CS M146: Introduction to Machine Learning
Neural Networks – Deep Learning

Aditya Grover

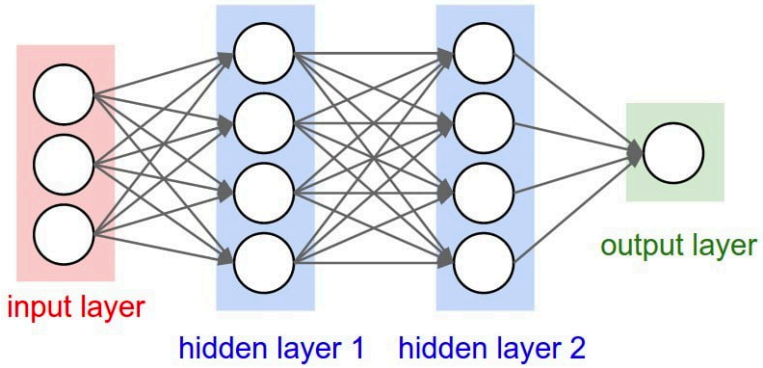
https://aditya-grover.github.io/ @adityagrover_

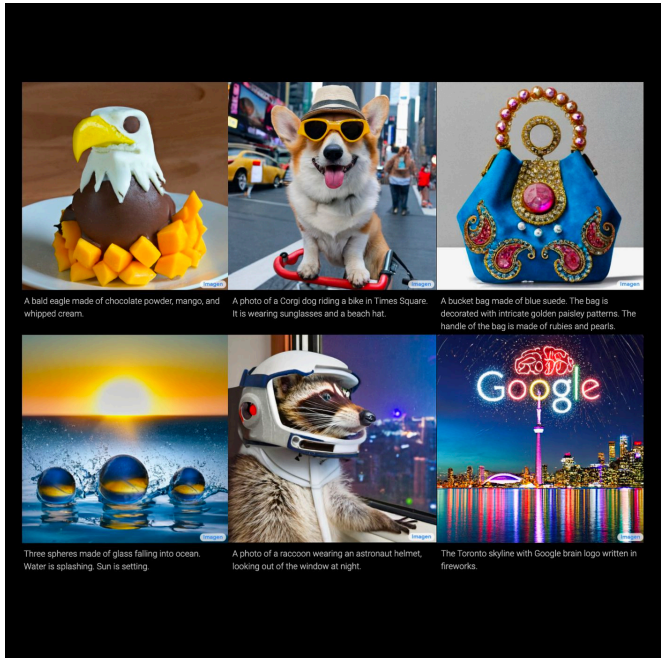
Deep Neural Networks

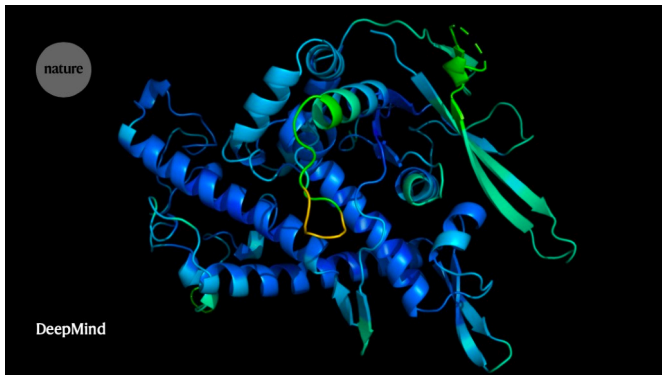
2018 Turing Award

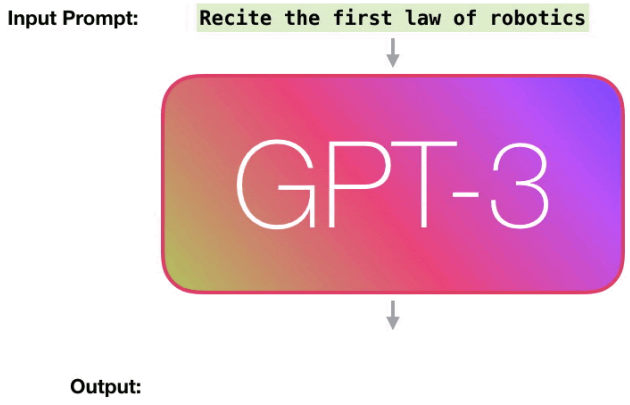
“For for conceptual and engineering breakthroughs that have made **deep neural networks** a critical component of computing”

Success Stories of Deep Neural Nets





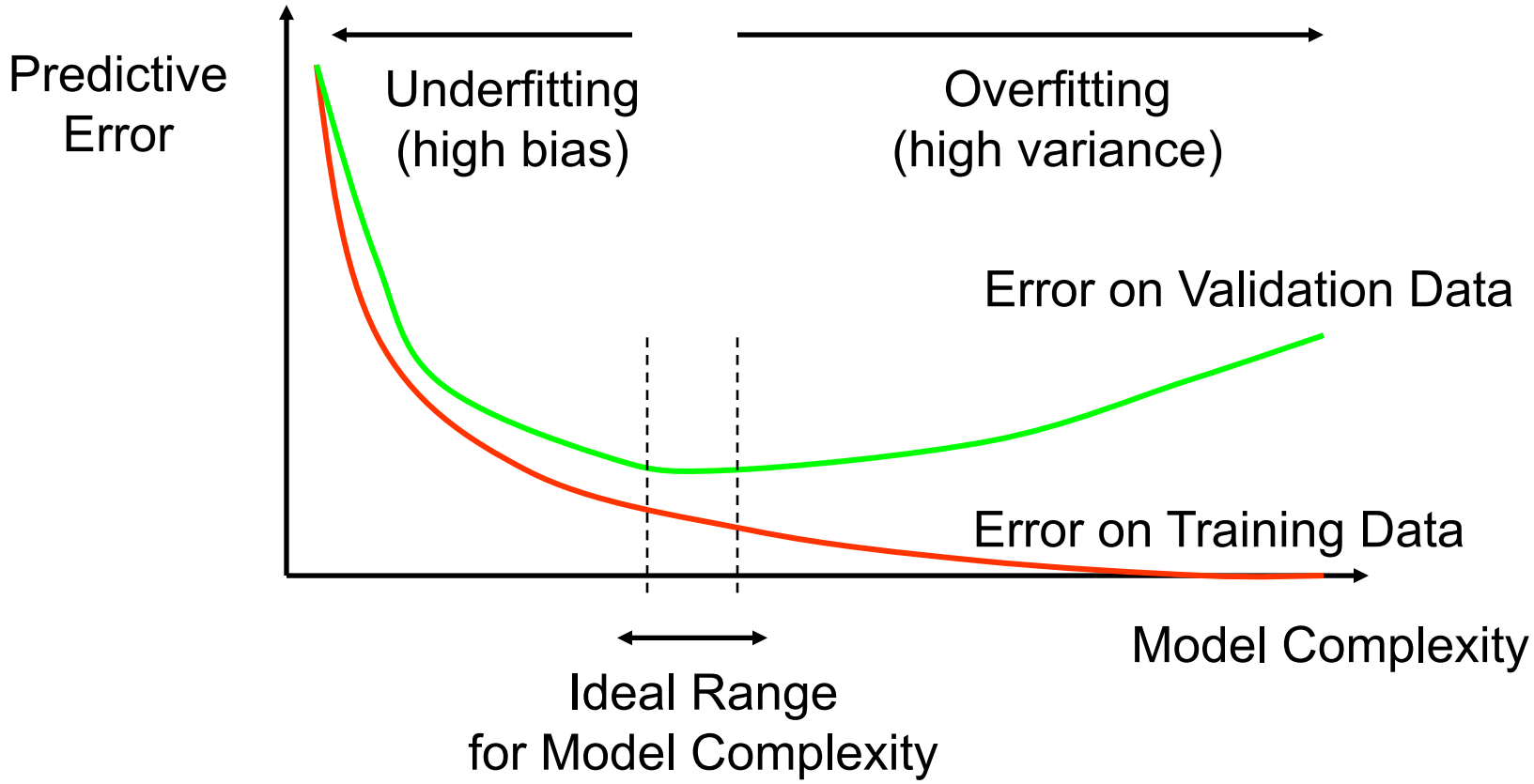




AND MANY MORE!

Rethinking Model Complexity

- Bias-Variance Tradeoff: More complex ‘hypothesis’ have low bias but can suffer from high variance



How To Measure Model Complexity?

- Number of unknown parameters (i.e., dimension of $\boldsymbol{\theta}$) is misleading
  - **Case 1:** $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$

    Number of unknowns: $d + 1$
  - **Case 2:** $h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{x}) = (u_0^2 - v_0^2) + (u_1^2 - v_1^2)x_1 + \cdots + \left(u_d^2 - v_d^2\right)x_d$

    Number of unknowns: $2d + 2$
  - Same complexity, different number of parameters
- Many other notions of complexity in **learning theory**
  - E.g., VC dimension, minimum description length, etc.
- Open question with ongoing research: What notion of modern complexity explains the mysterious success of **deep neural nets**?

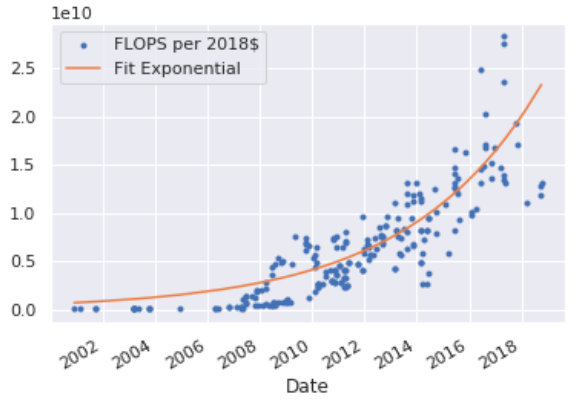
Deep Neural Nets

- Deep neural nets (DNN) is an informal term for any NN architecture with **many** hidden layers trained via **gradient methods**
  - Computer vision and NLP: 10-1000 layers (currently)
  - Reinforcement learning: 2-20 layers (currently)
- Classic notions of complexity fail to explain their success
  - Modern deep neural nets have billions of unknown parameters
  - Number of parameters >>> Number of datapoints
  - Yet, they do not overfit and generalize very well

Why Do Deep Neural Nets Work So Well?

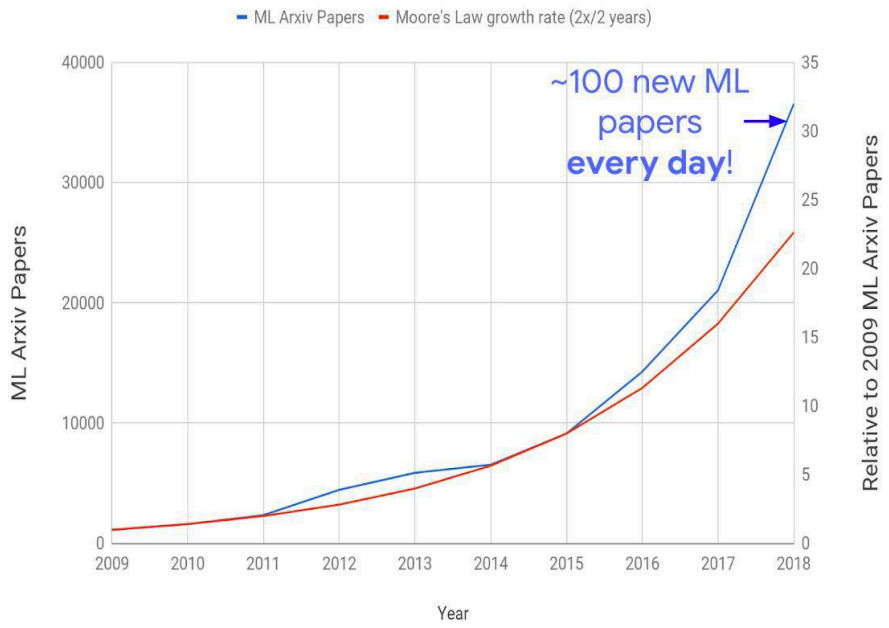
- 3 major secret sauce ingredients: **Data**, **compute**, **algorithms**
  - Sensor proliferation have contributed to huge datasets

  - Powerful computation, especially in GPU computing



Why Do Deep Neural Nets Work So Well?

- Algorithmic advancements: Address optimization (training) and generalization (testing) of DNNs in roughly 3 spheres
  - Architectures
  - Regularizers
  - Optimizers

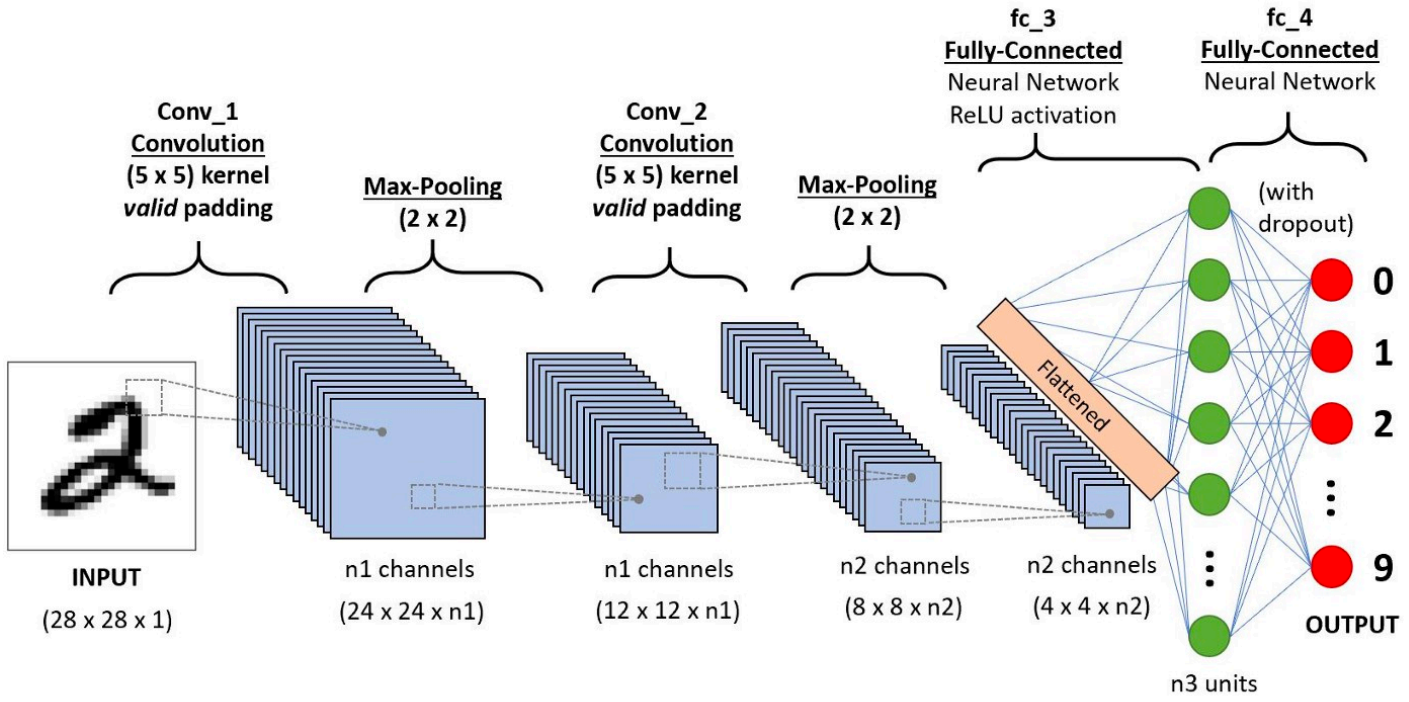


Neural Net Architectures

Architectures for Deep Neural Nets

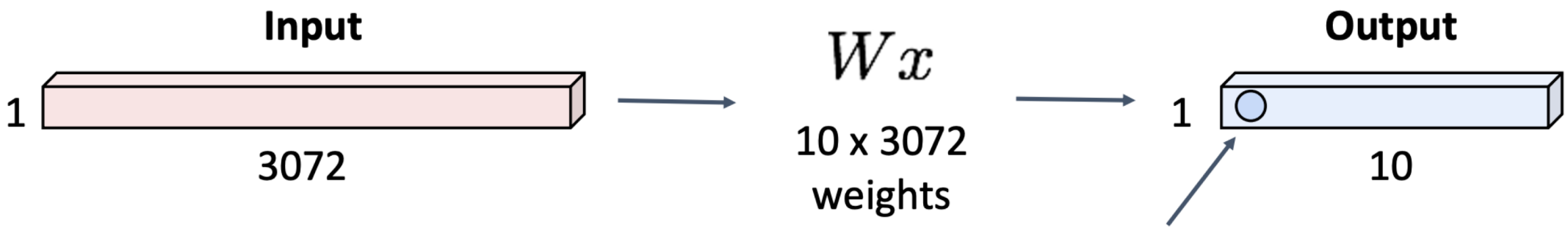
- Early architectures for DNNs were based on multi-layer perceptrons with sigmoid/tanh non-linearities
- Modern DNN architectures
  - Better connectivity structure:
    - E.g., use of convolutions for translation invariance in object recognition
    - Many architecture variants for many modalities: convolutional NN, recurrent NN, graph NN, transformers
  - Better backpropagation: Improve backpropogation of gradients for very deep nets
    - Vanishing gradients: magnitude of gradient is too low
    - Exploding gradients: magnitude of gradient is too high
    - Can be done by e.g., using alternate activation functions, normalizing activations

Convolutional Neural Networks



Fully-Connected Layer

32x32x3 image -> stretch to 3072 x 1

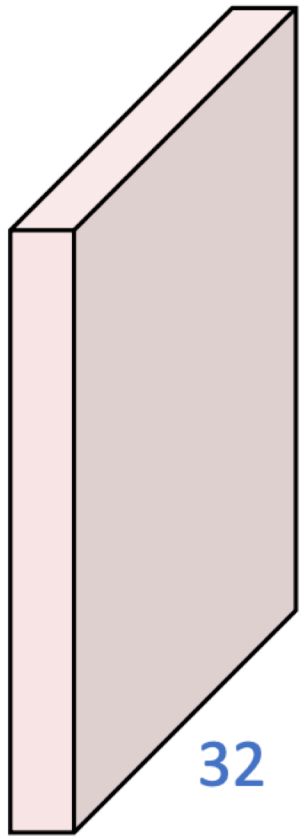


1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

Convolution Layer

Filters always extend the full depth of the input volume

3x32x32 image

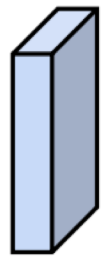


32 height

32 width

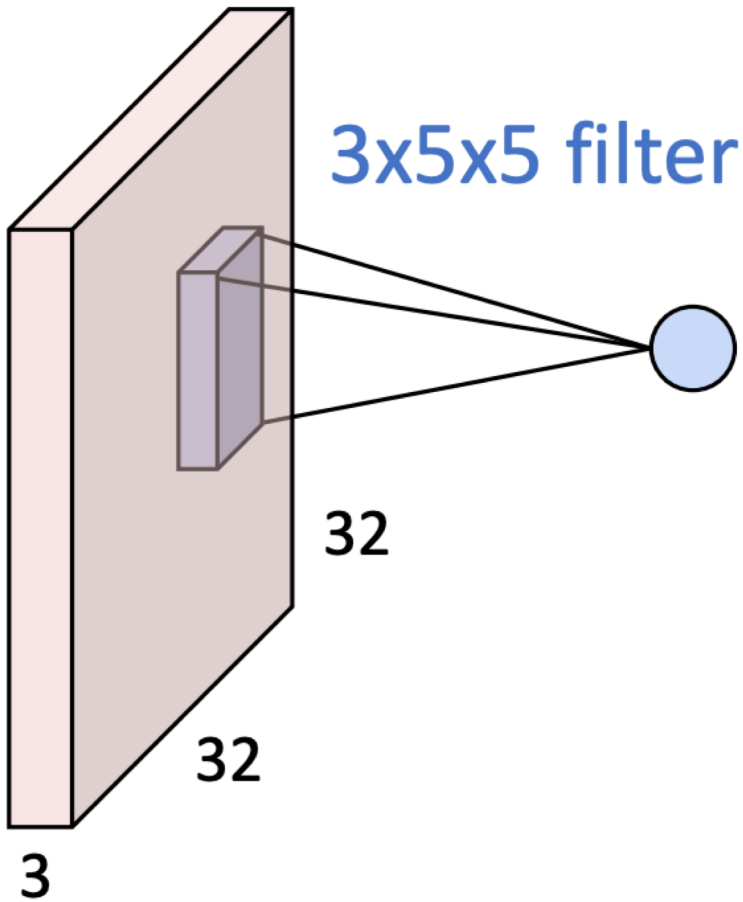
3 depth / channels

3x5x5 filter



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

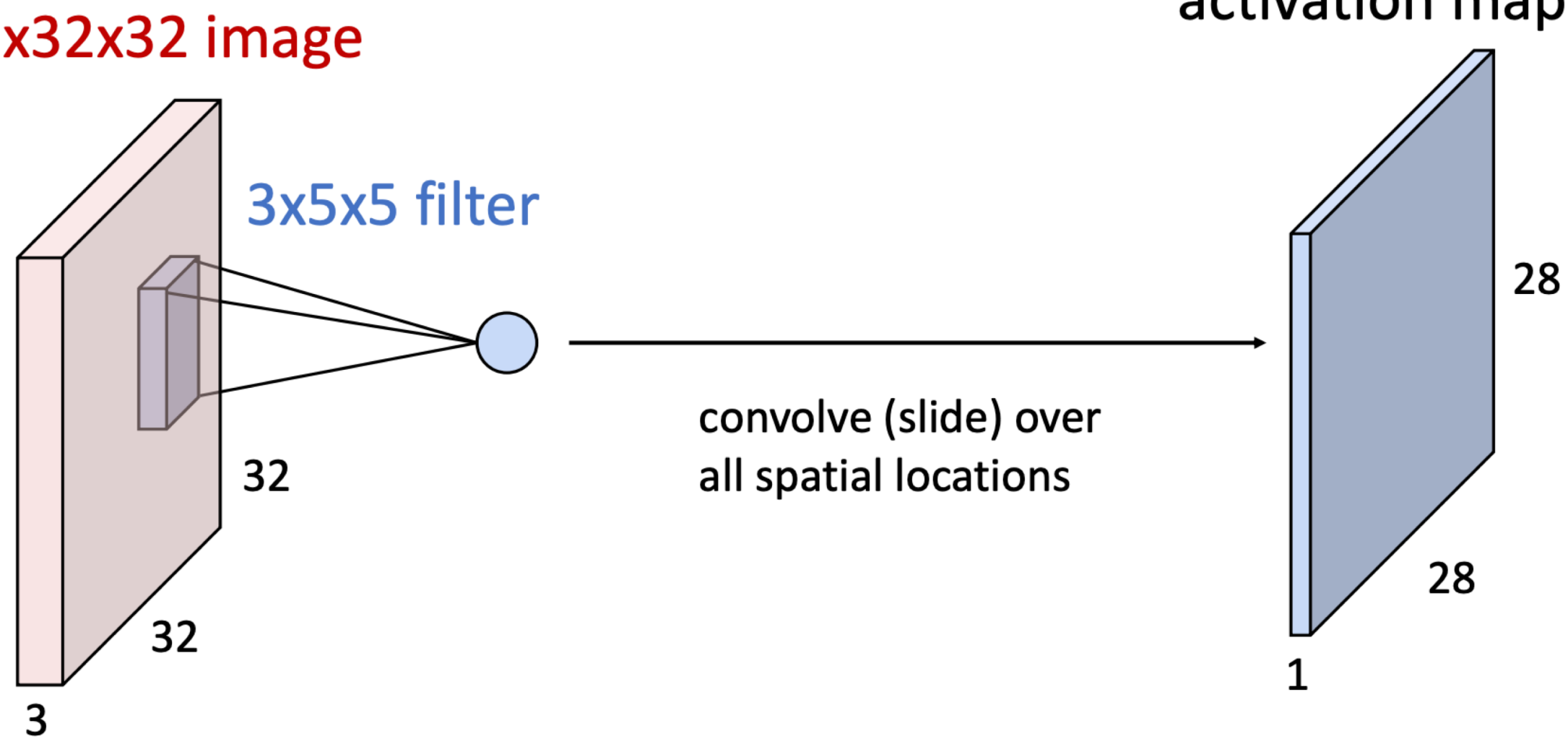
Convolution Layer



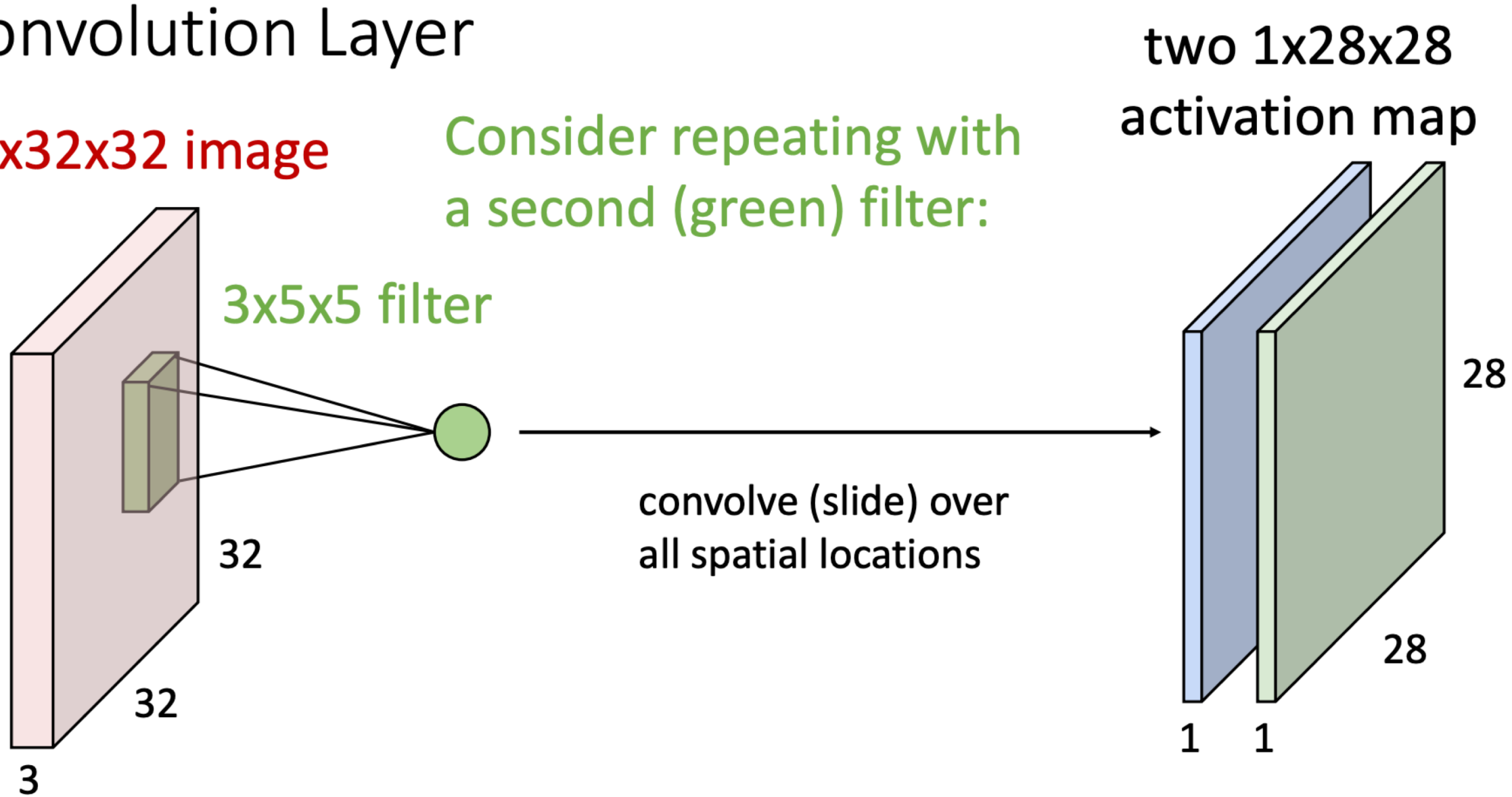
1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

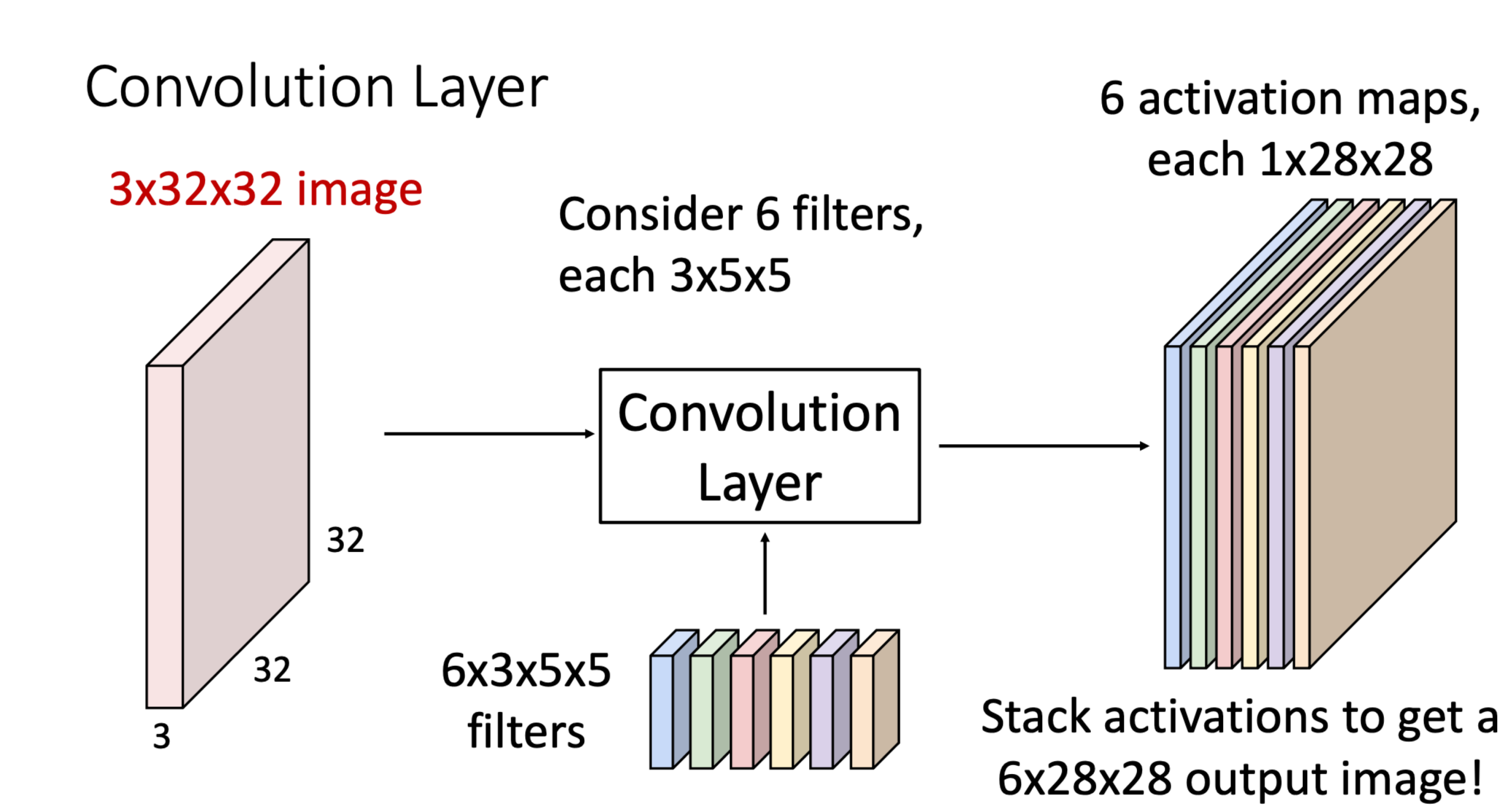
Convolution Layer



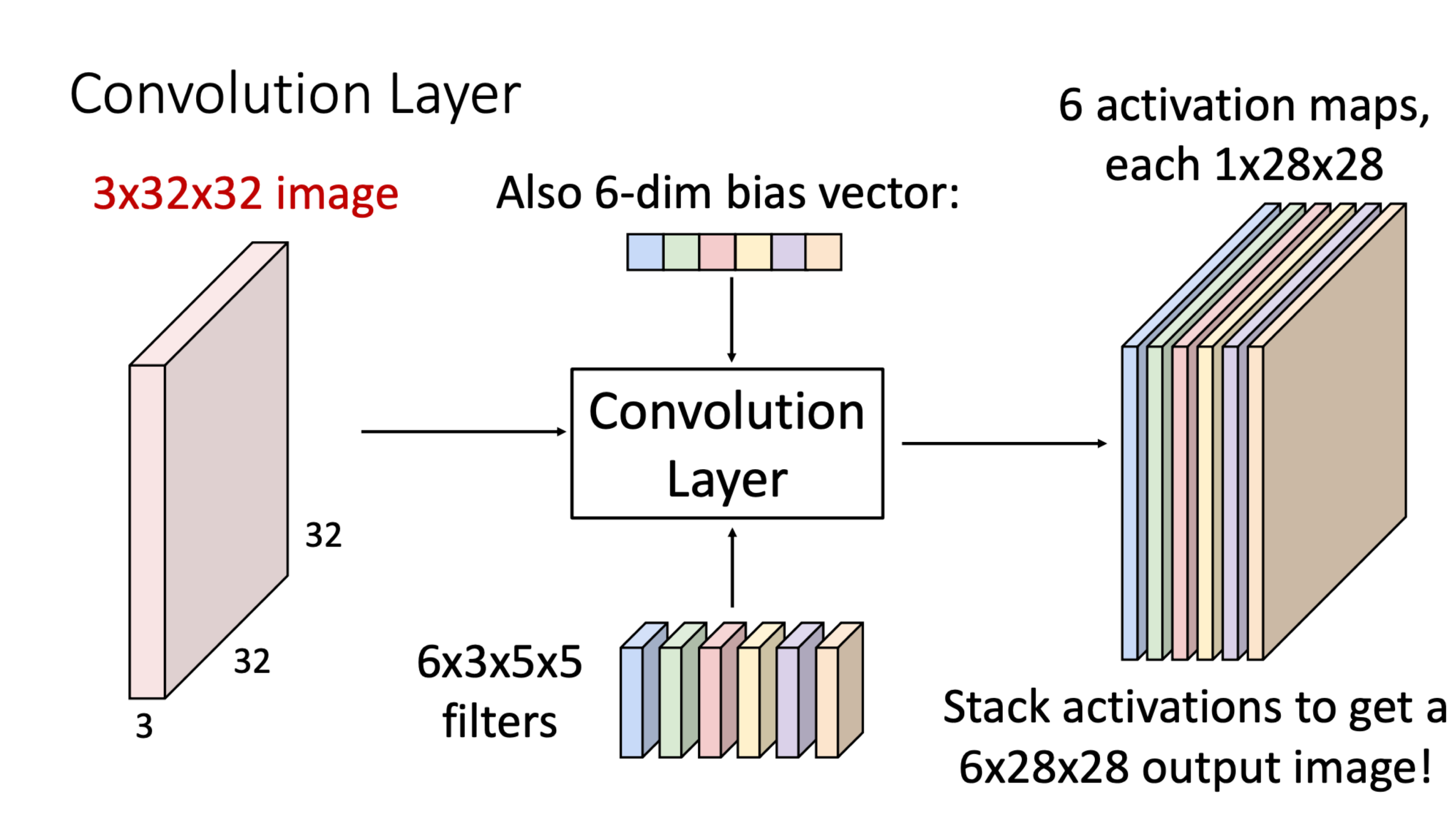
Convolution Layer



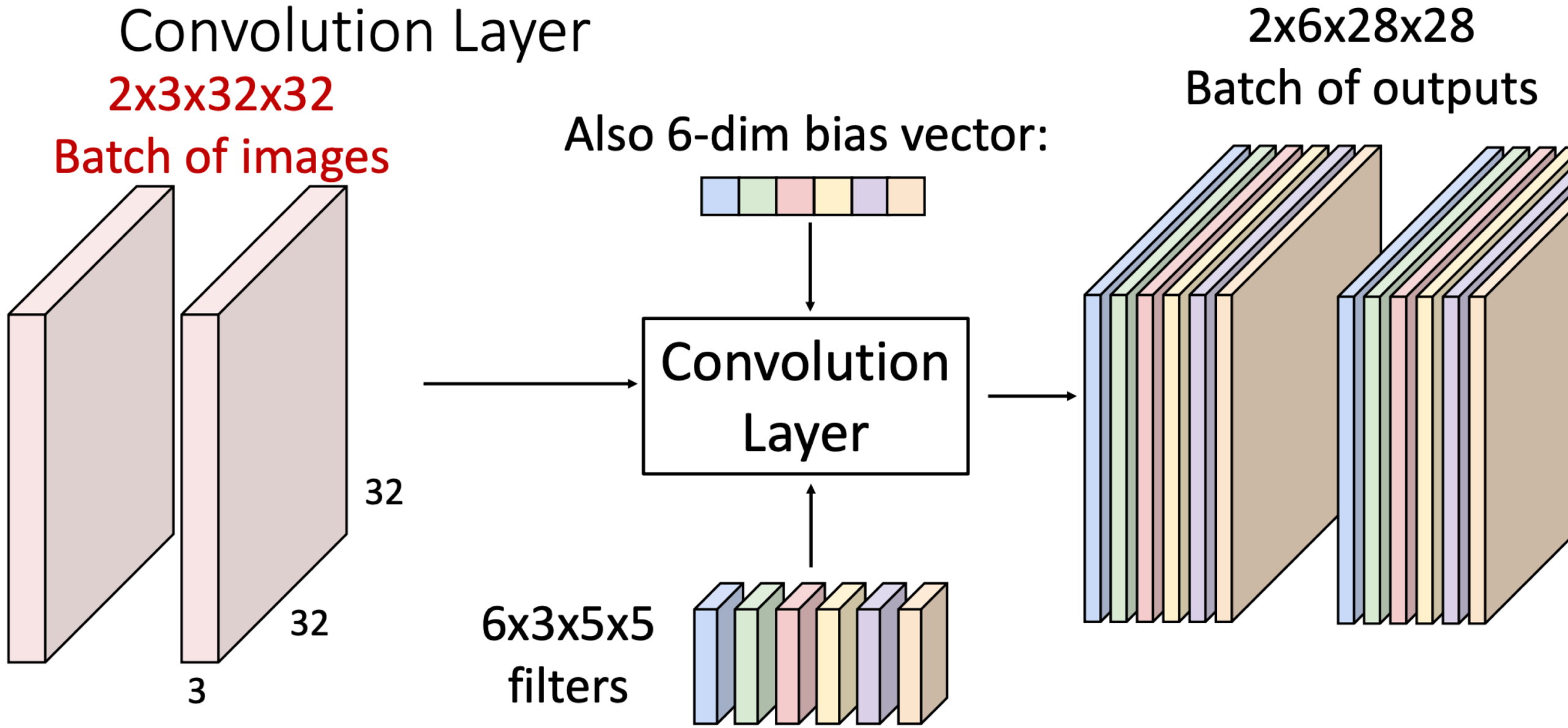
Convolution Layer



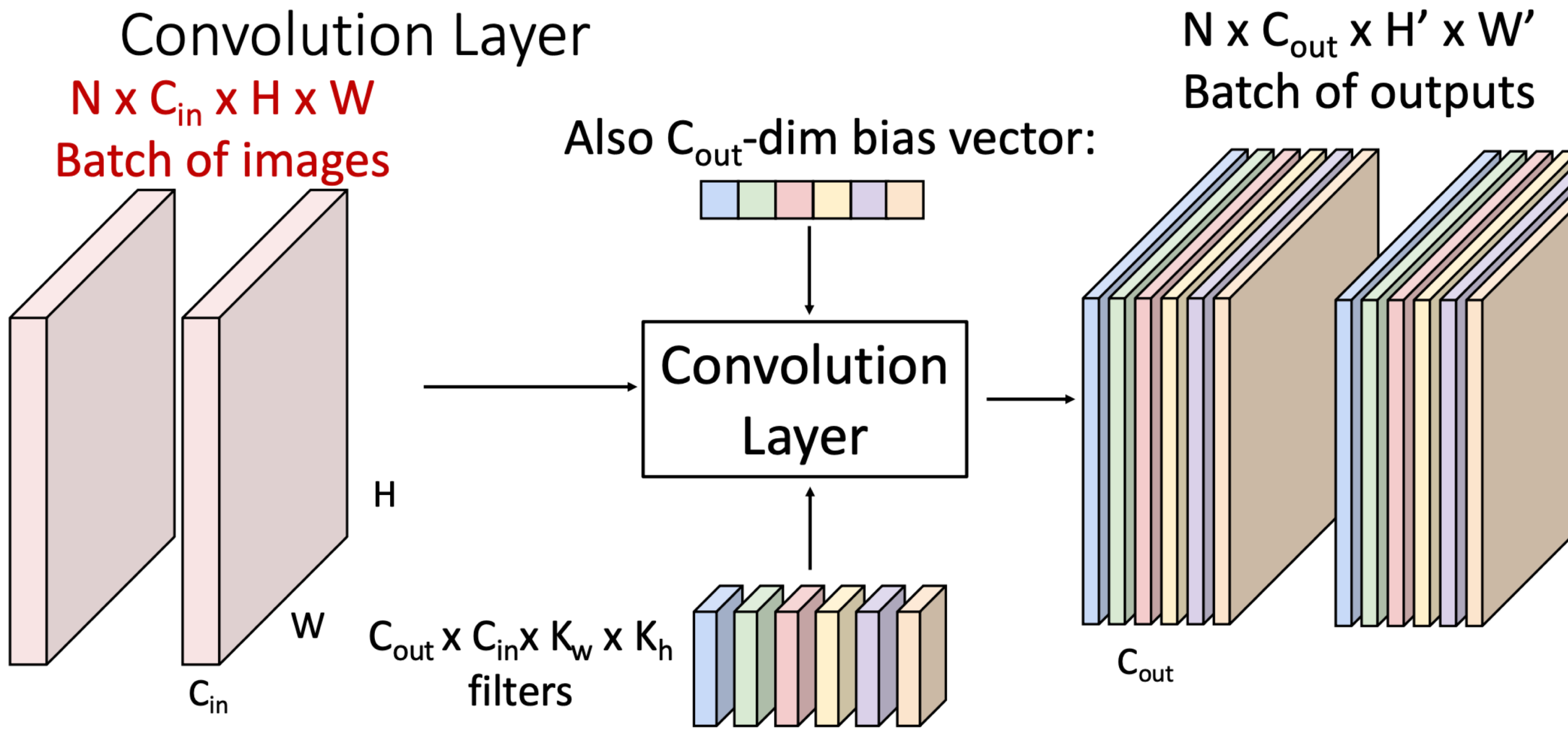
Convolution Layer



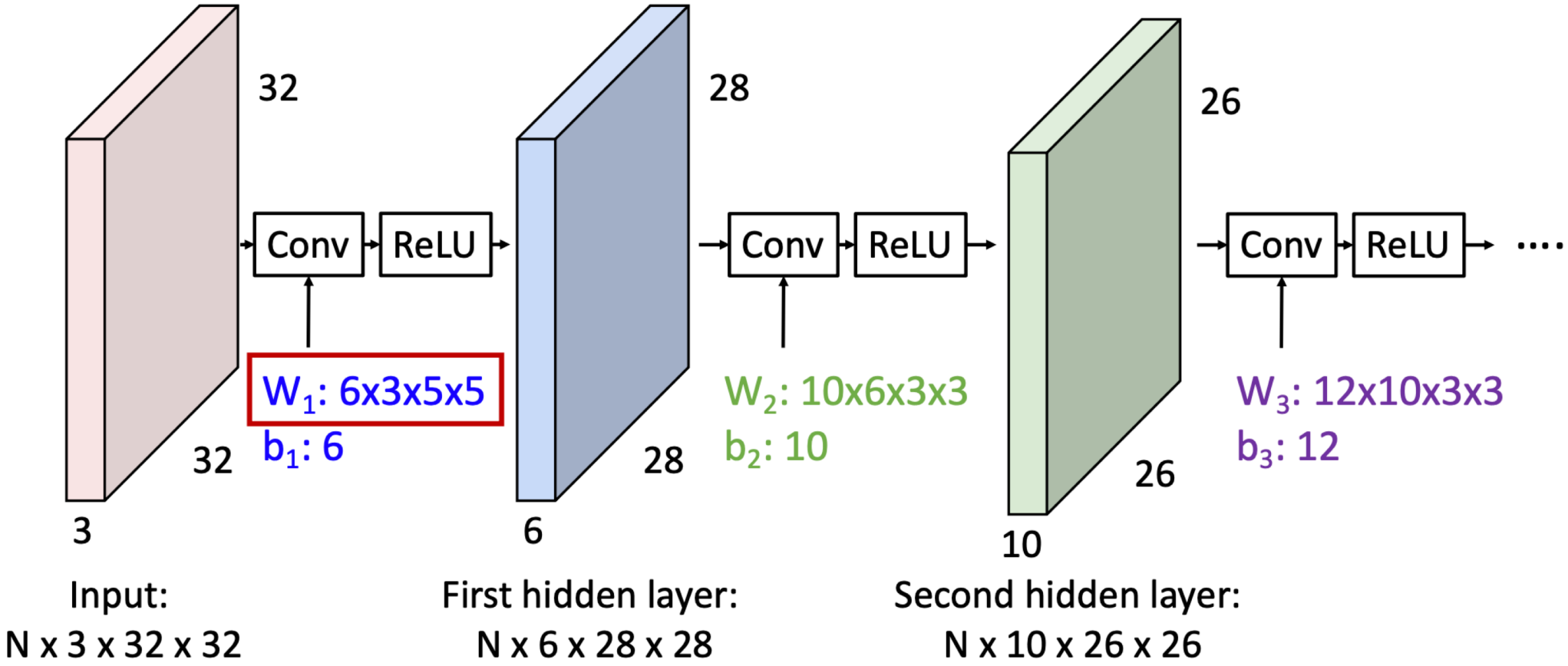
Convolution Layer



Convolution Layer

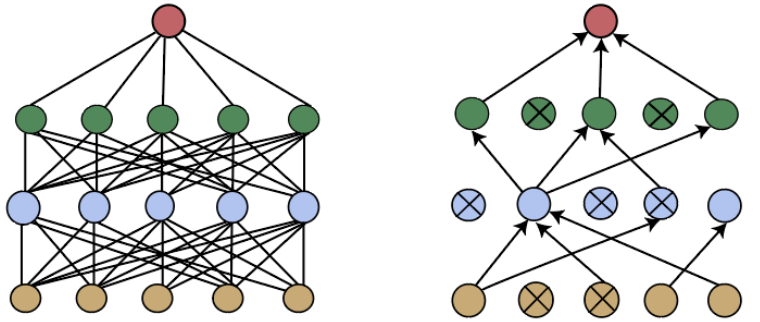


CNN: Stacking Convolutions with Non-Linearities



Neural Net Regularizers

Dropout



Standard Neural Network | After Applying Dropout

- **Key idea:** At every training iteration, dropout hidden units with certain probability (say 0.5). Deactivate it during testing.
- **Intuition:** An easy ensembling technique for DNNs!
  - At every training iteration, we have a new (subset) neural network.
  - During testing, we have the full (superset) network combining all subset NNs
- Works very well to reduce overfitting in DNNs

Dropout In Practice

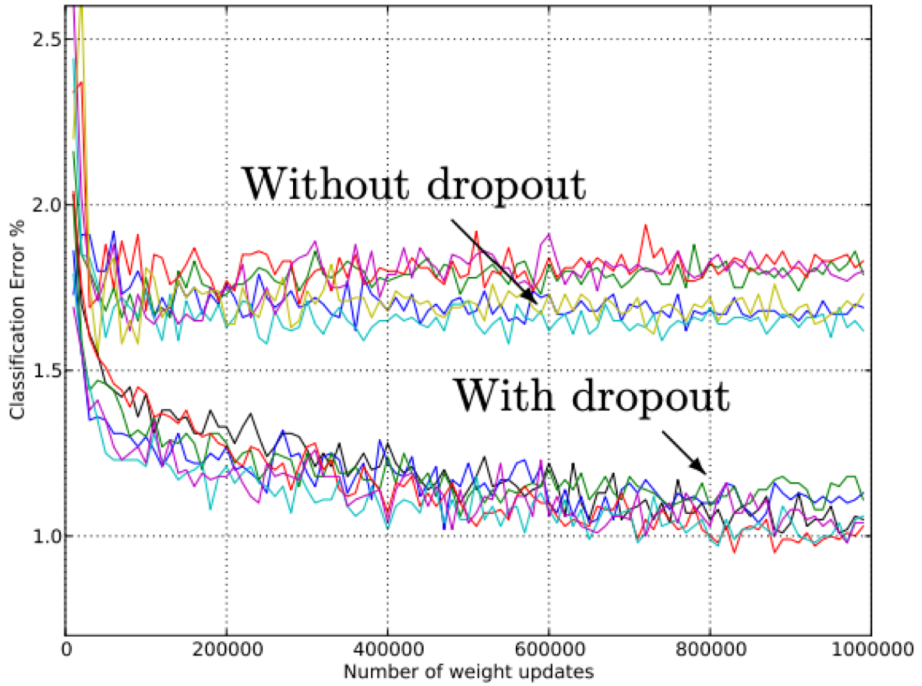


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Optimizing DNNs

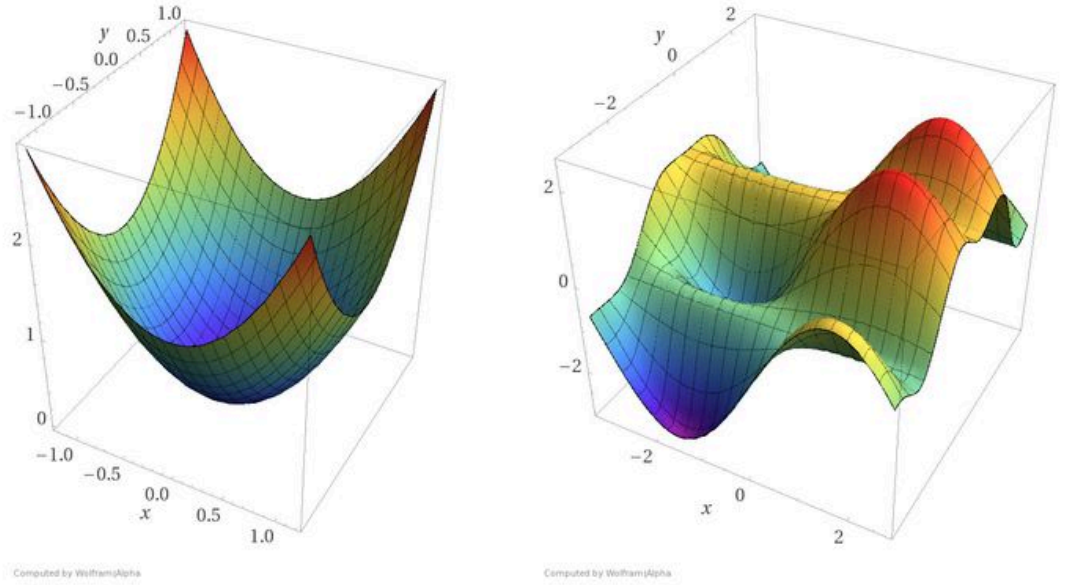
Optimizing DNNs

- So far, we have relied heavily on convex optimization problems
  - Unconstrained opt: Linear/logistic regression, perceptrons
  - Constrained opt: SVMs

- Good off-the-shelf solvers that work well in both theory and practice for these problems
  - e.g, gradient descent

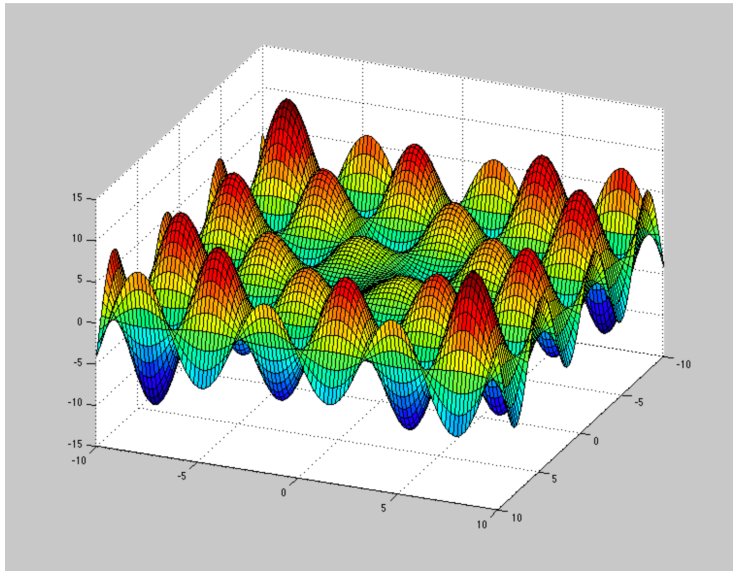
- **Key challenge:** Loss landscape of DNNs is highly non-convex w.r.t. unknown parameters

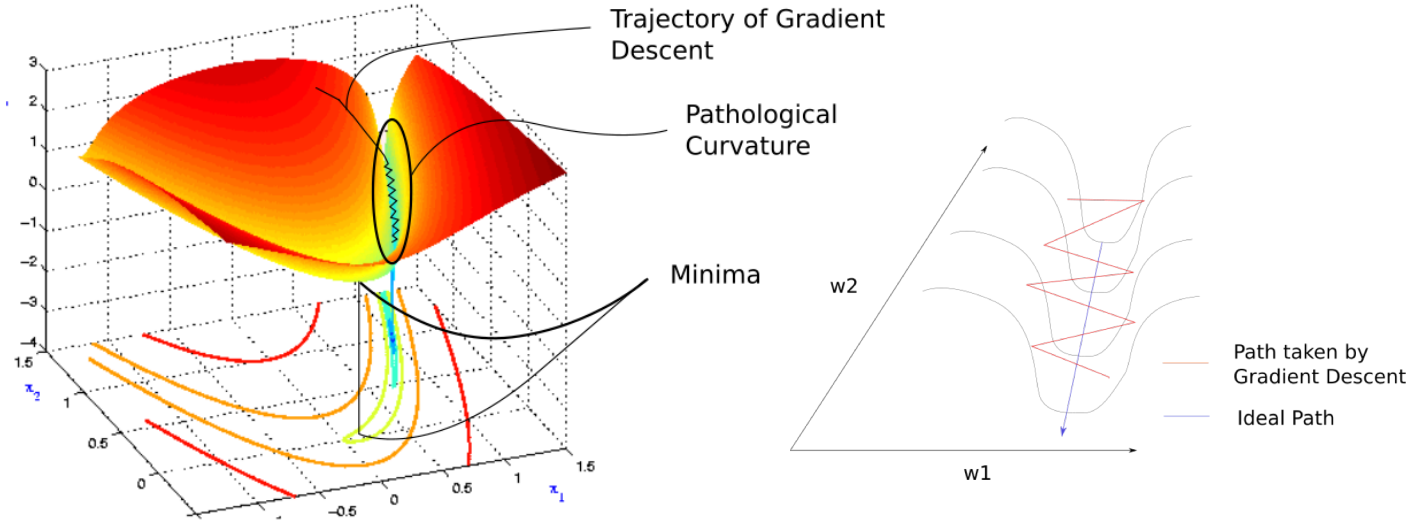
Optimizing DNNs



- While theory is lacking, very good empirical evidence in favor of gradient-based optimizers for optimizing deep neural networks
- Default gradient optimizers don’t work, need a few tweaks

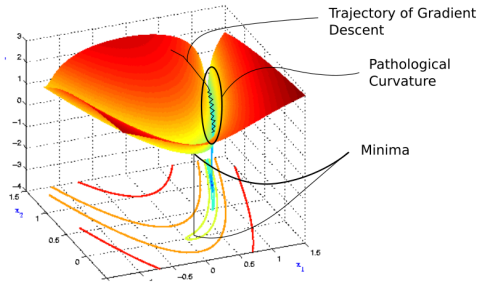
Non-Convex Optimization





- Gradient descent can struggle for optimizing non-convex objective functions. E.g.,
  - Can get stuck in local optima
  - Pathological curvatures such as “ravines” slow down convergence

Non-Convex Optimization



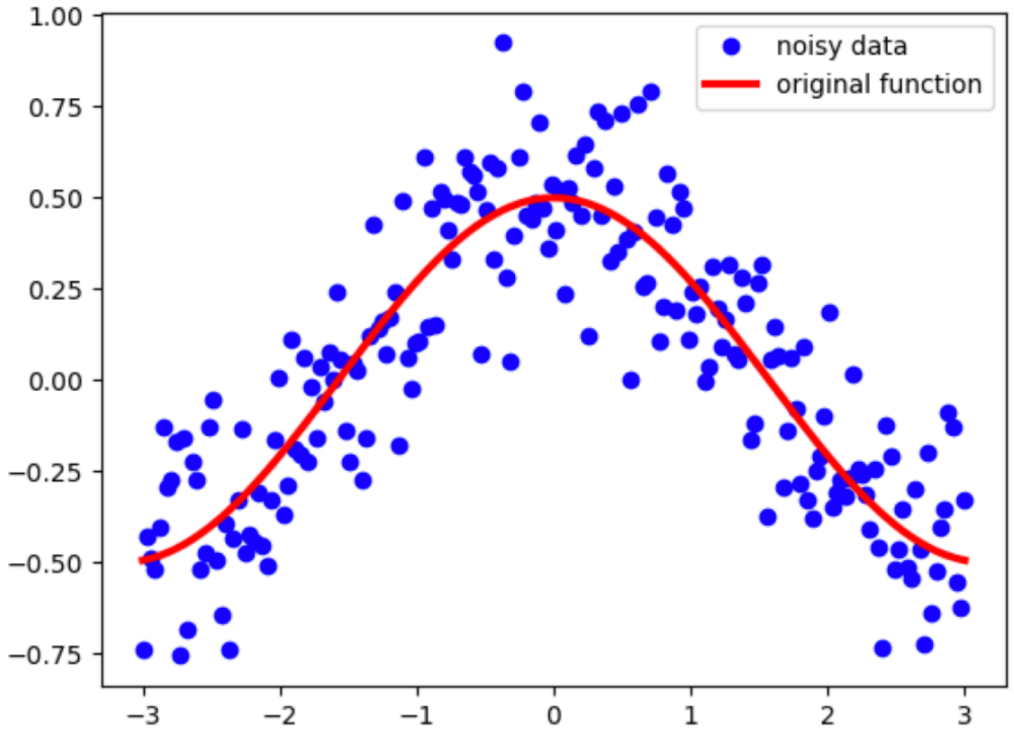
- Challenge: Pathological curvature of non-convex objectives

- **Candidate Solution:** Second order methods
  - **Pro:** Better at estimating curvature using second-order methods
  - **Con:** Require computing Hessian with $O(d^2)$ entries for d parameters
    - d is very large for DNNs. Second-order methods **do not scale!**

- **Momentum:** Use a heuristic to approximate the **rate of change of gradient** and use them for first-order optimization

Momentum

- Consider noisy samples from a noisy cosine function

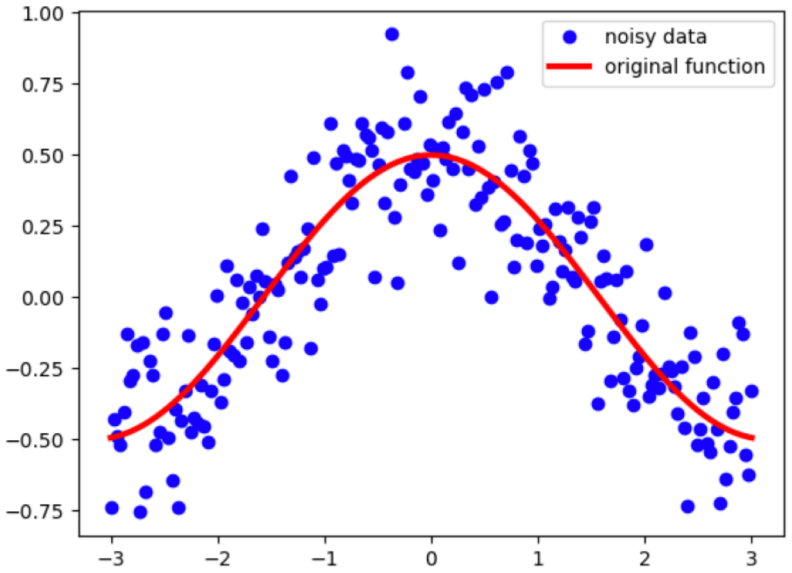


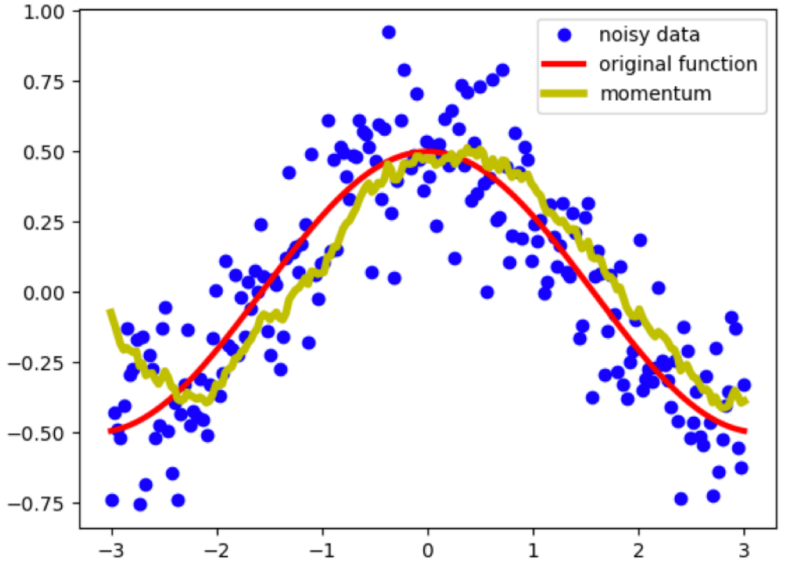
Out sequence.

Momentum

- Can smooth out a noisy function S_t by **moving averages**
- Initialize V_0 to be zero and compute

$$V_t = \beta V_{t-1} + (1-\beta) S_t$$
$$\beta \in [0,1]$$

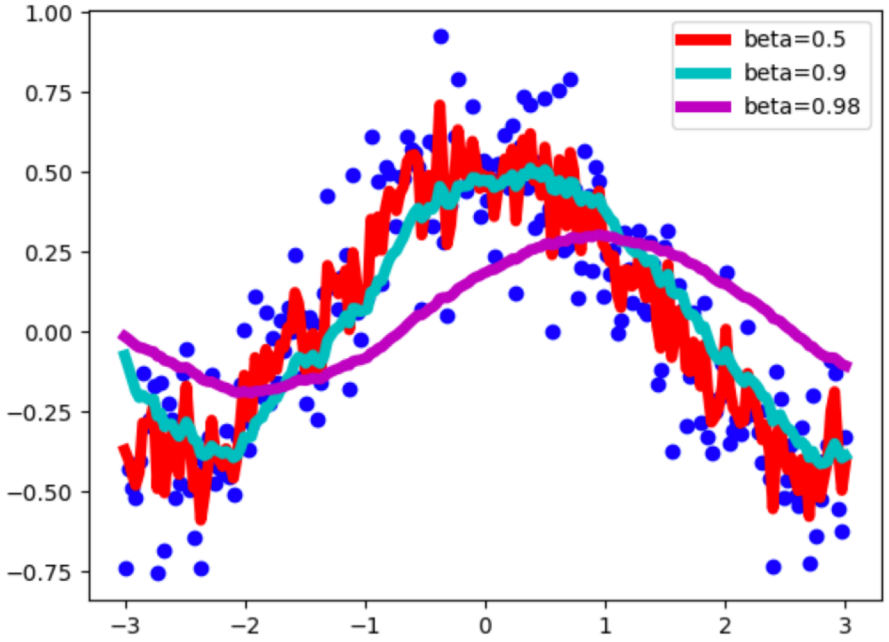




Out sequence.

Momentum

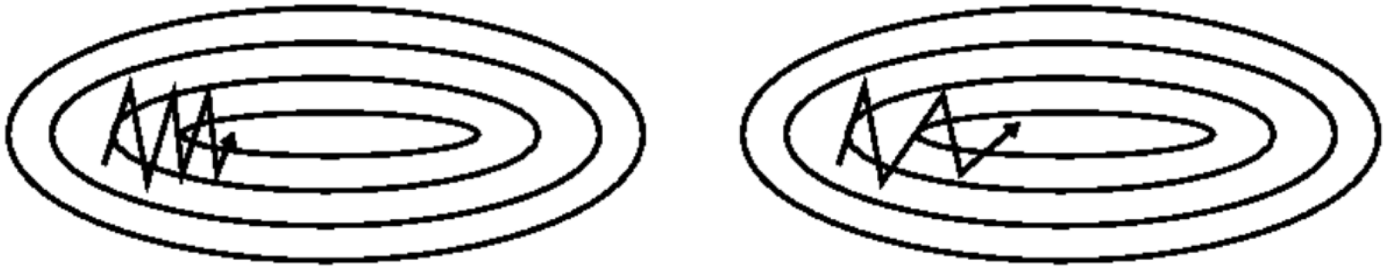
- β controls the extent of smoothing
  - Higher β, more smoothing



Gradient Descent with Momentum

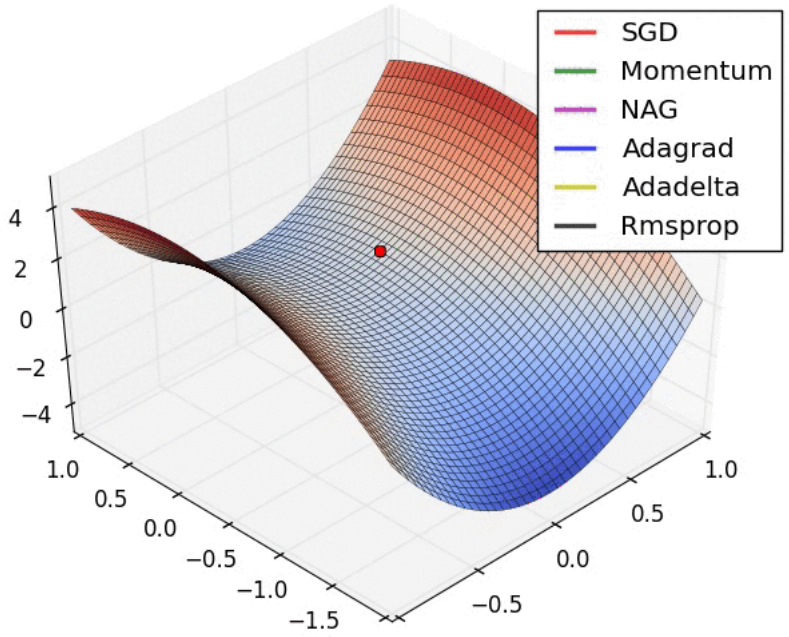
- **Key idea:** Use momentum for smoothing gradients of mini-batch gradient descent

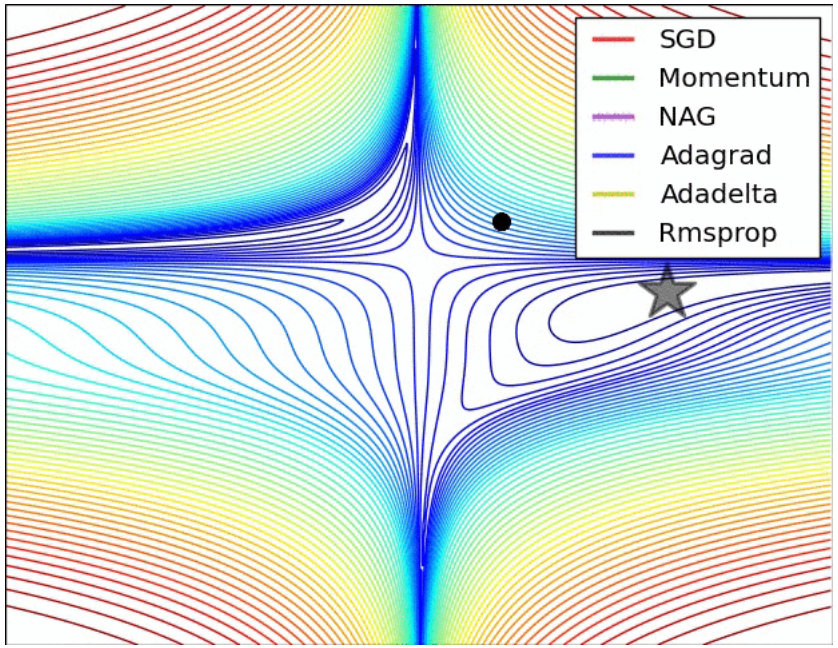
$$\boldsymbol{V}_0 = \mathbf{0}$$
$$\boldsymbol{V}_t = \beta \boldsymbol{V}_{t-1} + (1-\beta)\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$
$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \boldsymbol{V}_t$$



Left — SGD without momentum, right— SGD with momentum. (Source: Genevieve B. Orr)

Gradient Optimizers in Practice





Summary

- Neural Nets
  - Layered non-linear computation of hypothesis
  - Backpropagation for efficient computation of gradients using chain rule
- Deep Neural Nets
  - Big datasets due to sensor proliferation, internet
  - Fast computation fueled by advances in GPU computing
  - Powerful algorithms for scaling and generalization
    - Architectures: design of connectivity structure, activations, normalization strategies
    - Regularization techniques: dropout for efficient ensembling
    - First-Order Optimizers With Momentum