Data Mining In Action Course

Introduction in Deep Learning

Ashuha Arseniy[1,2]

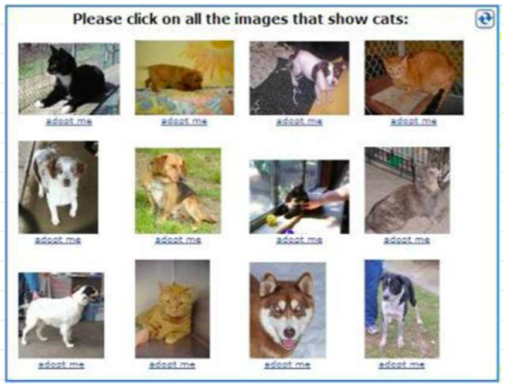
Bayesian Research Group[1], MIPT[2]

ars-ashuha.ru/slides

December 20, 2016

Cat vs Dogs

2006



Computer vision = 60%

$0.6^{12} = 0.00217$

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

$0.989^{12} = 0.875$

Microsoft Research

Our research Connections Careers About us

All Downloads Events Groups News People Projects Publications

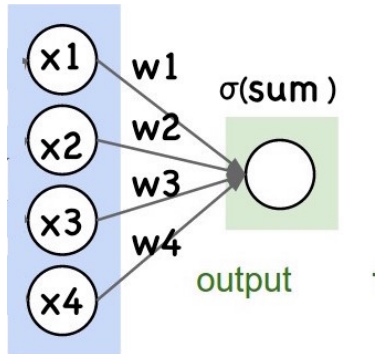
ASIRRA

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

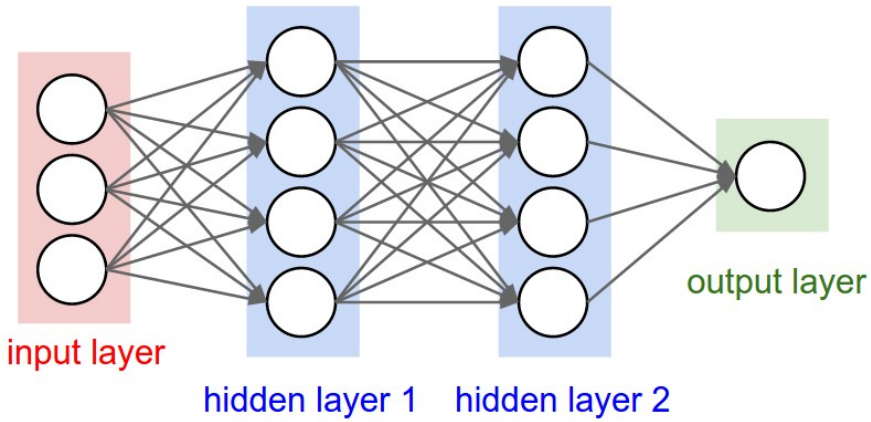
slides credit: prof.V. Lempitsky

Motivation

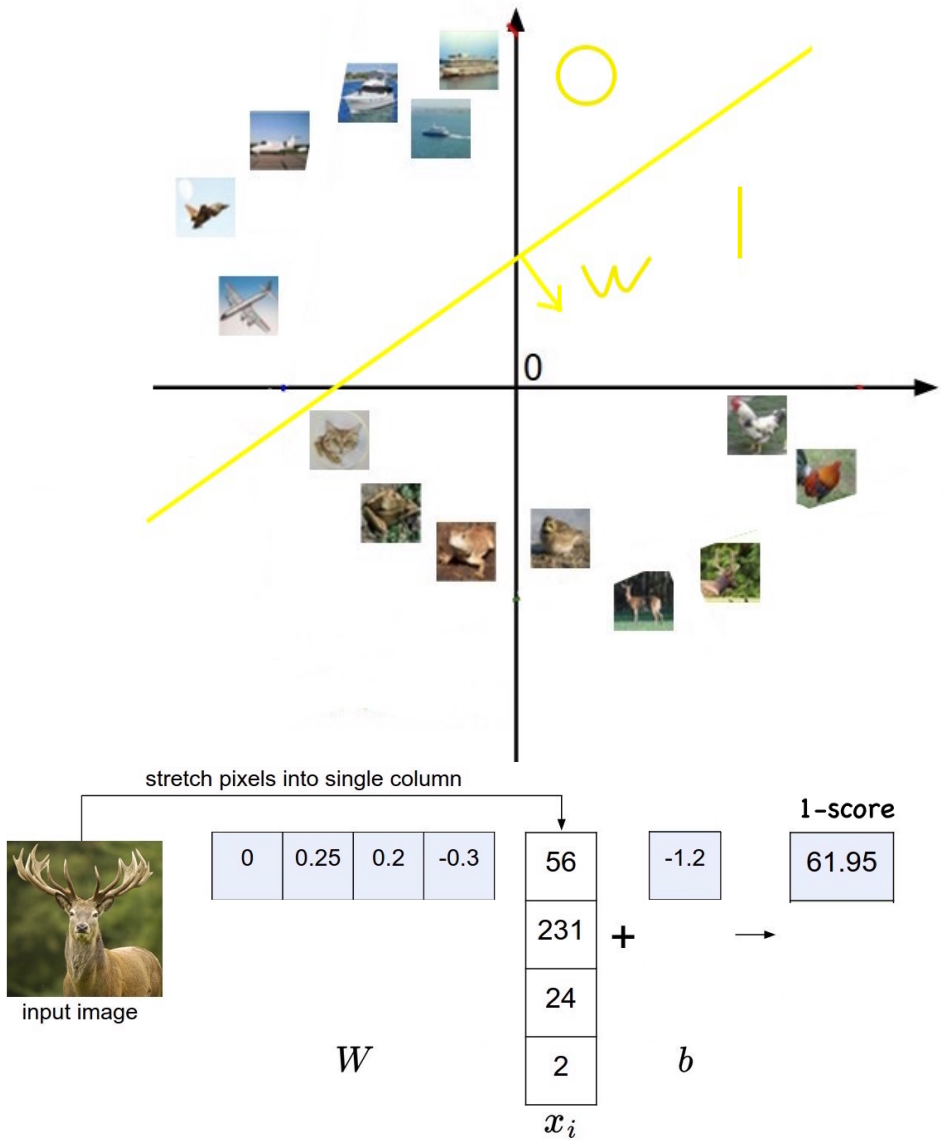
- We know that linear model is awesome
- Linear Models



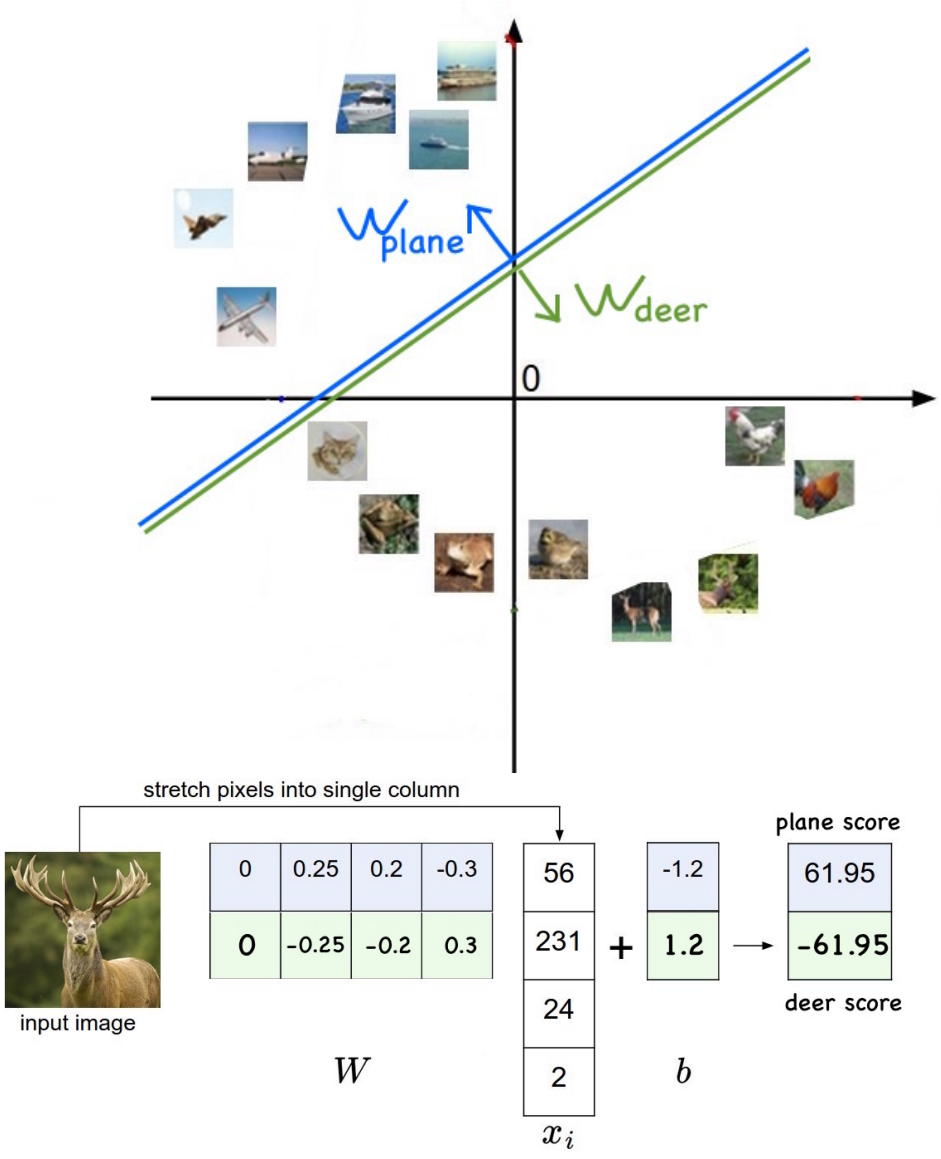
- But LM describe our data bad course linearity
- Neural Nets



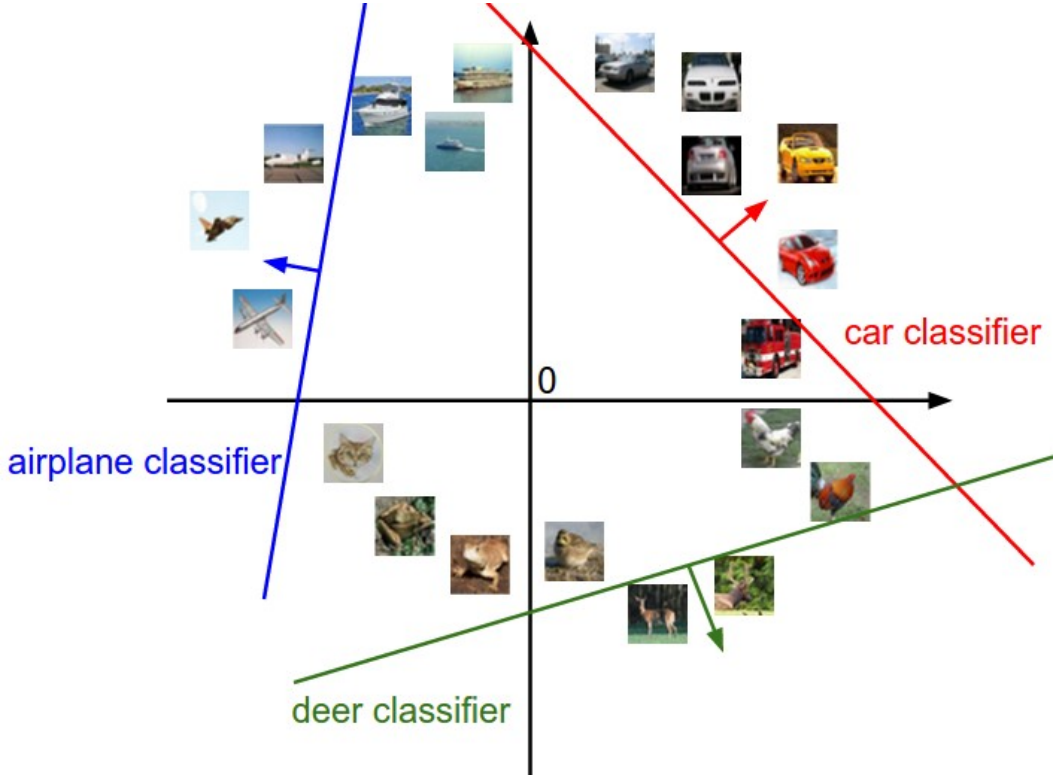
Linear Models

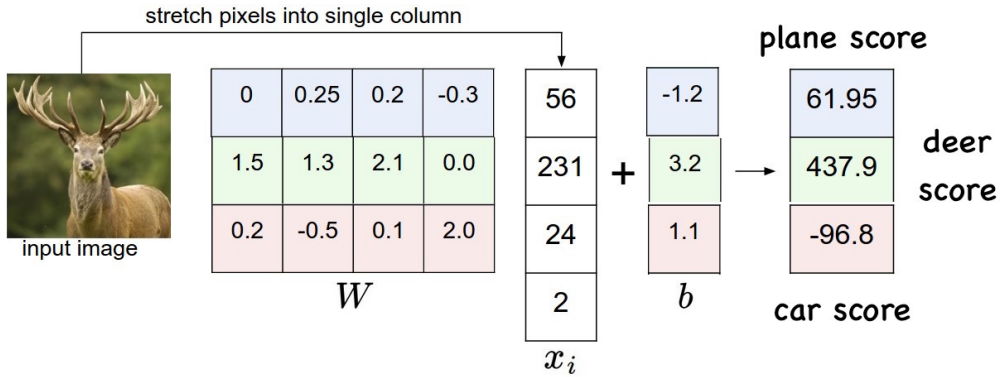


Linear Models



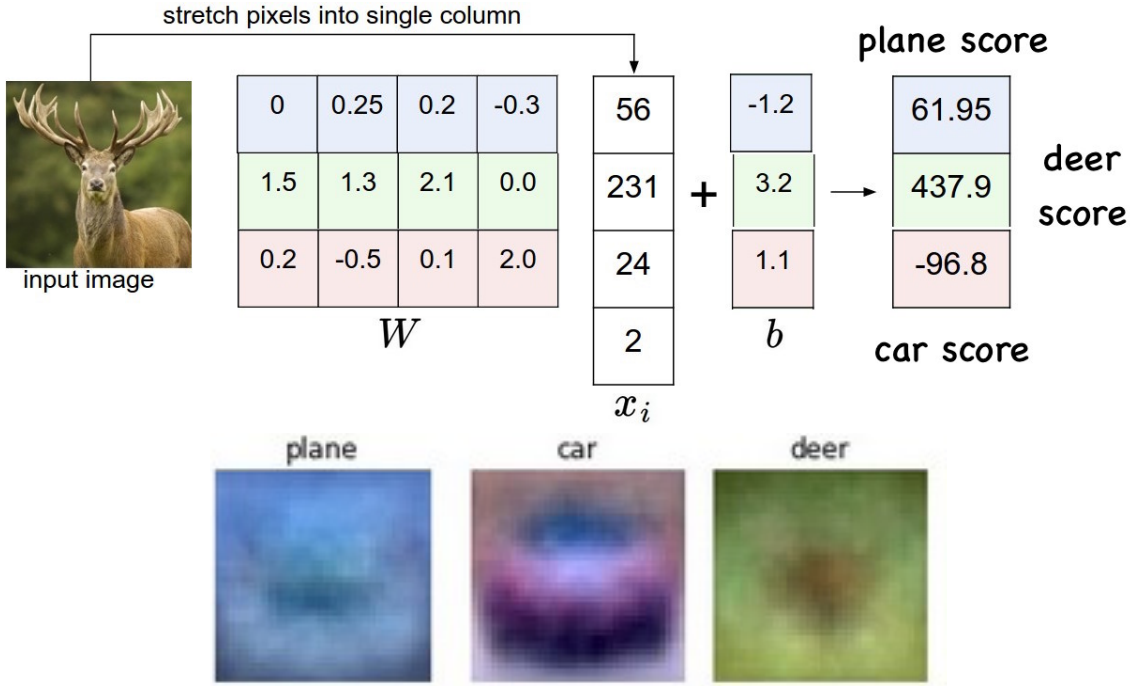
Linear Models





http://vision.stanford.edu/teaching/cs231n/linear-classify-demo/

Linear Models Plot weights



- add more classifiers

$$score(x) = W_2 \cdot (W_1 \cdot x + b_1) + b_2 = W \cdot x + b$$

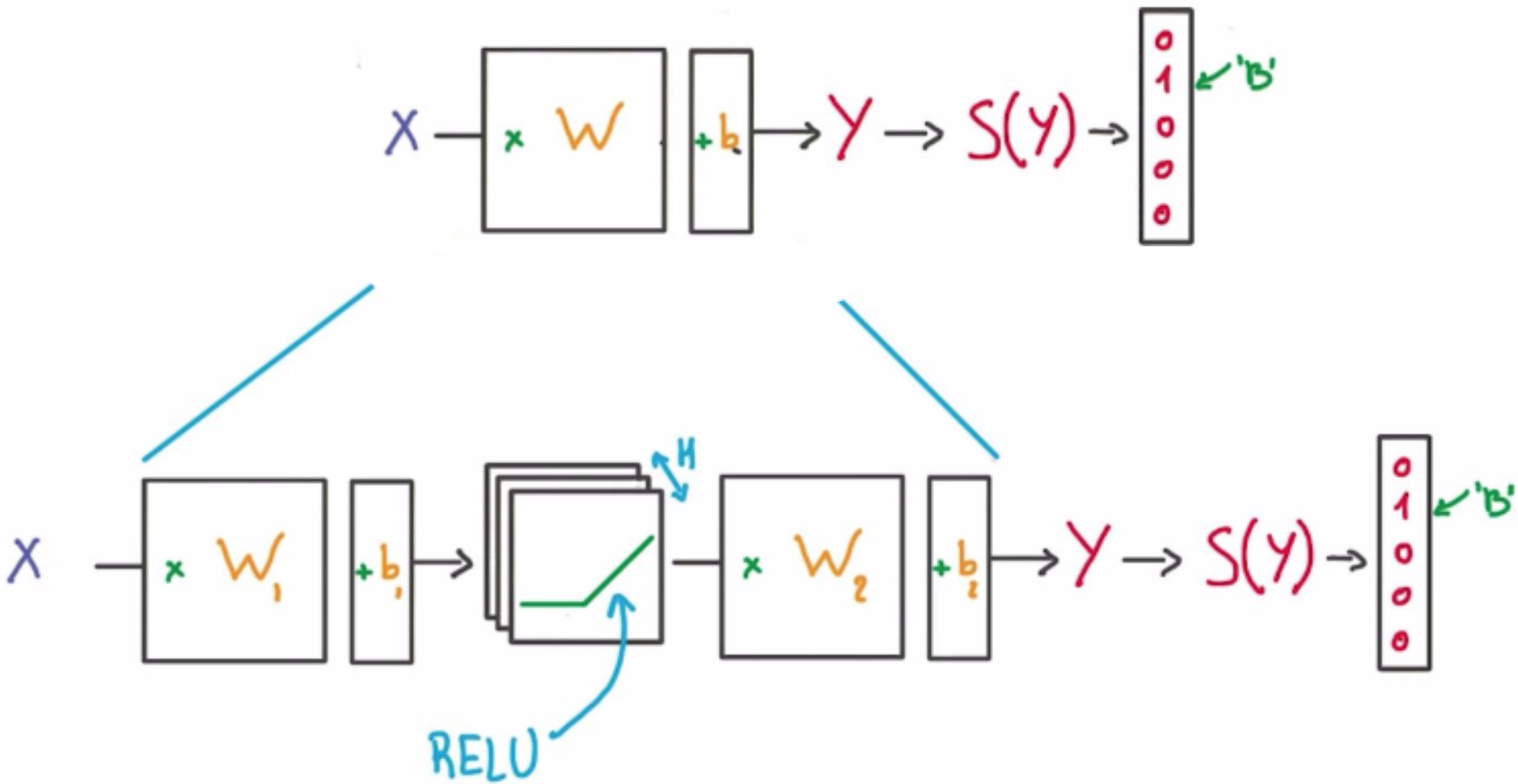
  it was a bad idea ☹

- let's add non-linearity

$$score(x) = W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2 \neq W \cdot x + b$$

  so we made our model more complex ☻

Neural Nets

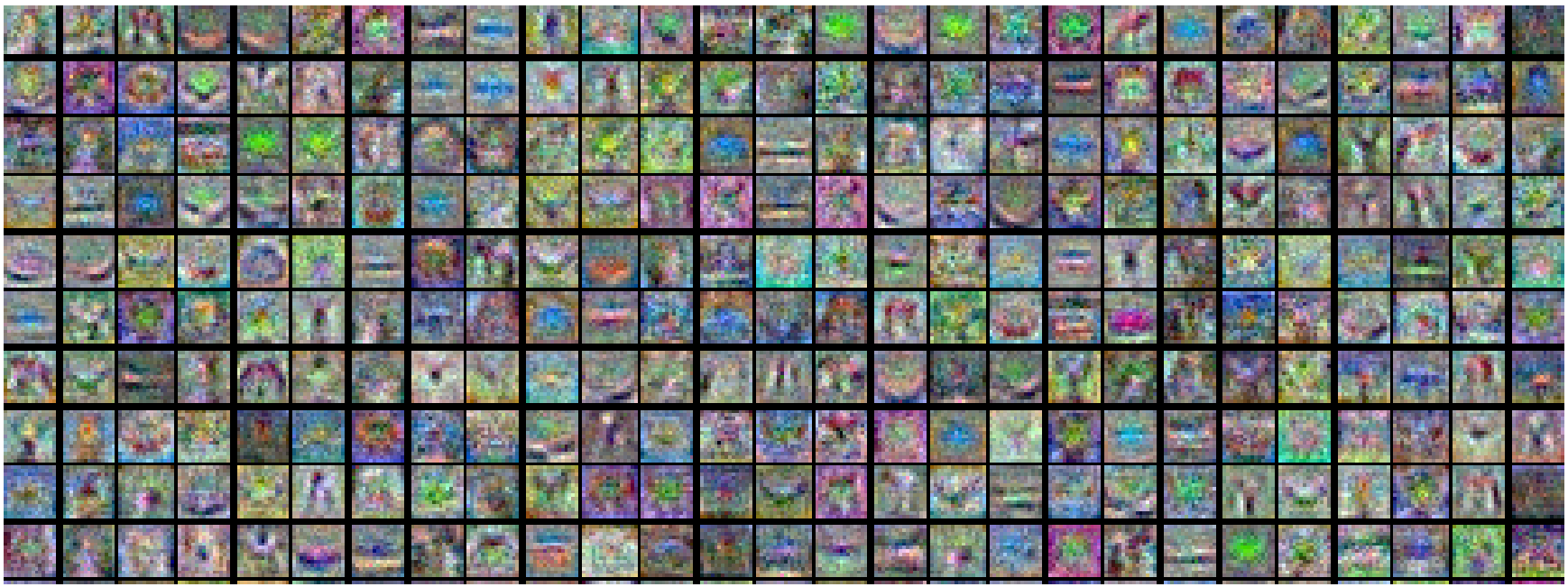


- Back Propagation Idea, evaluate loss gradient by W_1 through W_2
- Layer with linear transformation usually calls Dense or Fully Connected
- linear model vs non-linear vs deep nonlinear

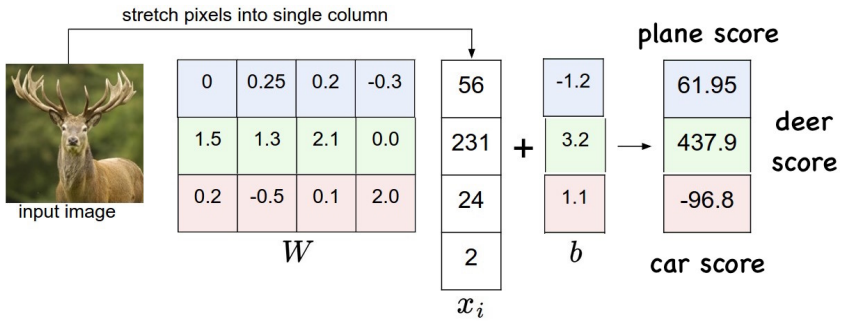
Recap

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

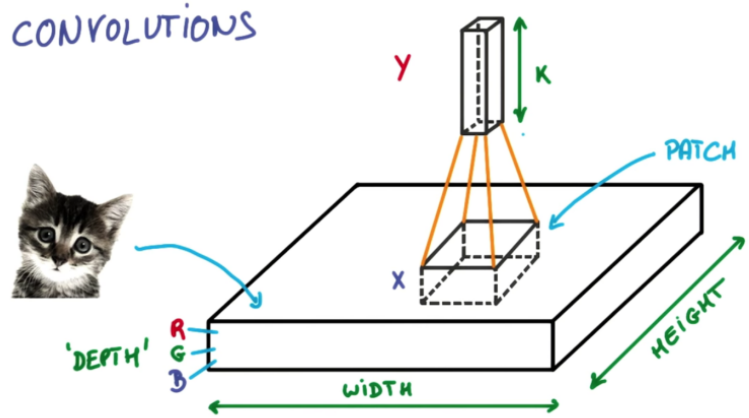
- Now we got a lot of feature extractor in matrix W_1 we call it Fist Layer



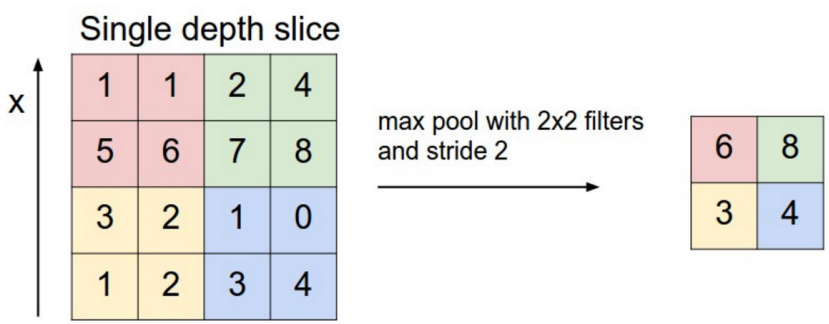
Layers



(a) Dense

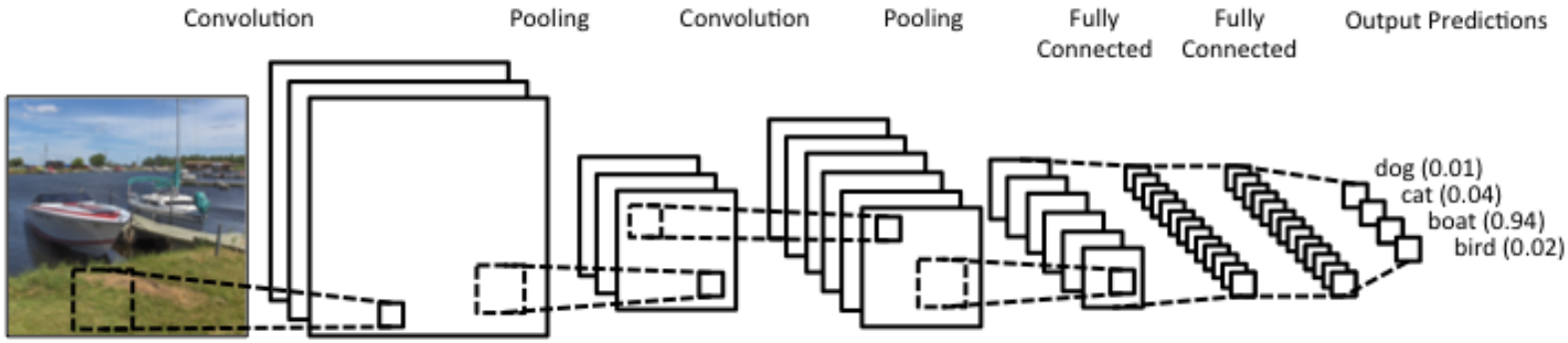


(b) Convolution

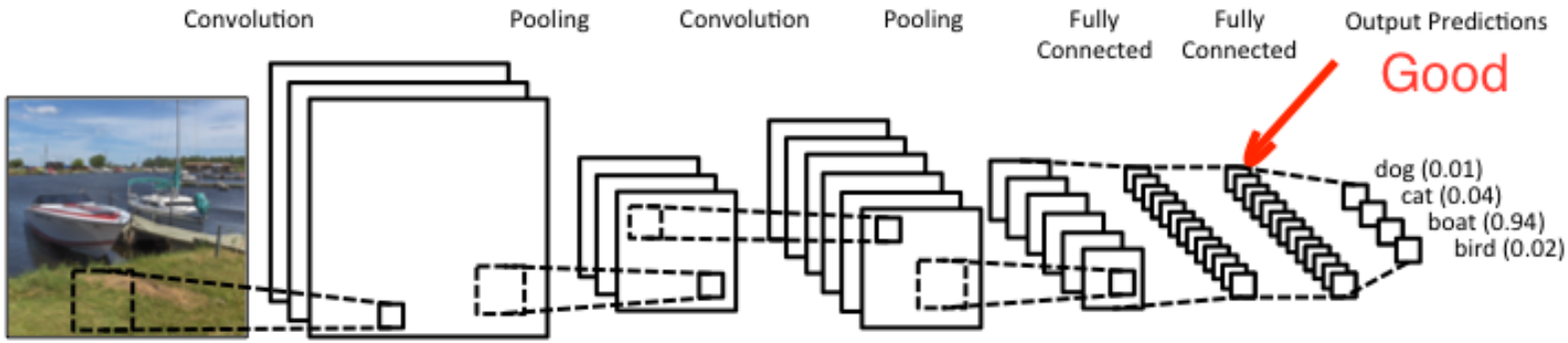


Conv Neural Nets

- Too many parameters for each deer position
- We can evaluate local features
- Convolution animation
- Convolution Neural Net



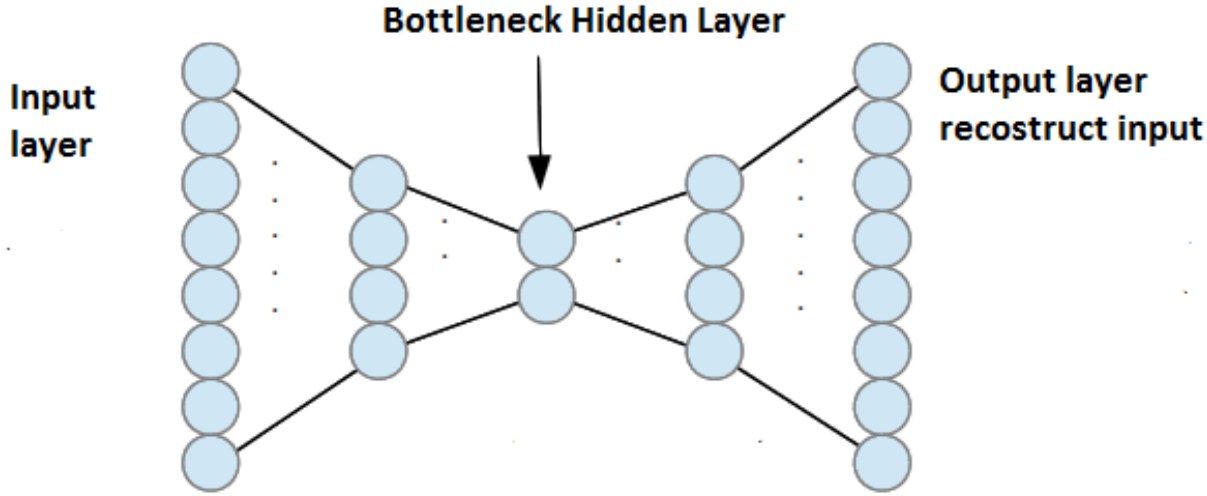
- Good Representation



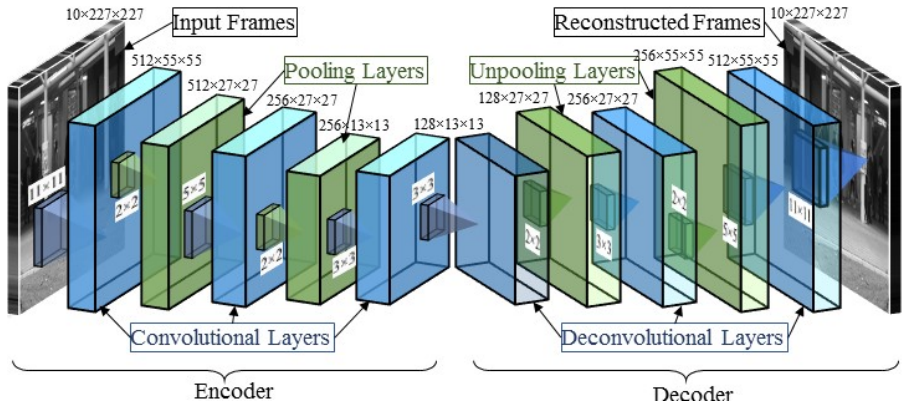
- http://cs.stanford.edu/people/karpathy/cnnembed/

Aauto-encoders

- What you should do if there are no training labels?
- We still want to understand smth about images?
- Fully Contented Auto-encoder

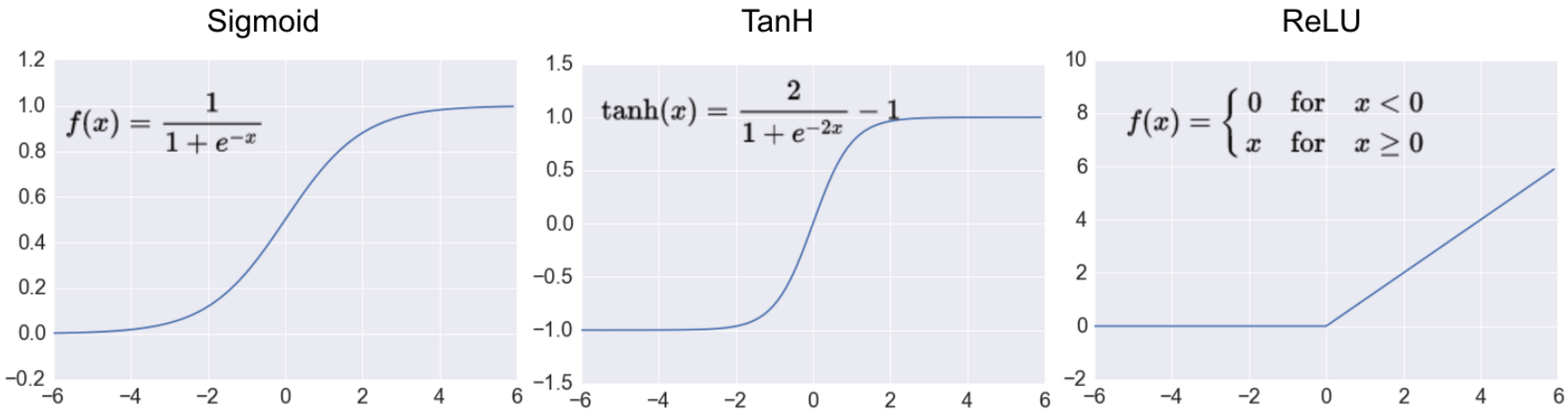


- Convolution Auto-encoder

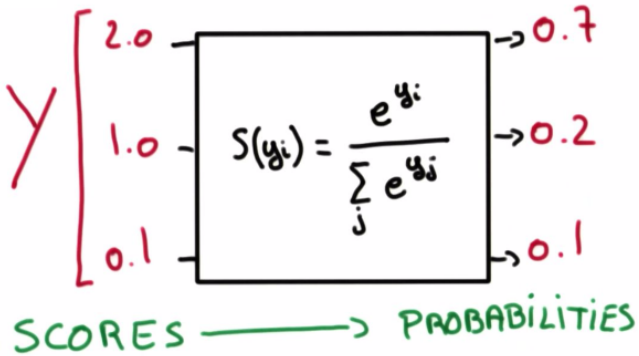


Nonlinearities

- Activation function



- From scores to probabilitys – SoftMax



- Large list is available hear lasagne.nonlinearities

Loss Function

- For regression task and Auto-encoders you can use MSE

$$L(\mathbf{y}, \hat{\mathbf{y}}) = (\mathbf{y} - \hat{\mathbf{y}})^2.mean()$$

- For classification Cross-Entropy or LogLoss

$$LogLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

$$CE = -\sum_i y_i \log \hat{y}_i$$

- For classification Hinge-Loss or SVM ☻

$$SVM = \quad max(0, -y \cdot \hat{y} + c)$$

$$Hinge = \sum_{i \neq y_i} max(0, \hat{y}_i - \hat{y}_{y_i} + c)$$

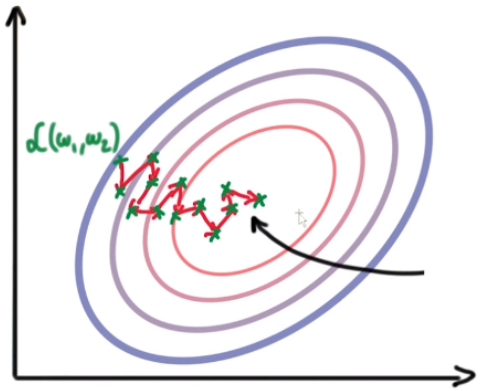
- You can make loss by yourself

Optimizers

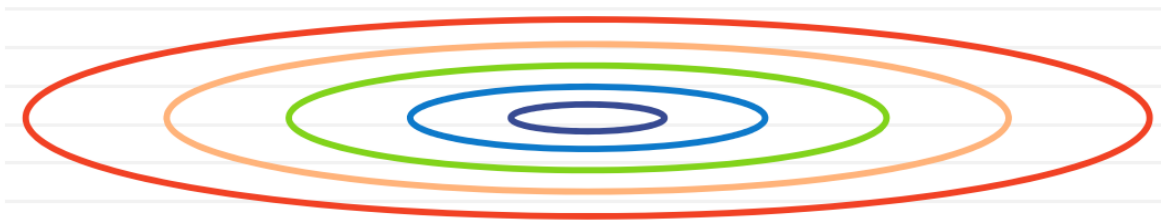
- Stochastic Optimization, we cant eval our function and it's grad
- Stochastic Gradient Decent, so we will compute unbiased estimation

$$w = w - \alpha \frac{\partial \hat{L}}{\partial w}$$

- Stochastic Gradient Decent with Momentum



- Stochastic Gradient Decent with Momentum and Rescaling (adam)

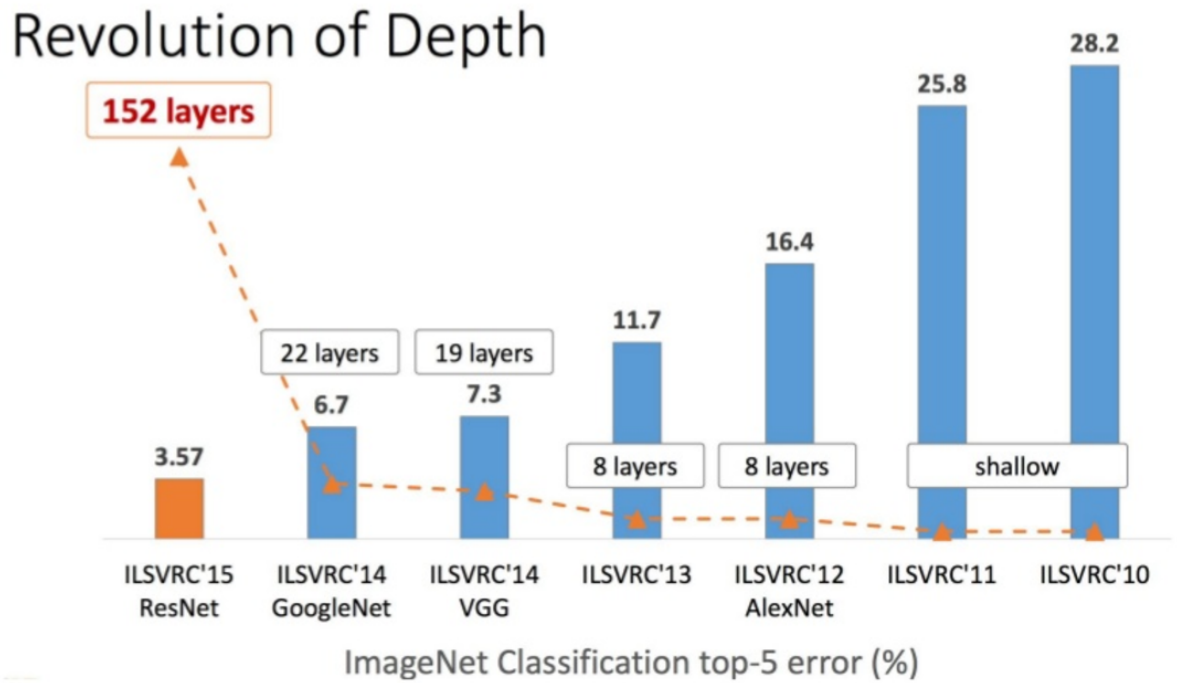


contours evaluation optimizers, saddle point

ImagNet Challenge

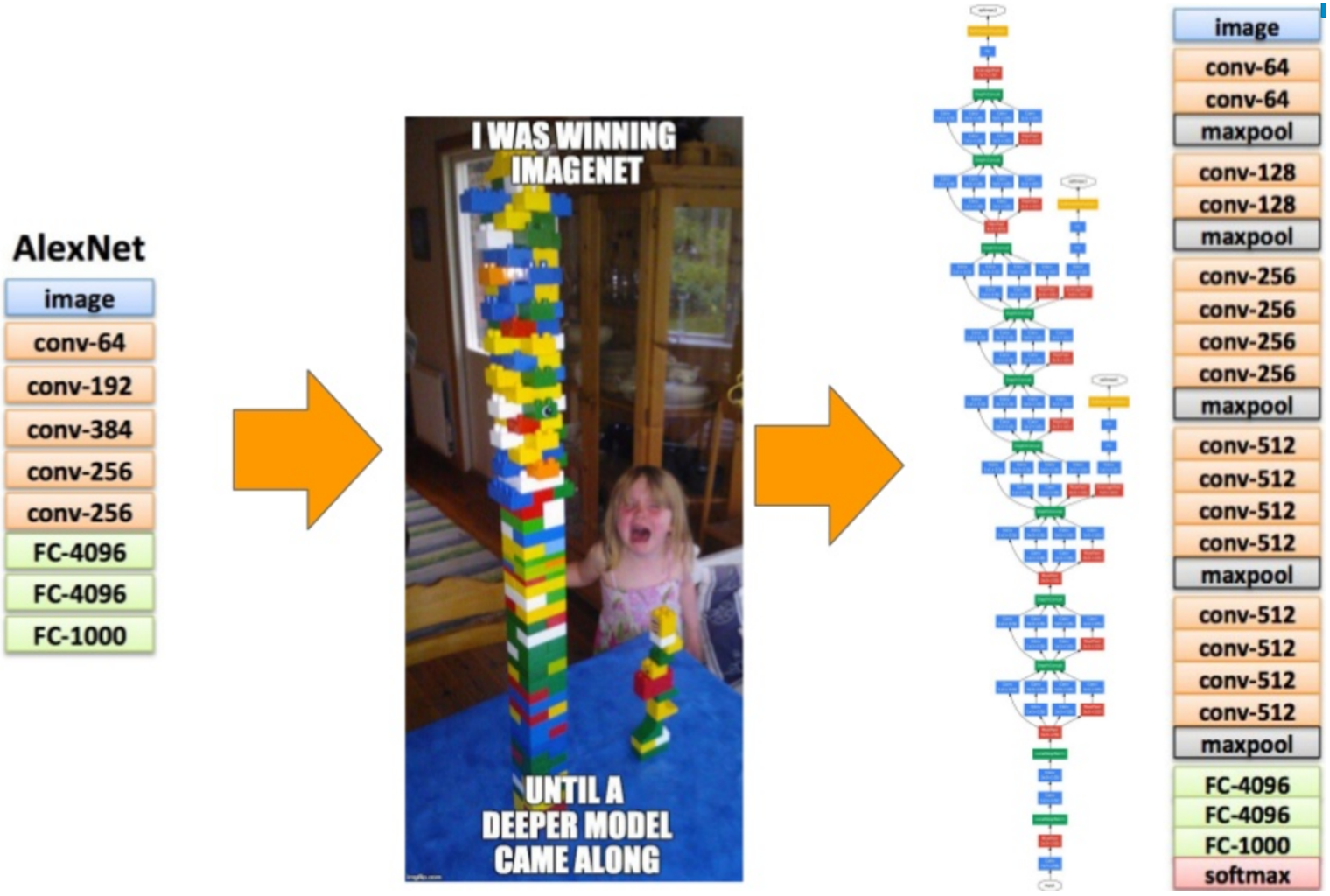
- Imagnet

- Errors-Number of Layers Plot vs Time

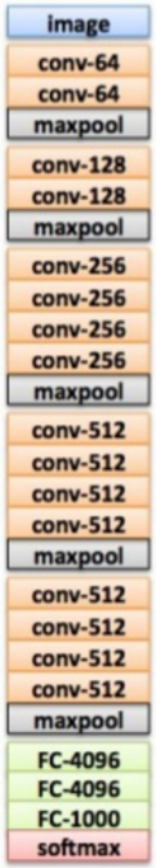


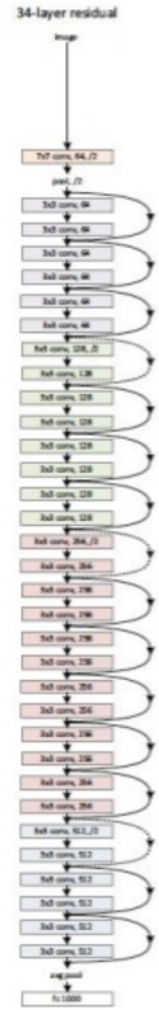
plot credit: Kaiming He

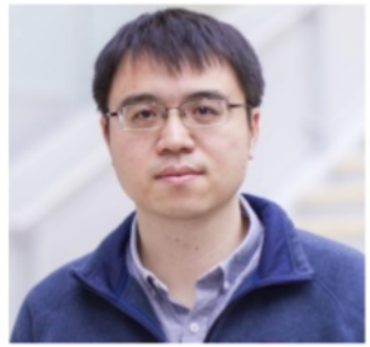
Popular architectures



Popular architectures







Microsoft Research

3.6% top 5 error…
with 152 layers !!

Code should be like this

```
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()

params = ll.get_all_params(nn, trainable=True)
updates = lasagne.updates.adam(loss, params)
```

Recap

Recap:

- Deep Architectures are cool
- Use Pre-trained Nets and Fine-tuning for your tasks
- Old math doesn't work if a lot of data

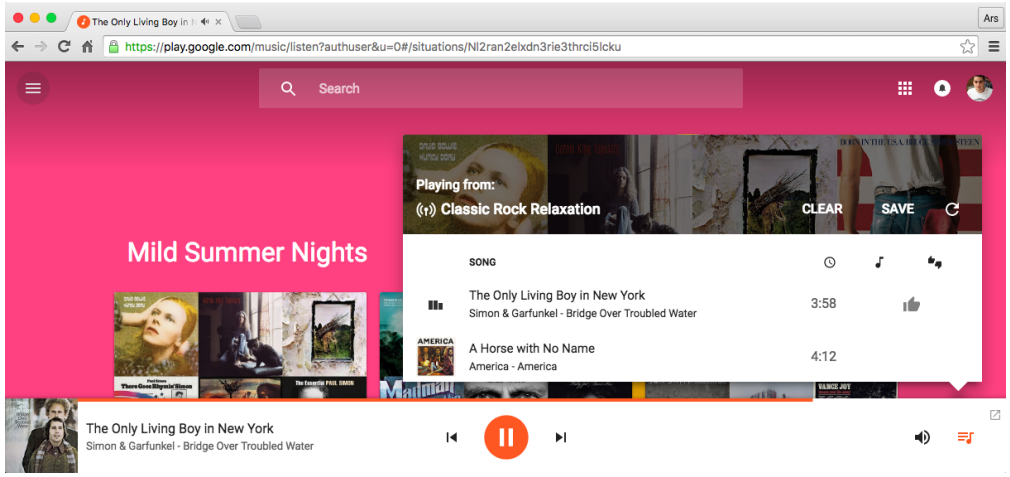
Good courses/books

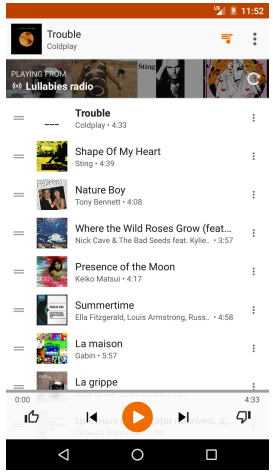
- https://github.com/Lasagne/Recipes/tree/master/modelzoo
- cs231n.stanford.edu
- udacity.com/course/deep-learning–ud730
- deeplearningbook.org

Current Status of your Field!

How to apply ML for Music Data to get Money?

- Your are working in a big music service as a data scientist





- In this service there's a lot of music data – mp3 files

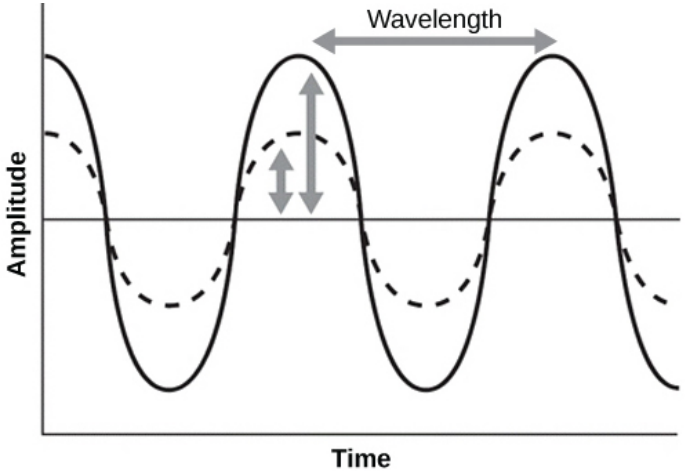
user_id	tracks_id
123	[1, 2, 3]
124	[1000, 11, 23, 23]
...	...
999999	[1]

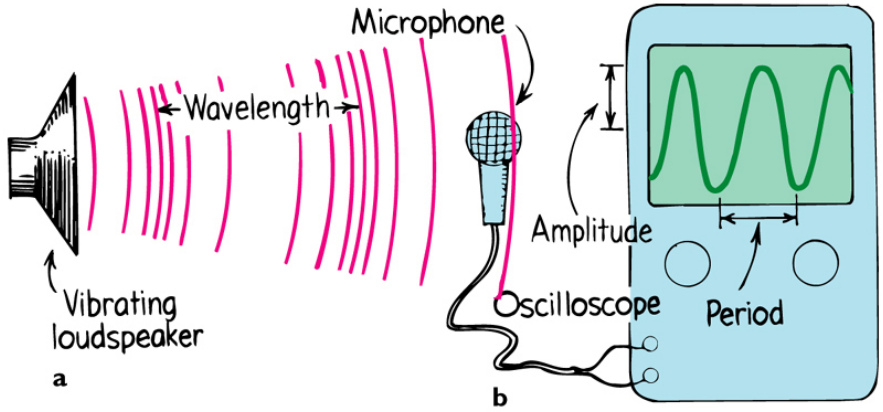
tracks_id	file
1	1.mp3
2	2.mp3
...	...
999999	999999.mp3

- You were given the task – make money using this data

What is the sound?

- Waves and Recording





- How to store sound? Store as big-big array with sampling frequency

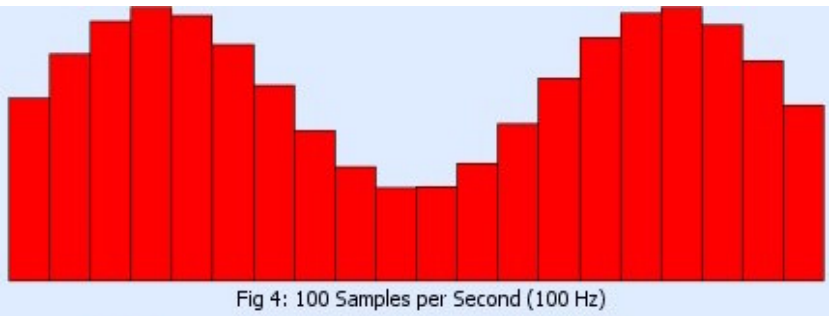


Fig 4: 100 Samples per Second (100 Hz)

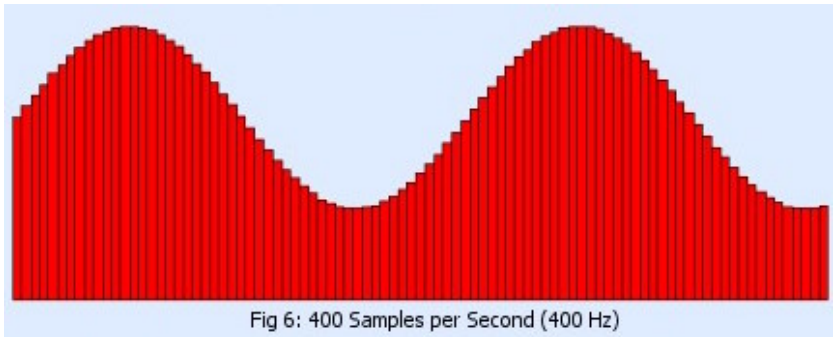


Fig 6: 400 Samples per Second (400 Hz)

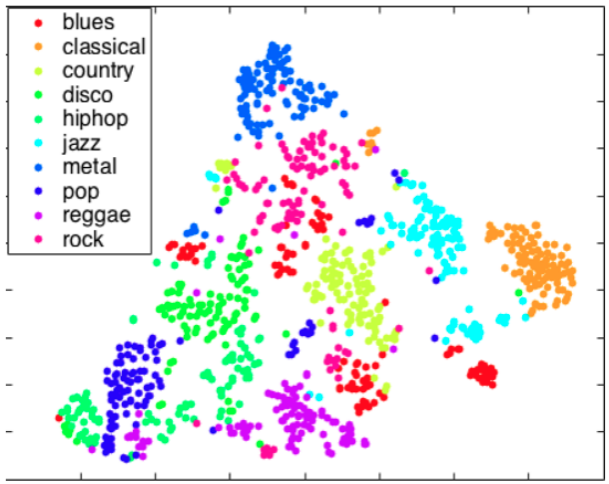
- $[1, 2, 3, 5, 3, 2, 1, 1, 1, 1, 2, 3, 5, 3, 2]$, Usually 16 000 float per second

Finding similar tracks

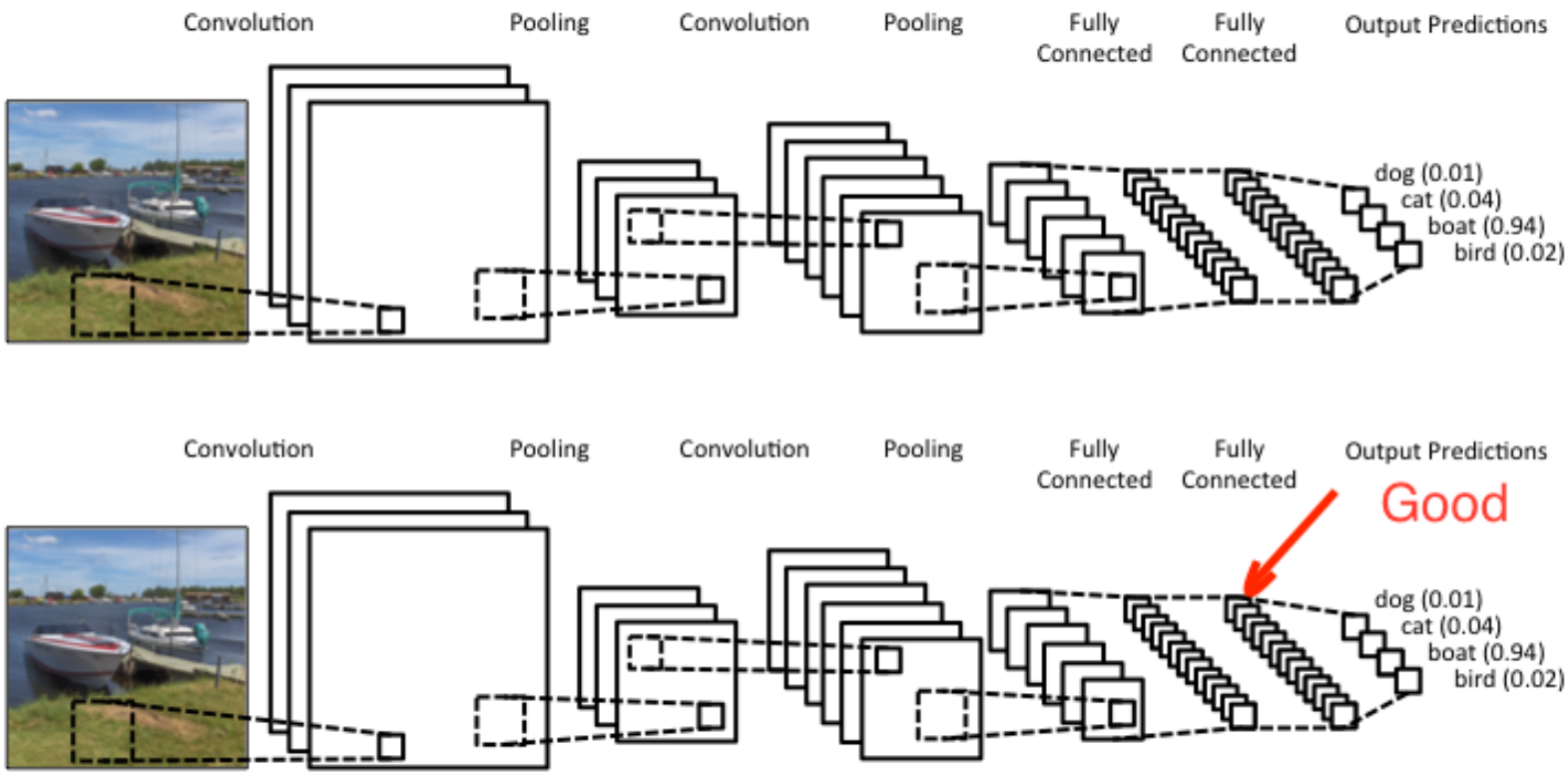
- How to find similar tracks using ML methods?

  Data: 30 sec * 16000 features, 10^7 items

  Task: define function of $similarity(track_i, track_j)$
- Why ordinary methods are so bad?
  - shift and noise tolerance, over-fitting
- Metric approach is still good idea, if we have a high level description
- Good representation of music track
  - Human – guitar, rock, Queen, 1997, UK, 3 min., ...
  - Computer – good small vector of numbers



Get good representation using Neural Nets

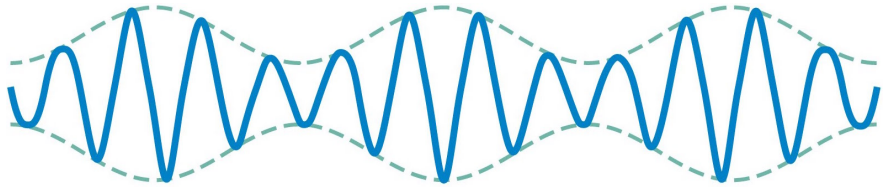


Problem

We need to get picture!

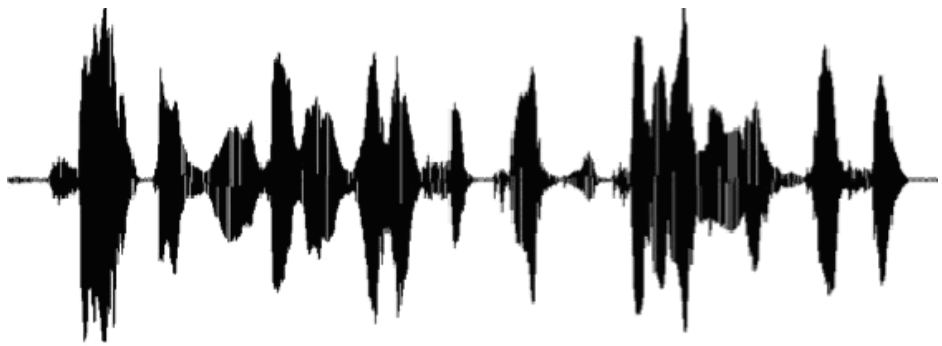
What is the sound part 2?

We have some wave



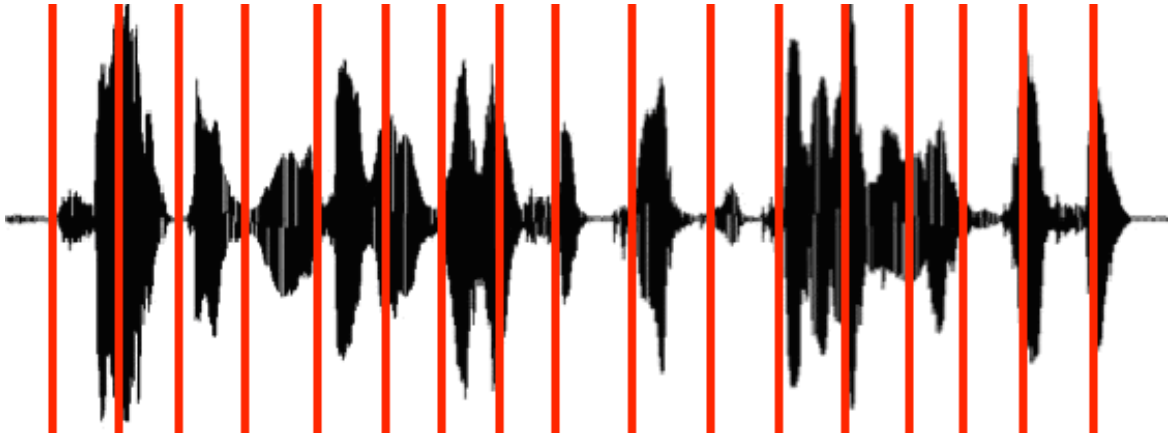
represent wave as a sum of two waves

sound is a combination of big waves range



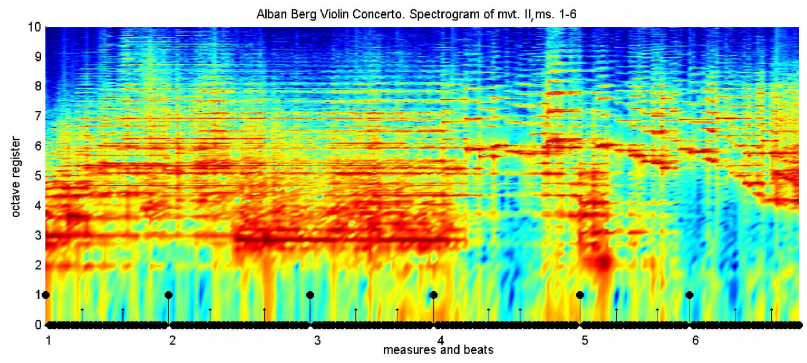
What we lost in our representation?

What is the sound part 2? Get Frequency



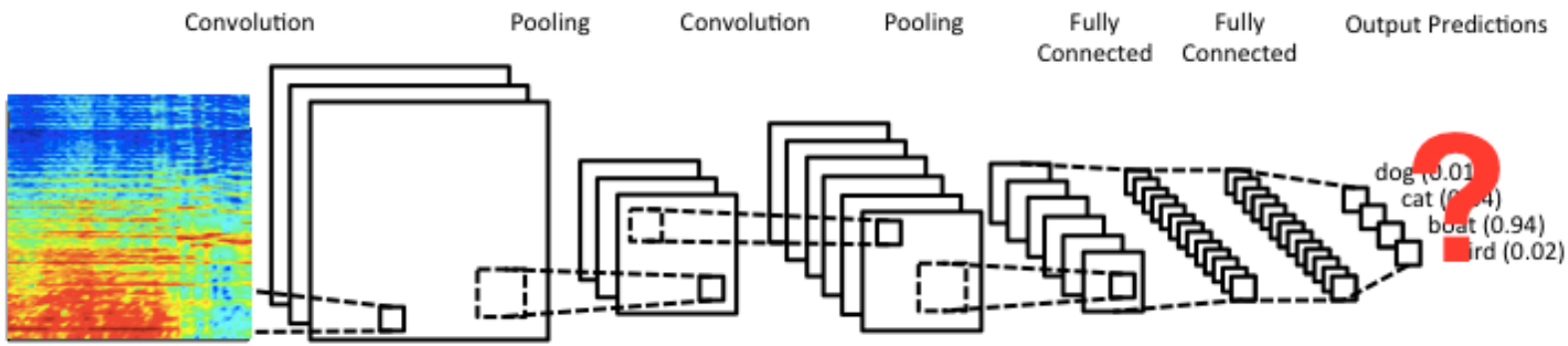
High Freq	1	2	1	1	2	1	1	2	1	1	2	1	1	2	1	1	2	1
Mid Freq	2	1	2	2	1	2	2	1	2	2	1	2	2	1	2	2	1	2
Low Freq	1	2	2	1	2	2	1	2	2	1	2	2	1	2	2	1	2	2

High Freq	1	2	1	1	2	1	1	2	1	1	2	1	1	2	1	1	2	1
Mid Freq	2	1	2	2	1	2	2	1	2	2	1	2	2	1	2	2	1	2
Low Freq	1	2	2	1	2	2	1	2	2	1	2	2	1	2	2	1	2	2



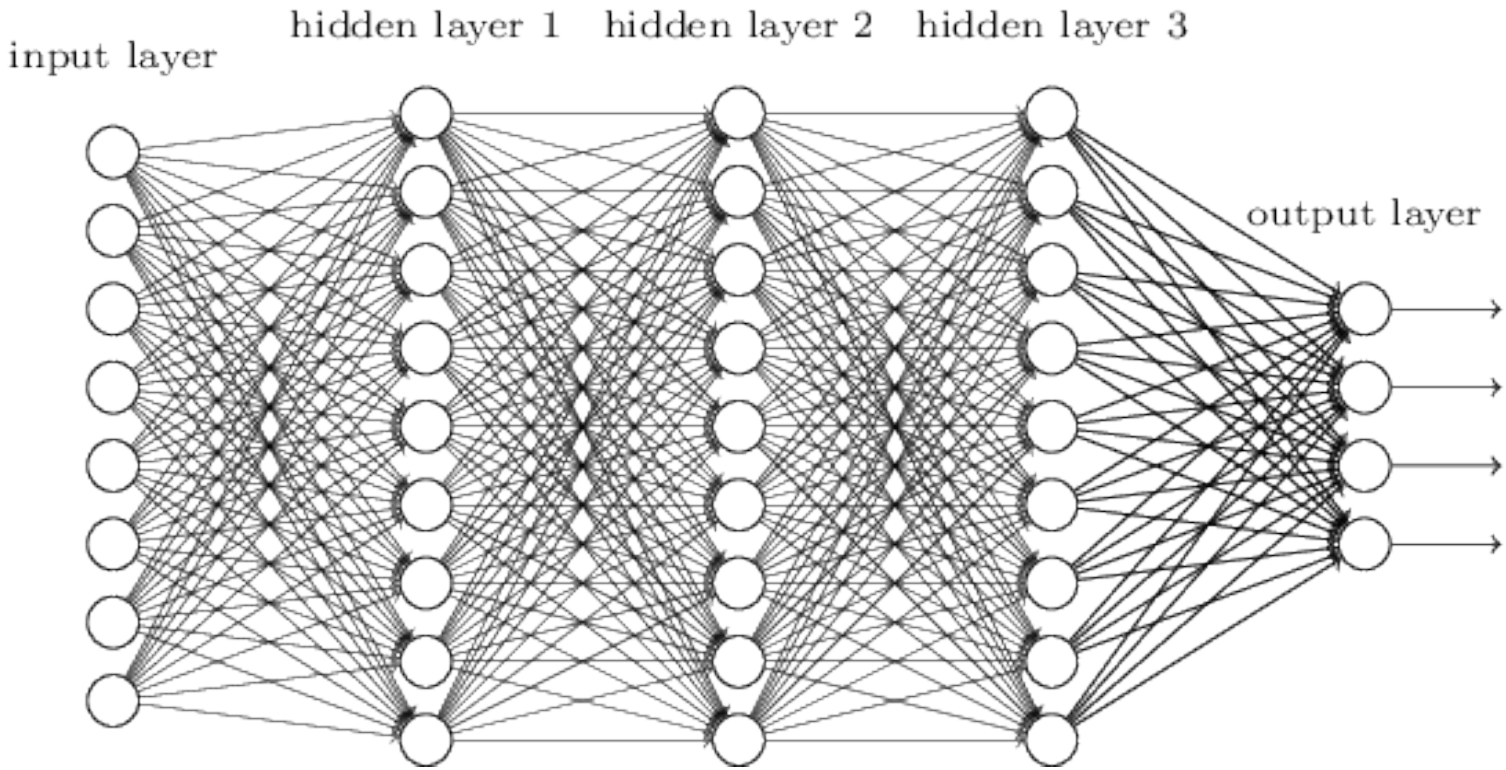
We need to train Neural Nets, but how can we do that?!

- But how can we train nets on music?
- Let's invent a fake machine learning task



  - genre classification
  - artist classification
  - rating prediction
  - .....

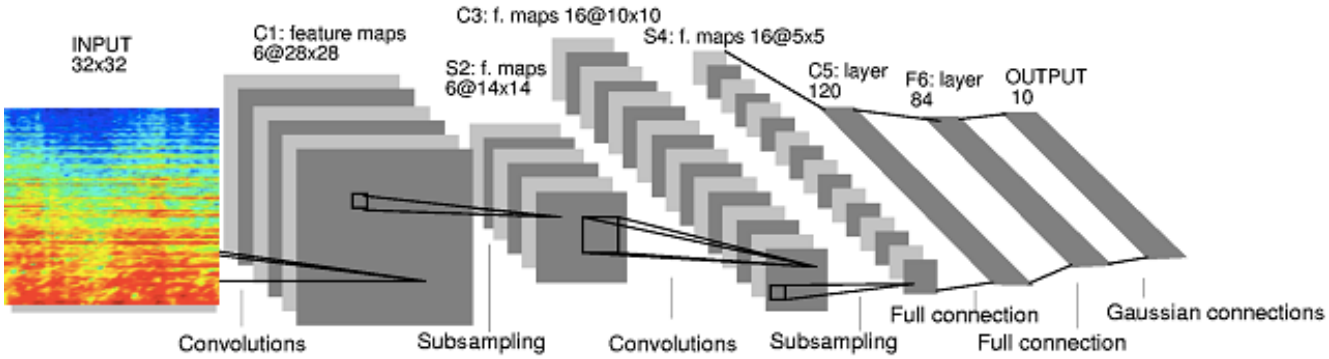
Fully connected NN



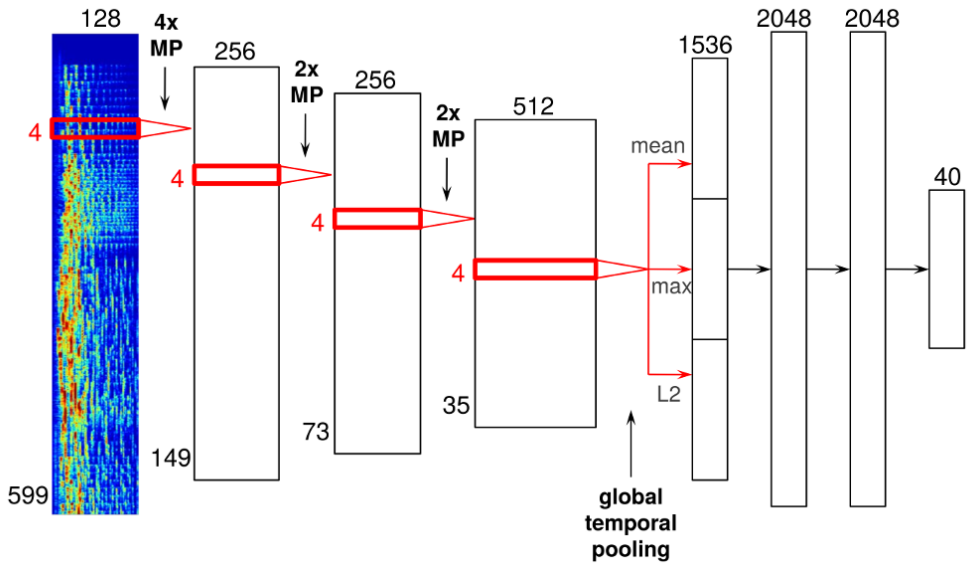
- too many parameters – number of weights $= 16^4 * neurons + ...$
- It doesn't work =(

Convolution NN

Let's invent some convolution architecture



important detail – pooling of time axis [Spotify))) Deep Learning]



How to measure quality of good representation?

What we have?

- We have represented each track as a vector
- But maybe our solution is too bad, how can we understand that?
- How to test "good representation"?

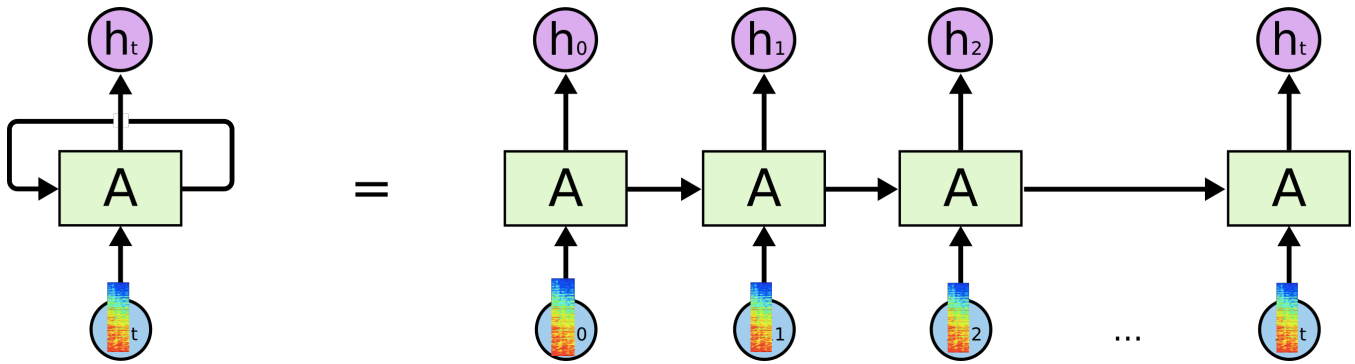
Let's invent the metrics:

- by hand
- using assessors
- recommendation quality
- using vectors to classify another labels

Let's adapt to Different length and Additional information

How to use any length?:

1. Average prediction for many patches
2. Recurrent neural net on many patches



3. Whatever?

How to take account?:

1. Lyrics

$$\text{Concat}(\text{TextRNN}, \text{Conv}) \rightarrow \text{FC} \rightarrow \text{Cost}$$

2. Genre, Artist, Year – embedding too, multi-cost task
3. ....

Current Status of your Field!

Thanks for your attention!