

keras-pandas

Brendan Herger, hergertarian.com
<http://keras-pandas.readthedocs.io/>
Slides: <https://goo.gl/wRkXz5>

keras-pandas

keras/examples at master · ker x

Brendan

← → ↻ GitHub, Inc. [US] | https://github.com/keras-team/keras/tree/master/examples ☆ ABP 🔊 🔗 ⚙️ ⋮

Keras examples directory

Vision models examples

[mnist_mlp.py](#) Trains a simple deep multi-layer perceptron on the MNIST dataset.

[mnist_cnn.py](#) Trains a simple convnet on the MNIST dataset.

[cifar10_cnn.py](#) Trains a simple deep CNN on the CIFAR10 small images dataset.

[cifar10_resnet.py](#) Trains a ResNet on the CIFAR10 small images dataset.

[conv_lstm.py](#) Demonstrates the use of a convolutional LSTM network.

[image_ocr.py](#) Trains a convolutional stack followed by a recurrent stack and a CTC logloss function to perform optical character recognition (OCR).

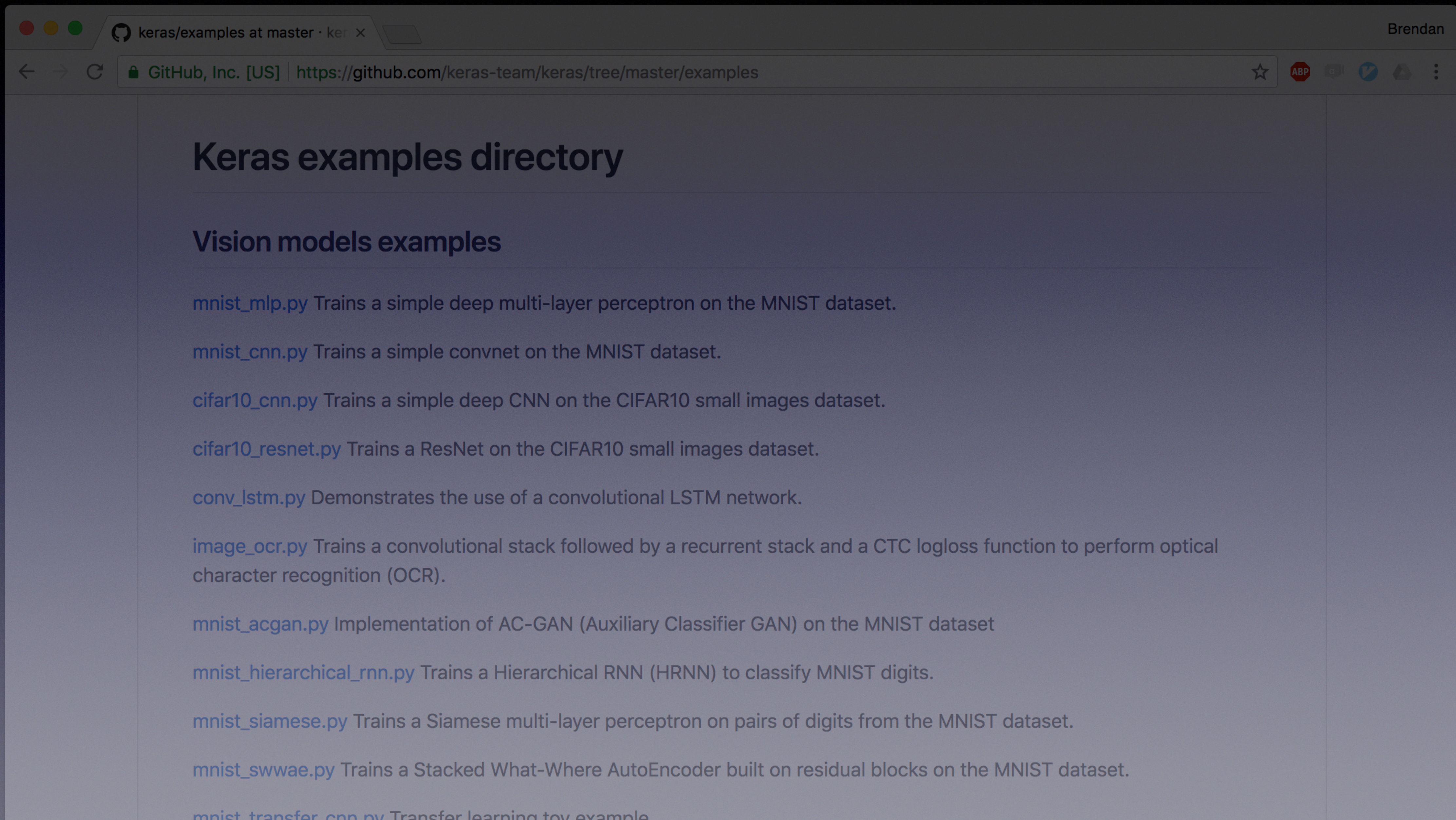
[mnist_acgan.py](#) Implementation of AC-GAN (Auxiliary Classifier GAN) on the MNIST dataset

[mnist_hierarchical_rnn.py](#) Trains a Hierarchical RNN (HRNN) to classify MNIST digits.

[mnist_siamese.py](#) Trains a Siamese multi-layer perceptron on pairs of digits from the MNIST dataset.

[mnist_swwae.py](#) Trains a Stacked What-Where AutoEncoder built on residual blocks on the MNIST dataset.

[mnist_transfer_cnn.py](#) Transfer learning toy example.



keras/examples at master · keras-team/keras

Brendan

← → ↺

GitHub, Inc. [US] | https://github.com/keras-team/keras/tree/master/examples

☆ ABP

Keras examples directory

Vision models examples

[mnist_mlp.py](#)

Trains a simple deep multi-layer perceptron on the MNIST dataset.

[mnist_cnn.py](#)

Trains a simple convnet on the MNIST dataset.

[cifar10_cnn.py](#)

Trains a simple deep CNN on the CIFAR10 small images dataset.

[cifar10_resnet.py](#)

Trains a ResNet on the CIFAR10 small images dataset.

[conv_lstm.py](#)

Demonstrates the use of a convolutional LSTM network.

[image_ocr.py](#)

Trains a convolutional stack followed by a recurrent stack and a CTC logloss function to perform optical character recognition (OCR).

[mnist_acgan.py](#)

Implementation of AC-GAN (Auxiliary Classifier GAN) on the MNIST dataset

[mnist_hierarchical_rnn.py](#)

Trains a Hierarchical RNN (HRNN) to classify MNIST digits.

[mnist_siamese.py](#)

Trains a Siamese multi-layer perceptron on pairs of digits from the MNIST dataset.

[mnist_swwae.py](#)

Trains a Stacked What-Where AutoEncoder built on residual blocks on the MNIST dataset.

[mnist_transfer_cnn.py](#)

Transfer learning toy example.

	keras/examples at master · keras-team/keras · GitHub	Brendan
	GitHub, Inc. [US] https://github.com/keras-team/keras/tree/master/examples	☆ ABP
	<h1>Keras examples directory</h1>	
	<h2>Vision models examples</h2> <hr/> <p>mnist_mlp.py Trains a simple deep multi-layer perceptron on the MNIST dataset.</p>	
	<p>mnist_cnn.py Trains a simple convnet on the MNIST dataset.</p>	
	<h2>Text & sequences examples</h2> <hr/> <p>addition_rnn.py Implementation of sequence to sequence learning for performing addition of two numbers (as strings).</p>	
	<p>image_ocr.py Demonstrates the use of a convolutional + RNN network to perform image OCR.</p>	
	<p>mnist_acgan.py Implementation of AC-GAN (Auxiliary Classifier GAN) on the MNIST dataset</p>	
	<p>mnist_hierarchical_rnn.py Trains a Hierarchical RNN (HRNN) to classify MNIST digits.</p>	
	<p>mnist_siamese.py Trains a Siamese multi-layer perceptron on pairs of digits from the MNIST dataset.</p>	
	<p>mnist_swwae.py Trains a Stacked What-Where AutoEncoder built on residual blocks on the MNIST dataset.</p>	
	<p>mnist_transfer_cnn.py Transfer learning toy example.</p>	

	keras/examples at master · keras-team/keras	Brendan
	https://github.com/keras-team/keras/tree/master/examples	
	<h1>Keras examples directory</h1>	
	<h2>Vision models examples</h2> <hr/> <p>mnist_mlp.py Trains a simple deep multi-layer perceptron on the MNIST dataset.</p>	
	<p>mnist_cnn.py Trains a simple convnet on the MNIST dataset.</p>	
	<h2>Text & sequences examples</h2> <hr/> <p>addition_rnn.py Implementation of sequence to sequence learning for performing addition of two numbers (as strings).</p>	
	<p>image_ocr.py Trains a convolutional stack followed by a recurrent stack and a CTC logloss function to perform optical character recognition (OCR).</p>	
	<h2>Generative models examples</h2> <hr/> <p>lstm_text_generation.py Generates text from Nietzsche's writings.</p>	
	<p>mnist_siamese.py Trains a Siamese multi-layer perceptron on pairs of digits from the MNIST dataset.</p>	
	<p>mnist_swwae.py Trains a Stacked What-Where AutoEncoder built on residual blocks on the MNIST dataset.</p>	
	<p>mnist_transfer_cnn.py Transfer learning toy example.</p>	

Intro
Hands On
Under the hood
Getting Started

Intro

DL is attainable. **keras-pandas** allows users to rapidly build and iterate on deep learning models

- **New users:** Lowering the barrier to entry, good starting point
- **Existing users:** Allows for rapid iteration, good starting point

Hands On

Old way

- **Highly customizable:** Data transformations, data format, input layers
- **Heuristic driven:** Involves high amount of domain expertise, neural network theory, and heuristics
- **Repetitive:** Time consuming & repetitive to create similarly formatted layers

keras-pandas way

- **Less customizable:** Batteries included defaults for each data type
- **Rapid:** Ability to build and iterate on models with a few function calls
- **Maintainable:** More consistent code base, with less redundancy

Getting started

We'll install `keras-pandas`

```
pip install -U keras-pandas
```

```
In [1]: from keras import Model
from keras.layers import Dense

from keras_pandas.Automater import Automater
from keras_pandas.lib import load_titanic
```

```
/Users/brendanherger/anaconda2/lib/python2.7/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 = np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

Data

Let's say we want to look at the the [titanic data set](#)

```
In [2]: observations = load_titanic()
observations.head(2)
```

Out[2]:

	survived	pclass		name	sex	age	siblings_spouses_abor	parents_children_abor	fare
0	0	3		Mr. Owen Harris Braund	male	22.0	1	0	7.2500

Under the hood

What's under the hood?

Every data type has three handlers:

- **Transformations:** A pipeline of SKLearn transformers, to process & format the data
- **Input layers:** Layers that are correctly formatted to accept the input, and learn it
- **Output layer & loss:** A layer that is correctly formatted to accept the output variable, and an appropriate loss

Additional data types

Available now:

- Numerical
- Categorical
- Binary
- Text

Coming soon:

- Time series
- Timestamps
- Support for rolling your own pipelines

Numerical

- **Transformations:** Imputer (null filling) & StandardScaler (normalization / whitening)
- **Input layers:** Standard input layer
- **Output layer & loss:** Single neuron output layer

Categorical


- **Transformations:** CategoricalImputer (null filling) & LabelEncoder (One hot encoding equivalent)
- **Input layers:** Input, Embedding, Flatten
- **Output layer & loss:** One neuron per categorical level

Getting Started

Getting started

- **Example:** Try the titanic example in README.md
- **Docs:** Near total coverage, dive deeper than this talk
- **Get involved:** Actively looking for collaborators & feedback

Getting started

 keras-pandas

latest

Search docs

CONTENTS:

keras-pandas

Quick Start

Usage

Contributing

Contact

Automater

lib

constants

transformations

Read the Docs

v: latest

For more info, check out the:

- Code
- Documentation
- Issue tracker
- Author's website

Quick Start

Let's build a model with the [titanic data set](#). This data set is particularly fun because this data set contains a mix of categorical and numerical data types, and features a lot of null values.

We'll `keras-pandas`

```
pip install -U keras-pandas
```

And then run the following snippet to create and train a model:

```
from keras import Model
from keras.layers import Dense

from keras_pandas.Automater import Automater
from keras_pandas.lib import load_titanic

observations = load_titanic()

# Transform the data set, using keras_pandas
categorical_vars = ['pclass', 'sex', 'survived']
numerical_vars = ['age', 'siblings_spouses_aborad', 'parents_children_aborad', 'fare']
text_vars = ['name']

auto = Automater(categorical_vars=categorical_vars, numerical_vars=numerical_vars, text_vars=text_vars, response_var='survived')
X, y = auto.fit_transform(observations)

# Start model with provided input nub
x = auto.input_nub
```

<https://keras-pandas.readthedocs.io/>

Next steps

- **Time series:** Smart defaults for time series models
- **Time stamps:** Sine / cosine decomposition, etc
- **Iterate:** Hear and respond to user feedback
- **Examples:** Find interesting data sets w/ mixed data types

Thanks!

Brendan Herger, hergertarian.com
<http://keras-pandas.readthedocs.io/>
Slides: <https://goo.gl/wRkXz5>

Thanks!

Appendix

Data types: Binary

(Same as categorical)

- **Transformations:** CategoricalImputer (null filling) & LabelEncoder (One hot encoding equivalent)
- **Input layers:** Input, Embedding, Flatten
- **Output layer & loss:** One neuron per categorical level

Data types:Text

- **Transformations:** EmbeddingVectorizer (custom pre-processing step with text preprocessing and indexing)
- **Input layers:** Input, Embedding, Bi-directional LSTM, Flatten
- **Output layer & loss:** Unsupported

Lessons learned

- See the blog: <https://www.hergertarian.com/cheat-sheet-publishing-a-python-package>

Pipelines

- **Text:** String -> tokens -> embedding -> bidirectional LSTM -> Flatten
- **Numerical:** Whiten -> Dense
- **Categorical:** OHE -> Entity Embedding -> Flatten
- **Boolean:** OHE -> Entity Embedding -> Flatten