Peraton LABS
LABORATORY FOR TELECOMMUNICATION SCIENCES

TorchSig: An Open-Source Signals Processing Machine Learning Toolkit

Team members:

Garrett Vanhoy, Luke Boegner, Manbir Gulati, Phil Vallance, Rob Miller, Bradley Comar, Silvija Kokalj-Filipovic, Dresden Feitzinger, Craig Lennon

Agenda

Overview

- TorchVision/Audio, PyTorch, Pytorch Lightning
- TorchSig Package Structure
- Modulation Classification

Adding a New Signal

- The Dataset Class
- **Hands-On**: Using a Dataset Class

Adding a New Transform

- The Transform Class
- **Hands-On:** Using a Transform Class

Models

- Pre-trained CV Models
- Custom models, loss, etc.

Peraton | LABS

Overview

Design Methodology

- Mirror APIs of existing frameworks backing SoTA results (TorchVision's Dataset and Transform)
- If possible, do not force dependency on a particular ML framework
- Make it easy to define new datasets that could exist on disk in many formats
- Make it easy to introduce impairments/augmentations/transforms that efficiently manipulate data before being presented to the model for training
- Provide many examples using a commonly used framework that supports multi-GPU or other accelerator-based training

TorchSig Package Structure

- **Datasets:** RadioML, Sig53, Synthetic
- **Models:** EfficientNet, XCiT
- **Transforms:**
  - General: Compose, Lambda, RandomApply, Concatenate, RandAugment
  - Deep Learning Techniques: CutMix, MixUp, CutOut, PatchShuffle
  - Expert Feature: InterleaveComplex, ComplexTo2D, Real/Imag, Spectrogram, Wavelet
  - Signal Processing: Normalize, RandomResample
  - Impairments: TimeShift, TimeCrop, FreqShift, IQImbalance, SpectralInversion, TimeReverse
  - Wireless Channel: TargetSNR, AddNoise, RayleighFading, PhaseShift
- **Utililities:**
  - Visualizers, SignalFileDataset, SignalTensorDataset

Modulation Classification

- TorchSig Methodology
  1. Define a Dataset class with *__getitem__(idx: int)* function that produces an example
  2. Define a Transforms pipeline that impairs/augments/transforms data
  3. Define a model, loss, optimizer, scheduler
  4. Torch/PyTorchLightning:
     1. Wrap Dataset in DataLoader with parameters: batch_size, num_workers,
     2. Wrap model, loss, optimizer, scheduler in LightningModule and implement train_step, val_step
     3. Run training with PL-Trainer (num_gpus, num_epochs, etc…)

Adding a New Signal

The TorchSig Dataset

- Inherits from torch.utils.data.Dataset
  - I know, not supposed to do that, it's probably not necessary.
  - A Dataset is just a __len__ and a __getitem__ implementation (Generator)

- Possibilities in __getitem__
  - Read data from a file in SigMF Format
  - Read data from a file in hdf5 format
  - Generate data using the idx as a seed for a random number generator
  - Request data from remote database

Modulation Classification Example

- We'll use the Sig53 Classifier Example as a Starting Point
- Change Sig53 Dataset into Modulations Dataset
- **Train with BPSK, QPSK:** No Transforms
- **Train with BPSK, QPSK:** AWGN

Adding a New Signal

- Modify ConstellationDataset to have a new “noise only” signal and Re-train
- **Train with BPSK, QPSK, Noise:** No Transforms
- **Train with BPSK, QPSK, Noise:** AWGN

Peraton | LABS

Adding a New Transform

The TorchSig Transform

- Mirrors TorchVision Transforms
  - Doesn't inherit from it though!
  - It's just a __call__(self, data) implementation.
    - Doesn't take a batch, a good DataLoader will parallelize calls to transform pipelines/datasets.

- Possibilities in _call_
  - Add an RF impairment
  - Call another transform (RandAugment, Compose)
  - Pass through (Identity)

- Target Transforms
  - If you want to modify the label for a piece of data based on a transform, you can do that. Won't cover.

Modulation Classification Example

- We'll use the Previous Example as a Starting Point

- **Train with BPSK, QPSK:** New Pipeline:
  - Normalize
  - RandomApply
    - RandomTimeShift
  - AWGN

Peraton | LABS

Models

The TorchSig Models

- Mirrors TorchVision Models
  - Doesn’t inherit from it though!
  - Many CV models can be used with num_channels=1 or 2
  - Other internals change with PyTorch’s dynamic graph

The TorchSig Models

- We'll use the Previous Example as a Starting Point

- **Train with BPSK, QPSK:** New Model
  - Dense Layers 128, 64, 32, 16 with softmax output