Ximantis Forecasting

ADVANCED MACHINE LEARNING MEETS MATHEMATICS

Re/discovering hidden dynamics – a ML approach

ALEXANDROS SOPASAKIS

TRAFIKVERKET
SWEDISH TRANSPORT ADMINISTRATION

Ximantis

LU INNOVATION SYSTEM
THE LINK BETWEEN ACADEMIA AND BUSINESS
LUNDS UNIVERSITET

LUNARC
Center for Scientific and Technical computing
LUNARC Lund University

CYBERCOM GROUP

Content

- Image processing
- Data generation for training
- ML training
- Real-time computations with ML

Content

- Image processing – needed or not?
- Data generation for training
- ML training
- Real-time computations with ML

Content

- Image processing – needed or not?
- Data generation for training – collection of ideas
- ML training
- Real-time computations with ML

Content

- Image processing – needed or not?
- Data generation for training – collection of ideas
- ML training – several choices
- Real-time computations with ML

Content

- Image processing – needed or not?
- Data generation for training – collection of ideas
- ML training – several choices
- Real-time computations with ML – a how to perspective

But first …. a few words about Ximantis

- What we do:

  forecasting traffic evolution & congestion

- High resolution predictions in real-time of the exact location and future time of traffic congestion - for a whole city

- Requires:
  - lots of historical data +
  - current data streaming and
  - real-time processing of all incoming information

How are such fast computations really done?

Here comes the math…

- Hybrid system of

  - Stochastic (patented) traffic model +

  - ML model

Main Statistical Mechanics Concepts
Describing the interacting vehicle system

We let Λ denote a lattice of N cells.

and consider the microscopic spin-like variable $\{\sigma_t\}_{t\geq 0}$ on Λ

We denote by s(x) the spin at location x

0 1 0 1 1 $\sigma(x)$

While we denote by $\sigma := \sigma_t$ the complete configuration of the lattice at time t.

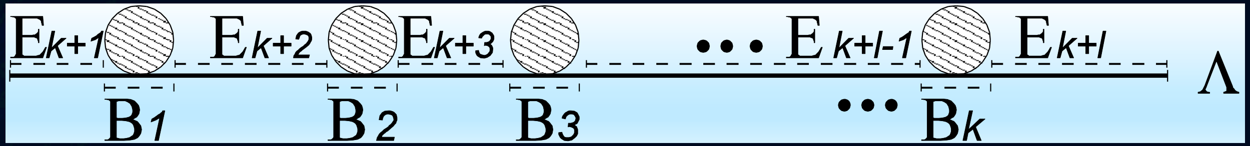
Configuration σ is an element of the configuration space $\Sigma = \{0,1\}^{\Lambda}$ and we write $\sigma = \{\sigma(x) : x \in \Lambda\}$

Lattice-free Arrhenius rates

Spin – flip rate for particles adsorbing/desorbing from/to the problem domain. The rates c are calculated from

$$c(i,\sigma) = \begin{cases} c_d \exp(-\beta U(i,\sigma)) & if \ \ \sigma(i) = 1 \\ c_a w(i) & if \ \ \sigma(i) = 0 \end{cases}$$

where $$w(i) = \begin{cases} |E_i| - |V| & if \ \ |E_i| > |V| \\ 0 & otherwise \end{cases}$$



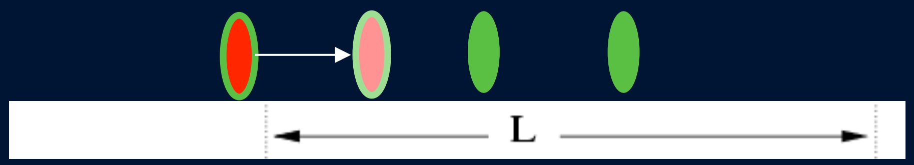
Incorporating the Physics and Creating the ASEP

We define the interaction potential

$$U(x,t) = \sum_{\substack{z \neq x \\ z \in \Lambda}} J(x-y)\sigma(z)$$

where

$$J(r) = \begin{cases} J_0, & \text{if } 0 < r < L \\ 0, & \text{otherwise} \end{cases}$$



and **parameter** *L* denotes the range of interactions.

Here **parameter** *Jo* denotes the strength of the interactions.

This potential enforces:

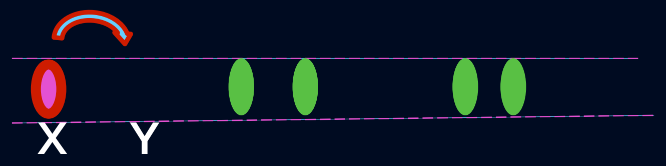
- Vehicles do not move backwards
- Local effect of the interactions

Building the continuous time, space Markov Chain

Microscopic Arrhenius Spin-Exchange Dynamics

We introduce a lattice-free Arrhenius spin-exchange rate $c(x,y,\sigma)$,

$$c(x,y,\sigma) = \frac{1}{\tau_0}\,\sigma(x)\,[1-\sigma(y)]\,w(y)\,e^{-U(x,\sigma)}$$



where **parameter** τ_0 denotes the characteristic time of the process and U is the interaction potential.

Incorporating interactions & multi-lane motion

Let's look once again

at the rate functional

to move forward

$$c(x,y,\sigma) = \frac{1}{\tau_0}\,\sigma(x)\,[1-\sigma(y)]\,w(y)\,e^{-U(x,\sigma)}$$

We incorporate lane-changing via an additional anisotropy type potential.
Thus our **total interaction potential** now consists of:

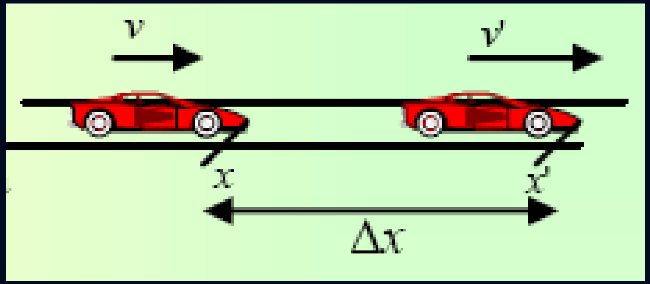
$$U_T(x) = U(x) + \boxed{U_a(x)}$$ where $$U_a(x) = \sum_{y=nn} \psi(x,y)(1-\sigma(y))$$

with $$\psi(x,y) = \begin{cases} k_l & \text{if y = left} \\ k_r & \text{if y = right} \\ k_f & \text{if y = forward} \end{cases}$$

Free Parameters and Calibration

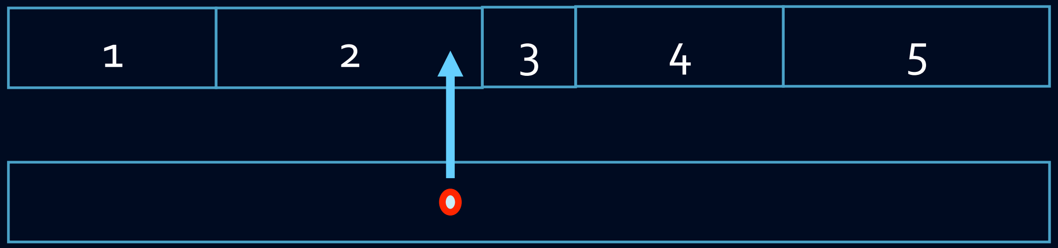
The model is characterized by the following three undetermined parameters:

- τ_0 - the characteristic time of the stochastic process
- J_0 - the strength of the interactions
- L - the interaction potential range
- .....



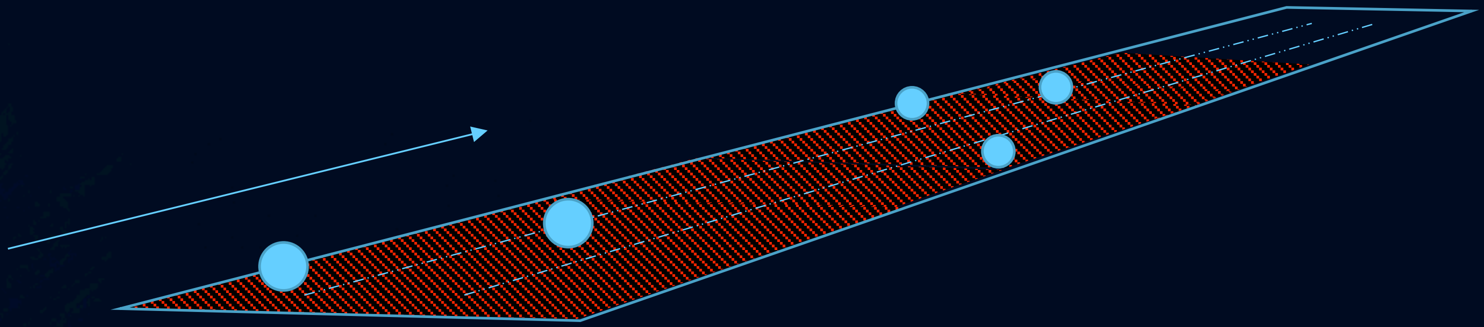
Monte Carlo simulation and prediction of road traffic

Probabilities



Random pick

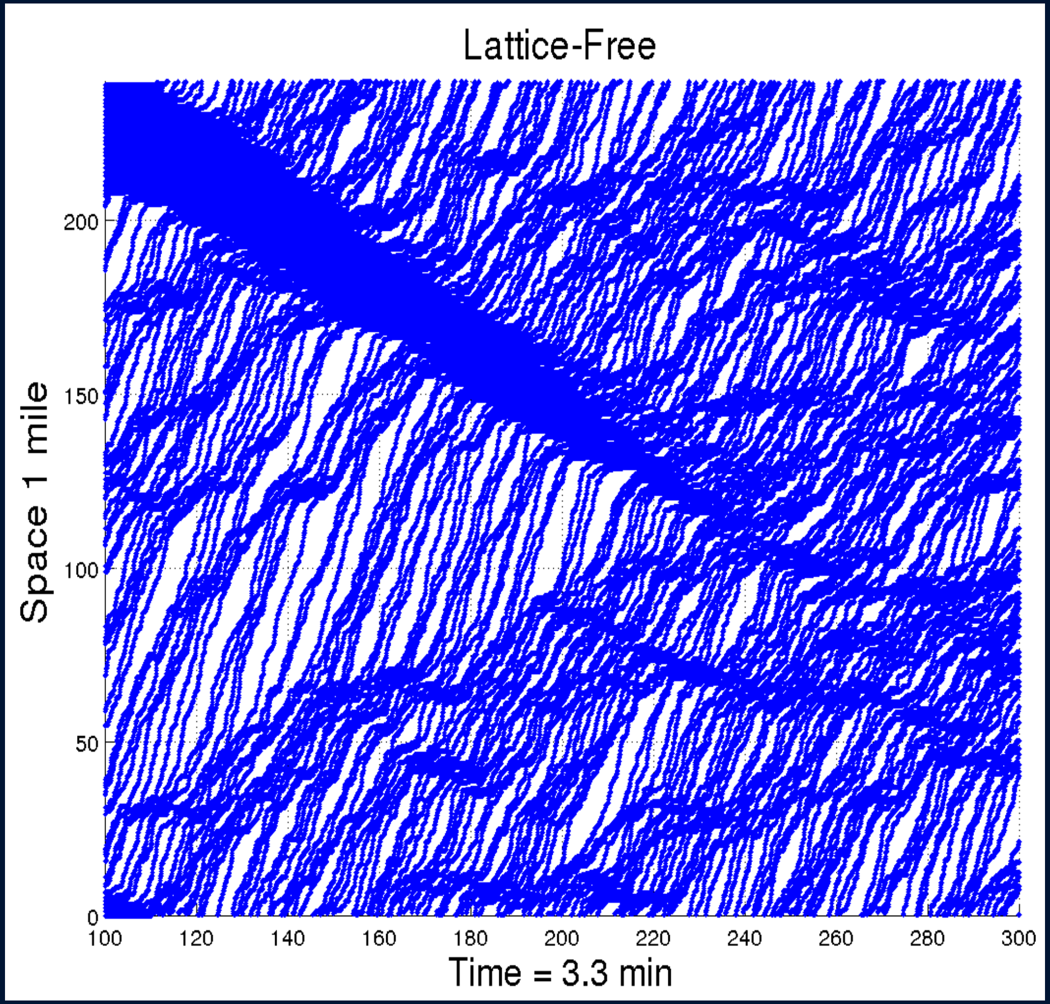
Monte Carlo moves vehicle 2



The probability of a spin-exchange between x and y during time $[t, t+\Delta t]$

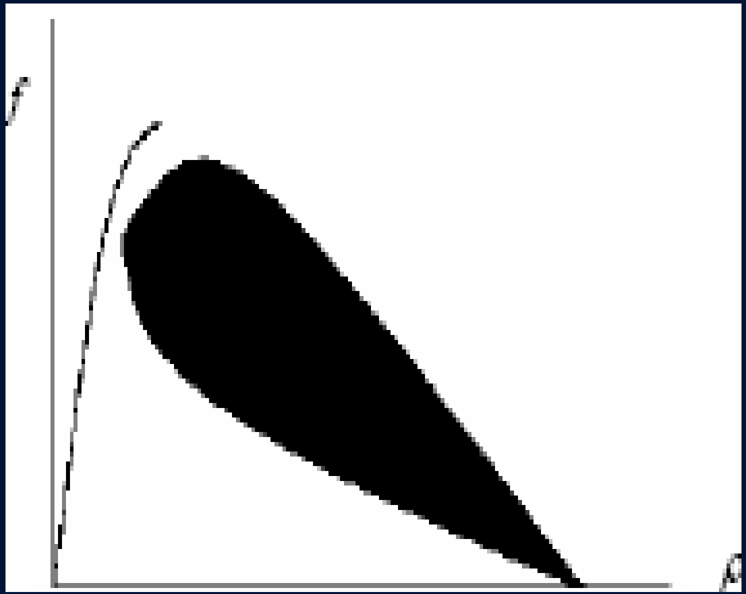
is $c(x, y, \sigma)\Delta t + O(\Delta t^2)$

Validation – Advanced Features
Timely breaking/returded acceleration

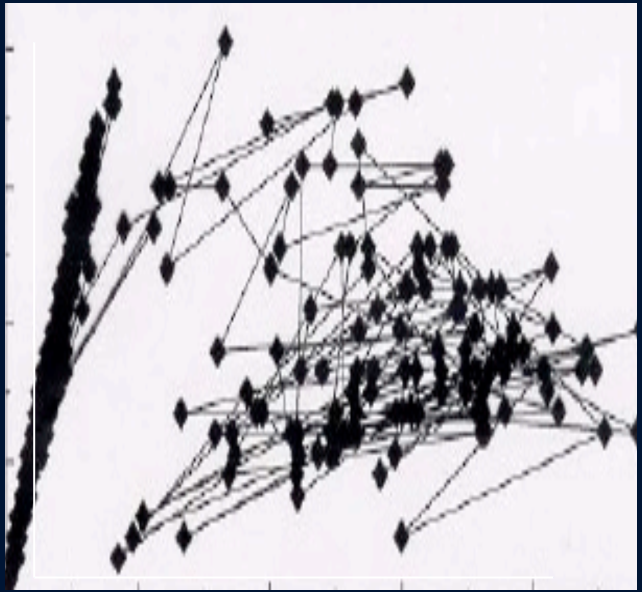


Validation
Fundamental Diagram

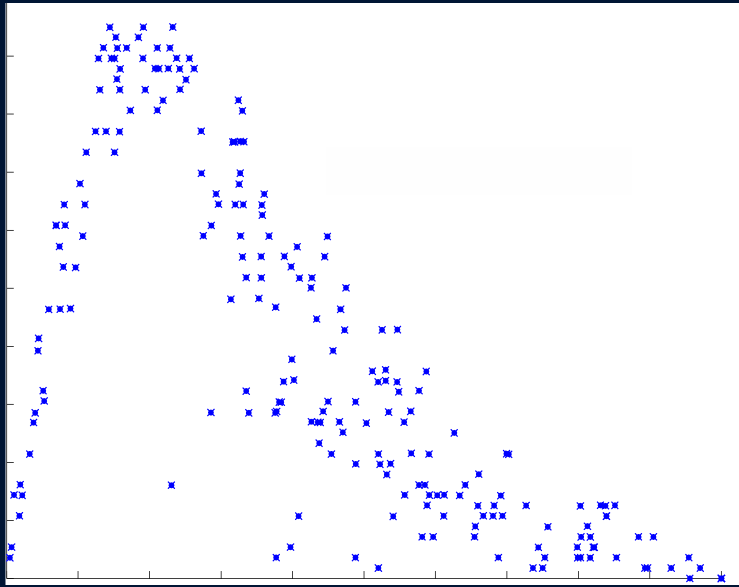
Flow-Density Diagram



Actual Data

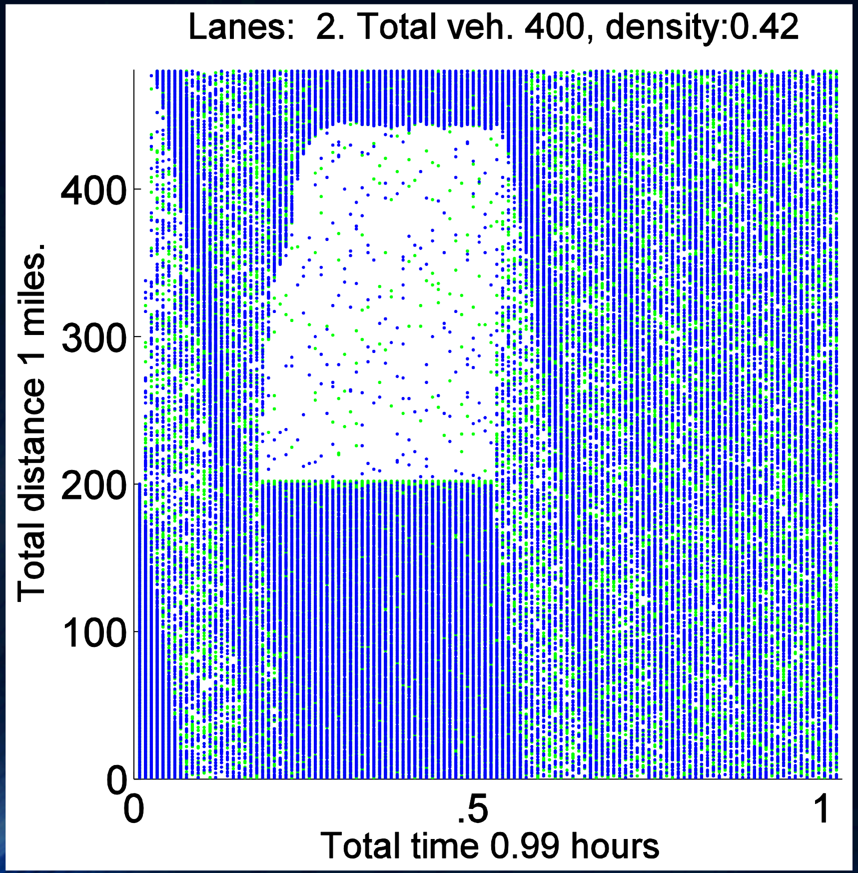


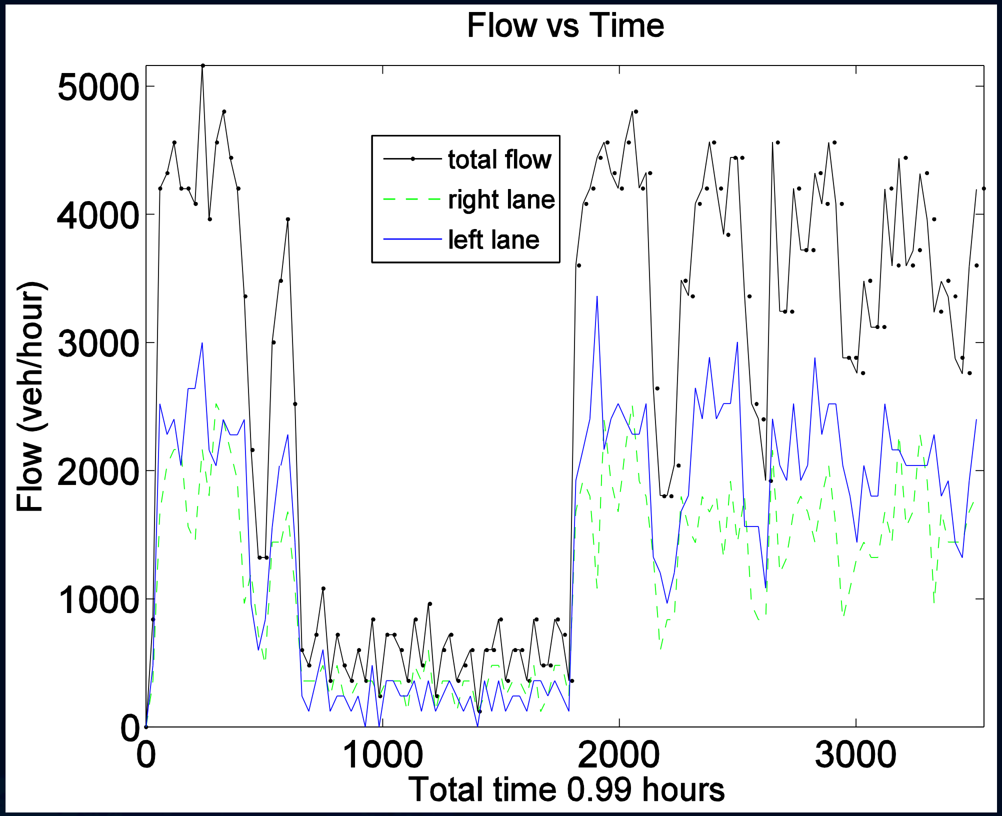
Simulated Data



Validation: an incident

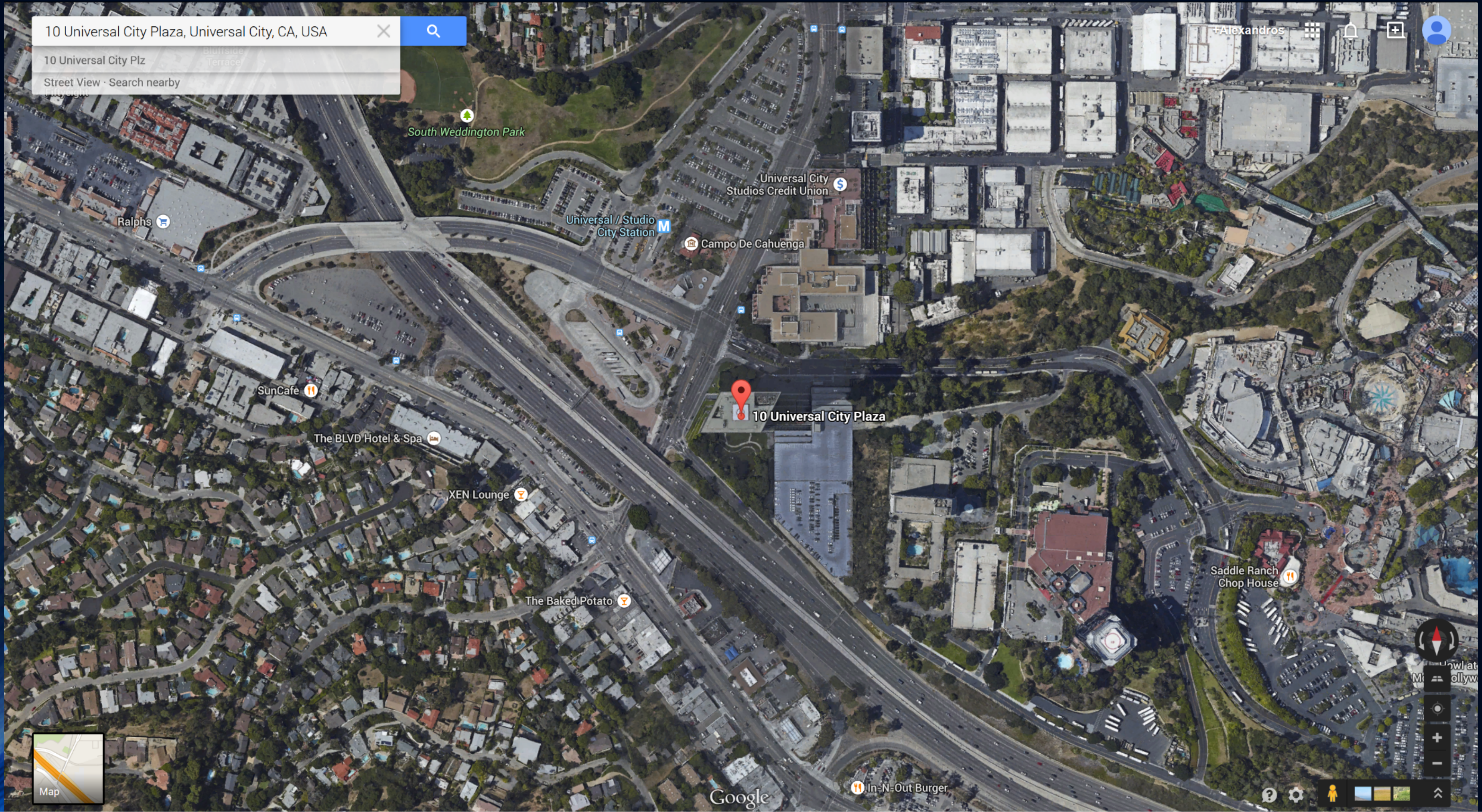
- Assume a 2-lane highway
- After some time Lane 1 is blocked due to an accident





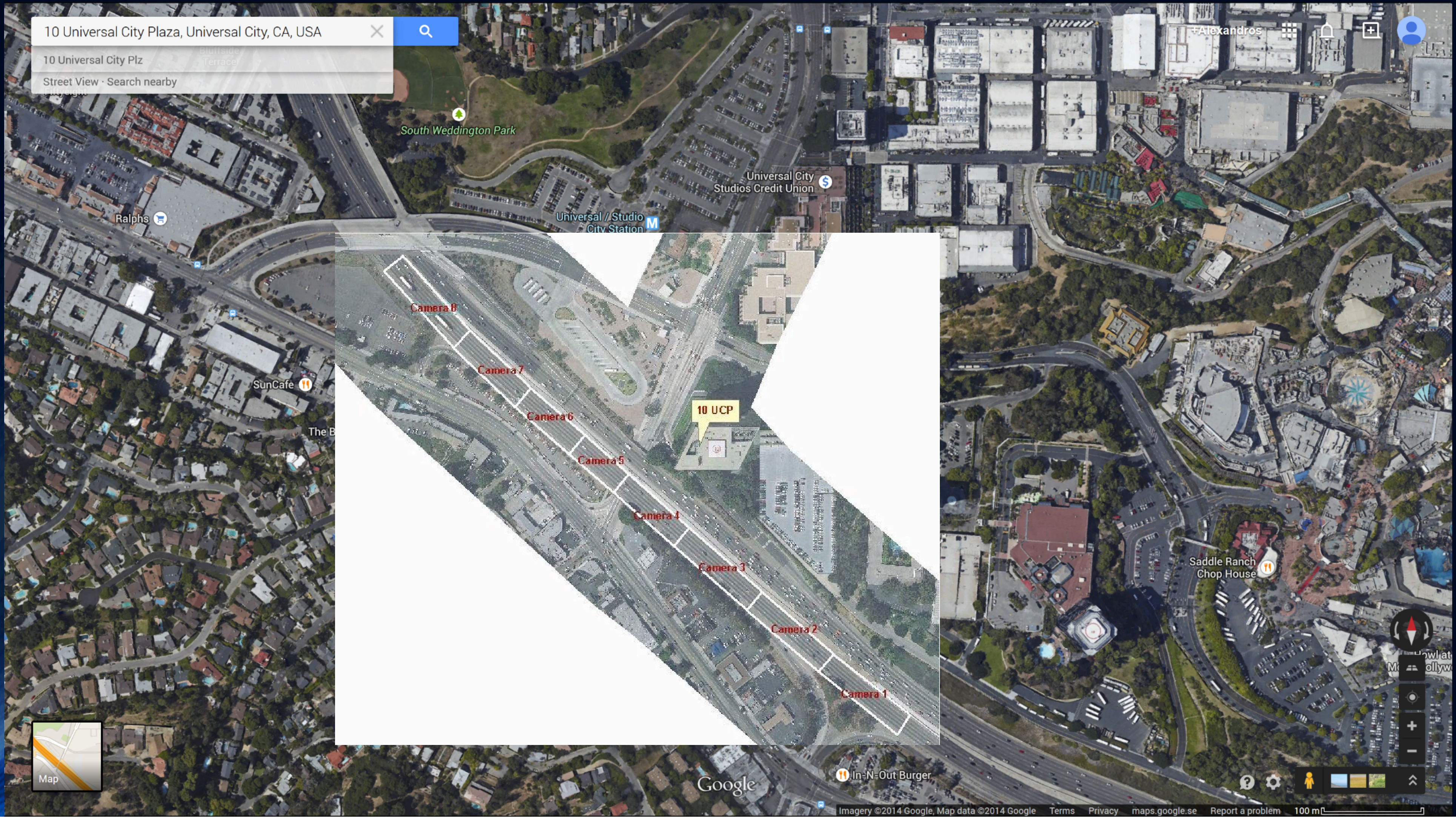
Test case: a real highway – the NGSIM project

- Highway U.S. 101 near Los Angeles, in California
- 5 lanes with entrances and exits.



Test case: a real highway – the NGSIM project

- Highway U.S. 101 near Los Angeles, in California
- 5 lanes with entrances and exits.



A Monte Carlo Multi-Lane, Multi-Class Vehicle simulation…

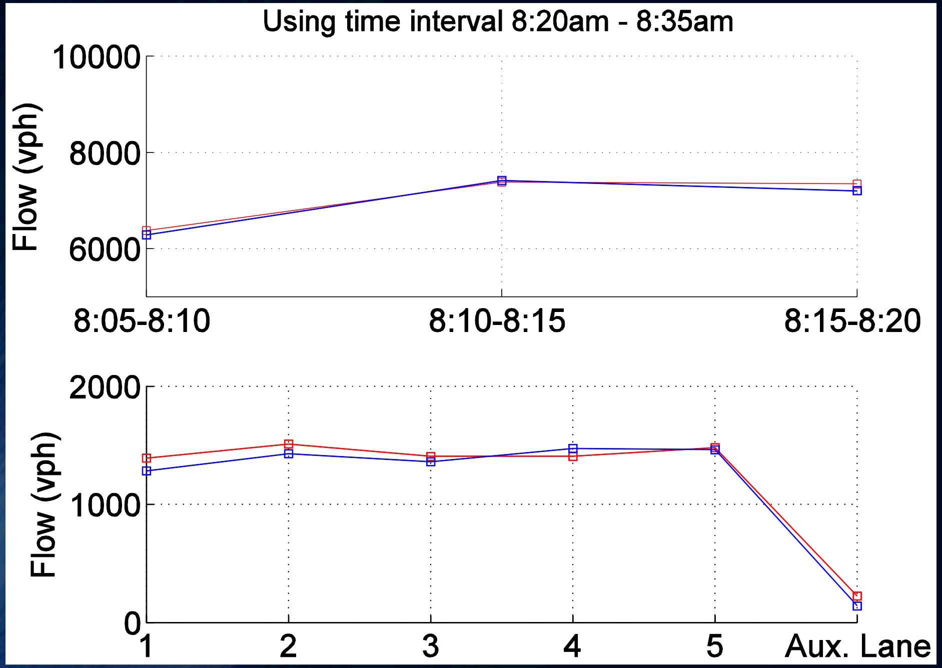
Real Data

Simulation

carSimulator.html

Test case: simulations vs reality

- Highway U.S. 101, Los Angeles, California
- 5 lanes with entrances and exits.
- 15 minutes intervals of very detailed rush hour data: 8:05am to 8:20am



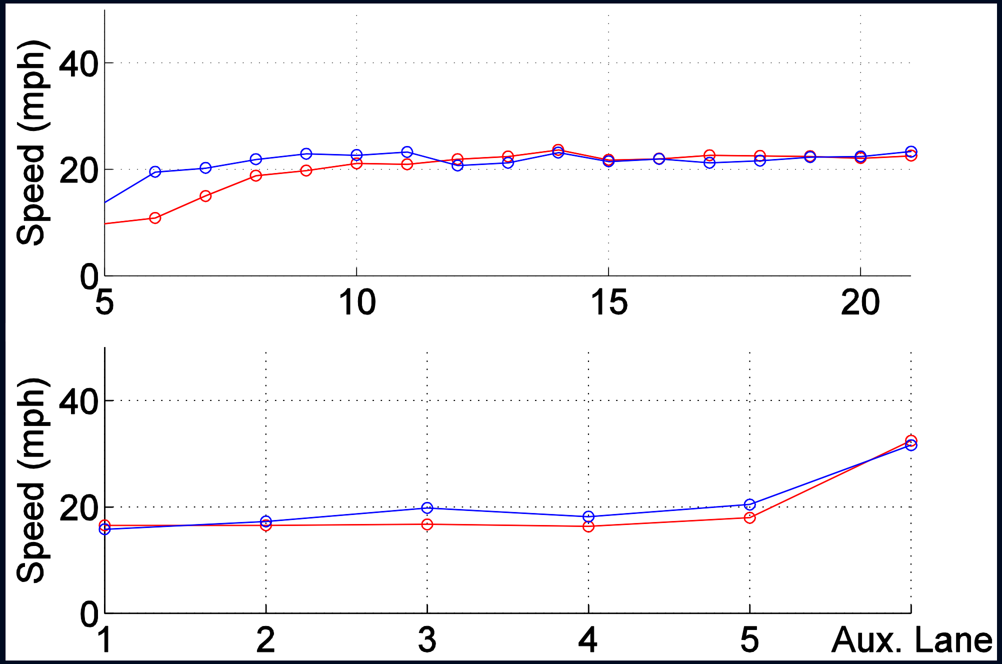
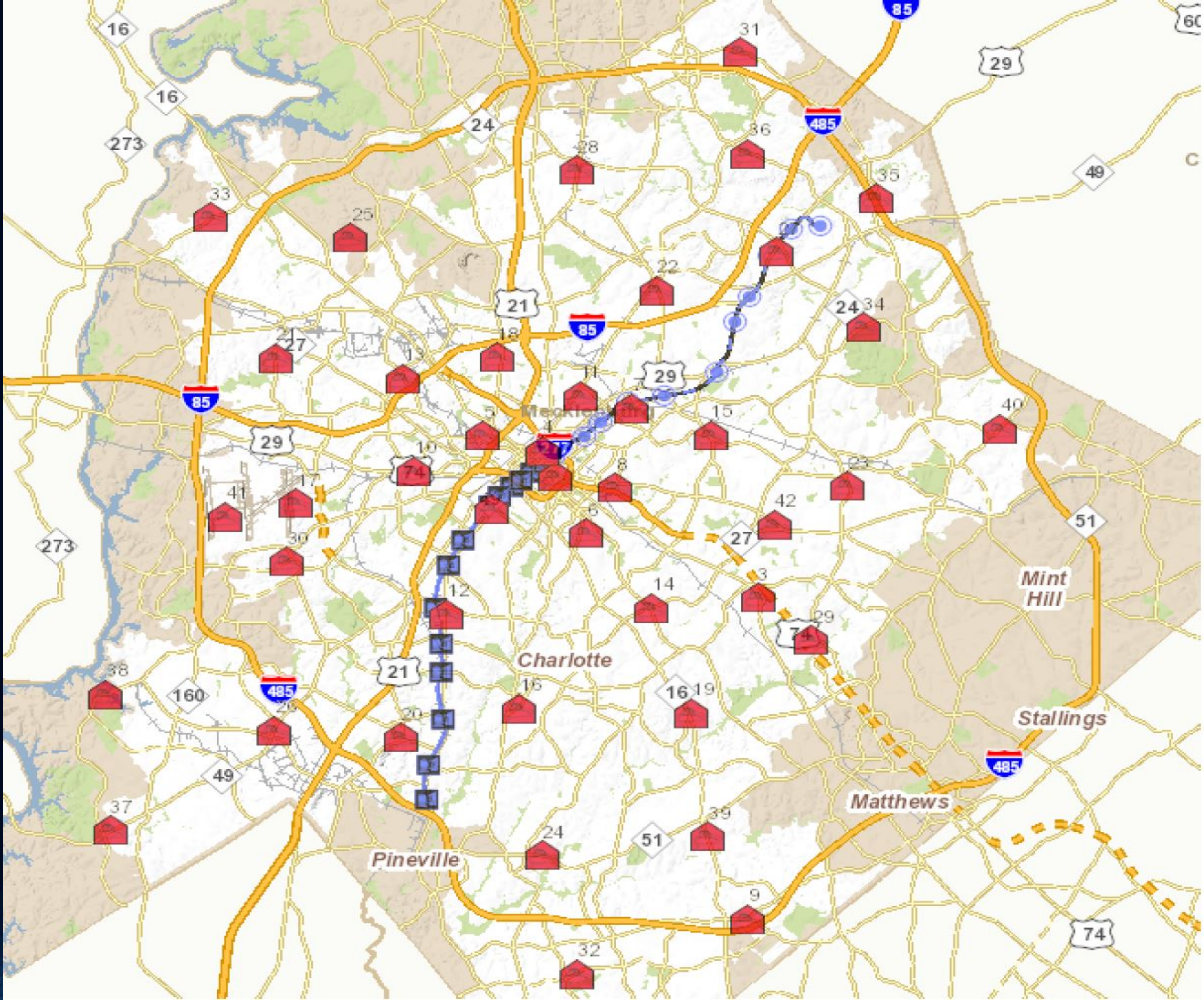


Image Processing – the data

Images or Video from

- Charlotte,

Charlotte

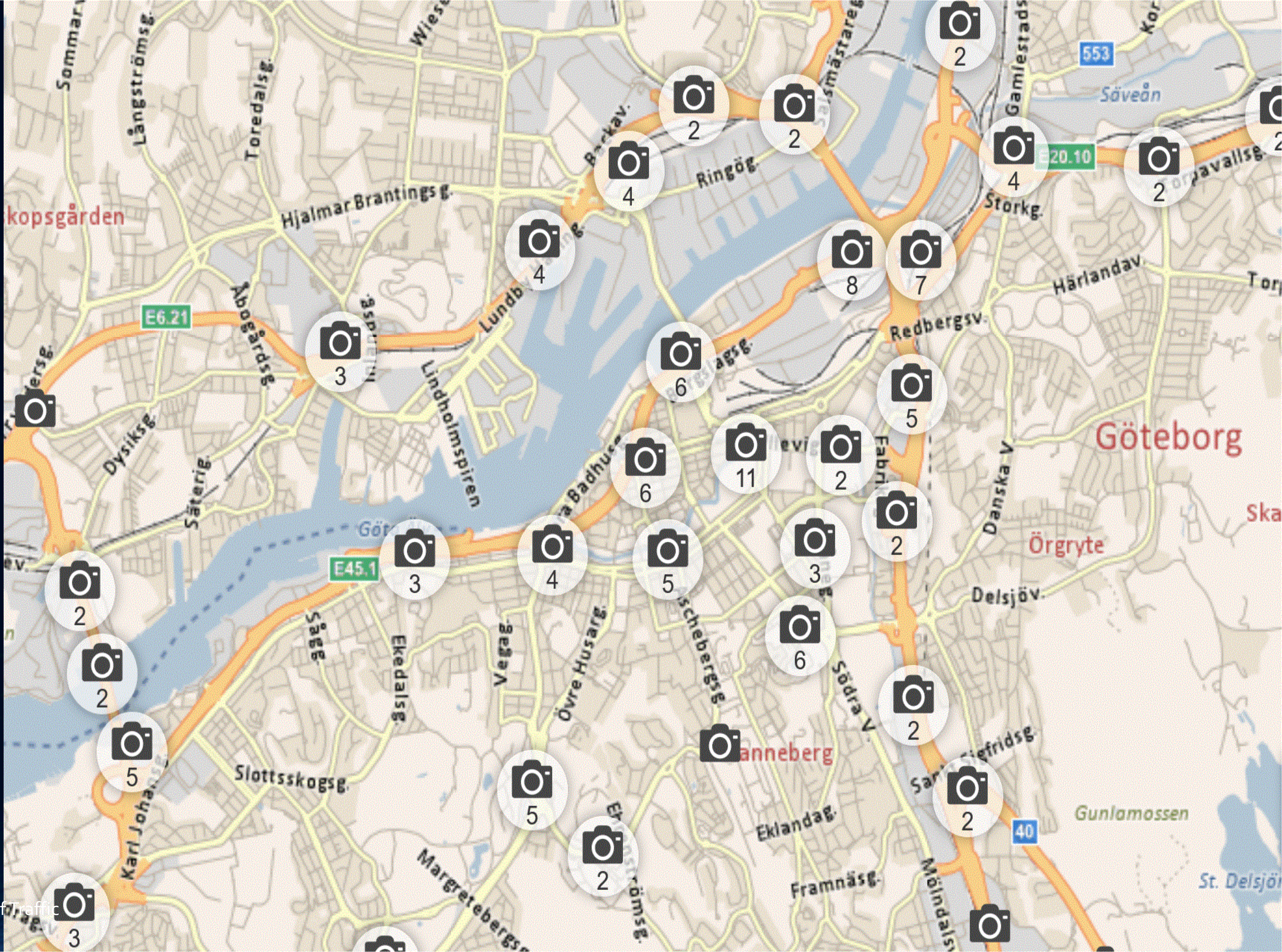


Data Collection

Images or Video from

- Charlotte,
- Goteborg

Goteborg



Göteborg
Hjalmar Brantingsg.
Lundby
Lindholmspiren
Ringög.
Redbergsv.
Härlandav.
Örgryte
Delsjöv.
Gunlamossen
Slottsskogsg.
Eklandag.
Framnäsg.
Säveån
Storkg.
Toredalsg.
Långströmsg.
Dysiksg.
Säterig.
Ekedalsg.
Vegag.
Övre Husarg.
Danska V.
Södra V.
Karl Johansg.
Margretebergsg.
E6.21
E45.1
E20.10
553
40
St. Delsjön

Images

Data Collection

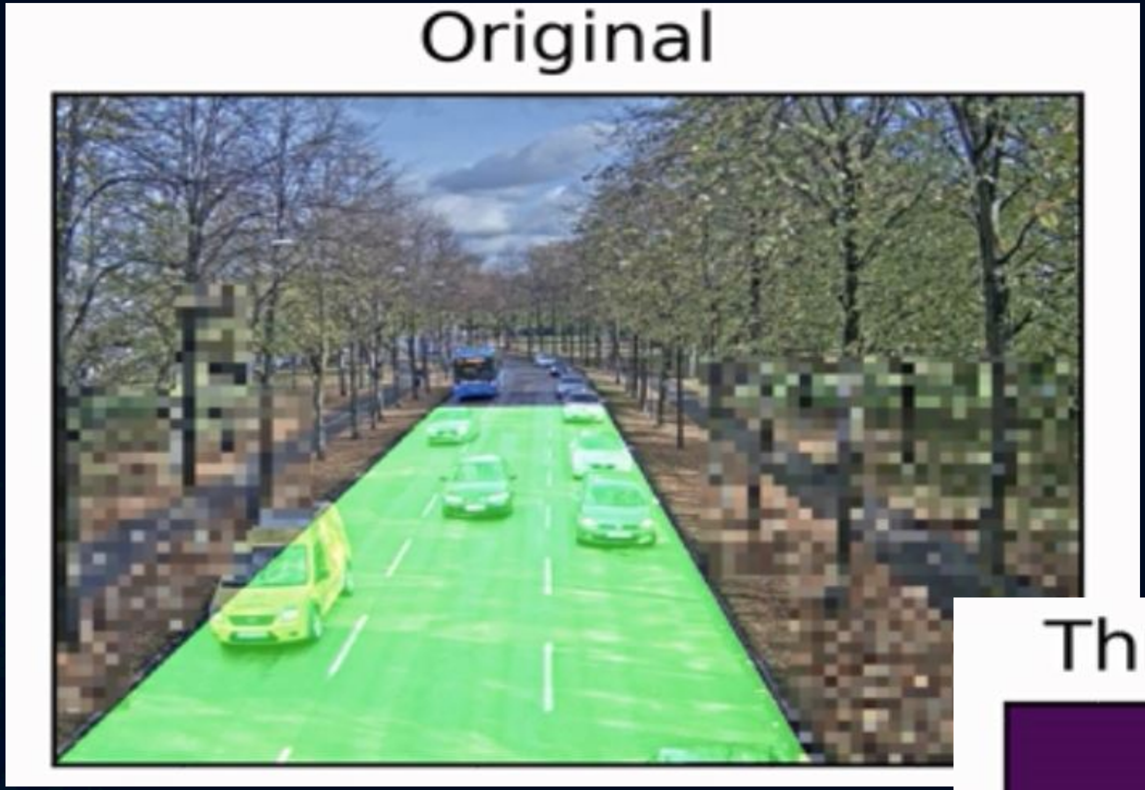
Images or Video from

- Charlotte,
- Goteborg,
- Stockholm
- etc

Image Processing and Counting

- Images

Images



Images

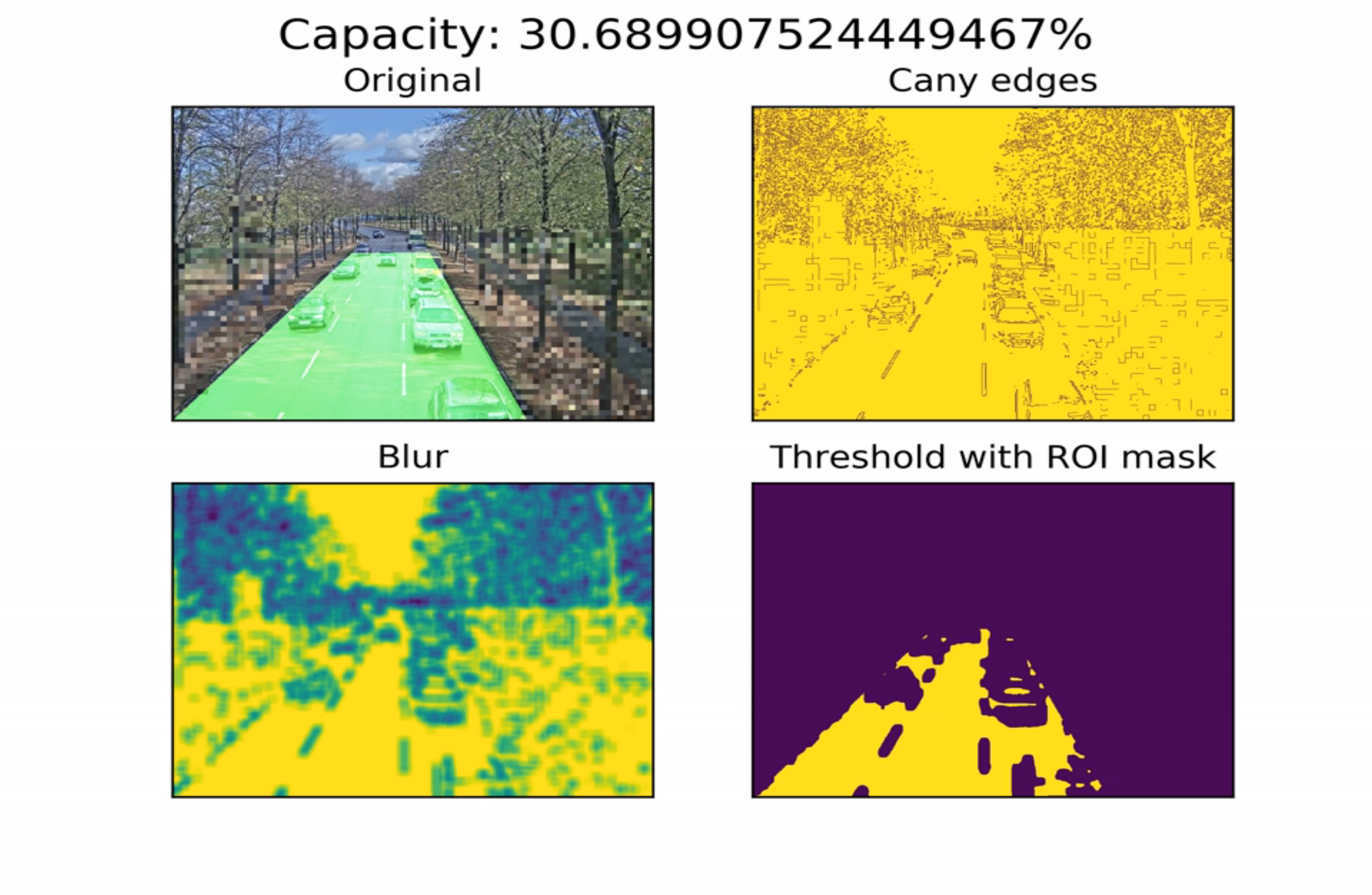
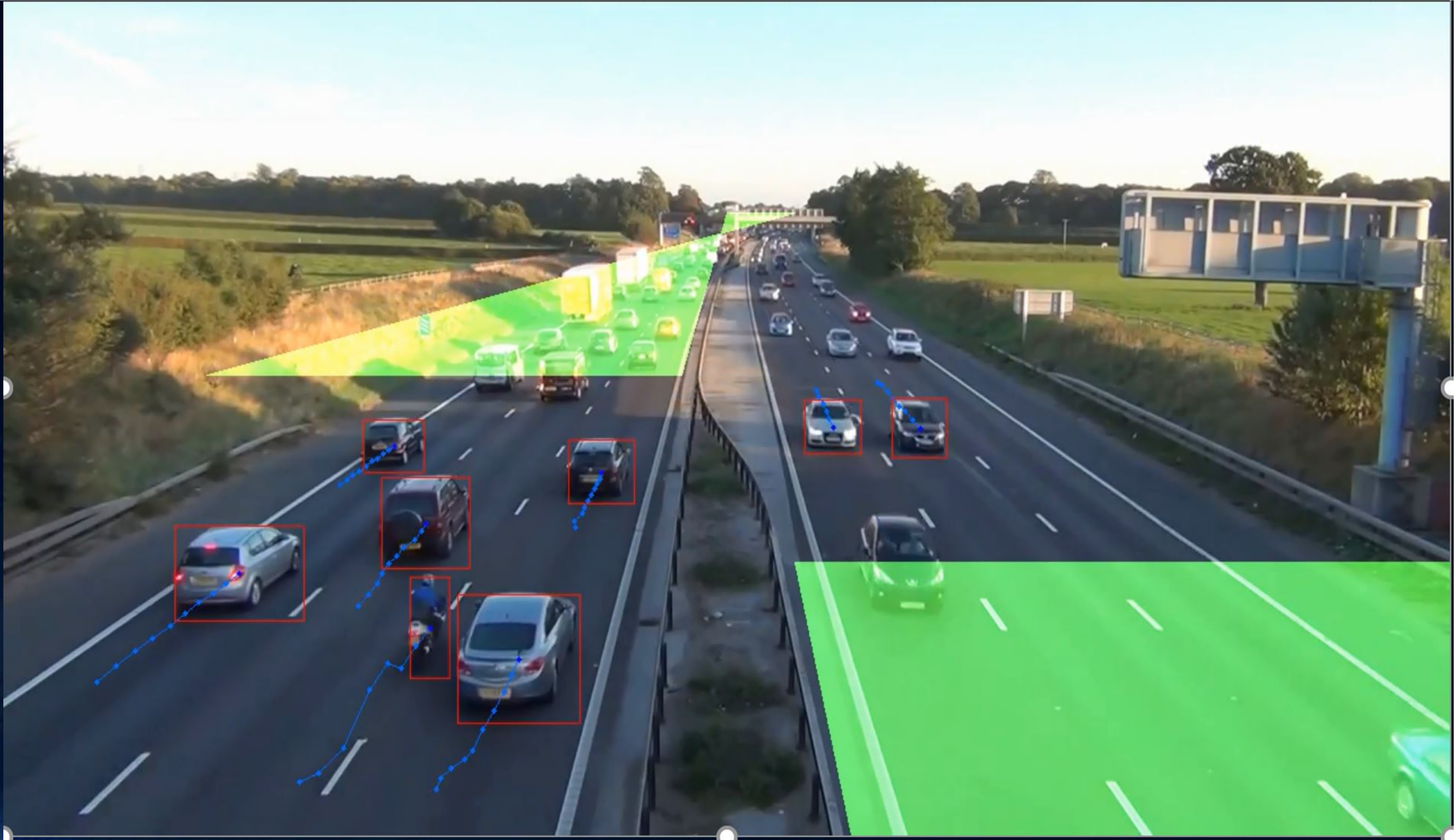

Capacity: 30.689907524449467%
Original
Cany edges
Blur
Threshold with ROI mask

Image Processing and Counting

- Images
- Video

Video

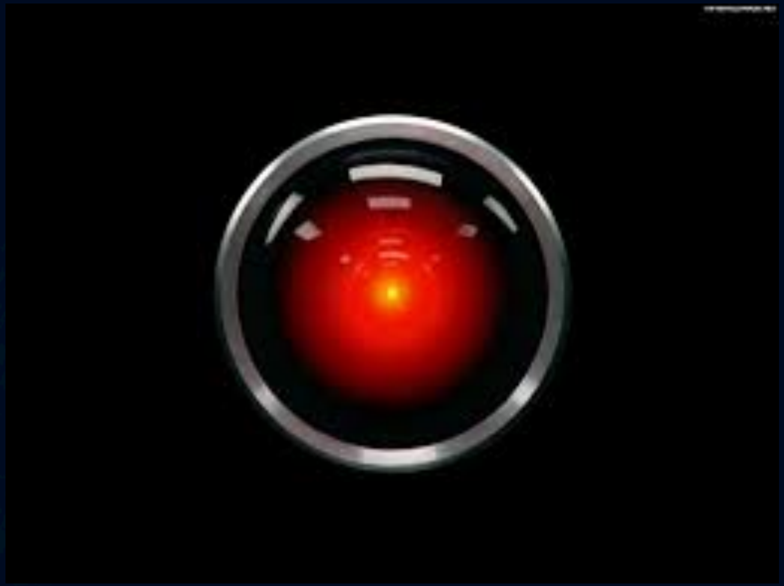
Video



Video

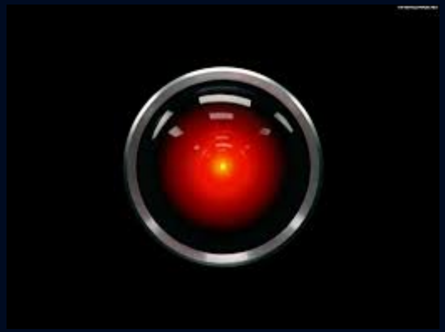
Generate Data in Real Time at our AWS cloud

2	13	437	1118846980200	16.467	35.381	6451137.641	1873344.962	14.5	4.9	2	40.00	0.00	2	0	0	0.00	0.00
2	14	437	1118846980300	16.447	39.381	6451140.329	1873342.000	14.5	4.9	2	40.00	0.00	2	0	0	0.00	0.00
2	15	437	1118846980400	16.426	43.381	6451143.018	1873339.038	14.5	4.9	2	40.00	0.00	2	0	0	0.00	0.00
2	16	437	1118846980500	16.405	47.380	6451145.706	1873336.077	14.5	4.9	2	40.00	0.00	2	0	0	0.00	0.00
2	17	437	1118846980600	16.385	51.381	6451148.395	1873333.115	14.5	4.9	2	40.00	0.00	2	0	0	0.00	0.00
2	18	437	1118846980700	16.364	55.381	6451151.084	1873330.153	14.5	4.9	2	40.00	0.00	2	0	0	0.00	0.00
2	19	437	1118846980800	16.344	59.381	6451153.772	1873327.192	14.5	4.9	2	40.00	0.00	2	0	0	0.00	0.00
2	20	437	1118846980900	16.323	63.379	6451156.461	1873324.230	14.5	4.9	2	40.02	0.25	2	0	0	0.00	0.00
2	21	437	1118846981000	16.303	67.383	6451159.149	1873321.268	14.5	4.9	2	40.03	0.13	2	0	0	0.00	0.00
2	22	437	1118846981100	16.282	71.398	6451161.838	1873318.307	14.5	4.9	2	39.93	-1.63	2	0	13	0.00	0.00
2	23	437	1118846981200	16.262	75.401	6451164.546	1873315.323	14.5	4.9	2	39.61	-4.54	2	0	13	0.00	0.00
2	24	437	1118846981300	16.254	79.349	6451167.199	1873312.382	14.5	4.9	2	39.14	-5.73	2	0	13	0.00	0.00
2	25	437	1118846981400	16.221	83.233	6451169.802	1873309.533	14.5	4.9	2	38.61	-5.15	2	0	13	0.00	0.00
2	26	437	1118846981500	16.201	87.043	6451172.358	1873306.719	14.5	4.9	2	38.28	-1.61	2	0	13	0.00	0.00
2	27	437	1118846981600	16.169	90.829	6451174.961	1873303.870	14.5	4.9	2	38.42	3.73	2	0	13	0.00	0.00
2	28	437	1118846981700	16.204	94.683	6451177.613	1873300.929	14.5	4.9	2	38.78	4.86	2	0	13	0.00	0.00
2	29	437	1118846981800	16.252	98.611	6451180.342	1873297.924	14.5	4.9	2	38.92	0.00	2	0	13	0.00	0.00
2	30	437	1118846981900	16.339	102.560	6451182.980	1873294.961	14.5	4.9	2	38.54	-8.59	2	0	13	0.00	0.00
2	31	437	1118846982000	16.400	106.385	6451185.537	1873292.122	14.5	4.9	2	37.51	-11.20	2	0	13	0.00	0.00
2	32	437	1118846982100	16.430	110.079	6451188.021	1873289.408	14.5	4.9	2	36.34	-10.86	2	0	13	0.00	0.00
2	33	437	1118846982200	16.435	113.628	6451190.424	1873286.817	14.5	4.9	2	35.50	-6.20	2	0	13	0.00	0.00
2	34	437	1118846982300	16.478	117.118	6451192.757	1873284.247	14.5	4.9	2	35.08	-1.89	2	0	13	0.00	0.00
2	35	437	1118846982400	16.520	120.600	6451195.109	1873281.656	14.5	4.9	2	34.96	0.18	2	0	13	0.00	0.00
2	36	437	1118846982500	16.562	124.096	6451197.462	1873279.065	14.5	4.9	2	34.98	0.25	2	0	13	0.00	0.00
2	37	437	1118846982600	16.605	127.597	6451199.814	1873276.473	14.5	4.9	2	35.00	0.04	2	0	13	0.00	0.00
2	38	437	1118846982700	16.647	131.099	6451202.167	1873273.882	14.5	4.9	2	34.99	-0.20	2	0	13	0.00	0.00
2	39	437	1118846982800	16.691	134.595	6451204.519	1873271.290	14.5	4.9	2	34.98	-0.02	2	0	13	0.00	0.00
2	40	437	1118846982900	16.727	138.081	6451206.879	1873268.700	14.5	4.9	2	35.10	1.95	2	0	13	0.00	0.00
2	41	437	1118846983000	16.796	141.578	6451209.191	1873266.113	14.5	4.9	2	35.49	5.55	2	0	13	0.00	0.00
2	42	437	1118846983100	16.795	145.131	6451211.610	1873263.514	14.5	4.9	2	36.20	8.99	2	0	13	0.00	0.00
2	43	437	1118846983200	16.724	148.784	6451214.156	1873260.882	14.5	4.9	2	37.15	10.44	2	0	13	0.00	0.00
2	44	437	1118846983300	16.588	152.559	6451216.824	1873258.213	14.5	4.9	2	38.12	9.30	2	0	13	0.00	0.00
2	45	437	1118846983400	16.376	156.449	6451219.616	1873255.522	14.5	4.9	2	38.76	4.36	2	0	13	0.00	0.00
2	46	437	1118846983500	16.064	160.379	6451222.548	1873252.829	14.5	4.9	2	38.95	-0.73	2	0	13	0.00	0.00
2	47	437	1118846983600	15.763	164.277	6451225.462	1873250.139	14.5	4.9	2	38.95	-1.15	2	0	13	0.00	0.00
2	48	437	1118846983700	15.471	168.150	6451228.376	1873247.450	14.5	4.9	2	38.99	1.90	2	0	13	0.00	0.00
2	49	437	1118846983800	15.226	172.044	6451231.290	1873244.760	14.5	4.9	2	39.18	3.47	2	0	13	0.00	0.00
2	50	437	1118846983900	14.979	176.000	6451234.204	1873242.071	14.5	4.9	2	39.34	0.02	2	0	13	0.00	0.00
2	51	437	1118846984000	14.720	179.959	6451237.144	1873239.374	14.5	4.9	2	39.20	-3.52	2	0	13	0.00	0.00
2	52	437	1118846984100	14.508	183.862	6451239.988	1873236.708	14.5	4.9	2	38.89	-3.28	2	0	13	0.00	0.00
2	53	437	1118846984200	14.331	187.716	6451242.770	1873234.057	14.5	4.9	2	38.73	-0.33	2	0	13	0.00	0.00
2	54	437	1118846984300	14.240	191.561	6451245.501	1873231.336	14.5	4.9	2	38.88	3.49	2	0	13	0.00	0.00
2	55	437	1118846984400	14.309	195.455	6451248.125	1873228.494	14.5	4.9	2	39.28	5.00	2	0	13	0.00	0.00
2	56	437	1118846984500	14.428	199.414	6451250.788	1873225.539	14.5	4.9	2	39.68	3.76	2	0	13	0.00	0.00
2	57	437	1118846984600	14.540	203.417	6451253.489	1873222.554	14.5	4.9	2	39.94	1.29	2	0	13	0.00	0.00
2	58	437	1118846984700	14.646	207.430	6451256.177	1873219.592	14.5	4.9	2	40.02	-0.22	2	0	13	0.00	0.00
2	59	437	1118846984800	14.751	211.431	6451258.866	1873216.630	14.5	4.9	2	40.00	-0.21	2	0	13	0.00	0.00
2	60	437	1118846984900	14.856	215.428	6451261.554	1873213.669	14.5	4.9	2	39.99	0.00	2	0	13	0.00	0.00
2	61	437	1118846985000	14.962	219.427	6451264.243	1873210.707	14.5	4.9	2	39.99	0.00	2	0	13	0.00	0.00
2	62	437	1118846985100	15.067	223.462	6451266.932	1873207.745	14.5	4.9	2	39.65	-5.35	2	0	13	0.00	0.00



I am sorry Dave, I am afraid I cannot do that!

AI



Possible and effective because:

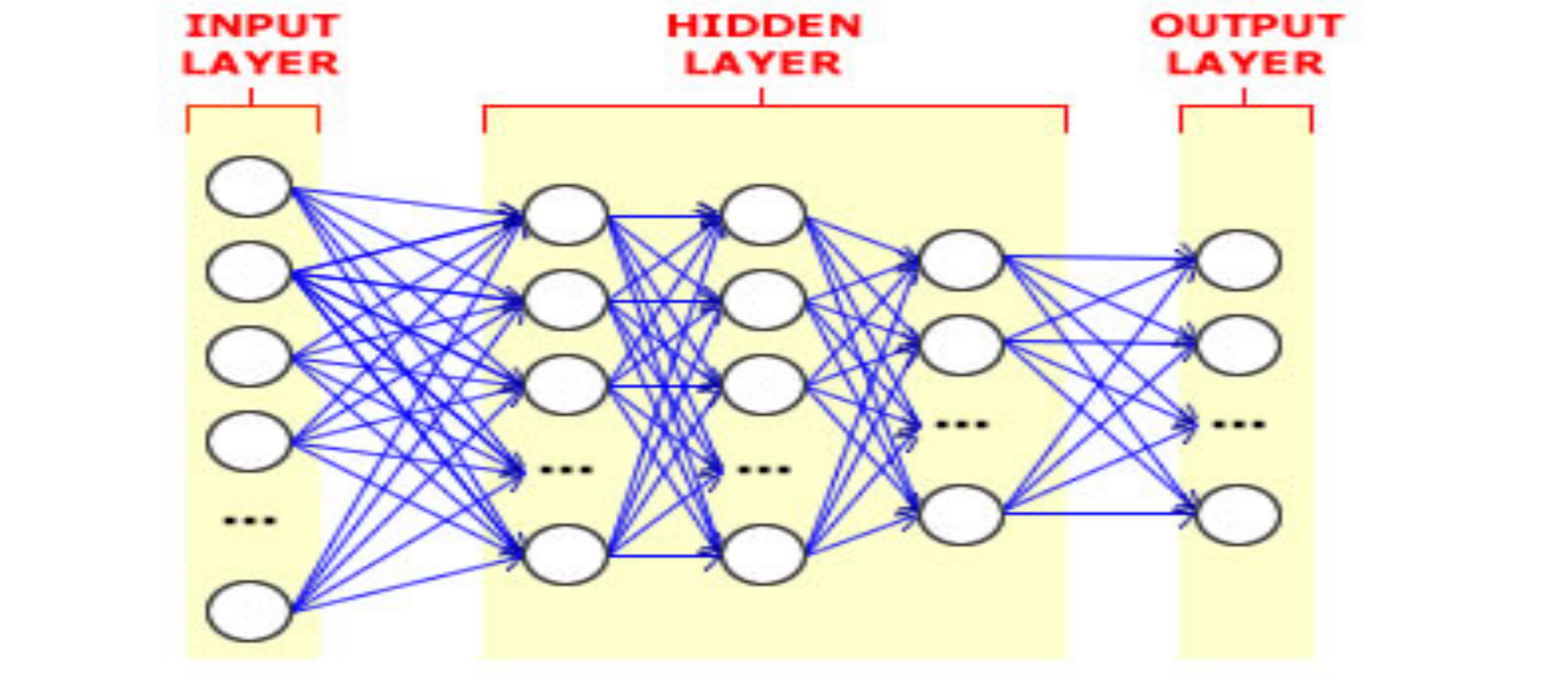
- Data
- Powerful computers

How it works:

- Pay attention and learn (powerful computers)
- Algorithm figures out Patterns (lots of data available to do that)
- Remember but not too much… (design the network properly and allow to forget!)

Neural Networks

- Neural Networks imitating how the brain synapses process information
- Basic components: input layer, hidden layers, **output layer** each comprised of **nodes** and connected by **weights**



Historical

- Fukushima (1980) – Neo-Cognitron
- LeCun (1998) – Convolutional Neural Networks (CNN)
  - Similarities to Neo-Cognitron
- Many layered MLP with backpropagation
  - Tried early but without much success
    - Very slow
    - Vanishing gradient
  - Relatively recent work demonstrated significant accuracy improvements by "patiently" training deeper MLPs with BP using fast machines (GPUs)
    - More general learning!
    - Much improved since 2012 with lots of extensions to the basic BP algorithm

Brief History of Neural Networks (NN)

- 1943: McCulloch & Pitts show that neurons can be combined to construct a Turing machine
- 1958: Rosenblatt shows that perceptrons will converge if what they are trying to learn can be represented
- 1969: Minsky & Papert showed the limitations of perceptrons, killing research for a decade
- 1985: The backpropagation algorithm revitalizes the field
  - Geoff Hinton et al
- 2006: The Hinton lab solves the training problem for DNNs

A few recent applications

http://www.cs.toronto.edu/~graves/handwriting.html

- Language identification (Gonzalez-Dominguez et al., 2014)
- Paraphrase detection (Cheng & Kartsaklis, 2015)
- Speech recognition (Graves, Abdel-Rahman, & Hinton, 2013)
- Handwriting recognition (Graves & Schmidhuber, 2009)
- Music composition (Eck & Schmidhuber, 2002) and lyric generation (Potash, Romanov, & Rumshisky, 2015)
- Robot control (Mayer et al., 2008)
- Natural language generation (Wen et al. 2015) (best paper at EMNLP)
- Named entity recognition (Hammerton, 2003)

Few Examples of Machine Learning Problems

- Pattern Recognition
  - Facial identities or facial expressions
  - Handwritten or spoken words (e.g., Siri)
  - Medical images
  - Sensor Data/IoT
- Optimization
  - Many parameters have "hidden" relationships that can be the basis of optimization
- Pattern Generation
  - Generating images or motion sequences
- Anomaly Detection
  - Unusual patterns in the telemetry from physical and/or virtual plants (e.g., data centers)
  - Unusual sequences of credit card transactions
  - Unusual patterns of sensor data from a nuclear power plant
    - or unusual sound in your car engine or …
- Prediction
  - Future stock prices or currency exchange rates
  - Network events
  - …

AI Learning?

- Learning is a procedure that consists of estimating the model parameters so that the learned model (algorithm) can perform a specific task
  - In Artificial Neural Networks, these parameters are the *weight matrix* ($w_{i,j}$'s)

- Types of learning considered here
  - Supervised
  - Unsupervised
  - Semi-supervised learning
  - Reinforcement learning

- Supervised learning
  - Present the algorithm with a set of inputs and their corresponding outputs
  - See how closely the actual outputs match the desired ones
    - Note generalization error (bias, variance)
  - Iteratively modify the parameters to better approximate the desired outputs (gradient descent)

- Unsupervised
  - Algorithm learns internal representations and important features

A typical machine learning task
What is a “2”?

Supervised learning

- The desired response (function) of the data is given
  - You are given the correct answer together with the training data

- There are many types of supervised learning learning algorithms

  These include: Artificial Neural Networks, Decision Trees, Ensembles (Bagging, Boosting, Random Forests, ...), k-NN, Linear Regression, Naive Bayes, Logistic Regression (and other CRFs), Support Vector Machines (and other Large Margin Classifiers), ...

- we will look later at an example which uses supervised learning on a deep neural network

Unsupervised learning

- Basic idea: Discover unknown structure in input data

- Data clustering and dimension reduction
  - More generally: find the relationships/structure in the data set

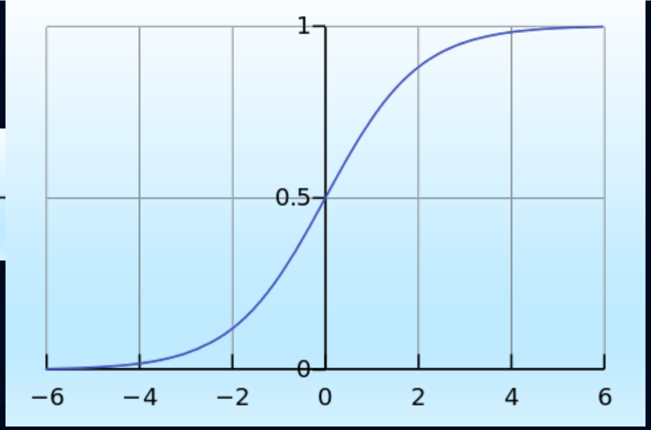
- No need for labeled data
  - The network itself finds the correlations in the data

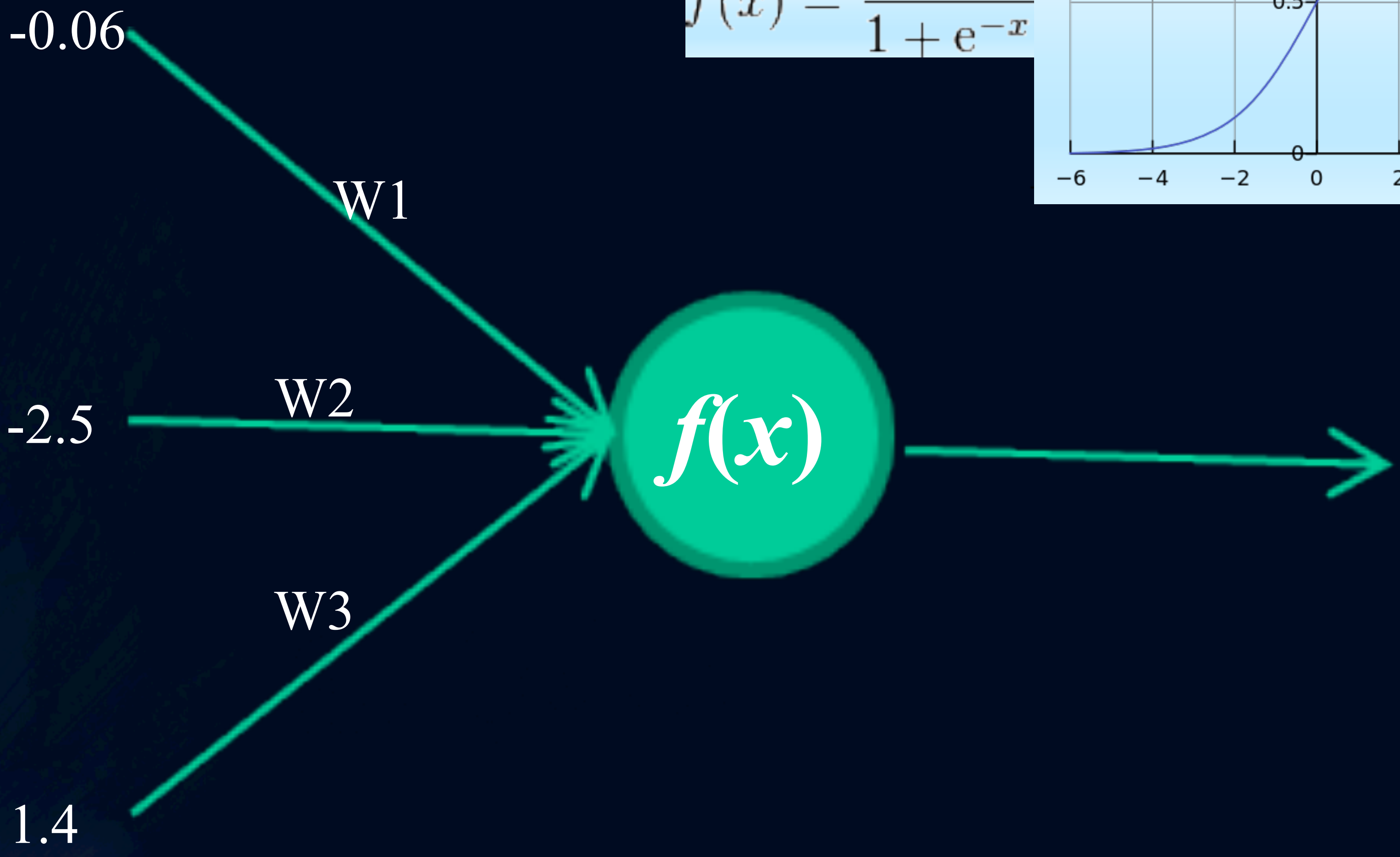
- Learning algorithms include (again, many algorithms)
  - K-Means Clustering
  - Auto-encoders/deep neural networks
  - Restricted Boltzmann Machines
    - Hopfield Networks
  - Sparse Encoders
  - …

Deep learning NN?

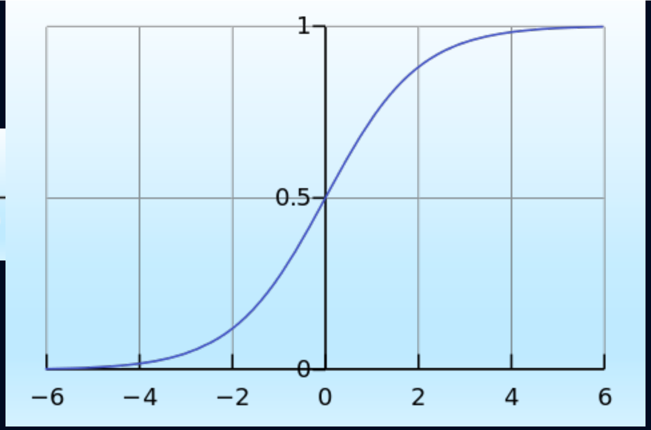
- Several hidden layers!
- Typically using convolutions to ascertain local structures
- Biological Plausibility – e.g. Visual Cortex
- Amazing results… in speech, NLP, vision/multimodal work
- Does its own feature selection!
- The big players (Google, Facebook, Baidu, Microsoft, IBM…) are doing a lot of this
- What’s new is hardware that can use these architectures at scale.
- Highly varying functions can be efficiently represented with deep architectures
  - Less weights/parameters to update than a less efficient shallow representation

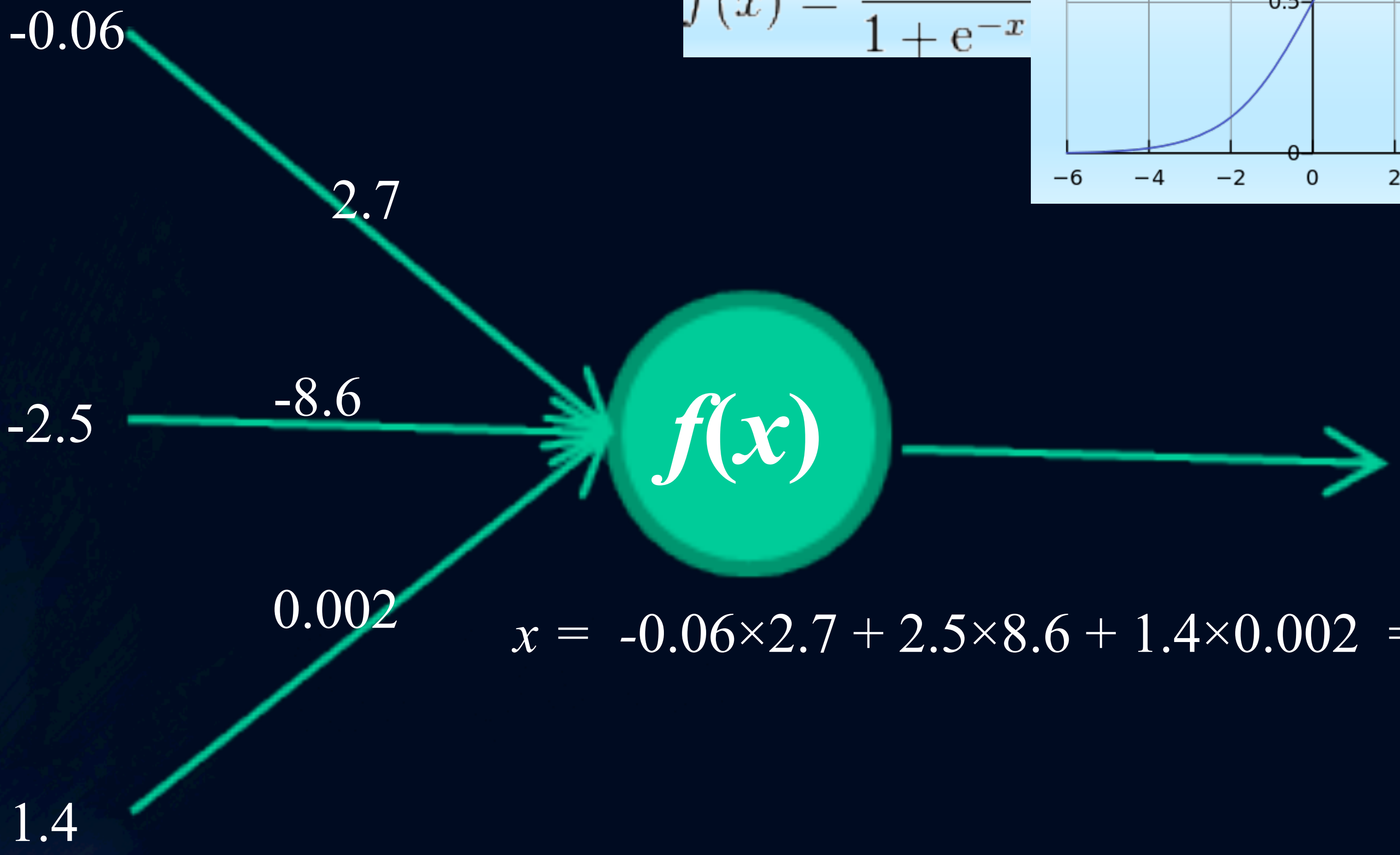
$$f(x) = \frac{1}{1+e^{-x}}$$





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



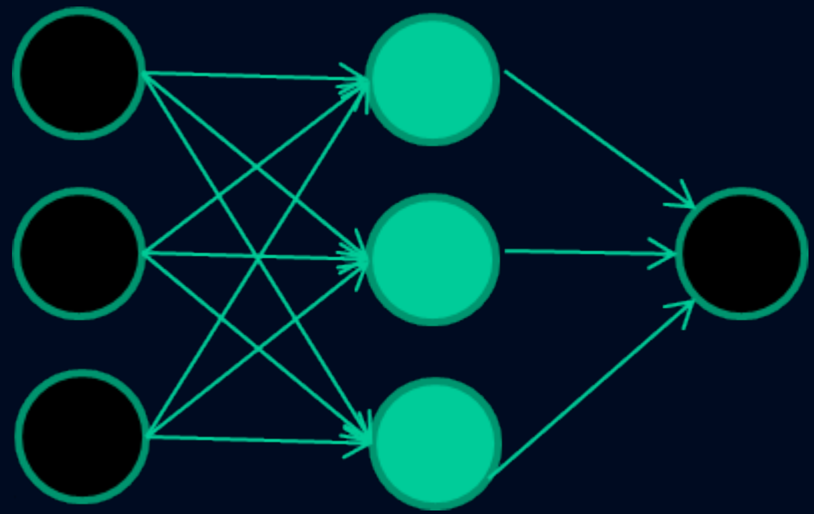


$$x = -0.06\times2.7 + 2.5\times8.6 + 1.4\times0.002 = 21.34$$

A dataset

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

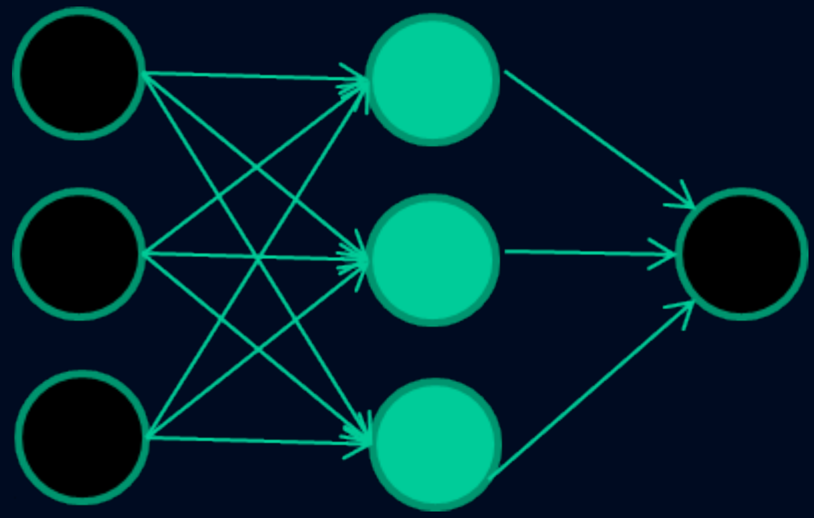
etc …



Training the neural network

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

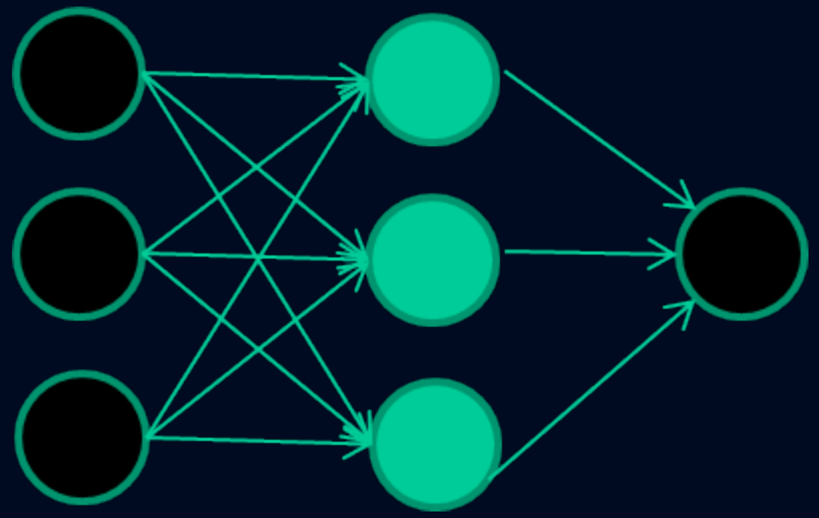
etc …



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

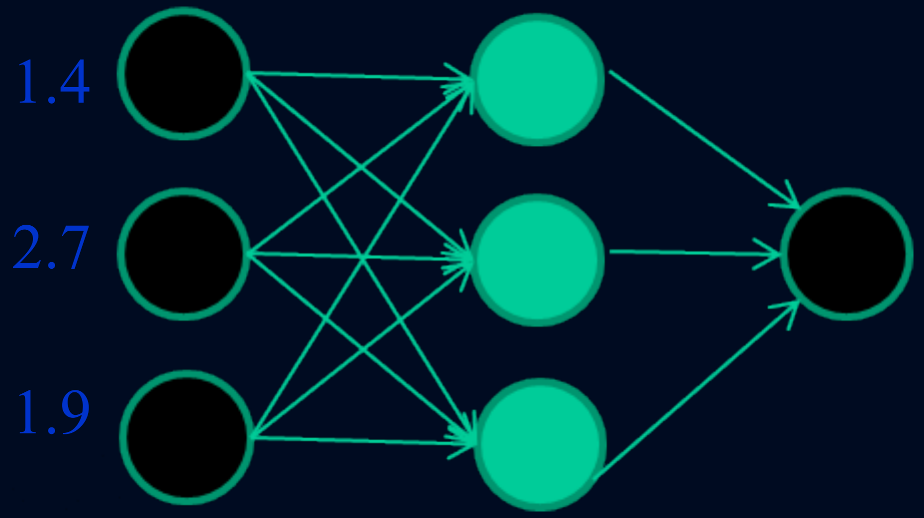
Initialise with random weights



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

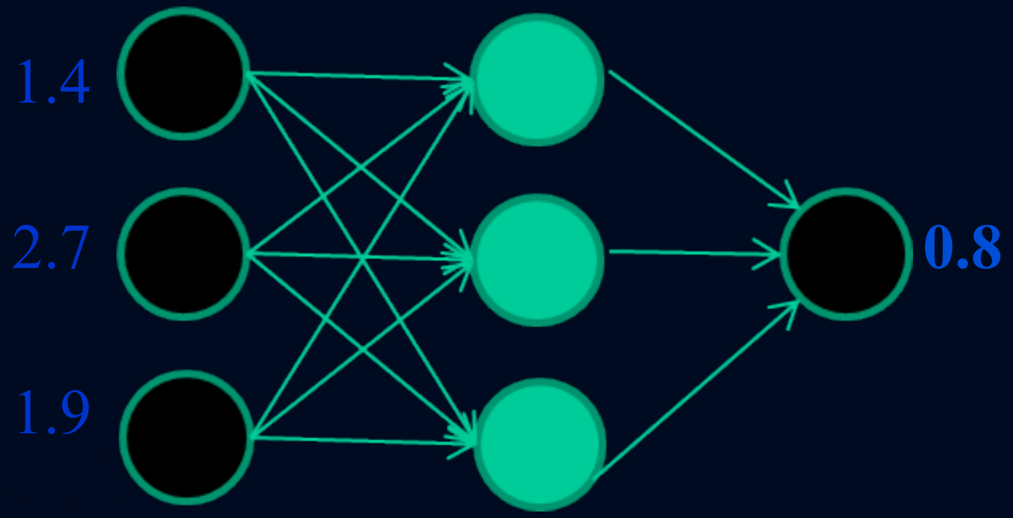
Present a training pattern



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

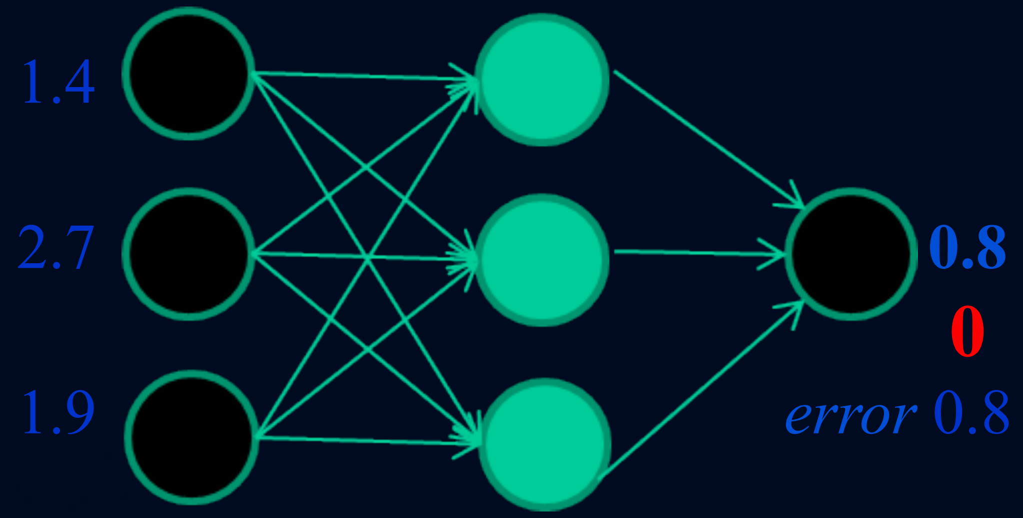
Feed it through to get output



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

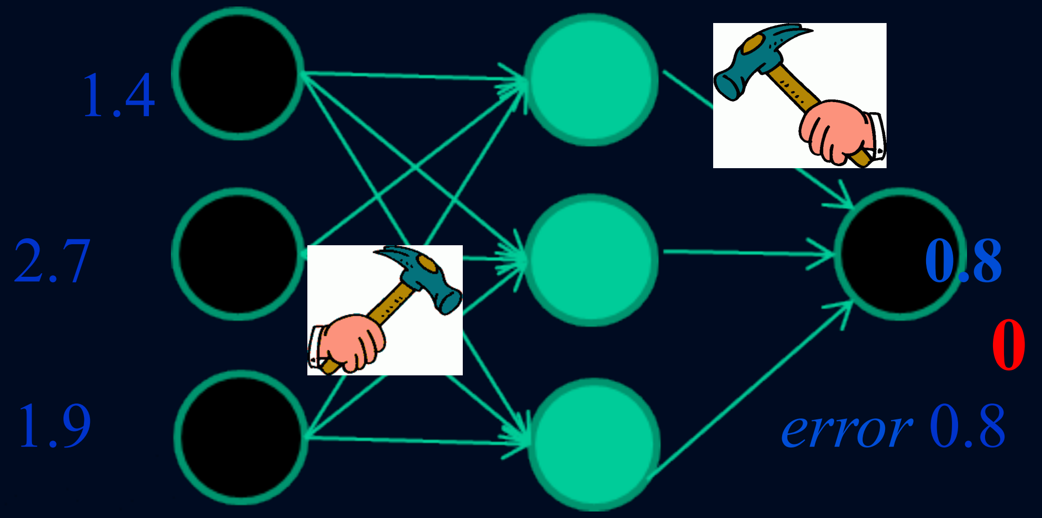
Compare with target output



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error

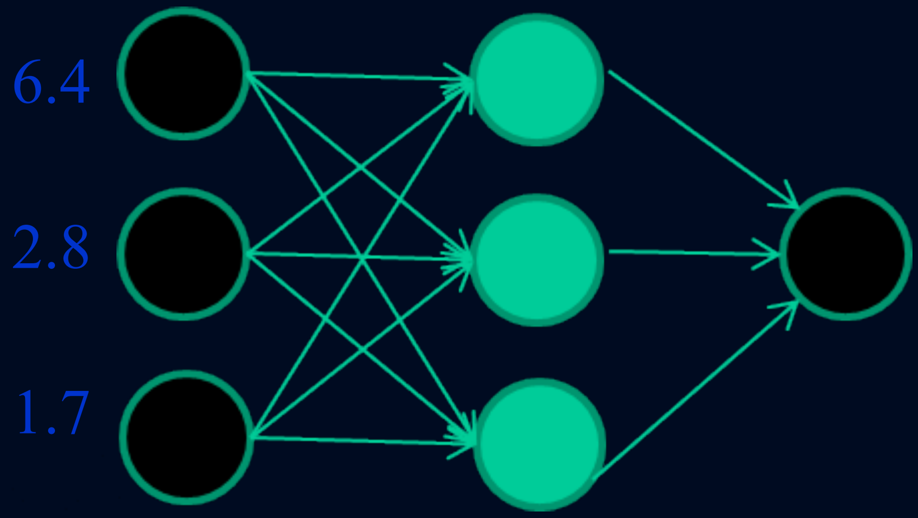


Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

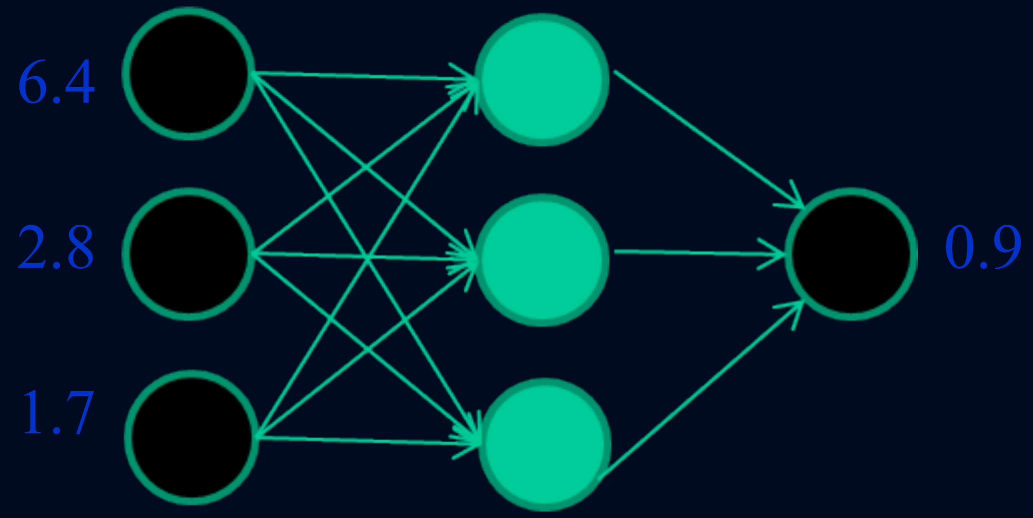
Present a training pattern



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

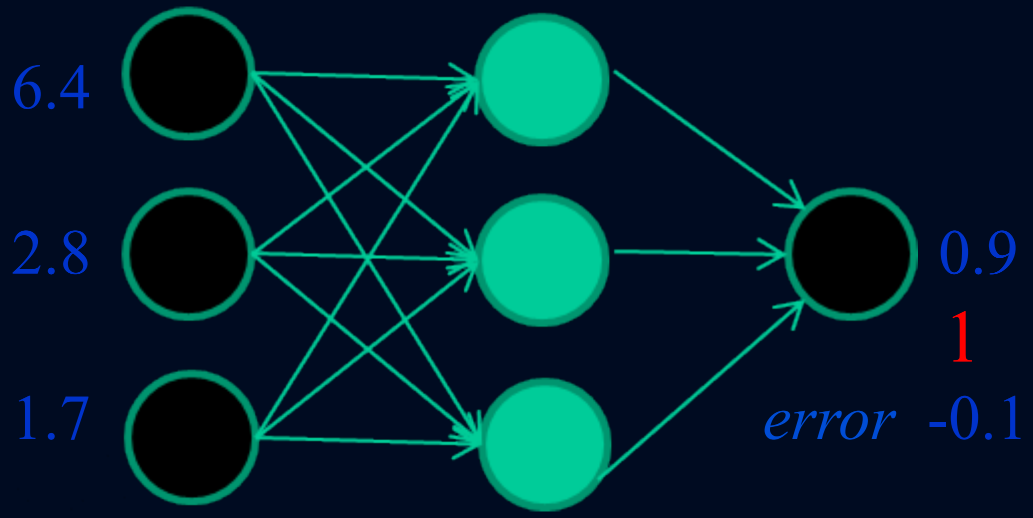
Feed it through to get output



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

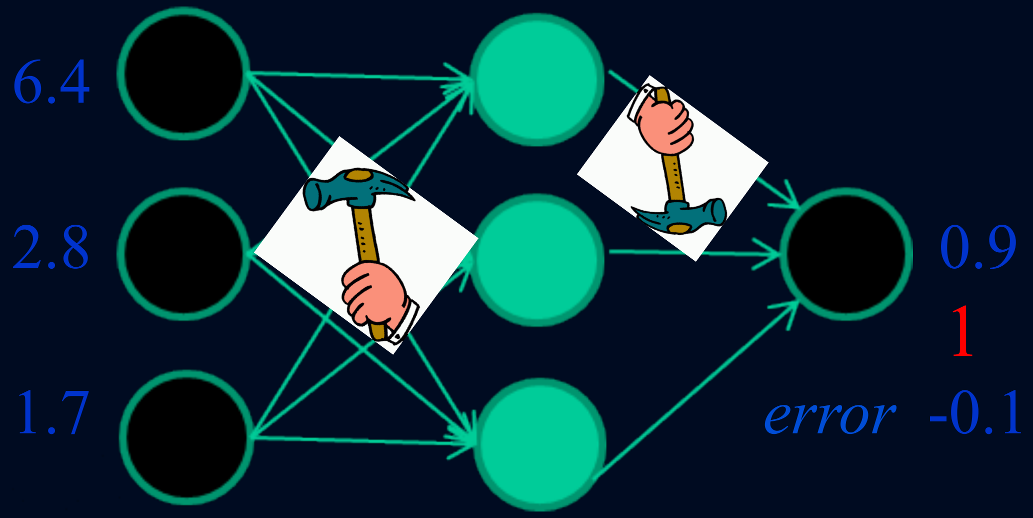
Compare with target output



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

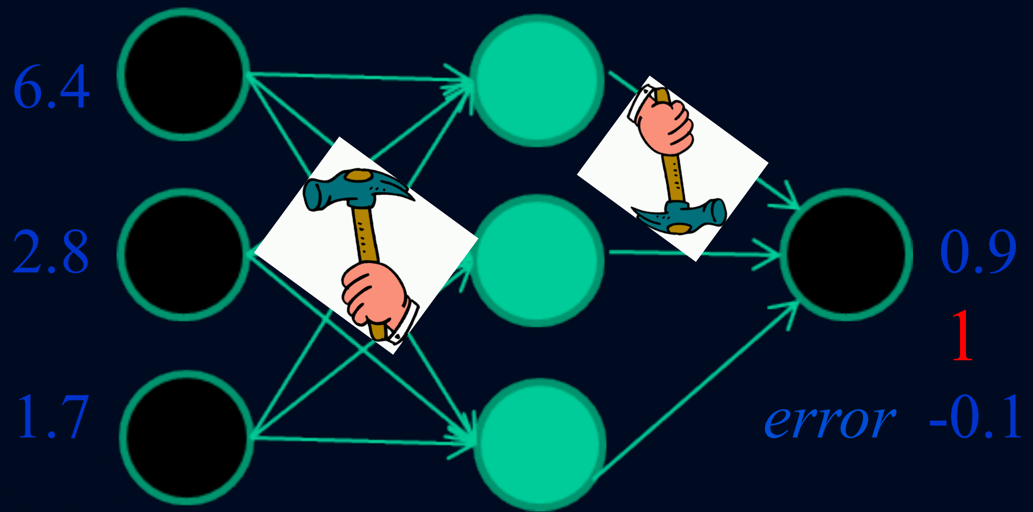
Adjust weights based on error



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

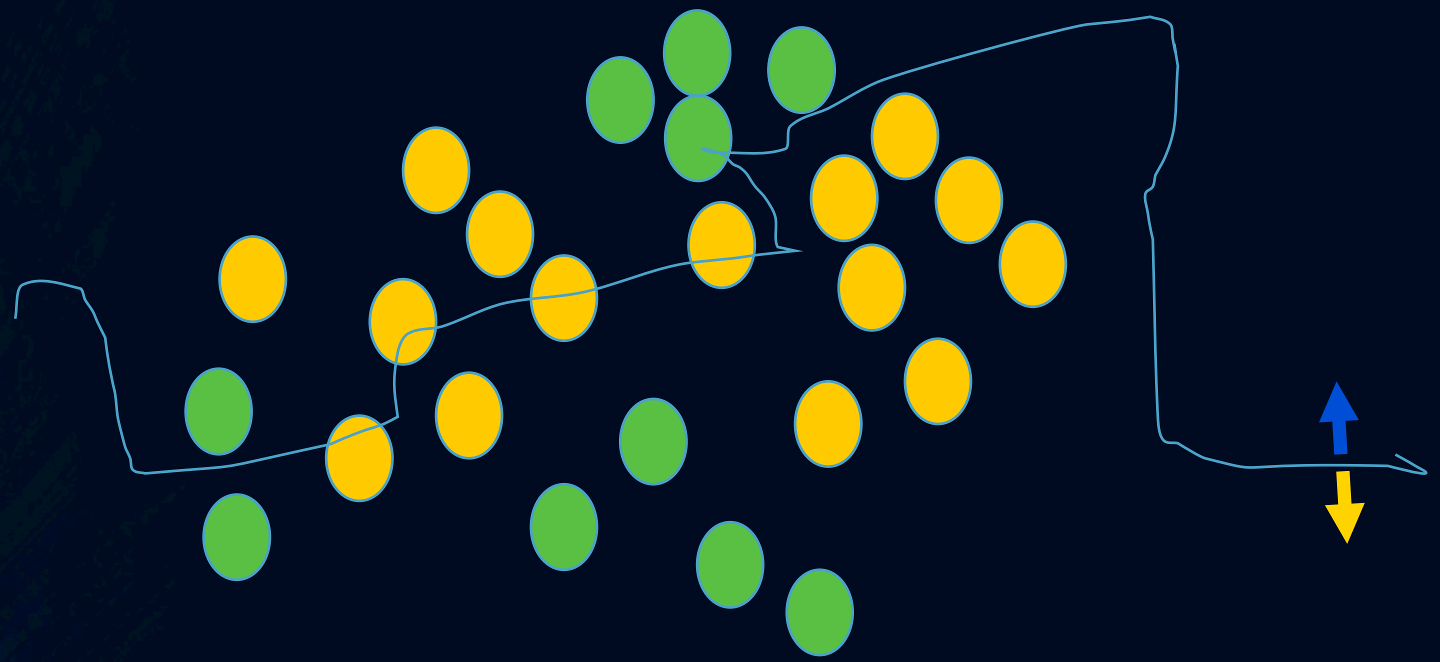
Keep repeating ….



Use Backpropagation Algorithm
to make
changes that will reduce the error

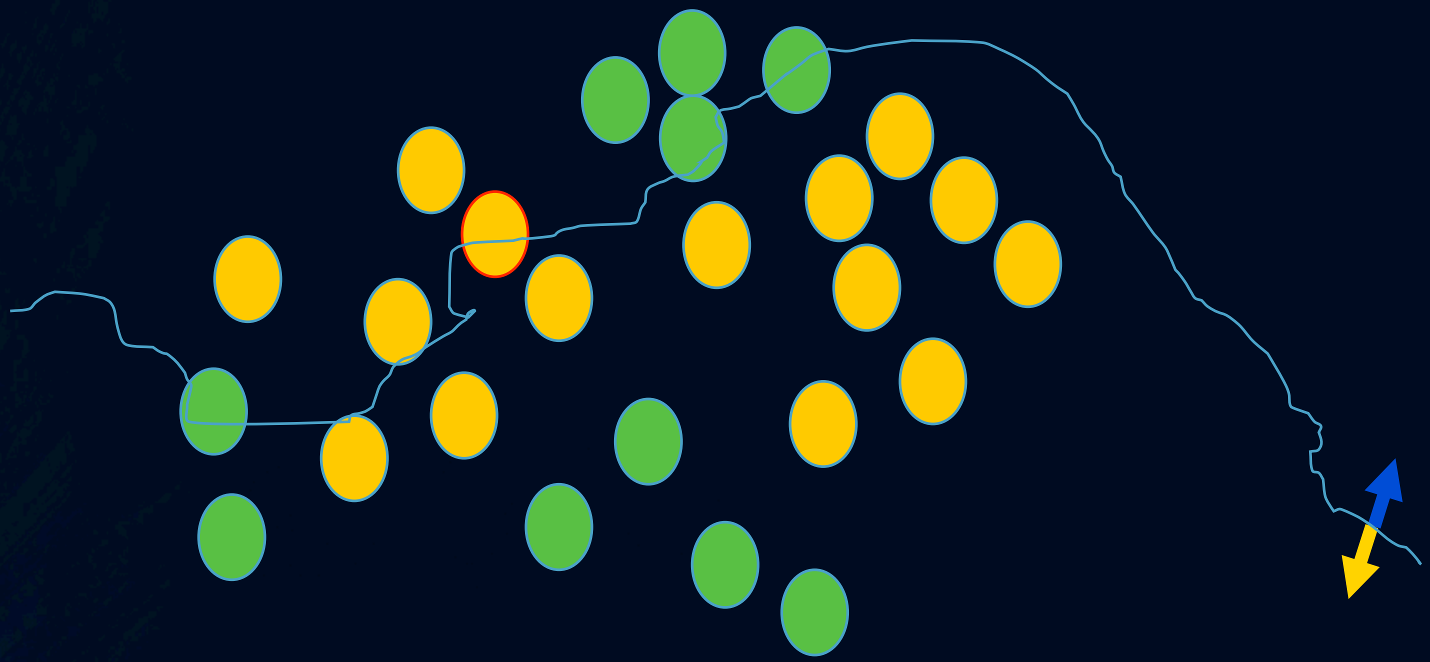
The decision boundary perspective…

Initial random weights



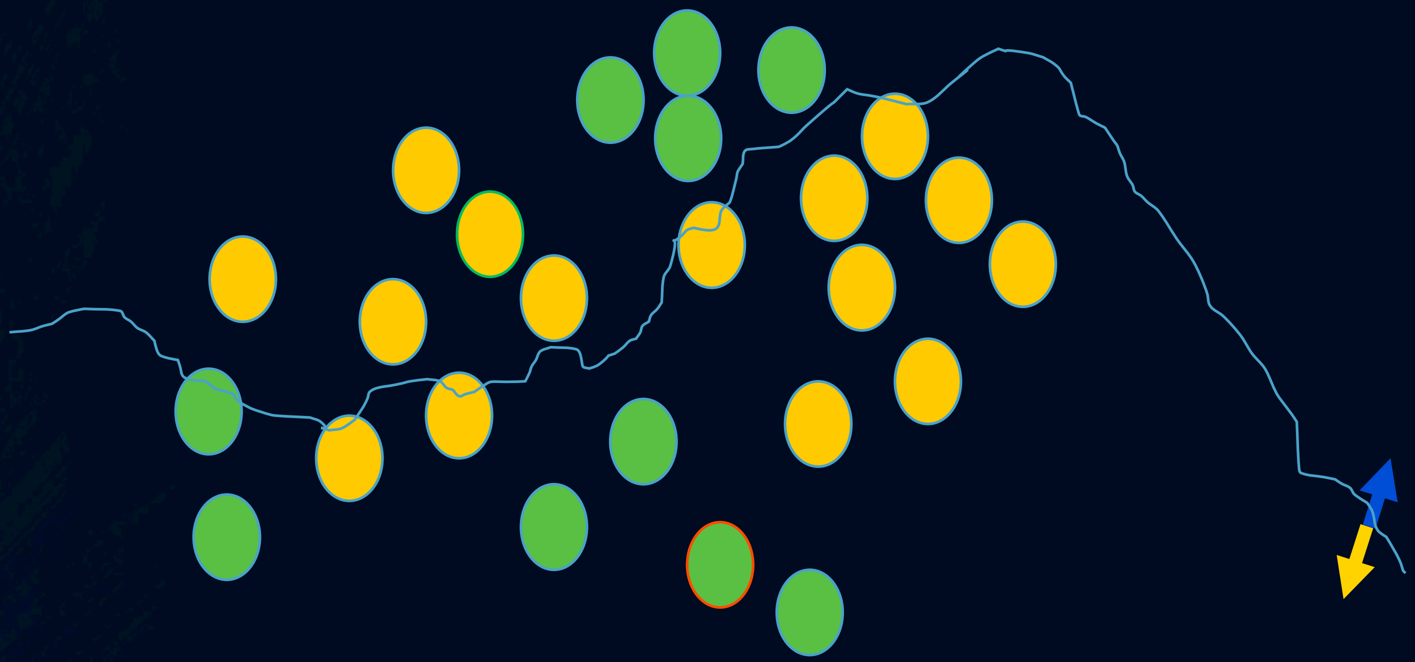
The decision boundary perspective…

Present a training instance / adjust the weights



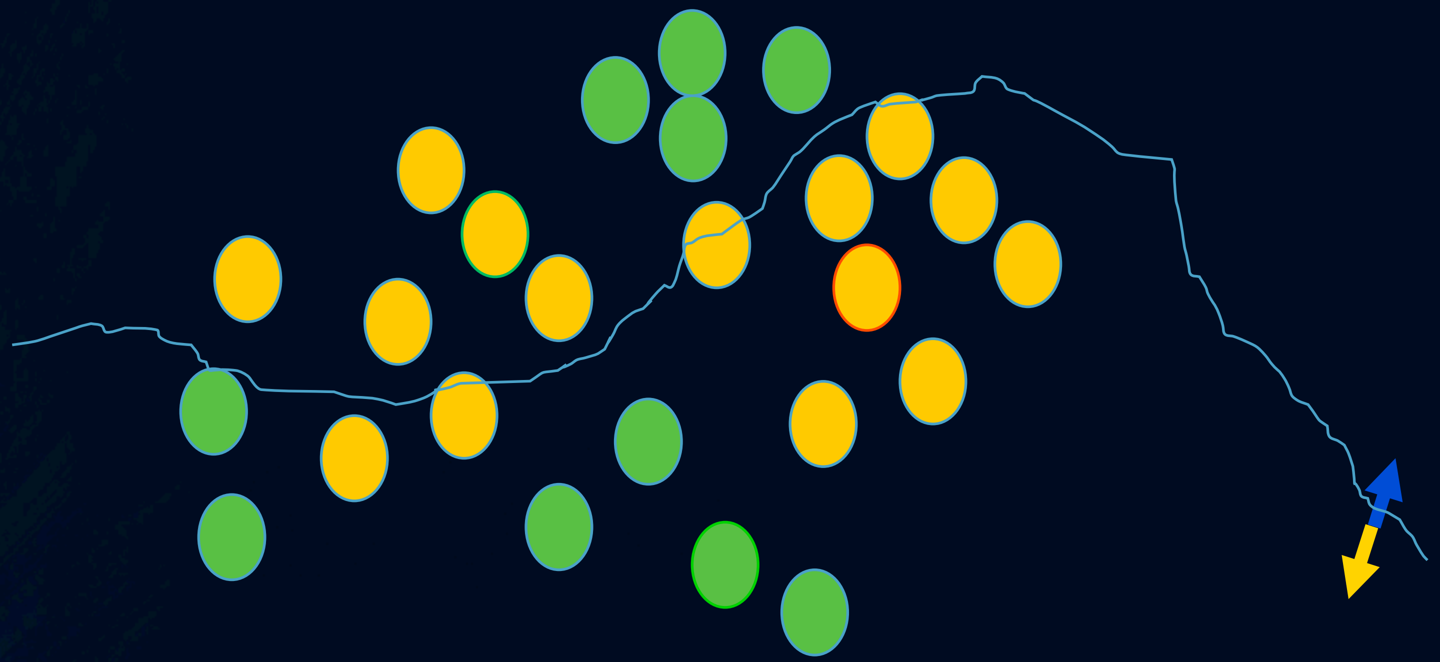
The decision boundary perspective…

Present a training instance / adjust the weights



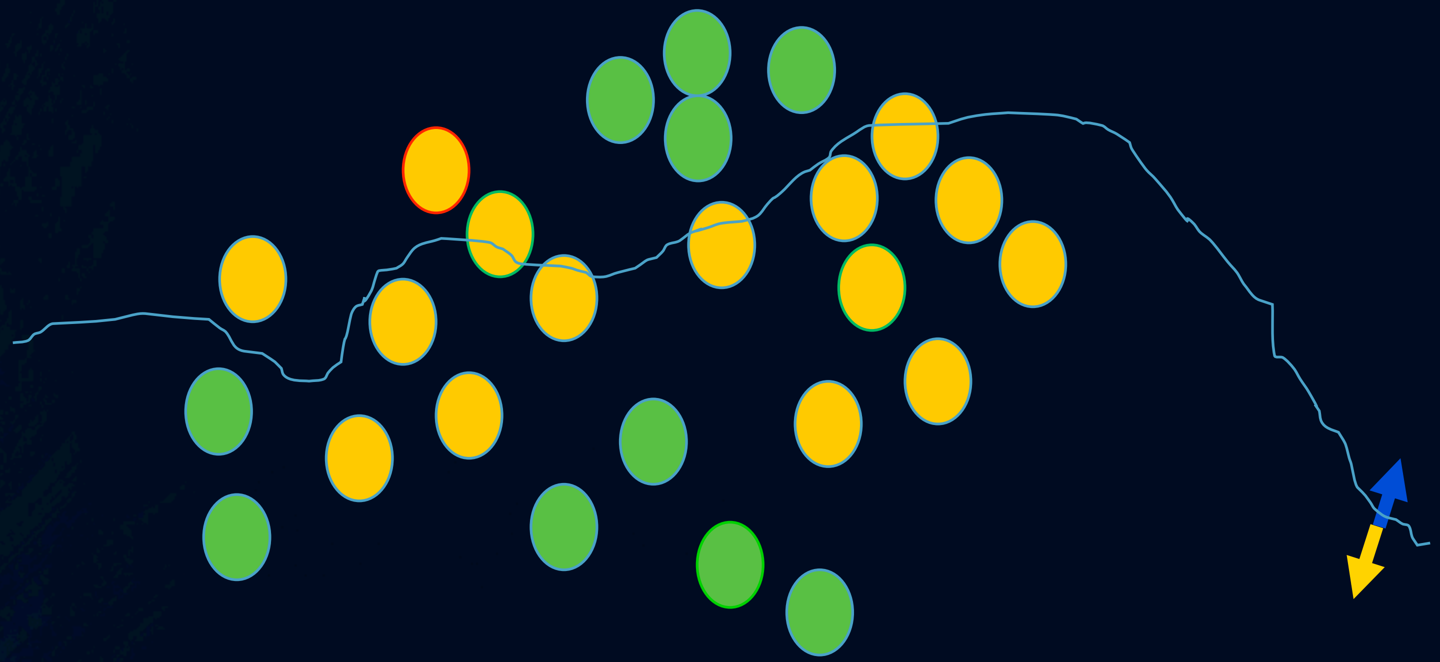
The decision boundary perspective…

Present a training instance / adjust the weights



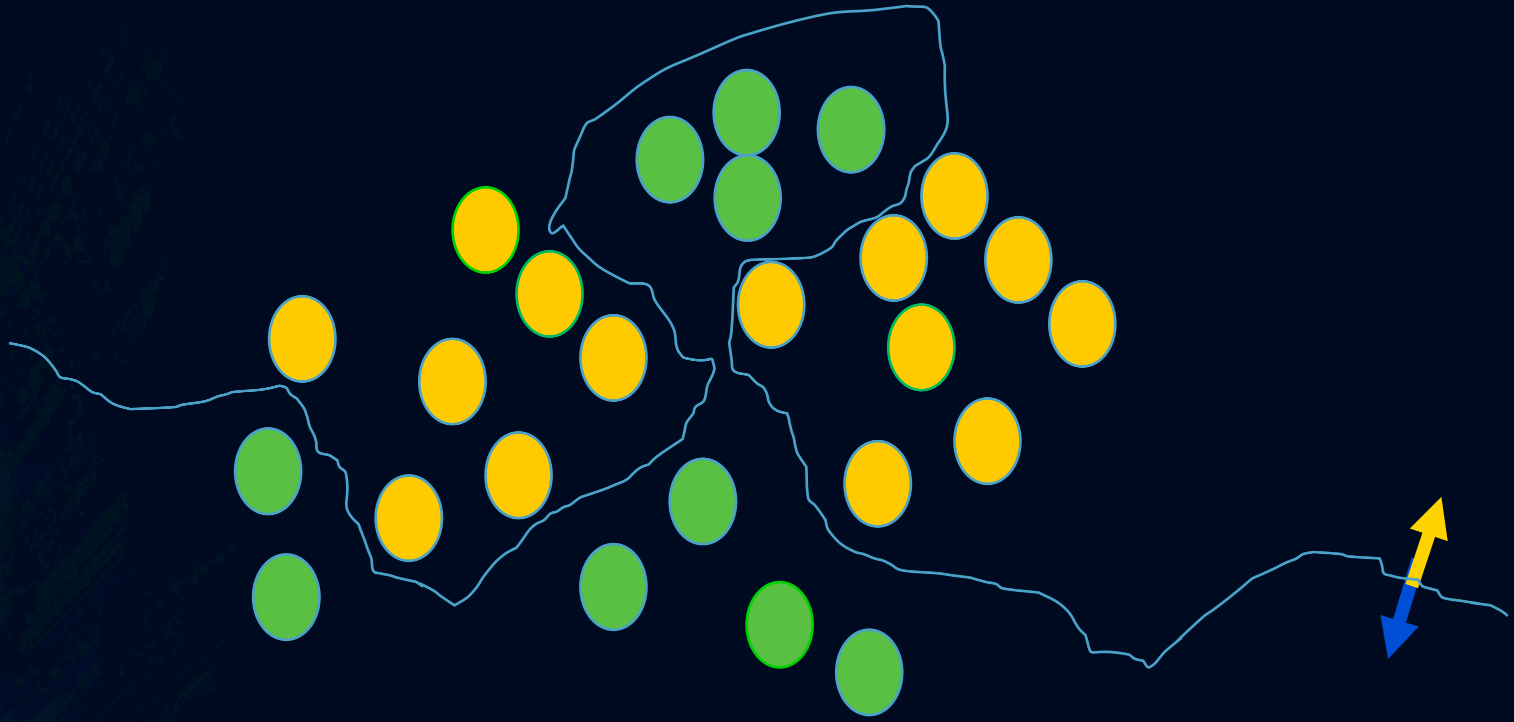
The decision boundary perspective…

Present a training instance / adjust the weights



The decision boundary perspective…

Eventually ….



Backpropagation

- Goal minimize network error:

$$E(x,w) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M}\left\| p_{n,m} - a_{n,m} \right\|^2$$

Each partial derivative of grad E is made up of derivatives of succesive activation functions and weights

$$\left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_N}\right]^T$$

IDEA: iteratively follow in the direction of the negative gradient (steepest descent direction) until we arrive at the stopping criterion:

$$\nabla_w E = 0$$

To achieve this, at each step, we update the weights based on its corresponding partial derivative

Thus the updating rule is...

$$-\gamma \frac{\partial E}{\partial w_n}$$

Gauss - Newton

- Thus the updating rule is : $$w_{m+1} = w_m - \gamma \frac{\partial E}{\partial w_m}$$

  but it can be computationally slow...

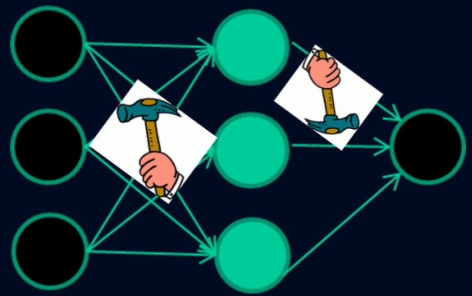
- On the other hand Gauss-Newton is computationally faster but not always stable (not always invertible H)

$$w_{m+1} = w_m - H^{-1} \frac{\partial E}{\partial w_m}$$

- We adapt it using the Levenberg-Marquardt algorithm $$w_{m+1} = w_m - (H + \mu I)^{-1} \frac{\partial E}{\partial w_m}$$

NN in general

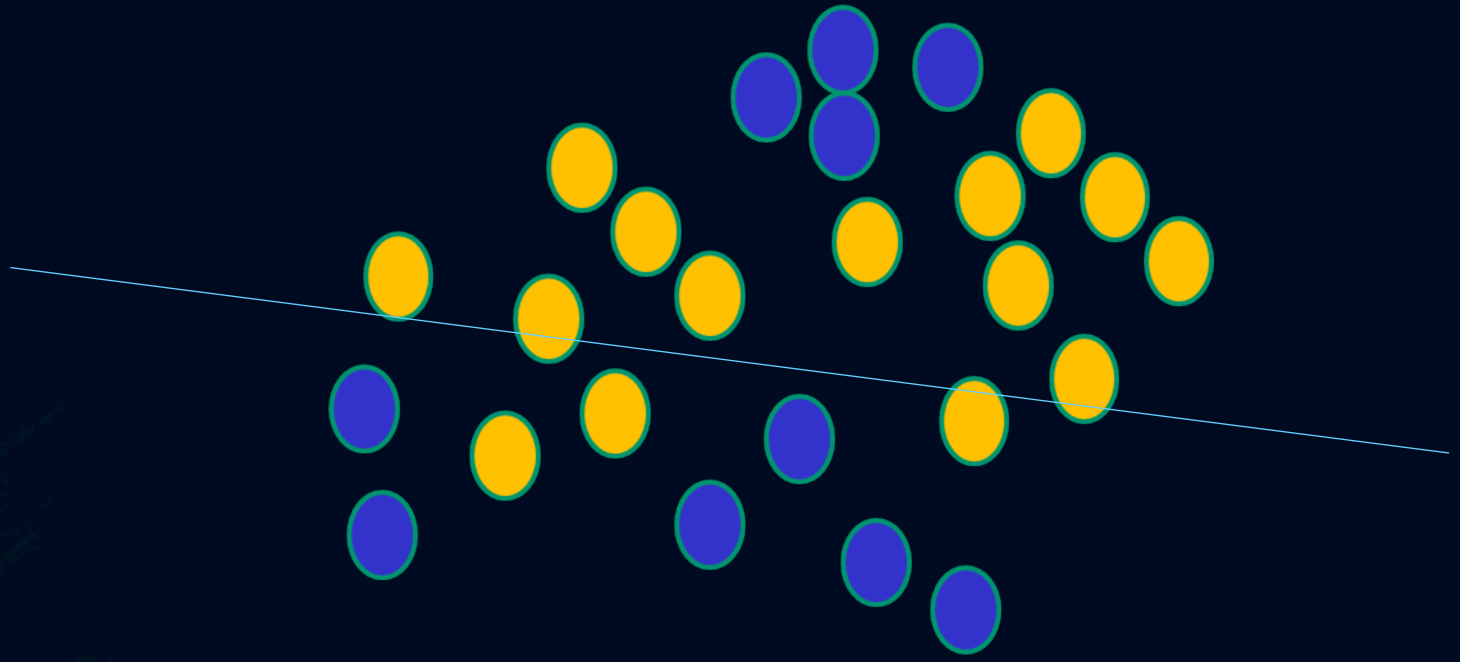
- Actually therefore NN are not very clever.
- They make thousands and thousands of mistakes from which they learn and forget each time and eventually make the network perform better
- eventually they learn and produce effective classifiers for many real applications



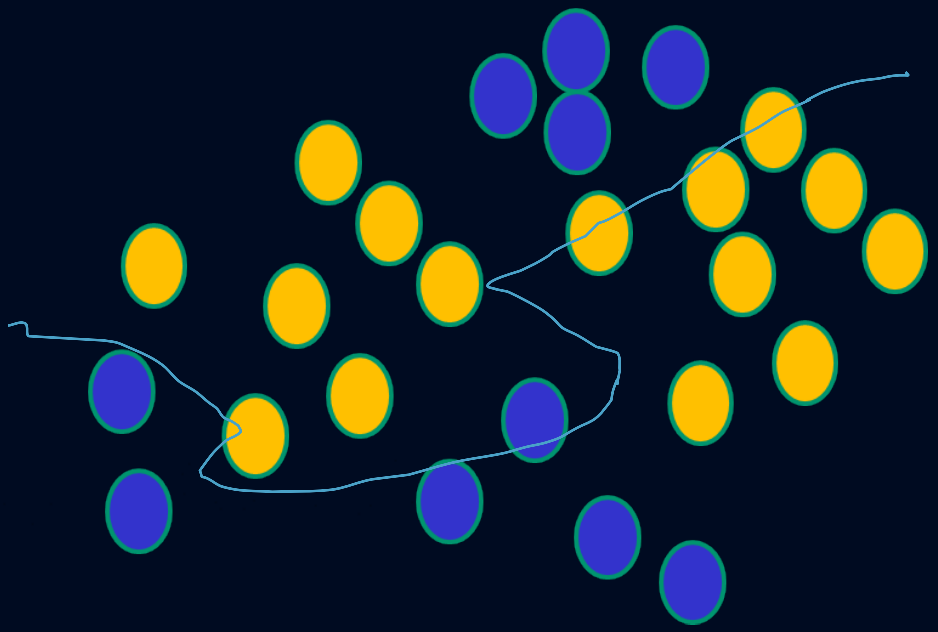
Design Issues and Challenges

- Regularization – preprocess data to avoid noisy outputs
- Over-fitting / Under-fitting
- Vanishing/Exploding Gradient

If $f(x)$ is linear, the NN can **only** draw straight decision boundaries (even if there are many layers of units)

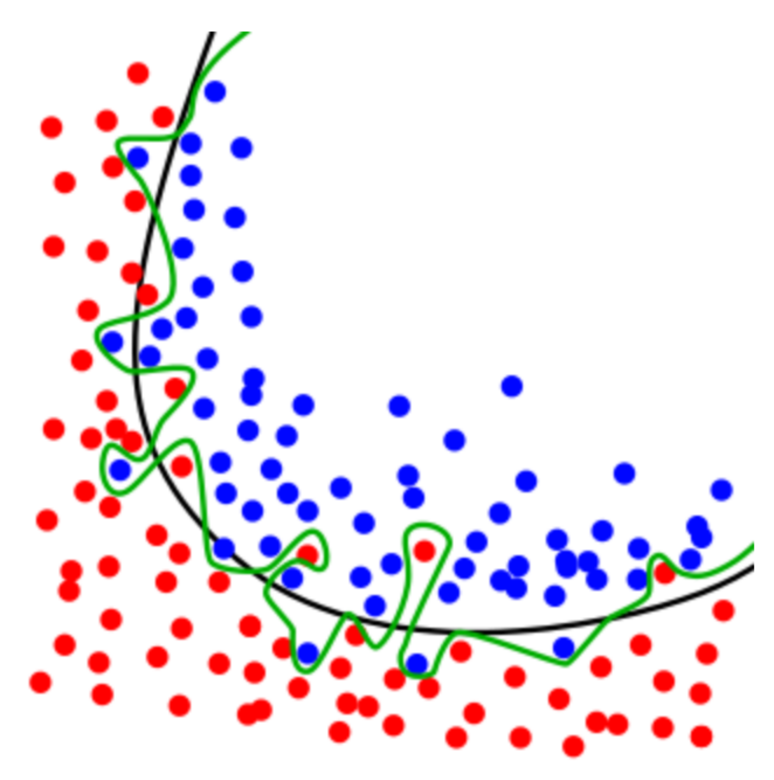


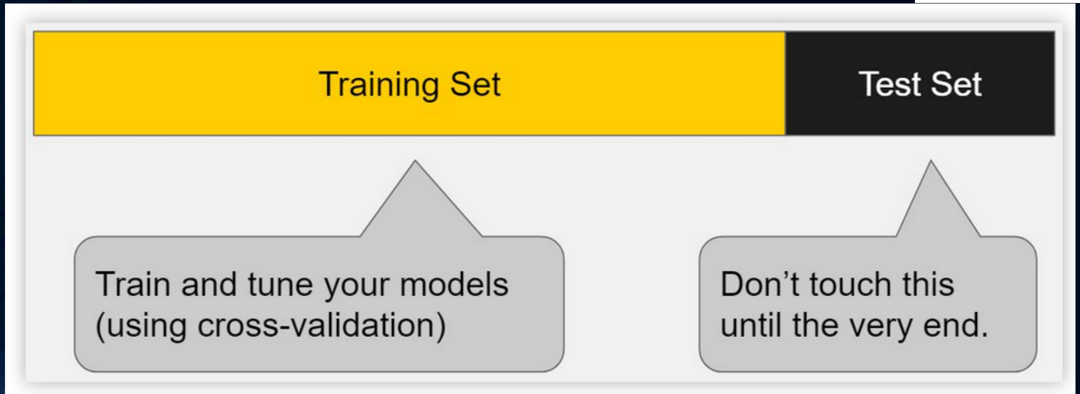
NNs use nonlinear *f*(x) so they can draw complex boundaries, but keep the data unchanged



Overfitting

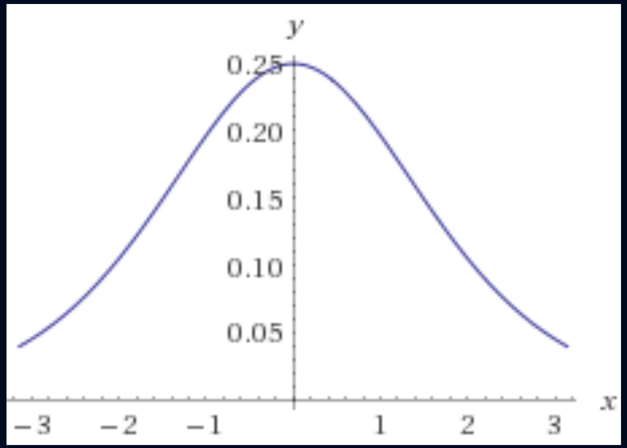
Key issue with machine learning:
How well can our model generalize from training data to new data?





The Vanishing Gradient Problem

- Deep neural networks use backpropagation.
- Back propagation uses the chain rule.
- The chain rule multiplies derivatives.
- Often these derivatives between 0 and 1.
- As the chain gets longer, products get smaller
- until they disappear.



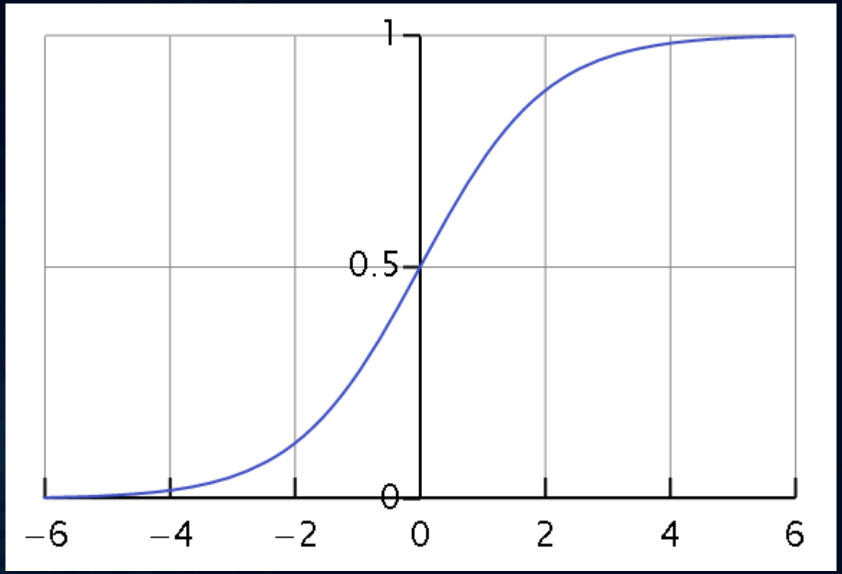
Wolfram|Alpha

Derivative of sigmoid function

Exploding Gradient

- With gradients larger than 1,
- you encounter the opposite problem
- with products becoming larger and larger
- as the chain becomes longer and longer,
- causing overlarge updates to parameters.
- This is the exploding gradient problem.

A classic activation function
Sigmoid Function

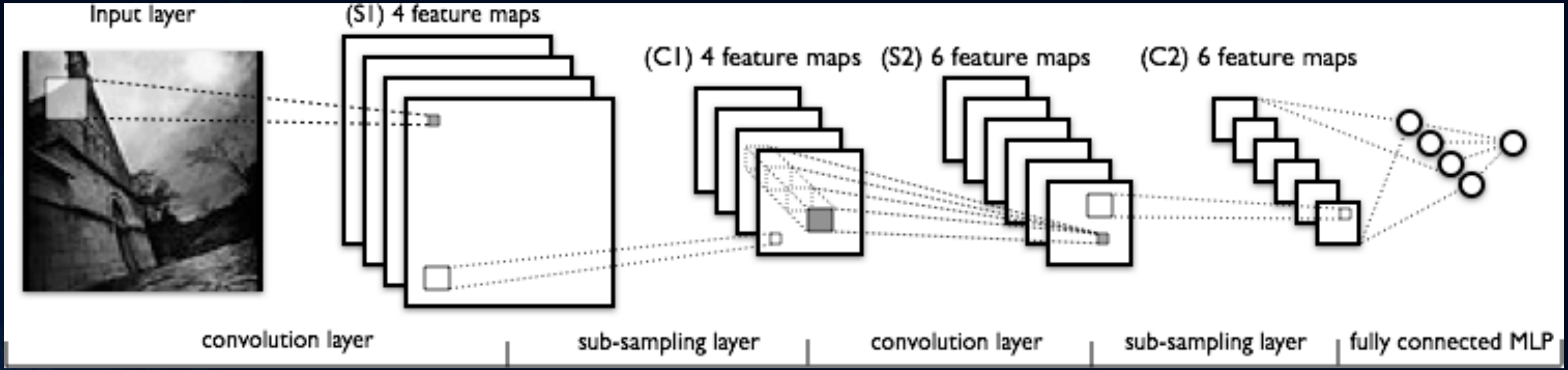


The sigmoid function ranges from 0 to 1. Most of the time, it's much closer to one extreme than the other. This can be nice if you want to classify something as either 0 or 1.

Also called the logistic function, this is a key component of logistic regression.

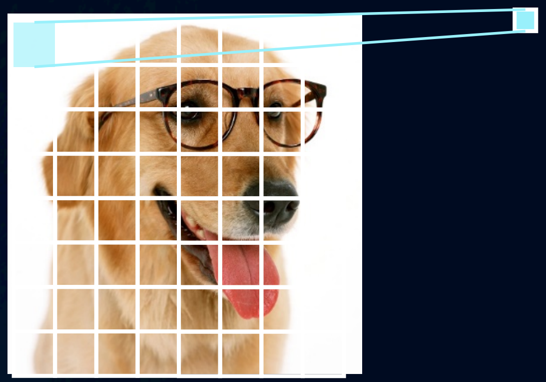
Given the shape of the curve, it's no surprise that sigmoid is a *nonlinear* function

CNN for Image Classification



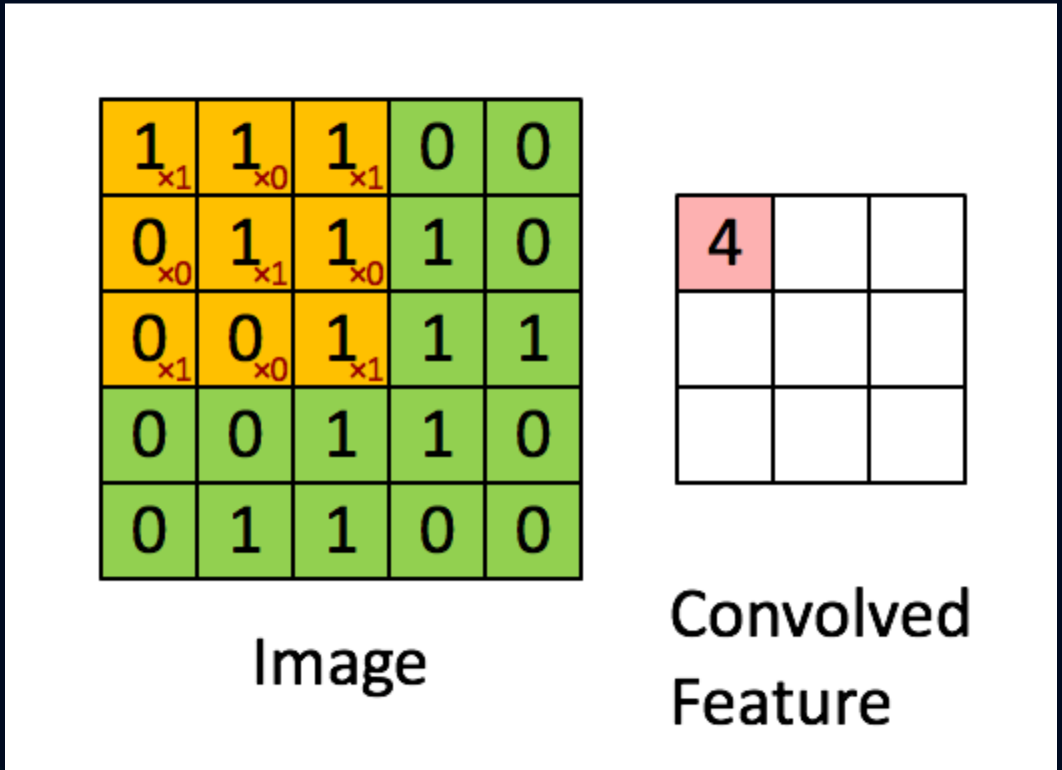
http://deeplearning.net/tutorial/lenet.html

CNN for Image Classification

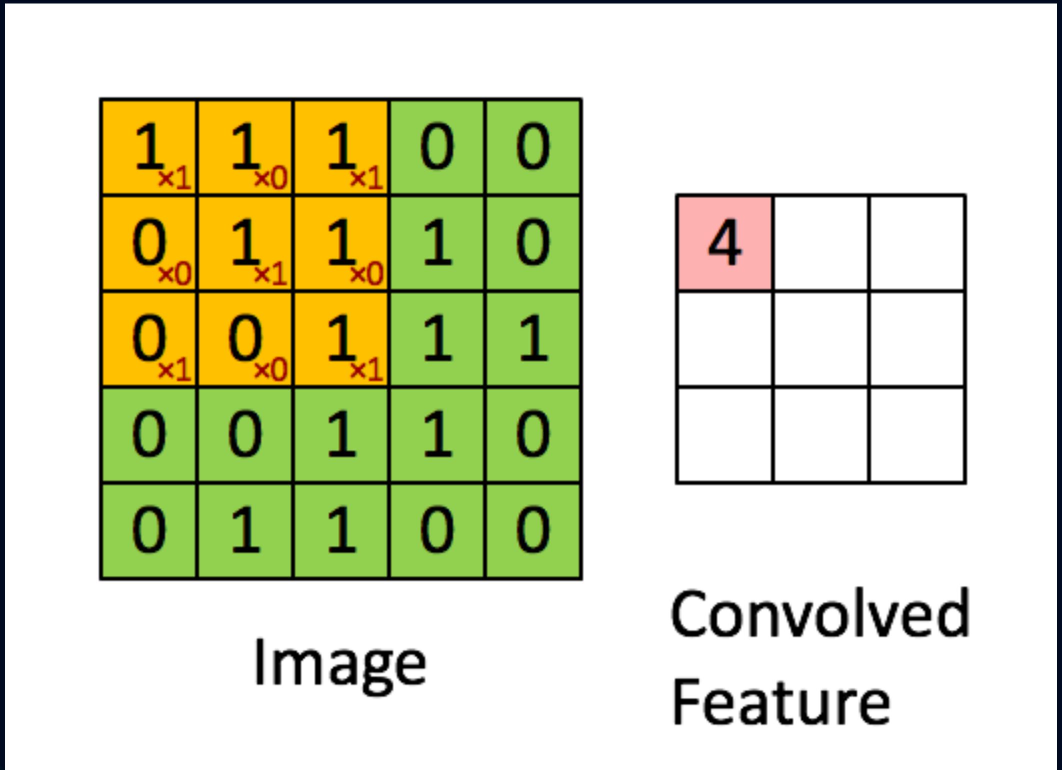


Convolutional Layer

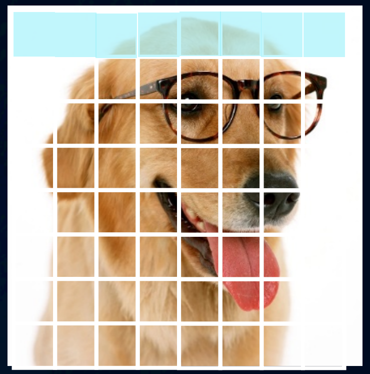
Toward a CNN - A convolution example

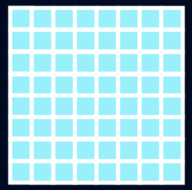


Toward a CNN - A convolution example



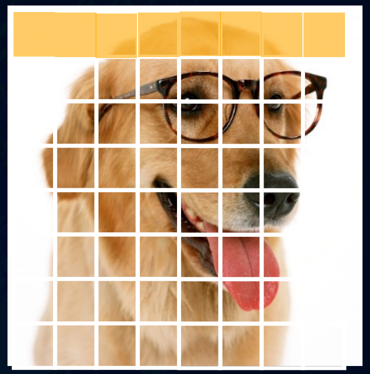
CNN for Image Classification

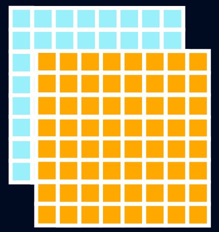




Convolutional Layer

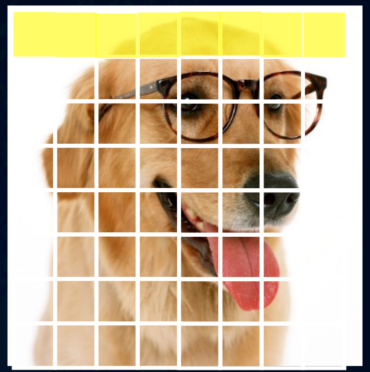
CNN for Image Classification

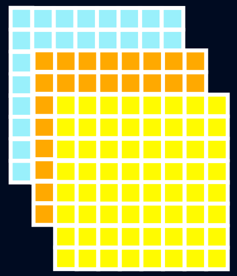




Convolutional Layer

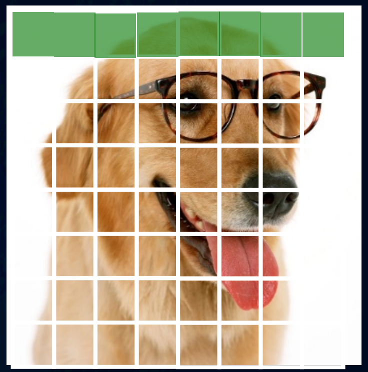
CNN for Image Classification

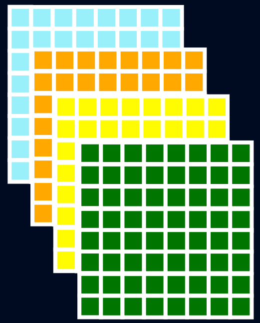




Convolutional Layer

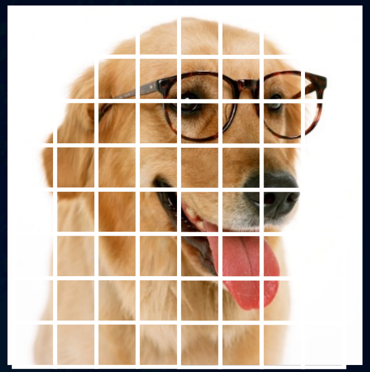
CNN for Image Classification

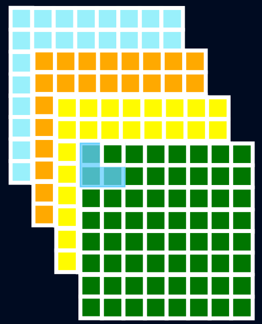


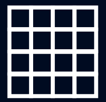


Convolutional Layer

CNN for Image Classification

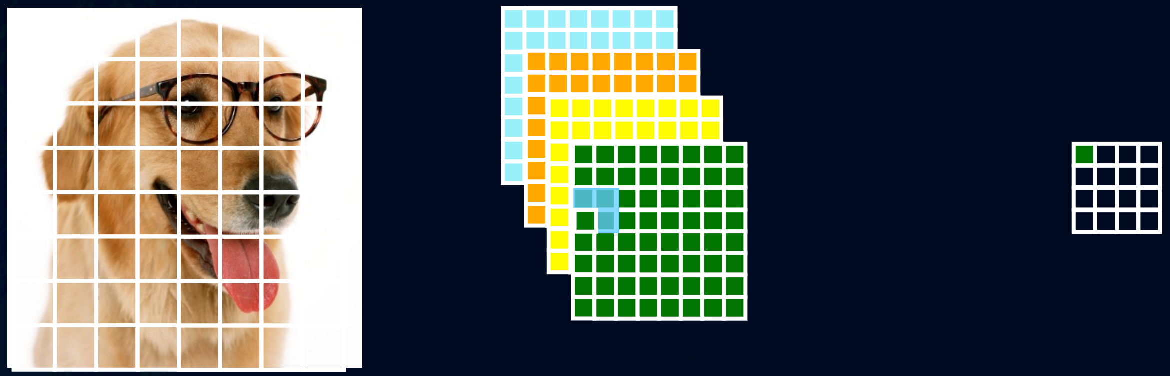






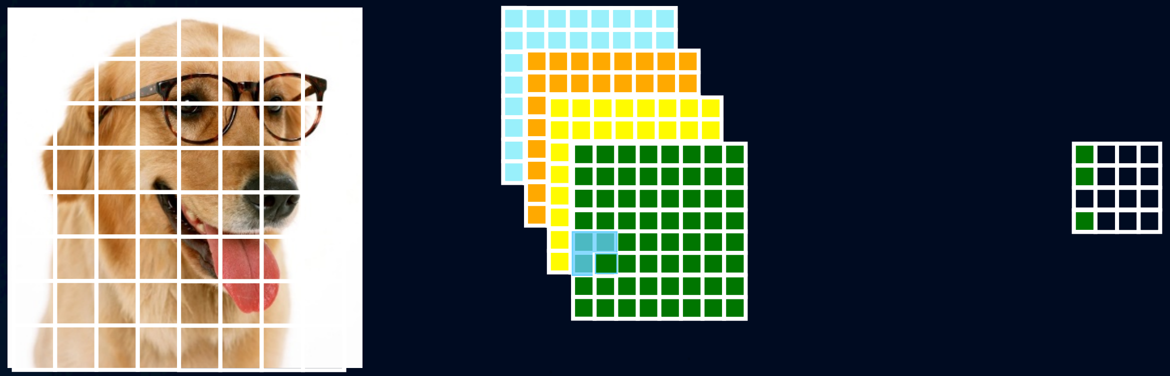
Max Pooling

CNN for Image Classification



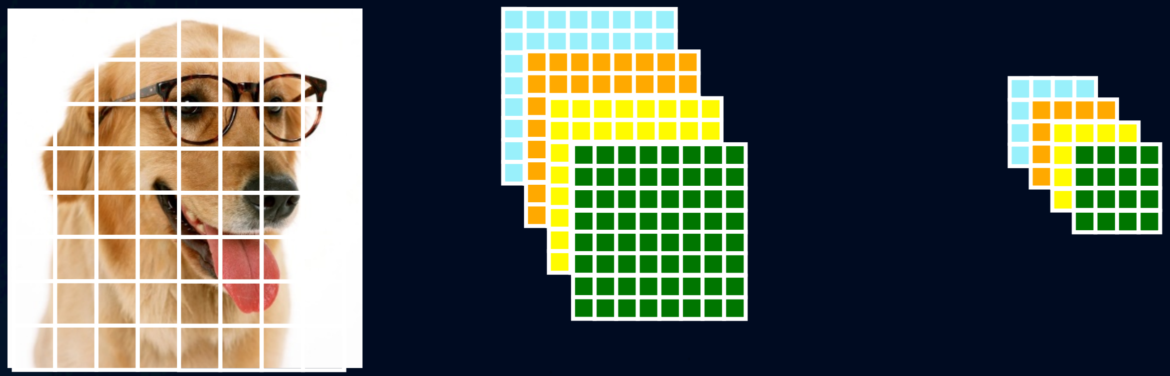
Max Pooling

CNN for Image Classification



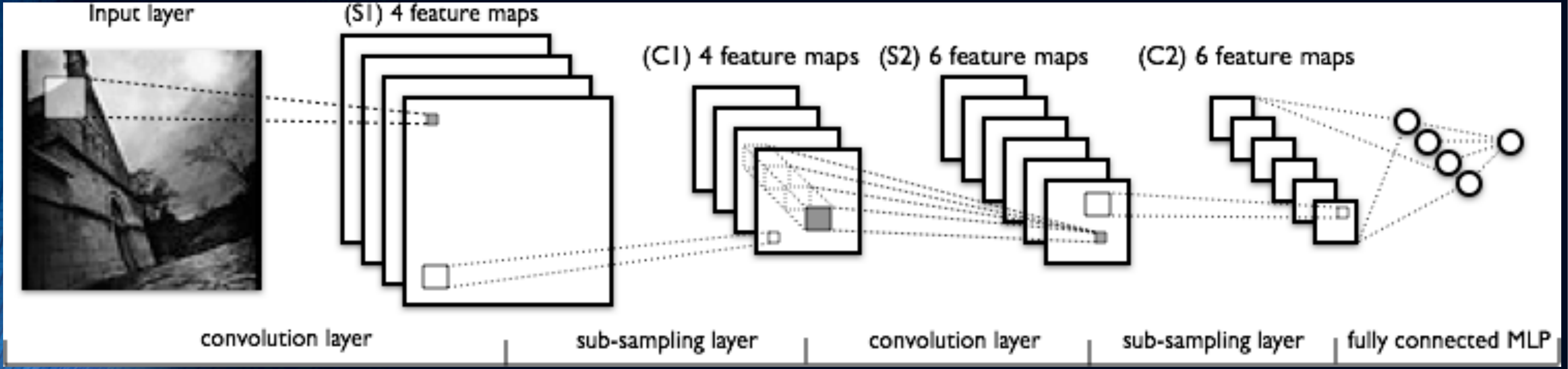
Max Pooling

CNN for Image Classification



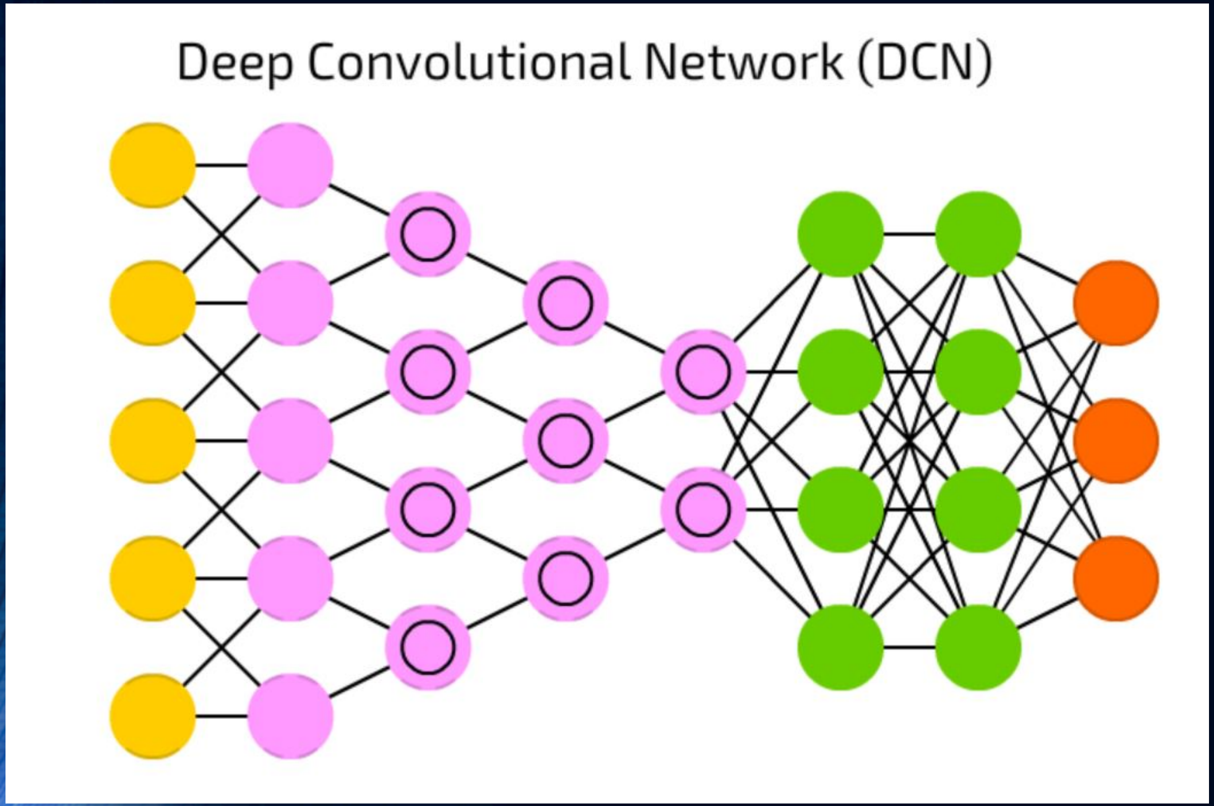
Max Pooling

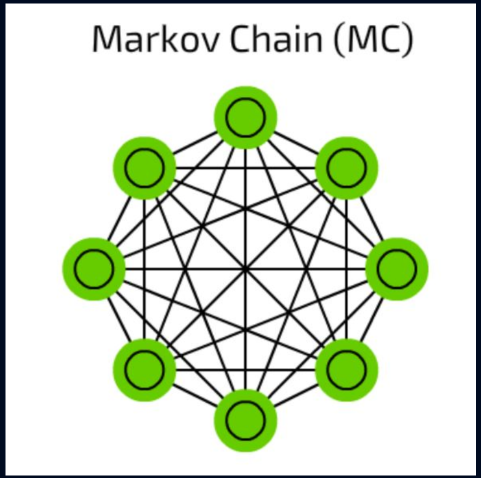
Deep CNN for Image Classification

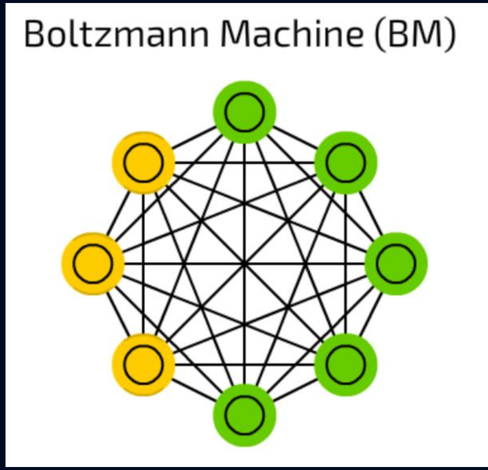


http://deeplearning.net/tutorial/lenet.html

CNN vs a Classical architectures

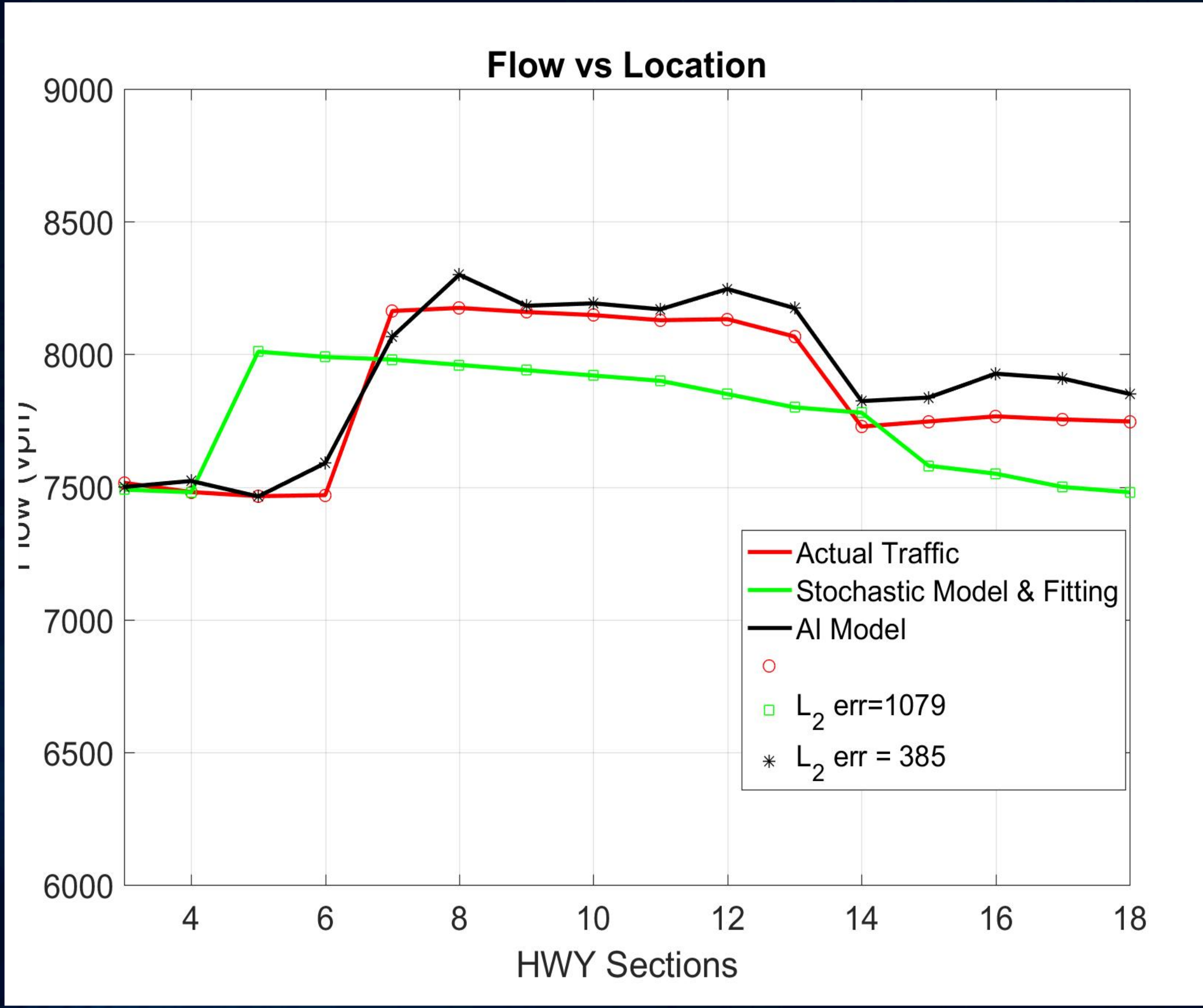






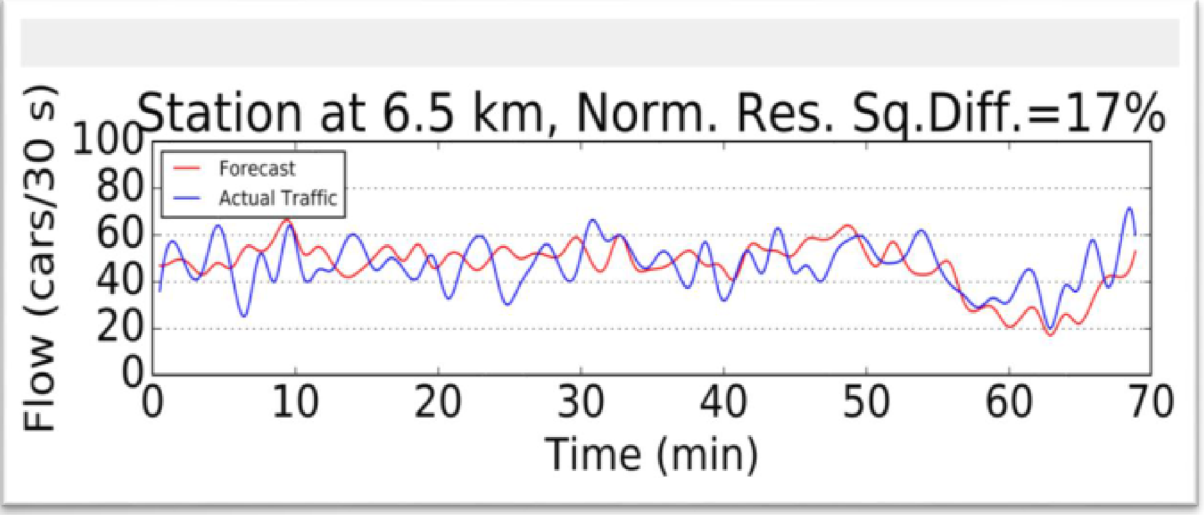
US 101

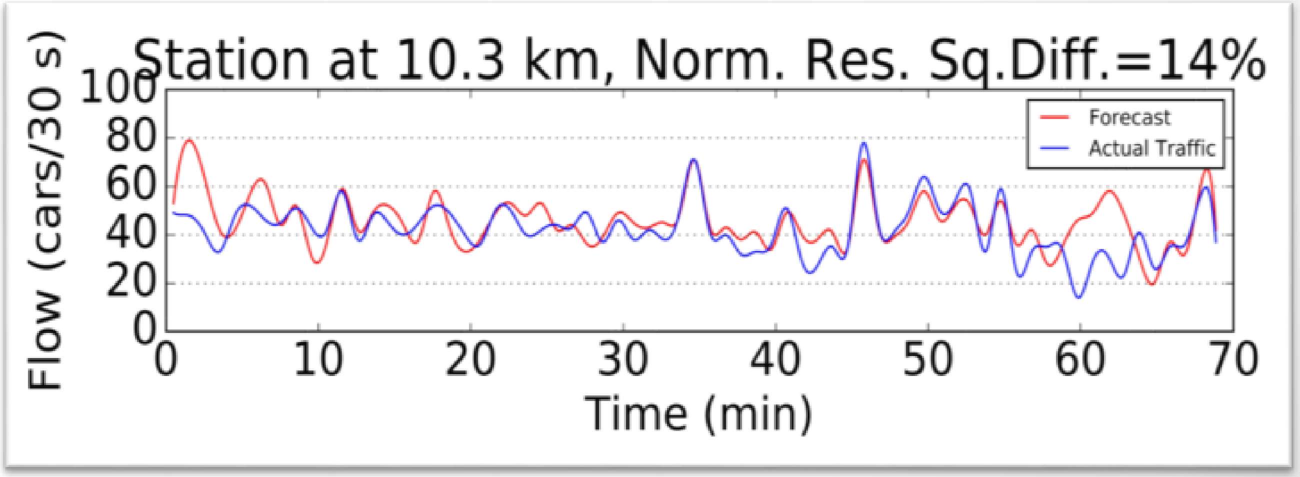
Actual data vs stochastic vs AI



US-880

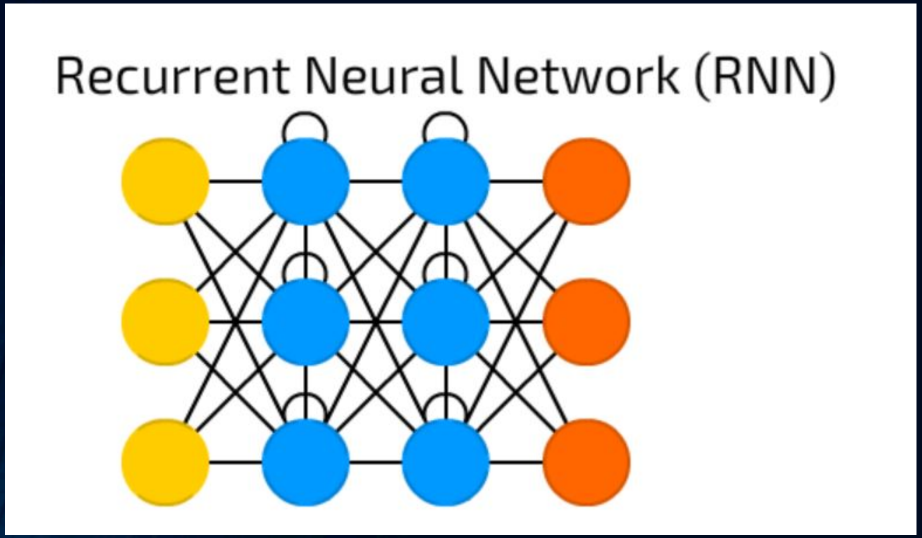
Forecasting over the next 1 hour

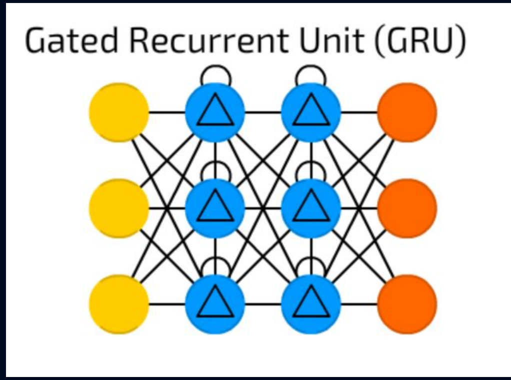


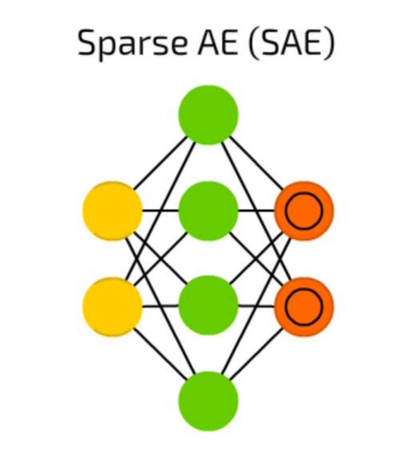


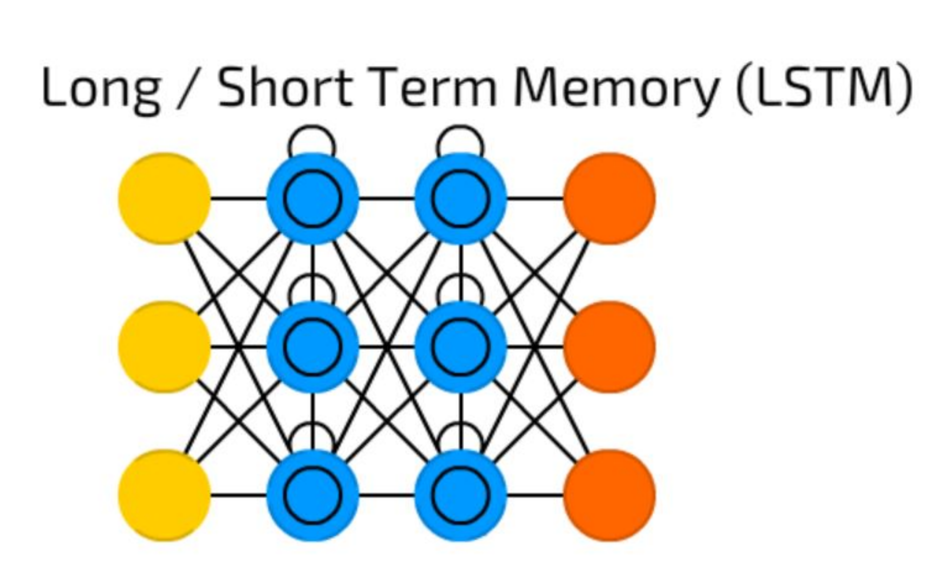
Accurate forecasting of traffic evolution for approximately the next 1 hour within a range of 10 km

Typical NN for time series type applications

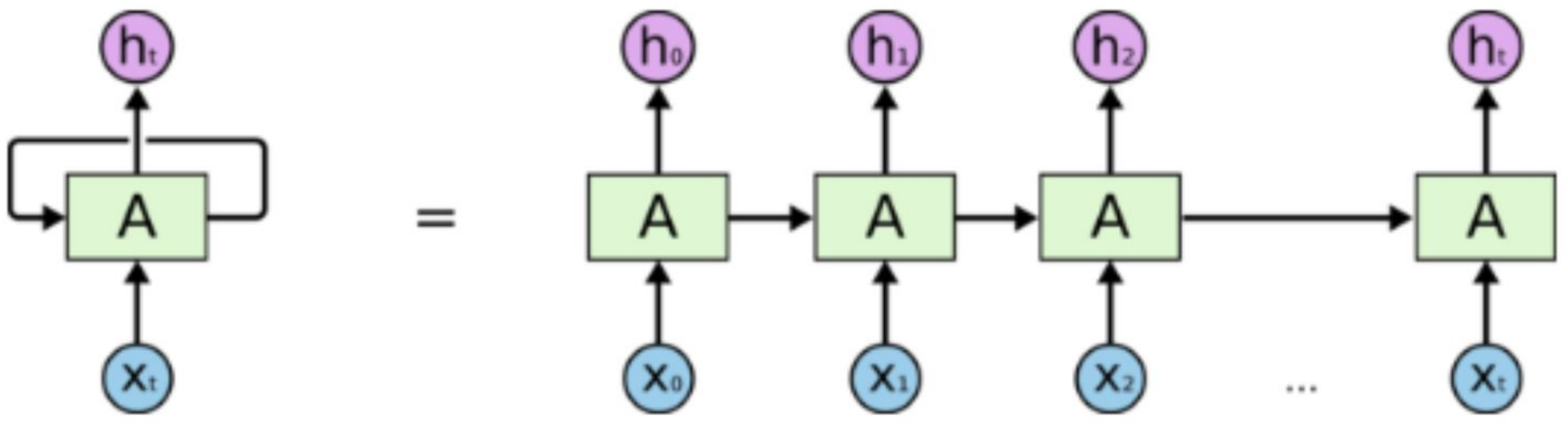








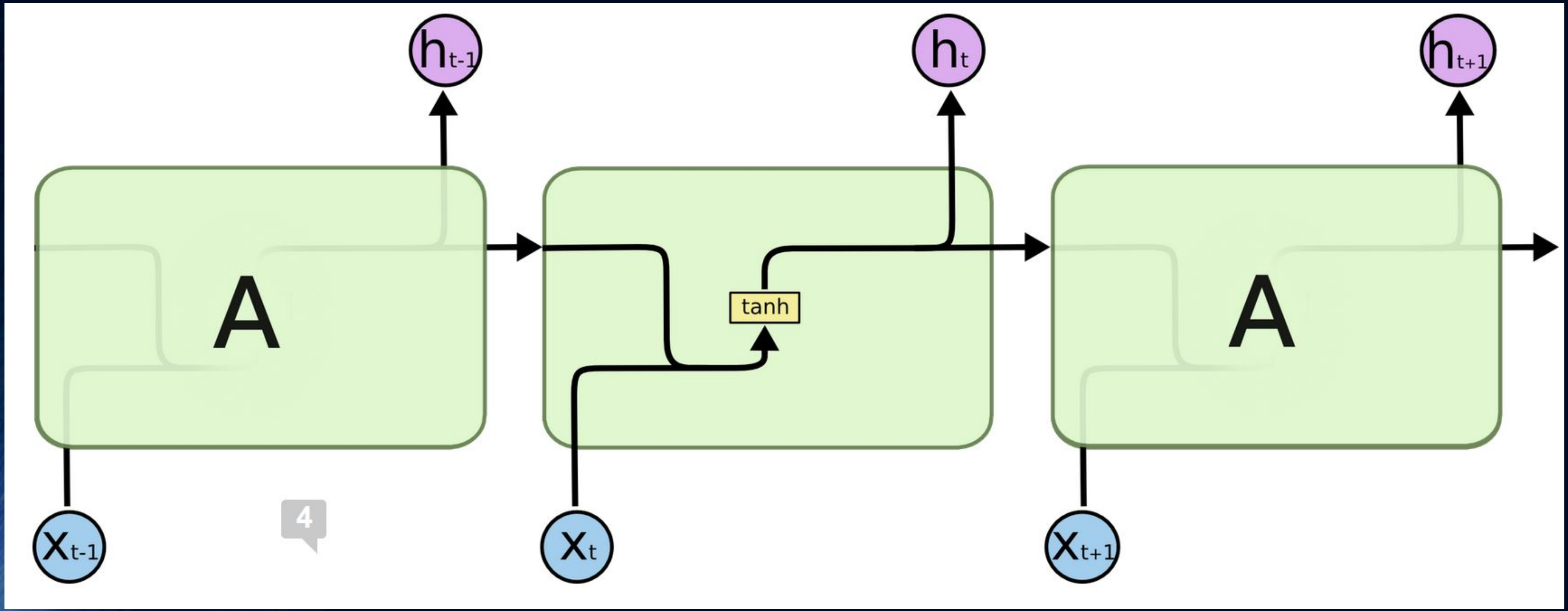
RNN



An unrolled recurrent neural network.

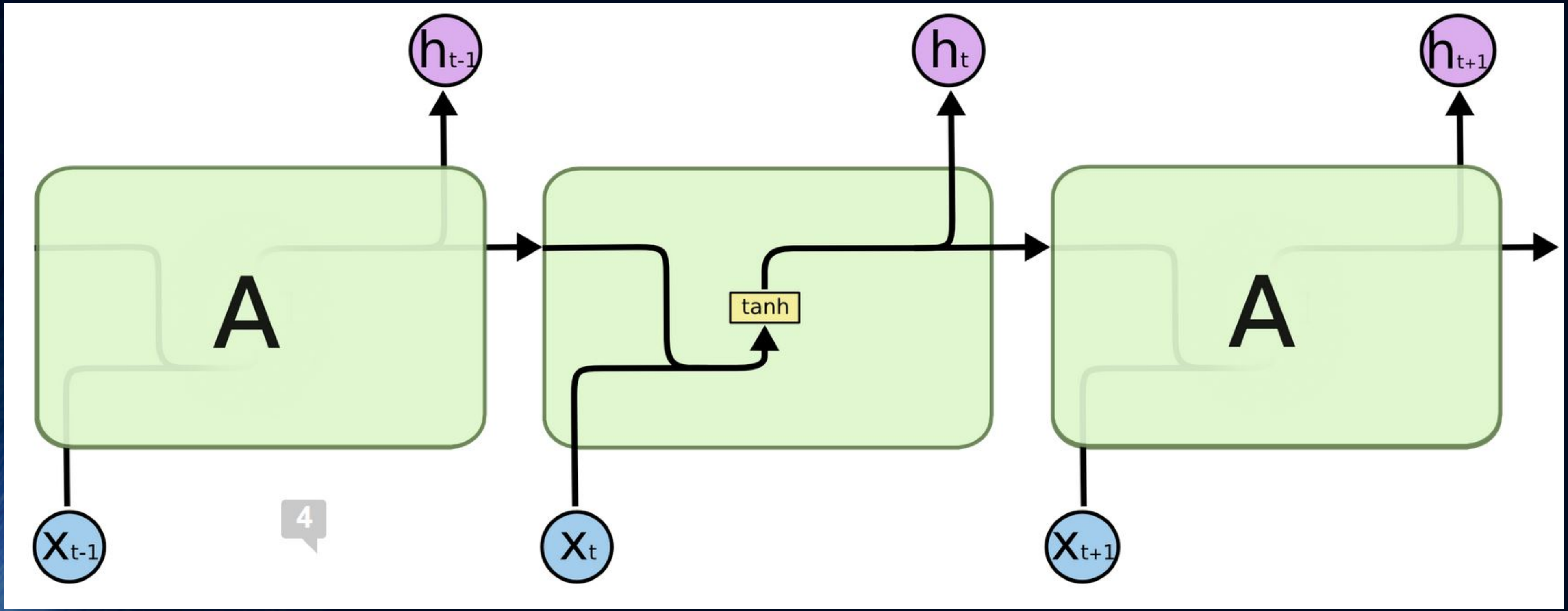
As the chain grows RNNs are not able to remember older lessons/training

RNN



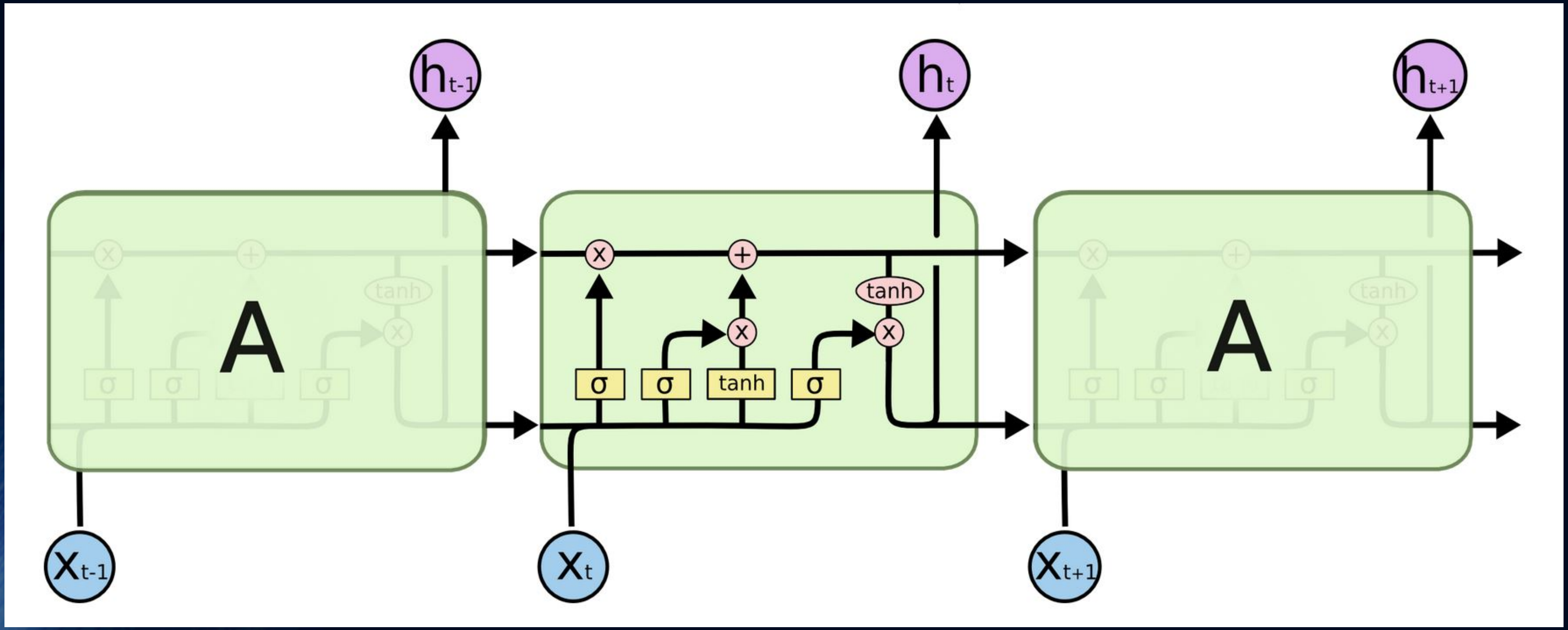
The repeating module in a RNN contains a tanh only

RNN & LSTMs (Long Short Term Memory) network



Instead the repeating module of a LSTM contains 4 interacting layers

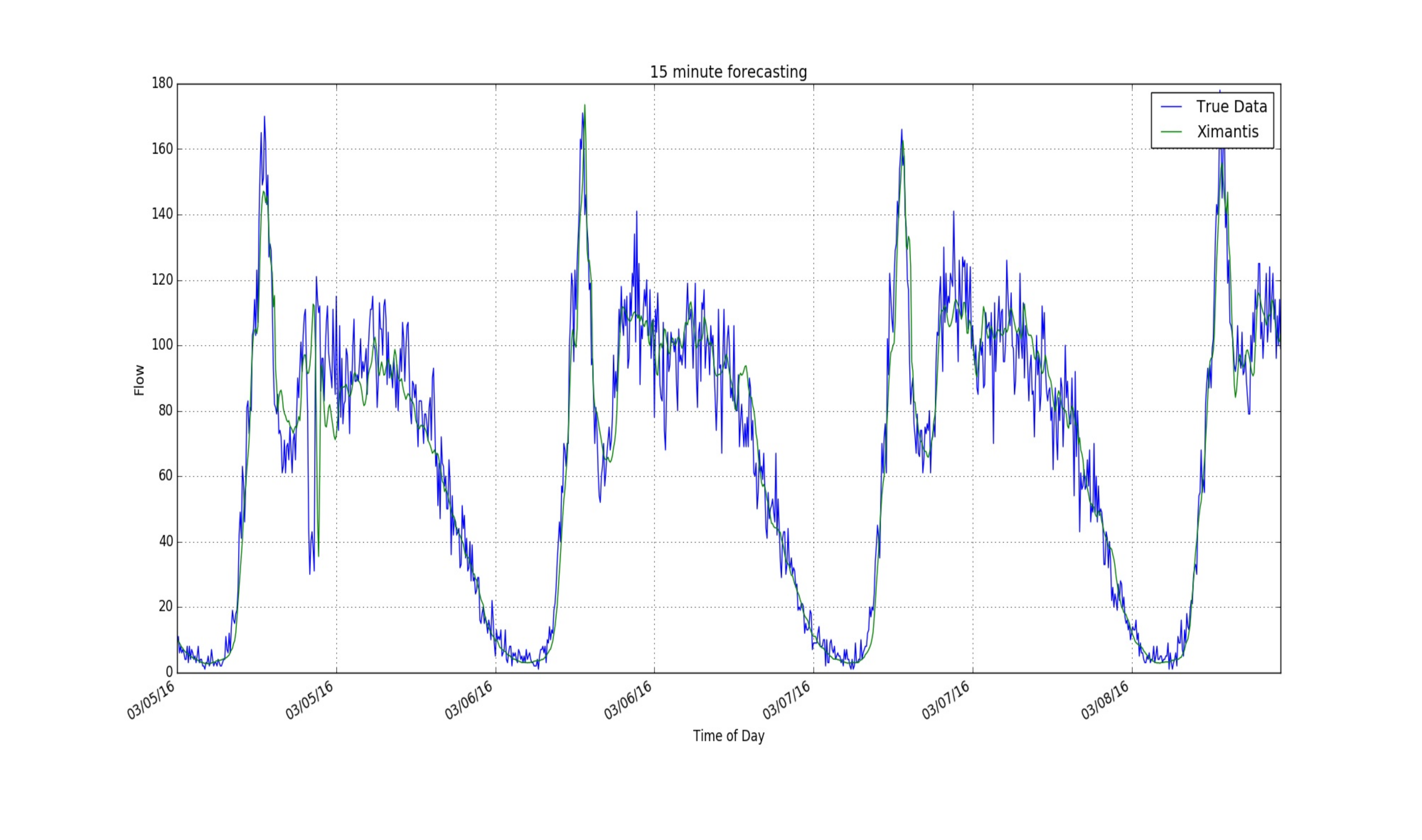
RNN & LSTMs
(Long Short Term Memory) network



Instead the repeating module of a LSTM contains 4 interacting layers

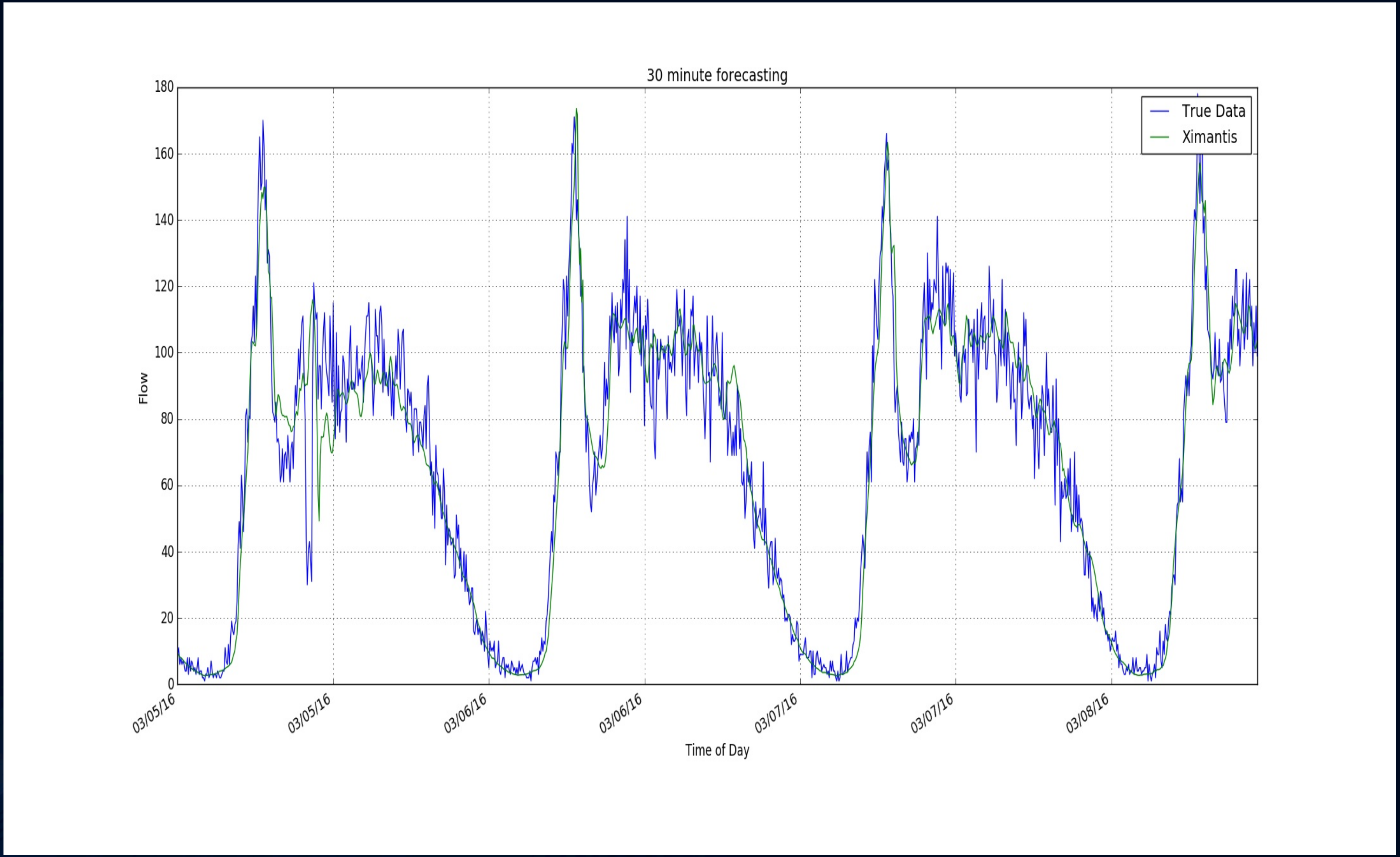
Forecast

- **15 min ahead**



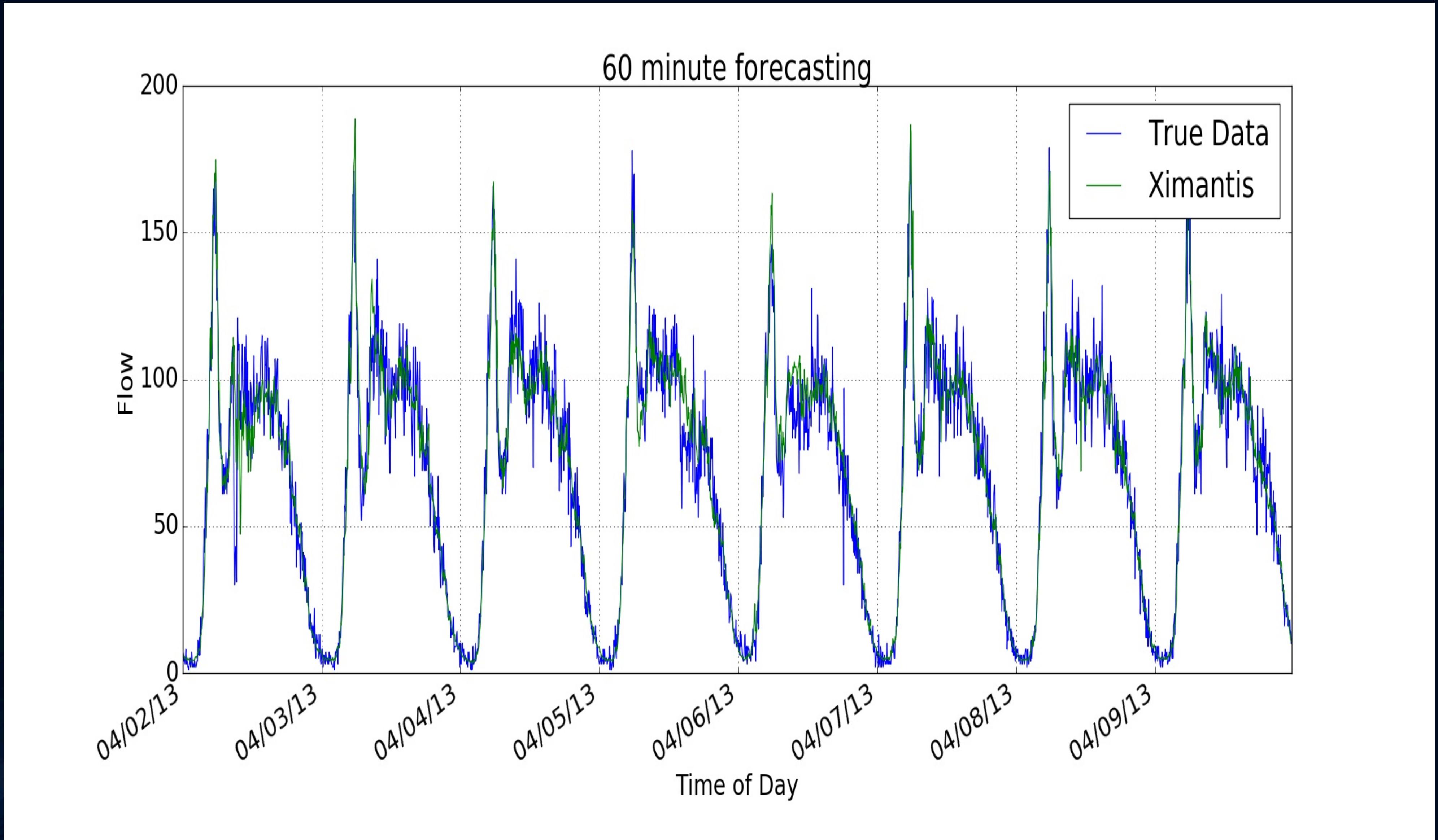
Forecast

- 15 min ahead
- **30 min ahead**



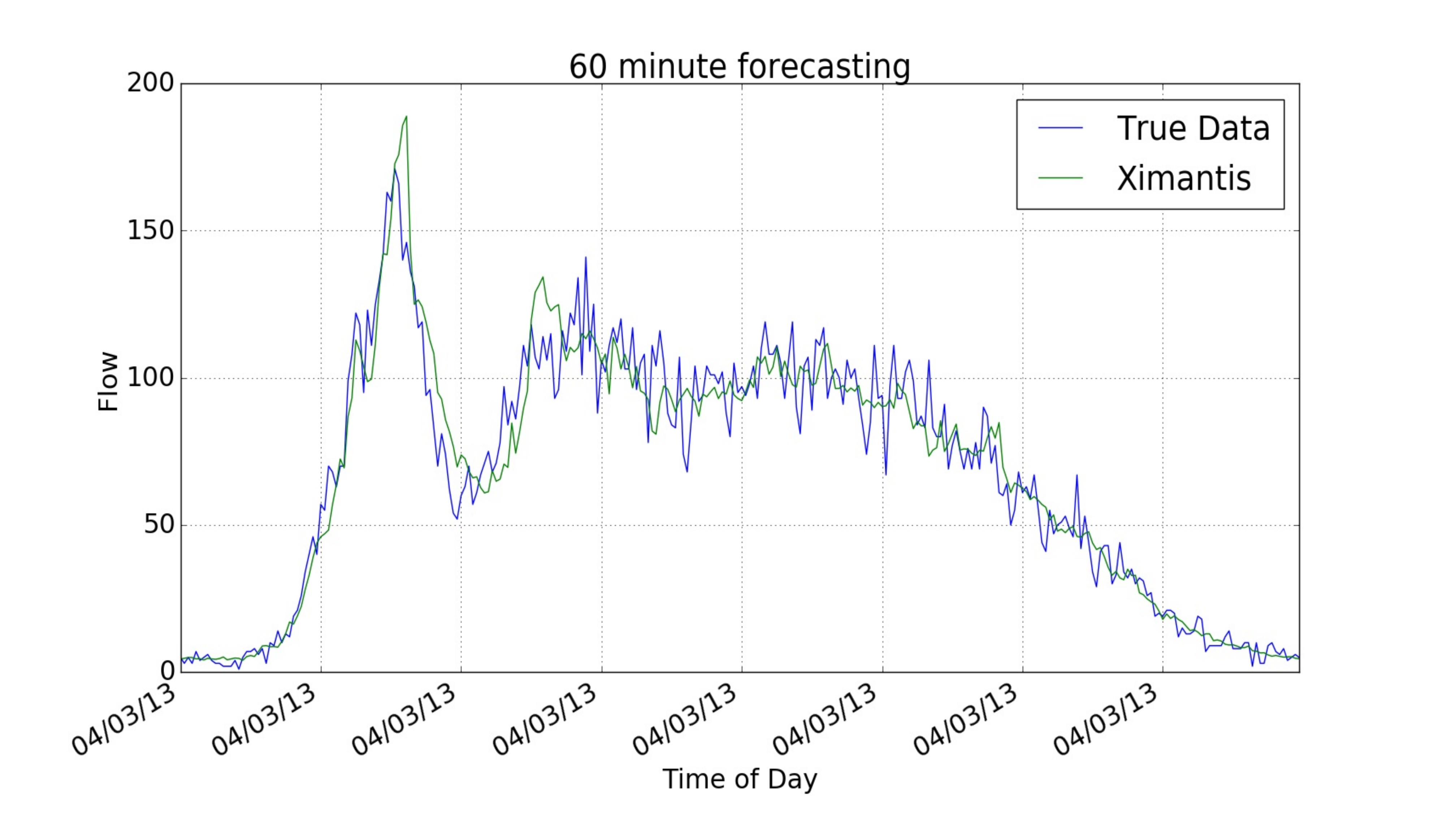
Forecast

- 15 min ahead
- 30 min ahead
- **60 min ahead**



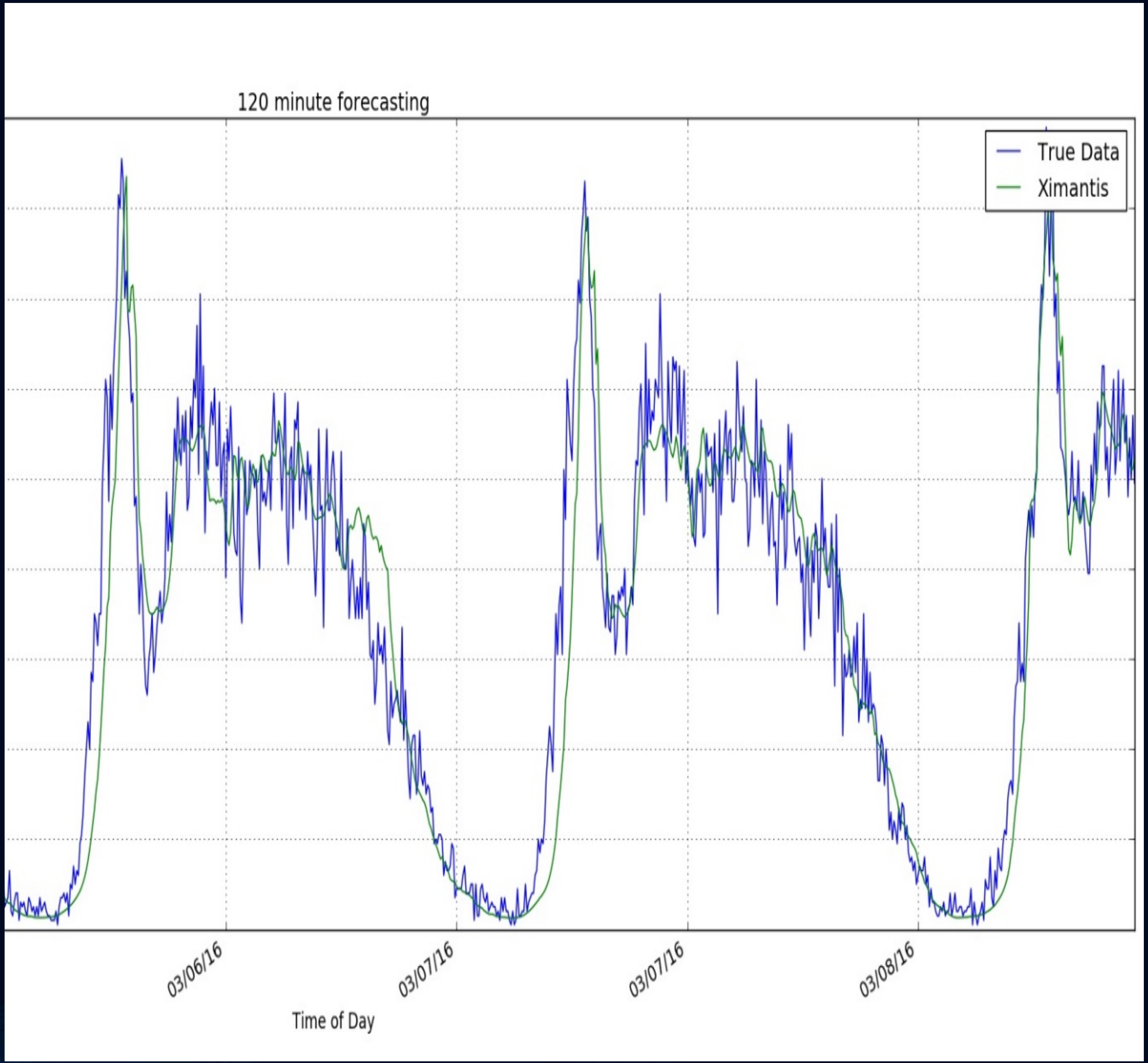
Forecast

- 15 min ahead
- 30 min ahead
- **60 min ahead**

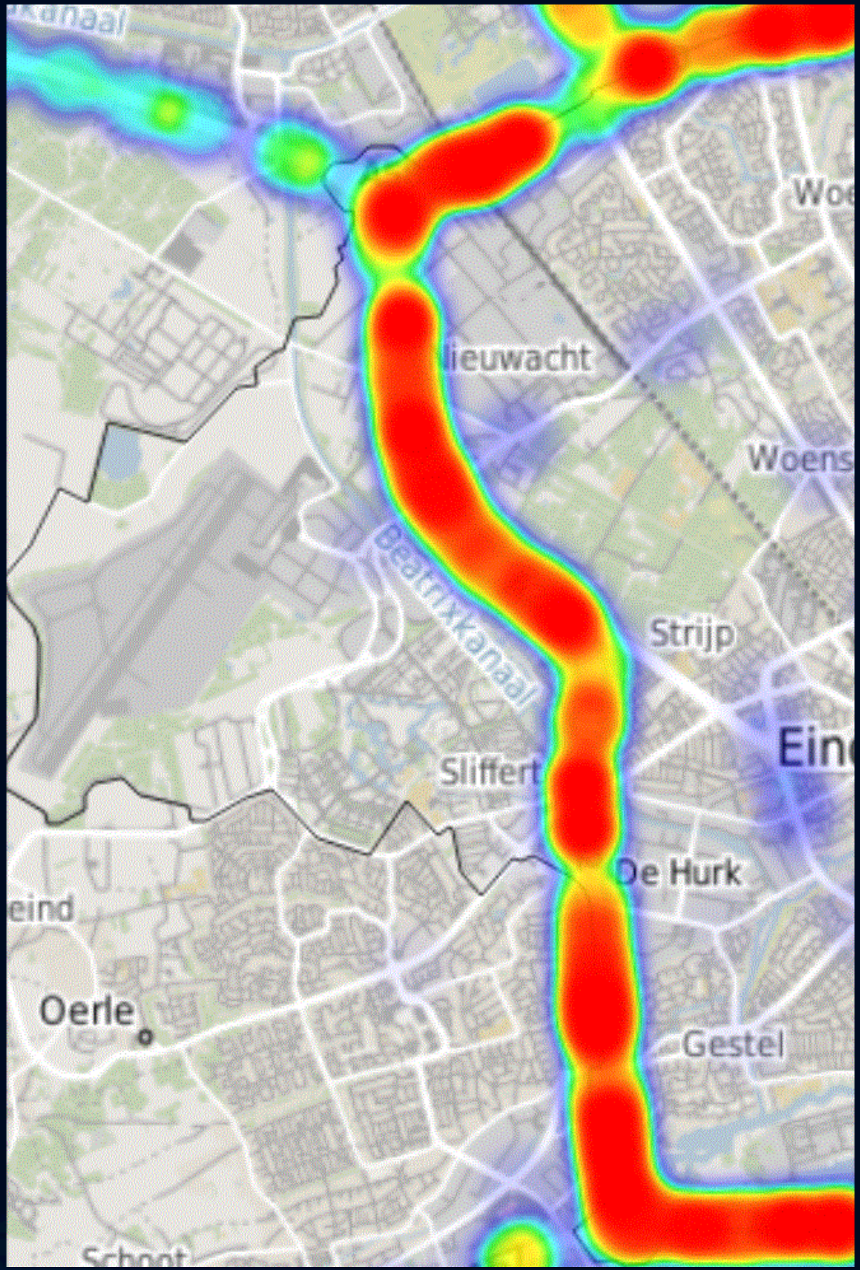


Forecast

- 15 min ahead
- 30 min ahead
- 60 min ahead
- **120 min …**



Traffic Heat Maps



Last word…

Last word… Attention!

- Look into Attention networks!

Could be better suited for your application than LSTMs, GRUs, SAEs, etc…

Thank you!

Credits

A.Schadschneider. Traffic flow: a statistical physics point of view. Physica A, 312:153, 2002

A.Sopasakis. Unstable flow theory and modeling. Math. Comput. Modelling, 35(5-6):623, 2002

A.Sopasakis. Formal asymptotic models of vehicular traffic. Model closures. SIAM J.Appl.Math.,63:1561,2003

A.Sopasakis. Stochastic noise approach to traffic flow modeling. Physica A, 342:741, Nov. 2004

Traffic video by Fernando Livschitz, www.bsvideos.com