POLITECNICO MILANO 1863

# Traffic Prediction

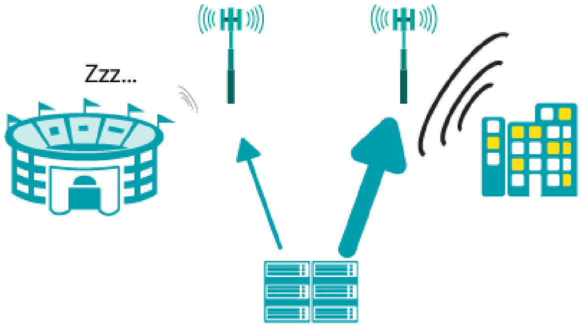
Francesco Musumeci

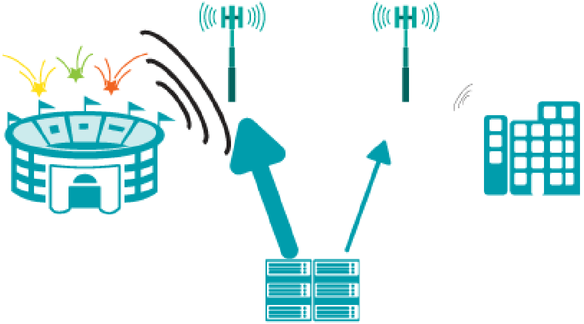
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)

Politecnico di Milano, Milano, Italy

# Traffic prediction
## Motivation

- Traffic carried by metropolitan/access networks (especially for mobile network backhauling/aggregation) is highly dynamic in space and time
  - Business vs. residential areas
  - Some areas with extremely-high peak rates only in specific days/hours (e.g., a stadium, theaters, university campus, etc.)



- Network (traffic transport) and computing (traffic processing) resources can be allocated (e.g., power on/off, scale no. of CPUs used, etc.) according to various strategies:
  - **Static**, based on peak-traffic: highly over-dimensioned
  - **Static**, but lower wrt peak: need to accept blocking some users (reduced service quality)
  - **Dynamically reconfigured**, based on (accurate) traffic estimation [1]


[1] 5G PPP Technology Board, "AI and ML – Enablers for Beyond 5G Networks", 2021, http://doi.org/10.5281/zenodo.4299895

# Traffic prediction

## Resources allocation based on traffic prediction: Example

- Aggregated traffic from multiple mobile base stations fluctuates between 100 Mbit/s and 100 Gbit/s and with average traffic 10 Gbit/s in a typical day
- Antenna segments can be turned on/off with granularity of 1 Gbit/s (assume total energy consumption is ***E KWh per Gbit/s***)*
- Energy consumption and service acceptance ratio for 1 day and for various resources allocation strategies:

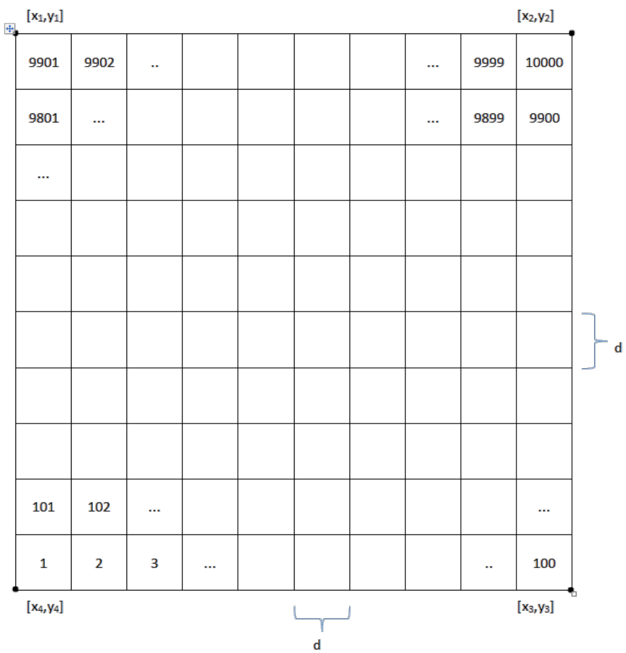
| Resources allocation based on… | Energy consumed for 1 day | Service acceptance ratio |
|---|---|---|
| Peak traffic | $E_{peak} = 24 * 100 * E$ | ≅ 100% |
| Average traffic | $E_{avg} = 24 * 10 * E$ | << 100% |
| Traffic prediction | $E_{avg} \leq E_{pred} << E_{avg}$ | ≅ 100% (?) |

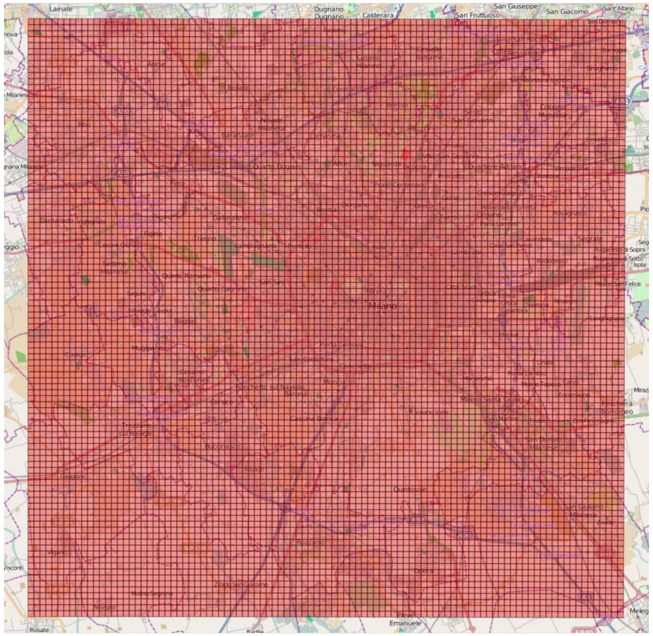
- Key challenges
  - How to predict traffic?
  - How much in advance? Minutes/hours/days?
  - What information is useful?

* https://www.huawei.com/en/media-center/transform/04/solving-network-mission-impossible?utm_source=DSMN8&utm_medium=LinkedIn

# Traffic prediction
## Dataset – Milano GRID

- Milano GRID represents the GPS coordinates of the squared cells superimposed to the Milan map [2]
- In total we have 10k squared cells (230m x 230m each), containing traffic information about mobile network



[2] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/QJWLFU

# Traffic prediction

## Dataset: CDR = SMS + calls + Internet activity

- Traffic information is expressed as Call Detail Records (CDRs) generated by the Telecom Italia cellular network for Milano
- CDRs log the user activity for billing purposes and network management. 5 types of CDRs are distinguished
  1. Received SMS
  2. Sent SMS
  3. Incoming Calls
  4. Outgoing Calls
  5. Internet
- We are not interested in the precise values of CDRs, but rather in their relative behaviour
- **Spatial aggregation**: different activity measurements are provided for each **square** of the Milano GRID (not per-base station)
- **Temporal aggregation**: activity measurements are obtained by temporally aggregating CDRs in timeslots of 10 minutes

# Traffic prediction
## Dataset used in the lab

- Data records have been collected for 61 days, (1st Nov. – 31st Dec. 2013)
  - 61 files, each recording traffic for a single day for all the cells
- We use data from 60 cells only to limit complexity during the lab (full dataset consists of 10k cells)
  - Cell IDs [1-20] + [4991-5010] + [9981-10000]
- Our dataset contains following fields:
  - **Cell ID**
  - **Time Stamp**: raw timestamp in milliseconds units with interval of 10 minutes
  - **Inbound/outbound SMS**: incoming/outgoing SMS in/from a cell within 10 minutes interval
  - **Inbound/outbound Call Activity**: the incoming/outgoing calls in/from a cell within 10 minutes interval
  - **Internet Activity**: the internet usage by mobile customers in a cell within 10 minutes interval

[2] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/QJWLFU

# Traffic prediction
## Dataset used in the lab

Time Stamp
Inbound/outbound Call Activity
Cell ID
Inbound/outbound SMS
Internet Activity

```
1 1383260400000 0.223226878214 0.156787005039 0.160937936917 0.0522748485286 11.0283663817
1 1383261000000 0.415039900219 0.119925720142 0.188777172915 0.160937936917 11.1271008757
1 1383261600000 0.384079316758 0.170952029685 0.13417624316 0.0546009297544 10.8927706028
1 1383262200000 0.707571503953 0.220815298616 0.0273004648772 0.0534378891415 8.62242459099
1 1383262800000 0.270678567546 0.192890564246 0.0534378891415 0.0807383540187 8.00992746245
1 1383263400000 0.0836887047371 0.243378102669 0.0273004648772 0.0273004648772 8.11841955409
1 1383264000000 0.165051328248 0.0849372437223 0.0534378891415 0.0053619303165 8.02626974851
1 1383264600000 0.216077637792 0.0261374242643 0.0017873101055 0.0546009297544 8.51417857718
1 1383265200000 0.242215062056 0.160313667424 0.108038818896 0.0261374242643 6.83342489638
1 1383265800000 0.321790375462 0.245704183895 0.0273004648772 0.108038818896 6.55460504455
1 1383266400000 0.13417624316 0.108038818896 0.0 0.0 7.33871601282
```

[2] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/QJWLFU

# Traffic prediction

## Calendar of data collection… will be useful for the coding tasks… ☺

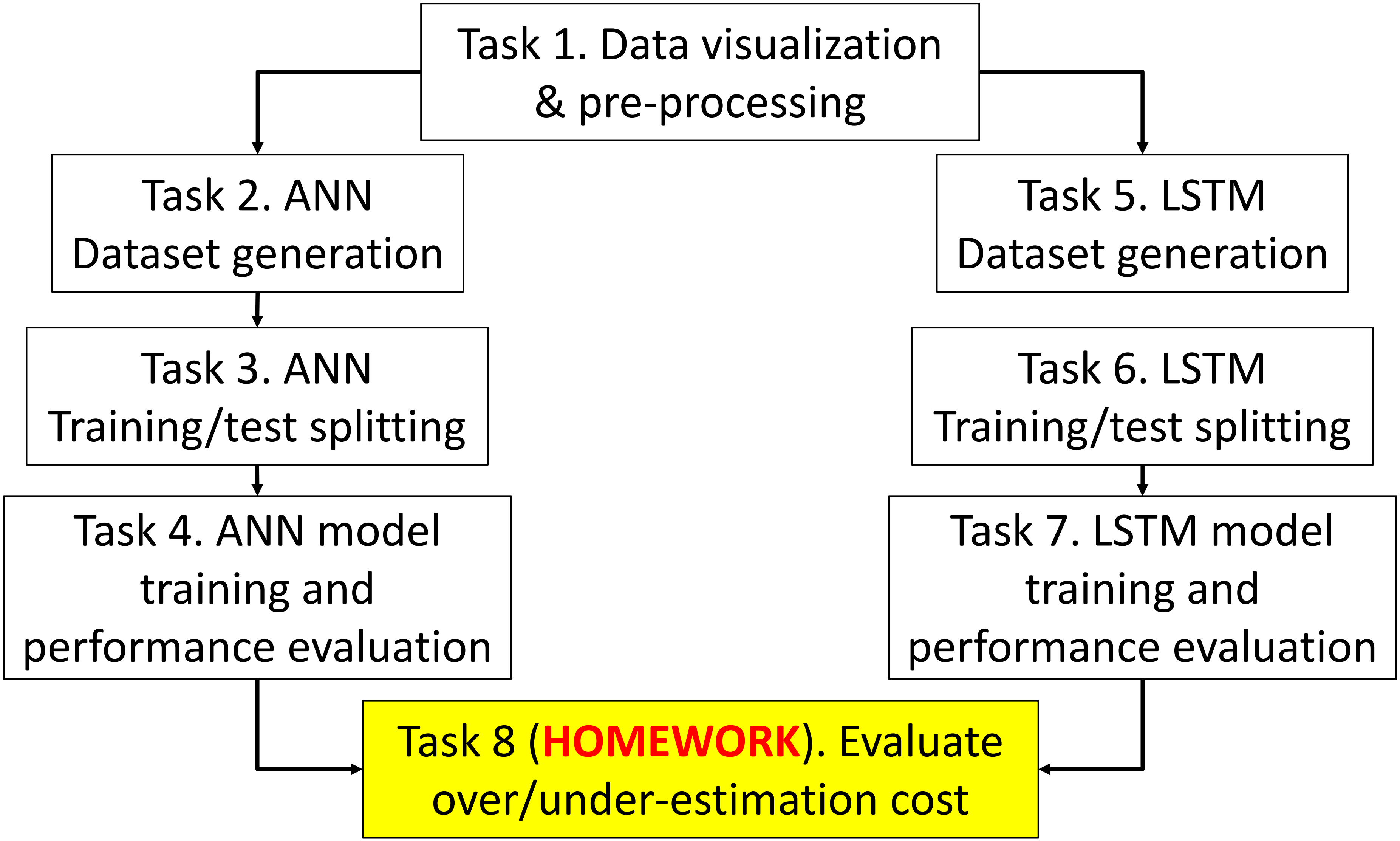
novembre 2013

| lu | ma | me | gi | ve | sa | do |
|---|---|---|---|---|---|---|
| 28 | 29 | 30 | 31 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 1 |

dicembre 2013

| lu | ma | me | gi | ve | sa | do |
|---|---|---|---|---|---|---|
| 25 | 26 | 27 | 28 | 29 | 30 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| 30 | 31 | 1 | 2 | 3 | 4 | 5 |

# Traffic prediction
## Lab overview



# Traffic prediction
## Task 1

1. Dataset pre-processing:
   a) Define function *load_dataset()* that takes in input cell ID, start day, number of days and type of traffic to be considered, and returns a pandas dataframe with data retrieved from proper source files
      - See details in the skeleton code
      - **N.B.1: traffic data should be aggregated per 1-hour period (now they are collected every 10 mins)**
      - **N.B.2: Indexes of days must correspond to the selected days (e.g., indexes 36-40 if selecting days Dec. 6th - 10th)**

   b) Call function *load_dataset()* for week Dec 2nd (Monday) - Dec 8th (Sunday) for cell ID = 3 considering Internet traffic, then display dataframe object in tabular form
      - **Already given in skeleton code**

# Traffic prediction

## Task 1a)-b): expected outputs

```
cell = 3
start_d = 32
num_d = 7
traffic_t = 5

dataframe = load_dataset(cell, start_d, num_d, traffic_t)

dataframe
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 14 | 15 | 16 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **32** | 55.560594 | 49.568379 | 39.700737 | 38.161090 | 36.137046 | 38.432246 | 46.179424 | 57.859469 | 52.696863 | 45.859077 | ... | 57.721614 | 59.159070 | 66.118096 | 67 |
| **33** | 51.270115 | 49.856398 | 45.644236 | 42.199057 | 36.384711 | 33.515705 | 47.304401 | 58.780178 | 52.744205 | 61.983305 | ... | 60.033429 | 62.184259 | 68.179293 | 70 |
| **34** | 61.046252 | 47.833819 | 47.067196 | 36.016166 | 31.364266 | 36.354097 | 47.467844 | 56.528149 | 53.667258 | 52.654684 | ... | 60.338401 | 63.774202 | 67.964699 | 74 |
| **35** | 60.846063 | 51.661429 | 40.679023 | 38.758770 | 33.022719 | 38.710436 | 49.928087 | 56.918719 | 52.866572 | 56.532455 | ... | 63.241110 | 71.288841 | 72.140565 | 71 |
| **36** | 53.486007 | 49.352207 | 40.949818 | 36.198185 | 33.945772 | 36.223636 | 46.311147 | 61.111922 | 59.704471 | 55.145789 | ... | 59.955586 | 67.526787 | 79.886751 | 76 |
| **37** | 58.997251 | 48.776801 | 43.377959 | 38.467685 | 37.563569 | 53.405300 | 39.549852 | 50.374114 | 64.941606 | 81.121539 | ... | 78.838760 | 70.754261 | 66.607295 | 73 |
| **38** | 68.333627 | 60.776944 | 52.663792 | 48.541061 | 42.859002 | 36.009043 | 41.799559 | 48.598480 | 57.296259 | 70.010881 | ... | 96.932566 | 95.622392 | 90.354642 | 88 |

7 rows × 24 columns

# Traffic prediction
## Task 1

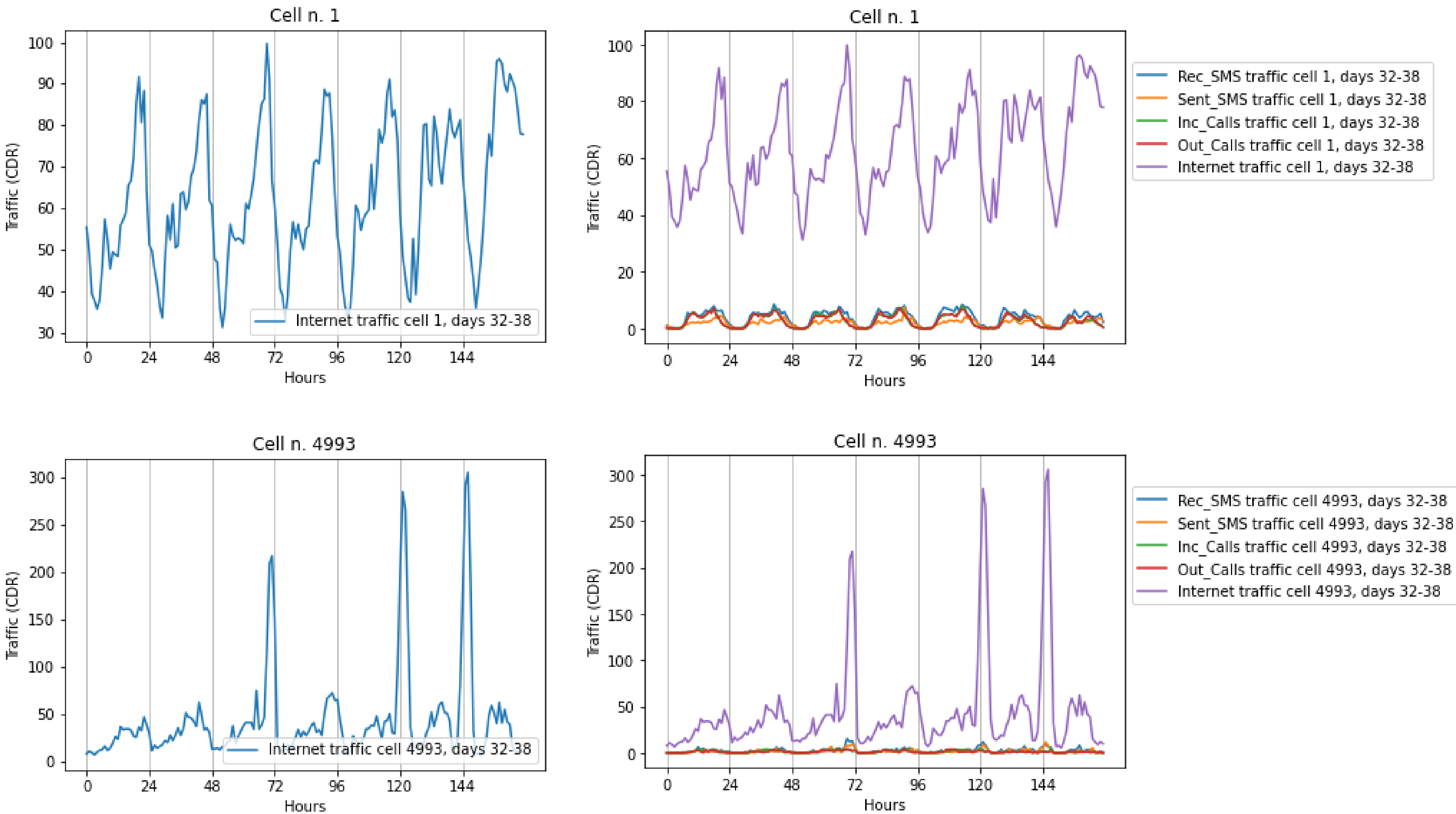
1. Dataset pre-processing:
   c) Define function *plot_trace()* that takes in input cell ID, start day, number of days and type of data traffic to be considered, calls function *load_dataset(),* plots and saves a figure with traffic trace vs time
      - **Already given in skeleton code**

   d) Call function *plot_trace()* for week Dec 2nd (Monday) - Dec 8th (Sunday) for cell IDs = 1, 4993, 9990, considering Internet traffic and plotting/saving also figures with all traffic types
      - **Already given in skeleton code**

   **What can we observe?**

# Traffic prediction

## Task 1c)-d): expected outputs



# Traffic prediction
## Task 2

2. Artificial Neural Networks (ANN) dataset generation
   a) Define function *generatedataset()* that takes in input a pandas dataframe and returns features matrix X and output y as numpy ndarrays. Features and output values in X and y should be normalized so as to be in [0,1] range
      - See details in the skeleton code
      - **N.B. Each feature in X should be normalized independently from other features. Same for the output y**
      - **See next slide for a visual representation of dataset (X,y)**
   b) Call function *load_dataset()* for the entire period (days 1 to 61) cell ID 5, and Internet traffic only, then call function *generatedataset()* and print min and max values for each feature and for both normalized and raw datasets
      - **Already given in skeleton code**

*Features are inspired by:
R. Alvizu, S. Troia, G. Maier and A. Pattavina, "Matheuristic with machine-learning-based prediction for software-defined mobile metro-core networks", in IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 9, no. 9, pp. D19-D30, Sept. 2017.

# Traffic prediction

## Task 2 – how does the dataset look like?

| | X | | | | | y |
|---|---|---|---|---|---|---|
| | **Feature1** | **Feature2** | **Feature3** | **Feature4** | **Feature5** | **Traffic** |
| | X[0,0] | X[0,1] | X[0,2] | X[0,3] | X[0,4] | y[0] |
| | X[1,0] | X[1,1] | X[1,2] | X[1,3] | X[1,4] | y[1] |
| 1st day | X[2,0] | X[2,1] | X[2,2] | X[2,3] | X[2,4] | y[2] |
| | … | … | … | … | … | … |
| | X[23,0] | X[23,1] | X[23,2] | X[23,3] | X[23,4] | y[23] |
| | X[24,0] | X[24,1] | X[24,2] | X[24,3] | X[24,4] | y[24] |
| | X[25,0] | X[25,1] | X[25,2] | X[25,3] | X[25,4] | y[25] |
| 2nd day | X[26,0] | X[26,1] | X[26,2] | X[26,3] | X[26,4] | y[26] |
| | … | … | … | … | … | … |
| | X[47,0] | X[47,1] | X[47,2] | X[47,3] | X[47,4] | y[47] |
| … | X[48,0] | X[48,1] | X[48,2] | X[48,3] | X[48,4] | y[48] |
| | … | … | … | … | … | … |

# Traffic prediction

## Task 2: expected outputs

```
    print('Feature {} has minimum raw value {}'.format(Xdata.columns[i], minvalue))
    print('Feature {} has Maximum raw value {}'.format(Xdata.columns[i], maxvalue))
    print('Feature {} has minimum normalized value {}'.format(Xdata.columns[i], minvaluenorm))
    print('Feature {} has Maximum normalized value {}'.format(Xdata.columns[i], maxvaluenorm))

```

```
Feature day_of_week has minimum raw value 0.0
Feature day_of_week has Maximum raw value 6.0
Feature day_of_week has minimum normalized value 0.0
Feature day_of_week has Maximum normalized value 1.0
Feature hour has minimum raw value 0.0
Feature hour has Maximum raw value 23.0
Feature hour has minimum normalized value 0.0
Feature hour has Maximum normalized value 1.0
Feature working_day has minimum raw value 0.0
Feature working_day has Maximum raw value 1.0
Feature working_day has minimum normalized value 0.0
Feature working_day has Maximum normalized value 1.0
Feature prev_week has minimum raw value 24.87095961303
Feature prev_week has Maximum raw value 152.19921217209998
Feature prev_week has minimum normalized value 0.0
Feature prev_week has Maximum normalized value 0.9999999999999998
Feature prev_day has minimum raw value 24.87095961303
Feature prev_day has Maximum raw value 152.19921217209998
Feature prev_day has minimum normalized value 0.0
Feature prev_day has Maximum normalized value 0.9999999999999998
```

# Traffic prediction
## Task 3

3. ANN Training/test splitting
    a) Define function *train_test_split()* that takes in input the dataset (X,y) and the desired amount of test data (number of hours/samples), and splits (X,y) into train/test sets ***with samples ordered chronologically***
    b) Call function *train_test_split()* for dataset (X,y) generated in task 2b), putting **the last 10 days** of samples in the test set. Verify dimensions of train, test and whole datasets
        - **Already given in skeleton code**

Task 3: expected output

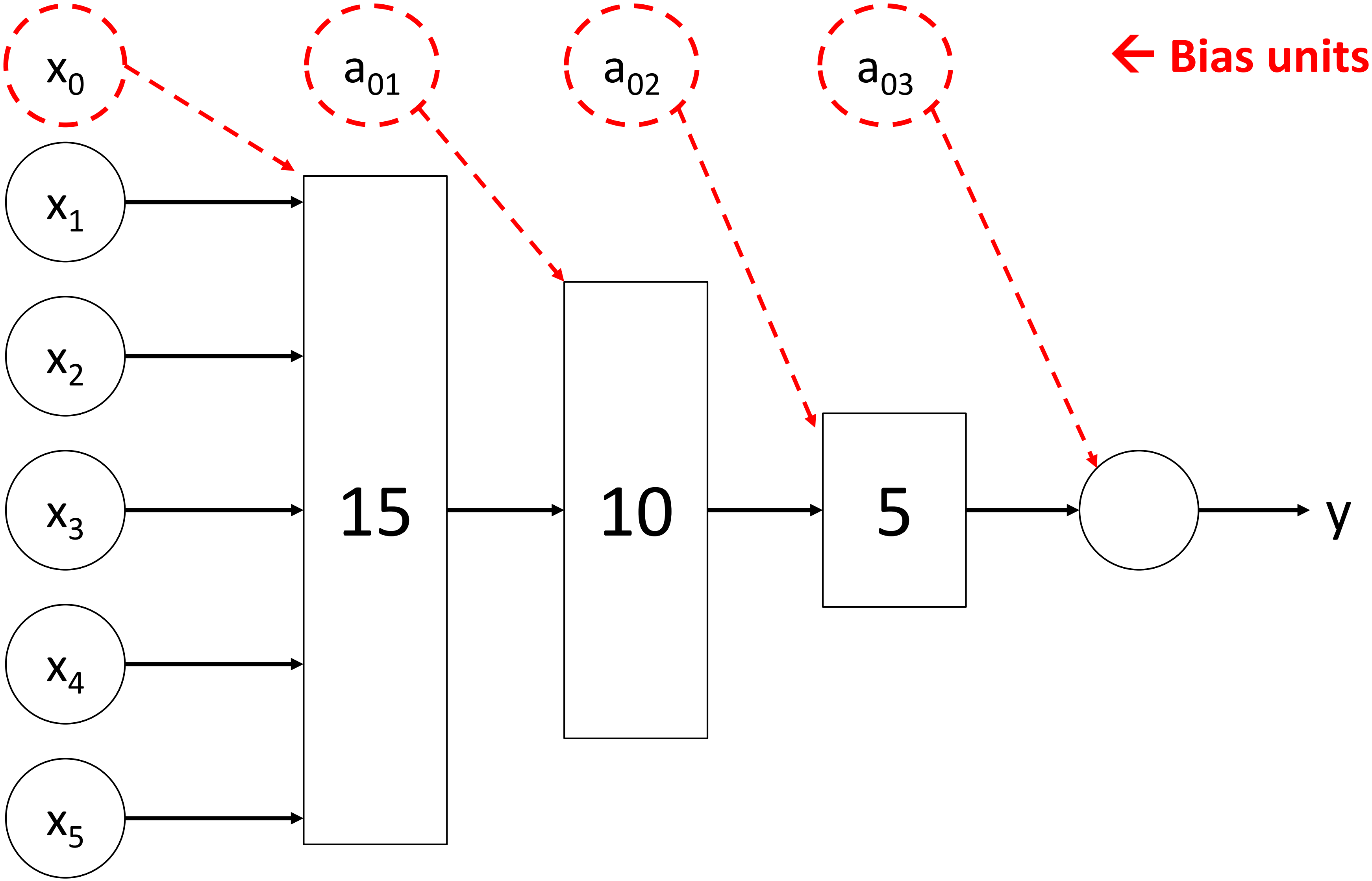
```
print(X.shape)
print(y.shape)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(1464, 5)
(1464, 1)
(1224, 5)
(1224, 1)
(240, 5)
(240, 1)
```

# Traffic prediction
## Task 4

4. ANN model training and performance evaluation
   a) Train an ANN with activation function = 'sigmoid', 3 hidden layers with [15, 10, 5] neurons, respectively. Print final TRAINING MSE and training duration, and save a figure with model structure. Figure with model structure should be saved as .png file in subfolder 'Results'.
      - **N.B. The ANN output layer is a single neuron that should output a real value (no activation function)**
      - **See next slide for a "visual" representation of model structure**

# ANN model structure



# Traffic prediction
## Task 4a): expected outputs

Model training duration [s]: 30.86
Training MSE: 0.0027662314080470598

Using command
*model_NN.summary()*

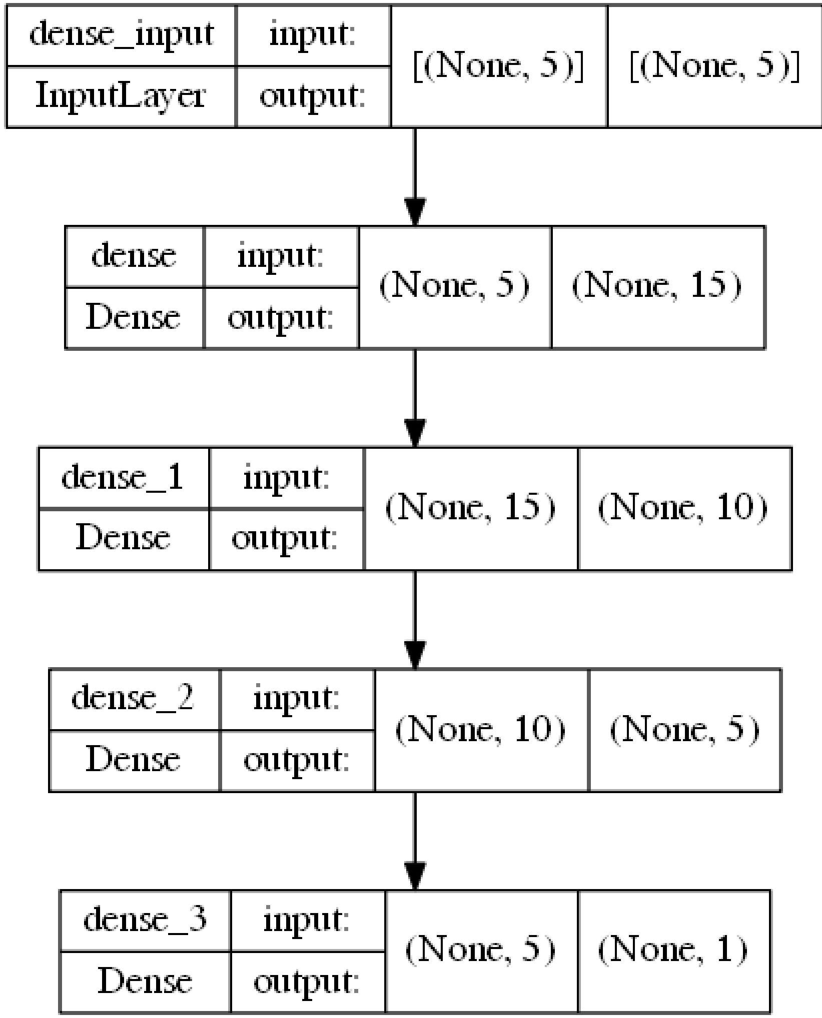
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 15)                90

 dense_1 (Dense)             (None, 10)                160

 dense_2 (Dense)             (None, 5)                 55

 dense_3 (Dense)             (None, 1)                 6

=================================================================
Total params: 311
Trainable params: 311
Non-trainable params: 0
_________________________________________________________________
```

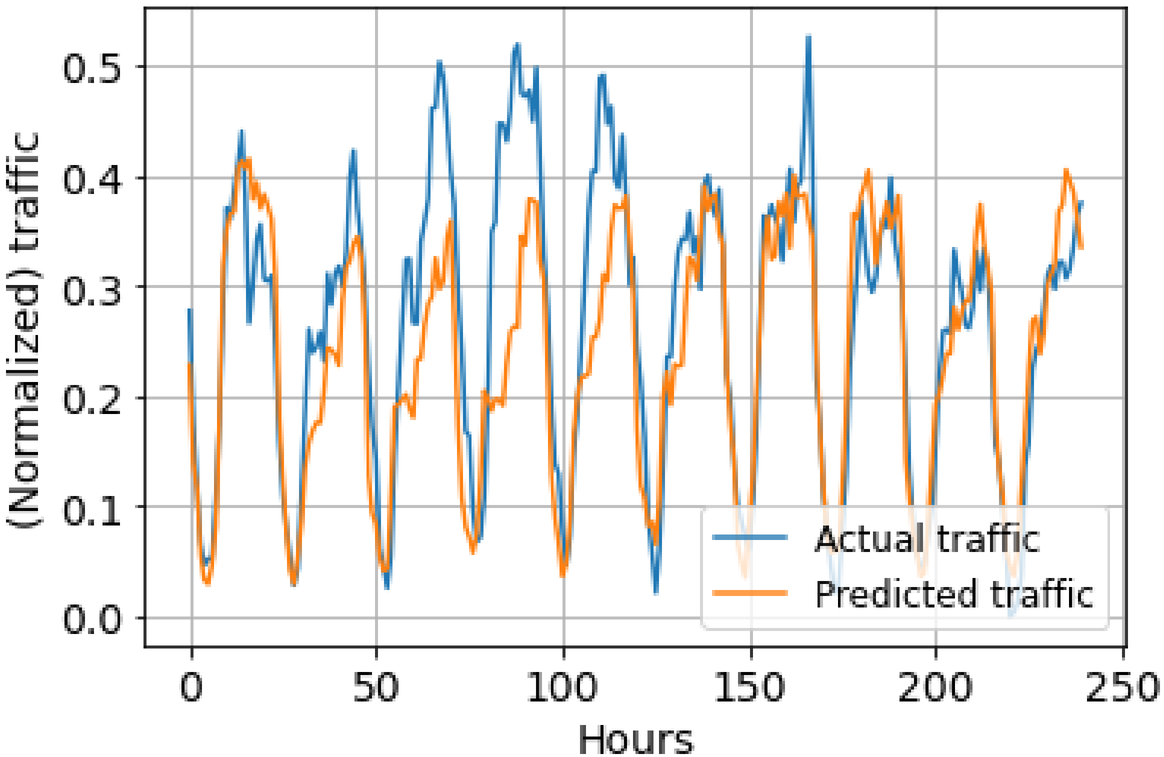


# Traffic prediction
## Task 4

4. ANN model training and performance evaluation
   b) Define function *performance_eval()* that takes in input result file name, ground-truth and predicted traffic, and prints results, plots predicted and ground-truth traffic, and returns performance metrics
      - See details in the skeleton code
      - **Already given in skeleton code**
   c) Perform prediction on the TEST SET using ANN model trained in task 4a), then call function *performance_eval()* to print/save results
      - **Already given in skeleton code**

# Traffic prediction
## Task 4b)-c): expected outputs

MSE: 0.0057034098272676454
MAE: 0.054462841001733334
R2 score: 0.6757174728311439



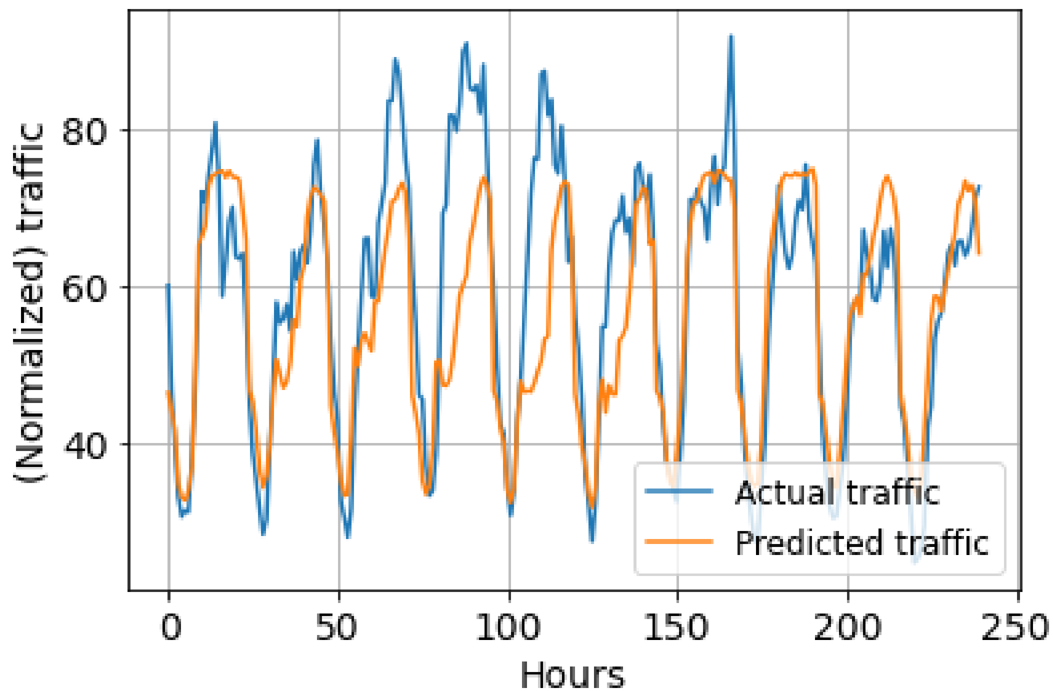
# Traffic prediction
## Task 4

4. ANN model training and performance evaluation **<u>with unscaled dataset</u>**
   d) Consider **unscaled dataset** obtained in task 2b) and repeat the following tasks with the new dataset:
   3b) (dataset split)
   4a) (ANN creation and training)
   4c) (performance evaluation)
      - **What are the main differences w.r.t. previous results?**

# Traffic prediction
## Task 4d): expected outputs

Model training duration [s]: 41.87
Training MSE: 49.723810234089555

MSE: 103.68584184256464
MAE: 7.416771994831083
R2 score: 0.6368949237888343



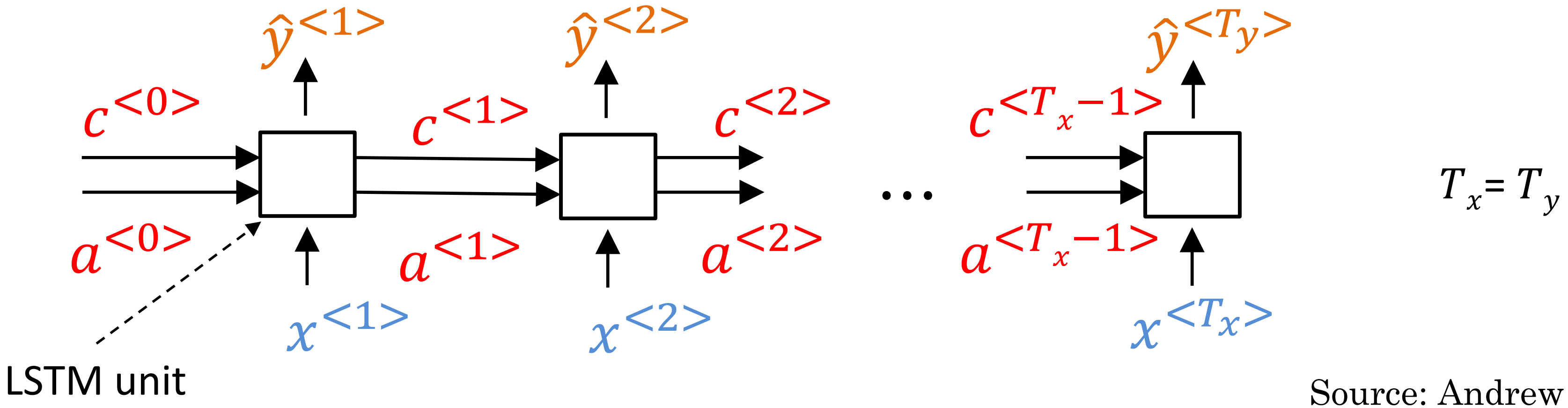
- Now we repeat tasks 2, 3, 4 considering Long-Short-Term-Memory (LSTM) networks instead of ANN

# LSTM-based prediction

## How does an LSTM work?

- Notation
  - $\boldsymbol{x}^{<1>}, \boldsymbol{x}^{<2>}, \ldots, \boldsymbol{x}^{<T_x>}$ **input sequence ($\boldsymbol{x}^{<t>}, t = 1, \ldots, T_x$)**
    - $T_x$ is the number of *time-steps* in the input sequence
  - $\hat{\boldsymbol{y}}^{<1>}, \hat{\boldsymbol{y}}^{<2>}, \ldots, \hat{\boldsymbol{y}}^{<T_y>}$ **output sequence ($\hat{\boldsymbol{y}}^{<t>}, t = 1, \ldots, T_y$)**
    - $T_y$ is the number of *time-steps* in the output sequence
    - $\hat{y}$ is the **predicted** value; the **ground truth is $\boldsymbol{y}^{<t>}, t = 1, \ldots, T_y$**
    - in general $T_x \neq T_y$
  - $\boldsymbol{c}^{<t>}$, state at time−step *t*
  - $\boldsymbol{a}^{<t>}$, activation at time-step *t*

Parameter sharing: the <u>same weights</u> are used by the LSTM unit in all the time steps

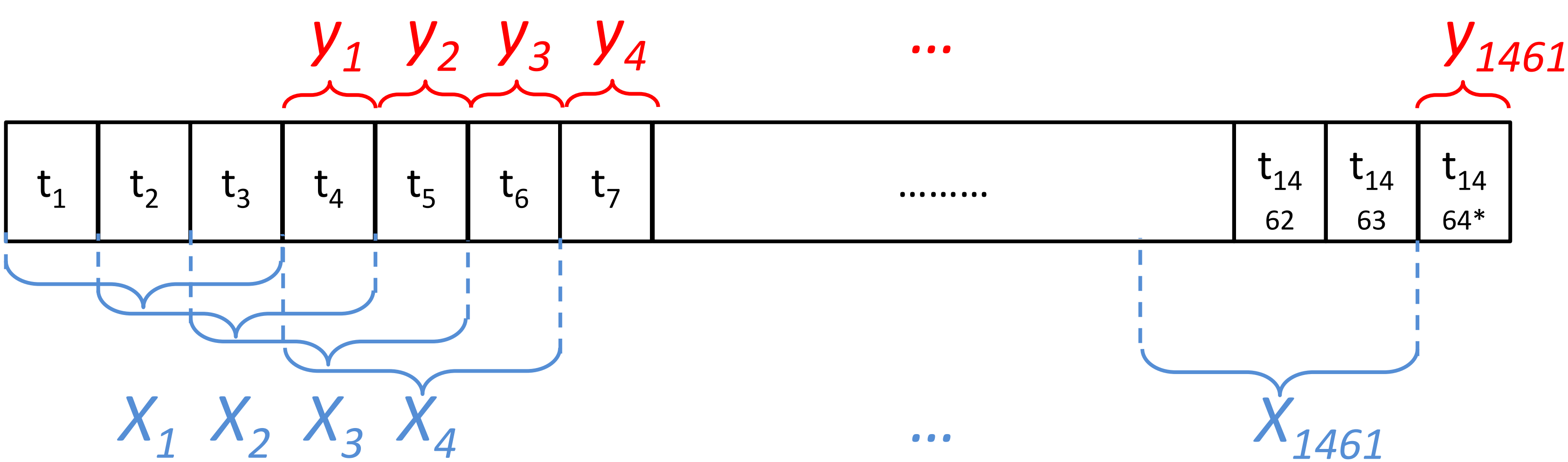



Source: Andrew Ng

# LSTM-based prediction

## Dataset for LSTM in our lab

- For a given cell ID and traffic type, the i-th element of our dataset has:
  - $y_i$ = *(output to be predicted)* = traffic at time $t_i$
  - $X_i$ = *(features)* = traffic at previous *n* hours $t_{i-1}$ , $t_{i-2}$ , … $t_{i-n}$
  - Example: *n*=3 (= "lookback" parameter)




*1464=24hrs*61days

# Traffic prediction
## Task 5

5. LSTM Dataset generation
   a) Define function *generatedataset_LSTM()* that takes in input the traffic dataset as created in task 2b) and number of previous days to consider as input for the LSTM prediction, and returns matrix X and output y as numpy ndarrays
      - **N.B. Features and output values in X and y should be normalized so as to be in [0,1] range**
      - See details in the skeleton code
   b) Call function *generatedataset_LSTM()* considering the dataframe created in task 2b) and **1.5 days "lookback"**, and store outputs in variables *XL, yL*. For each element in *(XL, yL)* verify that the dataset is consistent
      - **Already given in skeleton code**

Expected output (task 5.b):

Number of failed checks: 0

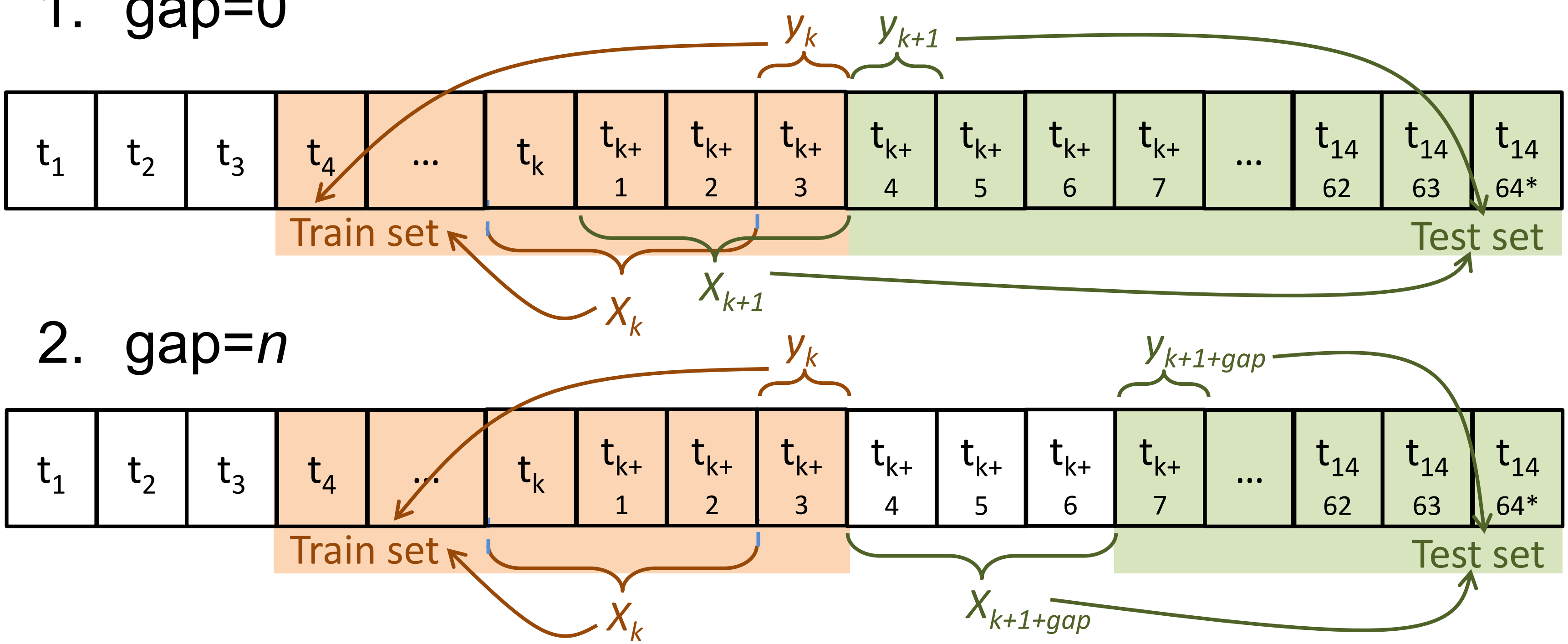
# Traffic prediction
## Task 6

6. LSTM Training/test splitting
   a) Define function *train_test_split_LSTM()* that takes in input the dataset (X,y), the desired amount of test data (number of hours/samples) and the desired gap between train and test sets, and splits (X,y) into train/test sets **with samples ordered chronologically**
      - **Already given in skeleton code**
      - ***See next slide for a "visual" representation of the *gap***
   b) Call function *train_test_split_LSTM()* and generate new training/test sets with the same size of test set used in task 3b) for ANN and gap between train and test sets that provides no overlap between train/test (i.e., **gap = lookback_days**). Verify dimensions of train, test and whole datasets
      - **Already given in skeleton code**

# Traffic prediction
## Task 6

- Hp: n=3 hours (*lookback*)
- How to split train/test sets? What is a "desirable" *gap*?

1. gap=0
2. gap=*n*




In case 2, we include in the test set only "unseen" traffic info

# Traffic prediction

## Task 6a)-b): expected outputs

```
print(XL.shape)
print(yL.shape)
print(X_train_LSTM.shape)
print(y_train_LSTM.shape)
print(X_test_LSTM.shape)
print(y_test_LSTM.shape)
```

```
(1428, 36, 1)
(1428, 1)
(1152, 36, 1)
(1152, 1)
(240, 36, 1)
(240, 1)
```

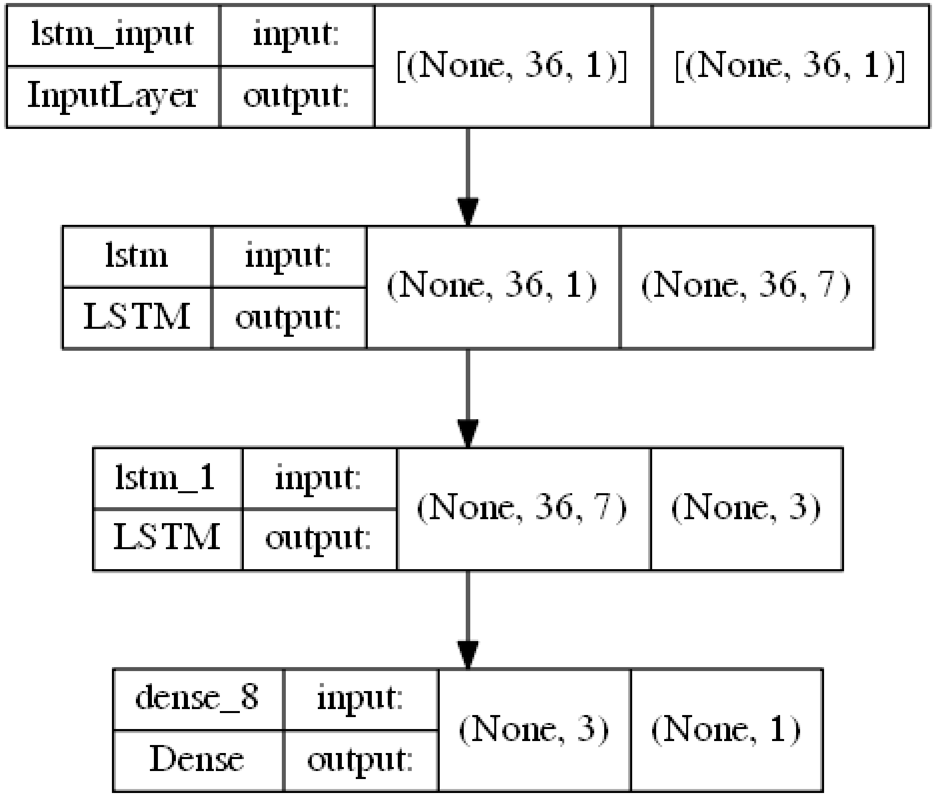
# Traffic prediction

## Task 7

7. LSTM model training and performance evaluation
    a) Build a LSTM with default activation function, 2 hidden layers with [7, 3] neurons, respectively. Print final TRAINING MSE and training duration, and save a figure with model structure. Figure with model structure should be saved as .png file in subfolder 'Results'
        - **N.B. The LSTM output layer is a single neuron that should output a real value (no activation function)**

# Traffic prediction

## Task 7a): expected outputs




LSTM training duration [s]: 54.6
LSTM Training MSE: 0.004076285260273084

Using command
*model_LSTM.summary()*

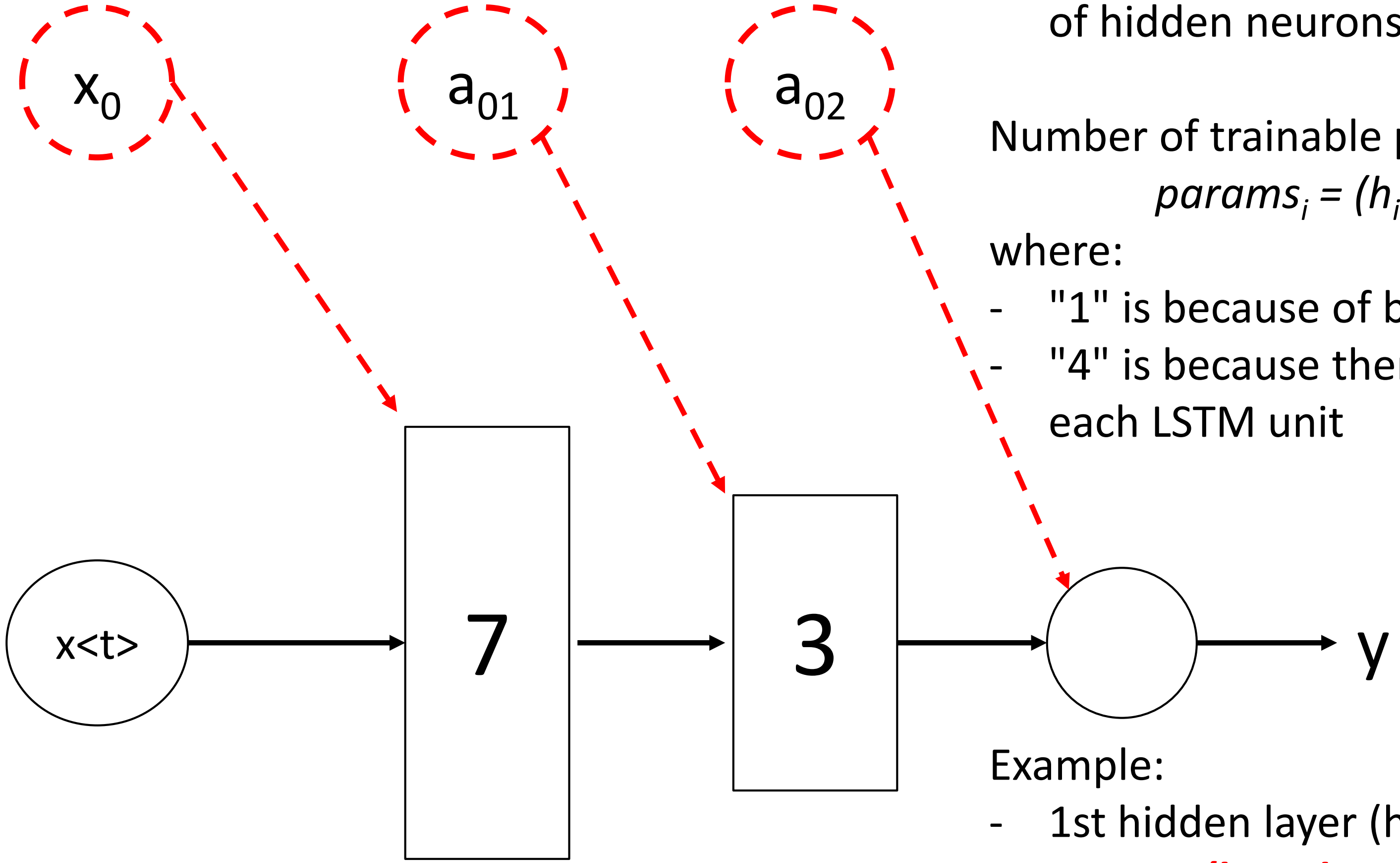
```
_________________________________________________________________
 Layer (type)                Output Shape
=================================================================
 lstm (LSTM)                 (None, 36, 7)             252

 lstm_1 (LSTM)               (None, 3)                 132

 dense_8 (Dense)             (None, 1)                 4

=================================================================
Total params: 388
Trainable params: 388
Non-trainable params: 0
_________________________________________________________________
```

# LSTM model structure




- $h_i$: n. of hidden neurons at layer i
- $d_i$: n. of inputs at layer i (features or n. of hidden neurons at layer i-1)

Number of trainable parameters at layer i:

$$params_i = (h_i + d_i + 1) * 4 * h_i$$

where:

- "1" is because of bias unit
- "4" is because there are 4 *gates* in each LSTM unit

Example:

- 1st hidden layer ($h_1$=7, $d_1$=1 feature):

$$\mathbf{params_1 = (h_1 + d_1 + 1)*4* h_1 = 252}$$

- 2nd hidden layer ($h_2$=3, $d_2$=$h_1$=7):

$$\mathbf{params_1 = (h_2 + d_2 + 1)*4* h_2 = 132}$$

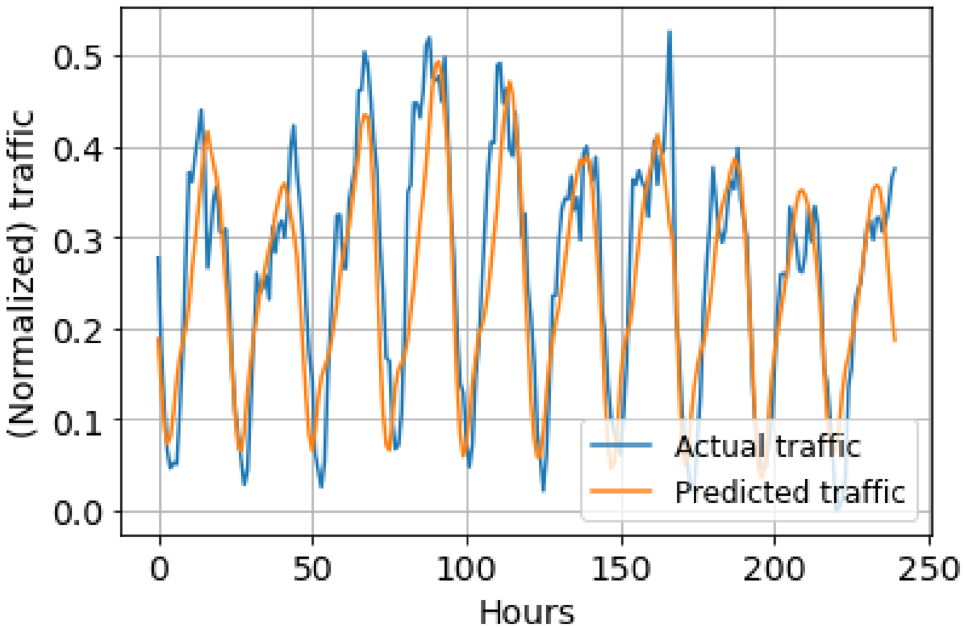
Verify how these change with different lookback days. What do you expect?

# Traffic prediction
## Task 7

7. LSTM model training and performance evaluation
   b) Perform prediction using LSTM model trained in task 7a), then call function *performance_eval()* to print/save results
      - **Already given in skeleton code**

Expected output:

MSE: 0.0047118326530661775
MAE: 0.054037450607771834
R2 score: 0.7320962289913078

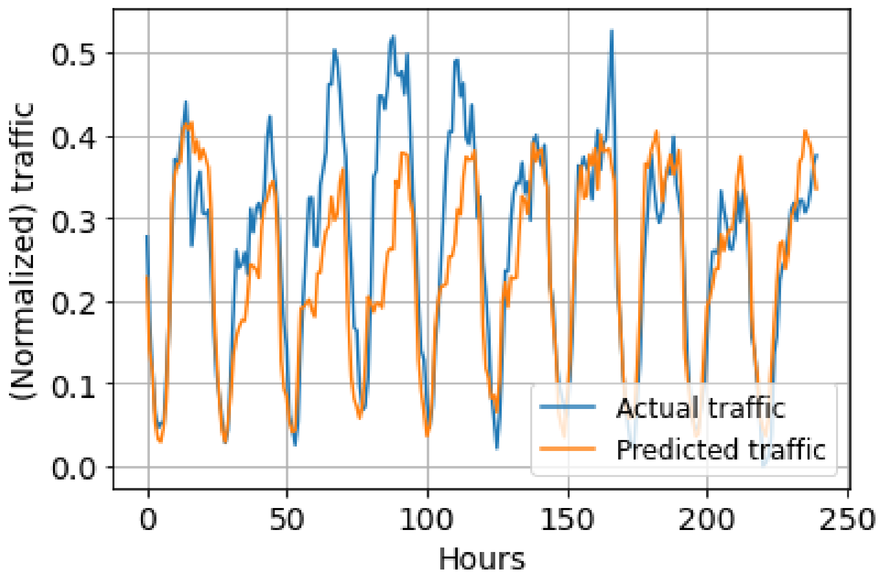


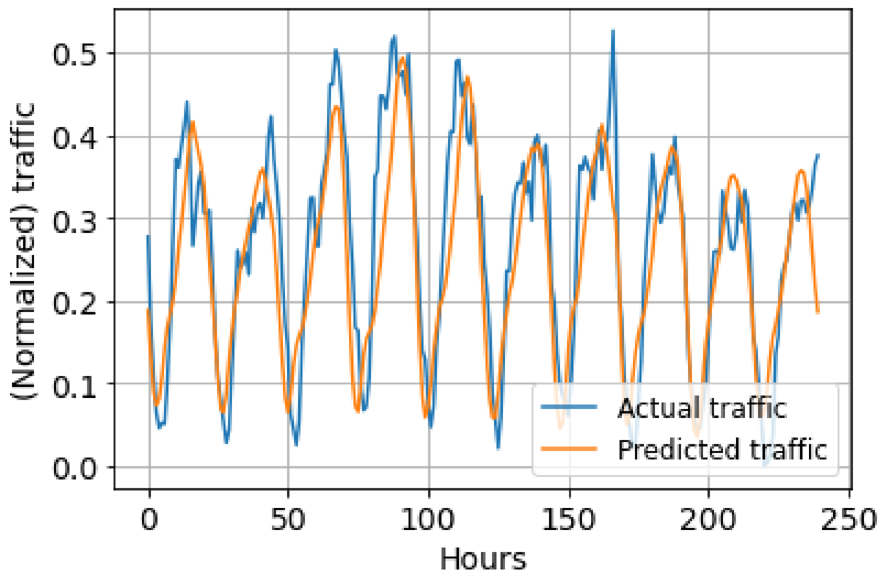

Now compare this result with that for ANN

# Traffic prediction

## Task 7 – comparison between ANN and LSTM

| ANN | LSTM |
|---|---|
| MSE: 0.005703409827267454 | MSE: 0.004711832653066177 |
| MAE: 0.05446284100173334 | MAE: 0.05403745060777183 |
| R2 score: 0.6757174728311439 | R2 score: 0.7320962289913078 |



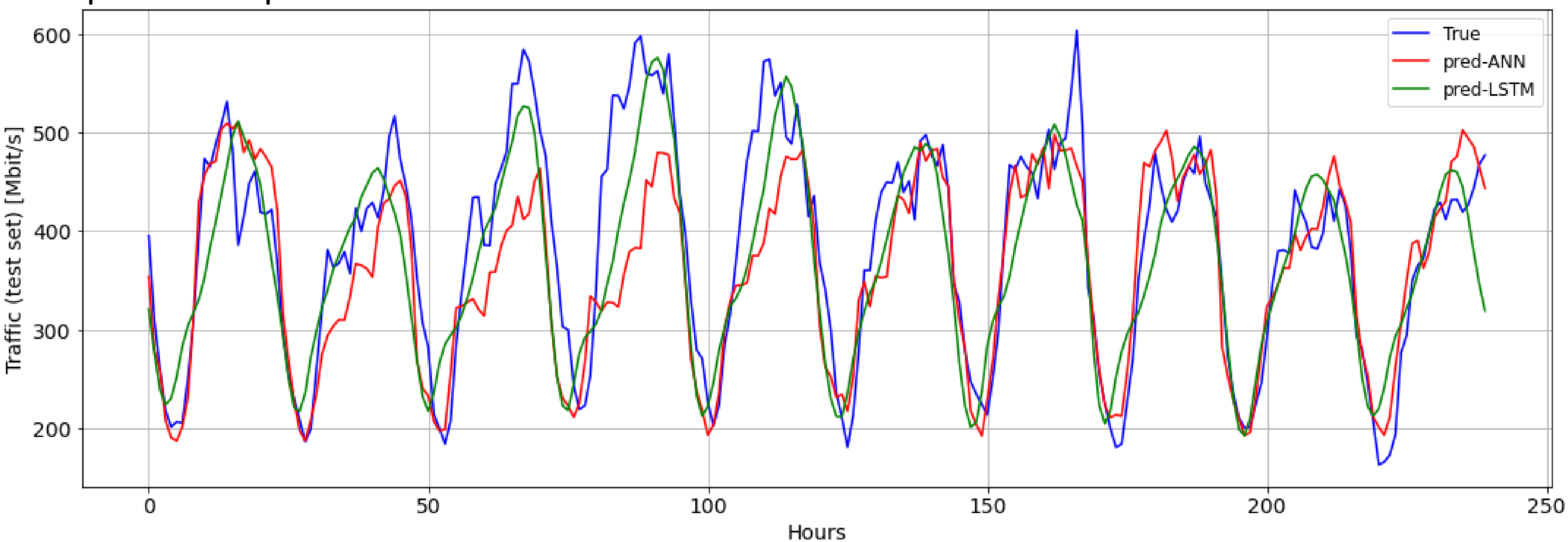
# Additional task

- We have measured the performance of ANN and LSTM predictors in terms of MSE, MAE, R2, but…
  - What is the impact of these metrics for a network operator?
  - How does an operator act after predicting traffic?
  - Do over/under-estimation of traffic provide the same effect?

# Traffic prediction
## Task 8

8. Evaluate the impact of traffic over/under-estimation
   a) Calculate min and max traffic values of the dataframe created in task 2b) and scaled ground-truth, ANN-predicted and LSTM-predicted traffic traces (test set) so as to have maximum traffic = 1 Gbit/s (so far we have worked with a dataset expressed in CDR units). Then, plot the three traffic traces in a single plot

Expected output:



# Traffic prediction

## Task 8a) - hints

1. Ground-truth ($y_{test}$), ANN-predicted ($y_{pred,ANN}$) and LSTM-predicted ($y_{pred,LSTM}$) traffic you have used so far are normalized between 0 and 1
   – First scaling step should be to convert [0,1] range into [min, max] range, (e.g., for $y_{test}$)

$$y_{unscaled}[CDR] = y_{test}[CDR] * (maxtraffic - mintraffic) + mintraffic$$

   – where:

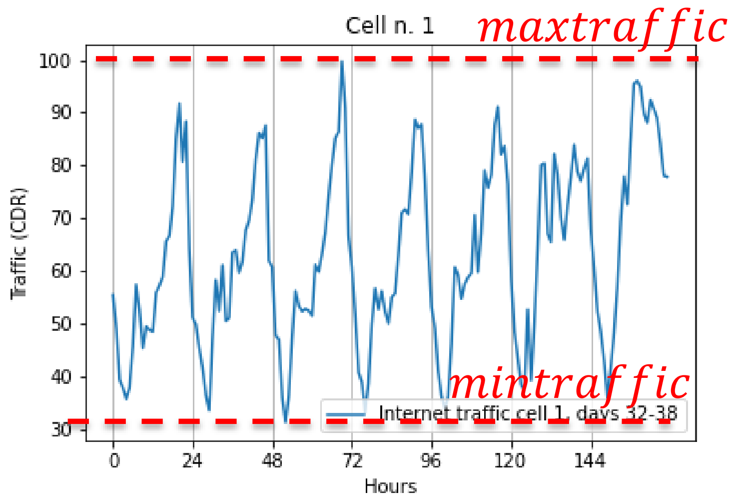
$$maxtraffic = maximum\ traffic\ for\ the\ cell\ \boldsymbol{over\ the\ entire\ 61-days}$$

$$mintraffic = minimum\ traffic\ for\ the\ cell\ \boldsymbol{over\ the\ entire\ 61-days}$$

2. Traffic up-scaling (i.e., from CDR into Gbit/s units) must be done so as to have the maximum traffic along the entire period of 61 days equal to 1 Gbit/s

- *Example: assume maxtraffic = 100 CDR (see fig.)*
  – *For a generic traffic value y [CDR], you should obtain the final upscaled value as*
  –

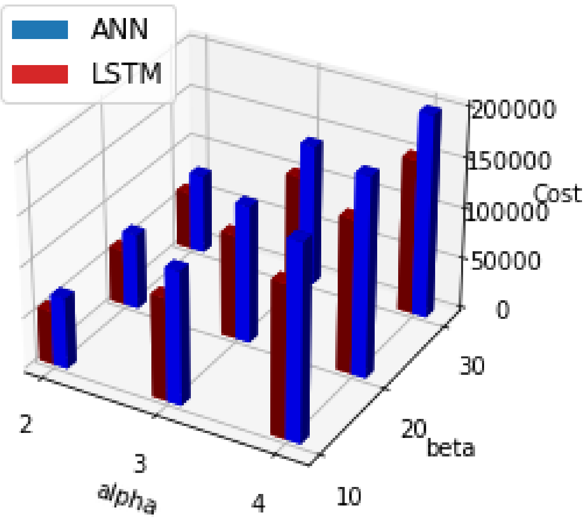
$$y\,[Gbit/s] = y_{unscaled}[CDR] * \frac{1\ Gbit/s}{100\ CDR}$$



# Traffic prediction
## Task 8

8. Evaluate the impact of traffic over/under-estimation
   b) Define function *evaluate_cost()* that takes in input ground-truth, ANN-predicted and LSTM-predicted traffic traces (scaled as in task 8a) and two cost parameters *alpha* and *beta* for over/under-provisioning and returns cost of over/under-provisioning for the ANN and LSTM cases, assuming a given resource allocation policy
      - See details in the skeleton code
   c) Use function *evaluate_cost()* with given over/under-provisioning cost weights using ground-truth, ANN-predicted and LSTM-predicted traffic traces above; plot results in a 3D graph

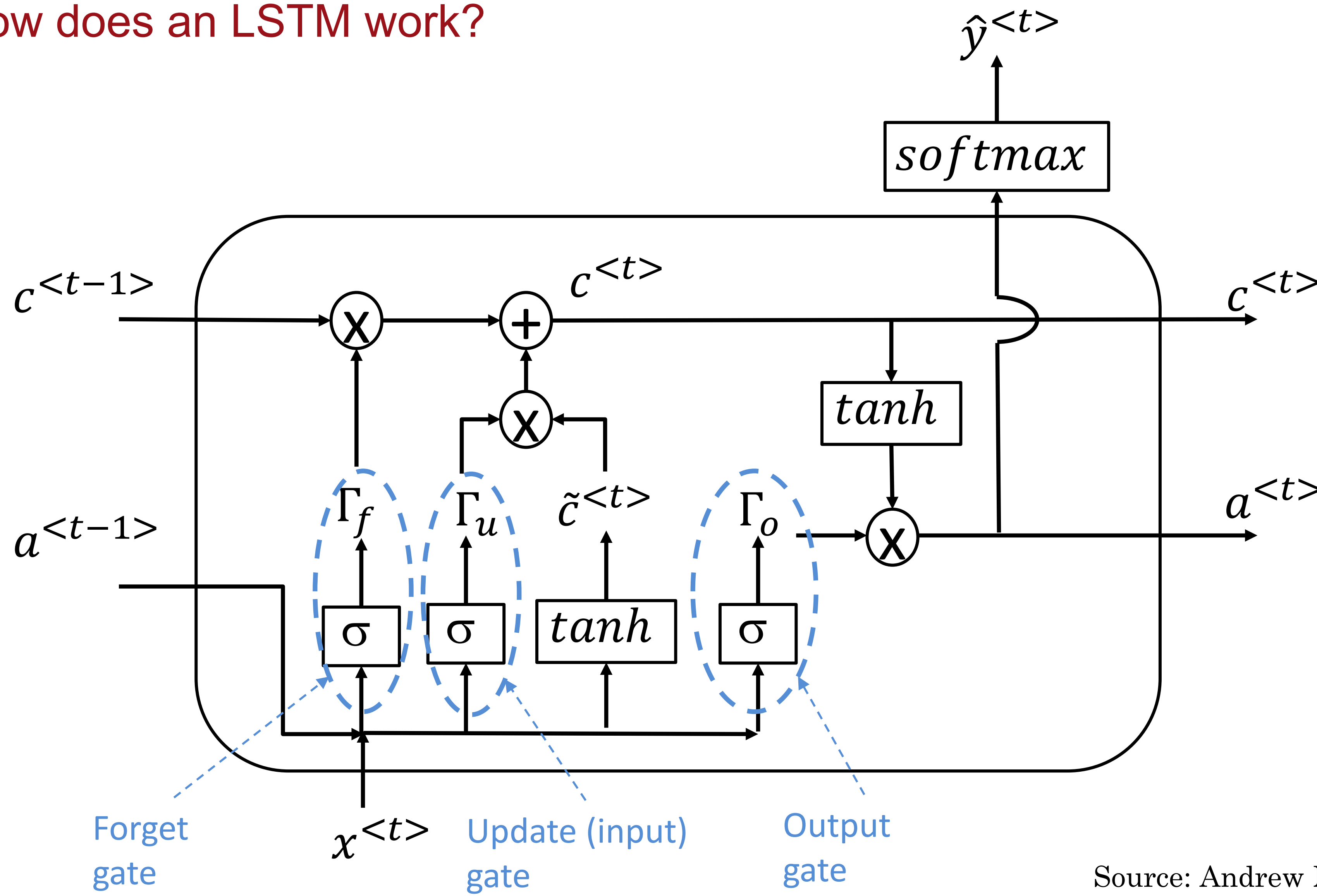
Expected output:



- Backup slides

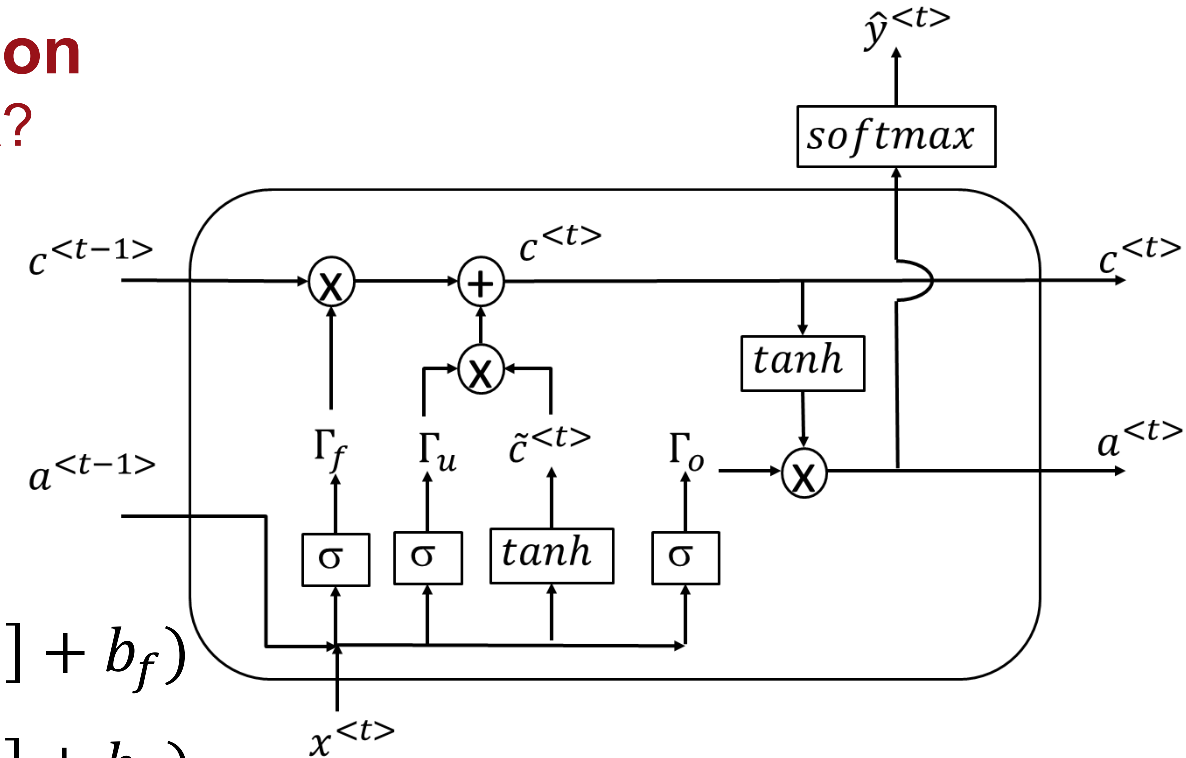
# LSTM-based prediction

## How does an LSTM work?




Source: Andrew Ng

# LSTM-based prediction
## How does an LSTM work?




$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$

$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$c^{<t>} = \Gamma_u \cdot \tilde{c}^{<t>} + \Gamma_f \cdot c^{<t-1>}$$

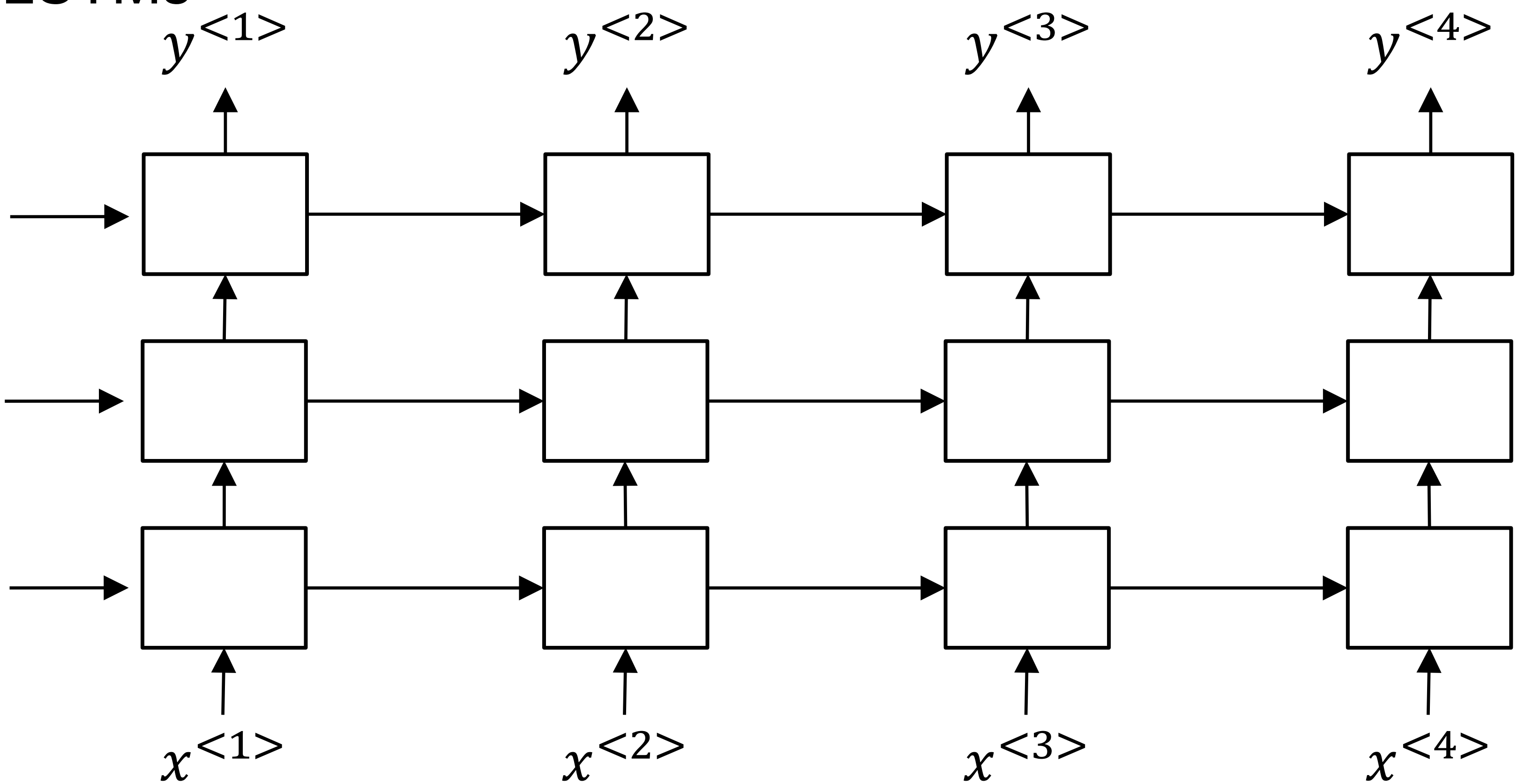
$$a^{<t>} = \Gamma_o \cdot \tanh(c^{<t>})$$

$$y^{<t>} = softmax(W_y\, a^{<t>} + b_y)$$

Source: Andrew Ng

# LSTM-based prediction
## Deep LSTMs

- As in Deep ANNs, more hidden layers can be used also in LSTMs




Source: Andrew Ng

# LSTM-based prediction

## How does an LSTM work?

- See also:
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
- https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/