Introduction to Deep Learning

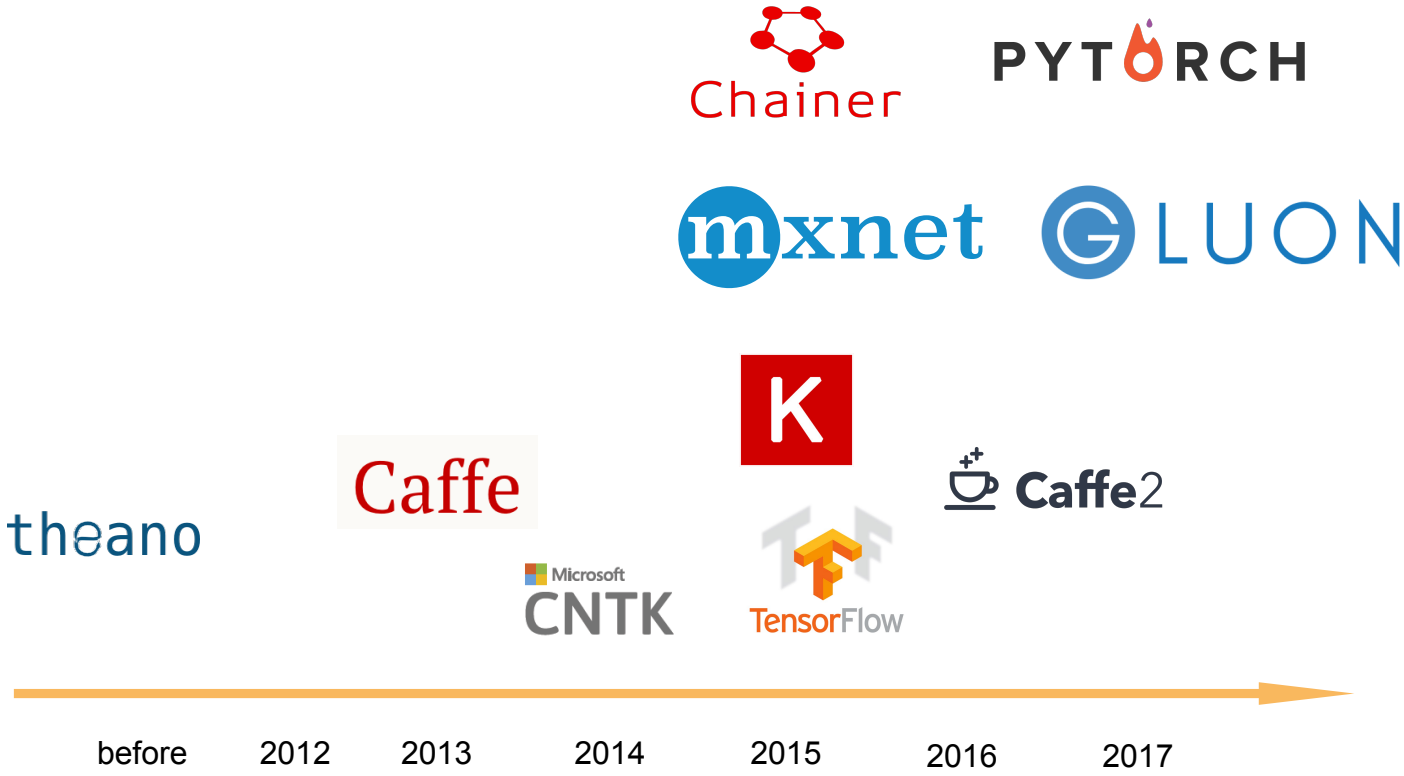
10. Layers, Blocks, Parameters and GPUs

STAT 157, Spring 2019, UC Berkeley

Alex Smola and Mu Li

courses.d2l.ai/berkeley-stat-157

aws


Chainer
PYTORCH
mxnet
GLUON
K
Caffe
Caffe2
theano
Microsoft
CNTK
TensorFlow
before
2012
2013
2014
2015
2016
2017

aws

Caffe

ResNet-101-deploy.prototext

```
layer {
        bottom: "data"
        top: "conv1"
        name: "conv1"
        type: "Convolution"
        convolution_param {
                num_output: 64
                kernel_size: 7
                pad: 3
                stride: 2
```

- Protobuf as the interface
- Good CV model coverage
- Portable
- Not flexible to develop

Tensorflow

Implement Adam

```
# m_t = beta1 * m + (1 - beta1) * g_t
m = self.get_slot(var, "m")
m_scaled_g_values = grad.values * (1 - beta1_t)
m_t = state_ops.assign(m, m * beta1_t,
                       use_locking=self._use_locking)
m_t = state_ops.scatter_add(m_t, grad.indices, m_scaled_g_values,
                            use_locking=self._use_locking)
```

- A domain specific language (DSL) for Python
- A rich set of operators
- Rich features
- Codes are not very easy to read

Keras

```
model = Sequential()
model.add(Dense(512, activation='relu',
                input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.compile(...)
model.fit(...)
```

- Simple DSL for Python, can use multiple backend (TensoFlow, MXNet, CNTK…)
- Easier to use than TensorFlow
- May be slower
- Less convenient to develop and debug

Pytorch

```
class Net(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        return out
```

- Torch tensors + chainer neural networks
- Easy to develop and debug
- Less convenient to deploy

MXNet

Implement Resnet

```
bn1 = sym.BatchNorm(data=data, fix_gamma=Fal
act1 = sym.Activation(data=bn1, act_type='re
conv1 = sym.Convolution(data=act1, num_filte
```

Implement Adam

```
coef2 = 1. - self.beta2**t
lr *= math.sqrt(coef2)/coef1

weight -= lr*mean/(sqrt(variance) + self.epsilon)
```

- Numpy-like Tensor + Keras-like neural networks
- Performance
- Usability

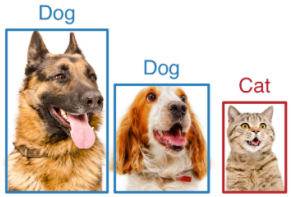
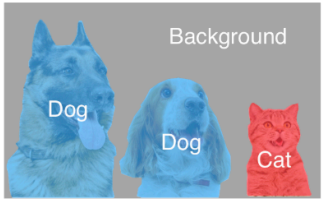
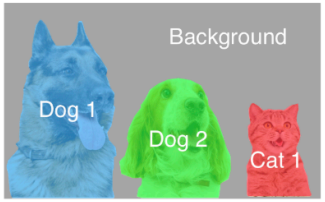
MXNet + Gluon

```
net = gluon.nn.Sequential()
net.add(gluon.nn.Dense(128, activation='relu'))
net.add(gluon.nn.Dense(64, activation='relu'))
net.add(gluon.nn.Dense(10))

loss = gluon.loss.SoftmaxCrossEntropyLoss()

for data, label in get_batch():
    with autograd.record():
        l = loss(net(data), label)
    l.backward()
    trainer.step(batch_size=data.shape[0])
```

- Numpy-like tensor + Chainer/pytorch-like neural network
- Easy to develop and debug
- Performance

GluonCV

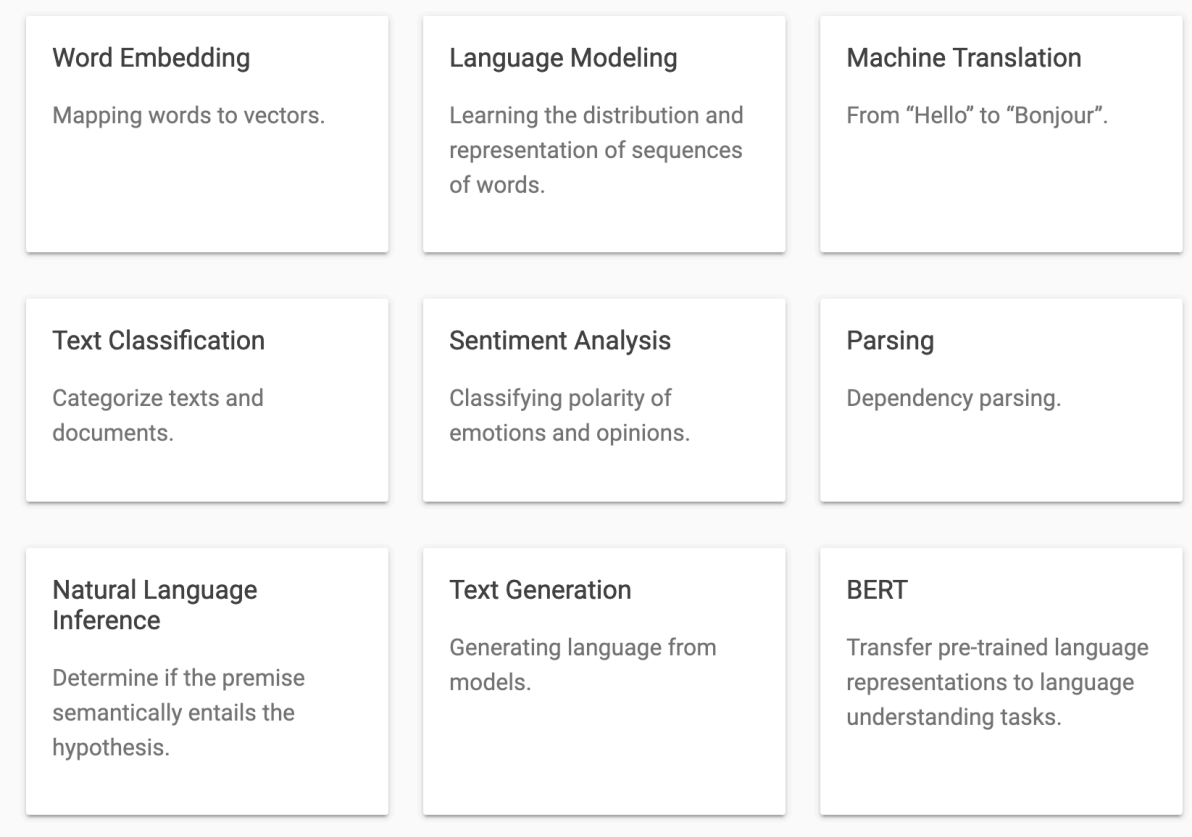
- Toolkit for computer vision
- https://gluon-cv.mxnet.io/
- Pre-trained models
- Training scripts to reproduce SOTA results

Application	Illustration	Available Models
Image Classification: recognize an object in an image.		50+ models, including ResNet, MobileNet, DenseNet, VGG, ...
Object Detection: detect multiple objects with their bounding boxes in an image.		Faster RCNN, SSD, Yolo-v3
Semantic Segmentation: associate each pixel of an image with a categorical label.		FCN, PSP, DeepLab v3
Instance Segmentation: associate each pixel of an image with an instance label.		Mask RCNN

courses.d2l.ai/berkeley-stat-157

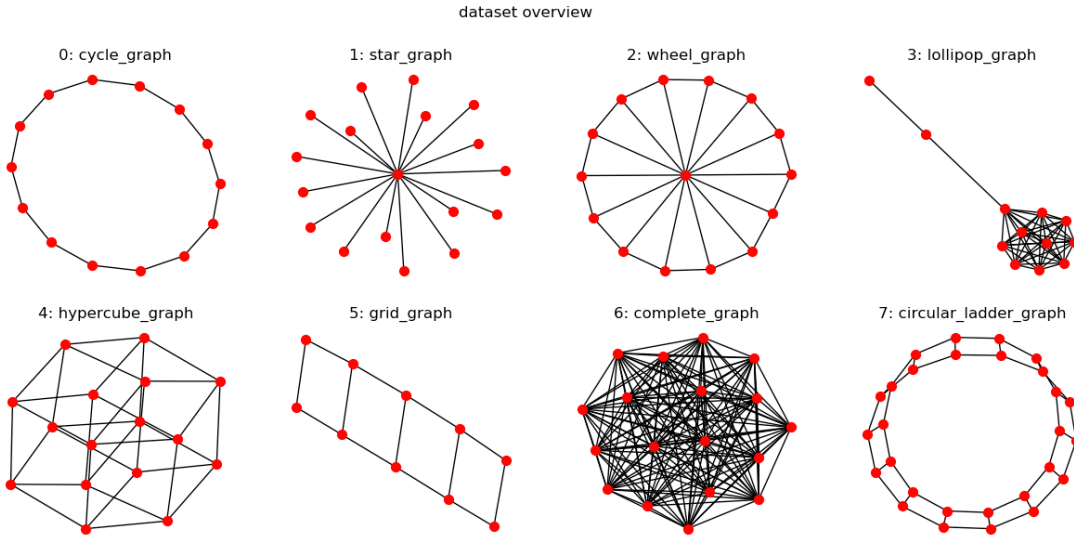
GluonNLP

- Toolkit for NLP
- https://gluon-nlp.mxnet.io/
- Pre-trained models
- Training scripts to reproduce SOTA results



DGL

- Toolkit for graph neural networks
- https://www.dgl.ai/
- Relatively new, but with good model coverage
- Has both MXNet and PyTorch backend



Roadmap for 2019

- More toolkits
  - Time series, AutoML, …
- 100% numpy-compatible
  - A new np package in mxnet
- Compiler integration
  - 50% performance boost on CPU/GPU
  - More hardware coverage: edge, ASIC, …

Gluon - Imperative Neural Network API

- Create layers and neural networks
  - gluon.nn
- Initialize and update parameters
  - gluon.Parameter
- Use GPUs