DeepLearning and OpenSource GIS

@o_courtin
Rosenblatt, The Perceptron (1958)

Some activation functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Plot</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td><img src="image" alt="Identity Plot" /></td>
<td>( f(x) = x )</td>
</tr>
<tr>
<td>Binary step</td>
<td><img src="image" alt="Binary step Plot" /></td>
<td>( f(x) = \begin{cases} 0 &amp; \text{for } x &lt; 0 \ 1 &amp; \text{for } x \geq 0 \end{cases} )</td>
</tr>
<tr>
<td>Logistic (a.k.a. Soft step)</td>
<td><img src="image" alt="Logistic Plot" /></td>
<td>( f(x) = \frac{1}{1 + e^{-x}} )</td>
</tr>
<tr>
<td>TanH</td>
<td><img src="image" alt="TanH Plot" /></td>
<td>( f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 )</td>
</tr>
<tr>
<td>ArcTan</td>
<td><img src="image" alt="ArcTan Plot" /></td>
<td>( f(x) = \tan^{-1}(x) )</td>
</tr>
<tr>
<td>Softsign [7,8]</td>
<td><img src="image" alt="Softsign Plot" /></td>
<td>( f(x) = \frac{x}{1 +</td>
</tr>
<tr>
<td>Inverse square root unit (ISRU) [9]</td>
<td><img src="image" alt="Inverse square root unit Plot" /></td>
<td>( f(x) = \frac{x}{\sqrt{1 + \alpha x^2}} )</td>
</tr>
<tr>
<td>Rectified linear unit (ReLU) [10]</td>
<td><img src="image" alt="Rectified linear unit Plot" /></td>
<td>( f(x) = \begin{cases} 0 &amp; \text{for } x &lt; 0 \ x &amp; \text{for } x \geq 0 \end{cases} )</td>
</tr>
<tr>
<td>Leaky rectified linear unit (Leaky ReLU) [11]</td>
<td><img src="image" alt="Leaky rectified linear unit Plot" /></td>
<td>( f(x) = \begin{cases} 0.01x &amp; \text{for } x &lt; 0 \ x &amp; \text{for } x \geq 0 \end{cases} )</td>
</tr>
</tbody>
</table>

http://yann.lecun.com/exdb/lenet/
Convolution animations

- No padding, no strides
- Arbitrary padding, no strides
- Half padding, no strides
- No padding, strides
- Padding, strides
- Padding, strides (odd)

https://github.com/vdumoulin/conv_arithmetic
Categorical judgments, decision making
Motor command
140-190 ms
MC
120-160 ms
PMC
100-130 ms
PFC
10-20 ms
Retina
d20-40 ms
LGN
30-50 ms
PIT
140-200 ms
V1
60-80 ms
V2
70-90 ms
V4
80-100 ms
AIT
To finger muscle 180-260 ms
To spinal cord 160-220 ms

Simple visual forms, edges, corners
Intermediate visual forms, feature groups, etc.
High level object descriptions, faces, objects

[picture from Simon Thorpe]
In the competition’s first year teams had varying success. Every team got at least 25% wrong.

In 2012, the team to first use deep learning was the only team to get their error rate below 25%.

The following year nearly every team got 25% or fewer wrong.

In 2017, 29 of 38 teams got less than 5% wrong.
U-Net: Convolutional Networks for Biomedical Image Segmentation

https://arxiv.org/abs/1505.04597
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.
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/
Pre-trained ResNet50 with ImageNet on IR-R-G

<table>
<thead>
<tr>
<th></th>
<th>Overall</th>
<th>Impervious</th>
<th>Building</th>
<th>Low Vegetation</th>
<th>Tree</th>
<th>Car</th>
<th>Clutter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation</td>
<td>85.8</td>
<td>89.1</td>
<td>91.8</td>
<td>82.0</td>
<td>83.3</td>
<td>93.7</td>
<td>63.2</td>
</tr>
<tr>
<td>Test</td>
<td>89.2</td>
<td>91.4</td>
<td>96.1</td>
<td>86.1</td>
<td>86.6</td>
<td>93.3</td>
<td>46.8</td>
</tr>
</tbody>
</table>
# The number of output labels

```python
nb_labels = 6
```

# The dimensions of the input images

```python
nb_rows = 256
nb_cols = 256
```

# A ResNet model with weights from training on ImageNet. This will be adapted via graph surgery into an FCN.

```python
base_model = ResNet50(
    include_top=False, weights='imagenet', input_tensor=input_tensor)
```

# Get final 32x32, 16x16, and 8x8 layers in the original ResNet by that layers's name.

```python
x32 = base_model.get_layer('final_32').output
x16 = base_model.get_layer('final_16').output
x8 = base_model.get_layer('final_x8').output
```

# Compress each skip connection so it has nb_labels channels.

```python
c32 = Convolution2D(nb_labels, (1, 1))(x32)
c16 = Convolution2D(nb_labels, (1, 1))(x16)
c8 = Convolution2D(nb_labels, (1, 1))(x8)
```
# Resize each compressed skip connection using bilinear interpolation.
# This operation isn't built into Keras, so we use a LambdaLayer
# which allows calling a Tensorflow operation.

def resize_bilinear(images):
    return tf.image.resize_bilinear(images, [nb_rows, nb_cols])

r32 = Lambda(resize_bilinear)(c32)
r16 = Lambda(resize_bilinear)(c16)
r8 = Lambda(resize_bilinear)(c8)

# Merge the three layers together using summation.
m = Add()([r32, r16, r8])

# Add softmax layer to get probabilities as output. We need to reshape
# and then un-reshape because Keras expects input to softmax to
# be 2D.
x = Reshape((nb_rows * nb_cols, nb_labels))(m)
x = Activation('softmax')(x)
x = Reshape((nb_rows, nb_cols, nb_labels))(x)

fcn_model = Model(input=input_tensor, output=x)
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.
Final results

Below we present a small sample of the final results from our models:

![Image of buildings](image)

<table>
<thead>
<tr>
<th>Type</th>
<th>Wavebands</th>
<th>Pixel resolution</th>
<th>#channels</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>grayscale</td>
<td>Panchromatic</td>
<td>0.31 m</td>
<td>1</td>
<td>3348 x 3392</td>
</tr>
<tr>
<td>3-band</td>
<td>RGB</td>
<td>0.31 m</td>
<td>3</td>
<td>3348 x 3392</td>
</tr>
<tr>
<td>16-band</td>
<td>Multispectral</td>
<td>1.24 m</td>
<td>8</td>
<td>837 x 848</td>
</tr>
<tr>
<td></td>
<td>Short-wave infrared</td>
<td>7.5 m</td>
<td>8</td>
<td>134 x 136</td>
</tr>
</tbody>
</table>

https://blog.deepsense.ai/deep-learning-for-satellite-imagery-via-image-segmentation/
Figure 1: Deep learning architectures for joint processing of optical and OpenStreetMap data.


https://hal.archives-ouvertes.fr/hal-01523573
Figure 4: Excerpt from the classification results on Potsdam
Figure 1. Faster RCNN

Figure 1. Faster RCNN

G tries to synthesize fake images that fool D

D tries to identify the fakes
G tries to synthesize fake images that fool D

D tries to identify the fakes

Real or fake pair?
Labelled Datasets

Volodymyr PhD: https://www.cs.toronto.edu/~7Evmnih/data/


DeepSAT: http://csc.lsu.edu/~7Esaikat/deepsat/
NLP
import spacy
from spacy import displacy

nlp = spacy.load('en')
doc = nlp("The original question: Can Machine think? I believe to be too meaningless to deserve discussion."")
displacy.serve(doc, style='dep')
import spacy
from spacy import displacy

dl = spacy.load('en')
doc = dl('"
The original question: Can Machine think?
I believe to be too meaningless to deserve discussion"
')
displacy.serve(doc, style='dep')
The Port of Paulsboro is located on the Delaware River and Mantua Creek in and around Paulsboro, in Gloucester County, New Jersey, US, approximately 78 miles (126 km) from the Atlantic Ocean. Traditionally one of the nation's busiest for marine transfer operations, notably for crude oil and petroleum products, such as jet fuel and asphalt, it is a port of entry with several facilities within a foreign trade zone. A part of the port is being redeveloped as an adaptable deep water omniport able to handle a variety of bulk and break bulk cargo, as well as shipping containers. It is targeted to become a manufacturing/assembly center for wind turbines for the development of wind power in New Jersey and other offshore wind power projects along the East Coast of the United States. The Paulsboro Marine Terminal, as it is known, is owned by the South Jersey Port Corporation and operated by Holt Logistics. The first ship is expected to arrive at the new facility in early 2017 carrying steel for NLMK. The first ship to call at the port, the Doric Warrior, carrying steel for NLMK, arrived March 3, 2017, marking the opening of the new facility.

import spacy

nlp = spacy.load('en_core_web_sm')
doc = nlp(text_intro)

for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
The Port of Paulsboro 2 23 WORK_OF_ART
the Delaware River 38 56 LOC
Mantua Creek 61 73 GPE
Paulsboro 88 97 GPE
Gloucester County 102 119 GPE
New Jersey 121 131 GPE
US 133 135 GPE
approximately 78 miles 137 159 QUANTITY
126 km 161 167 QUANTITY
the Atlantic Ocean 174 192 LOC
New Jersey 695 705 GPE
the East Coast 751 765 LOC
the United States 769 786 GPE
The Paulsboro Marine Terminal 788 817 ORG
the South Jersey Port Corporation 847 880 ORG
Holt Logistics 897 911 ORG
first 917 922 ORDINAL
eyear 2017 973 983 DATE
NLMK 1003 1007 ORG
first 1013 1018 ORDINAL
the Doric Warrior 1045 1061 FAC
NLMK 1082 1086 ORG
March 3, 2017 1096 1109 DATE
import spacy

nlp = spacy.load('en_core_web_sm')
doc = nlp(text_intro)

for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)

The Port of Paulsboro 2 23 WORK_OF_ART
the Delaware River 38 56 LOC
Mantua Creek 61 73 GPE
Paulsboro 88 97 GPE
Gloucester County 102 119 GPE
New Jersey 121 131 GPE
US 133 135 GPE
approximately 78 miles 137 159 QUANTITY
126 km 161 167 QUANTITY
the Atlantic Ocean 174 192 LOC
New Jersey 695 705 GPE
the East Coast 751 765 LOC
the United States 769 786 GPE
The Paulsboro Marine Terminal 788 817 ORG
the South Jersey Port Corporation 847 880 ORG
Holt Logistics 897 911 ORG
first 917 922 ORDINAL
early 2017 973 983 DATE
NLMK 1003 1007 ORG
first 1013 1018 ORDINAL
the Doric Warior 1045 1061 FAC
NLMK 1082 1086 ORG
March 3, 2017 1096 1109 DATE

spatial_ent = {}

for ent in doc.ents:
    if (ent.label_ == 'LOC' or ent.label_ == 'GPE'):
        try:
            spatial_ent[ent.text] += 1
        except:
            spatial_ent[ent.text] = 1

print(spatial_ent)

{'the East Coast': 1, 'the Atlantic Ocean': 1, 'New Jersey': 2, 'the United States': 1, 'the Delaware River': 1, 'Mantua Creek': 1, 'Paulsboro': 1, 'US': 1, 'Gloucester County': 1}
NLP extraction + Geoname Gazeeter + Classification
from mordecai import Geoparser
geo = Geoparser()

res = geo.geoparse(text_intro)

import pprint as pp
pp.pprint(res)
from mordecai import Geoparser
geo = Geoparser()
res = geo.geoparse(text_intro)
import pprint as pp
pp.pprint(res)
for r in res:
    if r['country_conf'] > 0.9:
        print(r['geo']['place_name'],
              'http://www.geonames.org/{}/'.format(r['geo']['geonameid']))

Delaware http://www.geonames.org/4142224/
Mantua Creek http://www.geonames.org/4502882/
Borough of Paulsboro http://www.geonames.org/4503518/
Gloucestor County http://www.geonames.org/4501944/
New Jersey http://www.geonames.org/5101760/
Atlantic Ocean Palm Inn http://www.geonames.org/6528913/
New Jersey http://www.geonames.org/5101760/
East Coast Baptist Church http://www.geonames.org/7238976/
United States http://www.geonames.org/6252001/
```python
for r in res:
    if r['country_conf'] > 0.9:
        print(r['geo'][place_name],
              "http://www.geonames.org/{}/".format(r['geo'][geonameid]))
```

Mantua Creek [http://www.geonames.org/4502882/](http://www.geonames.org/4502882/)
Gloucester County [http://www.geonames.org/4501944/](http://www.geonames.org/4501944/)
East Coast Baptist Church [http://www.geonames.org/7238976/](http://www.geonames.org/7238976/)
Time Series
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
THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
```python
import numpy as np
import sqlalchemy as sa
from wkb_raster import read_wkb_raster
from io import BytesIO

pg = sa.create_engine('postgresql://o:xxx@127.0.0.1:5432/ngi')
sql = "SELECT ST_AsBinary(rast) AS data FROM ngi.rgb LIMIT 10"
lines = pg.execute(sql)

for line in lines:
    read_wkb_raster(BytesIO(line.data))['bands'][0]['ndarray']
```
class SimpleIter(mx.io.DataIter):
    def __init__(self, data_names, data_shapes, data_gen,
                 label_names, label_shapes, label_gen, num_batches=10):
        self._provide_data = zip(data_names, data_shapes)
        self._provide_label = zip(label_names, label_shapes)
        self.num_batches = num_batches
        self.data_gen = data_gen
        self.label_gen = label_gen
        self.cur_batch = 0

    def __iter__(self):
        return self

    def reset(self):
        self.cur_batch = 0

    def __next__(self):
        return self.next()

@property
def provide_data(self):
    return self._provide_data

@property
def provide_label(self):
    return self._provide_label

def next(self):
    if self.cur_batch < self.num_batches:
        self.cur_batch += 1
        data = [mx.nd.array(g(d[l])) for d, l in zip(self._provide_data, self.data_gen)]
        label = [mx.nd.array(g(d[l])) for d, l in zip(self._provide_label, self.label_gen)]
        return mx.io.DataBatch(data, label)
    else:
        raise StopIteration

https://mxnet.incubator.apache.org/tutorials/basic/data.html
WITH origins AS (SELECT '{(855878, 6534055), (87821, 6533022), (873294, 6541341), (879027, 6524893)}::float[]') AS ul ,
tiles AS (SELECT row_number() OVER() as tid,
ST_SetSRID(
    ST_MakeEnvelope(ul[1][1], ul[1][2], ul[1][1] + 1250, ul[1][2] + 1250)
    , 2154
) AS geom
FROM origins, generate_subscripts((SELECT ul FROM origins), 1) AS i
)
tile_rast AS (SELECT tiles.tid,
ST_AddBand(
    ST_SetSRID(
        ST_MakeEmptyRaster(256, 256,
            ST_Xmin(tiles.geom)::float3,
            ST_Ymin(tiles.geom)::float3,
            2.6,
            2154),
            '8BUI') AS rast
FROM tiles
)
images AS (SELECT tile_rast.tid,
    tile_rast.rast AS tile_rast,
    ST_MapAlgebra(
        ST_AddBand(tile_rast.rast, '8BUI'::text), 1,
        ST_Resample(ST_Grayscale(ST_Union(image.geom))), tile_rast.rast, 'bilinear'), 1,
        '[rast2]' , NULL, 'FIRST', '[rast2]'
    ) AS rast
FROM tile_rast, LATERAL
(SELECT rast
FROM sat.s2
WHERE ST_Contains(s2Geom, tile_rast.geom)
) AS image
GROUP BY tile_rast.tid, tile_rast.tid
)
labels AS (SELECT tile_rast.tid,
ST_MapAlgebra(
    tile_rast.rast,
    ST_Intersection(tile_rast.geom, '8BUI'),
    '([rast2])::integer', NULL, 'FIRST', '([rast2])::integer'
) AS rast
FROM tile_rast, LATERAL
(SELECT ST_ClippingByBox2D(ST_Buffer(ST_Union(osm.way), 10),
    ST_Envelope(tile_rast.geom)) geom
FROM planet_osm_line osm
WHERE osm.highway IS NOT NULL AND (osm.route = 'road' OR osm.route IS NULL)
AND ST_Contains(osm.way, tile_rast.geom)
GROUP BY tile_rast.tid, tile_rast.tid
)
SELECT Box3D(images.rast) AS bbox,
    ST_AsBinary(images.rast) AS data,
    CASE WHEN labels.rast IS NULL
    THEN ST_AsBinary(labels.tile_rast)
    ELSE ST_AsBinary(labels.rast)
    END AS label
FROM labels RIGHT JOIN images ON images.tid = labels.tid
batch_size = 2
max_iter = 2

geo_iter = GeoIter(
    'postgresql://o:xxx@127.0.0.1:5433/osm_qa',
    (850000, 6524040, 890960, 6565000), 2154, (256, 256), (10, 2.5),

    ""
        SELECT ST_ClipByBox2D(ST_Buffer(ST_Union(osm.way), 6),
                        ST_Envelope(tile_rast.rast)) geom
    FROM planet_osm_line osm
    WHERE osm.highway IS NOT NULL AND (osm.route = 'road' OR osm.route IS NULL)
    AND ST_Intersects(osm.way, tile_rast.rast)
    ""
,

    ""
        SELECT ST_GrayScale(ST_Union(s2.rast)) AS rast
    FROM sat.s2
    WHERE ST_Intersects(s2.rast, tile_rast.rast)
    ""
,

    batch_size, max_iter)
MxNet RecordIO fast data loader

Data Format

Since the training of deep neural network often involves large amounts of data, the format we choose should be both efficient and convenient. To achieve our goals, we need to pack binary data into a splittable format. In MXNet, we rely on the binary recordIO format implemented in dmlc-core.

Binary Record

![Binary Record Diagram]

https://mxnet.incubator.apache.org/architecture/note_data_loading.html
Human Learning
Conclusions