Recurrent Neural Networks for Energy Forecasting

Nikil Pancha

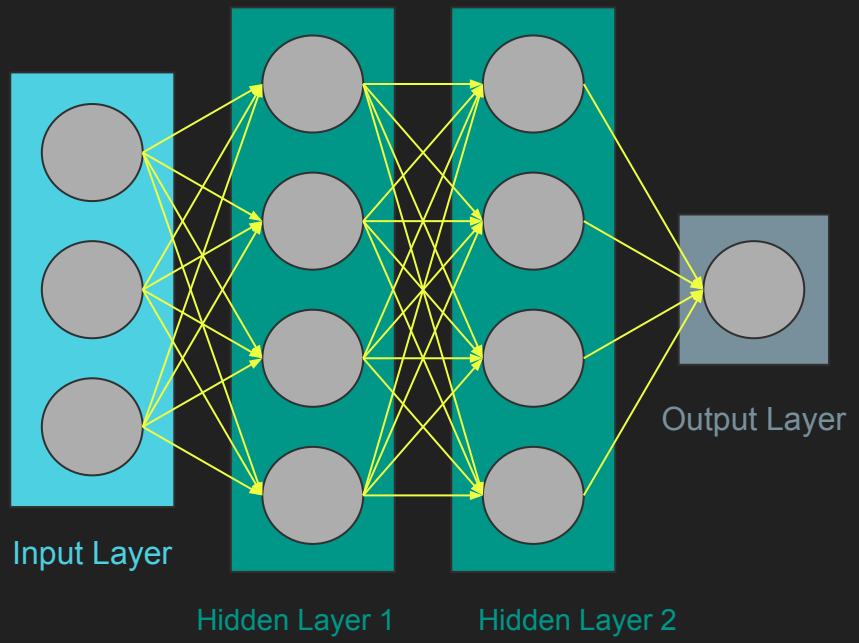
Neural Networks

Universal function approximators

Learnable weights

Non-convex

Trainable by stochastic gradient descent and backpropagation



Regularization

Weight norm regularization: penalty on norm of weights (usually $L_{1,1}$ or $L_{2,2}$)

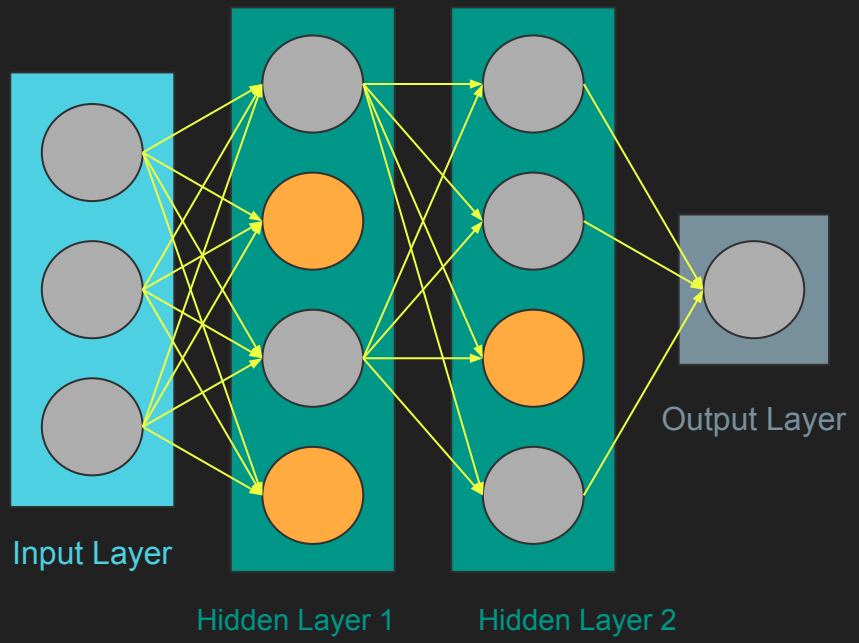
Batch normalization: after each layer, normalize the outputs (subtract batch mean and divide by standard deviation). Learn scale and offset to be applied after normalization

Dropout: Randomly set a proportion of nodes to output 0 for a particular batch.

Dropout

Usually drop 20% to 50% of nodes

Acts as a method of model averaging



Convolutional NNs

Learn weights of filters to apply to inputs

Widely used in image processing

Require fewer learnable parameters (local connectivity)

Inspired by how cats and monkeys see

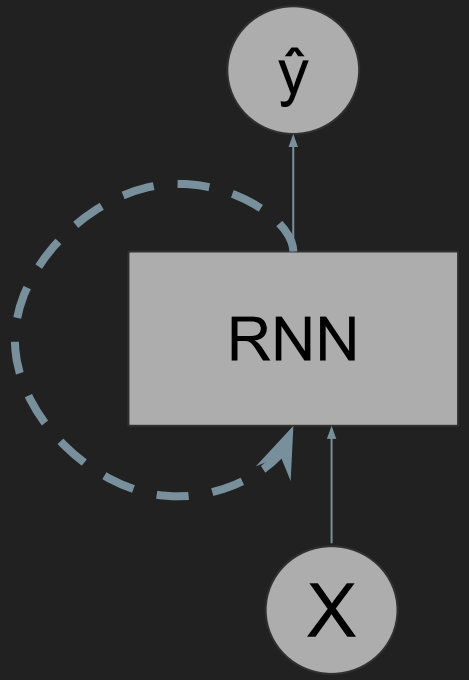
Recurrent Neural Networks

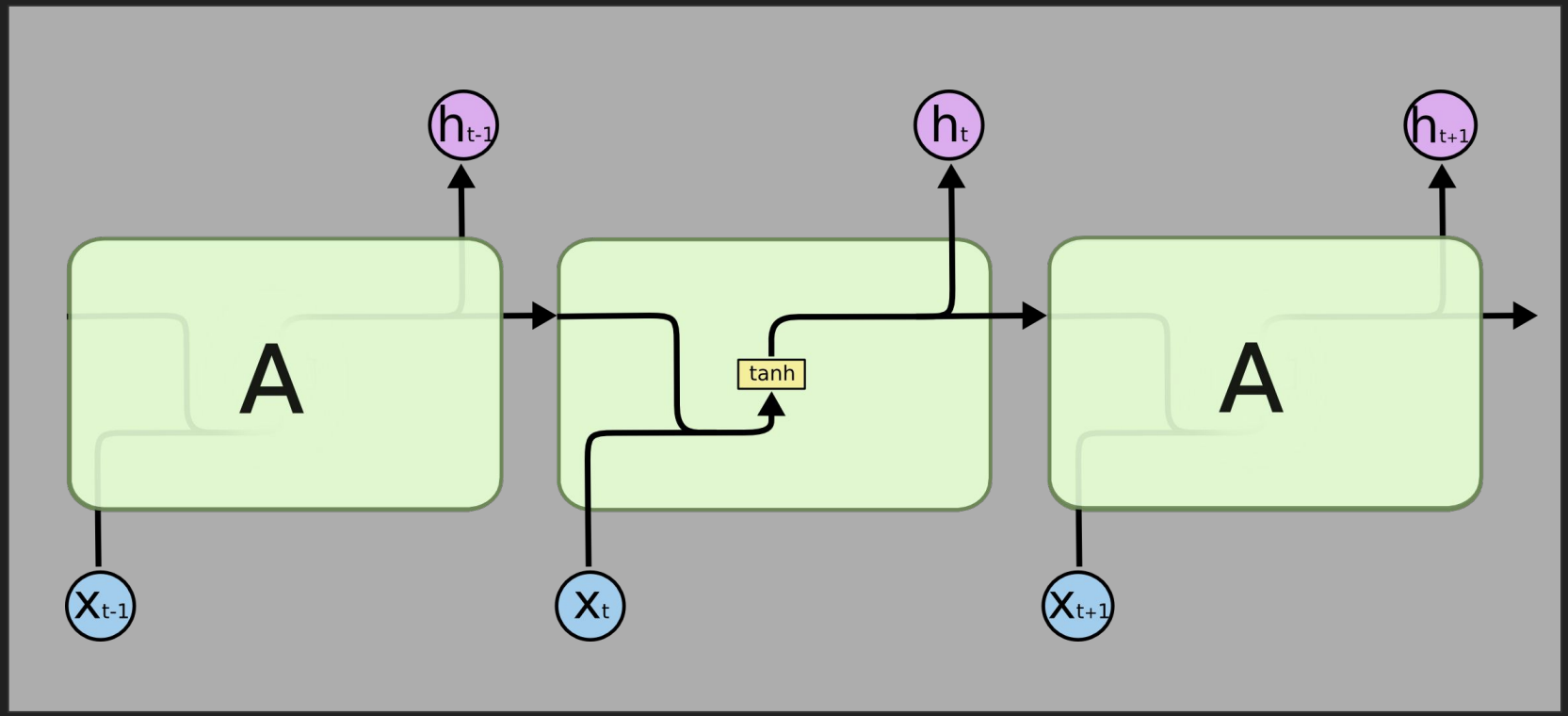
Used on sequential data

Trained by backpropagation through time (BPTT)

RNN uses:

- Audio
- Text
- Images
- Time series





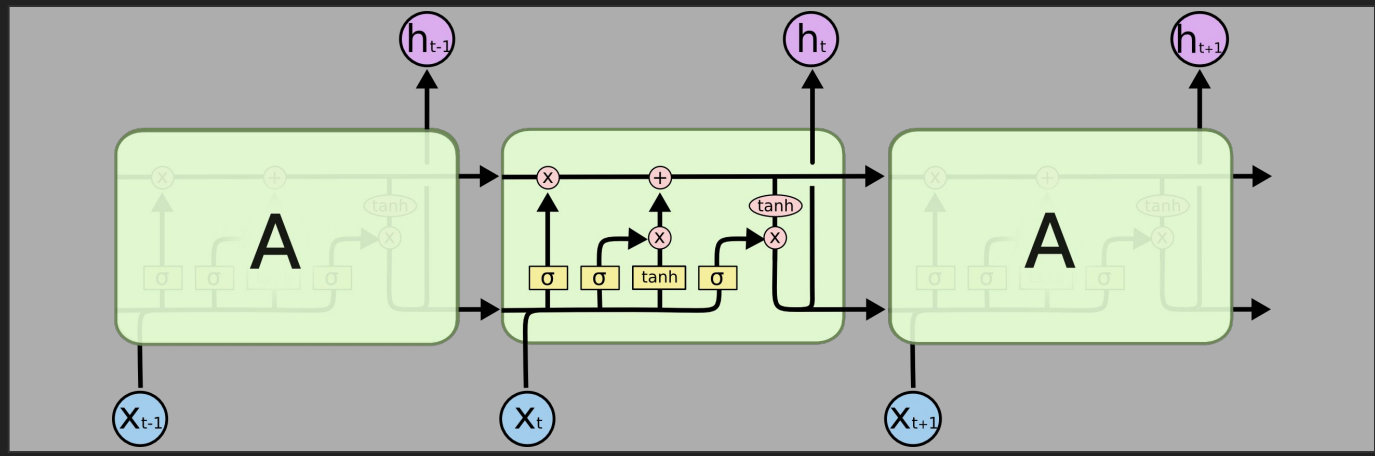
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

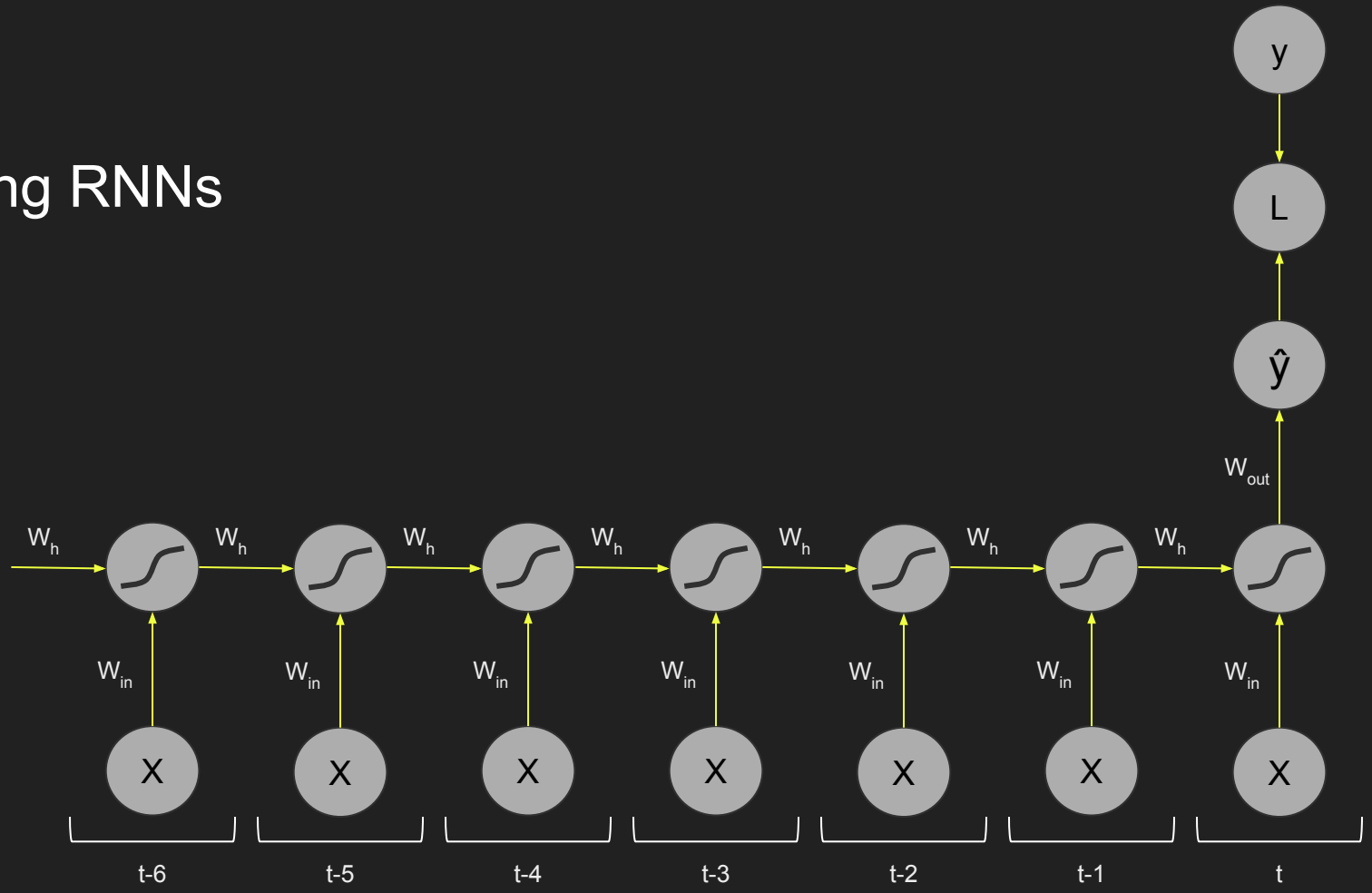
Allows long term dependencies

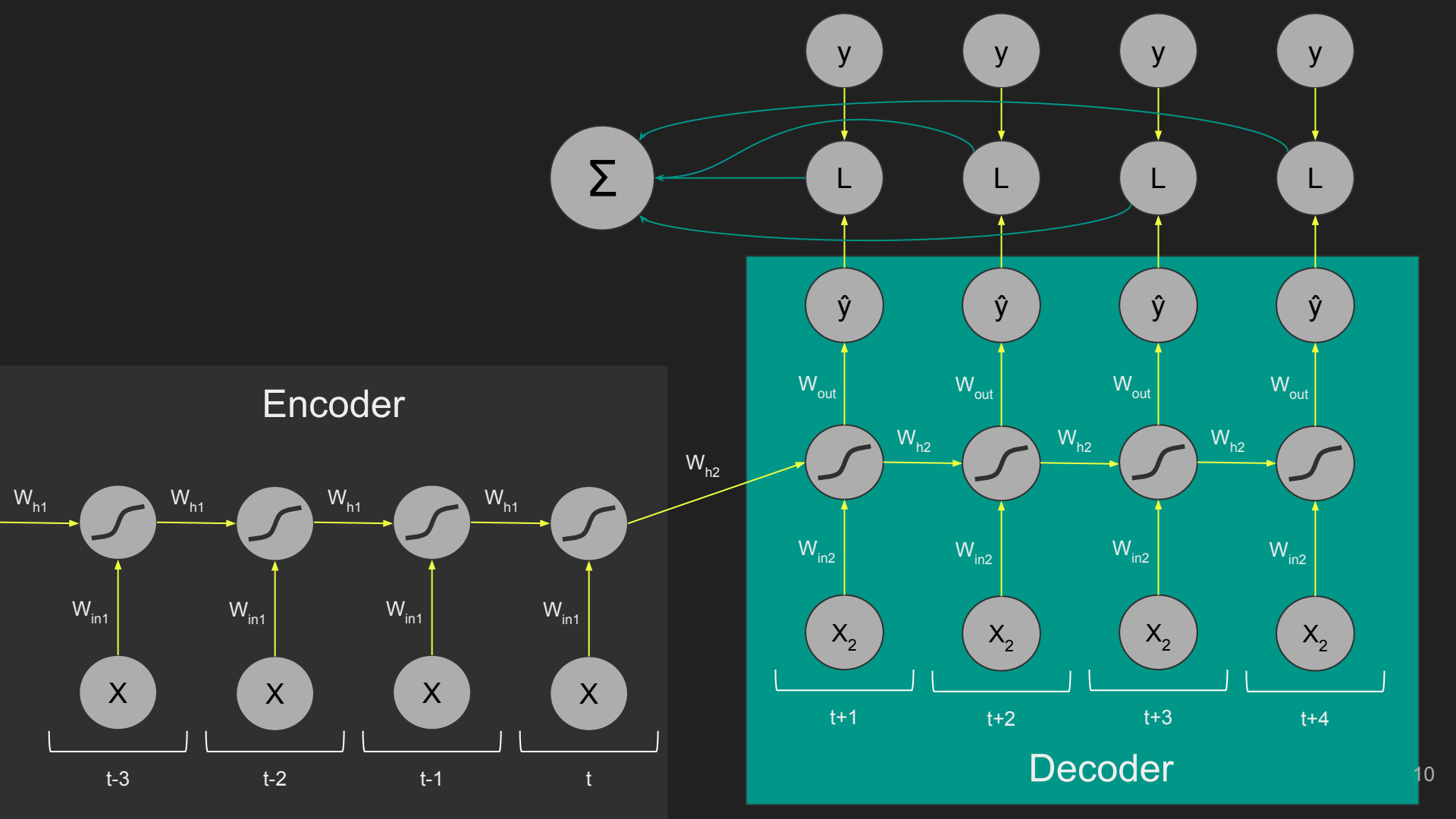
Adds several gates to control information flow:

- Forget
- Input
- Output



Unrolling RNNs




Encoder
Decoder
Σ
y
L
ŷ
X
X_2
W_{h1}
W_{in1}
W_{h2}
W_{in2}
W_{out}
t-3
t-2
t-1
t
t+1
t+2
t+3
t+4

Tools

Tensorflow

Keras

SciPy/NumPy/Pandas/Matplotlib

```
/*
 *  Copyright (c) 2006-2010, Intel Mobile Communications.  All rights reserved.
 *
 *   This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 *        This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *
 *  GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *    along with this program; if not, write to the Free Software Foundation,
 *  Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */
```

```
#include <linux/kexec.h>
#include <linux/errno.h>
#include <linux/io.h>
#include <linux/platform_device.h>
#include <linux/multi.h>
#include <linux/ckevent.h>

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>

#define REG_PG    vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)     (func)
```

```
#define SWAP_ALLOCATE(nr)     (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %%esp, %0, %3" : : "r" (0));   \
  if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
          pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
  PUT_PARAM_RAID(2, sel) = get_state_state();
  set_pid_sum((unsigned long)state, current_state_str(),
           (unsigned long)-1->lr_full; low;
}
```


http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma **??**. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{tikzcd}
S \arrow[r] \arrow[d] & & & \\
\xi \arrow[r] & \mathcal{O}_{X'} \arrow[dr] & & \\
\mathrm{gor}_s \arrow[u] & & & \\
 & = \alpha' \arrow[r] \arrow[d, leftrightarrow] & & \\
 & = \alpha' \arrow[r] & \alpha & X \arrow[d] \\
 & \mathrm{Spec}(K_\psi) & \mathrm{Mor}_{Sets} & \mathrm{d}(\mathcal{O}_{X_{X/k}}, \mathcal{G})
\end{tikzcd}$$

is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas **??**.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-}1(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{x}})$$

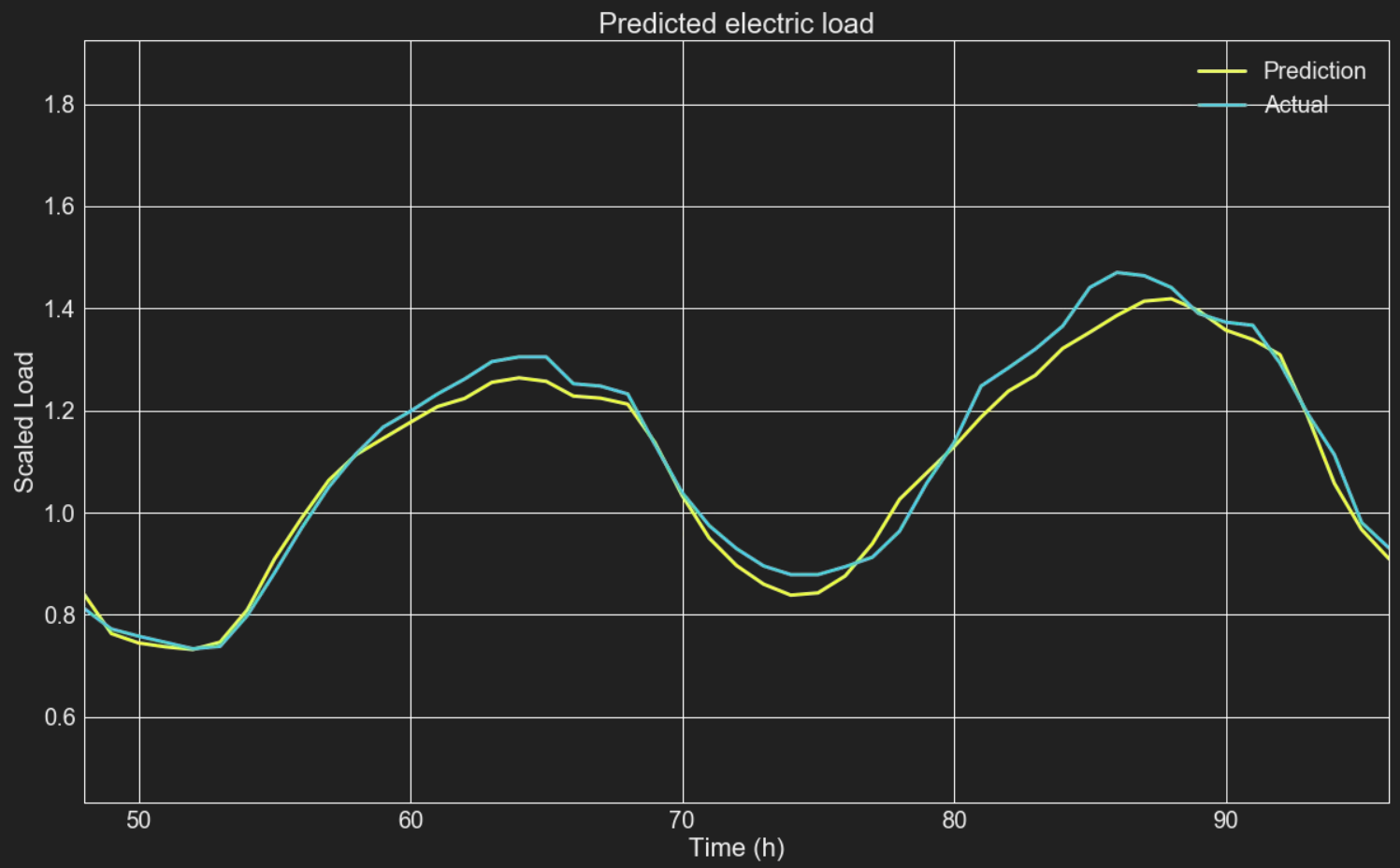
is an isomorphism of covering of $\mathcal{O}_{X_\iota}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition **??** and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

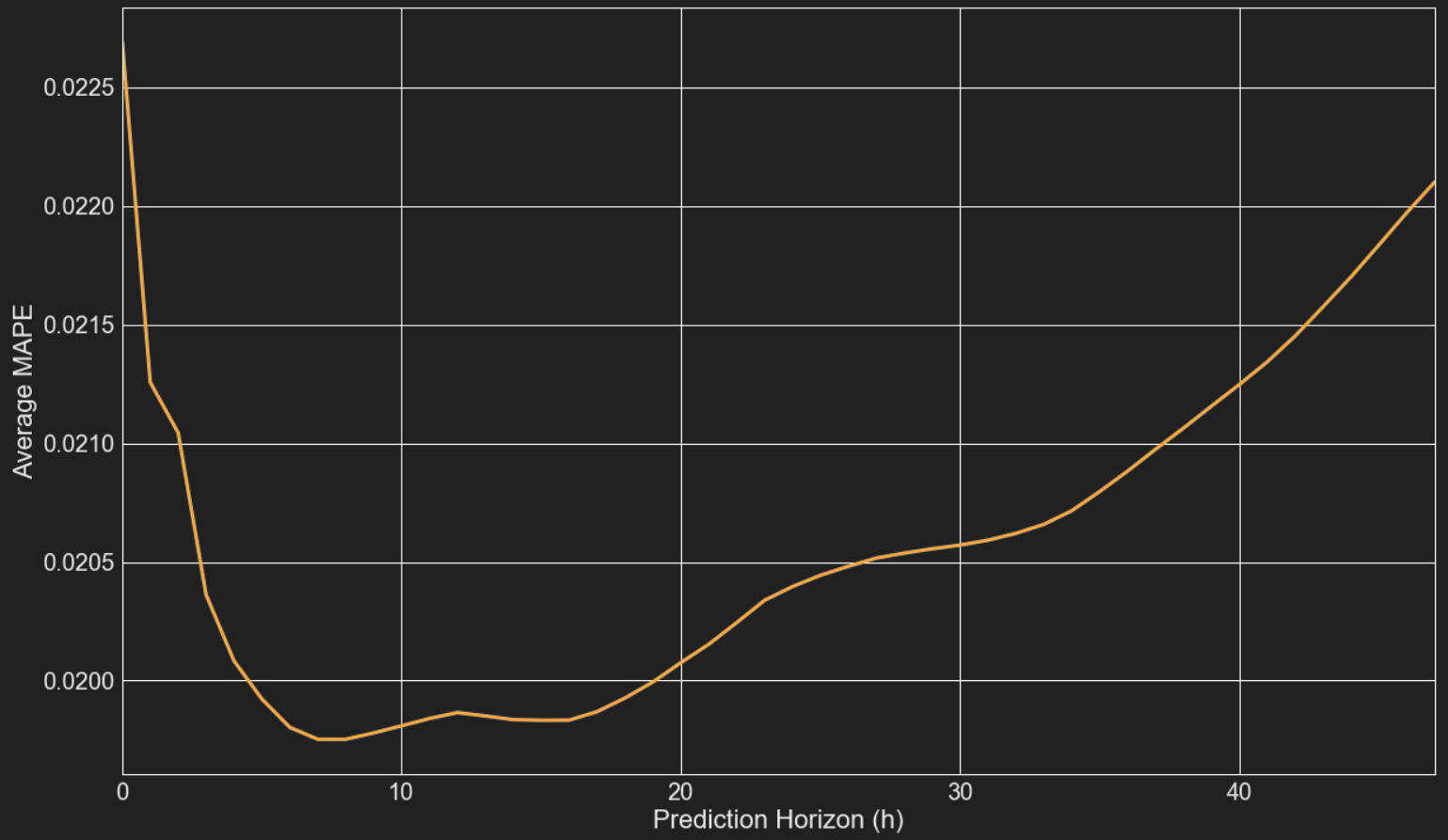
If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma **??**. This is a sequence of $\mathcal{F}$ is a similar morphism.

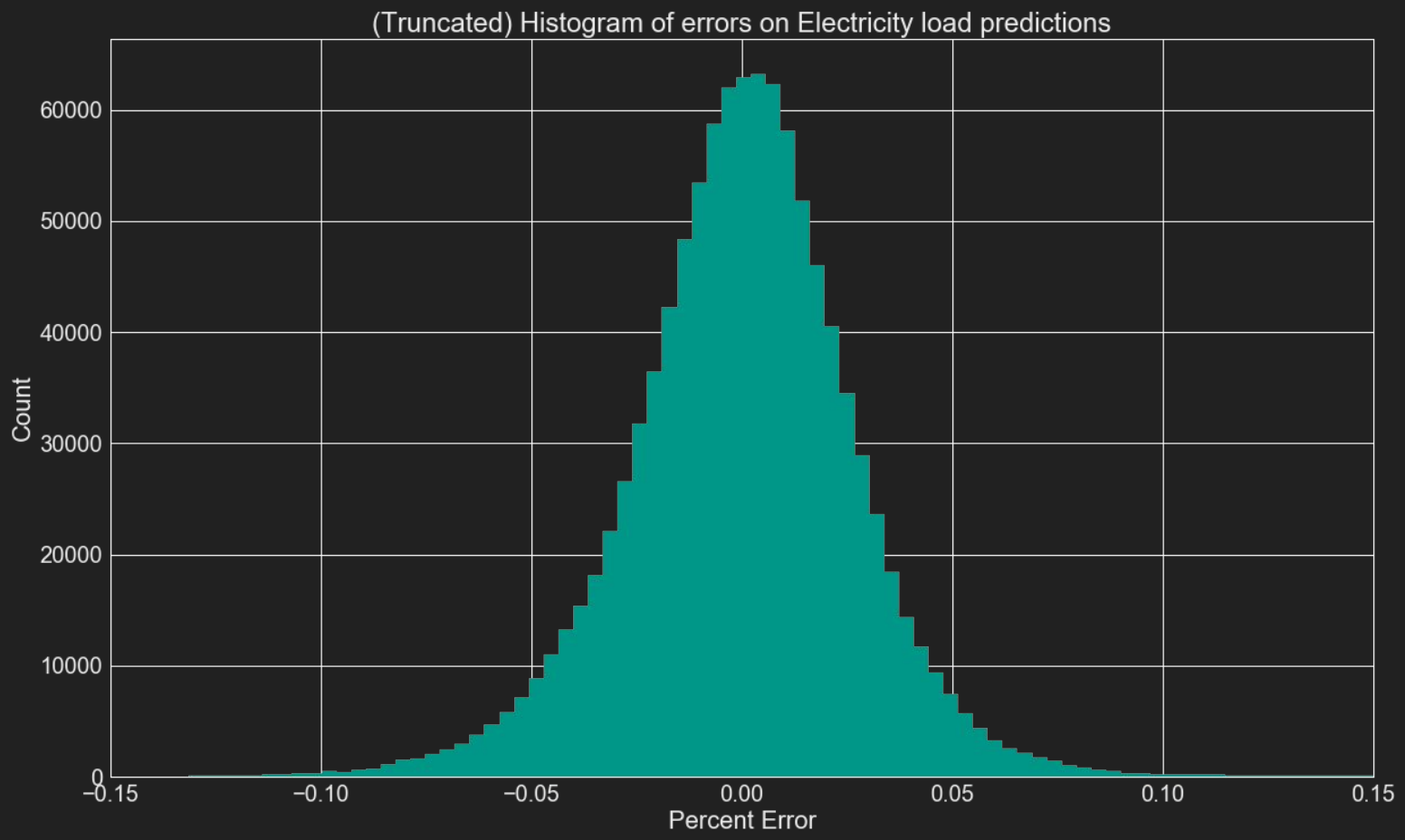
Input Variables

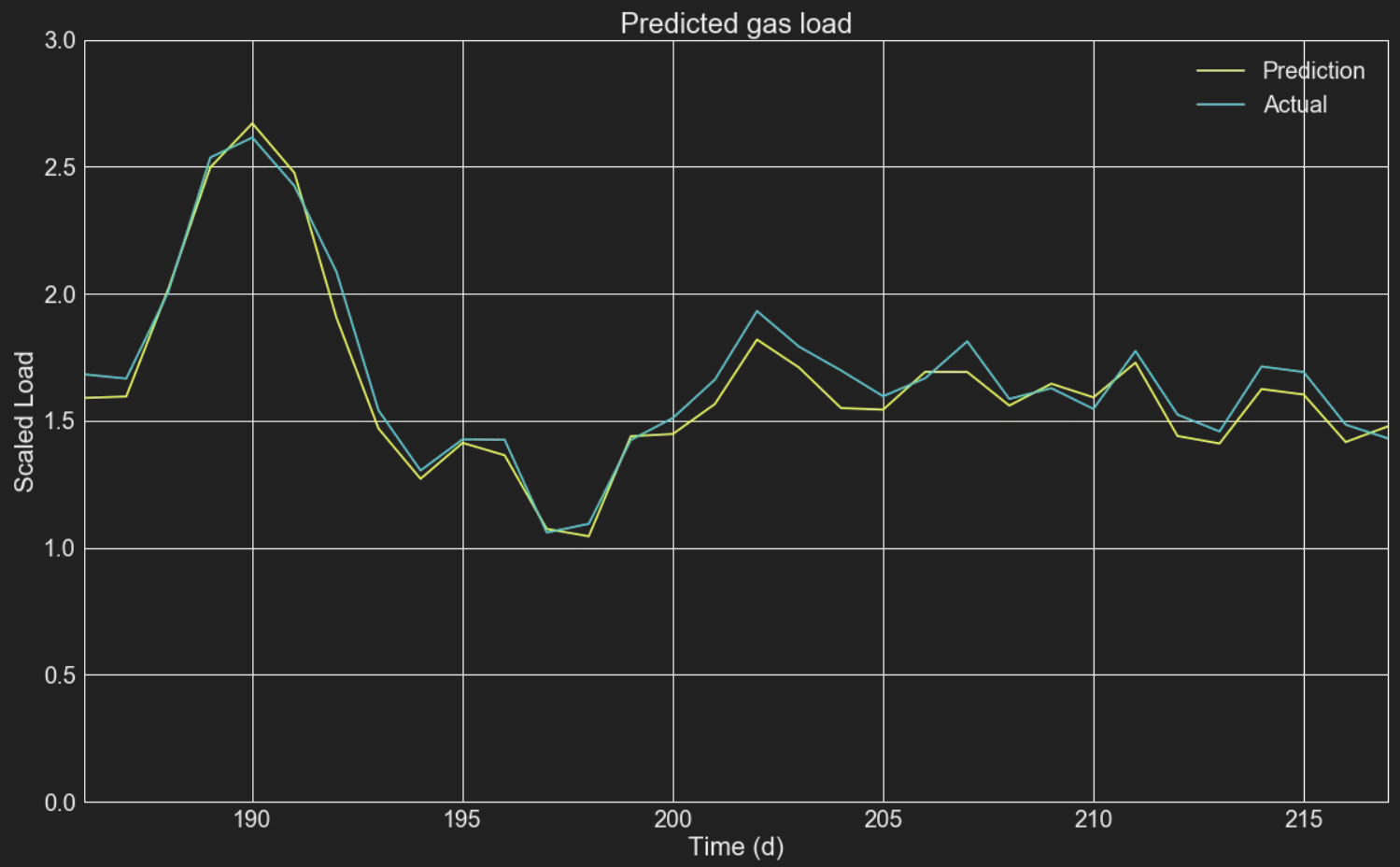
Electric: cos(hour), cos(dayofweek), cos(month), temperature, wind speed, cloud cover, is_holiday

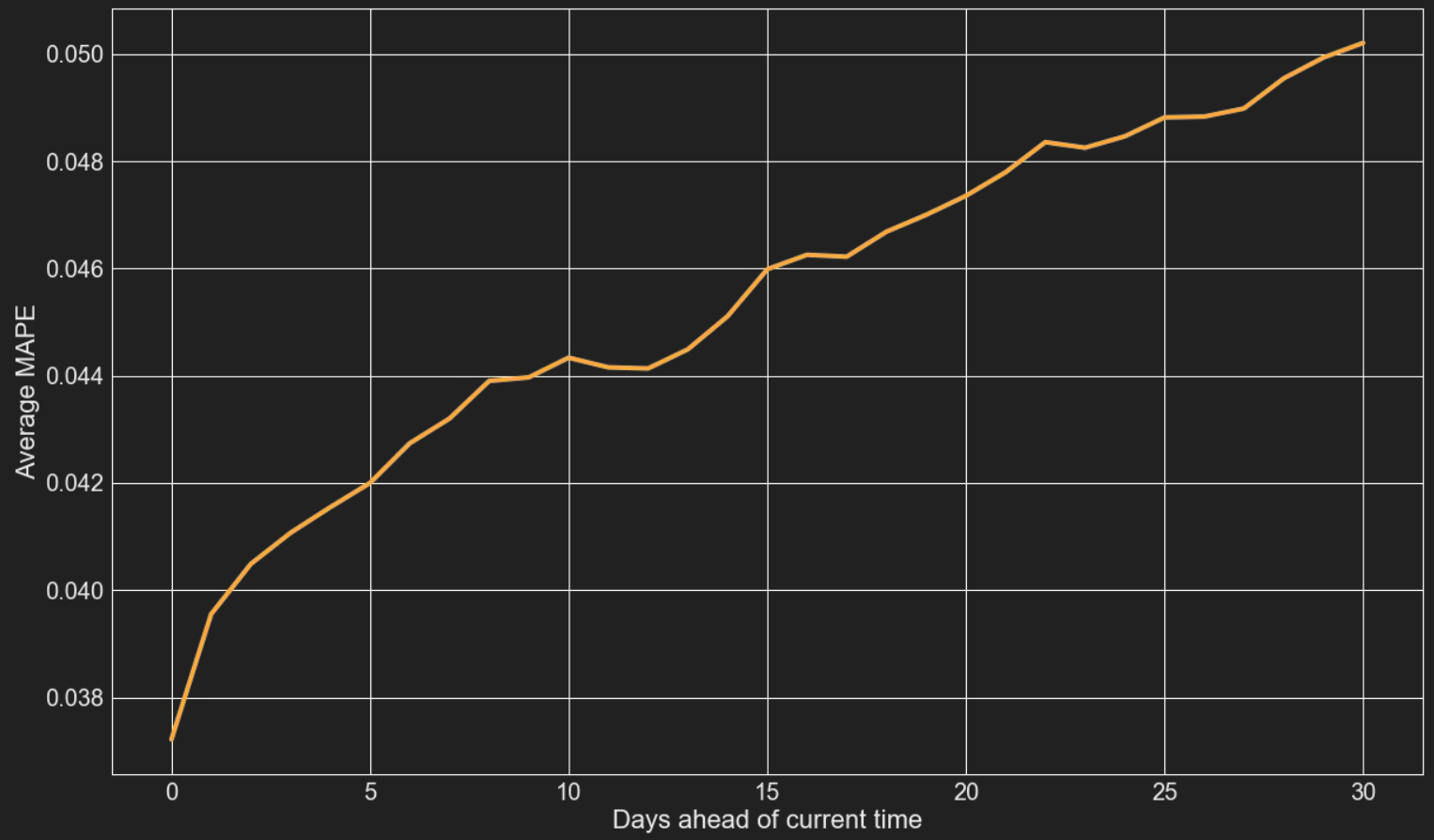
Gas: cos(dayofweek), cos(month), cos(dayofmonth), temperature, wind speed, is_holiday

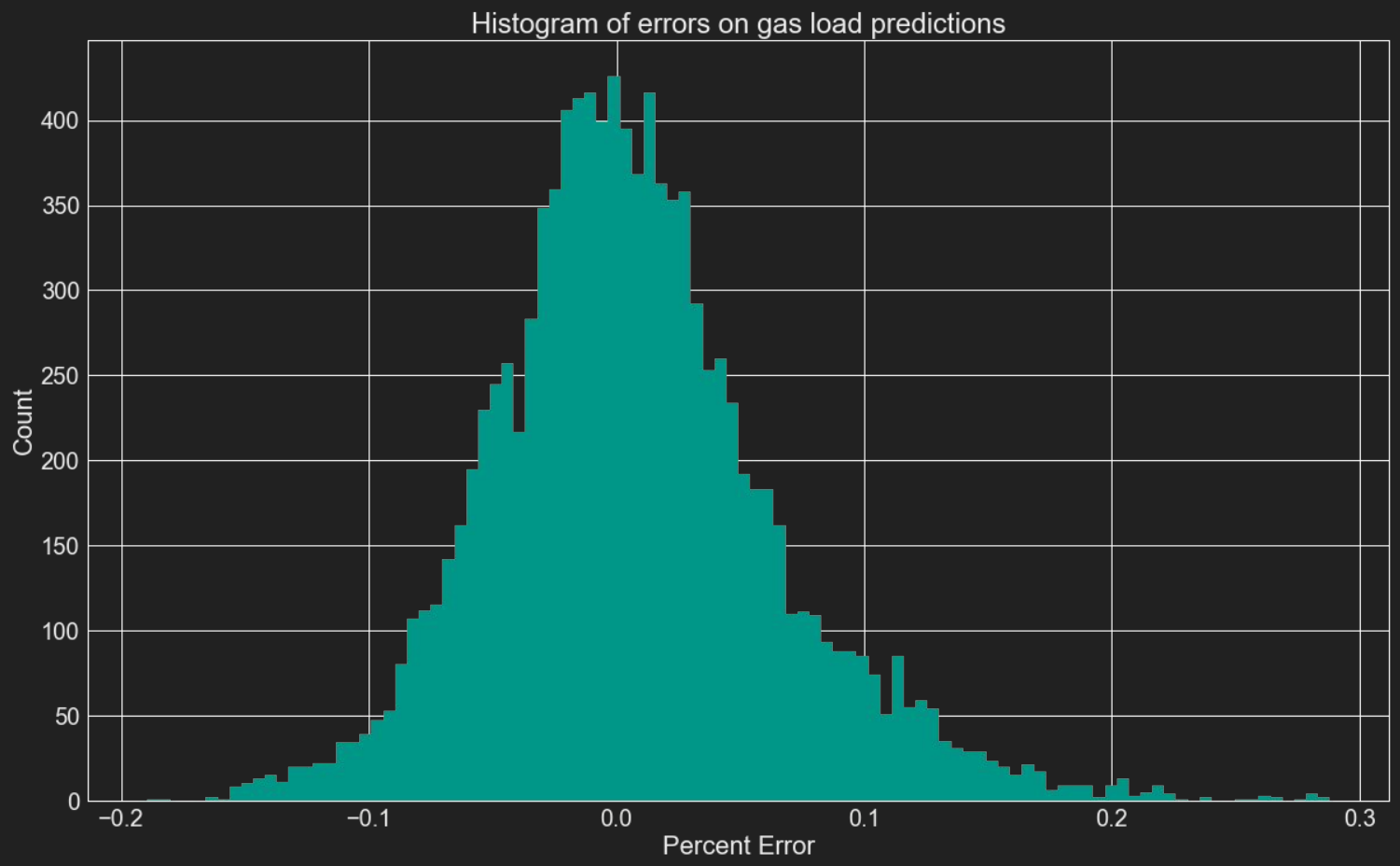

Predicted electric load
Prediction
Actual
Scaled Load
1.8
1.6
1.4
1.2
1.0
0.8
0.6
50
60
70
80
90
Time (h)


0.0225
0.0220
0.0215
0.0210
0.0205
0.0200
Average MAPE
0
10
20
30
40
Prediction Horizon (h)


(Truncated) Histogram of errors on Electricity load predictions
Count
60000
50000
40000
30000
20000
10000
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
Percent Error


Predicted gas load
Prediction
Actual
Scaled Load
Time (d)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
190
195
200
205
210
215


0.050
0.048
0.046
0.044
0.042
0.040
0.038
Average MAPE
0
5
10
15
20
25
30
Days ahead of current time


Histogram of errors on gas load predictions
Count
400
350
300
250
200
150
100
50
0
−0.2
−0.1
0.0
0.1
0.2
0.3
Percent Error

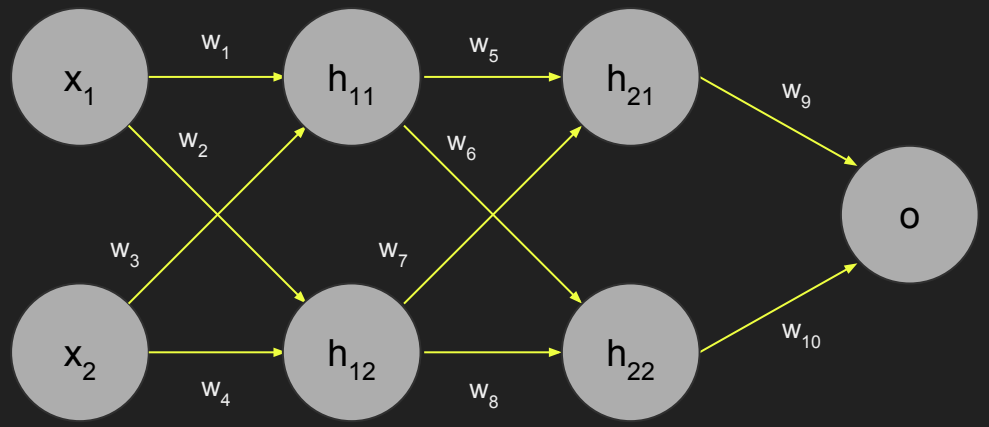
Conclusion / Future work

Autoregressive networks work better when there is less training data available

Try new methods:

- Attention is All You Need (Google Brain)
- Convolutional NNs
- Echo state networks

Backpropagation



$h_{11} = \sigma(v_{11})$, where $v_{11} = w_1x_1 + w_3x_2$. $\sigma'(u) = \sigma(u)*(1-\sigma(u))$.

References

Marino, D. L., Amarasinghe, K., & Manic, M. (2016). Building energy load forecasting using Deep Neural Networks. *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. doi:10.1109/iecon.2016.7793413

Karpathy, A. (2015). The unreasonable effectiveness of RNNs.

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi,A., and Jenssen, R. (2017). An overview and comparative analysis of recurrent neural networks for short term load forecasting. CoRR, abs/1705.04378.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017.

Cybenko, G. (1988). Approximation by superpositions of a sigmoidal function (Tech. Rep. No. 856). Urbana, IL: University of Illinois Urbana-Champaign Department of Electrical and Computer Engineering.