MxNet Gluon

Basics, Computer Vision, NLP (and even more NLP)

Part I (Basics)

Leonard Lausen
Haibin Lin
Alex Smola

aws

Outline

8:30-9:15	Installation and Basics (NDArray, AutoGrad, Libraries)
9:15-9:30	Neural Networks 101 (MLP, ConvNet, LSTM, Loss, SGD) - Part I
9:30-10:00	Break
10:00-10:30	Neural Networks 101 (MLP, ConvNet, LSTM, Loss, SGD) - Part II
10:30-11:00	Computer Vision 101 (Gluon CV)
11:00-11:30	Parallel and distributed training
11:30-12:00	Data I/O in NLP (and iterators)
12:00-13:30	Break
13:30-14:15	Embeddings
14:15-15:00	Language models (LM)
15:00-15:30	Sequence Generation from LM
15:30-16:00	Break
16:00-16:15	Sentiment analysis
16:15-17:00	Transformer Models & machine translation
17:00-17:30	Questions

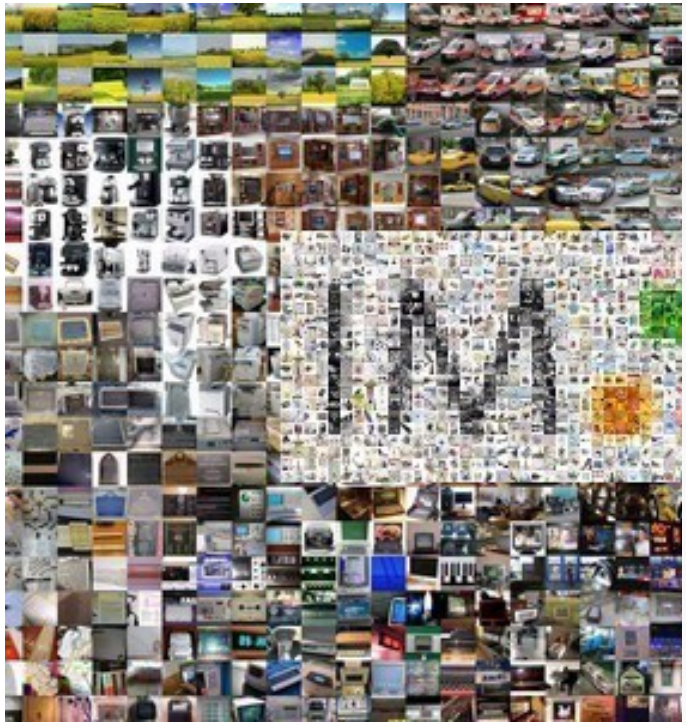
What can deep learning do?

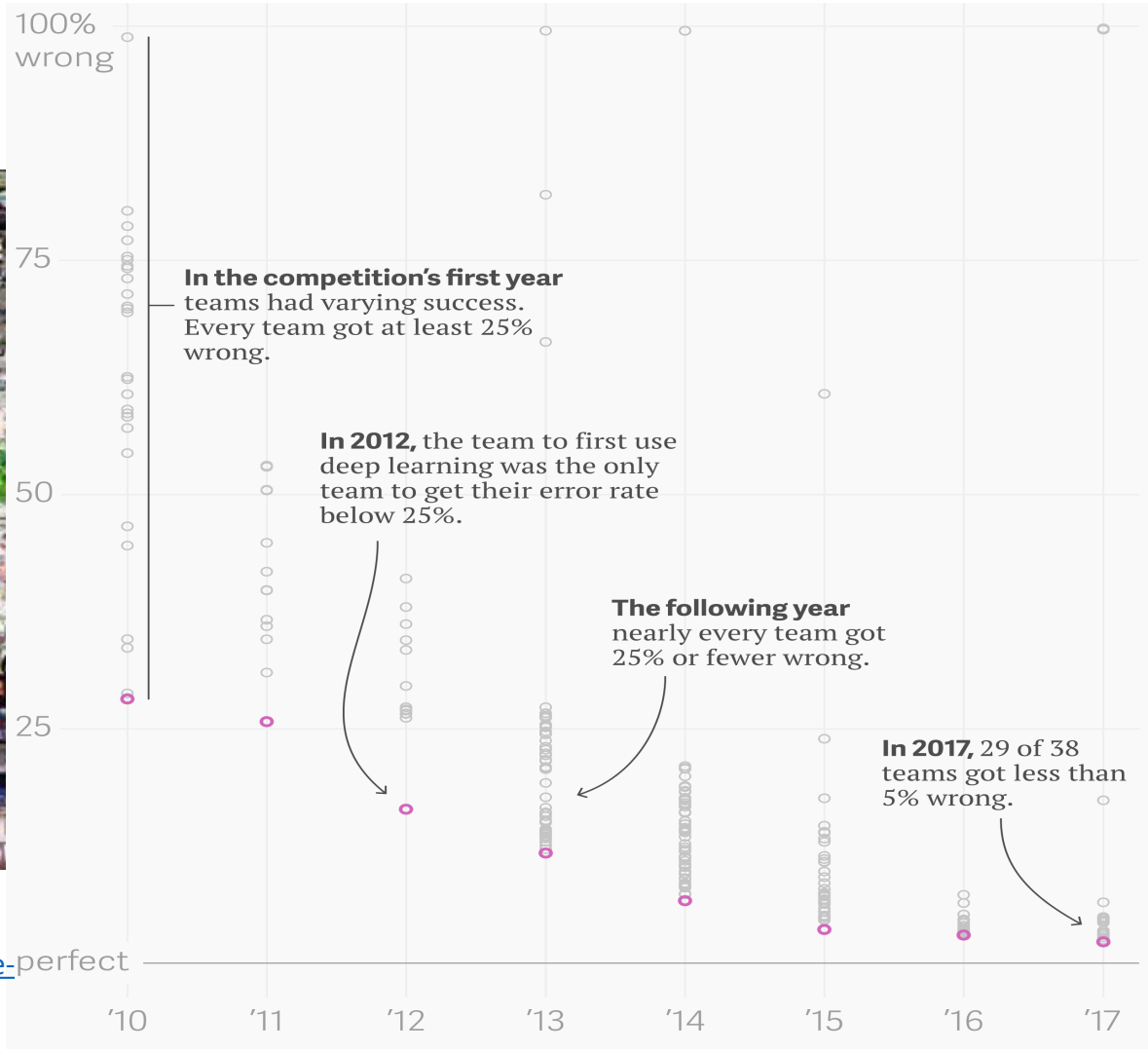
Classify Images

http://www.image-net.org/

aws

Classify Images





Yanofsky, Quartz
https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/

Detect and Segment Objects

https://github.com/matterport/Mask_RCNN

aws

Style transfer

https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

Style transfer

https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

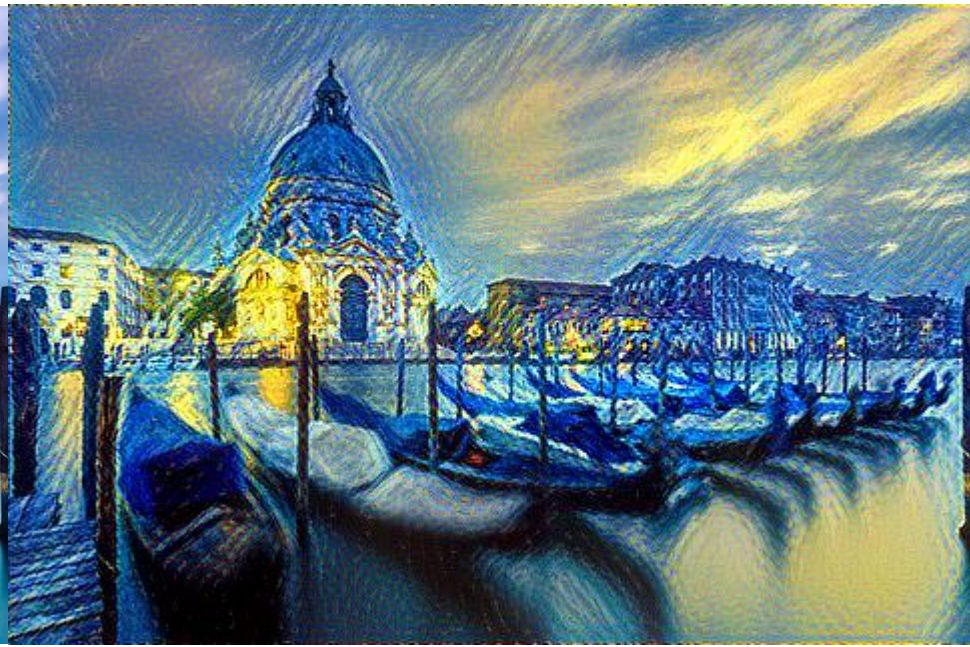
Style transfer

https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

Style transfer

https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

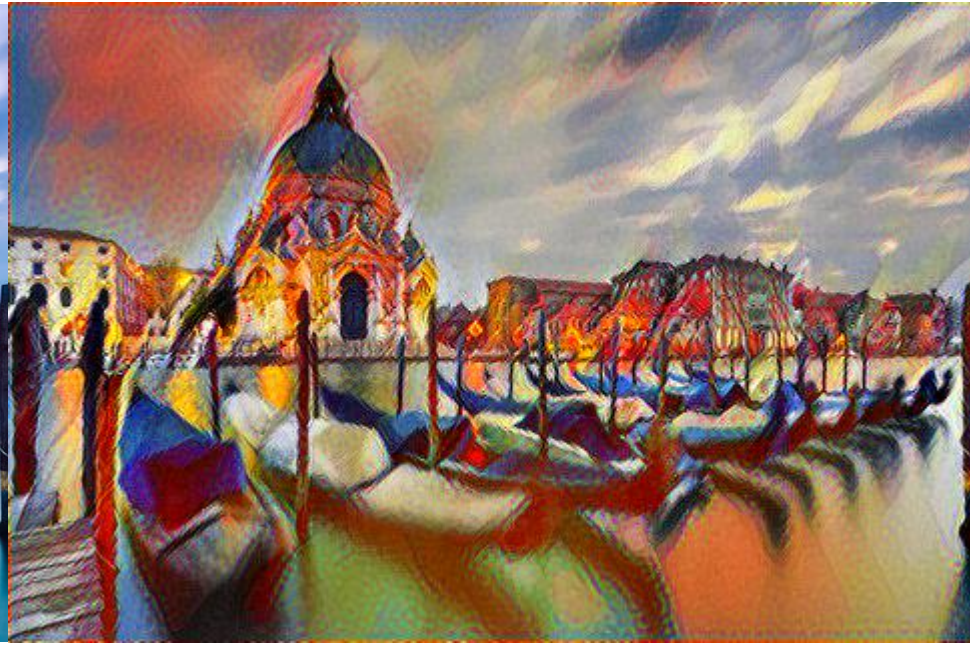
Style transfer



https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

aws

Style transfer



https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

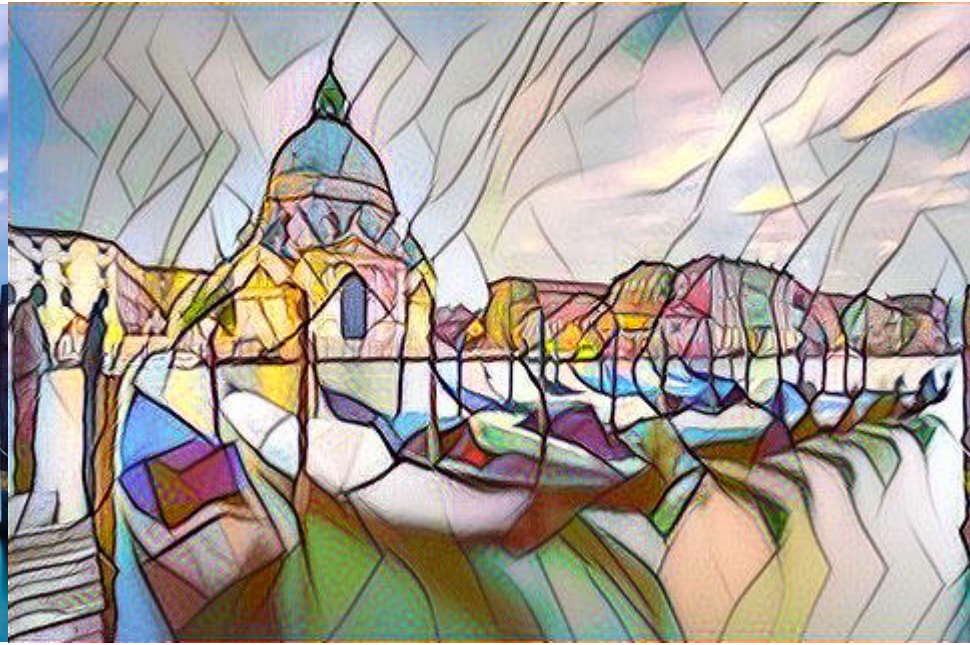
Style transfer

https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

Style transfer

https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

Style transfer



https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

Style transfer

https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

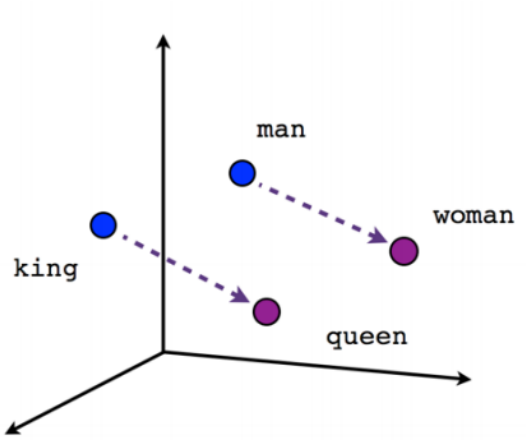
Style transfer

https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/

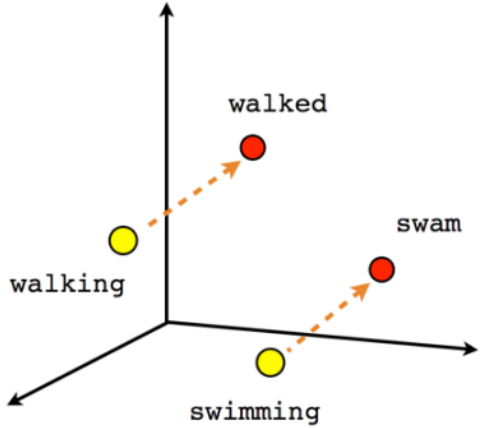
Synthesize Faces

Karras et al, ICLR 2018

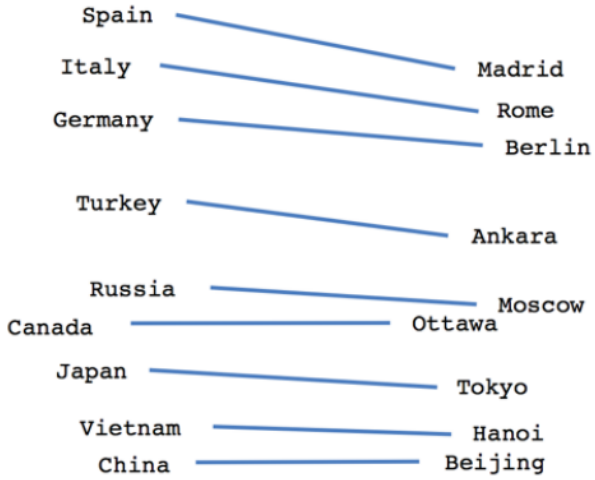
Analogies



Male-Female



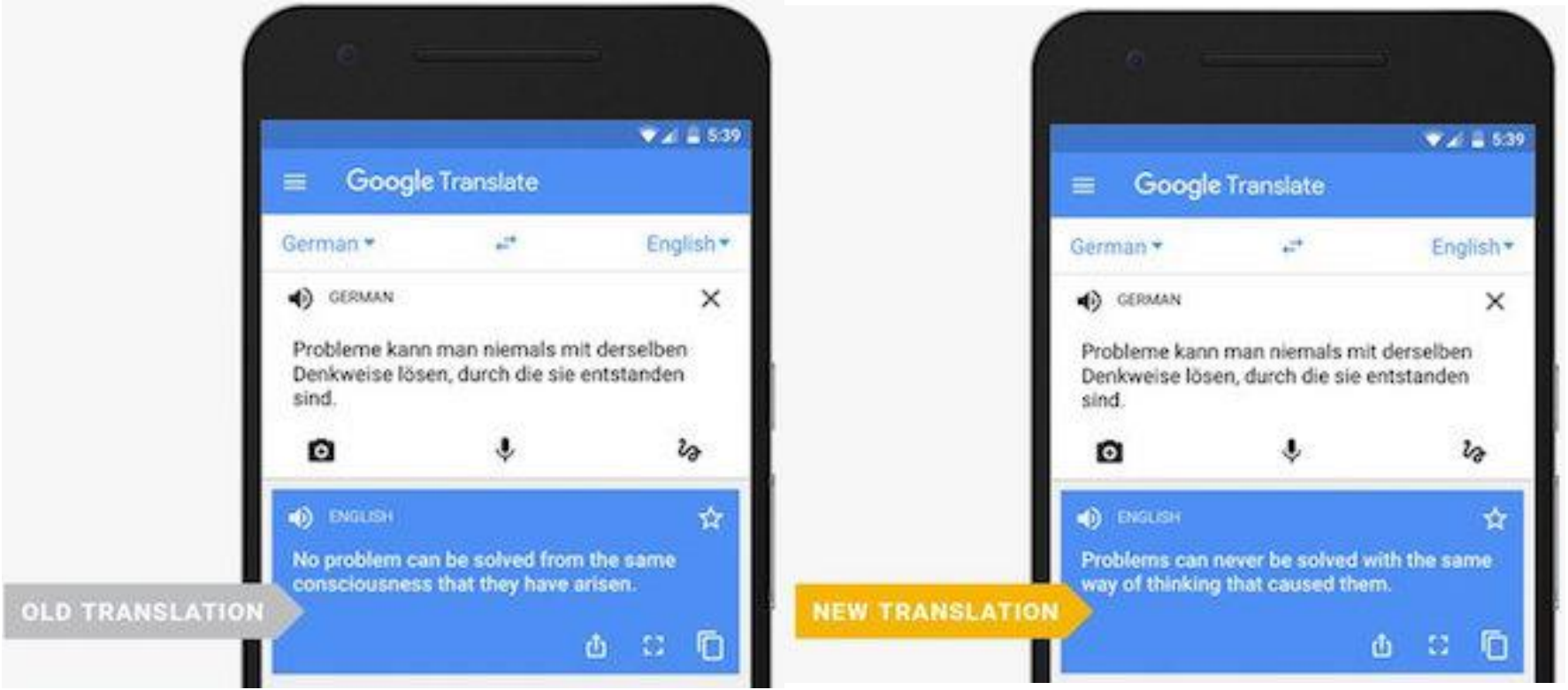
Verb tense



Country-Capital

https://www.tensorflow.org/tutorials/word2vec

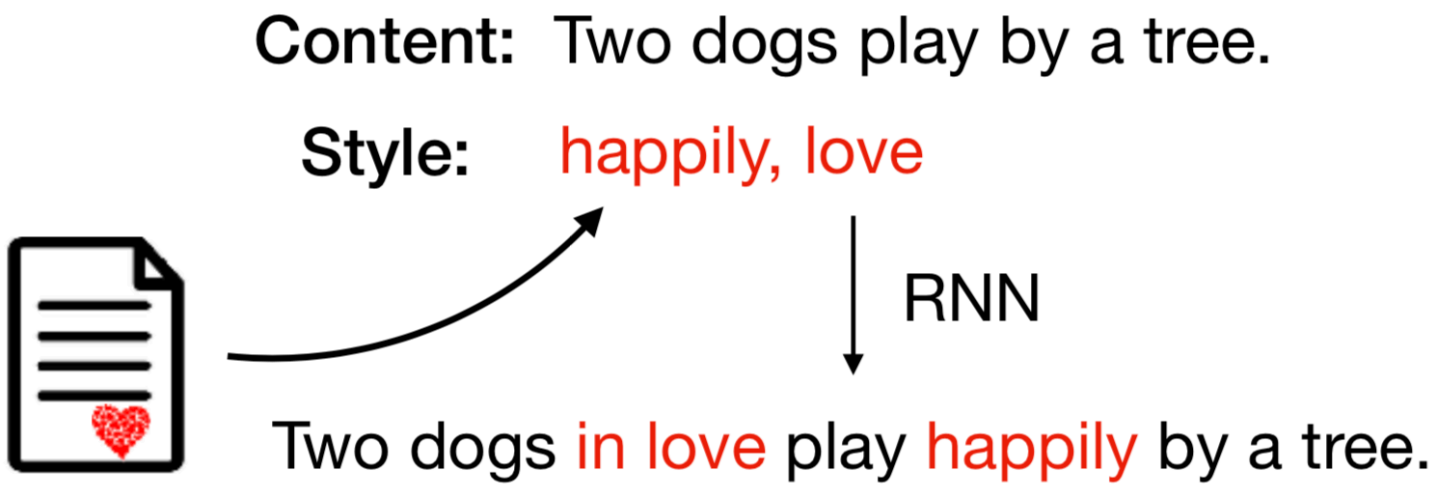
Machine Translation



https://www.pcmag.com/news/349610/google-expands-neural-networks-for-language-translation

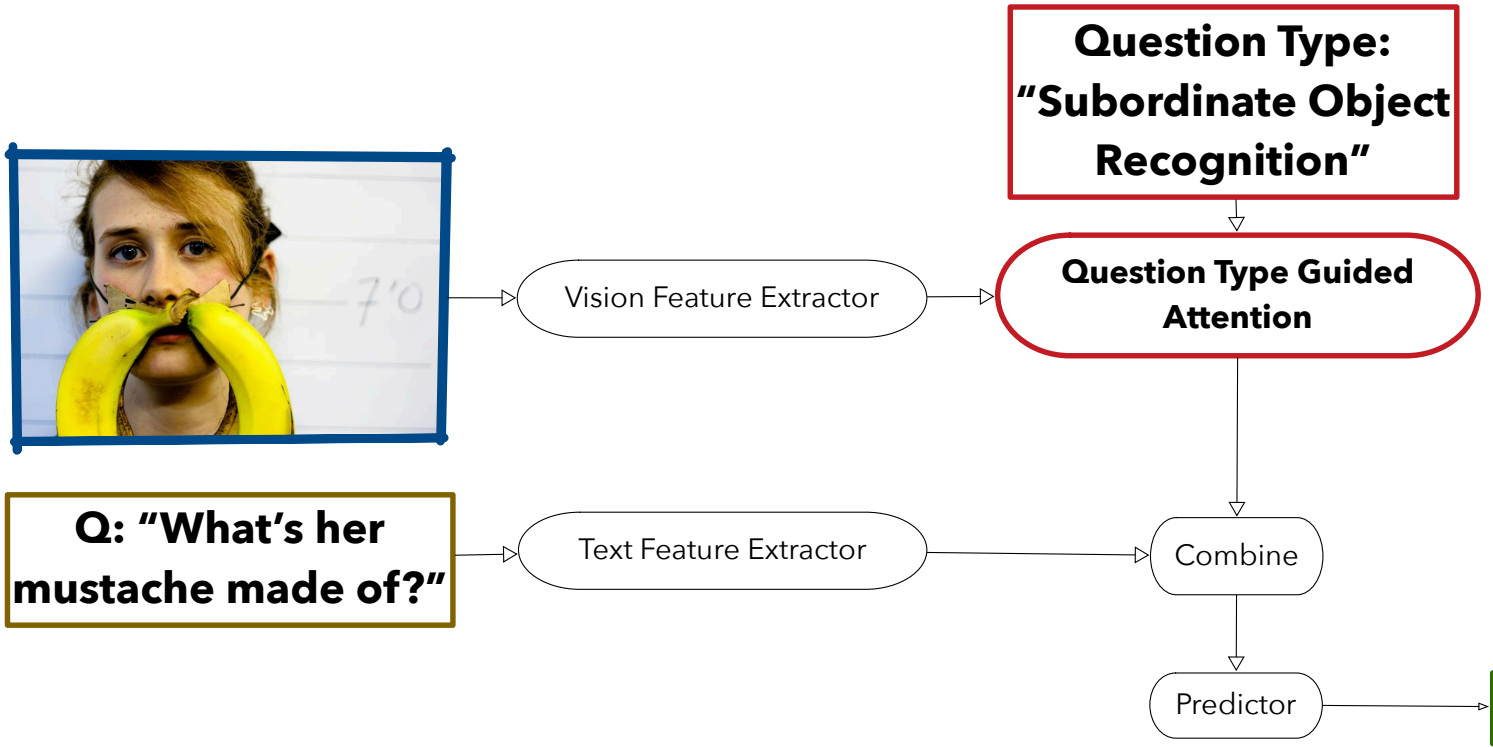
aws

Text synthesis



Li et al, NACCL, 2018

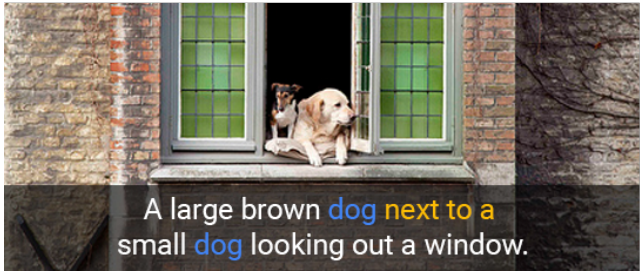
Question answering



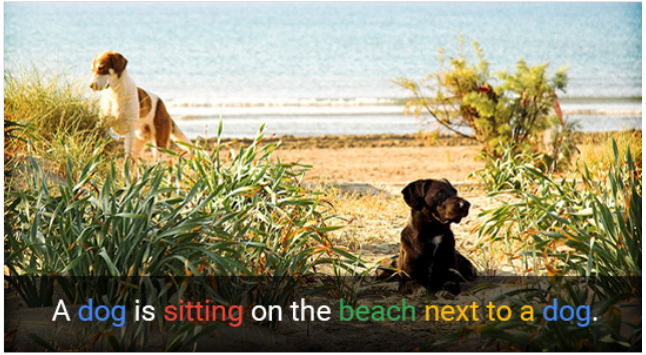
Shi et al, 2018, Arxiv

Image captioning

Human captions from the training set



Automatically captioned



Shallue et al, 2016
https://ai.googleblog.com/2016/09/show-and-tell-image-captioning-open.html

OK, I’m sold … I want this! Now!

Programming for deep learning used to be…

```
void computeLogregSoftmaxGrad(NVMatrix& labels, NVMatrix& probs, NVMatrix& target, bool add, float coeff) {
    int numCases = probs.getLeadingDim();
    int numOut = probs.getFollowingDim();
    assert(labels.getNumElements() == numCases);
    assert(probs.isContiguous());
    assert(target.isContiguous());
    assert(labels.isContiguous());
    assert(probs.isTrans());

    dim3 threads(LOGREG_GRAD_THREADS_X, LOGREG_GRAD_THREADS_Y);
    dim3 blocks(DIVUP(numCases, LOGREG_GRAD_THREADS_X), DIVUP(numOut, LOGREG_GRAD_THREADS_Y));
    if (!add) {
        target.resize(probs);
        kLogregSoftmaxGrad<false><<<blocks, threads>>>(probs.getDevData(), labels.getDevData(), target.getDevData(),
                                                       numCases, numOut, coeff);
    } else {
        kLogregSoftmaxGrad<true><<<blocks, threads>>>(probs.getDevData(), labels.getDevData(), target.getDevData(),
                                                      numCases, numOut, coeff);
    }

    getLastCudaError("computeLogregSoftmaxGrad: Kernel execution failed");
}
```

Krizhevsky et al, code from cuda-convnet (AlexNet)

Programming for deep learning today

```
loss_function = gluon.loss.SoftmaxCrossEntropyLoss()
loss = loss_function(output)
loss.backward()
```

Programming for deep learning today

```
loss_function = gluon.loss.SoftmaxCrossEntropyLoss()
loss = loss_function(output)
loss.backward()
```

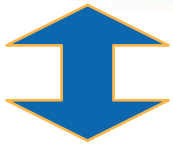
aws

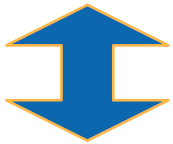
Programming for deep learning today

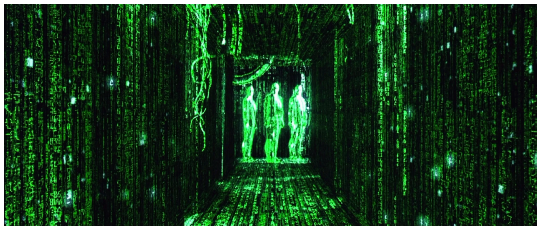
```
loss_function = gluon.loss.SoftmaxCrossEntropyLoss()
loss = loss_function(output)
loss.backward()
```

Programming for deep learning today


```
output = model(data)
```

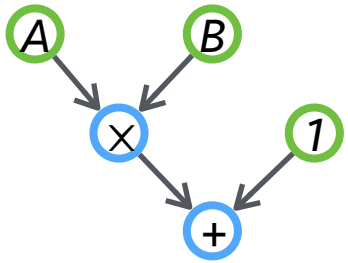






[crazy GPU stuff]

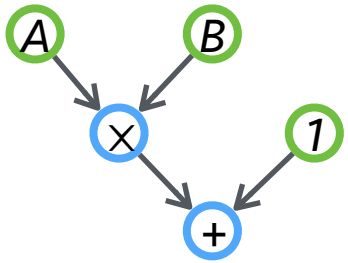
Declarative (Symbolic) Programs




```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + 1
f = compile(D)
d = f(A=np.ones(10),
      B=np.ones(10)*2)
```

- **Advantages**:
  - Opportunities for optimization
  - Easy to serialize models
  - More portable across languages
- **Disadvantages**:
  - Hard to debug
  - Unsuitable for dynamic graphs
  - Can’t use native code

Declarative (Symbolic) Programs




```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + 1
f = compile(D)
d = f(A=np.ones(10),
      B=np.ones(10)*2)
```

C can share memory with *D*

- **Advantages**:
  - Opportunities for optimization
  - Easy to serialize models
  - More portable across languages
- **Disadvantages**:
  - Hard to debug
  - Unsuitable for dynamic graphs
  - Can't use native code

Imperative Programs


```
import numpy as np
a = np.ones(10)
b = np.ones(10) * 2
c = b * a
print c
d = c + 1
```

Easy to debug, easy to code

- **Advantages**
  - Straightforward and flexible
  - Use language natively (loops, control flow, debugging)
- **Disadvantages**
  - Hard to optimize (Fixed with JIT compiler!)

MXNet Gluon

Q: Why GLUON?

A:  + 

Laptop Setup

- Get today's notebooks
  `git clone https://github.com/szha/KDD18-Gluon`
- Set up Conda environment
  - `conda env update --prune mxnet/kdd18_gluon_nlp`
  - `conda activate kdd18_gluon_nlp`

Resources

- Deep Learning — the Straight Dope
  https://gluon.mxnet.io/
- A 60-minute Gluon Crash Course
  https://gluon-crash-course.mxnet.io/
- GluonCV
  http://gluon-cv.mxnet.io/
- GluonNLP
  https://gluon-nlp.mxnet.io/
- MXNet User Forum
  http://discuss.mxnet.io/
- MXNet Documentation
  https://mxnet.apache.org/

aws

Outline

8:30-9:15	Installation and Basics (NDArray, AutoGrad, Libraries)
9:15-9:30	Neural Networks 101 (MLP, ConvNet, LSTM, Loss, SGD) - Part I
9:30-10:00	Break
10:00-10:30	Neural Networks 101 (MLP, ConvNet, LSTM, Loss, SGD) - Part II
10:30-11:00	Computer Vision 101 (Gluon CV)
11:00-11:30	Parallel and distributed training
11:30-12:00	Data I/O in NLP (and iterators)
12:00-13:30	Break
13:30-14:15	Embeddings
14:15-15:00	Language models (LM)
15:00-15:30	Sequence Generation from LM
15:30-16:00	Break
16:00-16:15	Sentiment analysis
16:15-17:00	Transformer Models & machine translation
17:00-17:30	Questions

NDArray and AutoGrad

Matrices and Vectors

- **A Simple ML algorithm**

```
initialize function f(x,w)
for (x,y) in data:
      l = loss(y, f(x,w))
      g = gradient(l)
      w = w - eta * g
```

- Need vectors to deal with w
- Need matrices to deal with chain rule for loss gradient
- Need high performance linear algebra

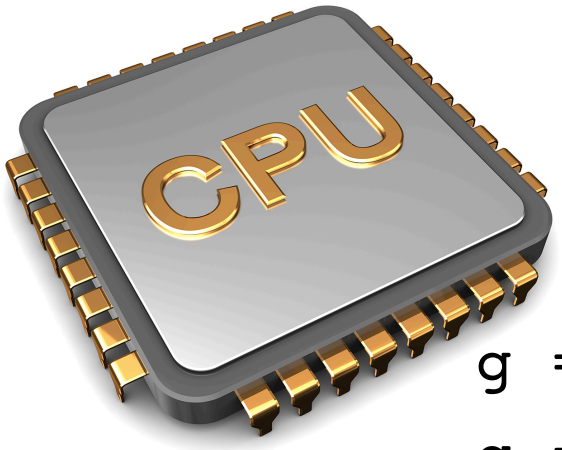
NDArray (Linear Algebra on GPUs in Python)

- **NumPy**
  - Limited to CPUs
  - No automatic differentiation
  - **Blocking (returns only once computation is performed)**
- **NDArray**
  - Multiple CPUs / GPUs via device context
  - Distributed systems in the cloud
  - **Nonblocking Python frontend (no GIL)**
  - **Lazy evaluation**

NDArray Device Contexts

```
context = mx.cpu()
```

```
context = mx.gpu(0)
context = mx.gpu(1)
```



```
g = copyto(c)
g = c.as_in_context(mx.gpu(0))
```

Automatic Differentiation

```
x = nd.array([[1, 2], [3, 4]])
x.attach_grad()
with ag.record():
    y = x * 2
    z = y * x
z.backward()
```

$$\partial_x \sum_i z[i](x) \textbf{ not } \partial_x z[i](x)$$

AutoGrad computes gradients without a declared compute graph