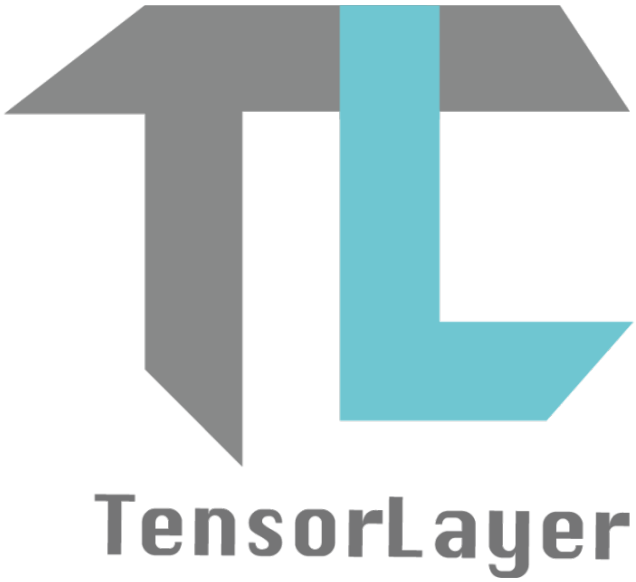


A Versatile Library for Efficient Deep Learning Development

Hao Dong, Akara Supratak, Luo Mai, Fangde Liu, Axel Oehmichen, Simiao Yu, Yike Guo

Imperial College London

Imperial College London

ACMmultimedia25

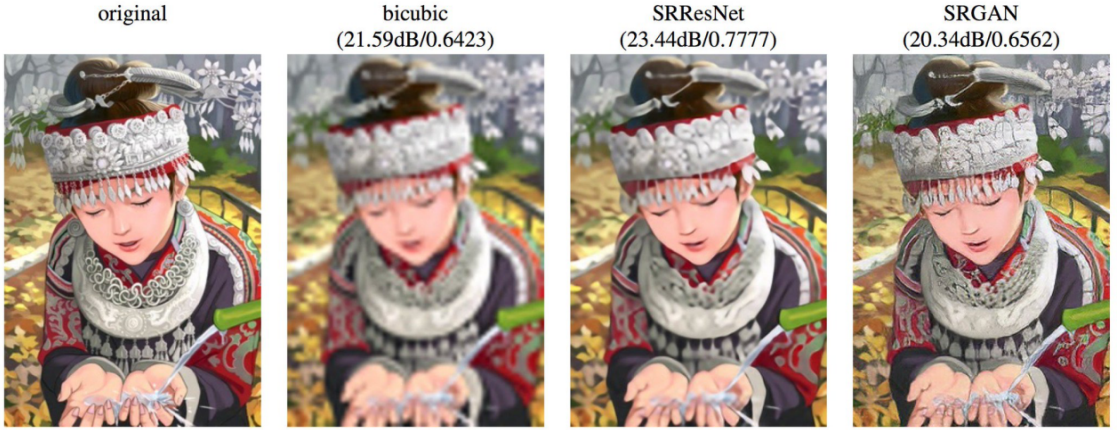
Outline

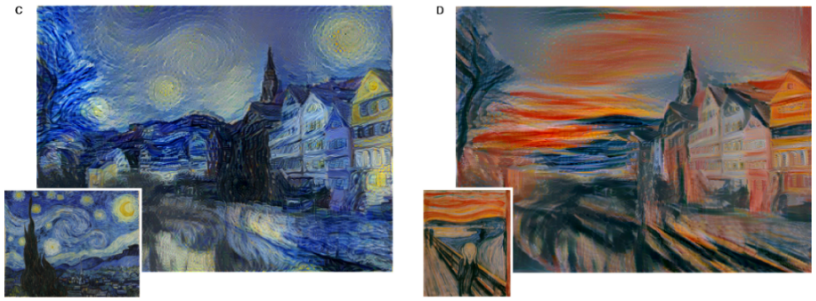
- Motivation
- What is TensorLayer?
- Design Principles
- Architecture
- Impact
- Future Plan

Motivation

- Successful multimedia applications with Deep Learning

bus parked on the street. a city street scene. front windshield of a bus. man walking on sidewalk. a silver car parked on the street. a city scene. a green traffic light. a building in the background. the bus has a number. a large building. a brick building. red brick building with windows. a blue sign with a white arrow. white lines on the road.





http://cs.stanford.edu/people/karpathy/densecap/eg/p1.jpeg
https://research.googleblog.com/2016/10/supercharging-style-transfer.html
https://twitter.com/smerity/status/776620914267992064

Motivation

TensorLayer

- Problem
  - Difficult to develop complex deep learning models
    - Different types of layers (e.g., CNNs and RNNs)
    - Complicated training (e.g., GANs)
  - Existing libraries
    - Too high-level of abstraction
    - Difficult to modify low-level operations
    - No support for model and data management in the distributed setting

What is TensorLayer?

- Python library built on top of Tensorflow from Google
- Easy-to-use modules
  - Data processing
  - Model training
  - Data and model management
- Designed for both researchers and software engineers
- Require less efforts for production-ready version

Design Principles

TensorLayer

1. Simplicity
2. Transparency
3. Composability
4. Performance

Design Principles - Simplicity

TensorLayer

- Provide high-level state-of-the-art deep learning modules

```
def model(x, reuse=False, is_train=False):
    with tf.variable_scope("model", reuse=reuse):
        tl.layers.set_name_reuse(reuse)
        net = InputLayer(x, name='input')

        net = Conv2d(net, 64, (5, 5), (1, 1), padding='SAME', name='cnn1')
        net = BatchNormLayer(net, is_train, act=tf.nn.relu, name='batch1')
        net = MaxPool2d(net, (3, 3), (2, 2), padding='SAME', name='pool1')

        net = Conv2d(net, 64, (5, 5), (1, 1), padding='SAME', name='cnn2')
        net = BatchNormLayer(net, is_train, act=tf.nn.relu, name='batch2')
        net = MaxPool2d(net, (3, 3), (2, 2), padding='SAME', name='pool2')

        net = FlattenLayer(net, name='flatten')
        net = DenseLayer(net, 384, act=tf.nn.relu, name='d1')
        net = DenseLayer(net, 192, act=tf.nn.relu, name='d2')
        net = DenseLayer(net, 10, name='output')
        return net
```

MultiplexerLayer

SubpixelConv1d

SlimNetsLayer

SpatialTransformer2dAffineLayer

SubpixelConv2d

DropconnectDenseLayer

Seq2Seq

ROIPoolingLayer

UnStackLayer

Design Principles - Transparency

TensorLayer

- Enable users to build a model using native Tensorflow APIs

```
def model(x, reuse=False, is_train=False):
    with tf.variable_scope("model", reuse=reuse):
        tl.layers.set_name_reuse(reuse)
        net = InputLayer(x, name='input')

        net = Conv2d(net, 64, (5, 5), (1, 1), padding='SAME', name='cnn1')
        net = BatchNormLayer(net, is_train, act=tf.nn.relu, name='batch1')
        net = MaxPool2d(net, (3, 3), (2, 2), padding='SAME', name='pool1')

        net = Conv2d(net, 64, (5, 5), (1, 1), padding='SAME', name='cnn2')
        net = BatchNormLayer(net, is_train, act=tf.nn.relu, name='batch2')
        net = MaxPool2d(net, (3, 3), (2, 2), padding='SAME', name='pool2')

        net = FlattenLayer(net, name='flatten')
        net = DenseLayer(net, 384, act=tf.nn.relu, name='d1')
        net = DenseLayer(net, 192, act=tf.nn.relu, name='d2')
        net = DenseLayer(net, 10, name='output')
        return net
```

Pass by function

Design Principles - Transparency

TensorLayer

- Enable users to build a model using native Tensorflow APIs

```
def model(x, reuse=False, is_train=False):
    with tf.variable_scope("model", reuse=reuse):
        tl.layers.set_name_reuse(reuse)
        net = InputLayer(x, name='input')

        net = Conv2d(net, 64, (5, 5), (1, 1), padding='SAME', name='cnn1')
        net = BatchNormLayer(net, is_train, act=tf.nn.relu, name='batch1')
        net = MaxPool2d(net, (3, 3), (2, 2), padding='SAME', name='pool1')

        net = Conv2d(net, 64, (5, 5), (1, 1), padding='SAME', name='cnn2')
        net = BatchNormLayer(net, is_train, act=tf.nn.relu, name='batch2')
        net = MaxPool2d(net, (3, 3), (2, 2), padding='SAME', name='pool2')

        net = FlattenLayer(net, name='flatten')
        net = DenseLayer(net, 384, act=tf.nn.relu, name='d1')
        net = DenseLayer(net, 192, act=tf.nn.relu, name='d2')
        net = DenseLayer(net, 10, name='output')
        return net
```

```
net = Conv2dLayer(net, act=tf.identity, shape=[5, 5, 3, 64],
        strides=[1, 1, 1, 1], padding='SAME', name='cnn1')

net = PoolLayer(net, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
        padding='SAME', pool=tf.nn.max_pool, name='pool1')
```

Advanced-version API

Pass by function

Design Principles - Transparency

TensorLayer

- Enable users to define their own computational operations

```
net = DenseLayer(net, 200, act=lambda x: tl.act.lrelu(x, 0.2), name='d1')

net = DenseLayer(net, 200, act=lambda x: 5*x, name='d2')
```

Lambda expression

```
net = ElementwiseLayer([net1, net2, net3], tf.add, name='ele')
```

tf.minimum, tf.maximum, tf.multiply, lambda x, y : x+2*y and etc

```
net = DynamicRNNLayer(net,
                      cell_fn=tf.contrib.rnn.BasicLSTMCell,
                      n_hidden=100,
                      name='rnn')
```

tf.contrib.rnn.(RNNCell, GRUCell, LSTMCell, LayerNormBasicLSTMCell and etc)

Design Principles - Composability

TensorLayer

- Users can glue different modules together (e.g., connected with TF-Slim and Keras).

```
x = tf.placeholder(tf.float32, shape=[None, 784])
y_ = tf.placeholder(tf.int64, shape=[None,])

def keras_block(x):
    x = Dropout(0.8)(x)
    x = Dense(800, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(800, activation='relu')(x)
    x = Dropout(0.5)(x)
    logits = Dense(10, activation='linear')(x)
    return logits

net = InputLayer(x, name='input')
net = LambdaLayer(net, fn=keras_block, name='keras')
```

Keras

Keras to TensorLayer

```
x = tf.placeholder(tf.float32, shape=[None, 299, 299, 3])

net = tl.layers.InputLayer(x, name='input')

with slim.arg_scope(inception_v3_arg_scope()):
    net = tl.layers.SlimNetsLayer(layer=net,
                                  slim_layer=inception_v3,
                                  slim_args={
                                    'num_classes' : 1001,
                                    'is_training' : False
                                  },
                                  name='InceptionV3')
```

TF-slim

TF-Slim to TensorLayer

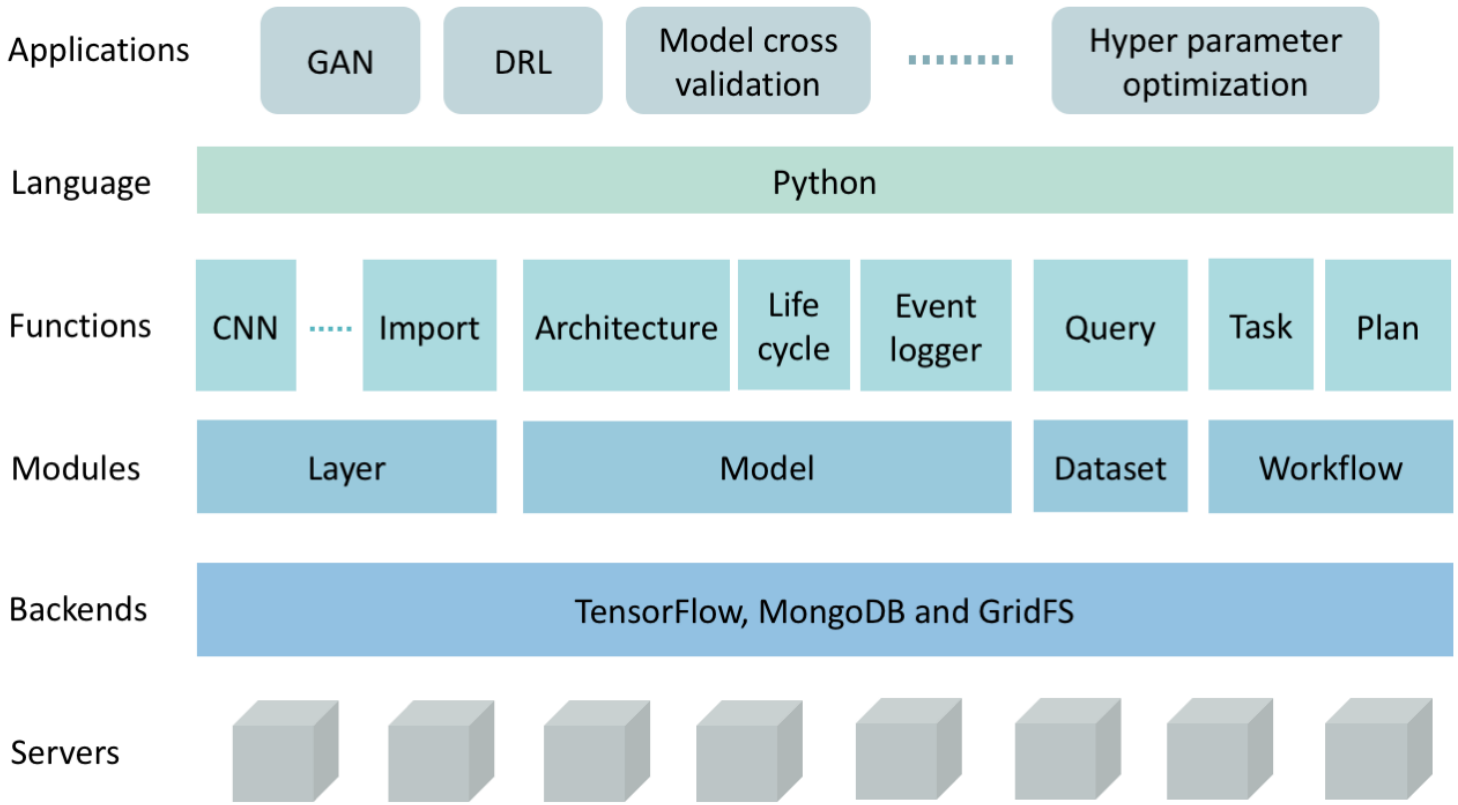
Design Principles - Performance

TensorLayer

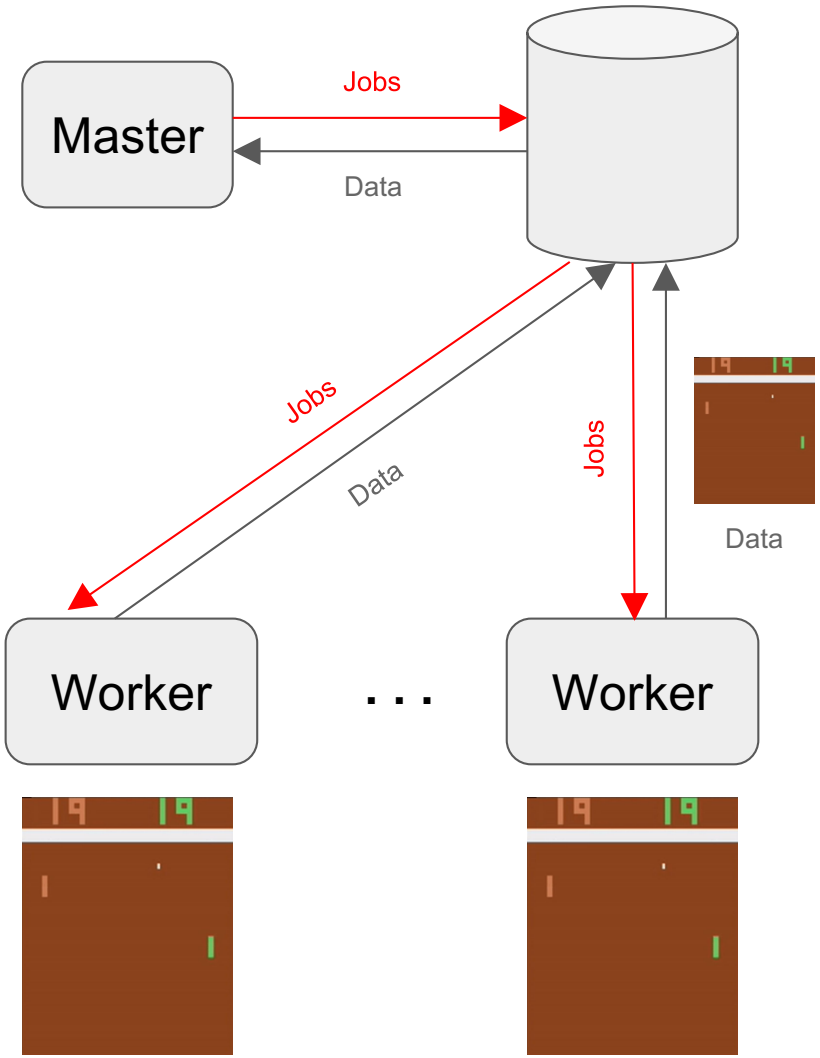
- Provide zero-cost abstraction (negligible computational overhead)

	CIFAR-10	PTB LSTM	Word2Vec
TensorLayer	2528 images/s	18063 words/s	58167 words/s
TensorFlow	2530 images/s	18075 words/s	58181 words/s

TensorLayer Architecture

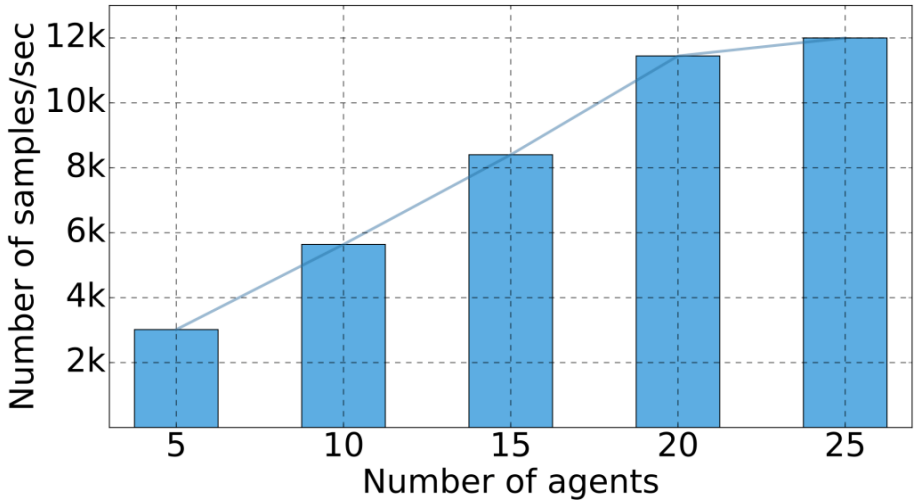


Example: Distributed Training



Everything is data !!!

- Jobs
- Data
- Train log
- Model
- etc.



Impact - Github

TensorLayer

~ 2500 Stars **~ 600 Forks** **~ Global Contributors**

Impact - Book from Community

Invited by Publishing House of Electronics Industry (PHEI, 电子工业出版社)

“Practical Deep Learning with TensorLayer”

10 contributors

Coming soon ~ Jan 2018

English version will come later

Installation and Documentation

Installation:

pip install tensorlayer

Platform:

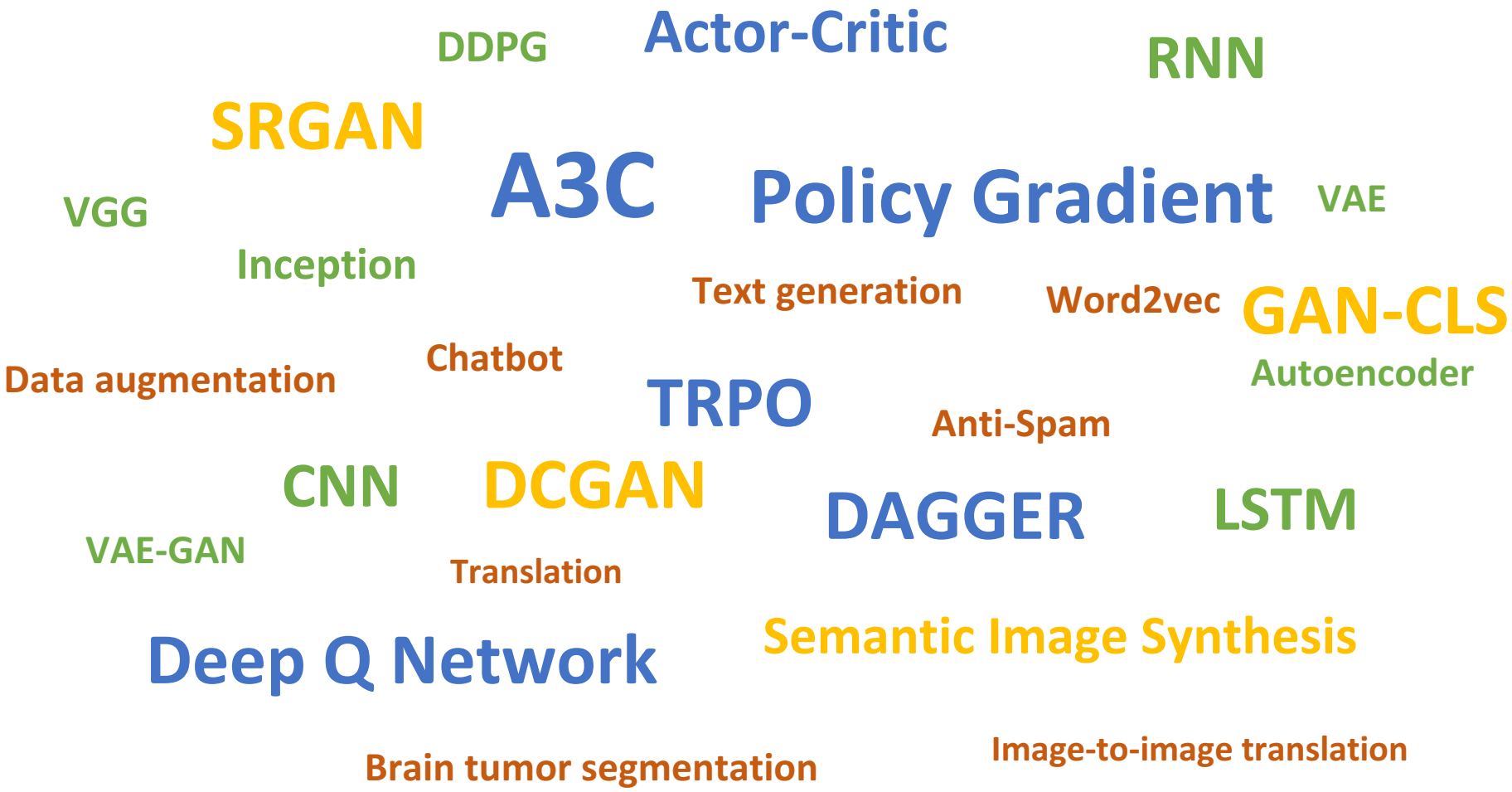
Linux, Windows, OSX

Documentation:

 http://tensorlayer.readthedocs.io/en/latest/ or http://tensorlayer.org

 http://tensorlayercn.readthedocs.io

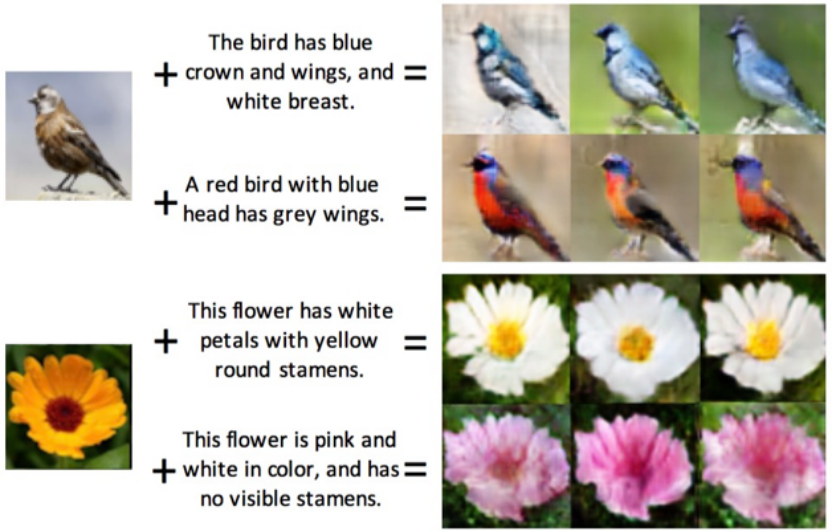
Tutorials and Examples

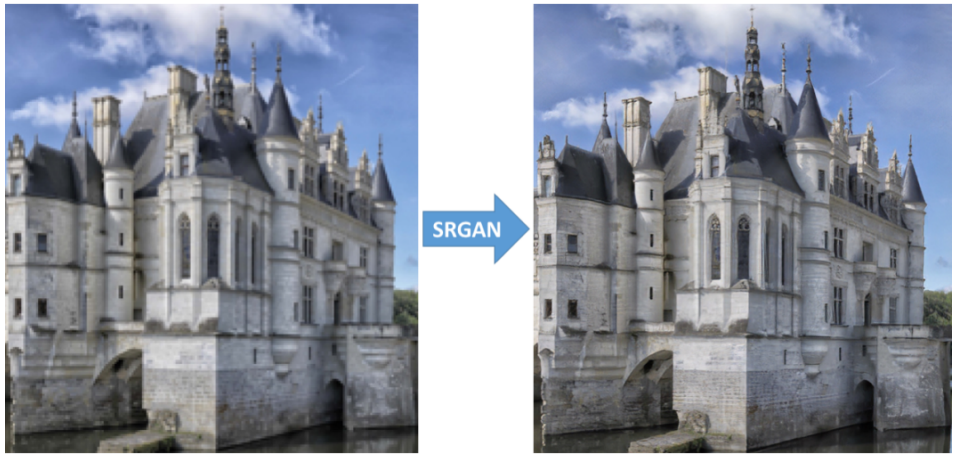


Research

Person A to B

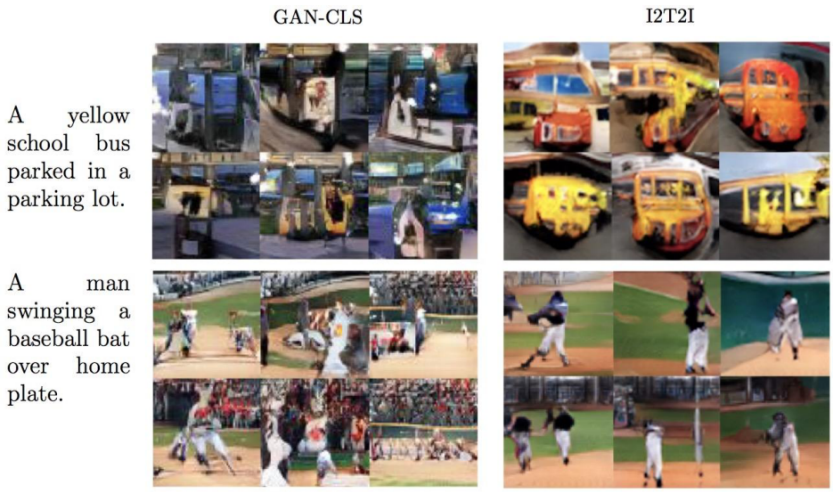
Person B to A





LG Image

Generated Image



Community

TensorLayer

GitHub

WeChat

slack

Future

TensorLayer 1.6 → 2.0

1. Extend dataset module supports more database systems
2. Tool for Model Compression
3. Tool for Cloud Serving

Q & A