COEN 166L: Assignment #6

1. Python Practice
2. Vacuum World
3. Pac-man I
4. Pac-man II
5. Face Recognition
6. **Neural Network**

No Demo;
Report & code due by Wed,12/11

Overview

- ## Problem 1- Recognition

Build a CNN network for recognition with the fashion MNIST dataset

1. **Epochs = 5.**
2. Error function is **cross-entropy**
3. Give the **recognition accuracy rate** and show the **confusion matrix**, both for the test set.

- ## Problem 2 - Image Compression

Implement an image compression system using a neural network.

1. **Epochs=10, Batch size=64.**
2. The loss/error function is the **mean-squared-error (mse)**
3. Train with different P values, P = 10, 50, 200
4. Calculate **PSNR**
5. Display the images compressed and without compressed

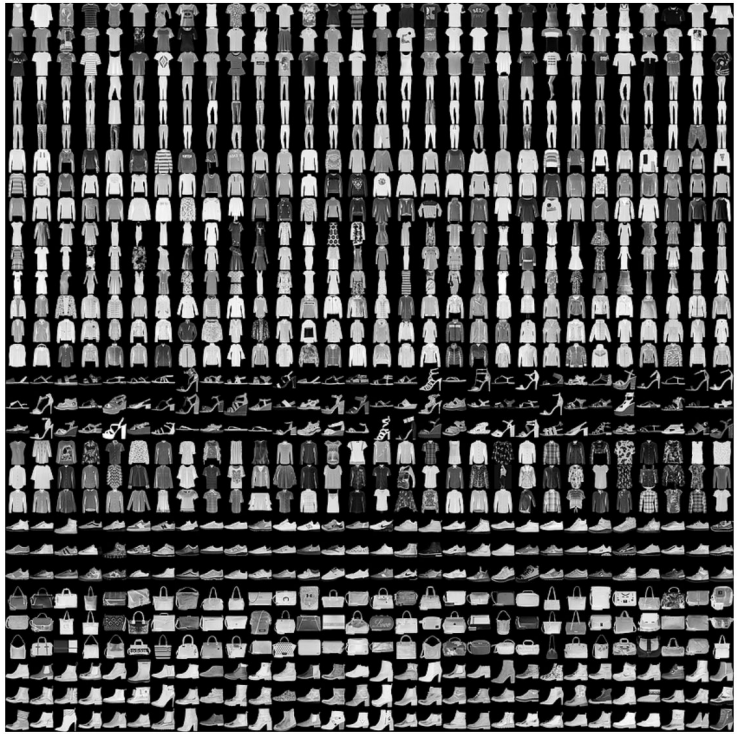
Fashion-MNIST

Star 6.9k | chat on gitter | README 中文 | README 日本語 | License MIT | Year in Review

▶ Table of Contents

`Fashion-MNIST` is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. We intend `Fashion-MNIST` to serve as a direct **drop-in replacement** for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

Here's an example how the data looks (*each class takes three-rows*):



https://github.com/zalandoresearch/fashion-mnist

Pre-work

Packages needed:

- Tensorflow (Keras)
- Numpy
- Sklearn
- matplotlib

Implementation Problem 1: Overall Structure

```python
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from sklearn.metrics import confusion_matrix

fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0


####------decide the model layers------
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)) # the 1st 2d-convolutional layer
# model.add()
# model.add()
# model.add()
# ... to be completed by yourself



#####------ compile the model ---------
model.compile()


### ------ evaluate the model ---------
model.evaluate()


#####------ train the model ------------
model.fit()


#####------ predict with the model ---------
model.predict()


#####------ calculate the confusion matrix -------
confusion_matrix()
```

Adopt the following convolutional neural network structure:

1. Input layer
2. 2d-convolutional layer: filter size 3x3, depth=32, ReLU activation function
3. 2x2 max pooling layer
4. 2d-convolutional layer: filter size 3x3, depth=64, ReLU activation function
5. 2x2 max pooling layer
6. 2d-convolutional layer: filter size 3x3, depth=64, ReLU activation function
7. Fully-connected layer: 64 units, ReLU activation function
8. (output) Fully-connected layer: 10 units, softmax activation function

(Refer to Lab assignment 6.pdf)

How to construct a neural network (CNN) ?

mnist_cnn.py

Refer to ‘mnist_cnn.py’ or mnist_nn.py’ about how to construct a CNN or neural network.
(Camino -> lecture ->camino->lecture session-> lecture slides folder)

model = model.**Sequential**()
model.**add**()
model.**add**()
model.**add**()
….

model.**summary**()

model.**compile**(optimizer = ‘ ’, loss=’ ‘ , metrics =‘ ‘)

model.**fit(** train_images, train_labels, epochs = 5 ,batch_size= 64)

loss, acc = model.**evaluate**(test_images, test_labels)

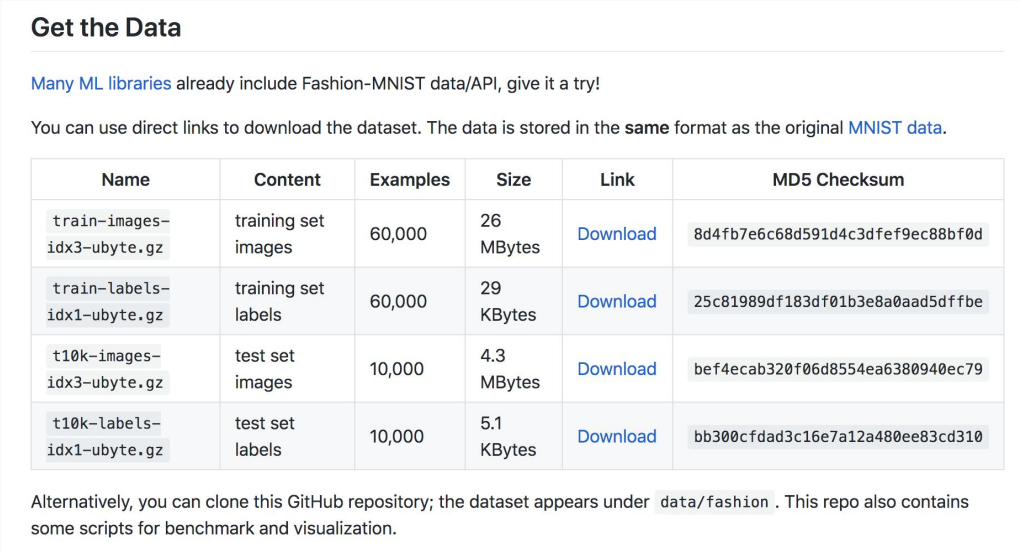
y_test_hat = model.**predict** (test_images)

```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import numpy as np

(train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0


model = models.Sequential()
# 32: number of filters
# (3,3): filter size
# batch size: default: 32
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

#model.summary()

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.summary()

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

model.fit(train_images, train_labels, epochs=5,batch_size=64)

test_loss, test_acc = model.evaluate(test_images, test_labels)

print(test_acc)

y_test_hat_mat = model.predict(test_images)
y_test_hat = np.argmax(y_test_hat_mat, axis=1)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(test_labels, y_test_hat, labels=range(10))
```

You may need to (probably not in this lab):

1.

If download using `fashion_mnist.load_data()` is too slow, try to download **manually** from the website, but you need to write some code to load the downloaded .gz data file



Get the Data

Many ML libraries already include Fashion-MNIST data/API, give it a try!

You can use direct links to download the dataset. The data is stored in the **same** format as the original MNIST data.

Name	Content	Examples	Size	Link	MD5 Checksum
`train-images-idx3-ubyte.gz`	training set images	60,000	26 MBytes	Download	`8d4fb7e6c68d591d4c3dfef9ec88bf0d`
`train-labels-idx1-ubyte.gz`	training set labels	60,000	29 KBytes	Download	`25c81989df183df01b3e8a0aad5dffbe`
`t10k-images-idx3-ubyte.gz`	test set images	10,000	4.3 MBytes	Download	`bef4ecab320f06d8554ea6380940ec79`
`t10k-labels-idx1-ubyte.gz`	test set labels	10,000	5.1 KBytes	Download	`bb300cfdad3c16e7a12a480ee83cd310`

Alternatively, you can clone this GitHub repository; the dataset appears under `data/fashion`. This repo also contains some scripts for benchmark and visualization.

https://github.com/zalandoresearch/fashion-mnist

2.

Save the model :
model.save

Load the model for the next time usage:
model.load

Can also serialize to other format:

- JSON HD5
- YAML HD5

https://machinelearningmastery.com/save-load-keras-deep-learning-models/

Implementation Problem 2: Overall Structure

The network includes compressed + decompressed, so the workflow is:

Original image - > network(first compress, then decompress) -> output decompressed image

Overall structure is similar to Problem 1

But some differences:

1. model.compile( )
2. model.fit() # input and output are both train images
3. model.predict() # in problem 2: predicted output is decompressed images
   (in problem 1: prediction output is probability of each class)
4. need to calculate PSNR between test_images and test_decompressed_images