Summary Report

2019/08/09

Tianfangtong Zhang

Index

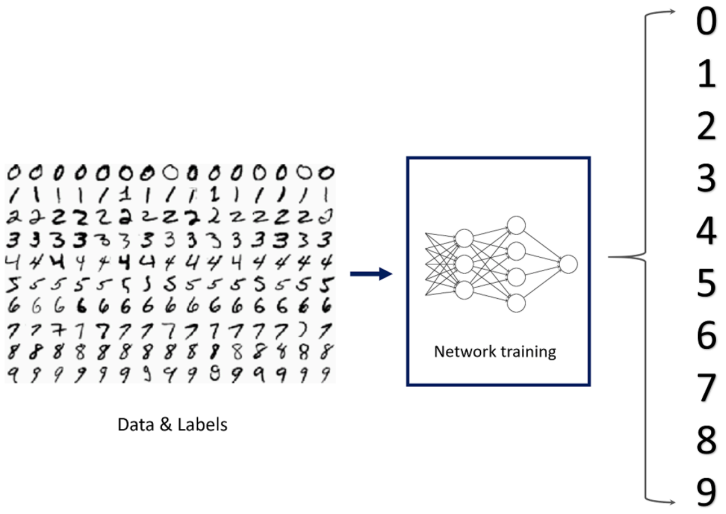
0 Project Background

0.1 MNIST

The MNIST dataset consists of handwritten digit images (from 0 to 9) which include 60,000 examples of training set and 10,000 examples of testing set. It is a subset of a larger set available from NIST. However, the official training set of 60,000 examples is divided into an actual training set of 50,000 examples and 10,000 validation examples. Additionally all digit images are black and white and sized-normalized and with a fix size image of 28 x 28 pixels. In general, each digit image is represented by a value between 255 and 0, such that 255 is black and 0 is white and anything in between is a different degree of grey.

This dataset is combined by two of NIST's database: Special Database 1 and Special Database 3. These two databases includes digits hand-written by high school students and employees of the United States Census Bureau.

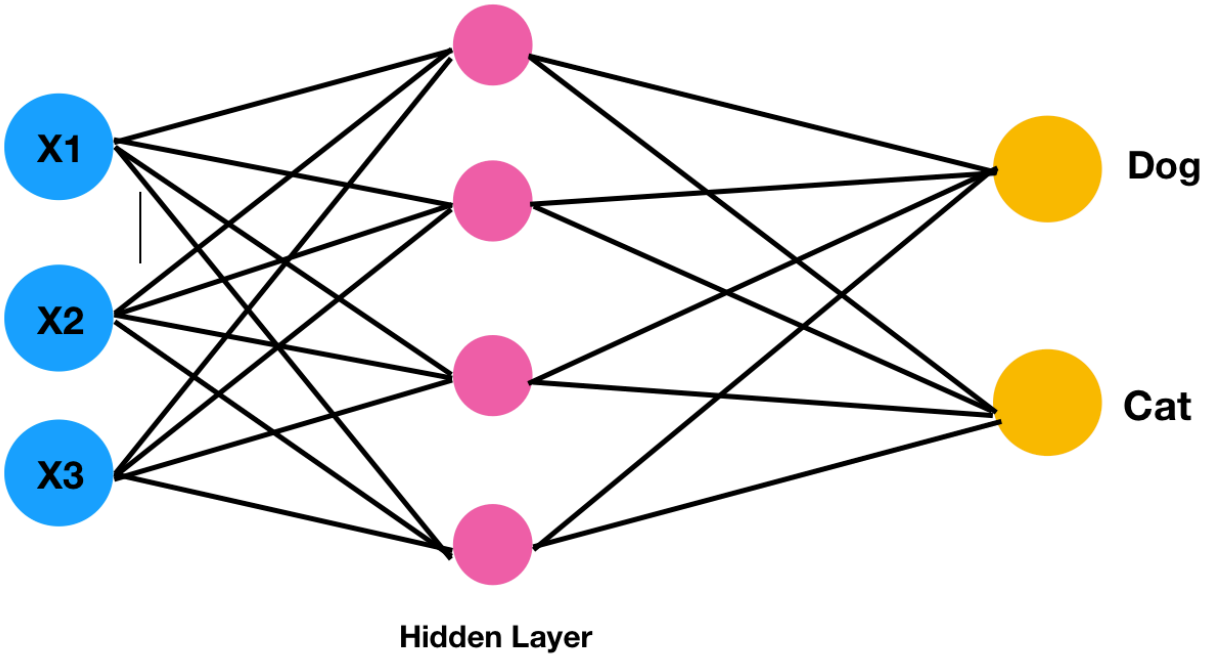
MNIST dataset is widely used in machine learning and image processing. On the other hand, this dataset is used by researchers to test and compare their research results with others.



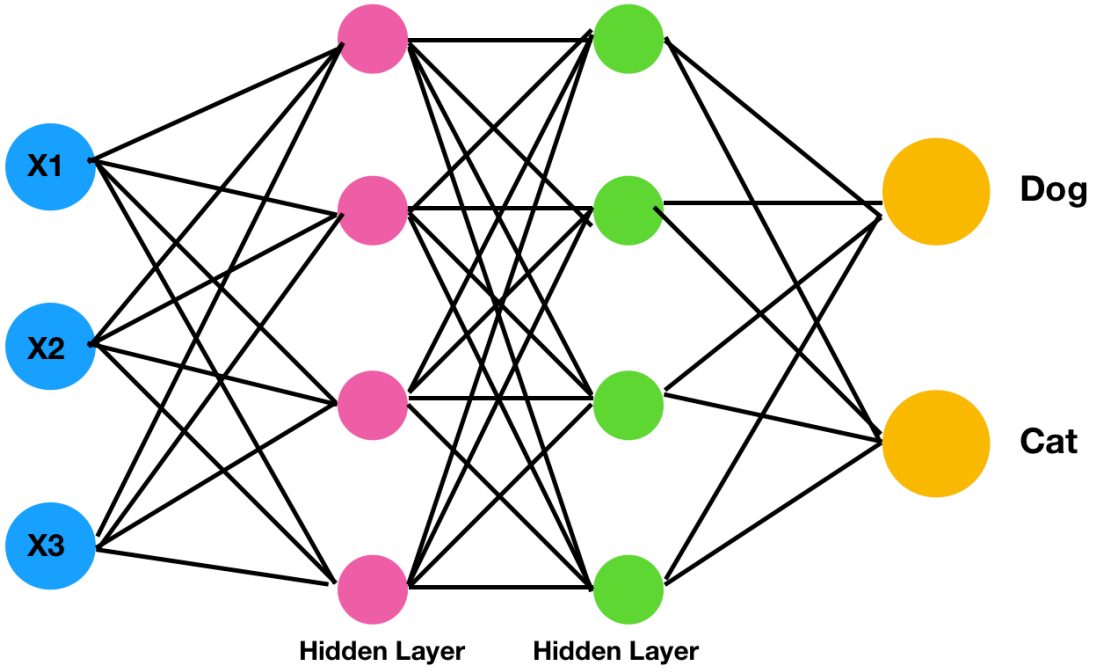
1 Project Knowledge Review

1. 1 Neural network

Let's have a brief overview of how neural networks work.



Suppose we get some input X1, X2 and X3. We try to map these three inputs two two outputs, cat and dog. In these case cat and dog are two neurones. We could use a single hidden layer. Then each of the input data connected to each of the neutrons in that first hidden layer, and each of these connection has its own unique weight. Even though we can connect the first hidden layer to the output layer, here is the problem that the relationship between X1 and cat or dog all the other inputs would only be linear relationships. Thus, if we're looking forward to map a nonlinear relationships which is probably going to be the case in a complex question. Then we need to have two or more Hidden Layer. Here we know that



one Hidden Layer means we just have a neural network, and two or more Hidden Layers means we have a quote-unquote deep neural network.

Then after these neurones fully connected all unique weights, each of those links (black lines) has a unique weight associated with it. Let's say at the final output layer, this output layer is almost certain to have just a sigmoid activation function and then we are going to say the cat is 0.81 and the dog is 0.19. Thus, in this case we have 81% confidence it is a cat and 19% confidence it is a dog. Then we think it is a cat.

1.2 TensorFlow

“TensorFlow is an open-source machine learning library for research and production. TensorFlow offers APIs for beginners and experts to develop for desktop, mobile, web and cloud.”

First make sure we have installed TensorFlow in our computer.

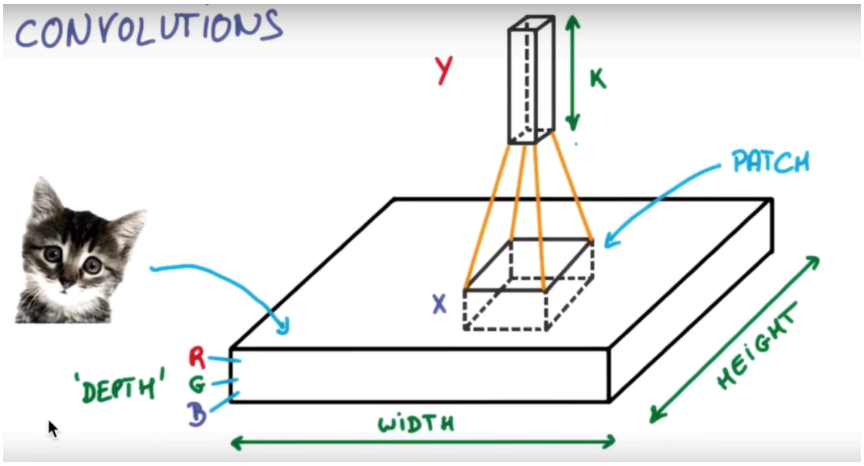
```
~ $ pip install tensorflow
```

Then the first thing we’re going to do is import a data set. We will start with a ‘hello word’ example.

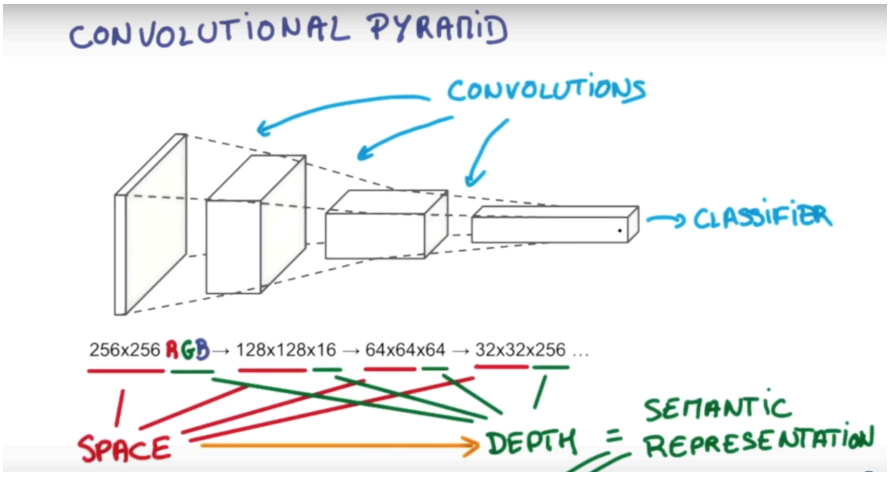
```
1   import tensorflow as tf
2
3   # 28 x 28 images of hand-written digits 0-9.
4   mnist = tf.keras.datasets.mnist
5
6   (x_train, y_train), (x_test, y_test) = mnist.load_data()
7
8   print(x_train[0])
9
10
```

The idea is to feed through the pixel values to the neural network and then have the neural network output which number it actually think the image is.
In this project, we use Keras with the backbend TensorFlow to build our model, and we are focus on the Convolution Neural Network.

1.3 Convolution Neural Network



The input image above is a cat. Images consist of three parameters: length, width and height. The height refers to the information generated by the computer when using color. If it is a black and white image, then the high unit is only 1. If it is a color photo, then it will be constructed by R, B, and G, and the height is 3.



Filters are constantly moving to collect small patches of pixels. The output after collection is a picture with a higher height, a smaller length and width. This image contains some edge information. Then repeating the convolution in the same steps, the length and width of the picture are compressed again, and the

height is increased, which gives a deeper understanding of the input picture. By adding the compressed and increased information to the normal layer, then we can classify the image. Since each time we do convolution, the layer will lose some information, then we use pooling to fix this problem.

Thus, in general, we have the following process.
IMAGE —> CONVONLUTION —> MAX POOLING —> CONVOLUTION —> MAX POOLING —> FULLY CONNECTED —> FULLY CONNECTED —> CLASSIFIER

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

(The image above is what it looks like in python)
Then we compile this model by model.compile(),

```
compile(optimizer, loss=None, metrics=None, loss_weights=None,
        sample_weight_mode=None, weighted_metrics=None, target_tensors=None)
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, batch_size=128, epochs=4, verbose=1, validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Here we use the optimizer Adam. “Adam is an algorithm for first-order gradient-based optimization of stochastic objective function. It is based on adaptive estimates of lower-order moments.” Adam realized both the benefits of AdaGrad and RMSProp. On the other hand, Adam is effective. Since it can achieves good result faster, it is popular used in the field of deep learning.

After running this python file, we get 99.33% test accuracy which is pretty good.

1.4 JSON

In this project, we need to save Model to JSON.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for machines to parse and generate and say for human to read and write. It is built on two structures, a collection of name/value pairs (in other language it is realized as object, record, struct, dictionary, has table, keyed list, or associative array) and an ordered list of values (in other language it is realized as array, vector, list or sequence).

Focus on our project, we want to save our neural network model to JSON. Since Keras provides the ability to describe and model using JSON format with to_json() function. Then we can use it to just save it to file and later we can load it via the mode_from_json() function to create a new model.
Similarly, for the weight, we can also easily use the save_weights() function to save it directly from the model, and later we can load it via load_weights() function.

```
model_json = model.to_json()
with open("model_mnist.json", "w") as json_file:
    json_file.write(model_json)

model.save_weights("model_mnist.h5")
```

1.5 Flask

“Flask is lightweight WSGI web application framework.” Flaks is an excellent micro framework that really makes it enjoyable to work with this back-end web applications.
Here we write an easy “hello world” function.

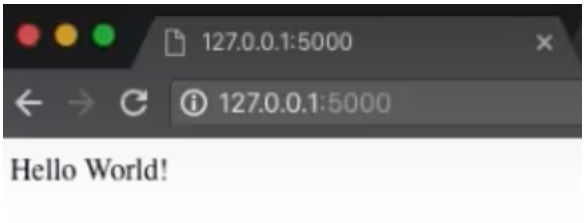
```
from flask import Flask
app= FLask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```

After run it we get.

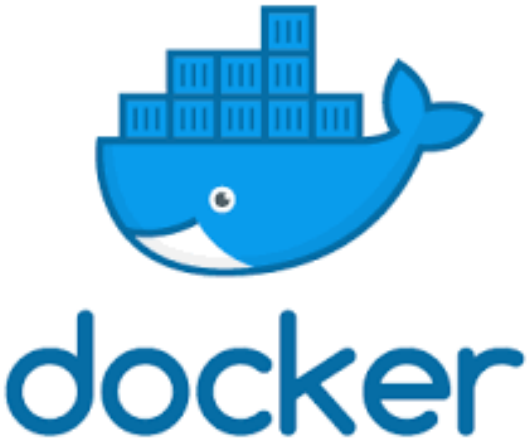
```
$ flask run
 * Serving Flask app "flaskblog.py"
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

We can know it is serving at http://127.0.0.1:5000/
127.0.0.1 should be the ip address of our local machine, and 5000 here is the port number. If we run http://127.0.0.1:5000/ in the web browser, we can get a website with “Hello World!”



1.6 Docker

Docker is mainly a software development platform and a kind of virtualization technology that makes it easy for us to develop and deploy apps inside of neatly packaged virtual containerized environments.
Docker containers can deploy to just about any machine without any compatibility issues so our software stays system agnostic, making software simpler to use , less work to develop, and easy to maintain and deploy.
These containers running on our computer or sever act like litter micro computers with very specific jobs, each with their own operating system and their own isolated CPU processes, Memory, and Network resources. Because of this, they can be easily added, removed, stopped and started again without affecting each other of the host machine.
Docker allows to run many Docker containers where we may only be able to run a few virtual machines. Docker communicates natively with the system kernel, by passing the middleman on Linux machines.



Here we will introduce docker cheat sheet. It includes all commands which we feel useful in this project.

COMMAND	DESCRIPTION
docker ps	List running containers
docker ps -a	List all containers
docker pull	Pull an image
docker images	List images
docker rm	Remove one or more container (only work for stopped containers)
docker —force rm	Force to remove one or more running container
docker start	Start one or more stopped containers
docker stop	Stop one or more running containers
docker —version	Show the docker version information
docker create network [network name]	Create a new network create
docker network rm	Remove the network
docker info	View details about docker installation
docker build —t=[image name]	Create a new docker image
docker run -p [host:port] [container name]	Publish a container’s port to the host
docker run -d -p [host:port] [container name]	Run the container in background with publish this container port to the host
docker run net=[network] -p [host:port] [container name]	Run the container with the network bridge and publish this container port to the host

1.7 Cassandra

Cassandra is a free open source no sequel database. It stores that values in the form of key value pairs. Cassandra in highly robust because it has a masterless replication so basically Cassandra works in the principle of clustering.

Cassandra was created by Facebook and later moved on to apache license. Facebook created Cassandra, so they inherited the features to BigTable and dynamoDB. This is why Cassandra is more popular and Cassandra is very robust in handling huge amount of data.
Here we can easily install Cassandra by using docker command.

```
~ $ sudo docker run --name cass -d cassandra:latest
```

(command sudo means programs to be executed as an administrator account, you can also use this command without sudo)

We can Strat the Cassandra by creating a KEYSPACE.
The following image how we create a new KEYSPACE in the terminal.

```
CREATE  KEYSPACE [IF NOT EXISTS] keyspace_name
   WITH REPLICATION = {
      'class' : 'SimpleStrategy', 'replication_factor' : N }
   }
```

In the project, we use Cassandra-driver, which may have a little different with the above attachment.
First we need to give the input of the variable KEYSPACE.

```
KEYSPACE = "mnist_key"
```

```
def createKeySpace():
    cluster = Cluster(contact_points=['mnist_cassandra'], port=9042)
    # cluster = Cluster(contact_points=['127.0.0.1'], port=9042)
    session = cluster.connect()

    log.info("Creating keyspace...")
    try:
        session.execute("""
            CREATE KEYSPACE %s
            WITH replication = { 'class': 'SimpleStrategy', 'replication_factor': '3' }
            """ % KEYSPACE)

        log.info("Setting keyspace...")
        session.set_keyspace(KEYSPACE)
```

In the project, we name KEYSPACE “mnist_key”. Then similar to the work we did in the terminal, we will assign the details of this KEYSPACE.

Then let’s start to create the table inside this KEYSPACE.
We can also create the table through terminal.

```
CREATE TABLE table_name
( ColumnName1 type PRIMARY KEY,
  ColumnName2 type,
  ColumnName3 type);
```

We must tell Cassandra which variable is PRIMARY KEY, since it specifies which node will hold a particular table row.
In the project we create the table by using Cassandra-driver.

```
log.info("Creating table...")
session.execute("""
    CREATE TABLE record_data(
        date timestamp,
        result text,
        PRIMARY KEY (date)
    )
    """)
```

It is very similar to the way we did in the terminal. Here we let “date” be the PRIMARY KEY.

The last thing we need to do is inserting the data into our table.

First way inserting data via terminal.

```
Insert into keyspace_name.table_name(
    ColumnName1,
    ColumnName2,
    ColumnName3 . . . .)
values (
    Column1Value,
    Column2Value,
    Column3Value . . . .)
```

In our project, we use Cassandra-driver.

```
def insertData(date, result):
    cluster = Cluster(contact_points=['mnist_cassandra1'], port=9042)
    # cluster = Cluster()
    session = cluster.connect(KEYSPACE)
    log.info("Inserting data...")
    try:
        session.execute("""
            INSERT INTO record_data (date, result)
            VALUES(%s, %s);
            """, (date, result))
    except Exception as e:
        log.error("Unable to insert data")
        log.error(e)
```

Then we can use the following command to call cqlsh, and take a look at our Cassandra.

```
$ docker exec -it mnist_cassandra cqlsh
```

Let’s introduce some helpful cql commands.

COMMAND	DESCRIPTION
USE KEYSPACE	Enter the KEYSPACE
select*from table_name	Show the table
DESC KEYSPCE	List all KEYSPACE

2 Summary

I’m so honoured to have the opportunity to participate this project. Unlike the usual projects or assignments I took in the university, this project is harder and more challenging. Most of the background of this project such as docker, Javascript, flask, mnist I have never seen before. I need to do lots of research so that I can understand and even use this new knowledge. We always do group work in the university, it is my first time to solo a whole project. I feel confidence and proud of myself I can finish this project.

Because of this project, I can have a interest in the machine learning. I can take a brief look of how neural network work. Even though, mnist is a patch in the machine learning field, I’m confident to learn more knowledge about machine learning in the following time.

At last, I would like to thank a lot to prof Zhang for teaching us the background and knowledge about this project and even more about big data, and giving me this opportunity to participate in this project and have access to machine learning.