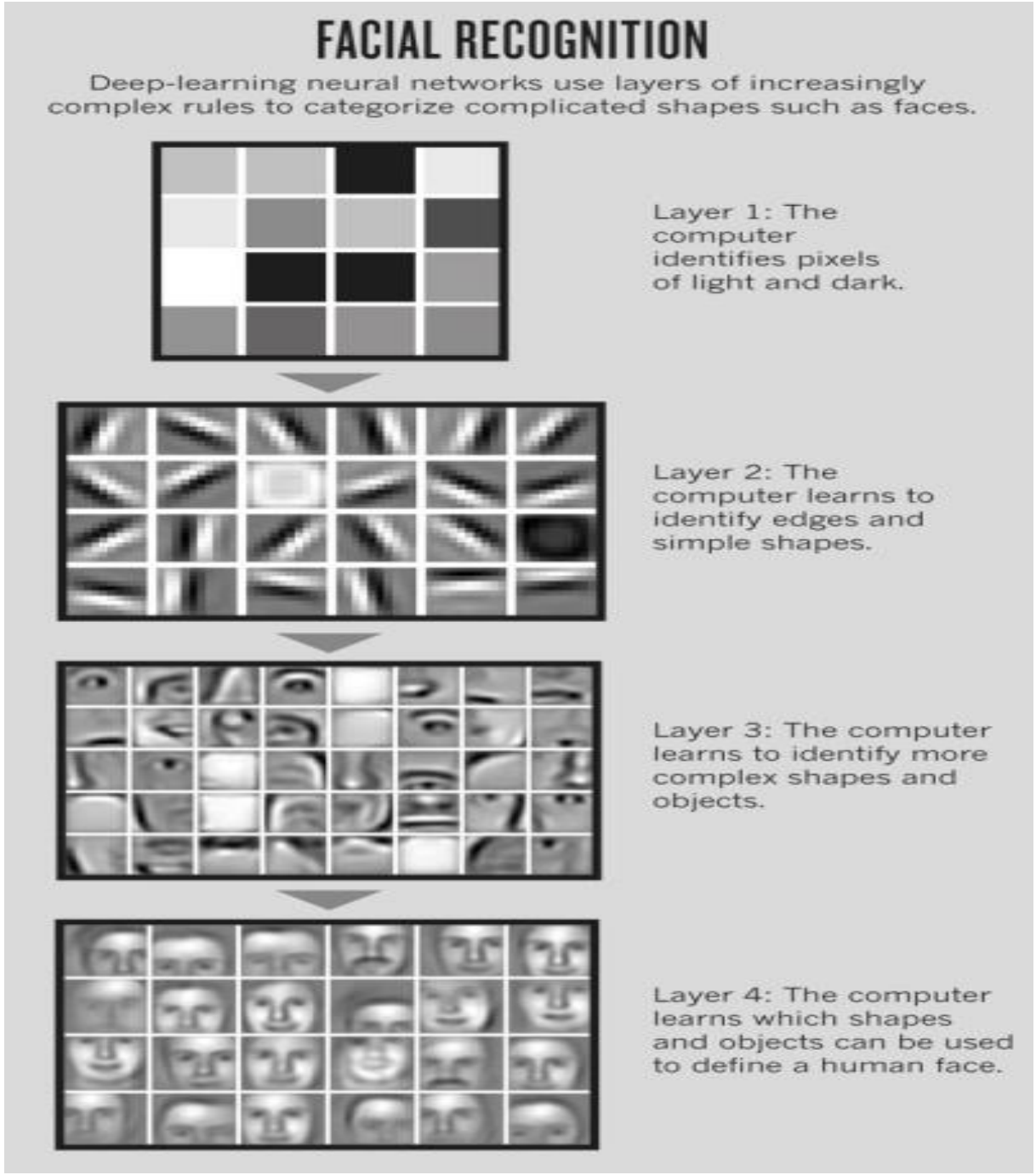
Deep Learning

Roland Olsson



The first deep learning success Classifying handwritten digits

Published test error rates without preprocessing for the MNIST dataset

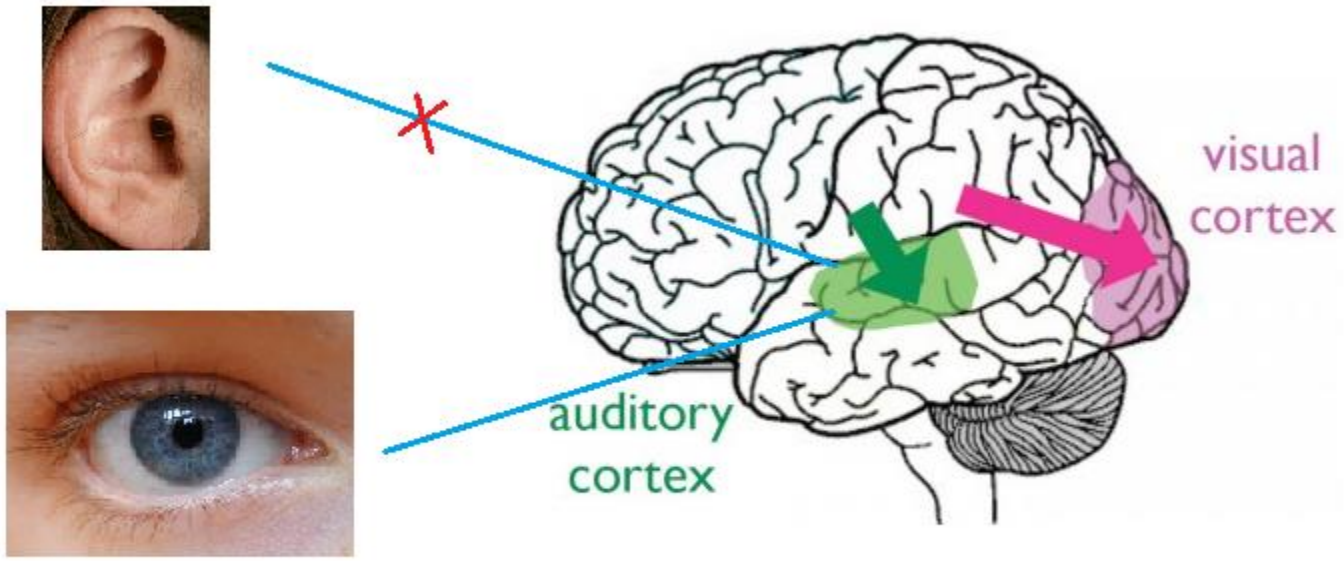
- 12% for linear discriminants
- 3.3% for 40 PCA + quadratic classifier
- 1.4% for SVM with Gaussian kernels
- 0.35% for NN with 5 hidden layers and elastic deformations

Other success stories

This is the first time a single type of model can compete with very many previous state-of-the-art results in machine learning.

Problems	Best Previous accuracy	Deep learning accuracy
Hollywood - Activity recognition	48%	53%
TIMIT - Phoneme Classification	79.2%	80.3%
CIFAR - Object classification	80.5%	82%
NORB – Object classification	94.4%	95%
AVLetters Lip reading	58.9%	65.8%
Paraphrase detection	76.1%	76.4%
…		

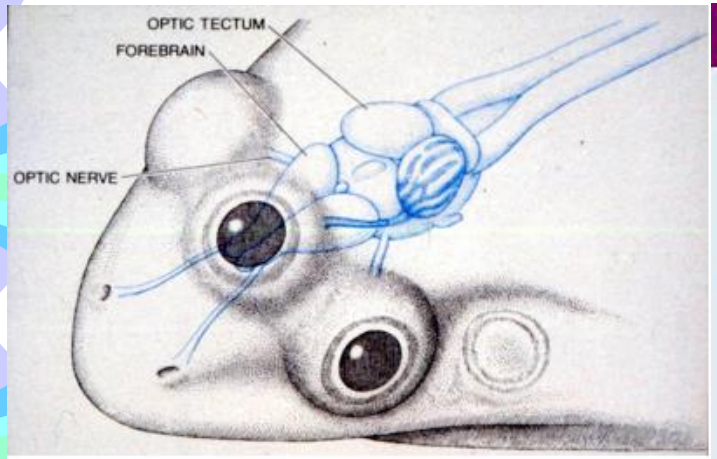
The single-algorithm hypothesis



Auditory Cortex learns to see

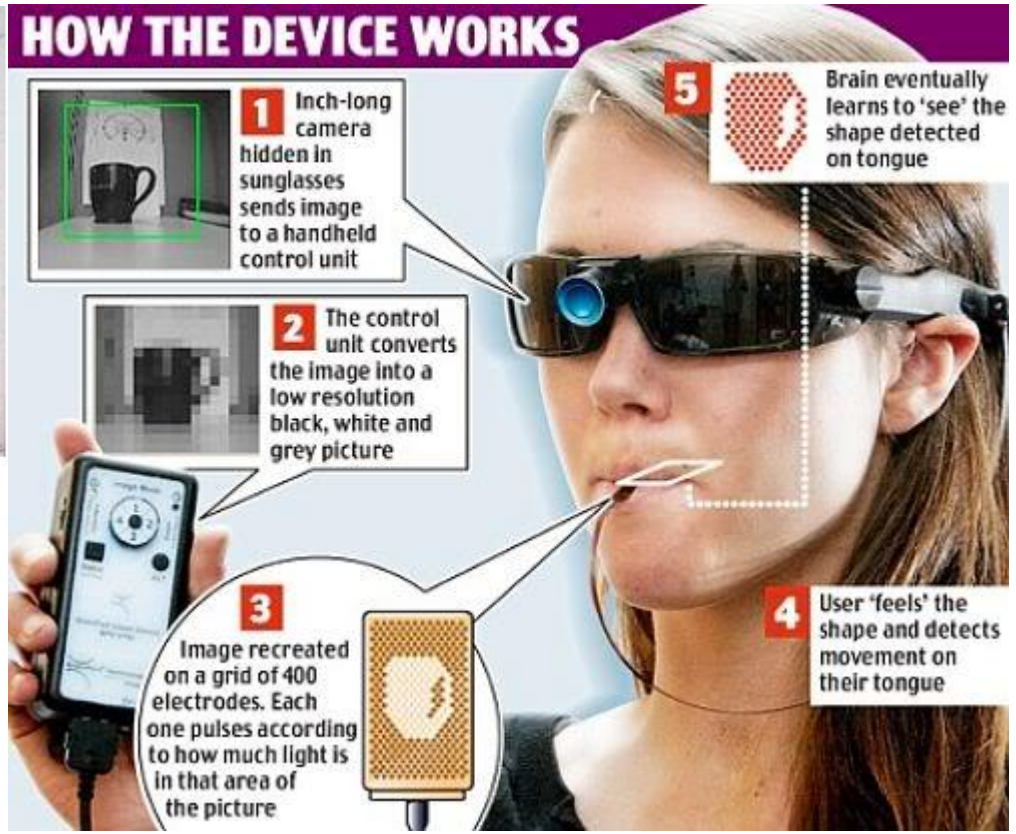
The same piece of brain tissue can process sight or sound or touch

The single-algorithm hypothesis



Implating a 3rd eye to frog

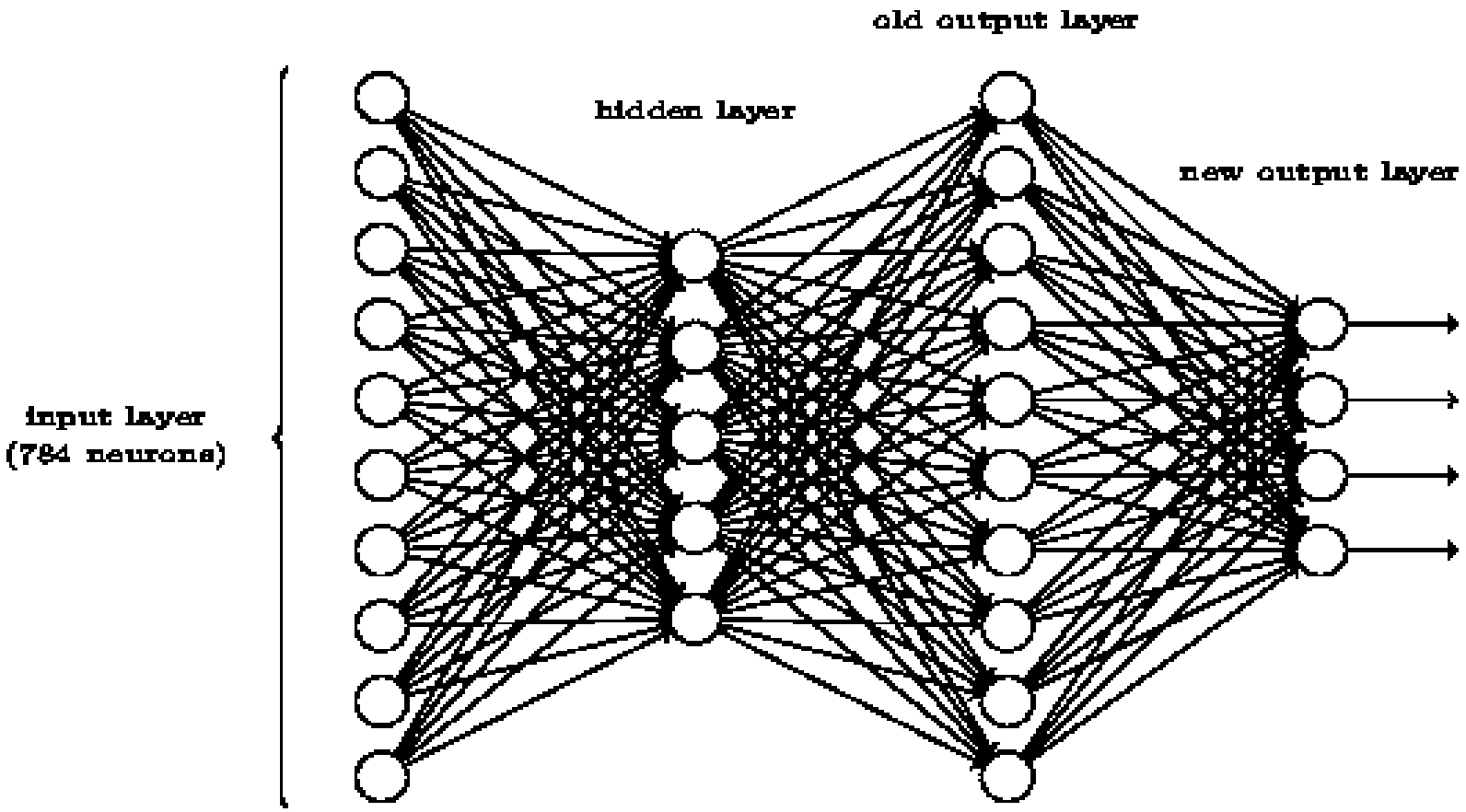
Frog can learn to use the 3rd eye



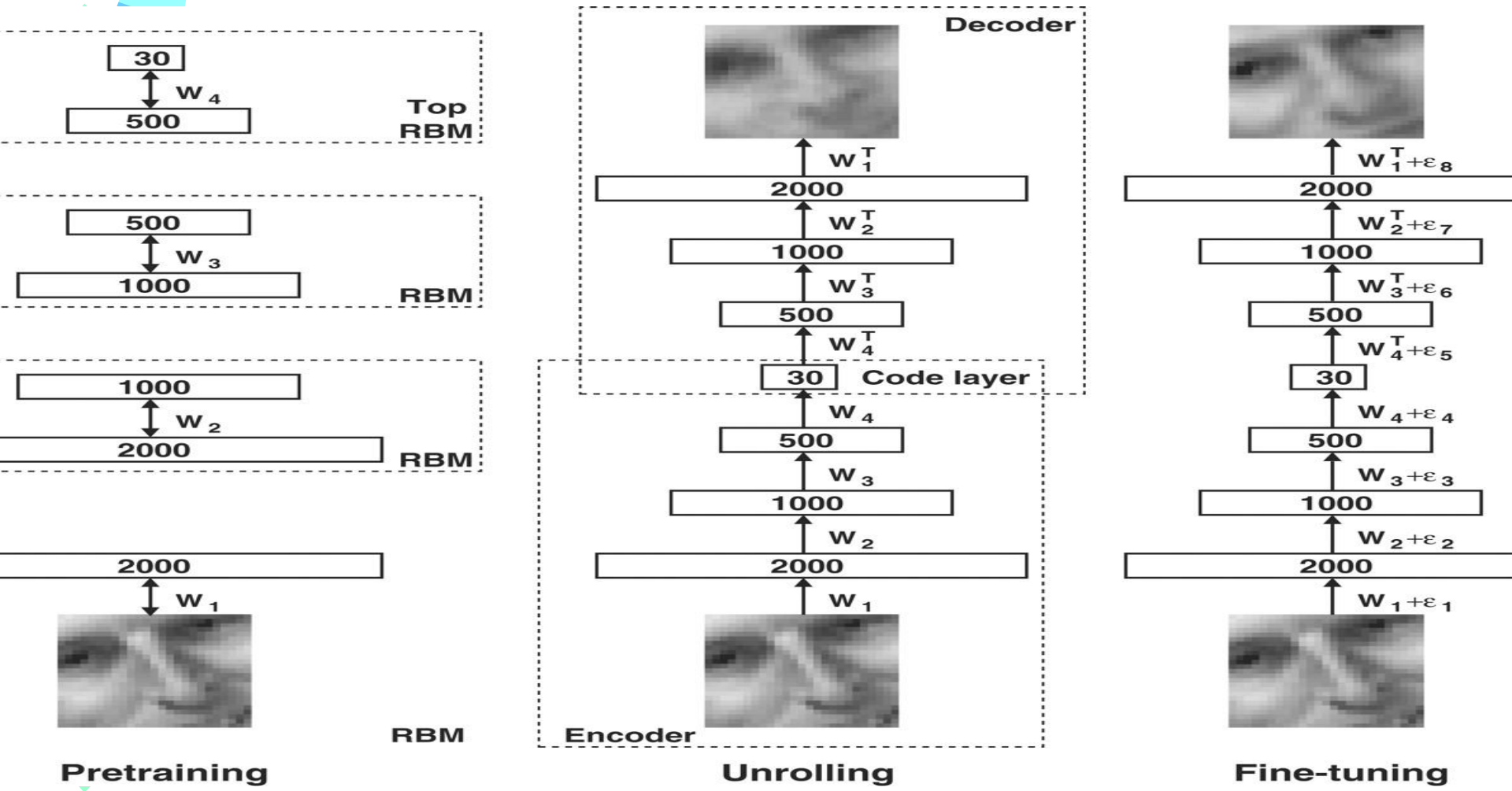
Seeing with your tounge

The brain is a general-purpose machine that can be tuned to specific tasks.

A deep neural net for MNIST



Basic architecture for an autoencoder neural network



Smart meter data

- Hourly data for 2012 and 2013 for about 8000 smart meters at Hvaler
- 8760 hours per year and 8000 smart meters give a total of 70 million instances per year.
- In the future, 2.7 million meters instead of 8000.
- Big data! Avoids overfitting in deep learning.

Deep neural net for modeling smart meter data

Use a deep subnet to model each individual building. For example, reduce the 8760 measurements for 2012 to say 4 parameters.

- Connect this net to a following one that has one input for each of the previous 48 hours.
- Train, validate and test the last net on data from 2013 and 2014.
- For each meter, also include geographic position, temperature, month, day of week

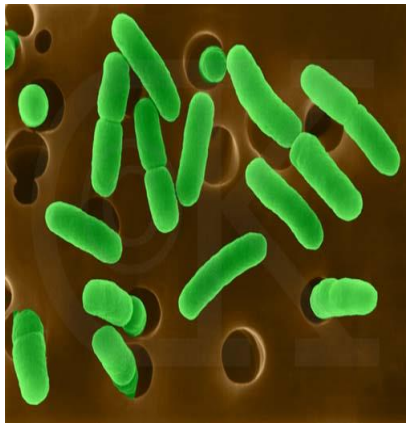
Deep neural net technologies

- Sparse initialization.
- Dropout regularization instead of L2.
- Minibatches instead of full batches.
- Optimized momentum schedule
- Massively parallel implementations
- Do not use the NN toolbox in Matlab.

Developing disruptive deep learning

- Learn from neuroscience e.g. episodic memory.
- Use automatic programming (ADATE) to generate:
  - New initial connections
  - New neuron designs
  - New regularization methods
  - New error measures

Automatic Design of Algorithms through Evolution (ADATE)



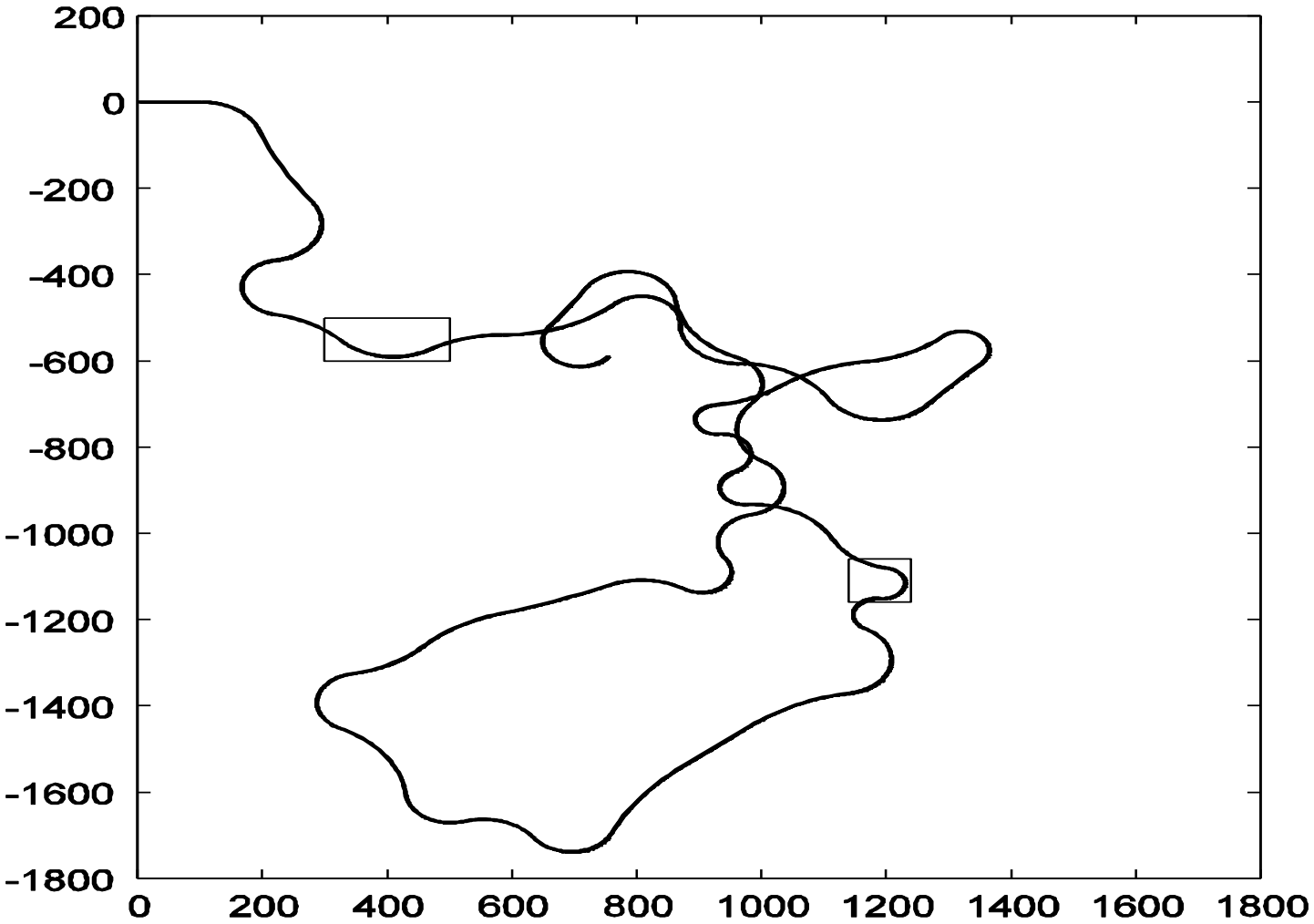
A control system example
Driving an autonomous car as fast as possible

- We have implemented a realistic physical simulation including wind resistance, tire stiffness, friction and other parameters.
- Driver inputs were chosen to speed and angles to the five points 20, 40, 60, 80 and 100 meters ahead of the car and in the middle of the road
- Driver output is steering, gas and brake
- Our methods are applicable to any control system learning or reinforcement learning scenario

Specification for fast driving of autonomous cars

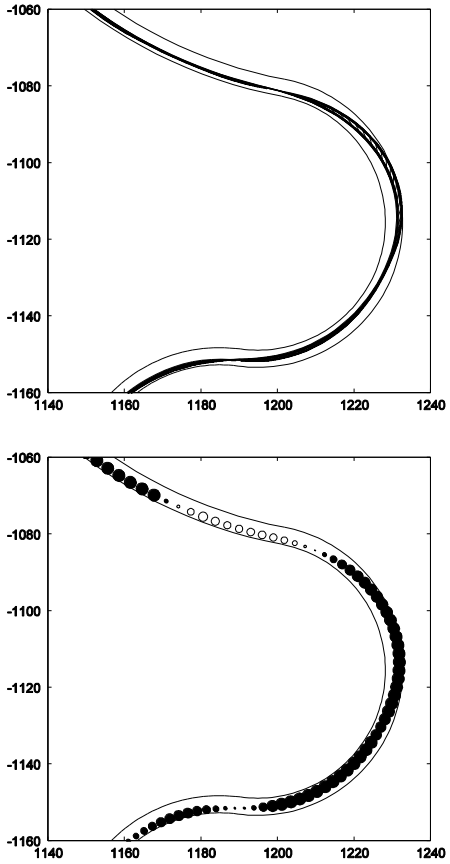
- Randomly generated flat tracks with varying widths and curve angles but constant friction
- Power and torque curves, brakes, car dimensions etc chosen to match a Golf class car
- 16 tracks, each about 3 km long used for training
- 96 tracks other tracks from the same probability distribution used for validation
- Yet another set of 96 tracks used for testing

An example of a random track

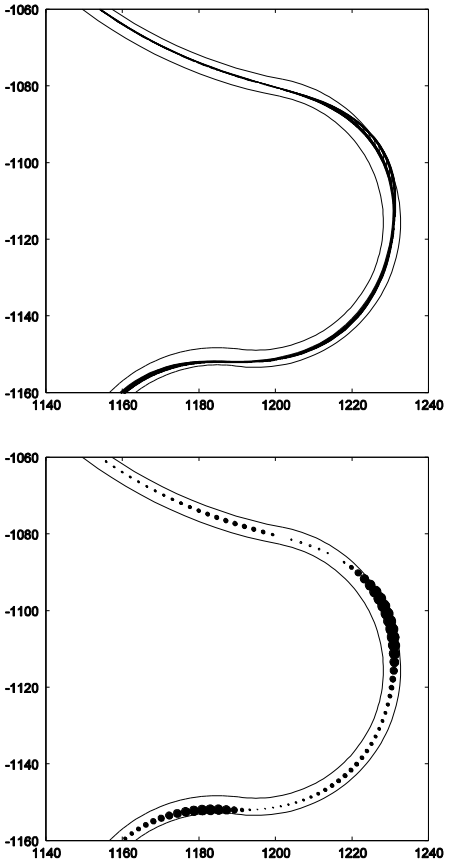


Skid marks and acceleration / braking for the best drivers

ADATE driver



ES-NN driver



A simple ADATE generated driver

```
fun f ( Us, Un, Width, DistToCenter,
          RotationSlipVelocity, Phi,
          Alpha10, Alpha20, Alpha30, Alpha40,
          Alpha50 )  =
  vector2d(
    tanh(  ( 0.3271902841577998 - Us ) /
              ( Us * Alpha30 *  Alpha30 ) -
           3.0 * Us )
       -
      Us,
    4.0 * Alpha20 - 2.0 * Phi )
```

The best ADATE generated driver

```
fun f ( Us, Un, Width, DistToCenter,
          RotationSlipVelocity, Phi,
          Alpha10, Alpha20, Alpha30, Alpha40,
          Alpha50 )  =
  vector2d(
    tanh(
          (  0.310296196852 - tanh( tanh Us  )  ) / ( Us * Alpha30 * Alpha30  ) -
          3.0 * Us )
        -
        (
          if Us < 32.9722111893 / 100 andalso Width < 3.99581671721 / 20 then
            Us
          else
             ( if Us  < 37.006446585587194 / 100 then
                 ~0.1183128271561453 / ( Alpha40 * Alpha40 )
               else
                 Alpha50 ) +
          Width ),
    4.0 * Alpha20 - 2.0 * Phi )
```

Experimental results for car racing

- We have generated driving algorithms using both automatic programming (ADATE) and neural networks trained with evolution strategies.
- The best ADATE generated driver has a mean velocity of 32.4 m/s whereas the best neural network driver manages 24.3 m/s on our test tracks.
- Our own attempts to write autonomous vehicle control algorithms failed miserably whereas automatic programming generated them easily.

Some features of ADATE

- Synthesis of primitively or generally recursive programs.
- Automatic invention of help functions where and when needed.
- “Loose” specifications requiring only evaluation (grading), not specific outputs.
- Kingdom based on size-evaluation value ordering and diversification methods.
- Starts with one initial program and grows/shrinks dynamically.
- ES / RP optimization of floating point constants