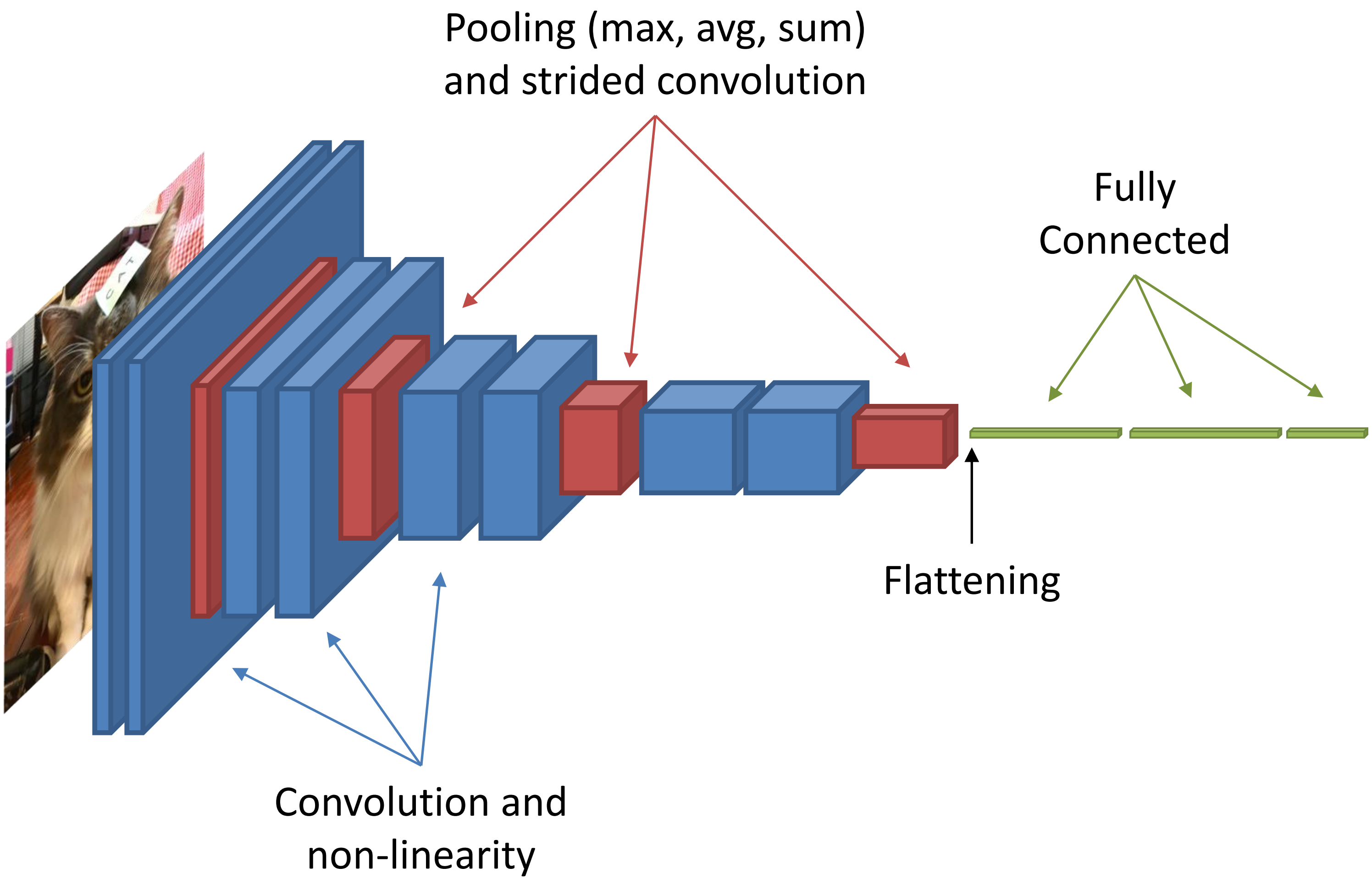
- 60000 images of 32x32 color pixels
- Organized in 10 or 100 classes (CIFAR10 and CIFAR100)

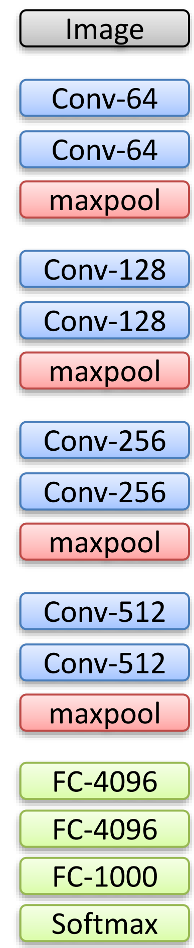
BASIC CNN MODELS

Typical CNN architecture

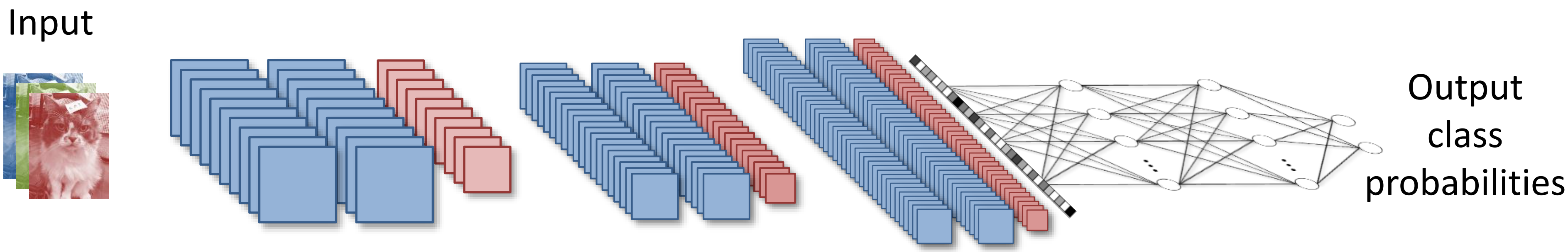


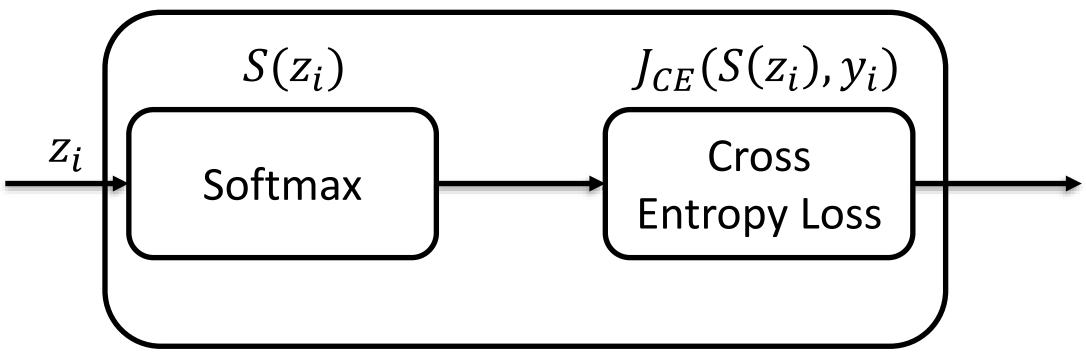
Different CNN Diagrams





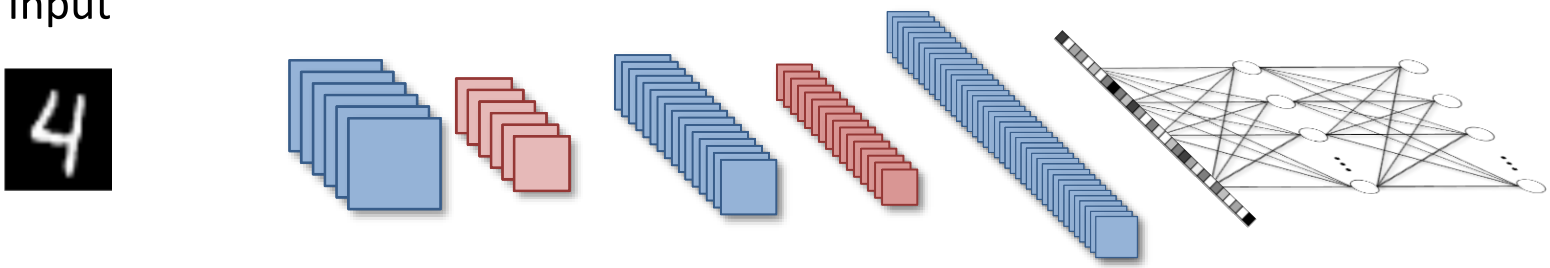
CNN architecture





$$S(z)_i = \frac{e^{z_i}}{\sum_j^C e^{z_j}} \qquad J_{CCE} = -\sum_i^C y_c \log(S(z)_i)$$

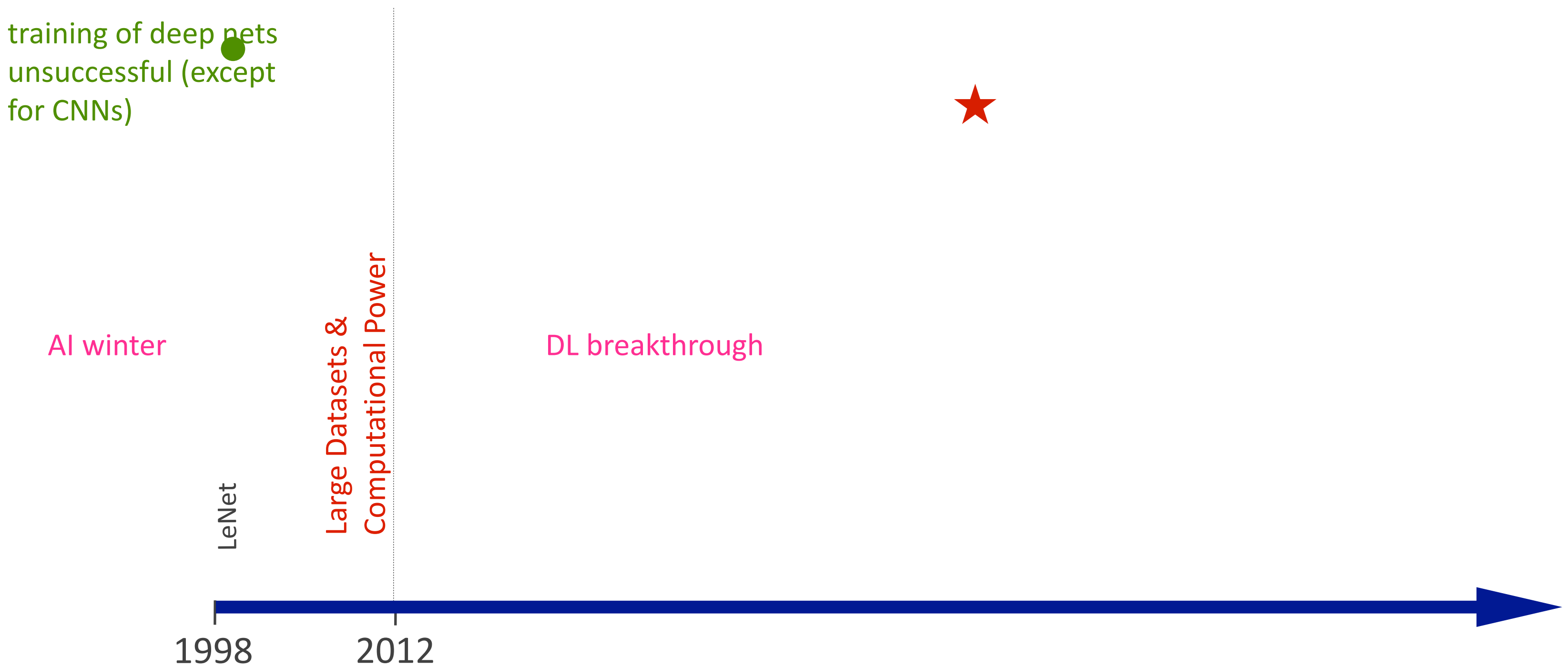
LeNet (1998)



Nonlinearity used: $tanh$

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural Computation

Image Classification Evolution



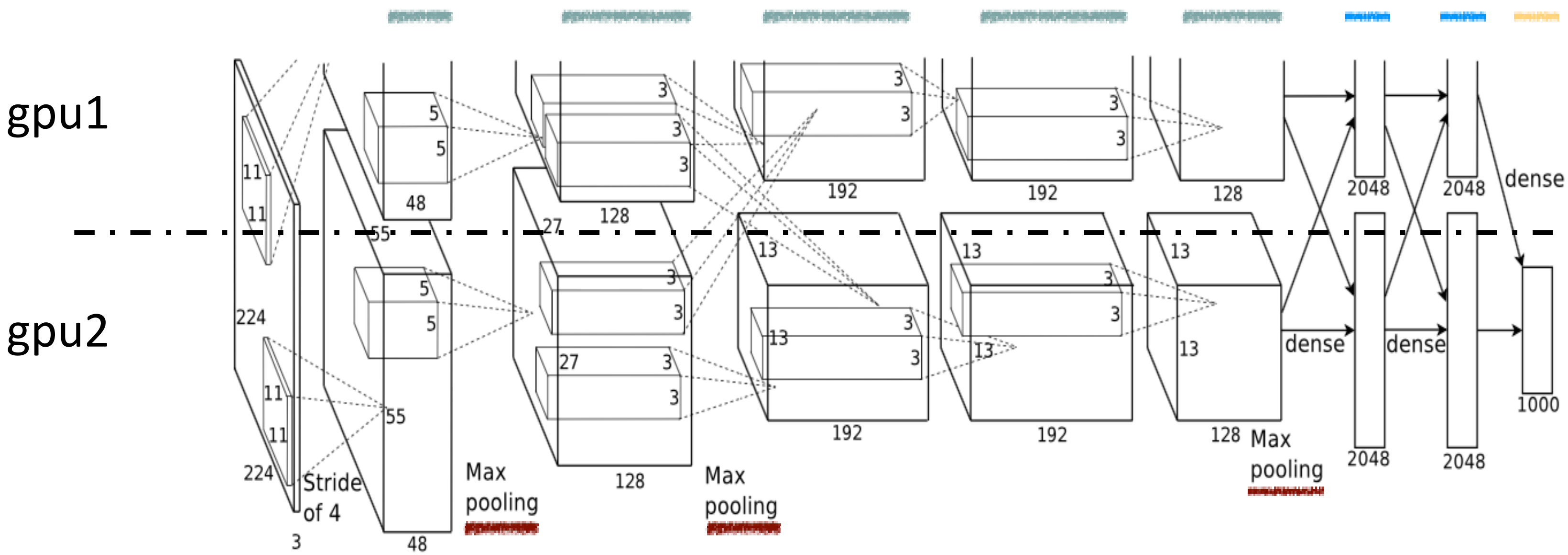
Slide credit: Pau Rodriguez and Adriana Romero. MCV M3 Architectures for image classification.

AlexNet, VGG, GoogLeNet

2012 - 2014

AlexNet (2012)

8 convolutional (11x11, 5x5, 3x3, 3x3, 3x3) and 3 fully-connected layers.
Took ~6 days on two GTX 580 3GB GPUs (own implementation)



Key concepts introduced to mainstream CNN methodology:

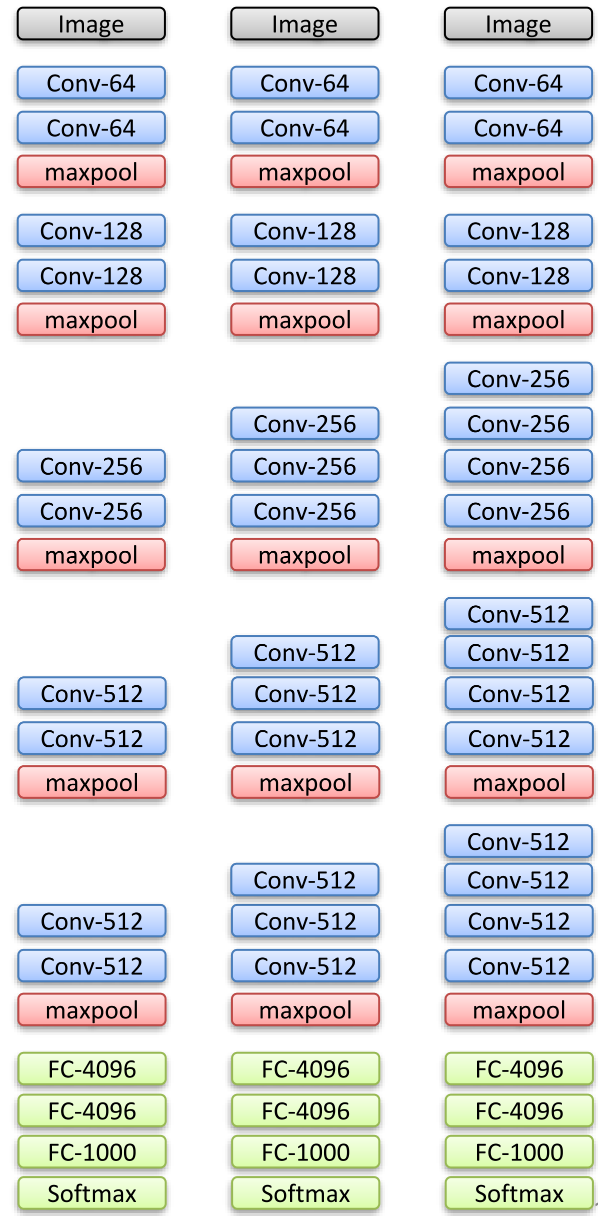
- **ReLU non-linearity** [see here for details]
- **Dropout** [see here for details]
- **Data Augmentation**

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

VGG (2014)

“Our main contribution is a thorough evaluation of **networks of increasing depth** using an architecture with **very small (3 × 3) convolution** filters […] significant improvement can be achieved by **pushing the depth to 16–19** weight layers.”

- “Very” deep network (up to 19 layers)
- **3x3 filters**, stride 1, padding 1
- **Stacked** convolutions
- 2x2 non-overlapping max-pooling
- # features increases as we go deeper
- ReLU non-linearity
- Data Augmentation
- Dropout



Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

VGG (2014)

Why 3x3 layers?

What is the receptive field if we stack two 3×3 convolutions?

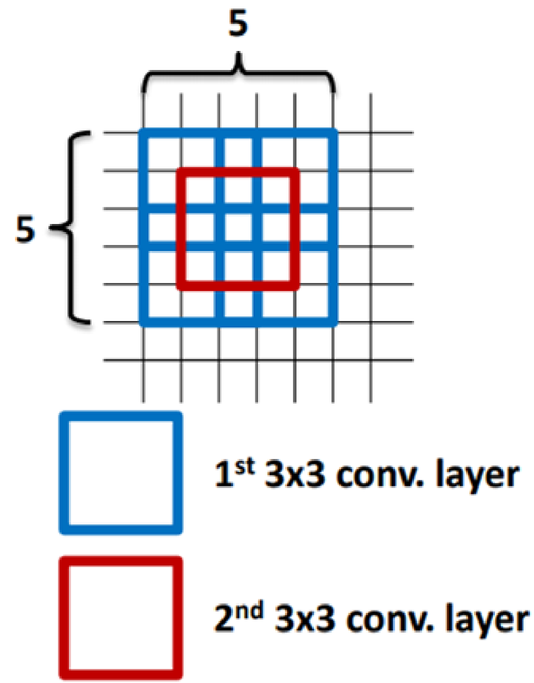
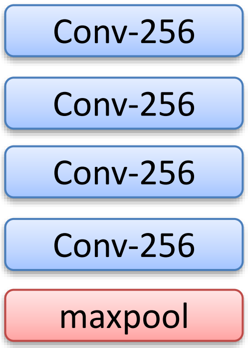


Image credit: http://www.robots.ox.ac.uk/~karen/pdf/ILSVRC_2014.pdf

VGG (2014)

Why 3x3 layers?

How many parameters are in a layer if we use (a) 3×3 filters? How many if we use (b) 5×5 filters?



(a) $3 \times 3 \times C \times C + C \cong 590\text{k}$ (for $C = 256$)

(b) $5 \times 5 \times C \times C + C \cong 1.6\text{M}$ (for $C = 256$)

VGG (2014)

Why 3x3 layers?

- Stacked conv. layers have a large receptive field
  - two 3x3 layers: *5x5 receptive field*
  - three 3x3 layers: *7x7 receptive field*
- More non-linearity
- Less parameters to learn (compared to using large kernels)
  - ~140M per net

GoogLeNet (2015)

GoogLeNet (2015)

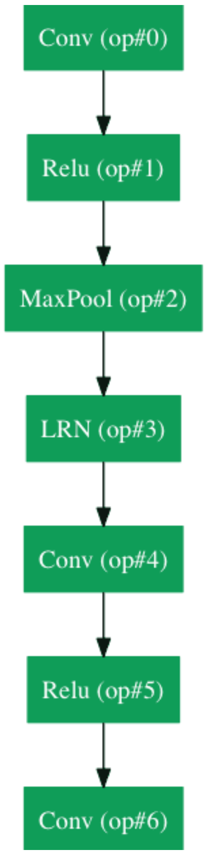
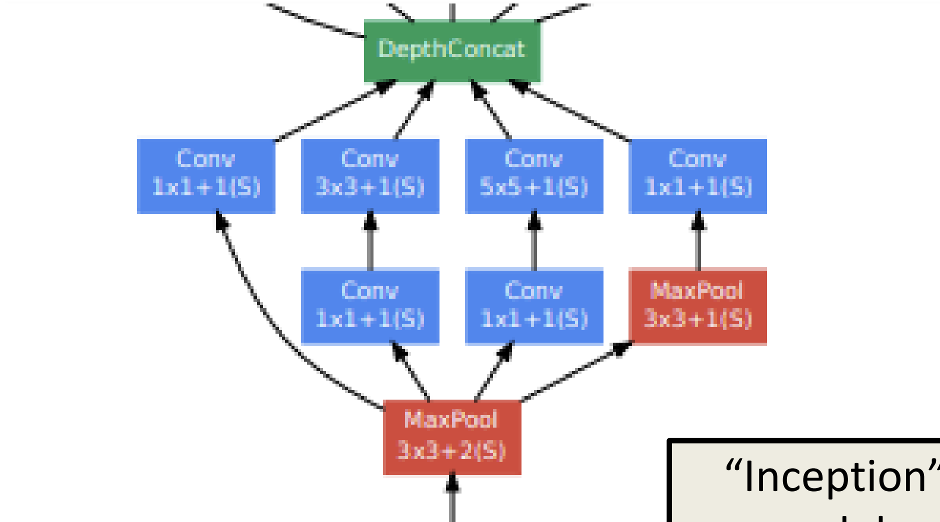
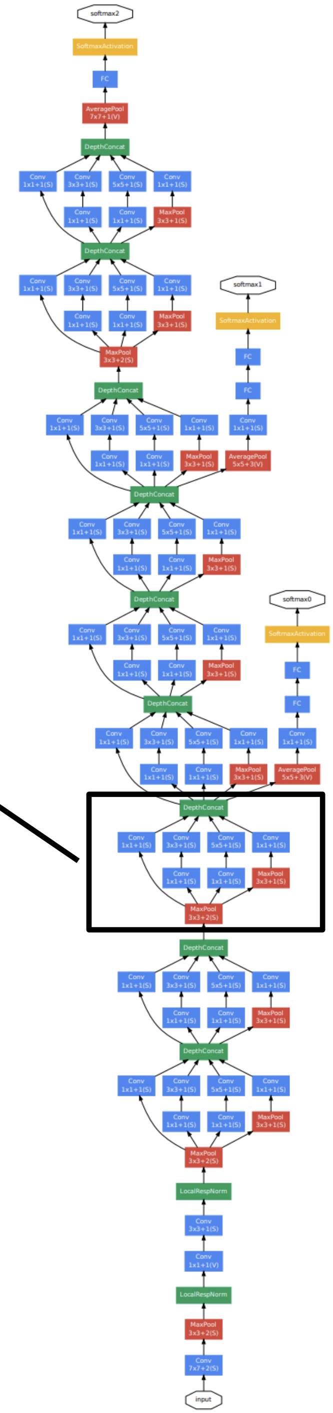


Image credit: [The 9 Deep Learning Papers You Need To Know About, Adit Deshpande]

GoogLeNet (2015)



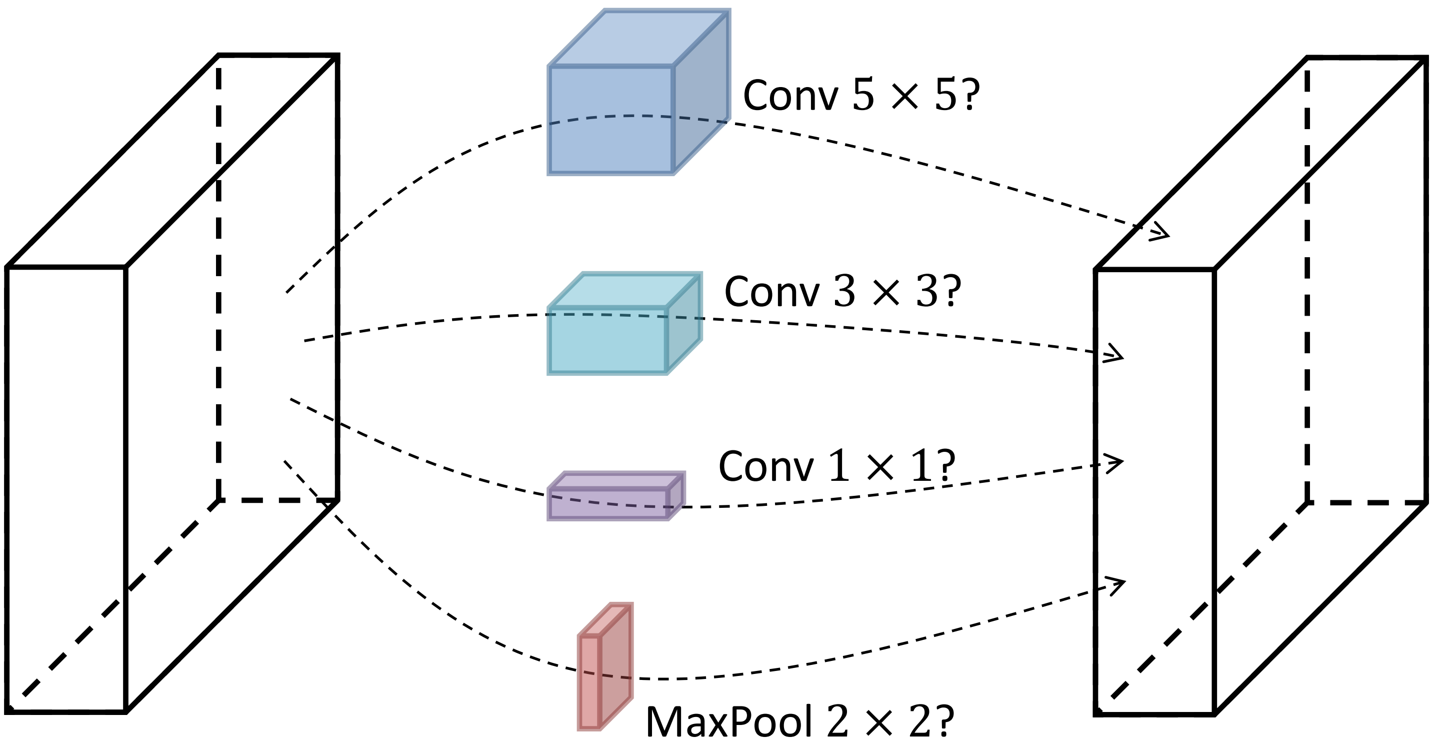
"Inception" module



- Convolution / FC
- Max / Avg Pooling
- Softmax
- Concatenation

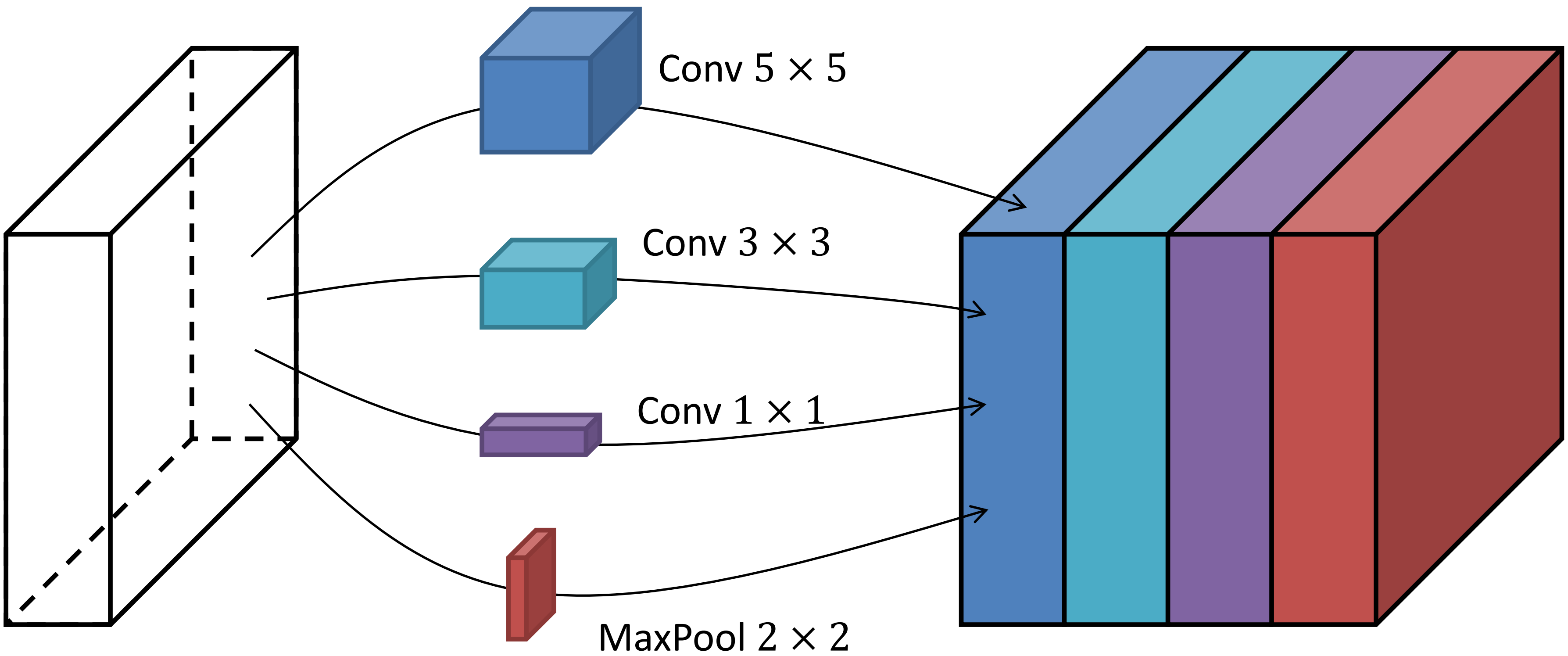
Szegedy et al, "Going deeper with convolutions", 2014

GoogLeNet (2015) – Inception Module



Which operation should we choose?

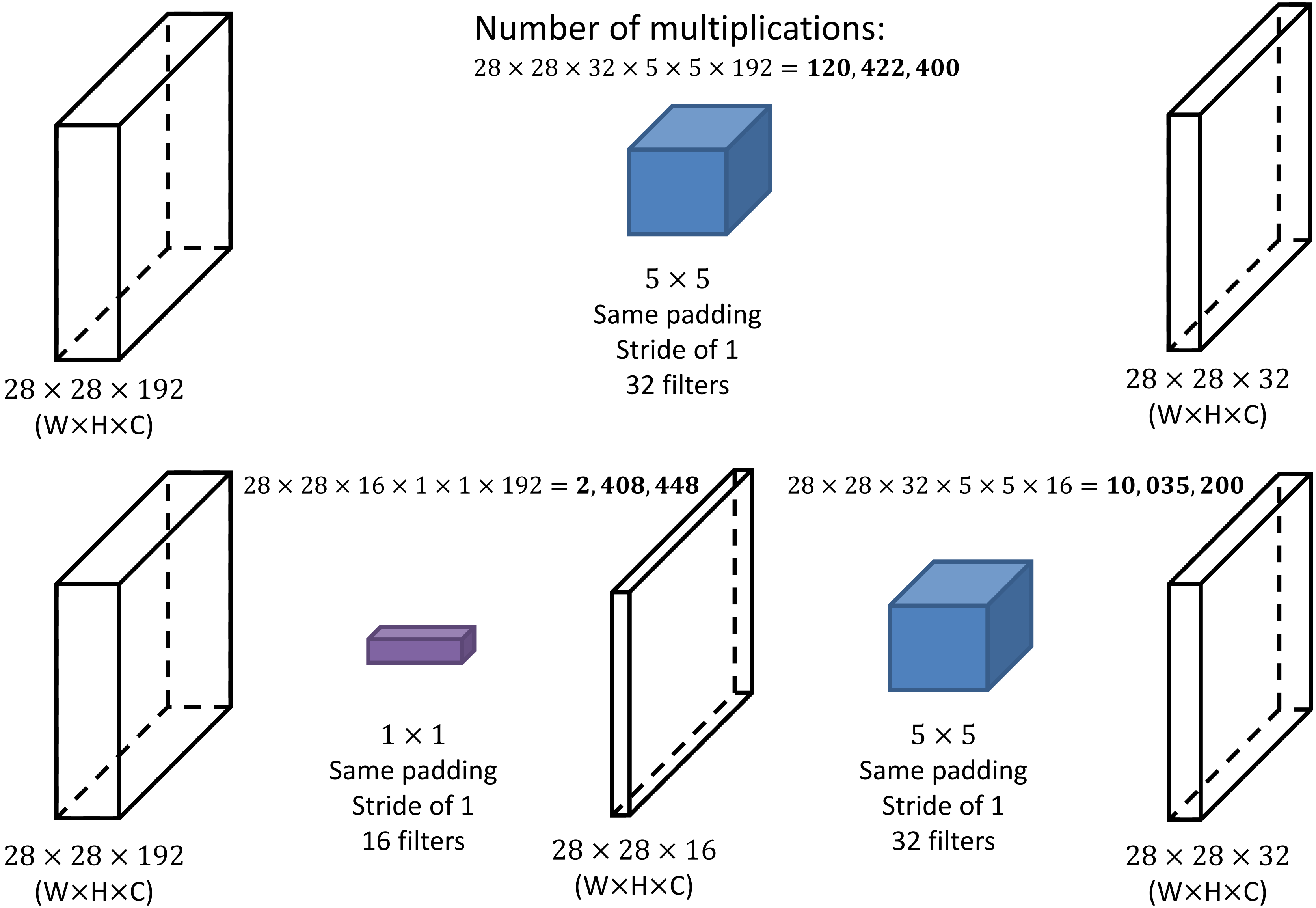
GoogLeNet (2015) – Inception Module



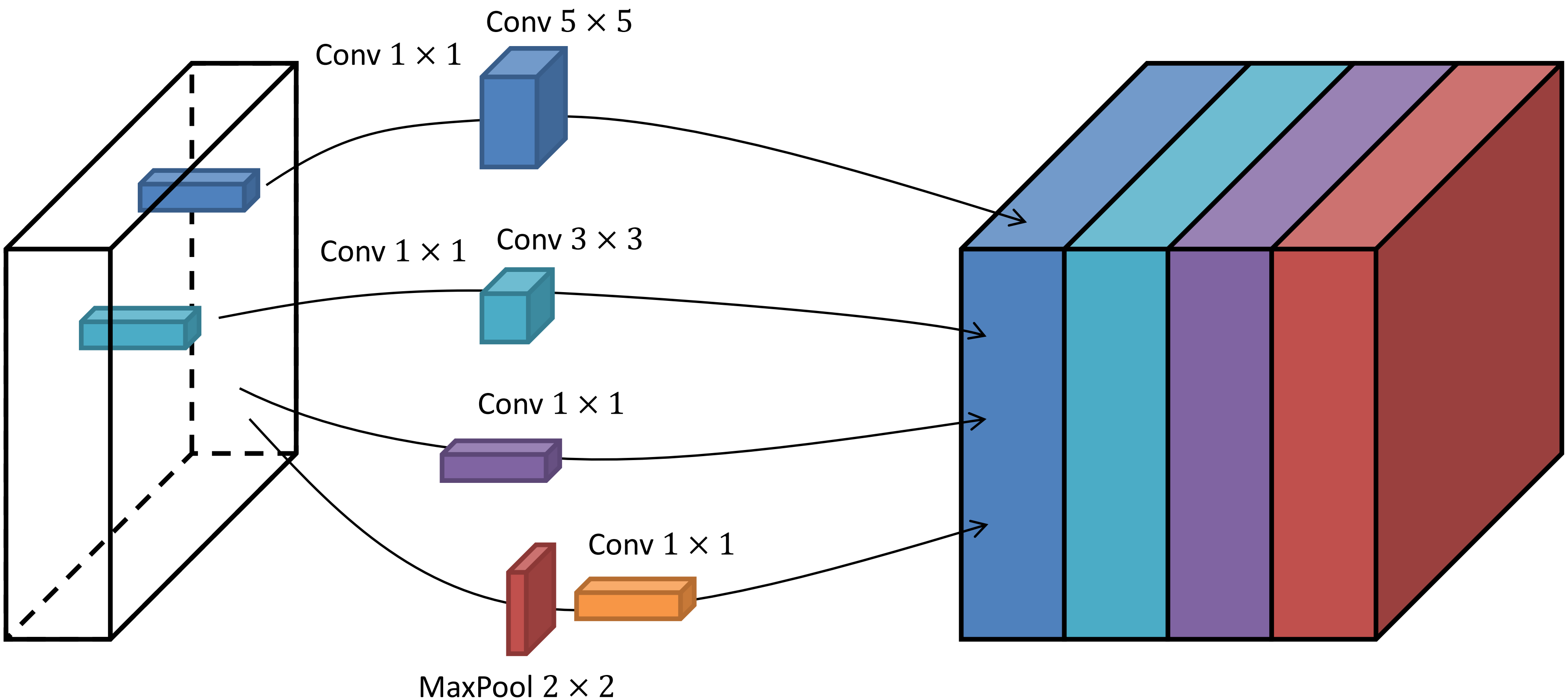
Let's do them all at once! Multi-scale

But the size explodes…

Reducing the Computational Cost



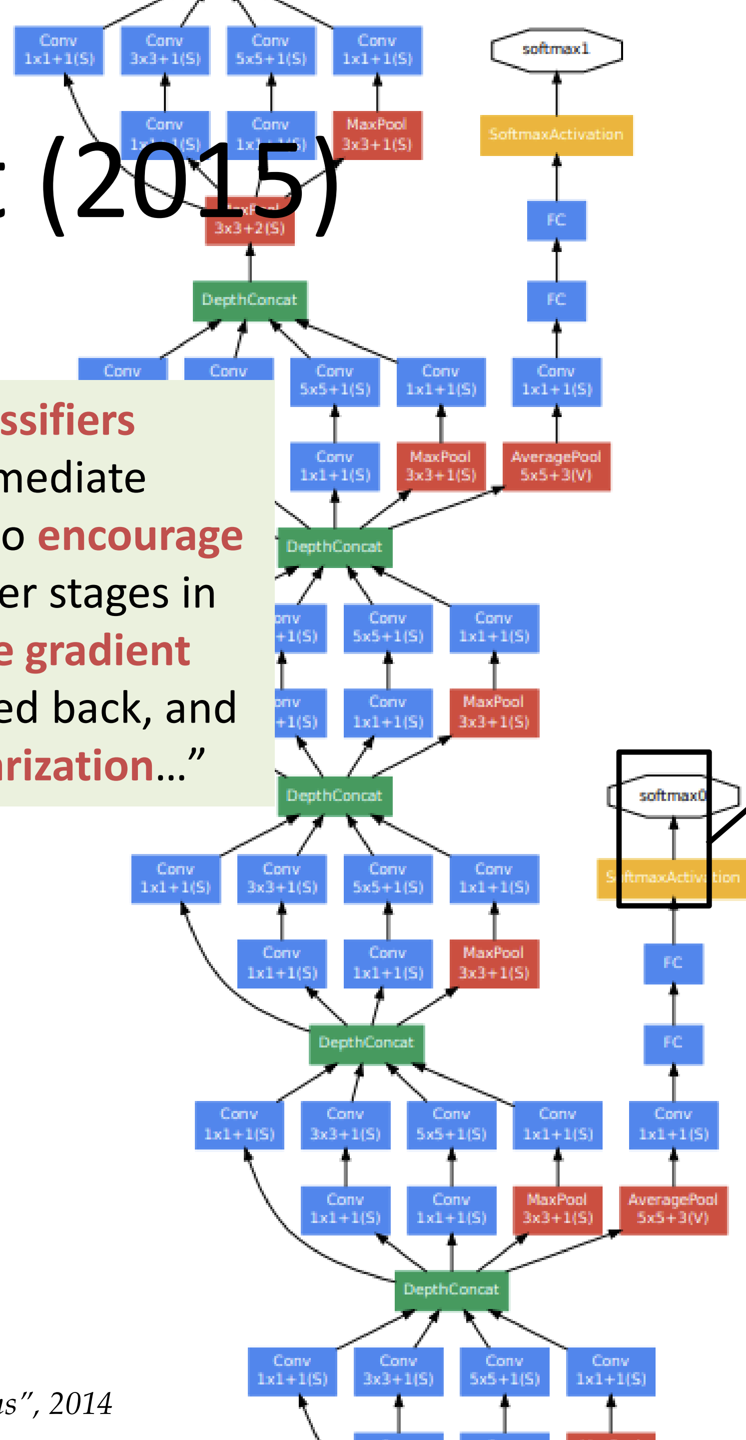
GoogLeNet (2015) – Inception Module



Reduce the number of channels, using 1×1 convolutions

1×1 *convolutions introduced in: Lin et al,* *[Network In Network](#)*, , *ICLR 2014*

GoogLeNet (2015)



"...By adding **auxiliary classifiers** connected to these intermediate layers, we would expect to **encourage discrimination** in the lower stages in the classifier, **increase the gradient signal** that gets propagated back, and provide **additional regularization**..."

Auxiliary supervision

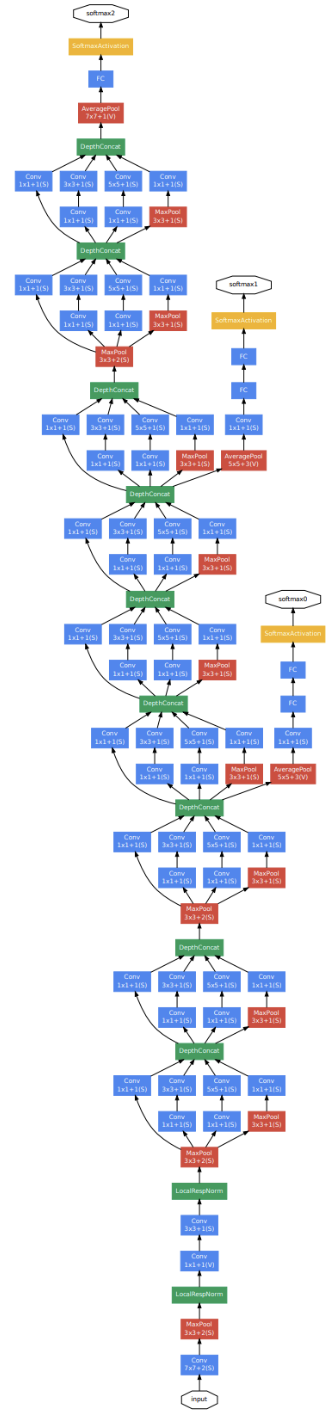
- Convolution / FC
- Max / Avg Pooling
- Softmax
- Concatenation

Szegedy et al, "Going deeper with convolutions", 2014

GoogLeNet (2015)

22-layer deep CNN, but reduced the number of parameters **from 60 million (AlexNet) to 4 million (12x)**

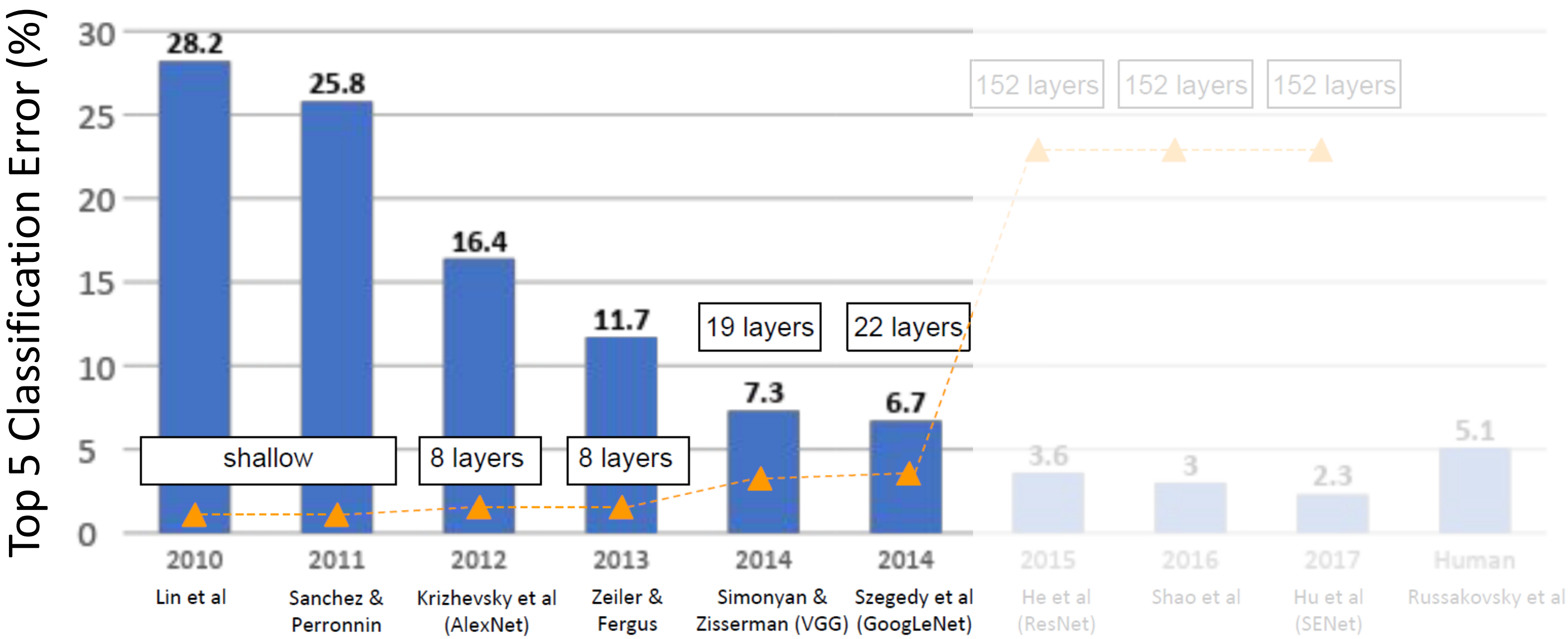
Low use of fully connected layers, they use average pooling instead



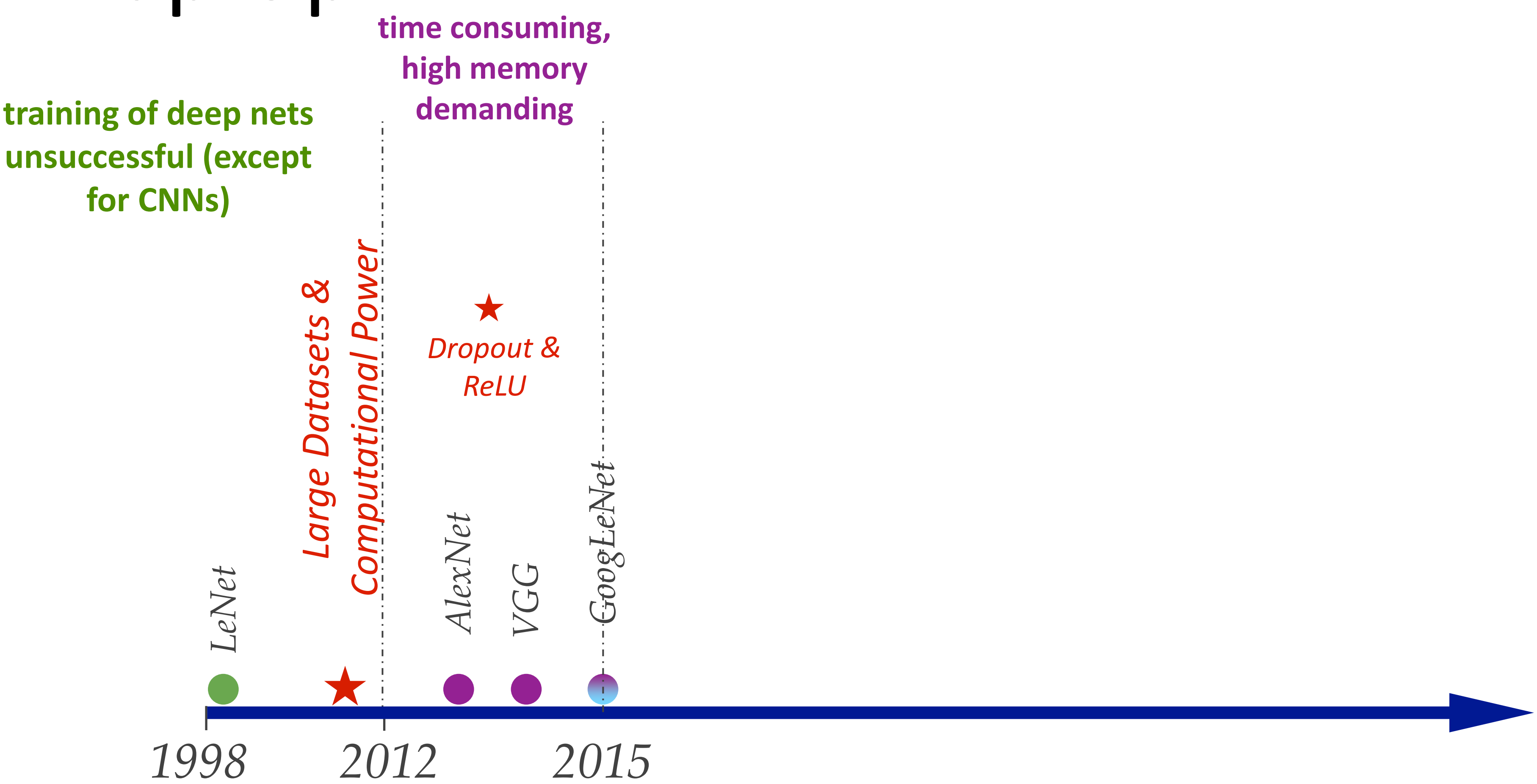
- Convolution / FC
- Max / Avg Pooling
- Softmax
- Concatenation

Szegedy et al, "Going deeper with convolutions", 2014

ImageNet Classification



Wrap Up



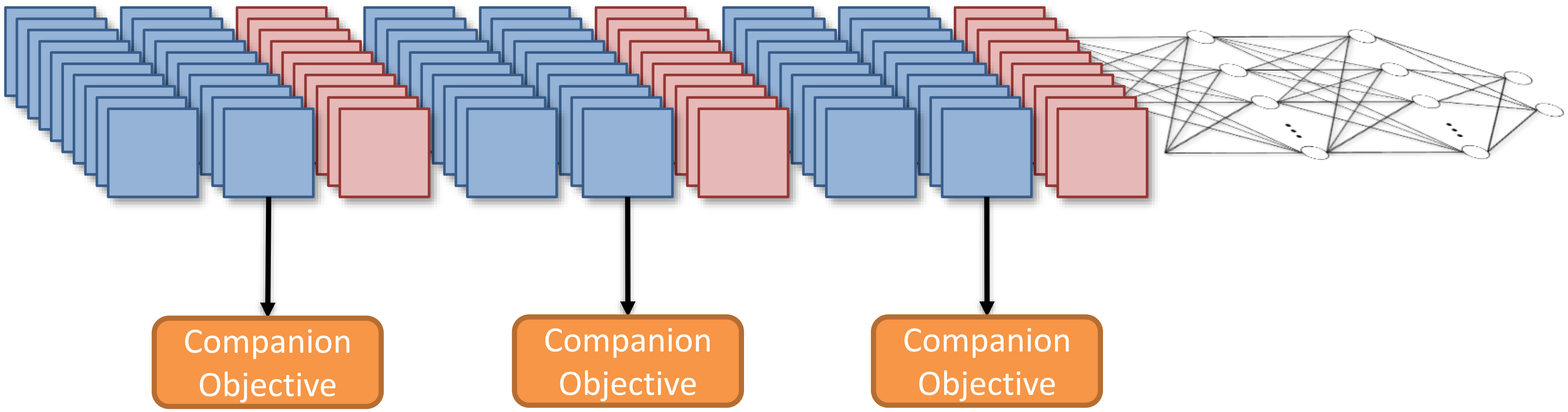
Slide credit: Pau Rodriguez and Adriana Romero. MCV M3 Architectures for image classification.

DSNs, FitNets, Highway Nets and ResNets

2015 - 2016

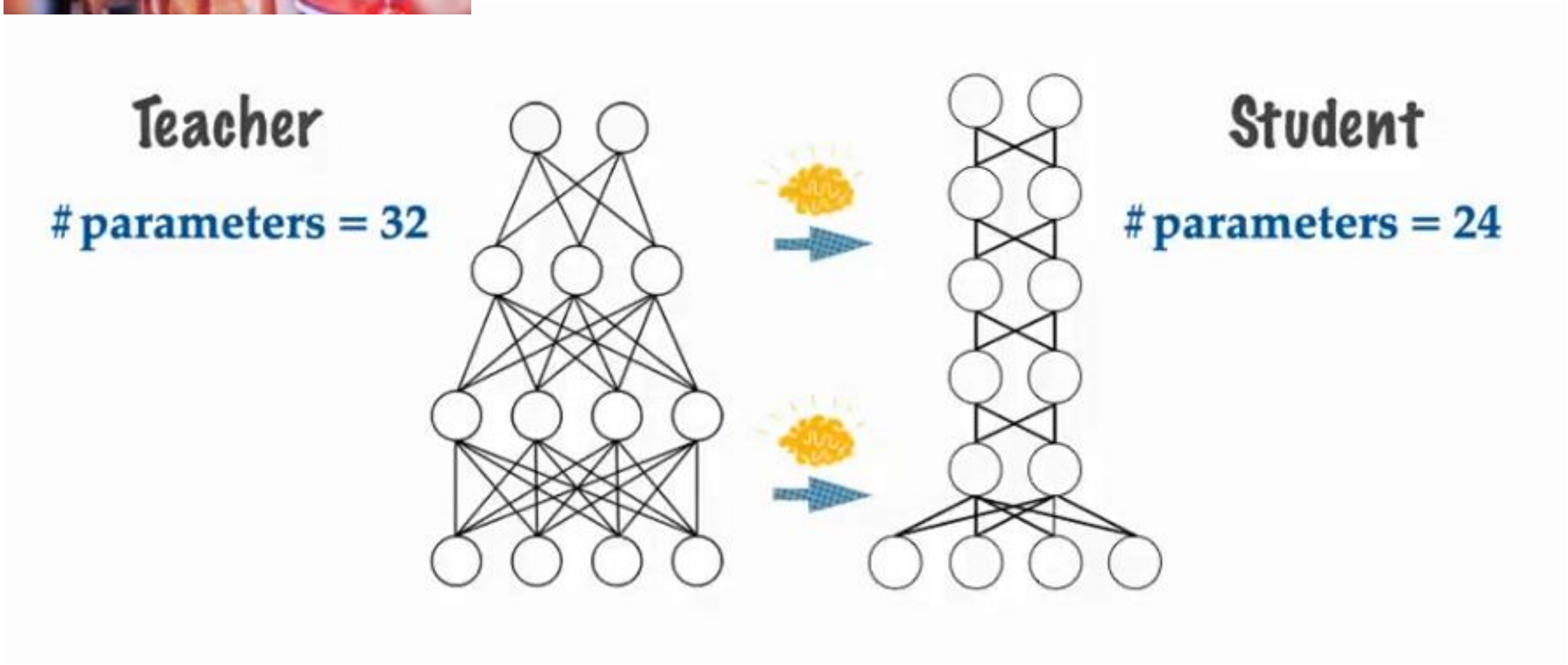
Deeply Supervised Networks

Adding intermediate supervision (discriminative loss)



Lee, Chen-Yu, et al. "Deeply-supervised nets." Artificial intelligence and statistics. PMLR, 2015

FitNets

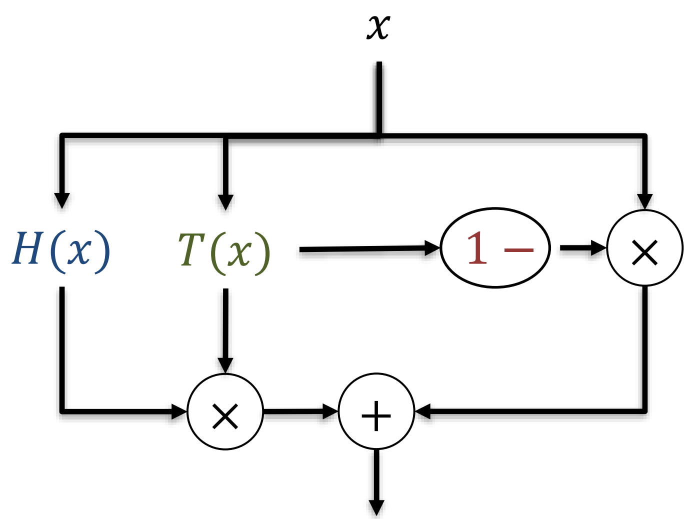


Knowledge transfer: trade width for depth to reduce the number of parameters

Key intuition: bigger is not better, just easier to optimise! Over-parameterisation helps optimisation.

A. Romero et al. "FitNets: Hints for Thin Deep Nets" ICLR 2015

Highway Networks



$$output = H(x)T(x) + x(1 - T(x))$$

x: Input

$H(x)$: Layer transform
(a block of operations)

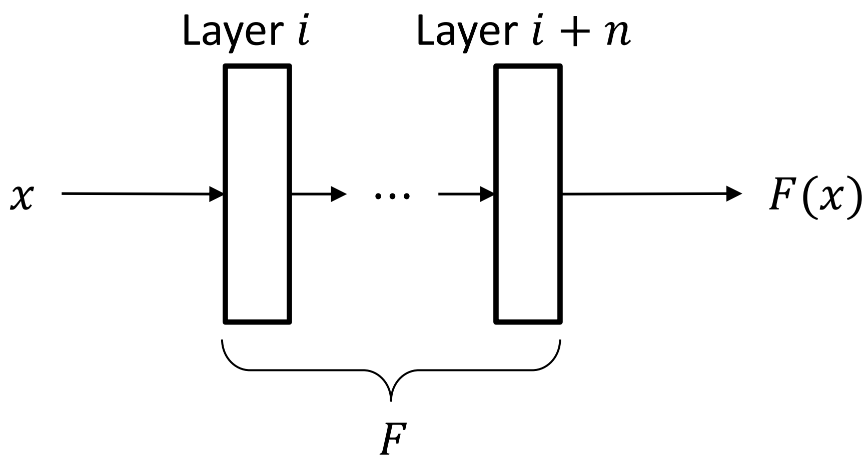
$T(x)$: Transform gate
(how much we transform the input)

$1 - T(x)$: Carry gate
(how much we preserve the input)

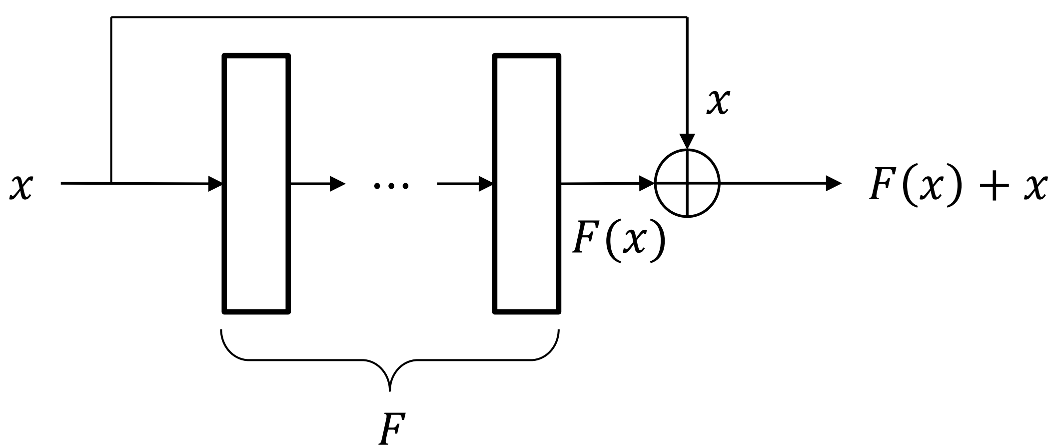
Key idea: provide an alternative, direct path for the gradient to flow back

Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. "Highway networks." ICML 2017

Residual connections



Traditional feedforward

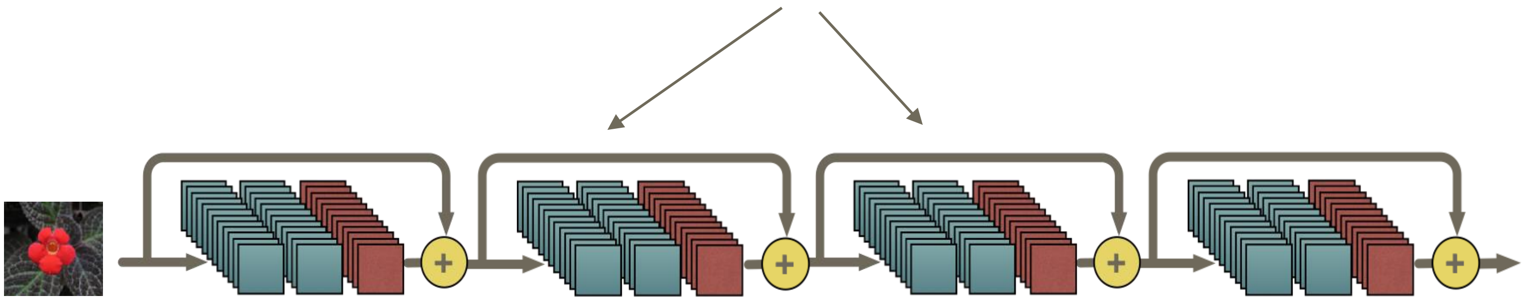


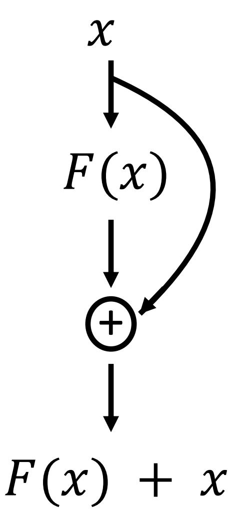
Feedforward with **residual** connection

One way to avoid vanishing gradient is to provide alternative routes for your gradients to flow through, a.k.a. residual connections

Residual Networks

Residual skip-connection



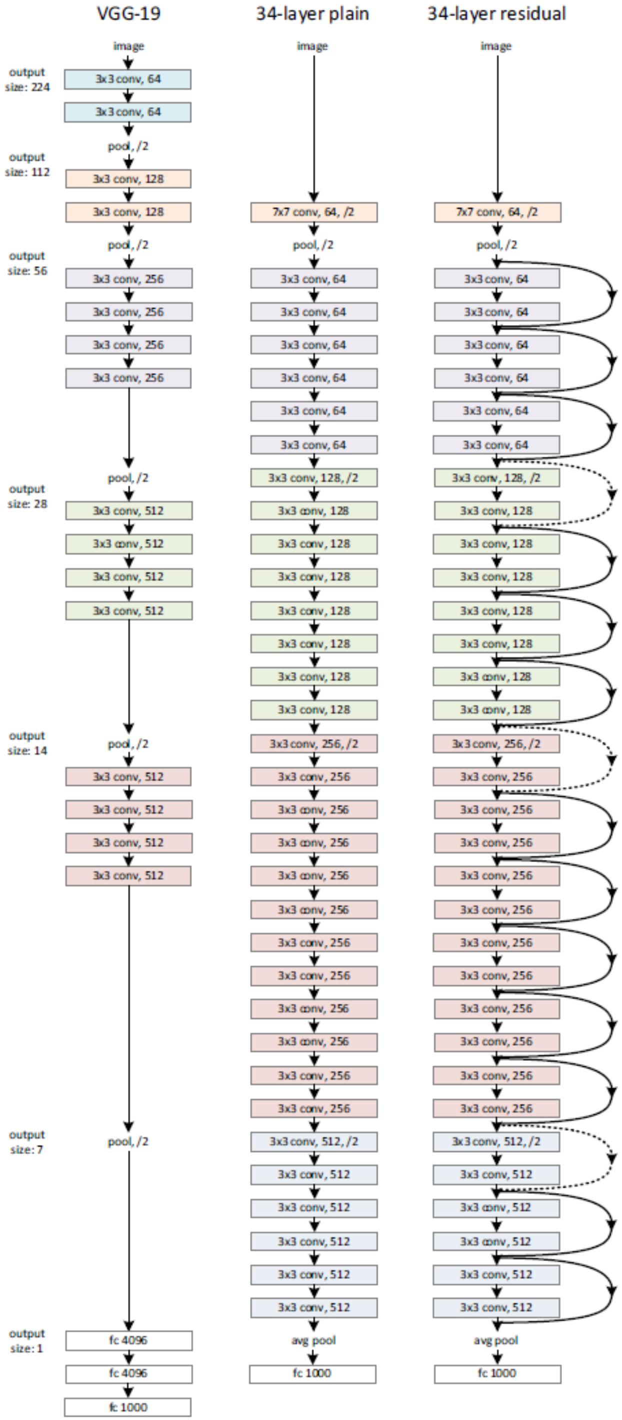


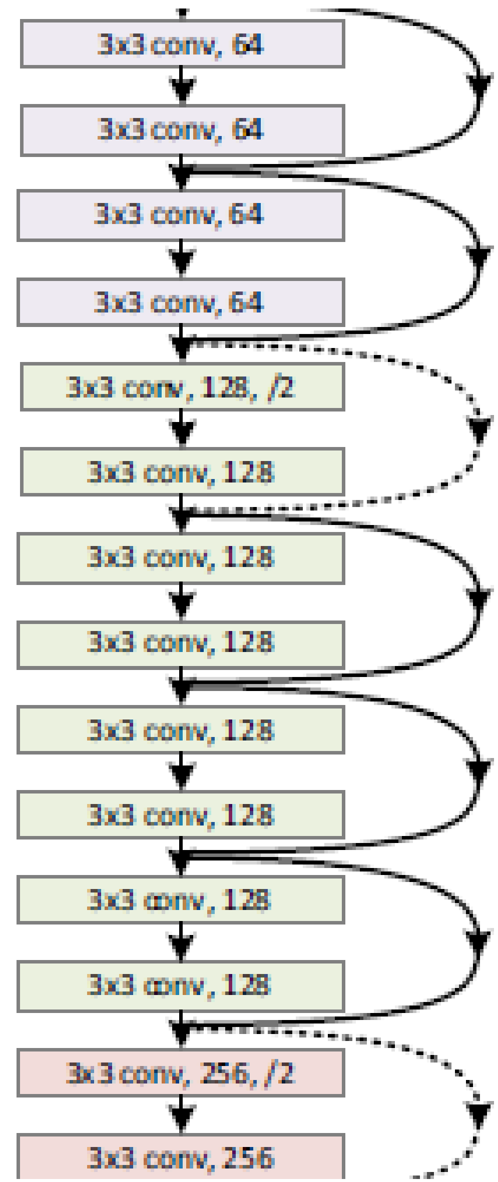
$$F(x) = \begin{pmatrix} \text{BN} \\ + \\ \text{ReLU} \\ + \\ \text{Conv.} \end{pmatrix} * n$$

- Compute a small change to the input to get a slightly altered representation
- Gradient flows easily to the bottom layers of the network
- **Ultra deep** architecture (152 layers)
- Adopts Batch Normalisation

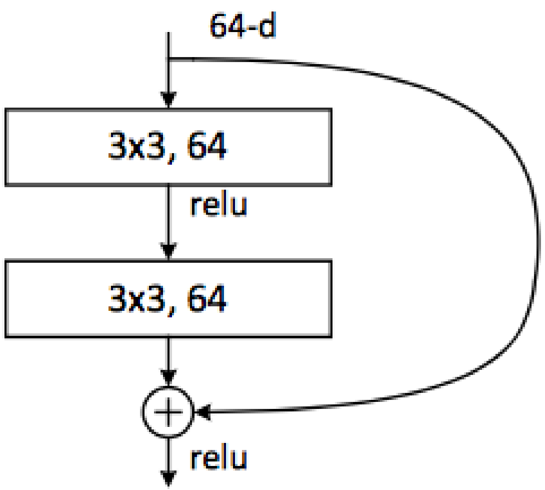
S. Ioffe, C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015
Kaiming, et al. "Deep residual learning for image recognition." CVPR. 2016

Residual Networks

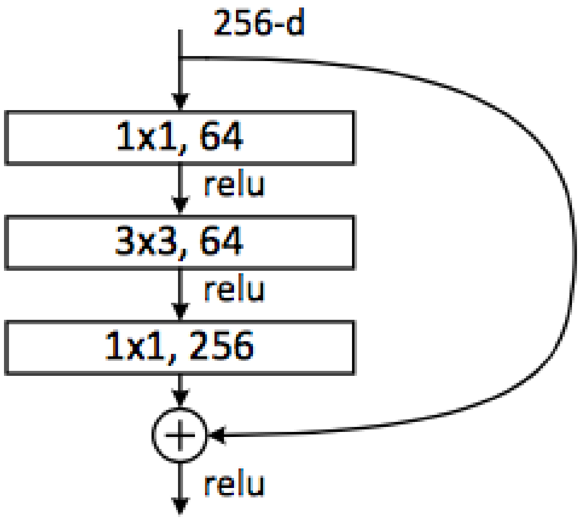




Residual Blocks



Basic Block



Bottleneck block

Residual Networks

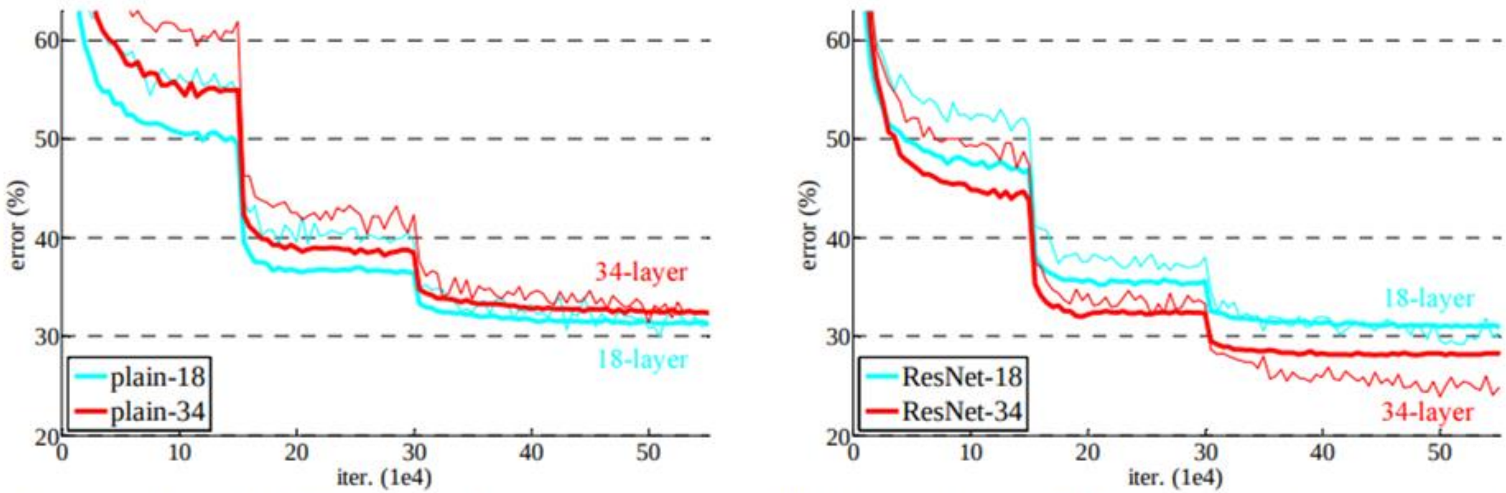
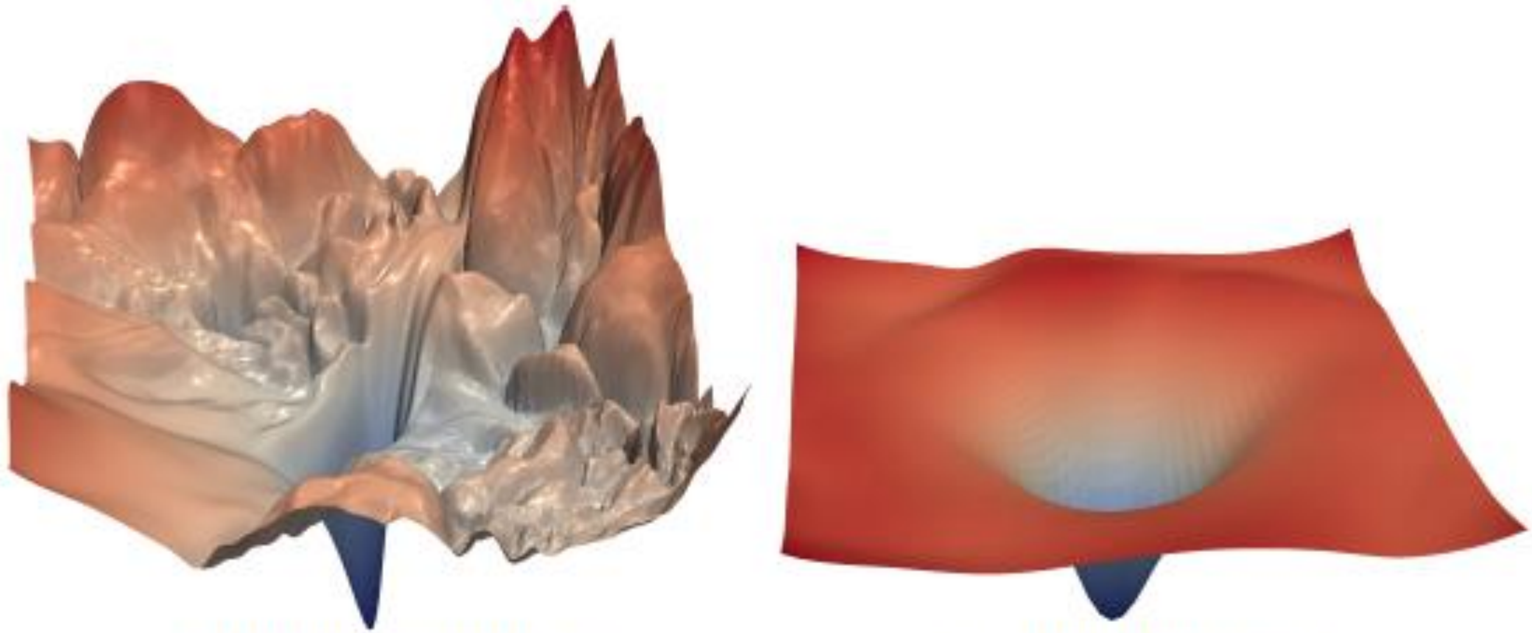


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Residual Networks



(a) without skip connections (b) with skip connections

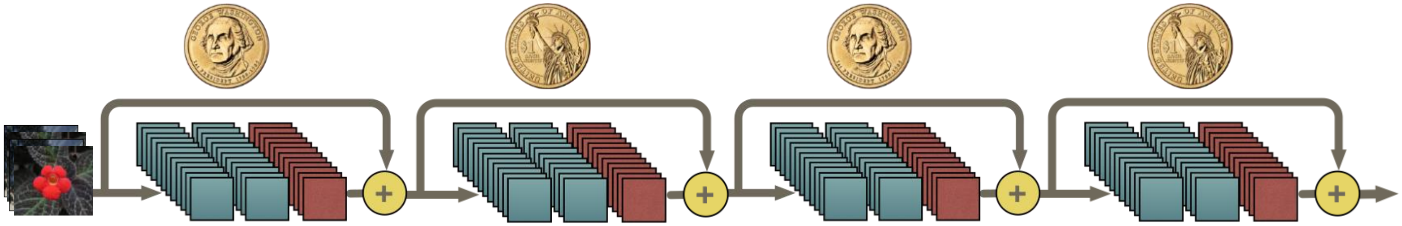
Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

Li, Hao, et al. "Visualizing the loss landscape of neural nets." arXiv preprint arXiv:1712.09913 (2017)

Stochastic depth, Dense Nets

2015 - 2016

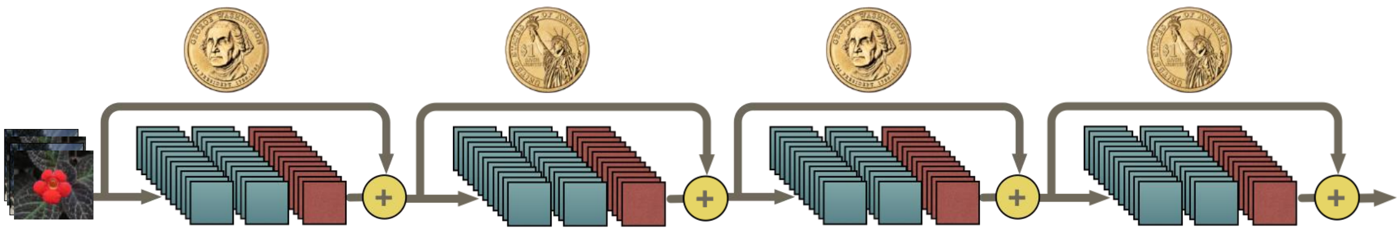
Stochastic Depth



Slide credit: Kilian Weinberger

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

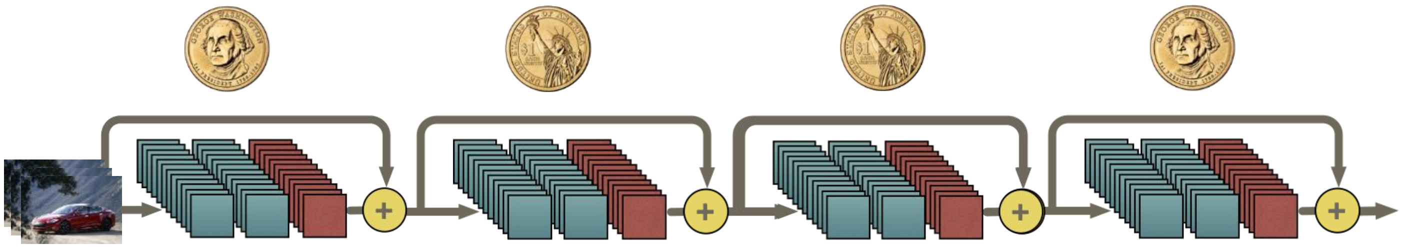
Stochastic Depth



Slide credit: Kilian Weinberger

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

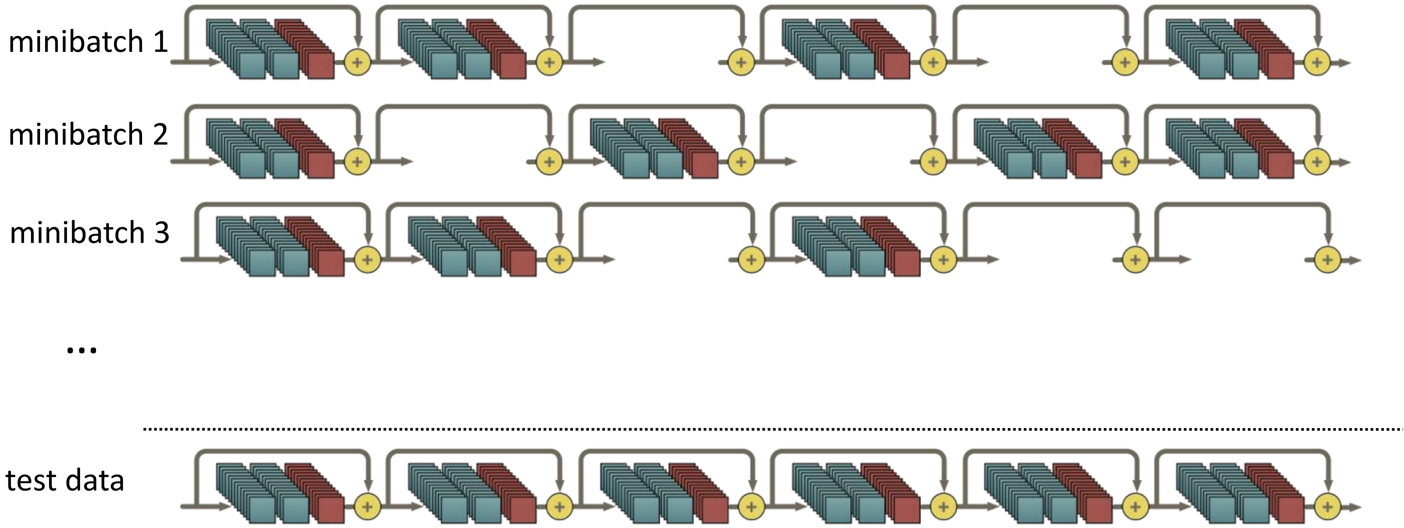
Stochastic Depth



Slide credit: Kilian Weinberger

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

Stochastic Depth

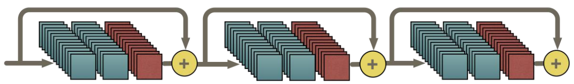


Slide credit: Kilian Weinberger

Huang et al, "[Deep Networks with Stochastic Depth]()", ECCV 2016

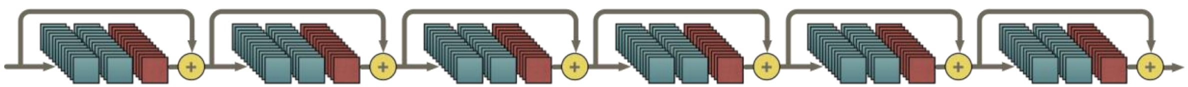
Stochastic Depth

training data



short during **training**

test data



deep during **testing**

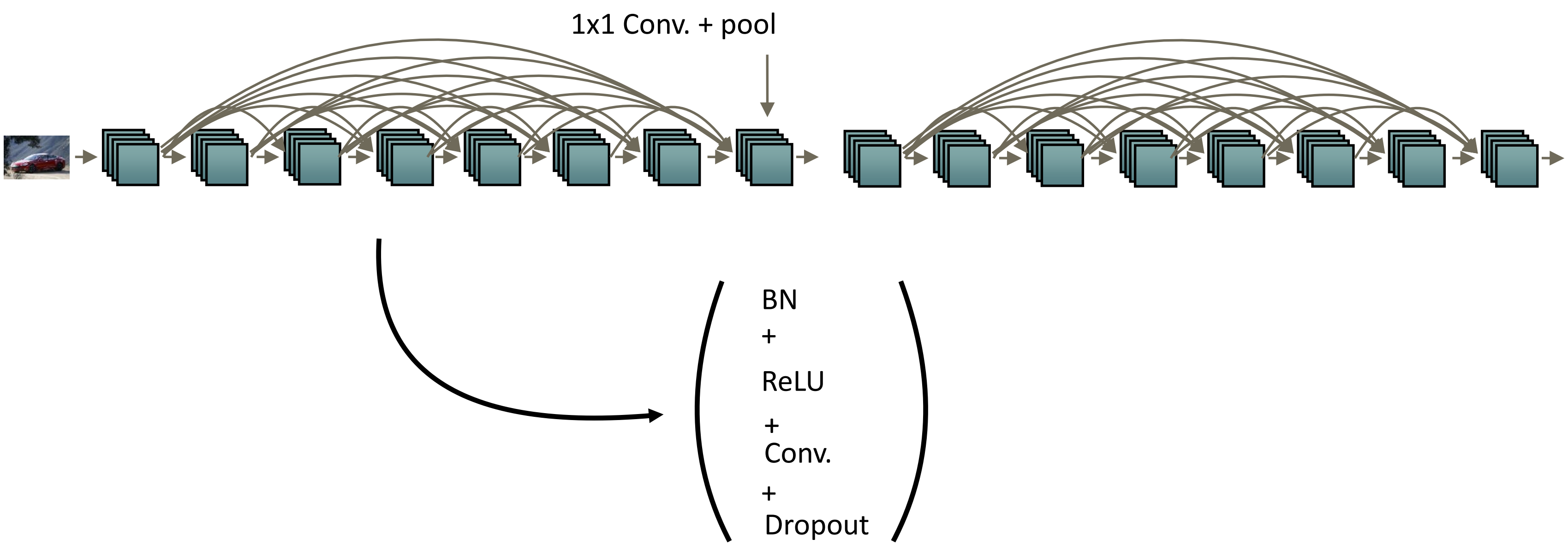
Key aspects:
- Implicit **ensemble** of 2^L models
- Improved **gradient flow**
- 25% speedup during training
- Lower error

Based on Kilian Weinberger slides

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

DenseNet

Connect every layer to every other layer of the same filter size (dense blocks).



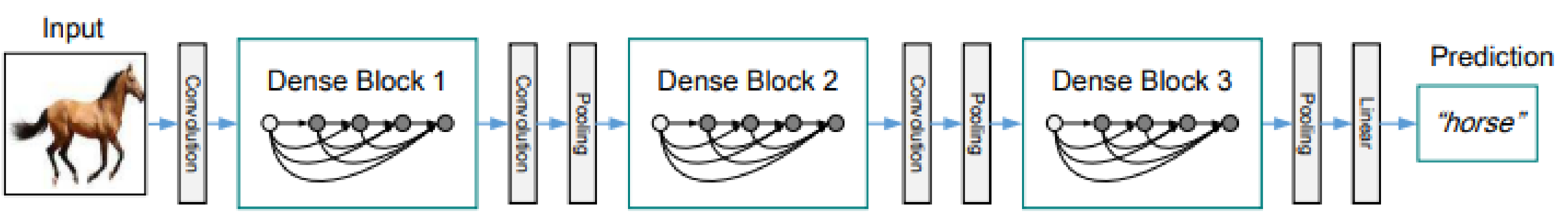
Slide credit: Kilian Weinberger

Huang, Gao, et al. "Densely connected convolutional networks." CVPR 2017

DenseNet

Example Dense Block (4 layers, growth rate 4):

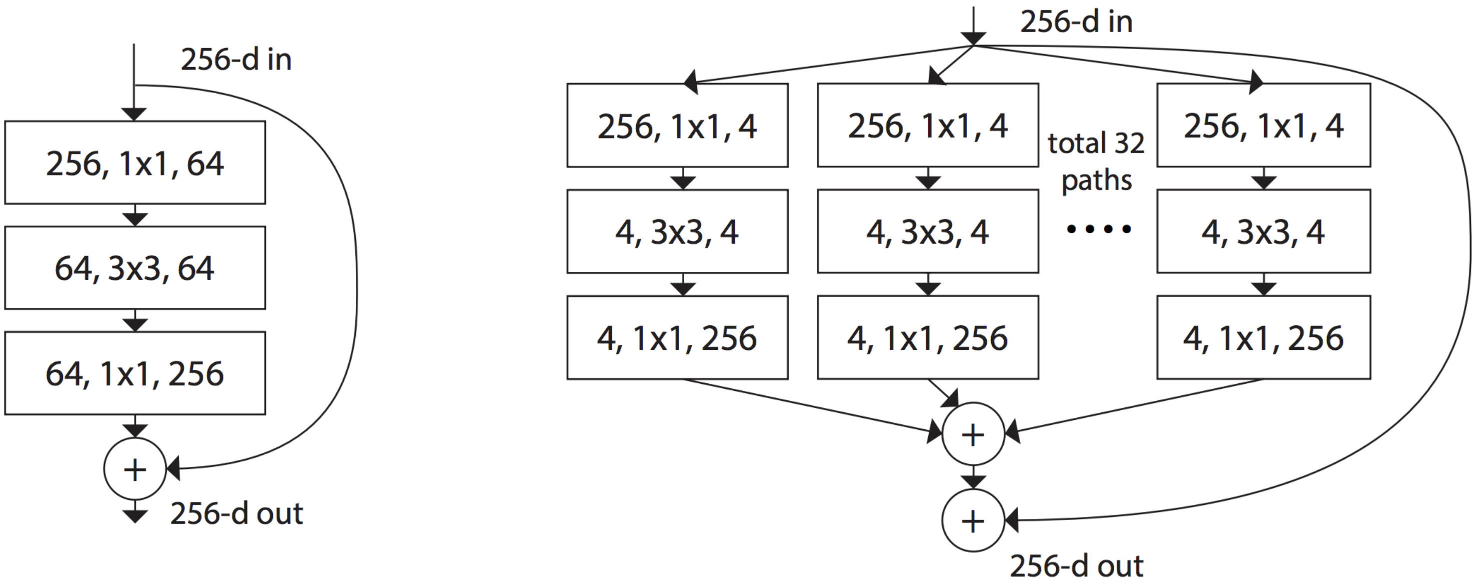
DenseNet can have very narrow layers (**small growth rate**). One explanation for this is that each layer has access to all the preceding feature-maps in its block. Each layer then adds a little bit of information (k feature-maps) to the global state of the network



Huang, Gao, et al. "Densely connected convolutional networks." CVPR 2017

ResNeXt

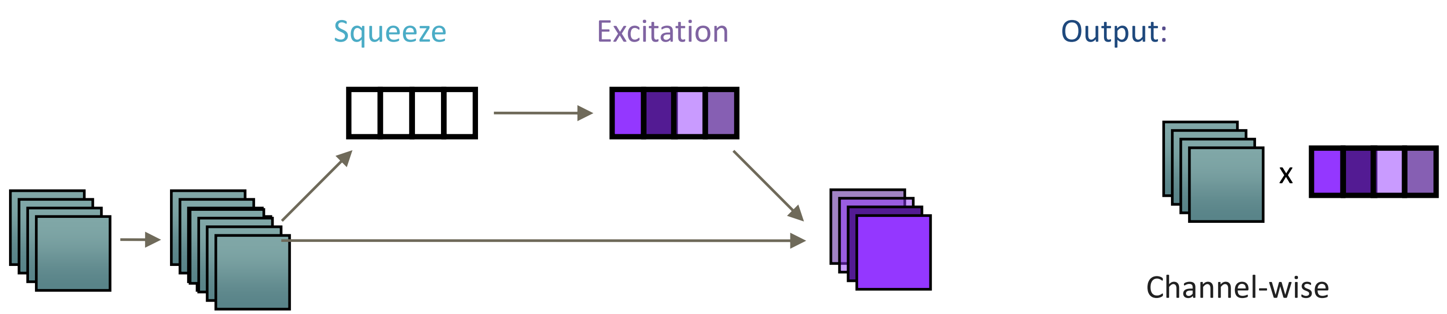
ResNeXt introduces the concept of “cardinality”. Apart from depth (how many layers) and width (how many filters).



Key concept: create a lot of paths with a very small number of channels using 1×1 convolutions

Xie, et al. "Aggregated Residual Transformations for Deep Neural Networks." CVPR 2017

Squeeze-&-Excitation Networks

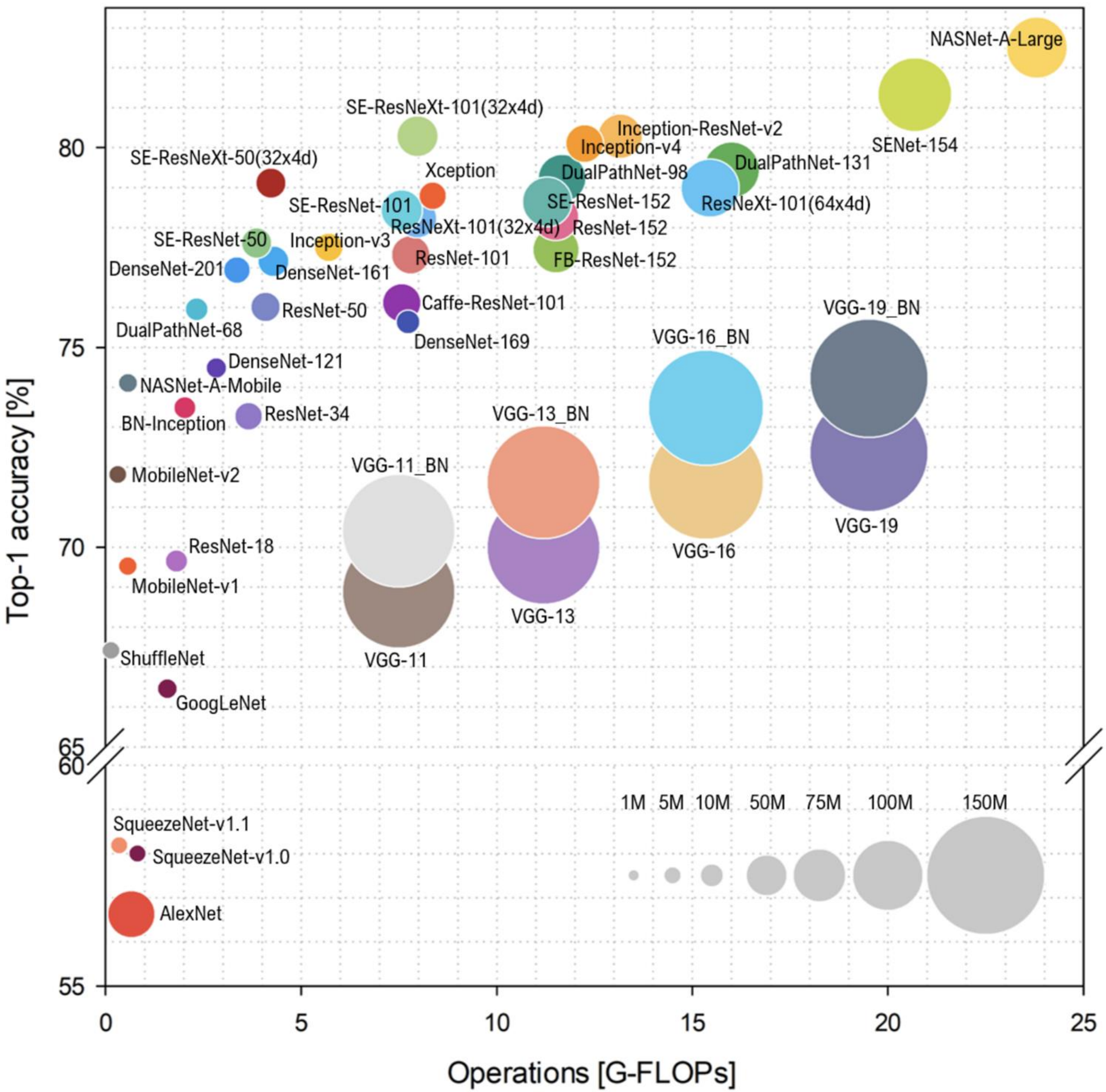


Squeeze: average pooling - captures channel-wise global info

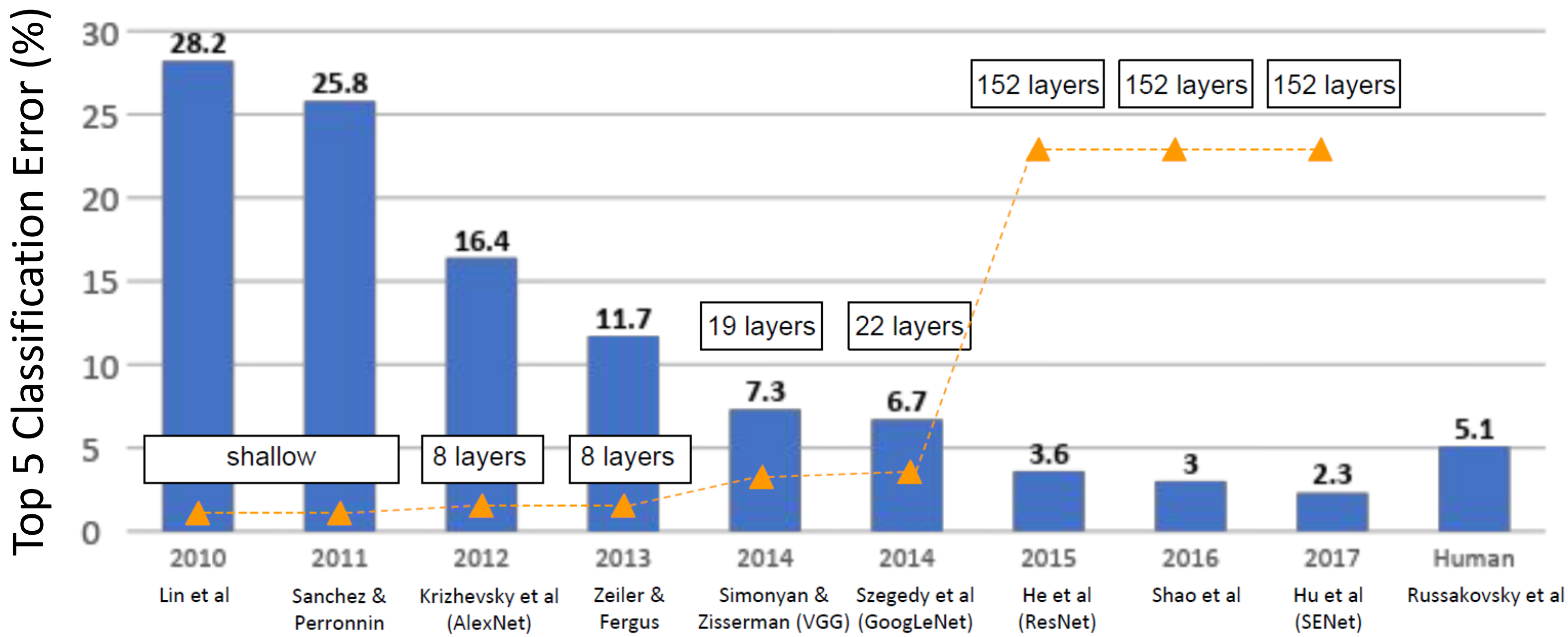
Excitation: non-linear channel interaction + gating (sigmoid)

Output: scales activations, leveraging global information at all levels

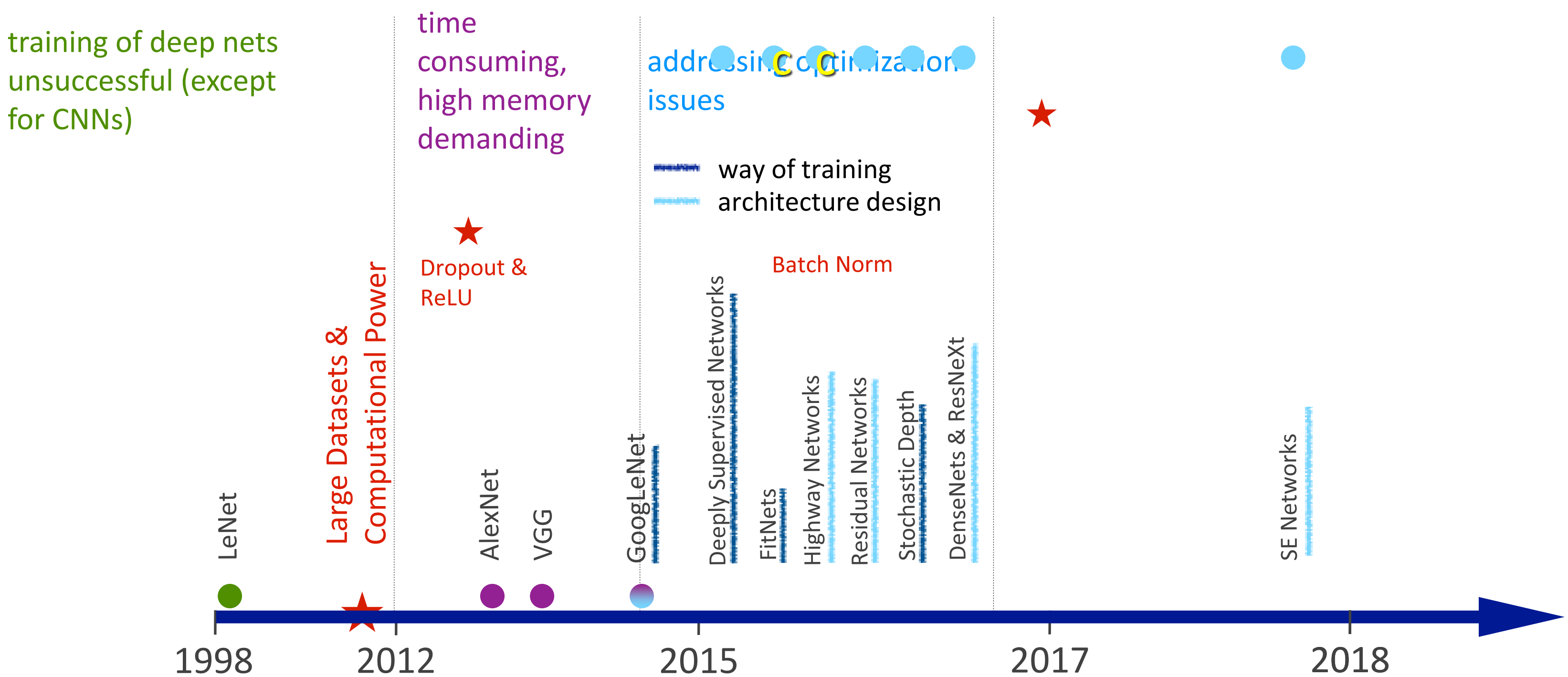
Hu, et al. "Squeeze-and-Excitation Network." CVPR 2018


NASNet-A-Large
SENet-154
SE-ResNeXt-101(32x4d)
Inception-ResNet-v2
Inception-v4
DualPathNet-131
DualPathNet-98
ResNeXt-101(64x4d)
SE-ResNet-152
ResNet-152
FB-ResNet-152
SE-ResNeXt-50(32x4d)
Xception
SE-ResNet-101
ResNeXt-101(32x4d)
ResNet-101
SE-ResNet-50
Inception-v3
DenseNet-201
DenseNet-161
Caffe-ResNet-101
ResNet-50
DualPathNet-68
DenseNet-169
DenseNet-121
NASNet-A-Mobile
BN-Inception
ResNet-34
MobileNet-v2
ResNet-18
MobileNet-v1
ShuffleNet
GoogLeNet
SqueezeNet-v1.1
SqueezeNet-v1.0
AlexNet
VGG-11_BN
VGG-11
VGG-13_BN
VGG-13
VGG-16_BN
VGG-16
VGG-19_BN
VGG-19
1M 5M 10M 50M 75M 100M 150M
Top-1 accuracy [%]
Operations [G-FLOPs]
55 60 65 70 75 80
0 5 10 15 20 25

ImageNet Classification



Wrap Up

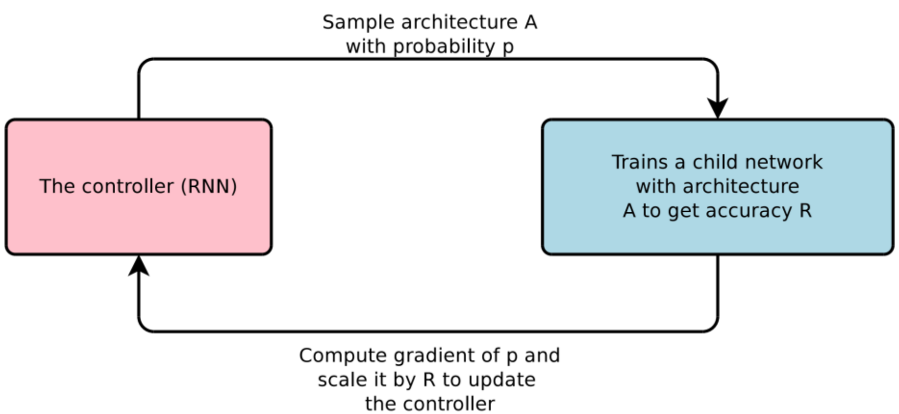


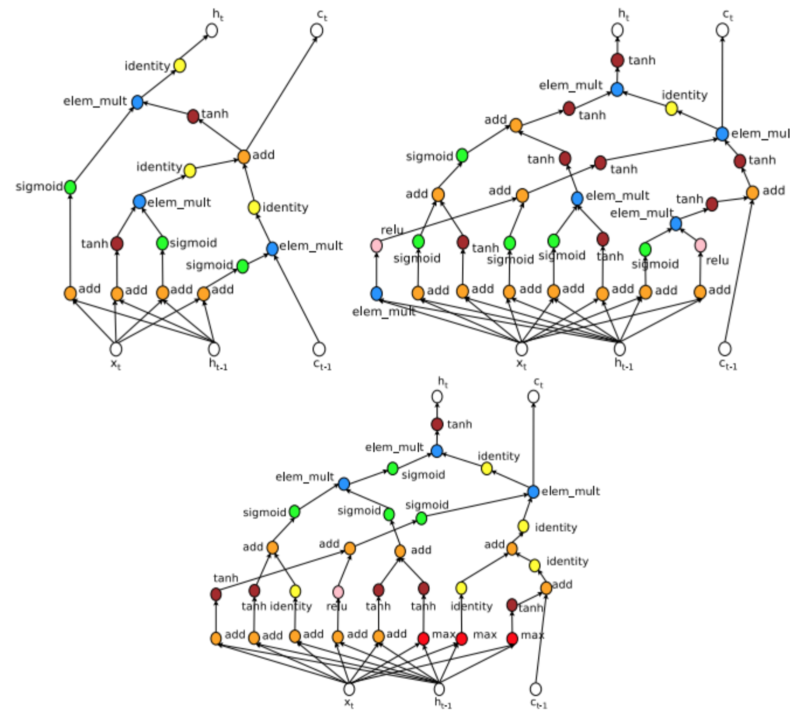
Slide credit: Pau Rodriguez and Adriana Romero. MCV M3 Architectures for image classification.

Neural Architecture Search

2018 - 2019

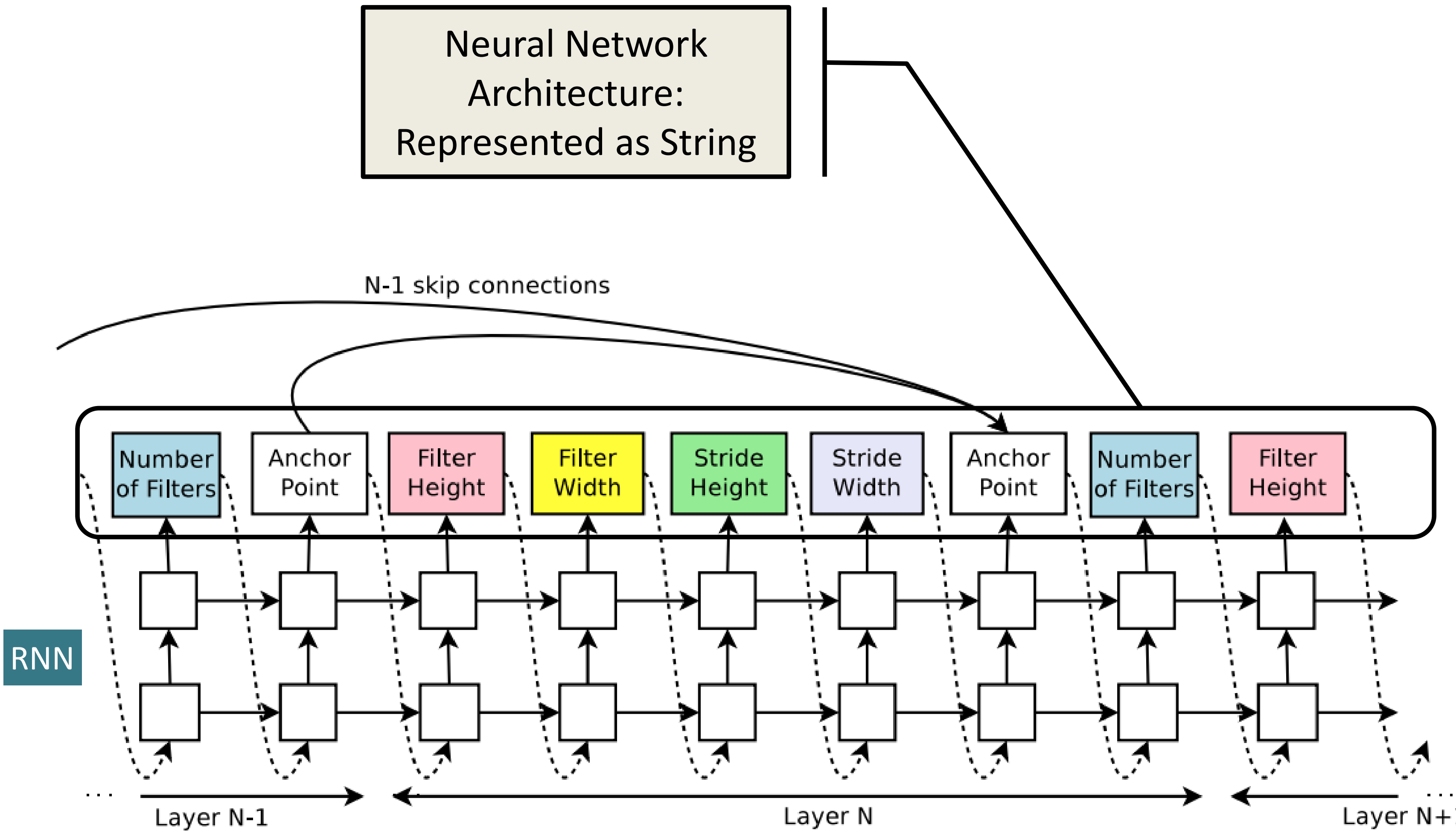
Neural Architecture Search (NAS)





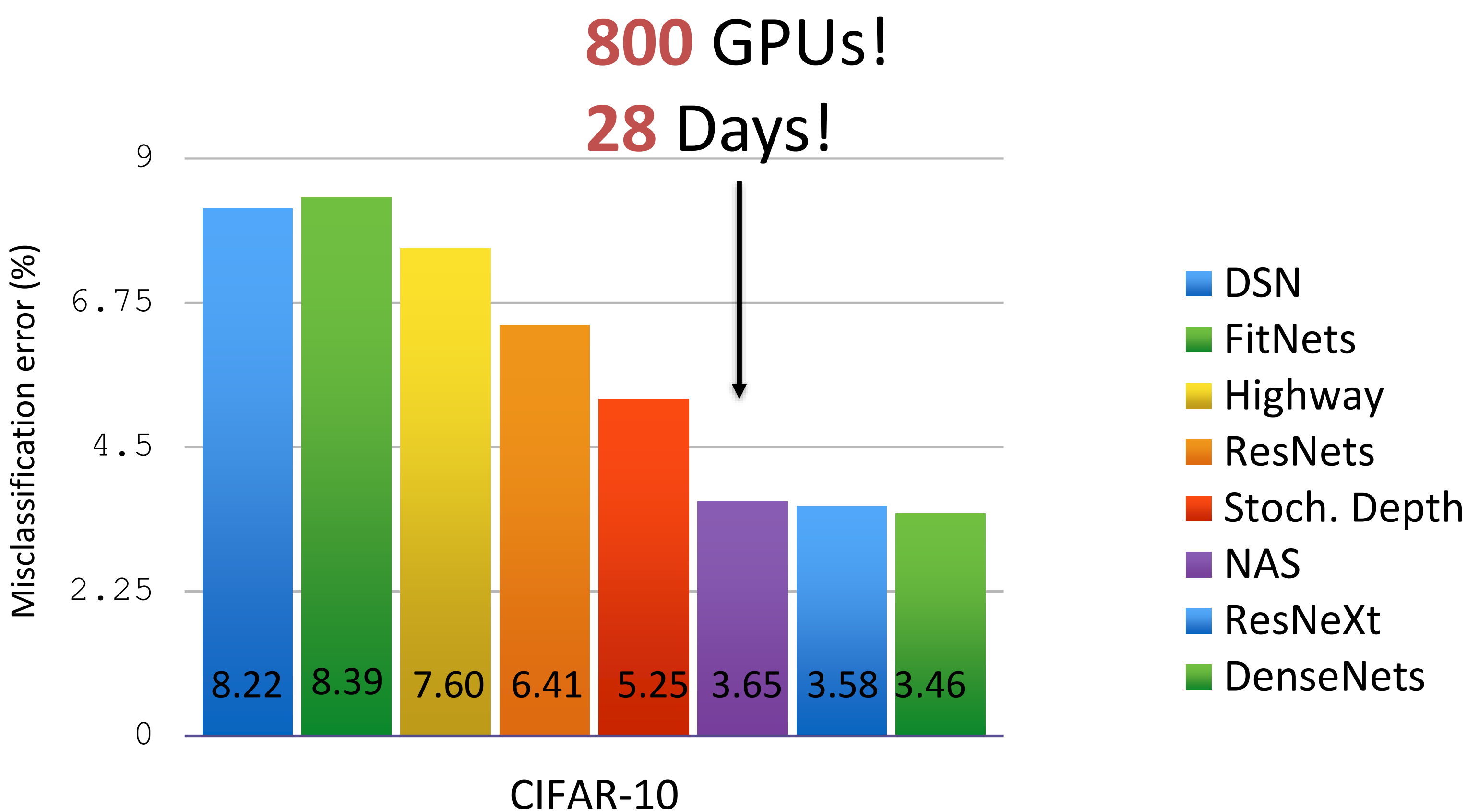
Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. ICLR 2017

Neural Architecture Search (NAS)



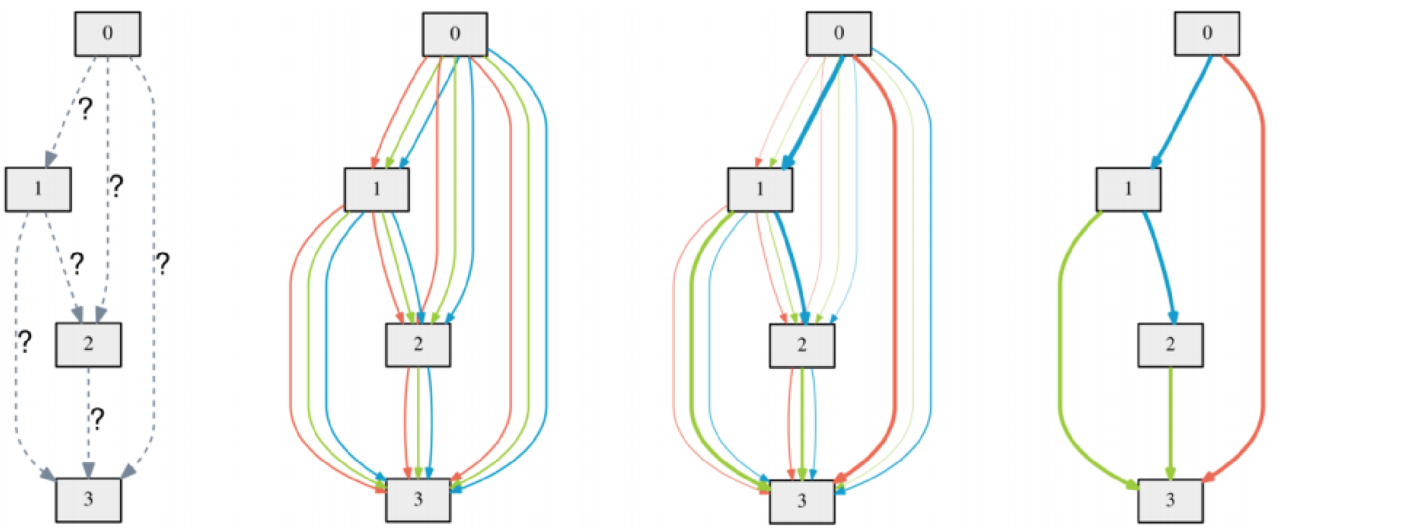
Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. ICLR 2017

CIFAR-10 Results



DARTS: Differentiable Architecture Search

Key concept: Start with all operations connected. Then slowly prune connections that are not being used. Completely remove them at the end for test.

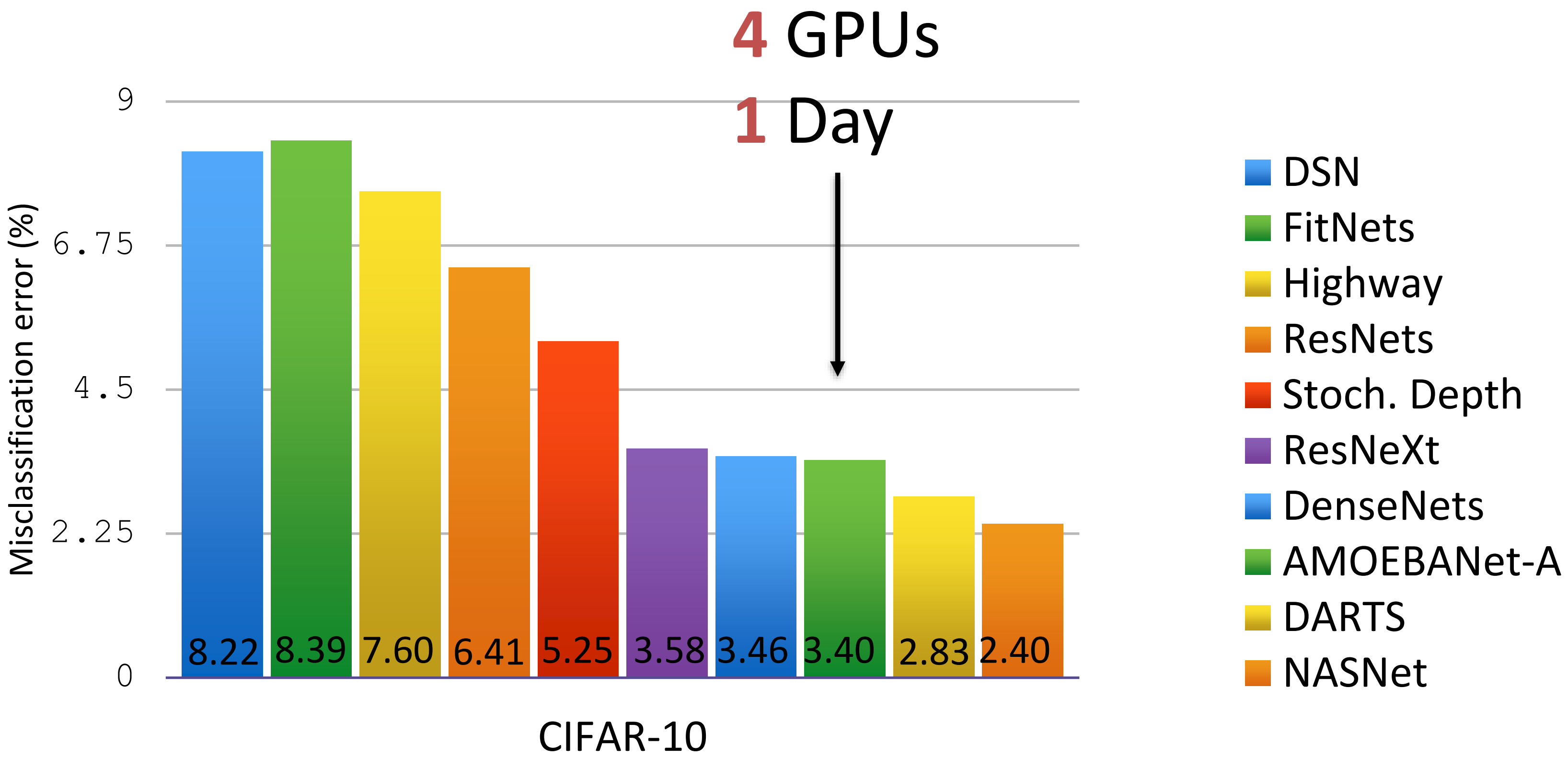


1 **2** **3** **4**

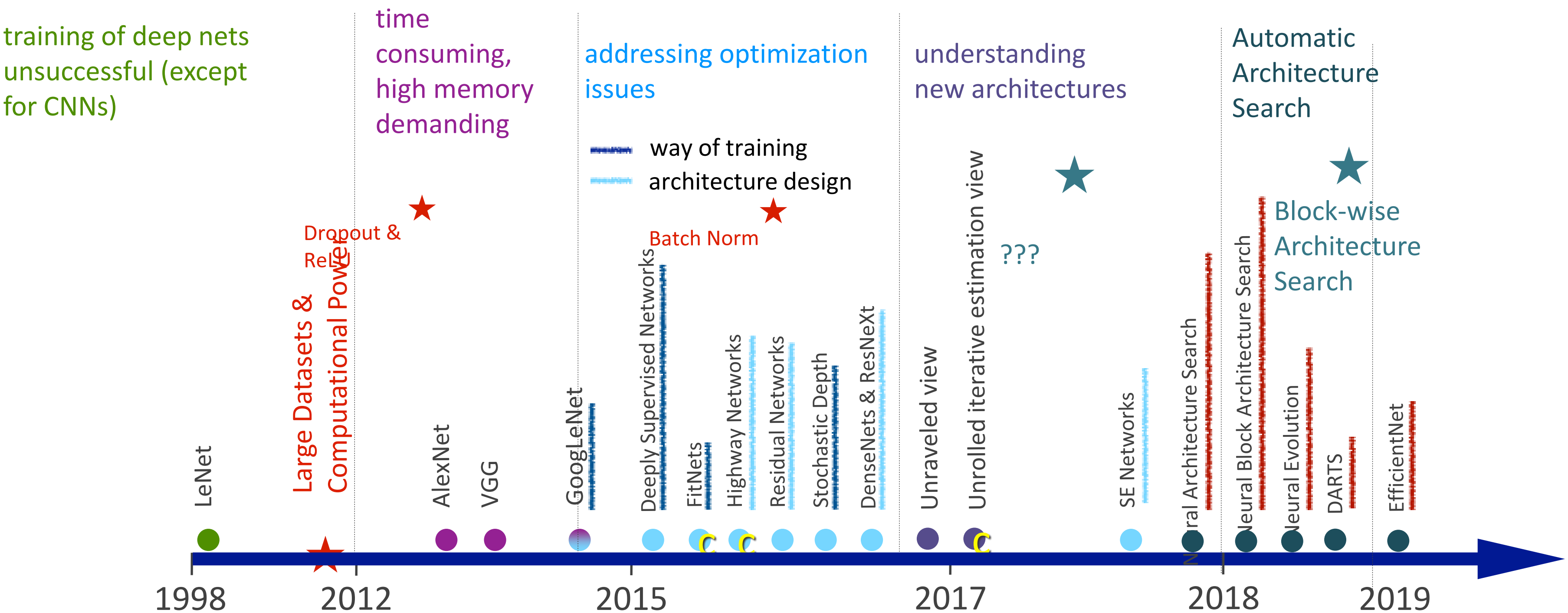
Relaxed → Constrained

Chu, X.,et al. "Fair darts: Eliminating unfair advantages in differentiable architecture search". ECCV 2020

CIFAR-10 Results

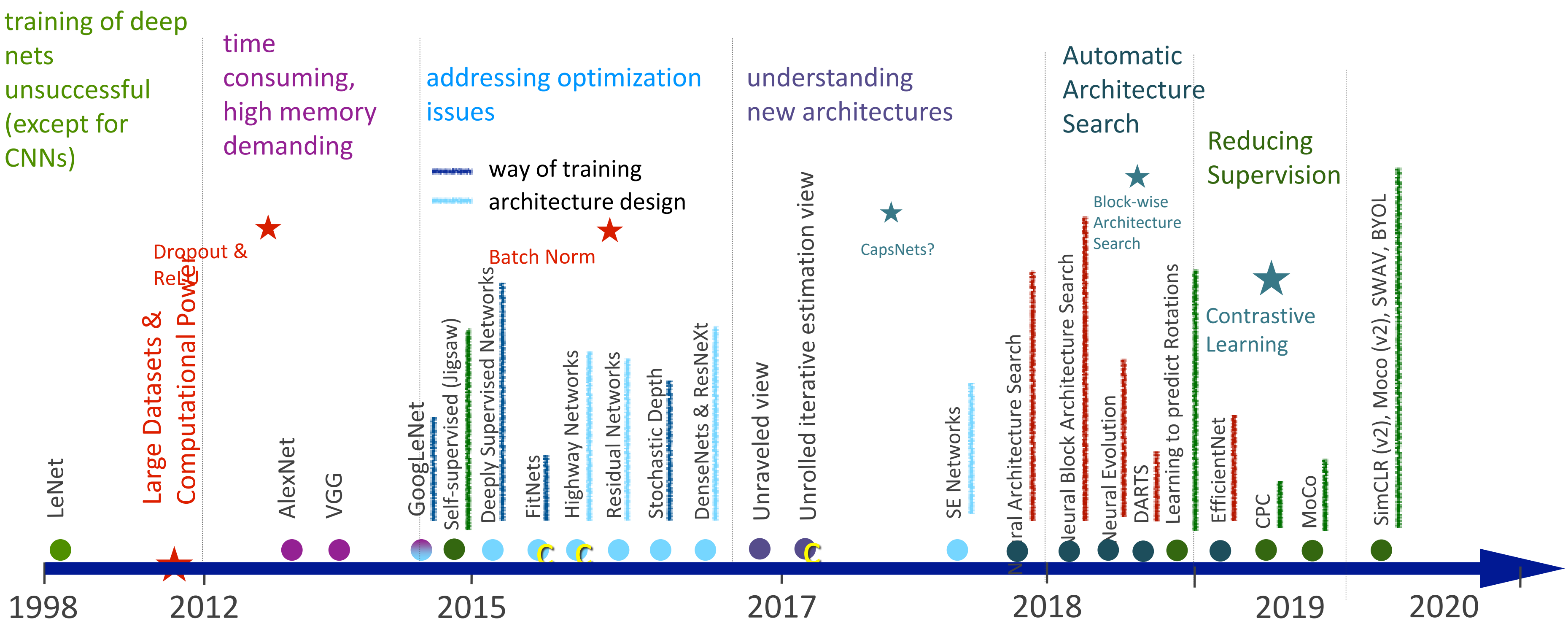


Wrap Up



Slide credit: Pau Rodriguez and Adriana Romero. MCV M3 Architectures for image classification.

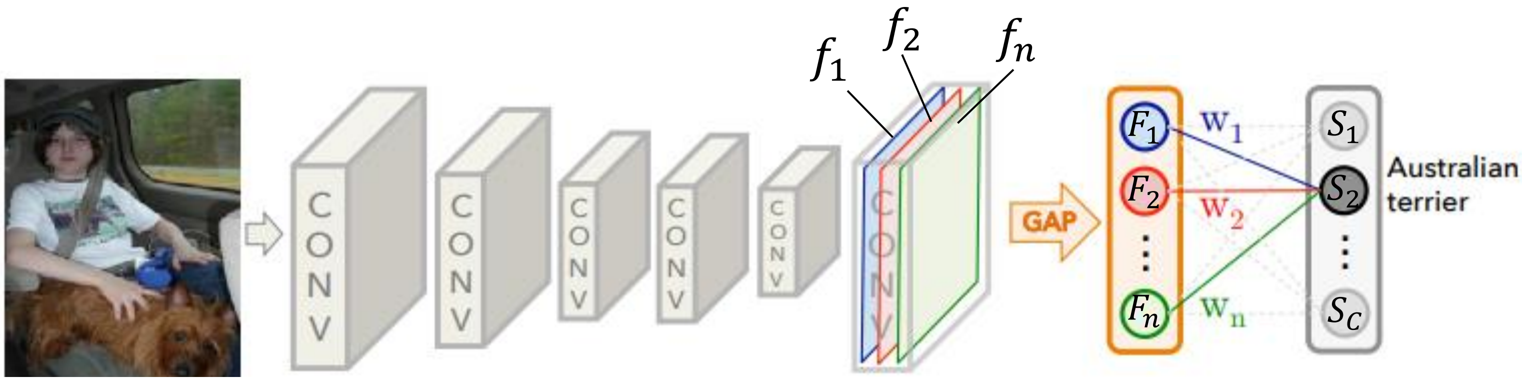
Wrap Up



Slide credit: Pau Rodriguez and Adriana Romero. MCV M3 Architectures for image classification.

GETTING INSIGHTS

Class Activation Mapping

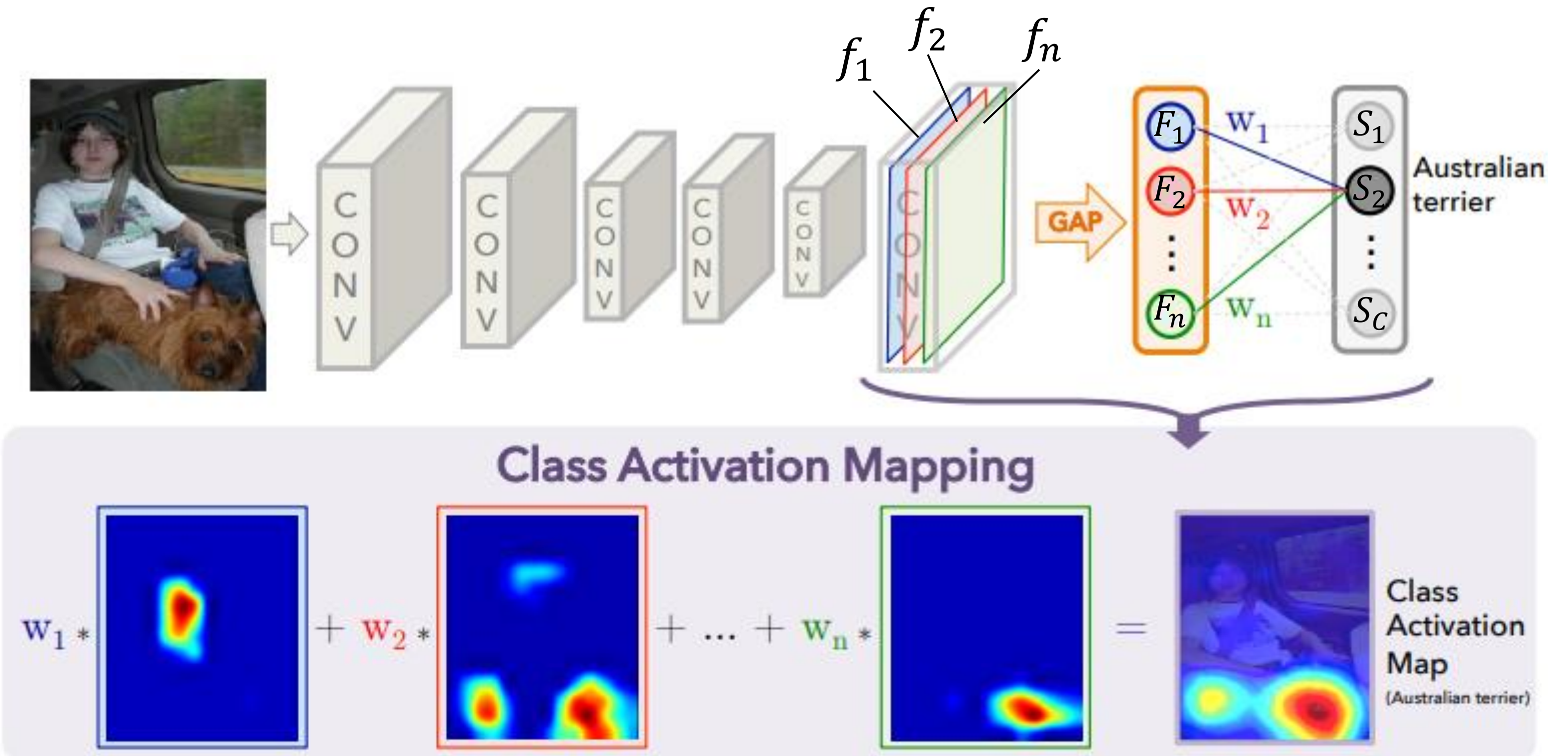


$$F_k = \sum_{x,y} f_k(x,y)$$

$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y)$$

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. CVPR'16

Class Activation Mapping



$$M_c(x,y) = \sum_k w_k^c \, f_k(x,y)$$

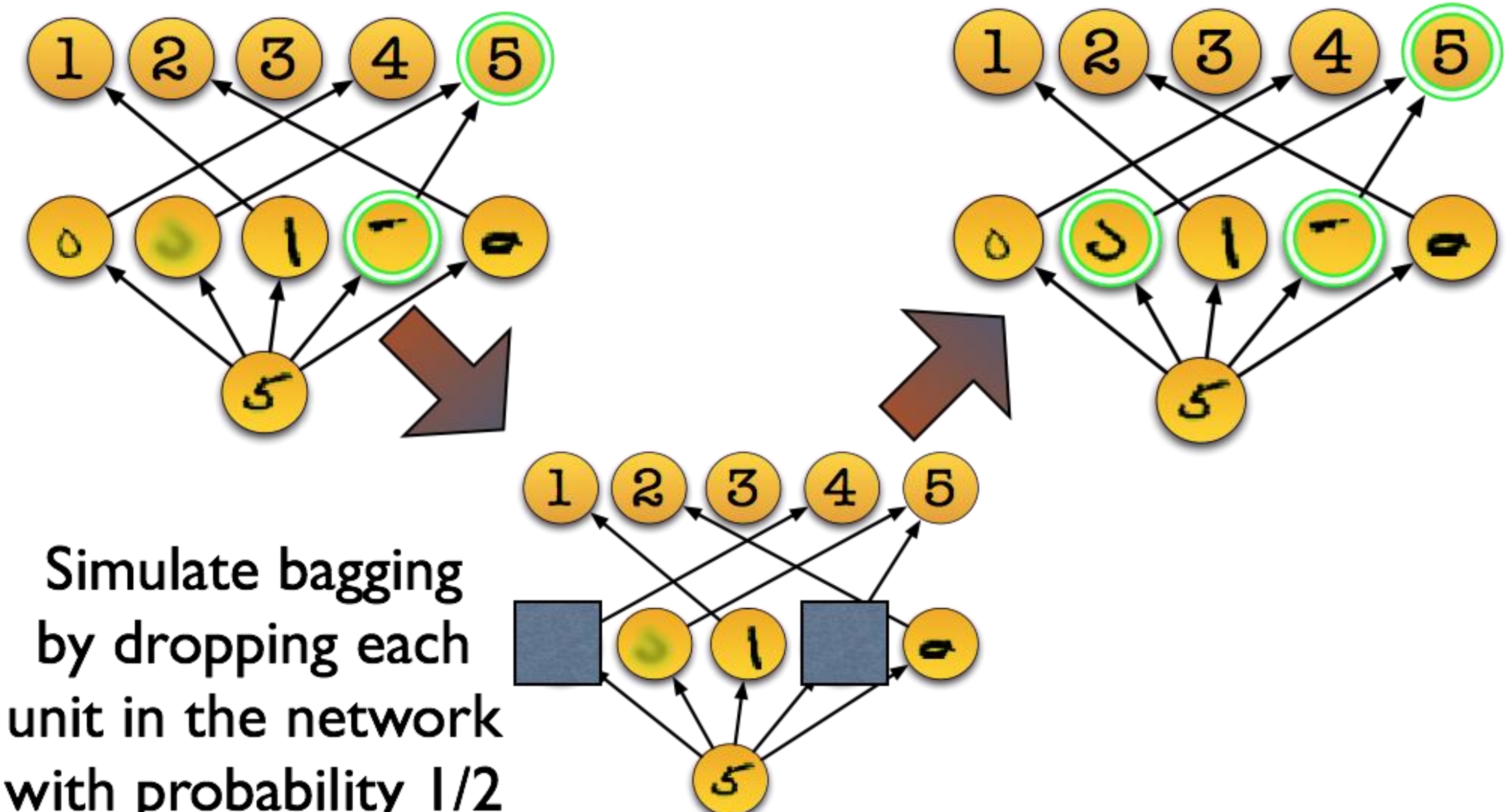
B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. CVPR'16

Class Activation Mapping

Seen in previous classes

RECAP OF IMPORTANT IDEAS

Dropout



Simulate bagging by dropping each unit in the network with probability 1/2

Slide credits: I. Goodfellow. Introduction to Deep Learning, AERFI Deep Learning Autumn School, Nov 5, 2015, Valencia.

Dropout in PyTorch

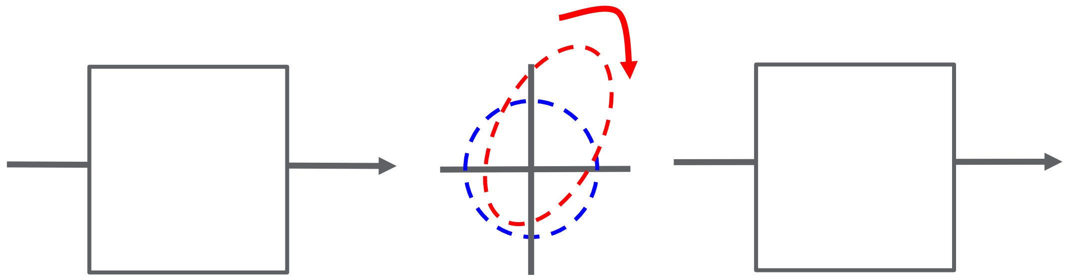
```
torch.nn.Dropout(p=0.5, inplace=False)
```

https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html

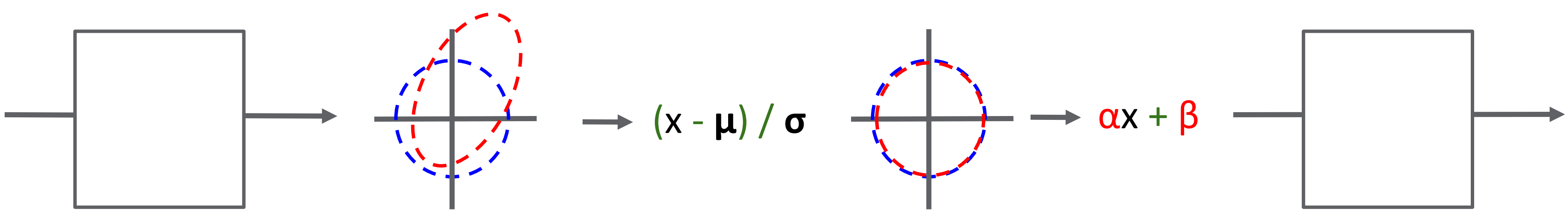
https://losslandscape.com/ https://youtu.be/2PqTW_p1fls

Batch Normalization

Before:



After:



Slide credits: I. Goodfellow. Deep Learning for Computer Vision, AERFI Deep Learning Autumn School, Nov 5, 2015, Valencia. Adapted from slides by Vincent Vanhoucke.

Batch Normalization in PyTorch

```
torch.nn.BatchNorm1d(num_features, eps=1e-05, momentum=0.1,
                     affine=True, track_running_stats=True)

torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1,
                   affine=True, track_running_stats=True)
```

https://pytorch.org/docs/stable/nn.html#normalization-layers

Data Augmentation

Best way to make a machine learning model generalize better is to train it on more data -> **Create new data!**

Image source: [Building powerful image classification models using very little data]

Data Augmentation in PyTorch

```
From torchvision import transforms

rgb_mean = (0.4914, 0.4822, 0.4465)
rgb_std = (0.2023, 0.1994, 0.2010)

transform_train = transforms.Compose([
    transforms.RandomCrop(32, padding=4),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(rgb_mean, rgb_std),
])
```

https://pytorch.org/vision/stable/transforms.html

Transfer Learning (supervised pre-training)

- Initialize the weights of your model using the optimal weights learnt on a similar task!
- Fine-tuning can provide a reasonably good model even when we have very few training data.

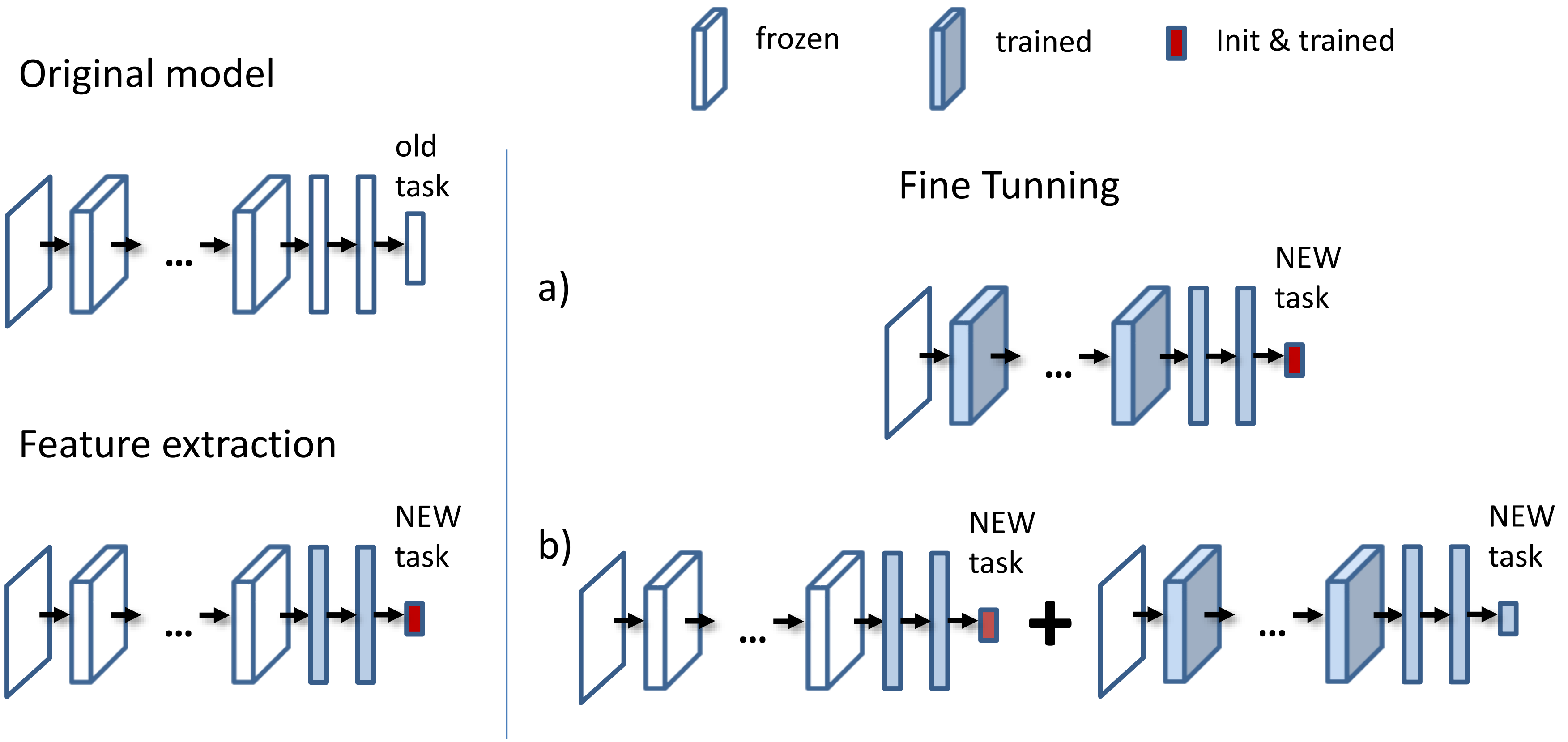
If you know how to recognize...

Transfer Learning

You will be able to recognize...

Image source: Yannis Ghazouani, 2016.

Transfer Learning (supervised pre-training)



Sharif Razavian, Ali, et al. "CNN features off-the-shelf: an astounding baseline for recognition." Proceedings of the IEEE CVPRW. 2014.

Transfer Learning

In practice, very few people train a CNN from scratch (with random initialization). Don’t have sufficient data and/or resources.

It is common to use pretrained CNNs and use them either as an initialization or a fixed feature extractor for the new task.

When and how to transfer learning?

- New dataset is small and similar to original dataset: feature ext.
- New dataset is large and similar to the original dataset: feat ext+fine tunn
- New dataset is small but very different from the original dataset: feat ext
- New dataset is large and very different from the original dataset: pre trained

Practical advice

- Constraints from pretrained models.
- Learning rates.

https://cs231n.github.io/transfer-learning/