Assignment_6_Wanat_final

August 4, 2019

In this notebook the MNIST data sets were examined in TensorFlow via three different ways. First, the data was classified via plain TensorFlow in a 2x2 completely crossed design benchmark experiment. This was a simple comparison of neural networks which examined the accuracy between two and five hidden layers. For each layer the number of nodes was evaluated at 10 and 20 nodes each. The processing time and accuracy for both the training and test data sets was recorded and compared.

Then a TensorFlow premade estimator called a DNN Classifier was used to create another model. This model contained two hidden layers with 128 and 32 nodes, respectively.

Finally, Keras was used in TensorFlow to create two models. One model contained two hidden layers with 128 and 32 nodes, respectively. The second model contained two hidden layers with 20 nodes each. The first Keras model's accuracy and loss was plotted in matplotlib. The second Keras model used a confusion matrix to compare and visualize the true values against the predicted values. Both Keras models were visualized utilizing TensorBoard.

1 Tensorflow

1.1 Import libraries

[1]:
```
# import libraries
import tensorflow as tf
import numpy as np
import keras
import time
import pandas as pd

#reset graph to remove duplicate nodes
tf.reset_default_graph()
```

```
Using TensorFlow backend.
```

[2]:
```
RANDOM_SEED = 1
tf.compat.v1.random.set_random_seed(RANDOM_SEED)
```

1.2 Data: import, prepare, reshape and normalize

[3]:
```
# download data set
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()

# prepare data, reshape and normalize
X_train = X_train.astype(np.float32).reshape(-1,28*28)/255.0
X_test = X_test.astype(np.float32).reshape(-1,28*28)/255.0
y_train = y_train.astype(np.int32)
y_test = y_test.astype(np.int32)
```

1.3 Set up: layers, cost function, optimizer

[4]:
```
# set parameters for neural nodes
n_inputs = 28*28  # MNIST
n_hidden1 = 10
n_hidden2 = 10
# there are 10 different numbers to be classified
n_outputs = 10

# create placeholders for variables
X = tf.placeholder(tf.float32, shape=(None, n_inputs), name="X")
y = tf.placeholder(tf.int32, shape=(None), name="y")


# create standard neural network layers
with tf.name_scope("dnn"):
    hidden1 = tf.layers.dense(X, n_hidden1, name="hidden1", activation=tf.nn.
 ↪relu)
    hidden2 = tf.layers.dense(hidden1, n_hidden2, name="hidden2", activation=tf.
 ↪nn.relu)
    logits = tf.layers.dense(hidden2, n_outputs, name="outputs")

# define cost function
with tf.name_scope("loss"):
    xentropy = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y,␣
 ↪logits=logits)
    loss = tf.reduce_mean(xentropy, name="loss")
tf.summary.scalar('xentropy', xentropy)
tf.summary.scalar('loss', loss)

# use optimizer to modify parameters to minimize cost function
learning_rate = 0.01

with tf.name_scope("train"):
    optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss)
```

```
# evaluate the model using accuracy
with tf.name_scope("eval"):
    correct = tf.nn.in_top_k(logits, y, 1)
    accuracy = tf.reduce_mean(tf.cast(correct, tf.float32))
tf.summary.scalar('accuracy', accuracy)

# save trained model parameters to disk
saver = tf.train.Saver()
```

```
WARNING:tensorflow:From <ipython-input-4-a2cea367309a>:15: dense (from
tensorflow.python.layers.core) is deprecated and will be removed in a future
version.
Instructions for updating:
Use keras.layers.dense instead.
WARNING:tensorflow:From /Users/jmwanat/anaconda3/envs/tf/lib/python3.7/site-
packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
```

1.4 Model with 2 layers and 10 nodes per layer

[5]:
```
# set parameters for neural nodes
n_inputs = 28*28  # MNIST
n_hidden1 = 10
n_hidden2 = 10
# there are 10 different numbers to be classified
n_outputs = 10
n_epochs = 40
batch_size = 50

# Need to shuffle X and y data sets or the model does not train on all numbers
# if you don't do this the model is very inaccurate because it is not looking
# at all the numbers
# Used function defined by Geron 2017
#https://github.com/ageron/handson-ml/blob/master/
 ↪10_introduction_to_artificial_neural_networks.ipynb
def shuffle_batch(X, y, batch_size):
    rnd_idx = np.random.permutation(len(X))
    n_batches = len(X) // batch_size
    for batch_idx in np.array_split(rnd_idx, n_batches):
        X_batch, y_batch = X[batch_idx], y[batch_idx]
        yield X_batch, y_batch
```

```
# Initialize a session so that we can run TensorFlow operations
with tf.Session() as session:

    t0 = time.time()
    # Run the global variable initializer to initialize all variables and␣
 ↪layers of the neural network
    session.run(tf.global_variables_initializer())

    for epoch in range(n_epochs):
        for X_batch, y_batch in shuffle_batch(X_train, y_train, batch_size):
        # Feed in the training data and do one step of neural network training
            session.run(optimizer, feed_dict={X: X_batch, y: y_batch})

        if epoch % 5 == 0:
            train_loss = session.run(loss, feed_dict={X: X_batch, y:y_batch})
            train_accuracy = accuracy.eval(feed_dict={X: X_batch, y: y_batch})
            print("Epoch: {}, Loss: {}, Accuracy: {}".
 ↪format(epoch,train_loss,train_accuracy))

        # Print the current training status to the screen
        print("Training pass: {}".format(epoch))

    test_accuracy = accuracy.eval(feed_dict={X: X_test, y: y_test})
    print("Test accuracy: {}".format(test_accuracy))
    # Training is now complete!
    print("Training is complete!")
    t1 = time.time()
    print("Time to train and evaluate model: {:.3f} seconds".format(t1-t0))
    total_time = t1 - t0

time1 = total_time
```

```
Epoch: 0, Loss: 0.6462172865867615, Accuracy: 0.8399999737739563
Training pass: 0
Training pass: 1
Training pass: 2
Training pass: 3
Training pass: 4
Epoch: 5, Loss: 0.2703838348388672, Accuracy: 0.8799999952316284
Training pass: 5
Training pass: 6
Training pass: 7
Training pass: 8
Training pass: 9
Epoch: 10, Loss: 0.33837810158729553, Accuracy: 0.8799999952316284
Training pass: 10
Training pass: 11
```

```
Training pass: 12
Training pass: 13
Training pass: 14
Epoch: 15, Loss: 0.16312584280967712, Accuracy: 0.9800000190734863
Training pass: 15
Training pass: 16
Training pass: 17
Training pass: 18
Training pass: 19
Epoch: 20, Loss: 0.19218683242797852, Accuracy: 0.9200000166893005
Training pass: 20
Training pass: 21
Training pass: 22
Training pass: 23
Training pass: 24
Epoch: 25, Loss: 0.1597401648759842, Accuracy: 0.9599999785423279
Training pass: 25
Training pass: 26
Training pass: 27
Training pass: 28
Training pass: 29
Epoch: 30, Loss: 0.22012335062026978, Accuracy: 0.9399999976158142
Training pass: 30
Training pass: 31
Training pass: 32
Training pass: 33
Training pass: 34
Epoch: 35, Loss: 0.2994080185890198, Accuracy: 0.9399999976158142
Training pass: 35
Training pass: 36
Training pass: 37
Training pass: 38
Training pass: 39
Test accuracy: 0.9345999956130981
Training is complete!
Time to train and evaluate model: 44.758 seconds
```

1.5 Model with 2 layers and 20 nodes per layer

[6]:
```
n_hidden1 = 20
n_hidden2 = 20
```

[7]:
```
with tf.Session() as session:

    t0 = time.time()
    # Run the global variable initializer to initialize all variables and␣
 ↪layers of the neural network
```

```
    session.run(tf.global_variables_initializer())

    for epoch in range(n_epochs):
        for X_batch, y_batch in shuffle_batch(X_train, y_train, batch_size):
        # Feed in the training data and do one step of neural network training#
            session.run(optimizer, feed_dict={X: X_batch, y: y_batch})

        if epoch % 5 == 0:
            train_loss2 = session.run(loss, feed_dict={X: X_batch, y:y_batch})
            train_accuracy2 = accuracy.eval(feed_dict={X: X_batch, y: y_batch})
            print("Epoch: {}, Loss: {}, Accuracy: {}".
↪format(epoch,train_loss2,train_accuracy2))

        # Print the current training status to the screen
        print("Training pass: {}".format(epoch))

    test_accuracy2 = accuracy.eval(feed_dict={X: X_test, y: y_test})
    print("Test accuracy: {}".format(test_accuracy2))
    # Training is now complete!
    print("Training is complete!")
    t1 = time.time()
    print("Time to train and evaluate model: {:.3f} seconds".format(t1-t0))
    total_time = t1 - t0

time2 = total_time
```

```
Epoch: 0, Loss: 0.6193461418151855, Accuracy: 0.8199999928474426
Training pass: 0
Training pass: 1
Training pass: 2
Training pass: 3
Training pass: 4
Epoch: 5, Loss: 0.41975173354148865, Accuracy: 0.8199999928474426
Training pass: 5
Training pass: 6
Training pass: 7
Training pass: 8
Training pass: 9
Epoch: 10, Loss: 0.17170919477939606, Accuracy: 0.9800000190734863
Training pass: 10
Training pass: 11
Training pass: 12
Training pass: 13
Training pass: 14
Epoch: 15, Loss: 0.18721885979175568, Accuracy: 0.9200000166893005
Training pass: 15
Training pass: 16
```

```
Training pass: 17
Training pass: 18
Training pass: 19
Epoch: 20, Loss: 0.1632814258337021, Accuracy: 0.9200000166893005
Training pass: 20
Training pass: 21
Training pass: 22
Training pass: 23
Training pass: 24
Epoch: 25, Loss: 0.15139424800872803, Accuracy: 0.9599999785423279
Training pass: 25
Training pass: 26
Training pass: 27
Training pass: 28
Training pass: 29
Epoch: 30, Loss: 0.07083943486213684, Accuracy: 1.0
Training pass: 30
Training pass: 31
Training pass: 32
Training pass: 33
Training pass: 34
Epoch: 35, Loss: 0.31635868549346924, Accuracy: 0.9200000166893005
Training pass: 35
Training pass: 36
Training pass: 37
Training pass: 38
Training pass: 39
Test accuracy: 0.9383999705314636
Training is complete!
Time to train and evaluate model: 50.065 seconds
```

1.6 Model with 5 layers and 10 nodes per layer

[8]:
```
#reset graph to remove duplicate nodes
tf.reset_default_graph()

n_hidden1 = 10
n_hidden2 = 10
n_hidden3 = 10
n_hidden4 = 10
n_hidden5 = 10

# create placeholders for variables
X = tf.placeholder(tf.float32, shape=(None, n_inputs), name="X")
y = tf.placeholder(tf.int32, shape=(None), name="y")

with tf.name_scope("dnn"):
```

```
    hidden1 = tf.layers.dense(X, n_hidden1, name="hidden1", activation=tf.nn.
 ↪relu)
    hidden2 = tf.layers.dense(hidden1, n_hidden2, name="hidden2", activation=tf.
 ↪nn.relu)
    hidden3 = tf.layers.dense(hidden2, n_hidden3, name="hidden3", activation=tf.
 ↪nn.relu)
    hidden4 = tf.layers.dense(hidden3, n_hidden4, name="hidden4", activation=tf.
 ↪nn.relu)
    hidden5 = tf.layers.dense(hidden4, n_hidden5, name="hidden5", activation=tf.
 ↪nn.relu)
    logits = tf.layers.dense(hidden5, n_outputs, name="outputs")

# define cost function
with tf.name_scope("loss"):
    xentropy = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y,␣
 ↪logits=logits)
    loss = tf.reduce_mean(xentropy, name="loss")


# use optimizer to modify parameters to minimize cost function
learning_rate = 0.01

with tf.name_scope("train"):
    optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss)
#    training_op = optimizer.minimize(loss)

# evaluate the model using accuracy
with tf.name_scope("eval"):
    correct = tf.nn.in_top_k(logits, y, 1)
    accuracy = tf.reduce_mean(tf.cast(correct, tf.float32))

with tf.Session() as session:

    t0 = time.time()
    # Run the global variable initializer to initialize all variables and␣
 ↪layers of the neural network
    session.run(tf.global_variables_initializer())

    for epoch in range(n_epochs):
        for X_batch, y_batch in shuffle_batch(X_train, y_train, batch_size):
        # Feed in the training data and do one step of neural network training#
            session.run(optimizer, feed_dict={X: X_batch, y: y_batch})

        if epoch % 5 == 0:
            train_loss3 = session.run(loss, feed_dict={X: X_batch, y:y_batch})
            train_accuracy3 = accuracy.eval(feed_dict={X: X_batch, y: y_batch})
```

```
            print("Epoch: {}, Loss: {}, Accuracy: {}".
 ↪format(epoch,train_loss3,train_accuracy3))

        # Print the current training status to the screen
        print("Training pass: {}".format(epoch))

    test_accuracy3 = accuracy.eval(feed_dict={X: X_test, y: y_test})
    print("Test accuracy: {}".format(test_accuracy3))
    # Training is now complete!
    print("Training is complete!")
    t1 = time.time()
    print("Time to train and evaluate model: {:.3f} seconds".format(t1-t0))
    total_time = t1 - t0

time3 = total_time
```

```
Epoch: 0, Loss: 1.1670645475387573, Accuracy: 0.6600000262260437
Training pass: 0
Training pass: 1
Training pass: 2
Training pass: 3
Training pass: 4
Epoch: 5, Loss: 0.37619295716285706, Accuracy: 0.8999999761581421
Training pass: 5
Training pass: 6
Training pass: 7
Training pass: 8
Training pass: 9
Epoch: 10, Loss: 0.18869760632514954, Accuracy: 0.9599999785423279
Training pass: 10
Training pass: 11
Training pass: 12
Training pass: 13
Training pass: 14
Epoch: 15, Loss: 0.15033932030200958, Accuracy: 0.9399999976158142
Training pass: 15
Training pass: 16
Training pass: 17
Training pass: 18
Training pass: 19
Epoch: 20, Loss: 0.09303572028875351, Accuracy: 0.9800000190734863
Training pass: 20
Training pass: 21
Training pass: 22
Training pass: 23
Training pass: 24
Epoch: 25, Loss: 0.11709718406200409, Accuracy: 0.9599999785423279
```

```
Training pass: 25
Training pass: 26
Training pass: 27
Training pass: 28
Training pass: 29
Epoch: 30, Loss: 0.044970039278268814, Accuracy: 1.0
Training pass: 30
Training pass: 31
Training pass: 32
Training pass: 33
Training pass: 34
Epoch: 35, Loss: 0.13280817866325378, Accuracy: 0.9599999785423279
Training pass: 35
Training pass: 36
Training pass: 37
Training pass: 38
Training pass: 39
Test accuracy: 0.9332000017166138
Training is complete!
Time to train and evaluate model: 58.981 seconds
```

1.7 Model with 5 layers and 20 nodes per layer

[9]:
```
n_hidden1 = 20
n_hidden2 = 20
n_hidden3 = 20
n_hidden4 = 20
n_hidden5 = 20

with tf.Session() as session:

    t0 = time.time()
    # Run the global variable initializer to initialize all variables and␣
 ↪layers of the neural network
    session.run(tf.global_variables_initializer())

    for epoch in range(n_epochs):
        for X_batch, y_batch in shuffle_batch(X_train, y_train, batch_size):
        # Feed in the training data and do one step of neural network training#
            session.run(optimizer, feed_dict={X: X_batch, y: y_batch})

        if epoch % 5 == 0:
            train_loss4 = session.run(loss, feed_dict={X: X_batch, y:y_batch})
            train_accuracy4 = accuracy.eval(feed_dict={X: X_batch, y: y_batch})
            print("Epoch: {}, Loss: {}, Accuracy: {}".
 ↪format(epoch,train_loss4,train_accuracy4))
```

```
        # Print the current training status to the screen
        print("Training pass: {}".format(epoch))

    test_accuracy4 = accuracy.eval(feed_dict={X: X_test, y: y_test})
    print("Test accuracy: {}".format(test_accuracy4))
    # Training is now complete!
    print("Training is complete!")
    t1 = time.time()
    print("Time to train and evaluate model: {:.3f} seconds".format(t1-t0))
    total_time = t1 - t0

time4 = total_time
```

```
Epoch: 0, Loss: 1.6826581954956055, Accuracy: 0.41999998688697815
Training pass: 0
Training pass: 1
Training pass: 2
Training pass: 3
Training pass: 4
Epoch: 5, Loss: 0.24401473999023438, Accuracy: 0.9200000166893005
Training pass: 5
Training pass: 6
Training pass: 7
Training pass: 8
Training pass: 9
Epoch: 10, Loss: 0.06349951773881912, Accuracy: 1.0
Training pass: 10
Training pass: 11
Training pass: 12
Training pass: 13
Training pass: 14
Epoch: 15, Loss: 0.07275746762752533, Accuracy: 1.0
Training pass: 15
Training pass: 16
Training pass: 17
Training pass: 18
Training pass: 19
Epoch: 20, Loss: 0.18821677565574646, Accuracy: 0.9599999785423279
Training pass: 20
Training pass: 21
Training pass: 22
Training pass: 23
Training pass: 24
Epoch: 25, Loss: 0.21585316956043243, Accuracy: 0.9399999976158142
Training pass: 25
Training pass: 26
Training pass: 27
```

```
Training pass: 28
Training pass: 29
Epoch: 30, Loss: 0.05702674388885498, Accuracy: 1.0
Training pass: 30
Training pass: 31
Training pass: 32
Training pass: 33
Training pass: 34
Epoch: 35, Loss: 0.1931140124797821, Accuracy: 0.9800000190734863
Training pass: 35
Training pass: 36
Training pass: 37
Training pass: 38
Training pass: 39
Test accuracy: 0.9318000078201294
Training is complete!
Time to train and evaluate model: 58.823 seconds
```

1.8 Summary of model performance with Tensorflow

[10]:
```
summary_models = {
    'Number of Layers' : [2, 2, 5, 5],
    'Number of Nodes/Layer' : [10, 20, 10, 20],
    'Processing Time' : [round(time1, 3), round(time2,3), round(time3,3),␣
 ↪round(time4,3)],
    'Train Accuracy' : [round(train_accuracy, 3), round(train_accuracy2, 3),␣
 ↪round(train_accuracy3, 3), round(train_accuracy4,3)],
    'Test Accruacy' : [round(test_accuracy, 3), round(test_accuracy2, 3),␣
 ↪round(test_accuracy3, 3), round(test_accuracy4, 3)]
}
```

[11]:
```
import pandas as pd
summary_models_df = pd.DataFrame(summary_models)
summary_models_df
```

[11]:

	Number of Layers	Number of Nodes/Layer	Processing Time	Train Accuracy	Test Accruacy
0	2	10	44.758	0.94	0.935
1	2	20	50.065	0.92	0.938
2	5	10	58.981	0.96	0.933
3	5	20	58.823	0.98	0.932

2 Tensorflow Premade Estimator: DNNClassifier

2.1 Import libraries

[12]:
```
import numpy as np
import tensorflow as tf
from tensorflow.contrib import learn
import time
```

2.2 Data: import, prepare, reshape and normalize

[13]:
```
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
X_train = X_train.astype(np.float32).reshape(-1,28*28)/255.0
X_test = X_test.astype(np.float32).reshape(-1,28*28)/255.0
y_train = y_train.astype(np.int32)
y_test = y_test.astype(np.int32)
```

2.3 Model: setup, train, and evaluation

[14]:
```
#reset graph to remove duplicate nodes
tf.reset_default_graph()

#define key parameters for model
epochs = 40
BATCH_SIZE = 50
n_inputs = 784
```

[15]:
```
#https://www.tensorflow.org/guide/premade_estimators
#A feature column is an object describing how the model should
#use raw input data from the features dictionary.
#When you build an Estimator model, you pass it a list of
#feature columns that describes each of the features you want the model to use.

#https://github.com/ageron/handson-ml/blob/master/
 ↪10_introduction_to_artificial_neural_networks.ipynb
feature_cols = [tf.feature_column.numeric_column("X", shape=[28 * 28])]
```

[16]:
```
# use optimizer to modify parameters to minimize cost function
learning_rate = 0.01
optimizer = tf.train.GradientDescentOptimizer(learning_rate)
```

[17]:
```
#Instantiate an estimator
classifier = tf.estimator.DNNClassifier(
    feature_columns=feature_cols,
    # Two hidden layers and each number represents the number of nodes
    hidden_units=[128, 32],
    # The model must choose between 10 classes
```

```
    n_classes=10,
    optimizer = optimizer)
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory:
/var/folders/5s/f9dl19z88xjgbj0059_sl_r00000gn/T/tmprv4yj7le
INFO:tensorflow:Using config: {'_model_dir':
'/var/folders/5s/f9dl19z88xjgbj0059_sl_r00000gn/T/tmprv4yj7le',
'_tf_random_seed': None, '_save_summary_steps': 100, '_save_checkpoints_steps':
None, '_save_checkpoints_secs': 600, '_session_config': allow_soft_placement:
true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000,
'_log_step_count_steps': 100, '_train_distribute': None, '_device_fn': None,
'_protocol': None, '_eval_distribute': None, '_experimental_distribute': None,
'_service': None, '_cluster_spec':
<tensorflow.python.training.server_lib.ClusterSpec object at 0x1a518ce470>,
'_task_type': 'worker', '_task_id': 0, '_global_id_in_cluster': 0, '_master':
'', '_evaluation_master': '', '_is_chief': True, '_num_ps_replicas': 0,
'_num_worker_replicas': 1}
```

[18]:
```
#A function that constructs the input data for training.
input_function = tf.estimator.inputs.numpy_input_fn(
    x = {"X" : X_train},
    y = y_train,
    batch_size=BATCH_SIZE,
    num_epochs=epochs,
    shuffle=True,
    queue_capacity=1000,
    num_threads=1)

#A function that constructs the input data for evaluation.
#this is for model evaluation
input_fn_test = tf.estimator.inputs.numpy_input_fn(
    x={"X": X_test},
    y=y_test,
    shuffle=False)
```

2.4 DNNClassifier Model with 2 layers and 128 and 32 nodes

[19]:
```
#Start timing for the model train/test
t0 = time.time()

# Train the Model
classifier.train(input_fn=input_function)

#Evaluate the test data sets with the model

evaluate_metrics = classifier.evaluate(input_fn=input_fn_test)

#Finish timing the model train/test
t1 = time.time()

#print the test accuracy
print('Test accuracy: {}'.format(evaluate_metrics['accuracy']))
print("Time to train and evaluate model: {:.3f} seconds".format(t1-t0))

total_time = t1 - t0
time5 = total_time
```

```
WARNING:tensorflow:From /Users/jmwanat/anaconda3/envs/tf/lib/python3.7/site-pack
ages/tensorflow_estimator/python/estimator/inputs/queues/feeding_queue_runner.py
:62: QueueRunner.__init__ (from tensorflow.python.training.queue_runner_impl) is
deprecated and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
WARNING:tensorflow:From /Users/jmwanat/anaconda3/envs/tf/lib/python3.7/site-pack
ages/tensorflow_estimator/python/estimator/inputs/queues/feeding_functions.py:50
0: add_queue_runner (from tensorflow.python.training.queue_runner_impl) is
deprecated and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Calling model_fn.
WARNING:tensorflow:From /Users/jmwanat/anaconda3/envs/tf/lib/python3.7/site-
packages/tensorflow/python/feature_column/feature_column_v2.py:2703: to_float
(from tensorflow.python.ops.math_ops) is deprecated and will be removed in a
future version.
Instructions for updating:
Use tf.cast instead.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
WARNING:tensorflow:From /Users/jmwanat/anaconda3/envs/tf/lib/python3.7/site-
packages/tensorflow/python/training/monitored_session.py:809:
```

```
start_queue_runners (from tensorflow.python.training.queue_runner_impl) is
deprecated and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Saving checkpoints for 0 into
/var/folders/5s/f9dl19z88xjgbj0059_sl_r00000gn/T/tmprv4yj7le/model.ckpt.
INFO:tensorflow:loss = 118.76428, step = 1
INFO:tensorflow:global_step/sec: 251.681
INFO:tensorflow:loss = 17.09964, step = 101 (0.398 sec)
INFO:tensorflow:global_step/sec: 384.839
INFO:tensorflow:loss = 16.45511, step = 201 (0.260 sec)
INFO:tensorflow:global_step/sec: 390.823
INFO:tensorflow:loss = 9.35091, step = 301 (0.256 sec)
INFO:tensorflow:global_step/sec: 393.707
INFO:tensorflow:loss = 13.502039, step = 401 (0.254 sec)
INFO:tensorflow:global_step/sec: 389.296
INFO:tensorflow:loss = 6.1646566, step = 501 (0.257 sec)
INFO:tensorflow:global_step/sec: 250.227
INFO:tensorflow:loss = 14.8763075, step = 601 (0.400 sec)
INFO:tensorflow:global_step/sec: 398.357
INFO:tensorflow:loss = 7.6176276, step = 701 (0.251 sec)
INFO:tensorflow:global_step/sec: 391.785
INFO:tensorflow:loss = 2.543083, step = 801 (0.255 sec)
INFO:tensorflow:global_step/sec: 377.334
INFO:tensorflow:loss = 8.507083, step = 901 (0.265 sec)
INFO:tensorflow:global_step/sec: 368.383
INFO:tensorflow:loss = 5.7035956, step = 1001 (0.271 sec)
INFO:tensorflow:global_step/sec: 393.408
INFO:tensorflow:loss = 2.931604, step = 1101 (0.254 sec)
INFO:tensorflow:global_step/sec: 253.2
INFO:tensorflow:loss = 5.593193, step = 1201 (0.395 sec)
INFO:tensorflow:global_step/sec: 395.651
INFO:tensorflow:loss = 8.267117, step = 1301 (0.253 sec)
INFO:tensorflow:global_step/sec: 396.09
INFO:tensorflow:loss = 2.0176091, step = 1401 (0.252 sec)
INFO:tensorflow:global_step/sec: 395.004
INFO:tensorflow:loss = 11.08103, step = 1501 (0.253 sec)
INFO:tensorflow:global_step/sec: 395.192
INFO:tensorflow:loss = 6.028353, step = 1601 (0.253 sec)
INFO:tensorflow:global_step/sec: 400.262
INFO:tensorflow:loss = 7.617423, step = 1701 (0.250 sec)
INFO:tensorflow:global_step/sec: 254.649
INFO:tensorflow:loss = 1.8847181, step = 1801 (0.393 sec)
INFO:tensorflow:global_step/sec: 394.741
INFO:tensorflow:loss = 6.759273, step = 1901 (0.253 sec)
INFO:tensorflow:global_step/sec: 402.113
INFO:tensorflow:loss = 3.7497807, step = 2001 (0.249 sec)
INFO:tensorflow:global_step/sec: 368.764
```

```
INFO:tensorflow:loss = 7.028991, step = 2101 (0.271 sec)
INFO:tensorflow:global_step/sec: 398.049
INFO:tensorflow:loss = 2.292898, step = 2201 (0.251 sec)
INFO:tensorflow:global_step/sec: 201.188
INFO:tensorflow:loss = 4.5321383, step = 2301 (0.497 sec)
INFO:tensorflow:global_step/sec: 362.095
INFO:tensorflow:loss = 4.3193803, step = 2401 (0.276 sec)
INFO:tensorflow:global_step/sec: 396.673
INFO:tensorflow:loss = 4.3176312, step = 2501 (0.252 sec)
INFO:tensorflow:global_step/sec: 398.656
INFO:tensorflow:loss = 19.427513, step = 2601 (0.251 sec)
INFO:tensorflow:global_step/sec: 392.711
INFO:tensorflow:loss = 1.4939963, step = 2701 (0.255 sec)
INFO:tensorflow:global_step/sec: 356.175
INFO:tensorflow:loss = 4.5341663, step = 2801 (0.281 sec)
INFO:tensorflow:global_step/sec: 222.071
INFO:tensorflow:loss = 0.423289, step = 2901 (0.450 sec)
INFO:tensorflow:global_step/sec: 377.648
INFO:tensorflow:loss = 2.0889826, step = 3001 (0.265 sec)
INFO:tensorflow:global_step/sec: 349.513
INFO:tensorflow:loss = 1.7537378, step = 3101 (0.286 sec)
INFO:tensorflow:global_step/sec: 380.302
INFO:tensorflow:loss = 11.714243, step = 3201 (0.263 sec)
INFO:tensorflow:global_step/sec: 400.614
INFO:tensorflow:loss = 2.9650261, step = 3301 (0.250 sec)
INFO:tensorflow:global_step/sec: 244.633
INFO:tensorflow:loss = 6.346114, step = 3401 (0.409 sec)
INFO:tensorflow:global_step/sec: 400.074
INFO:tensorflow:loss = 7.0580926, step = 3501 (0.250 sec)
INFO:tensorflow:global_step/sec: 402.902
INFO:tensorflow:loss = 2.4554636, step = 3601 (0.248 sec)
INFO:tensorflow:global_step/sec: 403.07
INFO:tensorflow:loss = 2.2096367, step = 3701 (0.248 sec)
INFO:tensorflow:global_step/sec: 391.911
INFO:tensorflow:loss = 4.511342, step = 3801 (0.255 sec)
INFO:tensorflow:global_step/sec: 388.52
INFO:tensorflow:loss = 3.570749, step = 3901 (0.257 sec)
INFO:tensorflow:global_step/sec: 248.755
INFO:tensorflow:loss = 1.3077859, step = 4001 (0.402 sec)
INFO:tensorflow:global_step/sec: 391.939
INFO:tensorflow:loss = 2.9412904, step = 4101 (0.255 sec)
INFO:tensorflow:global_step/sec: 398.281
INFO:tensorflow:loss = 10.12344, step = 4201 (0.251 sec)
INFO:tensorflow:global_step/sec: 316.579
INFO:tensorflow:loss = 6.2518497, step = 4301 (0.316 sec)
INFO:tensorflow:global_step/sec: 320.741
INFO:tensorflow:loss = 5.017774, step = 4401 (0.311 sec)
INFO:tensorflow:global_step/sec: 222.984
```

```
INFO:tensorflow:loss = 3.57402, step = 4501 (0.449 sec)
INFO:tensorflow:global_step/sec: 330.745
INFO:tensorflow:loss = 7.169818, step = 4601 (0.302 sec)
INFO:tensorflow:global_step/sec: 405.788
INFO:tensorflow:loss = 2.2278738, step = 4701 (0.246 sec)
INFO:tensorflow:global_step/sec: 381.426
INFO:tensorflow:loss = 1.8180633, step = 4801 (0.262 sec)
INFO:tensorflow:global_step/sec: 318.721
INFO:tensorflow:loss = 2.3576565, step = 4901 (0.314 sec)
INFO:tensorflow:global_step/sec: 400.617
INFO:tensorflow:loss = 6.147632, step = 5001 (0.249 sec)
INFO:tensorflow:global_step/sec: 224.736
INFO:tensorflow:loss = 4.4024186, step = 5101 (0.445 sec)
INFO:tensorflow:global_step/sec: 369.417
INFO:tensorflow:loss = 0.66626346, step = 5201 (0.271 sec)
INFO:tensorflow:global_step/sec: 403.587
INFO:tensorflow:loss = 3.644599, step = 5301 (0.248 sec)
INFO:tensorflow:global_step/sec: 412.261
INFO:tensorflow:loss = 0.2749519, step = 5401 (0.243 sec)
INFO:tensorflow:global_step/sec: 407.877
INFO:tensorflow:loss = 6.4694214, step = 5501 (0.245 sec)
INFO:tensorflow:global_step/sec: 359.441
INFO:tensorflow:loss = 3.3823988, step = 5601 (0.278 sec)
INFO:tensorflow:global_step/sec: 237.161
INFO:tensorflow:loss = 1.6316897, step = 5701 (0.422 sec)
INFO:tensorflow:global_step/sec: 398.256
INFO:tensorflow:loss = 3.993641, step = 5801 (0.251 sec)
INFO:tensorflow:global_step/sec: 362.535
INFO:tensorflow:loss = 2.3267453, step = 5901 (0.276 sec)
INFO:tensorflow:global_step/sec: 390.105
INFO:tensorflow:loss = 2.0570023, step = 6001 (0.256 sec)
INFO:tensorflow:global_step/sec: 401.522
INFO:tensorflow:loss = 3.6044338, step = 6101 (0.249 sec)
INFO:tensorflow:global_step/sec: 251.707
INFO:tensorflow:loss = 1.9370332, step = 6201 (0.397 sec)
INFO:tensorflow:global_step/sec: 388.297
INFO:tensorflow:loss = 5.6907663, step = 6301 (0.258 sec)
INFO:tensorflow:global_step/sec: 404.537
INFO:tensorflow:loss = 1.640714, step = 6401 (0.247 sec)
INFO:tensorflow:global_step/sec: 408.921
INFO:tensorflow:loss = 2.0861967, step = 6501 (0.244 sec)
INFO:tensorflow:global_step/sec: 374.459
INFO:tensorflow:loss = 2.7225478, step = 6601 (0.267 sec)
INFO:tensorflow:global_step/sec: 400.211
INFO:tensorflow:loss = 5.489012, step = 6701 (0.250 sec)
INFO:tensorflow:global_step/sec: 250.857
INFO:tensorflow:loss = 1.1617343, step = 6801 (0.399 sec)
INFO:tensorflow:global_step/sec: 405.601
```

```
INFO:tensorflow:loss = 10.951439, step = 6901 (0.247 sec)
INFO:tensorflow:global_step/sec: 404.298
INFO:tensorflow:loss = 1.2771016, step = 7001 (0.247 sec)
INFO:tensorflow:global_step/sec: 414.112
INFO:tensorflow:loss = 2.23488, step = 7101 (0.242 sec)
INFO:tensorflow:global_step/sec: 406.005
INFO:tensorflow:loss = 0.64209545, step = 7201 (0.246 sec)
INFO:tensorflow:global_step/sec: 403.198
INFO:tensorflow:loss = 0.3127751, step = 7301 (0.248 sec)
INFO:tensorflow:global_step/sec: 245.851
INFO:tensorflow:loss = 0.89203864, step = 7401 (0.407 sec)
INFO:tensorflow:global_step/sec: 410.806
INFO:tensorflow:loss = 0.9395554, step = 7501 (0.243 sec)
INFO:tensorflow:global_step/sec: 411.609
INFO:tensorflow:loss = 1.3646529, step = 7601 (0.243 sec)
INFO:tensorflow:global_step/sec: 339.355
INFO:tensorflow:loss = 7.59174, step = 7701 (0.297 sec)
INFO:tensorflow:global_step/sec: 312.216
INFO:tensorflow:loss = 1.6341159, step = 7801 (0.318 sec)
INFO:tensorflow:global_step/sec: 329.325
INFO:tensorflow:loss = 3.1952024, step = 7901 (0.304 sec)
INFO:tensorflow:global_step/sec: 248.245
INFO:tensorflow:loss = 3.5883088, step = 8001 (0.403 sec)
INFO:tensorflow:global_step/sec: 362.986
INFO:tensorflow:loss = 0.13352531, step = 8101 (0.275 sec)
INFO:tensorflow:global_step/sec: 388.512
INFO:tensorflow:loss = 0.84247124, step = 8201 (0.257 sec)
INFO:tensorflow:global_step/sec: 318.809
INFO:tensorflow:loss = 3.0789895, step = 8301 (0.314 sec)
INFO:tensorflow:global_step/sec: 317.804
INFO:tensorflow:loss = 0.05045923, step = 8401 (0.315 sec)
INFO:tensorflow:global_step/sec: 218.429
INFO:tensorflow:loss = 0.42555767, step = 8501 (0.458 sec)
INFO:tensorflow:global_step/sec: 325.193
INFO:tensorflow:loss = 3.9581342, step = 8601 (0.308 sec)
INFO:tensorflow:global_step/sec: 324.555
INFO:tensorflow:loss = 0.27723664, step = 8701 (0.308 sec)
INFO:tensorflow:global_step/sec: 357.411
INFO:tensorflow:loss = 0.03453445, step = 8801 (0.280 sec)
INFO:tensorflow:global_step/sec: 407.677
INFO:tensorflow:loss = 0.07690896, step = 8901 (0.245 sec)
INFO:tensorflow:global_step/sec: 239.1
INFO:tensorflow:loss = 2.3270397, step = 9001 (0.419 sec)
INFO:tensorflow:global_step/sec: 325.089
INFO:tensorflow:loss = 7.6445765, step = 9101 (0.307 sec)
INFO:tensorflow:global_step/sec: 326.341
INFO:tensorflow:loss = 10.388578, step = 9201 (0.306 sec)
INFO:tensorflow:global_step/sec: 331.627
```

```
INFO:tensorflow:loss = 0.13850689, step = 9301 (0.302 sec)
INFO:tensorflow:global_step/sec: 314.703
INFO:tensorflow:loss = 2.1301527, step = 9401 (0.318 sec)
INFO:tensorflow:global_step/sec: 220.5
INFO:tensorflow:loss = 0.47067863, step = 9501 (0.453 sec)
INFO:tensorflow:global_step/sec: 376.152
INFO:tensorflow:loss = 2.0725577, step = 9601 (0.266 sec)
INFO:tensorflow:global_step/sec: 349.456
INFO:tensorflow:loss = 5.390092, step = 9701 (0.286 sec)
INFO:tensorflow:global_step/sec: 284.853
INFO:tensorflow:loss = 0.49072325, step = 9801 (0.351 sec)
INFO:tensorflow:global_step/sec: 332.226
INFO:tensorflow:loss = 0.2393972, step = 9901 (0.301 sec)
INFO:tensorflow:global_step/sec: 225.403
INFO:tensorflow:loss = 0.70973283, step = 10001 (0.444 sec)
INFO:tensorflow:global_step/sec: 376.928
INFO:tensorflow:loss = 3.3582191, step = 10101 (0.265 sec)
INFO:tensorflow:global_step/sec: 204.45
INFO:tensorflow:loss = 0.08662148, step = 10201 (0.489 sec)
INFO:tensorflow:global_step/sec: 242.761
INFO:tensorflow:loss = 1.94311, step = 10301 (0.412 sec)
INFO:tensorflow:global_step/sec: 226.087
INFO:tensorflow:loss = 0.34580424, step = 10401 (0.442 sec)
INFO:tensorflow:global_step/sec: 422.896
INFO:tensorflow:loss = 5.1912923, step = 10501 (0.237 sec)
INFO:tensorflow:global_step/sec: 415.23
INFO:tensorflow:loss = 2.6176813, step = 10601 (0.240 sec)
INFO:tensorflow:global_step/sec: 406.161
INFO:tensorflow:loss = 0.059388675, step = 10701 (0.246 sec)
INFO:tensorflow:global_step/sec: 386.995
INFO:tensorflow:loss = 3.5993102, step = 10801 (0.258 sec)
INFO:tensorflow:global_step/sec: 419.594
INFO:tensorflow:loss = 1.4986124, step = 10901 (0.238 sec)
INFO:tensorflow:global_step/sec: 255.95
INFO:tensorflow:loss = 5.2115374, step = 11001 (0.391 sec)
INFO:tensorflow:global_step/sec: 382.004
INFO:tensorflow:loss = 0.80451065, step = 11101 (0.263 sec)
INFO:tensorflow:global_step/sec: 389.44
INFO:tensorflow:loss = 0.9666449, step = 11201 (0.256 sec)
INFO:tensorflow:global_step/sec: 397.277
INFO:tensorflow:loss = 0.0060063363, step = 11301 (0.252 sec)
INFO:tensorflow:global_step/sec: 418.615
INFO:tensorflow:loss = 7.026033, step = 11401 (0.239 sec)
INFO:tensorflow:global_step/sec: 387.591
INFO:tensorflow:loss = 1.0108013, step = 11501 (0.258 sec)
INFO:tensorflow:global_step/sec: 252.154
INFO:tensorflow:loss = 1.2242577, step = 11601 (0.397 sec)
INFO:tensorflow:global_step/sec: 422.871
```

```
INFO:tensorflow:loss = 4.992698, step = 11701 (0.236 sec)
INFO:tensorflow:global_step/sec: 420.599
INFO:tensorflow:loss = 0.12708662, step = 11801 (0.238 sec)
INFO:tensorflow:global_step/sec: 411.21
INFO:tensorflow:loss = 0.6929489, step = 11901 (0.243 sec)
INFO:tensorflow:global_step/sec: 411.335
INFO:tensorflow:loss = 0.43014222, step = 12001 (0.243 sec)
INFO:tensorflow:global_step/sec: 415.64
INFO:tensorflow:loss = 0.29651245, step = 12101 (0.241 sec)
INFO:tensorflow:global_step/sec: 248.126
INFO:tensorflow:loss = 1.4386566, step = 12201 (0.403 sec)
INFO:tensorflow:global_step/sec: 381.669
INFO:tensorflow:loss = 0.61943465, step = 12301 (0.262 sec)
INFO:tensorflow:global_step/sec: 405.071
INFO:tensorflow:loss = 7.410125, step = 12401 (0.247 sec)
INFO:tensorflow:global_step/sec: 396.77
INFO:tensorflow:loss = 0.6587486, step = 12501 (0.252 sec)
INFO:tensorflow:global_step/sec: 374.715
INFO:tensorflow:loss = 2.2452035, step = 12601 (0.267 sec)
INFO:tensorflow:global_step/sec: 387.468
INFO:tensorflow:loss = 1.9570531, step = 12701 (0.258 sec)
INFO:tensorflow:global_step/sec: 247.491
INFO:tensorflow:loss = 0.41404518, step = 12801 (0.404 sec)
INFO:tensorflow:global_step/sec: 389.545
INFO:tensorflow:loss = 1.6571838, step = 12901 (0.257 sec)
INFO:tensorflow:global_step/sec: 382.725
INFO:tensorflow:loss = 0.4618814, step = 13001 (0.261 sec)
INFO:tensorflow:global_step/sec: 309.577
INFO:tensorflow:loss = 6.609779, step = 13101 (0.323 sec)
INFO:tensorflow:global_step/sec: 343.972
INFO:tensorflow:loss = 0.18097413, step = 13201 (0.291 sec)
INFO:tensorflow:global_step/sec: 134.858
INFO:tensorflow:loss = 0.11767666, step = 13301 (0.742 sec)
INFO:tensorflow:global_step/sec: 244.376
INFO:tensorflow:loss = 0.18982166, step = 13401 (0.409 sec)
INFO:tensorflow:global_step/sec: 303.997
INFO:tensorflow:loss = 0.29746312, step = 13501 (0.331 sec)
INFO:tensorflow:global_step/sec: 319.483
INFO:tensorflow:loss = 7.3497615, step = 13601 (0.311 sec)
INFO:tensorflow:global_step/sec: 236.565
INFO:tensorflow:loss = 0.46301565, step = 13701 (0.422 sec)
INFO:tensorflow:global_step/sec: 407.269
INFO:tensorflow:loss = 0.39029408, step = 13801 (0.246 sec)
INFO:tensorflow:global_step/sec: 389.079
INFO:tensorflow:loss = 0.5778161, step = 13901 (0.257 sec)
INFO:tensorflow:global_step/sec: 351.889
INFO:tensorflow:loss = 0.06956926, step = 14001 (0.284 sec)
INFO:tensorflow:global_step/sec: 283.328
```

```
INFO:tensorflow:loss = 0.03140445, step = 14101 (0.353 sec)
INFO:tensorflow:global_step/sec: 197.121
INFO:tensorflow:loss = 5.312543, step = 14201 (0.507 sec)
INFO:tensorflow:global_step/sec: 400.846
INFO:tensorflow:loss = 2.6628466, step = 14301 (0.249 sec)
INFO:tensorflow:global_step/sec: 372.499
INFO:tensorflow:loss = 7.172105, step = 14401 (0.269 sec)
INFO:tensorflow:global_step/sec: 362.313
INFO:tensorflow:loss = 0.09079447, step = 14501 (0.278 sec)
INFO:tensorflow:global_step/sec: 225.246
INFO:tensorflow:loss = 2.1453846, step = 14601 (0.442 sec)
INFO:tensorflow:global_step/sec: 247.711
INFO:tensorflow:loss = 0.06689043, step = 14701 (0.404 sec)
INFO:tensorflow:global_step/sec: 329.578
INFO:tensorflow:loss = 0.7950106, step = 14801 (0.303 sec)
INFO:tensorflow:global_step/sec: 375.899
INFO:tensorflow:loss = 2.2633219, step = 14901 (0.266 sec)
INFO:tensorflow:global_step/sec: 395.809
INFO:tensorflow:loss = 2.1490197, step = 15001 (0.253 sec)
INFO:tensorflow:global_step/sec: 418.326
INFO:tensorflow:loss = 0.43245536, step = 15101 (0.239 sec)
INFO:tensorflow:global_step/sec: 244.228
INFO:tensorflow:loss = 1.3981624, step = 15201 (0.410 sec)
INFO:tensorflow:global_step/sec: 390.236
INFO:tensorflow:loss = 0.4949594, step = 15301 (0.256 sec)
INFO:tensorflow:global_step/sec: 396.3
INFO:tensorflow:loss = 0.027656555, step = 15401 (0.252 sec)
INFO:tensorflow:global_step/sec: 331.717
INFO:tensorflow:loss = 0.83661014, step = 15501 (0.301 sec)
INFO:tensorflow:global_step/sec: 366.07
INFO:tensorflow:loss = 0.022737639, step = 15601 (0.273 sec)
INFO:tensorflow:global_step/sec: 388.371
INFO:tensorflow:loss = 0.39254108, step = 15701 (0.257 sec)
INFO:tensorflow:global_step/sec: 248.889
INFO:tensorflow:loss = 0.81200194, step = 15801 (0.402 sec)
INFO:tensorflow:global_step/sec: 388.317
INFO:tensorflow:loss = 0.047483135, step = 15901 (0.258 sec)
INFO:tensorflow:global_step/sec: 381.259
INFO:tensorflow:loss = 0.11630434, step = 16001 (0.262 sec)
INFO:tensorflow:global_step/sec: 379.713
INFO:tensorflow:loss = 4.760697, step = 16101 (0.263 sec)
INFO:tensorflow:global_step/sec: 409.643
INFO:tensorflow:loss = 1.1894352, step = 16201 (0.244 sec)
INFO:tensorflow:global_step/sec: 408.694
INFO:tensorflow:loss = 0.65450037, step = 16301 (0.245 sec)
INFO:tensorflow:global_step/sec: 251.835
INFO:tensorflow:loss = 0.020233843, step = 16401 (0.397 sec)
INFO:tensorflow:global_step/sec: 398.952
```

```
INFO:tensorflow:loss = 1.2722898, step = 16501 (0.250 sec)
INFO:tensorflow:global_step/sec: 400.048
INFO:tensorflow:loss = 3.7937074, step = 16601 (0.250 sec)
INFO:tensorflow:global_step/sec: 418.982
INFO:tensorflow:loss = 1.1155664, step = 16701 (0.239 sec)
INFO:tensorflow:global_step/sec: 400.671
INFO:tensorflow:loss = 0.088040955, step = 16801 (0.250 sec)
INFO:tensorflow:global_step/sec: 414.57
INFO:tensorflow:loss = 4.5796657, step = 16901 (0.241 sec)
INFO:tensorflow:global_step/sec: 252.226
INFO:tensorflow:loss = 0.13771424, step = 17001 (0.396 sec)
INFO:tensorflow:global_step/sec: 410.48
INFO:tensorflow:loss = 3.5320654, step = 17101 (0.244 sec)
INFO:tensorflow:global_step/sec: 401.218
INFO:tensorflow:loss = 0.08058581, step = 17201 (0.249 sec)
INFO:tensorflow:global_step/sec: 415.495
INFO:tensorflow:loss = 0.001857093, step = 17301 (0.241 sec)
INFO:tensorflow:global_step/sec: 417.726
INFO:tensorflow:loss = 1.6179838, step = 17401 (0.239 sec)
INFO:tensorflow:global_step/sec: 406.011
INFO:tensorflow:loss = 2.700485, step = 17501 (0.247 sec)
INFO:tensorflow:global_step/sec: 255.35
INFO:tensorflow:loss = 0.12077571, step = 17601 (0.391 sec)
INFO:tensorflow:global_step/sec: 422.899
INFO:tensorflow:loss = 0.2655803, step = 17701 (0.236 sec)
INFO:tensorflow:global_step/sec: 385.698
INFO:tensorflow:loss = 1.6435707, step = 17801 (0.259 sec)
INFO:tensorflow:global_step/sec: 365.595
INFO:tensorflow:loss = 0.35510844, step = 17901 (0.273 sec)
INFO:tensorflow:global_step/sec: 381.461
INFO:tensorflow:loss = 0.06860767, step = 18001 (0.262 sec)
INFO:tensorflow:global_step/sec: 332.622
INFO:tensorflow:loss = 0.10810786, step = 18101 (0.301 sec)
INFO:tensorflow:global_step/sec: 226.919
INFO:tensorflow:loss = 0.20862275, step = 18201 (0.440 sec)
INFO:tensorflow:global_step/sec: 421.804
INFO:tensorflow:loss = 0.25833854, step = 18301 (0.237 sec)
INFO:tensorflow:global_step/sec: 418.505
INFO:tensorflow:loss = 10.3858385, step = 18401 (0.239 sec)
INFO:tensorflow:global_step/sec: 417.233
INFO:tensorflow:loss = 0.019068345, step = 18501 (0.240 sec)
INFO:tensorflow:global_step/sec: 419.579
INFO:tensorflow:loss = 0.14354932, step = 18601 (0.238 sec)
INFO:tensorflow:global_step/sec: 419.985
INFO:tensorflow:loss = 0.7775321, step = 18701 (0.238 sec)
INFO:tensorflow:global_step/sec: 243.797
INFO:tensorflow:loss = 0.09586828, step = 18801 (0.410 sec)
INFO:tensorflow:global_step/sec: 419.308
```

```
INFO:tensorflow:loss = 0.02213536, step = 18901 (0.238 sec)
INFO:tensorflow:global_step/sec: 407.546
INFO:tensorflow:loss = 9.080235, step = 19001 (0.245 sec)
INFO:tensorflow:global_step/sec: 393.192
INFO:tensorflow:loss = 3.8733435, step = 19101 (0.254 sec)
INFO:tensorflow:global_step/sec: 381.029
INFO:tensorflow:loss = 9.1783695, step = 19201 (0.262 sec)
INFO:tensorflow:global_step/sec: 399.187
INFO:tensorflow:loss = 2.7842374, step = 19301 (0.251 sec)
INFO:tensorflow:global_step/sec: 252.306
INFO:tensorflow:loss = 0.13110559, step = 19401 (0.396 sec)
INFO:tensorflow:global_step/sec: 402.74
INFO:tensorflow:loss = 2.763809, step = 19501 (0.248 sec)
INFO:tensorflow:global_step/sec: 401.707
INFO:tensorflow:loss = 0.19408211, step = 19601 (0.249 sec)
INFO:tensorflow:global_step/sec: 378.246
INFO:tensorflow:loss = 0.011854961, step = 19701 (0.265 sec)
INFO:tensorflow:global_step/sec: 392.771
INFO:tensorflow:loss = 0.46772593, step = 19801 (0.254 sec)
INFO:tensorflow:global_step/sec: 252.834
INFO:tensorflow:loss = 4.485722, step = 19901 (0.396 sec)
INFO:tensorflow:global_step/sec: 373.335
INFO:tensorflow:loss = 1.7263113, step = 20001 (0.268 sec)
INFO:tensorflow:global_step/sec: 280.344
INFO:tensorflow:loss = 0.7193835, step = 20101 (0.357 sec)
INFO:tensorflow:global_step/sec: 323.305
INFO:tensorflow:loss = 3.594382, step = 20201 (0.309 sec)
INFO:tensorflow:global_step/sec: 266.702
INFO:tensorflow:loss = 0.12410071, step = 20301 (0.375 sec)
INFO:tensorflow:global_step/sec: 197.877
INFO:tensorflow:loss = 0.95647, step = 20401 (0.505 sec)
INFO:tensorflow:global_step/sec: 375.391
INFO:tensorflow:loss = 0.015142913, step = 20501 (0.266 sec)
INFO:tensorflow:global_step/sec: 351.289
INFO:tensorflow:loss = 0.20923084, step = 20601 (0.285 sec)
INFO:tensorflow:global_step/sec: 376.326
INFO:tensorflow:loss = 2.3171773, step = 20701 (0.266 sec)
INFO:tensorflow:global_step/sec: 387.064
INFO:tensorflow:loss = 0.43515417, step = 20801 (0.259 sec)
INFO:tensorflow:global_step/sec: 244.554
INFO:tensorflow:loss = 0.2405378, step = 20901 (0.408 sec)
INFO:tensorflow:global_step/sec: 416.615
INFO:tensorflow:loss = 0.32160679, step = 21001 (0.240 sec)
INFO:tensorflow:global_step/sec: 410.44
INFO:tensorflow:loss = 0.048900295, step = 21101 (0.244 sec)
INFO:tensorflow:global_step/sec: 375.318
INFO:tensorflow:loss = 4.5284123, step = 21201 (0.266 sec)
INFO:tensorflow:global_step/sec: 369.104
```

```
INFO:tensorflow:loss = 0.59161913, step = 21301 (0.271 sec)
INFO:tensorflow:global_step/sec: 379.114
INFO:tensorflow:loss = 0.14450797, step = 21401 (0.264 sec)
INFO:tensorflow:global_step/sec: 249.813
INFO:tensorflow:loss = 0.78696036, step = 21501 (0.400 sec)
INFO:tensorflow:global_step/sec: 201.915
INFO:tensorflow:loss = 0.04675384, step = 21601 (0.495 sec)
INFO:tensorflow:global_step/sec: 298.48
INFO:tensorflow:loss = 1.4901152, step = 21701 (0.335 sec)
INFO:tensorflow:global_step/sec: 371.74
INFO:tensorflow:loss = 0.0027745552, step = 21801 (0.269 sec)
INFO:tensorflow:global_step/sec: 376.152
INFO:tensorflow:loss = 1.0340079, step = 21901 (0.266 sec)
INFO:tensorflow:global_step/sec: 256.425
INFO:tensorflow:loss = 0.03231575, step = 22001 (0.390 sec)
INFO:tensorflow:global_step/sec: 353.499
INFO:tensorflow:loss = 4.4245777, step = 22101 (0.283 sec)
INFO:tensorflow:global_step/sec: 407.259
INFO:tensorflow:loss = 0.61166614, step = 22201 (0.246 sec)
INFO:tensorflow:global_step/sec: 392.548
INFO:tensorflow:loss = 3.1310668, step = 22301 (0.255 sec)
INFO:tensorflow:global_step/sec: 389.9
INFO:tensorflow:loss = 3.330955, step = 22401 (0.257 sec)
INFO:tensorflow:global_step/sec: 237.203
INFO:tensorflow:loss = 0.057503693, step = 22501 (0.421 sec)
INFO:tensorflow:global_step/sec: 412.679
INFO:tensorflow:loss = 0.38263464, step = 22601 (0.242 sec)
INFO:tensorflow:global_step/sec: 423.571
INFO:tensorflow:loss = 0.20552903, step = 22701 (0.236 sec)
INFO:tensorflow:global_step/sec: 407.244
INFO:tensorflow:loss = 0.24284548, step = 22801 (0.246 sec)
INFO:tensorflow:global_step/sec: 398.22
INFO:tensorflow:loss = 0.003155413, step = 22901 (0.251 sec)
INFO:tensorflow:global_step/sec: 393.125
INFO:tensorflow:loss = 0.020468362, step = 23001 (0.254 sec)
INFO:tensorflow:global_step/sec: 405.142
INFO:tensorflow:loss = 1.1727828, step = 23101 (0.247 sec)
INFO:tensorflow:global_step/sec: 243.012
INFO:tensorflow:loss = 0.000101803045, step = 23201 (0.411 sec)
INFO:tensorflow:global_step/sec: 403.148
INFO:tensorflow:loss = 0.9033954, step = 23301 (0.248 sec)
INFO:tensorflow:global_step/sec: 382.794
INFO:tensorflow:loss = 0.009518543, step = 23401 (0.261 sec)
INFO:tensorflow:global_step/sec: 403.907
INFO:tensorflow:loss = 0.19528377, step = 23501 (0.248 sec)
INFO:tensorflow:global_step/sec: 406.438
INFO:tensorflow:loss = 1.3068238, step = 23601 (0.246 sec)
INFO:tensorflow:global_step/sec: 251.279
```

```
INFO:tensorflow:loss = 0.03775777, step = 23701 (0.398 sec)
INFO:tensorflow:global_step/sec: 417.554
INFO:tensorflow:loss = 0.022736287, step = 23801 (0.239 sec)
INFO:tensorflow:global_step/sec: 404.714
INFO:tensorflow:loss = 1.8452011, step = 23901 (0.247 sec)
INFO:tensorflow:global_step/sec: 417.235
INFO:tensorflow:loss = 0.02910289, step = 24001 (0.240 sec)
INFO:tensorflow:global_step/sec: 385.932
INFO:tensorflow:loss = 7.573097, step = 24101 (0.259 sec)
INFO:tensorflow:global_step/sec: 419.364
INFO:tensorflow:loss = 0.025665272, step = 24201 (0.238 sec)
INFO:tensorflow:global_step/sec: 393.508
INFO:tensorflow:loss = 0.2753815, step = 24301 (0.254 sec)
INFO:tensorflow:global_step/sec: 260.125
INFO:tensorflow:loss = 0.4245387, step = 24401 (0.384 sec)
INFO:tensorflow:global_step/sec: 411.483
INFO:tensorflow:loss = 0.6946127, step = 24501 (0.243 sec)
INFO:tensorflow:global_step/sec: 419.361
INFO:tensorflow:loss = 0.65968275, step = 24601 (0.239 sec)
INFO:tensorflow:global_step/sec: 416.79
INFO:tensorflow:loss = 0.29056656, step = 24701 (0.240 sec)
INFO:tensorflow:global_step/sec: 406.663
INFO:tensorflow:loss = 3.2016227, step = 24801 (0.246 sec)
INFO:tensorflow:global_step/sec: 415.498
INFO:tensorflow:loss = 0.23490275, step = 24901 (0.241 sec)
INFO:tensorflow:global_step/sec: 257.378
INFO:tensorflow:loss = 1.7595723, step = 25001 (0.389 sec)
INFO:tensorflow:global_step/sec: 415.512
INFO:tensorflow:loss = 1.3350389, step = 25101 (0.241 sec)
INFO:tensorflow:global_step/sec: 419.417
INFO:tensorflow:loss = 0.0056185205, step = 25201 (0.238 sec)
INFO:tensorflow:global_step/sec: 356.259
INFO:tensorflow:loss = 0.0020942013, step = 25301 (0.281 sec)
INFO:tensorflow:global_step/sec: 412.29
INFO:tensorflow:loss = 0.07401257, step = 25401 (0.243 sec)
INFO:tensorflow:global_step/sec: 402.633
INFO:tensorflow:loss = 0.14942342, step = 25501 (0.248 sec)
INFO:tensorflow:global_step/sec: 235.533
INFO:tensorflow:loss = 0.6253224, step = 25601 (0.425 sec)
INFO:tensorflow:global_step/sec: 369.772
INFO:tensorflow:loss = 4.3326173, step = 25701 (0.270 sec)
INFO:tensorflow:global_step/sec: 416.183
INFO:tensorflow:loss = 1.6731849, step = 25801 (0.240 sec)
INFO:tensorflow:global_step/sec: 411.777
INFO:tensorflow:loss = 2.6045926, step = 25901 (0.243 sec)
INFO:tensorflow:global_step/sec: 419.445
INFO:tensorflow:loss = 0.24810106, step = 26001 (0.238 sec)
INFO:tensorflow:global_step/sec: 403.338
```

```
INFO:tensorflow:loss = 2.81542, step = 26101 (0.248 sec)
INFO:tensorflow:global_step/sec: 256.582
INFO:tensorflow:loss = 4.7174525, step = 26201 (0.390 sec)
INFO:tensorflow:global_step/sec: 416.955
INFO:tensorflow:loss = 0.10414812, step = 26301 (0.240 sec)
INFO:tensorflow:global_step/sec: 395.096
INFO:tensorflow:loss = 0.14132825, step = 26401 (0.253 sec)
INFO:tensorflow:global_step/sec: 406.903
INFO:tensorflow:loss = 0.048349638, step = 26501 (0.246 sec)
INFO:tensorflow:global_step/sec: 414.06
INFO:tensorflow:loss = 0.15741107, step = 26601 (0.241 sec)
INFO:tensorflow:global_step/sec: 410.865
INFO:tensorflow:loss = 0.023086272, step = 26701 (0.243 sec)
INFO:tensorflow:global_step/sec: 255.865
INFO:tensorflow:loss = 0.118164055, step = 26801 (0.391 sec)
INFO:tensorflow:global_step/sec: 392.474
INFO:tensorflow:loss = 1.3084832, step = 26901 (0.255 sec)
INFO:tensorflow:global_step/sec: 397.265
INFO:tensorflow:loss = 0.107358694, step = 27001 (0.252 sec)
INFO:tensorflow:global_step/sec: 351.453
INFO:tensorflow:loss = 0.008322743, step = 27101 (0.285 sec)
INFO:tensorflow:global_step/sec: 408.018
INFO:tensorflow:loss = 5.776402, step = 27201 (0.245 sec)
INFO:tensorflow:global_step/sec: 408.609
INFO:tensorflow:loss = 0.13210742, step = 27301 (0.245 sec)
INFO:tensorflow:global_step/sec: 257.792
INFO:tensorflow:loss = 0.005082514, step = 27401 (0.388 sec)
INFO:tensorflow:global_step/sec: 373.666
INFO:tensorflow:loss = 0.28704378, step = 27501 (0.268 sec)
INFO:tensorflow:global_step/sec: 412.899
INFO:tensorflow:loss = 0.08398219, step = 27601 (0.242 sec)
INFO:tensorflow:global_step/sec: 409.254
INFO:tensorflow:loss = 0.16767268, step = 27701 (0.244 sec)
INFO:tensorflow:global_step/sec: 418.571
INFO:tensorflow:loss = 1.1595501, step = 27801 (0.239 sec)
INFO:tensorflow:global_step/sec: 415.183
INFO:tensorflow:loss = 0.2827449, step = 27901 (0.241 sec)
INFO:tensorflow:global_step/sec: 248.065
INFO:tensorflow:loss = 0.00977083, step = 28001 (0.403 sec)
INFO:tensorflow:global_step/sec: 403.636
INFO:tensorflow:loss = 0.019779464, step = 28101 (0.248 sec)
INFO:tensorflow:global_step/sec: 421.98
INFO:tensorflow:loss = 0.11059987, step = 28201 (0.237 sec)
INFO:tensorflow:global_step/sec: 415.367
INFO:tensorflow:loss = 0.008243097, step = 28301 (0.241 sec)
INFO:tensorflow:global_step/sec: 389.411
INFO:tensorflow:loss = 1.1650462, step = 28401 (0.257 sec)
INFO:tensorflow:global_step/sec: 388.377
```

```
INFO:tensorflow:loss = 0.12788452, step = 28501 (0.257 sec)
INFO:tensorflow:global_step/sec: 237.098
INFO:tensorflow:loss = 0.18881997, step = 28601 (0.422 sec)
INFO:tensorflow:global_step/sec: 383.714
INFO:tensorflow:loss = 0.18734635, step = 28701 (0.261 sec)
INFO:tensorflow:global_step/sec: 321.899
INFO:tensorflow:loss = 0.3146275, step = 28801 (0.311 sec)
INFO:tensorflow:global_step/sec: 370.905
INFO:tensorflow:loss = 0.244937, step = 28901 (0.269 sec)
INFO:tensorflow:global_step/sec: 365.394
INFO:tensorflow:loss = 0.0066863326, step = 29001 (0.274 sec)
INFO:tensorflow:global_step/sec: 248.606
INFO:tensorflow:loss = 0.0311957, step = 29101 (0.402 sec)
INFO:tensorflow:global_step/sec: 422.163
INFO:tensorflow:loss = 0.2660516, step = 29201 (0.237 sec)
INFO:tensorflow:global_step/sec: 295.011
INFO:tensorflow:loss = 0.49576533, step = 29301 (0.339 sec)
INFO:tensorflow:global_step/sec: 395.624
INFO:tensorflow:loss = 0.0616912, step = 29401 (0.253 sec)
INFO:tensorflow:global_step/sec: 346.167
INFO:tensorflow:loss = 0.16842334, step = 29501 (0.289 sec)
INFO:tensorflow:global_step/sec: 214.008
INFO:tensorflow:loss = 0.017992675, step = 29601 (0.467 sec)
INFO:tensorflow:global_step/sec: 349.236
INFO:tensorflow:loss = 1.4877839, step = 29701 (0.288 sec)
INFO:tensorflow:global_step/sec: 343.714
INFO:tensorflow:loss = 9.070291, step = 29801 (0.289 sec)
INFO:tensorflow:global_step/sec: 281.263
INFO:tensorflow:loss = 0.023184692, step = 29901 (0.356 sec)
INFO:tensorflow:global_step/sec: 327.315
INFO:tensorflow:loss = 0.43975648, step = 30001 (0.306 sec)
INFO:tensorflow:global_step/sec: 209.701
INFO:tensorflow:loss = 0.43953192, step = 30101 (0.477 sec)
INFO:tensorflow:global_step/sec: 287.581
INFO:tensorflow:loss = 4.9164653, step = 30201 (0.348 sec)
INFO:tensorflow:global_step/sec: 347.718
INFO:tensorflow:loss = 0.08000064, step = 30301 (0.288 sec)
INFO:tensorflow:global_step/sec: 411.247
INFO:tensorflow:loss = 0.002580046, step = 30401 (0.243 sec)
INFO:tensorflow:global_step/sec: 385.631
INFO:tensorflow:loss = 0.010013884, step = 30501 (0.260 sec)
INFO:tensorflow:global_step/sec: 245.746
INFO:tensorflow:loss = 0.09068731, step = 30601 (0.406 sec)
INFO:tensorflow:global_step/sec: 397.874
INFO:tensorflow:loss = 0.00059302046, step = 30701 (0.251 sec)
INFO:tensorflow:global_step/sec: 363.391
INFO:tensorflow:loss = 0.55528975, step = 30801 (0.275 sec)
INFO:tensorflow:global_step/sec: 369.569
```

```
INFO:tensorflow:loss = 0.7782805, step = 30901 (0.270 sec)
INFO:tensorflow:global_step/sec: 394.039
INFO:tensorflow:loss = 0.025206797, step = 31001 (0.254 sec)
INFO:tensorflow:global_step/sec: 382.349
INFO:tensorflow:loss = 0.095017746, step = 31101 (0.262 sec)
INFO:tensorflow:global_step/sec: 243.699
INFO:tensorflow:loss = 0.0043923026, step = 31201 (0.410 sec)
INFO:tensorflow:global_step/sec: 323.82
INFO:tensorflow:loss = 0.097631894, step = 31301 (0.312 sec)
INFO:tensorflow:global_step/sec: 373.094
INFO:tensorflow:loss = 0.006430723, step = 31401 (0.265 sec)
INFO:tensorflow:global_step/sec: 397.72
INFO:tensorflow:loss = 0.061967343, step = 31501 (0.251 sec)
INFO:tensorflow:global_step/sec: 352.727
INFO:tensorflow:loss = 0.000983816, step = 31601 (0.283 sec)
INFO:tensorflow:global_step/sec: 196.509
INFO:tensorflow:loss = 0.022956263, step = 31701 (0.509 sec)
INFO:tensorflow:global_step/sec: 385.444
INFO:tensorflow:loss = 0.0010773682, step = 31801 (0.259 sec)
INFO:tensorflow:global_step/sec: 378.418
INFO:tensorflow:loss = 2.0160468, step = 31901 (0.264 sec)
INFO:tensorflow:global_step/sec: 381.152
INFO:tensorflow:loss = 5.6833973, step = 32001 (0.262 sec)
INFO:tensorflow:global_step/sec: 397.127
INFO:tensorflow:loss = 0.0011593155, step = 32101 (0.252 sec)
INFO:tensorflow:global_step/sec: 308.822
INFO:tensorflow:loss = 0.8302722, step = 32201 (0.324 sec)
INFO:tensorflow:global_step/sec: 199.303
INFO:tensorflow:loss = 0.00038764707, step = 32301 (0.502 sec)
INFO:tensorflow:global_step/sec: 262.268
INFO:tensorflow:loss = 0.016388305, step = 32401 (0.381 sec)
INFO:tensorflow:global_step/sec: 370.272
INFO:tensorflow:loss = 0.016458249, step = 32501 (0.270 sec)
INFO:tensorflow:global_step/sec: 396.002
INFO:tensorflow:loss = 0.91006374, step = 32601 (0.253 sec)
INFO:tensorflow:global_step/sec: 410.159
INFO:tensorflow:loss = 0.34081808, step = 32701 (0.244 sec)
INFO:tensorflow:global_step/sec: 251.762
INFO:tensorflow:loss = 3.447937, step = 32801 (0.397 sec)
INFO:tensorflow:global_step/sec: 407.332
INFO:tensorflow:loss = 27.561783, step = 32901 (0.246 sec)
INFO:tensorflow:global_step/sec: 406.385
INFO:tensorflow:loss = 0.0028572907, step = 33001 (0.246 sec)
INFO:tensorflow:global_step/sec: 274.535
INFO:tensorflow:loss = 6.7917957, step = 33101 (0.364 sec)
INFO:tensorflow:global_step/sec: 400.433
INFO:tensorflow:loss = 0.015128328, step = 33201 (0.250 sec)
INFO:tensorflow:global_step/sec: 253.353
```

```
INFO:tensorflow:loss = 0.031083763, step = 33301 (0.395 sec)
INFO:tensorflow:global_step/sec: 396.195
INFO:tensorflow:loss = 1.4731597, step = 33401 (0.252 sec)
INFO:tensorflow:global_step/sec: 382.053
INFO:tensorflow:loss = 0.006582529, step = 33501 (0.262 sec)
INFO:tensorflow:global_step/sec: 360.228
INFO:tensorflow:loss = 0.10605495, step = 33601 (0.278 sec)
INFO:tensorflow:global_step/sec: 418.63
INFO:tensorflow:loss = 0.01183284, step = 33701 (0.239 sec)
INFO:tensorflow:global_step/sec: 397.588
INFO:tensorflow:loss = 1.2167237, step = 33801 (0.252 sec)
INFO:tensorflow:global_step/sec: 247.615
INFO:tensorflow:loss = 0.14220749, step = 33901 (0.404 sec)
INFO:tensorflow:global_step/sec: 349.35
INFO:tensorflow:loss = 0.01564147, step = 34001 (0.286 sec)
INFO:tensorflow:global_step/sec: 361.708
INFO:tensorflow:loss = 2.6214123, step = 34101 (0.276 sec)
INFO:tensorflow:global_step/sec: 391.773
INFO:tensorflow:loss = 0.028850105, step = 34201 (0.255 sec)
INFO:tensorflow:global_step/sec: 395.801
INFO:tensorflow:loss = 0.9652258, step = 34301 (0.253 sec)
INFO:tensorflow:global_step/sec: 410.376
INFO:tensorflow:loss = 0.5343601, step = 34401 (0.244 sec)
INFO:tensorflow:global_step/sec: 249.275
INFO:tensorflow:loss = 1.468349, step = 34501 (0.401 sec)
INFO:tensorflow:global_step/sec: 395.524
INFO:tensorflow:loss = 1.6433228, step = 34601 (0.253 sec)
INFO:tensorflow:global_step/sec: 397.736
INFO:tensorflow:loss = 0.29213047, step = 34701 (0.252 sec)
INFO:tensorflow:global_step/sec: 416.563
INFO:tensorflow:loss = 0.92784065, step = 34801 (0.240 sec)
INFO:tensorflow:global_step/sec: 410.332
INFO:tensorflow:loss = 0.21007898, step = 34901 (0.244 sec)
INFO:tensorflow:global_step/sec: 417.751
INFO:tensorflow:loss = 0.0926537, step = 35001 (0.239 sec)
INFO:tensorflow:global_step/sec: 250.988
INFO:tensorflow:loss = 4.0044575, step = 35101 (0.398 sec)
INFO:tensorflow:global_step/sec: 393.962
INFO:tensorflow:loss = 1.3515729, step = 35201 (0.254 sec)
INFO:tensorflow:global_step/sec: 389.302
INFO:tensorflow:loss = 0.25865254, step = 35301 (0.257 sec)
INFO:tensorflow:global_step/sec: 365.889
INFO:tensorflow:loss = 1.4566396, step = 35401 (0.273 sec)
INFO:tensorflow:global_step/sec: 392.263
INFO:tensorflow:loss = 0.010325138, step = 35501 (0.255 sec)
INFO:tensorflow:global_step/sec: 375.526
INFO:tensorflow:loss = 7.0131707, step = 35601 (0.266 sec)
INFO:tensorflow:global_step/sec: 253.783
```

```
INFO:tensorflow:loss = 0.020927299, step = 35701 (0.394 sec)
INFO:tensorflow:global_step/sec: 398.419
INFO:tensorflow:loss = 0.011849696, step = 35801 (0.251 sec)
INFO:tensorflow:global_step/sec: 404.035
INFO:tensorflow:loss = 3.2592578, step = 35901 (0.247 sec)
INFO:tensorflow:global_step/sec: 409.306
INFO:tensorflow:loss = 0.0019304237, step = 36001 (0.244 sec)
INFO:tensorflow:global_step/sec: 350.566
INFO:tensorflow:loss = 0.029294396, step = 36101 (0.285 sec)
INFO:tensorflow:global_step/sec: 238.434
INFO:tensorflow:loss = 0.034085453, step = 36201 (0.420 sec)
INFO:tensorflow:global_step/sec: 414.984
INFO:tensorflow:loss = 0.004848238, step = 36301 (0.241 sec)
INFO:tensorflow:global_step/sec: 394.408
INFO:tensorflow:loss = 0.3763652, step = 36401 (0.253 sec)
INFO:tensorflow:global_step/sec: 384.303
INFO:tensorflow:loss = 0.039408743, step = 36501 (0.260 sec)
INFO:tensorflow:global_step/sec: 395.711
INFO:tensorflow:loss = 1.6020982, step = 36601 (0.253 sec)
INFO:tensorflow:global_step/sec: 396.782
INFO:tensorflow:loss = 0.14904107, step = 36701 (0.252 sec)
INFO:tensorflow:global_step/sec: 248.697
INFO:tensorflow:loss = 0.25326118, step = 36801 (0.402 sec)
INFO:tensorflow:global_step/sec: 379.206
INFO:tensorflow:loss = 2.8431976, step = 36901 (0.264 sec)
INFO:tensorflow:global_step/sec: 404.401
INFO:tensorflow:loss = 3.1448936, step = 37001 (0.247 sec)
INFO:tensorflow:global_step/sec: 396.611
INFO:tensorflow:loss = 0.13905446, step = 37101 (0.252 sec)
INFO:tensorflow:global_step/sec: 388.94
INFO:tensorflow:loss = 0.00037189256, step = 37201 (0.257 sec)
INFO:tensorflow:global_step/sec: 390.393
INFO:tensorflow:loss = 0.6120128, step = 37301 (0.256 sec)
INFO:tensorflow:global_step/sec: 253.384
INFO:tensorflow:loss = 0.0161638, step = 37401 (0.395 sec)
INFO:tensorflow:global_step/sec: 407.644
INFO:tensorflow:loss = 0.00060715707, step = 37501 (0.245 sec)
INFO:tensorflow:global_step/sec: 410.578
INFO:tensorflow:loss = 0.00031660259, step = 37601 (0.244 sec)
INFO:tensorflow:global_step/sec: 413.358
INFO:tensorflow:loss = 0.19170636, step = 37701 (0.242 sec)
INFO:tensorflow:global_step/sec: 385.319
INFO:tensorflow:loss = 2.8298051, step = 37801 (0.260 sec)
INFO:tensorflow:global_step/sec: 395.498
INFO:tensorflow:loss = 0.2177741, step = 37901 (0.253 sec)
INFO:tensorflow:global_step/sec: 249.301
INFO:tensorflow:loss = 0.045841336, step = 38001 (0.401 sec)
INFO:tensorflow:global_step/sec: 384.315
```

```
INFO:tensorflow:loss = 1.2636109e-05, step = 38101 (0.260 sec)
INFO:tensorflow:global_step/sec: 414.777
INFO:tensorflow:loss = 1.4162133, step = 38201 (0.241 sec)
INFO:tensorflow:global_step/sec: 410.965
INFO:tensorflow:loss = 0.27094528, step = 38301 (0.243 sec)
INFO:tensorflow:global_step/sec: 415.201
INFO:tensorflow:loss = 0.010078376, step = 38401 (0.241 sec)
INFO:tensorflow:global_step/sec: 385.89
INFO:tensorflow:loss = 0.35707432, step = 38501 (0.259 sec)
INFO:tensorflow:global_step/sec: 257.201
INFO:tensorflow:loss = 0.02392611, step = 38601 (0.389 sec)
INFO:tensorflow:global_step/sec: 422.615
INFO:tensorflow:loss = 0.0006185092, step = 38701 (0.237 sec)
INFO:tensorflow:global_step/sec: 409.666
INFO:tensorflow:loss = 1.4539686, step = 38801 (0.244 sec)
INFO:tensorflow:global_step/sec: 393.674
INFO:tensorflow:loss = 0.0005634421, step = 38901 (0.254 sec)
INFO:tensorflow:global_step/sec: 367.38
INFO:tensorflow:loss = 2.3405397, step = 39001 (0.272 sec)
INFO:tensorflow:global_step/sec: 386.337
INFO:tensorflow:loss = 1.7092342, step = 39101 (0.259 sec)
INFO:tensorflow:global_step/sec: 258.781
INFO:tensorflow:loss = 1.9462471, step = 39201 (0.386 sec)
INFO:tensorflow:global_step/sec: 423.603
INFO:tensorflow:loss = 0.0016879421, step = 39301 (0.236 sec)
INFO:tensorflow:global_step/sec: 422.117
INFO:tensorflow:loss = 0.0021943832, step = 39401 (0.237 sec)
INFO:tensorflow:global_step/sec: 423.272
INFO:tensorflow:loss = 0.27620104, step = 39501 (0.236 sec)
INFO:tensorflow:global_step/sec: 404.088
INFO:tensorflow:loss = 0.107087724, step = 39601 (0.247 sec)
INFO:tensorflow:global_step/sec: 417.31
INFO:tensorflow:loss = 0.011011706, step = 39701 (0.240 sec)
INFO:tensorflow:global_step/sec: 254.928
INFO:tensorflow:loss = 0.114757776, step = 39801 (0.392 sec)
INFO:tensorflow:global_step/sec: 419.952
INFO:tensorflow:loss = 0.0017328602, step = 39901 (0.238 sec)
INFO:tensorflow:global_step/sec: 417.867
INFO:tensorflow:loss = 0.20214419, step = 40001 (0.239 sec)
INFO:tensorflow:global_step/sec: 415.973
INFO:tensorflow:loss = 0.00019847673, step = 40101 (0.240 sec)
INFO:tensorflow:global_step/sec: 421.891
INFO:tensorflow:loss = 0.07575661, step = 40201 (0.237 sec)
INFO:tensorflow:global_step/sec: 399.407
INFO:tensorflow:loss = 0.003445439, step = 40301 (0.250 sec)
INFO:tensorflow:global_step/sec: 238.788
INFO:tensorflow:loss = 0.05041053, step = 40401 (0.419 sec)
INFO:tensorflow:global_step/sec: 405.173
```

```
INFO:tensorflow:loss = 0.0114676505, step = 40501 (0.247 sec)
INFO:tensorflow:global_step/sec: 419.523
INFO:tensorflow:loss = 0.015533789, step = 40601 (0.238 sec)
INFO:tensorflow:global_step/sec: 414.53
INFO:tensorflow:loss = 0.0034927307, step = 40701 (0.241 sec)
INFO:tensorflow:global_step/sec: 407.488
INFO:tensorflow:loss = 0.05314668, step = 40801 (0.246 sec)
INFO:tensorflow:global_step/sec: 408.955
INFO:tensorflow:loss = 0.6328082, step = 40901 (0.244 sec)
INFO:tensorflow:global_step/sec: 244.776
INFO:tensorflow:loss = 0.9145455, step = 41001 (0.409 sec)
INFO:tensorflow:global_step/sec: 390.132
INFO:tensorflow:loss = 1.7744827, step = 41101 (0.256 sec)
INFO:tensorflow:global_step/sec: 405.409
INFO:tensorflow:loss = 0.057204023, step = 41201 (0.247 sec)
INFO:tensorflow:global_step/sec: 392.177
INFO:tensorflow:loss = 8.635128, step = 41301 (0.255 sec)
INFO:tensorflow:global_step/sec: 383.673
INFO:tensorflow:loss = 0.020850537, step = 41401 (0.260 sec)
INFO:tensorflow:global_step/sec: 365.994
INFO:tensorflow:loss = 0.02721893, step = 41501 (0.273 sec)
INFO:tensorflow:global_step/sec: 252.046
INFO:tensorflow:loss = 0.019438405, step = 41601 (0.397 sec)
INFO:tensorflow:global_step/sec: 381.547
INFO:tensorflow:loss = 0.34056336, step = 41701 (0.262 sec)
INFO:tensorflow:global_step/sec: 309.486
INFO:tensorflow:loss = 0.99568933, step = 41801 (0.323 sec)
INFO:tensorflow:global_step/sec: 268.692
INFO:tensorflow:loss = 0.1697374, step = 41901 (0.372 sec)
INFO:tensorflow:global_step/sec: 332.386
INFO:tensorflow:loss = 0.11820179, step = 42001 (0.301 sec)
INFO:tensorflow:global_step/sec: 231.956
INFO:tensorflow:loss = 0.0064964946, step = 42101 (0.431 sec)
INFO:tensorflow:global_step/sec: 376.696
INFO:tensorflow:loss = 0.014036303, step = 42201 (0.266 sec)
INFO:tensorflow:global_step/sec: 366.486
INFO:tensorflow:loss = 0.12771599, step = 42301 (0.273 sec)
INFO:tensorflow:global_step/sec: 319.824
INFO:tensorflow:loss = 0.3337269, step = 42401 (0.313 sec)
INFO:tensorflow:global_step/sec: 354.386
INFO:tensorflow:loss = 0.16199884, step = 42501 (0.283 sec)
INFO:tensorflow:global_step/sec: 232.851
INFO:tensorflow:loss = 0.10752998, step = 42601 (0.429 sec)
INFO:tensorflow:global_step/sec: 362.796
INFO:tensorflow:loss = 0.19530521, step = 42701 (0.276 sec)
INFO:tensorflow:global_step/sec: 329.791
INFO:tensorflow:loss = 0.0013950162, step = 42801 (0.303 sec)
INFO:tensorflow:global_step/sec: 324.987
```

```
INFO:tensorflow:loss = 0.012120719, step = 42901 (0.308 sec)
INFO:tensorflow:global_step/sec: 309.801
INFO:tensorflow:loss = 0.042721, step = 43001 (0.323 sec)
INFO:tensorflow:global_step/sec: 232.792
INFO:tensorflow:loss = 0.0014023589, step = 43101 (0.430 sec)
INFO:tensorflow:global_step/sec: 381.981
INFO:tensorflow:loss = 1.9083052, step = 43201 (0.262 sec)
INFO:tensorflow:global_step/sec: 384.657
INFO:tensorflow:loss = 4.2199636, step = 43301 (0.260 sec)
INFO:tensorflow:global_step/sec: 361.033
INFO:tensorflow:loss = 11.235496, step = 43401 (0.277 sec)
INFO:tensorflow:global_step/sec: 377.925
INFO:tensorflow:loss = 0.011806647, step = 43501 (0.265 sec)
INFO:tensorflow:global_step/sec: 370.744
INFO:tensorflow:loss = 0.01396229, step = 43601 (0.270 sec)
INFO:tensorflow:global_step/sec: 244.312
INFO:tensorflow:loss = 0.07190217, step = 43701 (0.409 sec)
INFO:tensorflow:global_step/sec: 402.325
INFO:tensorflow:loss = 0.0005938319, step = 43801 (0.249 sec)
INFO:tensorflow:global_step/sec: 423.848
INFO:tensorflow:loss = 0.0045281216, step = 43901 (0.236 sec)
INFO:tensorflow:global_step/sec: 295.892
INFO:tensorflow:loss = 0.44992262, step = 44001 (0.338 sec)
INFO:tensorflow:global_step/sec: 381.957
INFO:tensorflow:loss = 0.8427059, step = 44101 (0.262 sec)
INFO:tensorflow:global_step/sec: 391.555
INFO:tensorflow:loss = 0.6874251, step = 44201 (0.255 sec)
INFO:tensorflow:global_step/sec: 244.816
INFO:tensorflow:loss = 4.792183e-05, step = 44301 (0.408 sec)
INFO:tensorflow:global_step/sec: 393.374
INFO:tensorflow:loss = 0.037978616, step = 44401 (0.254 sec)
INFO:tensorflow:global_step/sec: 409.475
INFO:tensorflow:loss = 1.178717, step = 44501 (0.244 sec)
INFO:tensorflow:global_step/sec: 417.35
INFO:tensorflow:loss = 0.0032597298, step = 44601 (0.240 sec)
INFO:tensorflow:global_step/sec: 408.18
INFO:tensorflow:loss = 0.041579753, step = 44701 (0.245 sec)
INFO:tensorflow:global_step/sec: 412.254
INFO:tensorflow:loss = 0.10554001, step = 44801 (0.243 sec)
INFO:tensorflow:global_step/sec: 259.296
INFO:tensorflow:loss = 0.24150623, step = 44901 (0.386 sec)
INFO:tensorflow:global_step/sec: 422.81
INFO:tensorflow:loss = 0.046450704, step = 45001 (0.237 sec)
INFO:tensorflow:global_step/sec: 414.441
INFO:tensorflow:loss = 0.009447148, step = 45101 (0.241 sec)
INFO:tensorflow:global_step/sec: 415.433
INFO:tensorflow:loss = 2.2249033, step = 45201 (0.241 sec)
INFO:tensorflow:global_step/sec: 416.922
```

```
INFO:tensorflow:loss = 0.0048102294, step = 45301 (0.240 sec)
INFO:tensorflow:global_step/sec: 416.764
INFO:tensorflow:loss = 0.74210745, step = 45401 (0.240 sec)
INFO:tensorflow:global_step/sec: 258.159
INFO:tensorflow:loss = 0.0010051378, step = 45501 (0.387 sec)
INFO:tensorflow:global_step/sec: 396.551
INFO:tensorflow:loss = 0.005090255, step = 45601 (0.252 sec)
INFO:tensorflow:global_step/sec: 421.823
INFO:tensorflow:loss = 0.08577404, step = 45701 (0.237 sec)
INFO:tensorflow:global_step/sec: 419.461
INFO:tensorflow:loss = 0.0005924273, step = 45801 (0.239 sec)
INFO:tensorflow:global_step/sec: 416.729
INFO:tensorflow:loss = 0.07647703, step = 45901 (0.240 sec)
INFO:tensorflow:global_step/sec: 190.175
INFO:tensorflow:loss = 3.2177722, step = 46001 (0.526 sec)
INFO:tensorflow:global_step/sec: 296.292
INFO:tensorflow:loss = 0.4024702, step = 46101 (0.337 sec)
INFO:tensorflow:global_step/sec: 356.191
INFO:tensorflow:loss = 0.24424508, step = 46201 (0.281 sec)
INFO:tensorflow:global_step/sec: 405.459
INFO:tensorflow:loss = 0.2834914, step = 46301 (0.247 sec)
INFO:tensorflow:global_step/sec: 360.83
INFO:tensorflow:loss = 0.004323193, step = 46401 (0.277 sec)
INFO:tensorflow:global_step/sec: 372.302
INFO:tensorflow:loss = 0.00025293688, step = 46501 (0.269 sec)
INFO:tensorflow:global_step/sec: 249.725
INFO:tensorflow:loss = 0.059424113, step = 46601 (0.400 sec)
INFO:tensorflow:global_step/sec: 424.913
INFO:tensorflow:loss = 4.0730324, step = 46701 (0.235 sec)
INFO:tensorflow:global_step/sec: 423.397
INFO:tensorflow:loss = 0.0058783717, step = 46801 (0.236 sec)
INFO:tensorflow:global_step/sec: 396.47
INFO:tensorflow:loss = 0.03376456, step = 46901 (0.252 sec)
INFO:tensorflow:global_step/sec: 402.244
INFO:tensorflow:loss = 0.0037548477, step = 47001 (0.249 sec)
INFO:tensorflow:global_step/sec: 383.183
INFO:tensorflow:loss = 0.009259952, step = 47101 (0.261 sec)
INFO:tensorflow:global_step/sec: 257.052
INFO:tensorflow:loss = 3.4814148, step = 47201 (0.389 sec)
INFO:tensorflow:global_step/sec: 406.126
INFO:tensorflow:loss = 0.007909078, step = 47301 (0.246 sec)
INFO:tensorflow:global_step/sec: 423.259
INFO:tensorflow:loss = 0.096214235, step = 47401 (0.236 sec)
INFO:tensorflow:global_step/sec: 424.914
INFO:tensorflow:loss = 6.9974645e-05, step = 47501 (0.235 sec)
INFO:tensorflow:global_step/sec: 411.005
INFO:tensorflow:loss = 1.4071494, step = 47601 (0.243 sec)
INFO:tensorflow:global_step/sec: 408.526
```

```
INFO:tensorflow:loss = 0.031153241, step = 47701 (0.245 sec)
INFO:tensorflow:global_step/sec: 250.268
INFO:tensorflow:loss = 0.00072125066, step = 47801 (0.400 sec)
INFO:tensorflow:global_step/sec: 393.608
INFO:tensorflow:loss = 1.2634991, step = 47901 (0.254 sec)
INFO:tensorflow:Saving checkpoints for 48000 into
/var/folders/5s/f9dl19z88xjgbj0059_sl_r00000gn/T/tmprv4yj7le/model.ckpt.
INFO:tensorflow:Loss for final step: 0.01182653.
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2019-08-04T23:28:51Z
INFO:tensorflow:Graph was finalized.
WARNING:tensorflow:From /Users/jmwanat/anaconda3/envs/tf/lib/python3.7/site-
packages/tensorflow/python/training/saver.py:1266: checkpoint_exists (from
tensorflow.python.training.checkpoint_management) is deprecated and will be
removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
INFO:tensorflow:Restoring parameters from
/var/folders/5s/f9dl19z88xjgbj0059_sl_r00000gn/T/tmprv4yj7le/model.ckpt-48000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2019-08-04-23:28:52
INFO:tensorflow:Saving dict for global step 48000: accuracy = 0.975,
average_loss = 0.19617826, global_step = 48000, loss = 24.832691
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 48000:
/var/folders/5s/f9dl19z88xjgbj0059_sl_r00000gn/T/tmprv4yj7le/model.ckpt-48000
Test accuracy: 0.9750000238418579
Time to train and evaluate model: 142.398 seconds
```

[20]:
```
#https://www.tensorflow.org/guide/premade_estimators

#Classifier.evaluate provides the accuracy of the trained model on the test␣
 ↪data
#The eval_result dictionary also contains the
#average_loss (mean loss per sample),
#the loss (mean loss per mini-batch) and
#the value of the estimator's global_step
#(the number of training iterations it underwent).

evaluate_metrics
```

[20]:
```
{'accuracy': 0.975,
 'average_loss': 0.19617826,
 'loss': 24.832691,
 'global_step': 48000}
```

[21]:
```
print("Time to train and evaluate model: {:.3f} seconds".format(time5))
```

```
Time to train and evaluate model: 142.398 seconds
```

3 Tensorflow with Keras

3.1 Import libraries

[22]:
```
from keras.models import Sequential
from keras.layers import Dense, Activation
from tensorflow.python.keras.callbacks import TensorBoard
import time
import matplotlib.pyplot as plt

#reset graph to remove duplicate nodes
tf.reset_default_graph()
```

[23]:
```
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
X_train = X_train.astype(np.float32).reshape(-1,28*28)/255.0
X_test = X_test.astype(np.float32).reshape(-1,28*28)/255.0
y_train = y_train.astype(np.int32)
y_test = y_test.astype(np.int32)
```

3.2 Keras Model with 2 layers and 128 and 32 nodes

[24]:
```
#https://keras.io/getting-started/sequential-model-guide/
#Instantiate an estimator
model = Sequential()
#layer 1 with 128 nodes
model.add(Dense(128, input_dim=784))  #128 nodes
model.add(Activation('relu'))
#layer 2 with 32 nodes
model.add(Dense(32))                  #32 nodes
model.add(Activation('softmax'))
```

[25]:
```
# For a multi-class classification problem
#Before training a model, you need to configure
#the learning process, which is done via the compile method.
#When using the sparse_categorical_crossentropy loss, your targets should be␣
 ↪integer targets.

model.compile(optimizer='rmsprop',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

[26]:
```
t0 = time.time()
# Train the model, iterating on the data in batches of 50 samples
history = model.fit(X_train, y_train, epochs=40, batch_size=50)
# Evaluate the model
```

```
score = model.evaluate(X_test, y_test, batch_size=50)

t1 = time.time()
print('Total time to train and evaluate model: {:.3f}'.format(t1-t0))
time_keras1 = t1-t0
```

```
WARNING:tensorflow:From /Users/jmwanat/anaconda3/envs/tf/lib/python3.7/site-
packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.2929 -
acc: 0.9178
Epoch 2/40
60000/60000 [==============================] - 6s 100us/step - loss: 0.1338 -
acc: 0.9609
Epoch 3/40
60000/60000 [==============================] - 6s 101us/step - loss: 0.0955 -
acc: 0.9727
Epoch 4/40
60000/60000 [==============================] - 7s 113us/step - loss: 0.0750 -
acc: 0.9777
Epoch 5/40
60000/60000 [==============================] - 7s 111us/step - loss: 0.0621 -
acc: 0.9823
Epoch 6/40
60000/60000 [==============================] - 6s 103us/step - loss: 0.0527 -
acc: 0.9843
Epoch 7/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0450 -
acc: 0.9875
Epoch 8/40
60000/60000 [==============================] - 6s 105us/step - loss: 0.0396 -
acc: 0.9885
Epoch 9/40
60000/60000 [==============================] - 6s 105us/step - loss: 0.0343 -
acc: 0.9903
Epoch 10/40
60000/60000 [==============================] - 7s 109us/step - loss: 0.0303 -
acc: 0.9918
Epoch 11/40
60000/60000 [==============================] - 7s 116us/step - loss: 0.0272 -
acc: 0.9925
Epoch 12/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0229 -
```

```
acc: 0.9934
Epoch 13/40
60000/60000 [==============================] - 6s 105us/step - loss: 0.0213 -
acc: 0.9943
Epoch 14/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0183 -
acc: 0.9948
Epoch 15/40
60000/60000 [==============================] - 6s 105us/step - loss: 0.0164 -
acc: 0.9957
Epoch 16/40
60000/60000 [==============================] - 6s 105us/step - loss: 0.0144 -
acc: 0.9962
Epoch 17/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0129 -
acc: 0.9968
Epoch 18/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0113 -
acc: 0.9974
Epoch 19/40
60000/60000 [==============================] - 6s 105us/step - loss: 0.0102 -
acc: 0.9978
Epoch 20/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0091 -
acc: 0.9979
Epoch 21/40
60000/60000 [==============================] - 6s 105us/step - loss: 0.0083 -
acc: 0.9981
Epoch 22/40
60000/60000 [==============================] - 6s 107us/step - loss: 0.0073 -
acc: 0.9984
Epoch 23/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0068 -
acc: 0.9983
Epoch 24/40
60000/60000 [==============================] - 6s 103us/step - loss: 0.0065 -
acc: 0.9987
Epoch 25/40
60000/60000 [==============================] - 6s 107us/step - loss: 0.0057 -
acc: 0.9987
Epoch 26/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0050 -
acc: 0.9989
Epoch 27/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0043 -
acc: 0.9992
Epoch 28/40
60000/60000 [==============================] - 6s 107us/step - loss: 0.0038 -
```

```
acc: 0.9993
Epoch 29/40
60000/60000 [==============================] - 7s 116us/step - loss: 0.0035 -
acc: 0.9994
Epoch 30/40
60000/60000 [==============================] - 7s 110us/step - loss: 0.0034 -
acc: 0.9993
Epoch 31/40
60000/60000 [==============================] - 6s 108us/step - loss: 0.0031 -
acc: 0.9994
Epoch 32/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0030 -
acc: 0.9995
Epoch 33/40
60000/60000 [==============================] - 6s 108us/step - loss: 0.0027 -
acc: 0.9996
Epoch 34/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0028 -
acc: 0.9996
Epoch 35/40
60000/60000 [==============================] - 6s 107us/step - loss: 0.0023 -
acc: 0.9997
Epoch 36/40
60000/60000 [==============================] - 6s 107us/step - loss: 0.0024 -
acc: 0.9996
Epoch 37/40
60000/60000 [==============================] - 6s 108us/step - loss: 0.0020 -
acc: 0.9997
Epoch 38/40
60000/60000 [==============================] - 6s 104us/step - loss: 0.0021 -
acc: 0.9996
Epoch 39/40
60000/60000 [==============================] - 6s 106us/step - loss: 0.0020 -
acc: 0.9997
Epoch 40/40
60000/60000 [==============================] - 6s 108us/step - loss: 0.0017 -
acc: 0.9998
10000/10000 [==============================] - 0s 30us/step
Total time to train and evaluate model: 256.419
```

[27]:
```
score
```

[27]:
```
[0.1660349264065462, 0.9771000015735626]
```

[28]:
```
print(model.metrics_names)
```

```
['loss', 'acc']
```

[29]:
```
keras_test_loss, keras_test_accuracy = score
```

[30]:
```
print('Test loss: {}'.format(keras_test_loss))
```

```
Test loss: 0.1660349264065462
```

[31]:
```
print('Test accuracy: {}'.format(keras_test_accuracy))
```

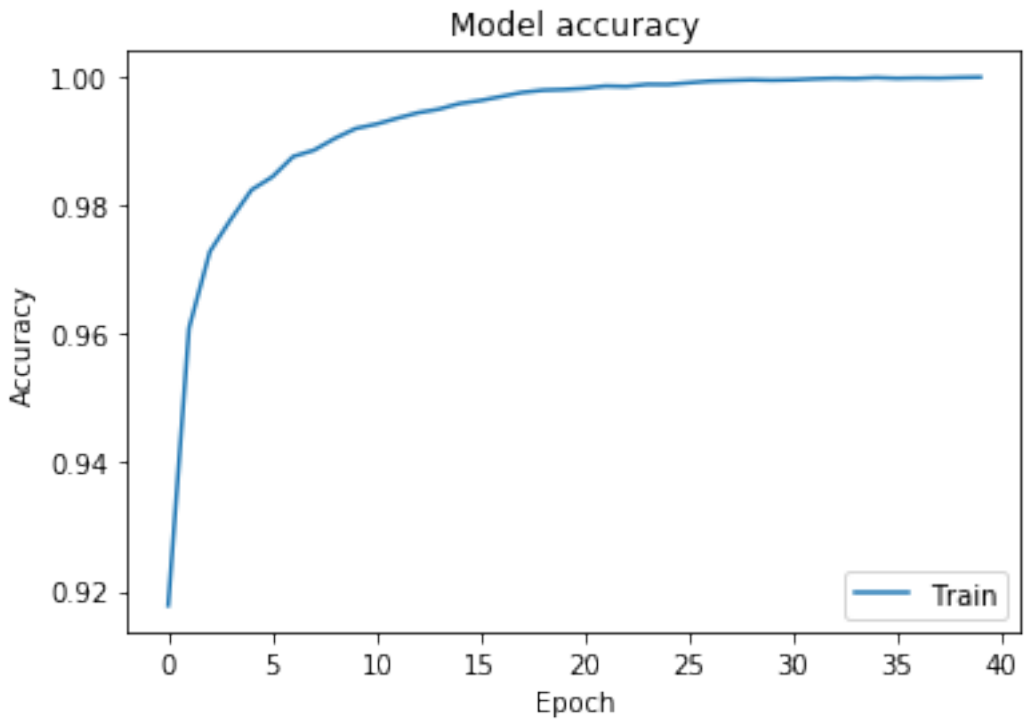
```
Test accuracy: 0.9771000015735626
```

[32]:
```
# list all data in history
print(history.history.keys())
```

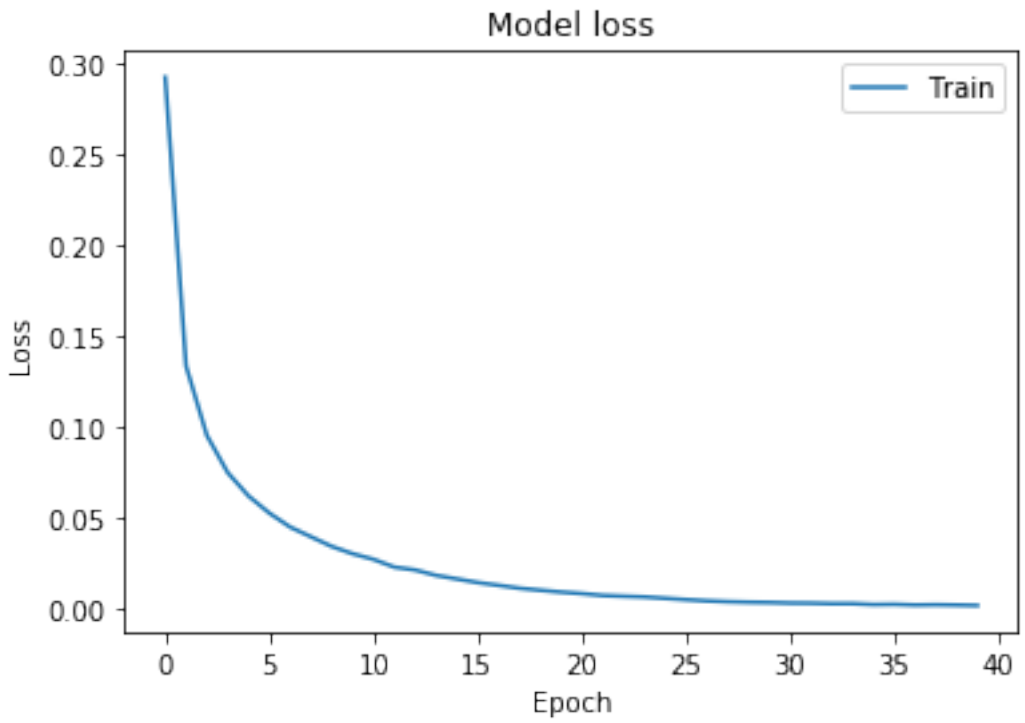
```
dict_keys(['loss', 'acc'])
```

3.3 Visualize Keras model accuracy and loss

[33]:
```
#https://keras.io/visualization/
#history = model.fit(x, y, validation_split=0.25, epochs=50, batch_size=16,␣
 ↪verbose=1)

# Plot training accuracy values
plt.plot(history.history['acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train'], loc='lower right')
plt.savefig('Keras1_train_accuracy.pdf')
plt.show()
```



[34]:
```
# Plot training loss values
plt.plot(history.history['loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train'], loc='upper right')
plt.savefig('Keras1_train_loss.pdf')
plt.show()
```



3.4 Re-initialize model weights

[35]:
```
#https://www.codementor.io/nitinsurya/
 ↪how-to-re-initialize-keras-model-weights-et41zre2g

# define function re-initialize model weights

def reset_weights(model):
    session = tf.keras.backend.get_session()
    for layer in model.layers:
        if hasattr(layer, 'kernel_initializer'):
            layer.kernel.initializer.run(session=session)
```

[37]:
```
# use function to re-intialize model weights
reset_weights(model)

#I don't really need this since I'm calling the second model
#a different name, but saving for future reference

# Clean up the TF session
keras.backend.clear_session()
```

3.5 Keras Model with 2 layers and 20 and 20 nodes

[38]:
```
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
X_train = X_train.astype(np.float32).reshape(-1,28*28)/255.0
X_test = X_test.astype(np.float32).reshape(-1,28*28)/255.0
y_train = y_train.astype(np.int32)
y_test = y_test.astype(np.int32)
```

[39]:
```
#reset graph to remove duplicate nodes
tf.reset_default_graph()

model2 = Sequential()
model2.add(Dense(20, input_dim=784))  #20 nodes
model2.add(Activation('relu'))
model2.add(Dense(20))                  #20 nodes
model2.add(Activation('softmax'))
```

[40]:
```
#https://www.youtube.com/watch?v=2U6Jl7oqRkM
tensorboard = TensorBoard(log_dir = 'logs/{}'.format(time.time()))
#setting up a tensorboard
```

[41]:
```
# For a multi-class classification problem
model2.compile(optimizer='rmsprop',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])
```

[42]:
```
t0 = time.time()
# Train the model, iterating on the data in batches of 50 samples
history2 = model2.fit(X_train, y_train, epochs=40, batch_size=50,␣
 ↪callbacks=[tensorboard])
# Evaluate the model
keras_test2_loss, keras_test2_accuracy = model2.evaluate(X_test, y_test,␣
 ↪batch_size=50)

t1 = time.time()
print('Total time to train and evaluate model: {:.3f}'.format(t1-t0))
time_keras2 = t1-t0
```

```
Epoch 1/40
60000/60000 [==============================] - 5s 83us/step - loss: 0.4620 -
acc: 0.8790
Epoch 2/40
60000/60000 [==============================] - 4s 60us/step - loss: 0.2602 -
acc: 0.9259
Epoch 3/40
60000/60000 [==============================] - 3s 54us/step - loss: 0.2280 -
acc: 0.9346
Epoch 4/40
60000/60000 [==============================] - 3s 53us/step - loss: 0.2072 -
```

```
acc: 0.9412
Epoch 5/40
60000/60000 [==============================] - 3s 54us/step - loss: 0.1908 -
acc: 0.9461
Epoch 6/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1772 -
acc: 0.9495
Epoch 7/40
60000/60000 [==============================] - 3s 54us/step - loss: 0.1653 -
acc: 0.9533
Epoch 8/40
60000/60000 [==============================] - 3s 54us/step - loss: 0.1564 -
acc: 0.9556
Epoch 9/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1501 -
acc: 0.9579: 1s - lo
Epoch 10/40
60000/60000 [==============================] - 3s 50us/step - loss: 0.1438 -
acc: 0.9598
Epoch 11/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1385 -
acc: 0.9607
Epoch 12/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1341 -
acc: 0.9625
Epoch 13/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1302 -
acc: 0.9634
Epoch 14/40
60000/60000 [==============================] - 3s 54us/step - loss: 0.1270 -
acc: 0.9644
Epoch 15/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1241 -
acc: 0.9651: 0s - loss: 0.1247 - acc: 0.96
Epoch 16/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1213 -
acc: 0.9663
Epoch 17/40
60000/60000 [==============================] - 3s 53us/step - loss: 0.1191 -
acc: 0.9670
Epoch 18/40
60000/60000 [==============================] - 3s 54us/step - loss: 0.1163 -
acc: 0.9680
Epoch 19/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1140 -
acc: 0.9688
Epoch 20/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1121 -
```

```
acc: 0.9690
Epoch 21/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1115 -
acc: 0.9692
Epoch 22/40
60000/60000 [==============================] - 3s 50us/step - loss: 0.1091 -
acc: 0.9700
Epoch 23/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1081 -
acc: 0.9700
Epoch 24/40
60000/60000 [==============================] - 3s 56us/step - loss: 0.1063 -
acc: 0.9710
Epoch 25/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1053 -
acc: 0.9706
Epoch 26/40
60000/60000 [==============================] - 3s 53us/step - loss: 0.1044 -
acc: 0.9711
Epoch 27/40
60000/60000 [==============================] - 3s 54us/step - loss: 0.1025 -
acc: 0.9716
Epoch 28/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.1012 -
acc: 0.9722
Epoch 29/40
60000/60000 [==============================] - 3s 53us/step - loss: 0.1001 -
acc: 0.9727
Epoch 30/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.0993 -
acc: 0.9733
Epoch 31/40
60000/60000 [==============================] - 3s 53us/step - loss: 0.0987 -
acc: 0.9733: 1s - lo
Epoch 32/40
60000/60000 [==============================] - 3s 53us/step - loss: 0.0974 -
acc: 0.9736
Epoch 33/40
60000/60000 [==============================] - 3s 52us/step - loss: 0.0961 -
acc: 0.9742
Epoch 34/40
60000/60000 [==============================] - 3s 54us/step - loss: 0.0960 -
acc: 0.9742
Epoch 35/40
60000/60000 [==============================] - 3s 54us/step - loss: 0.0947 -
acc: 0.9749
Epoch 36/40
60000/60000 [==============================] - 3s 54us/step - loss: 0.0940 -
```

```
acc: 0.9745
Epoch 37/40
60000/60000 [==============================] - 3s 55us/step - loss: 0.0935 -
acc: 0.9748
Epoch 38/40
60000/60000 [==============================] - 3s 53us/step - loss: 0.0924 -
acc: 0.9754
Epoch 39/40
60000/60000 [==============================] - 4s 60us/step - loss: 0.0917 -
acc: 0.9749
Epoch 40/40
60000/60000 [==============================] - 4s 62us/step - loss: 0.0915 -
acc: 0.9756
10000/10000 [==============================] - 0s 30us/step
Total time to train and evaluate model: 130.773
```

[43]:
```
print('Test loss: {}'.format(keras_test2_loss))
```

```
Test loss: 0.17423018945548394
```

[44]:
```
print('Test accuracy: {}'.format(keras_test2_accuracy))
```

```
Test accuracy: 0.9612999999523163
```

[45]:
```
#https://www.tensorflow.org/tutorials/keras/basic_regression

import pandas as pd
# I can look at a summary of the loss and accuracy for each epoch
hist = pd.DataFrame(history.history)
hist['epoch'] = history.epoch
hist
```

[45]:

	loss	acc	epoch
0	0.292889	0.917783	0
1	0.133763	0.960850	1
2	0.095487	0.972717	2
3	0.074998	0.977683	3
4	0.062079	0.982283	4
5	0.052658	0.984317	5
6	0.044953	0.987467	6
7	0.039592	0.988450	7
8	0.034263	0.990283	8
9	0.030258	0.991817	9
10	0.027172	0.992500	10
11	0.022880	0.993417	11
12	0.021318	0.994267	12
13	0.018339	0.994817	13

```
14  0.016370  0.995700     14
15  0.014378  0.996167     15
16  0.012850  0.996783     16
17  0.011257  0.997417     17
18  0.010158  0.997767     18
19  0.009125  0.997867     19
20  0.008302  0.998067     20
21  0.007258  0.998417     21
22  0.006831  0.998333     22
23  0.006451  0.998667     23
24  0.005695  0.998650     24
25  0.004970  0.998917     25
26  0.004285  0.999167     26
27  0.003831  0.999267     27
28  0.003549  0.999367     28
29  0.003390  0.999300     29
30  0.003090  0.999367     30
31  0.003010  0.999517     31
32  0.002749  0.999617     32
33  0.002753  0.999567     33
34  0.002254  0.999717     34
35  0.002383  0.999600     35
36  0.002010  0.999667     36
37  0.002140  0.999633     37
38  0.001982  0.999733     38
39  0.001744  0.999783     39
```

[ ]:
```
# To view Tensorboard, type in the terminal shell window:
#$ tensorboard --logdir=logs/
```

3.6 Summary of model performance with Keras

[46]:
```
summary_keras_models = {
    'Number of Layers' : [2, 2],
    'Number of Nodes/Layer' : ['128, 32', '20, 20'],
    'Processing Time' : [round(time_keras1, 3), round(time_keras2,3)],
    'Test Accruacy' : [round(keras_test_accuracy, 3),␣
 ↪round(keras_test2_accuracy, 3)]
}

#convert summary into a dataframe and print
summary_keras_models_df = pd.DataFrame(summary_keras_models)
summary_keras_models_df
```

[46]:

	Number of Layers	Number of Nodes/Layer	Processing Time	Test Accruacy
0	2	128, 32	256.419	0.977
1	2	20, 20	130.773	0.961

3.7 Confusion Matrix: Keras model2 with 2 layers and 20 nodes each

[47]:
```
# I got this idea from:
# https://www.youtube.com/watch?v=km7pxKy4UHU
```

[48]:
```
#import libraries
from sklearn.metrics import confusion_matrix
import itertools
from sklearn.utils.multiclass import unique_labels
```

[49]:
```
#make predictions using X_test data set
y_predictions = model2.predict_classes(X_test, batch_size=BATCH_SIZE, verbose=0)
```

[50]:
```
y_predictions
```

[50]:
```
array([7, 2, 1, ..., 4, 5, 6])
```

[51]:
```
# create labels for y values
test_labels = [0,1,2,3,4,5,6,7,8,9]
test_labels = np.array(test_labels)
print(test_labels)
```

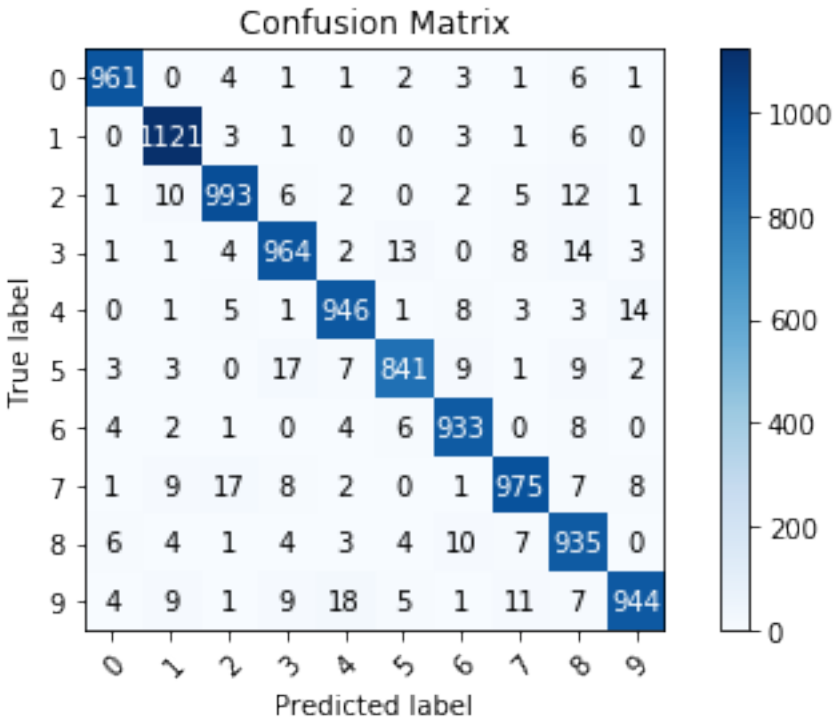
```
[0 1 2 3 4 5 6 7 8 9]
```

[52]:
```
#https://scikit-learn.org/stable/auto_examples/model_selection/
 ↪plot_confusion_matrix.html

def plot_confusion_matrix(y_true, y_pred, classes,
                          normalize=False,
                          title=None,
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'

    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    # Only use the labels that appear in the data
    classes = classes[unique_labels(y_true, y_pred)]
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
```

```
    print(cm)

    fig, ax = plt.subplots()
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
    # We want to show all ticks...
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           # ... and label them with the respective list entries
           xticklabels=classes, yticklabels=classes,
           title=title,
           ylabel='True label',
           xlabel='Predicted label')

    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")

    # Loop over data dimensions and create text annotations.
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center",
                    color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    return ax
```

[53]:
```
#create confusion matrix comparing y_test (true values) against
#y_predictions (predicted values)
plot_confusion_matrix(y_test, y_predictions, classes=test_labels,␣
 ↪title='Confusion Matrix')
plt.savefig('Keras2_confusion_matrix.pdf')
```

```
Confusion matrix, without normalization
[[ 961    0    4    1    1    2    3    1    6    1]
 [   0 1121    3    1    0    0    3    1    6    0]
 [   1   10  993    6    2    0    2    5   12    1]
 [   1    1    4  964    2   13    0    8   14    3]
 [   0    1    5    1  946    1    8    3    3   14]
 [   3    3    0   17    7  841    9    1    9    2]
 [   4    2    1    0    4    6  933    0    8    0]
 [   1    9   17    8    2    0    1  975    7    8]
 [   6    4    1    4    3    4   10    7  935    0]
 [   4    9    1    9   18    5    1   11    7  944]]
```


Confusion Matrix
True label
Predicted label
0 961 0 4 1 1 2 3 1 6 1
1 0 1121 3 1 0 0 3 1 6 0
2 1 10 993 6 2 0 2 5 12 1
3 1 1 4 964 2 13 0 8 14 3
4 0 1 5 1 946 1 8 3 3 14
5 3 3 0 17 7 841 9 1 9 2
6 4 2 1 0 4 6 933 0 8 0
7 1 9 17 8 2 0 1 975 7 8
8 6 4 1 4 3 4 10 7 935 0
9 4 9 1 9 18 5 1 11 7 944
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0

[ ]: