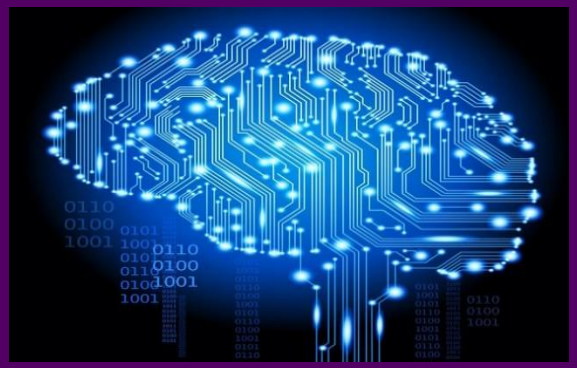
Deep Learning
MSiA 432



Theory and Applications

NORTHWESTERN
UNIVERSITY

COMPUTING RESOURCES

NORTHWESTERN UNIVERSITY

Overview

- GPU Cluster (Deepdish)
- Jupyter server
- Intel Devcloud

NORTHWESTERN UNIVERSITY

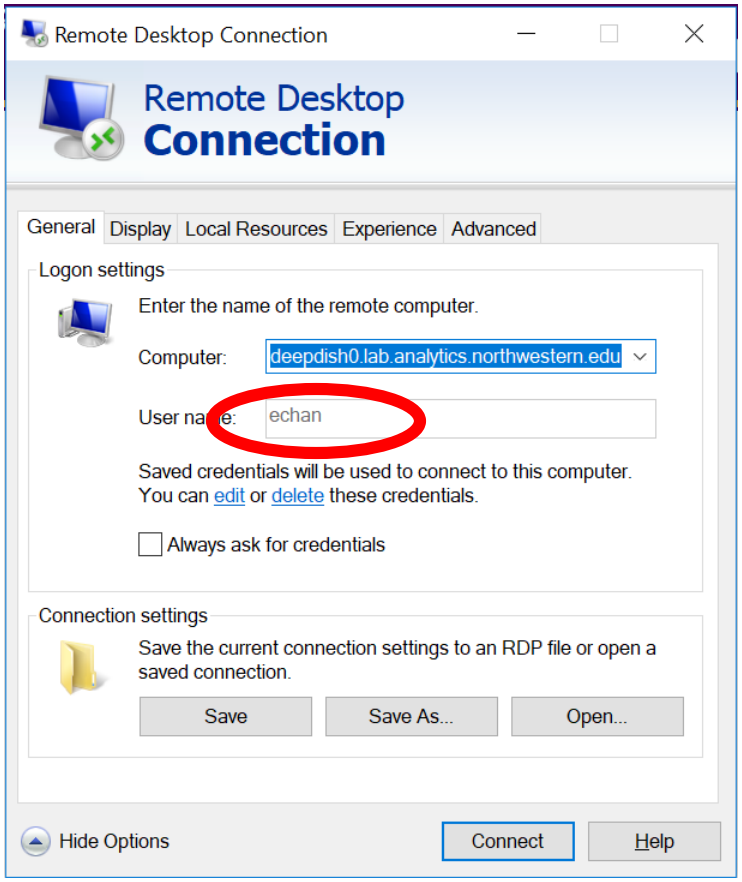
Deepdish GPU cluster

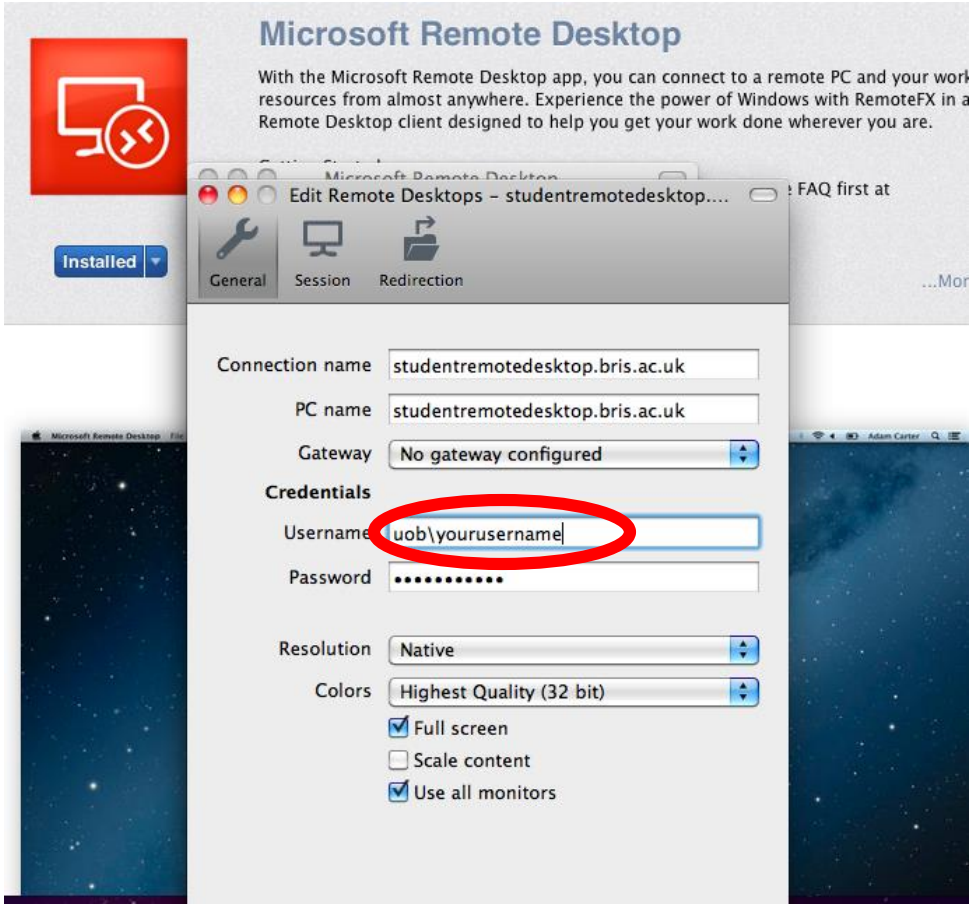
- 2 servers: deepdish0.lab.analytics.northwestern.edu, deepdish1.lab.analytics.northwestern.edu
- Each server has:
  - 4 NVIDIA GeForce 1080 GPUs
  - 128 GB memory
  - Intel i7 CPU with 6 cores (12 threads)
- Accessible via RDP
  - Windows – mstsc (remote desktop connection)
  - Mac – Microsoft remote desktop client
  - Linux – rdesktop client
- Also available via ssh

NORTHWESTERN UNIVERSITY

Accessing via RDP

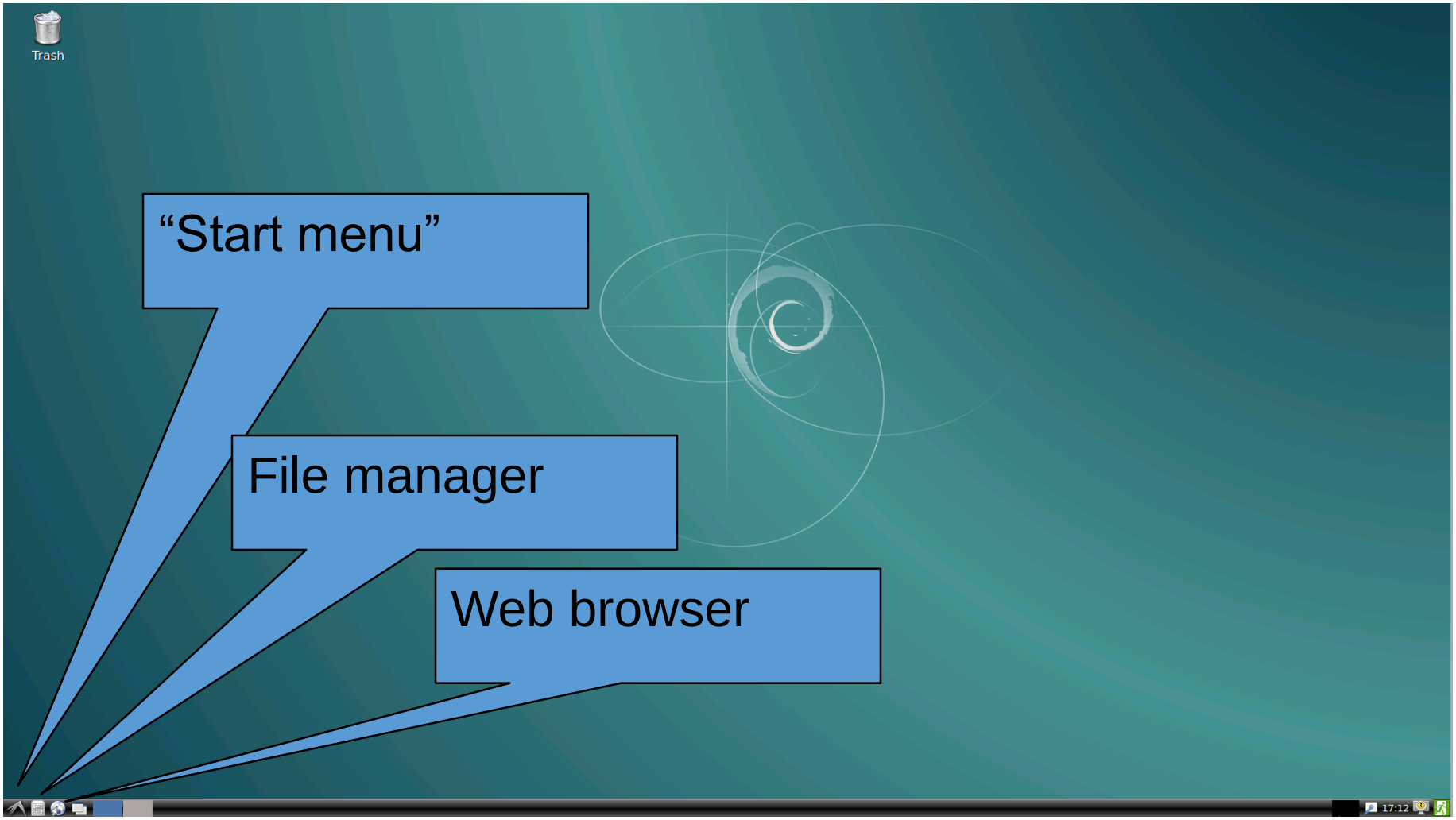
- Be sure to provide the credentials at the time of login





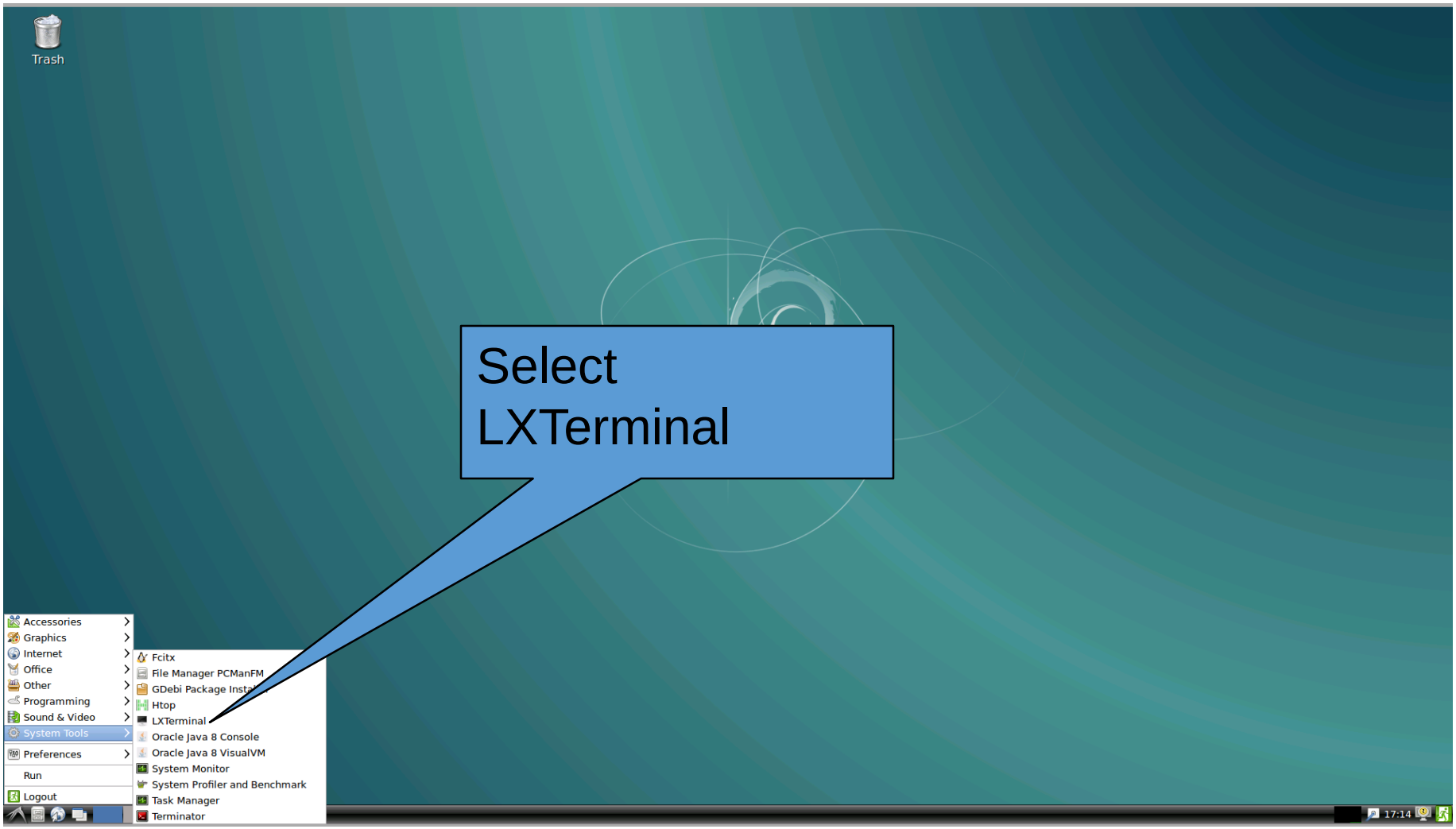
NORTHWESTERN UNIVERSITY

Desktop interface



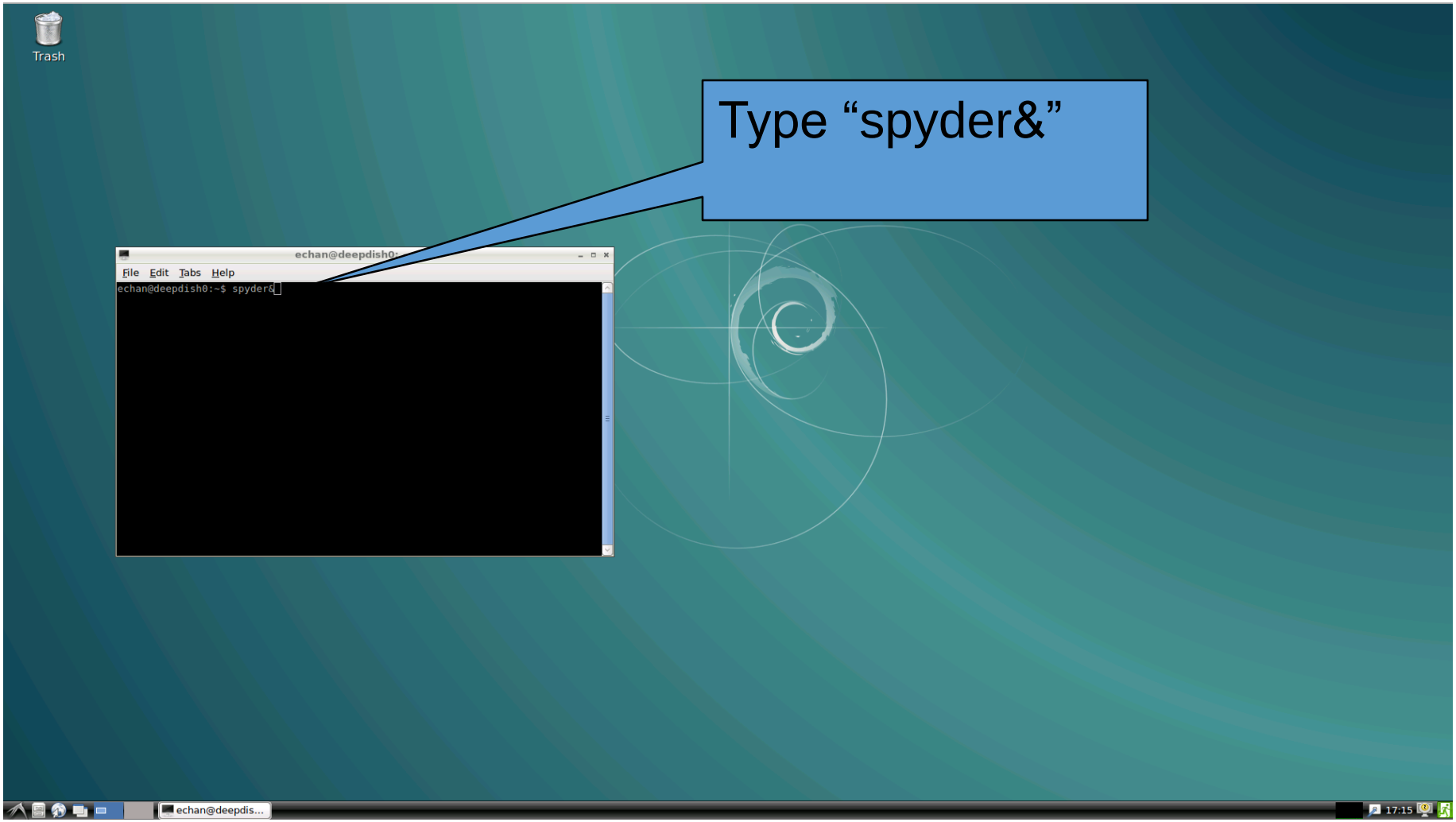
NORTHWESTERN UNIVERSITY

Starting a terminal



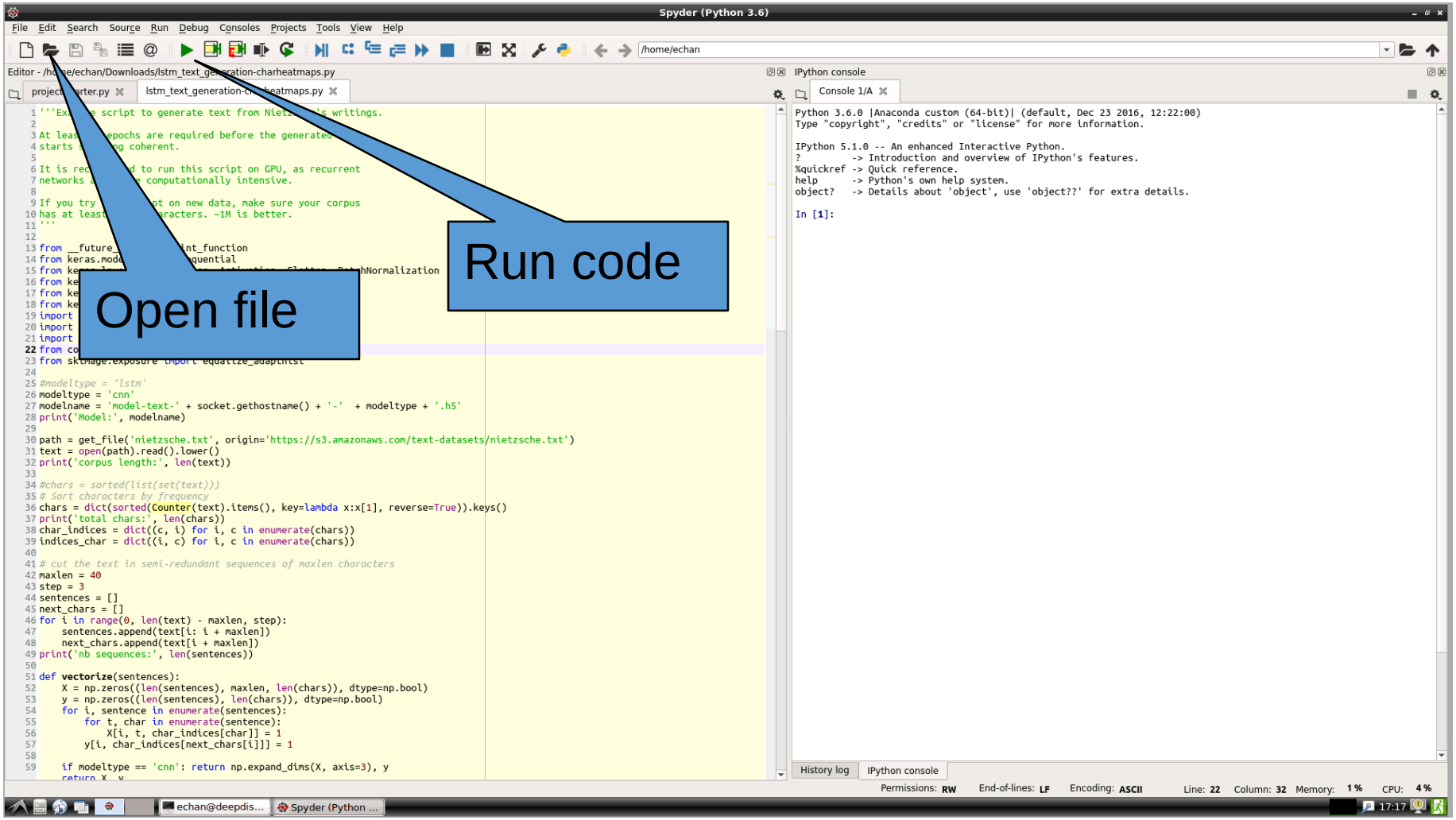
NORTHWESTERN UNIVERSITY

Starting Spyder



NORTHWESTERN UNIVERSITY

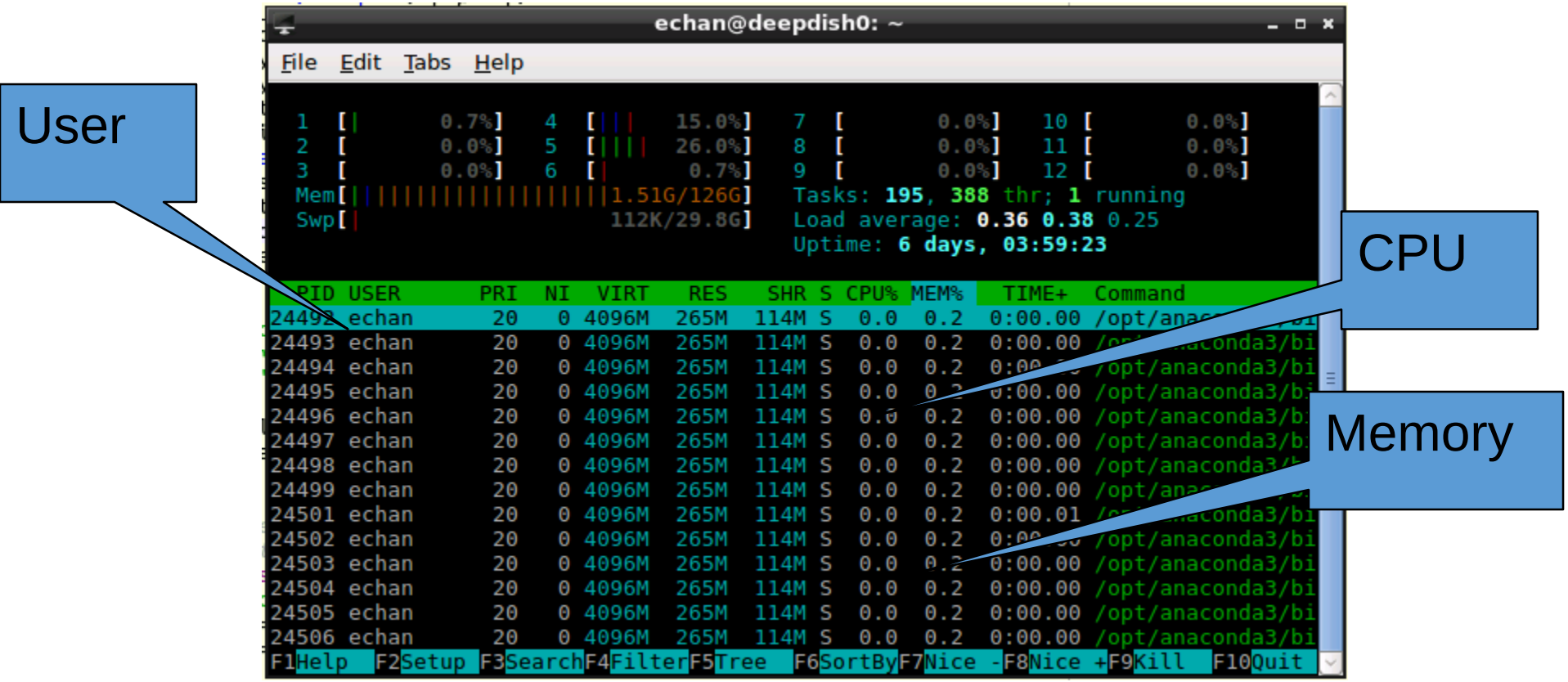
Using Spyder



NORTHWESTERN UNIVERSITY

Monitoring system resources

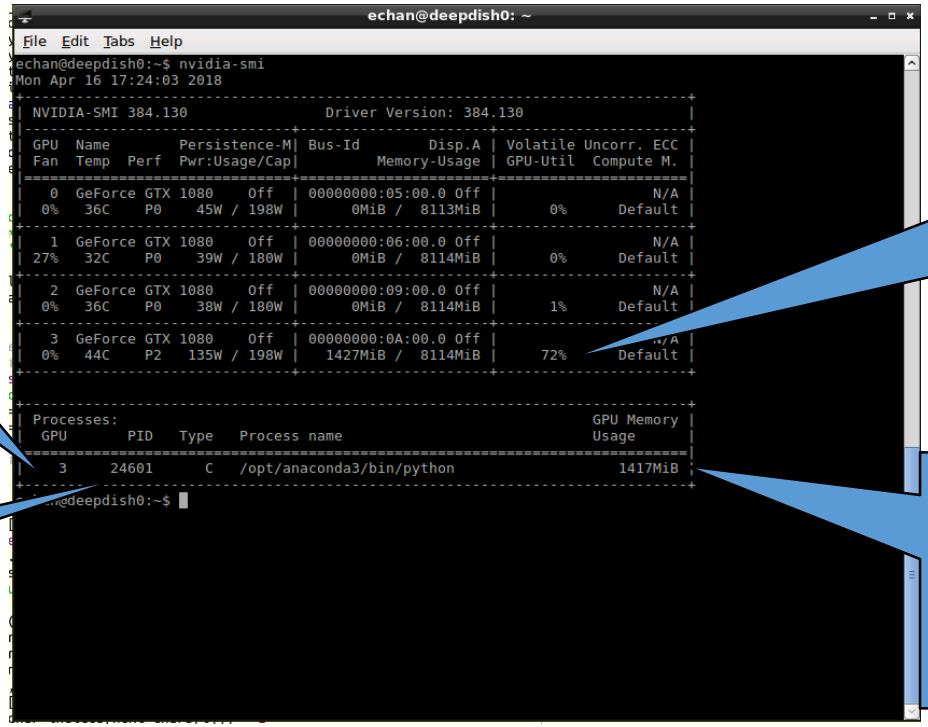
- System resources are scarce. Be “nice” to other groups.
- CPU/memory - htop



NORTHWESTERN UNIVERSITY

GPU

- Each user is assigned a single GPU at login per session. Please be courteous and only run a single job at a time to ensure fair allocation. We occasionally catch resource hogs and terminate their jobs.
- If a job fails, try the other server, but please terminate your current job.
- Tool to check usage is nvidia-smi



Using GPU 3

GPU currently 72% busy

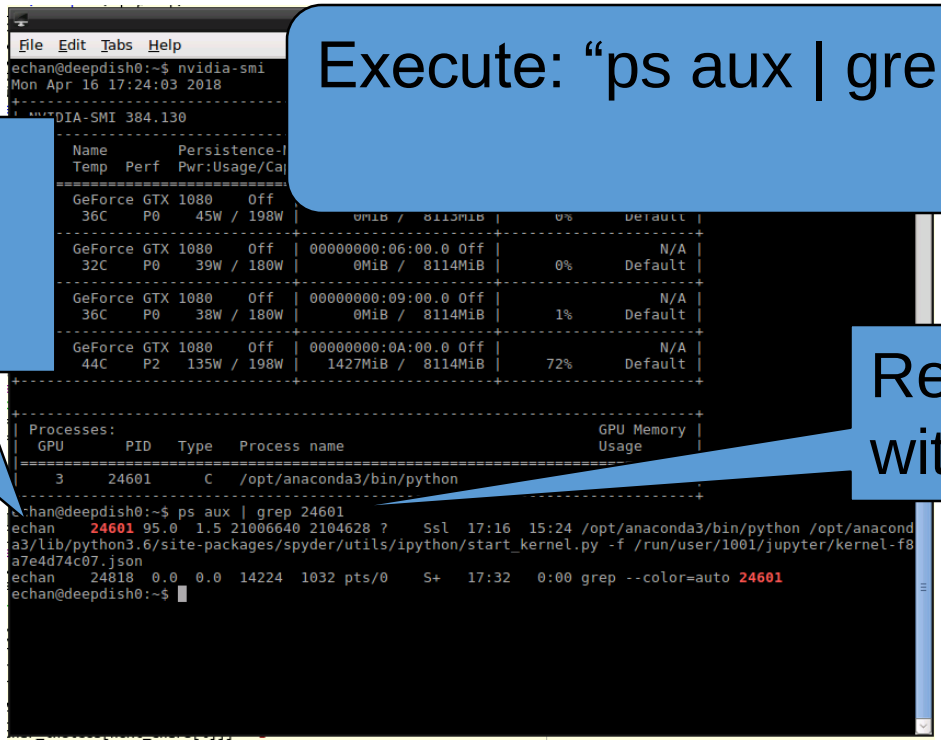
PID is 24601

1417/8114 MB memory used

NORTHWESTERN UNIVERSITY

Finding GPU hogs

- NVIDIA-SMI doesn't provide a direct way to map PID (process ID) to user
- Need to use ps aux to figure out the owner of the job



Execute: "ps aux | grep <jobid>"

Owner is listed here. Echan in this case.

Replace 24601 with the job #

NORTHWESTERN UNIVERSITY

GPU rules of the road

- Please have only one job per user execute on the cluster at a time. GPU resources are limited.
- If a user accidentally runs too many jobs, please let them know they have runaway jobs and ask them to terminate the excess jobs.
- Tensorflow by default pre-allocates all GPU memory, and this causes problems with other users. Please apply the fix described on the next slide.

NORTHWESTERN UNIVERSITY

Tensorflow GPU preallocation fix

- Tensorflow by default pre-allocates all GPU memory. This can cause other jobs to fail to launch.
- https://stackoverflow.com/questions/34199233/how-to-prevent-tensorflow-from-allocating-the-totality-of-a-gpu-memory
- The preferred fix is to paste the following code at the top of your Python file:

```
#%% GPU memory fix
import tensorflow as tf, keras.backend.tensorflow_backend as ktf
def get_session(gpu_fraction=0.4):
            gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=gpu_fraction, allow_growth=True)
            return tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))
ktf.set_session(get_session())
```

NORTHWESTERN UNIVERSITY

Jupyter notebooks

- Jupyter notebooks have been tested on:
  - https://app1.lab.analytics.northwestern.edu:8000
  - Intel AI Devcloud: https://software.intel.com/en-us/ai-academy/tools/devcloud
- Sometimes the latest packages are not installed by default, you can install them with “pip install --user <package>”. This installs the package in your home folder.
- Generally, you’ll need: tensorflow, keras, tqdm, seaborn

NORTHWESTERN UNIVERSITY

Installing local versions of packages

jupyter MSiA 432 Last Checkpoint: a minute ago (unsaved changes)

Control Panel | Logout

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python [conda env:anaconda3]

Code

```
In [16]: !pip install --upgrade --user keras seaborn tensorflow h5py tqdm

Requirement already up-to-date: keras in ./.local/lib/python3.5/site-packages
Requirement already up-to-date: seaborn in /opt/anaconda3/lib/python3.5/site-packages
Requirement already up-to-date: tensorflow in ./.local/lib/python3.5/site-packages
Requirement already up-to-date: h5py in ./.local/lib/python3.5/site-packages
Collecting tqdm
  Downloading https://file[illegible]org/packages/78/bc/de067ab2d700b91717dc5459d86a1877e2df31abfb90ab01a5a5a5ce30b4/tqdm
-4.23.0-py2.py3-none-any.whl ([illegible]
    100% |[illegible]/s eta 0:00:01
Requirement already up-to-date: six>=1[illegible]lib/python3.5/site-packages (from keras)
Requirement already up-to-date: scipy>=0.[illegible]5/site-packages (from keras)
Requirement already up-to[illegible]
Requirement already up-to[illegible]
Requirement already up-to[illegible]
Requirement already up-to[illegible]
Requirement already up-to[illegible]
Requirement already up-to[illegible]
Requirement already up-to[illegible]
Requirement already up-to[illegible]
Requirement already up-to[illegible]
Requirement already up-to[illegible]
Requirement already up-to[illegible]
Requirement already up-to[illegible]
tensorflow)
Requirement already up-to[illegible]
nsorflow)
Requirement already up-to[illegible]
orflow)
Requirement already up-to[illegible]
7.0->tensorflow)
Installing collected pack[illegible]
Successfully installed tq[illegible]
```

Optional local package install. ONLY DO THIS IF YOU ENCOUNTER PACKAGE ISSUES. System should already have working versions of packages installed.

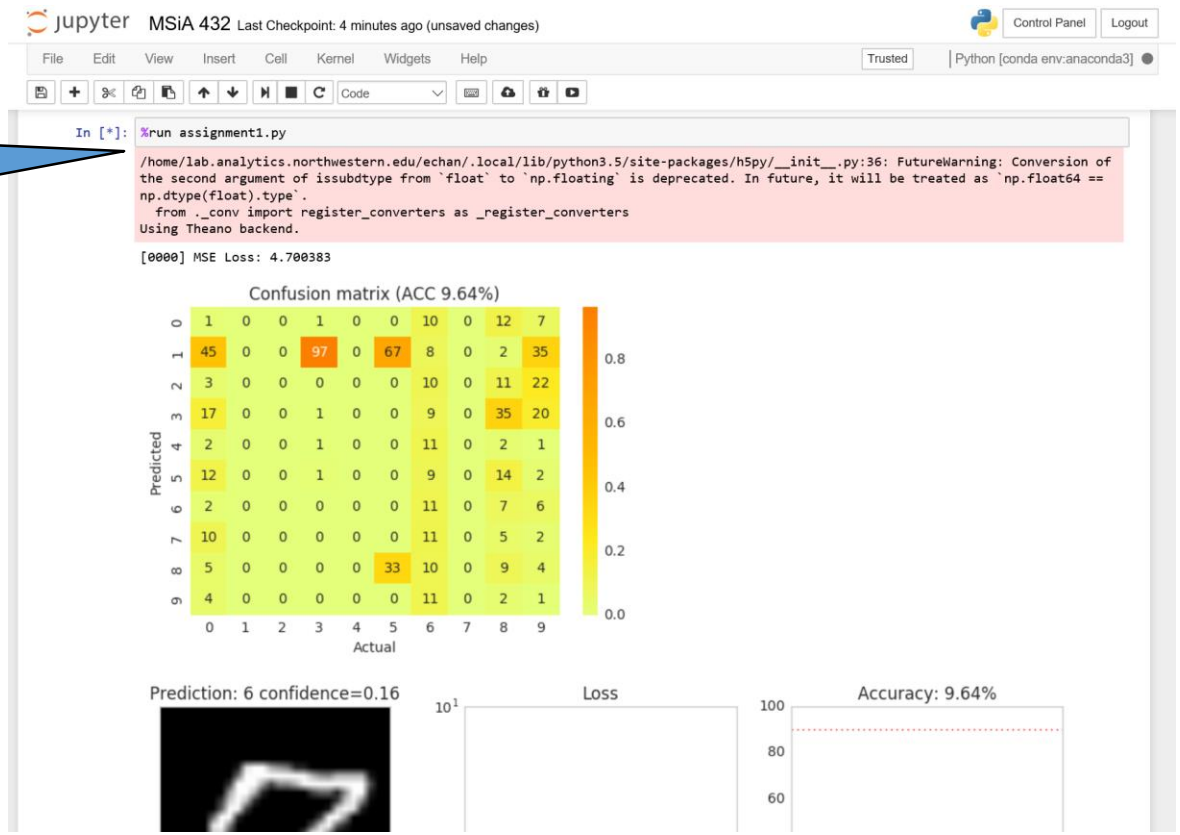
NB: Intel Devcloud needs this step

NORTHWESTERN UNIVERSITY

Running files

- Drag and drop/upload Python code to notebook
- Use “%run” magic to execute

%run <file>



NORTHWESTERN UNIVERSITY

Loading files

- Either copy/paste file contents or use “%load”

%load <file>

jupyter MSiA 432 Last Checkpoint: 5 minutes ago (unsaved changes)

Control Panel | Logout

File | Edit | View | Insert | Cell | Kernel | Widgets | Help

Trusted | Python [conda env:anaconda3]

Code

In [ ]:
```
%load assignment1.py
```

In [ ]:
```
# %load assignment1.py
# Homework 1 starter code: Simple neural network
import keras.datasets.mnist, numpy as np, os, matplotlib.pyplot as plt, matplotlib.cm as cm, seaborn, sklearn.manifold
os.makedirs('train', exist_ok=True)
(X, Y), (_, _) = keras.datasets.mnist.load_data()
#Y = np.random.randint(0, 9, size=Y.shape) # Uncomment this line for random labels extra credit
X = X.astype('float32').reshape((len(X), -1)).T / 255.0            # 784 x 60000
T = np.zeros((len(Y), 10), dtype='float32').T                      # 10 x 60000
for i in range(len(Y)): T[Y[i], i] = 1

#%% Setup: 784 -> 256 -> 128 -> 10
W1 = 2*np.random.rand(784, 256).astype('float32').T - 1
W2 = 2*np.random.rand(256, 128).astype('float32').T - 1
W3 = 2*np.random.rand(128,  10).astype('float32').T - 1

lr = 1e-5              # Learning rate, decrease if optimization isn't working
def sigmoid(x): return 1.0/(1.0 + np.e**-x)
losses, accuracies, hw1, hw2, hw3, ma = [], [], [], [], [], []
for i in range(1000): # Do not change this, we will compare performance at 1000 epochs
    # Forward pass
    L1 = sigmoid(W1.dot(X))
    L2 = sigmoid(W2.dot(L1))
    L3 = sigmoid(W3.dot(L2))
    # Backward pass
    dW3 = (L3 - T) * L3*(1 - L3)
    dW2 = W3.T.dot(dW3)*(L2*(1-L2))
    dW1 = W2.T.dot(dW2)*(L1*(1-L1))
    # Update
    W3 -= lr*np.dot(dW3, L2.T)
    W2 -= lr*np.dot(dW2, L1.T)
    W1 -= lr*np.dot(dW1, X.T)

    loss = np.sum((L3 - T)**2)/len(T.T)
    print("[%04d] MSE Loss: %0.6f" % (i, loss))
```

NORTHWESTERN UNIVERSITY

Long-running jobs

- Jupyter wasn't really built for long-running jobs
- YMMV, but it seems to work OK for jobs that run a few hours. Sometimes the jobs time out.
- Use Spyder for longer-running jobs
- You have been warned…