# Deep Learning:
## Searching for Images

Emily Fox & Carlos Guestrin
Machine Learning Specialization
University of Washington

# Visual product recommender

# I want to buy new shoes, but…





Too many options online…




# Text search doesn't help...





"Dress shoes"

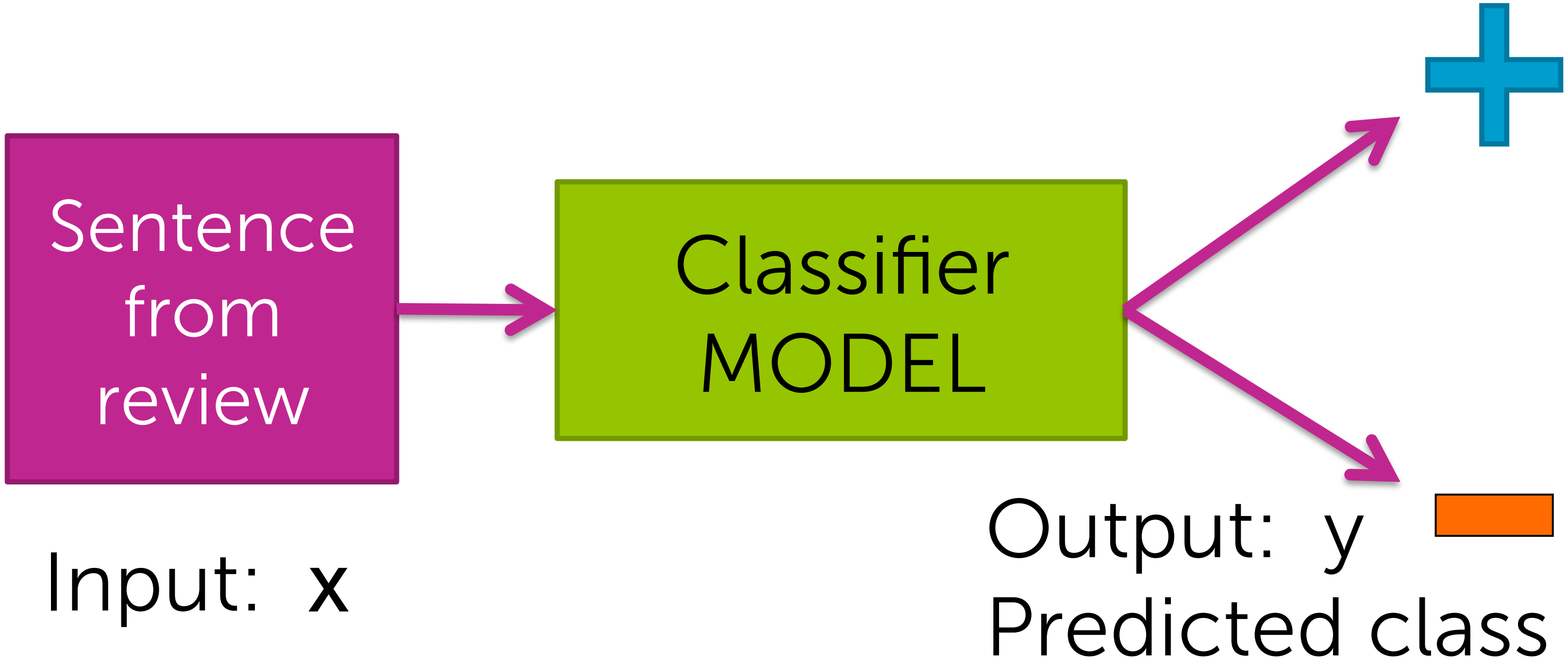




# Visual product search demo

# Features are key to machine learning

# Goal: revisit classifiers, but using more complex, non-linear features



# Image classification


Input: x
Image pixels

Output: y
Predicted object

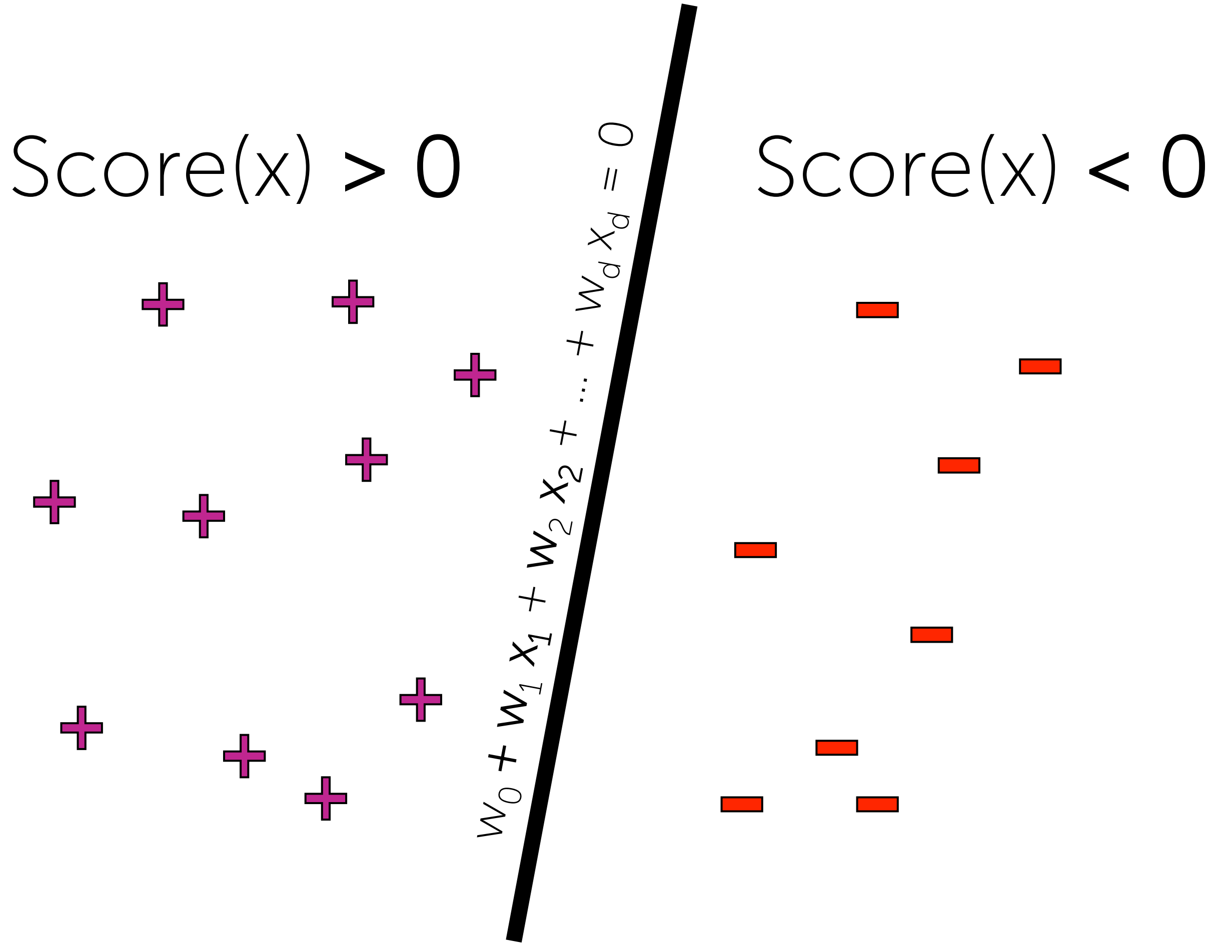
Neural networks

↓

Learning *very* non-linear features

# Linear classifiers

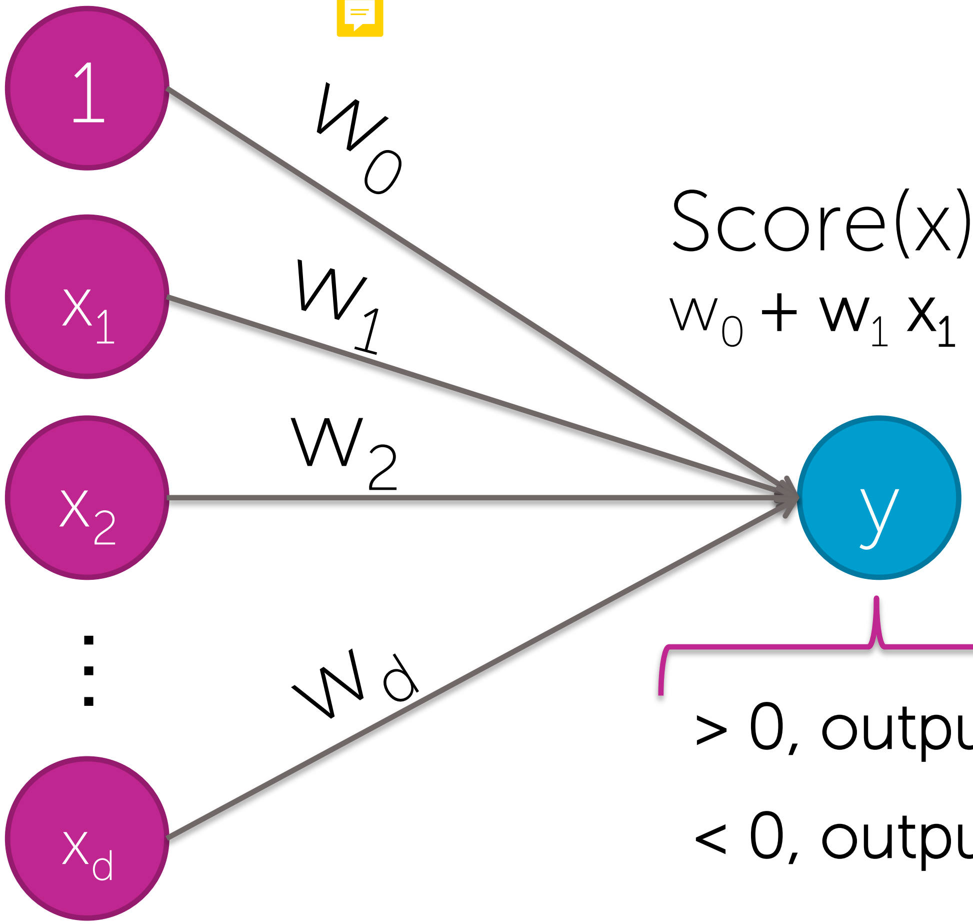
$$\text{Score}(x) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d$$



# Graph representation of classifier: useful for defining neural networks

Input

Output




Score(x) =

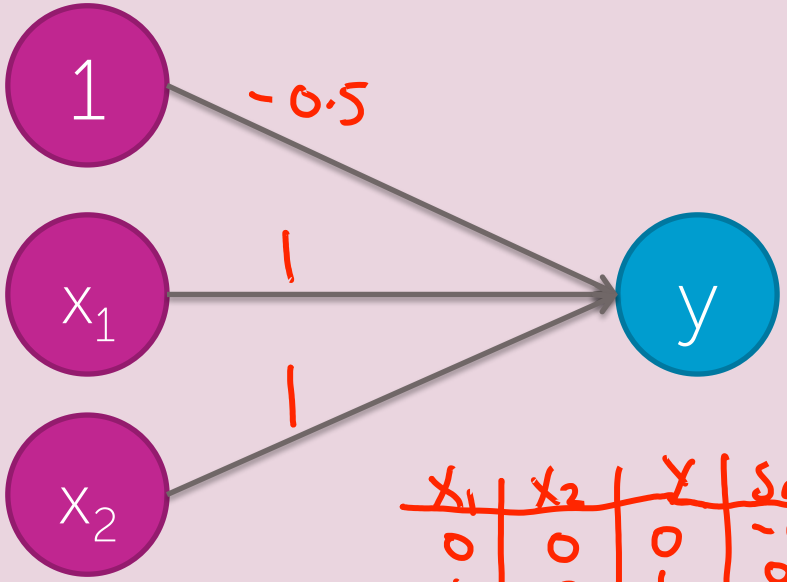
$w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d$

> 0, output 1

< 0, output 0

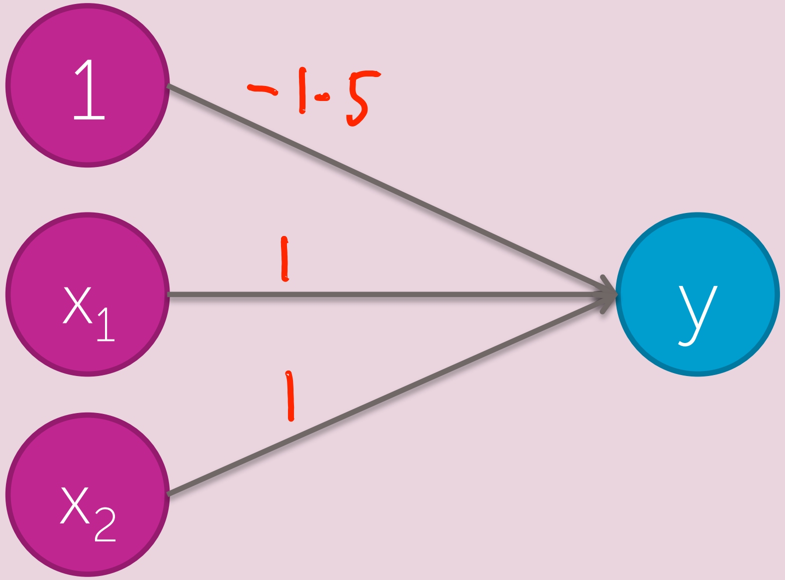
# What can a linear classifier represent?

## $x_1$ OR $x_2$

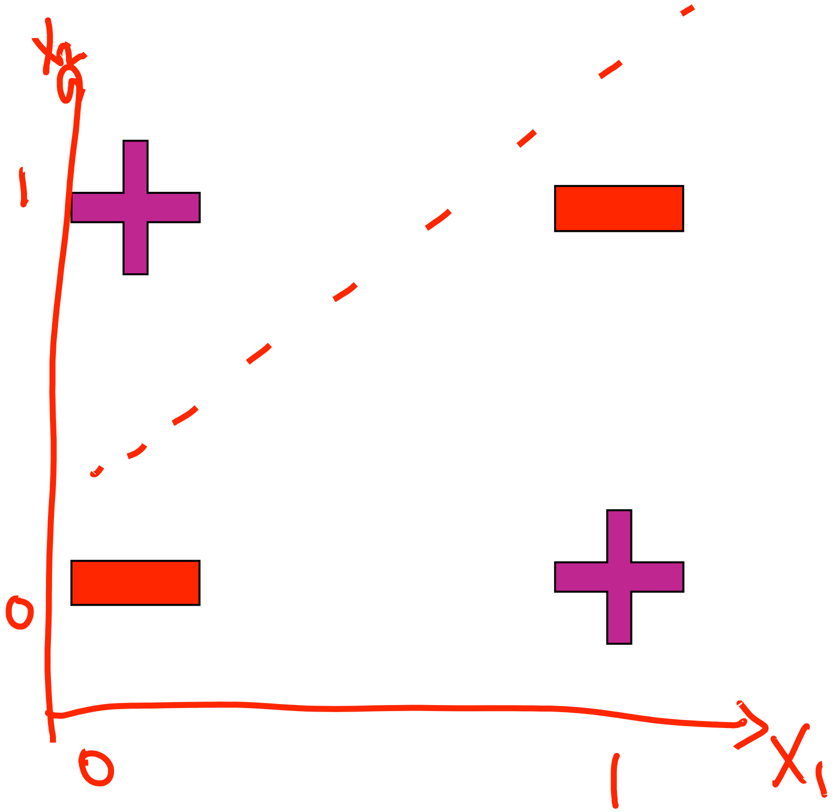



| $x_1$ | $x_2$ | $y$ | Score |
|---|---|---|---|
| 0 | 0 | 0 | -0.5 |
| 1 | 0 | 1 | 0.5 |
| 0 | 1 | 1 | 0.5 |
| 1 | 1 | 1 | 1.5 |

## $x_1$ AND $x_2$



# What can't a simple linear classifier represent?

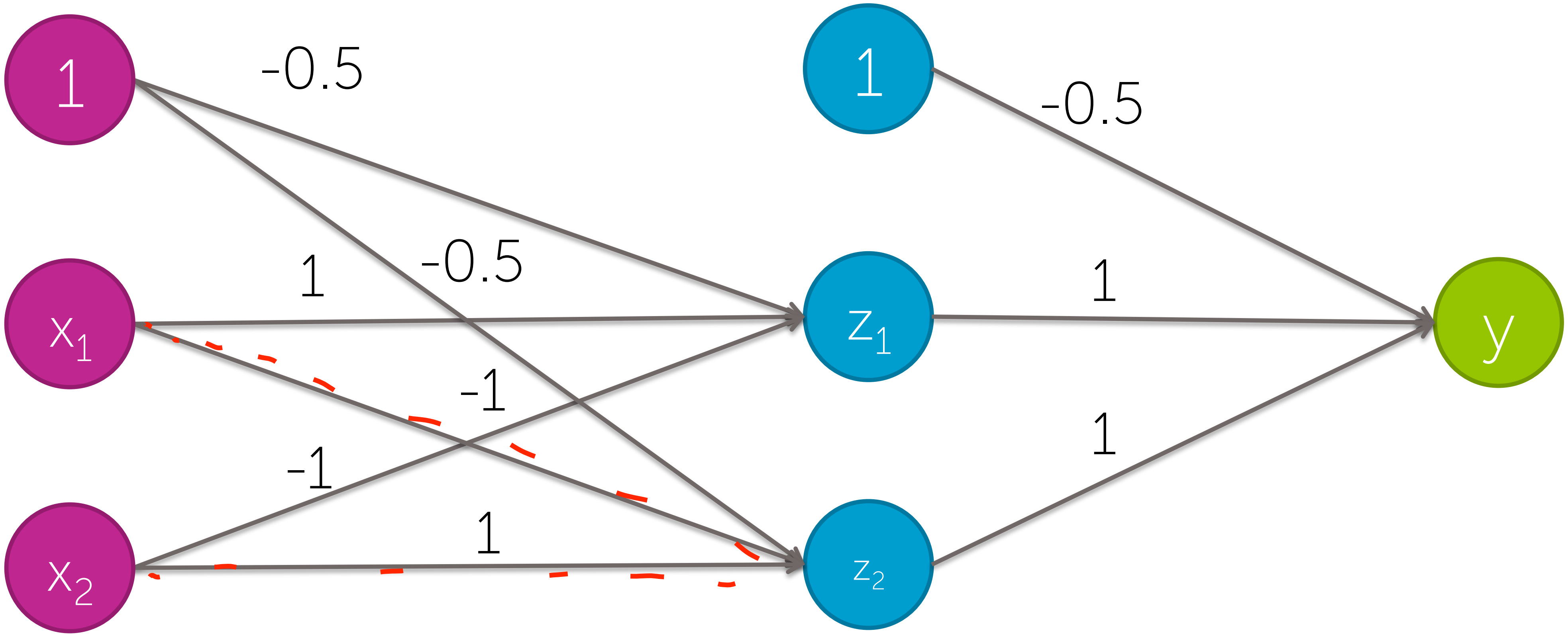


**XOR**
the counterexample to everything

**Need non-linear features**

# Solving the XOR problem: Adding a layer

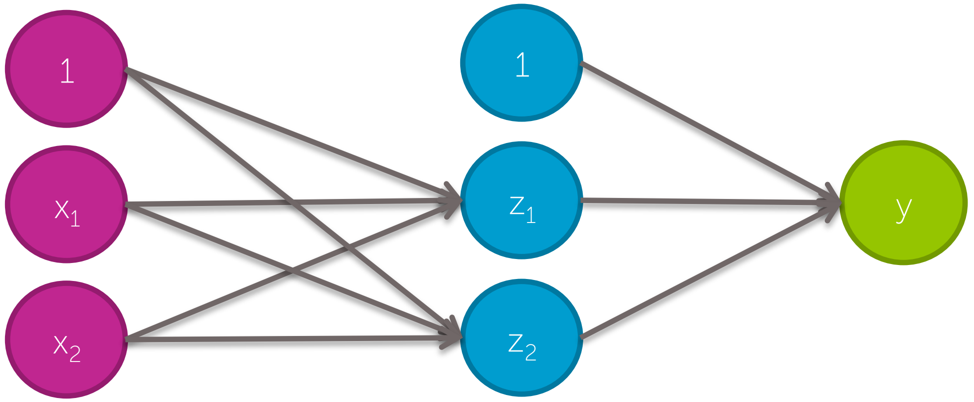
$$\text{XOR} = \underbrace{x_1 \text{ AND NOT } x_2}_{z_1} \text{ OR } \underbrace{\text{NOT } x_1 \text{ AND } x_2}_{z_2}$$




Thresholded to 0 or 1

# A neural network

- Layers and layers and layers of linear models and non-linear transformations

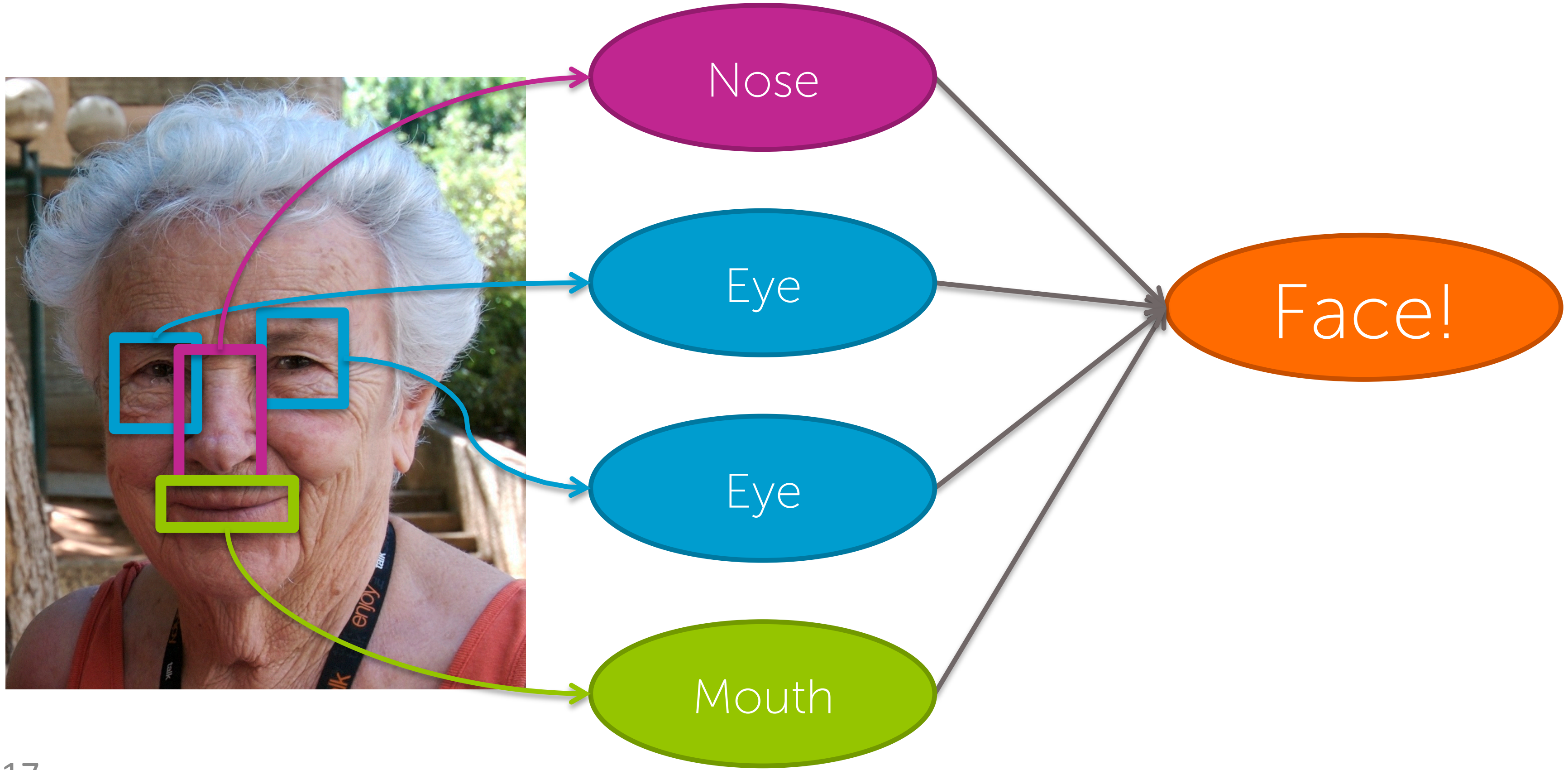



- Around for about 50 years
  - Fell in "disfavor" in 90s
- In last few years, big resurgence
  - Impressive accuracy on several benchmark problems
  - Powered by huge datasets, GPUs, & modeling/learning alg improvements

# Application of deep learning to computer vision

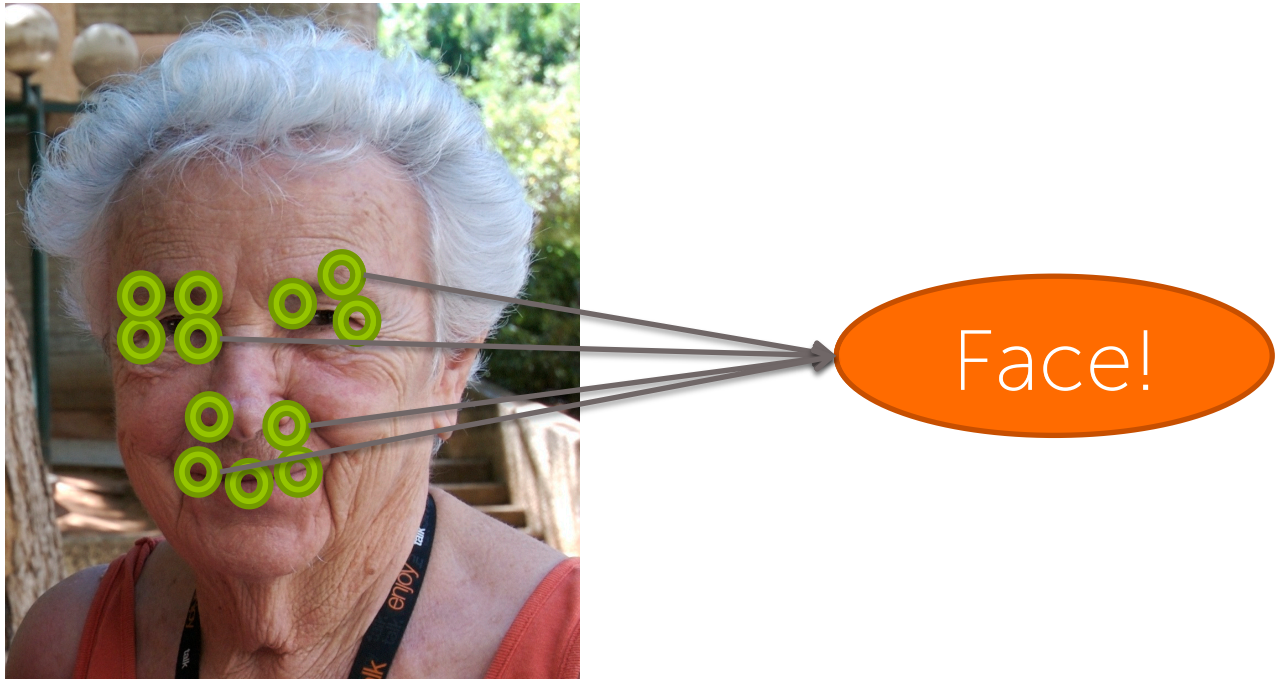
# Image features

- Features = local detectors
  - Combined to make prediction
  - (in reality, features are more low-level)

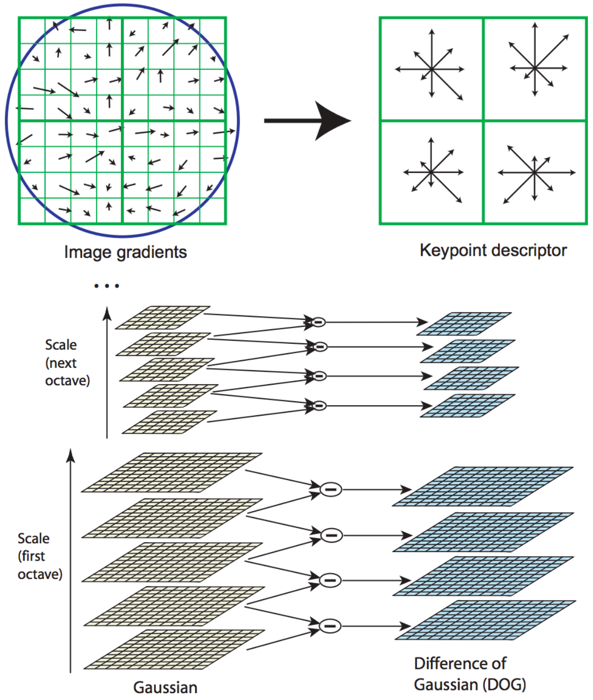


# Typical local detectors look for locally "interesting  points" in image

- *Image features*: collections of locally interesting points
  - Combined to build classifiers



# Many hand created features exist for finding interest points…



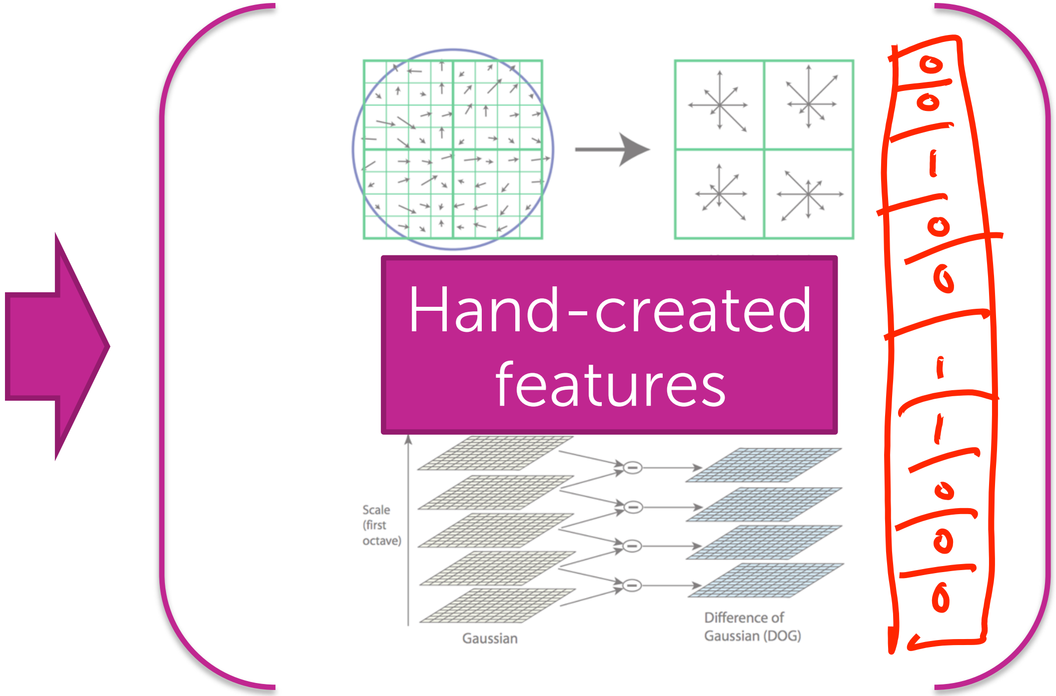

*SIFT* [Lowe '99]

- *Spin Images* [Johnson & Herbert '99]
- *Textons* [Malik et al. '99]
- *RIFT* [Lazebnik '04]
- *GLOH* [Mikolajczyk & Schmid '05]
- *HoG* [Dalal & Triggs '05]
- …

# Standard image classification approach

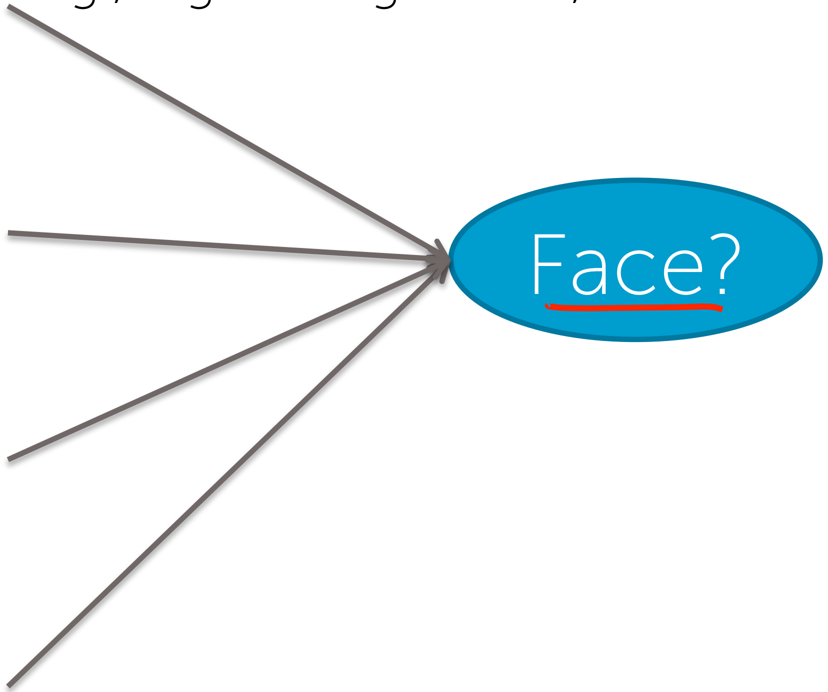
Input

Extract features

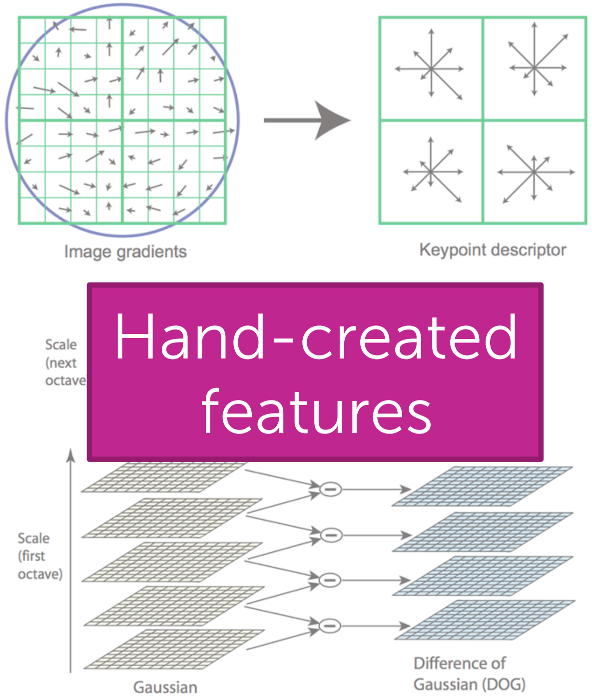



Use simple classifier

e.g., logistic regression, SVMs



# Many hand created features exist for finding interest points…




*SIFT* [Lowe '99]

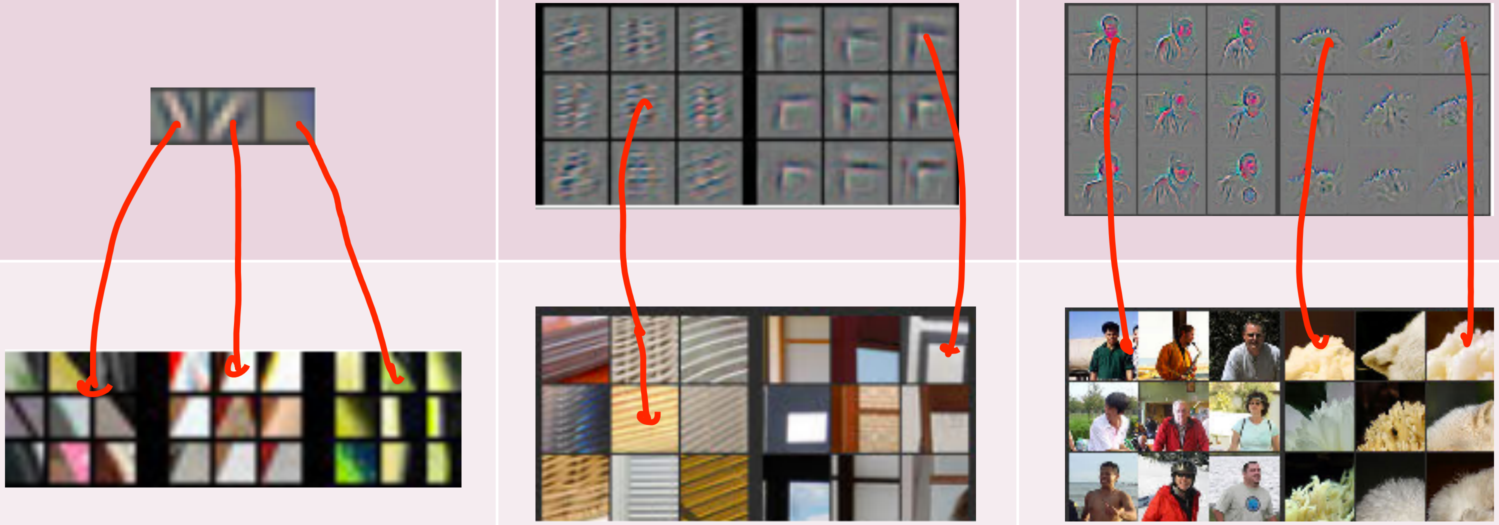
- *Spin Images* [Johnson & Herbert '99]
- *Textons* [Malik et al. '99]
- *RIFT* [Lazebnik '04]
- *GLOH* [Mikolajczyk & Schmid '05]
- *HoG* [Dalal & Triggs '05]
- …

# … but very painful to design

# Deep learning:
# *implicitly learns features*

Layer 1 → Layer 2 → Layer 3 → Prediction

| | Layer 1 | Layer 2 | Layer 3 |
|---|---|---|---|
| Example detectors learned | | | |
| Example interest points detected | | | |



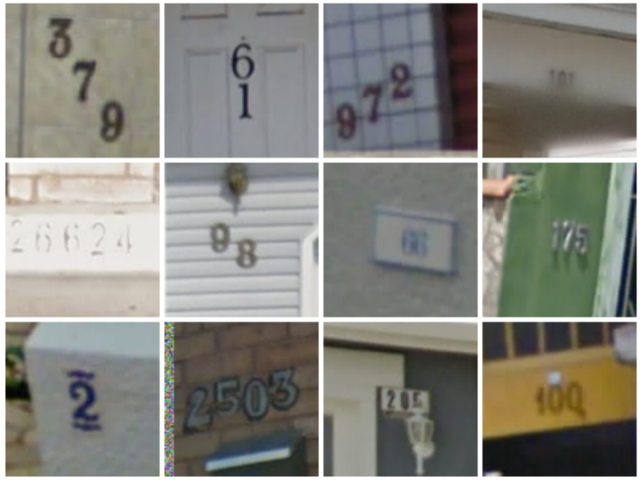
[Zeiler & Fergus '13]

# Deep learning performance

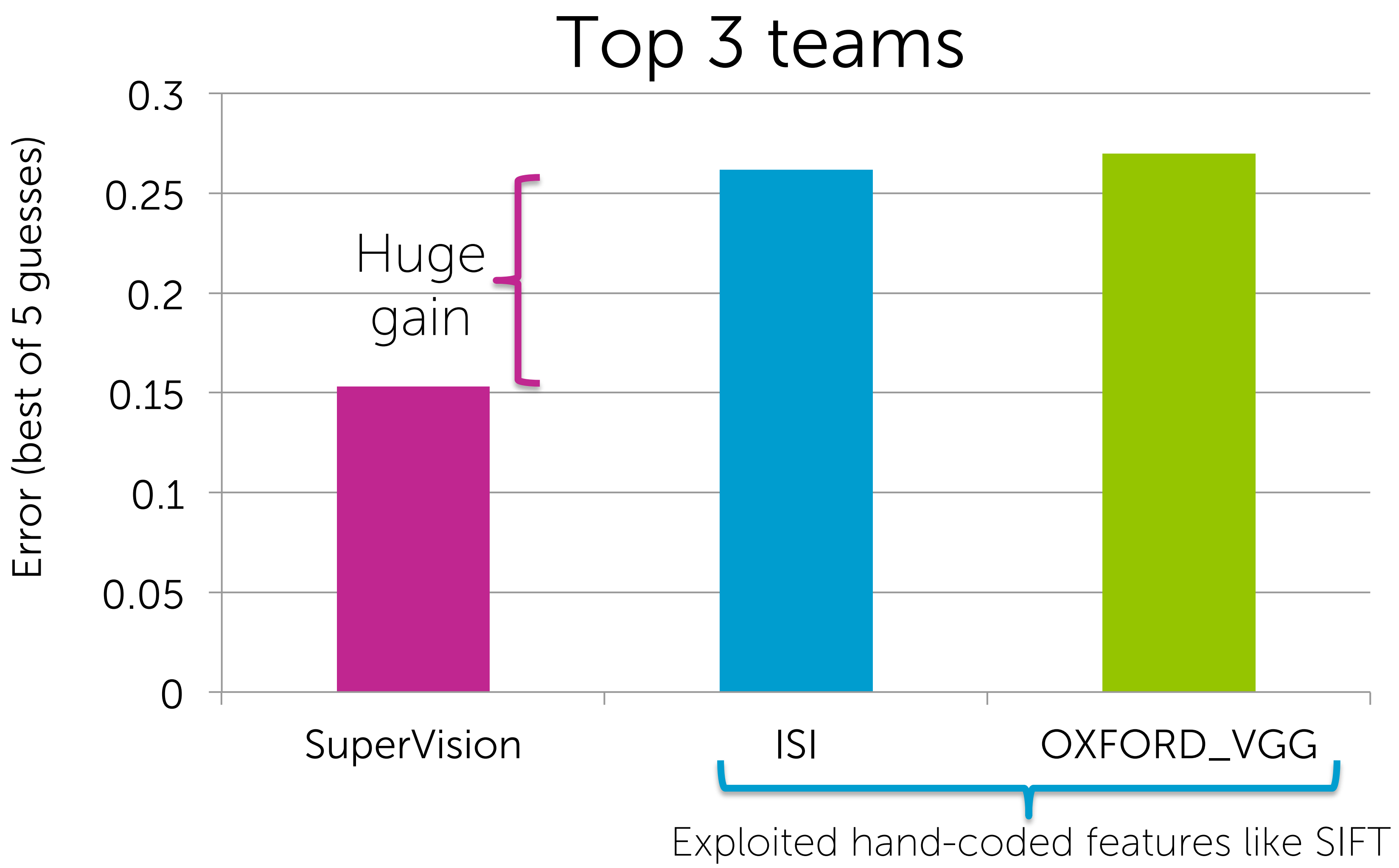
# Sample results using deep neural networks

- German traffic sign recognition benchmark
  – 99.5% accuracy (IDSIA team)


- House number recognition
  – 97.8% accuracy per character [Goodfellow et al. '13]

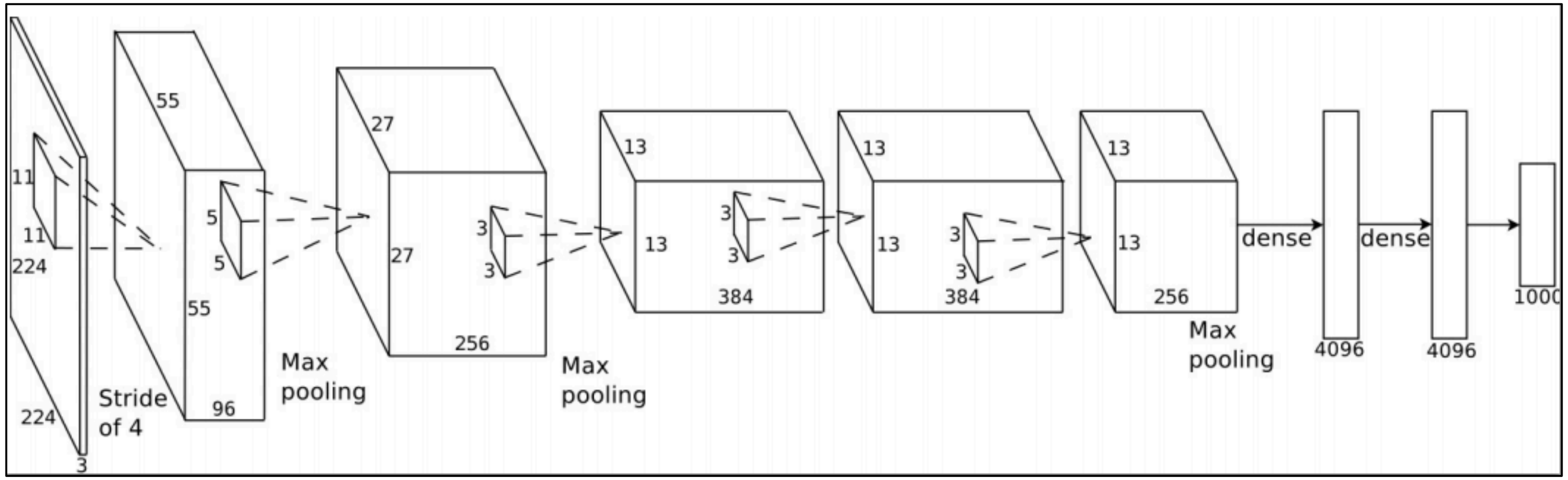


# ImageNet 2012 competition: 1.2M training images, 1000 categories



# ImageNet 2012 competition: 1.2M training images, 1000 categories

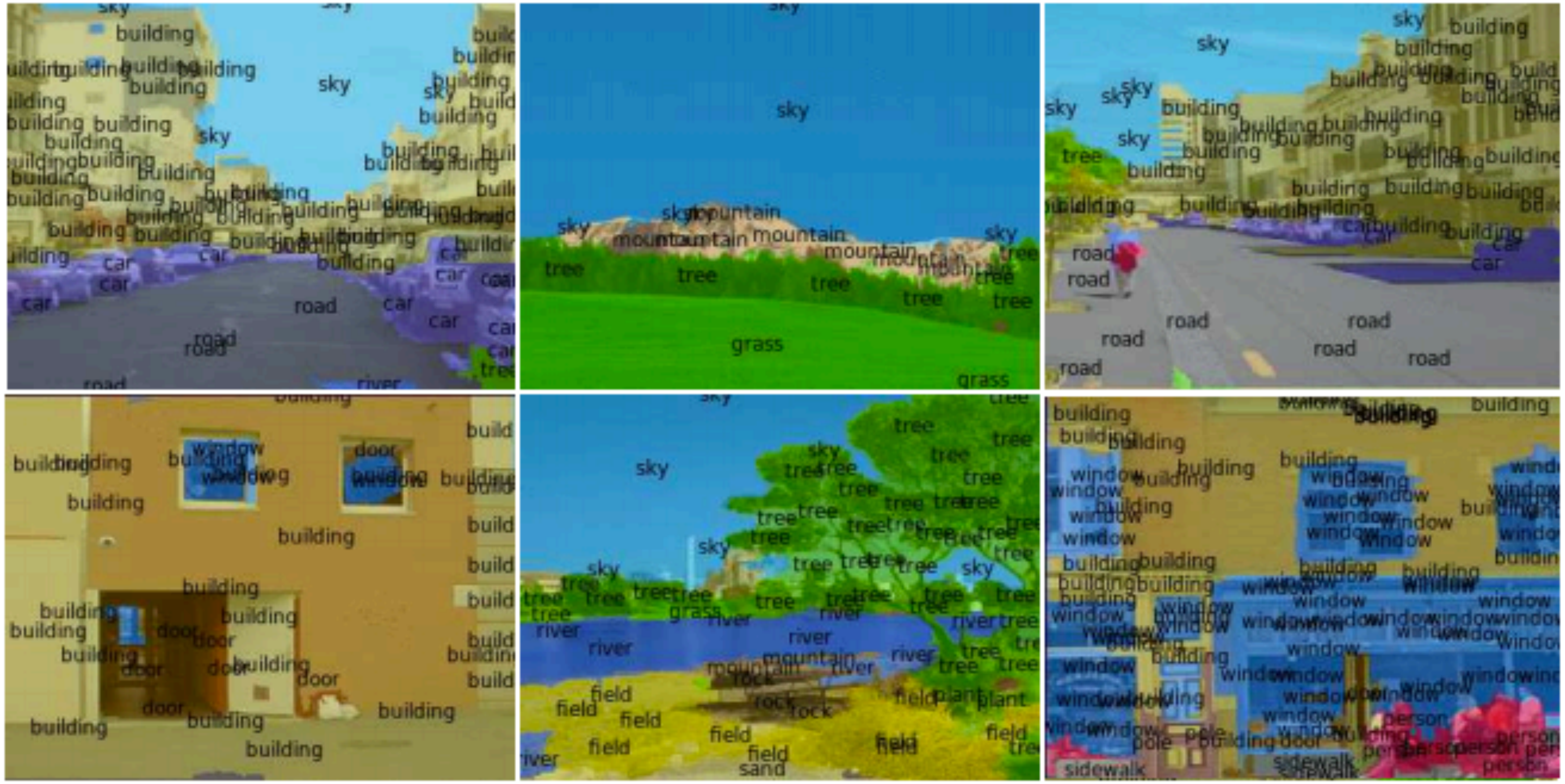
Winning entry: SuperVision
8 layers, 60M parameters [Krizhevsky et al. '12]




Achieving these amazing results required:
- New learning algorithms
- GPU implementation

# Deep learning in computer vision

# Scene parsing with deep learning




[Farabet et al. '13]

# Retrieving similar images

Input Image

Nearest neighbors

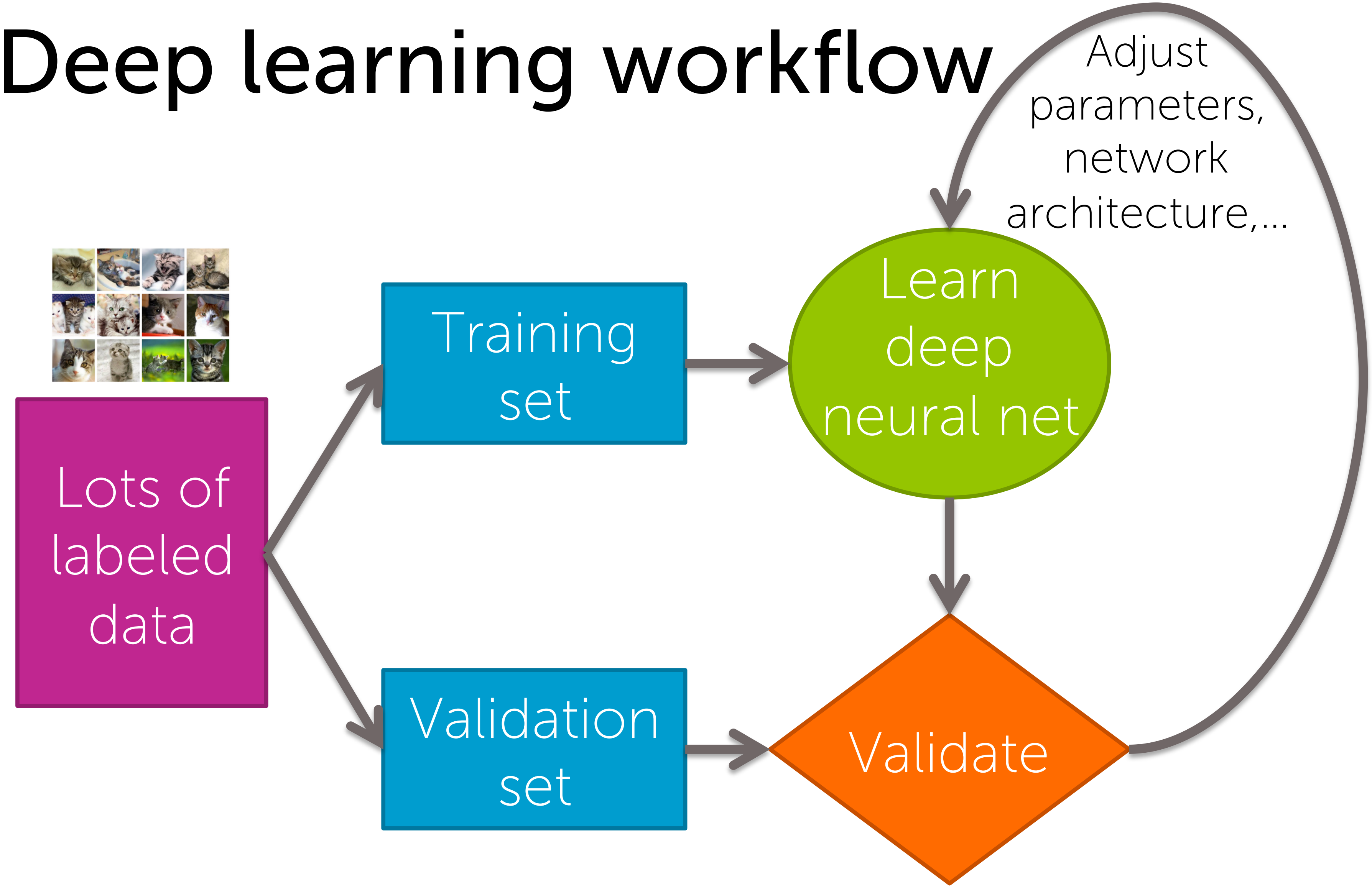
# Challenges of deep learning

# Deep learning score card

## Pros

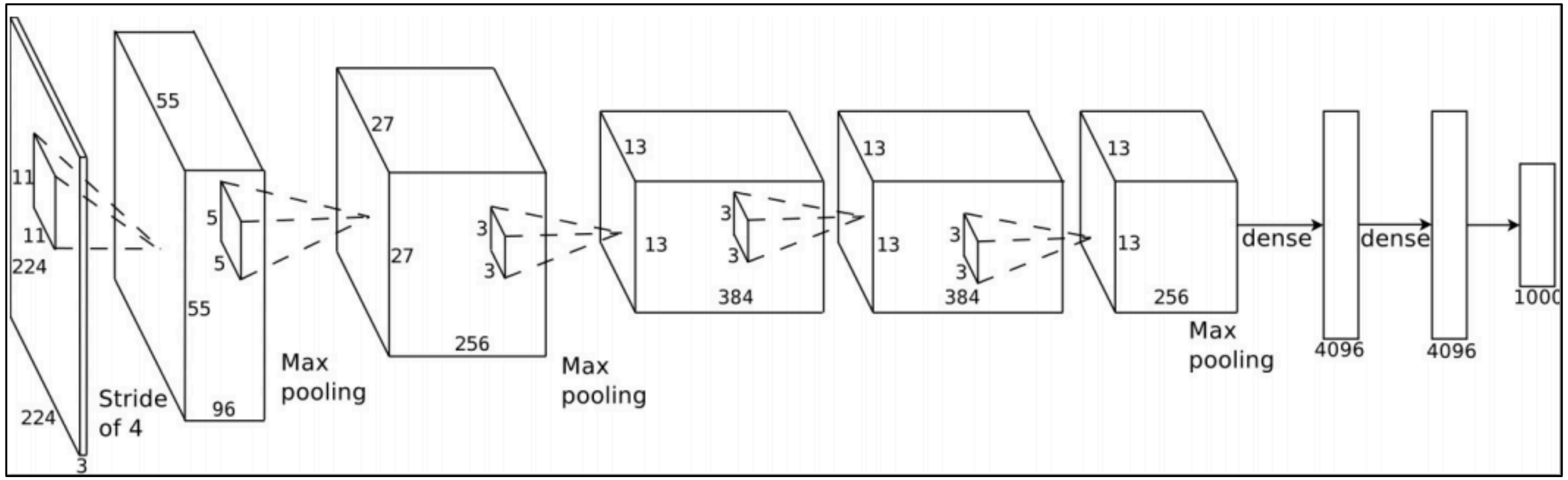
- Enables learning of features rather than hand tuning
- Impressive performance gains
  - Computer vision
  - Speech recognition
  - Some text analysis
- Potential for more impact

# Deep learning workflow



# Many tricks needed to work well…

Different types of layers, connections,…
needed for high accuracy




[Krizhevsky et al. '12]

# Deep learning score card

## Pros

- Enables learning of features rather than hand tuning
- Impressive performance gains
  - Computer vision
  - Speech recognition
  - Some text analysis
- Potential for more impact

## Cons

- Requires a lot of data for high accuracy
- Computationally really expensive
- Extremely hard to tune
  - Choice of architecture
  - Parameter types
  - Hyperparameters
  - Learning algorithm
  - …

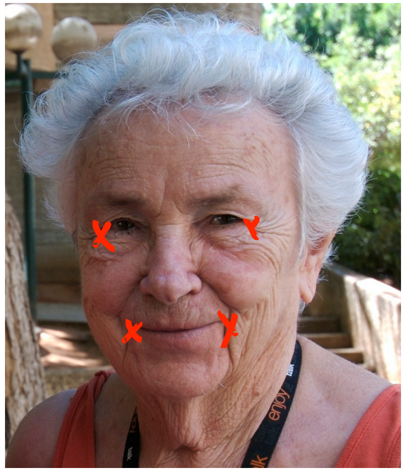
Computational cost+ so many choices
=
incredibly hard to tune

# Deep features:

## Deep learning
## +
## Transfer learning

# Standard image classification approach

Input



Extract features

Can we learn features from data, even when we don't have data or time?

Use simple classifier

e.g., logistic regression, SVMs

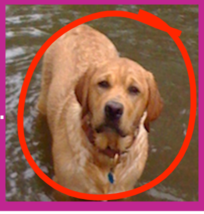
Face?

# Transfer learning: *Use data from one task to help learn on another*

Old idea, explored for deep learning by Donahue et al. '14 & others

Lots of data:

vs.



Learn neural net

Great accuracy on cat v. dog

Some data:

Neural net as feature extractor

+

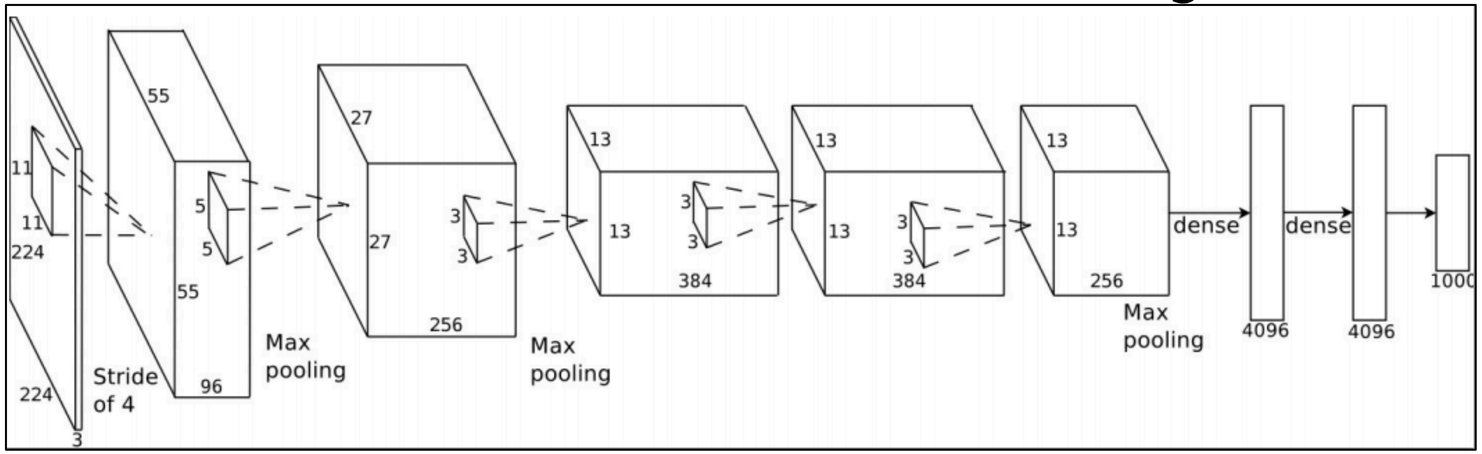
Simple classifier

Great accuracy on 101 categories

# What's learned in a neural net

vs.

Neural net trained for Task 1: cat vs. dog

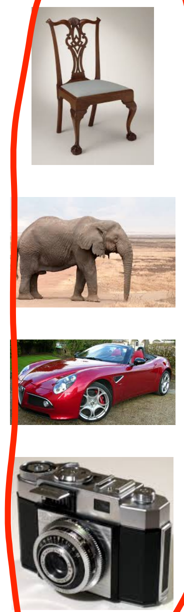



More generic
Can be used as feature extractor

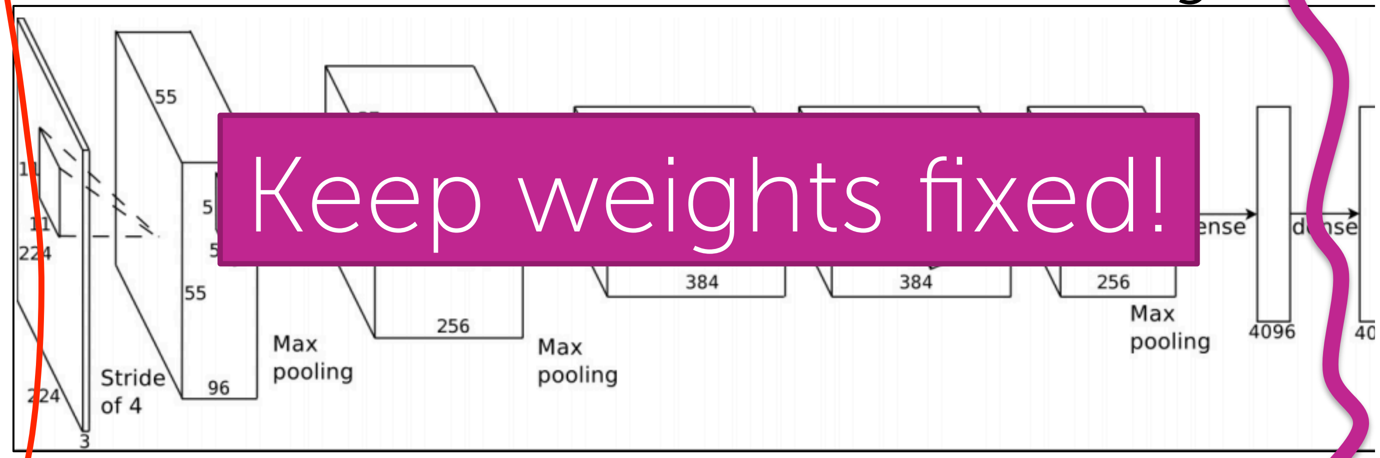
Very specific
to Task 1
Should be ignored
for other tasks

# Transfer learning in more detail…

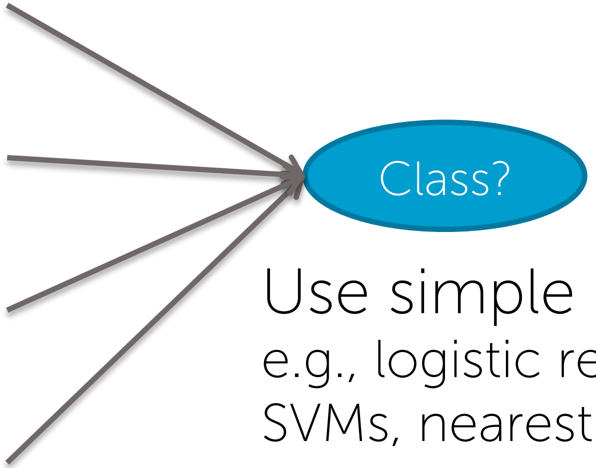
For Task 2, predicting 101 categories,
learn only end part of neural net



Neural net trained for Task 1: cat vs. dog




Keep weights fixed!

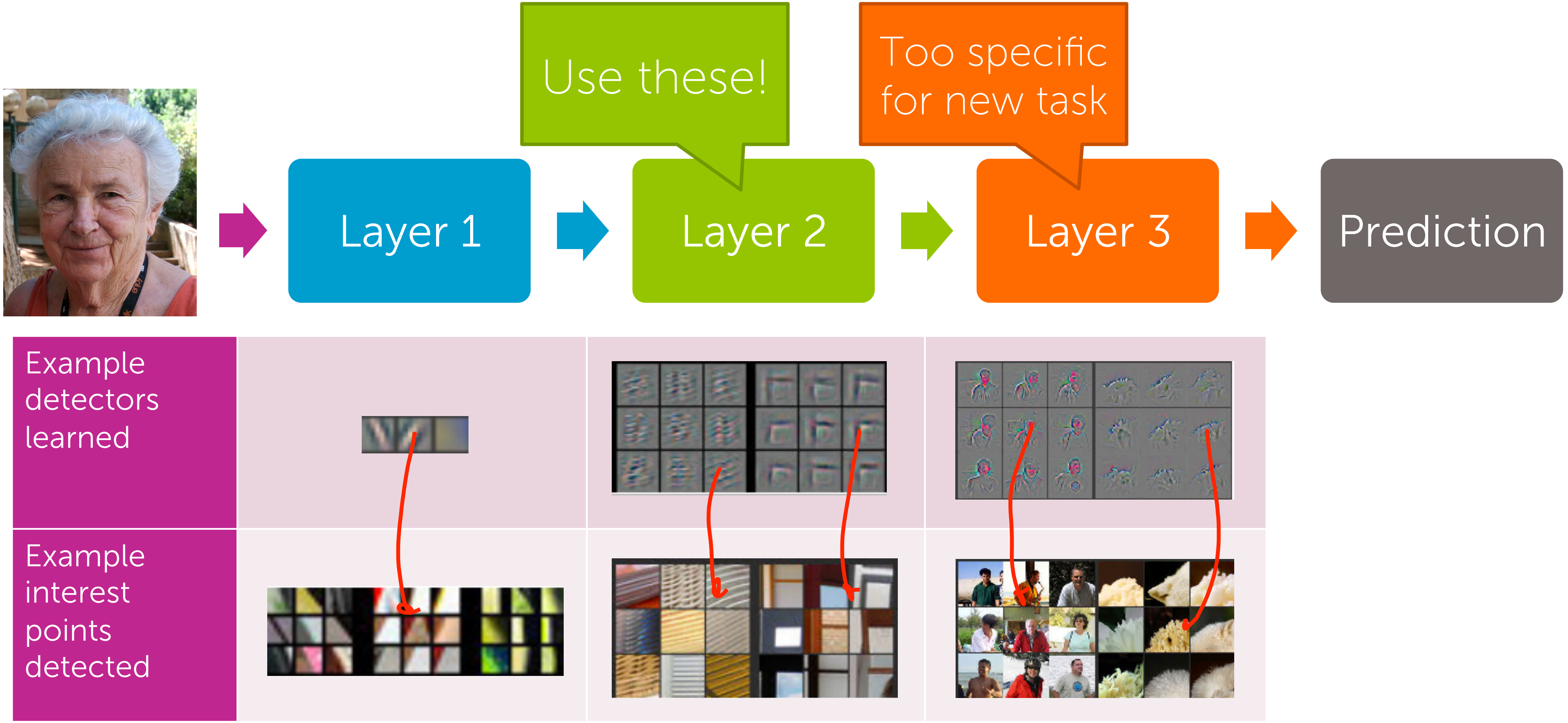



Use simple classifier
e.g., logistic regression,
SVMs, nearest neighbor,…

More generic
Can be used as feature extractor

~~Very specific to Task 1 Should be ignored for other tasks~~

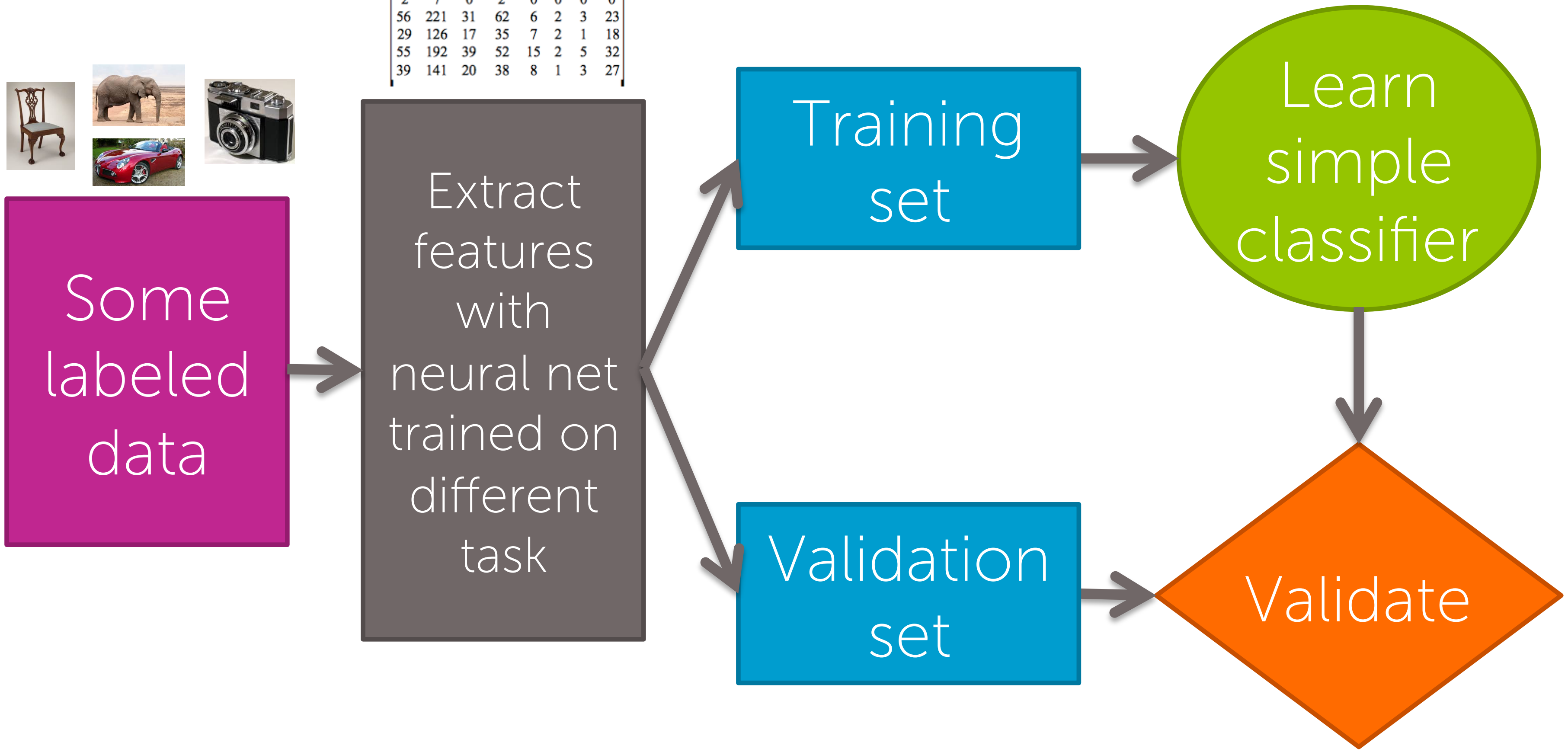
# Careful where you cut:
## *latter layers may be too task specific*




[Zeiler & Fergus '13]

# Transfer learning with deep features workflow

$$H=\begin{bmatrix} 20 & 77 & 21 & 31 & 4 & 5 & 1 & 13 \\ 35 & 141 & 26 & 46 & 9 & 0 & 4 & 28 \\ 76 & 301 & 49 & 91 & 23 & 2 & 7 & 43 \\ 70 & 280 & 53 & 84 & 21 & 5 & 10 & 54 \\ 71 & 258 & 44 & 76 & 19 & 4 & 4 & 32 \\ 21 & 86 & 12 & 25 & 5 & 0 & 4 & 22 \\ 92 & 350 & 53 & 102 & 19 & 1 & 5 & 40 \\ 2 & 7 & 0 & 2 & 0 & 0 & 0 & 0 \\ 56 & 221 & 31 & 62 & 6 & 2 & 3 & 23 \\ 29 & 126 & 17 & 35 & 7 & 2 & 1 & 18 \\ 55 & 192 & 39 & 52 & 15 & 2 & 5 & 32 \\ 39 & 141 & 20 & 38 & 8 & 1 & 3 & 27 \end{bmatrix}$$



# How general are deep features?

compology

# Summary of deep learning

# What you can do now…

- Describe multi-layer neural network models
- Interpret the role of features as local detectors in computer vision
- Relate neural networks to hand-crafted image features
- Describe some settings where deep learning achieves significant performance boosts
- State the pros & cons of deep learning model
- Apply the notion of transfer learning
- Use neural network models trained in one domain as features for building a model in another domain
- Build an image retrieval tool using deep features