7. - Huawei MindSpore AI Development Framework

7.2 - MindSpore Application Development

MindSpore logo link

Gilvan Maia

10 Setembro, 2020

Instituto Universidade Virtual

Universidade Federal do Ceará

HUAWEI

Disclaimer

The following content is heavily based on HCIA-AI Course material by Huawei Technologies Co., Ltd., authored by Zhang Luxiao and Yuan Meng. Distribution is not allowed.

Index

- Environment Setup
- Programming Concepts
- Study Case

Environment Setup

MindSpore Environment Setup

Instructions for MindSpore 0.7.0, depends on Python 3.7.5

Can be compiled from source code, binary releases, or Docker images

We are using the pip install method, after setting a Miniconda manager

1. Activate your environment
2. pip install https://ms-release.obs.cn-north-4.myhuaweicloud.com/0.7.0-beta/MindSpore/ascend/ubuntu_x86/mindspore_ascend-0.7.0-cp37-cp37m-linux_x86_64.whl

Optionally, you can also install MindInsight for easy-to-use debugging and tuning capabilities (not covered)

3. pip install https://ms-release.obs.cn-north-4.myhuaweicloud.com/0.7.0-beta/MindInsight/ascend/ubuntu_x86/mindinsight-0.7.0-cp37-cp37m-linux_x86_64.whl

HUAWEI

Programming Concepts

Tensors

MindSpore stores data in tensors, which have some common operations

- asnumpy()
- size()
- dim()
- dtype()
- set_dtype()
- tensor_add(other: Tensor)
- tensor_mul(other: Tensor)
- shape()
- __Str__# (conversion into strings)

Main MindSpore Expression Modules

Module	Description
communication	Data loading module, which defines the dataloader and dataset and processes data such as images and texts.
dataset	Dataset processing module, which reads and pro-processes data.
common	Defines tensor, parameter, dtype, and initializer.
context	Defines the context class and sets model running parameters, such as graph and PyNative switching modes.
nn	Defines MindSpore cells (neural network units), loss functions, and optimizers.
ops ops.composite ops.operations	Defines basic operators and registers reverse operators.
train	Training model and summary function modules.
utils	Utilities, which verify parameters. This parameter is used in the framework.

official 0.7.0 API documentation here

Operation

```
class Softmax(Cell):
    """
    Softmax activation function. (doc string)
    """

    def __init__(self, axis=-1):
        super(Softmax, self).__init__()
        self.softmax = _selected_ops.Softmax(axis)

    def construct(self, x):
        return self.softmax(x)
```

Common operations in MindSpore

mindspore.ops.operations: tensor array-related operators
ExpandDims, Squeeze, Concat, OnesLike, Select, StridedSlice, ScatterNd, etc

mindspore.ops.operations: mathematical and NN operators
AddN, Cos, Sub, Sin, LogicalAnd, MatMul, LogicalNot, RealDiv, Less, ReduceMean, Argmax, L2Normalize

mindspore.nn: building blocks, i.e., layers and units for NNs
Conv2D, MaxPool, AvgPool, Flatten, Softmax, ReLU, Sigmoid, ReLU, BatchNorm, SoftmaxCrossEntropy, SigmoidCrossEntropy, SGD, etc

mindspore.nn.probability: random operators
mindspore.dataset.transforms.vision: process mage augmentations

Cell (1/2)

Cell defines the basic module for calculation. The objects of the cell can be directly executed.

- __init__() initializes and verifies modules such as parameters, cells, and primitives.
- construct() defines the execution process. In graph mode, a graph is compiled for execution and is subject to specific syntax restrictions.
- bprop() is optional and implements the reverse direction (backpropagation) of customized modules. Automatic differential is used by default to calculate the reverse of the construct part.

Cells (2/2)

Predefined cells in MindSpore include common building blocks

- Loss functions (Softmax Cross Entropy With Logits and MSELoss),
- Optimizers (Momentum, SGD, and Adam)
- Network training functions (aka network training package), such as
  - TrainOneStepCell network gradient calculation and update based on an optimizer
  - WithLossCell, wraps a network with loss function
  - WithGradCell, for gradient calculation

MindSporeIR

MindSporeIR is a compact, efficient, and flexible graph-based functional IR

It represents functional semantics such as free variables, high-order functions, and recursion

MindSporeIR is a program carrier in the process of automatic differentiation and compilation optimization

Each graph represents a function definition graph and consists of ParameterNodes, ValueNodes, and ComplexNodes (CNodes)

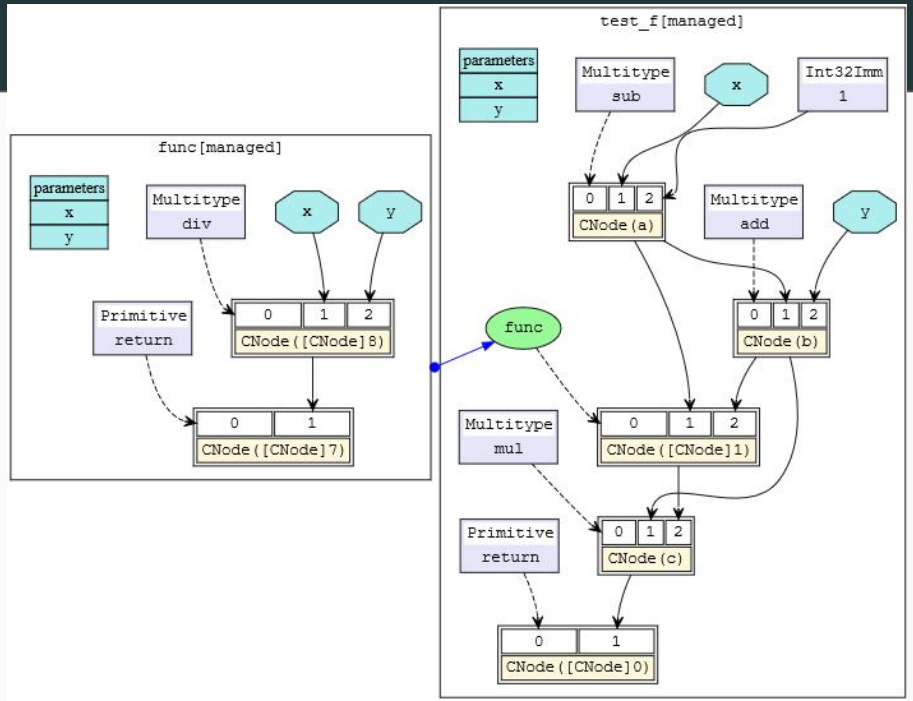
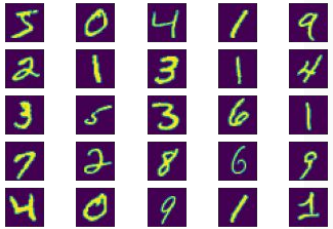


Illustration of the def-use relationship

Study Case

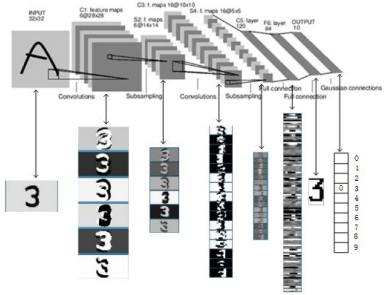
MNIST Digit Classification

Data



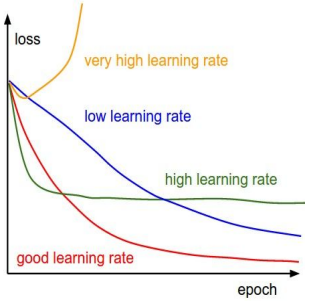
- 1. Data loading
- 2. Pre-processing pipeline

Network



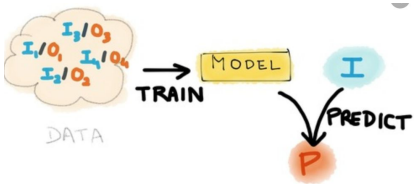
- 3. Network definition
- 4. Weight initialization
- 5. Network execution

Model



- 6. Loss function
- 7. Optimizer
- 8. Training iteration
- 9. Model evaluation

Application



- 10. Model I/O
- 11. Inference

Link for LeNet5 figure

HUAWEI

Modules and Cells Covered

```
import mindspore.dataset as ds
import mindspore.nn as nn
from mindspore import context

from mindspore.nn.metrics import Accuracy
from mindspore.nn.loss import SoftmaxCrossEntropyWithLogits

from mindspore.common import dtype as mstype
from mindspore.common.initializer import TruncatedNormal

from mindspore.train import Model
from mindspore.train.serialization import load_checkpoint, load_param_into_net
from mindspore.train.callback import ModelCheckpoint, CheckpointConfig, LossMonitor

import mindspore.dataset.transforms.vision.c_transforms as CV
import mindspore.dataset.transforms.c_transforms as C
from mindspore.dataset.transforms.vision import Inter
```

Thank You!

Next: Lab Guide 06