Machine Learning Using Tensorflow

Week 5:
Keras and MNIST Handwritings

Shu-Ting Pi, PhD
UC Davis

TensorFlow ™

MNIST Handwriting Numbers

What is MNIST?

Human Handwriting Numbers

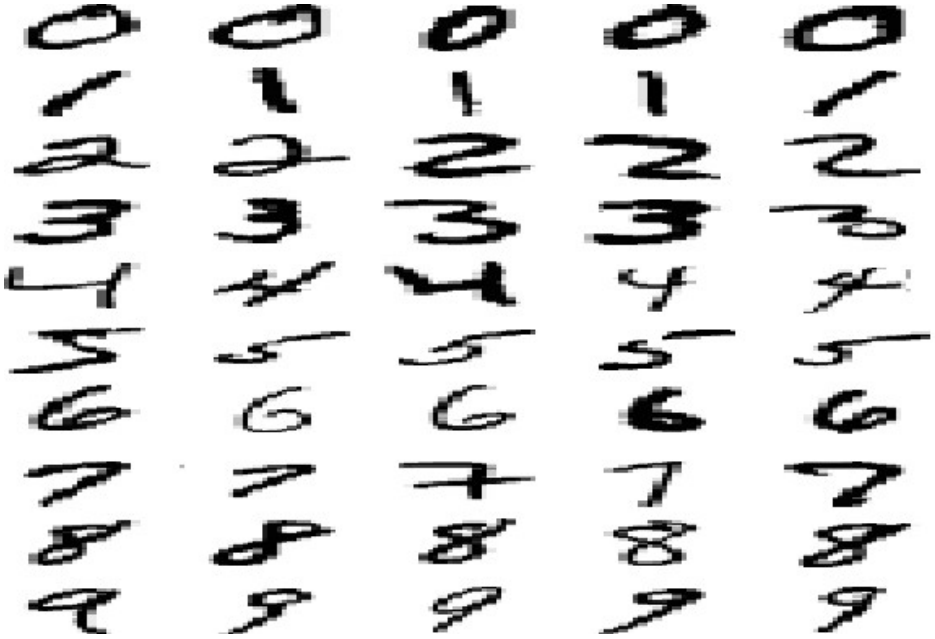
- 60000 train set
- 10000 test set

Each Picture

- 28x28 pixels

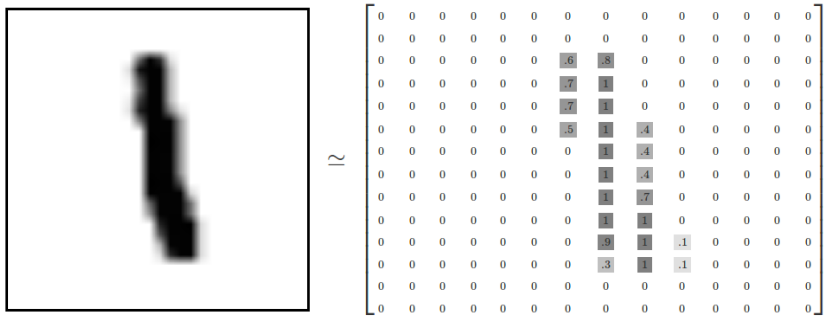
Record High

- 0.83% error via H2O.ai

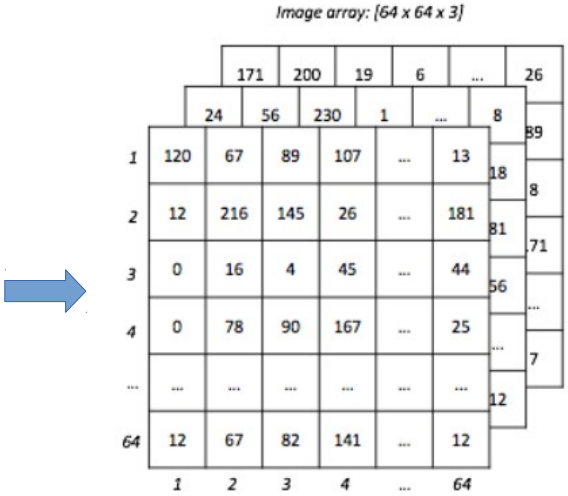


How to deal with pictures?

A figure is essentially a data array

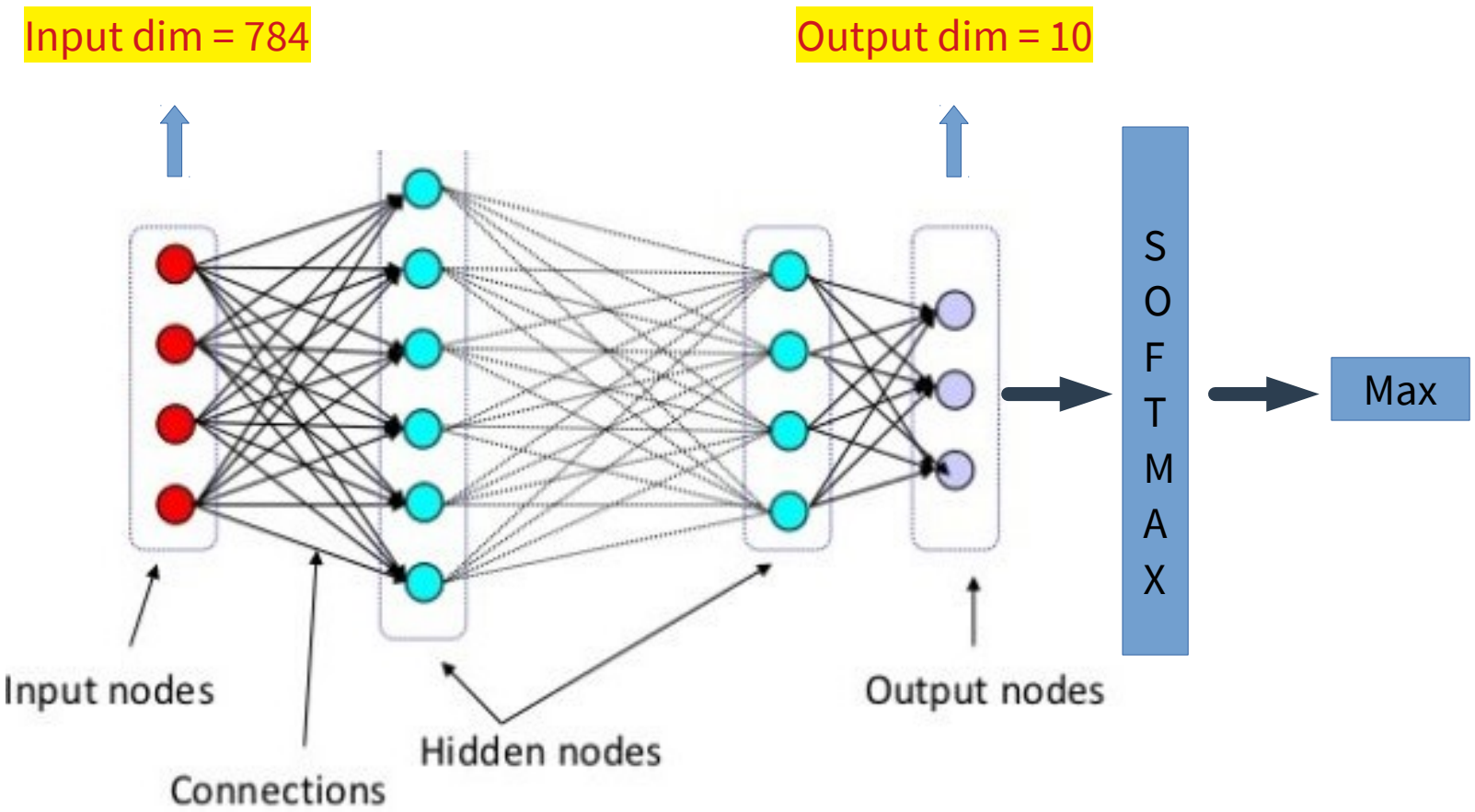


Single Channel



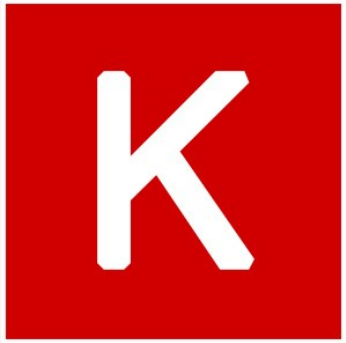
Multi Channel (RGB)

MINST Recognition Using MLP



Keras

What is Keras



High-level API

Fast Growing of Keras

Deep learning libraries: GitHub activity from February 11 to April 12, 2017

new contributors from 2017-02-11 to 2017-04-12

#1:	131	tensorflow/tensorflow
#2:	63	fchollet/keras
#3:	51	pytorch/pytorch
#4:	49	dmlc/mxnet
#5:	18	Theano/Theano
#6:	11	BVLC/caffe
#7:	11	Microsoft/CNTK
#8:	9	tflearn/tflearn
#9:	9	pfnet/chainer
#10:	8	torch/torch7
#11:	5	deeplearning4j/deeplearning4j
#12:	4	NVIDIA/DIGITS
#13:	3	baidu/paddle

new forks from 2017-02-11 to 2017-04-12

#1:	4192	tensorflow/tensorflow
#2:	991	fchollet/keras
#3:	810	BVLC/caffe
#4:	517	deeplearning4j/deeplearning4j
#5:	414	dmlc/mxnet
#6:	307	pytorch/pytorch
#7:	244	Microsoft/CNTK
#8:	211	tflearn/tflearn
#9:	134	torch/torch7
#10:	131	Theano/Theano
#11:	116	baidu/paddle
#12:	88	NVIDIA/DIGITS
#13:	55	pfnet/chainer

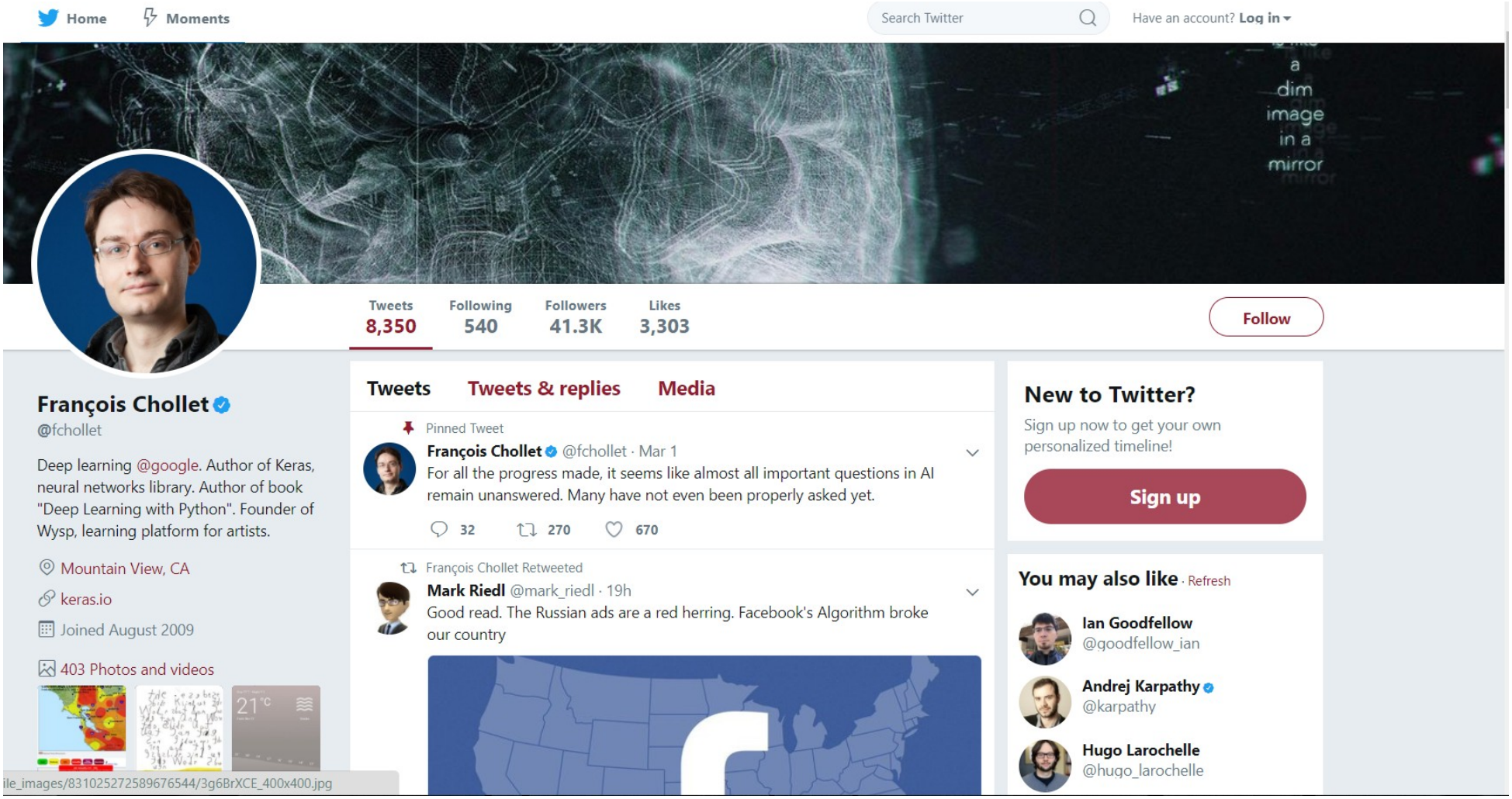
new issues from 2017-02-11 to 2017-04-12

#1:	1175	tensorflow/tensorflow
#2:	568	fchollet/keras
#3:	499	dmlc/mxnet
#4:	286	pytorch/pytorch
#5:	257	Microsoft/CNTK
#6:	239	deeplearning4j/deeplearning4j
#7:	219	baidu/paddle
#8:	173	Theano/Theano
#9:	171	BVLC/caffe
#10:	112	NVIDIA/DIGITS
#11:	84	tflearn/tflearn
#12:	57	pfnet/chainer
#13:	47	torch/torch7

aggregate activity from 2017-02-11 to 2017-04-12

#1:	36.64	tensorflow/tensorflow
#2:	12.52	fchollet/keras
#3:	8.53	dmlc/mxnet
#4:	6.09	BVLC/caffe
#5:	5.92	pytorch/pytorch
#6:	5.12	deeplearning4j/deeplearning4j
#7:	4.12	Microsoft/CNTK
#8:	2.93	Theano/Theano
#9:	2.86	baidu/paddle
#10:	2.17	tflearn/tflearn
#11:	1.68	NVIDIA/DIGITS
#12:	1.38	torch/torch7
#13:	1.12	pfnet/chainer

What is Keras



Official Resource of Keras

Keras Documentation

Search docs

Home

Keras: The Python Deep Learning library

You have just found Keras.

Guiding principles

Getting started: 30 seconds to Keras

Installation

Switching from TensorFlow to CNTK or Theano

Support

Why this name, Keras?

Getting started

Guide to the Sequential model

Guide to the Functional API

FAQ

Models

About Keras models

Sequential

Model (functional API)

Layers

About Keras layers

GitHub Next »

Docs » Home Edit on GitHub

Keras: The Python Deep Learning library

You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

Keras is compatible with: **Python 2.7-3.6**.

Guiding principles

- **User friendliness.** Keras is an API designed for human beings, not machines. I
center. Keras follows best practices for reducing cognitive load: it offers cons
the number of user actions required for common use cases, and it provides cl
user error.
- **Modularity.** A model is understood as a sequence or a graph of standalone, fu

The Keras Blog

Keras is a Deep Learning library for Python, that is simple, modular, and extensible.

Archives Github Documentation Google Group

A ten-minute introduction to sequence-to-sequence learning in Keras

I see this question a lot -- how to implement RNN sequence-to-sequence learning in Keras? Here is a short introduction.

Note that this post assumes that you already have some experience with recurrent networks and Keras.

Fri 29 September 2017
By Francois Chollet
In Tutorials.

What is sequence-to-sequence learning?

Sequence-to-sequence learning (Seq2Seq) is about training models to convert sequences from one domain (e.g. sentences in English) to sequences in another domain (e.g. the same sentences translated to French).

```
"the cat sat on the mat" -> [Seq2Seq model] -> "le chat etait assis sur le tapis"
```

This can be used for machine translation or for free-from question answering (generating a natural language answer given a natural language question) -- in general, it is applicable any time you need to generate text.

There are multiple ways to handle this task, either using RNNs or using 1D convnets. Here we will focus on RNNs.

The trivial case: when input and output sequences have the same length

When both input sequences and output sequences have the same length, you can implement such models simply with a Keras LSTM or GRU layer (or stack thereof). This is the case in this example script that shows how to teach a RNN to learn to add numbers, encoded as character strings:

Installation

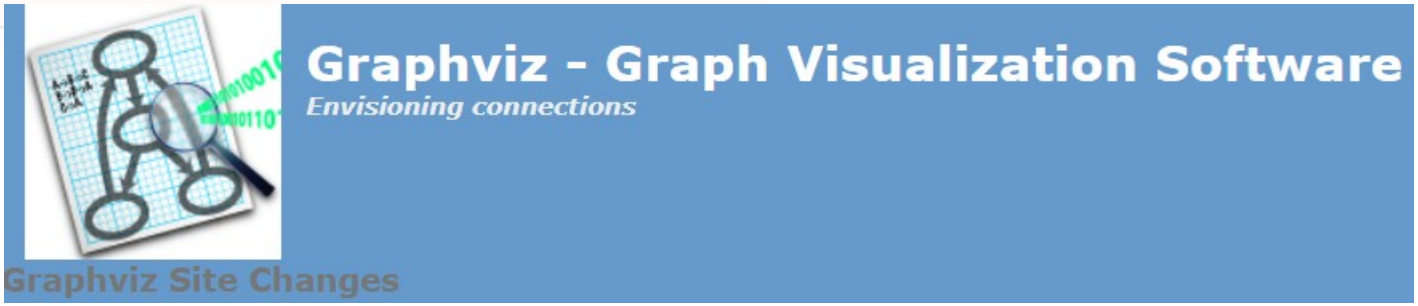
You may also consider installing the following **optional dependencies**:

- cuDNN (recommended if you plan on running Keras on GPU).
- HDF5 and h5py (required if you plan on saving Keras models to disk).
- graphviz and pydot (used by visualization utilities to plot model graphs).

Then, you can install Keras itself. There are two ways to install Keras:

- **Install Keras from PyPI (recommended):**

```
sudo pip install keras
```



Models in Keras

Sequential Model

Using a list

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

Using add method

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(32, input_dim=784))
model.add(Activation('relu'))
model.add(Dense(10))
model.add(Activation('softmax'))
```

Functional API

```
from keras.layers import Input, Dense
from keras.models import Model

# This returns a tensor
inputs = Input(shape=(784,))

# a layer instance is callable on a tensor, and returns a tensor
x = Dense(64, activation='relu')(inputs)
x = Dense(64, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)

# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=predictions)
```

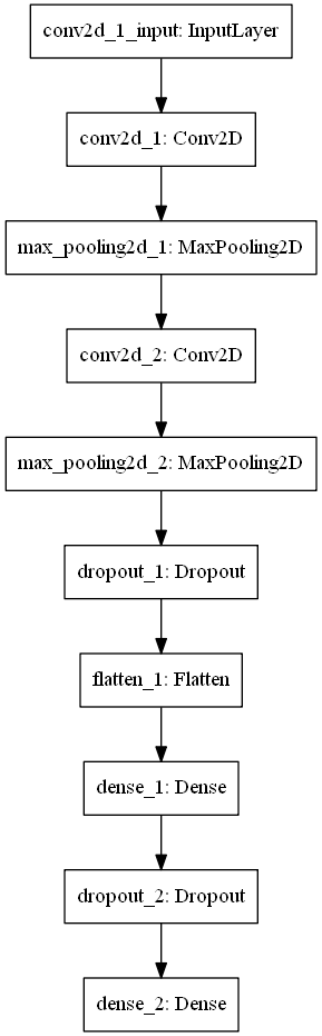
How to plot a model

Prerequisite

- install graphviz
- pip install graphviz, pip install pydot

How to

- from keras.utils import plot_model
- plot_model(model,to_file='model.png')



How to run a model

```
# Train model ========================================
model.compile(loss='categorical_crossentropy',
  optimizer='adam', metrics=['accuracy'])

train_history=model.fit(x=x_train,y=y_train,validation_split=0.2,
                        epochs=10, batch_size=200,verbose=2)
```

model.compile

- define postprocess after output, e.g: loss, optimizer, metrics

model.fit

- perform training
- output will be an object contains train history

save and reload a model

Save model:

- model.save(model,'model.h5')

Reload model:

- from keras.model import load_model
- model=model.load('model.h5')