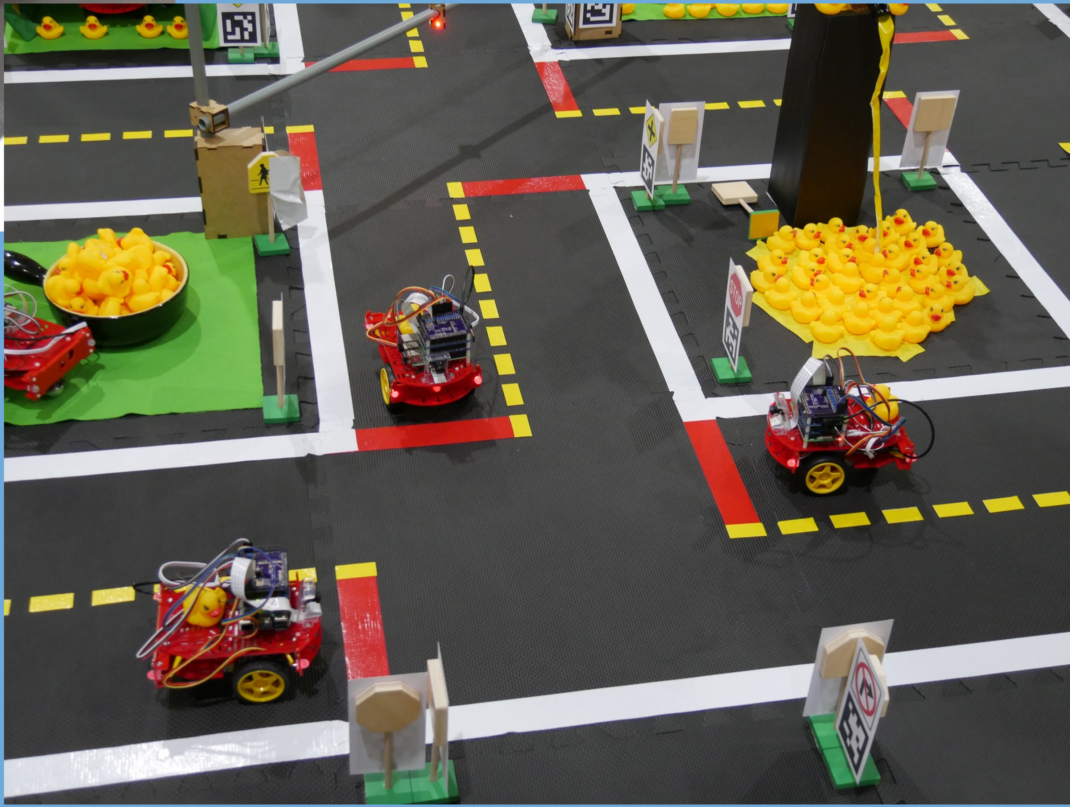
CS 3630!



Lecture 19:

Learning CNNs

Many slides adapted from Stanford's CS231N by Fei-Fei Li, Justin Johnson, Serena Yeung, as well as Slides by Marc'Aurelio Ranzato (NYU), Dhruv Batra & Devi Parikh (Georgia Tech)

Outline

1. Intra-class variability
2. Supervised Learning
3. Regression and Classification Losses
4. Stochastic Gradient Descent
5. Calculating Gradients

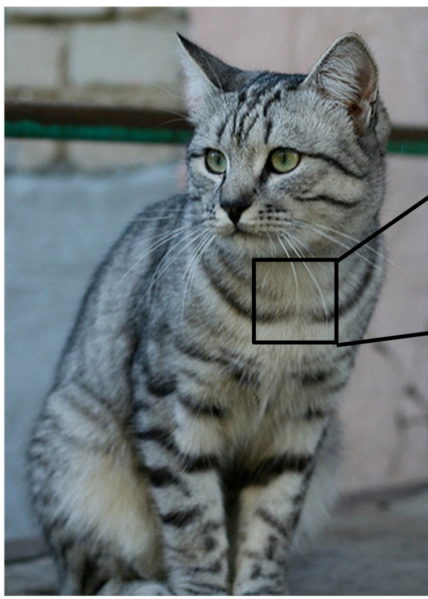
Image Classification: A core task in Computer Vision

This image by Nikita is licensed under CC-BY 2.0

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

$\longrightarrow$ cat

The Problem: Semantic Gap



This image by Nikita is

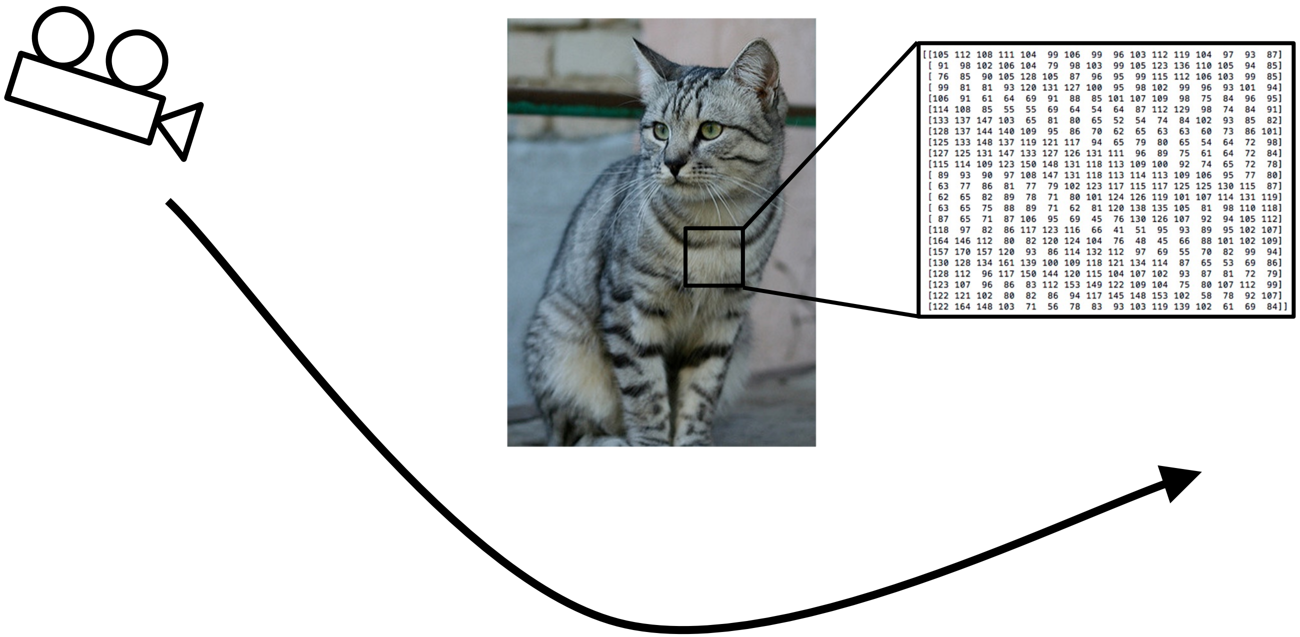

```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Challenges: Viewpoint variation



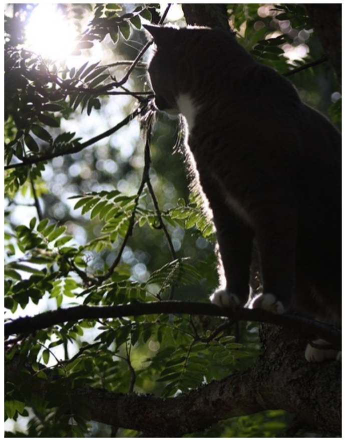
All pixels change when the camera moves!

Challenges: Illumination

This image is CC0 1.0 public domain

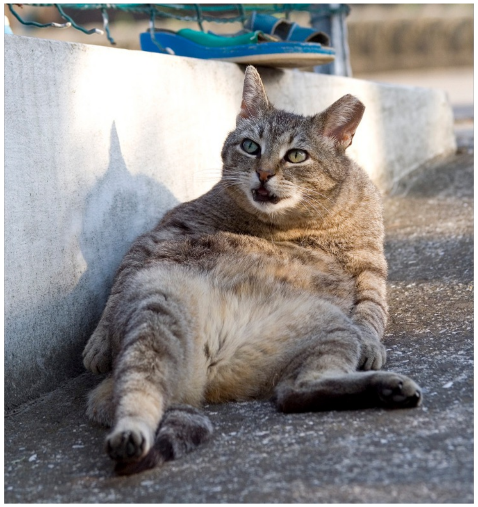
This image is CC0 1.0 public domain

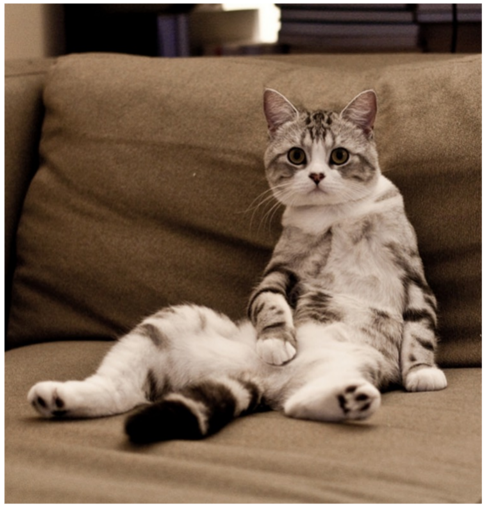
This image is CC0 1.0 public domain



This image is CC0 1.0 public domain

Challenges: Deformation





Challenges: Occlusion

Challenges: Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Challenges: Intraclass variation

Outline

1. Intra-class variability
2. Supervised Learning
3. Regression and Classification Losses
4. Stochastic Gradient Descent
5. Calculating Gradients

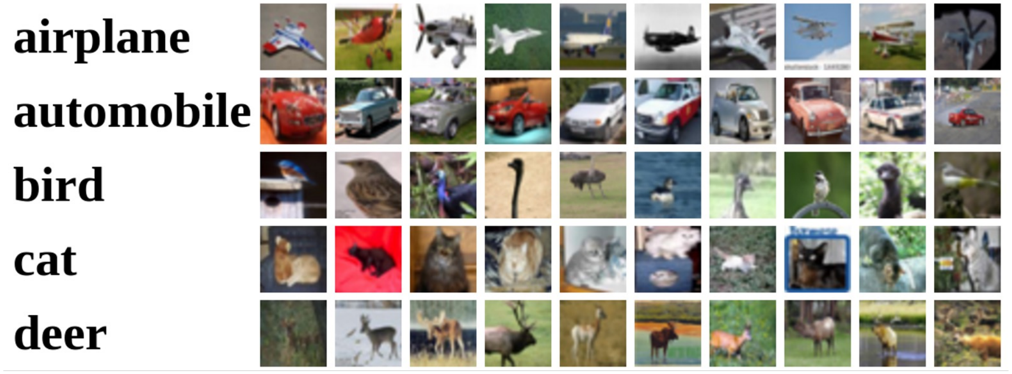
ML: A Data-Driven Approach

1. Collect a dataset of images x and labels y
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

```python
def train(images, labels):
  # Machine learning!
  return model
```

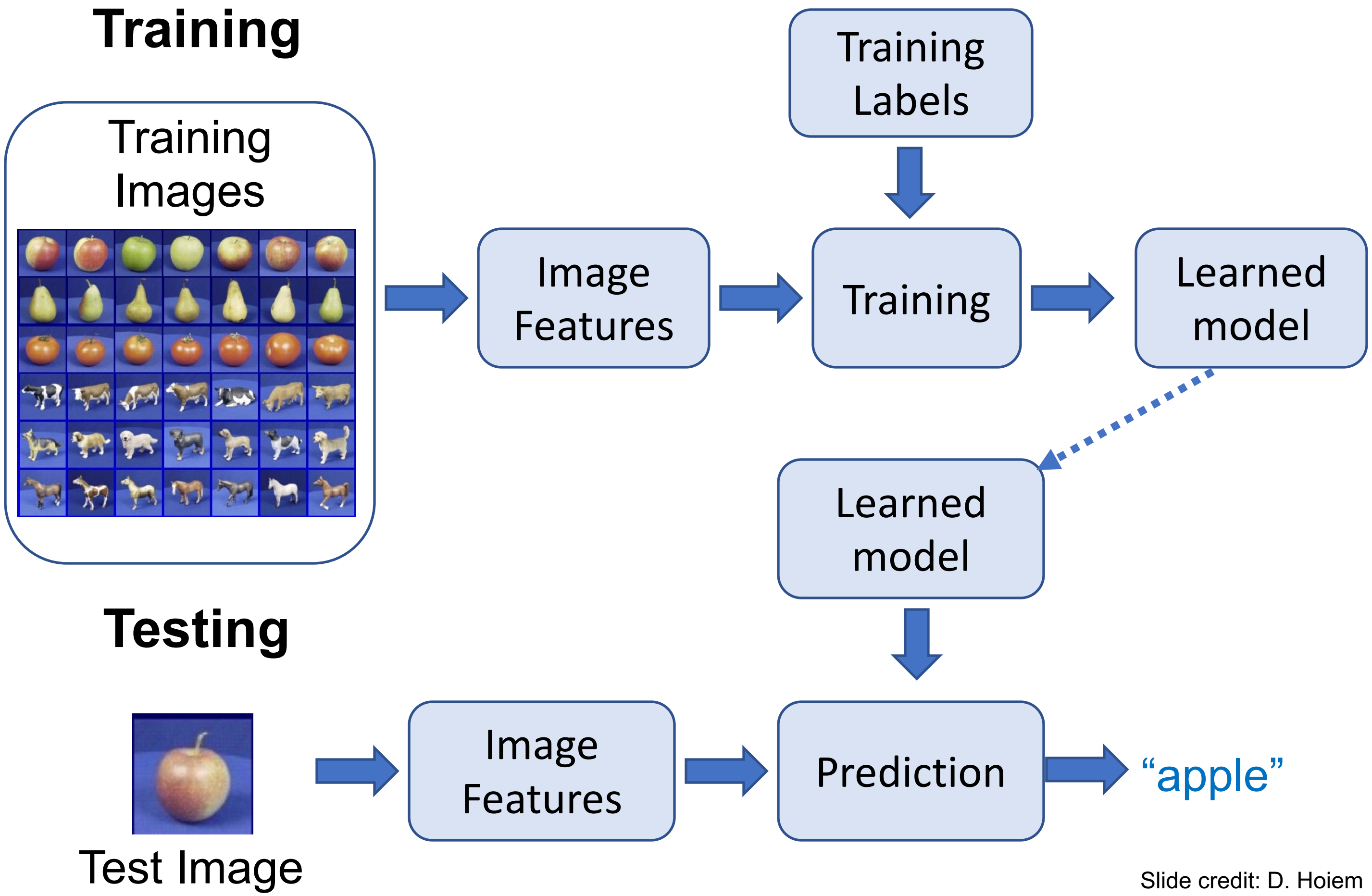
```python
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Example training set



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Steps



Slide credit: D. Hoiem

Outline

1. Intra-class variability
2. Supervised Learning
3. Regression and Classification Losses
4. Stochastic Gradient Descent
5. Calculating Gradients

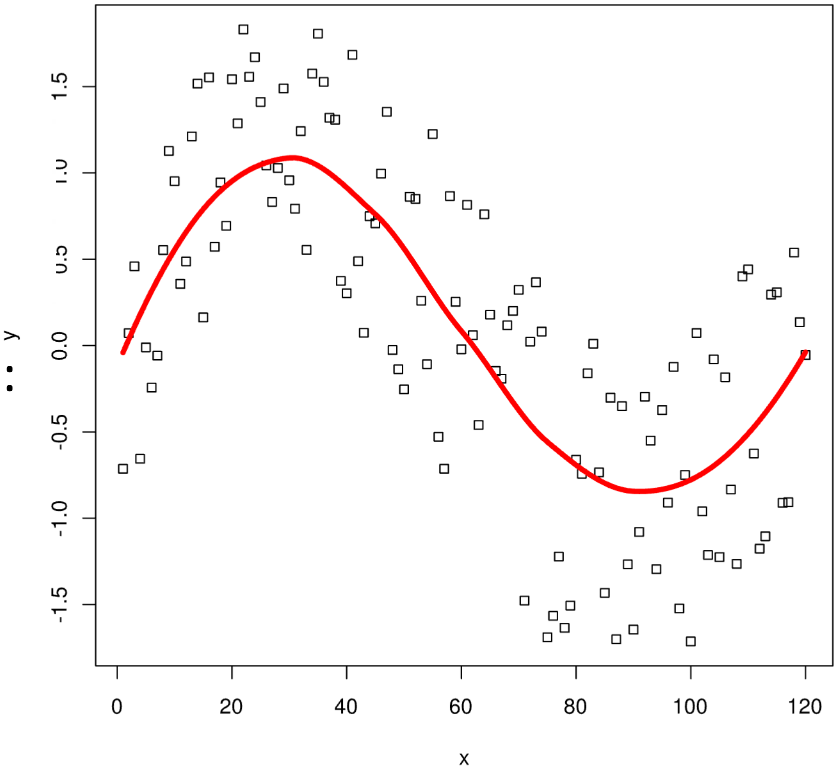
Two different learning problems are classification and regression

- **Regression**: continuous labels y
- Sum-square-difference loss:

$$L_{\mathrm{SSD}}(W;D) \doteq \sum_{(x,y)\in D} |f(x;W) - y|^2$$

- Example: direct 3D pose regression:

https://paperswithcode.com/task/6d-pose-estimation



By Kierano - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=7034448

Two different learning problems are classification and regression

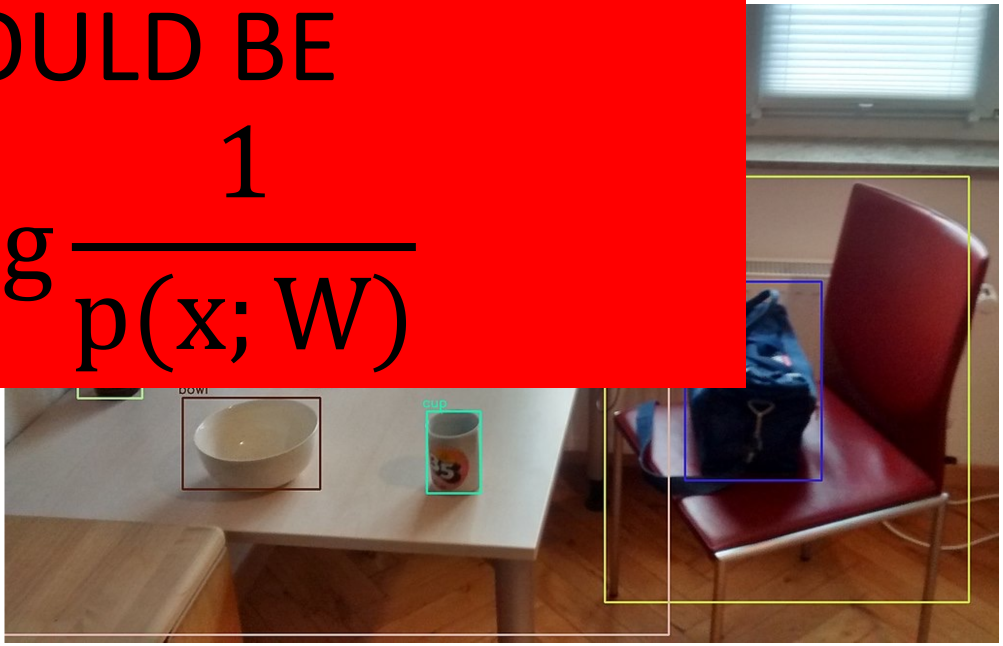
- **Classification**: discrete labels y
- Cross-entropy loss:

$L_{\mathrm{CE}}(W$

FIX THIS LOSS FUNCTION --- SHOULD BE

$$\sum\sum\log\frac{1}{p(x;W)}$$

- Aver
- Exar



(MTheiler), CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

How to get probabilities? **Softmax** converts a set of C class "scores" into "probabilities"

- If we need to classify inputs into C different classes, we put C units in the last layer to produce C *one-vs.-others* scores $f_1, f_2, \ldots, f_C$
- Apply *softmax* function to convert these scores to probabilities:

$$\mathrm{softmax}(f_1, \ldots, f_c) = \left(\frac{\exp(f_1)}{\sum_j \exp(f_j)}, \ldots, \frac{\exp(f_C)}{\sum_j \exp(f_j)}\right)$$

If one of the inputs is much larger than the others, then the corresponding softmax value will be close to 1 and others will be close to 0

Outline

1. Intra-class variability
2. Supervised Learning
3. Regression and Classification Losses
4. Stochastic Gradient Descent
5. Calculating Gradients

How to minimize the loss by changing the weights?
Strategy: **Follow the slope of the loss function**

Strategy: **Follow the slope**

In 1-dimension, the derivative of a function:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In multiple dimensions, the **gradient** is the vector of (partial derivatives) along each dimension

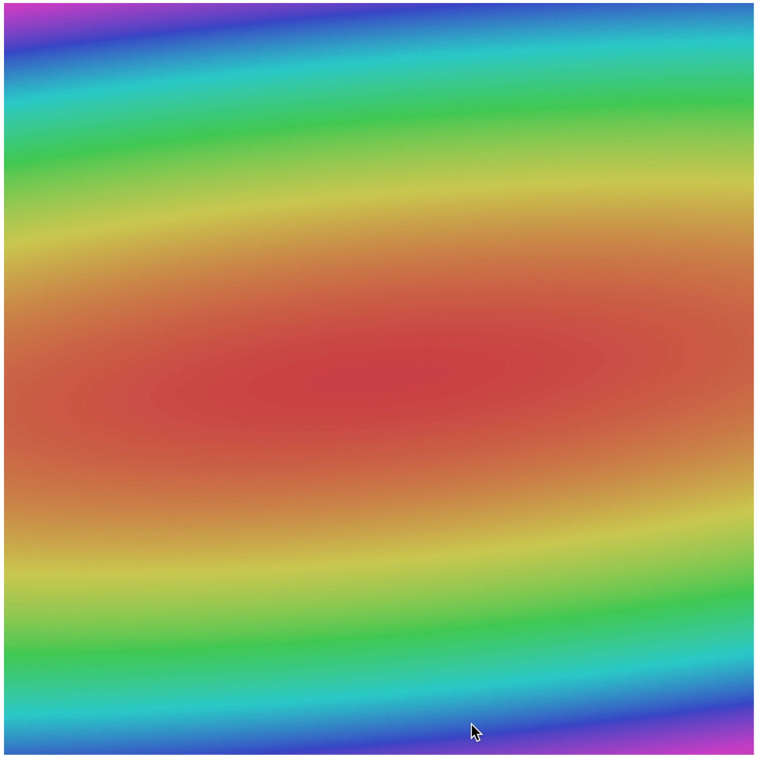
The slope in any direction is the **dot product** of the direction with the gradient

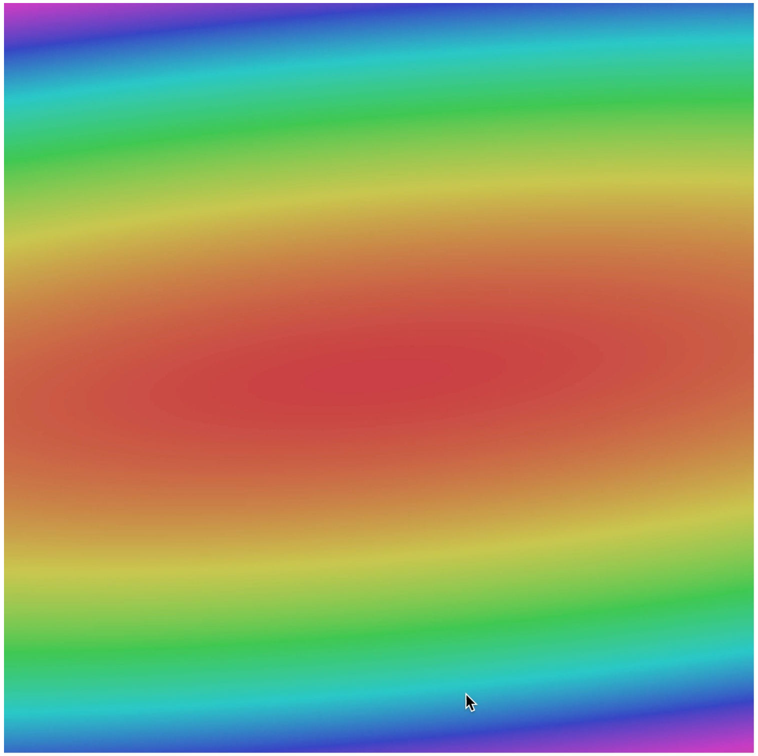
The direction of steepest descent is the **negative gradient**

Gradient Descent

```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```





Stochastic Gradient Descent (SGD)

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$$

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

Full sum expensive when N is large!

Approximate sum using a **minibatch** of examples
32 / 64 / 128 common

```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Outline

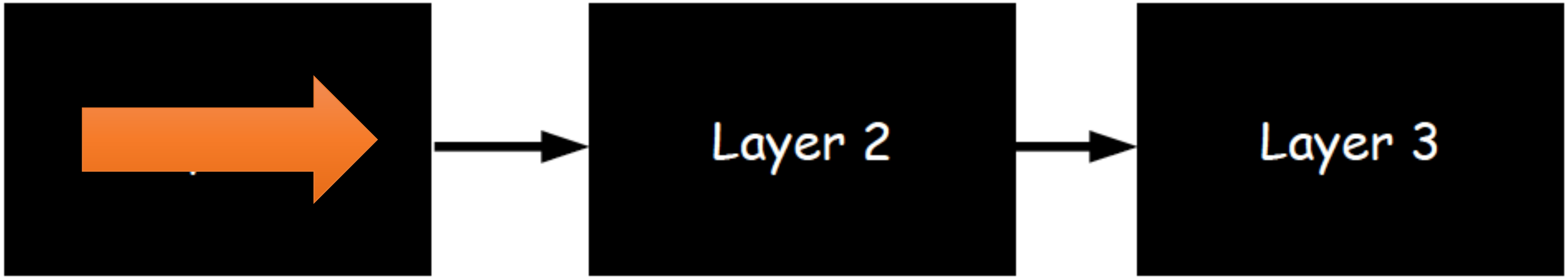
1. Intra-class variability
2. Supervised Learning
3. Regression and Classification Losses
4. Stochastic Gradient Descent
5. Calculating Gradients

How do we *really* compute gradients?

- Analytic or “Manual” Differentiation
- Symbolic Differentiation
- Numerical Differentiation
- Automatic Differentiation!
    - Forward mode AD
    - Reverse mode AD
        - aka “backpropagation”
    - Implemented in specialized frameworks:
        - pytorch (Facebook)
        - TensorFlow (Google) frameworks
    - Main computation, mainly done on GPU (or TPU)

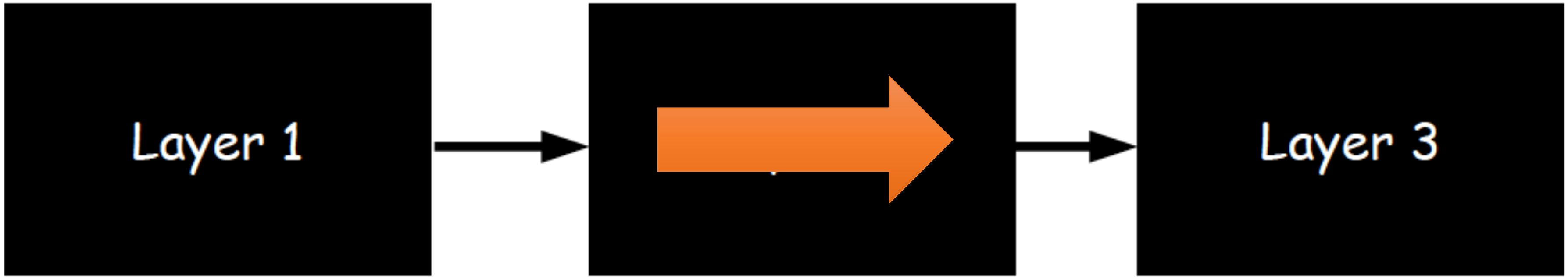
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]



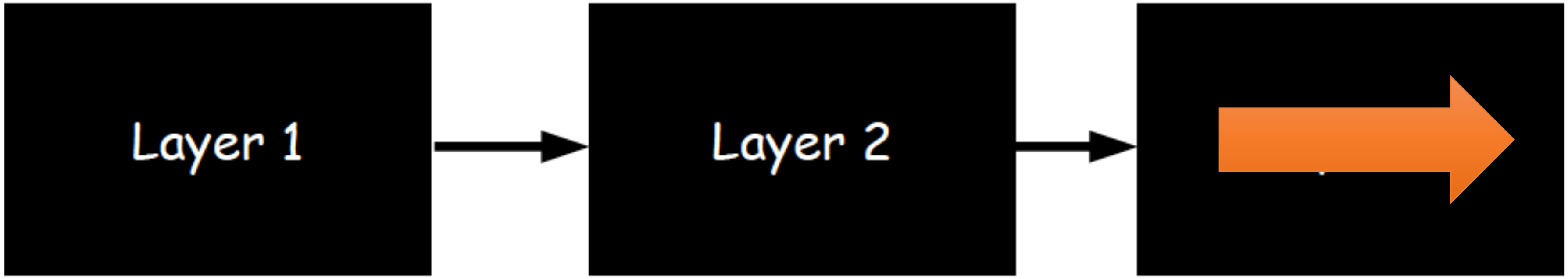
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]



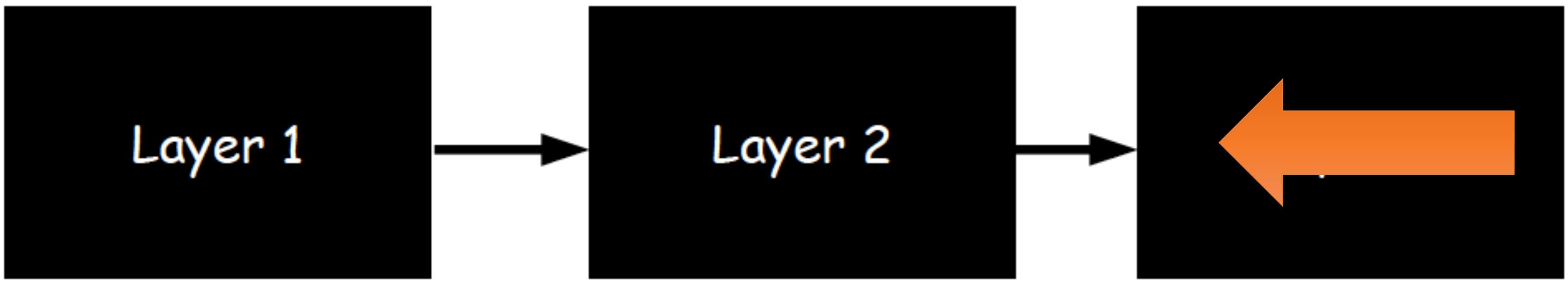
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]



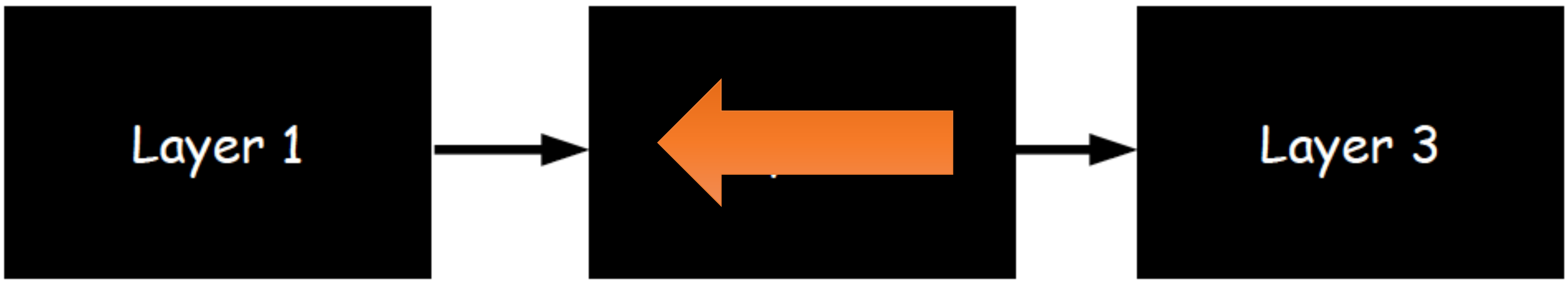
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]



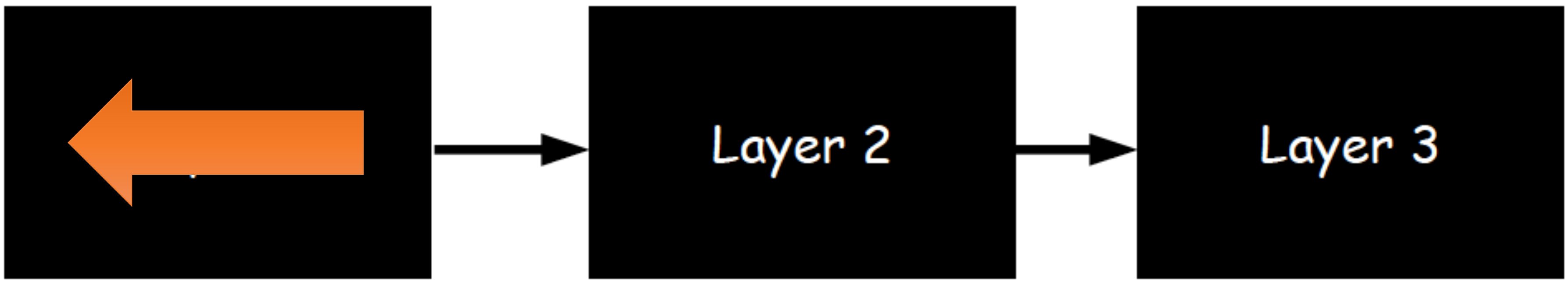
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]



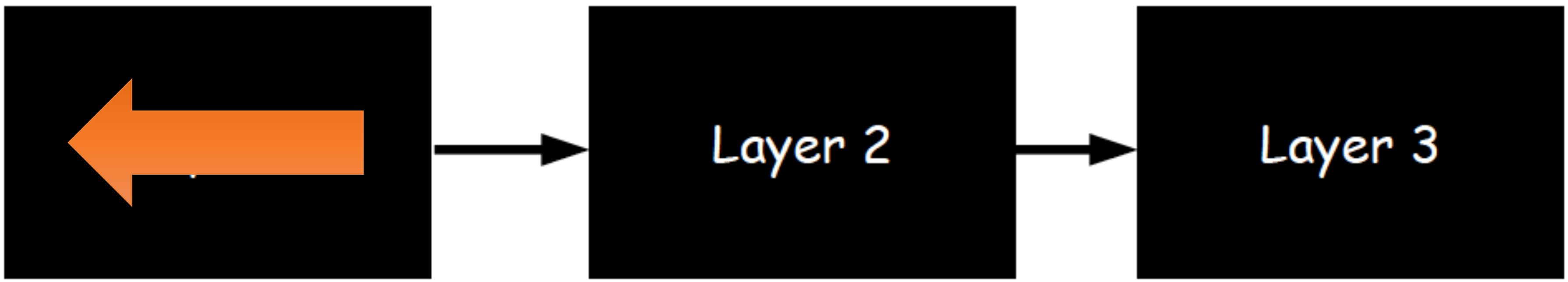
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]



Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]
- Step 3: Use gradient to update parameters



$$\theta \leftarrow \theta - \eta \frac{dL}{d\theta}$$

Outline

1. Intra-class variability: viewpoint, lighting, instance
2. Supervised Learning: label + optimization
3. Regression and Classification: SSD + CE
4. Stochastic Gradient Descent: mini-batches
5. Calculating Gradients: back-propagation