Deep Learning Part 1
Convolutional Networks

Slides by:

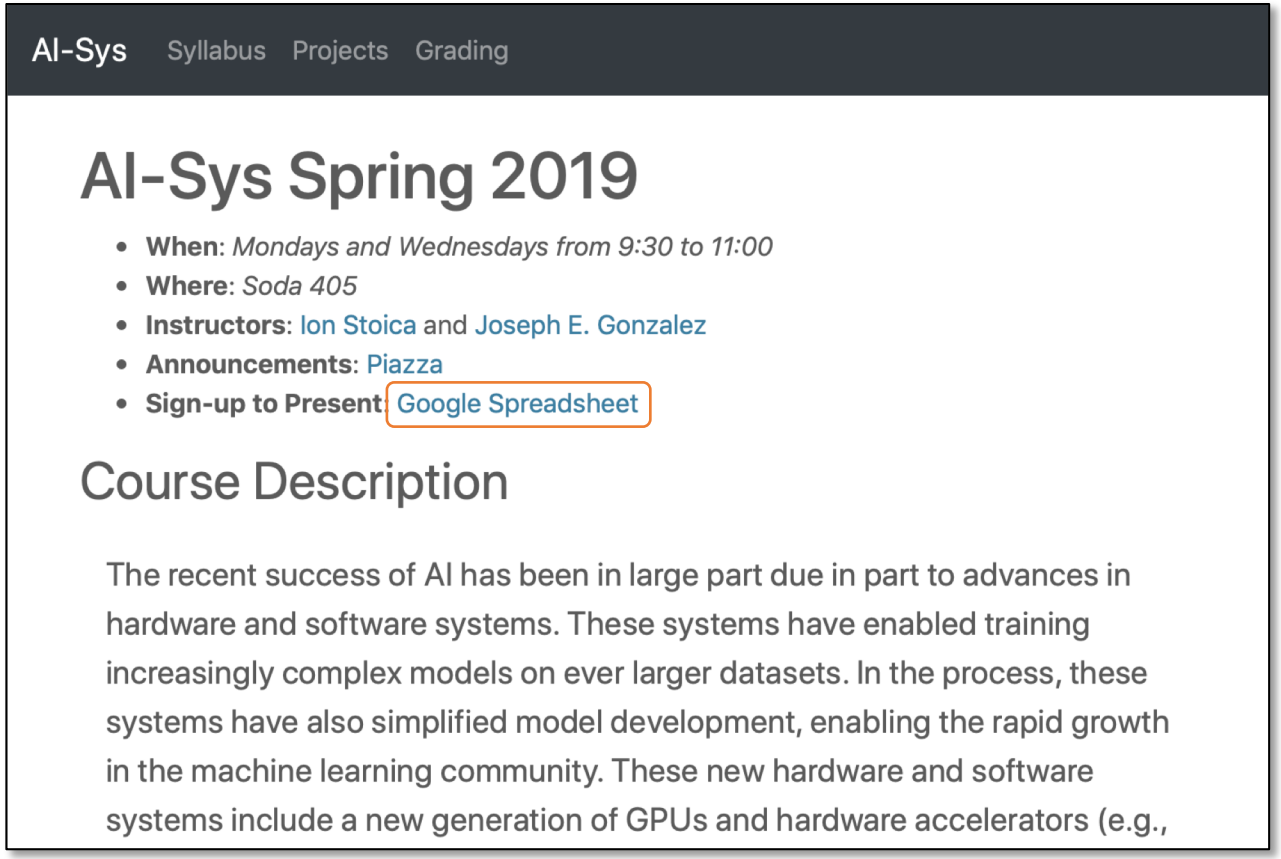
Joseph E. Gonzalez,

jegonzal@berkeley.edu

Quick Logistics

- Please sign up to present
- Complete reading questions before lecture

http://bit.ly/aisys-sp19



Outline

- Crash course in supervised learning
  - Loss Minimization and Function Approximators
  - Features Functions
- Convolutional Networks
  - Intuition
  - LeNet
- Two Papers:
  - AlexNet Paper: First Big Deep Learning ILSVRC Result
  - Inception v4: Latest Big Deep Learning ILSVRC Result

Machine Learning Crash Course

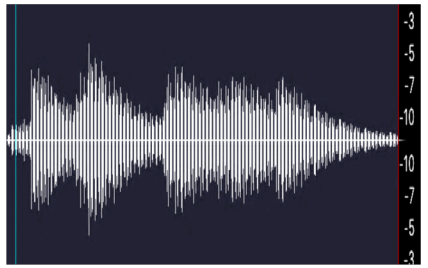
Supervised Learning

Machine Learning ≈ Function Approximation

Object Recognition

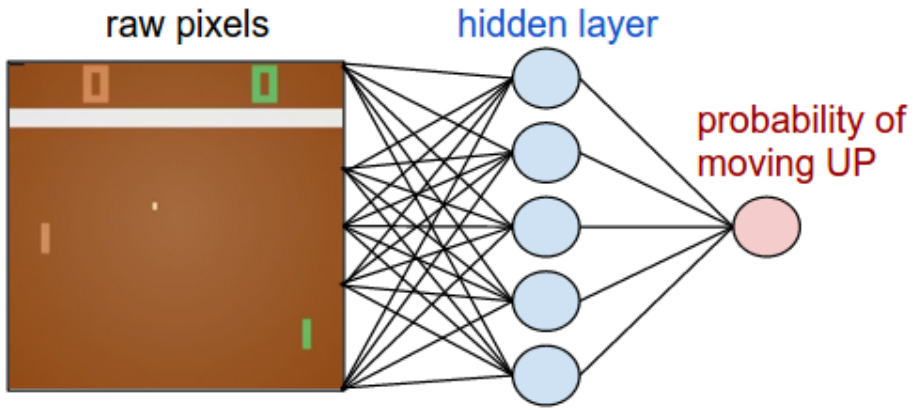
Label:*Cat*

Speech Recognition

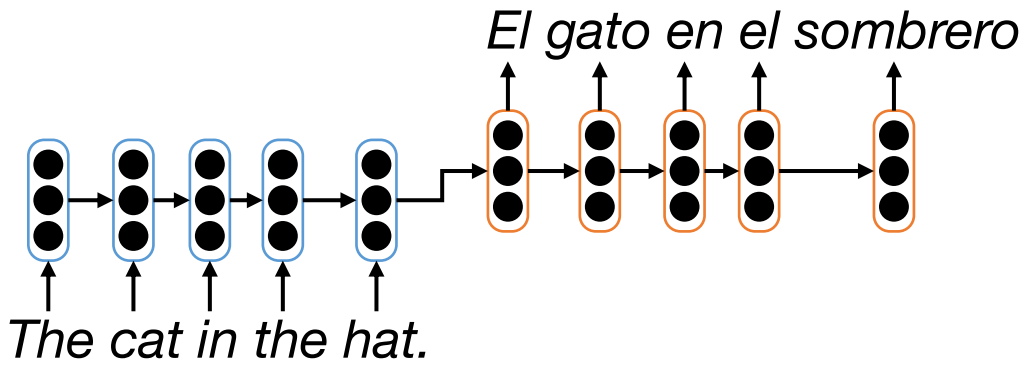


“The cat in the hat”

Robotic Control



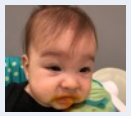
Machine Translation



Supervised Machine Learning

- Given data containing the function **inputs** and **outputs**

Data

Input	Output
X_1 	Y_1 cat
X_2 	Y_2 *baby*
…	…
X_n 	Y_n *baby*

Model

$$f_\theta(x) \rightarrow y$$

Parameters

Goal

$$\theta^* = \arg\min_\theta \mathbb{E}_D \left[L\left(f_\theta(x), y \right) \right]$$

Loss

Over all future data

Training (approximates the goal over training data):

$$\hat{\theta} = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} L\left(f_\theta(x_i), y_i \right)$$

Focus of next two lectures

$$f_\theta(x) \rightarrow y$$

- How do we make our functions sufficiently expressive
- Capture domain knowledge / assumptions
- Easy to train and compute

Classical* Machine Learning

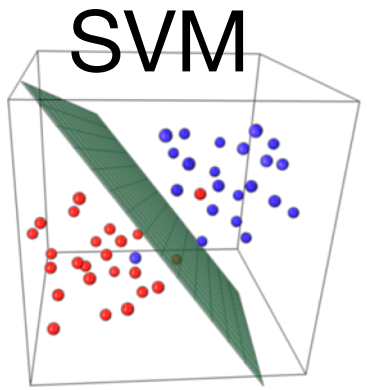
- Break function into features and learned model

$$f_\theta(x) = g_\theta\left(\phi(x)\right)$$

Generic Easy to Train Function (Linear Model)

Hand Engineered Features

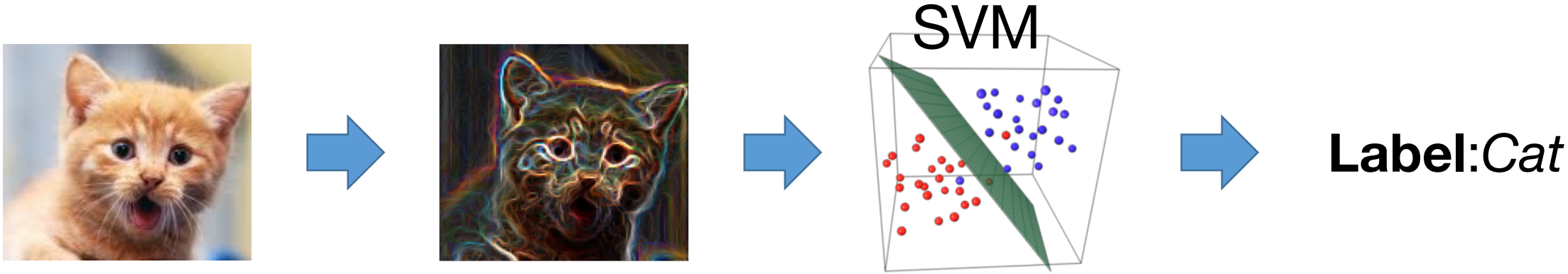
- Example



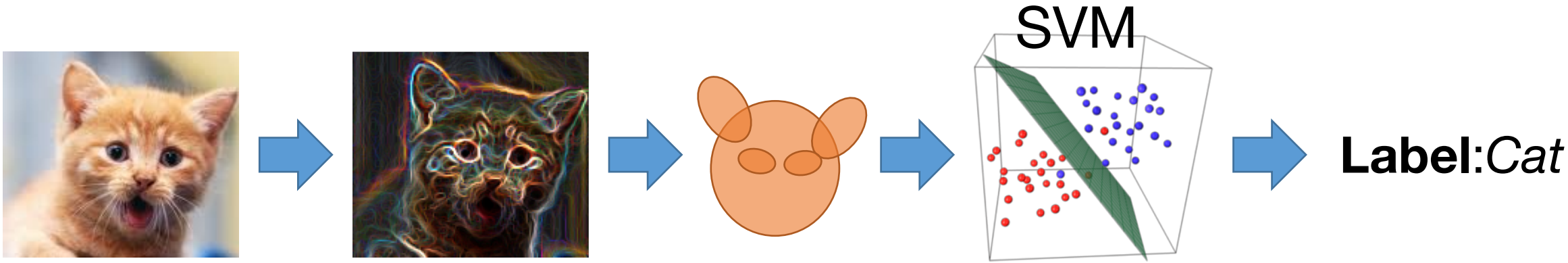
Label:*Cat*

* This is also very frequently used today as well.

Function Approximation Pipeline

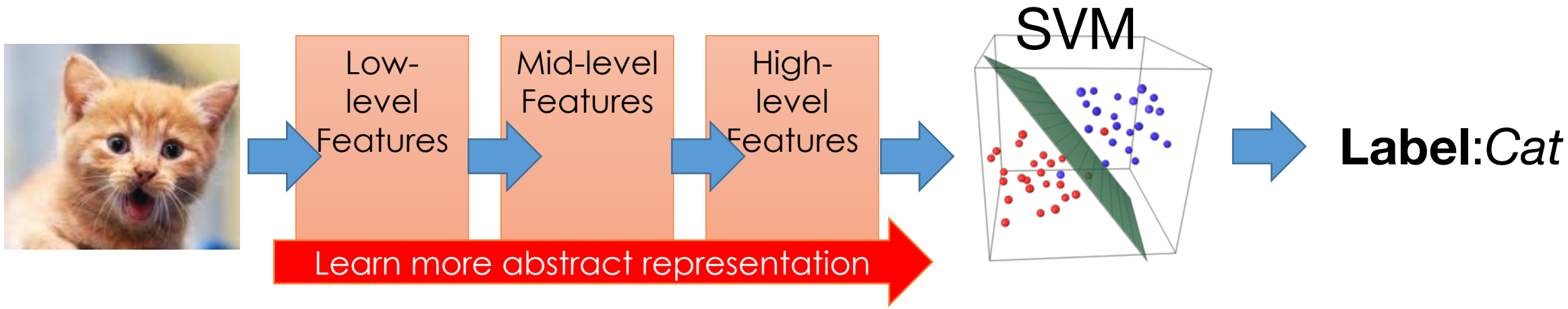


Often build multiple layers of features to abstract the input



Deep learning tries to automate this process.

Function Approximation Pipeline



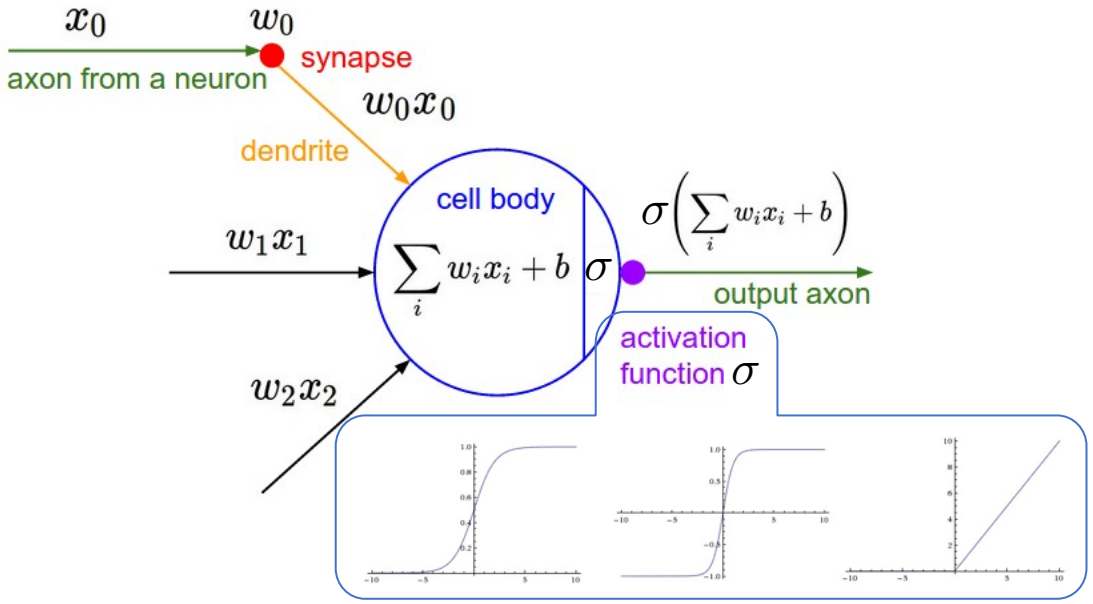
Deep Learning: automatically *learns a deep hierarchy of abstract features* along with the classifier.

- Typically using neural networks
  - composable **general** function approximators

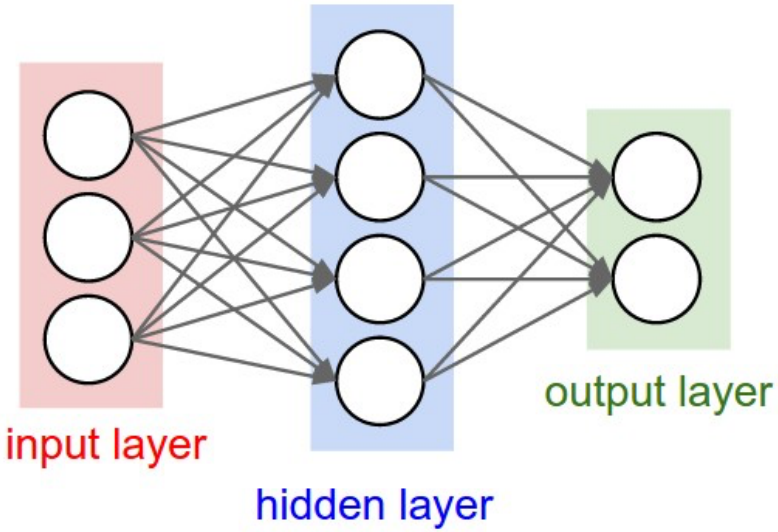
Can we build a generic function for all tasks?

$$f_\theta(x) \rightarrow y$$

Single "Neuron"



Multi-layer Neural Network



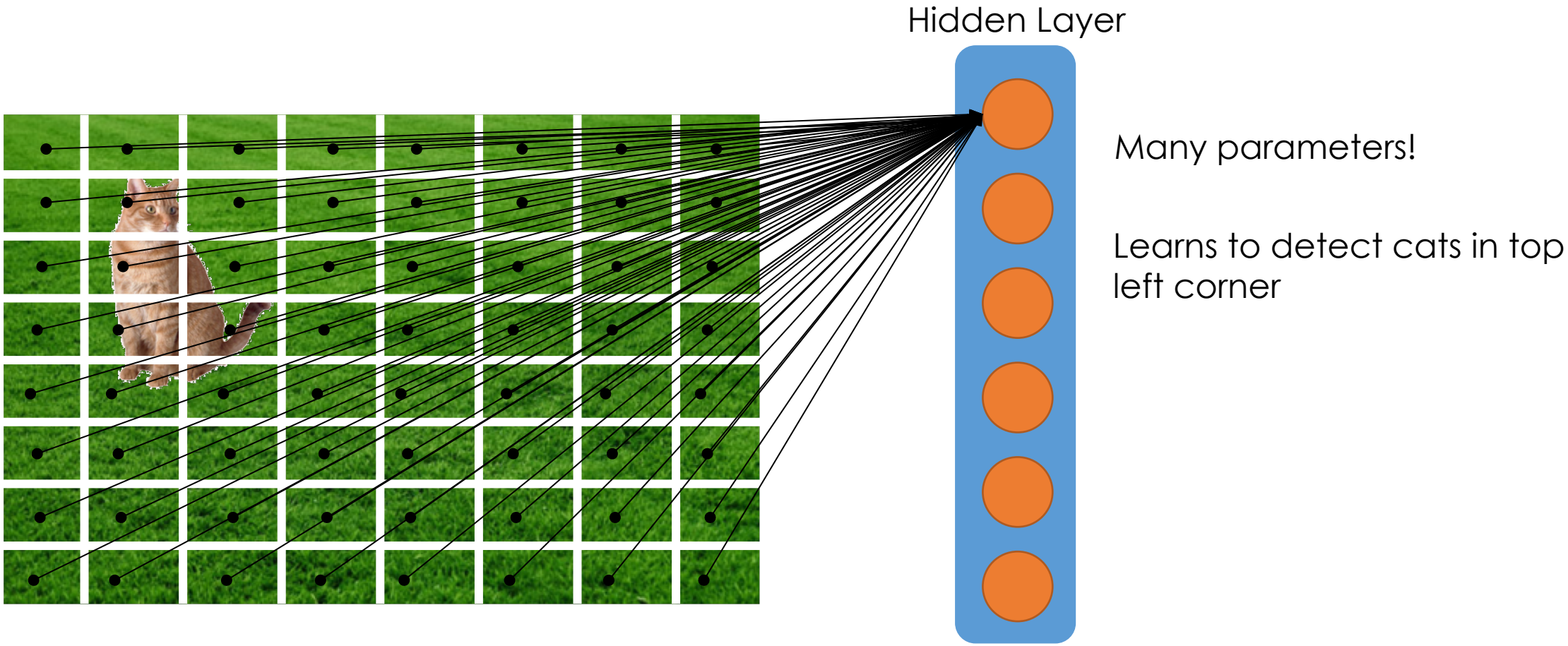
Shown to be **Universal Approximators** [Holt et al. 1989]

- Given enough hidden units, a single hidden layer network can approximate any continuous function.

Too generic for most tasks → difficult to train

Images courtesy of http://cs231n.github.io/

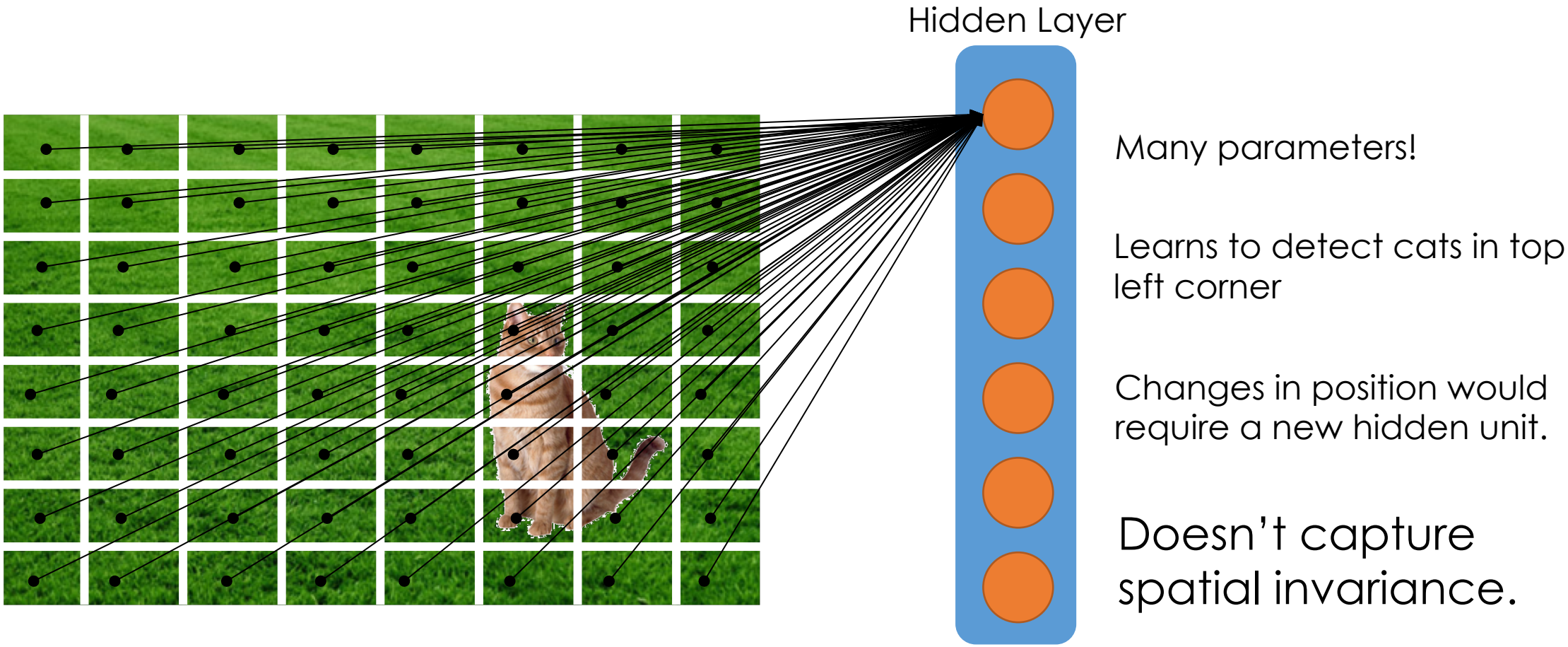
Focus on Computer Vision Tasks



Many parameters!

Learns to detect cats in top left corner

Focus on Computer Vision Tasks



Many parameters!

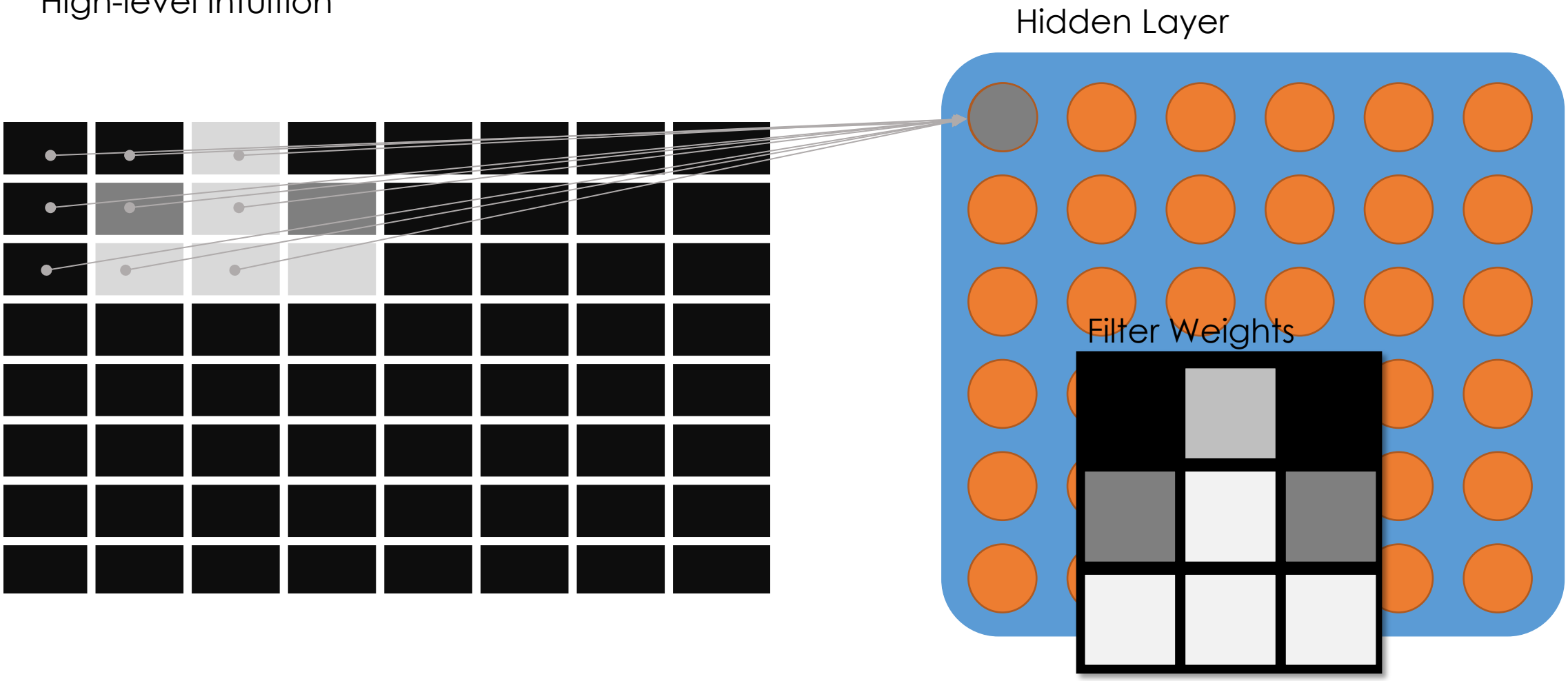
Learns to detect cats in top left corner

Changes in position would require a new hidden unit.

Doesn't capture spatial invariance.

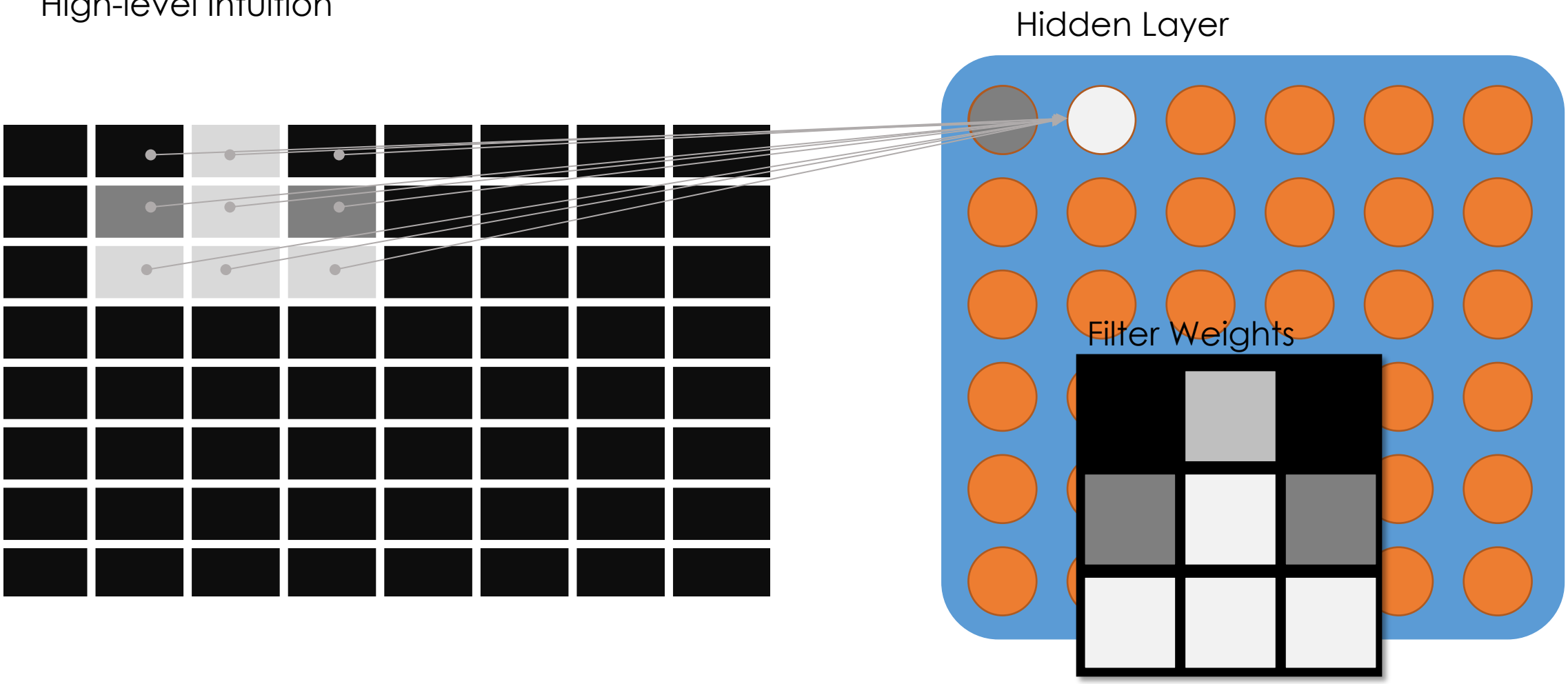
Convolutional Networks

High-level Intuition



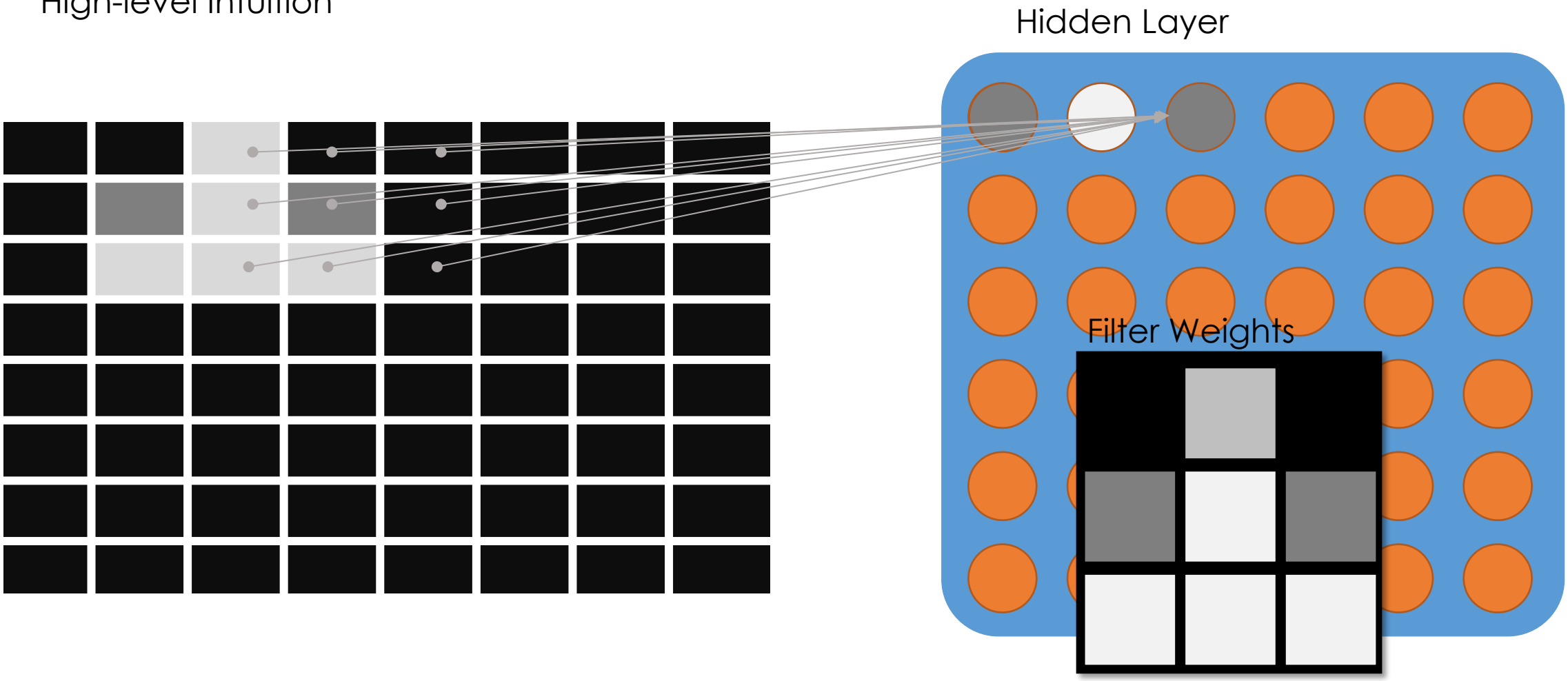
Convolutional Networks

High-level Intuition



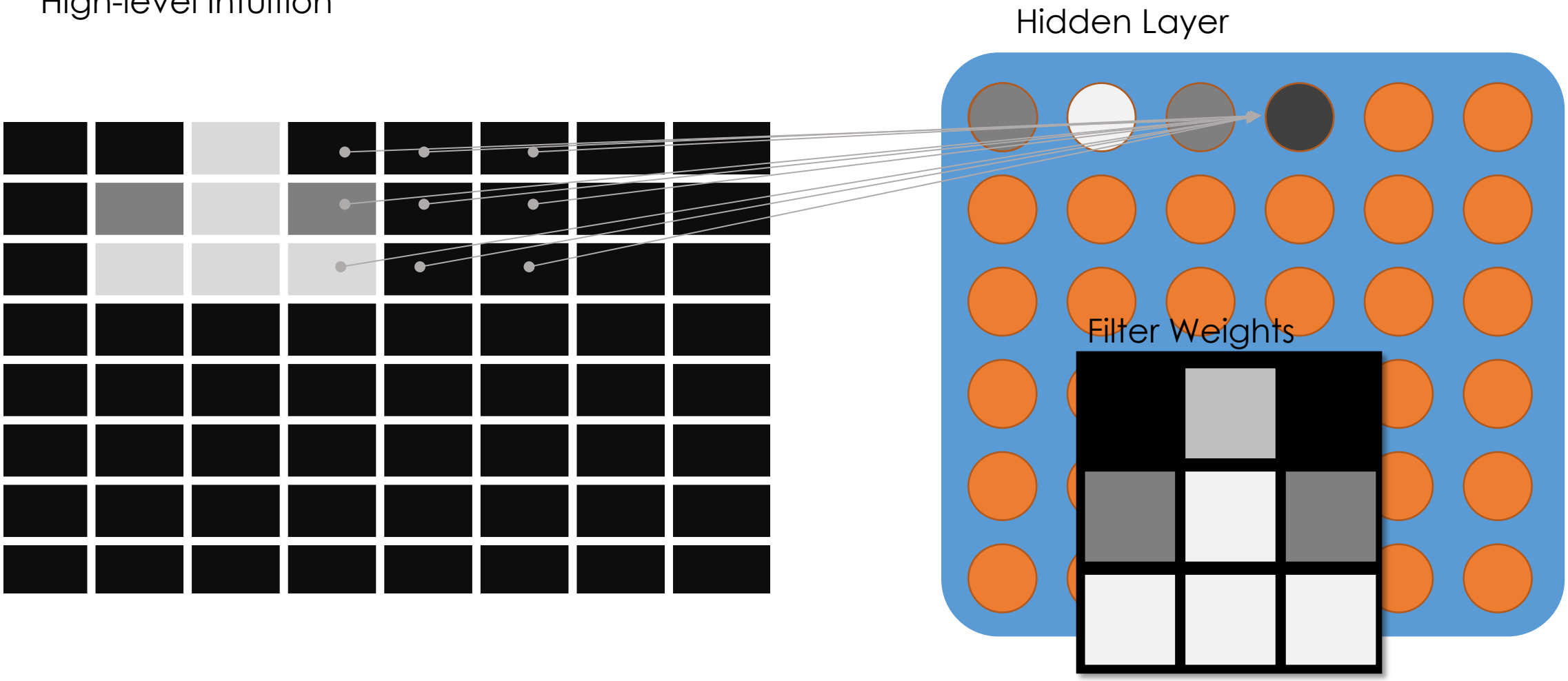
Convolutional Networks

High-level Intuition



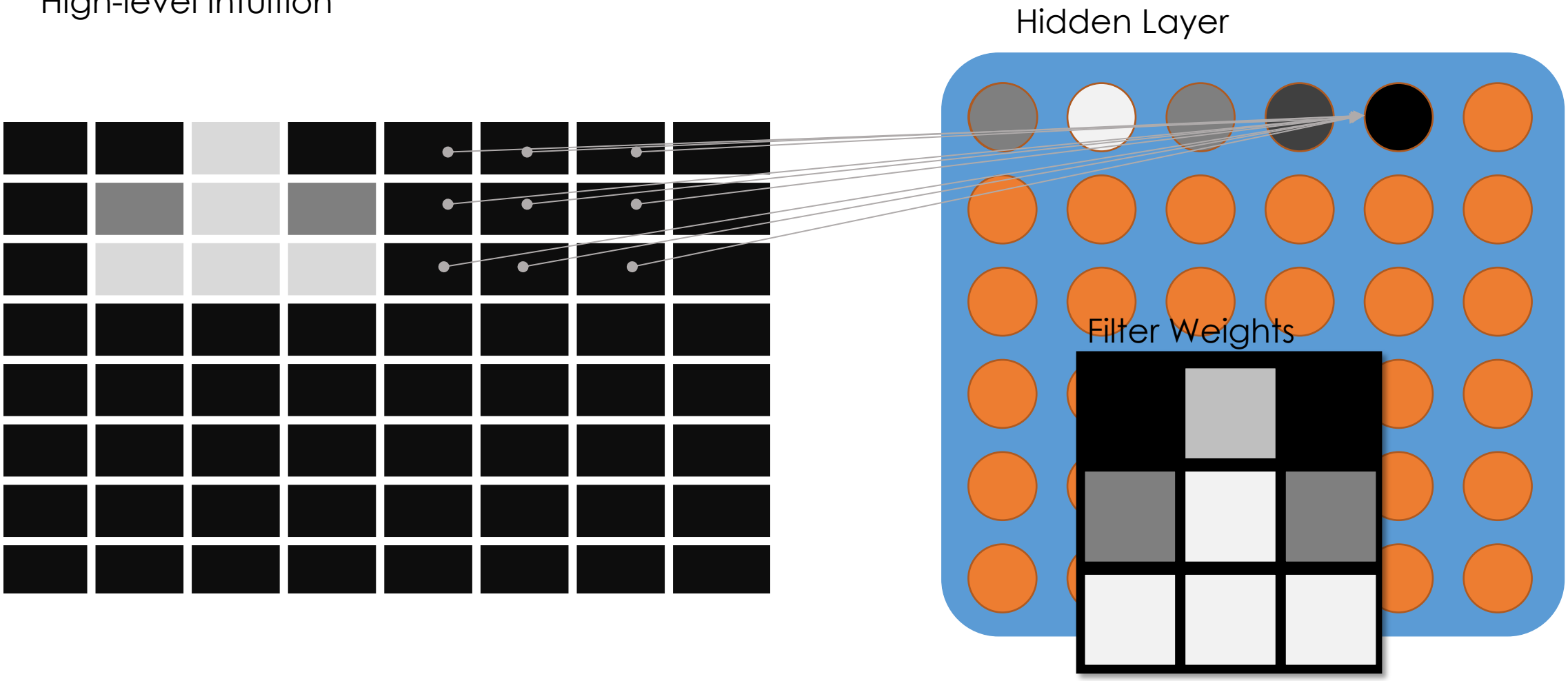
Convolutional Networks

High-level Intuition



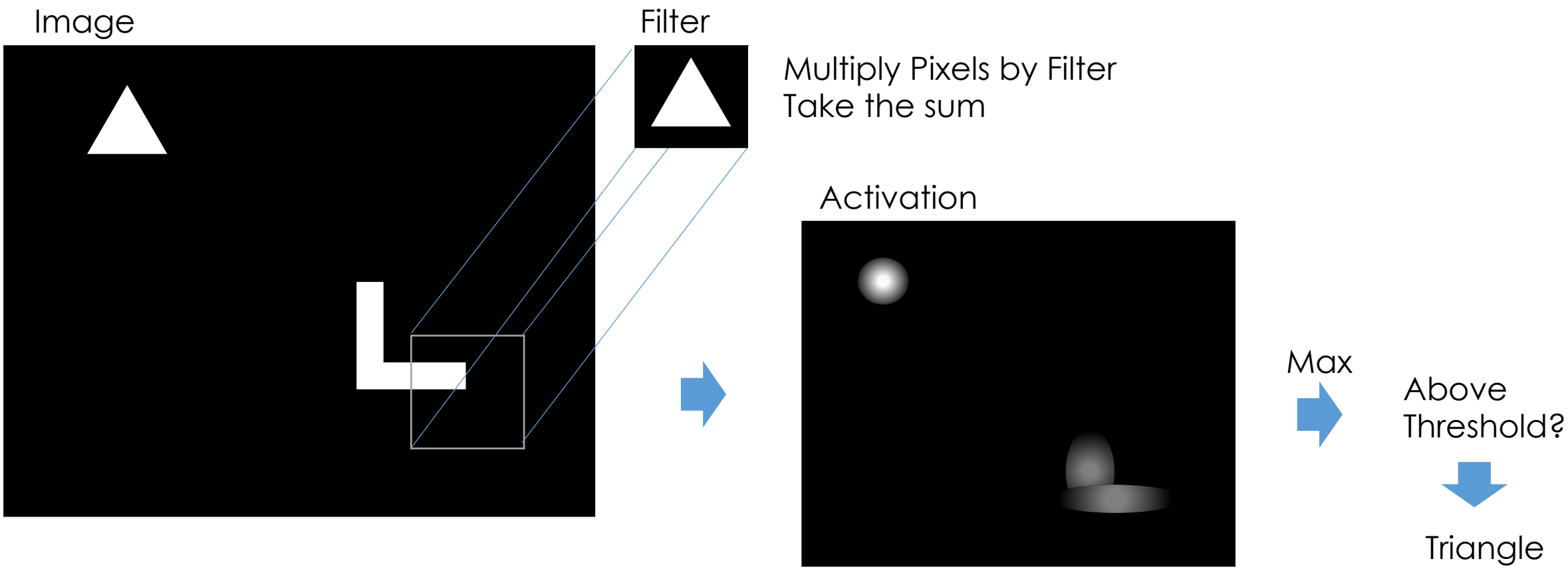
Convolutional Networks

High-level Intuition



Convolutional Networks

High-level Intuition



We can learn the filter.

LeNet [1989]

Communicated by Dana Ballard

Backpropagation Applied to Handwritten Zip Code Recognition

Y. LeCun
B. Boser
J. S. Denker
D. Henderson
R. E. Howard
W. Hubbard
L. D. Jackel
AT&T Bell Laboratories Holmdel, NJ 07733 USA

The ability of learning networks to generalize can be greatly enhanced by providing constraints from the task domain. This paper demonstrates how such constraints can be integrated into a backpropagation network through the architecture of the network. This approach has been successfully applied to the recognition of handwritten zip code digits provided by the U.S. Postal Service. A single network learns the entire recognition operation, going from the normalized image of the character to the final classification.

1 Introduction

Previous work performed on recognizing simple digit images (LeCun 1989) showed that good generalization on complex tasks can be obtained by designing a network architecture that contains a certain amount of a priori knowledge about the task. The basic design principle is to reduce the number of free parameters in the network as much as possible without overly reducing its computational power. Application of this principle increases the probability of correct generalization because it results in a specialized network architecture that has a reduced entropy (Denker *et al* 1987; Patarnello and Carnevali 1987, Tishby *et al*. 1989; LeCun 1989), and a reduced Vapnik–Chervonenkis dimensionality (Baum and Haussler 1989).

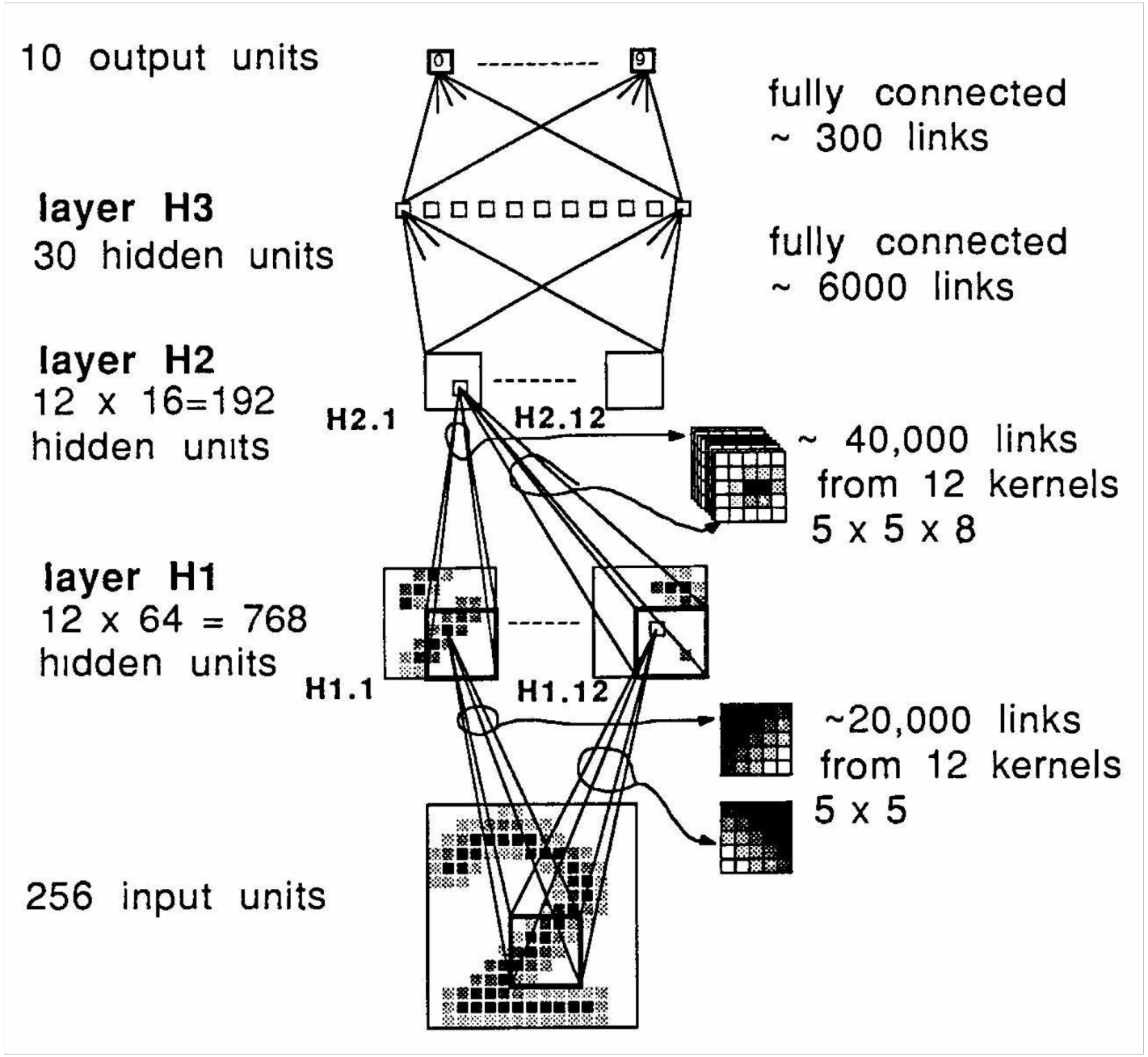
In this paper, we apply the backpropagation algorithm (Rumelhart *et*

- Credited with introduced weight sharing equivalent to convolution.
- High accuracy on digit recognition
  - An important problem for postal routing

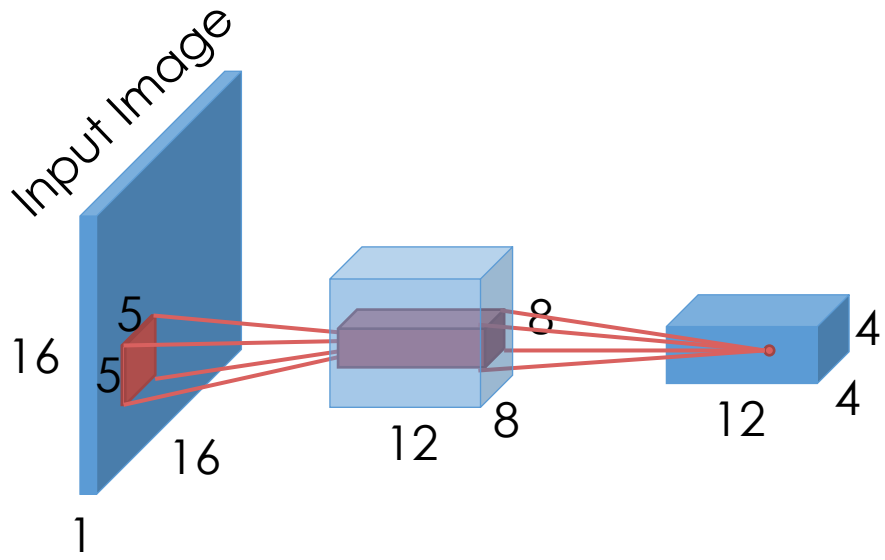
Predecessor to MNIST Digits Dataset

- Used backpropagation and stochastic gradient during training
- Implemented on DSP for fast inference
  - 30 digits per second on DSP
    - (10 to 12) for entire pipeline

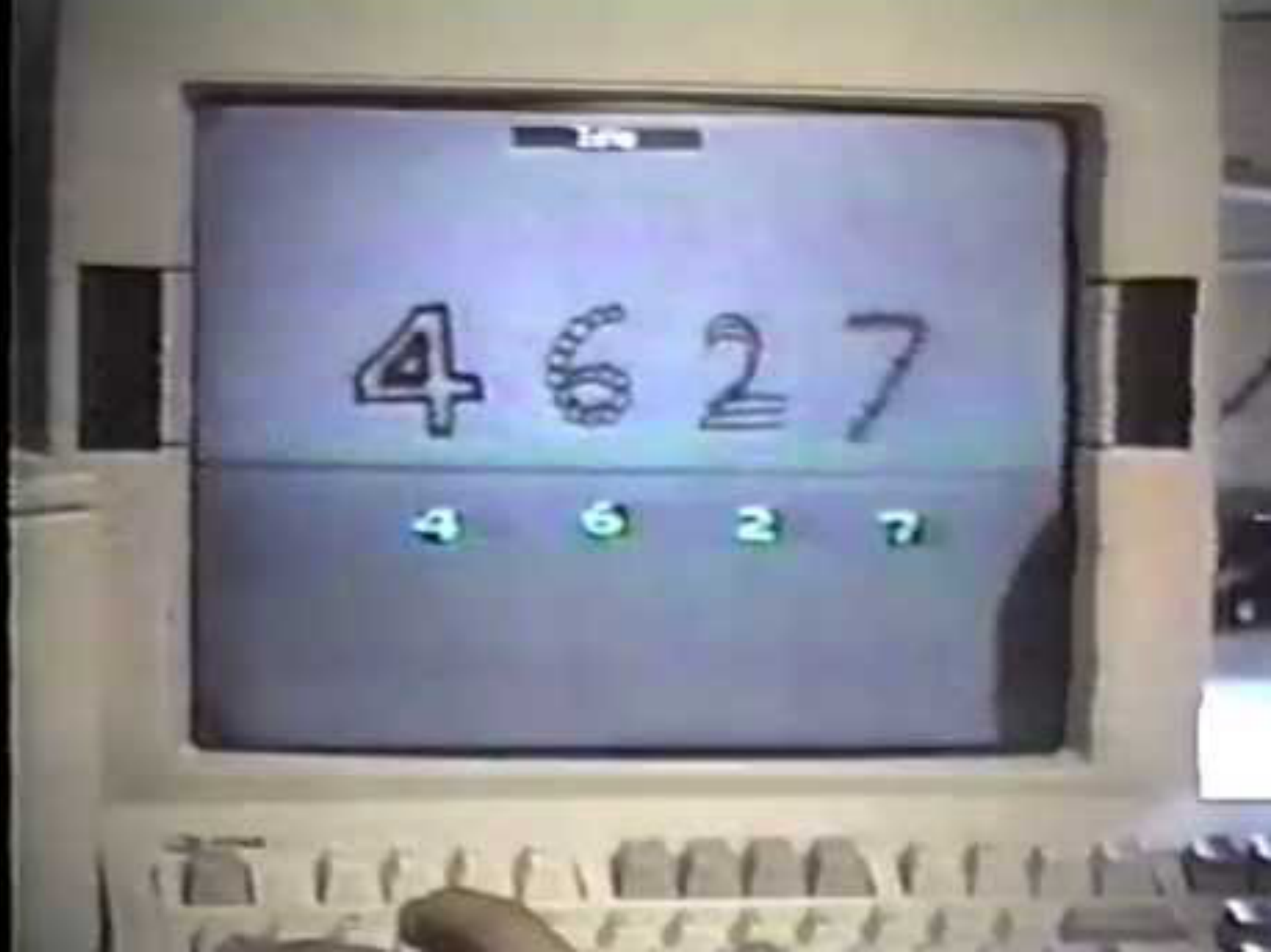
LeNet Architecture



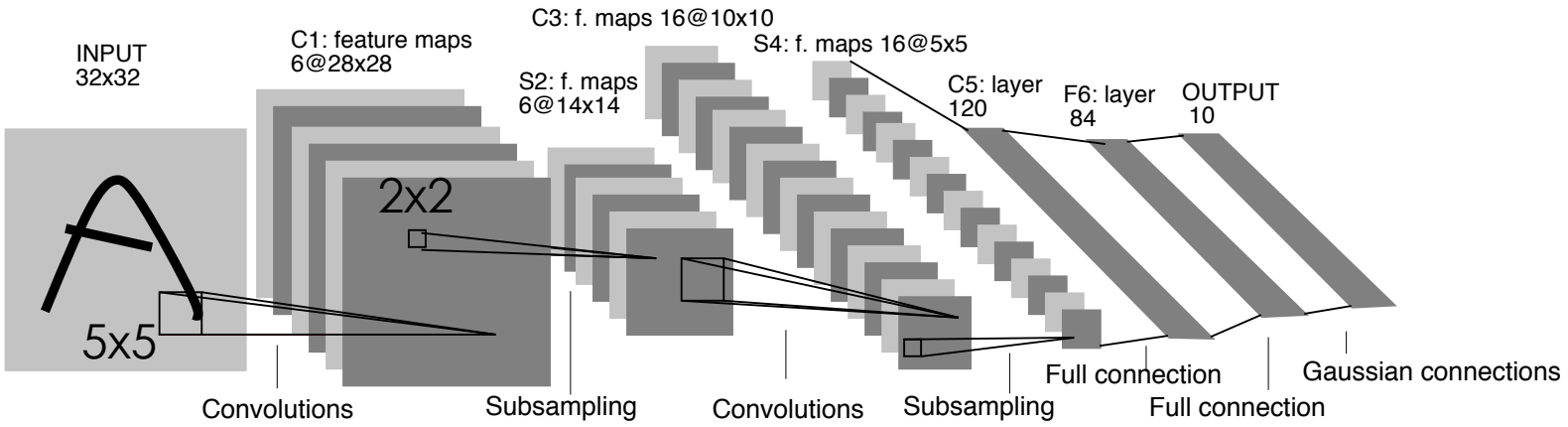
Tensor Illustration



Yann LeCun
1993 LeNet1
Demo



LeNet5 [1998]



- Directly used convolution terminology
- Major commercial success in check reading systems
  - reading between 10 and 20% of all the checks in the US in the early 2000s.

Papers

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky, Illya Sutskever, Geoffrey E. Hinton

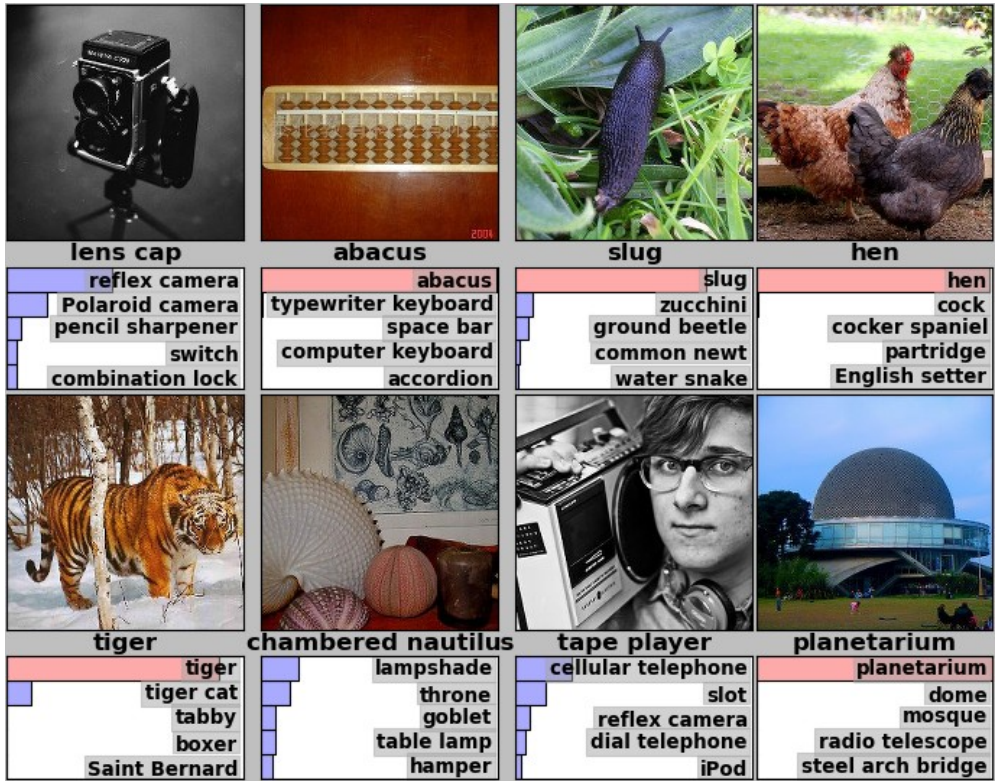
Presented by Joseph E. Gonzalez

TL;DR; This paper describe the deep convolutional architecture, training techniques, and system innovations that resulted in the winning entry for the ILSVRC-2012 Benchmark. This model substantially outperformed the next best model that year.

What is the problem being solved?

Image-Net **Benchmark** (ILSVRC-2012)

- 1.2M hand labeled (1000 classes) real-world images
  - Images have varying resolutions
- Multiple tasks including
  - **Classification** – algorithm may return 5 predictions per image
  - **Classification with Localization** – Classification + a bounding box
  - **Fine-Grained Classification**; – 100+ dog subcategories.



About the Problem

- Is this an important problem?
  - Not directly, however the resulting models could be readily applied to a wide range of more specialized tasks.
- Is this problem hard?
  - Yes! Humans often make mistakes.
  - Prediction accuracy on earlier related tasks was relatively low
- Is this problem novel?
  - Yes! The ILSVRC had only been around for two years.

What are the metrics of success?

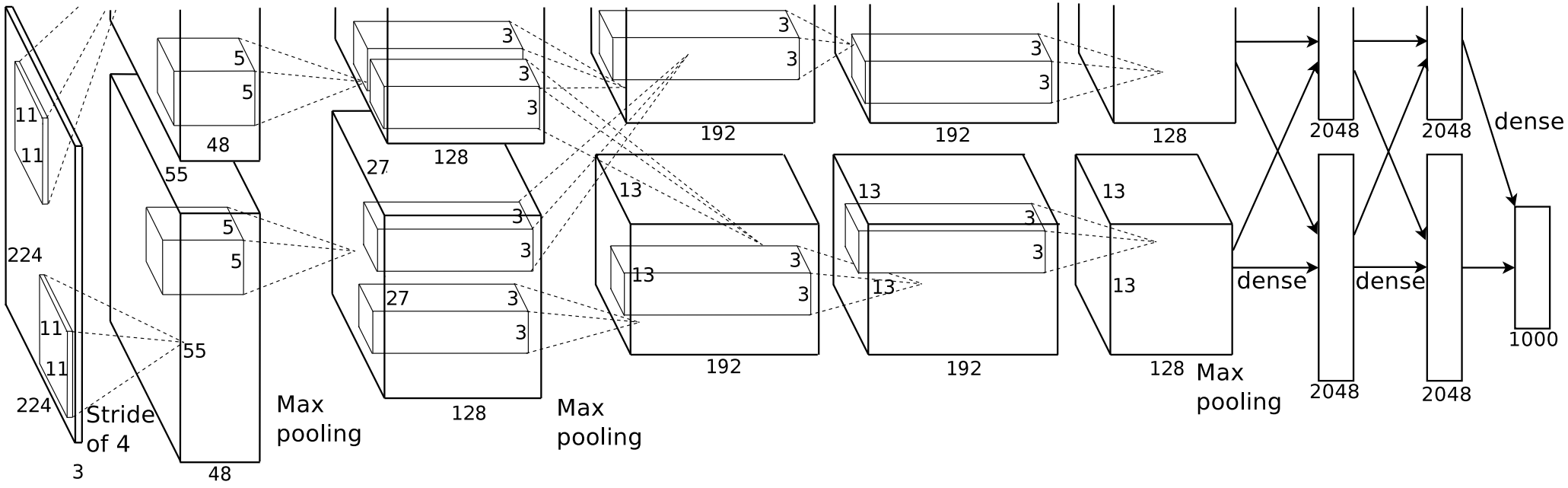
- Top-1 and top-5 **validation** and **test** accuracy
  - ILSVRC published 100,000 **test** images without labels and 50,000 **validation** images with labels
  - Top-5 Accuracy: model predict 5 possible categories
    - The model is correct if the human label matches any of the top 5
  - Top-1 Accuracy: model prediction must match human label
    - Human labels were occasionally wrong and multiple items in each image.
- Test results reported at the end of the competition
  - Difficult to overfit.
- Did not directly include computation time

What is the key innovation?

- **Eliminated the use of engineered features** and achieved more than a 40% reduction in error compared to next best model (which used hand engineered features).
- Combined innovations in **neural network architecture** with innovations in **hardware accelerated training**
  1. Extensive use of ReLU non-linearities
  2. Training on Multiple GPUs
  3. Local Response Normalization
  4. Reduced Overfitting using data augmentation and dropout

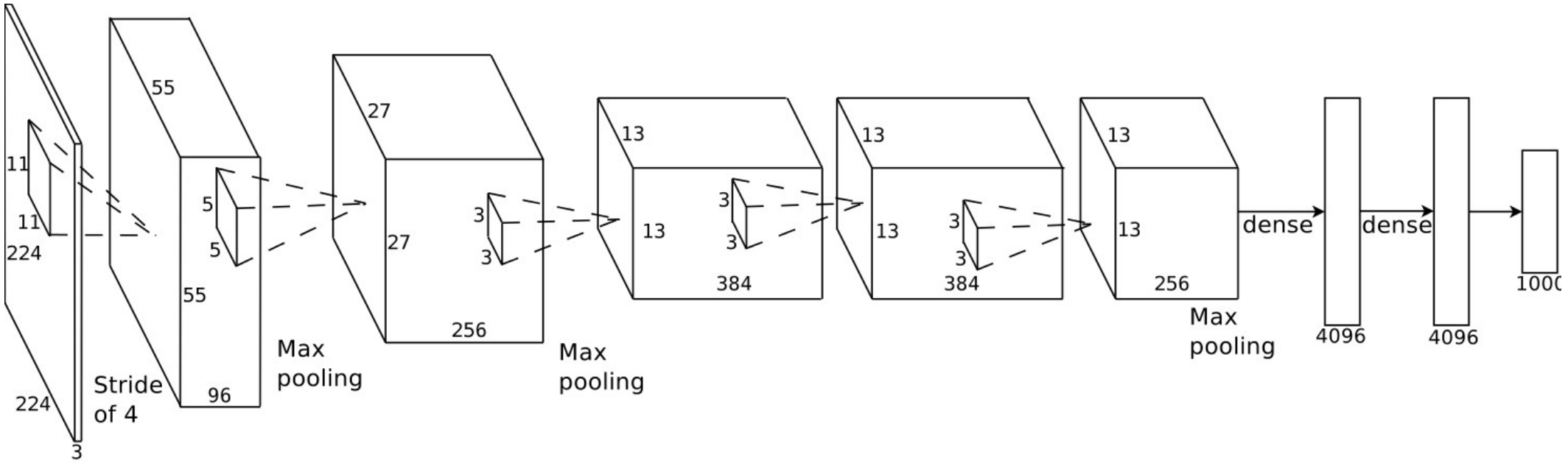
The Actual AlexNet* Architecture

from the paper



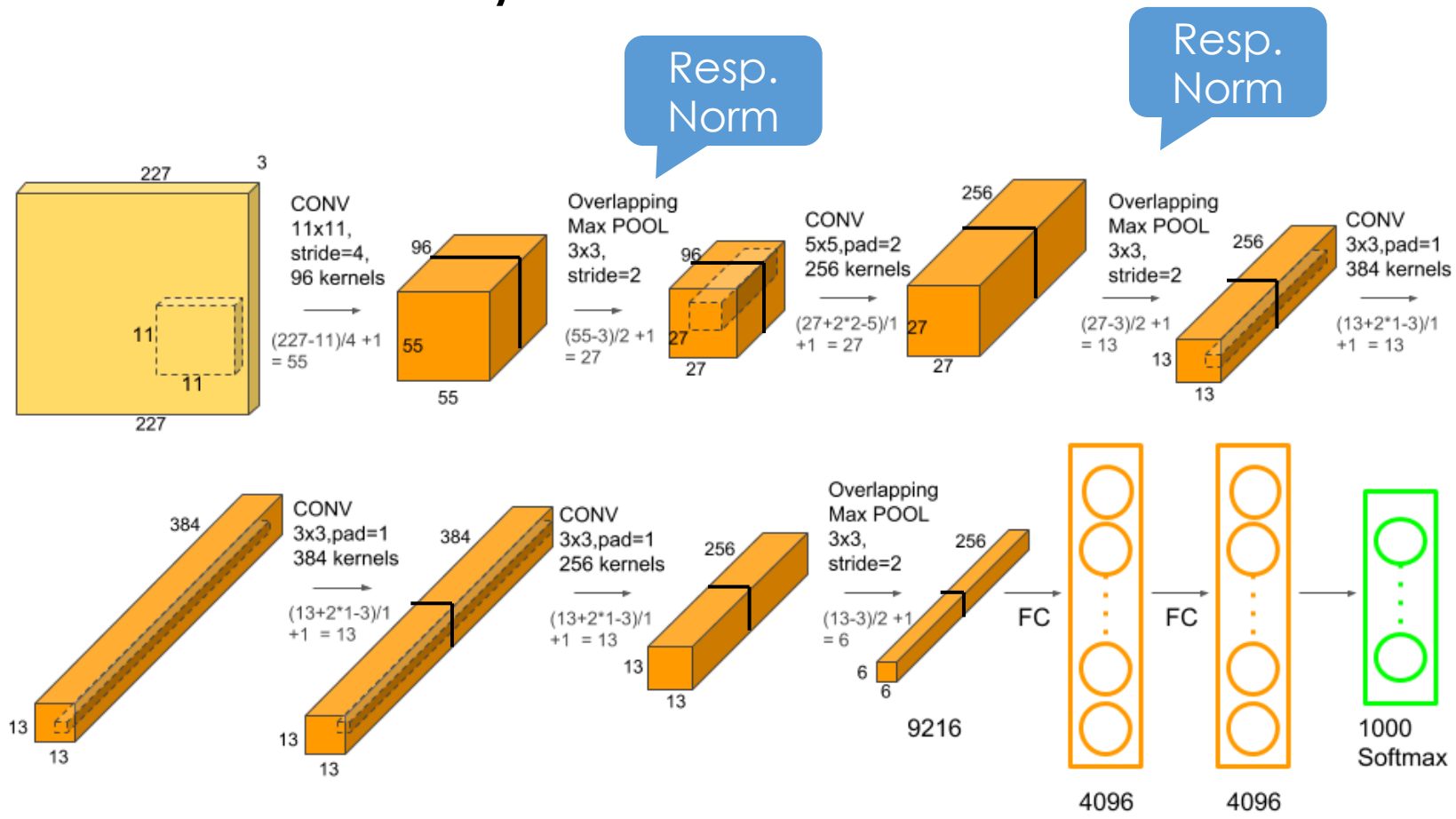
*Posthumously Named

The AlexNet* Architecture
Without GPU Partitioning



*Posthumously Named

The AlexNet* Architecture
(Cleaner Presentation)

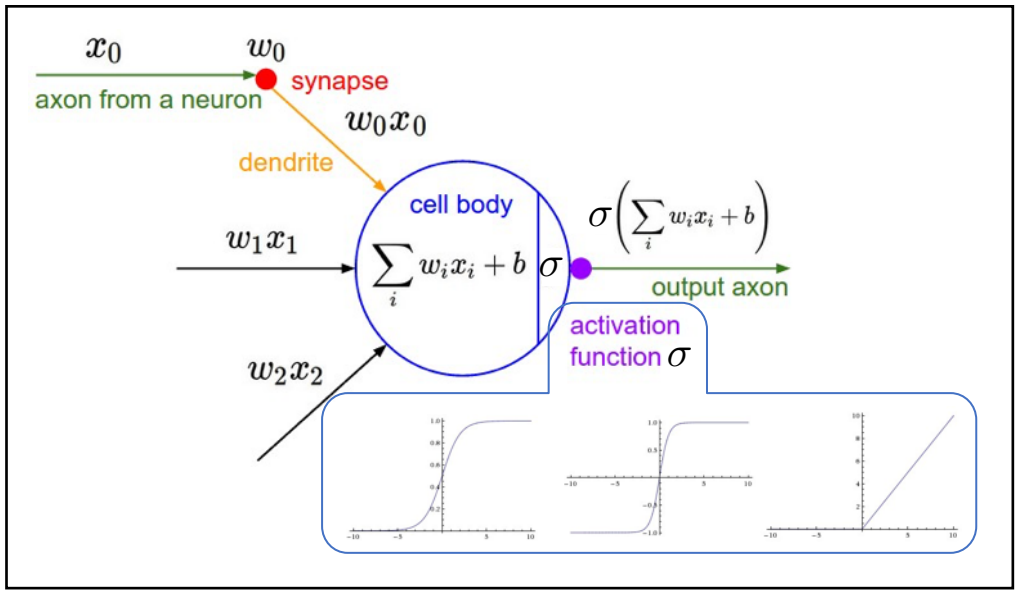


ReLU Nonlinearities are applied after every conv and FC layer

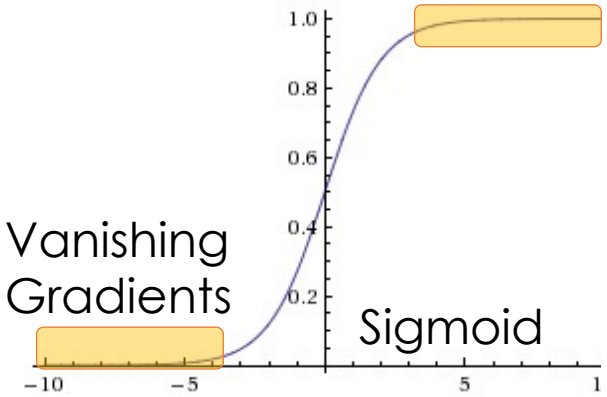
Image from https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/

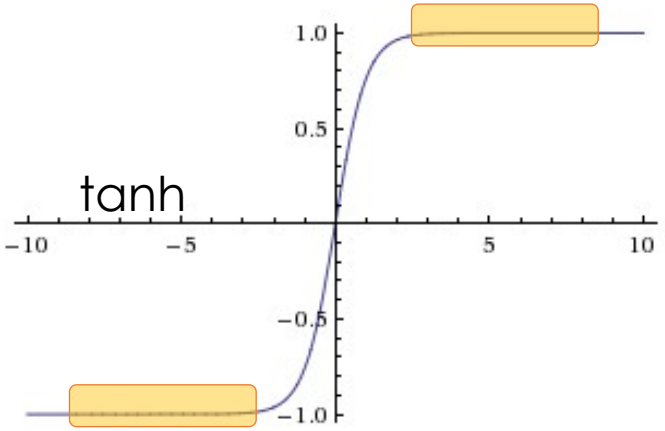
Rectified Linear Units (ReLU)

Recall

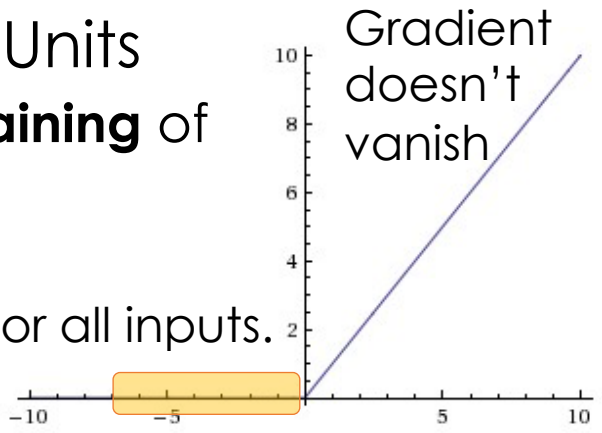


- Historically it was common to use





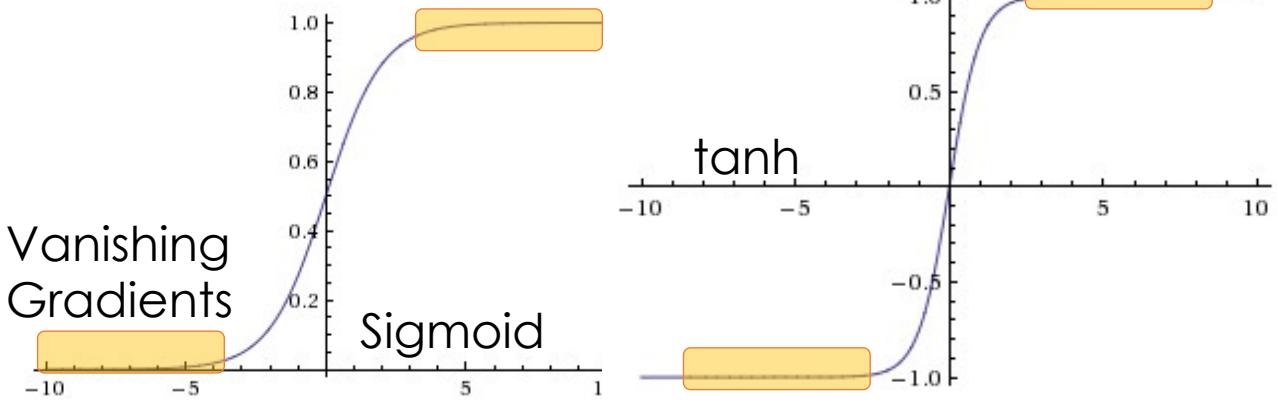
- Rectified Linear Units
  - **Enable faster training** of deep networks.



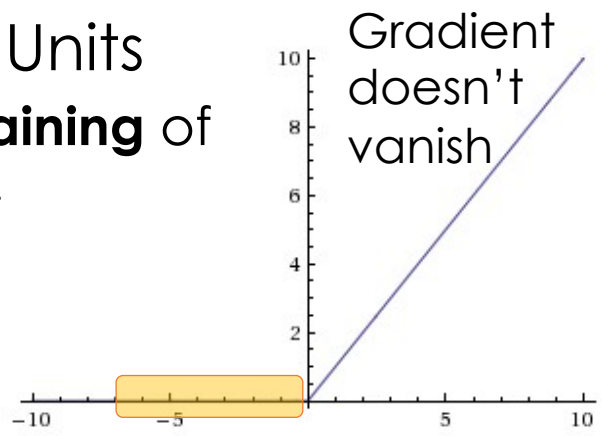
"*Dying ReLU*" is only an issue if an activation is zero for all inputs.

Rectified Linear Units (ReLU)

- Historically it was common to use

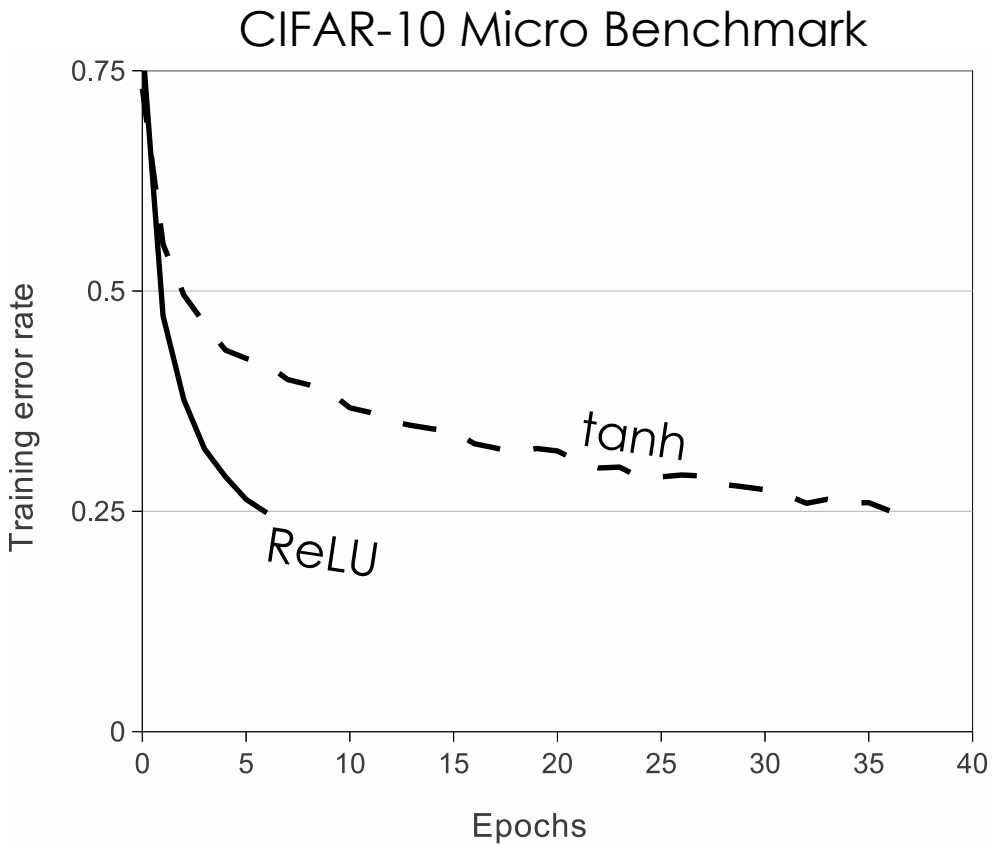


- Rectified Linear Units
  - **Enable faster training** of deep networks.



"Dying ReLU" is only an issue if an activation is zero for all inputs.

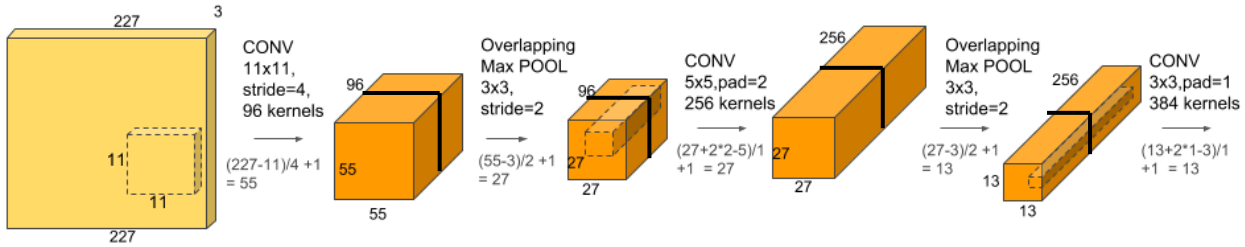
Lots of ablation studies



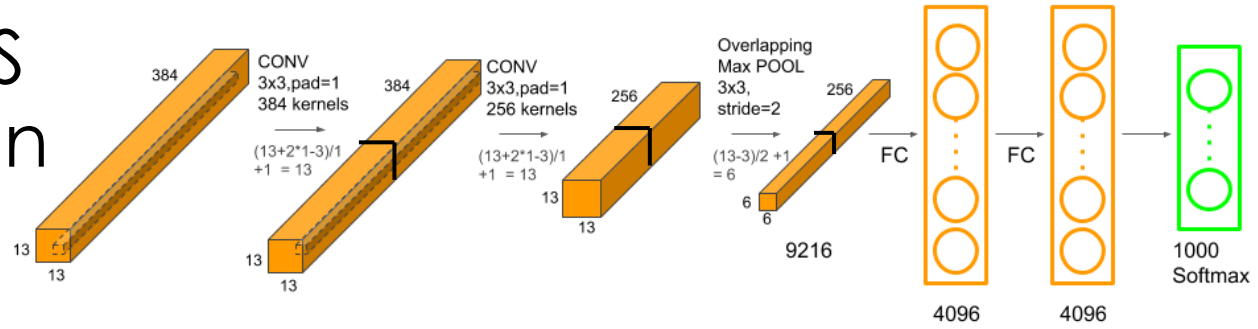
Passes through dataset.

Training on Multiple GPUs

- Limited by GPU **memory** using Nvidia GTX 580 (3GB RAM)
  - 60M Parameters ~ **240 MB**
  - Need to cache activation maps for backpropagation
    - Batch size = 128
    - 128 * (227*227*3 + 55*55*96*2 + 96*27*27 + 256*27*27*2 + 256*13*13 + 13*13*384 + 384*13*13 + 256*13*13 + 4096 + 4096 + 1000) Parameters ~ **718MB**
    - That assuming no overhead and single precision values

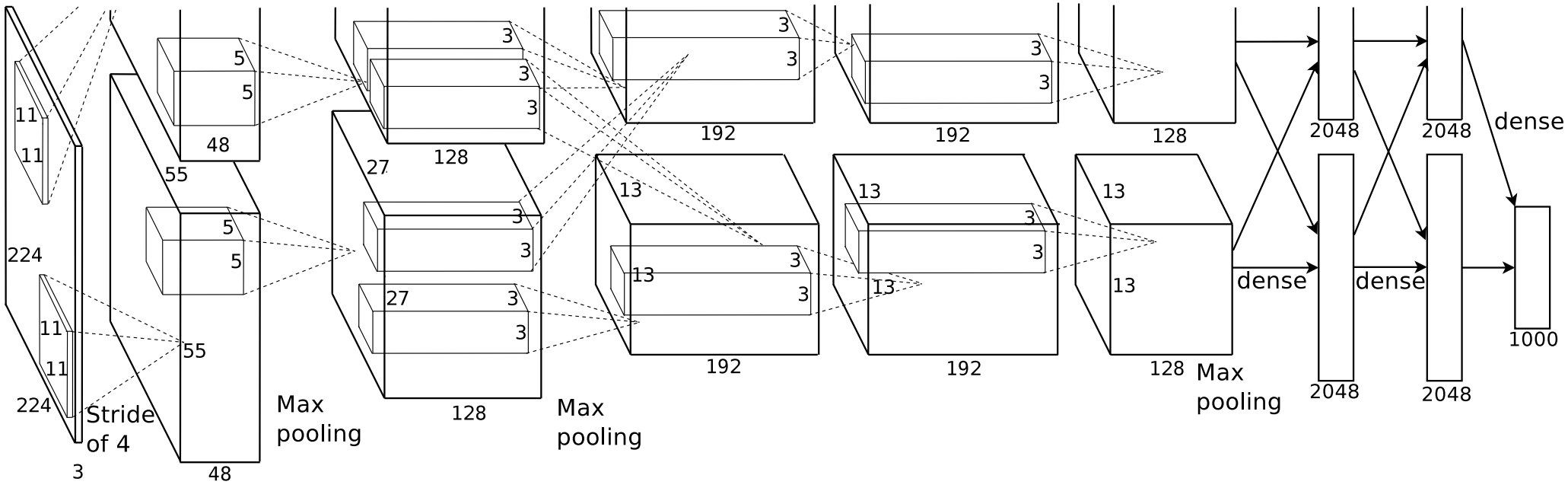


- Tuned splitting across GPUS to balance communication and computation



The Actual AlexNet* Architecture

from the paper



*Posthumously Named

Interesting Consequence of Partitioned Training

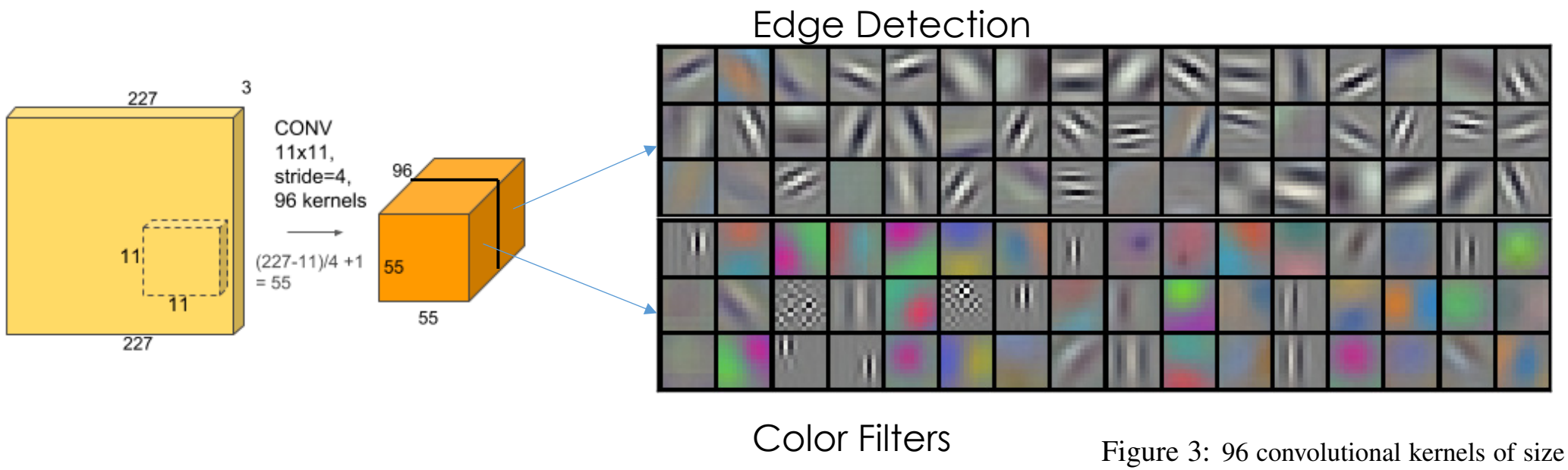
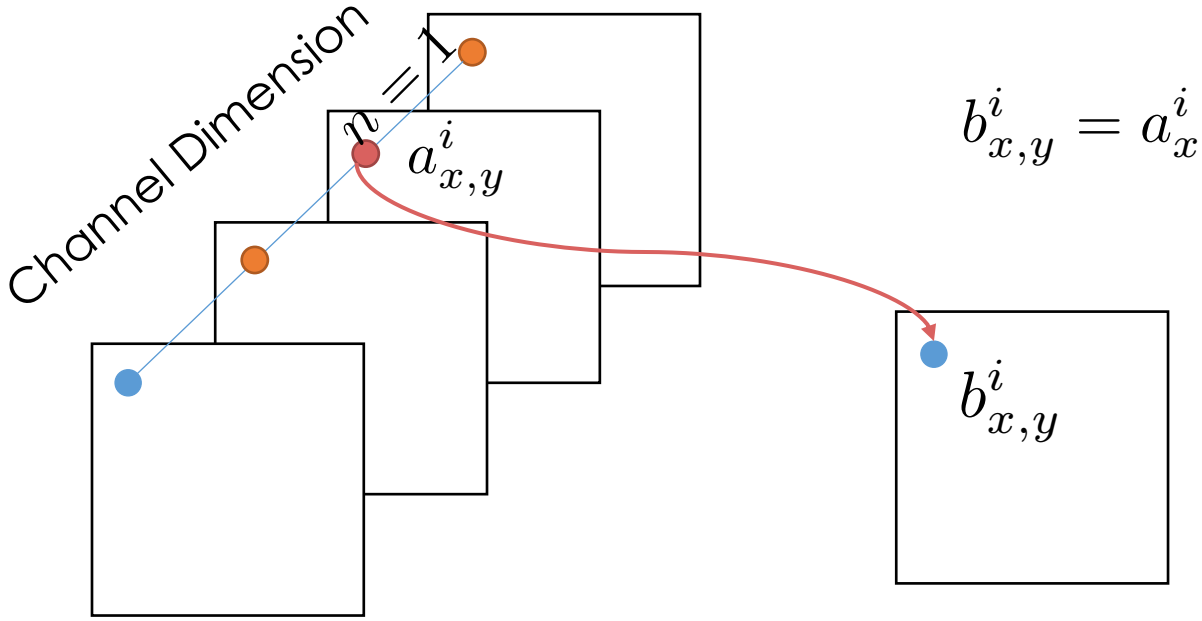


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

Local Response Normalization

- The use of ReLU non-linearity → large activation values
  - Needed a mechanism to normalize large activations



$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

Parameters used in the paper:

$$k = 2, \quad n = 5, \quad \alpha = 10^{-4}, \quad \beta = 0.75$$

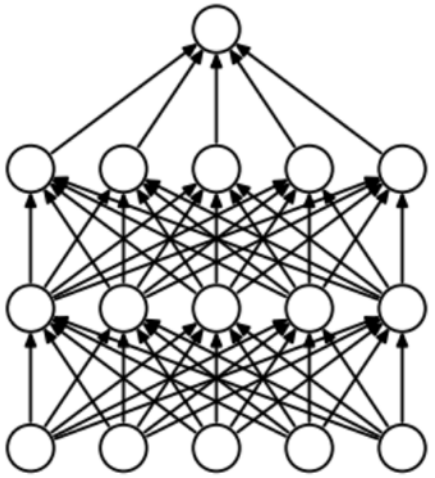
Determined using cross validation on a smaller network.

Data Augmentation
(Established Technique)

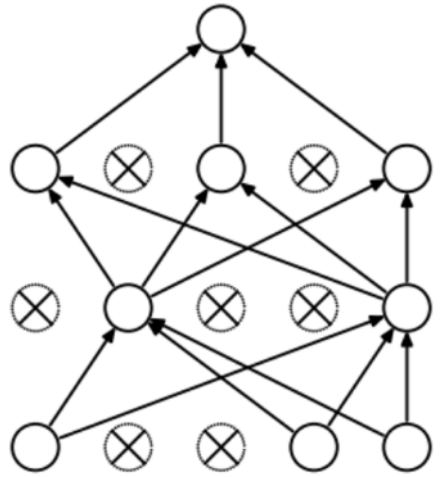
- Used small translations and reflections of images
- Added random transformations along eigen color-space.
- Image transformations done in Python on CPU while GPU is training on previous batch.
  - Today this is increasingly a bottleneck

Dropout on Fully Connected Layers
(Established Technique.)

- Randomly zero activations (with prob. 0.5) during training
- Reduces co-adaptation of neurons
- Ensures more robust features
- Simulates training an ensemble of sparse architectures



(a) Standard Neural Net



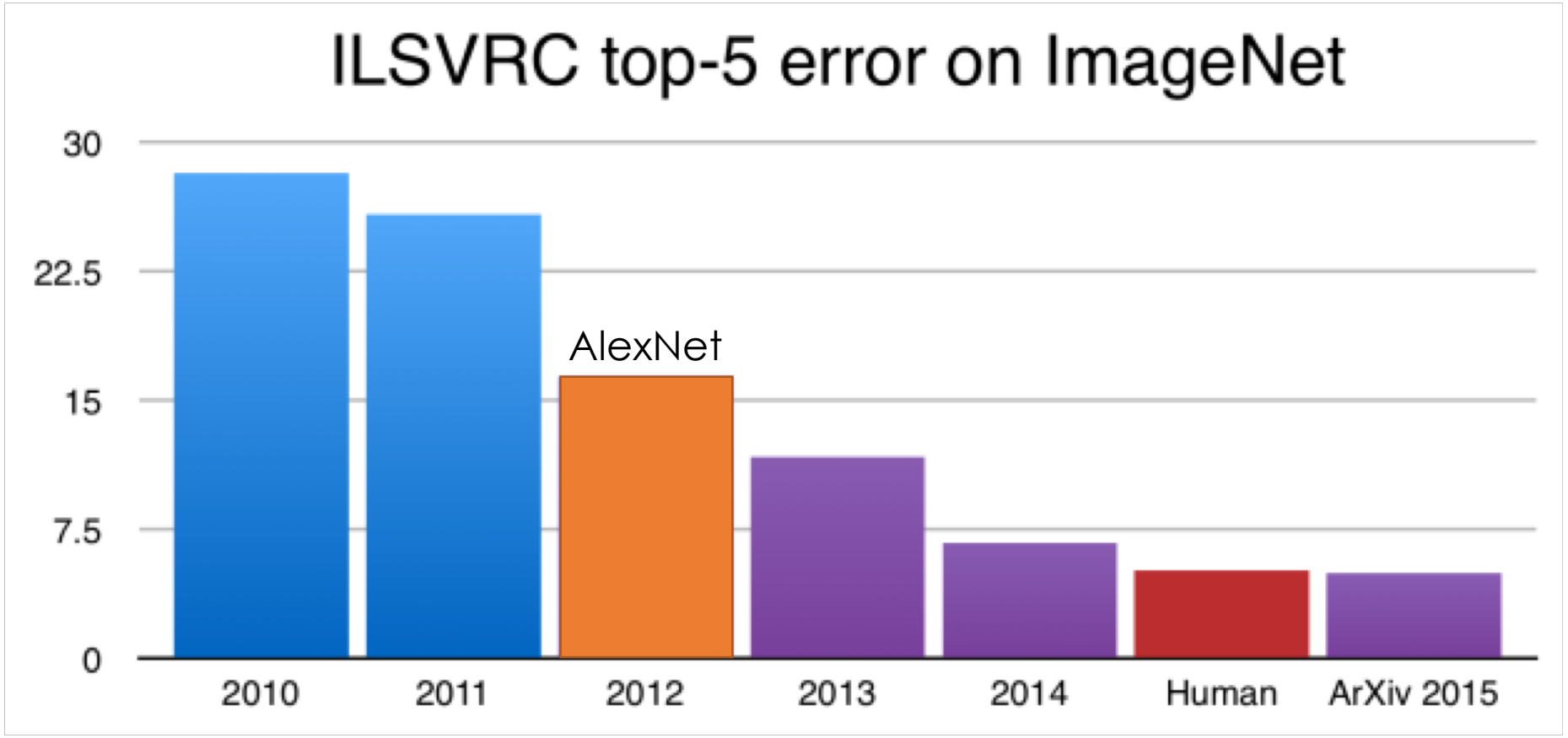
(b) After applying dropout.

Key Results

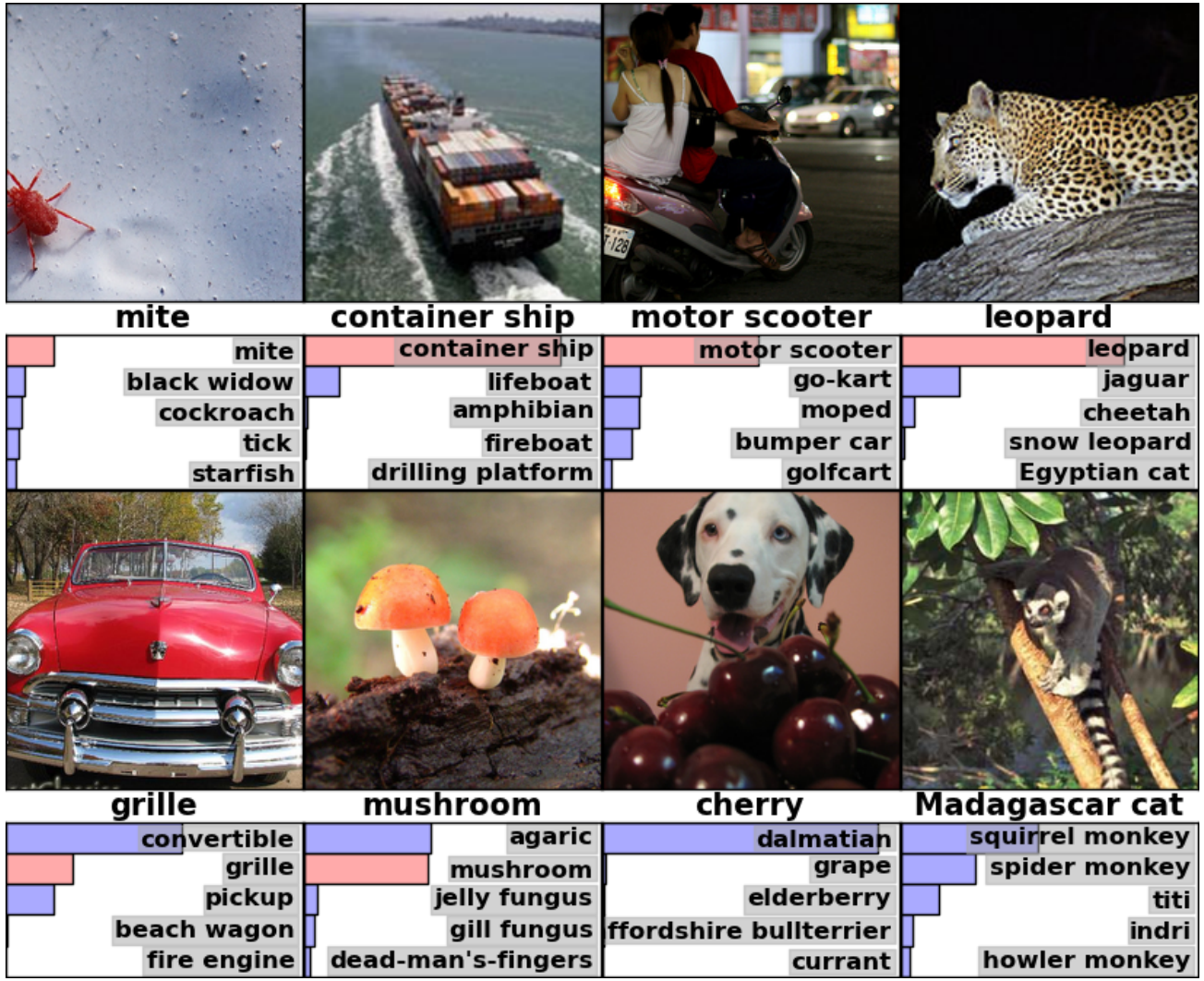
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

Put into historical context



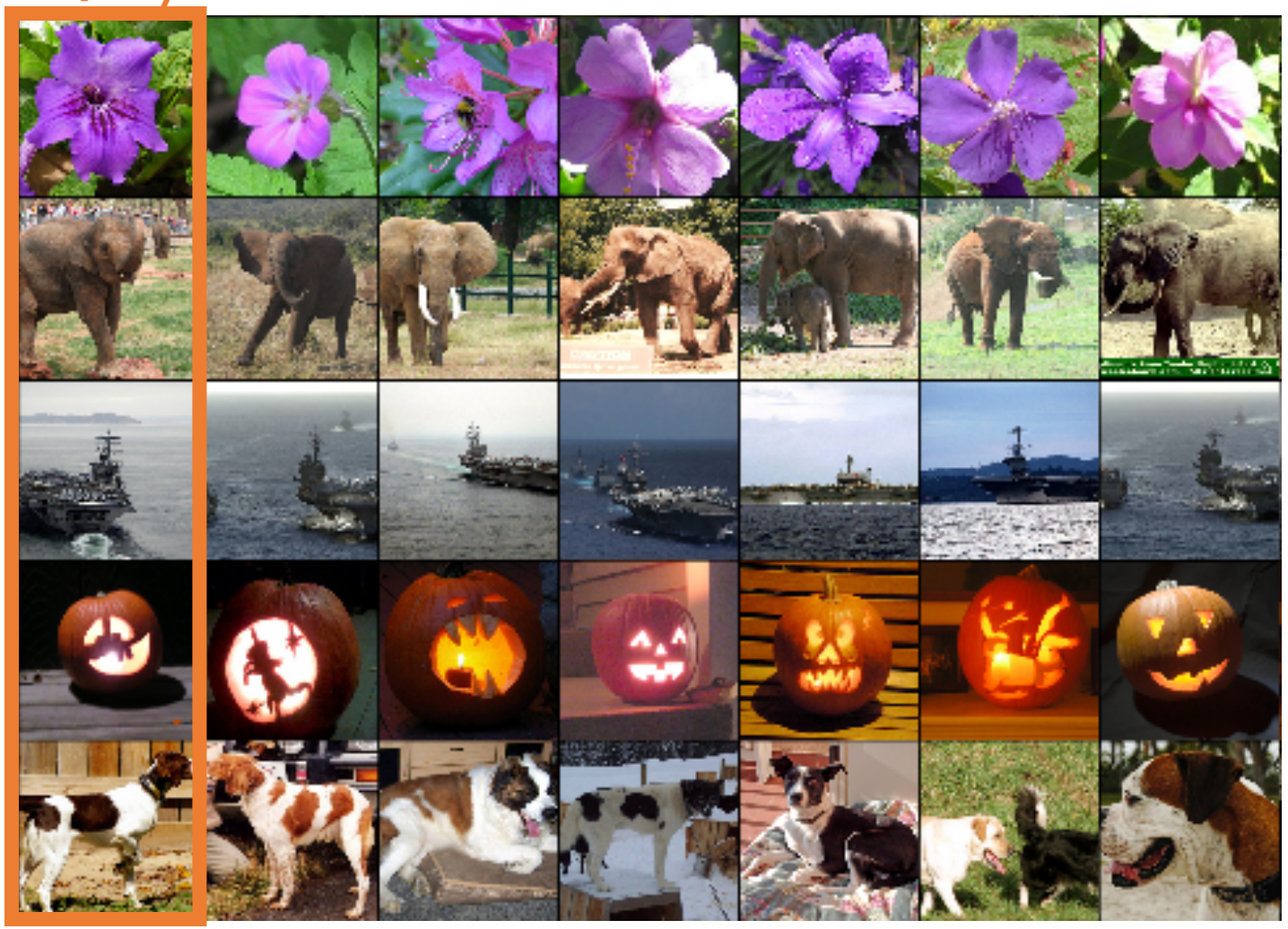
Qualitative Results



Good Embeddings …

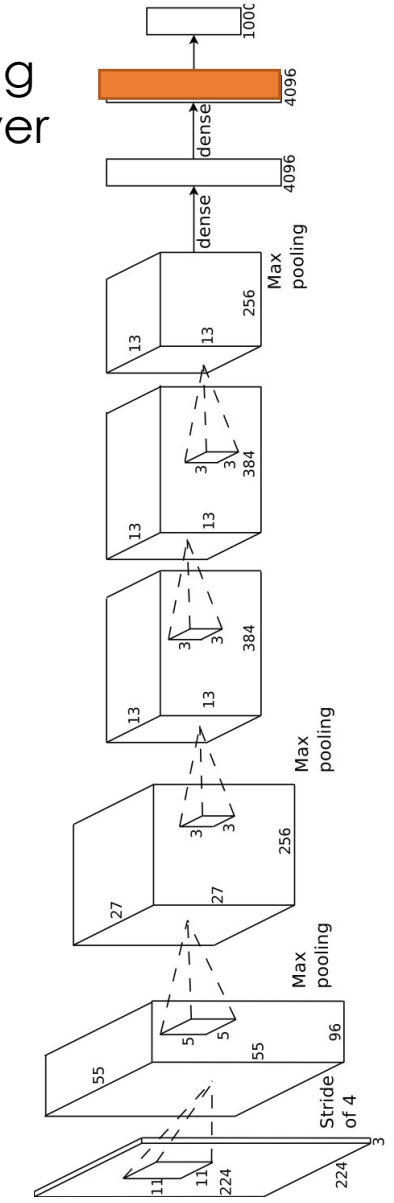
This will later be the foundation of **many** papers

Query



Images with largest dot product with query

Embedding Layer



Limitations and Opportunities

- Model architecture is determined by system, what happens when GPUs get faster or bigger?
- Multiple fully connected layers → many parameters
- Images are assumed fixed input size
- Many hyper-parameters to tune
- Implementation in CUDA is hard to extend

Long term impact

- Cited by over 34,000 papers
- Used as features in many other computer vision models
- Follow-up work on Caffe made the models more accessible
- The model, code, and many of the techniques have largely been replaced but this work shaped contemporary computer vision and helped elevate deep learning in the ML and CV communities.

ImageNet 2012 → 2015