TensorBoard

Mohamed Achraf BEN MOHAMED, PhD.

LaTICE laboratory

mohamedachraf@google.com

May 27, 2017

Overview

1. Introduction
2. Vizualize the tensorflow graph
    - FileWriter class
    - Demo 1
3. Cleaning the graph
    - Node Names
    - Name Space
    - Demo 2
4. Collect some summaries
    - tf.summury.histogram
    - tf.summury.scalar
    - Demo 3
5. Hyperparameter Search
    - Different Learning Rates
    - Demo 4

Exemple de Classification Supervisée

Authentification de billets de banque

Les données ont été extraites d'images provenant de spécimens authentiques de billets de banque.

Quatre paramètres ont été enregistrés :

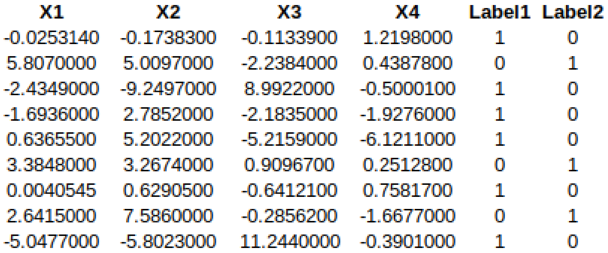
1. Variance
2. Skewness
3. Curtosis
4. Entropy

Deux classes : Vrais billets (1) ou Faux billets (0)

Source : https://archive.ics.uci.edu/ml/datasets/banknote+authentication

Exemple de Classification Supervisée

Authentification de billets de banque

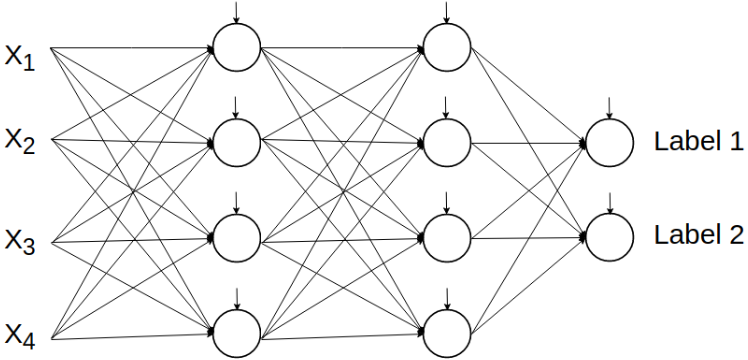


X1	X2	X3	X4	Label1	Label2
-0.0253140	-0.1738300	-0.1133900	1.2198000	1	0
5.8070000	5.0097000	-2.2384000	0.4387800	0	1
-2.4349000	-9.2497000	8.9922000	-0.5000100	1	0
-1.6936000	2.7852000	-2.1835000	-1.9276000	1	0
0.6365500	5.2022000	-5.2159000	-6.1211000	1	0
3.3848000	3.2674000	0.9096700	0.2512800	0	1
0.0040545	0.6290500	-0.6412100	0.7581700	1	0
2.6415000	7.5860000	-0.2856200	-1.6677000	0	1
-5.0477000	-5.8023000	11.2440000	-0.3901000	1	0

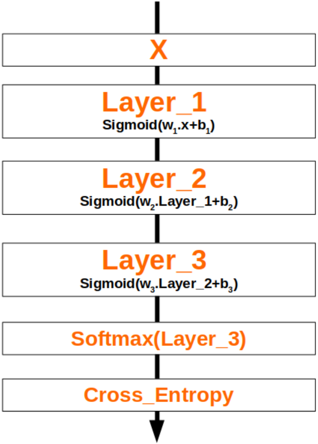
$m = 1372$

Source : https://archive.ics.uci.edu/ml/datasets/banknote+authentication

Architecture

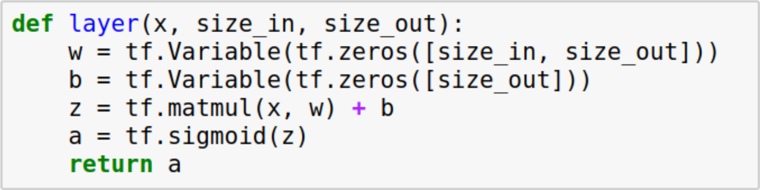


Model


X
Layer_1
Sigmoid(w1.x+b1)
Layer_2
Sigmoid(w2.Layer_1+b2)
Layer_3
Sigmoid(w3.Layer_2+b3)
Softmax(Layer_3)
Cross_Entropy

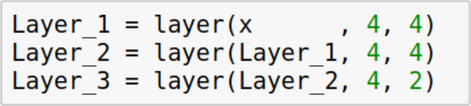
Code

Layer definition



```
def layer(x, size_in, size_out):
    w = tf.Variable(tf.zeros([size_in, size_out]))
    b = tf.Variable(tf.zeros([size_out]))
    z = tf.matmul(x, w) + b
    a = tf.sigmoid(z)
    return a
```

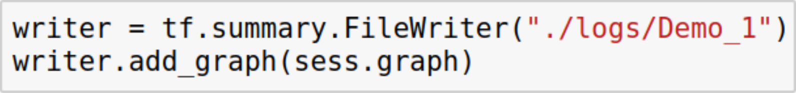
Model



```
Layer_1 = layer(x      , 4, 4)
Layer_2 = layer(Layer_1, 4, 4)
Layer_3 = layer(Layer_2, 4, 2)
```

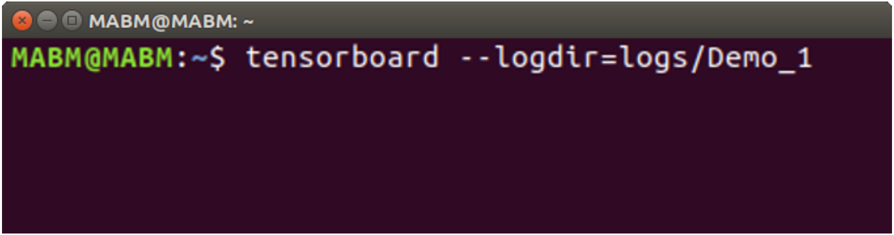
FileWriter class

Vizualize the tensorflow graph



```
writer = tf.summary.FileWriter("./logs/Demo_1")
writer.add_graph(sess.graph)
```

Turn on Tensorboard



```
MABM@MABM:~$ tensorboard --logdir=logs/Demo_1
```

DEMO 1

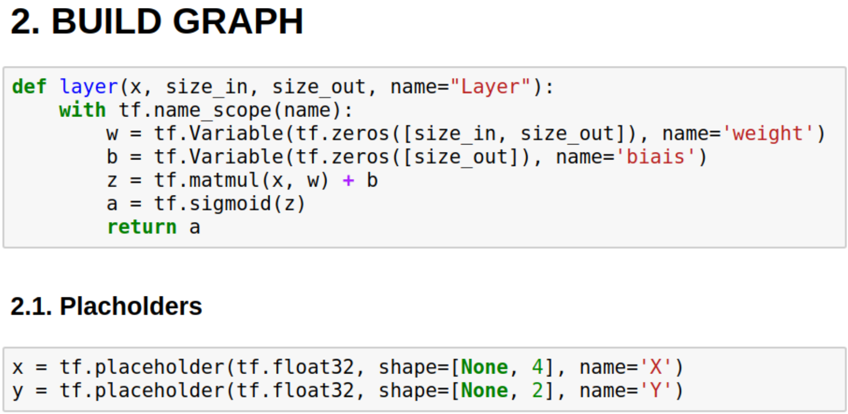
money_1.ipynb

https://github.com/LaTICE-laboratory-Monastir-unit/

Cleaning the graph

- Node Names
- Name Scope

Node Names



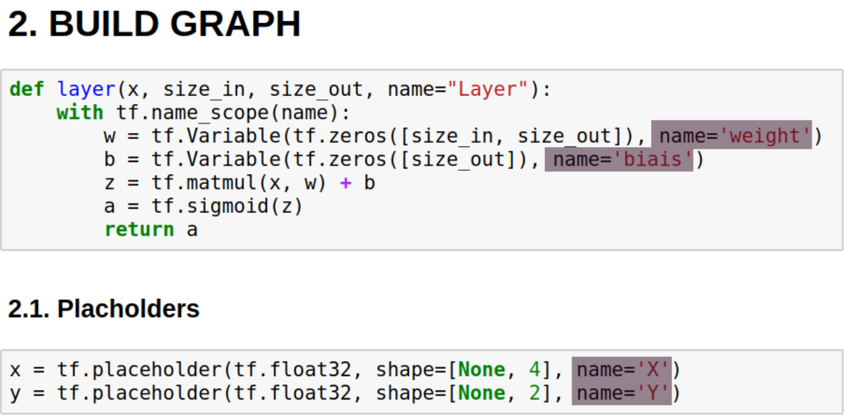
2. BUILD GRAPH

```
def layer(x, size_in, size_out, name="Layer"):
    with tf.name_scope(name):
        w = tf.Variable(tf.zeros([size_in, size_out]), name='weight')
        b = tf.Variable(tf.zeros([size_out]), name='biais')
        z = tf.matmul(x, w) + b
        a = tf.sigmoid(z)
        return a
```

2.1. Placholders

```
x = tf.placeholder(tf.float32, shape=[None, 4], name='X')
y = tf.placeholder(tf.float32, shape=[None, 2], name='Y')
```

Node Names



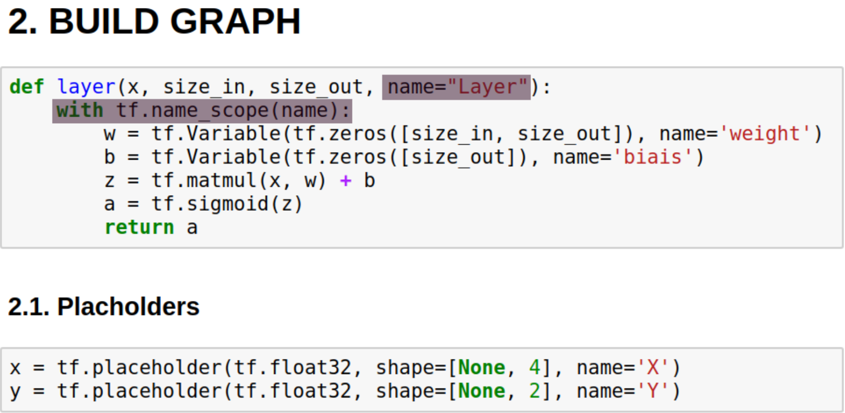
2. BUILD GRAPH

```
def layer(x, size_in, size_out, name="Layer"):
    with tf.name_scope(name):
        w = tf.Variable(tf.zeros([size_in, size_out]), name='weight')
        b = tf.Variable(tf.zeros([size_out]), name='biais')
        z = tf.matmul(x, w) + b
        a = tf.sigmoid(z)
        return a
```

2.1. Placholders

```
x = tf.placeholder(tf.float32, shape=[None, 4], name='X')
y = tf.placeholder(tf.float32, shape=[None, 2], name='Y')
```

Name Space



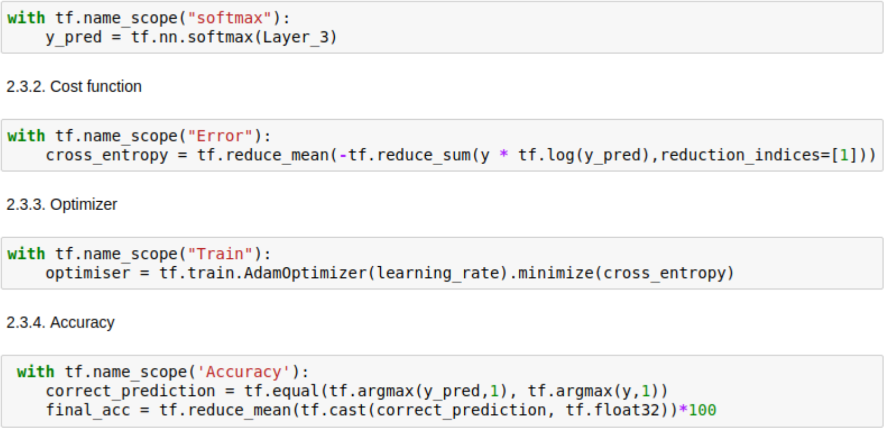
2. BUILD GRAPH

```
def layer(x, size_in, size_out, name="Layer"):
    with tf.name_scope(name):
        w = tf.Variable(tf.zeros([size_in, size_out]), name='weight')
        b = tf.Variable(tf.zeros([size_out]), name='biais')
        z = tf.matmul(x, w) + b
        a = tf.sigmoid(z)
        return a
```

2.1. Placholders

```
x = tf.placeholder(tf.float32, shape=[None, 4], name='X')
y = tf.placeholder(tf.float32, shape=[None, 2], name='Y')
```

Name Space



```
with tf.name_scope("softmax"):
    y_pred = tf.nn.softmax(Layer_3)
```

2.3.2. Cost function

```
with tf.name_scope("Error"):
    cross_entropy = tf.reduce_mean(-tf.reduce_sum(y * tf.log(y_pred),reduction_indices=[1]))
```

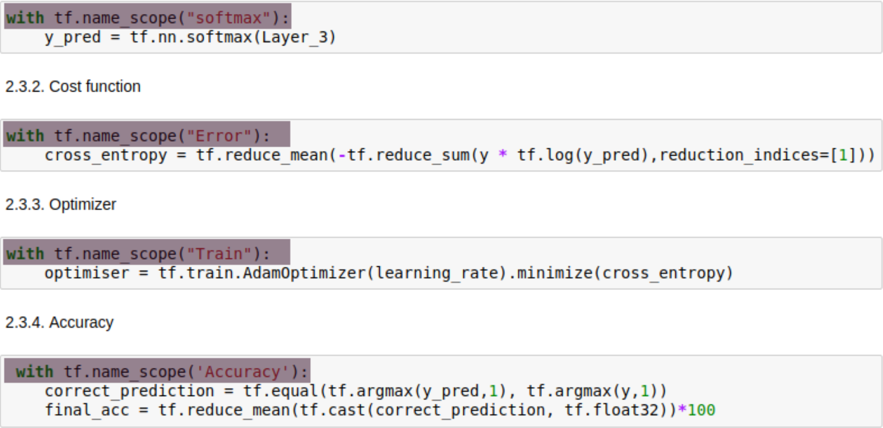
2.3.3. Optimizer

```
with tf.name_scope("Train"):
    optimiser = tf.train.AdamOptimizer(learning_rate).minimize(cross_entropy)
```

2.3.4. Accuracy

```
 with tf.name_scope('Accuracy'):
    correct_prediction = tf.equal(tf.argmax(y_pred,1), tf.argmax(y,1))
    final_acc = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))*100
```

Name Space



```
with tf.name_scope("softmax"):
    y_pred = tf.nn.softmax(Layer_3)
```

2.3.2. Cost function

```
with tf.name_scope("Error"):
    cross_entropy = tf.reduce_mean(-tf.reduce_sum(y * tf.log(y_pred),reduction_indices=[1]))
```

2.3.3. Optimizer

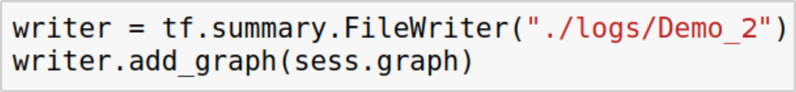
```
with tf.name_scope("Train"):
    optimiser = tf.train.AdamOptimizer(learning_rate).minimize(cross_entropy)
```

2.3.4. Accuracy

```
 with tf.name_scope('Accuracy'):
    correct_prediction = tf.equal(tf.argmax(y_pred,1), tf.argmax(y,1))
    final_acc = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))*100
```

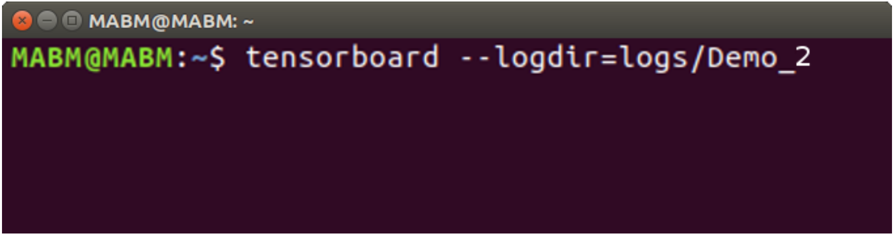
FileWriter class

Vizualize the tensorflow graph



```
writer = tf.summary.FileWriter("./logs/Demo_2")
writer.add_graph(sess.graph)
```

Turn on Tensorboard



```
MABM@MABM:~$ tensorboard --logdir=logs/Demo_2
```

DEMO 2

IPYNB

money_2.ipynb

GitHub

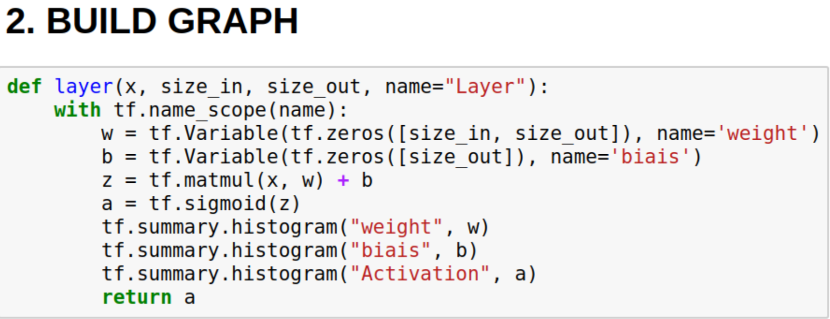
https://github.com/LaTICE-laboratory-Monastir-unit/

summary (n) : Tensorflow operation that summarized data.

Examples :

- **tf.summary.scalar**
- **tf.summary.histogram**

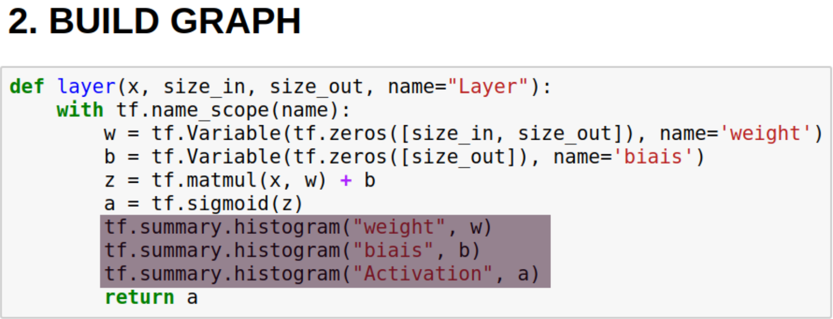
Histogram



2. BUILD GRAPH

```
def layer(x, size_in, size_out, name="Layer"):
    with tf.name_scope(name):
        w = tf.Variable(tf.zeros([size_in, size_out]), name='weight')
        b = tf.Variable(tf.zeros([size_out]), name='biais')
        z = tf.matmul(x, w) + b
        a = tf.sigmoid(z)
        tf.summary.histogram("weight", w)
        tf.summary.histogram("biais", b)
        tf.summary.histogram("Activation", a)
        return a
```

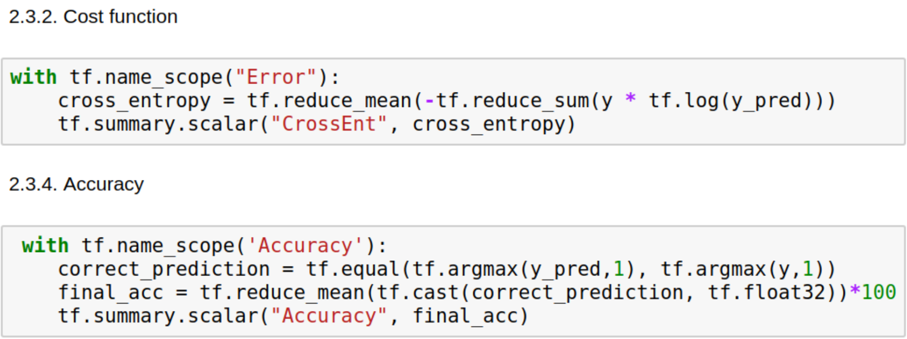
Histogram



2. BUILD GRAPH

```
def layer(x, size_in, size_out, name="Layer"):
    with tf.name_scope(name):
        w = tf.Variable(tf.zeros([size_in, size_out]), name='weight')
        b = tf.Variable(tf.zeros([size_out]), name='biais')
        z = tf.matmul(x, w) + b
        a = tf.sigmoid(z)
        tf.summary.histogram("weight", w)
        tf.summary.histogram("biais", b)
        tf.summary.histogram("Activation", a)
        return a
```

Scalar



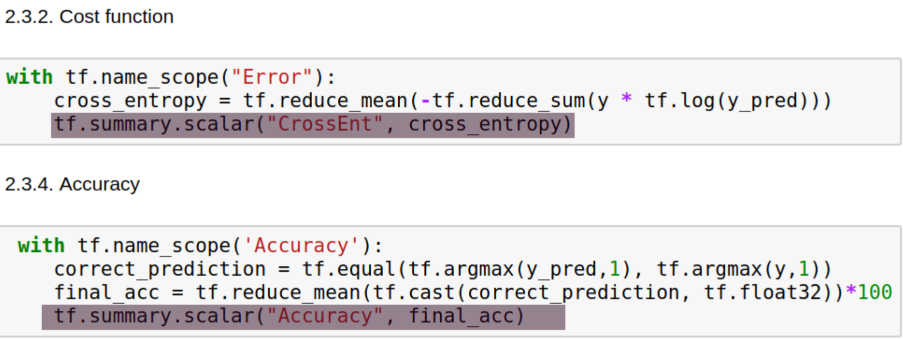
2.3.2. Cost function

```
with tf.name_scope("Error"):
    cross_entropy = tf.reduce_mean(-tf.reduce_sum(y * tf.log(y_pred)))
    tf.summary.scalar("CrossEnt", cross_entropy)
```

2.3.4. Accuracy

```
 with tf.name_scope('Accuracy'):
    correct_prediction = tf.equal(tf.argmax(y_pred,1), tf.argmax(y,1))
    final_acc = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))*100
    tf.summary.scalar("Accuracy", final_acc)
```

Scalar



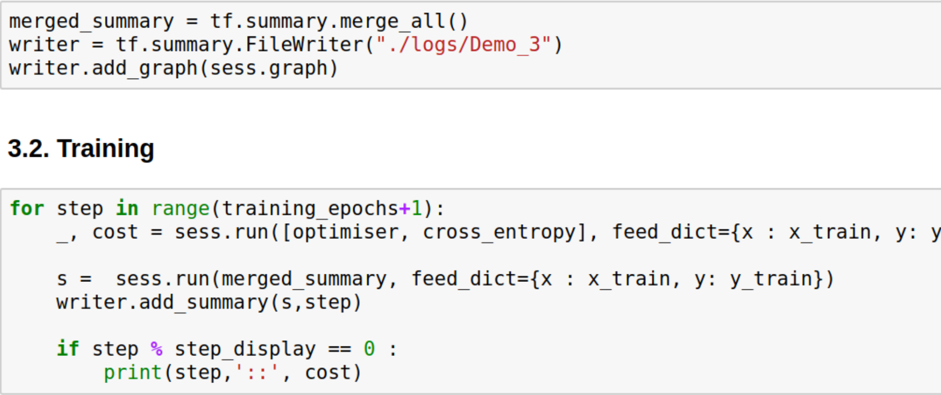
2.3.2. Cost function

```
with tf.name_scope("Error"):
    cross_entropy = tf.reduce_mean(-tf.reduce_sum(y * tf.log(y_pred)))
    tf.summary.scalar("CrossEnt", cross_entropy)
```

2.3.4. Accuracy

```
 with tf.name_scope('Accuracy'):
    correct_prediction = tf.equal(tf.argmax(y_pred,1), tf.argmax(y,1))
    final_acc = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))*100
    tf.summary.scalar("Accuracy", final_acc)
```

Merge Summaries

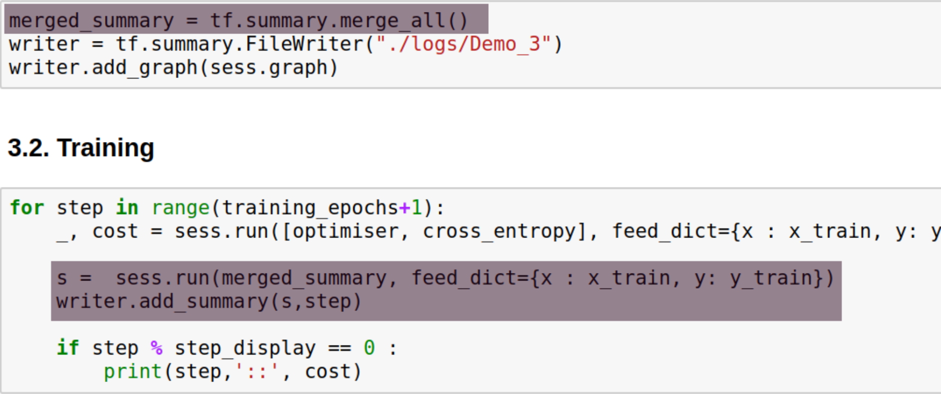


```
merged_summary = tf.summary.merge_all()
writer = tf.summary.FileWriter("./logs/Demo_3")
writer.add_graph(sess.graph)
```

3.2. Training

```
for step in range(training_epochs+1):
    _, cost = sess.run([optimiser, cross_entropy], feed_dict={x : x_train, y: y

    s =  sess.run(merged_summary, feed_dict={x : x_train, y: y_train})
    writer.add_summary(s,step)

    if step % step_display == 0 :
        print(step,'::', cost)
```

Merge Summaries



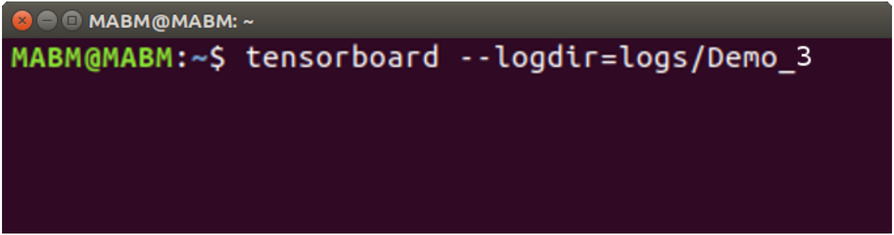
```
merged_summary = tf.summary.merge_all()
writer = tf.summary.FileWriter("./logs/Demo_3")
writer.add_graph(sess.graph)
```

3.2. Training

```
for step in range(training_epochs+1):
    _, cost = sess.run([optimiser, cross_entropy], feed_dict={x : x_train, y: y

    s =  sess.run(merged_summary, feed_dict={x : x_train, y: y_train})
    writer.add_summary(s,step)

    if step % step_display == 0 :
        print(step,'::', cost)
```

FileWriter class

Turn on Tensorboard



```
MABM@MABM:~$ tensorboard --logdir=logs/Demo_3
```

DEMO 3

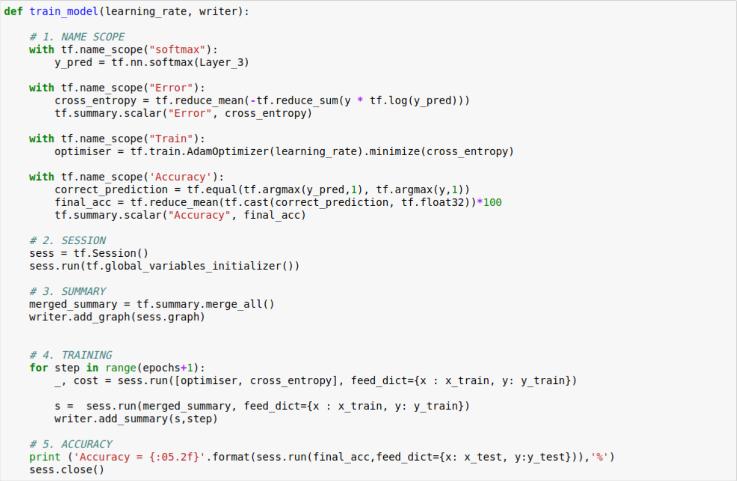
money_3.ipynb

https://github.com/LaTICE-laboratory-Monastir-unit/

Hyperparameter Search

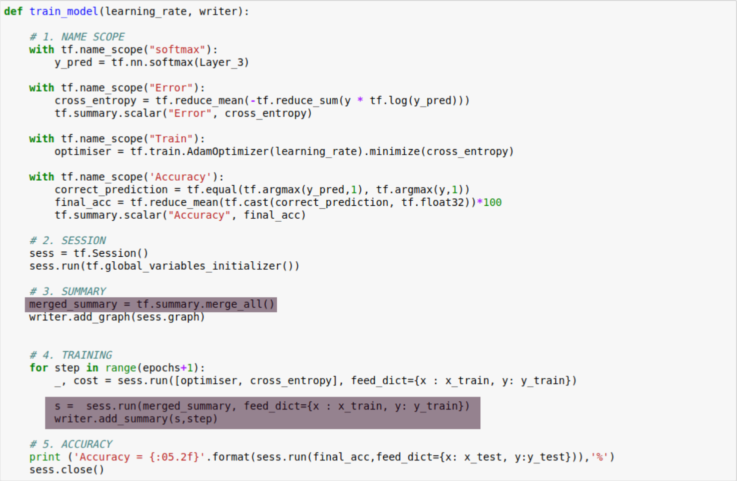
What's the best learning rates ?
What's the best model architecture ?

Hyperparameter



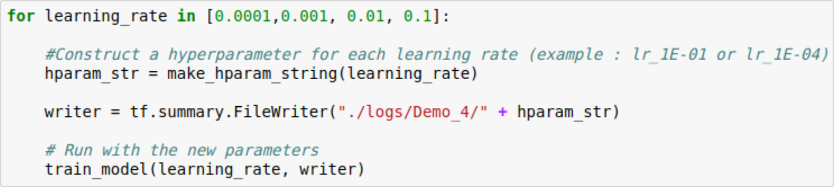
```python
def train_model(learning_rate, writer):

    # 1. NAME SCOPE
    with tf.name_scope("softmax"):
        y_pred = tf.nn.softmax(Layer_3)

    with tf.name_scope("Error"):
        cross_entropy = tf.reduce_mean(-tf.reduce_sum(y * tf.log(y_pred)))
        tf.summary.scalar("Error", cross_entropy)

    with tf.name_scope("Train"):
        optimiser = tf.train.AdamOptimizer(learning_rate).minimize(cross_entropy)

    with tf.name_scope('Accuracy'):
        correct_prediction = tf.equal(tf.argmax(y_pred,1), tf.argmax(y,1))
        final_acc = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))*100
        tf.summary.scalar("Accuracy", final_acc)

    # 2. SESSION
    sess = tf.Session()
    sess.run(tf.global_variables_initializer())

    # 3. SUMMARY
    merged_summary = tf.summary.merge_all()
    writer.add_graph(sess.graph)


    # 4. TRAINING
    for step in range(epochs+1):
        _, cost = sess.run([optimiser, cross_entropy], feed_dict={x : x_train, y: y_train})

        s =  sess.run(merged_summary, feed_dict={x : x_train, y: y_train})
        writer.add_summary(s,step)

    # 5. ACCURACY
    print ('Accuracy = {:05.2f}'.format(sess.run(final_acc,feed_dict={x: x_test, y:y_test})),'%')
    sess.close()
```

Hyperparameter

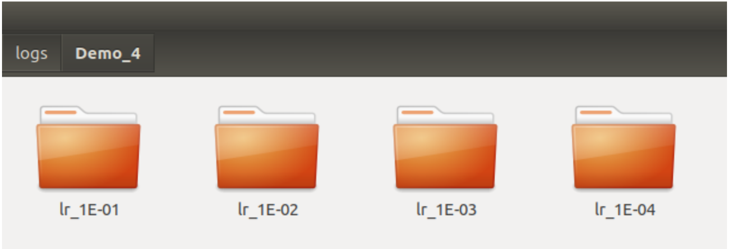


```
def train_model(learning_rate, writer):

    # 1. NAME SCOPE
    with tf.name_scope("softmax"):
        y_pred = tf.nn.softmax(Layer_3)

    with tf.name_scope("Error"):
        cross_entropy = tf.reduce_mean(-tf.reduce_sum(y * tf.log(y_pred)))
        tf.summary.scalar("Error", cross_entropy)

    with tf.name_scope("Train"):
        optimiser = tf.train.AdamOptimizer(learning_rate).minimize(cross_entropy)

    with tf.name_scope('Accuracy'):
        correct_prediction = tf.equal(tf.argmax(y_pred,1), tf.argmax(y,1))
        final_acc = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))*100
        tf.summary.scalar("Accuracy", final_acc)

    # 2. SESSION
    sess = tf.Session()
    sess.run(tf.global_variables_initializer())

    # 3. SUMMARY
    merged_summary = tf.summary.merge_all()
    writer.add_graph(sess.graph)


    # 4. TRAINING
    for step in range(epochs+1):
        _, cost = sess.run([optimiser, cross_entropy], feed_dict={x : x_train, y: y_train})

        s =  sess.run(merged_summary, feed_dict={x : x_train, y: y_train})
        writer.add_summary(s,step)

    # 5. ACCURACY
    print ('Accuracy = {:05.2f}'.format(sess.run(final_acc,feed_dict={x: x_test, y:y_test})),'%')
    sess.close()
```

Hyperparameter

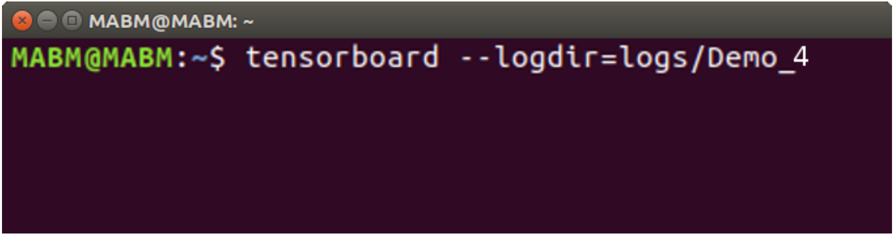


```
for learning_rate in [0.0001,0.001, 0.01, 0.1]:

    #Construct a hyperparameter for each learning rate (example : lr_1E-01 or lr_1E-04)
    hparam_str = make_hparam_string(learning_rate)

    writer = tf.summary.FileWriter("./logs/Demo_4/" + hparam_str)

    # Run with the new parameters
    train_model(learning_rate, writer)
```

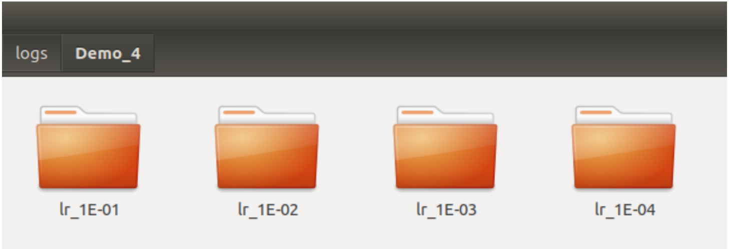


FileWriter class

Turn on Tensorboard



```
MABM@MABM:~$ tensorboard --logdir=logs/Demo_4
```



DEMO 4

money_4.ipynb

https://github.com/LaTICE-laboratory-Monastir-unit/

References

"TensorBoard: Visualizing Learning — TensorFlow.". Google, n.d. Web. 23 May 2017. Website: $https : //www.tensorflow.org/get_s tarted/summaries_a nd_t ensorboard$

Dandelion Man (Feb 15, 2017), Hands-on TensorBoard (TensorFlow Dev Summit 2017). Retrieved from https://www.youtube.com/watch?v=eBbEDRsCmv4

Fin.