# Machine Learning Course at MIPT

## Deep Introduction to Deep Learning

**Valentin Malykh**

ml-mipt.github.io, val.maly.hk

MIPT
HIGHER SCHOOL OF SYSTEMS ENGINEERING

iPavlov.ai

April 10th, 2018

Credit A. Karpathy

# Deep Neural Nets

## 2006




Computer vision = 60%

$0.6^{12} = 0.00217$

Credit Prof. V. Lempitsky

# Deep Neural Nets

**2014**

Completed • Swag • 215 teams

**Dogs vs. Cats**

Wed 25 Sep 2013 – Sat 1 Feb 2014 (8 months ago)

Dashboard ▼ Private Leaderboard - Dogs vs. Cats

This competition has completed. This leaderboard reflects the final standings. See someone

| # | Δ1w | Team Name * in the money | Score | Entries | Last Submission UTC (Best – Las |
|---|---|---|---|---|---|
| 1 | — | Pierre Sermanet * | 0.98914 | 5 | Sat, 01 Feb 2014 21:43:19 (- |
| 2 | ↑26 | orchid * | 0.98309 | 17 | Sat, 01 Feb 2014 23:52:30 |
| 3 | — | Owen | 0.98171 | 15 | Sat, 01 Feb 2014 17:04:40 (- |
| 4 | new | Paul Covington | 0.98171 | 3 | Sat, 01 Feb 2014 23:05:20 |
| 5 | ↓3 | Maxim Milakov | 0.98137 | 24 | Sat, 01 Feb 2014 18:20:58 |

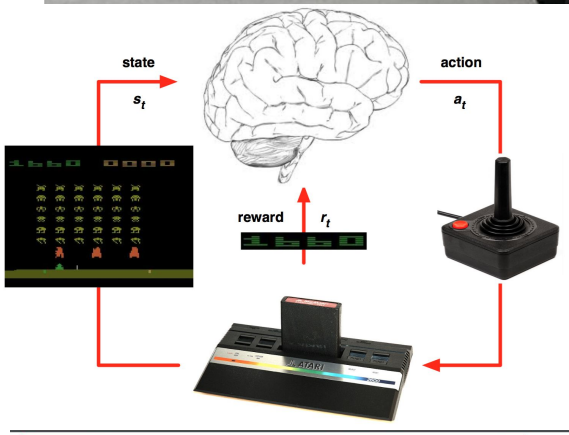
$$0.989^{12} = 0.875$$

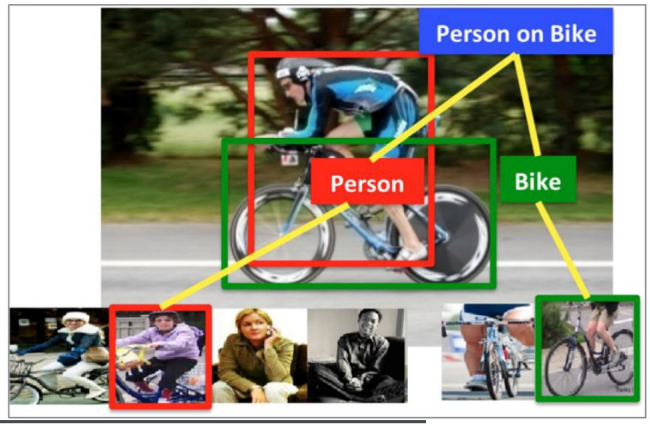
After 8 years of operation, Asirra is shutting down effective October 1, 2014. Thank you to all of our users!

Credit Prof. V. Lempitsky

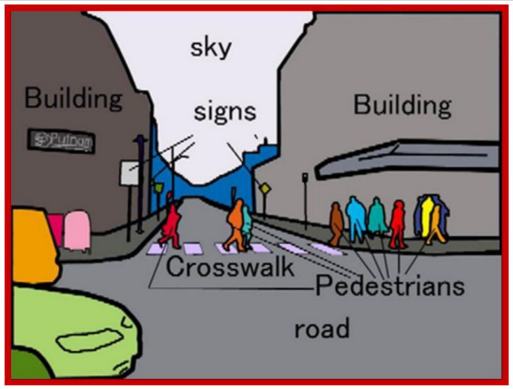
# Real world problems

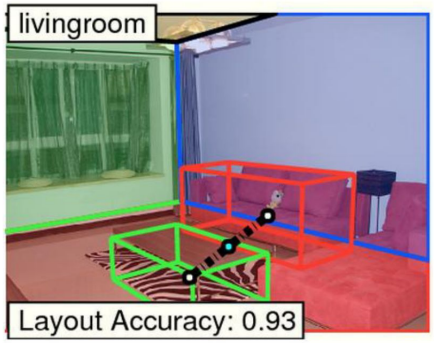
- Object detection
- Action classification
- Image captioning
- …



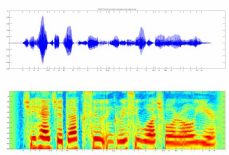




"man in black shirt is playing guitar."

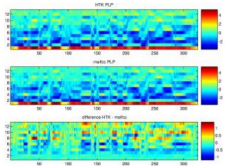



Audio Features



Spectrogram



MFCC

# Image Classification

**Image Classification**: a core task in Computer Vision

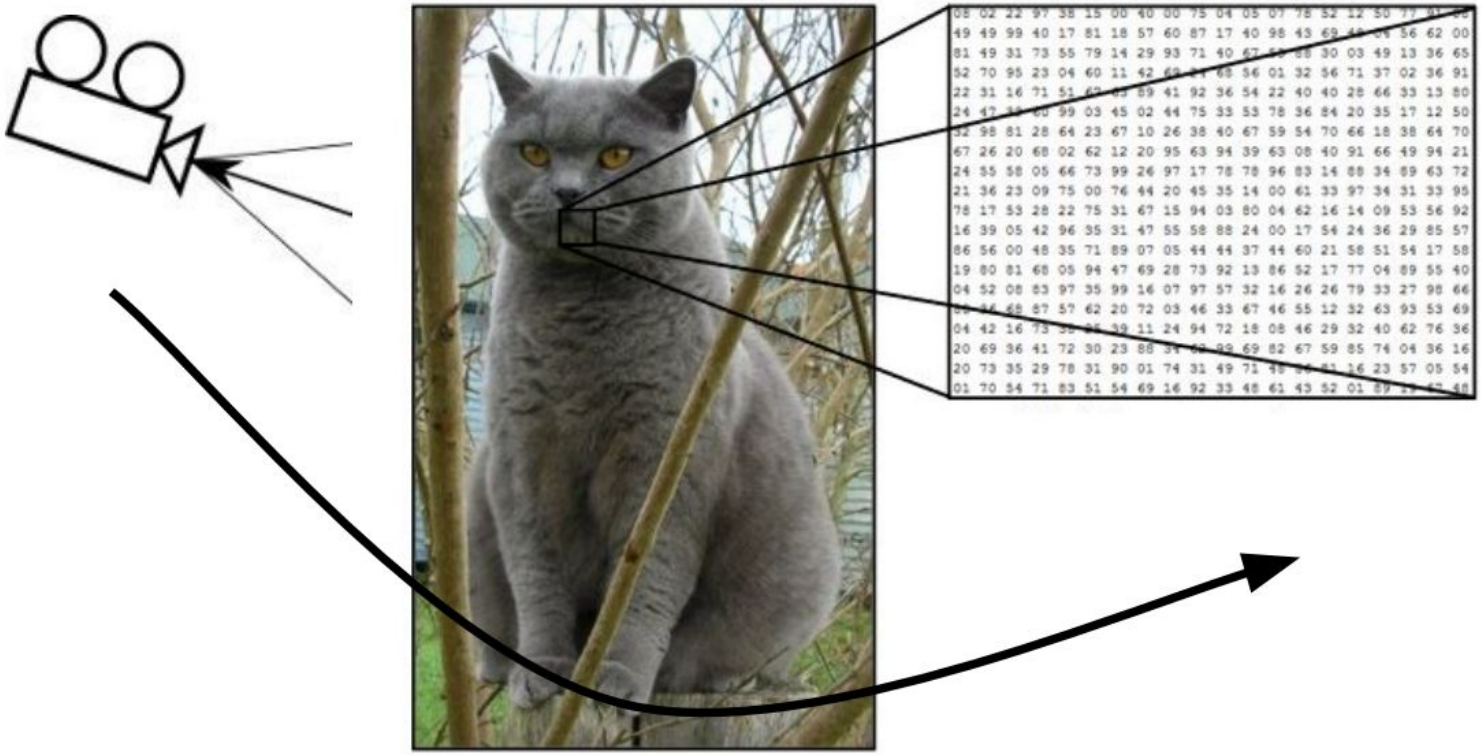
(assume given set of discrete labels)
{dog, cat, truck, plane, ...}



cat

# Image Classification

## Challenges: Viewpoint Variation



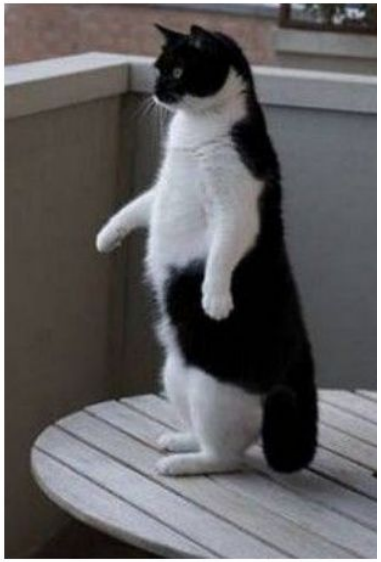
# Image Classification

## Challenges: Illumination

# Image Classification

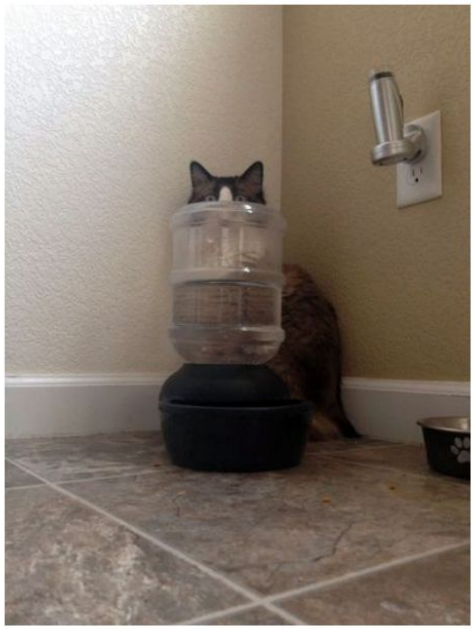
## Challenges: Deformation



# Image Classification

## Challenges: Occlusion



# Image Classification

Challenges: Background clutter

# Image Classification

## Challenges: Intraclass variation

# Image classifier

```
def train(train_images, train_labels):
  # build a model for images -> labels...
  return model

def predict(model, test_images):
  # predict test_labels using the model...
  return test_labels
```


Q: How to make an image classifier :)?

# Parametric Approach

image parameters

f(**x**,**W**)

**10** numbers, indicating class scores

**[32x32x3]**
array of numbers 0...1
(3072 numbers total)

# Linear Classifier

$$f(x, W) = Wx$$

**[32x32x3]**
array of numbers 0...1

**10** numbers, indicating class scores

Q: How many parameters are here ?

# Linear Classifier

Parametric approach: **Linear classifier**

$$f(x, W) = Wx$$

3072x1

10x1

10x3072

**10** numbers, indicating class scores

**[32x32x3]**
array of numbers 0...1

parameters, or “weights”

# Linear Classifier

Parametric approach: **Linear classifier**

$$f(x, W) = Wx$$

3072x1

(+b) 10x1

10x1

10x3072

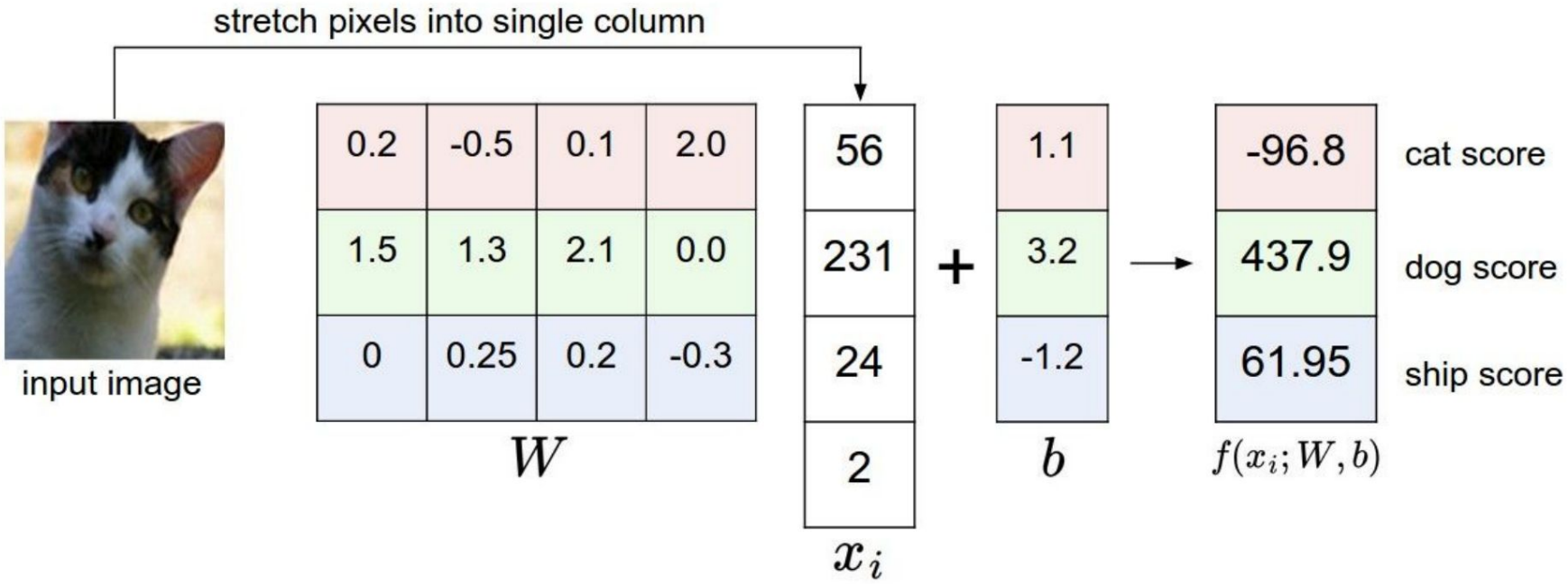
**10** numbers, indicating class scores

**[32x32x3]**
array of numbers 0...1

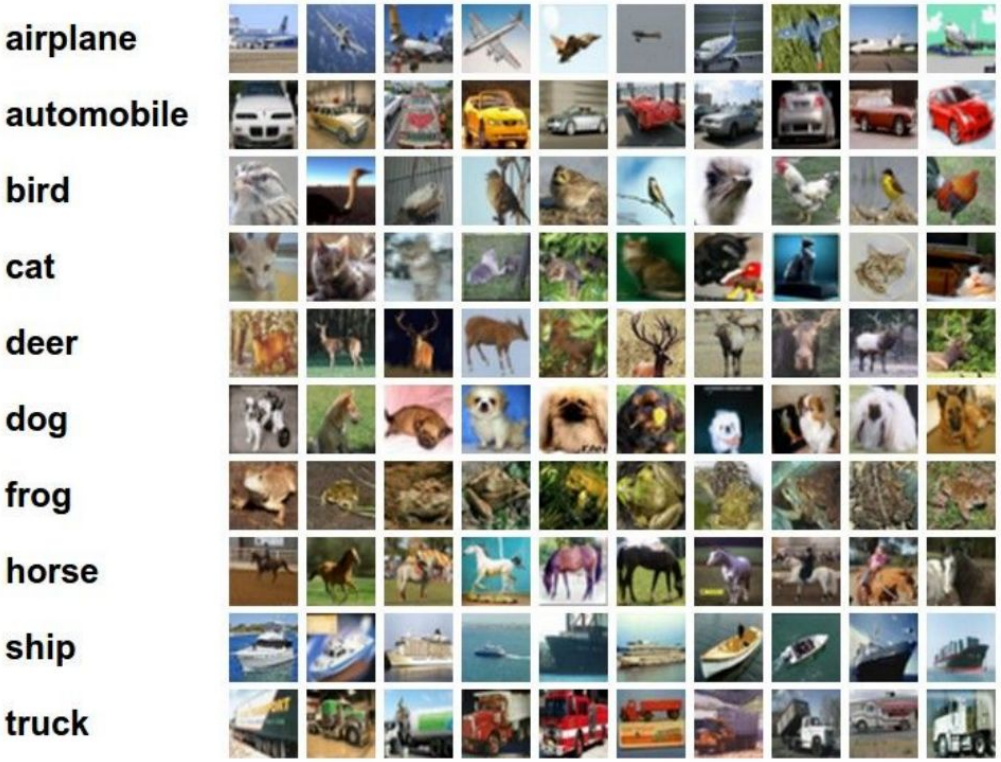
parameters, or “weights”

# Linear Classifier

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



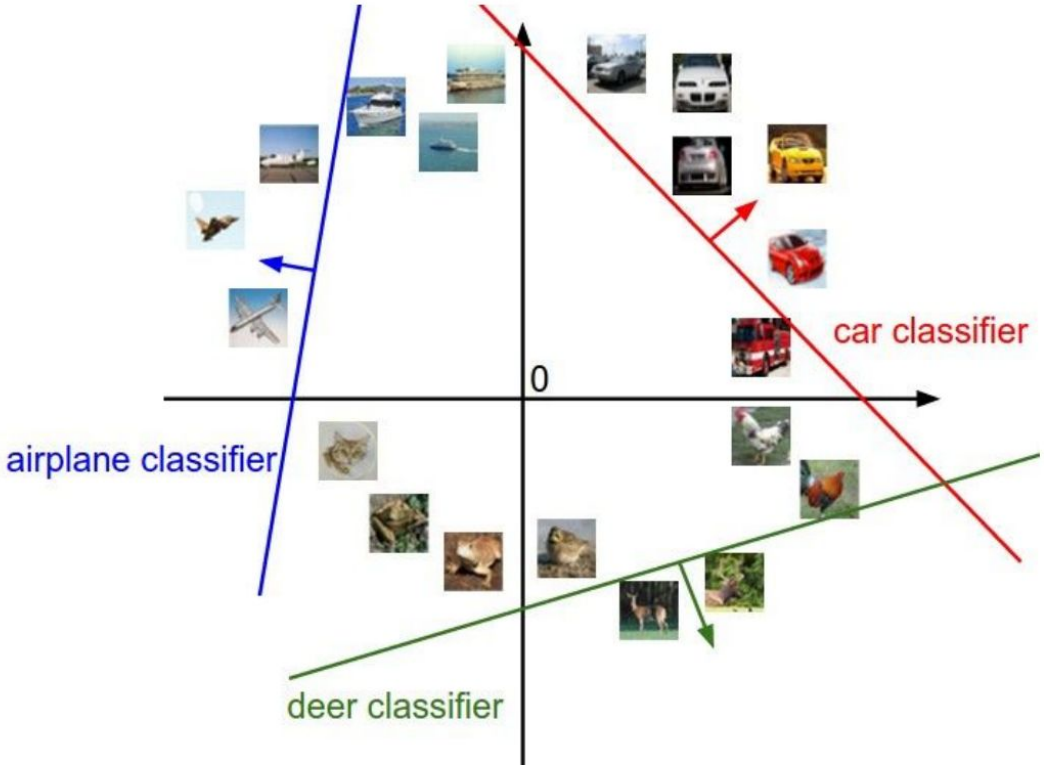
# Interpreting a Linear Classifier




$$f(x_i, W, b) = Wx_i + b$$

Q: what does the linear classifier do, in Russian?

# Interpreting a Linear Classifier

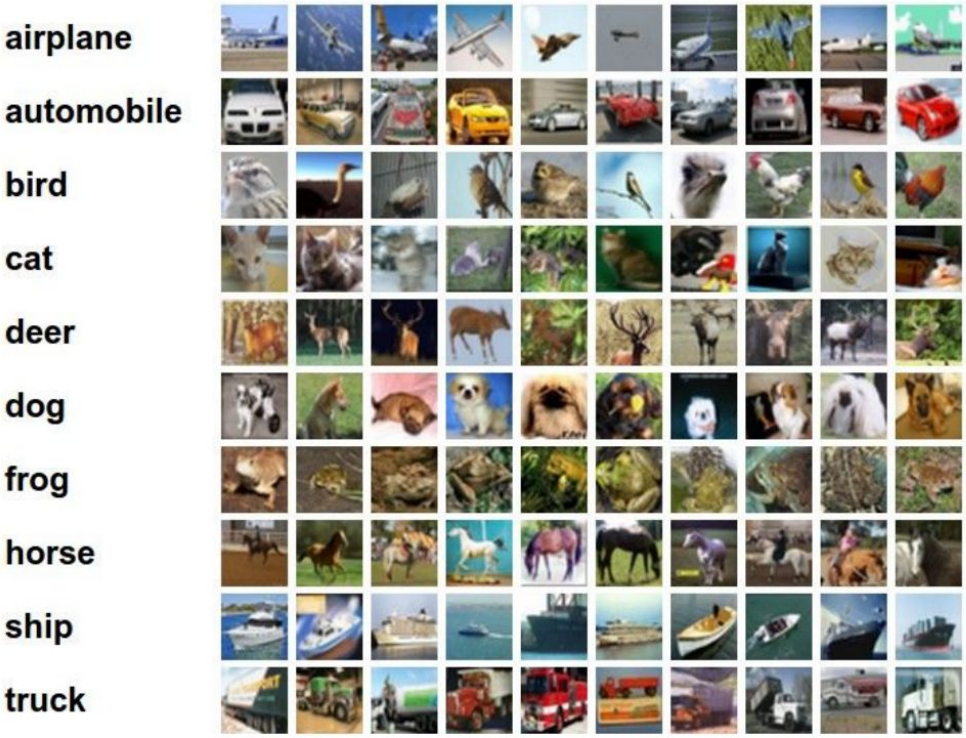



$$f(x_i, W, b) = Wx_i + b$$

**[32x32x3]**
array of numbers 0...1
(3072 numbers total)

- In linear models we get only one weight vector per class

# Interpreting a Linear Classifier




$$f(x_i, W, b) = Wx_i + b$$

Q2: what would be a very hard set of classes for a linear classifier to distinguish?

# TODO



| | | | |
|---|---|---|---|
| airplane | -3.45 | -0.51 | 3.42 |
| automobile | -8.87 | **6.04** | 4.64 |
| bird | 0.09 | 5.31 | 2.65 |
| cat | **2.9** | -4.22 | 5.1 |
| deer | 4.48 | -4.19 | 2.64 |
| dog | 8.02 | 3.58 | 5.55 |
| frog | 3.78 | 4.49 | **-4.34** |
| horse | 1.06 | -4.37 | -1.5 |
| ship | -0.36 | -2.09 | -4.79 |
| truck | -0.72 | -2.93 | 6.14 |

## TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.

2. Come up with a way of efficiently finding the parameters that minimize the loss function. **(optimization)**

# Loss Function: SVM

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

| | | | |
|---|---|---|---|
| cat | **3.2** | 1.3 | 2.2 |
| car | 5.1 | **4.9** | 2.5 |
| frog | -1.7 | 2.0 | **-3.1** |

**Multiclass SVM loss:**

Given an example $(x_i, y_i)$
where $x_i$ is the image and
where $y_i$ is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q: what if the sum was instead over all classes? (including j = y_i)

Q: what if we used a mean instead of a sum here?

# Loss Function: Softmax

**Softmax Classifier** (Multinomial Logistic Regression)

cat **3.2**

car 5.1

frog -1.7

**scores = unnormalized log probabilities of the classes.**

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ where $$s = f(x_i; W)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y = y_i|X = x_i)$$

---

in summary: $$L_i = -\log(\frac{e^{sy_i}}{\sum_j e^{s_j}})$$

# TODO



| | | | |
|---|---|---|---|
| airplane | -3.45 | -0.51 | 3.42 |
| automobile | -8.87 | **6.04** | 4.64 |
| bird | 0.09 | 5.31 | 2.65 |
| cat | **2.9** | -4.22 | 5.1 |
| deer | 4.48 | -4.19 | 2.64 |
| dog | 8.02 | 3.58 | 5.55 |
| frog | 3.78 | 4.49 | **-4.34** |
| horse | 1.06 | -4.37 | -1.5 |
| ship | -0.36 | -2.09 | -4.79 |
| truck | -0.72 | -2.93 | 6.14 |

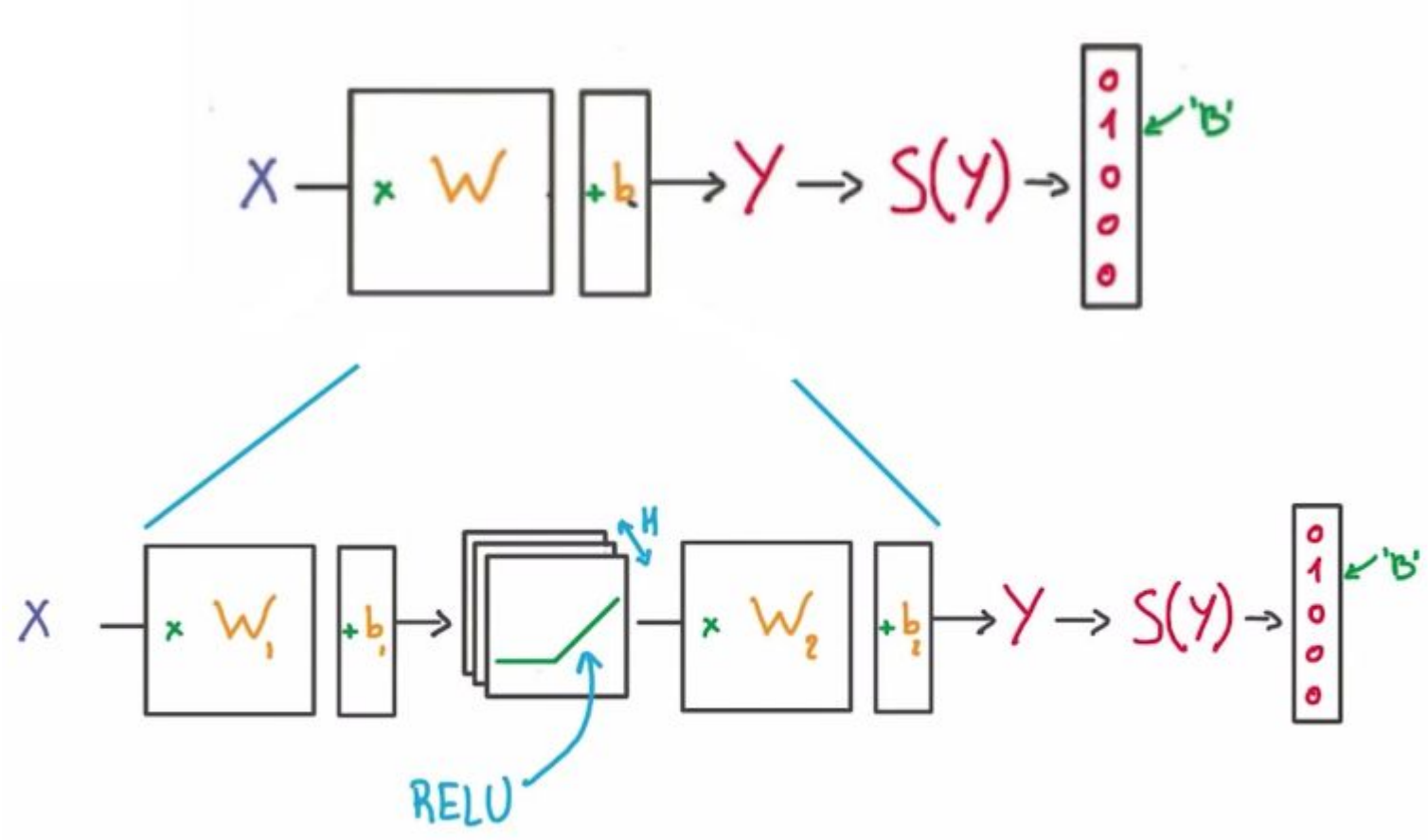
TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.

+

2. Come up with a way of efficiently finding the parameters that minimize the loss function. **(optimization)**

# Linear Classifier




Q: What does DNN do in Russian?

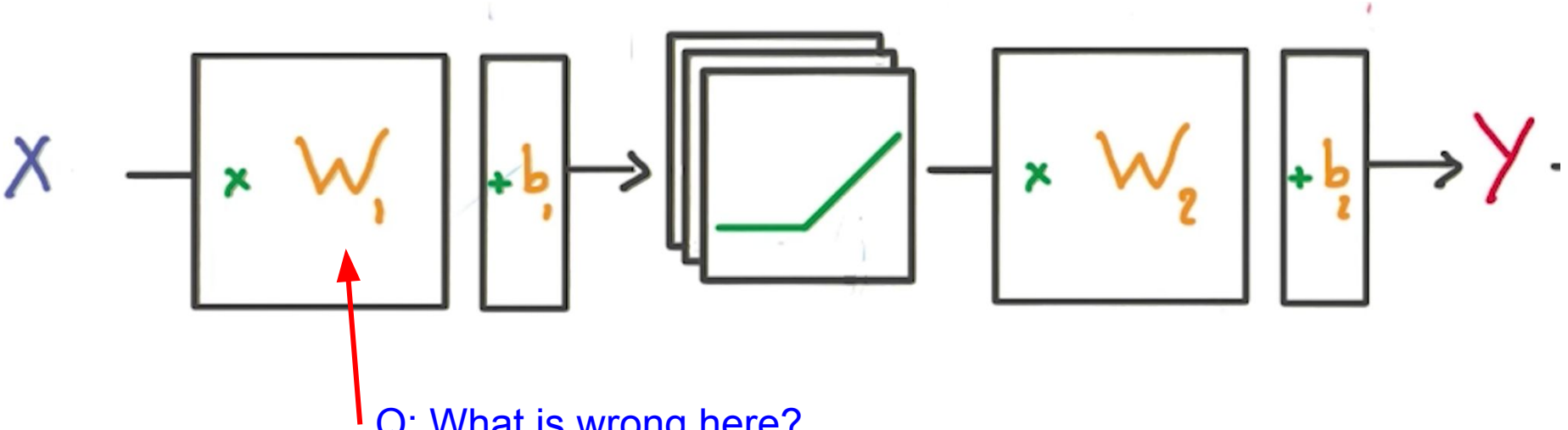
# Linear Classifier

- In linear models we get only one weight vector per class


- Now we got a lot of feature extractor in matrix $W_1$ we call it Fist Layer

linear vs non-linear vs deep nonlinear

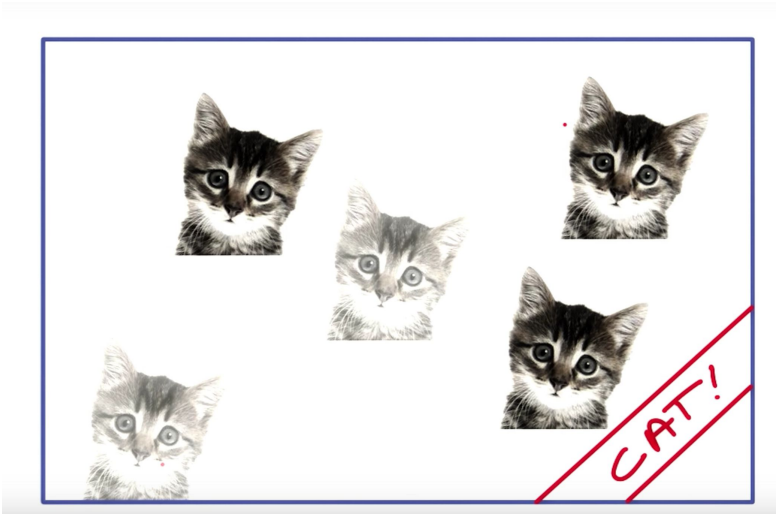
# Statistical Invariance

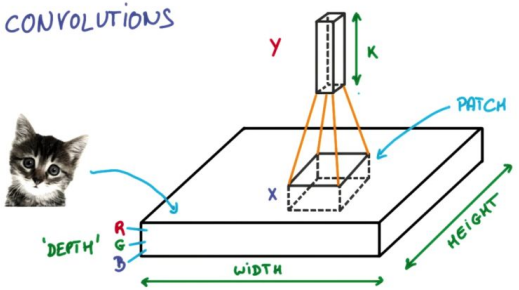



Q: What is wrong here?

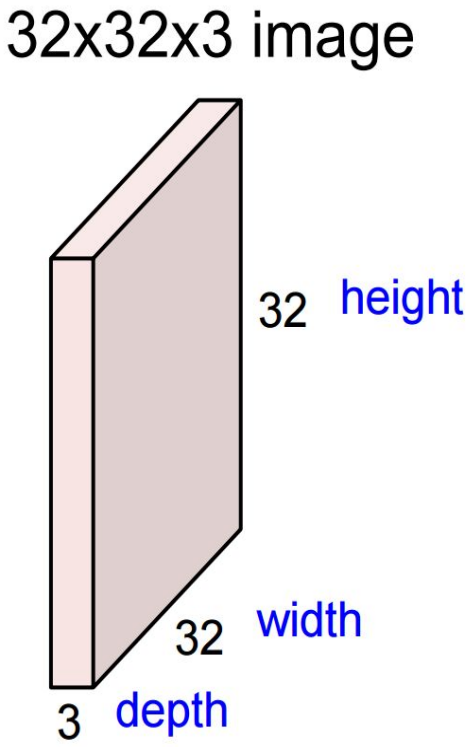
**Overfitting**

**200x200x3 x 1000**



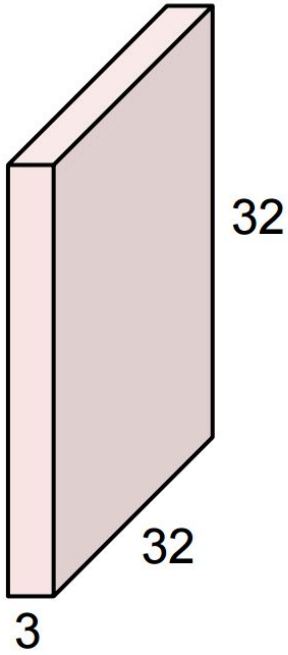



# Convolution Layer

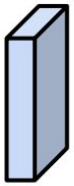


# Convolution Layer

32x32x3 image




5x5x3 filter



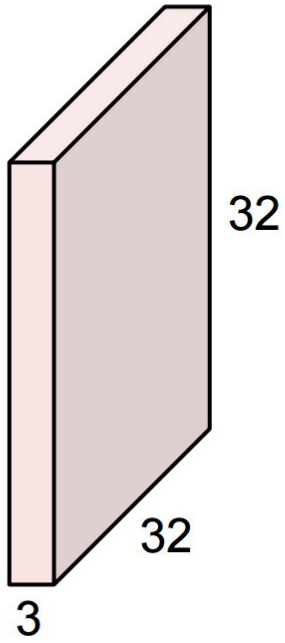
**Convolve** the filter with the image
i.e. “slide over the image spatially,
computing dot products”

# Convolution

## Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

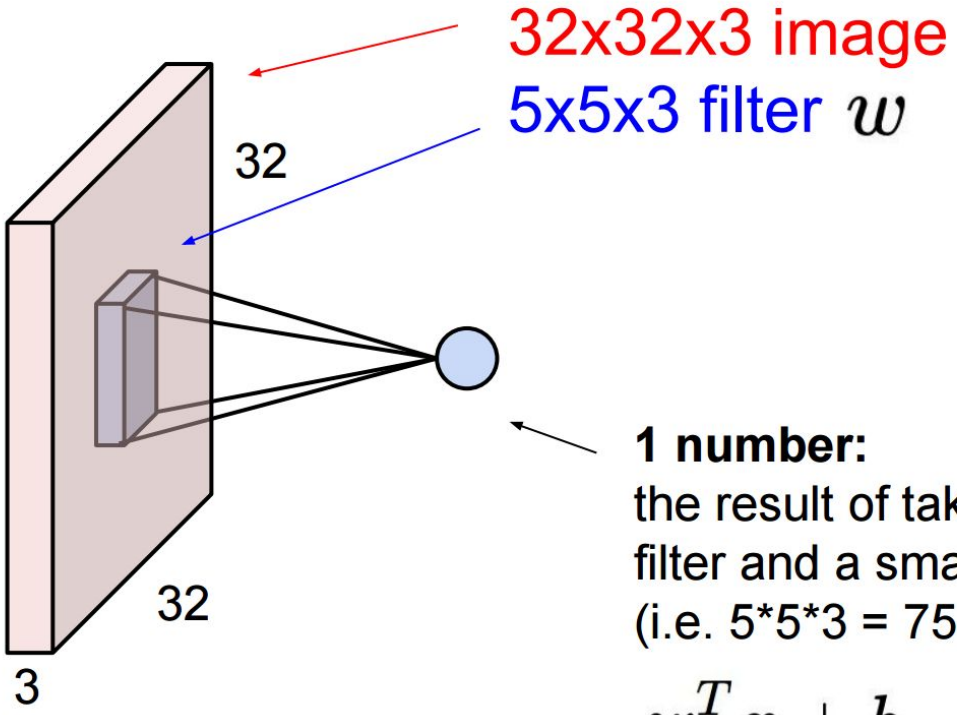



5x5x3 filter

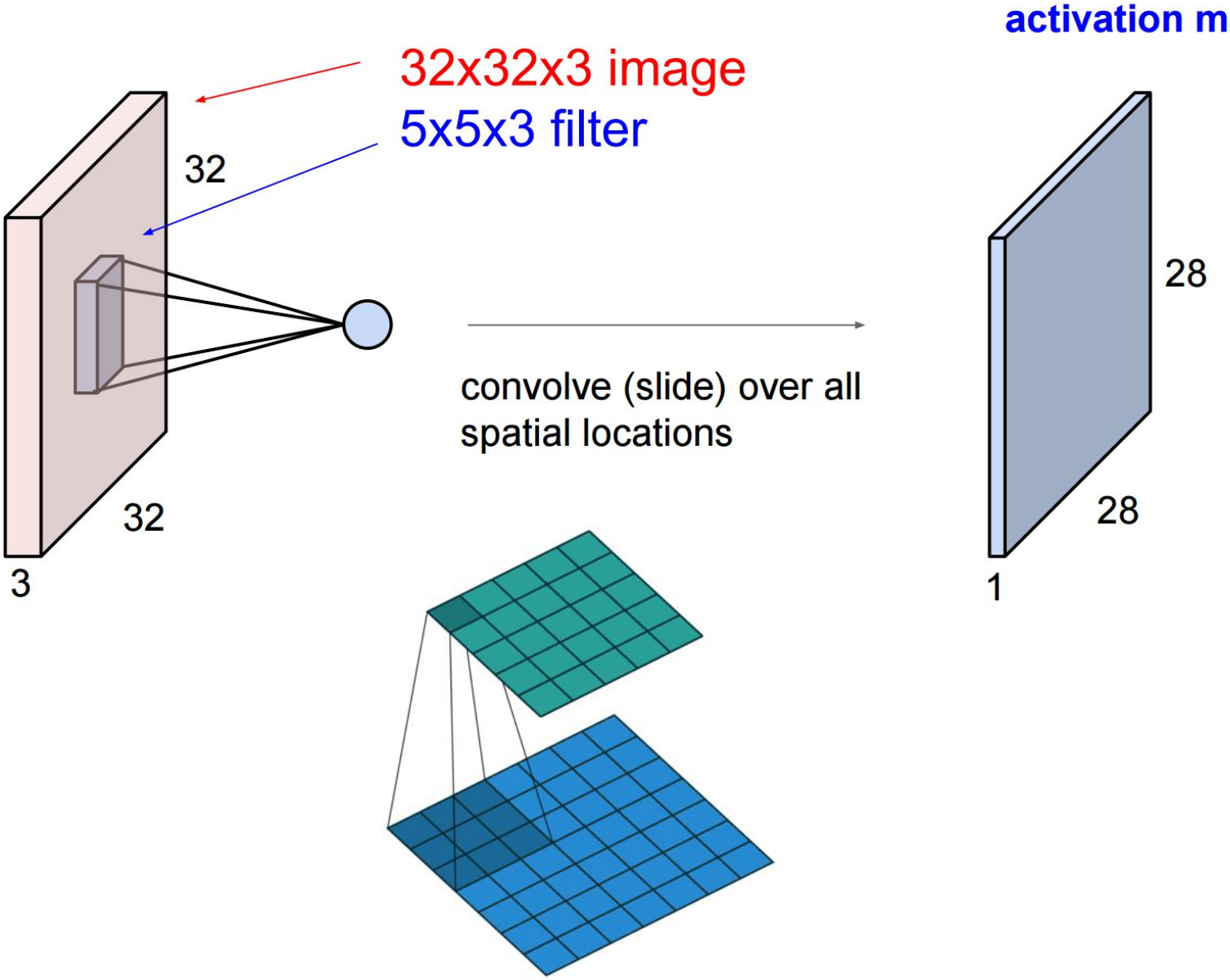
**Convolve** the filter with the image
i.e. “slide over the image spatially, computing dot products”

# Convolution

## Convolution Layer



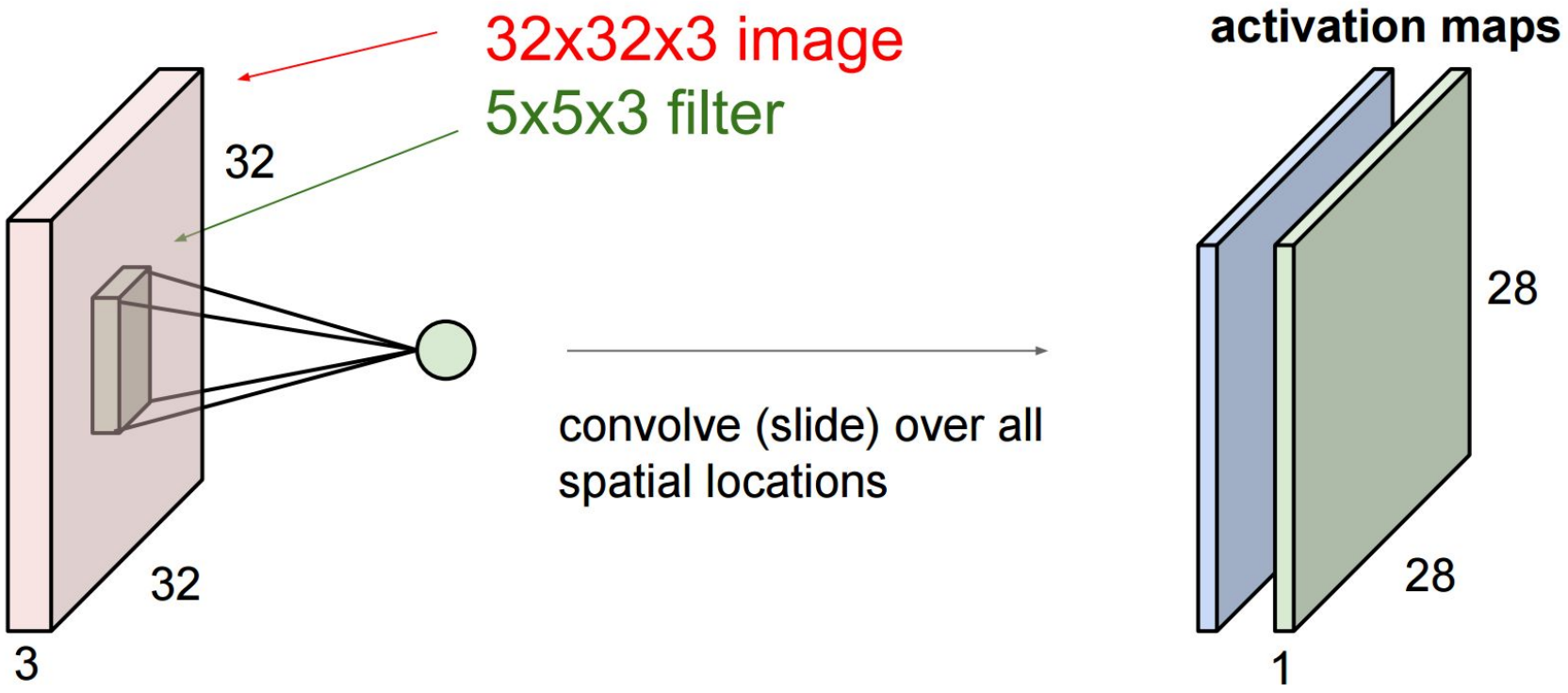
# Convolution Layer



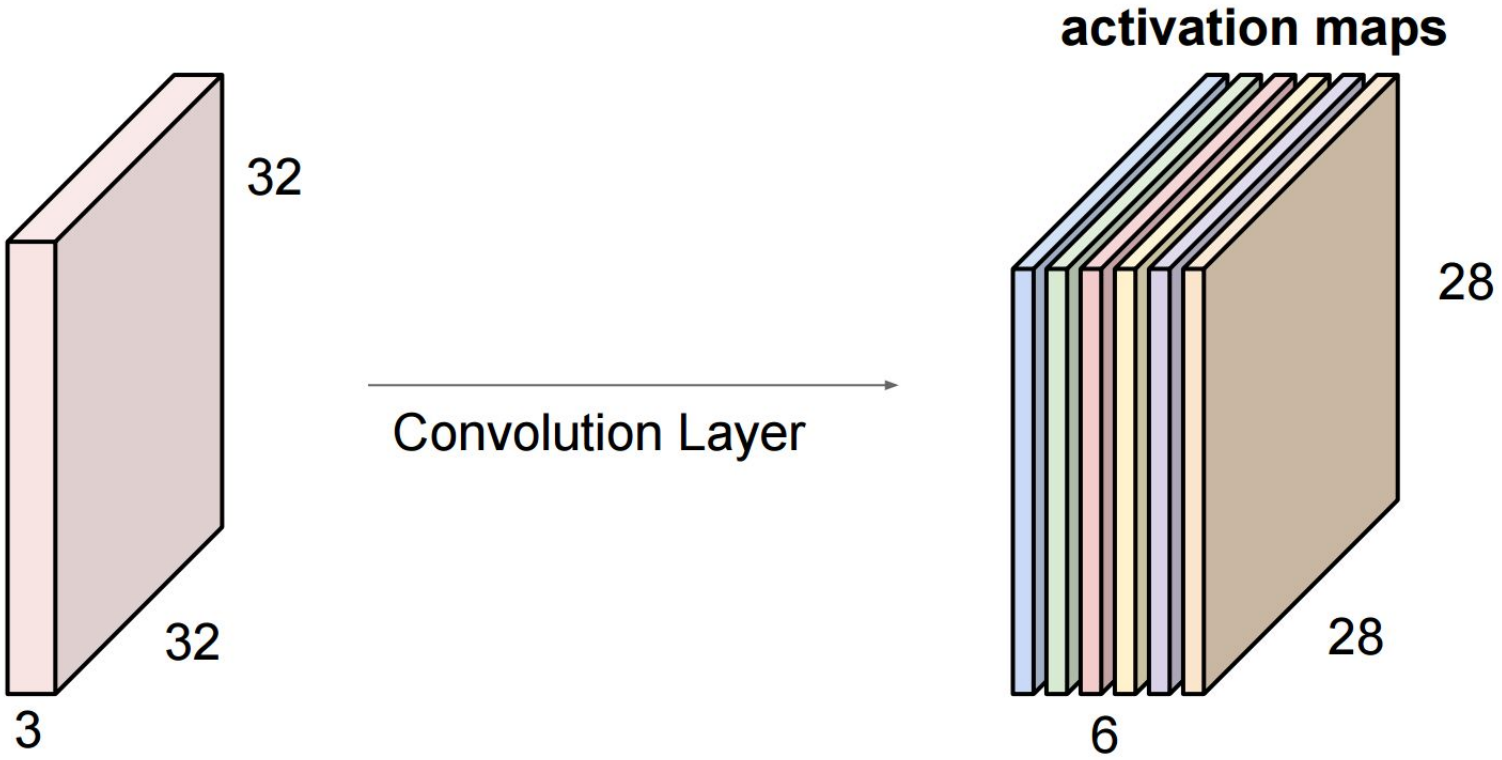
# Convolution

## Convolution Layer

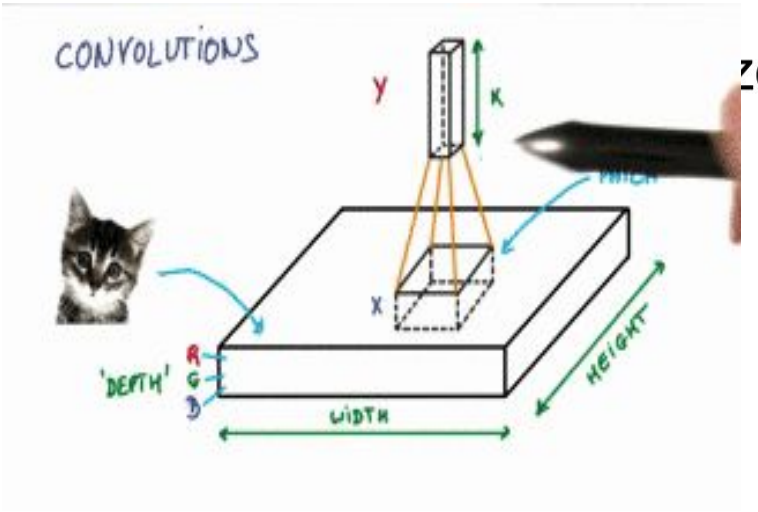
consider a second, green filter



For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



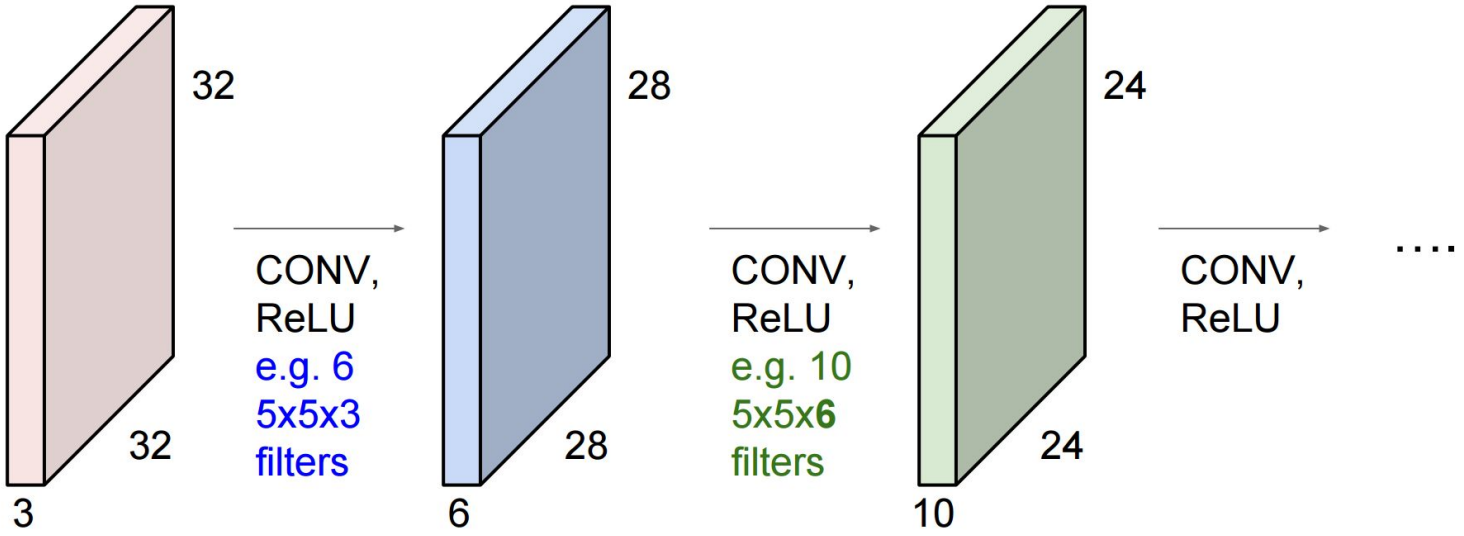

We stack these




ze 28x28x6!

# Convolution

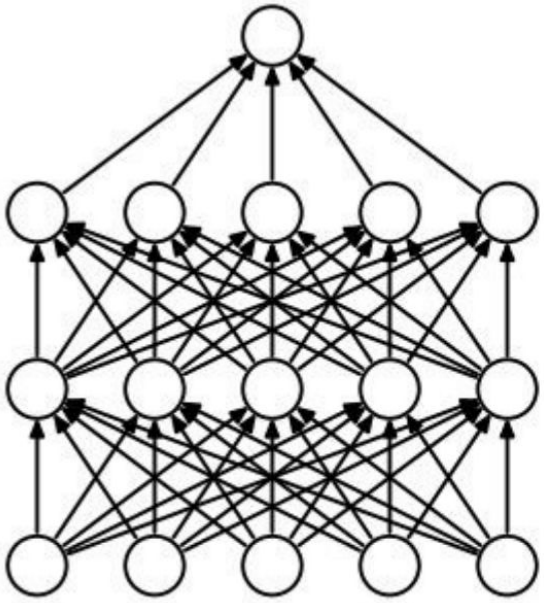
**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



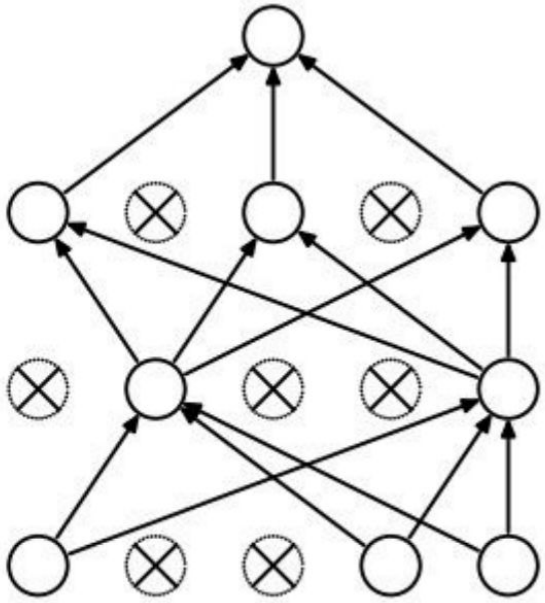
# Dropout

## Regularization: **Dropout**

"randomly set some neurons to zero in the forward pass"



(a) Standard Neural Net



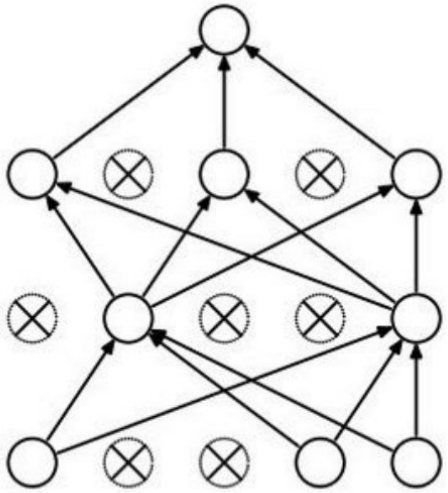
(b) After applying dropout.

*[Srivastava et al., 2014]*

# Dropout

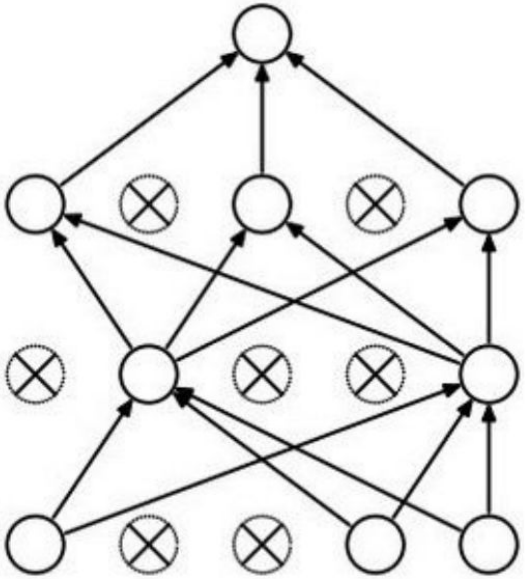
```python
p = 0.5 # probability of keeping a unit active. higher = less dropout

def train_step(X):
  """ X contains the data """

  # forward pass for example 3-layer neural network
  H1 = np.maximum(0, np.dot(W1, X) + b1)
  U1 = np.random.rand(*H1.shape) < p # first dropout mask
  H1 *= U1 # drop!
  H2 = np.maximum(0, np.dot(W2, H1) + b2)
  U2 = np.random.rand(*H2.shape) < p # second dropout mask
  H2 *= U2 # drop!
  out = np.dot(W3, H2) + b3

  # backward pass: compute gradients... (not shown)
  # perform parameter update... (not shown)
```

Example forward pass with a 3-layer network using dropout



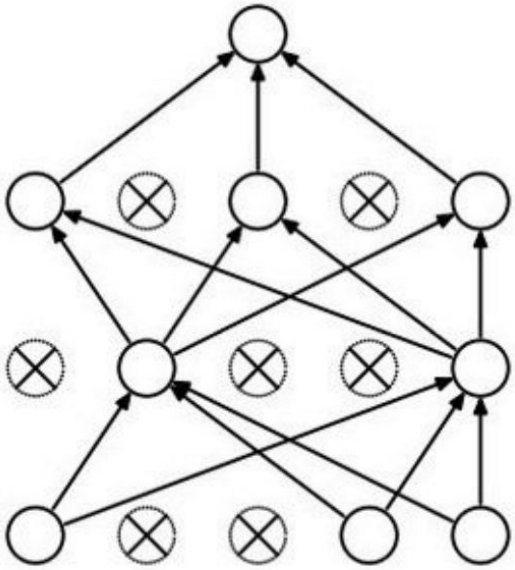
# Dropout

Waaaait a second…
How could this possibly be a good idea?

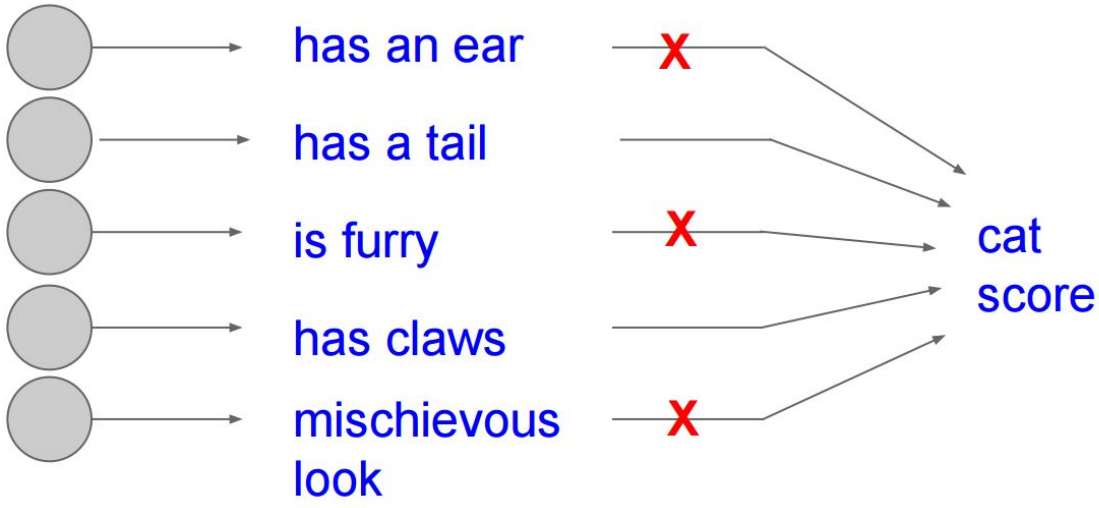


# Dropout

Waaaait a second…
How could this possibly be a good idea?

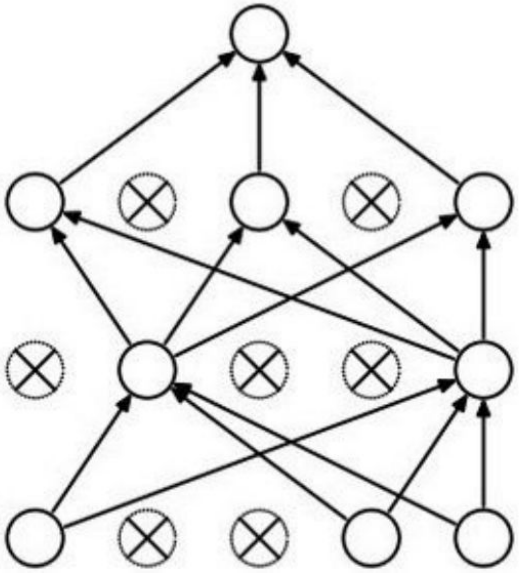


Forces the network to have a redundant representation.



# Dropout

Waaaait a second…
How could this possibly be a good idea?

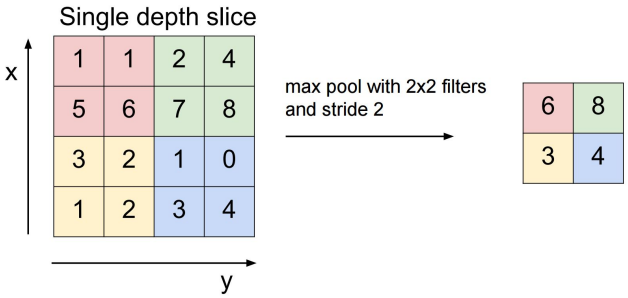


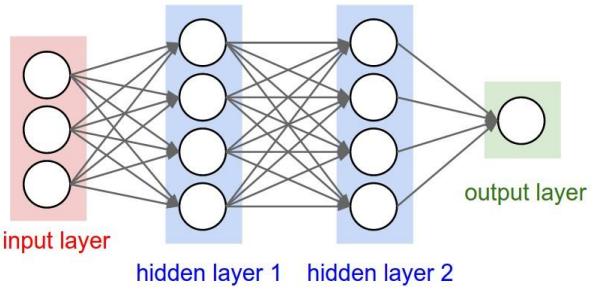
Another interpretation:

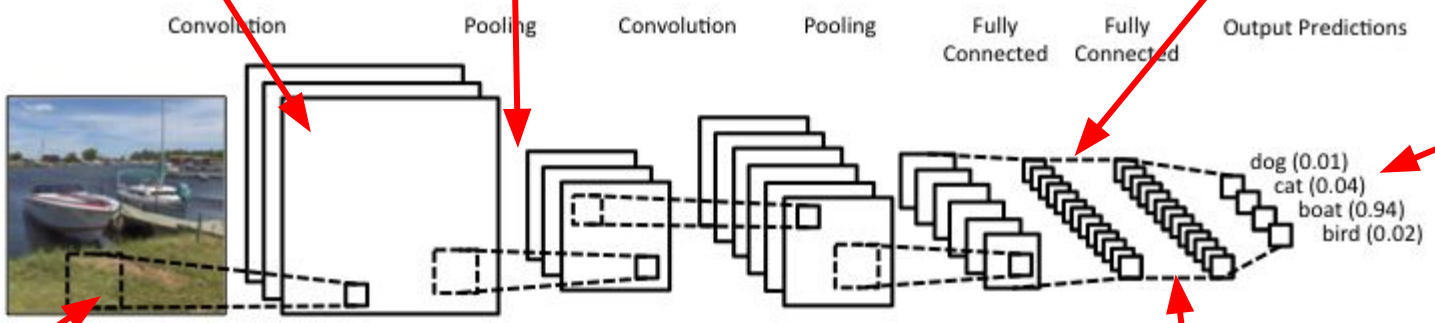
Dropout is training a large ensemble of models (that share parameters).

Each binary mask is one model, gets trained on only ~one datapoint.

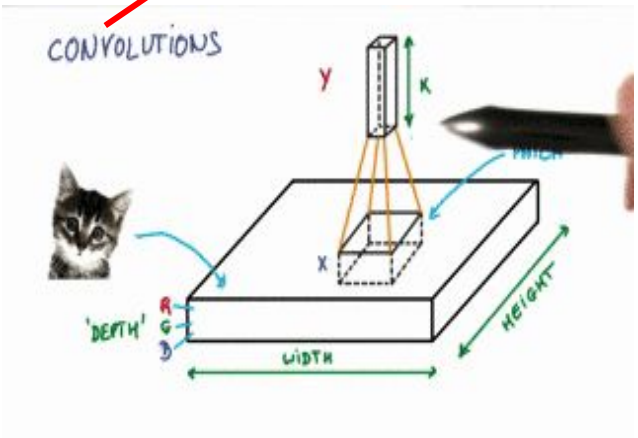
# MAX POOLING

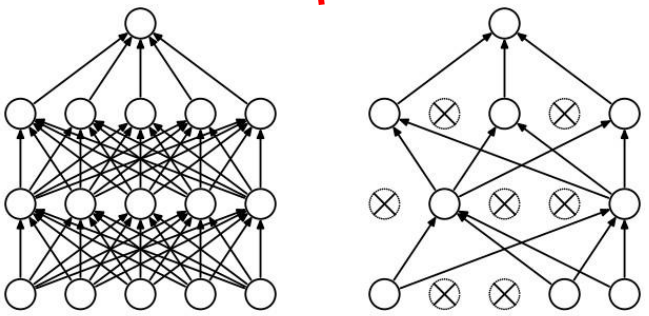



$$\frac{e^{s_k}}{\sum_j e^{s_j}}$$



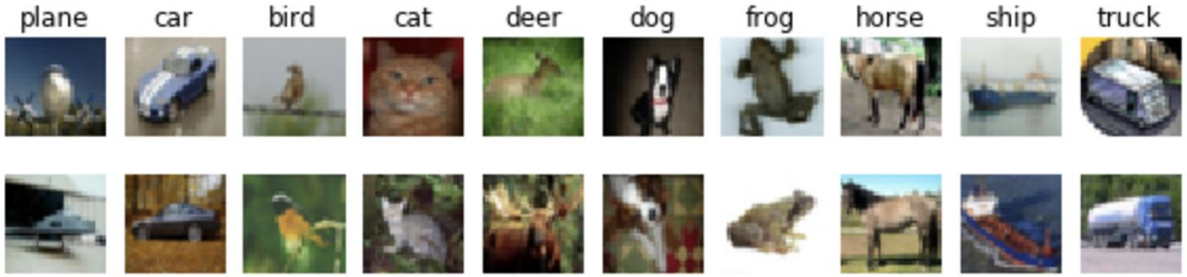
(a) Standard Neural Net (b) After applying dropout.

# Coding

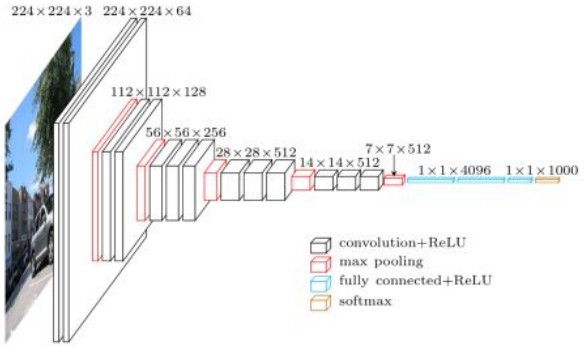
```
from lasagne import layers as ll

nn = ll.InputLayer(shape=(None, 1, 28, 28), input_var)

nn = ll.Conv2DLayer(nn, num_filters=32, filter_size=5)
nn = ll.MaxPool2DLayer(nn, pool_size=(2, 2))

nn = ll.DropOutLayer(nn, 0.5)
nn = ll.DenseLayer(nn, 10, nonlinearity=softmax)

loss = crossentropy(ll.get_output(nn), target_var).mean()
```

# Road Map

1. Training Data




2. Complex Parametric Function + Loss

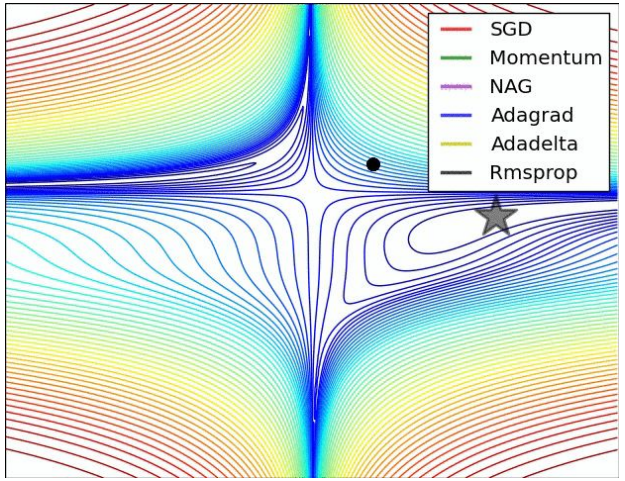


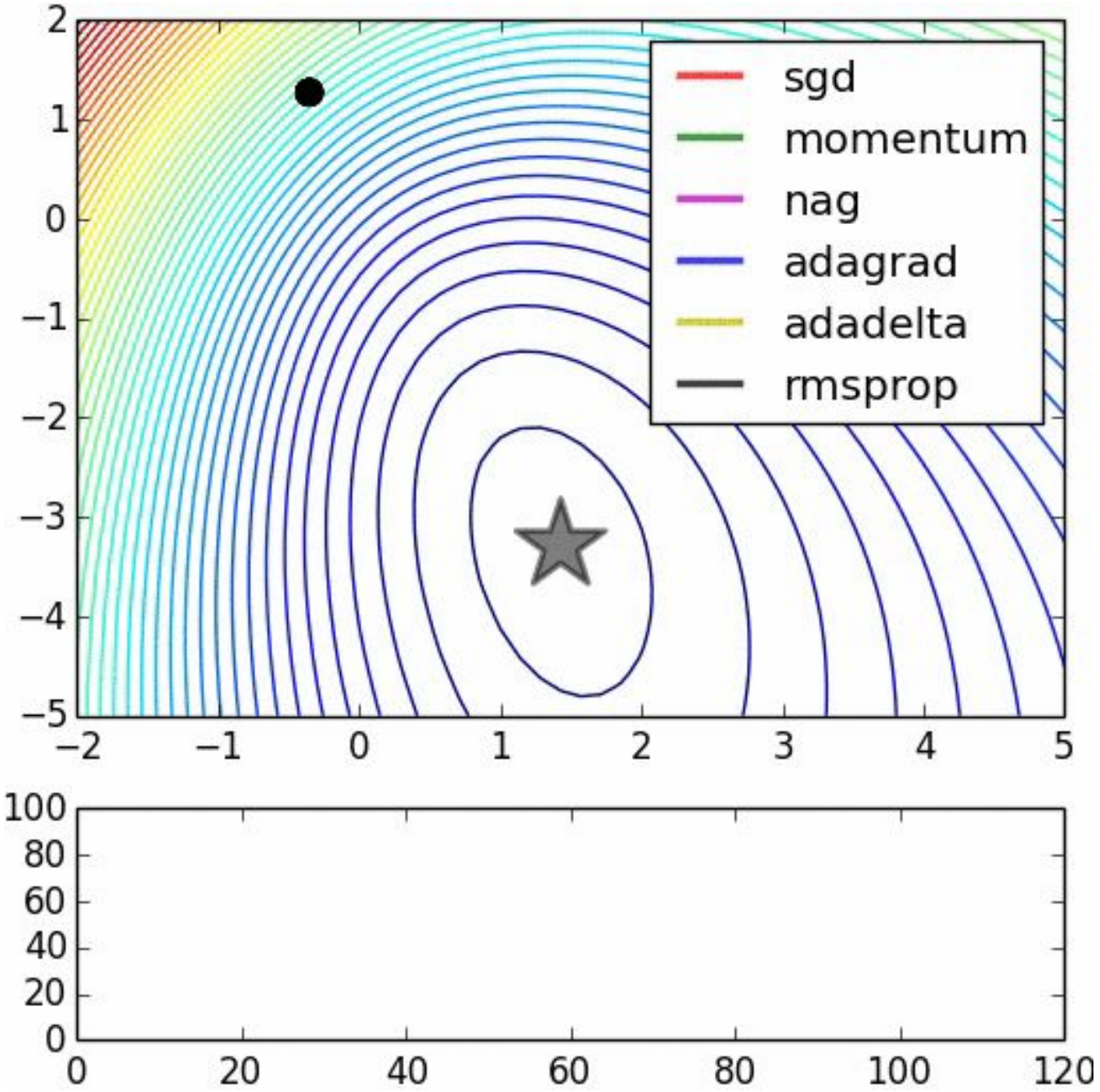

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

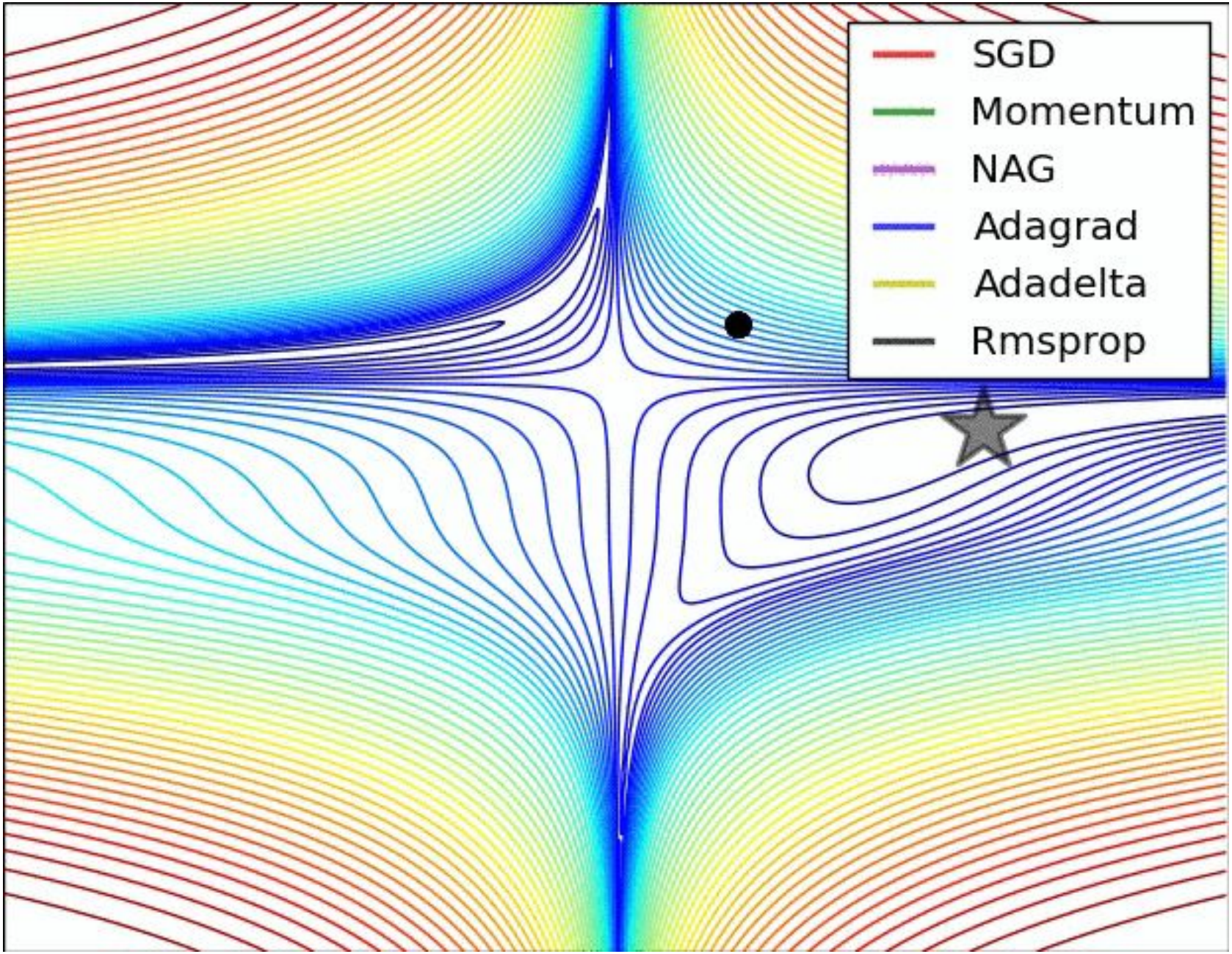
$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \sum_k W_k^2$$

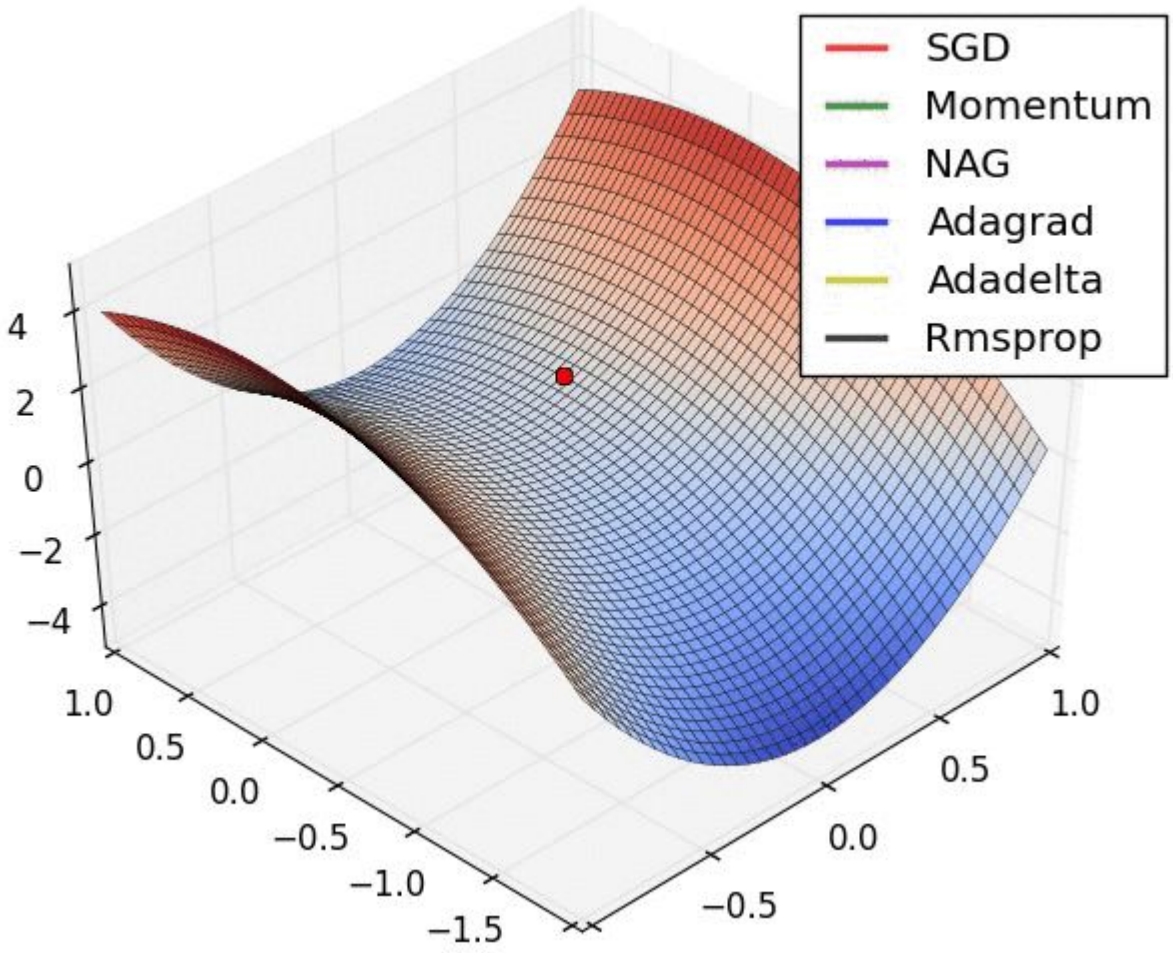
3. How to evaluate Gradients and Optimization

want $\nabla_W L$




sgd
momentum
nag
adagrad
adadelta
rmsprop
2
1
0
−1
−2
−3
−4
−5
−2
−1
0
1
2
3
4
5
100
80
60
40
20
0
0
20
40
60
80
100
120


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

# Administrative

- First assignment will come out at the end of week at worst
- !!! Отзывы о лекции https://goo.gl/forms/zeZiu1fSgrpPGp6T2 !!!

# Next Time

- DL Frameworks
- Modern Conv Arch
- Visualizing, style-transfer, detection, AE

# Good courses/books

- cs231n.stanford.edu, https://goo.gl/75Zi5m
- udacity.com/course/deep-learning–ud730
- deeplearningbook.org