

# Python DataScience frameworks integration with PostgreSQL

@o\_courtin

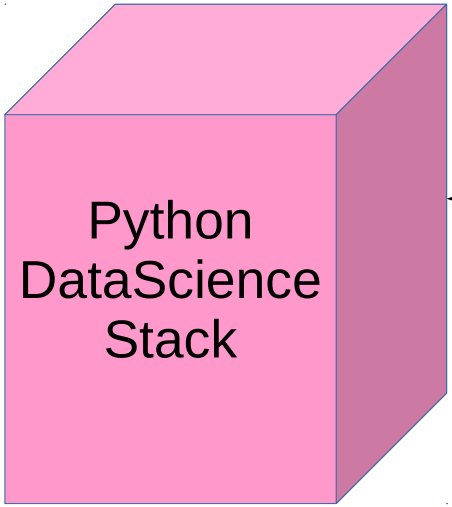


Deep Learning with PyTorch

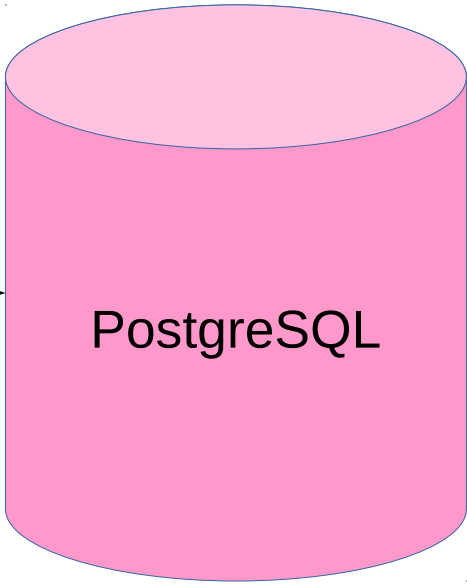
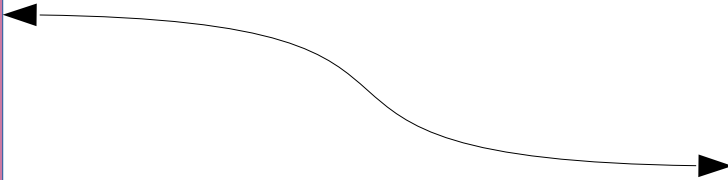


PyStan: The Python Interface to Stan





NumPy  
DataFrame



Pg DataTypes  
WKT Raster

```
In [1]: %matplotlib notebook

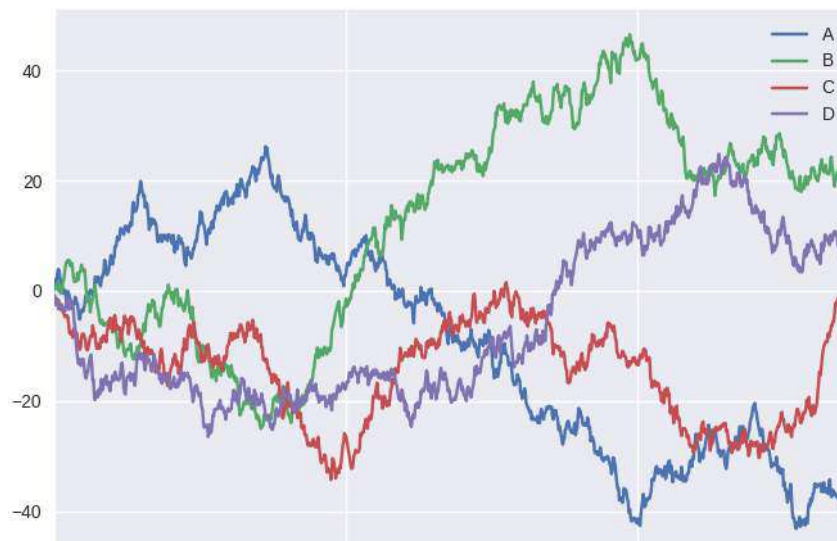
import pandas as pd
import numpy as np
import matplotlib

from matplotlib import pyplot as plt
import seaborn as sns

ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000', periods=1000))
ts = ts.cumsum()

df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index,
                  columns=['A', 'B', 'C', 'D'])
df = df.cumsum()
df.plot(); plt.legend(loc='best')
```

Figure 1



66 enregistrements

Aucun filtre actif.

### Filtres

Rechercher...

#### type

- turnstile 50
- area 16

#### classes 2

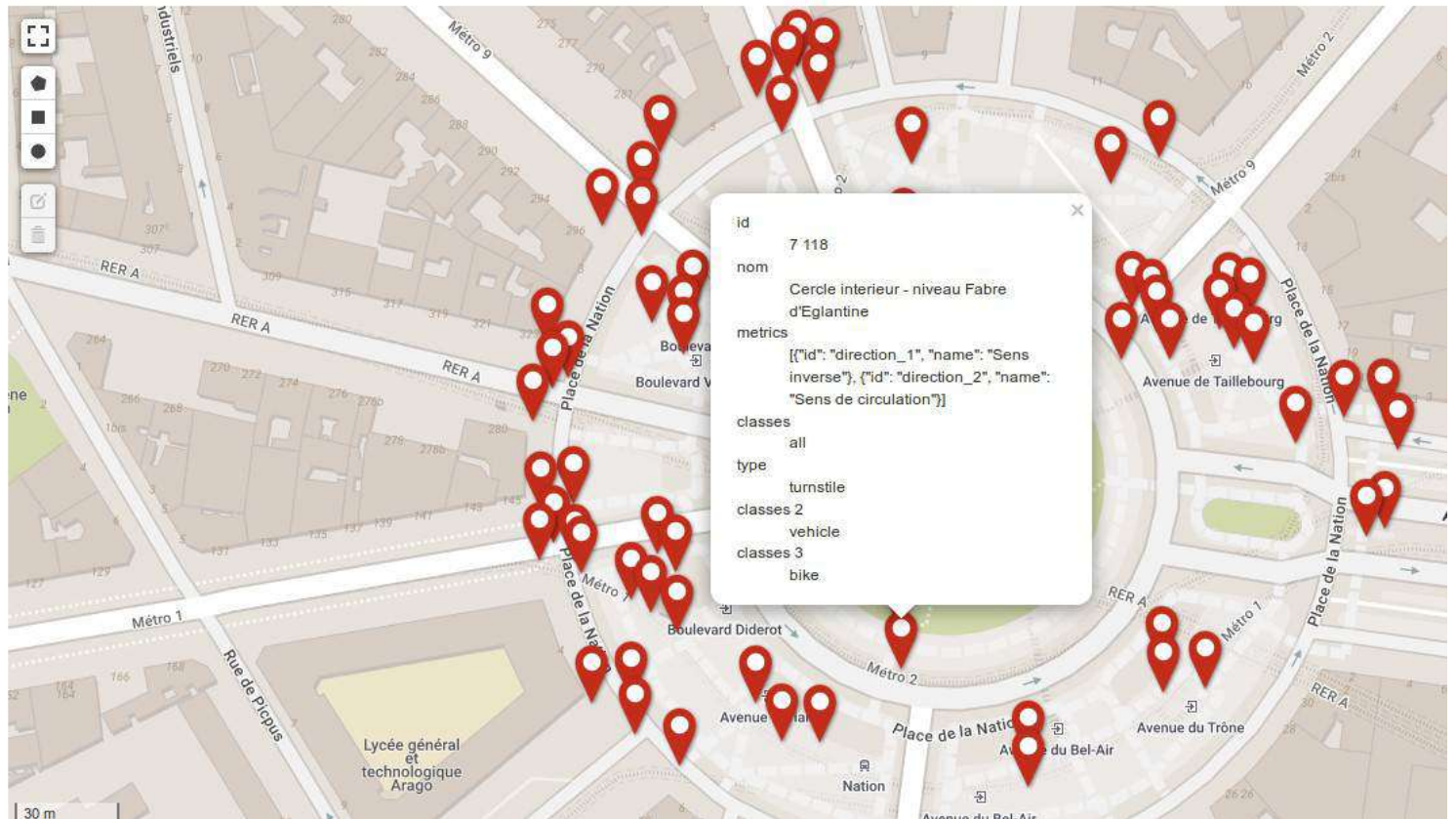
- ped 47
- vehicle 17
- bike 2

#### classes 3

- bike 21
- ped 1

## Place de la Nation - Points de mesure flux

Informations Tableau Carte Analyse Export API





 dlakata Fix typo in docs

ce5ed65 on Jul 25

3 contributors



176 lines (130 sloc) | 5.81 KB

Raw

Blame

History



## Loading CSVs into SQL Databases

When faced with the problem of loading a larger-than-RAM CSV into a SQL database from within Python, many people will jump to pandas. The workflow goes something like this:

```
>>> import sqlalchemy as sa
>>> import pandas as pd
>>> con = sa.create_engine('postgresql://localhost/db')
>>> chunks = pd.read_csv('filename.csv', chunksize=100000)
>>> for chunk in chunks:
...     chunk.to_sql(name='table', if_exists='append', con=con)
```

There is an unnecessary and very expensive amount of data conversion going on here. First we convert our CSV into an iterator of DataFrames, then those DataFrames are converted into Python data structures compatible with SQLAlchemy. Those Python objects then need to be serialized in a way that's compatible with the database they are being sent to. Before you know it, more time is spent converting data and serializing Python data structures than on reading data from disk.

```
In [1]: pg_uri = 'postgresql://o:xxx@127.0.0.1:5432/db'
```

```
In [2]: from odo import odo
odo('/tmp/iot.csv', pg_uri + '::iot')
```

```
/usr/local/lib/python3.5/dist-packages/odo/backends/pandas.py:94: FutureWarning: pandas.tslib
is deprecated and will be removed in a future version.
You can access NaTType as type(pandas.NaT)
@convert.register((pd.Timestamp, pd.Timedelta), (pd.tslib.NaTType, type(None)))
```

```
Out[2]: Table('iot', MetaData(bind=Engine(postgresql://o:***@127.0.0.1:5432/db)), Column('Metric', Text(), table=<iot>), Column('Value', FLOAT(), table=<iot>, nullable=False), Column('Value type', Text(), table=<iot>), Column('Host', BigInteger(), table=<iot>, nullable=False), Column('Class', Text(), table=<iot>), Column('Timestamp', DateTime(), table=<iot>), Column('data_direction_1', Text(), table=<iot>), Column('data_direction_2', Text(), table=<iot>), Column('data_measure_name1', Text(), table=<iot>), Column('data_measure_name2', Text(), table=<iot>), schema=None)
```

```
In [3]: !psql -c "\d iot" db
```

Column	Type	Modifiers
Metric	text	
Value	double precision	not null
Value type	text	
Host	bigint	not null
Class	text	
Timestamp	timestamp without time zone	
data_direction_1	text	
data_direction_2	text	
data_measure_name1	text	
data_measure_name2	text	

```
In [4]: !psql -c "SELECT count(*) \n\nFROM iot" db
```

```
count\n-----\n2198006\n(1 row)
```



```
In [5]: from sqlalchemy import create_engine
import pandas as pd

pg = create_engine(pg_uri)
```

```
In [6]: df = pd.read_sql_query("""
        SELECT "Value" v,
               "Timestamp" ts
        FROM iot
        ORDER BY ts
        """, pg)

print(df)|
```

	v	ts
0	24.0	2016-10-11 17:10:00
1	137.0	2016-10-11 17:10:00
2	372.0	2016-10-11 17:10:00
3	30.0	2016-10-11 17:10:00
4	251.0	2016-10-11 17:10:00
5	284.0	2016-10-11 17:10:00
6	46.0	2016-10-11 17:10:00

```
In [7]: np = df.values  
print(np)
```

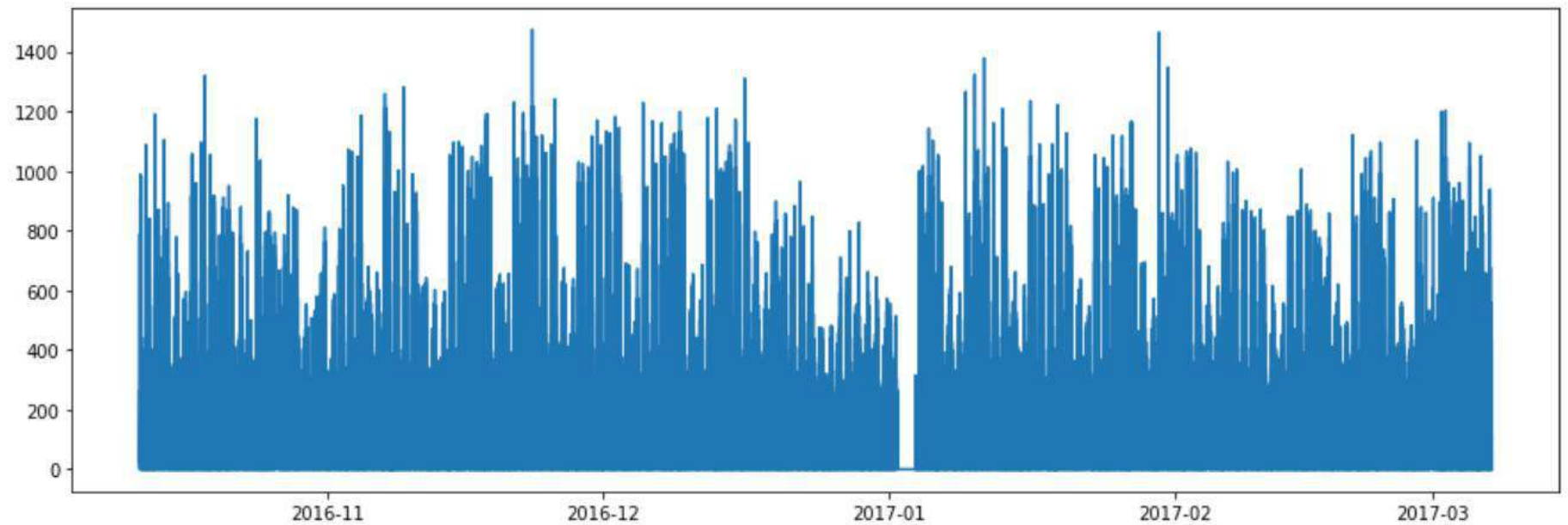
```
[[24.0 Timestamp('2016-10-11 17:10:00')]  
 [137.0 Timestamp('2016-10-11 17:10:00')]  
 [372.0 Timestamp('2016-10-11 17:10:00')]  
 ...  
 [3.0 Timestamp('2017-03-07 11:40:00')]  
 [0.0 Timestamp('2017-03-07 11:40:00')]  
 [0.0 Timestamp('2017-03-07 11:40:00')]]
```

```
In [8]: a_df = pd.DataFrame({'v':np[:,0], 'ts': np[:,1]})  
print (a_df)
```

	ts	v
0	2016-10-11 17:10:00	24
1	2016-10-11 17:10:00	137
2	2016-10-11 17:10:00	372
3	2016-10-11 17:10:00	30
4	2016-10-11 17:10:00	251
5	2016-10-11 17:10:00	284
6	2016-10-11 17:10:00	46

```
In [9]: import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(15, 5))  
plt.plot(np[:,1], np[:,0])  
plt.show()
```





quantopian / warp\_prism

Watch 32

Star 5

Fork 2

Code

Issues 0

Pull requests 1

Projects 0

Insights

# warp\_prism

Quickly move data from postgres to numpy or pandas.

## API

```
to_arrays(query, *, bind=None)
```

Run the query returning a the results as np.ndarrays.

```
to_dataframe(query, *, bind=None, null_values=None)
```

Run the query returning a the results as a pd.DataFrame.

[https://github.com/quantopian/warp\\_prism](https://github.com/quantopian/warp_prism)

```
In [10]: !psql -c 'CREATE TABLE foo AS (SELECT "Value" v, "Timestamp" ts FROM iot)' db
```

```
SELECT 2198006
```

```
In [11]: import warp_prism as wp  
from odo import resource
```

```
np = wp.to_arrays(resource(pg_uri + '::foo'))
```

```
print(np)
```

```
{'v': (array([ 7.,  8., 13., ..., 88., 43., 13.]), array([ True,  True,  True, ...,  True,  
 True,  True], dtype=bool)), 'ts': (array(['2016-11-22T07:20:00.000000', '2016-11-22T07:20:  
00.000000',  
      '2016-11-22T06:40:00.000000', ..., '2017-01-27T00:20:00.000000',  
      '2017-01-27T00:20:00.000000', '2017-01-27T00:20:00.000000'], dtype='datetime64[us]'), a  
rray([ True,  True,  True, ...,  True,  True,  True], dtype=bool))}
```



```
In [12]: import warp_prism as wp
         from odo import resource

         df = wp.to_dataframe(resource(pg_uri + '::foo'))
         print(df)
```

	v	ts
0	7.0	2016-11-22 07:20:00
1	8.0	2016-11-22 07:20:00
2	13.0	2016-11-22 06:40:00
3	10.0	2016-11-22 07:20:00
4	16.0	2016-11-22 06:40:00
5	59.0	2016-11-22 06:40:00
6	25.0	2016-11-22 06:40:00

```
CREATE OR REPLACE FUNCTION signal_correlate(a float[], b float[])  
RETURNS numeric  
AS $$
```

```
from scipy import signal  
import numpy as np
```

```
return np.argmax(signal.correlate(a, b)) - len(a)
```

```
$$ LANGUAGE plpythonu;
```

## Still TODO

### **warp\_prism:**

add plain SQL query support (and VIEW handling)

### **odo:**


schema handling in uri

release to package: <https://github.com/blaze/odo/pull/443>

### **asyncpkg / asyncio:**

Evaluate if/how could be use in this context



**Ben Hamner**  @benhamner · Nov 12

Easy parts of applying machine learning:

`.fit()`

`.predict()`

Hard parts:

`.clean()`

`.transform()`

`.get_data()`

`.frame_problem()`

`.debug()`

`.handle_nonstationarities()`

`.handle_missing_inputs()`



35



792

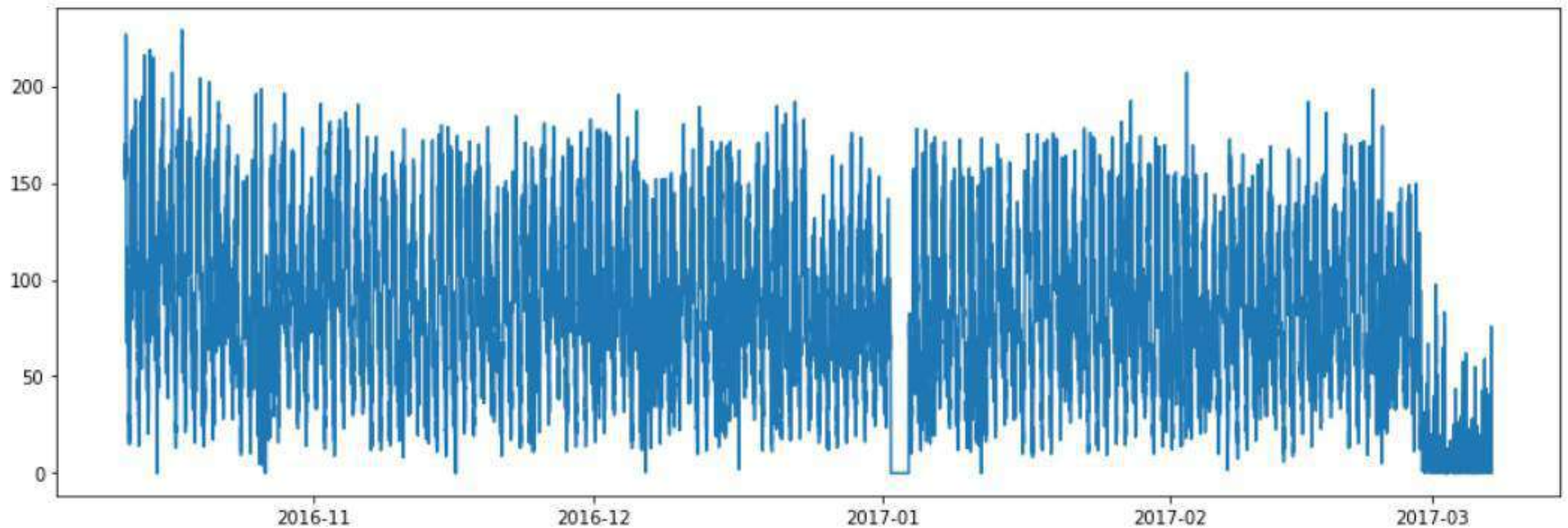


2.0K



```
df = pd.read_sql_query("""  
    WITH a AS (SELECT "Value" v, "Timestamp" ts, "Host" host FROM iot)  
    SELECT avg(v), ts  
    FROM a  
    WHERE host IN ('7062', '6196', '7118')  
    GROUP BY ts  
    ORDER BY ts  
    """, pg)  
data = df.values
```

```
import matplotlib.pyplot as plt  
plt.figure(figsize=(15, 5))  
plt.plot(data[:,1], data[:,0])  
plt.show()
```



```

when = pd.read_sql_query("""
WITH a AS (
    SELECT "Timestamp" ts, "Value" v, "Host" host
    FROM iot
),
b AS (
    SELECT ts, v, host,
    sum(v) OVER (PARTITION by host ORDER BY ts
    ROWS BETWEEN CURRENT ROW AND 20 FOLLOWING) as cv
    FROM a
    WHERE host = '7062'
    ORDER BY ts
)
SELECT ts, v, host
FROM b
WHERE cv = 0
ORDER BY ts
LIMIT 1

""", pg)
print(when.values)

```

```
[[Timestamp('2017-01-01 22:30:00') 0.0 7062]]
```



```
import numpy as np
import numpy.fft as fft
```

```
df = pd.read_sql_query("""
```

```
WITH a AS (
```

```
    SELECT "Timestamp" ts, "Value" v, "Host" host
    FROM iot
    WHERE "Timestamp" < '31/12/2016'
```

```
)
```

```
    SELECT avg(v), ts
    FROM a
    WHERE host IN ('7062', '6196', '7118')
    GROUP BY ts
    ORDER BY ts
```

```
""", pg)
```

```
data = df.values
```

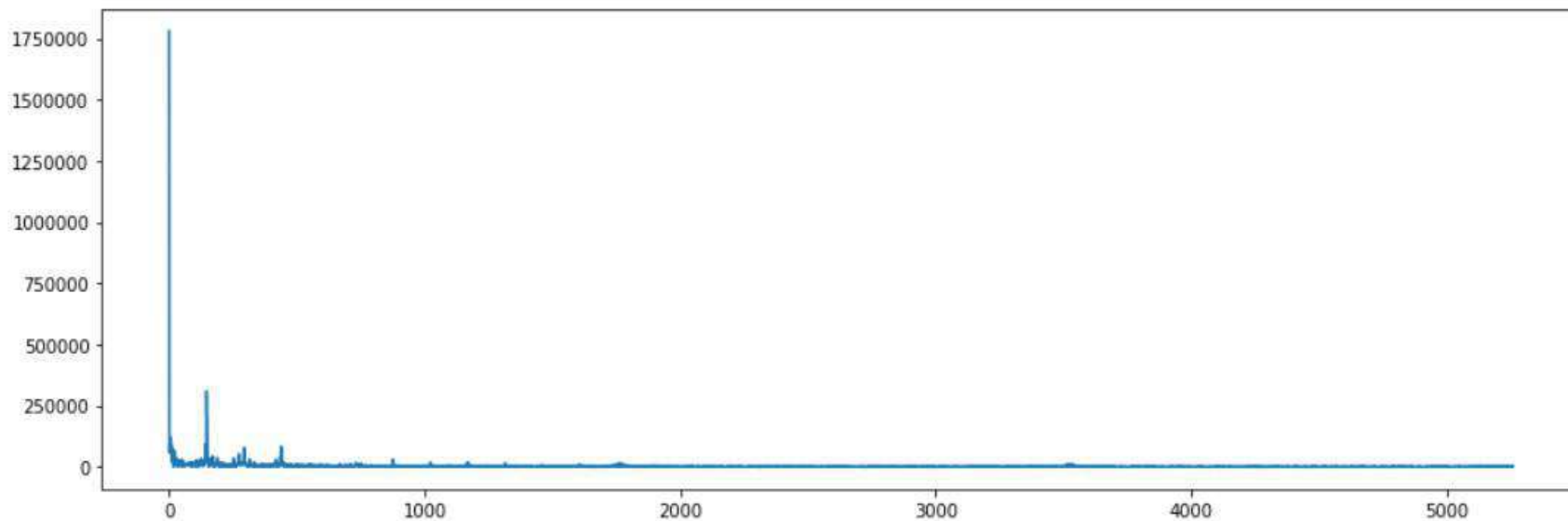
```
dft = np.abs(fft.rfft(data[:,0]))
```

```
N = round(len(dft)/2)
```

```
plt.figure(figsize=(15, 5))
```

```
plt.plot(dft[:N])
```

```
plt.show()
```



Quant

# Prophet: Automatic Forecasting Procedure

---

Prophet is a procedure for forecasting time series data. It is based on an additive model where non-linear trends are fit with yearly and weekly seasonality, plus holidays. It works best with daily periodicity data with at least one year of historical data. Prophet is robust to missing data, shifts in the trend, and large outliers.

Prophet is [open source software](#) released by Facebook's [Core Data Science team](#). It is available for download on [CRAN](#) and [PyPI](#).

<https://github.com/facebook/prophet>

# How Prophet works

At its core, the Prophet procedure is an **additive regression model** with four main components:

- A piecewise linear or logistic growth curve trend. Prophet automatically detects changes in trends by selecting changepoints from the data.
- A yearly seasonal component modeled using Fourier series.
- A weekly seasonal component using dummy variables.
- A user-provided list of important holidays.

<https://research.fb.com/prophet-forecasting-at-scale/>

```
import pandas as pd
```

```
df = pd.DataFrame({'y': data[:,0], 'ds': data[:,1]})
```

```
from fbprophet import Prophet
```

```
m = Prophet()
```

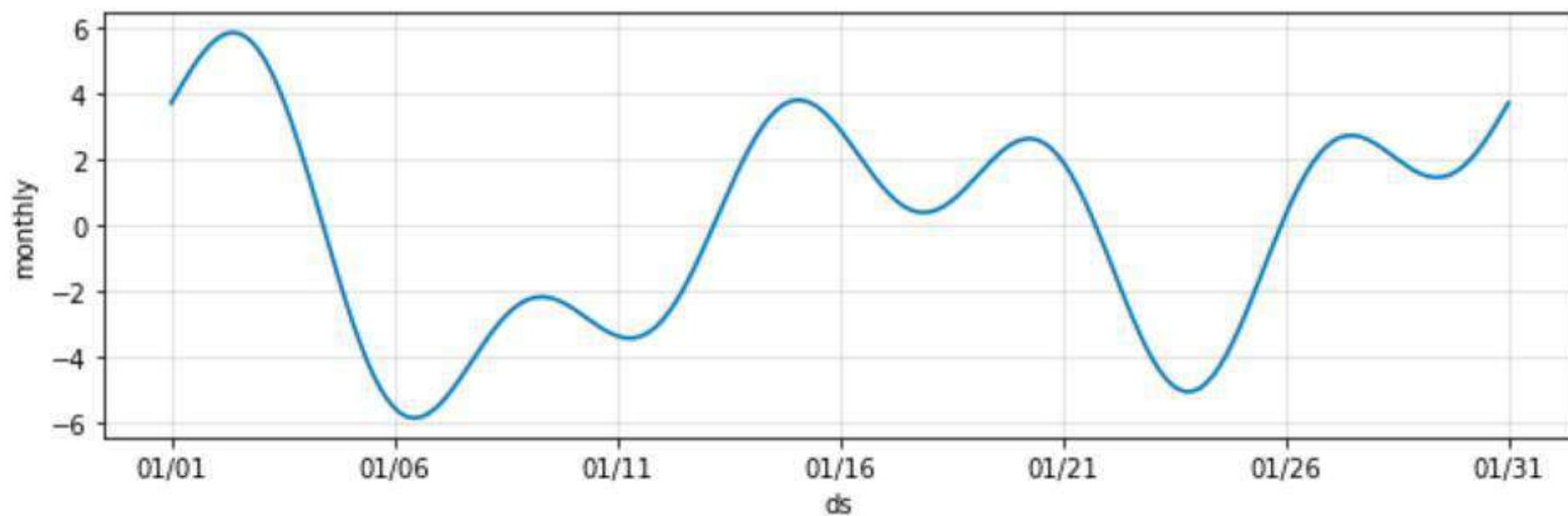
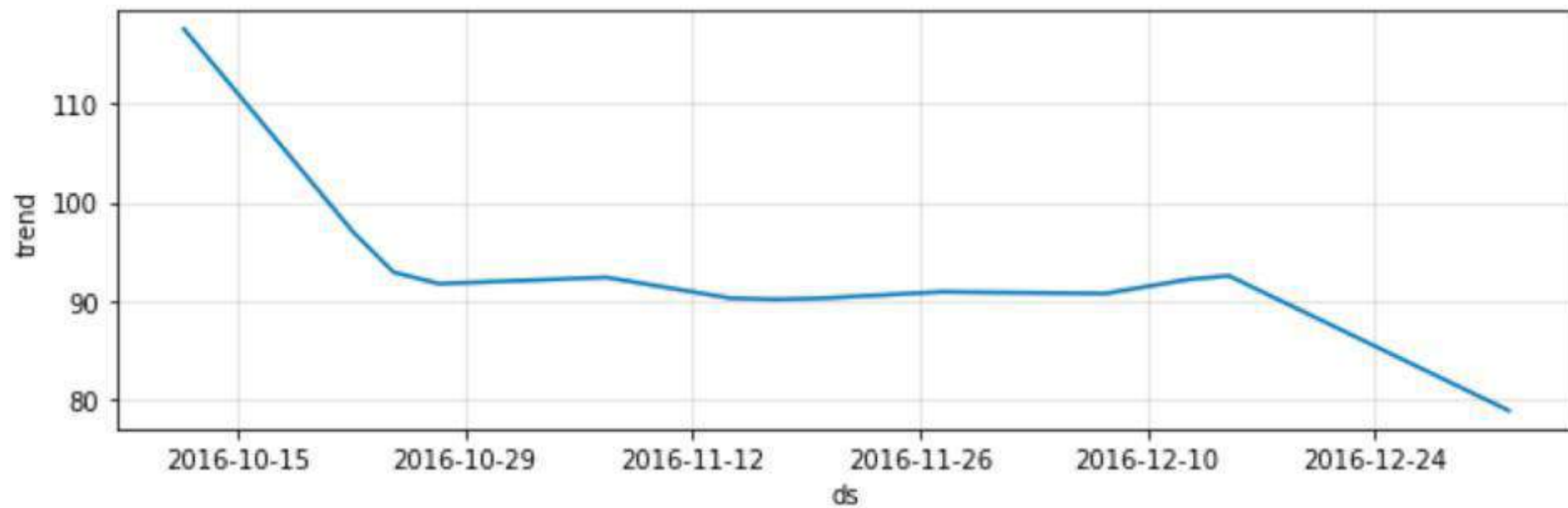
```
m.add_seasonality(name='monthly', period=30, fourier_order=5)
```

```
m.fit(df)
```

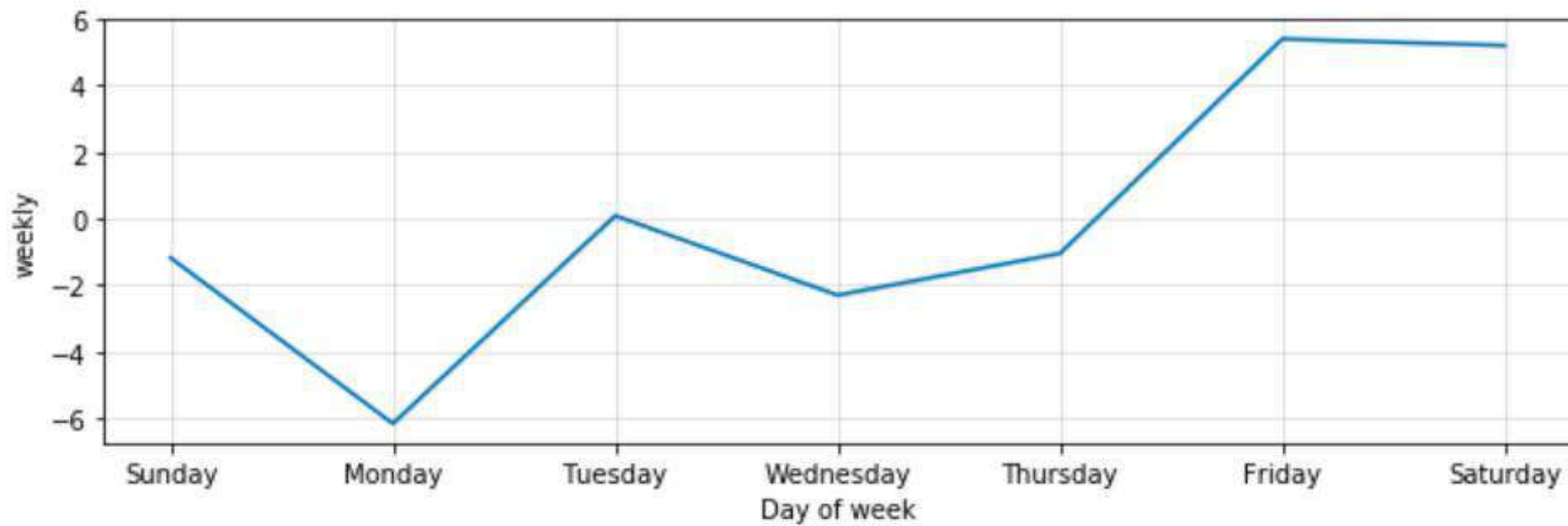
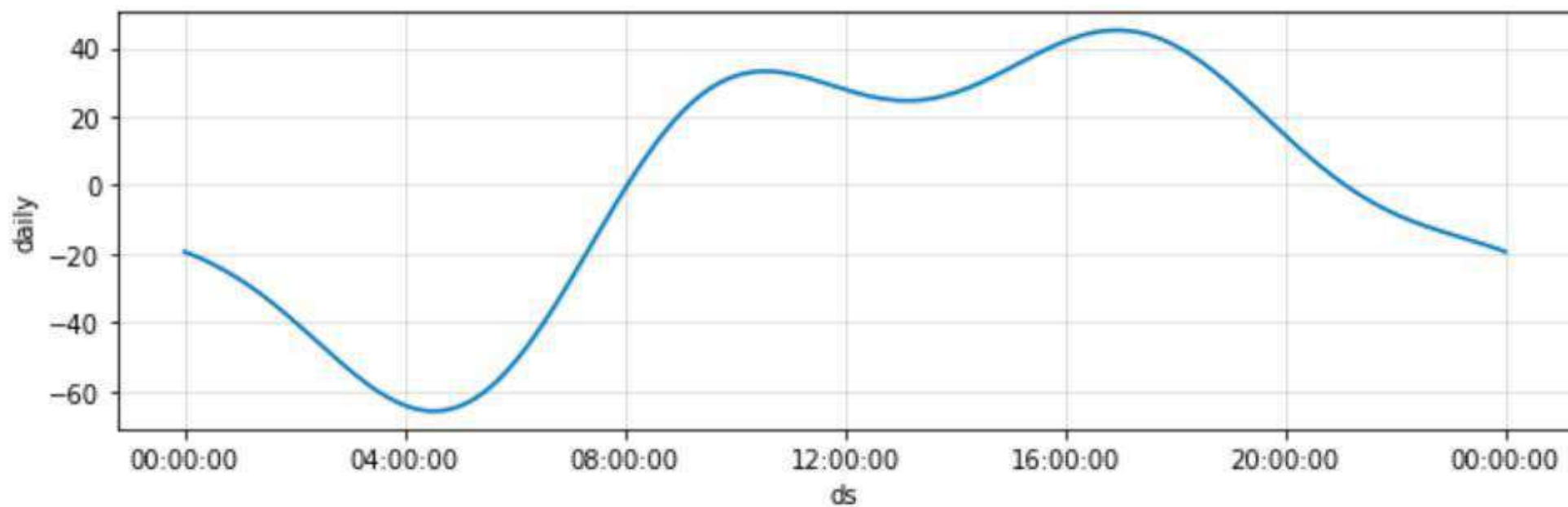
```
future = m.make_future_dataframe(periods=30, freq='H')
```

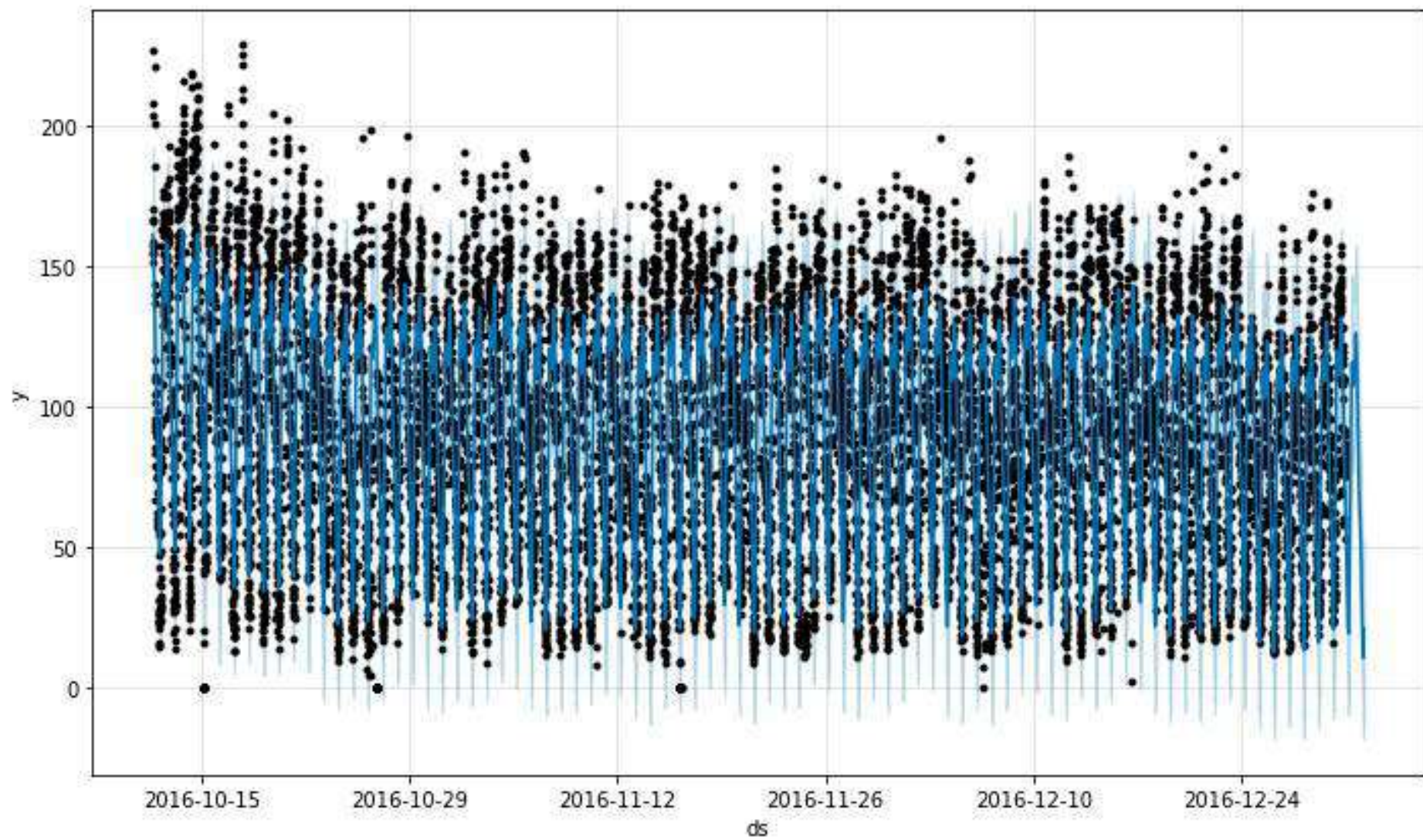
```
forecast = m.predict(future)
```

```
m.plot_components(forecast)
```









## To go further: RNN LSTM

<http://www.jakob-aungiers.com/articles/a/LSTM-Neural-Network-for-Time-Series-Prediction>

<http://www.deeplearningbook.org/contents/rnn.html>

<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

**M4COMPETITION**  
Forecast. Compete. Excel.

HOME ABOUT RULES DATES PRIZES EXPECTATIONS RESOURCES REGISTER NOW

# M<sup>4</sup> Competition

An open forecasting competition for individuals, teams and institutions to participate.



<https://www.m4.unic.ac.cy/>

NLP



# Industrial-Strength Natural Language Processing

IN PYTHON

## Fastest in the world

spaCy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. Independent research has confirmed that spaCy is the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

FACTS & FIGURES

## Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive. We like to think of spaCy as the Ruby on Rails of Natural Language Processing.

GET STARTED

## Deep learning

spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with TensorFlow, PyTorch, scikit-learn, Gensim and the rest of Python's awesome AI ecosystem. With spaCy, you can easily construct linguistically sophisticated statistical models for a variety of NLP problems.

READ MORE

<https://spacy.io/>



```
In [1]: from sqlalchemy import create_engine
import pandas as pd

pg_uri = 'postgresql://o:xxx@127.0.0.1:5432/db'
pg = create_engine(pg_uri)

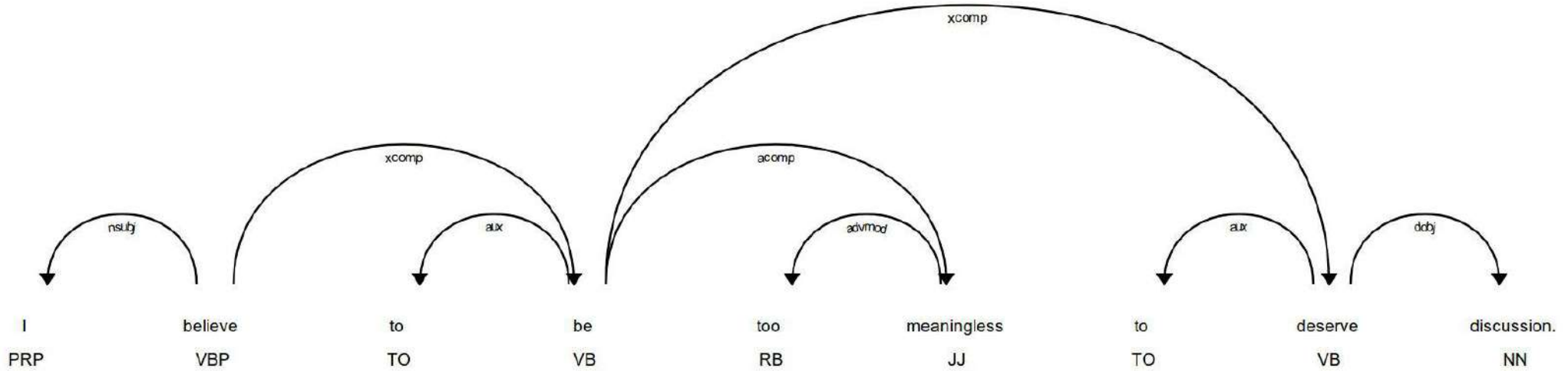
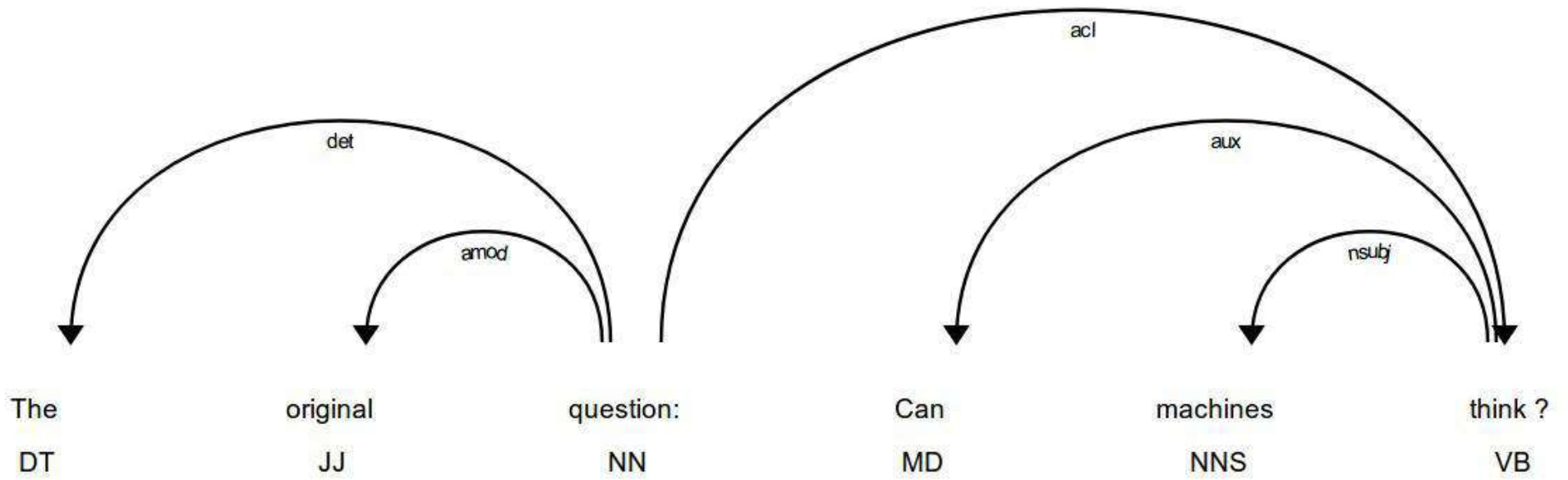
df = pd.read_sql_query("""
    SELECT
        'The original question: Can machines think ? I believe to be too meaningless to deserve discussion.'
        -- Turing Quote
    """, pg)

print(df.values[0][0])
```

The original question: Can machines think ? I believe to be too meaningless to deserve discussion.

```
In [2]: import spacy
from spacy import displacy

nlp = spacy.load('en')
doc = nlp(df.values[0][0])
displacy.serve(doc, style='dep')
```



## Machine Comprehension

Machine Comprehension (MC) answers natural language questions by selecting an answer span within an evidence text. The AllenNLP toolkit provides the following MC visualization, which can be used for any MC model in AllenNLP. This page demonstrates a reimplementation of **BIDAF (Seo et al, 2017)**, or Bi-Directional Attention Flow, a widely used MC baseline that achieved state-of-the-art accuracies on [the SQuAD dataset](#) (Wikipedia sentences) in early 2017.

Enter text or

### Passage

Robotics is an interdisciplinary branch of engineering and science that includes mechanical engineering, electrical engineering, computer science, and others. Robotics deals with the design, construction, operation, and use of robots, as well as computer systems for their control, sensory feedback, and information processing. These technologies are used to develop machines that can substitute for humans. Robots can be used in any situation and for any purpose, but today many are used in dangerous environments (including bomb detection and de-activation), manufacturing

### Question

What do robots that resemble humans attempt to do?

RUN >

### Answer

replicate walking, lifting, speech, cognition

### Passage Context

Robotics is an interdisciplinary branch of engineering and science that includes mechanical engineering, electrical engineering, computer science, and others. Robotics deals with the design, construction, operation, and use of robots, as well as computer systems for their control, sensory feedback, and information processing. These technologies are used to develop machines that can substitute for humans. Robots can be used in any situation and for any purpose, but today many are used in dangerous environments (including bomb detection and de-activation), manufacturing processes, or where humans cannot survive. Robots can take on any form but some are made to resemble humans in appearance. This is said to help in the acceptance of a robot in certain replicative behaviors usually performed by people. Such robots attempt to **replicate walking, lifting, speech, cognition**, and basically anything a human can do.

<http://demo.allennlp.org/machine-comprehension>

Vision

WHEN A USER TAKES A PHOTO,  
THE APP SHOULD CHECK WHETHER  
THEY'RE IN A NATIONAL PARK...

SURE, EASY GIS LOOKUP.  
GIMME A FEW HOURS.

... AND CHECK WHETHER  
THE PHOTO IS OF A BIRD.

I'LL NEED A RESEARCH  
TEAM AND FIVE YEARS.



IN CS, IT CAN BE HARD TO EXPLAIN  
THE DIFFERENCE BETWEEN THE EASY  
AND THE VIRTUALLY IMPOSSIBLE.

# WKB Raster

---

Read WKB rasters to Numpy arrays.

## Docs

```
wkb_raster.read_wkb_raster(wkb)
```

## Parameters

- `wkb` – file-like object. Binary raster in WKB format.

---

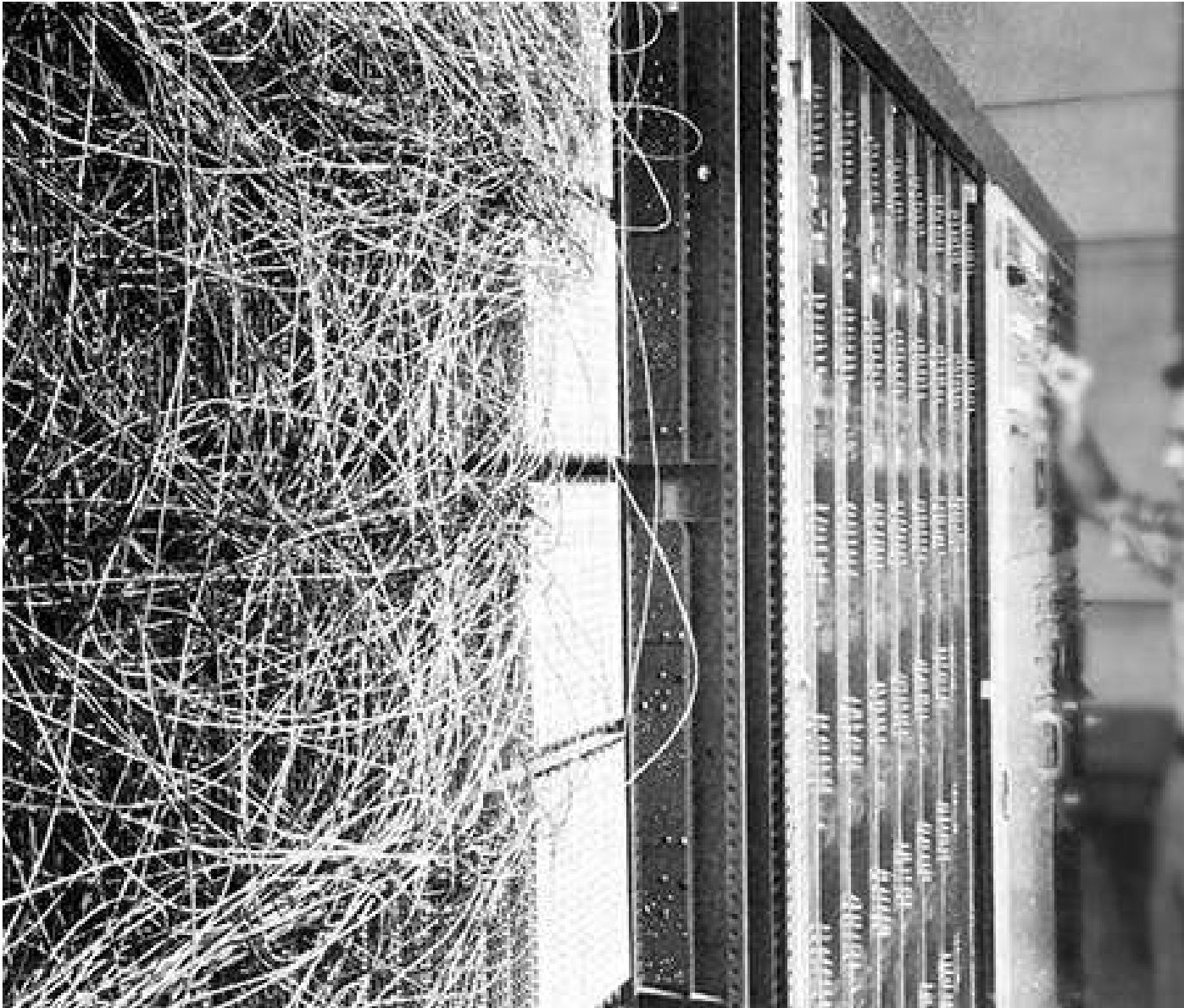
With WKB from PostGIS Raster. Use [ST\\_AsBinary](#) to return the WKB representation of the raster.

```
SELECT ST_AsBinary(rast) FROM rasters;
```

Wrap the binary buffer in `cStringIO.StringIO` :

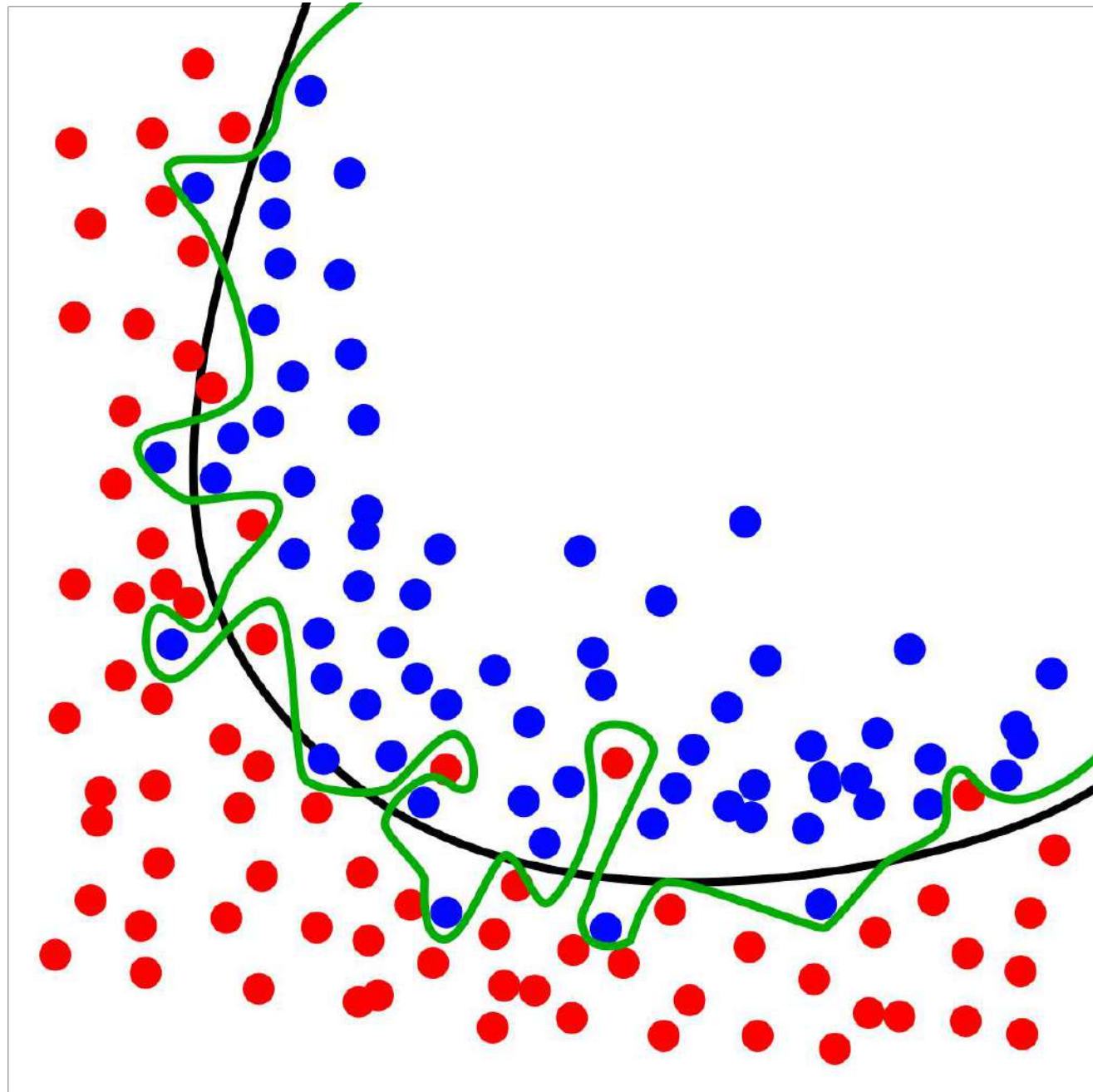
```
from cStringIO import StringIO
from wkb_raster import read_wkb_raster

raster = read_wkb_raster(StringIO(buf))
raster['bands'][0]
```



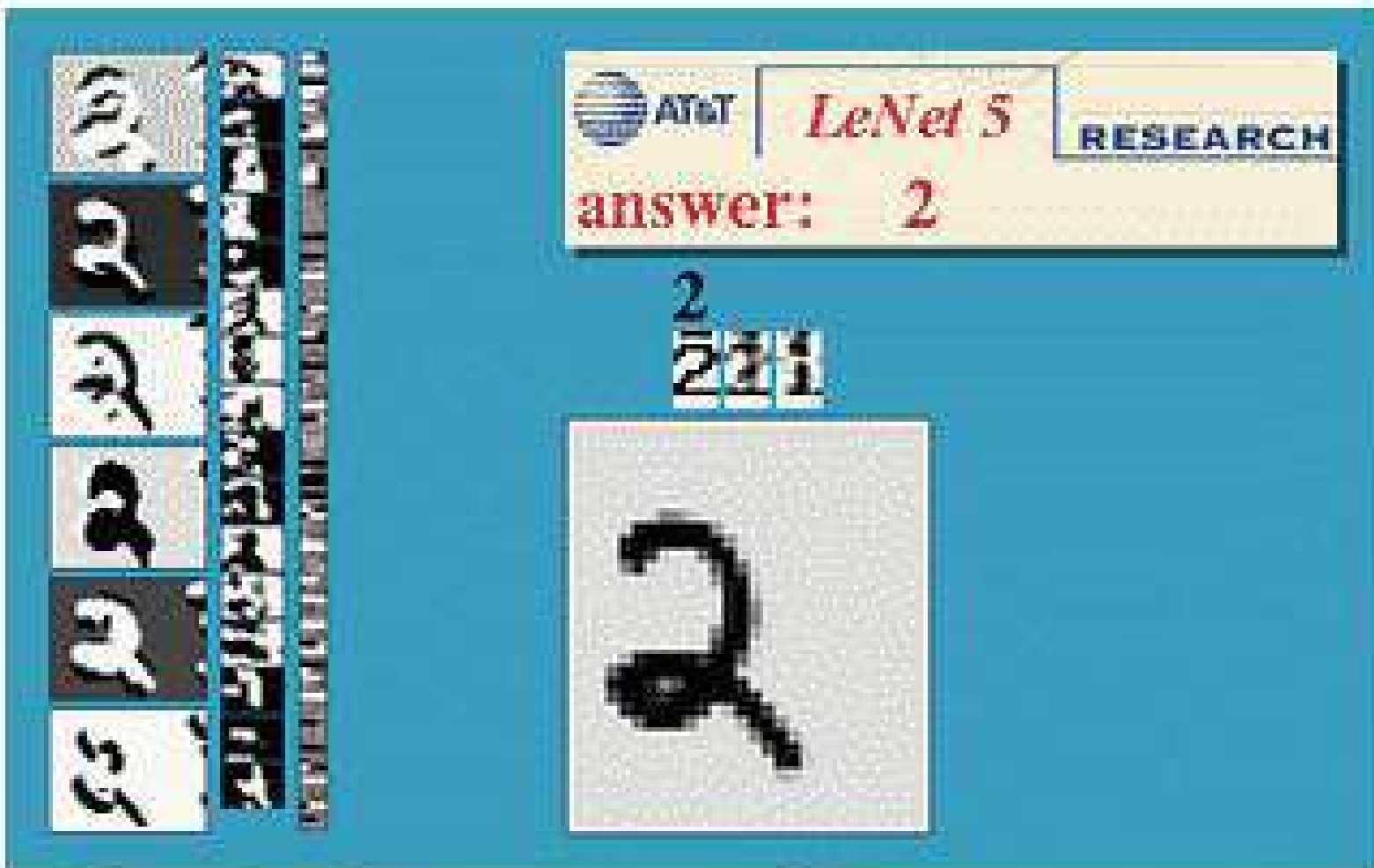
Rosenblatt, The Perceptron (1958)

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.335.3398&rep=rep1&type=pdf>

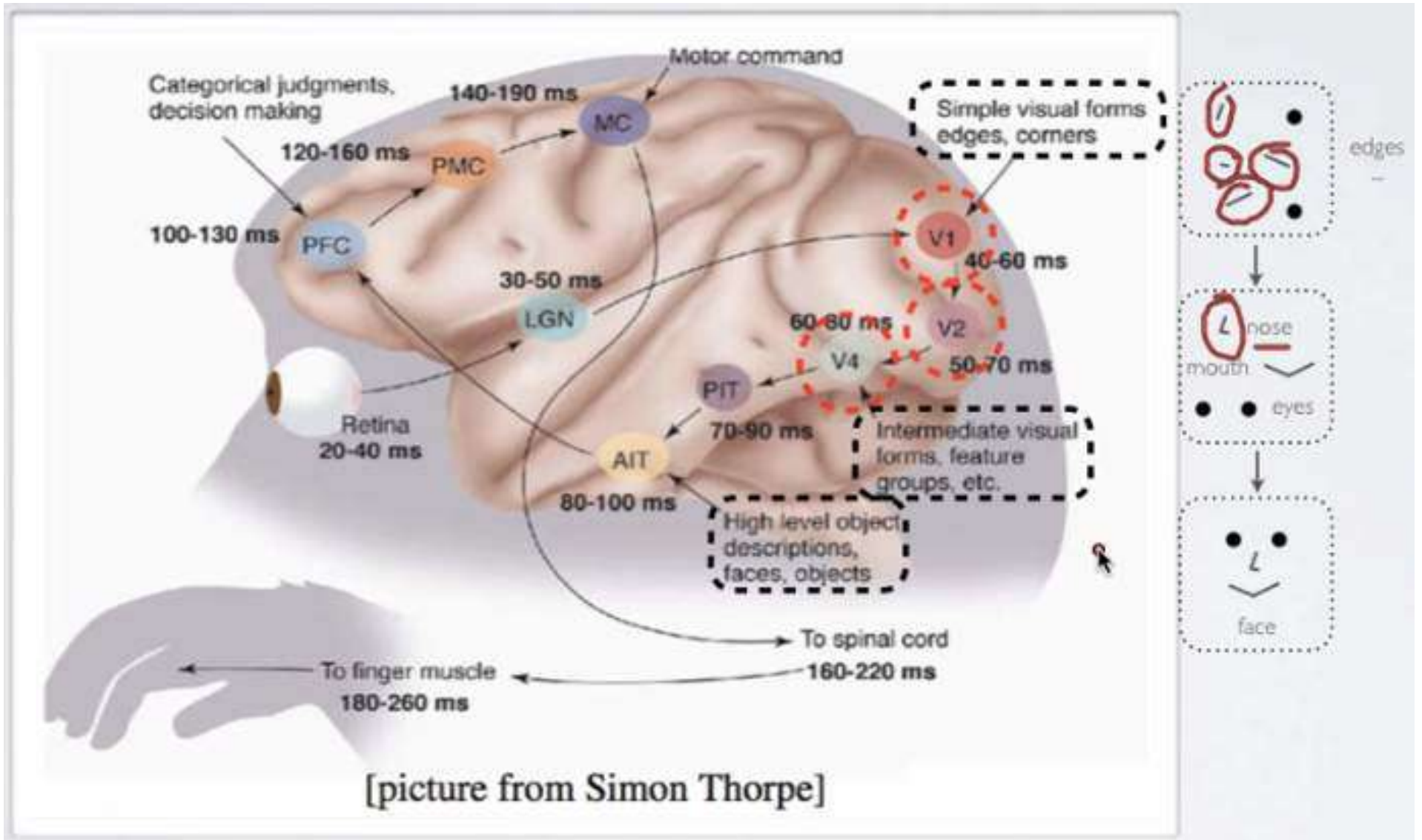


<https://en.wikipedia.org/wiki/Overfitting>

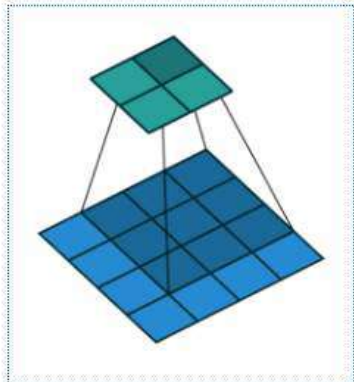
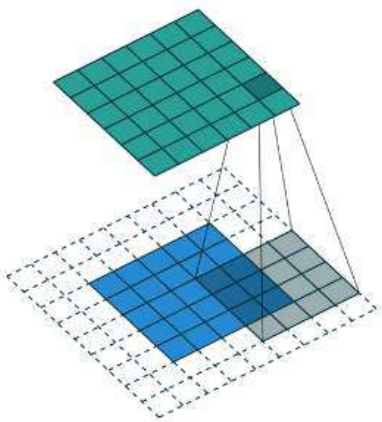
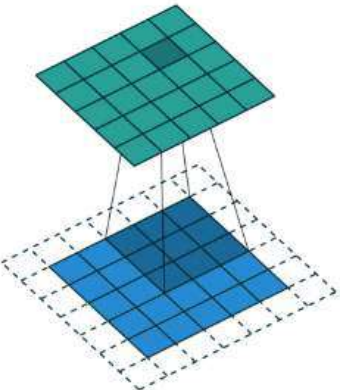
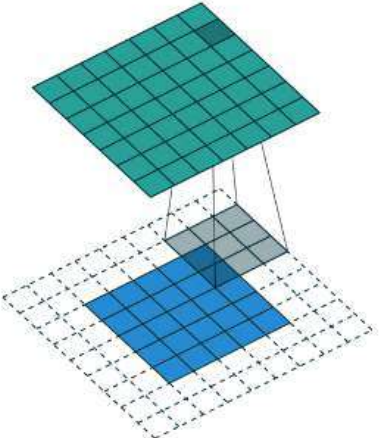
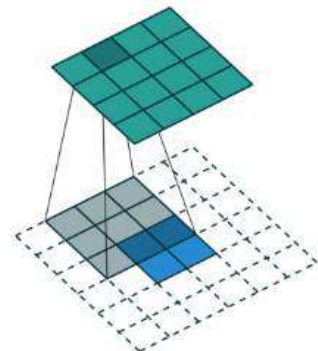
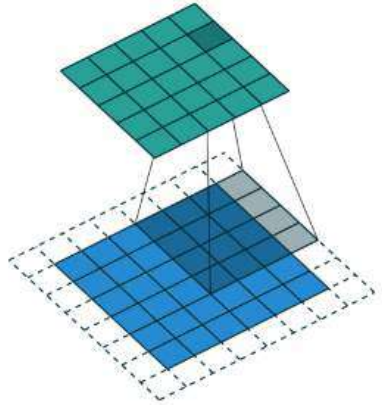
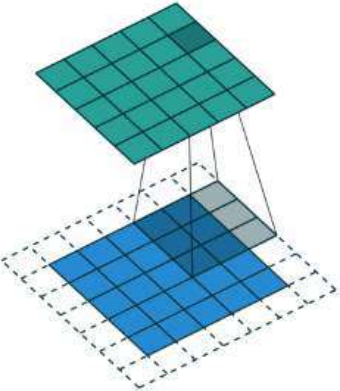
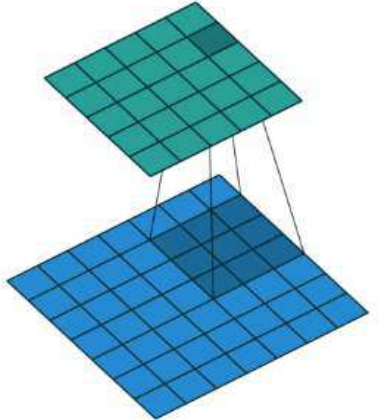




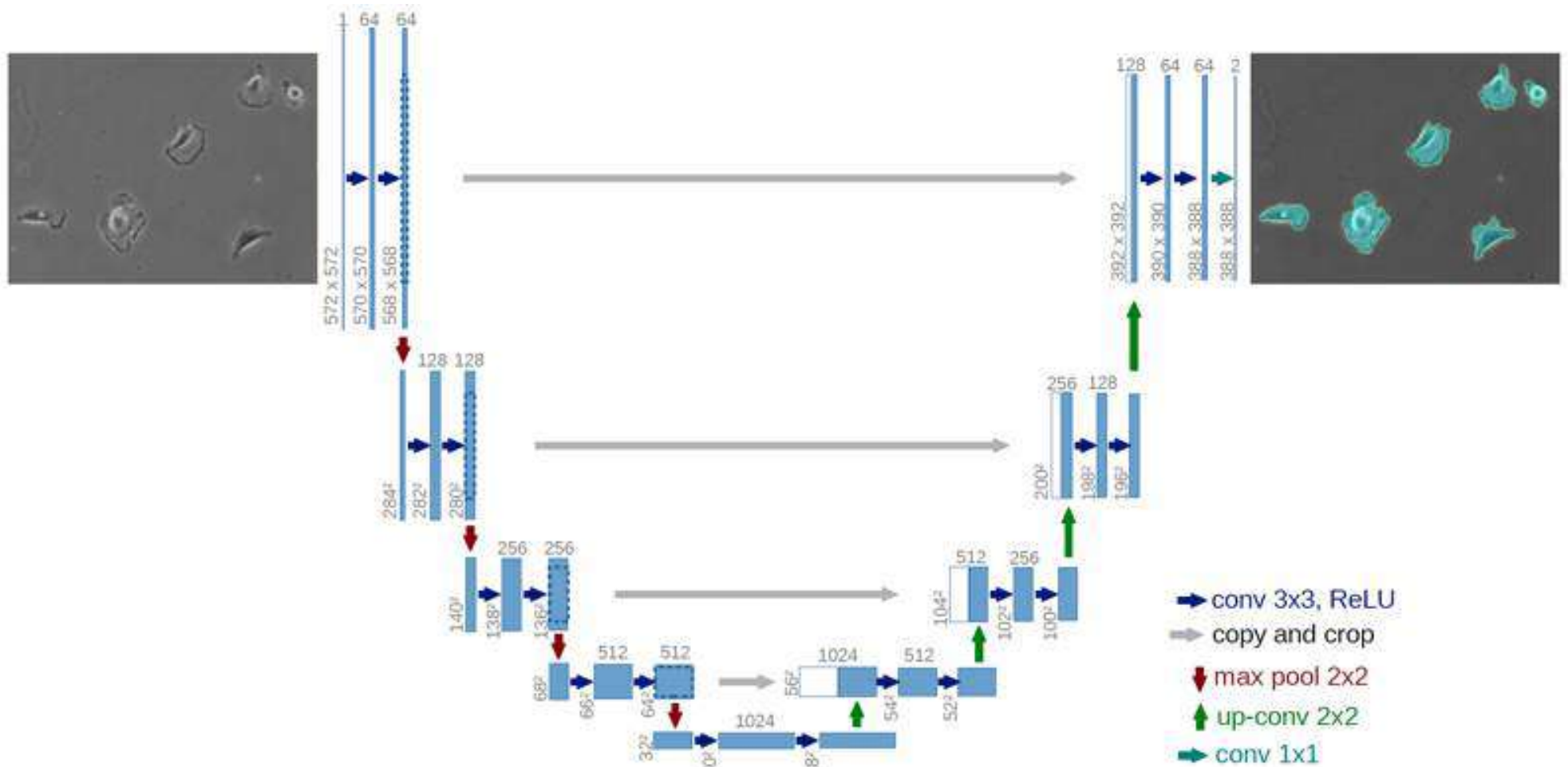
LeCun – LeNet 5 (1998)  
<http://yann.lecun.com/exdb/lenet/>



# Convolution animations

			
No padding, no strides	Arbitrary padding, no strides	Half padding, no strides	Full padding, no strides
			
No padding, no strides, transposed	Arbitrary padding, no strides, transposed	Half padding, no strides, transposed	Full padding, no strides, transposed

[https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic)



## U-Net: Convolutional Networks for Biomedical Image Segmentation

<https://arxiv.org/abs/1505.04597>

■ SOFTWARE DEVELOPMENT

# Deep Learning for Semantic Segmentation of Aerial Imagery

By Rob Emanuele on May 30th, 2017

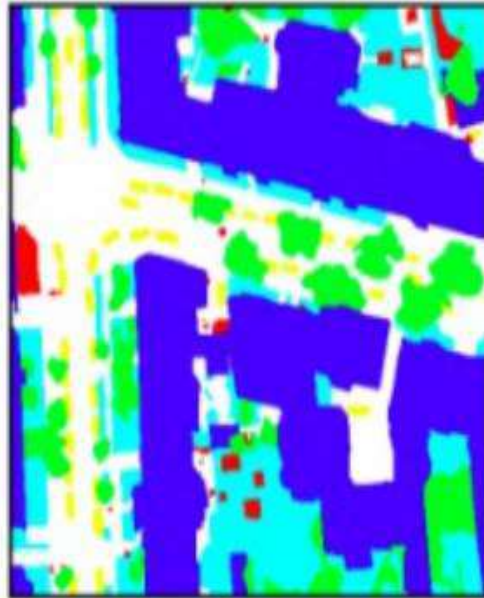
<https://www.azavea.com/blog/2017/05/30/deep-learning-on-aerial-imagery/>



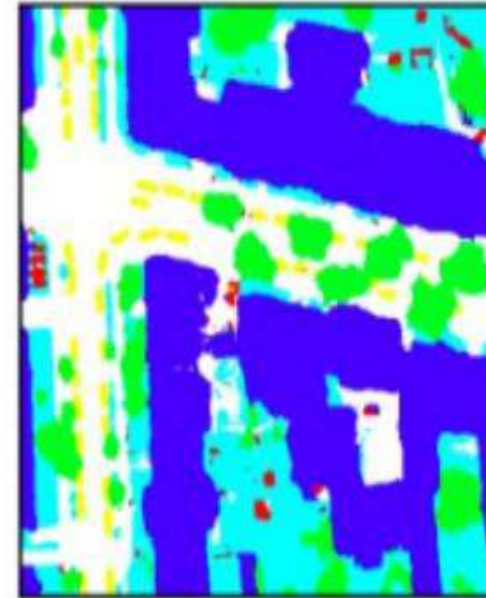
RGB



Ground Truth



Prediction



- Impervious
- Building
- Low vegetation
- Tree
- Car
- Clutter

	Overall	Impervious	Building	Low Vegetation	Tree	Car	Clutter
Validation	85.8	89.1	91.8	82.0	83.3	93.7	63.2
Test	89.2	91.4	96.1	86.1	86.6	93.3	46.8

```
1.  # The number of output labels
2.  nb_labels = 6
3.
4.  # The dimensions of the input images
5.  nb_rows = 256
6.  nb_cols = 256
7.
8.  # A ResNet model with weights from training on ImageNet. This will
9.  # be adapted via graph surgery into an FCN.
10. base_model = ResNet50(
11.     include_top=False, weights='imagenet', input_tensor=input_tensor)
12.
13. # Get final 32x32, 16x16, and 8x8 layers in the original
14. # ResNet by that layers's name.
15. x32 = base_model.get_layer('final_32').output
16. x16 = base_model.get_layer('final_16').output
17. x8 = base_model.get_layer('final_x8').output
18.
19. # Compress each skip connection so it has nb_labels channels.
20. c32 = Convolution2D(nb_labels, (1, 1))(x32)
21. c16 = Convolution2D(nb_labels, (1, 1))(x16)
22. c8 = Convolution2D(nb_labels, (1, 1))(x8)
23.
```

```
23.
24. # Resize each compressed skip connection using bilinear interpolation.
25. # This operation isn't built into Keras, so we use a LambdaLayer
26. # which allows calling a Tensorflow operation.
27. def resize_bilinear(images):
28.     return tf.image.resize_bilinear(images, [nb_rows, nb_cols])
29.
30. r32 = Lambda(resize_bilinear)(c32)
31. r16 = Lambda(resize_bilinear)(c16)
32. r8 = Lambda(resize_bilinear)(c8)
33.
34. # Merge the three layers together using summation.
35. m = Add()([r32, r16, r8])
36.
37. # Add softmax layer to get probabilities as output. We need to reshape
38. # and then un-reshape because Keras expects input to softmax to
39. # be 2D.
40. x = Reshape((nb_rows * nb_cols, nb_labels))(m)
41. x = Activation('softmax')(x)
42. x = Reshape((nb_rows, nb_cols, nb_labels))(x)
43.
44. fcn_model = Model(input=input_tensor, output=x)
```





## Dstl Satellite Imagery Feature Detection

Can you train an eye in the sky?

\$100,000 · 419 teams · 8 months ago

[Overview](#)

[Data](#)

[Kernels](#)

[Discussion](#)

[Leaderboard](#)

[Rules](#)

### Overview

[Description](#)

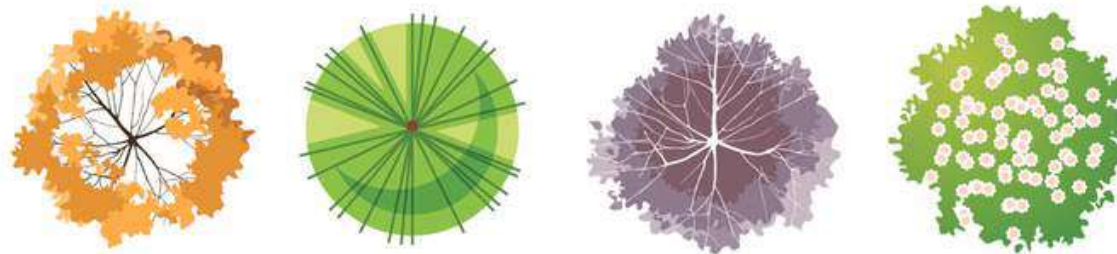
[Evaluation](#)

[Prizes](#)

[Data Processing  
Tutorial](#)

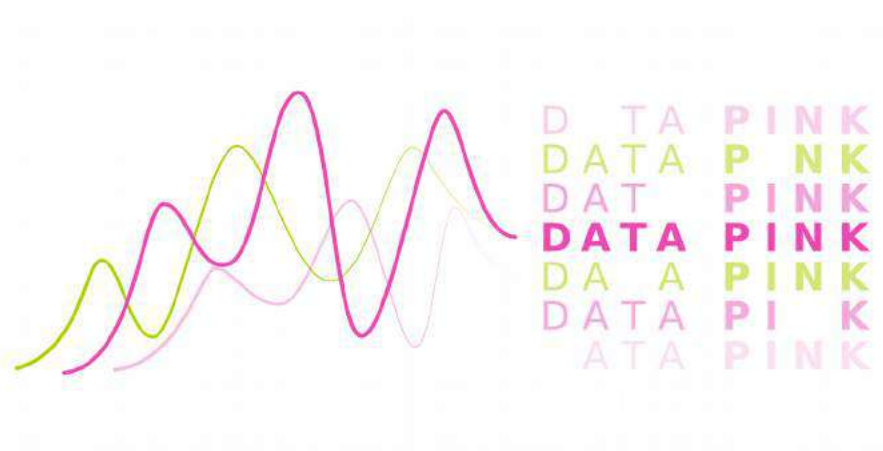
[Timeline](#)

The proliferation of satellite imagery has given us a radically improved understanding of our planet. It has enabled us to better achieve everything from mobilizing resources during disasters to monitoring effects of global warming. What is often taken for granted is that advancements such as these have relied on labeling features of significance like building footprints and roadways fully by hand or through imperfect semi-automated methods.



As these large, complex datasets continue to increase exponentially in number, the [Defence Science and Technology Laboratory \(Dstl\)](#) is seeking novel solutions to alleviate the burden on their image analysts. In this competition, Kagglers are challenged to accurately classify features in overhead imagery. Automating feature labeling will not only help Dstl make smart decisions more quickly around the defense and security of the UK, but also bring innovation to computer vision methodologies applied to satellite imagery.

# Conclusions



@data\_pink

[www.datapink.com](http://www.datapink.com)