Super-Resolution on RISC-V

Filip Blagojević
Outline

▪ RISC-V - Introduction
▪ Locally booted OS on RISC-V
▪ Requirements for Super-Resolution on RISC-V
▪ Photo-Realistic Single Image Super-Resolution Using Generative Adversarial Network
RISC-V

- RISC-V – Open ISA based on RISC principles
- RISC-V ISA can be used freely (anyone can design, manufacture and sell RISC-V chips and software)
- Designed to be used in high-end computing as well as embedded systems
- Open RISC-V implementations are available
  - https://riscv.org/risc-v-cores/
- RISC-V started at UC Berkeley in 2010 and is supported by industry, academia and volunteers
Locally Booted OS on RISC-V

- We started from Fedora-RISCV kernel image with NBD-backed root FS:
  - https://github.com/rwmjones/fedora-riscv-kernel/tree/sifive_u540

- Building kernel is simple but booting relies on the NBD root filesystem

- Booting from a local root filesystem requires several steps including changes in the Makefile (fedora-riscv-kernel)
  - The SD card is required to have the first partition written with the bbl (Berkeley Boot Loader)
  - Second partition needs to have the kernel image (see above)
  - Third partition has rootfs (download Fedora RISC-V disk image from: https://fedorapeople.org/groups/risc-v/disk-images/stage4-disk.img.xz)
Makefile Changes

- Fedora-RISC-V kernel script Makefile changes:

1) `ROOTFS = UUID=e06a1845-3577-4e35-92a9-015b3042b3f2` → `ROOTFS = / dev / <root fs partition>`

2) `CMDLINE="root=$(ROOTFS) netroot=nbd:$NBD rootfstype=ext4 rw rootdelay=5 ip=dhcp rootwait console=ttysI0"` → `CMDLINE="root=$(ROOTFS) rootfstype=ext4 rw rootdelay=5 ip=dhcp rootwait console=ttysI0"`

3) `dracut -m "nbd network base" $@ -t $(uname -r) --no-kernel --force -v` → `dracut $@ -t $(uname -r) --no-kernel --force -v`

- Currently dnf package manager not working when booted from the SD card
Which Machine Learning Framework?

- Many to choose from ...
  - Tensorflow
    - Relies on Bazel to compile
    - Bazel relies on JDK which is not fully available on RISC-V
  - Torch
    - Lots of Python dependencies
    - Requires luaJIT which is not yet fully available on RISC-V
  - Theano – lack of GAN examples
  - Keras, Caffe

- We chose Pytorch!
  - Few dependencies

DAC, June 2018
OpenBLAS on RISC-V

- OpenBLAS is a linear algebra library with optimized routines

- RISC-V is not yet supported

- We compiled generic C and fortran routines, instead of architecture specific
  - Current RISC-V hardware does not support vector instructions

- Significant performance gains:
  - Switching to OpenBLAS improved the performance of the full GAN inference by a factor of more than 20x
  - From minutes to ~16s
Super-Resolution on Risc-V

- Based on GAN approach for super-resolution
  - Train generative model with the goal of fooling a differentiable discriminator, trained to distinguish super-resolved images from real images
  - generator can learn to create solutions that are highly similar to real images and difficult to classify by discriminator
  - HR images are only available during the training. LR images are obtained from HR, by applying Gaussian filter followed by downsampling
- Utilize deep residual network with skip connection
Generator and Discriminator Networks

**Generator Network**

- Input
- Conv
- PreLU
- Conv
- BN
- PreLU
- Conv
- BN
- Elementwise Sum
- Conv
- PixelShuffler x2
- PreLU
- Conv
- ISR

**Discriminator Network**

- Input
- Conv
- Leaky ReLU
- Conv
- BN
- Leaky ReLU
- Conv
- BN
- Leaky ReLU
- Conv
- BN
- Dense (1024)
- Leaky ReLU
- Dense (1)
- Sigmoid
- ?

Western Digital
Results

- We use implementation from Aitor Ruano, https://github.com/aitorzip/PyTorch-SRGAN
- We perform training on celebrity image set (CelebA), ~250K images
- We scale down the starting image 4 times (176x176px → 44x44px)
Results

- We also scale down the staring image 8 times (176x176px → 22x22px)

178x218

44x44

22x22

HR
Fake

HR
Fake
Thank You!