Deep Learning

Lecture 3

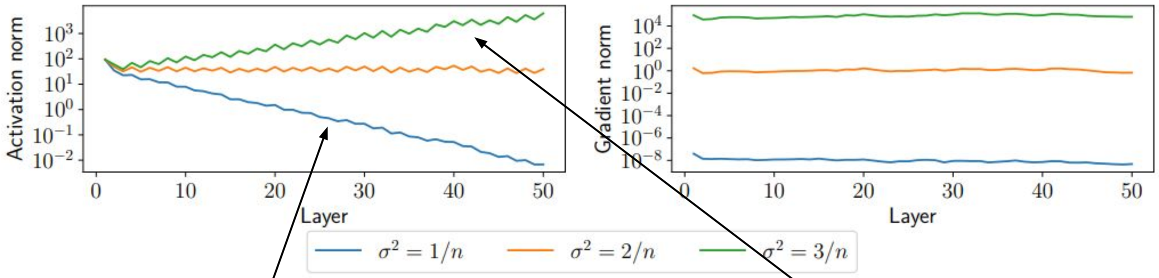
Recap

- Gradient descent for neural networks
  - SGD
  - Momentum
  - Adaptive learning rate
- Weight decay
  - L2 regularization in NN
  - Adam vs AdamW
- Dropout
  - Ensemble
  - Difference on train() and eval() modes

Contents

Motivation for weight initialization

- Suppose we have the following initialization: $w_i \sim \mathcal{N}\left(0, \frac{c}{n}\right)$
- Calculate 1) Activation norm, and 2) Gradient norm for each layer



$\sigma^2 = 3/n \Longrightarrow$ NaN

$\sigma^2 = 2/n \Longrightarrow$ Works

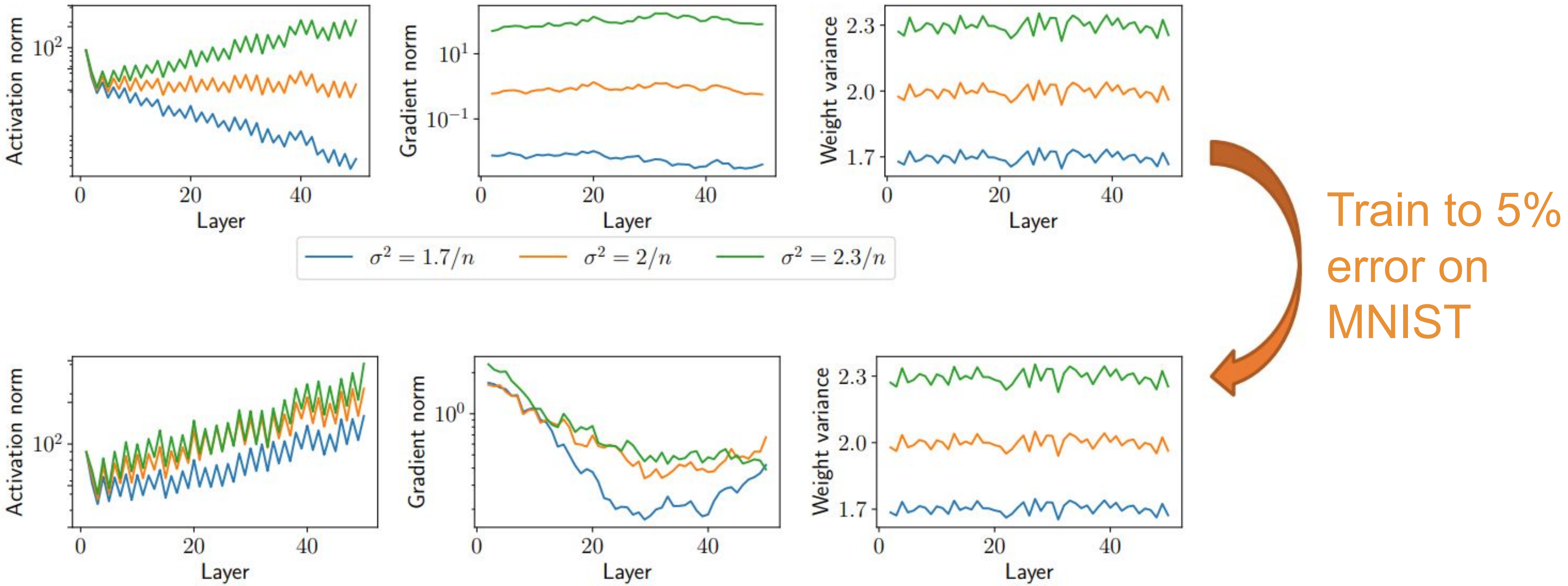
$\sigma^2 = 1/n \Longrightarrow$ No progress

Exploding Gradient

Vanishing Gradient

Weights after training

The problem is even more fundamental, however: even when trained successfully, the effects/scales present at initizalition **persist** throughout training



Weights Initialization

What kind of initialization can work?

- Zero initialization or constant initialization?*
    - Degeneration of a neural network into a single neuron

- Random initialization?
    - Does not ensure the avoidance of vanishing and exploding gradient

- More smart approaches?
    - Formalize the required neural network properties

*Why is it a bad idea? See this post

Idea

To prevent the gradients of the network's activations from vanishing or exploding, we will stick to the following rules of thumb:

1. The *mean* of the activations should be zero.
2. The *variance* of the activations should stay the same across every layer.

$$a^{l-1} = g^{l-1}(z^{l-1})$$

$$z^l = W^l a^{l-1} + b^l$$

$$a^l = g^l(z^l)$$

Desired condition:

$$\mathbb{E}(a^{l-1}) = \mathbb{E}(a^l)$$

$$Var(a^{l-1}) = Var(a^l)$$

https://www.deeplearning.ai/ai-notes/initialization/index.html

Xavier and He initialization

Xavier initialization (for sigmoid and tanh)

$$W^{[l]} \sim \mathcal{N}\left(\mu = 0, \sigma^2 = \frac{1}{n^{[l-1]}}\right) \qquad b^{[l]} = 0$$

He initialization (for ReLU)

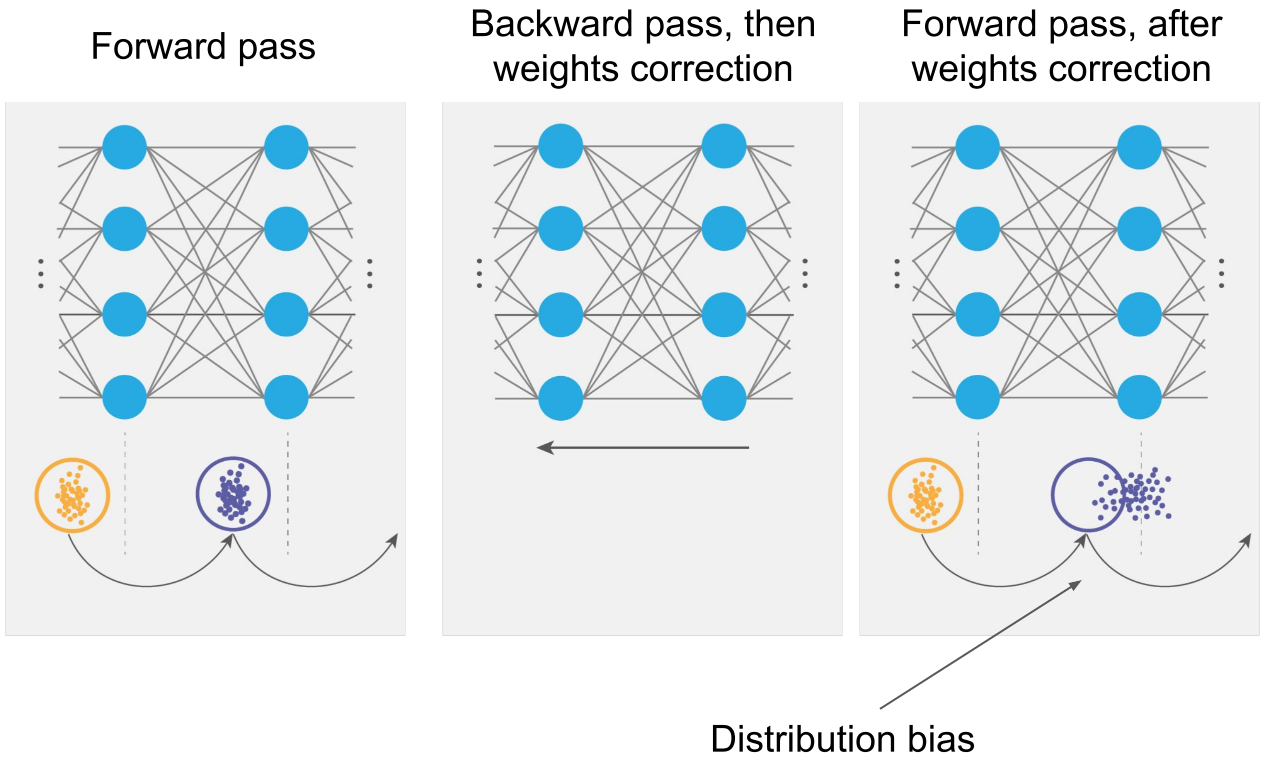
$$W^{[l]} \sim \mathcal{N}\left(\mu = 0, \sigma^2 = \frac{2}{n^{[l-1]}}\right) \qquad b^{[l]} = 0$$

$W^{[l]}$ – weights

$b^{[l]}$ – bias

$n^{[l-1]}$ – hidden size

Internal Covariate Shift

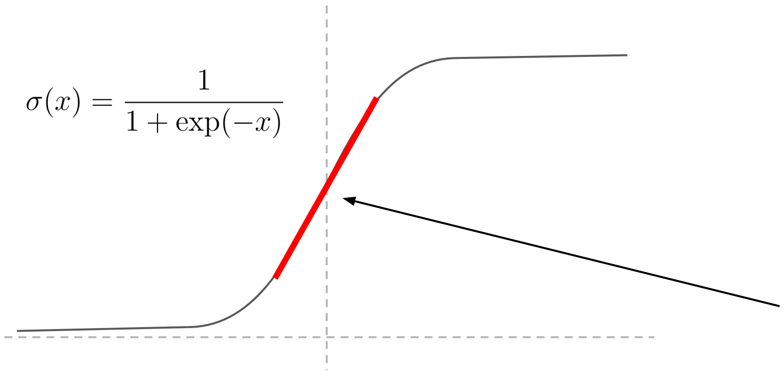


Naive approach

- Simply normalize each feature (each neuron output), using the mean and variance from batch:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sigma_B + \varepsilon}$$

- After normalizing, an activation function seems to be linear

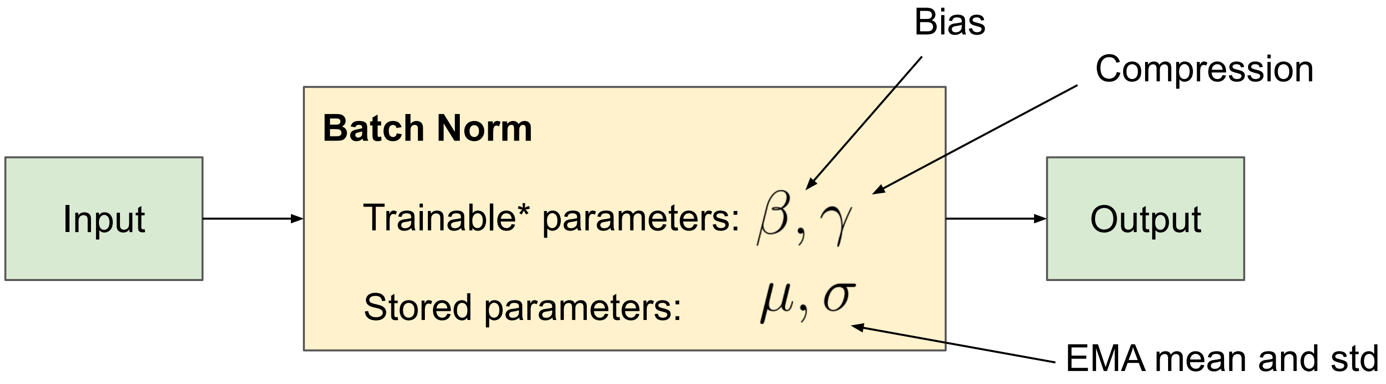


- Thus, we obtain just a linear one-layer network…

Batch Normalization

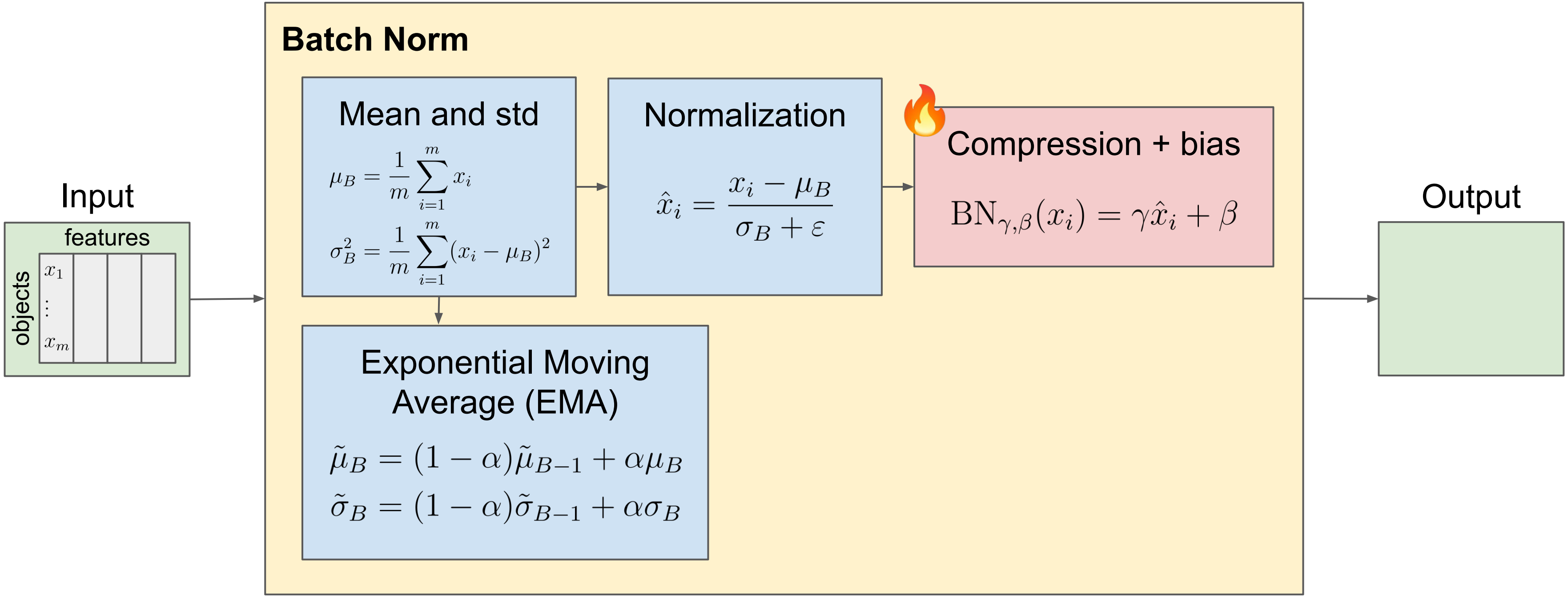
- We can give the neural network the ability to move the distribution of layer outputs:
  - Bias
  - Compression

  Trainable parameters!
- During the **training stage**, we calculate statistics on the batch
- How to make a prediction on a single object from **test**?
  - Calculate mean and variance on the entire dataset? Not enough memory…
  - Estimate them? Exponential Moving Average (EMA)!



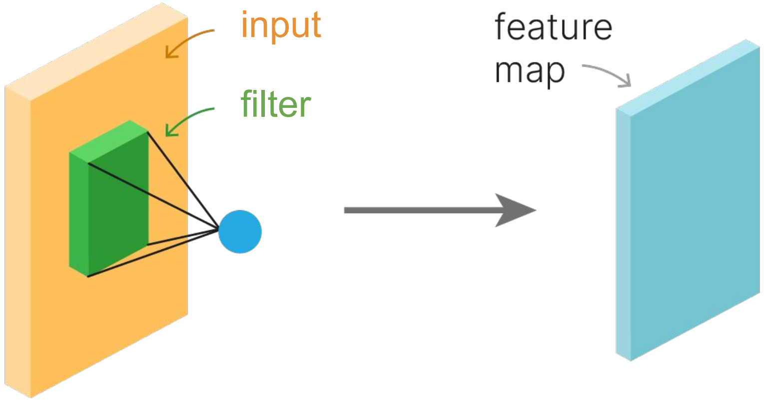
*How to calculate the gradient? See this post on Kevin Zakka’s Blog

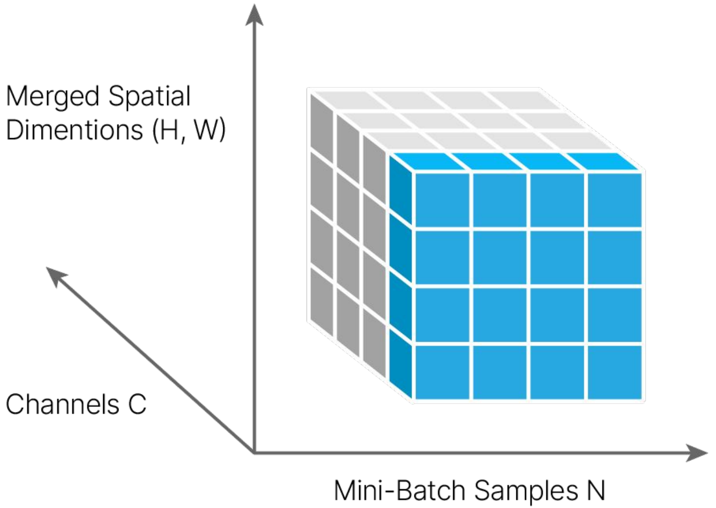
Batch Normalization: scheme



Batch Normalization: visualization

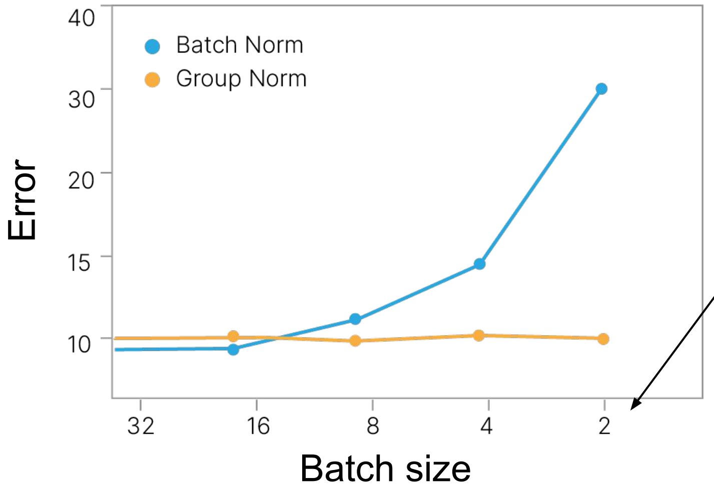
- In case of the feature maps, Batch Norm is applied **for each channel** (like feature)





Batch Normalization: Practical Recommendations

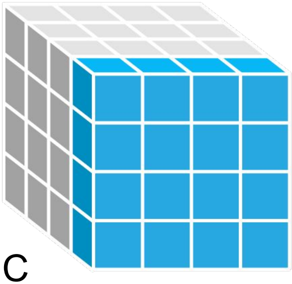
- Shuffle objects between epochs when using Batch Norm to ensure diverse batches for training parameters
- Remove bias in the layer following Batch Norm, as the bias parameter in Batch Norm takes on this role.
- Use before activation function
- Smaller batch sizes result in poorer performance of Batch Norm



Small batch size!

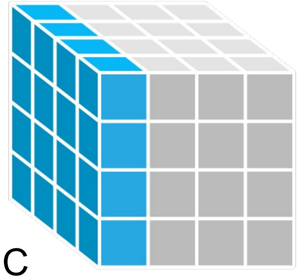
Other Normalizations

Batch Norm



Classical architectures

Layer Norm



Modern architectures
(e.g. transformers)

Instance Norm

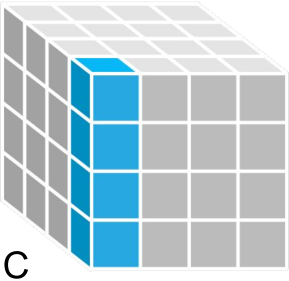
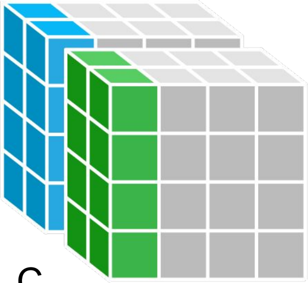


Image generation
(contrast balance)

Group Norm



Used sometimes…

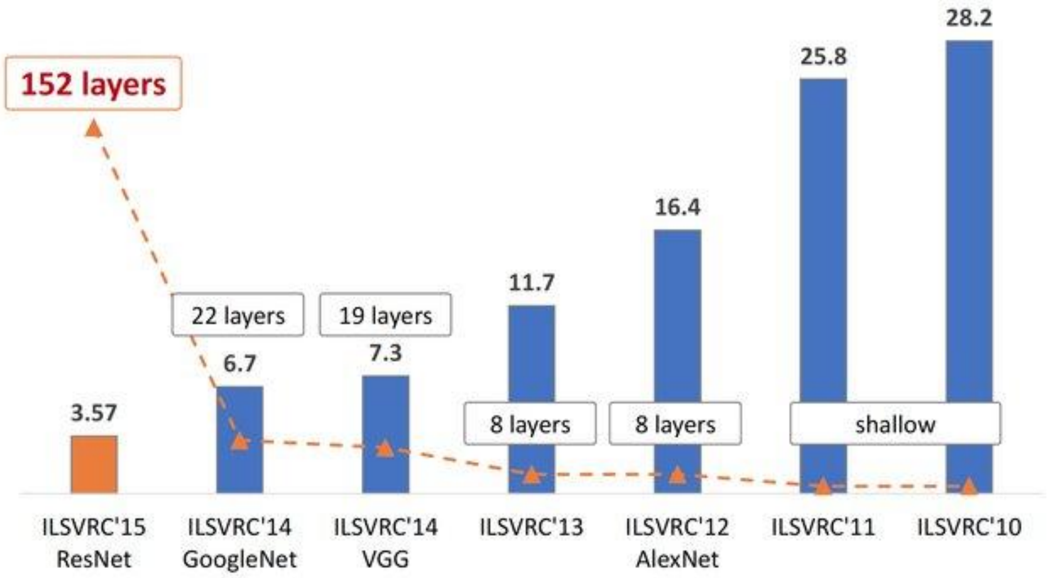
Convolutional NN

ImageNet

Everything starts with data!

- The ImageNet dataset contains 14,197,122 annotated images.
- Total number of non-empty WordNet synsets: 21,841.
- Since 2010 the dataset is used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a benchmark in image classification and object detection.

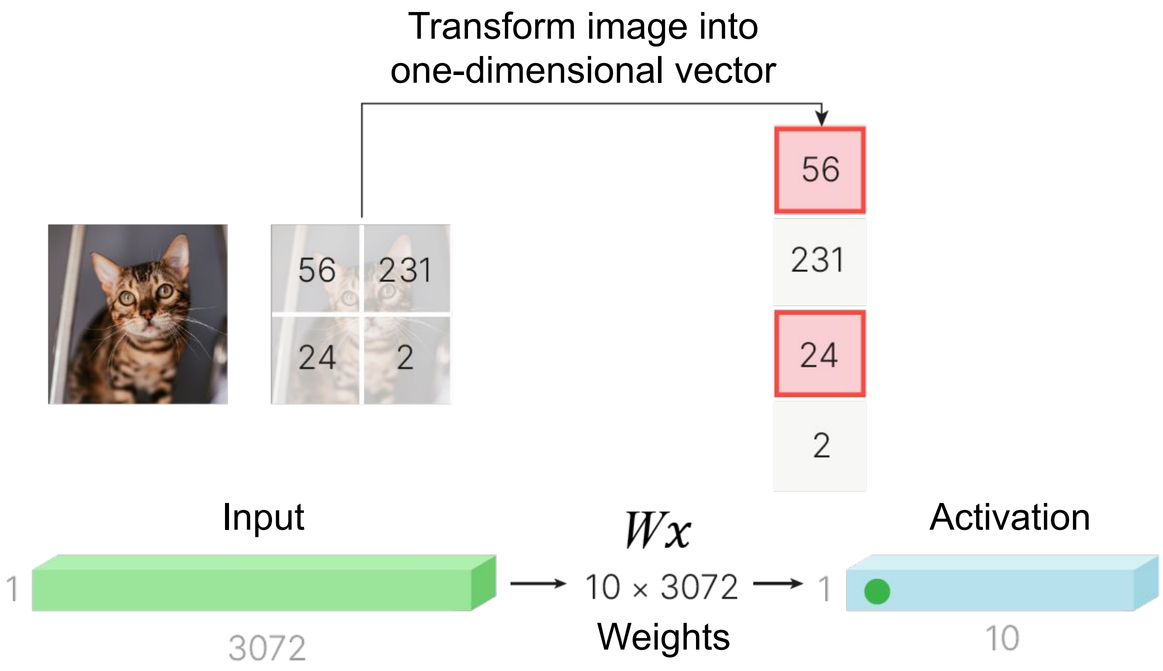
ImageNet Results

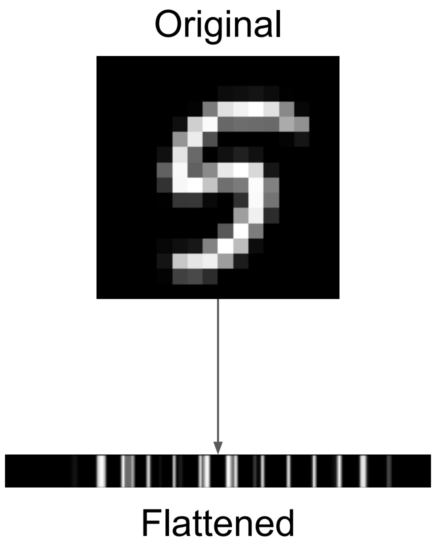


Why we can't use MLP for images?

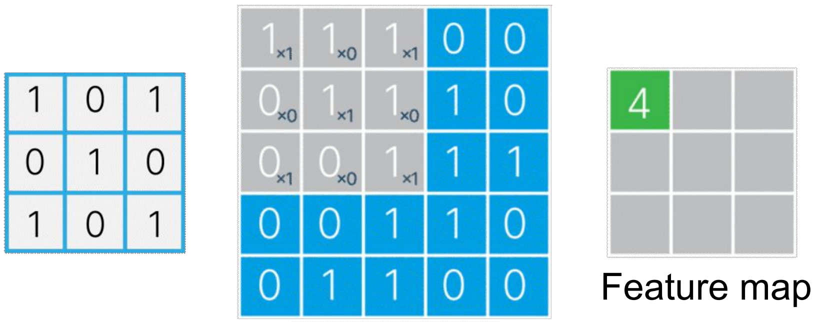
- Too many parameters
- Fixed dimension of images
- Features will be too correlated

Inherent problem of MLP





Sliding window (filter)



Image

Original

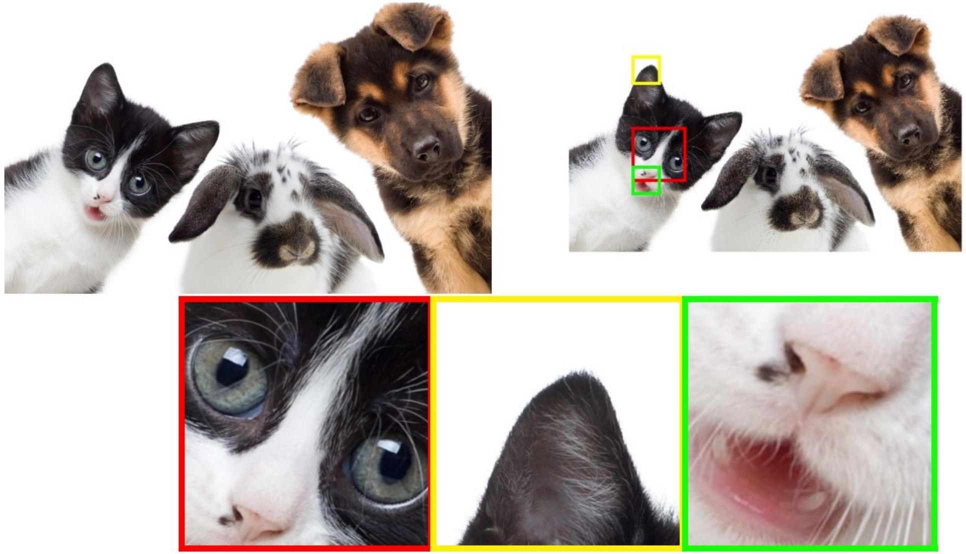
Gaussian

Sobel

*More info about Gaussian filter and Sobel Edge Detector

Motivation for convolution

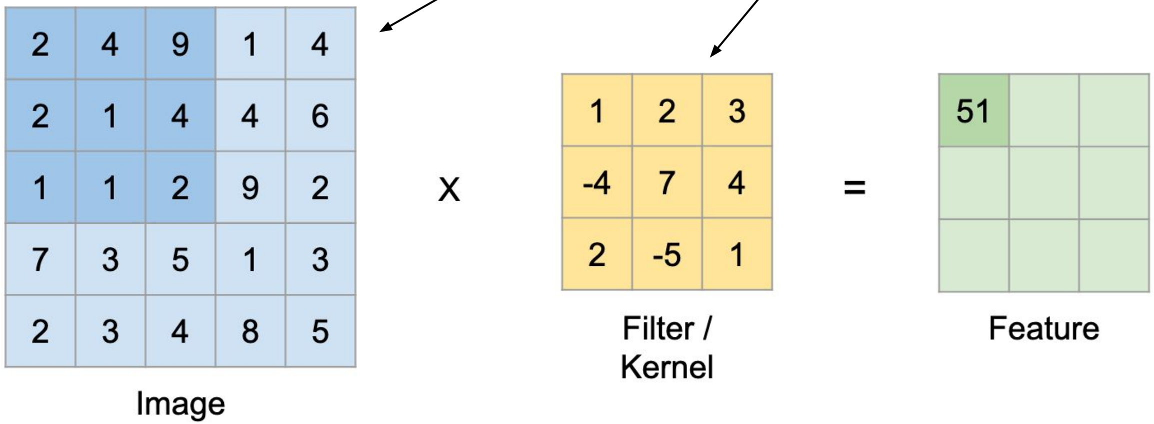
- To perform identification, we are seeking for noticeable parts of an object
- However, these parts appear at different locations
- Want to perform a feature search over the whole picture
- **Idea:** Let the neural network choose the filters by itself



Convolution

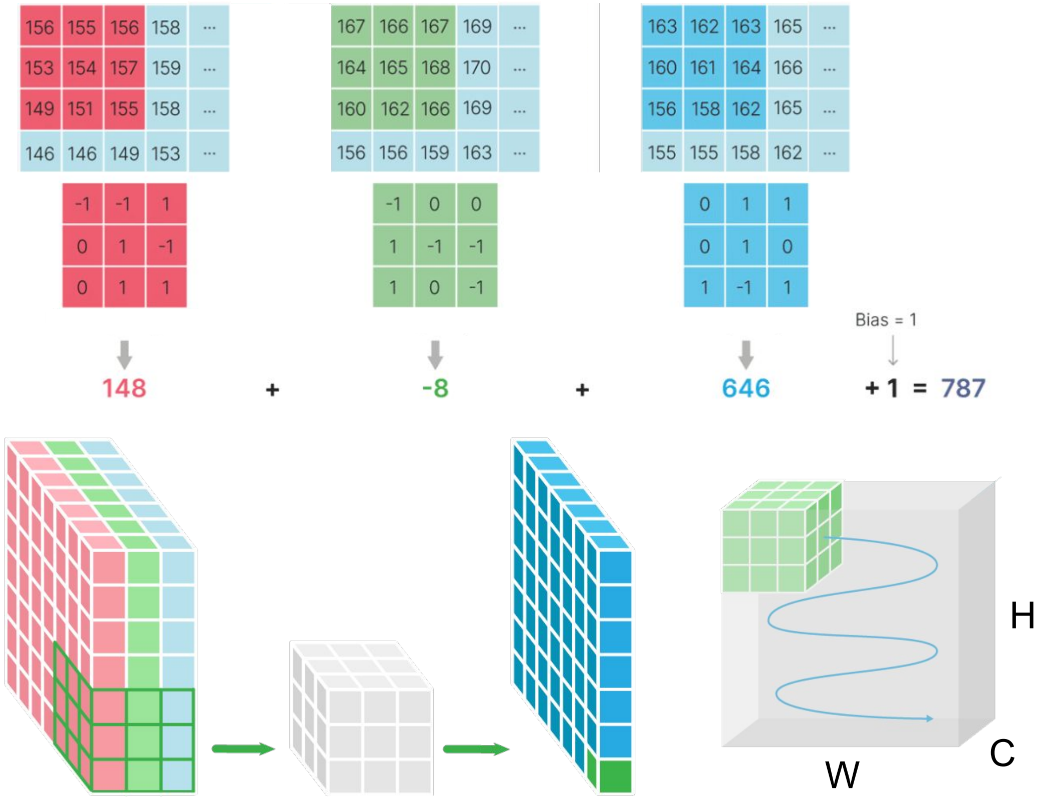
Mathematically, 2D convolution* can be written in the following form

$$Y(i,j) = \sum_{u,v} X(i+u, j+v) K(u,v)$$



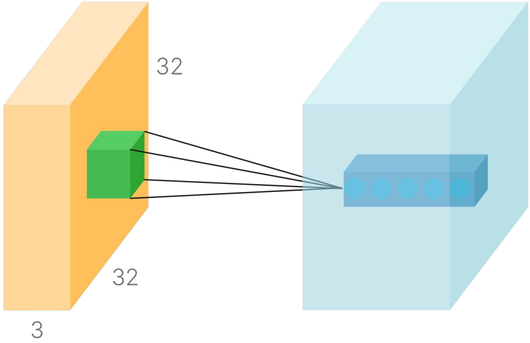
*How to calculate the gradient? See [this post](#) on Pavithra Solai Blog

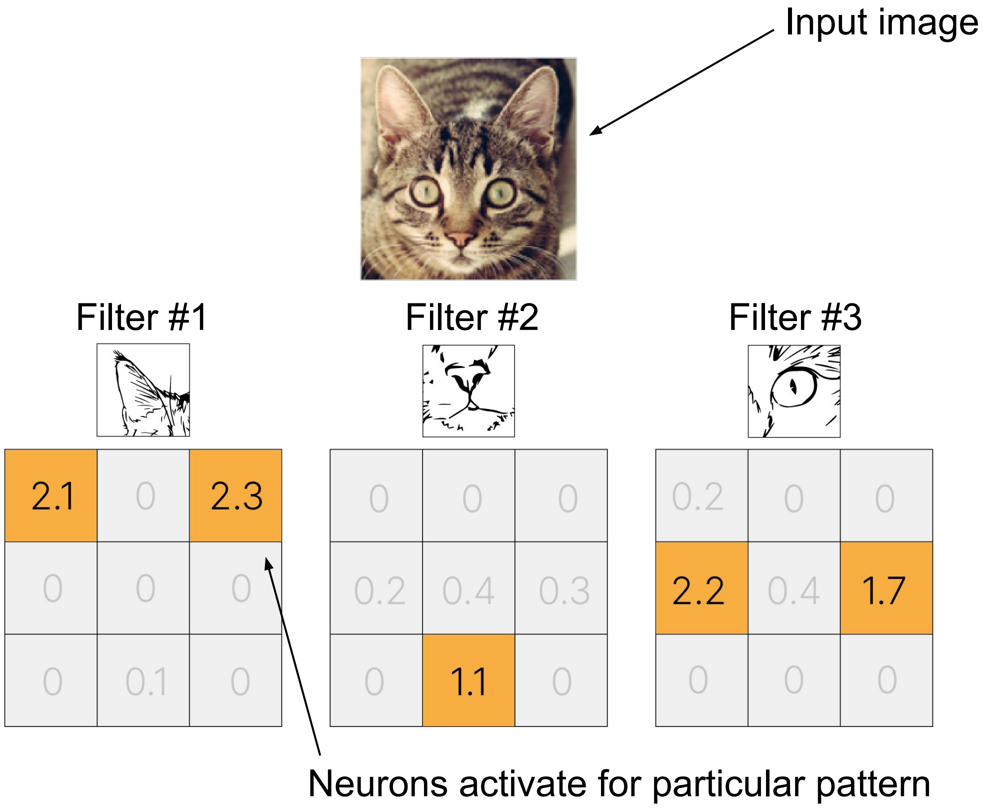
Color images / multiple input channels



Multiple output channels

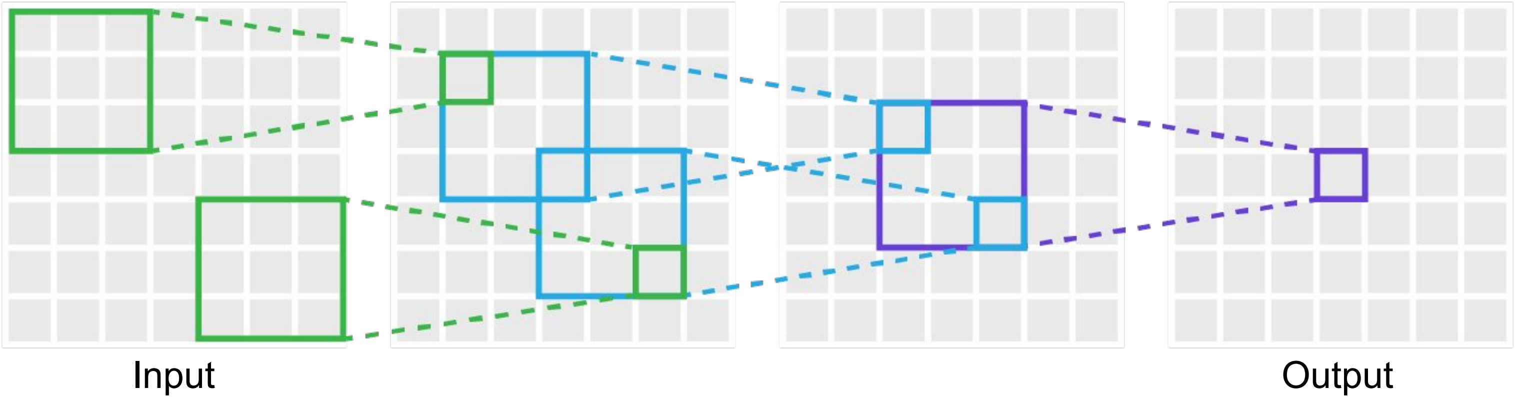
- In fully-connected neural networks, each neuron activates for particular pattern (red car, blue chair, etc.)
- We want CNN neurons to do the same, e.g. ear, nose, eye
- Thus, for **each pattern** we should have a neuron, i.e. a **filter**



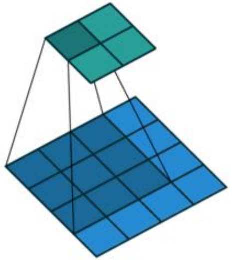


Receptive field

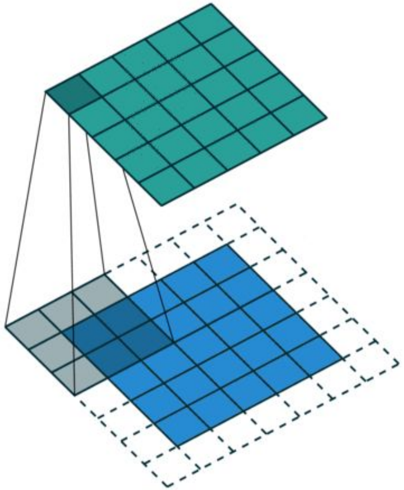
- Usually complex features consist of less complex features: e.g. a cat has a face, the face has ears, the ears have edges
- CNN layers, close to the:
    - Input → detect less complex features
    - Output → detect more complex features, combining previous ones
- Receptive field is the input image region that a particular neuron is “looking at”



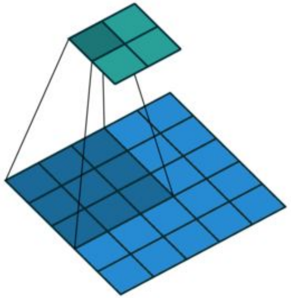
How to save dimension / increase receptive field?



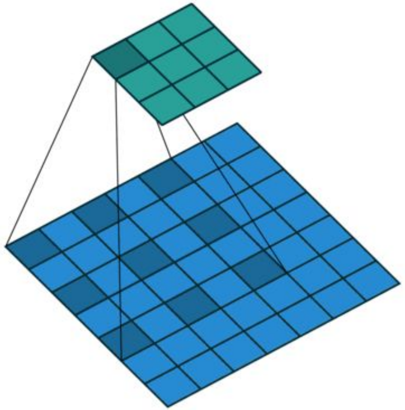
No padding,
no strides



Padding 1,
no strides

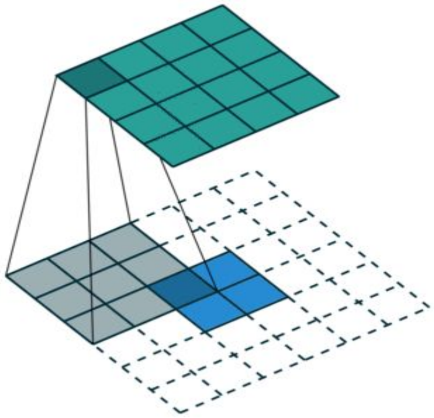


No padding,
stride = 2

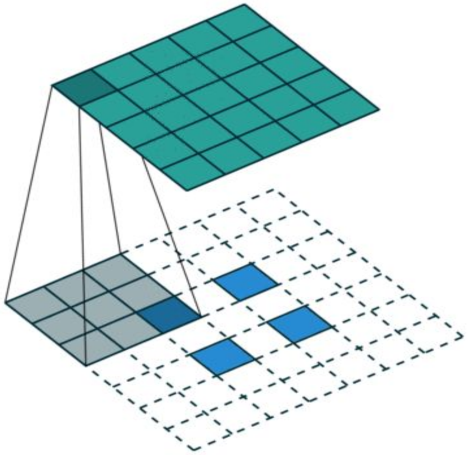


No padding,
no stride,
dilation = 2

Upsampling with convolutional layer

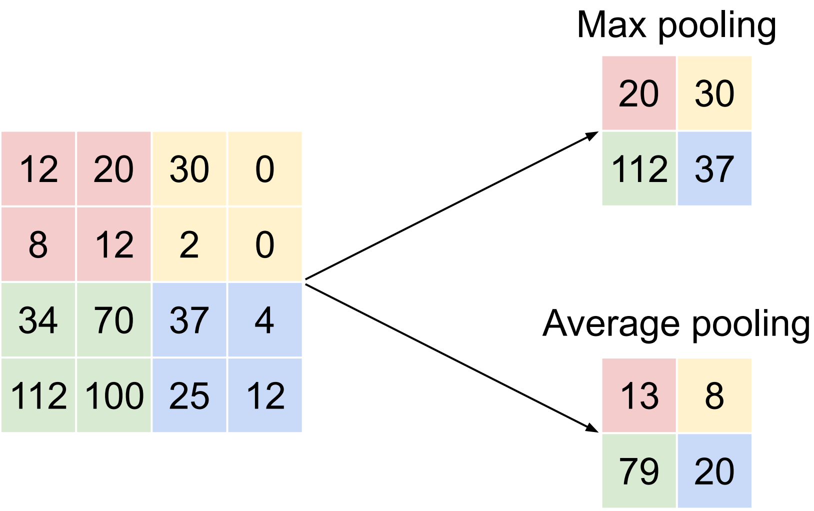


Transposed convolution



Transposed convolution
stride = 2

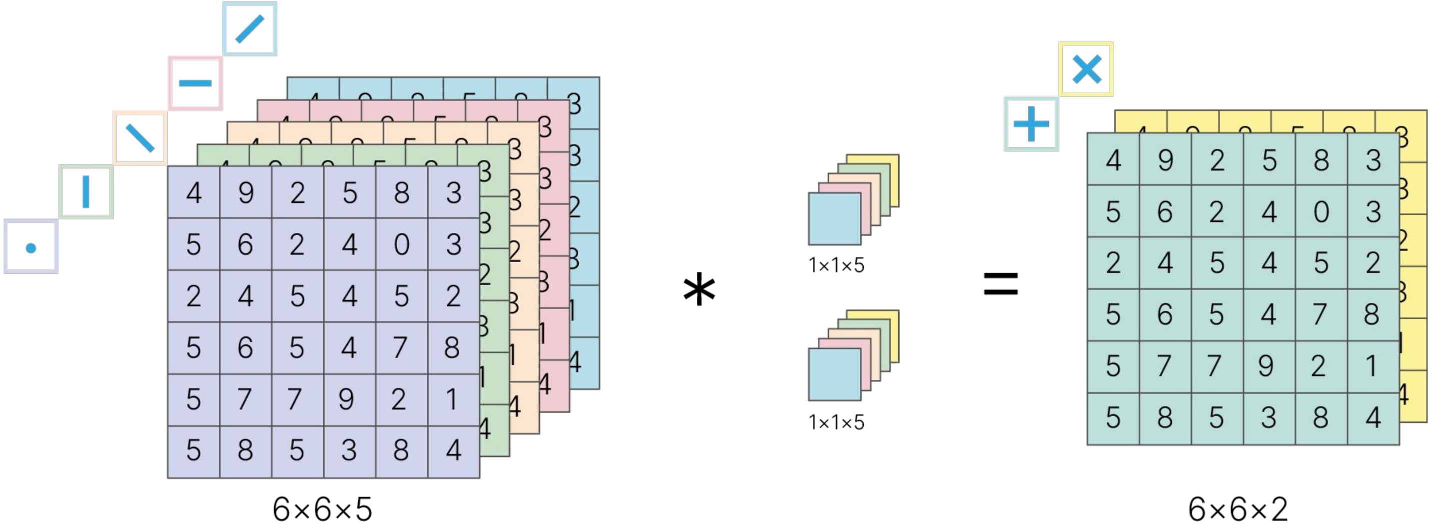
Pooling



- Usual motivation: adding invariance to small shifts
- Several max-poolings can accumulate invariance to stronger shifts
- Pooling is applied for each channel, so the total number of channels remains unchanged

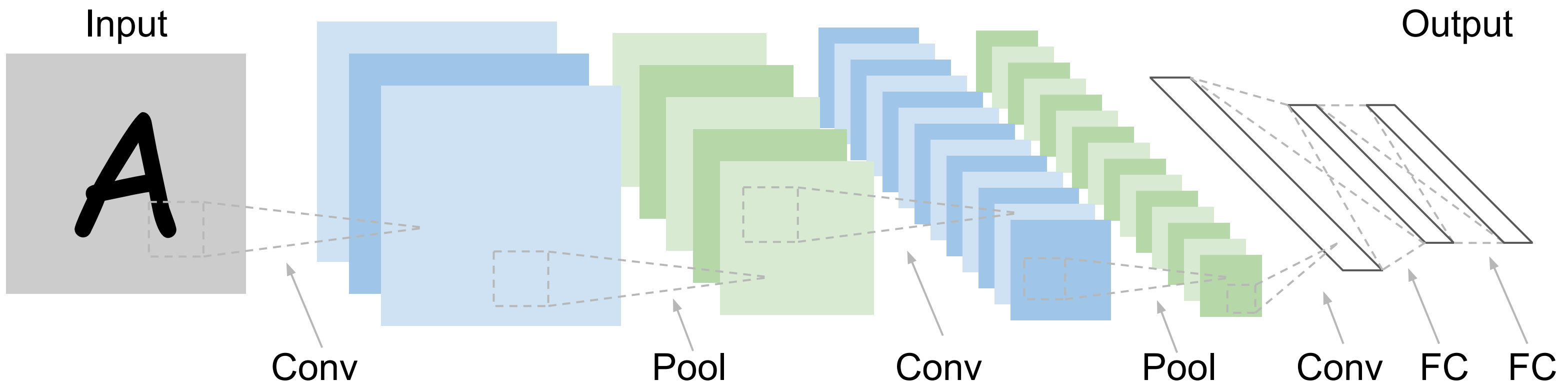
Convolution 1x1

- Reducing the number of channels leads to a generalization of features
- It is often obtained through the 1x1 convolutional layer
- Example: RGB → Grayscale image, i.e. $\mathrm{Brightness} = \frac{1}{3}R + \frac{1}{3}G + \frac{1}{3}B$



LeNet: example of CNN

- In the 1989, Yann LeCun et al. introduced the LeNet-5 neural network
- Character recognition: handwriting and machine printing
- Seven layers: 3 convolutional, 2 pooling, 2 fully-connected
- Activation function: tanh



Recap

- Vanishing and exploding gradient
- Weights initialization
- Internal Covariate Shift
- Normalization
  - Batch Norm
  - Layer Norm
  - Instance Norm
  - Group Norm
- Convolutional NN

TIME FOR A BREAK