Deep Learning on ROCm

ROCm Tutorial – Part 5
Introduction

- GPUs have become the accelerator of choice for Deep Neural Networks (DNNs)
- DNNs are rapidly changing the world we live in today by providing intelligent data driven decisions across multiple industries
- This tutorial serves as an introduction to scientists who want to leverage the power of ROCm for accelerating DNNs
AMD is fully committed to this goal

Specialized GPUs in the form of its Instinct line of products to accelerate deep learning

Ports open source libraries such as TensorFlow and PyTorch to support ROCm based AMD GPUs

Supports the development of faster interconnects for multi-GPU communication
Prerequisites

- This tutorial assumes you have a working ROCm installation

- Knowledge on the following materials:
  - Docker
  - Deep Learning Basics
  - Familiarity with Pytorch
  - Familiarity with TensorFlow
  - Familiarity with Keras
Online Guides

1. All online guides for ROCm can be found here ( https://rocmdocs.amd.com/en/latest/ )

2. Following guides are helpful for more knowledge about ROCm support for Machine Learning community:

3. MIOpen: Core Machine Learning Library for ROCm

4. A gallery of different DNN models in ROCm:
   • https://github.com/IntuitionMachine/SEEDBank

5. Information about RCCL library used internally during DNN training:
   • https://rccl.readthedocs.io/en/develop/
Goals

- Setup process for TensorFlow and PyTorch using prebuilt Docker images will be shown.
- We will learn to run popular DNN models such as CNN and LSTMs on these frameworks.
- We will also learn how to do Multi-GPU DNN training on ROCm.
TensorFlow Installation and MNIST Example
MNIST Dataset

- MNIST is one of the most popular datasets used in the Deep Learning community
- It has a corpus of handwritten digits
  - 60k training samples
  - 10k test samples
- The goal is to train a deep neural network to identify the handwritten digits (0-9)
- More information about this dataset can be found here [http://yann.lecun.com/exdb/mnist/]
Deep Neural Network Configuration

- We are going to use a very simple Deep Neural Network for our task:

- Network has 4 layers
  1. Input layer (784 neurons)
  2. Hidden layer 1 (128 neurons)
  3. Hidden layer 2 (256 neurons)
  4. Output Layer (10 neurons)

- Loss function will be a cross entropy loss that applies the softmax function

- A batch size of 256 is used

- Learning rate is 0.1 and total training steps are 2000

- Gradient descent is used to update the weights and bias values
A Deep Neural Network for classifying digits on the MNIST dataset
Concepts

- The recommended way to install TensorFlow is to use AMD’s prebuilt Docker images.
- Users must ensure that they have a working ROCm installation on their system.
  - If not, please follow our installation tutorial and install ROCm depending on your OS version.
- Please follow the hands-on tutorial Chapter 5.1: TensorFlow_ROCm.
Demo: Training a DNN on MNIST dataset using TensorFlow
PyTorch Installation and LSTM Example
Word Level Language Modelling

- For this example, we are going to use the provided PyTorch example on Word Level Language Modelling

- This example trains a multilayer Long-Term Short Memory network (LSTM)

- LSTMs are useful for capturing time series data

- Examples of time series data: speech, text etc.

- The example will run a LSTM network on Wikitext2 dataset
  https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/
A LSTM architecture for modelling time series data
LSTM Network Configuration

The LSTM Network used in the example has the following configuration:

- 2 hidden layers
- 200 hidden neurons per layer
- 0.2 dropout rate
- 0.25 gradient clipping
- 200 size of word embeddings
Installation Concepts

- The recommended way to install PyTorch is to use AMD’s prebuilt docker images.
- With each release of ROCm, a docker image will be released.
- Please follow the hands-on tutorial Chapter 5.2: PyTorch ROCm.
Demo: Training a LSTM network on PyTorch
AMD has developed a rich source of Deep Learning examples for the interested user

This is the SEEDBank collection which can be found here: https://github.com/IntuitionMachine/SEEDBank

The repository contains a curated collection of different Deep Learning examples developed using TensorFlow Keras and are ready to run on ROCm

Each example has a corresponding python notebook that demonstrates the example in detail

Some examples might require the user to install missing python packages that are not part of the TensorFlow docker image
Multi-GPU Example on Keras
Multi-GPU Deep Learning

- Training Deep Neural Networks on multiple-GPUs is a common practice
- Using multi-GPUs enables faster training times as well as train larger models that cannot fit on a single GPU
- ROCm seamlessly supports Multi-GPU Deep Learning
- Users can develop their multi-GPU training models using high-level frameworks
- In this example, we will be using the Keras library to train a DNN on the MNIST dataset using multiple GPUs
Demo: Training CNN on multiple-GPUs using Keras
Conclusion

In this tutorial we have looked at how to set up high level Machine learning frameworks on ROCm based systems.

High-level framework such as TensorFlow and PyTorch are easy to use on ROCm systems.

Existing code is directly portable.

No effort required from the developer.

Deep Learning research community can start using ROCm based GPUs for their work without any issues.

AMD is committed to expanding this support to more and more Machine learning frameworks over time.