Deep Learning in Computer Vision
Lecture 1.3

Morten Rieger Hannemose, mohan@dtu.dk

Topics of this lecture

- Better optimization
  - “Tricks” for better stability and faster training
  - Other optimizers
- Common architectures for classification
- Transfer Learning
- Heatmaps
- Adversarial examples

Last lecture

- In a convolutional layer features and channels are synonyms

- Stride>1 is used for reducing the size of the image
  - Often you will either either use:
    - convolutions with stride=2 or,
    - MaxPooling (with stride=2)

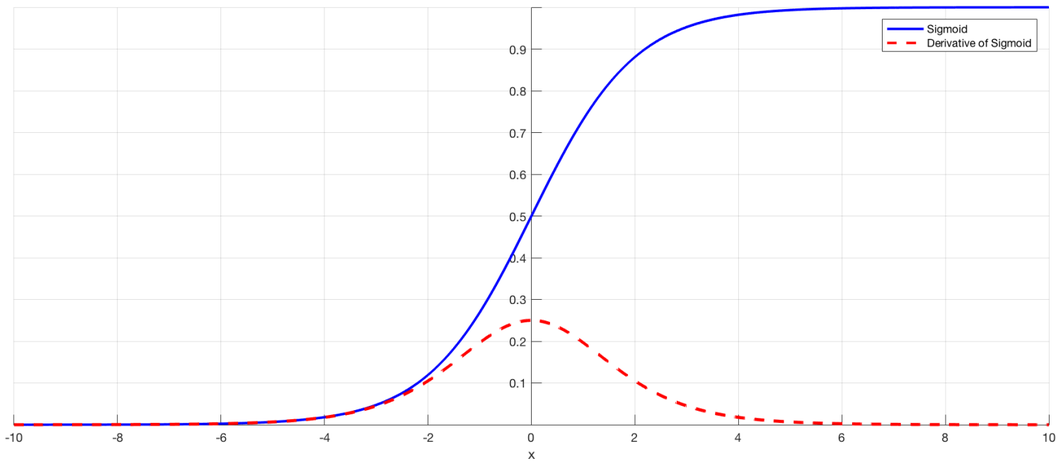
Optimization

Better optimization

- A lot has been done to do faster training of deep neural networks
- The gradients in a deep neural networks are unstable
  - Many gradients are multiplied together
    - Vanishing gradient / exploding gradient
- Better optimization methods

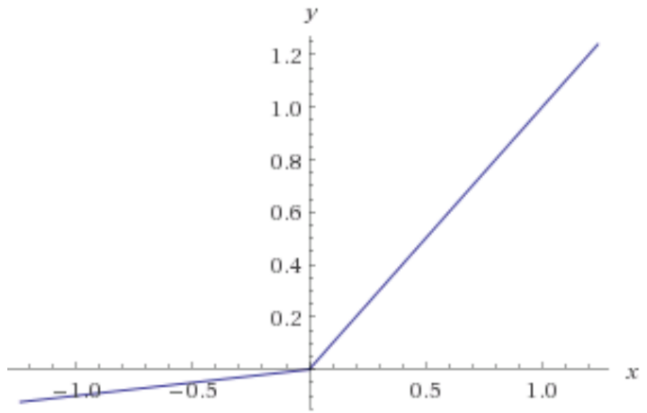
Vanishing gradients

- The gradients in the early layers of a network is the product of the gradients from later layers
  - If we use a sigmoid activation function we may see the ‘vanishing gradient problem’



ReLU

- Solution to vanishing gradients?
  - Use ReLU activation function
- Drawbacks
  - No gradient below 0
  - Not differentiable in 0
  - Can “die” (weights get updated such that a neuron will never activate)
  - Usually not a problem
- Sometimes people use a ‘Leaky ReLU’ to overcome this



Unstable gradients

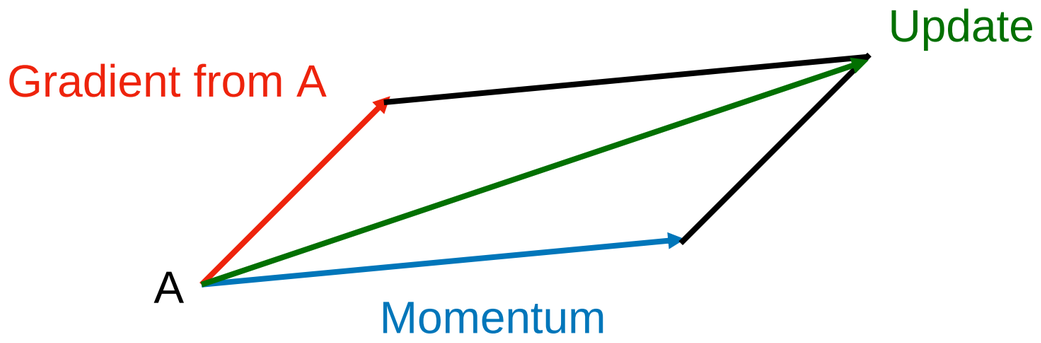
- In general, deep neural networks have unstable gradients and layers may not learn at the same rate.
  - Some weights will have large gradients

Using other optimisers

- Simply using SGD is sufficient for many tasks
- Sometimes, however, training takes a long time which may be reduced if we can take steps better adapted to the loss landscape
- Many SGD alterations have been invented to do just this

SGD with momentum

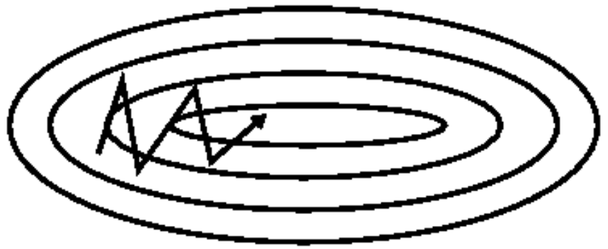
- Consider a ball rolling down a hillside
  - Just because the ball meets a small obstacle does not mean that it completely changes direction
  - Instead it keeps some of the velocity in its current direction and add some velocity in another direction
  - We call this momentum
  - We can use this idea to take better steps than SGD



SGD with momentum

- If a surface curves more steeply in one dimension than in another it takes many steps for SGD to move to the minimum
- SGD would oscillate across the slopes with high curvature with only minor progress towards the minima

SGD



SGD with momentum

SGD with momentum

- Standard SGD

$$w_{j,k}^{\ell} \rightarrow w_{j,k}^{\bar{\ell}} = w_{j,k}^{\ell} - \alpha \frac{\partial L}{\partial w_{j,k}^{\ell}} \qquad \mathbf{W} \rightarrow \bar{W} = \mathbf{W} - \alpha \nabla_W \mathcal{L}$$

- SGD + momentum

$$\mathbf{W} \rightarrow \bar{W} = \mathbf{W} - \mathbf{v}_t$$

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \alpha \nabla_W \mathcal{L}$$

- Typically $\gamma = 0.9$

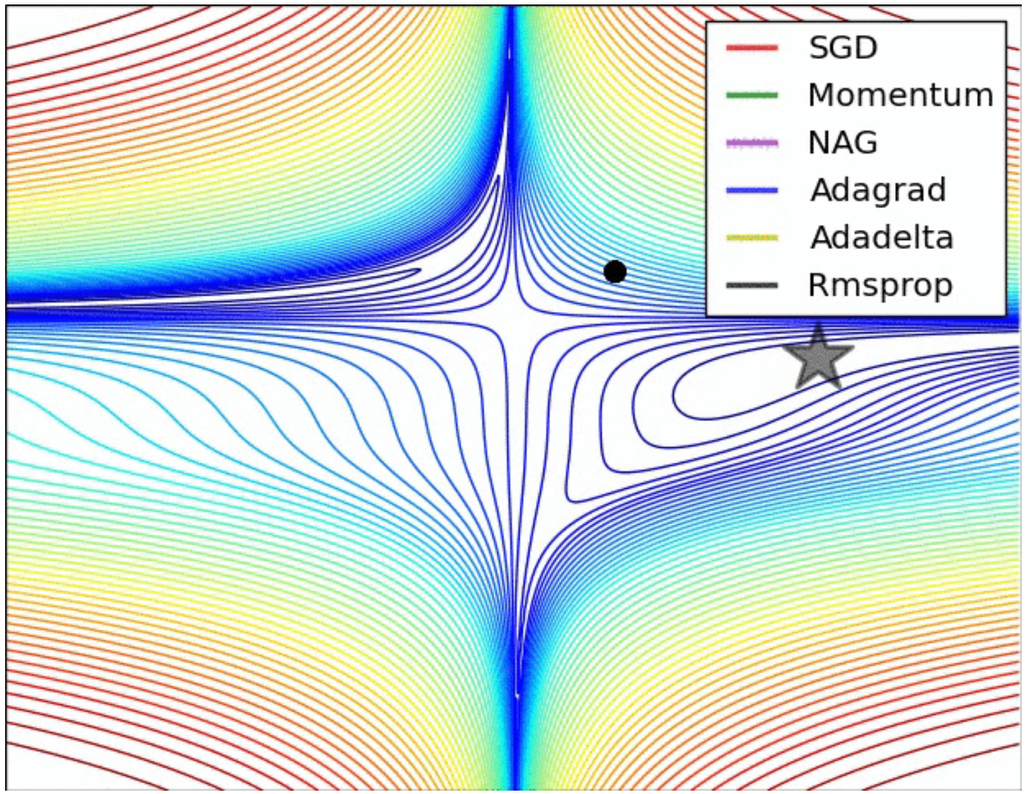
RMSProp

- Root Mean Square Propagation (RMSProp)
- The idea is to adapt the learning rate α for each individual parameter such we take similar sized steps for all parameters
- v_t is a moving average of the squared magnitude of the gradient

$$v_t = \gamma v_{t-1} + (1-\gamma)\left(\frac{\partial \mathcal{L}}{\partial w_{j,k}^{\ell}}\right)^2$$

$$w_{j,k}^{\ell} \rightarrow w_{j,k}^{\bar{\ell}} = w_{j,k}^{\ell} - \frac{\alpha}{\sqrt{v_t}} \frac{\partial \mathcal{L}}{\partial w_{j,k}^{\ell}}$$

Animation



Adam

- Adaptive Moment Estimation (Adam) is a combination of momentum and RMSProp

$$m_w^t = \gamma_1 m_w^{t-1} + (1-\gamma_1)\frac{\partial \mathcal{L}}{\partial w_{j,k}^\ell}$$

$$v_w^t = \gamma_2 v_w^{t-1} + (1-\gamma_2)\left(\frac{\partial \mathcal{L}}{\partial w_{j,k}^\ell}\right)^2$$

- Bias corrections

$$\hat{m_w^t} = \frac{m_w^t}{1-\gamma_1^{t+1}}, \quad \hat{v_w^t} = \frac{v_w^t}{1-\gamma_2^{t+1}}$$

- Update

$$w_{j,k}^\ell \rightarrow w_{j,k}^{\bar{\ell}} = w_{j,k}^\ell - \alpha \frac{m_w^t}{\sqrt{v_w^t}}$$

Learning rate annealing

- Often it is helpful to change to learning rate over time when training deep neural networks.
  - If we are far from a minimum, we want a higher learning rate
  - If we are close to a minimum, we want a lower learning rate.
- If the learning rate is too large the updated parameters will just bounce around and not reach a minimum
- If it is too small, it takes ‘forever’ to reach a minimum

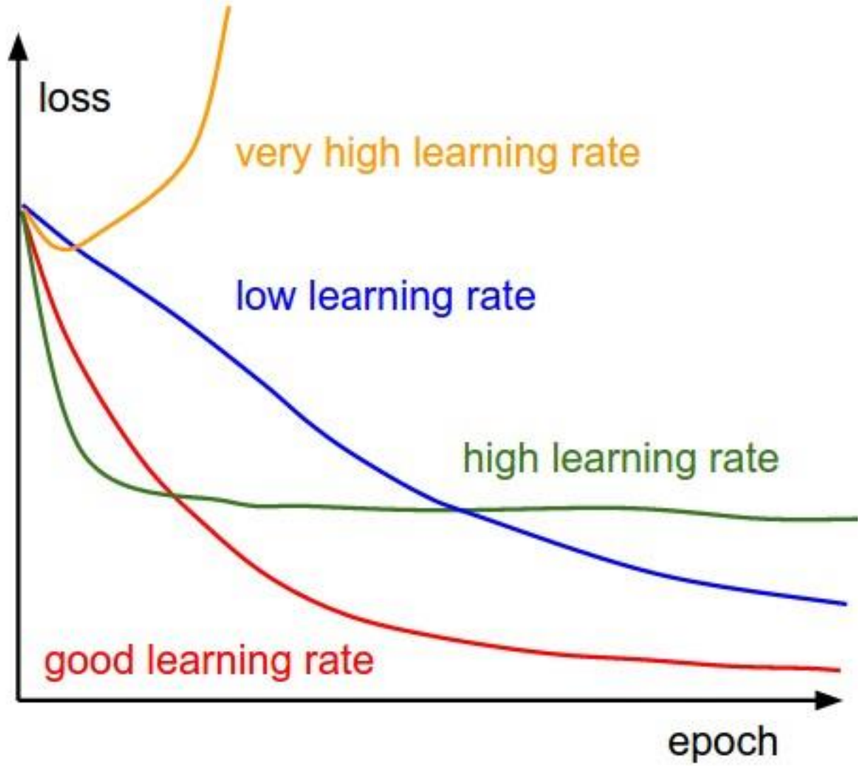
Learning rate annealing

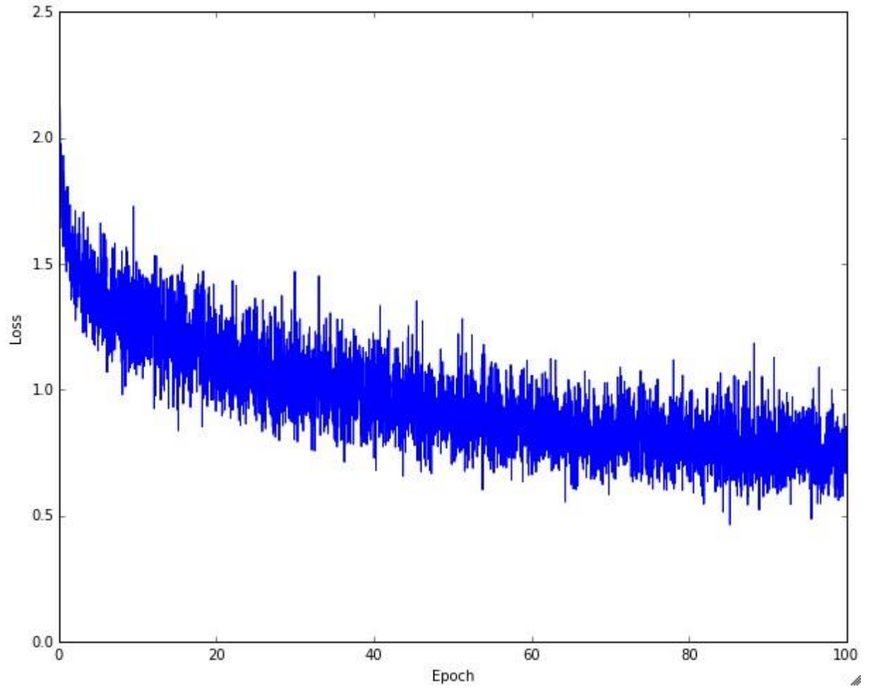
- Two common types of learning rate decay are
- Step decay
  - Reduce the learning rate by some factor $k < 1$ every few epochs. The factor and how many epochs is very dependent on the problem

- Exponential decay
  - Where a_0, k are hyperparameters and t is the time step (epoch number)

$$\alpha = \alpha_0 e^{-kt}$$

Learning rate annealing





Optimizers in practice

- Adam works well enough in most cases
  - Learning rate decay helps as well
- For generalization SGD+momentum can be better, but is slower to train
- When experimenting with learning rates, try changing it by a factor of 10

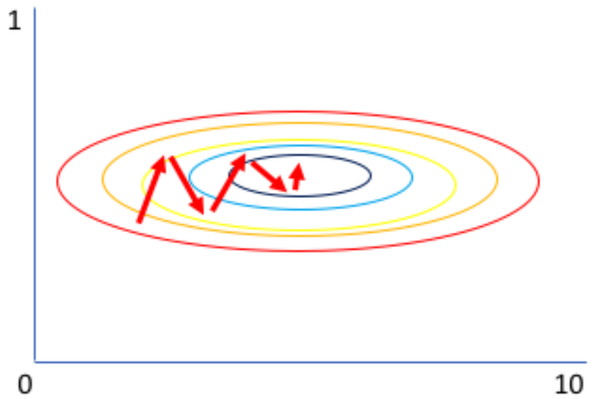
- Variants of Adam
  - RAdam
- AdaBelief
- LARS (for huge batchsizes)

Input normalization

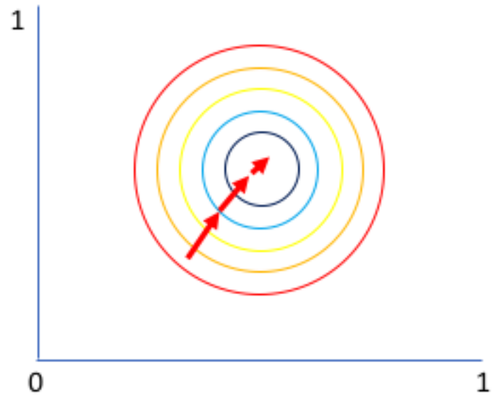
- Consider two images:
  - One of a person in very dark room
  - One of a person in bright sunlight
- We want to train a network on both types of images
- The difference between them is mainly their means (and std. deviations)

Input normalization

- We (almost) always normalize our input to some specified range ([0,1] or [-1,1]) or to zero mean and std. dev. 1
- Intuitively this makes it easier for the network to learn since the output will also be in the same(ish) range



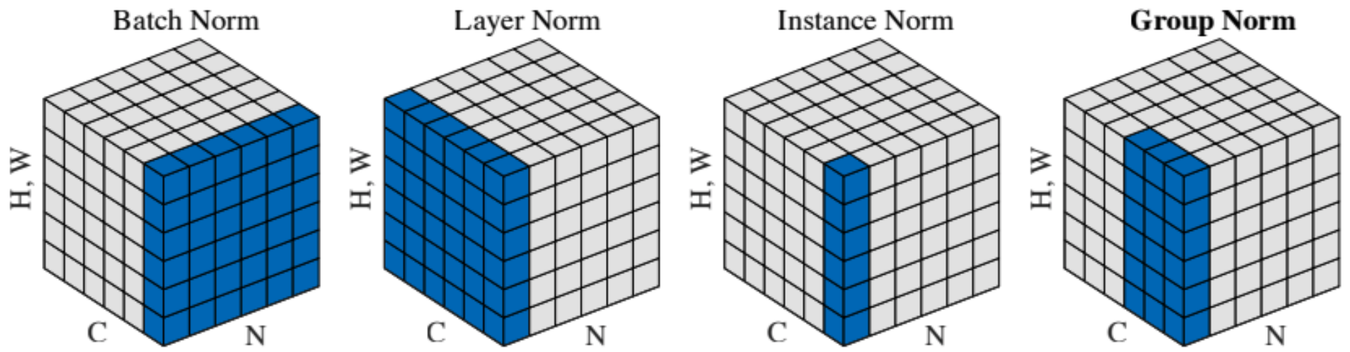
Gradient of larger parameter dominates the update



Both parameters can be updated in equal proportions

Batch normalisation

- We normalize the input to the network for faster convergence and more stability
  - Why not do the same for the outputs from the hidden layers?
- This is called batch normalisation
- Batch normalisation normalises the output of each layer by subtracting the mean and dividing by the standard deviation of the of mini-batch



Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Ioffe & Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Batch normalization

- Batch normalization makes the mean and standard deviation of the features into learnable parameters.
- Problematic for small batch sizes (<16)
  - Variance of mean and std. dev. gets too high
  - Other normalizations work better in this case

Batch normalization at test time

- At test time we might only process images one at a time
  - We cannot do mini-batch statistics

- We keep a running average of the minibatch statistics across the minibatches to use at test time

$$\mu_t = \gamma \mu_{t-1} + (1 - \gamma)\mu_B$$

Important architectures

(but first, a break)

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- The ILSVRC is a large-scale challenge for object detection and image classification
- ImageNet has more than 14 million images in more than 20.000 categories
- The annual challenge has spurred some of the most widely used architectures for visual recognition today

ImageNet Large Scale Visual Recognition Challenge

- The classification challenge has 1000 categories
- Includes fine grained recognition – 90 different dog species
- Top-1 / Top-5 error

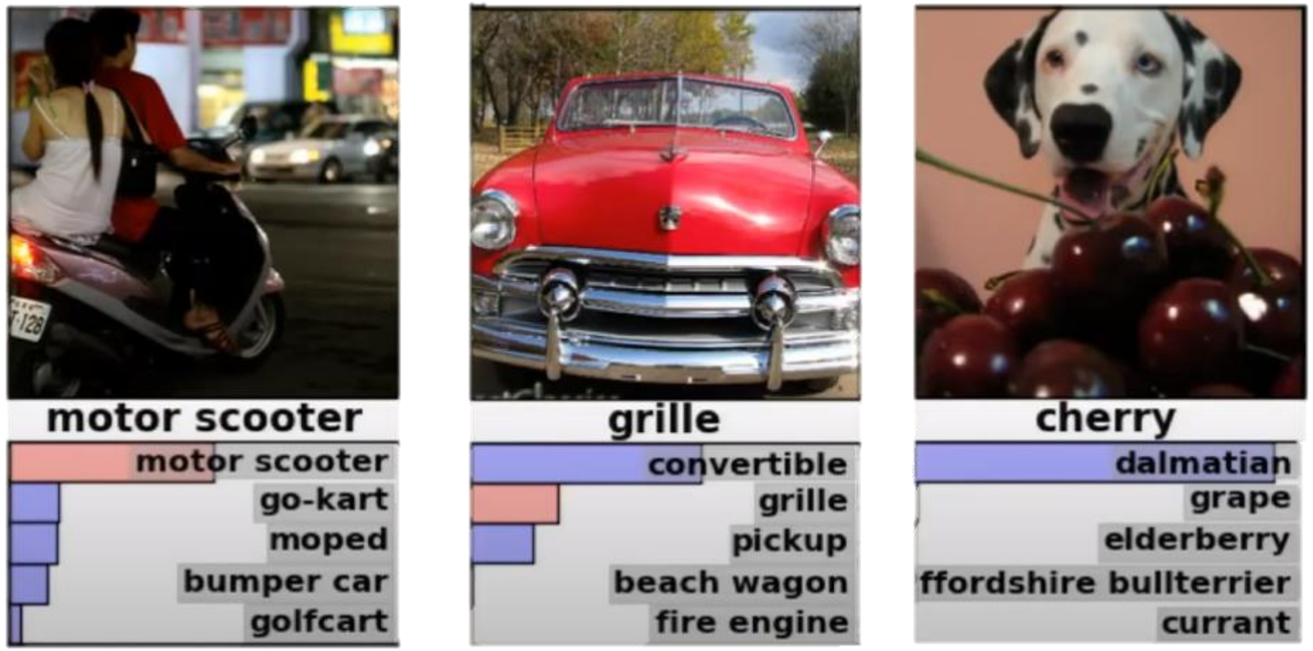
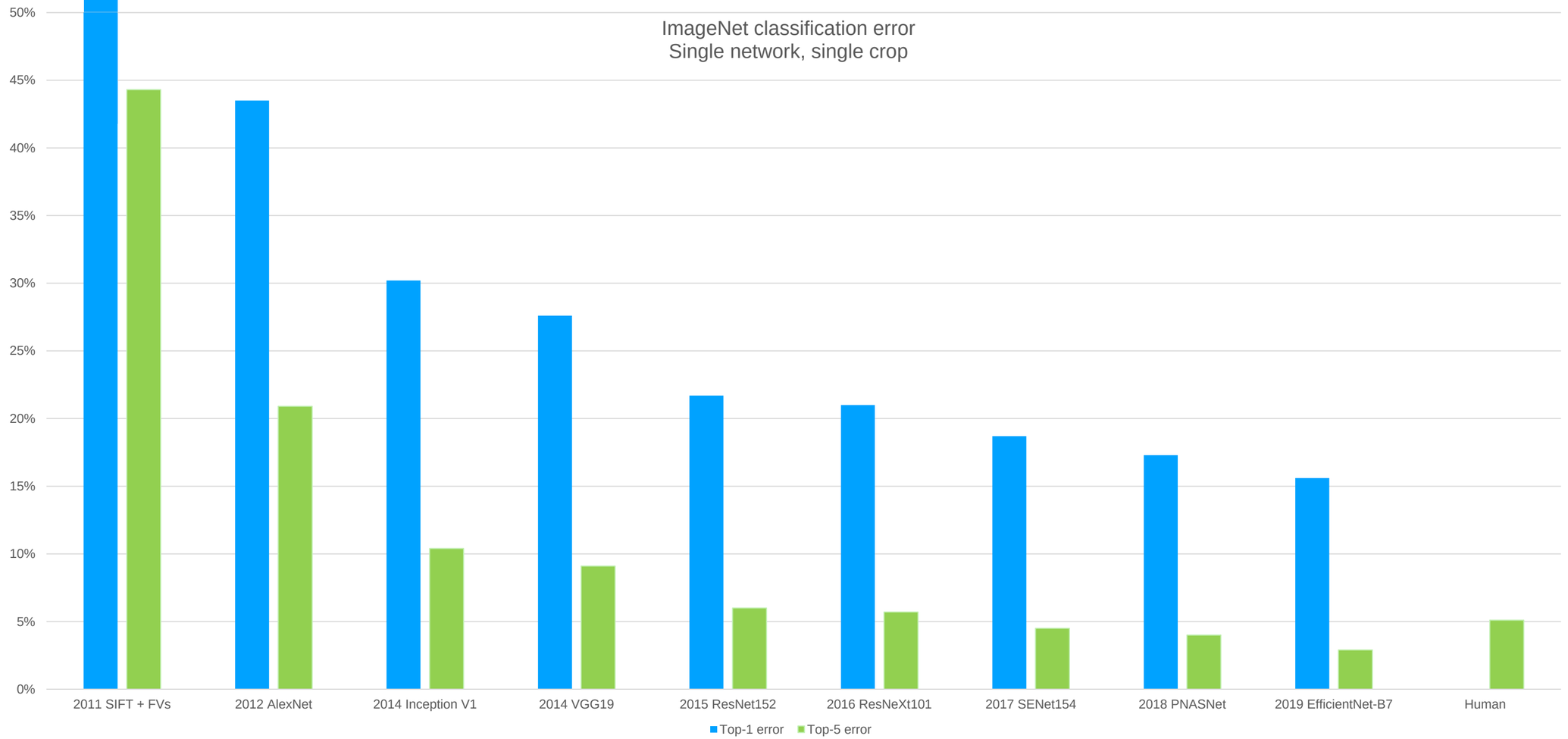


Figure from slides by Geoff Hinton, 2013

Performance of selected models



You will find contradicting numbers online and in the papers. This is my attempt at showing a fair comparision of a single network [no ensembles] being shown one image [no random crops/scale jittering].
Most numbers from [1], except EfficientNet[2] and human performance[3], which is based on only one human (Andrej Kaparthy).

[1] Recht, Benjamin, et al. "Do imagenet classifiers generalize to imagenet?." *arXiv preprint arXiv:1902.10811* (2019).
[2] Tan, Mingxing, and Quoc V. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *arXiv preprint arXiv:1905.11946* (2019).
[3] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." *International journal of computer vision* 115.3 (2015): 211-252.

AlexNet

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

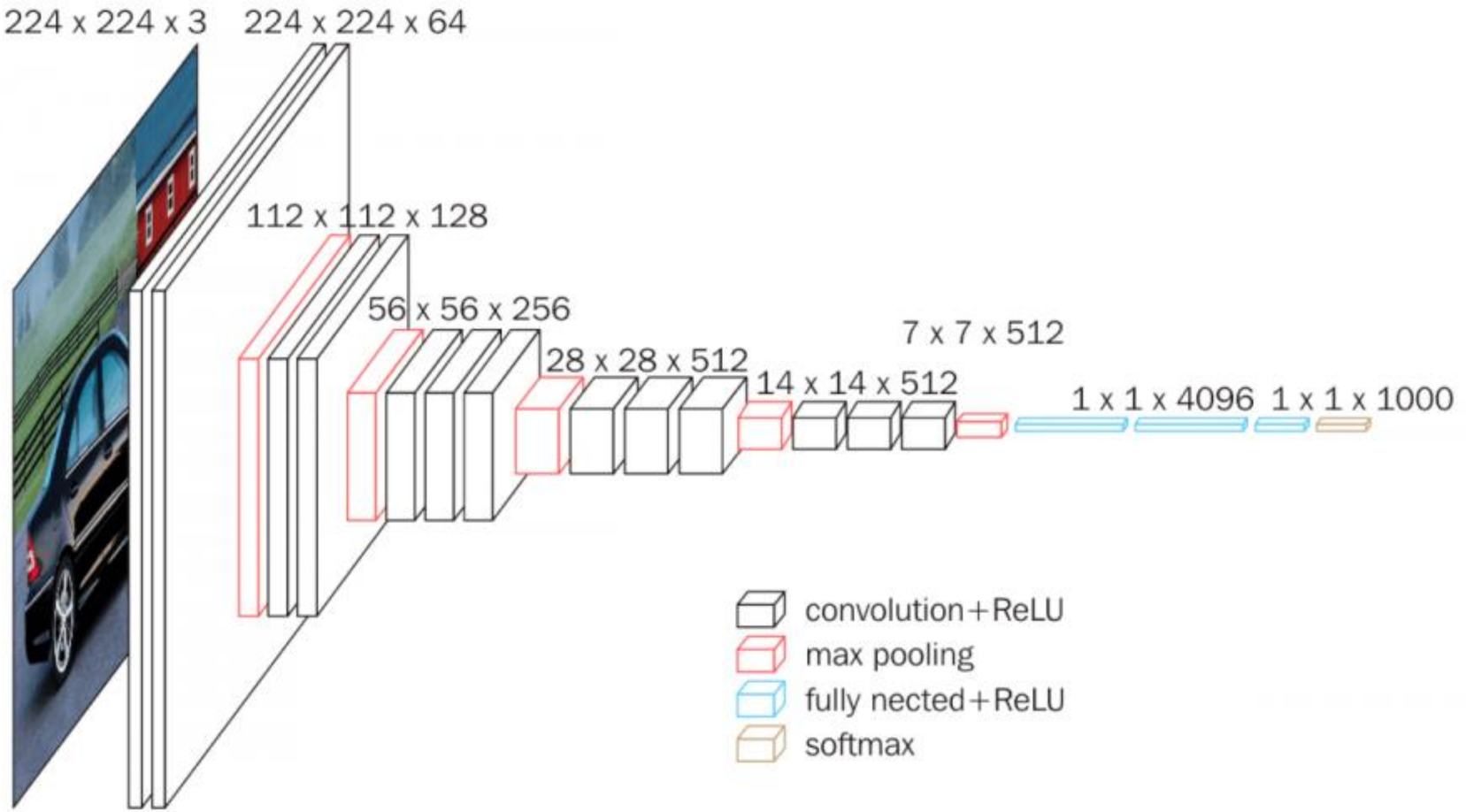
- Tremendous breakthrough in classification performance on ImageNet
- The secrets behind it’s success?
  - ReLU (no vanishing gradients)
  - GPU (more computational power)

VGG

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

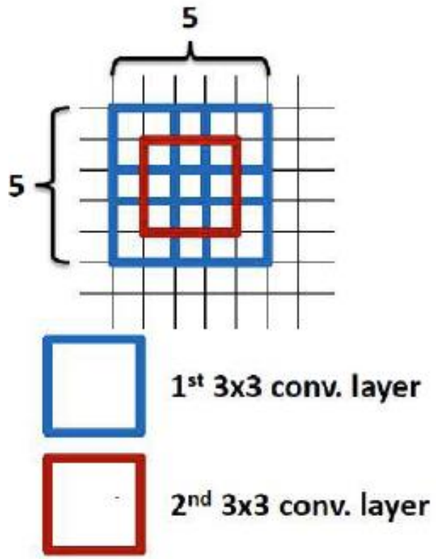
- Pushed the standard convolution neural network with 3x3 convolutions to 19 layers in 2014
- Runner-up of ILSVRC 2014 (Their best model had 19 layers)
- VGG16 has around 138 million parameters
  - Using 4 GPUs it took 2-3 weeks to train

VGG

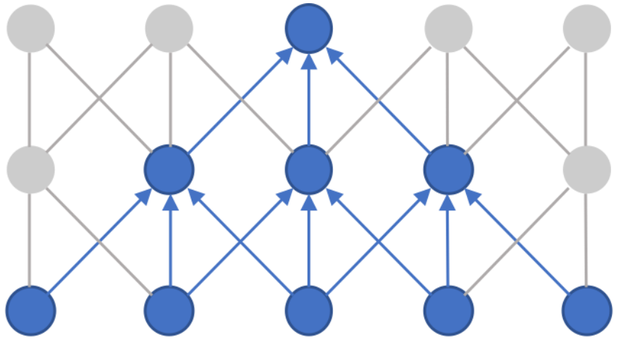

224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

3x3 convolutions

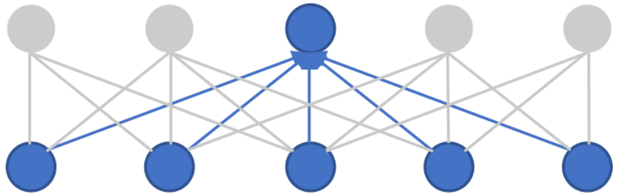
- Why only use 3x3 convolutions?
- Hint: What's the difference between two 3x3 convolutions and one 5x5?
  - Stacking layers increases the receptive field
    - Two 3x3 layers - 5x5 receptive field
    - Three 3x3 layers - 7x7 receptive field
    - …
  - Fewer parameters to learn



3x3 convolutions



Two 3x3 convolutions
Parameters: $2 \cdot (3^2 + 1) = 20$
Operations: $2 \cdot (2 \cdot 3^2) = 36$



One 5x5 convolution
Parameters: $5^2 + 1 = 26$
Operations: $2 \cdot 5^2 = 50$

Three 3x3 convolutions
Parameters: $3 \cdot (3^2 + 1) = 30$
Operations: $3 \cdot (2 \cdot 3^2) = 54$

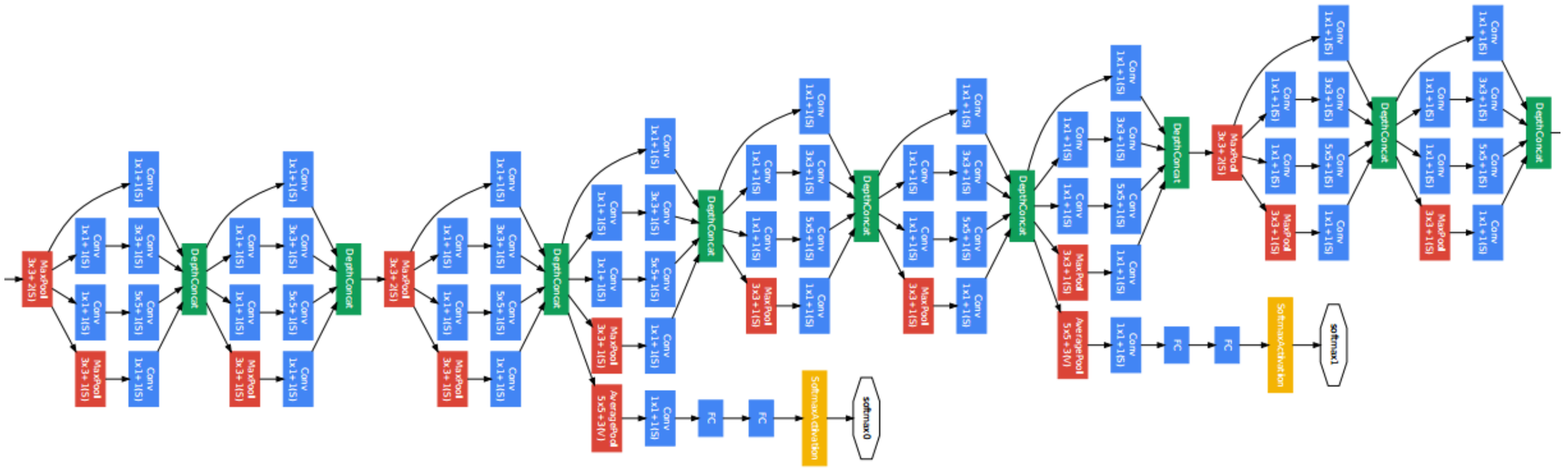
One 7x7 convolution
Parameters: $7^2 + 1 = 50$
Operations: $2 \cdot 5^2 = 98$

GoogLeNet/Inception v1

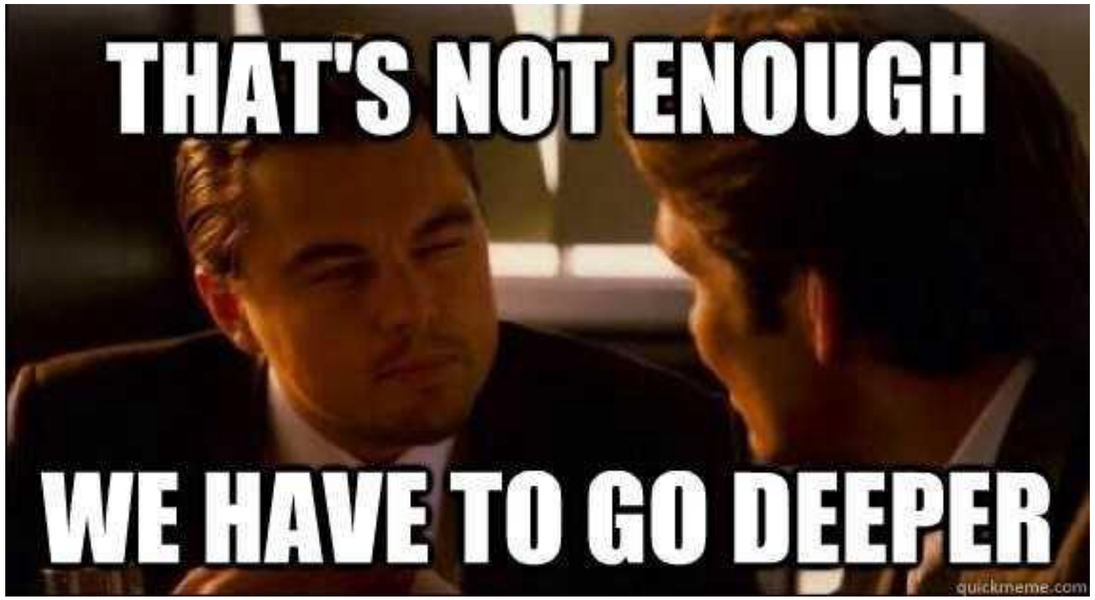
Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

- Winner of the ILSVRC 2014 Classification challenge
  - VGG-19 performed better when only using one network

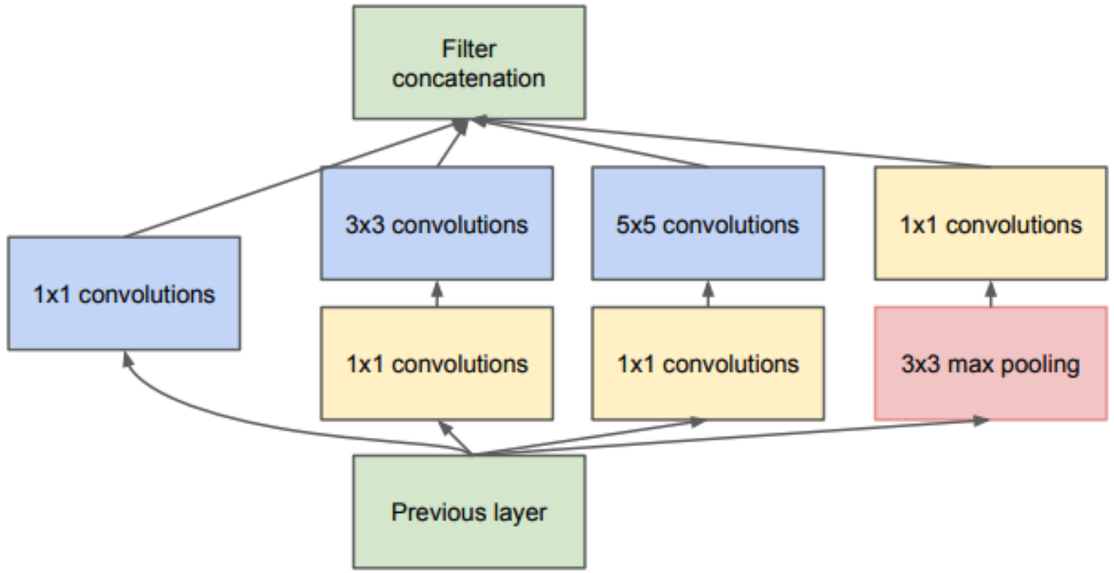
- Introduced the concept of blocks (network in network)



GoogLeNet/Inception v1



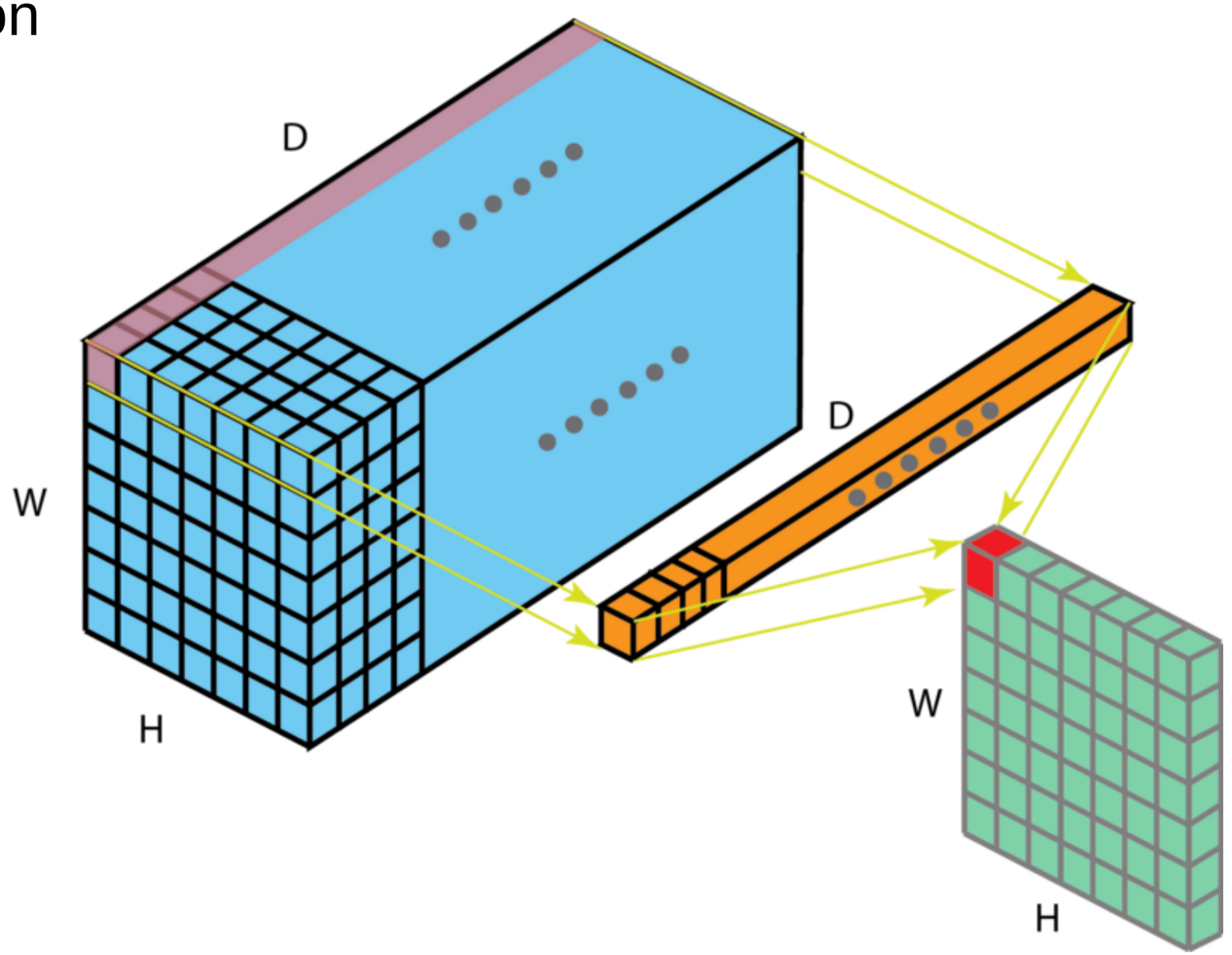
Inception block



(b) Inception module with dimension reductions

1x1 convolution

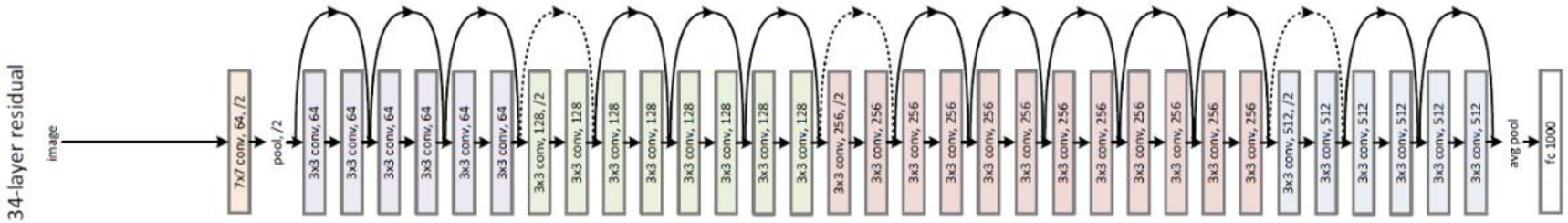
- 1x1 convolutions are fully connected neural networks operating on a single spatial location



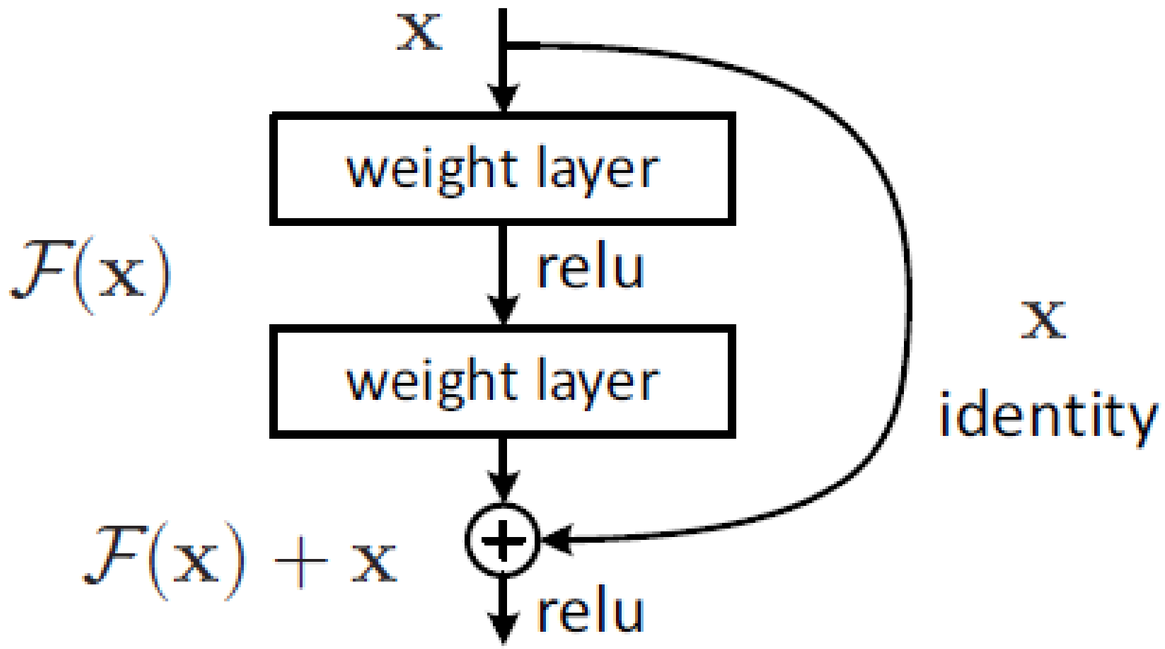
ResNet

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

- Won the ILSVRC 2015 Challenge
  - Popularized skip/residual connections
  - Could train over 1000 layers
  - Models with 18, 34, 50, 101 and 152 layers
  - Best result with 152

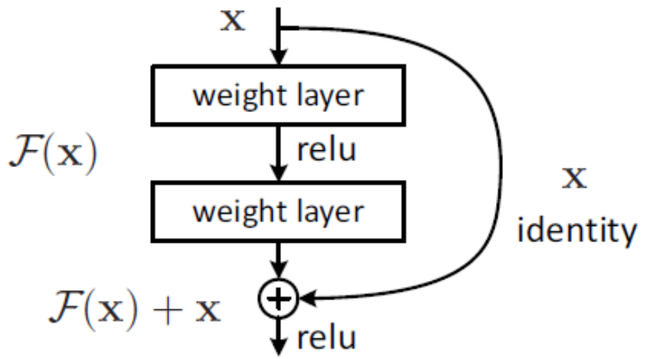


The building block of ResNets



ResNets

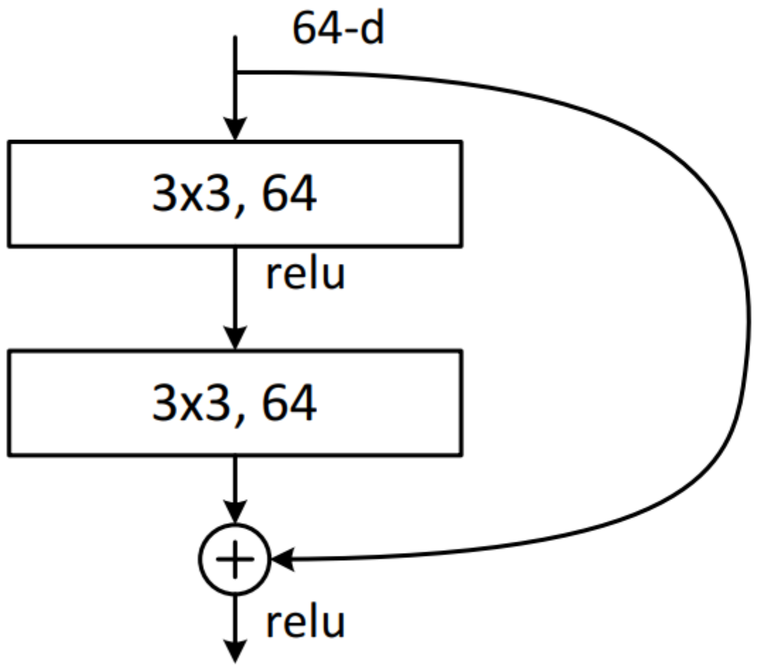
- Why can we train very deep networks using residual blocks?
- If our weights are zero in a residual block, we just feed the values through - so it can function as a shallower network
- In back propagation the gradients can use the identity mappings to move faster back to the early layers



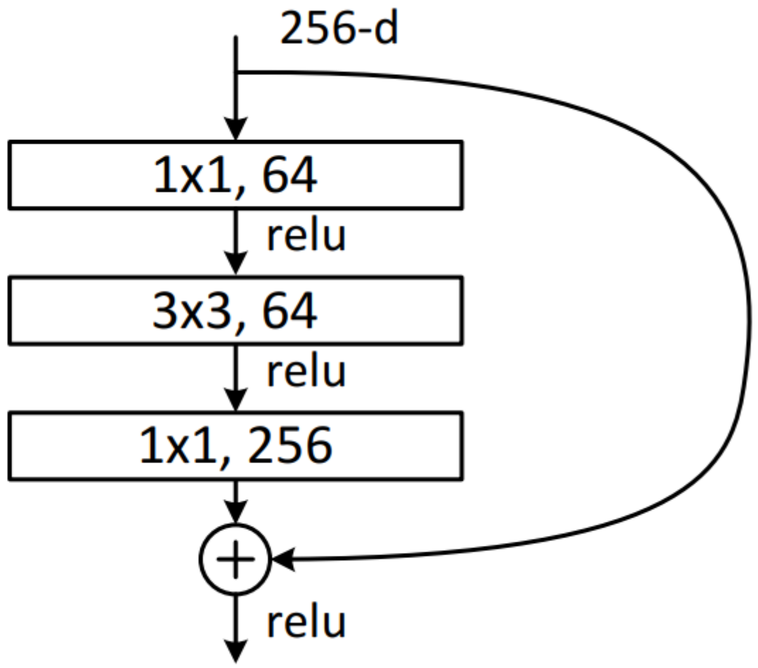
ResNets

- For networks with more than 34 layers, they used a bottleneck block
- A quarter of the features inside the bottleneck

Residual block



Residual block with bottleneck



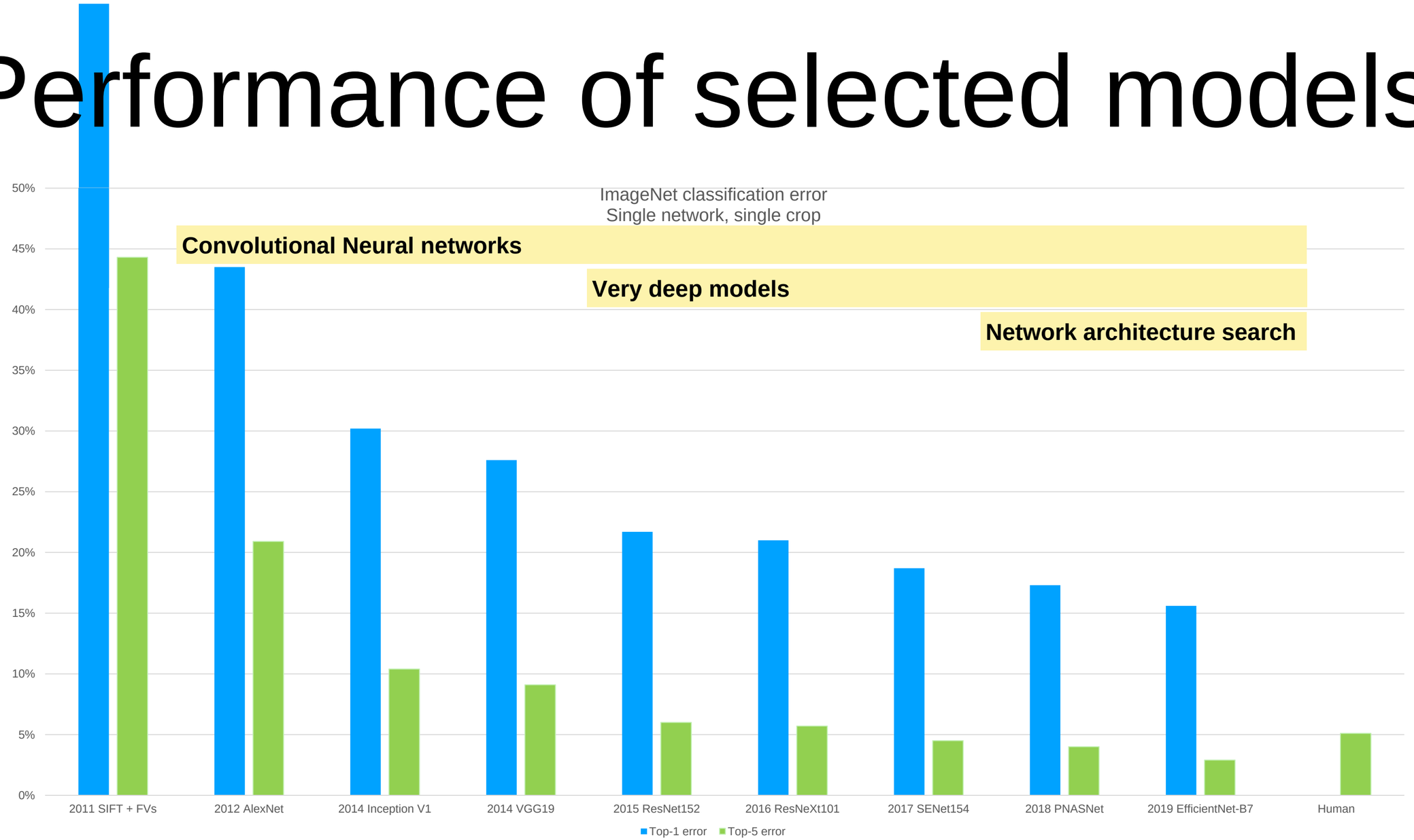
3x3 convolutions

- Sometimes models use larger convolutions on the input image
  - Resnet uses a 7x7 convolution with stride 2 and 64 channels
  - Why do you think this is the case?
  - Discuss with your neighbor

3x3 convolutions

- The key is the number of channels that are created
- The input has only three channels so convolutions are relatively cheap
- One 7x7 convolution on the input
  - Number of parameters: $7^2 * (3 * 64) + 64 = 9\,472$
- Three 3x3 convolutions on the input
  - Number of parameters: $3^2 * (3 * 64 + 64^2 + 64^2) + 3 * 64 = 75\,648$
- Do note that these are not equivalent, but do have the same field of view

Performance of selected models



You will find contradicting numbers online and in the papers. This is my attempt at showing a fair comparision of a single network [no ensembles] being shown one image [no random crops/scale jittering].
Most numbers from [1], except EfficientNet[2] and human performance[3], which is based on only one human (Andrej Kaparthy).

[1] Recht, Benjamin, et al. "Do imagenet classifiers generalize to imagenet?." *arXiv preprint arXiv:1902.10811* (2019).
[2] Tan, Mingxing, and Quoc V. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *arXiv preprint arXiv:1905.11946* (2019).
[3] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." *International journal of computer vision* 115.3 (2015): 211-252.

Efficientnet

Network architecture search for an optimal small model and scale that up

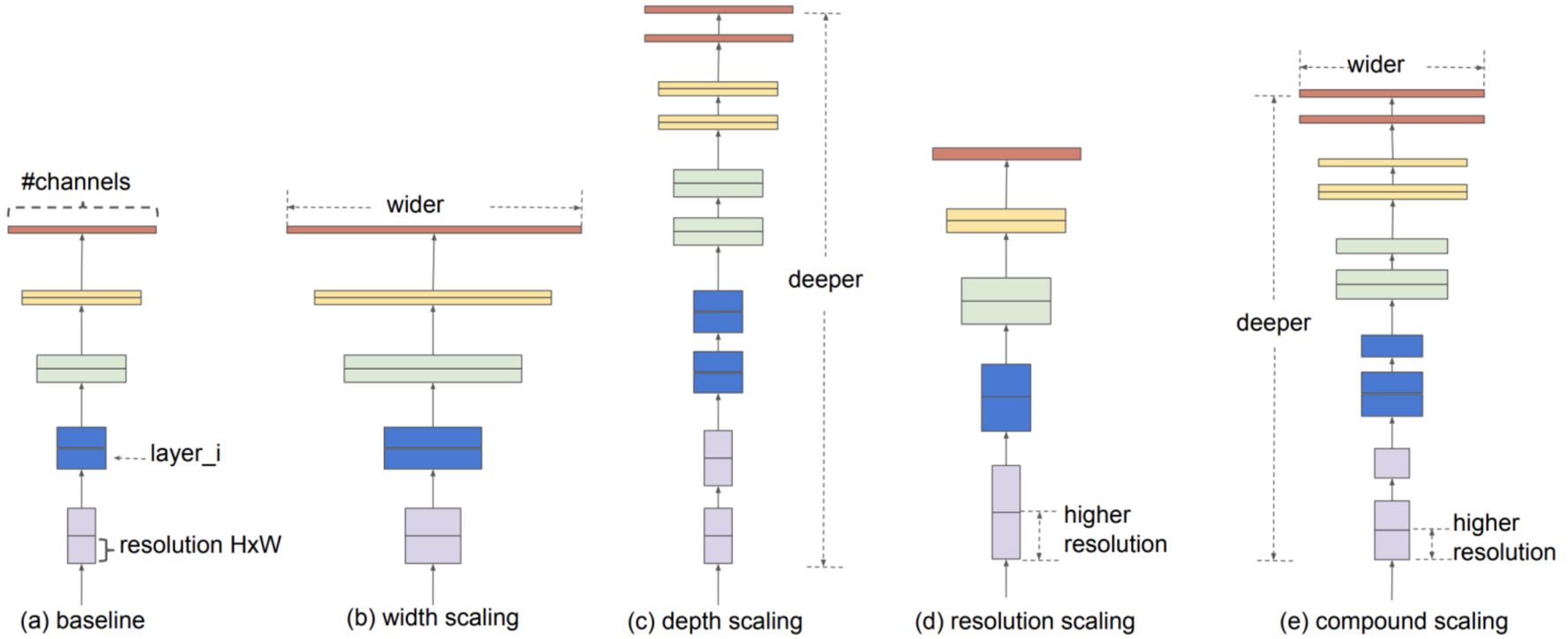
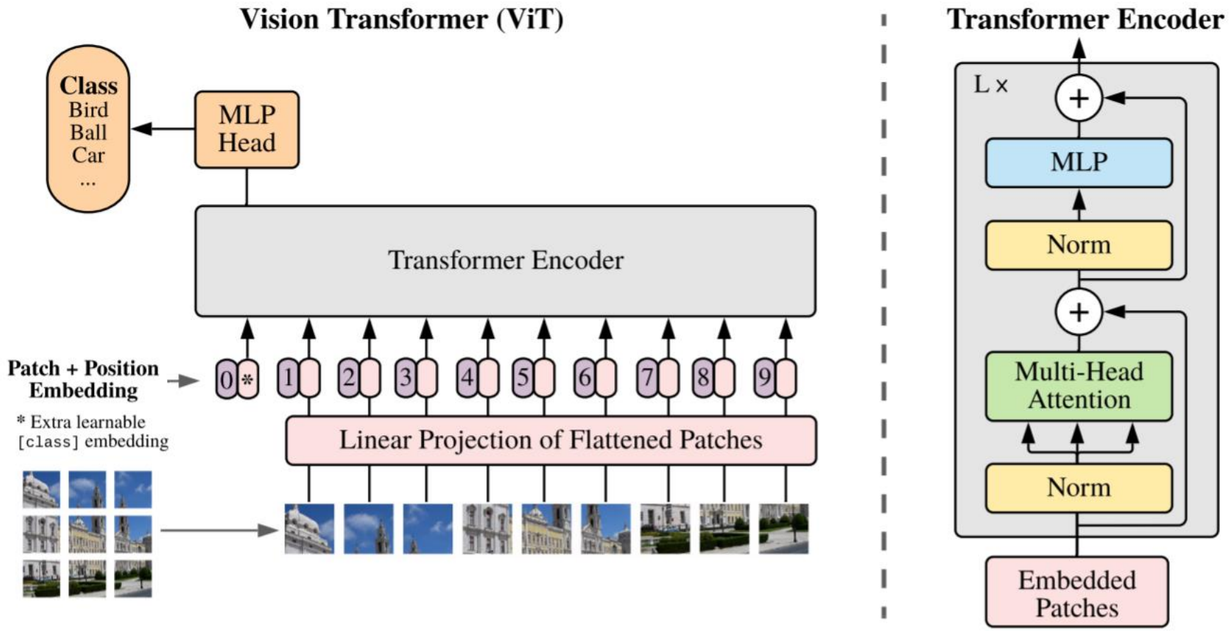


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

Are CNNs already old-fashioned?

- Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale."
- Liu, Zhuang, et al. "A convnet for the 2020s." *CVPR* 2022.

- Not part of the curriculum

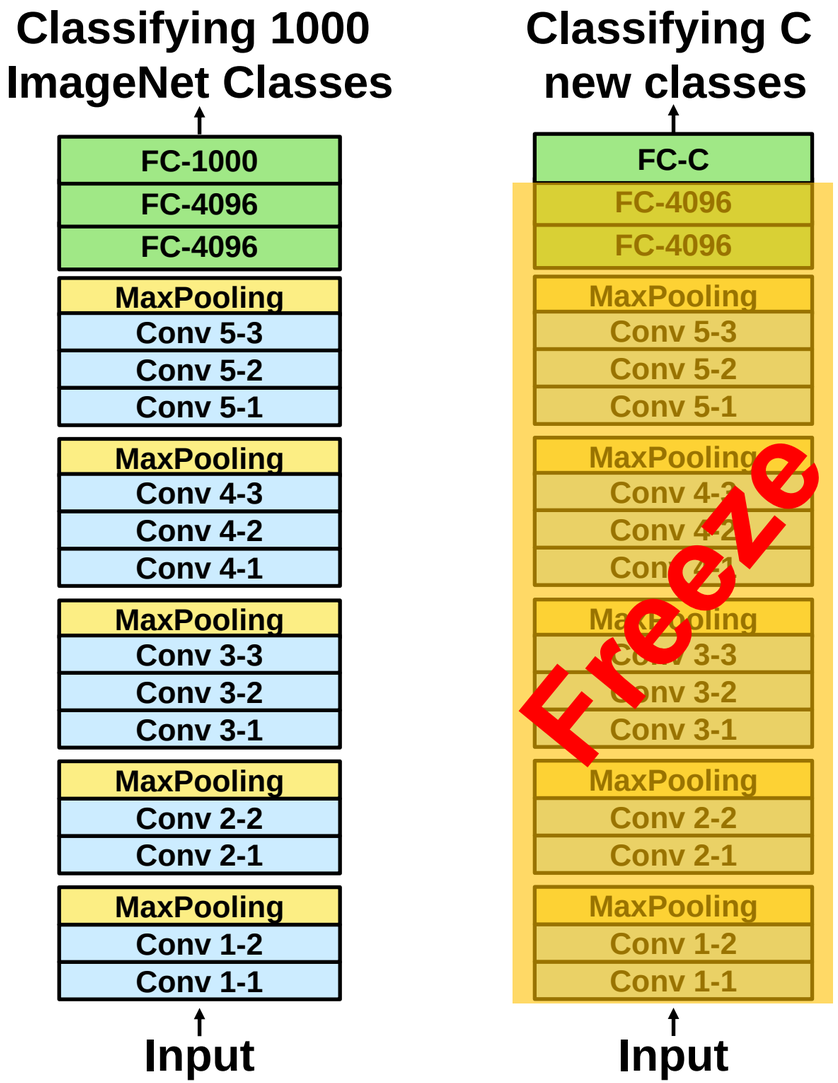


Transfer Learning

- Recall from last lecture:
- You never have enough training data

- What if I could get help from people who have more data than I?
  - *People have very good models for ImageNet classification. Hmm!*

Transfer Learning



Transfer Learning

	Very similar dataset	Very different dataset
Very little data	Use classifier on top layer	Worst case – very hard Try extracting features at different depths and use classical ML
More data	Finetune the last few layers	Finetune the last few layers, and train the rest of the layers with a lower learning-rate

Inspired by cs231n

Transfer Learning

- Transfer learning is everywhere!
  - *Many of the advances of deep learning in computer vision involve using a network trained for classification in a different setting*

You have learned

- Importance of input normalization
- Batch normalization
- Optimizing
  - Learning rate annealing
  - SGD+momentum
  - RMSProp
  - Adam
- Important architectures
  - VGG
  - Inception v1
  - Resnet
- Transfer learning
- Heatmaps
- Adversarial examples

Group forming

- Remember to sign up as a group on learn!