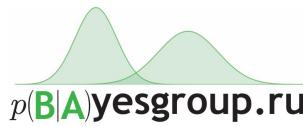


# Machine Learning Course at MIPT

## Deep Introduction to Deep Learning

Arsenii Ashukha

[ml-mipt.github.io](https://ml-mipt.github.io), [ars-ashuha.ru](http://ars-ashuha.ru)

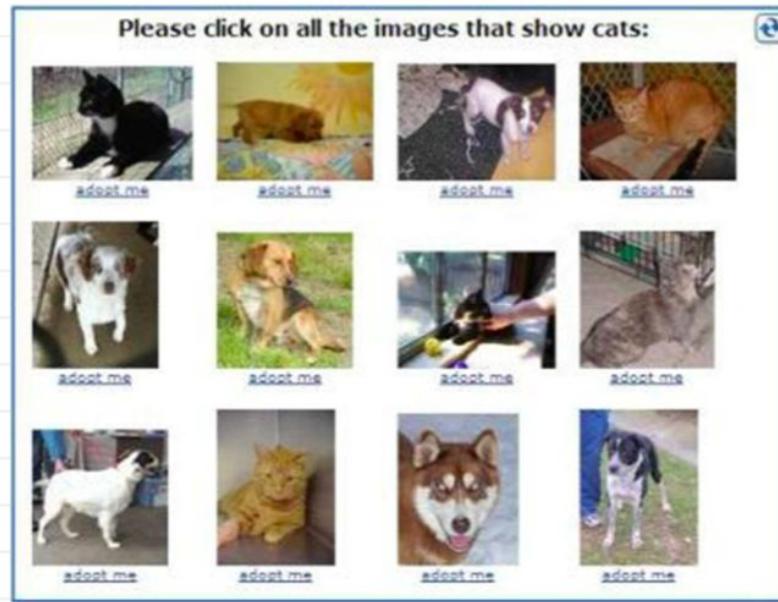


April 4th, 2017

# Deep Neural Nets

2006

CAPSCHA



Computer vision = 60%

$$0.6^{12} = 0.00217$$

# 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 else

#	Δ1w	Team Name * <small>in the money</small>	Score ⓘ	Entries	Last Submission UTC (Best – Last)
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

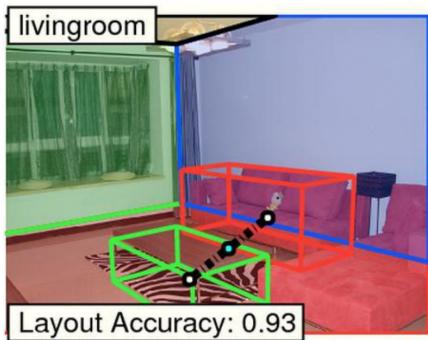
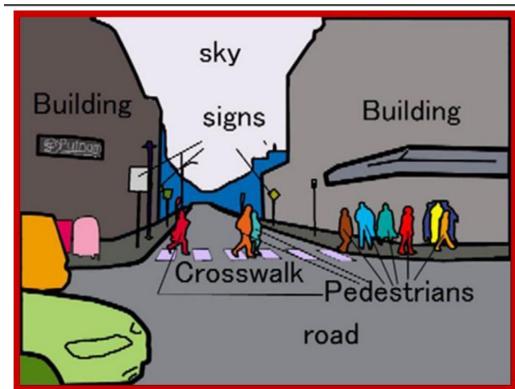
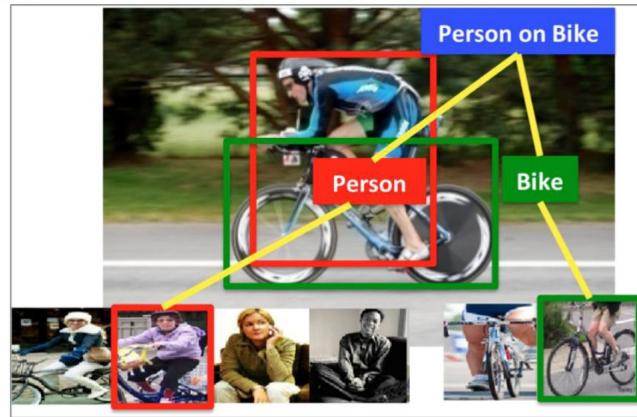
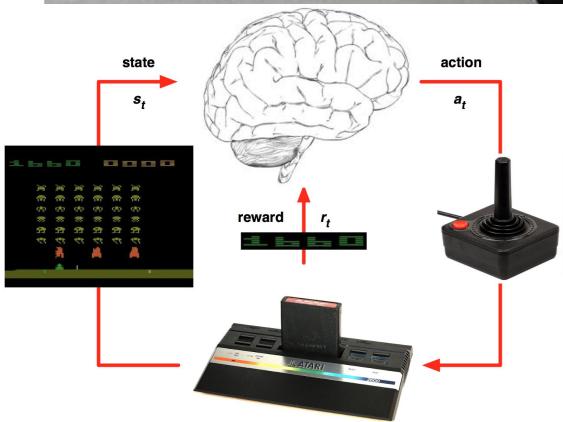
$0.989^{12} = 0.875$

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

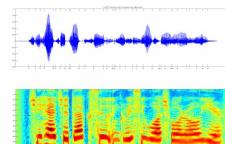
# Real world problems



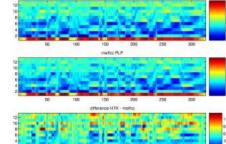
- Object detection
- Action classification
- Image captioning
- ...



## Audio Features



Spectrogram



# Image Classification

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



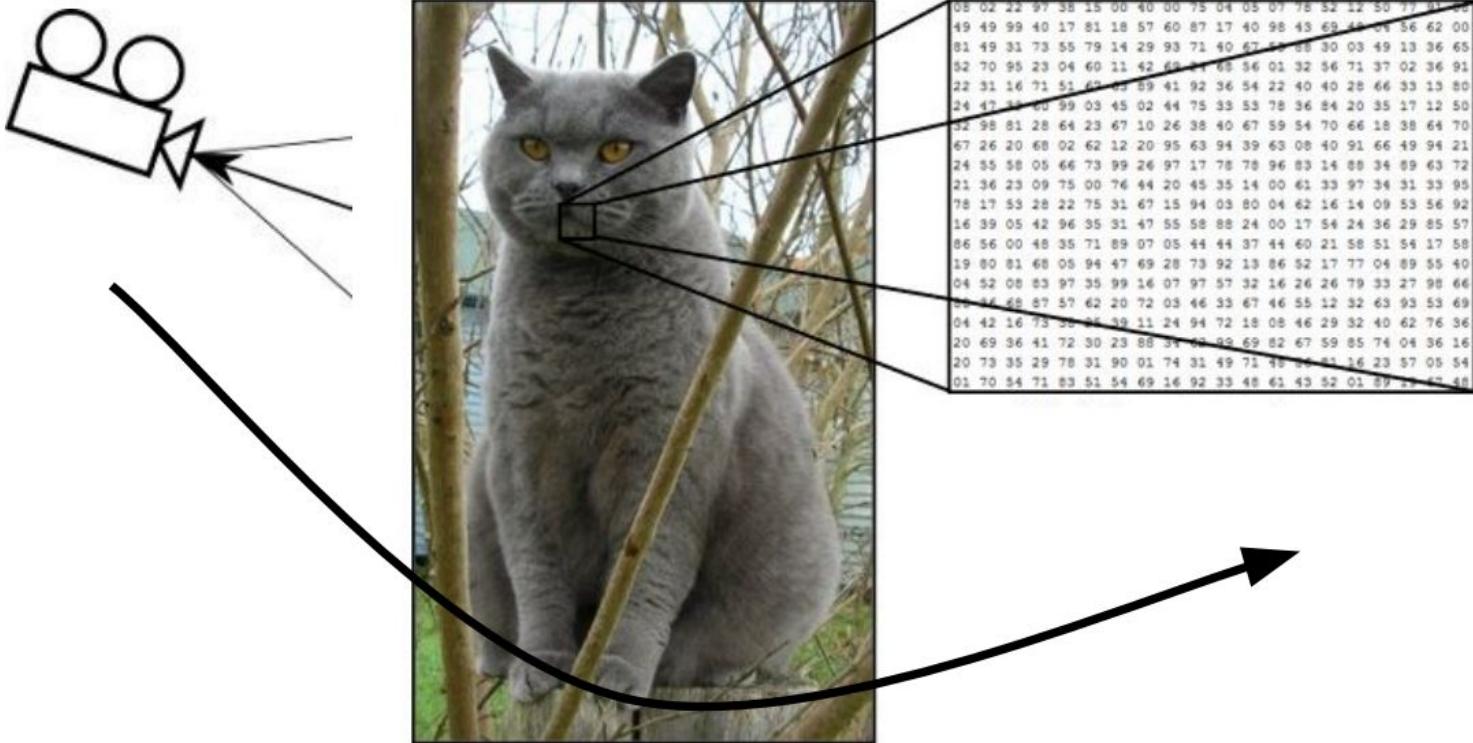
(assume given set of discrete labels)  
{dog, cat, truck, plane, ...}



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: Intra-class 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
```

Example training set



Q: How to make an image classifier :)?

# Parametric Approach



image    parameters

$$f(\mathbf{x}, \mathbf{W})$$

10 numbers,  
indicating class  
scores

**[32x32x3]**

array of numbers 0...1  
(3072 numbers total)

# Linear Classifier

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



**10** numbers,  
indicating class  
scores

**[32x32x3]**

array of numbers 0...1

Q: How many parameters are here ?

# Linear Classifier

## Parametric approach: Linear classifier

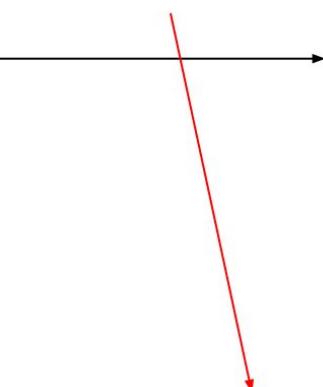


$$f(x, W) = \boxed{W} \boxed{x} \quad \textcolor{blue}{3072 \times 1}$$

$10 \times 1 \qquad \textcolor{red}{10 \times 3072}$

A mathematical equation showing the computation of a linear classifier. The function  $f(x, W)$  is defined as the product of the weight matrix  $W$  and the input vector  $x$ . The dimensions of the input vector  $x$  are given as  $10 \times 1$ , and the dimensions of the weight matrix  $W$  are given as  $10 \times 3072$ .

$[32 \times 32 \times 3]$   
array of numbers 0...1



10 numbers,  
indicating class  
scores

parameters, or “weights”

# Linear Classifier

Parametric approach: **Linear classifier**



$$f(x, W) = \boxed{W} \boxed{x} \quad \begin{matrix} 3072 \times 1 \\ (+b) \quad 10 \times 1 \end{matrix}$$

$f(x, W)$  is the function output, where  $x$  is the input vector and  $W$  is the weight matrix. The dimensions of the input vector  $x$  are  $10 \times 1$ , and the dimensions of the weight matrix  $W$  are  $10 \times 3072$ .

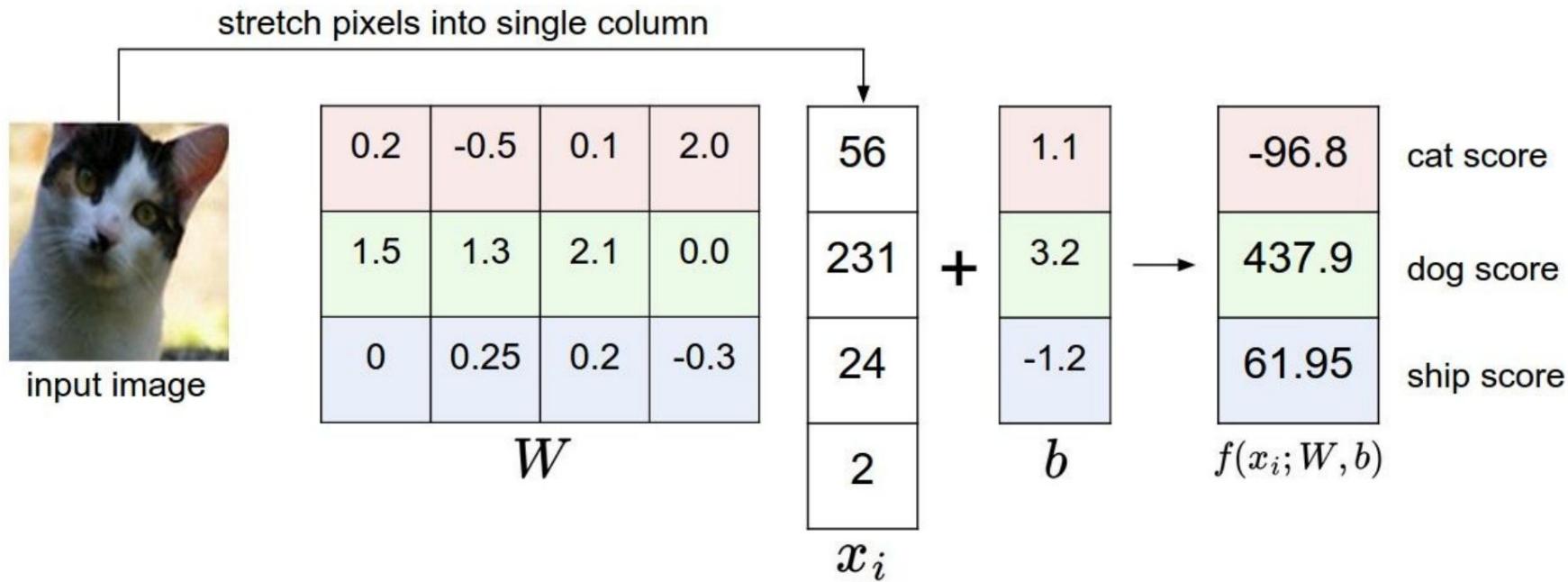
$[32 \times 32 \times 3]$   
array of numbers 0...1

10 numbers,  
indicating class  
scores

parameters, or “weights”

# Linear Classifier

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



# Linear Classifier

## Interpreting a Linear Classifier

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



truck

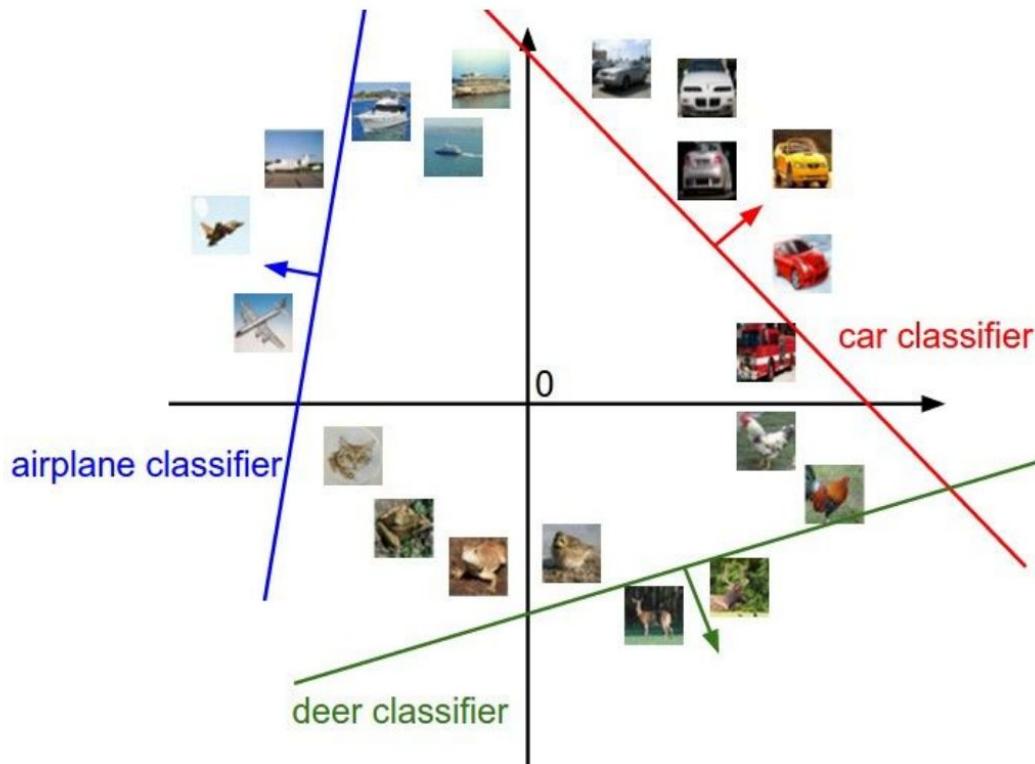


$$f(x_i, W, b) = Wx_i + b$$

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

# Linear Classifier

## Interpreting a Linear Classifier



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



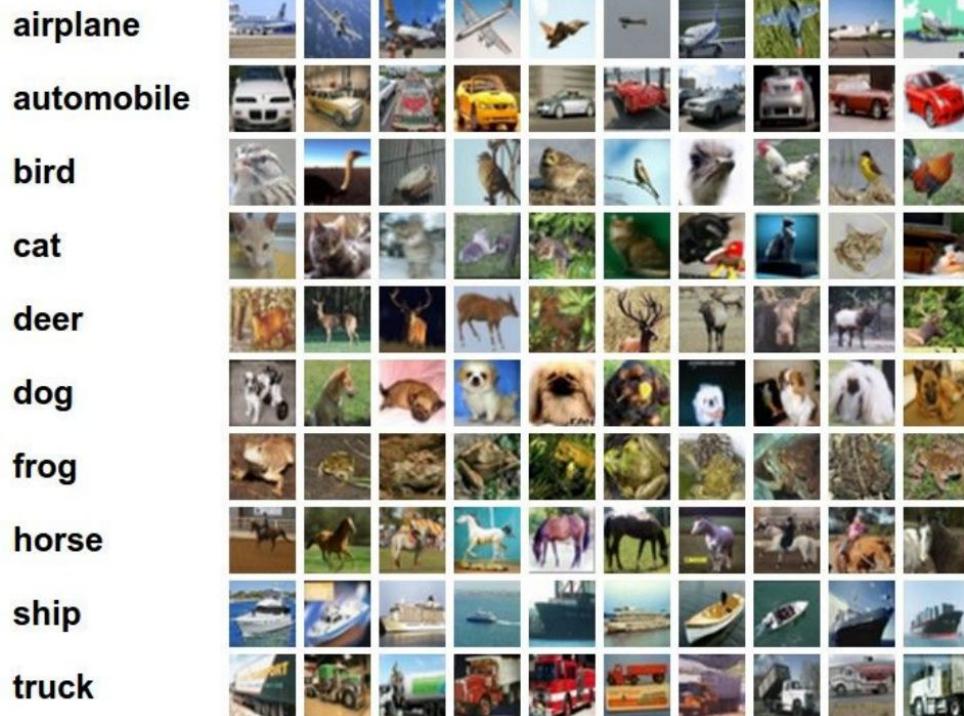
**[32x32x3]**  
array of numbers 0...1  
(3072 numbers total)

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



## Linear Classifier

# 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	<b>6.04</b>	4.64
bird	0.09	5.31	2.65
cat	<b>2.9</b>	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	<b>-4.34</b>
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	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

## 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)



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

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

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

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

3.2

5.1

-1.7

---

in summary:  $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

# TODO



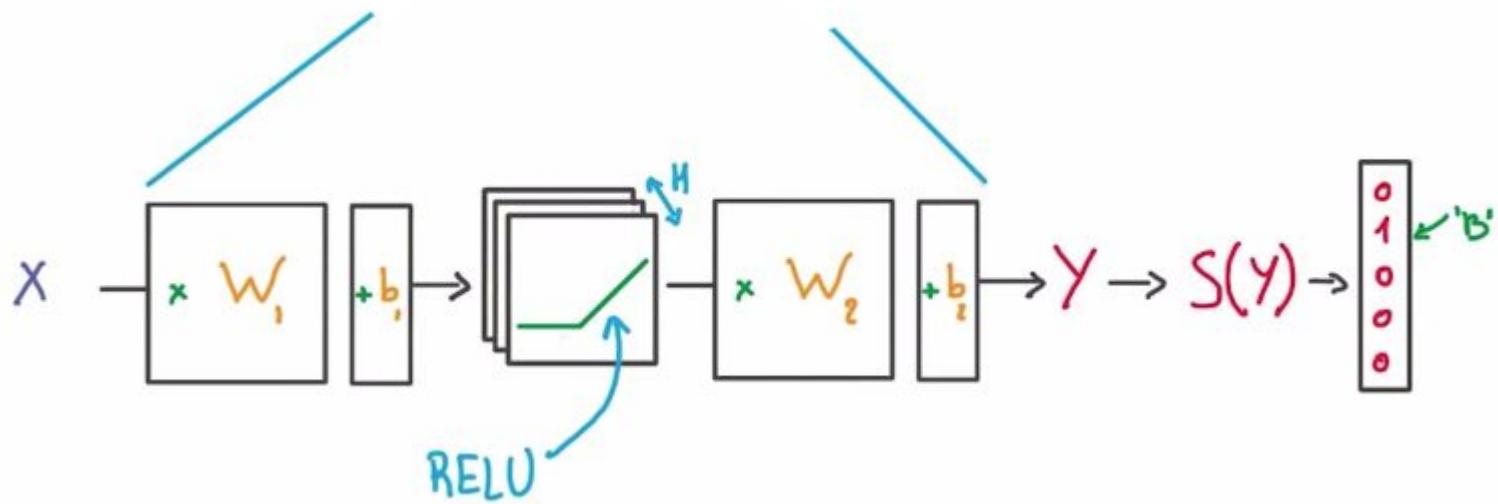
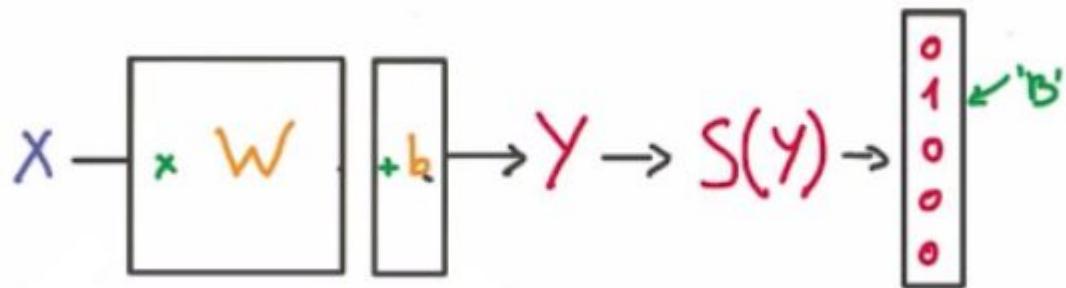
airplane	-3.45	-0.51	3.42
automobile	-8.87	<b>6.04</b>	4.64
bird	0.09	5.31	2.65
cat	<b>2.9</b>	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	<b>-4.34</b>
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

## 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



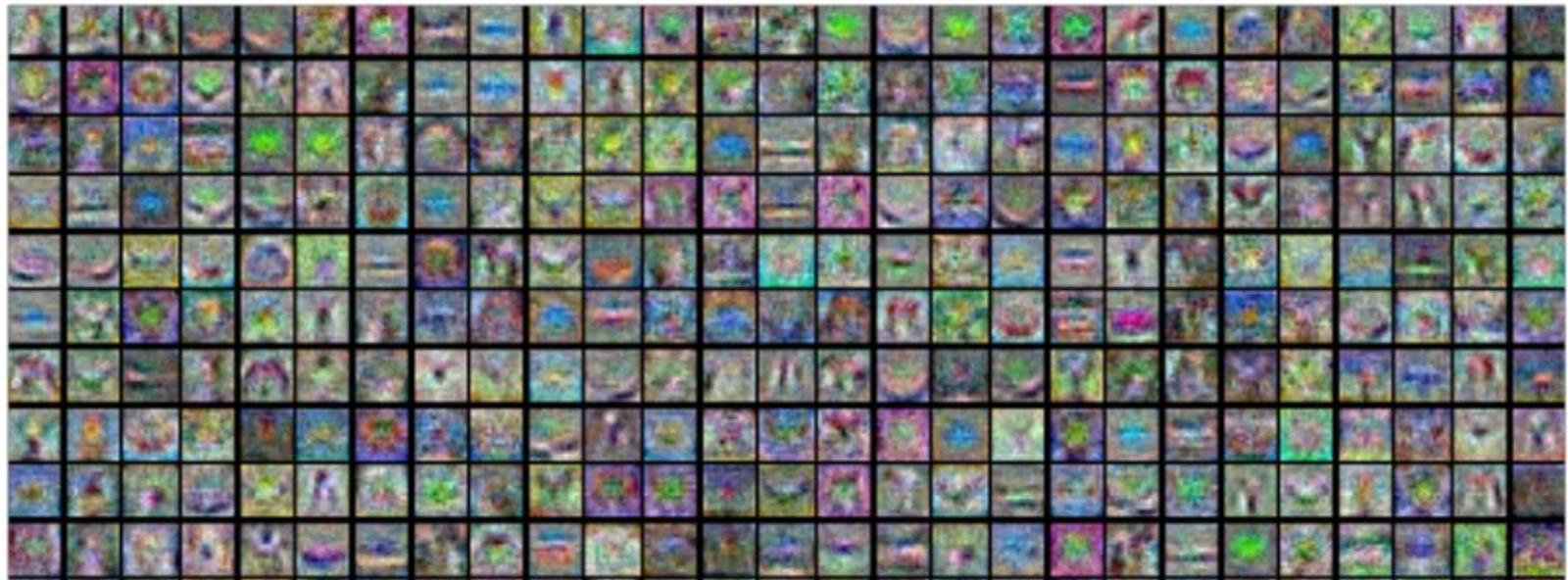
Q: What does DNN do in Russian?

# Linear Classifier

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

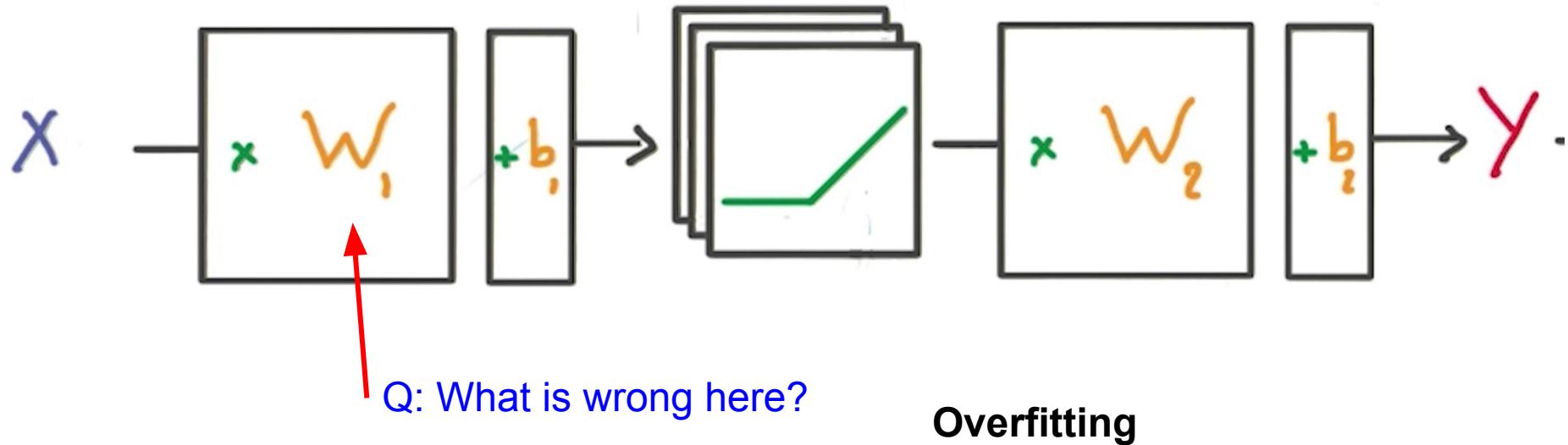


- ▶ Now we got a lot of feature extractor in matrix  $W_1$  we call it Fist Layer

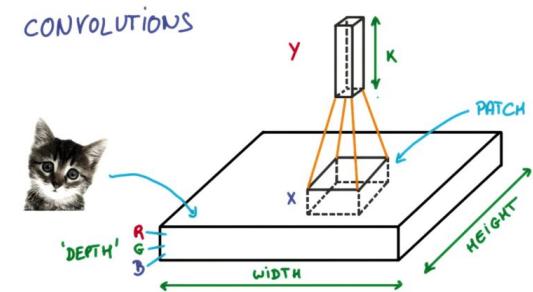


[linear](#) vs [non-linear](#) vs [deep nonlinear](#)

# Statistical Invariance



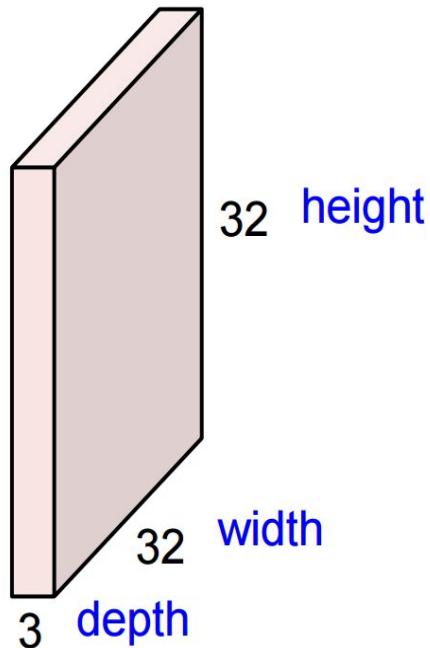
**200x200x3 x 1000**



# Convolution

## Convolution Layer

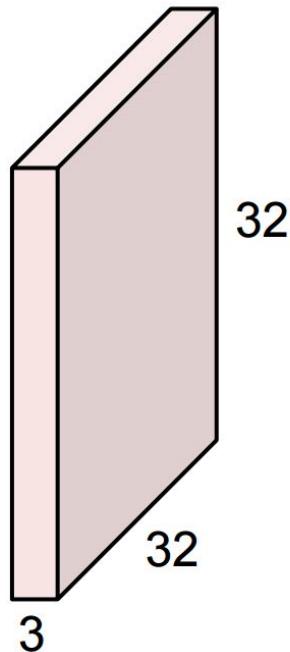
32x32x3 image



## Convolution

# Convolution Layer

32x32x3 image



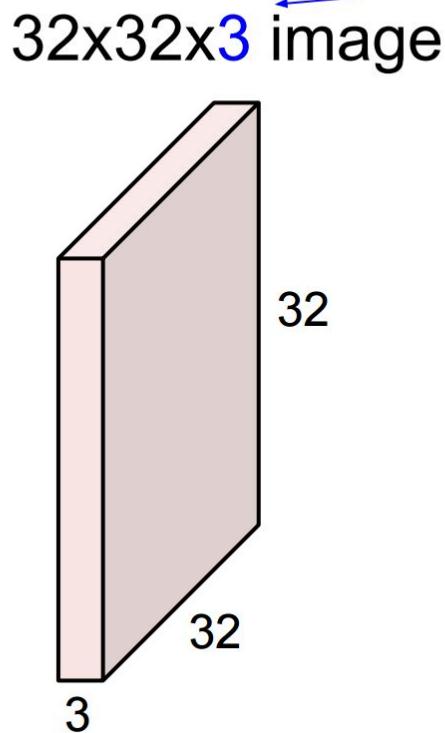
5x5x3 filter



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

# Convolution

## Convolution Layer



$32 \times 32 \times 3$  image

$5 \times 5 \times 3$  filter

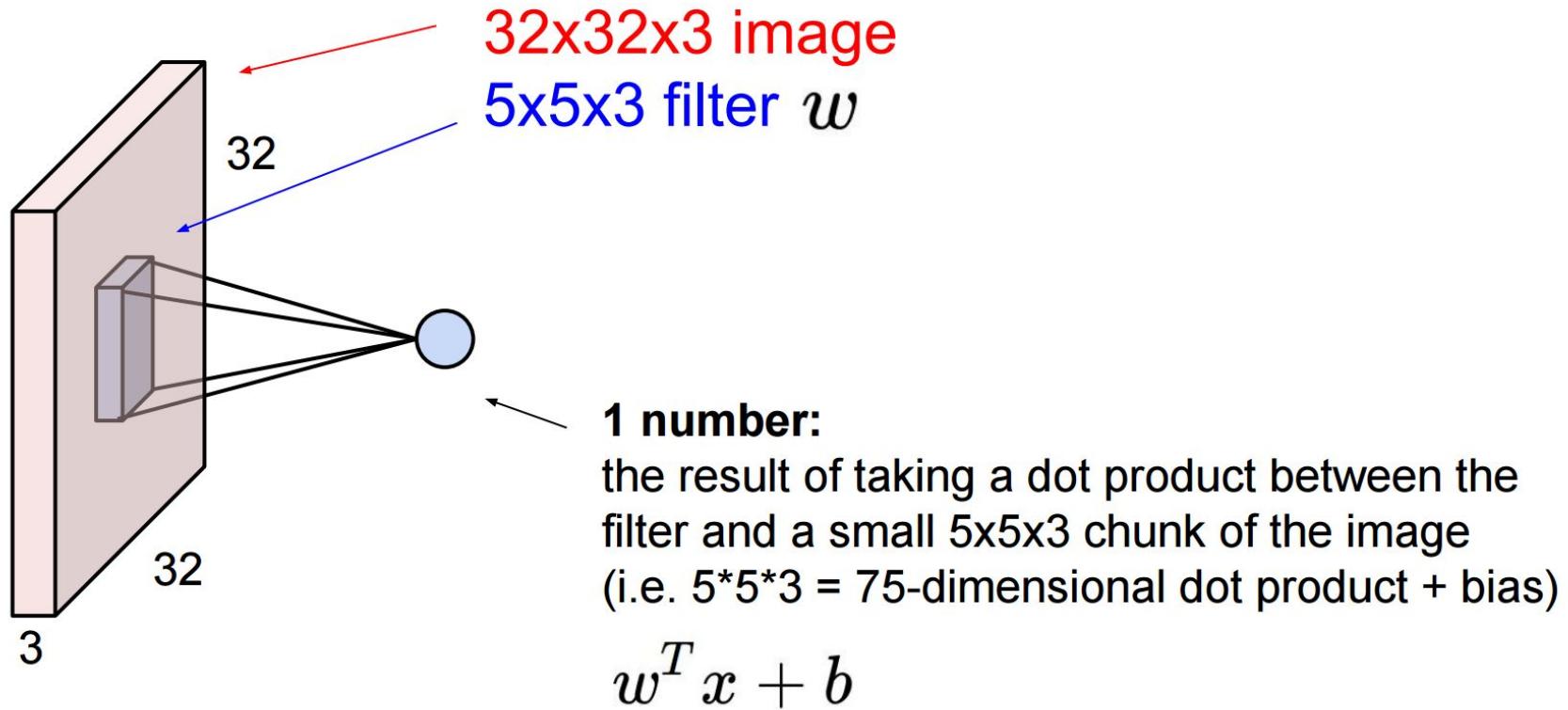
Filters always extend the full depth of the input volume



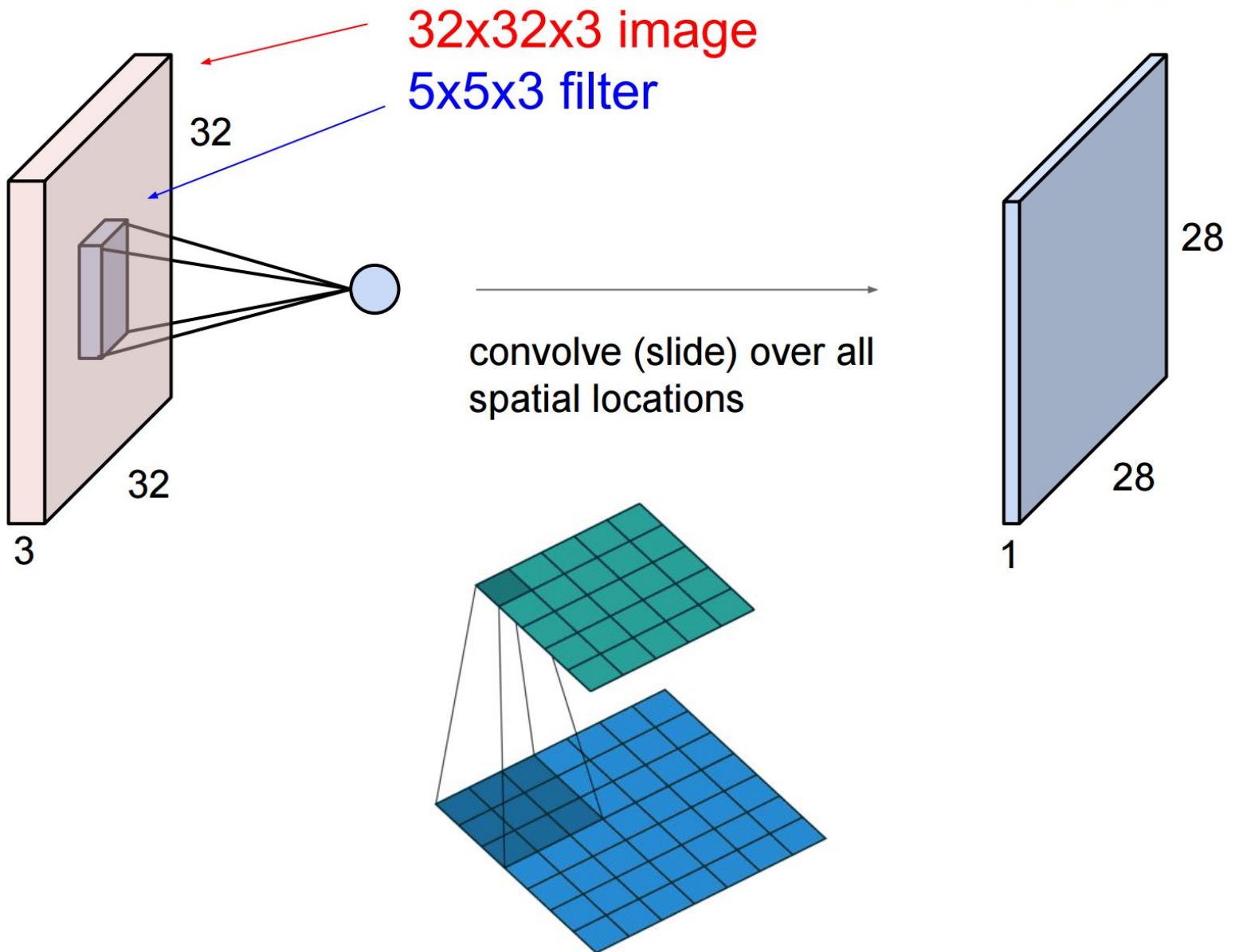
**Convolve** the filter with the image  
i.e. “slide over the image spatially,  
computing dot products”

# Convolution

## Convolution Layer



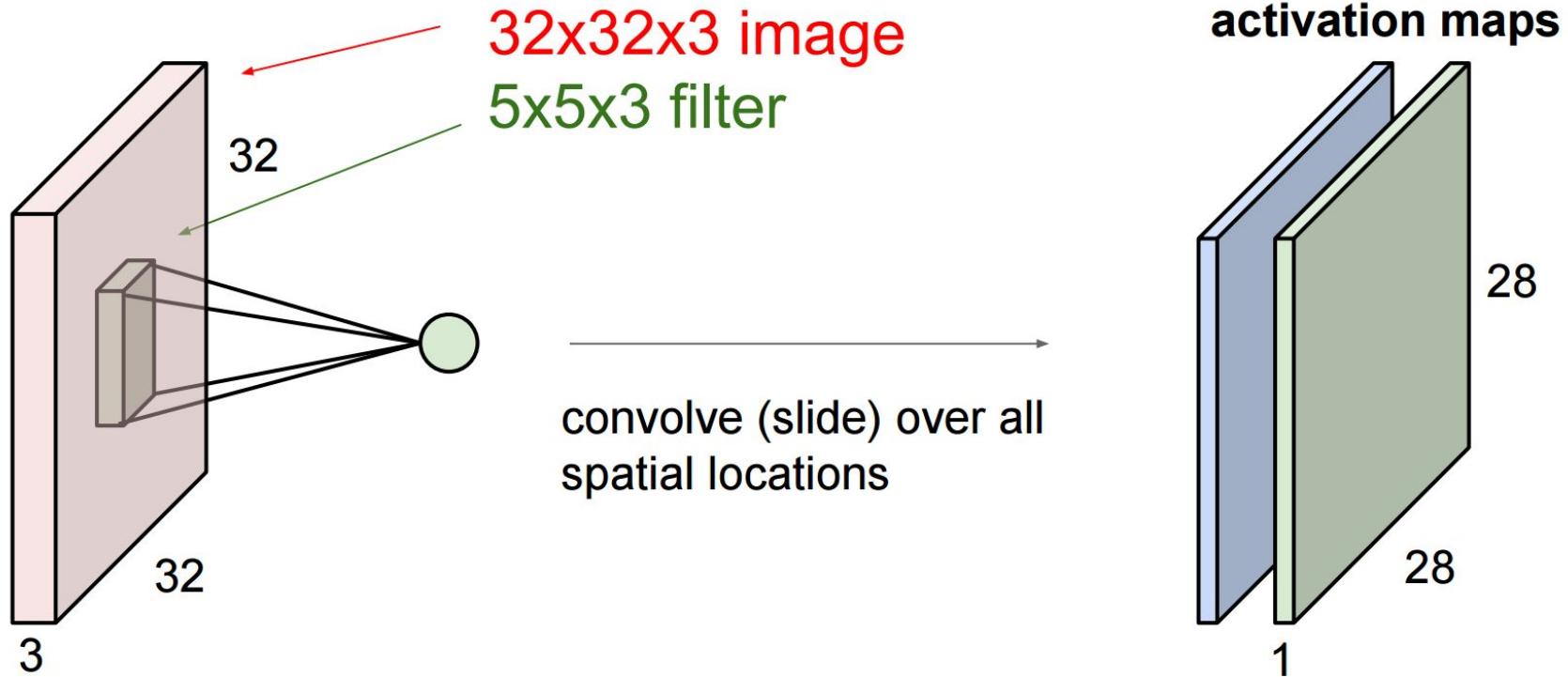
# Convolution Layer



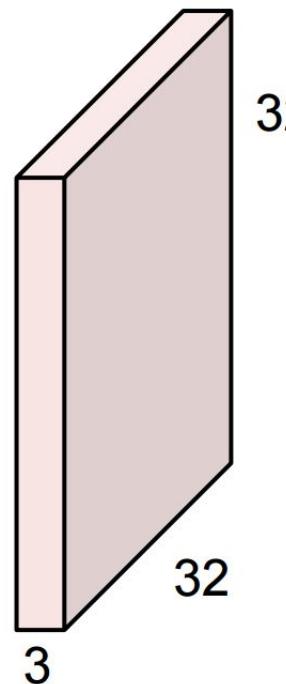
# Convolution

## Convolution Layer

consider a second, green filter

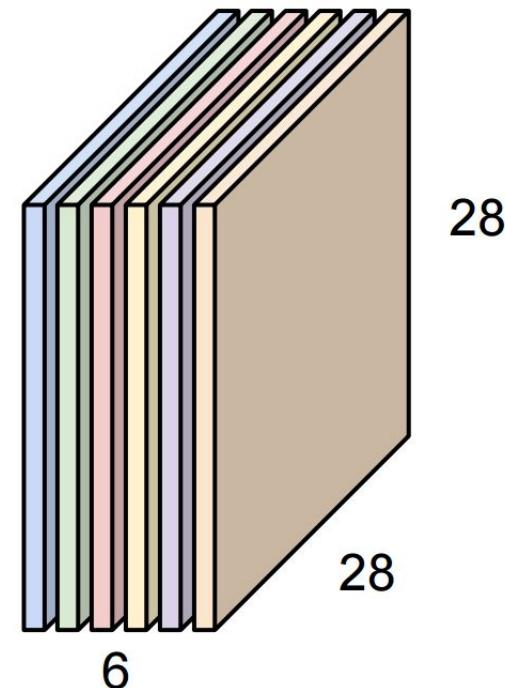


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



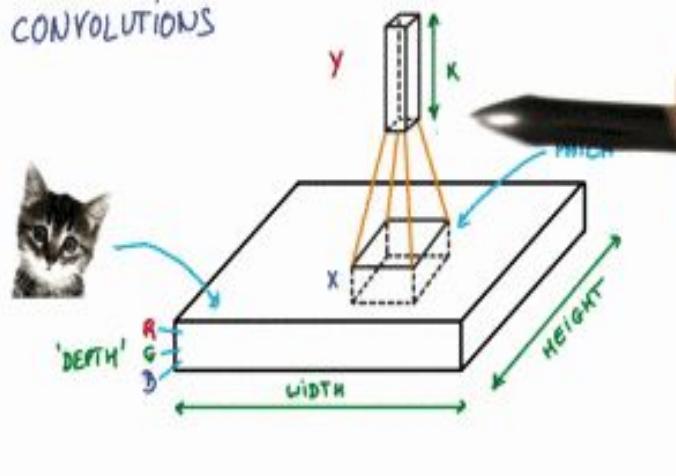
Convolution Layer

activation maps



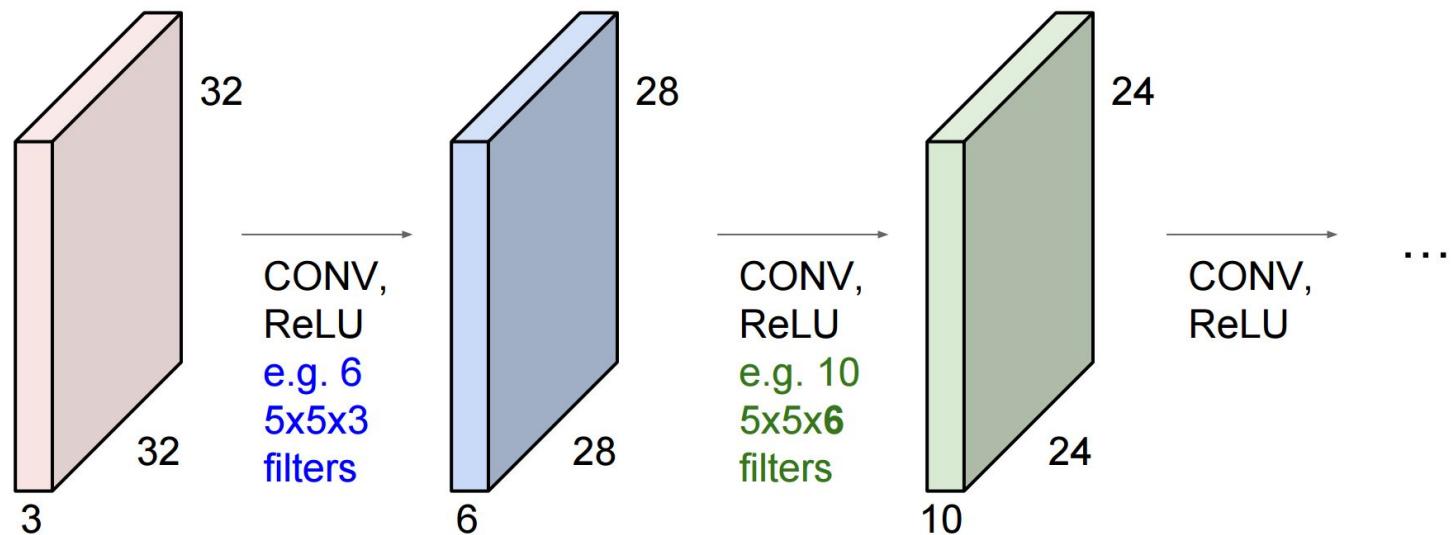
We stack these

CONVOLUTIONS  
size 28x28x6!



# Convolution

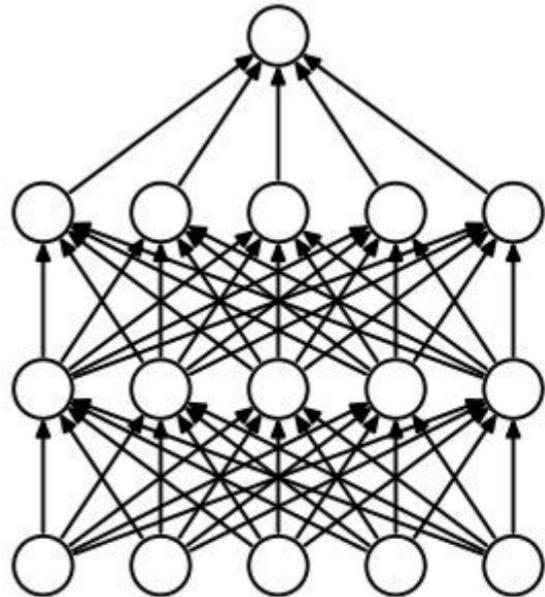
**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



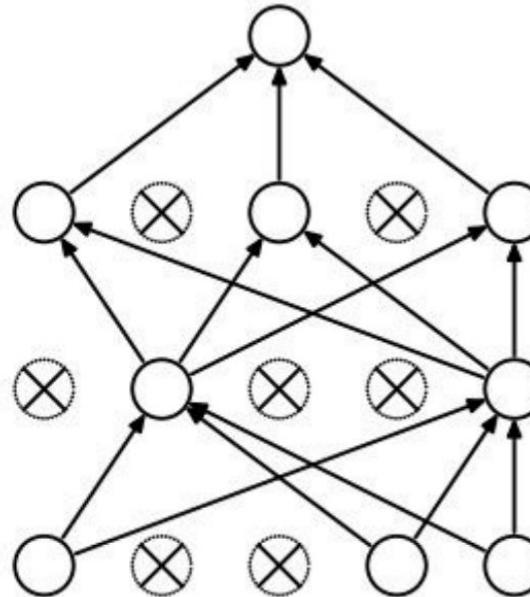
# Dropout

## 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

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

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

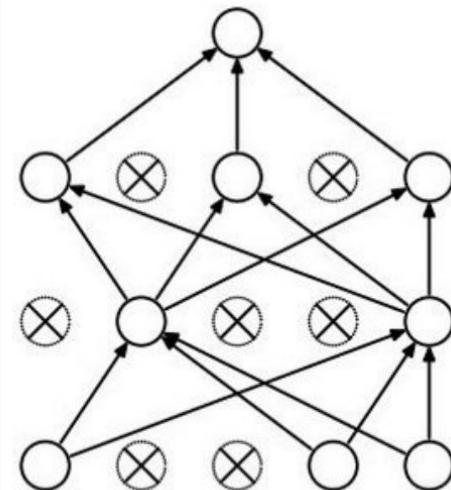
    # forward pass for example 3-layer neural network
    H1 = np.maximum(0, np.dot(W1, X) + b1)
    U1 = np.random.rand(*H1.shape) < p # first dropout mask
    H1 *= U1 # drop!

    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)
```

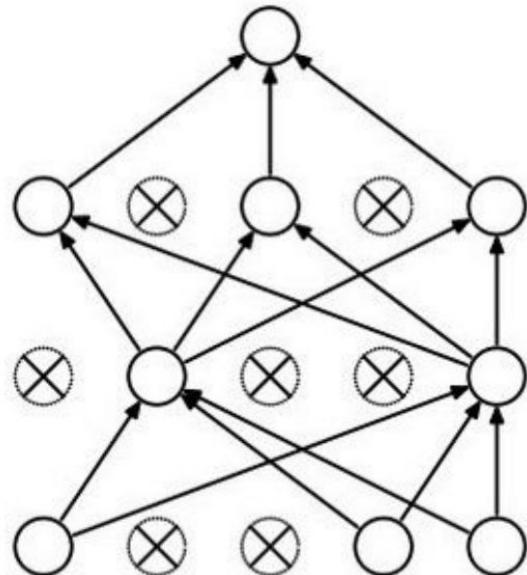
Example forward pass with a 3-layer network using dropout



# Dropout

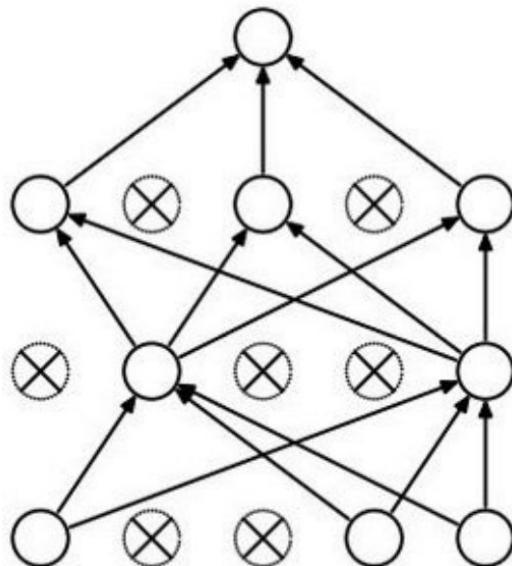
Waaaait a second...

How could this possibly be a good idea?



# Dropout

Waaaaait a second...  
How could this possibly be a good idea?



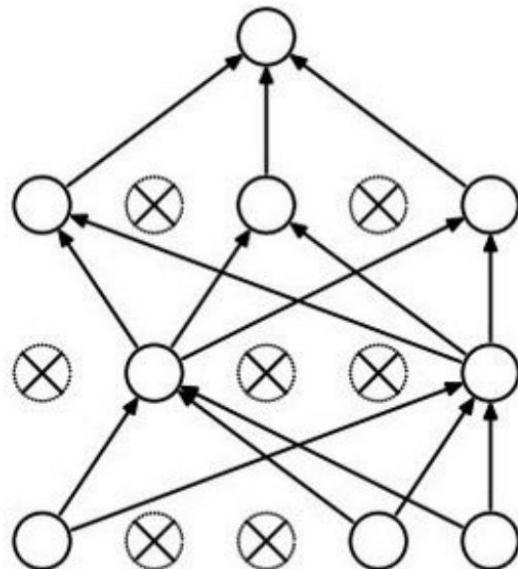
Forces the network to have a redundant representation.



# Dropout

Waaaait a second...

How could this possibly be a good idea?

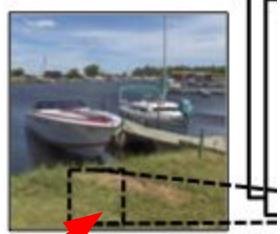
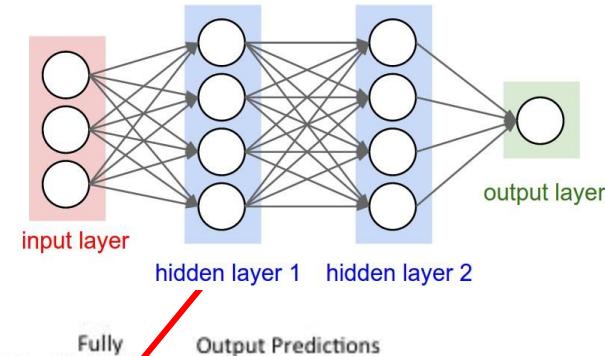
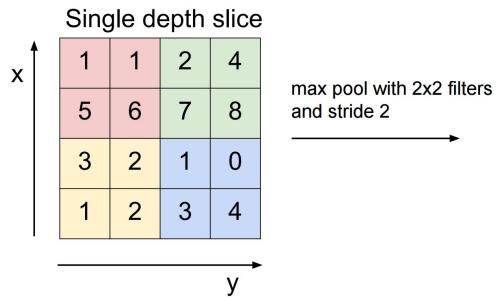


Another interpretation:

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



Convolution

Pooling

Convolution

Pooling

Fully

Fully

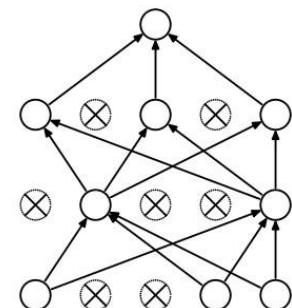
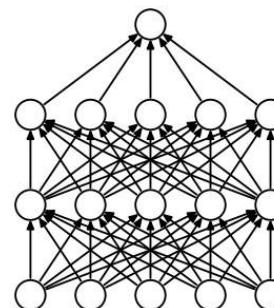
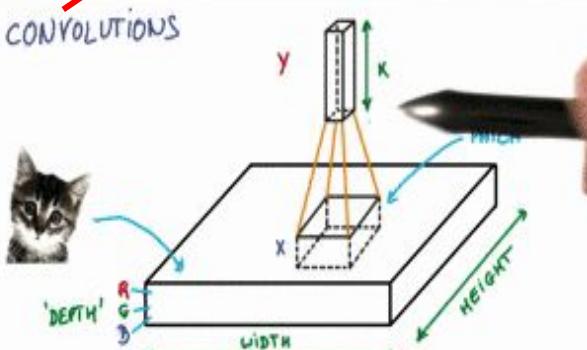
# image parameters

# $f(\mathbf{x}, \mathbf{W})$

.01)  
(0.04)  
oat (0.94)  
bird (0.02)

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

CONVOLUTIONS

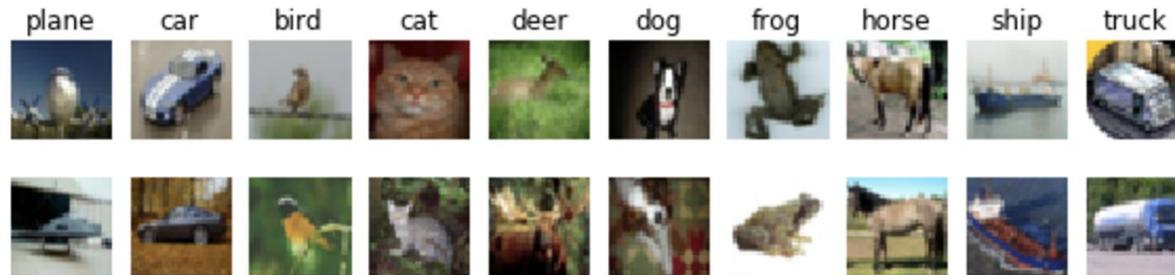


# Coding

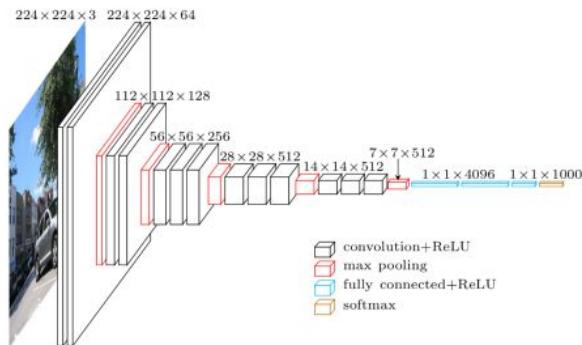
```
from lasagne import layers as ll  
  
nn = ll.InputLayer(shape=(None, 1, 28, 28), input_var)  
nn = ll.Conv2DLayer(nn, num_filters=32, filter_size=5)  
nn = ll.MaxPool2DLayer(nn, pool_size=(2, 2))  
nn = ll.DropoutLayer(nn, 0.5)  
nn = ll.DenseLayer(nn, 10, nonlinearity=softmax)  
loss = crossentropy(ll.get_output(nn), target_var).mean()
```

# Road Map

## 1. Training Data



## 2. Complex Parametric Function + Loss

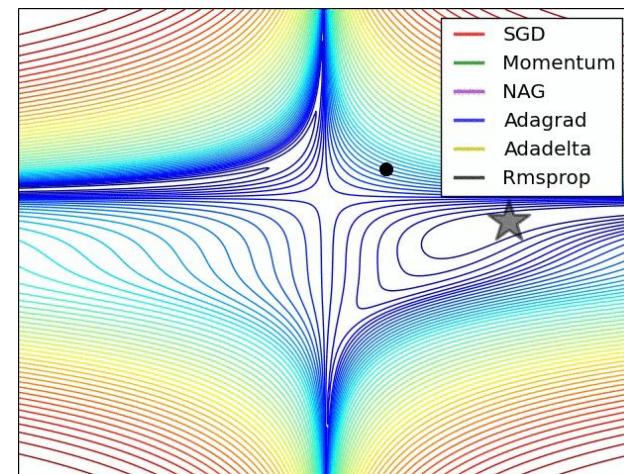


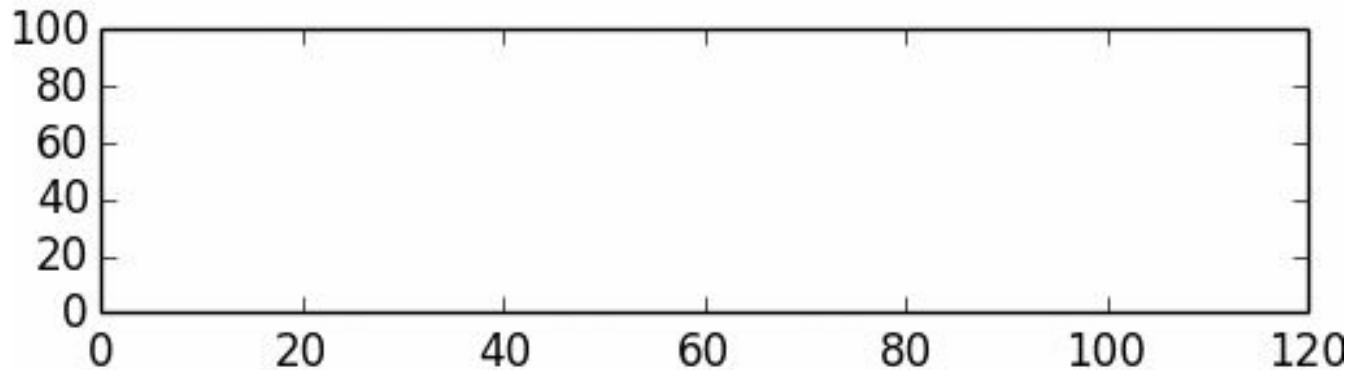
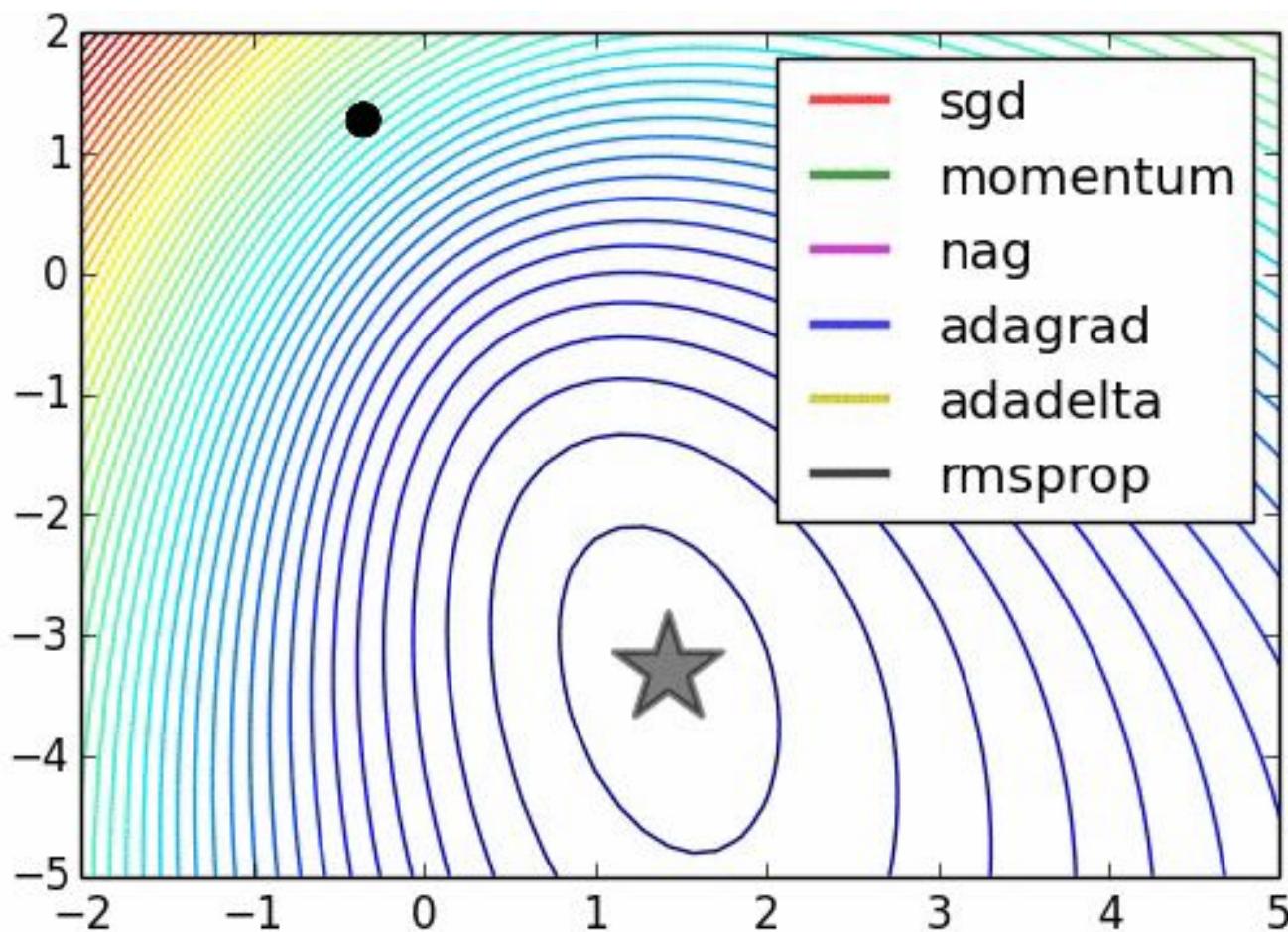
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

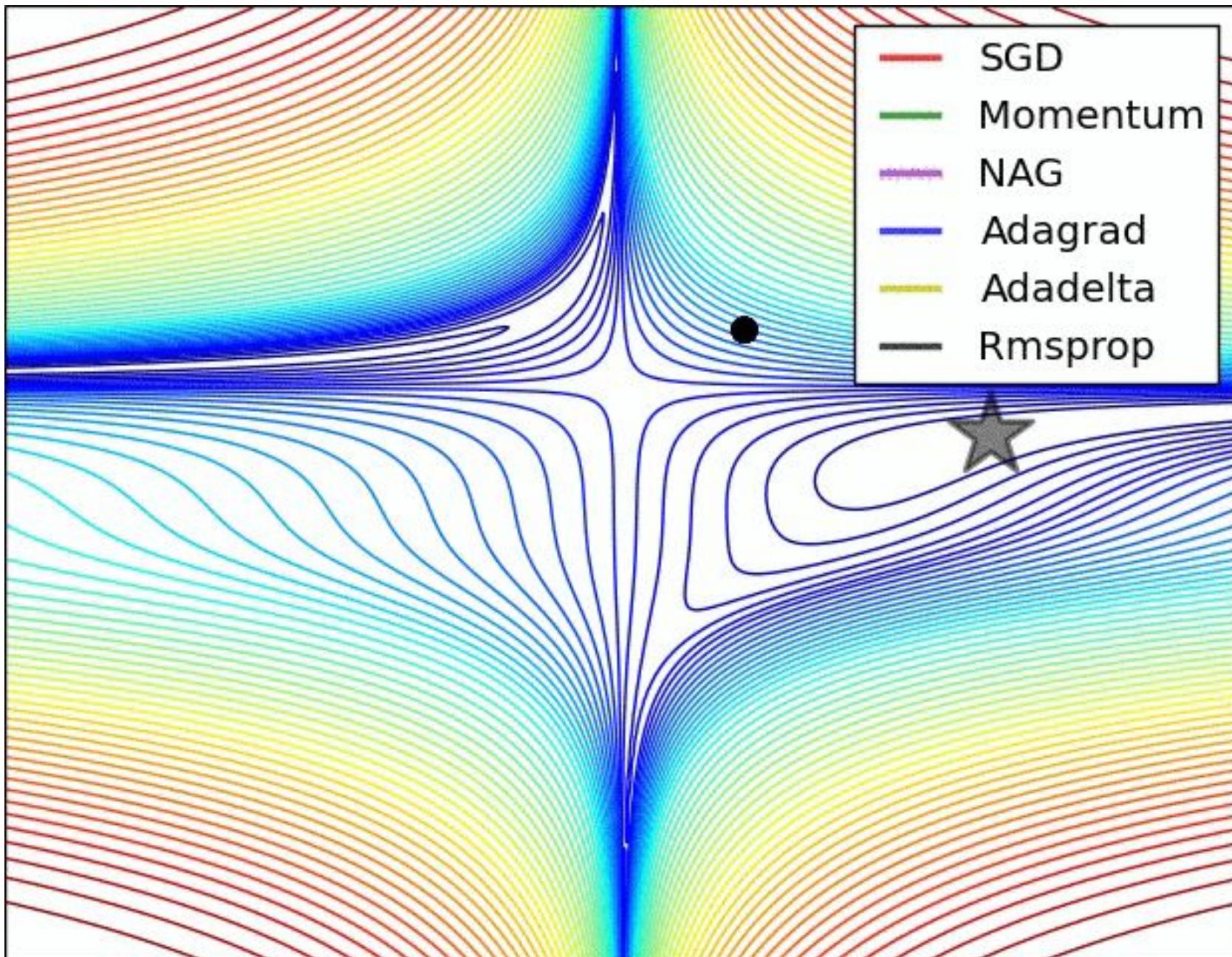
$$L = \frac{1}{N} \sum_{i=1}^N L_i + \sum_k W_k^2$$

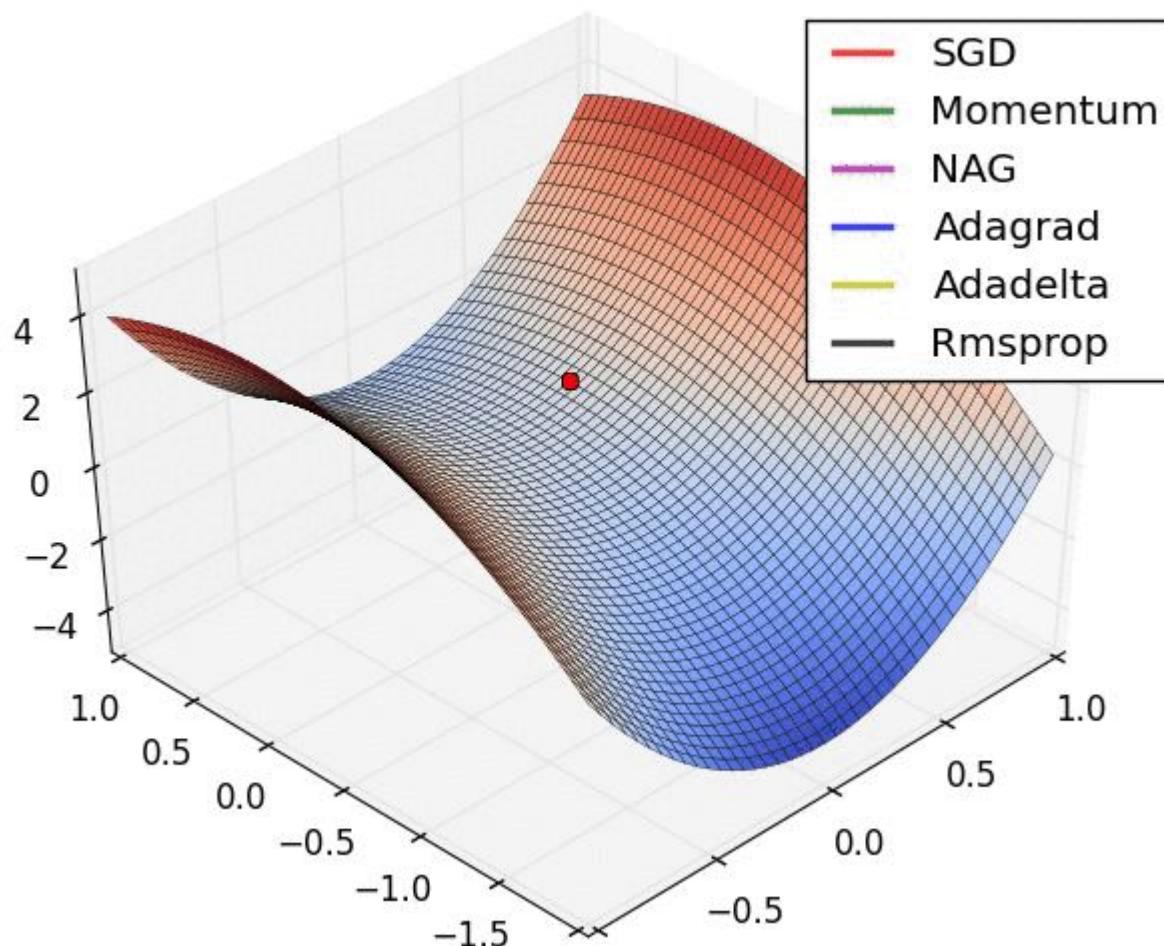
## 3. How to evaluate Gradients and Optimization

want  $\nabla_W L$









# Administrative

- First assignment will come out at the end of week at worst
- !!! Отзывы о лекции [goo.gl/gMeYNL](http://goo.gl/gMeYNL) о семинаре [goo.gl/5hIPD0](http://goo.gl/5hIPD0) !!!

# Next Time

- Theano
- Modern Conv Arch
- Visualizing, style-transfer, detection, AE

# Good courses/books

- [cs231n.stanford.edu](http://cs231n.stanford.edu), <https:// goo.gl/75Zi5m>
- [udacity.com/course/deep-learning-ud730](http://udacity.com/course/deep-learning-ud730)
- [deeplearningbook.org](http://deeplearningbook.org)