SOFTWARE FOR LAB

Remote Desktop Software: Download NoMachine **now** for best performance from www.nomachine.com/download

- Alternatively you may use a VNC client or the provided browser-based VNC option

SSH Access Software (optional): PuTTy for Windows can be downloaded from www.putty.org

- Alternatively you may use a provided browser-based SSH option

GPU TECHNOLOGY CONFERENCE
April 4-7, 2016 | Silicon Valley

INTRODUCTION TO CNTK

Jonathan Bentz, NVIDIA, GTC 2016

PRESENTED BY
NVIDIA.

WHAT THIS LAB IS NOT

Intro to machine learning from first principles

Rigorous mathematical formalism of neural networks

Survey of all the features and options of CNTK

Secret sauce to win the ImageNet competition

WHAT THIS LAB IS

Discussion/Demonstration how CNTK works and some of its functionalities

Hands-on exercises using CNTK for neural network training

BACKGROUND ASSUMPTIONS

You are familiar with convolutional neural networks.

You know about things like forward and backpropagation, activations, stochastic gradient descent (SGD), convolutions, pooling, bias.

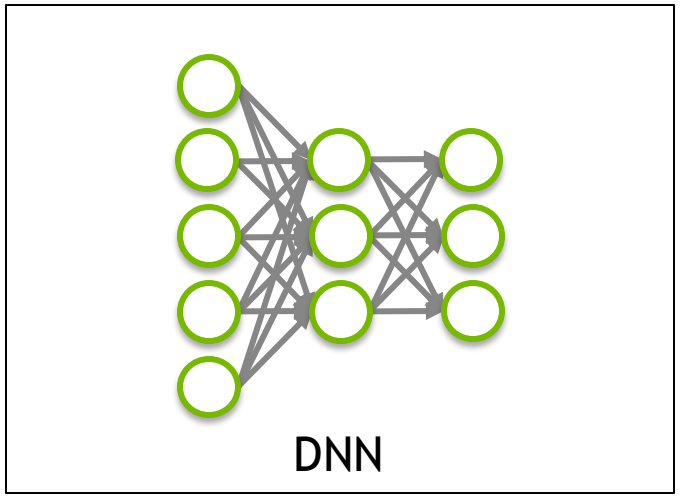
WHAT I HOPE YOU TAKE AWAY

Overview of how CNTK works

Ability to setup and train a convolutional neural network

Enough info to be “dangerous” i.e., you can setup your own NN with CNTK and know where to go to learn more

THE BIG BANG IN MACHINE LEARNING



"Google's AI engine also reflects how the world of computer hardware is changing. (It) depends on machines equipped with GPUs… And it depends on these chips more than the larger tech universe realizes."

WIRED

NVIDIA GPU THE ENGINE OF DEEP LEARNING

WATSON

CHAINER

THEANO

MATCONVNET

TENSORFLOW

CNTK

TORCH

CAFFE

NVIDIA CUDA
ACCELERATED COMPUTING PLATFORM

CNTK

WHAT IS CNTK?

Computational Network Toolkit, from Microsoft (cntk.ai). Available on GitHub.

Performance—Runs on CPUs and GPUs, including clusters.

Flexibility—CNN, RNN, LSTM, fully customizable.

Extensibility—plug-in architecture allows user-defined add-ons.

Applications—Speech recognition, Machine translation, Image recognition, Image captioning, Text processing and relevance, Language understanding, language modeling.

http://research.microsoft.com/pubs/226641/CNTK-Tutorial-NIPS2015.pdf

DESIGN GOALS

Inspiration from Legos

- Each brick is very simple and performs a specific function
- Create arbitrary objects by combining many bricks

CNTK enables the creation of existing and novel models by combining simple functions in arbitrary ways

http://research.microsoft.com/pubs/226641/CNTK-Tutorial-NIPS2015.pdf

FUNCTIONALITY

CPU and GPU with **focus on GPU cluster**

Windows and Linux

Automatic numerical differentiation

Fully modular design of computational networks, execution engine, learning algorithms, model descriptions, data readers.

Plug-n-play to design your network

http://research.microsoft.com/pubs/226641/CNTK-Tutorial-NIPS2015.pdf

CONFIG FILE

`cntk configFile=yourExp.cntk`

Basis for the entire CNTK job—no code required, just write a text file

Define variables for the NN task

All complex information embedded in blocks within config file

Main blocks are **`train, test, eval`**

Sub blocks are File Reader, Network Builder, Learning algorithm

TRAIN

https://github.com/Microsoft/CNTK/wiki/Train%2C-Test%2C-Eval

Define a network

- Define each layer e.g., convolution, activation, pooling, etc.

Specify a learning algorithm and its parameters

- SGD, epochSize, learningRatesPerMB, maxEpochs, etc.

Declare a FileReader

- Tells CNTK how to parse the training data properly

TEST

https://github.com/Microsoft/CNTK/wiki/Train%2C-Test%2C-Eval

Declare a FileReader

Tell CNTK how to parse the test data properly

Specify a learned model to use

FILE READER

https://github.com/Microsoft/CNTK/wiki/Reader-block

Defines how CNTK will parse the input data.

Various readers, we will use UCIFastReader, assumes data in UCI format

Each row is an observation and each column is a data field of some type

In the MNIST dataset each row has 784 values, one for each pixel in the image

Specify the filename, features and labels

SIMPLE NETWORK DEFINITION

https://github.com/Microsoft/CNTK/wiki/Simple-Network-Builder

Simple Network Builder for easy usage but less optimization

Specify things like layer size, layer type, training criterion etc.

NETWORK DEFINITION LANGUAGE (NDL)

https://github.com/Microsoft/CNTK/wiki/NDL-Basic-concepts

Variables

Parameters—weight matrices

Inputs—input values into the network, features and labels.

Computation—various operations performed

Top Layer—last layer in NN which gives the output

Back Propagation—specified by learning algorithm, e.g., SGD

Error Prediction—defines how errors are predicted in training

NDL SPECIAL NODES

CNTK will build your network based on the Inputs/Parameters/Computations

CNTK requires some special node definitions to finalize the NN creation

FeatureNodes--features

LabelNodes--labels

CriteriaNodes—error function to minimize for training

EvalNodes—evaluate how well the network is doing

OutputNodes—output of the NN

CNTK EXECUTION

`cntk configFile=<configfile>`

Parses input file for syntax errors

Multiple passes through the NN to verify inputs/dimensions/layers all “make sense” and create computational graph

Train: Train the NN using user-defined network. Upon completion saves the model in a file

Test: Loads the trained model and then tests the NN for accuracy

CNTK OUTPUT--TRAINING

```
Finished Epoch[ 3 of 5]: [Training Set] TrainLossPerSample =
0.15781009; EvalErrPerSample = 0.12918334;
AvgLearningRatePerSample = 0.003125; EpochTime=1.55619

Starting Epoch 4: learning rate per sample = 0.003125  effective
momentum = 0.900000  momentum as time constant = 303.7 samples
```

CNTK OUTPUT--TESTING

```
Final Results: Minibatch[1-625]: Samples Seen = 10000
EvalErrorPrediction: ErrorPrediction/Sample = 0.0967
SquareError: SquareError/Sample = 0.12366719
```

SETTING UP THE LAB

CONNECTION INSTRUCTIONS

Navigate to nvlabs.qwiklab.com

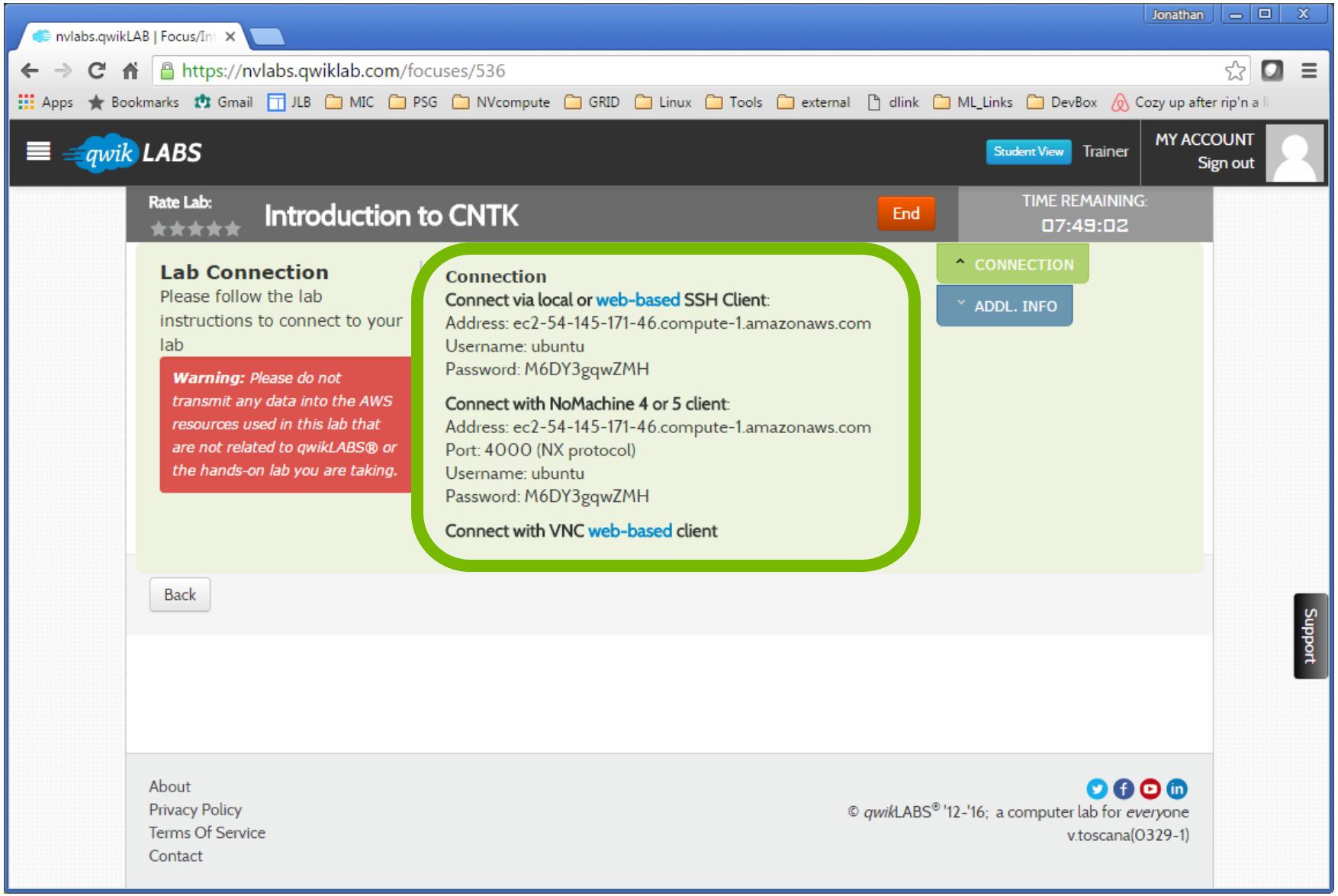
Login or create a new account

Select the “Instructor-Led Hands-on Labs” Class

Find the lab called “Introduction to CNTK”, select it, click Select, and finally click Start

After a short wait, lab instance Connection information will be shown

Please ask Lab Assistants for help!


nvlabs.qwikLAB | Focus/In
Jonathan
https://nvlabs.qwiklab.com/focuses/536
Apps Bookmarks Gmail JLB MIC PSG NVcompute GRID Linux Tools external dlink ML_Links DevBox Cozy up after rip'n a
qwik LABS
Student View
Trainer
MY ACCOUNT
Sign out
Rate Lab:
Introduction to CNTK
End
TIME REMAINING:
07:49:02
CONNECTION
ADDL. INFO
Lab Connection
Please follow the lab instructions to connect to your lab
Warning: Please do not transmit any data into the AWS resources used in this lab that are not related to qwikLABS® or the hands-on lab you are taking.
Connection
Connect via local or web-based SSH Client:
Address: ec2-54-145-171-46.compute-1.amazonaws.com
Username: ubuntu
Password: M6DY3gqwZMH
Connect with NoMachine 4 or 5 client:
Address: ec2-54-145-171-46.compute-1.amazonaws.com
Port: 4000 (NX protocol)
Username: ubuntu
Password: M6DY3gqwZMH
Connect with VNC web-based client
Back
Support
About
Privacy Policy
Terms Of Service
Contact
© qwikLABS® '12-'16; a computer lab for everyone
v.toscana(0329-1)

SAVING WORK (SSH)

You can scp files (Linux/Mac/Cygwin) using the following command and the generated password:

```
scp ubuntu@<ec2-address>:/file/location .
```

Or use Winscp from here: www.winscp.net

HANDWRITTEN DIGIT RECOGNITION

HELLO WORLD of machine learning?

MNIST data set of handwritten digits from Yann Lecun's website

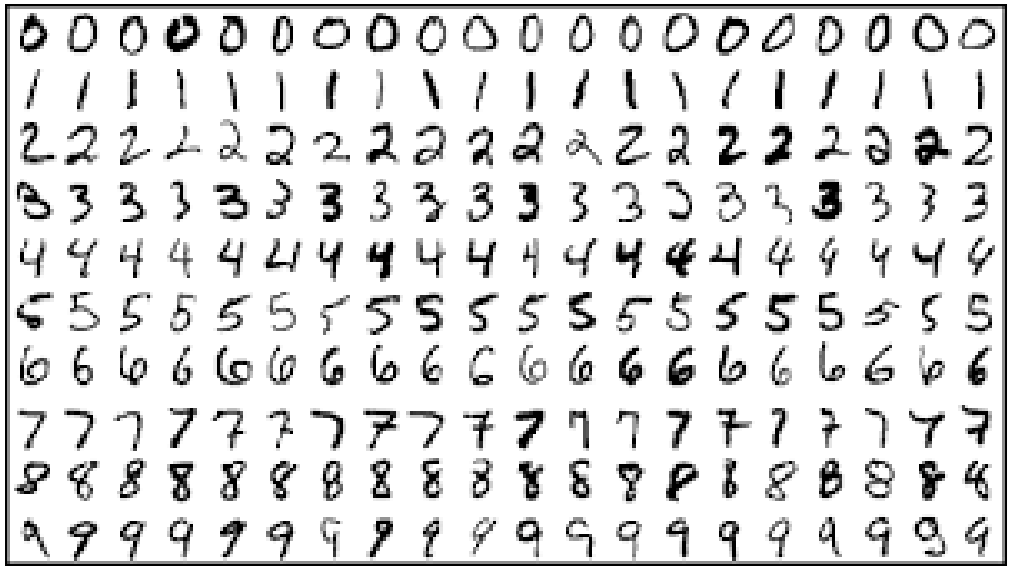
All images are 28x28 grayscale

Pixel values from 0 to 255

60k training examples, 10k test examples

Input vector of size 784

Output value is integer from 0-9



SOFTWARE FOR LAB

Remote Desktop Software: Download NoMachine **now** for best performance from www.nomachine.com/download

- Alternatively you may use a VNC client or the provided browser-based VNC option

SSH Access Software (optional): PuTTy for Windows can be downloaded from www.putty.org

- Alternatively you may use a provided browser-based SSH option

SETUP

Files are located at `gpu-edu-workshops/exercises/cntk`

Solutions are at `gpu-edu-workshops/exercise_solutions/cntk`

In the .cntk files you'll see "**FIXME**" appears in various places.

This is where you need to change the files according to the comments.

https://github.com/Microsoft/CNTK/wiki

http://www.cntk.ai/

Handouts that we provide

ONE HIDDEN LAYER SIMPLE

SIMPLE NETWORK BUILDER

Task 1

CNTK has pre-built defaults for creating simple networks

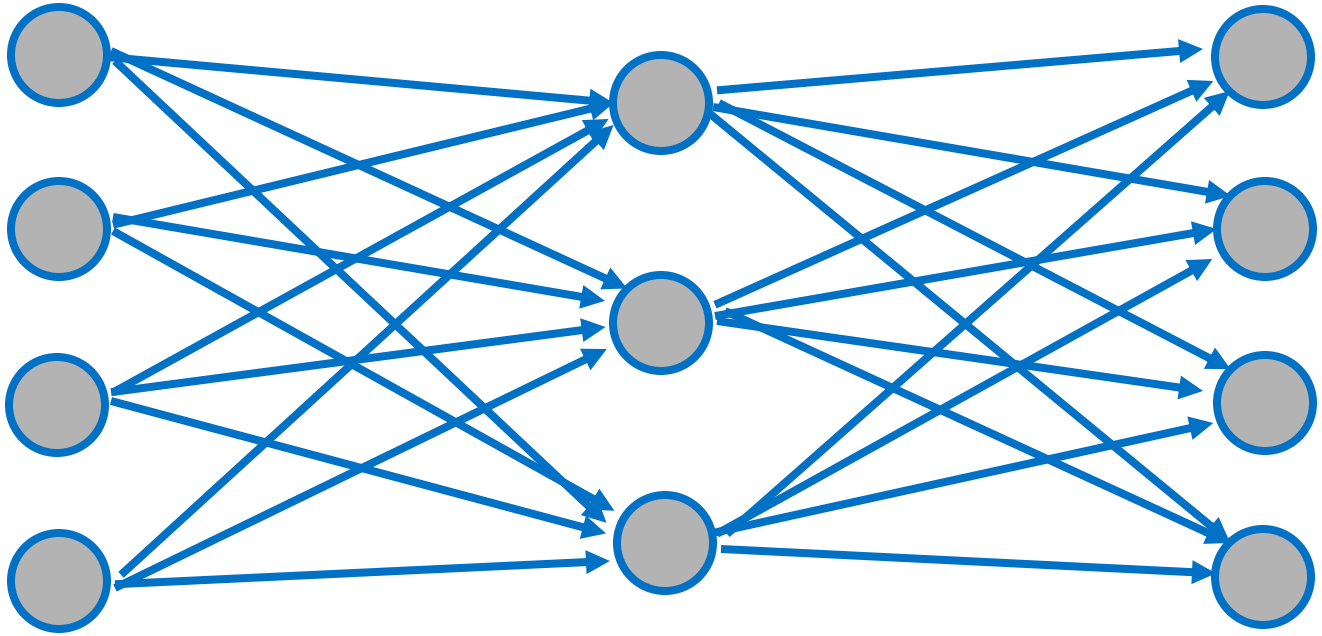
User supplies the activation type, layer sizes, training criteria and eval criteria

Typically use SGD

User supplies the Reader type

NEURAL NETWORK

One hidden layer only



Input layer
784 pixels

hidden layer
25

output layer
10 (0-9)

TASK 1

./runCNTK.sh

gpu-edu-workshops/exercises/cntk/simpleMNIST/simpleMNIST.cntk

Create a NN with one hidden layer by completing the .cntk file and then run the training and testing. Change “FIXME” to proper CNTK commands.

Experiment with activations, minimization function, epochs, learning rates, etc.

What’s the best accuracy you can achieve?

Add “topoplot” to your top-level command to generate a network graph

```
/usr/bin/dot –Tjpg model.dot –omodel.jpg

eog model.jpg
```

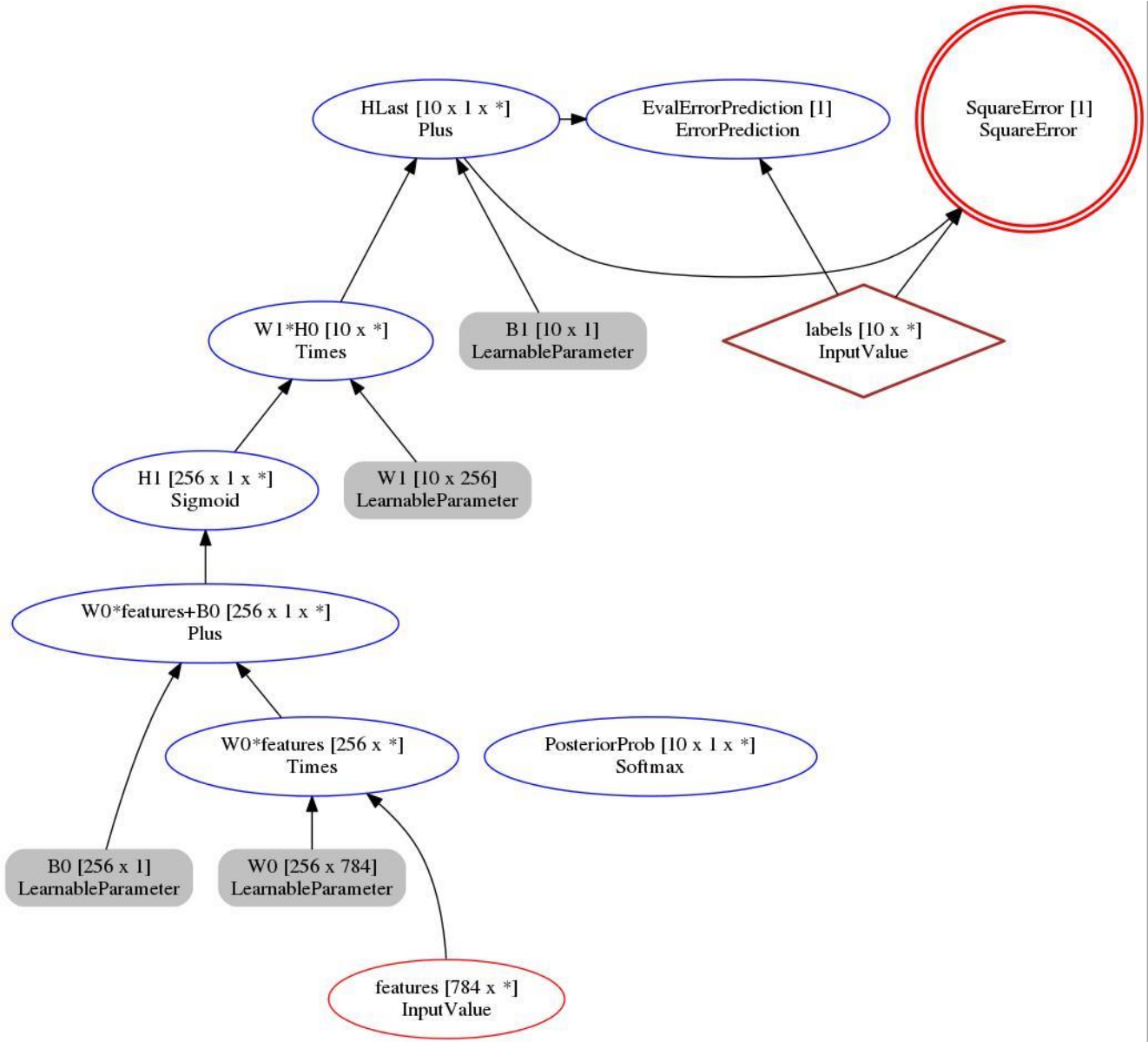
SOLUTION

```
layerSizes = 784:256:10

layerTypes = Sigmoid

trainingCriterion = SquareError

maxEpochs = 30
```



RESULTS

Square Error minimization

Sigmoid Activation

Bias terms included automatically by Simple Network Builder

5 epochs

```
Final Results: Minibatch[1-625]: Samples Seen = 10000
EvalErrorPrediction: ErrorPrediction/Sample = 0.0967
SquareError: SquareError/Sample = 0.12366719
```

ONE HIDDEN LAYER NDL

TASK 2

`gpu-edu-workshops/exercises/cntk/nnMNIST/nnMNIST.cntk`

Create a NN with one hidden layer.

Need to use NDL now, more parameters to define but more flexibility too.

What's the best accuracy you can achieve? Feel free to adjust parameters.

What happens if you remove the second layer of activations?

Add "topoplot" to your top-level command to generate a network graph then plot

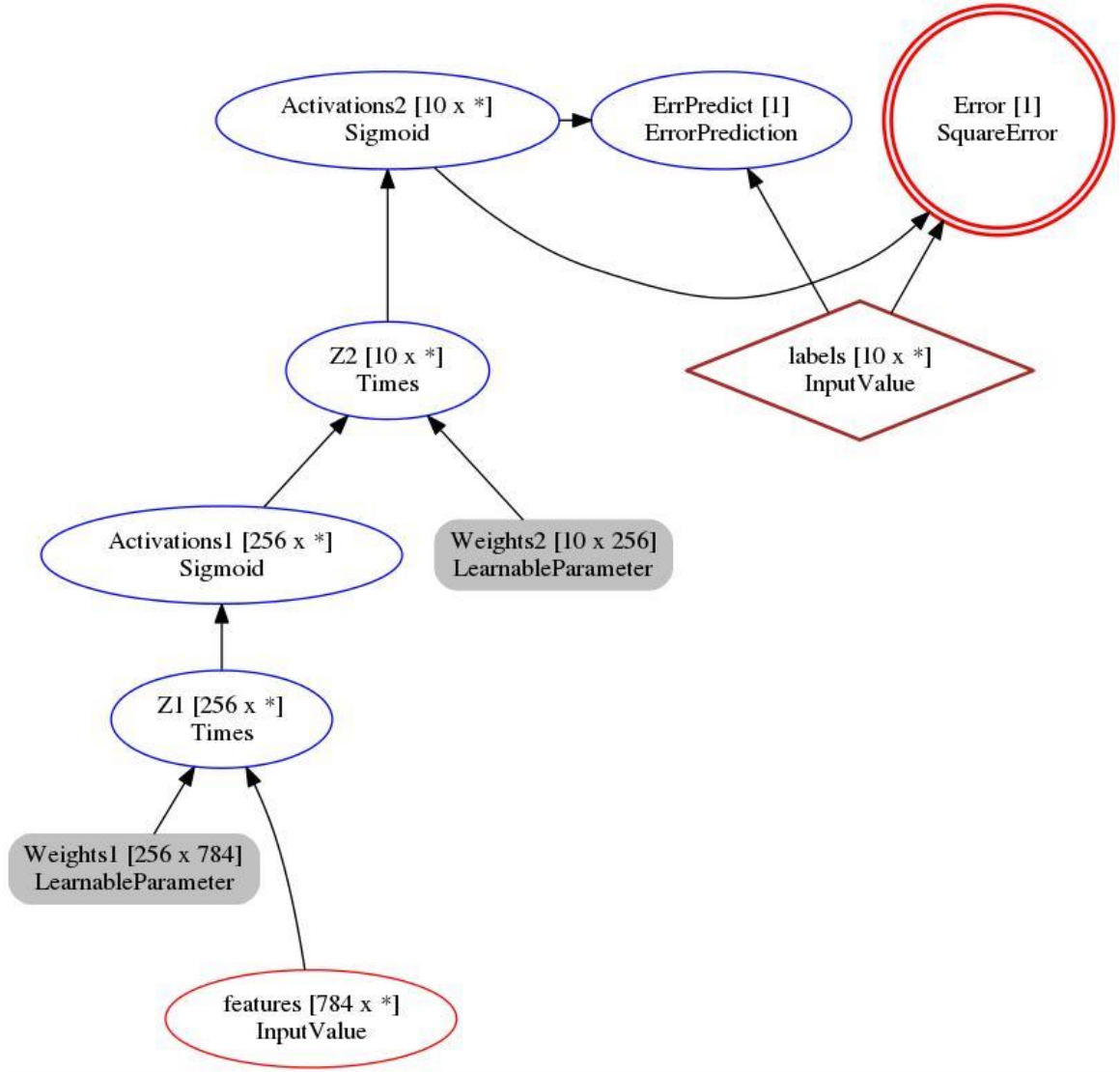
SOLUTION

```
InputDim = 784
HiddenDim = 256
OutputDim = 10

Weights1 = LearnableParameter(HiddenDim, InputDim)
Activations1 = Sigmoid( Z1 )

Weights2 = LearnableParameter(OutputDim, HiddenDim)
Activations2 = Sigmoid( Z2 )

minibatchSize = 32
maxEpochs = 5
```



RESULTS

Hidden dimension 256

Sigmoid Activations

No Bias terms

```
5 epochs

Final Results: Minibatch[1-625]: Samples Seen = 10000
ErrPredict: ErrorPrediction/Sample = 0.0662    Error:
SquareError/Sample = 0.063384753
```

TASK 3

`exercises/cntk/nnMNISTBias/nnMNISTBias.cntk`

Add bias and normalization factors to the one layer NN

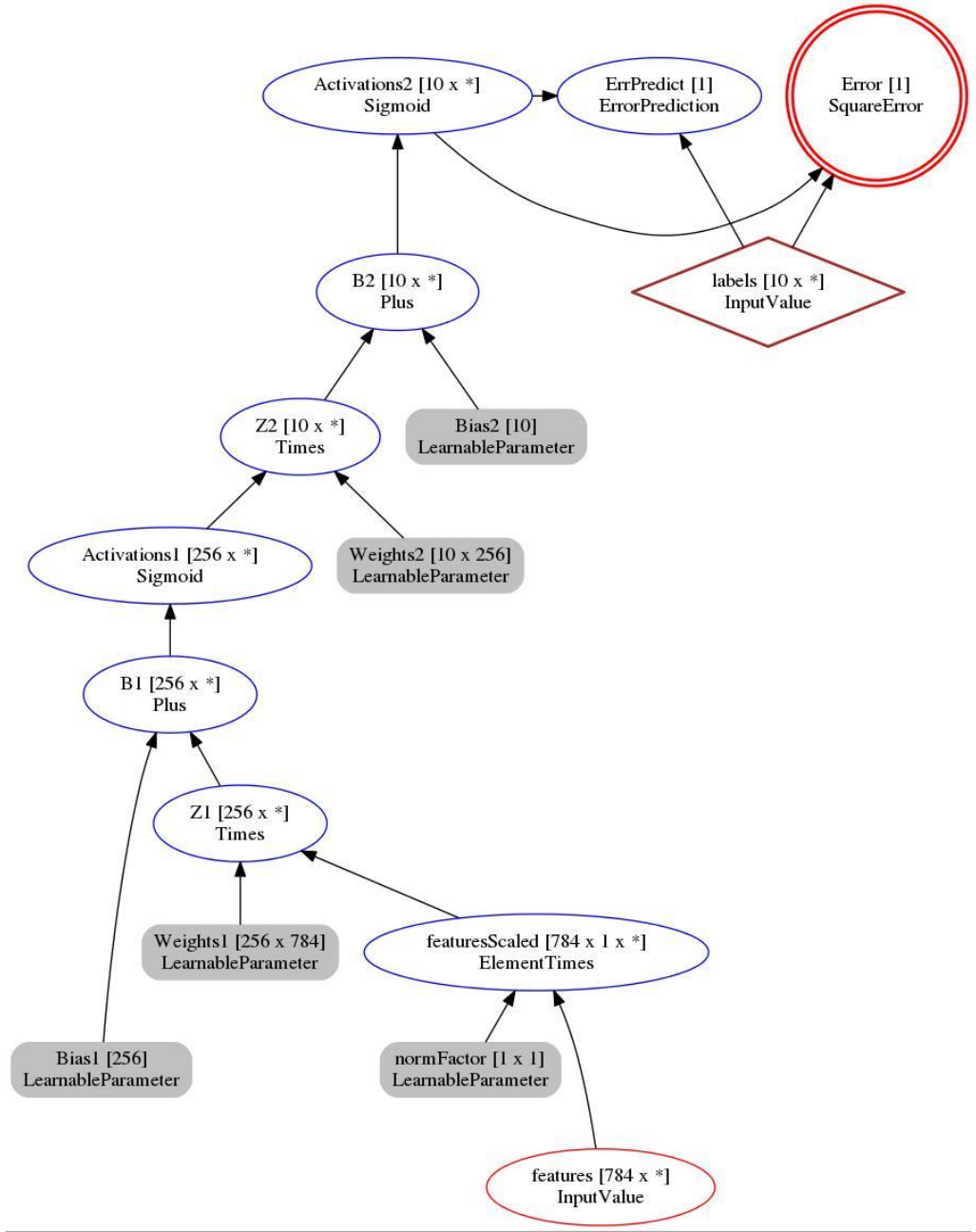
What's the best accuracy you can achieve? Feel free to adjust parameters.

Plot the network again.

What happens when adding more epochs?

SOLUTION

```
InputDim = 784
HiddenDim = 256
OutputDim = 10
features = InputValue( InputDim )
labels = InputValue( OutputDim )
Weights1 = LearnableParameter( HiddenDim, InputDim )
Bias1 = LearnableParameter( HiddenDim )
Activations1 = Sigmoid( B1 )
Weights2 = LearnableParameter( OutputDim, HiddenDim )
Bias2 = LearnableParameter( OutputDim )
Activations2 = Sigmoid( B2 )
Error = SquareError( labels, Activations2 )
minibatchSize = 32
learningRatesPerMB = 0.1
maxEpochs = 30
```



RESULTS

Hidden dimension 256, Sigmoid, bias and normalization terms

```
5 epochs

Final Results: Minibatch[1-625]: Samples Seen = 10000
ErrPredict: ErrorPrediction/Sample = 0.0991    Error:
SquareError/Sample = 0.096377637

30 epochs

Final Results: Minibatch[1-625]: Samples Seen = 10000
ErrPredict: ErrorPrediction/Sample = 0.06    Error:
SquareError/Sample = 0.052395992
```

RESULTS

Hidden dimension 256, bias and normalization terms

Error criterion changed to CrossEntropyWithSoftmax and activations to ReLU

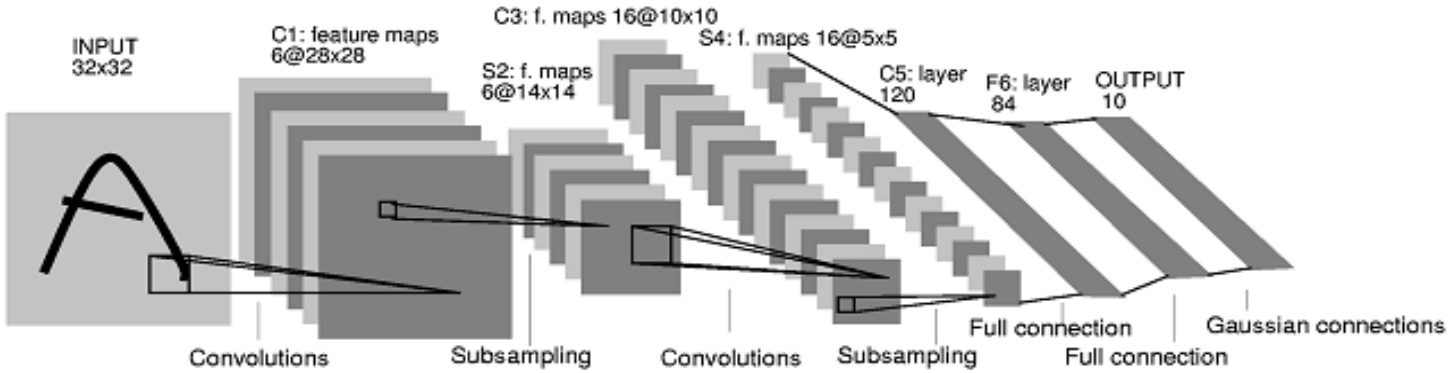
```
30 epochs

Final Results: Minibatch[1-625]: Samples Seen = 10000
ErrPredict: ErrorPrediction/Sample = 0.0179    Error:
CrossEntropyWithSoftmax/Sample = 0.064061735    Perplexity =
1.0661582
```

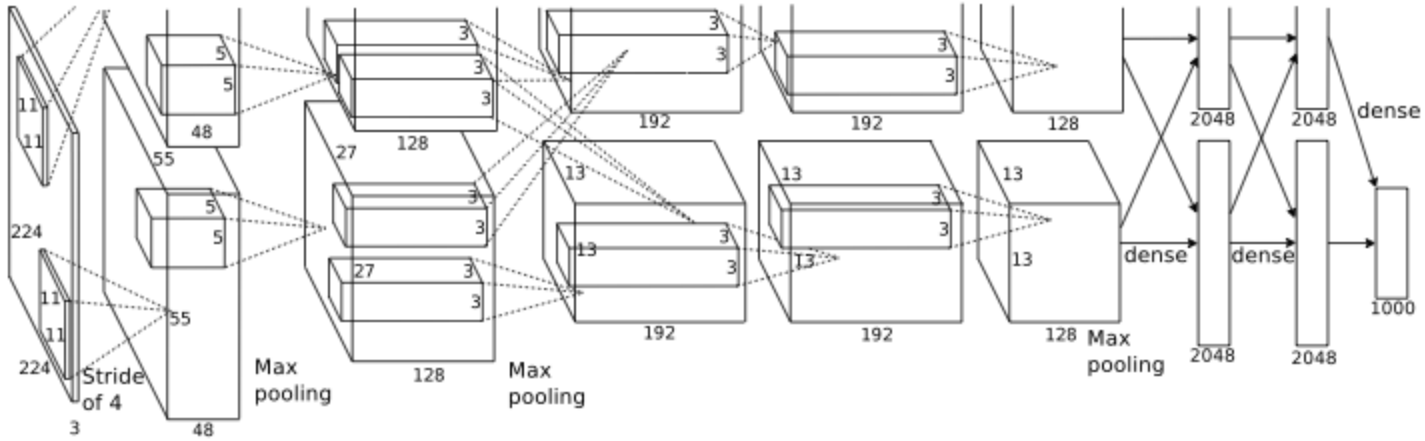
Significant improvement using ReLU and CrossEntropyWithSoftmax!!!

CONVOLUTIONAL NEURAL NETWORK

CNN BREAKTHROUGHS



LeCun et al. 1989-1998: Handwritten Digit Recognition



Krizhevsky, Hinton et al., 2012: ImageNet Winner

TWO NEW TYPES OF LAYERS

Convolution layers

- Previous examples focused on each pixel.
- What if features encompass multiple pixels?
- Can use convolutions to capture larger receptive fields

Pooling layers

- Essentially a down-sampling method retaining information while eliminating some computational complexity

TASK 4

`exercises/cntk/cnnMNIST/cnnMNIST.cntk`

Finish the CNN, a variant of LeNet

Convolution1, 5x5 kernel, stride 1

Maxpooling1, 2x2 window, stride 2

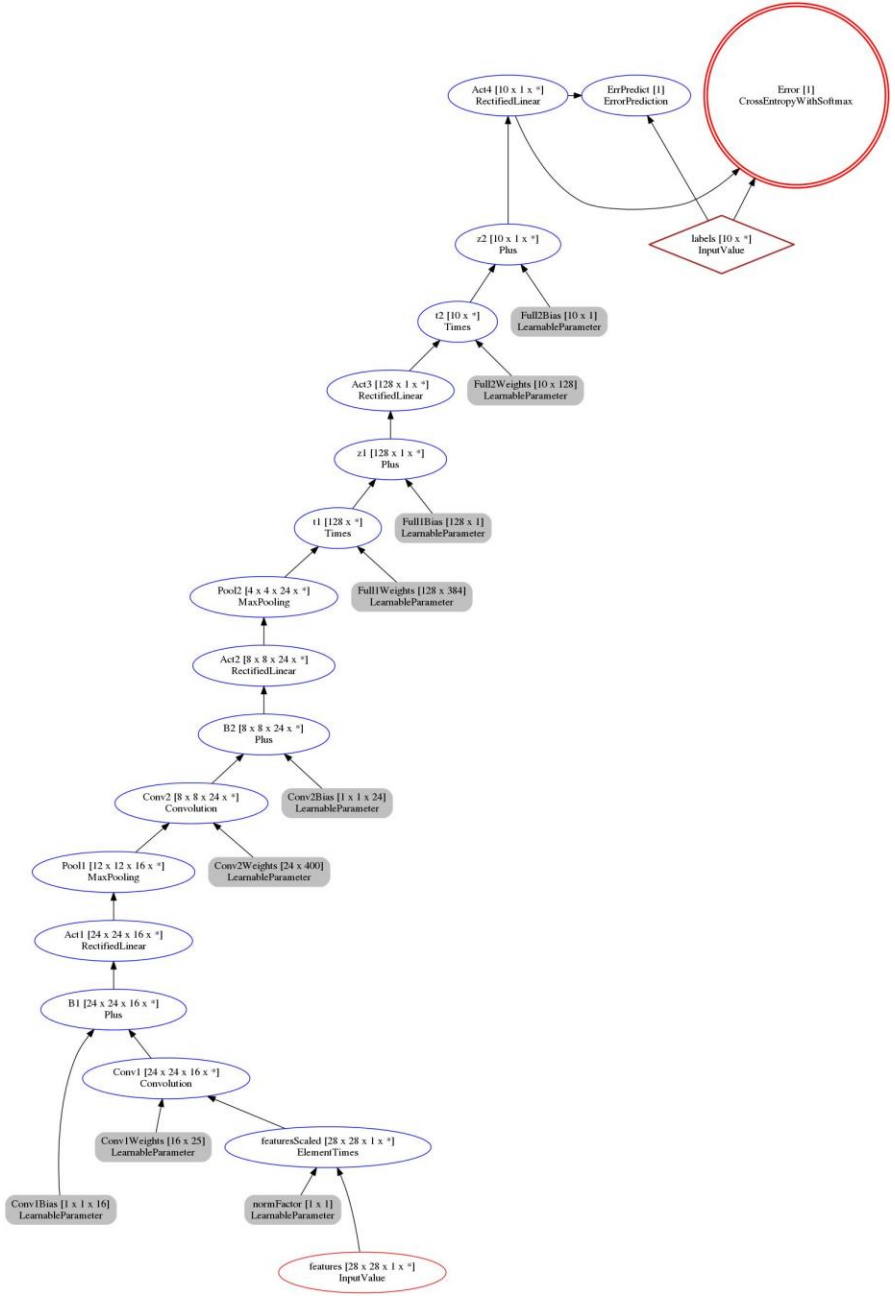
Convolution2, 5x5 kernel, stride 1

Maxpooling2, 2x2 window, stride 2

Activation and fully connected layers as well

SOLUTION

```
imageWidth = 28
imageHeight = 28
kernelWidth = 5
kernelHeight = 5
Conv1 = Convolution( Conv1Weights, featuresScaled, kernelWidth,
  kernelHeight, numFeatureMaps, 1, 1, zeroPadding=false,
  imageLayout=$imageLayout$ )
Act1 = RectifiedLinear( B1 )
Pool1 = MaxPooling( Act1, 2, 2, 2, 2, imageLayout=$imageLayout$ )
Conv2 = Convolution( Conv2Weights, Pool1, kernelWidth, kernelHeight,
  numFeatureMaps2, 1, 1, zeroPadding=false, imageLayout=$imageLayout$)
Act2 = RectifiedLinear( B2 )
Pool2 = MaxPooling( Act2, 2, 2, 2, 2, imageLayout=$imageLayout$ )
Act3 = RectifiedLinear( z1 )
Act4 = RectifiedLinear( z2 )
```



RESULTS

ReLU, Bias and normalization terms

```
5 epochs

Final Results: Minibatch[1-625]: Samples Seen = 10000
ErrPredict: ErrorPrediction/Sample = 0.0998    Error:
CrossEntropyWithSoftmax/Sample = 0.24861992

30 epochs

Final Results: Minibatch[1-625]: Samples Seen = 10000
ErrPredict: ErrorPrediction/Sample = 0.0072    Error:
CrossEntropyWithSoftmax/Sample = 0.034182469
```

LAB SUMMARY

CNTK provides framework for flexible, user-defined networks.

Intro for how to use CNTK to design networks and run training/testing jobs.

Never even discussed RNN, LSTM, running parallel on GPU clusters, etc!

MORE INFO

jbentz@nvidia.com with questions

www.cntk.ai

https://github.com/Microsoft/CNTK

https://developer.nvidia.com/deep-learning

S6839 - Leveraging Microsoft Azure's GPU N-Series for Compute Workloads and Visualization, 04/06, 16:00-16:25, Room LL21B

S6843 - Deep Learning in Microsoft with CNTK, 04/06, 16:30-16:55, Hall 3

FURTHER HANDS-ON TRAINING

Check out the Self-Paced labs at the conference.

Deep Learning, CUDA, OpenACC, Tools and more!

Just grab a seat and take any available lab

Located in the lower-level outside of LL20C

You will also receive Credits to take additional labs at nvidia.qwiklab.com

Log in using the same account you used in this lab

GPU TECHNOLOGY CONFERENCE

April 4-7, 2016 | Silicon Valley

THANK YOU

JOIN THE CONVERSATION

#GTC16

PRESENTED BY

NVIDIA.