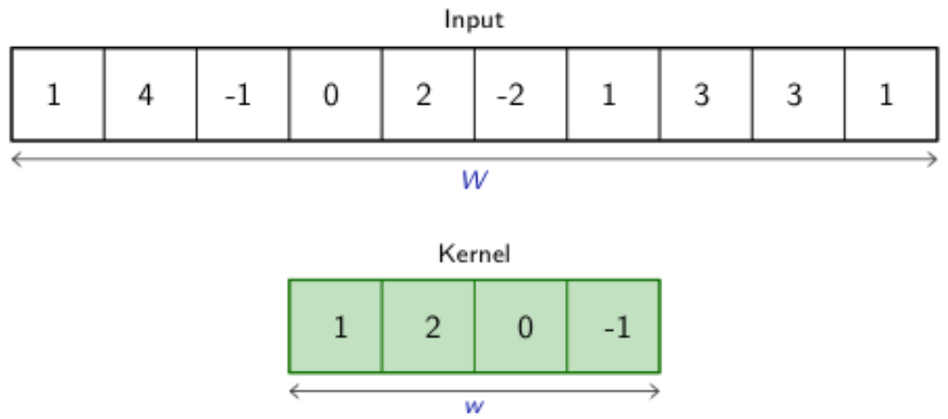
Convolutional neural networks & Modus Operandi of Deep Learning

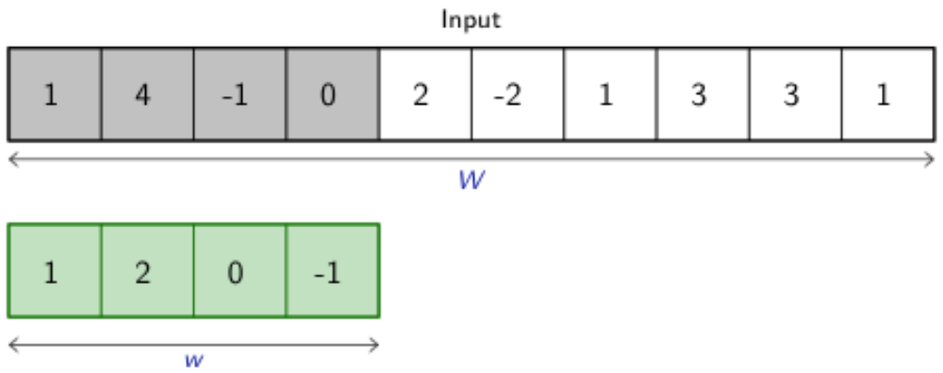
Lecture 13 of ML for physicists

Convolutional layers

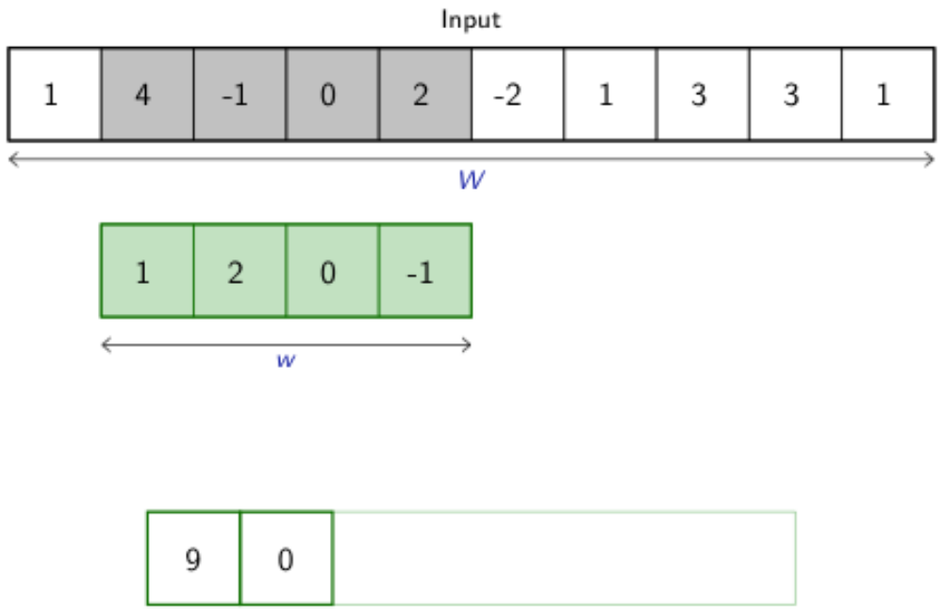
Convolution 1d



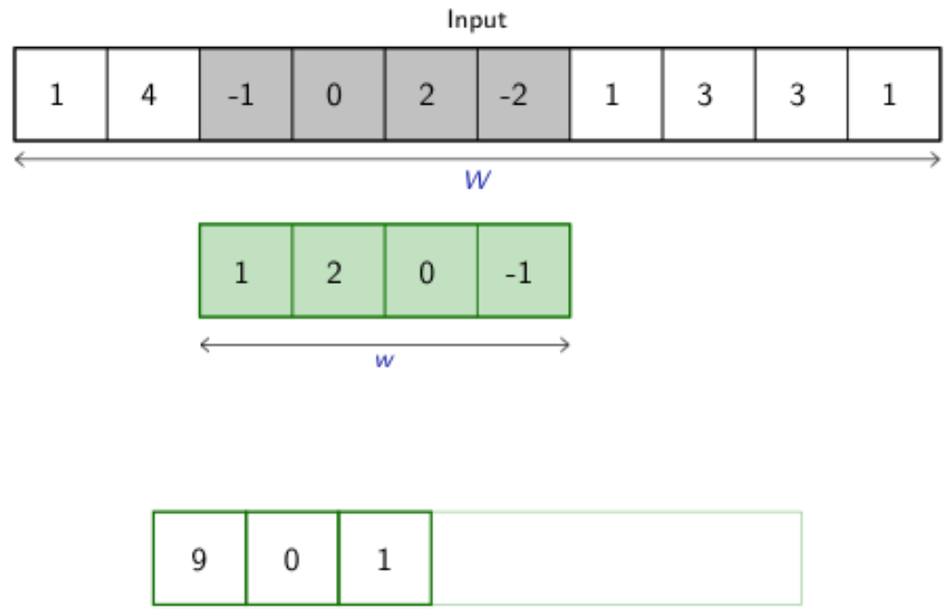
Convolution 1d



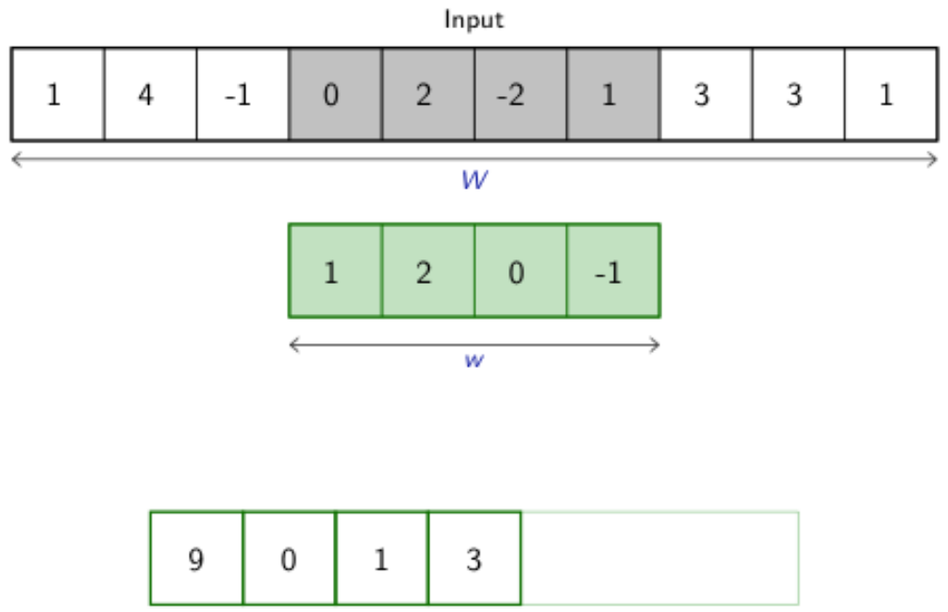
Convolution 1d



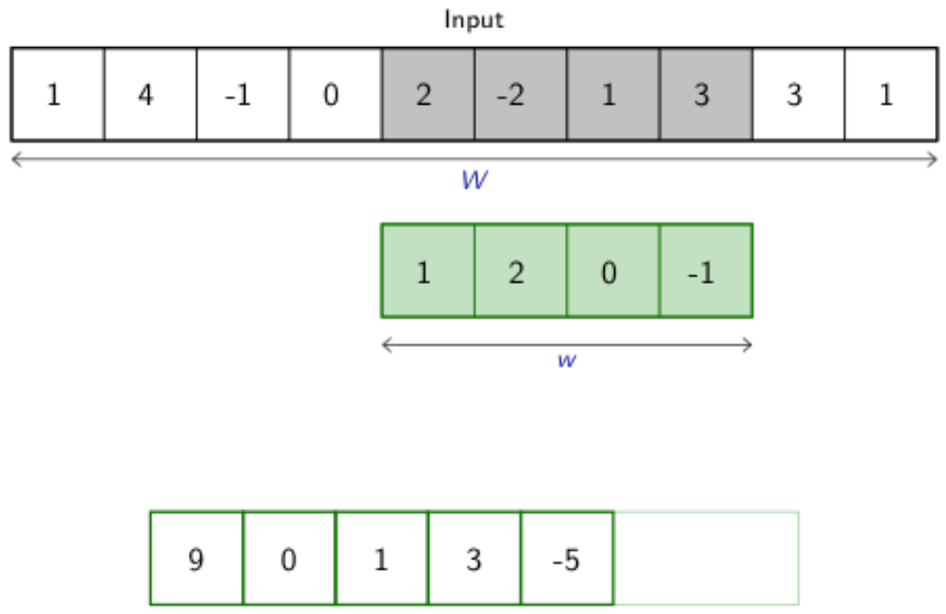
Convolution 1d



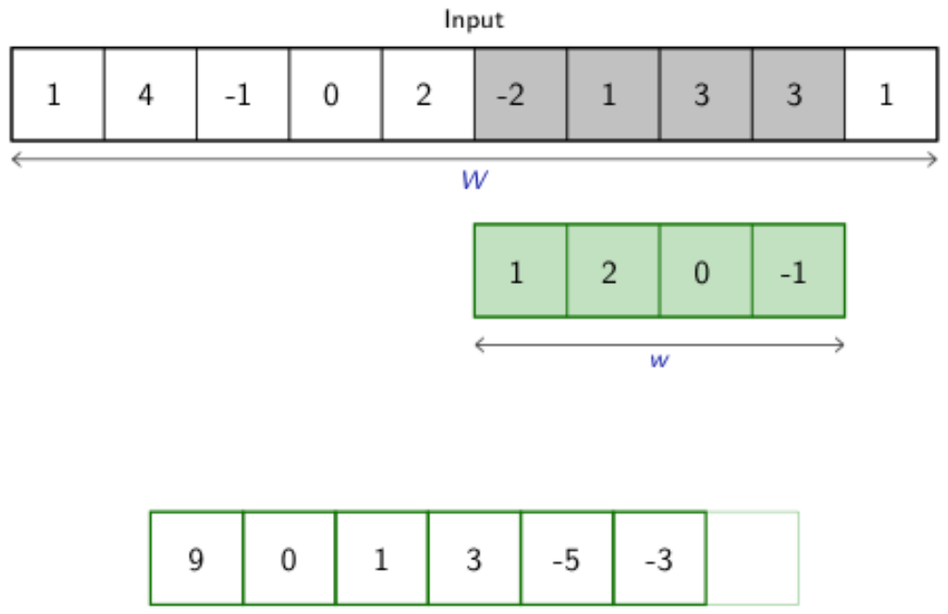
Convolution 1d



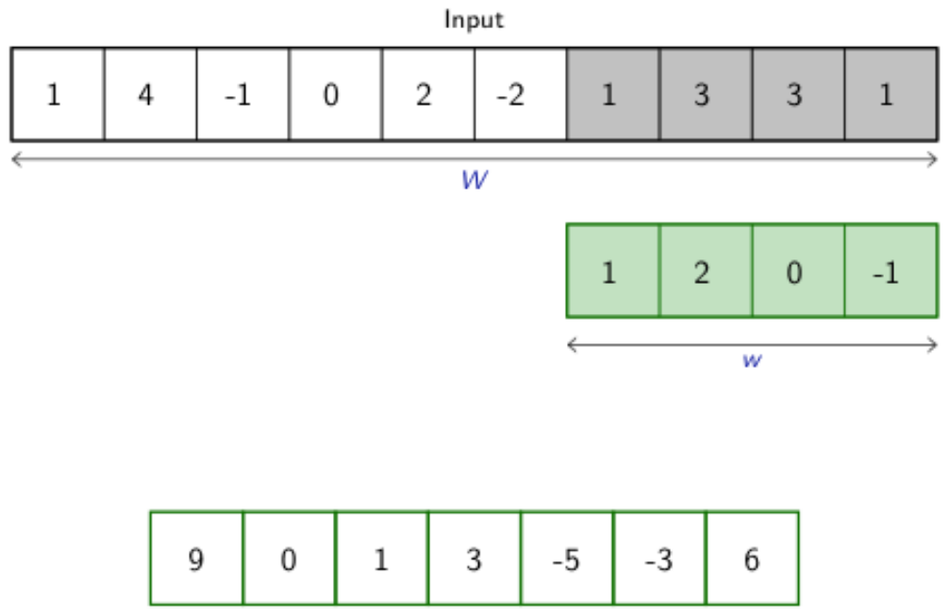
Convolution 1d



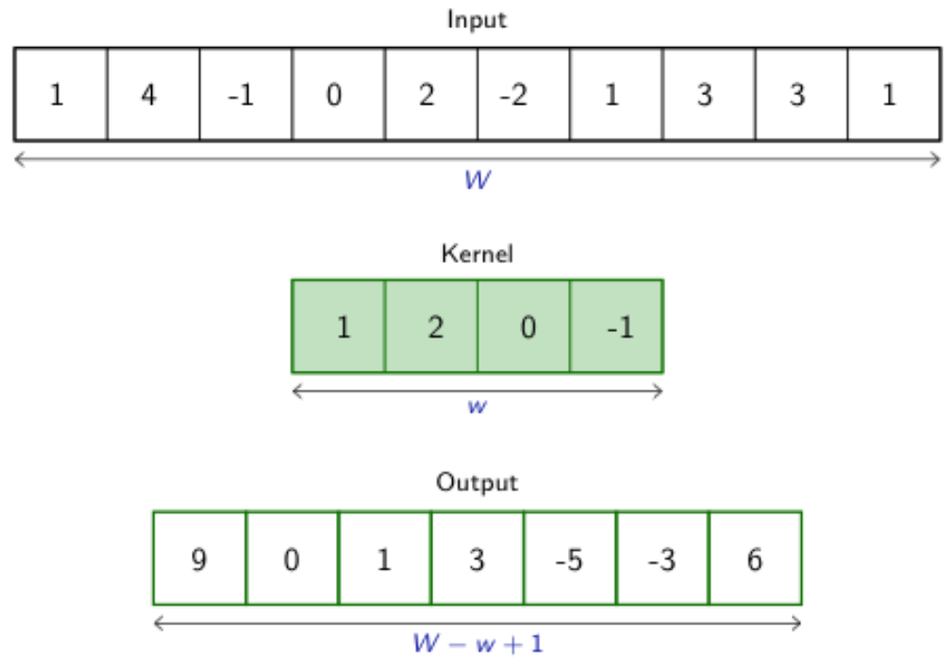
Convolution 1d



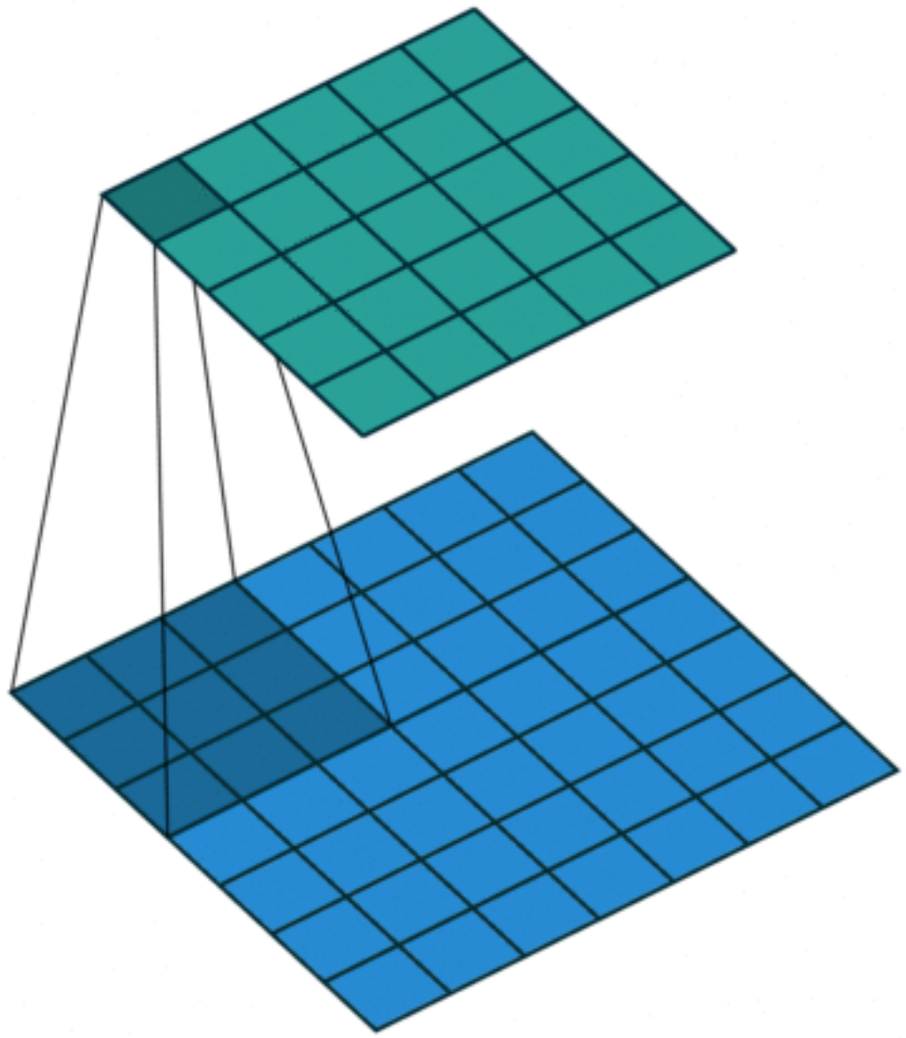
Convolution 1d



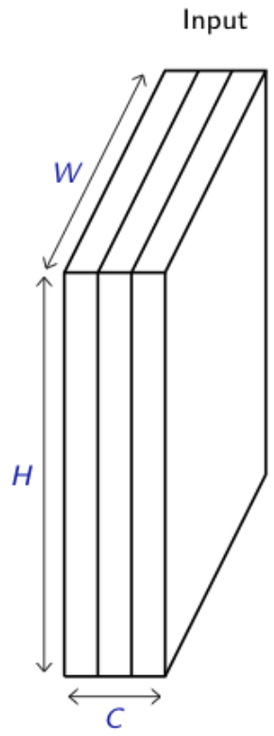
Convolution 1d



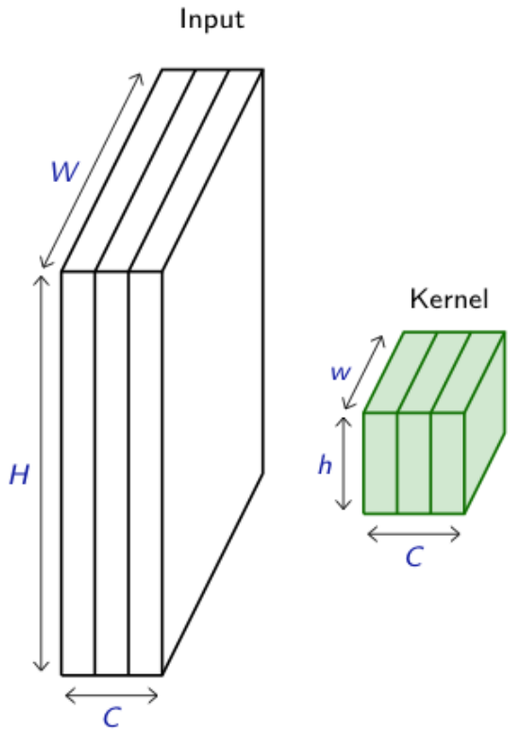
Convolutional filter



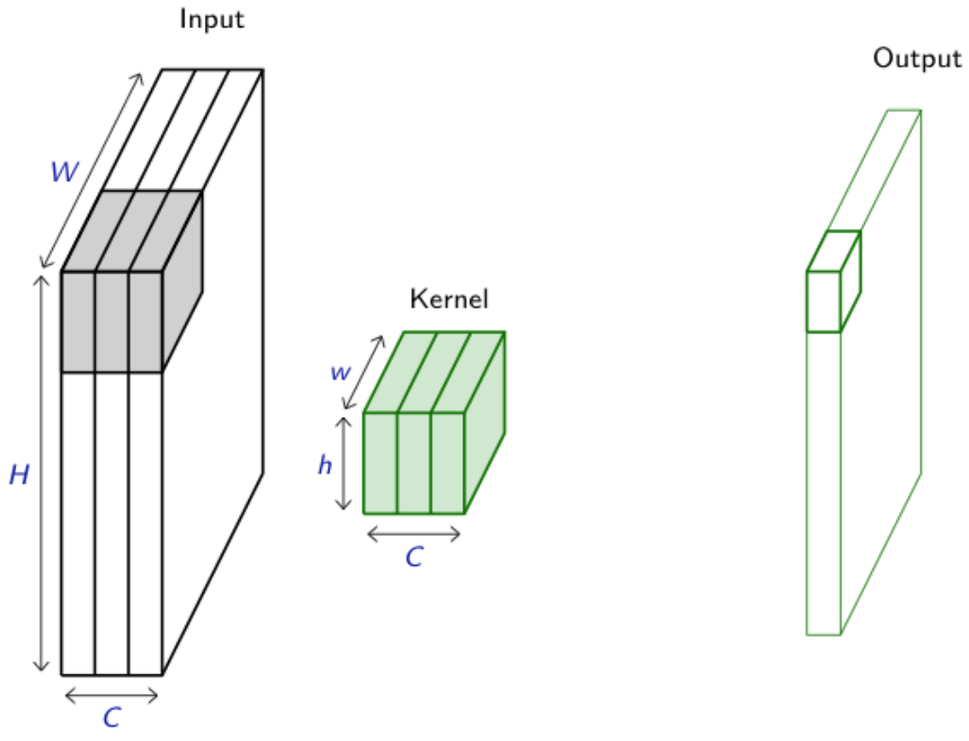
Convolution 2d



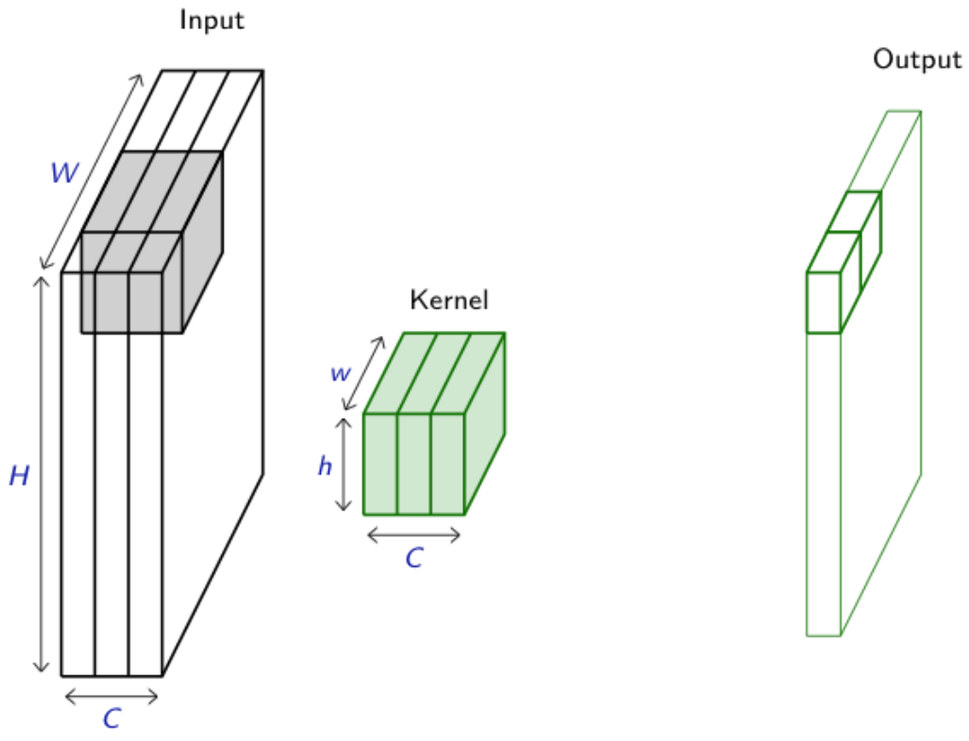
Convolution 2d



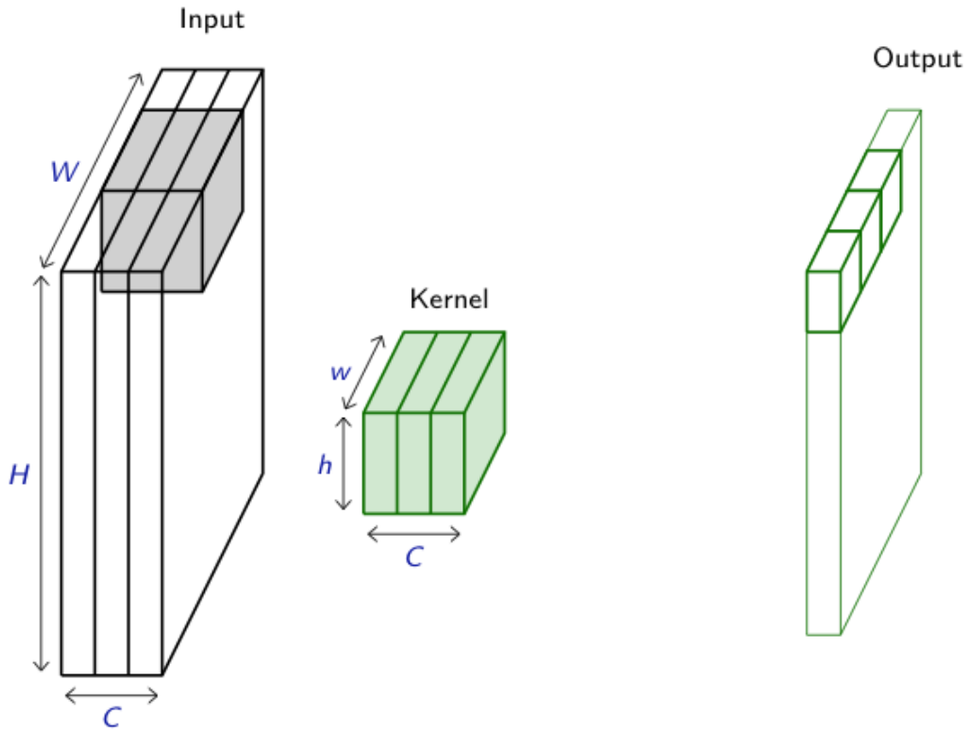
Convolution 2d



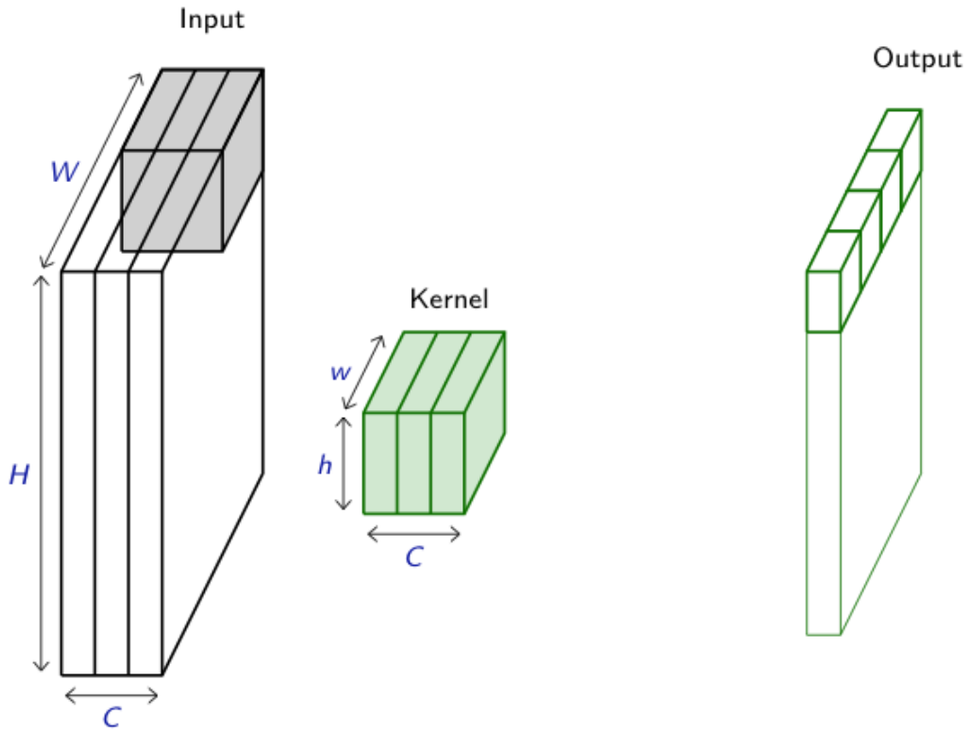
Convolution 2d



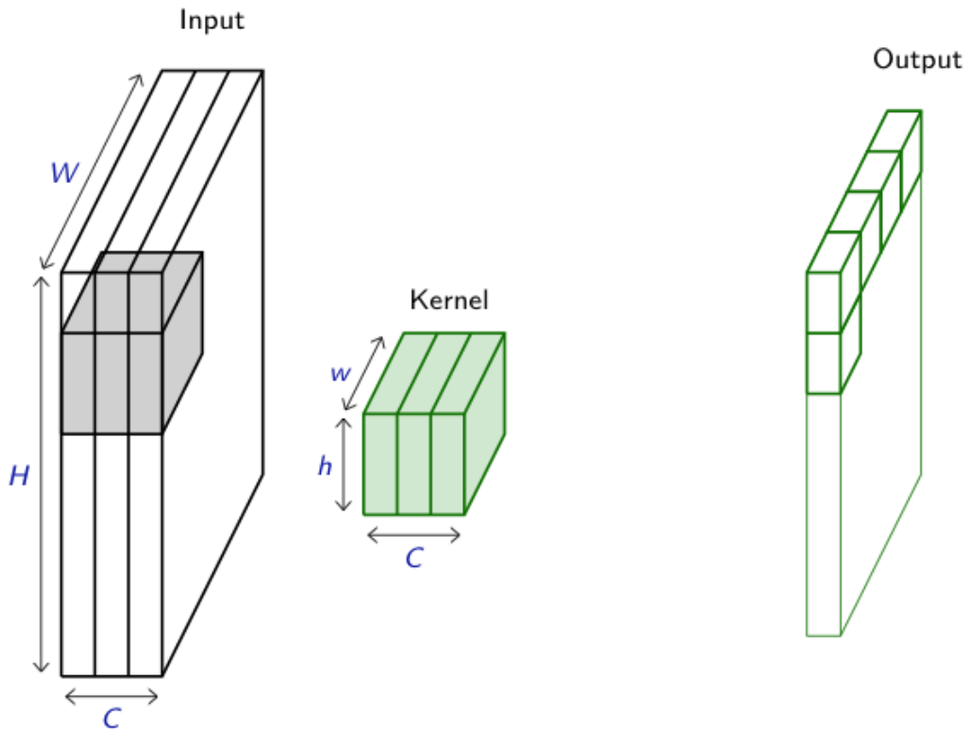
Convolution 2d



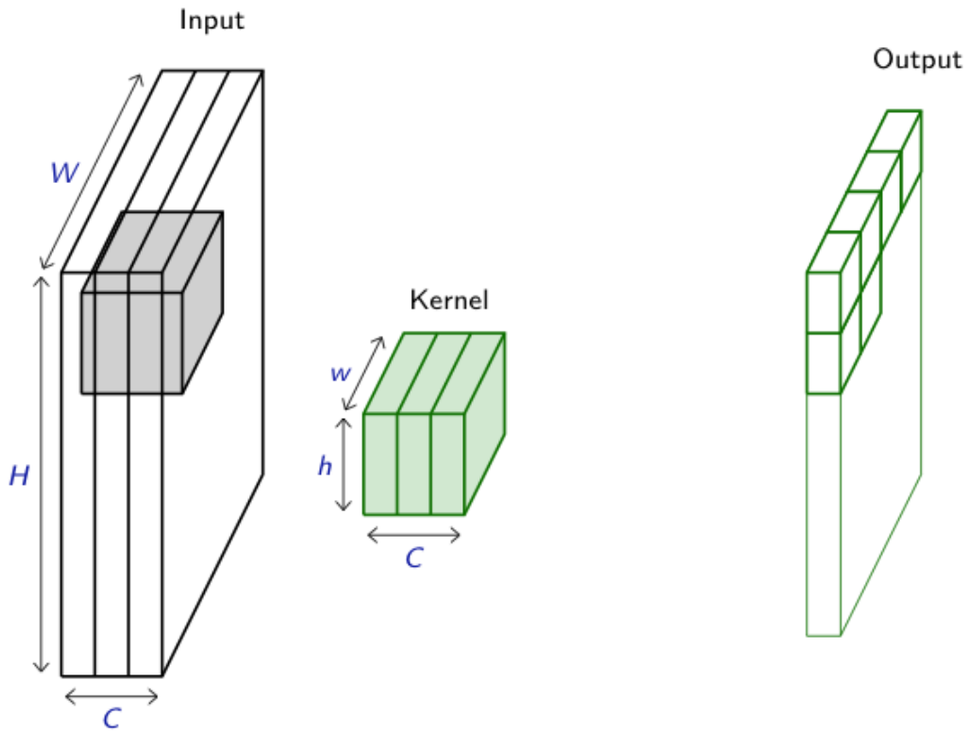
Convolution 2d



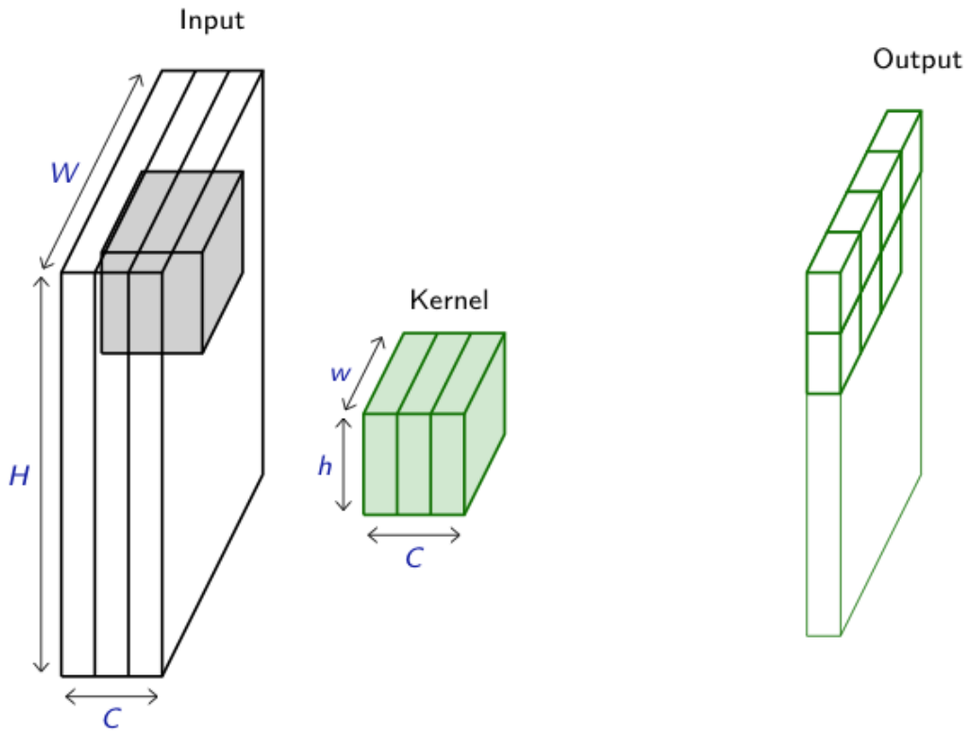
Convolution 2d



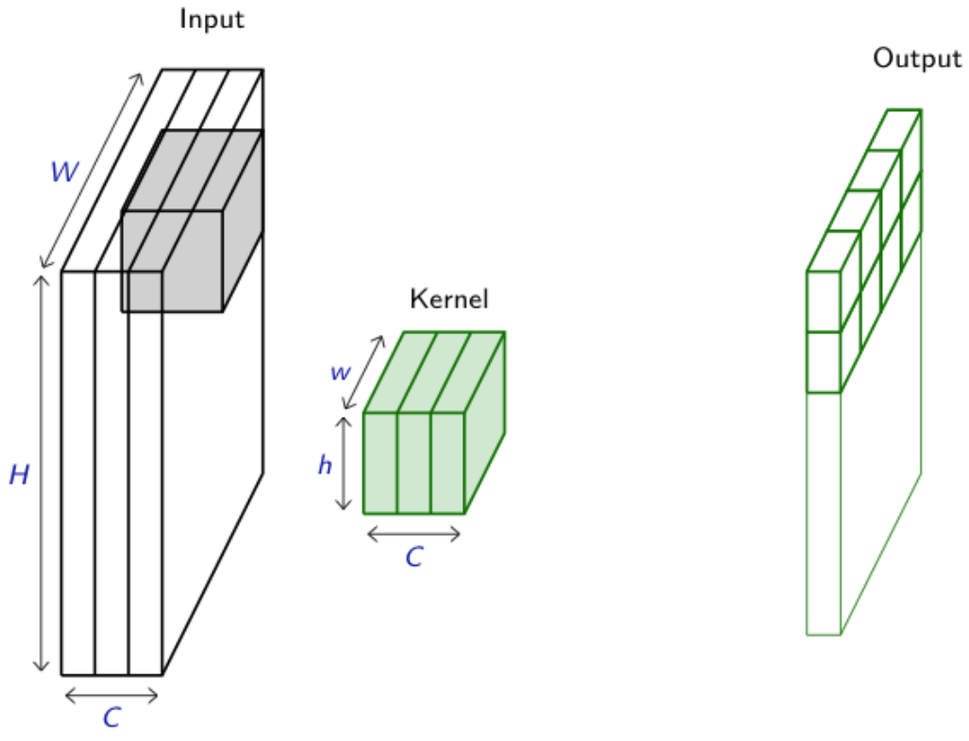
Convolution 2d



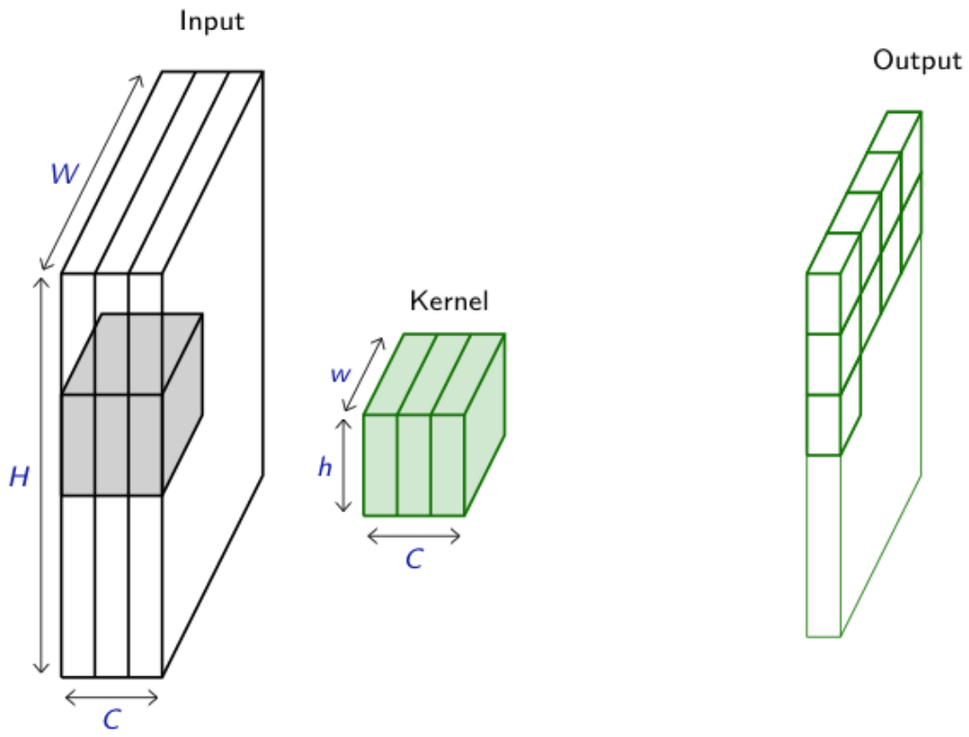
Convolution 2d



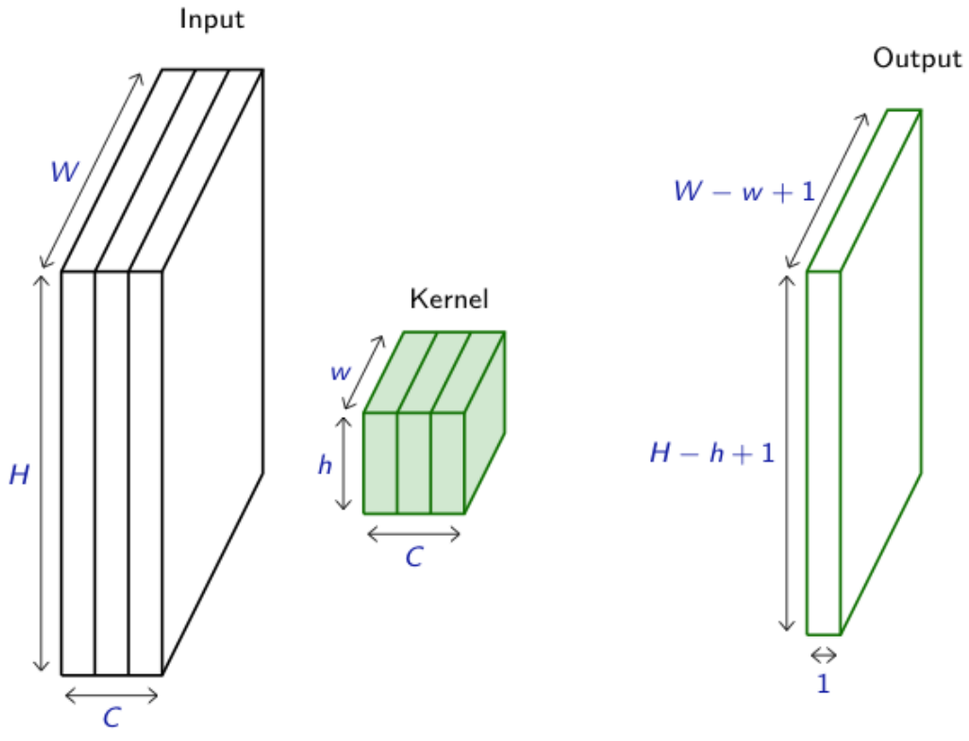
Convolution 2d



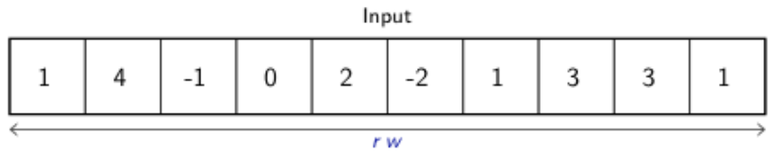
Convolution 2d



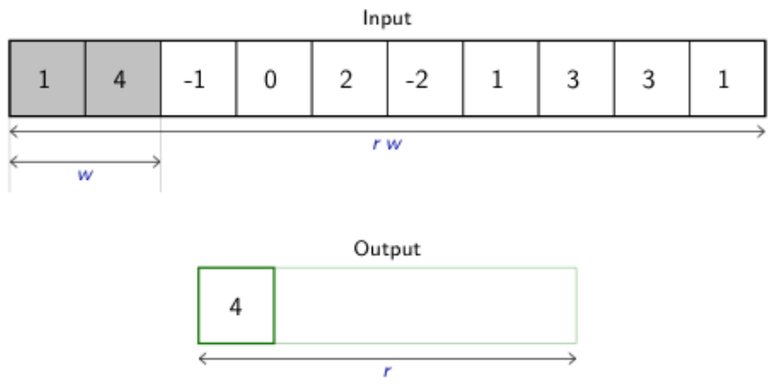
Convolution 2d



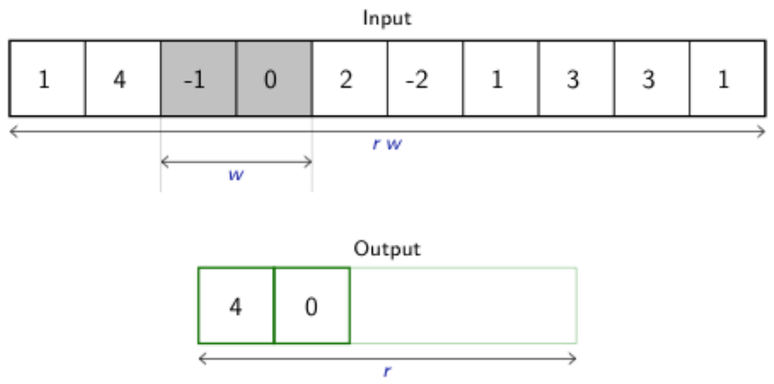
Max-Pooling 1d



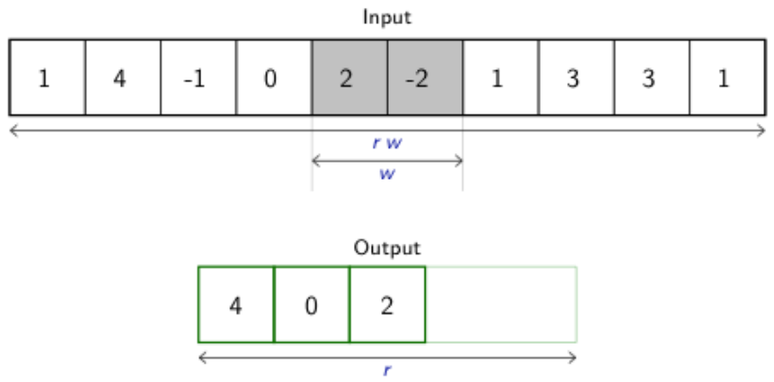
Max-Pooling 1d



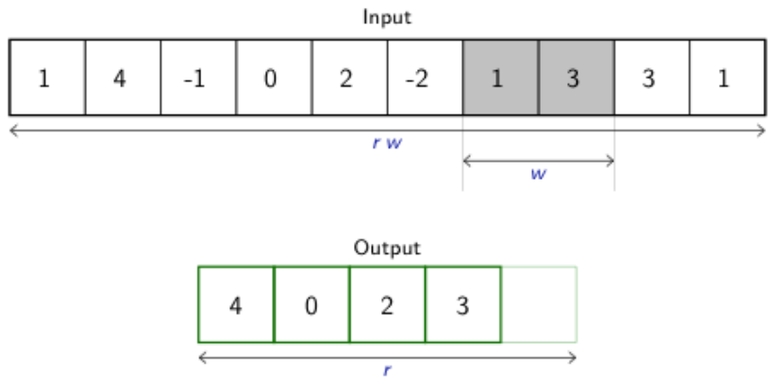
Max-Pooling 1d



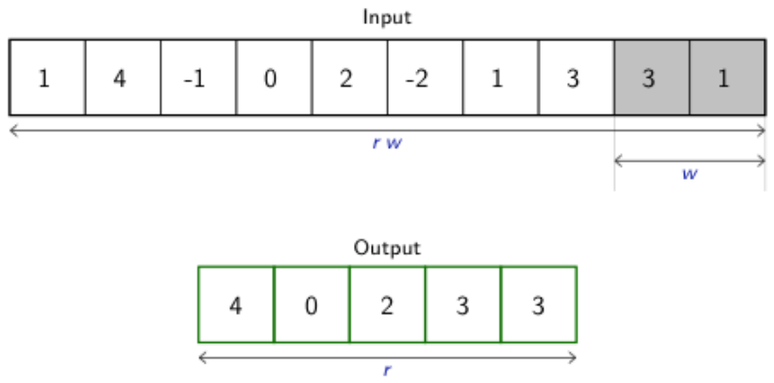
Max-Pooling 1d



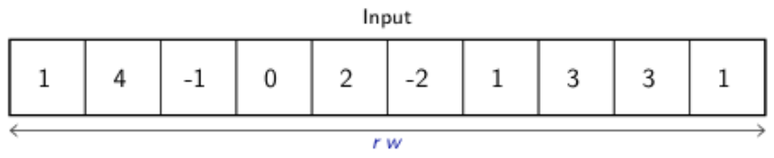
Max-Pooling 1d

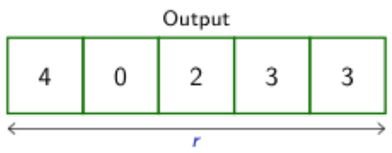


Max-Pooling 1d

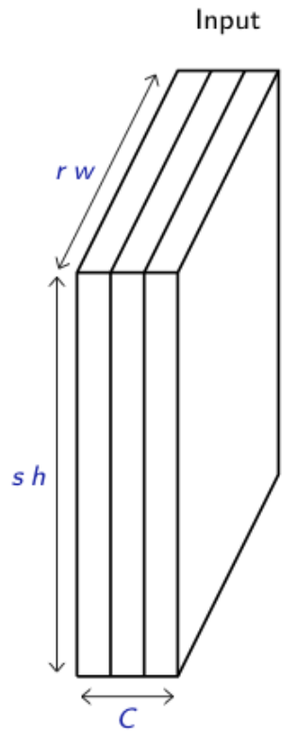


Max-Pooling 1d

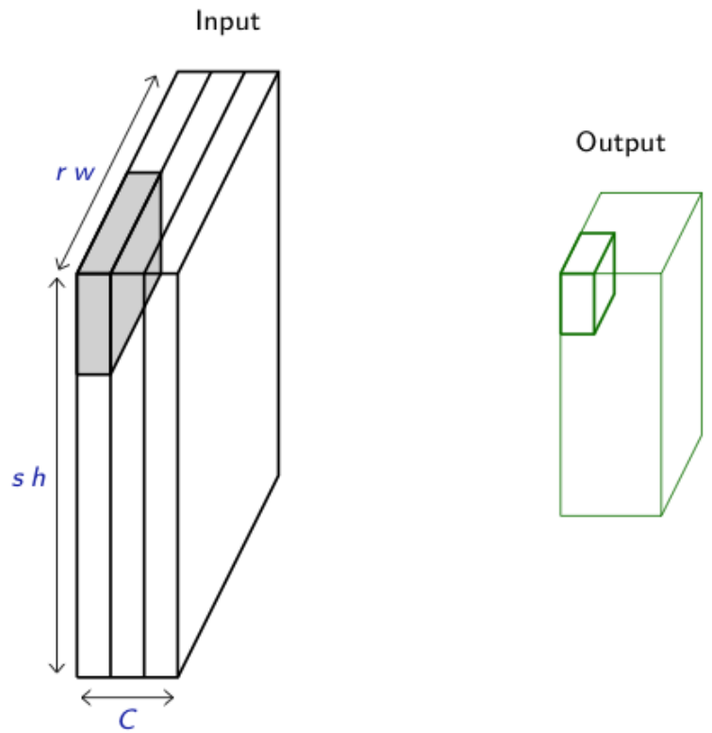




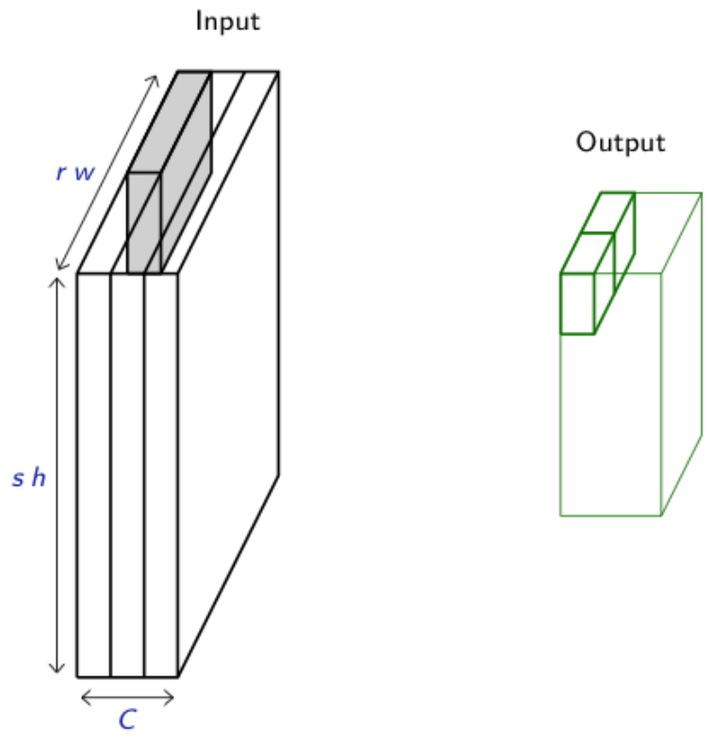
Max-Pooling 2d



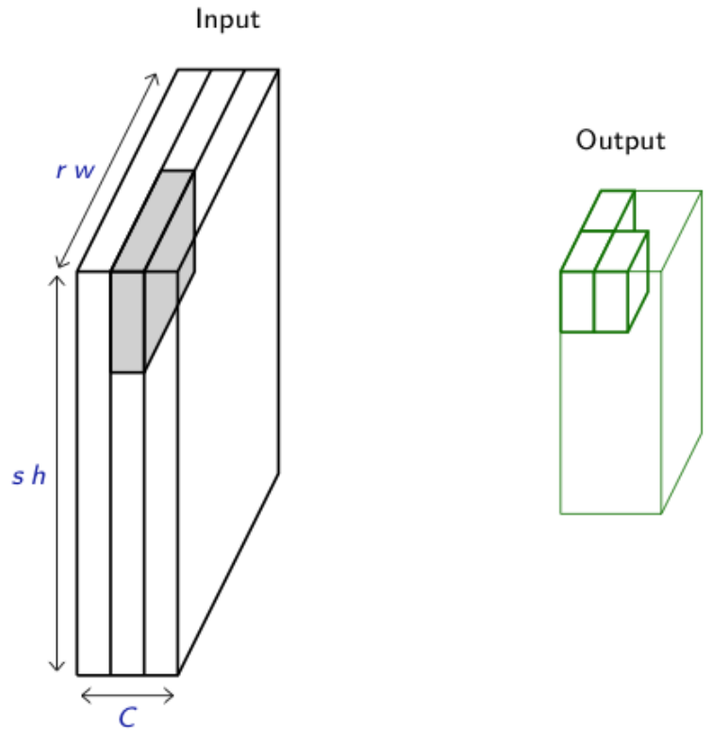
Max-Pooling 2d



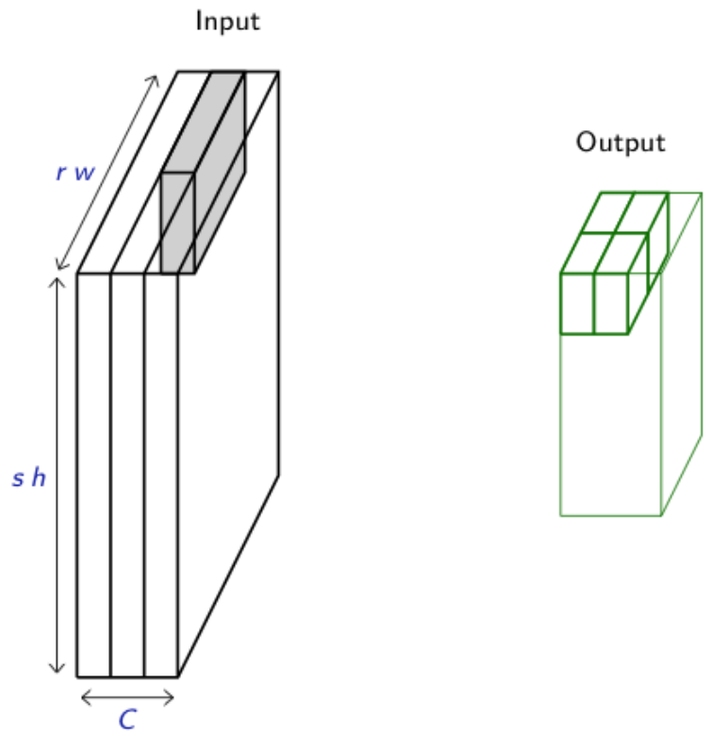
Max-Pooling 2d



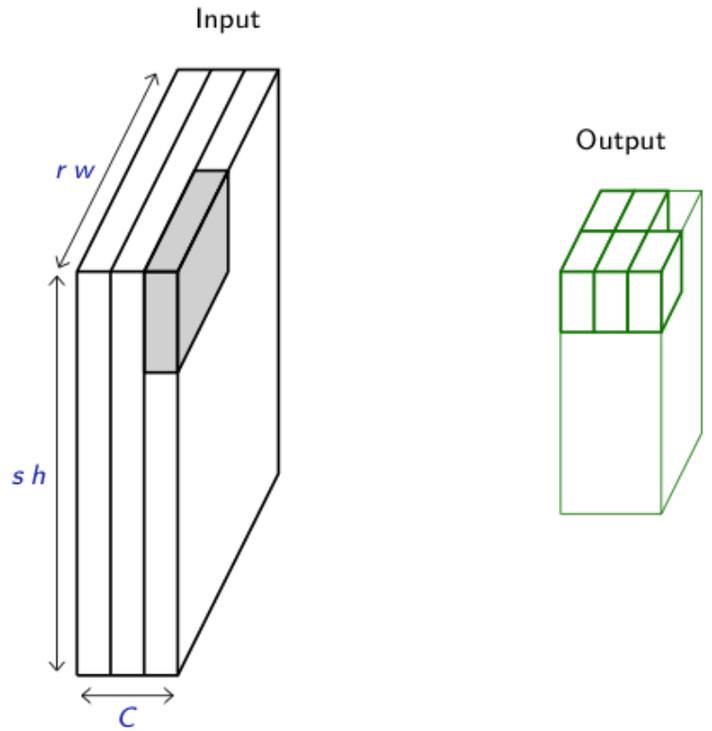
Max-Pooling 2d



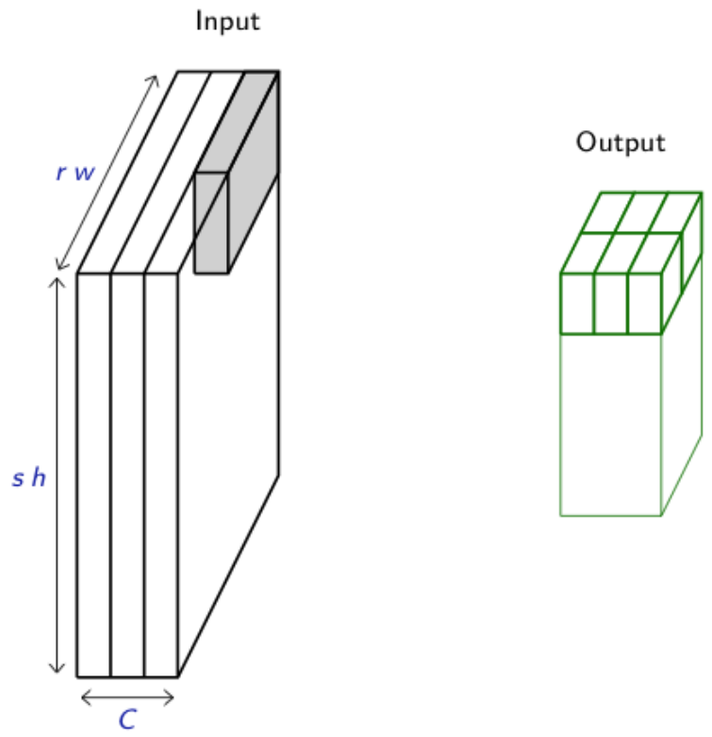
Max-Pooling 2d



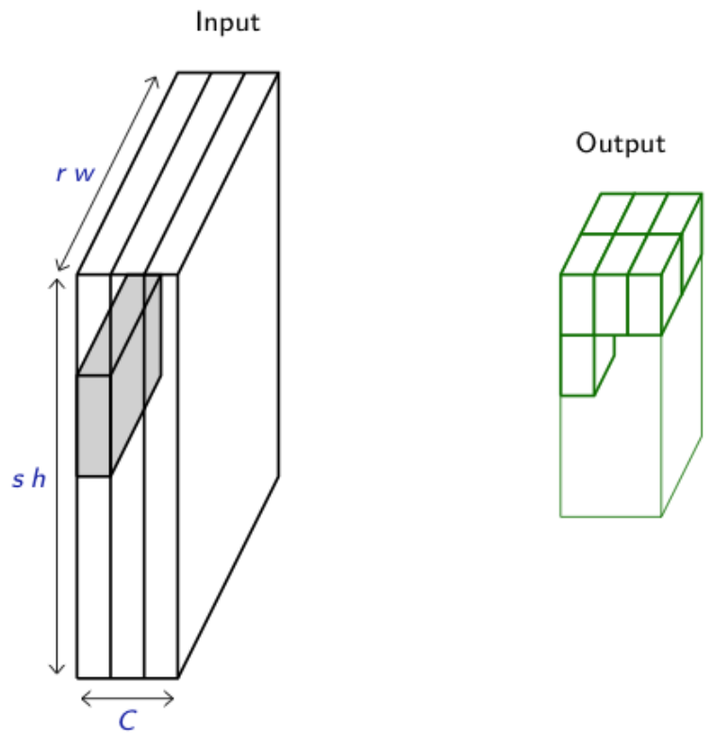
Max-Pooling 2d



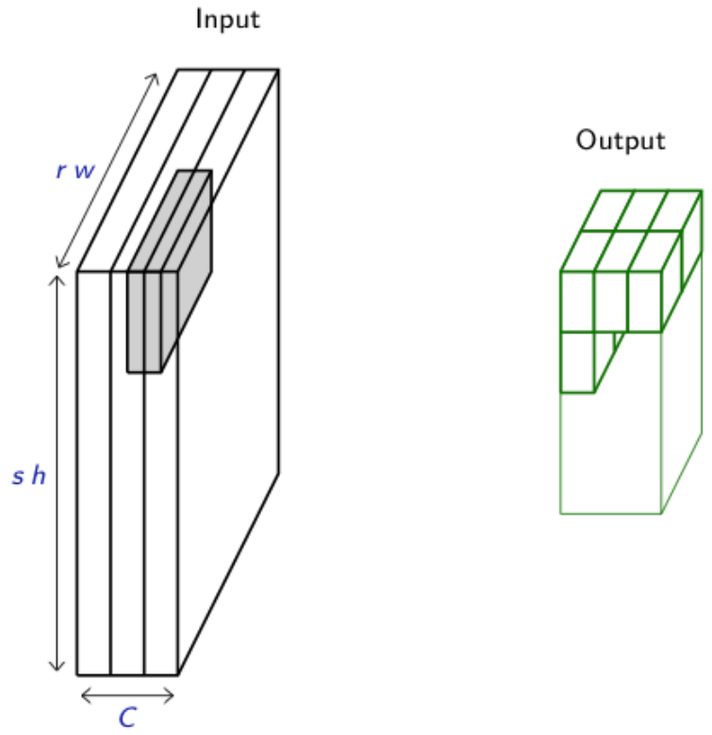
Max-Pooling 2d



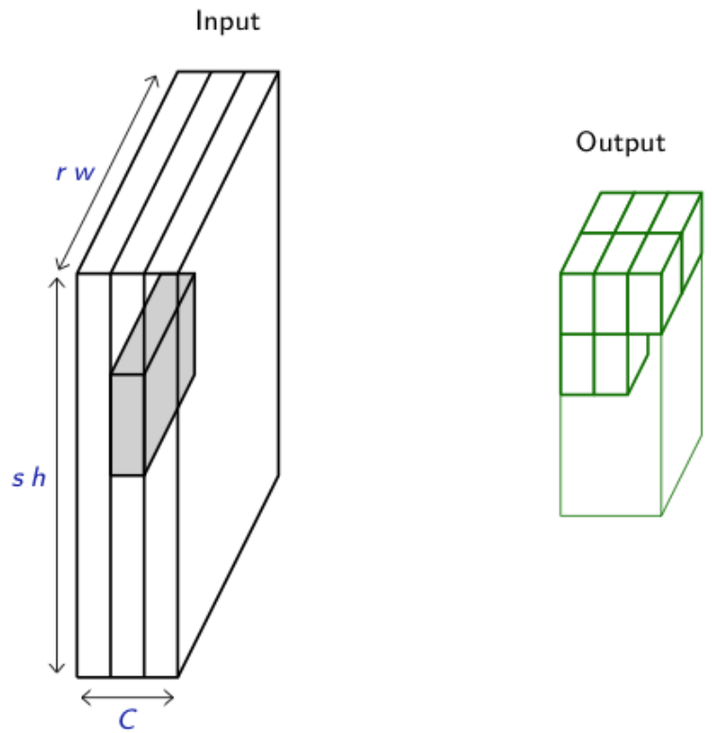
Max-Pooling 2d



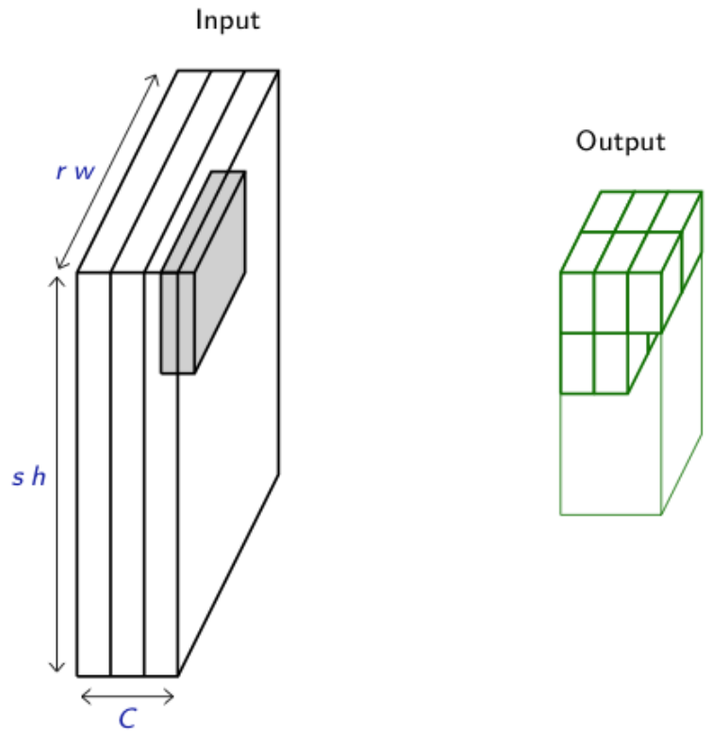
Max-Pooling 2d



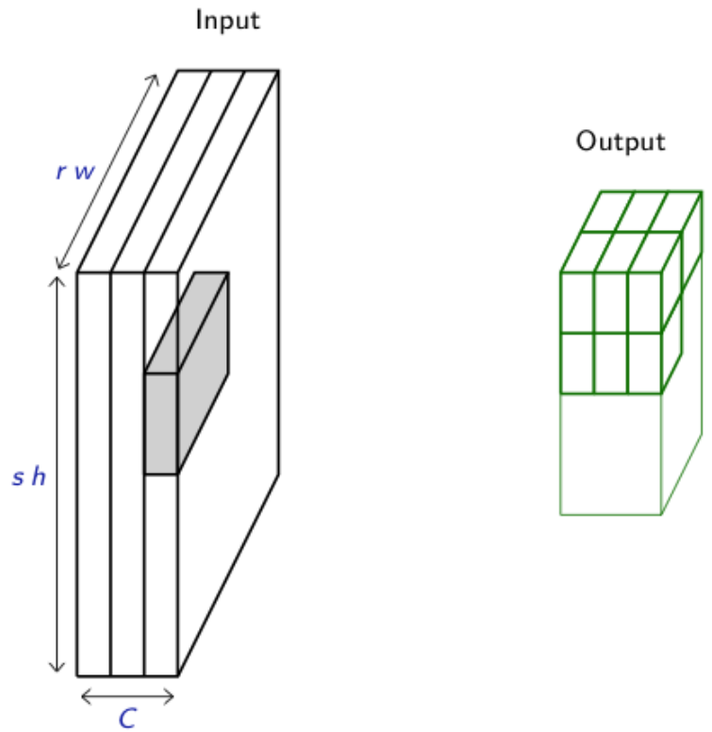
Max-Pooling 2d



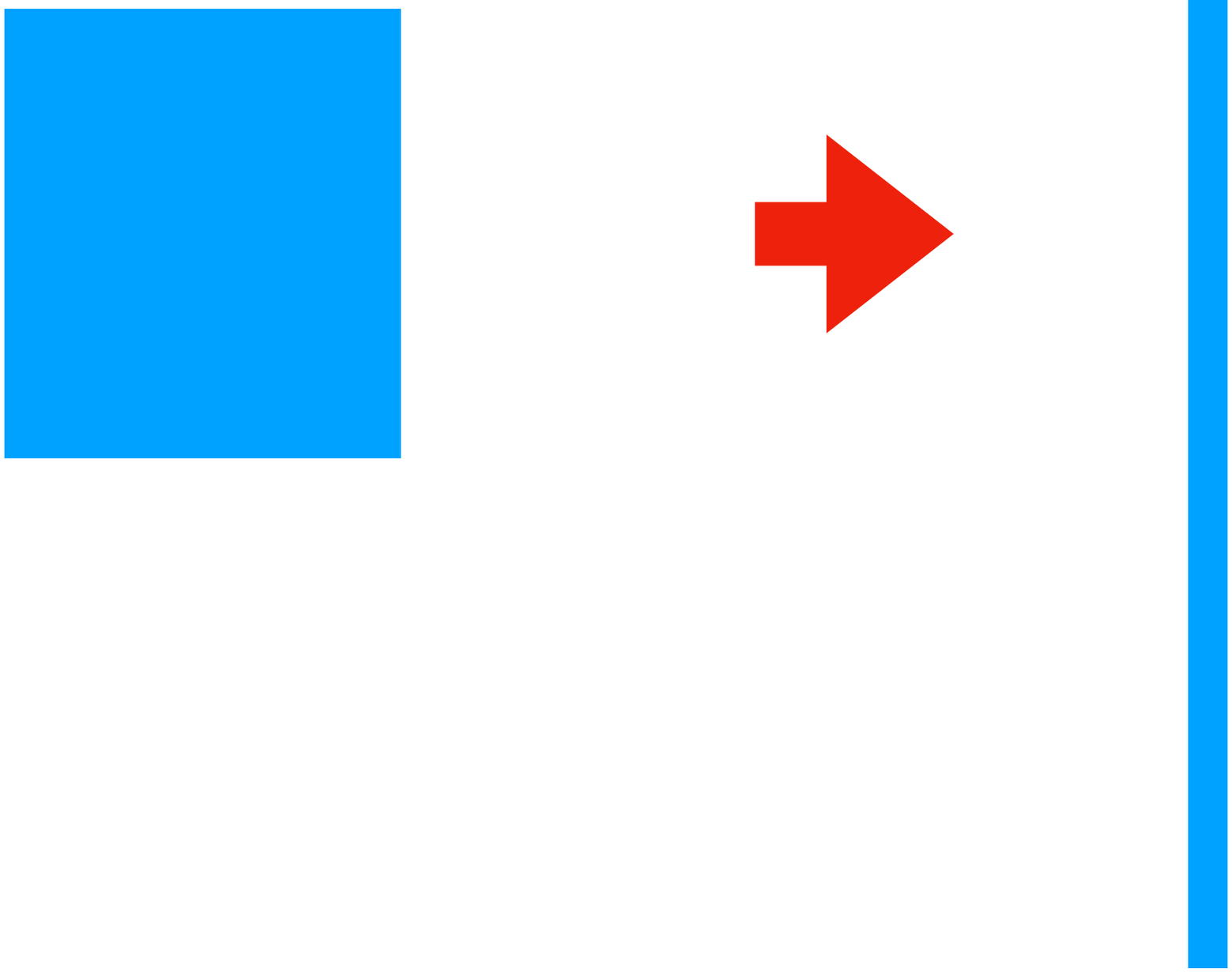
Max-Pooling 2d



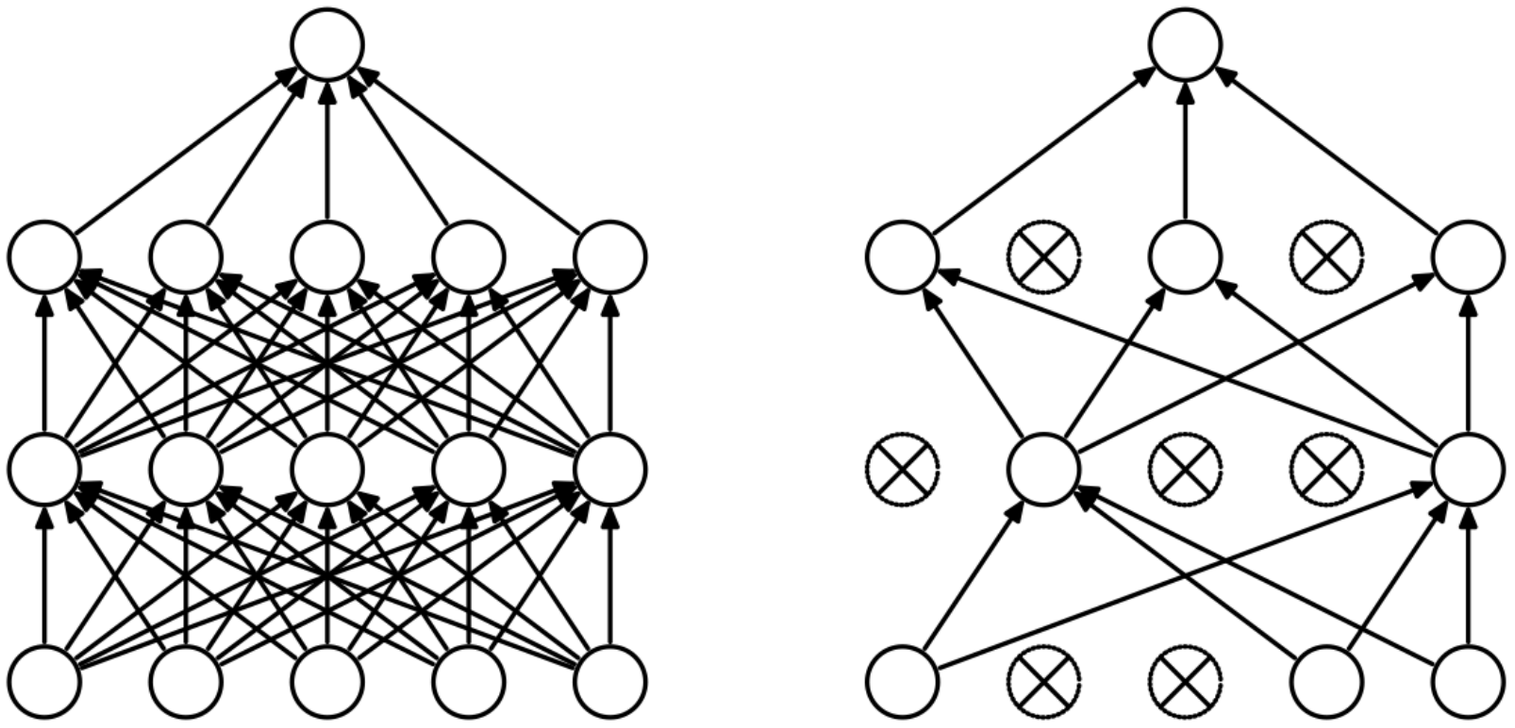
Max-Pooling 2d



Flatten
2D->1D



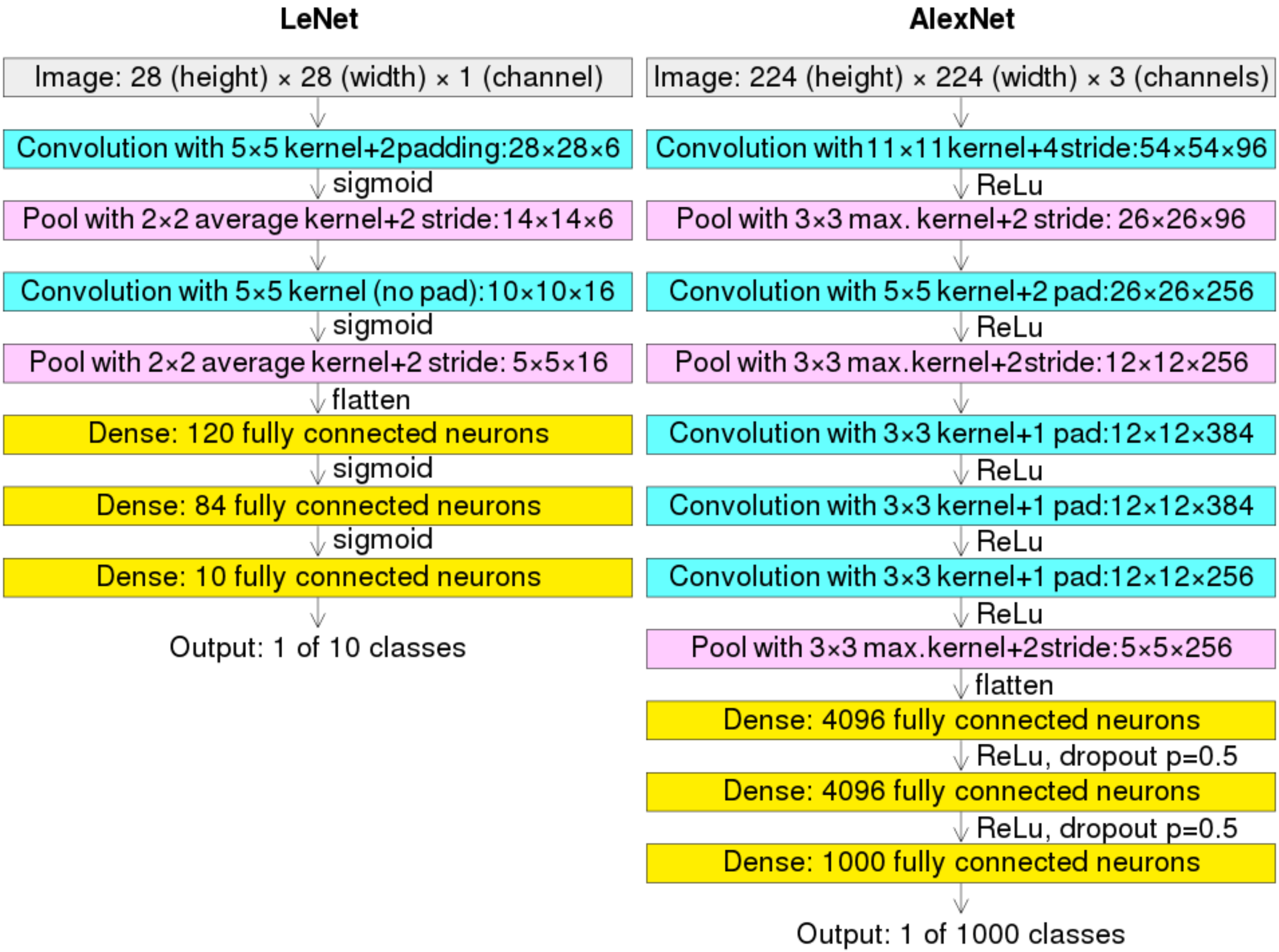
Dropout

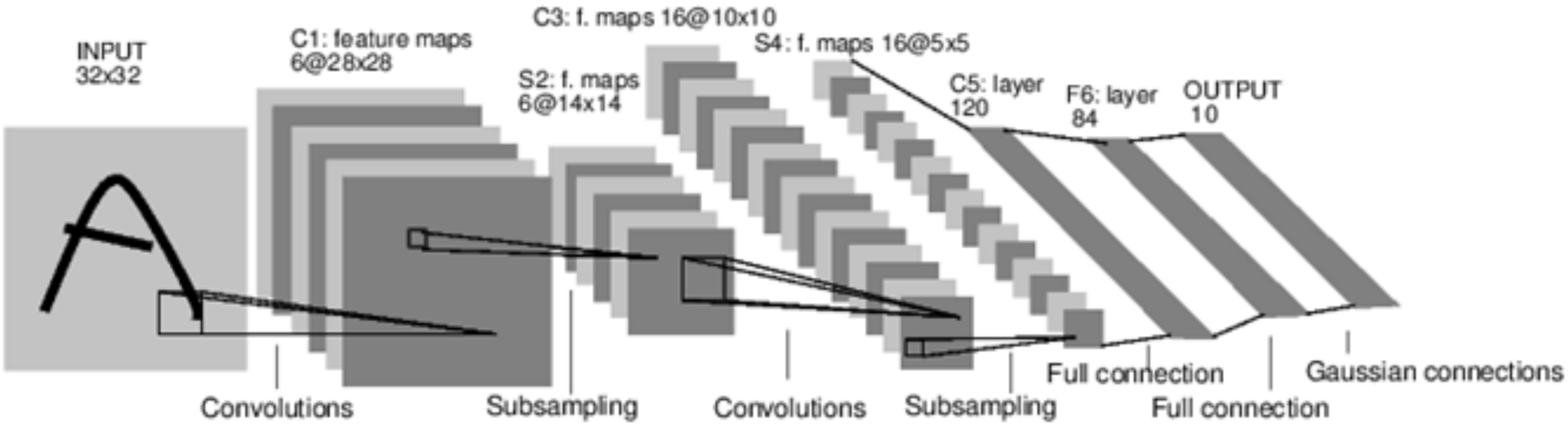


(a) Standard Neural Net

(b) After applying dropout.

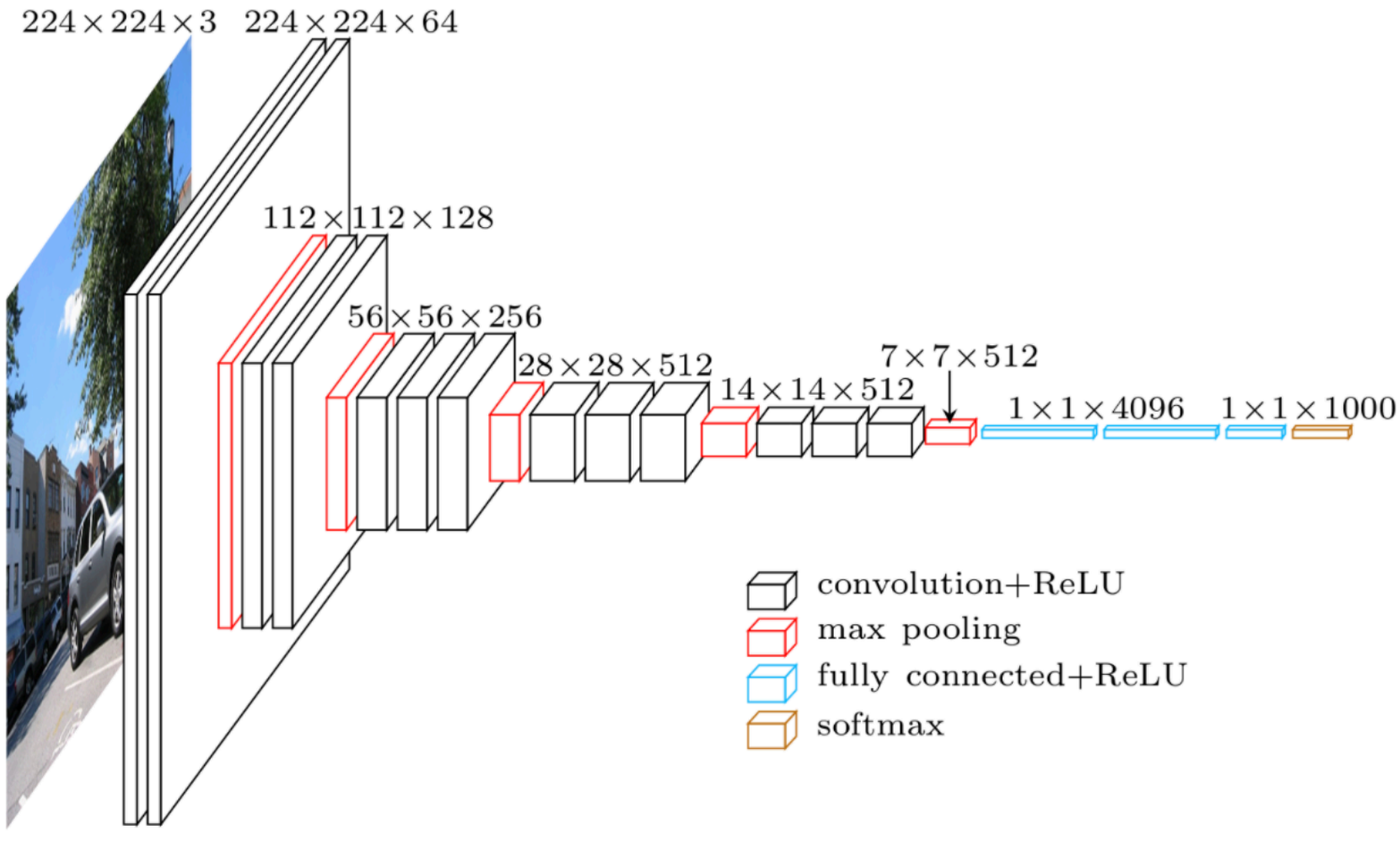
Each time we load an example into a minibatch, we randomly sample a different binary mask to apply to all of the input and hidden units in the network. The mask for each unit is sampled independently from all of the others


LeNet
Image: 28 (height) × 28 (width) × 1 (channel)
Convolution with 5×5 kernel+2padding:28×28×6
sigmoid
Pool with 2×2 average kernel+2 stride:14×14×6
Convolution with 5×5 kernel (no pad):10×10×16
sigmoid
Pool with 2×2 average kernel+2 stride: 5×5×16
flatten
Dense: 120 fully connected neurons
sigmoid
Dense: 84 fully connected neurons
sigmoid
Dense: 10 fully connected neurons
Output: 1 of 10 classes
AlexNet
Image: 224 (height) × 224 (width) × 3 (channels)
Convolution with 11×11 kernel+4stride:54×54×96
ReLu
Pool with 3×3 max. kernel+2 stride: 26×26×96
Convolution with 5×5 kernel+2 pad:26×26×256
ReLu
Pool with 3×3 max.kernel+2stride:12×12×256
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×256
ReLu
Pool with 3×3 max.kernel+2stride:5×5×256
flatten
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 1000 fully connected neurons
Output: 1 of 1000 classes

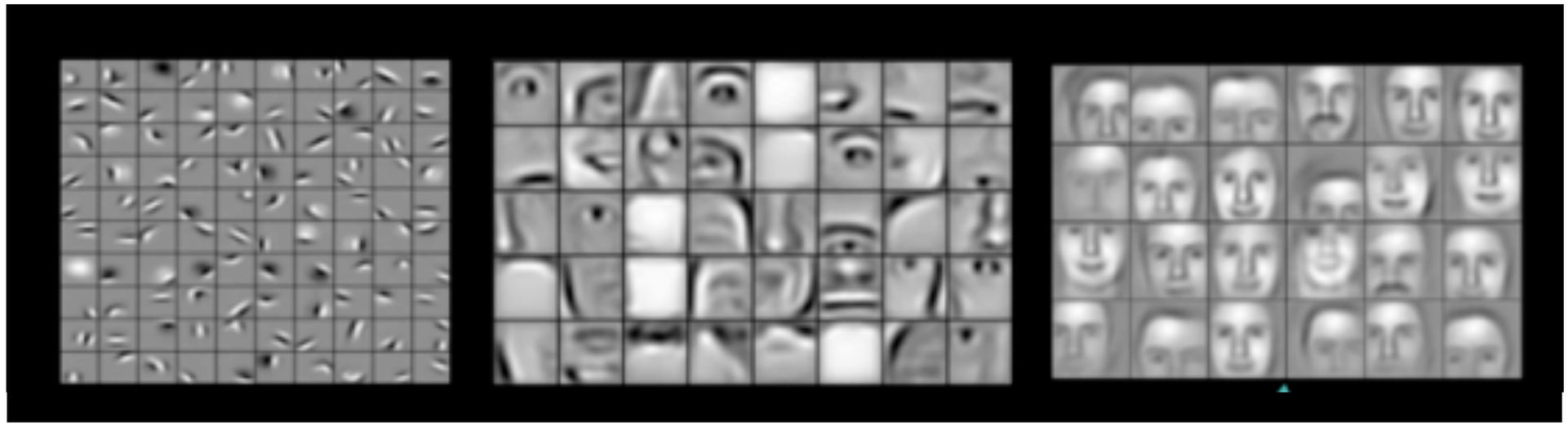


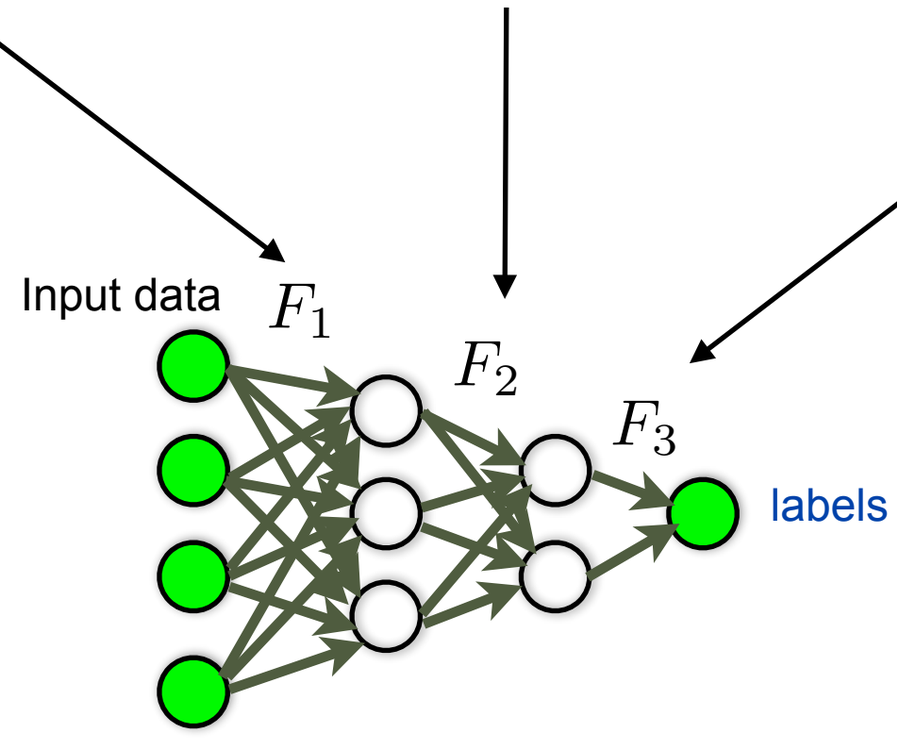
A Full Convolutional Neural Network (LeNet)

VGG-16

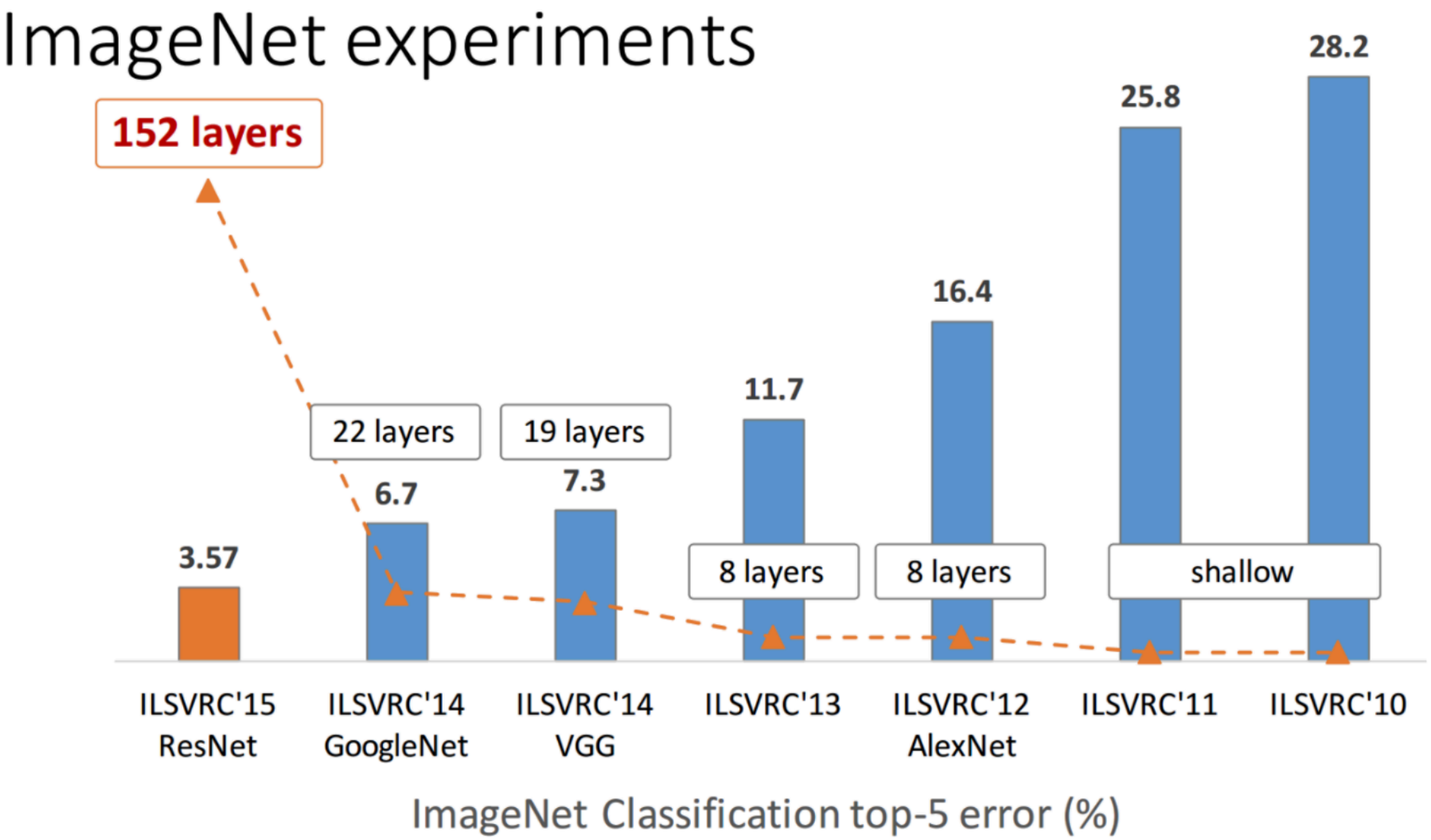

224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

Hierarchy of features





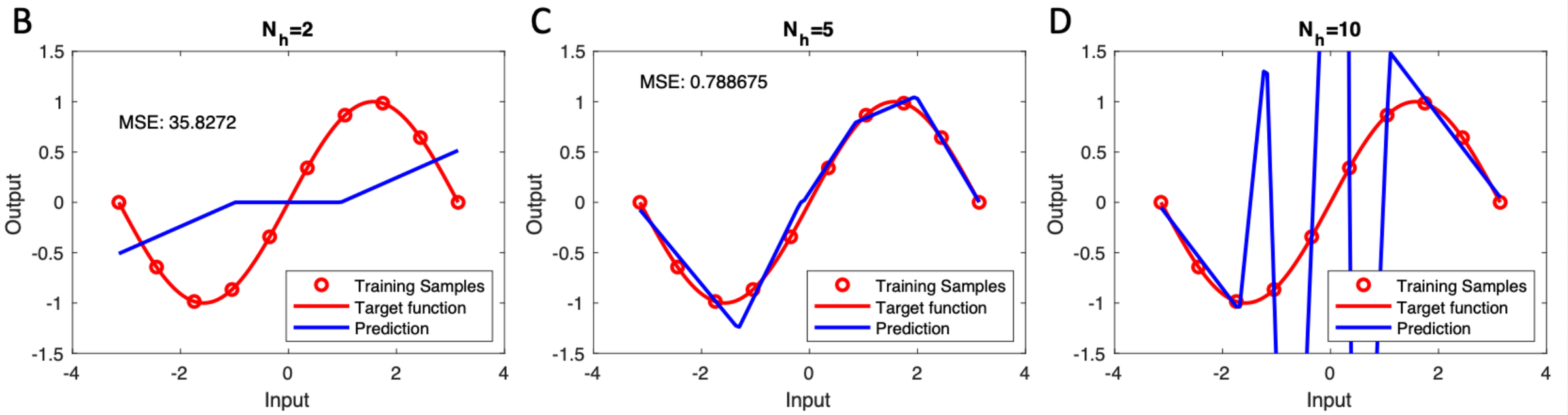
Deeper is better

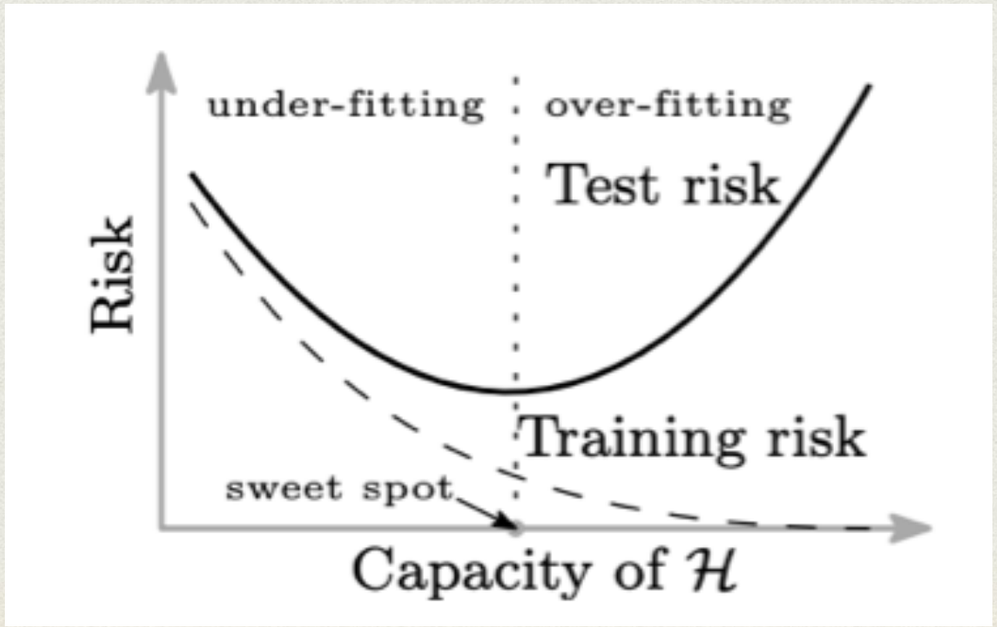


Deep Learning

Modus Operandi

OVERFITTING





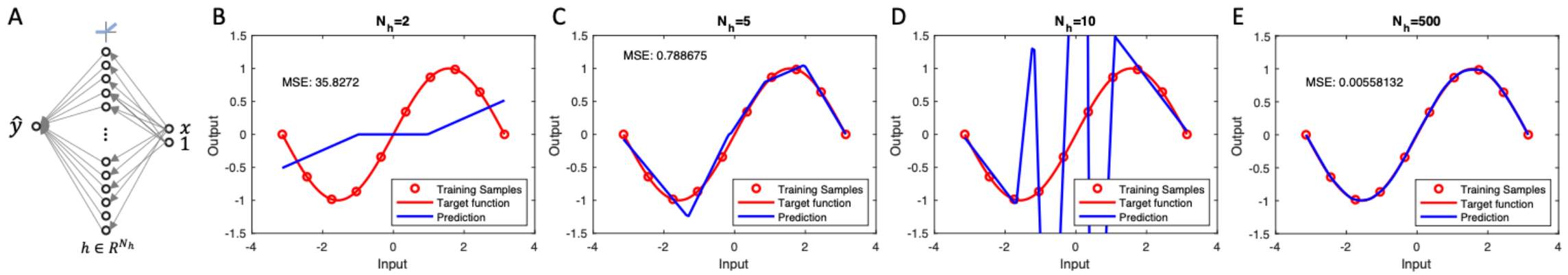
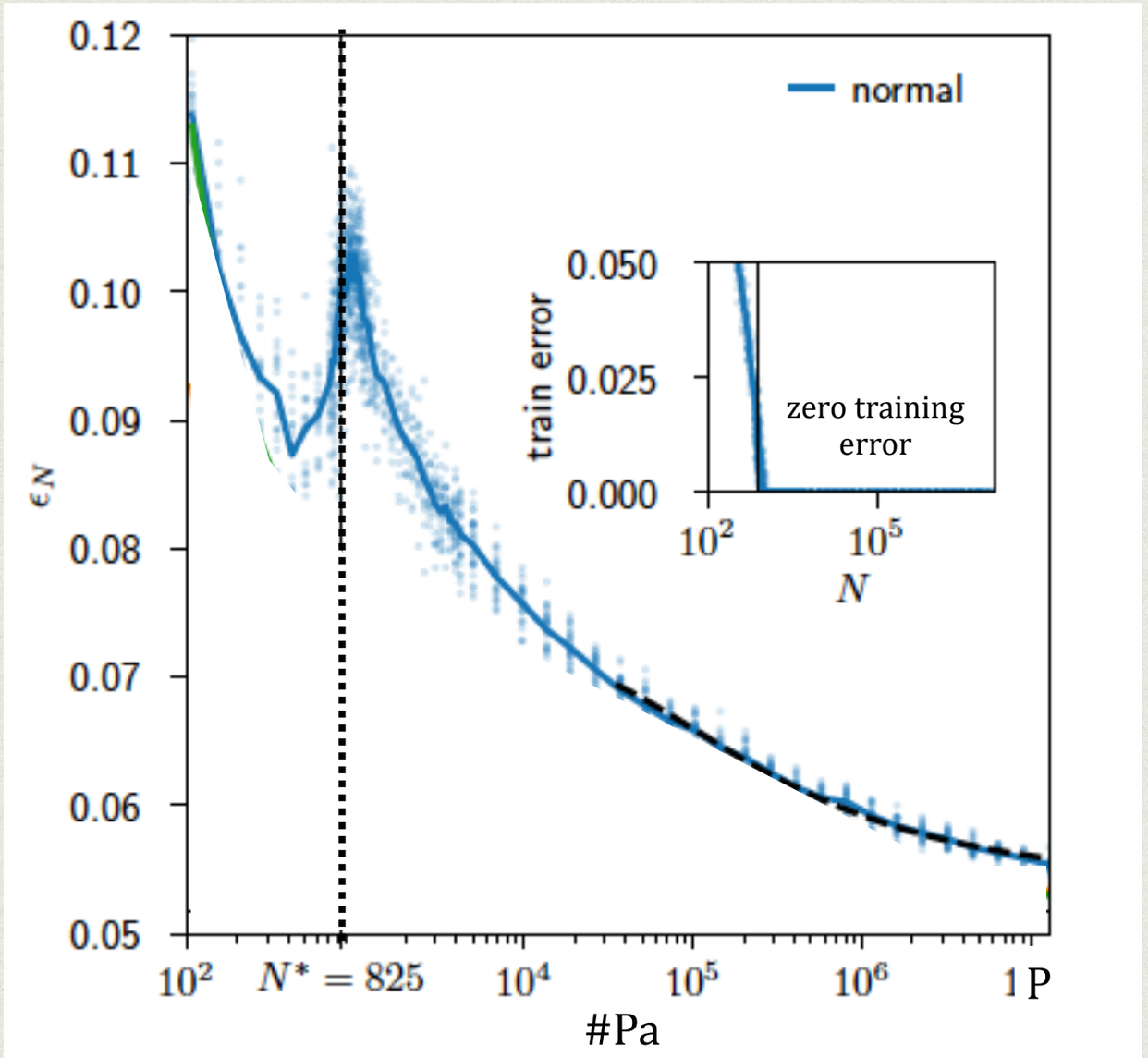


Figure 1: Over-parameterization and generalization via minimum norm solutions. (A) A simple ReLU network with random first layer weights and trained second layer weights. The network receives a scalar input and bias term, and is trained to minimize squared error on ten points (red circles) from a target sinusoid (red curve) shown in panels B-E. (B-E) Blue curves show example functions learned by networks with differing numbers of hidden units $N_h = \{2, 5, 10, 500\}$. Networks in panels B, C, & D show the standard progression from underfitting the training data to overfitting, with a happy medium in panel C. However the large network in panel E generalizes best. This network has $50\times$ more parameters than training examples but generalizes well, because among the infinity of solutions attaining zero training error we have chosen a low norm solution. In this work we derive training algorithms for nonlinear deep networks that explicitly seek minimum norm solutions in the underdetermined, accurate label regime, allowing good generalization in large networks.

Saxe, Advani'17

DOUBLE-DESCENT



Parity-MNIST, 5 layers, FCN, hinge loss, no regularisation

[Geiger et al. '18]

DOUBLE-DESCENT

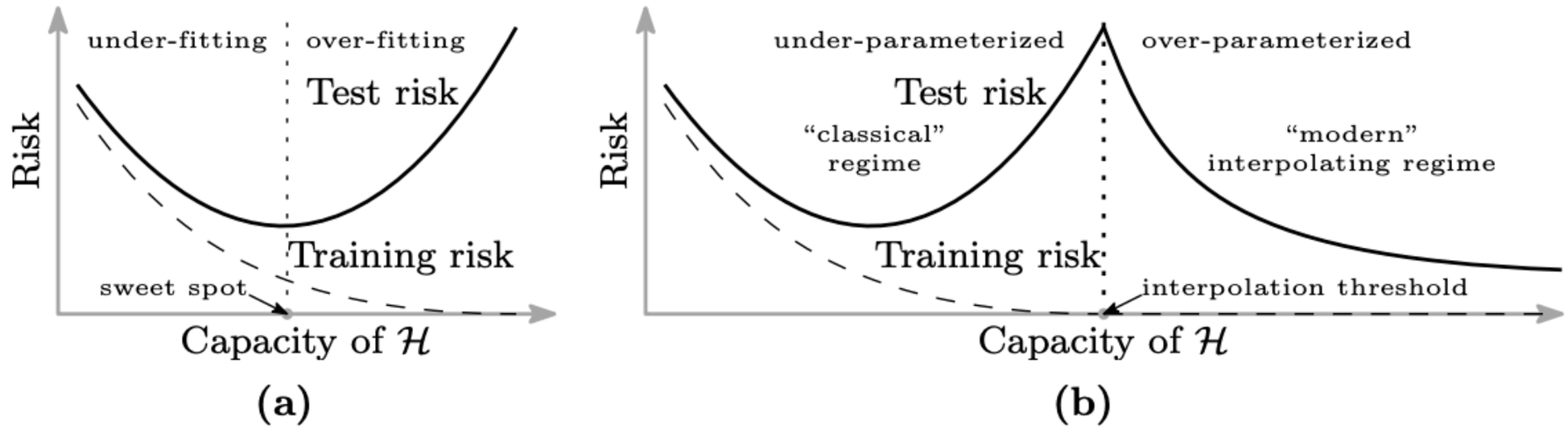


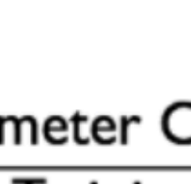
Figure 1: **Curves for training risk (dashed line) and test risk (solid line).** (**a**) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (**b**) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Belkin, Hsu, Ma, Mandal'19

Overparametrization

Parameter Count / Num Training Samples

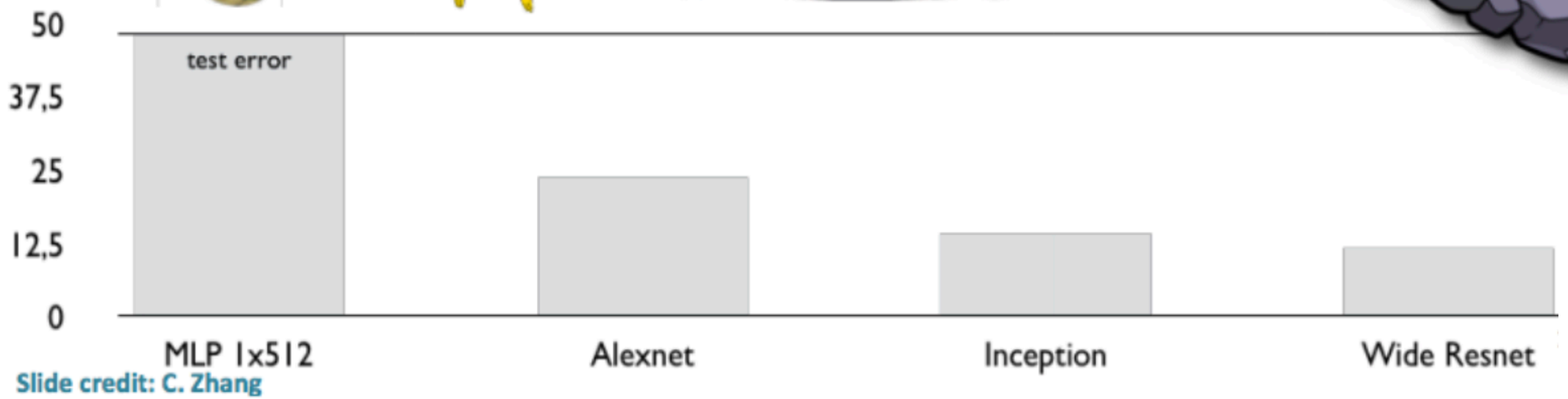
MLP 1x512
p/n: 24



Alexnet
p/n: 28

Inception
p/n: 33

Wide Resnet
p/n: 179



Slide credit: C. Zhang

UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

Chiyuan Zhang*
Massachusetts Institute of Technology
chiyuan@mit.edu

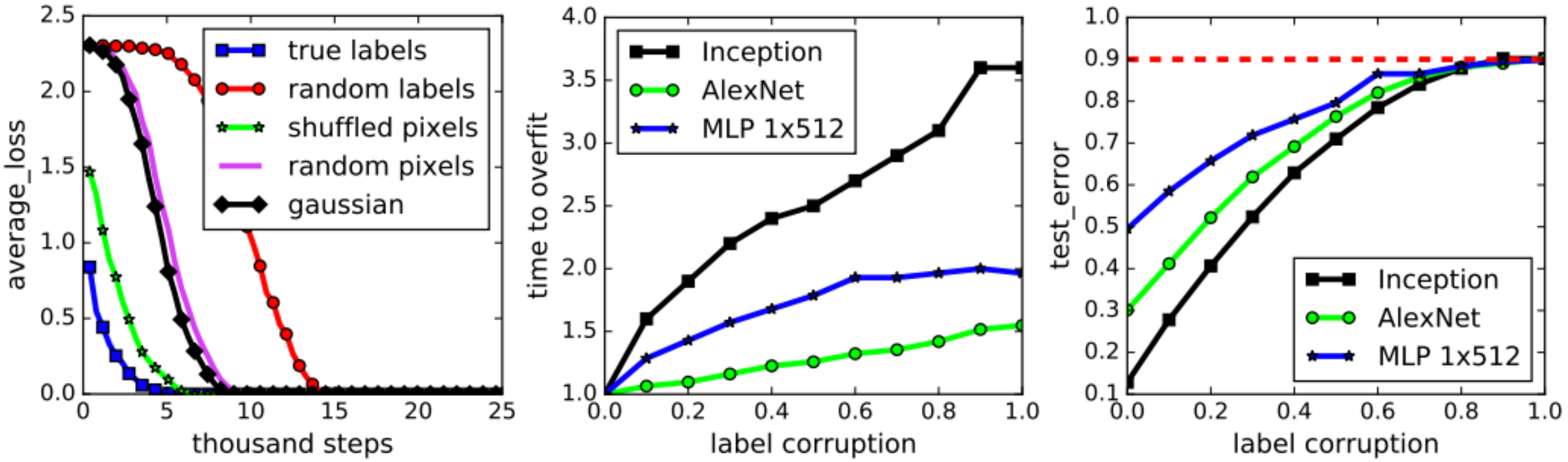
Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht†
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com

- State-of-the-art neural networks are able to fit random labels.
- Classical bounds on the generalisation error (VC, Rademacher) are void, as they rely on not being able to fit random labels.



(a) learning curves (b) convergence slowdown (c) generalization error growth

Figure 1: Fitting random labels and random pixels on CIFAR10. (a) shows the training loss of various experiment settings decaying with the training steps. (b) shows the relative convergence time with different label corruption ratio. (c) shows the test error (also the generalization error since training error is 0) under different label corruptions.

Bad Global Minima Exist and SGD Can Reach Them

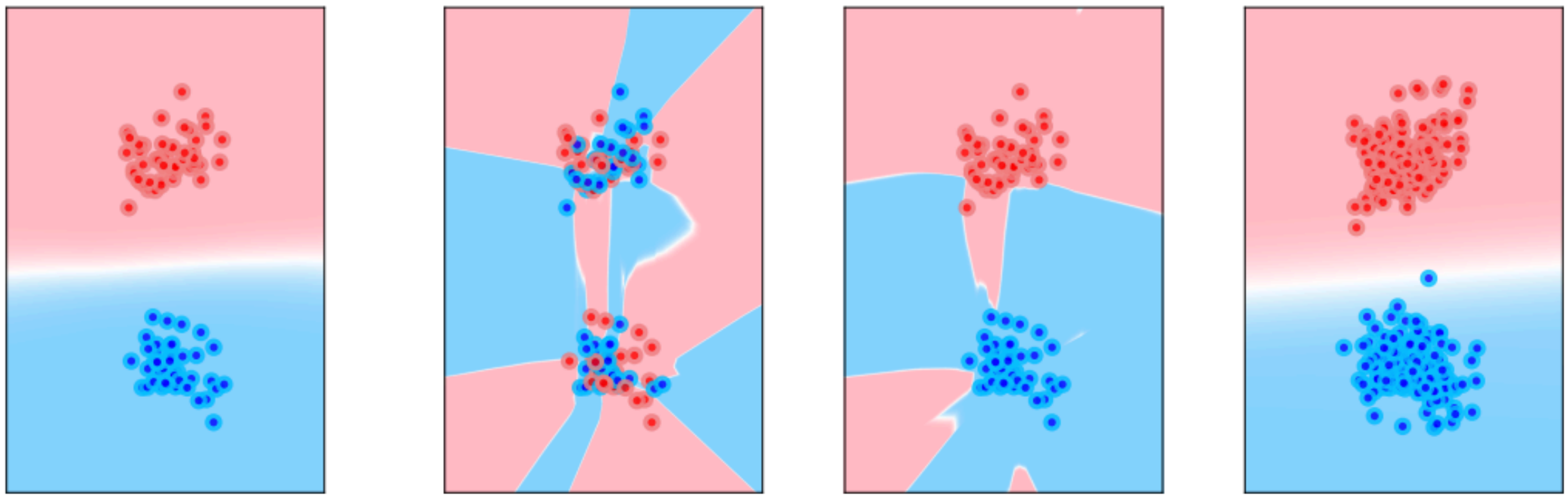
Shengchao Liu, Dimitris Papailiopoulos
University of Wisconsin–Madison

Dimitris Achlioptas
University of California, Santa Cruz

Abstract

Several recent works have aimed to explain why severely overparameterized models, generalize well when trained by Stochastic Gradient Descent (SGD). The emergent consensus explanation has two parts: the first is that there are "no bad local minima", while the second is that SGD performs implicit regularization by having a bias towards low complexity models. We revisit both of these ideas in the context of image classification with common deep neural network architectures. Our first finding is that there exist bad *global* minima, *i.e.*, models that fit the training set perfectly, yet have poor generalization. Our second finding is that given only *unlabeled* training data, we can easily construct initializations that will cause SGD to quickly converge to such bad global minima. For example, on CIFAR, CINIC10, and (Restricted) ImageNet, this can be achieved by starting SGD at a model derived by fitting random labels on the training data: while subsequent SGD training (with the correct labels) will reach zero training error, the resulting model will exhibit a test accuracy degradation of up to 40% compared to training from a random initialization. Finally, we show that regularization seems to provide SGD with an escape route: once heuristics such as data augmentation are used, starting from a complex model (adversarial initialization) has no effect on the test accuracy.

1. Random initialization + Training with true labels.
2. Random initialization + Training with random labels.
3. Random initialization + Training with random labels + Training with true labels.
4. Random initialization + Training with random labels + Training with true labels using data augmentation[1] and l_2 regularization.

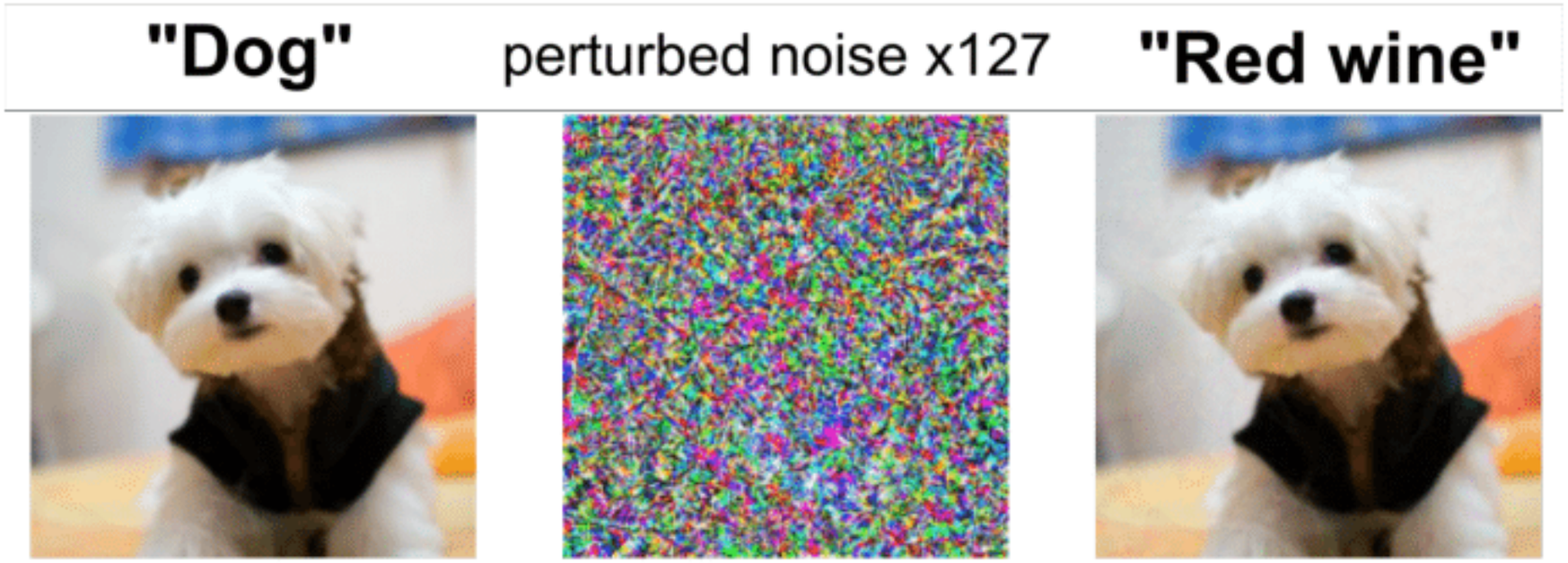


(a) Setting 1 (b) Setting 2 (c) Setting 3 (d) Setting 4

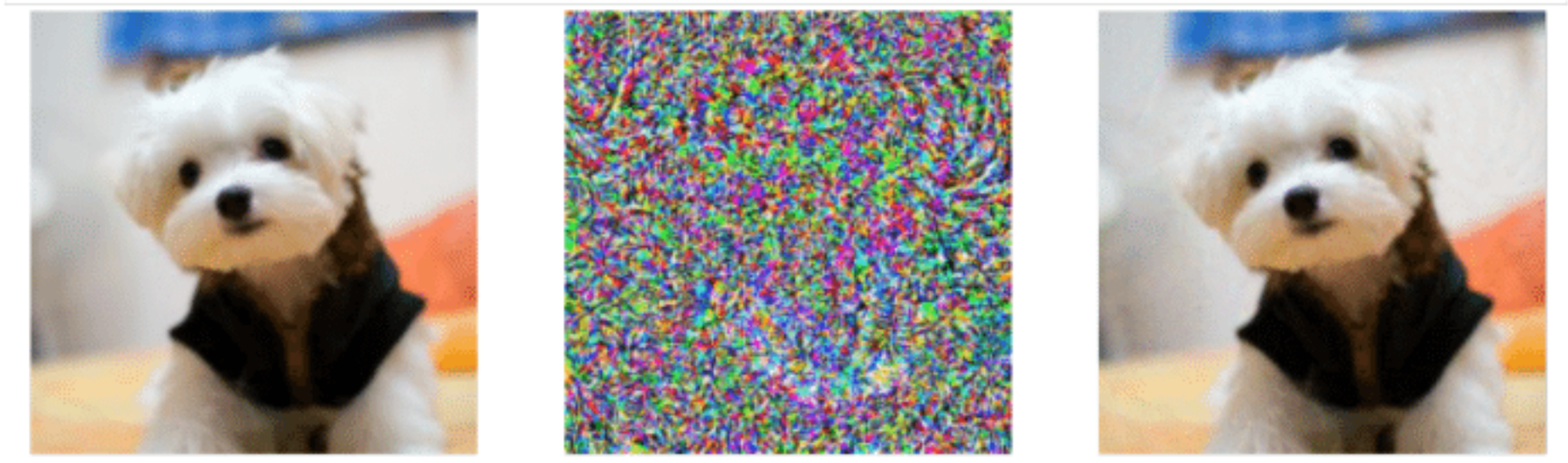
Figure 1: The decision boundary of the model reached by SGD in Settings 1–4, respectively.

Adversarial examples

a)



b)



Adversarial examples

classified as

Stop Sign

+

=

classified as

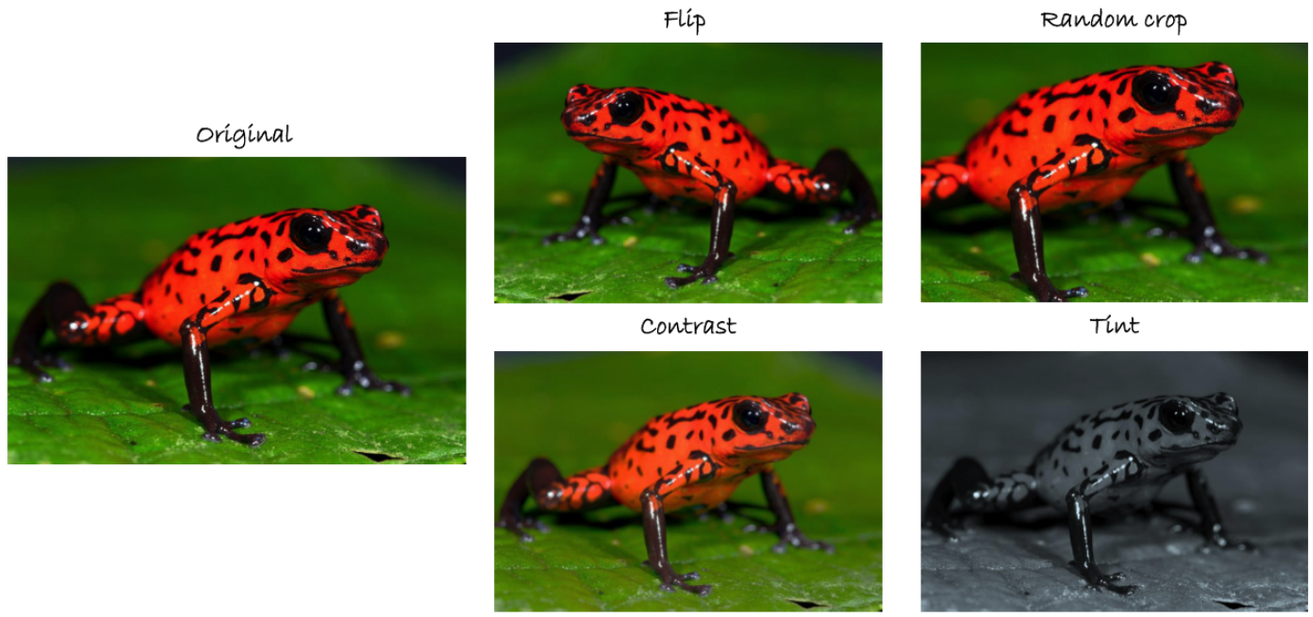
Max Speed 100

What do you do when do not have enough data?

You create more!

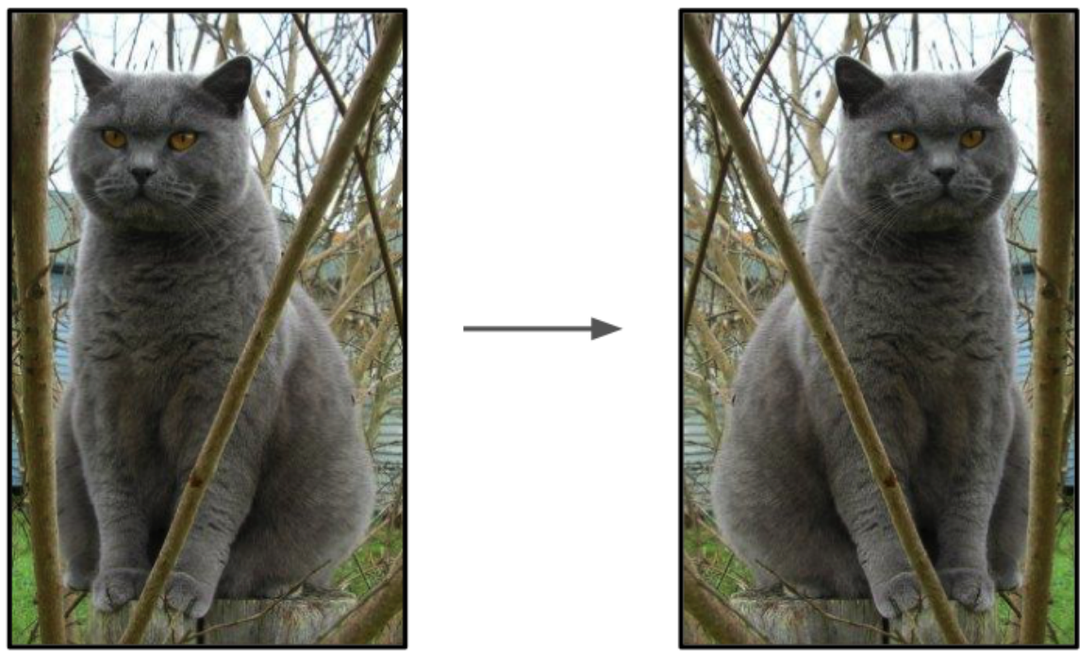
Data augmentation

- Changing the pixels without changing the label
- Train on transformed data
- Widely used



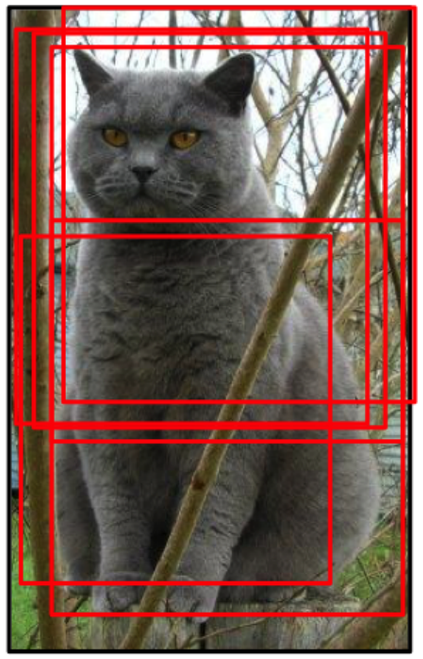
Data augmentation

Horizontal flips



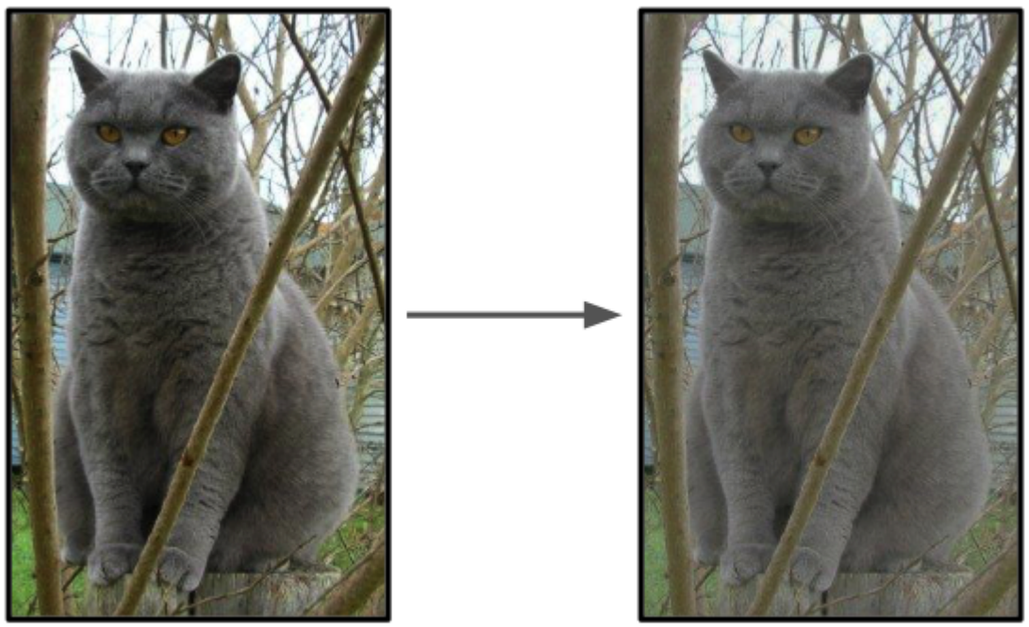
Data augmentation

Random crops/scales



Data augmentation

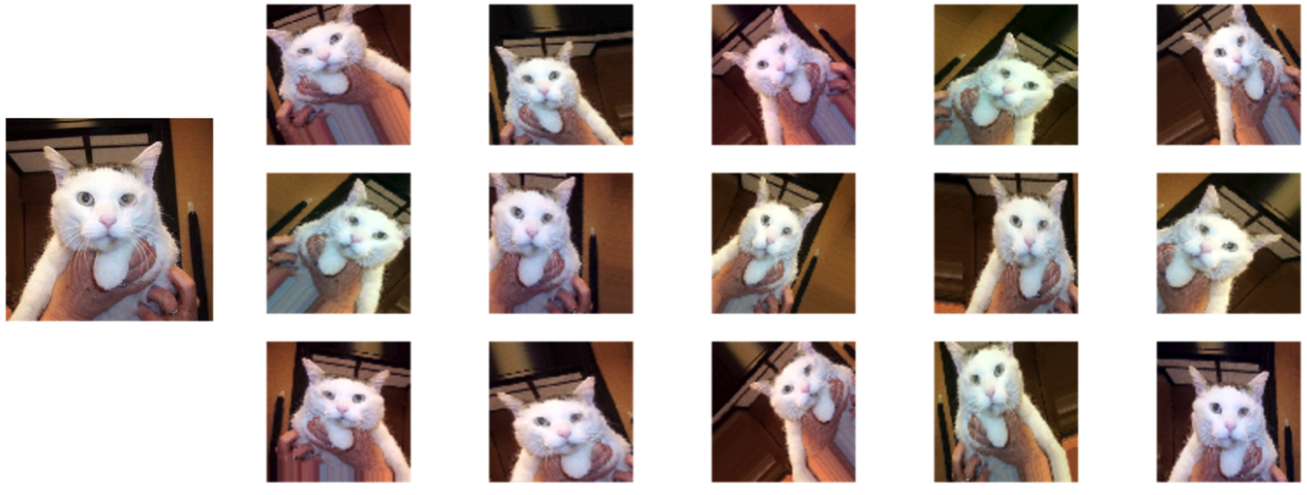
Color jitter



- randomly jitter color, brightness, contrast, etc.

Data augmentation

- Various techniques can be mixed
- Domain knowledge helps in finding new data augmentation techniques
- Very useful for small datasets



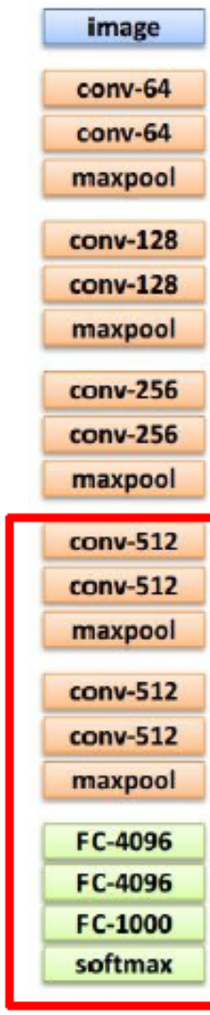
Transfer Learning

Transfer Learning with CNNs

1. Train on ImageNet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

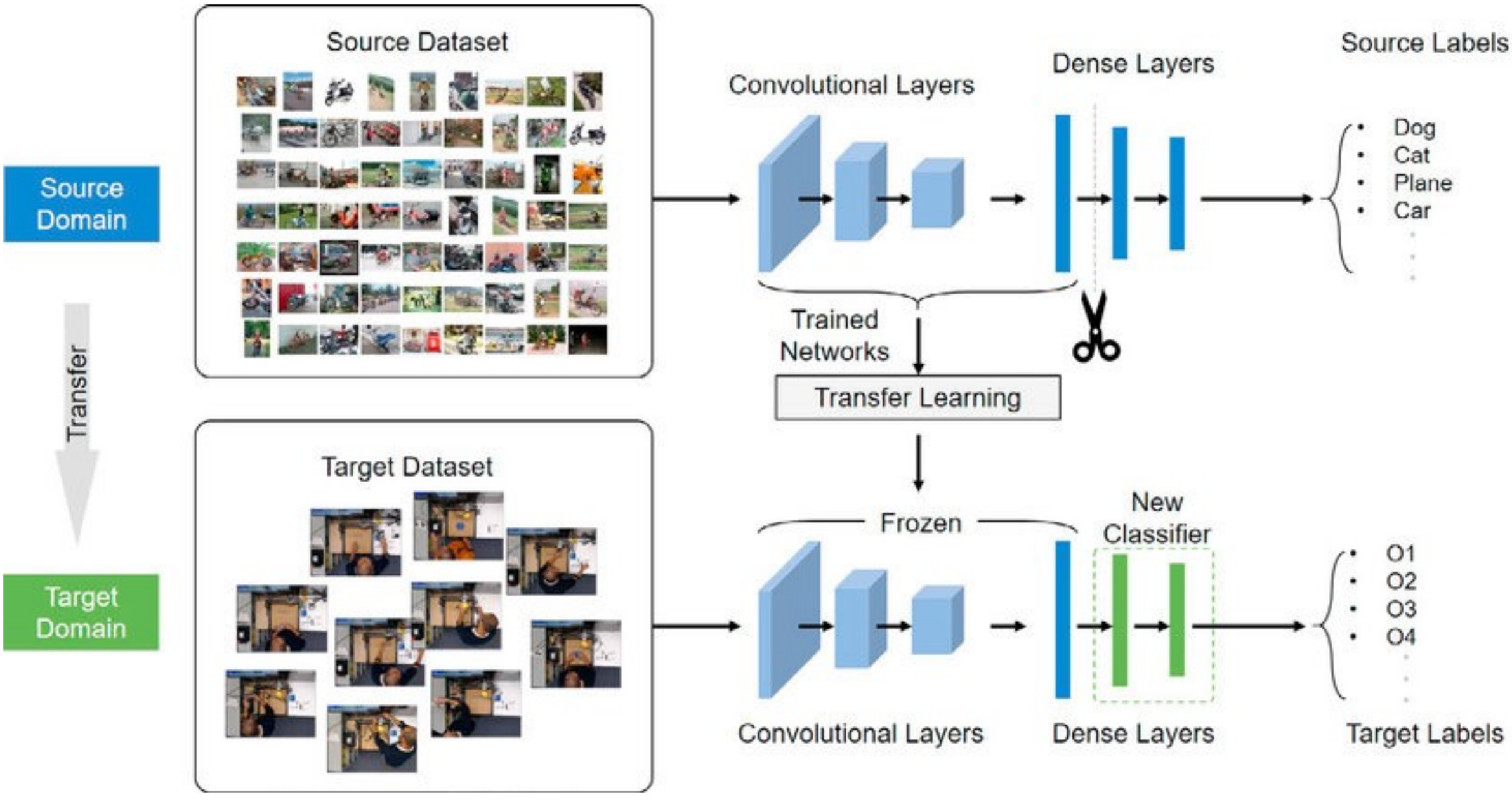
i.e. swap the Softmax layer at the end



3. If you have medium sized dataset, **"finetune"** instead: use the old weights as initialization, train the full network or only some of the higher layers

retrain bigger portion of the network, or even all of it.

Transfer Learning



“A Neural Algorithm of Artistic Style”, https://arxiv.org/pdf/1508.06576.pdf