for Image Understanding:

Deep Learning with Convolutional Neural Nets

@graphific

Roelof Pieters

PhD candidate at KTH & Data Science consultant at Graph Technologies

roelof@graph-technologies.com

PyData

London 2015

What is Deep Learning?

A Definition

"Deep learning is a set of algorithms in machine learning that attempt to learn in multiple levels, corresponding to different levels of abstraction." (much debated definition)

Deep learning is:

- A host of statistical machine learning techniques
- Enables the automatic learning of feature hierarchies
- Generally based on artificial neural networks

Old vs new school?

Manually designed features are often over-specified, incomplete and take a long time to design and validate

Learned features are easy to adapt, fast to learn

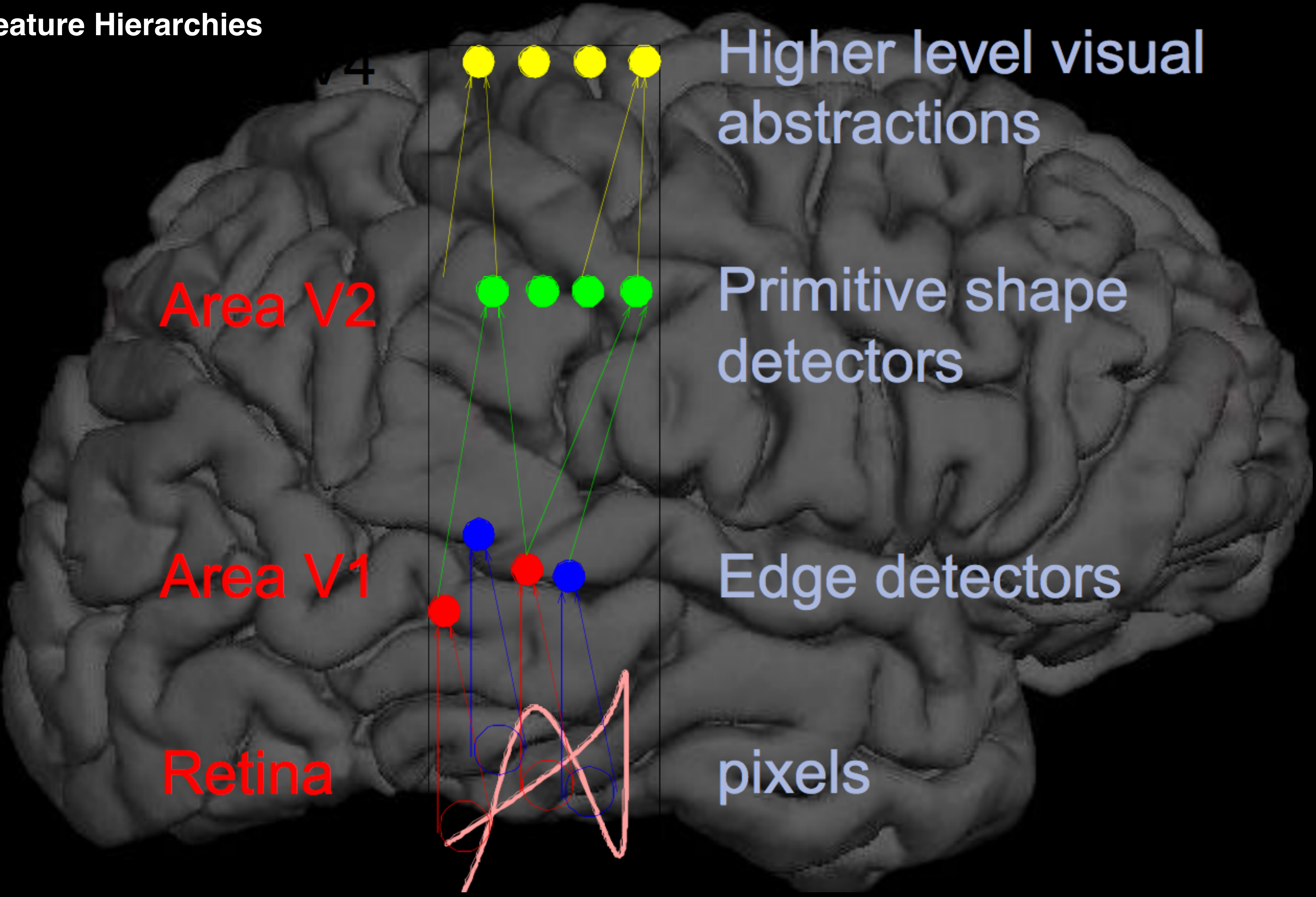
Deep learning provides a very flexible, (possibly?) universal, learnable framework for **representing** world, visual and linguistic information.

Deep learning can learn **unsupervised** (from raw text/audio/images/whatever content) and **supervised** (with specific labels like positive/negative)

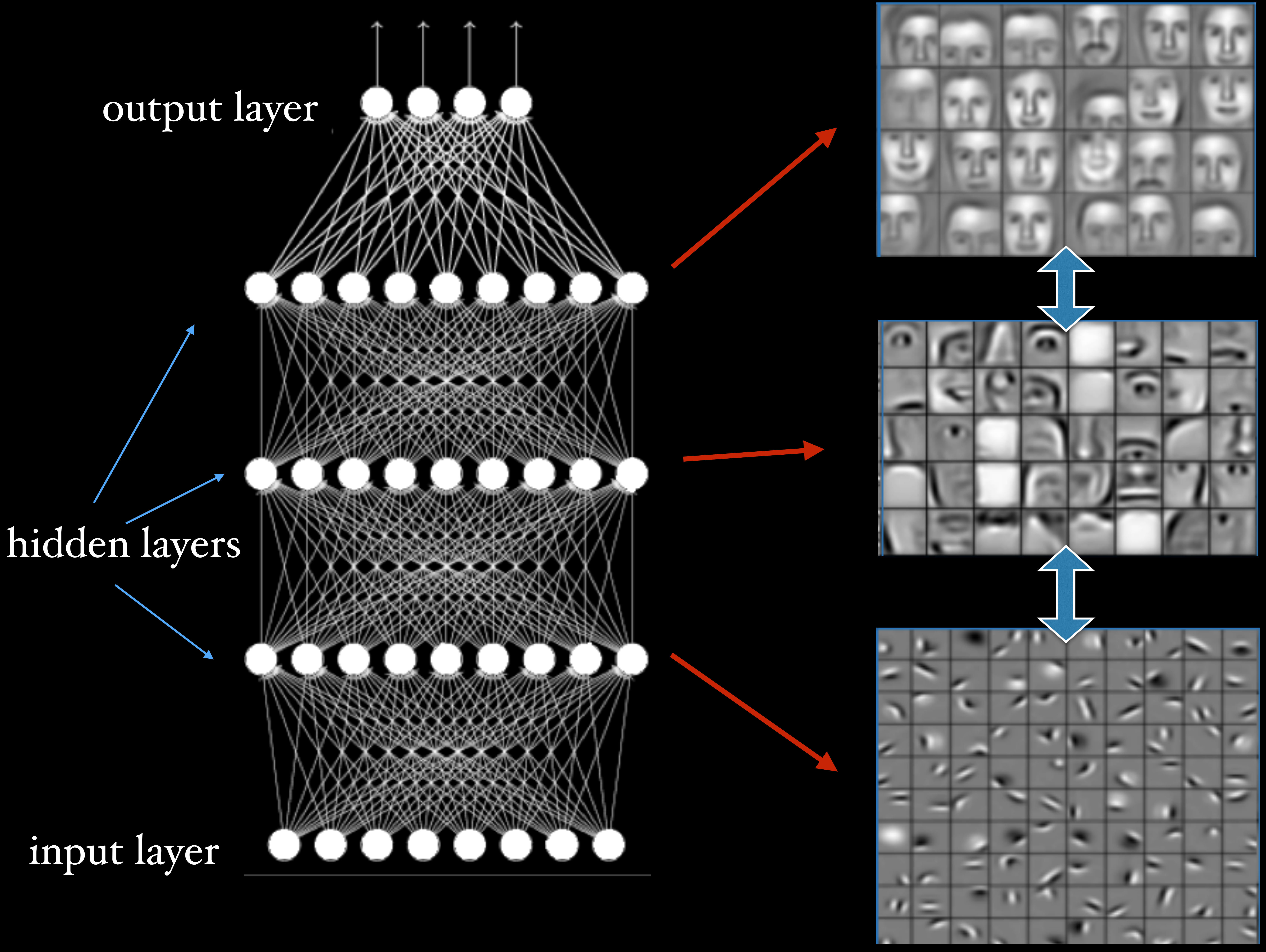
Summary by Richard Socher.

No More Handcrafted Features !

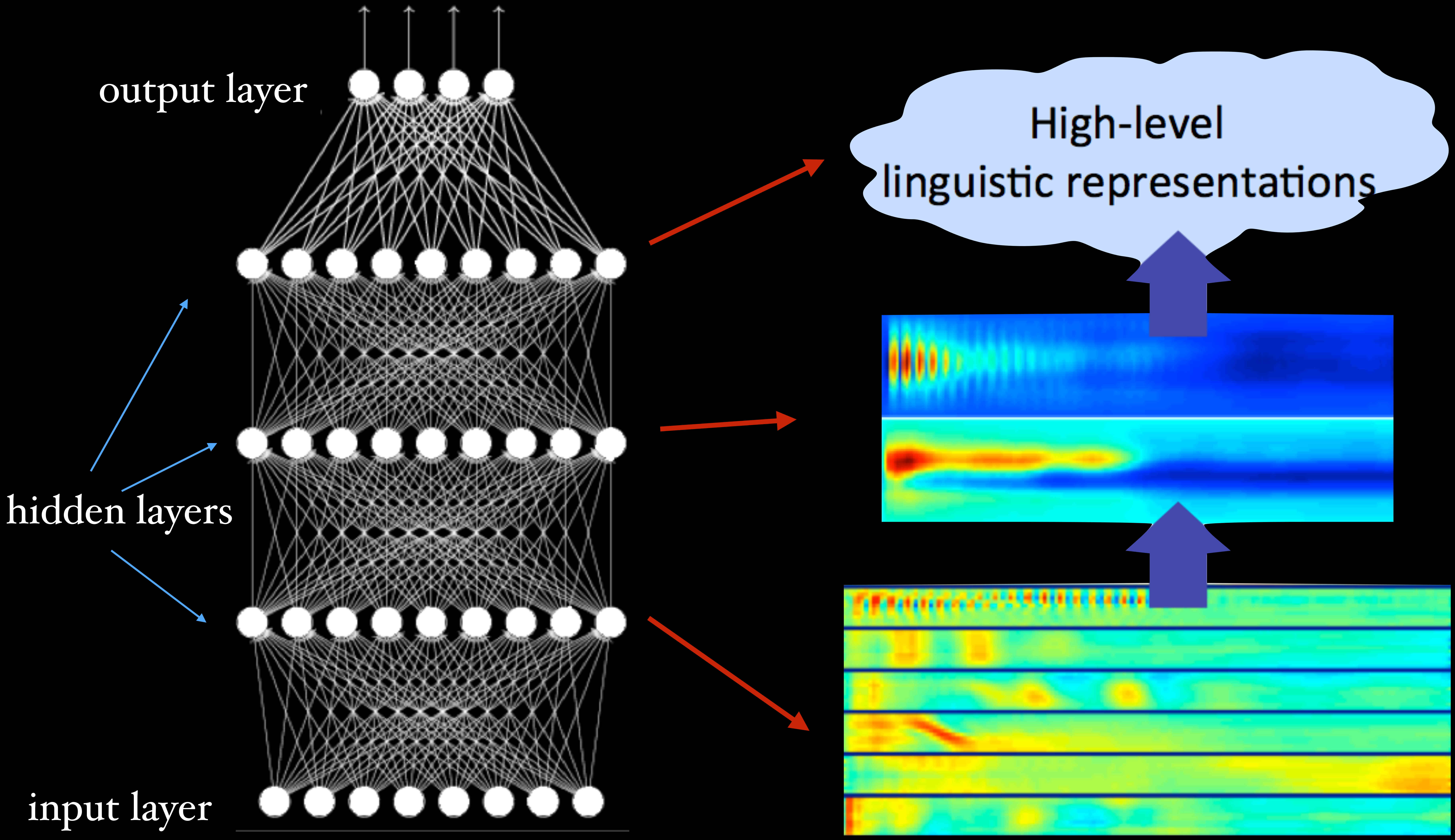
“Brain”-like: Feature Hierarchies



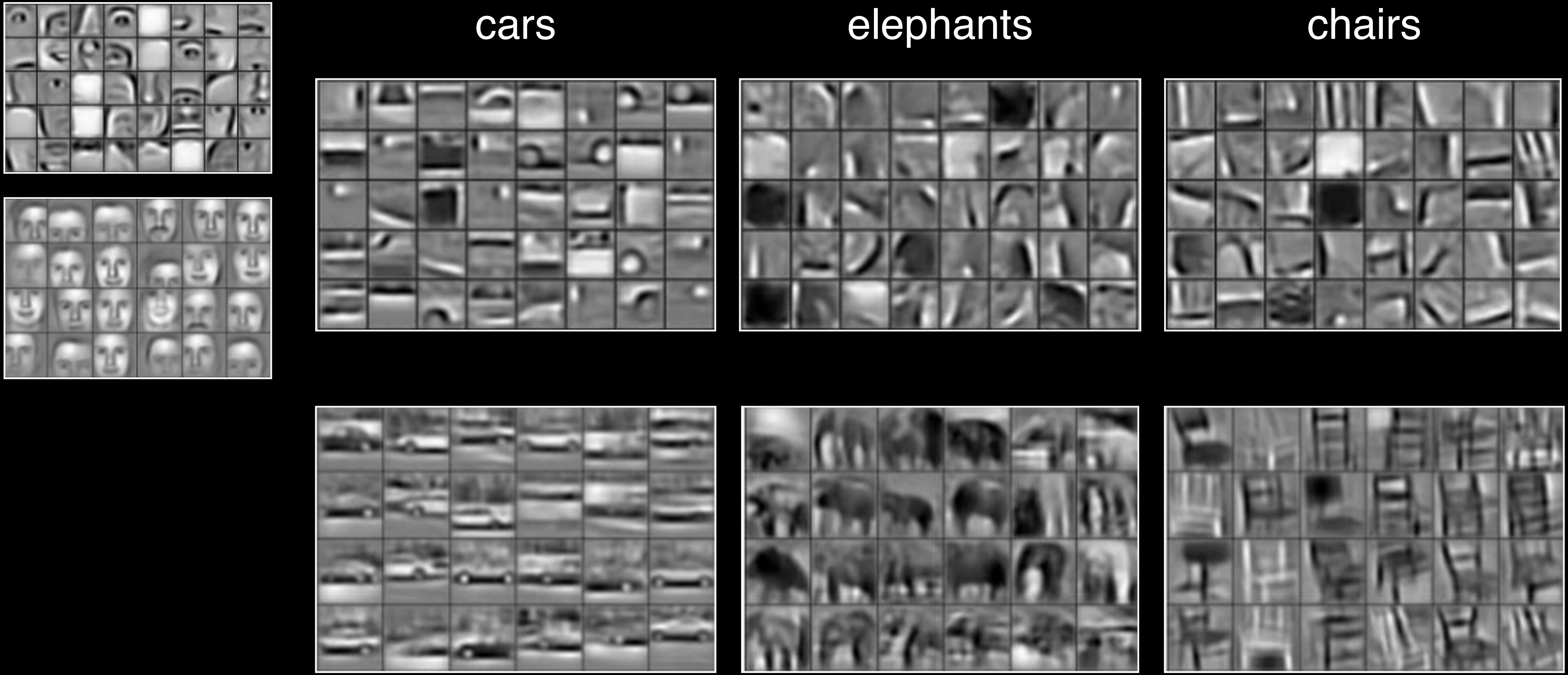
Feature Hierarchies: Vision



Feature Hierarchies: Audio



Feature Hierarchies: And so on…



History: audio recognition

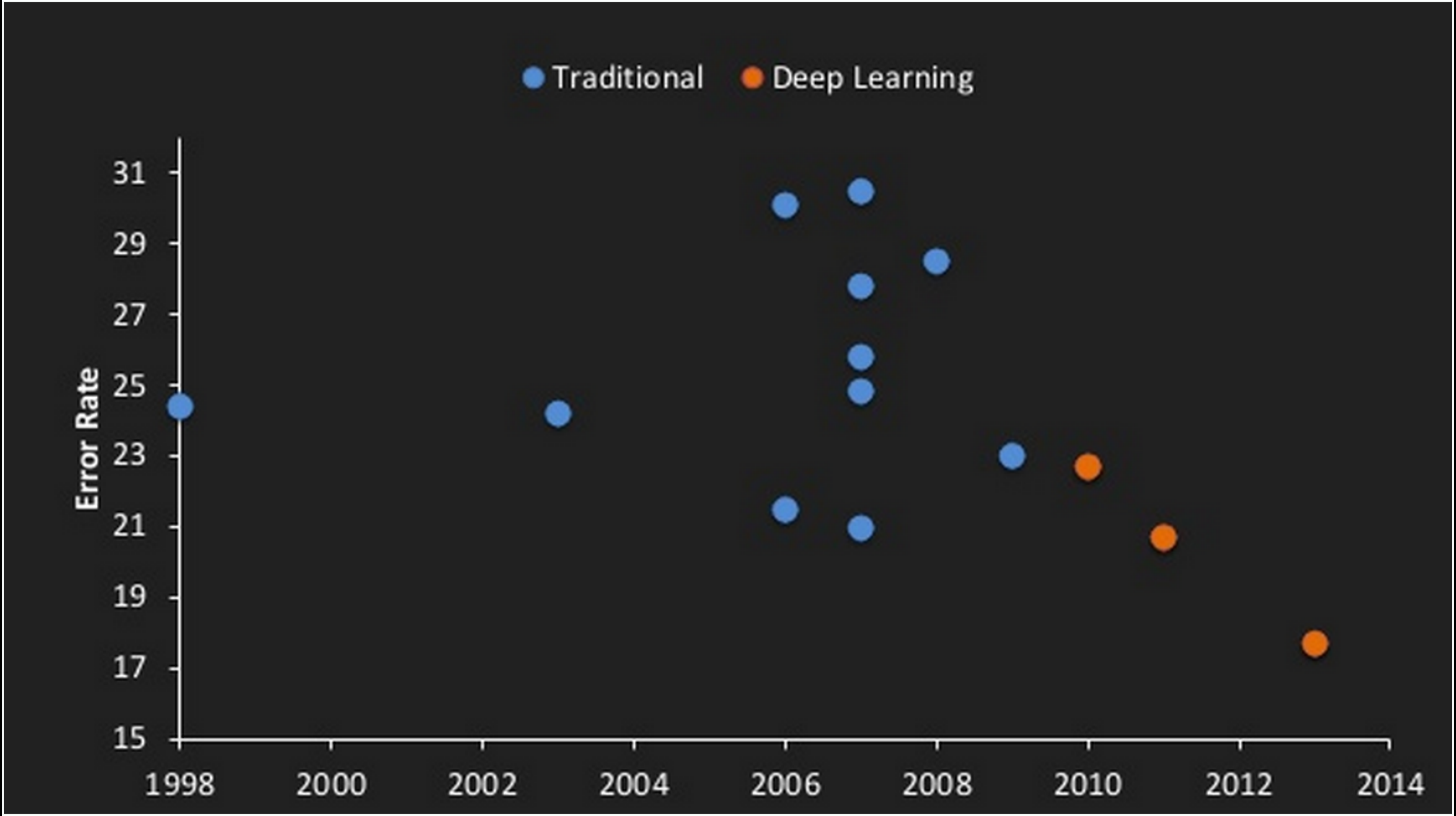


chart by Clarifai

History: image recognition

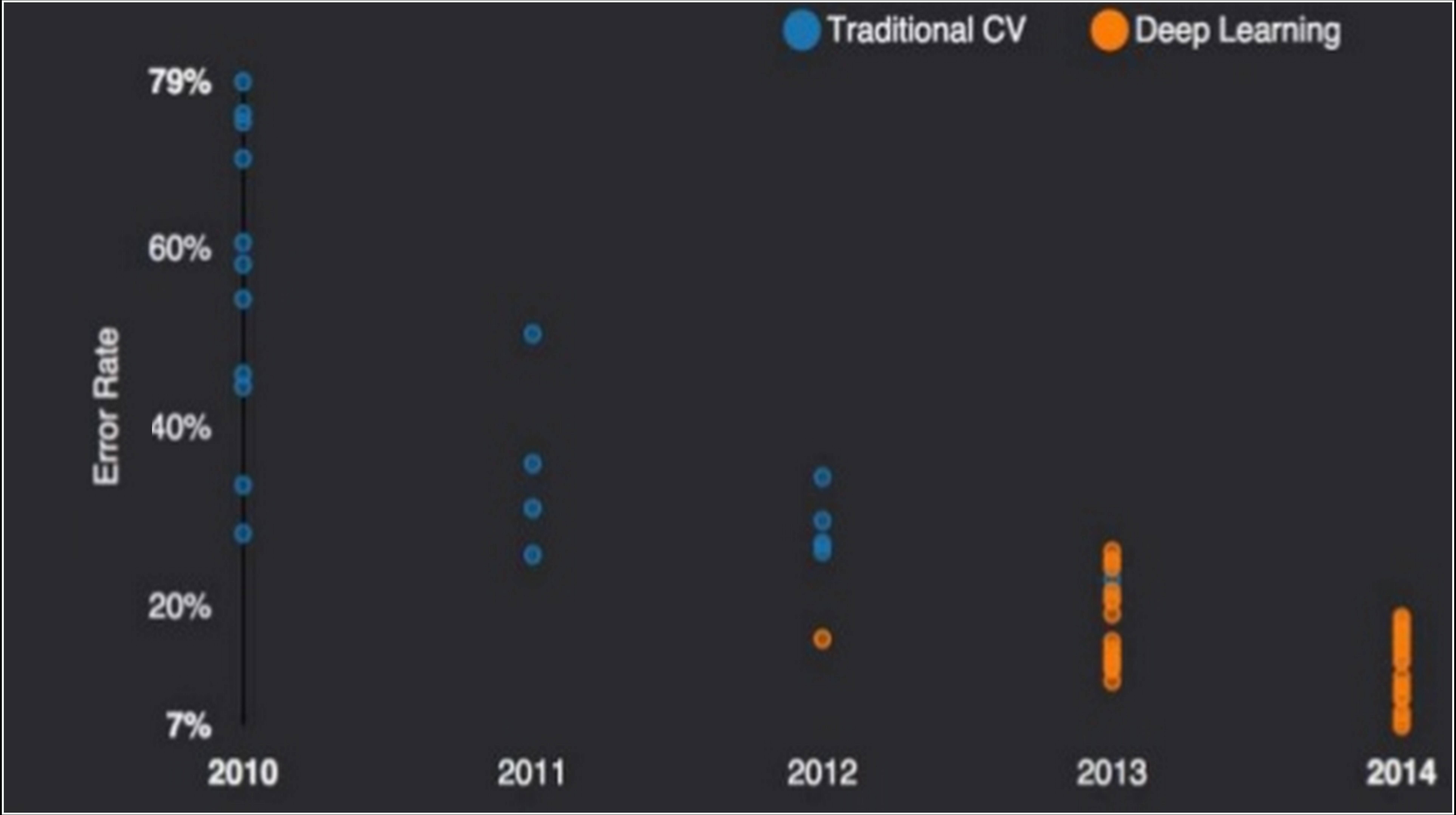
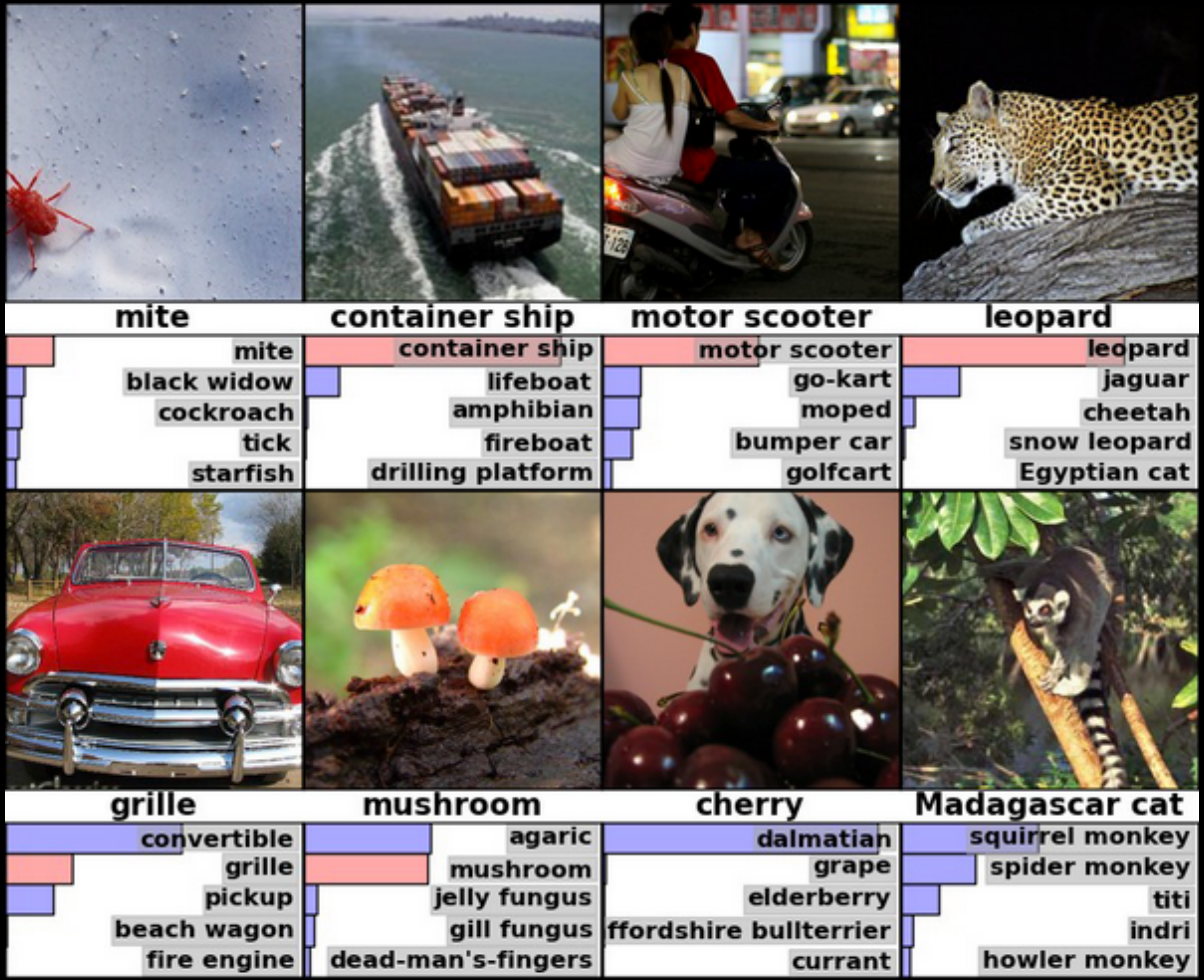


chart by Clarifai

History: image recognition



Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks, ILSVRC2010

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

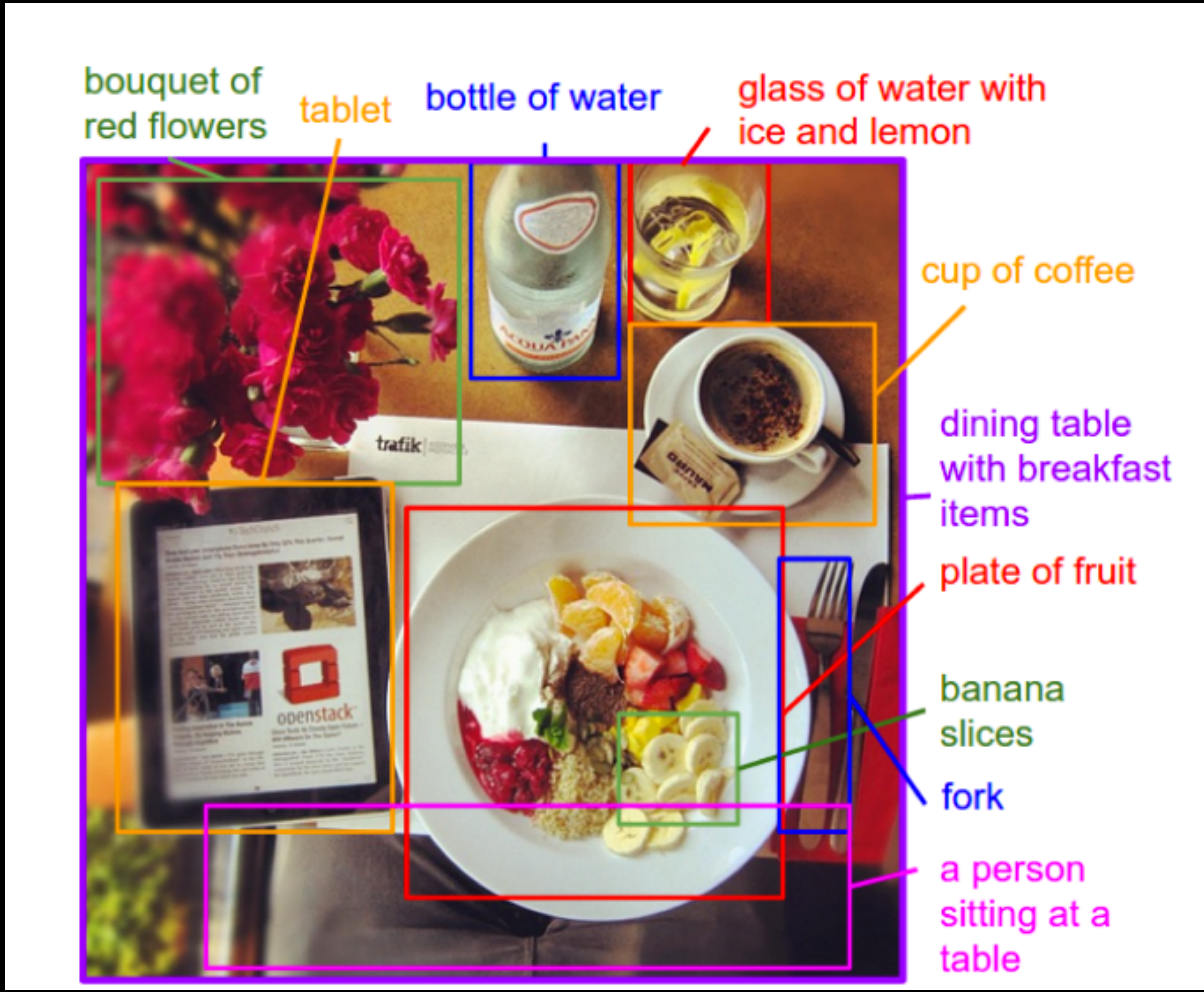
Daan Wierstra **Martin Riedmiller**

DeepMind Technologies

Abstract

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

Image-Text: Joint Visual Semantic embeddings



Karpathy, A., Fei Fei, L. (2015)
Deep Visual-Semantic Alignments for Generating Image Descriptions

Video Scene Detection

Beyond Short Snippets: Deep Networks for Video Classification

Joe Yue-Hei Ng[1]
yhng@umiacs.umd.edu

Matthew Hausknecht[2]
mhauskn@cs.utexas.edu

Sudheendra Vijayanarasimhan[3]
svnaras@google.com

Oriol Vinyals[3]
vinyals@google.com

Rajat Monga[3]
rajatmonga@google.com

George Toderici[3]
gtoderici@google.com

[1]University of Maryland, College Park [2]University of Texas at Austin [3]Google, Inc.

Abstract

Convolutional neural networks (CNNs) have been extensively applied for image recognition problems giving state-of-the-art results on recognition, detection, segmentation and retrieval. In this work we propose and evaluate several deep neural network architectures to combine image information across a video over longer time periods than previously attempted. We propose two methods capable of handling full length videos. The first method explores various convolutional temporal feature pooling architectures, examining the various design choices which need to be made when adapting a CNN for this task. The second proposed method explicitly models the video as an ordered sequence of frames. For this purpose we employ a recurrent neural network that uses Long Short-Term Memory (LSTM) cells which are connected to the output of the underlying CNN.

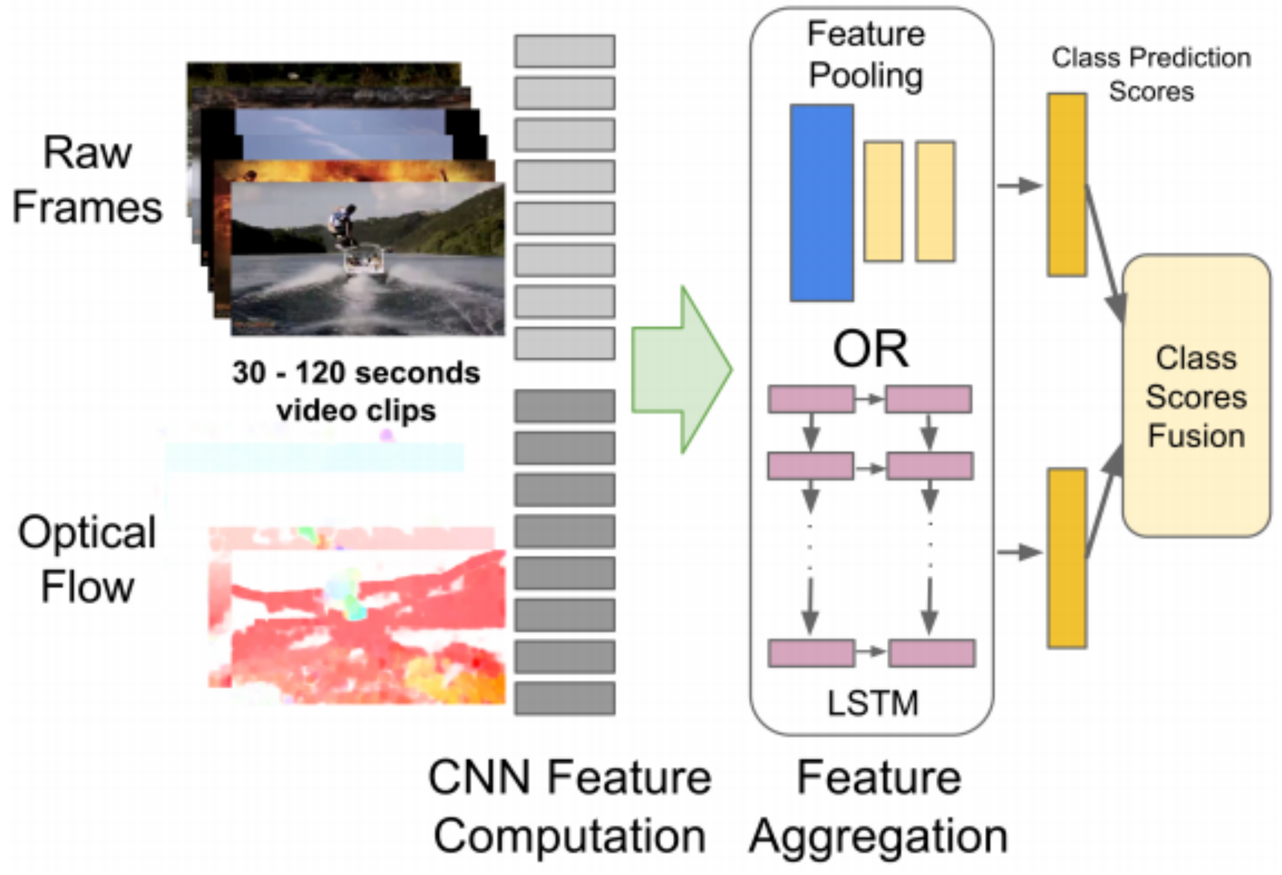


Figure 1: Overview of our approach.

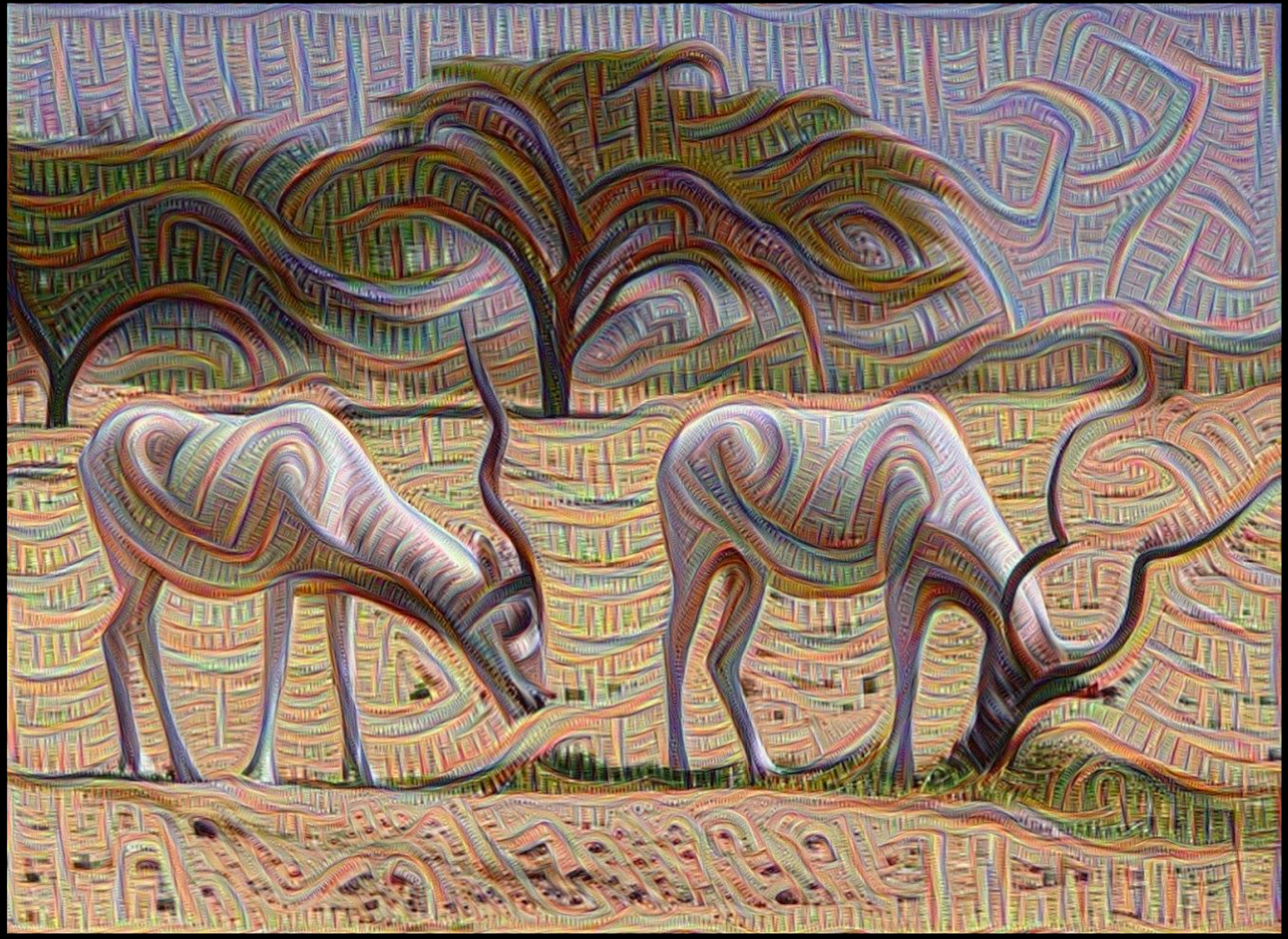
Inceptionism: Going Deeper into Neural Networks

Posted: Wednesday, June 17, 2015

g+1 4.9k Tweet 2,910 Like 26k

Posted by Alexander Mordvintsev, Software Engineer, Christopher Olah, Software Engineering Intern and Mike Tyka, Software Engineer

http://googleresearch.blogspot.co.uk/2015/06/inceptionism-going-deeper-into-neural.html

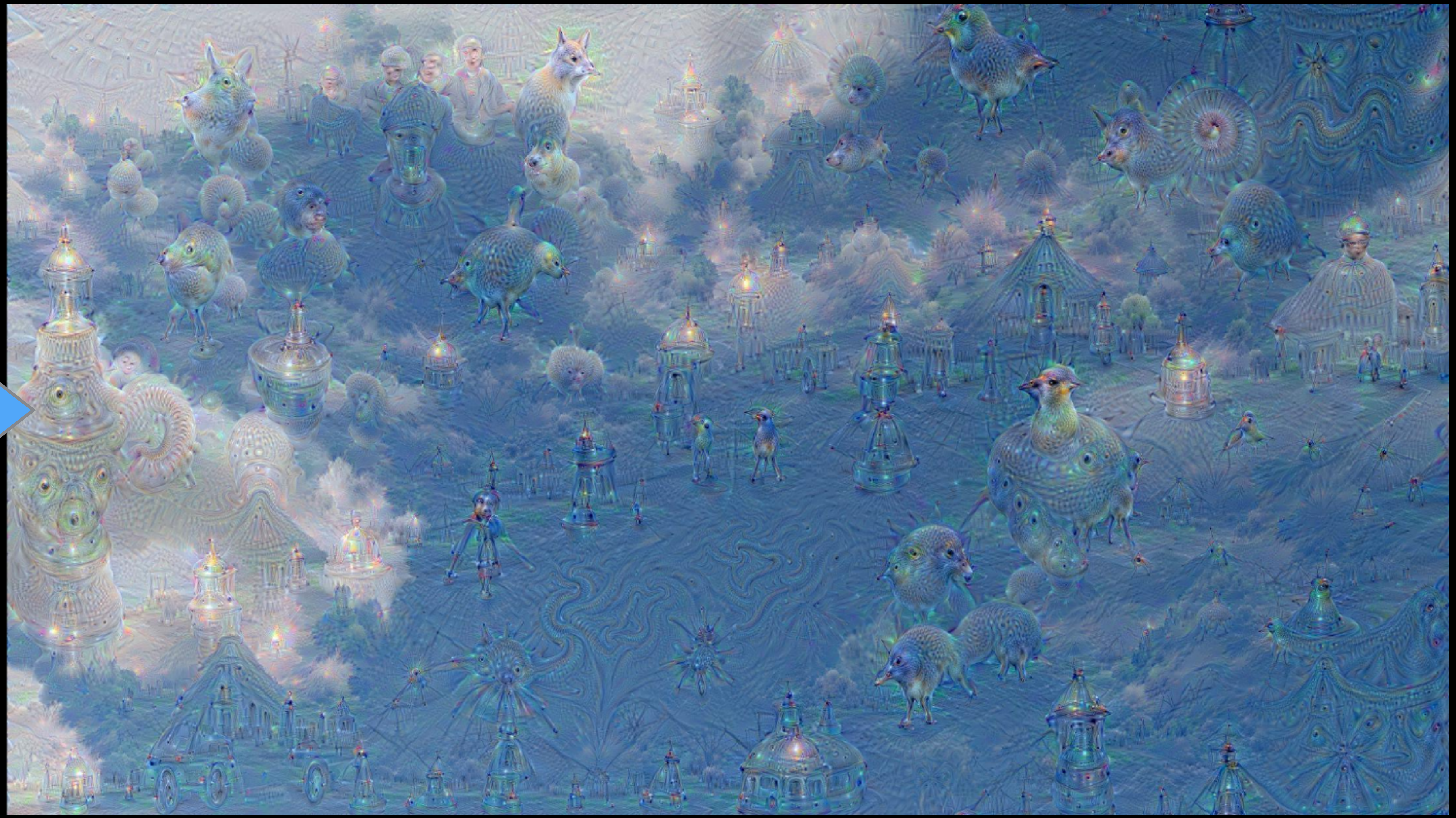


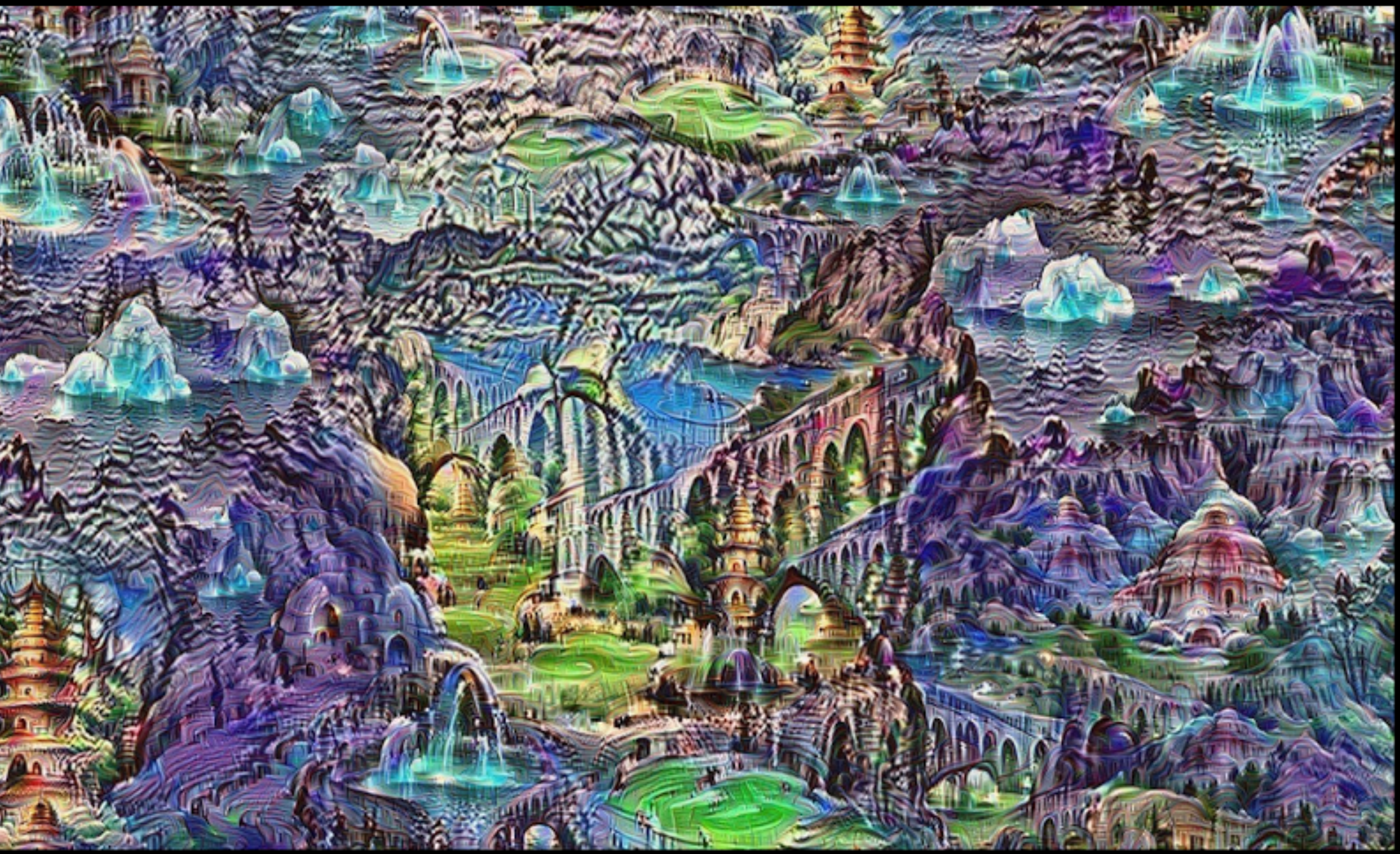
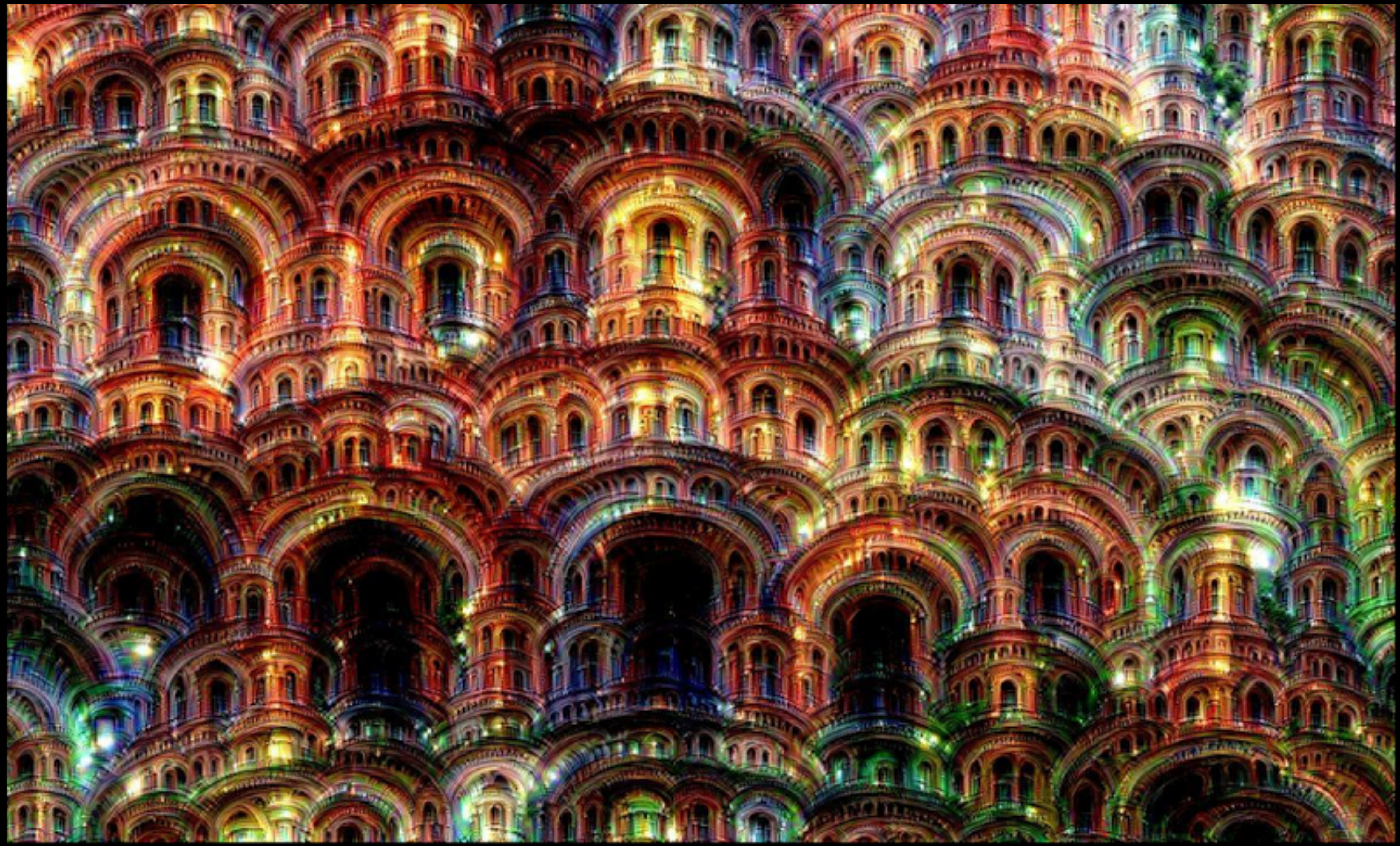
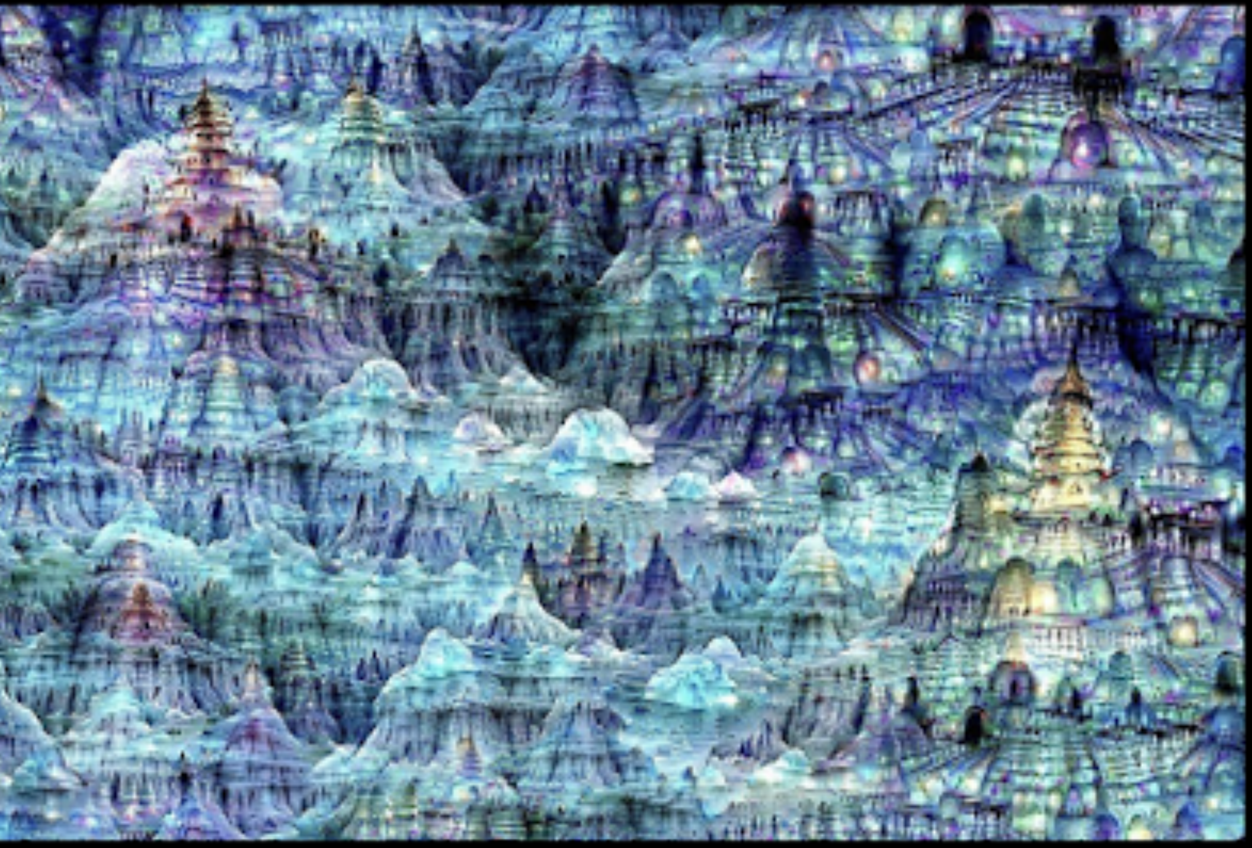
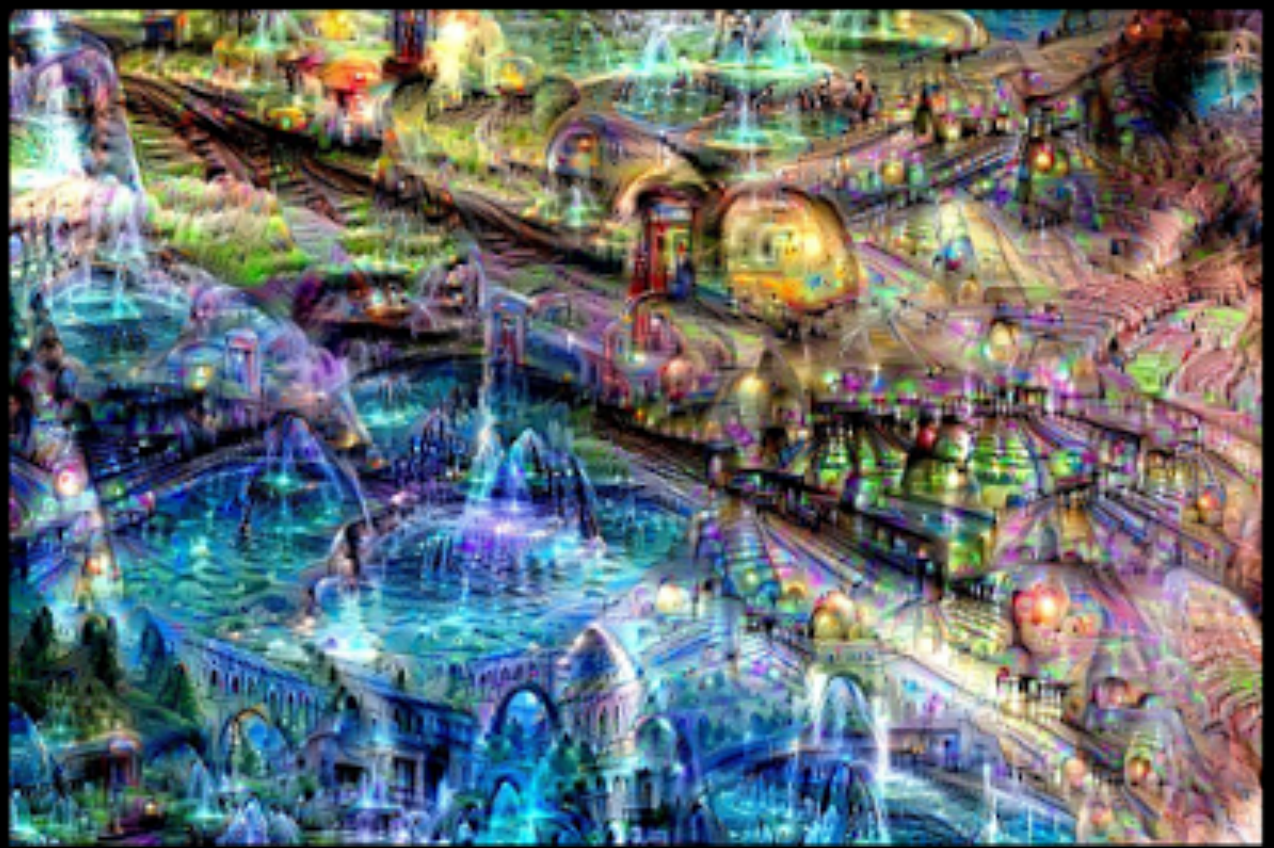
Inceptionism: Going Deeper into Neural Networks

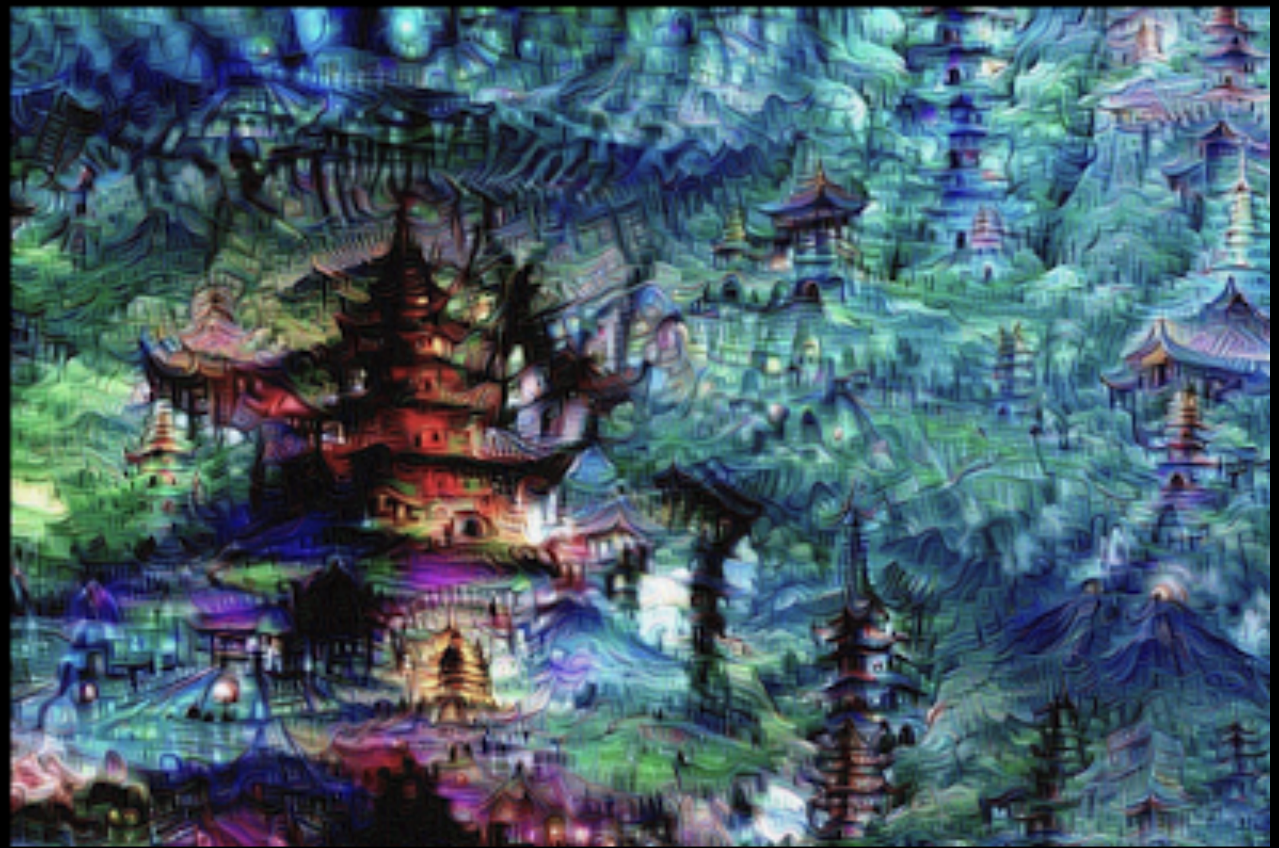
Posted: Wednesday, June 17, 2015

Posted by Alexander Mordvintsev, Software Engineer, Christopher Olah, Software Engineering Intern and Mike Tyka, Software Engineer

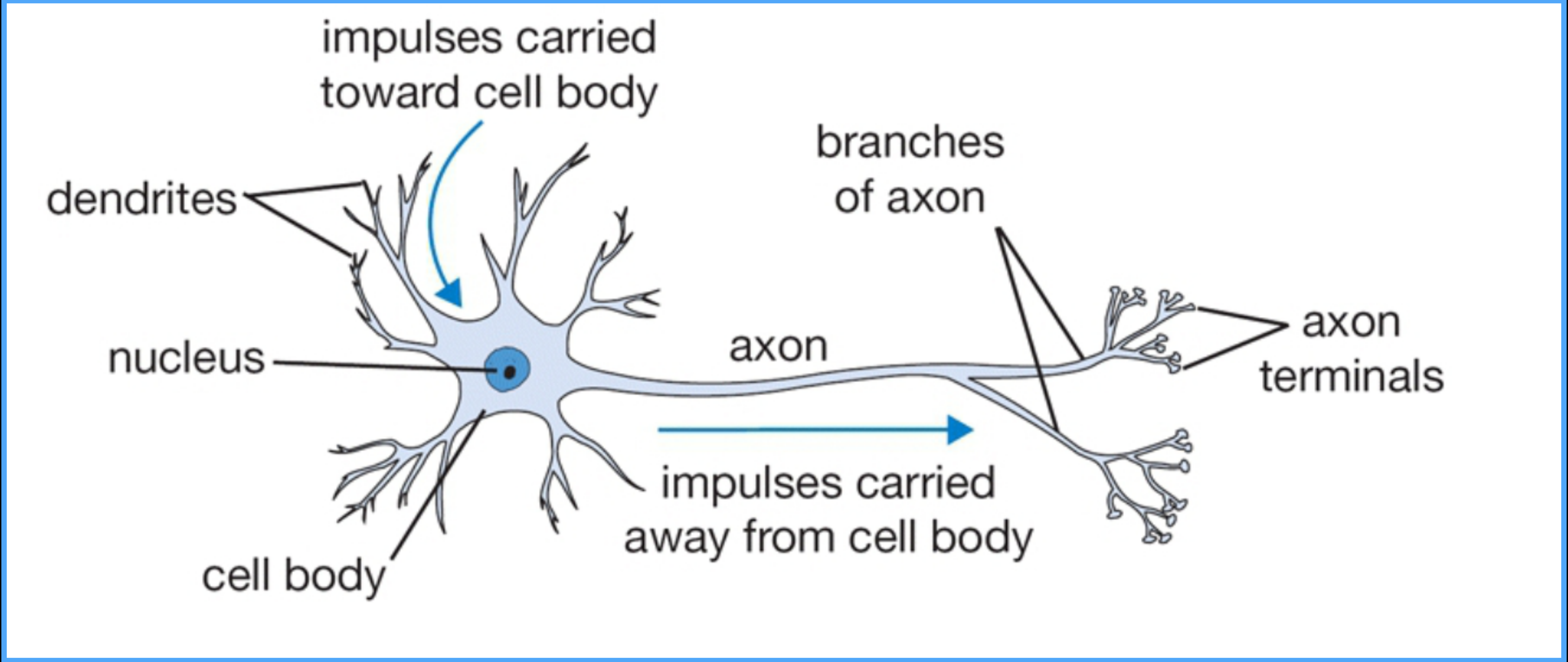
http://googleresearch.blogspot.co.uk/2015/06/inceptionism-going-deeper-into-neural.html

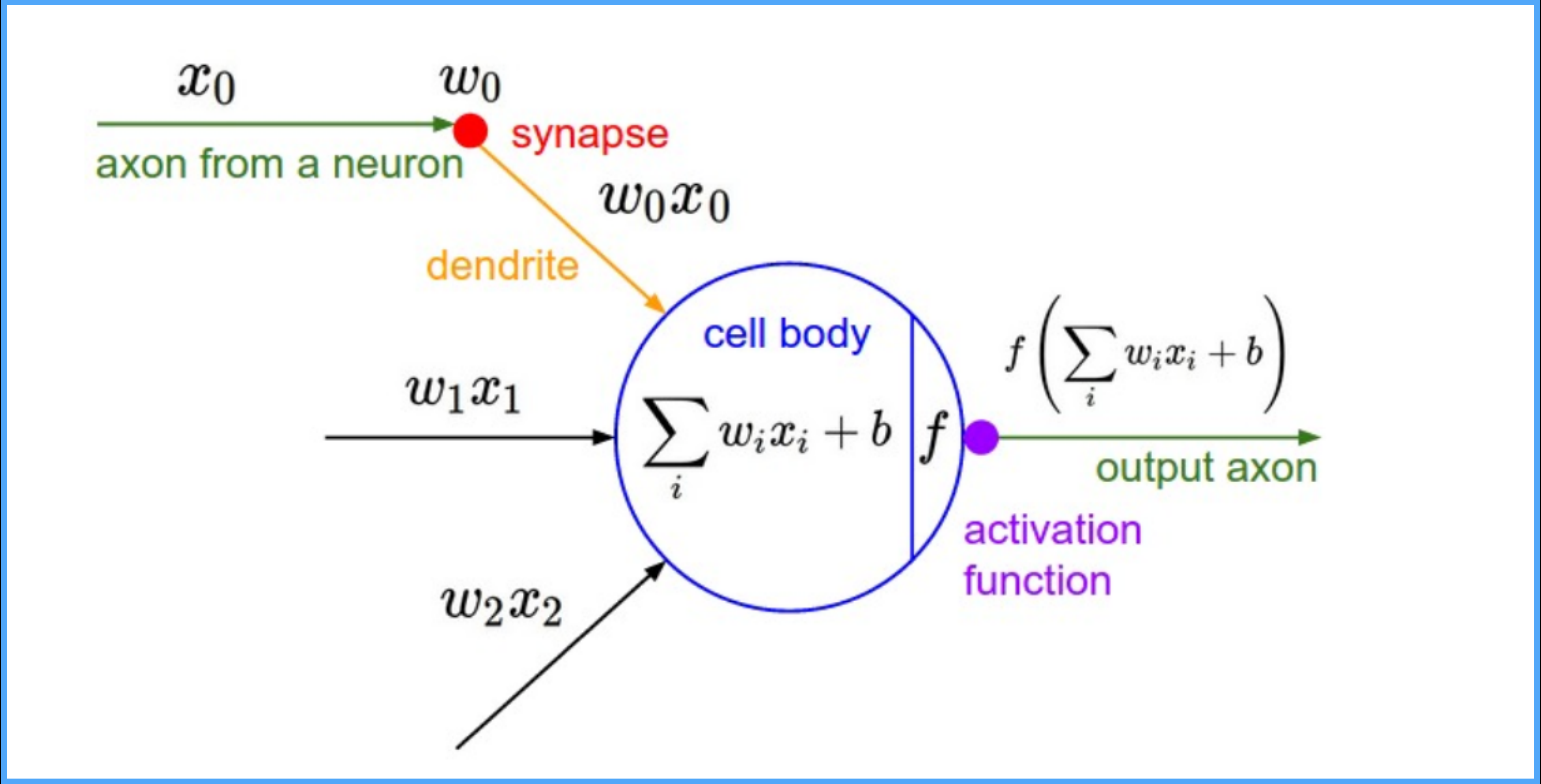




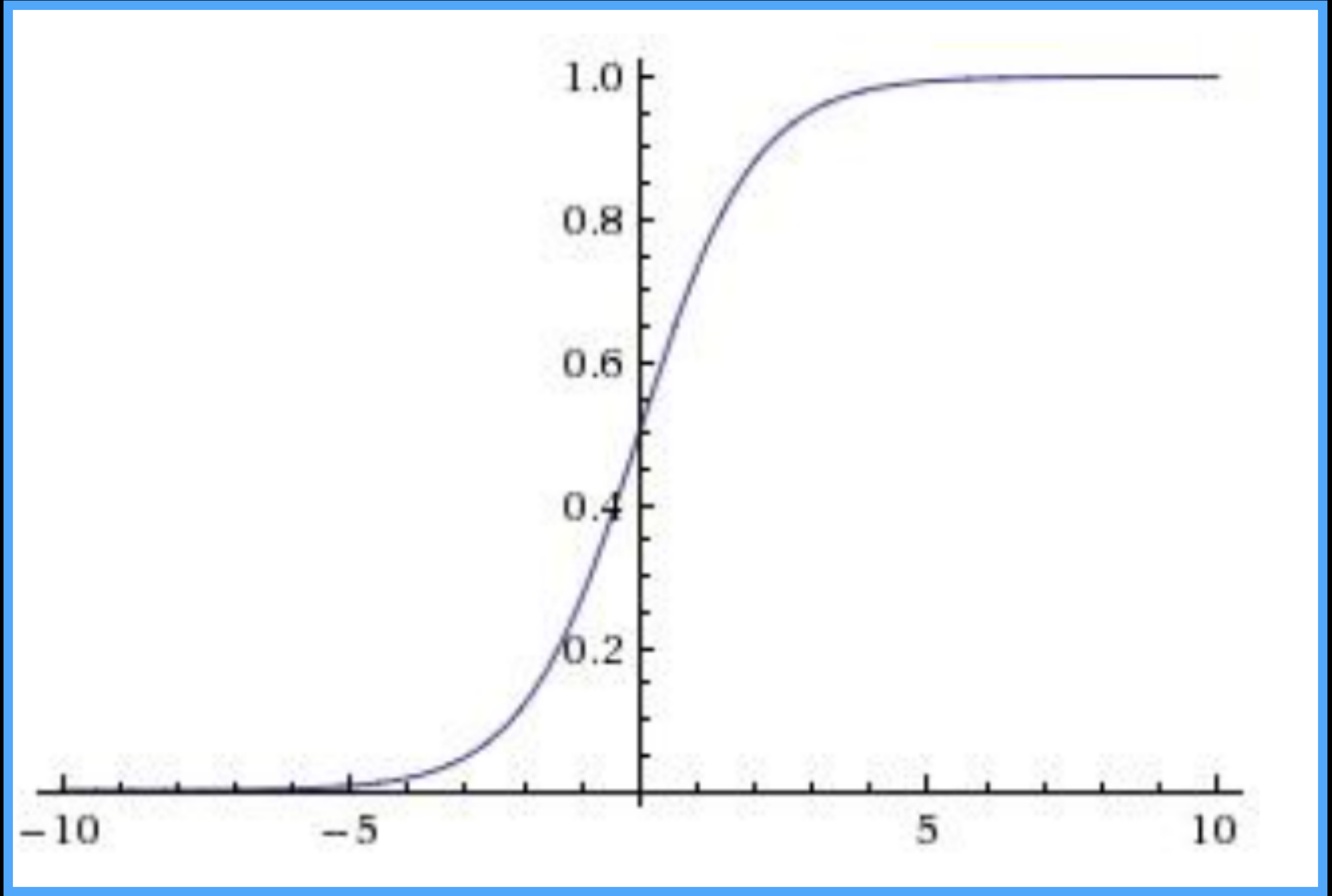
How does Deep Learning work?


impulses carried
toward cell body
dendrites
branches
of axon
axon
terminals
nucleus
axon
impulses carried
away from cell body
cell body


x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$f\left(\sum_i w_i x_i + b\right)$
$w_1 x_1$
$\sum_i w_i x_i + b$
f
output axon
activation function
$w_2 x_2$

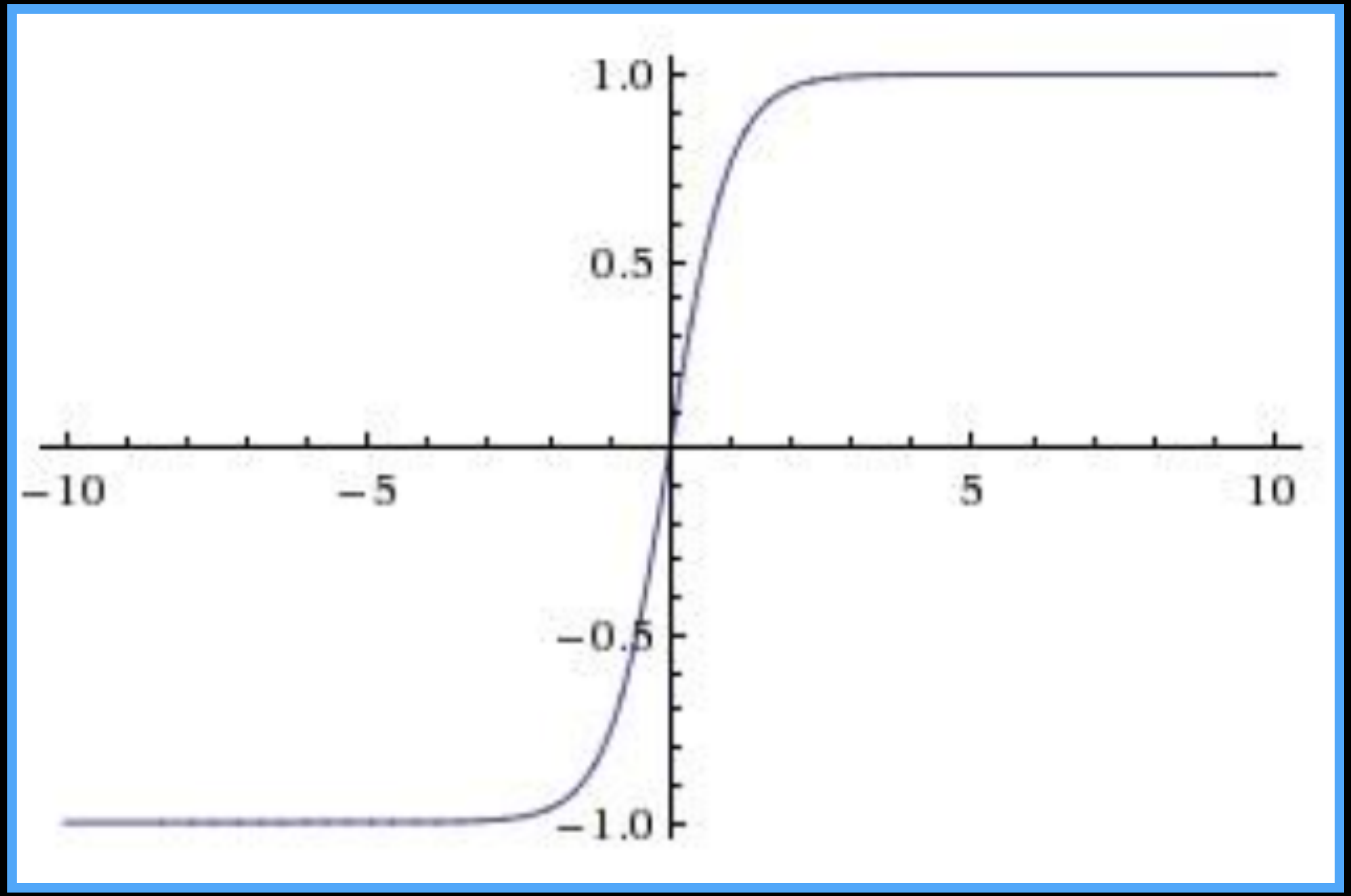
Activation Functions

Sigmoid



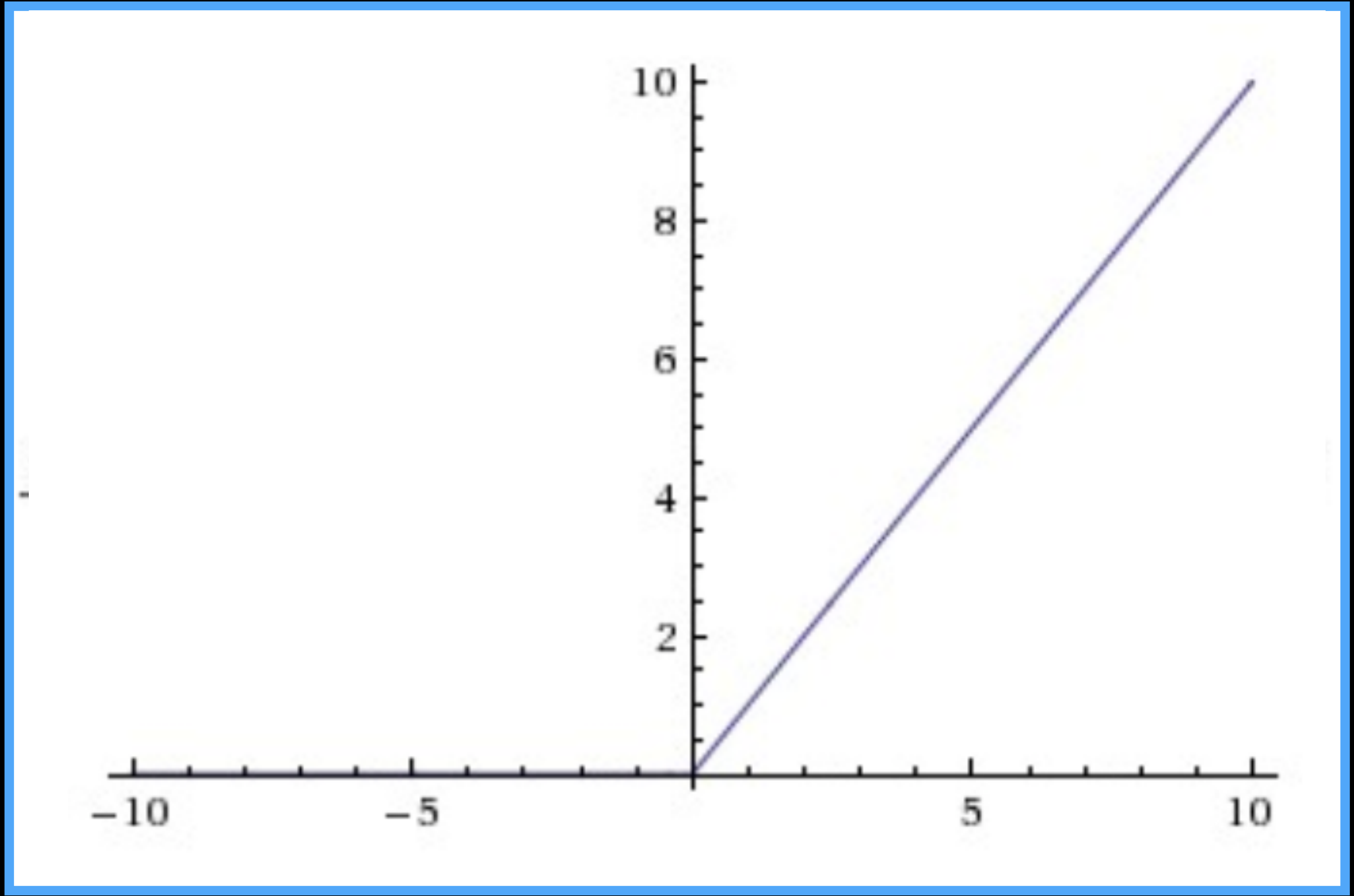
$$f(x) = \frac{1}{1+exp(-x)}$$

Tanh



$$f(x) = tanh(x)$$

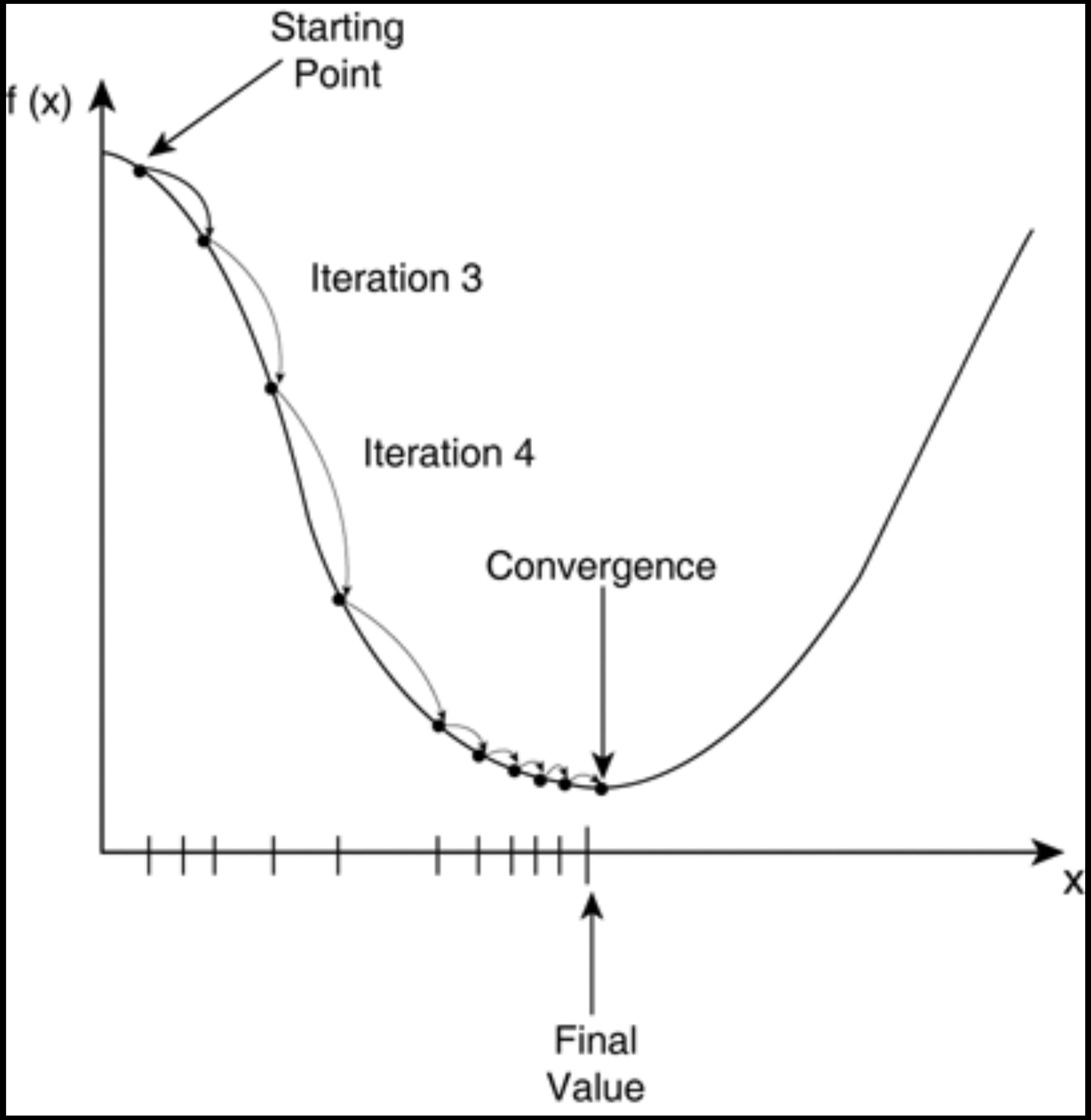
ReLU

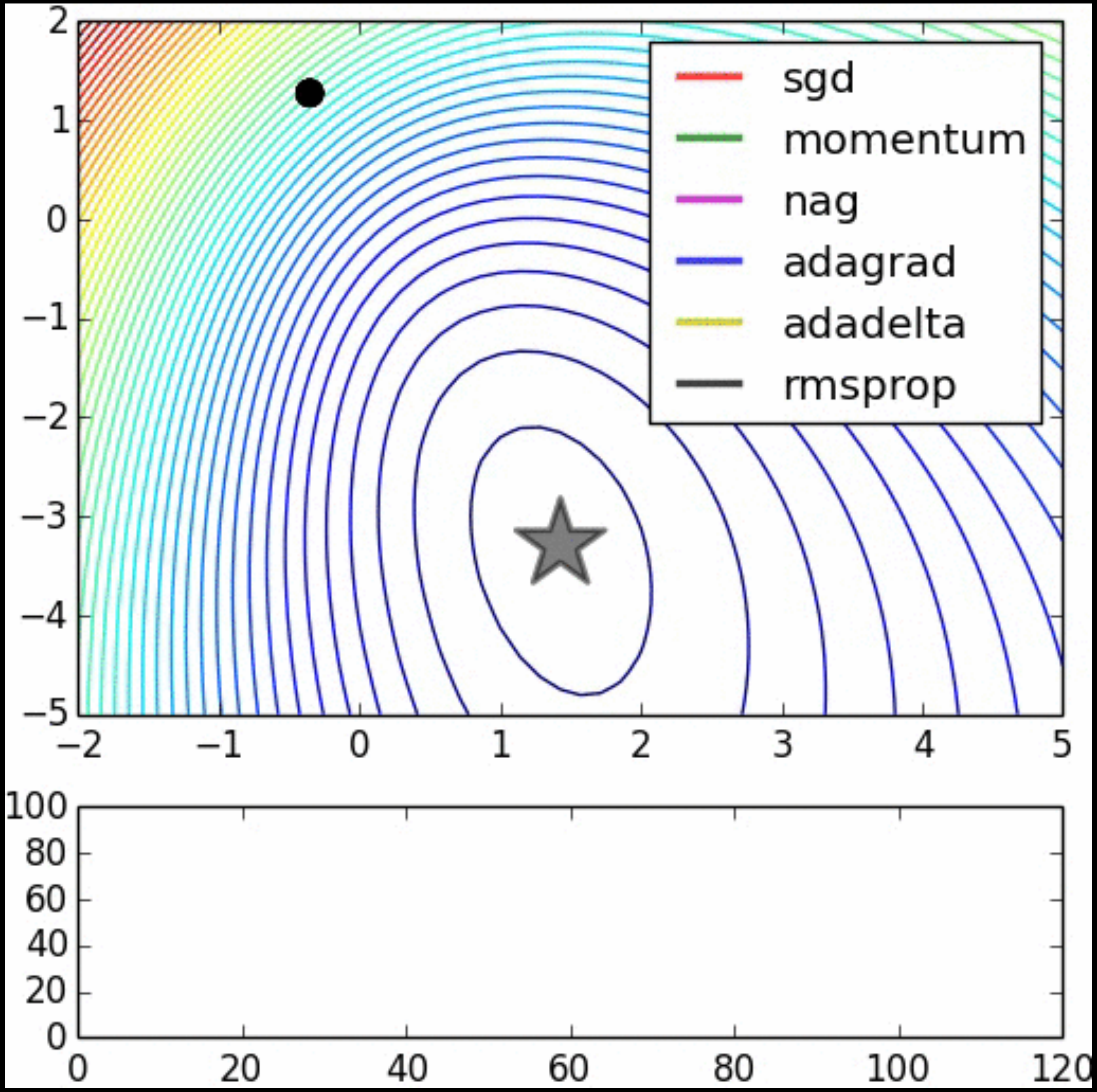


$$f(x) = \sum_{i=1}^{\mathrm{inf}} \sigma(x - i + 0.5)$$

often approximated by just $max(0, x\ )$

optimization strategies





So what’s there for **Python?**

python has a wide range of deep learning-related libraries available

and of course: scikit learn

High level

Lasagne (theano-extension, models in python code, theano not hidden)

Keras (theano-wrapper, models in python code, abstracts theano away)

Pylearn2 (wrapper for theano, yaml, experiment-oriented)

Caffe (computer-vision oriented DL framework, model-zoo, prototxt model definitions) pythonification ongoing!

theano (efficient gpu-powered math)

Low level

python has a wide range of deep learning-related libraries available

and of course: scikit learn

High level

Lasagne lasagne.readthedocs.org/en/latest

Keras keras.io

Pylearn2 deeplearning.net/software/pylearn2

Caffe caffe.berkeleyvision.org

Low level

theano deeplearning.net/software/theano

python has a wide range of deep learning-related libraries available

and of course: scikit learn

we will use lasagne in our examples

High level

Lasagne lasagne.readthedocs.org/en/latest

Keras keras.io

Pylearn2 deeplearning.net/software/pylearn2

Caffe caffe.berkeleyvision.org

theano deeplearning.net/software/theano

Low level

Doing Deep Learning?

Training a (deep) Neural Network

1. Preprocess the data
2. Choose architecture
3. Train
4. Optimize/Regularize
5. Tips/Tricks

1. Preprocess the data
2. Choose architecture
3. Train
4. Optimize/Regularize
5. Tips/Tricks

1. Preprocess the data

- Mean subtraction
- Normalization
- PCA and Whitening

1. Preprocess the data: Normalization

- Mean subtraction

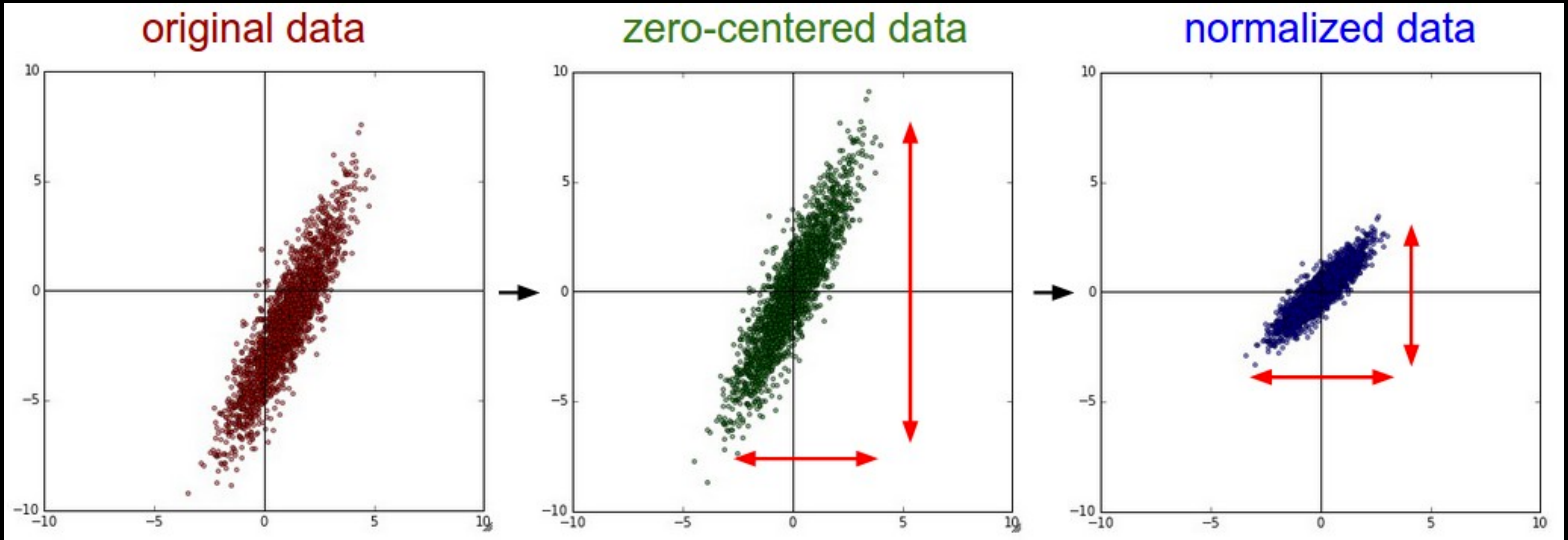
```
X -= np.mean(X)
```

- Normalization

```
X /= np.std(X)
```

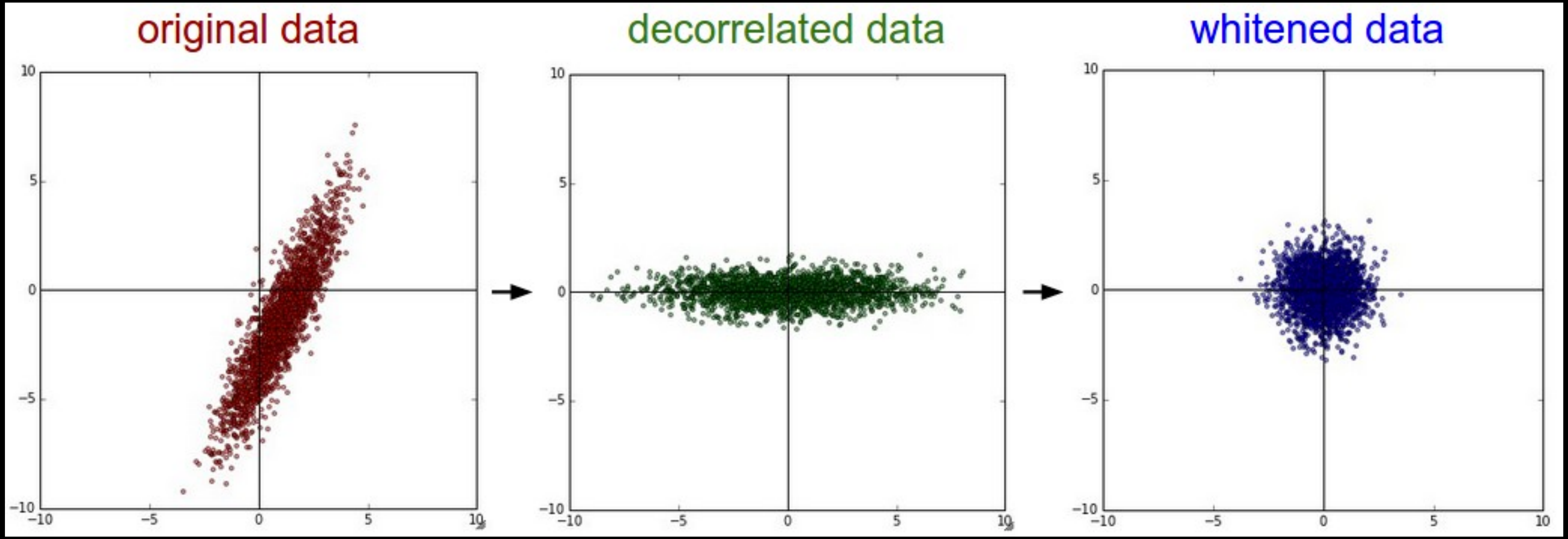
- PCA and Whitening

(mean image visualised of cifar-10)



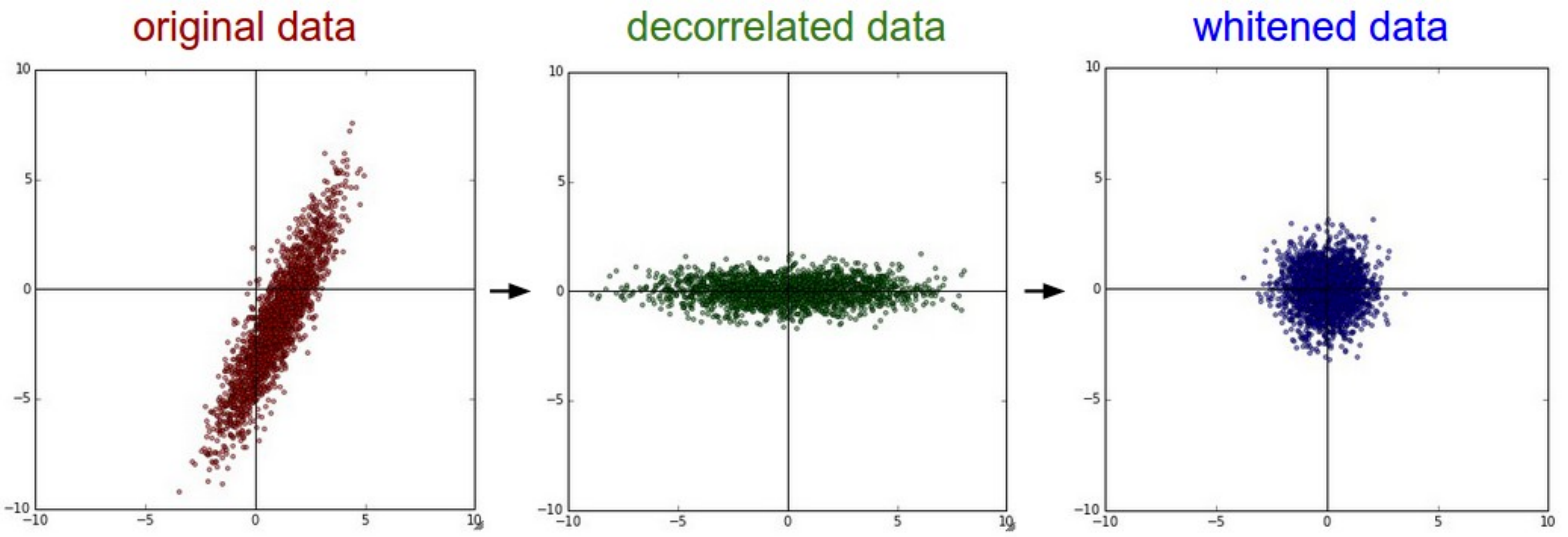
1. Preprocess the data: PCA & Whitening

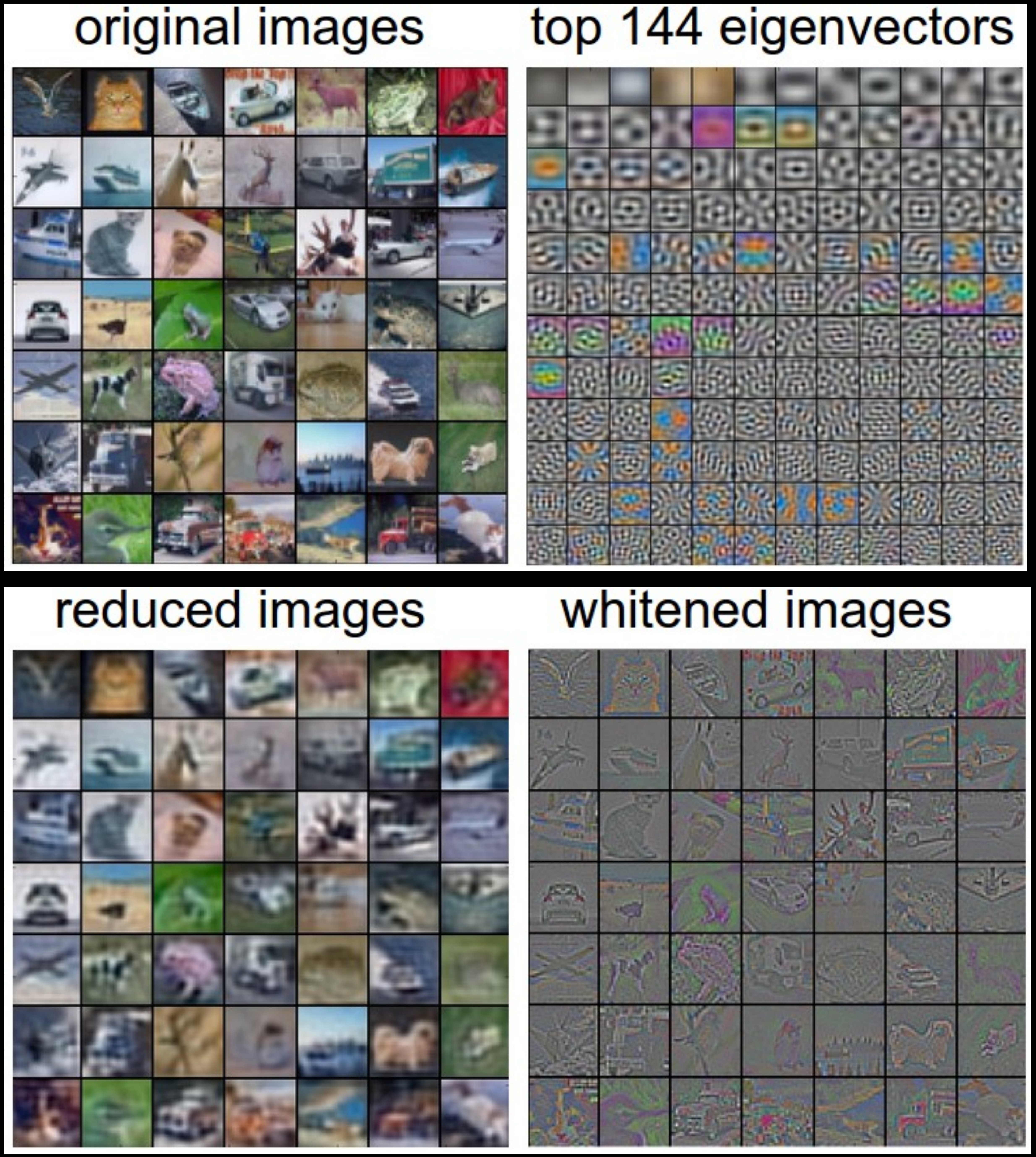
- Mean subtraction
- Normalization
- PCA and Whitening



1. Preprocess the data: PCA & Whitening

```
# Assume input data matrix X of size [N x D]
X -= np.mean(X, axis = 0) # zero-center the data (important)
cov = np.dot(X.T, X) / X.shape[0] # get the data covariance matrix
U,S,V = np.linalg.svd(cov) # compute the SVD factorization of the data
covariance matrix
Xrot = np.dot(X, U) # decorrelate the data
Xrot_reduced = np.dot(X, U[:,:100]) # Xrot_reduced becomes [N x 100]
# whiten the data:
# divide by the eigenvalues (which are square roots of the singular values)
Xwhite = Xrot / np.sqrt(S + 1e-5)
```




original images
top 144 eigenvectors
reduced images
whitened images

1. Preprocess the data, the right way

Warning:

- compute preprocessing statistics on training data
- apply on all (training / validation / test) data

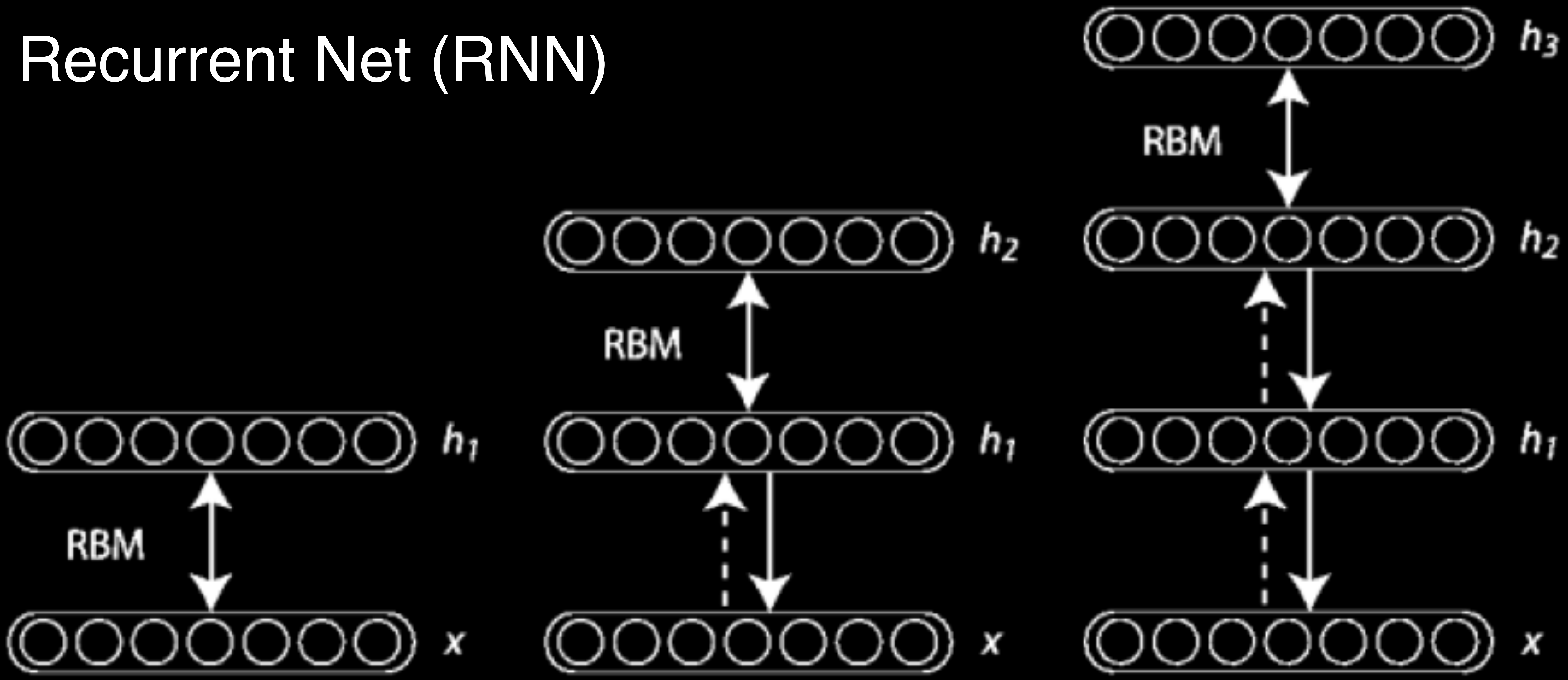
Training a (deep) Neural Network

1. Preprocess the data
2. Choose architecture
3. Train
4. Optimize/Regularize
5. Tips/Tricks

2. Choosing the right architecture

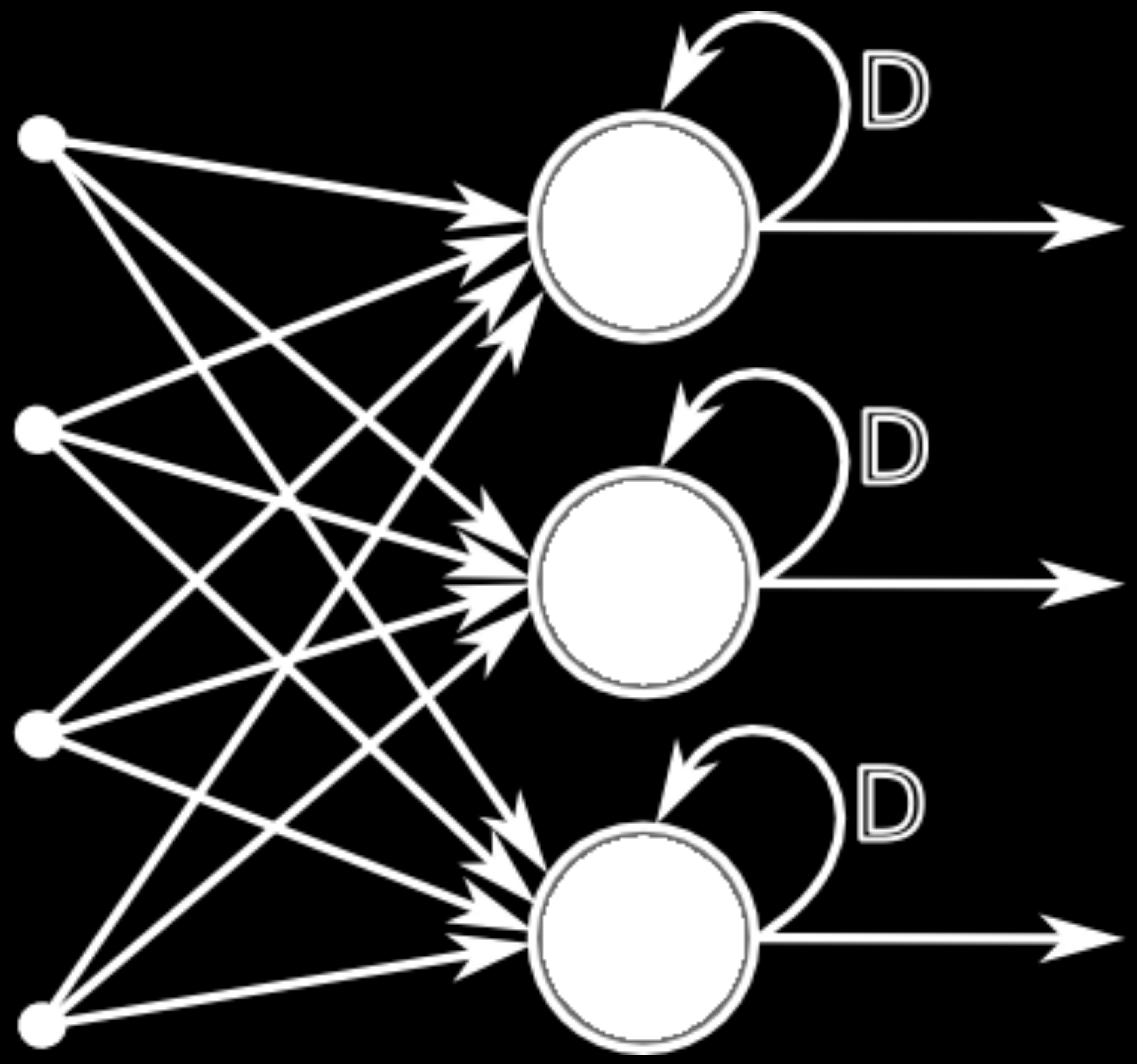
- Deep Belief Network (DBN)
- Convolutional Net (CNN)
- Recurrent Net (RNN)

- Deep Belief Network (DBN)
- Convolutional Net (CNN)
- Recurrent Net (RNN)



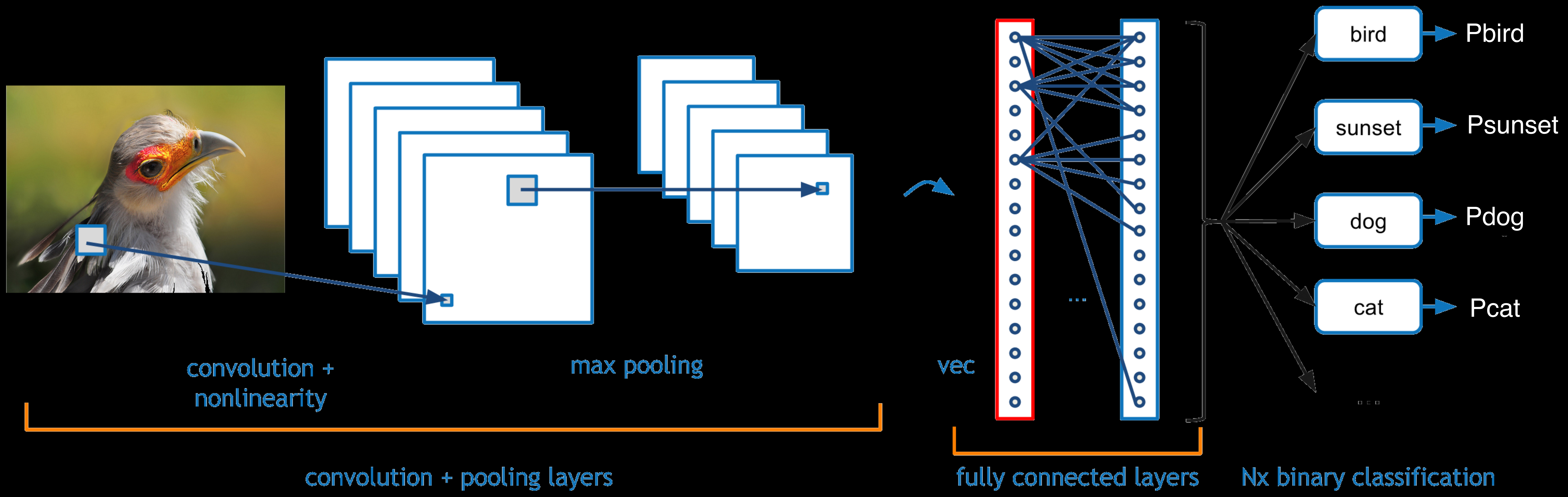


- Deep Belief Network (DBN)
- Convolutional Net (CNN)
- Recurrent Net (RNN)

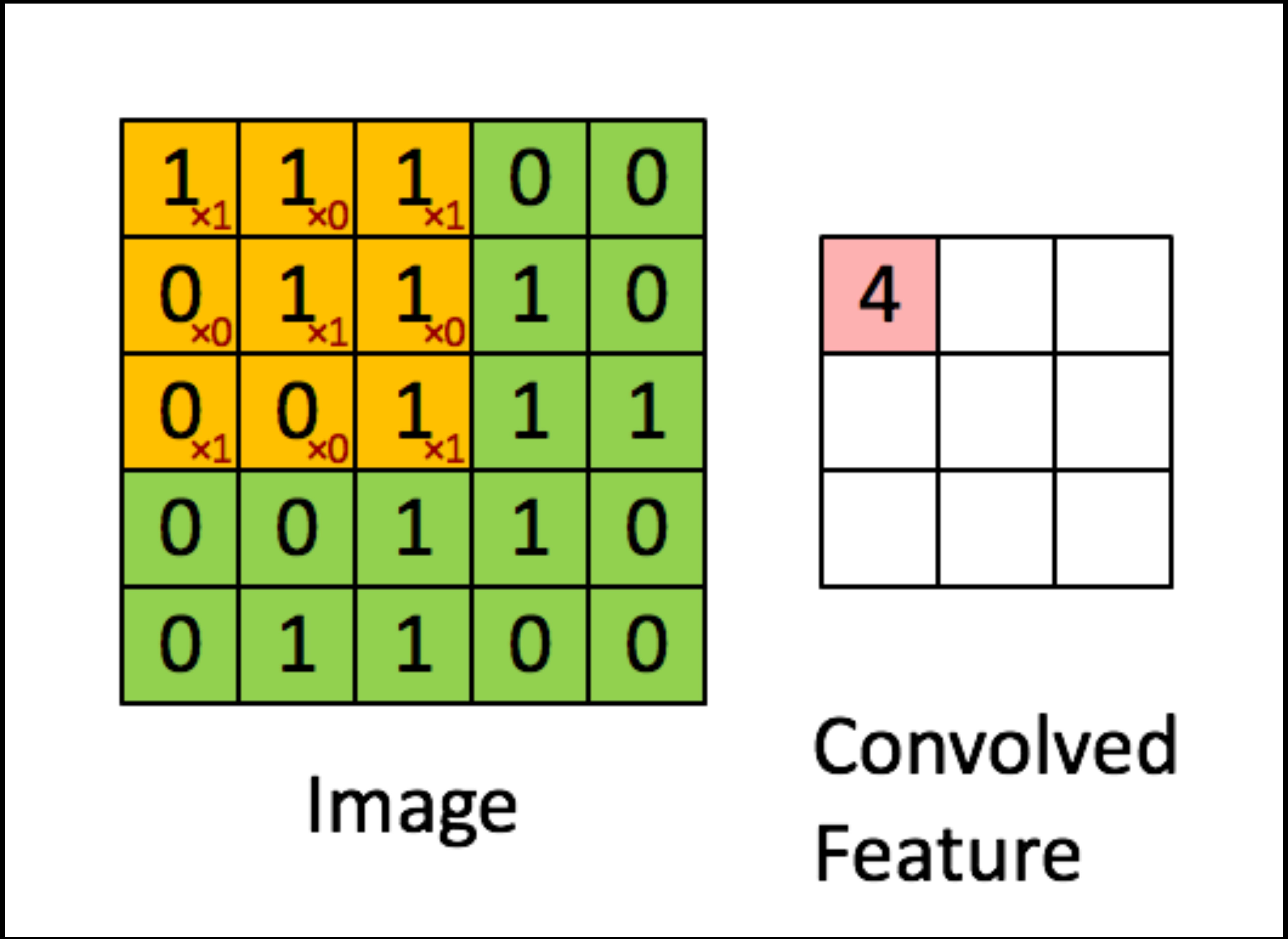


- Deep Belief Network (DBN)
- Convolutional Net (CNN)
- Recurrent Net (RNN)

Convolutional Neural Net

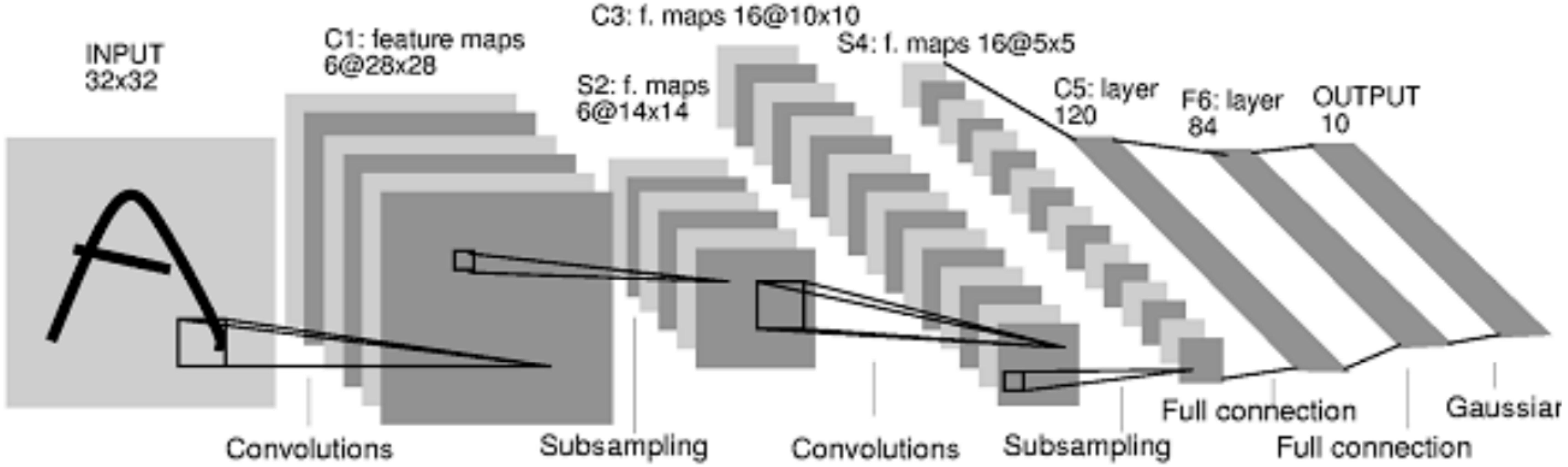


Convolutional Neural Net



1998

LeCun et al.



# of transistors

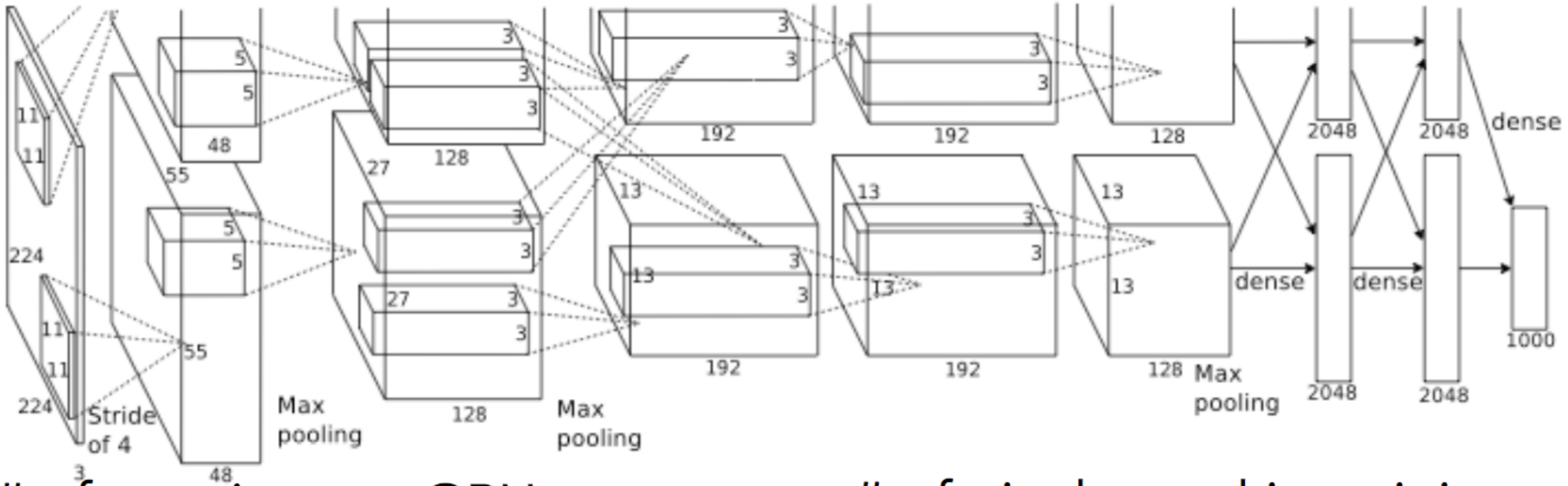
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

10^{14} IMAGENET

Year 2014

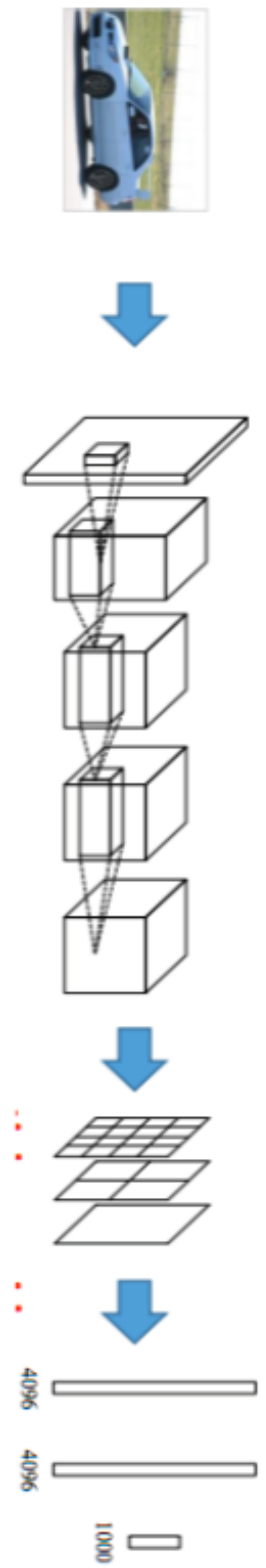
GoogLeNet

[Szegedy arxiv 2014]

VGG

[Simonyan arxiv 2014]

MSRA



[He arxiv 2014]

DrawCNN: visualizing the units' connections

Agrawal, et al. Analyzing the performance of multilayer neural networks for object recognition. ECCV, 2014
Szegedy, et al. Intriguing properties of neural networks.arXiv preprint arXiv:1312.6199, 2013
Zeiler, M. et al. Visualizing and Understanding Convolutional Networks, ECCV 2014

1. Preprocess the data
2. Choose architecture
3. Train
4. Optimize/Regularize
5. Tips/Tricks

Training a (deep) Neural Network

1. Preprocess the data
2. Choose architecture
3. Train (Code Finally!)
4. Optimize/Regularize
5. Tips/Tricks

Training a (deep) Neural Network

Lasagne

```
net = NeuralNet(
    layers=[
        ('input', layers.InputLayer),
        ('conv1', Conv2DLayer),
        ('pool1', MaxPool2DLayer),
        ('dropout1', layers.DropoutLayer),
        ('conv2', Conv2DLayer),
        ('pool2', MaxPool2DLayer),
        ('dropout2', layers.DropoutLayer),
        ('conv3', Conv2DLayer),
        ('pool3', MaxPool2DLayer),
        ('dropout3', layers.DropoutLayer),
        ('hidden4', layers.DenseLayer),
        ('dropout4', layers.DropoutLayer),
        ('hidden5', layers.DenseLayer),
        ('output', layers.DenseLayer),
        ],
    input_shape=(None, 1, 96, 96),
    conv1_num_filters=32, conv1_filter_size=(3, 3), pool1_pool_size=(2, 2),
    dropout1_p=0.1,
    conv2_num_filters=64, conv2_filter_size=(2, 2), pool2_pool_size=(2, 2),
    dropout2_p=0.2,
    conv3_num_filters=128, conv3_filter_size=(2, 2), pool3_pool_size=(2, 2),
    dropout3_p=0.3,
    hidden4_num_units=1000,
    dropout4_p=0.5,
    hidden5_num_units=1000,
    output_num_units=30, output_nonlinearity=None,

    update_learning_rate=theano.shared(float32(0.03)),
    update_momentum=theano.shared(float32(0.9)),
```

(…)

layer definitions

layer parameters

Training a (deep) Neural Network

Lasagne

```
net = NeuralNet(
    layers=[
        ('input', layers.InputLayer),
        ('conv1', Conv2DLayer),
        ('pool1', MaxPool2DLayer),
        ('dropout1', layers.DropoutLayer),
        ('conv2', Conv2DLayer),
        ('pool2', MaxPool2DLayer),
        ('dropout2', layers.DropoutLayer),
        ('conv3', Conv2DLayer),
        ('pool3', MaxPool2DLayer),
        ('dropout3', layers.DropoutLayer),
        ('hidden4', layers.DenseLayer),
        ('dropout4', layers.DropoutLayer),
        ('hidden5', layers.DenseLayer),
        ('output', layers.DenseLayer),
        ],
    input_shape=(None, 1, 96, 96),
    conv1_num_filters=32, conv1_filter_size=(3, 3), pool1_pool_size=(2, 2),
    dropout1_p=0.1,
    conv2_num_filters=64, conv2_filter_size=(2, 2), pool2_pool_size=(2, 2),
    dropout2_p=0.2,
    conv3_num_filters=128, conv3_filter_size=(2, 2), pool3_pool_size=(2, 2),
    dropout3_p=0.3,
    hidden4_num_units=1000,
    dropout4_p=0.5,
    hidden5_num_units=1000,
    output_num_units=30, output_nonlinearity=None,

    update_learning_rate=theano.shared(float32(0.03)),
    update_momentum=theano.shared(float32(0.9)),
```

```
    batch_iterator_train=FlipBatchIterator(batch_size=48),
    on_epoch_finished=[
        AdjustVariable('update_learning_rate', start=0.02, stop=0.00001),
        AdjustVariable('update_momentum', start=0.9, stop=0.999),
        EarlyStopping(patience=250),
        ],
    max_epochs=10000,
    verbose=1,
    )
```

hyper
parameters

Training a (deep) Neural Network

Lift off!

```
X, y = loadstuff()
net = NeuralNet(...)
net.fit(X, y)
with open('/mnt/nets/netx.pickle', 'wb') as f:
    pickle.dump(net, f, -1)
```

epoch	train loss	valid loss	train/val	dur
1	0.11094	0.04377	2.53447	2.97s
2	0.01819	0.00842	2.16100	2.98s
3	0.00800	0.00707	1.13217	2.98s
4	0.00671	0.00667	1.00530	2.97s
5	0.00635	0.00631	1.00533	2.97s
6	0.00611	0.00608	1.00492	2.98s
7	0.00592	0.00587	1.00827	2.97s
8	0.00575	0.00569	1.01183	2.97s
9	0.00561	0.00552	1.01514	2.97s
10	0.00548	0.00538	1.01815	2.97s

Training a (deep) Neural Network

1. Preprocess the data
2. Choose architecture
3. Train → Debug
4. Optimize/Regularize
5. Tips/Tricks

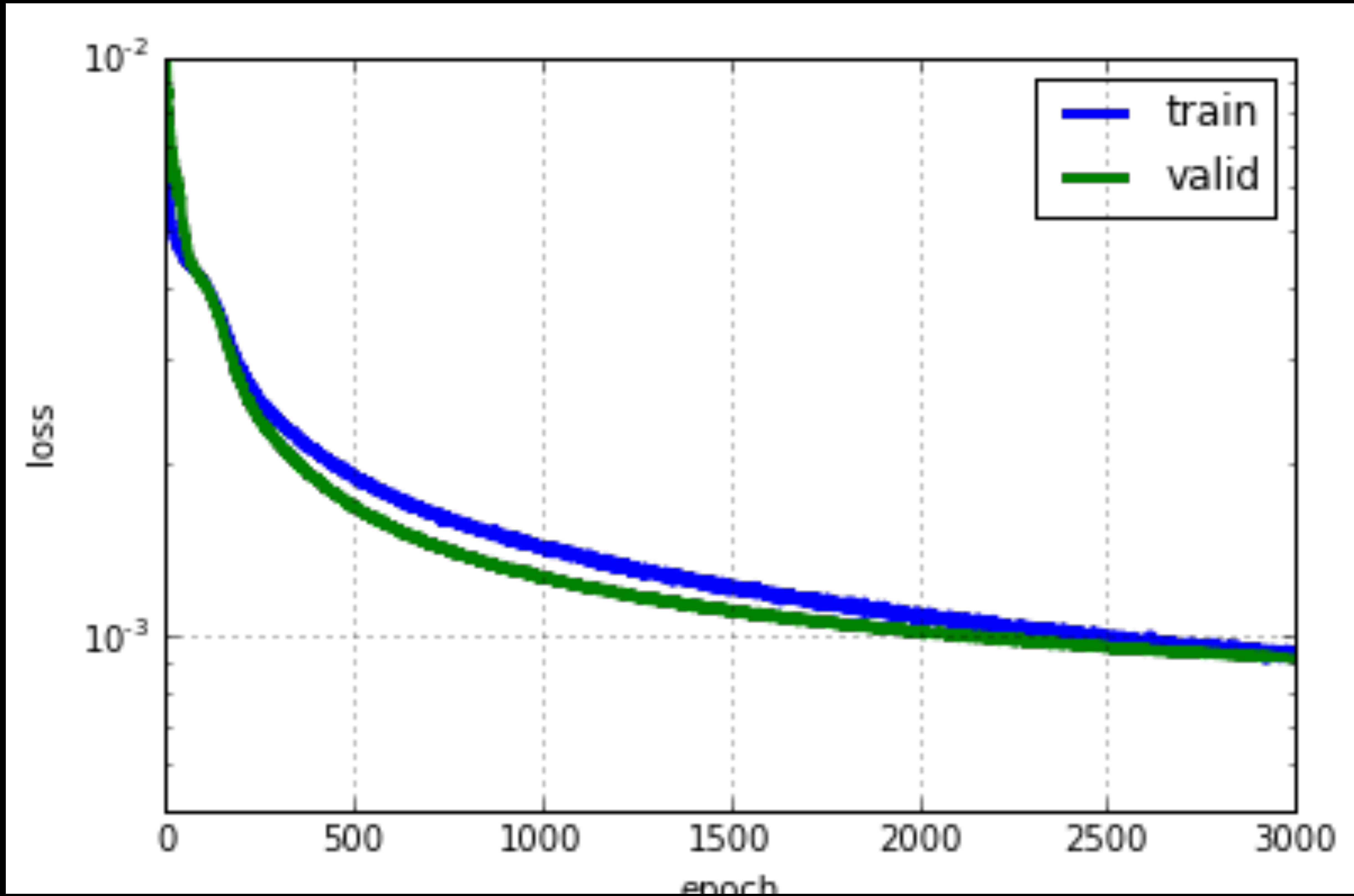
Debug Training: Visualize Loss Curve

Lasagne

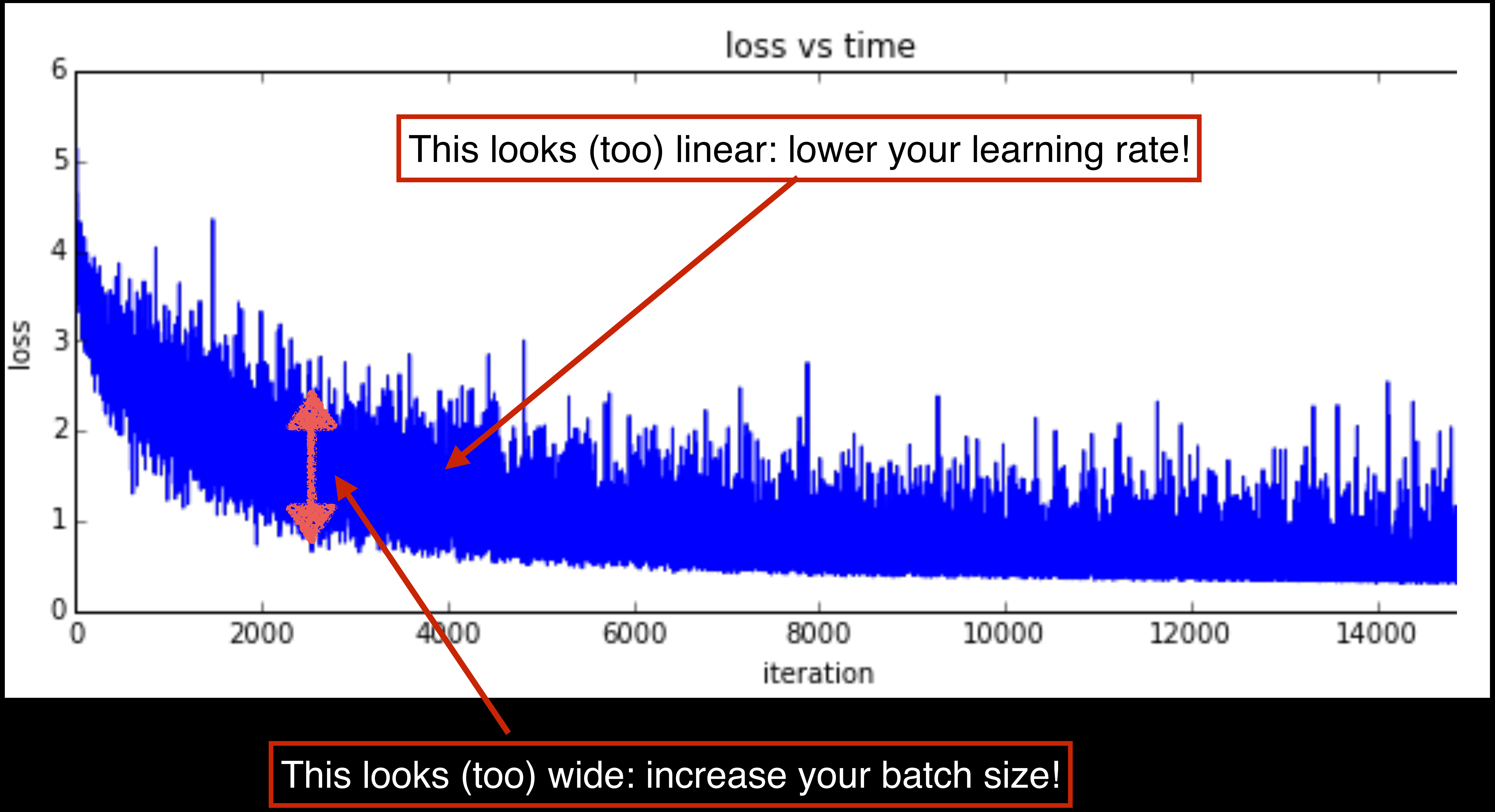
```
import numpy as np
import matplotlib.pyplot as pyplot
%matplotlib inline

import cPickle as pickle
with open('/mnt/nets/netx.pickle', 'rb') as f:
    net = pickle.load(f)

train_loss = np.array([i["train_loss"] for i in net.train_history_])
valid_loss = np.array([i["valid_loss"] for i in net.train_history_])
pyplot.plot(train_loss, linewidth=3, label="train")
pyplot.plot(valid_loss, linewidth=3, label="valid")
pyplot.grid()
pyplot.legend()
pyplot.xlabel("epoch")
pyplot.ylabel("loss")
pyplot.ylim(1e-4*5, 1e-2)
pyplot.yscale("log")
pyplot.show()
```

Debug Training: Visualize Loss Curve

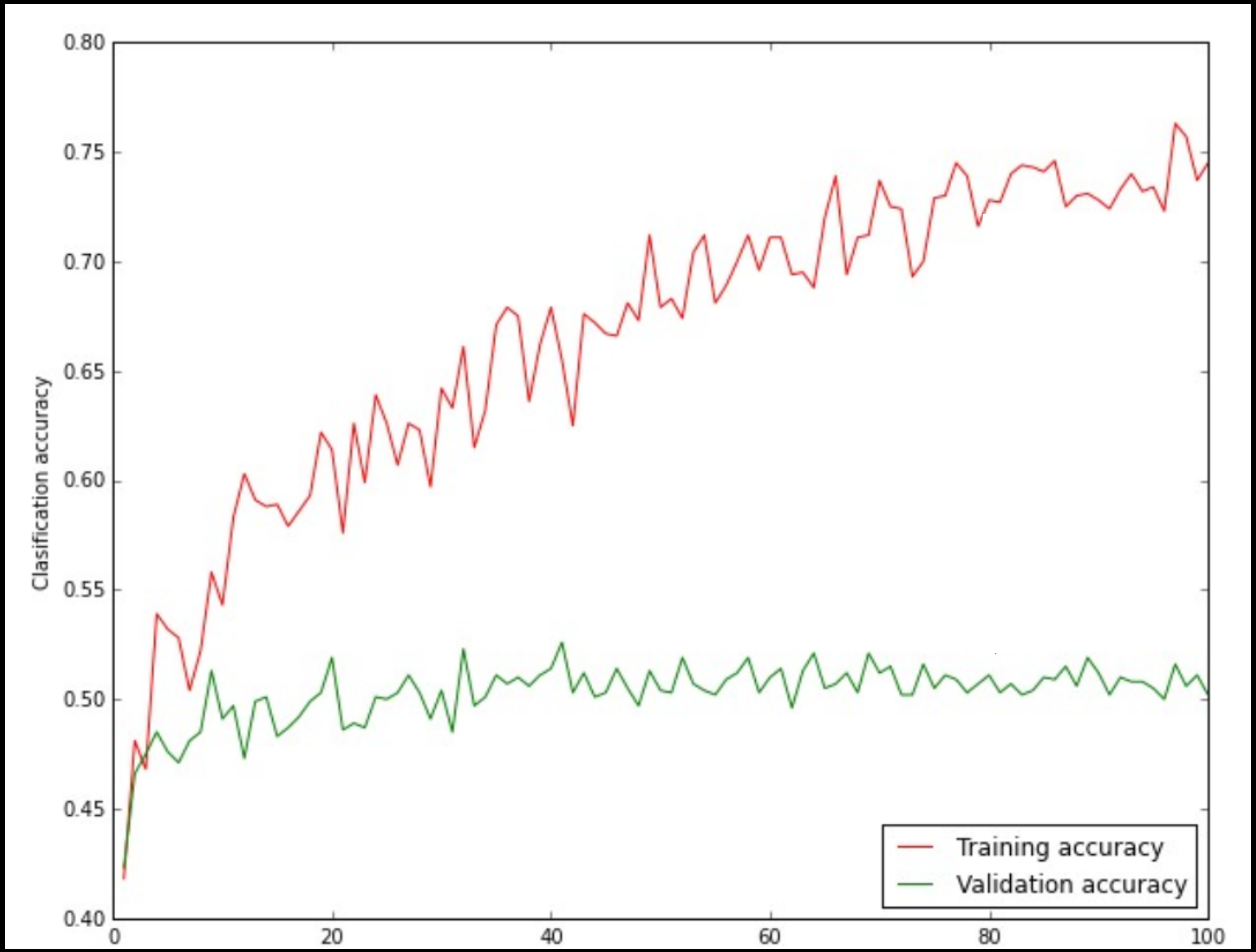
```
import numpy as np
import matplotlib.pyplot as pyplot
%matplotlib inline

import cPickle as pickle
with open('/mnt/nets/netx.pickle', 'rb') as f:
    net = pickle.load(f)

train_loss = np.array([i["train_loss"] for i in net.train_history_])
valid_loss = np.array([i["valid_loss"] for i in net.train_history_])
pyplot.plot(train_loss, linewidth=3, label="train")
pyplot.plot(valid_loss, linewidth=3, label="valid")
pyplot.grid()
pyplot.legend()
pyplot.xlabel("epoch")
pyplot.ylabel("loss")
pyplot.ylim(1e-4*5, 1e-2)
pyplot.yscale("log")
pyplot.show()
```



Debug Training: Visualize Loss Curve



Debug Training: Visualize Accuracy

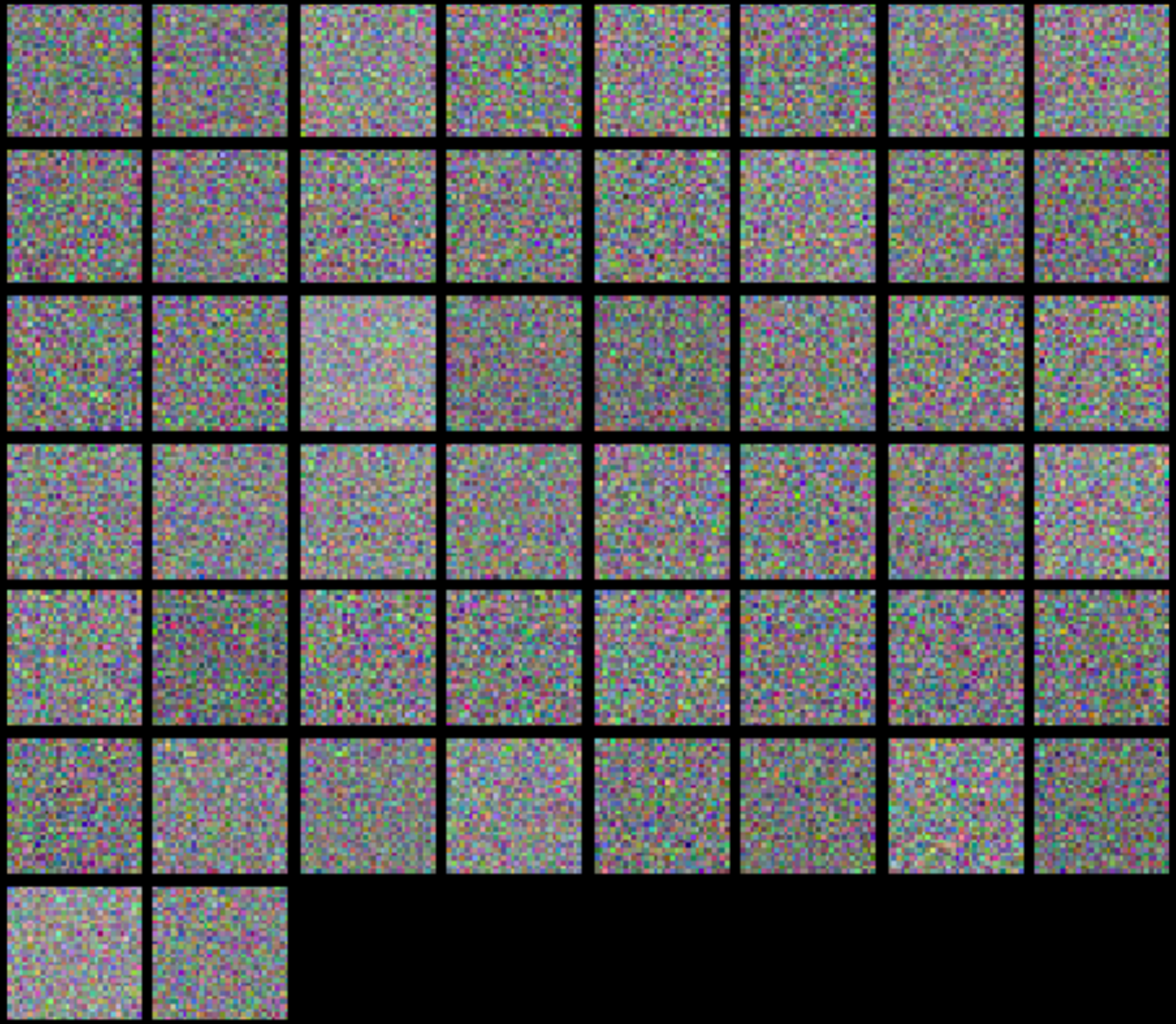


big gap: overfitting: regularize!

no gap: underfitting
(increase model size)

Debug Training: Visualize Weights

(usually: first layer)



complete mess, doesn't get past the random initialisation

Debug Training: Visualize Weights

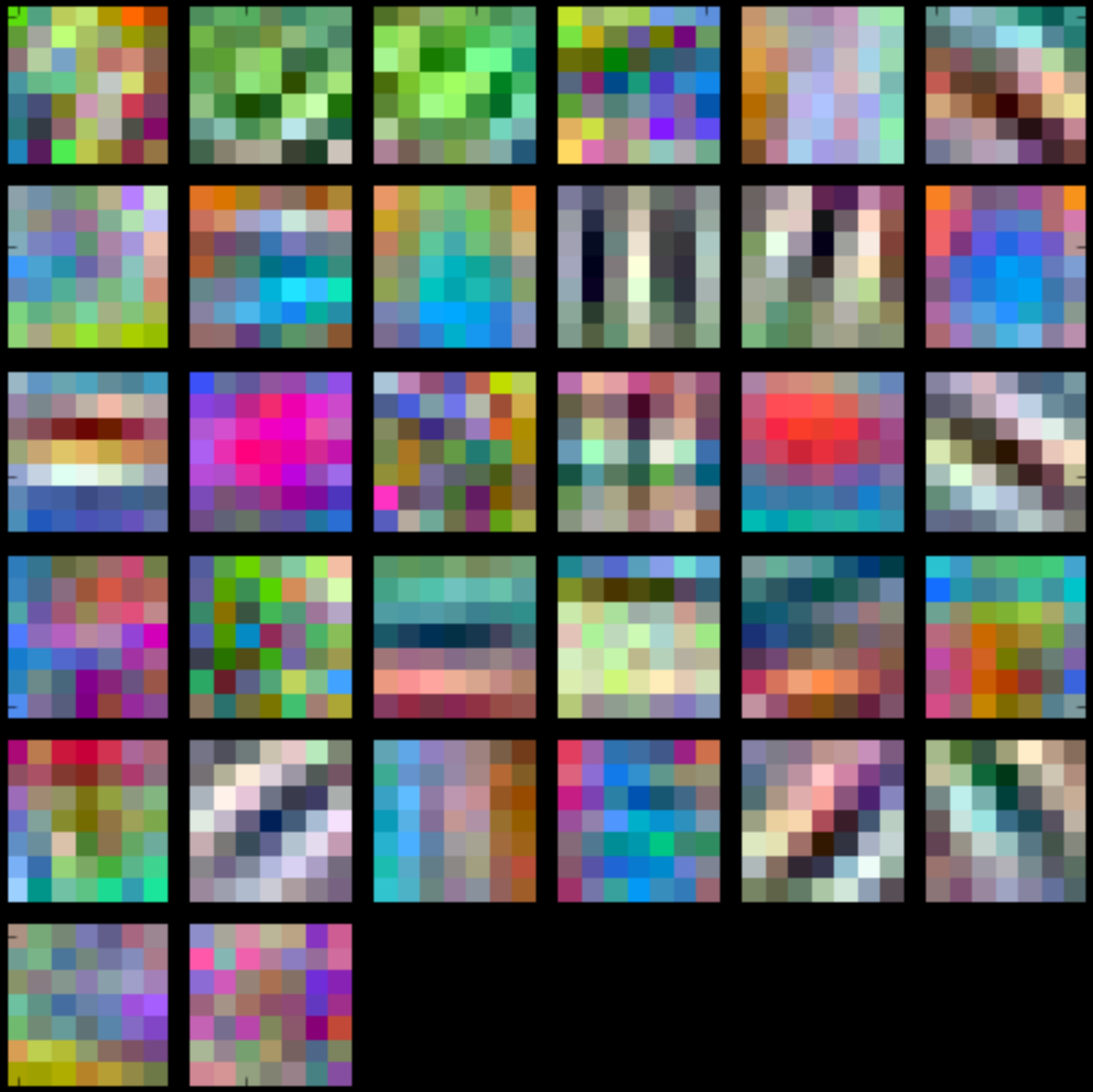
(usually: first layer)

better, but still “noisy” weights

mostly solvable by stronger regularisation

Debug Training: Visualize Weights

(usually: first layer)



good: now operates as “edge detector”

1. Preprocess the data
2. Choose architecture
3. Train ⟶ Debug
4. Optimize/Regularize
5. Tips/Tricks

Optimize / Regularize

- Tweak Hyperparameters / Architecture
- Data Augmentation
- Dropout
- Batch Normalization

- Tweak Hyperparameters / Architecture
- Data Augmentation
- Dropout
- Batch Normalization

Choosing Hyperparameters

- Grid search won't work on your millions + parameters
- Random Search? Mwah…
- Bayesian Optimization: Yeh baby!
  - Spearmint: https://github.com/HIPS/Spearmint
  - Hypergrad: https://github.com/HIPS/hypergrad

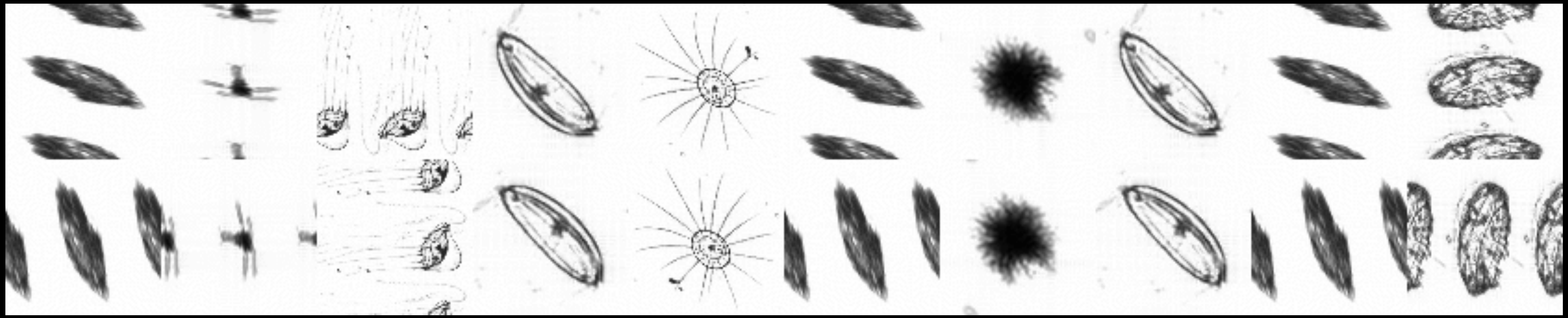
Overfitting

- Tweak Hyperparameters / Architecture
- Data Augmentation
- Dropout
- Batch Normalization

Data Augmentation

Data Augmentation

* rotation: random with angle between 0° and 360° (uniform)
* translation: random with shift between -10 and 10 pixels (uniform)
* rescaling: random with scale factor between 1/1.6 and 1.6 (log-uniform)
* flipping: yes or no (bernoulli)
* shearing: random with angle between -20° and 20° (uniform)
* stretching: random with stretch factor between 1/1.3 and 1.3 (log-uniform)

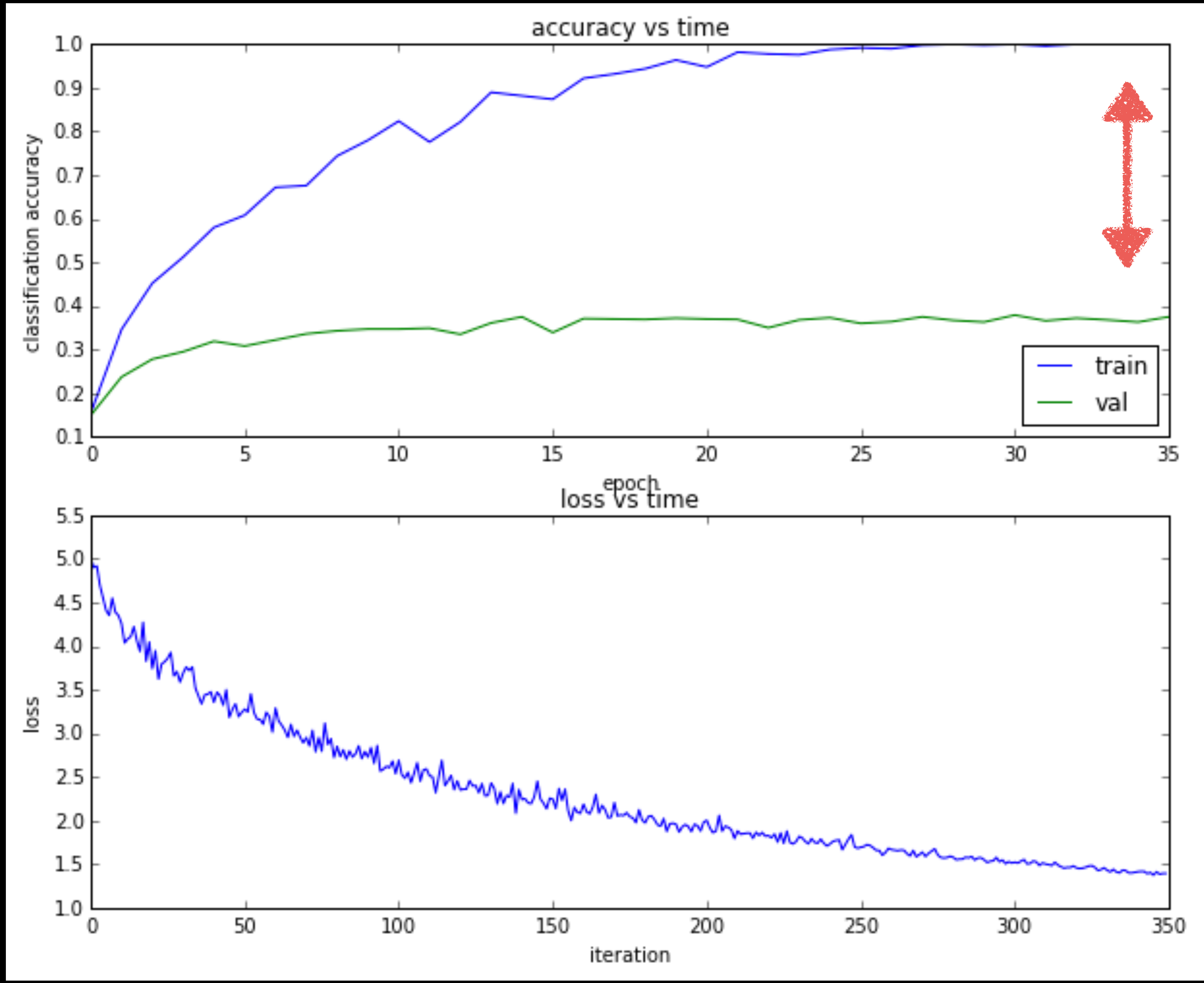


(realtime data augmentation at Kaggle's #1 National Data Science Bowl ≈ Deep Sea ≈ team)

http://benanne.github.io/2015/03/17/plankton.html

- Tweak Hyperparameters / Architecture
- Data Augmentation
- Dropout
- Batch Normalization

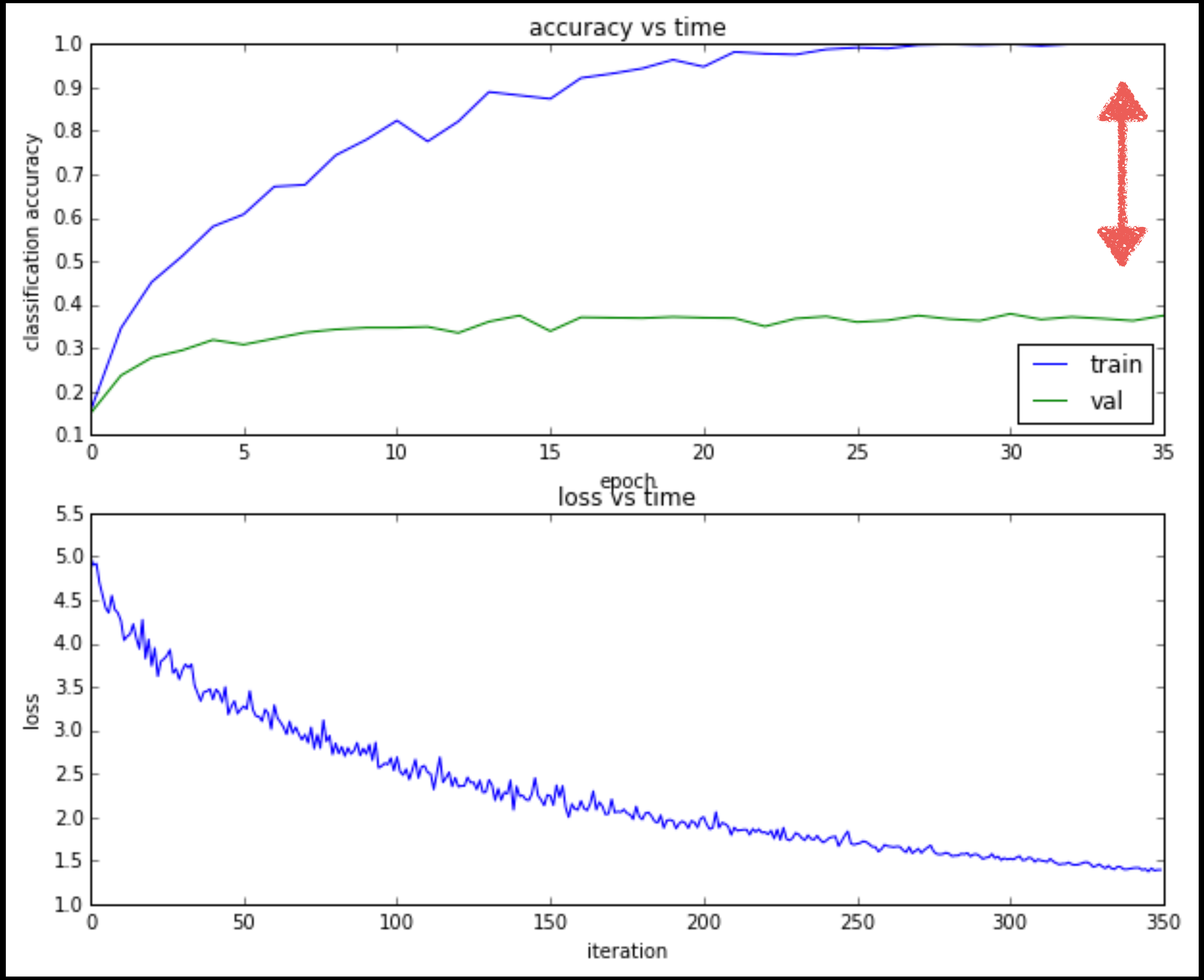
Dropout as Regularization



(naively trained net)

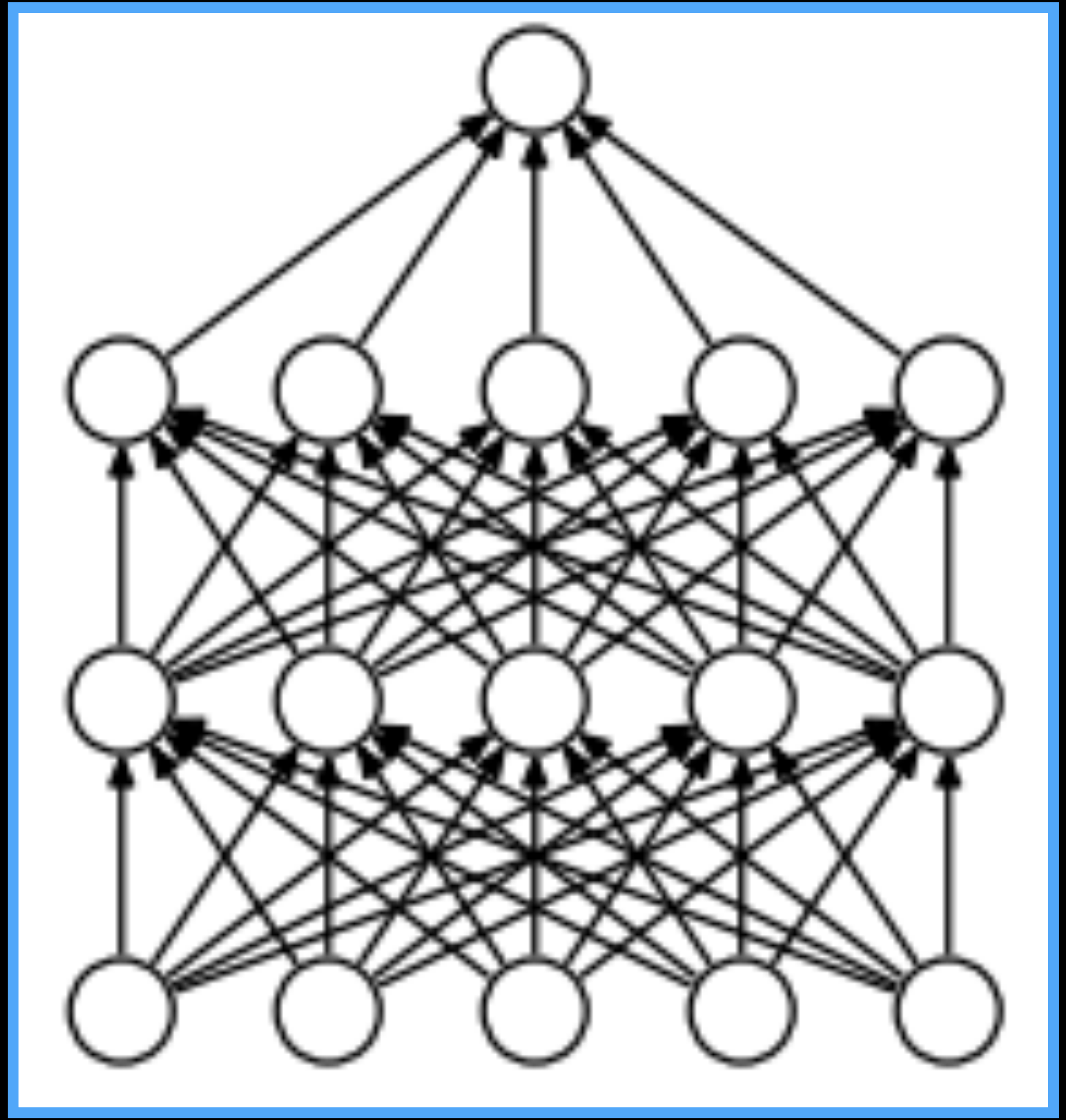
Overfits !

Dropout as Regularization



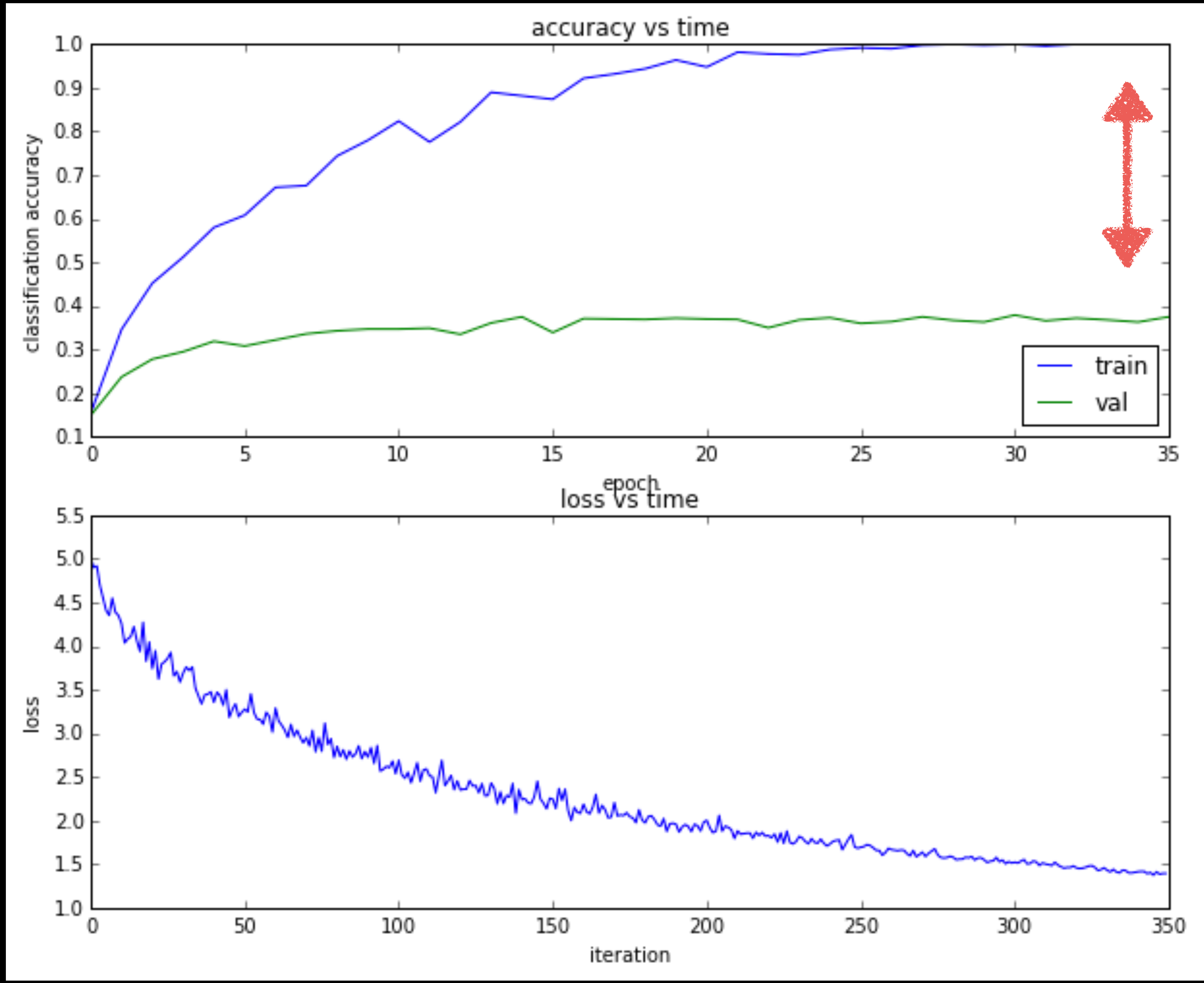
(naively trained net)

Overfits !



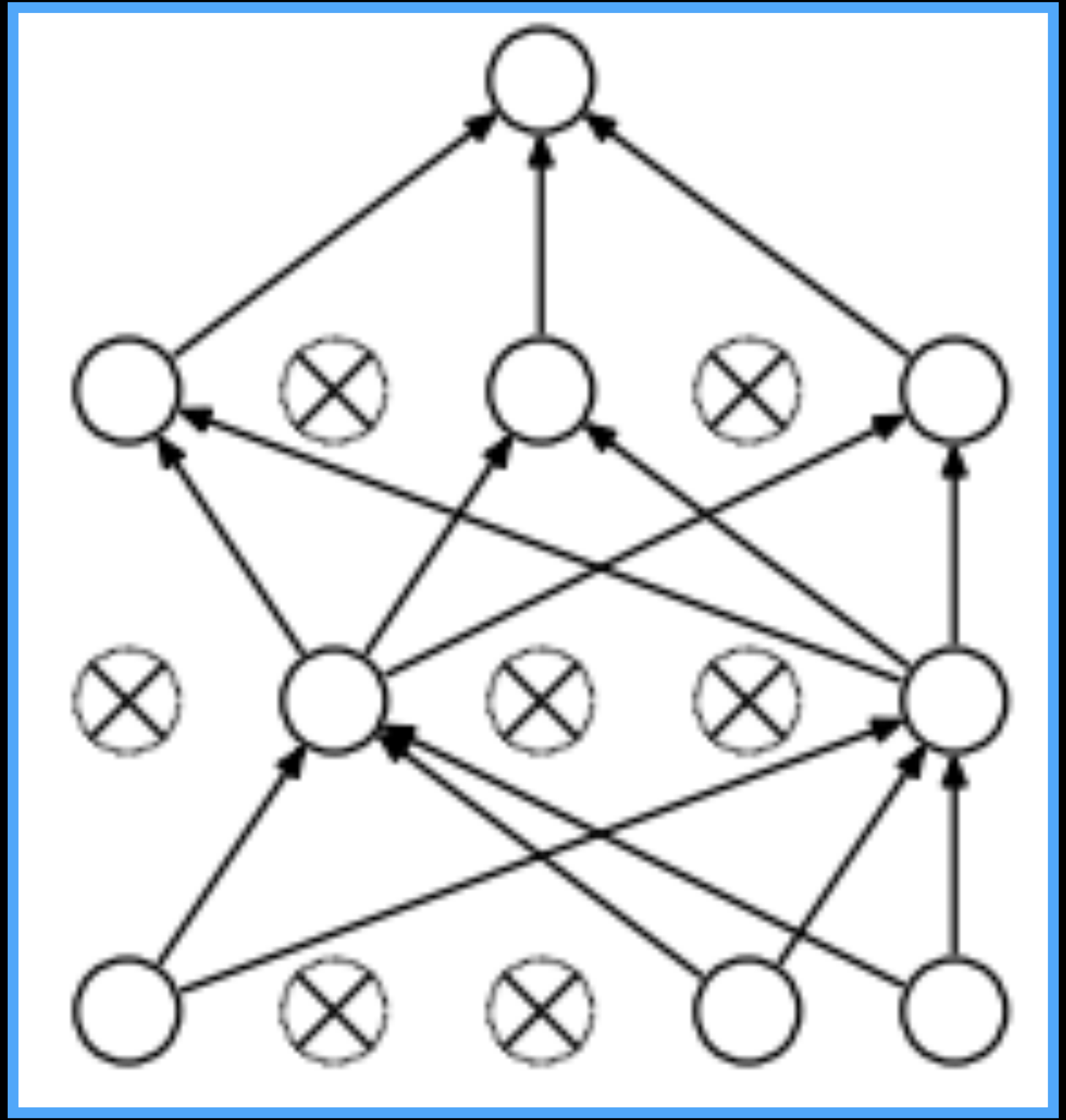
Dropout!

Dropout as Regularization



(naively trained net)

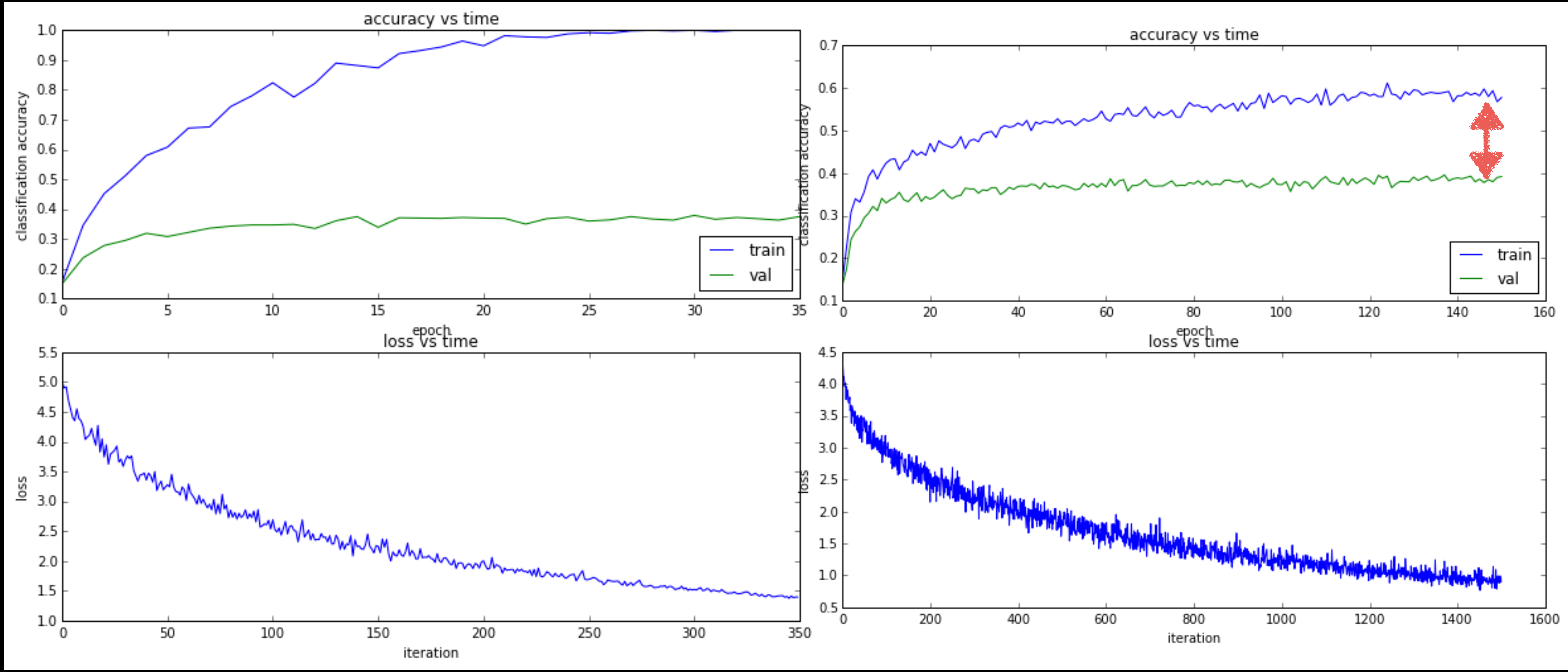
Overfits !



Dropout!

Dropout as Regularization

less strongly overfitted &
can run for more epochs higher accuracy



(naively trained net)

(net with dropout)

Overfitting

- Tweak Hyperparameters / Architecture
- Data Augmentation
- Dropout
- Batch Normalization

Batch Normalization as regularization

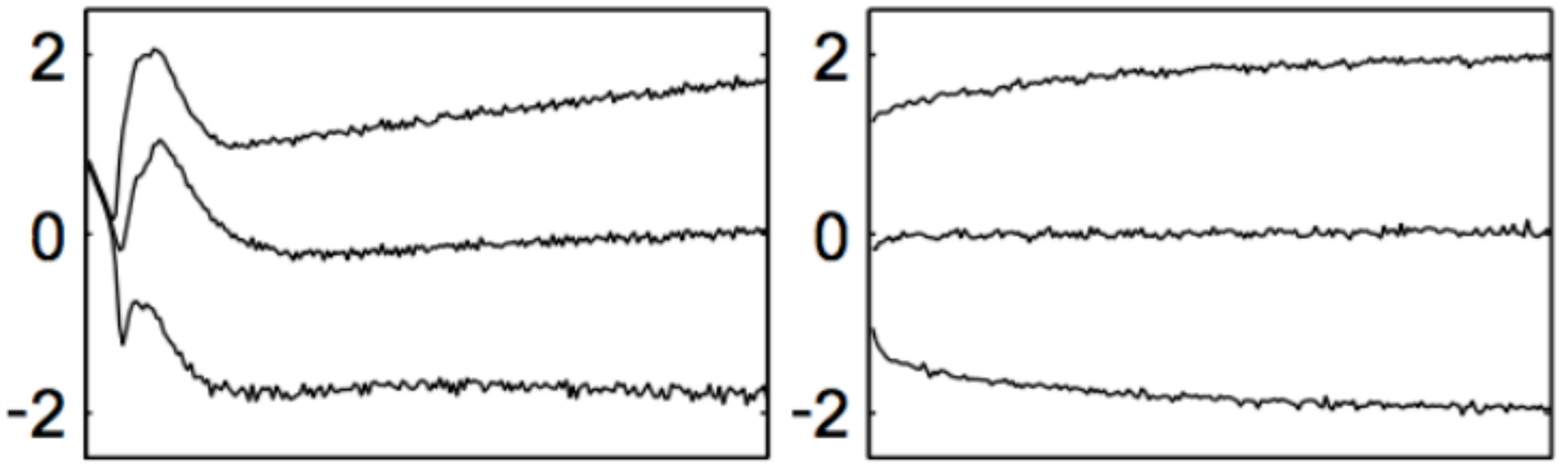
- Normalize the activations in each layer within a minibatch
- Learn the mean and variance of each layer as parameters

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



(b) Without BN (c) With BN

85%

50%

15%

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
S Ioffe and C Szegedy (2015)

Training a (deep) Neural Network

1. Preprocess the data
2. Choose architecture
3. Train ⟶ Debug
4. Optimize/Regularize
5. Further Tips & Tricks to improve Model Accuracy

Other “Tricks”

- Ensembles
- Finetuning pre-trained/earlier-trained net
- Sticking extracted layer features in another classifier (ie SVM)

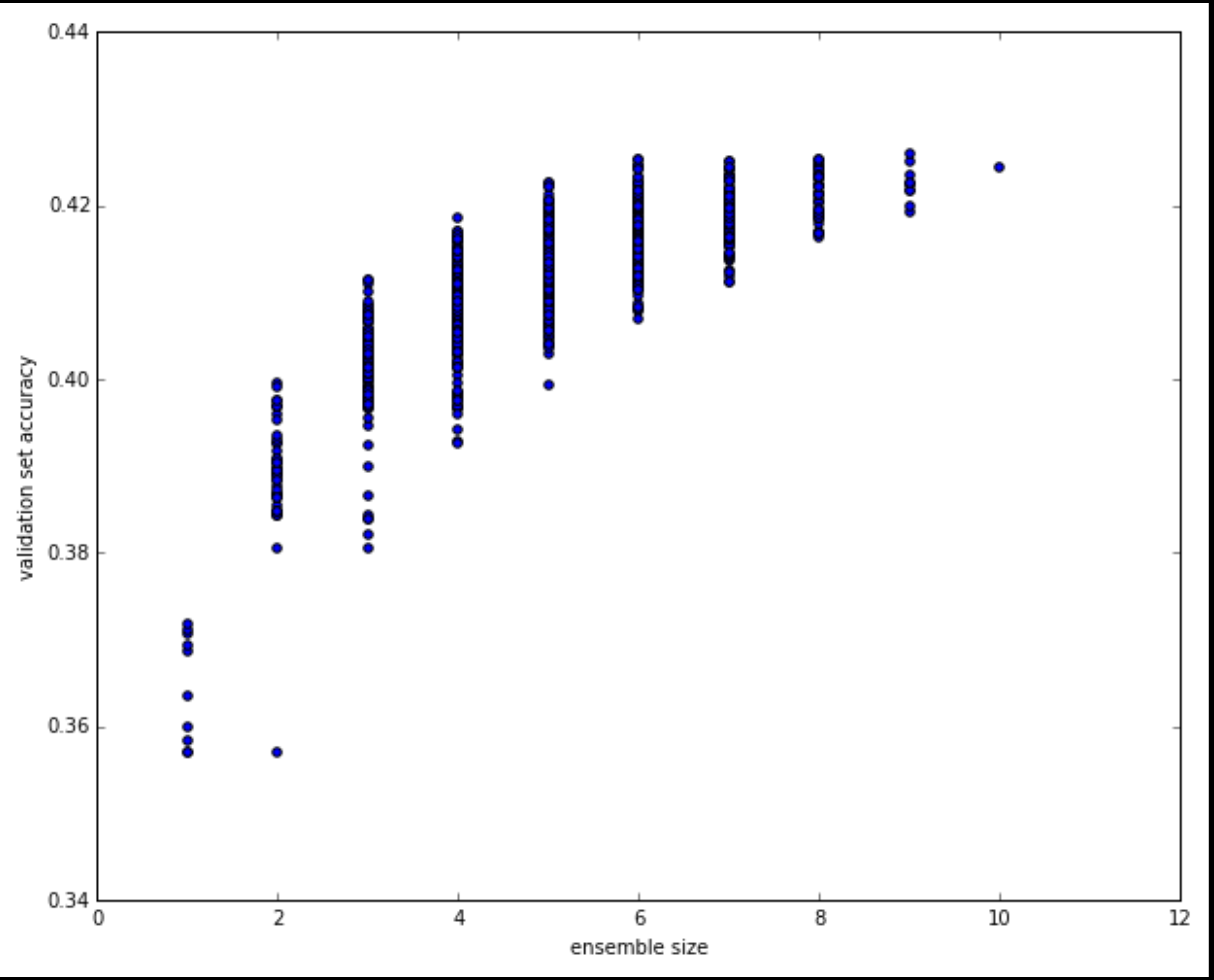
Other "Tricks"

- Ensembles
- ~~Finetuning pre-trained/earlier-trained net~~
- ~~Sticking extracted layer features in another classifier (ie SVM)~~

Ensembles

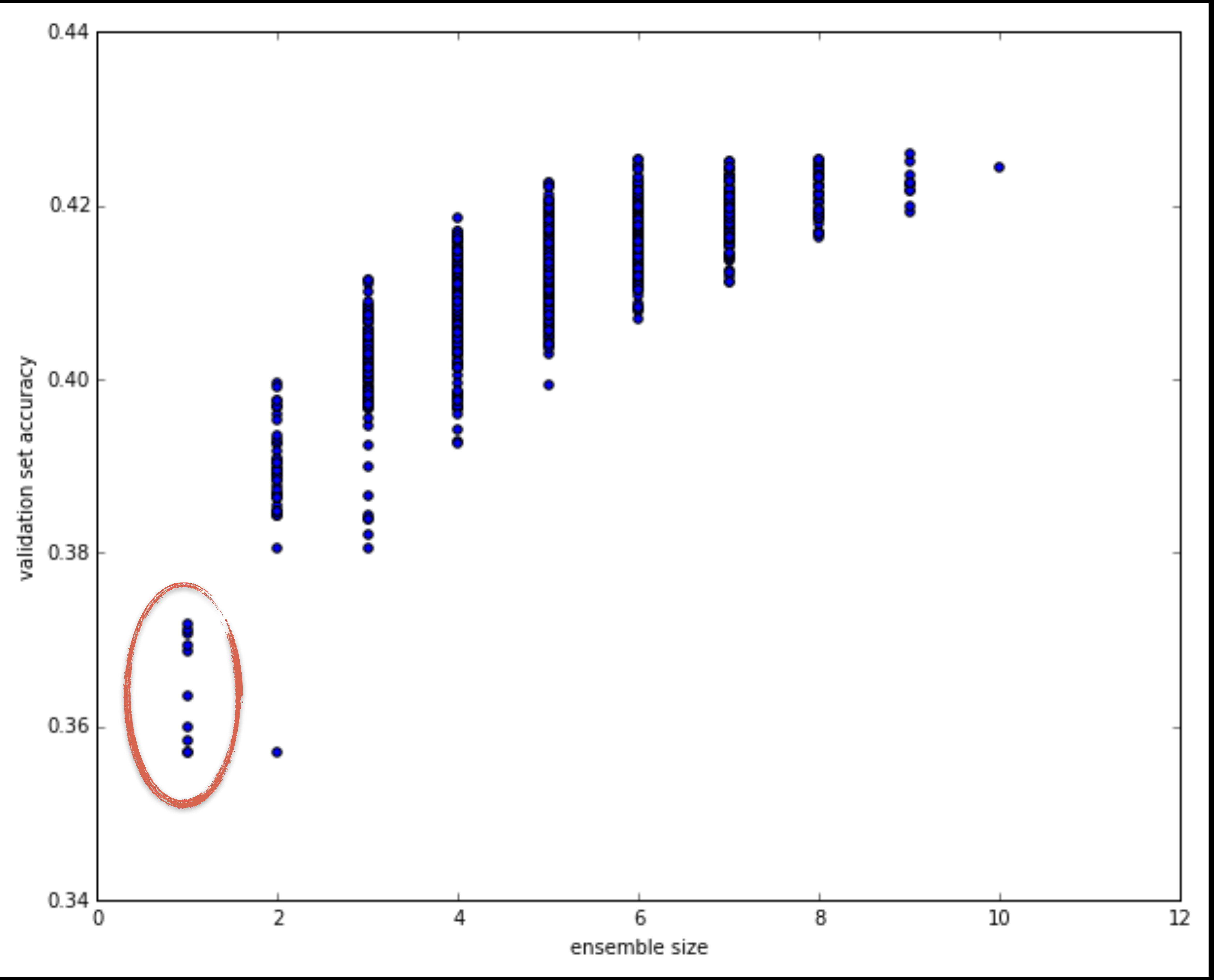
- majority vote when hard predictions (ie classes)
- average vote for soft predictions (continious scale)
- make sure classifiers are uncorrelated
- cross validate ensemble weights (by grid search, or rank average)
- stacked
- blending

Ensembles



(10 similar nets with varying hyperparameters on same tiny-imagenet dataset)

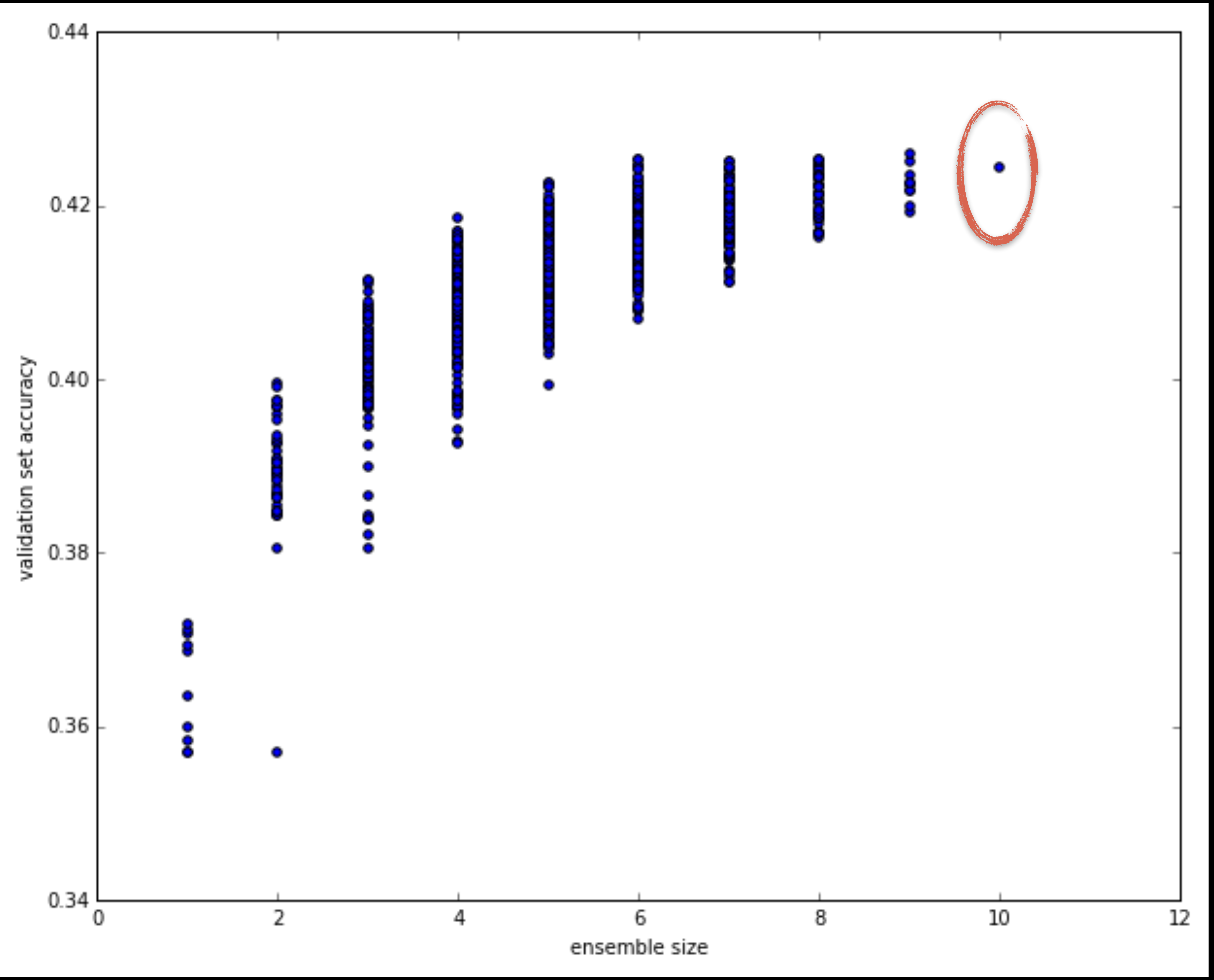
Ensembles



avg: 0.3647

(10 similar nets with varying hyperparameters on same tiny-imagenet dataset)

Ensembles

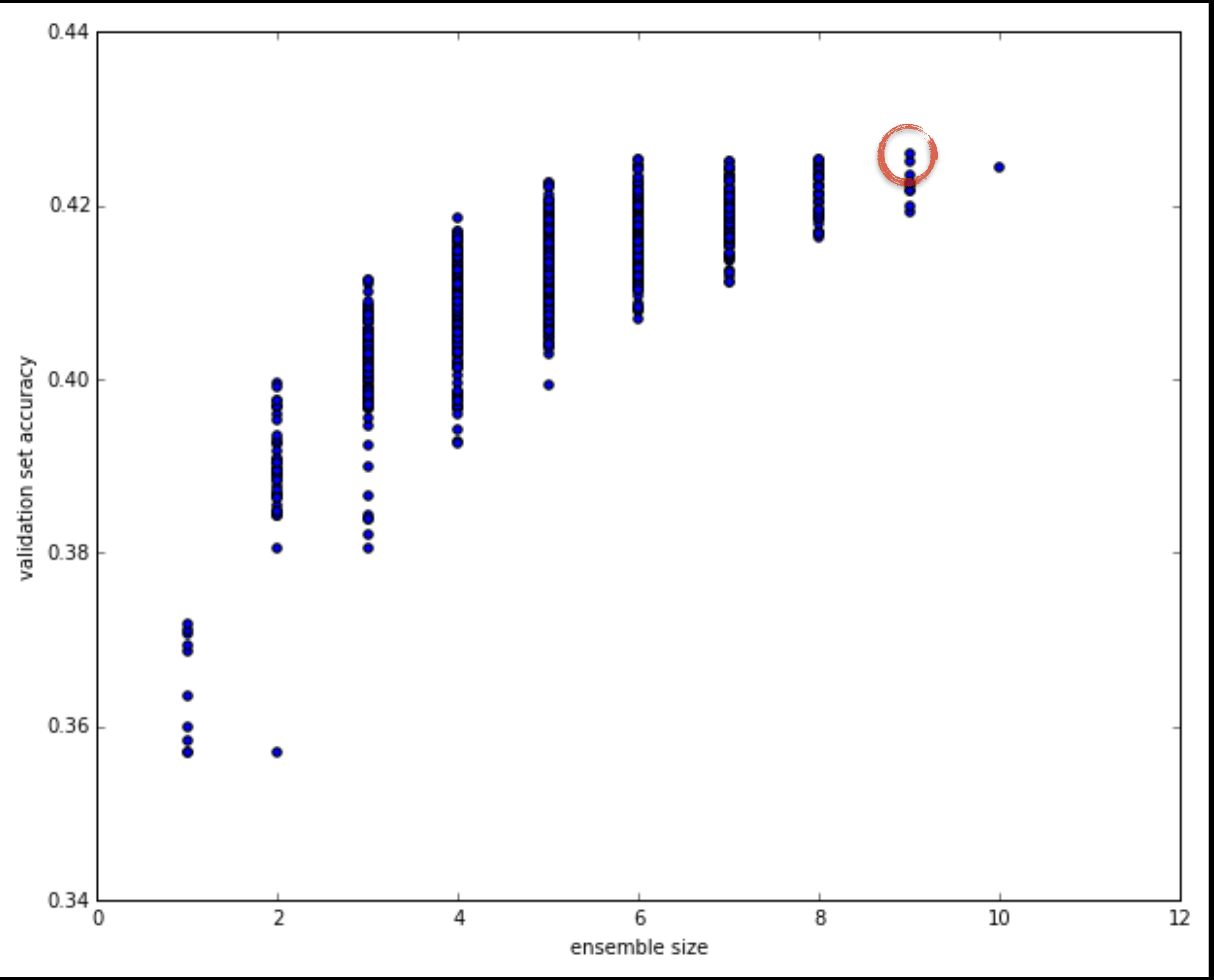


avg: 0.3647

predict by mean of all: 0.4244

(10 similar nets with varying hyperparameters on same tiny-imagenet dataset)

Ensembles



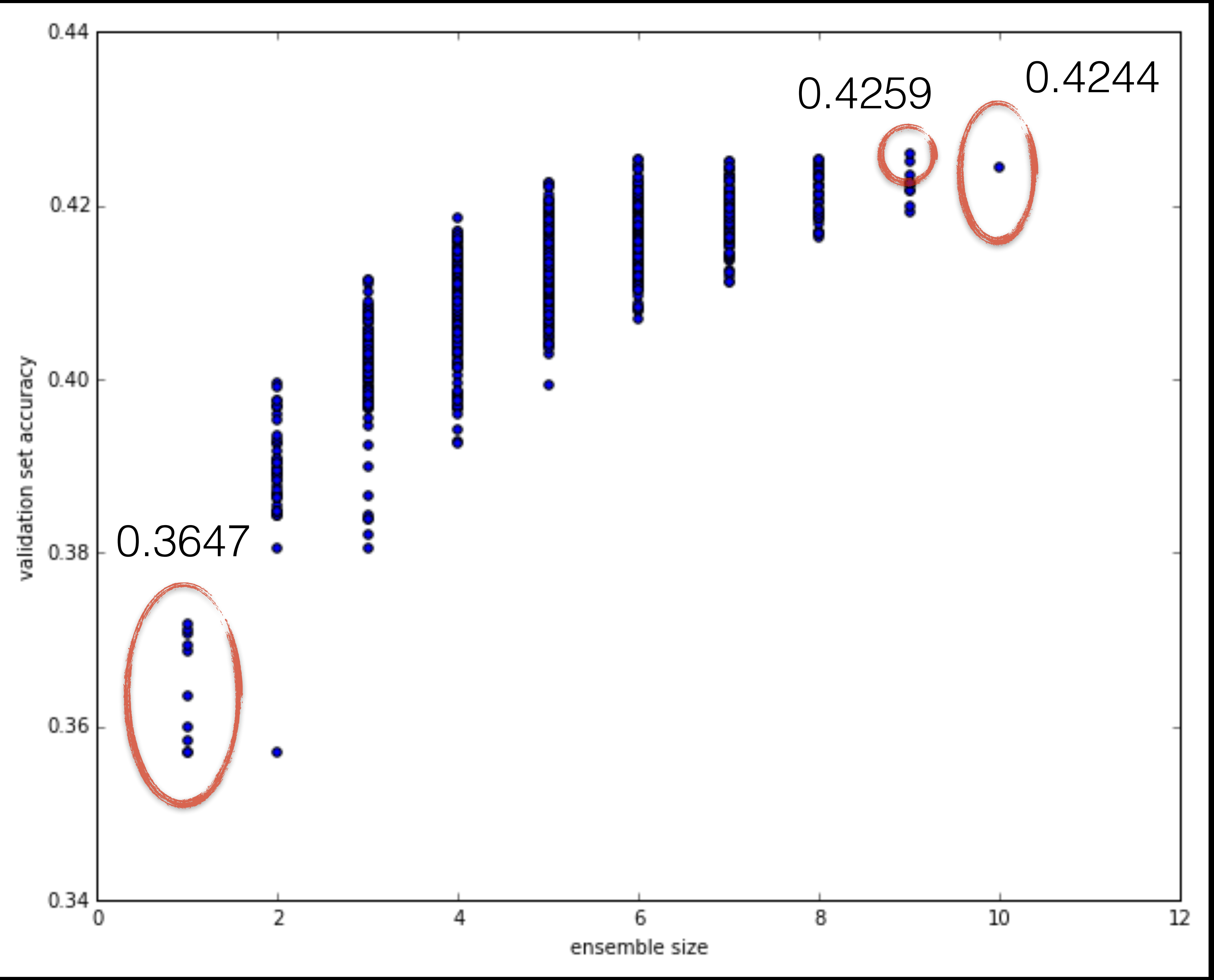
avg: 0.3647

predict by mean of all: 0.4244

leave out model9: 0.4259

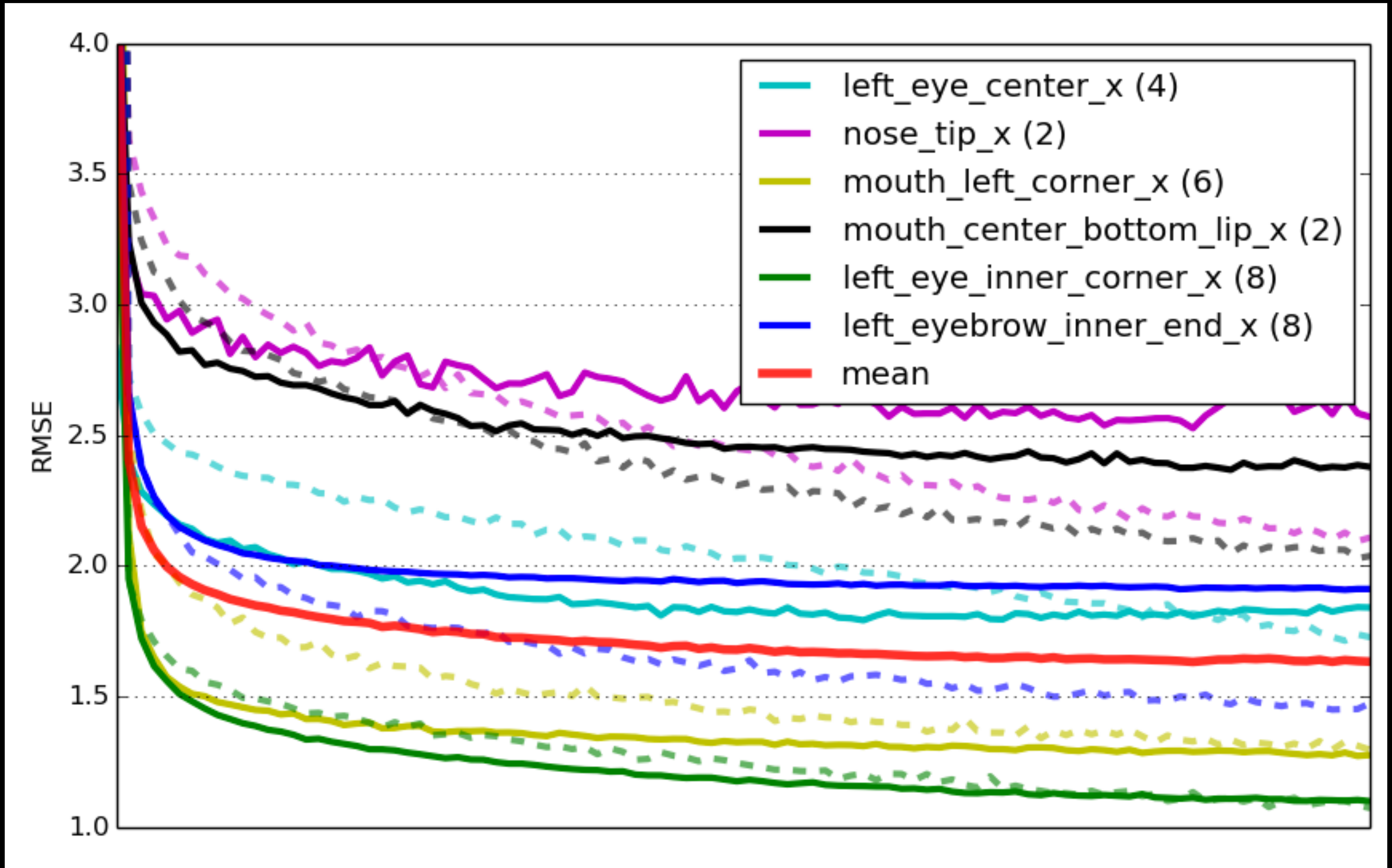
(10 similar nets with varying hyperparameters on same tiny-imagenet dataset)

Ensembles



(10 similar nets with varying hyperparameters on same tiny-imagenet dataset)

"Specialists" Ensemble



(ensembling specialist nets by Daniel Nouri, Kaggle facial keypoint tutorial)
danielnouri.org/notes/2014/12/17/using-convolutional-neural-nets-to-detect-facial-keypoints-tutorial/

try it yourself :)

3 similar nets trained on the same data but with different hyper parameters.

disclaimer: Kaggle is not real life, people...

RMSE's:

- 2,08449
- 2,04575
- 2.01565

together:

1.93397

Dashboard

Public Leaderboard - Facial Keypoints Detection

This leaderboard is calculated on approximately 50% of the test data.
The final results will be based on the other 50%, so the final standings may be different.

See someone using multiple accounts?
Let us know.

#	Δ1w	Team Name * in the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑3	**Roelof Pieters** *	1.93397	6	Wed, 17 Jun 2015 09:53:38

Your Best Entry ↑

Number One!

You jumped into first by improving your score by 0.08169.

You just moved up 2 positions on the leaderboard. Tweet this!

#	Δ1w	Team Name	Score	Entries	Last Submission UTC (Best – Last Submission)
2	↓1	stuttno	1.95351	1	Fri, 22 May 2015 07:48:19
3	↓1	Xiang&Zhen&Shuping	1.96004	1	Wed, 22 Apr 2015 13:50:46
4	↑3	Johannes Höhne	2.01850	3	Fri, 12 Jun 2015 15:30:40
5	↓2	OuluQXB	2.03985	6	Tue, 26 May 2015 01:29:06 (-2.6d)
6	↓1	Mattia 2	2.09702	6	Sun, 10 May 2015 22:40:40 (-46.8h)
7	↓1	Zhihang Fan	2.09735	7	Mon, 27 Apr 2015 03:55:54
8	—	OlivierH	2.15010	4	Wed, 03 Jun 2015 10:36:30 (-40.9d)

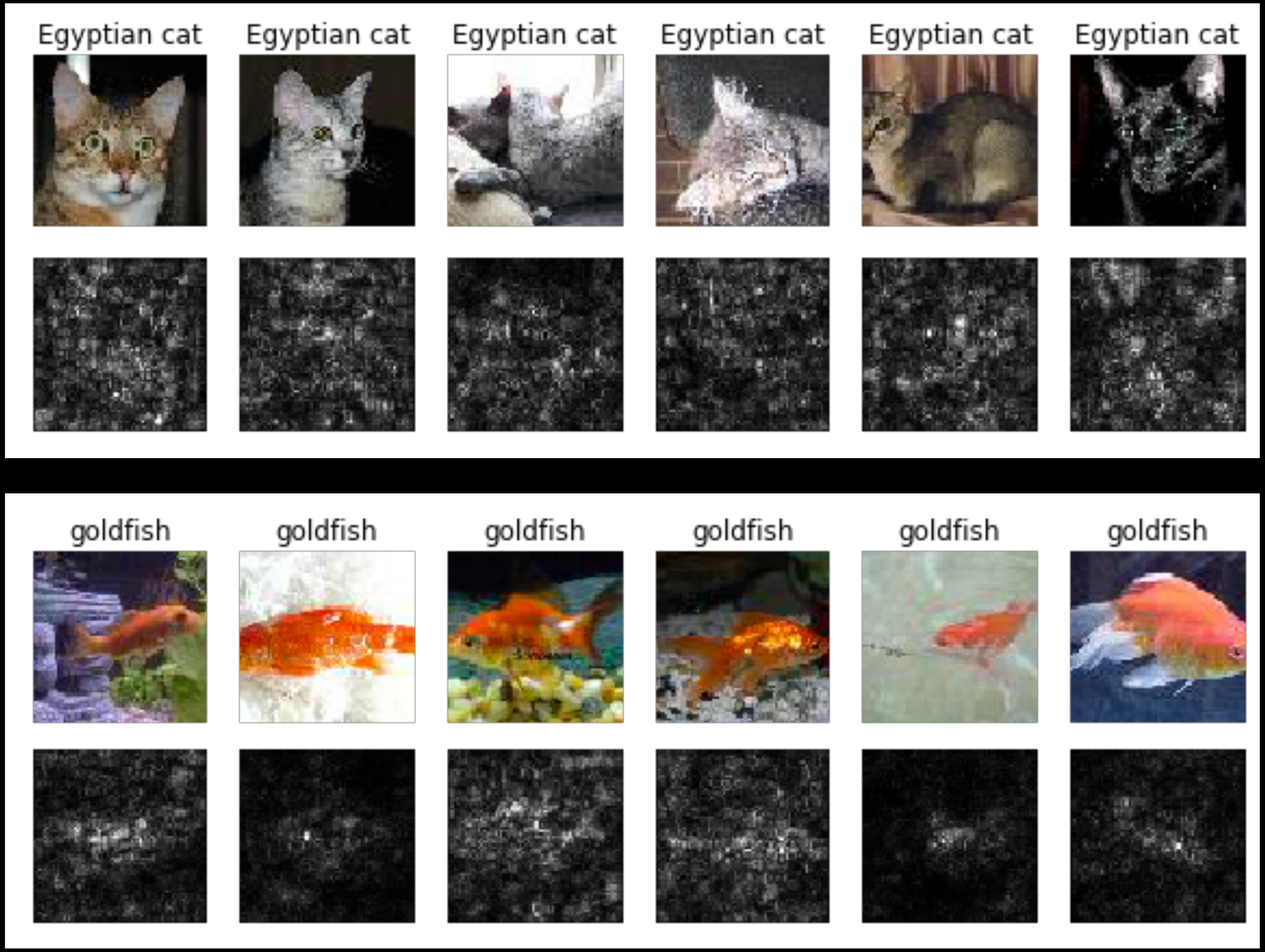
https://www.kaggle.com/c/facial-keypoints-detection/

but beware… / no free lunch:

Machine learning systems can easily be **fooled**

Saliency Maps

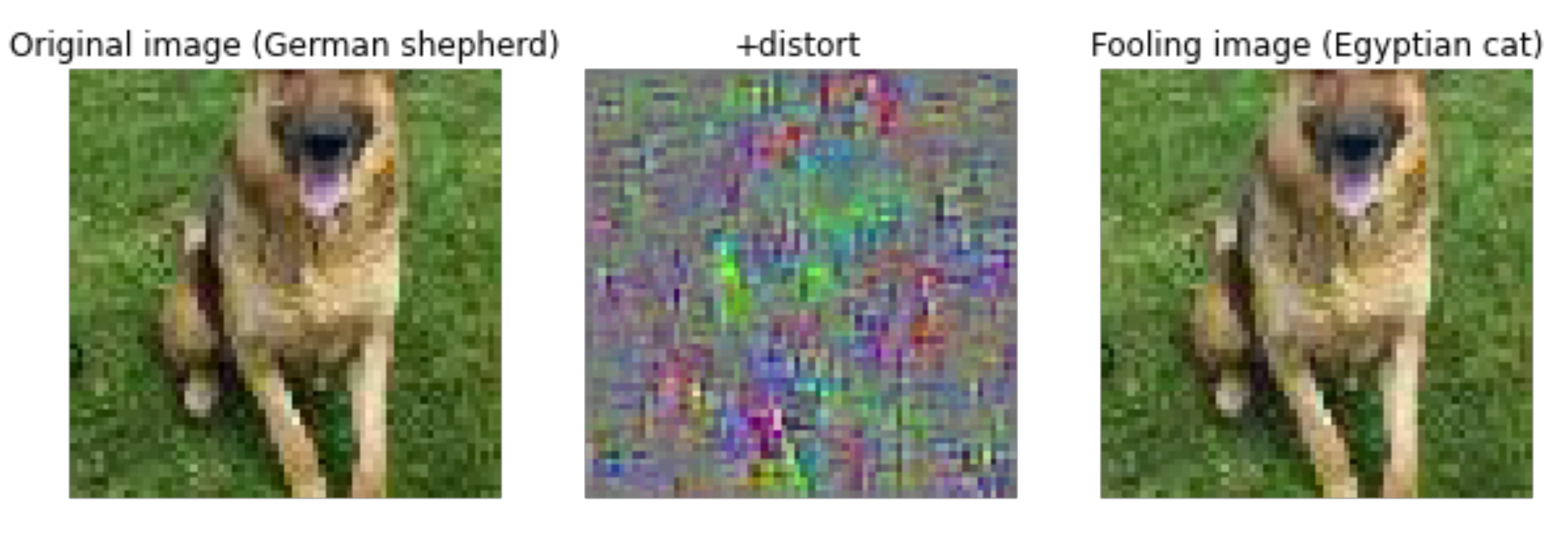
first we predict on a pixel level



K. Simonyan, A. Vedaldi, A. Zisserman , "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014

Fooling ConvNets

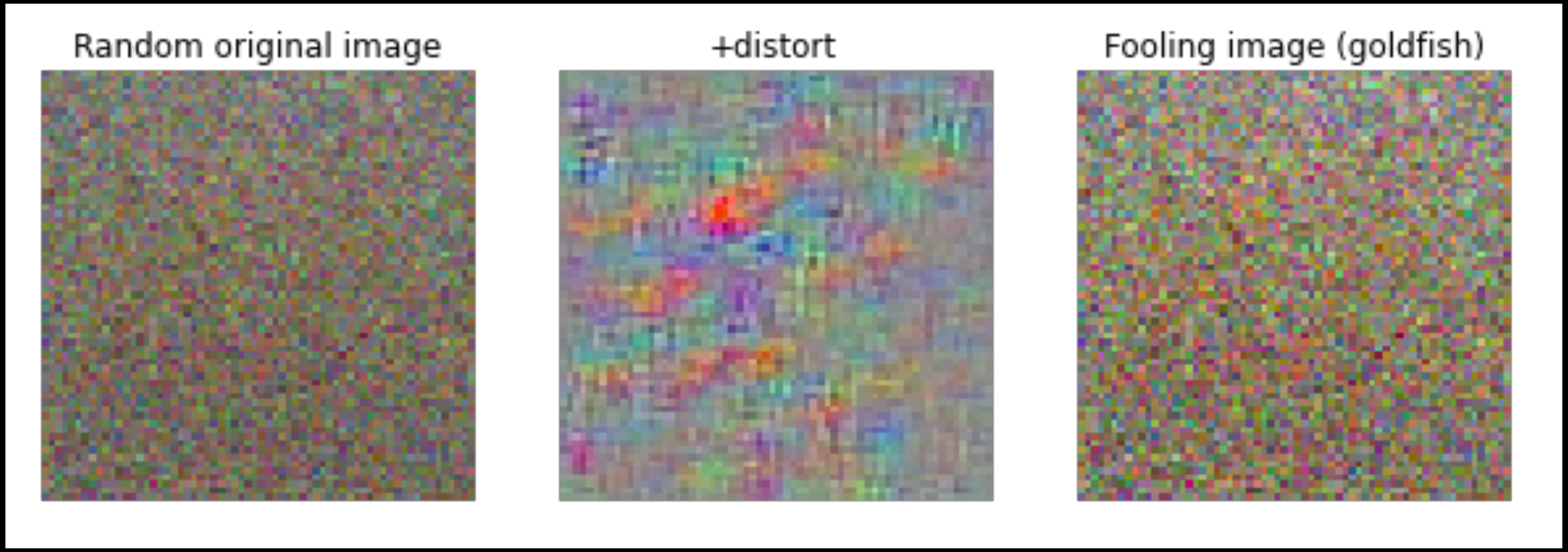
then we do our “magic”

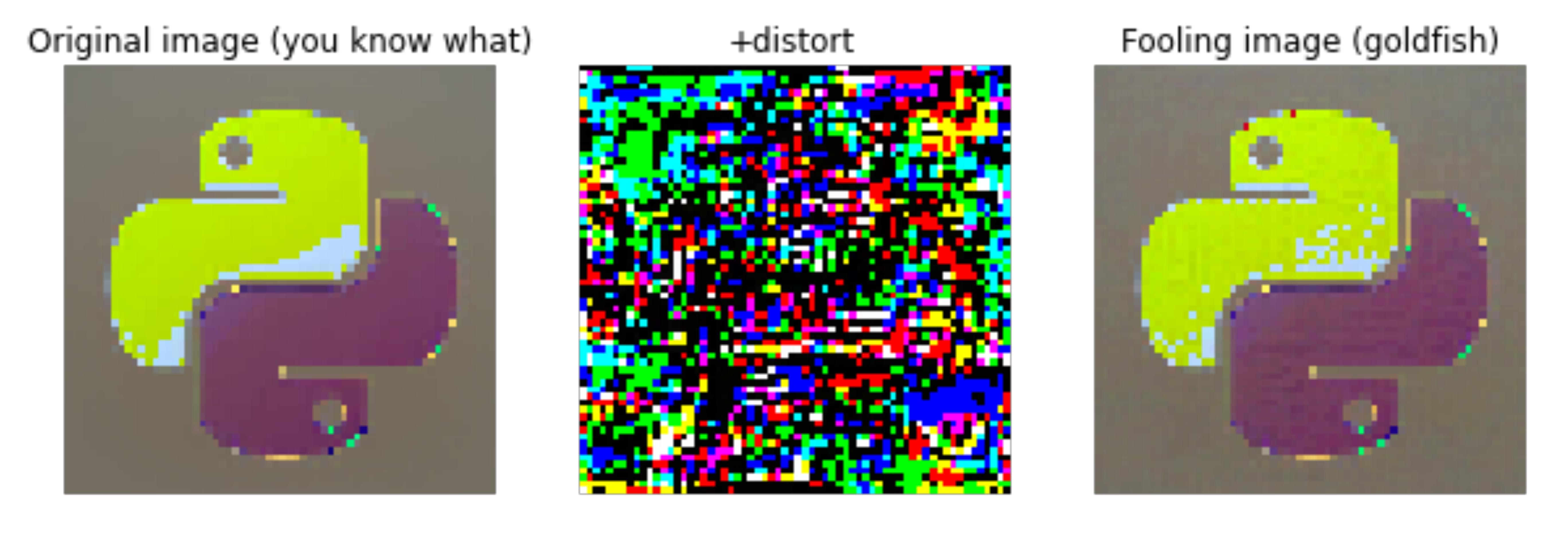


Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint, 2013.
Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images." arXiv preprint

Fooling ConvNets

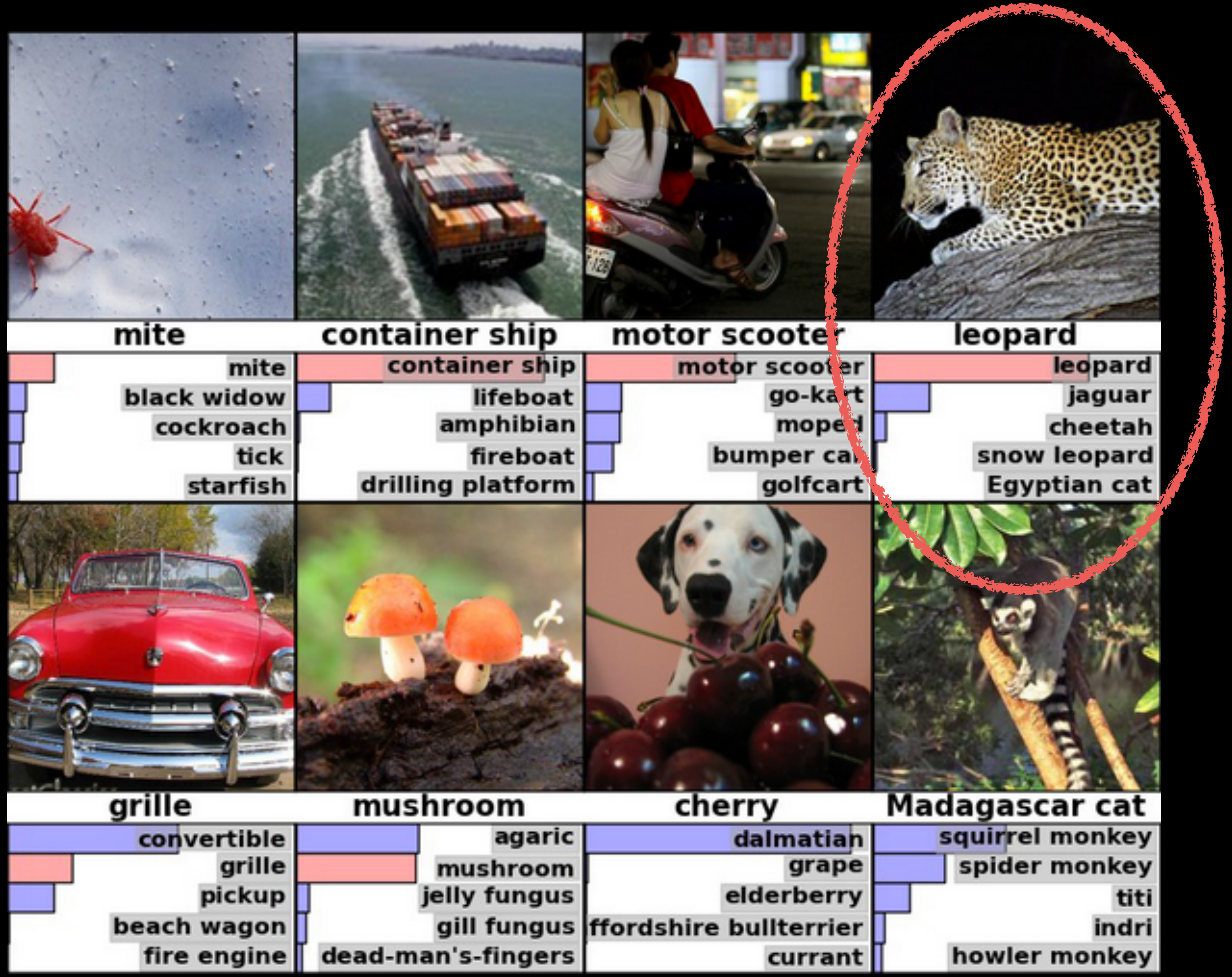
then we do our "magic"



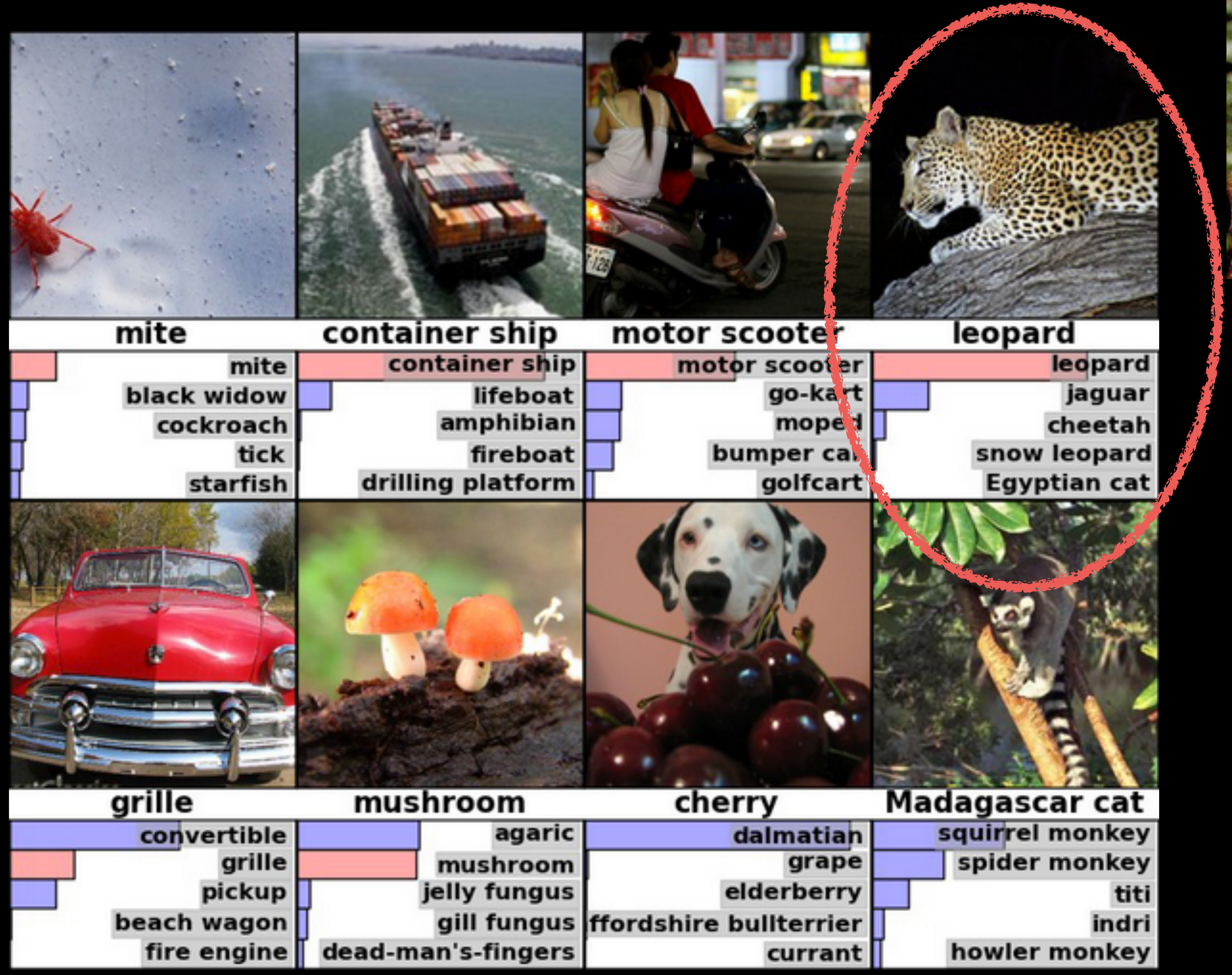


Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint, 2013.
Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images." arXiv preprint

Failing ConvNets



Failing ConvNets

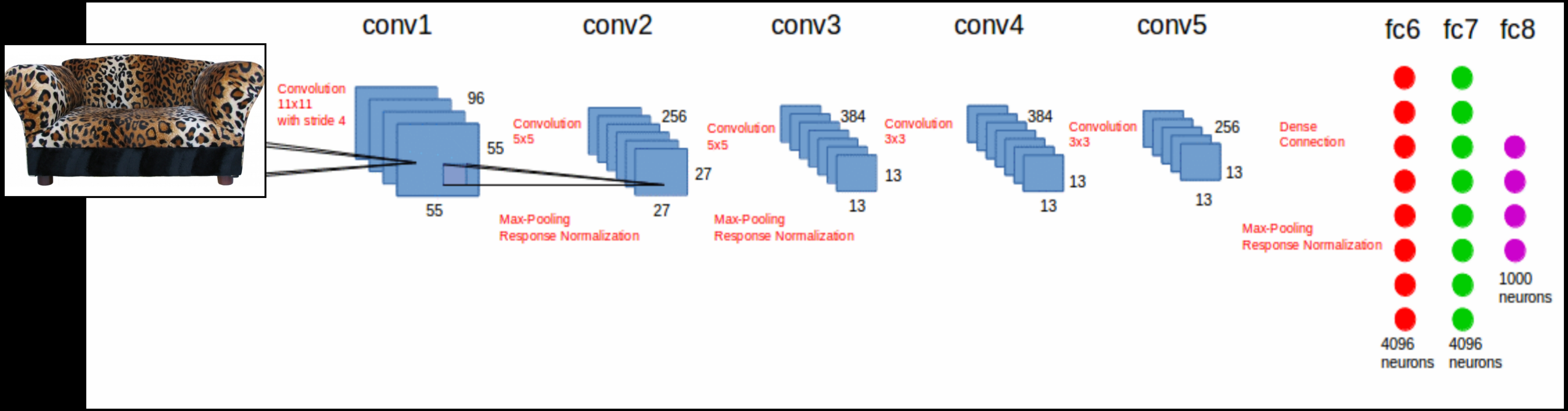


"Suddenly, a leopard print sofa appears", rocknrollnerd.github.io

Failing ConvNets

“Suddenly, a leopard print sofa appears”, rocknrollnerd.github.io

Failing ConvNets

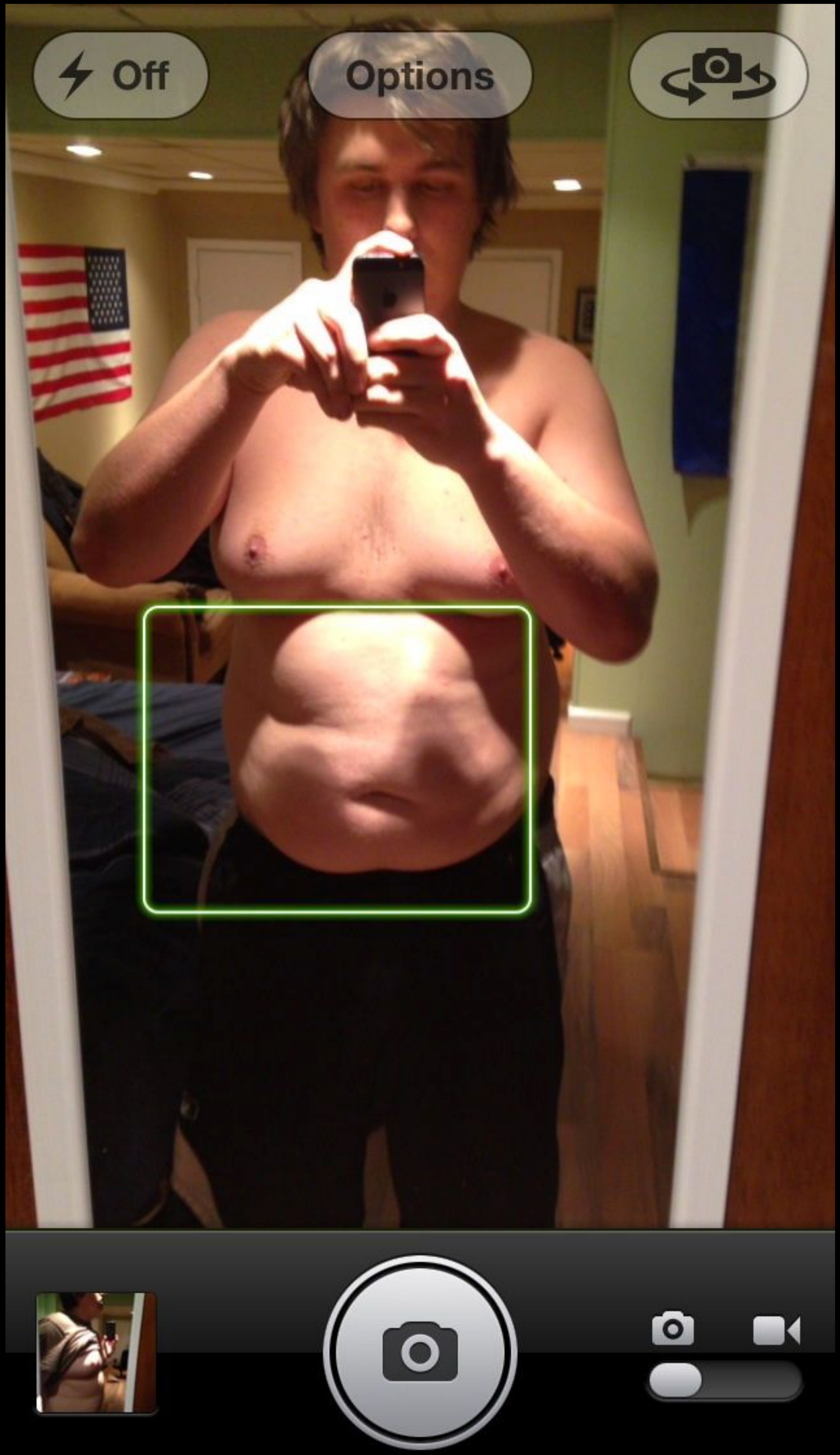



```
>> predicted class: 290
```

```
288 n02128385 leopard, Panthera pardus
289 n02128757 snow leopard, ounce, Panthera uncia
290 n02128925 jaguar, panther, Panthera onca, Felis onca
291 n02129165 lion, king of beasts, Panthera leo
292 n02129604 tiger, Panthera tigris
```

Whoops.

“Suddenly, a leopard print sofa appears”, rocknrollnerd.github.io


Off
Options

thanks for listening ;)

questions?

or find me @graphific

Wanna Play?

- Computer Vision:
  Fei-Fei Li & Andrej Karpathy, Stanford course “Convolutional Neural Networks for Visual Recognition”
  http://vision.stanford.edu/teaching/cs231n
- Natural Language Processing:
  Richard Socher, Stanford course “Deep Learning for Natural Language Processing”,
  http://cs224d.stanford.edu/
- Neural Nets:
  Geoffrey Hinton, Coursera/Toronto, “Neural Networks for Machine Learning"
  https://www.coursera.org/course/neuralnets
- Bunch of tutorials:
  http://deeplearning.net/tutorial/
- Book:
  Yoshua Bengio, et al, “Deep Learning”
  http://www.iro.umontreal.ca/~bengioy/dlbook/
- UFLDL Tutorial
  http://deeplearning.stanford.edu/tutorial/
- Reading Lists:
  http://deeplearning.net/reading-list/
  http://memkite.com/deep-learning-bibliography/
- Podcast
  Talking Machines, http://www.thetalkingmachines.com/