MxNet Gluon

Basics, Computer Vision, NLP (and even more NLP)
Part III (Gluon Computer Vision)

Leonard Lausen
Haibin Lin
Alex Smola

aws

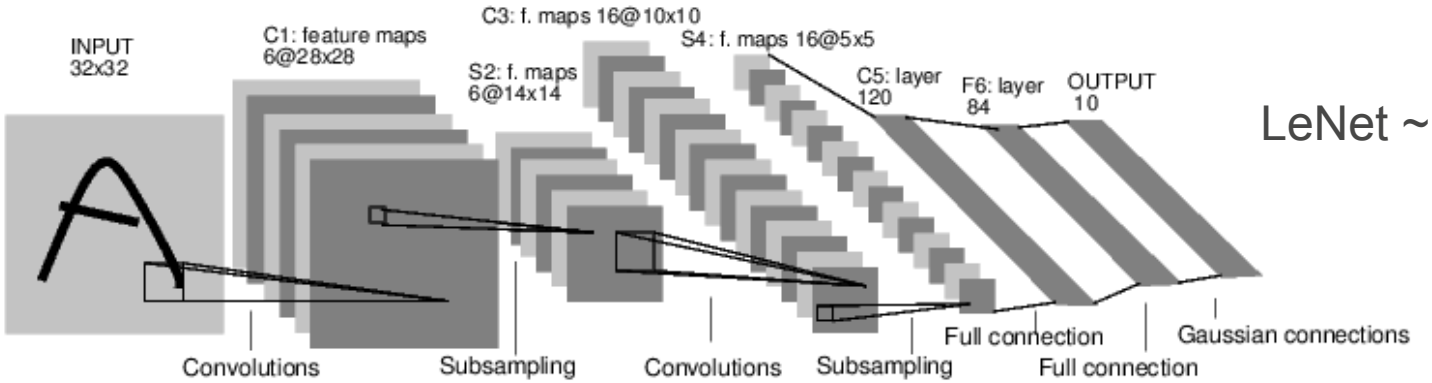
Outline

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

Computer Vision Architectures

LeNet 1995

- Multiple convolutions blow up dimensionality

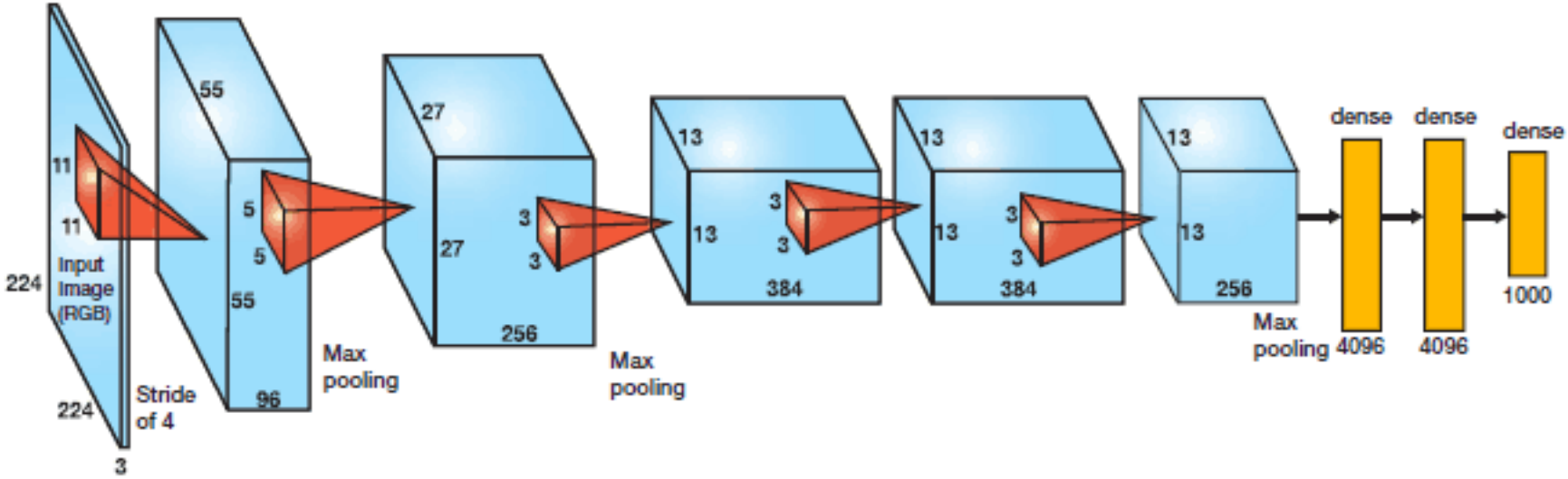


LeNet ~1995

- Subsampling - average over patches (works OK)
- MaxPooling - pick the maximum over patches (much better)

image credit - Le Cun et al, 1998

AlexNet (Krizhevsky et al., 2012)



- More convolutional layers
- More channels
- More filters
- More data

More computation

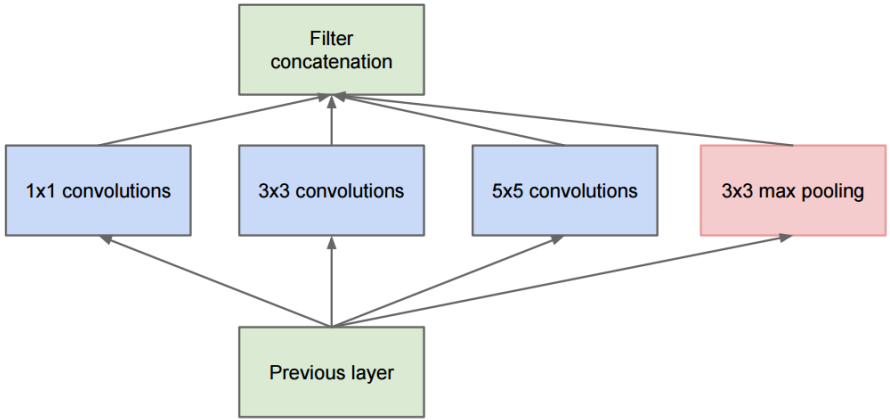
aws

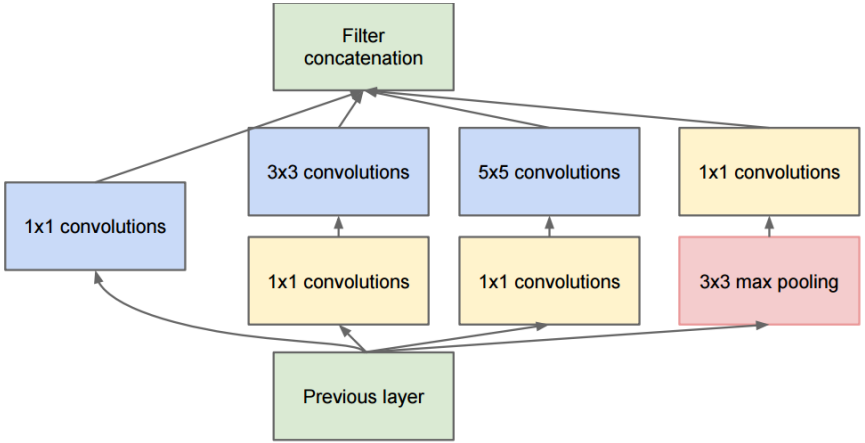
Deep and Narrow or Wide and Shallow

- Want to reach receptive field of size k
  - Use one large filter (linear mix of many, then nonlinearity)
  - Use several small filters (many linear mixes of few)
    This has fewer parameters
- Simonyan & Zisserman, 2014 find that deep and narrow wins

Fancy structures

- Compute different filters
- Compose one big vector from all of them
- Layer them iteratively

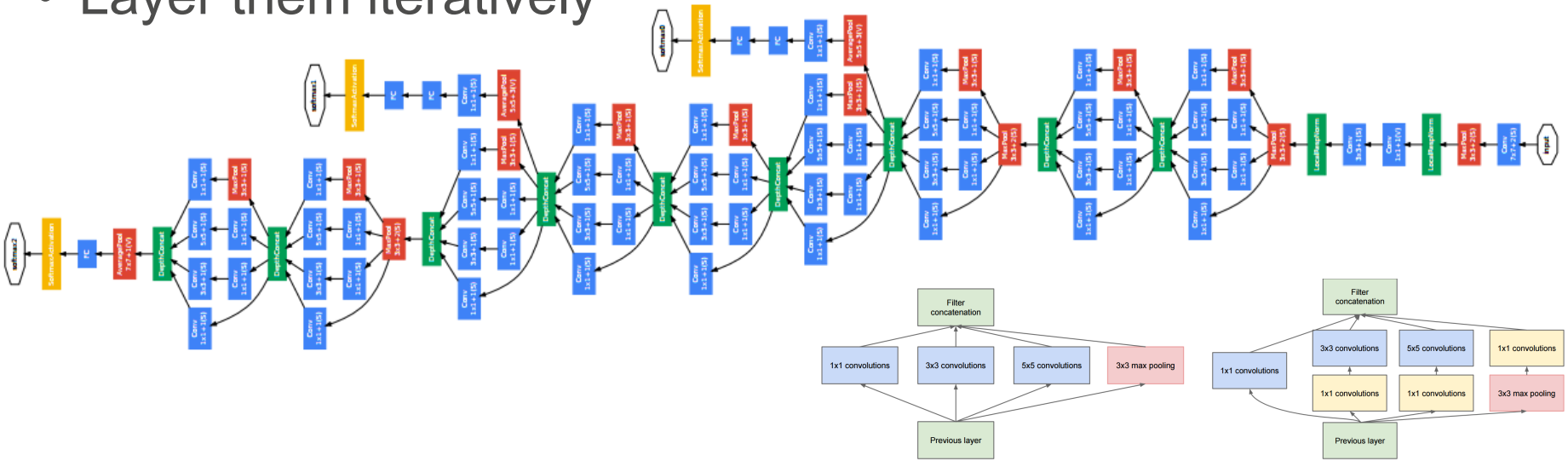




Szegedy et al. arxiv.org/pdf/1409.4842v1.pdf

Fancy structures

- Compute different filters
- Compose one big vector from all of them
- Layer them iteratively



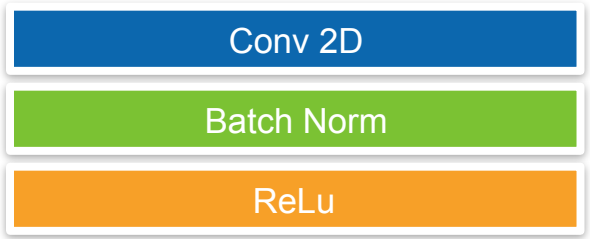
Szegedy et al. arxiv.org/pdf/1409.4842v1.pdf

amazon web services

Networks of networks - Sequential Composition

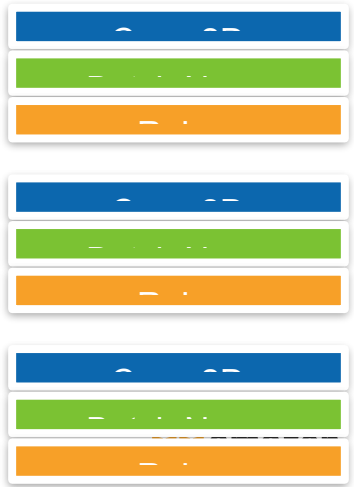
- **Define Single Block**

```
def _make_basic_conv(**kwargs):
  out = nn.HybridSequential(prefix='')
  out.add(nn.Conv2D(use_bias=False, **kwargs))
  out.add(nn.BatchNorm(epsilon=0.001))
  out.add(nn.Activation('relu'))
  return out
```



- **Compose Multiple Blocks into Branch**

```
def _make_branch(*conv_settings):
  out = nn.HybridSequential(prefix='')
  out.add(nn.MaxPool2D(pool_size=3, strides=2))
  setting_names = ['channels', 'kernel_size', 'strides', 'padding']
  for setting in conv_settings:
      kwargs = {}
      for i, value in enumerate(setting):
          if value is not None:
              kwargs[setting_names[i]] = value
      out.add(_make_basic_conv(**kwargs))
  return out
```



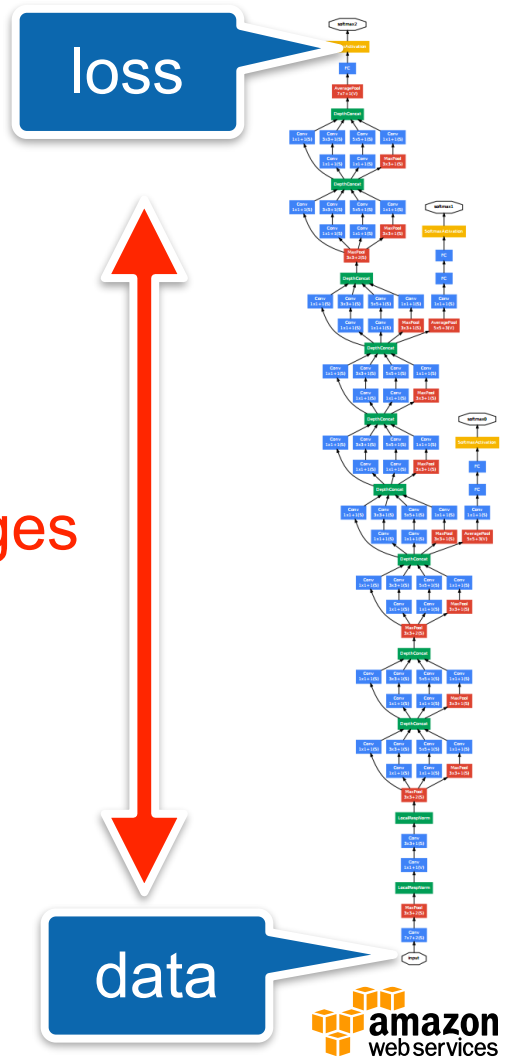
Networks of networks - Parallel Composition

- **Parallel composition of branches**

```
def def _make_A(pool_features, prefix):
    out = HybridConcurrent(concat_dim=1, prefix=prefix)
    with out.name_scope():
        out.add(_make_branch(None, (64, 1, None, None)))
        out.add(_make_branch(None, (48, 1, None, None),
                                   (64, 5, None, 2)))
        out.add(_make_branch(None, (64, 1, None, None),
                                   (96, 3, None, 1),
                                   (96, 3, None, 1)))
        out.add(_make_branch('avg',(pool_features, 1, None, None)))
    return out
```

Batch Norm (Ioffe et al., 2015)

- Loss occurs at last layer
  - Last layers learn quickly
- Data is inserted at bottom layer
  - Bottom layers change - **everything** changes
  - Last layers need to relearn many times
  - Slow convergence
- This is like covariate shift
  Can we avoid changing last layers while learning first layers?



Batch Norm (Ioffe et al., 2015)

- Can we avoid changing last layers while learning first layers?
- Fix mean and variance

$$\mu_B = \frac{1}{|B|}\sum_{i \in B} x_i \text{ and } \sigma_B^2 = \frac{1}{|B|}\sum_{i \in B}(x_i - \mu_B)^2 + \epsilon$$

and adjust it separately

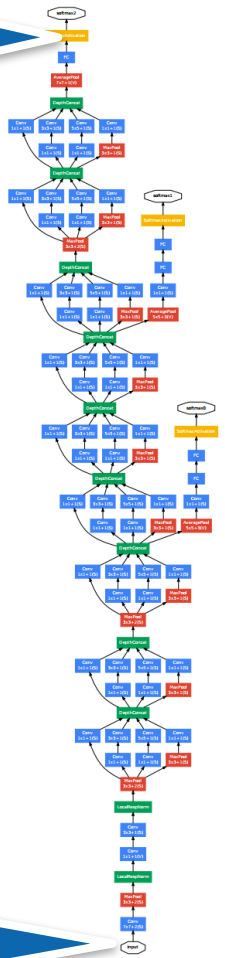
$$x_{i+1} = \gamma \frac{x_i - \mu_B}{\sigma_B} + \beta$$

mean

variance

loss

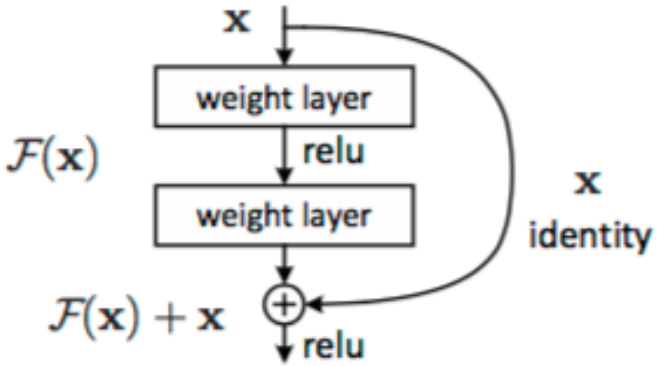
data

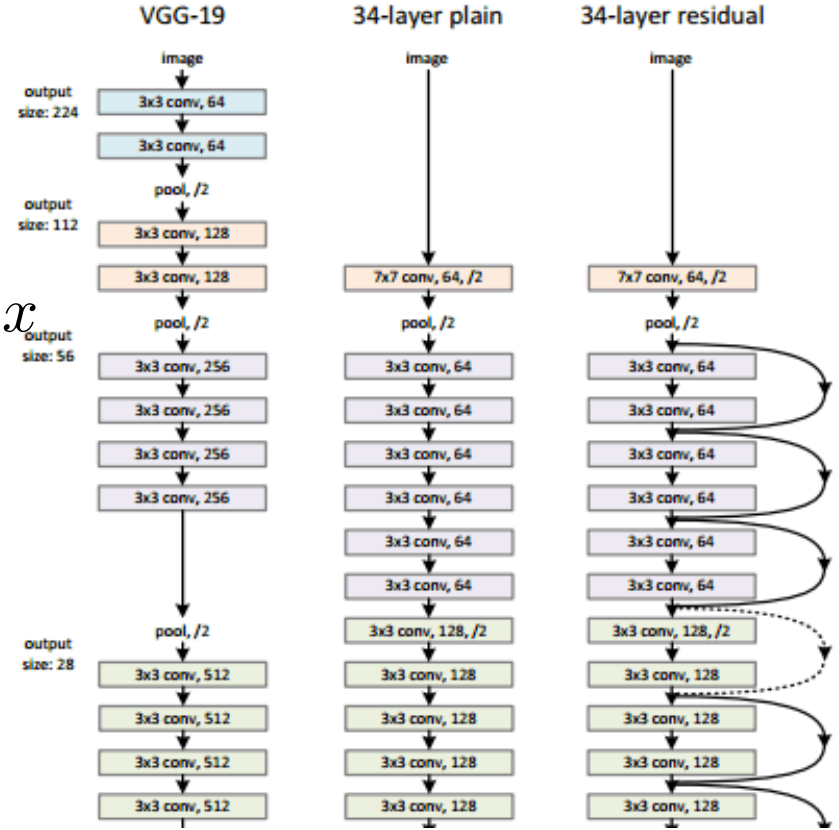


ResNet (He et al., 2015)

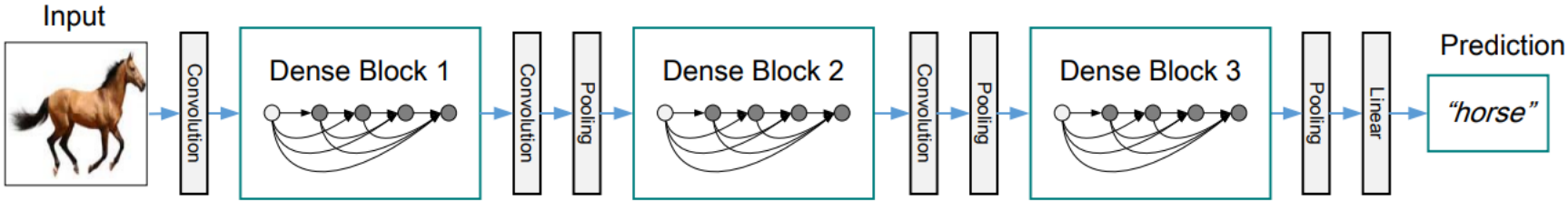
- In regular layer simple function is given by f(x) = 0
- Simple function should be $f(x) = x$
- Key idea - ‘Taylor expansion’

$$x + f(x)$$





DenseNet (Huang et al., 2016)



- Simple Network

  $f(x)$

- In ResNet 'Taylor expansion' ends after one term

  $x + f(x)$

- In DenseNet use multiple steps

$$[x, f(x), g([x, f(x)]), h([x, f(x), g([x, f(x)])]), \ldots]$$

Biggest Challenge

Biggest Challenge

Reproduce

Real world stories

- Back in 2016, the same ImageNet models trained by MXNet achieves on average 1% worse accuracy compared to Torch.
- Tried almost everything to debug, even developed a plugin to run Torch code inside MXNet to compare the results easily.

Real world stories

- Back in 2016, the same ImageNet models trained by MXNet achieves on average 1% worse accuracy compared to Torch.
- Tried almost everything to debug, even developed a plugin to run Torch code inside MXNet to compare the results easily.

- Transcoding training images using 95 JPEG quality rather than 85 solved the problem.

Real world stories

- Using another open source DL framework, a similar problem happened: trained model accuracies cannot match previous internal version.
- Spent months to figure out why, no clue …

Real world stories

- Using another open source DL framework, a similar problem happened: trained model accuracies cannot match previous internal version.
- Spent months to figure out why, no clue …

- The order of data augmentation is different from previous version.

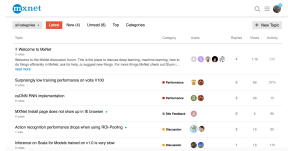
GluonCV approach

- Reproduction of important papers in recent years

- Training scripts (as well as tuned hyper-parameters) to reproduce the results

- Considerate APIs and modules that are easy to follow and understand, to simplify inventions

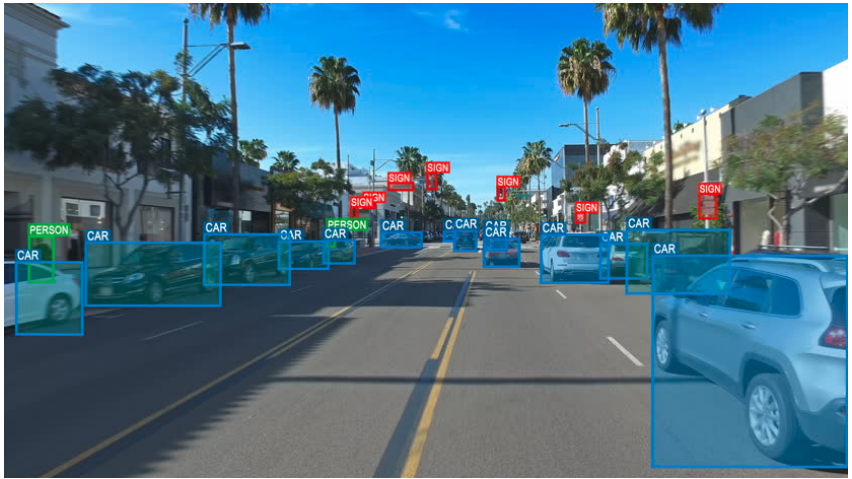
- Community support, feel free to ask and discuss



aws

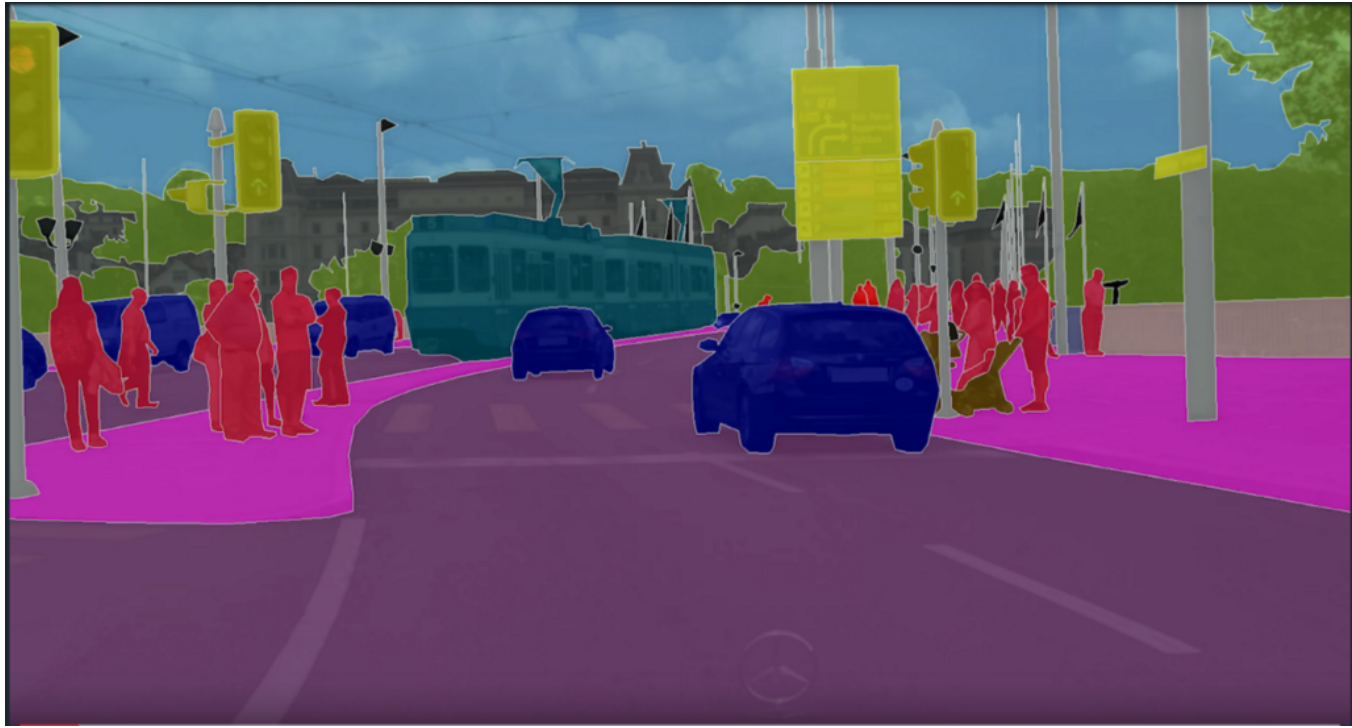
GluonCV Object Detection

- SSD and YOLOv3 - fastest solution
- Faster-RCNN, RFCN and FPN
  slower but more accurate, especially for tiny objects
- Mask-RCNN
  simultaneous object detection and semantic segmentation



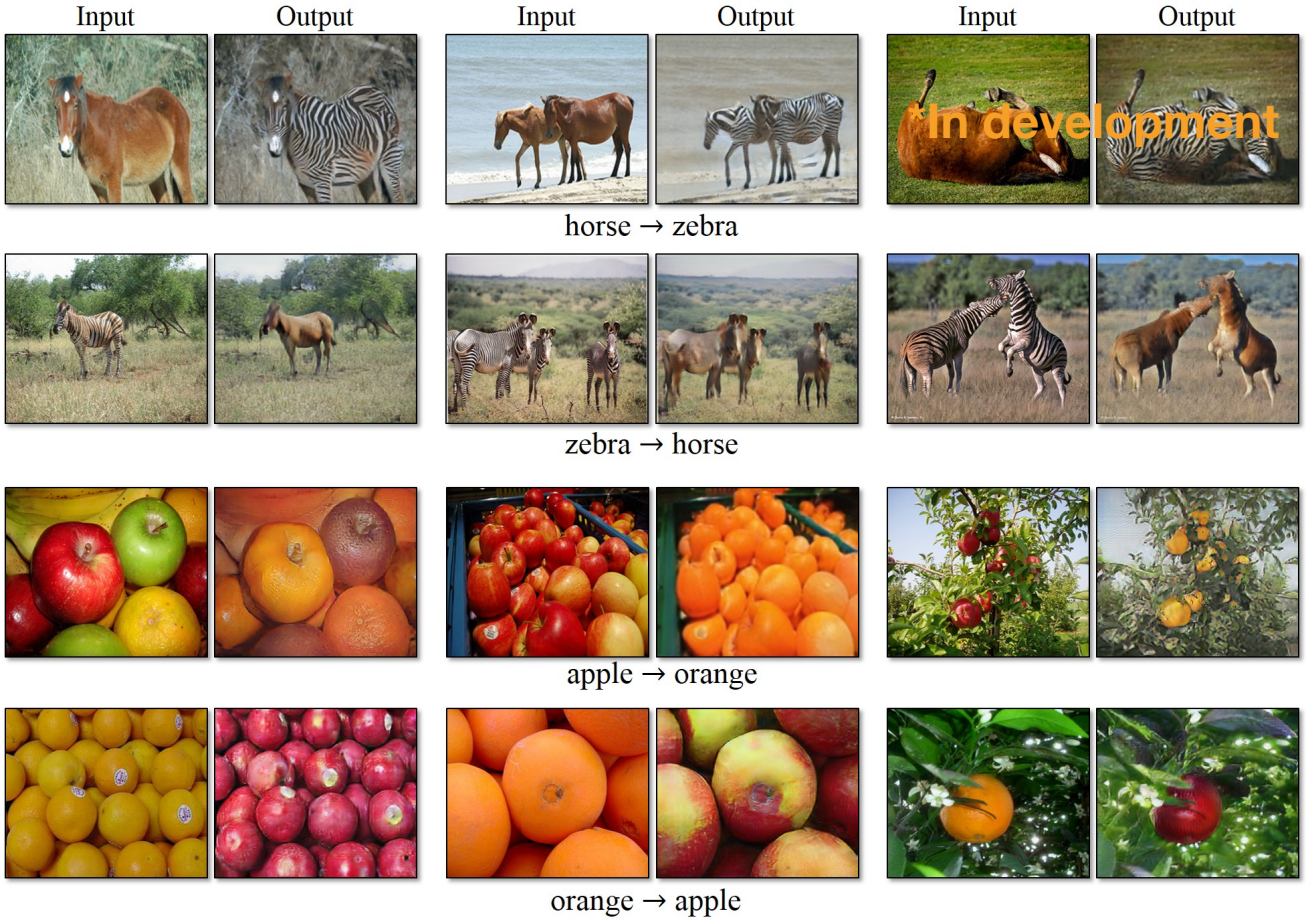
GluonCV Semantic Segmentation

- FCN
- PSPNet
- Mask-RCNN
- DeepLab



GluonCV Style Transfer (MSGNet)

GluonCV Generative Adversarial Networks



But this isn’t why you’re here …