Zheyue Wang
Professor: Fan Zhang
Project: Remote project
9/15/2019

Project Report

Tensorflow is the most popular open-source machine learning framework as it works fast and flexible. One of the classical examples of using TensorFlow is the machine can recognize whether it is the dog or cat by submitting the photo of the cat and dog. In fact, the machine is trained to know which is a cat or dog by using the simple model. Each pixel on the photo is a neuron, every neuron connects to each other becomes the neural network structure, which is similar to the neural network structure of the human. Therefore, after feeding those data to the model and training it, the machine can be able to know which is dog or cat.

Tensorflow combinate with the MNST data set is another classical example of applying the neural network structure. The MNST data set (Mixed National Institute of Standards and Technology database) contains the training set with 60,000 examples and a test set with 10,000 examples. The set consists of four-part, which are: a training image set, a training labels set, a test image set, and a test labels set. Those packages are not just a text file or an image file, those are the compressed binary file. According to the website, the training set has 60,000 use cases, which means that the file contains 60,000 tags, each with a value between 0 and 9. Going back to the training tag set, the offset represents the byte offset, which is how much the binary value of this attribute is offset. Type represents the type of value for this attribute; Value represents what the value of this property is; Description is to describe what it is.

Feeding the MNST data set in the module, training it, the machine can get the rough prediction of the number. However, there was some residual between the model and our prediction. The module we use by given is a linear regression module, which is a simple module that might lead residual from the real result. In order to minimize the residual, we add the cross-entropy to reduce the residual and decrease the gradient. During the process of feed data, we fist have some problems with saving and restore our module. After we add tf.train.Saver(), we find that it creates 3 new folders about the memory point save for the model, which are： checkpoint, model.ckpt, model.ckpt.meta. Checkpoint file saves a list of all model files in a directory; model.ckpt.meta file saves the structure of the TensorFlow calculation graph, kind of the network structure of the neural network, which records the information of nodes in the calculation graph. Thus, we find that model.ckpt.meta is the file we will take to reread from our saving model. What's more, we also find that when we run the softmax.py file, it automatically downloads the MNIST data set. However, since those packages are compressed binary files, we write some code to unpack those files and training them. Another part that we used to have some problems is the session. When we first try to save our model, I just randomly put the session saver() anywhere. Later, we receive the error reminder form the terminal that we should not do that as we do not have a variable. In fact, we later learn that the session saver () should always have the variable at the front. Another problem of the session is when we use with tf.Session().as sess. After several incorrect tries, we found that as long as the code goes outside of with tf.Session().as sess, the session will automatically close. Therefore, we change it with tf.Session().as_default() as sess.

Besides using TensorFlow, this project also requires other people can use our service. In this case, we need to use Flask and RESTful. When the user accesses a Web address in a browser and makes an HTTP request, the Web framework handles the request, assigns different access addresses to the corresponding code, and generates HTML to create an HTTP response with content. Basically Web framework is a software framework that helps the developer write Web applications more easily and efficiently. Flask is a microframework because it only implements the core functions of a Web application: Flask consists of two main dependencies (Werkzeug, which provides routing, debugging, and Web server gateway interfaces, and Jinja2, which provides templates). RESTful(Resource Representational State Transfer) is popular recently as it provides services for Web, iOS and Android through a uniform set of interfaces. What's more, RESTful has GET(to get resources), POST(is used to create a new resource), PUT (is used to update resources), DELETE (is used to delete resources).

During the process of using flask, we got some troubles with a related running environment. Fist of all, I have problems with calling my function in another flask file. I struggle a lot as the system always shows that I did not import the flask model. Later I realize that the reason I will get this error as I did not install the flask within the TensorFlow, instead, I download it at the base. What's more, I find that I could not run my softmax.py file as I did not put the related database in Flask.

In conclusion, the prediction which I put in flask has low accuracy, and my assumption is that the linear regression model can only use for the simplest calculation. As it deals with complex training, the accuracy decrease. Besides, the file we use for training the dataset is softmax instead of the deep file. The deep.py file contains the convolute neutron training which can deal with more complex calculations and raise the accusation.