AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

AGH UNIVERSITY OF SCIENCE
AND TECHNOLOGY

Scalability analysis of neural networks on multiple GPGPUs

Andrzej Dorobisz, Marcin Pietroń, Maciej Wielgosz,
Michał Karwatowski, Kazimierz Wiatr

Zakopane 07.03.2019

Agenda

- Deep learning frameworks
- Network architectures
- Scalability with batch size
- Scalability for multiple nodes

Caffe

- UC Berkeley

```
import caffe
caffe.set_device(0)
caffe.set_mode_gpu()
net = caffe.Net('conv.prototxt', caffe.TEST)
net.forward()
```

http://caffe.berkeleyvision.org/

```
name: "convolution"
input: "data"
input_dim: 1
input_dim: 1
input_dim: 100
input_dim: 100
layer {
  name: "conv"
  type: "Convolution"
  bottom: "data"
  top: "conv"
  convolution_param {
    num_output: 3
    kernel_size: 5
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

AGH

TensorFlow™

» Google Brain

```
def model_fn(features, labels, mode):
    logits_train = conv_net(features, num_classes, dropout,
reuse=False,
                            is_training=True)
    logits_test = conv_net(features, num_classes, dropout, reuse=True,
                           is_training=False)
    pred_classes = tf.argmax(logits_test, axis=1)
    ...
    estim_specs = tf.estimator.EstimatorSpec(
        mode=mode,
        predictions=pred_classes,
        loss=loss_op,
        train_op=train_op,
        eval_metric_ops={'accuracy': acc_op})
    return estim_specs

model = tf.estimator.Estimator(model_fn)
input_fn = tf.estimator.inputs.numpy_input_fn(
    x={'images': mnist.train.images}, y=mnist.train.labels,
    batch_size=batch_size, num_epochs=None, shuffle=True)

model.train(input_fn, steps=num_steps)
```

```
def conv_net(x_dict, n_classes, dropout, reuse, is_training):
    with tf.variable_scope('ConvNet', reuse=reuse):
        x = x_dict['images']
        x = tf.reshape(x, shape=[-1, 28, 28, 1])
        conv1 = tf.layers.conv2d(x, 32, 5, activation=tf.nn.relu)
        conv1 = tf.layers.max_pooling2d(conv1, 2, 2)
        conv2 = tf.layers.conv2d(conv1, 64, 3, activation=tf.nn.relu)
        conv2 = tf.layers.max_pooling2d(conv2, 2, 2)
        fc1 = tf.contrib.layers.flatten(conv2)
        fc1 = tf.layers.dense(fc1, 1024)
        fc1 = tf.layers.dropout(fc1, rate=dropout, training=is_training)
        out = tf.layers.dense(fc1, n_classes)
return out
```

https://www.tensorflow.org/

AGH

MatConvNet

» MATLAB toolbox

```
opts.train.batchSize = 100 ;
opts.train.numEpochs = 100 ;
opts.train.useGpu = true ;
opts.train.learningRate = 0.001 ;

opts = vl_argparse(opts, varargin) ;
imdb.images.data = bsxfun(@minus, imdb.images.data,
mean(imdb.images.data,4)) ;
if opts.train.useGpu
      imdb.images.data = gpuArray(imdb.images.data) ;
end

[net, info] = cnn_train(net, imdb, @getBatch, ...
      opts.train, ...
      'val', find(imdb.images.set == 3)) ;
```

```
net.layers = {} ;
net.layers{end+1} = struct('type', 'conv', ...
                           'filters', f*randn(5,5,1,20, 'single'), ...
                           'biases', zeros(1, 20, 'single'), ...
                           'stride', 1, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [2 2], ...
                           'stride', 2, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'conv', ...
                           'filters', f*randn(5,5,20,50, 'single'),...
                           'biases', zeros(1,50,'single'), ...
                           'stride', 1, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [2 2], ...
                           'stride', 2, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'conv', ...
                           'filters', f*randn(4,4,50,500, 'single'),...
                           'biases', zeros(1,500,'single'), ...
                           'stride', 1, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'relu') ;
net.layers{end+1} = struct('type', 'conv', ...
                           'filters', f*randn(1,1,500,10, 'single'),...
                           'biases', zeros(1,10,'single'), ...
                           'stride', 1, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'softmaxloss') ;
```

http://www.vlfeat.org/matconvnet/

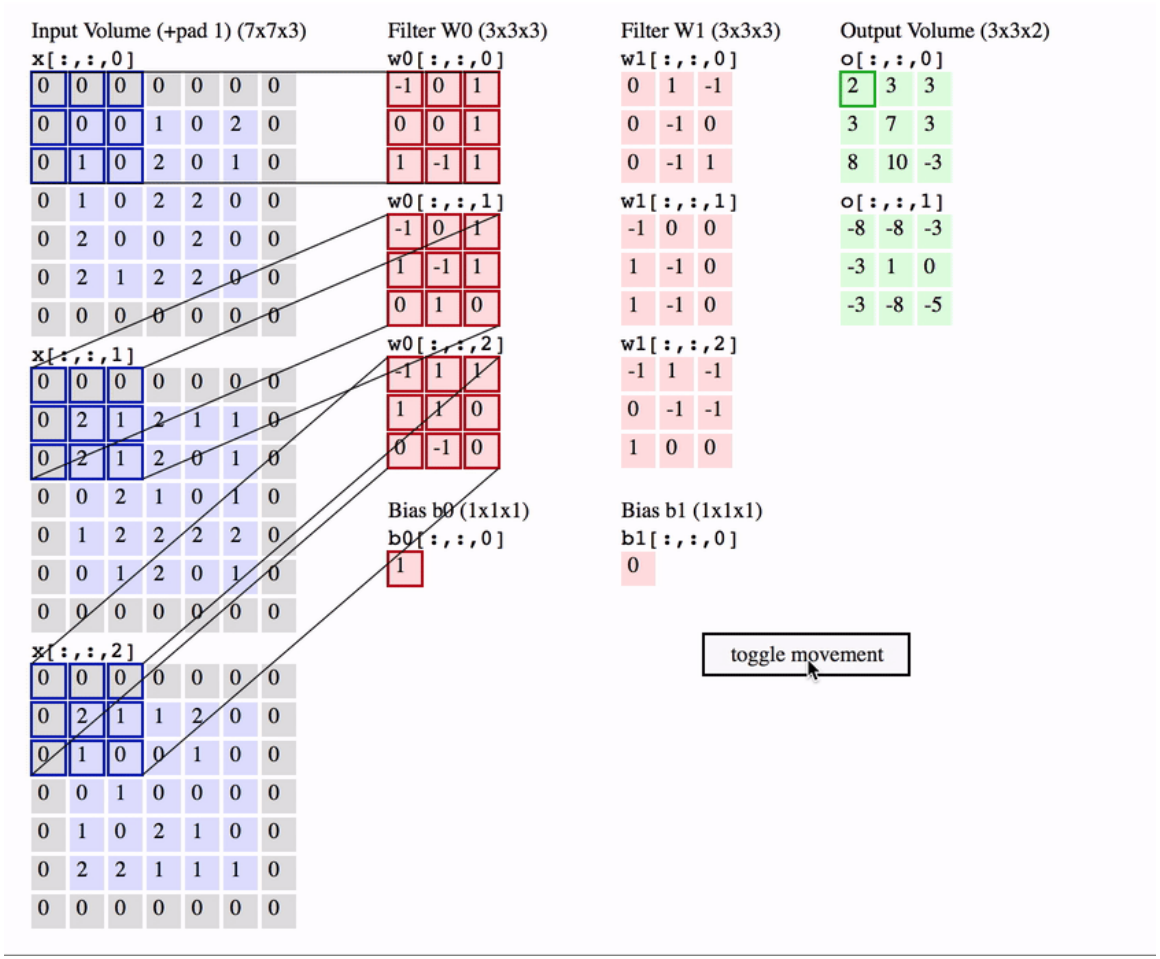
» Facebook

```
def train(args, model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()

device = torch.device("cuda" if use_cuda else "cpu")
model = Net().to(device)

for epoch in range(1, args.epochs + 1):
    train(args, model, device, train_loader, optimizer, epoch)
```

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 20, 5, 1)
        self.conv2 = nn.Conv2d(20, 50, 5, 1)
        self.fc1 = nn.Linear(4*4*50, 500)
        self.fc2 = nn.Linear(500, 10)


    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, 2, 2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2, 2)
        x = x.view(-1, 4*4*50)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
return F.log_softmax(x, dim=1)
```

https://pytorch.org/

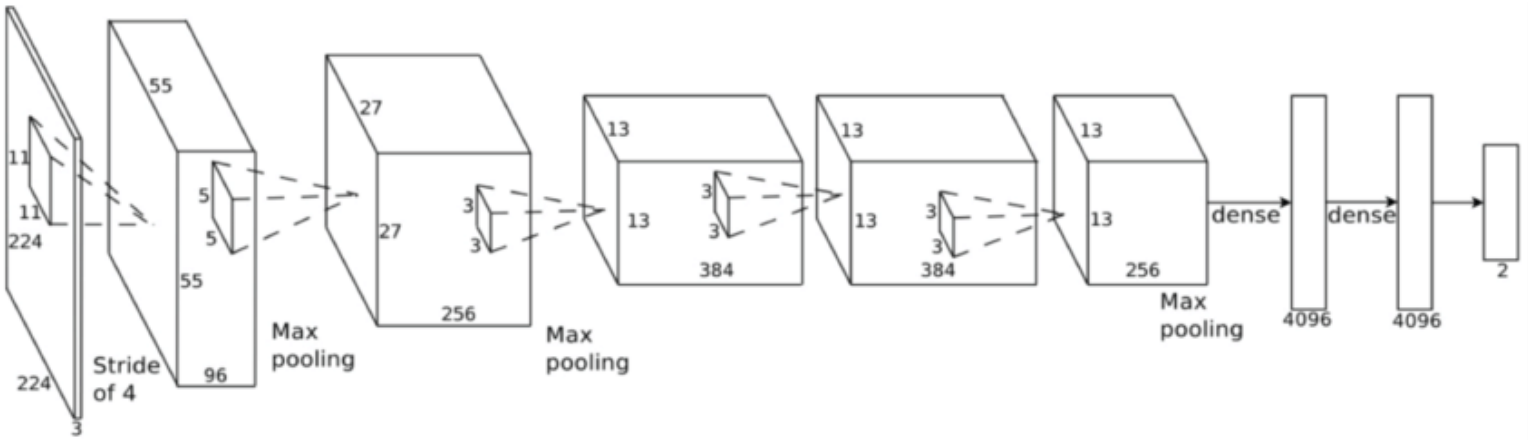
Convolutional Layer



$$y_i = b_i + \sum_{j=1}^{M} F_{ij} * x_j$$

http://cs231n.github.io/convolutional-networks/

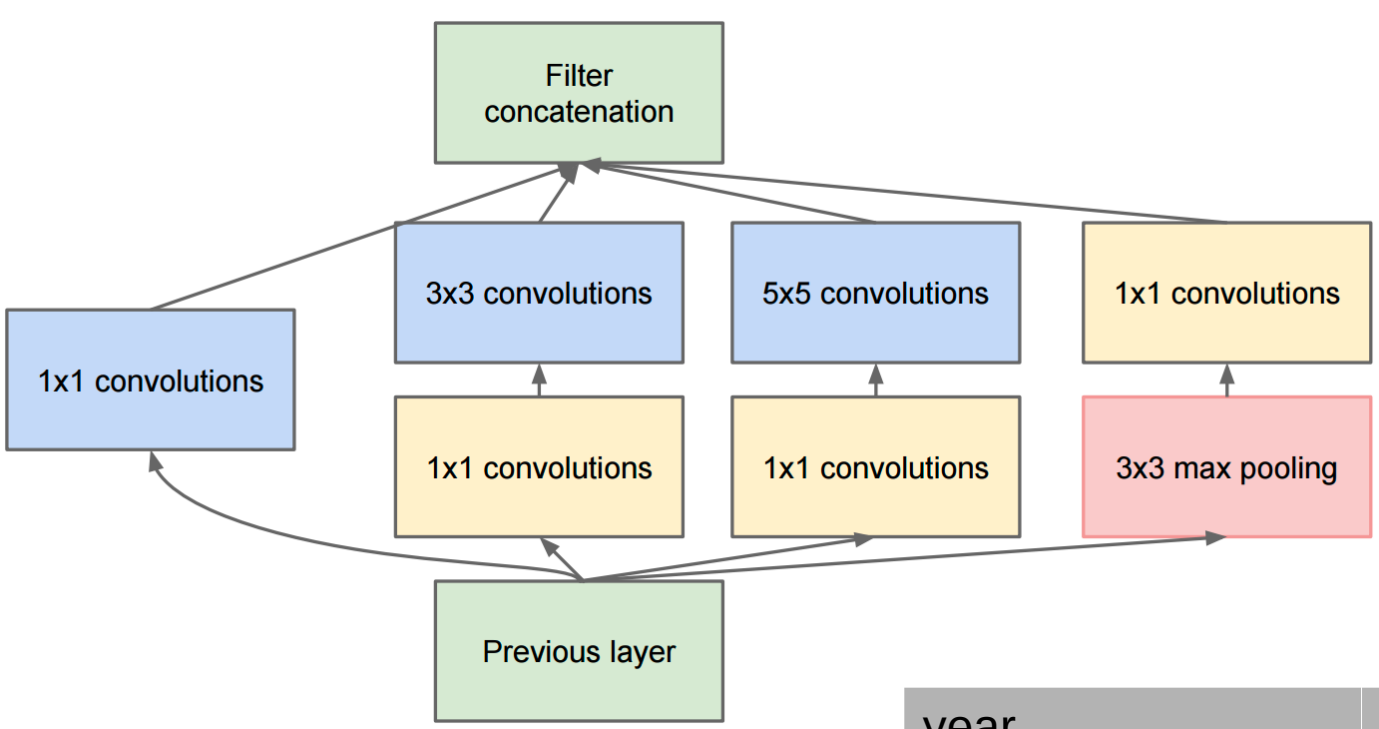
AlexNet



A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classifi-cation with deep convolutional neural networks,” inAdvances inneural information processing systems, 2012, pp. 1097–1105

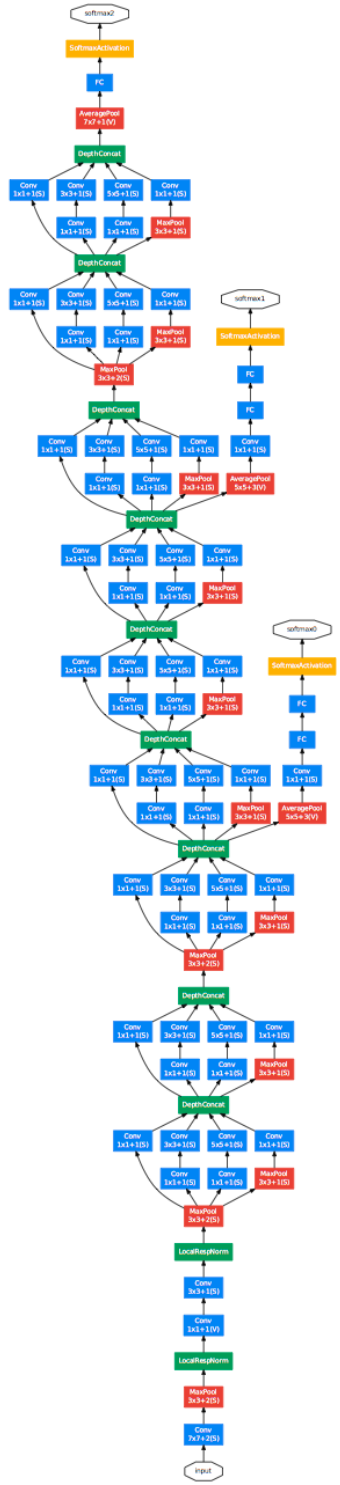
year	2012
# of conv lyers	5
# of lf layers	3
# of parameters	61M
# of MACs	724M

Inception v1

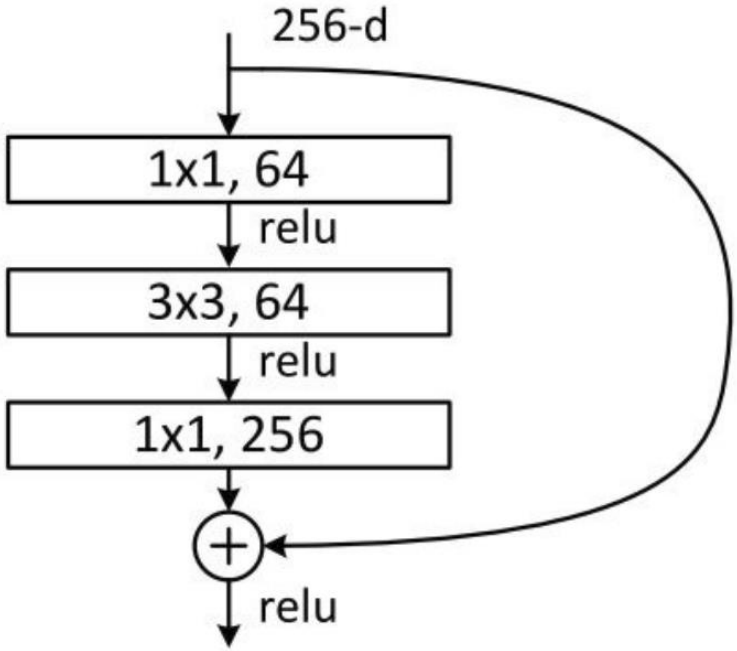


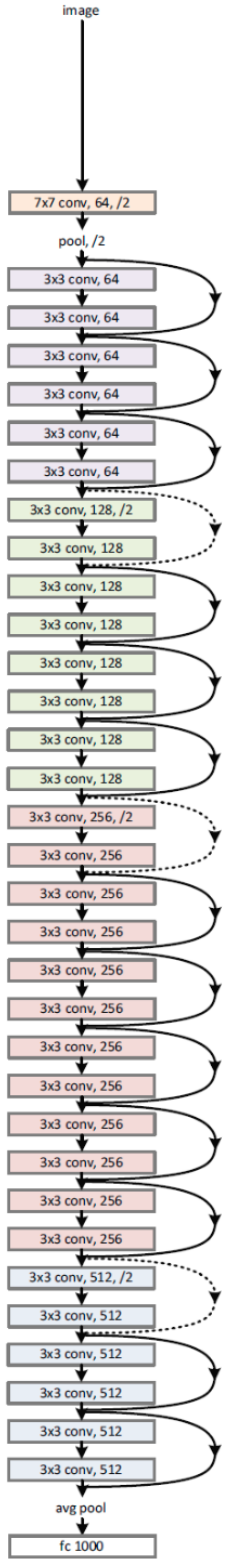
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

year	2014
# of conv lyers	21
# of lf layers	1
# of parameters	7M
# of MACs	1.43G



ResNet-50

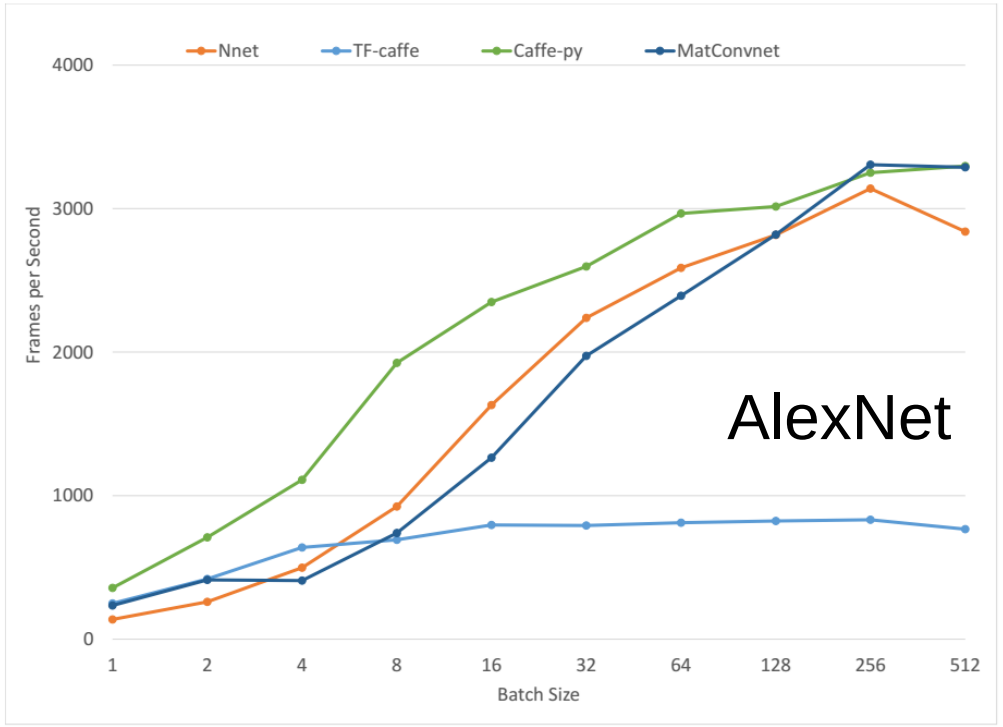


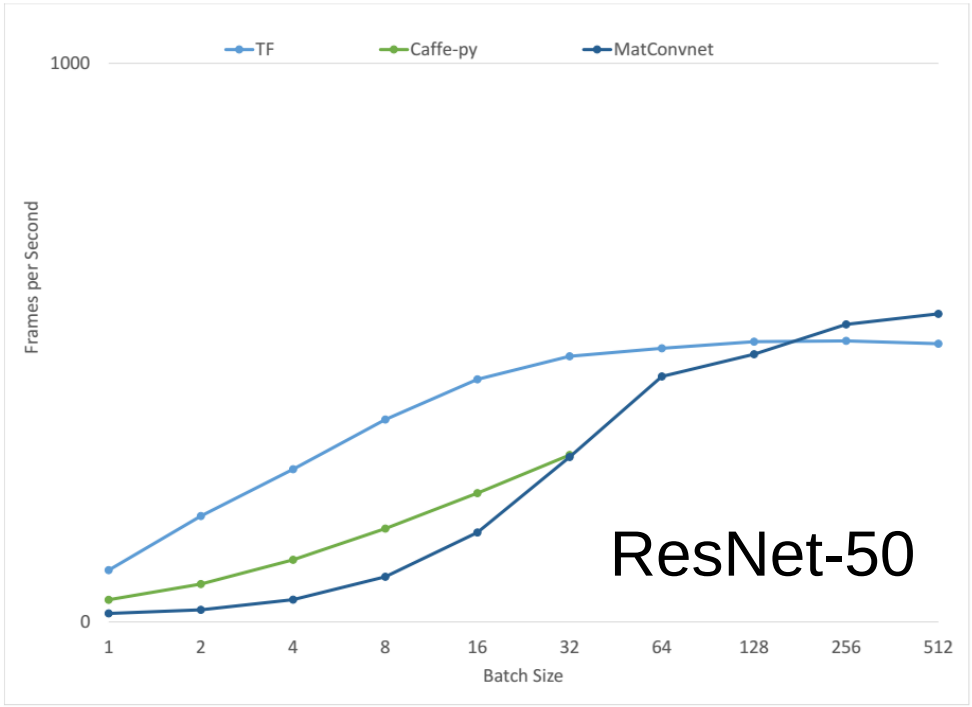


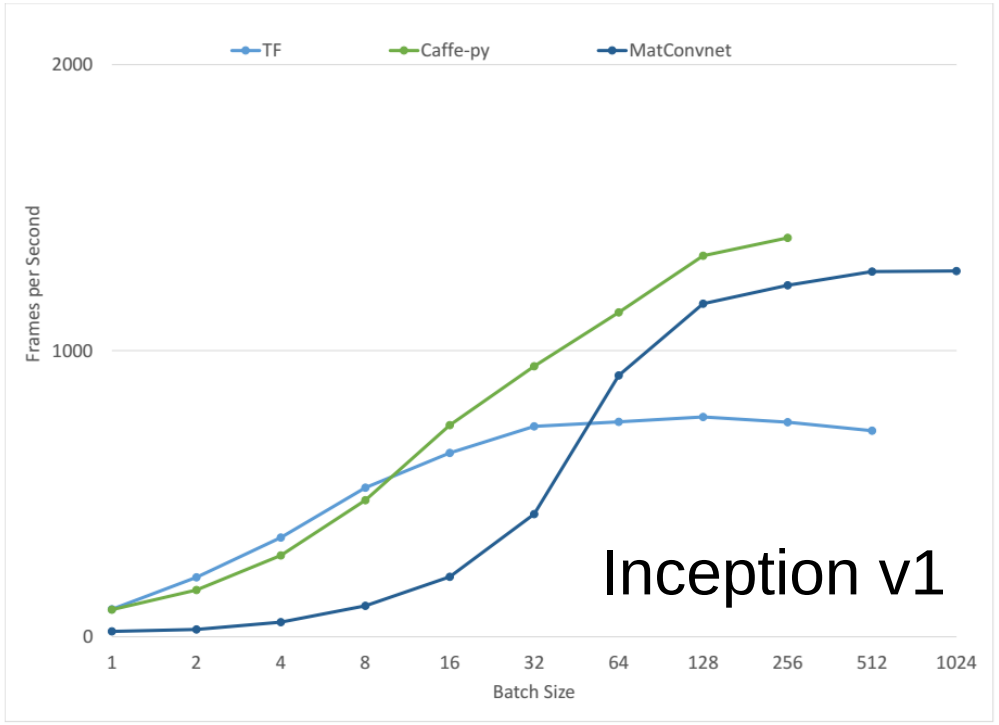
He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

year	2015
# of conv lyers	49
# of lf layers	1
# of parameters	25.5M
# of MACs	3.9G

Inference


Nnet
TF-caffe
Caffe-py
MatConvnet
4000
3000
2000
1000
0
Frames per Second
1
2
4
8
16
32
64
128
256
512
Batch Size
AlexNet


TF
Caffe-py
MatConvnet
1000
0
Frames per Second
1
2
4
8
16
32
64
128
256
512
Batch Size
ResNet-50


TF
Caffe-py
MatConvnet
2000
1000
0
Frames per Second
1
2
4
8
16
32
64
128
256
512
1024
Batch Size
Inception v1

Training

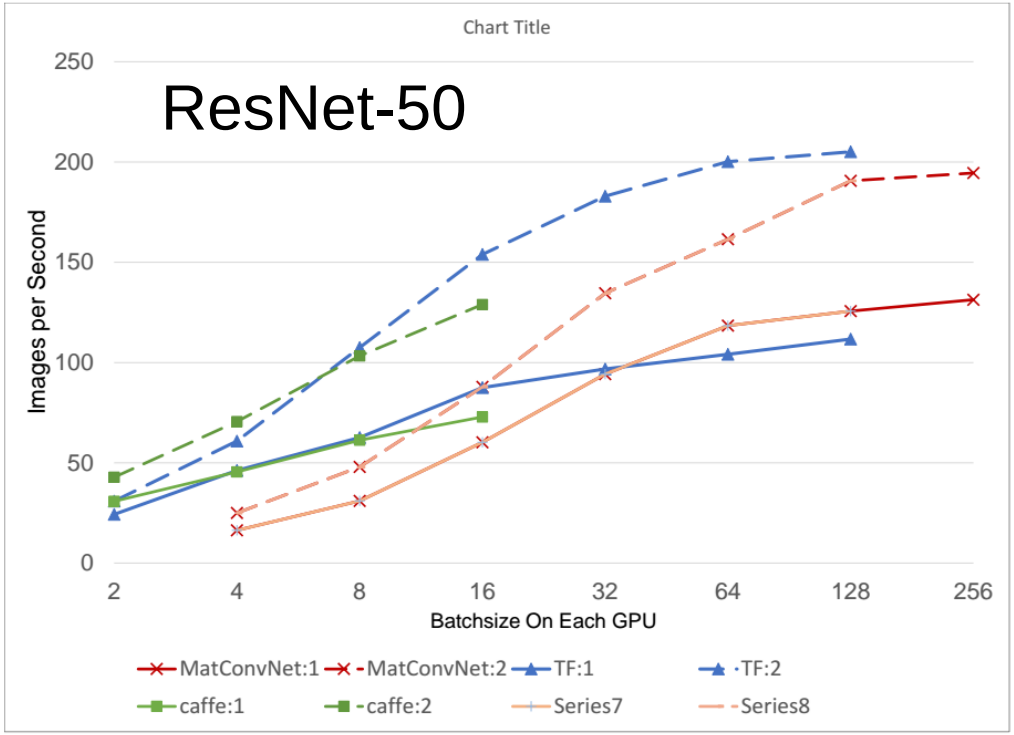
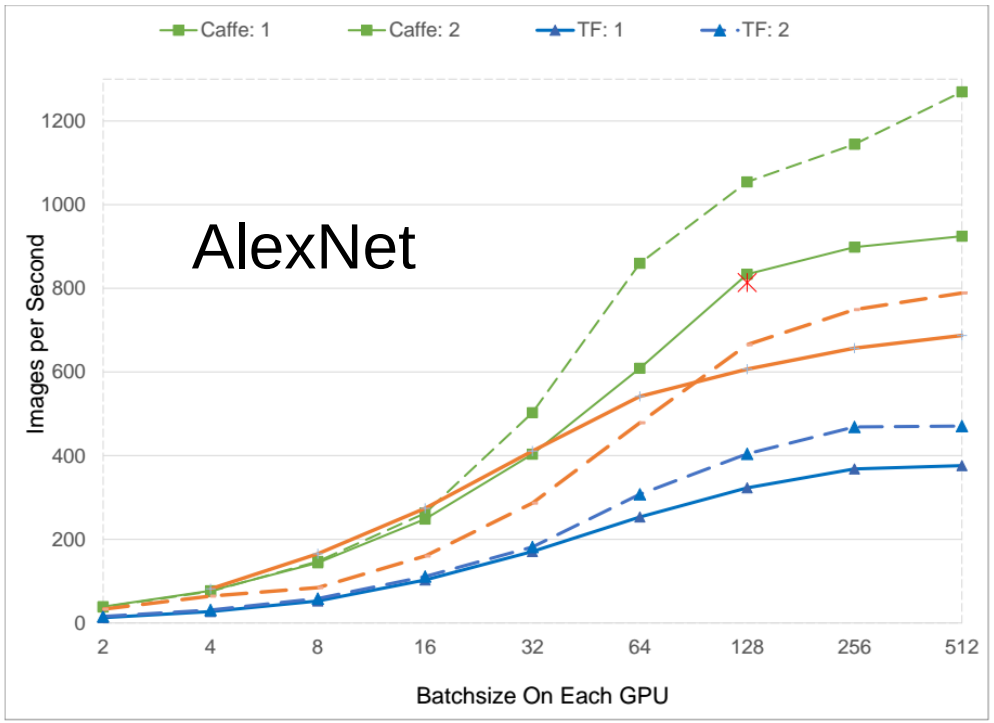
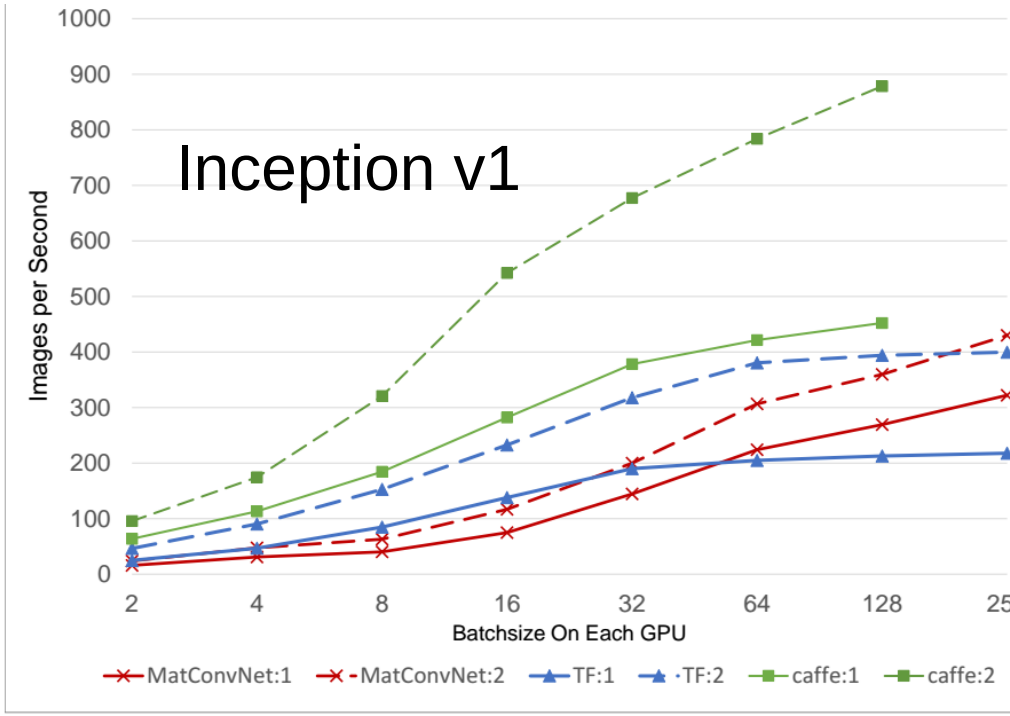


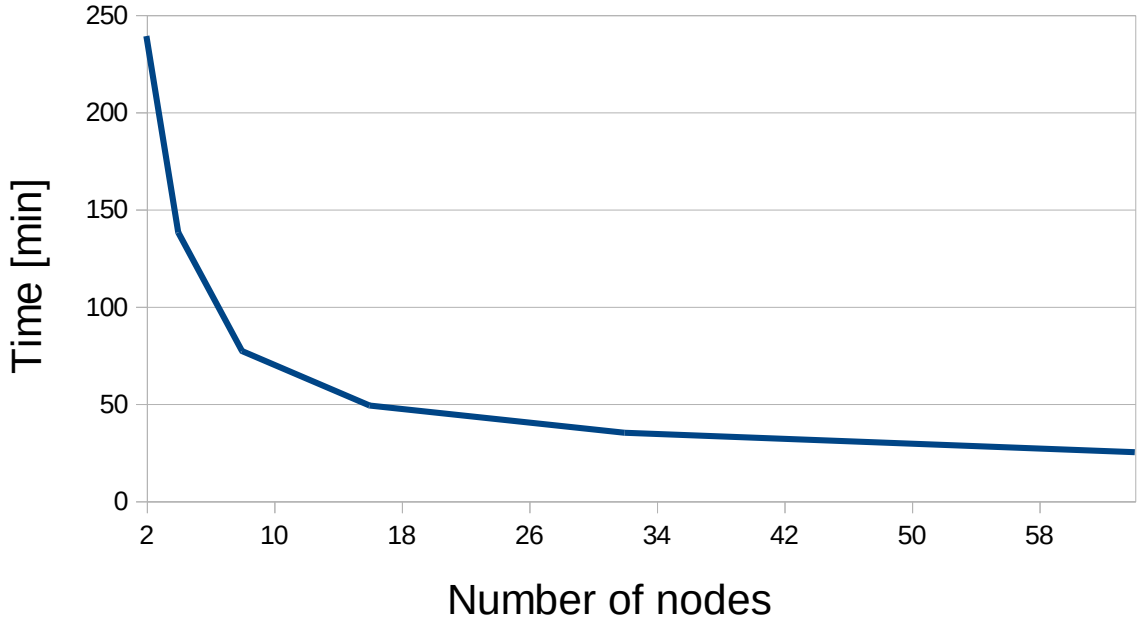
Chart Title
ResNet-50
Images per Second
Batchsize On Each GPU
MatConvNet:1
MatConvNet:2
TF:1
TF:2
caffe:1
caffe:2
Series7
Series8


Caffe: 1
Caffe: 2
TF: 1
TF: 2
AlexNet
Images per Second
Batchsize On Each GPU


Inception v1
Images per Second
Batchsize On Each GPU
MatConvNet:1
MatConvNet:2
TF:1
TF:2
caffe:1
caffe:2

Scalability for multiple nodes

ResNet-50 batch:64



Network	Batch size	Nodes	Time [min]
ResNet-152	32	4	1077
ResNet-152	32	8	696

AGH

Questions

?