TensorFlow

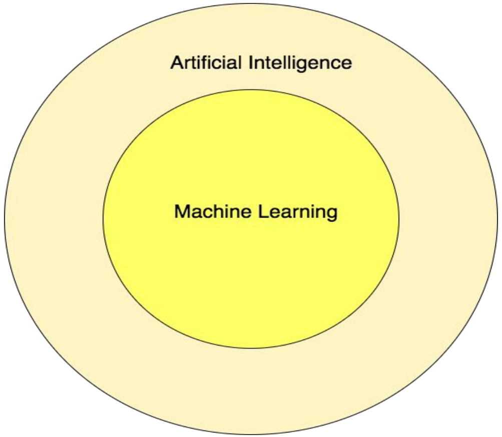
Agenda

- Introduction
- Machine Learning
- Artificial Neural Networks
- TensorFlow Basics
- Image Recognition Example
- Conclusion

Introduction

- Open source library by Google for Machine Learning
- Successor to DistBelief made by Google Brain
- Scalability and portability core feature.
- Main uses are pattern recognition using Artificial Neural Networks
- Large community with wide range of applications internally and outside Google

Machine Learning (ML)



- Application of Artificial Intelligence (AI)
  AI - Broader concept, Machines perform “intelligent” tasks
  ML - Give machines data and make them learn by themselves.

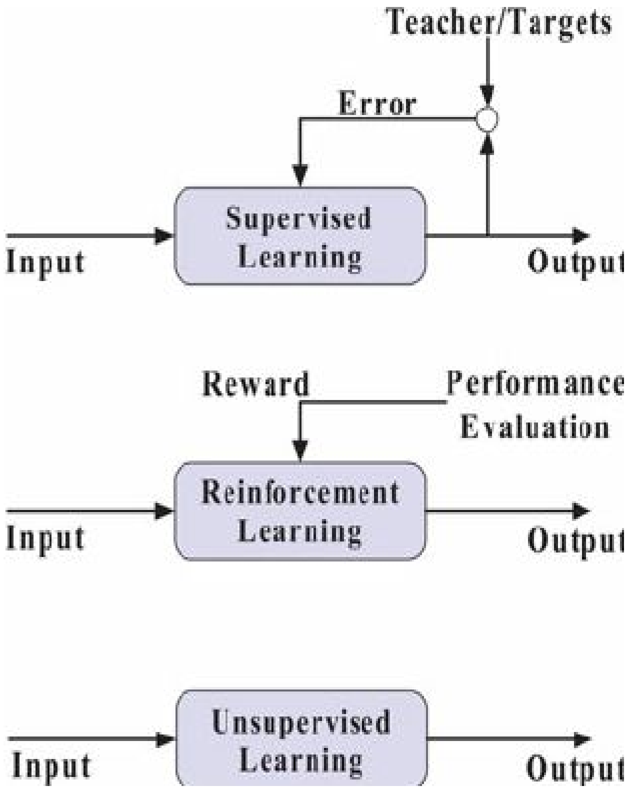
- *ML : "A Field of study that gives computers the ability to learn without being explicitly programmed.”*
  - Arthur Samuel, 1959
- Requires a large amount of data and computational power

Different ML tasks

SUPERVISED LEARNING

UNSUPERVISED LEARNING

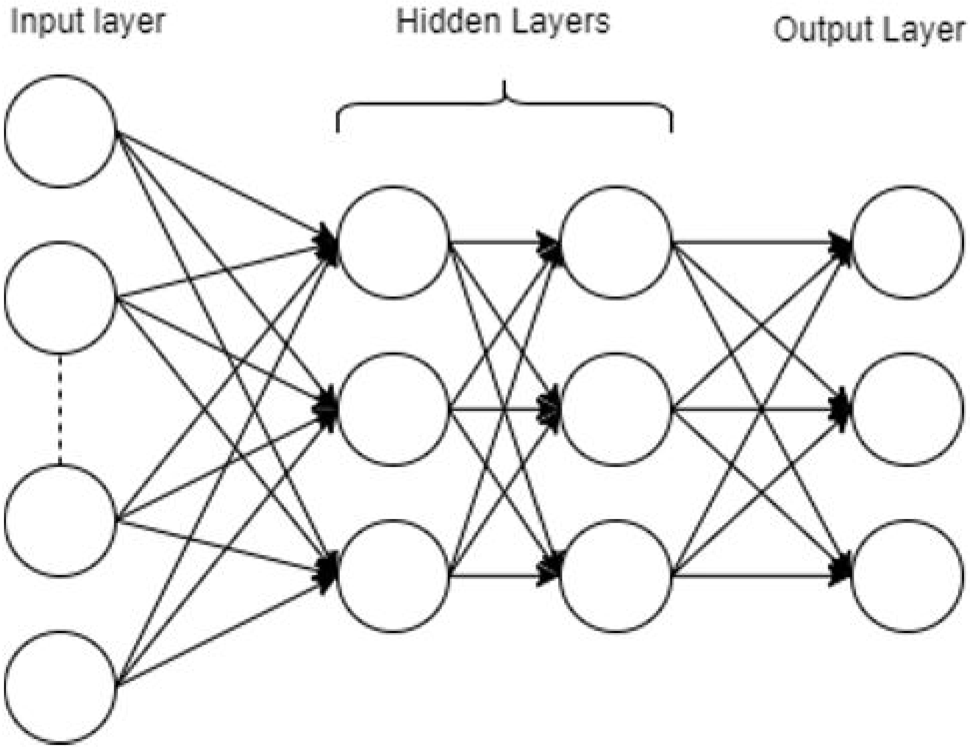
REINFORCEMENT LEARNING



Artificial Neural Networks

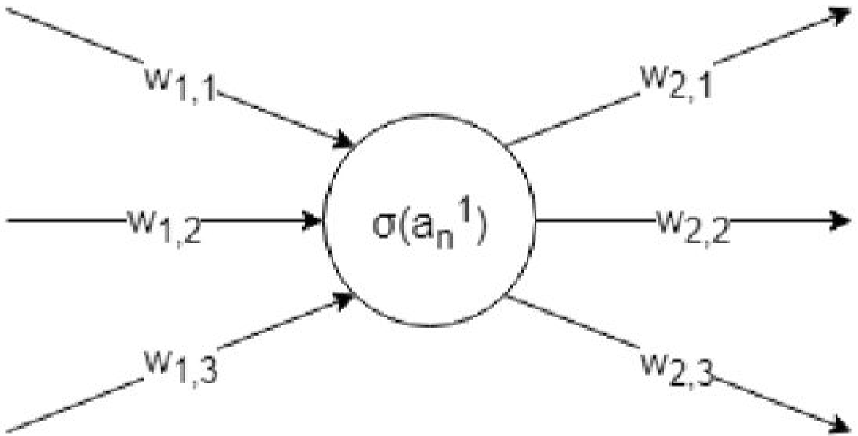
- Network Architecture
- Forward propagation
- Node activation / how they work
- Mathematical model of Neural network
- Gradient computation

Network Architecture



Forward propagation

$$a_n^1 = w_{1,1}a_1^0 + w_{1,2}a_2^0 + w_{1,3}a_3^0 + ... + w_{1,n}a_n^0$$



Mathematical model

$$\begin{bmatrix} a_1^2 \\ a_2^2 \\ a_3^2 \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{bmatrix} \begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \end{bmatrix} - \begin{bmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \end{bmatrix} \right)$$

Which can be rewritten for a more general case as

$$a^{n+1} = \sigma(Wa^n - b^n)$$

Gradient computation

$$J = \frac{1}{2}\sum(y - \hat{y})^2$$

$$\nabla J = (\frac{\delta J}{\delta w_{1,1}}, \frac{\delta J}{\delta b_1}, ..., \frac{\delta J}{\delta w_{j,k}}, \frac{\delta J}{\delta b_j})$$

TensorFlow Basics

- Tensors
- Computational graph and Sessions
- TensorBoard
- Placeholders
- Variables
- Training and Optimization

TensorFlow Basics

What is a Tensor?

A multidimensional array

Different

* ranks
* types

Rank	Math entity
0	Scalar (magnitude only)
1	Vector (magnitude and direction)
2	Matrix (table of numbers)
3	3-Tensor (cube of numbers)
n	n-Tensor (you get the idea)

```
mystr = tf.Variable(["Hello"], tf.string)
cool_numbers  = tf.Variable([3.14159, 2.71828], tf.float32)
first_primes = tf.Variable([2, 3, 5, 7, 11], tf.int32)
its_very_complicated = tf.Variable([(12.3, -4.85), (7.5, -6.23)], tf.complex64)
```

Computational graph and Sessions

Computational graph

- series of TensorFlow opts / nodes arranged into a graph

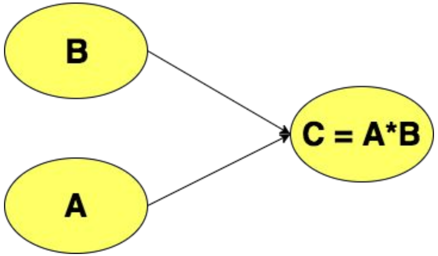
Session

- for graph evaluation

```
import tensorflow as tf

# Bulid graph
a = tf.constant([[-1.0, -1.0, -1.0],[-1.0, -1.0, -1.0]])
b = tf.constant(1.0, shape=[3, 2]) # an other way of defining a
    tensor
c = tf.matmul(a,b)

# Create a session object
sess = tf.Session()

# Run the graph
print(sess.run(c))
```

This produce the output:

```
[[-3.  -3.]
 [-3.  -3.]]
```



TensorBoard

- interactive visualization tool

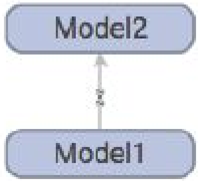
```
writer = tf.summary.FileWriter('output_folder', sess.graph)
```

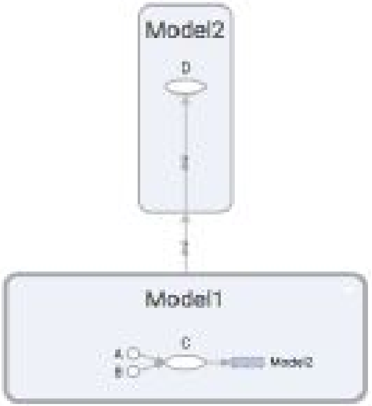
- Run the command: **tensorboard –logdir=path/to/log-directory**
- In a web browser, navigate to: **localhost:6006**

Add name and name scopes for better readability

```
import tensorflow as tf
a = tf.constant(-1.0, shape=[2, 3], name = 'A')
b = tf.constant(1.0, shape=[3, 2], name = 'B')
c = tf.matmul(a,b, name = 'C')

sess = tf.Session()
print(sess.run(c))
writer = tf.summary.FileWriter("output", sess.graph)
```


```
import tensorflow as tf
with tf.name_scope('Model1'):
    a = tf.constant(-1.0, shape=[2, 3], name = 'A')
    b = tf.constant(1.0, shape=[3, 2], name = 'B')
    c = tf.matmul(a,b, name = 'C')
with tf.name_scope('Model2'):
    d = tf.matmul(c,c, name = 'D')
sess = tf.Session()
print(sess.run(d))
writer = tf.summary.FileWriter("output", sess.graph)
```





Placeholders

```
import tensorflow as tf

# create placeholder
x = tf.placeholder(dtype=tf.float32)

# define session object in order to evaluate
sess = tf.Session()

# run and print place holder
print(sess.run(x)) # will fail since x is not provided
with values

# make random numbers with numpy, 4X4 tensor
rand_array = np.random.rand(4,4)

print(sess.run(x,feed_dict={x: rand_array})) # will
work
```

- Must be provided with values at a later stage

Variables

- trainable parameters
- initial value and explicitly initialized

```
v = tf.Variable([1.2,1.3])
sess = tf.Session()
initialize = tf.global_variables_initializer()
sess.run(initialize)
```

Training

- Adjust the Variables in our model to minimize a cost function
- tf.train choose optimization algorithm
- Base Class : Optimizer
  - provides methods to compute gradients
- GradientDecentOptimizer

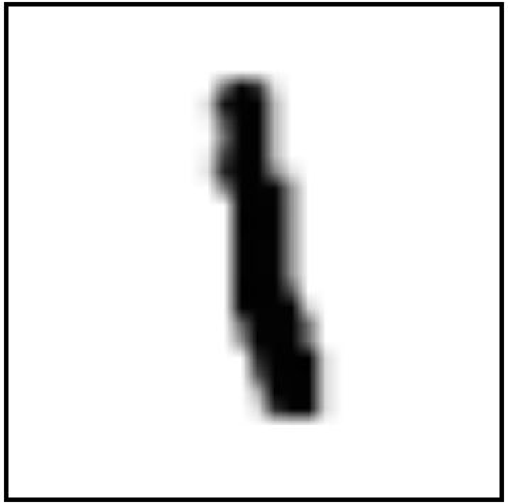
```
learning_rate = 0.01
optimizer =  tf.train.GradientDescentOptimizer(learning_rate)
train_step = optimizer.minimize(cost_function)
```

$$V_i = V_{i-1} - \alpha \frac{\partial J}{\partial V}\bigg|_{i-1}$$

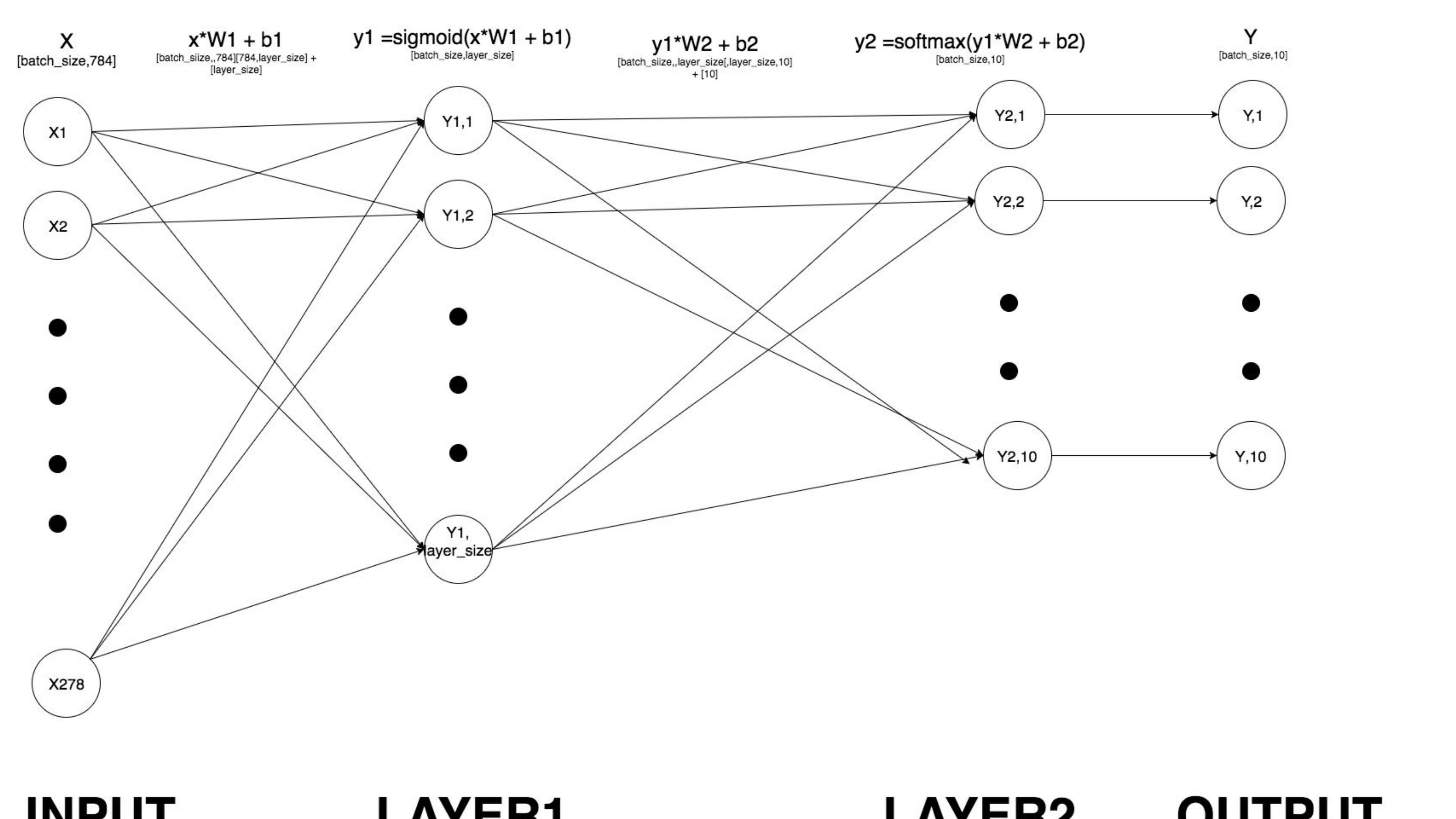
Image recognition example

- Digit recognition example
- Use MNIST database for training and validation data
- Example code
- Showcase model

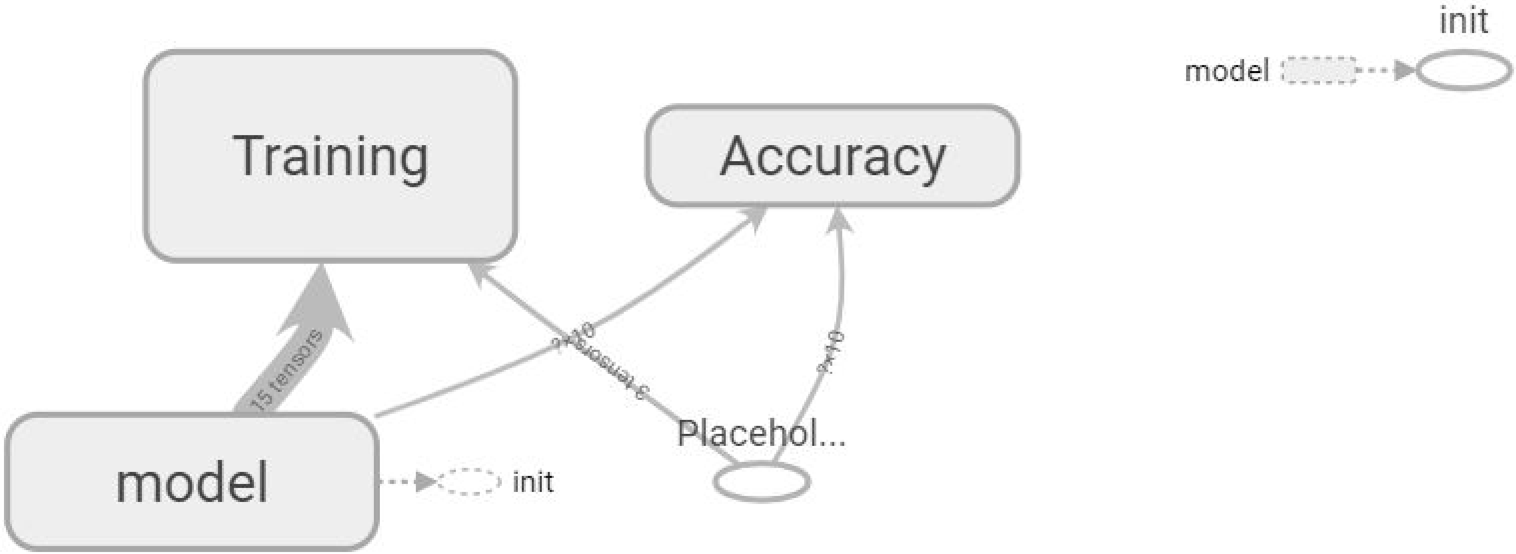
The MNIST database

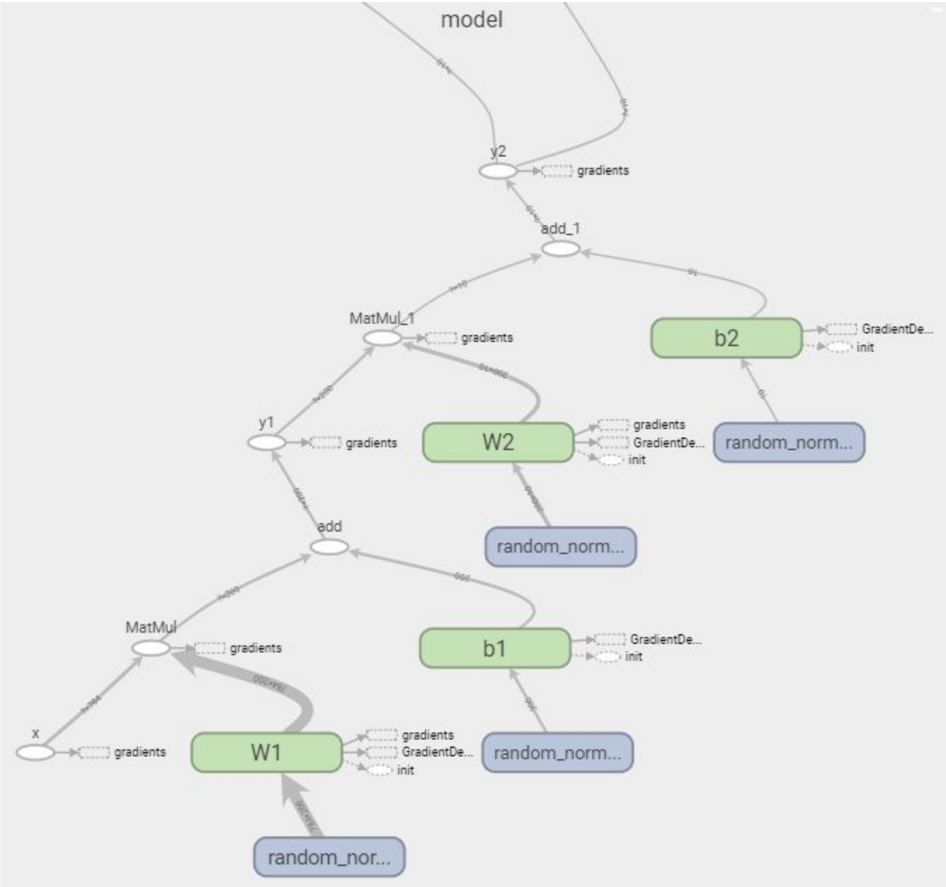


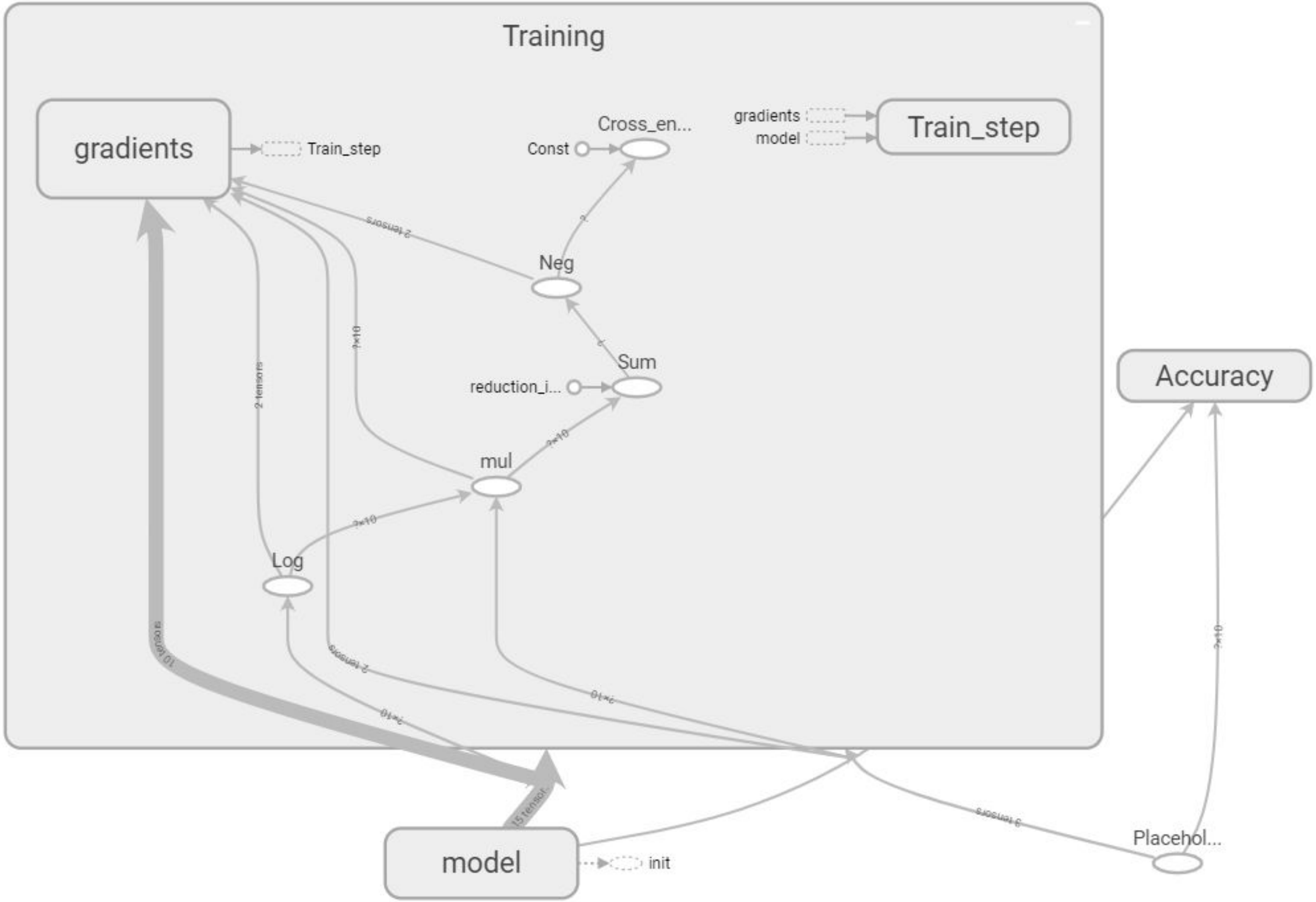
$$\simeq \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$


X
[batch_size,784]
x*W1 + b1
[batch_siize,,784][784,layer_size] + [layer_size]
y1 =sigmoid(x*W1 + b1)
[batch_size,layer_size]
y1*W2 + b2
[batch_siize,,layer_size[,layer_size,10] + [10]
y2 =softmax(y1*W2 + b2)
[batch_size,10]
Y
[batch_size,10]
X1
X2
X278
Y1,1
Y1,2
Y1, layer_size
Y2,1
Y2,2
Y2,10
Y,1
Y,2
Y,10
INPUT
LAYER1
LAYER2
OUTPUT

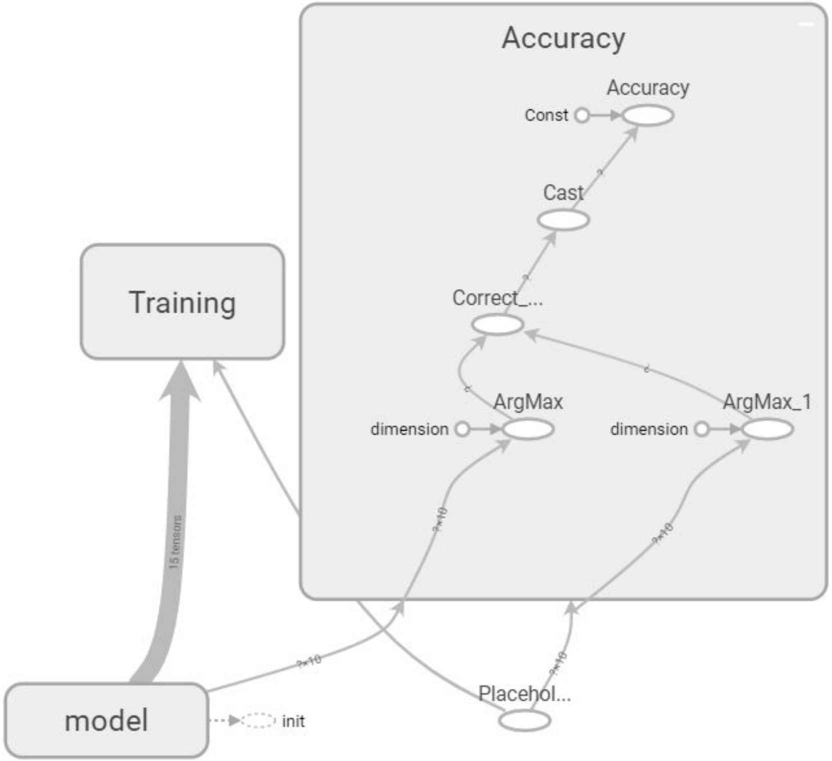
```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

with tf.name_scope('model'):
    #input
    x = tf.placeholder(tf.float32, [None, 784],name = 'x')

    #layer 1
    layer_size = 200
    W1 = tf.Variable(tf.random_normal([784, layer_size]),name = 'W1')
    b1 = tf.Variable(tf.random_normal( [layer_size]),name = 'b1')
    y1 = tf.nn.sigmoid(tf.matmul(x, W1) + b1,name = 'y1')

    #layer 2
    W2 = tf.Variable(tf.random_normal([layer_size,10]),name = 'W2')
    b2 = tf.Variable(tf.random_normal([10]),name = 'b2')
    y2 = tf.nn.softmax(tf.matmul(y1, W2) + b2,name = 'y2')

    #output
    y = y2
    #desired output
    y_ = tf.placeholder(tf.float32, [None, 10])

with tf.name_scope('Training'):
    cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y),
    reduction_indices=[1]),name ='Cross_entropy')
    learning_rate = 1
    train_step = tf.train.GradientDescentOptimizer(learning_rate,name ='Train_step').minimize(cross_entropy)

with tf.name_scope('Accuracy'):
    correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1),name ='Correct_prediction')
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32),name ='Accuracy')

sess = tf.InteractiveSession() # create session object
tf.global_variables_initializer().run() # initialize variables
tf.summary.FileWriter("graph", sess.graph) # TensorBoard visualization

batch_size=100
for i in range(10000):
    batch_xs, batch_ys = mnist.train.next_batch(batch_size)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_:
      mnist.test.labels}))
```


Training
Accuracy
15 tensors
3 tensors
Placehol...
model
init
model
init

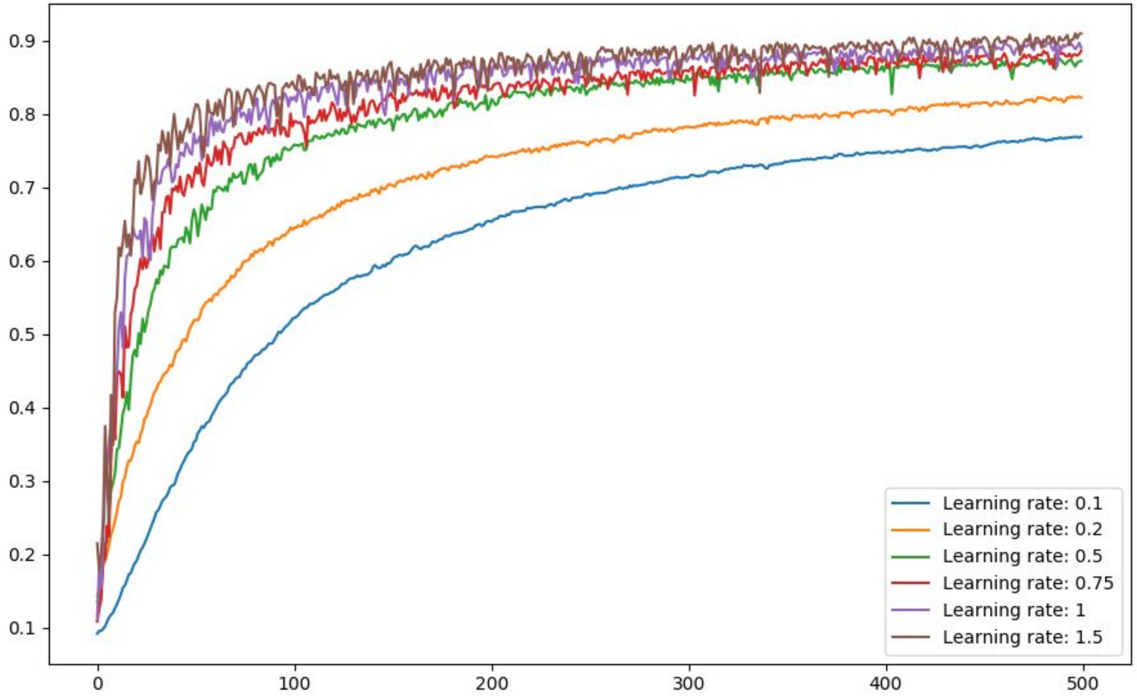

model
y2
gradients
add_1
MatMul_1
gradients
b2
GradientDe...
init
y1
gradients
W2
gradients
GradientDe...
init
random_norm...
add
random_norm...
MatMul
gradients
b1
GradientDe...
init
x
gradients
W1
gradients
GradientDe...
init
random_norm...
random_nor...



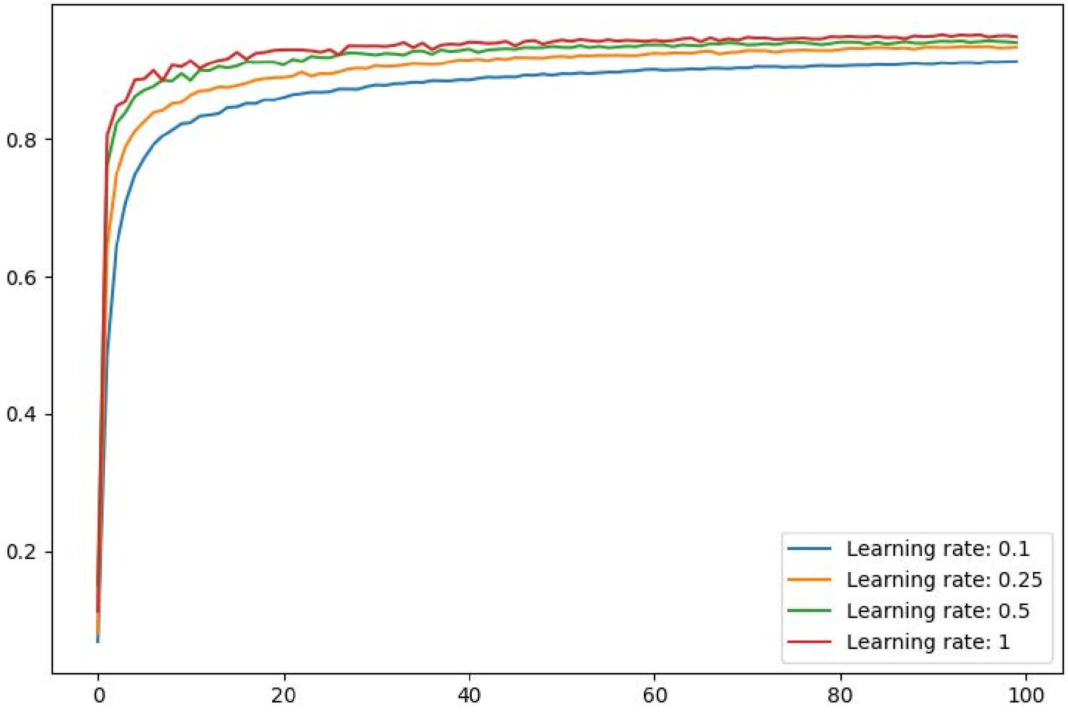
Training
gradients
Train_step
Const
Cross_en...
gradients
model
Train_step
2 tensors
Neg
?x10
2 tensors
Sum
reduction_i...
?x10
mul
?x10
Log
Accuracy
10 tensors
2 tensors
?x10
?x10
15 tensors
?x10
3 tensors
Placehol...
model
init


Accuracy
Accuracy
Const
Cast
Correct_...
ArgMax
ArgMax_1
dimension
dimension
Training
15 tensors
?×10
?×10
?×10
?×10
Placehol...
model
init

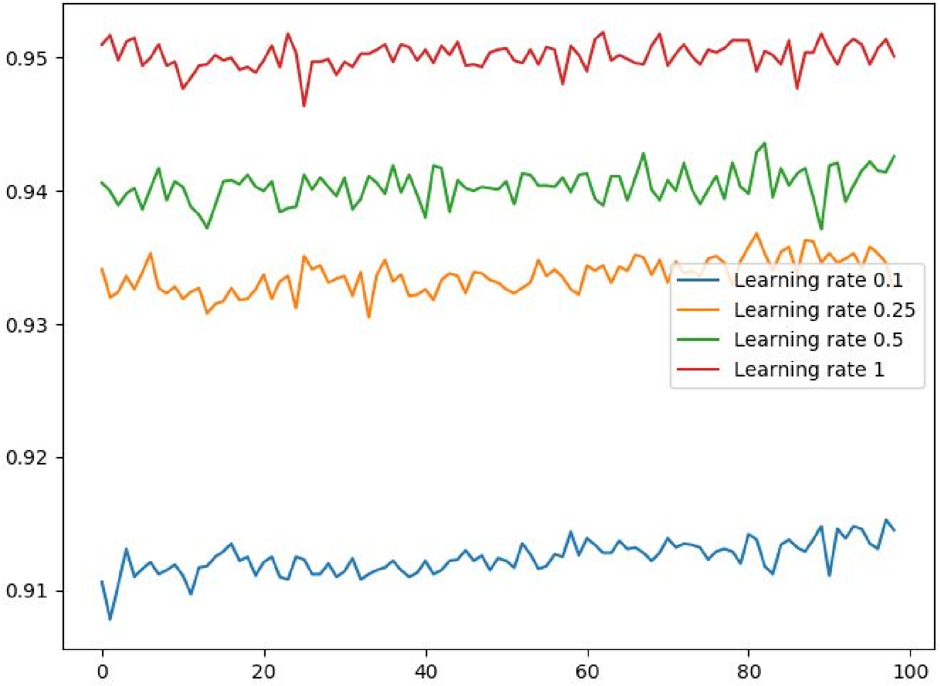
First 500 learning steps



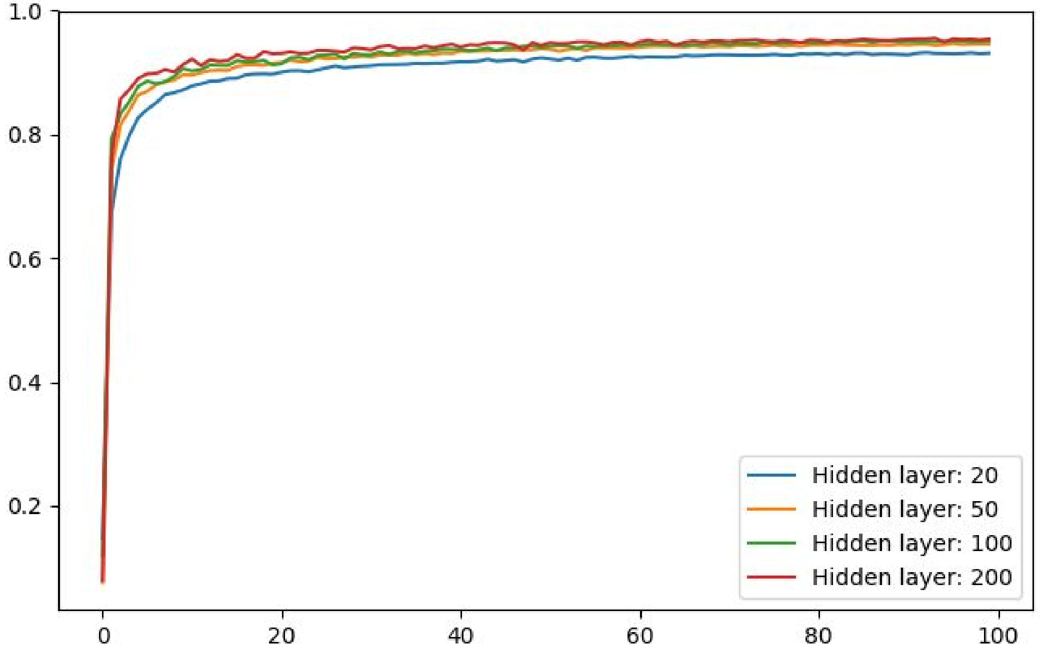
10 000 training steps



Final 100 training steps

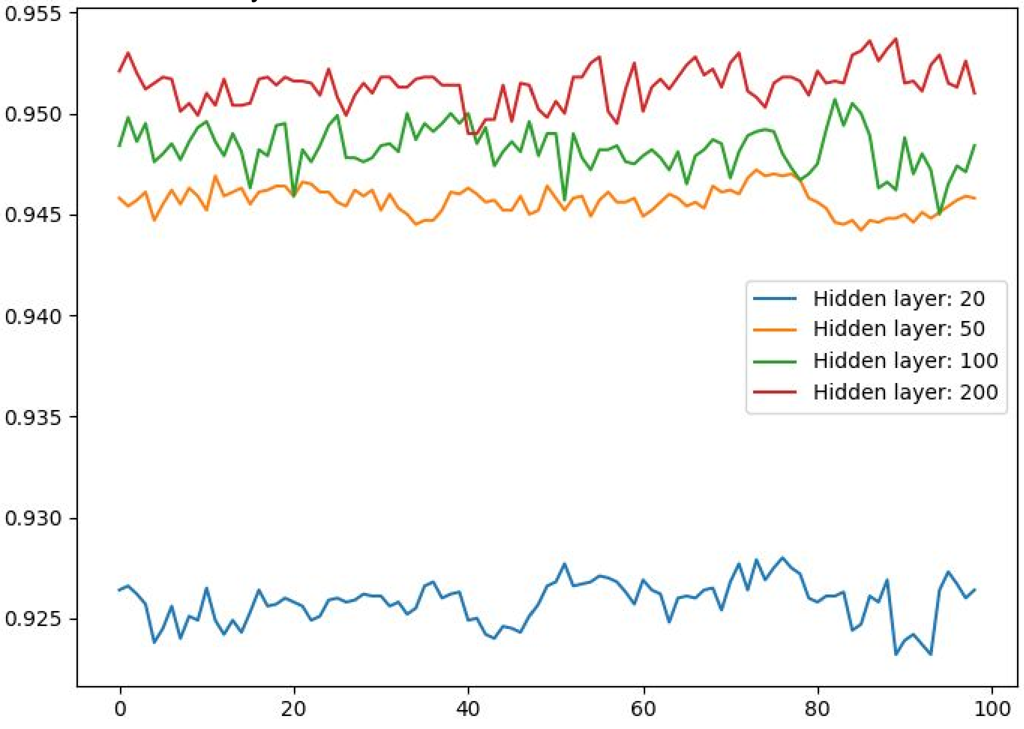


10 000 training steps

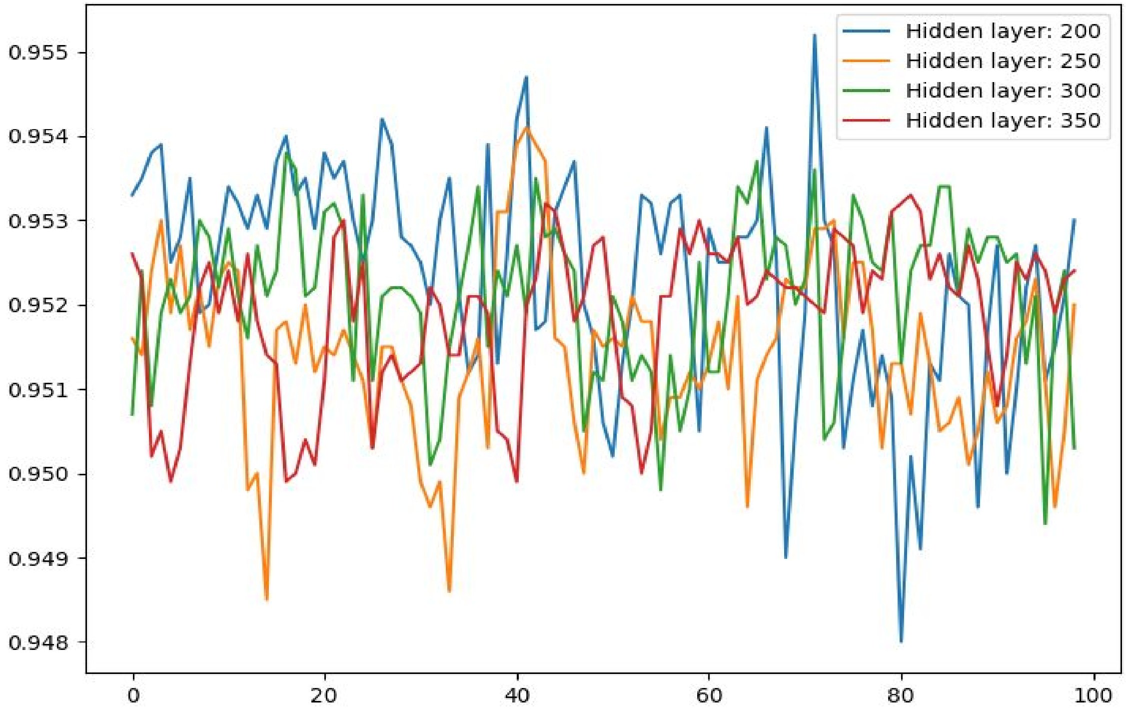


Last 100 training steps

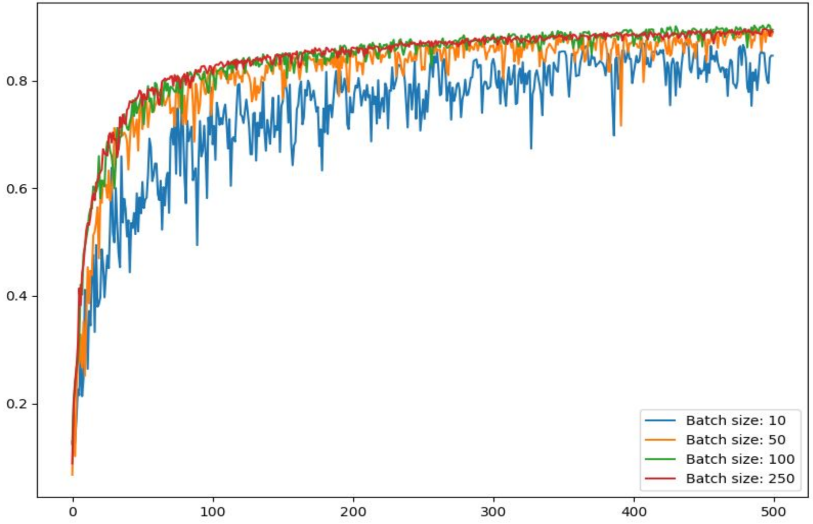
Accuracy plotted for different hidden layers

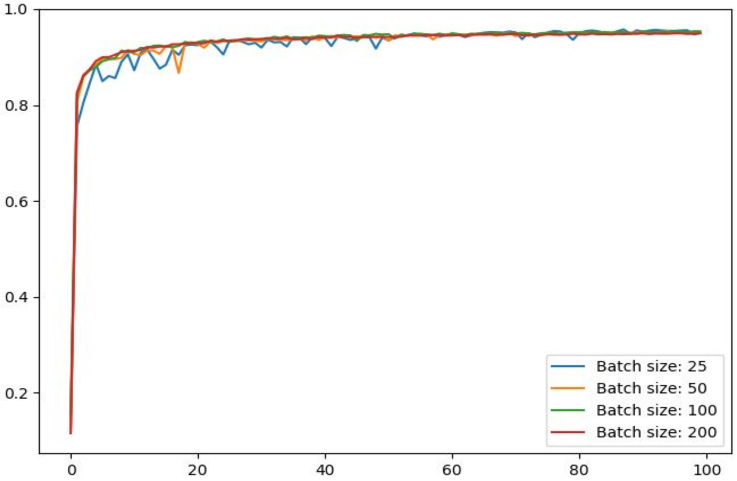


Final 100 training steps, for more nodes



Different batch sizes





Different numbers of hidden layers