آموزش یادگیری عمیق
Deep Learning

« راه اندازی بستر کدنویسی »

سعید محققی / دانشگاه شاهد / ۹۹ - ۱۳۹۸

برنامه‌نویسی یادگیری عمیق

۱- راهنمای انتخاب سخت‌افزار

۲- نرم‌افزارهای برنامه نویسی در حوزه یادگیری عمیق

۳- نحوه راه‌اندازی یک بستر نرم‌افزاری

۴- کدهای نمونه

سخت افزار

- **موارد مهم**

1. پردازنده گرافیکی (GPU)
2. پردازنده مرکزی (CPU)
3. حافظه RAM
4. مادربرد (Motherboard)
5. منبع تغذیه (Power)

پردازنده گرافیکی (GPU)

- شرکت NVidia
- انتخاب کارت گرافیکی بر اساس امتیاز Computational Capability

https://developer.nvidia.com/cuda-gpus/

- حداقل امتیاز مورد قبول: 3.5
- امتیاز مناسب: 5 و بالاتر

پردازنده مرکزی (CPU)

- شرکت Intel
- اهمیت کم‌تر به دلیل استفاده از GPU
- رده‌بندی CPU ها

https://www.cpubenchmark.net/

حافظه RAM و مادربرد

- **RAM**
- از نوع DDR4
- حداقل 16 GB

- **Motherboard**
- پشتیبانی از سوکت CPU
- پشتیبانی از تعداد مورد نظر RAM و کارت گرافیکی

منبع تغذیه

- نرم‌افزار آنلاین محاسبه توان مصرفی قطعات کامپیوتر

https://green.ir/calculator

نرم‌افزار

- **زبان‌های برنامه‌نویسی**

1. *Python*
2. *Matlab (>2018)*
3. *C++*
4. *Java*

نرم‌افزار

- **بسترهای کدنویسی**

1. Tensorflow
2. Theano
3. Caffe / Caffe 2
4. Torch / PyTorch

5. CNTK

6. DeepLearning4j

7. MatConvNet

مقایسه بسترهای نرم‌افزاری

Software	Interface	Owner
TensorFlow	Python, C++, Java	Google Research
Theano	Python, C++	Montreal Institute for Learning Algorithms (MILA)
Caffe / Caffe 2	Python, C++, Matlab	Berkeley Vision and Learning Center (BVLC)
Torch / PyTorch	Lua / Python	Ronan Collobert & others
CNTK	Python, C++	Microsoft Research
Deeplearning4j	Java, Scala, C	Skymind
Matlab	Matlab	MathWorks

کتاب‌خانه‌های سطح بالا

Library	Platform
PyLearn2	Theano
Blocks	Theano
Lasagna	Theano
Keras	Theano / TensorFlow / CNTK
TFLearn	TensorFlow
TF-Slim	TensorFlow
TensorLayer	TensorFlow

راه‌اندازی یک بستر کدنویسی

- مشخصات کلی

سیستم عامل	Windows / Linux
زبان برنامه‌نویسی	Python
بستر نرم‌افزاری	TensorFlow
کتاب‌خانه سطح بالا	Keras

پیش‌نیازها

- **CUDA Toolkit** (8.0)
  - *Download: https://developer.nvidia.com/cuda-toolkit/*
- **cuDNN** (5 or 5.1)
  - *Download: https://developer.nvidia.com/cudnn/*
  - *Copy to "C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0"*

راه‌اندازی Python

1. نصب Anaconda برای پایتون ۳
2. اجرای دستور زیر در پنجره command prompt ویندوز:

- conda update conda

  - (نیاز به اتصال به اینترنت)

راه‌اندازی Tensorflow

1. نصب برای CPU
2. نصب برای GPU
3. تست ورژن tensorflow

- conda install tensorflow
- conda install tensorflow-gpu
- conda list tensorflow

- (نیاز به اتصال به اینترنت)

تست import

- اجرای دستورات زیر در پنجره command prompt ویندوز:

› ipython

>> import tensorflow as tf

>> tf.test.is_gpu_available()

داده‌های keras

- دیتاست‌های استاندارد keras در کتاب‌خانه keras.datasets
  - *mnist*
  - *cifar10 / cifar100*
  - *reuters*
  - *imdb*
  - *boston_housing*
- دیتاست‌های استاندارد keras بعد از دانلود در مسیر زیر قرار می‌گیرند
- C:\Users\<username>\.keras\datasets

شروع کدنویسی

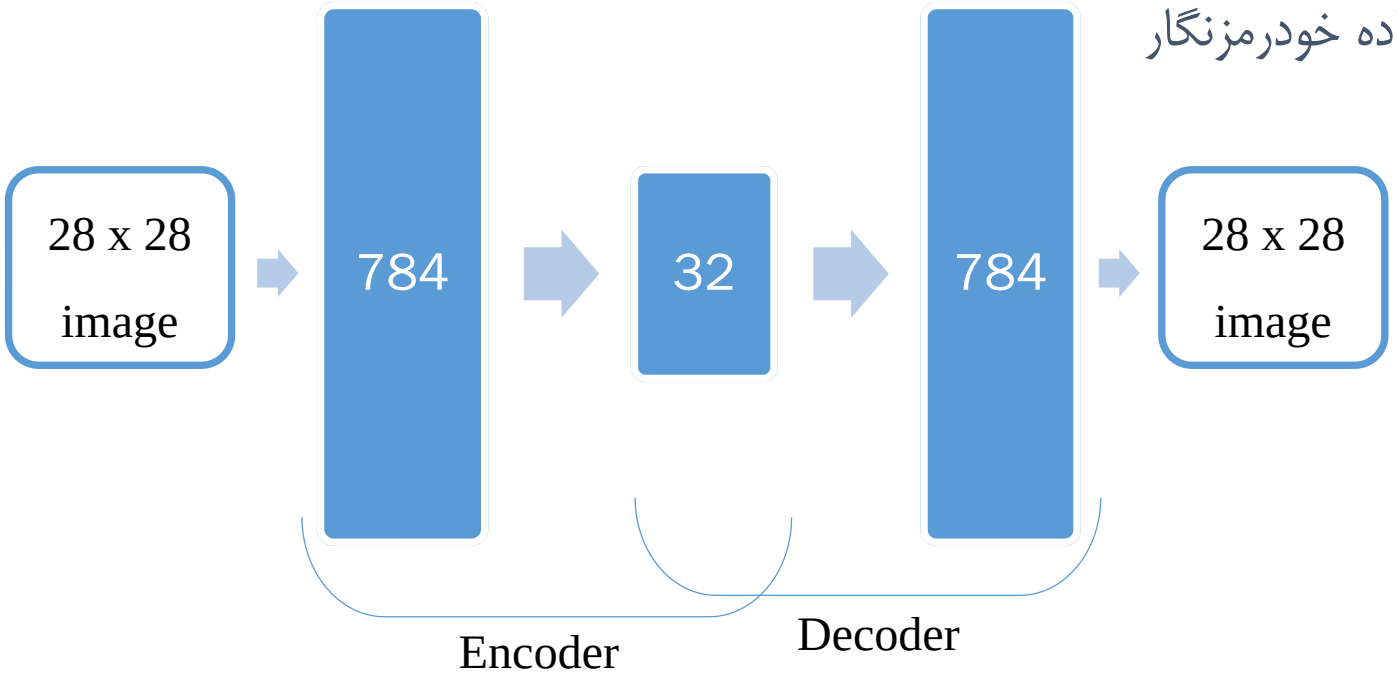
- روش ۱:
  - نوشتن کدها در محیط python در پنجره command prompt
- روش ۲:
  - نوشتن کدها در یک فایل متنی با پسوند .py
  - اجرای فایل از command prompt

شروع کدنویسی

- روش ۳:
  - *کدنویسی و اجرا در محیط Jupyter*
- روش ۴:
  - *کدنویسی و اجرا در برنامه هایی مانند Spyder یا VS Code*

کدنویسی خودرمزنگار با Keras

- مدل ساده خودرمزنگار



کدنویسی خودرمزنگار با Keras

Import کردن توابع و کتاب‌خانه‌های مورد نیاز

```
from tensorflow.keras.layers import Input, Dense

from tensorflow.keras.models import Model

from tensorflow.keras.datasets import mnist

import numpy as np
```

کدنویسی خودرمزنگار در Keras

- بارگذاری داده‌های MNIST

```
(x_train, _), (x_test, _) = mnist.load_data()
```

- در صورتی که فایل mnist.npz در پوشه datasets موجود نباشد در ابتدای اجرا، این فایل دانلود شده و در پوشه datasets ذخیره می‌شود.

ایجاد لایه‌ها

- داده ورودی (۷۸۴ نقطه برای هر تصویر ۲۸ X ۲۸)

```
input_img = Input(shape=(784,))
```

- لایه encoder (۳۲ نورون)

```
encoded = Dense(32, activation='relu')(input_img)
```

- لایه decoder (۷۸۴ نورون)

```
decoded = Dense(784, activation='sigmoid')(encoded)
```

ایجاد مدل

- تعریف مدل

```
autoencoder = Model(input_img, decoded)
```

- کامپایل کردن مدل

```
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

- نمایش یک گزارش از مشخصات و پارامترهای مدل

```
autoencoder.summary()
```

آماده‌سازی داده‌ها

- نرمالیزه کردن مقادیر بین 0 و 1

```
x_train = x_train.astype('float32') / 255.
```

- تغییر ابعاد داده‌ها: (60000, 28, 28) → (60000, 784)

```
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
```

معادل این دستور → `x_train = x_train.reshape(60000, 784)`

- نمایش ابعاد داده‌ها

```
print(x_train.shape)
```

آموزش مدل

- تعداد گام‌ها

```
n_epochs = 10
```

- شروع آموزش

```
autoencoder.fit(x_train, x_train,
                epochs=n_epochs,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))
```

تست کدهای نمونه

- خودرمزنگار کانولوشنی برای حذف نویز تصاویر: “dCAE_keras.py”
- مثال‌های keras در پوشه “keras_examples”
  - *توضیحات هر کد در فایل “README.md”*

پایان