

THEANO AND KERAS TUTORIAL

2015. 12. 2.

서울대학교 컴퓨터공학부 바이오지능연구실

김지섭 박사과정 연구원 jkim@bi.snu.ac.kr






GPU 컴퓨팅의 필요성

- 다루는 문제의 복잡도가 증가할수록 모델이 커지고 보다 많은 연산이 요구됨
- 컨볼루션 연산, 통합 연산은 모두 **행렬 연산**
- CUDA를 사용하여 컨볼루션 신경망을 구현한 경우 CPU에 비해 **약 10배 이상 속도 향상**
 - AlexNet으로 ILSVRC 데이터 학습시 CUDA를 이용한 경우 약 4~5일 정도 소모됨

딥러닝 프레임워크 비교

Accelerated by



	기반 언어	CNN	CUDA	Symbolic 연산	기타 모델 지원
 Decaf / Caffe a Berkeley Vision Project	C++, Protobuf	0	0		
	Lua	0	0		RNN 및 다양한 Optimizer 제공. 기타 기본 ML 라이브러리 제공
	Python	0	0	0	RBM, DBN, AE, LSTM 등 대부분의 딥러닝 모델. 일반적인 확장 가능
	Python, Theano	0	0	0	RBM, DBN, AE, LSTM, GRU 등 최신 모델. 다양한 Activation과 Optimizer 제공
MatConvNet	MATLAB	0	0		
	Python, C++	0	0	0	RNN, Word2Vec 등 몇몇 모델 제공. 기타 모델은 정확히 알려진 바 없음.

THEANO

➤ 개요:

- LISA Lab에서 만든 Python 기반의 오픈소스 Package (<http://deeplearning.net/software/theano/>)
- Symbolic 연산 철학

➤ 장점:

- Symbolic 연산 철학으로 간결하고 빠르게 모델 구현 가능
- Symbolic 미분(Auto-Diff)이 가능하므로 Back-Propagation 등을 직접 구현할 필요가 없음
- 동일한 코드를 CPU와 GPU에서 모두 사용 가능
- Python 기반이므로, numpy, scipy, matplotlib, ipython 등 다양한 python 패키지와의 연동 용이

➤ 단점:

- 에러 메시지가 번잡한 편
- GPU연산의 경우 float만 지원

기본 SYMBOLIC 연산

➤ 예제: $y = 2x^2 + 5x$ 함수의 구현

일반적인 Python	Theano
<pre>def compute(x): y=2*x^2+5*x return y compute(2)</pre>	<pre>x = T.scalar() y = 2*x^2+5*x compute = theano.function([x], y) compute(2)</pre> <p>← Symbolic 변수 정의 ← Symbolic Expression ← 컴파일</p>

SYMBOLIC 미분 연산

➤ 예제: $y = 2x^2 + 5x$ 함수의 미분

일반적인 Python	Theano
<pre>def diff(x): y=4*x+5 return y diff(2)</pre>	<pre>x = T.scalar() y = 2*x^2+5*x y_prime = T.grad(y, x) ← Symbolic 미분 diff = theano.function([x], y_prime) diff(2)</pre>

사람이 직접 미분한 식을 입력해야 함

Symbolic 미분을 통해 자동으로 도함수가 계산됨



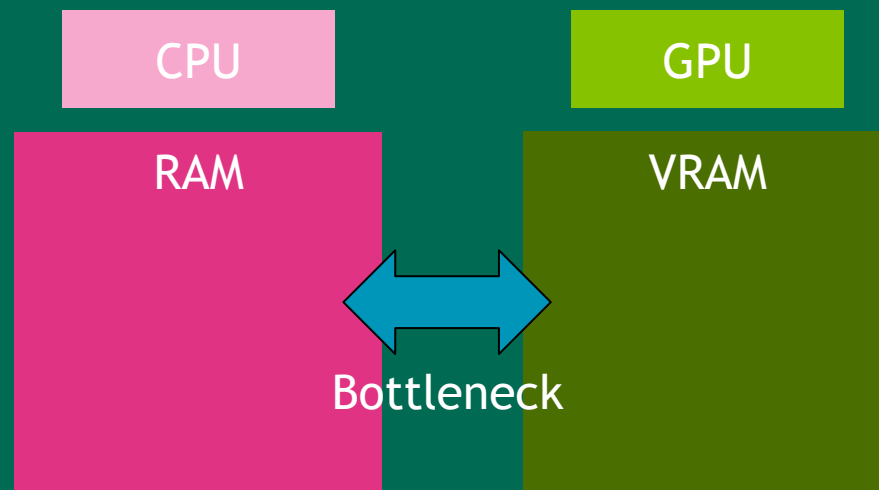
복잡한 Back-Propagation 계산을 직접 구현할 필요가 없음

GPU 연산 관련 문법: SHARED

➤ 기능: VRAM과 RAM 사이의 데이터 전송

➤ `shared_var = theano.shared(numpy_array)`

➤ `numpy_array = shared_var.get_value()`



GPU 연산 관련 문법: GIVENS

- 기능: Symbolic 변수에 **Shared** 데이터를 대입

[예제] $y = 2 * x$ 일때, x 에 10을 대입 계산하는 두 가지 구현 방법

➤ 방법1)

- `compute = theano.function([x], 2*x);`

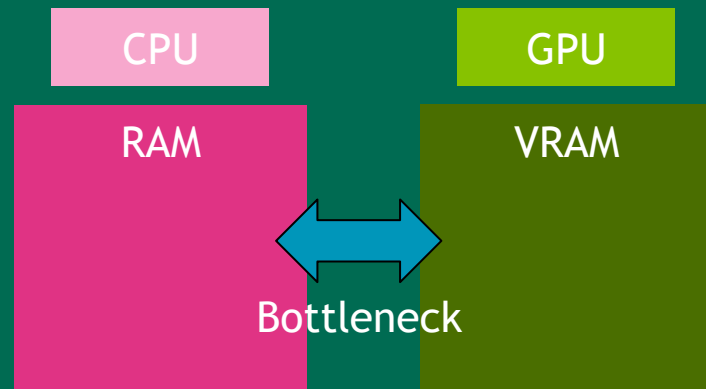
- `compute(10)` ← 실행시 **RAM** → **VRAM** → GPU 연산

➤ 방법2)

- `x_value = theano.shared(10)`

- `compute = theano.function([], 2*x, givens=[(x, x_value)])`

- `compute()` ← 실행시 **VRAM** → GPU 연산



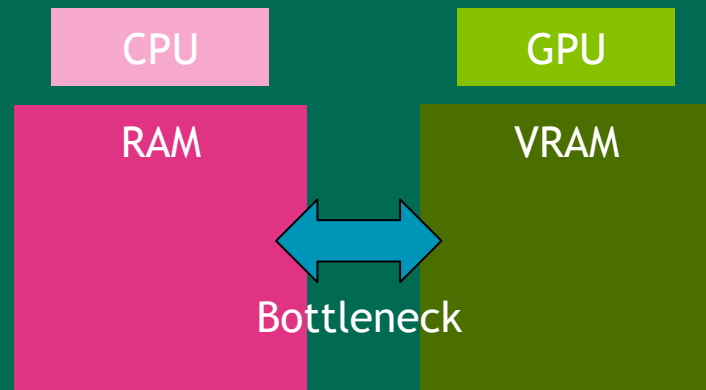
GPU 연산 관련 문법: UPDATES

➤ 기능: GPU연산 결과를 이용해 **Shared** 데이터를 수정

➤ `x_val = theano.shared(0)`

➤ `increase = theano.function([], x_val, updates=(x_val, x_val+1))`

➤ `increase()` ← 실행시 **RAM**을 거치지 않고, GPU내에서 계속 `x_val`을 1씩 증가시킴



THEANO 데모 시연

https://github.com/nzer0/theano_short_demo

KERAS

➤ 개요:

- Theano에 기반한 딥러닝 프레임워크 (<http://keras.io>)
- RBM, DBN, AE, LSTM, GRU 등 최신 모델, 다양한 Activation과 Optimizer 제공

➤ 장점:

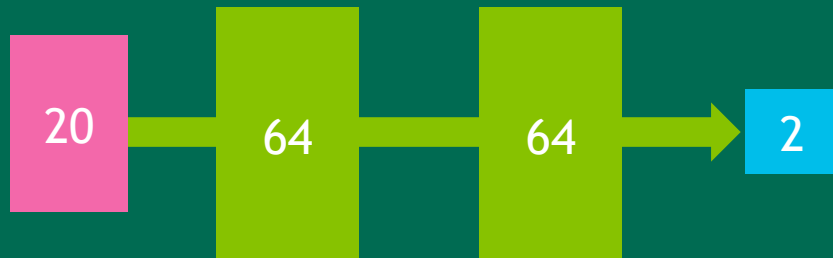
- 블록 조립하듯 간단히 모델을 만들 수 있음 (Caffe, Torch 등과 유사)
- Theano에 기반하므로, 새로운 모델을 추가하기에 비교적 용이
- 학습된 모델을 HDF5 및 JSON 포맷으로 저장 및 로드하는 기능 제공

➤ 단점:

- API 문서가 다소 미비

MLP 예제

```
model = Sequential()  
model.add(Dense(64, input_dim=20, init='uniform',  
activation='tanh'))  
model.add(Dropout(0.5))  
model.add(Dense(64, init='uniform', activation='tanh'))  
model.add(Dropout(0.5))  
model.add(Dense(2, init='uniform', activation='softmax'))  
  
sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)  
model.compile(loss='mean_squared_error', optimizer=sgd)
```



Options:

- Dropout(0.5)
- SGD Optimizer

THANK YOU

Contact me: jkim@bi.snu.ac.kr