TensorFlow

www.huawei.com

HUAWEI

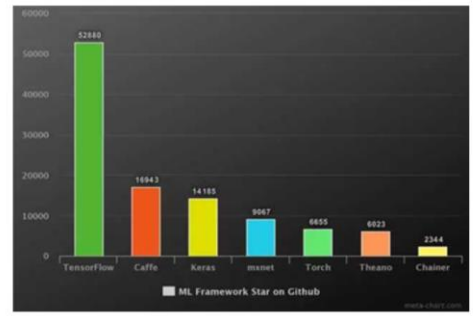
Objectives

- After completing this course, you will be able to:
  - Understand what TensorFlow is and its characteristics.
  - Master the basic knowledge of TensorFlow and the methods for environment setup.
  - Understand TensorFlow modules.
  - Master basic development steps using TensorFlow.
  - Understand other deep learning frameworks.

Contents

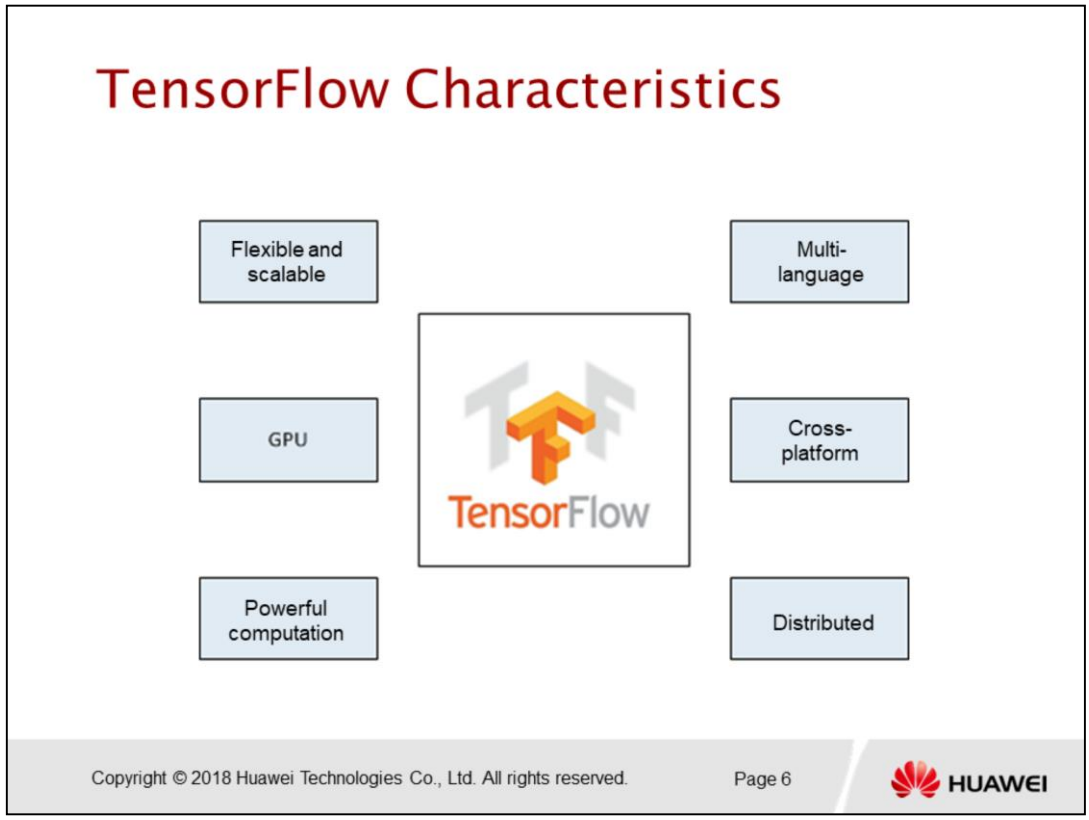
What Is TensorFlow

- TensorFlow is Google's second-generation software library for dataflow programming.
- The TensorFlow framework supports various deep learning algorithms, as well as many computing platforms other than those for deep learning. The system stability is high.
- TensorFlow is open-source, which facilitates maintenance and update and improves the development efficiency.



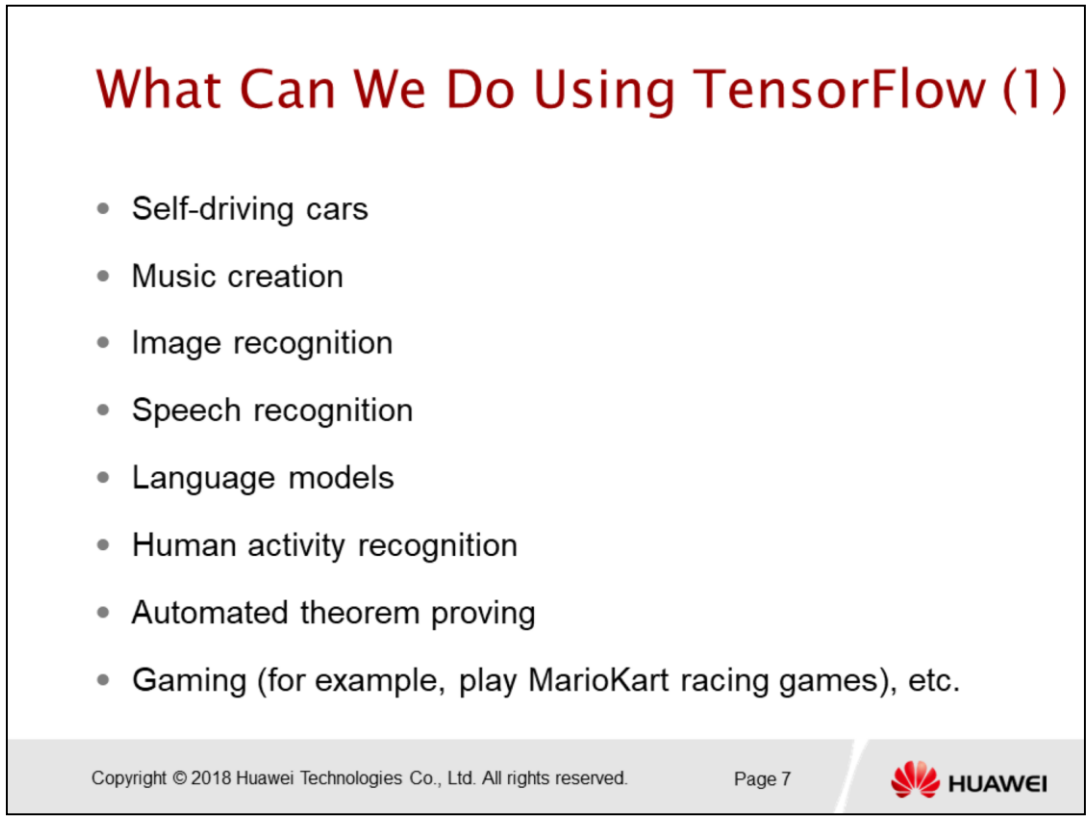
- TensorFlow is an open-source software library for machine learning across various perception and language understanding tasks. Currently, TensorFlow has been used by 50 teams to research on and produce a wide range of Google's commercial products, such as voice recognition, Gmail, Google Album, and Google Search. Many of these products once adopted DistBelief, the predecessor of TensorFlow. For example, AlphaGo was developed using TensorFlow.
- Starting in 2011, Google Brain built DistBelief as the first deep learning framework to adapt to artificial neural network algorithms.
  - Established neural network models, such as convolutional neural network (CNN), multilayer perception (MLP) neural network, long short-term memory (LSTM) neural network, etc.
  - Training: In the training process, the weight of each node is calculated. A large amount of floating-point arithmetic is required, and users are unaware of this process.
  - Inference: In the inference process, computation is small in amount and can be converted to fixed-point arithmetic. Users are aware of this process, requiring shorter delays.

Contents



- Flexible and scalable: TensorFlow can run on different computers, ranging from smartphones to computer clusters. You can use TensorFlow to quickly generate your training model on various devices.
- Multi-language: C++ and Python are supported.
- GPU: For parallel processing mode on a large-scale computer cluster, the performance of TensorFlow is slightly lower than that of the CNTK. However, when using personal machines, TensorFlow allows automatic selection of CPUs or GPUs according to the machine configuration, which is more user-friendly and intelligent.
- Cross-platform: TensorFlow is cross-platform. Models generated using TensorFlow can meet diverse requirements of more users. TensorFlow is now available for Mac, Linux, and Windows operating systems. The concept of "right out of the box" is supported.
- Powerful computation capability: A computational graph in TensorFlow is a directed acyclic graph (DAG), which optimizes the computation process.
- Distributed:
  - Currently, only a few deep learning frameworks are natively distributed. They are TensorFlow, CNTK, DeepLearning4J, and MXNet. In the case of a single GPU, most deep learning frameworks depend on the cuDNN. Therefore, as long as the hardware computing capability or memory allocation is not strikingly different, the training speed will not vary greatly.
  - However, for large-scale deep learning, a huge amount of data makes it difficult for a single machine to complete training within a limited time. TensorFlow supports distributed training.
  - The TensorFlow design does not excel in optimizing communications between

different devices. For a single device, CPU is used for reduction. For distributed communications among multiple devices, socket-based RPC, instead of the faster RDMA, is used. Therefore, the distributed performance may not be the best.



- Ryan Zotti/Self-Driving-Car Ryan Zotti: Builds a mini-version self-driving car based on Raspberry Pi, OpenCV, and TensorFlow.
- tensorflow/magenta: A project of the Google Brain team that uses TensorFlow to automatically generate music.
- tensorflow/tensorflow: The most widely known example is handwritten digit recognition based on MNIST dataset.
- pannous/tensorflow-speech-recognition: Speech recognition using the TensorFlow deep learning framework.
- https://github.com/hunkim/word-rnn-tensorflow Multi-layer Recurrent Neural Networks (LSTM, RNN) for word-level language models in Python using TensorFlow.
- guillaume-chevalier/LSTM-Human-Activity-Recognition: Recognizes human body movements based on LSTM RNN.
- tensorflow/deepmatht: The Deepmath project seeks to improve automated theorem proving using deep learning and other machine learning techniques.
- kevinhughes27/TensorKart: Achieves self-driving MarioKart with TensorFlow. This is mind-blowing.

What Can We Do Using TensorFlow (2)

Artistic style transfer

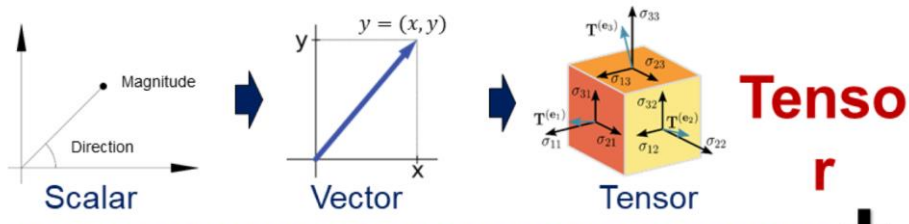
Facial recognition

HUAWEI

- Artistic style transfer: The Neural-Style can generate various interesting pictures.
- Facial recognition: A Microsoft app detects people's ages in photos.

Contents

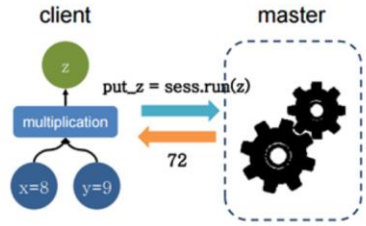
TensorFlow



Tenso

r

+



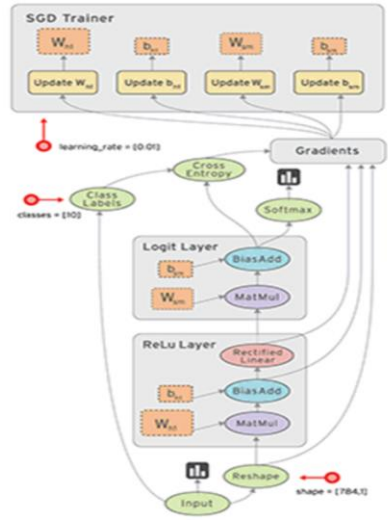
Flow

Data flow graph

HUAWEI

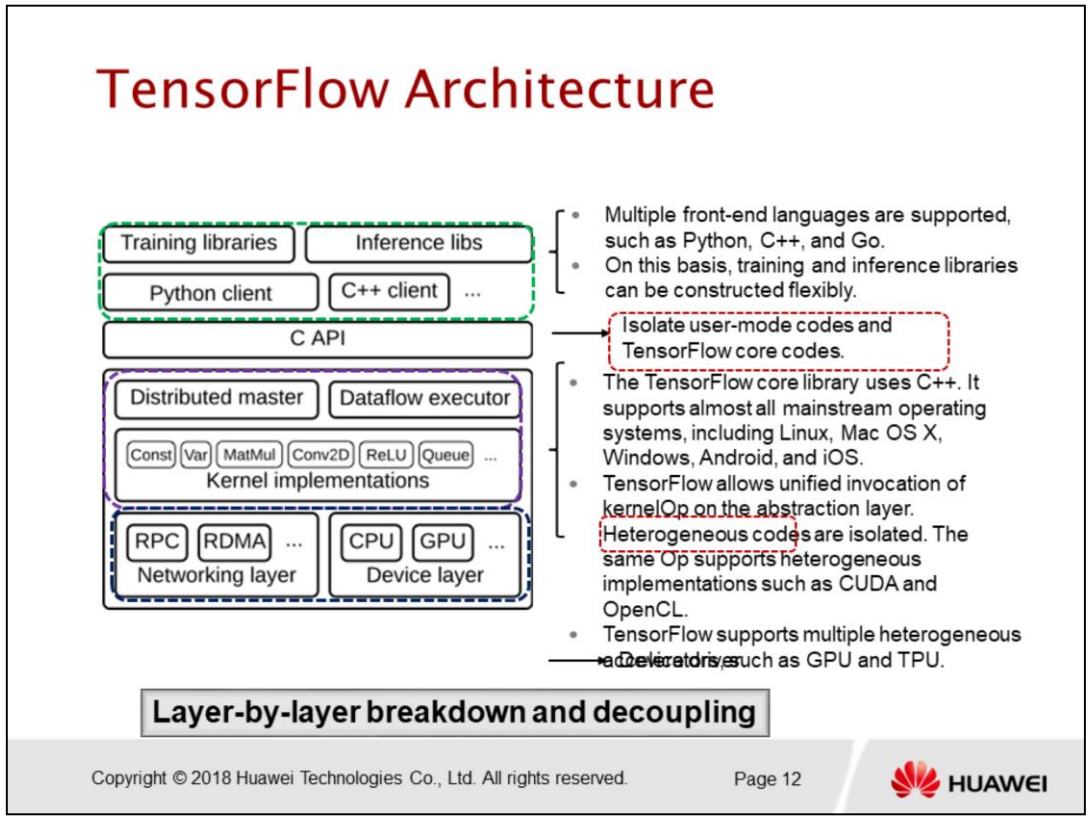
TensorFlow Computation Process

- TensorFlow is a programming system that describes computation in the form of a graph. It is a framework that implements and executes machine learning/deep learning algorithms in the form of tensors on graphs.



HUAWEI

- Building the computation graph: It can be understood as a front-end work, defining a set of operations organized into a directed acyclic graph (DAG) according to the data flow. The operations include but are not limited to: initialization, multiplication, reading, assignment, network feed-forward computation, loss computation, and gradient backpropagation.
- Execution: It can be considered as a back-end task. On a specified GPU or CPU, run the session.run () command to execute computation graphs (C++). Generally, an execution is performed repeatedly.
- Data flow graphs describe mathematical computation with a directed acyclic graph (DAG) of nodes and edges. Nodes typically implement mathematical operations, but can also represent endpoints to feed in data, push out results, or read/write persistent variables. Edges describe the input/output relationships between nodes. These data edges carry the tensors that flow between the nodes as multidimensional data arrays. The flow of tensors through the graph is where TensorFlow gets its name. Nodes are assigned to computational devices and execute asynchronously and in parallel once all the tensors on their incoming edges become available.



- The kernel engine of TensorFlow is implemented by using C++. The efficiency is high. The kernel encapsulation of TensorFlow implements basic machine learning algorithms such as NN and CNN.
- The front-end application layer supports Python or C++. Python (one of the most widely used programming languages in the AI field) is used more commonly.
- At the application layer, if other Python-based time-consuming operations (such as data preprocessing) are being performed, the efficiency may be low. However, this is not caused by TensorFlow.

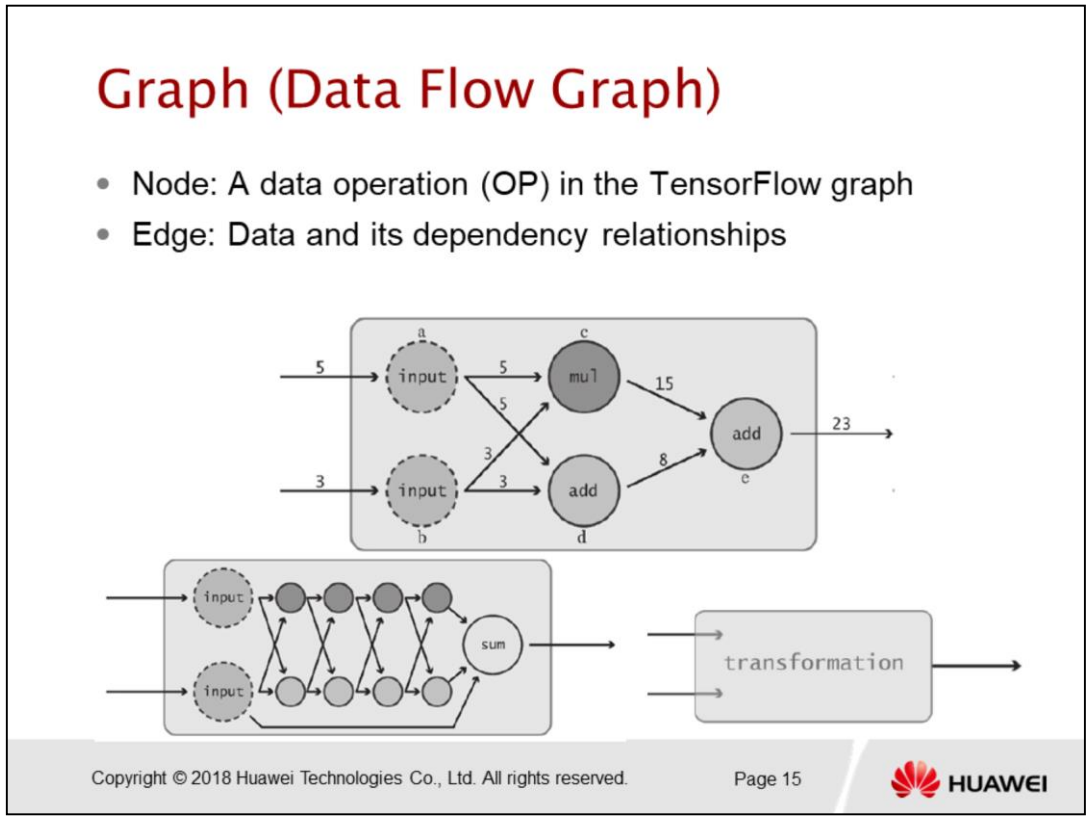
Basic Concepts of TensorFlow

- Tensor
- Graph
  - Node
  - Edge
- Operator
- Session
  - Feed
  - Fetch
- Variable

Tensor

- Tensor is the primary data structure in TensorFlow programs. Tensors are N-dimensional (where N could be 1, 2, 3, 4, or very large) data structures. In a running graph, tensors are the data that flows between nodes.
- The data structure contained in a tensor is: name + shape + type.

- Usage of tensors:
  - Use intermediate computation results to enhance code readability.
  - Obtain computation results through sessions.



- In TensorFlow implementation, a machine learning algorithm is presented as a graph. A node in the graph is an operation. The outputs of a node ranges from 0 to multiple. The next slide lists some operators implemented by TensorFlow.
- A client-side program interacts with the TensorFlow system by creating a session.
- To create a graph, the session interface supports extensions to complement the current graph.
- The Run interface allows users to specify a tensor as the input argument to feed into the computation diagram to obtain a set of outputs.
- TensorFlow uses graphs to represent computation tasks. A node in a graph is called an operation (Op in short). One Op obtains 0 or more tensors, executes computation, and generates 0 or more tensors.
- A tensor is a type of multidimensional array. For example, you can represent a group of image sets as a four-dimensional floating-point array. The four dimensions are batch, height, width, and channel.

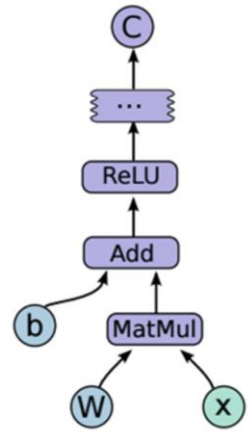
TensorFlow Operators

Category	Example
Element – wise mathematical Operations	Add, Sub, Div, Exp, Log, Greater, Less, Equal…
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle…
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant…
Stateful operations	Variable, Assign, AssignAdd…
Neural-net building blocks	SoftMax, Singmoid, ReLU, Convolution2D, MAxPool…
Checkpointing operations	Save, Restore…
Queue and synchronization operations	Enqueue, Dequeue, Mutex Acquire, Mutex Release…
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

- TensorFlow supports multiple common operators, such as matrix multiplication and addition.
- A kernel function is a specific implementation of operators and can run on particular types of devices (such as CPU or GPU).
- TensorFlow uses the registration mechanism to determine supported operator sets and kernel functions for further extension.

Session

- A session owns and manages all resources when the TensorFlow program is running.
- A client uses a session to interact with the TensorFlow system. Generally, an empty graph is generated when a session is created. Nodes and edges are added to the session to form a graph for execution.



- Adopt the Python context manager to use sessions to solve the problem of resource release upon an exception.
- Instead of using a variable to hold a session, you can replace the session class with InteractiveSession.
- Session is the statement that Tensorflow executes to control and output files. Run session.run () to obtain the operation results you want to know or the parts you want to calculate.

Feed

- The feed mechanism directly connects the tensor to a node in the graph, and temporarily replaces the tensor value on the node. A feed is not created when the graph is formed. Rather, it is applied for when graph execution is triggered. That is, the feed data is initialized in the form of parameters when the "run" or "eval" command is executed. After the execution is complete, the replaced feed data disappears, but the behavior of the node defined in the graph does not change. Generally, the feed mechanism is used together with tf.placeholder ().

Fetch

- The fetch mechanism retrieves the result of an operation in the graph. The fetch application occurs when graph execution is triggered, not when the graph is generated. To fetch the tensor value of one or more nodes, you can call the run () method on the session object and use the list of the nodes to be fetched as parameters to execute the graph.

Variable

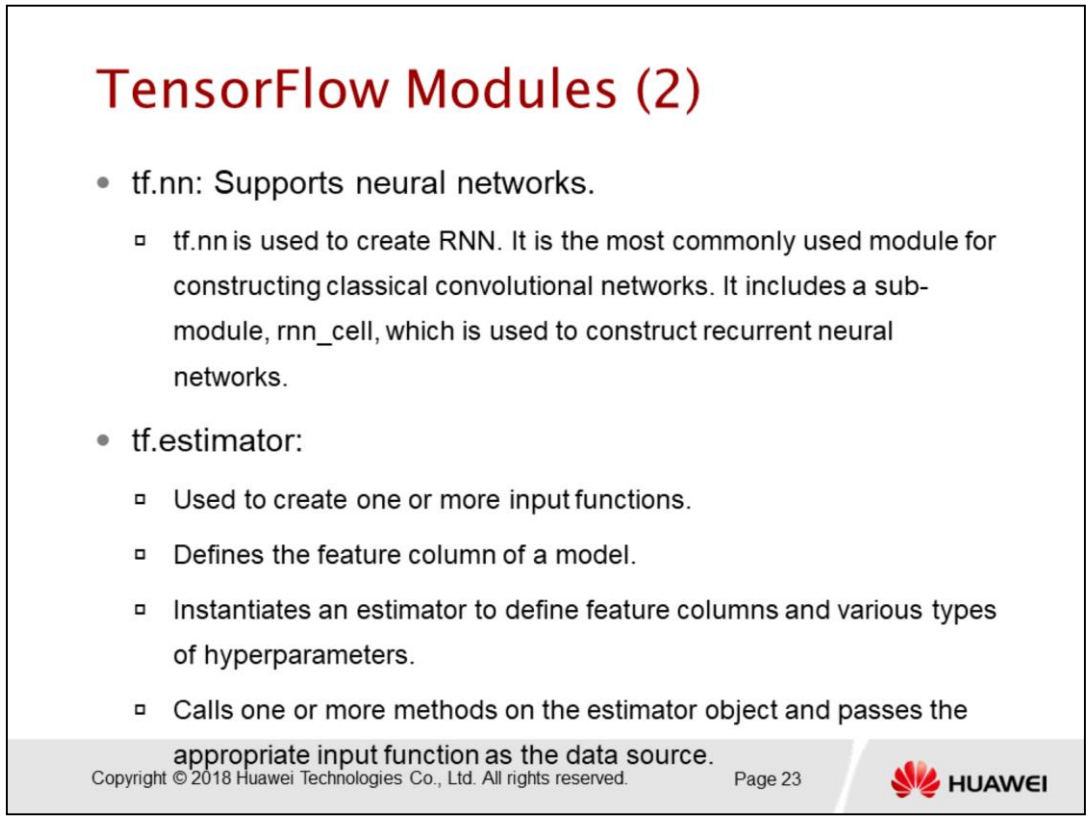
- A variable maintains state across multiple calls to run during graph execution. For example, in a neural network, it is used to identify coefficients such as w and b. The variable in TensorFlow is actually a variable object in Python.
- The initial value of a variable in TensorFlow can be set to a random number, a constant, or a number calculated based on the initial values of other variables.

- A placeholder is a variable that you can assign data to at a later date. It is a place in memory where you will store value later on. To transfer data from an external system to TensorFlow, you need to use tf.placeholder ().

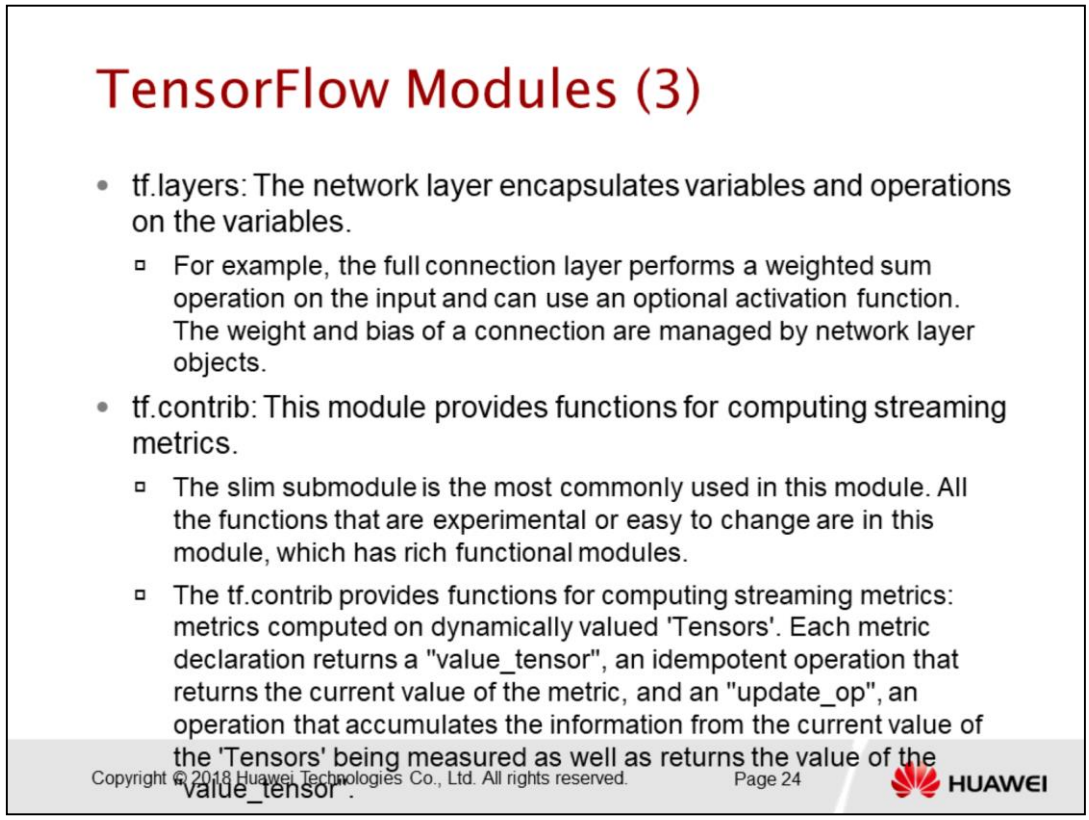
Contents

TensorFlow Modules (1)

- tf.app: A generic entry point script
- tf.nn: Supports neural networks.
- tf.bitwise: Used to manipulate the binary representations of integers.
- tf.compat: A function compatible with Python 2 and Python 3
- tf.contrib: Relatively complex; related to Monte Carlo integration, conditional random field, the builder, regularization, initialization, optimization, and feature columns.
- tf.estimator: An advanced tool for processing models
- tf.errors: Types of TensorFlow errors

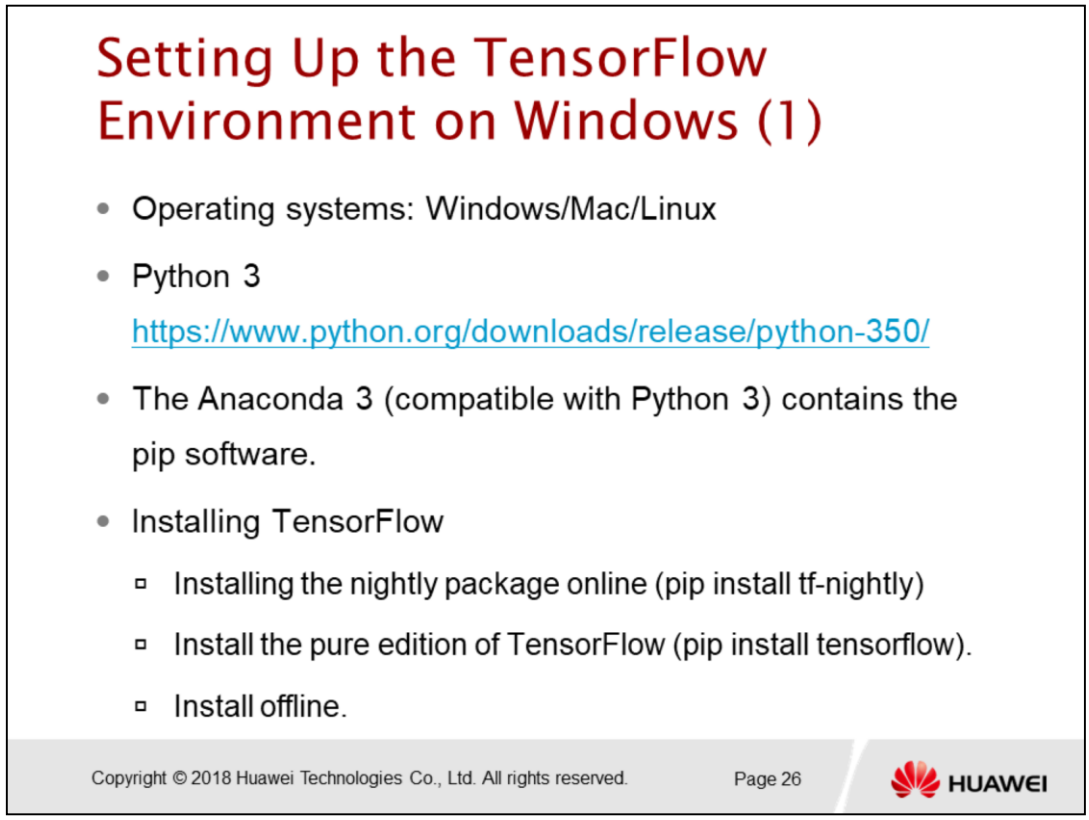


- Estimator: Used for training, evaluation, and prediction, and exported as a service.
- tf.estimator:
  - Export module: A practical method for exporting an estimator.
  - input module: A practical method for creating a simple input_fns.
- Advantages of the estimator:
  - Estimator-based models can run on a single server, or run on multiple distributed servers without the need for modification. Moreover, you can run estimator-based models on CPUs, GPUs, or TPUs.
  - Estimators simplify the implementation of internal sharing among model developers.
  - You can compile art models of certain state using advanced and visual codes. In short, using an estimator is usually simpler than using a low-level API.
  - Estimators are built on tf.layers, which simplifies customization.
  - Estimators build the graph for you. In other words, you don't have to build the graph.
  - Estimators provide a secure distributed training cycle.

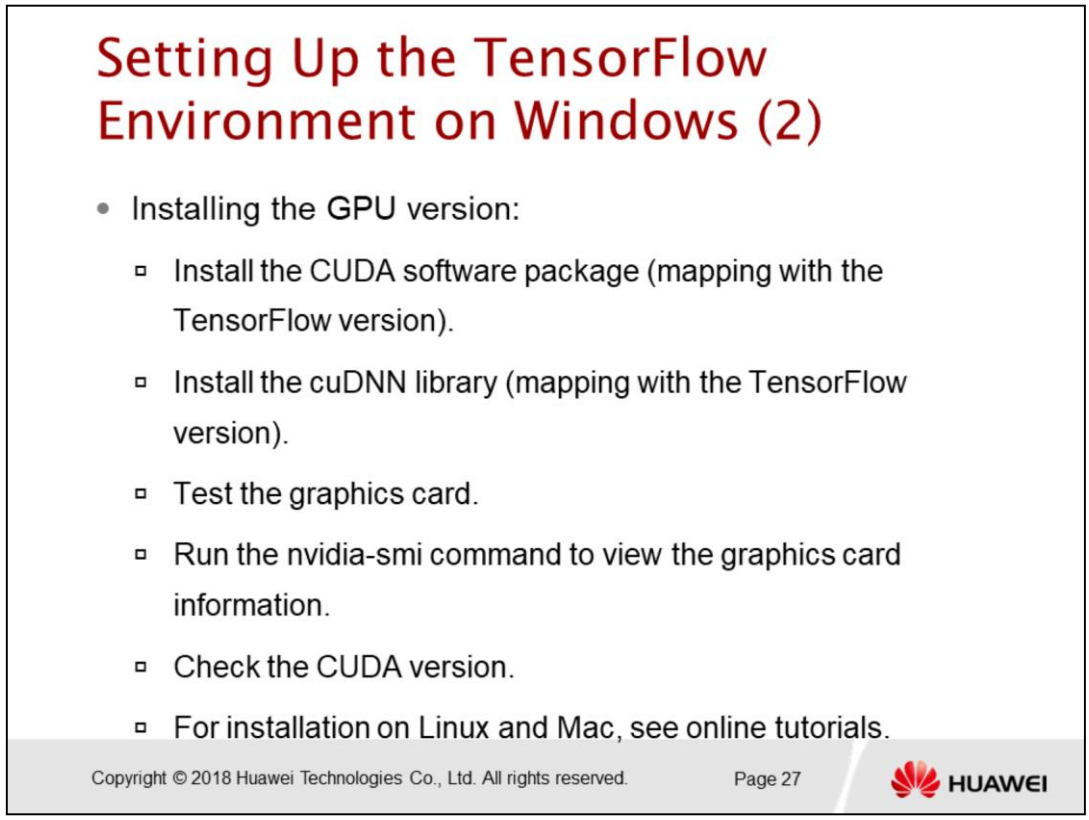


- Estimator: Used for training, evaluation, and prediction, and exported as a service.
  - Used to create one or more input functions.
  - Defines the feature column of a model.
  - Instantiates an estimator to define feature columns and various types of hyperparameters.
  - Calls one or more methods on the estimator object and passes the appropriate input function as the data source.

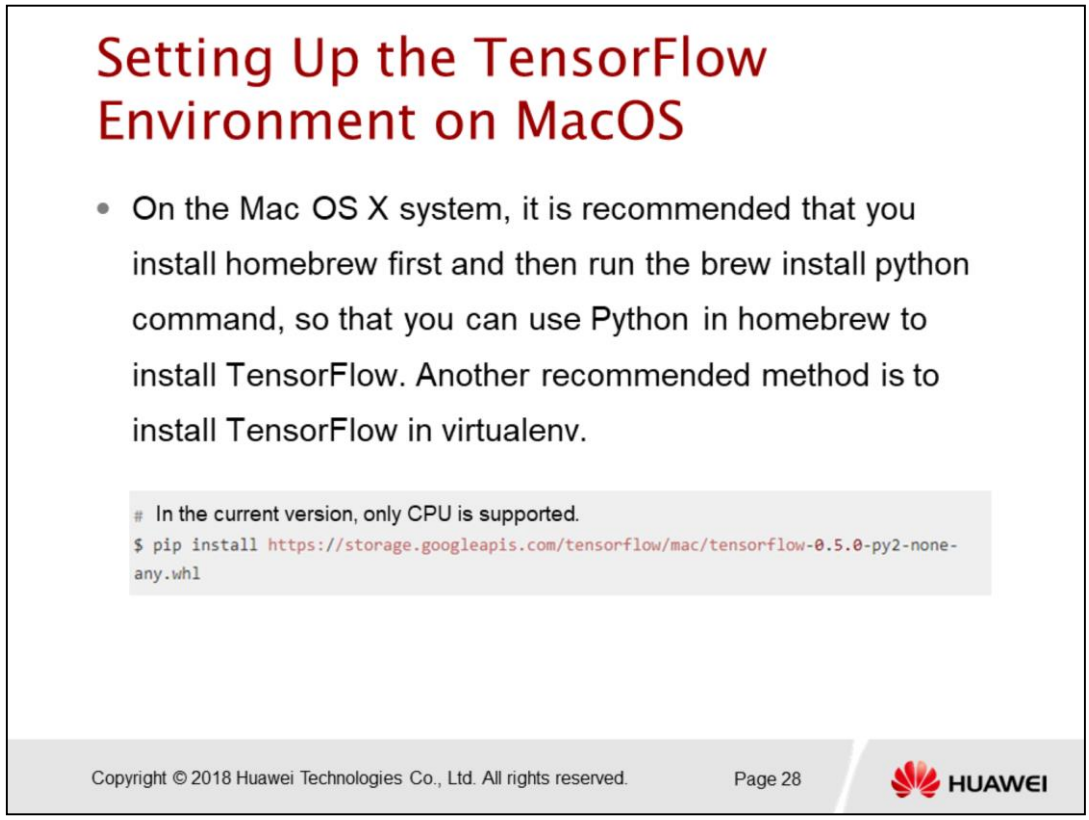
Contents



- Run the following command to install the GPU version: pip install tensorflow-gpu.
- If TensorFlow has been installed, upgrade it to the latest version. Uninstall the original version and install the latest version.
- By default, the nightly package installs the libraries that need to be depended on.



- TensorFlow 1.0-1.5 supports only CUDA 8.0.
- TensorFlow 1.0-1.2 uses cuDNN 5.1.
- TensorFlow 1.3 and later versions use cuDNN 6.0.
- To test the graphics card, you need to install the NVIDIA to the default path and add the path to the Path environment variable.
- Viewing the CUDA version (nvcc-V).
- Installation can also be performed on Ubauntu/Linux and Max OS X by running the pip command.
- Docker can also be used to run TensorFlow. Install Docker. You can run the following command to start the container:
  - $ docker run -it b.gcr.io/tensorflow/tensorflow
- The default Docker image contains only a minimum set of libraries required to start and run TensorFlow. We additionally provide the following container, which can also be installed by running the preceding docker run command.
- b.gcr.io/tensorflow/tensorflow-full: For image installation, TensorFlow is completely installed from the source code, including all the tools for compiling and running
- TensorFlow. On this image, you can directly use the source code for testing without installing any of the above depended-upon parts.

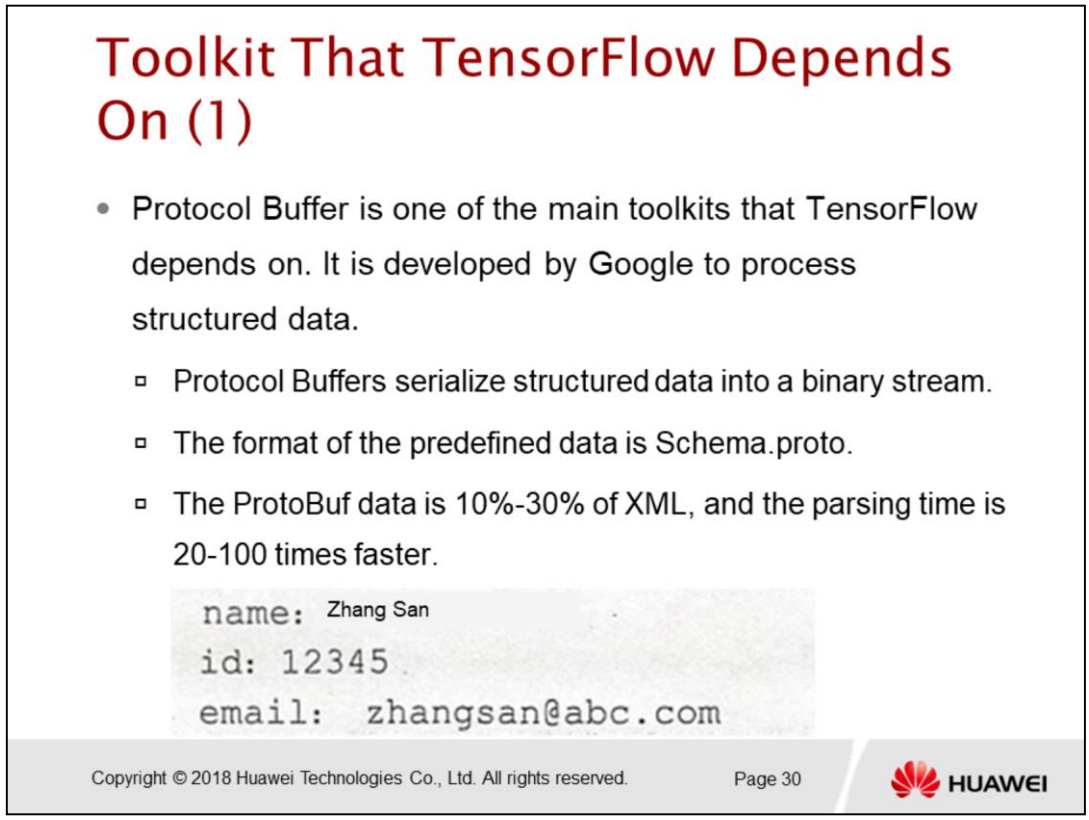


- The protobuf used by TensorFlow depends on the six-1.10.0. However, the six-1.4.1 has been installed in the default Python environment of Apple. This version may be difficult to upgrade. The following methods can be used to solve this problem:
  - Upgrade the "six" of the entire system.
  - Use homebrew to install an isolated Python copy.
  - Compile or use TensorFlow in virtualenv.

Setting Up the TensorFlow Environment on Linux

- The simplest installation mode on Linux is to use pip.

```
# Only the CPU version is used.
$ pip install https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-0.5.0-cp27-none-linux_x86_64.whl

# Start the version supporting GPU. (The version can only be installed after installing the CUDA sdk.
$ pip install https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow-0.5.0-cp27-none-linux_x86_64.whl
```



- It is mainly used to process data of multiple attributes. Serialization is to transform structured data into a data stream (string). Protocol Buffers are used to serialize structured data or restore serialized data streams to the original structured data. It is generally known as processing structured data.

Toolkit That TensorFlow Depends On (1)

- Bazel (automatic build tool)
  - Compared with the traditional Makefile and Ant, Bazel is faster, more scalable, and flexible.
  - The basic concept is workspace. It can be considered a folder that contains the source code required for software compilation and the soft-make address of the output.
  - Only the py_binary, py_library, and py_test compilation modes are supported.

```
-rw-rw-r--  root root 208    BUILD
-rw-rw-r--  root root 48     hello_lib.py
-rw-rw-r--  root root 47     hello_main.py
-rw-rw-r--  root root 0      WORKSPACE
```

Running TensorFlow

- Open the terminal in Python and enter the following command:

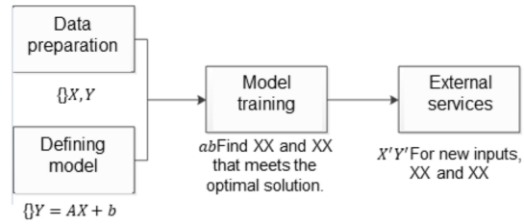
```
$ python

>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> print sess.run(hello)
Hello, TensorFlow!
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> print sess.run(a+b)
42
>>>
```

Contents

HUAWEI

Development Using TensorFlow

- Define the input node.
- Define "learning parameter" variables.
- Define the operation.
- Optimize functions and objectives.
- Initialize all variables.
- Perform model training to iterate and update parameters to the optimal solution.
- Test the model.
- Use the model.



- First, define the model, that is, the graph.
- Initialize variables.
- Create a session.
- Run the graph in the session.
- Close the session.

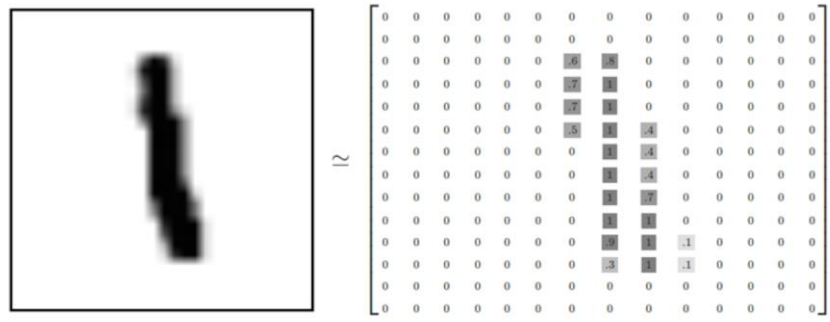
Preparing Data (1)

- MNIST is a classic problem in machine learning. The problem is solved by identifying 28 x 28 pixel grayscale images of handwritten digits as corresponding numbers ranging from 0 to 9.
- The MNIST dataset is an example of TensorFlow, you don't need to download it.

- MNIST is an entry-level computer vision dataset that contains a variety of handwritten digital images. It also contains a tag for each image, telling us that this is a few digits. For example, the tags for the above four images are 5, 0, 4, and 1, respectively.
- The official website of the MNIST dataset is Yann LeCun's website. From this website, you can download the Python source code for automatically downloading and installing the MNIST dataset. You can download this code and then import it into your project with the given code, or you can copy and paste it directly into your code file.
- The downloaded dataset is divided into two parts: 60,000 rows of training data (mnist.train) and 10,000 rows of test data (mnist.test). Such segmentation is important. A separate set of test data must be used in the machine learning model design, not for training but to evaluate the performance of the model, making it easier to generalize the design model to other datasets (generalization).

Preparing Data (2)

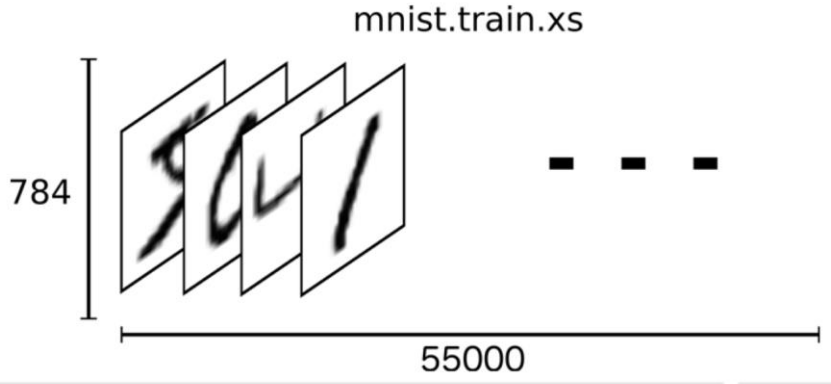
- An image is a matrix of 28 pixels x 28 pixels, which can be represented by a two-dimensional integer matrix of the same size.



- Each MNIST data unit consists of two parts: an image containing handwritten digits and a corresponding lab. We set these images to "xs"and set these tags to "ys". Both the training dataset and the test dataset contain xs and ys.
- Here we can simply use a one-dimensional array containing 28 pixels x 28 pixels for a total of 784 pixels to represent each image, because later we only use softmax regression to identify and classify images. (Flattening a two-dimensional structure of pixels into one-dimensional array results in the loss of the picture's two-dimensional nature. This is obviously not ideal, and the best computer vision methods will mine and use such structural information.)

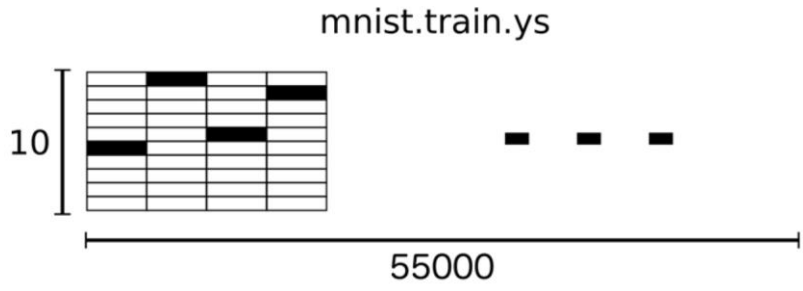
MNIST Dataset (1)

- The MNIST training dataset may be a 55000 x 784 tensor, that is, a multidimensional array. The first dimension represents the index of an image, and the second dimension represents the index of any pixel in the image (the pixel value in the tensor is between 0 and 1).



MNIST Dataset (2)

- mnist.train.labels is a two-dimensional array of 55000 x 10, as is shown in the following figure:



HUAWEI

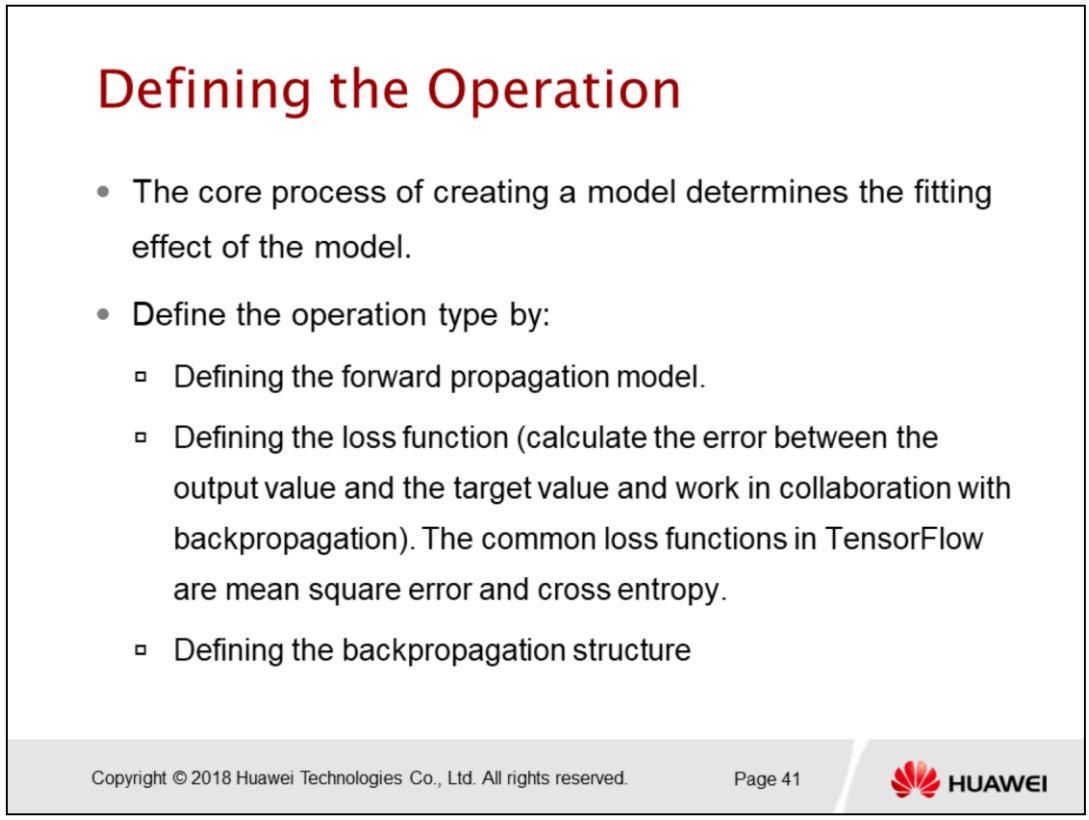
- The tag value of a handwritten digit image in MNIST ranges from 1 to 9, which describes the actual number represented in a given image. For this tutorial, we make the tag data "one-hot vectors." A one-hot vector is 0 except for one digit. For convenience, the index position corresponding to a vector is set to 1, and other location elements are set to 0. For example, [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] indicates 3.
- The MNIST database contains three datasets: training dataset, test dataset, and validation dataset.

Defining the Input Node

- TensorFlow has the following methods for defining input nodes:
  - Defined by placeholders (commonly used)
  - Defined by dictionary types (used when there are many inputs)
  - Directly defined (seldom used)
- The input images are 550000 x 784 matrices. Therefore, create a placeholder x of [None, 784], and a placeholder y of [None and 10], and use the feed mechanism to input images and tags.

Defining Learning Parameters

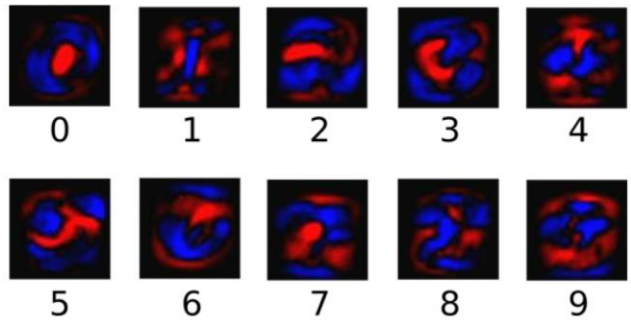
- Define "learning parameter" variables.
  - Directly defined
  - Defined by dictionary
- The learning parameters include weight values and bias values. In TensorFlow, learning parameters are defined by variables.
- A variable represents a modifiable tensor, which is defined in the TensorFlow graph. Learning parameters defined by variables can be used for computing input values, or be modified in computation.



- To find the minimum value in backpropagation, the function must be traceable.
- The selection of the loss function depends on the type of the tag data. If the input is a real number and an unbounded value, the mean square error function is used. If the input is a bit vector, use the cross entropy function.
- Defining the backpropagation structure:
  - Compile the training model to obtain proper parameters.
  - Each time, change the values of the learning parameters, and the loss value decreases, so that the output result is closer to the tag data.

Creating a Model

- The softmax model can be used to assign probabilities to different objects. Even after we train more elaborate models, the final step also needs to use softmax to assign probabilities.
- The image below shows the weight of each pixel in a picture that the model learns for a particular number class. Red represents negative weights and blue represents positive weights.



- Softmax regression is equivalent to logic regression in situations involving multiple classes.
- Softmax regression consists of two steps:
  - First, in order to get evidence that a given picture belongs to a specific numeric class, we perform a weighted sum on the picture pixel values. If this pixel has strong evidence that the picture does not belong to this class, the corresponding weight is negative, and conversely, if the pixel has favorable evidence to support this picture belongs to this class, then the weight is a positive number.
- The training process of the softmax regression model has several steps. To start with (in step 0), the model parameters are initialized. Perform repeated iterations for specified numbers of training (see below) or until the parameters converge.
  - Step 0: Initialize the weight matrix and the bias value using zero (or a small random value).
  - Step 1: For each class, k, calculate the linear combination of the input feature and the weight vector. That is, for each training sample, calculate the score of each class.
  - Step 2: Apply the softmax activation function to convert the score to a probability.
  - Step 3: Calculate the loss of the entire training set. Hopefully the model can predict the high probability of the target class and the low probability of other

classes.

- Step 4: Calculate the gradient of the loss function relative to each weight vector and bias.
- Step 5: Update the weight and bias of each class, k.

Defining a Model

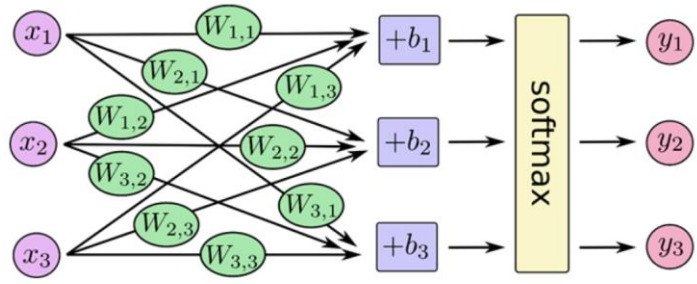
- We also need to add an extra bias because the input tends to have some extraneous interference. So for a given input image x, the evidence proving that it represents the digit i can be expressed as: $evidence_i = \sum_j W_{i,j} x_j + b_i$
- Wherein W_i indicates weight. b_i indicates the bias of the numeric class i. j indicates the pixel index of a given image, x, and is used for summing the pixels. Then, use the softmax function to convert the evidence to the probability y.

$$y = softmax(evidence)$$

- Here the softmax can be seen as an activation function or a link function that converts the output of the linear function we define to the format we want, that is, the probability distribution for the 10 numeric classes. Therefore, given a picture, its fit for each digit can be converted into a probability value by the softmax function.

Regression Model (1)

- Add a bias to the weighted sum of the inputs x_i, and then input the biases to the softmax function.



- The input values are evaluated as exponents and the result values are then regularized. This exponentiation implies that the larger evidence corresponds to the multiplier weight value in the larger hypothesis model (hypothesis). Conversely, having less evidence means having smaller multiplier coefficients in the hypothetical model. Assume that the weights in the model cannot be zero or negative. Softmax then normalizes these weights so that their sum is equal to one to construct an effective probability distribution.
- The softmax regression model can be explained by the diagram. For the xs weighted summation of the inputs, add a bias and add them to the softmax function.

Regression Model (2)

- In actual cases, the calculation process is expressed using vectors, as shown in the following figure:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

Further, it can be written in a more compact way:
$y = softmax(W_x + b)$.

- We can also express this calculation process using vectors: multiply by the matrix and add vector. This helps to increase the computational efficiency.

Optimizing Functions and Objectives

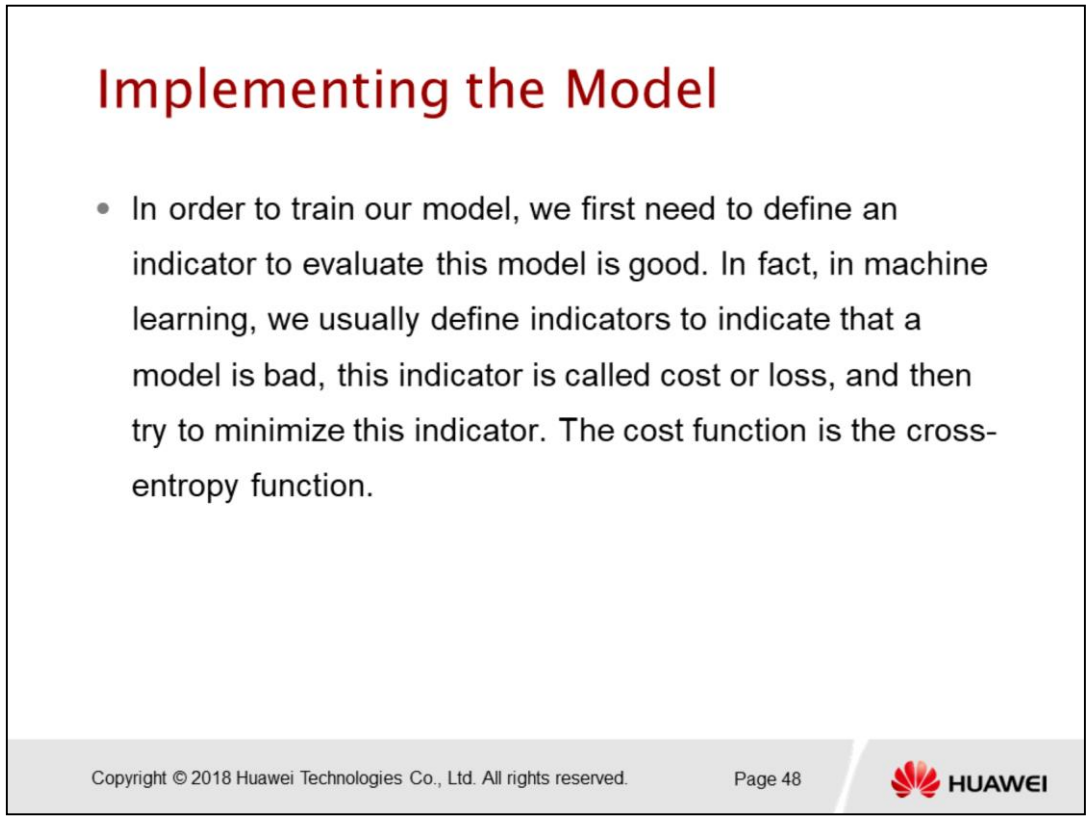
- This process is completed in backpropagation. The backpropagation process is to transfer the error back along the reverse direction of forward propagation, involving L1 and L2 regularization, impulse adjustment, learning rate adaptation, and the Adam random gradient descent algorithm.

Initializing

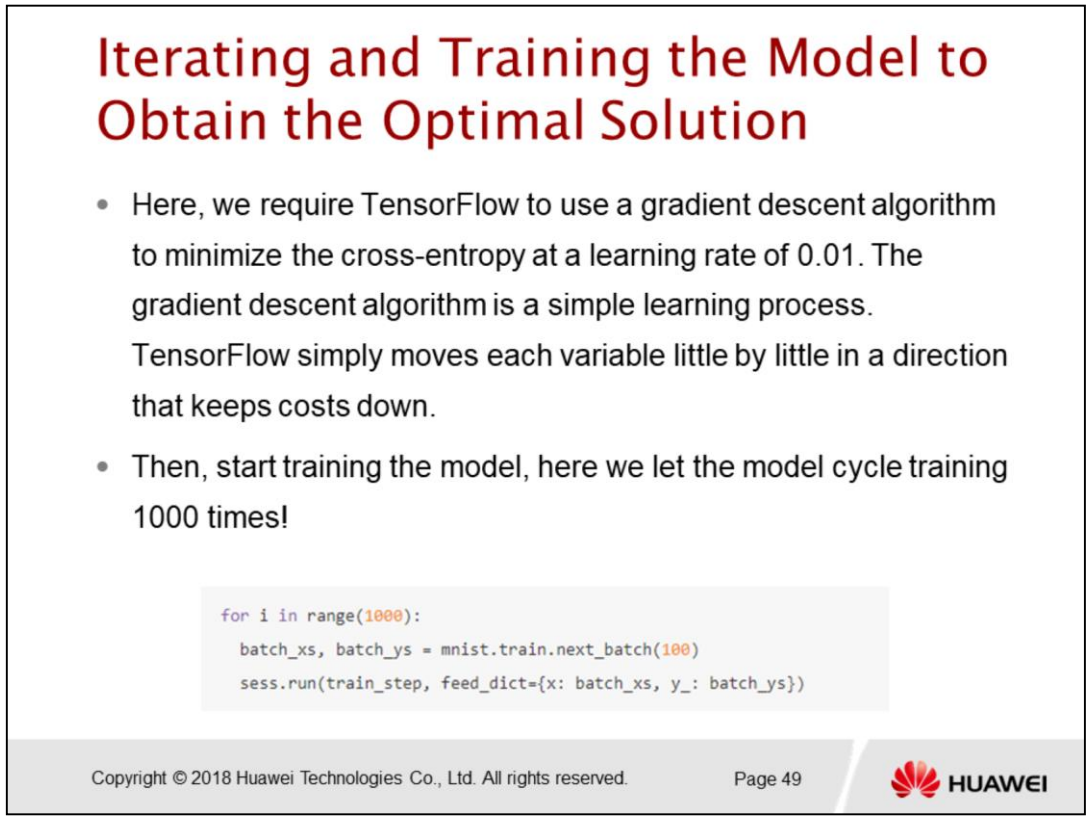
- To create a model, you need to create a huge number of weights and biases. The weights in the model should be initialized with a small amount of noise to break the symmetry and avoid the zero gradient.
- To avoid repeated initialization during model creation, we define two functions for initialization.

```
def weight_variable(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)

def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)
```


- Given that ReLU neurons are used, a relatively small positive number is used to initialize the bias item, so as to avoid the problem that the outputs of neuron nodes are constantly zero (dead neurons).



- Cross entropy results from the information compression coding technology in information theory. It is used to measure the inefficiency of our predictions used to describe the truth. It measures the low efficiency of our prediction for describing the truth. Cross entropy is not only used to measure a single pair of predictions and true values, but is to measure the sum of the cross entropies of all 100 images. The prediction performance for 100 data points is better than that of a single data point.
- TensorFlow has a graph that describes each of your computational units, it can automatically use the backpropagation algorithm to effectively determine how your variables affect the cost value you want to minimize. Then, TensorFlow will use your optimization algorithm to constantly modify variables to reduce costs.
- The above cross entropy is not limited to a single image, but applies to the entire available dataset.
- Next, with the objective of minimizing the cost function, we train the model to obtain corresponding parameter values (that is, the weight and bias). TensorFlow knows your computation process. It automatically uses the backpropagation algorithm to obtain the corresponding parameter changes and their influence on minimizing the cost function. You can then choose an optimization algorithm to determine how to minimize the cost function.



- What TensorFlow actually does here is that it adds a series of new calculations to the back of the graph that describes your calculations for backpropagation and gradient descent. Then, it returns to you just a single operation. When running this operation, it uses the gradient descent algorithm to train your model, fine-tune your variables, and continuously reduce costs.
- Now, we have set up our model. Before running the calculation, we need to add an operation to initialize the variable we created:
- Now we can Session start our model in one and initialize the variables:
- Using a small amount of random data for training is called stochastic training – more specifically, random gradient descent training.
- Ideally, we want to use each of our data to perform each step of the training because it gives us better training results, but obviously, this requires a lot of computational overhead. Therefore, we can use different data subsets for each training. This will not only reduce the computational overhead but also maximize the overall characteristics of the dataset.
- In each step of the cycle, we randomly grab 100 batches of data points in the training data. Then we use these data points as parameters to replace the previous placeholders train_step.
- The preceding process is called random gradient descent training. It is suitable in our model training, because it can ensure the running efficiency and the correctness of the

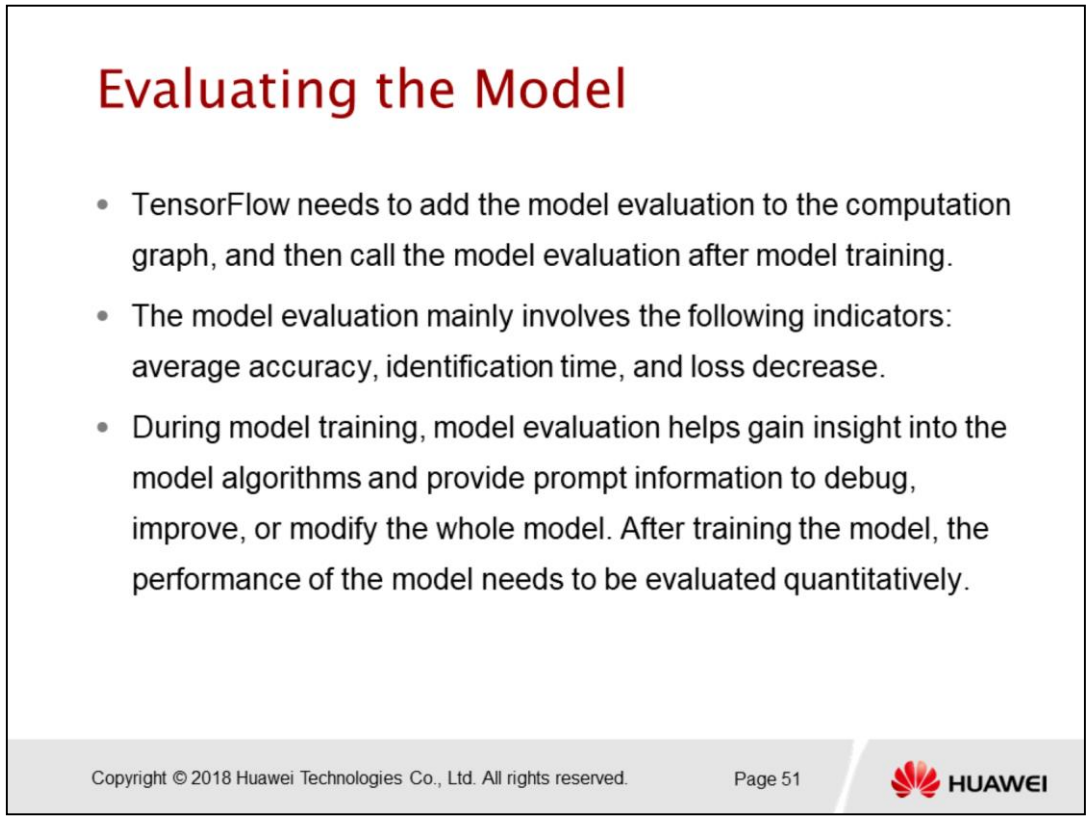
program. (In theory, we should use all the training data to get the correct gradient descent direction during each cycle, but it will be very time-consuming.)

Testing the Model

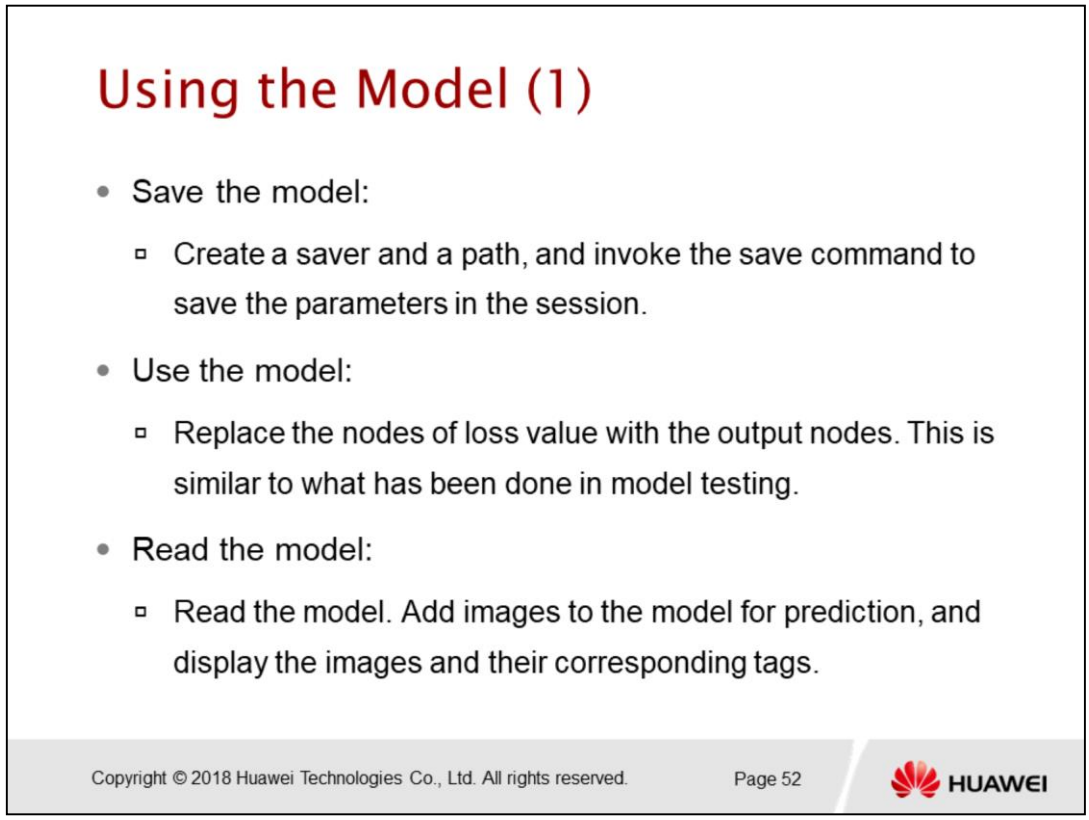
- Use the test data in MNIST to test the model. First, find out the labels that are correct. Since the tag is a vector consisting of 0 and 1, the maximum value, that is, 1, of the index position is located a category label. To determine the proportion of the correct predictors, we can convert the Boolean values into floating point numbers and then average them. we calculate the correctness of the learned model on the test dataset.
- For tf.argmax (y, 1), the return of the model prediction for any input x to the tag value, and tf.argmax (y_,1), the representative of the correct label, we can use tf.equal to test whether the prediction is a true tag match (the same as the index position indicates a match).

```
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
```


- We can use images to predict the accuracy of a category.
- First, the tf.argmax () function is used to obtain the predicted and actual label values of the images, and then the tf.equal () function is used to test whether the predicted value is consistent with the actual value. Finally, we get the accuracy of the model on the test dataset.
- The classification algorithm model predicts the classification value based on the numeric input, and the actual target is the sequence of 1 and 0. We need to measure the distance between the predicted value and the actual value. The loss function of the classification algorithm model cannot easily interpret whether a model is good or bad. Therefore, it is usually the proportion of the accurate classification prediction results that is used for evaluating a model.
- The algorithm for testing the error rate is as follows: Check whether the prediction result is the same as the actual label. If they are the same, the result is correct. If they are different, the result is incorrect. Divide the number of correct results by the total number of results to obtain the accuracy rate.



- However, model training does not always require model evaluation. We will show how to use it in regression algorithms and classification algorithms.
- Ideally, model evaluation requires a training dataset and a test dataset, and sometimes even a validation dataset.
- If you want to evaluate a model, you need to use a large number of data points. Once batch training is completed, we can reuse the model to predict batches of data points. However, if you want to complete random training, you have to create a separate estimator to handle batches of data points.
- No matter how the algorithm model predicts, we need to test the algorithm model. This is very important. Model evaluation should be performed on both the training and test datasets to determine whether the model is overfitting.
- The final result is 0.905. The result is very poor, but this is only a learning example. After some improvements, the accuracy rate can be increased to 97%. The best model can even get more than 99.7% accuracy.



- The model.ckpt file must exist in a specified folder. The place where the tmp/model.ckpt is stored needs at least one folder. Otherwise, the file cannot be saved.
- When the model is restored, just like when it is being stored, the tmp/model.ckpt is used. The file name is different from that of the other three files.
- Once the TensorFlow session is disabled, all the weights and biases of the training are lost. You need to retrain the model after restarting it. You can save your process using a class named tf.train.Saver.

Using the Model (2)

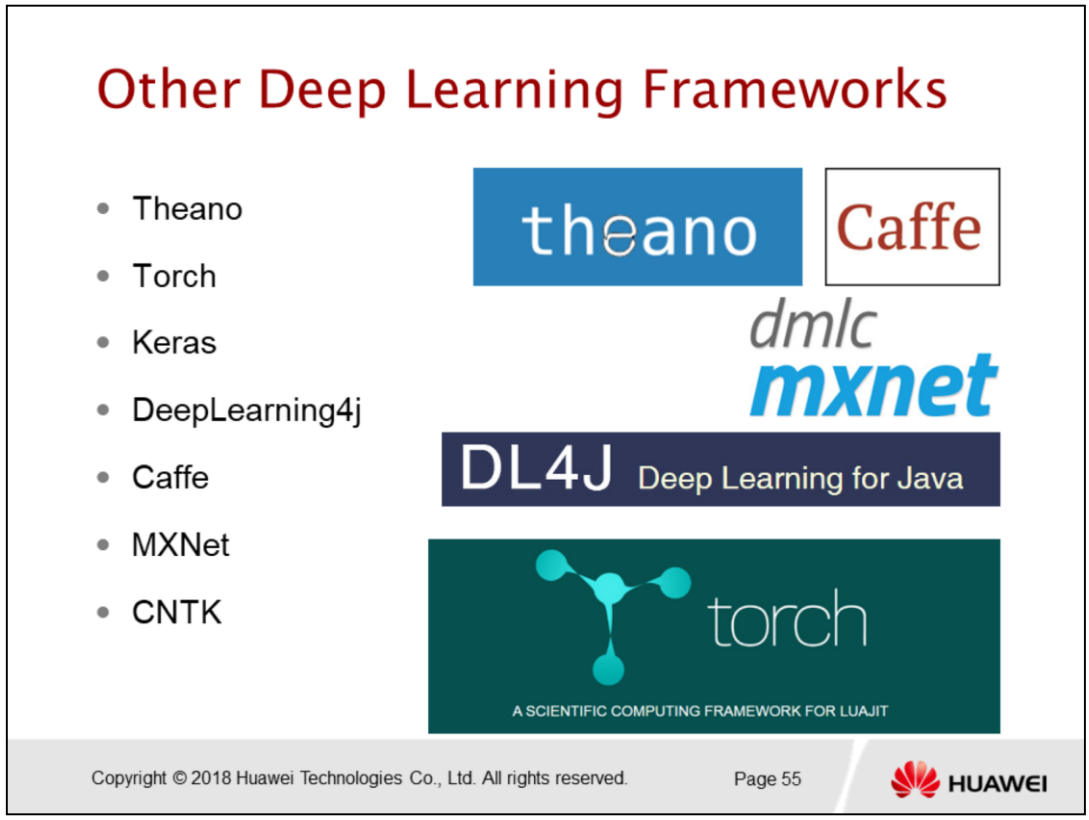
- Read the model. Add two images to the model to predict results and display the labels corresponding to the two images.
- The definition of the network model remains unchanged during code execution. Create a new session. All operations are performed in the new session.

```
After 0 training step(s), validation accuracy using average model is 0.103
After 1000 training step(s), validation accuracy using average model is 0.9044
After 2000 training step(s), validation accuracy using average model is 0.9174
After 3000 training step(s), validation accuracy using average model is 0.9258
After 4000 training step(s), validation accuracy using average model is 0.93
After 5000 training step(s), validation accuracy using average model is 0.9346
After 6000 training step(s), validation accuracy using average model is 0.94
After 7000 training step(s), validation accuracy using average model is 0.9422
After 8000 training step(s), validation accuracy using average model is 0.9472
After 9000 training step(s), validation accuracy using average model is 0.9498
After 10000 training step(s), test accuracy using average model is 0.9475
```


- The output of each line is the accuracy of the model. The accuracy rate of 1000 iterations is 91%. The accuracy rate of 10,000 iterations is 95%. Some errors occur twice, so the prediction may be incorrect.

Contents

HUAWEI

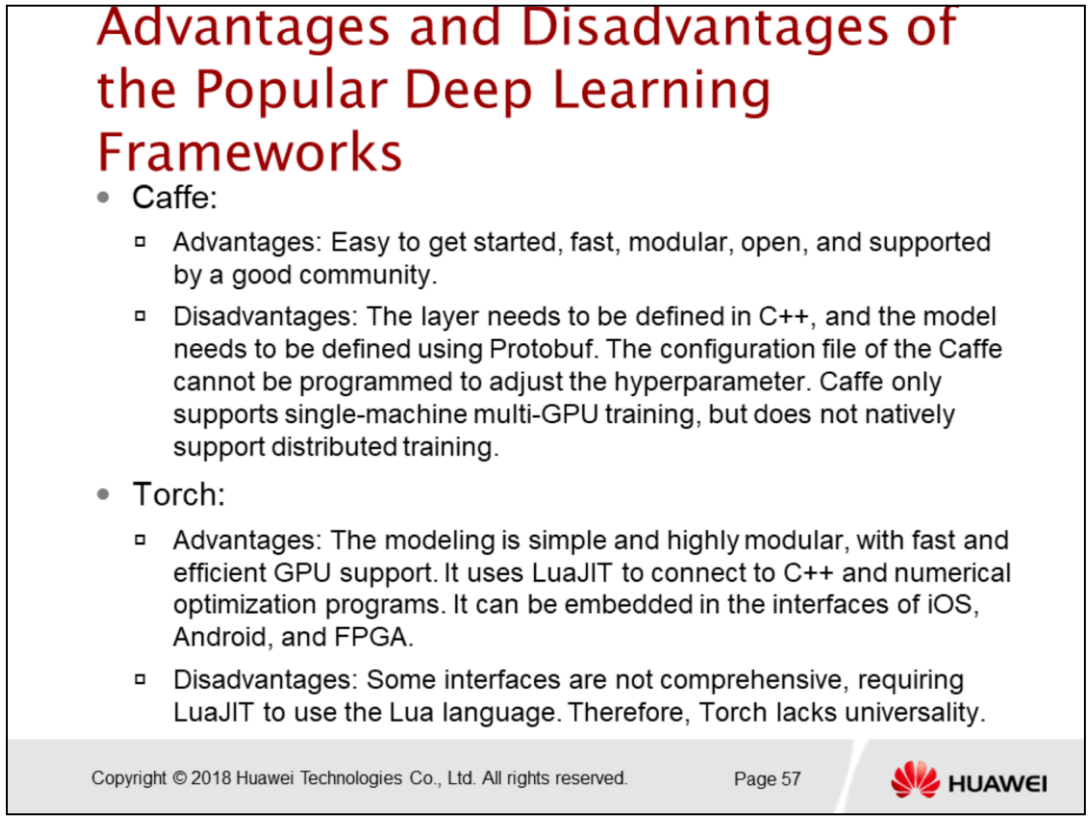


- Theano is a deep learning and machine learning framework that has been developed for more than 10 years. It is a Python library that allows to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. It features great scalability.
- Torch has good scalability, but some interfaces are not comprehensive. It needs LuaJIT to use the Lua language. Therefore, it lacks universality.
- Keras: It is a combination of the Theano framework and the TensorFlow front end. It has good transportability.
- DeepLearning4j: It is written in Java and Soala, and is applied to deep learning software on the Hadoop and Spark systems.
- Caffe: It has a powerful image classification framework. It is the deep learning framework that is the easiest to test and evaluate performance.
- MXNet: It is a deep learning library with good transportability and scalability. It provides some features of Torch, Theano, and Caffe. It is currently one of the most popular frameworks.
- CNTK: It is a deep learning toolkit developed by Microsoft. It features fast speed and a custom neural network description language, Brain Script. It is not as mature as TensorFlow, but inherently couples with Visual Studio.

Hardware Supported by Deep Learning Frameworks

Property	Caffe	Neon	TensorFlow	Theano	Torch
Core	C++	Python	C++	Python	Lua
CPU	✓	✓	✓	✓	✓
Multi-threaded CPU	✓Blas	x Only data loader	✓Eigen	✓Blas, conv2D, limited OpenMP	✓**Widely used**
GPU	✓	✓customized Nvidia backend	✓	✓	✓
Multi-GPU	✓(only data parallel)	✓	✓**Most flexible**	x Experimental version available	✓
Nvidia cuDNN	✓	x	✓	✓	✓
Quick deploy. on standard models	✓**Easiest**	✓	✓	x Via secondary libraries	✓
Auto. gradient computation	✓	✓Supports Op-Tree	✓	✓**Most flexible** (also over loops)	✓

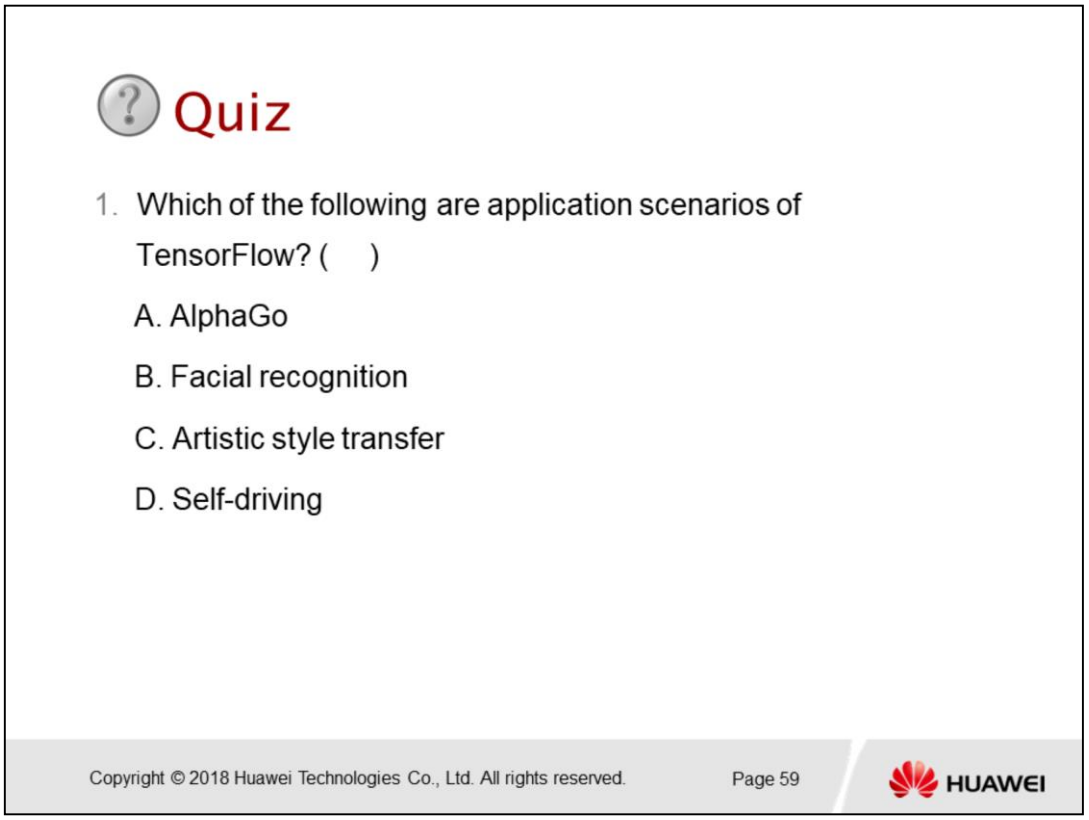
- The figure shows the general support of each framework for different hardware.



- TensorFlow, Torch, and MXNet all have intuitive and modular architecture, making development easier. However, TensorFlow compilation is faster than Theano. TensorFlow is clumsier than Torch, and more difficult to comprehend.

Conclusion

- This chapter describes the concepts, characteristics, basic knowledge, and commonly used modules of TensorFlow. It then introduces how to set up the development environment and create TensorFlow programs. It also describes the basic development steps using TensorFlow and internal mechanisms of TensorFlow. In the end, it touches upon other deep learning frameworks.


Quiz
1. Which of the following are application scenarios of TensorFlow? ( )
A. AlphaGo
B. Facial recognition
C. Artistic style transfer
D. Self-driving
Copyright © 2018 Huawei Technologies Co., Ltd. All rights reserved.
Page 59
HUAWEI

- Answer: ABCD

More Information

- Huawei E-Learning website:
  - http://support.huawei.com/learning/Index!toTrainIndex
- Huawei Support case library:
  - http://support.huawei.com/enterprise/servicecenter?lang=zh

Thank You

www.huawei.com