CIFAR-10 based Similarity Analysis

Evaluation Assignment

Submitted by: Hassan Rasheed

Position: Machine Learning Engineer

Contents

Implementation Details:

Model: The model used is SqueezeNet (Forrest N. Iandola, 2017) in the form of a Siamese Neural Network (Gregory Koch, 2015).

Platform: Google Colaboratory has been used to train and test the model. (The code is also hosted there)

Dataset Used: CIFAR-10

Classes: Automobile, Dog, Horse

DL Frameworks: Keras, Tensorflow

```
import keras, tensorflow as tf
```

Metric Used to Evaluate Distances between the output embeddings: Euclidean Distance

```
def euclidean_distance(vects):
    x,y = vects
    sum_square = K.sum(K.square(x-y), axis=1, keepdims=True)
    return K.sqrt(K.maximum(sum_square, K.epsilon()))
```

Loss Function: Contrastive Loss

```
def contrastive_loss(y_true, y_pred):
    margin = 1
    square_pred = K.square(y_pred)
    margin_square = K.square(K.maximum(margin - y_pred, 0))
    return K.mean(y_true * square_pred + (1-y_true) * margin_square)
```

Flow of the Implementation:

Data is loaded from Keras Datasets:

```
(x_traino, y_traino), (x_testo, y_testo) = cifar10.load_data()
```

The required classes are filtered:

```
for i in range (len(y_traino)):
    if y_traino[i] == 1:
      y_traino[i] = 0
      x_train.append(x_traino[i])
      y_train.append(y_traino[i])
    if y_traino[i] == 5:
      y_traino[i] = 1
      x_train.append(x_traino[i])
      y_train.append(y_traino[i])
    if y_traino[i] == 7:
      y_traino[i] = 2
      x_train.append(x_traino[i])
      y_train.append(y_traino[i])

for i in range (len(y_testo)):
    if y_testo[i] == 1:
      y_testo[i] = 0
      x_test.append(x_testo[i])
      y_test.append(y_testo[i])
    if y_testo[i] == 5:
      y_testo[i] = 1
      x_test.append(x_testo[i])
      y_test.append(y_testo[i])
    if y_testo[i] == 7:
      y_testo[i] = 2
      x_test.append(x_testo[i])
      y_test.append(y_testo[i])
```

Positive and Negative pairs are created for the dataset:

```
digit_indices = [np.where(y_train == i)[0] for i in range(num_classes)]
tr_pairs, tr_y = create_pairs(x_train, digit_indices)

digit_indices = [np.where(y_test == i)[0] for i in range(num_classes)]
te_pairs, te_y = create_pairs(x_test, digit_indices)
```

Network is defined:

```
# network definition
base_network = create_base_network(input_shape)

input_a = Input(shape=input_shape, name='input_a')
input_b = Input(shape=input_shape, name='input_b')

# because we re-use the same instance `base_network`,
# the weights of the network
# will be shared across the two branches
processed_a = base_network(input_a)
processed_b = base_network(input_b)

distance = Lambda(euclidean_distance,
                  output_shape=eucl_dist_output_shape)([processed_a, processed_b])

model = Model([input_a, input_b], distance)
```

Model Summary

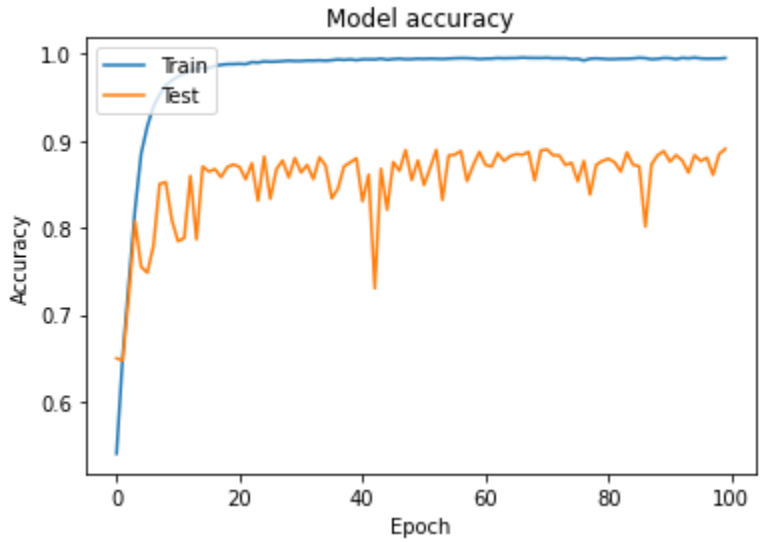
```
model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_a (InputLayer)            [(None, 32, 32, 3)]  0
__________________________________________________________________________________________________
input_b (InputLayer)            [(None, 32, 32, 3)]  0
__________________________________________________________________________________________________
model (Model)                   (None, 10)           378914      input_a[0][0]
                                                                 input_b[0][0]
__________________________________________________________________________________________________
lambda (Lambda)                 (None, 1)            0           model[1][0]
                                                                 model[2][0]
==================================================================================================
Total params: 378,914
Trainable params: 378,350
Non-trainable params: 564
__________________________________________________________________________________________________
```

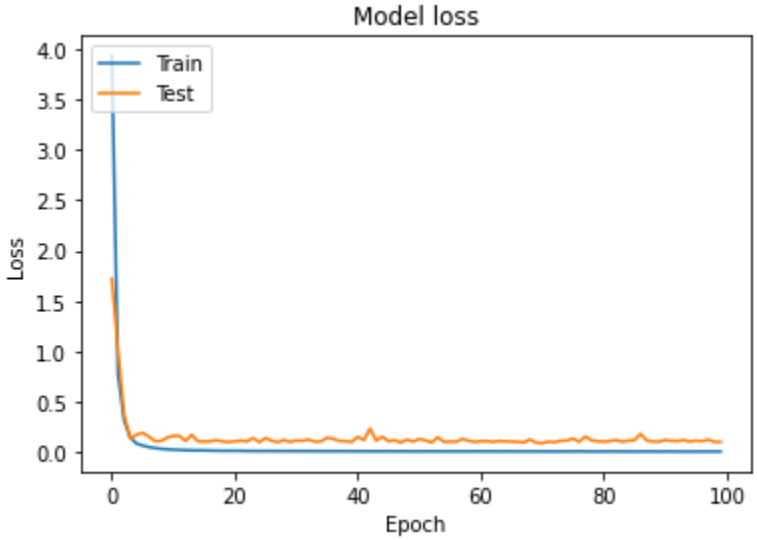
Model Training:

```
# train
rms = RMSprop()
model.compile(loss=contrastive_loss, optimizer=rms, metrics=[accuracy])
checkpointer = ModelCheckpoint(filepath='model.hfs5', save_best_only=True)
history = model.fit([tr_pairs[:, 0], tr_pairs[:, 1]],
                    tr_y, batch_size=128, epochs=epochs,
                    validation_data=([te_pairs[:, 0], te_pairs[:, 1]], te_y),
                    callbacks=[checkpointer])
```

Training Results:



Accuracy on the training set is 99.54%
Accuracy on the test set is 89.08%



Loss is 0.0075 after the 100th Epoch.

Testing with Image Input:

After navigating to the project root directory enter the command in the following format to get similar images from the test dataset.

```
python script.py --image_path <image_path> --data_path <data_path>
```

or

```
python script.py –i <image_path> -d <data_path>
```

Implementations can be accessed:

At Google Colab:

https://drive.google.com/drive/folders/1k9Jyc1tbysPUT41bW5sD-jqfvP7Yu1hR?usp=sharing

On GitHub:

https://github.com/hassanrasheedk/cifar10-similarity-analysis

Through the Google Drive:

https://drive.google.com/drive/folders/1dzVhjYVNRClUNgVk9SrF6mbq2w9AmDX7?usp=sharing

References

Forrest N. Iandola, S. H. (2017). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *International Conference on Learning Representations.*

Gregory Koch, R. Z. (2015). Siamese Neural Networks for One-shot Image Recognition. *International Conference on Machine Learning.* Lille.

Keras CIFAR: https://keras.io/api/datasets/cifar10/

CIFAR-10: https://www.cs.toronto.edu/~kriz/cifar.html