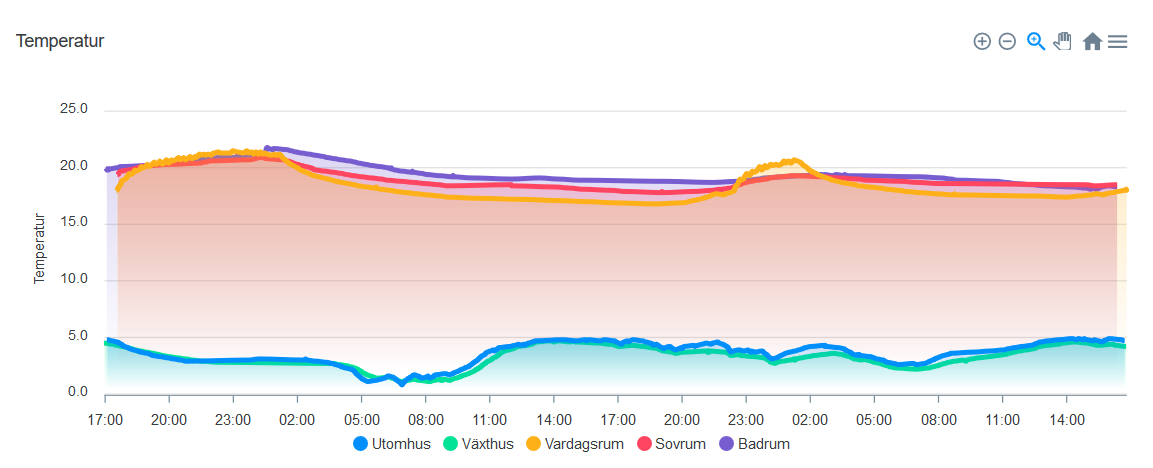
**Laboration 3**

**Inledning**

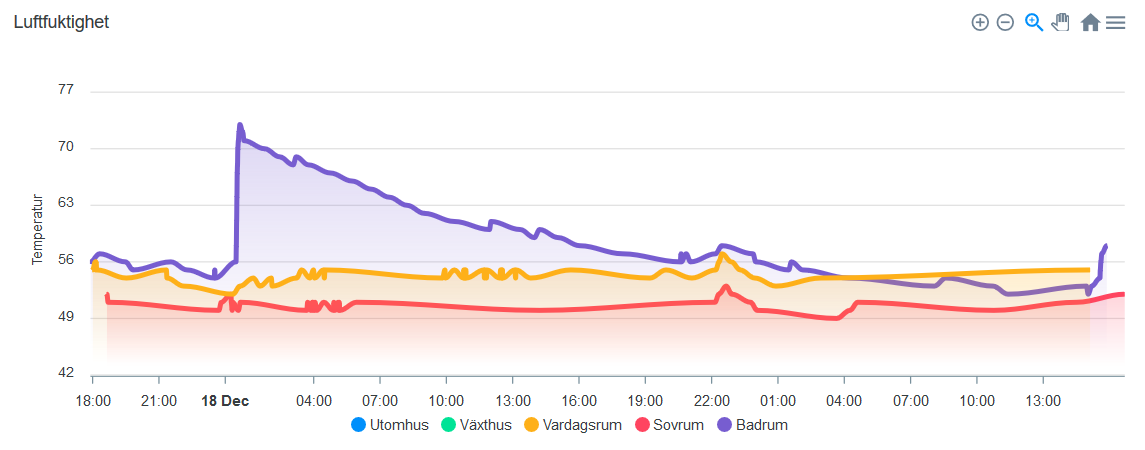
Mitt området som jag tänkte undersöka är temperaturprognos för utomhustemperaturen där mitt hus är placerat. Ska se om jag kan få fram en prognos för utomhustemperaturen där mitt hus är placerat som är bättre än den prognos som SMHI ger.

**Bakgrund**

Jag har under det senaste halvåret samlat in temperatur- och luftfuktighets-data från 5 olika sensorer som mäter luftfuktighet samt temperatur inomhus (vardagsrum, sovrum, badrum) och utomhus (utomhus, växthus) genom att ta i mot datan med hjälp av en raspberry pi som kommunicerar med sensorerna genom 433Mhz och lagrar temperatur och luftfuktighet för de fem sensorerna i en MySQL databas på min rapsberry pi. På min raspberry pi ligger sedan en Django webbserver som hämtar data från databasen och presenterar den som ett API, som en webbsida sedan läser för att presentera data:n som en graf.

Figur 1: Temperaturdata senaste 48H 2019-12-19 17:00

SMHI ger en prognos för temperatur för mitt område, men jag tycker det skulle vara intressant om jag lyckas göra en prognos som är nära den SMHI ger med den datorkraft jag har här hemma i form av en bärbar laptop samt min raspberry pi. Tanken är att göra en prognos för de kommande 24-timmarna och jämföra med 24-timmars prognosdata från SMHI. SMHI har säkert stor datorkraft och gjort modeller för temperaturens utveckling över dygnet.

Figur 2: Luftfuktighetsdata 48H 2019-12-19 17:00

**Frågeställning**

Min frågeställning är om det går att göra en prognos som är nära den SMHI ger för de kommande 24 timmarna, med den datorkraft jag har i form av en raspberry pi och en bärbar dator.

**Metod**

*Source: export\_outdoor.sql, expand\_data.py, model\_train.py*

Data från de olika temperatur- och luftfuktighets-sensorerna ligger lagrad på en MySQL-databas på raspberry pi:n. Det första steget är att exportera data från databasen till en csv-fil, vilket görs med *export\_outdoor.sql.*

Datamängden ser ut på följande sätt:

"7889";"1";"215";"19.200000000";"2019-08-12 16:43:09"

"7890";"2";"215";"80.000000000";"2019-08-12 16:43:09"

"7891";"1";"215";"19.300000000";"2019-08-12 16:44:45"

"7895";"1";"215";"19.400000000";"2019-08-12 16:51:57"

"7897";"2";"215";"79.000000000";"2019-08-12 16:52:45"

"7898";"2";"215";"78.000000000";"2019-08-12 16:55:09"

"7900";"2";"215";"79.000000000";"2019-08-12 16:58:21"

"7901";"2";"215";"78.000000000";"2019-08-12 16:59:09"

"7903";"1";"215";"19.300000000";"2019-08-12 16:59:57"

"7904";"1";"215";"19.100000000";"2019-08-12 17:07:09"

"7906";"2";"215";"79.000000000";"2019-08-12 17:07:57"

...

För att spara plats i databasen så läggs endast värden till i databasen om det är nuvarande värdet skiljer sig från det föregående. För att träna modellerna ska det finnas ett värde för varje timme under perioden som används. Det första steget blir då att expandera datamängden i csv-filen så att det blir en datapunkt för varje timme. Detta görs med hjälp av *expand\_data.py*, vilket sparar den expanderade datamängden i en ny csv fil. Datamängden ser ut på följande sätt:

2019-08-12 16:00:00,19.238749999999996,79.81166666666668

2019-08-12 17:00:00,19.110416666666666,77.58583333333334

2019-08-12 18:00:00,18.355500000000003,79.45416666666667

2019-08-12 19:00:00,17.678166666666666,80.48166666666667

2019-08-12 20:00:00,17.028833333333335,83.175

2019-08-12 21:00:00,15.920166666666667,87.20416666666668

2019-08-12 22:00:00,14.86625,93.08333333333333

2019-08-12 23:00:00,14.306333333333333,96.1875

2019-08-13 00:00:00,13.800833333333333,98.0

2019-08-13 01:00:00,13.05938888888889,99.0

2019-08-13 02:00:00,12.42888888888889,99.0

...

Perioden som det finns data mellan är 2019-08-12 16:00:00 till 2019-12-05 22:00:00, det finns nu data från sensommar till vinter under lite mindre än ett halvår med de variationer som följer.

De modeller som kommer att testas är alla-till-alla nät och ett återkopplande nät i form av ett GRU nät (tänkte även använda LSTM, men det tog för lång tid att träna modellen). För att göra en prognos för 24 timmar fram i tiden används data från fem dagar bakåt i tiden. För att göra detta har jag gjort 24 modeller som vardera som representerar en av de kommande 24 timmarna. I scriptet *model\_train.py* läses datan in från csv-filen till 24 numpy arrayer som kommer användas för att träna modellerna. Varje datamängd *i* har shape= (2623, 120, 2) som är antalet datapunkter för fem dagar bak i tiden, innehållande både utomhus temperatur och luftfuktighet för den timme *i* fram i tiden samt 24 numpy arrayer, där array *n* som innehåller den faktiska temperaturen för timme *n* fram i tiden med shape = (2623, 1). Datamängdens ordning randomiseras sedan för att delas upp i en 20% testmängd.

Figur 6: Validation och training loss för timme 24

**Alla-till-all nät**

Det alla-till-all nät som testades såg ut på fäljande sätt:

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

flatten\_2 (Flatten) (None, 240) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_5 (Dense) (None, 64) 15424

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_6 (Dense) (None, 64) 4160

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_7 (Dense) (None, 64) 4160

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_8 (Dense) (None, 1) 65

=================================================================

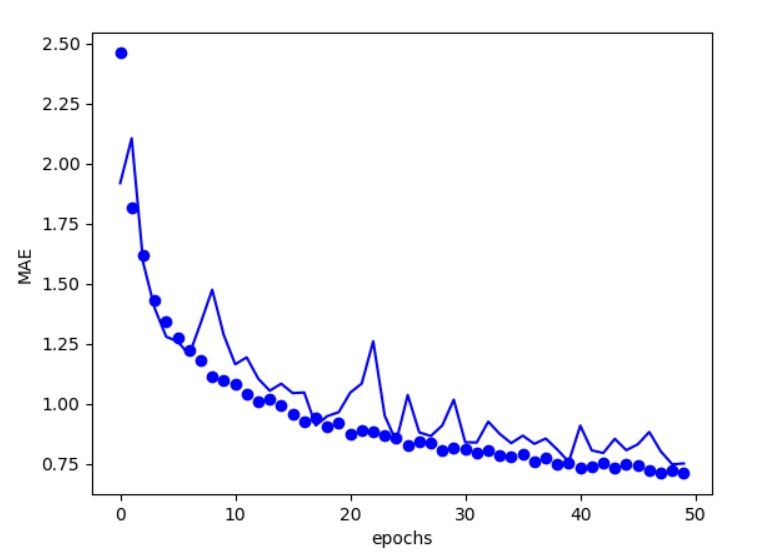
Total params: 23,809

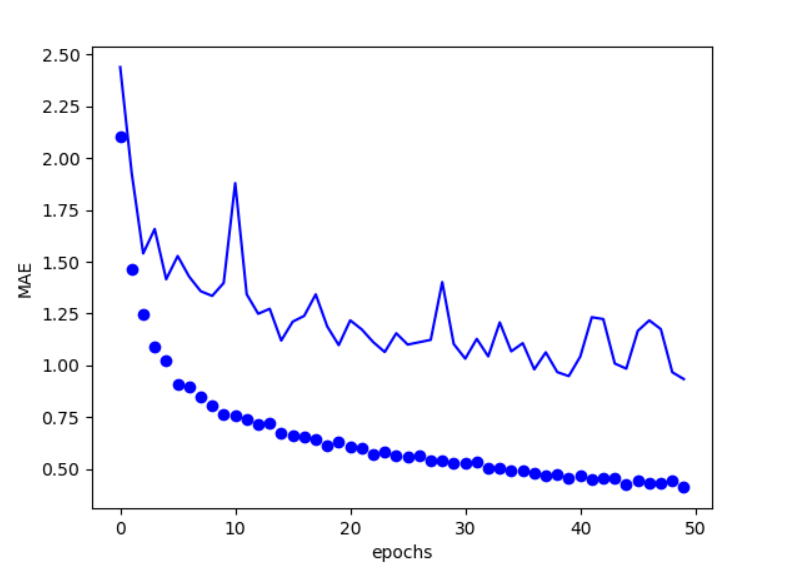
Trainable params: 23,809

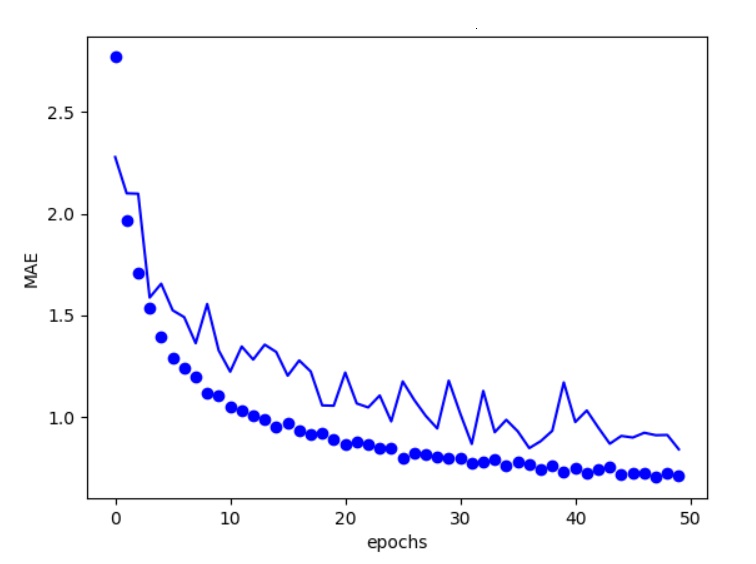
Non-trainable params: 0

Näten tränades sedan i 50 epoker, och medeltalet för MEA i grader blev för de 24 näten: 0.966

Nedan visas training loss (punkter) och validation loss (linjer) för nätet 1, 13 och 24, som representerar antal timmar fram i tiden. Modellerna samt dess vikter sparades sedan ner till fil.

  
Figur 3: Validation och training loss för timme 13

  
Figur 4: Validation och training loss för timme 1

  
Figur 5: Validation och training loss för timme 23

**Återkopplat nät GRU**

Det återkopplande GRU nät som testades såg ut på följande vis:

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

gru\_1 (GRU) (None, 120, 32) 3360

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

gru\_2 (GRU) (None, 32) 6240

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 1) 33

=================================================================

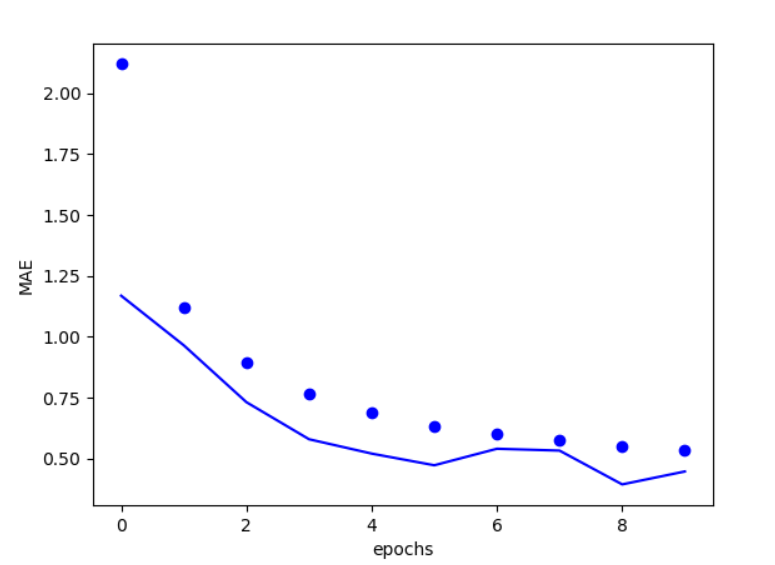
Total params: 9,633

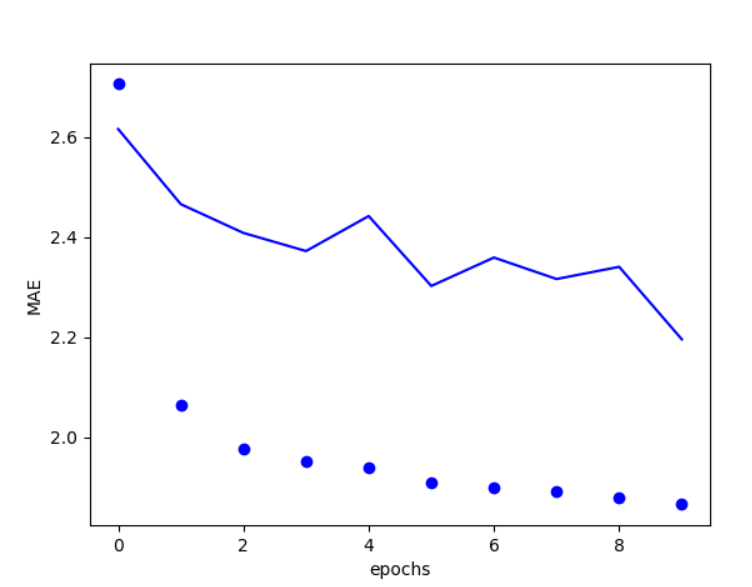
Trainable params: 9,633

Non-trainable params: 0

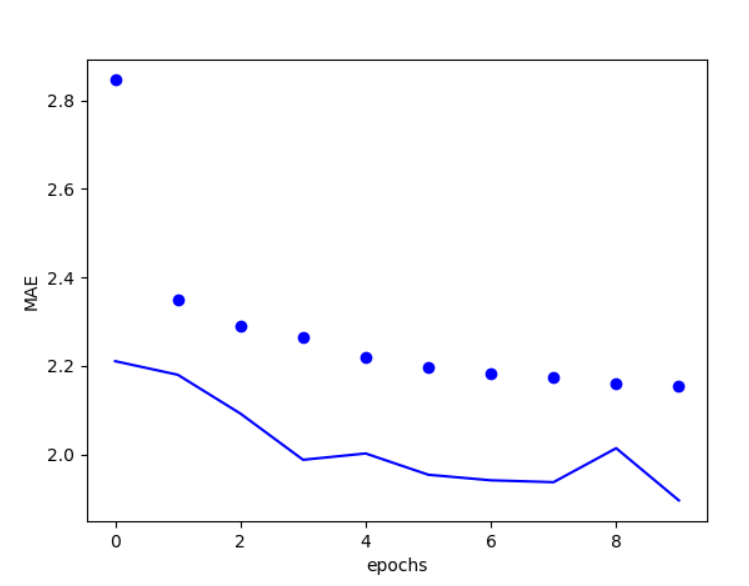
Näten tränades sedan i 10 epoker, och medeltalet för MEA i grader blev för de 24 näten: 1.775

Nedan visas training loss (punkter) och validation loss (linjer) för nätet 1, 13 och 24, som representerar antal timmar fram i tiden. Modellerna samt dess vikter sparades sedan ner till fil.

  
Figur 7: Validation och training loss för timme 1

  
Figur 8: Validation och training loss för timme 13

**Raspberry pi**

  
Figur 9: Validation och training loss för timme 23

*Source:*

Modellerna från alla till alla nät samt GRU skickades sedan över till min rapsberry pi.

**Resultat**

**Diskussion**