CAFFE: Convolutional Architecture for Fast Feature Embedding

Yangqing Jia* , Evan Shelhamer* , Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell, ACM MM'14

2024.10.15

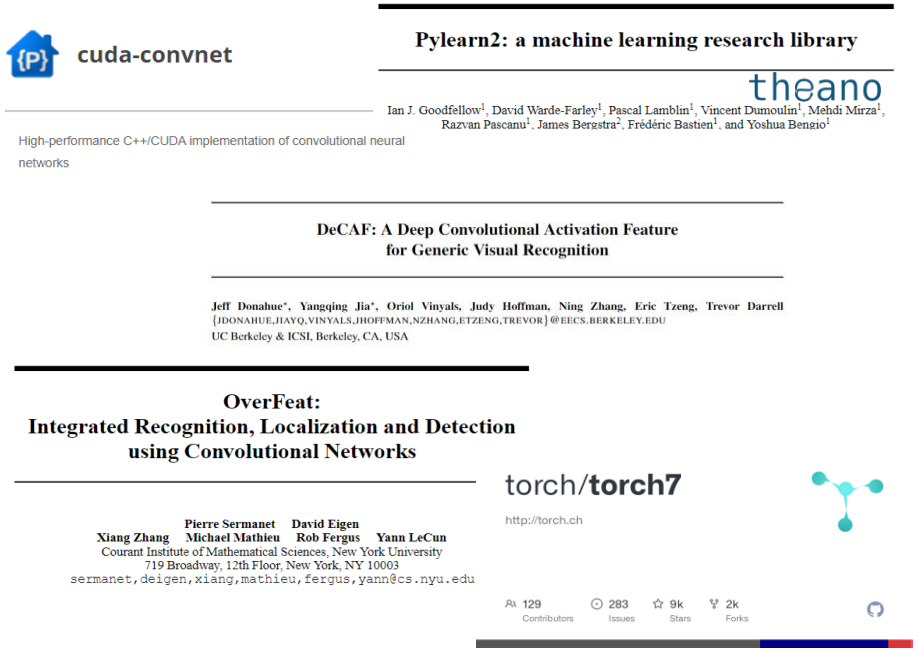
Presentation by Oh, Yeojin

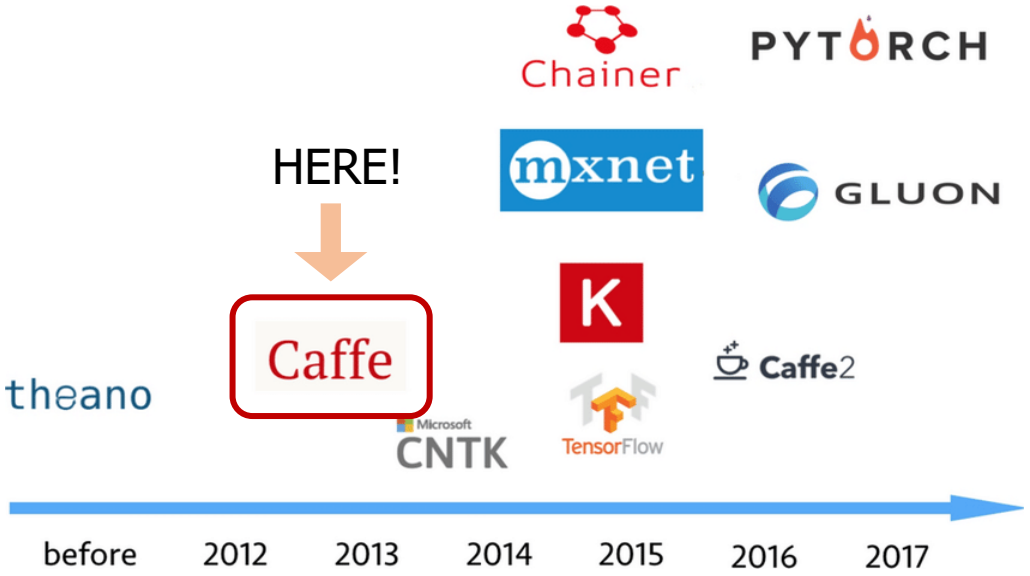
yeojinoh@dankook.ac.kr

Contents

1. Introduction

Deep learning framework in the past…



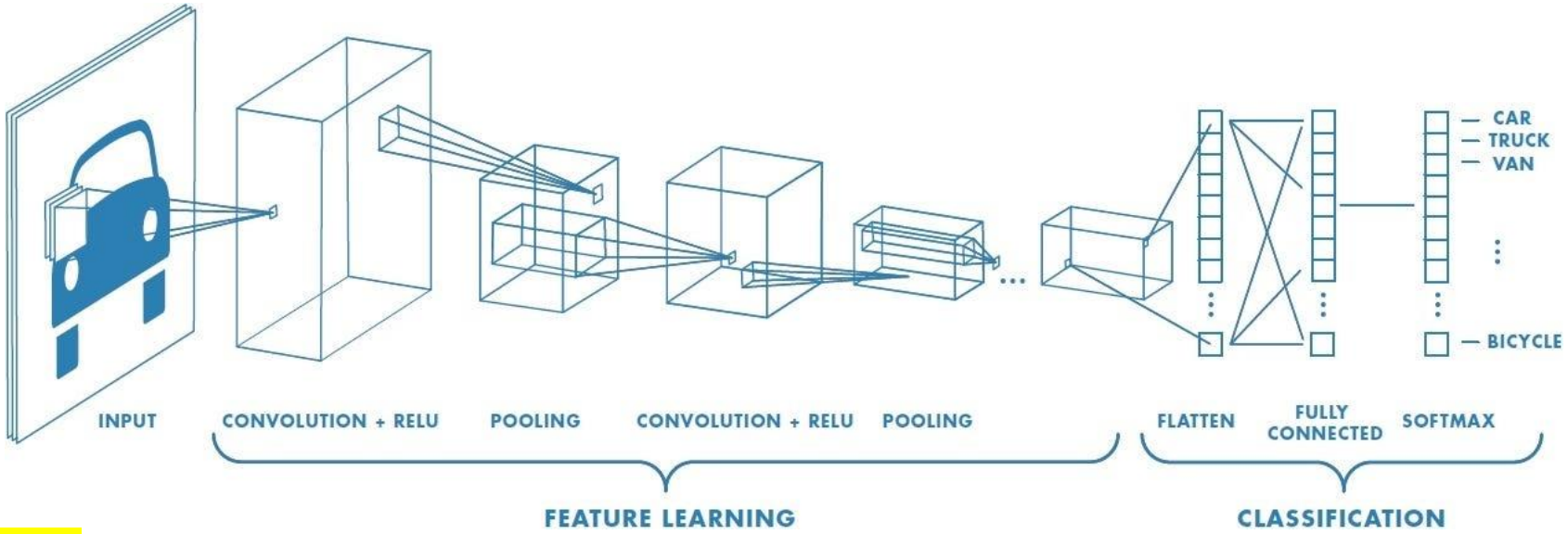


Problem

1. It can be hard for researchers of engineers to replicate published results
2. Released trained models alone can't support rapid research progress or emerging commercial applications
3. Few toolboxes offer truly off-the-shelf deployment of state-of-the-art model
4. Those that do are often not computationally efficient and thus unsuitable for commercial deployment

2. Background

CNN process



1. Convolution filter – Feature extraction from images
2. ReLU (Rectified Linear Unit) – Addition of nonlinearity
3. Pooling – Reduce the spatial dimensions of the input and enhance feature invariance
4. Fully connected – Multiply the input vector by the weights to compute the output
5. Softmax – Derive the prediction based on the probability values to determine which class the image belongs to
6. Loss Function – Calculate the difference between the model's prediction and the actual ground truth
7. Backward Propagation – Calculate the gradient for each layer and update the weights using methods like SGD

3. Motivation

Comparison to related software

Framework	License	Core language	Binding(s)	CPU	GPU	Open source	Training	Pretrained models	Development
Caffe	BSD	C++	Python, MATLAB	✓	✓	✓	✓	✓	distributed
cuda-convnet [7]	unspecified	C++	Python		✓	✓	✓		discontinued
Decaf [2]	BSD	Python		✓		✓	✓	✓	discontinued
OverFeat [9]	unspecified	Lua	C++,Python	✓				✓	centralized
Theano/Pylearn2 [4]	BSD	Python		✓	✓	✓	✓		distributed
Torch7 [1]	BSD	Lua		✓	✓	✓	✓		distributed

- **Cuda-convnet**
  - A CNN training framework utilizing GPU acceleration for training models
- **OverFeat**
  - Focused on inference using pretrained models
- **Theano/Pylearn2, Torch7**
  - No automatic CPU-GPU switching
  - focused on training, so the importance of pretrained models is low

4. Design - Caffe

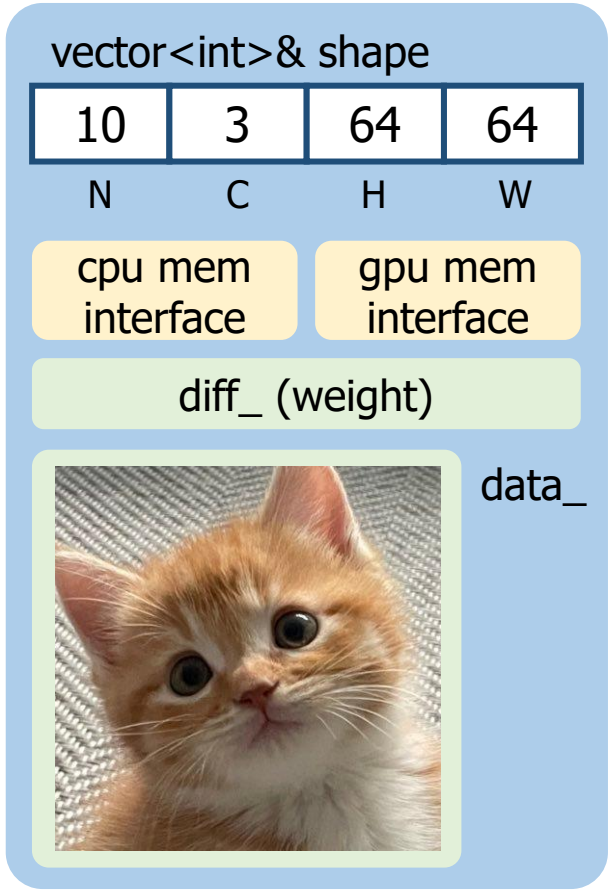
Highlights of CAFFE

- **Modularity**
  - It allows easy extension to new data formats, network layers, and loss function
- **Separation of representation and implementation**
  - Caffe model definitions – config files written by protocol Buffer language
  - Caffe architecture – Network architectures in form of Directed Acyclic Graph (DAG)
  - It can Switch between a CPU and GPU with just one function call
- **Test coverage**
  - Every single module in Caffe has a test, and no new code is accepted into the project without corresponding tests
- **Python and MATLAB bindings**
  - For rapid prototyping and interfacing with existing research code
- **Pre-trained reference models**
  - It provides reference models for visual tasks for reproducible research

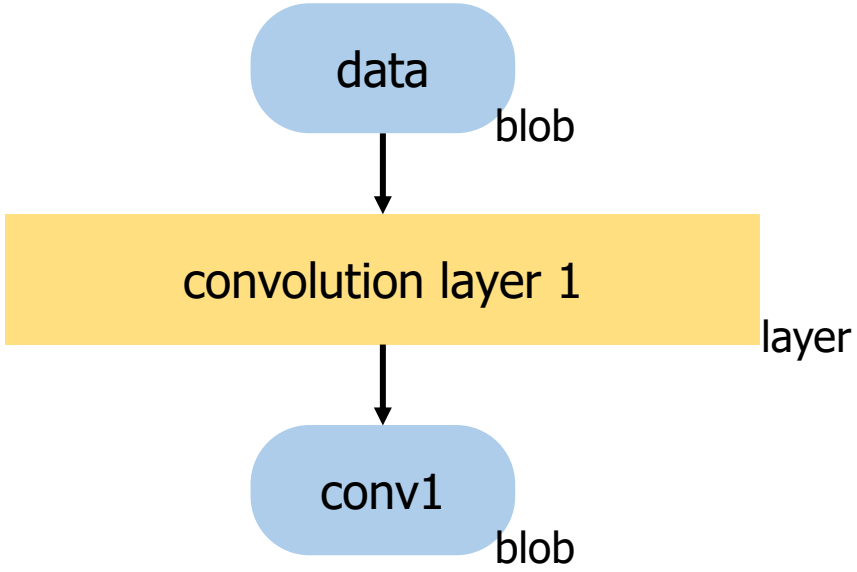
4. Design - Caffe

Architecture of CAFFE – Data Storage(blob)/layers

N: number of image batch
C: number of channel (RGB = 3 or Greyscale = 1)
H,W: pixel height and width



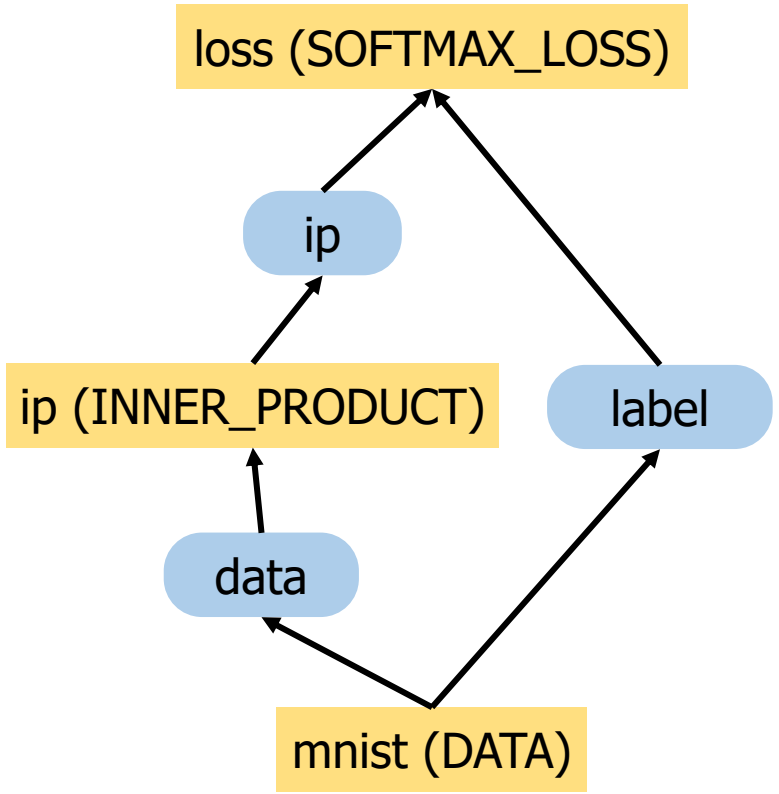
Blob structure



Example of operation

4. Design - Caffe

Networks and Run Mode



Directed acyclic graph (DAG)

- Training a Network
  - Stochastic Gradient Descent (SGD) Algorithm
- Vital to training
  - Learning rate decay schedules
  - Momentum
  - Snapshots for stopping and resuming
- Fine-tuning
  - The adaption of an existing model to new architectures or data
- Run mode
  - The same blob can be executed on either the CPU or GPU
  - Each layer has a computation routine for both CPU and GPU

5. Applications and Examples

Object Classification

Figure 2: An example of the Caffe object classification demo. Try it out yourself online!

Learning Semantic Features

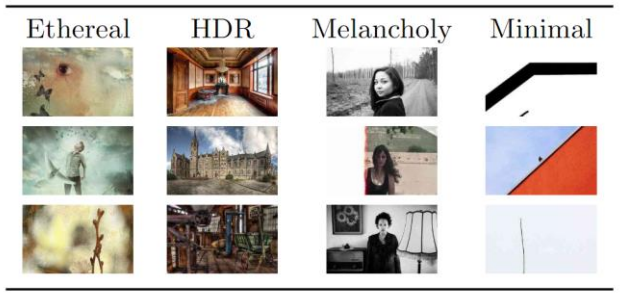


Figure 4: Top three most-confident positive predictions on the Flickr Style dataset, using a Caffe-trained classifier.

Object Detection

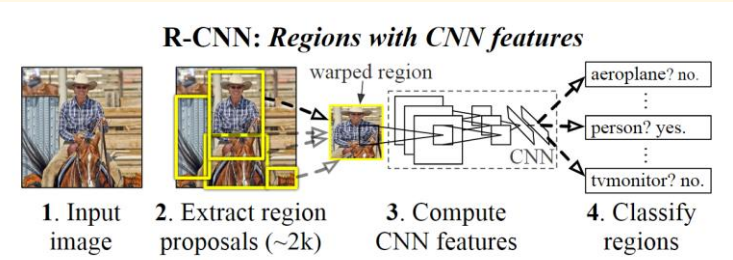
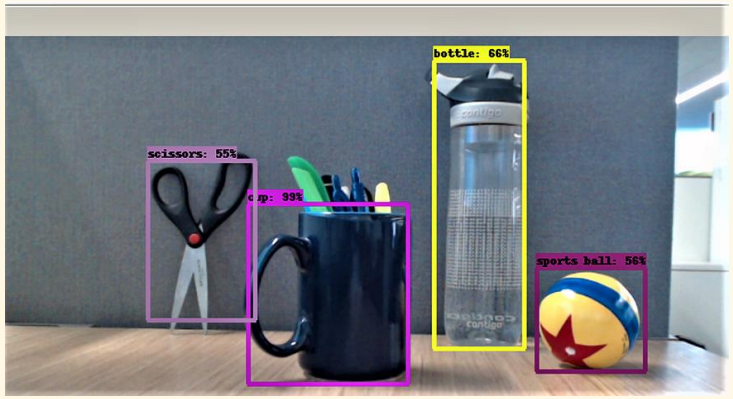


Figure 5: The R-CNN pipeline that uses Caffe for object detection.



Example

6. Conclusion

- Deep Learning training frameworks can be actively applied in research
- The separation of representation and implementation enables smooth CPU-GPU transitions on heterogeneous platforms
- Adopted and improved in various fields beyond vision recognition

Thank you!
Q & A ?

2024.10.15
Presentation by Oh, Yeojin
Yeojinoh@dankook.ac.kr