CNN on CIFAR-10: Final Project ADTA 5550 Deep Learning With Big Data

Yog Chaudhary

Set Path for Dataset

In [1]:
```
 # Put file path as a string here

CIFAR_DIR = 'CIFAR_10_DATA/'
```

Load Data

In [2]:
```
CIFAR_DIR
```

Out[2]:
```
'CIFAR_10_DATA/'
```

In [3]:
```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        cifar_dict = pickle.load(fo, encoding='bytes')
    return cifar_dict
```

In [4]:
```
dirs = ['batches.meta','data_batch_1','data_batch_2','data_batch_3','data_batch_4','da

all_data = [0,1,2,3,4,5,6]

for i,direc in zip(all_data,dirs):
    all_data[i] = unpickle(CIFAR_DIR+direc)
```

In [5]:
```
batch_meta = all_data[0]
data_batch1 = all_data[1]
data_batch2 = all_data[2]
data_batch3 = all_data[3]
data_batch4 = all_data[4]
data_batch5 = all_data[5]
test_batch = all_data[6]
```

In [6]:
```
batch_meta
```

Out[6]:
```
{b'num_cases_per_batch': 10000,
 b'label_names': [b'airplane',
  b'automobile',
  b'bird',
  b'cat',
  b'deer',
  b'dog',
  b'frog',
  b'horse',
  b'ship',
  b'truck'],
 b'num_vis': 3072}
```

In [7]:
```
data_batch1.keys()
```

Out[7]:
```
dict_keys([b'batch_label', b'labels', b'data', b'filenames'])
```

Display several examples of single images using matplotlib

In [8]:
```
import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np
```

In [9]:
```
X = data_batch1[b"data"]
```

In [10]:
```
X = X.reshape(10000, 3, 32, 32).transpose(0,2,3,1).astype("uint8")
```

In [11]:
```
X[0].max()
```

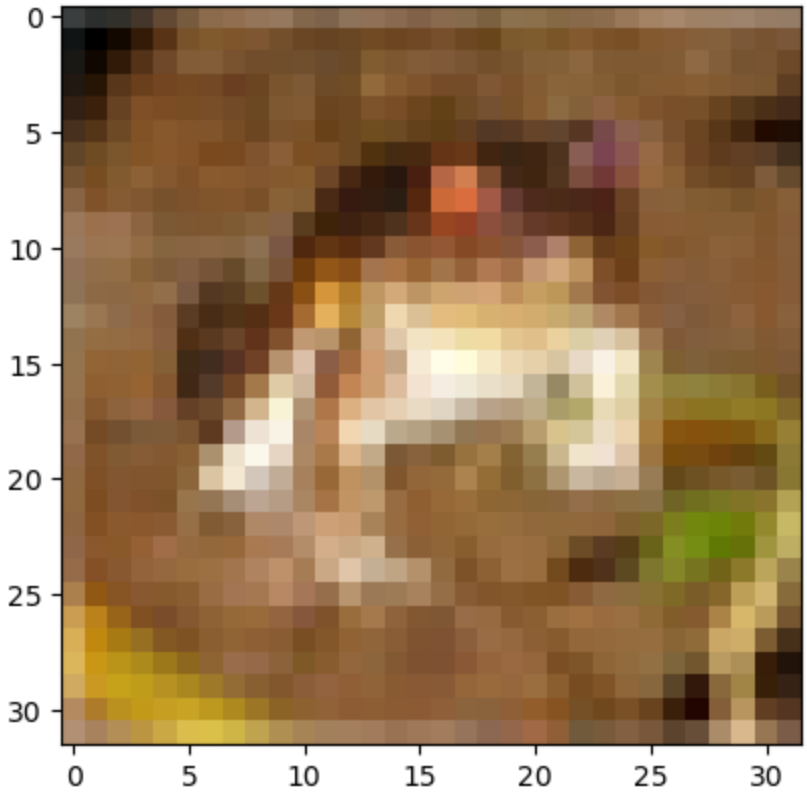
Out[11]:
```
255
```

In [12]:
```
(X[0]/255).max()
```

Out[12]:
```
1.0
```

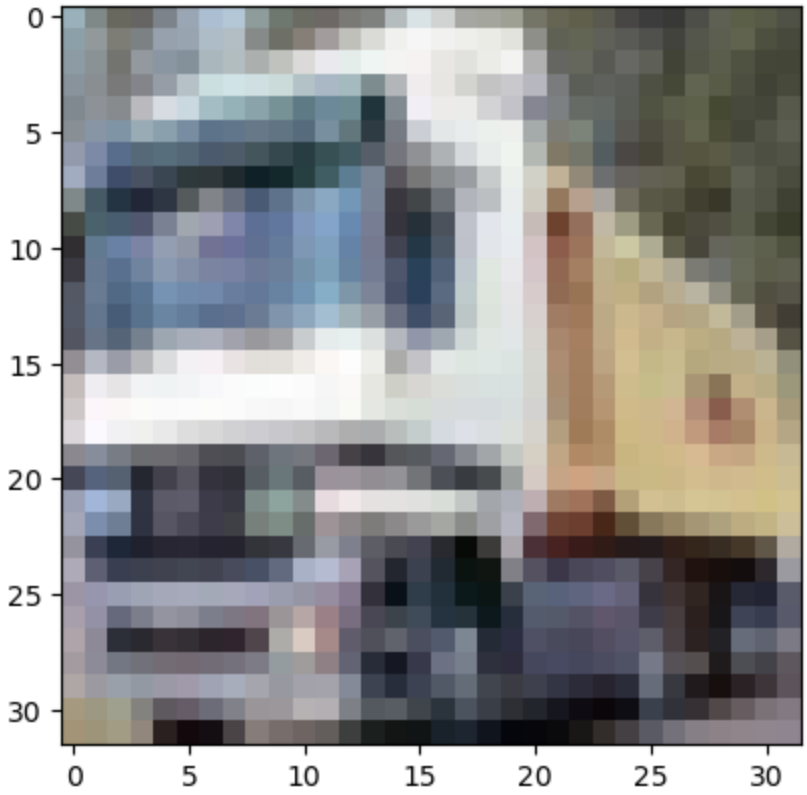
In [13]:
```
plt.imshow(X[0])
```

Out[13]:
```
<matplotlib.image.AxesImage at 0x7ffb040bf310>
```



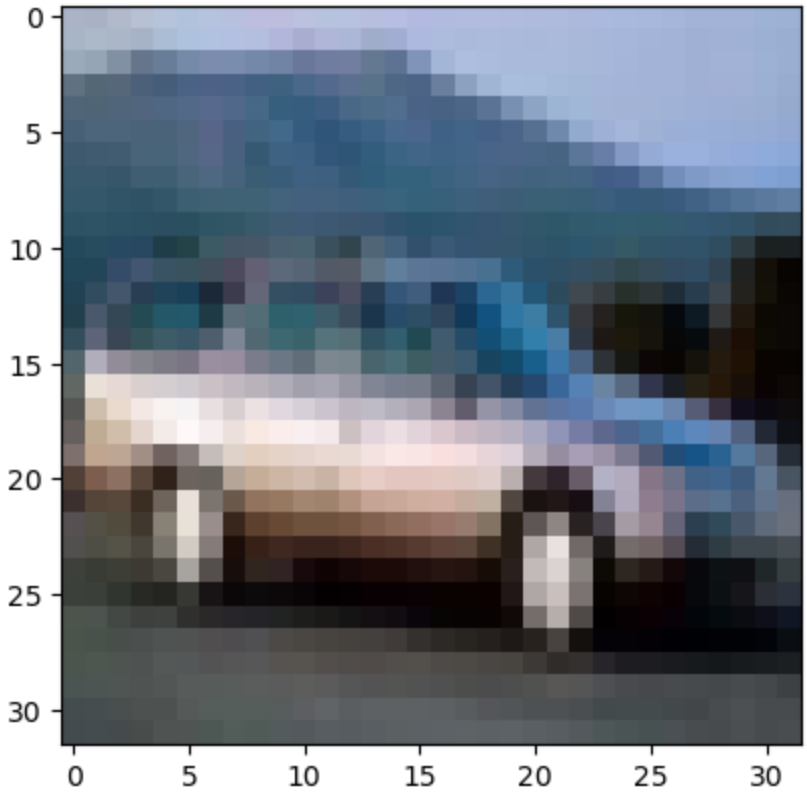
In [14]:
```
plt.imshow(X[1])
```

Out[14]:
```
<matplotlib.image.AxesImage at 0x7ffac7d41150>
```



In [15]:
```
plt.imshow(X[4])
```

Out[15]:
```
<matplotlib.image.AxesImage at 0x7ffac7a8d0d0>
```



Supporting Functions to Rearrange Data

Encode Labels into One-Hot Format

In [16]:
```
def one_hot_encode(vec, vals=10):
    '''
    For use to one-hot encode the 10- possible labels
    '''
    n = len(vec)
    out = np.zeros((n, vals))
    out[range(n), vec] = 1
    return out
```

Set Up Image Data: Make it Ready to be Fed into 1st Conv Layer

In [17]:
```
class CifarHelper():

    def __init__(self):
        self.i = 0

        self.all_train_batches = [data_batch1,data_batch2,data_batch3,data_batch4,data
        self.test_batch = [test_batch]

        self.training_images = None
        self.training_labels = None
```

```
        self.test_images = None
        self.test_labels = None

    def set_up_images(self):

        print("Setting Up Training Images and Labels")

        self.training_images = np.vstack([d[b"data"] for d in self.all_train_batches])
        train_len = len(self.training_images)

        self.training_images = self.training_images.reshape(train_len,3,32,32).transpo
        self.training_labels = one_hot_encode(np.hstack([d[b"labels"] for d in self.al

        print("Setting Up Test Images and Labels")

        self.test_images = np.vstack([d[b"data"] for d in self.test_batch])
        test_len = len(self.test_images)

        self.test_images = self.test_images.reshape(test_len,3,32,32).transpose(0,2,3,
        self.test_labels = one_hot_encode(np.hstack([d[b"labels"] for d in self.test_b


    def next_batch(self, batch_size):
        x = self.training_images[self.i:self.i+batch_size].reshape(100,32,32,3)
        y = self.training_labels[self.i:self.i+batch_size]
        self.i = (self.i + batch_size) % len(self.training_images)
        return x, y
```

Set up image data: Calling CifarHelper.set_up_images()

In [18]:
```
# Before Your tf.Session run these two lines
ch = CifarHelper()
ch.set_up_images()

# During your session to grab the next batch use this line
# (Just like we did for mnist.train.next_batch)
# batch = ch.next_batch(100)
```

```
Setting Up Training Images and Labels
Setting Up Test Images and Labels
```

Define Supporting Functions to Build, Train, and Test CNN Model

In [19]:
```
# initialize weights is filter
# function returns a tf.variable used to store weights in a filter values are random
def initialize_weights(filter_shape):
    init_random_dist = tf.truncated_normal(filter_shape, stddev=0.1)
    return tf.Variable(init_random_dist)
```

In [20]:
```
# initialize bias
#value is initialized to 0.1

def initialize_bias(bias_shape):
    initial_bias_vals = tf.constant(0.1, shape=bias_shape)
    return tf.Variable(initial_bias_vals)
```

In [21]:
```
# Setting up convolutional layer
#return:outputs of layer: the dot product:inputs*weight

def create_convolution_layer_and_compute_dot_product(inputs,filter_shape):
     #initialize the wights in filter
    filter_initialized_with_weights=initialize_weights(filter_shape)

     #create a convolution layer
    conv_layer_outputs=tf.nn.conv2d(inputs,filter_initialized_with_weights, strides=[1

    #return the convolution layer output
    return (conv_layer_outputs)
```

In [22]:
```
def create_relu_layer_compute_dotproduct_plus_b(inputs,filter_shape):

    #initialize bias for each input channel
    b=initialize_bias([filter_shape[3]])

    #perform the computation first by adding: inputs(x*W)+b
    #create a ReLu layer associated with the preceding convolution layer
    relu_layer_outputs=tf.nn.relu(inputs+b)

    #return the output of the ReLu Layer
    return(relu_layer_outputs)
```

In [23]:
```
def create_fully_connected_layer_and_compute_dotproduct_plus_bias(inputs,output_size):

    input_size=int(inputs.get_shape()[1])

    #initilaize weight of the filter of the FC layer
    W=initialize_weights([input_size,output_size])

    #initialize the bias: each bias one output channel
    b=initialize_bias([output_size])

    fc_xW_plus_bias_outputs=tf.matmul(inputs,W)+b

    #retrun the results:outputs
    return (fc_xW_plus_bias_outputs)
```

In [24]:
```
def create_maxpool2by2_and_reduce_spatial_size(inputs):

    # create a pooling layer
    pooling_layer_outputs=tf.nn.max_pool(inputs,ksize=[1,2,2,1], strides=[1,2,2,1],pad

    # return the pooling layer
    return(pooling_layer_outputs)
```

In [25]:
```
def create_fully_connected_layer_and_compute_dotproduct_plus_bias(inputs,output_size):
```

```
    input_size=int(inputs.get_shape()[1])

    # initilaize weight of the filter of the FC layer
    W=initialize_weights([input_size,output_size])

    # initialize the bias: each bias one output channel
    b=initialize_bias([output_size])

    fc_xW_plus_bias_outputs=tf.matmul(inputs,W)+b

    # retrun the results:outputs
    return (fc_xW_plus_bias_outputs)
```

PHASE I: Build Convolutional Neural Network

In [26]:
```
import tensorflow as tf
```

Create Placeholders for Inputs and Labels: x and y_true

In [27]:
```
# PLACEHOLDER

# Create a placeholder for the inputs data: x
# x: a 2D array
# x: a placeholder that can hold any number of rows/record

x = tf.placeholder(tf.float32, shape=[None, 32, 32, 3])
```

```
WARNING:tensorflow:From /tmp/ipykernel_8876/3140936945.py:7: The name tf.placeholder
is deprecated. Please use tf.compat.v1.placeholder instead.
```

In [28]:
```
# PLACEHOLDER

# Create a placeholder for the labels of the inputs data: y_true
# y_true: a 2D array
# y_true: Can hold any number of rows/records

y_true = tf.placeholder(tf.float32, [None, 10])
```

Reshape the Input Placeholder x: NOT NEED TO RESHAPE DATA HERE

In [29]:
```
# DO NOTHING -- DON'T NEED TO RESHAPE - DATA IS ALREADY IN GOOD SHAPE TO BE FED INTO 1
```

Create 1st Convolution Layer and so on

In [30]:
```
# create 1st convolutional Layer, ReLu Layer, and perform computation: X*W+b

# create 1st convolutional Layer
conv_layer_1_outputs=create_convolution_layer_and_compute_dot_product(x,filter_shape=[

# create 1st ReLu Layer
conv_relu_layer_1_outputs=create_relu_layer_compute_dotproduct_plus_b(conv_layer_1_out
```

```
WARNING:tensorflow:From /tmp/ipykernel_8876/2449496641.py:4: The name tf.truncated_no
rmal is deprecated. Please use tf.random.truncated_normal instead.
```

In [31]:
```
# create 1st polling layer and reduce spatial size

pooling_layer_1_outputs=create_maxpool2by2_and_reduce_spatial_size(conv_relu_layer_1_c
```

```
WARNING:tensorflow:From /tmp/ipykernel_8876/3316794812.py:4: The name tf.nn.max_pool
is deprecated. Please use tf.nn.max_pool2d instead.
```

In [32]:
```
# create 2nd convolutional Layer, ReLu Layer, and perform computation: X*W+b

# create 2nd convolutional Layer
conv_layer_2_outputs=create_convolution_layer_and_compute_dot_product(pooling_layer_1_

# create 2nd ReLu Layer
conv_relu_layer_2_outputs=create_relu_layer_compute_dotproduct_plus_b(conv_layer_2_out

# create 2nd polling layer and reduce spatial size

pooling_layer_2_outputs=create_maxpool2by2_and_reduce_spatial_size(conv_relu_layer_2_c
```

In [33]:
```
# reshape and flatten the output of the 1st pooling layer
pooling_layer_2_outputs_flat=tf.reshape(pooling_layer_2_outputs,[-1, 8 * 8 * 64])
```

PHASE II: Train and Test CNN Model on CIFAR-10 Dataset

In [34]:
```
# create 1st FC layer,ReLu Layer, and output Data to dropout layer

fc_layer_1_outputs=create_fully_connected_layer_and_compute_dotproduct_plus_bias(pooli

# create the ReLu layer of the 1st FC layer

fc_relu_layer_1_outputs=tf.nn.relu(fc_layer_1_outputs)
```

In [35]:
```
# create dropout layer and dropout fraction of outputs randomly

# Declare a placeholder to hold the value of probability
hold_prob=tf.placeholder(tf.float32)

# droput
fc_dropout_outputs=tf.nn.dropout(fc_relu_layer_1_outputs,keep_prob=hold_prob)
```

```
WARNING:tensorflow:From /tmp/ipykernel_8876/861275382.py:7: calling dropout (from ten
sorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a futu
re version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob
`.
```

In [36]:
```
# create final FC layer
y_pred=create_fully_connected_layer_and_compute_dotproduct_plus_bias(fc_dropout_output
```

In [37]:
```
# define loss function:cross-entropy with logits i.e with the final outputs

softmax_cross_entropy_loss=tf.nn.softmax_cross_entropy_with_logits(labels=y_true,logit

# compute the mean of loss
cross_entropy_mean=tf.reduce_mean(softmax_cross_entropy_loss)
```

```
WARNING:tensorflow:From /tmp/ipykernel_8876/3944891787.py:3: softmax_cross_entropy_wi
th_logits (from tensorflow.python.ops.nn_ops) is deprecated and will be removed in a
future version.
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See `tf.nn.softmax_cross_entropy_with_logits_v2`.
```

In [38]:
```
# create an optimizer to optimize CNN model and set learning rate
# get an ADAM optimizer

optimizer=tf.train.AdamOptimizer(learning_rate=0.001)
```

```
WARNING:tensorflow:From /tmp/ipykernel_8876/3832092856.py:4: The name tf.train.AdamOp
timizer is deprecated. Please use tf.compat.v1.train.AdamOptimizer instead.
```

In [39]:
```
# create a trainer to traing CNN model
# create a CNN model trainer that can train the model
# and optimize the model by minimizing the softmax cross_entropy loss

cnn_trainer=optimizer.minimize(cross_entropy_mean)
```

In [40]:
```
# create a variable initializer to initialize all variable
vars_initializer=tf.global_variables_initializer()
```

```
WARNING:tensorflow:From /tmp/ipykernel_8876/3029592834.py:2: The name tf.global_varia
bles_initializer is deprecated. Please use tf.compat.v1.global_variables_initializer
instead.
```

CIFAR-10.train: 50000 images

In [41]:
```
# x: CIFAR-10.train: 50000 images
steps = 5000
```

In [42]:
```
with tf.Session() as sess:
    # First, run vars_initializer to initialize
```

```
    sess.run(vars_initializer)

    for i in range(steps):
        # Each batch: 100 images
        batch = ch.next_batch(100)

        # Train the model
        # Dropout keep_prob (% to keep): 0.5 --> 50% will be dropped out
        sess.run(cnn_trainer, feed_dict={x: batch[0], y_true: batch[1], hold_prob: 0.5

        # Test the model: at each 100th step
        # Run this block of code for each 100 times of training, each time run a batch
        if i % 100 == 0:
            print('ON STEP: {}'.format(i))
            print('ACCURACY: ')

            # Compare to find matches of y_pred and y_true
            matches = tf.equal(tf.argmax(y_pred, 1), tf.argmax(y_true, 1))

            # Cast the matches from integers to tf.float32
            # Calculate the accuracy using the mean of matches
            acc = tf.reduce_mean(tf.cast(matches, tf.float32))

            # Test the model at each 100th step
            # Using test dataset
            # Dropout: NONE because of test, not training
            test_accuracy = sess.run (acc, feed_dict = {x:ch.test_images, y_true: ch.t
            print(test_accuracy)
            print('\n')
```

```
WARNING:tensorflow:From /tmp/ipykernel_8876/3303403197.py:1: The name tf.Session is d
eprecated. Please use tf.compat.v1.Session instead.
```

```
User settings:

   KMP_AFFINITY=granularity=fine,verbose,compact,1,0
   KMP_BLOCKTIME=0
   KMP_SETTINGS=1
   OMP_NUM_THREADS=8

Effective settings:

   KMP_ABORT_DELAY=0
   KMP_ADAPTIVE_LOCK_PROPS='1,1024'
   KMP_ALIGN_ALLOC=64
   KMP_ALL_THREADPRIVATE=128
   KMP_ATOMIC_MODE=2
   KMP_BLOCKTIME=0
   KMP_CPUINFO_FILE: value is not defined
   KMP_DETERMINISTIC_REDUCTION=false
   KMP_DEVICE_THREAD_LIMIT=2147483647
   KMP_DISP_HAND_THREAD=false
   KMP_DISP_NUM_BUFFERS=7
   KMP_DUPLICATE_LIB_OK=false
   KMP_FORCE_REDUCTION: value is not defined
   KMP_FOREIGN_THREADS_THREADPRIVATE=true
   KMP_FORKJOIN_BARRIER='2,2'
   KMP_FORKJOIN_BARRIER_PATTERN='hyper,hyper'
   KMP_FORKJOIN_FRAMES=true
   KMP_FORKJOIN_FRAMES_MODE=3
   KMP_GTID_MODE=3
   KMP_HANDLE_SIGNALS=false
   KMP_HOT_TEAMS_MAX_LEVEL=1
   KMP_HOT_TEAMS_MODE=0
   KMP_INIT_AT_FORK=true
   KMP_ITT_PREPARE_DELAY=0
   KMP_LIBRARY=throughput
   KMP_LOCK_KIND=queuing
   KMP_MALLOC_POOL_INCR=1M
   KMP_MWAIT_HINTS=0
   KMP_NUM_LOCKS_IN_BLOCK=1
   KMP_PLAIN_BARRIER='2,2'
   KMP_PLAIN_BARRIER_PATTERN='hyper,hyper'
   KMP_REDUCTION_BARRIER='1,1'
   KMP_REDUCTION_BARRIER_PATTERN='hyper,hyper'
   KMP_SCHEDULE='static,balanced;guided,iterative'
   KMP_SETTINGS=true
   KMP_SPIN_BACKOFF_PARAMS='4096,100'
   KMP_STACKOFFSET=64
   KMP_STACKPAD=0
   KMP_STACKSIZE=8M
   KMP_STORAGE_MAP=false
   KMP_TASKING=2
   KMP_TASKLOOP_MIN_TASKS=0
   KMP_TASK_STEALING_CONSTRAINT=1
   KMP_TEAMS_THREAD_LIMIT=8
   KMP_TOPOLOGY_METHOD=all
   KMP_USER_LEVEL_MWAIT=false
   KMP_USE_YIELD=1
   KMP_VERSION=false
   KMP_WARNINGS=true
   OMP_AFFINITY_FORMAT='OMP: pid %P tid %i thread %n bound to OS proc set {%A}'
   OMP_ALLOCATOR=omp_default_mem_alloc
```

```
   OMP_CANCELLATION=false
   OMP_DEBUG=disabled
   OMP_DEFAULT_DEVICE=0
   OMP_DISPLAY_AFFINITY=false
   OMP_DISPLAY_ENV=false
   OMP_DYNAMIC=false
   OMP_MAX_ACTIVE_LEVELS=2147483647
   OMP_MAX_TASK_PRIORITY=0
   OMP_NESTED=false
   OMP_NUM_THREADS='8'
   OMP_PLACES: value is not defined
   OMP_PROC_BIND='intel'
   OMP_SCHEDULE='static'
   OMP_STACKSIZE=8M
   OMP_TARGET_OFFLOAD=DEFAULT
   OMP_THREAD_LIMIT=2147483647
   OMP_TOOL=enabled
   OMP_TOOL_LIBRARIES: value is not defined
   OMP_WAIT_POLICY=PASSIVE
   KMP_AFFINITY='verbose,warnings,respect,granularity=fine,compact,1,0'

2024-03-08 18:30:12.115327: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94]
CPU Frequency: 2200150000 Hz
2024-03-08 18:30:12.116344: I tensorflow/compiler/xla/service/service.cc:168] XLA ser
vice 0x5556b918cb40 initialized for platform Host (this does not guarantee that XLA w
ill be used). Devices:
2024-03-08 18:30:12.116378: I tensorflow/compiler/xla/service/service.cc:176]   Strea
mExecutor device (0): Host, Default Version
2024-03-08 18:30:12.116498: I tensorflow/core/common_runtime/process_util.cc:136] Cre
ating new thread pool with default inter op setting: 2. Tune using inter_op_paralleli
sm_threads for best performance.
OMP: Info #212: KMP_AFFINITY: decoding x2APIC ids.
OMP: Info #210: KMP_AFFINITY: Affinity capable, using global cpuid leaf 11 info
OMP: Info #154: KMP_AFFINITY: Initial OS proc set respected: 0-7
OMP: Info #156: KMP_AFFINITY: 8 available OS procs
OMP: Info #157: KMP_AFFINITY: Uniform topology
OMP: Info #179: KMP_AFFINITY: 1 packages x 4 cores/pkg x 2 threads/core (4 total core
s)
OMP: Info #214: KMP_AFFINITY: OS proc to physical thread map:
OMP: Info #171: KMP_AFFINITY: OS proc 0 maps to package 0 core 0 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 4 maps to package 0 core 0 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 1 maps to package 0 core 1 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 5 maps to package 0 core 1 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 2 maps to package 0 core 2 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 6 maps to package 0 core 2 thread 1
OMP: Info #171: KMP_AFFINITY: OS proc 3 maps to package 0 core 3 thread 0
OMP: Info #171: KMP_AFFINITY: OS proc 7 maps to package 0 core 3 thread 1
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9070 thread 0 bound to OS proc set 0
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9070 thread 1 bound to OS proc set 1
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9073 thread 2 bound to OS proc set 2
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9074 thread 3 bound to OS proc set 3
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9075 thread 4 bound to OS proc set 4
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9076 thread 5 bound to OS proc set 5
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9077 thread 6 bound to OS proc set 6
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9079 thread 8 bound to OS proc set 0
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9078 thread 7 bound to OS proc set 7
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9071 thread 9 bound to OS proc set 1
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9082 thread 12 bound to OS proc set 4
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9083 thread 13 bound to OS proc set 5
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9084 thread 14 bound to OS proc set 6
```

```
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9080 thread 10 bound to OS proc set 2
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9081 thread 11 bound to OS proc set 3
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9085 thread 15 bound to OS proc set 7
OMP: Info #250: KMP_AFFINITY: pid 8876 tid 9086 thread 16 bound to OS proc set 0
```

```
ON STEP: 0
ACCURACY:
0.1


ON STEP: 100
ACCURACY:
0.368


ON STEP: 200
ACCURACY:
0.4066


ON STEP: 300
ACCURACY:
0.4594


ON STEP: 400
ACCURACY:
0.4691


ON STEP: 500
ACCURACY:
0.5002


ON STEP: 600
ACCURACY:
0.5264


ON STEP: 700
ACCURACY:
0.5269


ON STEP: 800
ACCURACY:
0.5449


ON STEP: 900
ACCURACY:
0.557


ON STEP: 1000
ACCURACY:
0.5746


ON STEP: 1100
ACCURACY:
0.5835
```

```
ON STEP: 1200
ACCURACY:
0.5967


ON STEP: 1300
ACCURACY:
0.6046


ON STEP: 1400
ACCURACY:
0.6084


ON STEP: 1500
ACCURACY:
0.6133


ON STEP: 1600
ACCURACY:
0.6334


ON STEP: 1700
ACCURACY:
0.626


ON STEP: 1800
ACCURACY:
0.6337


ON STEP: 1900
ACCURACY:
0.6433


ON STEP: 2000
ACCURACY:
0.6485


ON STEP: 2100
ACCURACY:
0.6497


ON STEP: 2200
ACCURACY:
0.6596


ON STEP: 2300
ACCURACY:
0.6536
```

```
ON STEP: 2400
ACCURACY:
0.6552


ON STEP: 2500
ACCURACY:
0.6258


ON STEP: 2600
ACCURACY:
0.6611


ON STEP: 2700
ACCURACY:
0.6663


ON STEP: 2800
ACCURACY:
0.6516


ON STEP: 2900
ACCURACY:
0.668


ON STEP: 3000
ACCURACY:
0.6487


ON STEP: 3100
ACCURACY:
0.6681


ON STEP: 3200
ACCURACY:
0.6722


ON STEP: 3300
ACCURACY:
0.6698


ON STEP: 3400
ACCURACY:
0.683


ON STEP: 3500
ACCURACY:
0.6709
```

```
ON STEP: 3600
ACCURACY:
0.6735


ON STEP: 3700
ACCURACY:
0.6777


ON STEP: 3800
ACCURACY:
0.675


ON STEP: 3900
ACCURACY:
0.6773


ON STEP: 4000
ACCURACY:
0.6736


ON STEP: 4100
ACCURACY:
0.6757


ON STEP: 4200
ACCURACY:
0.6802


ON STEP: 4300
ACCURACY:
0.68


ON STEP: 4400
ACCURACY:
0.6782


ON STEP: 4500
ACCURACY:
0.681


ON STEP: 4600
ACCURACY:
0.6782


ON STEP: 4700
ACCURACY:
0.6846
```

```
ON STEP: 4800
ACCURACY:
0.6716


ON STEP: 4900
ACCURACY:
0.6882
```

In [ ]: