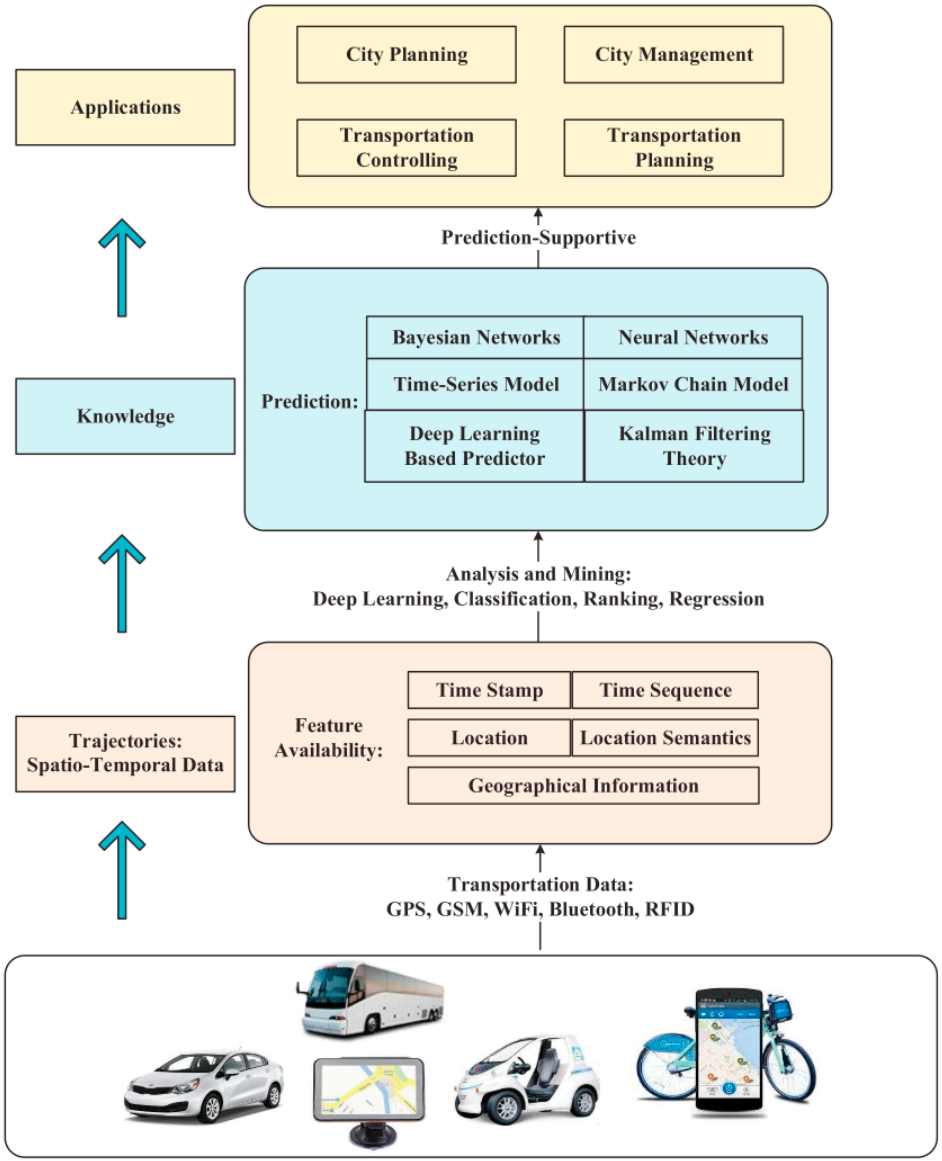
Deep Learning: Introducing the toolbox

Overview

The Deep Learning Toolbox

- Data
- Architecture
- Activation functions
- Loss function
- Regularization
- Software frameworks

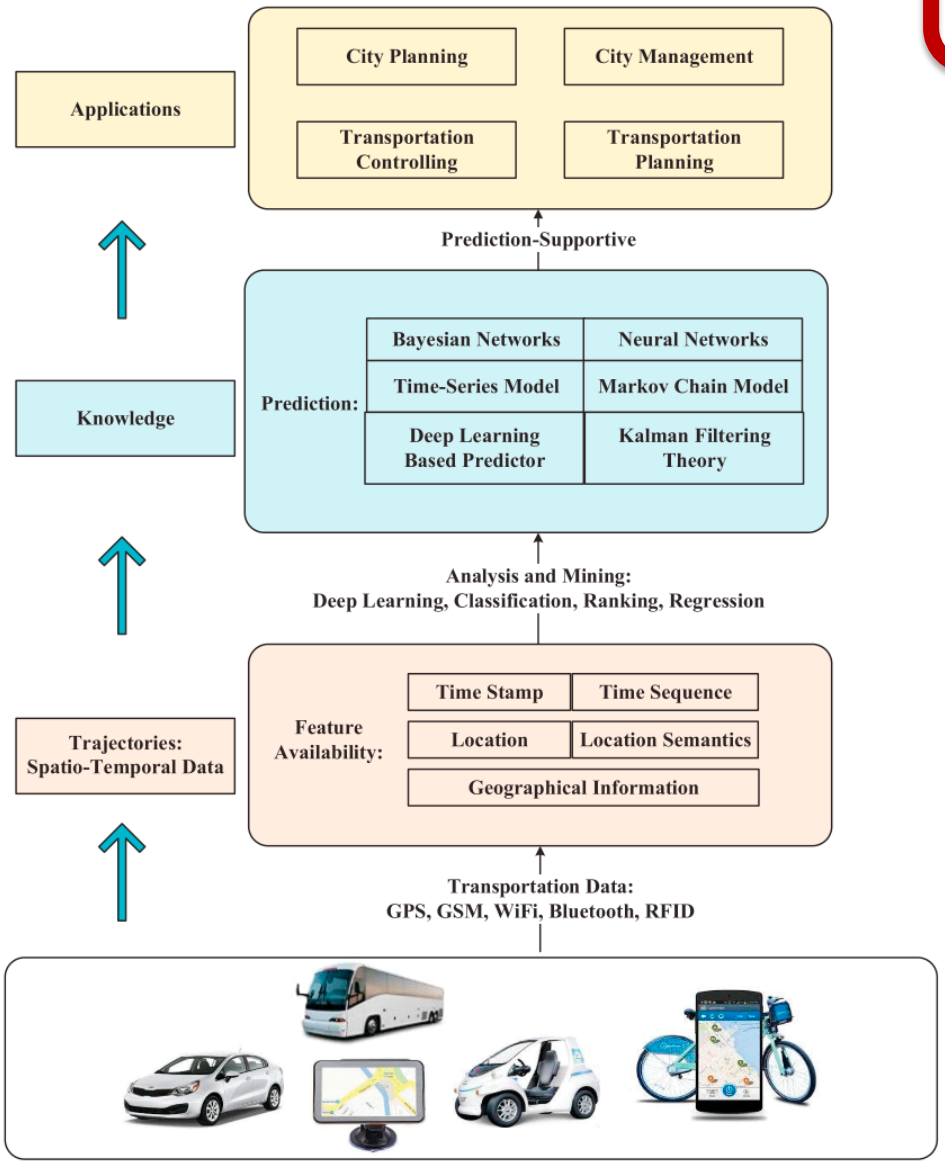
Data



Source	Tools	Data
Smart Card	Smart Card	OD Flows, Travel Time
GPS	GPS	Vehicle Position, Vehicle Density, Vehicle Speed
Video	Video Camera	Vehicle Position, Vehicle Speed, Vehicle Density, Vehicle Classification
Road Site Sensor	Induction Loops, Road Tubes, Microwave Radar, LIDAR/Infared Acoustic, Toll Plazas	Vehicle Position, Vehicle Speed, Vehicle Density, Vehicle Classification
Floating Car Sensor	License Plate Recognition, Transponders	Travel Time, OD Flows
Wide Area Sensor	GPS, Cell phone Tracking, Airborne Sensors	Travel Time, OD Flows
Connected and Autonomous Vehicles (CAVs)	Diverse Sensors	Coordinate, speed, acceleration, safety data,
Passive Collection	Social Media, Mobile Phone Data	Travel Time, OD Flows
Other Sources	Smart Grid, Smart Meters, Cellular Service, Dedicated Tests	Electric and Energy Consumption, Location, Channel Data

Data

Most common!



Source	Tools	Data
Smart Card	Smart Card	OD Flows, Travel Time
GPS	GPS	Vehicle Position, Vehicle Density, Vehicle Speed
Video	Video Camera	Vehicle Position, Vehicle Speed, Vehicle Density, Vehicle Classification
Road Site Sensor	Induction Loops, Road Tubes, Microwave Radar, LIDAR/Infared Acoustic, Toll Plazas	Vehicle Position, Vehicle Speed, Vehicle Density, Vehicle Classification
Floating Car Sensor	License Plate Recognition, Transponders	Travel Time, OD Flows
Wide Area Sensor	GPS, Cell phone Tracking, Airborne Sensors	Travel Time, OD Flows
Connected and Autonomous Vehicles (CAVs)	Diverse Sensors	Coordinate, speed, acceleration, safety data,
Passive Collection	Social Media, Mobile Phone Data	Travel Time, OD Flows
Other Sources	Smart Grid, Smart Meters, Cellular Service, Dedicated Tests	Electric and Energy Consumption, Location, Channel Data

Data

Smart Card Data: a sample of two weeks' data of a major subway system

New York City TLC Trip Record [Link]

NHTS 2017

Blue Bikes Comprehensive Trip Histories [Link]

Open transit data toolkit [Link]

- MTA Bus Time® Historical Data [Link]
  - Bus speeds
  - headways
- Subway Turnstile Data
  - Turnstile data tracks the number of entries and exits of passengers in the subway system

Didi Opendata [Link]

New York City MTA GTFS data [Link]

NYC open data including many transportation datasets [Link]

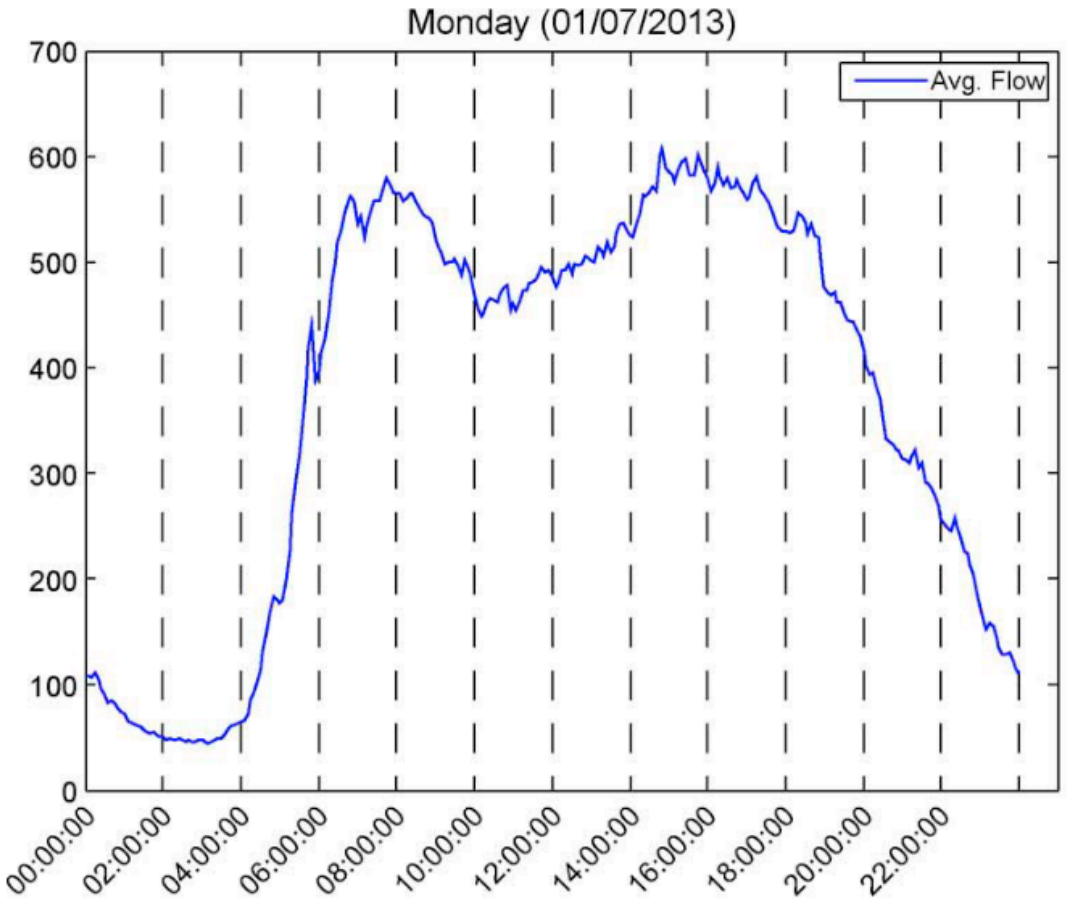
Flow provides a way of creating a simulation based on OpenStreetMap data [Link]

Google Street View [Link]

IOT data [link]

A simple problem

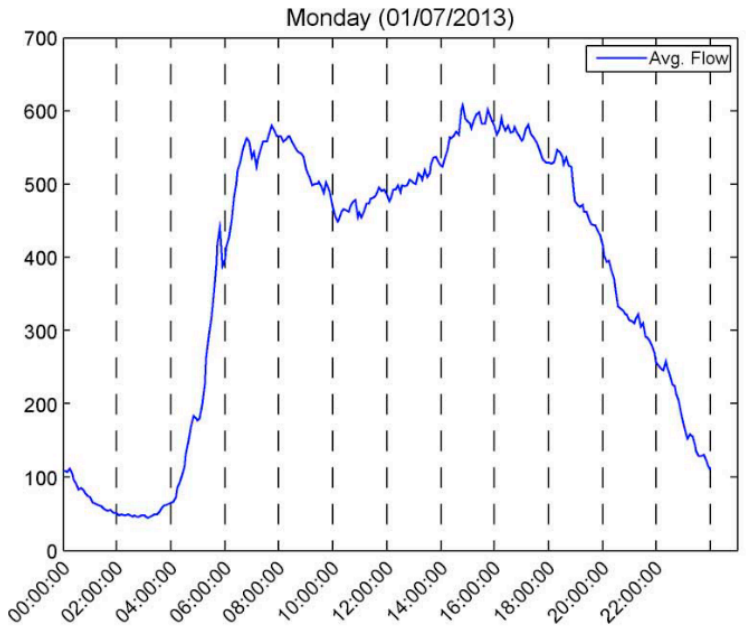
- Predict traffic flow for the next 15 minutes



Traditional approaches

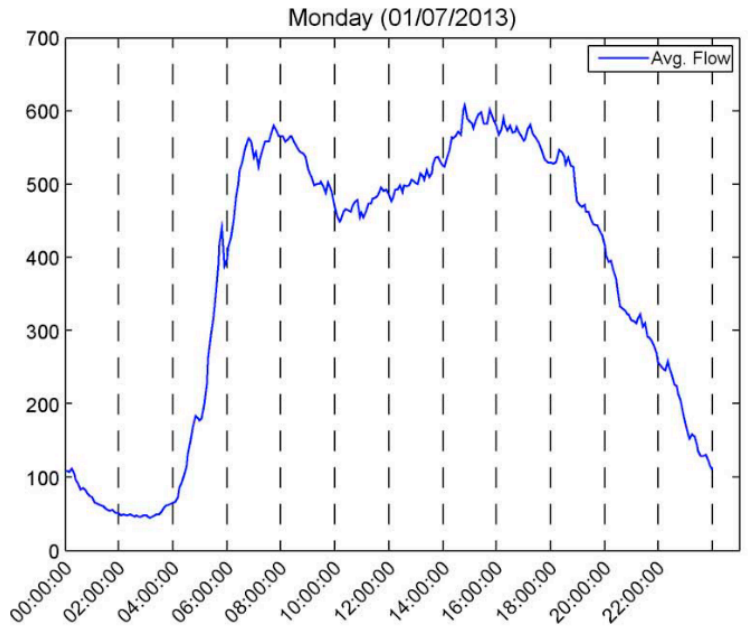
- ARIMA
  - ARIMAX
  - SARIMA

$$y_t = \sum_{j=1}^{r} \phi_j y_{t-j} + \sum_{j=1}^{r-1} \psi_j \epsilon_{t-j} + \epsilon_t$$



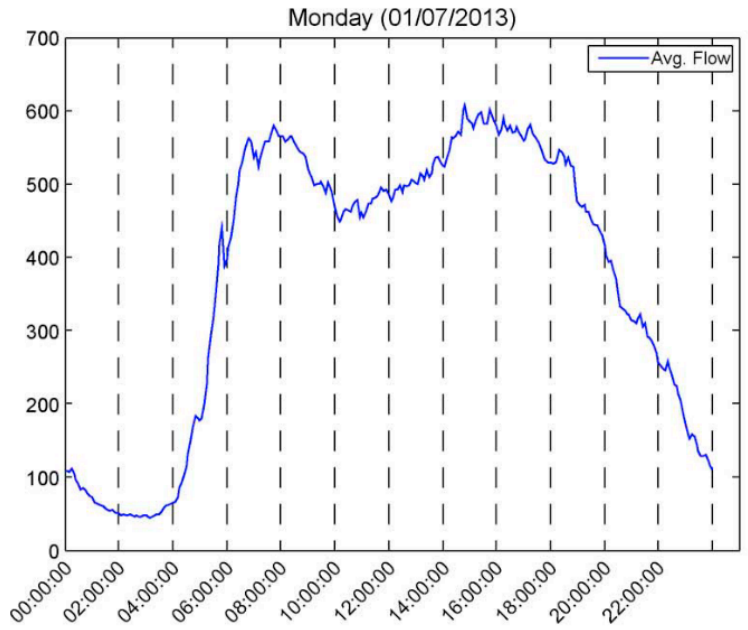
Traditional approaches

- ARIMA
  - ARIMAX
  - SARIMA
- KNN



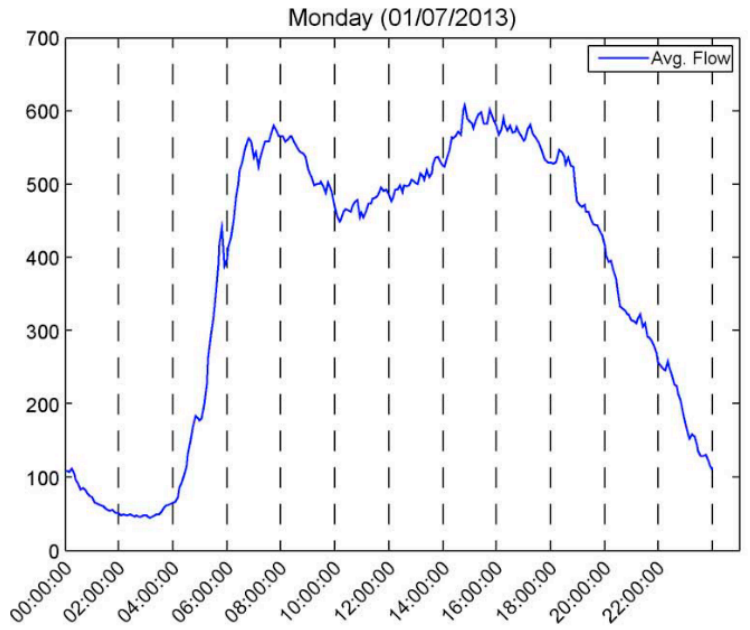
Traditional approaches

- ARIMA
  - ARIMAX
  - SARIMA
- KNN
- ANN (1 hidden layer)



Traditional approaches

- ARIMA
  - ARIMAX
  - SARIMA
- KNN
- ANN (1 hidden layer)
- SVM



Our first DL model

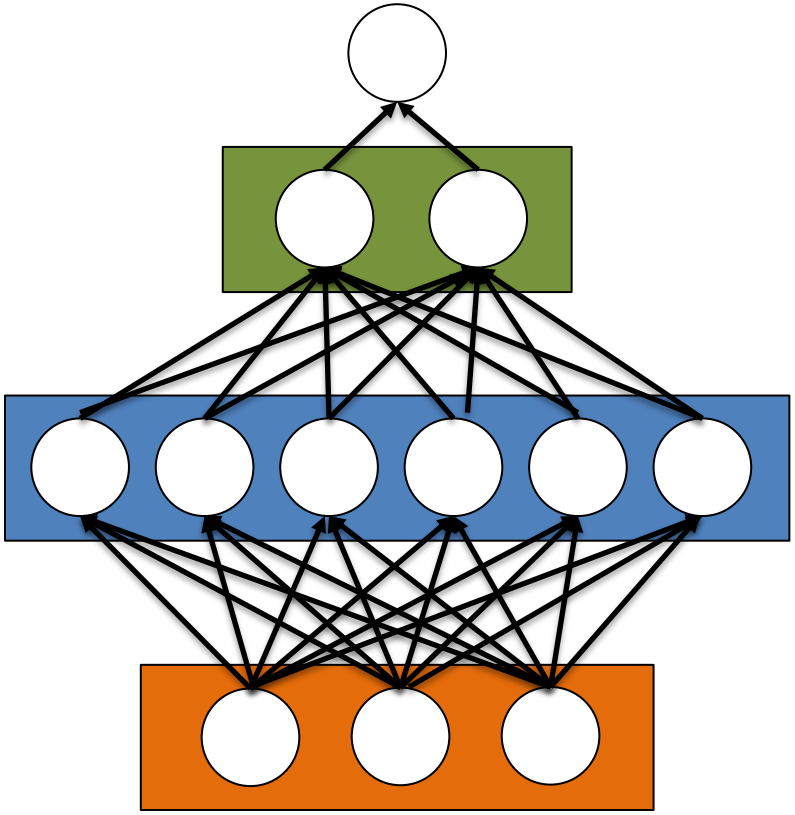
- What is “deep”, exactly?

Our first DL model

- What is “deep”, exactly?
    - More than 1 hidden layer

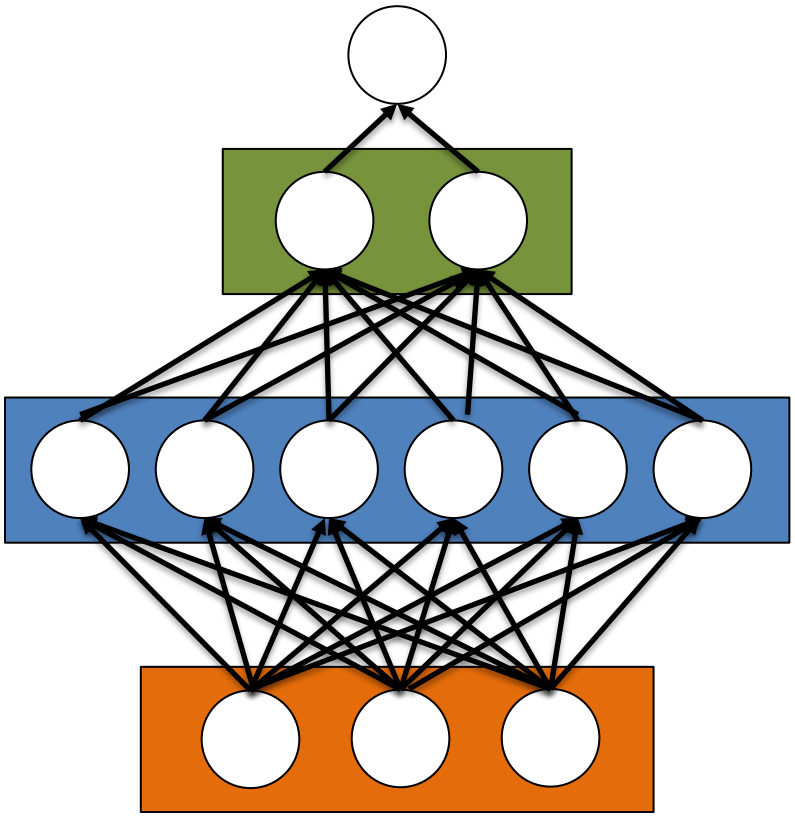
Our first DL model

- What is “deep”, exactly?
    - More than 1 hidden layer



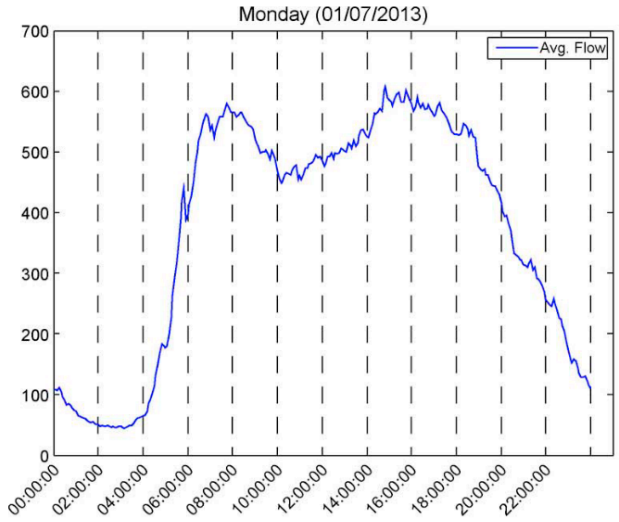
Our first DL model

- What is “deep”, exactly?
  - More than 1 hidden layer



Our first DL model

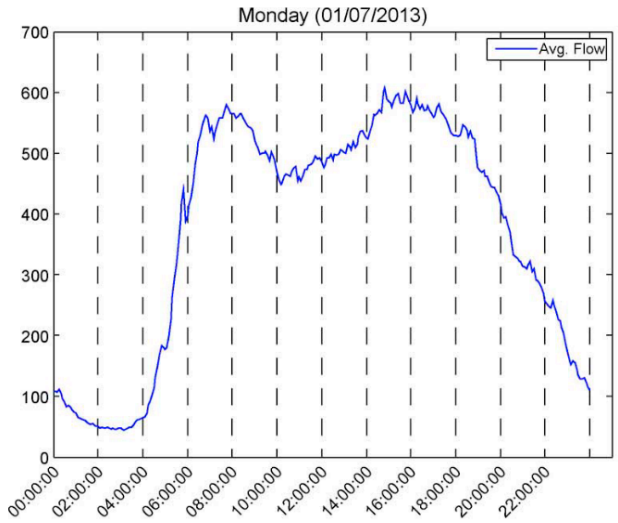
- How should we represent our input data?
  - Data collected every 30 s
  - Aggregated over 5 min intervals
  - $D = \{x_t, x_{t-1}, \ldots, x_1\}$



Our first DL model

- $D = \{x_t, x_{t-1}, \ldots, x_1\}$
- Let's turn it into a supervised learning problem
  - Choose a lookback horizon h

$$\{x_t\} \quad \{x_{t-1}, x_{t-2}, \ldots, x_{t-h}\}$$
$$\{x_{t-1}\} \quad \{x_{t-2}, x_{t-1}, \ldots, x_{t-h-1}\}$$



Our first DL model

- $D = \{x_t, x_{t-1}, \ldots, x_1\}$
- Let's turn it into a supervised learning problem
  - Choose a lookback horizon h

$$\{x_t\} \quad \{x_{t-1}, x_{t-2}, \ldots, x_{t-h}\}$$
$$\{x_{t-1}\} \quad \{x_{t-2}, x_{t-1}, \ldots, x_{t-h-1}\}$$

- Note: fixed input size

Our first DL model

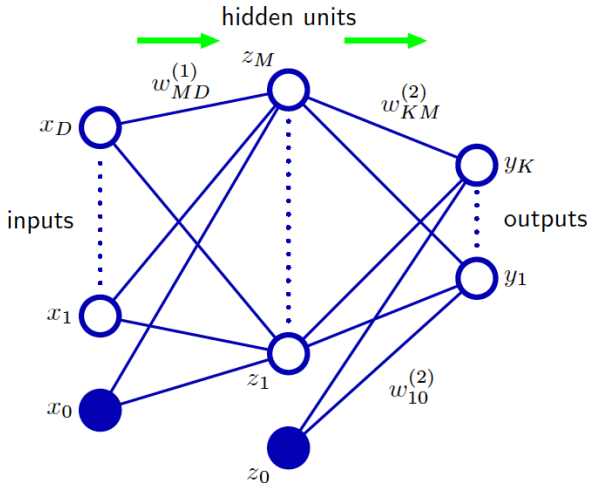
- ~~Data~~
- Architecture
- Activation functions
- Losses
- Regularization
- Software frameworks

Let's start with the simplest architecture

D-dimensional input

1 hidden layer comprising M hidden units

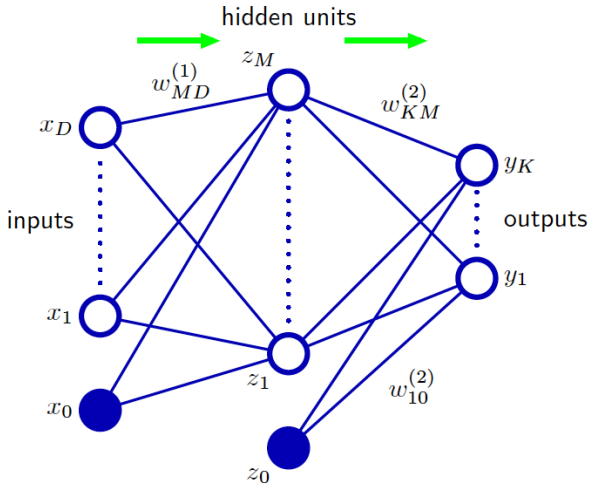
K-dimensional output



Let's start with the simplest architecture

First, we construct M linear combinations of the input variables

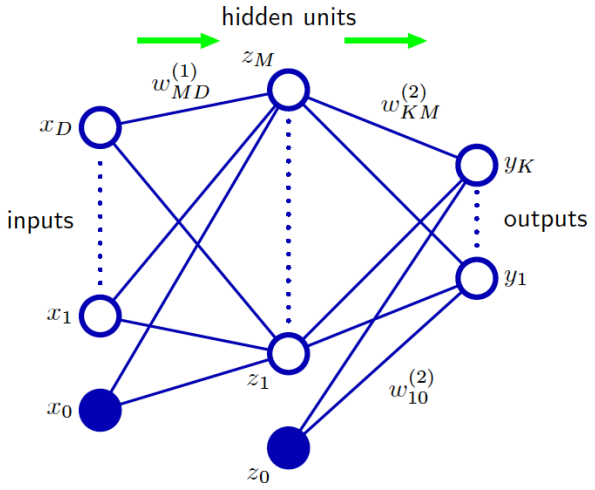
$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$



Let's start with the simplest architecture

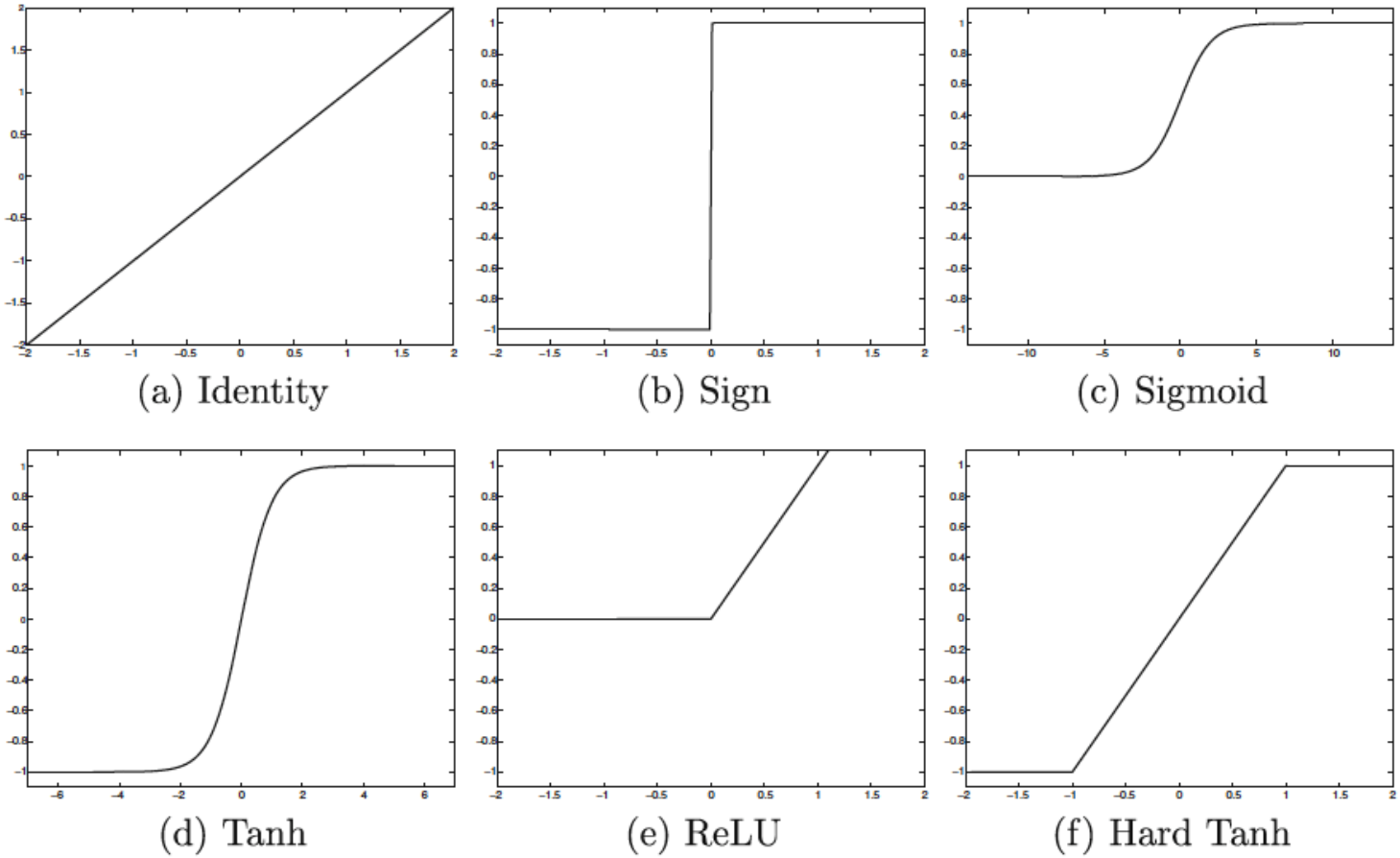
Then, we pass the output through a nonlinear activation function

$$z_j = h(a_j).$$



Let's start with the simplest architecture

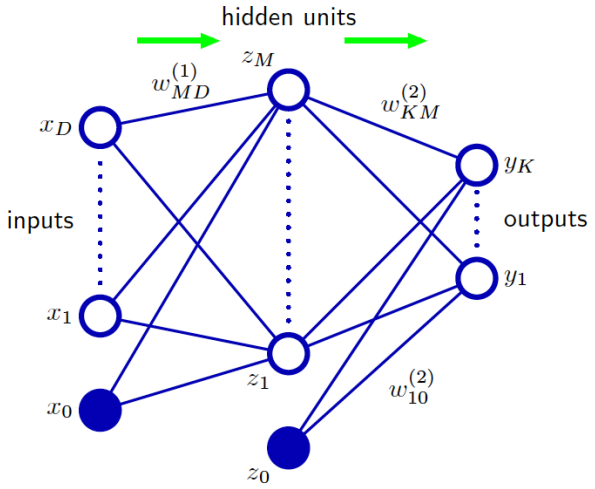
Examples of activation functions



Let's start with the simplest architecture

The outputs are then again linearly combined to give output unit activations

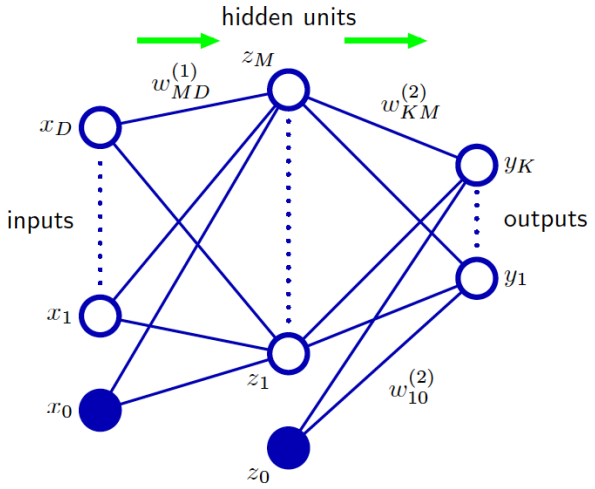
$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$



Let's start with the simplest architecture

Finally, the output is passed through another activation function

$$y_k = \sigma(a_k)$$



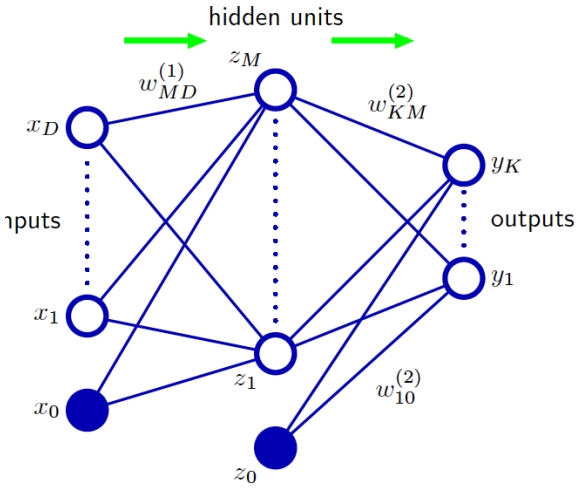
For regression, it's the identity function

For classification, it's SoftMax $\dfrac{\exp(a_k)}{\sum_j \exp(a_j)}$

Let's start with the simplest architecture

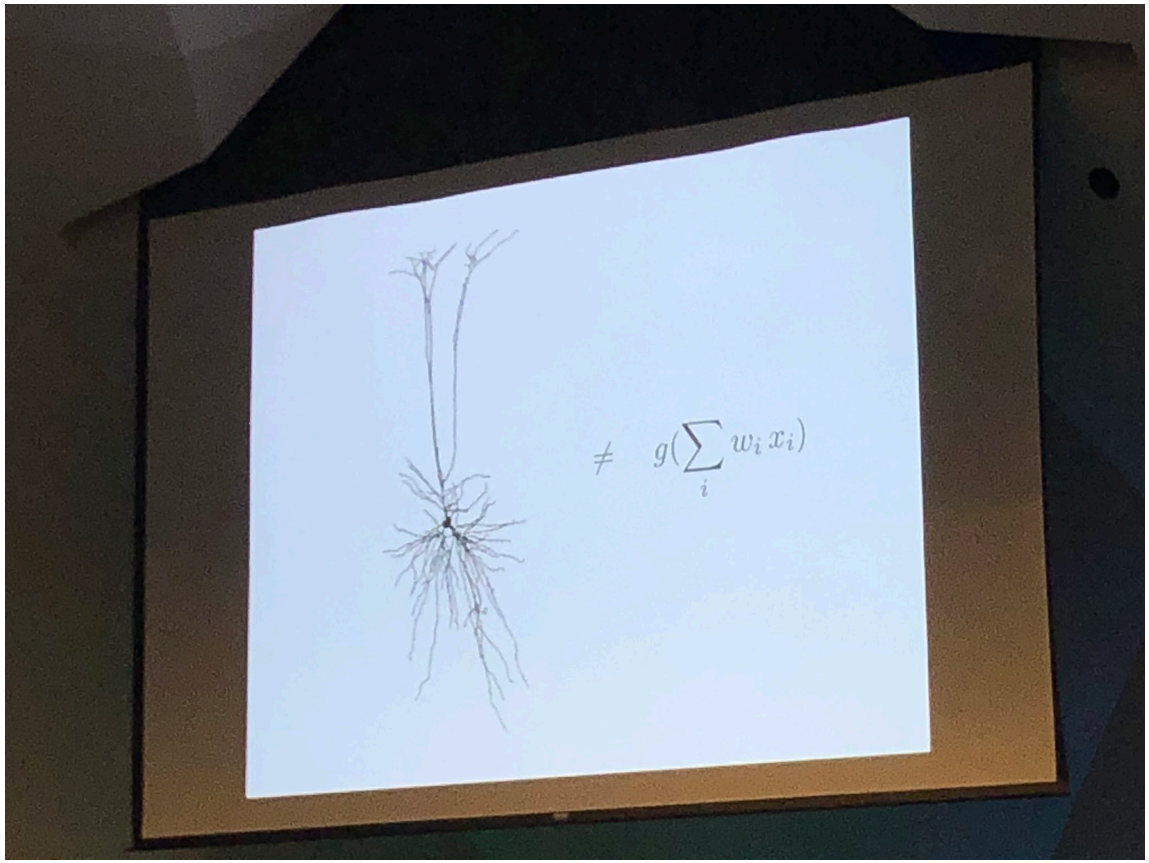
In summary

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$



Let's start with the simplest architecture

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$



Our first DL model

- ~~Data~~
- ~~Architecture~~
- ~~Activation functions~~
- Losses
- Regularization
- Software frameworks

Our first DL model

- Loss function
    - Quantifies how unhappy you will be if your predictions are wrong

Our first DL model

- Loss function
  - Quantifies how unhappy you will be if your predictions are wrong
  - For classification

$$L(f(x_i), y_i) = \mathbf{1}_{f(x_i) \neq y_i}$$

  - For regression

$$L(f(x_i), y_i) = (f(x_i) - y_i)^2$$

Module: tf.losses

Classes

`class Reduction` : Types of loss reduction.

Functions

`absolute_difference(...)` : Adds an Absolute Difference loss to the training procedure.

`add_loss(...)` : Adds a externally defined loss to the collection of losses.

`compute_weighted_loss(...)` : Computes the weighted loss.

`cosine_distance(...)` : Adds a cosine-distance loss to the training procedure. (deprecated arguments)

`get_losses(...)` : Gets the list of losses from the loss_collection.

`get_regularization_loss(...)` : Gets the total regularization loss.

`get_regularization_losses(...)` : Gets the list of regularization losses.

`get_total_loss(...)` : Returns a tensor whose value represents the total loss.

`hinge_loss(...)` : Adds a hinge loss to the training procedure.

`huber_loss(...)` : Adds a Huber Loss term to the training procedure.

`log_loss(...)` : Adds a Log Loss term to the training procedure.

`mean_pairwise_squared_error(...)` : Adds a pairwise-errors-squared loss to the training procedure.

`mean_squared_error(...)` : Adds a Sum-of-Squares loss to the training procedure.

`sigmoid_cross_entropy(...)` : Creates a cross-entropy loss using tf.nn.sigmoid_cross_entropy_with_logits.

`softmax_cross_entropy(...)` : Creates a cross-entropy loss using tf.nn.softmax_cross_entropy_with_logits_v2.

`sparse_softmax_cross_entropy(...)` : Cross-entropy loss using `tf.nn.sparse_softmax_cross_entropy_with_logits`.

Our first DL model

- Optimize!

$$\underline{\text{minimize}} \sum_{i=1}^{n} L(f(x_i), y_i)$$

Our first DL model

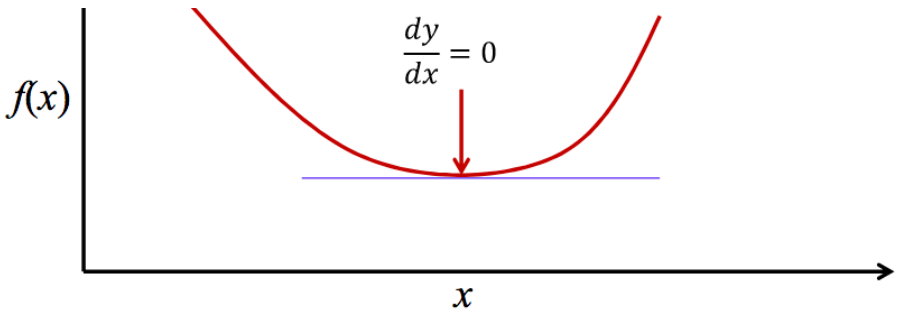
- Optimize!

$$\underline{\text{minimize}} \sum_{i=1}^{n} L(f(x_i), y_i)$$

- One-dimensional

$$\frac{df(x)}{dx} = 0$$

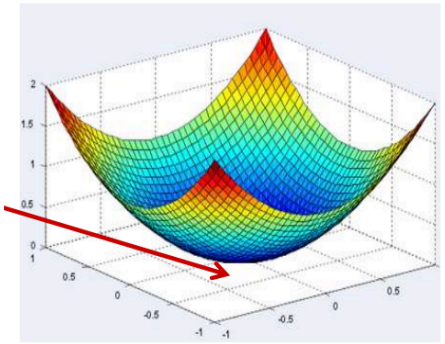
$$f''(x_{soln}) = \frac{df'(x_{soln})}{dx}$$

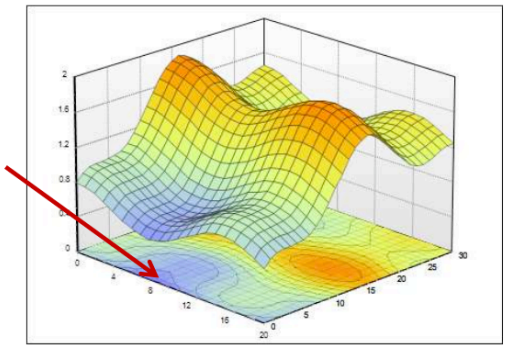


Our first DL model

- Multidimensional

$$\nabla f(X) = \left[\frac{\partial f(X)}{\partial x_1} \quad \frac{\partial f(X)}{\partial x_2} \quad \cdots \quad \frac{\partial f(X)}{\partial x_n}\right]$$



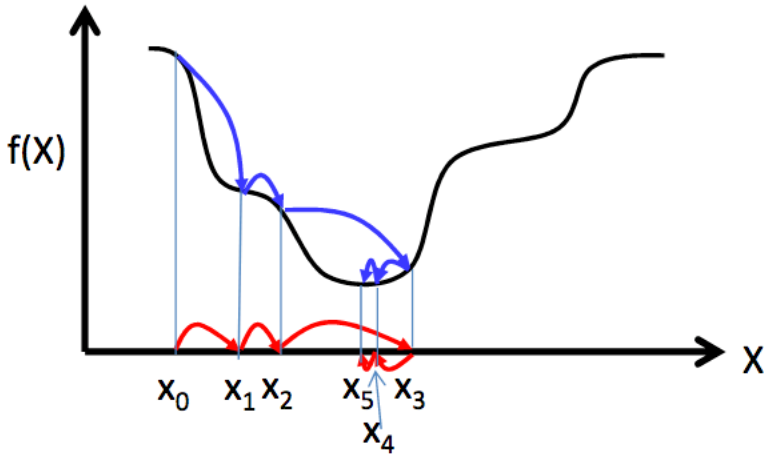


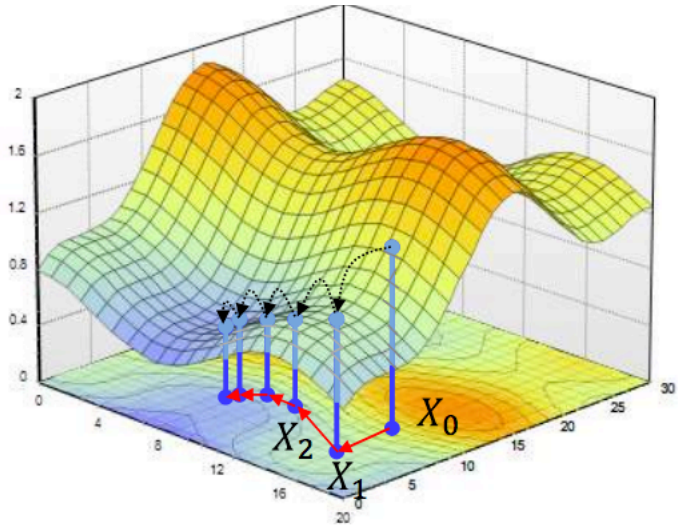
- Hessian matrix for the second derivatives

$$\nabla^2 f(x_1,\ldots,x_n) := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Our first DL model

- Often it is not possible to simply solve this optimization problem
  - The function may have an intractable form
- In these situations, iterative solutions are used
  - Begin with a guess for the optimal values, and then refine it iteratively until the correct value is obtained





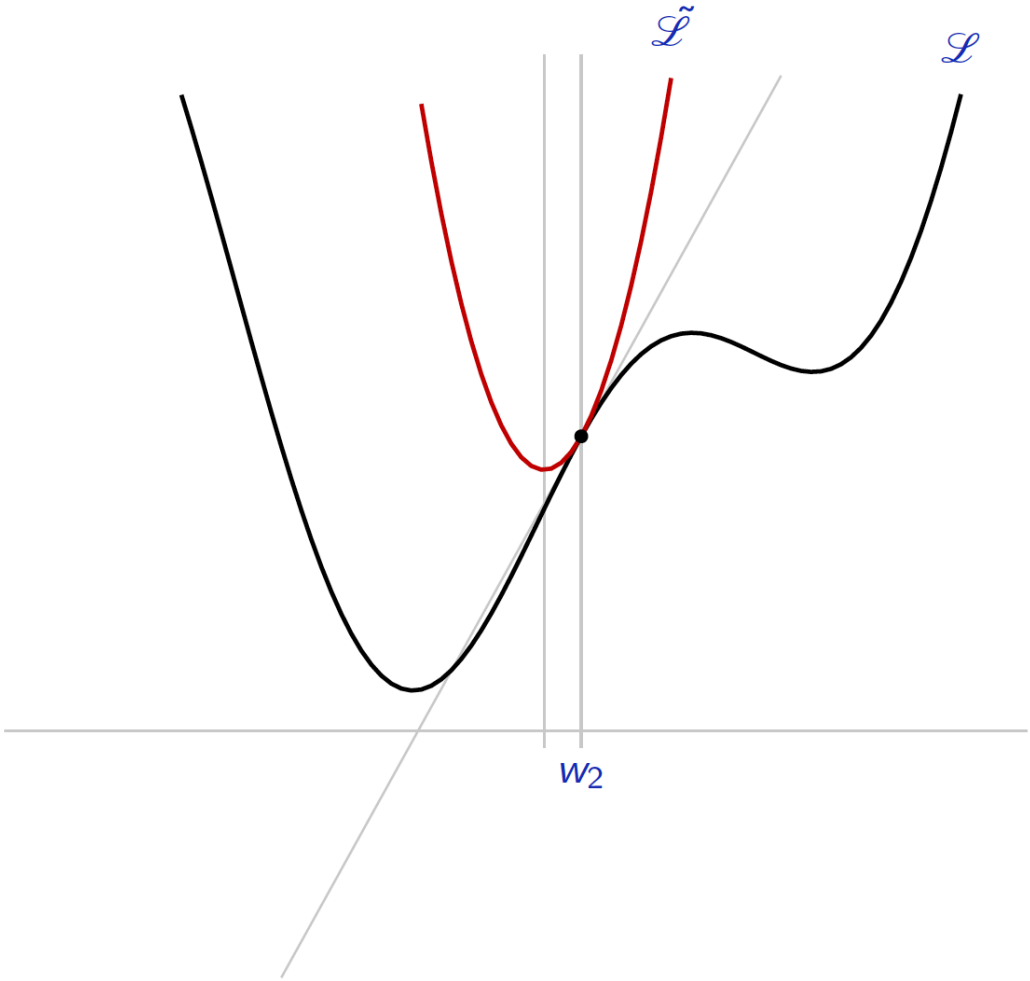
Our first DL model

- (Vanilla) Gradient descent

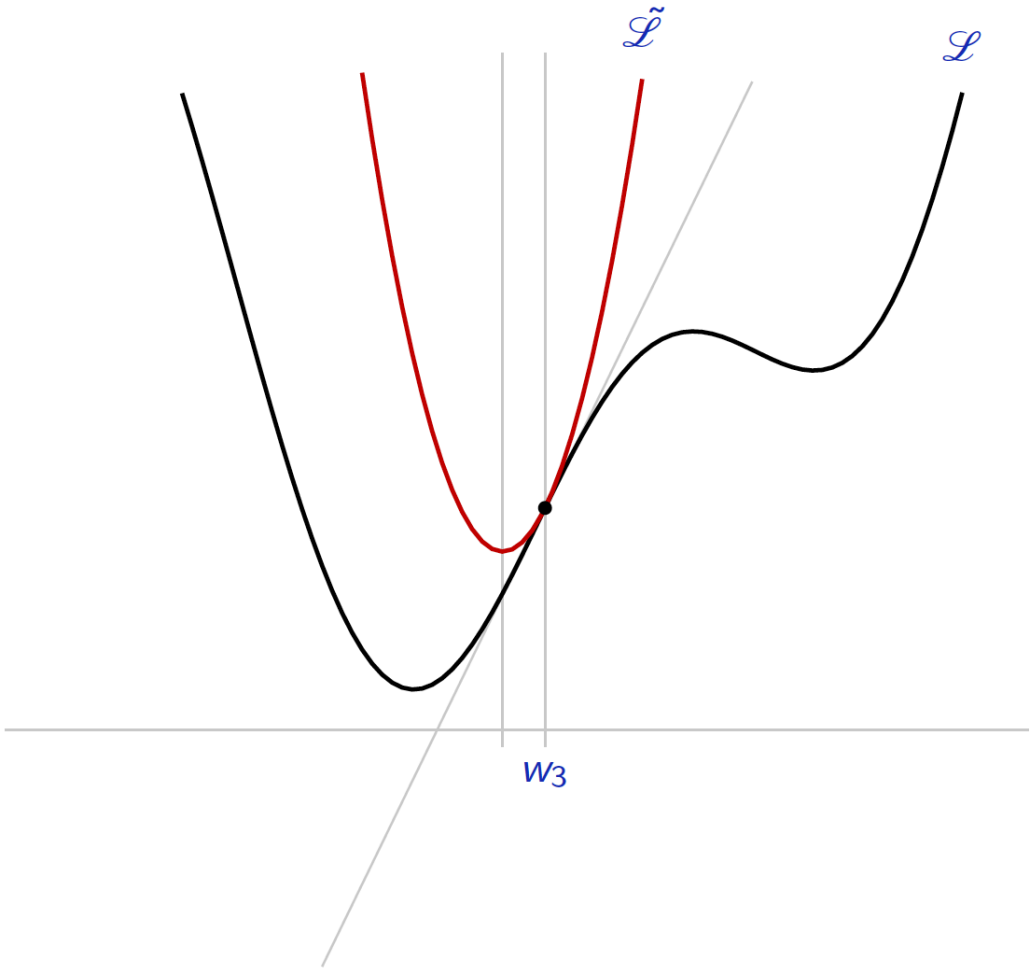
$$w^{k+1} = w^k - \eta^k \nabla f(w^k)^T$$

- Step-size (learning rate) is a hyperparameter

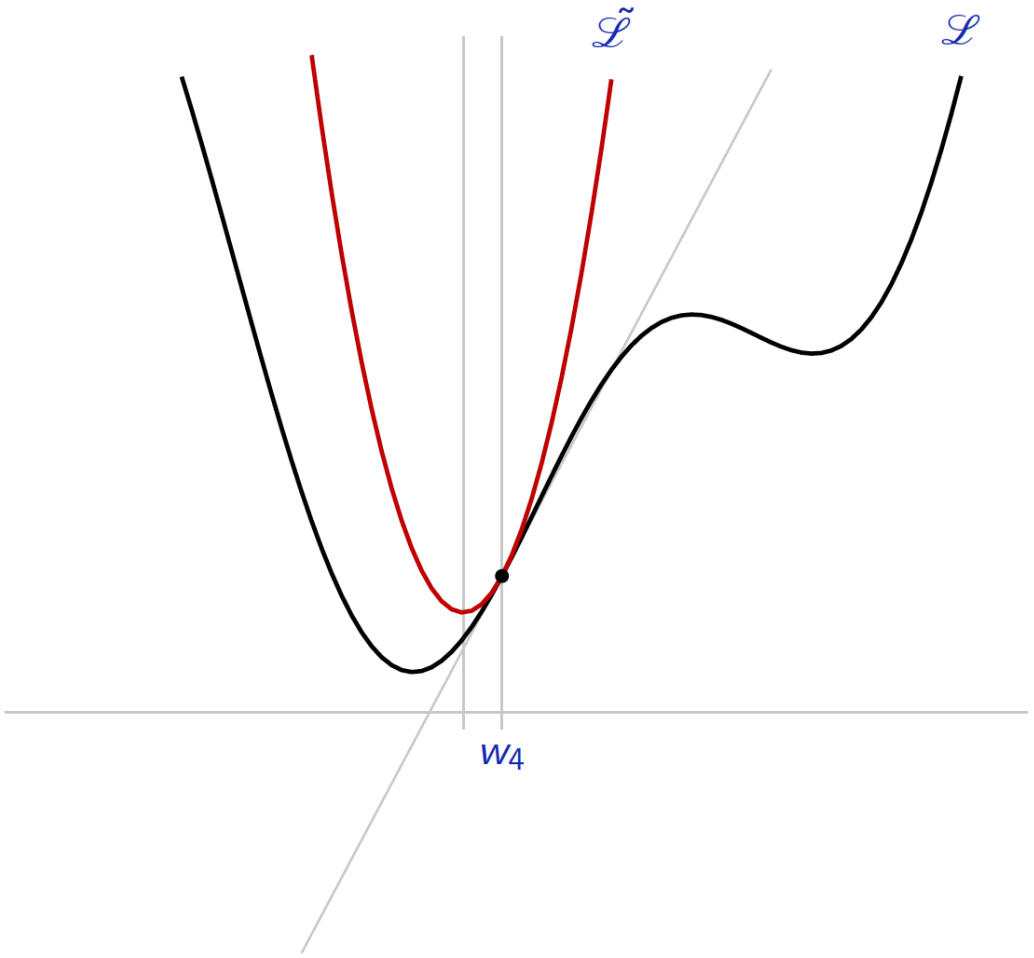
$\eta = 0.125$



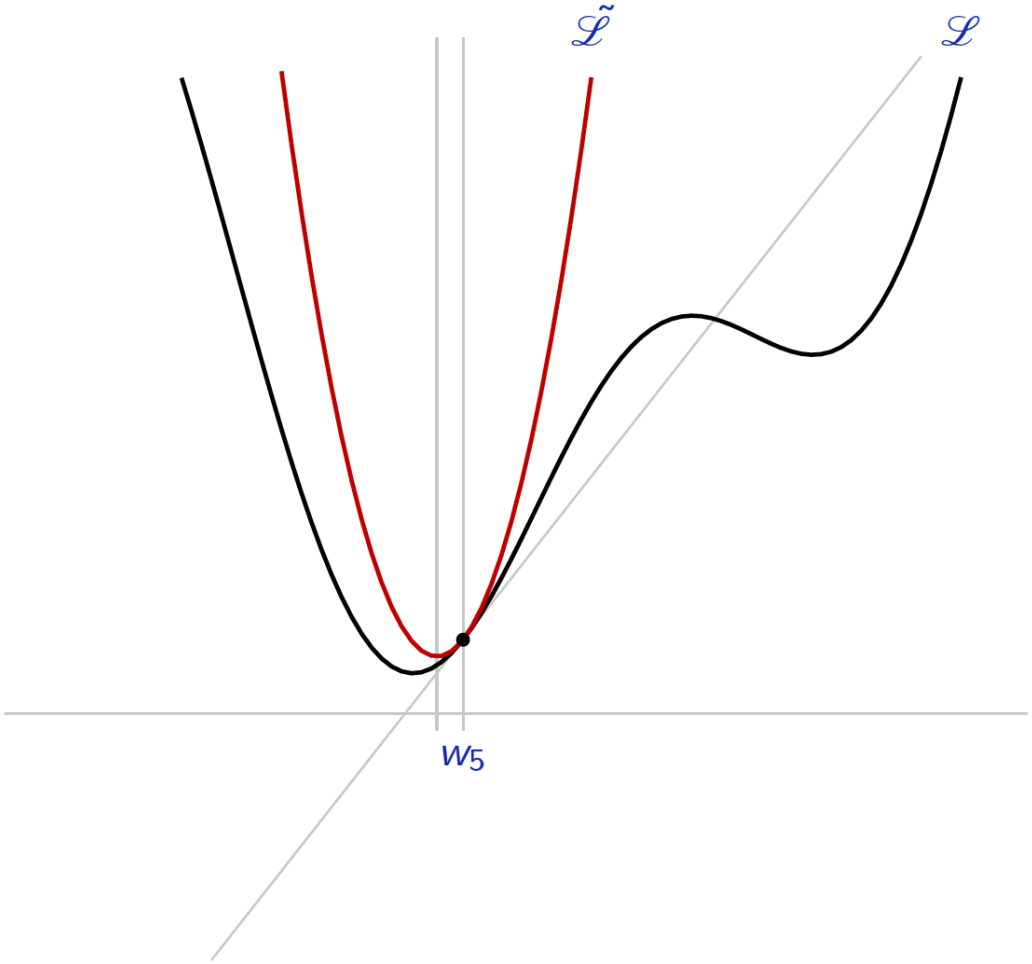
$\eta = 0.125$

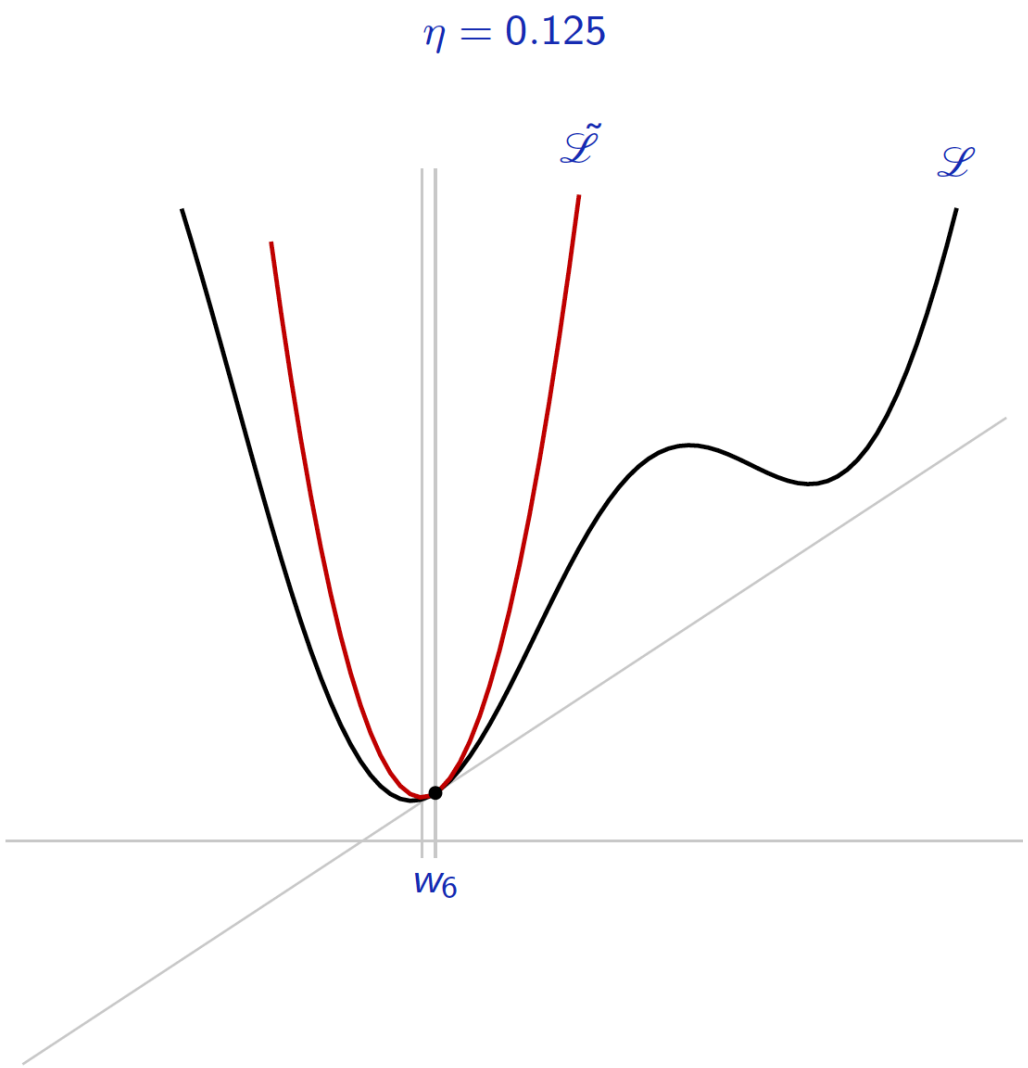


$\eta = 0.125$

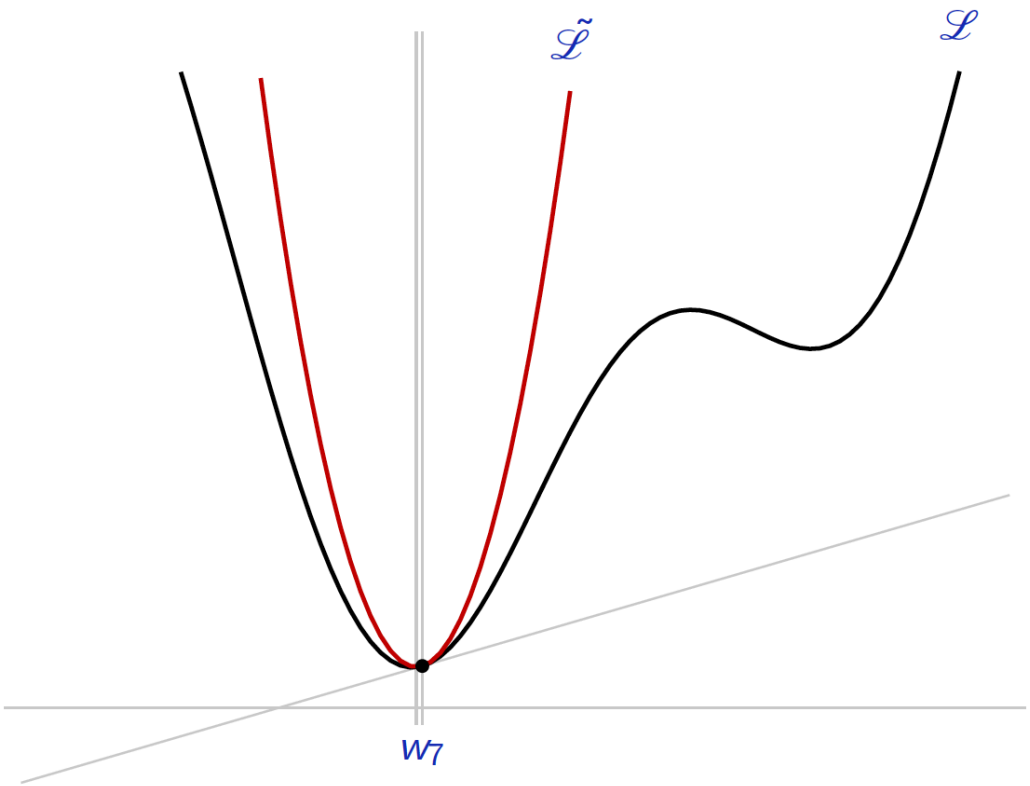


$\eta = 0.125$

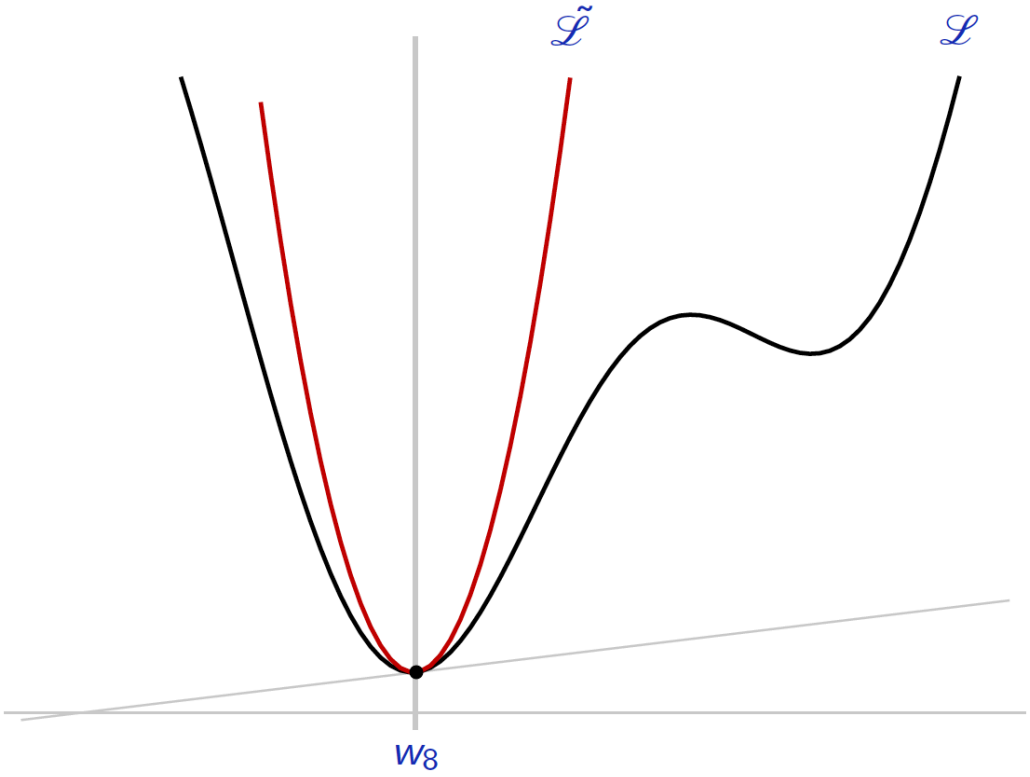


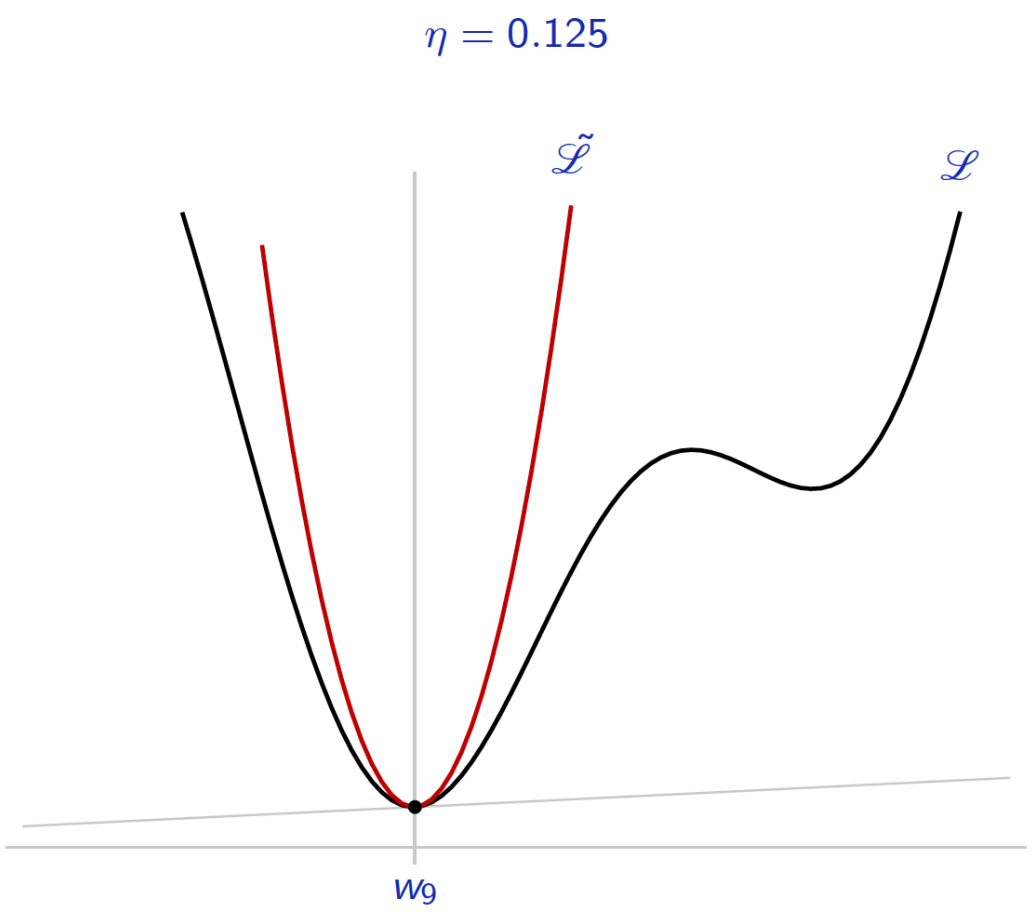

$\eta = 0.125$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_6

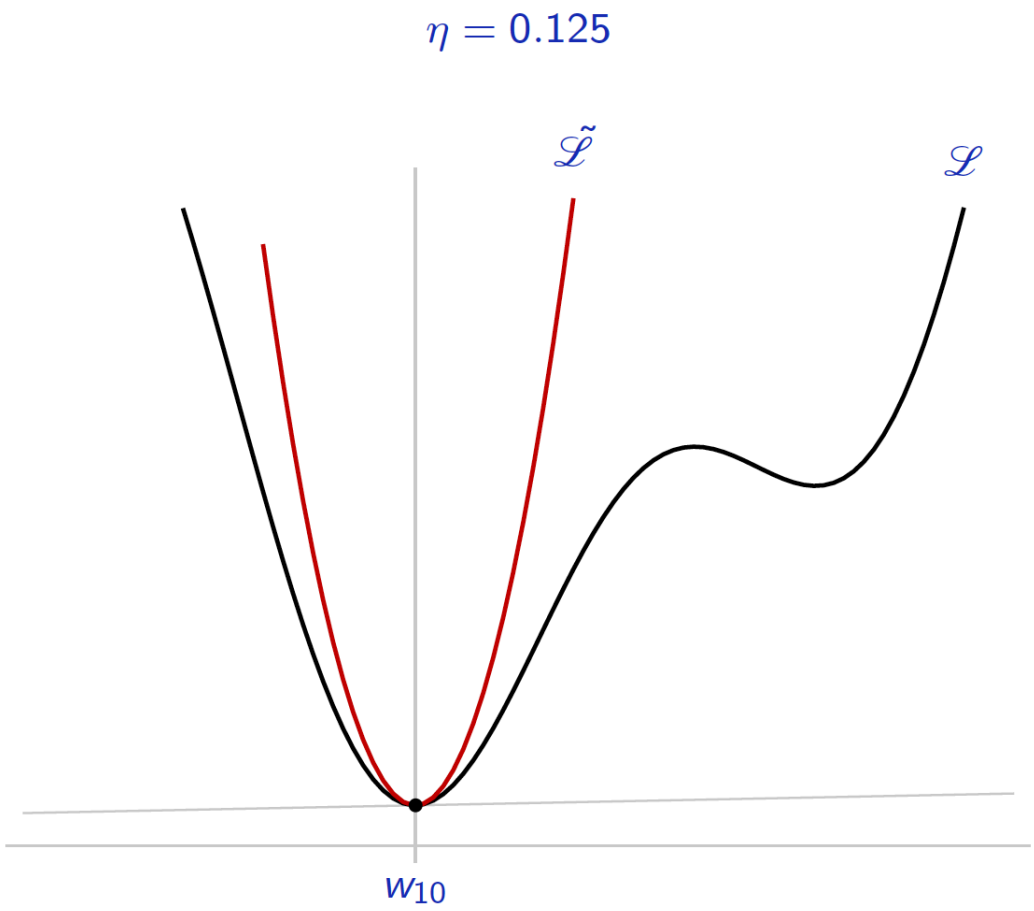
$\eta = 0.125$

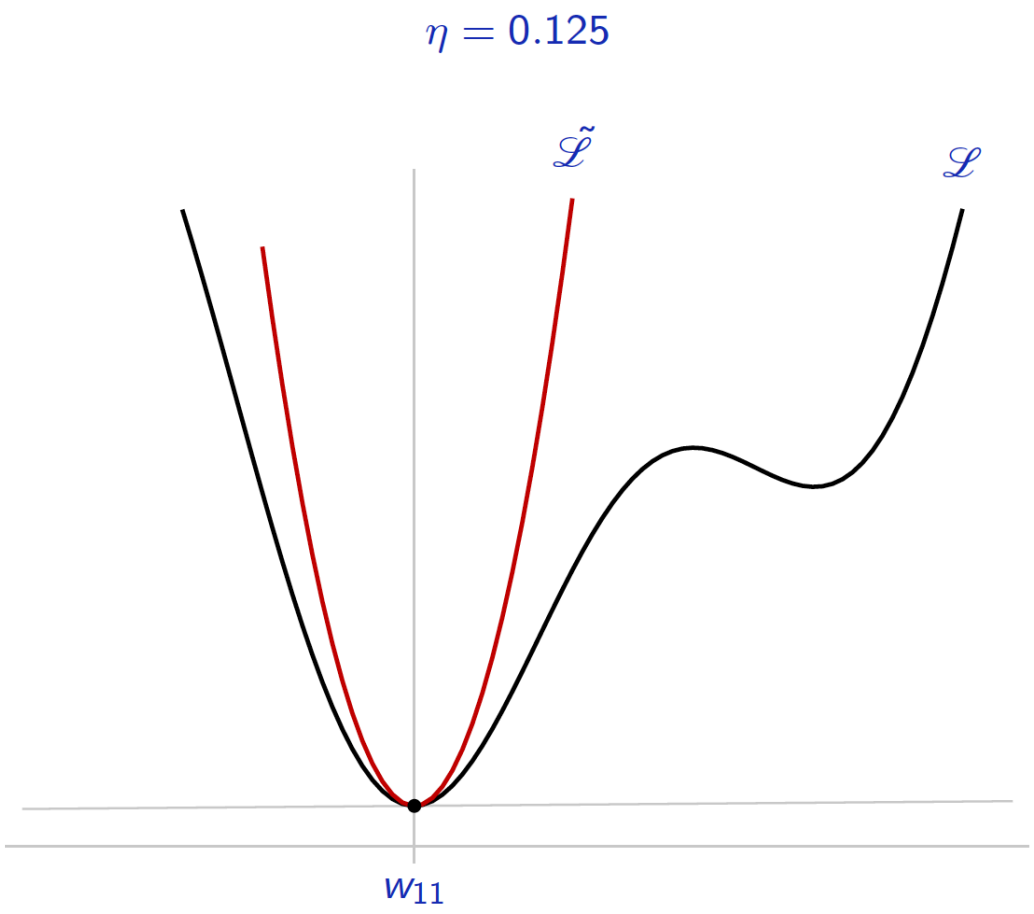


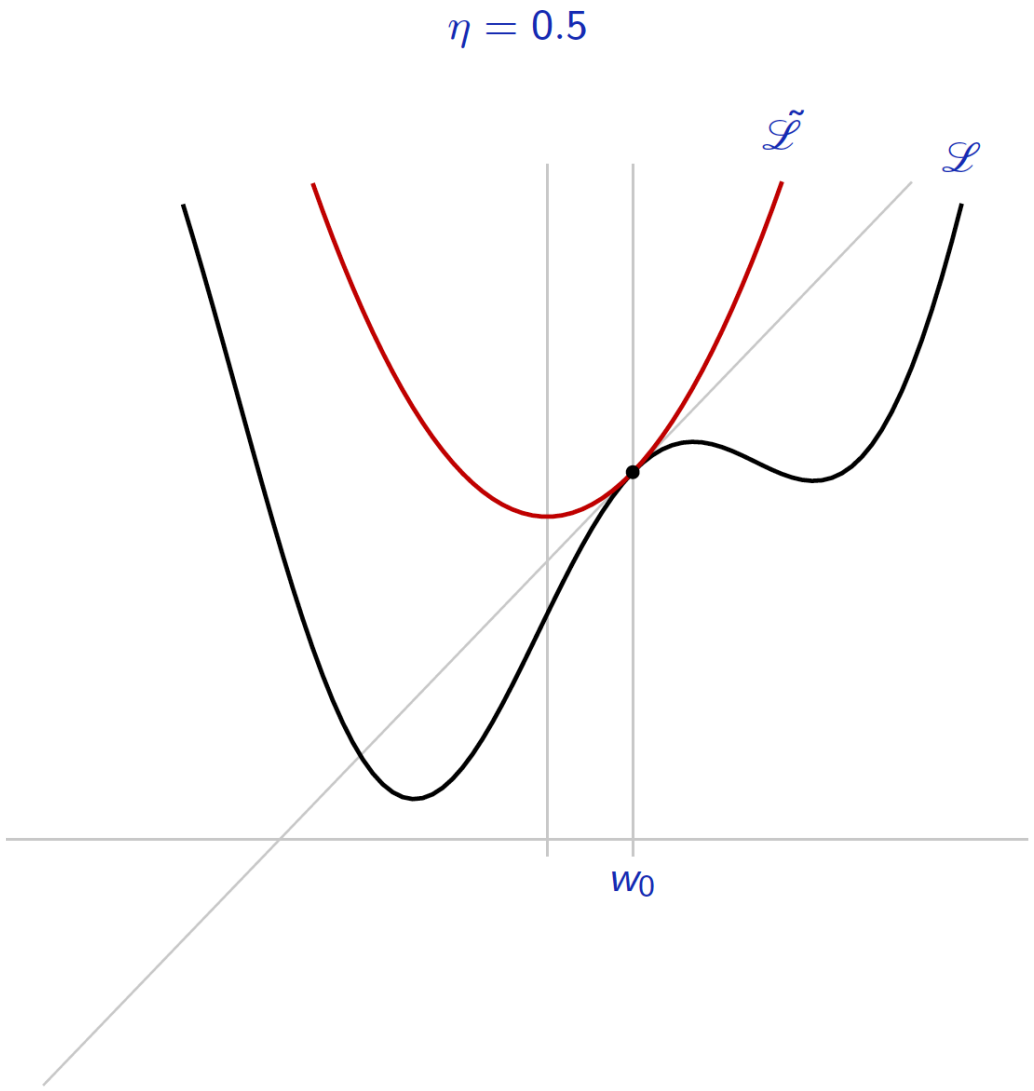
$\eta = 0.125$

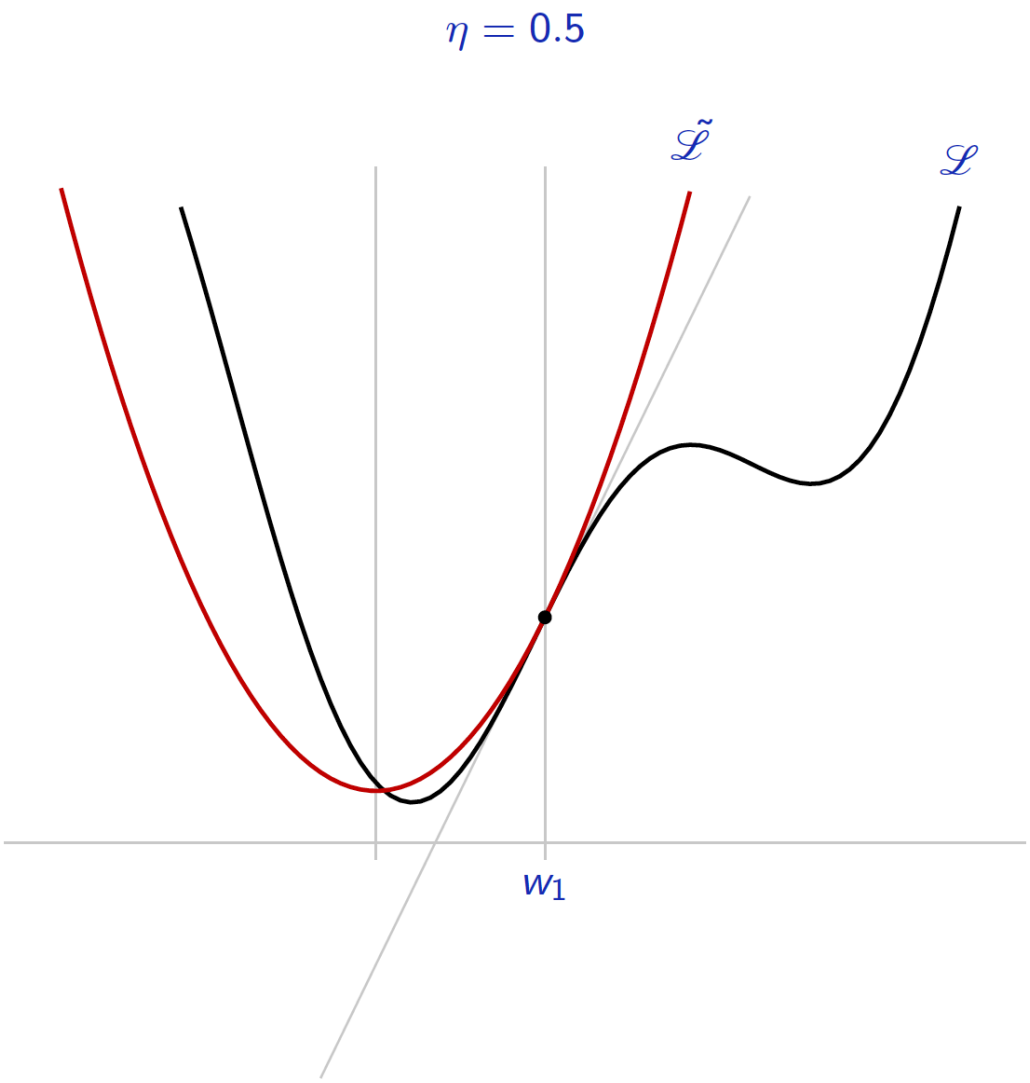


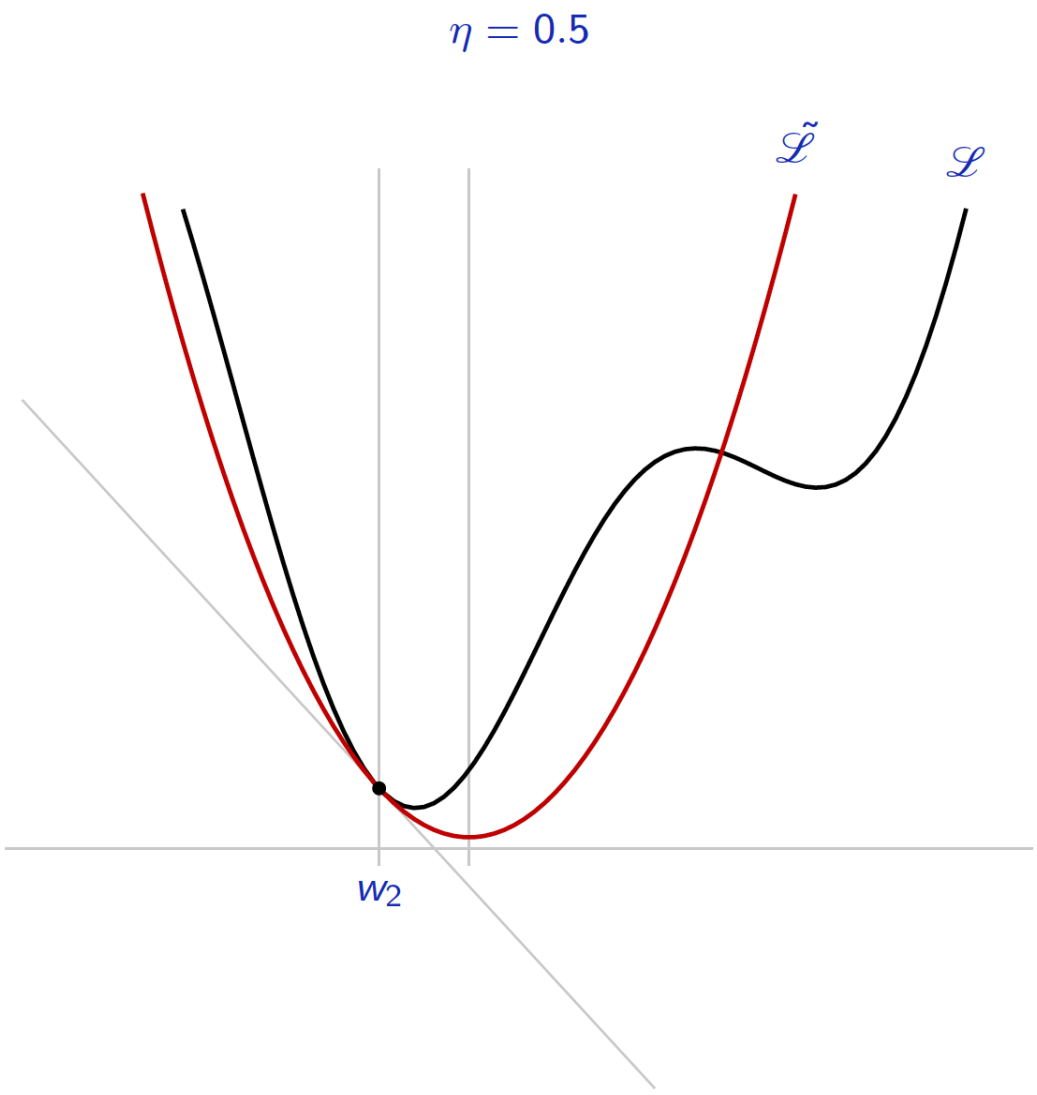

$\eta = 0.125$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_9

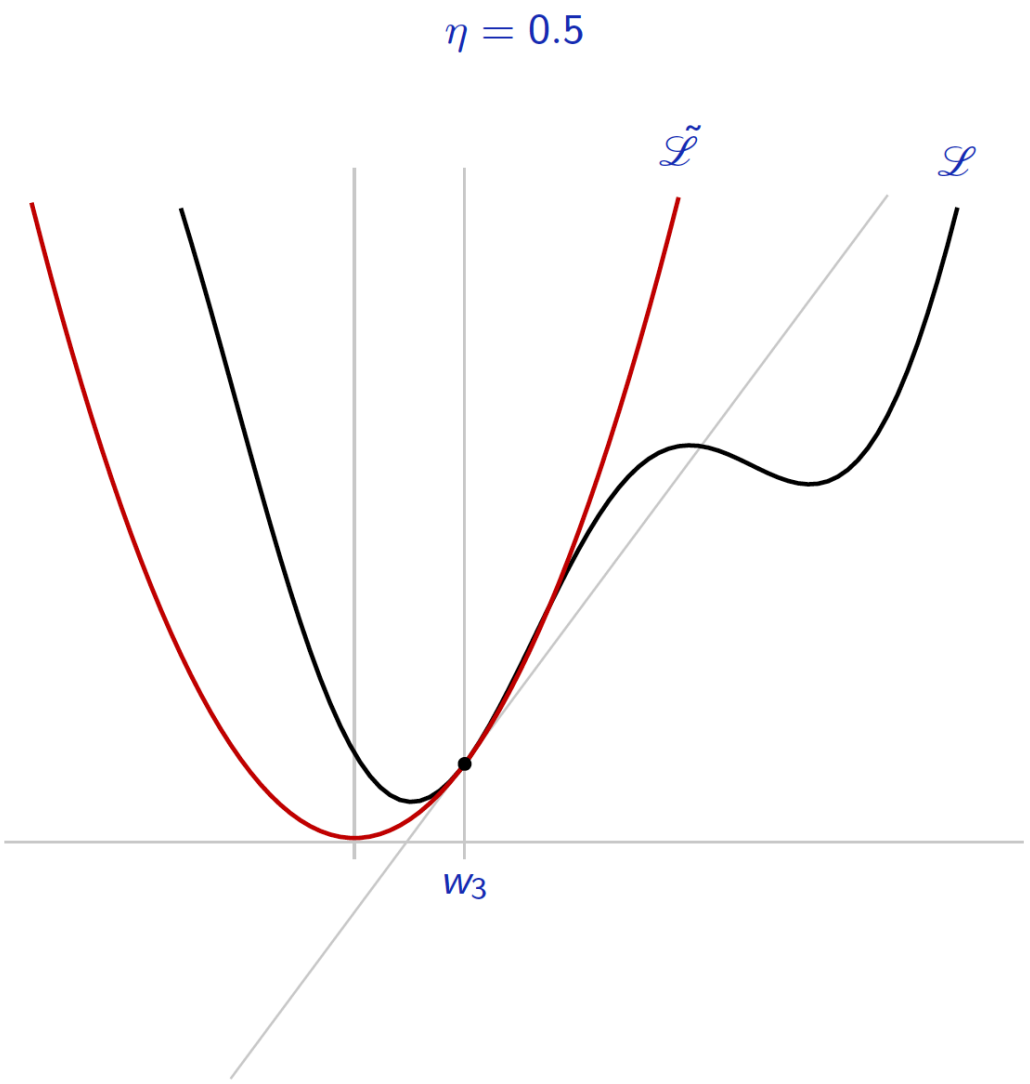

$\eta = 0.125$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_{10}


$\eta = 0.125$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_{11}

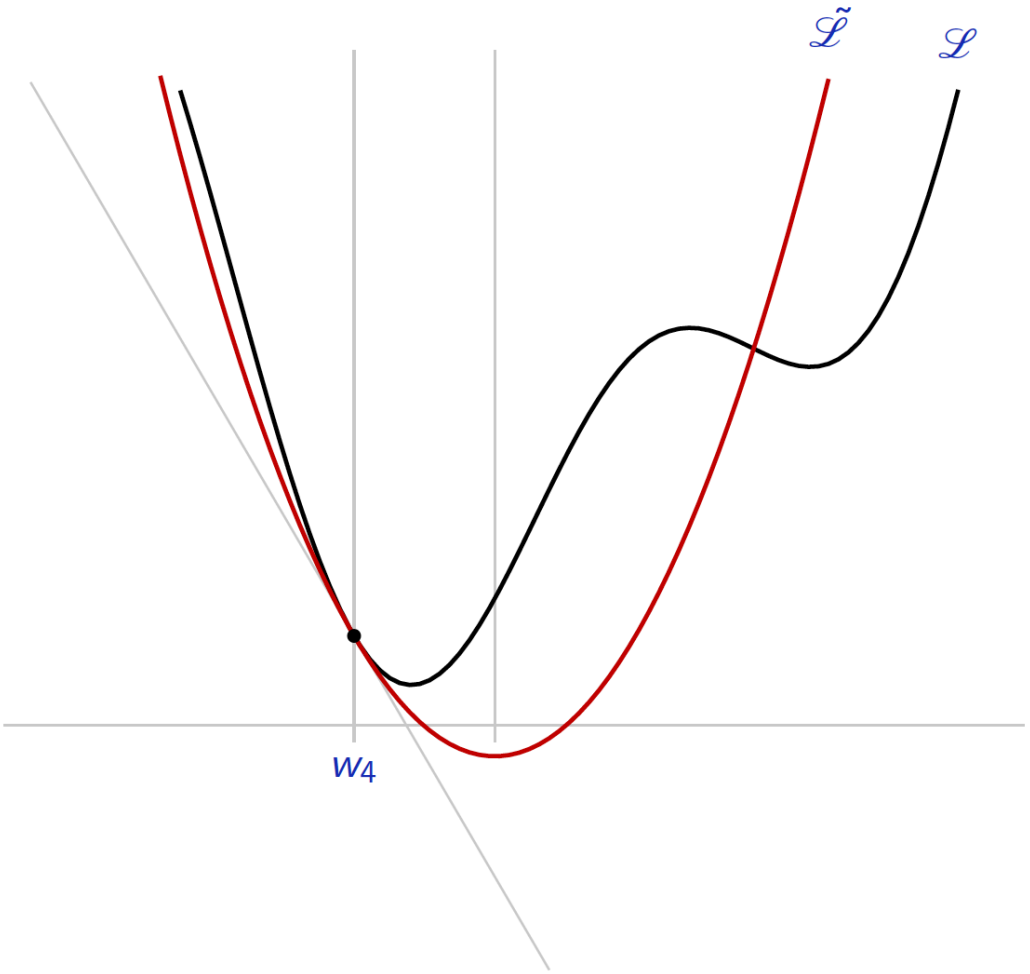

$\eta = 0.5$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_0


$\eta = 0.5$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_1

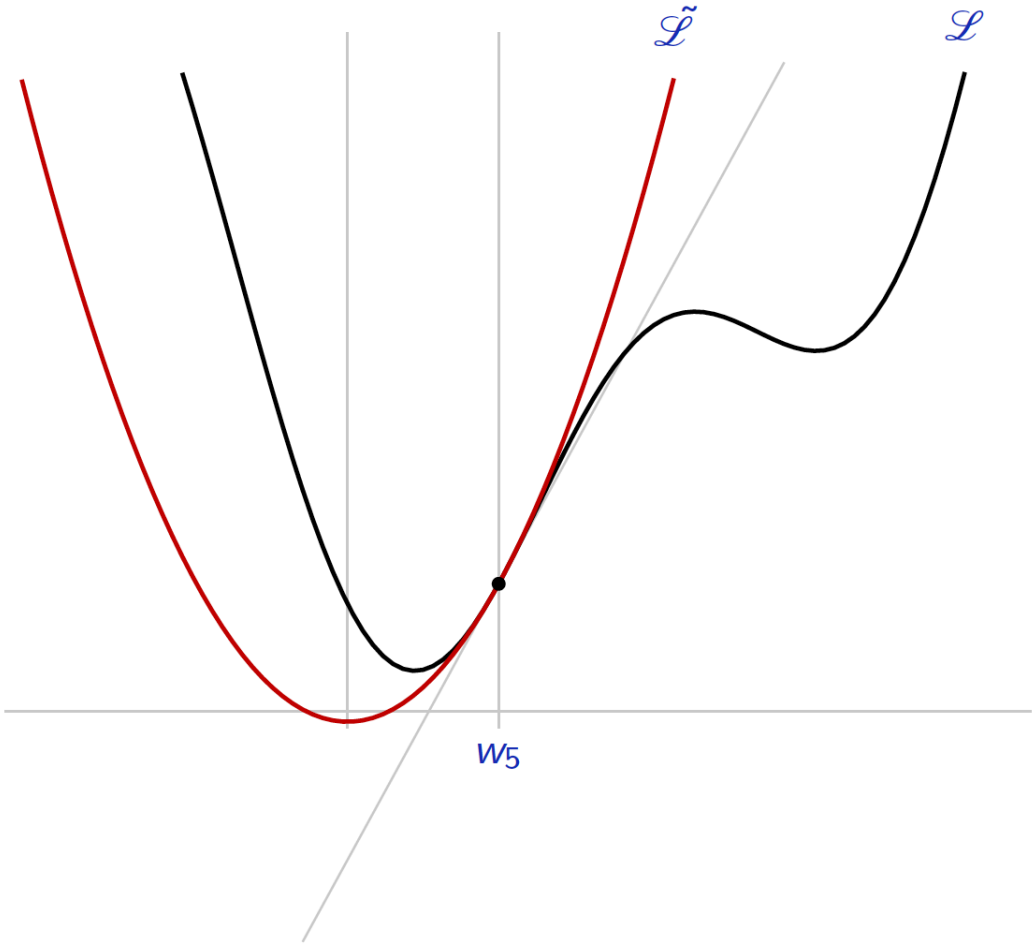

$\eta = 0.5$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_2

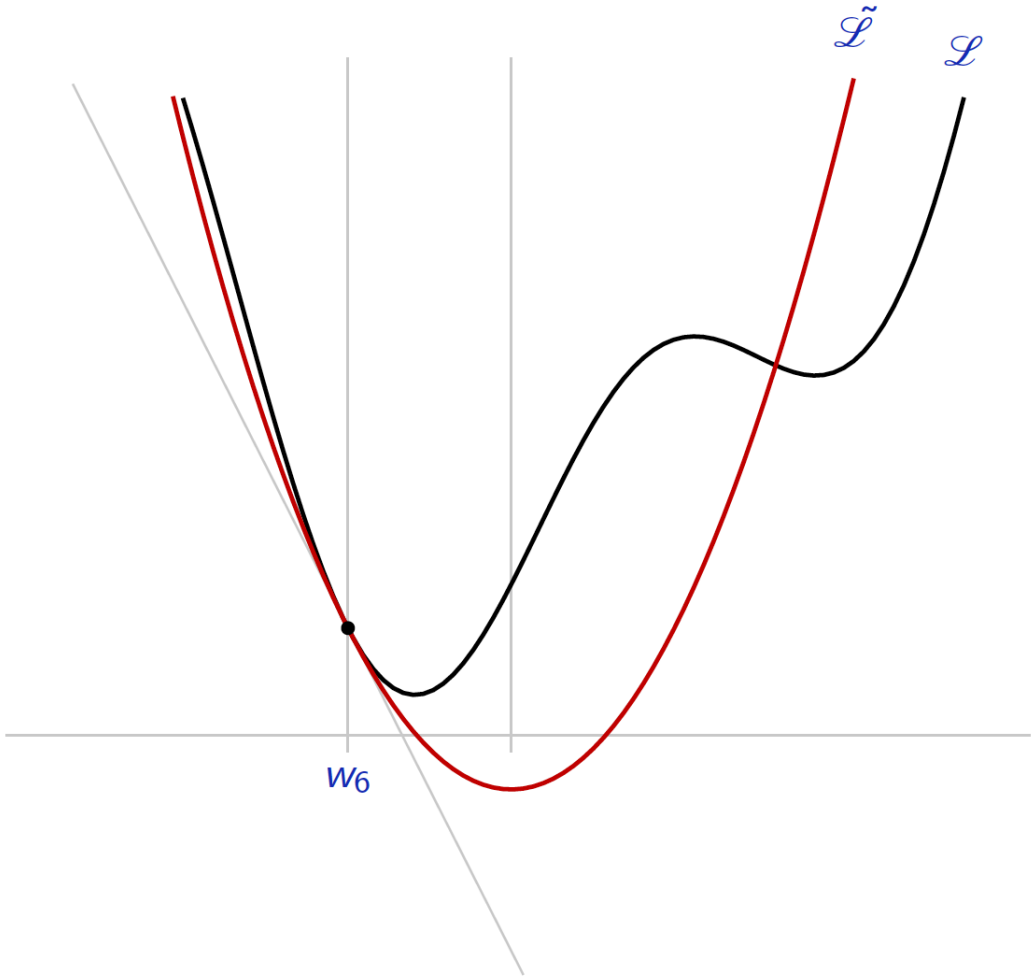

$\eta = 0.5$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_3

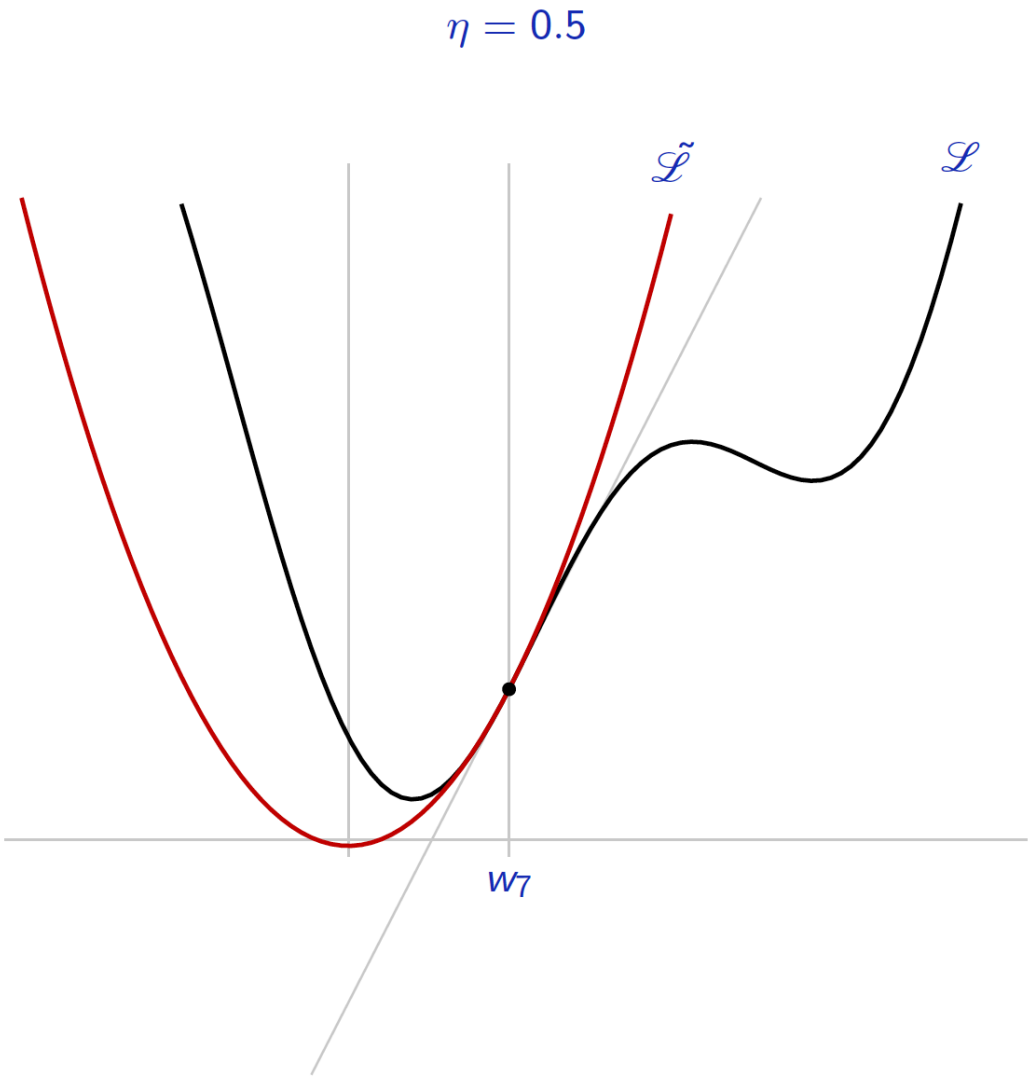
$\eta = 0.5$

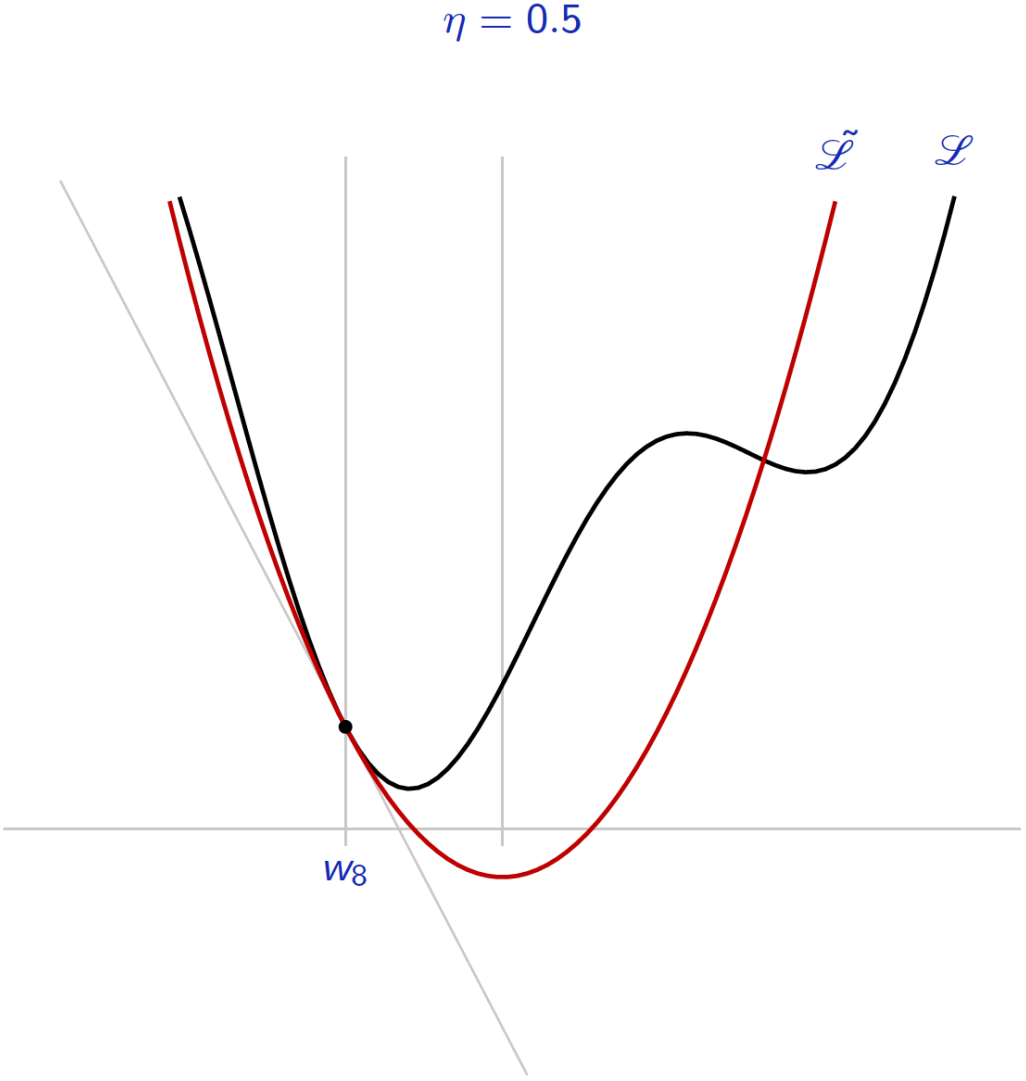


$\eta = 0.5$

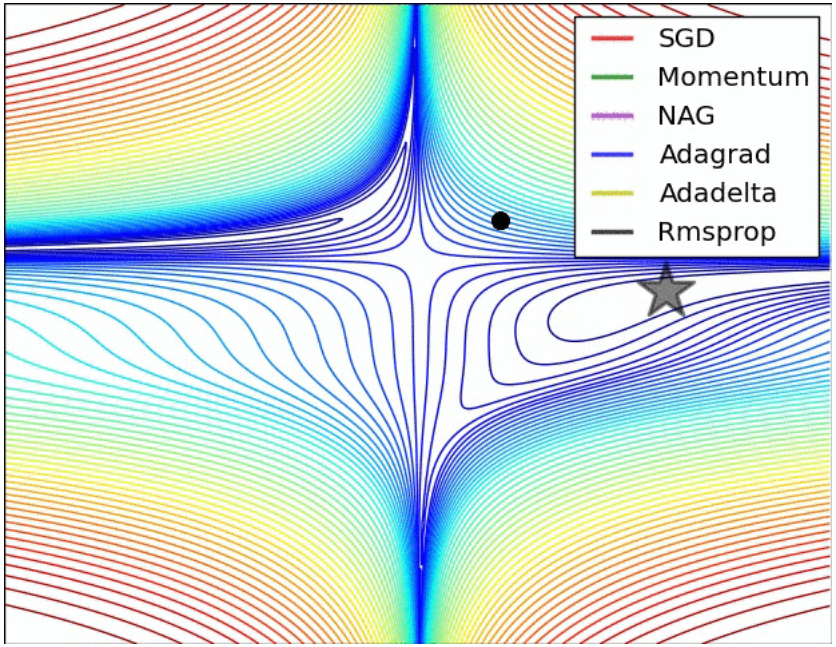


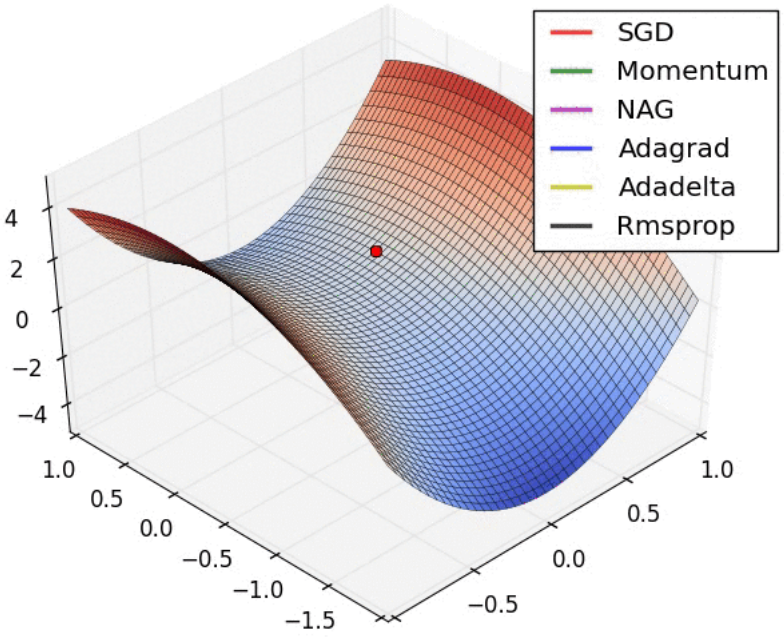

$\eta = 0.5$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_6


$\eta = 0.5$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_7


$\eta = 0.5$
$\tilde{\mathscr{L}}$
$\mathscr{L}$
w_8

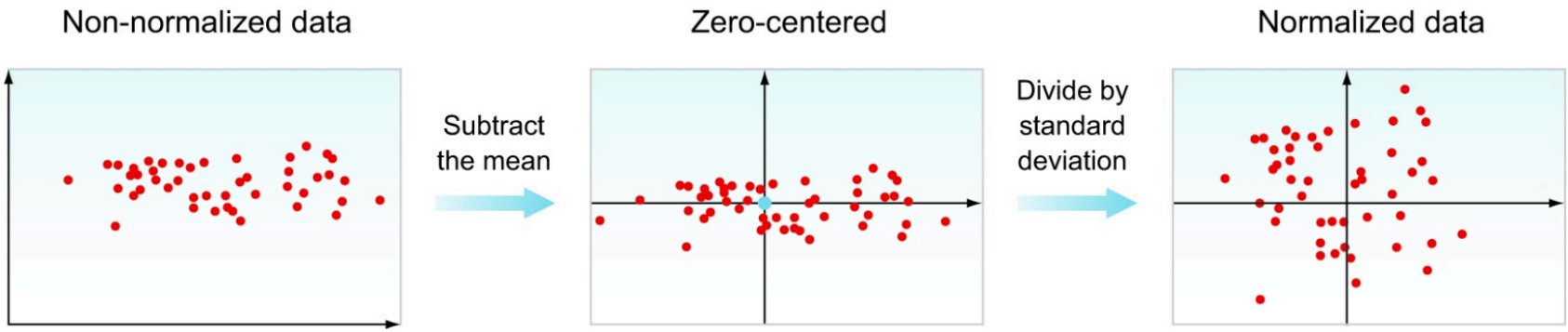
Gradient descent variants



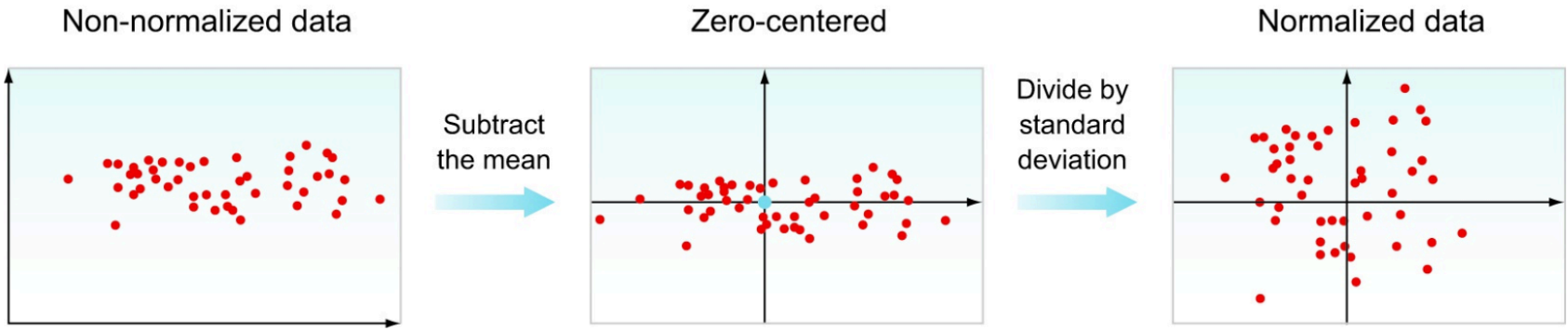


(http://ruder.io/optimizing-gradient-descent/)

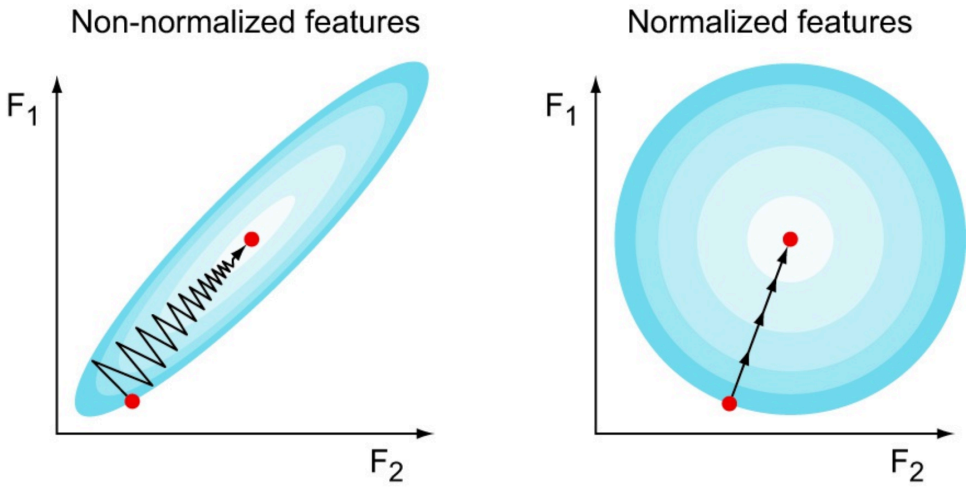
Always normalize your data



Always normalize your data



Gradient descent with and without feature scaling



Our first DL model

- Data

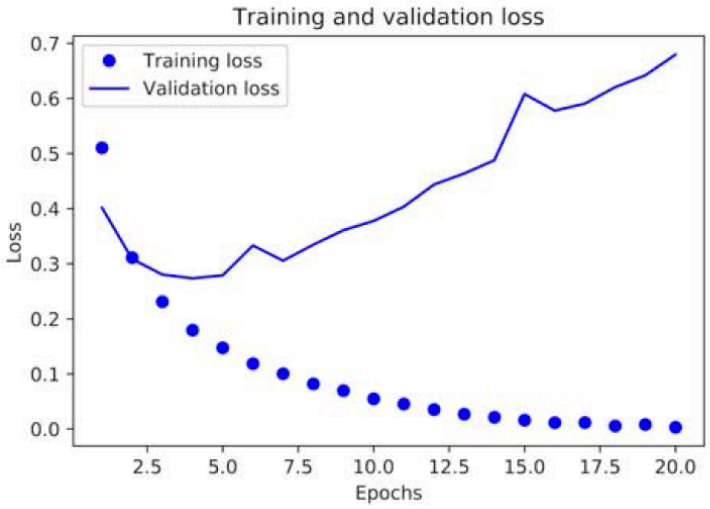
$\{x_t\}$ $\{x_{t-1}, x_{t-2}, \ldots, x_{t-h}\}$

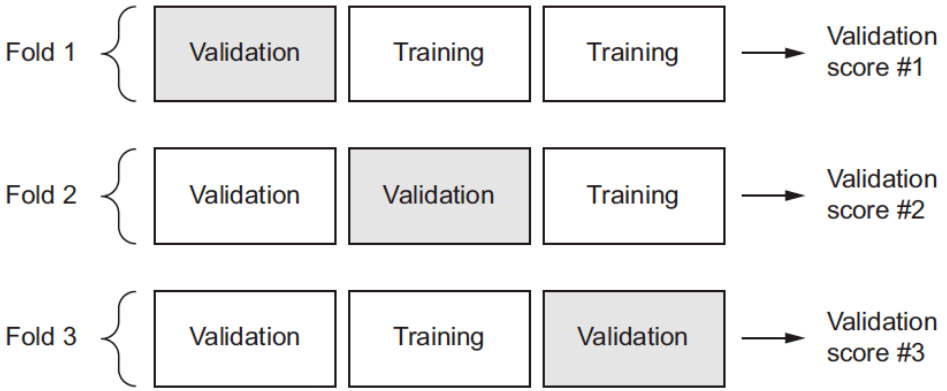
$\{x_{t-1}\}$ $\{x_{t-2}, x_{t-1}, \ldots, x_{t-h-1}\}$

- Architecture — 1 hidden layer
- Activation functions — ReLU
- Loss function — $L(f(x_i), y_i) = (f(x_i) - y_i)^2$
- Regularization
- Software frameworks

Overfitting

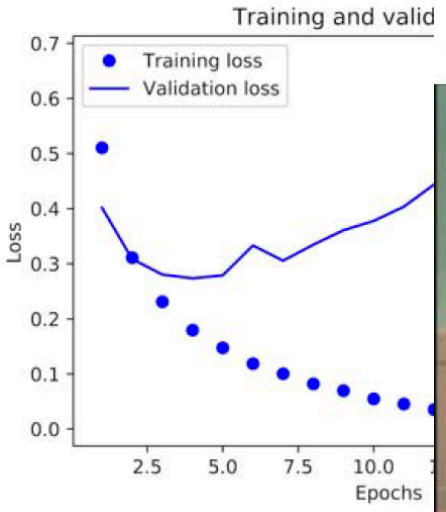
- If a model has a high capacity, it will overfit to the training data and perform poorly on unseen data

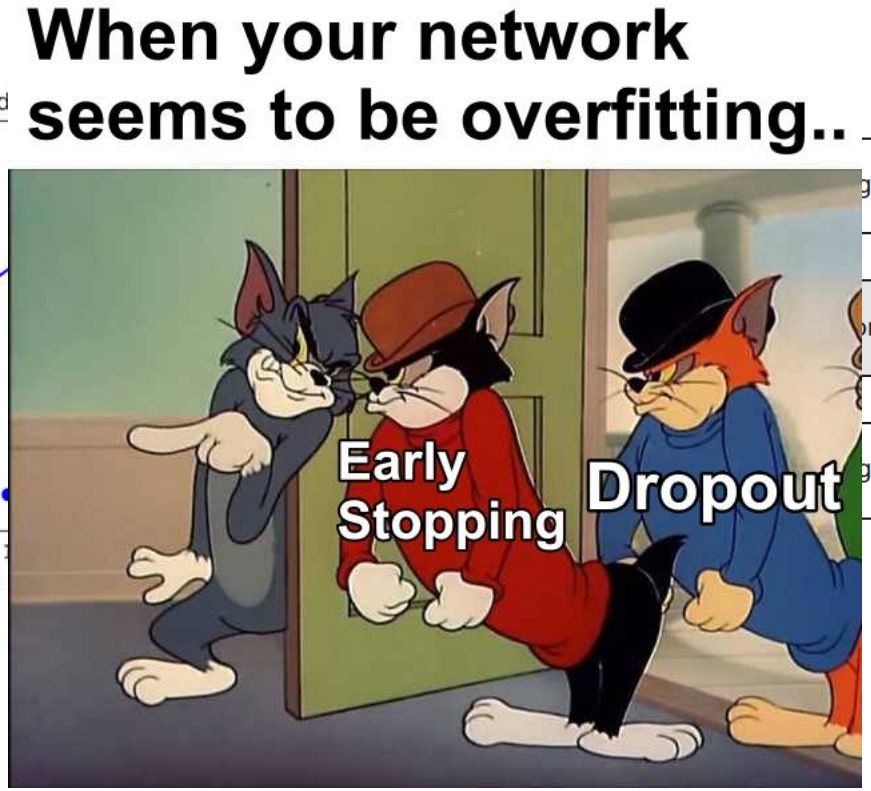


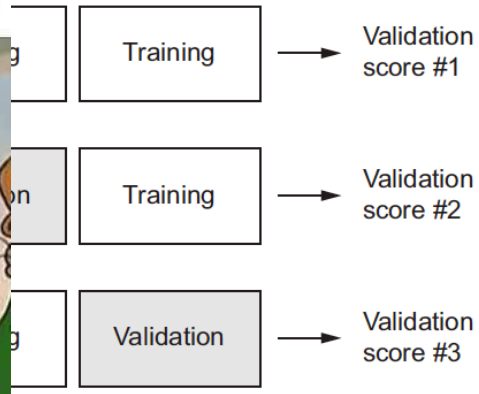


Overfitting

- If a model has a high capacity, it will overfit to the training data and perform poorly on unseen data

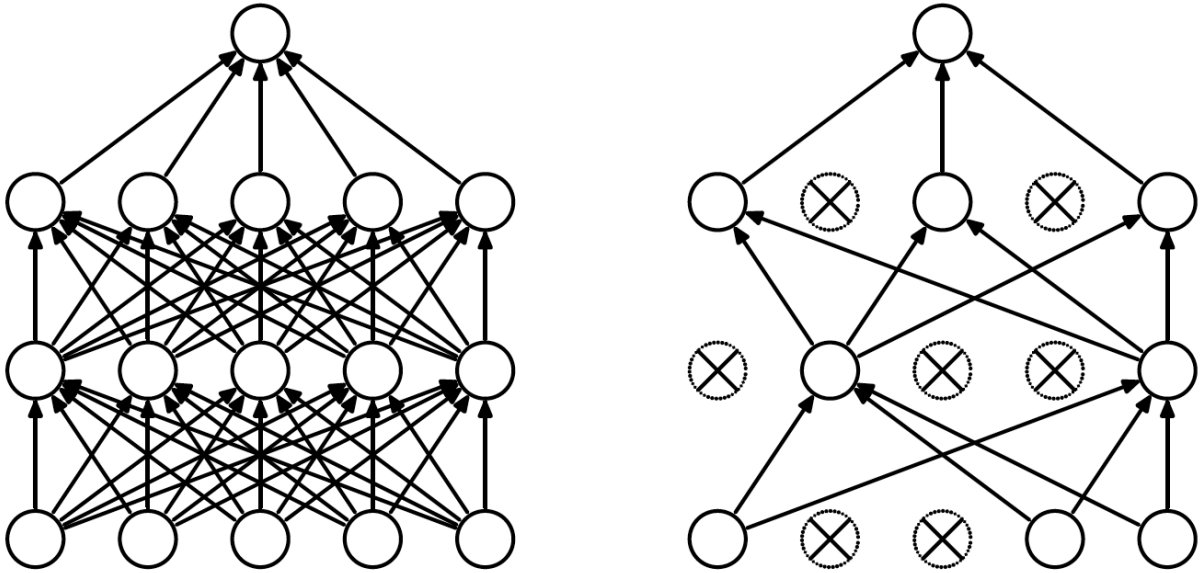






Regularization

- Dropout



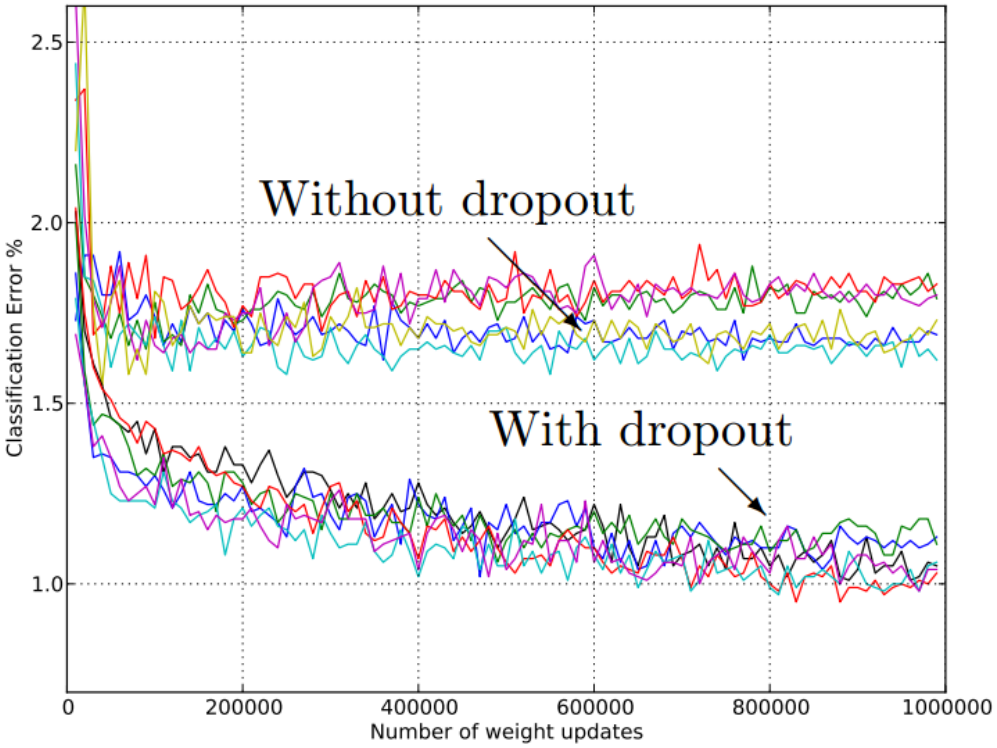
(a) Standard Neural Net

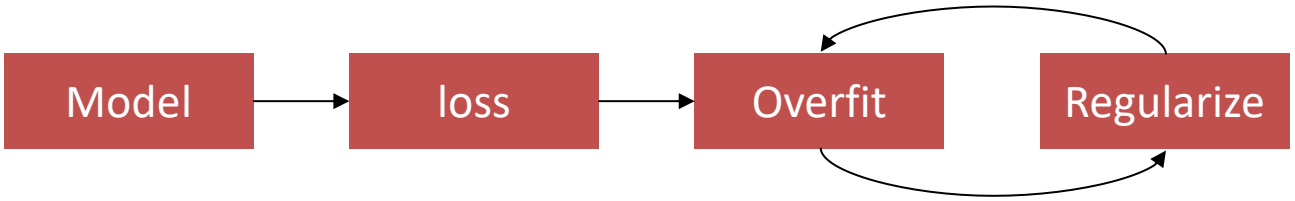
(b) After applying dropout.

https://tinyurl.com/yxhx39tb

Regularization

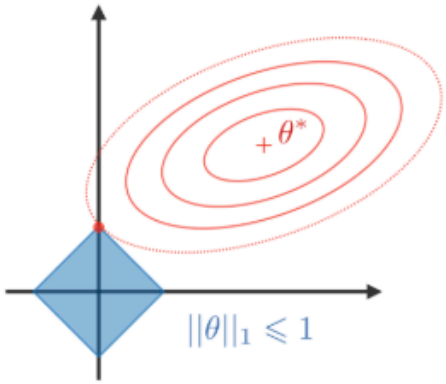
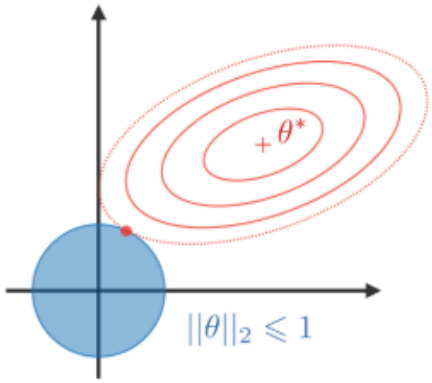
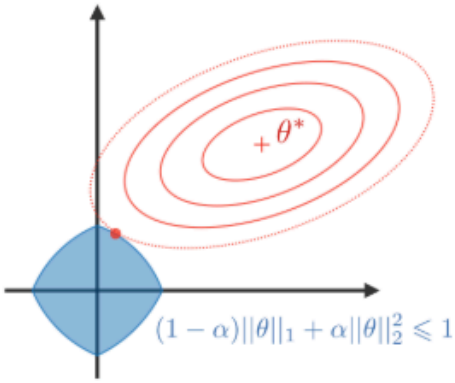
- Dropout





Regularization

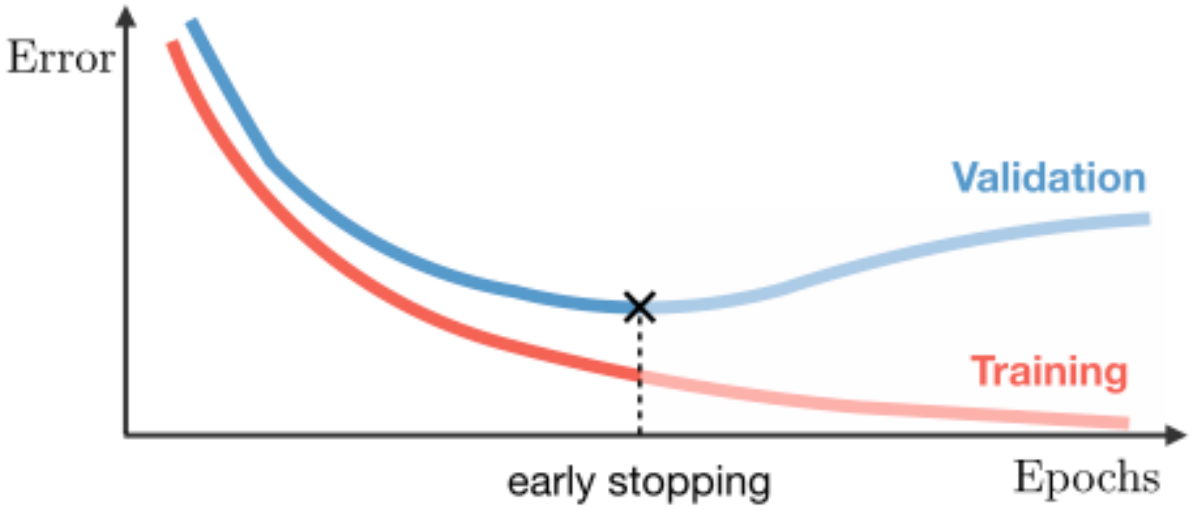
- Weight regularization

LASSO	Ridge	Elastic Net
• Shrinks coefficients to 0 • Good for variable selection	Makes coefficients smaller	Tradeoff between variable selection and small coefficients
		
$\ldots + \lambda \|\|\theta\|\|_1$ $\lambda \in \mathbb{R}$	$\ldots + \lambda \|\|\theta\|\|_2^2$ $\lambda \in \mathbb{R}$	$\ldots + \lambda \left[(1-\alpha)\|\|\theta\|\|_1 + \alpha \|\|\theta\|\|_2^2 \right]$ $\lambda \in \mathbb{R}, \alpha \in [0,1]$

https://tinyurl.com/yxhx39tb

Regularization

- Early stopping



https://tinyurl.com/yxhx39tb

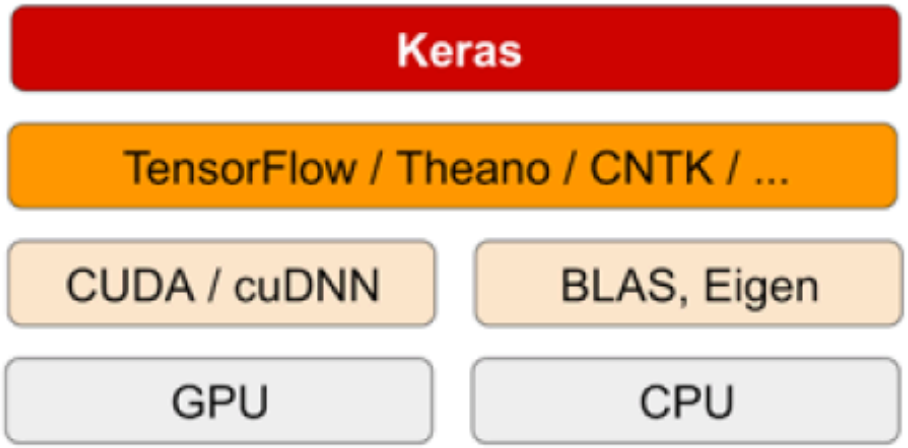
Our first DL model

- ~~Data~~
- ~~Architecture~~
- ~~Activation functions~~
- ~~Losses~~
- ~~Regularization~~
- Software frameworks

Software frameworks

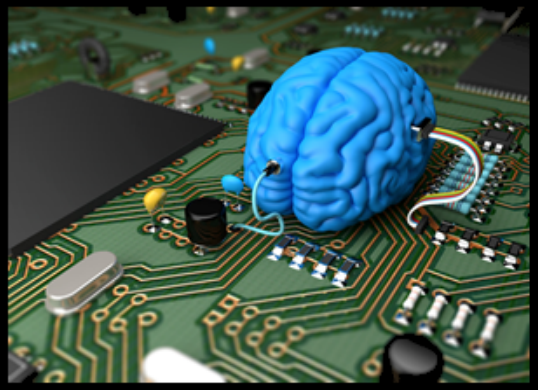
- Many, many frameworks

Lab 1



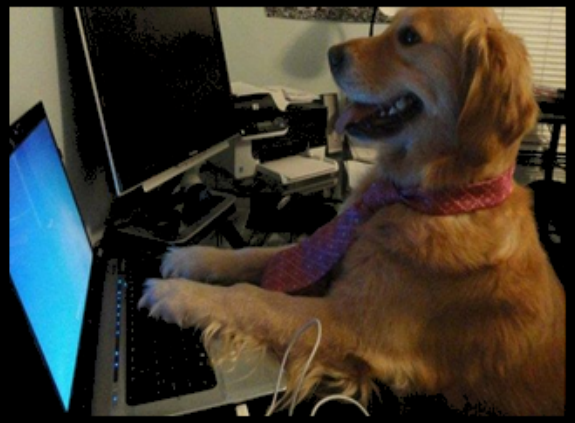
Deep Learning

What society thinks I do



What my friends think I do

What other computer scientists think I do



What mathematicians think I do

What I think I do

```
In [1]:
import keras
Using TensorFlow backend.
```

What I actually do

In the real-world …

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Deep learning for short-term traffic flow prediction

Nicholas G. Polson [a], Vadim O. Sokolov [b,*]

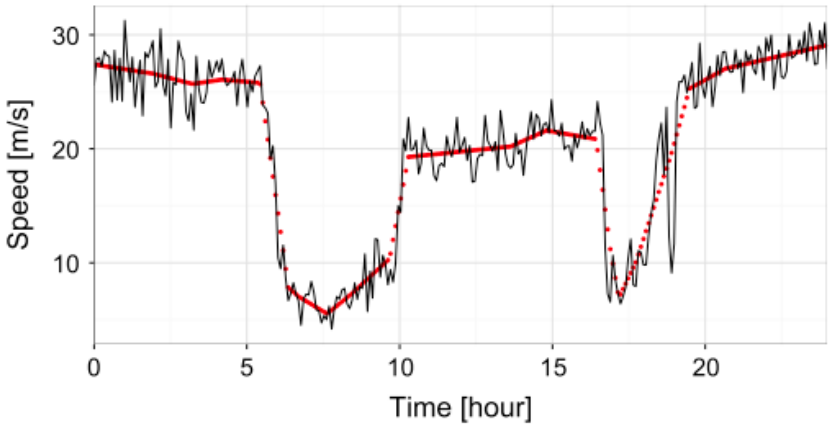
[a] Booth School of Business, University of Chicago, Chicago, IL 60637, USA
[b] Systems Engineering and Operations Research, George Mason University, Fairfax, VA 22030, USA

- Road sensor data from Interstate I-155

$$\begin{bmatrix} x_{1,t-40} & \cdots & x_t \\ \vdots & \vdots & \vdots \\ x_{n,t-40} & \cdots & x_{n,t} \end{bmatrix}$$

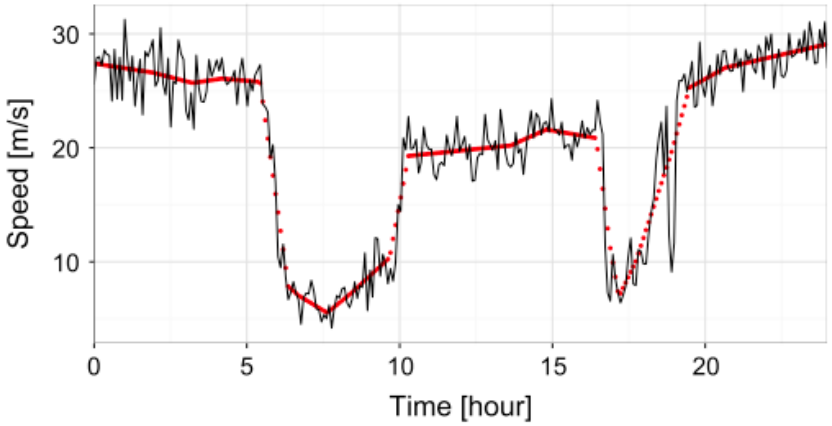
I-155 highlighted in red

In the real-world …

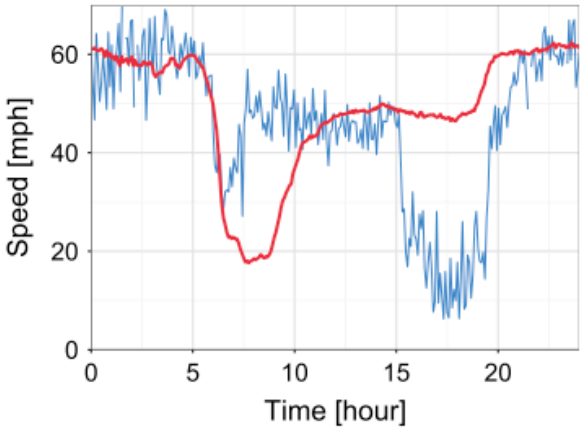


Normal day

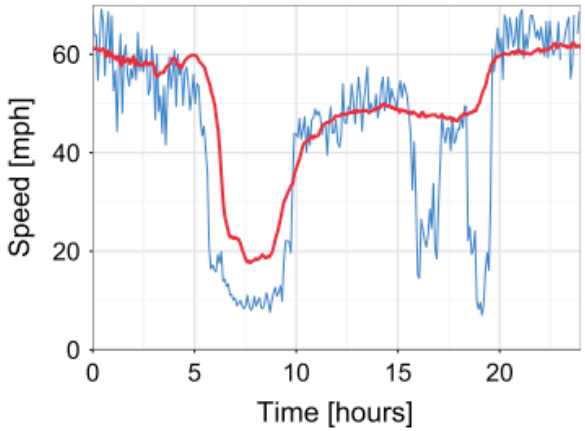
In the real-world …



Normal day



(a) Chicago Bears football game

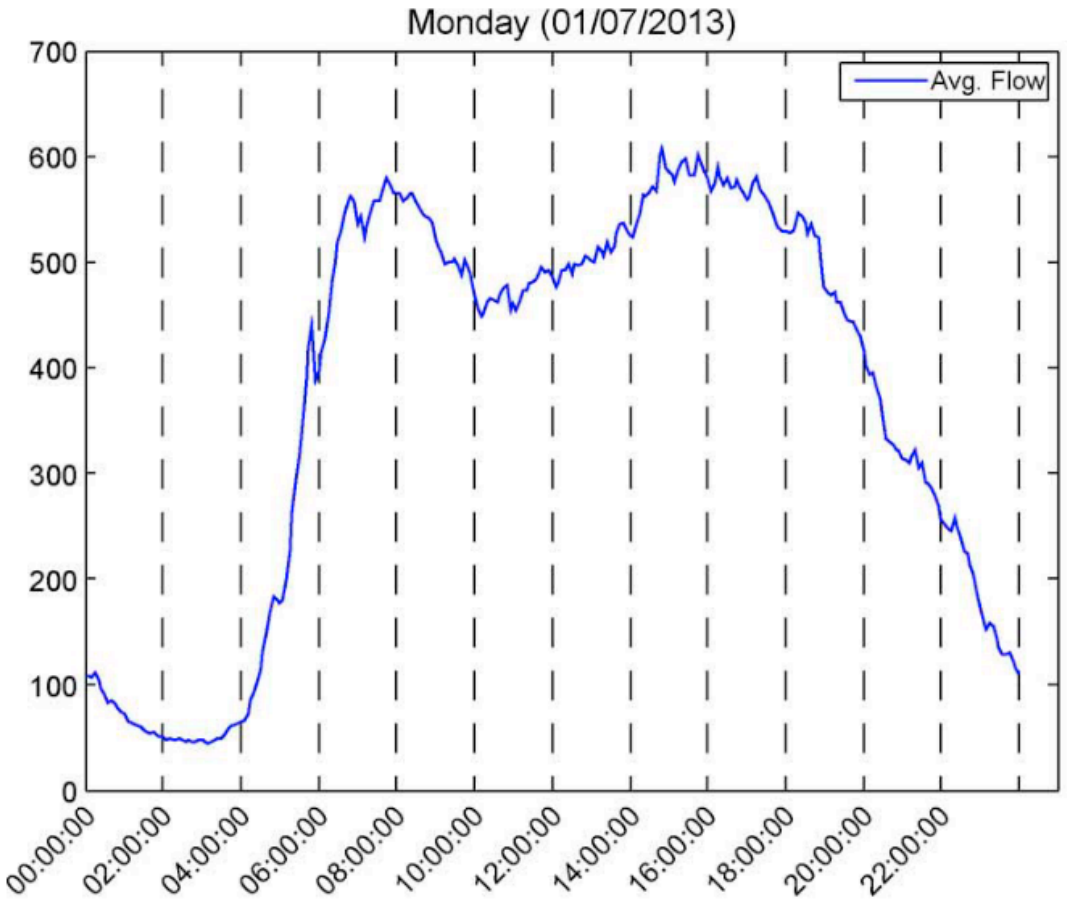


(b) Snow weather

Events

A simple problem

- Predict traffic flow for the next 15 minutes



A bit more complicated problem

- Predict traffic flow for the next 15 minutes
- There are 15,000 detectors
- Do we train 15,000 models?

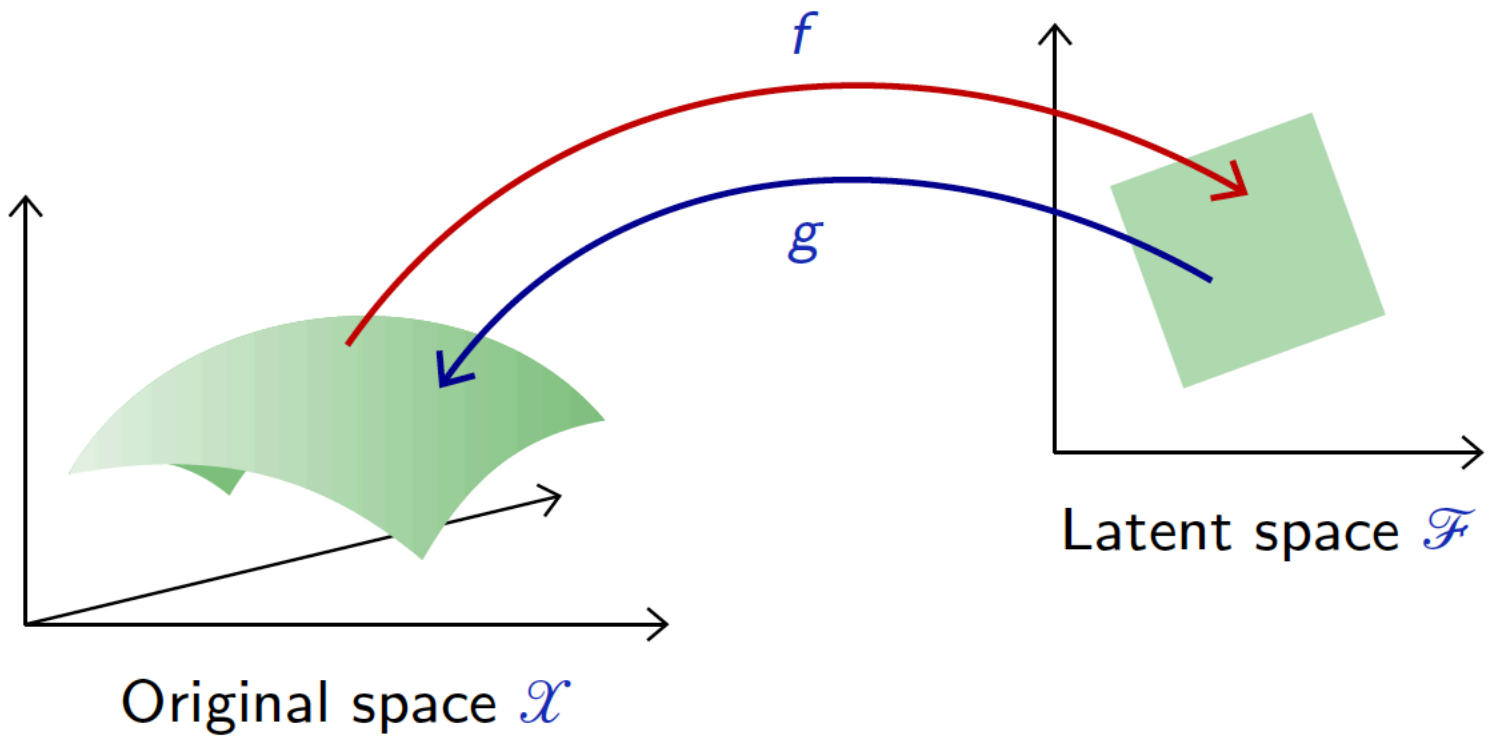
A bit more complicated problem

- Predict traffic flow for the next 15 minutes
- There are 15,000 detectors
- Do we train 15,000 models?

- First idea:
  - There is a lot of overlap between the detectors
  - Signal/noise ratio is low
  - The useful information must be in a lower dimension than 15,000
  - **Dimensionality reduction!**

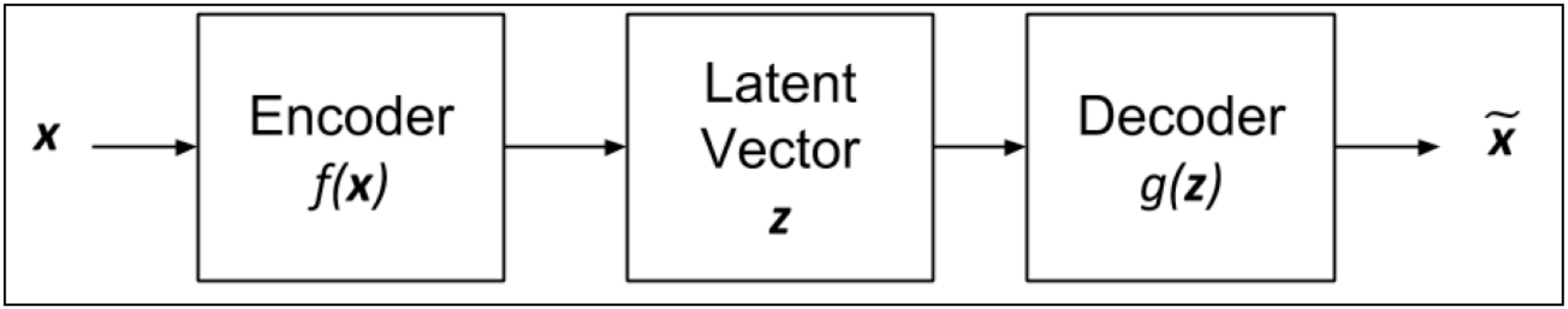
Deep Autoencoder

- Try to learn a latent representation of the input



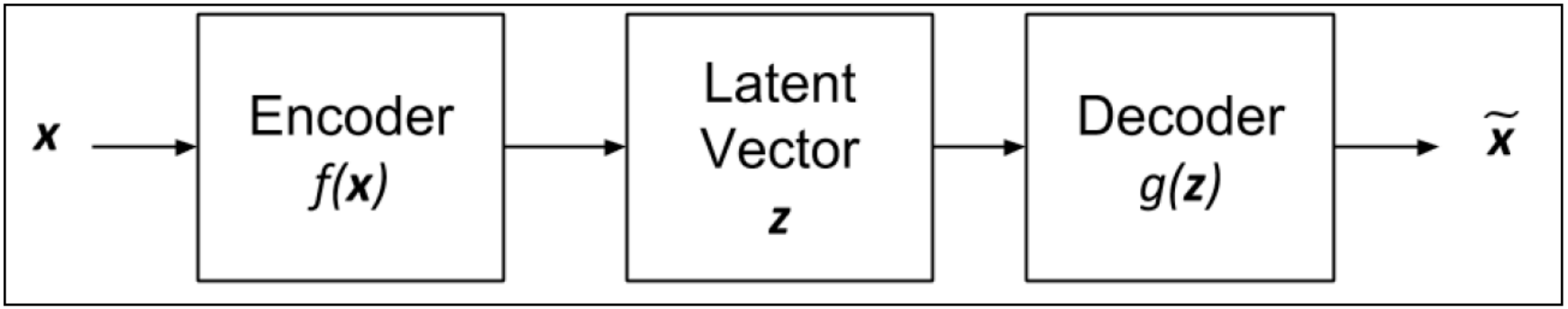
Deep Autoencoder

- Try to learn a latent representation of the input



Deep Autoencoder

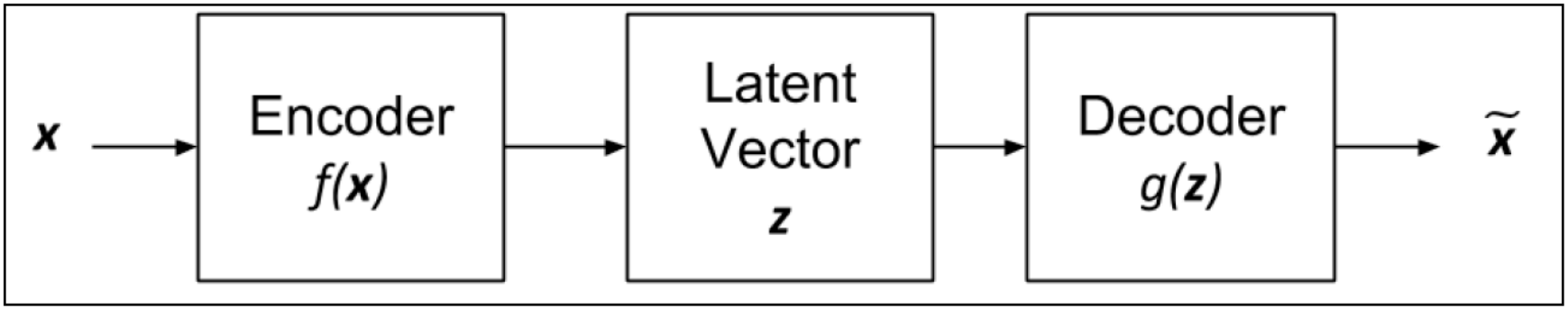
- Try to learn a latent representation of the input

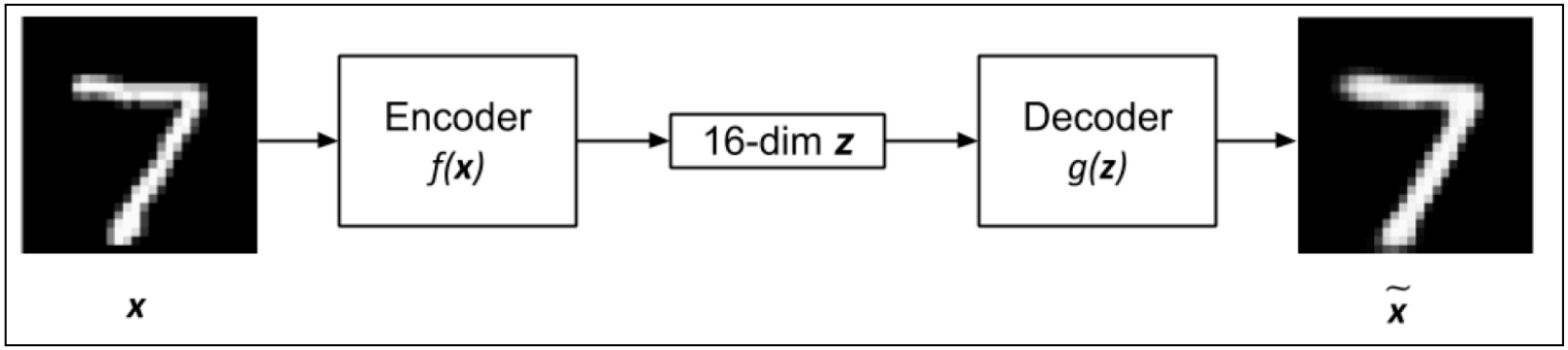


$$y(x) = f(W_1 x + b)$$
$$z(x) = g(W_2 y(x) + c)$$

Deep Autoencoder

- Try to learn a latent representation of the input



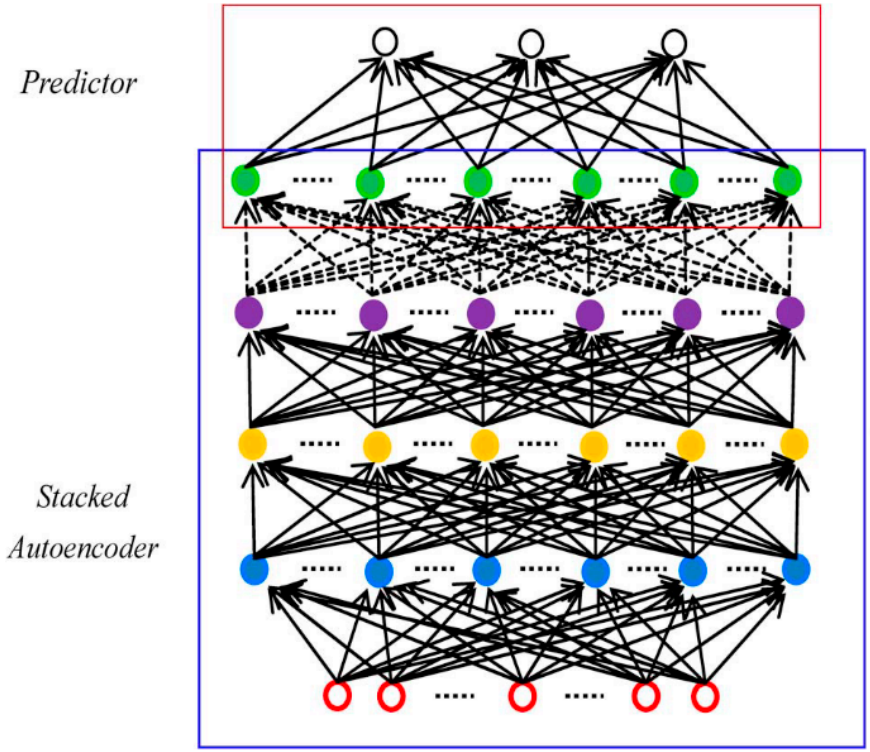


In the real-world …

IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 2, APRIL 2015

Traffic Flow Prediction With Big Data: A Deep Learning Approach

Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang, *Fellow, IEEE*



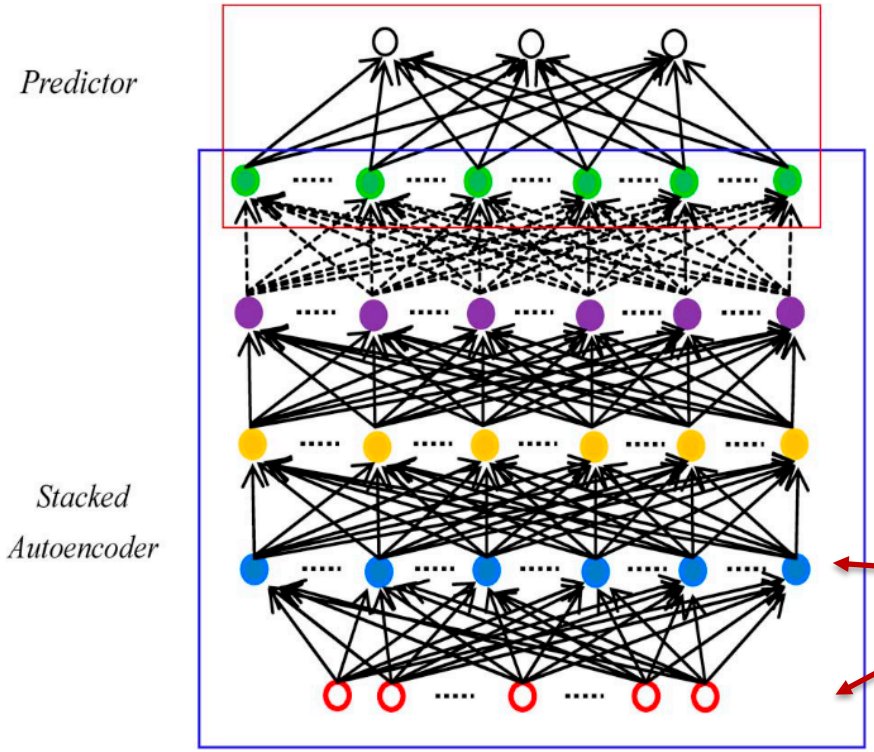
Feature extraction

In the real-world …

IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 2, APRIL 2015 865

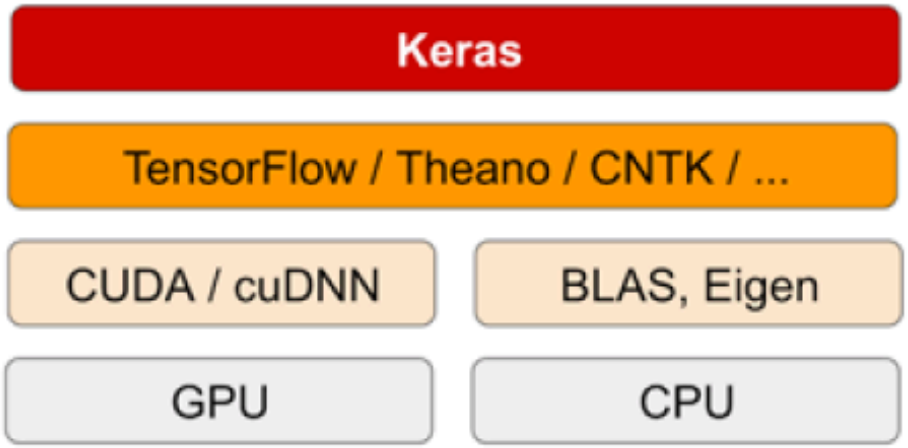
Traffic Flow Prediction With Big Data: A Deep Learning Approach

Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang, *Fellow, IEEE*



Do you see the problem?

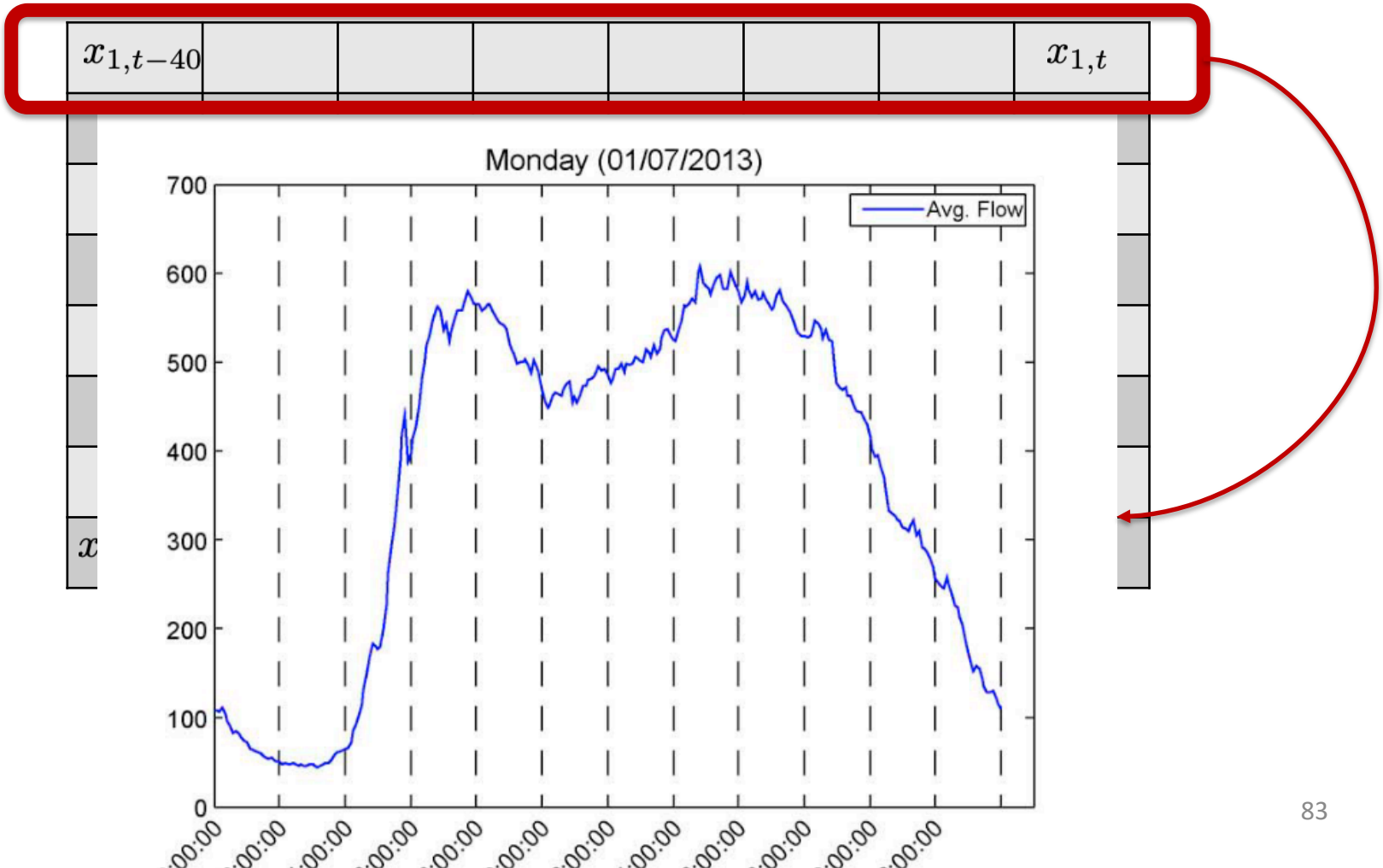
Lab 2



Problems with MLP

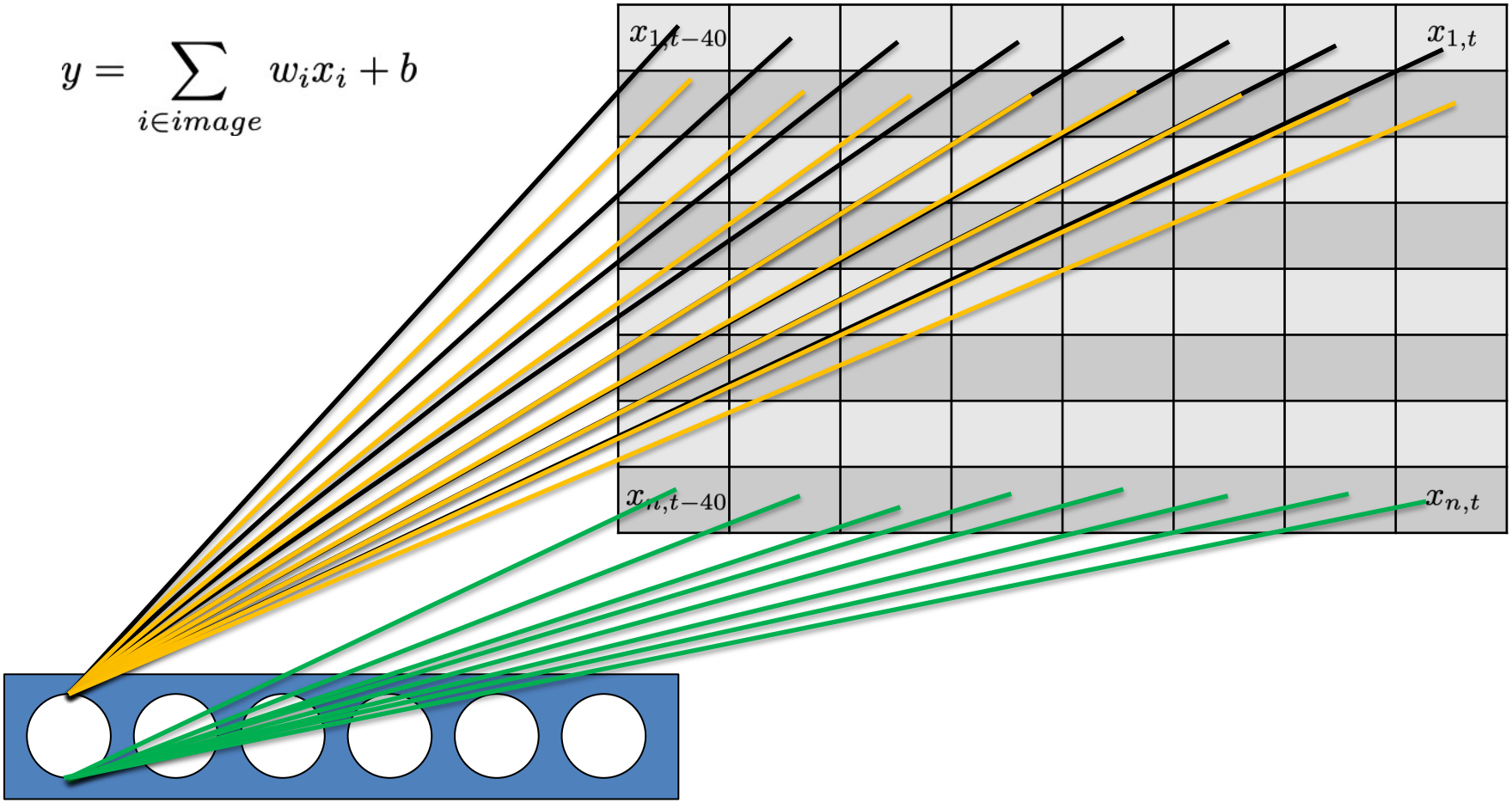
$x_{1,t-40}$							$x_{1,t}$
$x_{n,t-40}$							$x_{n,t}$

Problems with MLP



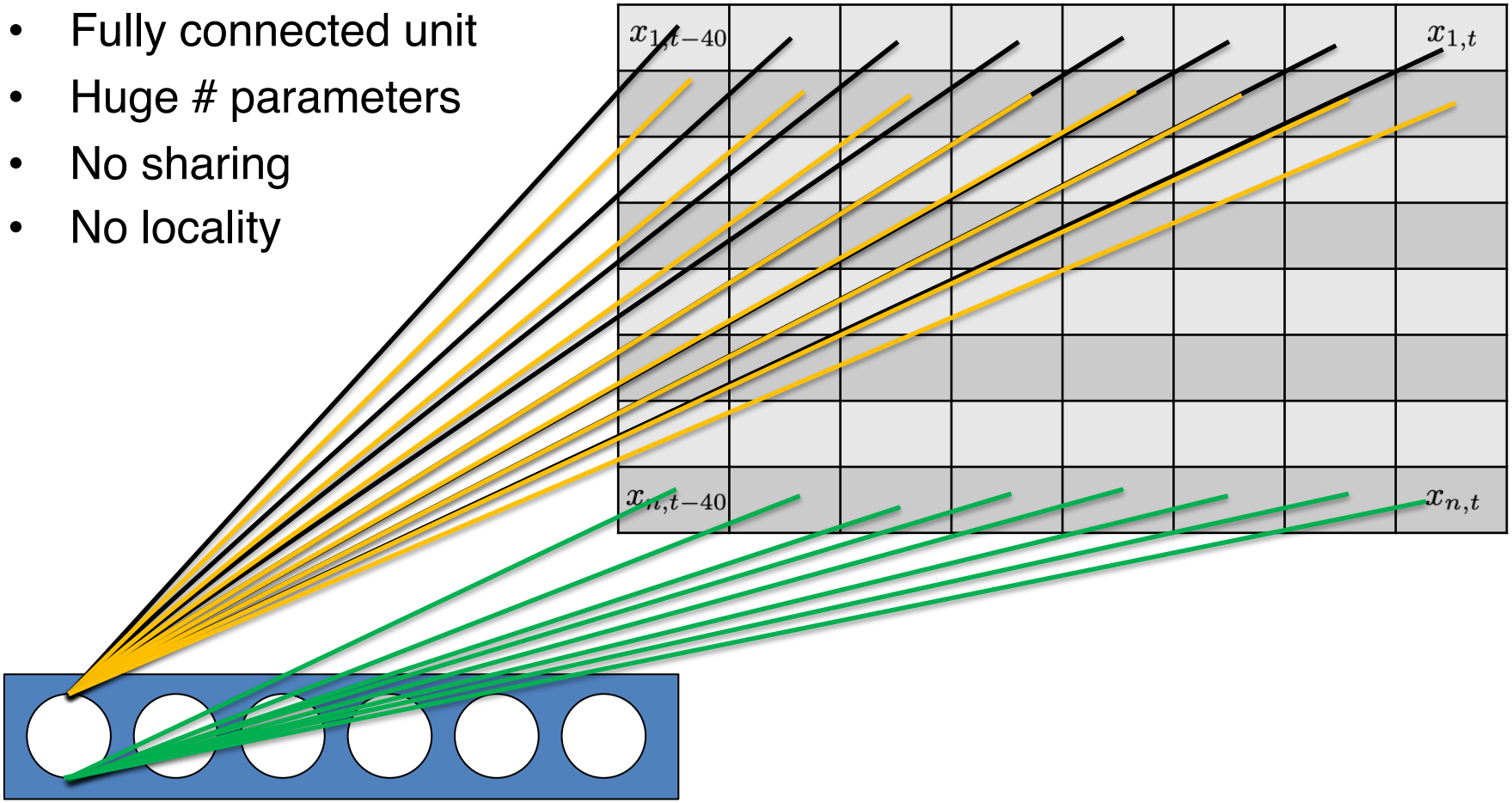
Problems with MLP

$$y = \sum_{i \in image} w_i x_i + b$$



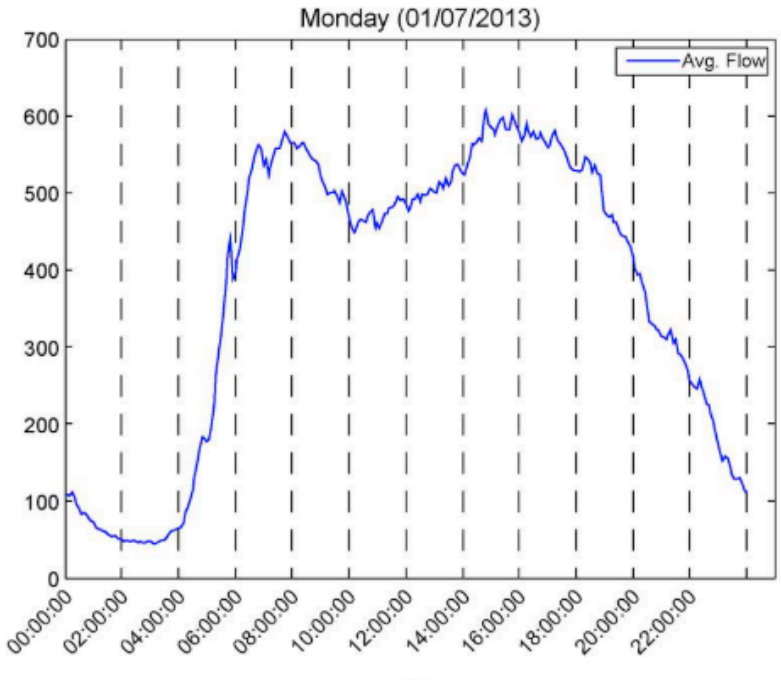
Problems with MLP

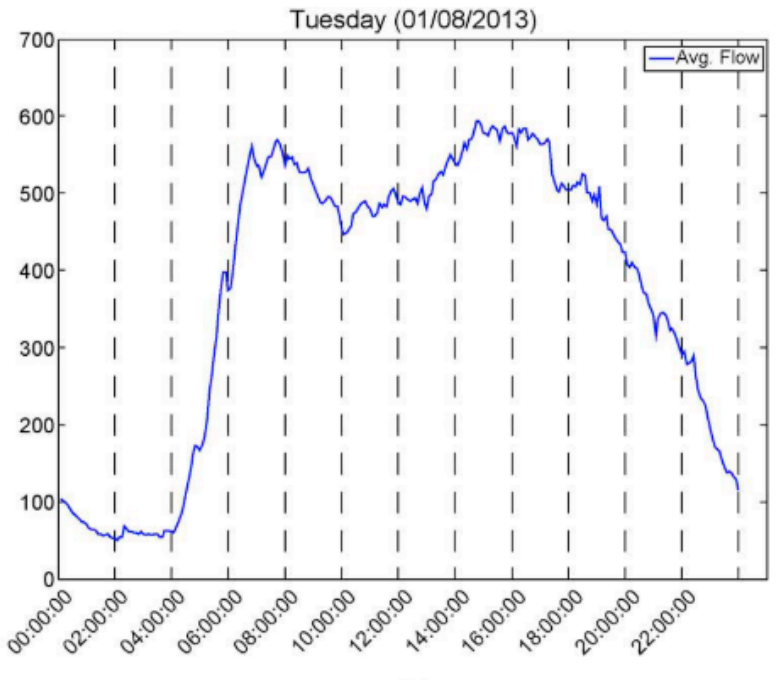
- Fully connected unit
- Huge # parameters
- No sharing
- No locality



Problems with MLP

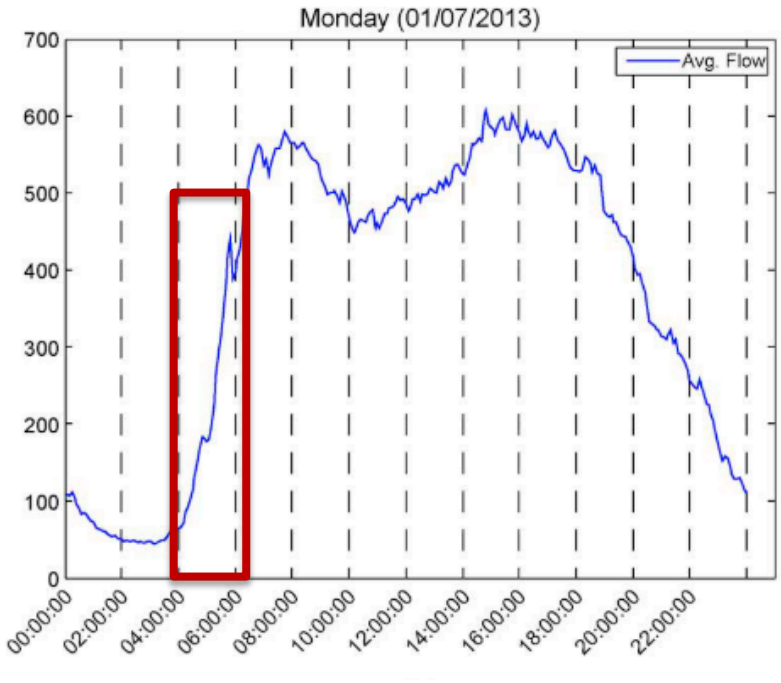
- locality

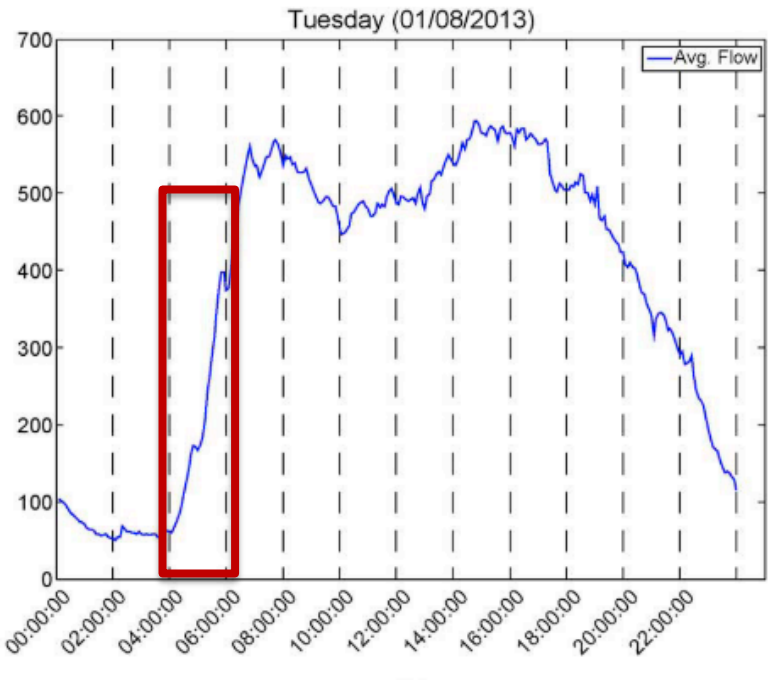




Problems with MLP

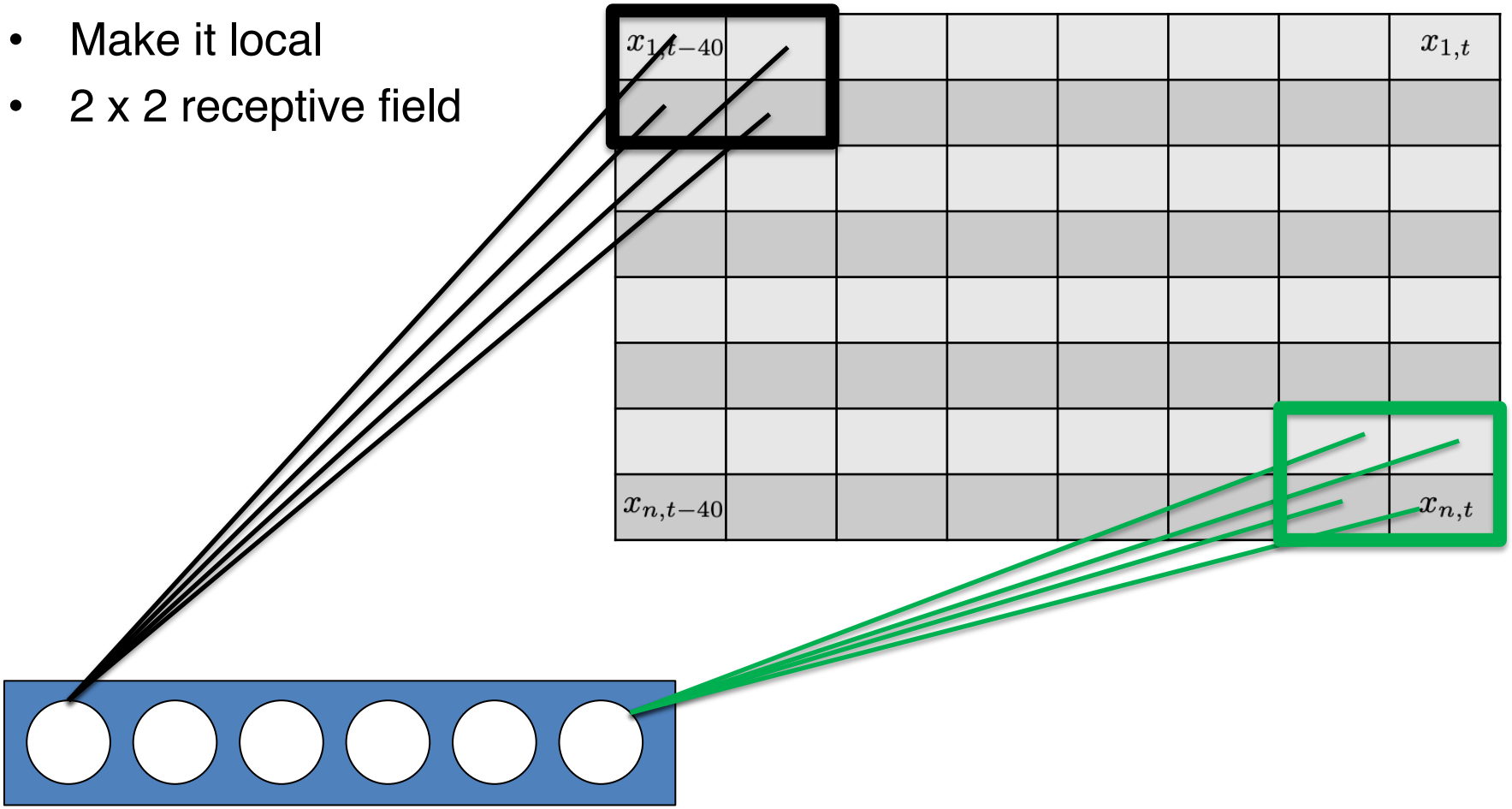
- locality





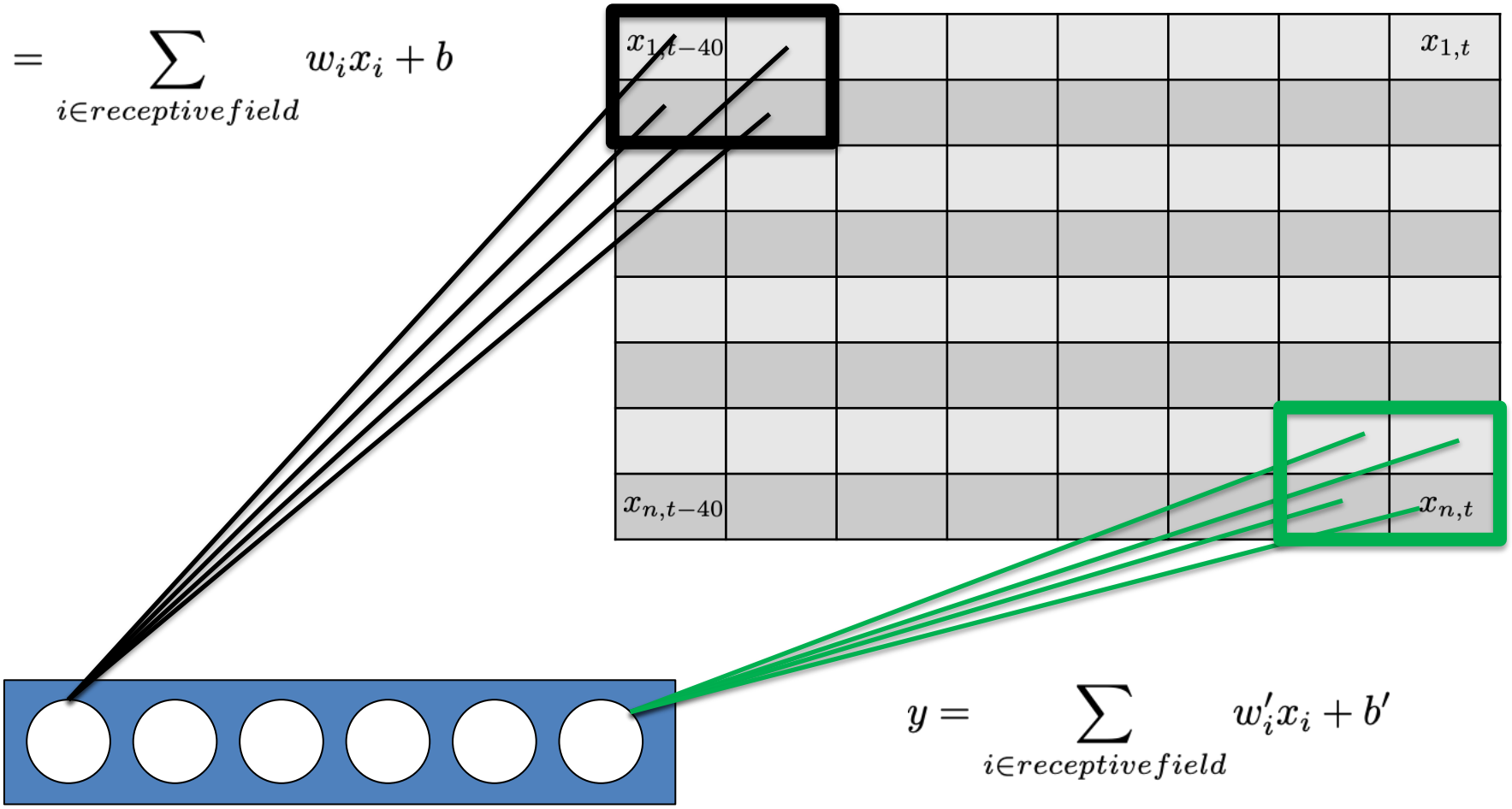
Make it local

- Make it local
- 2 x 2 receptive field



Make it local

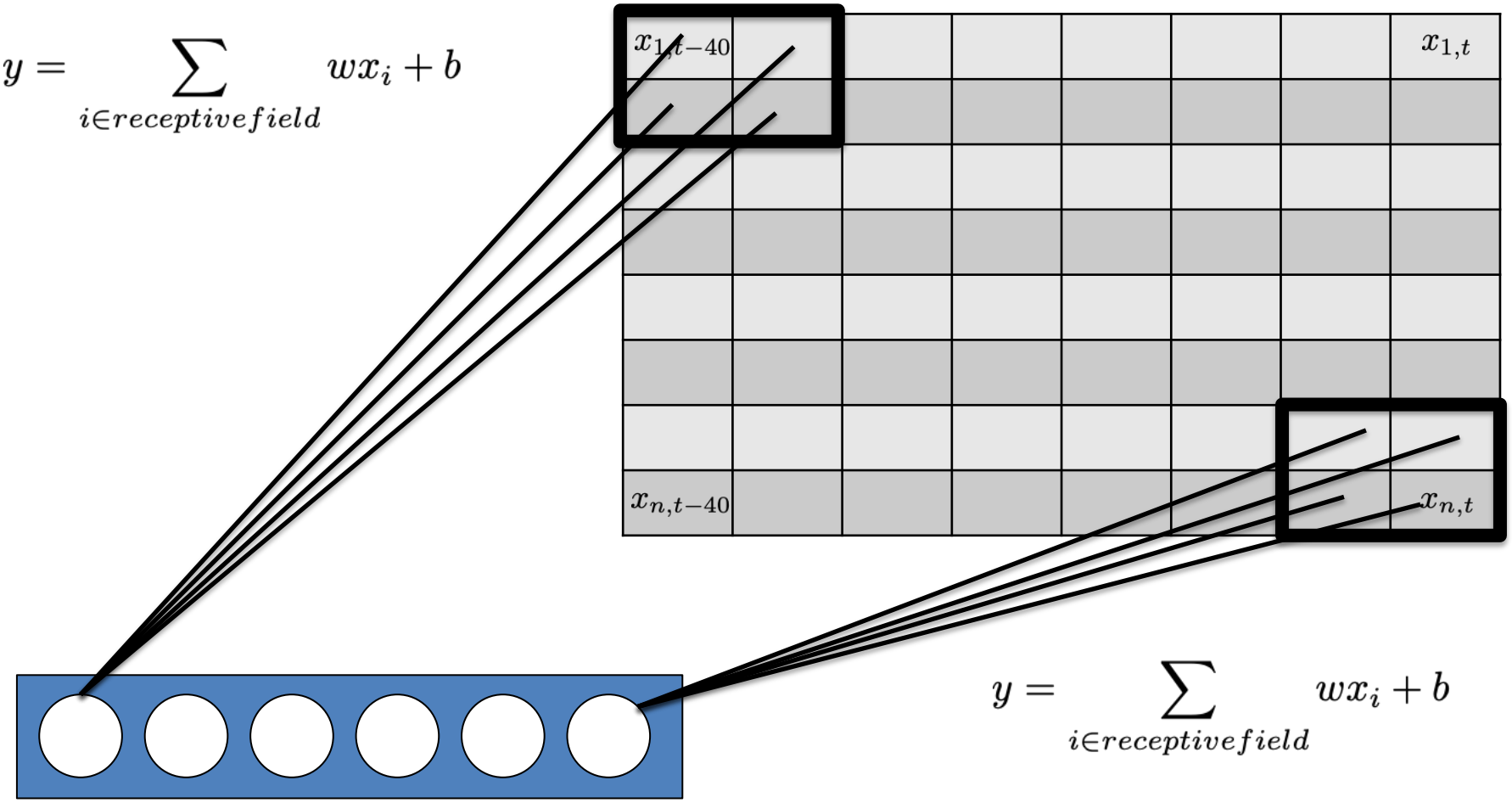
$$y = \sum_{i \in receptive field} w_i x_i + b$$



$$y = \sum_{i \in receptive field} w'_i x_i + b'$$

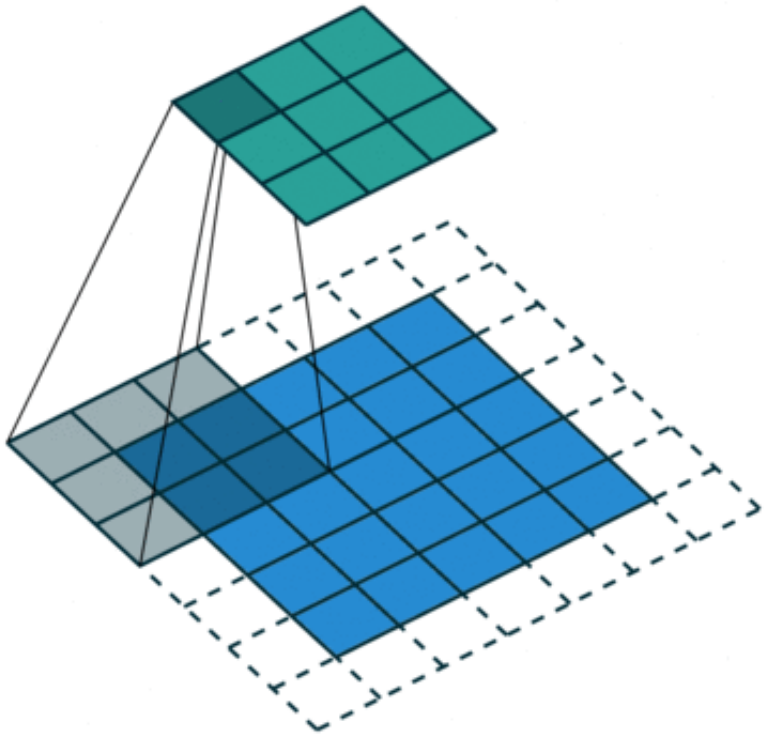
Share weights

$$y = \sum_{i \in receptive field} w x_i + b$$

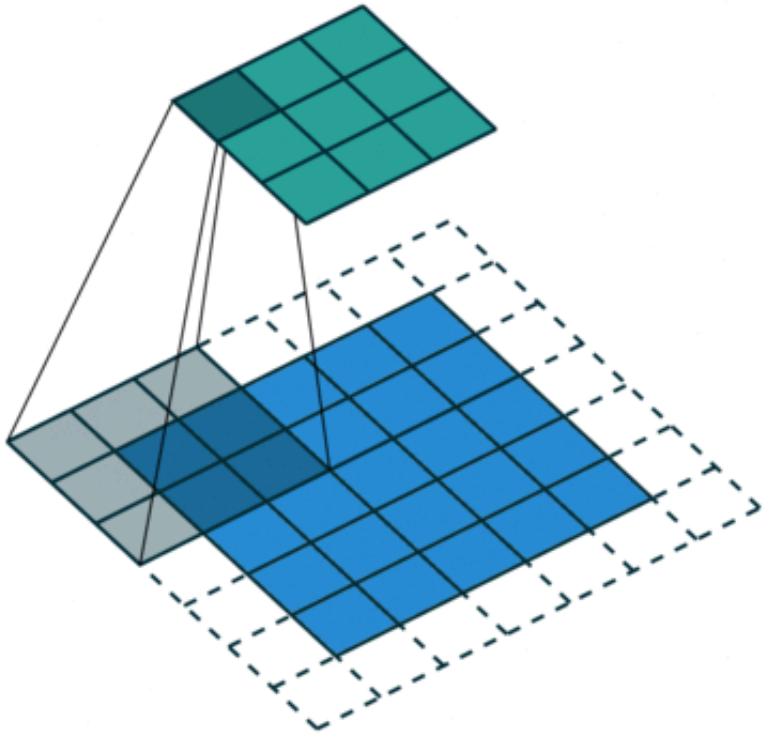


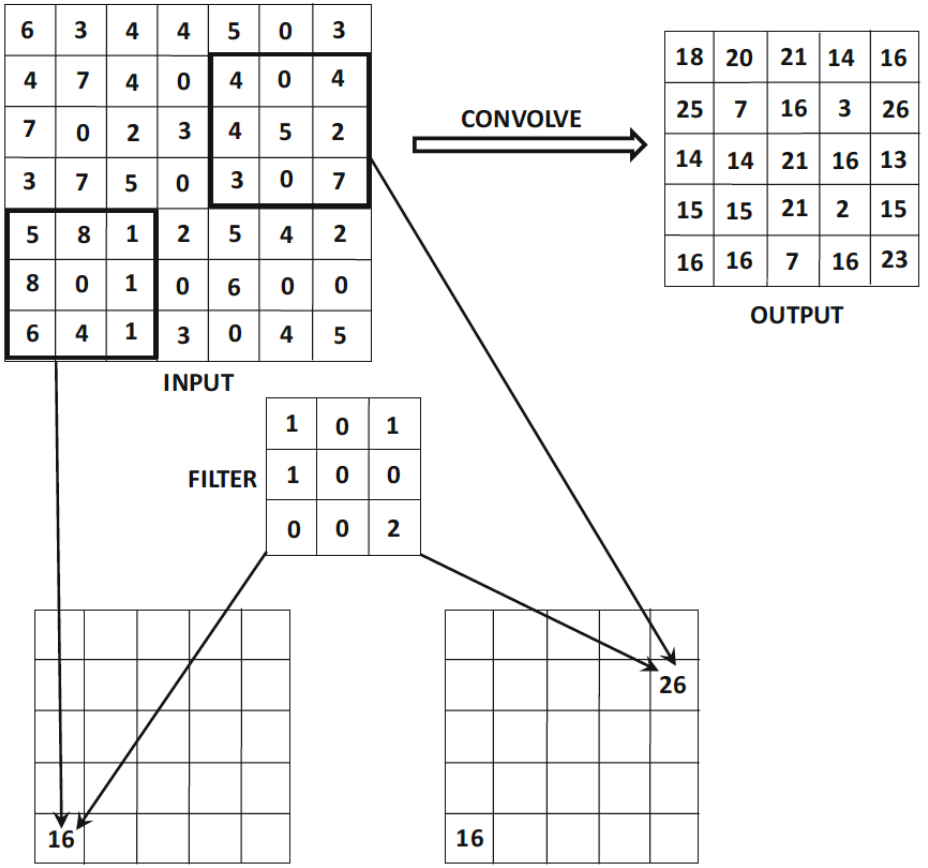
$$y = \sum_{i \in receptive field} w x_i + b$$

Convolutional Neural Net



Convolutional Neural Net

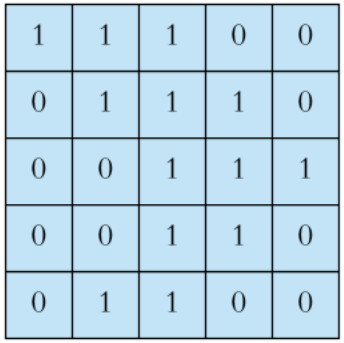




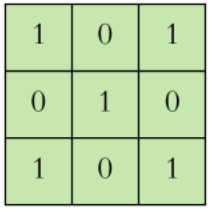
Convolutional Neural Net

https://tinyurl.com/y3wgholx

- Example:



Input

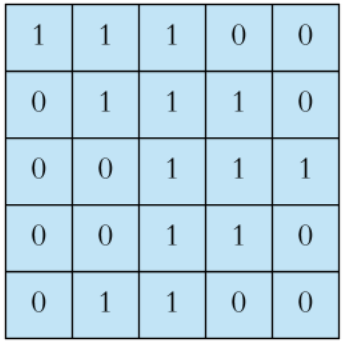


Filter / Kernel

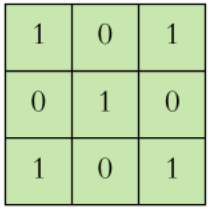
Convolutional Neural Net

https://tinyurl.com/y3wgholx

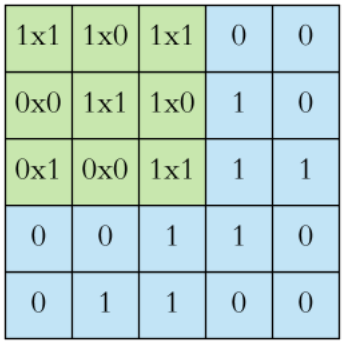
- Example:



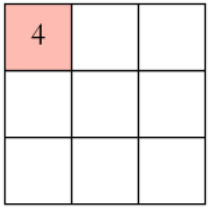
Input



Filter / Kernel



Input x Filter

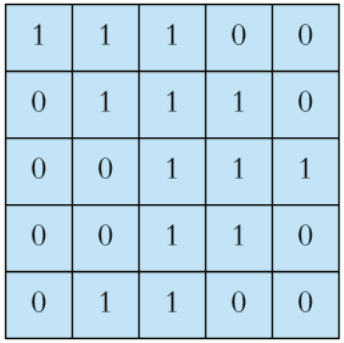


Feature Map

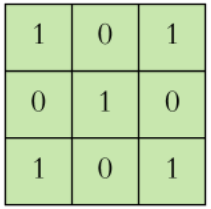
Convolutional Neural Net

https://tinyurl.com/y3wgholx

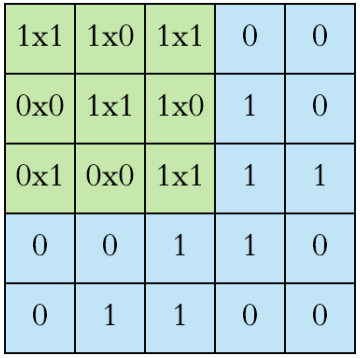
- Example:

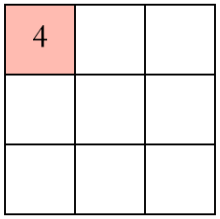


Input



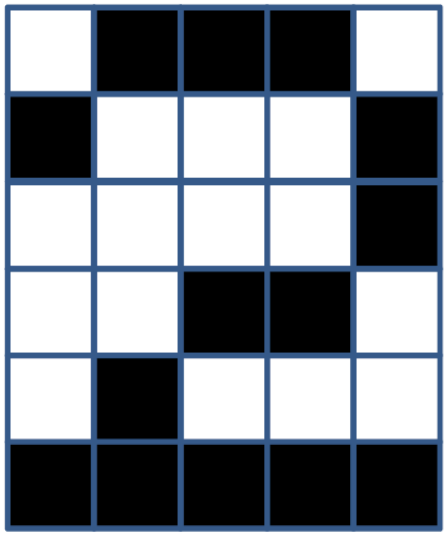
Filter / Kernel



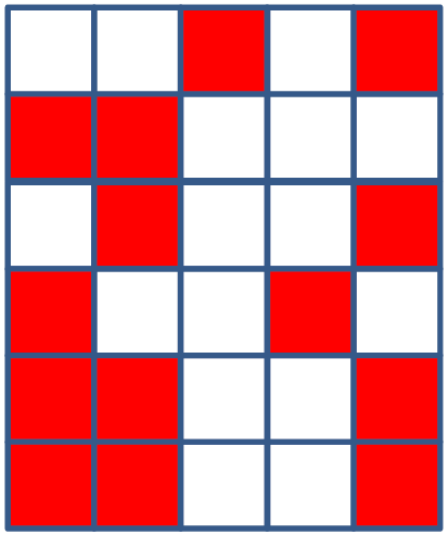


Another interpretation

W

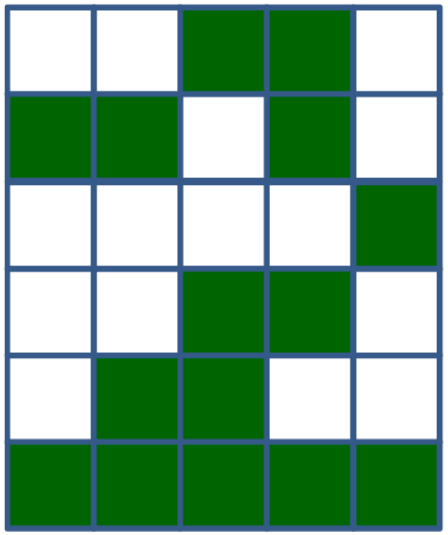


X



Correlation = 0.57

X

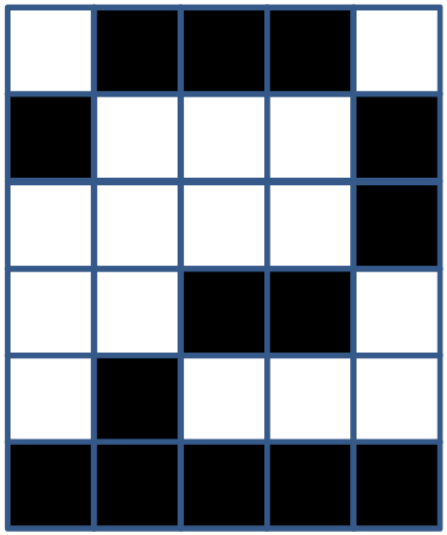


Correlation = 0.82

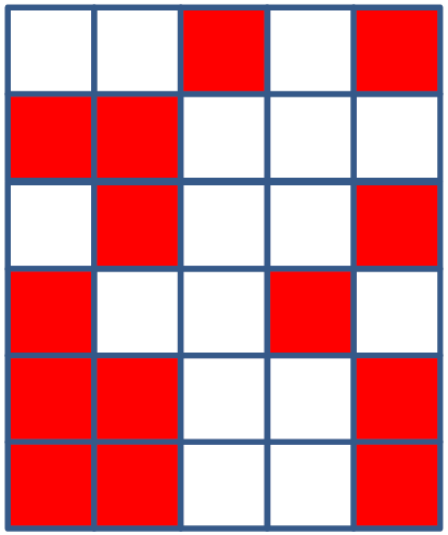
$$y = \begin{cases} 1 \; if \sum_i w_i x_i \geq T \\ 0 \; else \end{cases}$$

Another interpretation

W

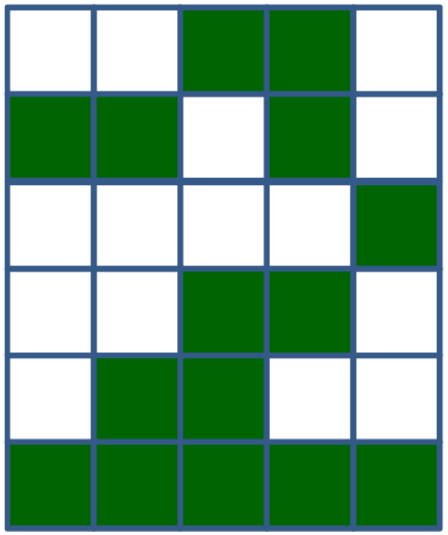


X



Correlation = 0.57

X



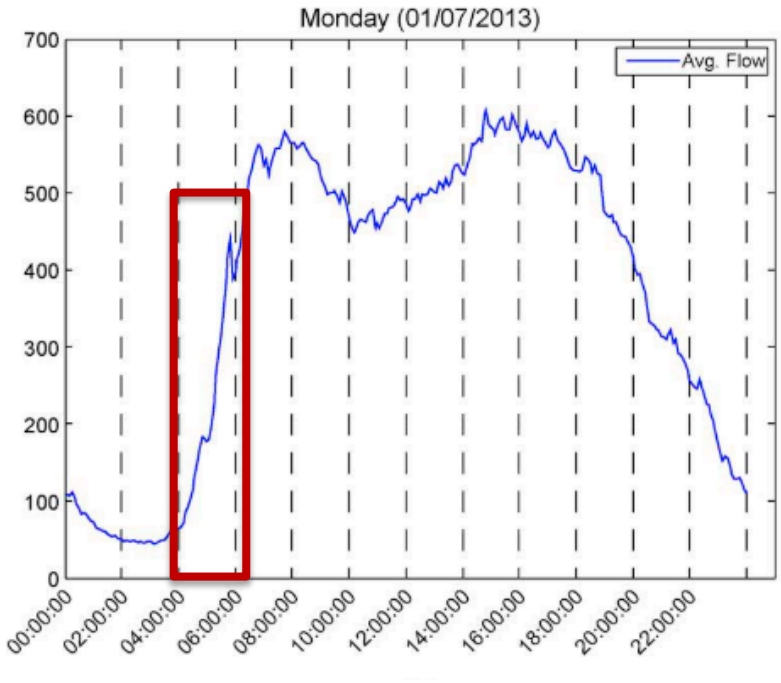
Correlation = 0.82

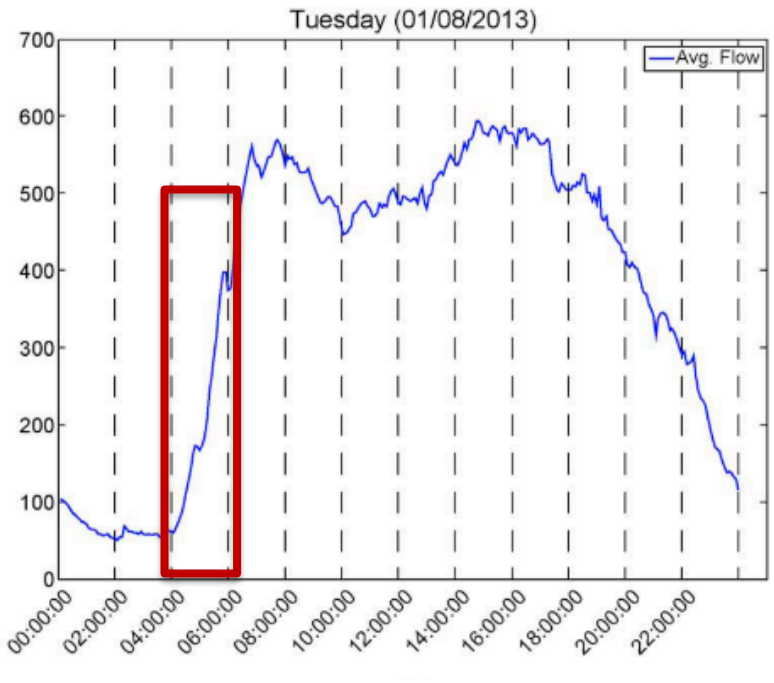
$$y = \begin{cases} 1 \; if \sum_i w_i x_i \geq T \\ 0 \; else \end{cases}$$

The green pattern is more correlated with the weights

CNN

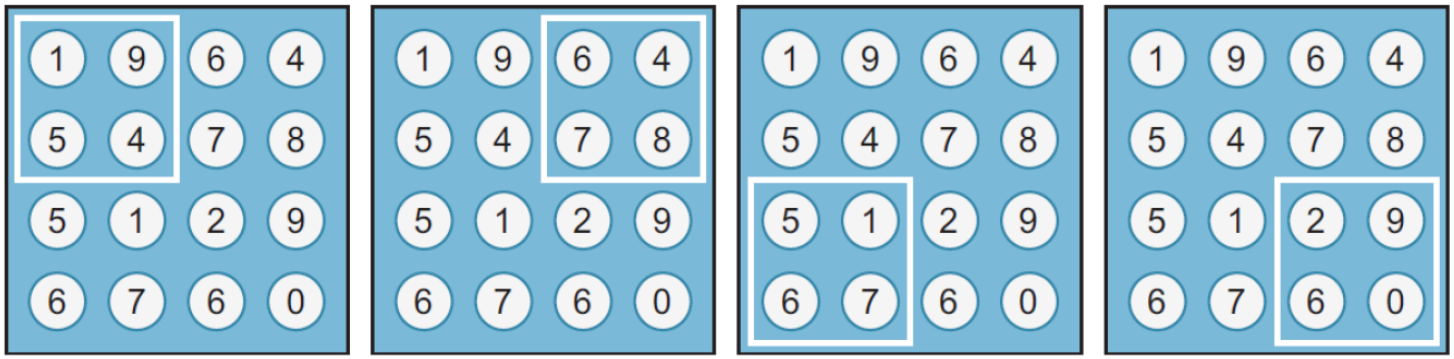
- What it inputs differ only slightly?
  - How do we account for jitter?





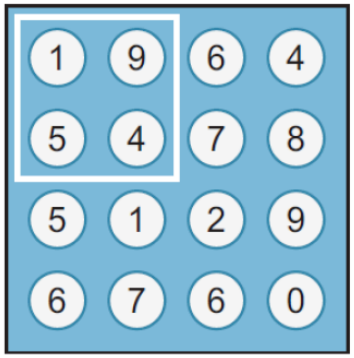
CNN

- What it inputs differ only slightly?
  - How do we account for jitter?
- Pooling!



CNN

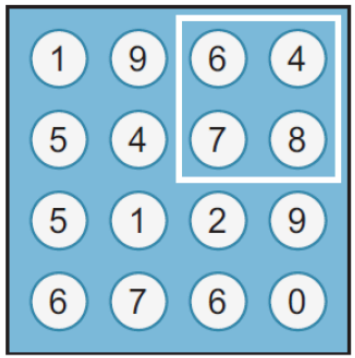
- What it inputs differ only slightly?
  - How do we account for jitter?
- Pooling!



CNN

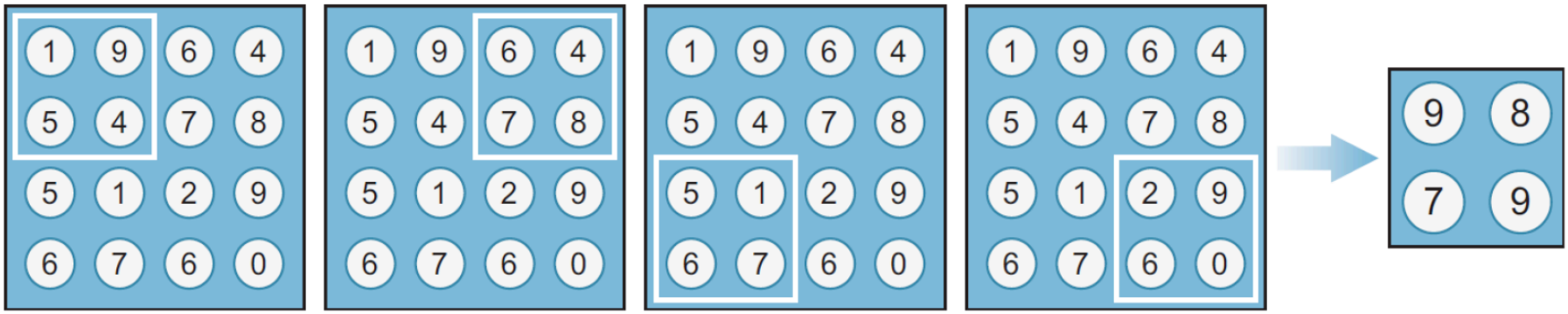
- What it inputs differ only slightly?
  - How do we account for jitter?

- Pooling!



CNN

- What it inputs differ only slightly?
  - How do we account for jitter?
- Pooling!



A typical CNN architecture

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Review

Enhancing transportation systems via deep learning: A survey

Yuan Wang[a], Dongxiang Zhang[b,*], Ying Liu[b], Bo Dai[b], Loo Hay Lee[a]

[a] Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore
[b] School of Computer Science and Engineering, University of Electronic Science and Technology of China, China

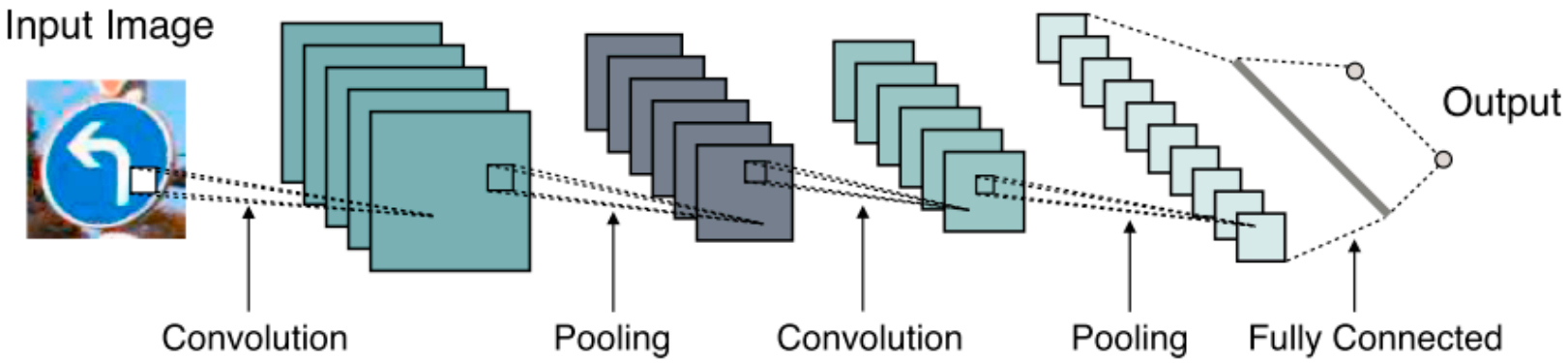


Fig. 3. Model architecture of convolutional neural network.

Applications based on “true” image data

- Need to estimate passenger flow
- Traditionally, automatic passenger counting can be done by contact-type counters, optical sensors, and vision-based systems.

- Contact-type:
  - Time consuming, problematic when demand is high
- Optical sensors:
  - Typically undercount

- How about vision-based models?

Applications based on “true” image data

- Camera feed from buses

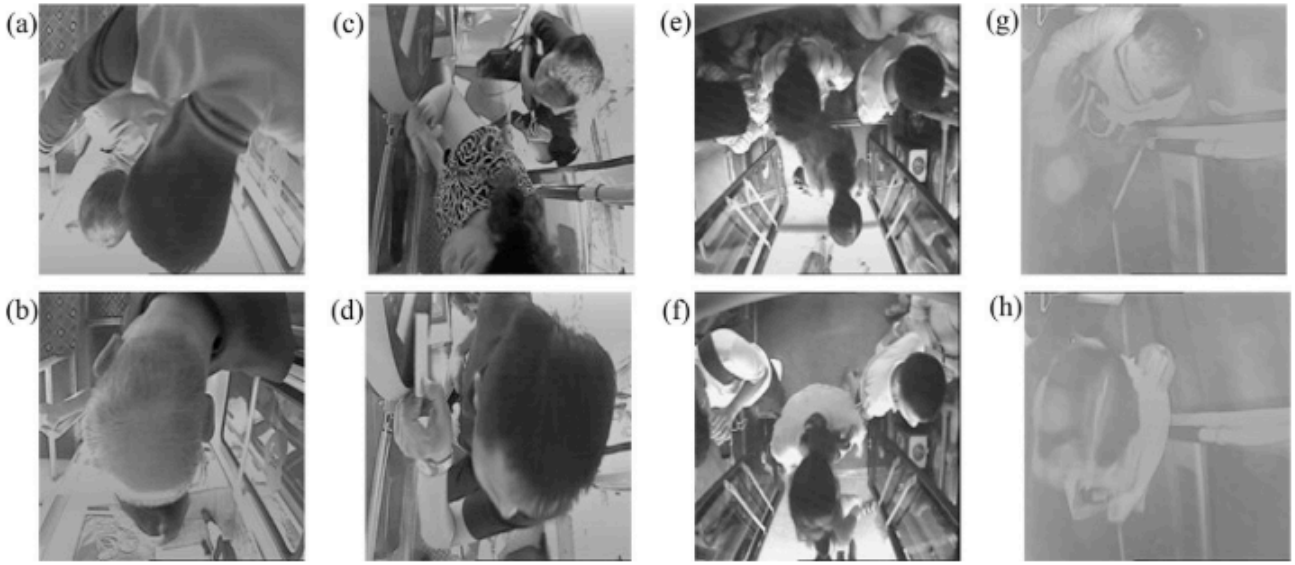
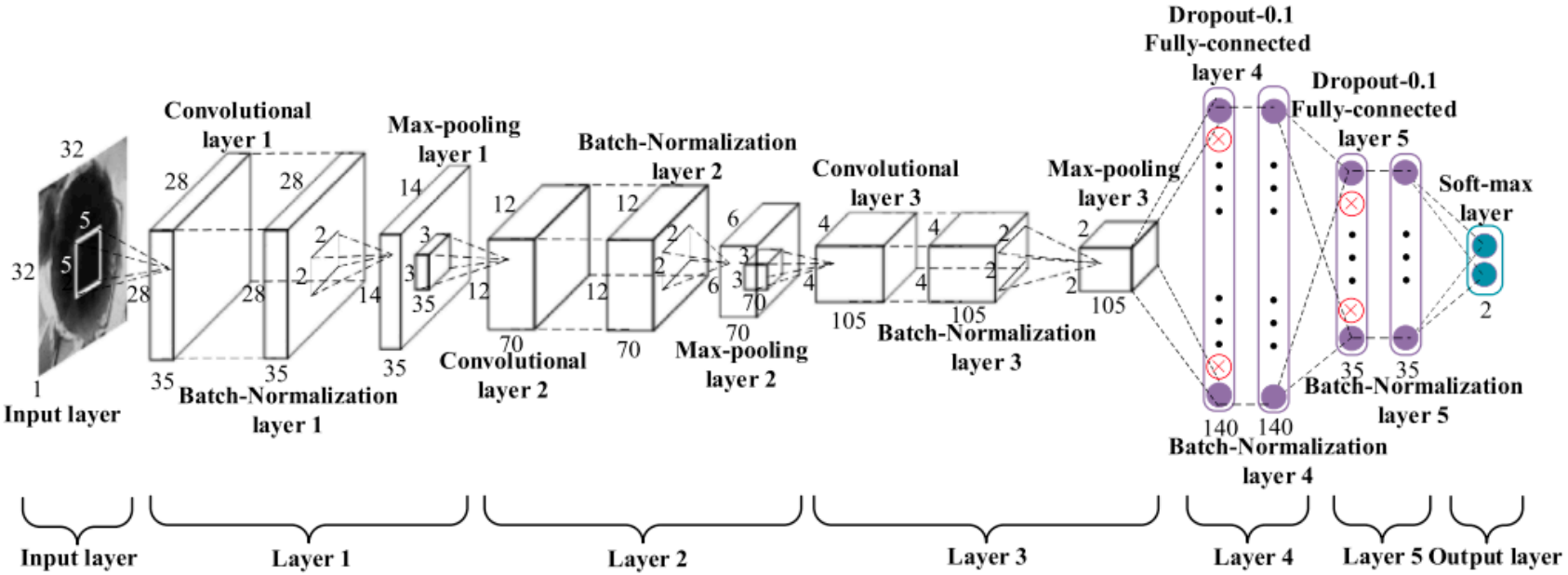


Fig. 1. Examples of the public transportation scenarios.

"Passenger flow estimation based on convolutional neural network in public transportation system."
Knowledge-Based Systems 123 (2017)

Applications based on “true” image data

- Camera feed from buses



"Passenger flow estimation based on convolutional neural network in public transportation system."
Knowledge-Based Systems 123 (2017)

Applications based on “true” image data

- Camera feed from buses

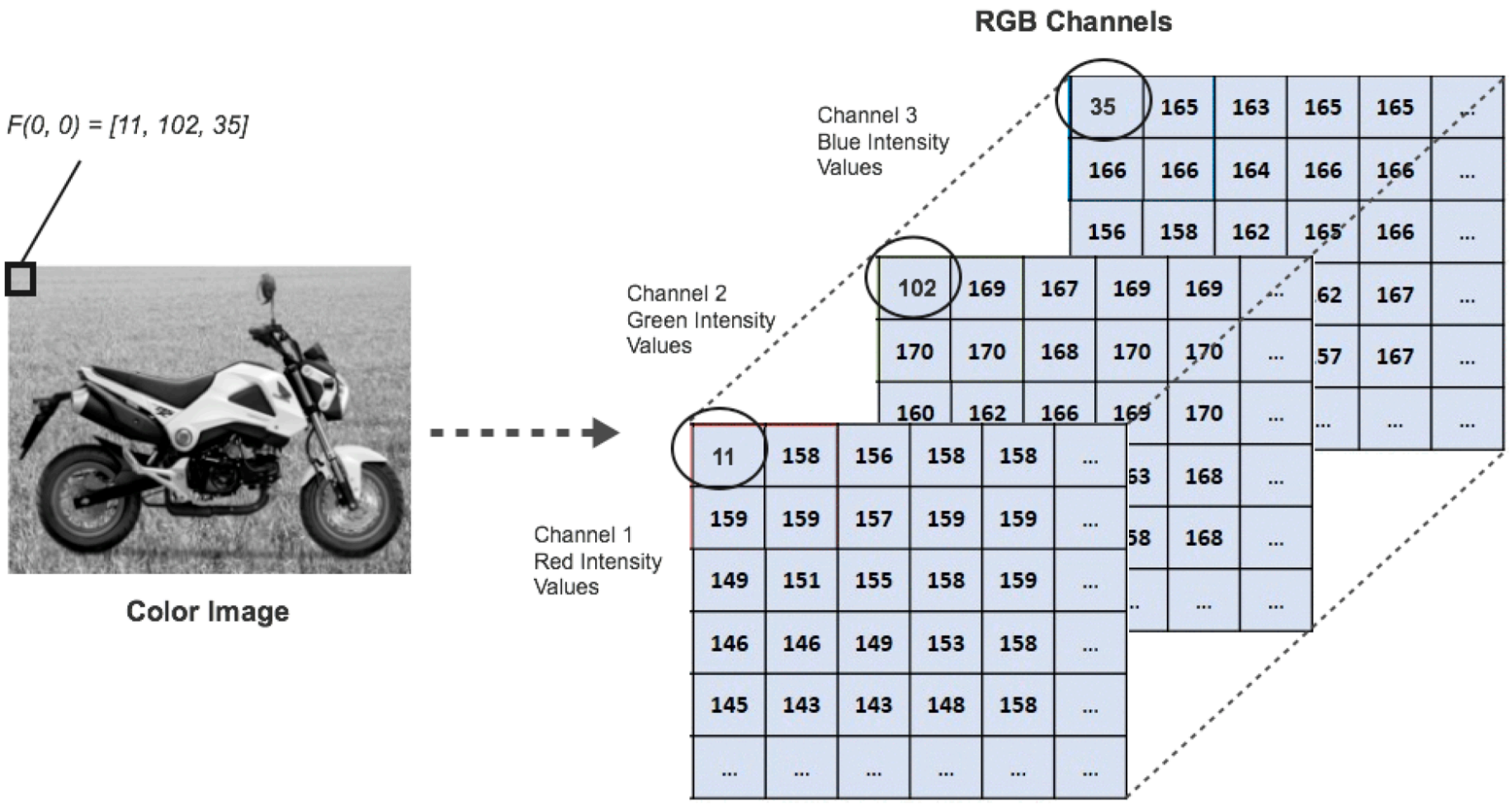
"Passenger flow estimation based on convolutional neural network in public transportation system."
Knowledge-Based Systems 123 (2017)

Let’s clarify what an image is

28 x 28
= 784 pixels

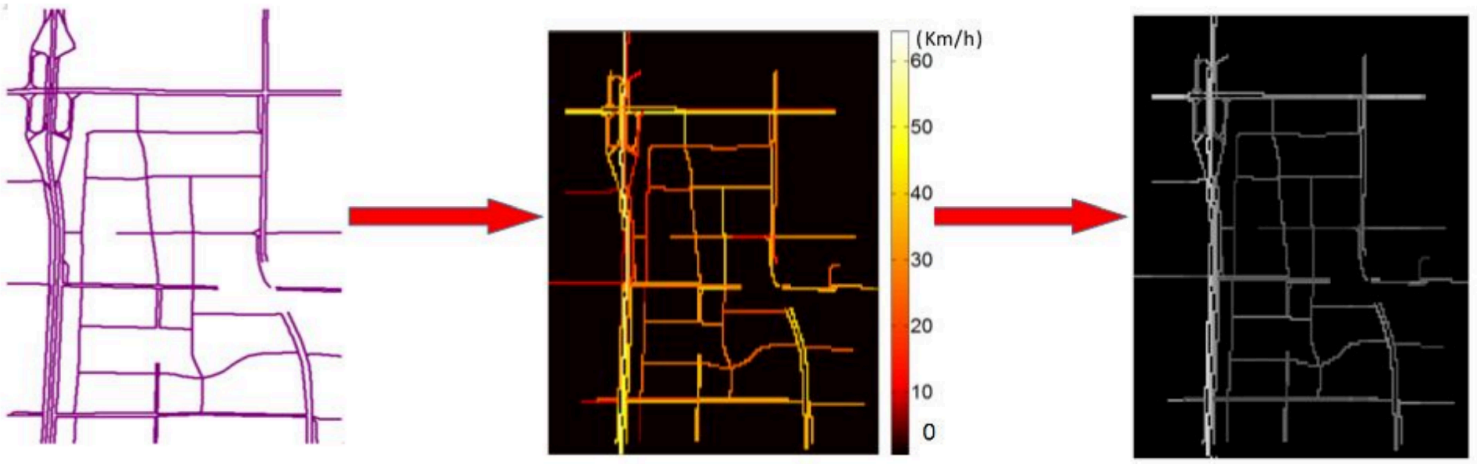
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	87	157	156	187	215	81	0	0	0	0	0
0	0	0	0	0	0	0	0	5	25	54	25	96	140	215	215	251	254	254	254	254	254	163	0	0	0	0	0
0	0	0	0	0	0	0	0	157	254	254	254	0	254	254	254	254	254	254	254	254	241	100	0	0	0	0	0
0	0	0	0	0	0	0	0	156	117	184	214	214	156	117	49	19	19	141	254	241	158	23	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	173	254	246	70	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	247	254	219	58	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	43	203	254	250	121	15	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	43	145	254	254	229	111	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	4	137	245	254	254	238	40	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	130	254	254	254	254	254	235	106	23	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	55	196	196	114	194	223	254	254	216	23	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	75	245	253	55	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	150	254	134	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	200	254	134	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	246	253	59	0	0	0	0	0	0	0	0	0
0	0	0	17	11	18	0	0	0	0	0	0	0	0	67	254	254	170	0	0	0	0	0	0	0	0	0	0
0	0	0	128	254	159	0	0	0	0	0	0	20	140	248	254	254	9	0	0	0	0	0	0	0	0	0	0
0	0	0	154	254	229	80	79	79	161	176	218	254	254	254	104	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	29	203	254	254	254	215	254	254	254	250	95	11	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	80	155	155	156	111	58	58	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Let's clarify what an image is



Other applications of CNN

- Traffic Prediction in Transportation Networks



Network

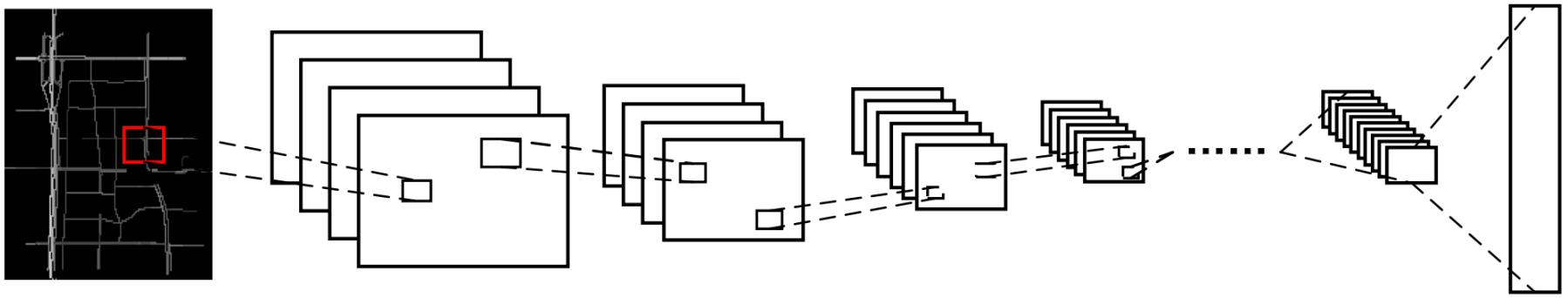
Speeds

Normalized Speeds

"Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks."
Sensors 17, no. 7 (2017)

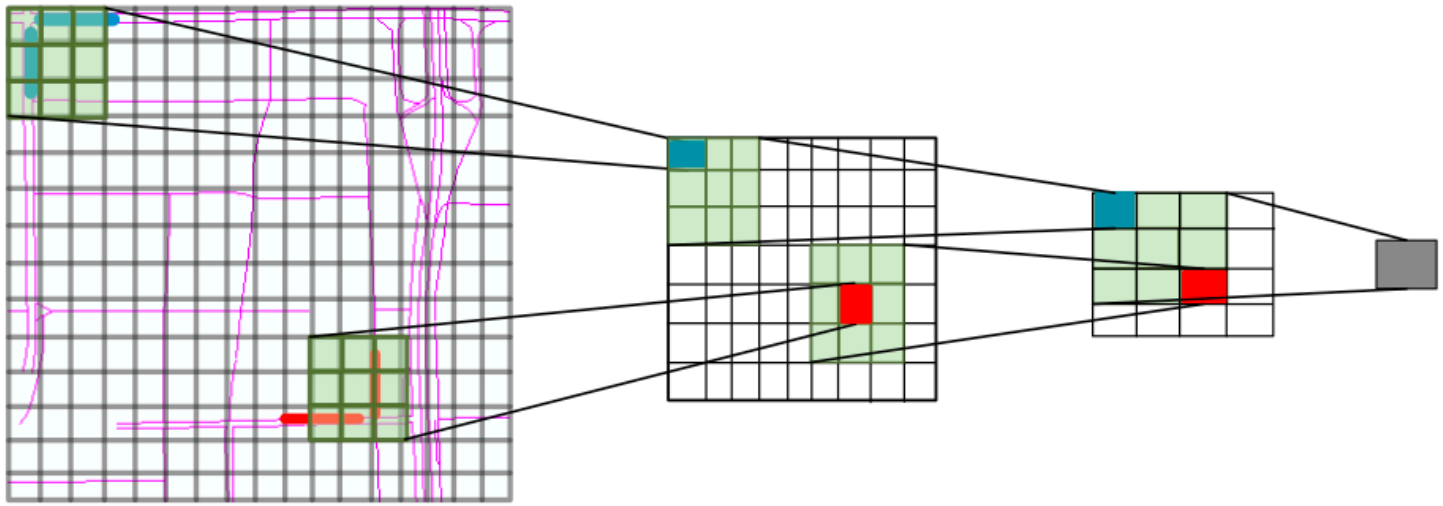
Other applications of CNN

- Traffic Prediction in Transportation Networks



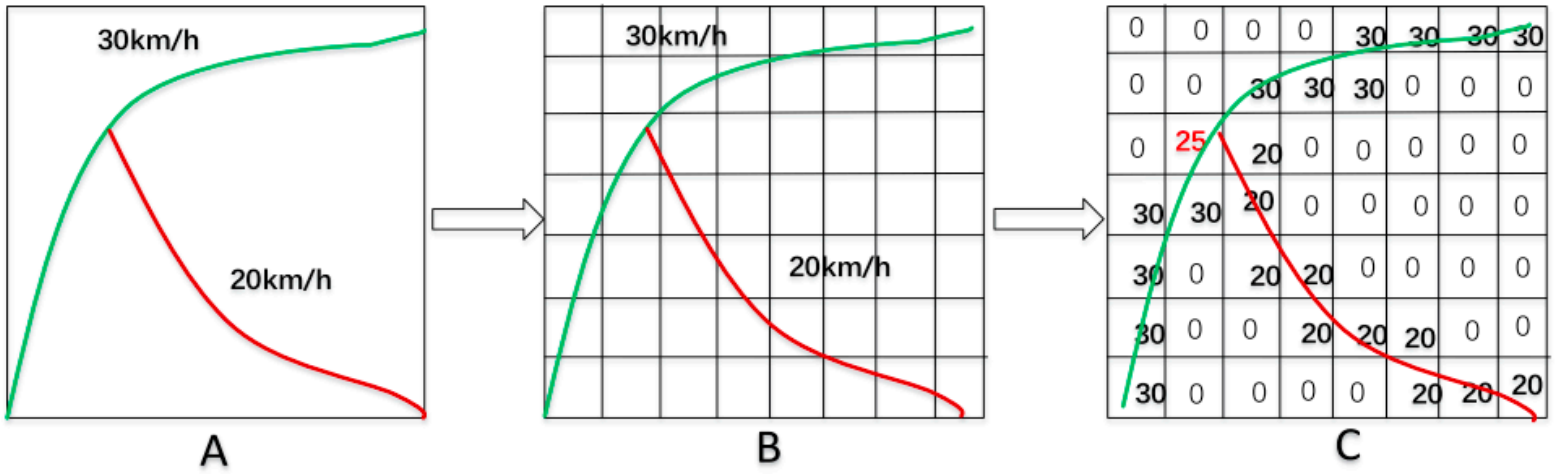
Other applications of CNN

- Let's look a little bit deeper

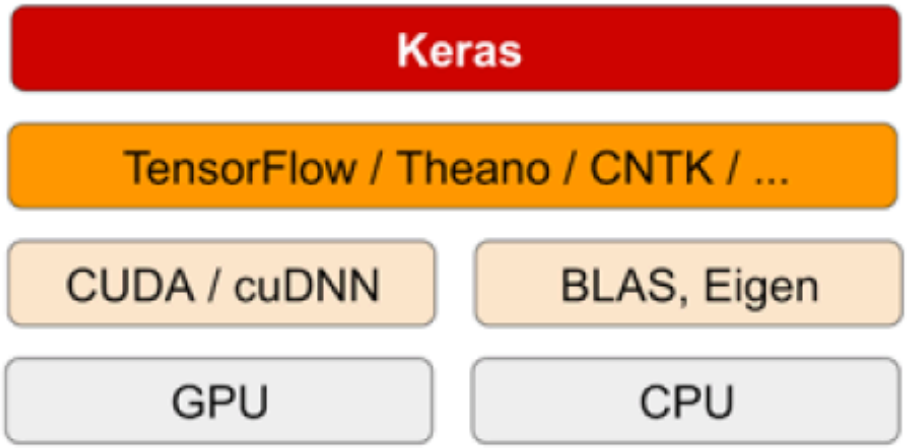


Other applications of CNN

- Let's look a little bit deeper



Lab 3



References

Deep Learning: Practice and Trends
Reed, de Freitas, Vinyals

CMU Deep learning course 2019
Bhiksha Raj