

# **Architectures for Image Classification**

Master in Computer Vision Barcelona

Pau Rodríguez, January 2022

pau.rodriguez@servicenow.com

*Original slides (2017): Adriana Romero (adrianars@fb.com)*

# **Outline**

# Outline

- Problem, metrics & datasets

# Outline

- Problem, metrics & datasets
- CNN basics

# Outline

- Problem, metrics & datasets
- CNN basics
- 2012-2014: AlexNet, VGG & GoogLeNet

# Outline

- Problem, metrics & datasets
- CNN basics
- 2012-2014: AlexNet, VGG & GoogLeNet
- 2014-2015: DSN, FitNets, Highway Networks & ResNets

# Outline

- Problem, metrics & datasets
- CNN basics
- 2012-2014: AlexNet, VGG & GoogLeNet
- 2014-2015: DSN, FitNets, Highway Networks & ResNets
- 2016-2017: Stochastic Depth, DenseNets, ResNeXt, SE Networks

# Outline

- Problem, metrics & datasets
- CNN basics
- 2012-2014: AlexNet, VGG & GoogLeNet
- 2014-2015: DSN, FitNets, Highway Networks & ResNets
- 2016-2017: Stochastic Depth, DenseNets, ResNeXt, SE Networks
- From representation view to iterative estimation

# Outline

- Problem, metrics & datasets
- CNN basics
- 2012-2014: AlexNet, VGG & GoogLeNet
- 2014-2015: DSN, FitNets, Highway Networks & ResNets
- 2016-2017: Stochastic Depth, DenseNets, ResNeXt, SE Networks
- From representation view to iterative estimation
- 2017-2018: Neural Architecture search, CapsNets

# Outline

- Problem, metrics & datasets
- CNN basics
- 2012-2014: AlexNet, VGG & GoogLeNet
- 2014-2015: DSN, FitNets, Highway Networks & ResNets
- 2016-2017: Stochastic Depth, DenseNets, ResNeXt, SE Networks
- From representation view to iterative estimation
- 2017-2018: Neural Architecture search, CapsNets
- 2018-2020: Self-supervised learning and Transformers

# Outline

- Problem, metrics & datasets
- CNN basics
- 2012-2014: AlexNet, VGG & GoogLeNet
- 2014-2015: DSN, FitNets, Highway Networks & ResNets
- 2016-2017: Stochastic Depth, DenseNets, ResNeXt, SE Networks
- From representation view to iterative estimation
- 2017-2018: Neural Architecture search, CapsNets
- 2018-2020: Self-supervised learning and Transformers
- 2020-2022: Understanding Transformers and Improving ResNets

# Problem, metrics & datasets

# Image Classification

Which class does the image belong to?



# Image Classification

Which class does the image belong to?



(Zhou et al., 2015)

# Image Classification

Which class does the image belong to?



# Image Classification

Which class does the image belong to?



# Image Classification

Which class does the image belong to?



coast



coast



coast



coast



mountain

# Image Classification

which class does the image belong to?



Evaluation Metrics

Accuracy = \_\_\_\_\_



# Image Classification

which class does the image belong to?



Evaluation Metrics

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



# Image Classification

which class does the image belong to?



Evaluation Metrics

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



# Image Classification

which class does the image belong to?



Evaluation Metrics

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

$$\text{Miscl. error} = 1 - \text{Accuracy}$$



# Image Classification

which class does the image belong to?



Evaluation Metrics

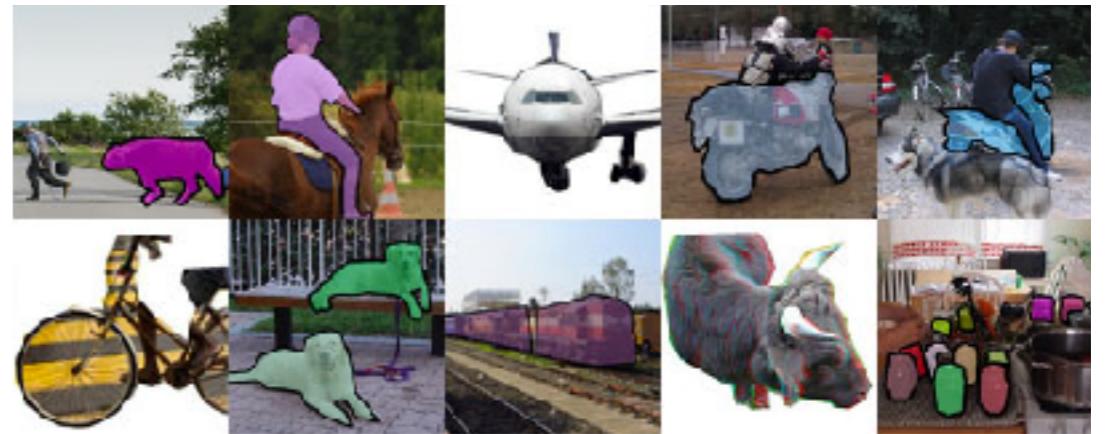
$$\text{Accuracy} = \frac{\text{\# correct predictions}}{\text{\# total predictions}}$$

$$\text{Miscl. error} = 1 - \text{Accuracy}$$

Top-k Error: k guesses to match the GT



# Datasets



*from the COCO Dataset*

# Datasets



*from ILSVRC 2016, Places Dataset*



*from the COCO Dataset*

# Datasets



- ◆ Scene photographs from image search engines, non-uniform distribution of images per category.



*from ILSVRC 2016, Places Dataset*



*from the COCO Dataset*

# Datasets



- ◆ Scene photographs from image search engines, non-uniform distribution of images per category.
- ◆ 8M train, 36K val, 328K test, 365 classes.



*from ILSVRC 2016, Places Dataset*



*from the COCO Dataset*

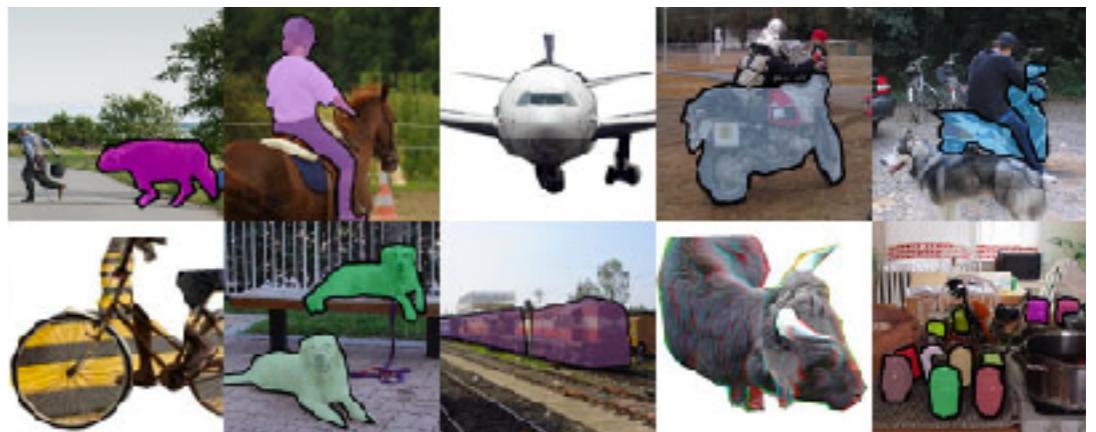
# Datasets



- ◆ Scene photographs from image search engines, non-uniform distribution of images per category.
- ◆ 8M train, 36K val, 328K test, 365 classes.
- ◆ Standardized evaluation (top-k error).



*from ILSVRC 2016, Places Dataset*



*from the COCO Dataset*

# Datasets



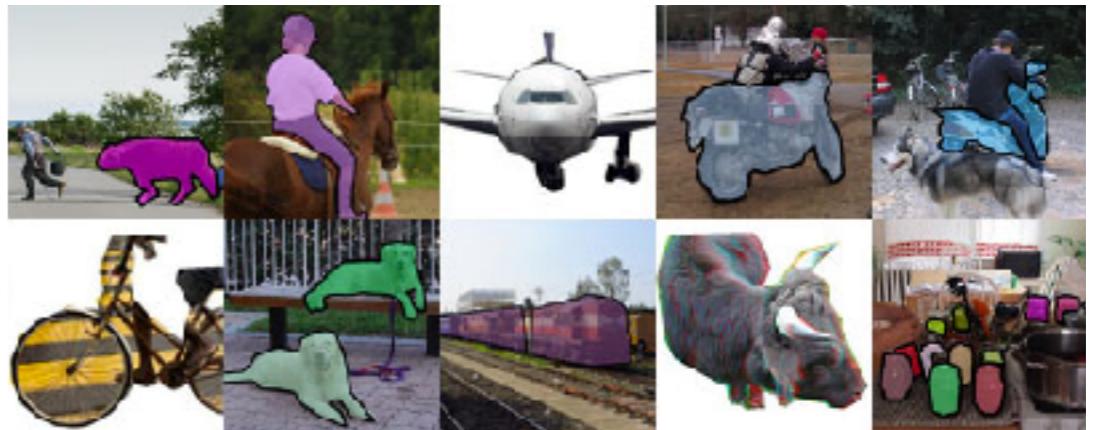
- ◆ Scene photographs from image search engines, non-uniform distribution of images per category.
- ◆ 8M train, 36K val, 328K test, 365 classes.
- ◆ Standardized evaluation (top-k error).



*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.



*from the COCO Dataset*

# Datasets



- ◆ Scene photographs from image search engines, non-uniform distribution of images per category.
- ◆ 8M train, 36K val, 328K test, 365 classes.
- ◆ Standardized evaluation (top-k error).



*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*

# Datasets



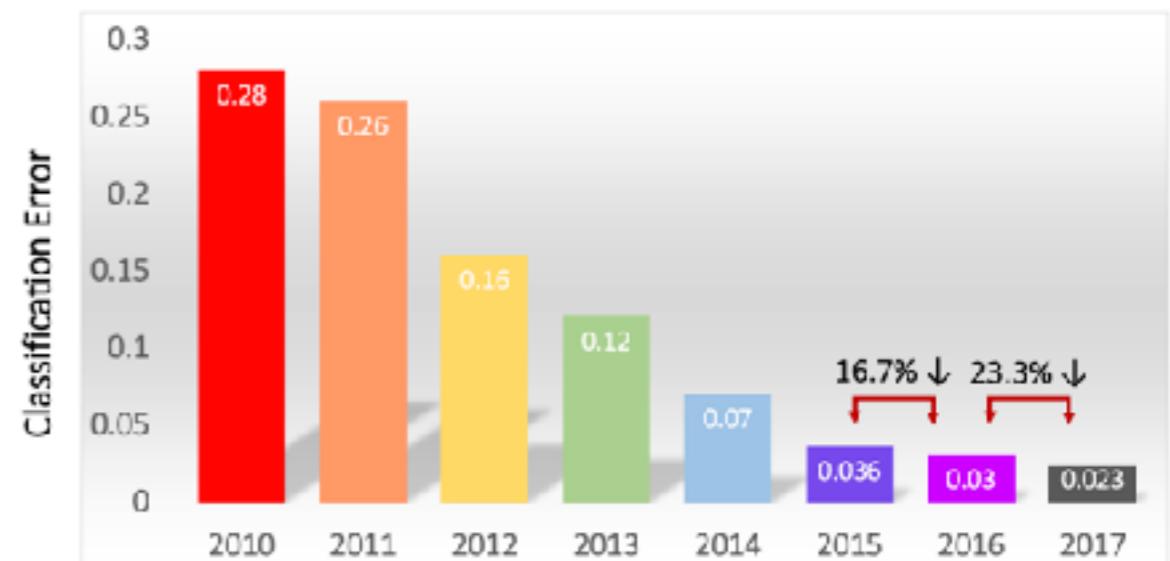
- ◆ Scene photographs from image search engines, non-uniform distribution of images per category.
- ◆ 8M train, 36K val, 328K test, 365 classes.
- ◆ Standardized evaluation (top-k error).
- ◆ Results over years.



**Output:**  
Scale  
T-shirt  
Steel drum  
Drumstick  
Mud turtle



**Output:**  
Scale  
T-shirt  
Giant panda  
Drumstick  
Mud turtle



# Prototype Datasets

## MNIST



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

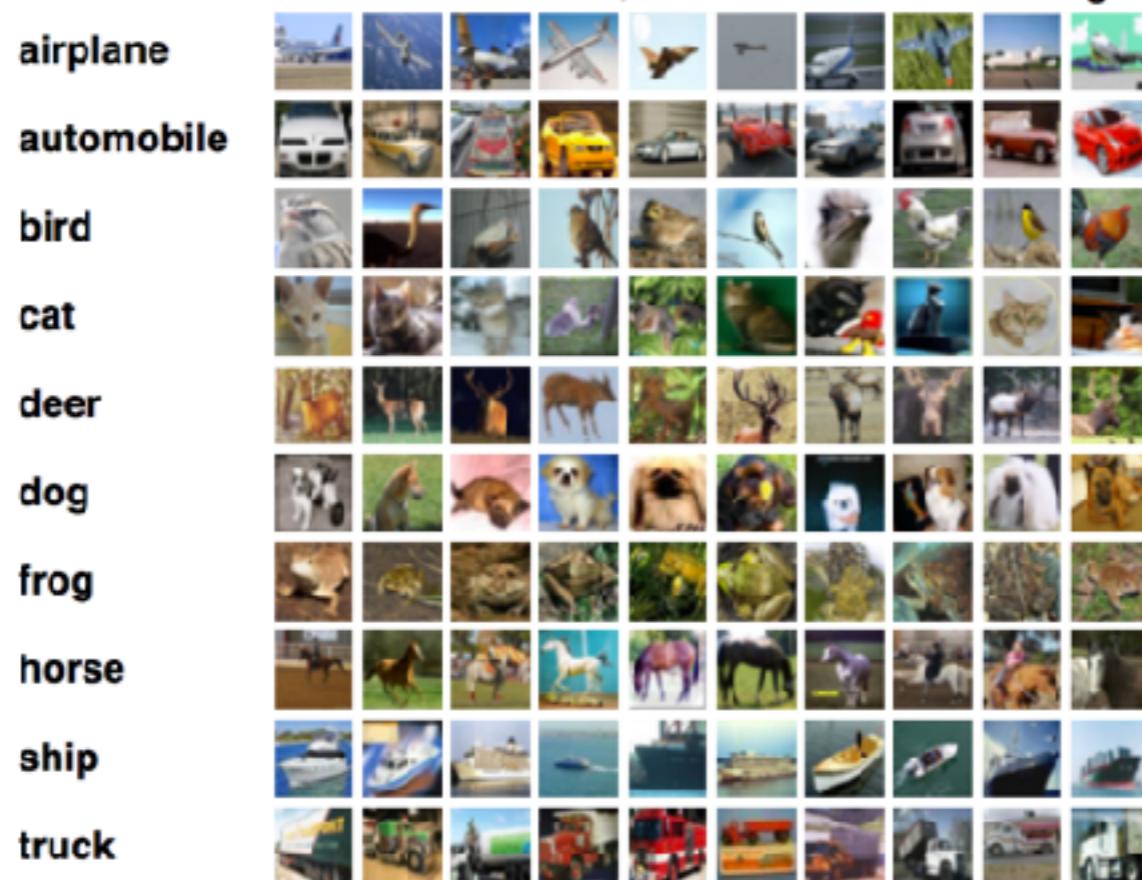
## Fashion-Mnist



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

# Prototype Datasets

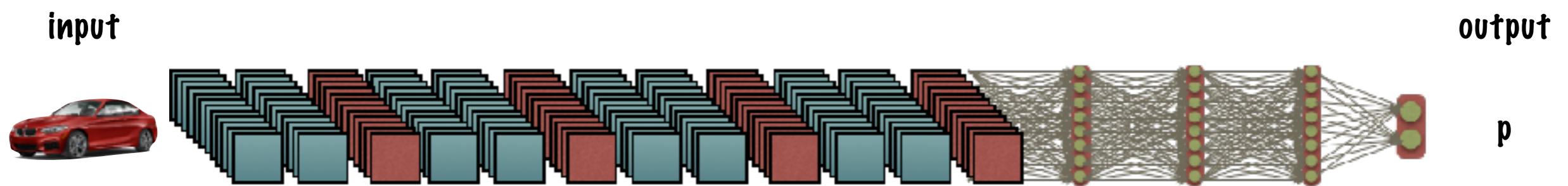
## CIFAR



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

# CNN basics

# CNN architecture



*Figure courtesy of Kilian Weinberger*

# CNN architecture

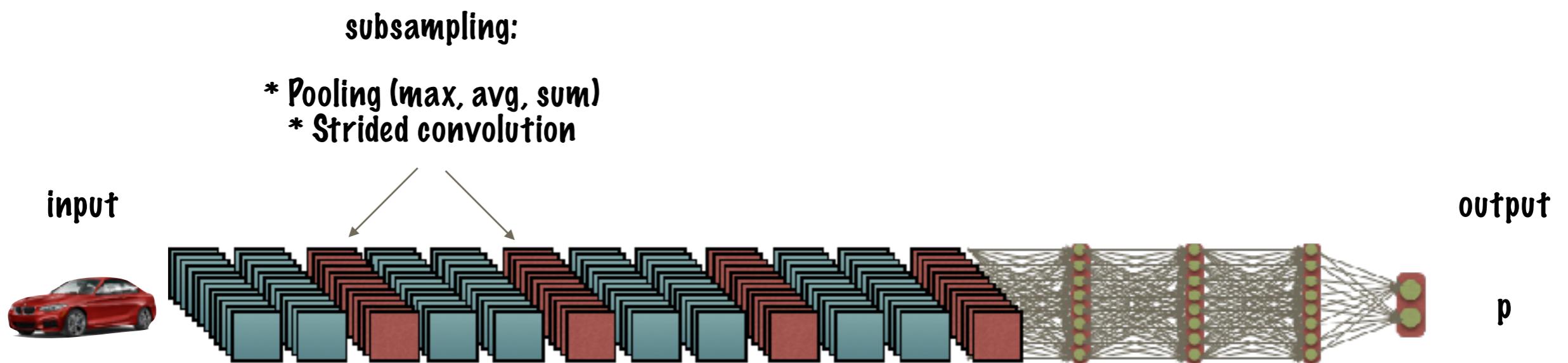


Figure courtesy of Kilian Weinberger

# CNN architecture

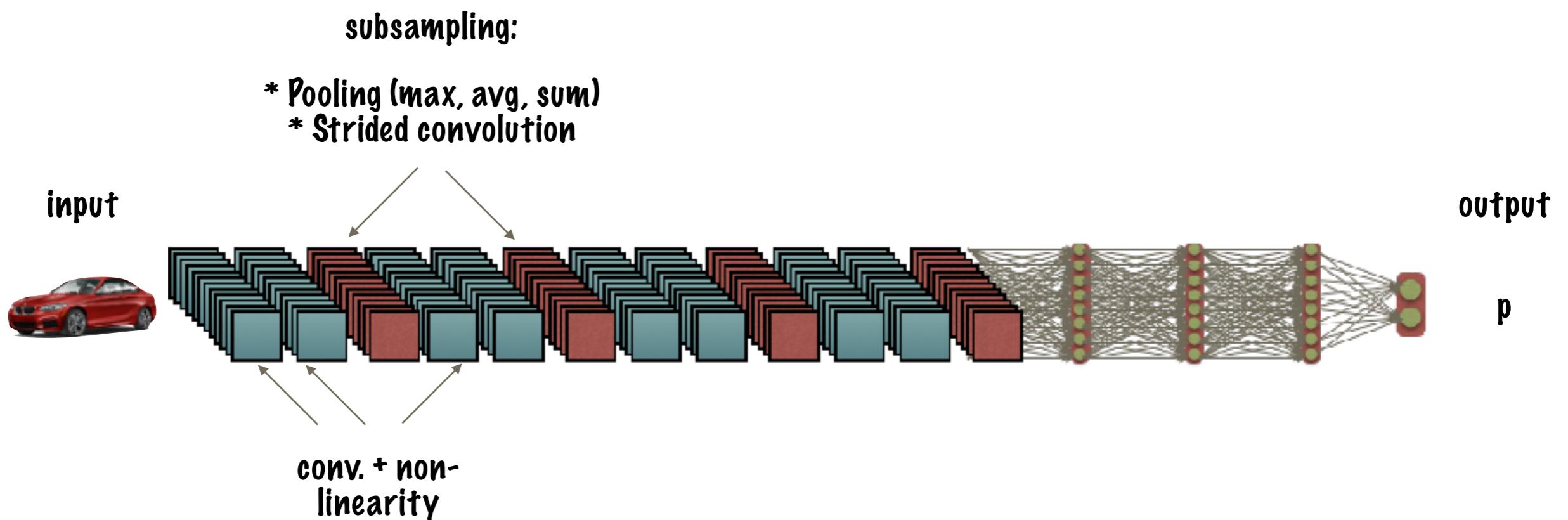
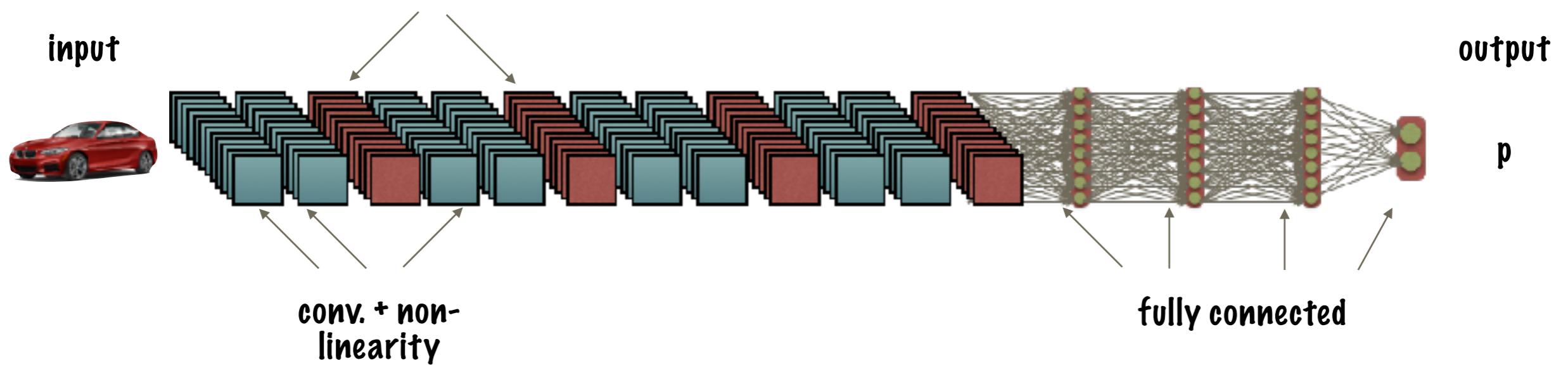


Figure courtesy of Kilian Weinberger

# CNN architecture

## subsampling:

- \* Pooling (max, avg, sum)
  - \* Strided convolution



*Figure courtesy of Kilian Weinberger*

# CNN architecture



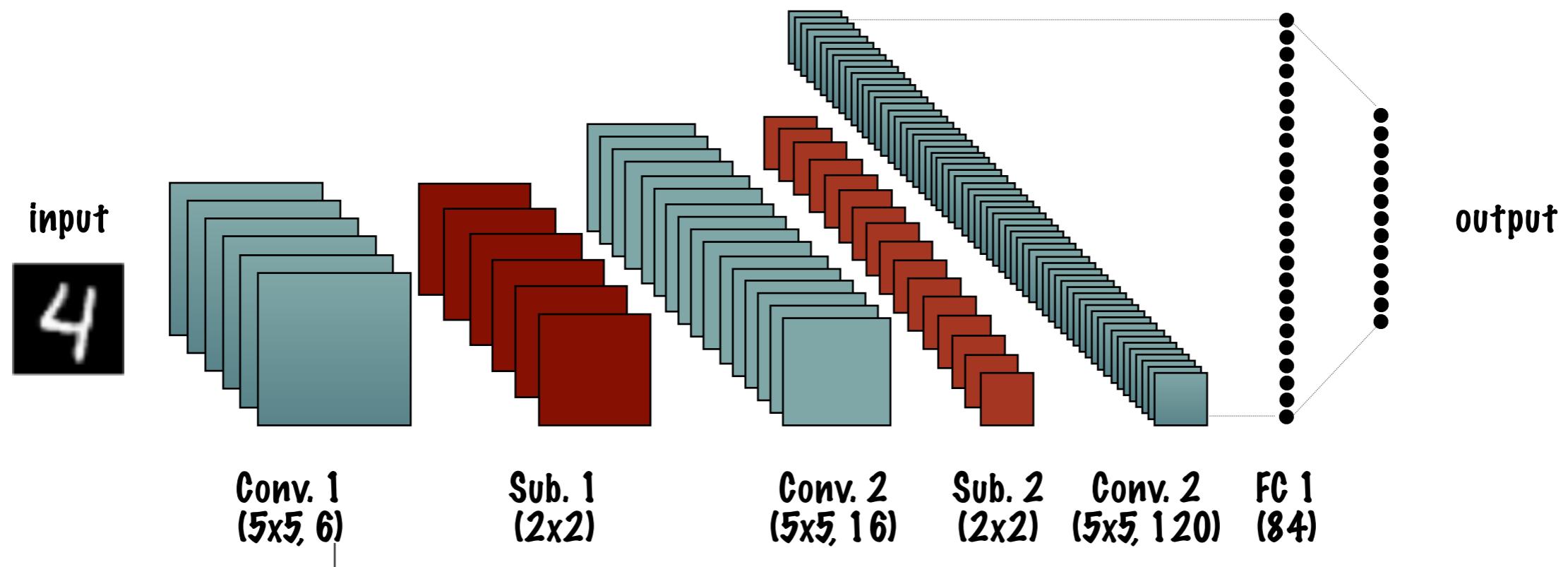
Loss: categorical crossentropy

$$l_i = - \sum_j t_{i,j} \log(p_{i,j})$$

trained with SGD

# LeNet, 1998

Object recognition:



Non-linearity: scaled tanh

# Wrap Up



# Wrap Up



# Wrap Up

training of deep  
nets unsuccessful  
(except for CNNs)

AI winter



# Wrap Up

training of deep  
nets unsuccessful  
(except for CNNs)

AI winter



from <http://quincypublicschools.com/library/contact-school/book-stack/>  
from [https://en.wikipedia.org/wiki/GeForce\\_10\\_series](https://en.wikipedia.org/wiki/GeForce_10_series)

# Wrap Up

training of deep  
nets unsuccessful  
(except for CNNs)

AI winter



from <http://quincypublicschools.com/library/contact-school/book-stack/>  
from [https://en.wikipedia.org/wiki/GeForce\\_10\\_series](https://en.wikipedia.org/wiki/GeForce_10_series)

# Wrap Up

training of deep  
nets unsuccessful  
(except for CNNs)

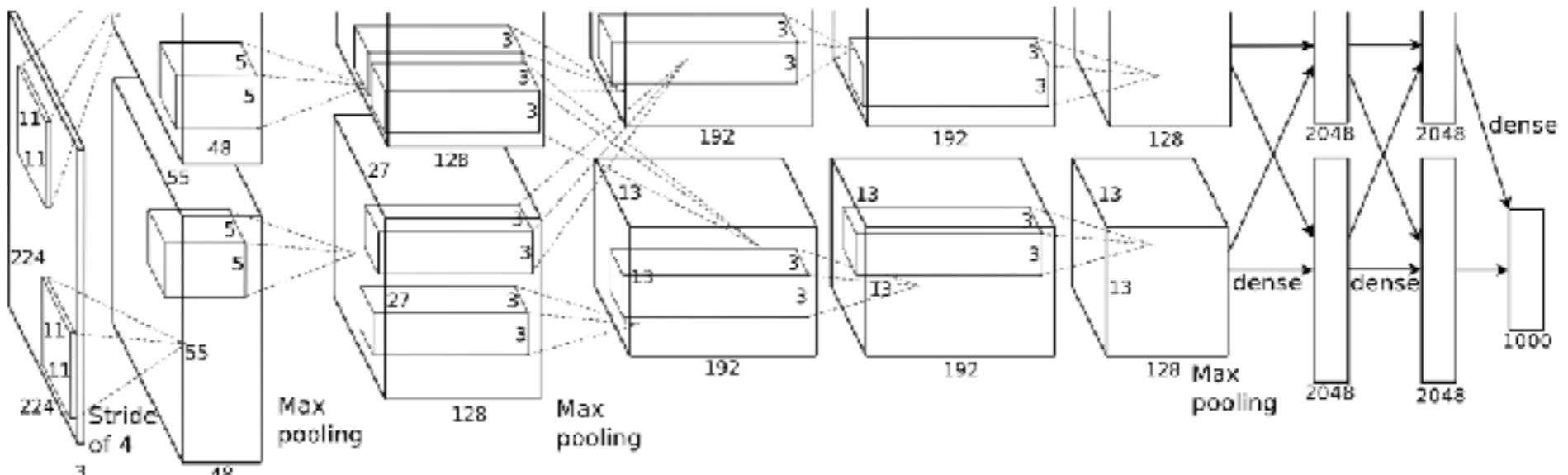
AI winter



**2012 - 2014:**  
**AlexNet, VGG & GoogLeNet**

# AlexNet, 2012

Image classification:

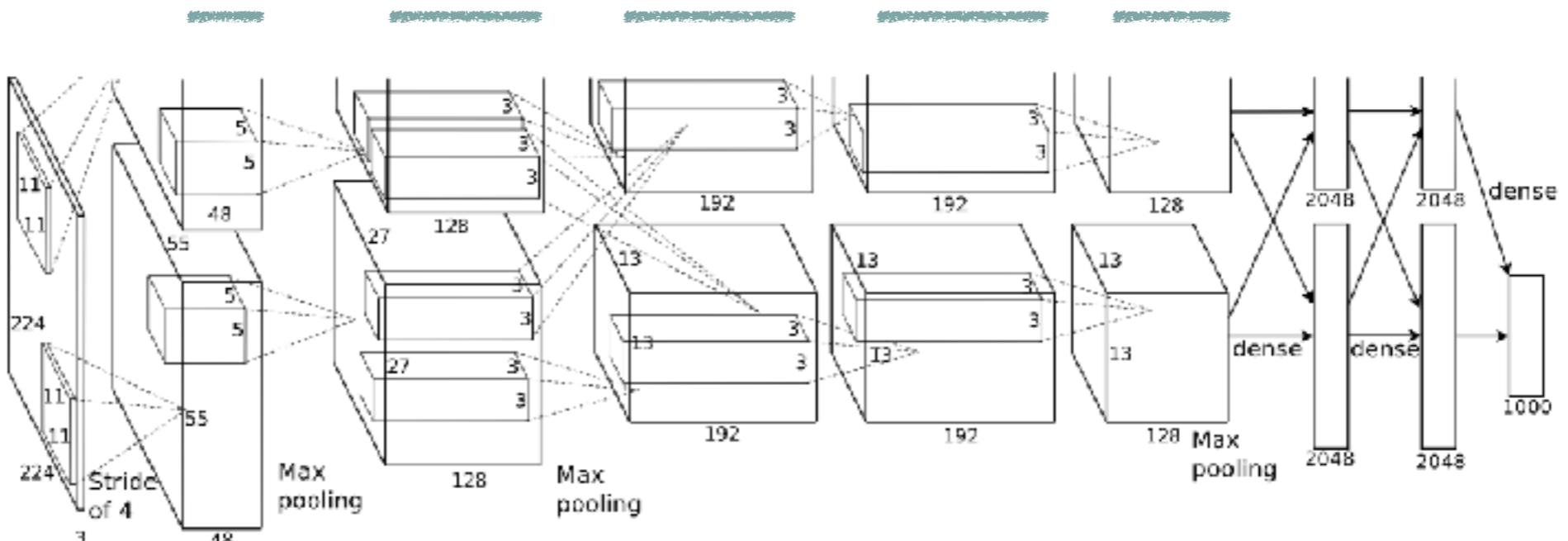


from [Krizhevsky et al., 2012]

[Krizhevsky et al. 2012]

# AlexNet, 2012

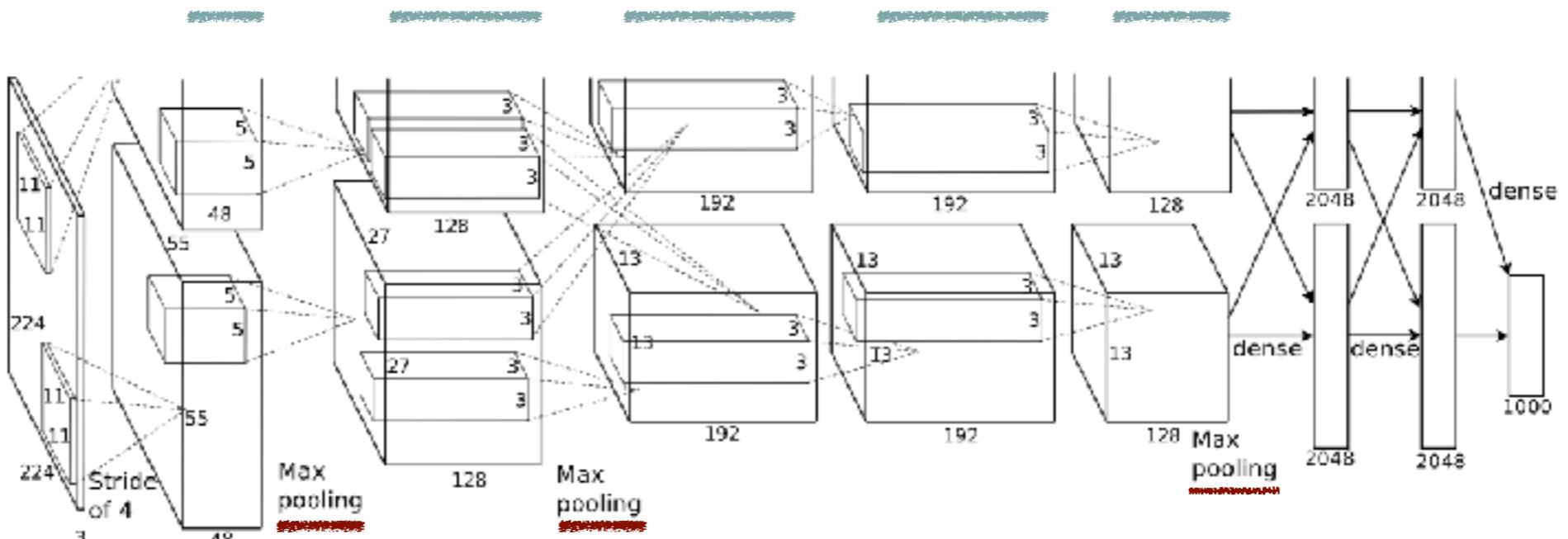
Image classification:



from [Krizhevsky et al., 2012]

# AlexNet, 2012

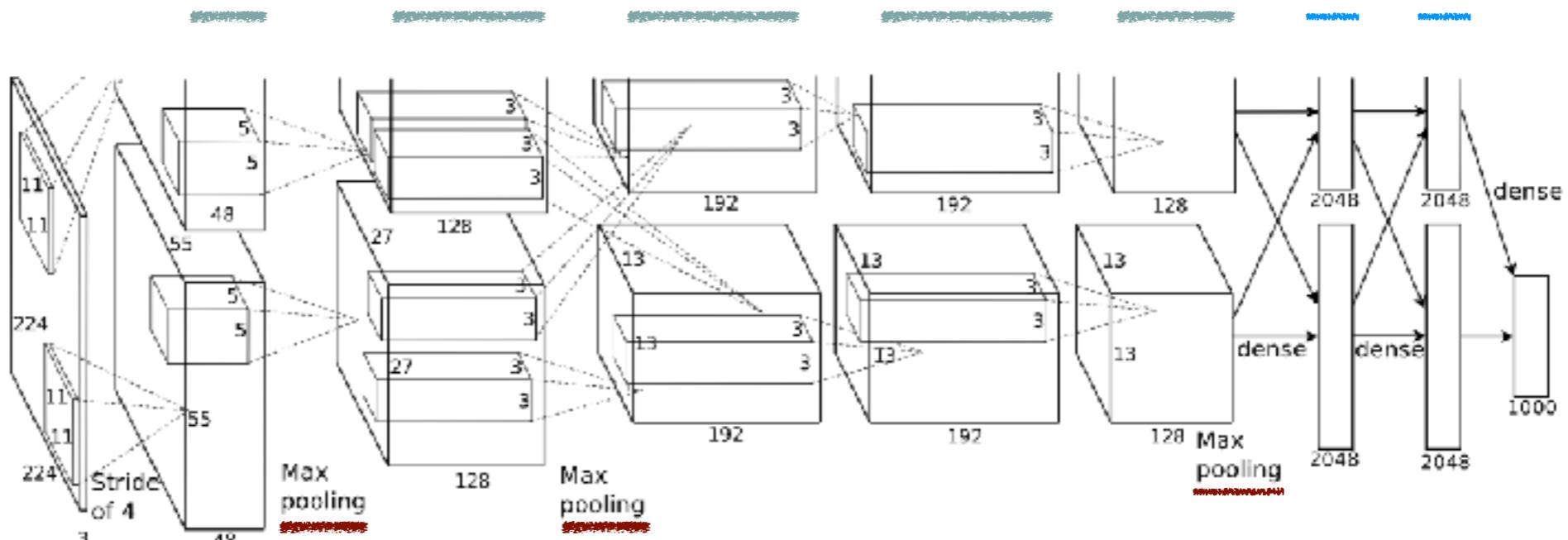
Image classification:



from [Krizhevsky et al., 2012]

# AlexNet, 2012

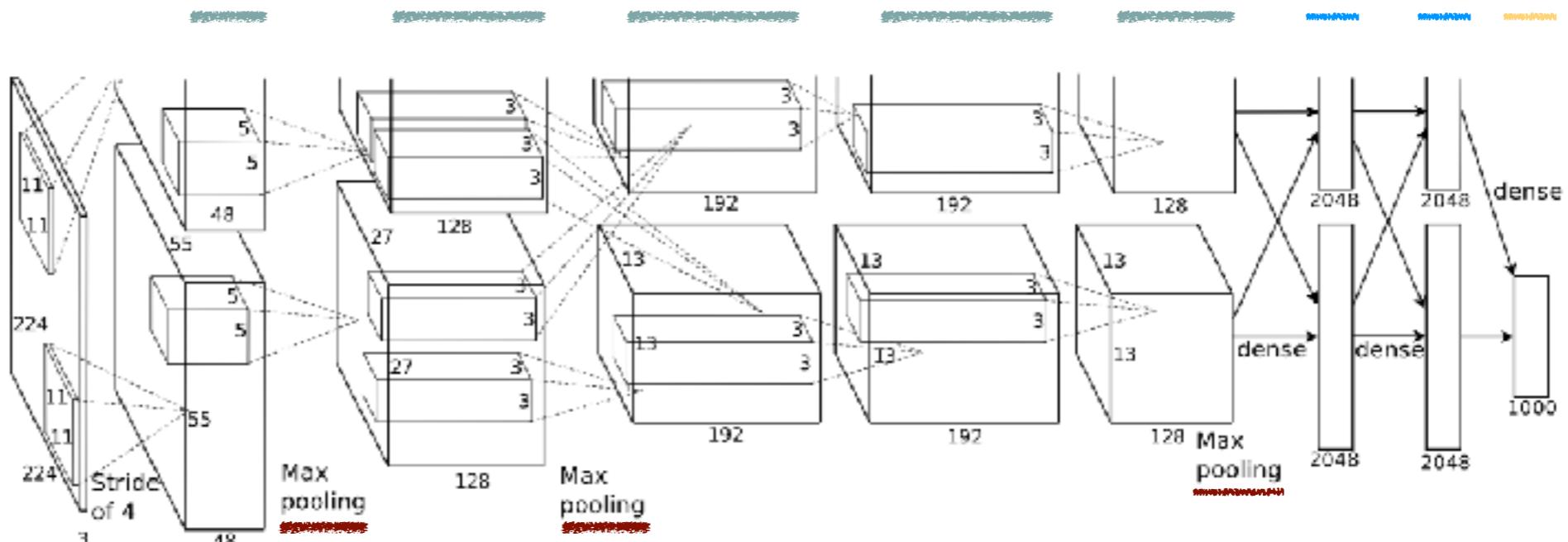
Image classification:



from [Krizhevsky et al., 2012]

# AlexNet, 2012

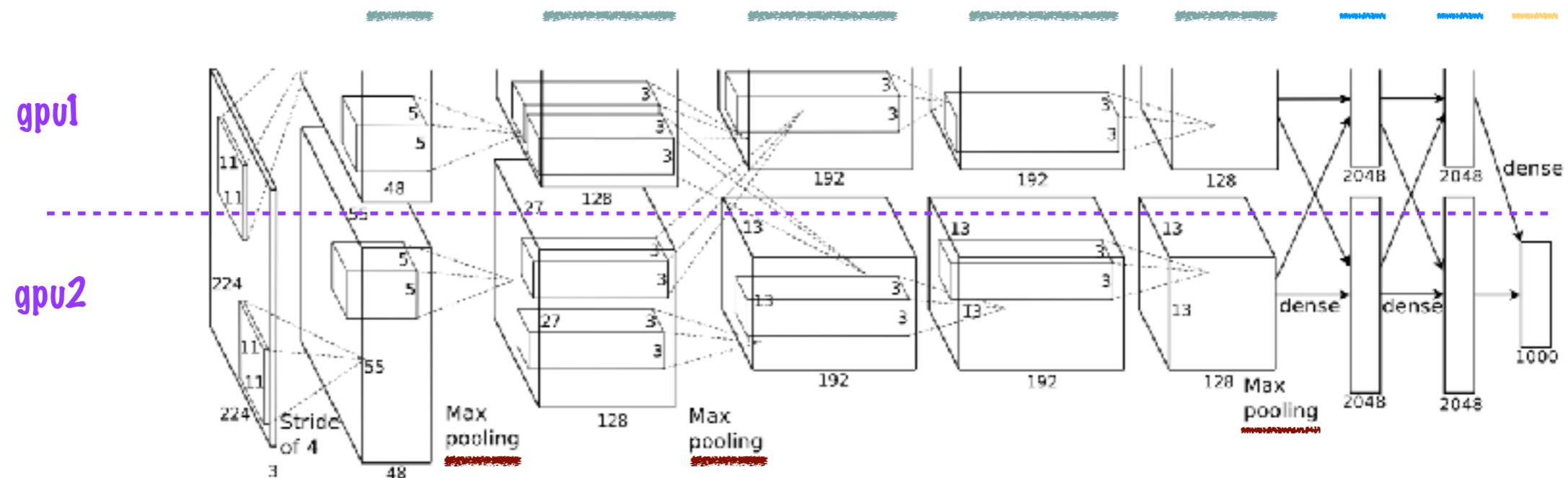
Image classification:



from [Krizhevsky et al., 2012]

# AlexNet, 2012

Image classification:

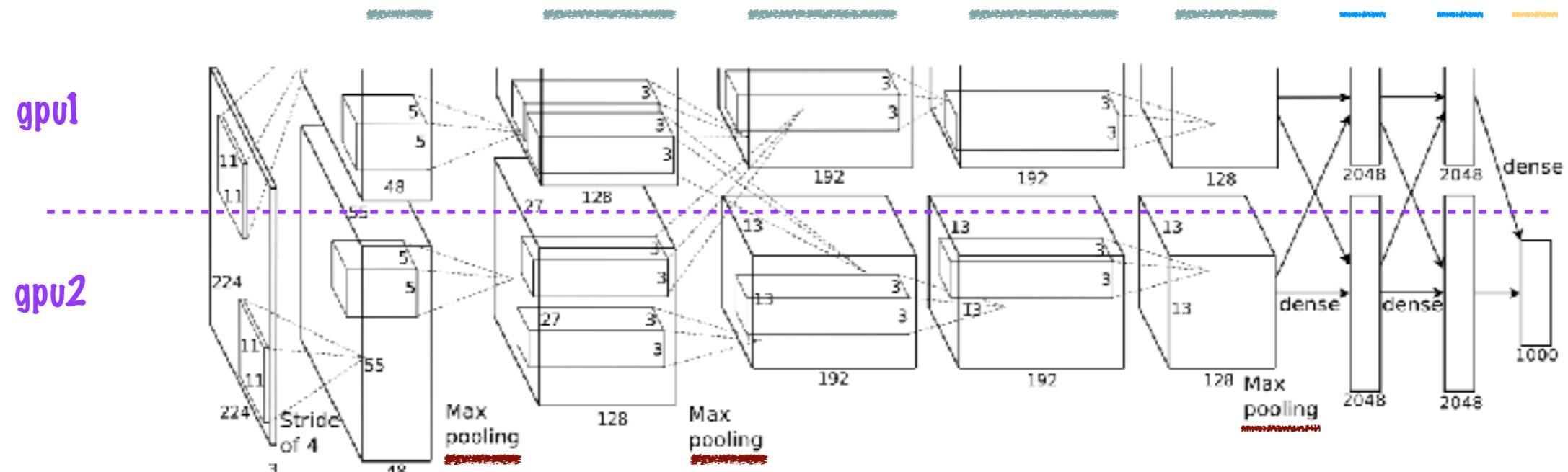


from [Krizhevsky et al., 2012]

[Krizhevsky et al. 2012]

# AlexNet, 2012

Image classification:



from [Krizhevsky et al., 2012]

- \* Won ILSVRC 2012 challenge by a large margin
- \* ReLU non-linearity
- \* Dropout
- \* Data Augmentation

# VGG, 2014

Image classification:



# VGG, 2014

Image classification:



- (Very) deep network (up to 19 layers)

# VGG, 2014

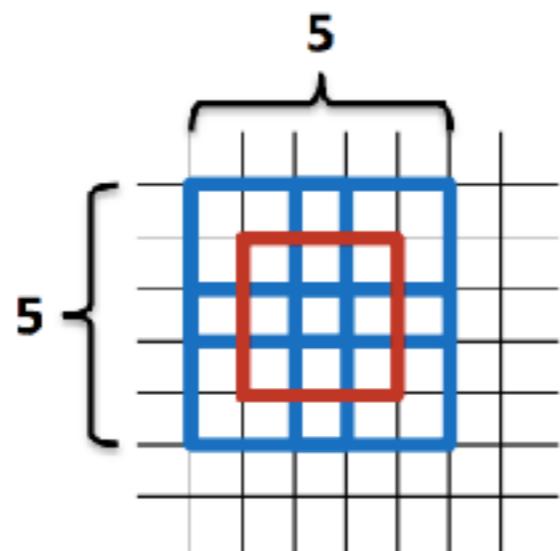
Image classification:



- (Very) deep network (up to 19 layers)
- 3x3 filters, stride 1, pad 1

# VGG, 2014

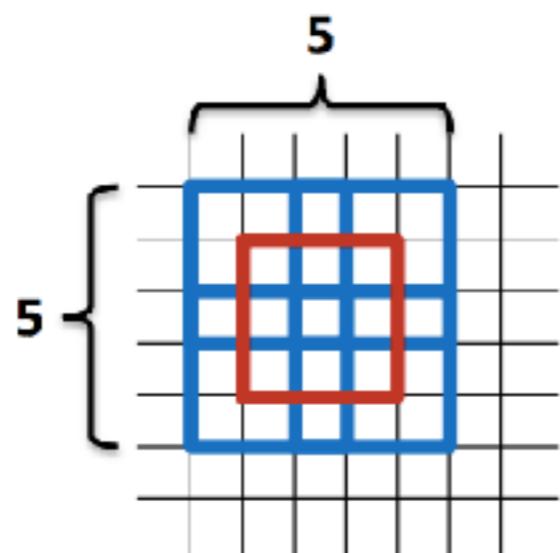
Image classification:



- (Very) deep network (up to 19 layers)
- 3x3 filters, stride 1, pad 1
- Stacked conv.:
  - \* bigger RF
  - \* more non-linearity

# VGG, 2014

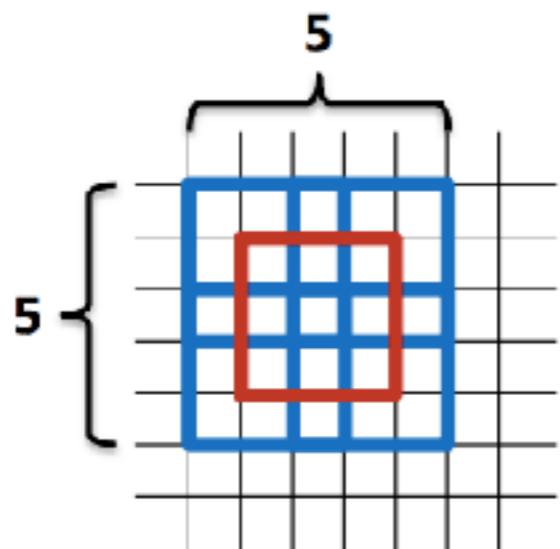
Image classification:



- (Very) deep network (up to 19 layers)
- 3x3 filters, stride 1, pad 1
- Stacked conv.:
  - \* bigger RF
  - \* more non-linearity
- 2x2 non-overlapping max-pooling

# VGG, 2014

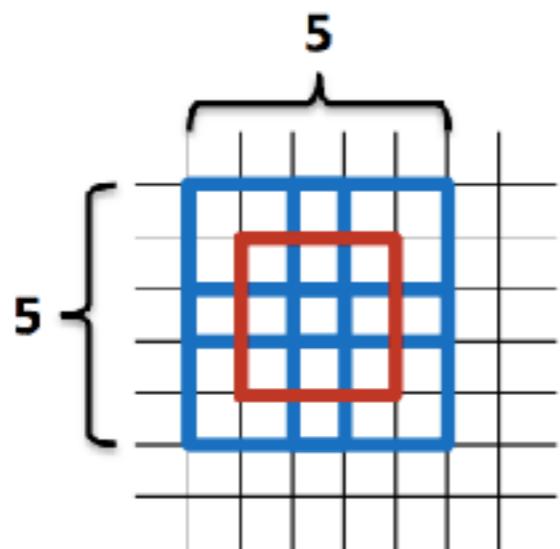
Image classification:



- (Very) deep network (up to 19 layers)
- 3x3 filters, stride 1, pad 1
- Stacked conv.:
  - \* bigger RF
  - \* more non-linearity
- 2x2 non-overlapping max-pooling
- # features increases as we go deeper

# VGG, 2014

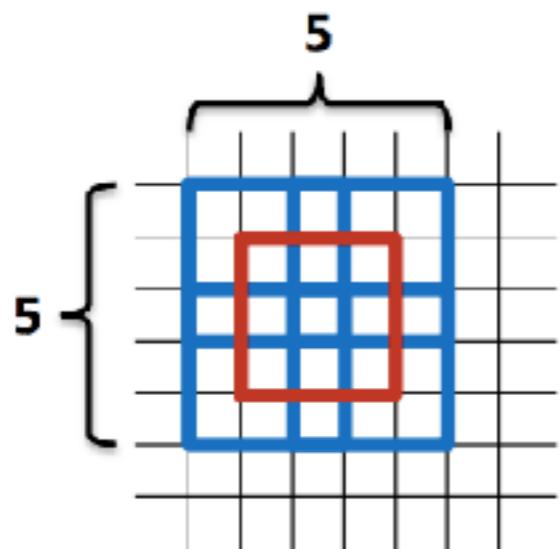
Image classification:



- (Very) deep network (up to 19 layers)
- 3x3 filters, stride 1, pad 1
- Stacked conv.:
  - \* bigger RF
  - \* more non-linearity
- 2x2 non-overlapping max-pooling
- # features increases as we go deeper
- ReLU non-linearity

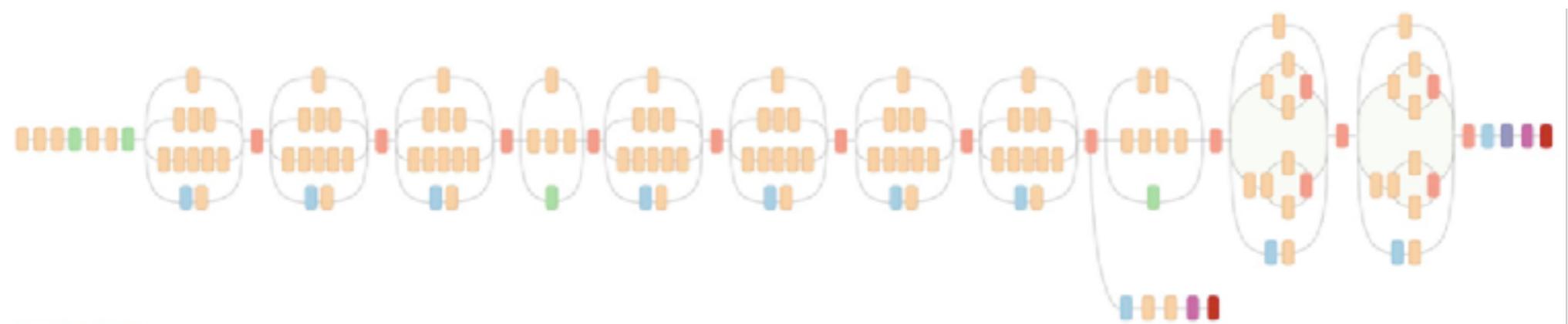
# VGG, 2014

Image classification:



- (Very) deep network (up to 19 layers)
- 3x3 filters, stride 1, pad 1
- Stacked conv.:
  - \* bigger RF
  - \* more non-linearity
- 2x2 non-overlapping max-pooling
- # features increases as we go deeper
- ReLU non-linearity
- Data Augmentation & dropout

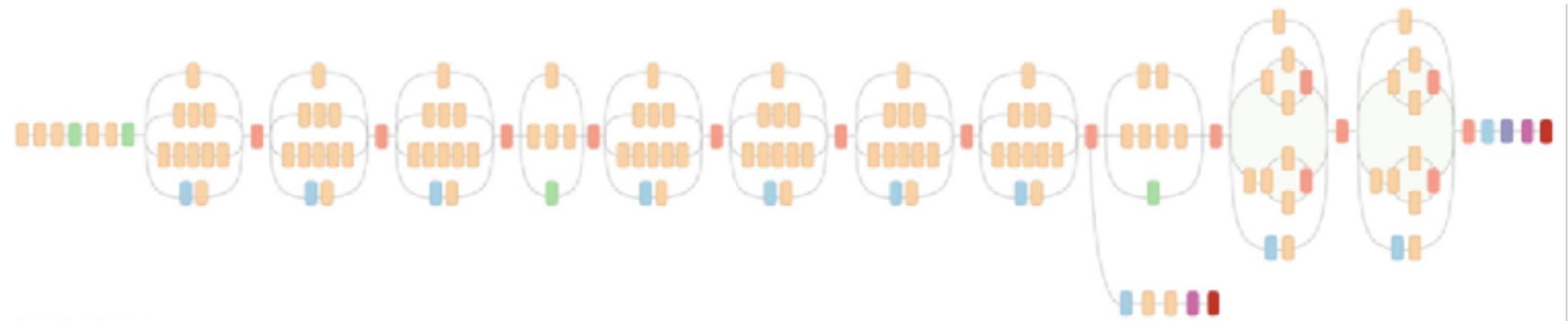
# GoogLeNet, 2015



[Szegedy et al. 2015]

Figure: <https://adesshpande3.github.io/adesshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>

# GoogLeNet, 2015

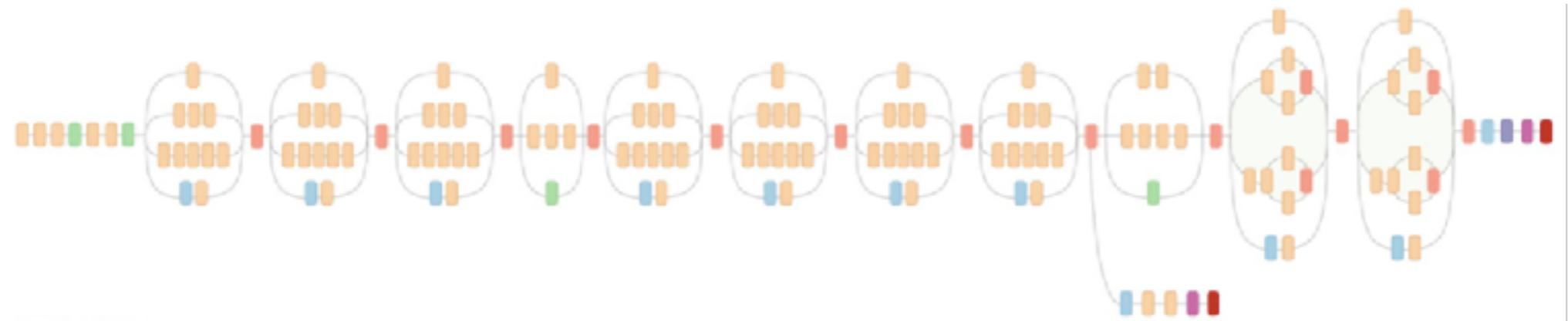


- \* Won ILSVRC 2014 challenge

[Szegedy et al. 2015]

Figure: <https://adesshpande3.github.io/adesshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>

# GoogLeNet, 2015

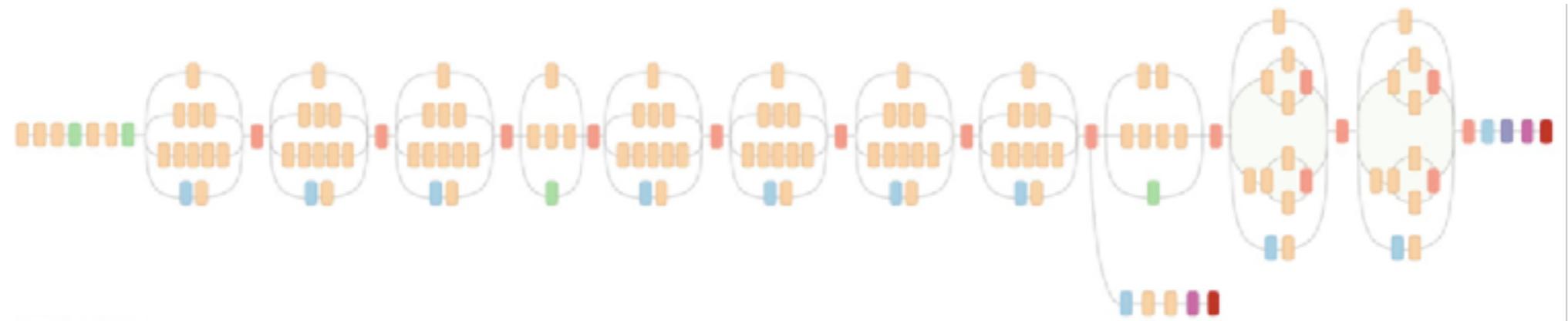


- \* Won ILSVRC 2014 challenge
- \* 12x fewer parameters than AlexNet

[Szegedy et al. 2015]

Figure: <https://adesshpande3.github.io/adesshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>

# GoogLeNet, 2015

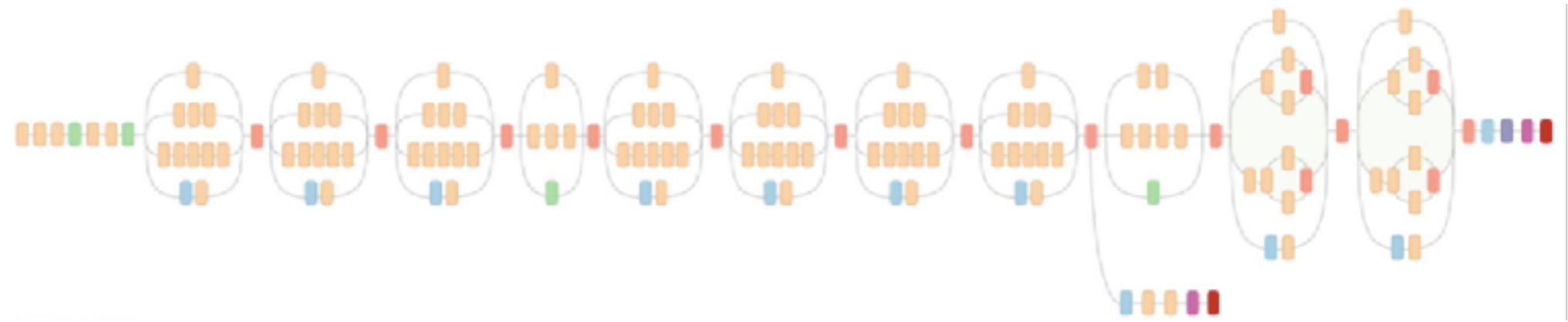


- \* Won ILSVRC 2014 challenge
- \* 12x fewer parameters than AlexNet
- \* 22 layers

[Szegedy et al. 2015]

Figure: <https://adesshpande3.github.io/adesshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>

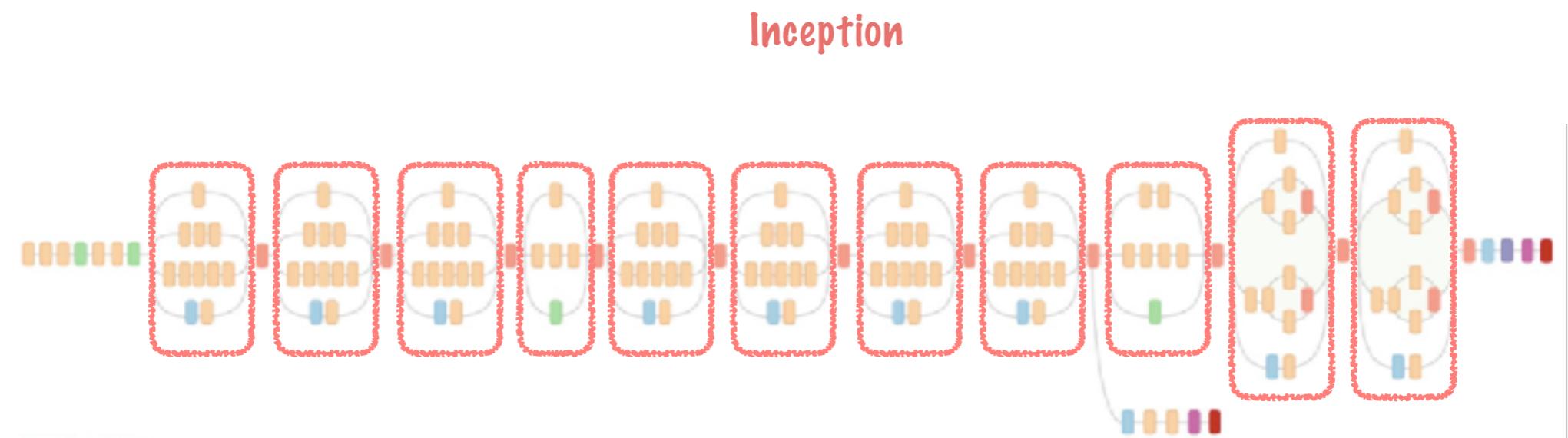
# GoogLeNet, 2015



- \* Won ILSVRC 2014 challenge
- \* 12x fewer parameters than AlexNet
- \* 22 layers
- \* No FC layers (avg pooling)

[Szegedy et al. 2015]

# GoogLeNet, 2015

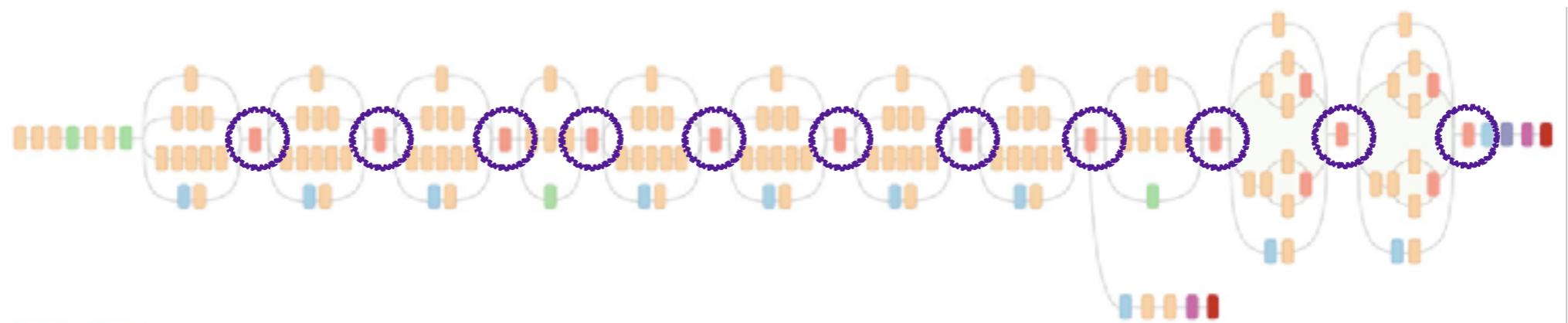


- \* Won ILSVRC 2014 challenge
- \* 12x fewer parameters than AlexNet
- \* 22 layers
- \* No FC layers (avg pooling)

[Szegedy et al. 2015]

# GoogLeNet, 2015

Concatenation

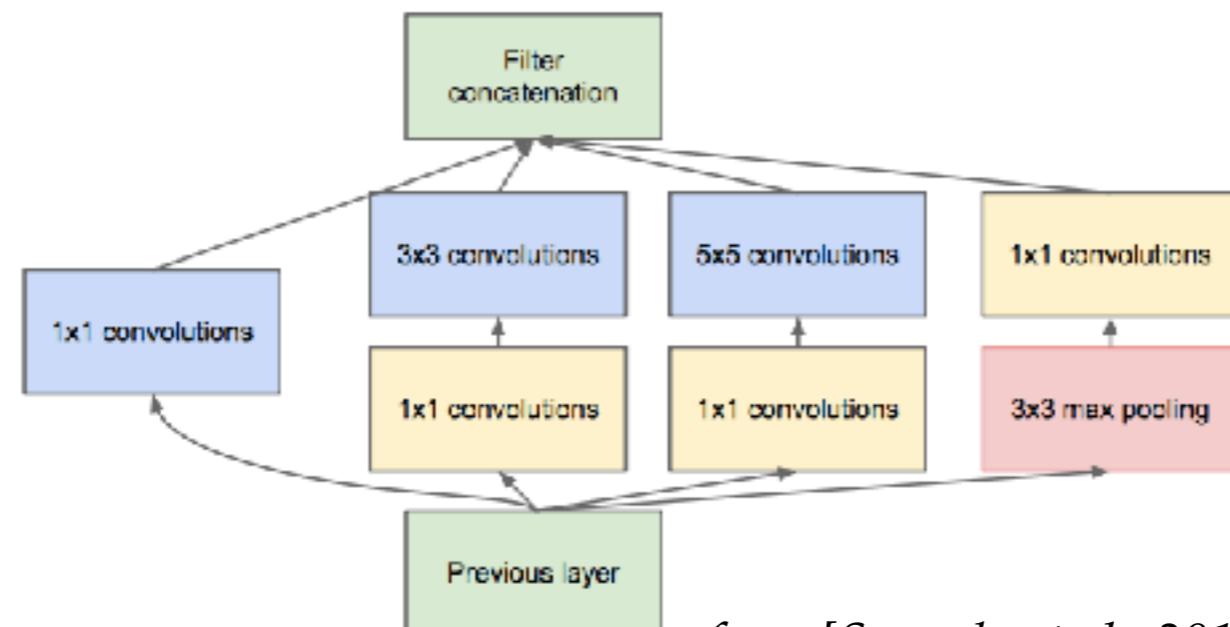


- \* Won ILSVRC 2014 challenge
- \* 12x fewer parameters than AlexNet
- \* 22 layers
- \* No FC layers (avg pooling)

[Szegedy et al. 2015]

# GoogLeNet, 2015

Inception module:

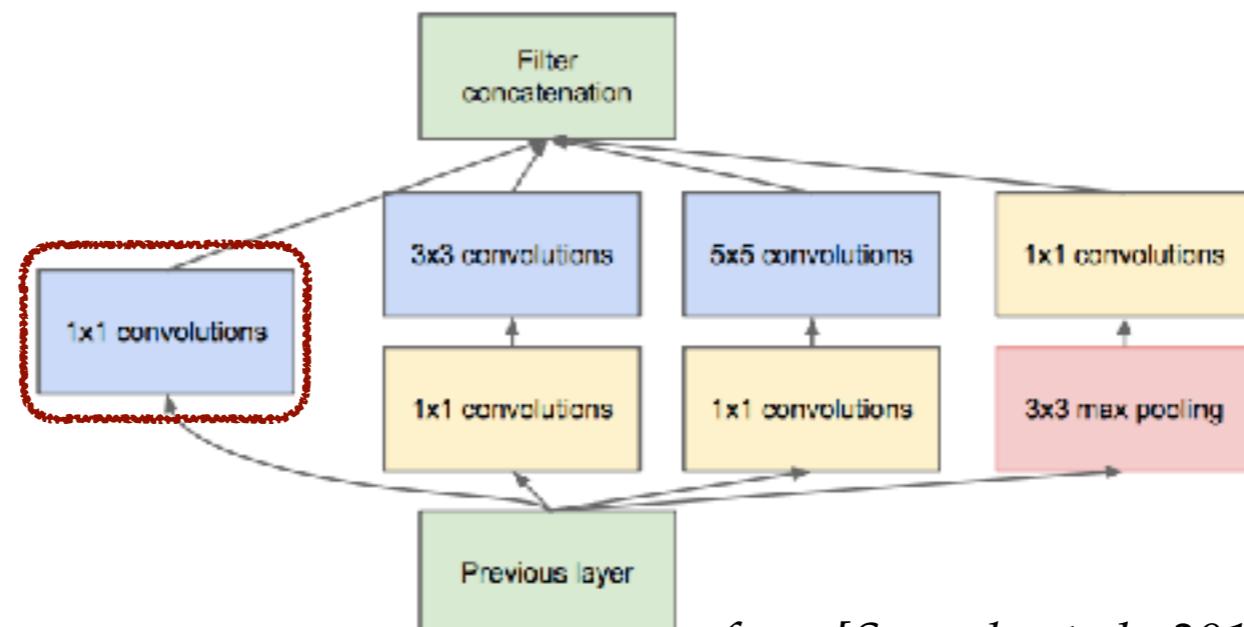


from [Szegedy et al., 2015]

# GoogLeNet, 2015

Inception module:

**1x1 conv.**

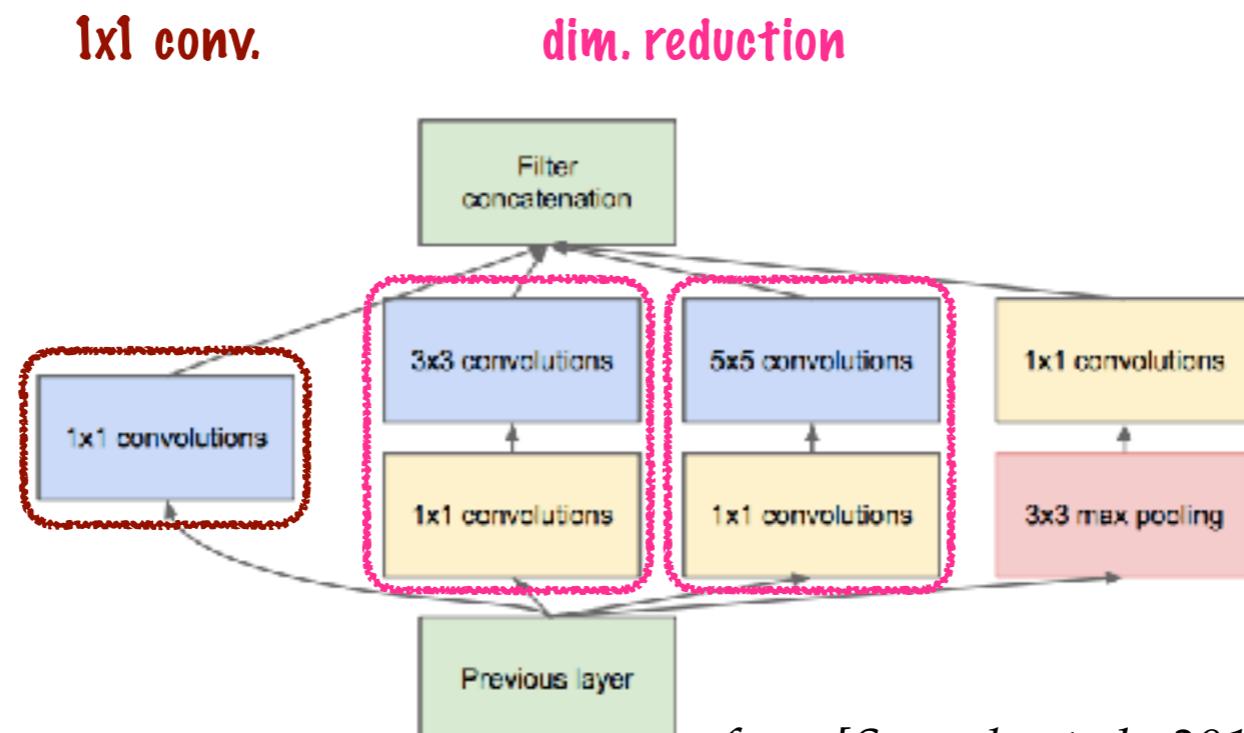


from [Szegedy et al., 2015]

[Szegedy et al. 2015]

# GoogLeNet, 2015

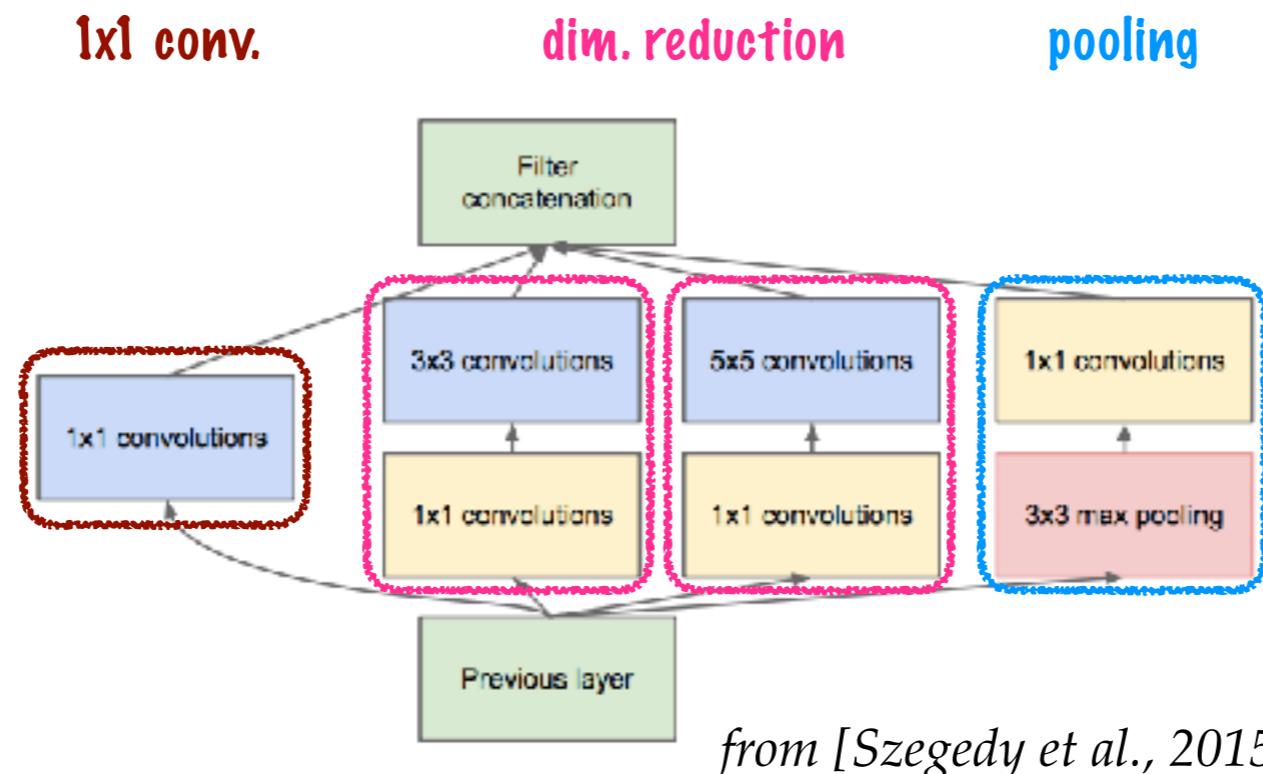
Inception module:



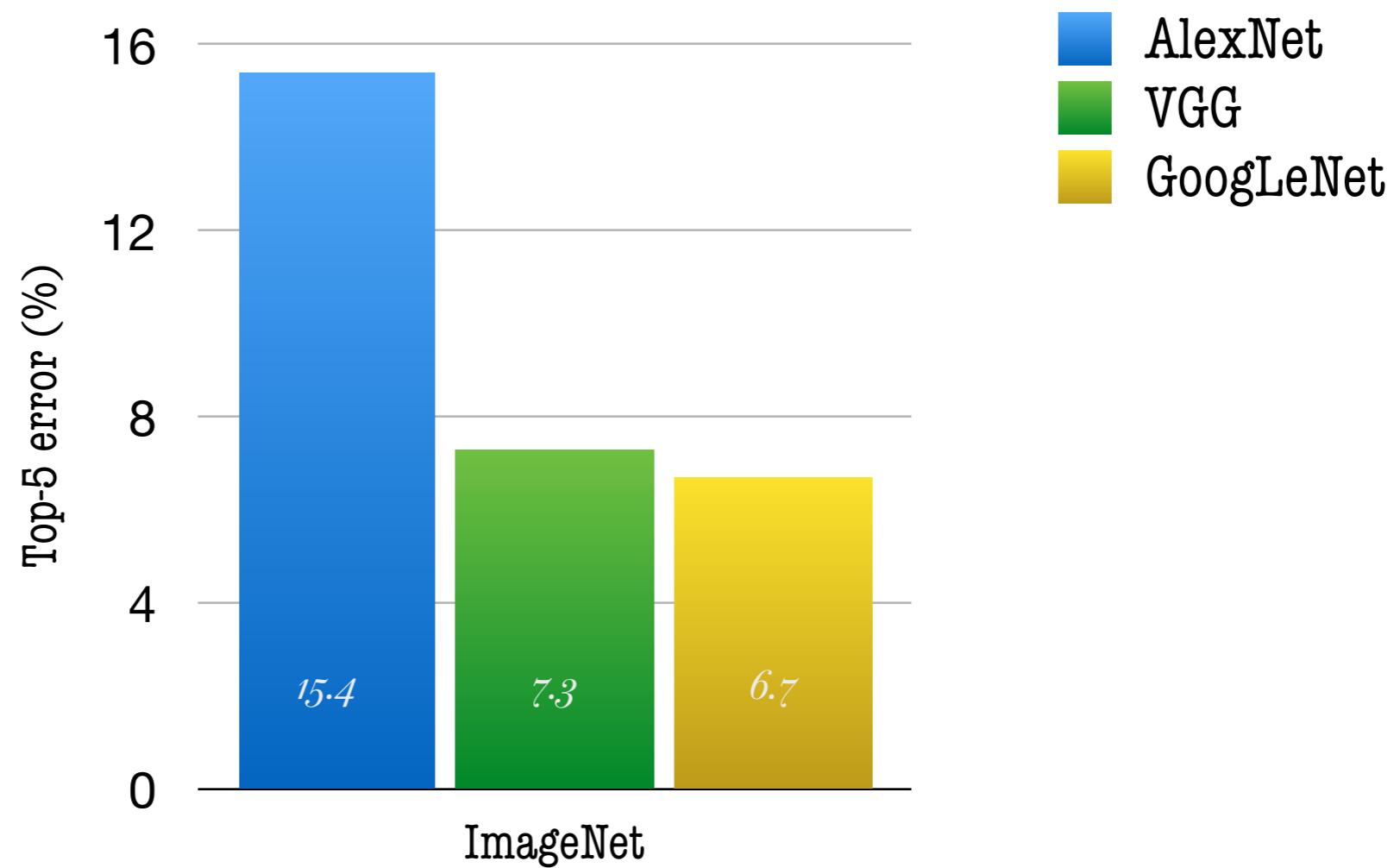
from [Szegedy et al., 2015]

# GoogLeNet, 2015

Inception module:

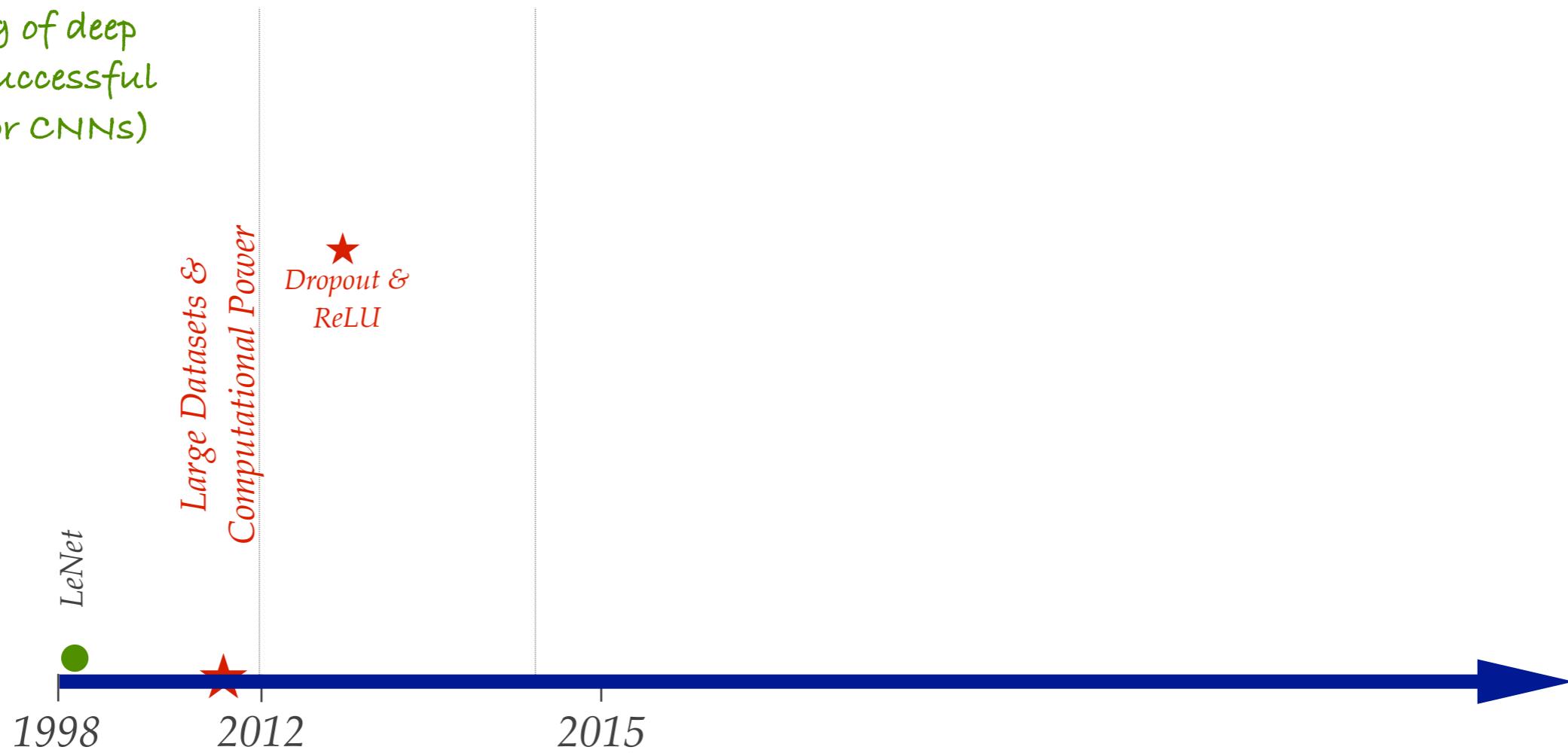


# ImageNet Results



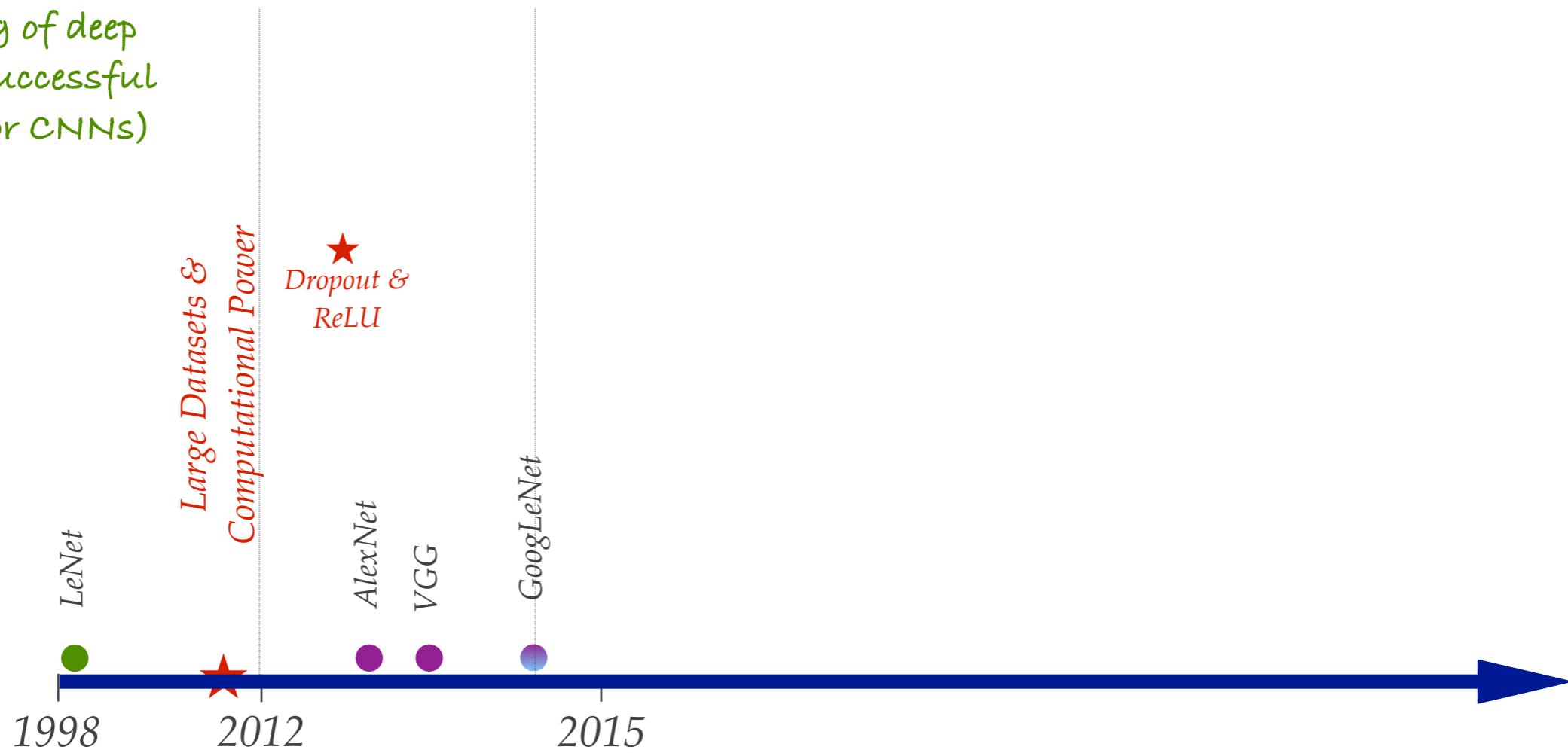
# Wrap Up

training of deep  
nets unsuccessful  
(except for CNNs)

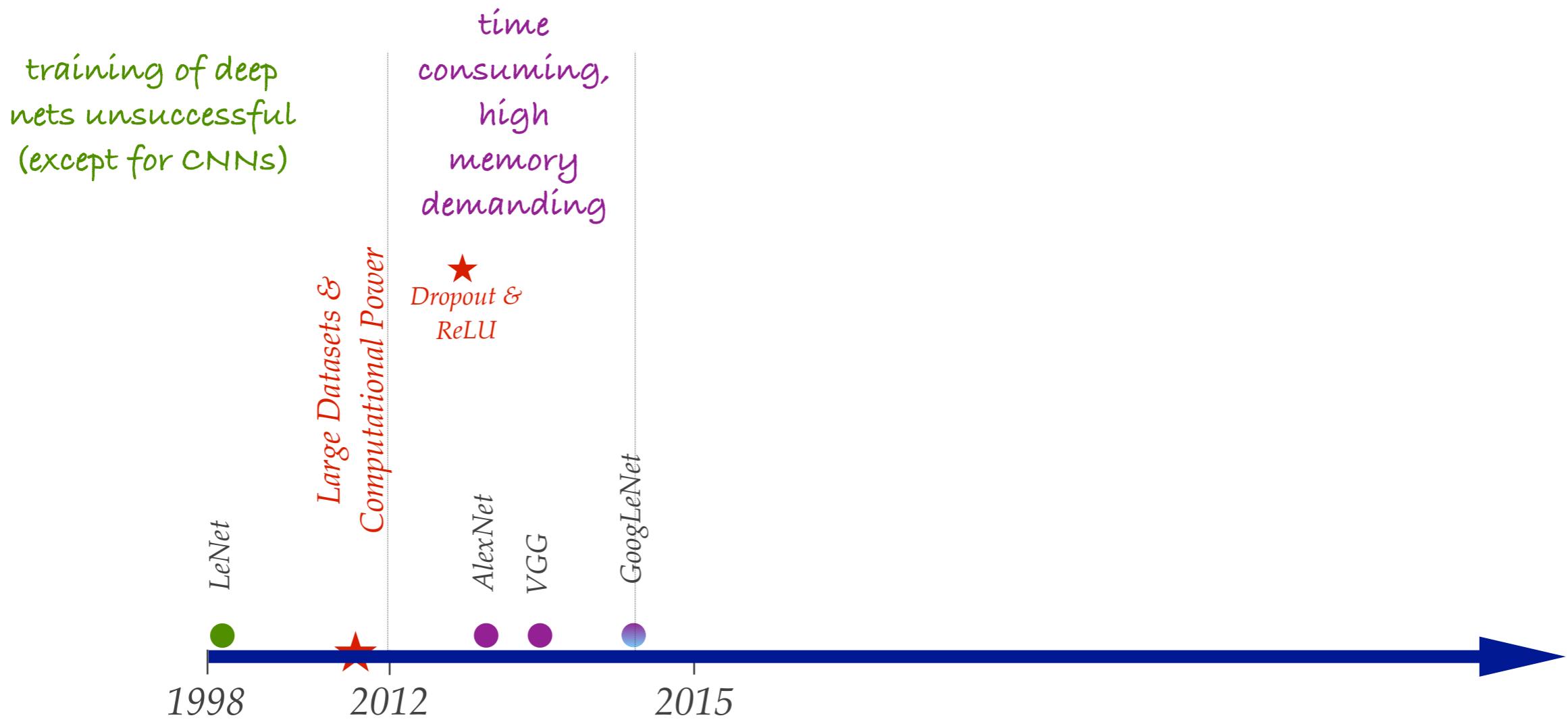


# Wrap Up

training of deep  
nets unsuccessful  
(except for CNNs)



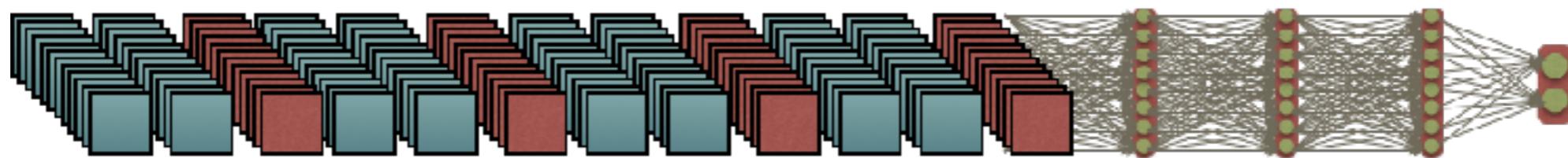
# Wrap Up



**2014 - 2015:  
DSN, FitNets, Highway Nets &  
ResNets**

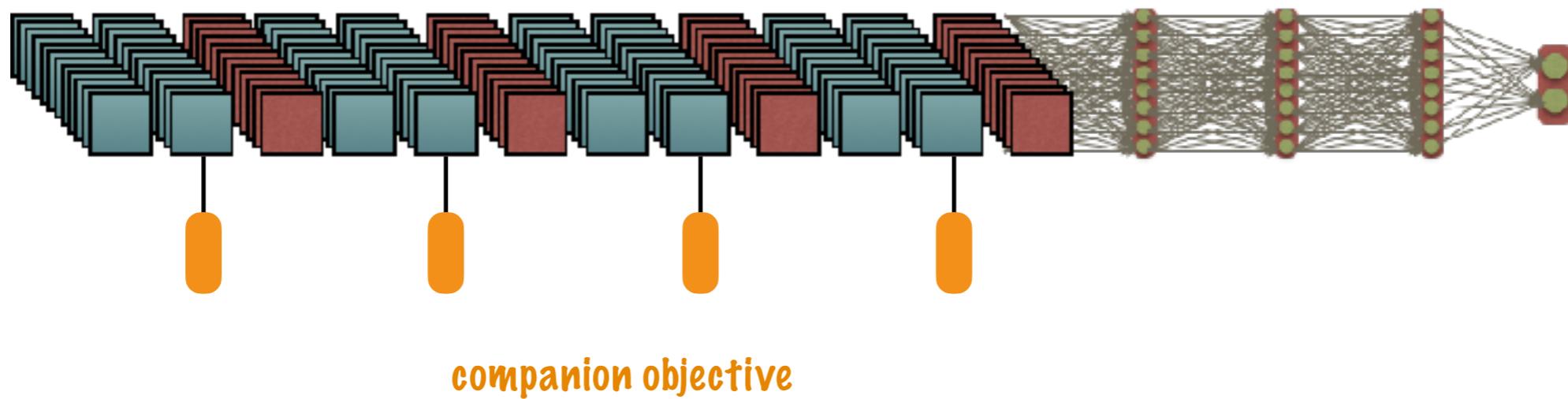
# Deeply Supervised Networks

Adding intermediate supervision (discriminative loss):



# Deeply Supervised Networks

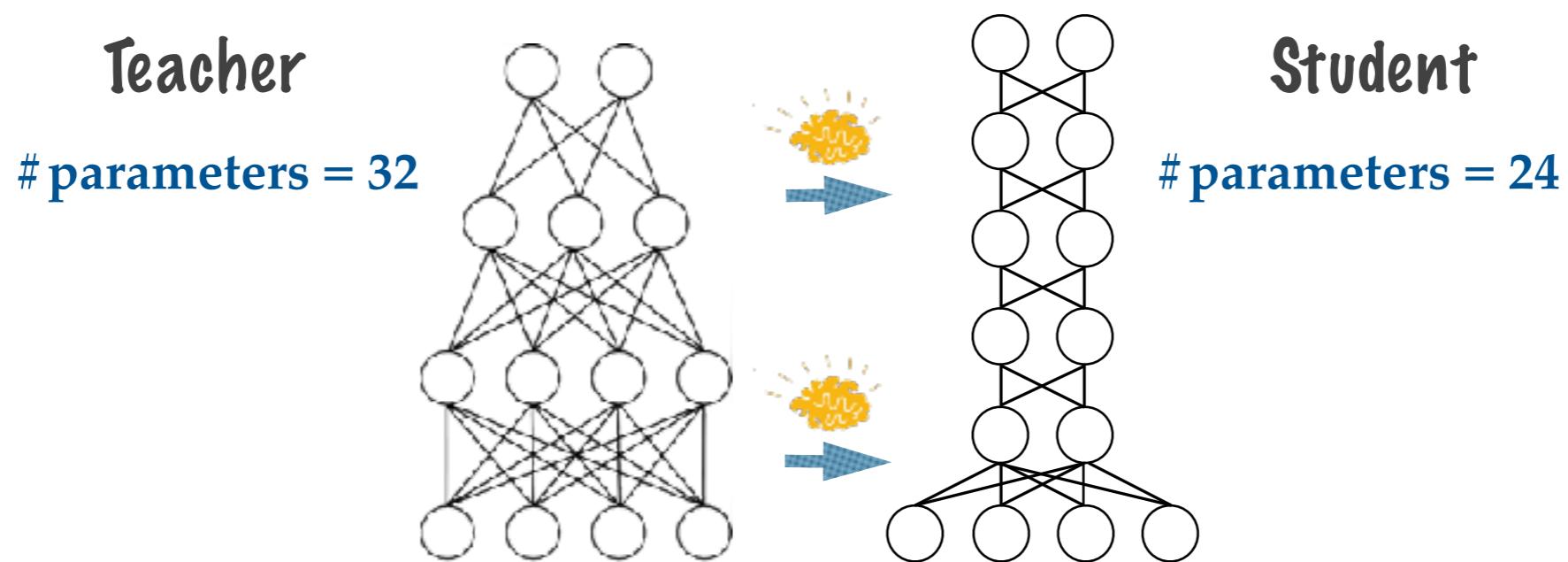
Adding intermediate supervision (discriminative loss):



provides direct and early supervision

# FitNets

Knowledge Transfer:



Trade width for depth to reduce the number of parameters

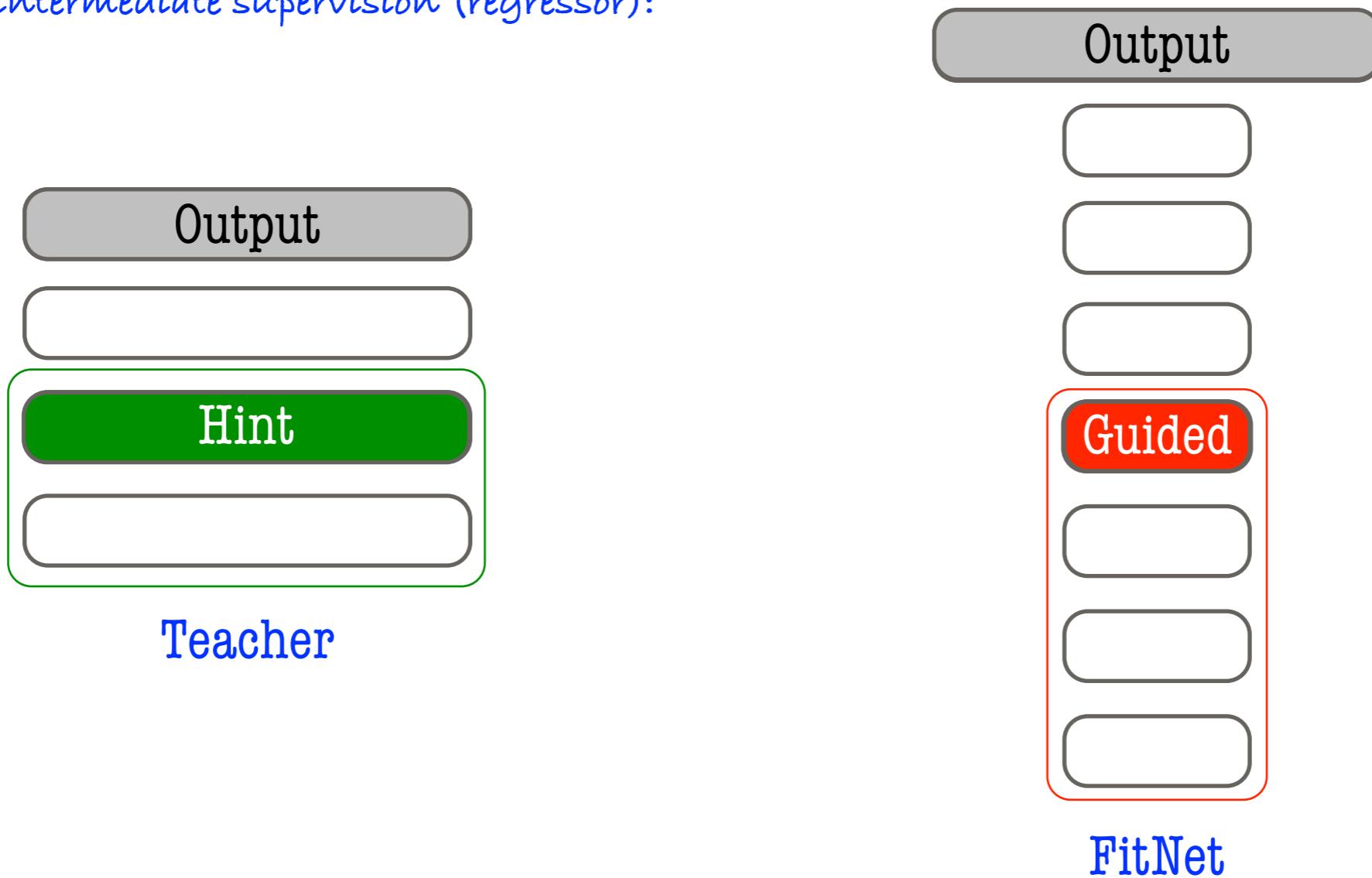
# FitNets

Adding intermediate supervision (regressor):



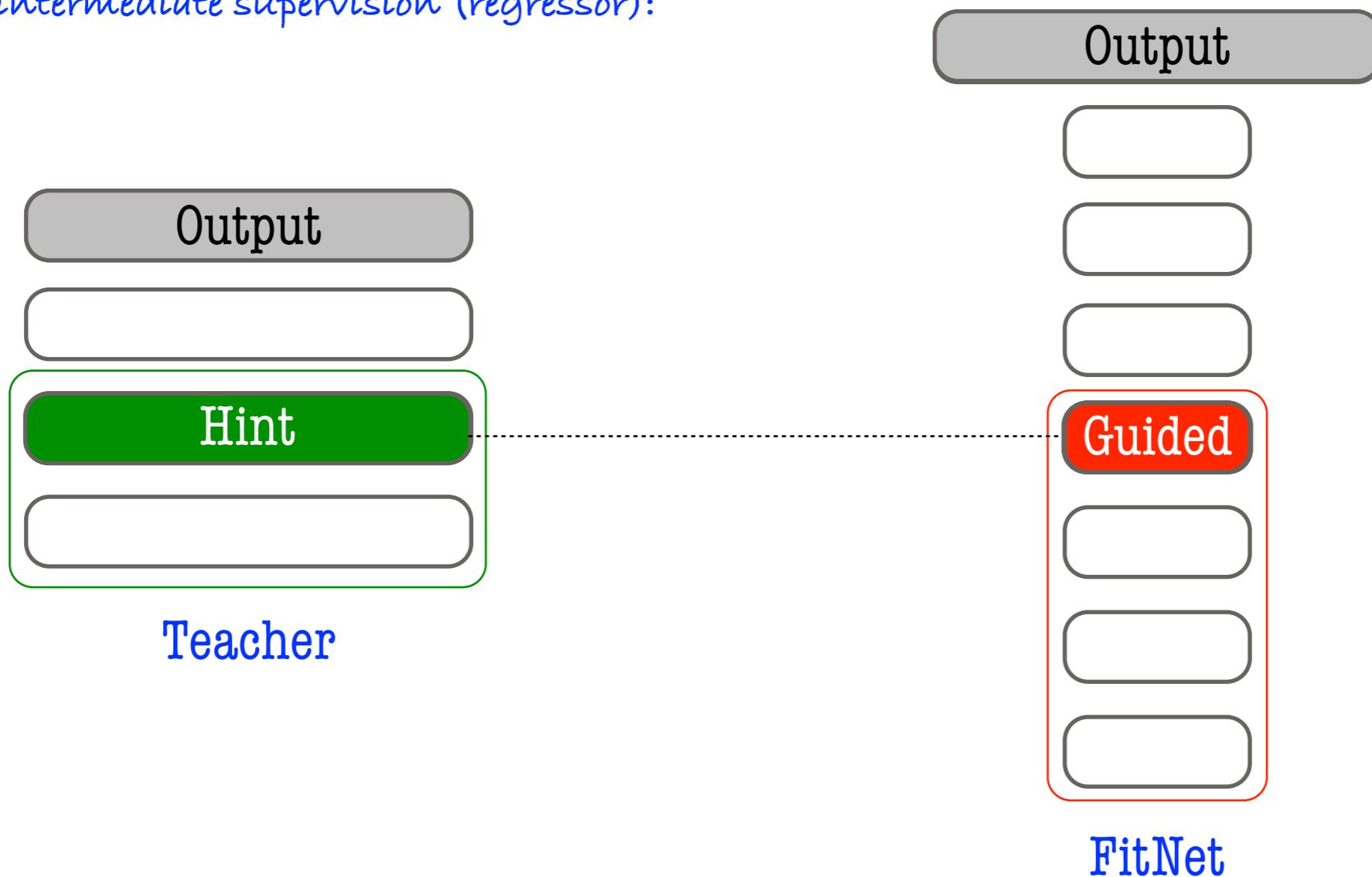
# FitNets

Adding intermediate supervision (regressor):



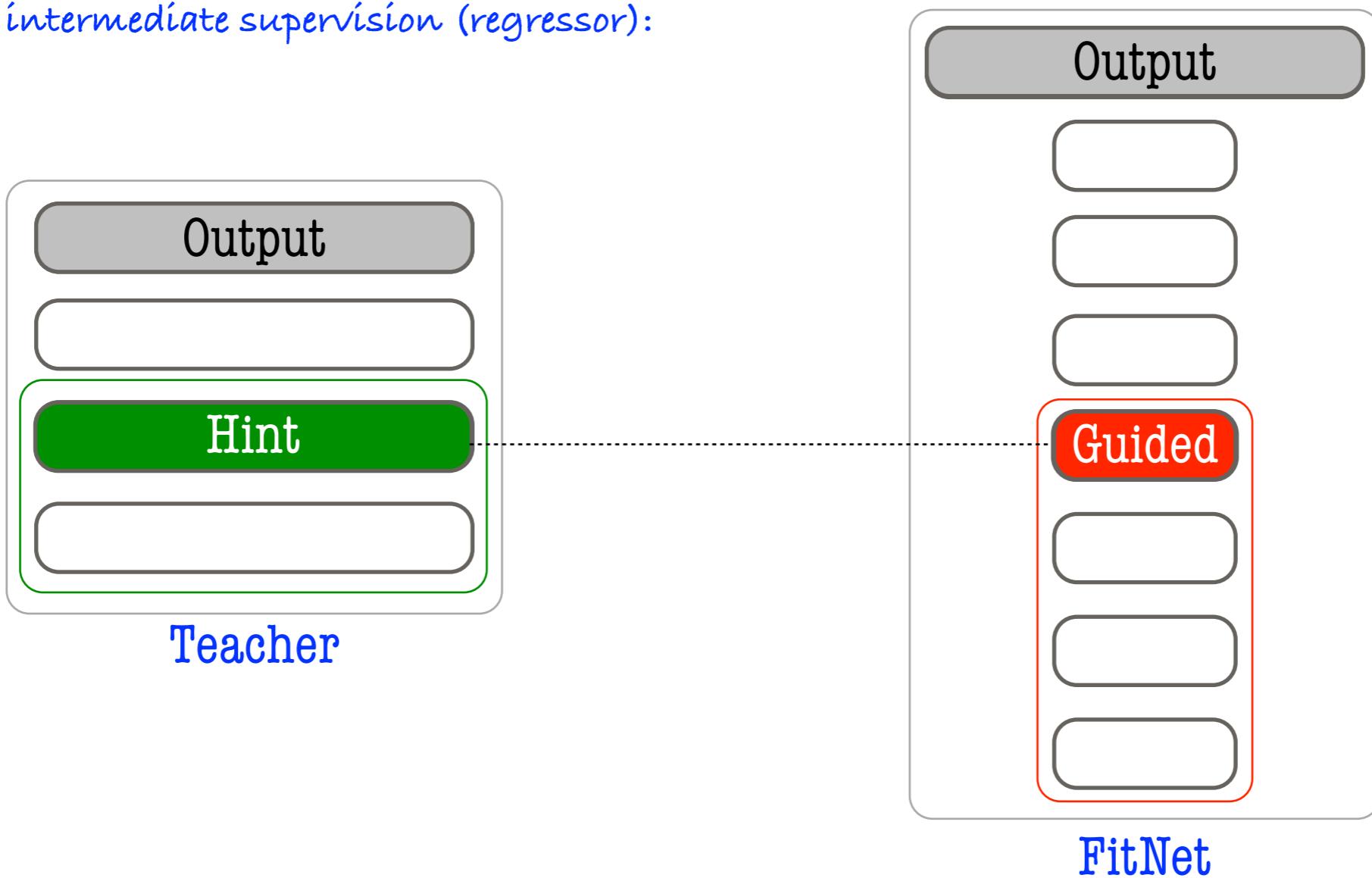
# FitNets

Adding intermediate supervision (regressor):



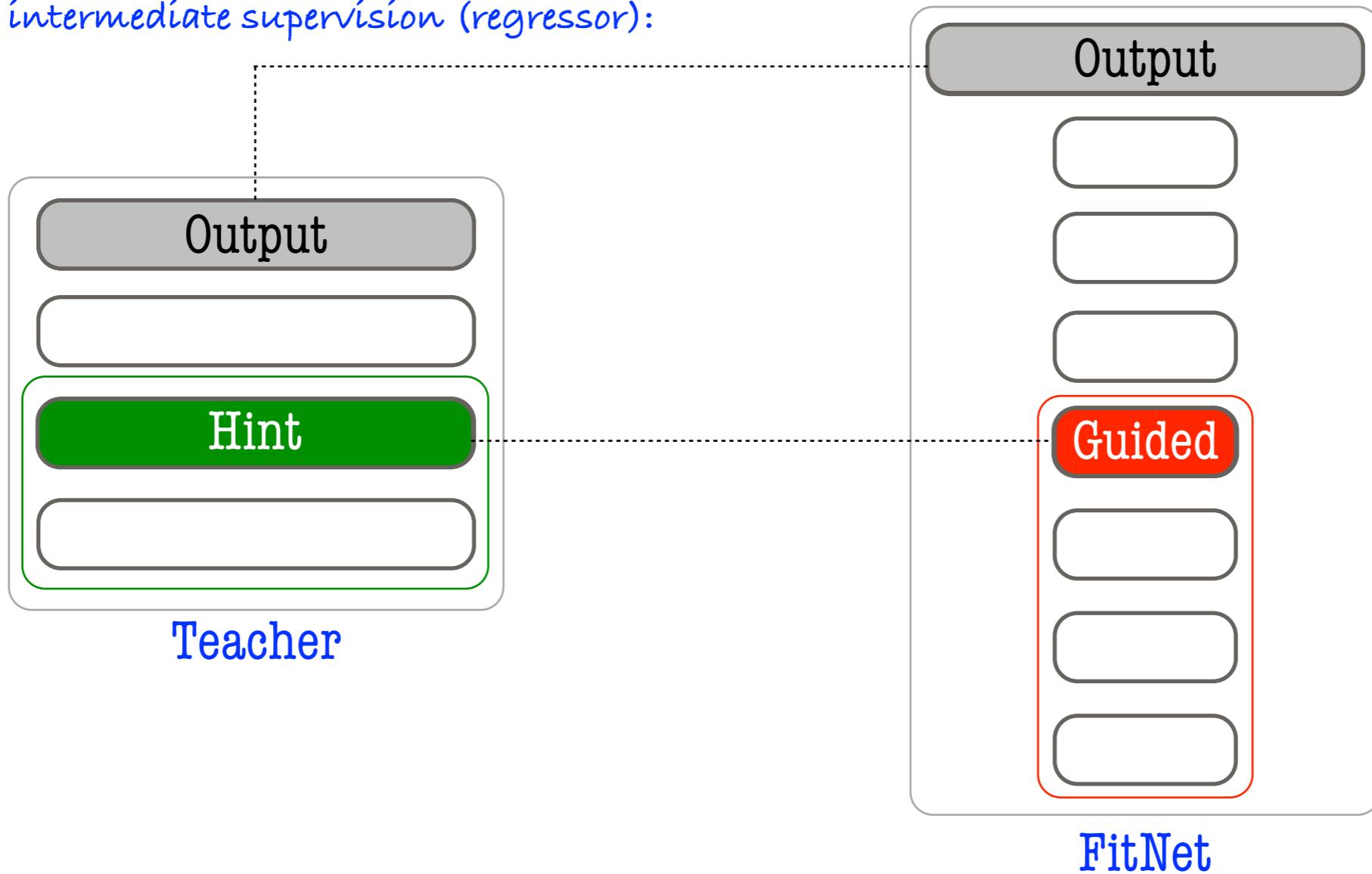
# FitNets

Adding intermediate supervision (regressor):



# FitNets

Adding intermediate supervision (regressor):



# FitNets

Accuracy is retained!! Why?

*Gradient Descent Provably Optimizes Over-parameterized Neural Networks, in ICLR2019*

[Romero et al. 2015]

# FitNets

Accuracy is retained!! Why?

With **enough hidden nodes**, randomly initialized **gradient descent converges** a globally optimal solution with a linear convergence rate for the quadratic loss function.

*Gradient Descent Provably Optimizes Over-parameterized Neural Networks, in ICLR2019*

# FitNets

Accuracy is retained!! Why?

With **enough hidden nodes**, randomly initialized **gradient descent converges** a globally optimal solution with a linear convergence rate for the quadratic loss function.

*Gradient Descent Provably Optimizes Over-parameterized Neural Networks, in ICLR2019*

# FitNets

Accuracy is retained!! Why?

With **enough hidden nodes**, randomly initialized **gradient descent converges** a globally optimal solution with a linear convergence rate for the quadratic loss function.

Take-home message: over-parametrization helps optimization

*Gradient Descent Provably Optimizes Over-parameterized Neural Networks, in ICLR2019*

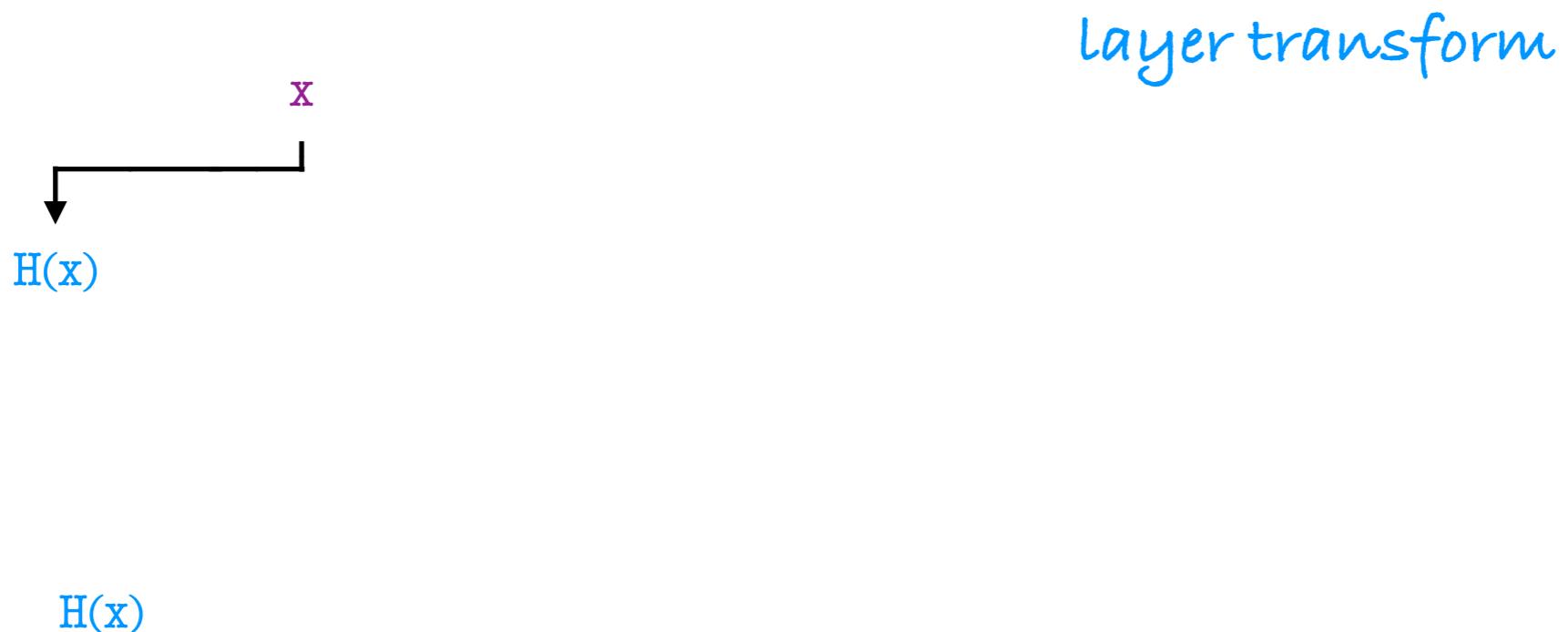
# Highway Networks

New architectural design:

X

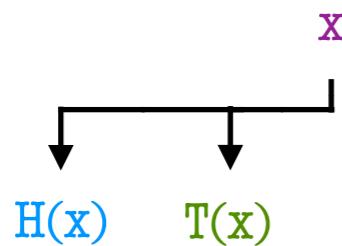
# Highway Networks

New architectural design:



# Highway Networks

New architectural design:

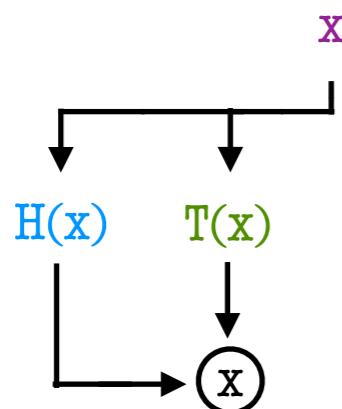


layer transform  
transform gate

$H(x)$      $T(x)$

# Highway Networks

New architectural design:



layer transform

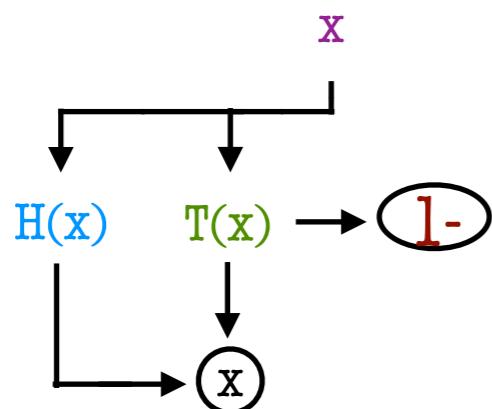
transform gate

how much we transform  $x$

$$H(x) * T(x)$$

# Highway Networks

New architectural design:



layer transform

transform gate

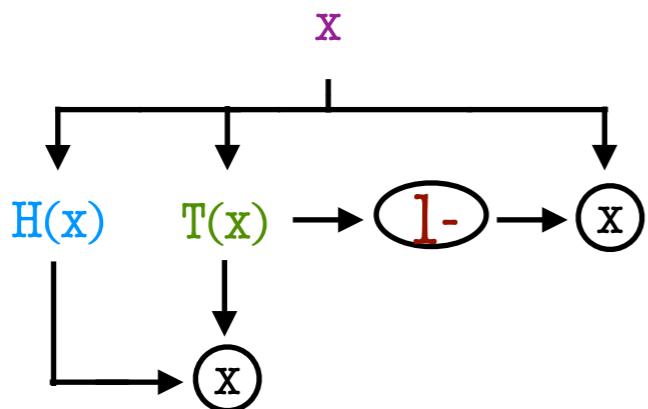
how much we transform  $x$

carry gate

$$H(x) * T(x) \quad (1-T(x))$$

# Highway Networks

New architectural design:



layer transform

transform gate

how much we transform  $x$

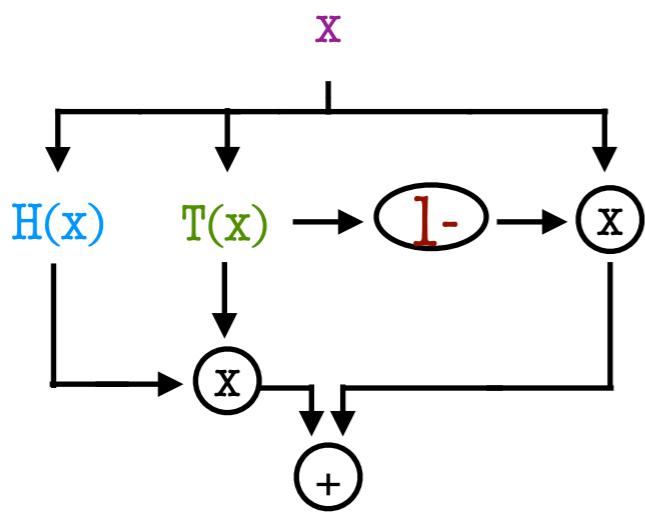
carry gate

how much we preserve  $x$

$$H(x) * T(x) \quad x * (1-T(x))$$

# Highway Networks

New architectural design:



layer transform

transform gate

how much we transform  $x$

carry gate

how much we preserve  $x$

output

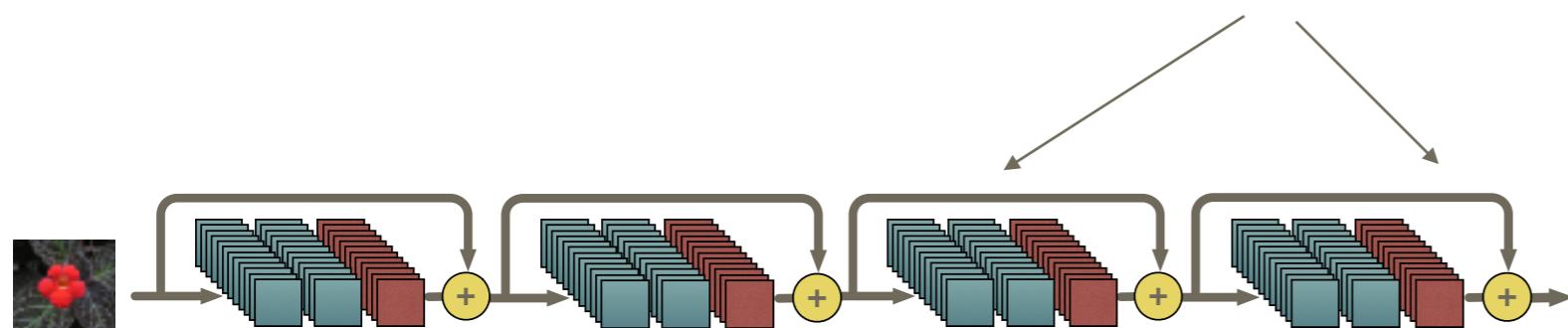
$$H(x) * T(x) + x * (1-T(x))$$

initialization biased towards carry behavior

# Residual Networks

New architectural design:

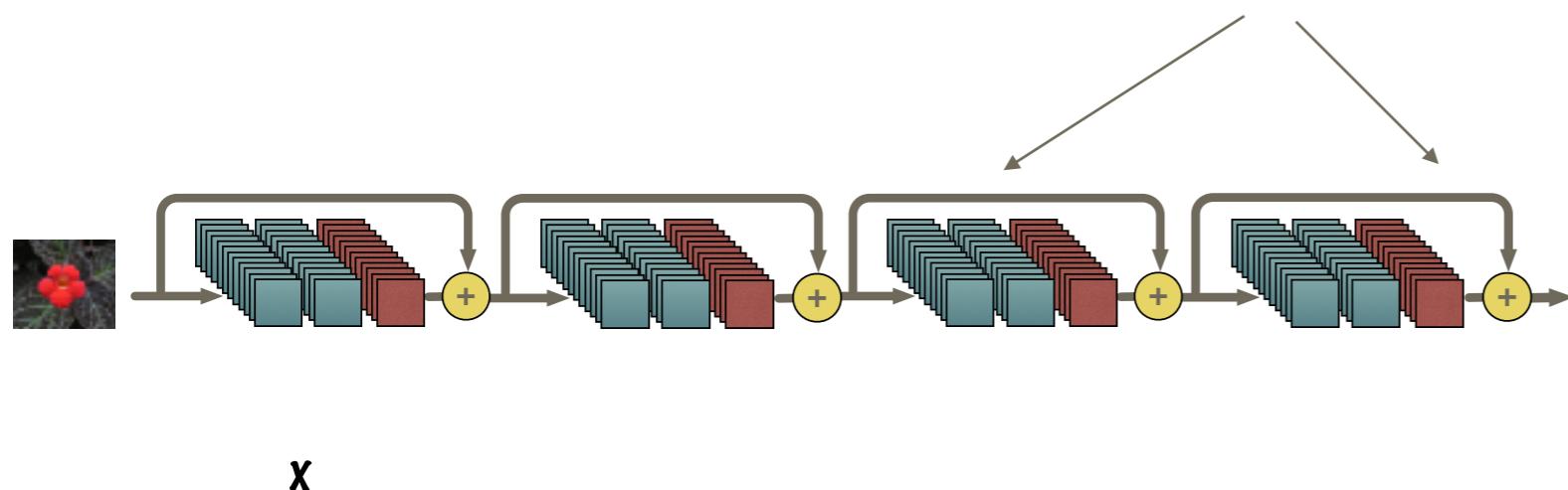
Residual skip-connection



# Residual Networks

New architectural design:

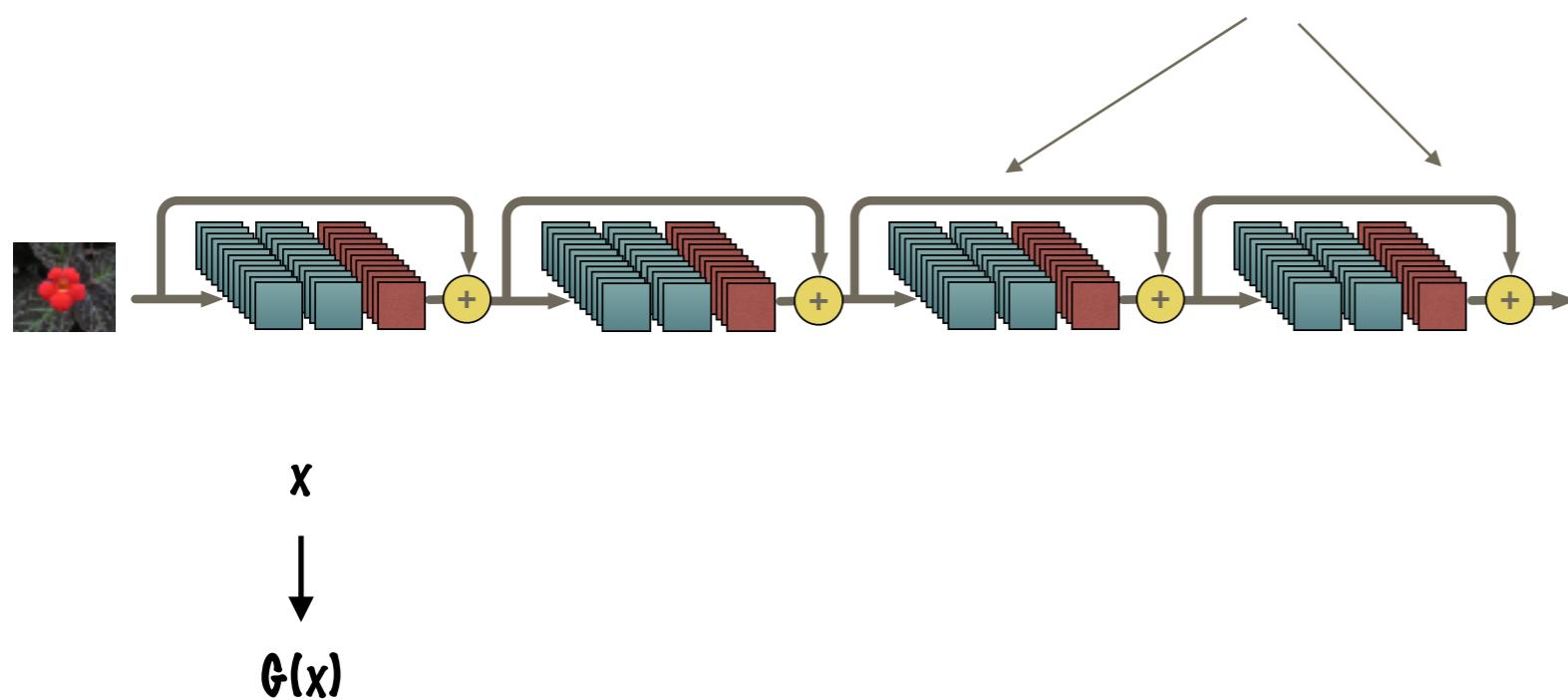
Residual skip-connection



# Residual Networks

New architectural design:

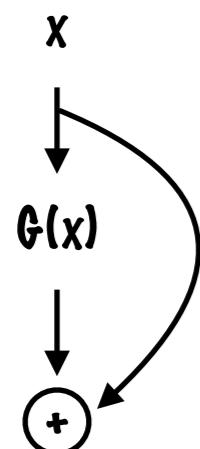
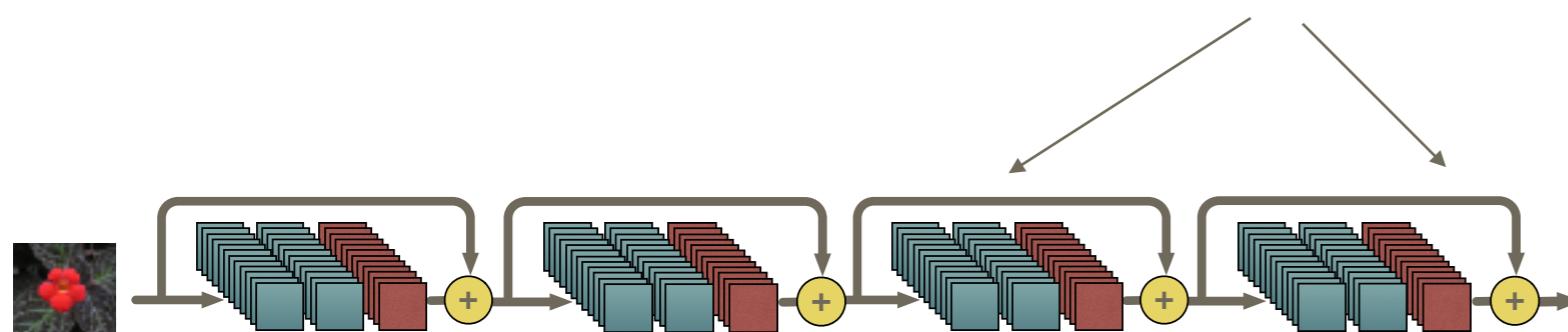
Residual skip-connection



# Residual Networks

New architectural design:

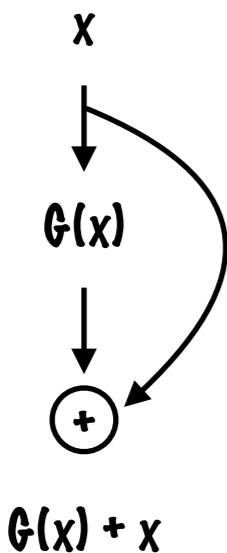
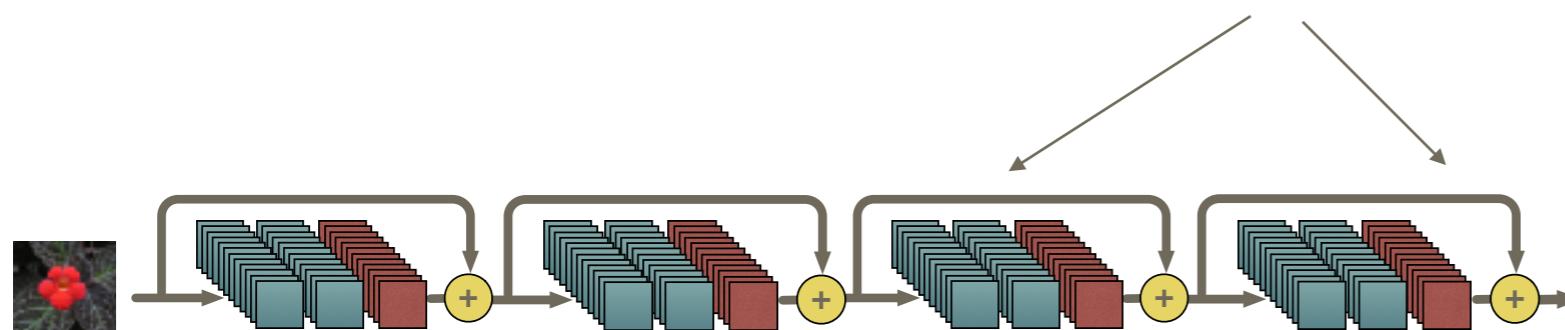
Residual skip-connection



# Residual Networks

New architectural design:

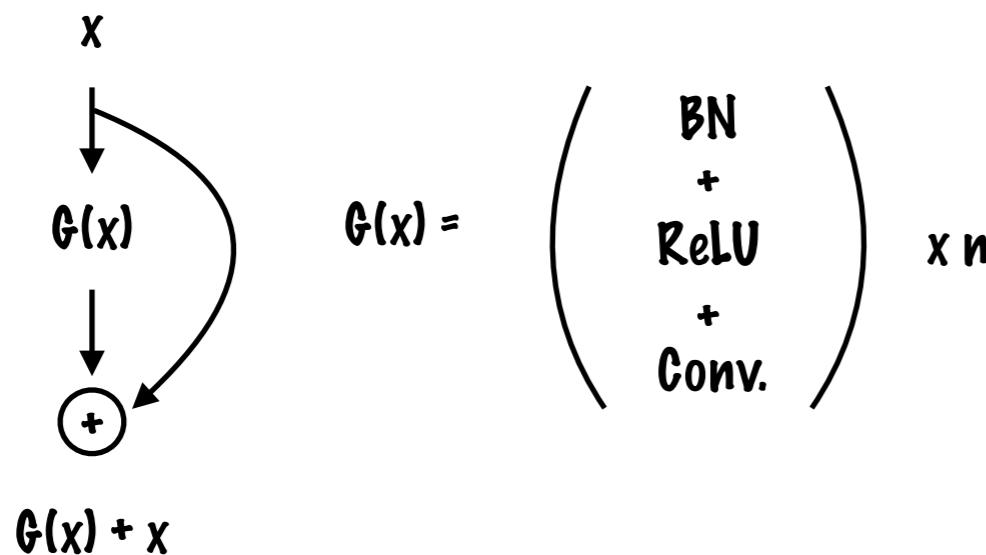
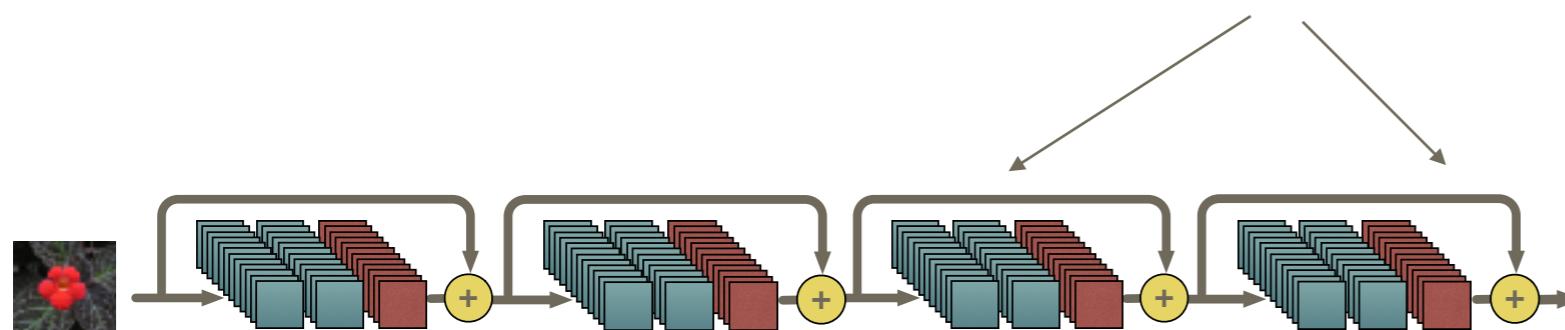
Residual skip-connection



# Residual Networks

New architectural design:

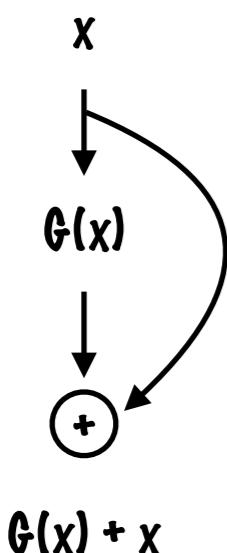
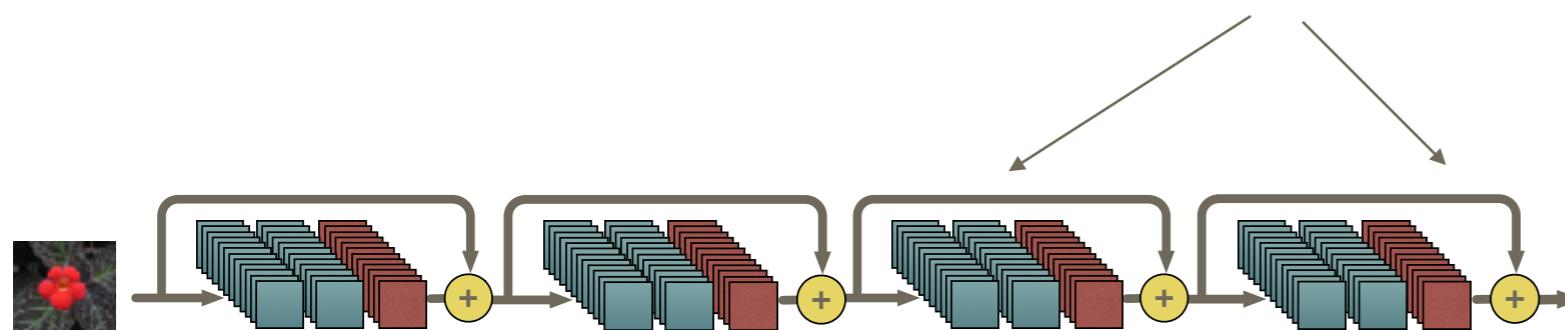
Residual skip-connection



# Residual Networks

New architectural design:

Residual skip-connection



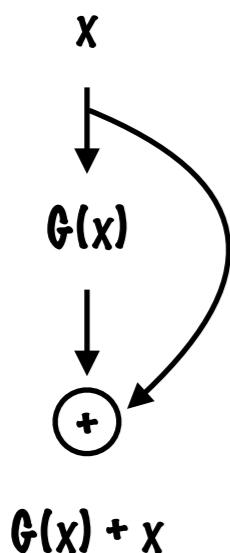
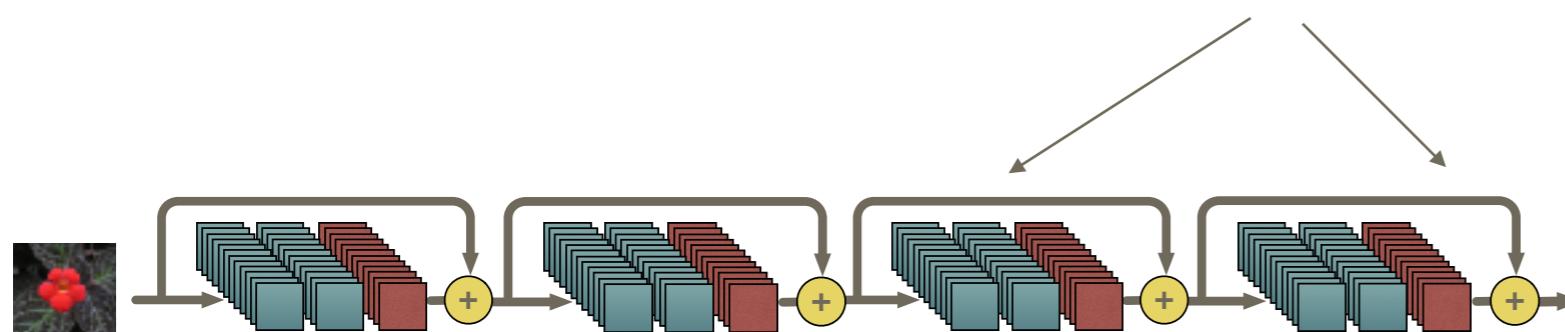
$$G(x) = \left( \begin{array}{c} \text{BN} \\ + \\ \text{ReLU} \\ + \\ \text{Conv.} \end{array} \right) x n$$

\* Compute a small change to the input to get a slightly altered representation

# Residual Networks

New architectural design:

Residual skip-connection



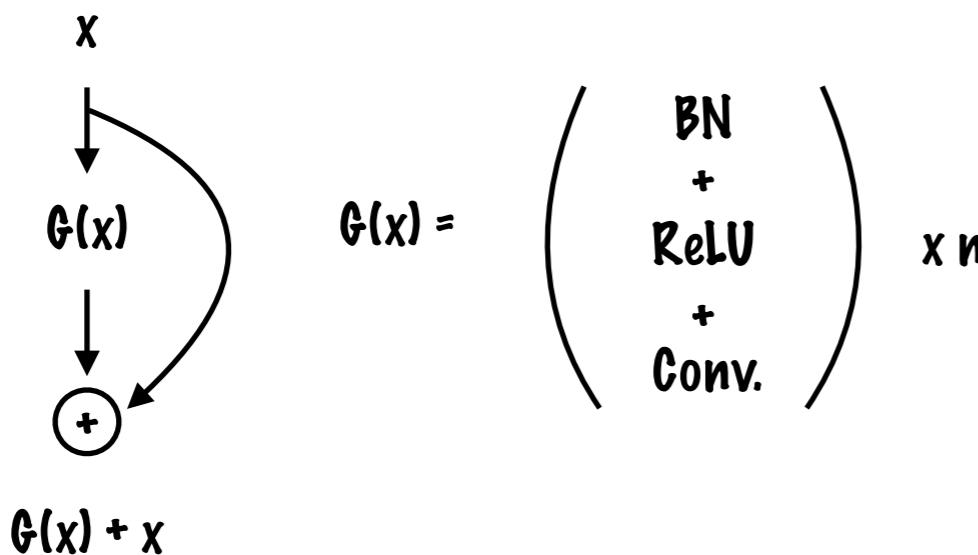
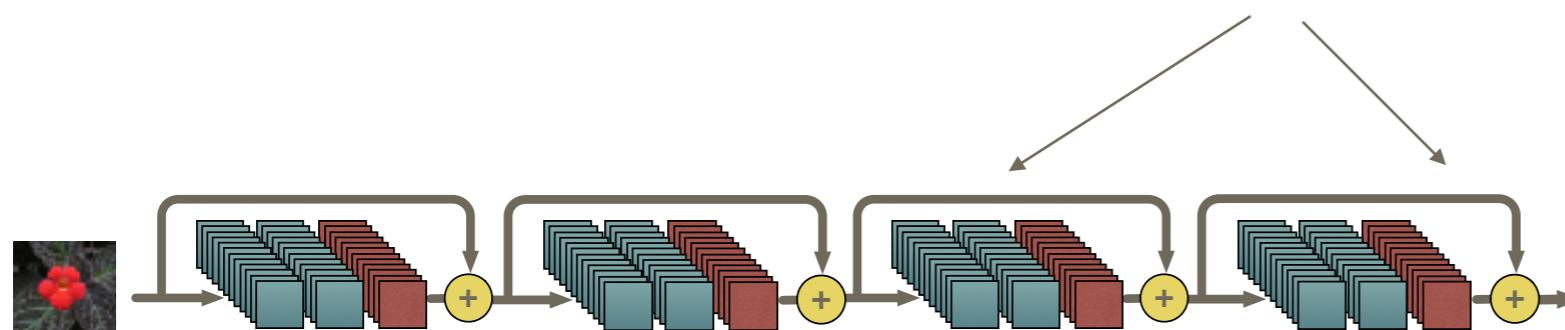
$$G(x) = \left( \begin{array}{c} \text{BN} \\ + \\ \text{ReLU} \\ + \\ \text{Conv.} \end{array} \right) x$$

- \* Compute a small change to the input to get a slightly altered representation
- \* Gradient flows easily to the bottom layers of the network

# Residual Networks

New architectural design:

Residual skip-connection

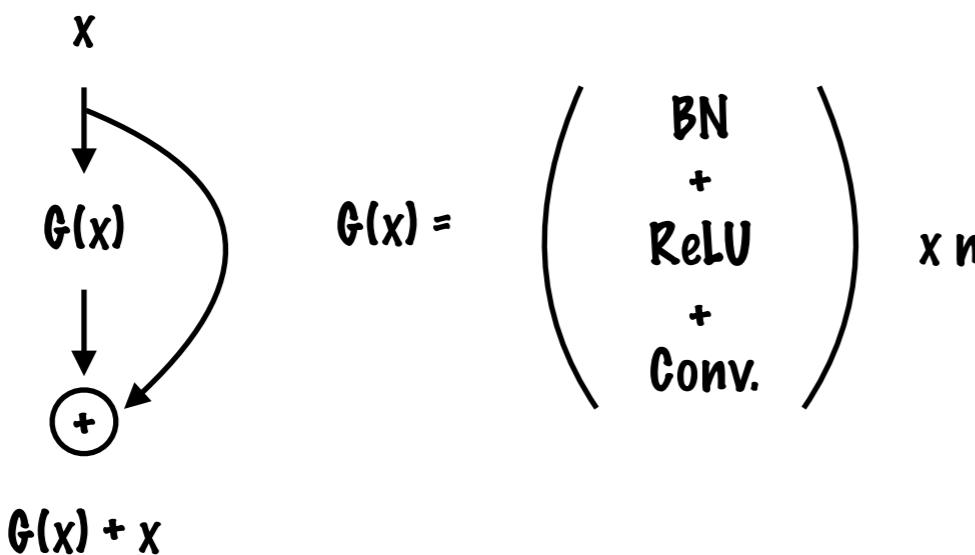
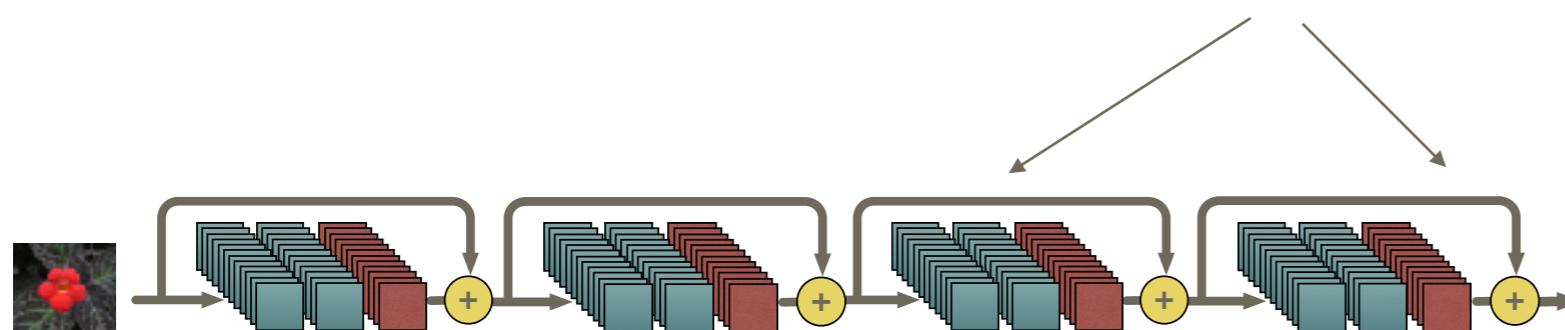


- \* Compute a small change to the input to get a slightly altered representation
- \* Gradient flows easily to the bottom layers of the network
- \* Ultra deep architecture (152 layers)

# Residual Networks

New architectural design:

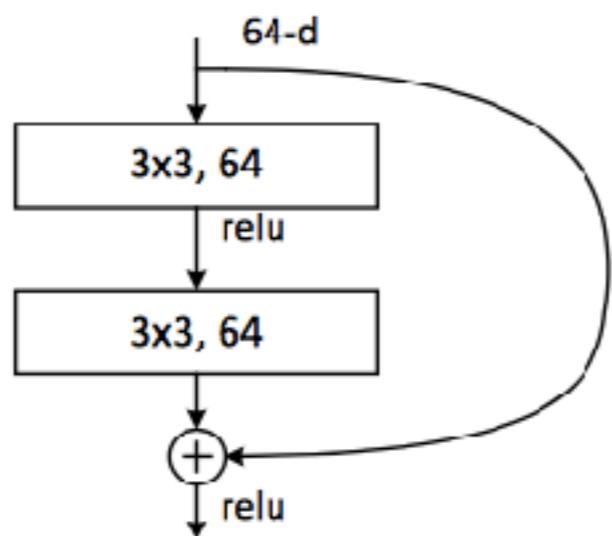
Residual skip-connection



- \* 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)
- \* Winner of ILSVRC 2015

# Residual Networks

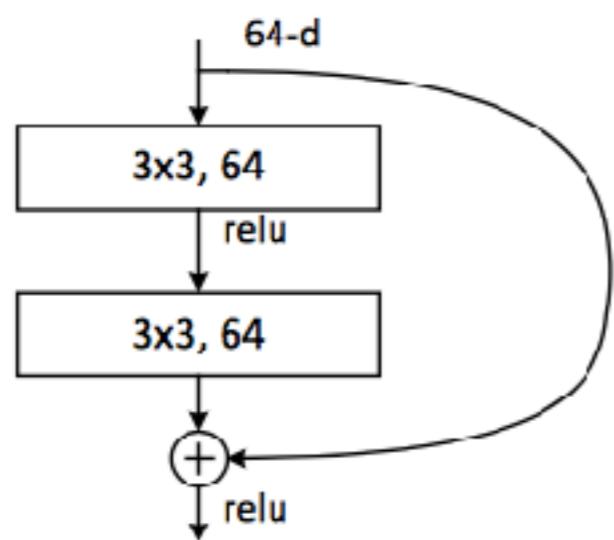
Residual blocks:



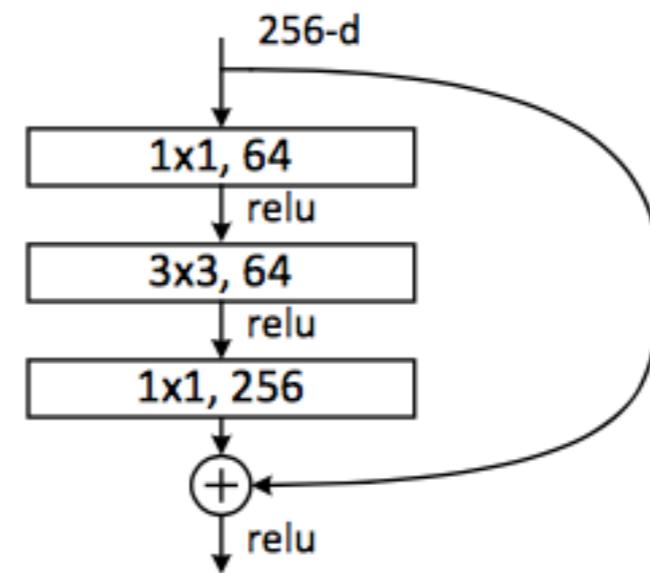
Basic Block

# Residual Networks

Residual blocks:



Basic Block



Bottleneck block

# **CNN vs Highway vs ResNet**

# CNN vs Highway vs ResNet

## CNNs

$$x \downarrow F(x)$$

$$F(x) = \left( \begin{array}{c} \text{Conv.} \\ + \\ \text{ReLU} \end{array} \right) x n$$

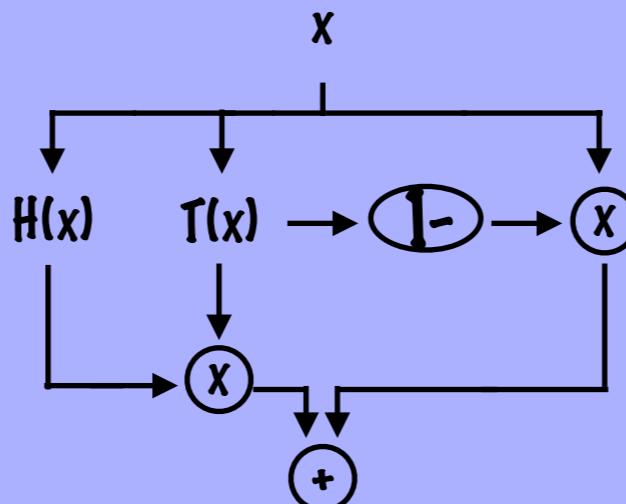
# CNN vs Highway vs ResNet

## CNNs

$$x \downarrow \\ F(x)$$

$$F(x) = \left( \begin{array}{c} \text{Conv.} \\ + \\ \text{ReLU} \end{array} \right) x n$$

## Highway Nets



$$H(x) * T(x) + x * (1-T(x))$$

$$H(x) = \left( \begin{array}{c} \text{Conv.} \\ + \\ \text{ReLU} \end{array} \right)$$

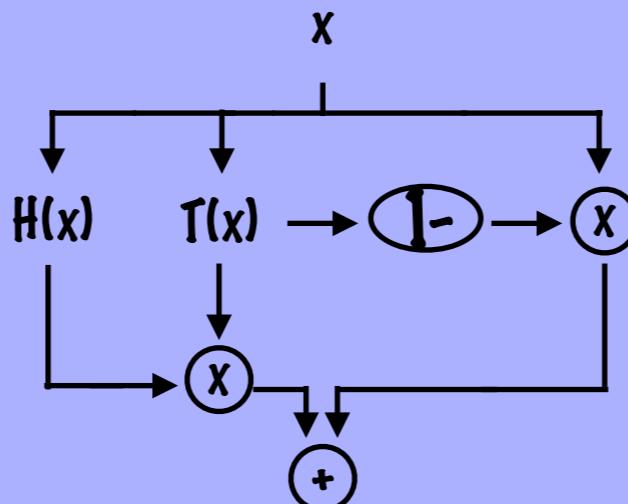
# CNN vs Highway vs ResNet

## CNNs

$$x \downarrow \\ F(x)$$

$$F(x) = \left( \begin{array}{c} \text{Conv.} \\ + \\ \text{ReLU} \end{array} \right) x n$$

## Highway Nets



$$H(x) * T(x) + x * (1-T(x))$$

$$H(x) = \left( \begin{array}{c} \text{Conv.} \\ + \\ \text{ReLU} \\ \text{or sigmoid} \end{array} \right)$$

or  $T(x)$

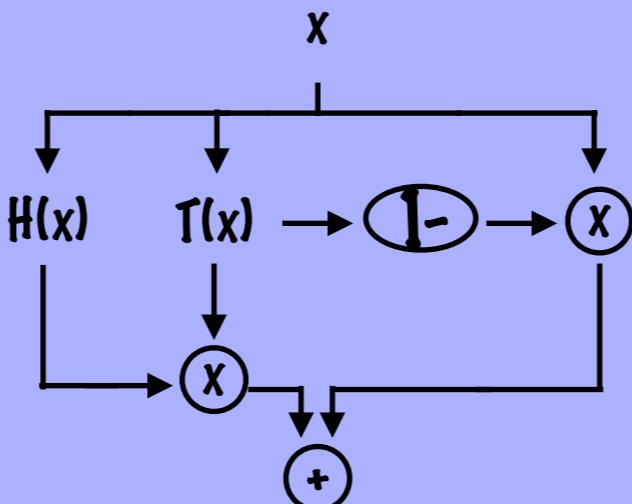
# CNN vs Highway vs ResNet

## CNNs

$$x \downarrow \\ F(x)$$

$$F(x) = \left( \begin{array}{c} \text{Conv.} \\ + \\ \text{ReLU} \end{array} \right) x n$$

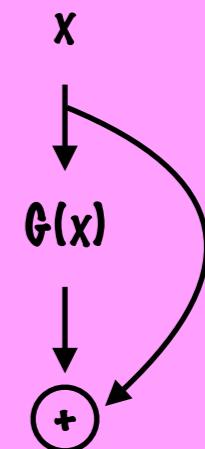
## Highway Nets



$$H(x) * T(x) + x * (1-T(x))$$

$$H(x) = \left( \begin{array}{c} \text{Conv.} \\ + \\ \text{ReLU} \\ \text{or sigmoid} \end{array} \right)$$

## ResNets

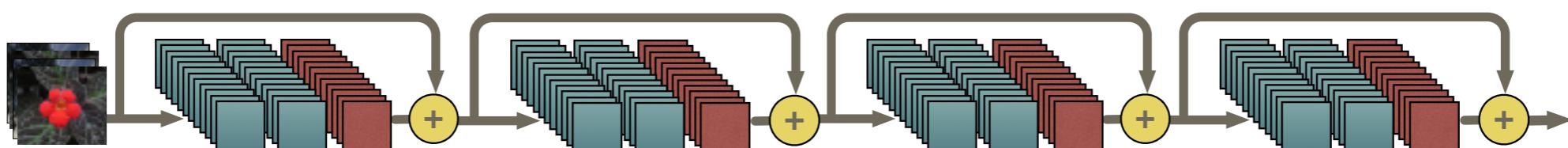


$$G(x) + x$$

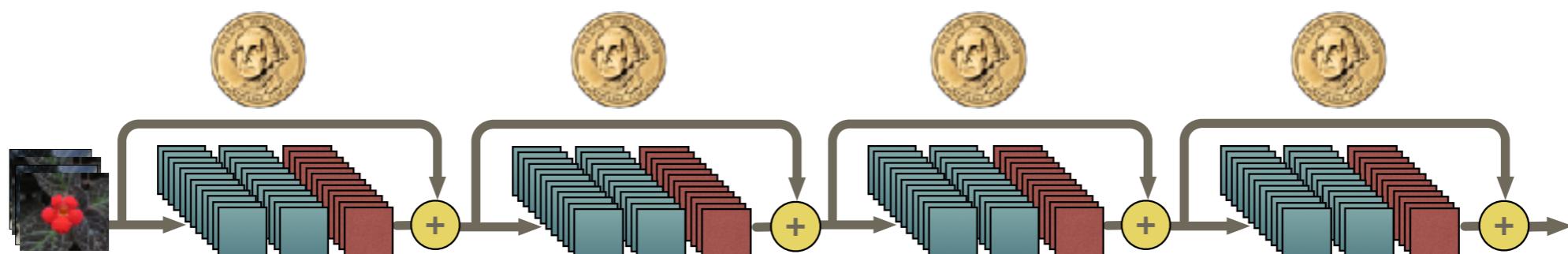
$$G(x) = \left( \begin{array}{c} \text{BN} \\ + \\ \text{ReLU} \\ + \\ \text{Conv.} \end{array} \right) x n$$

**2016 - 2017:  
Stochastic depth, DenseNets,  
ResNeXt, SE Networks**

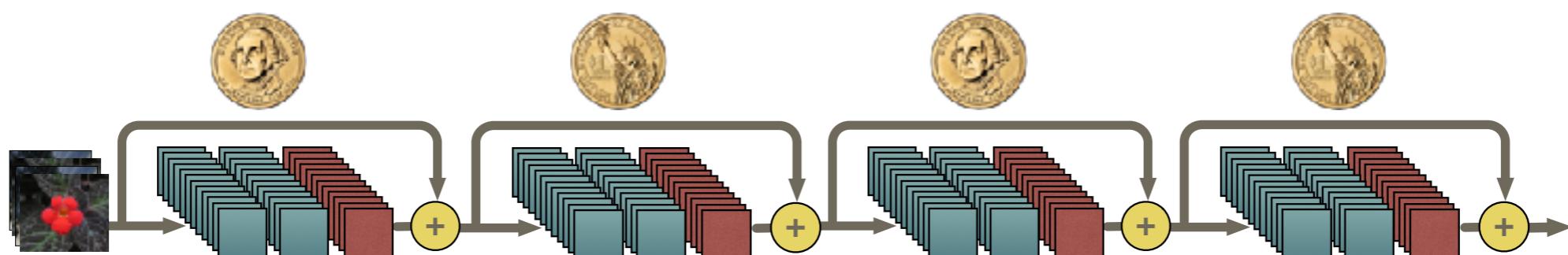
# Stochastic Depth



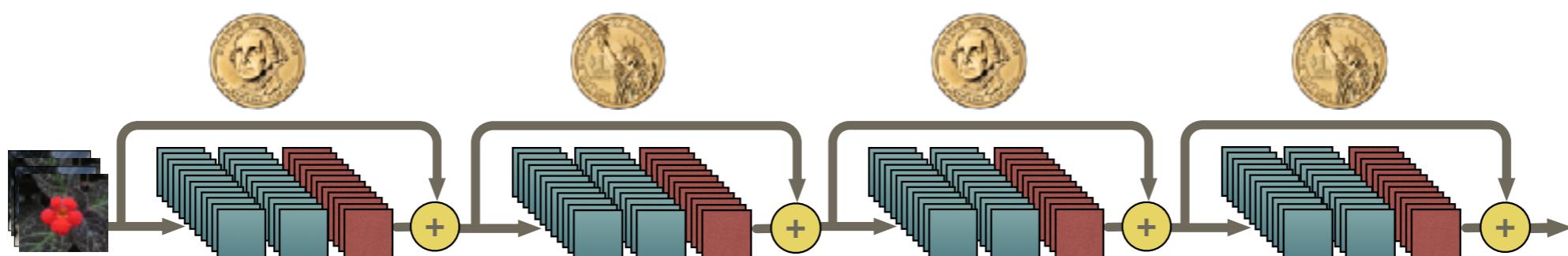
# Stochastic Depth



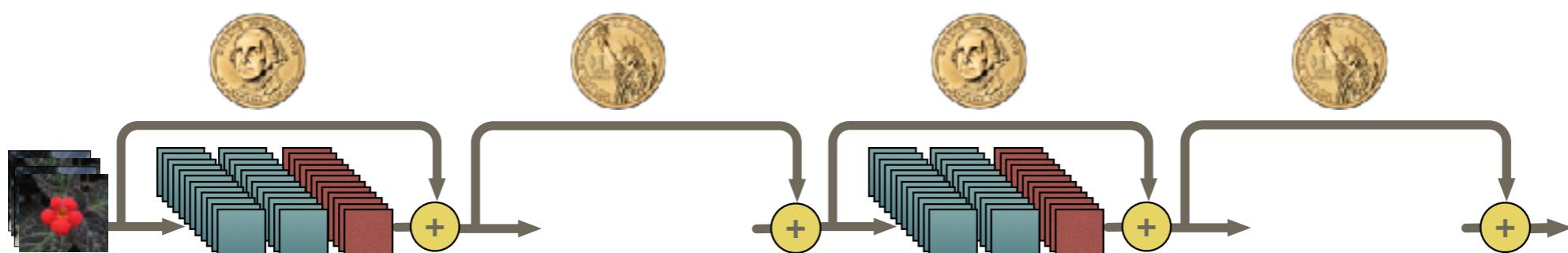
# Stochastic Depth



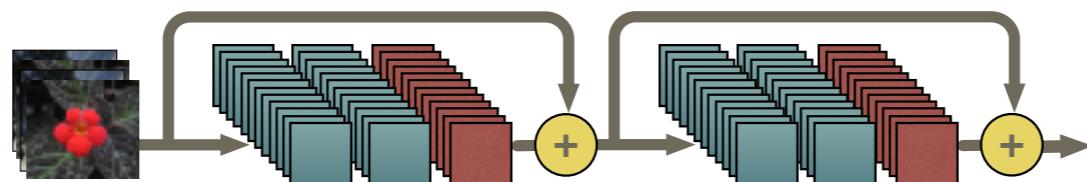
# Stochastic Depth



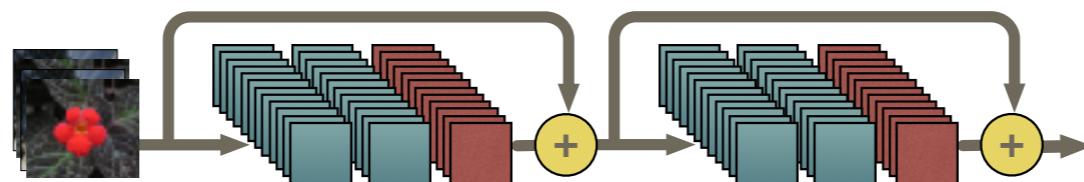
# Stochastic Depth



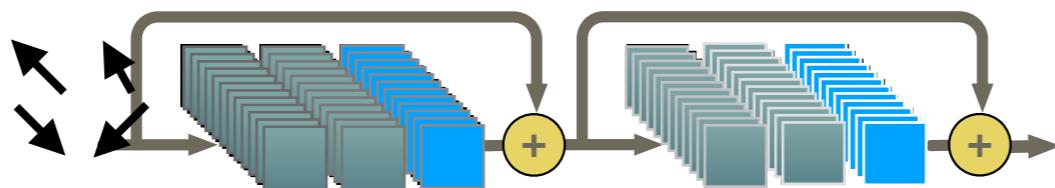
# Random Depth



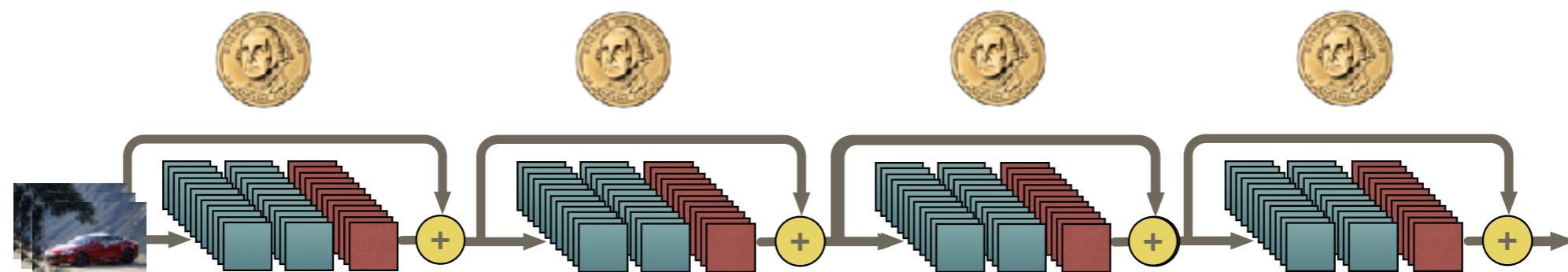
# Random Depth



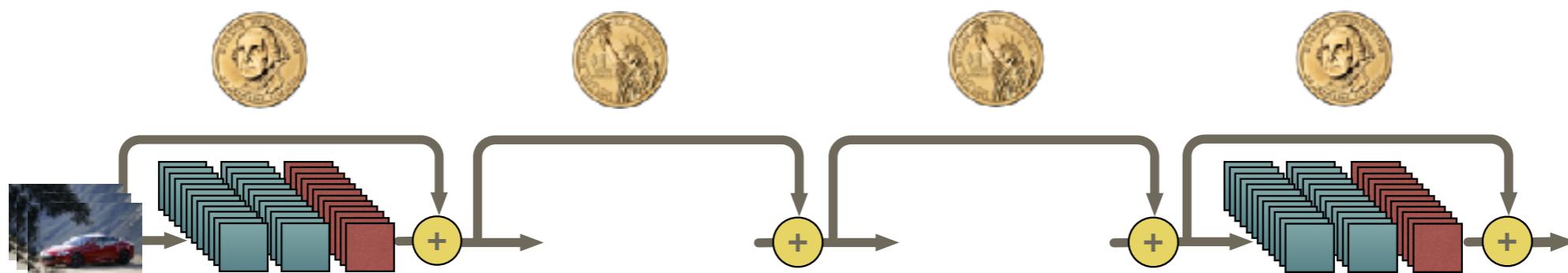
# Random Depth



# Stochastic Depth



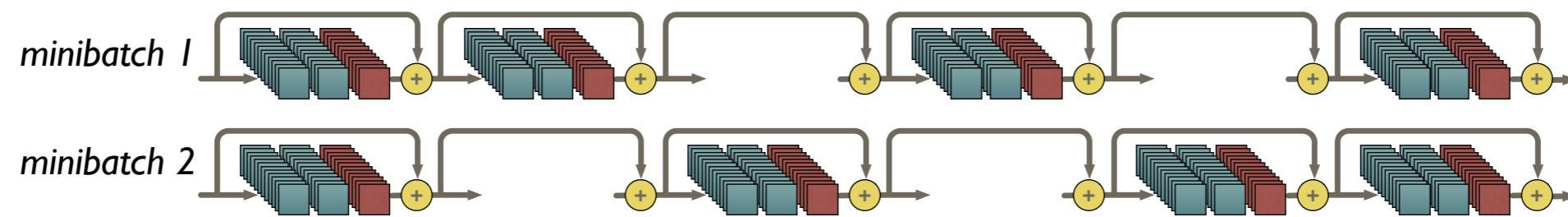
# Stochastic Depth



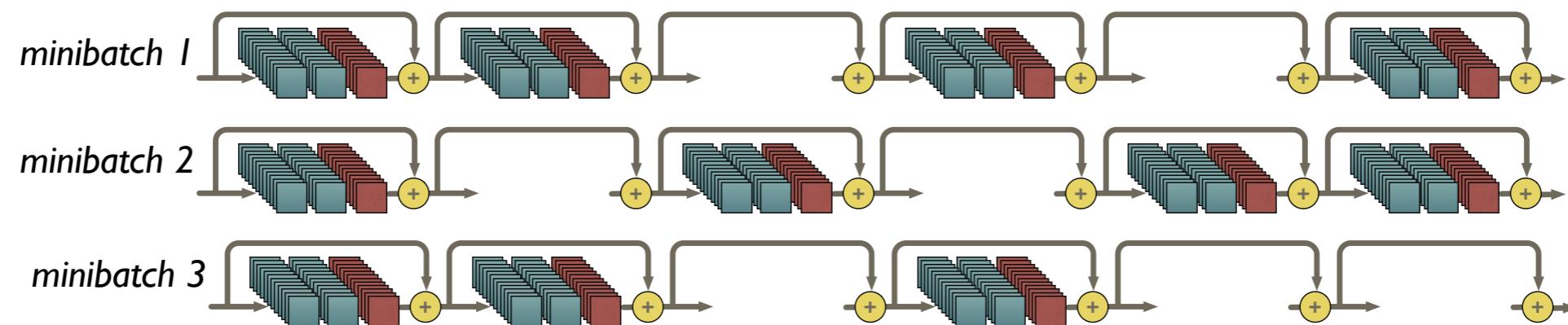
# Stochastic Depth



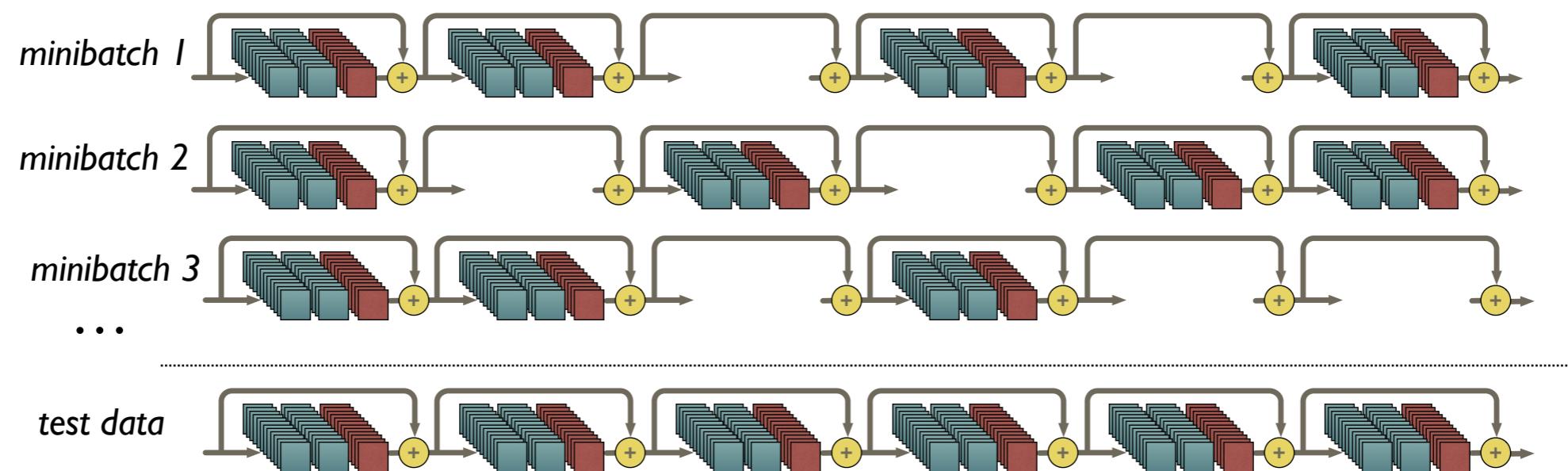
# Stochastic Depth



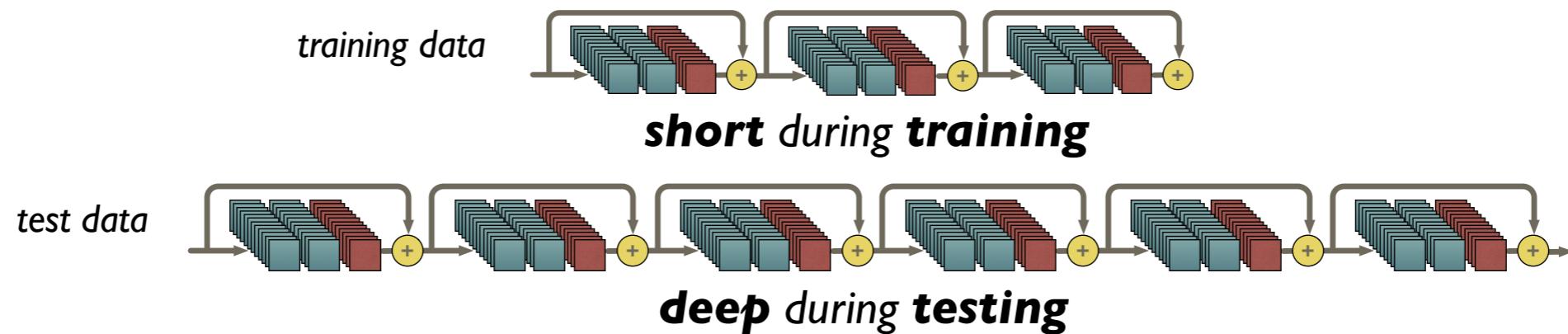
# Stochastic Depth



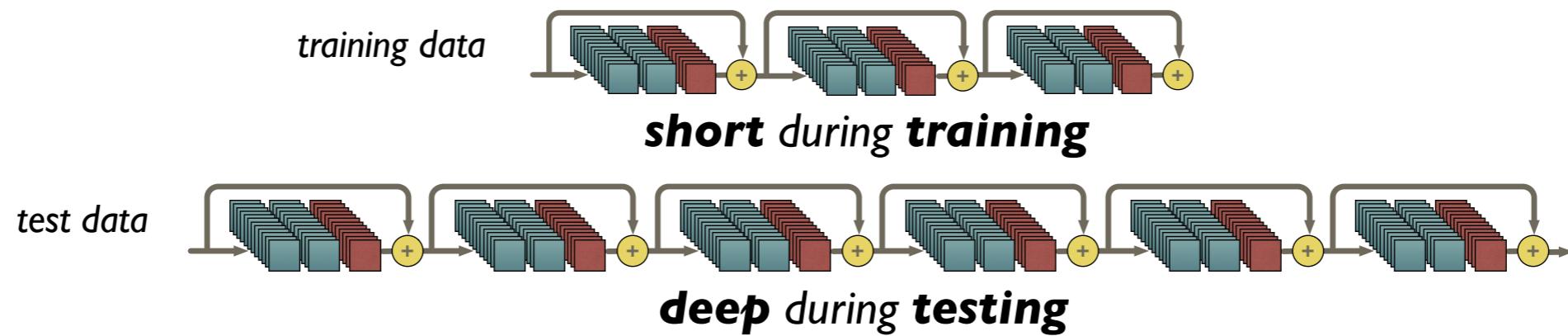
# Stochastic Depth



# Stochastic Depth

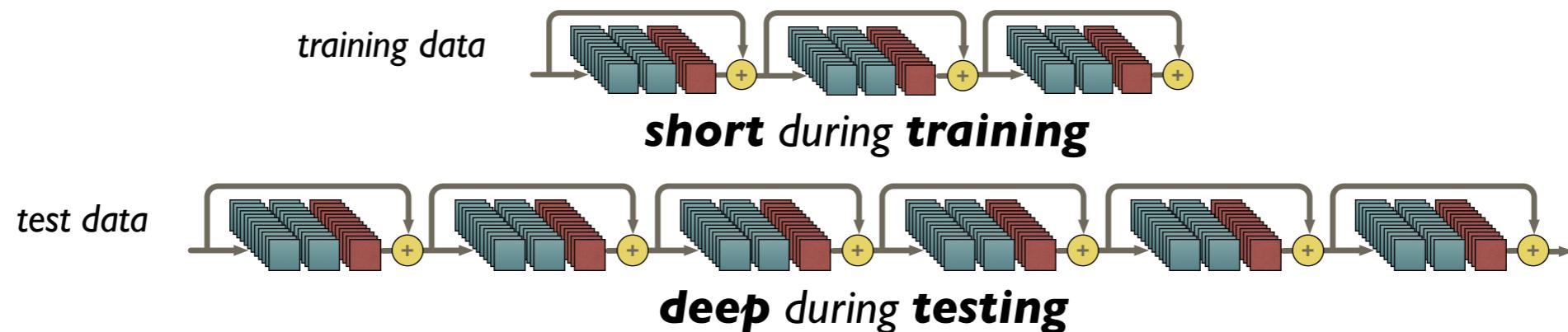


# Stochastic Depth



implicit ensemble of  $2^L$  models

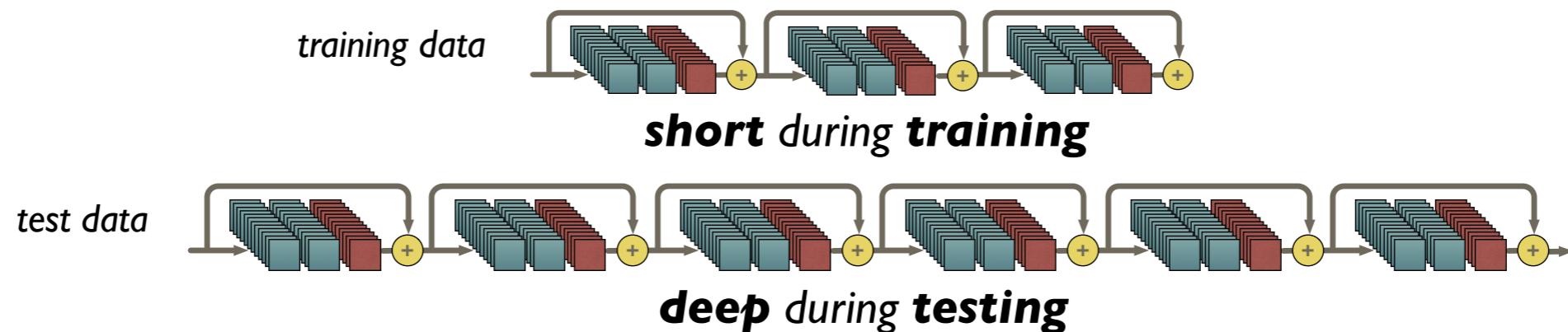
# Stochastic Depth



implicit ensemble of  $2^L$  models

Improved information flow

# Stochastic Depth

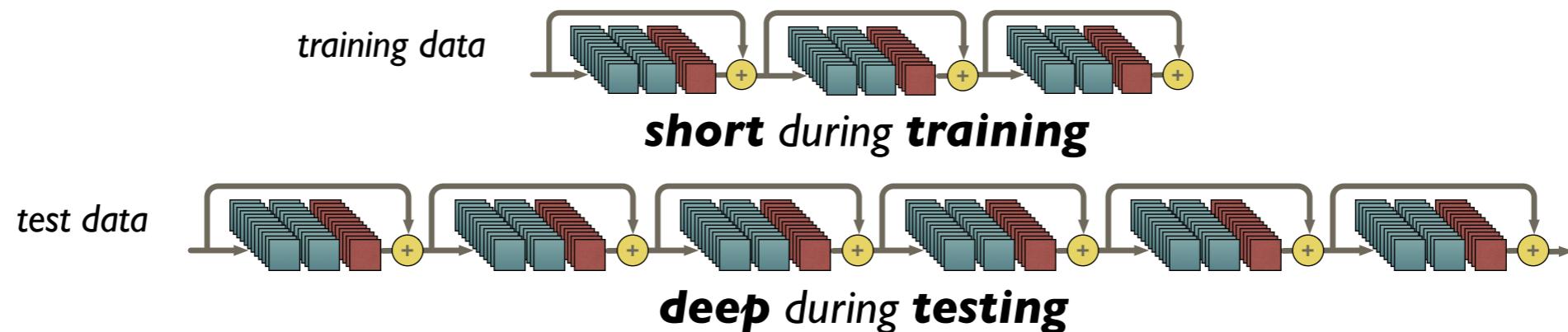


implicit ensemble of  $2^L$  models

Improved information flow

25% speedup during training

# Stochastic Depth



implicit ensemble of  $2^L$  models

Improved information flow

25% speedup during training

lower error



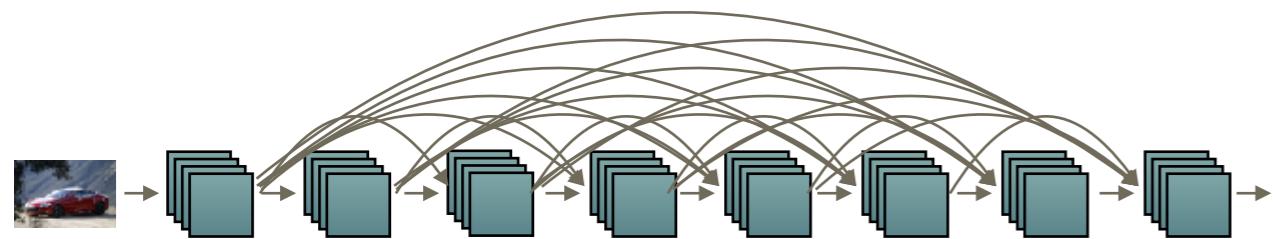
*Slide courtesy of Kilian Weinberger*



*Slide courtesy of Kilian Weinberger*

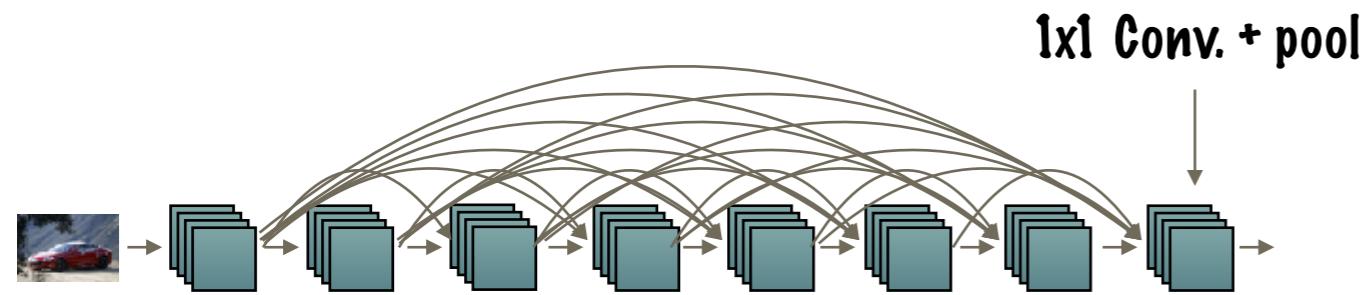
# DenseNet

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



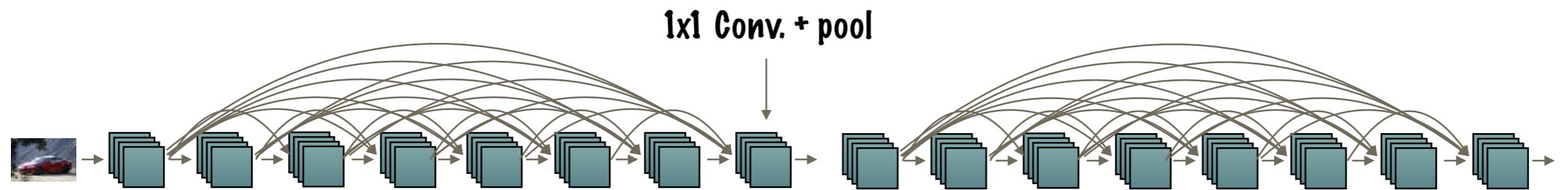
# DenseNet

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



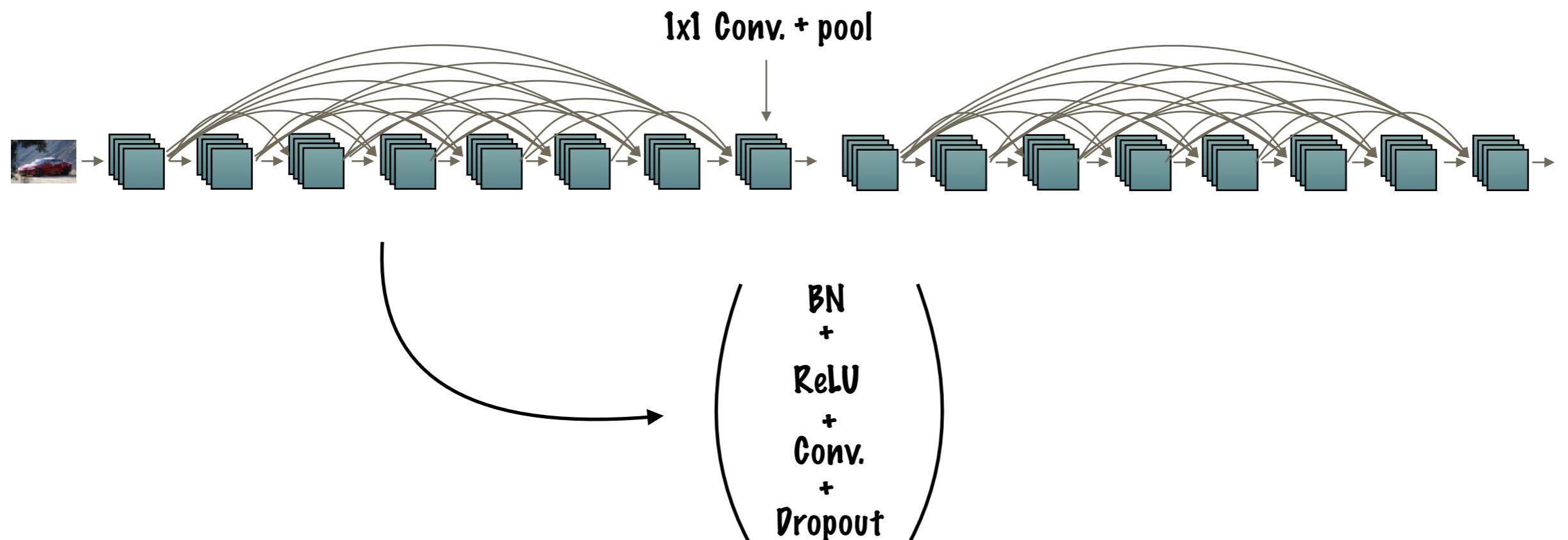
# DenseNet

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

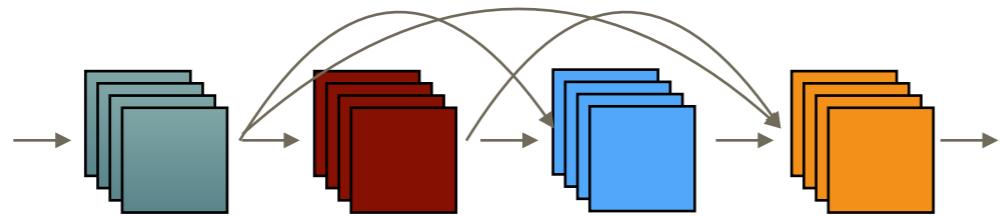


# DenseNet

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



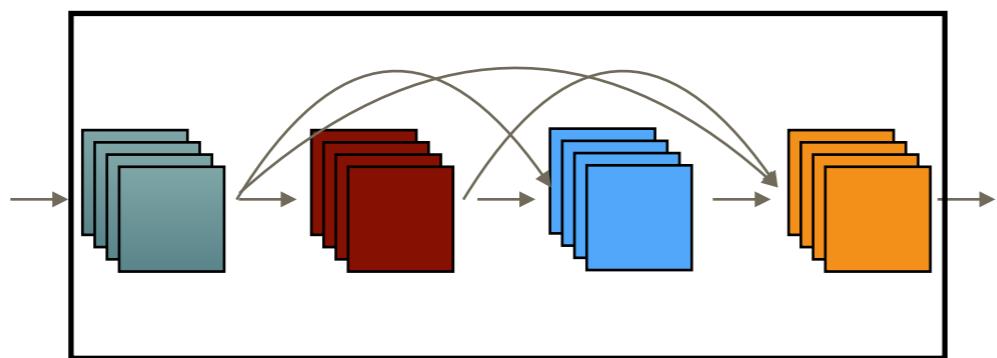
# DenseNet



[Huang et al. 2016]

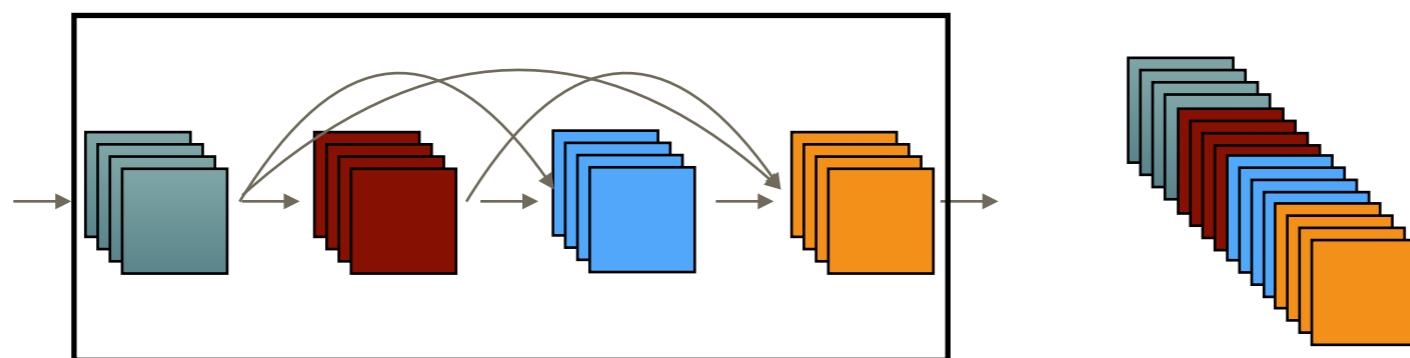
# DenseNet

Dense Block (4 layers, growth rate 4)



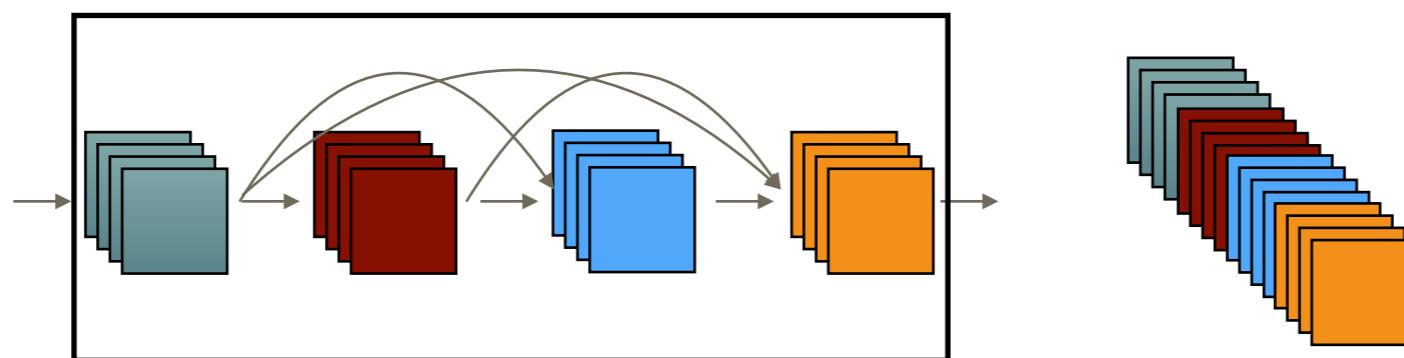
# DenseNet

Dense Block (4 layers, growth rate 4)



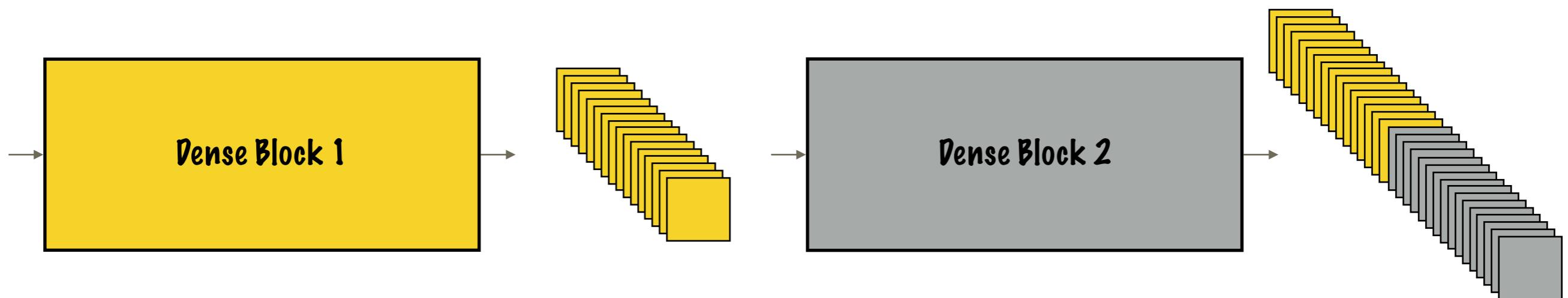
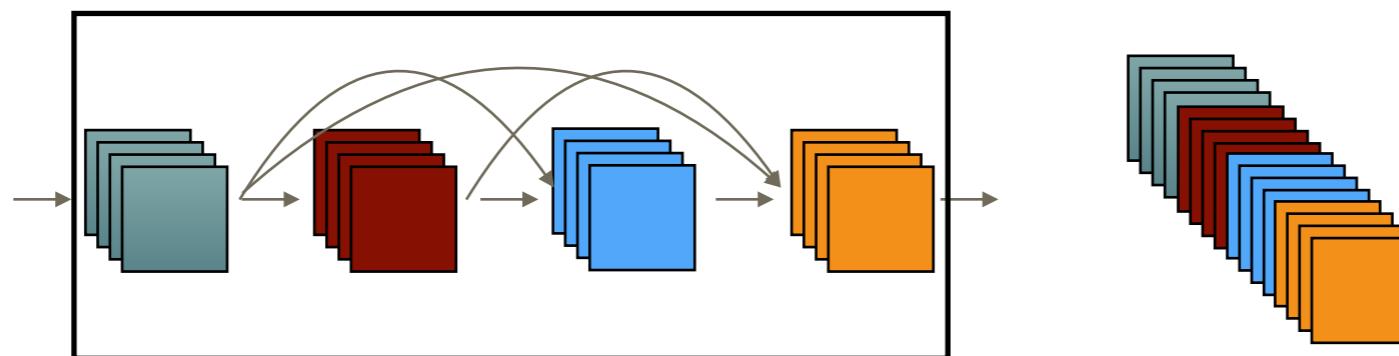
# DenseNet

Dense Block (4 layers, growth rate 4)

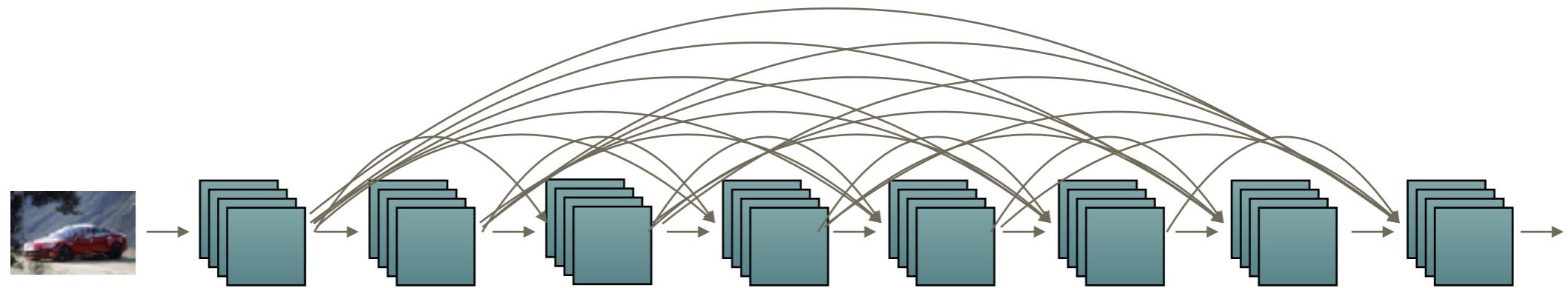


# DenseNet

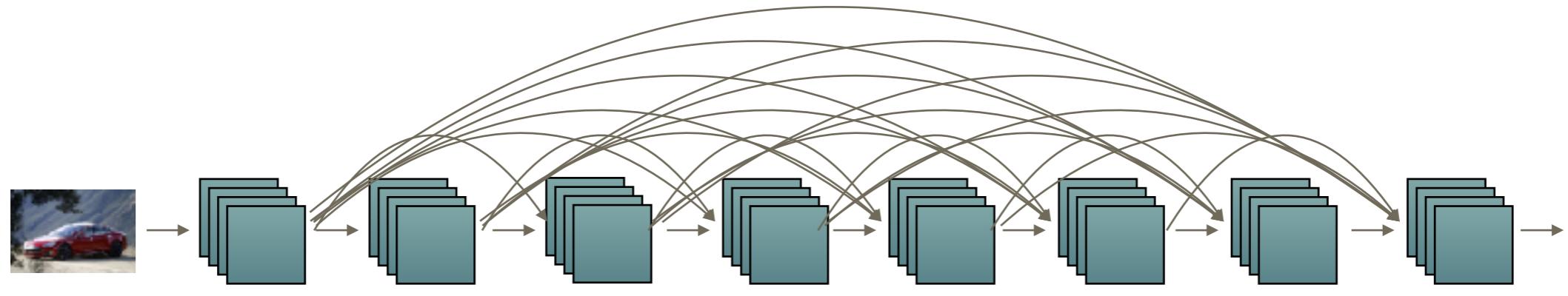
Dense Block (4 layers, growth rate 4)



# DenseNet

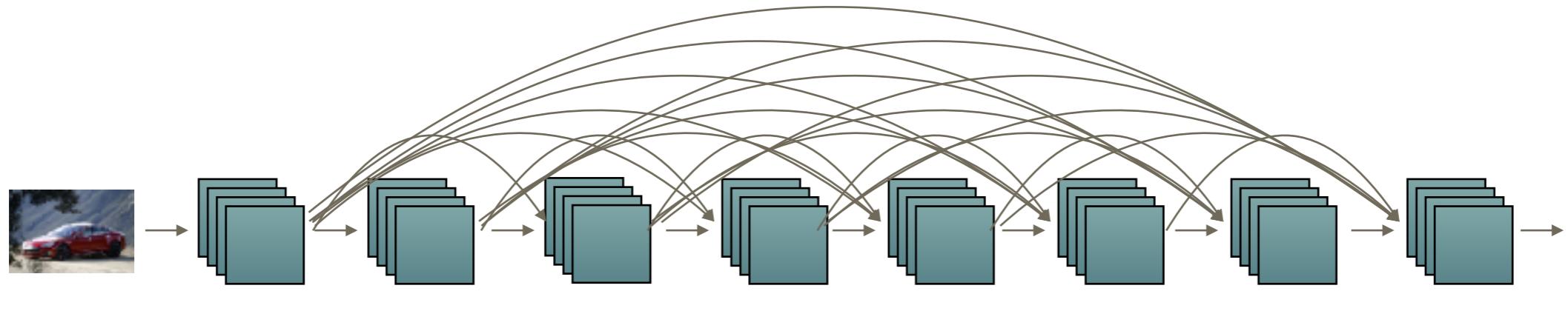


# DenseNet



*ensemble of models of different depth*

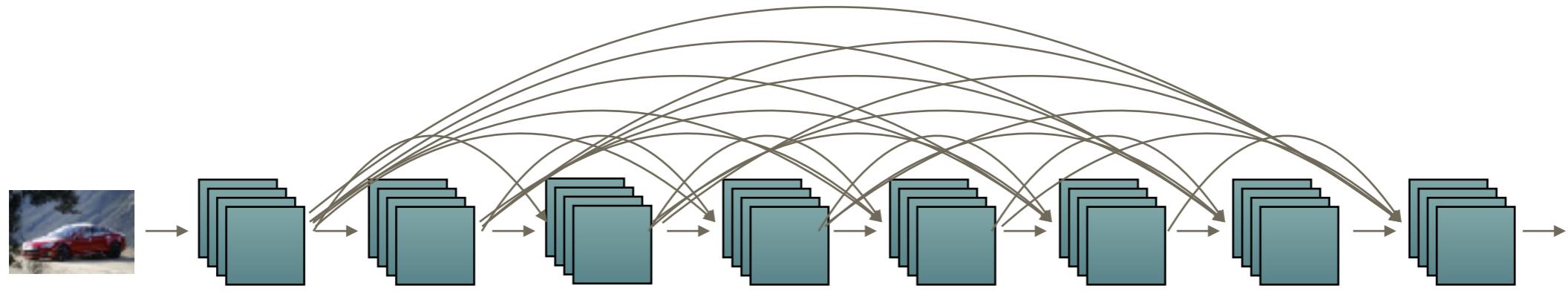
# DenseNet



ensemble of models of different depth

removes reliance to single layer

# DenseNet

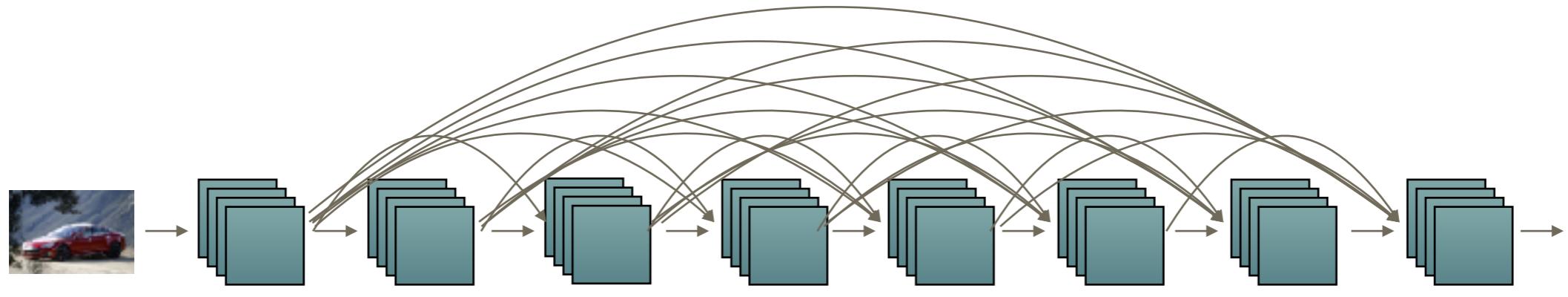


ensemble of models of different depth

removes reliance to single layer

information flow through direct connections

# DenseNet



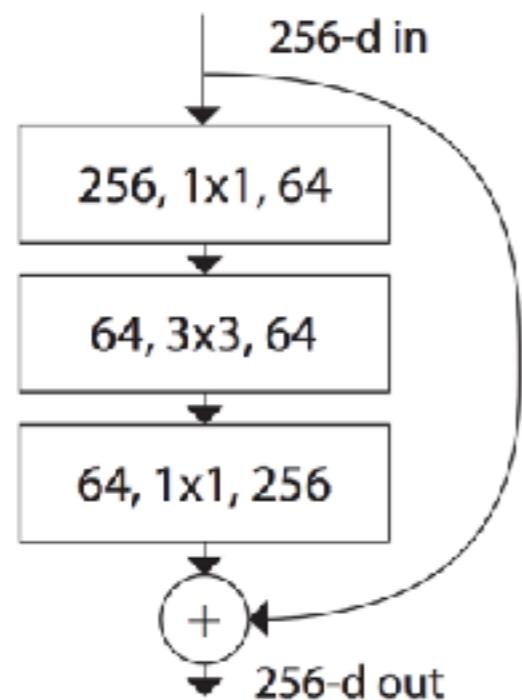
ensemble of models of different depth

removes reliance to single layer

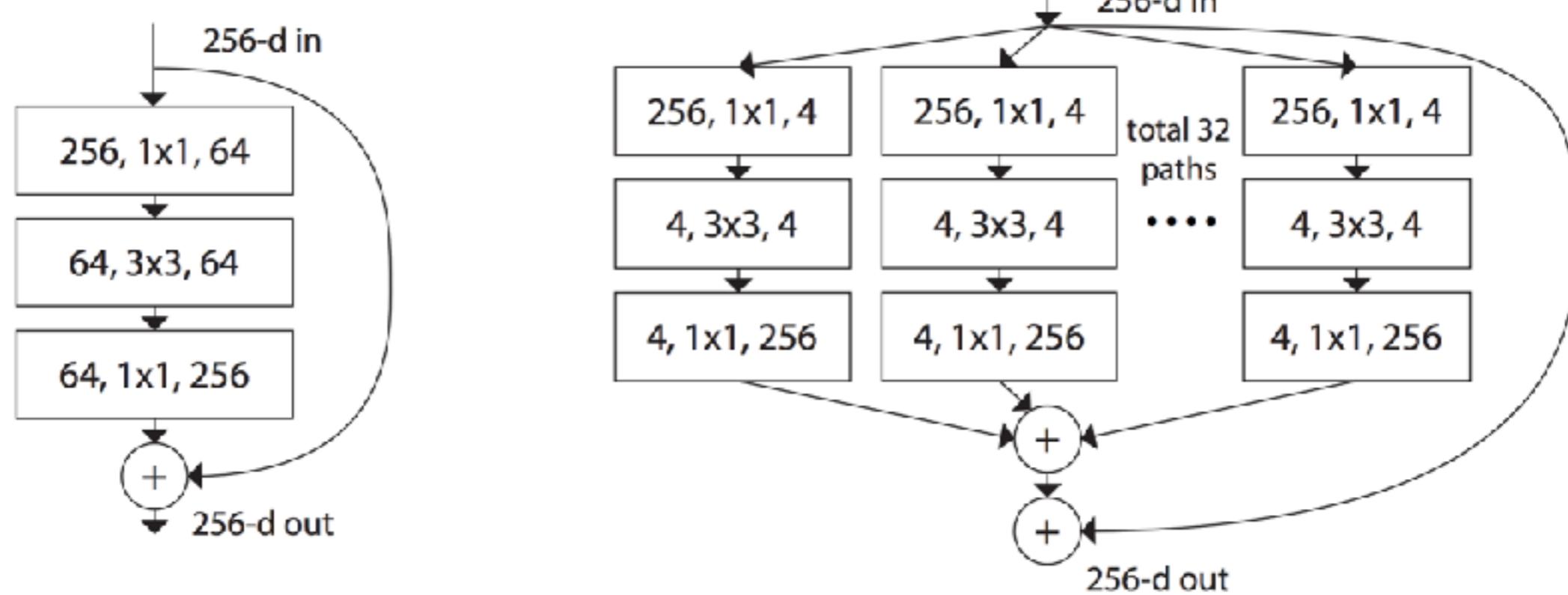
information flow through direct connections

do not include redundancy

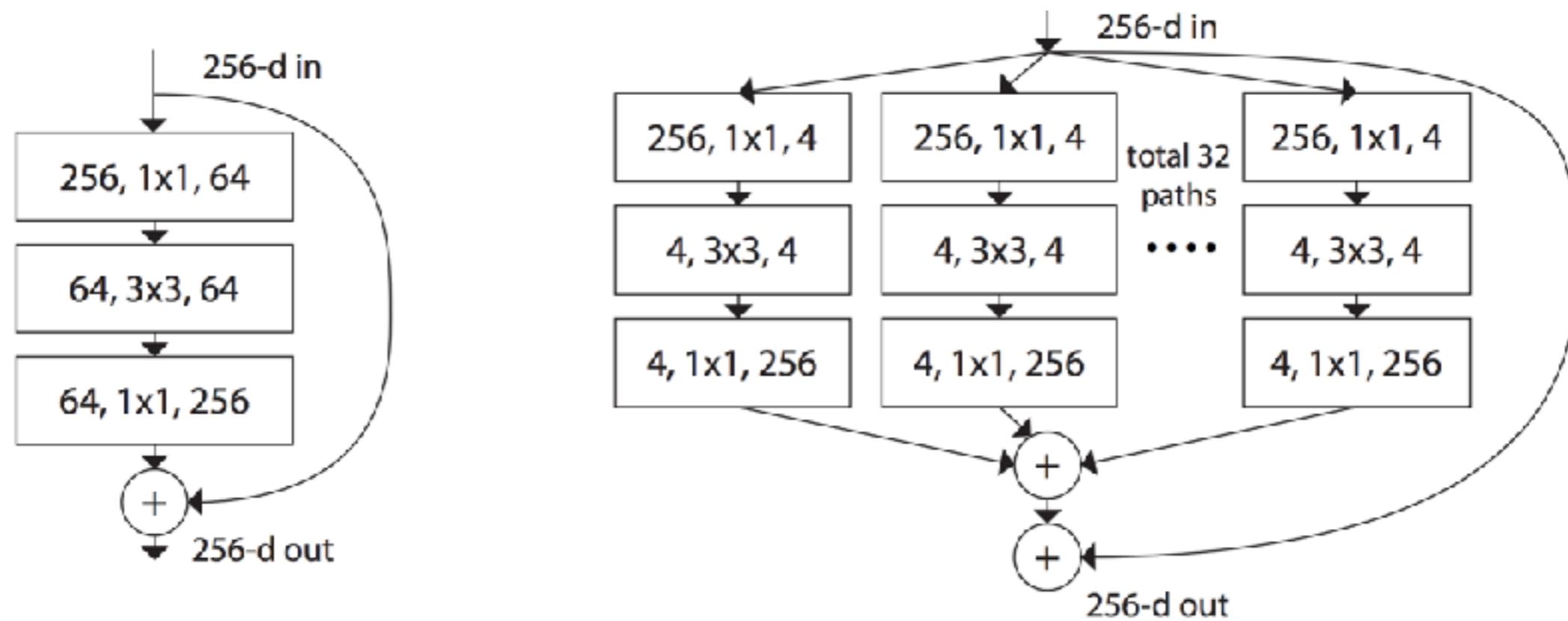
# ResNeXt



# ResNeXt

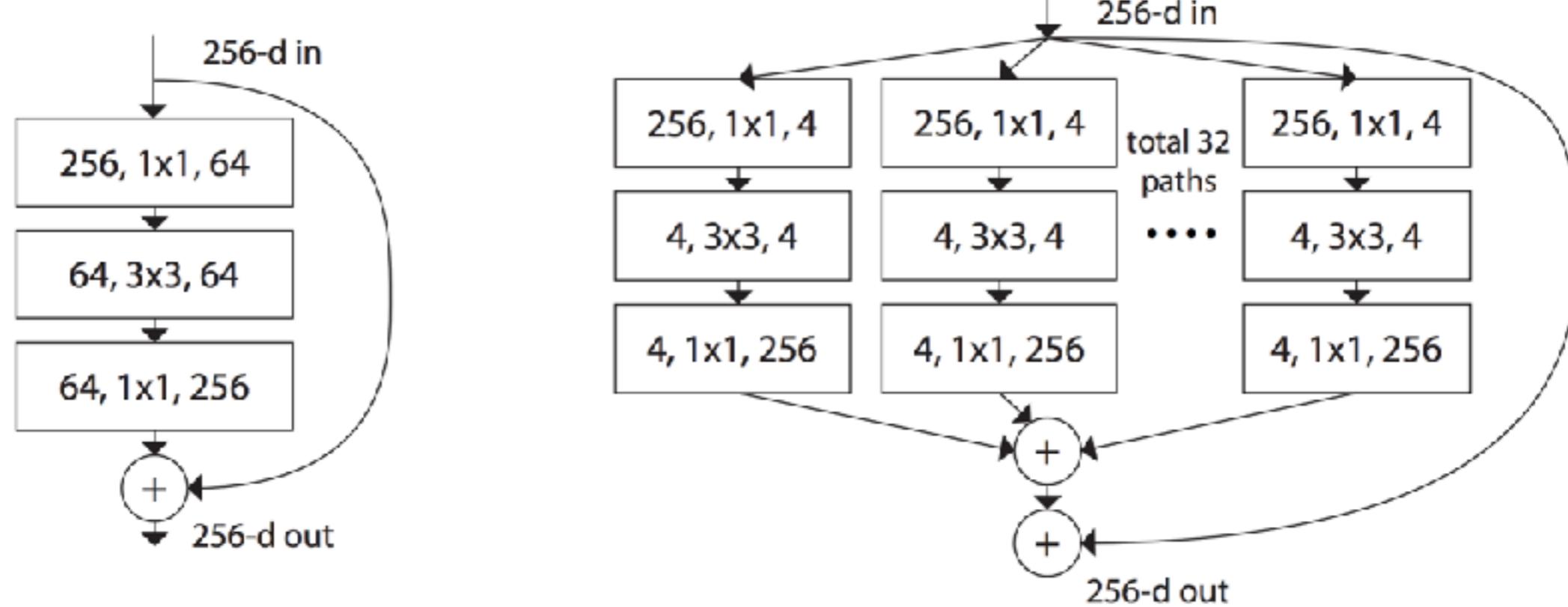


# ResNeXt



split (lower dim.), transform (specialized filters), merge strategy

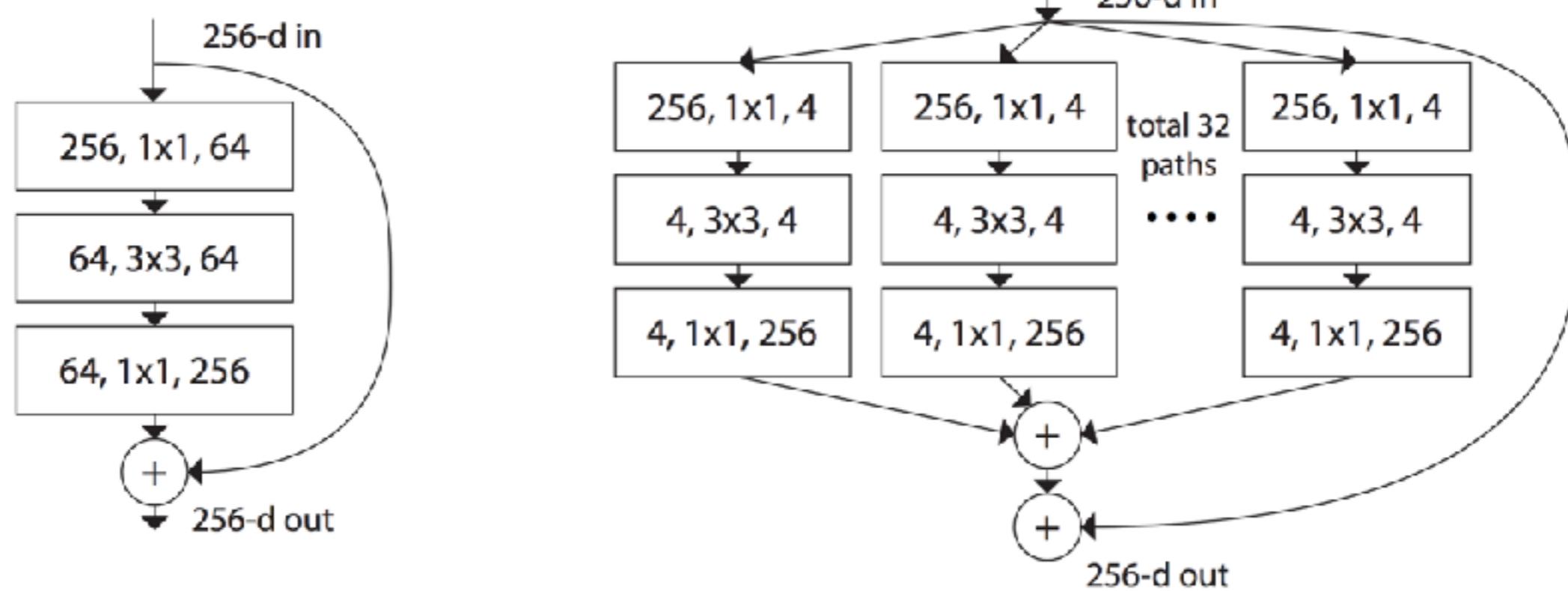
# ResNeXt



split (lower dim.), transform (specialized filters), merge strategy

all transformations to be aggregated have the same topology

# ResNeXt

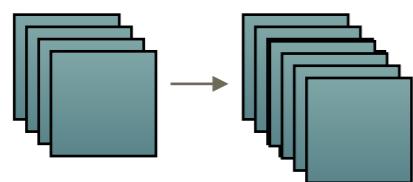


split (lower dim.), transform (specialized filters), merge strategy

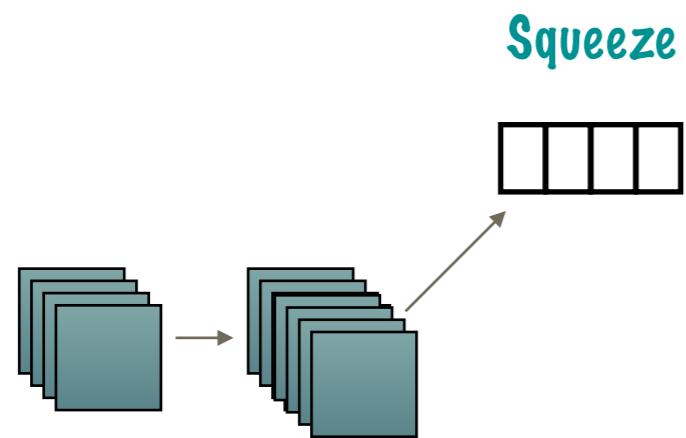
all transformations to be aggregated have the same topology

increasing cardinality shows better results than increasing depth/width

# Squeeze-&-Excitation Networks

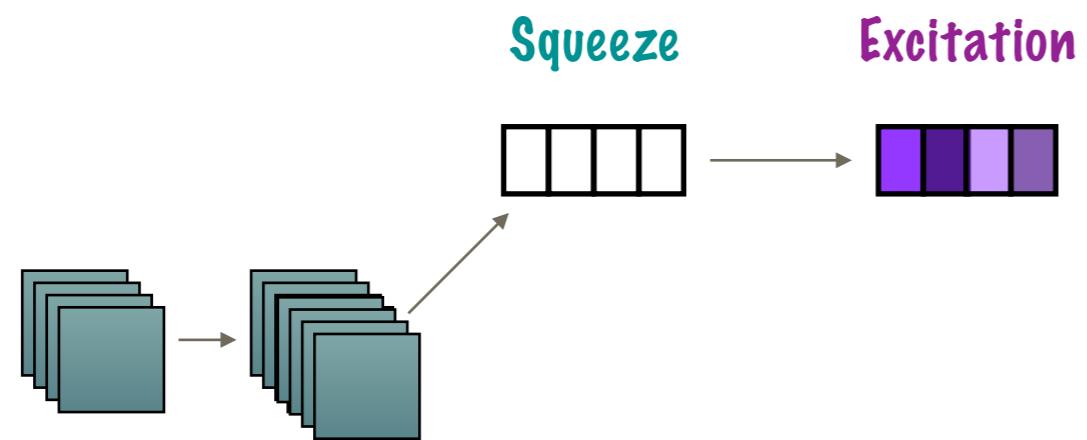


# Squeeze-&-Excitation Networks



Squeeze: avg pooling - captures channel-wise global info

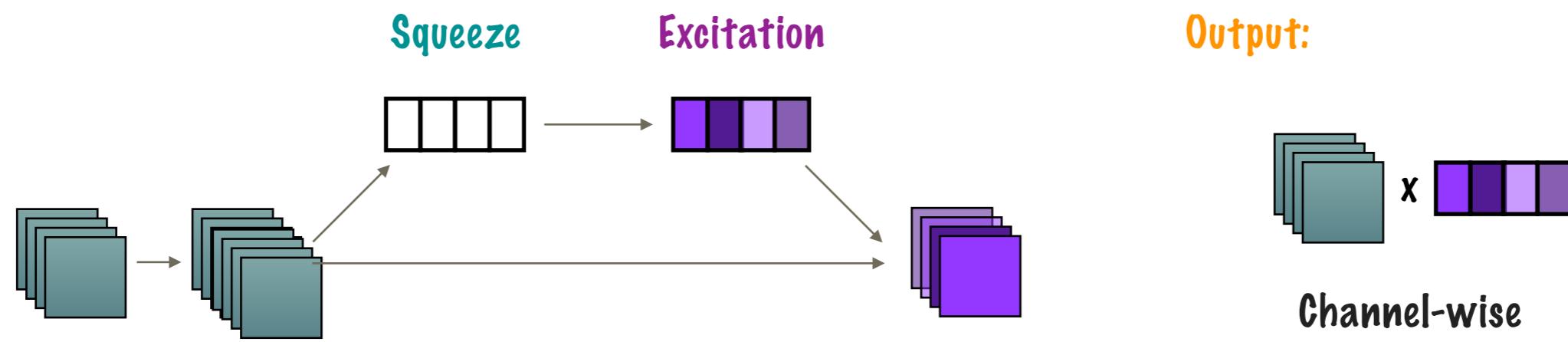
# Squeeze-&-Excitation Networks



Squeeze: avg pooling - captures channel-wise global info

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

# Squeeze-&-Excitation Networks

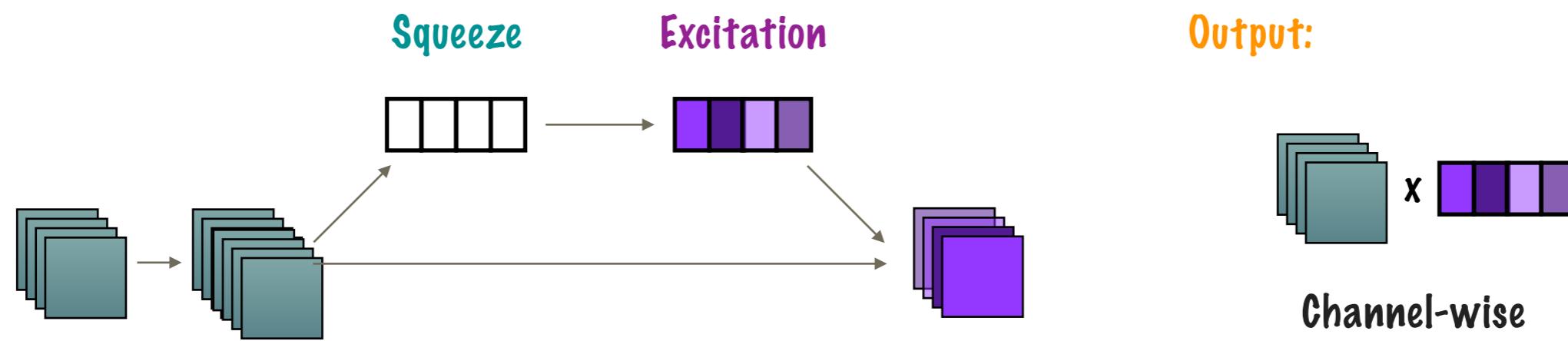


Squeeze: avg pooling - captures channel-wise global info

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

Output: scales activations, leveraging global information at all levels

# Squeeze-&-Excitation Networks



Squeeze: avg pooling - captures channel-wise global info

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

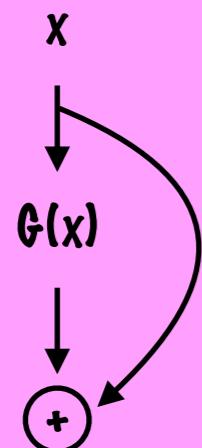
Output: scales activations, leveraging global information at all levels

Won ILSVRC 2017 challenge

# **ResNet vs ResNeXt vs DenseNet**

# ResNet vs ResNeXt vs DenseNet

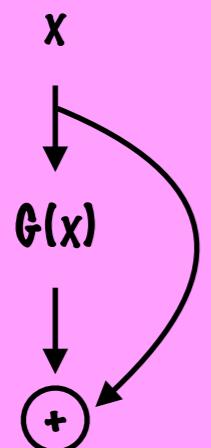
## ResNets



$$G(x) = \left( \begin{array}{c} \text{BN} \\ + \\ \text{ReLU} \\ + \\ \text{Conv.} \end{array} \right) x n$$

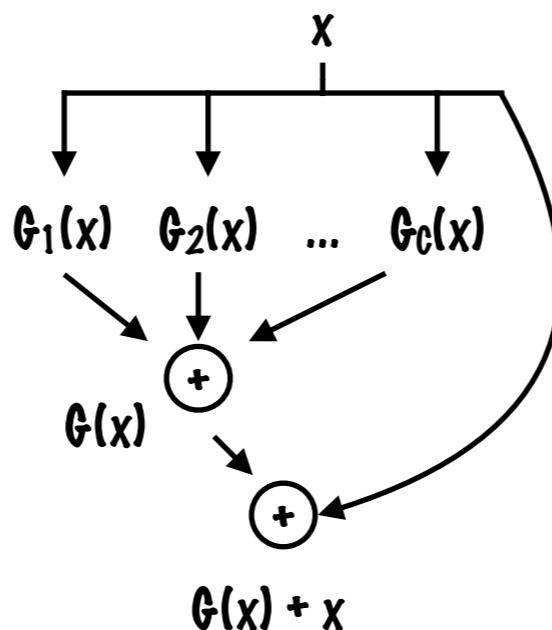
# ResNet vs ResNeXt vs DenseNet

## ResNets



$$G(x) = \left( \begin{array}{c} \text{BN} \\ + \\ \text{ReLU} \\ + \\ \text{Conv.} \end{array} \right) x n$$

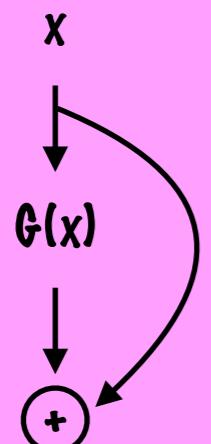
## ResNeXts



$$G_i(x) = \left( \begin{array}{c} \text{BN} \\ + \\ \text{ReLU} \\ + \\ \text{Conv.} \end{array} \right) x n$$

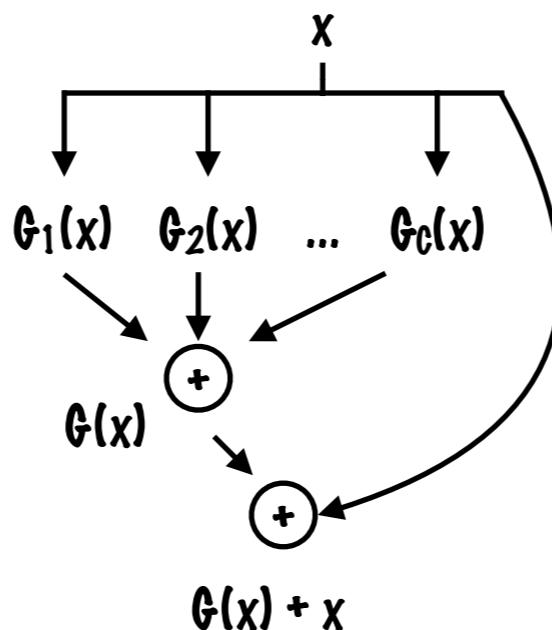
# ResNet vs ResNeXt vs DenseNet

## ResNets



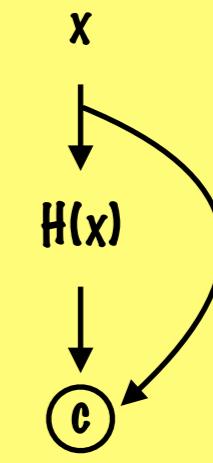
$$G(x) = \left( \begin{array}{c} \text{BN} \\ + \\ \text{ReLU} \\ + \\ \text{Conv.} \end{array} \right) x n$$

## ResNeXts



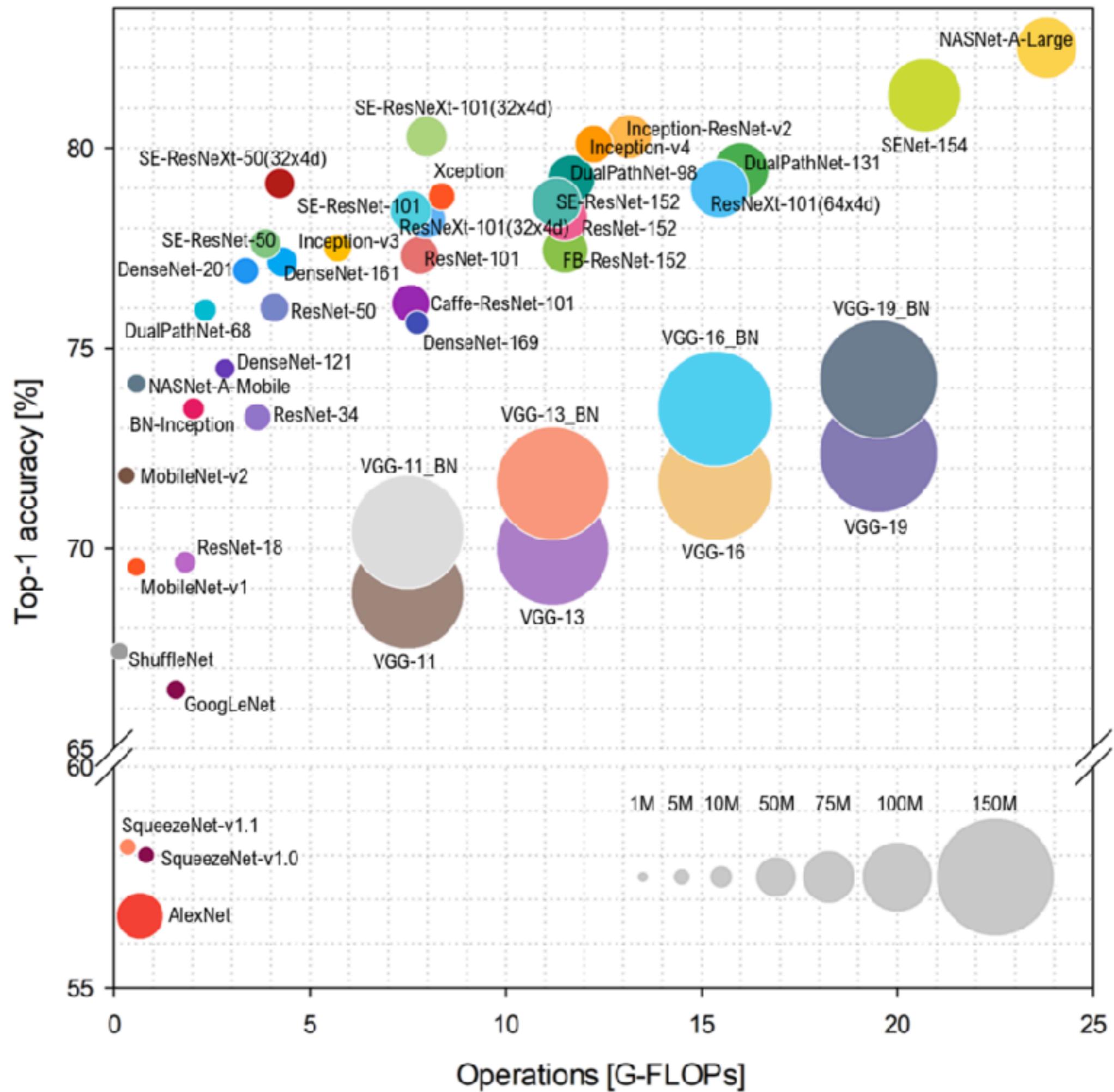
$$G_i(x) = \left( \begin{array}{c} \text{BN} \\ + \\ \text{ReLU} \\ + \\ \text{Conv.} \end{array} \right) x n$$

## DenseNets

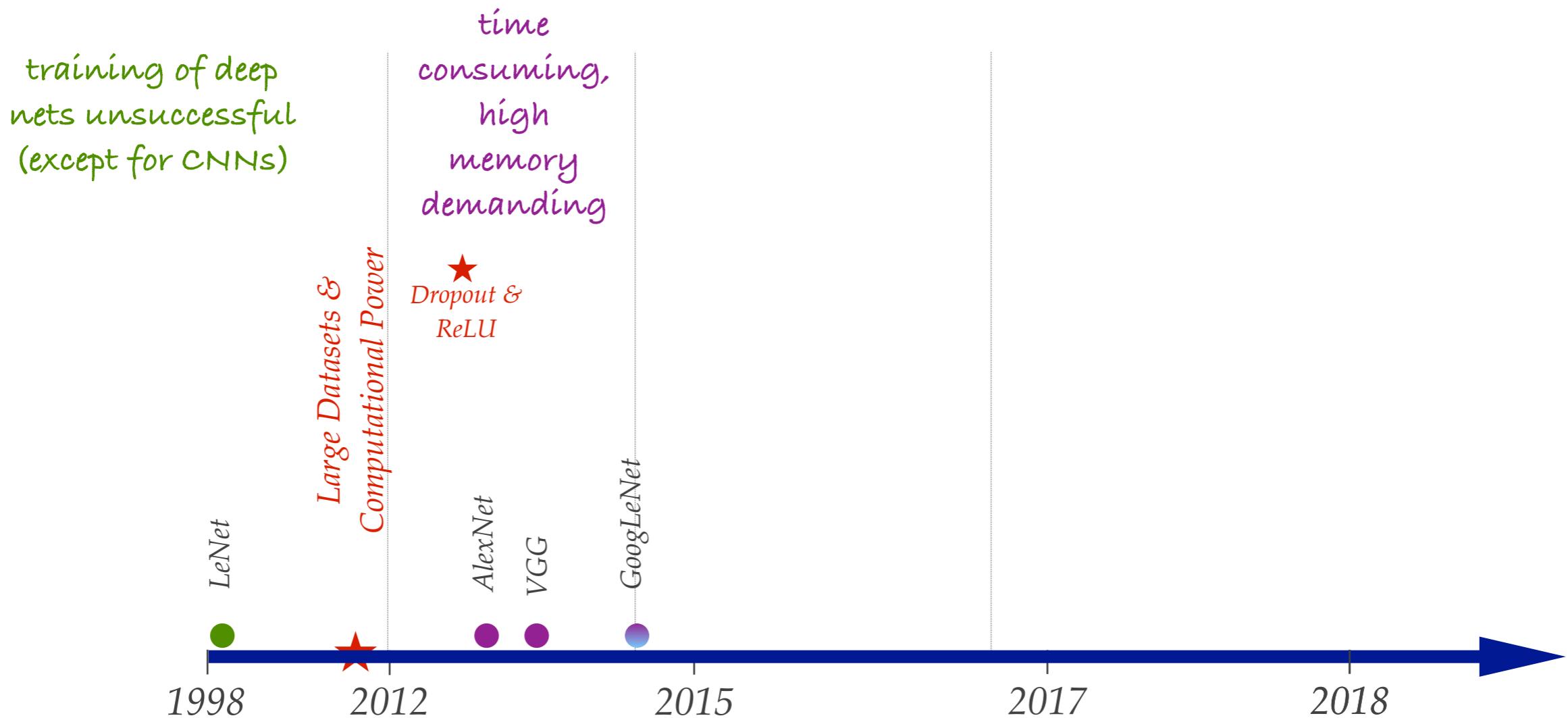


$$[H(x), x]$$

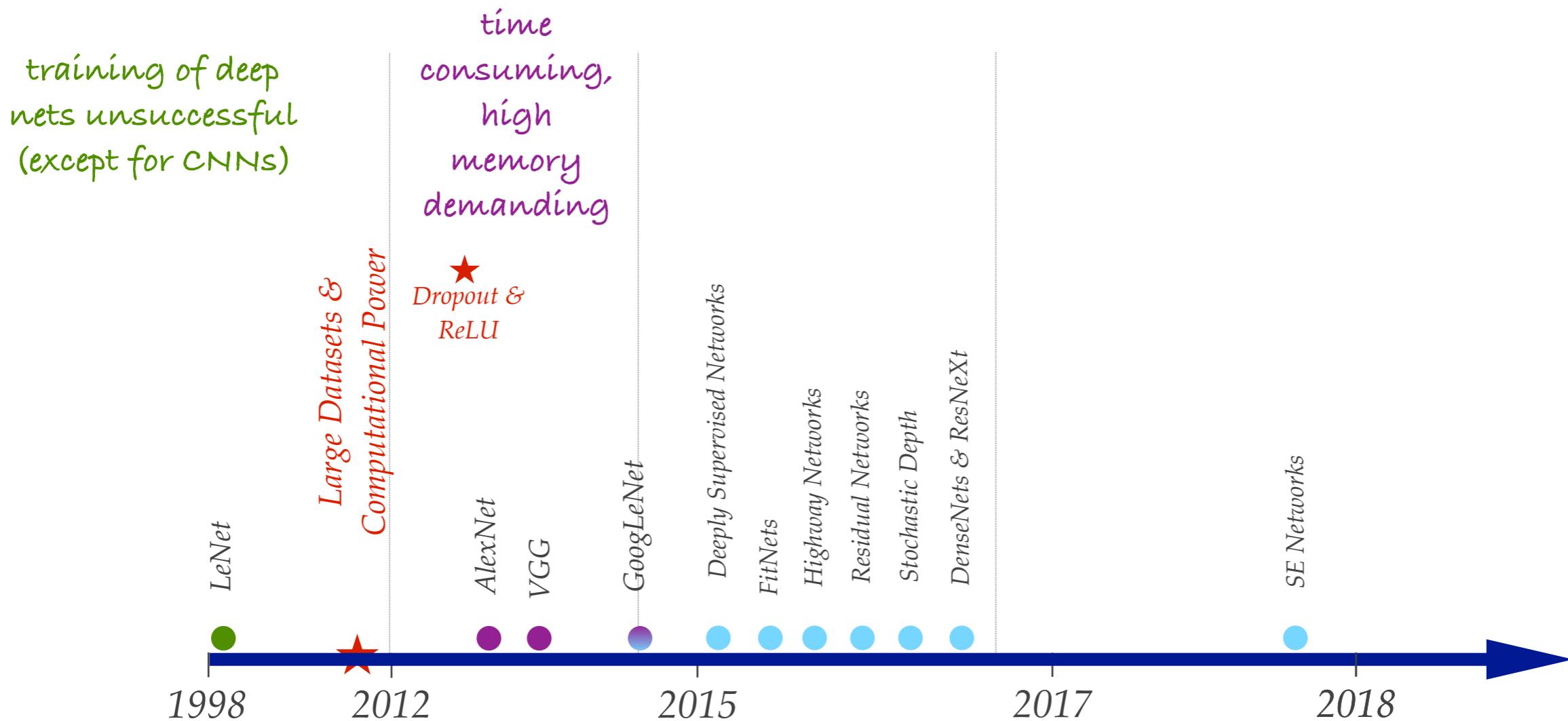
$$H(x) = \left( \begin{array}{c} \text{BN} \\ + \\ \text{ReLU} \\ + \\ \text{Conv.} \\ + \\ \text{Dropout} \end{array} \right)$$



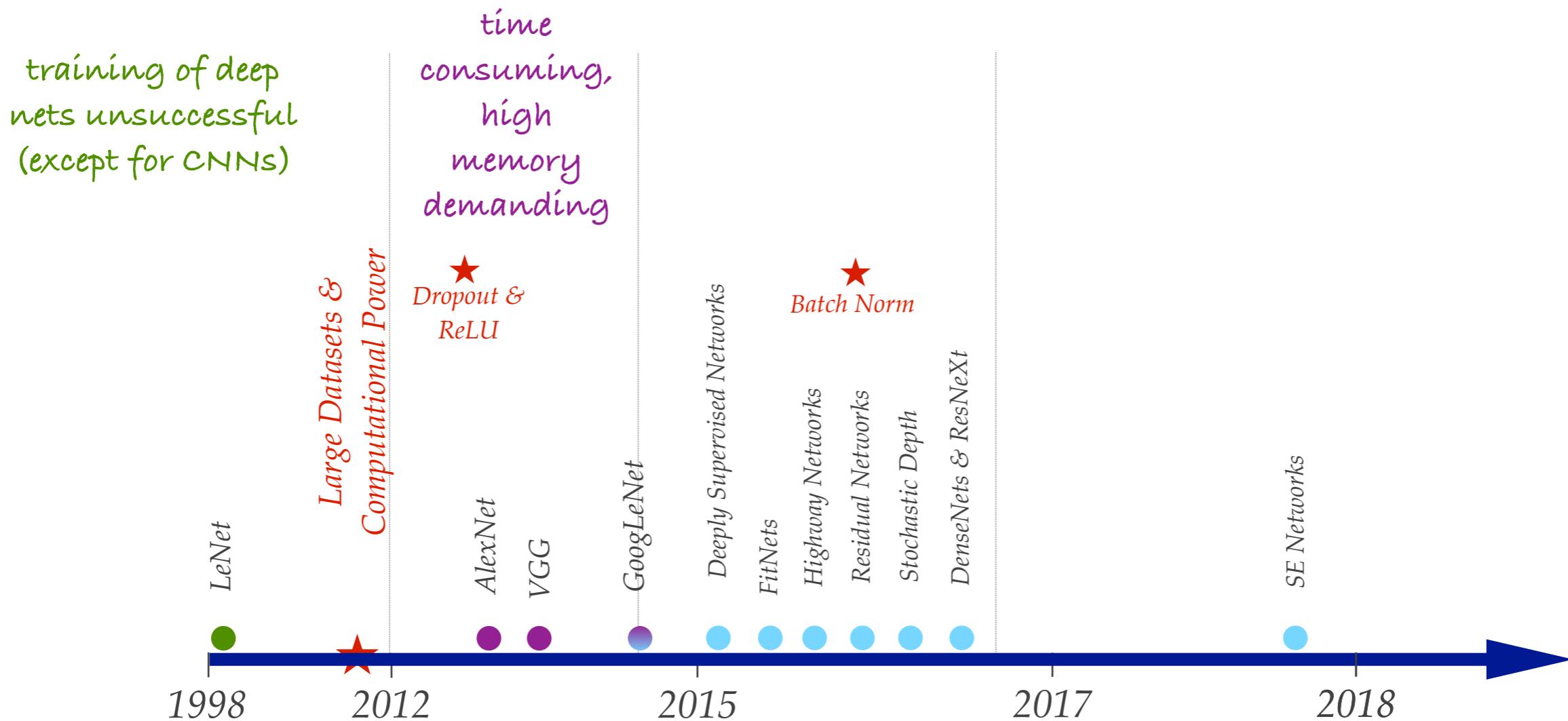
# Wrap Up



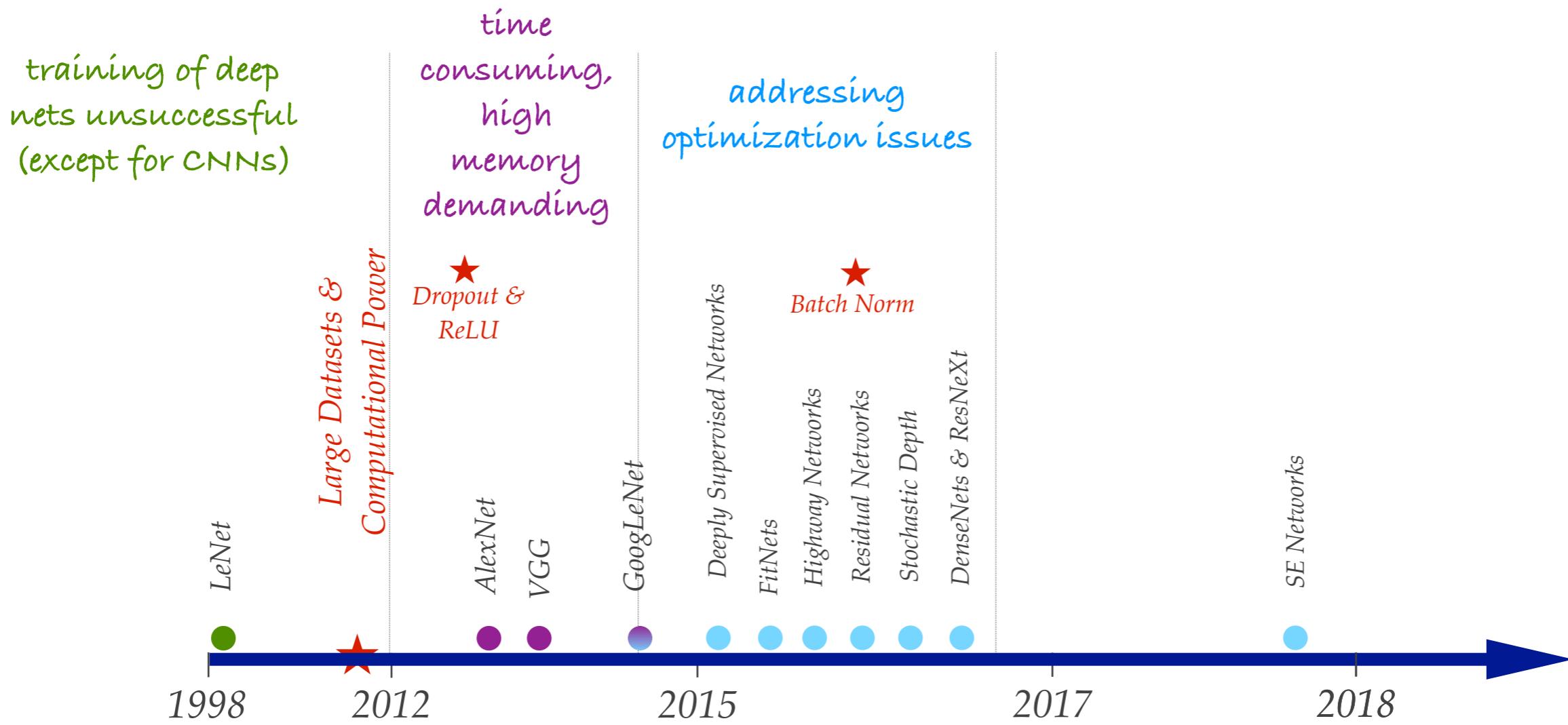
# Wrap Up



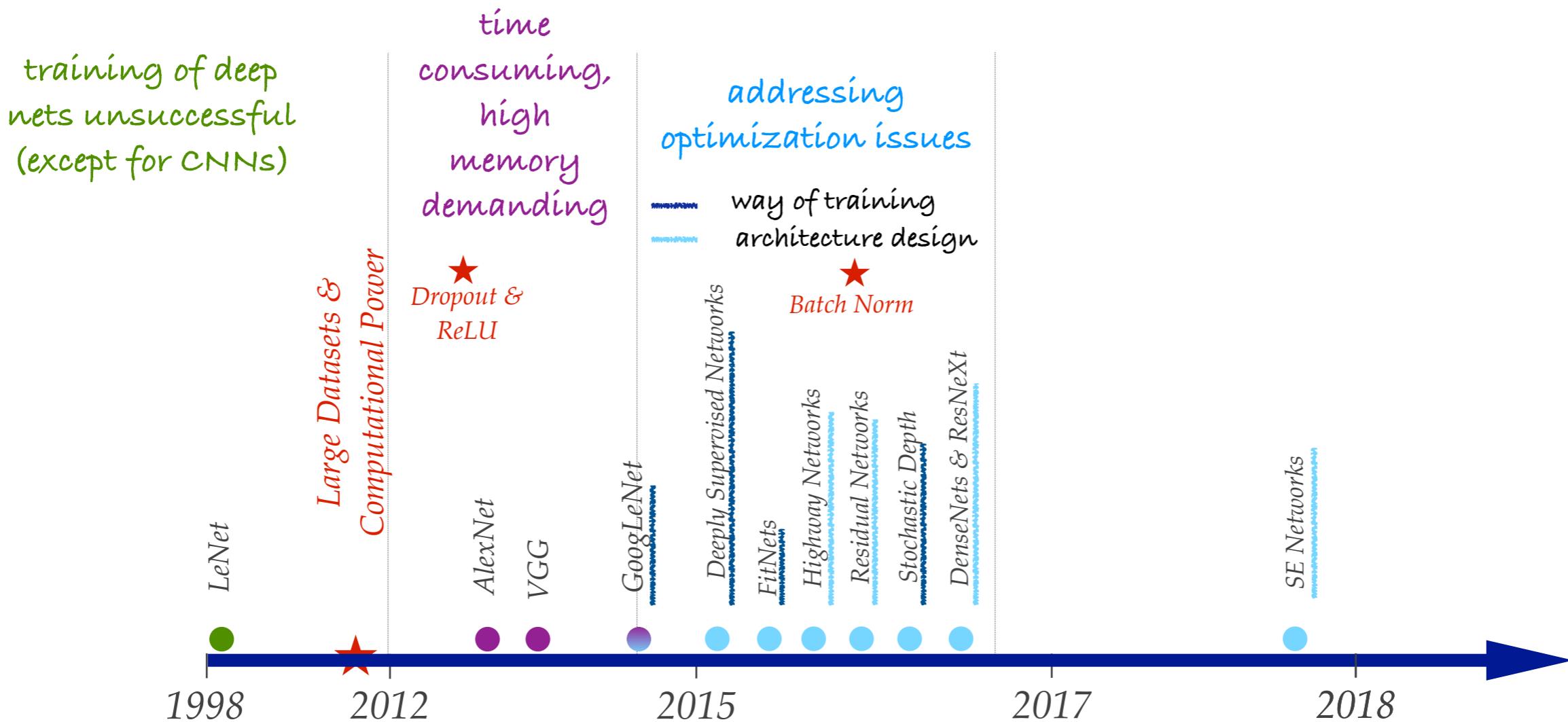
# Wrap Up



# Wrap Up



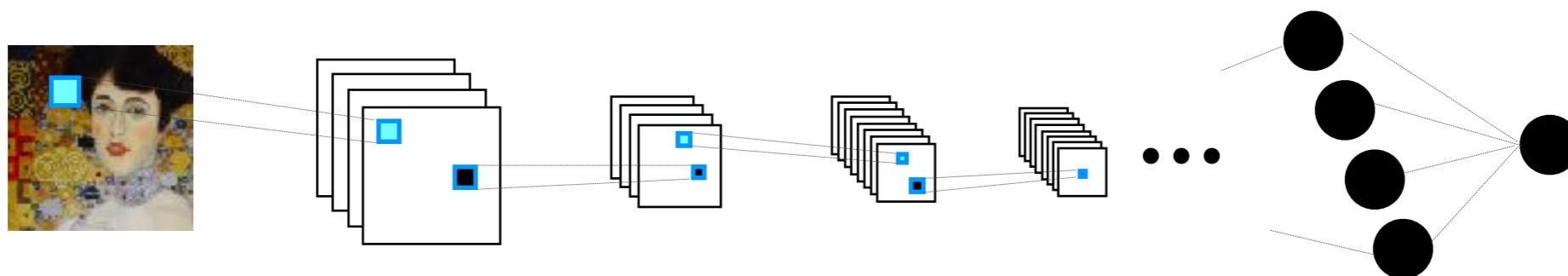
# Wrap Up



# Feature hierarchy vs. iterative inference

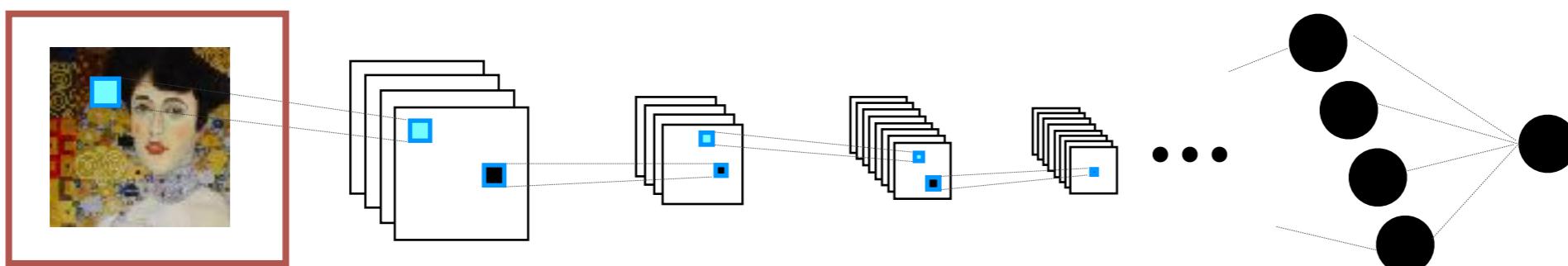
# Representation view

The traditional view:



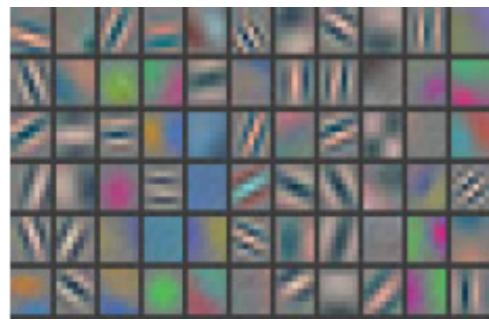
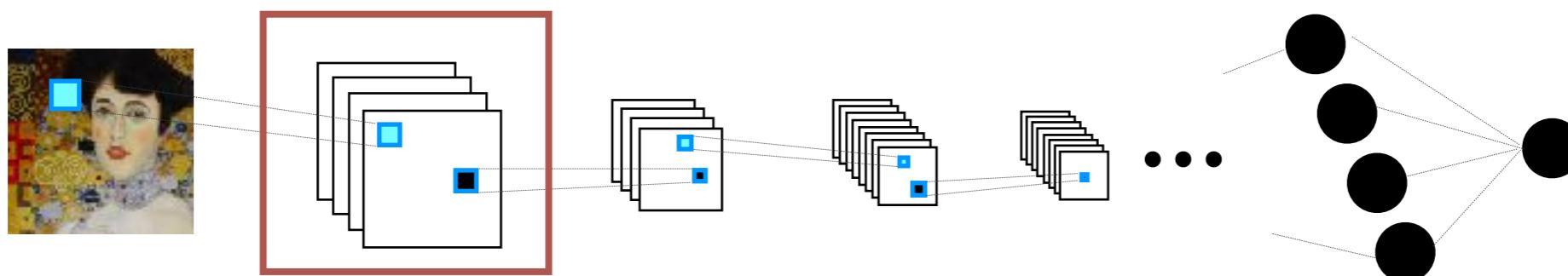
# Representation view

The traditional view:



# Representation view

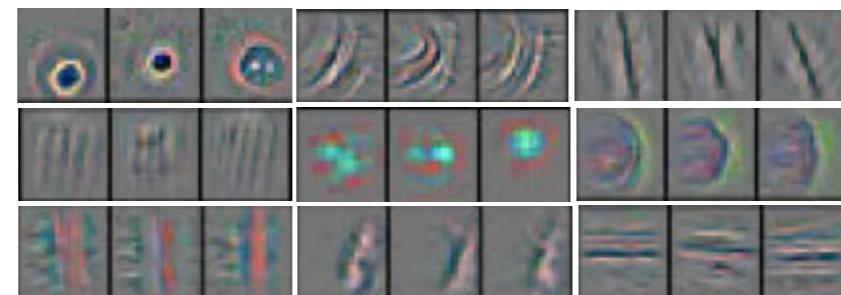
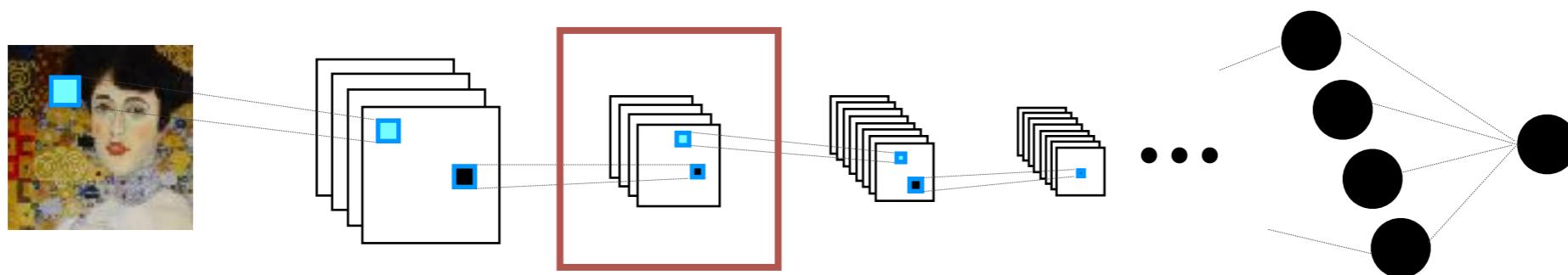
The traditional view:



(Zeiler and Fergus, 2013)

# Representation view

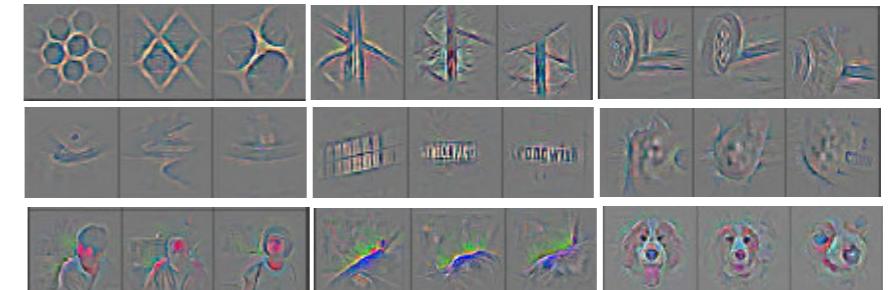
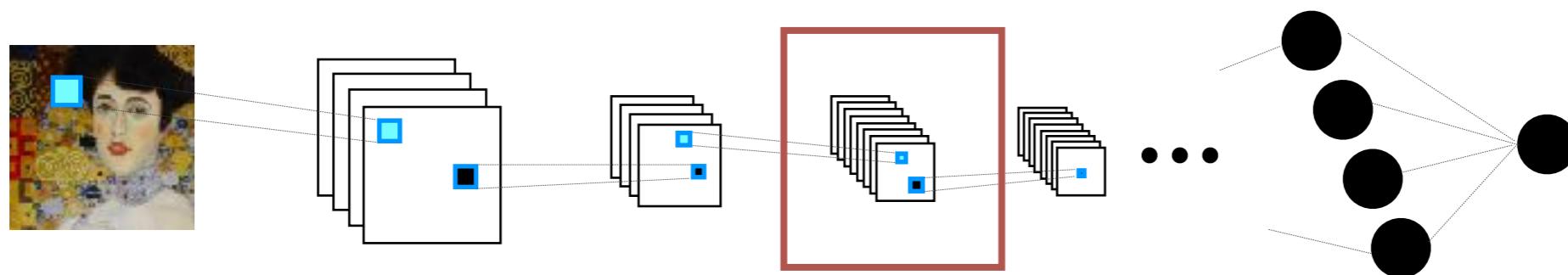
The traditional view:



(Zeiler and Fergus, 2013)

# Representation view

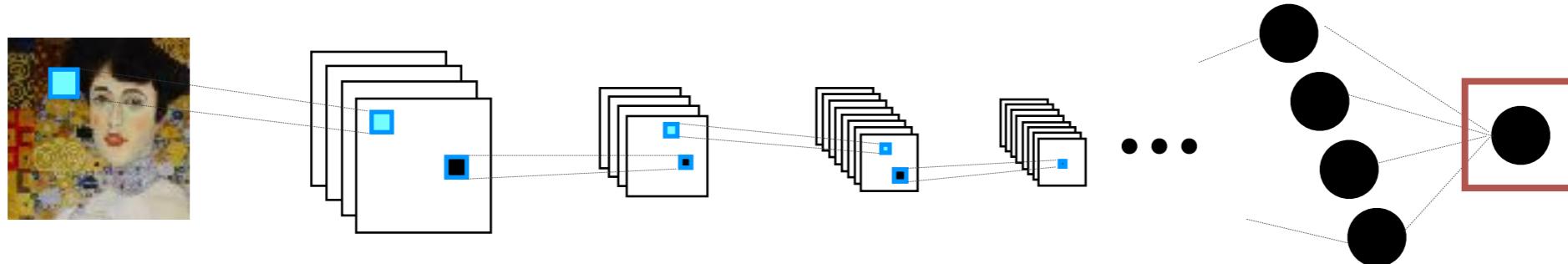
The traditional view:



(Zeiler and Fergus, 2013)

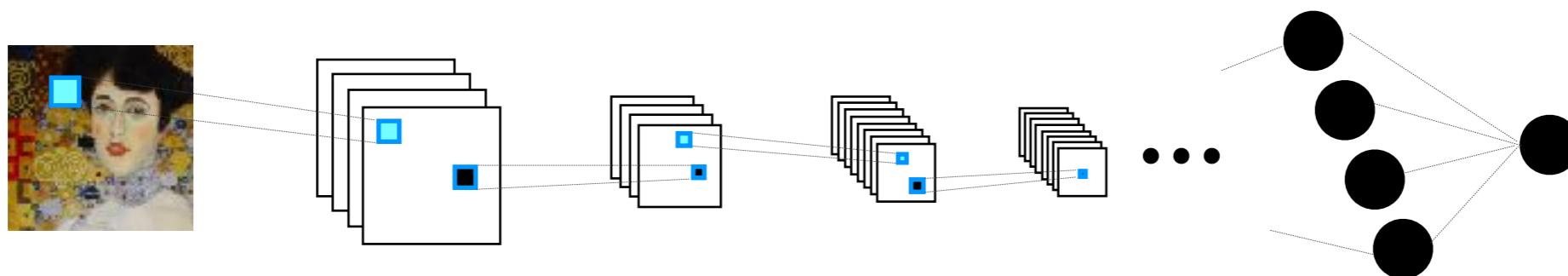
# Representation view

The traditional view:



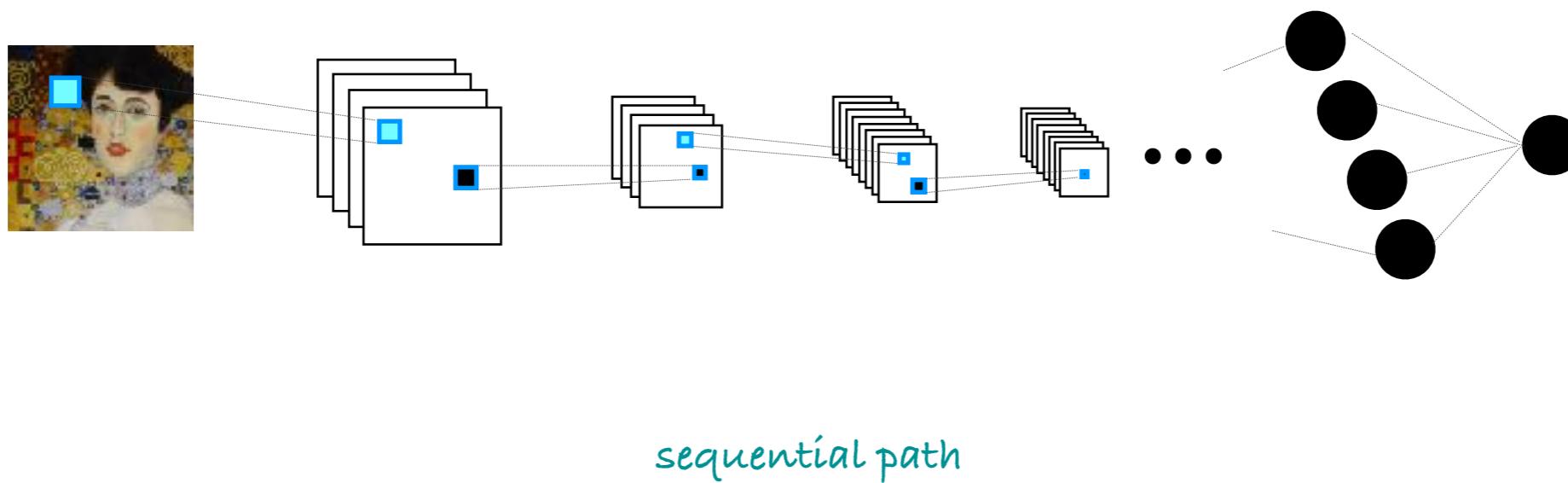
# Representation view

The traditional view:



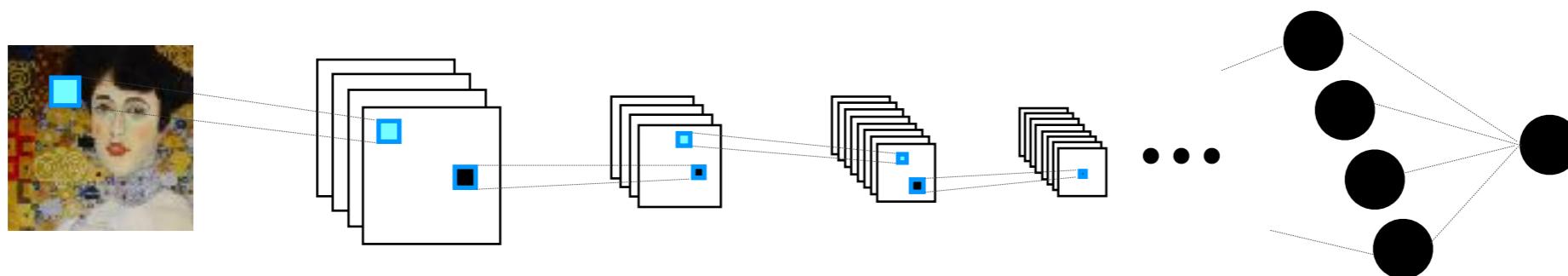
# Representation view

The traditional view:



# Representation view

The traditional view:

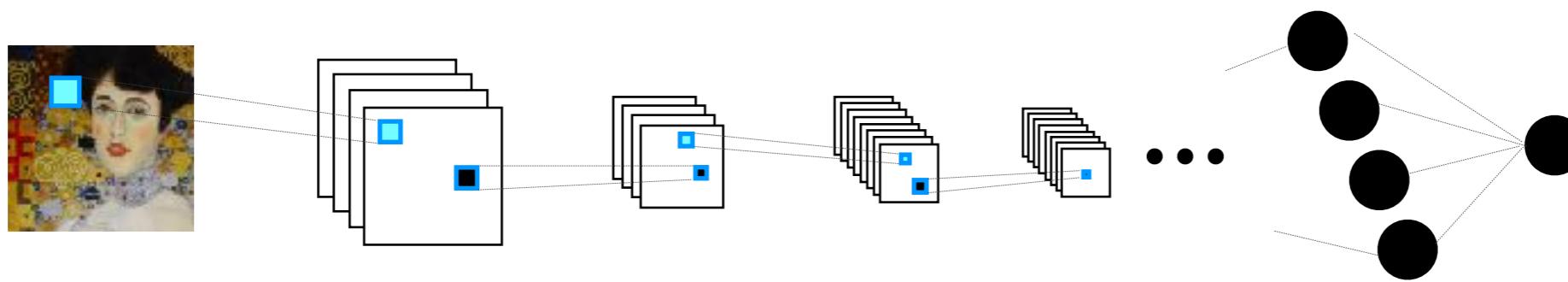


sequential path

each output depends only on the output of the previous layer

# Representation view

The traditional view:



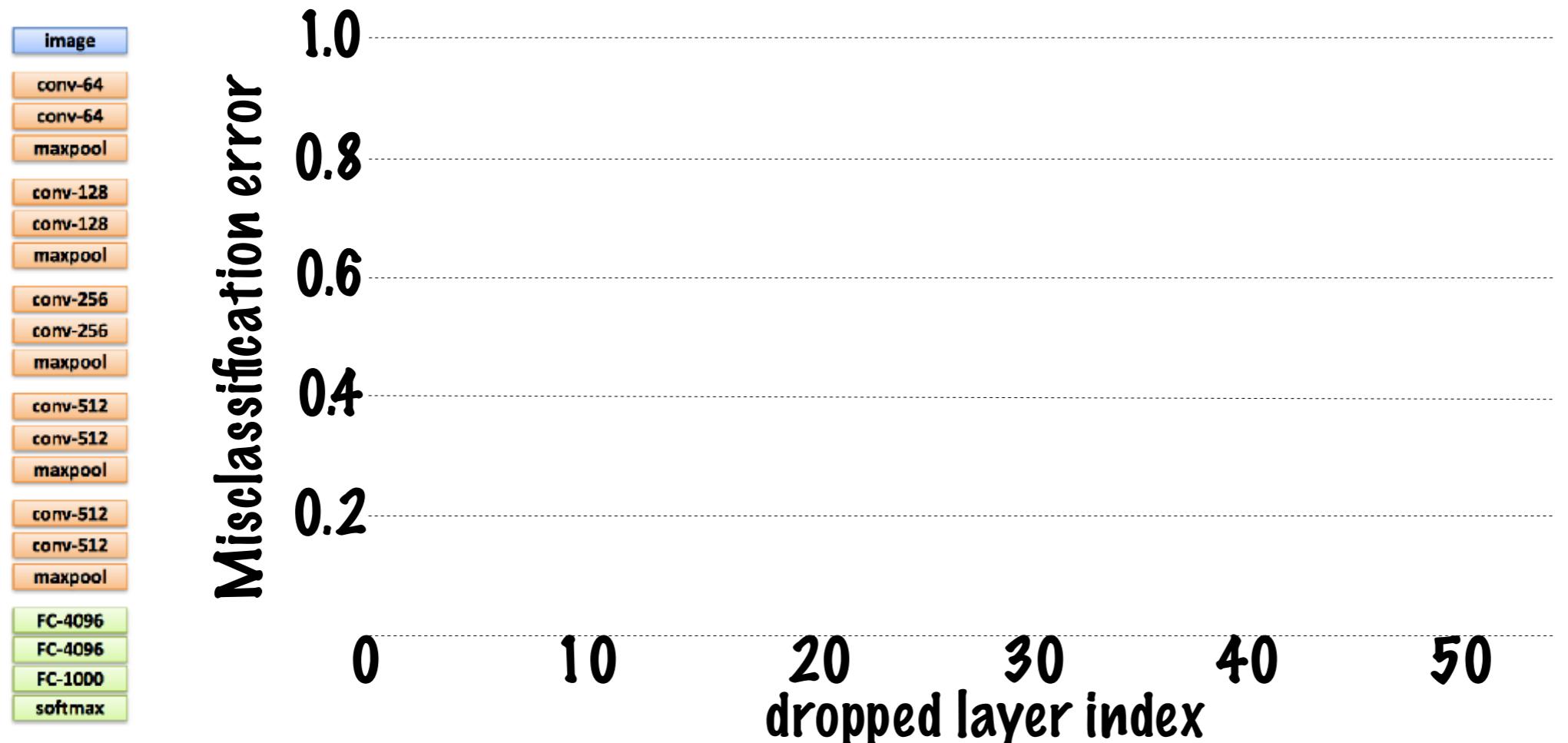
sequential path

each output depends only on the output of the previous layer

hierarchical computation of increasingly abstract features

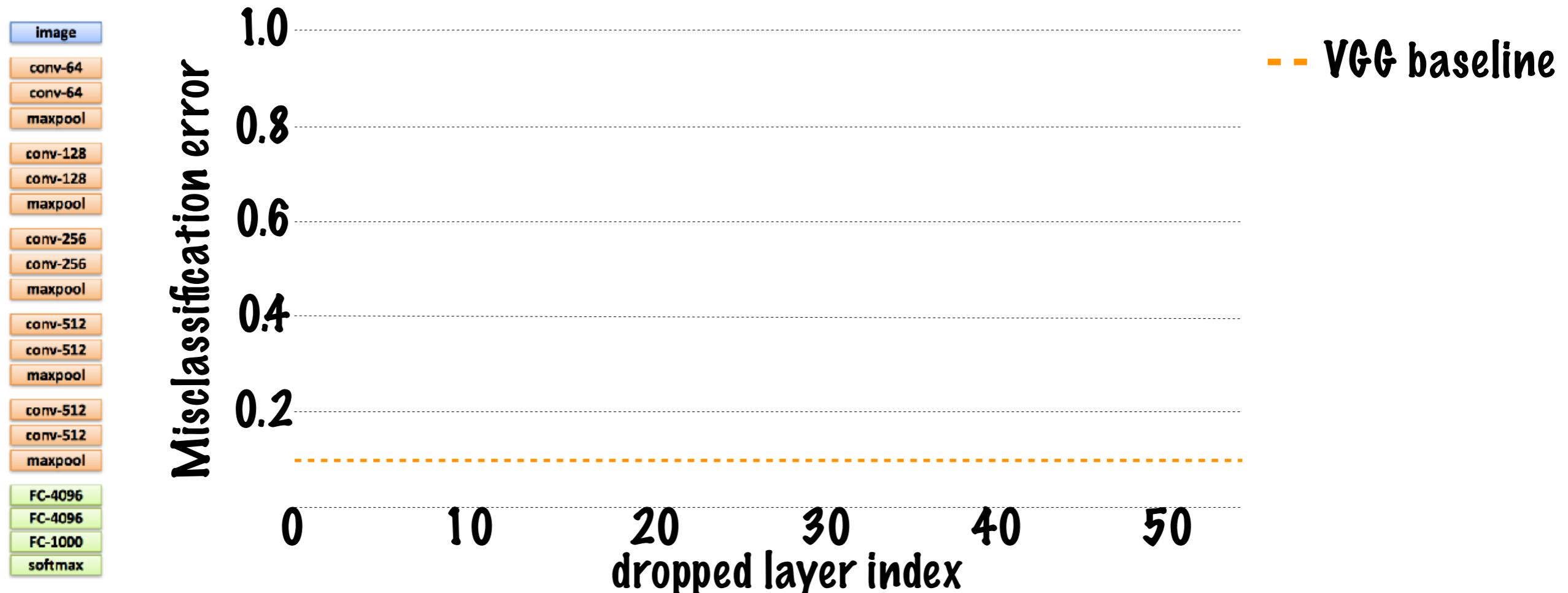
# Lesion Study

Dropping VGG layers:



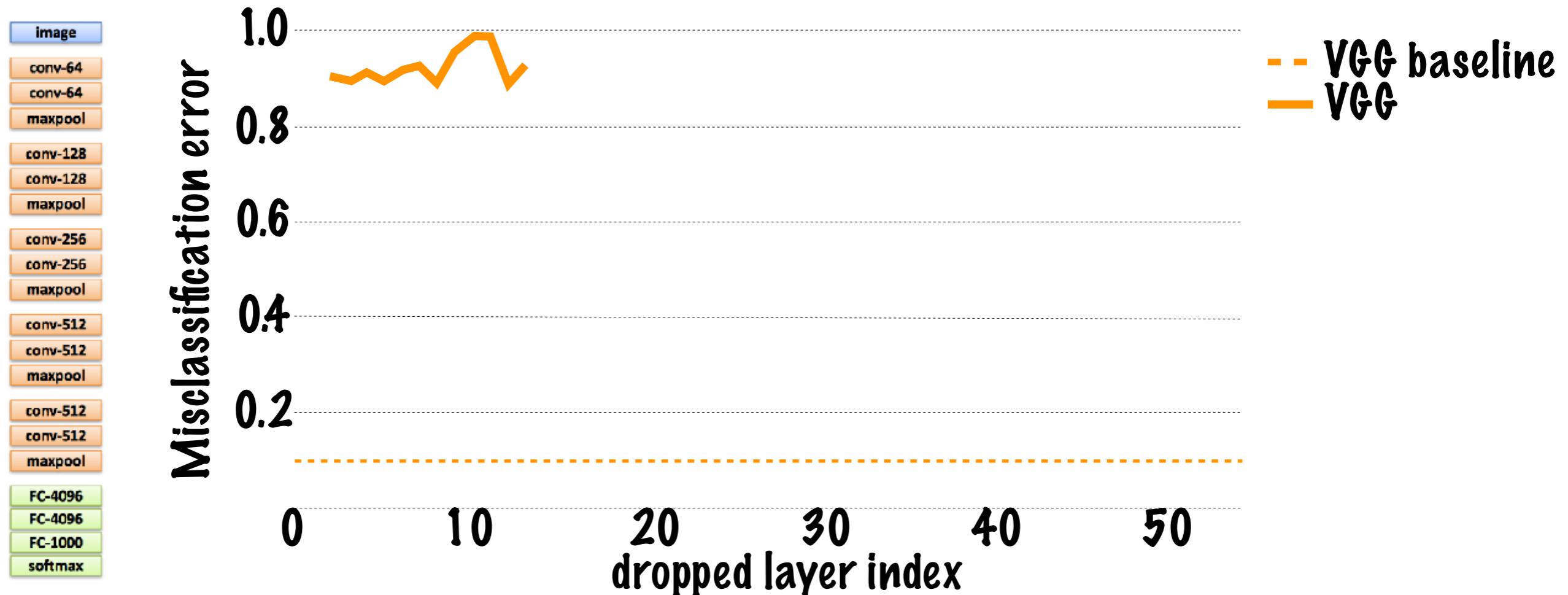
# Lesion Study

Dropping VGG layers:



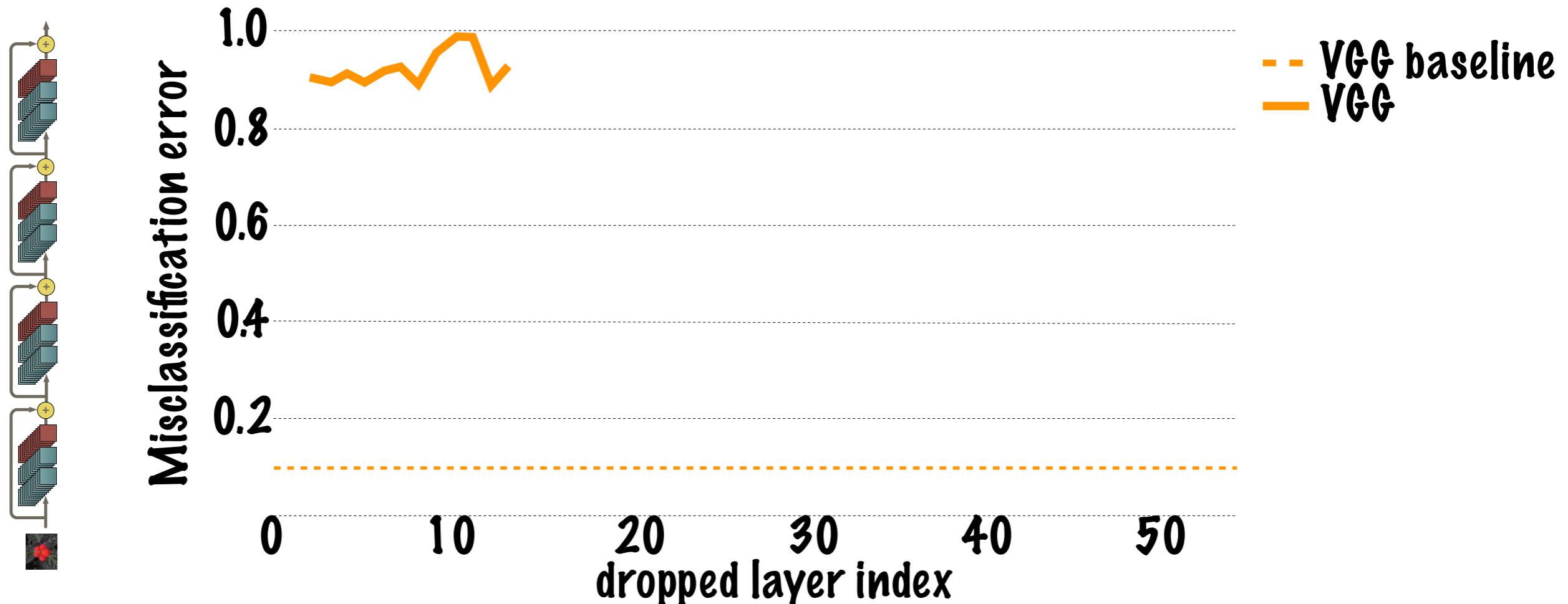
# Lesion Study

Dropping VGG layers:



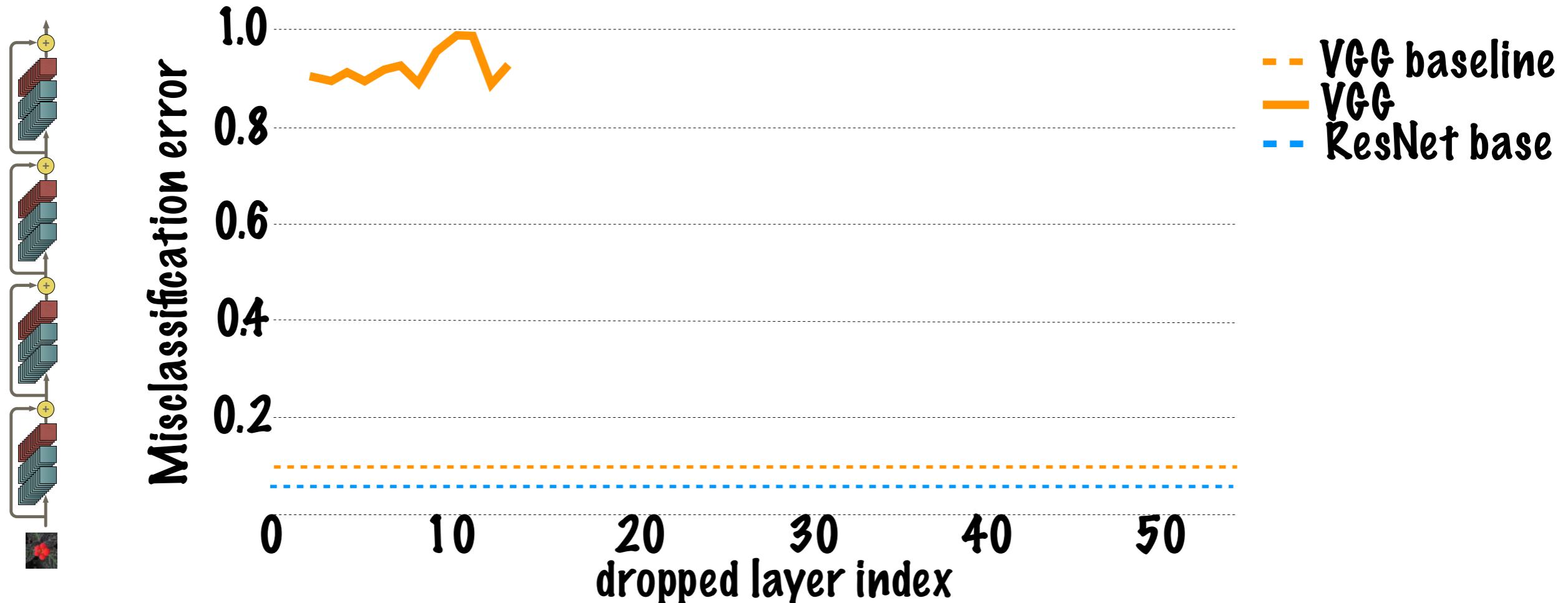
# Lesion Study

Dropping VGG layers:



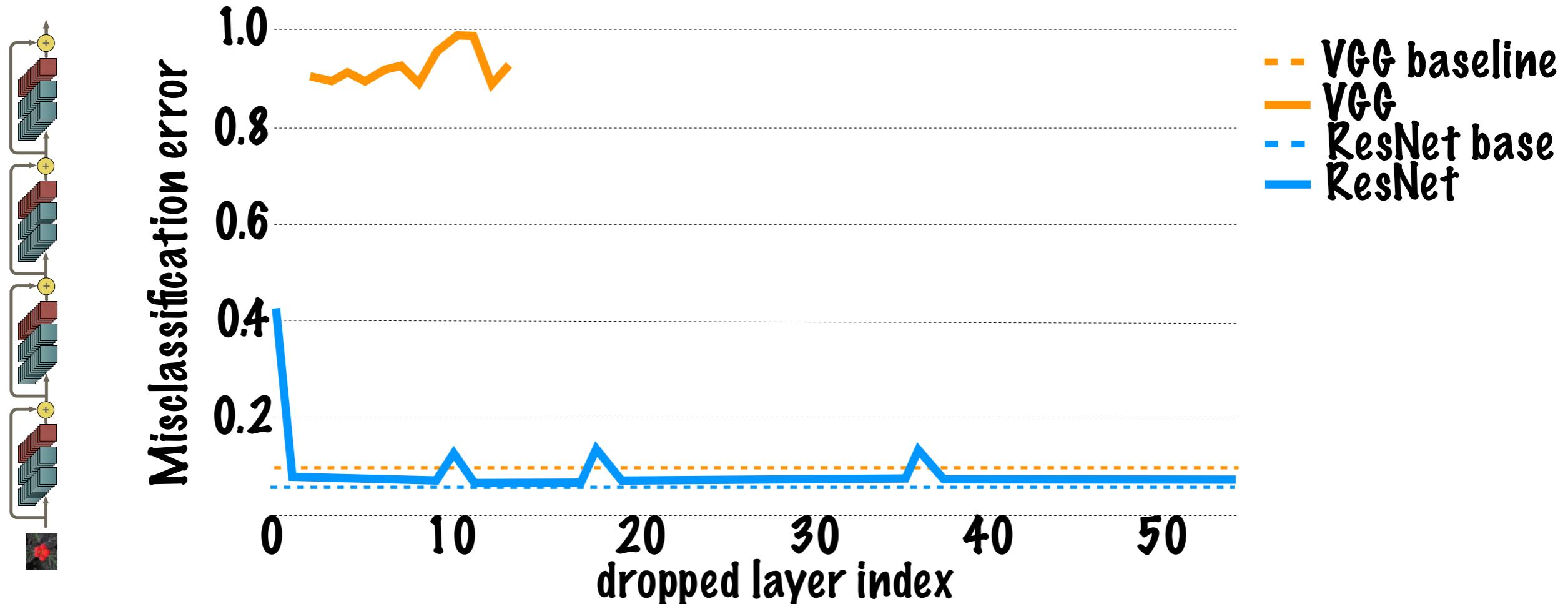
# Lesion Study

Dropping VGG layers:



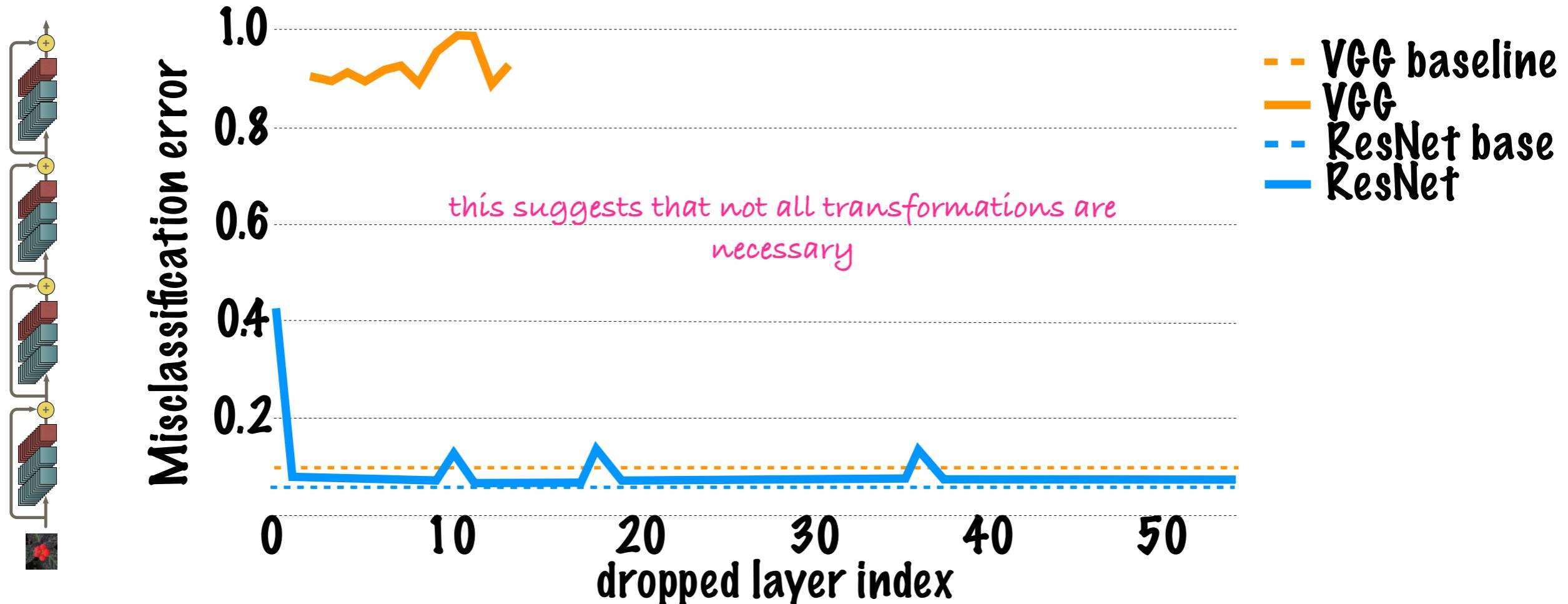
# Lesion Study

Dropping VGG layers:



# Lesion Study

Dropping VGG layers:



# Lesion Study

Swapping k pairs of ResNet blocks within the same resolution:

# Lesion Study

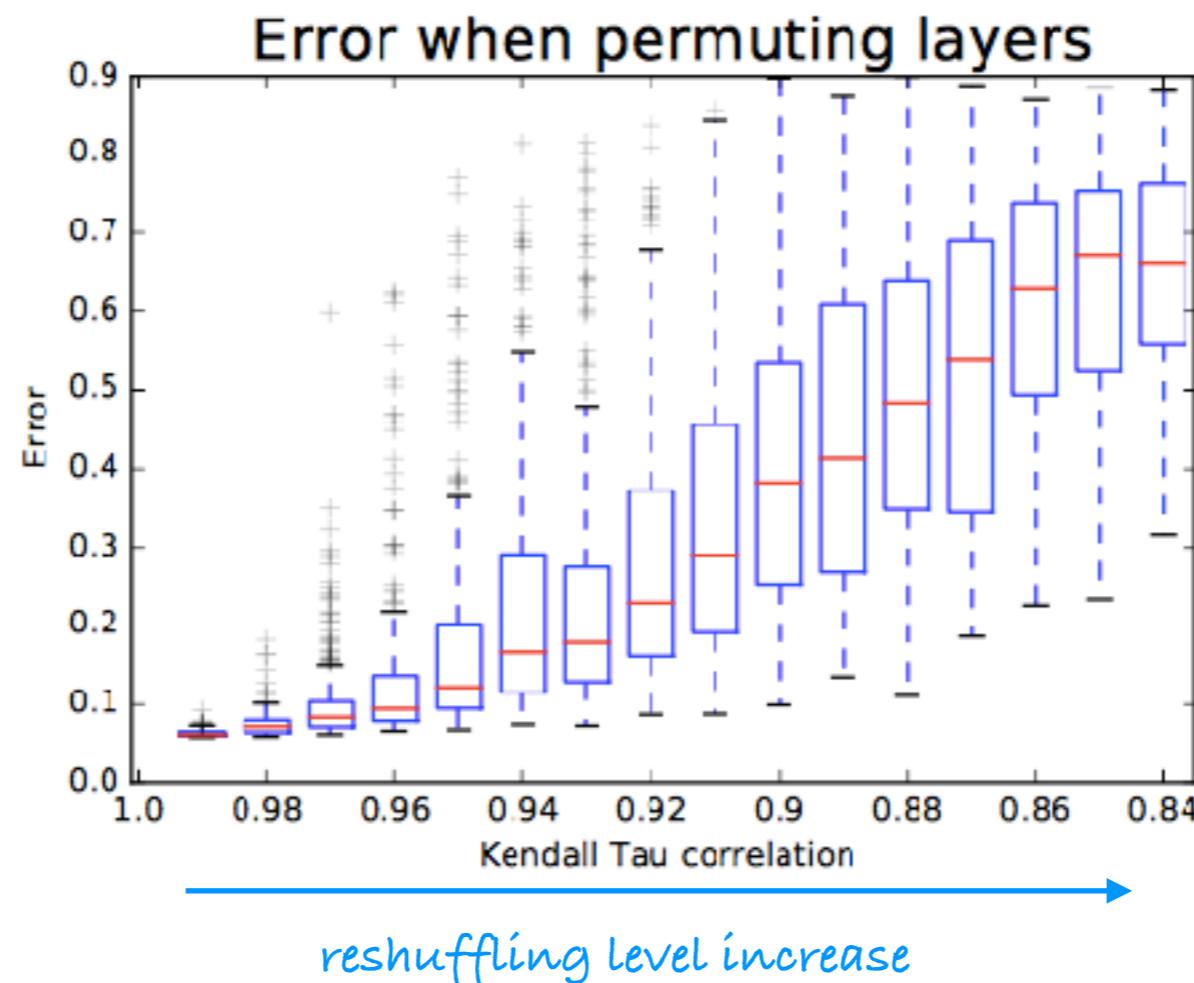
Swapping k pairs of ResNet blocks within the same resolution:

Moving high level  
transformations  
before low level ones

# Lesion Study

Swapping k pairs of ResNet blocks within the same resolution:

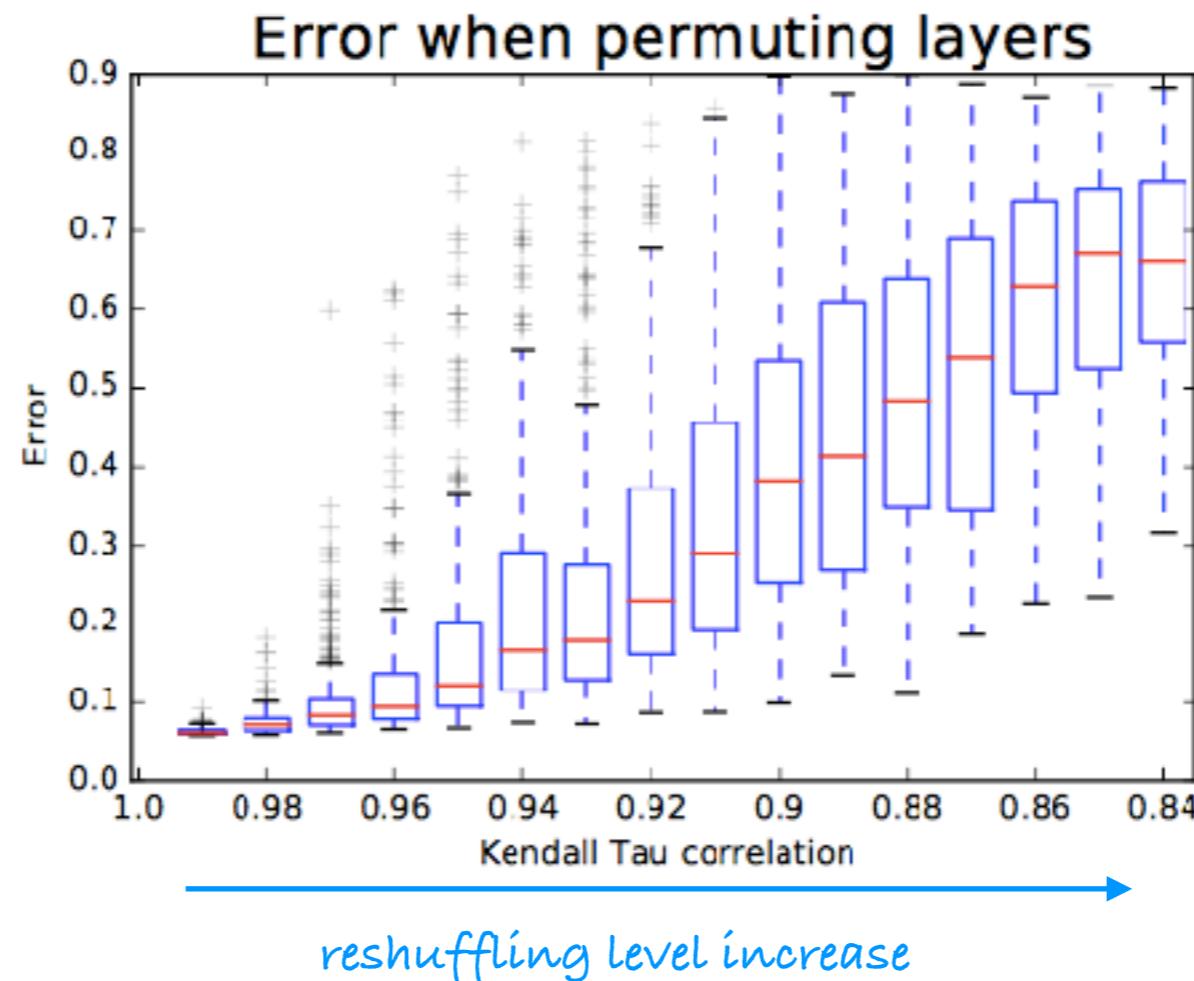
Moving high level  
transformations  
before low level ones



# Lesion Study

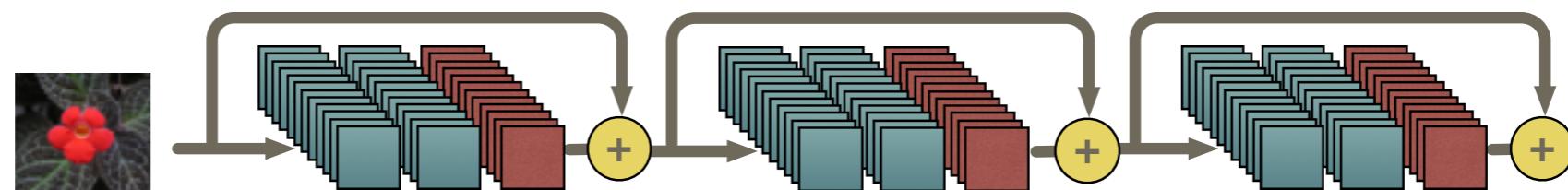
Swapping  $k$  pairs of ResNet blocks within the same resolution:

Moving high level transformations before low level ones

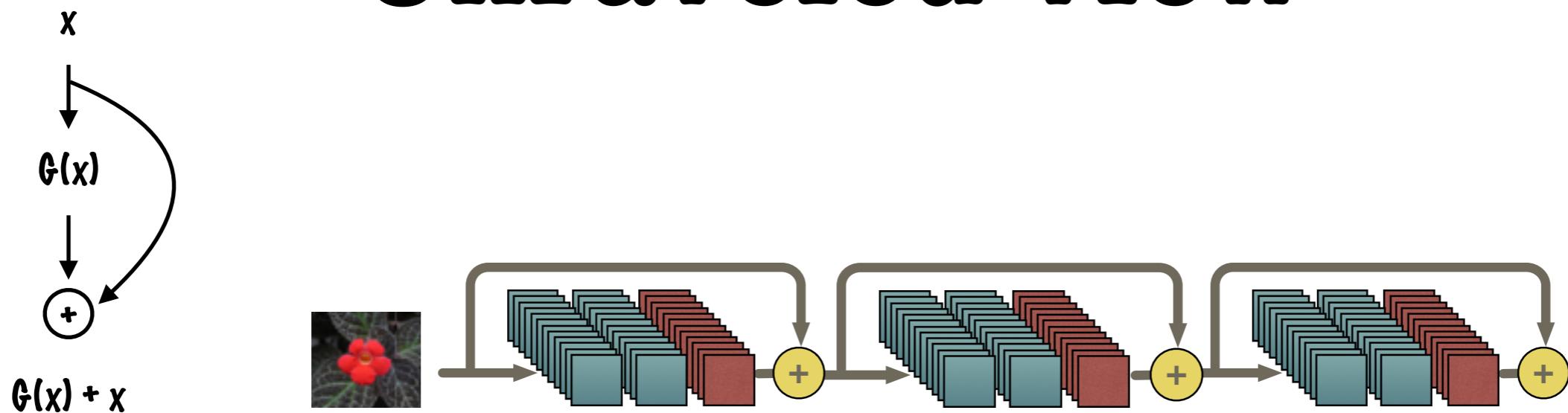


this suggests  
ResNets defy the  
classical  
representation view

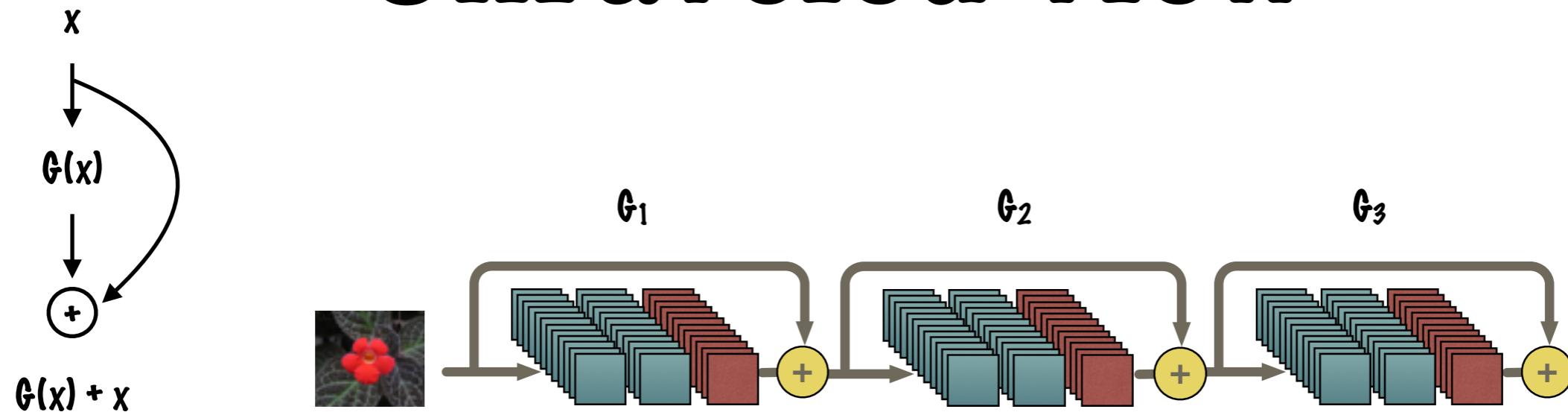
# Unraveled view



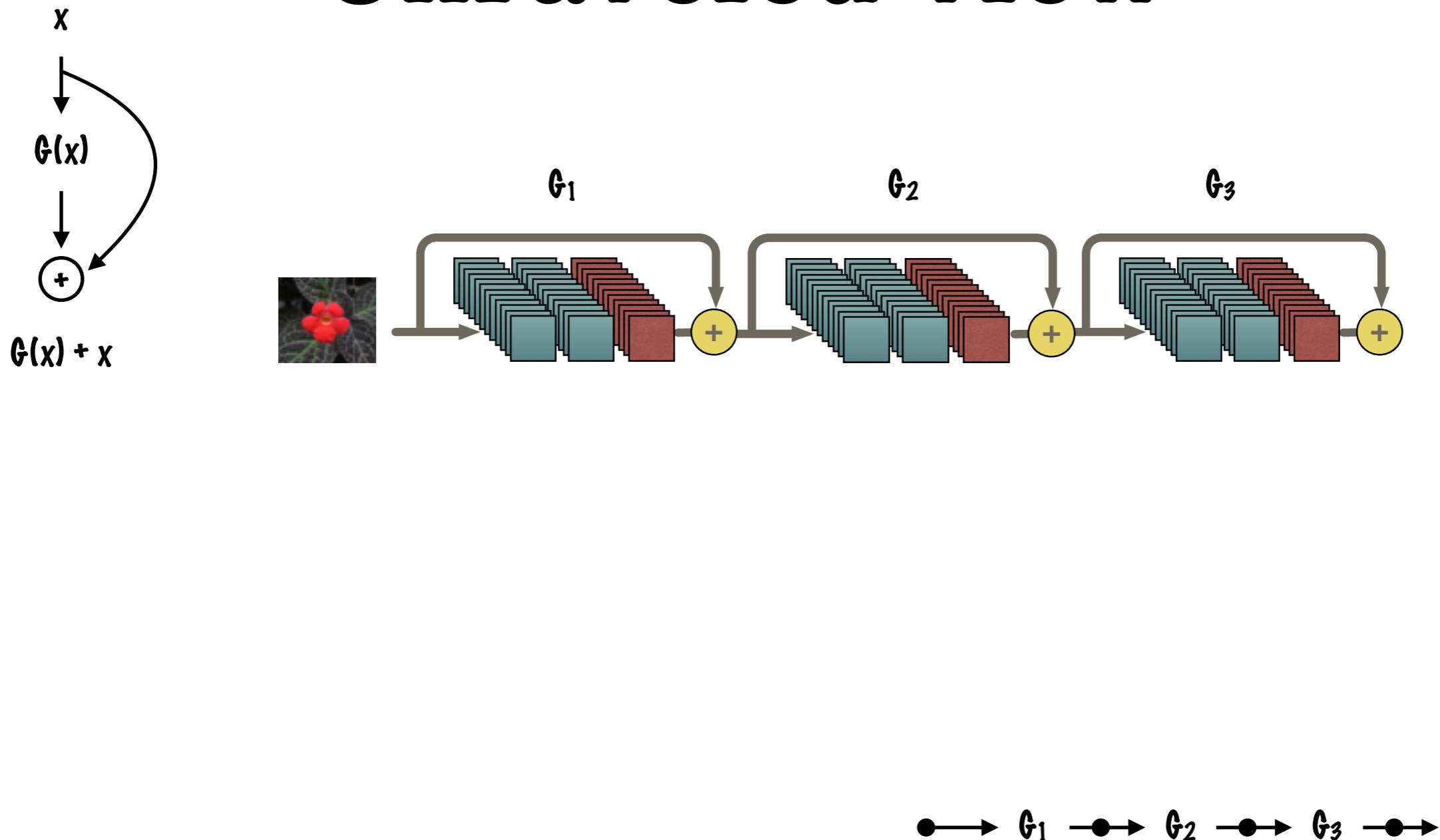
# Unraveled view



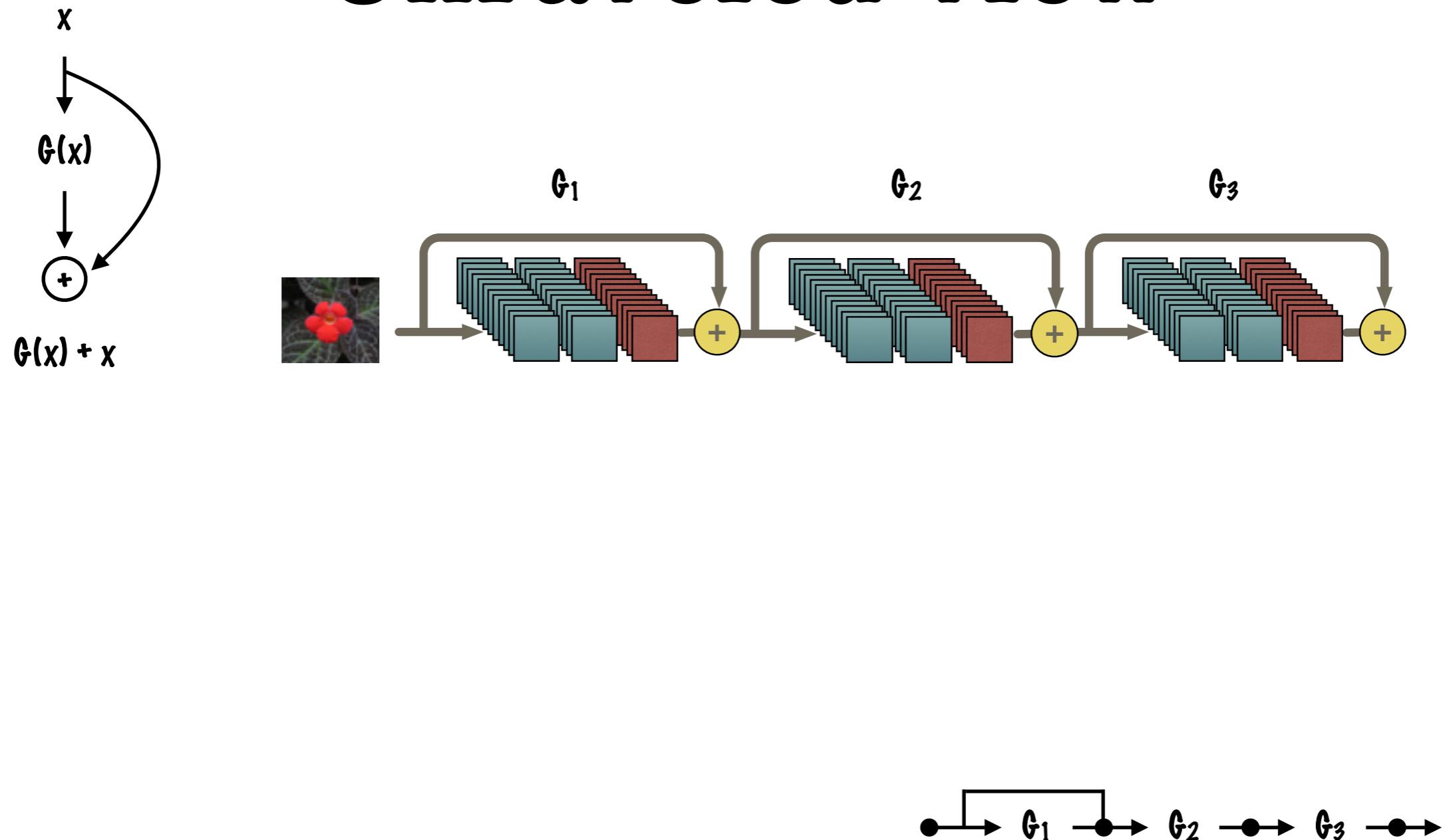
# Unraveled view



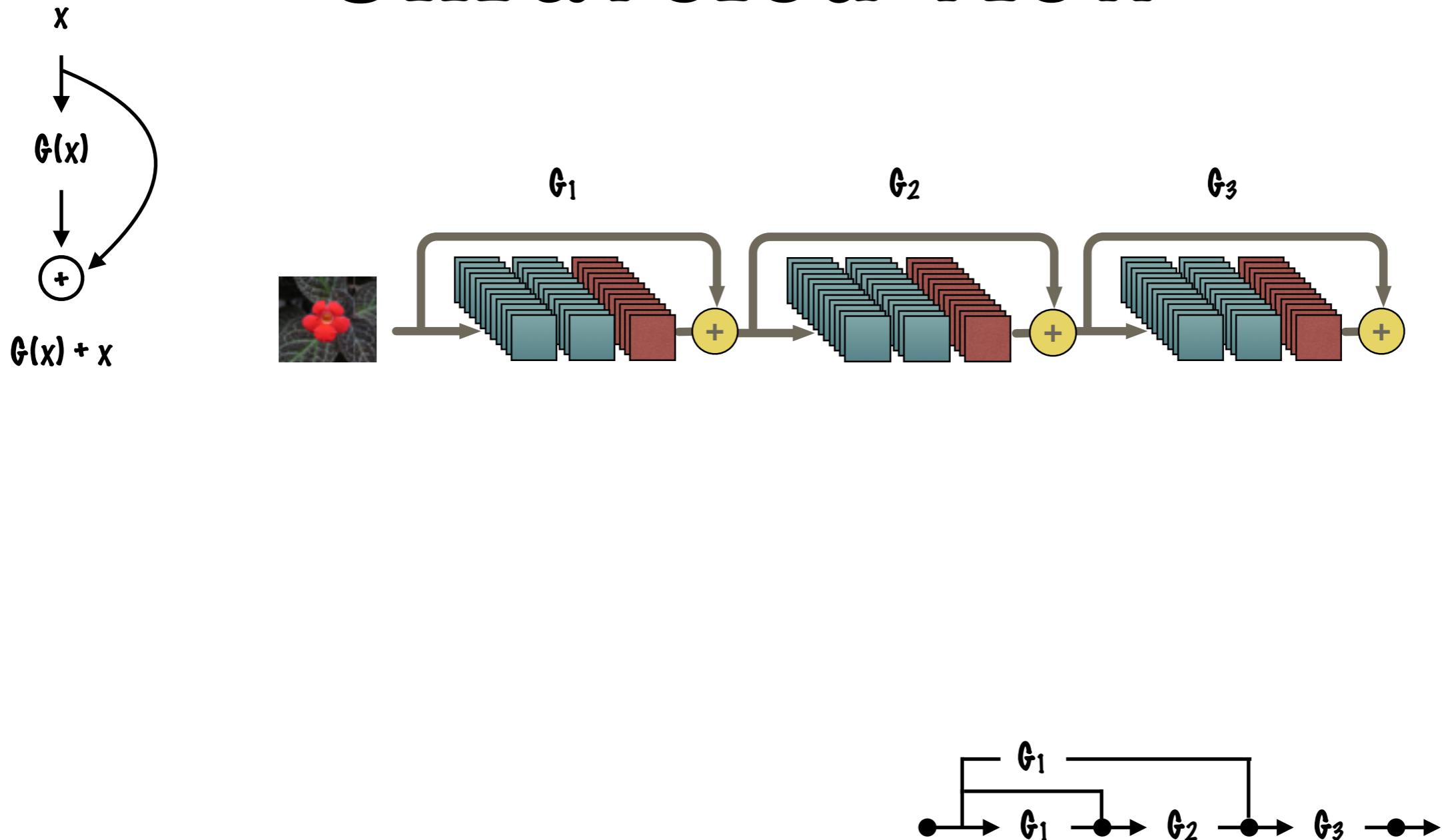
# Unraveled view



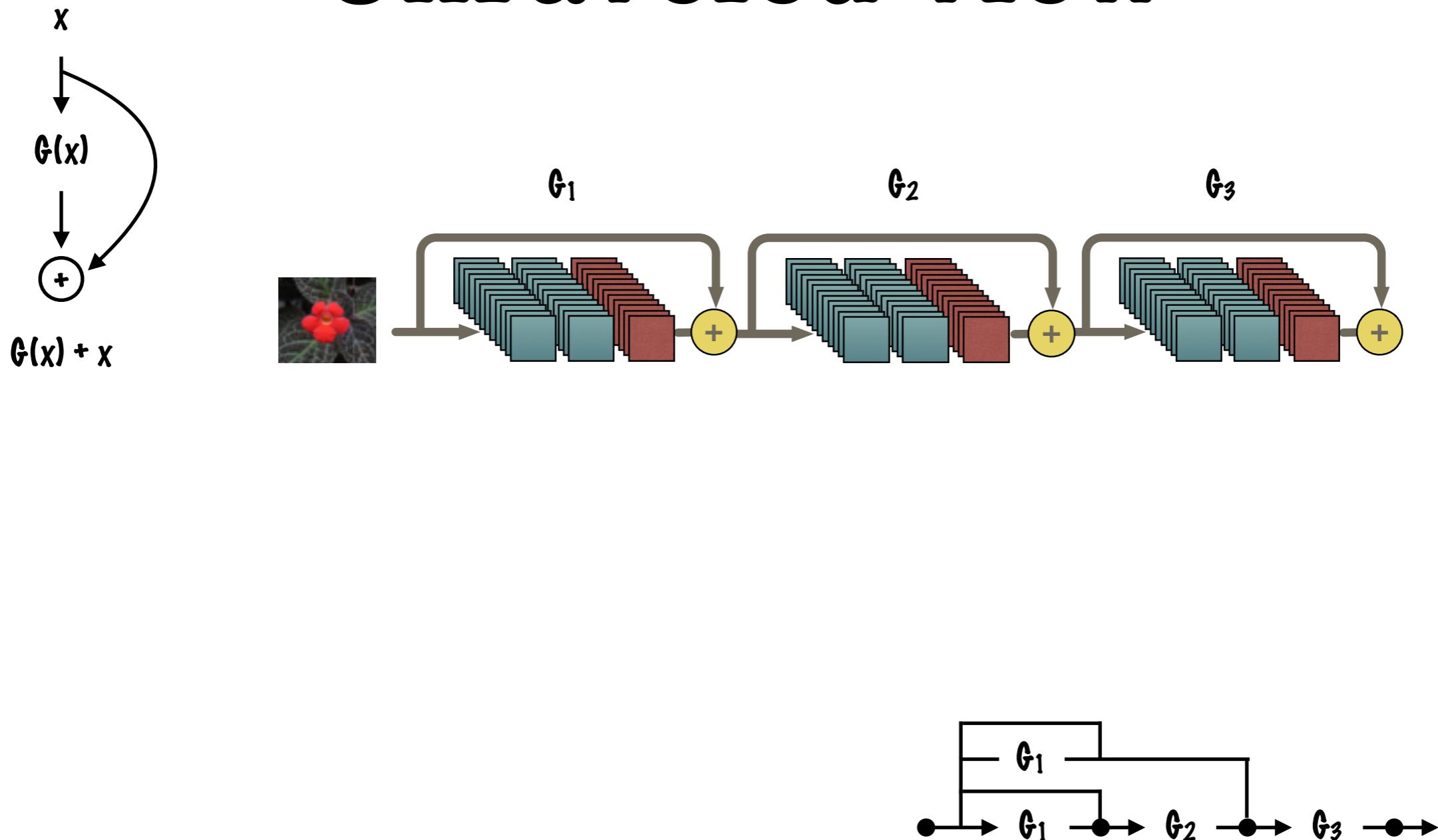
# Unraveled view



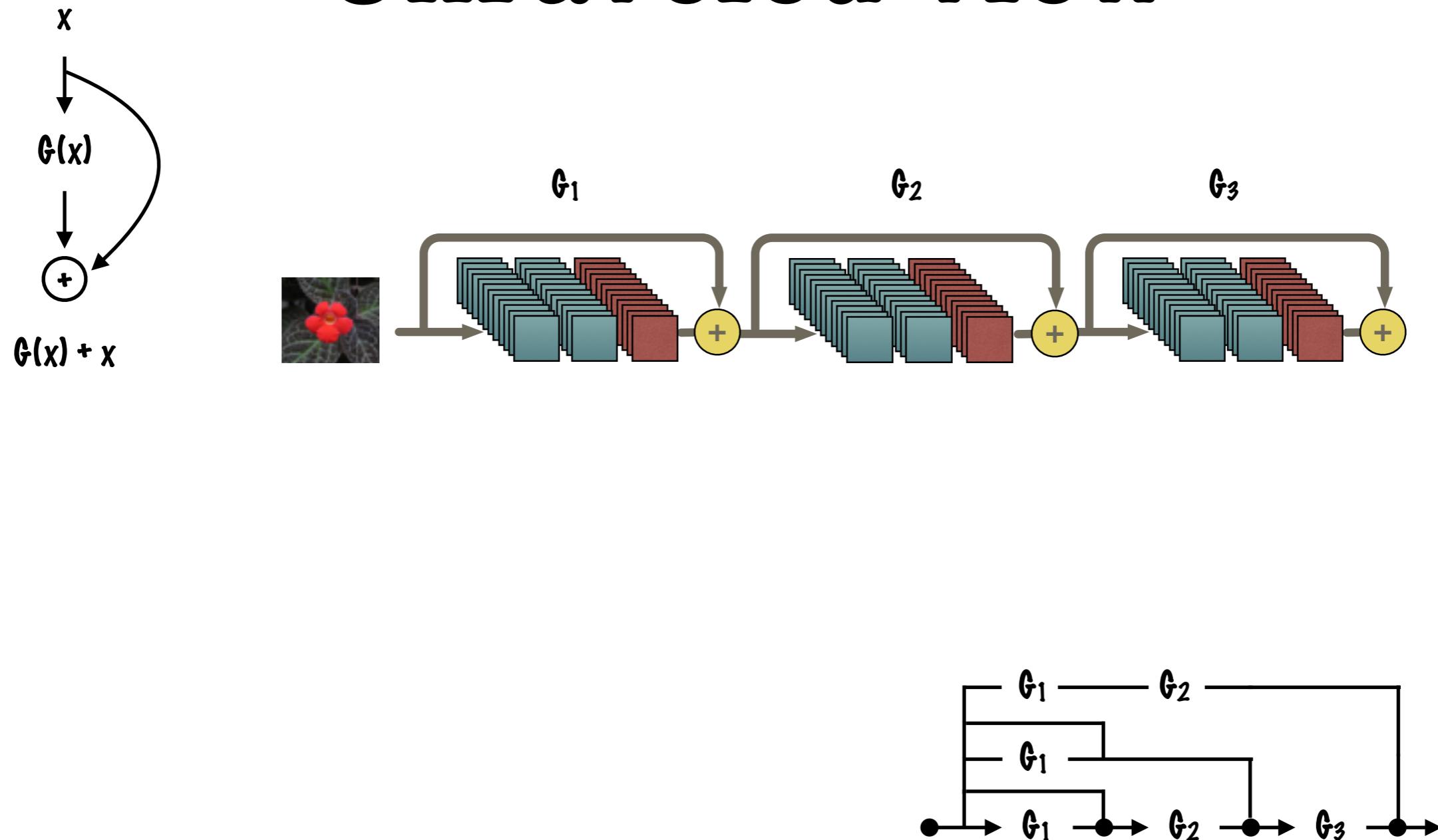
# Unraveled view



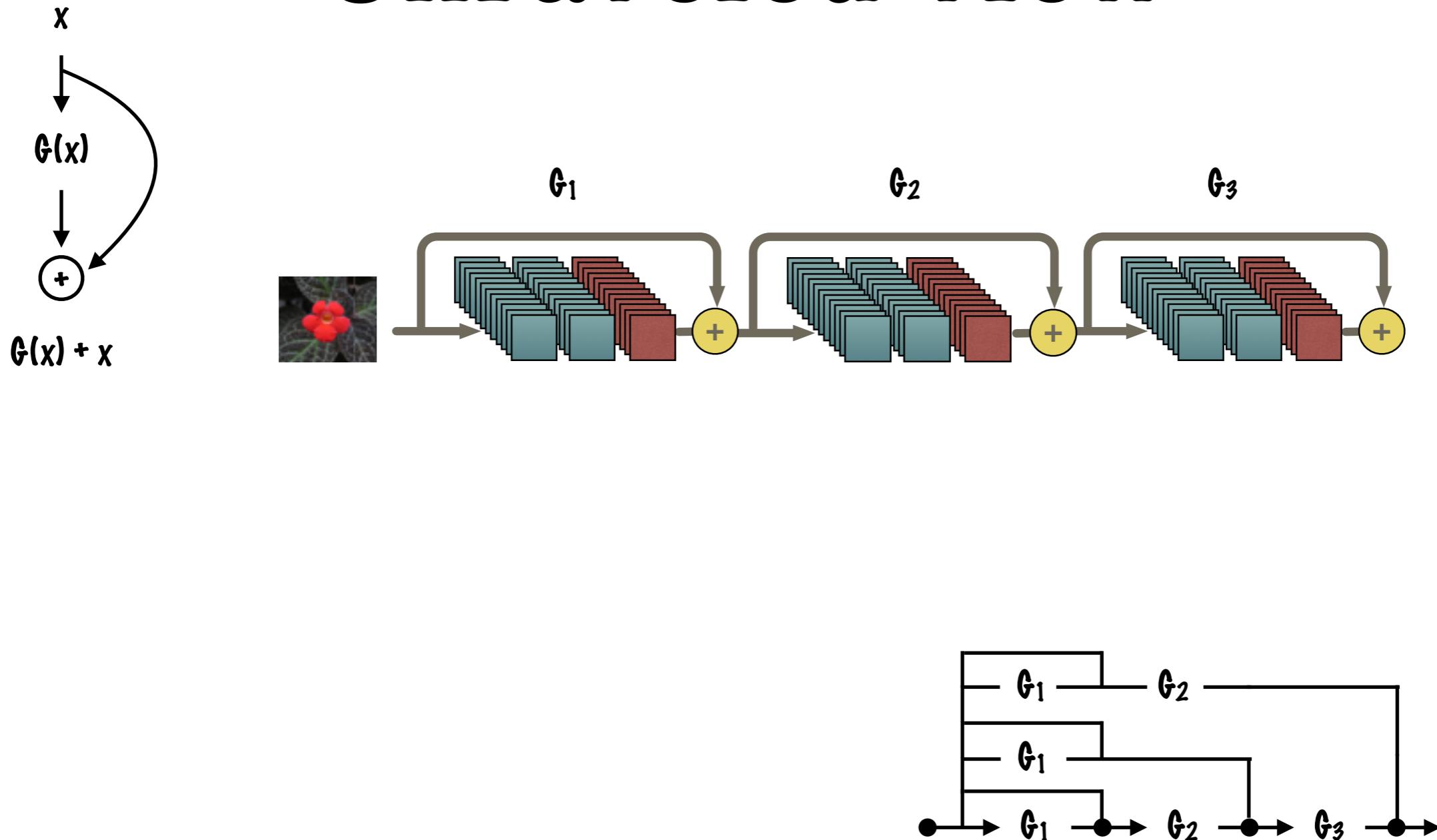
# Unraveled view



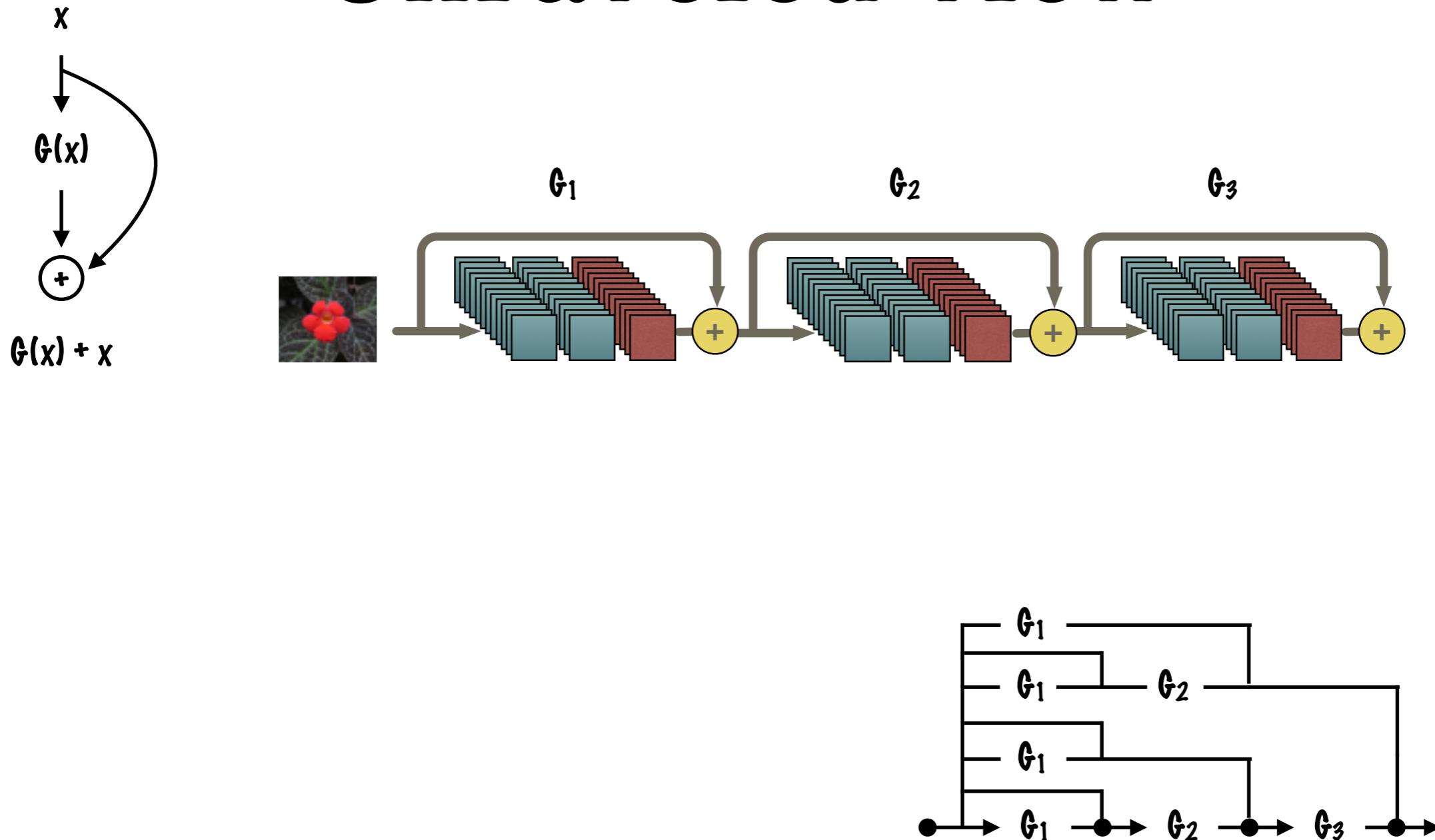
# Unraveled view



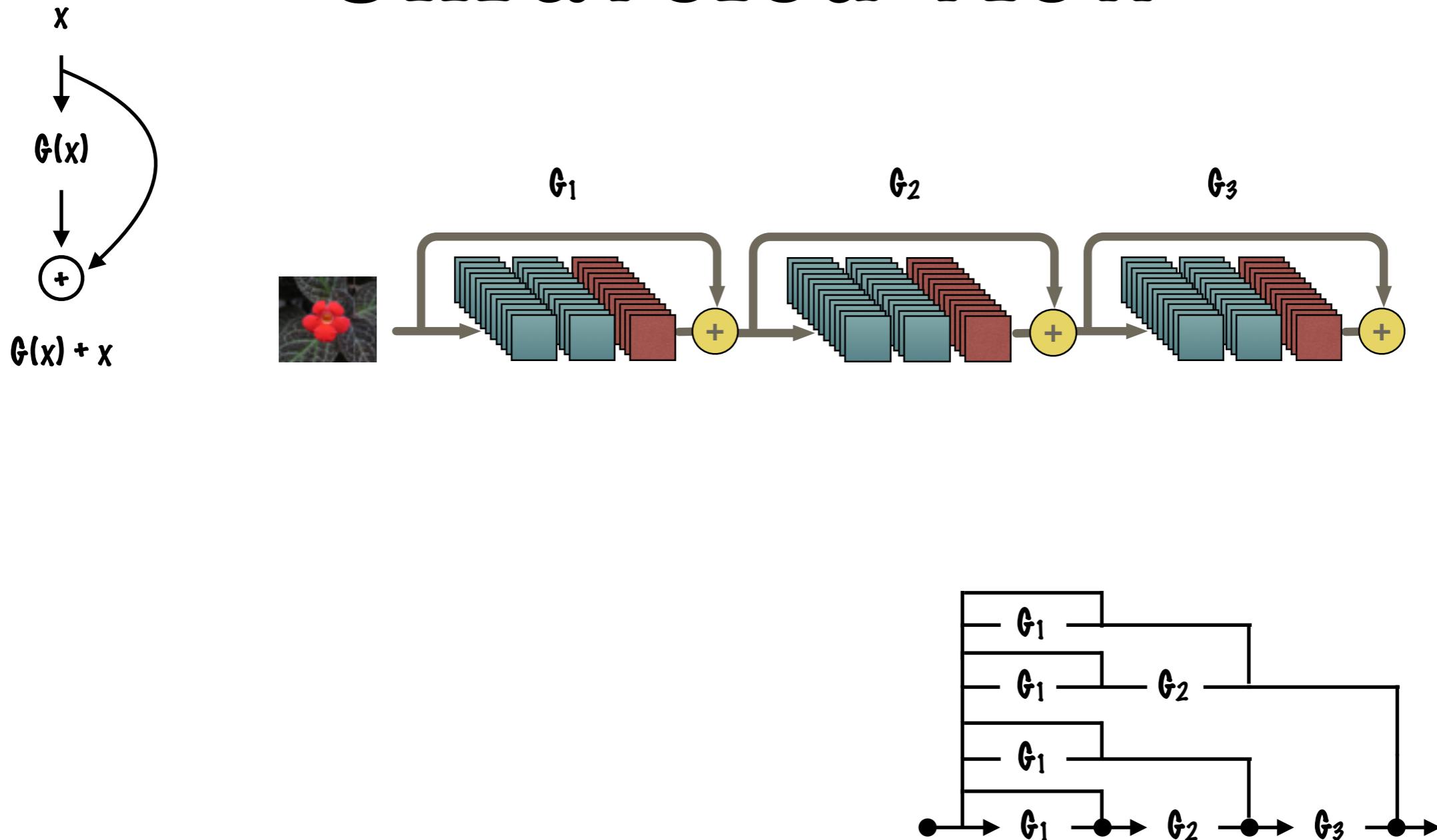
# Unraveled view



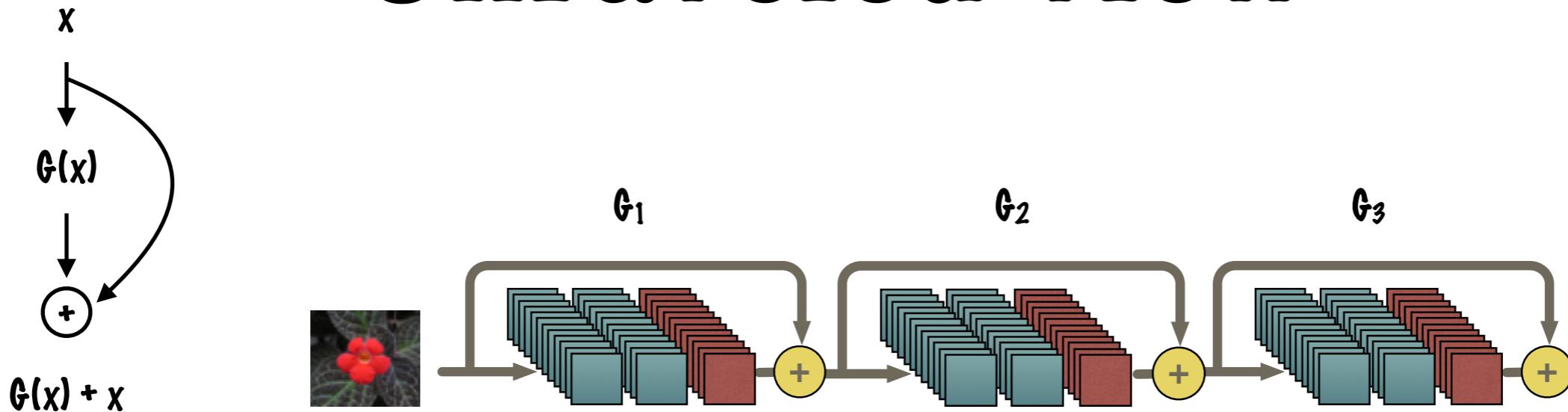
# Unraveled view



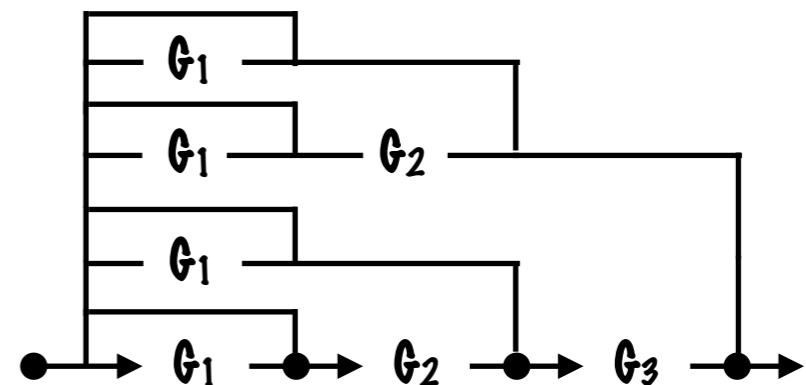
# Unraveled view



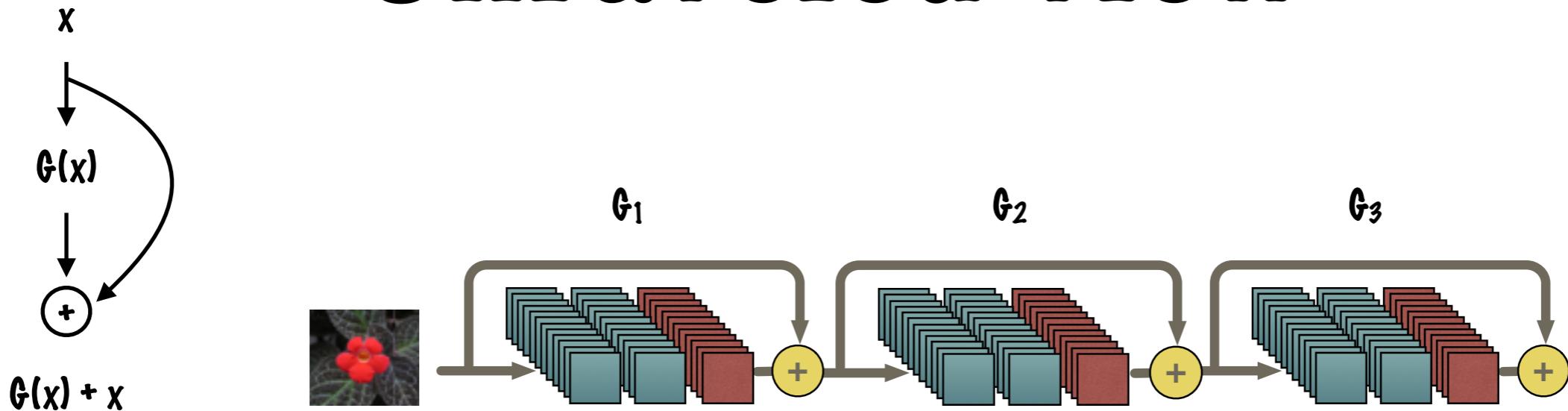
# Unraveled view



data flows along many paths

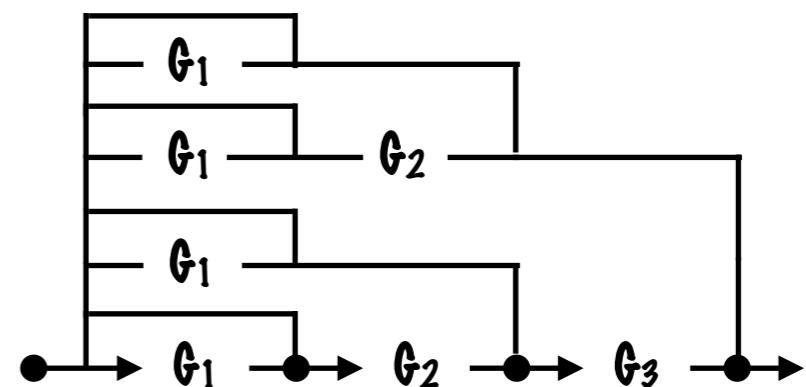


# Unraveled view

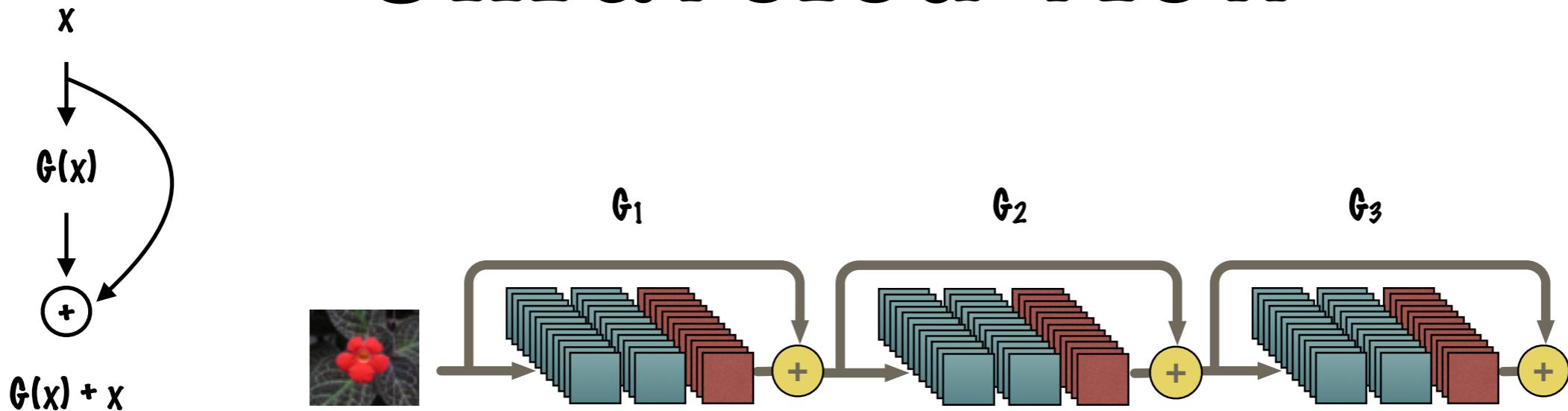


data flows along many paths

$2^n$  paths



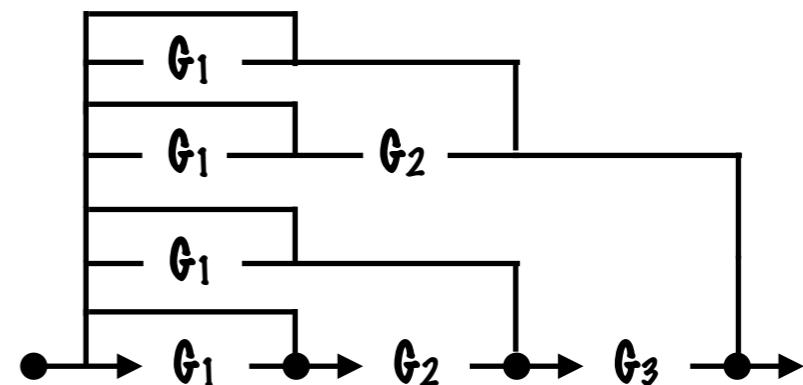
# Unraveled view



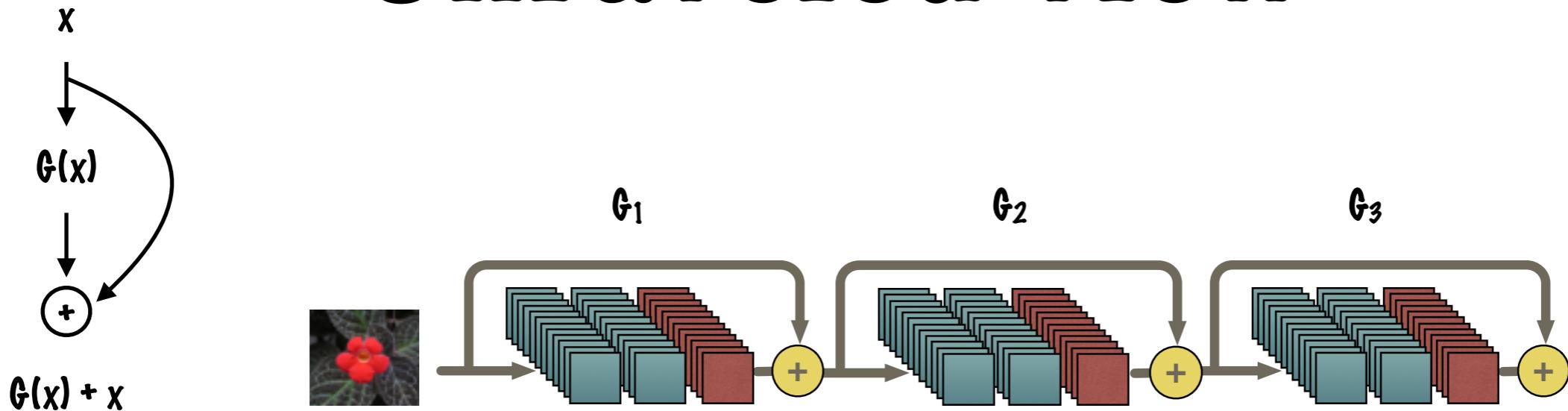
data flows along many paths

each block is fed data from  
different path configurations

$2^n$  paths



# Unraveled view

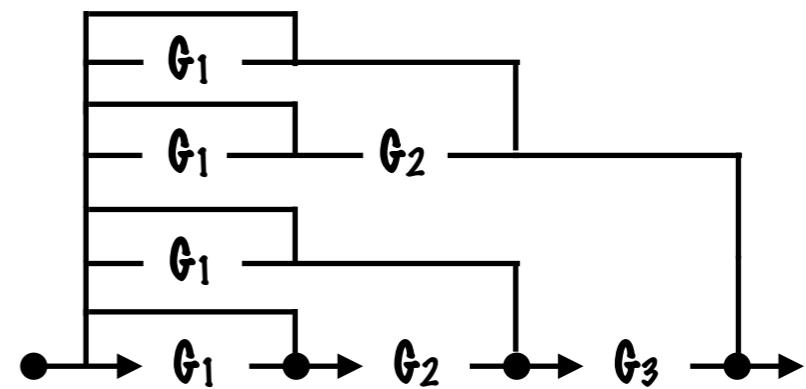


data flows along many paths

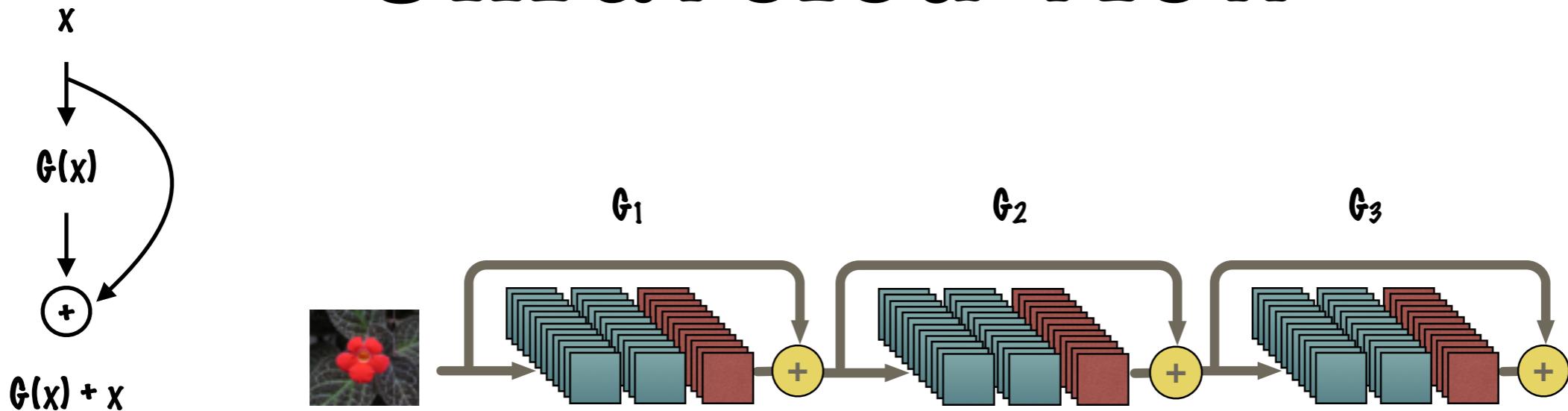
each block is fed data from different path configurations

path have varying length and go through a different subset of layers

$2^n$  paths



# Unraveled view



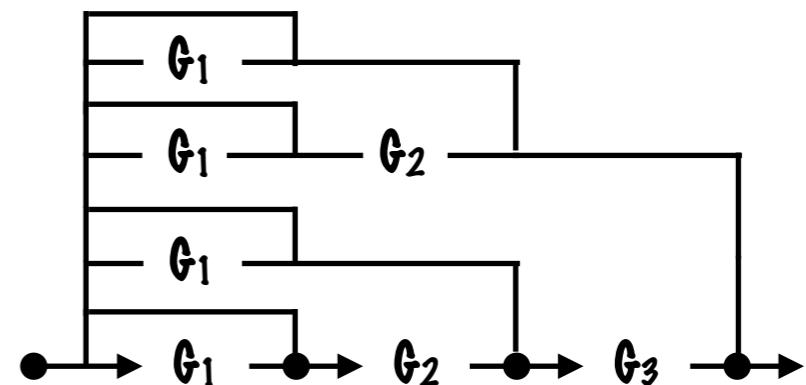
data flows along many paths

each block is fed data from  
different path configurations

path have varying length and go through a  
different subset of layers

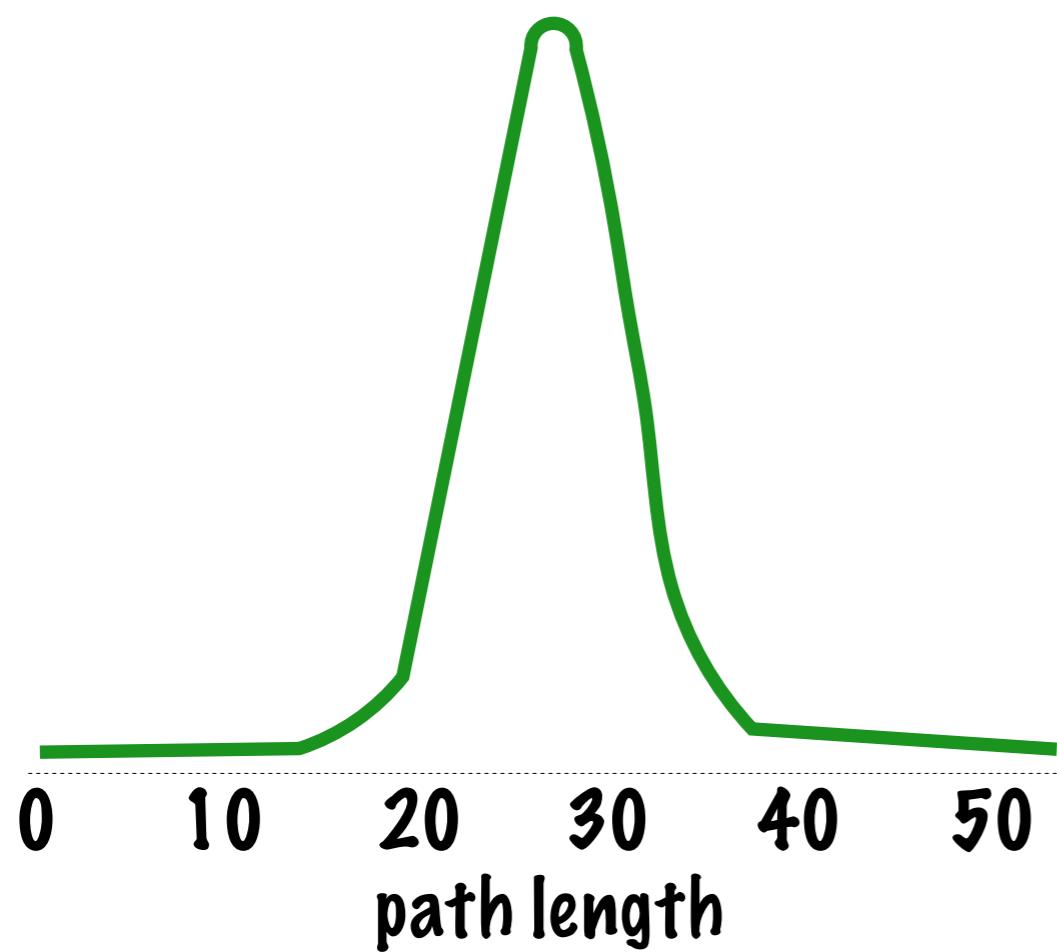
ResNets as ensembles of  
varying length networks

2<sup>n</sup> paths



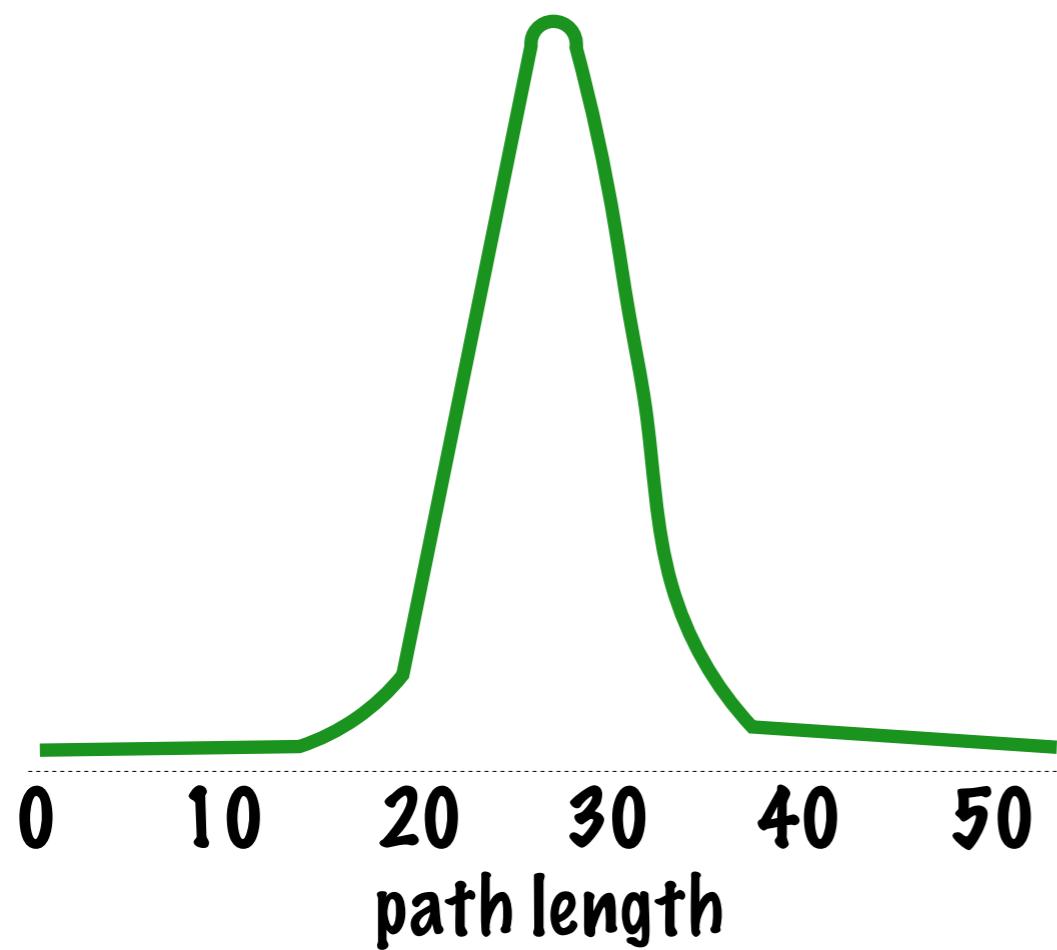
# Gradient magnitude & path length

distribution of path length

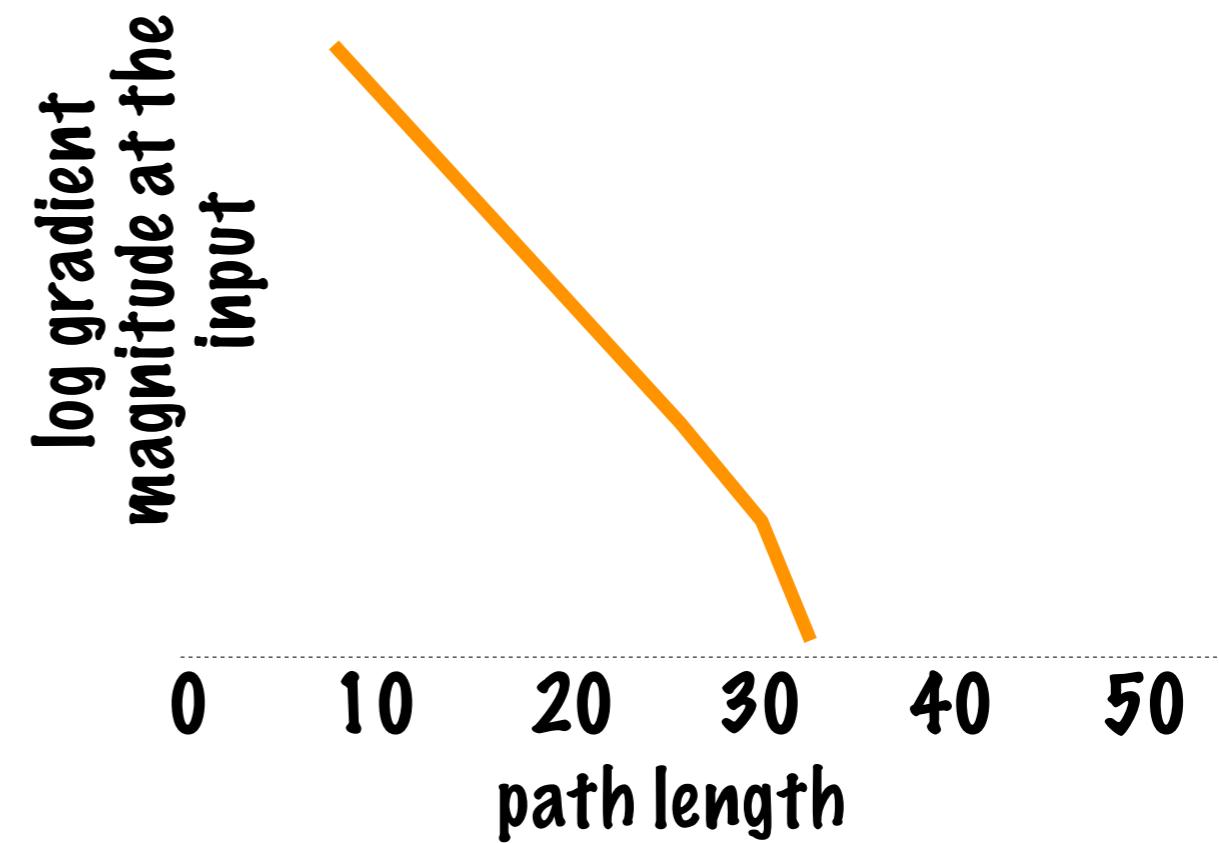


# Gradient magnitude & path length

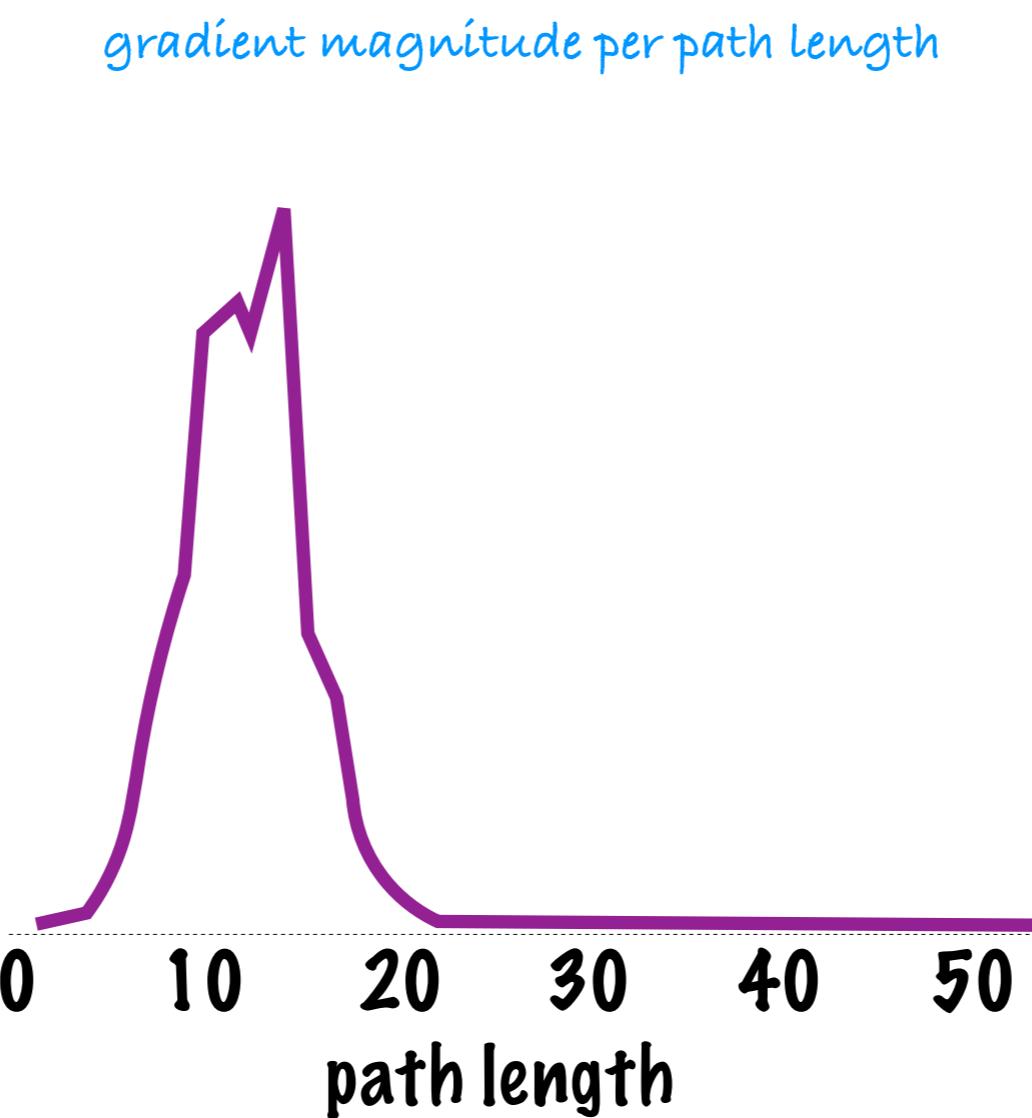
distribution of path length



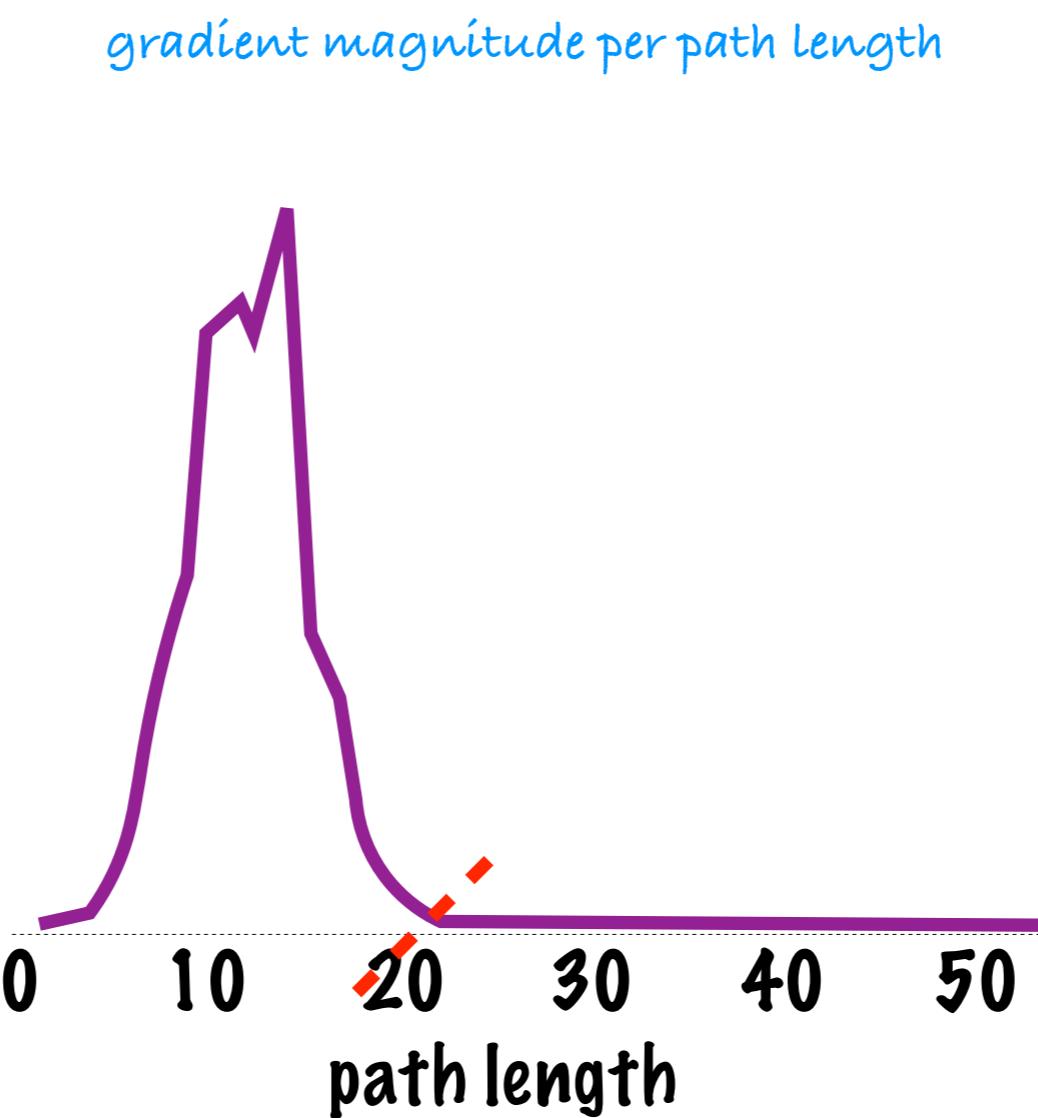
gradient magnitude per path length



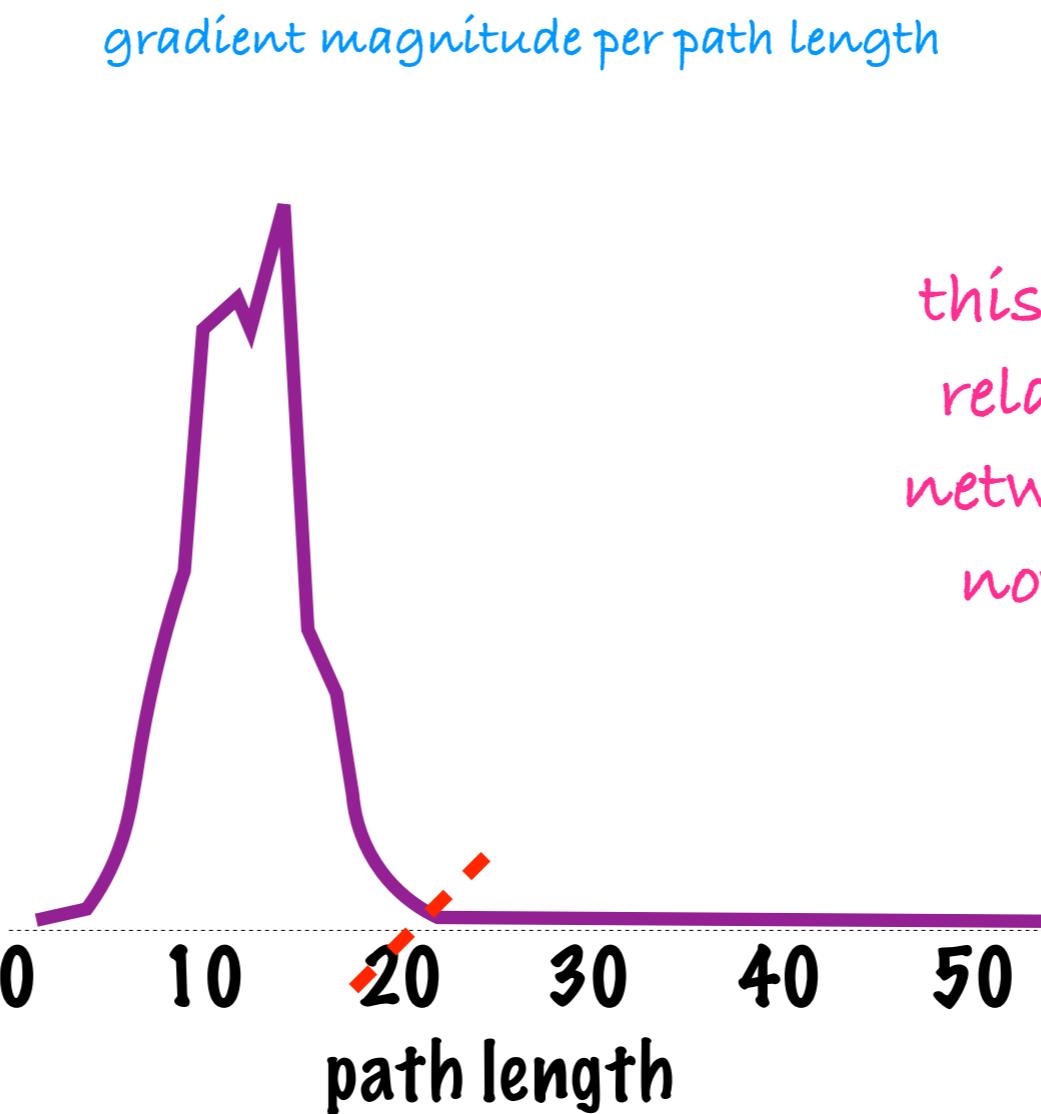
# Gradient magnitude & path length



# Gradient magnitude & path length

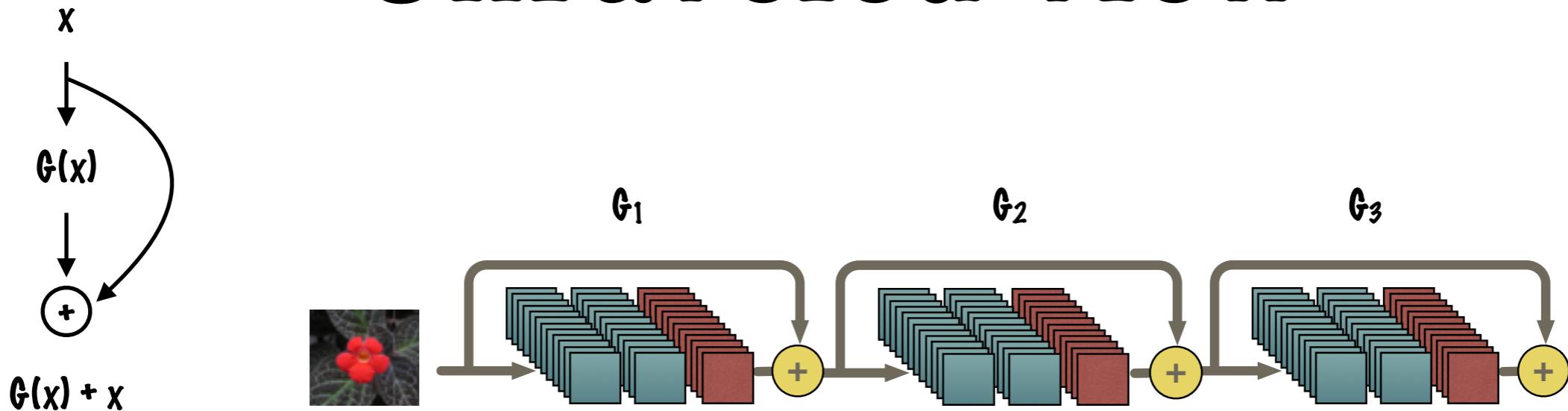


# Gradient magnitude & path length



this suggests only  
relatively shallow  
networks contribute  
noticeably to the  
gradient

# Unraveled view



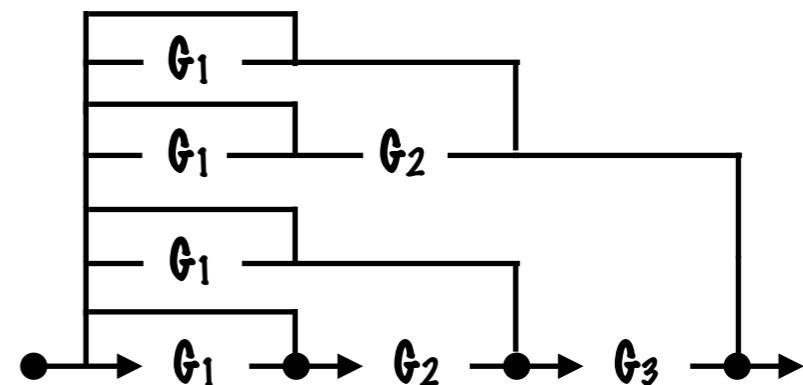
data flows along many paths

each block is fed data from  
different path configurations

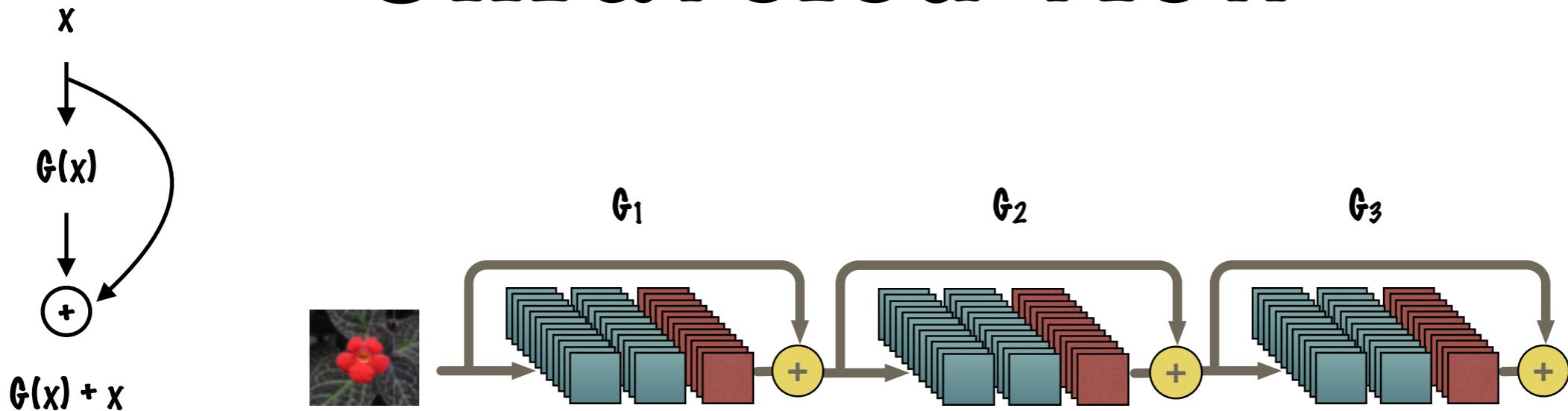
path have varying length and go through a  
different subset of layers

ResNets as ensembles of  
varying length networks

$2^n$  paths



# Unraveled view



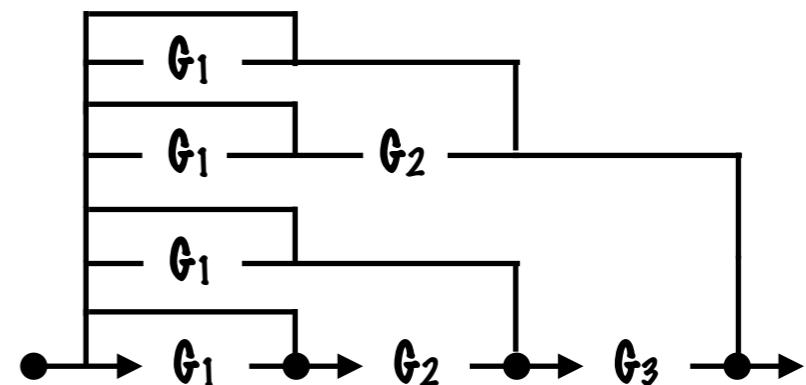
data flows along many paths

each block is fed data from different path configurations

path have varying length and go through a different subset of layers

ResNets as ensembles of relatively shallow networks

$2^n$  paths

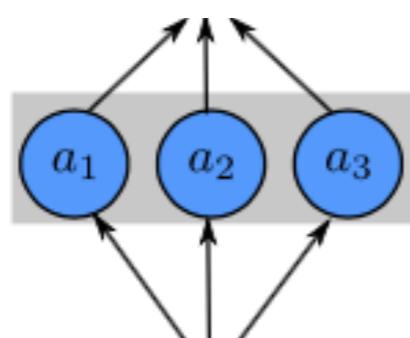




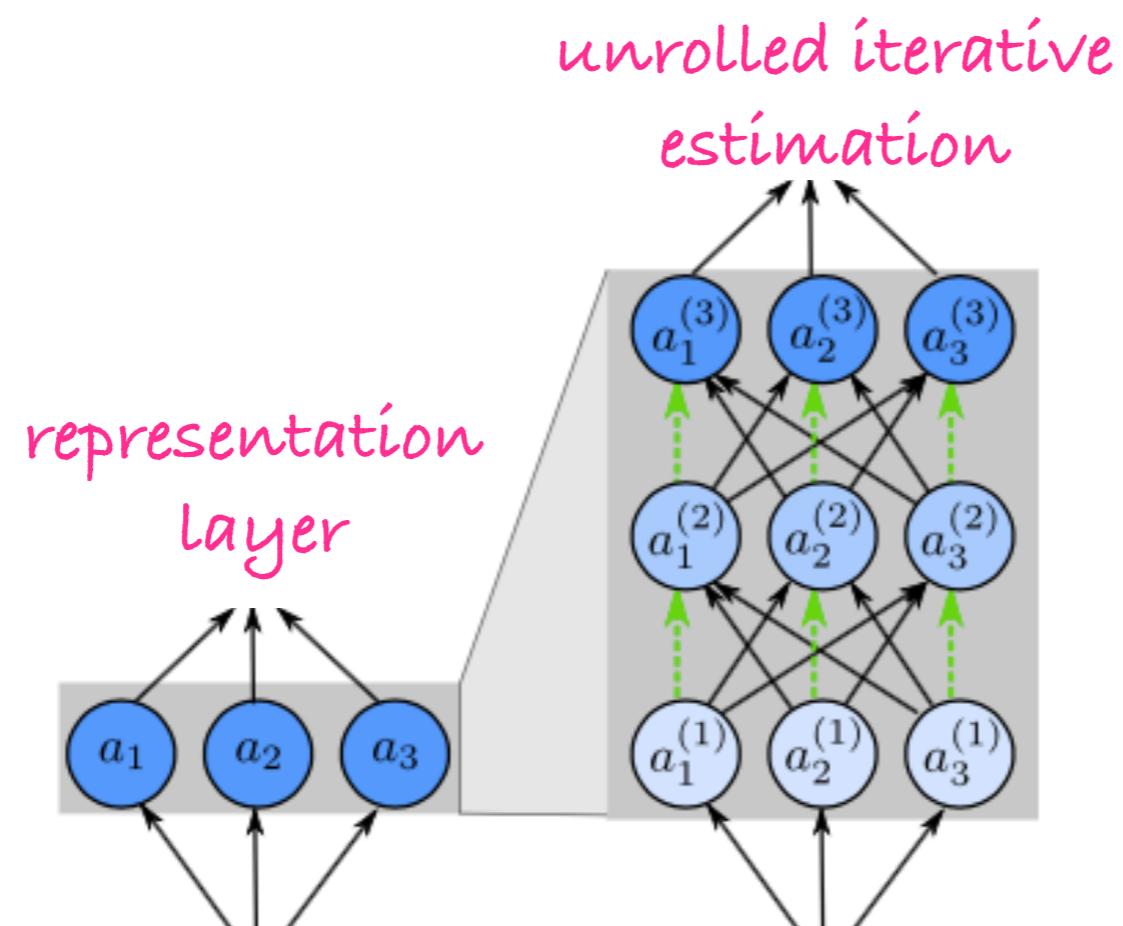
but... how can a layer in a subnetwork successfully operate with changing input representations?

# Unrolled iterative estimation view

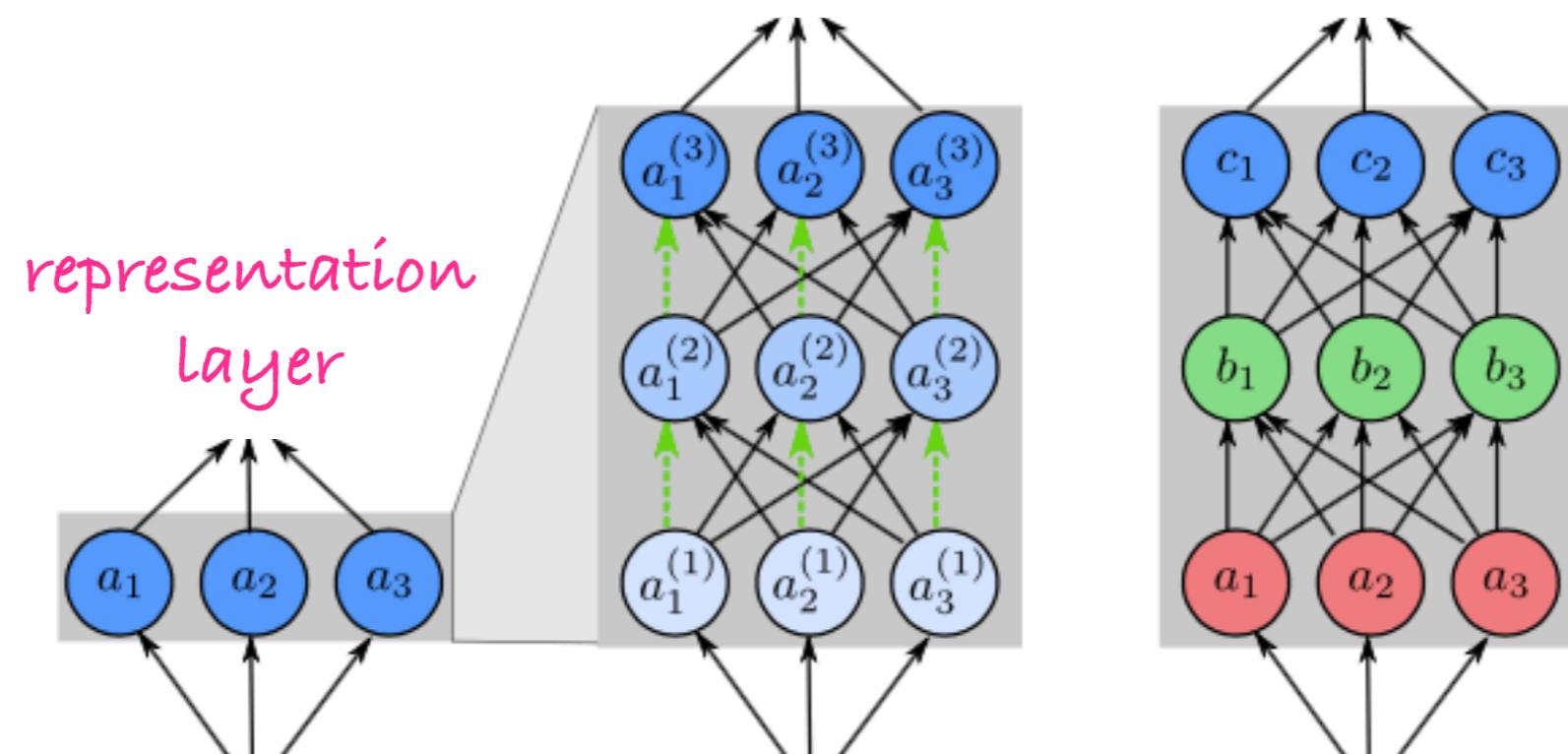
representation  
layer



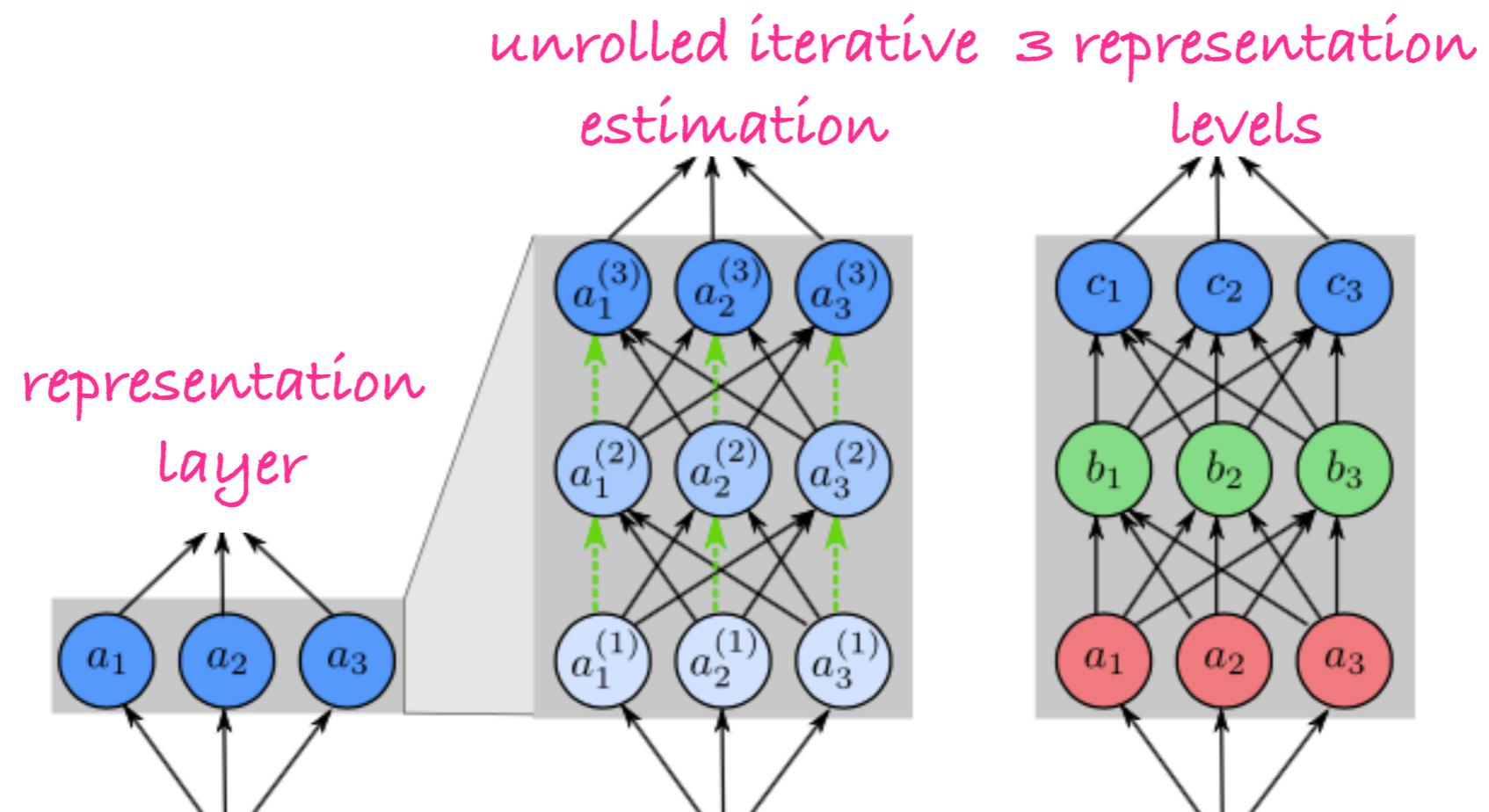
# Unrolled iterative estimation view



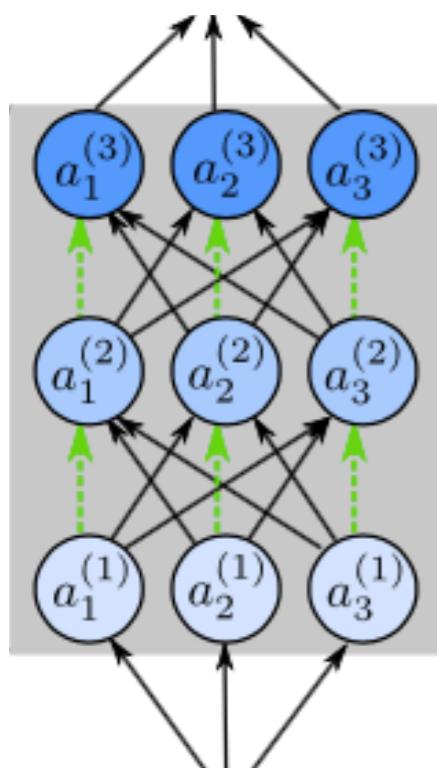
# Unrolled iterative estimation view



# Unrolled iterative estimation view

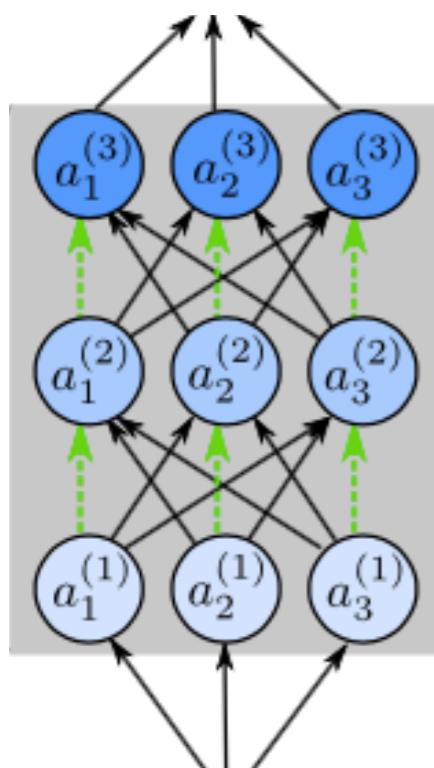


# Unrolled iterative estimation view



# Unrolled iterative estimation view

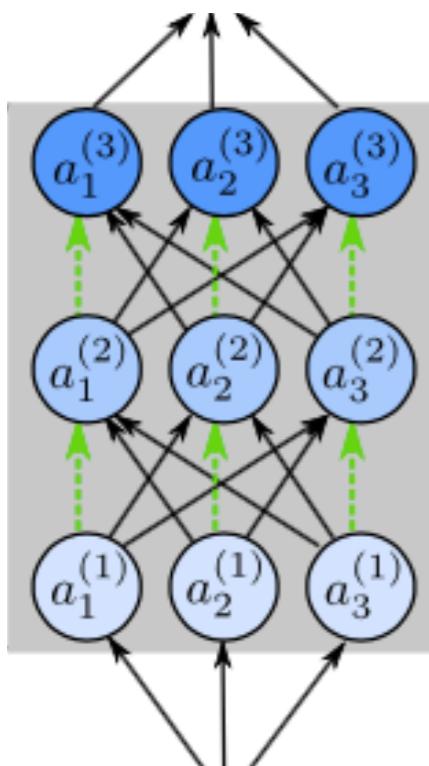
a group of successive layers iteratively refine their input



# Unrolled iterative estimation view

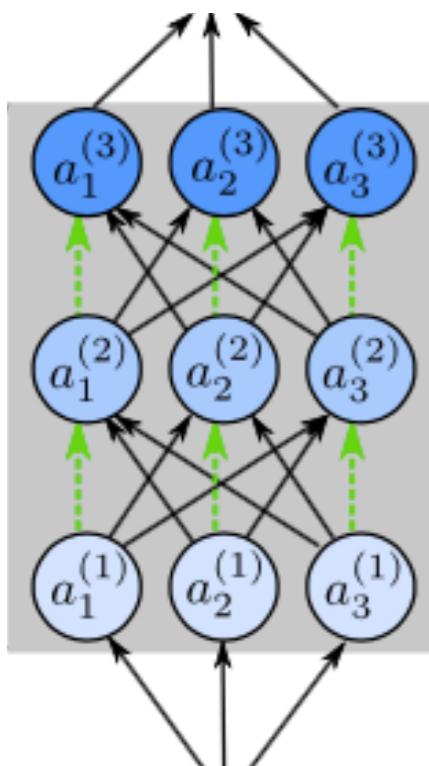
a group of successive layers iteratively refine their input

the group of layers preserve feature identity

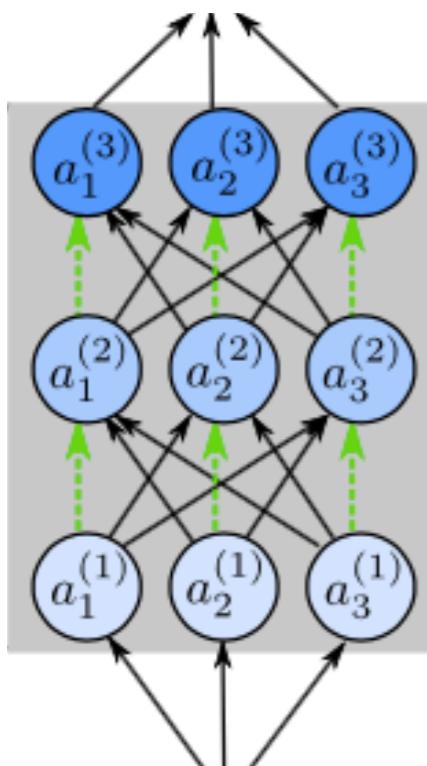


# Unrolled iterative estimation view

a group of successive layers iteratively refine their input  
the group of layers preserve feature identity  
new representation levels come with dimensionality change



# Unrolled iterative estimation view



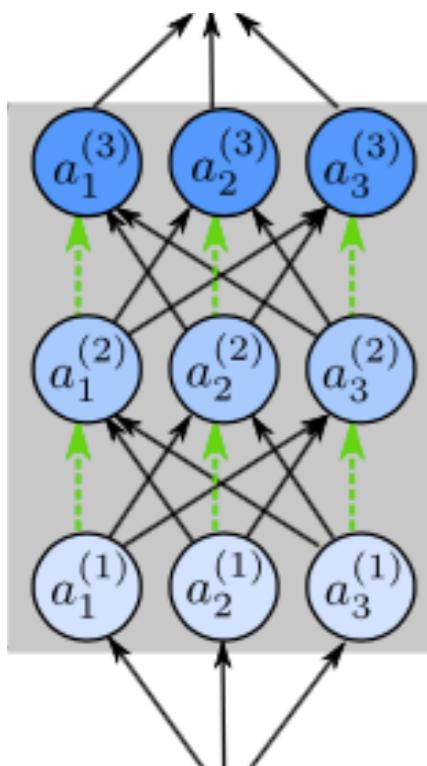
a group of successive layers iteratively refine their input

the group of layers preserve feature identity

new representation levels come with dimensionality  
change

Dropping layers: mild effects, since it does not change  
the representation level fed to the next level, only its  
quality

# Unrolled iterative estimation view



a group of successive layers iteratively refine their input

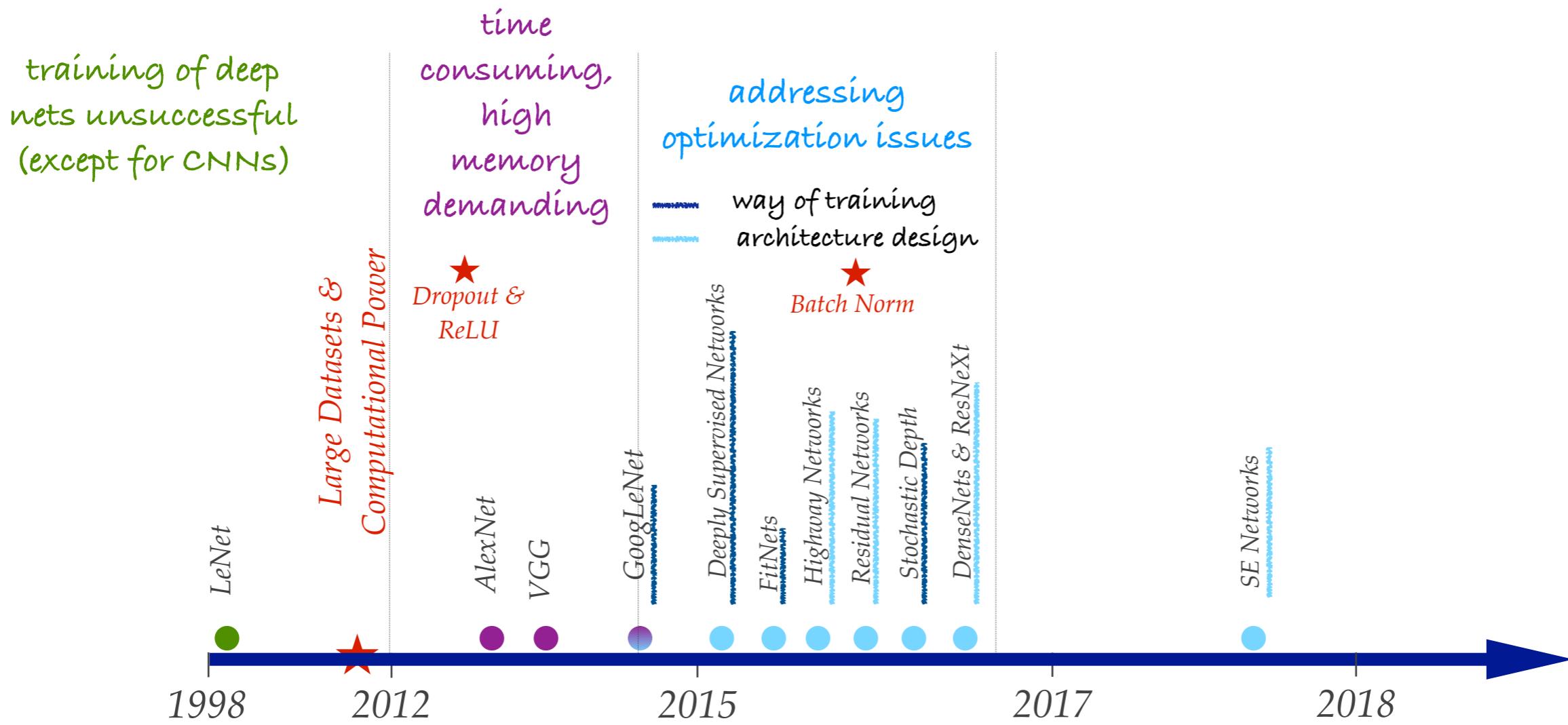
the group of layers preserve feature identity

new representation levels come with dimensionality  
change

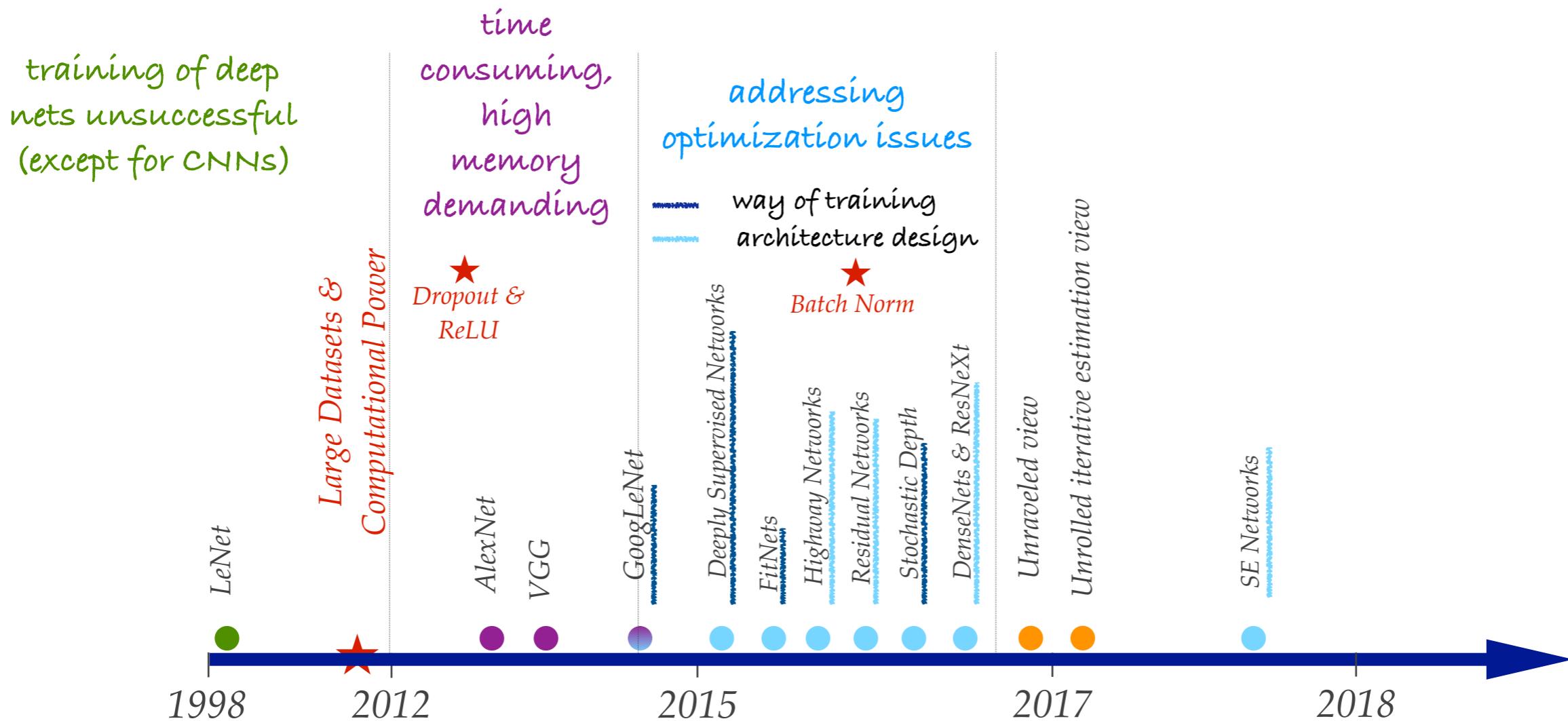
Dropping layers: mild effects, since it does not change  
the representation level fed to the next level, only its  
quality

Swapping layers: all layers within the same stage  
work with the same input and output representation

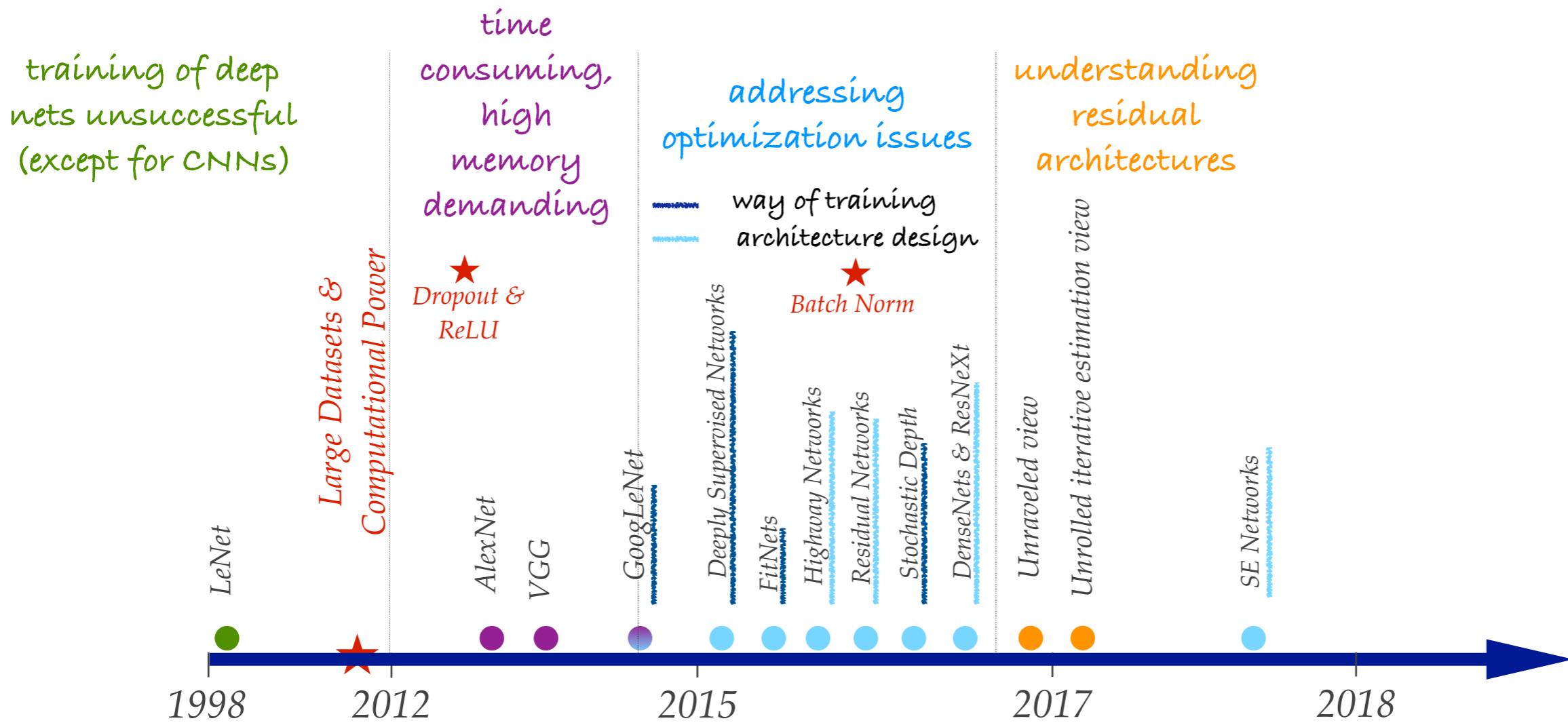
# Wrap Up



# Wrap Up



# Wrap Up



# **Neural Architecture Search**

# Neural Architecture Search

We do not trust ourselves to model ML solutions:

# Neural Architecture Search

We do not trust ourselves to model ML solutions:

train solution on data

# Neural Architecture Search

We do not trust ourselves to model ML solutions:

train solution on data

Do we trust our ability to design architectures?

# Neural Architecture Search

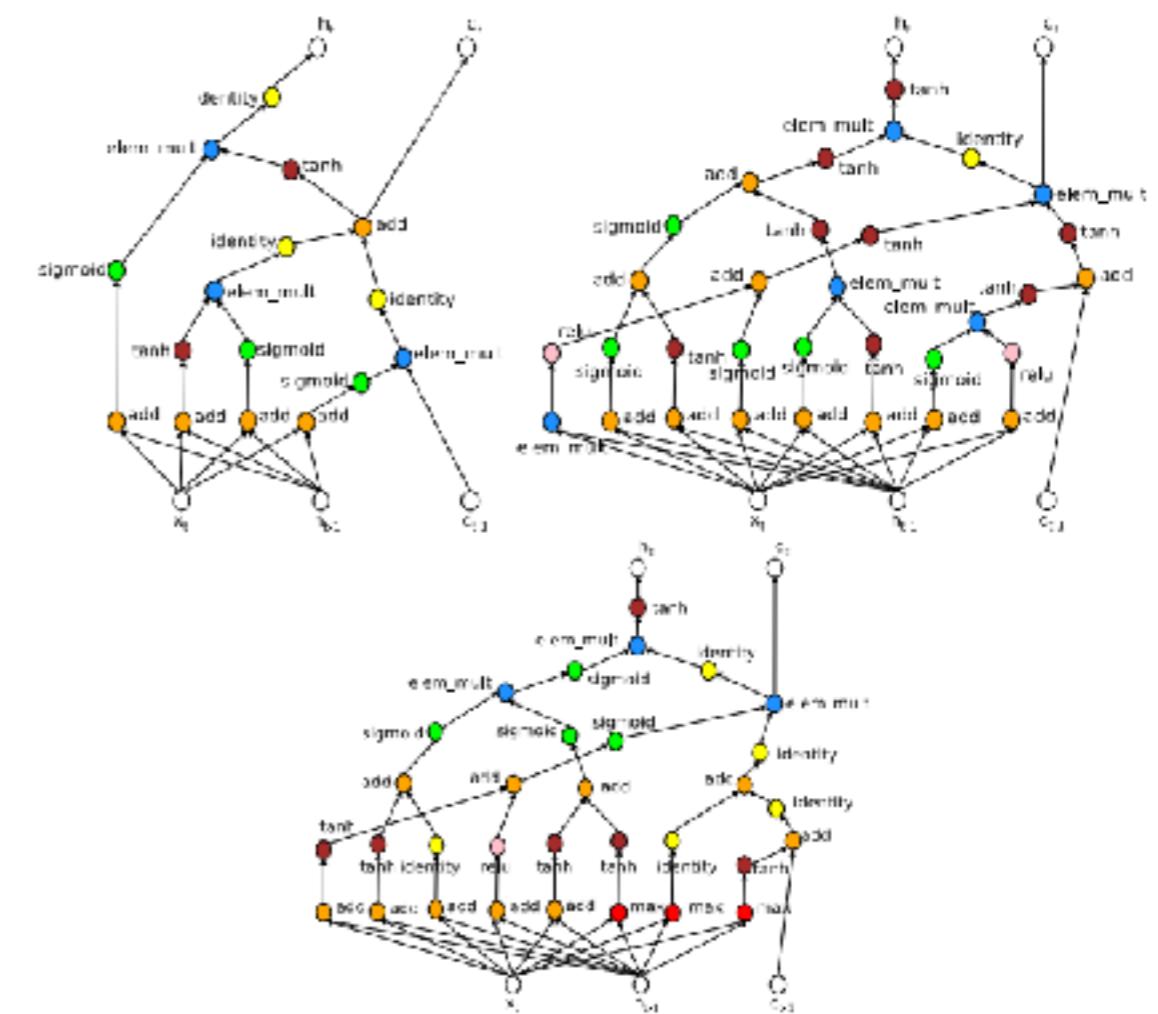
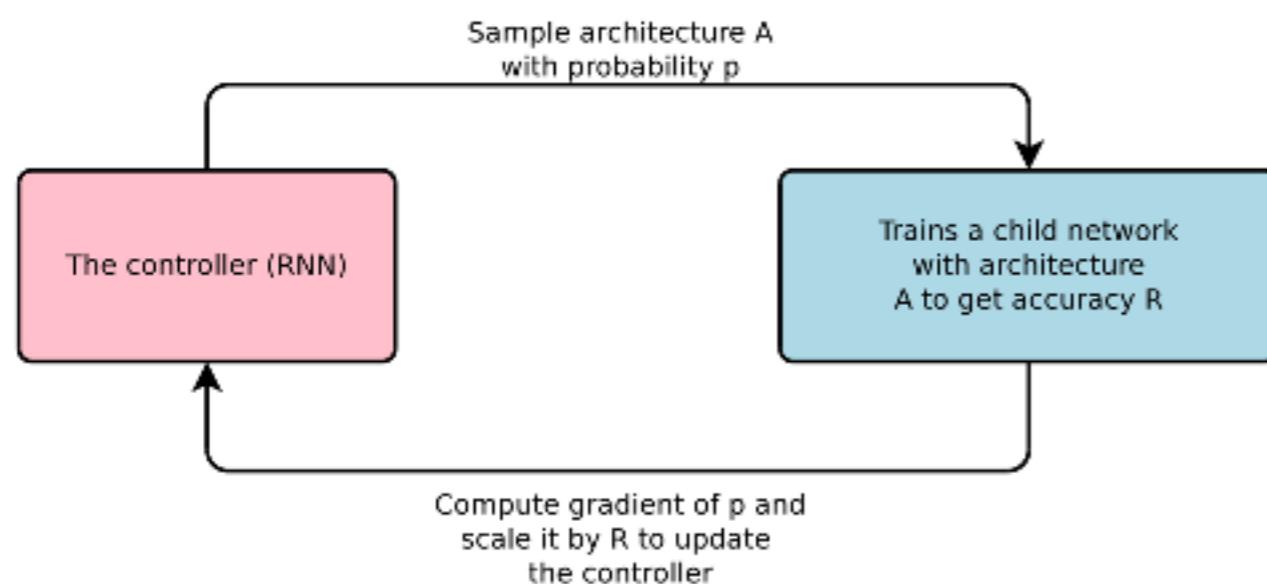
We do not trust ourselves to model ML solutions:

train solution on data

Do we trust our ability to design architectures?

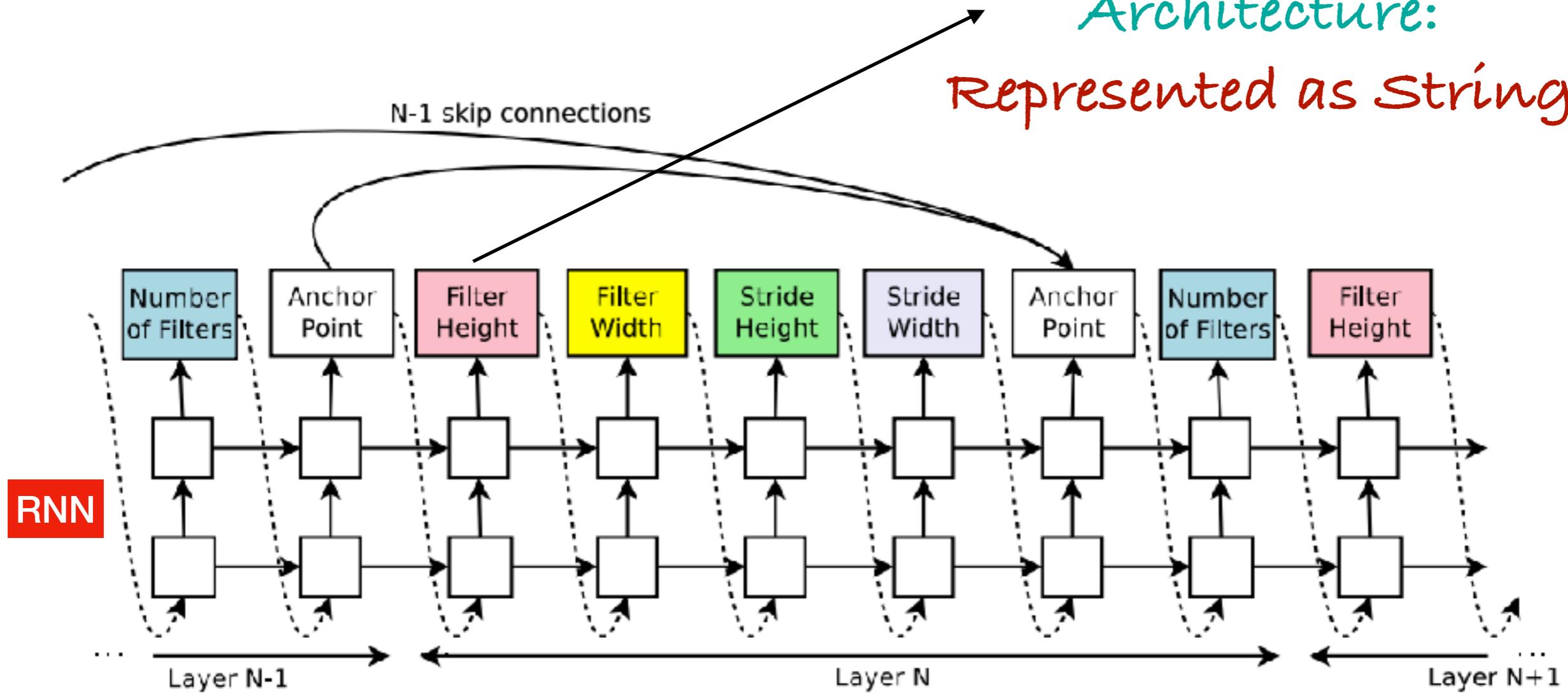
Neural Architecture Search

# Neural Architecture Search



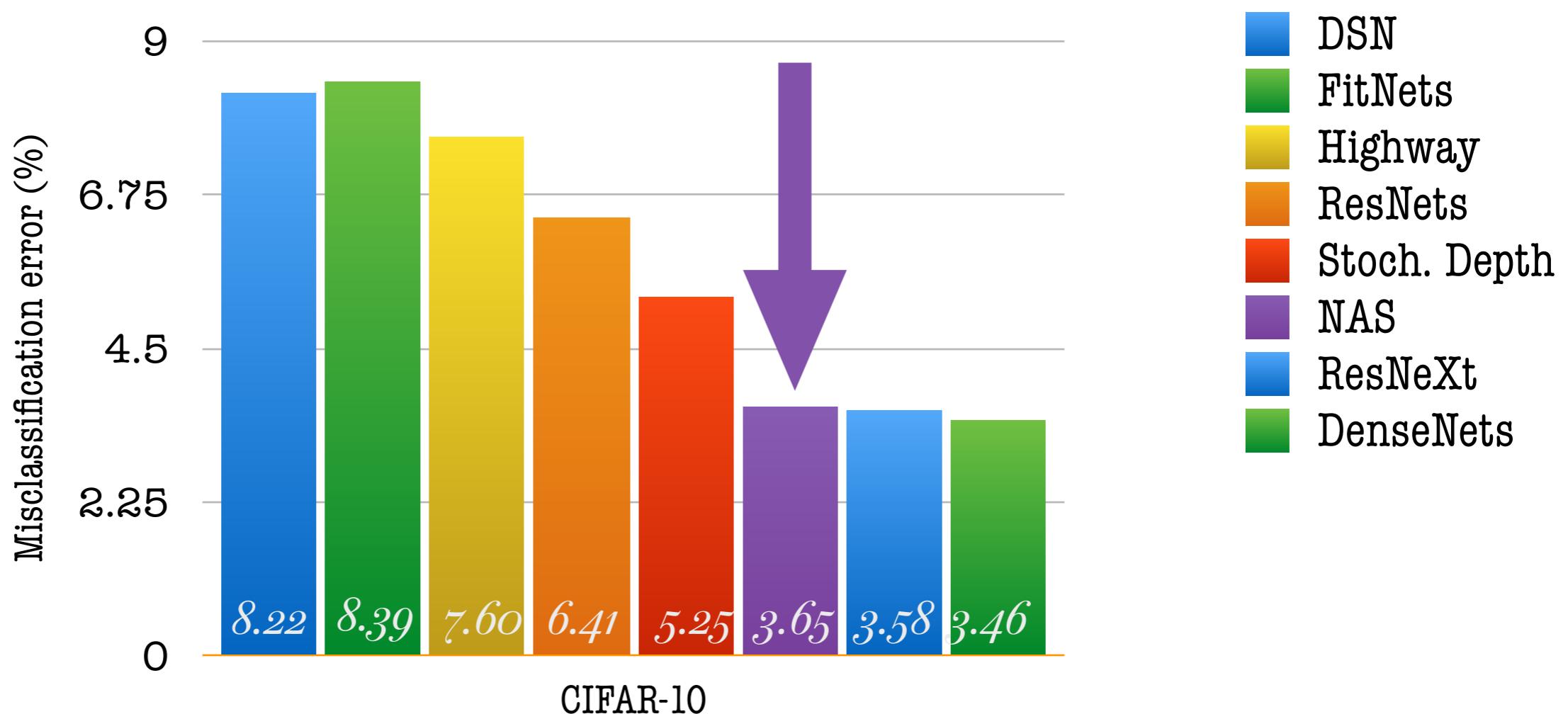
# Neural Architecture Search (NAS)

Neural Network  
Architecture:  
Represented as String



# CIFAR-10 Results

ML Image classification dataset:

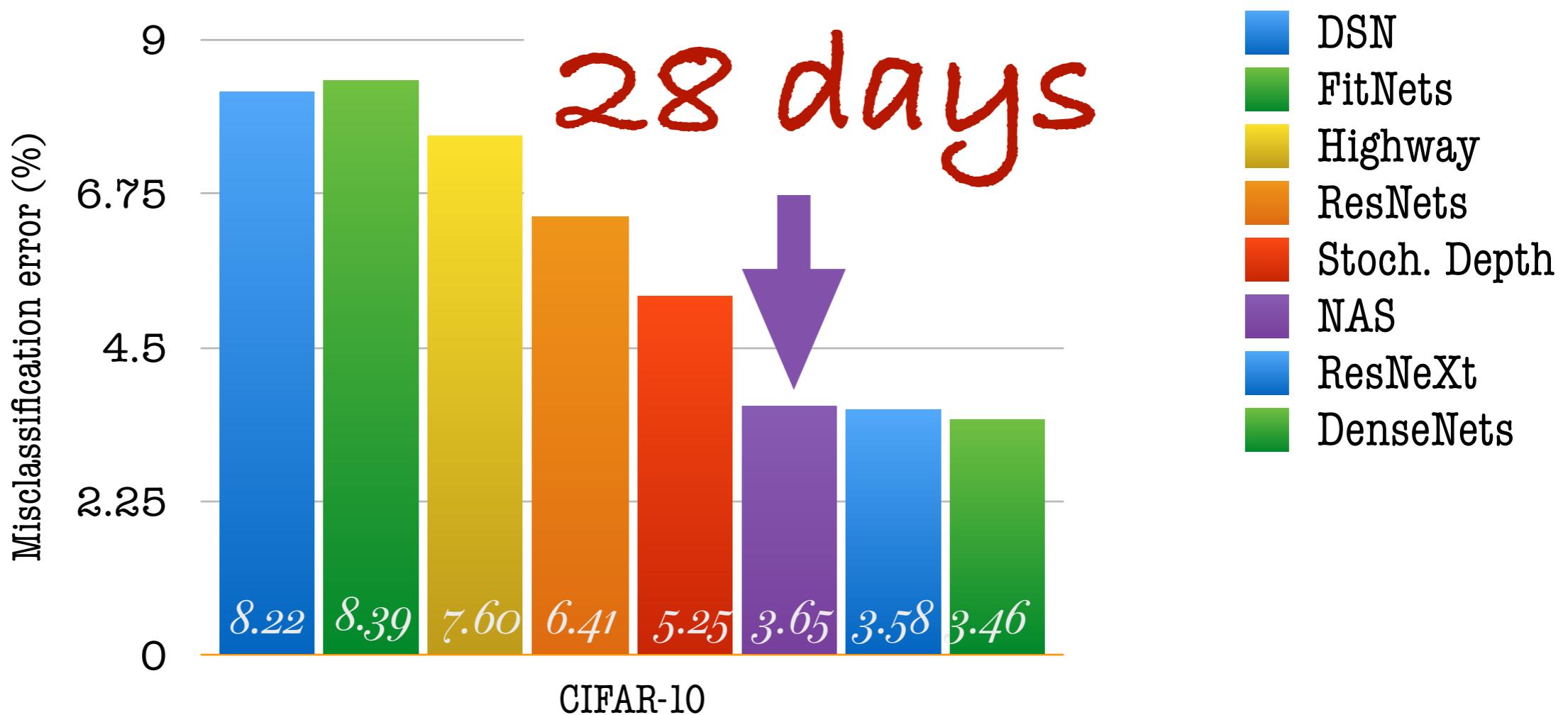


# CIFAR-10 Results

ML Image classification dataset:

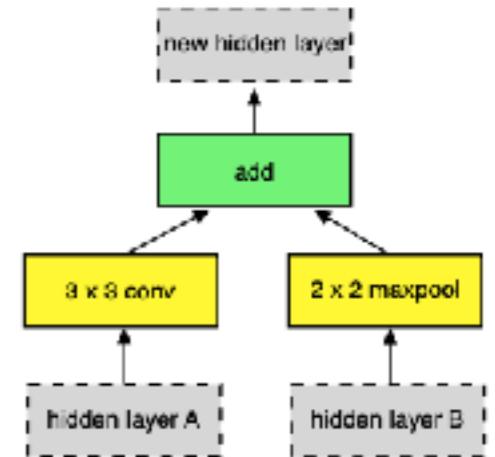
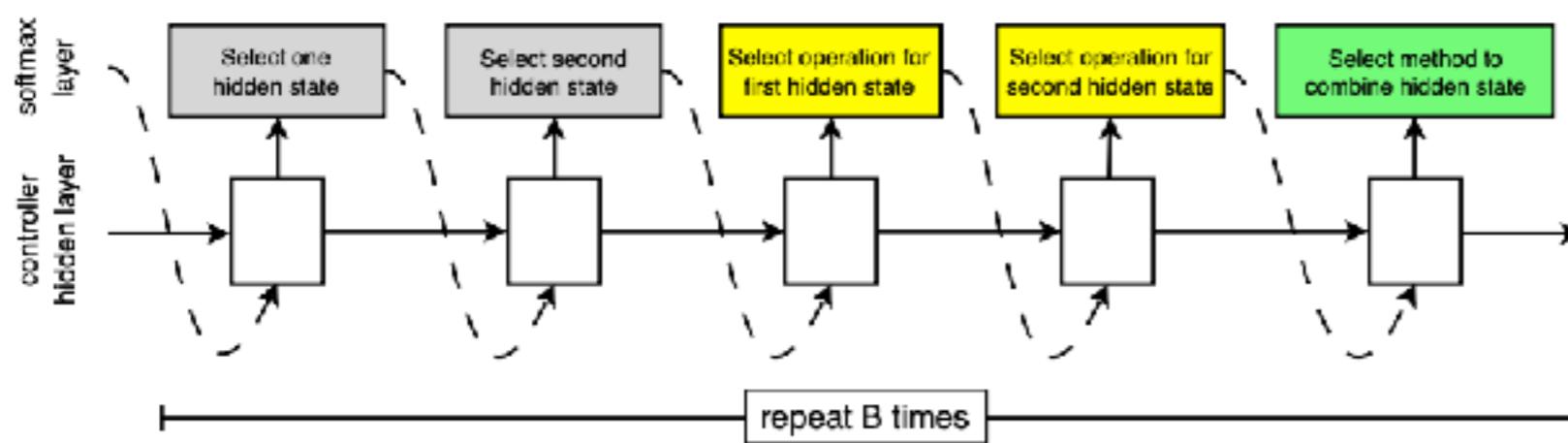
800 GPU

28 days



# Learning Transferable Architectures for Scalable Image Recognition

Instead of searching a whole architecture,  
search for a block  
repeat B times

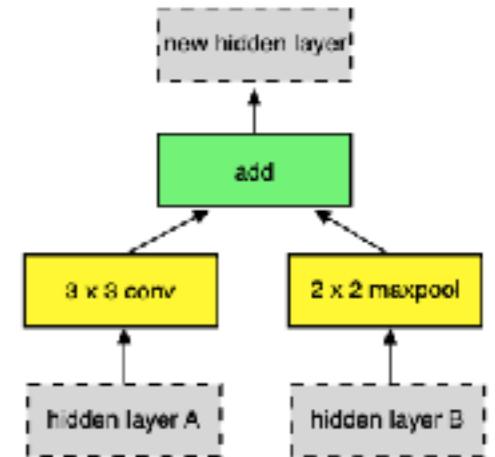
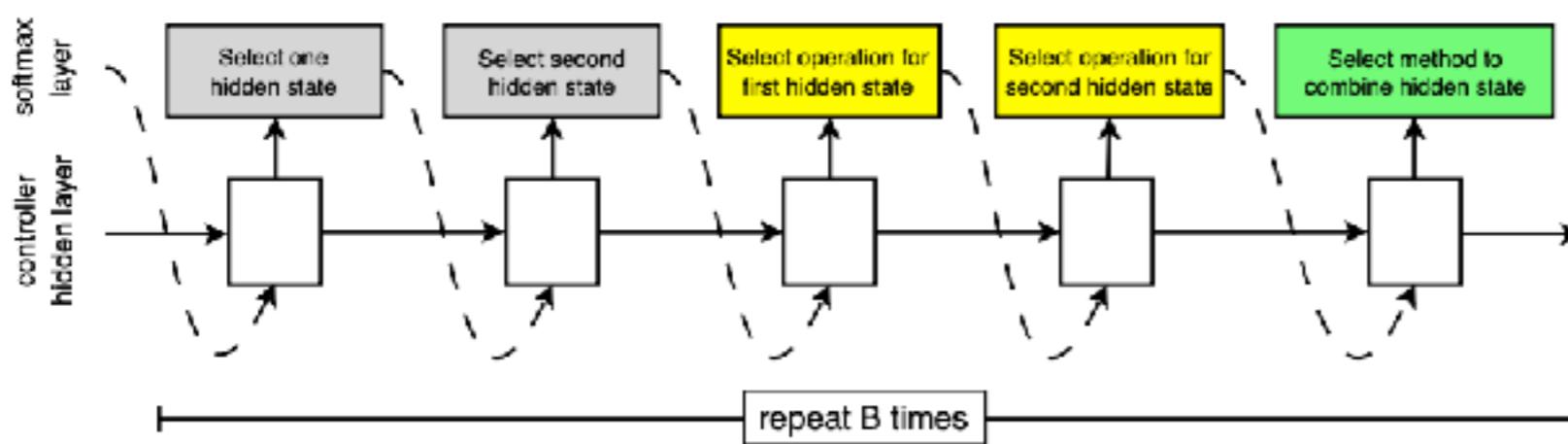


# Learning Transferable Architectures for Scalable Image Recognition

Instead of searching a whole architecture,  
search for a block  
repeat B times

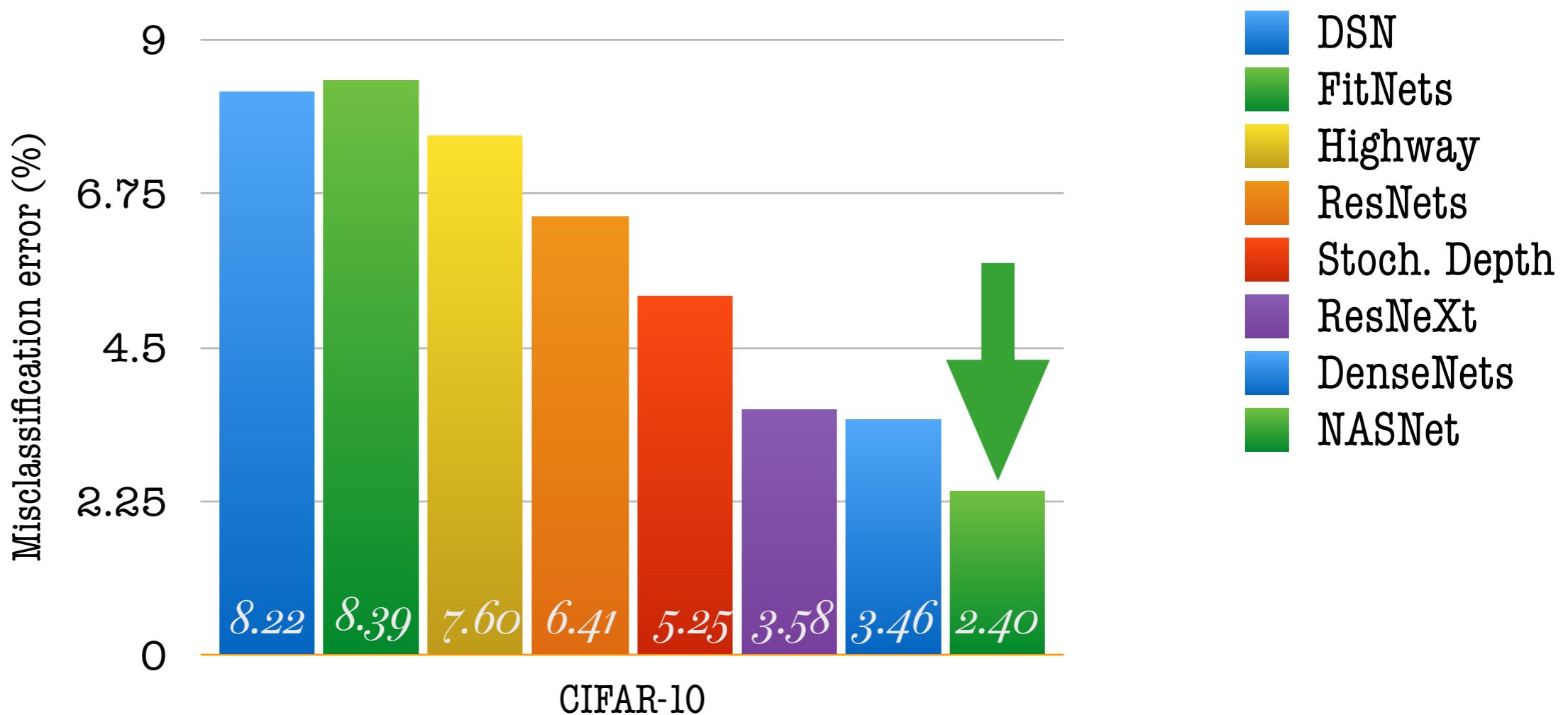
=

Reduced Search Space



# CIFAR-10 Results

ML Image classification dataset:

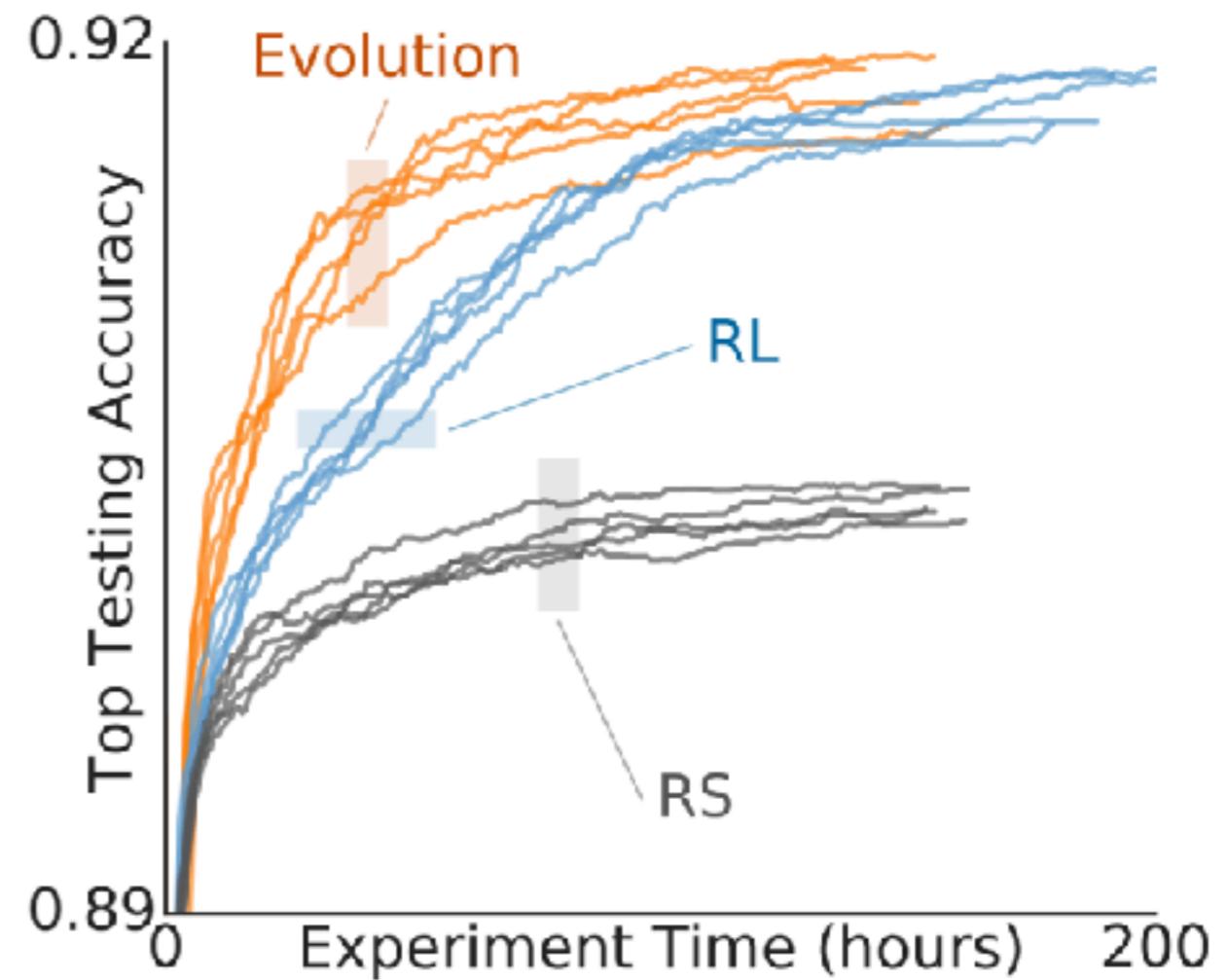
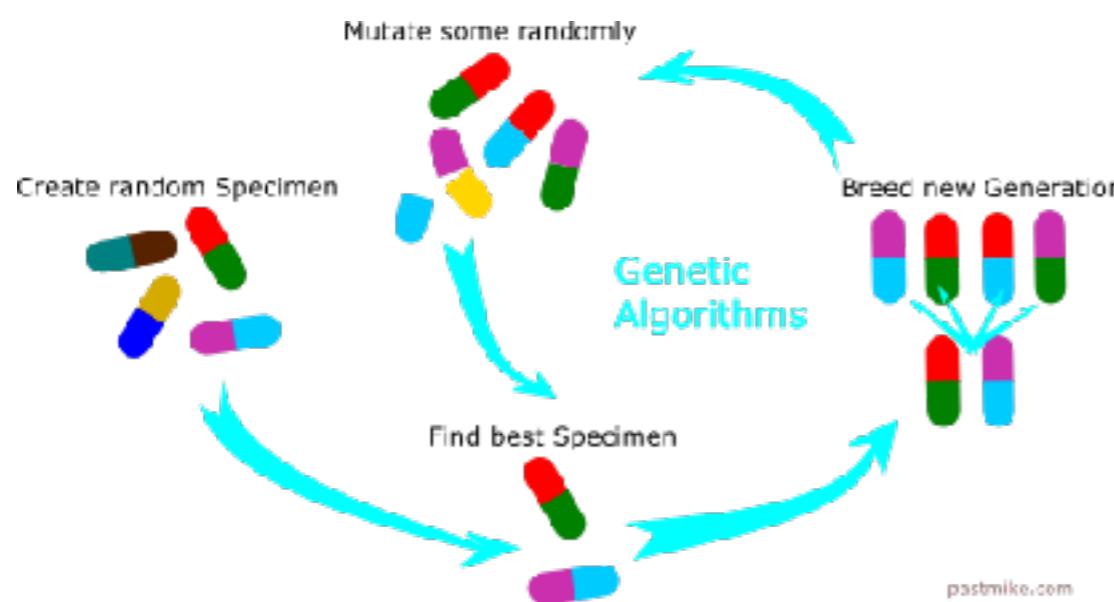


# CIFAR-10 Results

ML Image classification dataset:

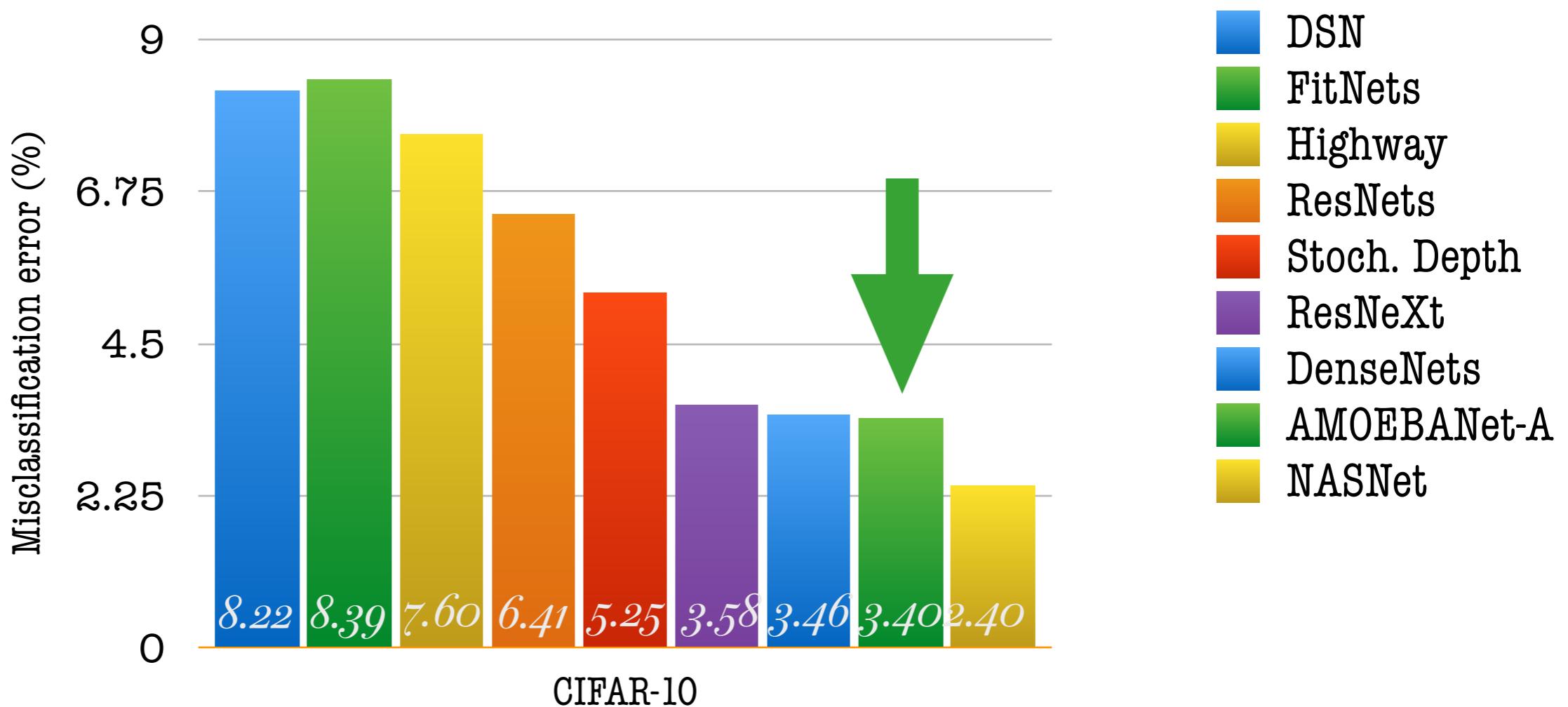


# Regularized Evolution for Image Classifier Architecture Search



# CIFAR-10 Results

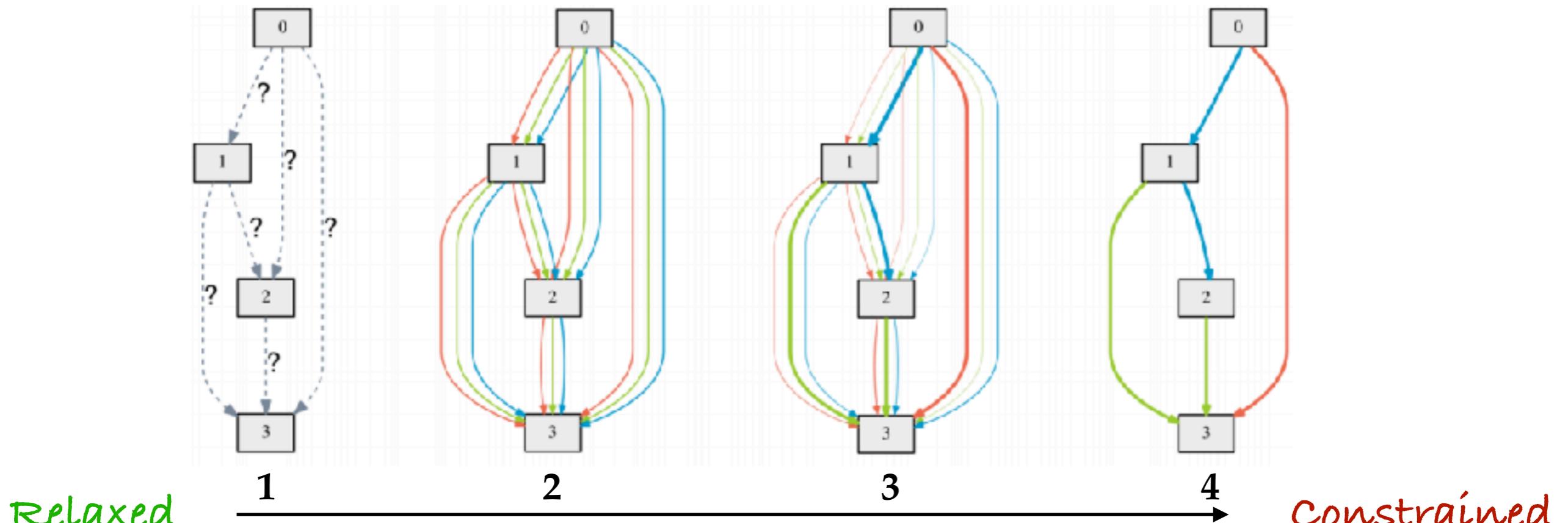
ML Image classification dataset:



# DARTS: Differentiable Architecture Search

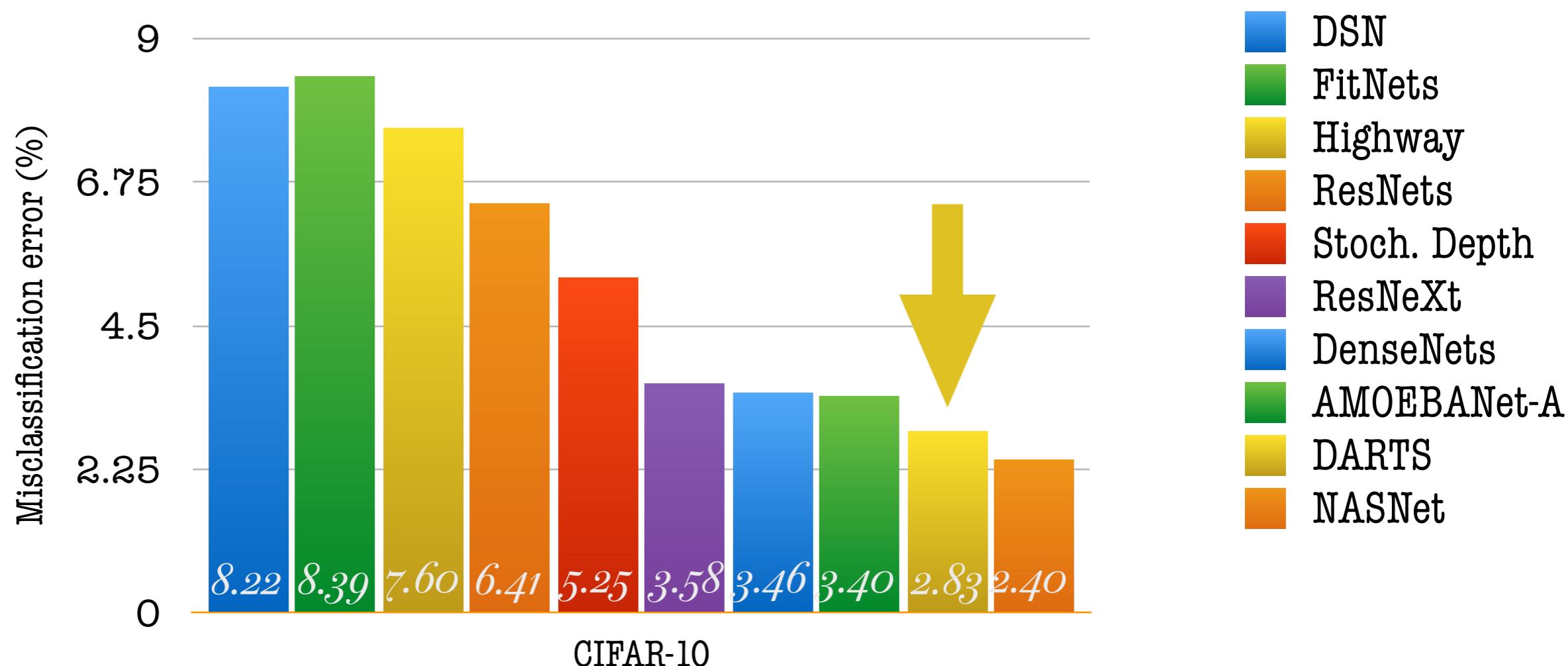
Make architecture search differentiable,  
faster than RL, and evolutionary.

Explore all the solutions at the same time.



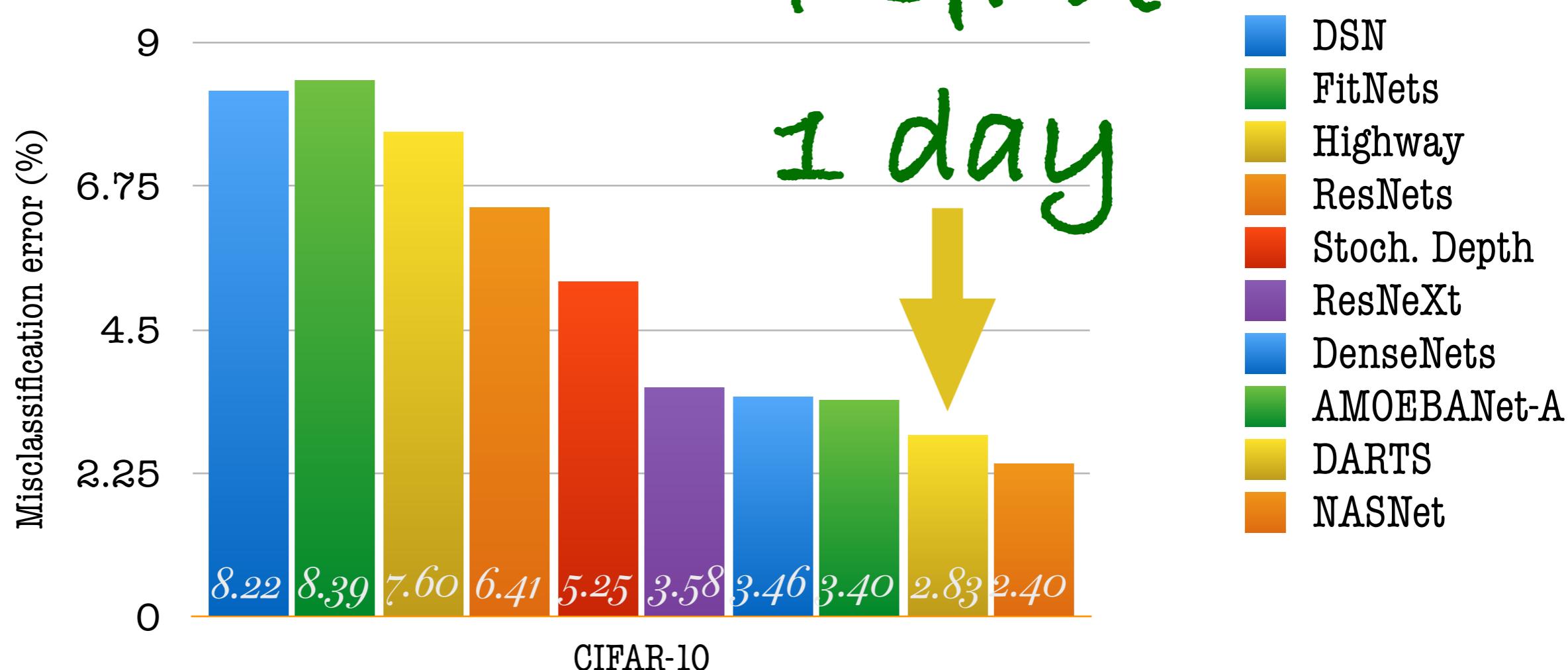
# CIFAR-10 Results

ML Image classification dataset:



# CIFAR-10 Results

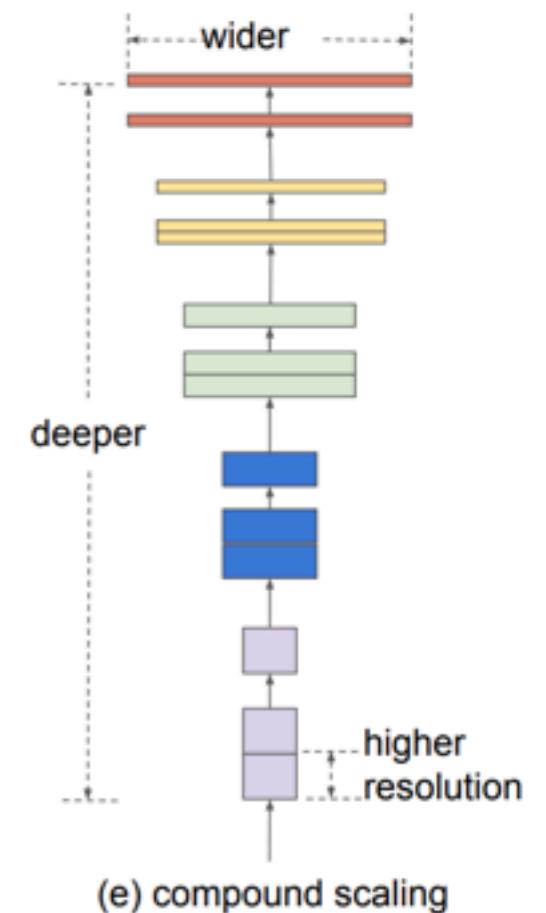
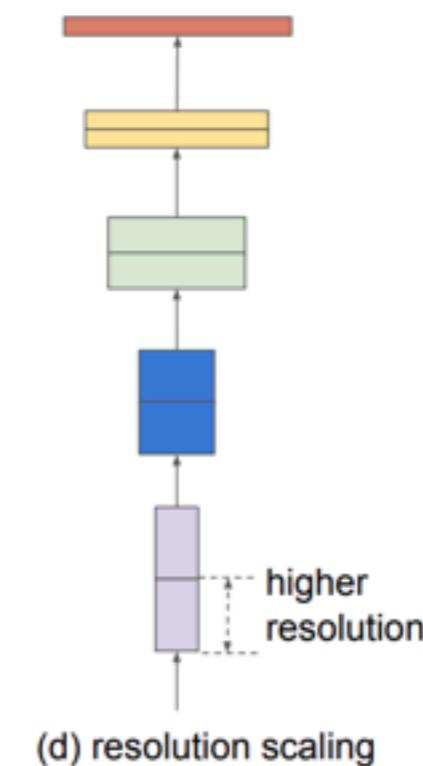
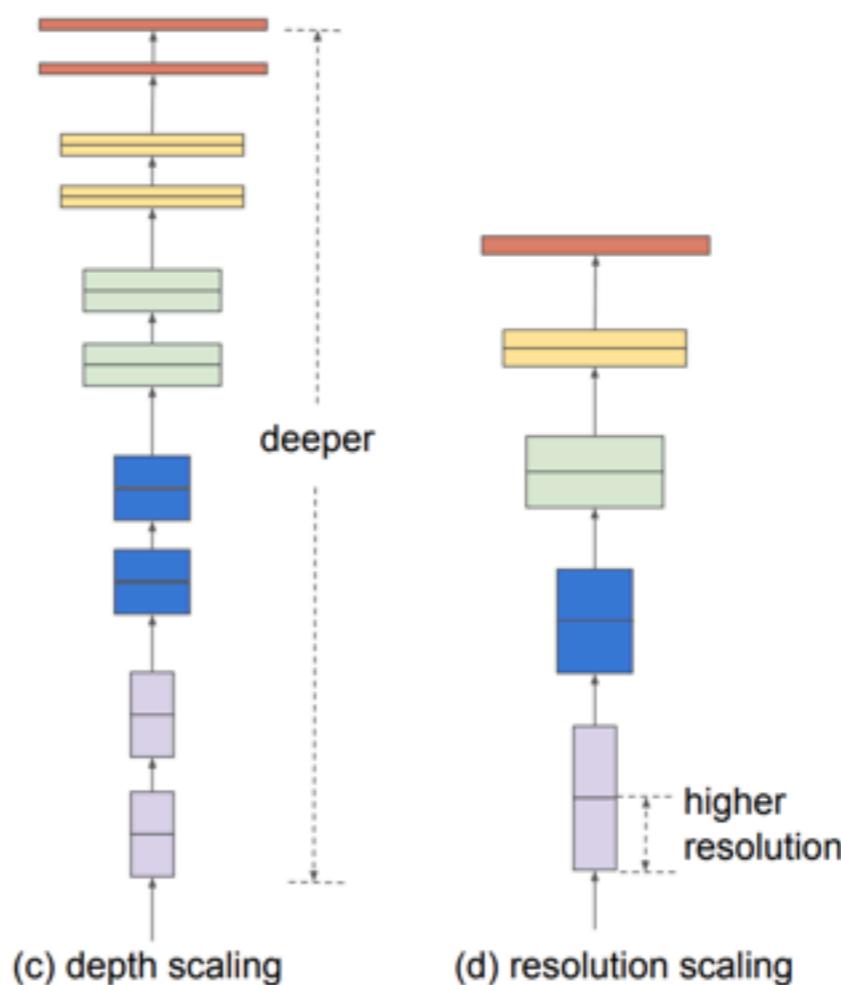
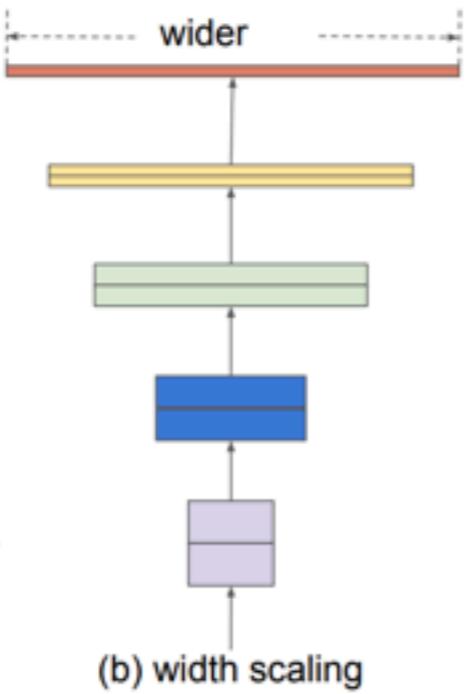
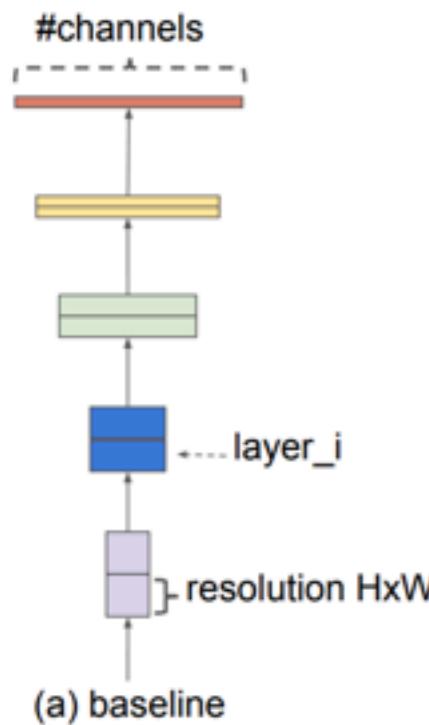
ML Image classification dataset:



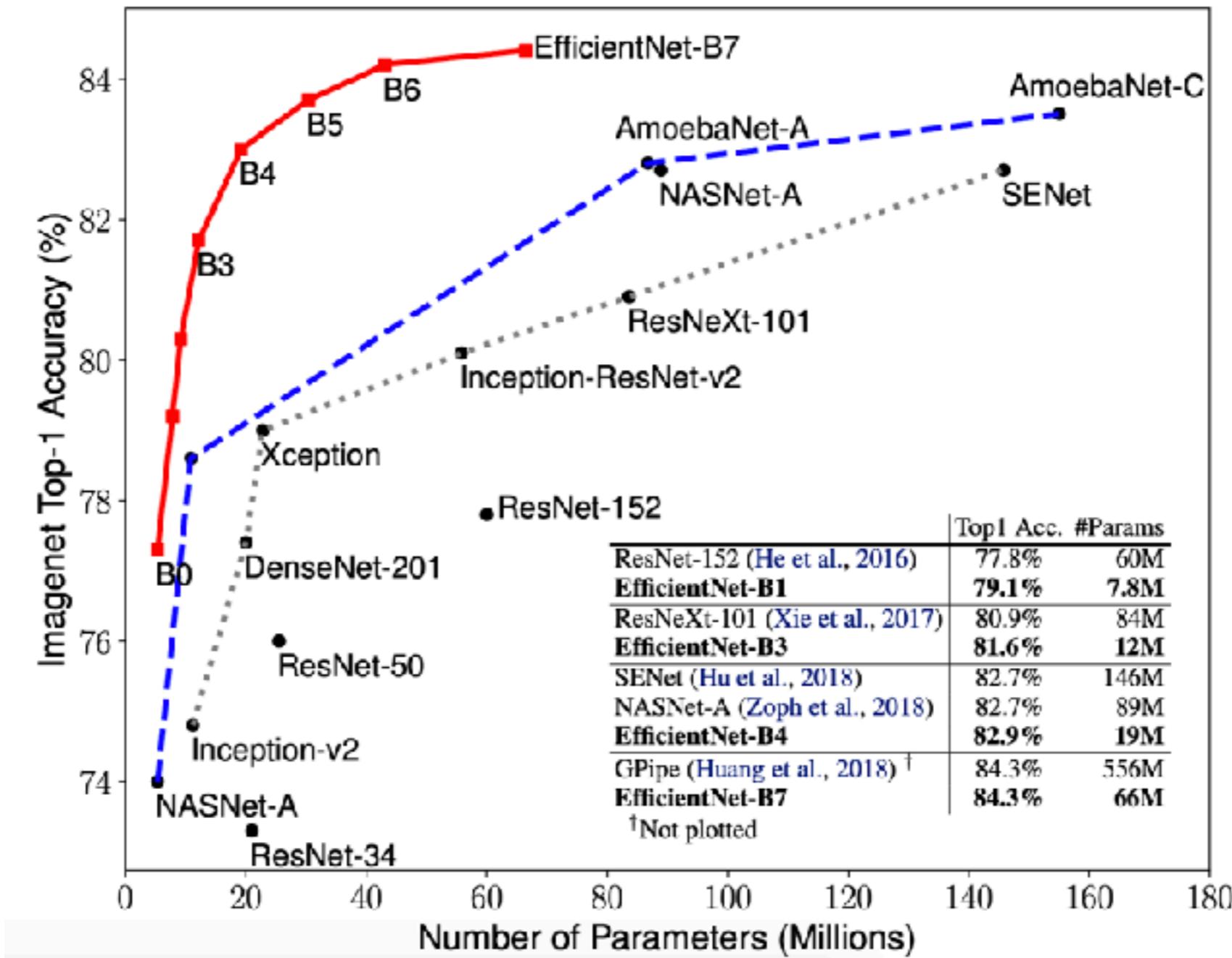
# EfficientNet

1. Use NAS to find an efficient block  
Optimizing ACCxFLOPS

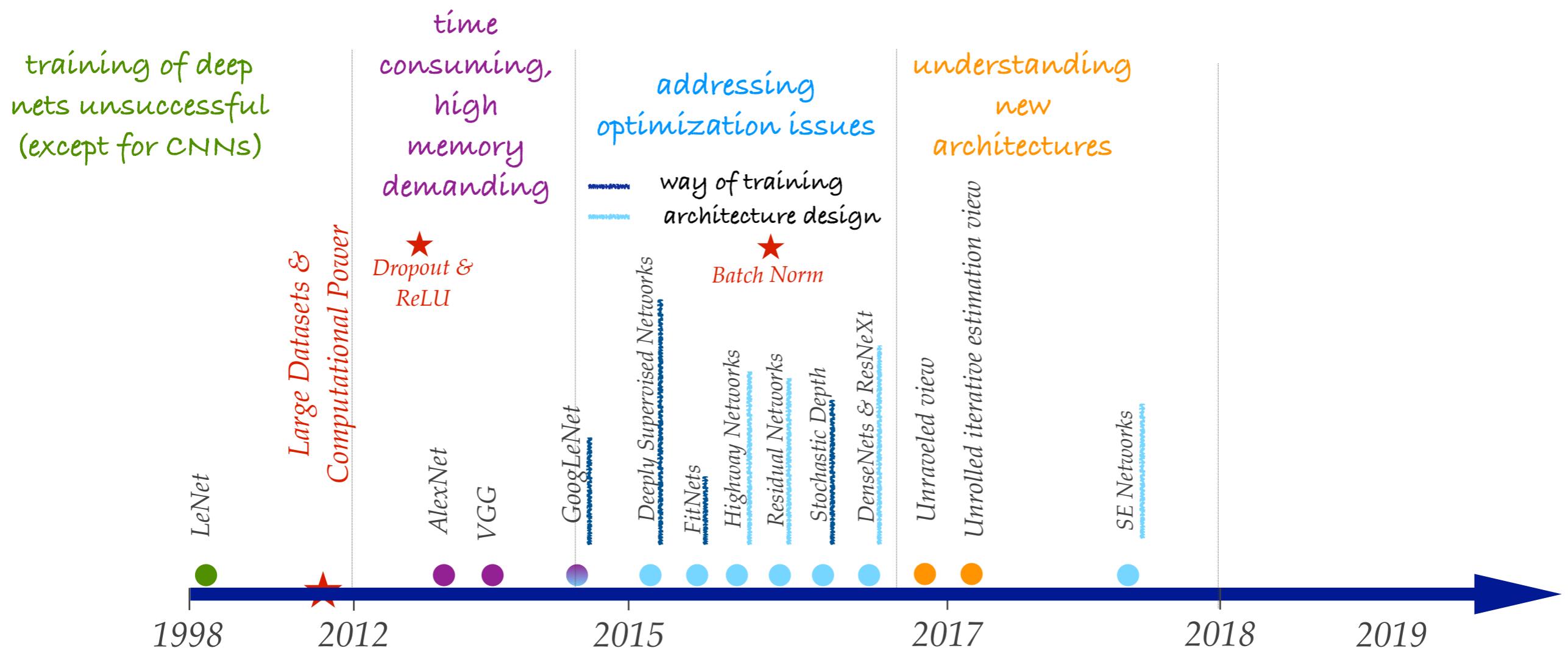
2. Scale!!!!



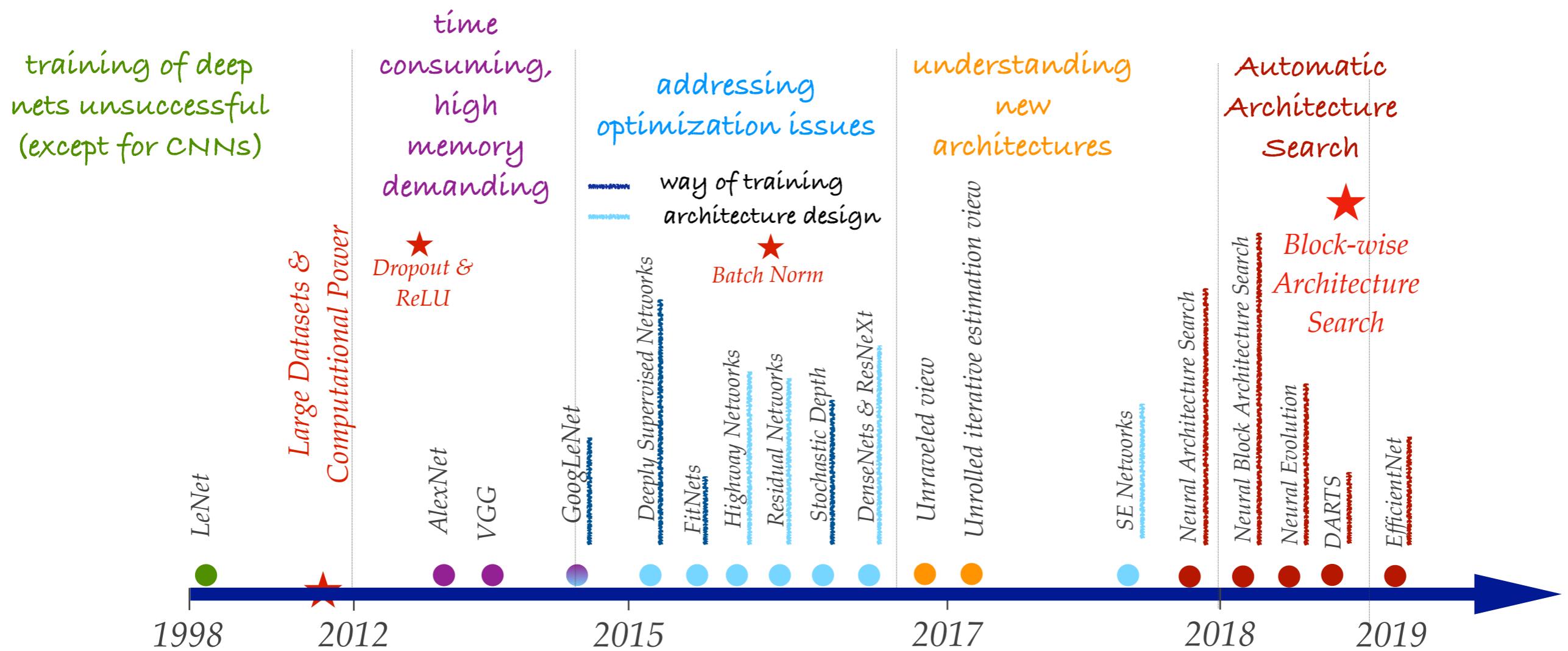
# EfficientNet



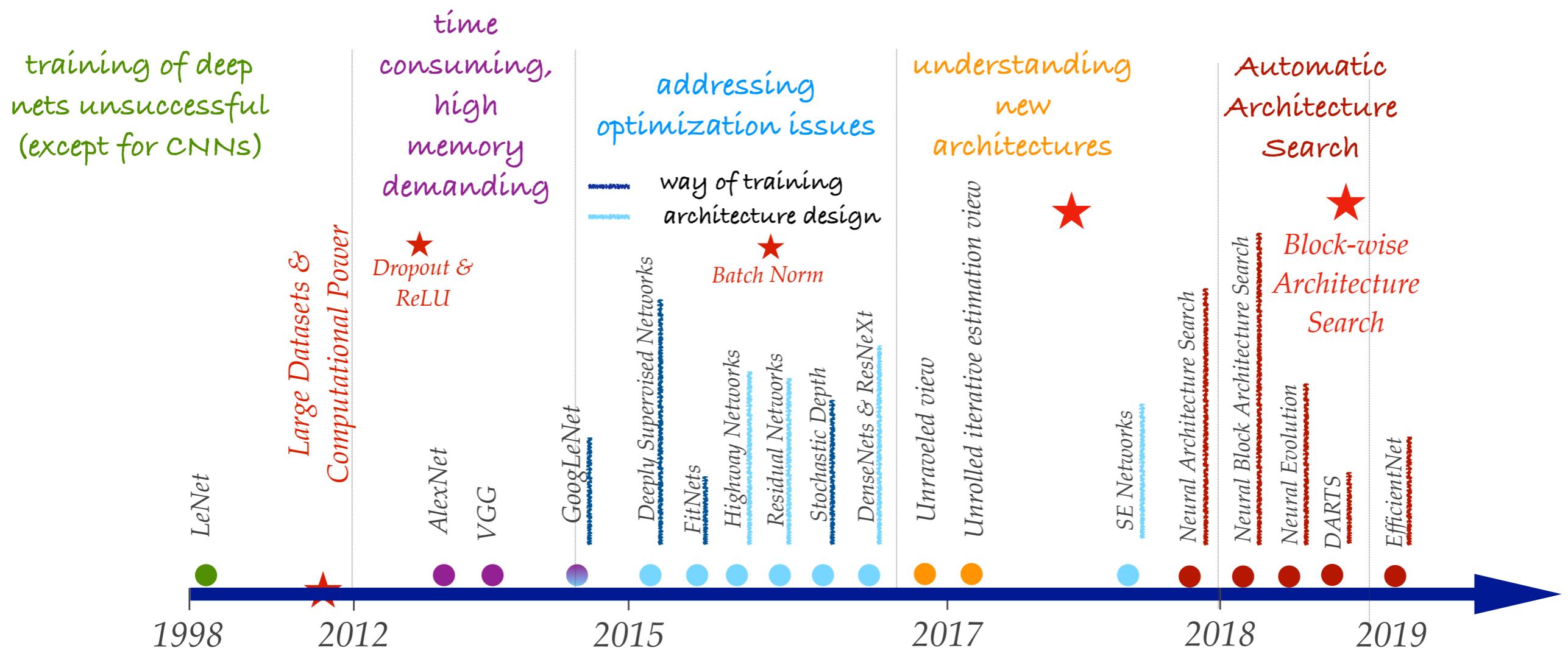
# Wrap Up



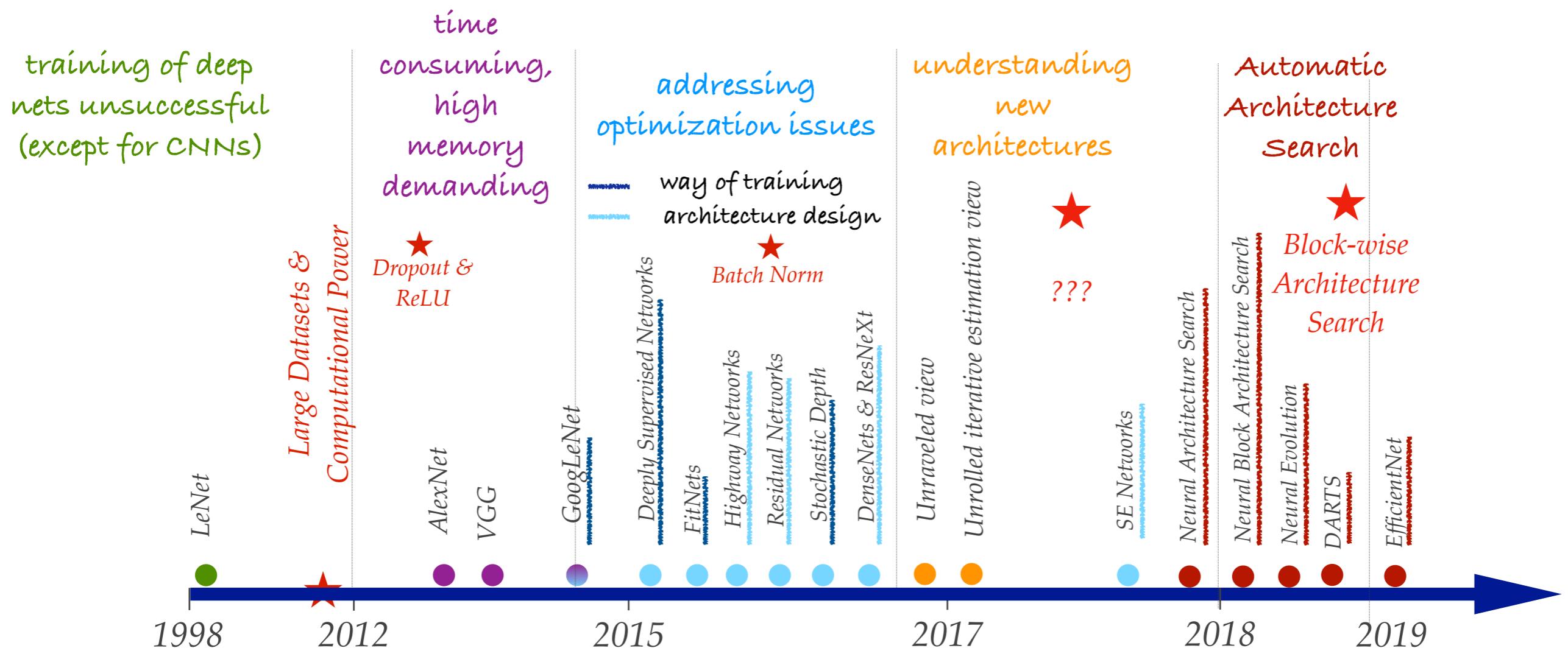
# Wrap Up



# Wrap Up

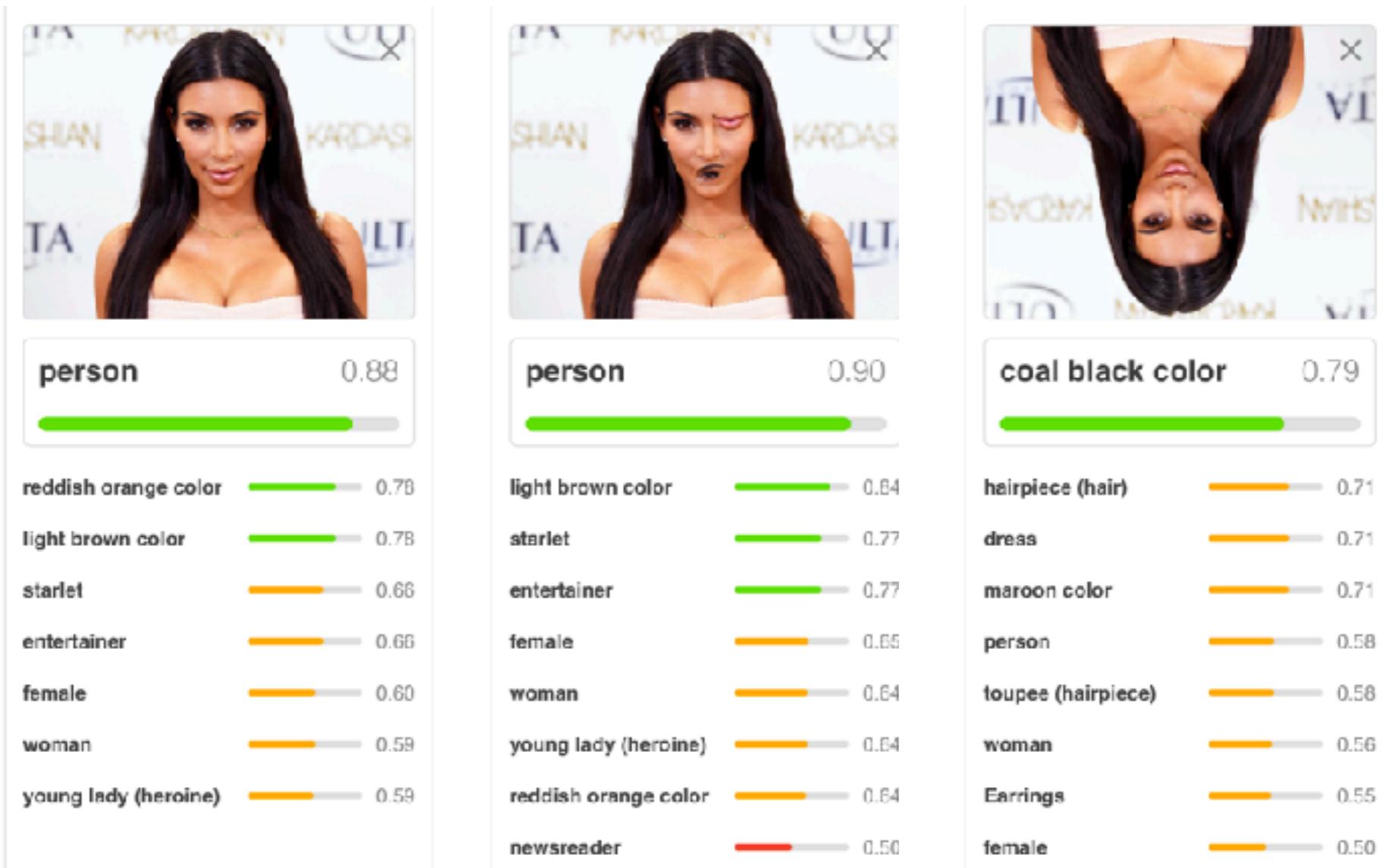


# Wrap Up



# CapsNets

## CNNs Problem

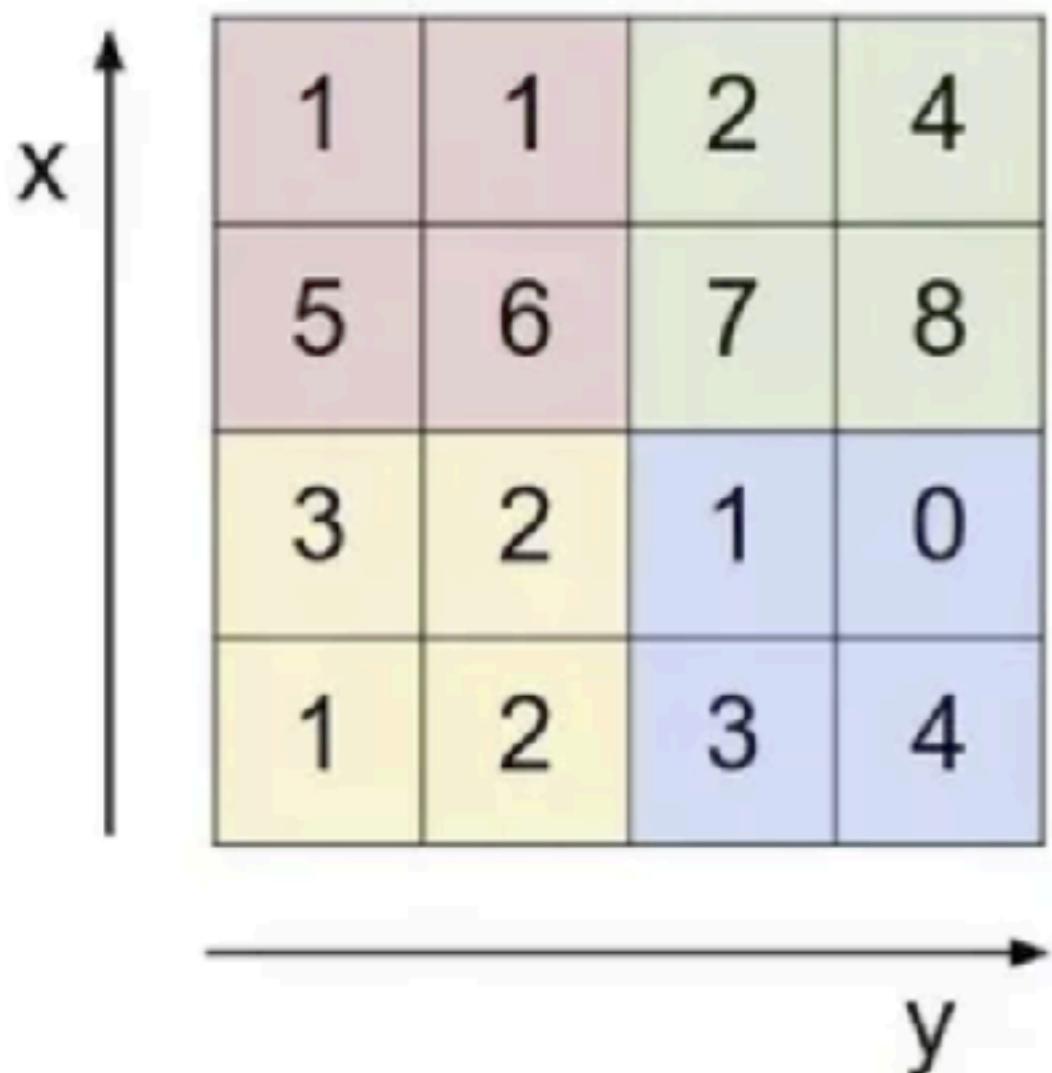


[Sabour et al. 2017]

# CapsNets

Maxpool

Single depth slice



max pool with 2x2 filters  
and stride 2

A 2x2 grid representing the max pool output. It has four cells containing the values 6, 8, 3, and 4. The top-left cell is pink, top-right is light green, bottom-left is yellow, and bottom-right is light blue.

6	8
3	4

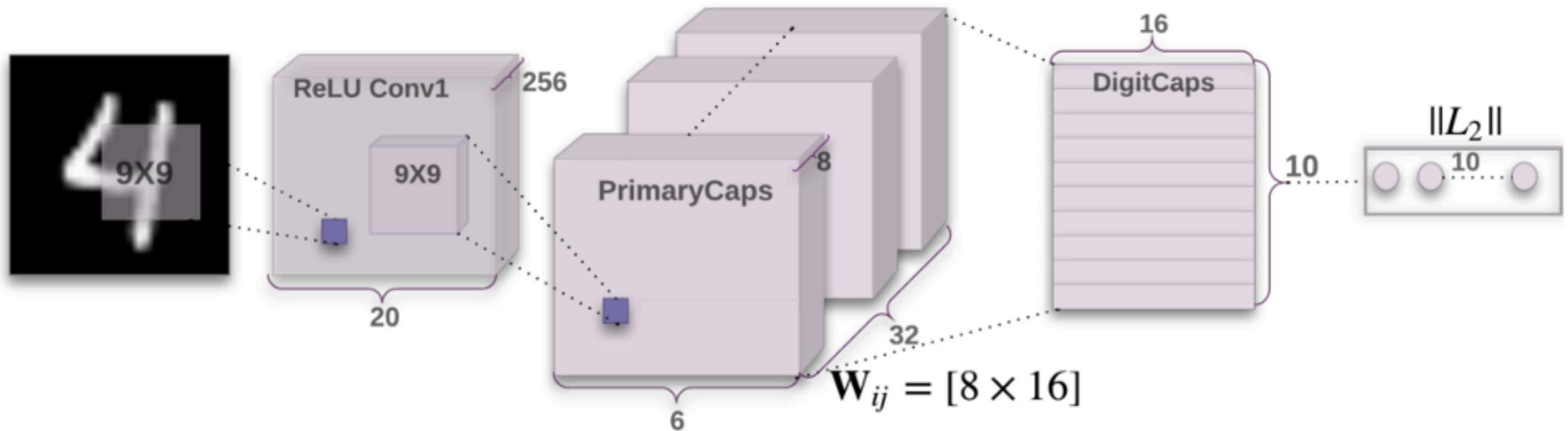
# CapsNets

Solution

scalar activations -> tensor Activations

tensor magnitude: presence of a feature

tensor direction: pose information!

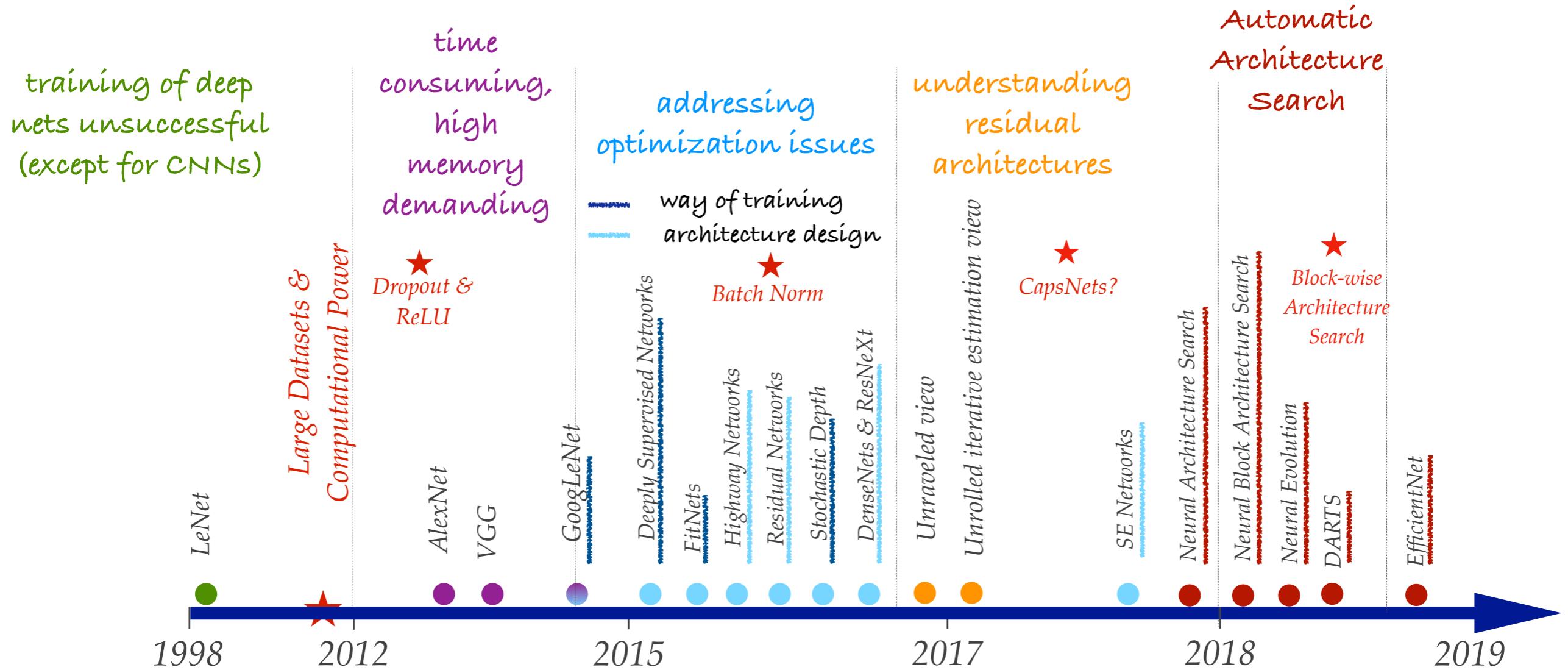


# CapsNets

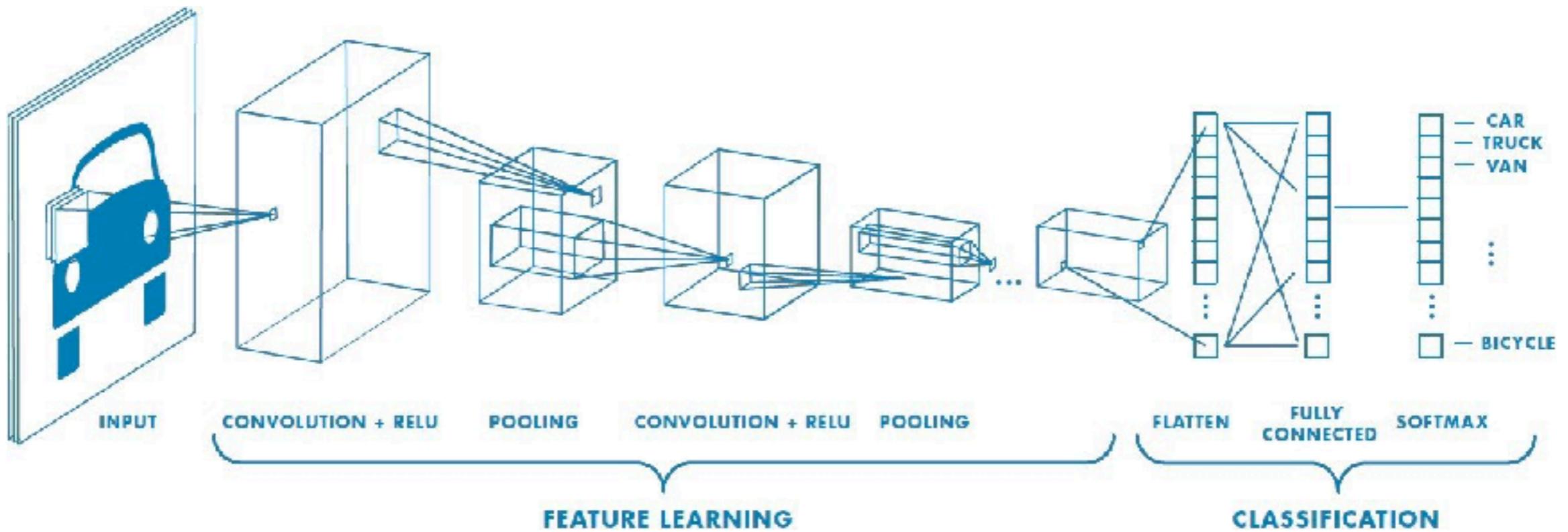
45% Better on viewpoint invariance!!



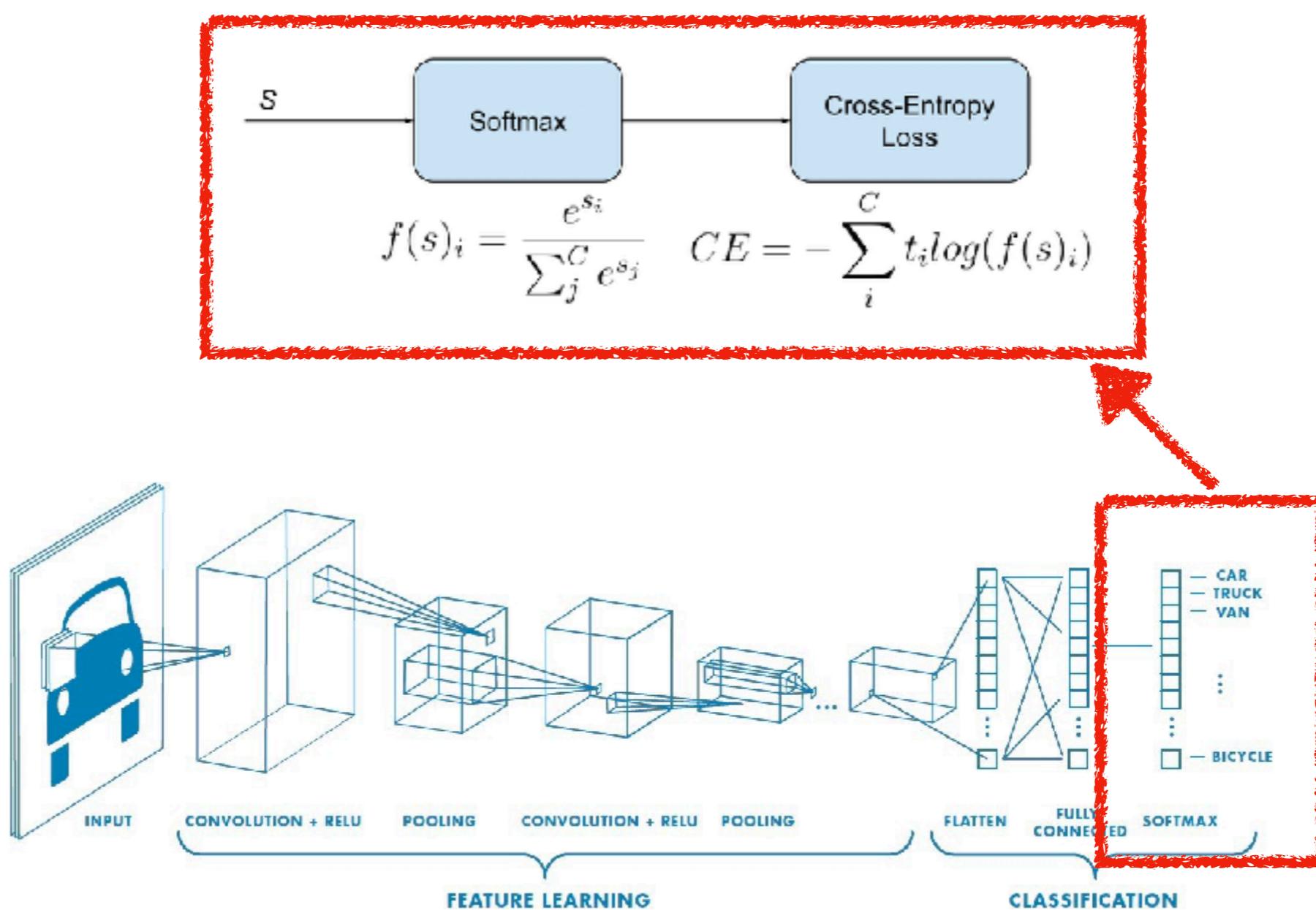
# Wrap Up



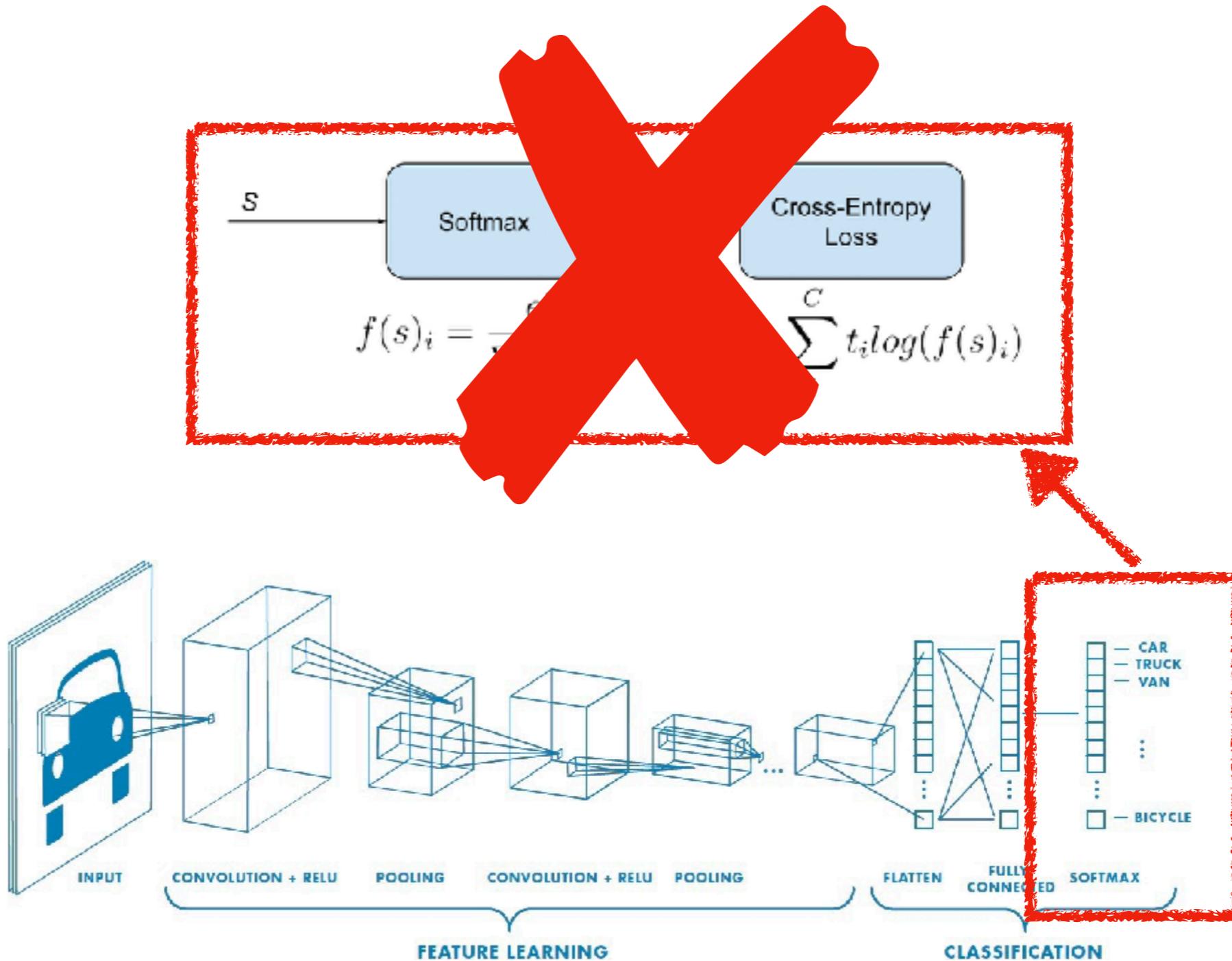
# Supervised Learning so far



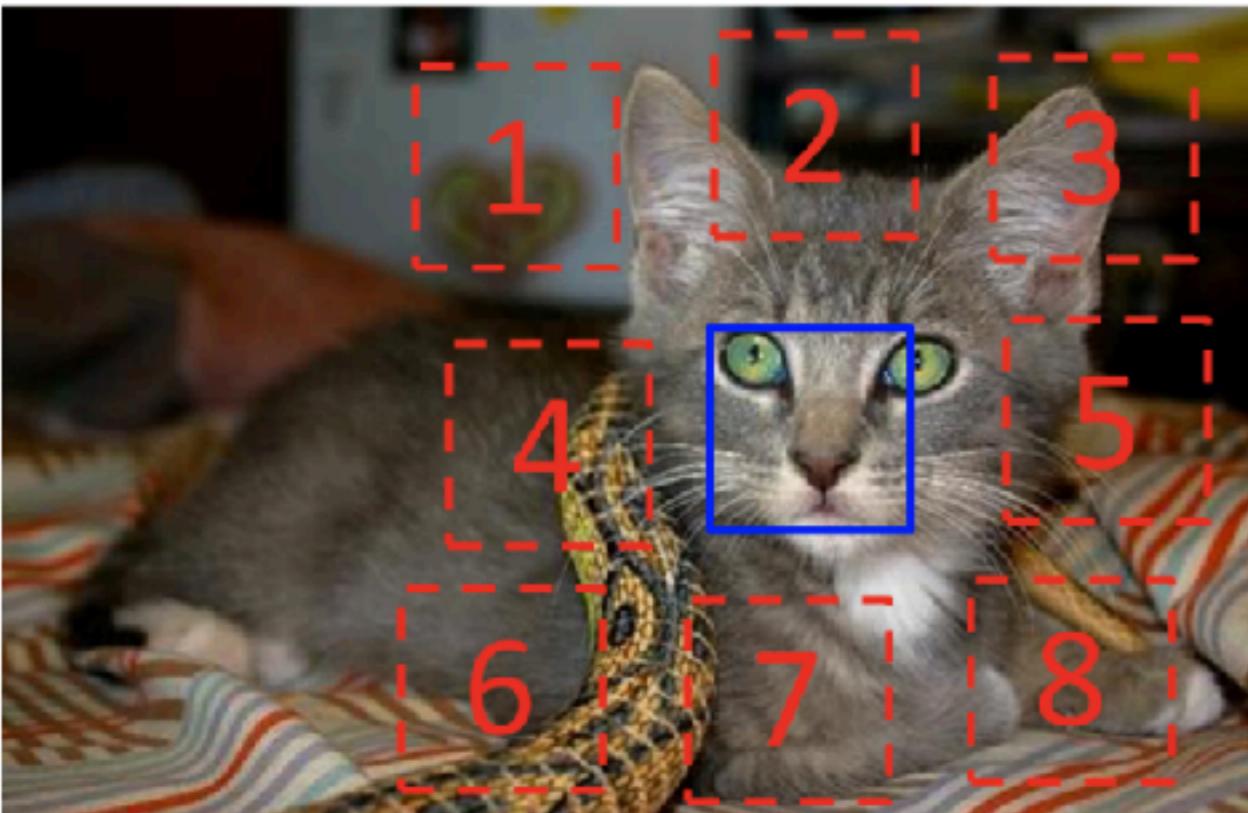
# Supervised Learning so far



# The Self-Supervised Revolution



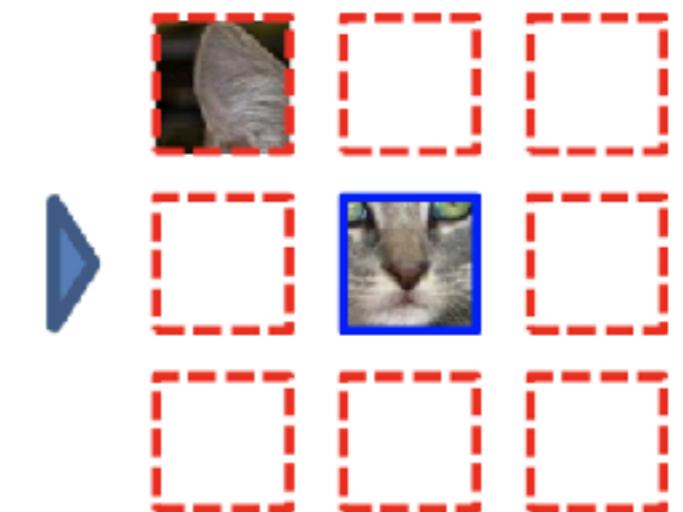
# Learning from pretext tasks



$$X = (\text{[cat eye, ear]}, \text{[cat ear, fur]}); Y = 3$$

jigsaw puzzle  
what is the relative position?

Example:



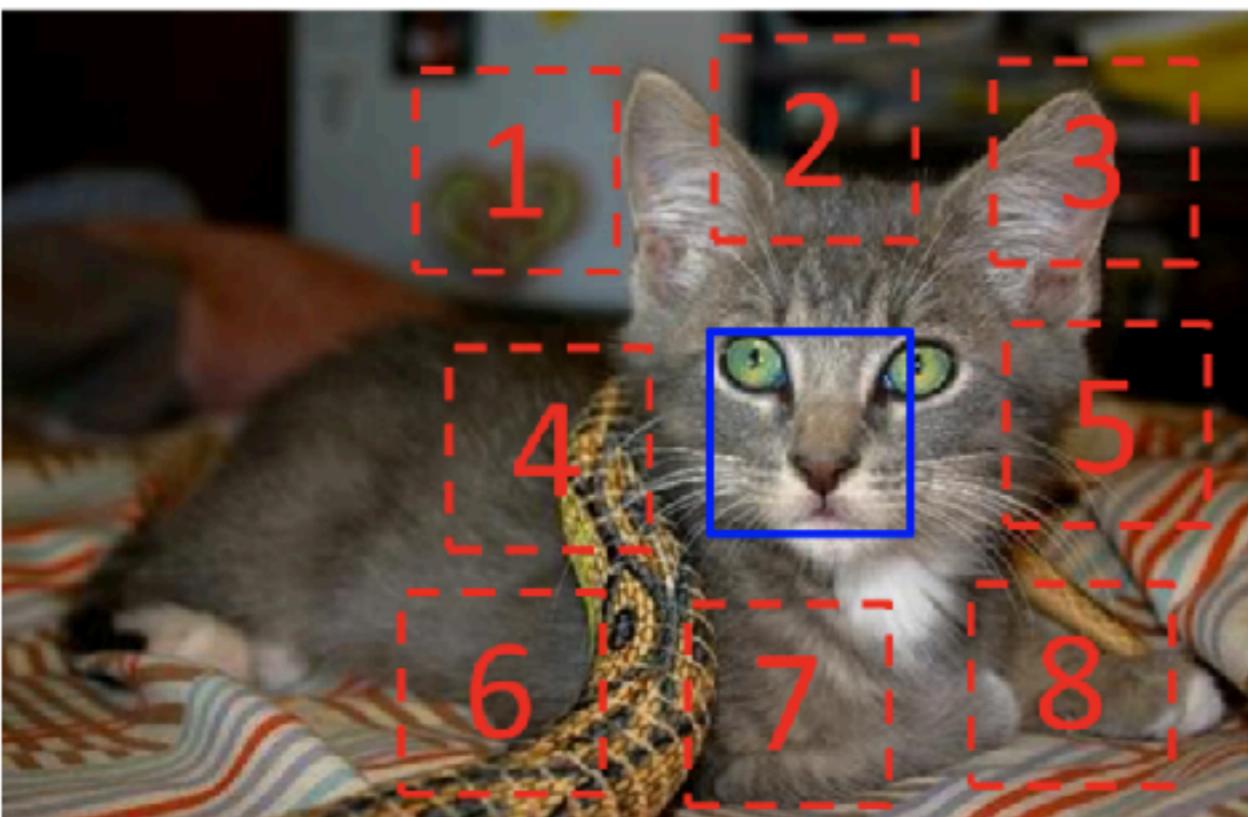
Question 1:



Question 2:



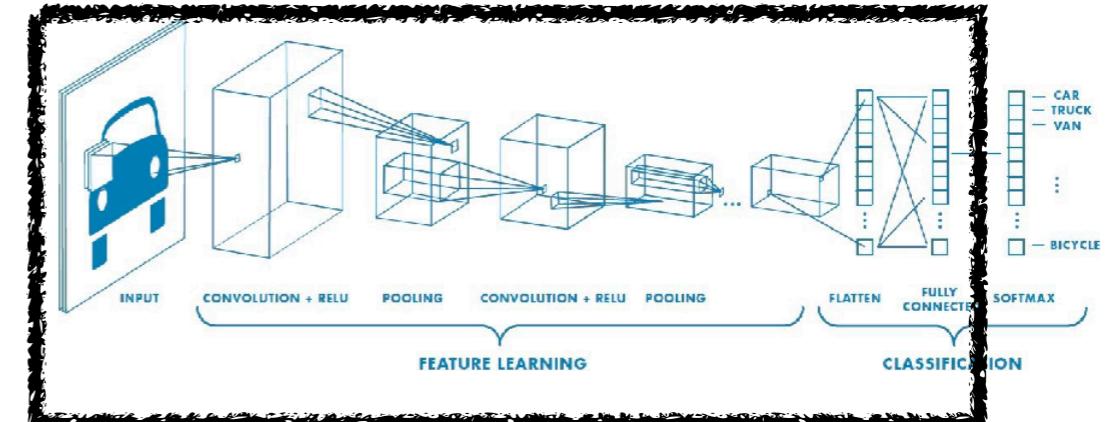
# Learning from pretext tasks



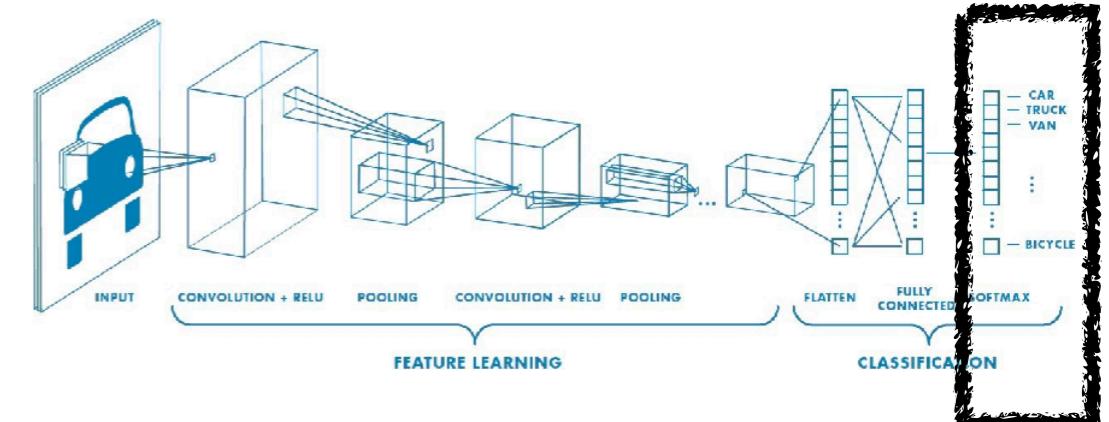
$$X = (\text{[eye]}, \text{[ear]}); Y = 3$$

jigsaw puzzle  
what is the relative position?

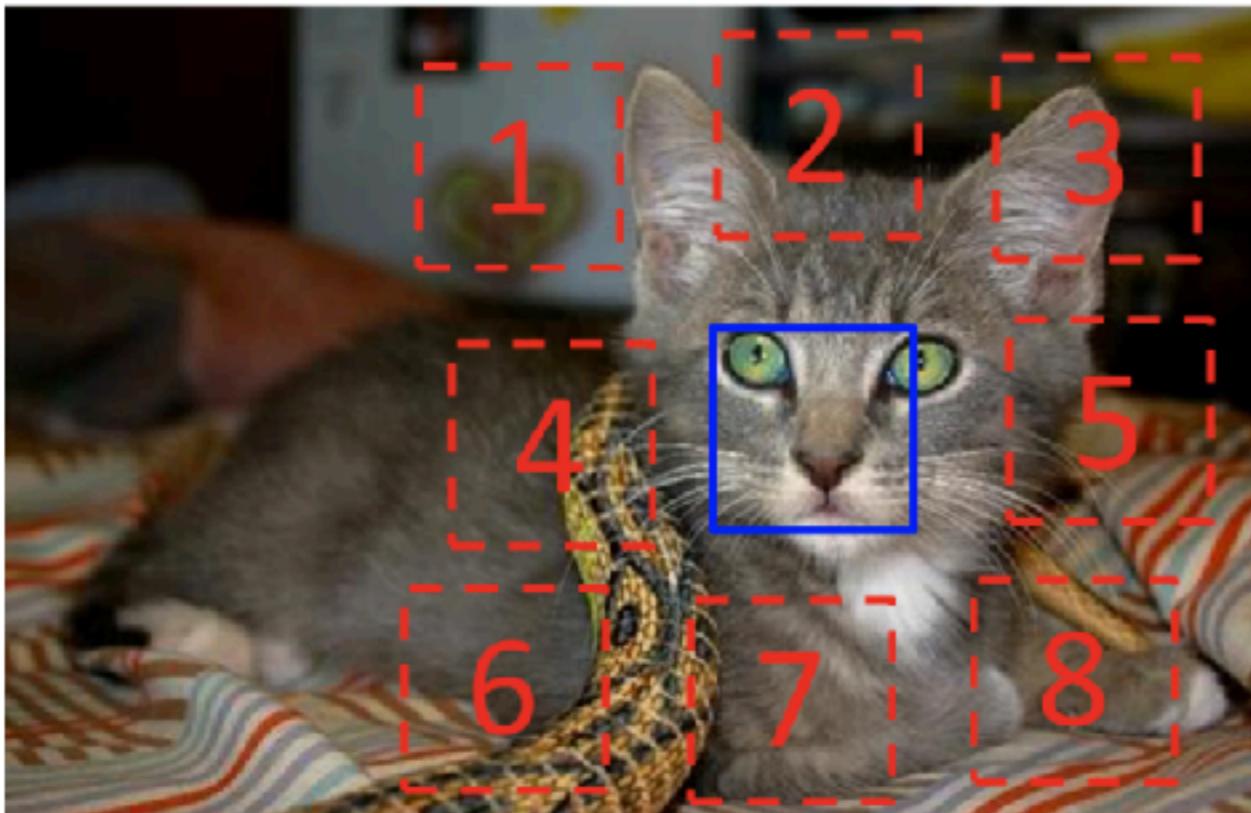
1. Train Feature Representation



2. Finetune classifier w/ some labeled data



# Learning from pretext tasks

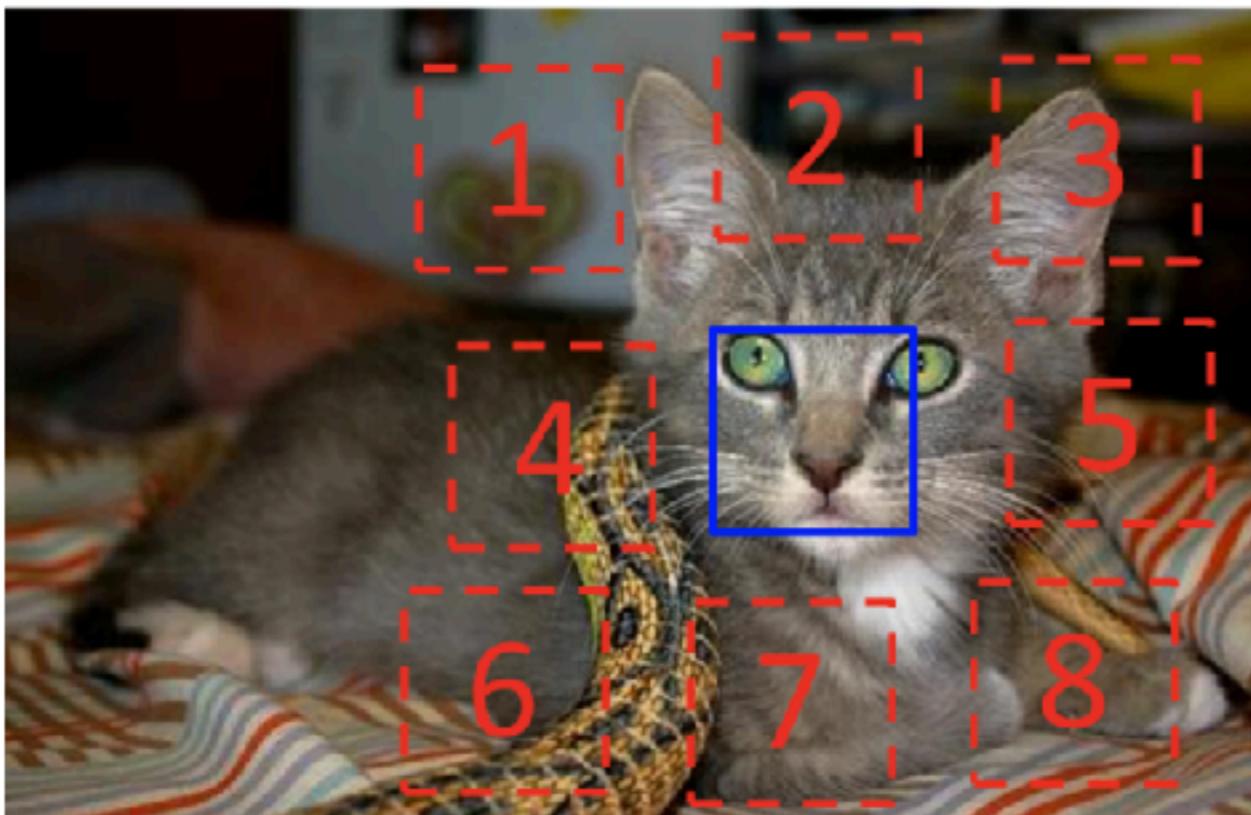


$$X = (\text{[cat eye]}, \text{[ear]}); Y = 3$$

Results are good when a lot of unlabeled data is available and little labeled data available

jigsaw puzzle  
what is the relative position?

# Learning from pretext tasks



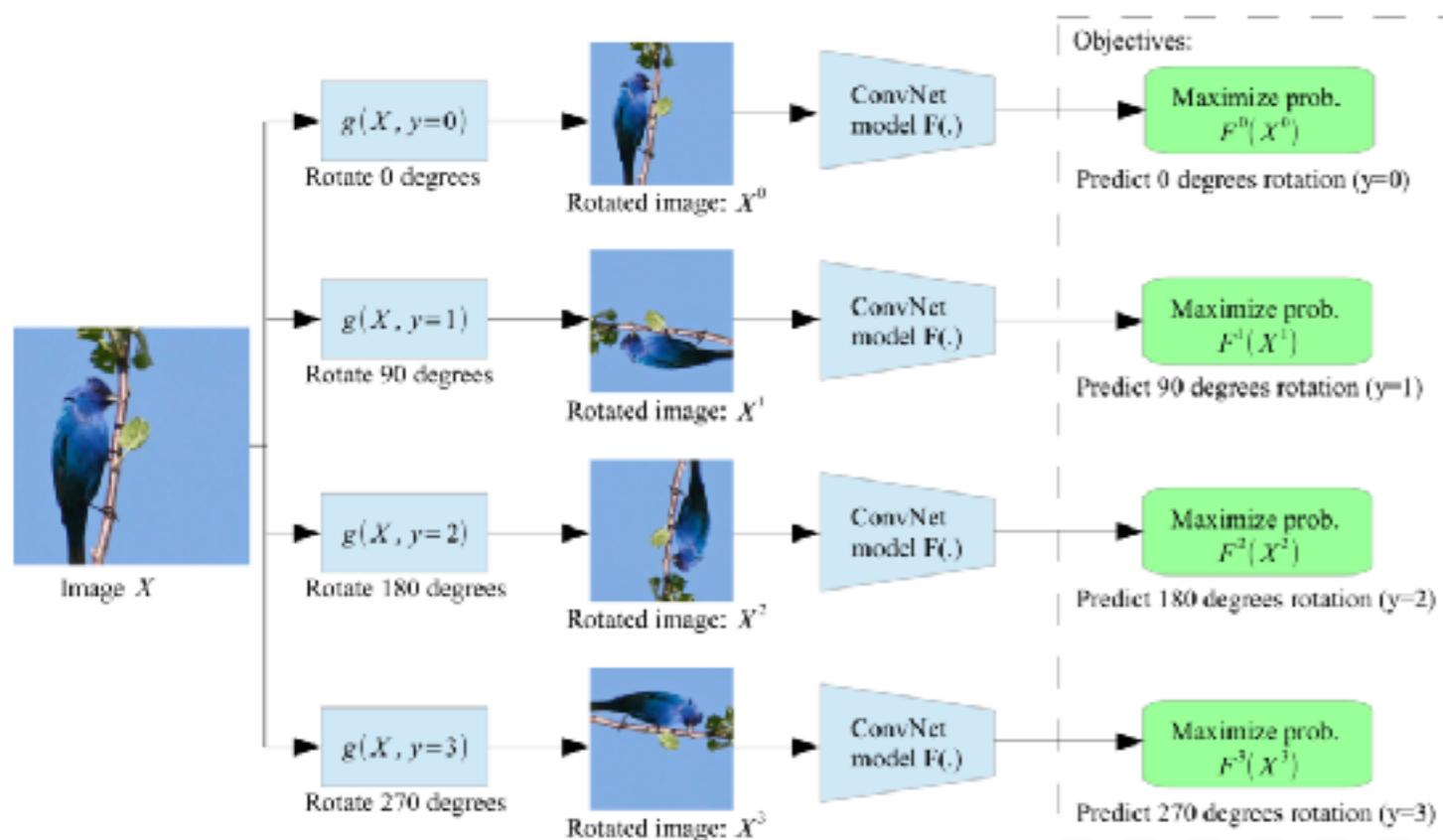
$$X = (\text{[cat eye]}, \text{[ear]}); Y = 3$$

jigsaw puzzle  
what is the relative position?

Shortcut problem: The Network always tries to "cheat".  
Ex: network can use the camera lens distortion

(Doersch et al. found a solution for this particular problem)

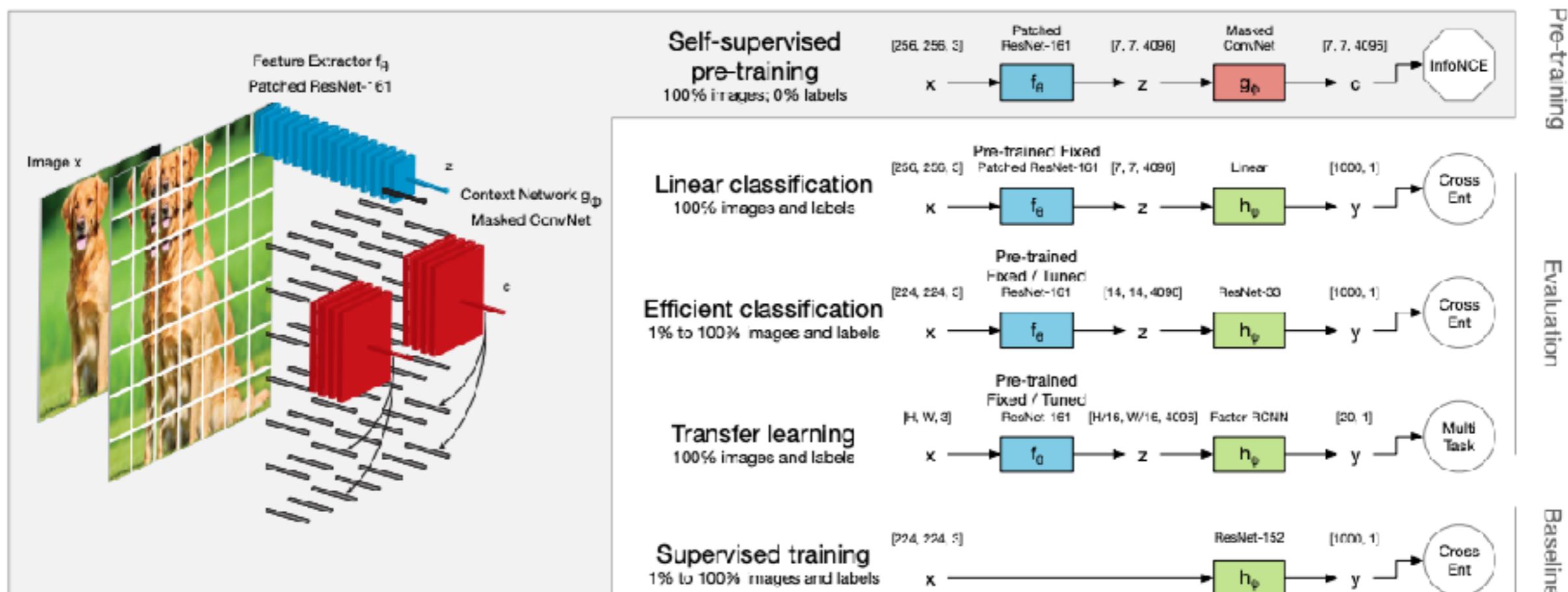
# Learning from pretext tasks



There are other pretext tasks  
Rotation  
In-painting  
Colorization  
Up to your imagination...

# Contrastive Predictive Coding

The real self-supervised revolution

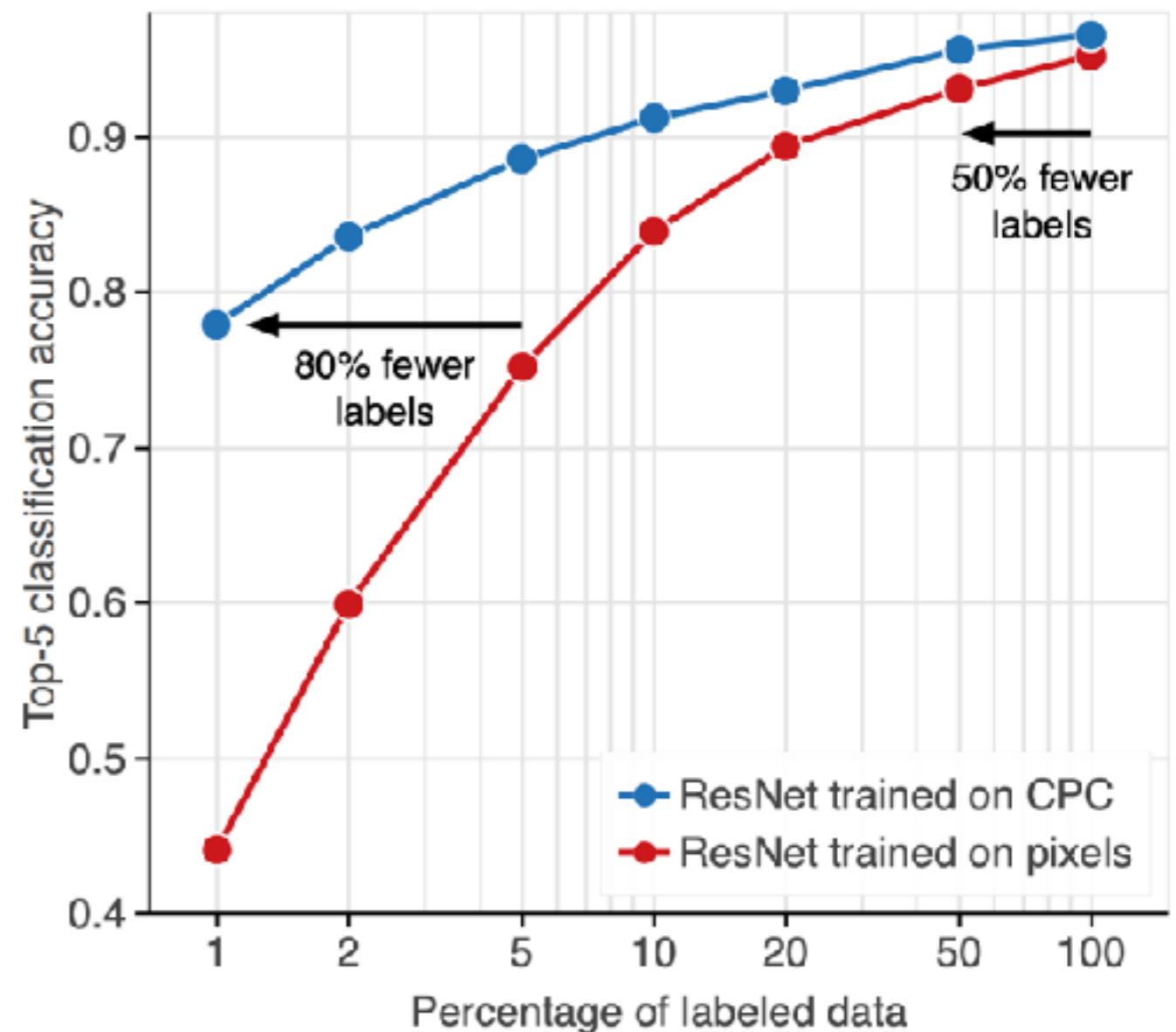


Can we match supervised classification without labels?

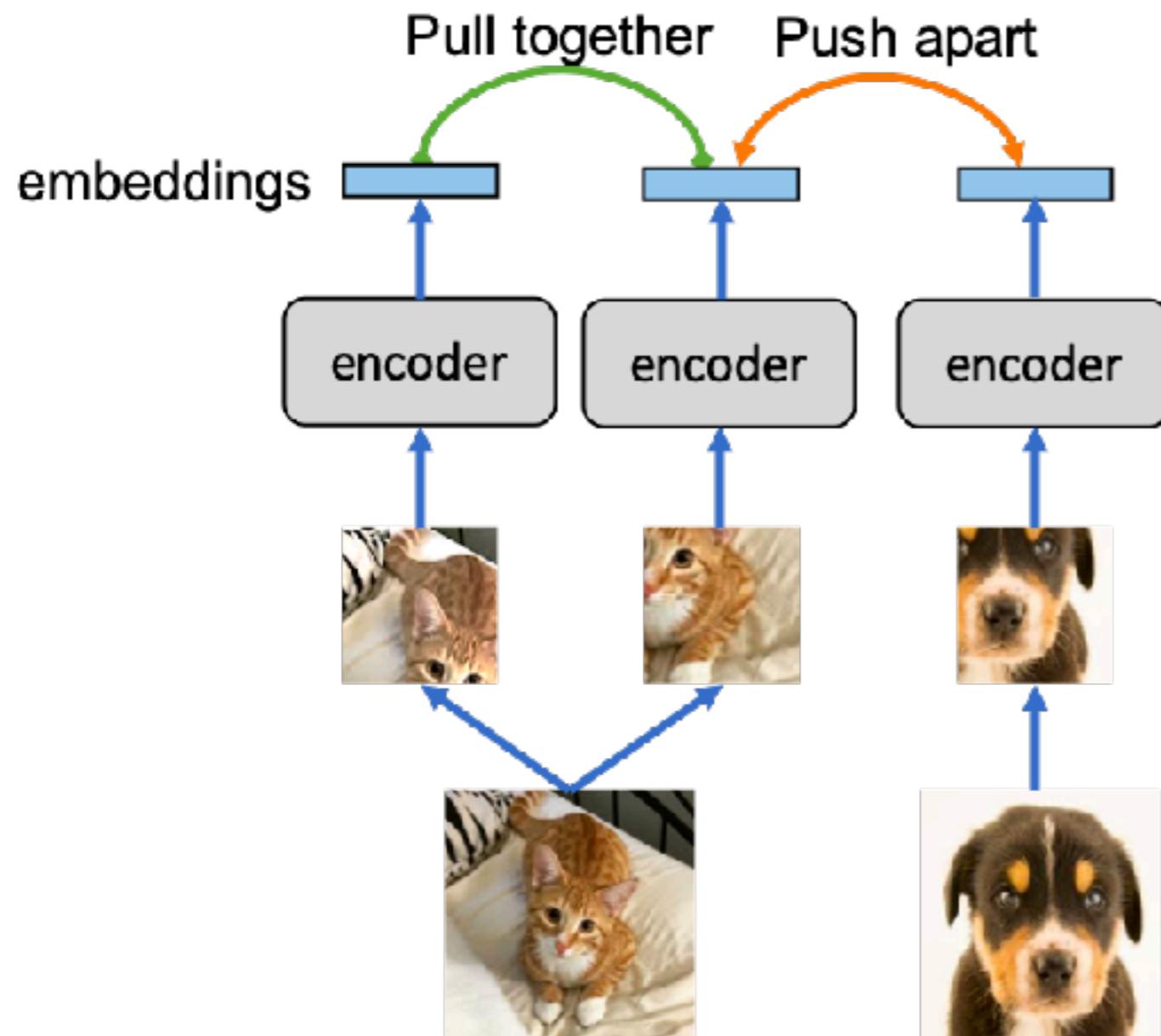
# Contrastive Predictive Coding

Match ImageNet performance  
with only 50% of the Labels!!

\*Surpass with all Labels  
\*(when fine-tuning the whole architecture)



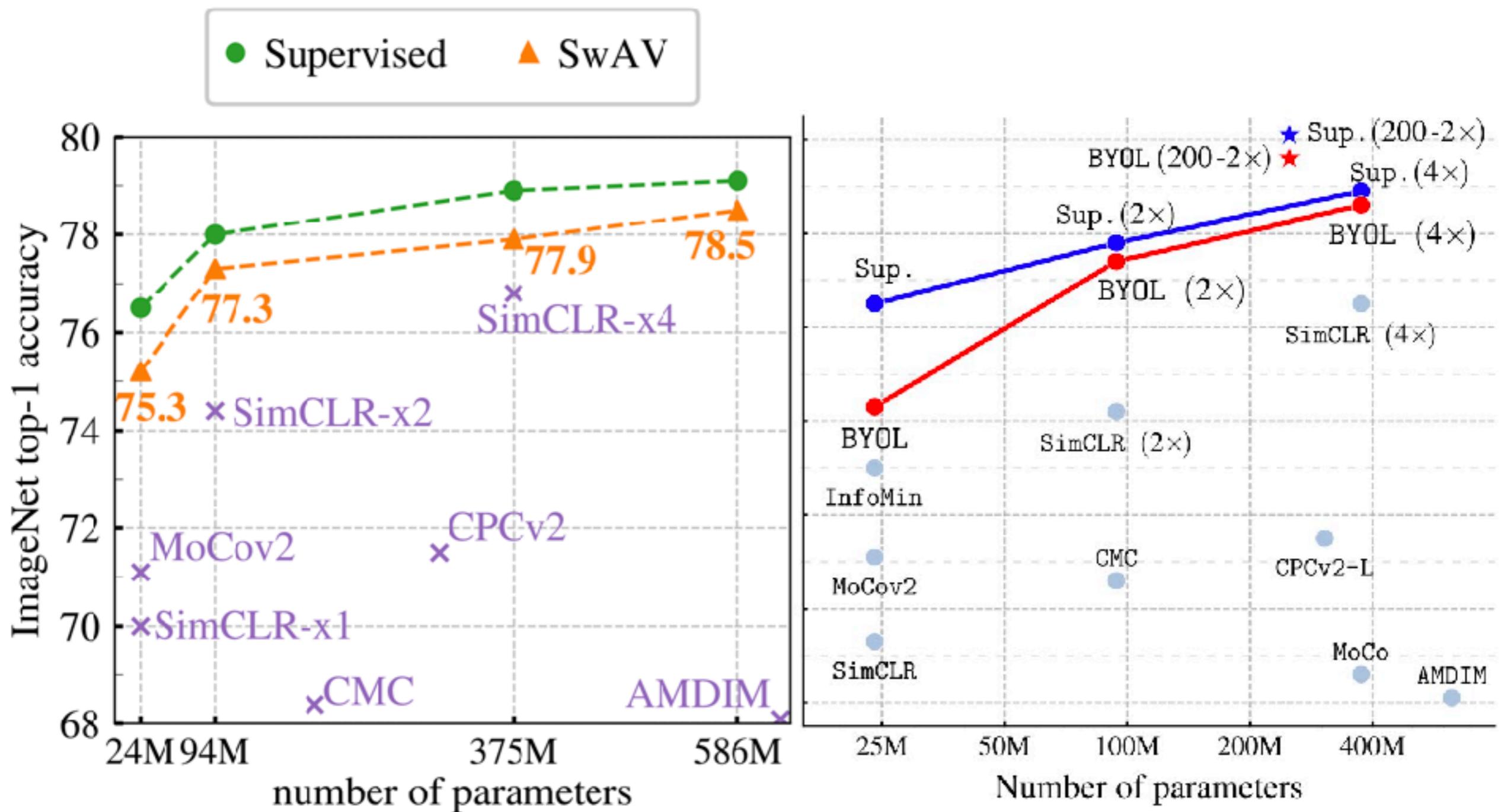
# MoCo and SimCLR (Contrastive Learning)



$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k^+ / \tau)}{\sum_{i=0}^K \exp(q \cdot k_i / \tau)}$$

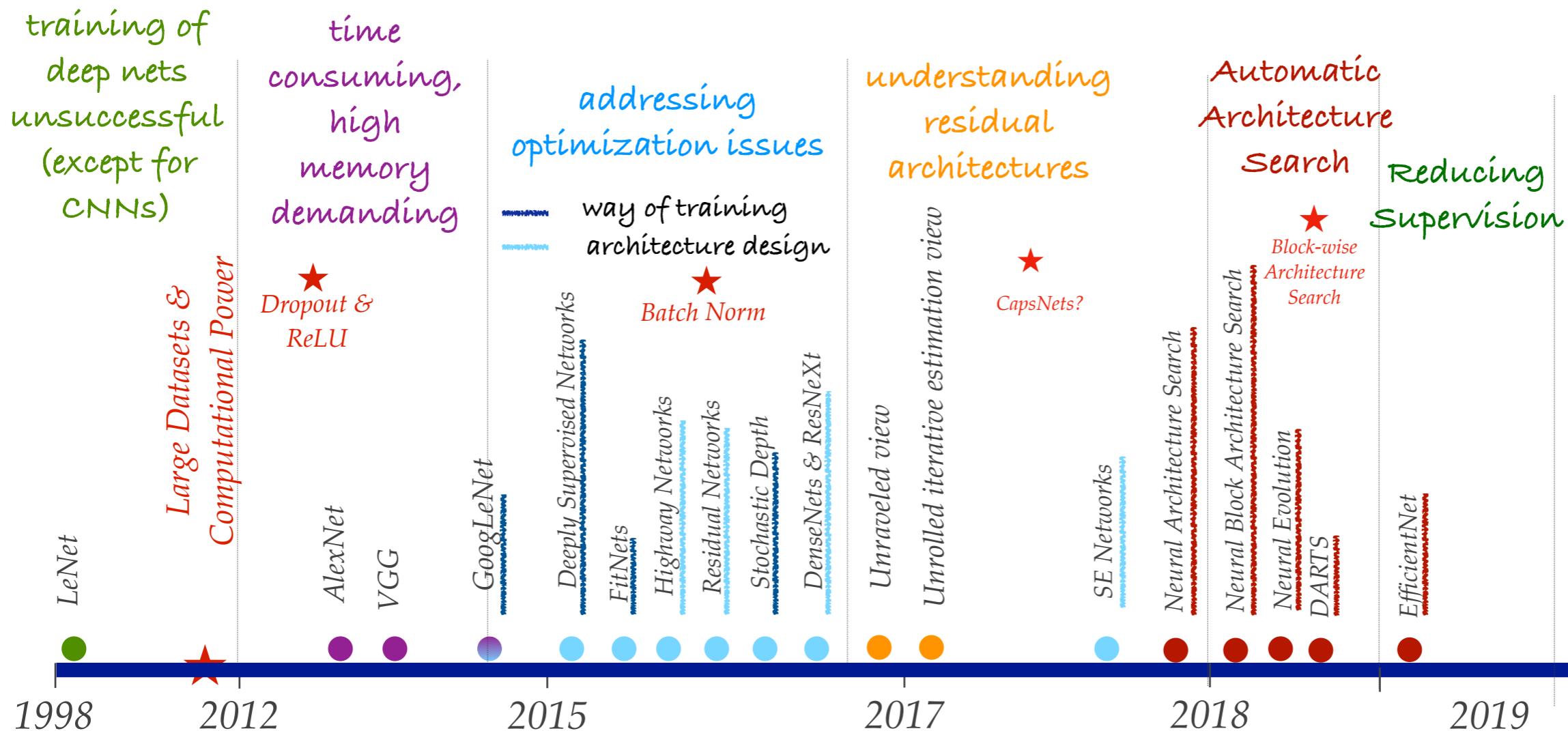
$q$  (query) = original image embedding  
 $k^+$  (pos. key) = transformations of the original image  
 $k_i$  (neg. keys) = other images in the dataset  
 $\tau$  = temperature hyperparameter

# Comparison of Self-Supervised Learning Methods Seen so Far

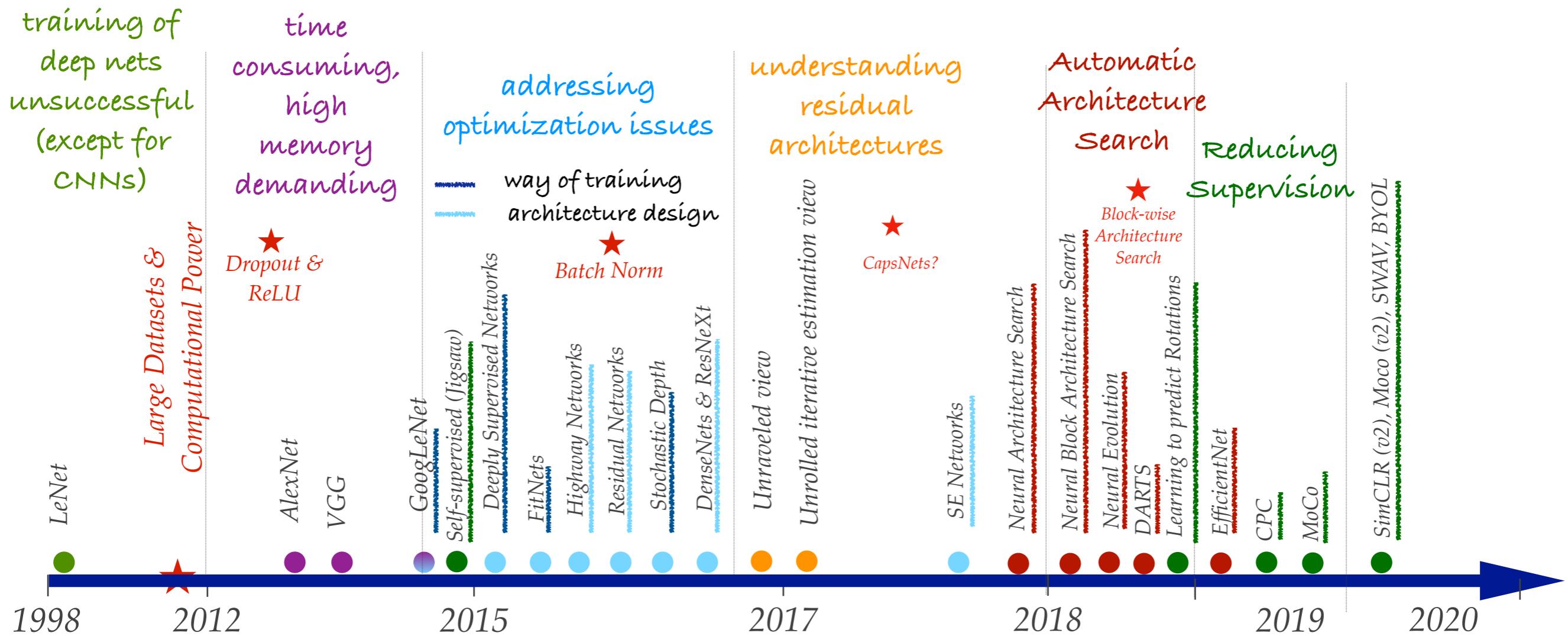


Frozen features + linear classifier accuracy

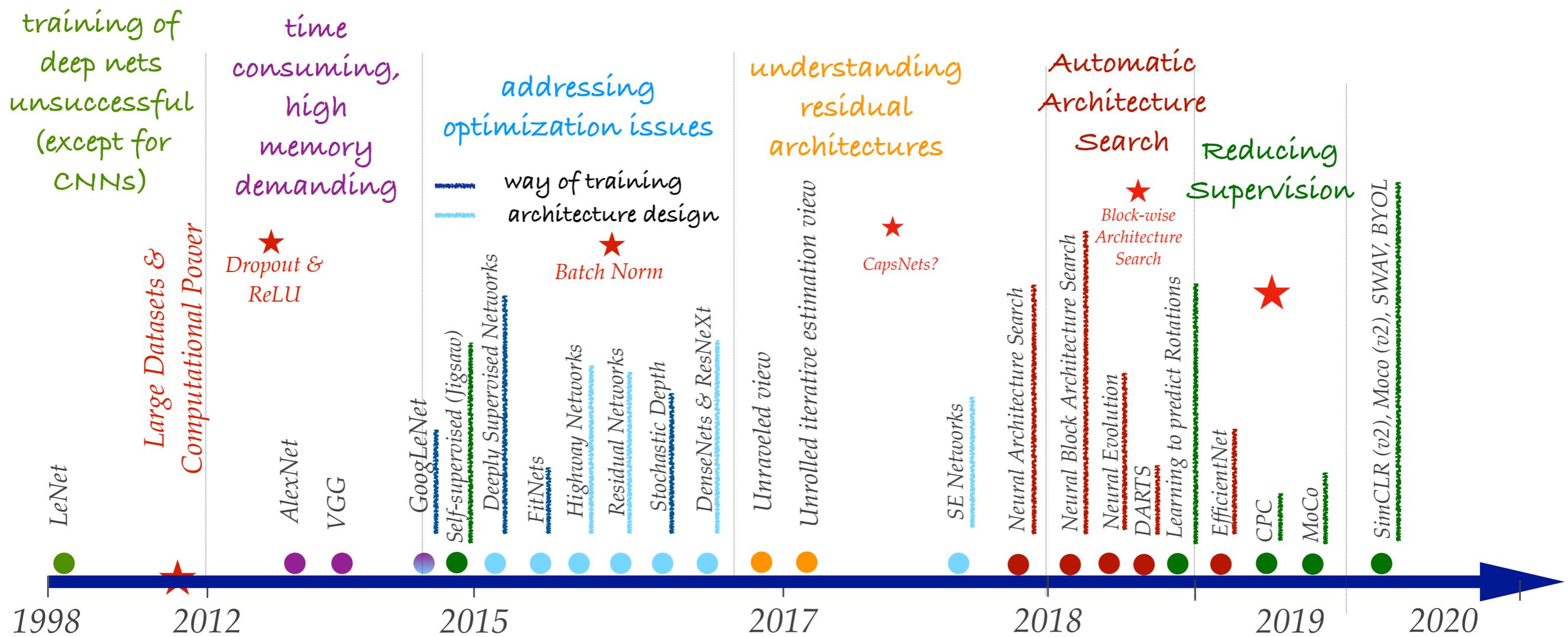
# Wrap Up



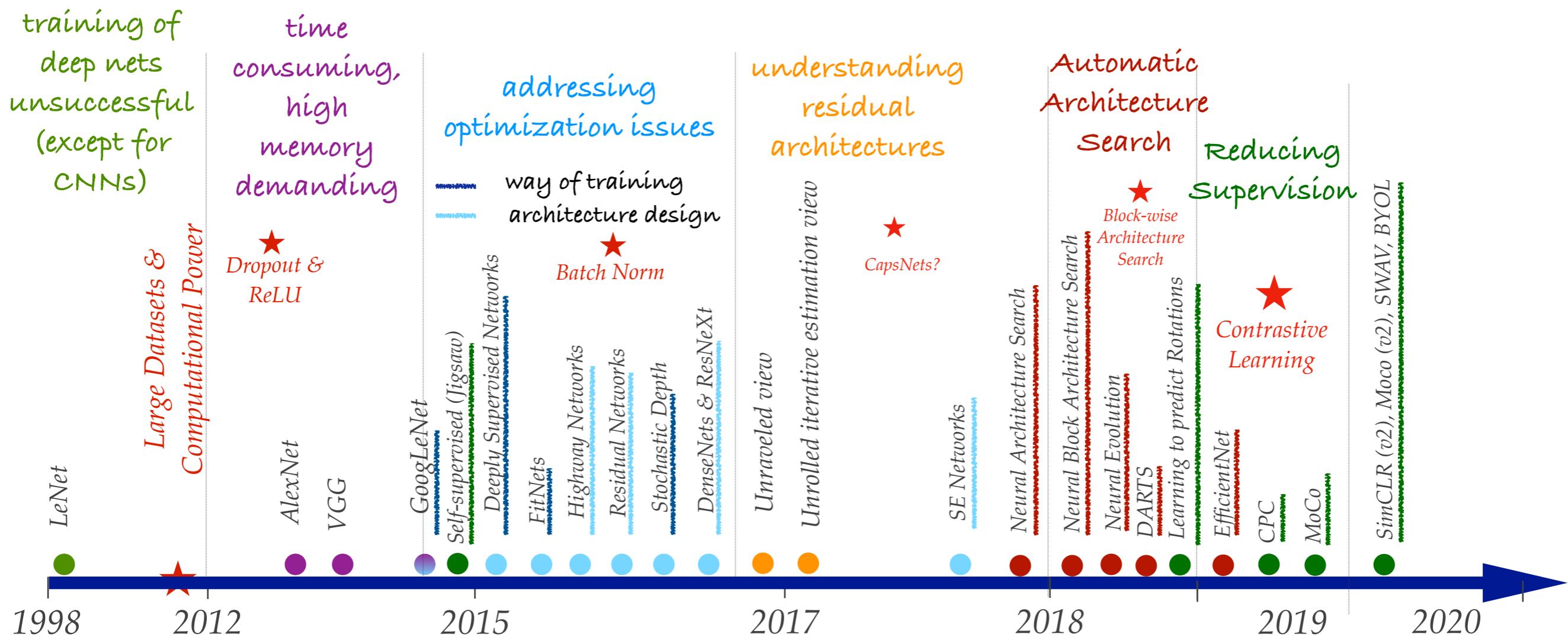
# Wrap Up



# Wrap Up



# Wrap Up



# Attention is All you Need

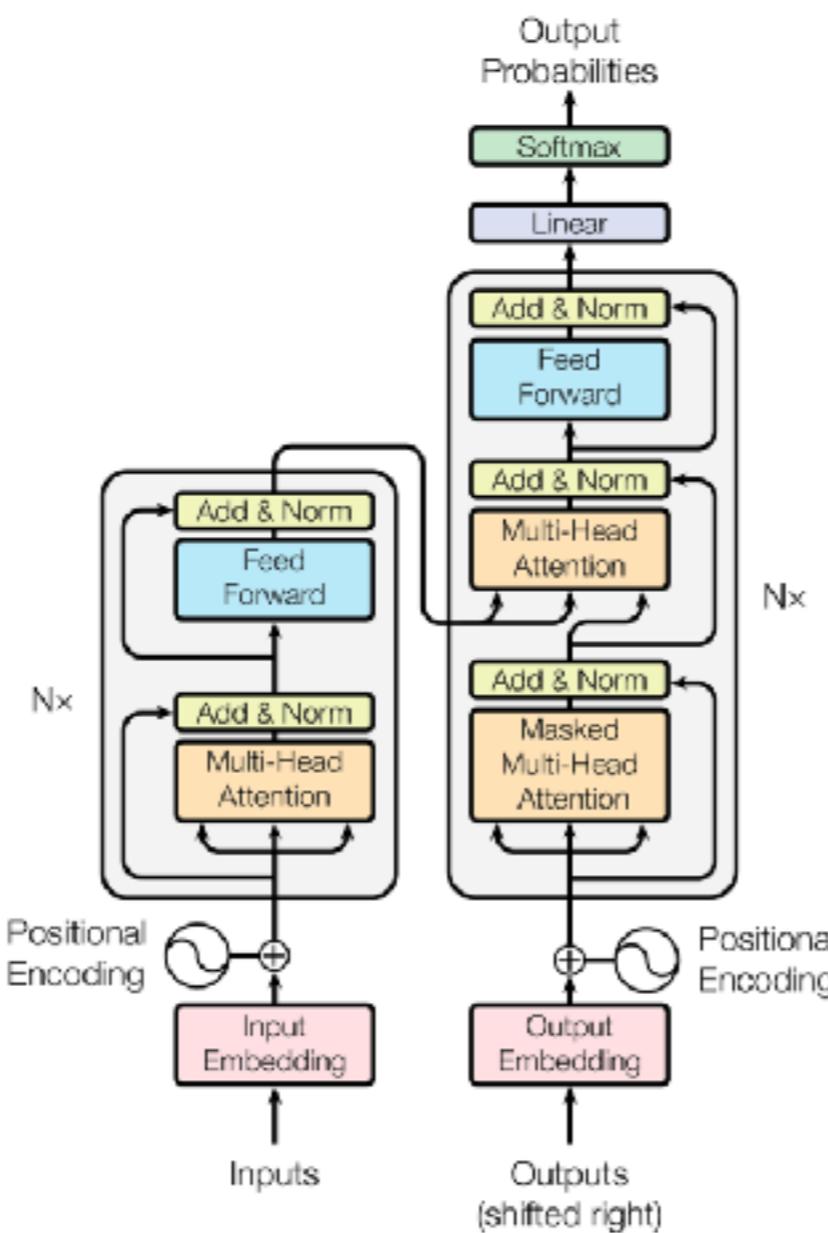


Figure 1: The Transformer - model architecture.

[Vaswani et al. 2017]

# Attention is All you Need

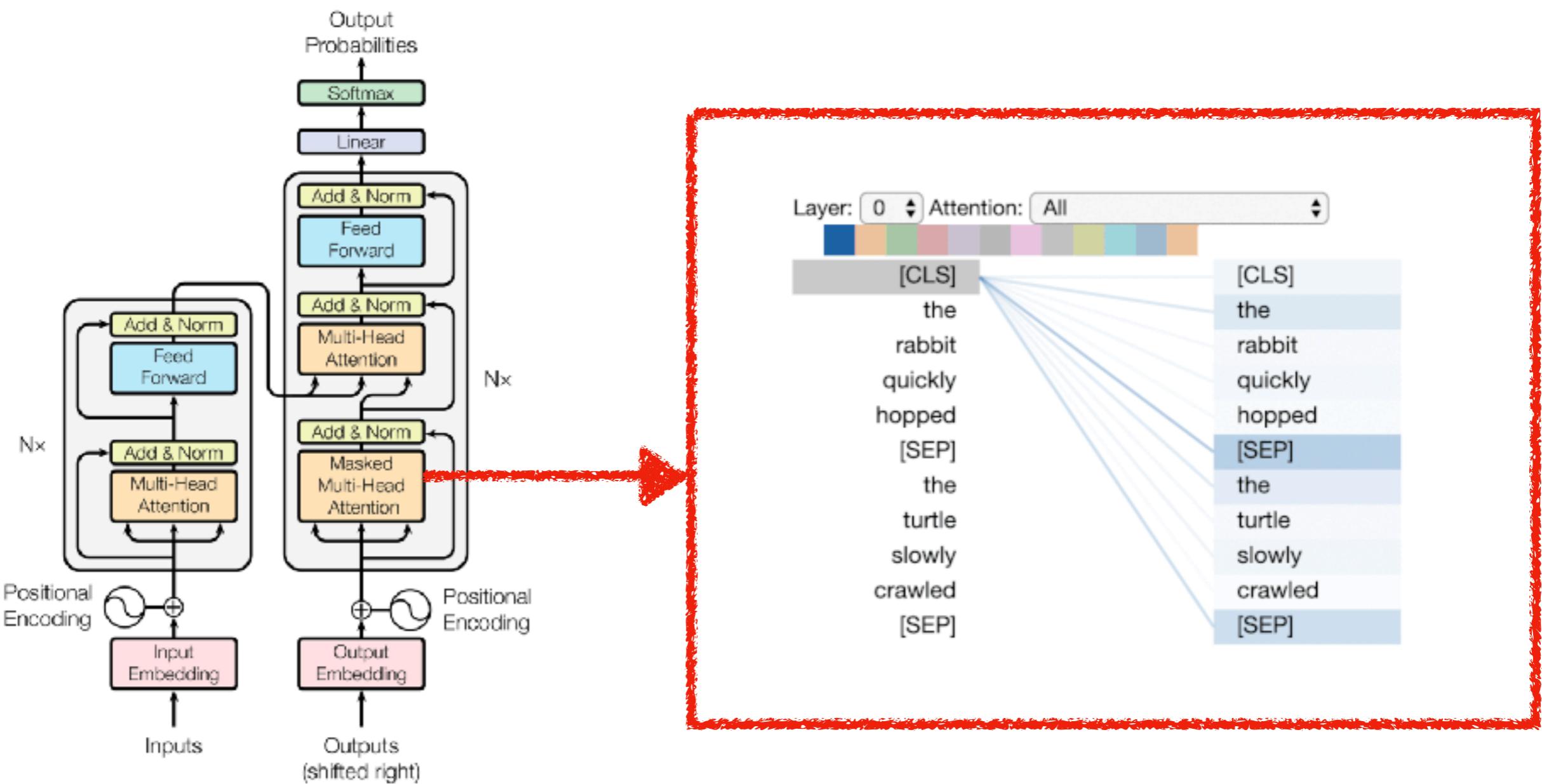


Figure 1: The Transformer - model architecture.

[Vaswani et al. 2017]

# Attention is All you Need

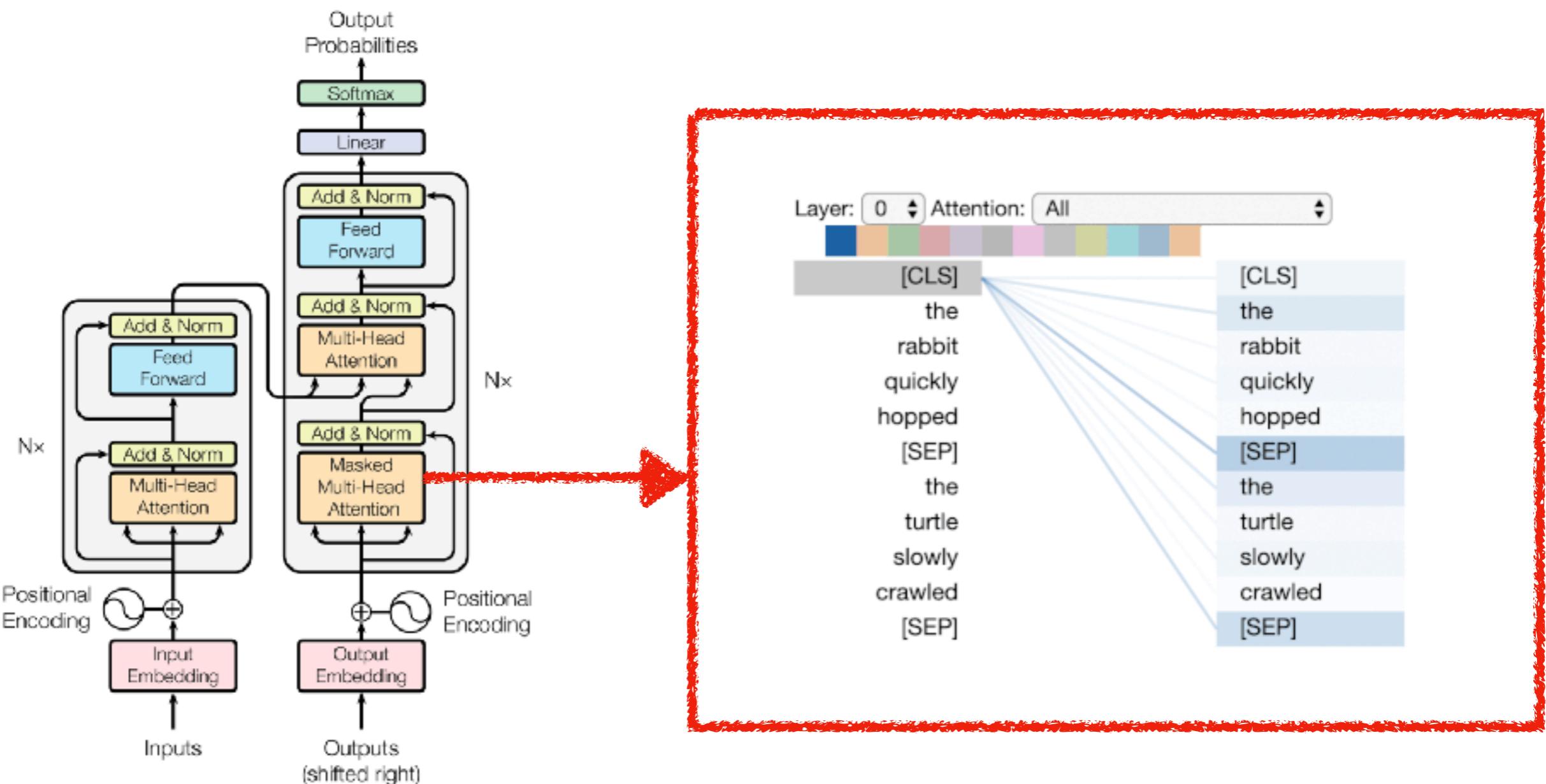


Figure 1: The Transformer - model architecture.

[Vaswani et al. 2017]

# Vision Transformer



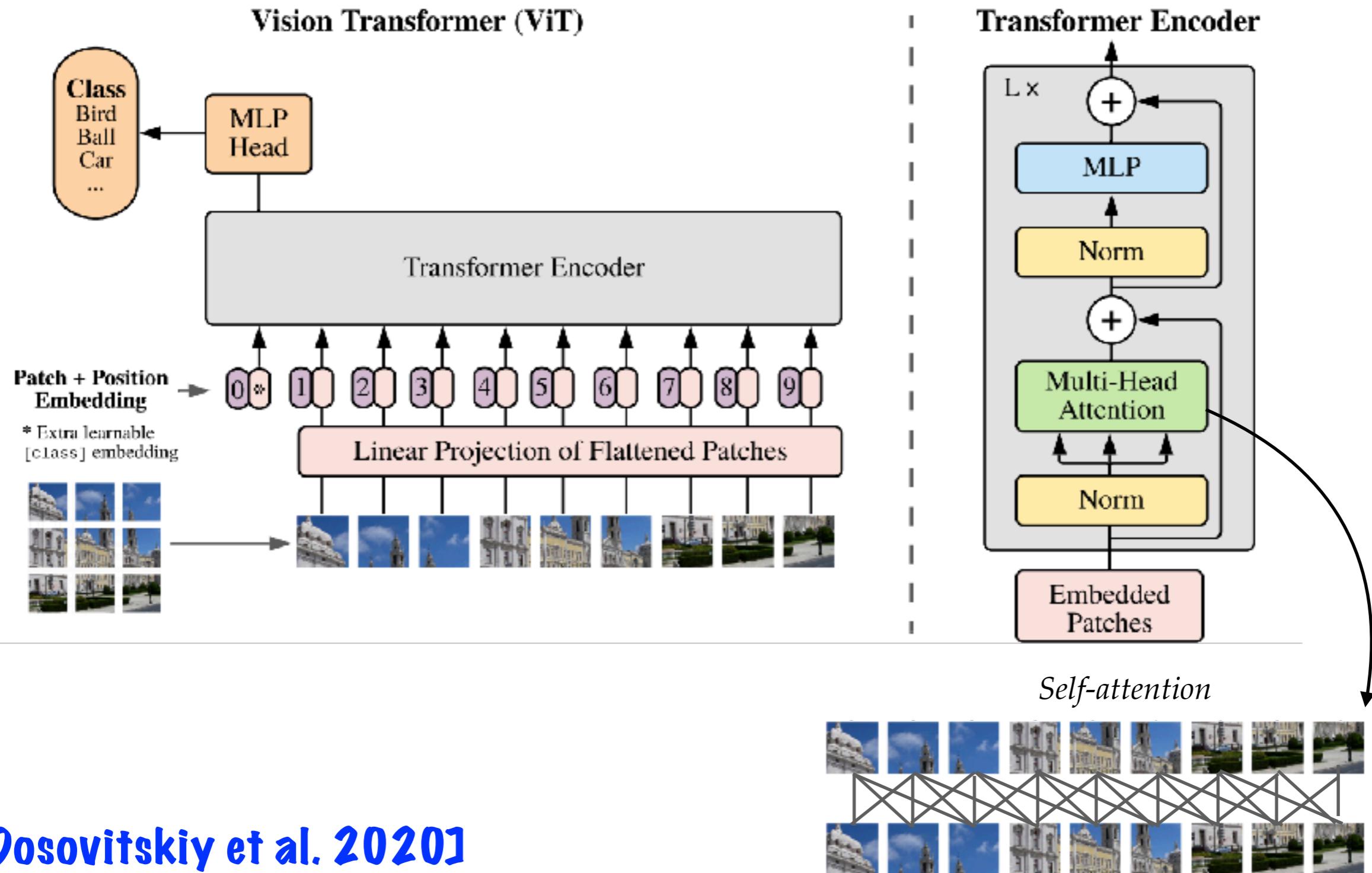
[Dosovitskiy et al. 2020]

# Vision Transformer

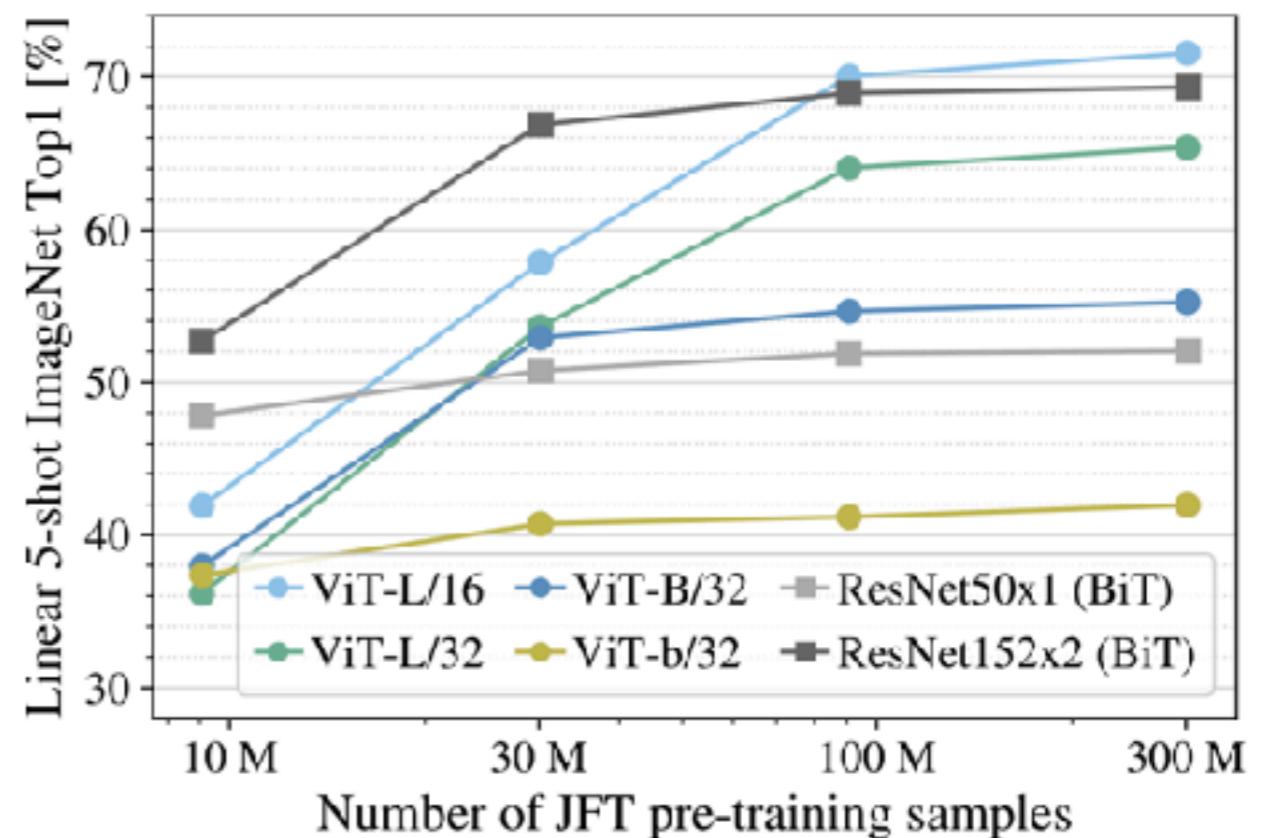
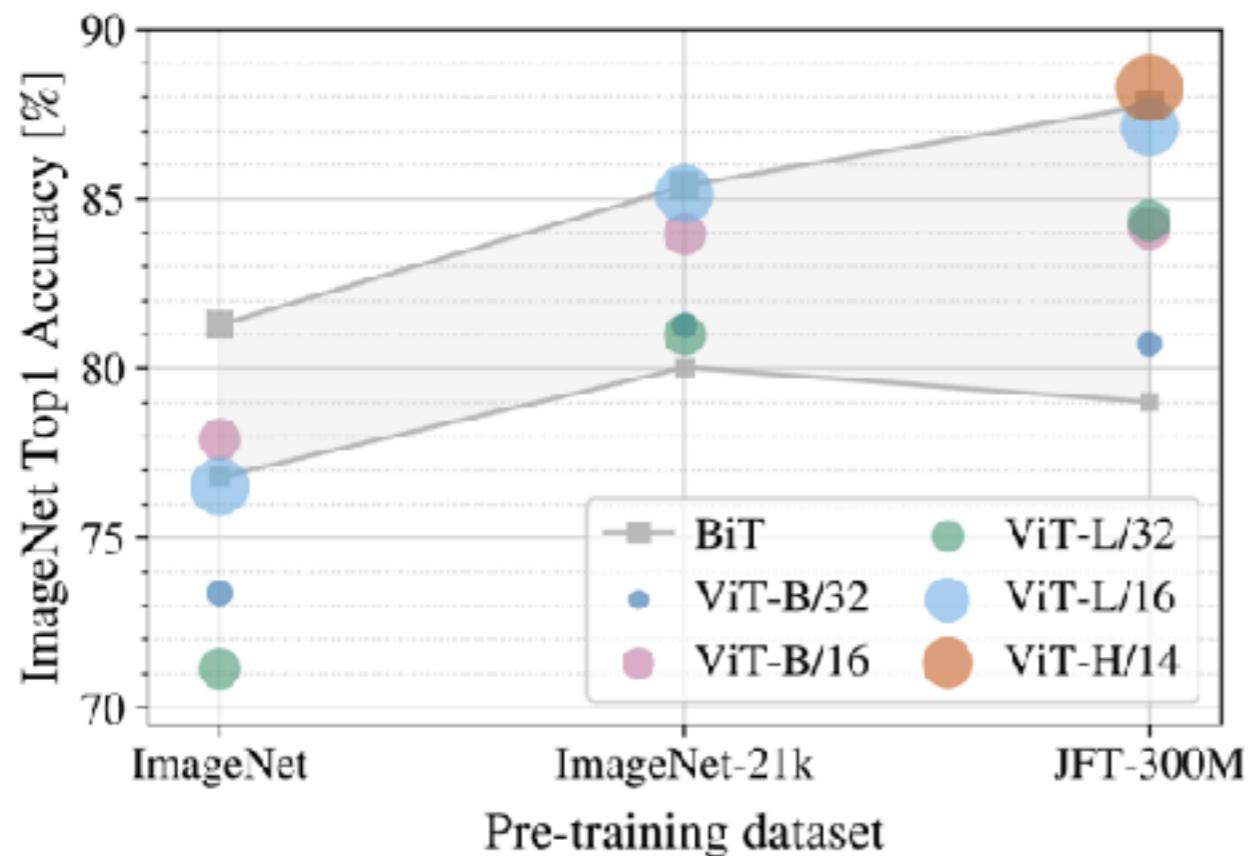


[Dosovitskiy et al. 2020]

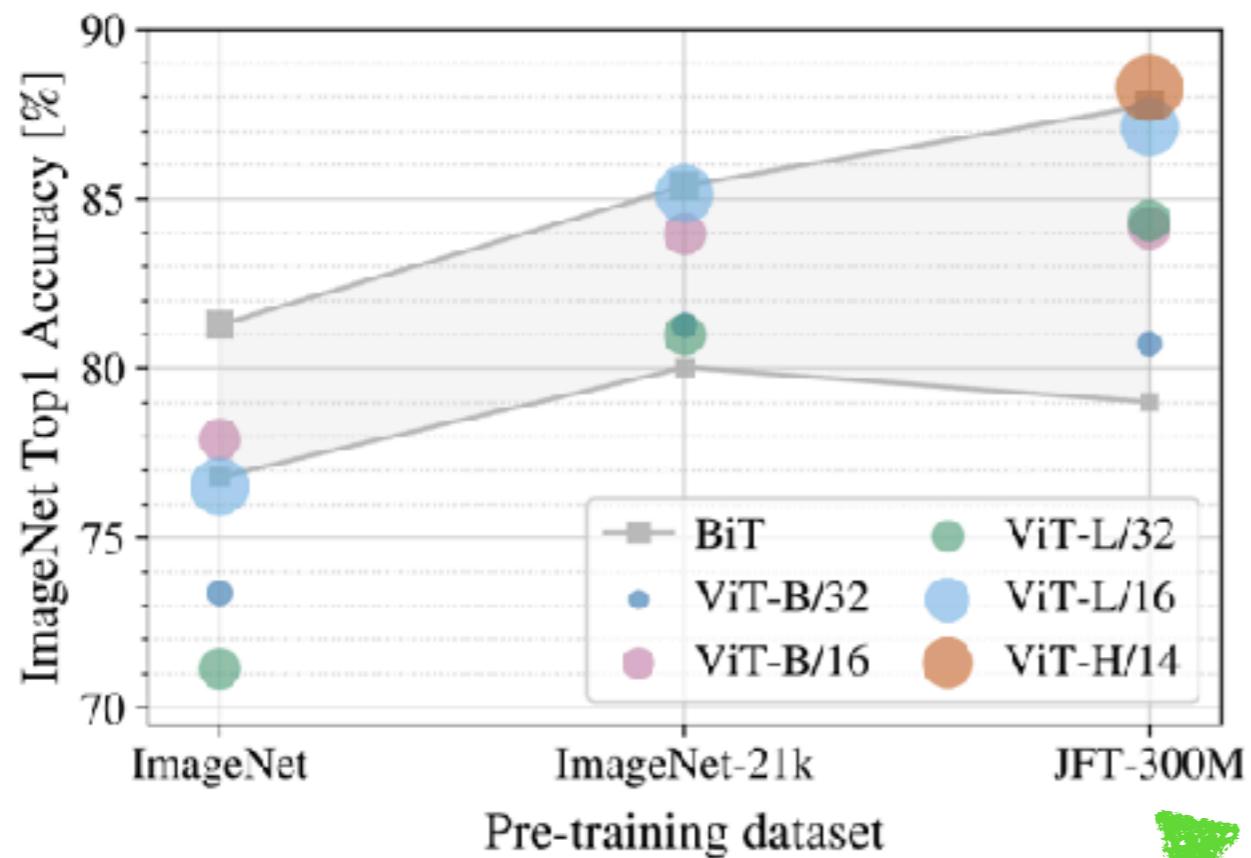
# Vision Transformer



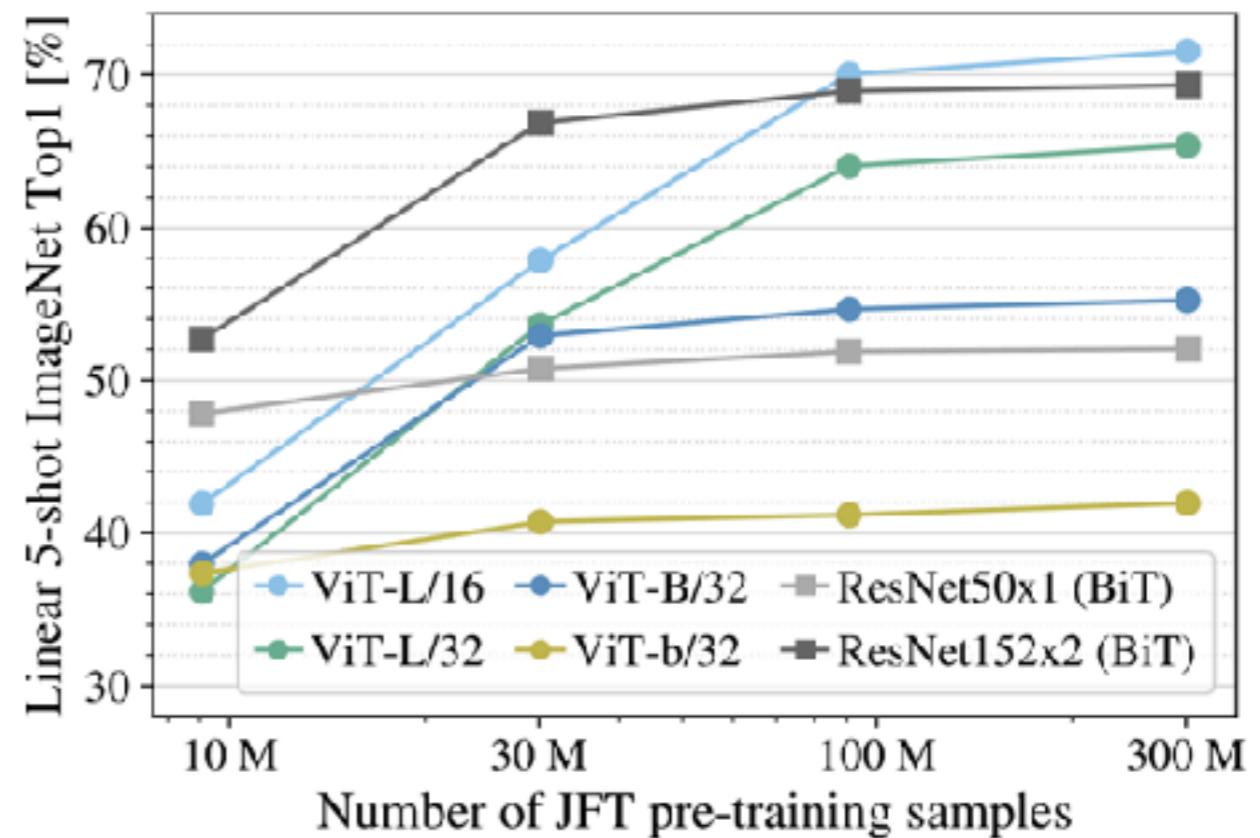
# Vision Transformer



# Vision Transformer

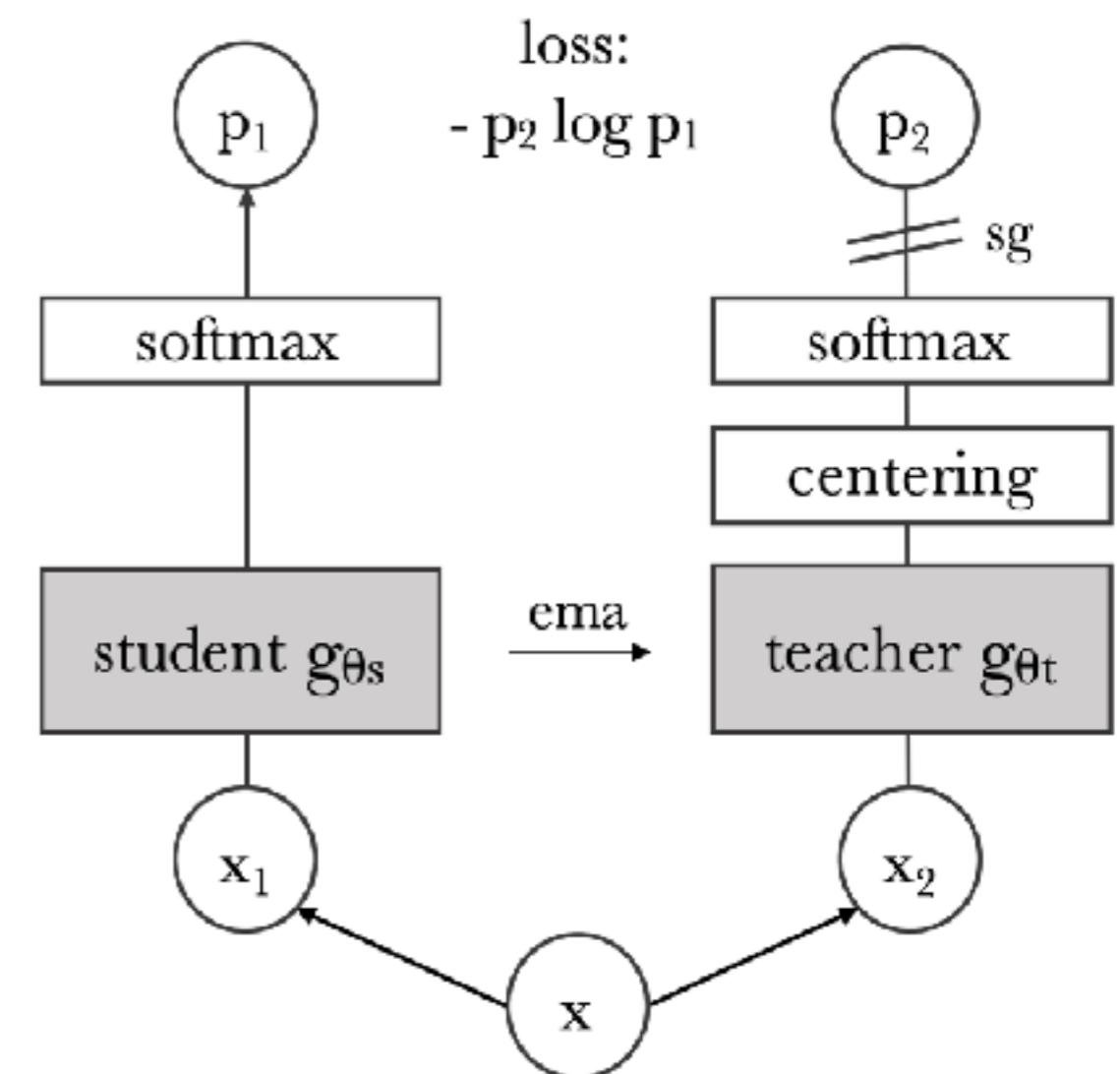


Shines when  
trained on larger  
datasets



# DINO: Transformers Meet Self-Supervised Learning

Student and teacher are both  
similar to Vision Transformer



# DINO: Transformers Meet Self-Supervised Learning

Method	Mom.	SK	MC	Loss	Pred.	$k$ -NN	Lin.
1 DINO	✓	✗	✓	CE	✗	72.8	76.1
2	✗	✗	✓	CE	✗	0.1	0.1
3	✓	✓	✓	CE	✗	72.2	76.0
4	✓	✗	✗	CE	✗	67.9	72.5
5	✓	✗	✓	MSE	✗	52.6	62.4
6	✓	✗	✓	CE	✓	71.8	75.6
7 BYOL	✓	✗	✗	MSE	✓	66.6	71.4
8 MoCov2	✓	✗	✗	INCE	✗	62.0	71.6
9 SwAV	✗	✓	✓	CE	✗	64.7	71.8

SK: Sinkhorn-Knopp, MC: Multi-Crop, Pred.: Predictor  
CE: Cross-Entropy, MSE: Mean Square Error, INCE: InfoNCE

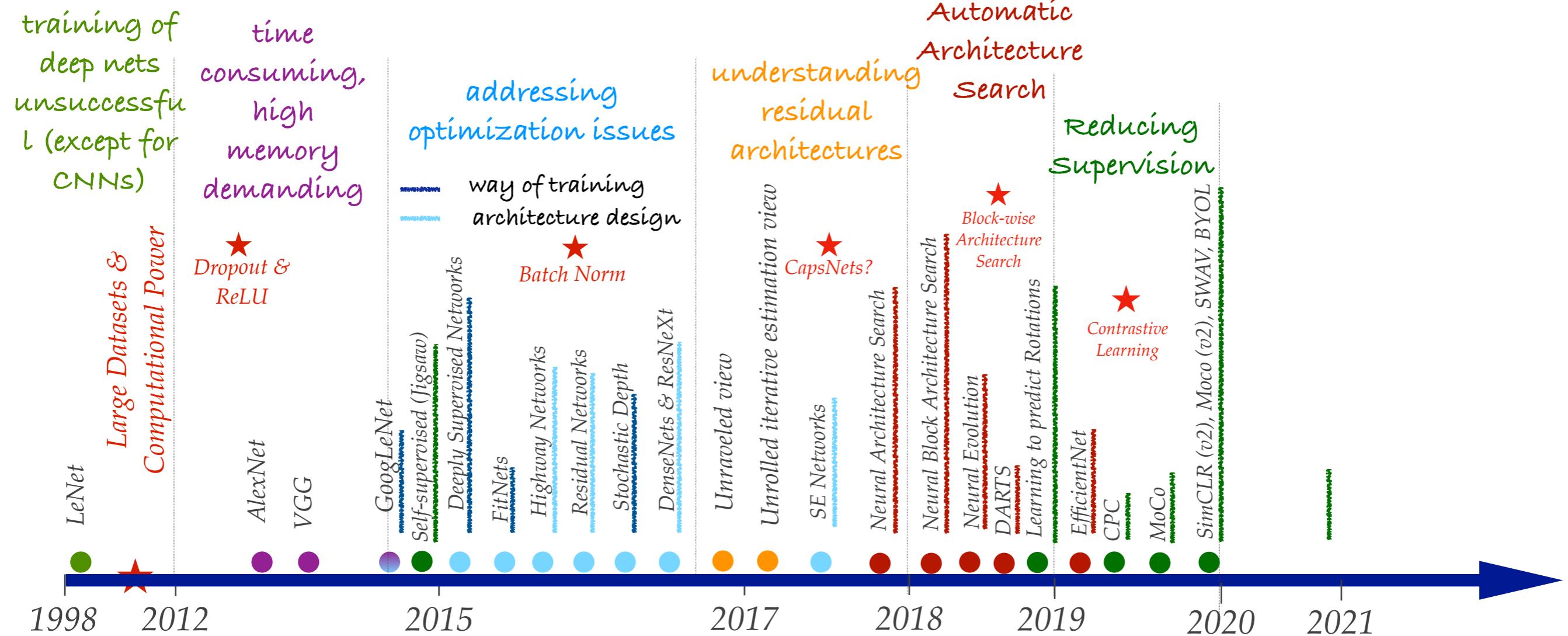
# DINO: Visualizing Attention Masks



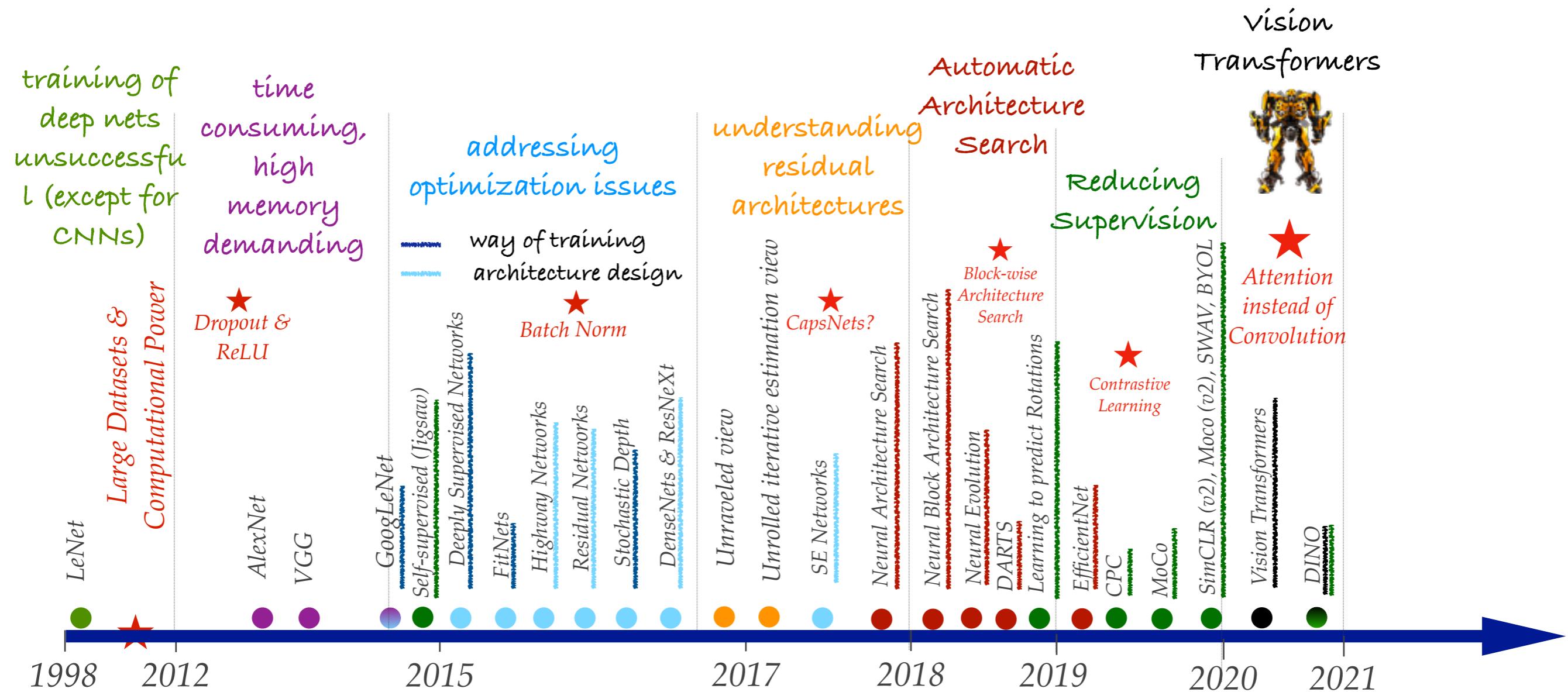
# DINO: Visualizing Attention Masks



# Wrap Up

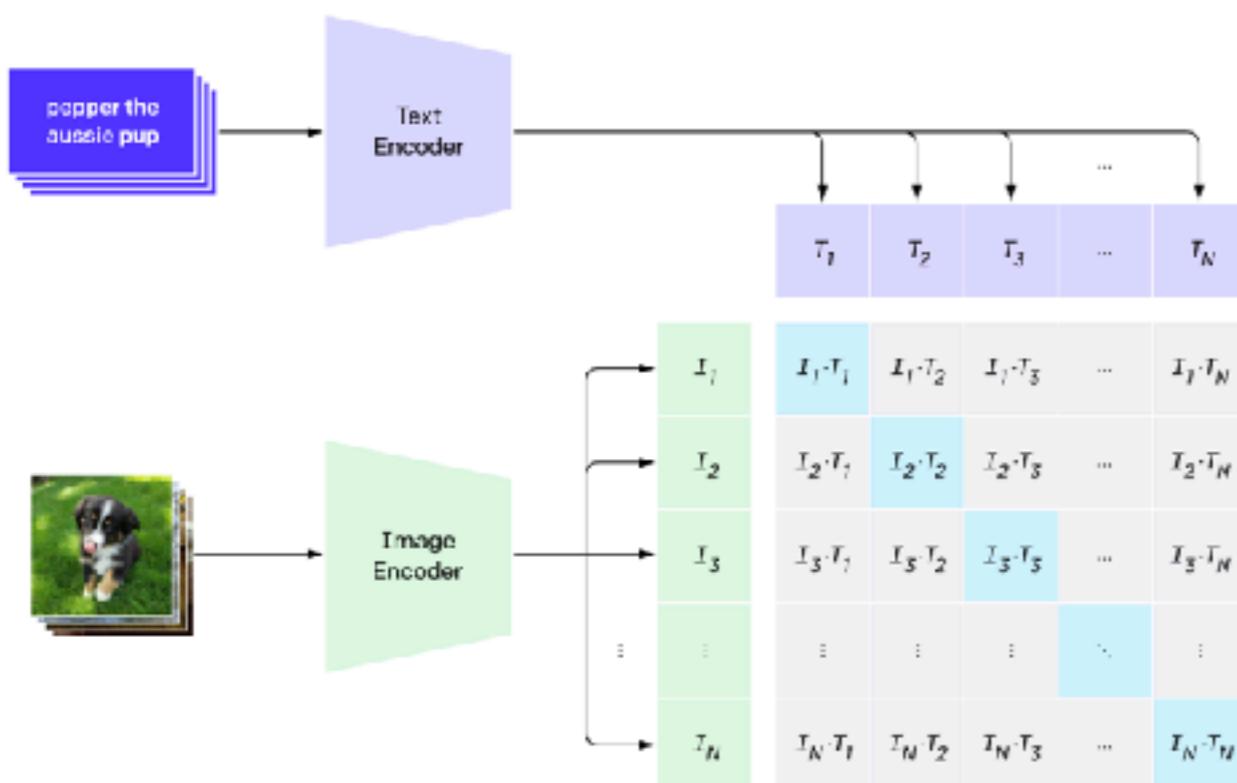


# Wrap Up

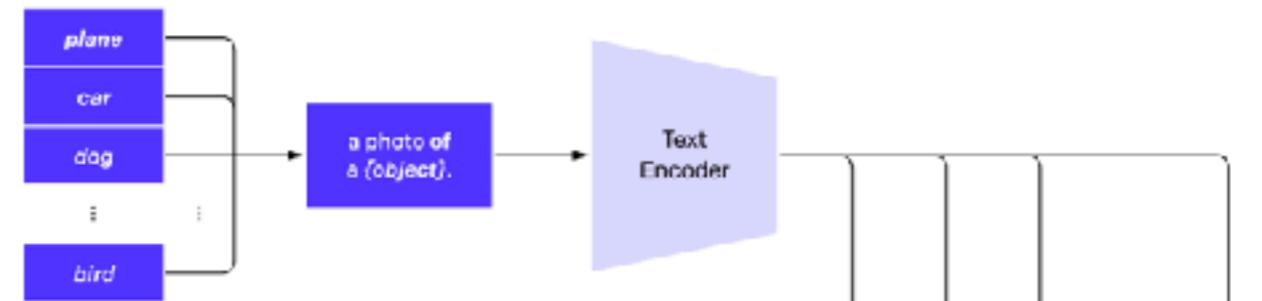


# CLIP

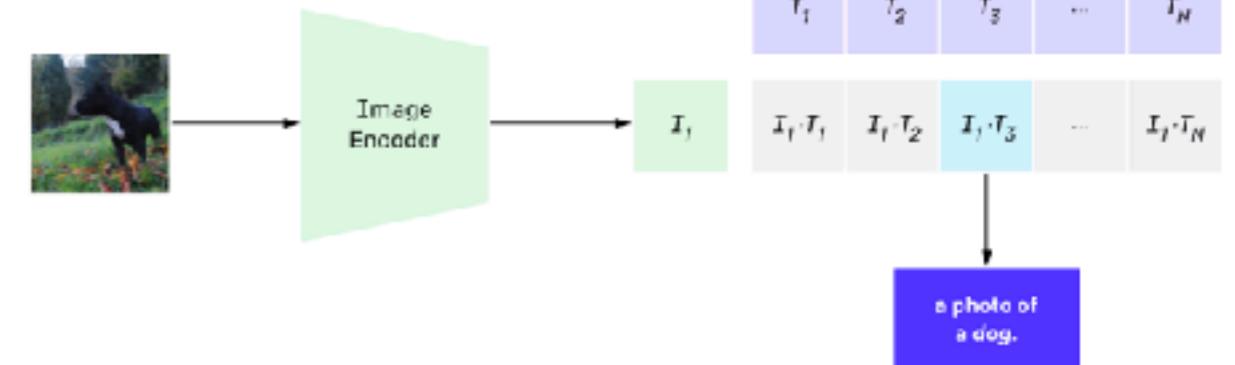
## 1. Contrastive pre-training



## 2. Create dataset classifier from label text

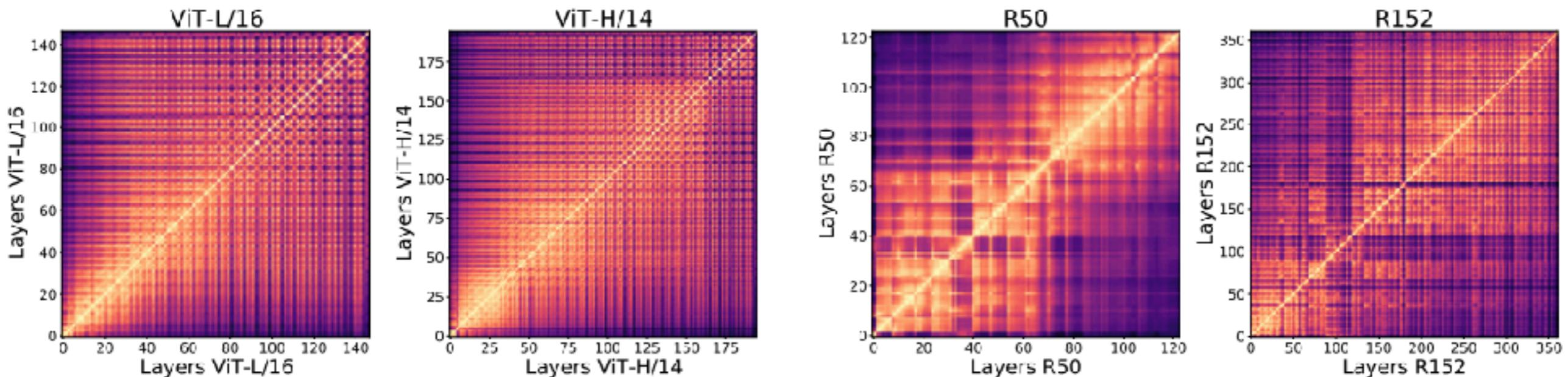


## 3. Use for zero-shot prediction



# Do Vision Transformers See Like Convolutional Neural Networks?

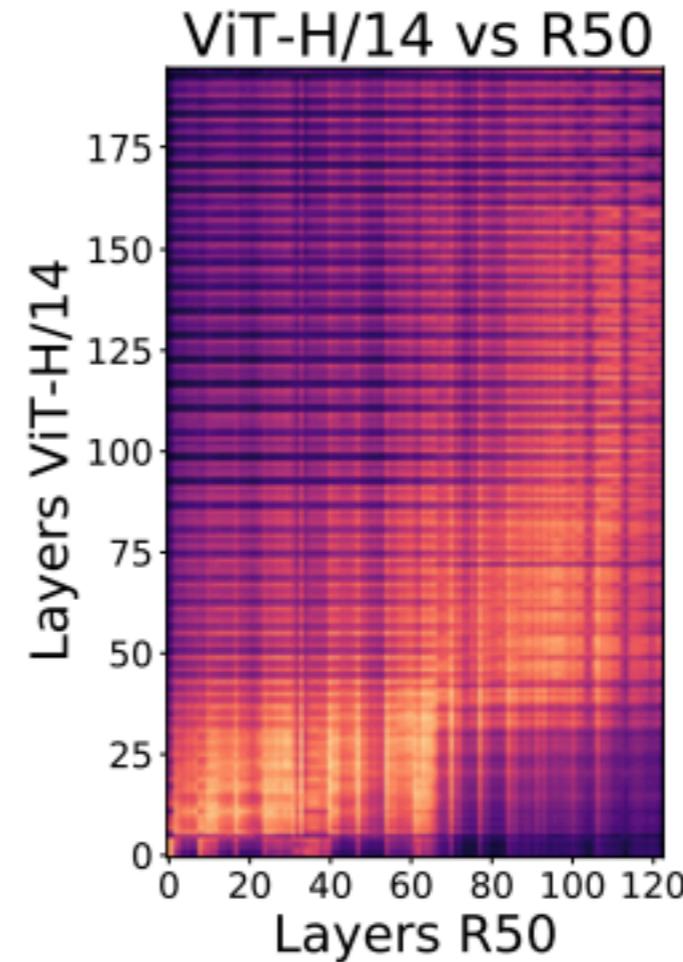
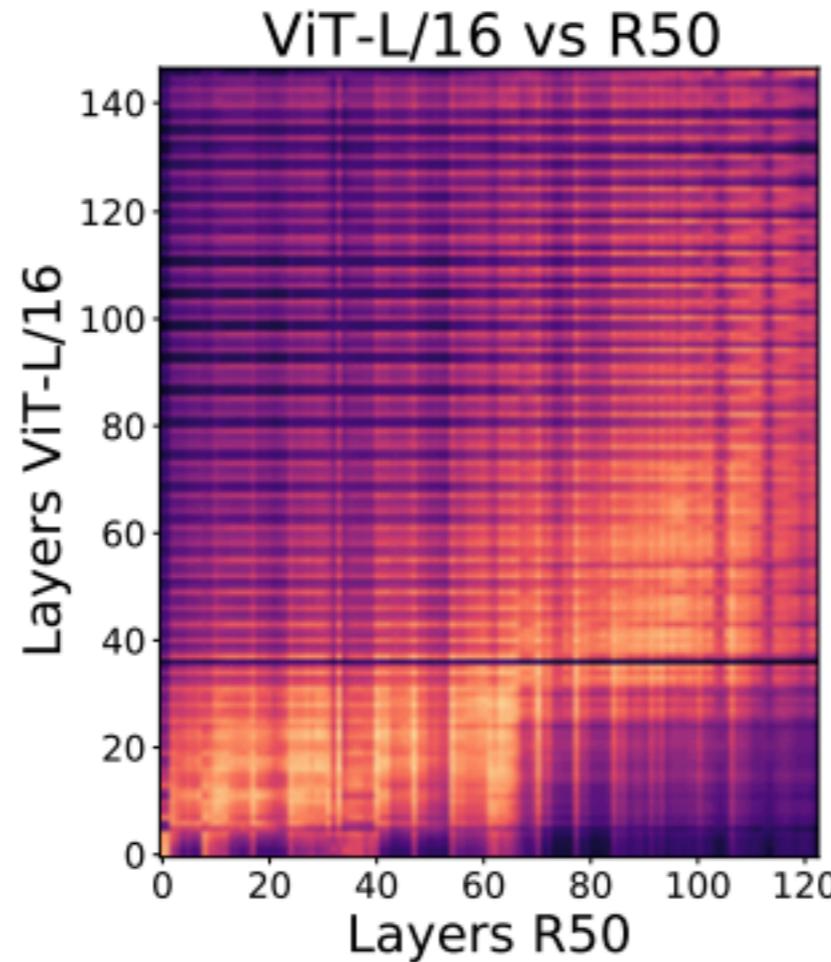
*Similarity between representations at different depths with Centered Kernel Alignment (CKA)*



Transformer representations  
at different depths are similar

CNNs: different representations at  
different depths

# Do Vision Transformers See Like Convolutional Neural Networks?

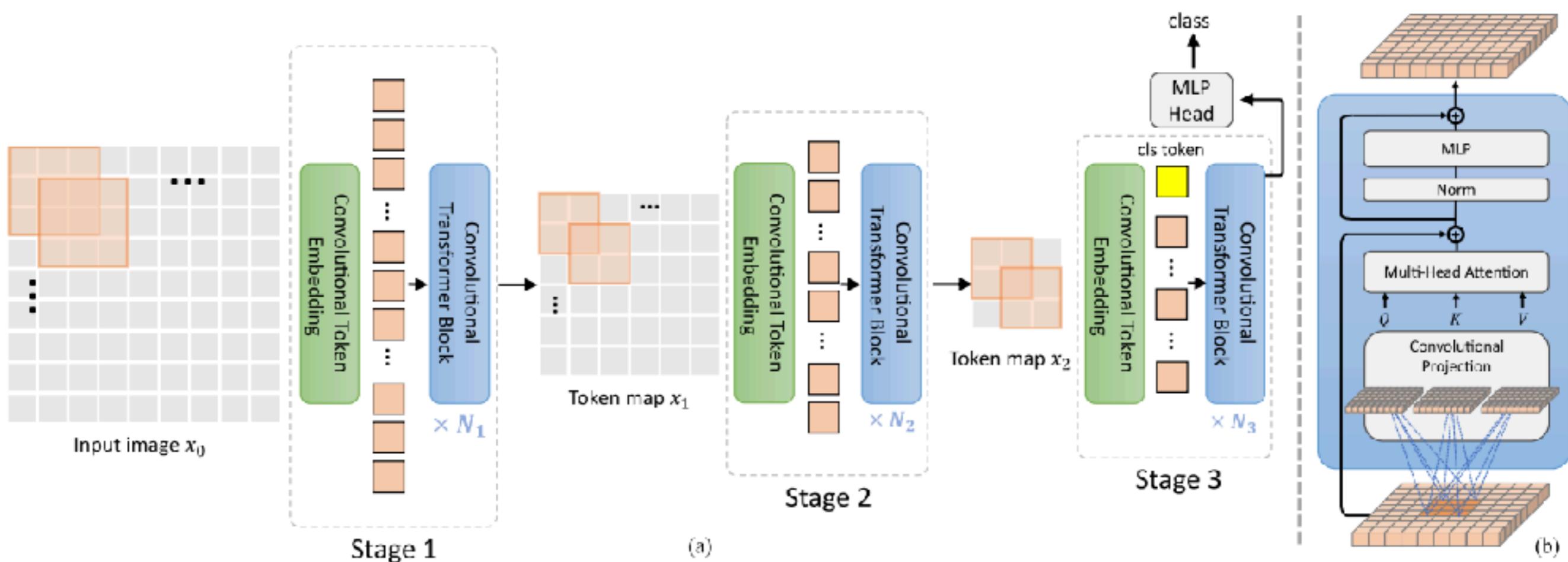


First layers of ViT resemble ResNet!

Transformers might end up learning something similar to convolutions!

BUT: With less training data, lower attention layers do not learn to attend locally!!!! So learning to do convolutions from scratch is data expensive!

# Going Back to Convolutions?



CvT: Introducing Convolutions to Vision Transformers

# Going Back to Convolutions?

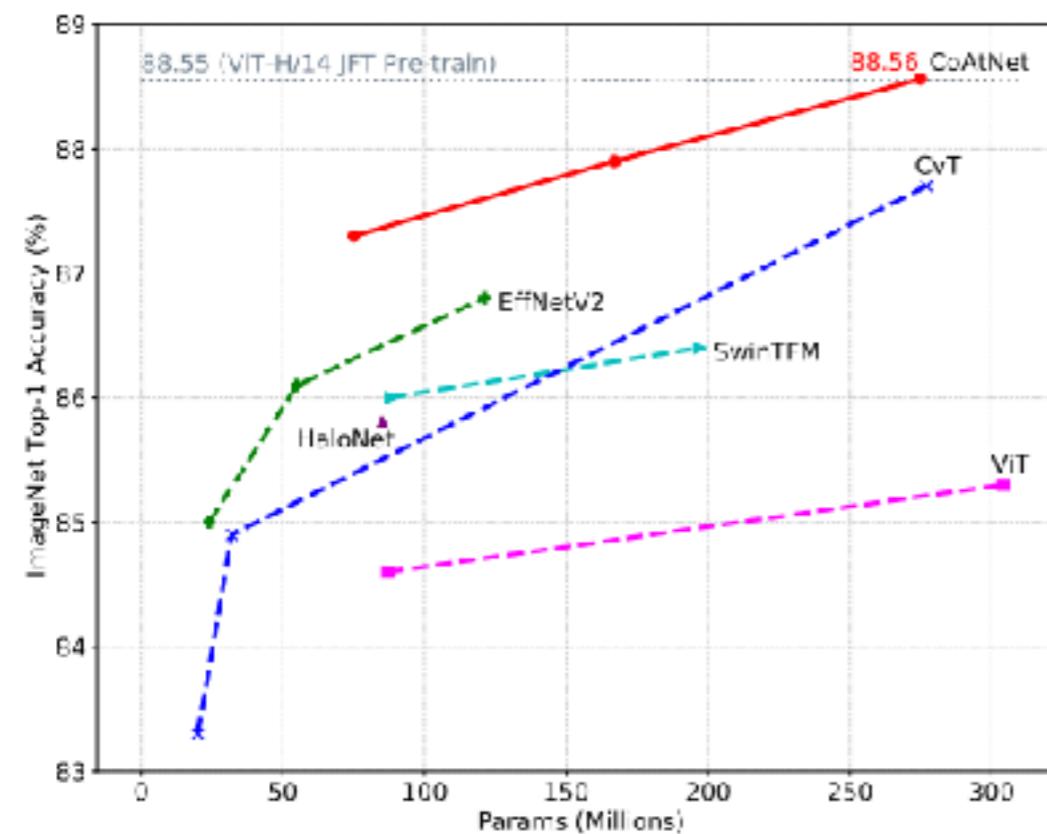
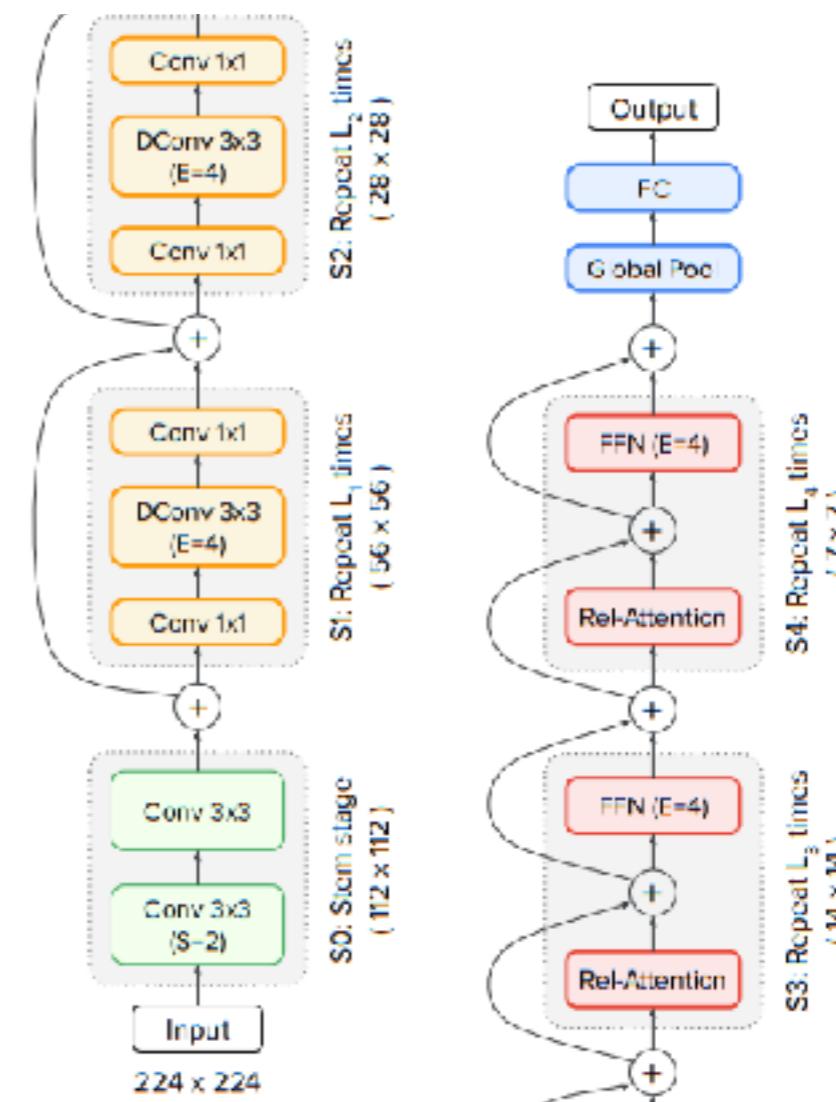


Figure 3: Accuracy-to-Params scaling curve under  $\text{ImageNet-21K} \Rightarrow \text{ImageNet-1K}$  setting.

Transformers become more data efficient  
with some convolutions

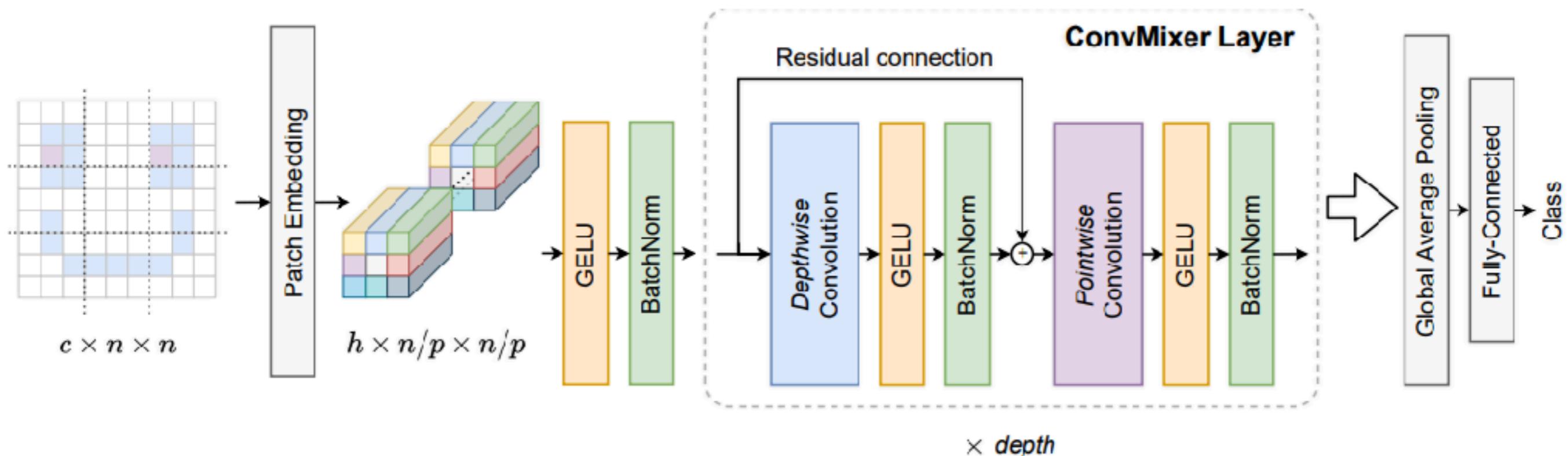


CoAtNet: Marrying Convolution and  
Attention for All Data Sizes

[Dai et al. 2021]

# Do We Need Attention?

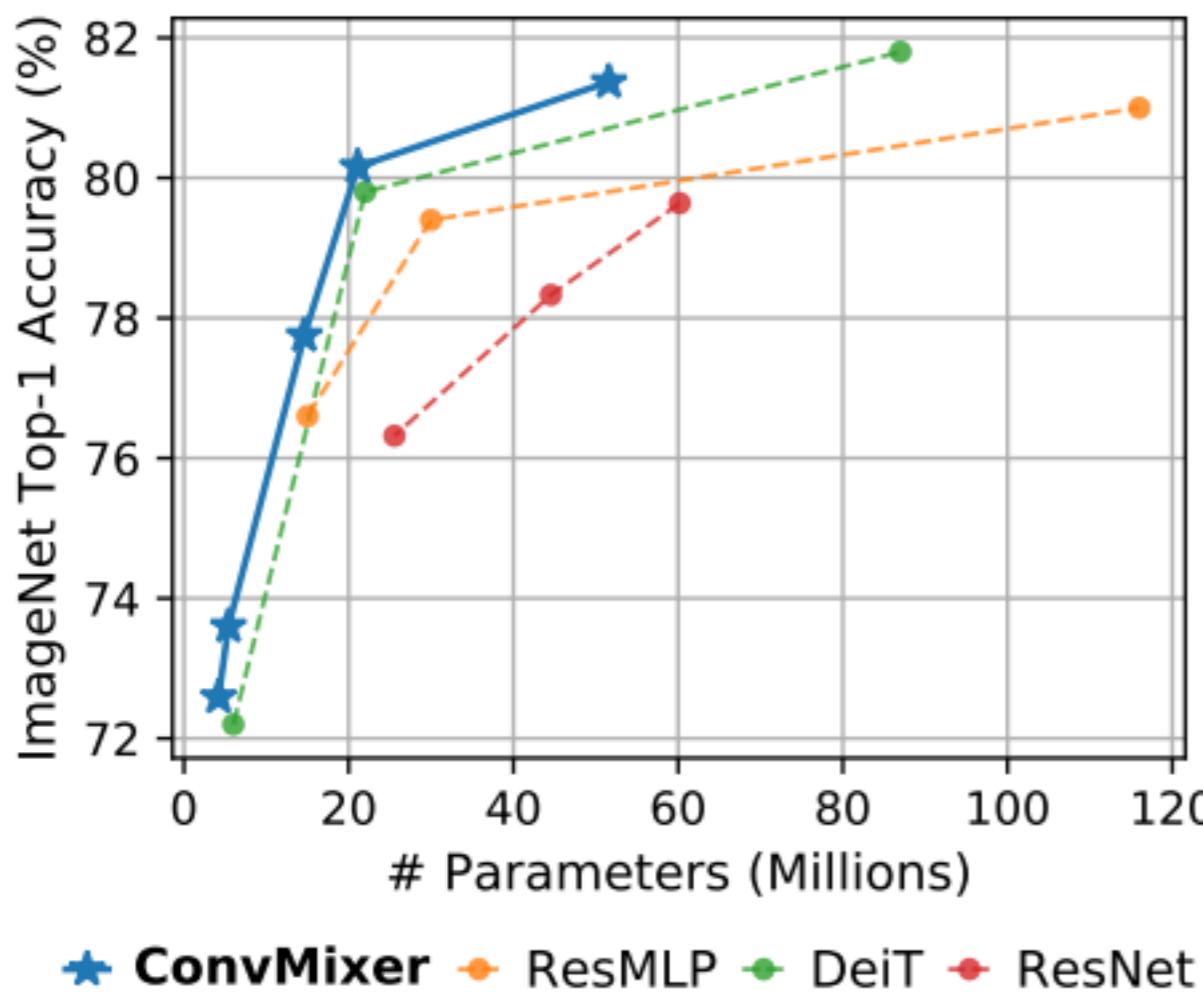
CONVOLUTIONS ATTENTION MLPs  
PATCHES ARE ALL YOU NEED? 🤖



```
1 def ConvMixer(h, depth, kernel_size=9, patch_size=7, n_classes=1000):
2     Seq, ActBn = nn.Sequential, lambda x: Seq(x, nn.GELU(), nn.BatchNorm2d(h))
3     Residual = type('Residual', (Seq,), {'forward': lambda self, x: self[0](x) + x})
4     return Seq(ActBn(nn.Conv2d(3, h, patch_size, stride=patch_size)),
5                *[Seq(Residual(ActBn(nn.Conv2d(h, h, kernel_size, groups=h, padding="same"))),
6                      ActBn(nn.Conv2d(h, h, 1))) for i in range(depth)],
7                nn.AdaptiveAvgPool2d((1,1)), nn.Flatten(), nn.Linear(h, n_classes))
```

# Do We Need Attention?

CONVOLUTIONS ATTENTION MLPs  
PATCHES ARE ALL YOU NEED? 🧑



## MLP-Mixer: An all-MLP Architecture for Vision

Ilya Tolstikhin\*, Neil Houlsby\*, Alexander Kolesnikov\*, Lucas Beyer\*,  
Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner,  
Daniel Keysers, Jakob Uszkoreit, Mario Lucic, Alexey Dosovitskiy

\*equal contribution

Google Research, Brain Team

{tolstikhin, neilhoulsby, akolesnikov, lbeier,  
xzhai, unterthiner, jessicayung<sup>†</sup>, andstein,  
keysers, usz, lucic, adosovitskiy}@google.com

<sup>†</sup>work done during Google AI Residency



Andrej Karpathy



@karpathy

...

Errr ok wow, I am shook by the new ConvMixer architecture

[openreview.net/forum?id=TVHS5](https://openreview.net/forum?id=TVHS5)... "the first model that achieves the elusive dual goals of 80%+ ImageNet top-1 accuracy while also fitting into a tweet" 😲

# Do We Need Transformers?

Improvements	Top-1	$\Delta$
ResNet-200	79.0	—
+ Cosine LR Decay	79.3	+0.3
+ Increase training epochs	78.8 <sup>†</sup>	-0.5
+ EMA of weights	79.1	+0.3
+ Label Smoothing	80.4	+1.3
+ Stochastic Depth	80.6	+0.2
+ RandAugment	81.0	+0.4
+ Dropout on FC	80.7 <sup>‡</sup>	-0.3
+ Decrease weight decay	82.2	+1.5
+ Squeeze-and-Excitation	82.9	+0.7
+ ResNet-D	83.4	+0.5

Many of these tricks are used to train transformers, so previous comparisons were unfair!!

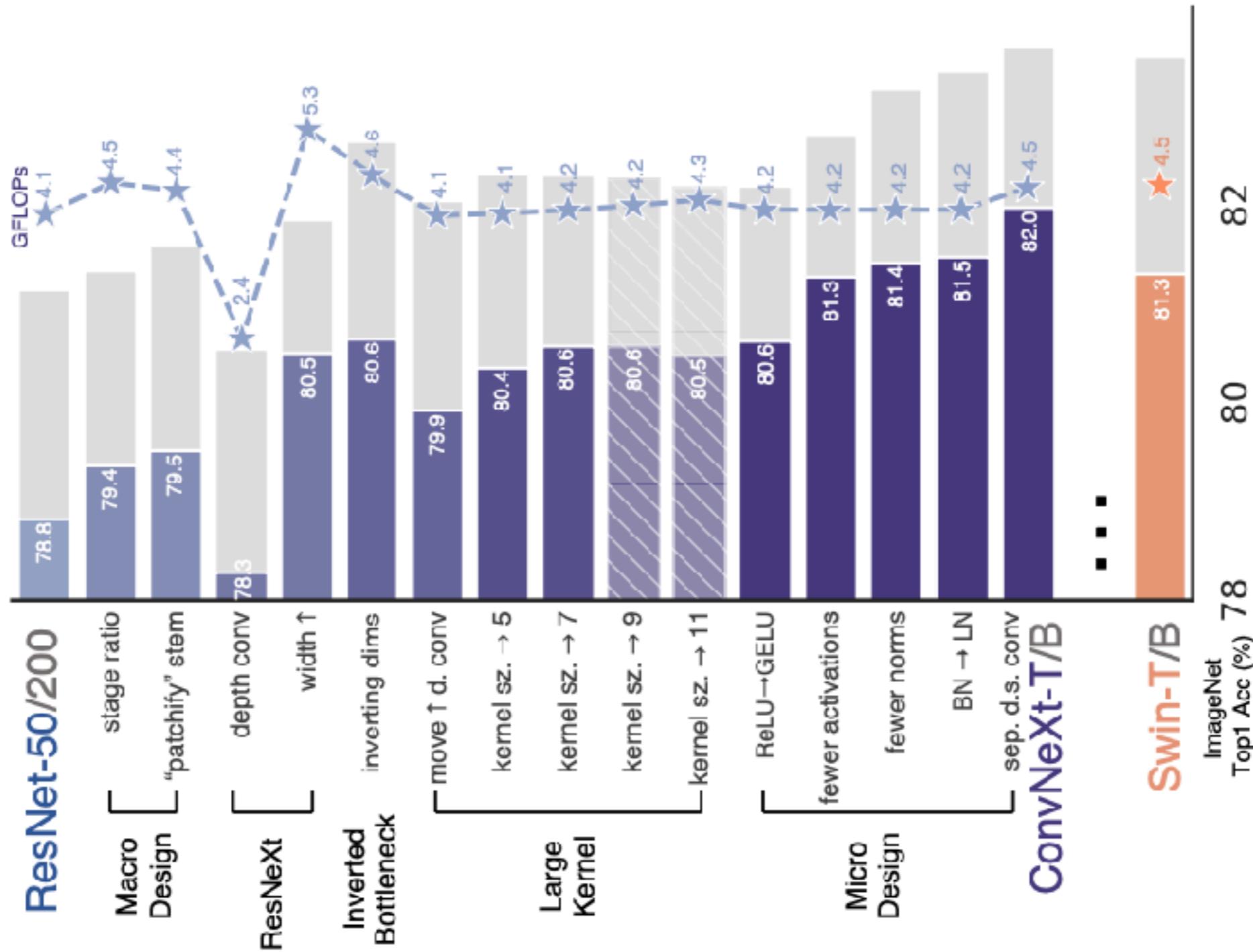
Table 1. Additive study of the ResNet-RS training recipe. The colors refer to Training Methods, Regularization Methods

and Architecture Improvements. The baseline ResNet-200 was trained for the standard 90 epochs using a stepwise learning rate decay schedule. The image resolution is  $256 \times 256$ . All numbers are reported on the ImageNet validation-set and averaged over 2 runs. <sup>†</sup> Increasing training duration to 350 epochs only becomes useful once the regularization methods are used, otherwise the accuracy drops due to over-fitting. <sup>‡</sup> dropout hurts as we have not yet decreased the weight decay (See Table 2 for more details).

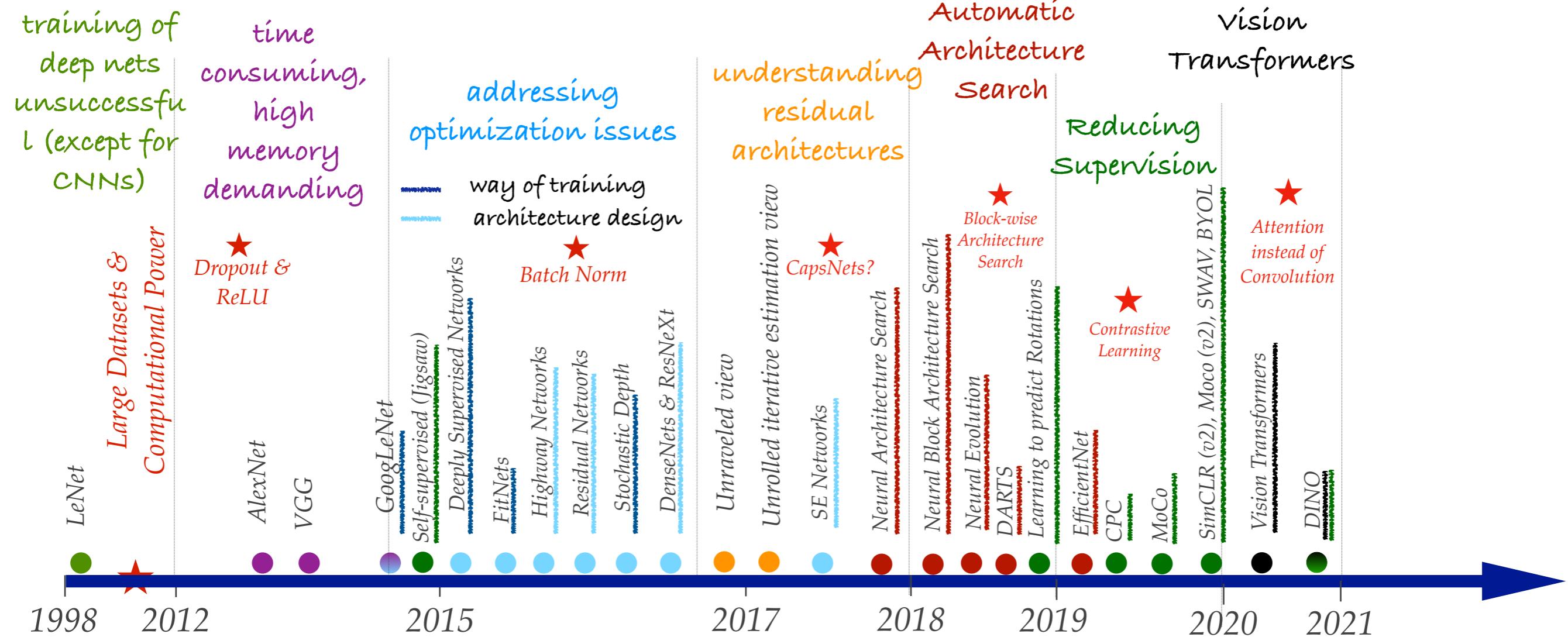
# ResNet strikes back: An improved training procedure in timm

Architecture	# params ×10 <sup>6</sup>	FLOPs ×10 <sup>9</sup>	Throughput (im/s)	Peak mem (MB)	Top-1 Acc.	Real Acc.	V2 Acc.
ResNet-18 [13]	11.7	1.8	7960.5	588	71.5	79.4	59.4
ResNet-34 [13]	21.8	3.7	4862.6	642	76.4	83.4	65.1
ResNet-50 [13]	25.6	4.1	2536.6	1,155	80.4	85.7	68.7
ResNet-101 [13]	44.5	7.9	1547.9	1,264	81.5	86.3	70.3
ResNet-152 [13]	60.2	11.6	1094.0	1,355	82.0	86.4	70.6
RegNetY-4GF [32]	20.6	4.0	1690.6	1,585	81.5	86.7	70.7
RegNetY-8GF [32]	39.2	8.1	1122.3	2,139	82.2	86.7	71.1
RegNetY-16GF [32]	83.6	16.0	694.1	3,052	82.0	86.4	71.2
RegNetY-32GF [32]	145.0	32.4	431.5	3,366	82.5	86.6	71.7
SE-ResNet-50 [20]	28.1	4.1	2174.8	1,193	80.0	85.8	68.8
SENet-154 [20]	115.1	20.9	511.5	2,414	81.7	86.0	71.2
ResNet-50-D [14]	25.6	4.4	2418.8	1,205	80.7	85.9	68.9
ResNeXt-50-32x4d [51]	25.0	4.3	1727.5	1,247	80.5	85.5	68.4
EfficientNet-B0 [41]	5.3	0.4	3701.5	932	77.0	83.8	65.0
EfficientNet-B1 [41]	7.8	0.7	2365.2	1,077	79.2	85.3	67.7
EfficientNet-B2 [41]	9.2	1.0	1786.8	1,318	80.4	86.0	69.3
EfficientNet-B3 [41]	12.0	1.8	1082.4	2,447	81.4	86.7	70.4
EfficientNet-B4 [41]	19.0	4.2	561.3	5,058	81.6	85.9	70.8
ViT-Ti [45]	5.7	1.3	3497.7	346	74.7	82.1	62.4
ViT-S [45]	22.0	4.6	1762.3	682	80.6	85.6	69.4
ViT-B [11]	86.6	17.6	771.0	1,544	80.4	84.8	69.4
timm [50] specific architectures							
ECA-ResNet50-T	25.6	4.4	2139.7	1,155	81.3	86.1	69.9
EfficientNetV2-rw-S [42]	23.9	8.8	823.1	2,339	80.6	84.8	69.2
EfficientNetV2-rw-M [42]	53.2	18.5	456.8	2,916	82.3	87.1	71.7
ECA-Resnet269-D	102.1	70.6	168.1	4,134	83.3	86.9	71.9

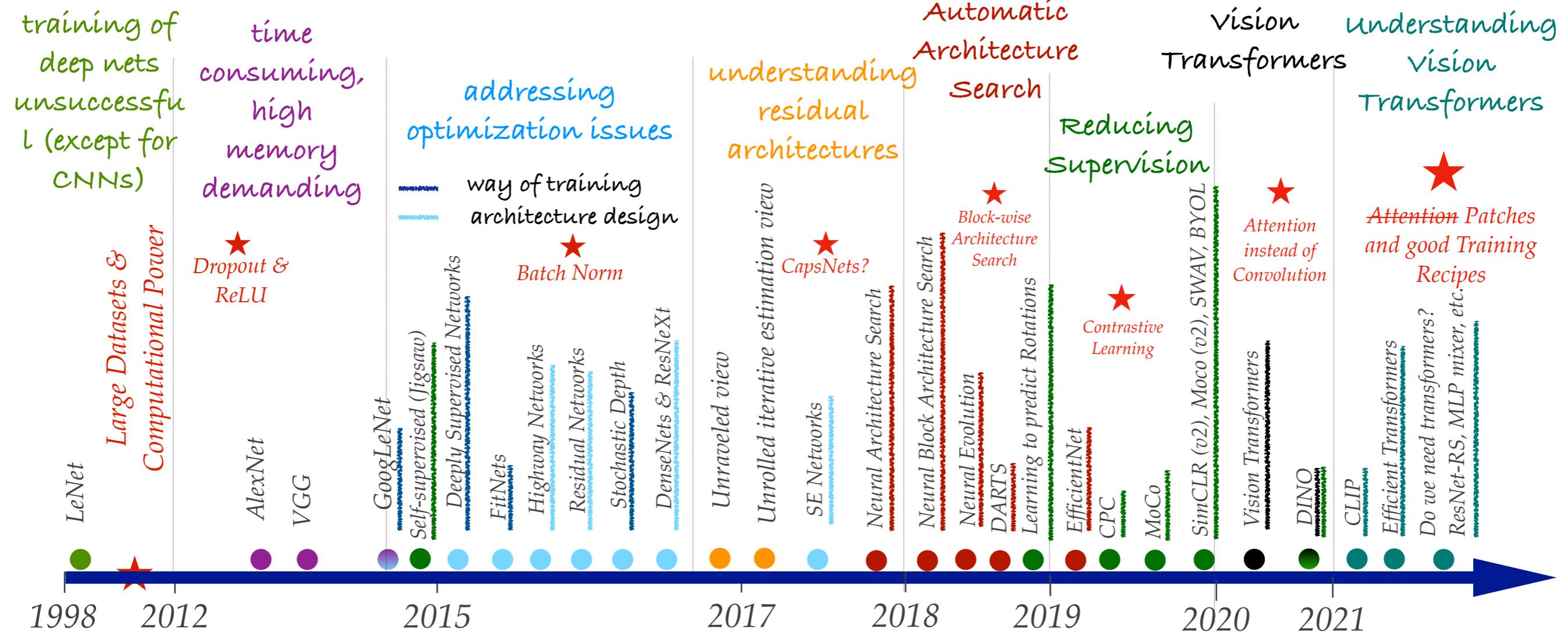
# ConvNeXt: A ConvNet for the 2020s



# Wrap Up



# Wrap Up



# References

- [LeCun et al. 1998]** LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86(11), 2278–2324, 1998.
- [Krizhevsky et al. 2012]** Krizhevsky A., Sutskever I., Hinton G., **ImageNet Classification with Deep Convolutional Neural Networks**, NeurIPS, 2012.
- [Simonyan et al. 2015]** Simonyan K., Zisserman A., **Very Deep Convolutional Networks for Large-Scale Image Recognition**, ICLR, 2015.
- [Szegedy et al. 2015]** Szegedy C., Liu W., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A., **Going Deeper with Convolutions**, CVPR, 2015.
- [Lee et al. 2014]** Lee C.Y., Xie S., Gallagher P.W., Zhang Z., Tu Z., **Deeply-Supervised Nets**, AISTATS, 2015.
- [Romero et al. 2015]** Romero A., Ballas N., Ebrahimi Kahou S., Chassang A., Gatta C, Bengio Y., **FitNets: Hints for Thin Deep Nets**, ICLR, 2015.
- [Srivastava et al. 2015]** Srivastava R. K., Greff K., Schmidhuber J., **Training Very Deep Networks**, NeurIPS 2015.
- [Doersch et al. 2015]** Doersch, Carl, Abhinav Gupta, and Alexei A. Efros. "Unsupervised visual representation learning by context prediction." ICCV. 2015.
- [He et al. 2015]** He K., Zhang X., Ren S., Sun J., **Deep Residual Learning for Image Recognition**, CVPR, 2016.
- [Huang et al. 2016]** Huang G., Sun Y., Liu Z., Sedra D., Weinberger K. Q., **Deep Networks with Stochastic Depth**, ECCV, 2016.
- [Veit et al. 2016]** Veit, A., Wilber, M., Belongie, S., **Residual Networks Behave Like Ensembles of Relatively Shallow Networks**, NeurIPS, 2016.
- [Zoph et al. 2017]** (2017). Neural architecture search with reinforcement learning. arXiv preprint arXiv:
- [Huang et al. 2016]** Huang G., Liu Z., Weinberger K. Q., van der Maaten L., **Densely Connected Convolutional Networks**, CVPR, 2017.
- [Greff et al. 2017]** Greff K., Srivastava R.K., Schmidhuber J., **Highway and Residual Networks Learn Unrolled Iterative Estimation**, ICLR, 2017.
- [Xie et al. 2017]** Xie S., Girshick R., Dollar P., Tu Z., He K., **Aggregated Residual Transformations for Deep Neural Networks**, CVPR, 2017.
- [Hu et al. 2017]** Hu J., Shen L., Sun G., **Squeeze-and-Excitation Networks**, arXiv, 2017.
- [Sabour et al. 2017]** Sabour, S., Frosst, N., & Hinton, G. E. (2017). **Dynamic routing between capsules**. NeurIPS
- [Vaswani et al. 2017]** Vaswani, Ashish, et al. "**Attention is All you Need.**" NeurIPS (2017): 5998-6008.
- [Zoph et al. 2018]** Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). **Learning transferable architectures for scalable image recognition**. CVPR
- [Real et al. 2018]** Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2018). **Regularized evolution for image classifier architecture search**. arXiv preprint arXiv:1802.01548.
- [Liu et al. 2018]** Liu, Hanxiao, Karen Simonyan, and Yiming Yang. "**Darts: Differentiable architecture search.**" arXiv preprint arXiv:1806.09055 (2018).

- [Tan et al. 2019]** Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International Conference on Machine Learning. PMLR, 2019.
- [Gidaris et al. 2018]** Gidaris, Spyros, Praveer Singh, and Nikos Komodakis. "Unsupervised representation learning by predicting image rotations." ICLR 2018.
- [Lowe et al. 2019]** Löwe, S., O'Connor, P., & Veeling, B. (2019). Putting An End to End-to-End: Gradient-Isolated Learning of Representations. In Advances in Neural Information Processing Systems (pp. 3033-3045).
- [Henaff et al. 2019]** Hénaff, O. J., Razavi, A., Doersch, C., Eslami, S. M., & Oord, A. V. D. (2019). Data-efficient image recognition with contrastive predictive coding. NeurIPS.
- [He et al. 2020]** He, Kaiming, et al. "Momentum contrast for unsupervised visual representation learning." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.
- [Chen et al. 2020]** Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." International conference on machine learning. PMLR, 2020.
- [Dosovitskiy et al. 2020]** Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).
- [Radford et al. 2021]** Radford, Alec, et al. "Learning transferable visual models from natural language supervision." arXiv preprint arXiv:2103.00020 (2021).
- [Raghu, Maithra, et al.]** "Do Vision Transformers See Like Convolutional Neural Networks?." Advances in Neural Information Processing Systems 34 (2021).
- [Caron, Mathilde, et al.]** "Emerging properties in self-supervised vision transformers." arXiv preprint arXiv:2104.14294 (2021).
- [Wu, Haiping, et al.]** "Cvt: Introducing convolutions to vision transformers." arXiv preprint arXiv:2103.15808 (2021).
- [Dai, Zihang, et al.]** "CoAtNet: Marrying Convolution and Attention for All Data Sizes." arXiv preprint arXiv:2106.04803 (2021).
- [Bello, Irwan, et al.]** "Revisiting resnets: Improved training and scaling strategies." arXiv preprint arXiv:2103.07579 (2021).
- [Wightman, Ross, Hugo Touvron, and Hervé Jégou.]** "Resnet strikes back: An improved training procedure in timm." arXiv preprint arXiv:2110.00476 (2021).

# **Architectures for Image Classification**

Master in Computer Vision Barcelona

Pau Rodriguez, January 2019

[pau.rodriguez@elementai.com](mailto:pau.rodriguez@elementai.com)

Adriana Romero, February 2018