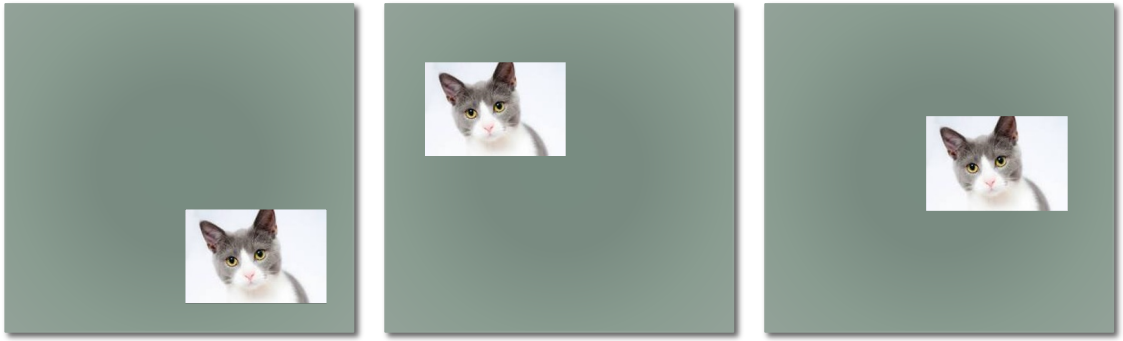
CONVOLUTIONAL NETWORKS

Neural networks

- Fully connected networks
  - Neuron
  - Non-linearity
  - Softmax layer
- DNN training
  - Loss function and regularization
  - SGD and backprop
  - Learning rate
  - Overfitting – dropout

Convolutional Neural Networks (CNNs)

- Consider an image of a cat. DNNs need different neurons to learn every possible location a cat can be



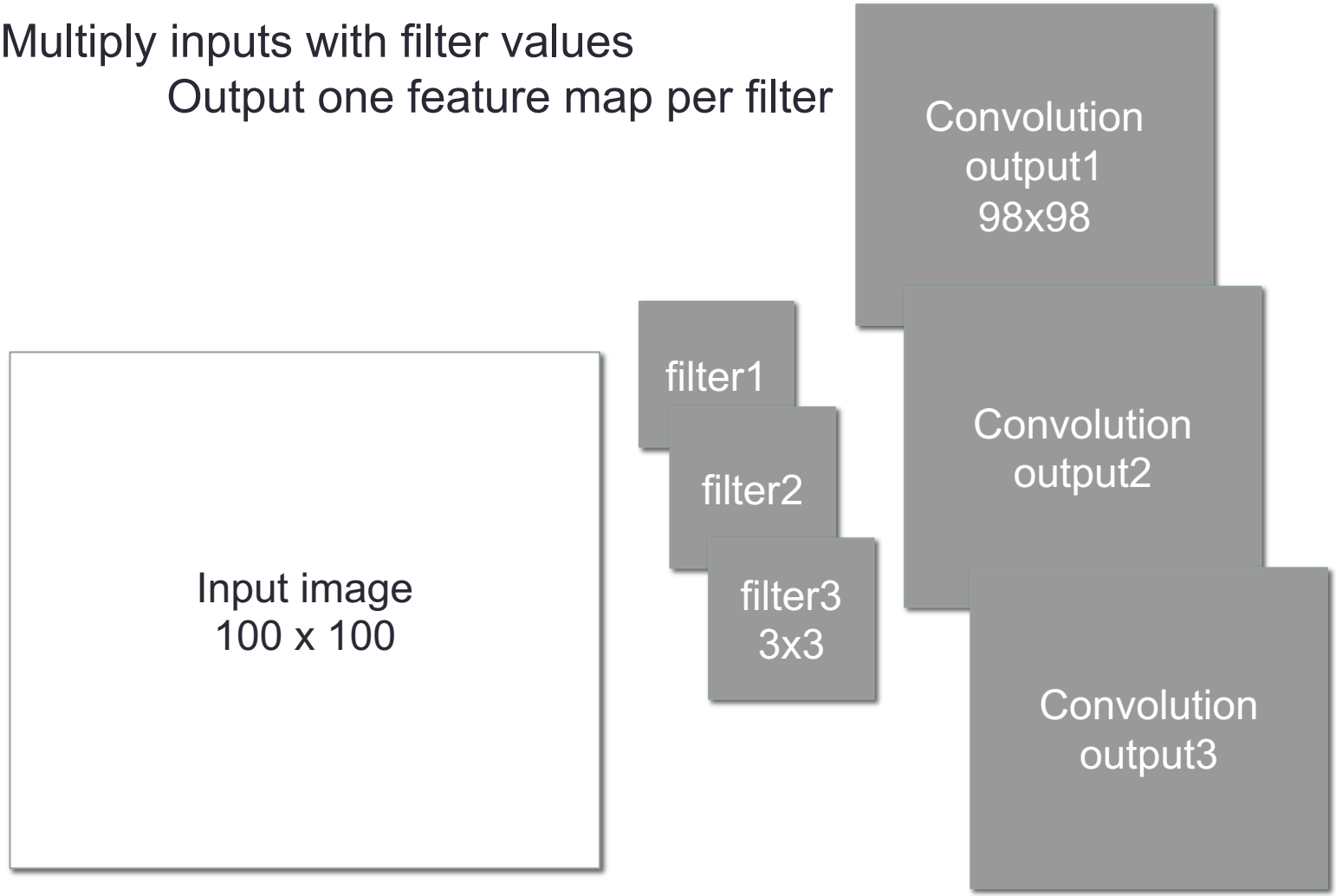
- Can we use the same parameters to learn that a cat exists regardless of location?

- 2 parts: convolutional layer and pooling layer

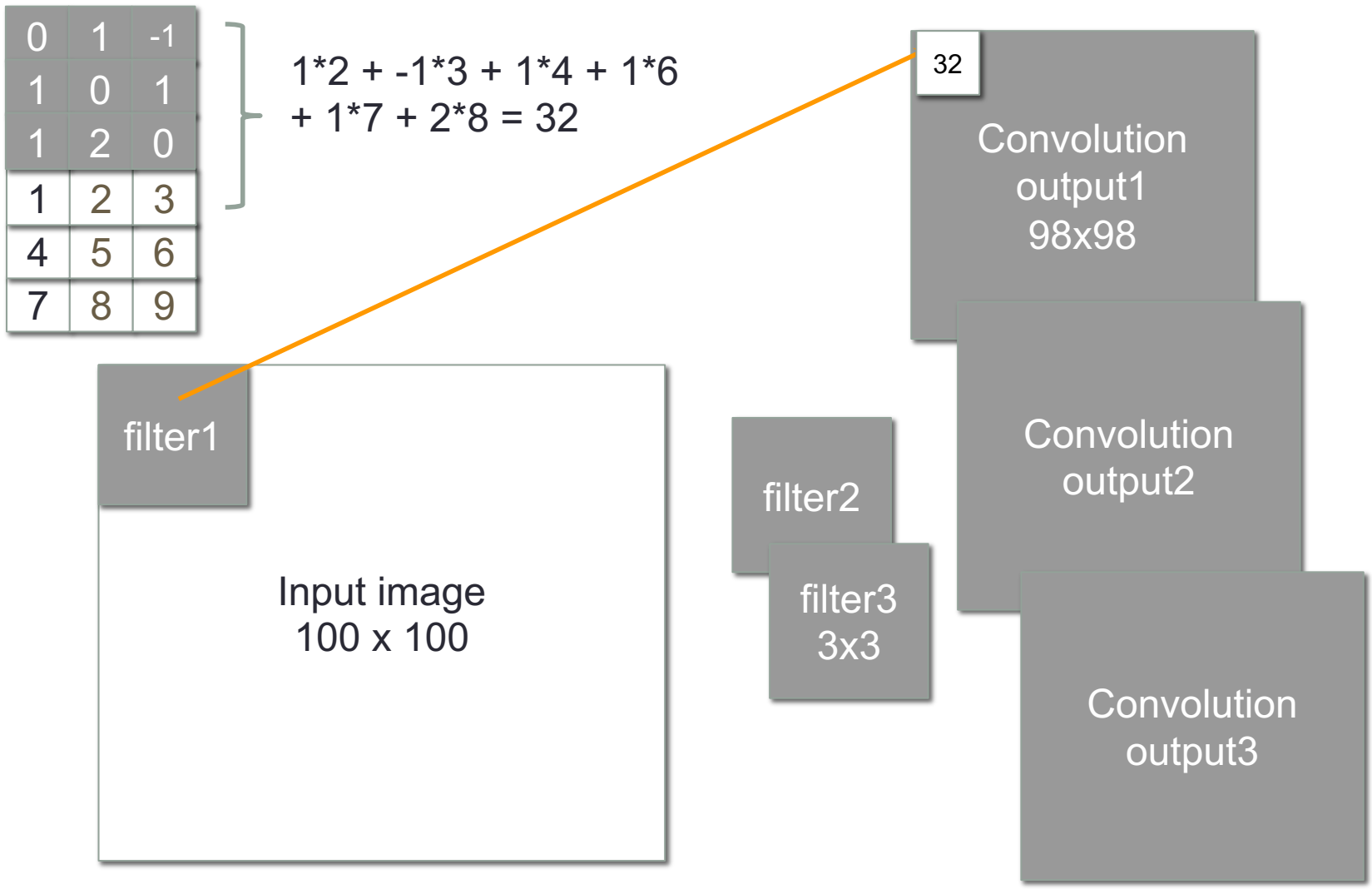
Convolutional filters

Multiply inputs with filter values

Output one feature map per filter

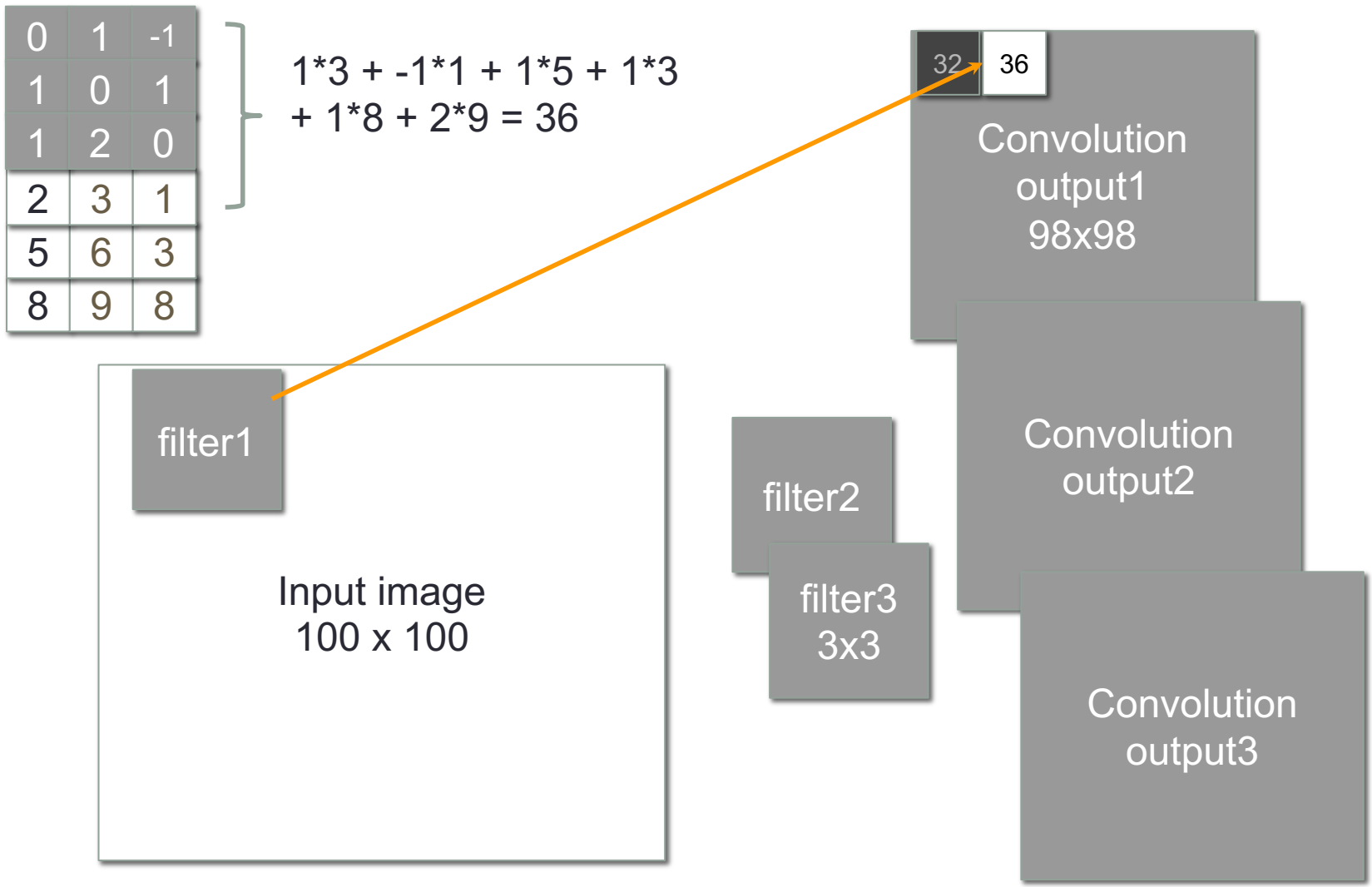


Convolutional filters

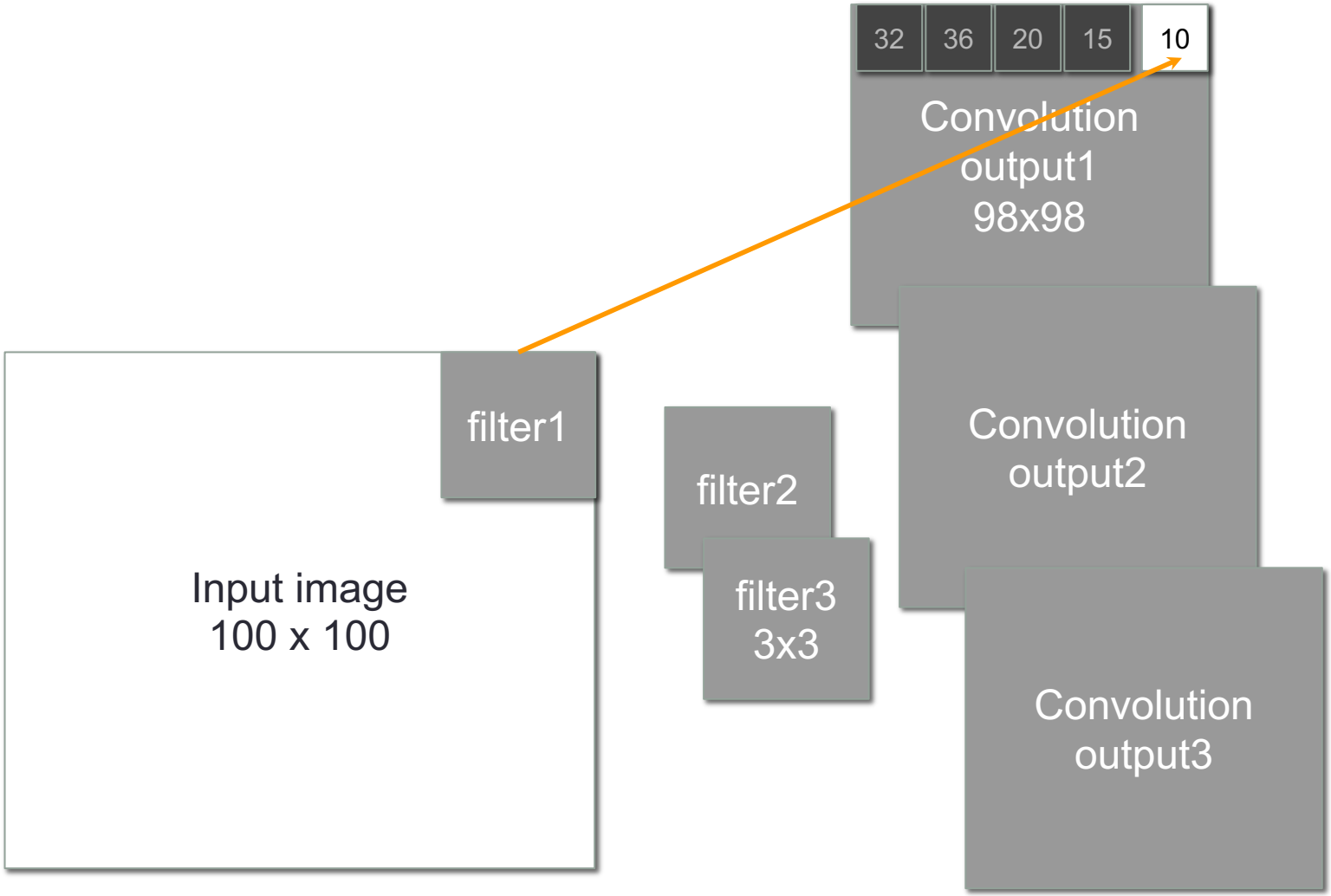


Convolutional filters

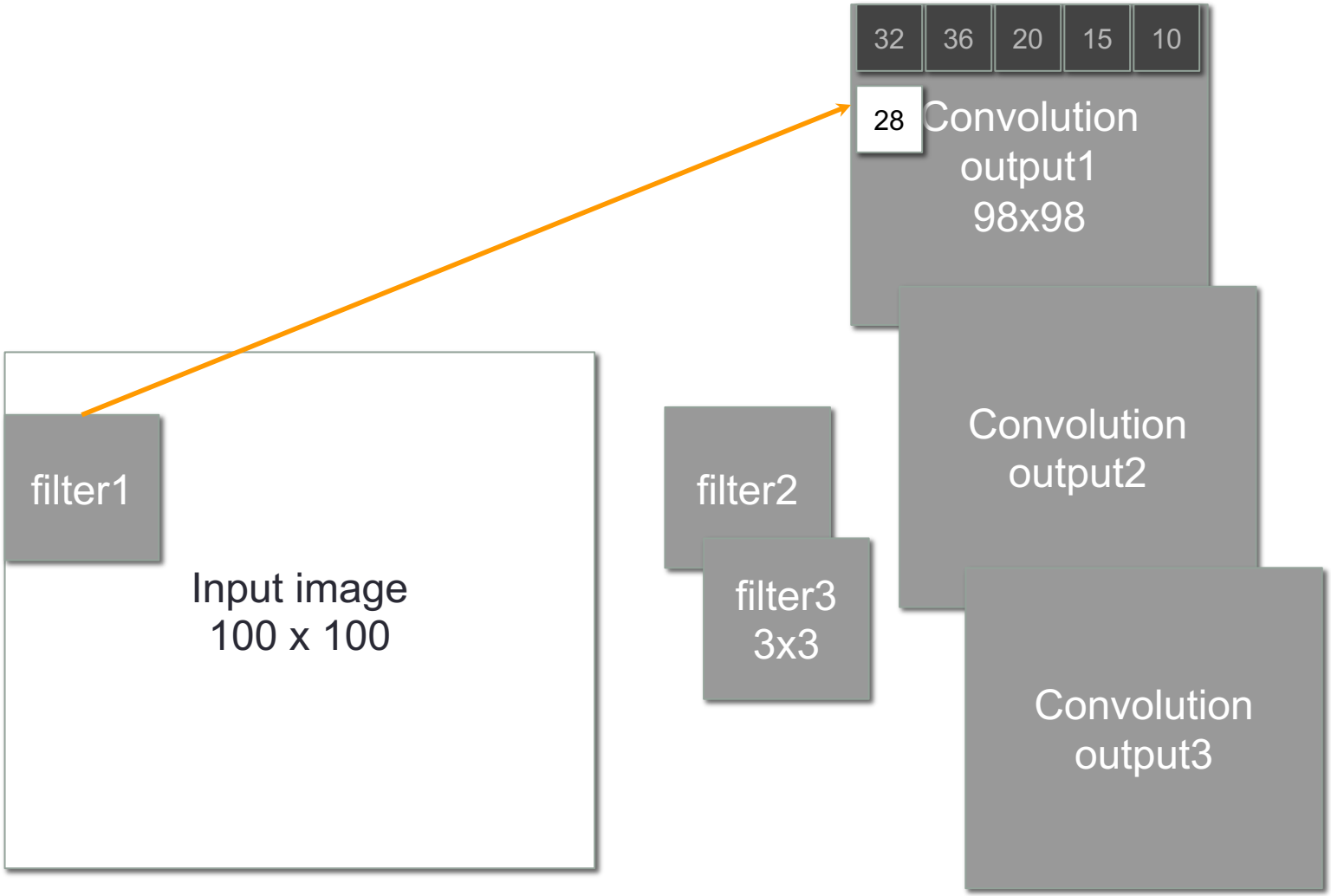
Stride of 1



Convolutional filters

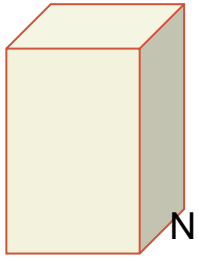


Convolutional filters



Convolutional filters

N filters means N feature maps
You get a 3 dimensional output



32	36	20	15	10
28	72	0	12	50
32	36	18	9	2
17	6	2	17	1

16

filter2

filter1

filter3
3x3

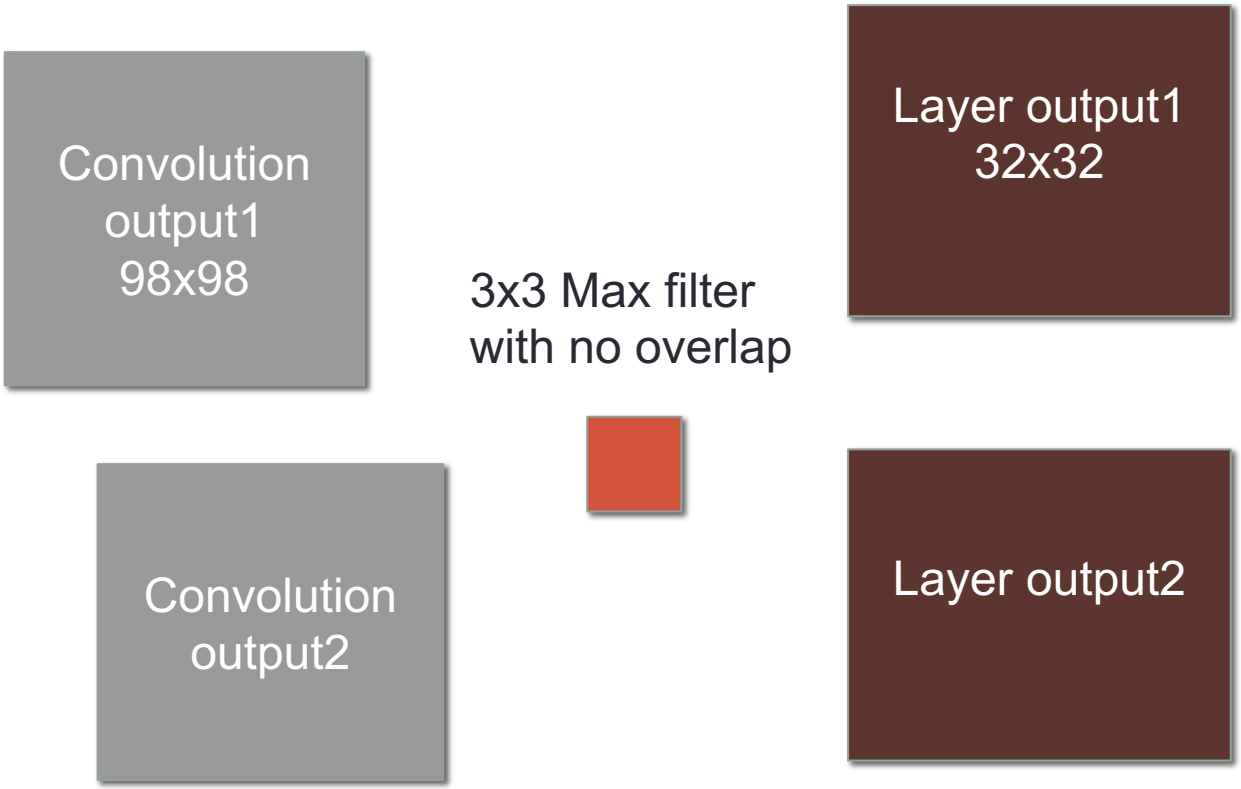
Input image
100 x 100

Convolution
output2

Convolution
output3

Pooling/subsampling

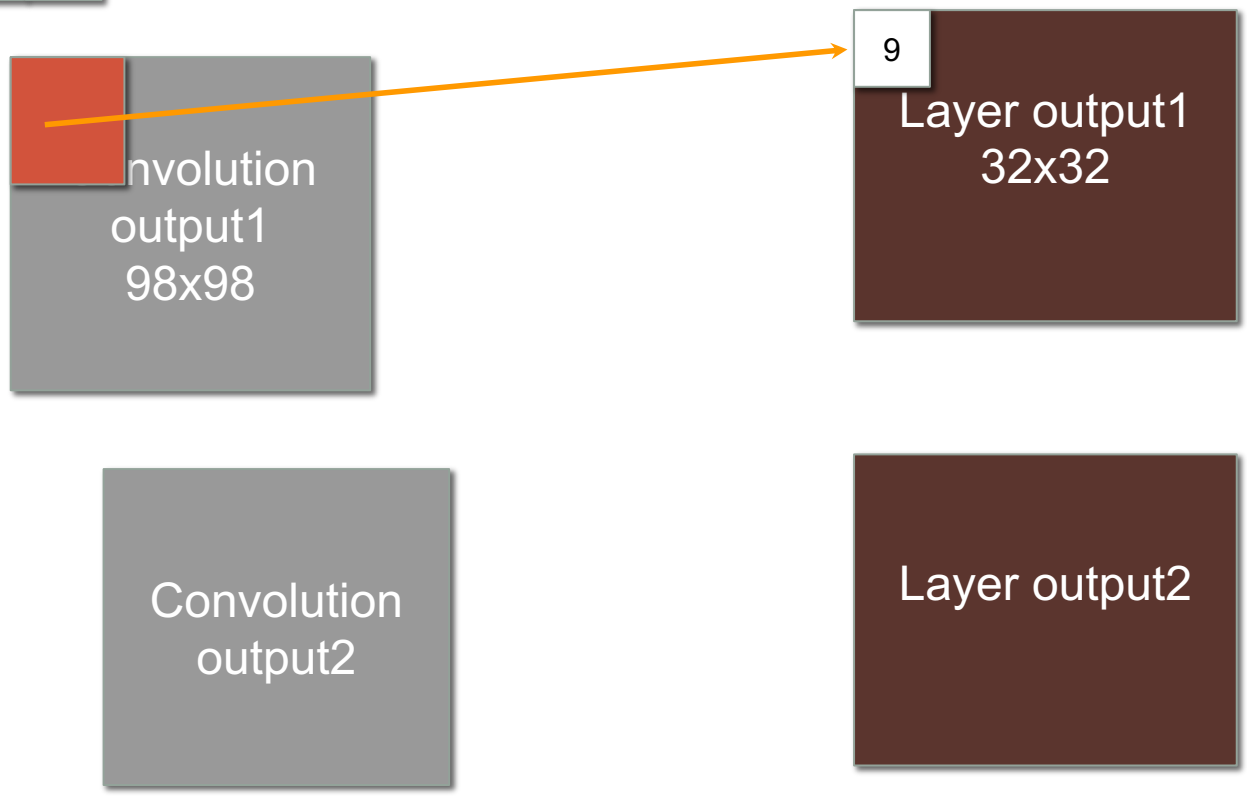
Reduce dimension of the feature maps



Pooling/subsampling

1	2	3
4	5	6
7	8	9

Max = 9

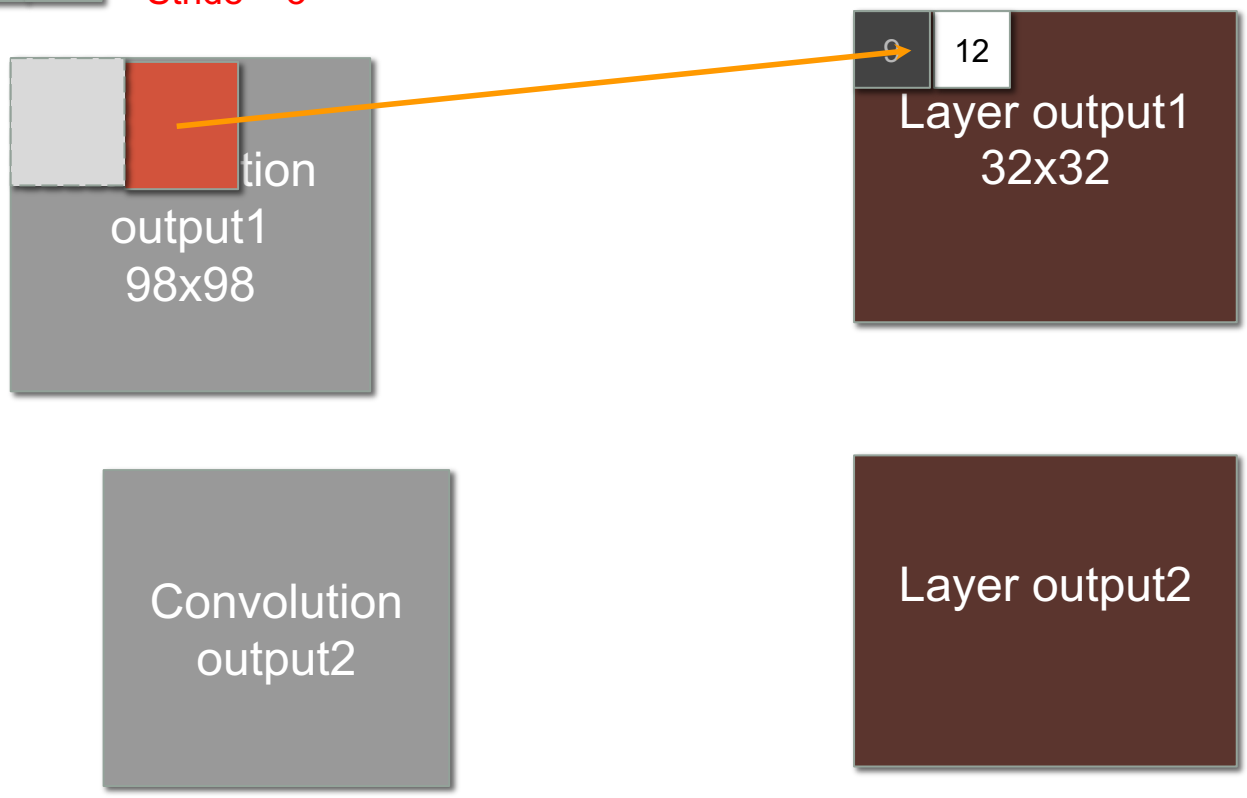


Pooling/subsampling

5	2	1
5	7	1
9	5	12

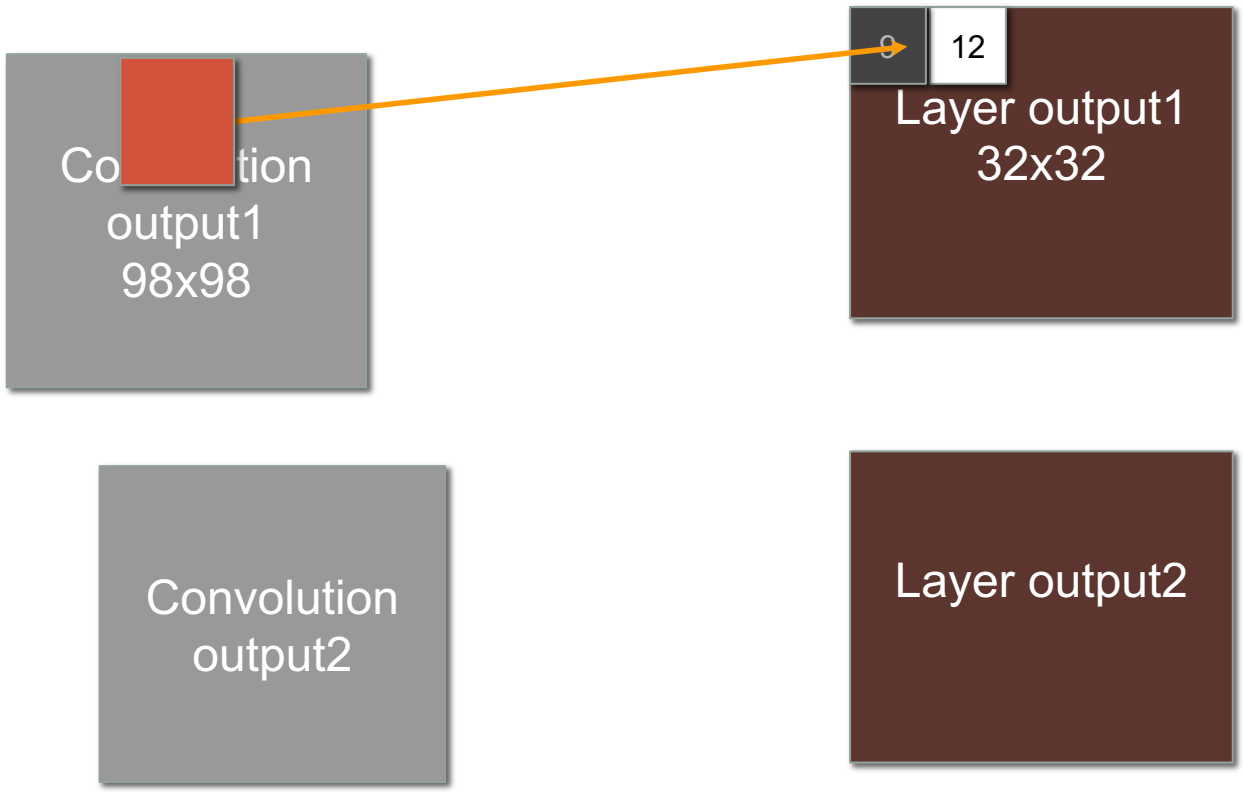
Max = 12

Stride = 3



Pooling/subsampling

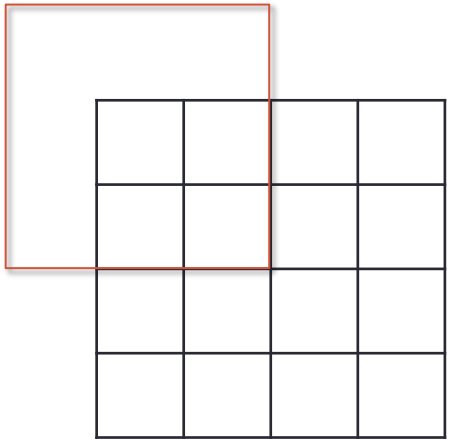
Can use other functions besides max
Example, average



Convolution puzzle

5 filters 3x3 filter pad, stride 1, pad 1

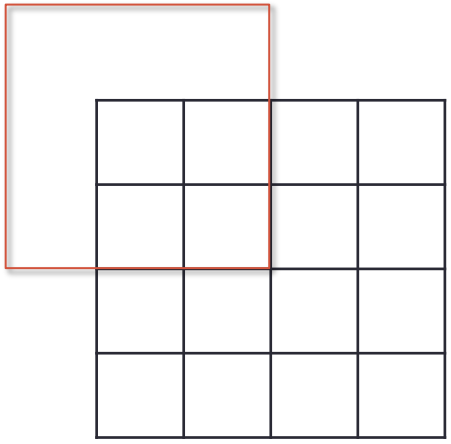
What is the output size?

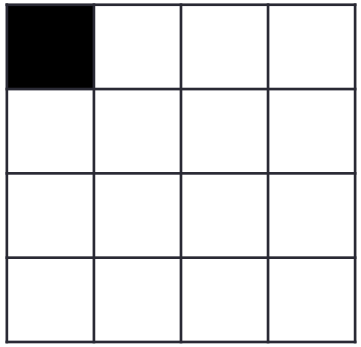


Convolution puzzle

5 filters 3x3 filter pad, stride 1, pad 1

What is the output size?

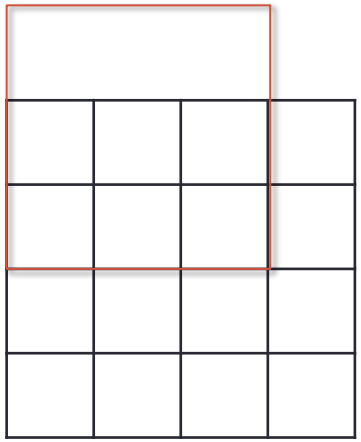


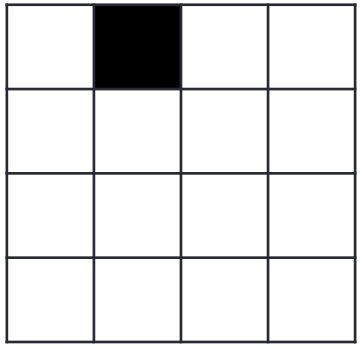


Convolution puzzle

5 filters 3x3 filter pad, stride 1, pad 1

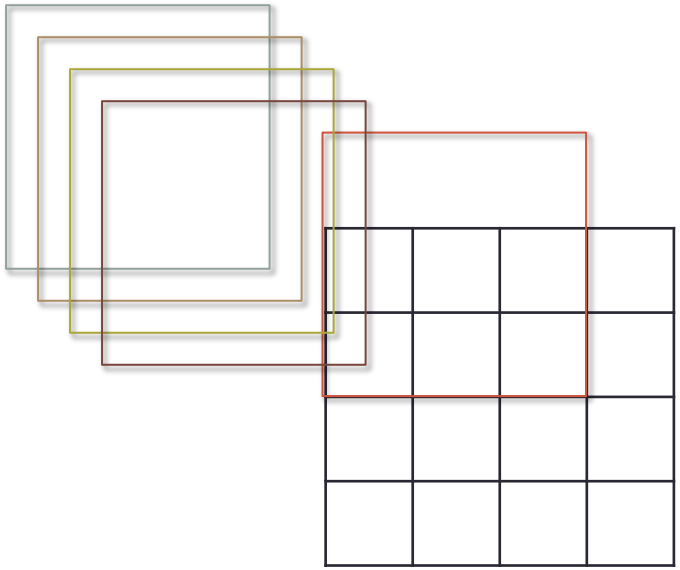
What is the output size?

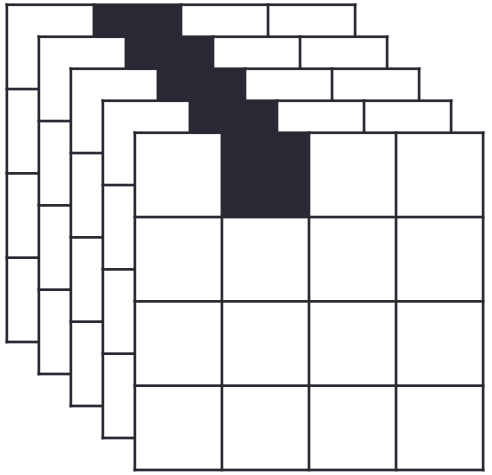




Convolution puzzle

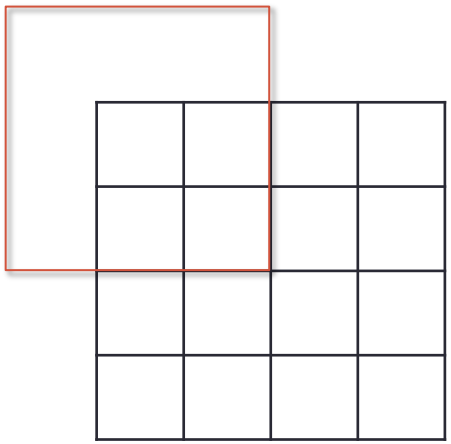
5 filters 3x3 filter pad, stride 1, pad 1





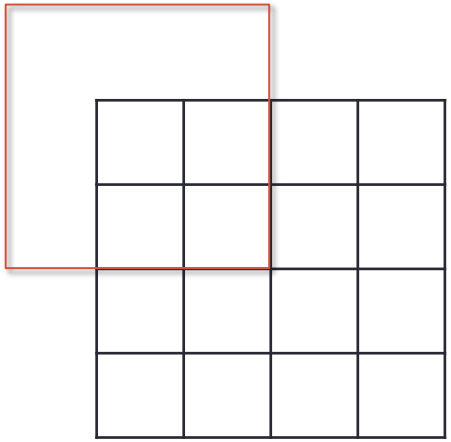
Convolution puzzle

3x3 filter pad, stride 2, pad 1



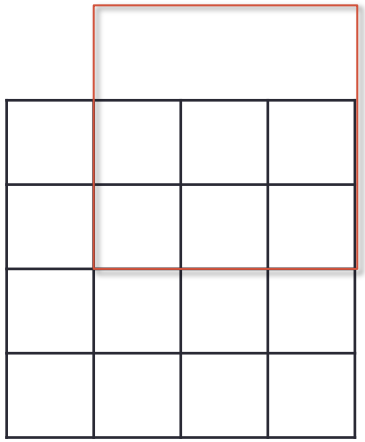
Convolution puzzle

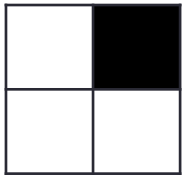
3x3 filter pad, stride 2, pad 1



Convolution puzzle

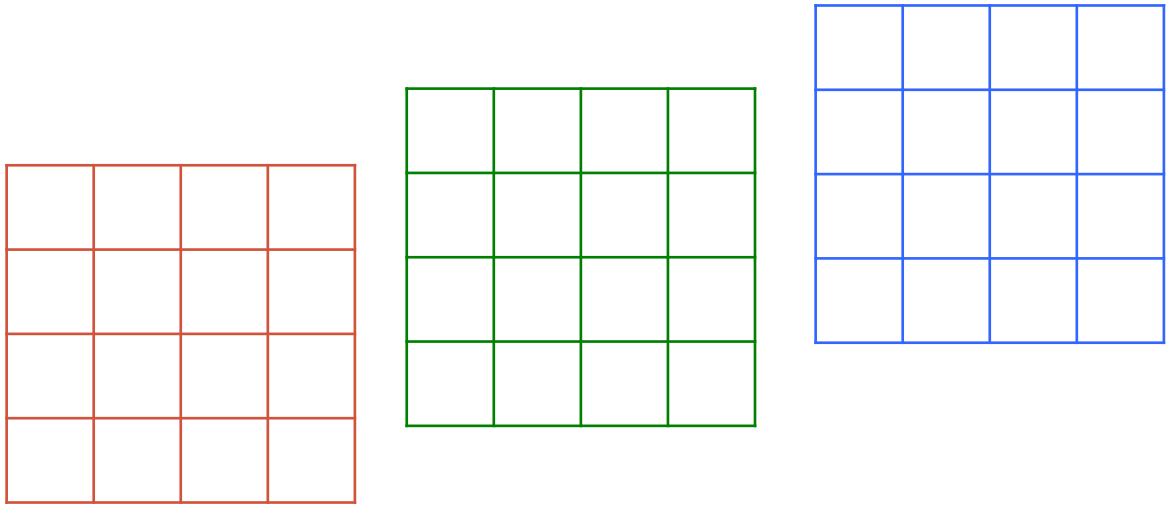
3x3 filter pad, stride 2, pad 1





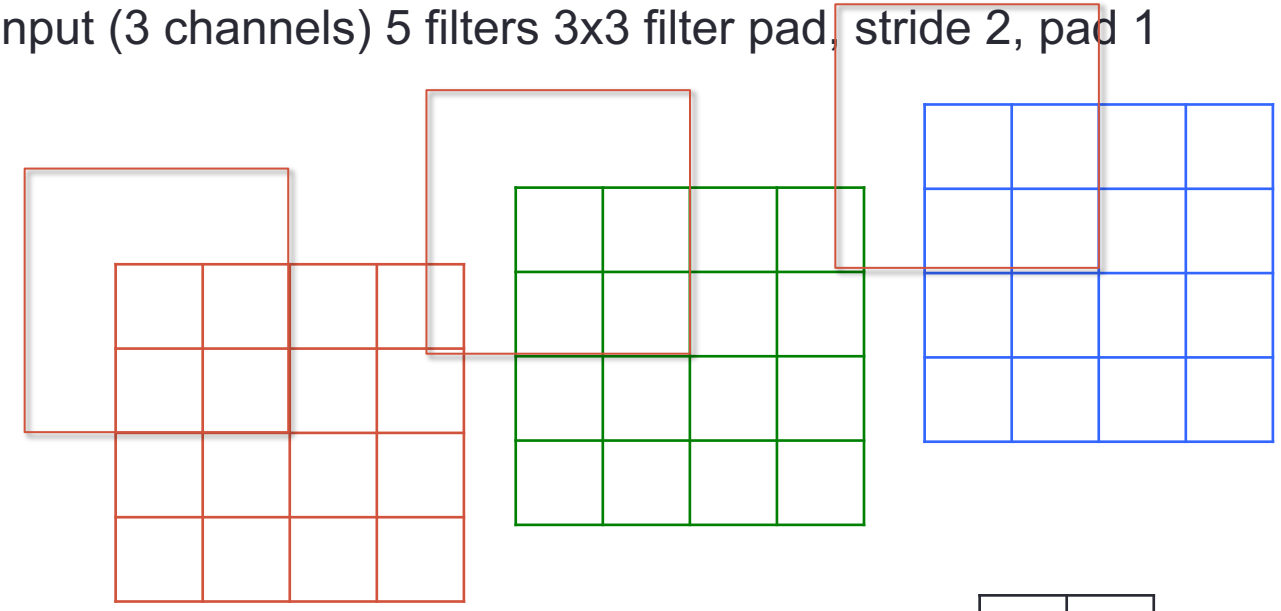
Convolution puzzle

RGB input (3 channels) 5 filters 3x3 filter pad, stride 2, pad 1

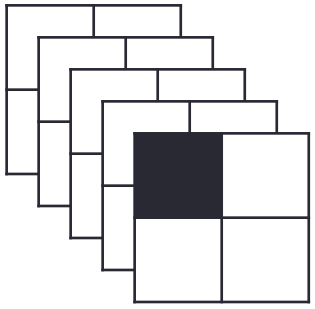


Convolution puzzle

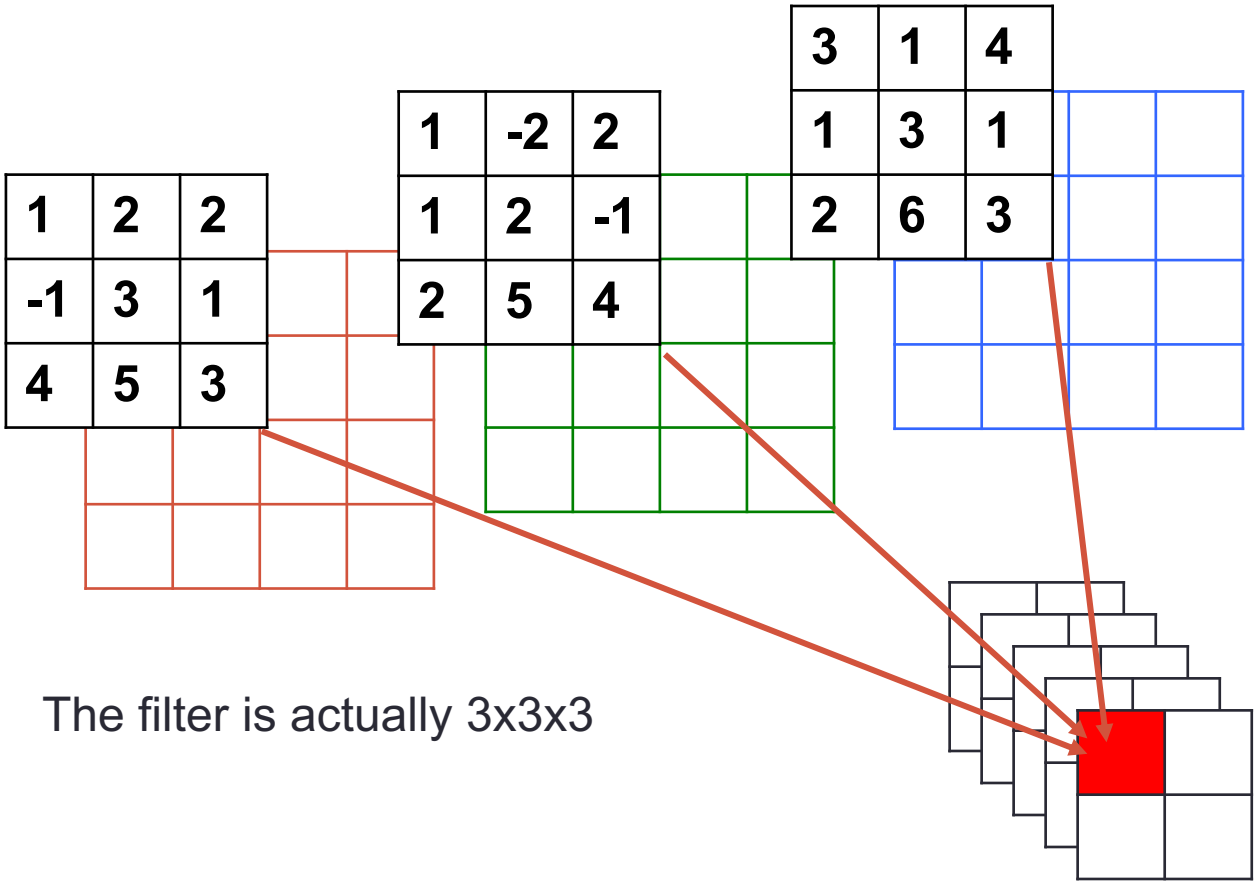
RGB input (3 channels) 5 filters 3x3 filter pad, stride 2, pad 1



The filter is actually 3x3x3



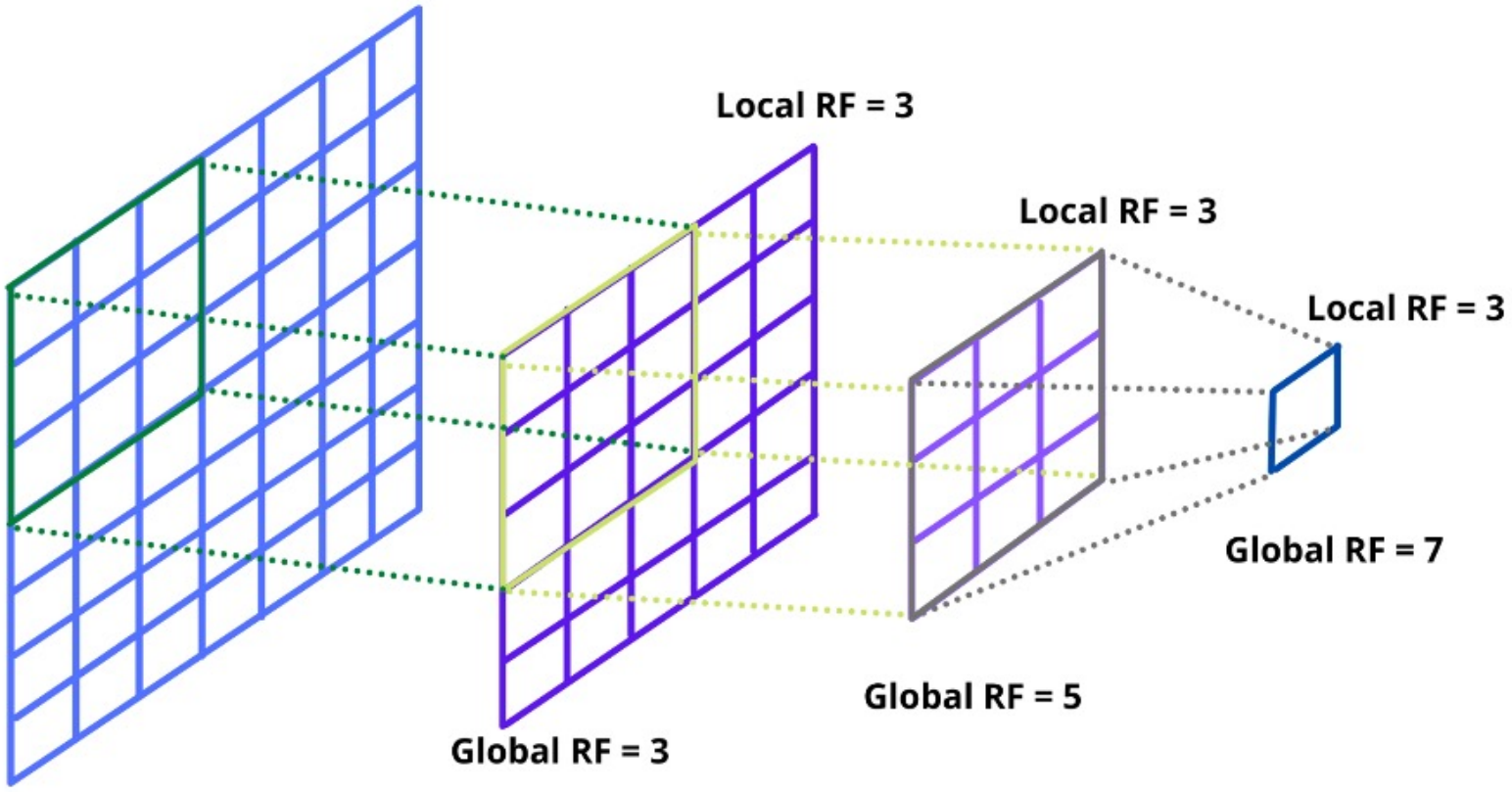
Convolution puzzle



The filter is actually 3x3x3

Receptive field

- The size of the input the filter covers



Batch normalization

- Recent technique for (implicit) regularization
- Normalize every mini-batch at various batch norm layers to standard Gaussian (different from global normalization of the inputs)
- Place batch norm layers before non-linearities
- Faster training and better generalizations

For each mini-batch that goes through batch norm

1. Normalize by the mean and variance of the mini-batch for each dimension
2. Shift and scale by learnable parameters

Replaces dropout in some networks

https://arxiv.org/abs/1502.03167

$$\hat{x} = \frac{x - \mu_b}{\sigma_b}$$

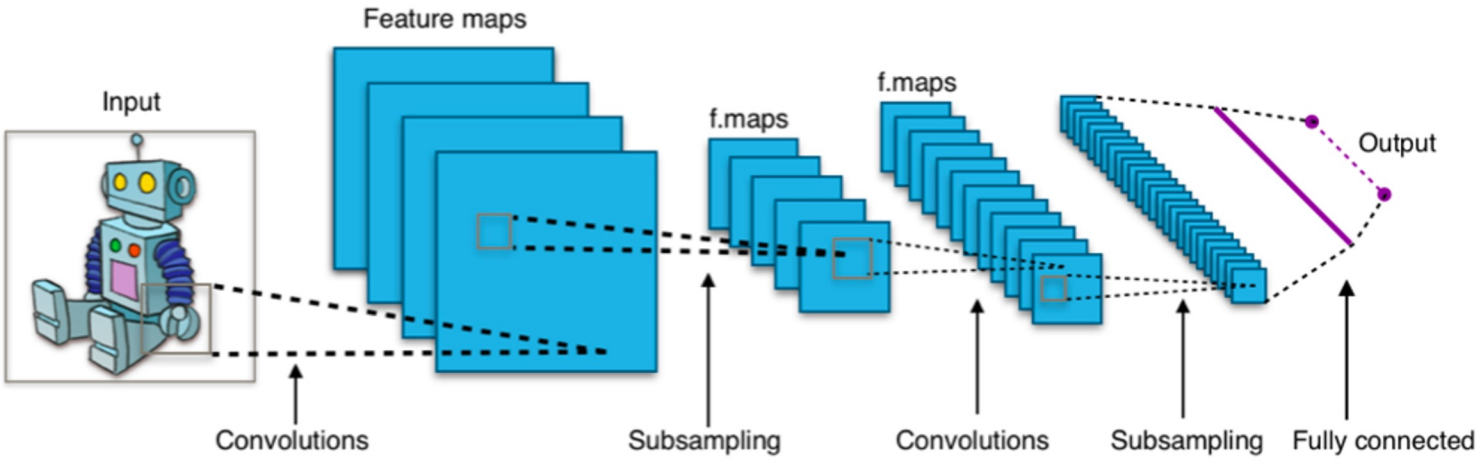
$$y = \alpha\hat{x} + \beta$$

Dropout vs batchnorm

- You can add dropout in the hidden layers (0-0.5)
- Or input layers (0-0.2 is typical)
  - “Noising” the inputs, data augmentation
- Dropout in computer vision
  - use batchnorm instead (convolution layers leak output when using dropout)
  - Circumvent with dropblock https://arxiv.org/abs/1810.12890

CNN overview

- Filter size, number of filters, filter shifts, and pooling rate are all parameters
- Usually followed by a fully connected network at the end
  - CNN is good at learning low level features
  - DNN combines the features into high level features and classify



https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

Lab

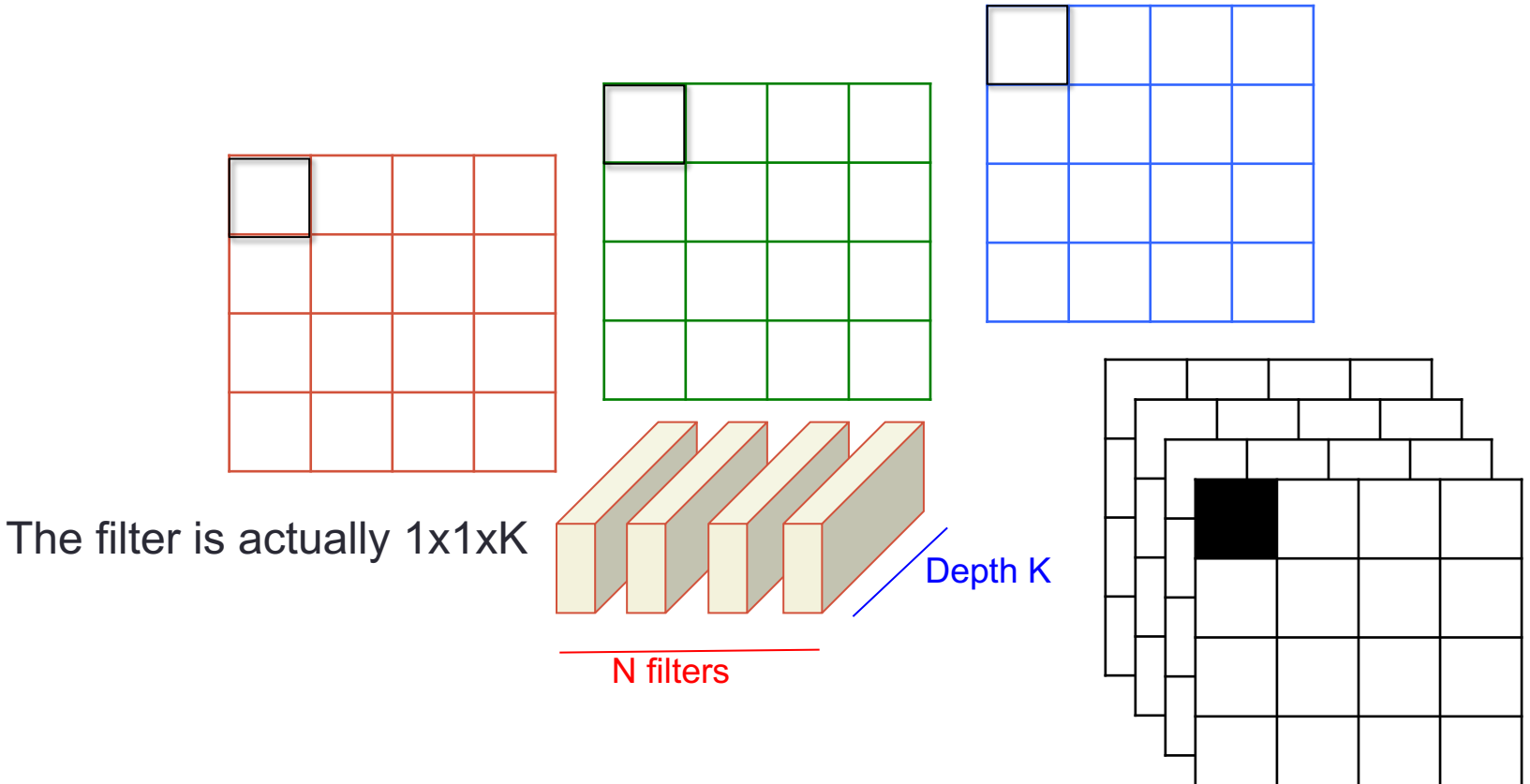
https://colab.research.google.com/drive/12XydvRkaZgULZ31UA5E84_3NWIBJTWmn?usp=sharing

Common schemes

- INPUT -> [CONV -> RELU -> POOL]*N -> [FC -> RELU]*M -> FC
- INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL]*N -> [FC -> RELU]*M -> FC
- If you working with images, just use a winning architecture.

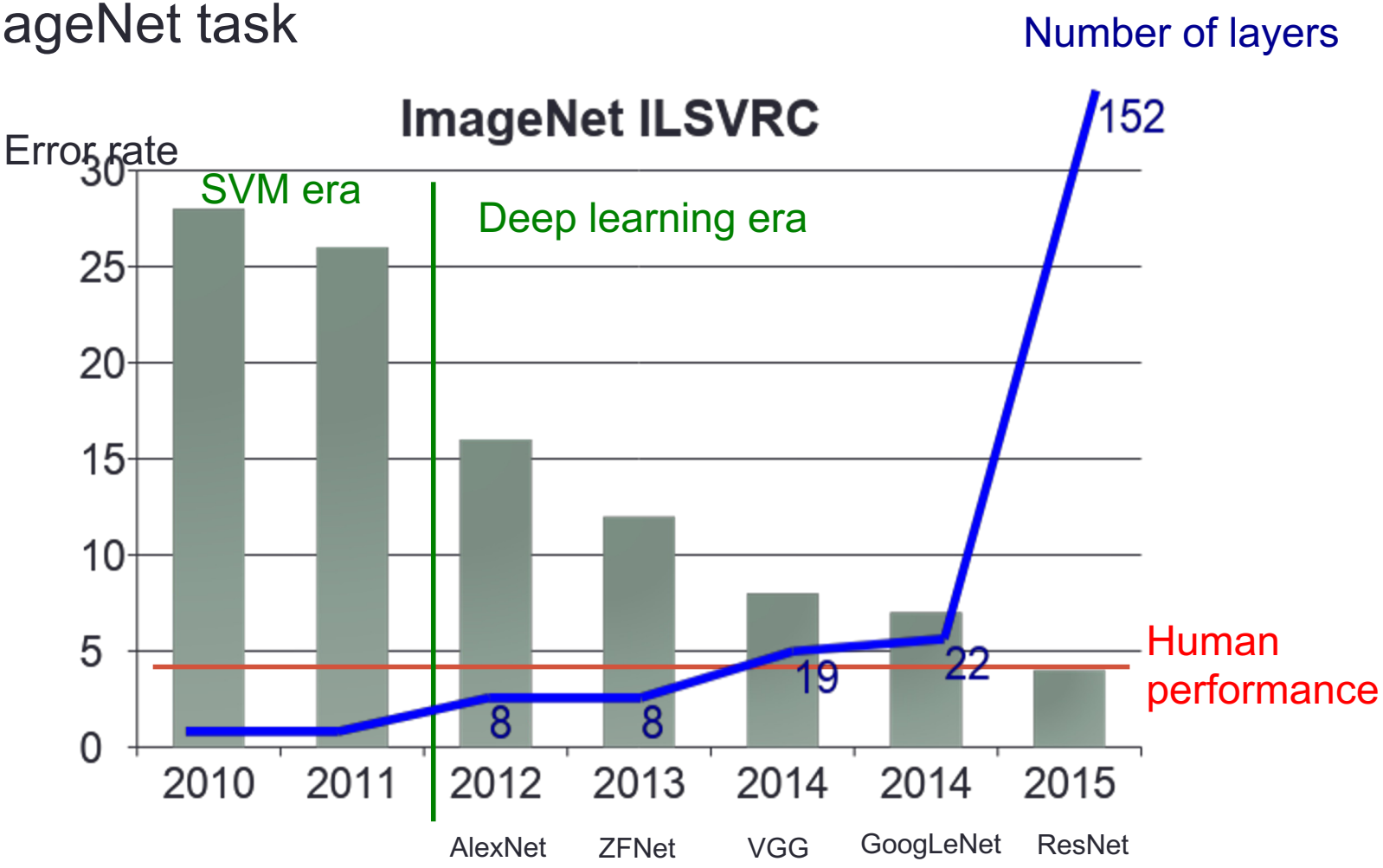
1x1 convolution

Reduces the dimension of feature maps



A brief history of imagenet architectures

- ImageNet task



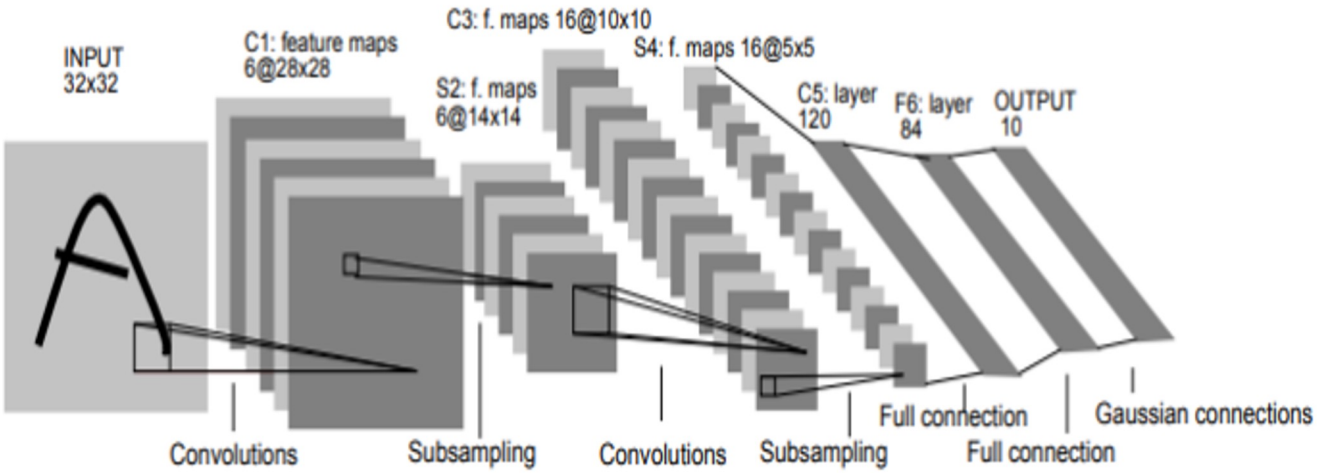
Olga Russakovsky, ImageNet Large Scale Visual Recognition Challenge, 2014 https://arxiv.org/abs/1409.0575

LeNet

Convolutions and poolings followed by fully connected layers

Tanh activations

Ability to handle larger images limited by compute

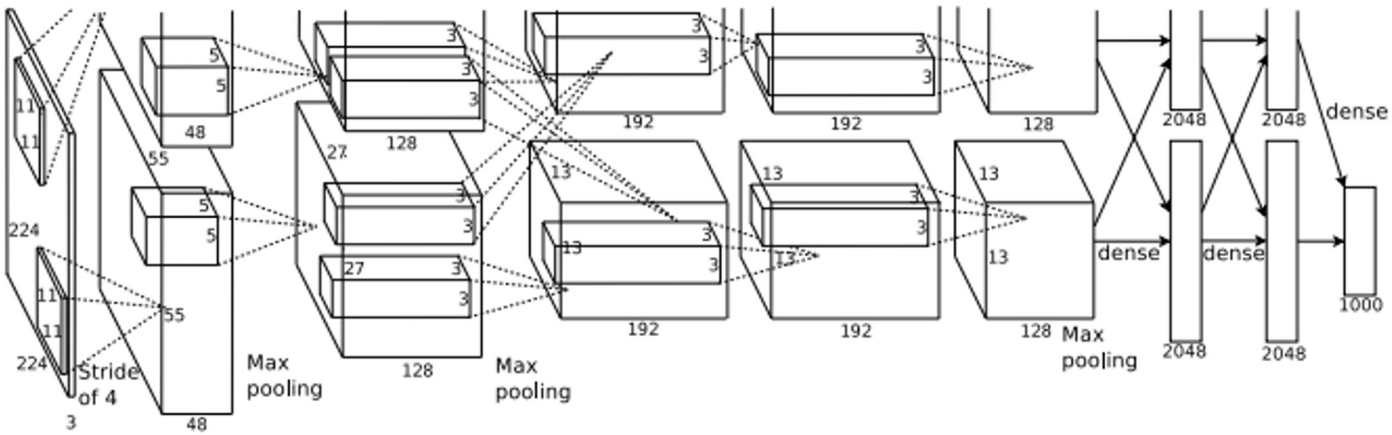


Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition,” 1998.

AlexNet

Convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum

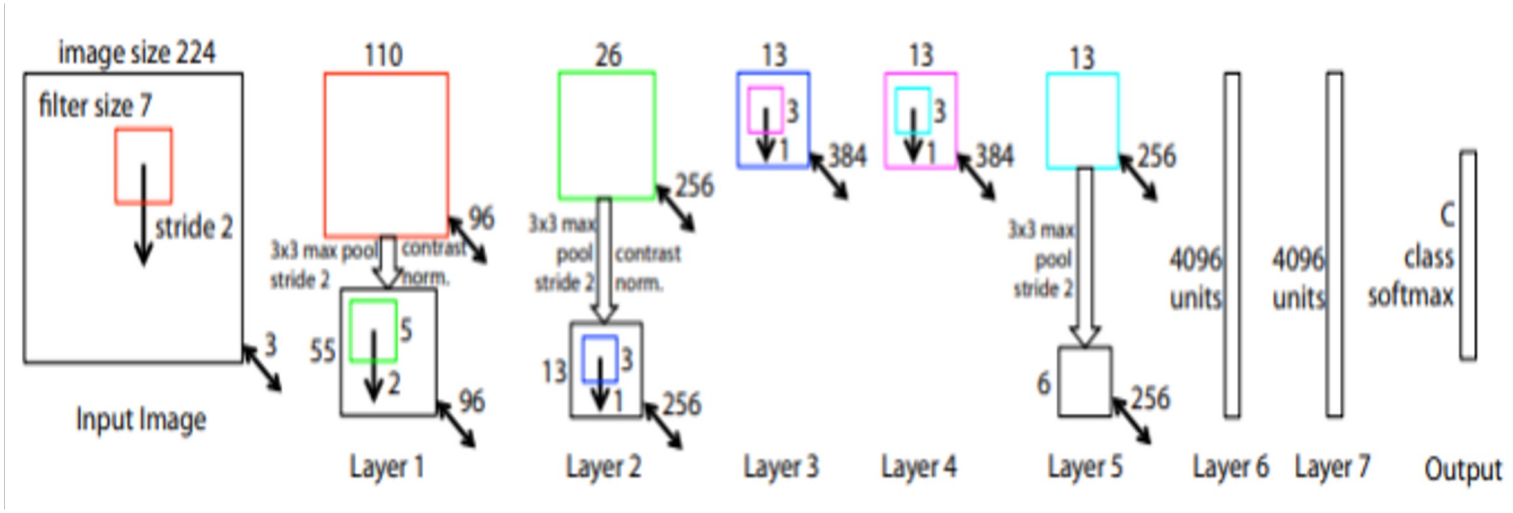
Two pipelines to fit into two GPUs



Alex Krizhevsky, et al. “ImageNet Classification with Deep Convolutional Neural Networks”, 2012.

ZFNet

Tweaking hyperparameters



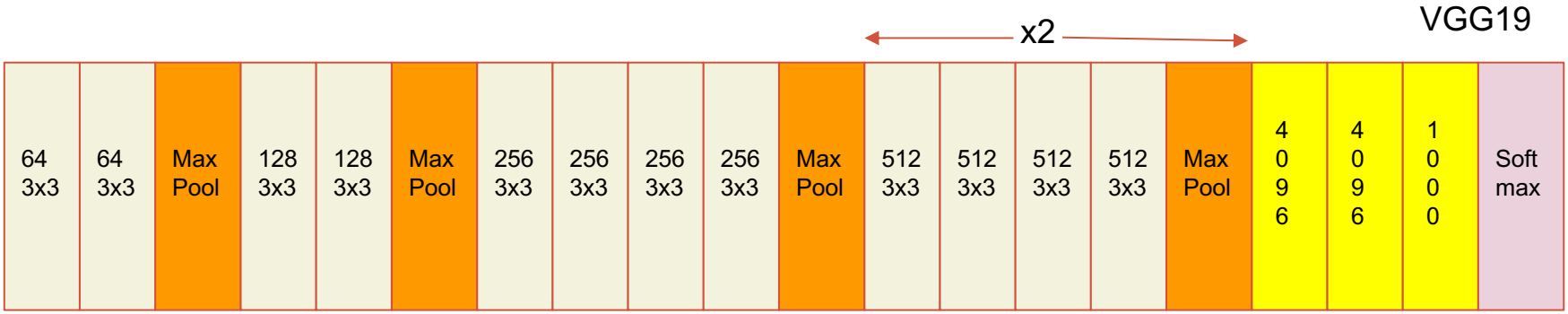
Matthew D. Zeiler, Rob Fergus, “Visualizing and Understanding Convolutional Networks,” 2013

VGG

Uniform 3x3 convolutional filters

19 layers! Pushing the limits of conventional wisdom at that time.

Used by many since pre-train weights are publically available



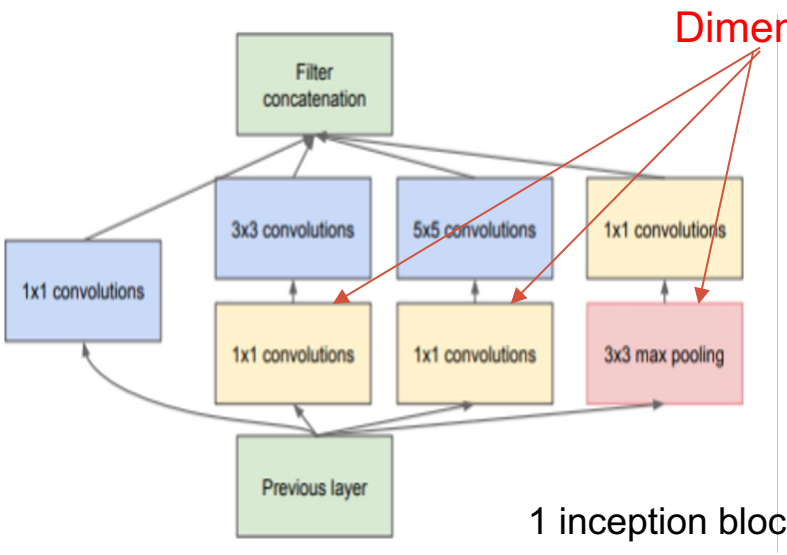
Karen Simonyan, Andrew Zisserman"Very Deep Convolutional Networks for Large-Scale Image Recognition" 2014

GoogLeNet (Inception v1)

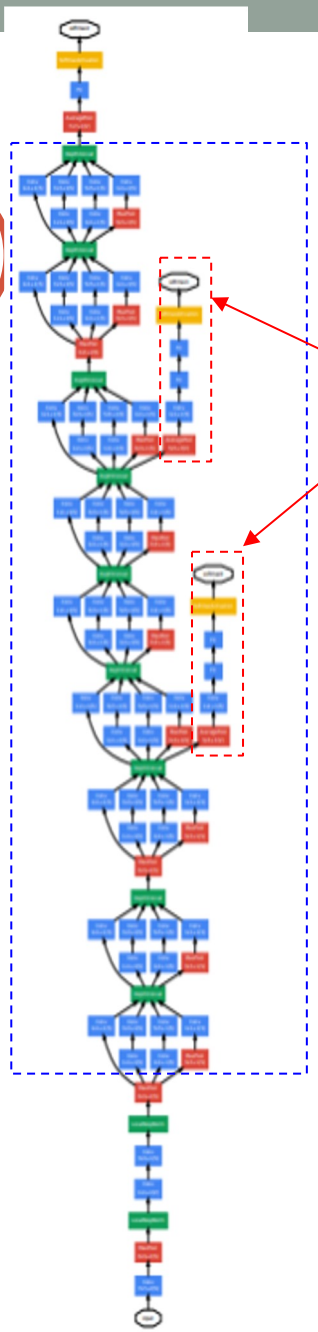
Multiple filter sizes per layer (objects come in different scales)

Dimensionality reduction via 1x1 convolution

Multiple softmax losses to help the gradient problem



1 inception block

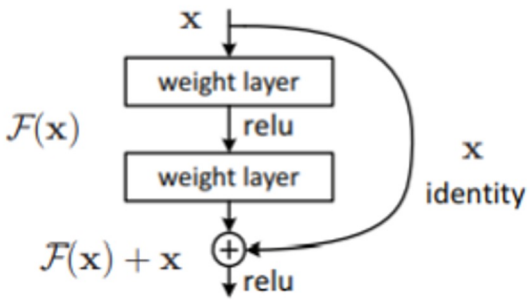


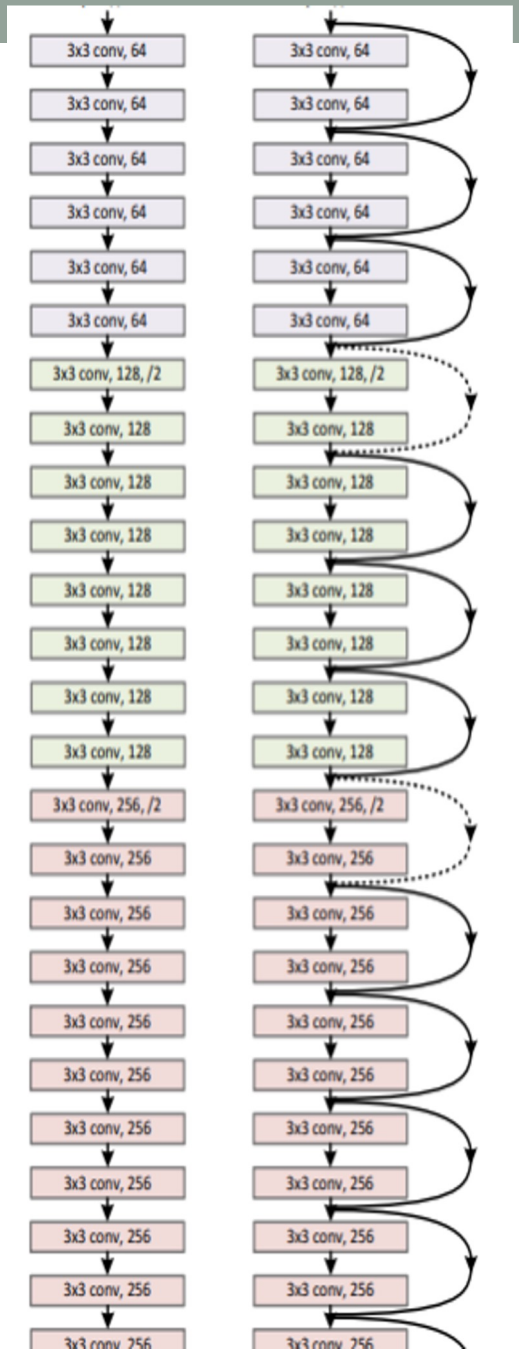
Christian Szegedy, et al. “Going Deeper with Convolutions,” 2014

ResNet (Residual Network)

Batch norm

Extra “skip connections” to reduce the vanishing gradient problem

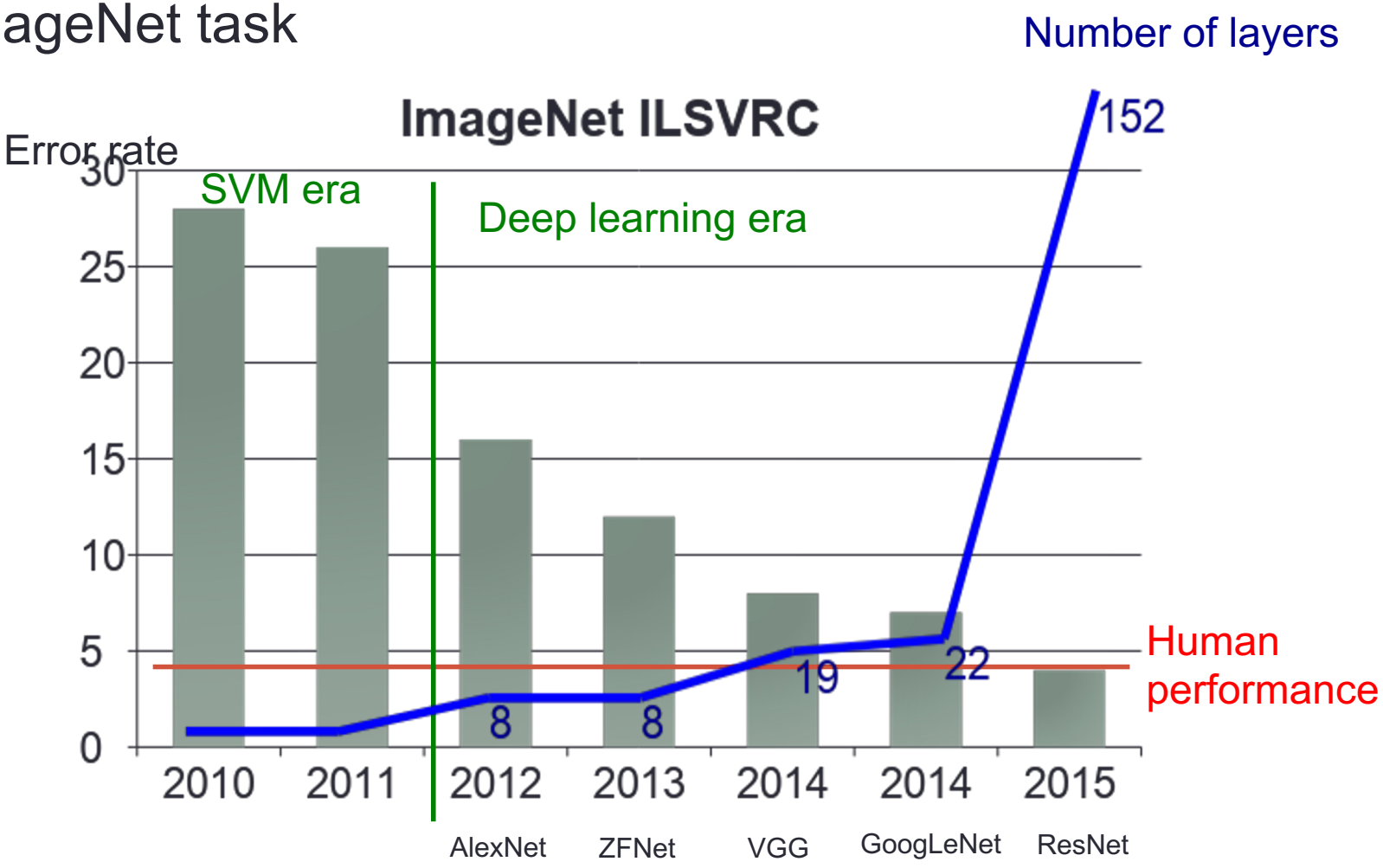




Kaiming He, et al. “Deep Residual Learning for Image Recognition” 2015

A brief history of imagenet architectures

- ImageNet task



Olga Russakovsky, ImageNet Large Scale Visual Recognition Challenge, 2014 https://arxiv.org/abs/1409.0575

Inception v2+v3

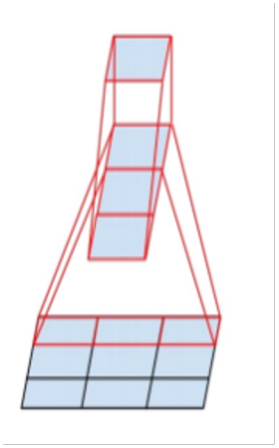
Implement 5x5 with two 3x3s

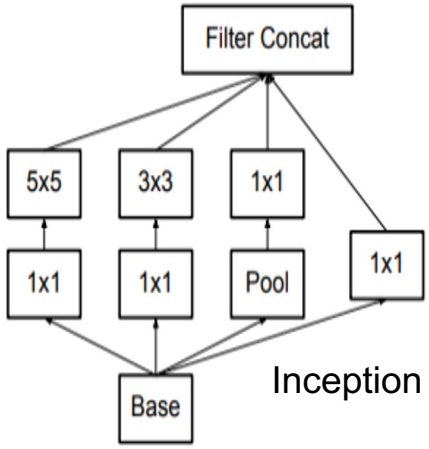
Factorized convolution

3x3 -> 3x1 and 1x3

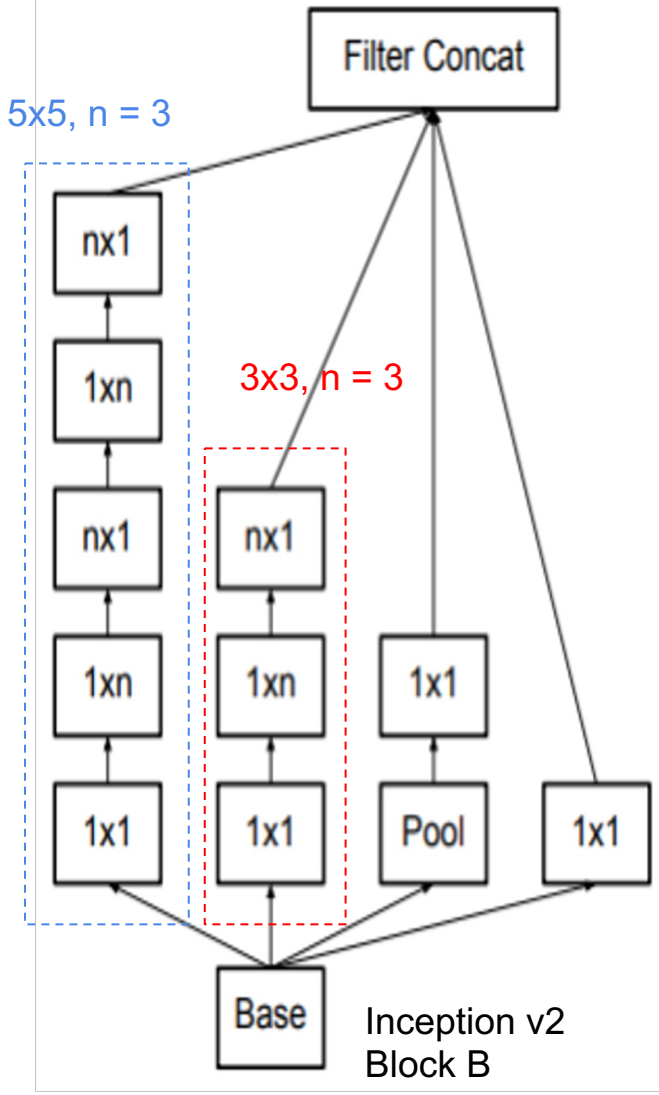
3 types of inception blocks

RMSprop, Batch norm, label smoothing





Inception v1



Inception v2
Block B

Christian Szegedy, “Rethinking the Inception Architecture for Computer Vision,” 2015

Xception

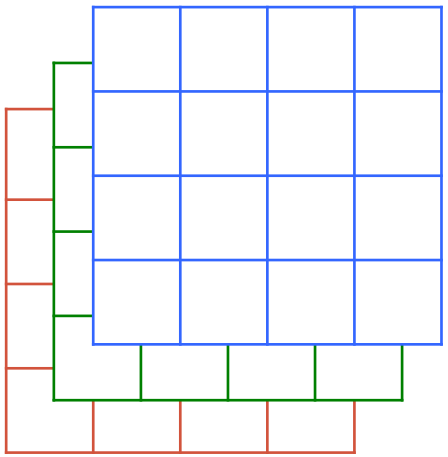
Depthwise separable convolution: two-step convolution

1. Depthwise convolution
2. 1x1 convolution

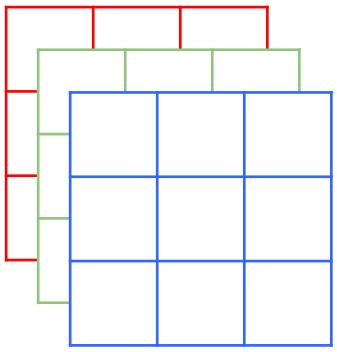
Typical convolution 3x3 filter is 3x3xinput channel

Depthwise convolution 3x3 filter is 3x3x1
1 filter per input channel

input



One regular convolution filter



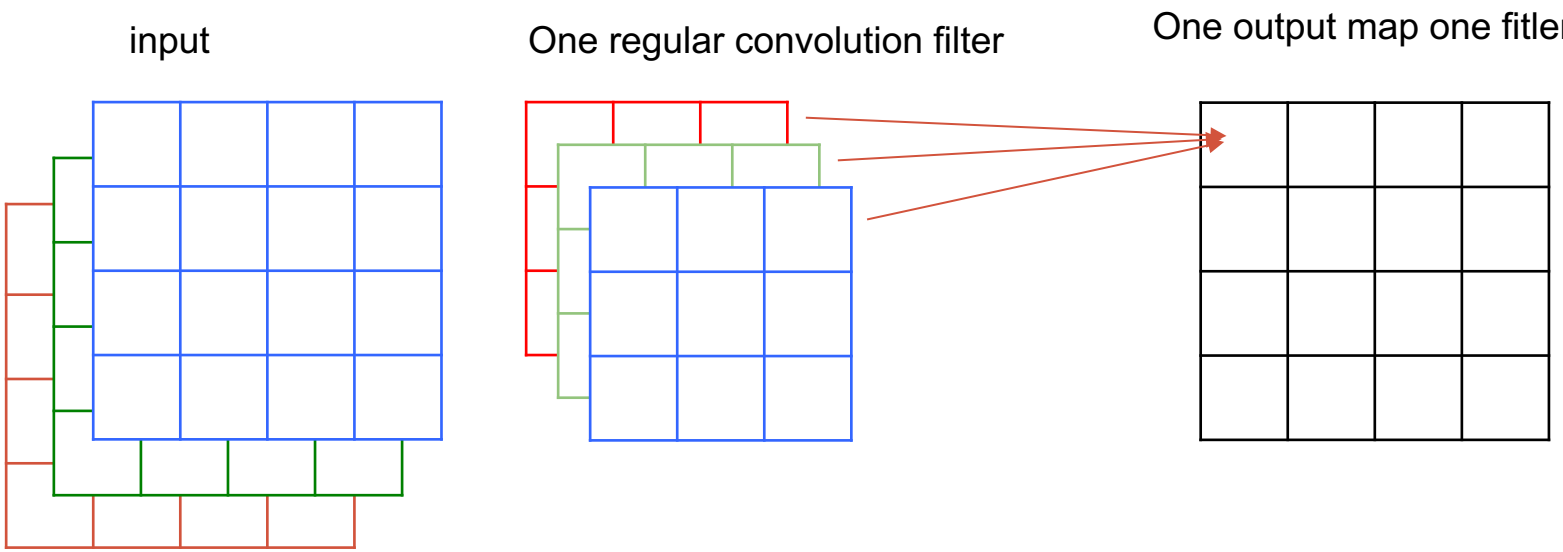
Xception

Depthwise separable convolution: two-step convolution

1. Depthwise convolution
2. 1x1 convolution

Typical convolution 3x3 filter is 3x3xinput channel

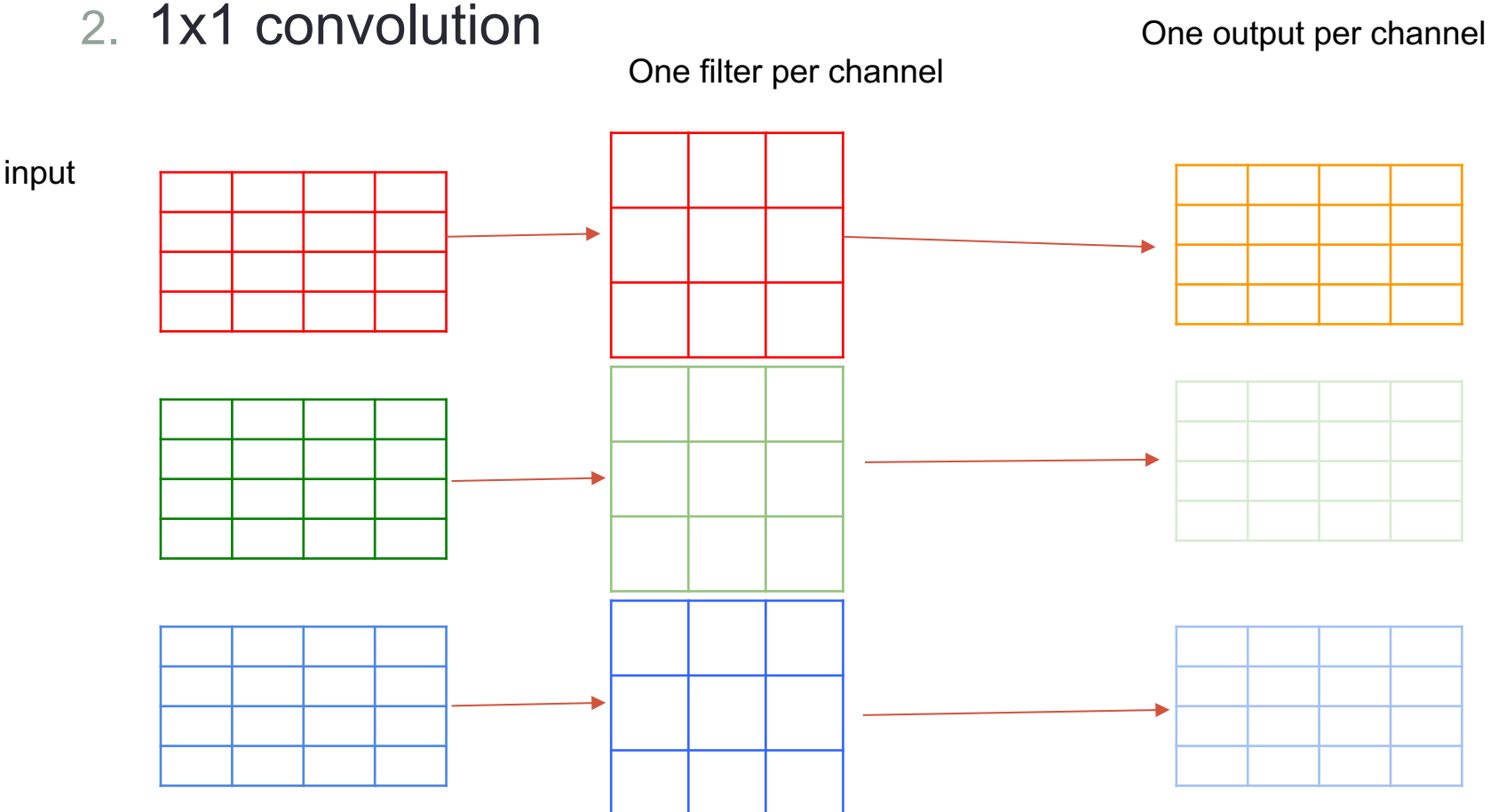
Depthwise convolution 3x3 filter is 3x3x1
1 filter per input channel



Xception

Depthwise separable convolution: two-step convolution

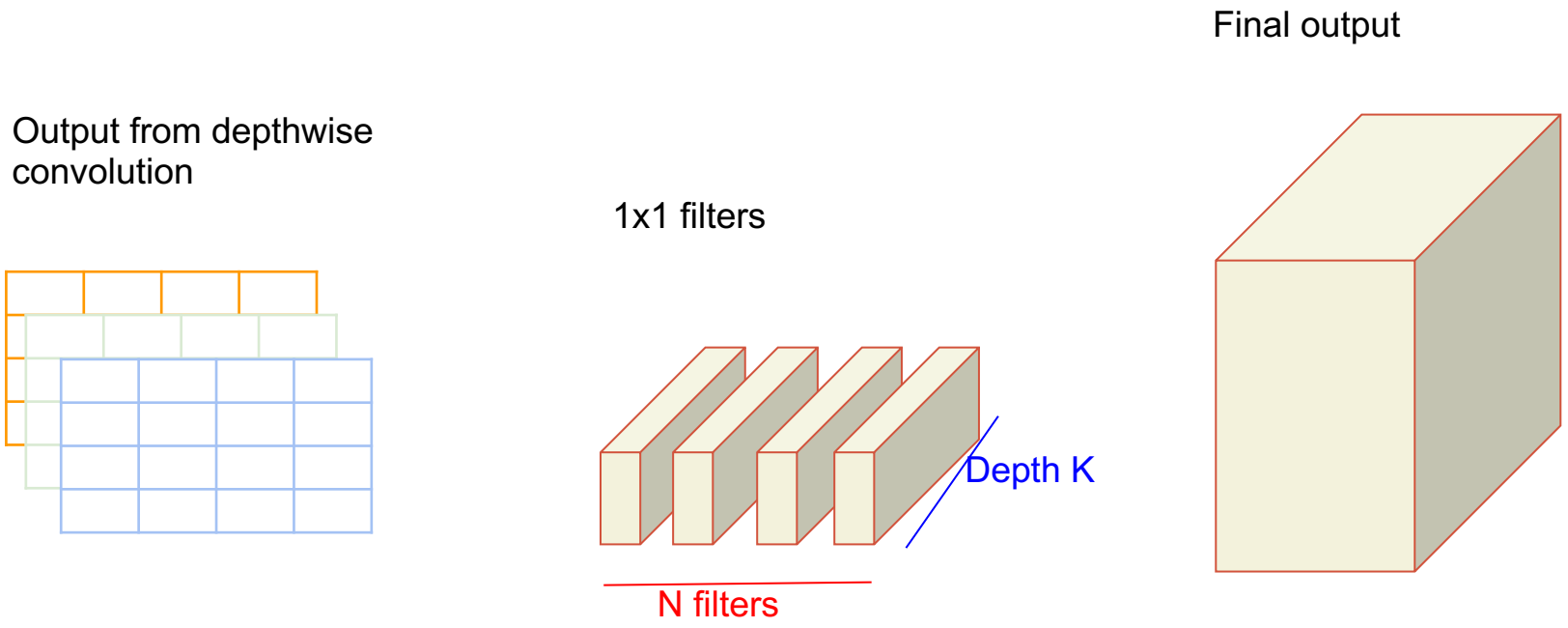
1. Depthwise convolution
2. 1x1 convolution



Xception

Depthwise separable convolution: two-step convolution

1. Depthwise convolution
2. 1x1 convolution



Xception

Replace convolutions in inception with depthwise separable convolutions

Smaller model

Faster compute

Comparable with Inception v3 while much faster

Used in MobileNet and other models

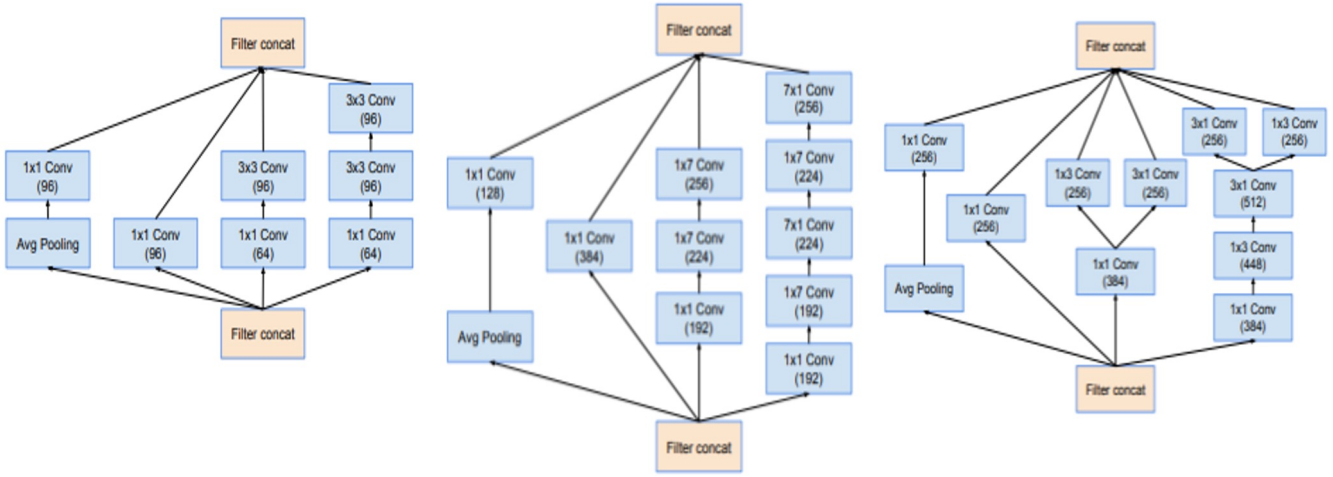
François Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions” 2016.
Andrew G. Howard, et al., “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications” 2017.

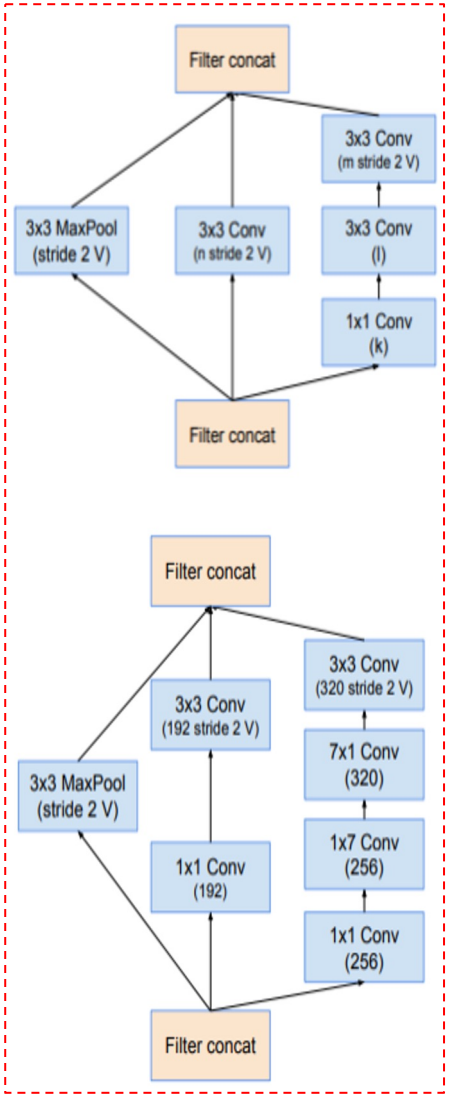
Inception v4

Same three types of inception blocks from v3

Add two types of reduction blocks for reducing the size of the grid (super pooling blocks)

Inception blocks





Reduction blocks

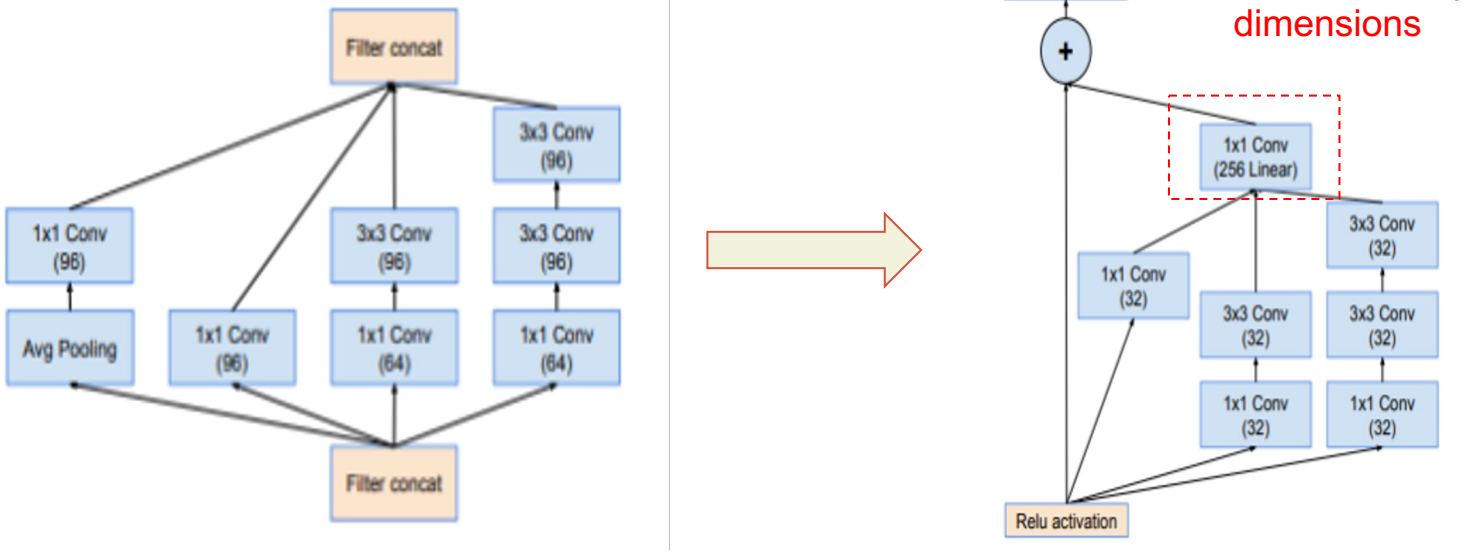
Christian Szegedy, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning” 2016.

Inception-ResNet v1-v2

Introduce residual connections into inception blocks

Poolings changed to additions, 1x1 added to keep dimensions for the residual

Similar idea to ResNeXt



Christian Szegedy, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning” 2016.

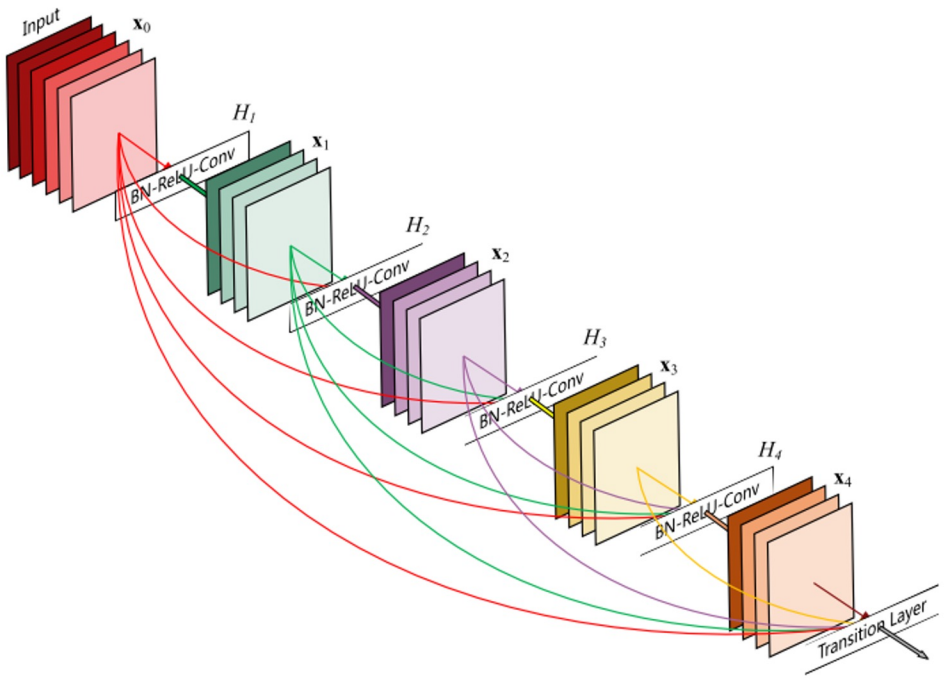
DenseNet

Instead of adding the residuals, concatenates the feature maps from previous layers

Densely connect multiple layers

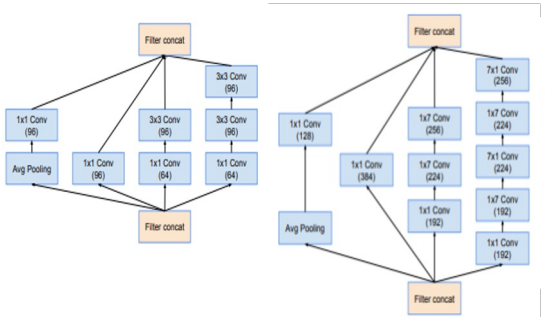
Multi-scale feature maps

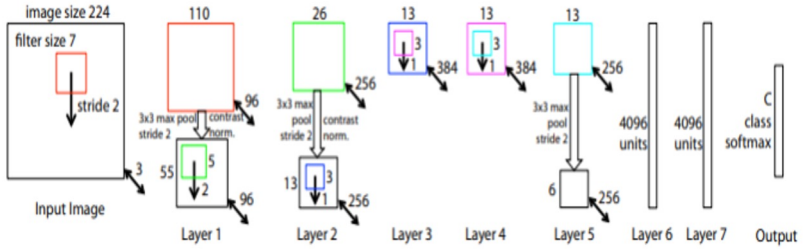
Smaller model due to smaller number of channels

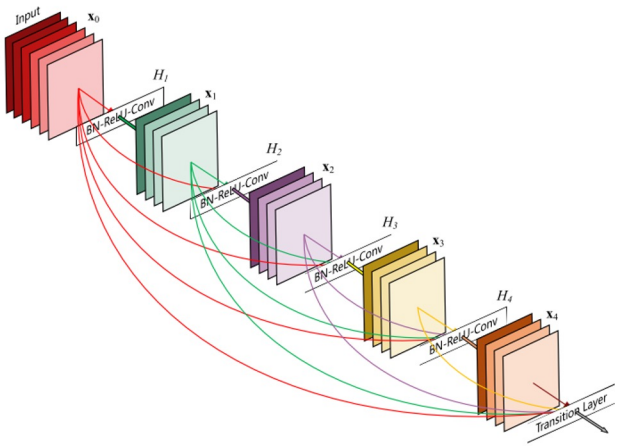


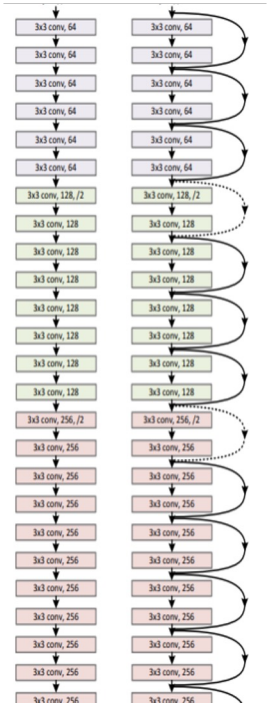
Densely Connected Convolutional Networks 2016

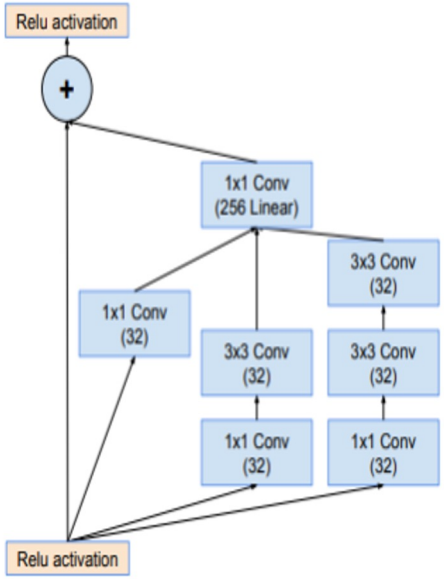
Do people keep tweaking network architectures until the end of time?











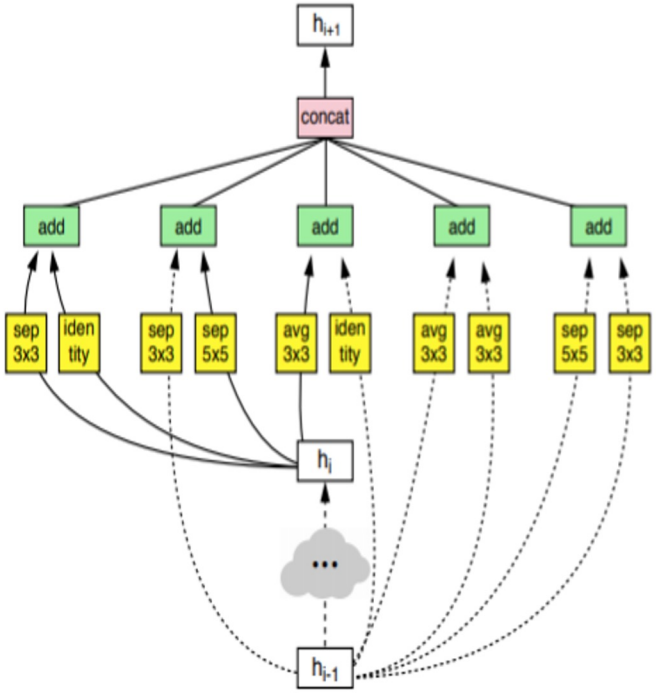
NasNet

Neural Architecture Search Network

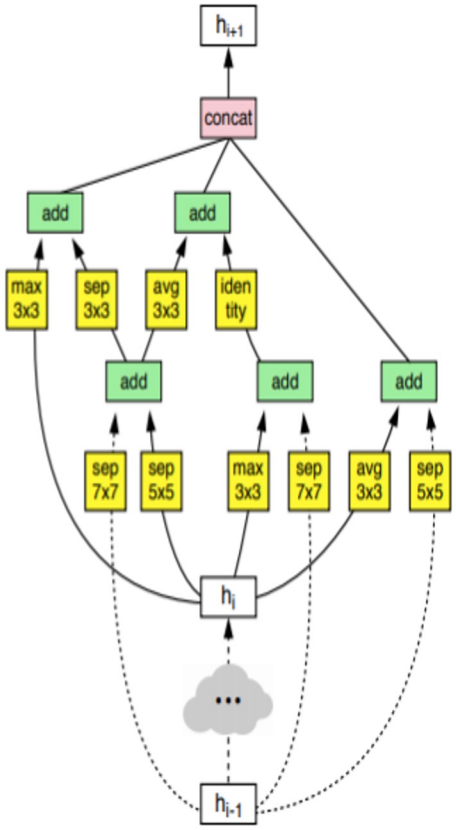
Use reinforcement learning to search for the best network configuration

Lowers compute while maintaining high accuracy

Precursor of EfficientNet



Inception block



Reduction block

Barret Zoph, et al. “Learning Transferable Architectures for Scalable Image Recognition,” 2017.

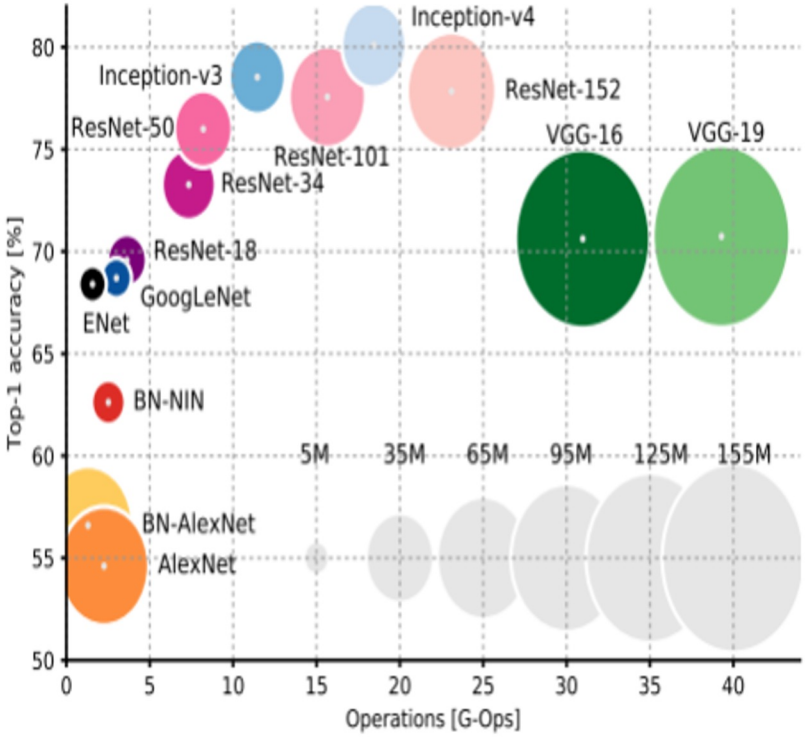
Object classifiers summary

Most successful tricks:

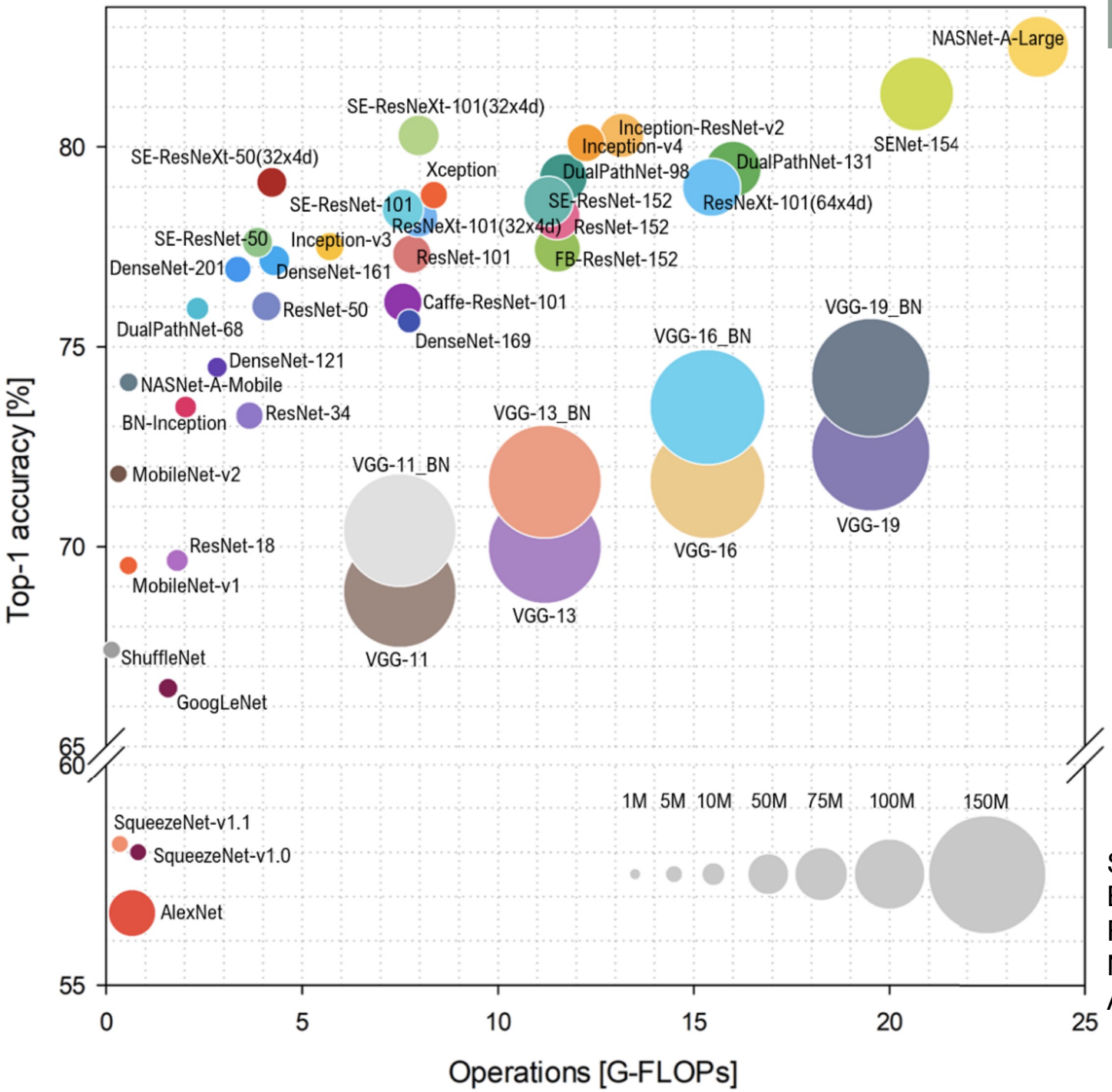
Dropout, residual, batch norms, multi-path, data augmentation

These object classifiers are often called backbones and used by other models

Comparing accuracy, model size, transferability and compute.



Alfredo Canziani, et al. “An analysis of deep neural network models for practical applications,” 2017



Simone Bianco, et al., Benchmark Analysis of Representative Deep Neural Network Architectures, 2019

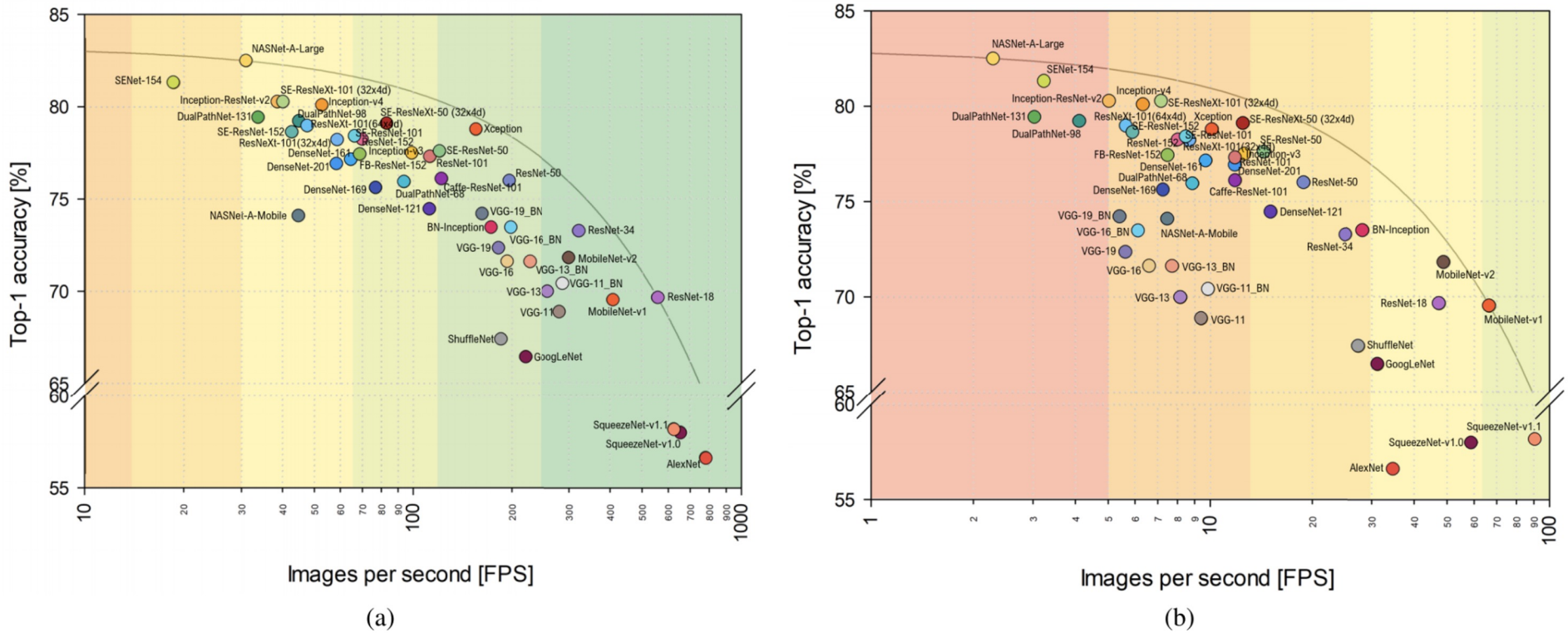


FIGURE 3. **Top-1 accuracy *vs.* number of images processed per second (with batch size 1) using the Titan Xp (a) and Jetson TX1 (b).**

Simone Bianco, et al., Benchmark Analysis of Representative Deep Neural Network Architectures, 2019

Other stuff to look for

Layers

Dilated convolution https://towardsdatascience.com/understanding-2d-dilated-convolution-operation-with-examples-in-numpy-and-tensorflow-with-d376b3972b25

Mobile inverted BottleNeck (inverted linear residual) https://arxiv.org/abs/1801.04381

Squeeze-and-excitation block https://arxiv.org/abs/1709.01507

Backbones

Feature Pyramid Networks (multi-scale backbone) https://arxiv.org/abs/1612.03144

SqueezeNet (small network) https://arxiv.org/abs/1602.07360

EfficientNetv2 https://arxiv.org/abs/2104.00298

Vision Transformers https://arxiv.org/abs/2010.11929

Transformers

- A model based on attention modeling.
- Used in almost every new deep learning architecture
- Key to ChatGPT, Alphafold, latest version of stablediffusion, etc.

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

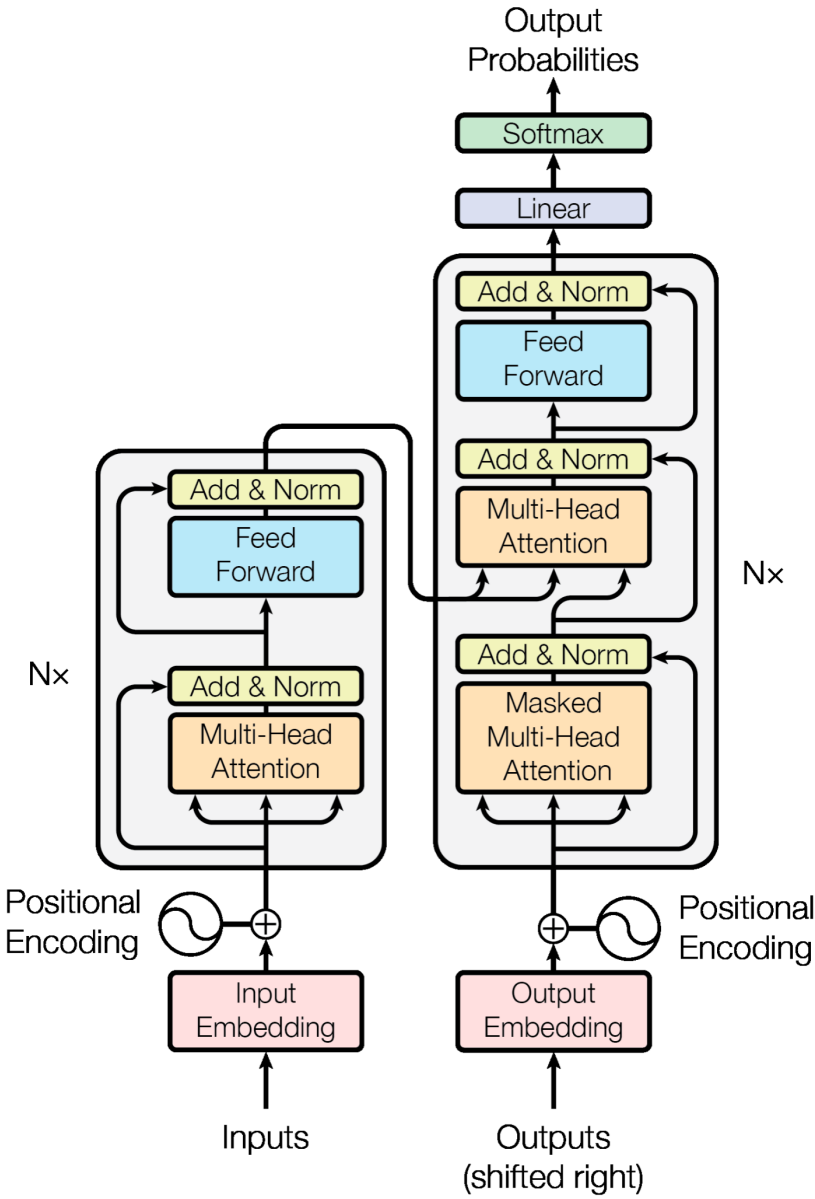
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.



CNN Summary

Building blocks

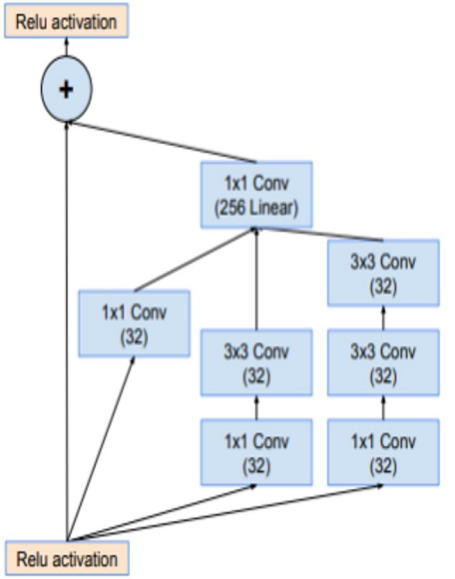
Residual connection

Filter factorization

5x5 -> two 3x3

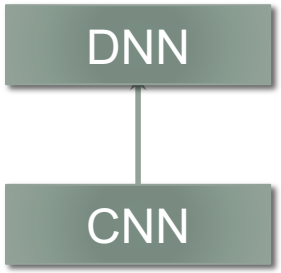
depthwise vs matrix factorization

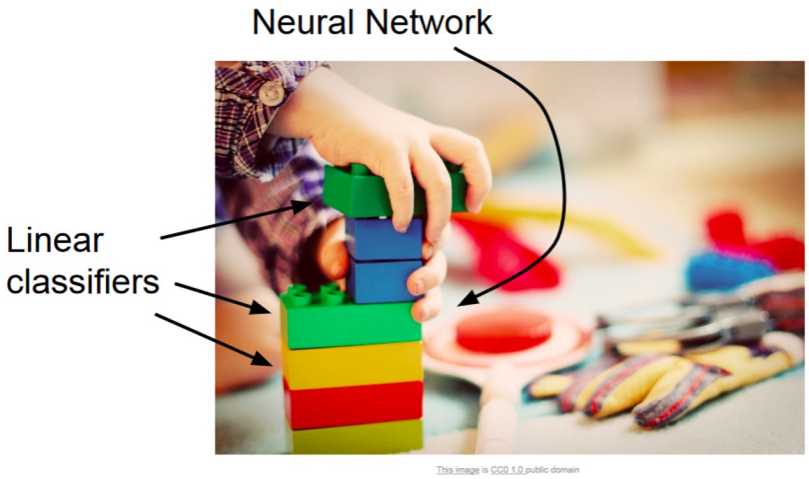
1x1 convolution



DNN Legos

- Typical models usually has multiple layer types
  - CNN: local structure in the feature. Used for feature learning.
  - DNN: Good for mapping features for classification. Usually used in final layers





https://www.analyticsvidhya.com/blog/2017/08/10-advanced-deep-learning-architectures-data-scientists/

Transfer learning

Have you ever seen this creature before?
Can you guess whether it is land or water animal?

You can transfer your knowledge in the past

Transfer learning

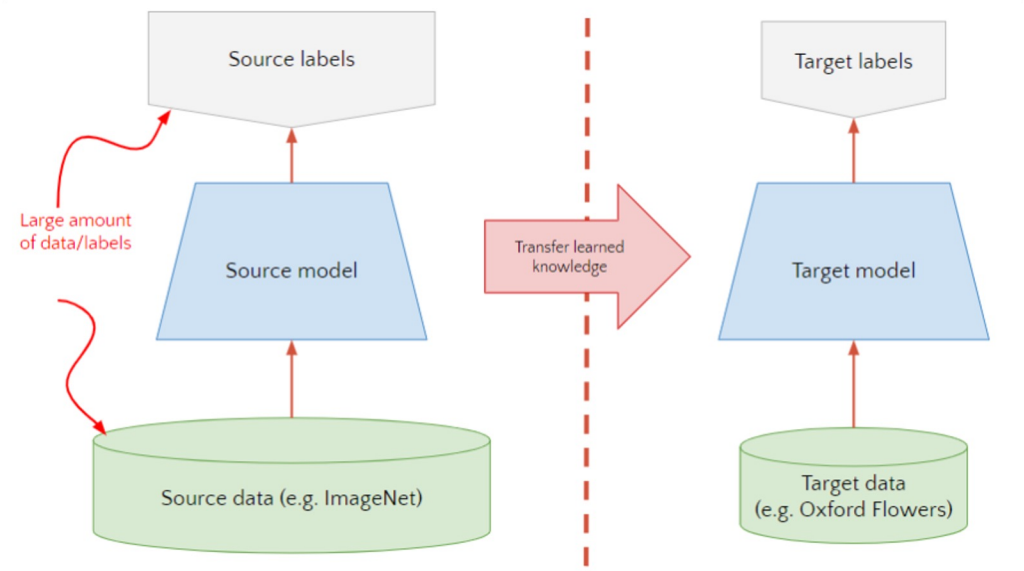
Myth: you can't do deep learning unless you have a million labelled examples for your problem.

Reality:

- *You can **transfer** learned representations from **a related task***
- You can train on a nearby **surrogate objective** for which it is easy to generate labels

Transfer learning (basics)

- We know networks captures good representations
- Can we use it for other tasks?
- Use trained networks to initialize a new network for a different task.
- Re-train the network using SGD on new data.



For CV tasks, we call the pre-trained network backbones

Transfer learning idea

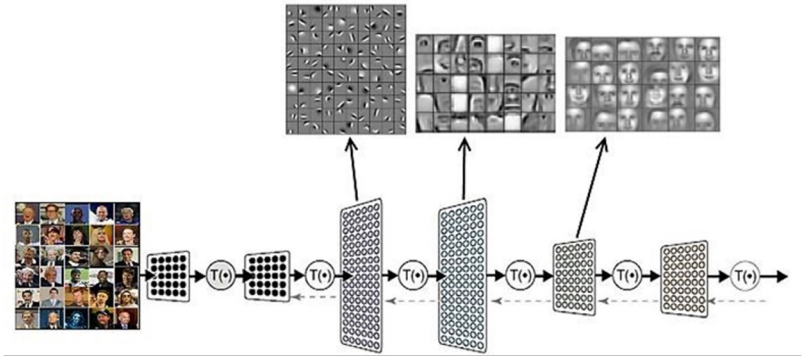
Instead of training a deep network from scratch for your task, you can

- Take a network trained on **a different domain** for a **different source task**
- **Adapt (fine-tune)** it for your domain and your **target task**

This lecture will talk about how to do this.

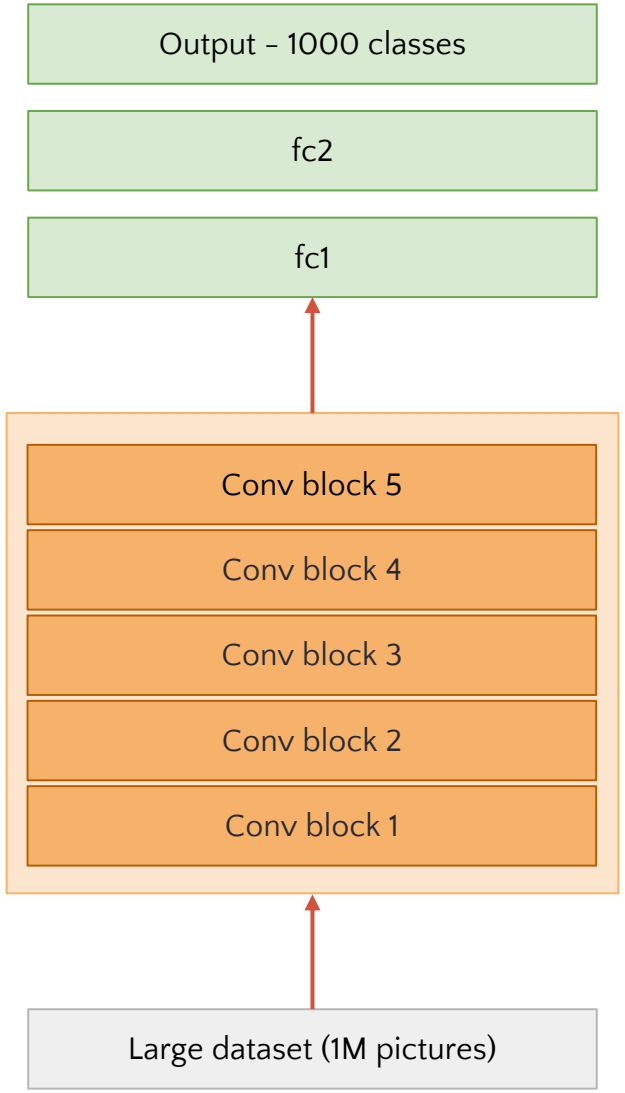
Variations:

- Different domain, same task
- Different domain, different task

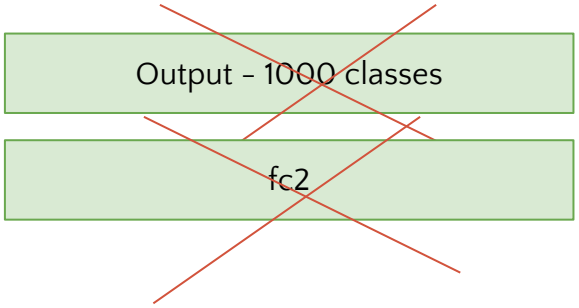


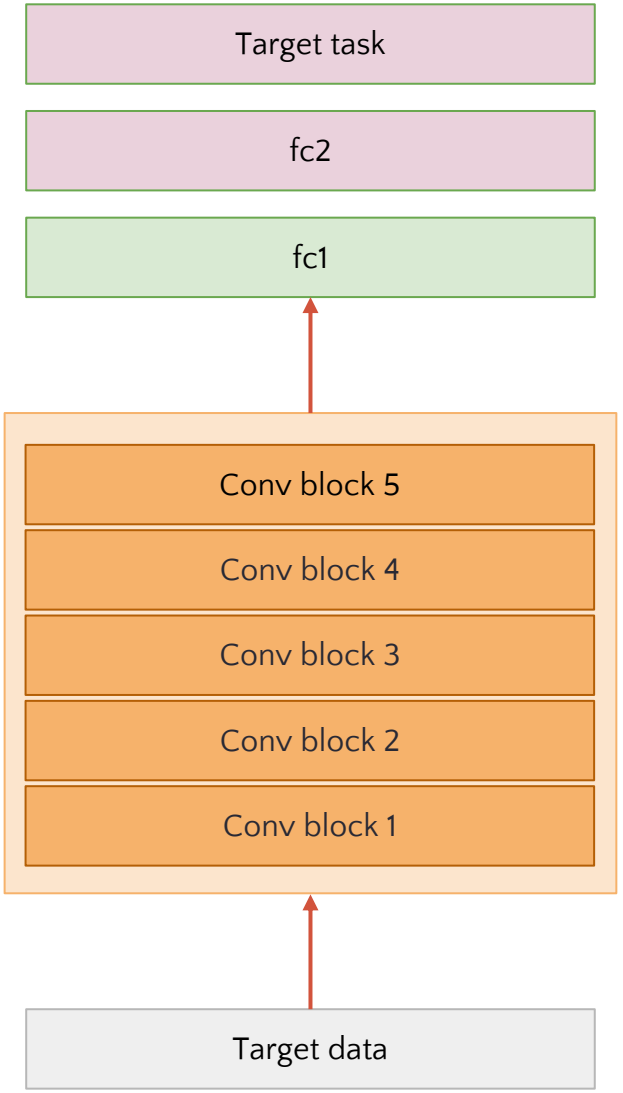
Transfer learning

1. A model is trained on large dataset
   Ex ImageNet



Transfer learning

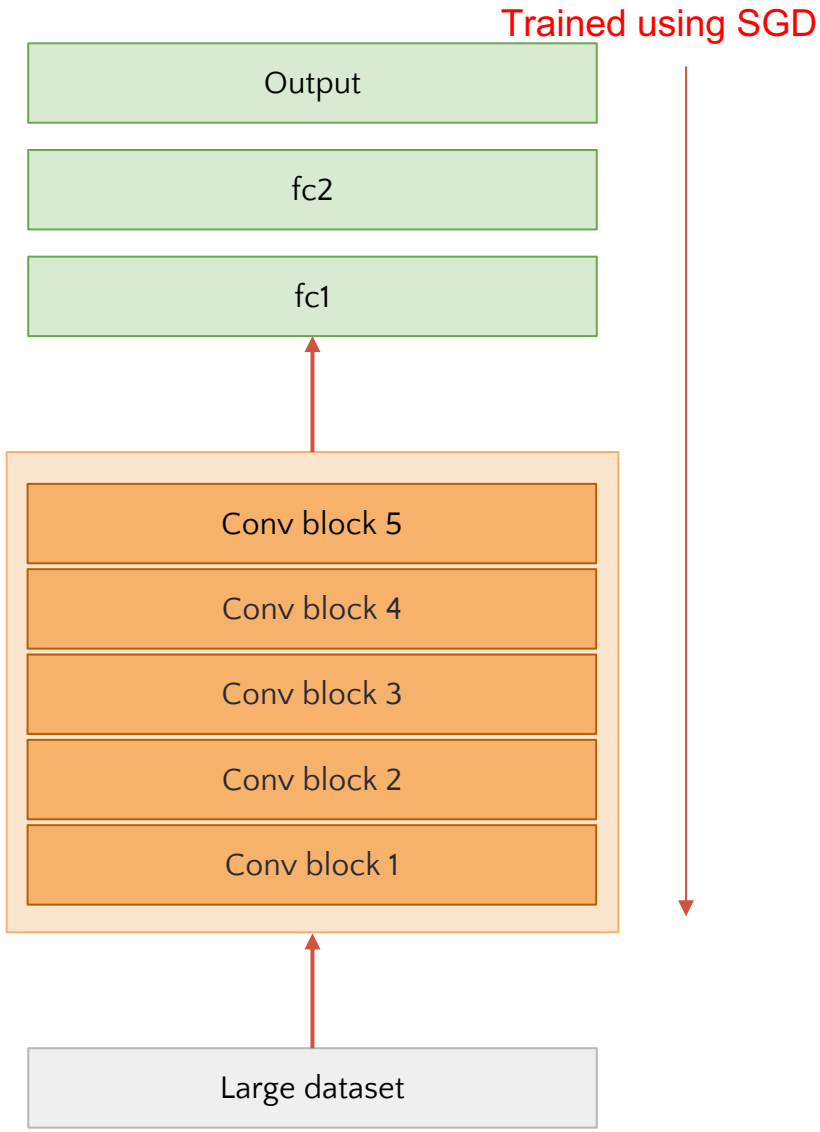




2. Replace the top layers with target task

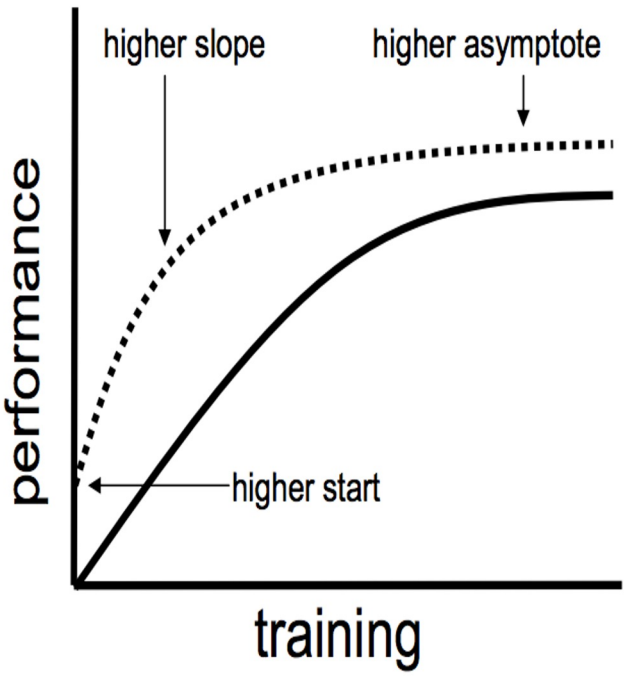
Transfer learning

3. Train all layers (unfreezed weights)



Benefits to transfer learning

1. Higher start
2. Higher slope (converge faster)
3. Higher asymptote (if small data)



Chapter 11: Transfer Learning, Handbook of Research on Machine Learning Applications, 2009.

Transfer learning today (1)

- With larger models (billions of parameters), we need more efficient techniques such as LoRA https://arxiv.org/abs/2106.09685
- Or some kind of regularization so that the model does not degrade on small datasets

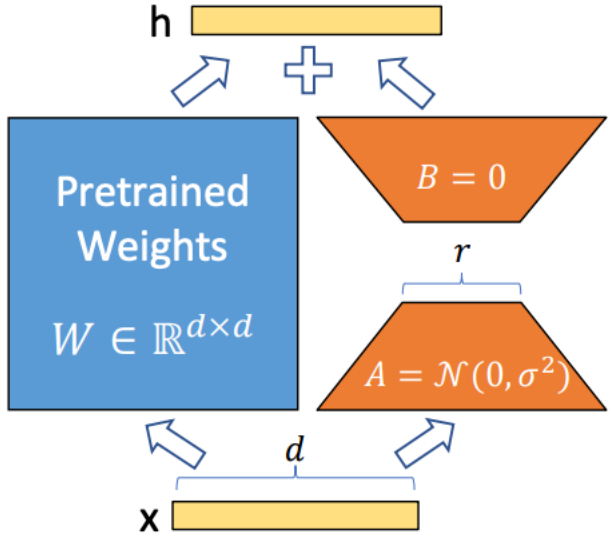
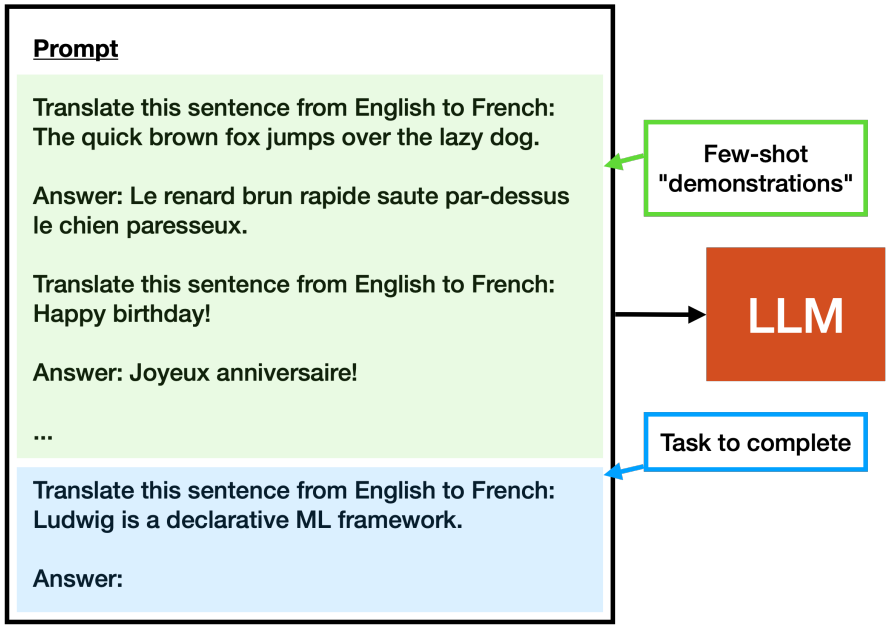


Figure 1: Our reparametrization. We only train A and B.

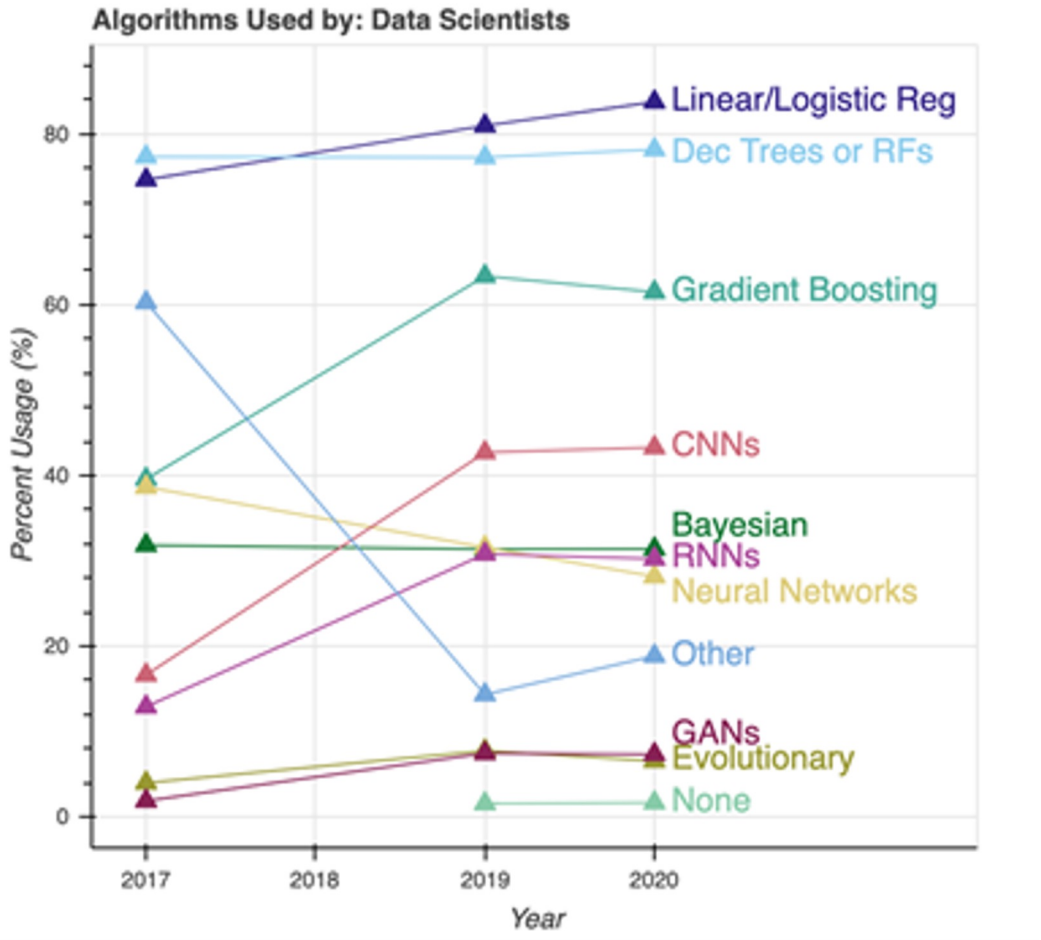
Transfer learning today (2)

- With the rise of LLMs, in-context learning has become more common.
- No need to finetune the model. User can prompt with examples.
  - Some argue in-context learning has the same effect as finetuning (https://openreview.net/forum?id=fzbHRjAd8U)



When to deep learning?

What's to use?



https://medium.com/analytics-vidhya/ongoing-kaggle-survey-picks-the-topmost-data-science-trends-7c19ec7606a1

What's the best?

We evaluate **179 classifiers** arising from **17 families** (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalized linear models, nearest-neighbors, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods), implemented in Weka, R (with and without the caret package), C and Matlab, including all the relevant classifiers available today. We use **121 data sets**, which represent **the whole UCI** data base (excluding the large-scale problems) and other own real problems, in order to achieve significant conclusions about the classifier behavior, not dependent on the data set collection. **The classifiers most likely to be the bests are the random forest** (RF) versions, the best of which (implemented in R and accessed via caret) achieves 94.1% of the maximum accuracy overcoming 90% in the 84.3% of the data sets. However, the difference is not statistically significant with the second best, the SVM with Gaussian kernel implemented in C using LibSVM, which achieves 92.3% of the maximum accuracy. A few models are clearly better than the remaining ones: random forest, SVM with Gaussian and polynomial kernels, extreme learning machine with Gaussian kernel, C5.0 and avNNet (a committee of multi-layer perceptrons implemented in R with the caret package). The random forest is clearly the best family of classifiers (3 out of 5 bests classifiers are RF), followed by SVM (4 classifiers in the top-10), neural networks and boosting ensembles (5 and 3 members in the top-20, respectively).

Most data sets are small (<1000) in this paper

http://jmlr.org/papers/volume15/delgado14a/delgado14a.pdf

Revisiting Deep Learning Models for Tabular Data

Yury Gorishniy[*†‡] **Ivan Rubachev**[†♣] **Valentin Khrulkov**[†] **Artem Babenko**[†♣]

† Yandex, Russia
‡ Moscow Institute of Physics and Technology, Russia
♣ National Research University Higher School of Economics, Russia

Abstract

The existing literature on deep learning for tabular data proposes a wide range of novel architectures and reports competitive results on various datasets. However, the proposed models are usually not properly compared to each other and existing works often use different benchmarks and experiment protocols. As a result, it is unclear for both researchers and practitioners what models perform best. Additionally, the field still lacks effective baselines, that is, the easy-to-use models that provide competitive performance across different problems.

In this work, we perform an overview of the main families of DL architectures for tabular data and raise the bar of baselines in tabular DL by identifying two simple and powerful deep architectures. The first one is a ResNet-like architecture which turns out to be a strong baseline that is often missing in prior works. The second model is our simple adaptation of the Transformer architecture for tabular data, which outperforms other solutions on most tasks. Both models are compared to many existing architectures on a diverse set of tasks under the same training and tuning protocols. We also compare the best DL models with Gradient Boosted Decision Trees and conclude that there is still no universally superior solution. The source code is available at https://github.com/yandex-research/rtdl.

https://arxiv.org/abs/2106.11959

Table 1: Dataset properties. Notation: "RMSE" ~ root-mean-square error, "Acc." ~ accuracy.

	CA	AD	HE	JA	HI	AL	EP	YE	CO	YA	MI
#objects	20640	48842	65196	83733	98050	108000	500000	515345	581012	709877	1200192
#num. features	8	6	27	54	28	128	2000	90	54	699	136
#cat. features	0	8	0	0	0	0	0	0	0	0	0
metric	RMSE	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	RMSE	Acc.	RMSE	RMSE
#classes	–	2	100	4	2	1000	2	–	7	–	–

Table 4: Results for ensembles of GBDT and the main DL models. For each model-dataset pair, the metric value averaged over three ensembles is reported. See supplementary for standard deviations. Notation follows Table 3.

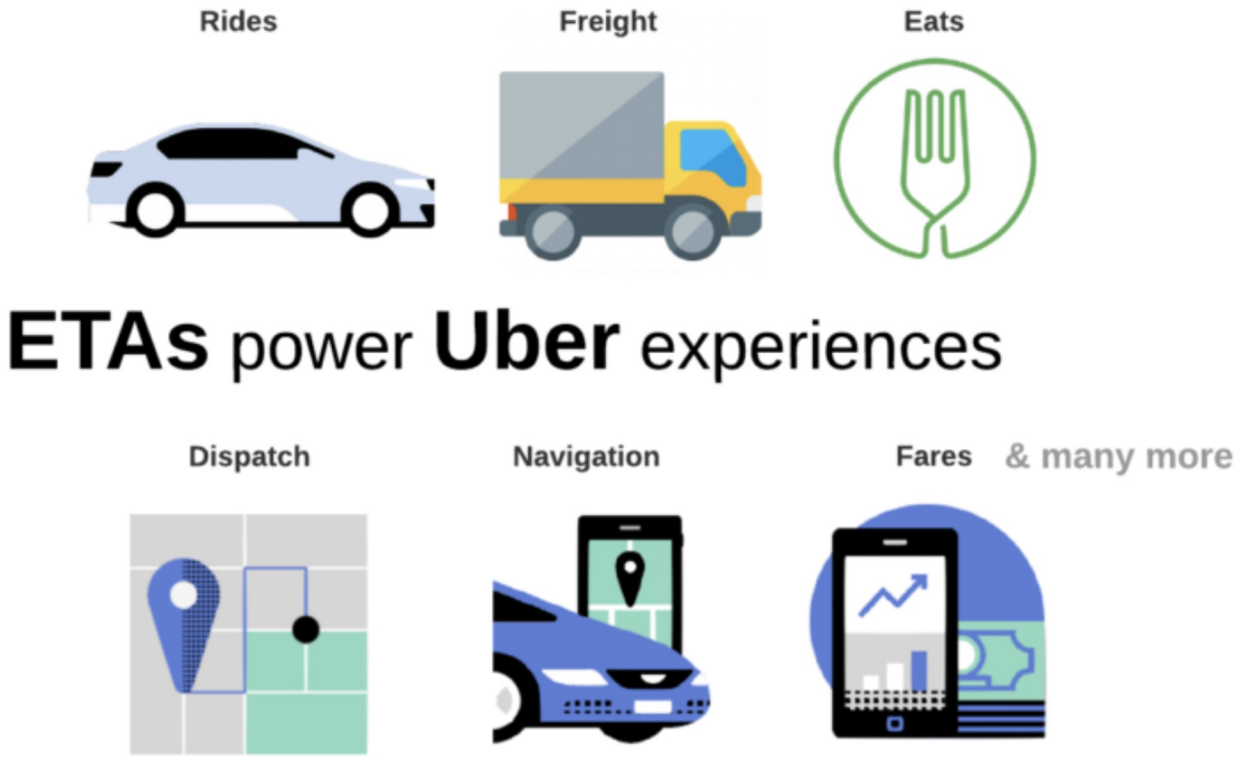
	CA ↓	AD ↑	HE ↑	JA ↑	HI ↑	AL ↑	EP ↑	YE ↓	CO ↑	YA ↓	MI ↓
Default hyperparameters											
XGBoost	0.462	**0.874**	0.348	0.711	0.717	0.924	0.8799	9.192	0.964	0.761	0.751
CatBoost	**0.428**	0.873	0.386	0.724	0.728	0.948	0.8893	8.885	0.910	0.749	0.744
FT-Transformer	0.454	0.860	**0.395**	**0.734**	**0.731**	**0.966**	**0.8969**	**8.727**	**0.973**	**0.747**	**0.742**
Tuned hyperparameters											
XGBoost	0.431	0.872	0.377	0.724	0.728	–	0.8861	8.819	0.969	**0.732**	0.742
CatBoost	**0.423**	**0.874**	0.388	0.727	0.729	–	0.8898	8.837	0.968	0.740	**0.741**
ResNet	0.478	0.857	0.398	0.734	0.731	0.966	0.8976	8.770	0.967	0.751	0.745
FT-Transformer	0.448	0.860	**0.398**	**0.739**	**0.731**	**0.967**	**0.8984**	**8.751**	**0.973**	0.747	0.743

DeepETA: How Uber Predicts Arrival Times Using Deep Learning

Xinyu Hu, Olcay Cirit, Tanmay Binaykiya, and Ramit Hora

February 10, 2022

0



https://eng.uber.com/deepeta-how-uber-predicts-arrival-times/

For several years, Uber used gradient-boosted decision tree ensembles to refine ETA predictions. The ETA model and its training dataset grew steadily larger with each release. To keep pace with this growth, Uber's Apache Spark™ team contributed upstream improvements [1, 2] to XGBoost to allow the model to grow ever deeper, making it one of the largest and deepest XGBoost ensembles in the world at that time. Eventually, we reached a point where increasing the dataset and model size using XGBoost became untenable. To continue scaling the model and improving accuracy, we decided to explore deep learning because of the relative ease of scaling to large datasets using data-parallel SGD [3]. To justify switching to deep learning we needed to overcome three main challenges:

- **Latency:** The model must return an ETA within a few milliseconds at most.
- **Accuracy:** The mean absolute error (MAE) must improve significantly over the incumbent XGBoost model.
- **Generality:** The model must provide ETA predictions globally across all of Uber's lines of business such as mobility and delivery.

Final exam

- I can only say about the neural network part
  - Theory only (no coding questions)
    - Key concepts: architectures, loss, training, regularization
  - Understand how to perform dense and CNN computations
    - Input and output size
    - Number of paramters
    - Receptive field