Dynamic Deep Learning

A paradigm shift in AI research and Tools

Soumith Chintala

Facebook AI Research

Overview

Examples in products and research → A Dynamic Trend → Tools for AI *keeping up with change*

PYTORCH

Chainer

mxnet

theano

Caffe

K

Examples of AI Today

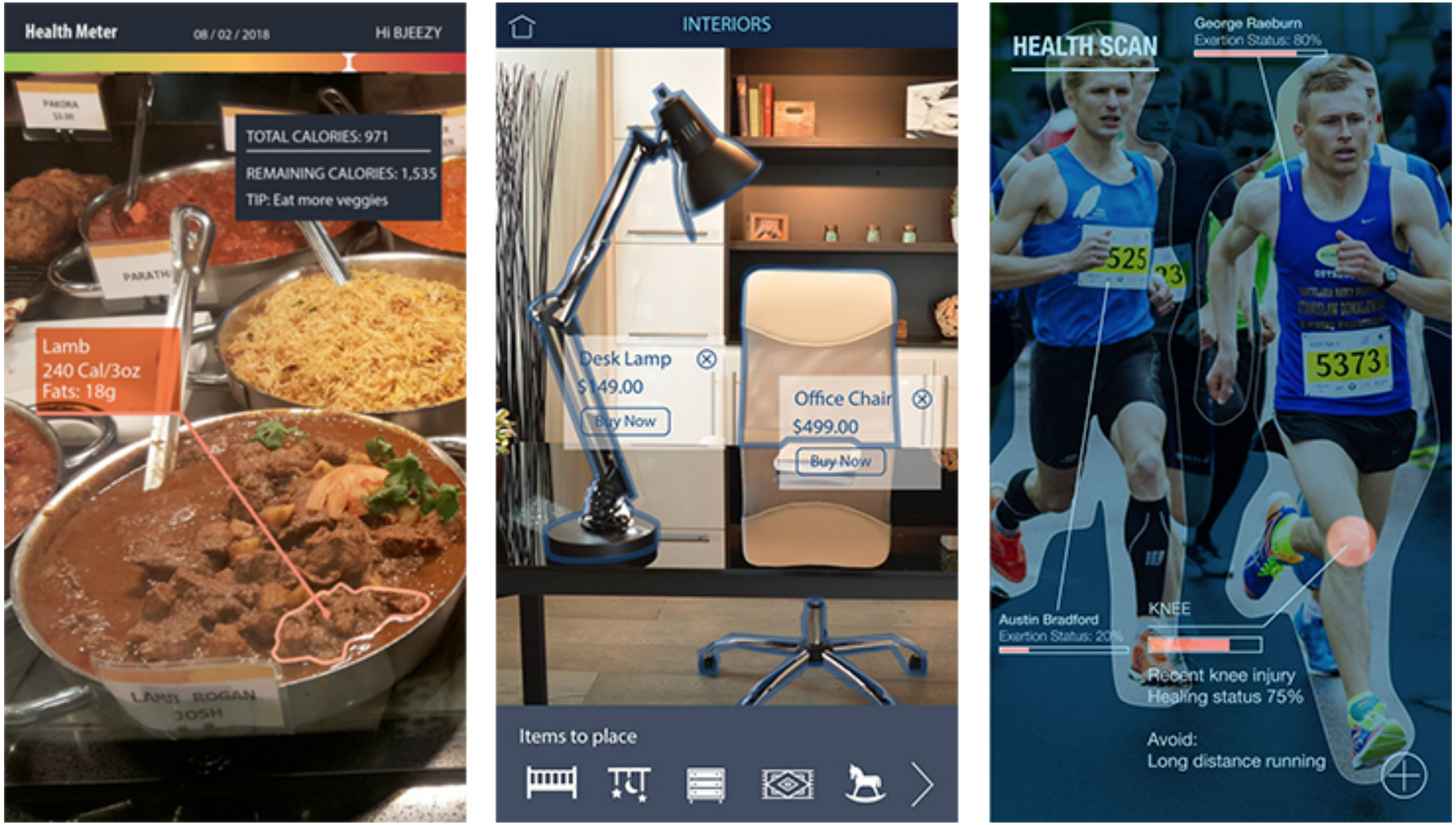
Captioning

Examples → Trends → Tools for AI

Self Driving Cars

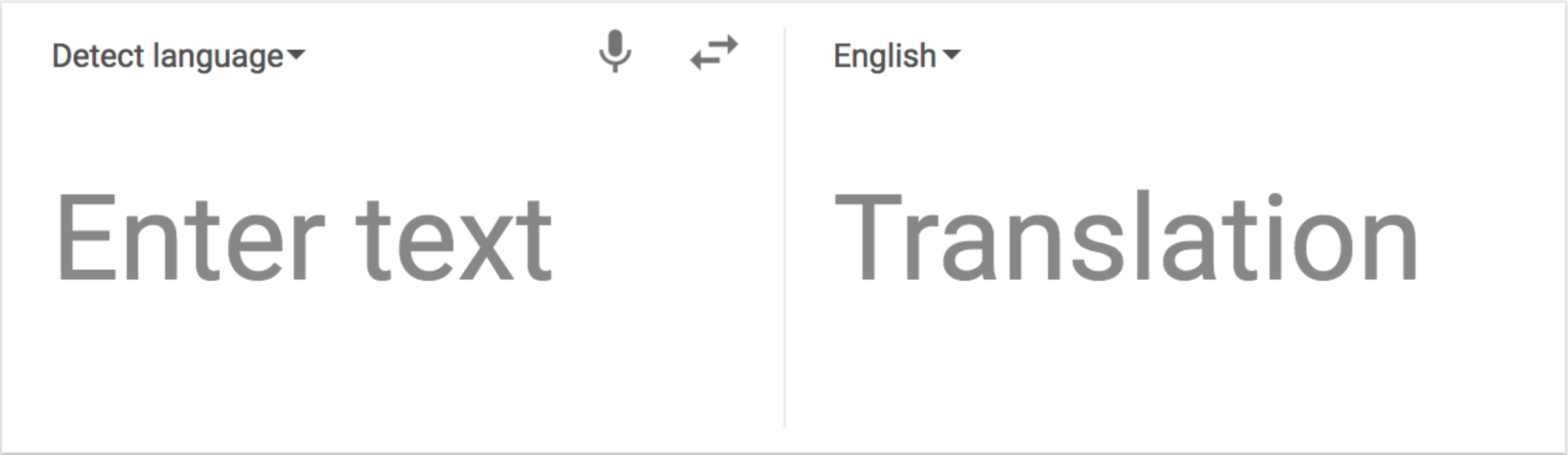
Examples → Trends → Tools for AI

Smart Apps



Examples → Trends → Tools for AI

Machine Translation

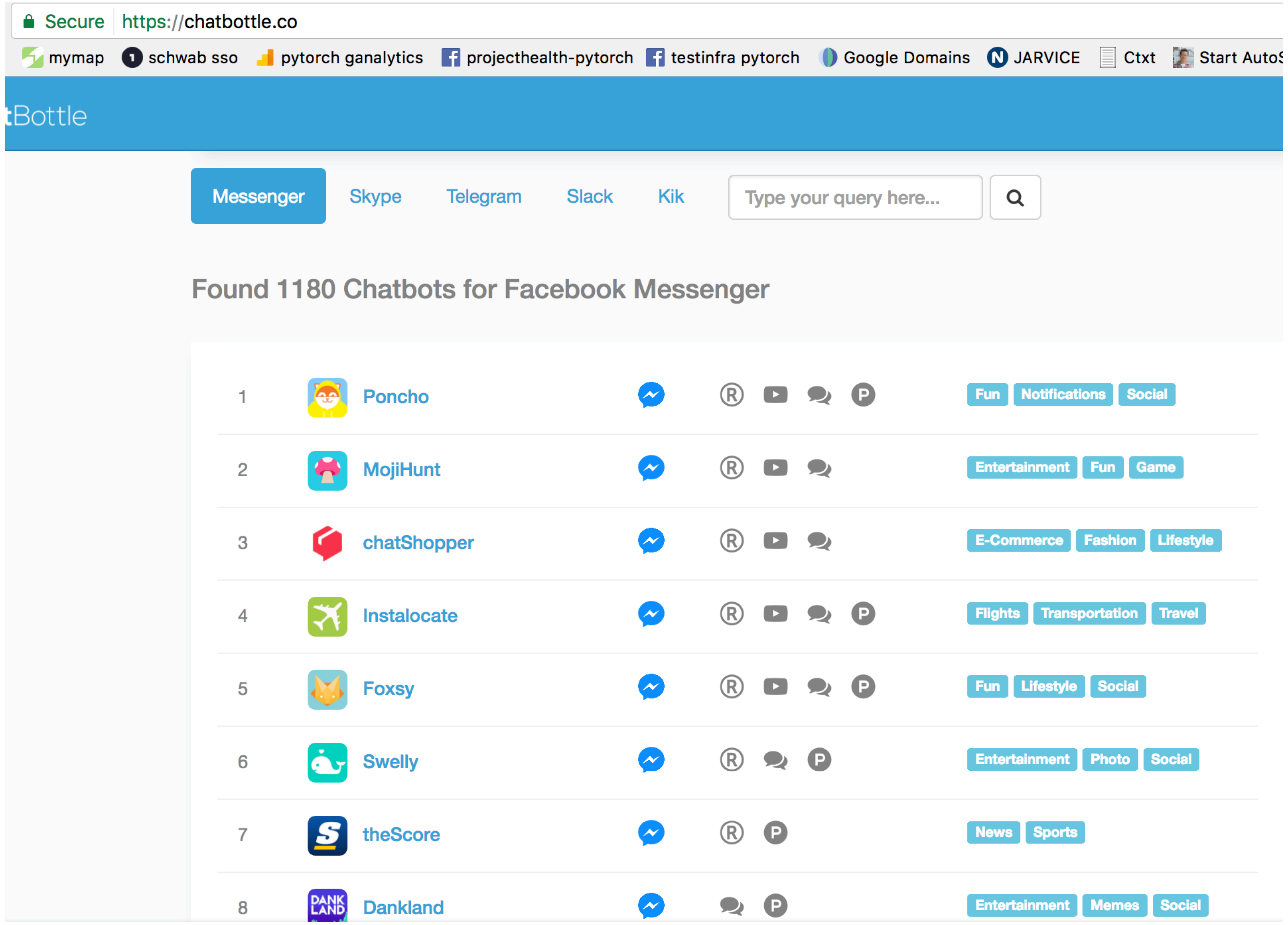


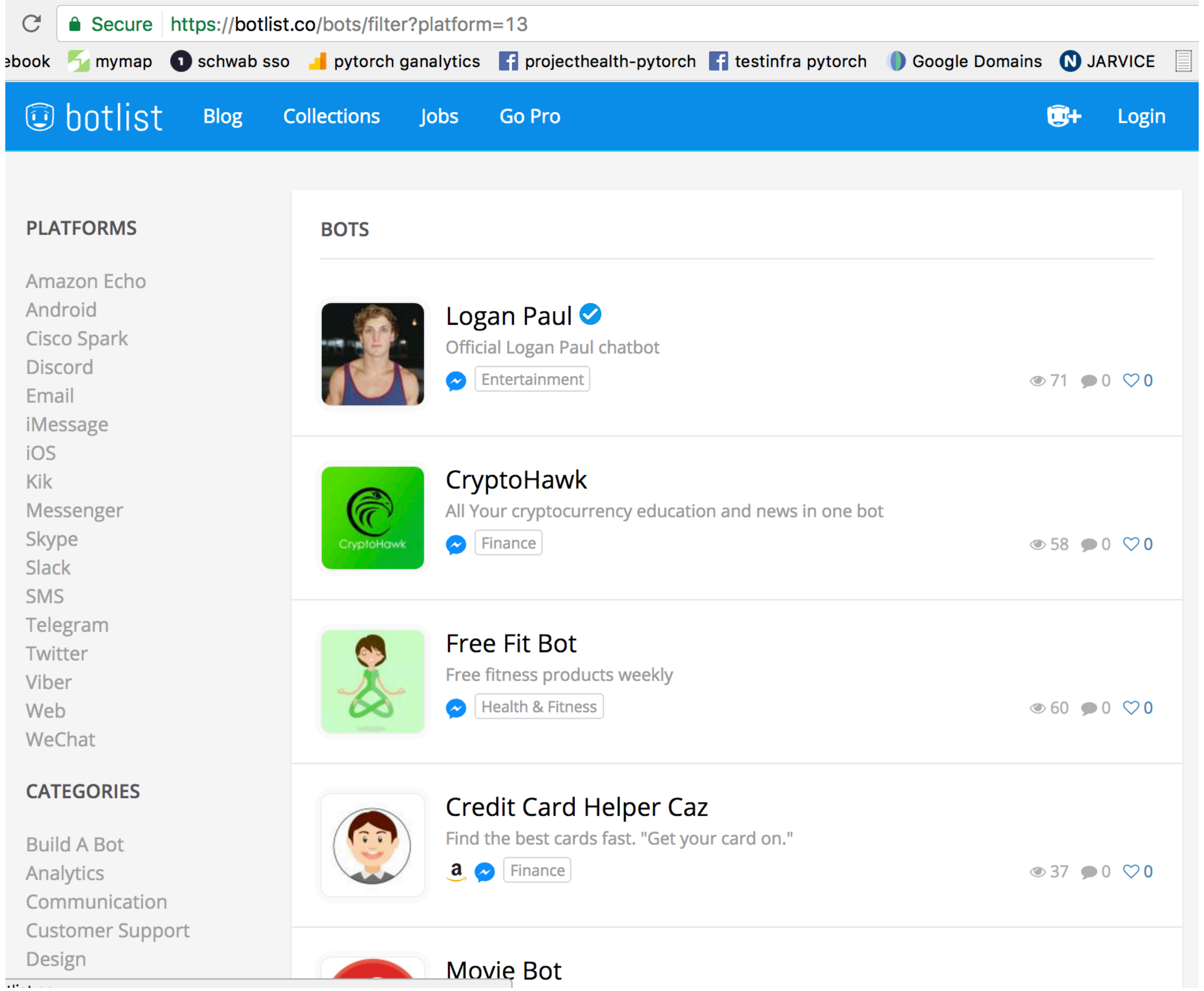
Google Translate

BING TRANSLATOR

Examples → Trends → Tools for AI

Chatbots





Examples → Trends → Tools for AI

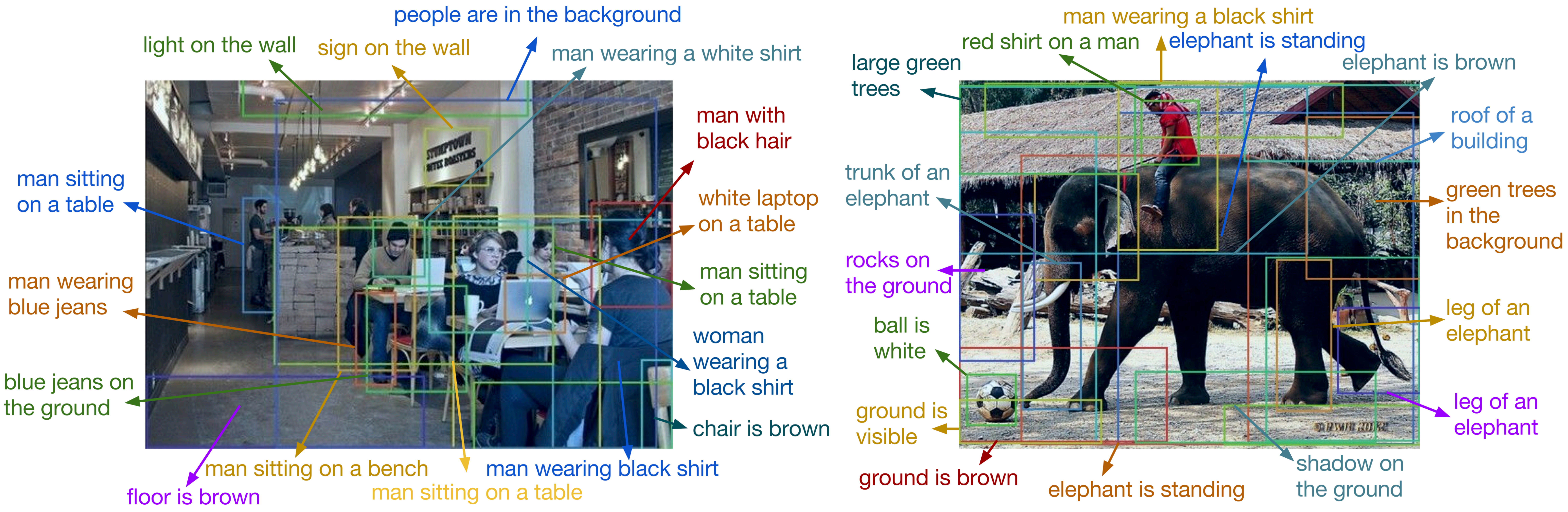
Image Understanding

SharpMask - Piotr Dollar & team

Image Understanding

DenseCap by Justin Johnson & group

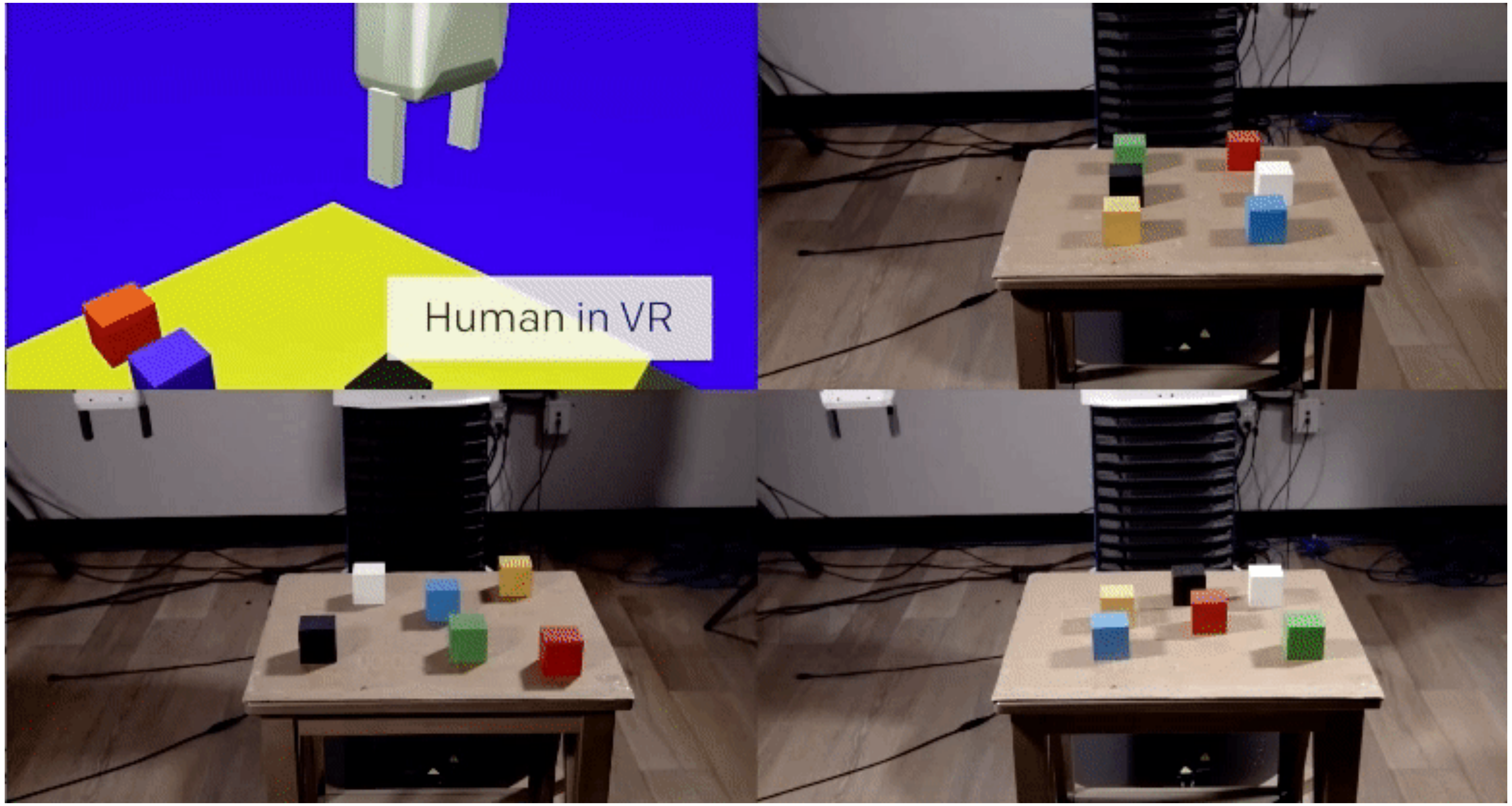
https://github.com/jcjohnson/densecap



Examples → Trends → Tools for AI

Robotics

One-shot imitation learning - Duan et. al. at OpenAI

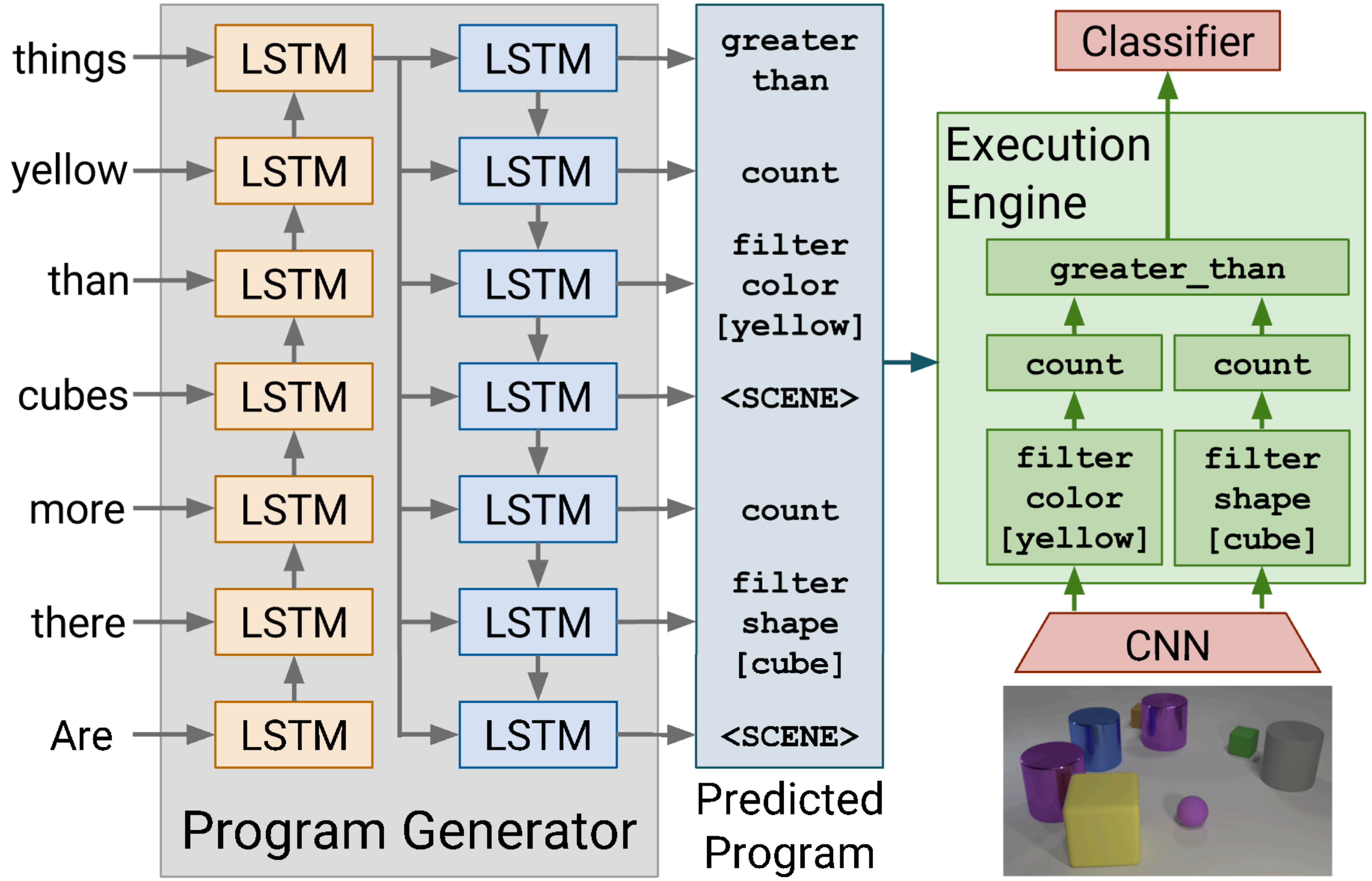


Examples → Trends → Tools for AI

Question Answering

Inferring and Executing Programs for Visual Reasoning
- Johnson et. al. at Facebook

Question: *Are there more cubes than yellow things?* **Answer**: *Yes*



Examples → Trends → Tools for AI

Memory Augmented

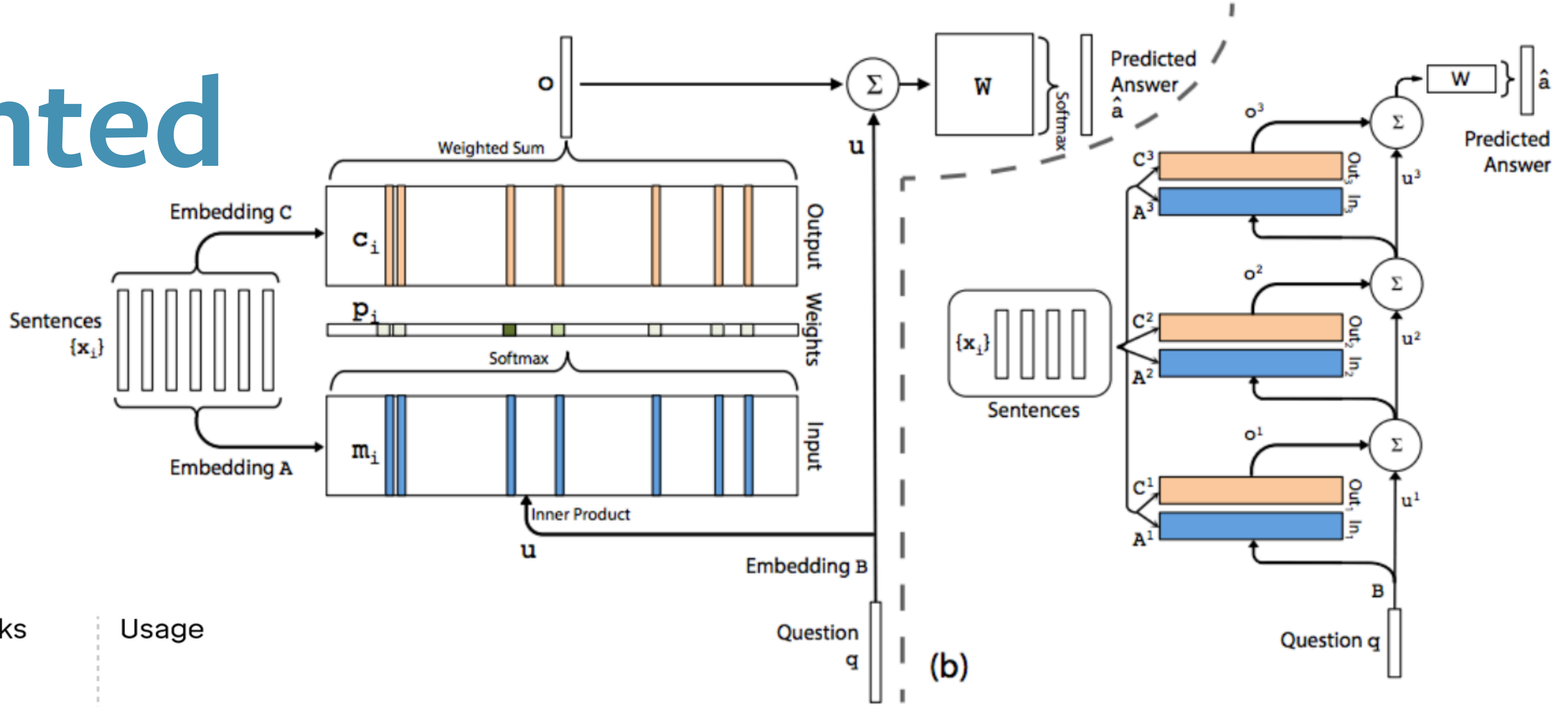
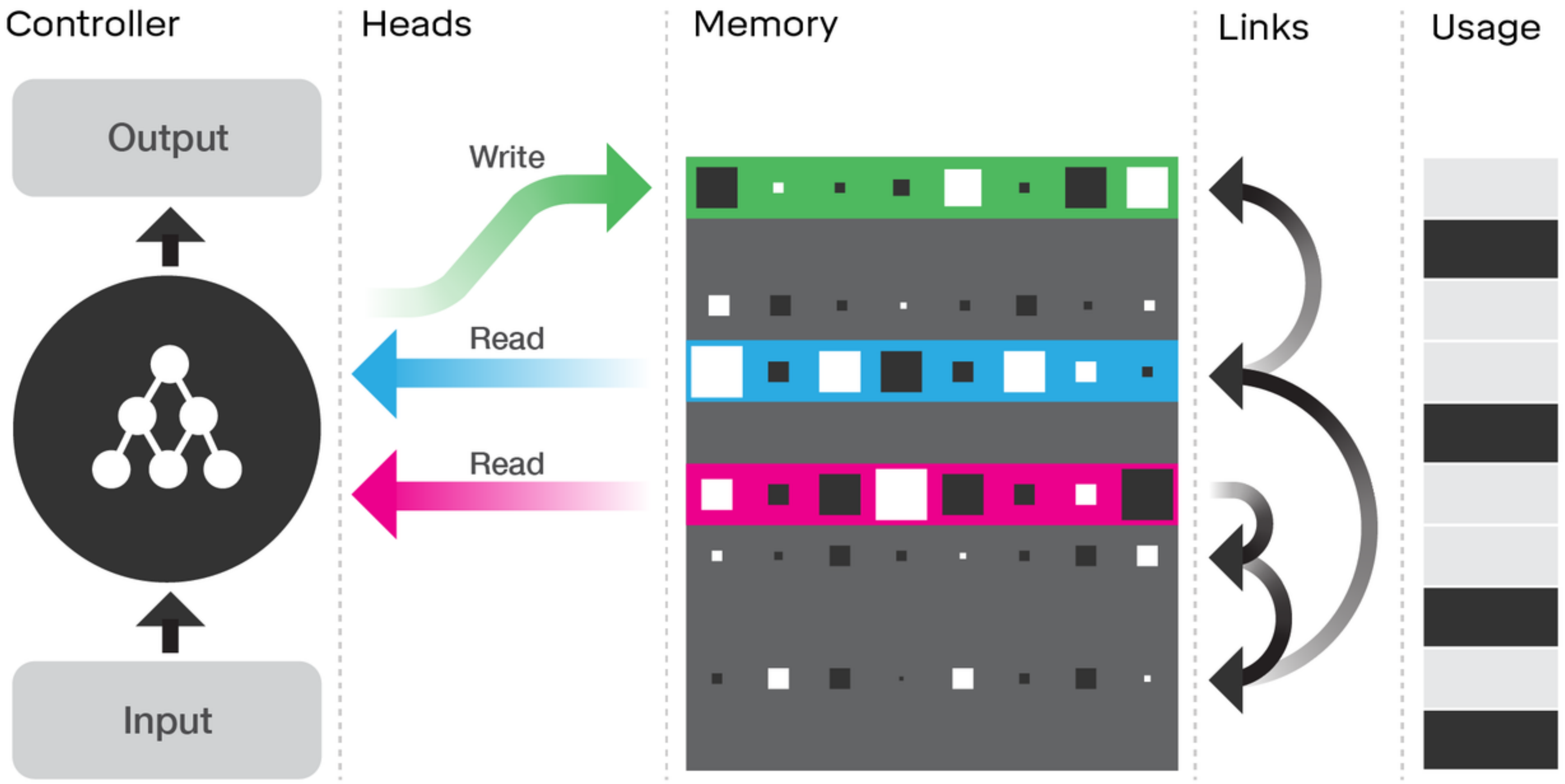


Illustration of the DNC architecture

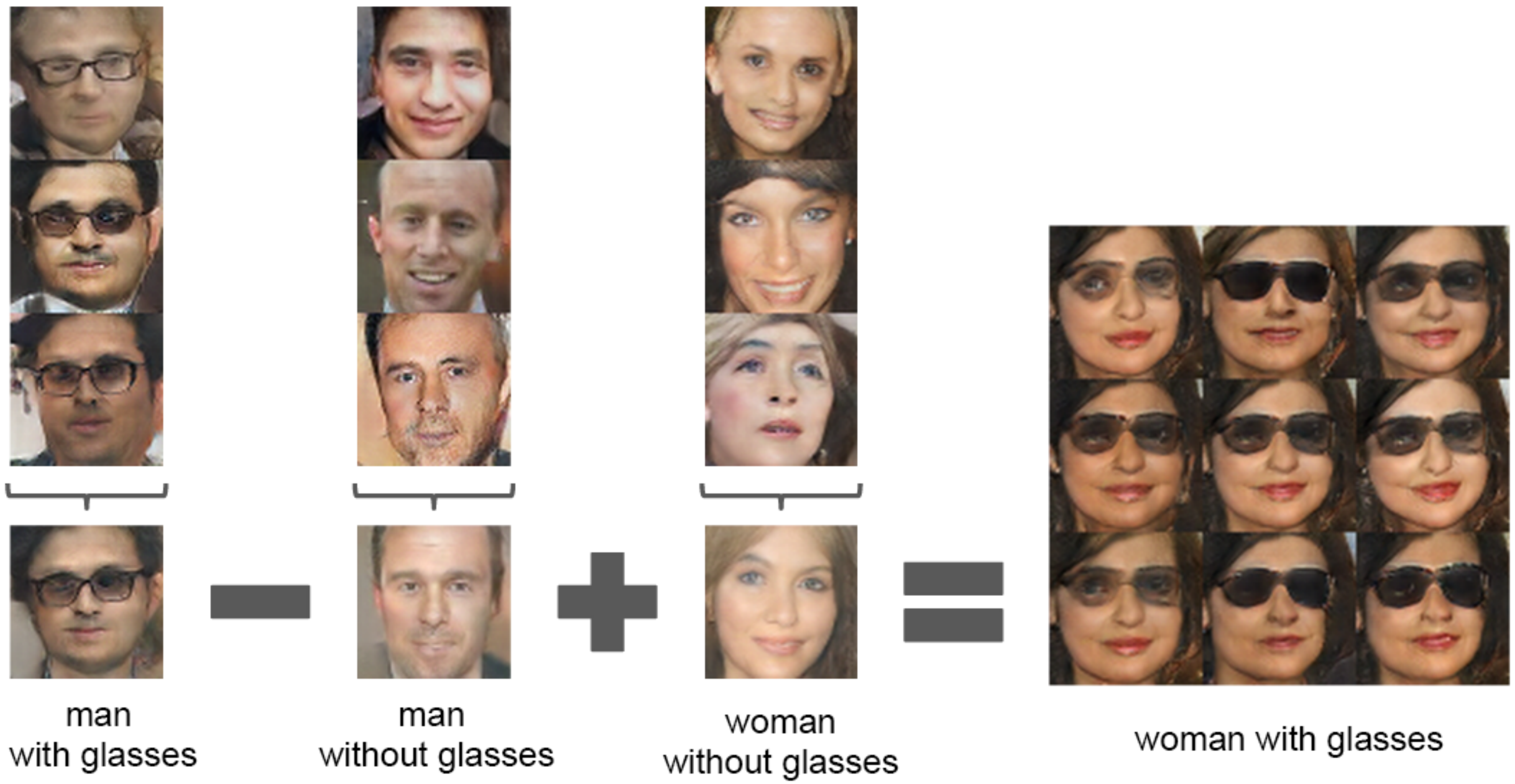


- Memory Networks - Facebook
- Differentiable Neural Computer - Deepmind
-

Examples → Trends → Tools for AI

Adversarial Networks

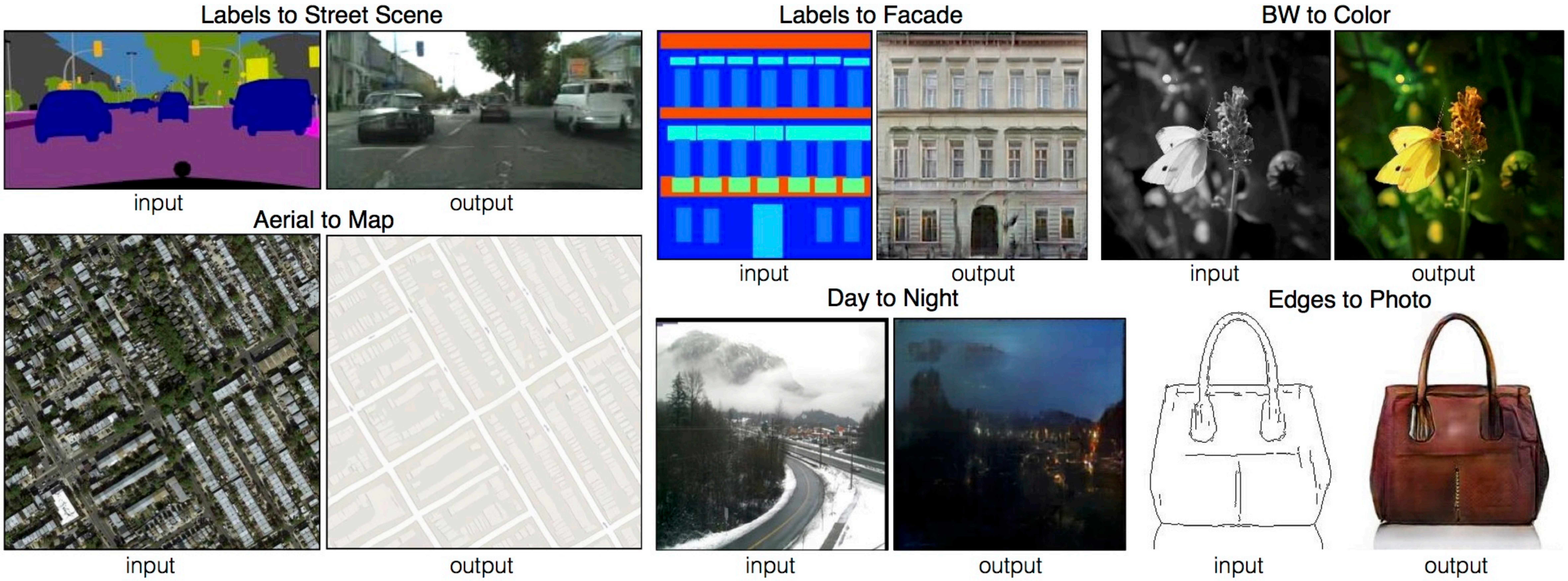
DCGAN by Radford et. al.



Examples → Trends → Tools for AI

Adversarial Nets

pix2pix by Isola, Zhu, Zhou, Efros @ UCBerkeley



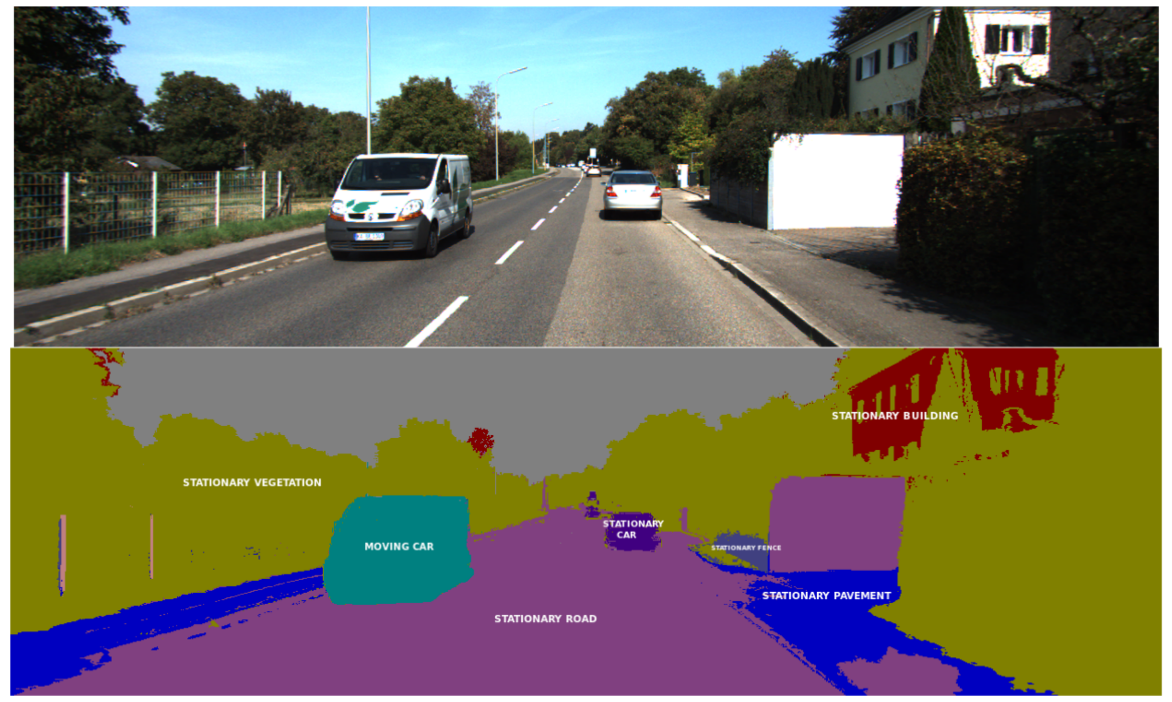
Examples → Trends → Tools for AI

Adversarial Nets

Cycle GAN by Zhu, Park, Isola, Efros @ UCBerkeley

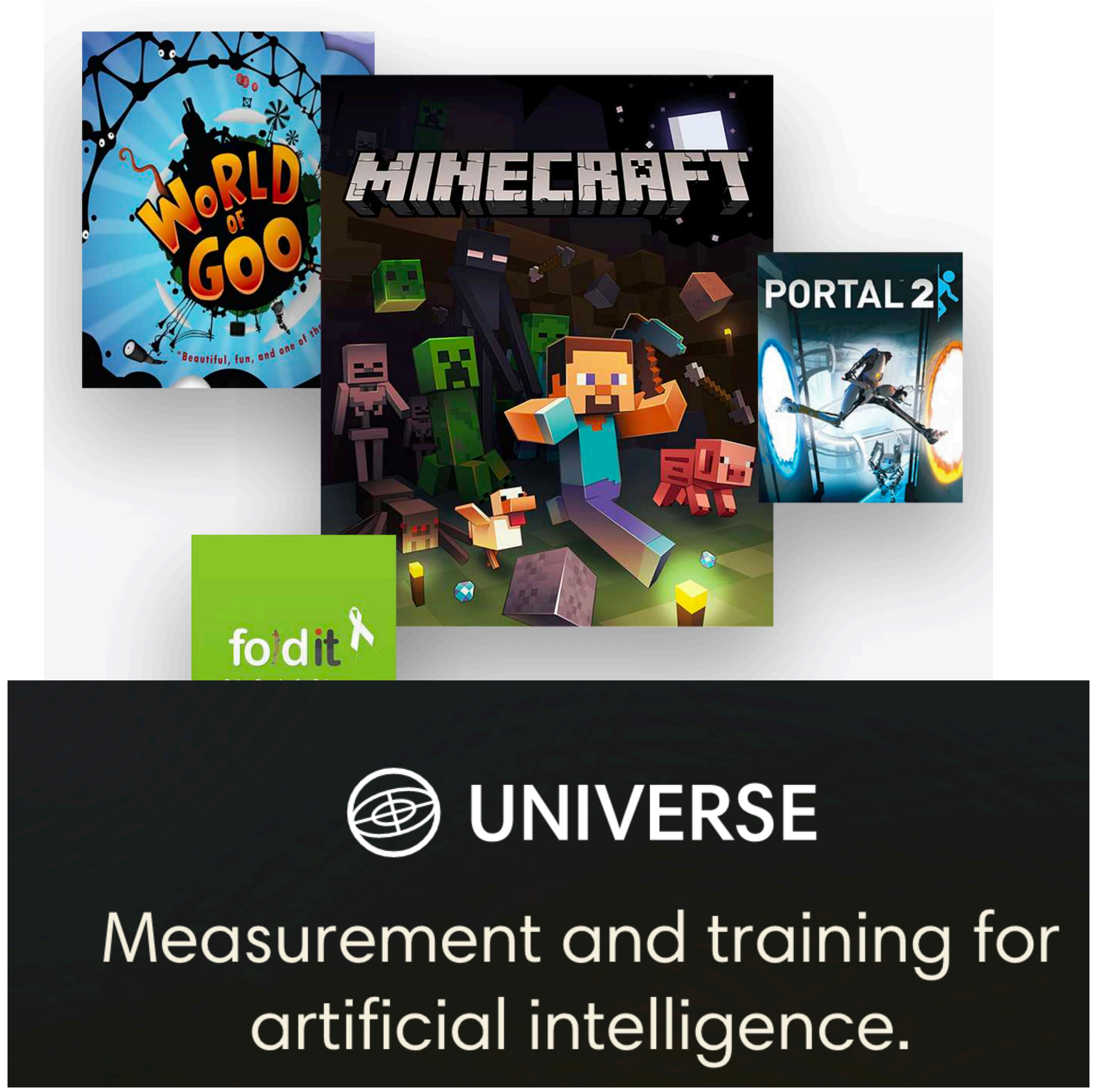
Examples → Trends → Tools for AI

Agents



Cars

Video games



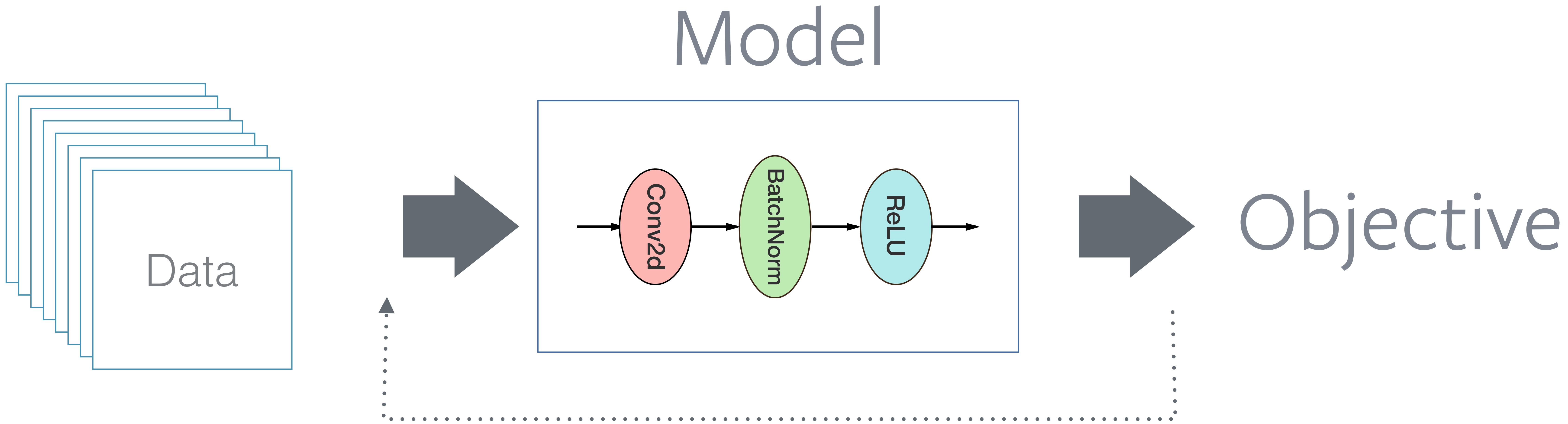
Internet

Examples → Trends → Tools for AI

Trends

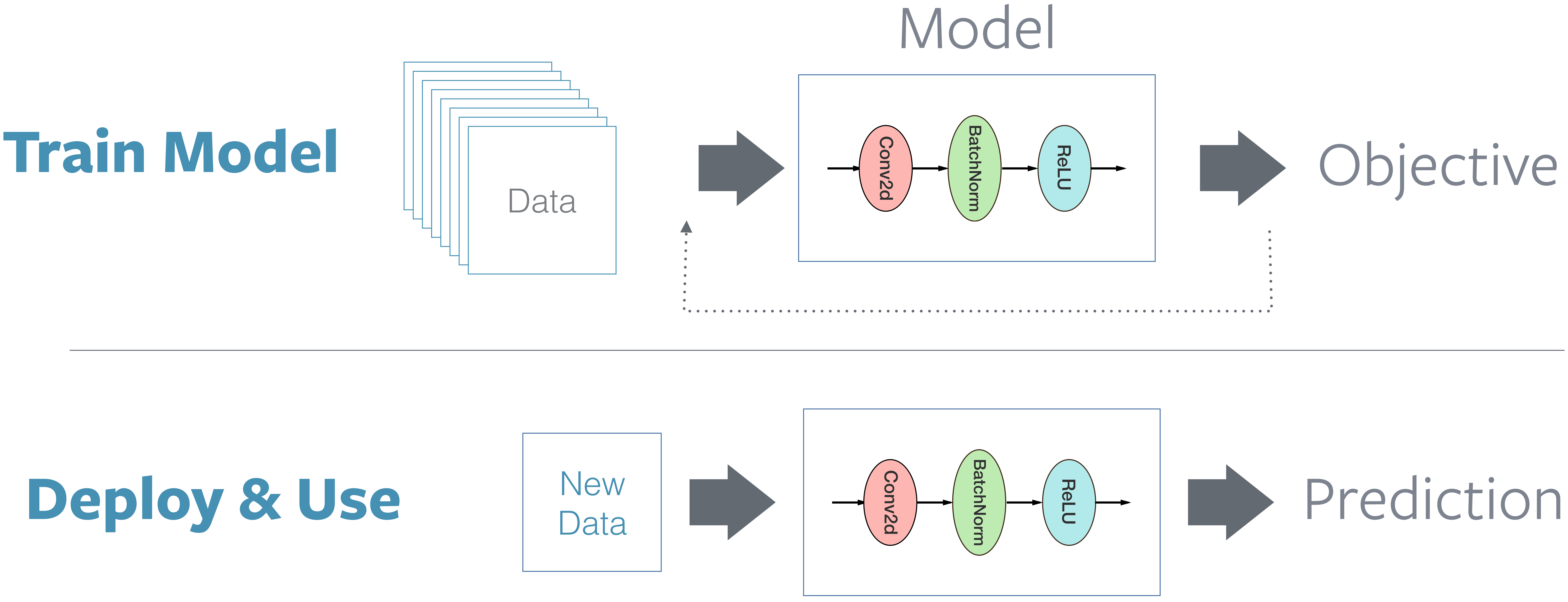
The static kind

Train Model



Examples → **Trends** → Tools for AI

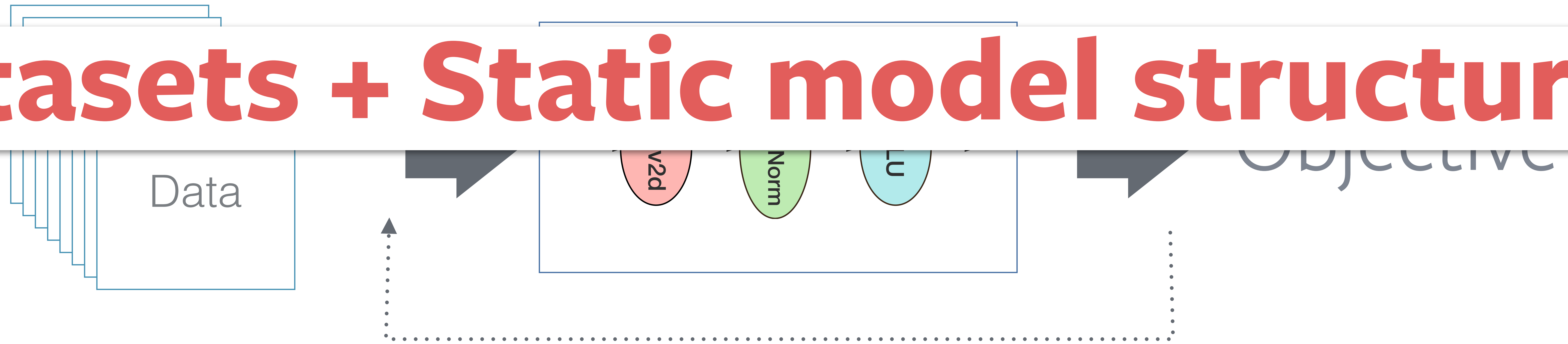
The static kind



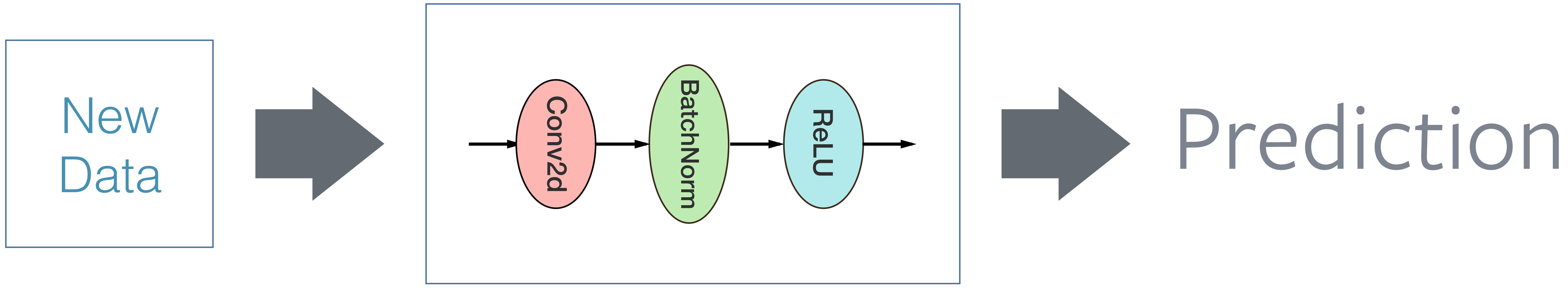
Examples → **Trends** → Tools for AI

The static kind

Static datasets + Static model structure



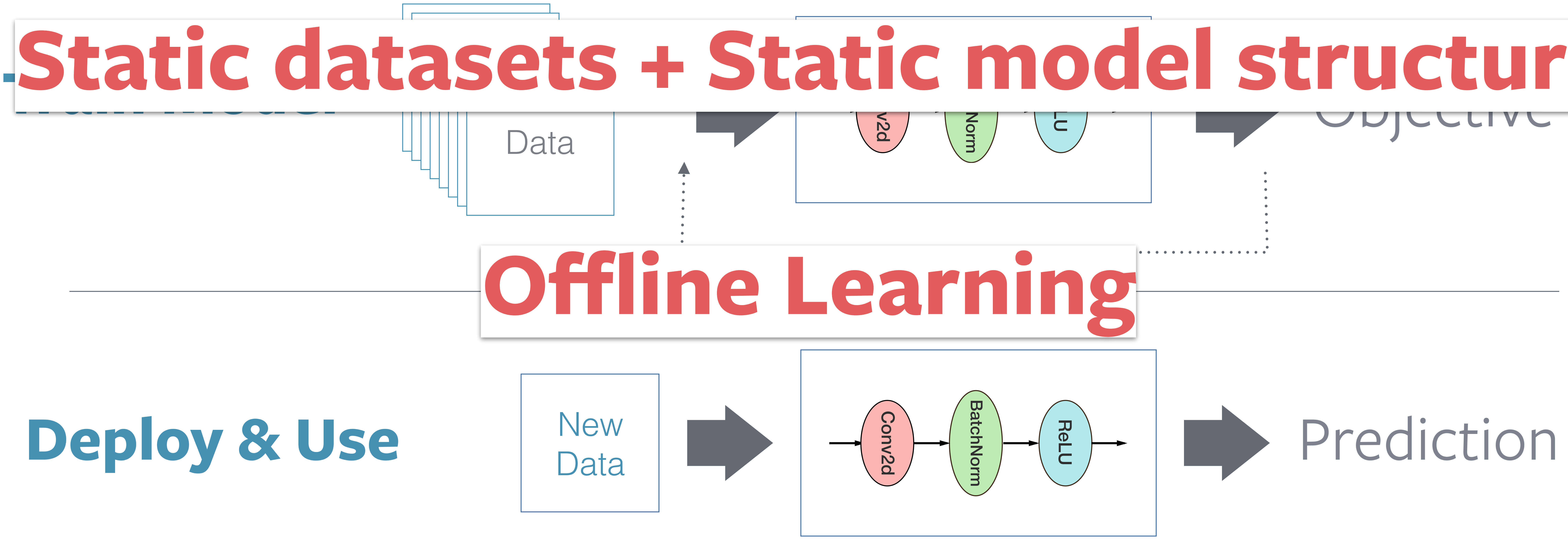
Deploy & Use



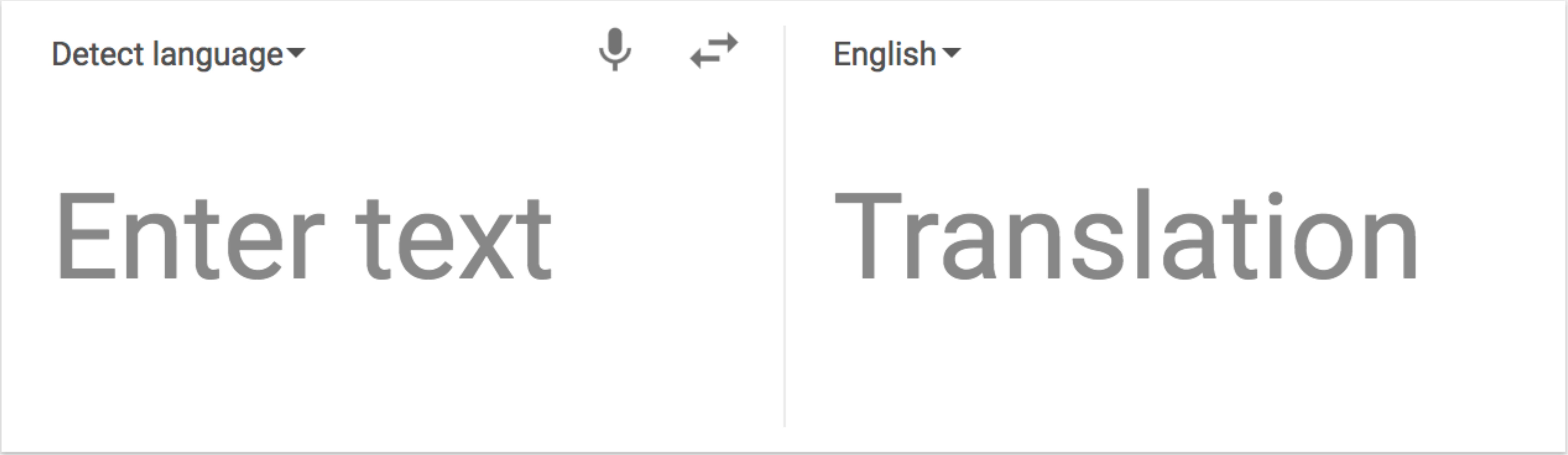
The static kind

Static datasets + Static model structure

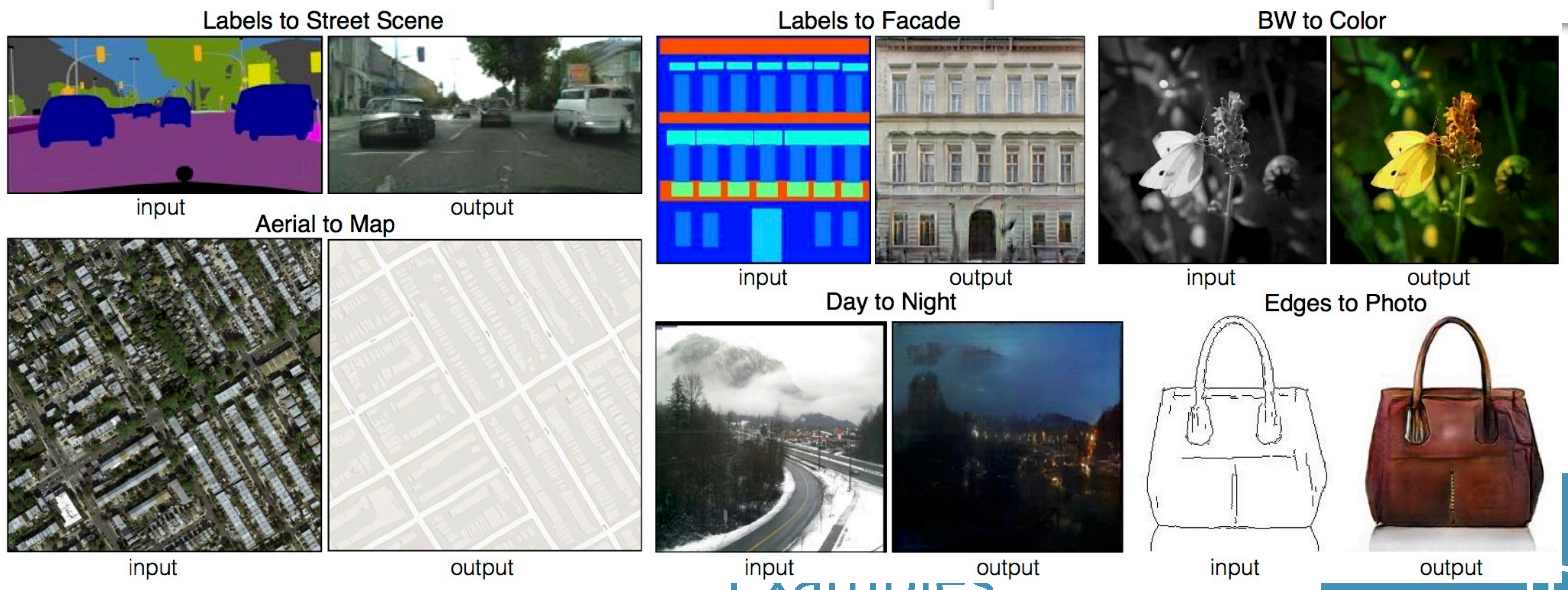
Offline Learning



The static kind

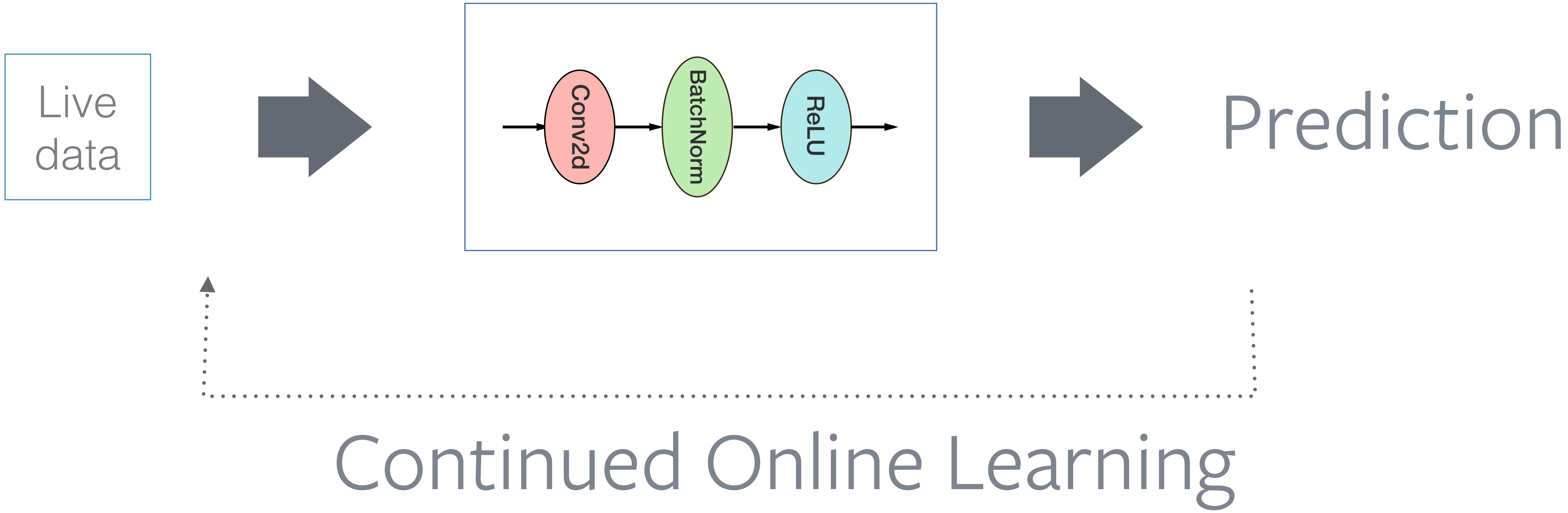


Offline Learning



Tools for AI

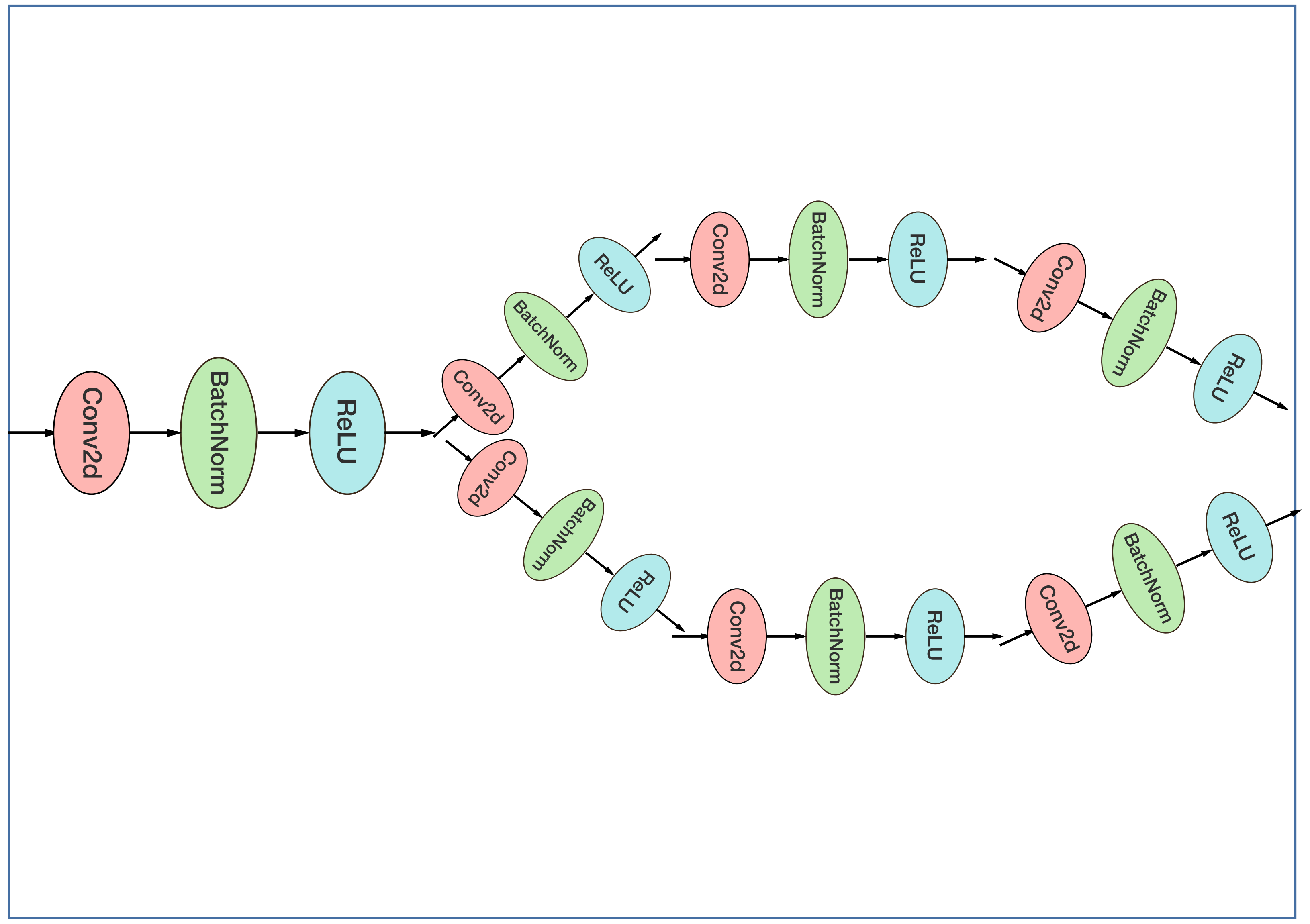
The dynamic kind



Examples → Trends → Tools for AI

The dynamic kind

Sample-**1**



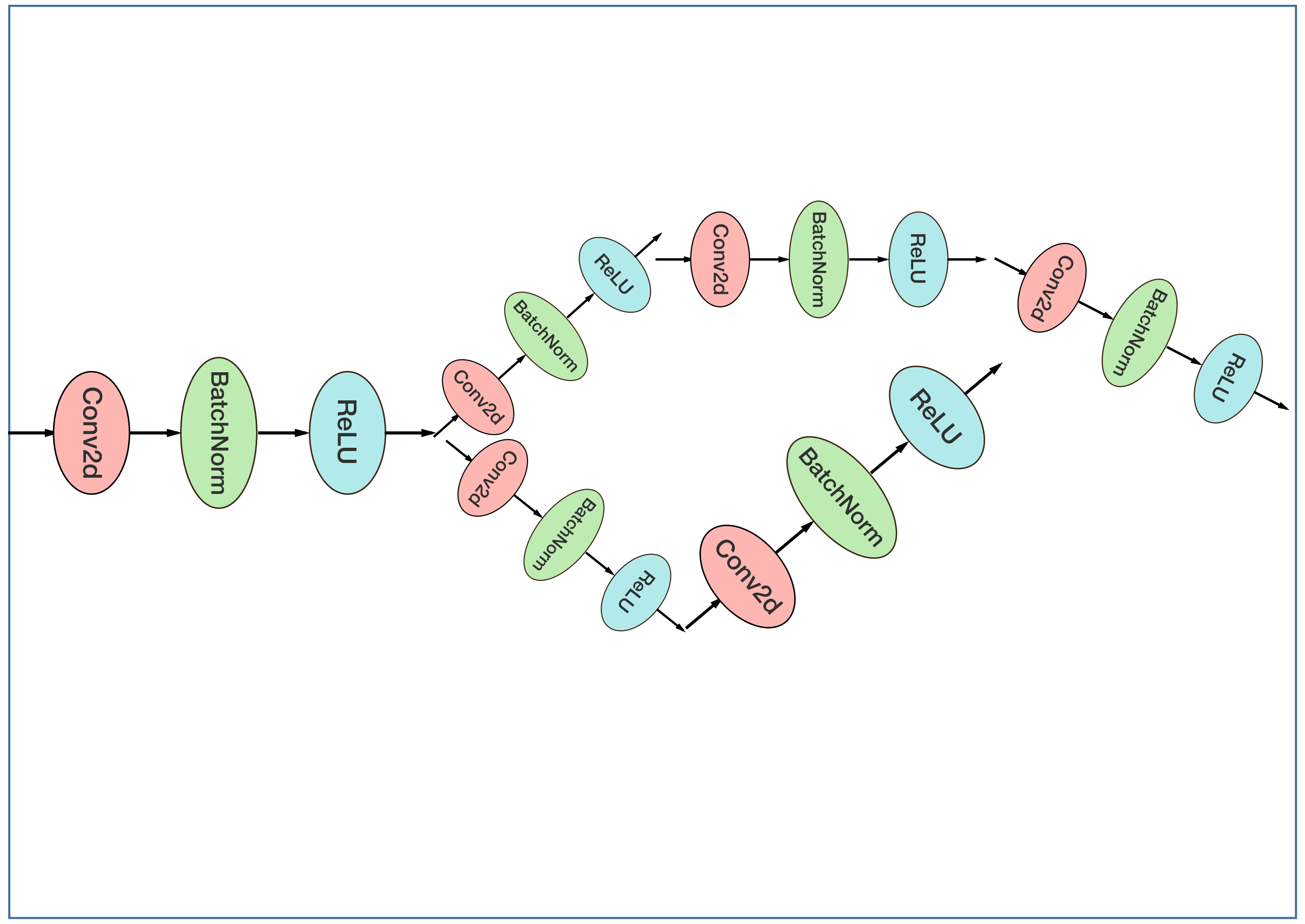
Prediction

Data-dependent change in model structure

Examples → **Trends** → Tools for AI

The dynamic kind

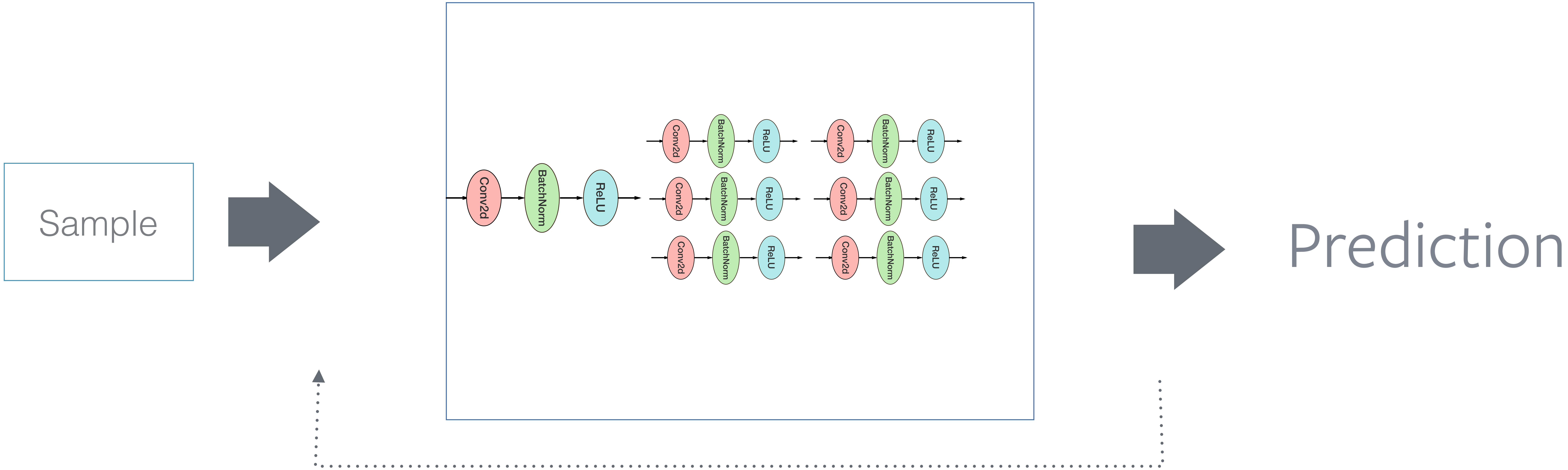
Sample-**2**



Prediction

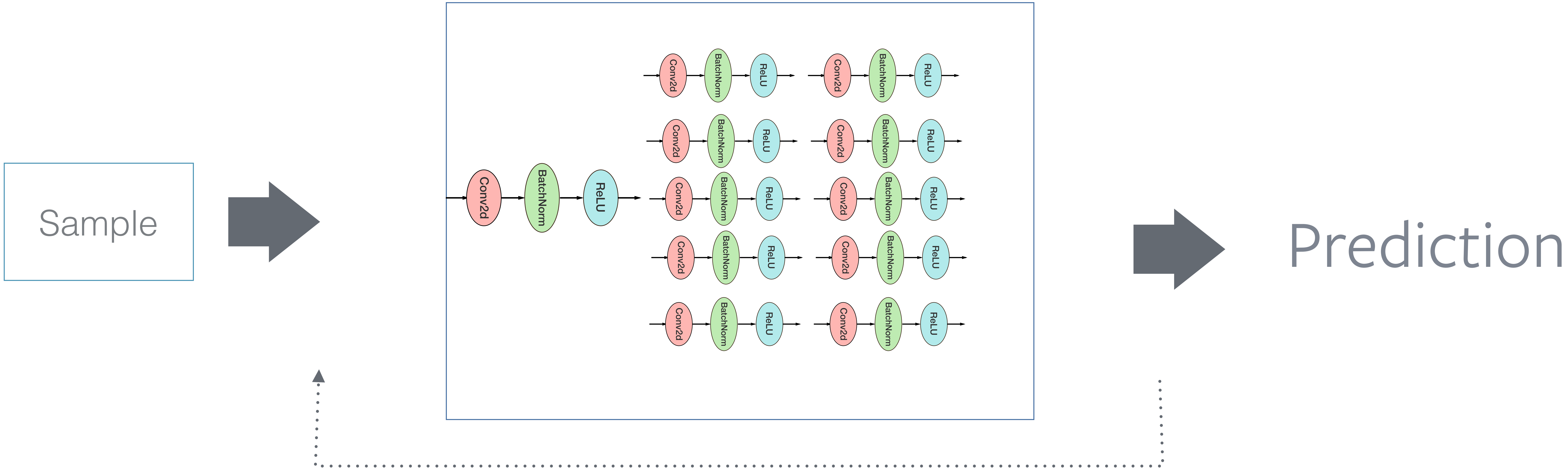
Data-dependent change in model structure

The dynamic kind



Change in model-capacity at runtime

The dynamic kind



Change in model-capacity at runtime

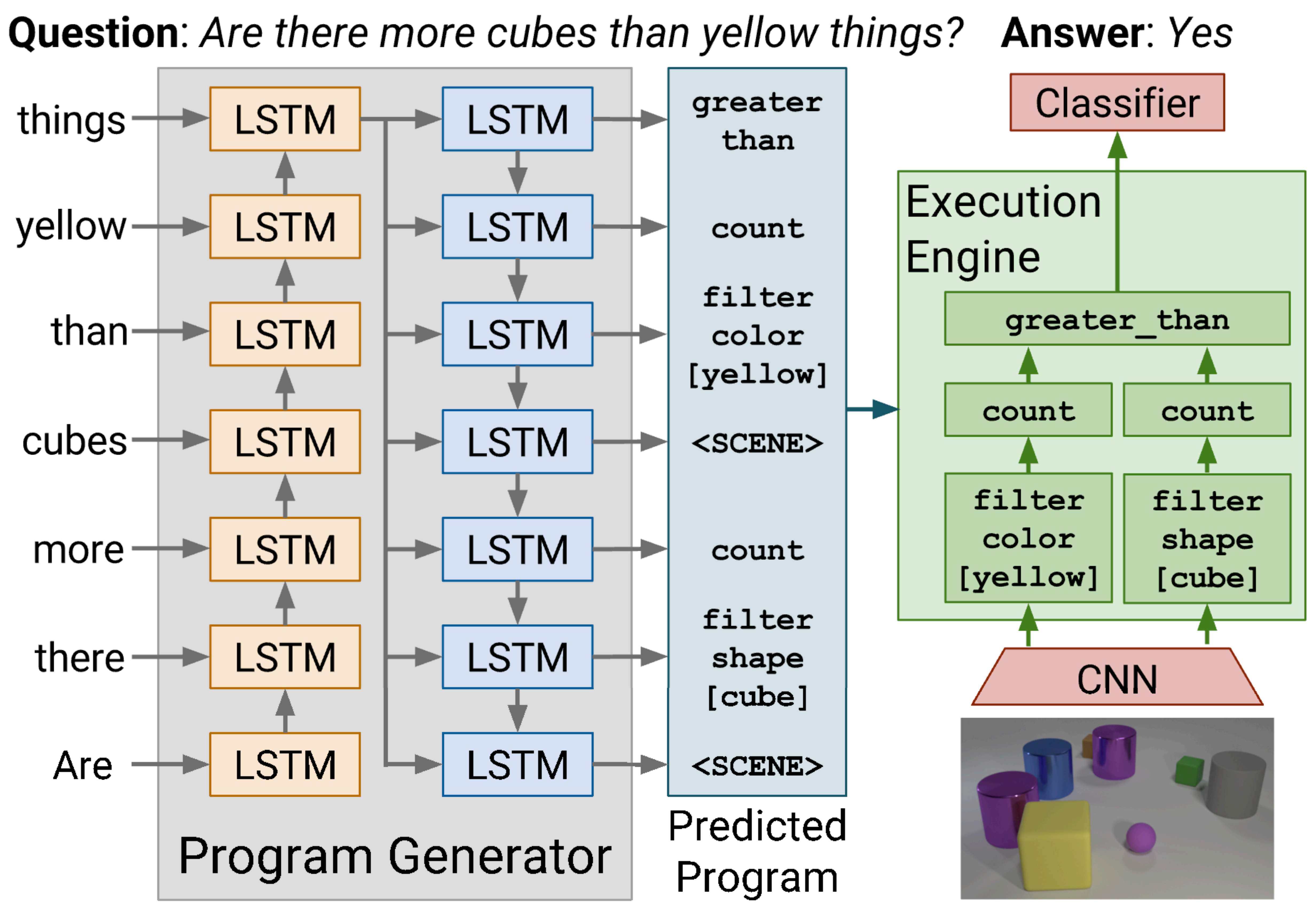
The dynamic kind

Self-driving Cars

Examples → **Trends** → Tools for AI

The dynamic kind

Inferring and Executing Programs for Visual Reasoning
- Johnson et. al. at Facebook



Examples → **Trends** → Tools for AI

The dynamic kind

Memory augmented

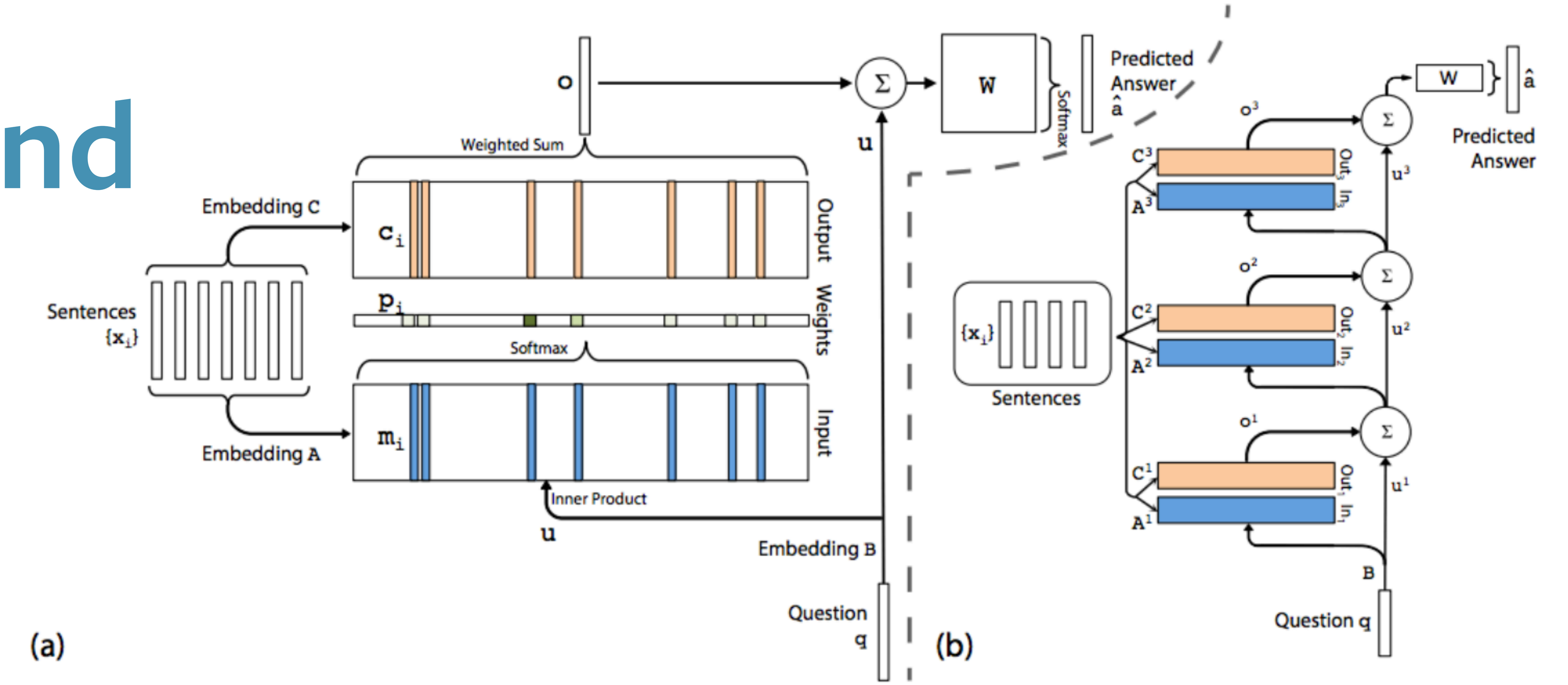
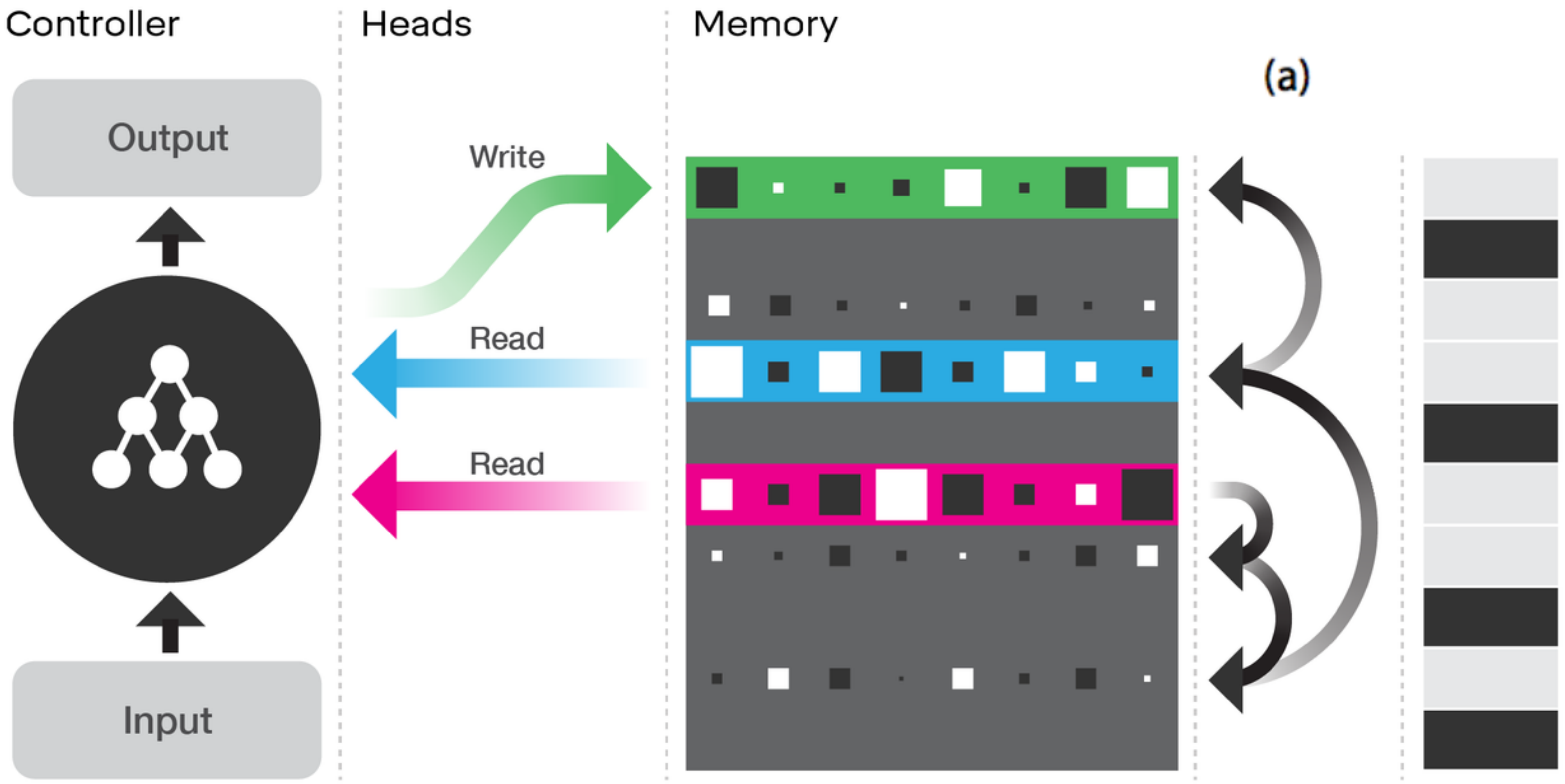


Illustration of the DNC architecture

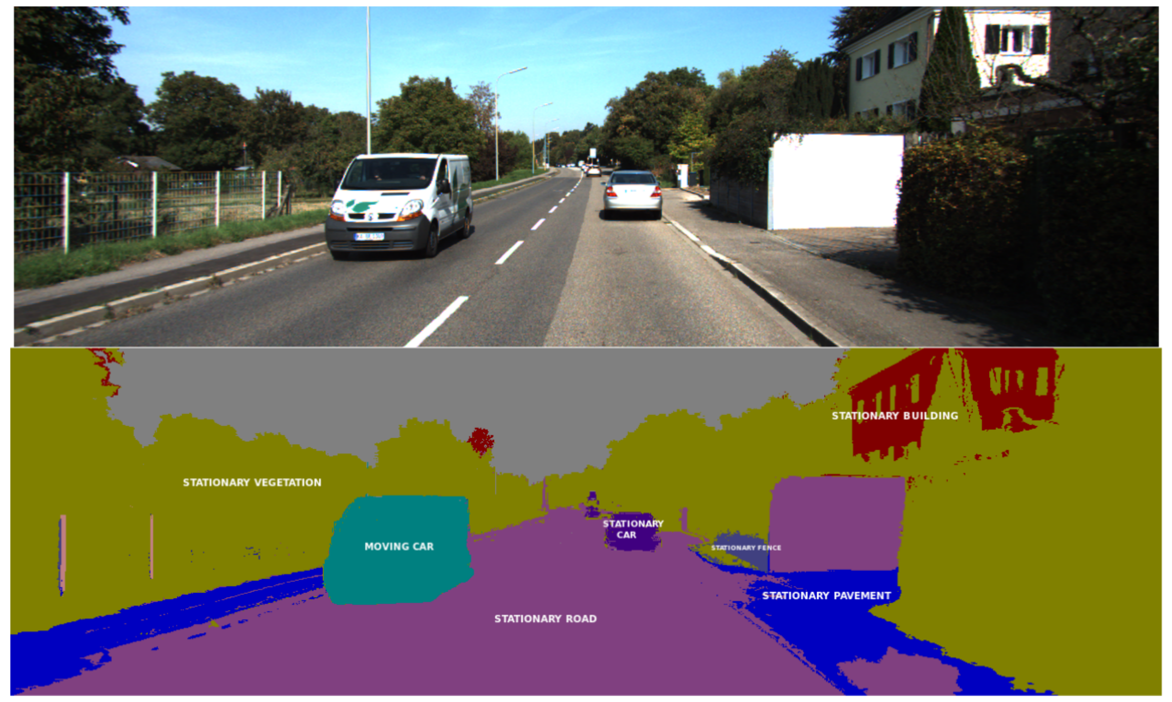


Memory Networks

- \- Facebook
- Differentiable Neural Computer
- \- Deepmind

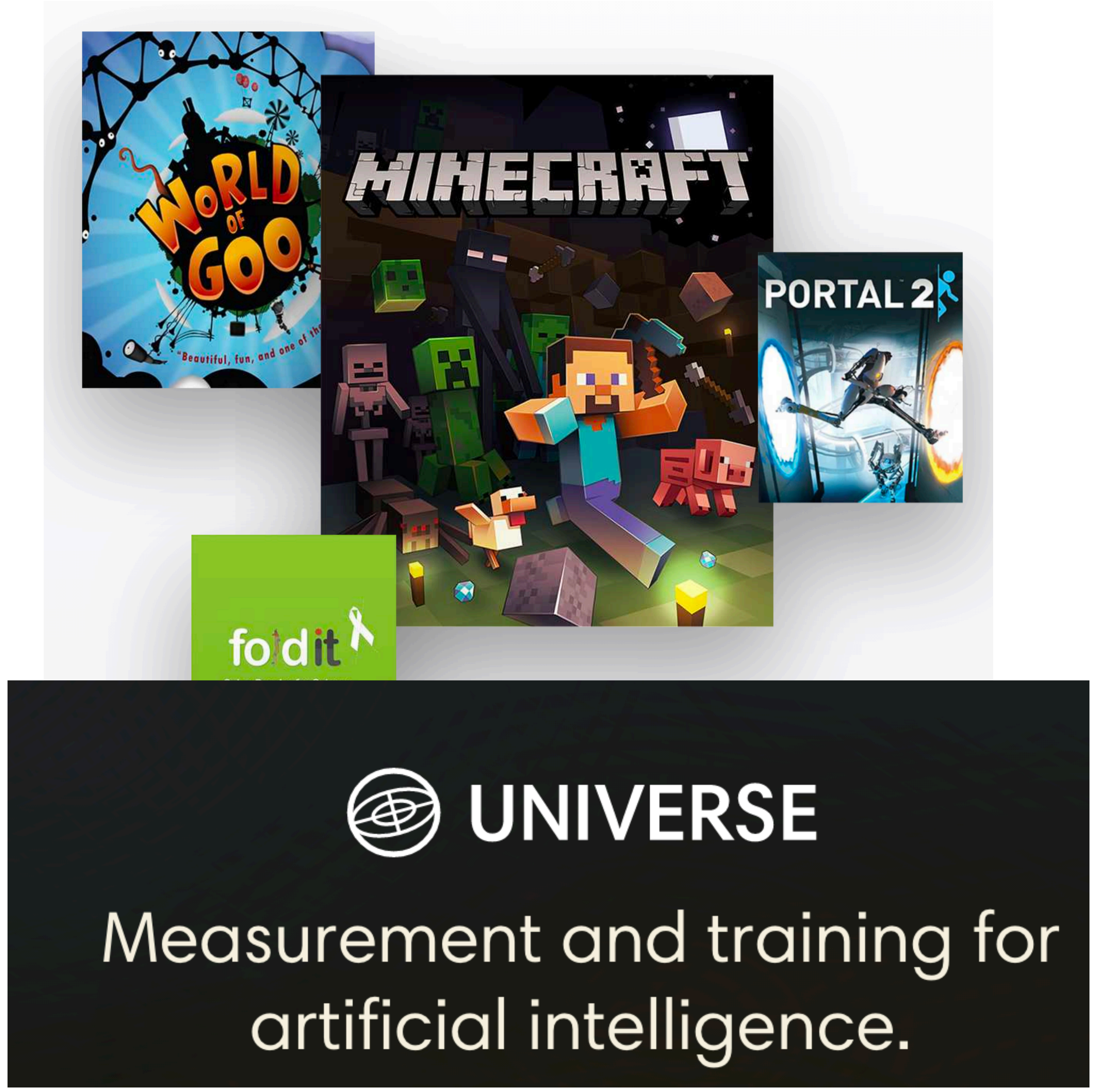
Examples → **Trends** → Tools for AI

The dynamic kind



Cars

Video games



Internet

Examples → **Trends** → Tools for AI

The dynamic kind

self-adding new memory or layers

changing evaluation path based on inputs

online learning

Tools

A next-gen framework for AI

- Interop with many dynamic environments
  - Connecting to car sensors should be as easy as training on a dataset
  - Connect to environments such as OpenAI Universe

A next-gen framework for AI

- Interop with many dynamic environments
  - Connecting to car sensors should be as easy as training on a dataset
  - Connect to environments such as OpenAI Universe
- Dynamic Neural Networks
  - Change behavior and structure of neural network at runtime

A next-gen framework for AI

- Interop with many dynamic environments
  - Connecting to car sensors should be as easy as training on a dataset
  - Connect to environments such as OpenAI Universe
- Dynamic Neural Networks
  - Change behavior and structure of neural network at runtime
- Minimal Abstractions
  - more complex AI systems means harder to debug without a simple API

A next-gen framework for AI

- Interop with many dynamic environments
  - Connecting to car sensors should be as easy as training on a dataset
  - Connect to environments such as OpenAI Universe
- Dynamic Neural Networks
  - Change behavior and structure of neural network at runtime
- Minimal Abstractions
  - more complex AI systems means harder to debug without a simple API
- FAST

A next-gen framework for AI

- Interop with many dynamic environments
  - Connecting to car sensors should be as easy as training on a dataset
  - Connect to environments such as OpenAI Universe
- Dynamic Neural Networks
  - Change behavior and structure of neural network at runtime
- Minimal Abstractions
  - more complex AI systems means harder to debug without a simple API
- FAST

Tools for AI research and deployment

Many machine learning tools and deep learning frameworks

PYTORCH

Chainer

mxnet

TensorFlow

theano

Caffe

K

Microsoft CNTK

Examples → Trends → **Tools for AI**

Tools for AI research and deployment

Static graph frameworks

Dynamic graph frameworks (more naturally enable dynamic deep learning)

Examples → Trends → Tools for AI

Static graph Frameworks

- Model is constructed and compiled once and reused many times
- Hard to change the model on the fly
- harder to debug in a complex system

Dynamic graph Frameworks

- Model is constructed on the fly at runtime
- Change behavior, structure of model
- Imperative style of programming

Examples → Trends → Tools for AI

PyTorch

automatic differentiation engine

Ndarray library with GPU support

gradient based optimization package

Deep Learning

Reinforcement Learning

Numpy-alternative

```python
# -*- coding: utf-8 -*-
import numpy as np

# N is batch size; D_in is input dimension;
# H is hidden dimension; D_out is output dimension.
N, D_in, H, D_out = 64, 1000, 100, 10

# Create random input and output data
x = np.random.randn(N, D_in)
y = np.random.randn(N, D_out)

# Randomly initialize weights
w1 = np.random.randn(D_in, H)
w2 = np.random.randn(H, D_out)

learning_rate = 1e-6
for t in range(500):
    # Forward pass: compute predicted y
    h = x.dot(w1)
    h_relu = np.maximum(h, 0)
    y_pred = h_relu.dot(w2)

    # Compute and print loss
    loss = np.square(y_pred - y).sum()
    print(t, loss)

    # Backprop to compute gradients of w1 and w2 with respect to loss
    grad_y_pred = 2.0 * (y_pred - y)
    grad_w2 = h_relu.T.dot(grad_y_pred)
    grad_h_relu = grad_y_pred.dot(w2.T)
    grad_h = grad_h_relu.copy()
    grad_h[h < 0] = 0
    grad_w1 = x.T.dot(grad_h)

    # Update weights
    w1 -= learning_rate * grad_w1
    w2 -= learning_rate * grad_w2
```

Numpy

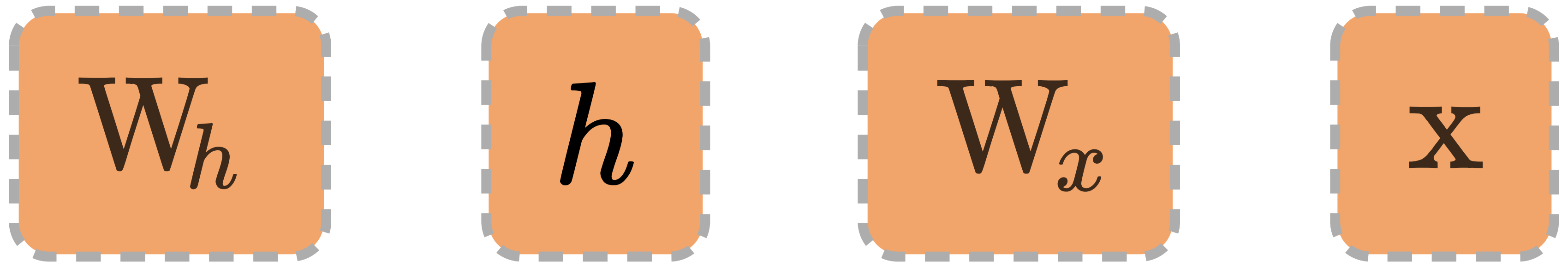
```python
import torch


dtype = torch.FloatTensor
# dtype = torch.cuda.FloatTensor # Uncomment this to run on GPU

# N is batch size; D_in is input dimension;
# H is hidden dimension; D_out is output dimension.
N, D_in, H, D_out = 64, 1000, 100, 10

# Create random input and output data
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)

# Randomly initialize weights
w1 = torch.randn(D_in, H).type(dtype)
w2 = torch.randn(H, D_out).type(dtype)

learning_rate = 1e-6
for t in range(500):
    # Forward pass: compute predicted y
    h = x.mm(w1)
    h_relu = h.clamp(min=0)
    y_pred = h_relu.mm(w2)

    # Compute and print loss
    loss = (y_pred - y).pow(2).sum()
    print(t, loss)

    # Backprop to compute gradients of w1 and w2 with respect to loss
    grad_y_pred = 2.0 * (y_pred - y)
    grad_w2 = h_relu.t().mm(grad_y_pred)
    grad_h_relu = grad_y_pred.mm(w2.t())
    grad_h = grad_h_relu.clone()
    grad_h[h < 0] = 0
    grad_w1 = x.t().mm(grad_h)

    # Update weights using gradient descent
    w1 -= learning_rate * grad_w1
    w2 -= learning_rate * grad_w2
```

PyTorch

PyTorch Autograd

```
from torch.autograd import Variable
```

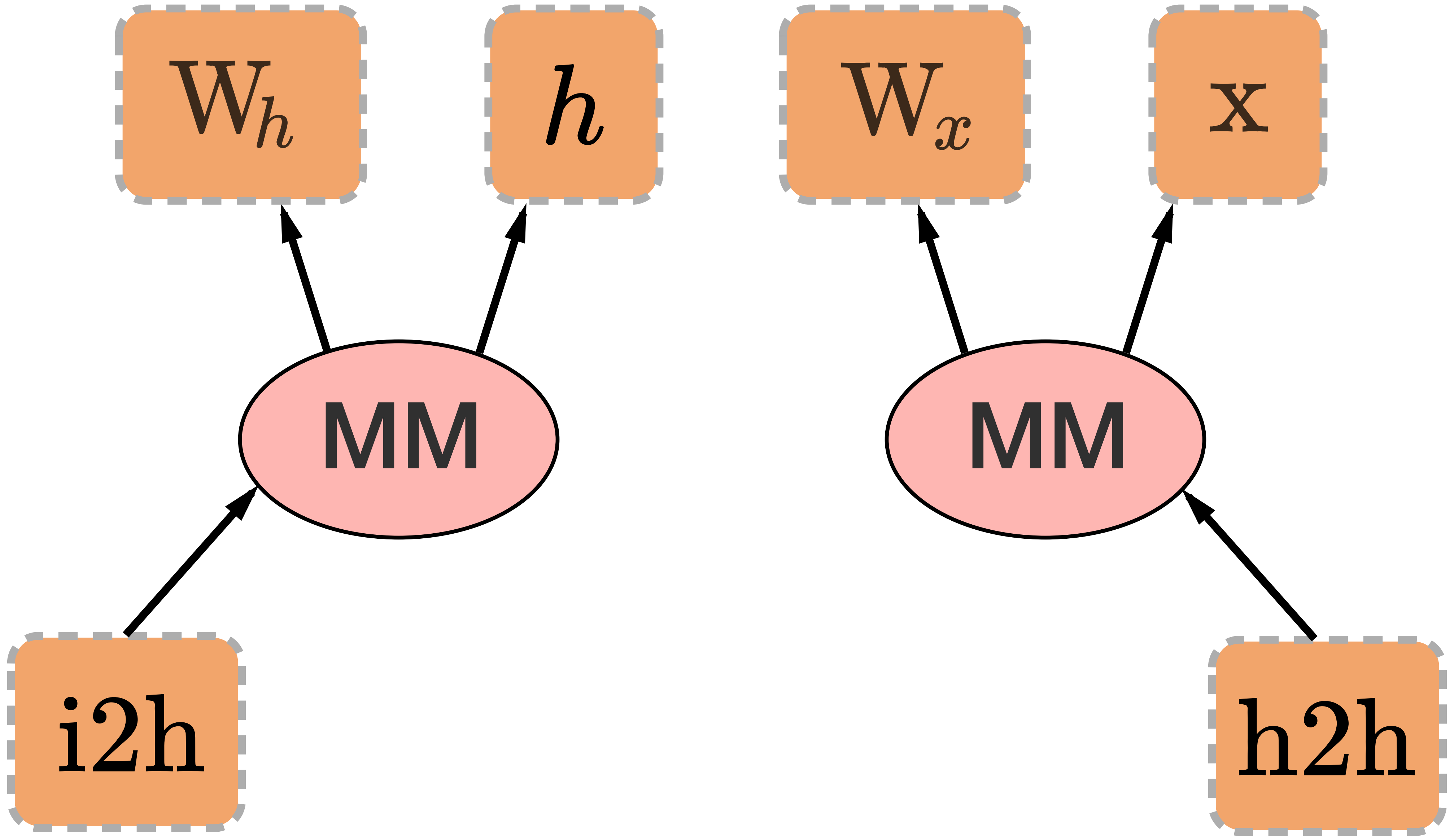
PyTorch Autograd




```python
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```

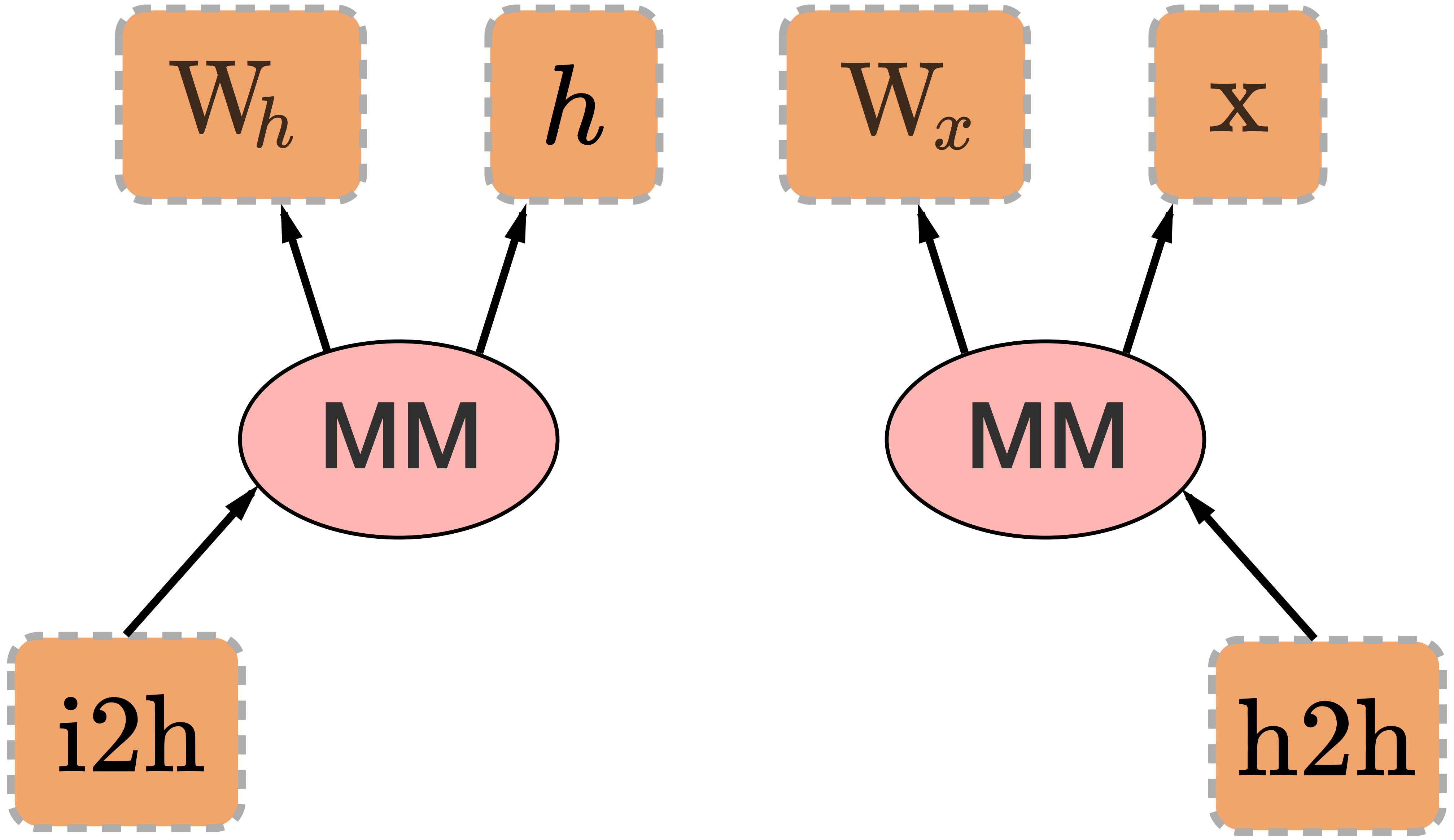
PyTorch Autograd

```python
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
```



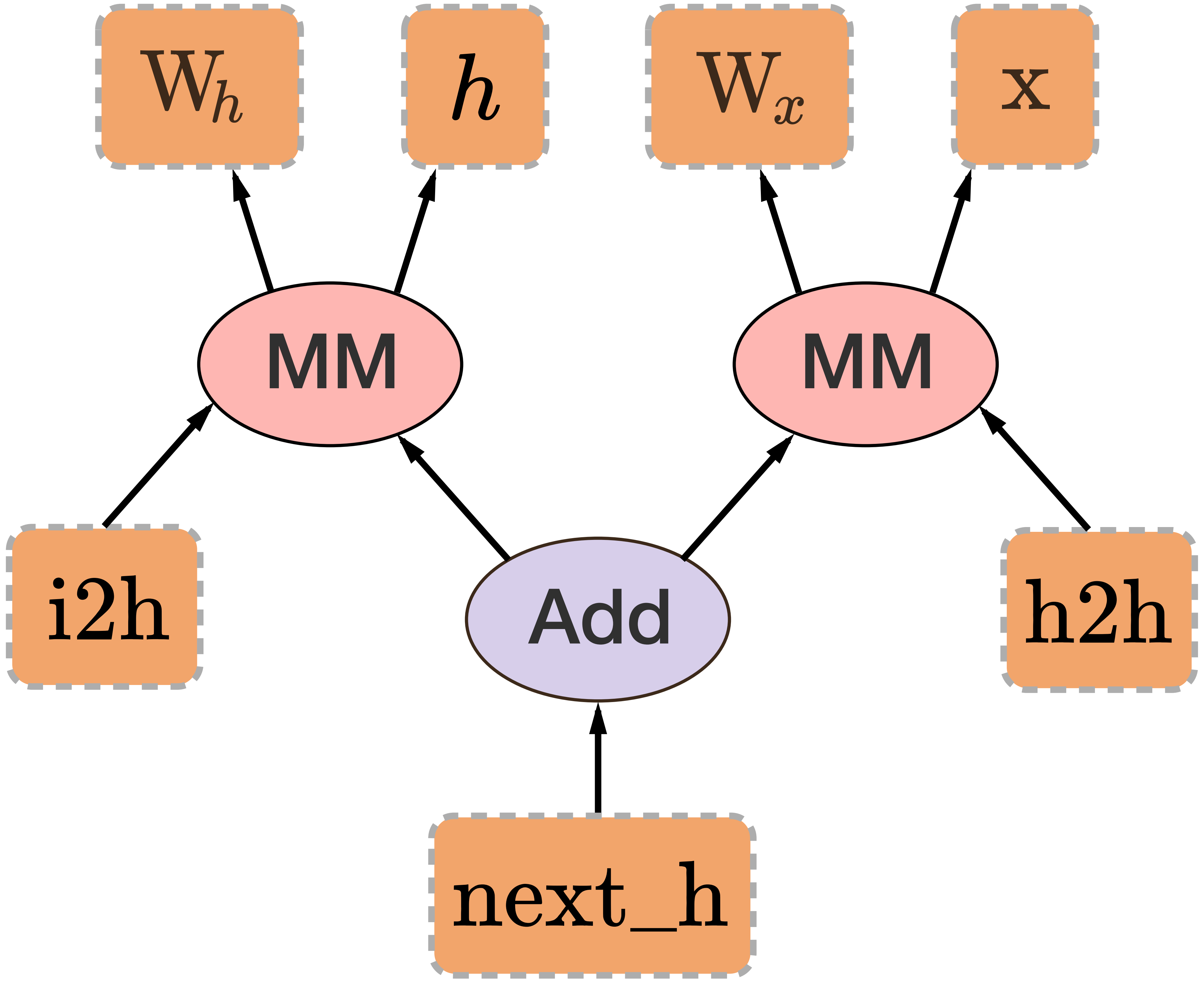
PyTorch Autograd

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
next_h = i2h + h2h
```



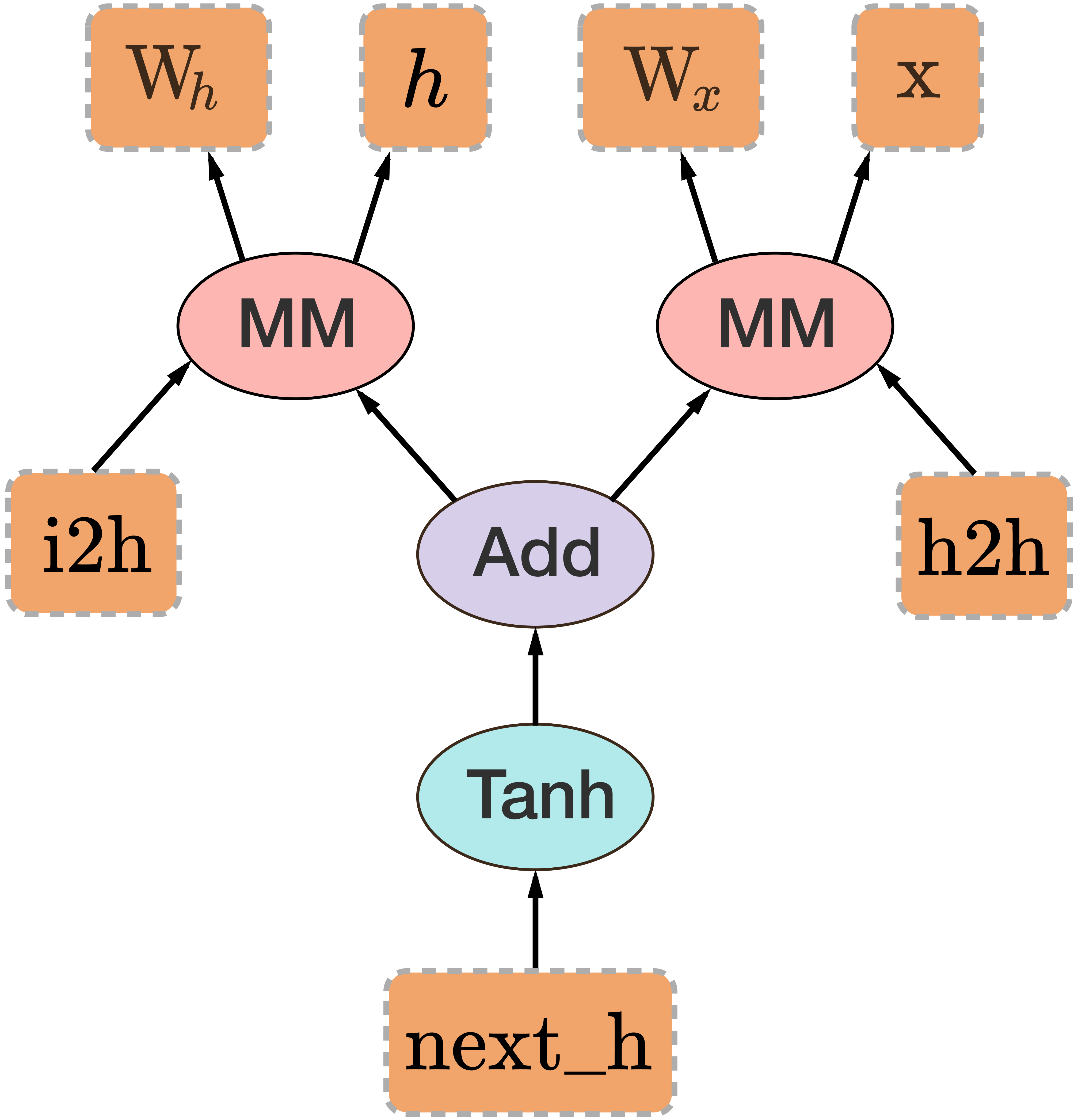
PyTorch Autograd

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
next_h = i2h + h2h
```



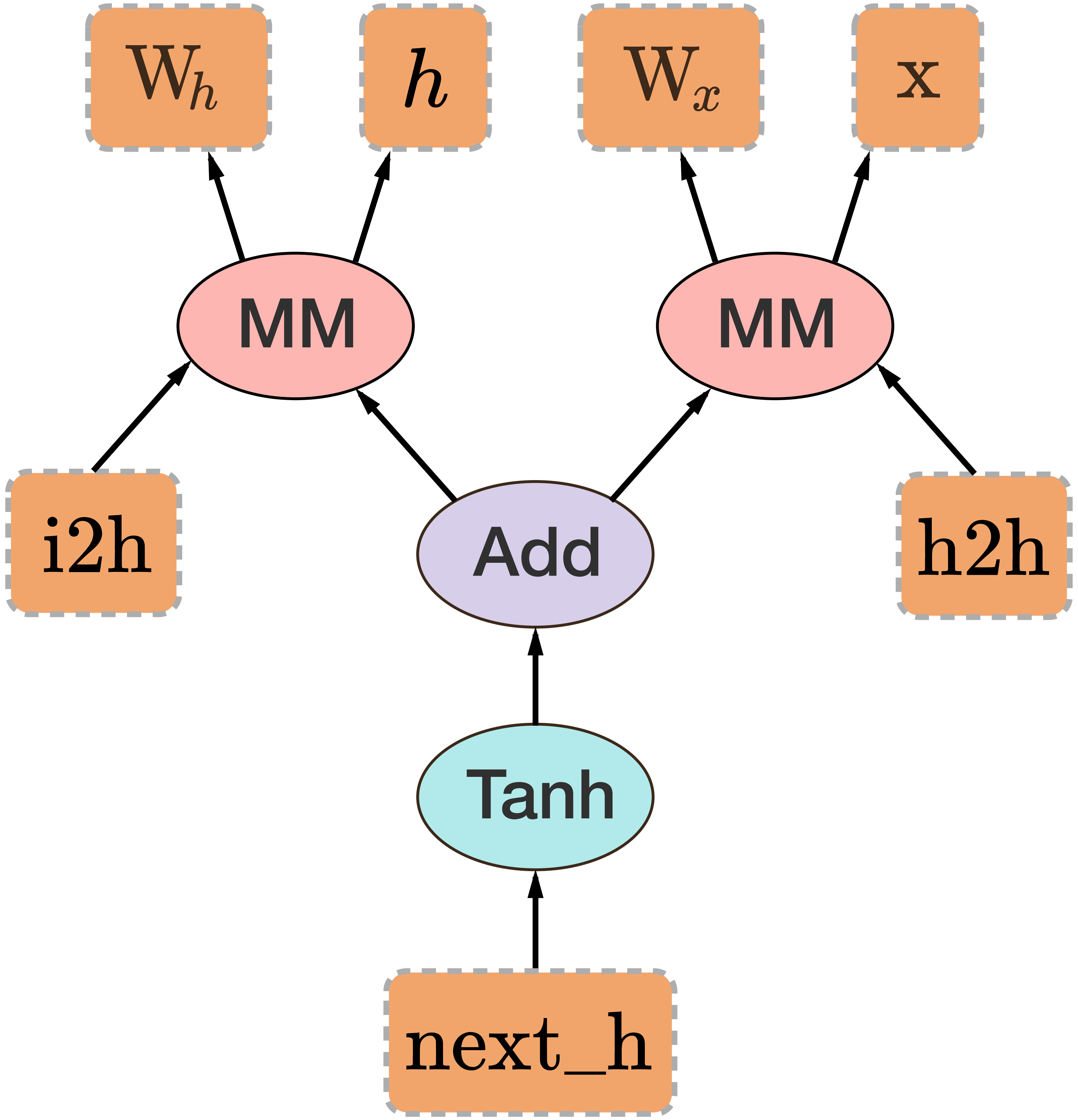
PyTorch Autograd

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
next_h = i2h + h2h
next_h = next_h.tanh()
```



PyTorch Autograd

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
next_h = i2h + h2h
next_h = next_h.tanh()

next_h.backward(torch.ones(1, 20))
```



PyTorch

- Naturally enables dynamic deep learning
- easy to interface with a wide range of interactive environments
  - because of an imperative style of programming
  - because of deep Python integration
- as fast as anything else out there on average

PYTORCH

http://pytorch.org

With ❤ from

facebook

NVIDIA.

salesforce

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIE
PARIS INSTITUTE OF TECHNOLOGY

Carnegie Mellon University

UNIVERSITE PIERRE & MARIE CURIE

Digital Reasoning

Stanford University

UNIVERSITY OF OXFORD

NYU

inria

ENS
ÉCOLE NORMALE SUPÉRIEURE

EPFL
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Berkeley
UNIVERSITY OF CALIFORNIA