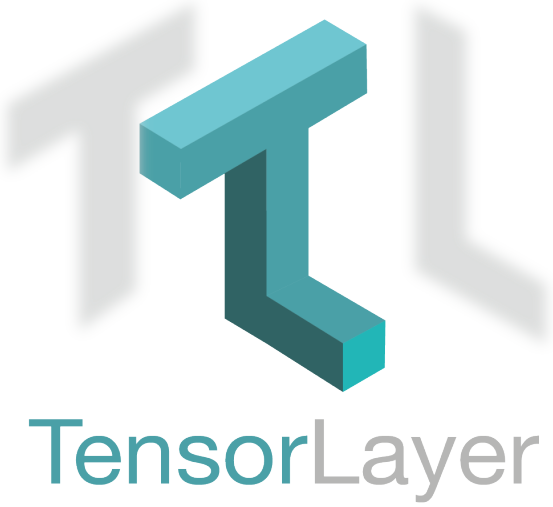
Imperial College London



TensorLayer: A Versatile Library for Deep Learning Development

Hao Dong, Luo Mai, Simiao Yu

Imperial College London

Acknowledge: Akara Supratak, Guo Li, Fangde Liu, Axel Oehmichen, Yike Guo

Outline

TensorLayer

- Motivation
- Design
- Impact
- Example & Research
- Future

Motivation

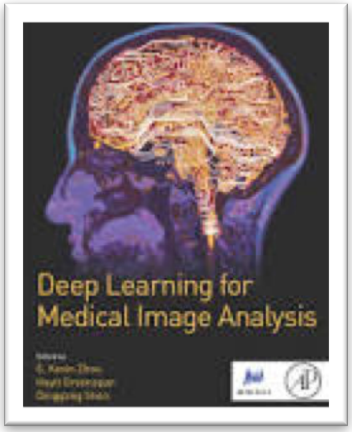
Motivation

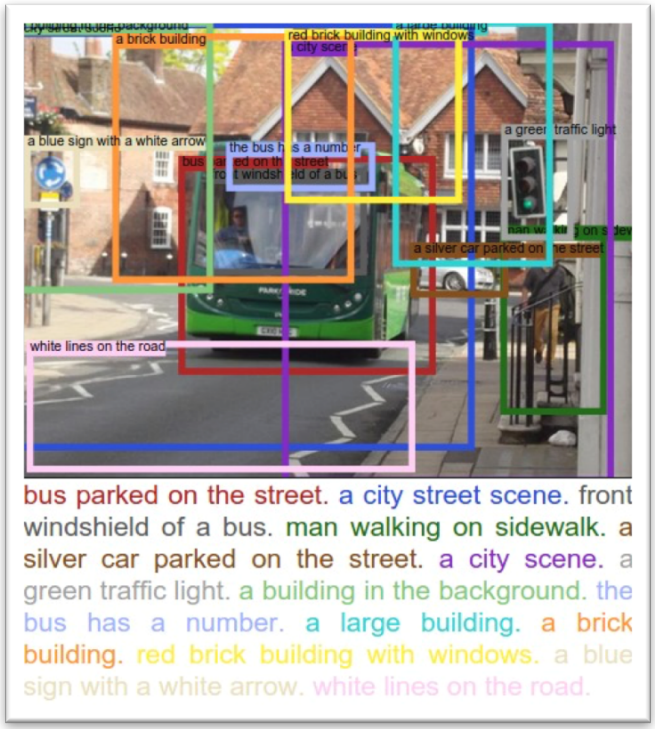
TensorLayer

Deep learning has enabled many AI systems to exhibit unprecedented performance even beyond human

- Computer vision
- Natural language processing
- Medical imaging and diagnosis
- Game playing
- …

AlphaGo





http://cs.stanford.edu/people/karpathy/densecap/eg/p1.jpeg

Motivation

TensorLayer

Key reasons for deep learning success

- Big data and systems
  - ImageNet, Youtube8M, Hadoop Ecosystem
- Algorithm breakthroughs
  - AlexNet, GoogleNet, AlphaGo, AlphaZero
- Affordable commodity hardware
  - GPU, TPU, FPGA and ASIC
- Fast innovations through open-source
  - Data, algorithm and hardware can be accessed by anyone

Motivation

TensorLayer

Open-source deep learning platforms

~90000 GitHub stars

Key reasons for TensorFlow

- Largest user base
- Widest production adoption
- Well-maintained documents
- Battlefield-proof quality

PYTORCH

~12000 GitHub stars

Microsoft CNTK

~14000 GitHub stars

Caffe2

~7000 GitHub stars

mxnet

~12000 GitHub stars

PaddlePaddle

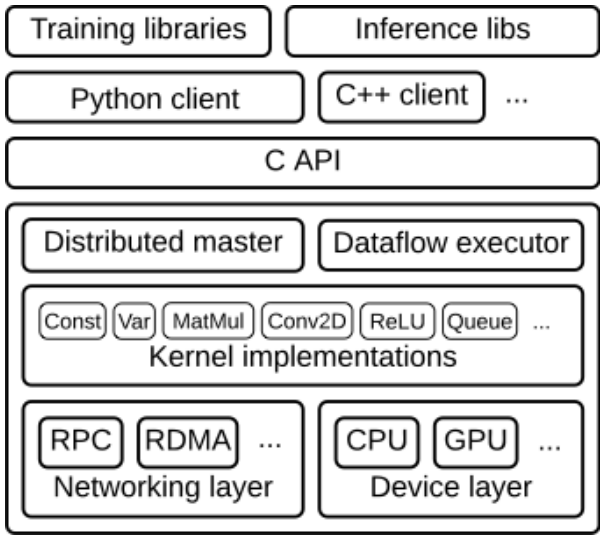
~6000 GitHub stars

Motivation

TensorLayer

Google TensorFlow is powerful but hard to master

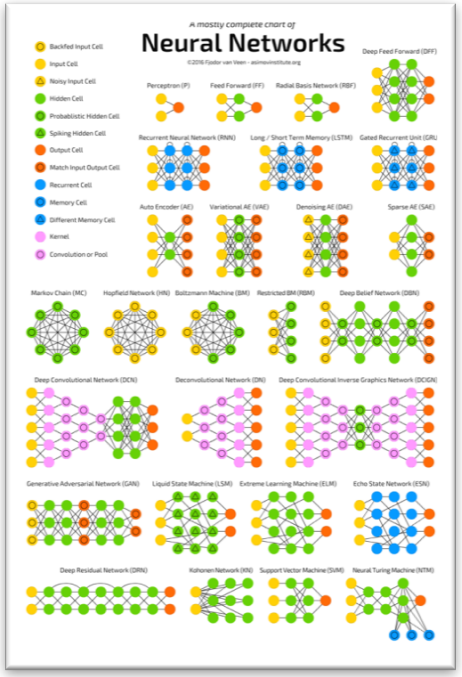
TensorFlow low-level interface:
dataflow graph, placeholder,
session, queue runner, devices …



Abstraction gap

Bridged by wrappers:
TensorLayer, Keras
and TFLearn

Deep learning high-level
elements: neural networks,
layers and tensors



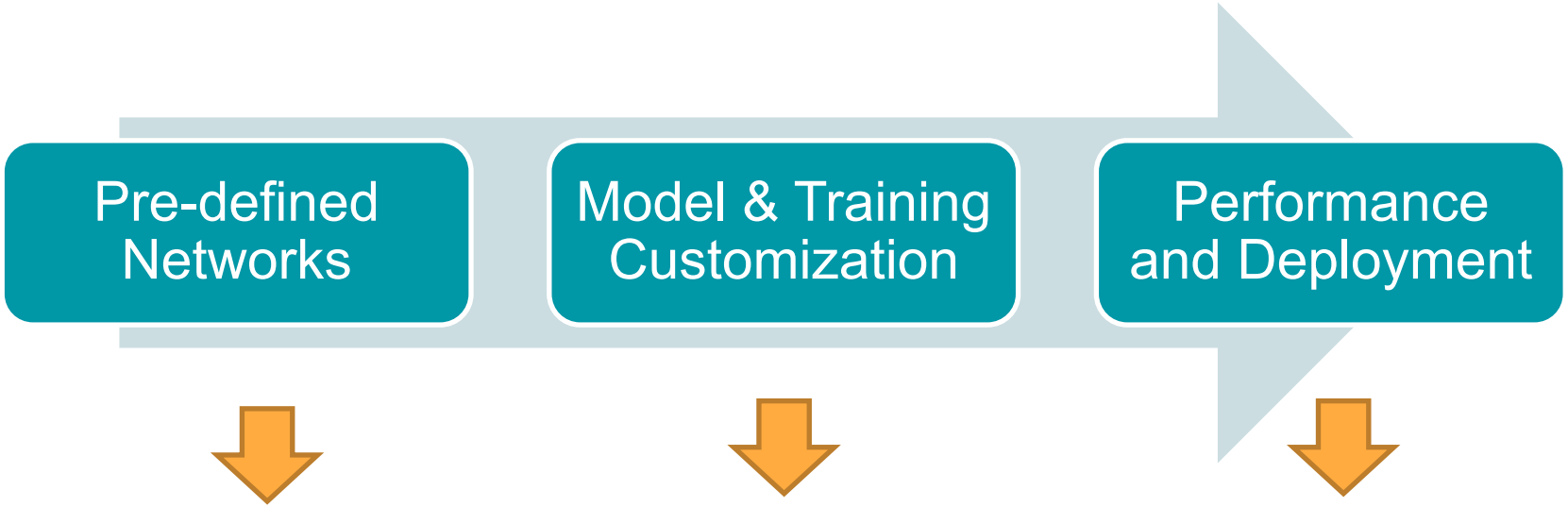
https://www.tensorflow.org/extend/architecture

http://www.asimovinstitute.org/neural-network-zoo/

Motivation

TensorLayer

Library requirements during development stages



- ✓ Reference networks
- ✓ Abstract layers
- ✓ Reusing modules

Bootstrap stage

- ✓ Customizable layers
- ✓ Data pre/post-processing
- ✓ Training control

Tuning stage

- ✓ Cloud deployment
- ✓ Distributed training
- ✓ Model management

Deployment stage

Motivation

TensorLayer

Comparing TensorLayer, Keras and TFLearn

	TensorLayer	Keras	TFLearn
Abstraction	Modest	High	Modest
Reference layers	Rich	Limited	Limited
Portable backend	No	Yes	No
Flexibility	Excellent	Limited	Good
Performance & deployment	Excellent	Limited	Good

TensorLayer stands at a unique spot for its **simplicity**, **flexibility** and **performance**.

Design

Design Highlights

TensorLayer

Simplicity

- High-level state-of-the-art deep learning layers
- Mass examples and tutorials
- Mass helper functions for data preparation, augmentation, visualization and evaluation
- Import modules from Keras and TFSlim

Transparency

- Easy to access TensorFlow operations
- Easy to insert customized model and training logics

Performance

- Zero performance downgrade compared to raw TensorFlow
- Easy to scale from laptops to clouds

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
```

DeformableConv2dLayer

MultiplexerLayer

SubpixelConv1d

SlimNetsLayer

SpatialTransformer2dAffineLayer

SubpixelConv2d

ConvLSTMLayer

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
```

Simplified-version API

```
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

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 - Transparency

TensorLayer

- Enable users to define their own training logic

Training Toolbox

```
tl.utils.fit(sess, network, train_op, cost, X_train, y_train, x, y_,
        acc=acc, batch_size=500, n_epoch=500, print_freq=5,
        X_val=X_val, y_val=y_val, eval_train=False)
```

Data Iteration Toolbox

```
for epoch in range(n_epoch):
    start_time = time.time()
    for X_train_a, y_train_a in tl.iterate.minibatches(
                            X_train, y_train, batch_size, shuffle=True):
        sess.run(train_op, feed_dict={x: X_train_a, y_: y_train_a})

    if epoch + 1 == 1 or (epoch + 1) % print_freq == 0:
        print("Epoch %d of %d took %fs" % (epoch + 1,
                                    n_epoch, time.time() - start_time))
        train_loss, train_acc, n_batch = 0, 0, 0
        for X_train_a, y_train_a in tl.iterate.minibatches(
                            X_train, y_train, batch_size, shuffle=True):
            err, ac = sess.run([cost_test, acc],
                                    feed_dict={x: X_train_a, y_: y_train_a})
            train_loss += err; train_acc += ac; n_batch += 1
        print("   train loss: %f" % (train_loss/ n_batch))
        print("   train acc: %f" % (train_acc/ n_batch))
        val_loss, val_acc, n_batch = 0, 0, 0
        for X_val_a, y_val_a in tl.iterate.minibatches(
                                X_val, y_val, batch_size, shuffle=True):
            err, ac = sess.run([cost_test, acc],
                                    feed_dict={x: X_val_a, y_: y_val_a})
            val_loss += err; val_acc += ac; n_batch += 1
        print("   val loss: %f" % (val_loss/ n_batch))
        print("   val acc: %f" % (val_acc/ n_batch))
```

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 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

Titan X Pascal GPU 2017

Design Principles - Performance

TensorLayer

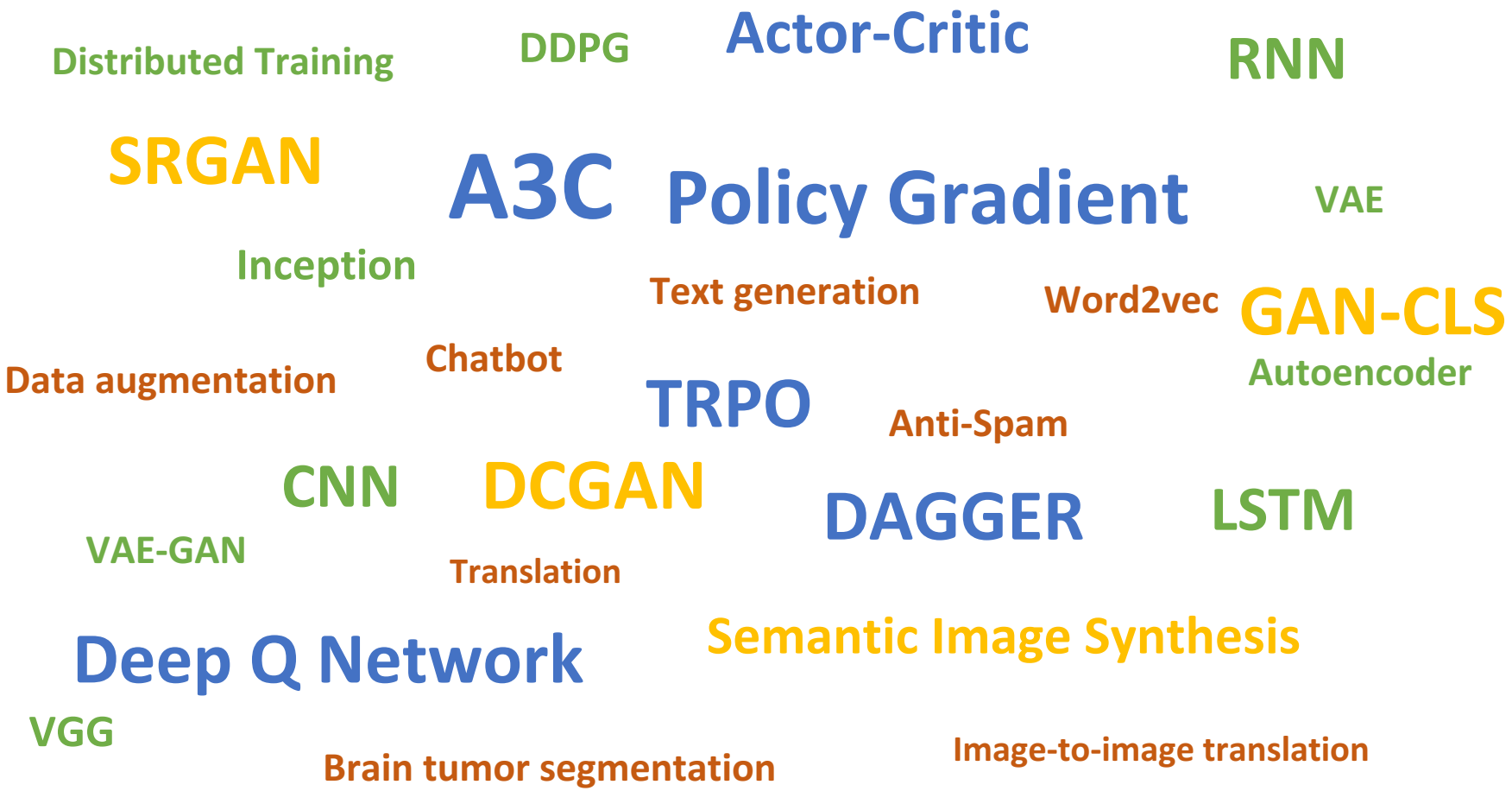
Scaling your models from laptops to clouds

- Scenario: Train your models using multiple GPUs in **one machine**
  - Start with TensorLayer MNIST and InceptionV3 ImageNet examples to get familiar with the distributed APIs
  - Collaborate with the **Good AI Lab (ClusterOne)**

- Scenario: Train your models using a **multi-server cluster**
  - Play with the simple Jupter Notebook that back up by public clouds including Google Cloud Engine, Amazon AWS and Microsoft Azure
  - Collaborate with the **Google Kubeflow team**

Tutorials and Examples

TensorLayer



Installation and Documentation

TensorLayer

Installation:

pip install tensorlayer

Platform:

Linux, Windows, OSX

Documentation:

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

 http://tensorlayercn.readthedocs.io

 Coming soon

Impact

Impact - Github

TensorLayer

Jan 2018 ~ 3000 Stars ~ 700 Forks ~ Global Contributors

清華大學
Tsinghua University

Stanford University

LINKÖPING UNIVERSITY

Tencent 腾讯

PENGUINS INNOVATE

ReFUEL4

UCLA

Good AI Lab

Bloomberg

Google

Alibaba Group

小米
xiaomi.com

Microsoft

Impact - Media and Award

TensorLayer

机器之心 Synced

THENEXTPLATFORM

Good AI Lab

Humans of AI

TensorLayer: A Versatile Library for Efficient Deep Learning Development. *H. Dong, A. Supratak et al. ACM MM 2017.*

Impact - Book from Community

TensorLayer

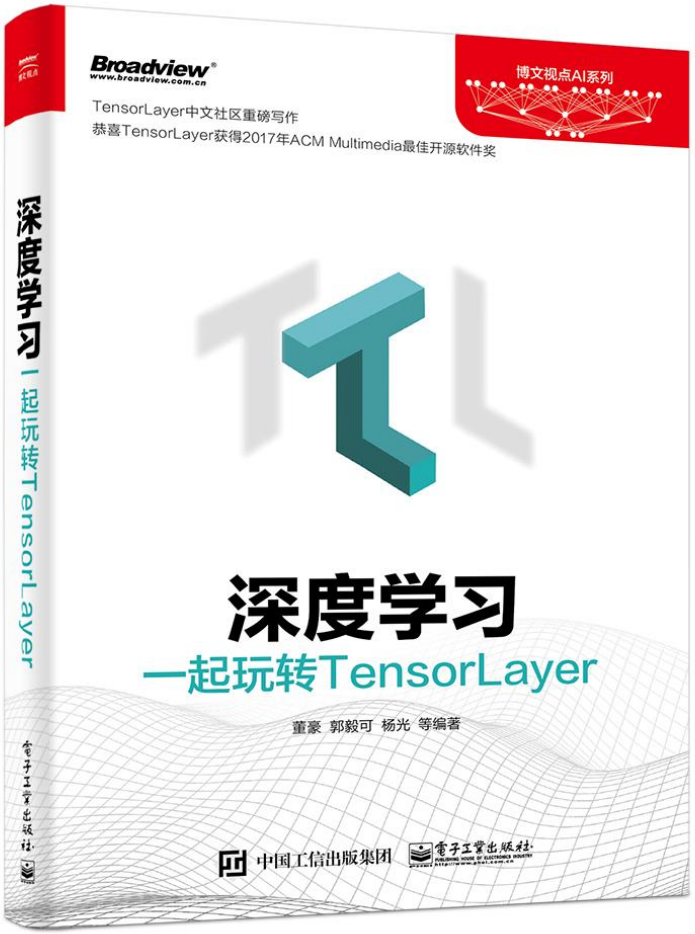
Invited by Publishing House of Electronics Industry (PHEI, 电子工业出版社)

“Deep Learning using TensorLayer”

10 contributors

Published date: Mar 2018

English version will come later



Imperial College London

Example & Research

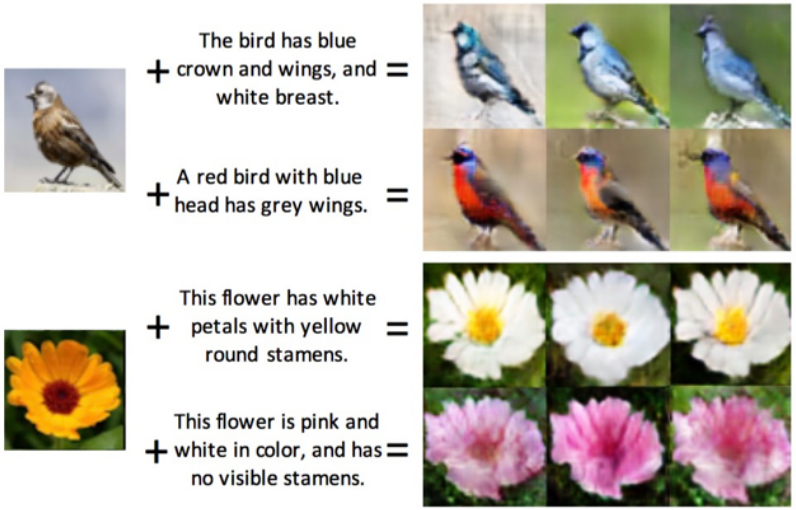
Research - Computer Vision

TensorLayer

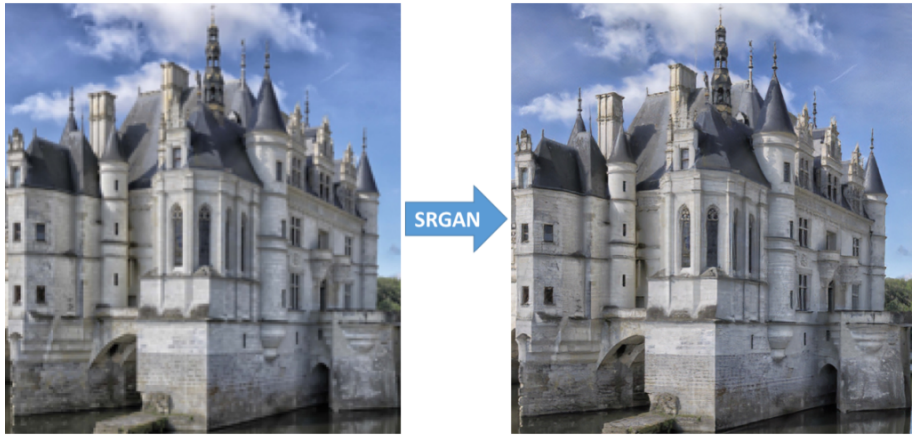
Unsupervised Image to Image Translation

Person A to B Person B to A

Semantic Image Synthesis

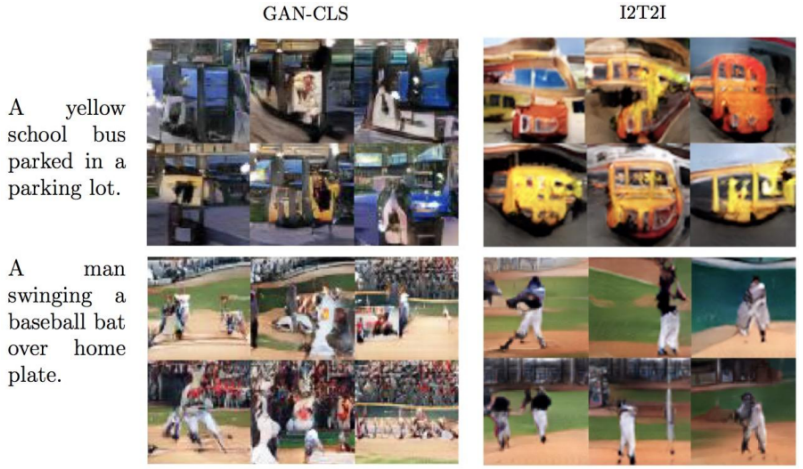


Super Resolution



LG Image Generated Image

Text to Image Synthesis

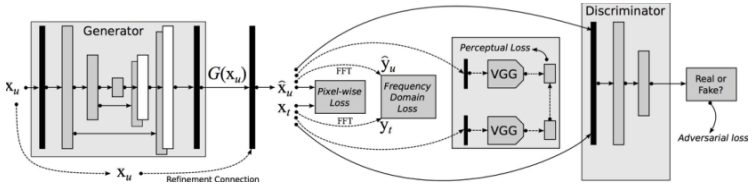


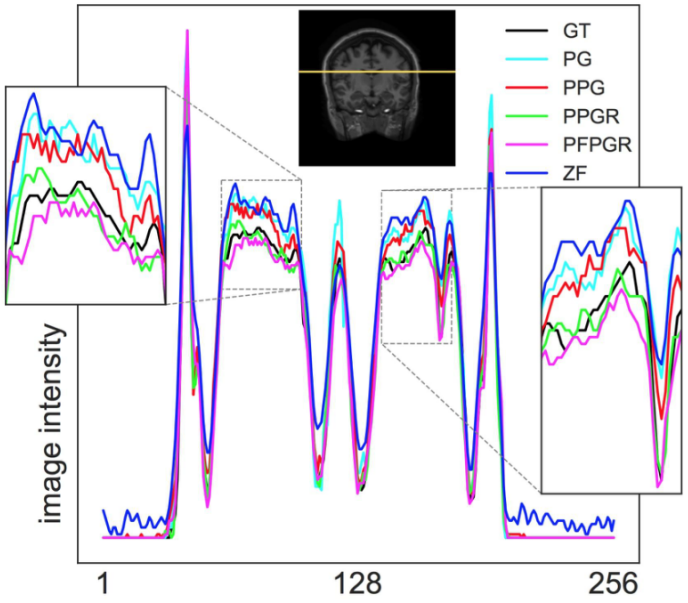
Semantic Image Synthesis via Adversarial Learning. *H. Dong, S. Yu et al. ICCV 2017.*

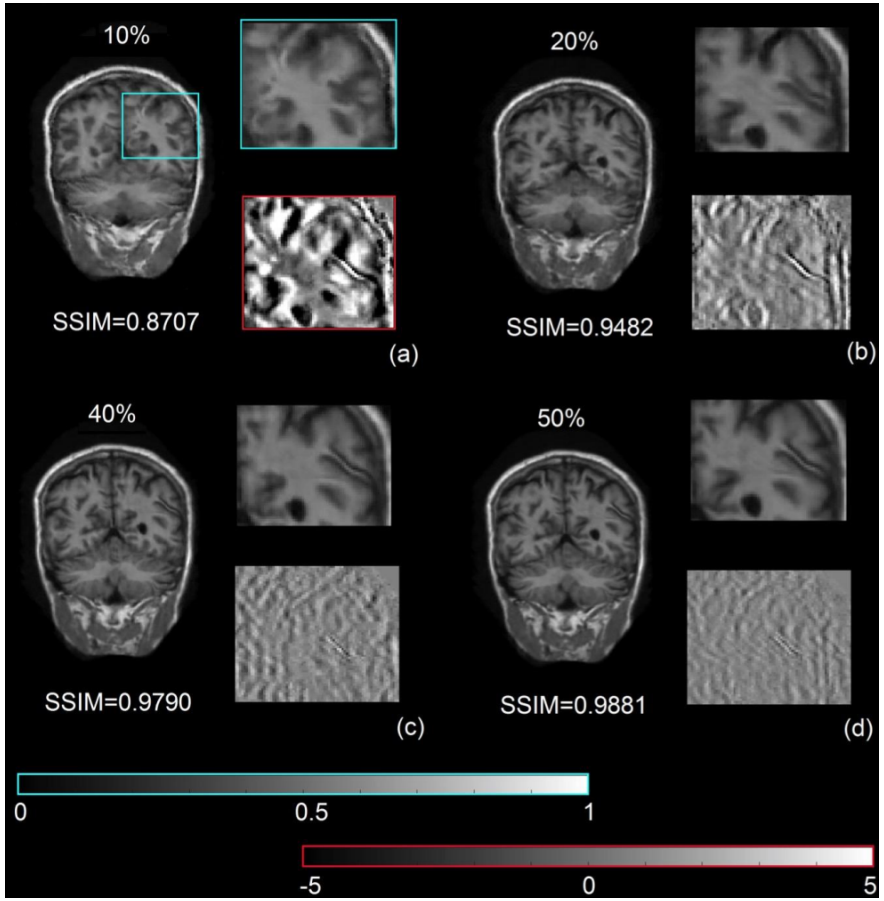
Research - Medical Applications

TensorLayer

Super Resolution for Medical Image





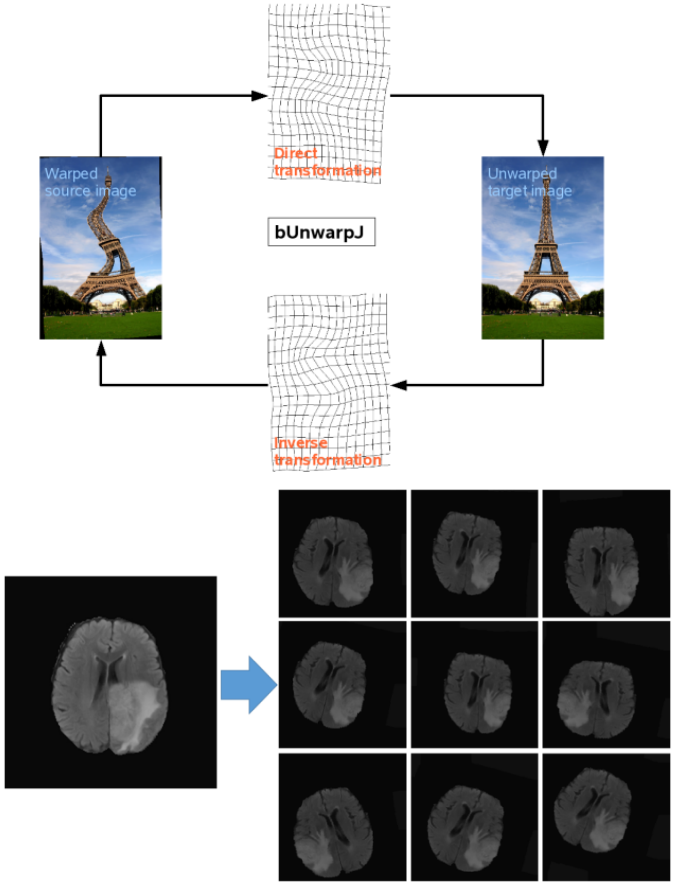


Imperial College London

DAGAN: Deep De-Aliasing Generative Adversarial Networks for Fast Compressed Sensing MRI Reconstruction. *G. Yang, S. Yu, H. Dong et al. TMI 2018.*

Research - Medical Applications

Elastic Transformation for MRI Segmentation




```
def distort_imgs(data):
    """ data augumentation """
    x1, x2, x3, x4, y = data
    x1, x2, x3, x4, y = tl.prepro.flip_axis_multi([x1, x2, x3, x4, y],
                        axis=1, is_random=True) # left right
    x1, x2, x3, x4, y = tl.prepro.elastic_transform_multi([x1, x2, x3, x4, y],
                        alpha=720, sigma=24, is_random=True)
    x1, x2, x3, x4, y = tl.prepro.rotation_multi([x1, x2, x3, x4, y], rg=20,
                        is_random=True, fill_mode='constant') # nearest, constant
    x1, x2, x3, x4, y = tl.prepro.shift_multi([x1, x2, x3, x4, y], wrg=0.10,
                        hrg=0.10, is_random=True, fill_mode='constant')
    x1, x2, x3, x4, y = tl.prepro.shear_multi([x1, x2, x3, x4, y], 0.05,
                        is_random=True, fill_mode='constant')
    x1, x2, x3, x4, y = tl.prepro.zoom_multi([x1, x2, x3, x4, y],
                        zoom_range=[0.9, 1.1], is_random=True,
                        fill_mode='constant')
    return x1, x2, x3, x4, y
```

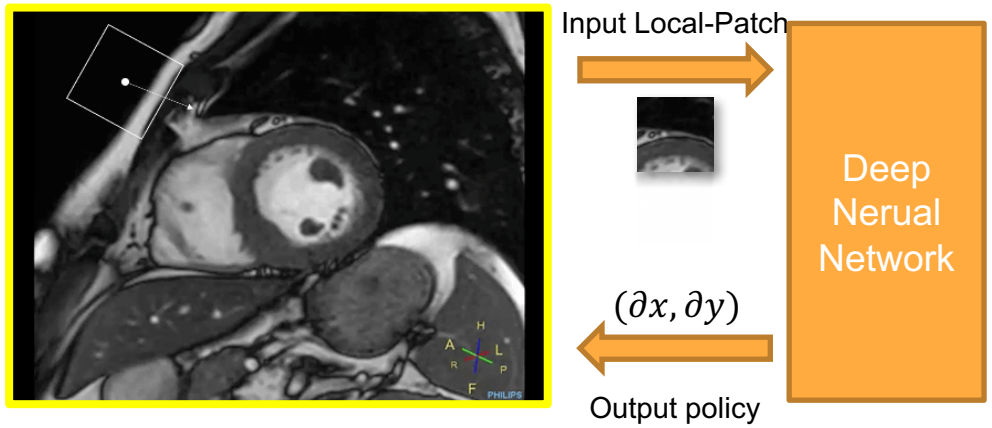
Automatic Brain Tumor Detection and Segmentation Using U-Net based Fully Convolutional Networks.
H. Dong, G. Yang, F. Liu et al. MIUA 2017.

Research - Medical Applications

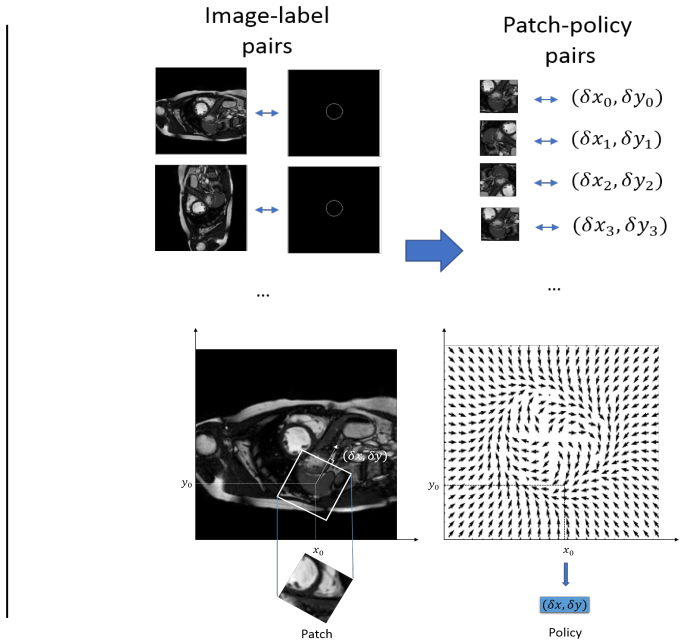
TensorLayer

Deep Poincare Map (DPM) for MRI Segmentation

Animation of Inference



Training



Unlike tradition End-to-End DL method.

DPM is a iterative process to segment ROI.

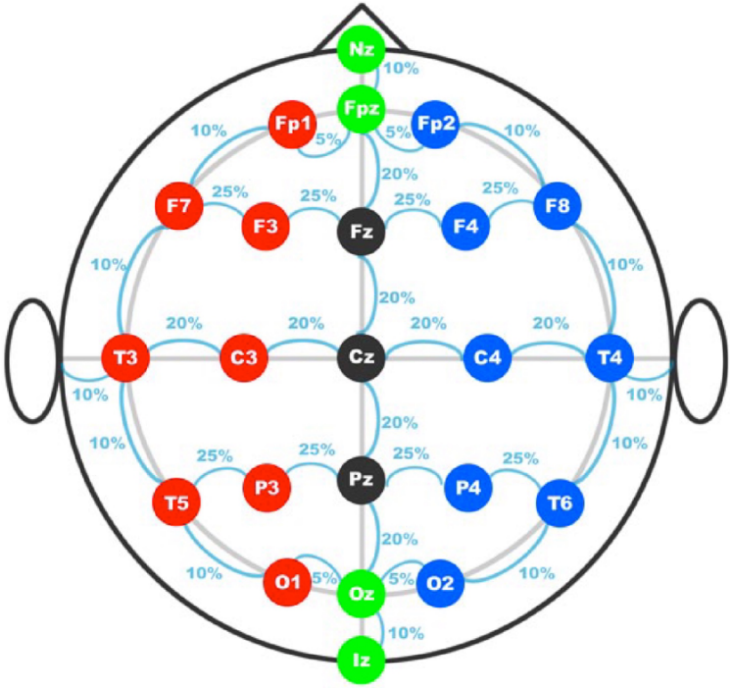
We will generate a dynamical system for each so that we can sample local patch and its corresponding policy

Deep Poincare Map for Robust Medical Image Segmentation. *Y. Mo, F. Liu et al. arXiv 2017.*

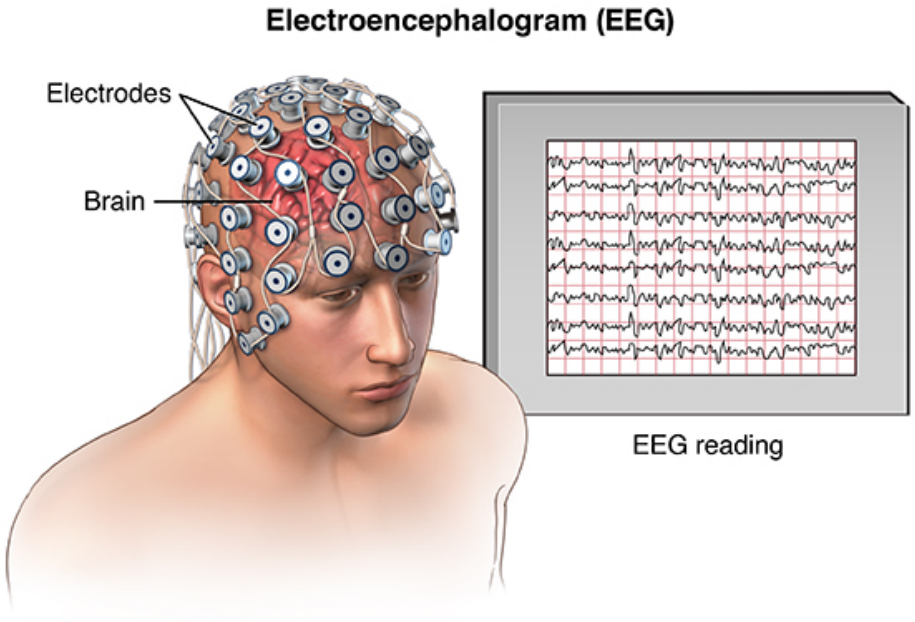
Research - Medical Applications

TensorLayer

Wearable EEG Sleep Scoring



10/20 EEG electrode standard



Multi-channel EEG device

Mixed Neural Network Approach for Temporal Sleep Stage Classification. *H. Dong, A. Supratak et al. TNSRE 2016.*

DeepSleepNet: a Model for Automatic Sleep Stage Scoring based on Raw Single-Channel EEG.
A. Supratak, H. Dong et al. TNSRE 2017.

Research - Medical Applications

TensorLayer

Wearable EEG Sleep Scoring

83~86% accuracy

CONFUSION MATRIX FROM CROSS-VALIDATION USING MNN AND F4-EOG (LEFT) WHEN SEQUENCE LENGTH IS 5. THE LEFT COLUMN IS THE ACTUAL CLASS LABELED BY SLEEP EXPERT, AND THE TOP ROW IS THE PREDICTED CLASS CALCULATED BY THE CLASSIFIER.

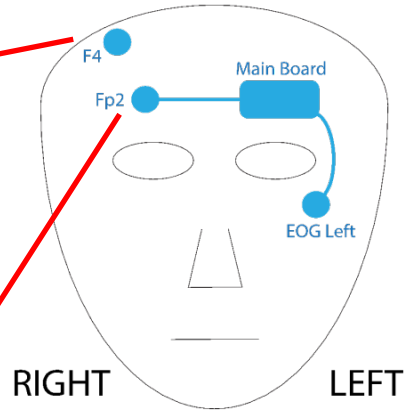
ACC = 85.92% MF1 = 80.50%

SL=5	W	N1	N2	N3	REM	RE	PR	F1
W	5022	577	188	19	395	80.95	88.49	84.55
N1	407	2468	989	4	965	51.07	62.75	56.31
N2	130	630	27254	1021	763	91.46	90.02	90.73
N3	13	0	1236	6399	5	83.61	85.94	84.76
REM	103	258	609	0	9611	90.83	81.87	86.12

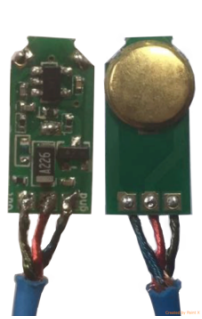
CONFUSION MATRIX FROM CROSS-VALIDATION USING MNN AND *Fp2-EOG (Left)* WHEN SEQUENCE LENGTH IS 5

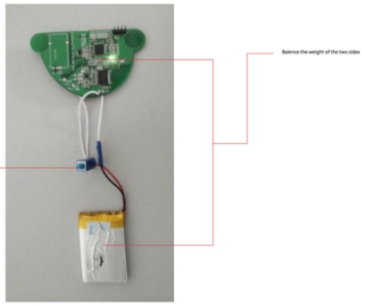
ACC = 83.35% MF1 = 76.97%

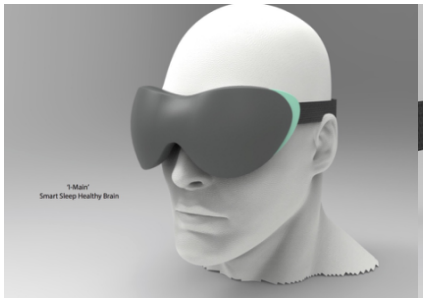
SL=5	W	N1	N2	N3	REM	RE	PR	F1
W	4604	795	294	32	479	74.21	86.04	79.69
N1	405	2208	1292	9	919	45.69	57.08	50.75
N2	208	605	27199	897	889	91.28	86.65	88.91
N3	24	1	1689	5936	3	77.56	86.22	81.66
REM	110	259	914	11	9287	87.77	80.22	83.83

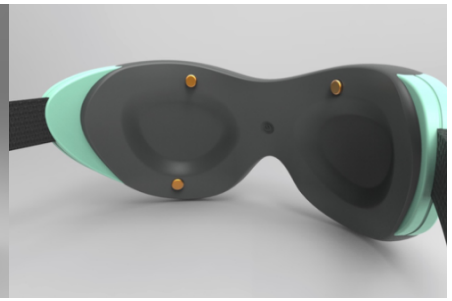


Active Dry Electrode









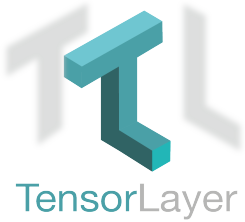
Mixed Neural Network Approach for Temporal Sleep Stage Classification. *H. Dong, A. Supratak et al. TNSRE 2016.*

DeepSleepNet: a Model for Automatic Sleep Stage Scoring based on Raw Single-Channel EEG. *A. Supratak, H. Dong et al. TNSRE 2017.*

A New Soft Material based In-the-ear EEG Recording Technique *H. Dong, PM. Matthews, Y. Guo. EMBC 2016.*

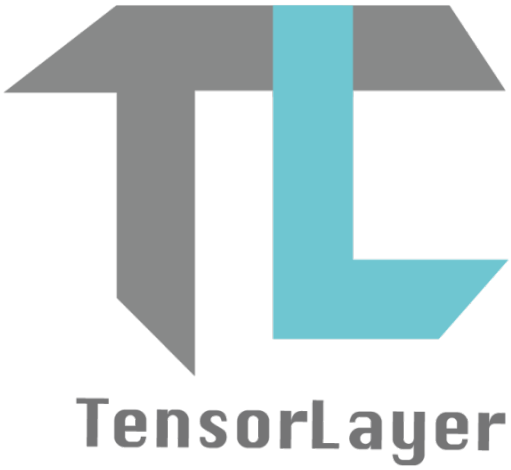
Future

Future



Moving from the current 1.8 to the 2.0 milestone

- Support more domain-specific layers
- Support for integrated command line interface
- Support for distributed training and large dataset
- Support for model compression



Happy Chinese New Year

新春快乐

Imperial College London

GitHub WeChat slack QQ www.QQ.com