COMPSCI - 433.7 Machine Learning with TensorFlow - Final Project

Sharmistha Maitra

PROJECT

Image classification for CIFAR-10 dataset
using CNN in TensorFlow

CIFAR-10 dataset image classification using TF

CIFAR-10 dataset is a well known dataset for image classification

Url: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

Contains 60,000 color images: 32 x 32 pixels, 3 channels (RGB)

CIFAR-10 dataset image classification using TF

Ten classes of images

0: airplane
1: automobile
2: bird
3: cat
4: deer
5: dog
6: frog
7: horse
8: ship
9: truck

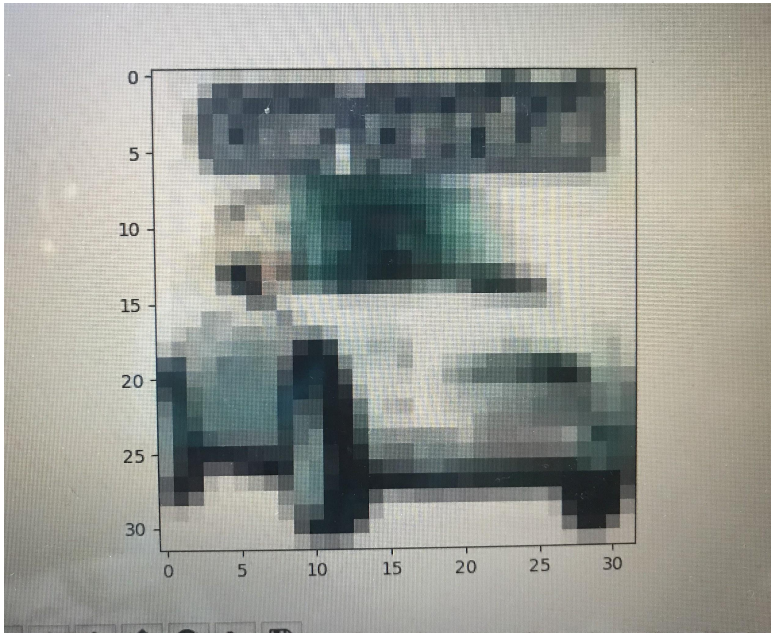
CIFAR-10 dataset image classification using TF

Entire dataset https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

Extracted into 6 smaller datasets (in a folder under working directory) .
5 datasets for training and 1 dataset for testing(each contains 10,000 images)

- Data_batch_1
- Data_batch_2
- Data_batch_3
- Data_batch_4
- Data_batch_5
- Test_batch

Exploring images(32x32 pixels) in CIFAR-10 dataset



First image
display_feature_label(features, labels, 0)
Label: automobile

(using plt.imshow to display the image)

(exploring data_batch_1)

Exploring images(32x32 pixels) in CIFAR-10 dataset

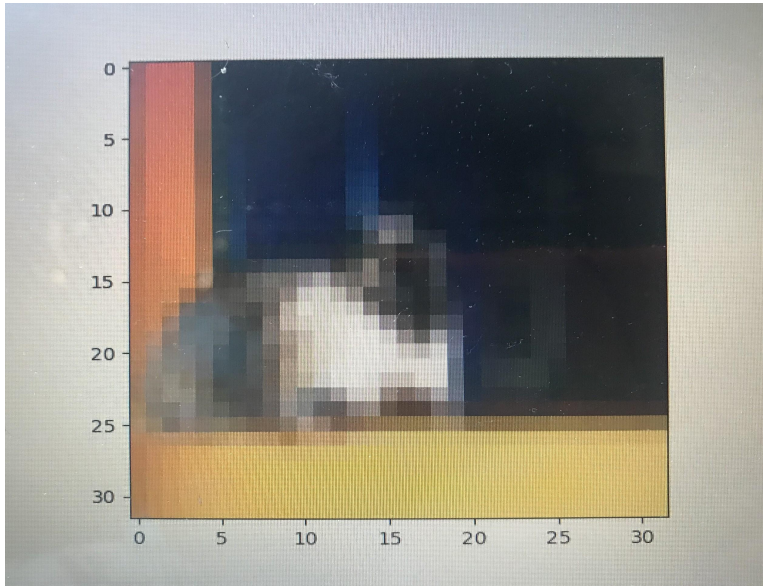


Image at index 7 in data_batch_1

display_feature_label(features, labels, 7)
Label: cat

(using plt.imshow to display the image)

(exploring data_batch_1)

Exploring images(32x32 pixels) in CIFAR-10 dataset

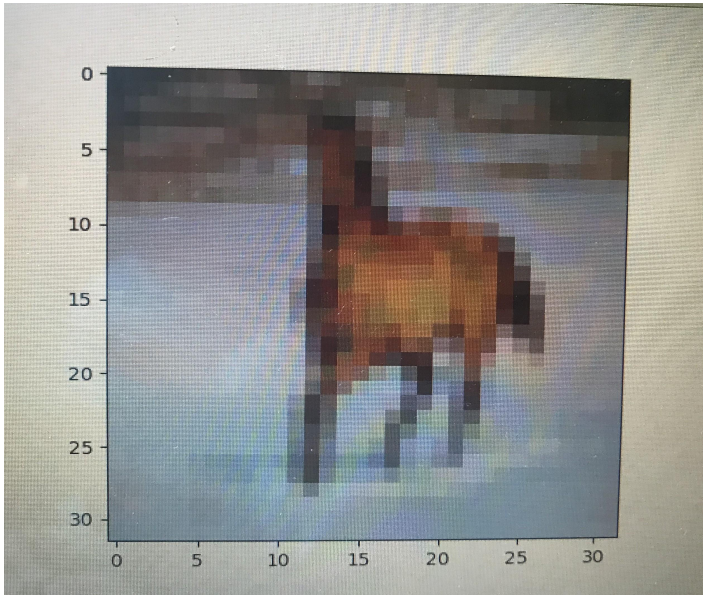


Image at index 10 in data_batch_1

display_feature_label(features, labels, 10)
Label: horse

(using plt.imshow to display the image)

(exploring data_batch_1)

CNN architecture - 1

Input images(?, 32, 32, 3) => conv1(?, 32, 32, 32) => conv2(?, 16, 16, 64) =>

pool3(?, 8, 8, 64) => conv4(?, 3, 3, 128) => pool5(?, 2, 2, 128) =>

pool5_flat(?, 512) => fully_conn1(?, 128) => fully_con2(?, 64) => logits_layer(?, 10)

Two convolution layers in the beginning to build better representation of data

CNN architecture - 2

Input images(?, 32, 32, 3) => dropout_layer (shut off x% of neurons) =>

conv1(?, 32, 32, 32) => conv2(?, 16, 16, 64) =>

pool3(?, 8, 8, 64) => conv4(?, 3, 3, 128) =>

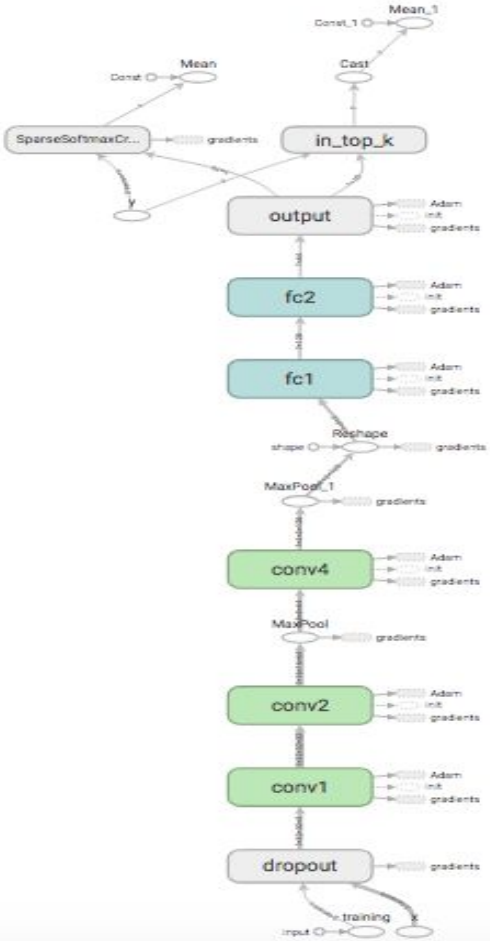
pool5(?, 2, 2, 128) => pool5_flat(?, 512) =>

fully_conn1(?, 128) => fully_con2(?, 64) => logits_layer(?, 10)

Two convolution layers in the beginning to build better representation of data

Dropout layer shuts off 30% of neurons, forces other neurons to learn new features , reduce overfitting on training data

Main Graph



Auxiliary Nodes

Accuracy metrics: 1 conv layer (after input layer)

0 Train accuracy: 0.16 Test_accuracy: 0.154

1 Train accuracy: 0.265 Test_accuracy: 0.244

2 Train accuracy: 0.345 Test_accuracy: 0.3285

3 Train accuracy: 0.44 Test_accuracy: 0.3665

4 Train accuracy: 0.475 Test_accuracy: 0.404

5 Train accuracy: 0.49 Test_accuracy: 0.4115

6 Train accuracy: 0.555 Test_accuracy: 0.433

7 Train accuracy: 0.595 Test_accuracy: 0.4495

8 Train accuracy: 0.61 Test_accuracy: 0.4435

9 Train accuracy: 0.61 Test_accuracy: 0.4535

After 10 epochs,
Train accuracy: 0.61 Test_accuracy: 0.4535

Accuracy metrics: 2 conv layers (after input layer)

0 Train accuracy: 0.275 Test_accuracy: 0.23

1 Train accuracy: 0.35 Test_accuracy: 0.3455

2 Train accuracy: 0.465 Test_accuracy: 0.4015

3 Train accuracy: 0.55 Test_accuracy: 0.459

4 Train accuracy: 0.63 Test_accuracy: 0.472

5 Train accuracy: 0.665 Test_accuracy: 0.477

6 Train accuracy: 0.73 Test_accuracy: 0.482

7 Train accuracy: 0.71 Test_accuracy: 0.478

8 Train accuracy: 0.73 Test_accuracy: 0.4645

9 Train accuracy: 0.835 Test_accuracy: 0.5095

After 10 epochs,
Train accuracy: 0.835, Test_accuracy: 0.5095

Accuracy metrics: batch size 200

######## batch_size = 200, train_size = 8000. Model trained over 8000/200 = 40 iterations. MOdel run on data_batch_1 ########

0 Train accuracy: 0.195 Test_accuracy: 0.2085

1 Train accuracy: 0.37 Test_accuracy: 0.3735

2 Train accuracy: 0.47 Test_accuracy: 0.411

3 Train accuracy: 0.53 Test_accuracy: 0.435

4 Train accuracy: 0.585 Test_accuracy: 0.456

5 Train accuracy: 0.635 Test_accuracy: 0.4685

6 Train accuracy: 0.66 Test_accuracy: 0.445

7 Train accuracy: 0.635 Test_accuracy: 0.4335

8 Train accuracy: 0.715 Test_accuracy: 0.4425

9 Train accuracy: 0.795 Test_accuracy: 0.4635

After 10 epochs,
Train accuracy: 0.795 Test_accuracy: 0.4635

Accuracy metrics: batch size 100

######## batch_size = 100, train_size = 8000. Model trained over 8000/100 = 80 iterations. MOdel run on data_batch_1 ########

0 Train accuracy: 0.37 Test_accuracy: 0.316

1 Train accuracy: 0.52 Test_accuracy: 0.4115

2 Train accuracy: 0.49 Test_accuracy: 0.379

3 Train accuracy: 0.54 Test_accuracy: 0.4375

4 Train accuracy: 0.64 Test_accuracy: 0.445

5 Train accuracy: 0.67 Test_accuracy: 0.4595

6 Train accuracy: 0.68 Test_accuracy: 0.4495

7 Train accuracy: 0.71 Test_accuracy: 0.4755

8 Train accuracy: 0.73 Test_accuracy: 0.482

9 Train accuracy: 0.78 Test_accuracy: 0.453

After 10 epochs:
Not much improvement on accuracy.

Observation: test _accuracy much lower than train_accuracy. Could be because of over fitting ??

Accuracy metrics: 30% neuron dropouts after input layer

0 Train accuracy: 0.17 Test_accuracy: 0.1505
1 Train accuracy: 0.27 Test_accuracy: 0.2395
2 Train accuracy: 0.25 Test_accuracy: 0.234
3 Train accuracy: 0.22 Test_accuracy: 0.271
4 Train accuracy: 0.27 Test_accuracy: 0.303
5 Train accuracy: 0.4 Test_accuracy: 0.291
6 Train accuracy: 0.37 Test_accuracy: 0.3275
7 Train accuracy: 0.41 Test_accuracy: 0.321
8 Train accuracy: 0.39 Test_accuracy: 0.3465
9 Train accuracy: 0.42 Test_accuracy: 0.3565
10 Train accuracy: 0.43 Test_accuracy: 0.3565
11 Train accuracy: 0.46 Test_accuracy: 0.3875
12 Train accuracy: 0.45 Test_accuracy: 0.3745
13 Train accuracy: 0.52 Test_accuracy: 0.373
14 Train accuracy: 0.46 Test_accuracy: 0.4005
15 Train accuracy: 0.54 Test_accuracy: 0.427
16 Train accuracy: 0.53 Test_accuracy: 0.4235
17 Train accuracy: 0.5 Test_accuracy: 0.4115
18 Train accuracy: 0.5 Test_accuracy: 0.39
19 Train accuracy: 0.55 Test_accuracy: 0.403
20 Train accuracy: 0.62 Test_accuracy: 0.4505
21 Train accuracy: 0.6 Test_accuracy: 0.439

After 22 epochs,

Test_accuracy is close to train_accuracy. Over fitting taken care of. Shutting off some neurons forced other neurons to learn new features.

Accuracy metrics: 2 conv layers, 30% neuron dropout, 22 epochs, trained over all 5 data batches

0 Train accuracy: 0.26 Test_accuracy: 0.2505
1 Train accuracy: 0.39 Test_accuracy: 0.347
2 Train accuracy: 0.43 Test_accuracy: 0.3865
3 Train accuracy: 0.53 Test_accuracy: 0.438
4 Train accuracy: 0.52 Test_accuracy: 0.454
5 Train accuracy: 0.55 Test_accuracy: 0.4705
6 Train accuracy: 0.54 Test_accuracy: 0.4925
7 Train accuracy: 0.52 Test_accuracy: 0.4875
8 Train accuracy: 0.59 Test_accuracy: 0.505
9 Train accuracy: 0.55 Test_accuracy: 0.5075
10 Train accuracy: 0.63 Test_accuracy: 0.5245
11 Train accuracy: 0.59 Test_accuracy: 0.528
12 Train accuracy: 0.58 Test_accuracy: 0.5205
13 Train accuracy: 0.56 Test_accuracy: 0.52
14 Train accuracy: 0.63 Test_accuracy: 0.531
15 Train accuracy: 0.52 Test_accuracy: 0.494
16 Train accuracy: 0.57 Test_accuracy: 0.532
17 Train accuracy: 0.5 Test_accuracy: 0.4815
18 Train accuracy: 0.6 Test_accuracy: 0.5455
19 Train accuracy: 0.53 Test_accuracy: 0.516
20 Train accuracy: 0.55 Test_accuracy: 0.5075
21 Train accuracy: 0.54 Test_accuracy: 0.512
22 Train accuracy: 0.54 Test_accuracy: 0.5195
23 Train accuracy: 0.5 Test_accuracy: 0.504
24 Train accuracy: 0.55 Test_accuracy: 0.5255

The highest accuracy achieved was 52%