SafeAI 모두의연구소 오리엔테이션

Episys Science, Inc.
이상수

모두의연구소 Safe AI 연구실 개설 계획서

아주 많은 데이터로 훈련된 딥러닝 모델은 아주 정확한 성능을 낼 수 있지만, 때로는 그렇지 못합니다. 예를 들어, 데이터셋의 크기가 작거나, 데이터 자체에 노이즈가 많거나, 원 데이터셋의 분포로부터 오지 않은 데이터(Out-of-distribution data)에 대해서도 높은 확률값을 우리에게 줄 수 있습니다. 어떻게든 결정을 내려야 하는 자율주행 자동차를 예를 들자면 치명적인 결과를 낼 수 있을 것입니다. 이렇게 불확실성을 정확하게 표현할 수 있는 연구의 중요성이 높아지고 있고, 또 앞으로 해결해야할 아주 큰 이슈들 중 하나임에는 틀림없습니다.

Safe AI 연구실은, 넓은 범위의 딥러닝 모델들에 내재되어 있는 예측 불확실성을 다루는 논문들에 대해 Tensorflow, Pytorch 등 딥러닝 라이브러리로의 프로토타입들을 서로 도와가며 구현해보고, 공유하며 함께 고민하는 연구실이 되고자 합니다. 또, 센서의 오작동으로 발생할 수 있는 예측하지 못했던 입력의 변화에 대해서도 대응할 수 있는 방법을 연구합니다. (e.g. Surprise Based Learning, https://ieeexplore.ieee.org/document/4599429/)

EpiSci

이런 주제들로 같이 연구합니다:

연구자들의 논문에는 실험에 직접 사용했던 코드를 참고할 수 있는 경우가 많습니다. 논문에 직접 사용했던 코드를 볼 수 있기도 하고, 때로는 다른 사람들이 자신의 해당 논문 구현물을 공유해 주기도 합니다. 하지만 조금 더 들여다보면 다른 사람들이 아주 쉽게 가져다 쓸 수 있을 만큼은 정리되어 있지 않은 경우도 있는데요, 이 부분을 조금 더 보완해서 '쉽게 가져다 쓸 수 있는 코드'를 작성하는 방법을 같이 연구하고 고민할 수 있으면 좋겠습니다.
또한 딥러닝 프레임워크에는 상위 레벨 API에서 공식적으로, 혹은 비공식적으로 지원하는 추상화된 논문 구현 모델이 많이 있습니다.(VGG, ResNet, MobileNet, ...) 이 모델들의 내부 구현 코드 또한 참고하여, 우리가 직접 불확실성 모델들을 어떻게 하면 다른 사람들이 더 정확하게, 그리로 더 편하게 가져다 쓸 수 있을지 같이 생각해 봅니다.

- 개인별 / 팀별 구현할 논문을 선정하고, 구현 진행상황 및 과정을 공유합니다
- 구현한 실험 결과들에 대해 어떻게 하면 더 성능을 높일 수 있을지, 다른 방법을 사용하면 어떨지 직접 실험해보고, 시각화해 봅니다.

EpiSci

Deep learning frameworks

- Tensorflow -> tf.contrib (slim, learn ..)
- Keras -> Keras.application
- Pytorch -> torchvision.models

tensorflow.contrib

TensorFlow contrib

Any code in this directory is not officially supported, and may change or be removed at any time without notice.

The contrib directory contains project directories, each of which has designated owners. It is meant to contain features and contributions that eventually should get merged into core TensorFlow, but whose interfaces may still change, or which require some testing to see whether they can find broader acceptance. We are trying to keep duplication within contrib to a minimum, so you may be asked to refactor code in contrib to use some feature inside core or in another project in contrib rather than reimplementing the feature.

When adding a project, please stick to the following directory structure: Create a project directory in `contrib/`, and mirror the portions of the TensorFlow tree that your project requires underneath `contrib/my_project/`.

For example, let's say you create foo ops in two files: `foo_ops.py` and `foo_ops_test.py`. If you were to merge those files directly into TensorFlow, they would live in `tensorflow/python/ops/foo_ops.py` and `tensorflow/python/kernel_tests/foo_ops_test.py`. In `contrib/`, they are part of project `foo`, and their full paths are `contrib/foo/python/ops/foo_ops.py` and `contrib/foo/python/kernel_tests/foo_ops_test.py`.

Why we would use tf.contfrib.* over tf.***?**
https://stackoverflow.com/questions/44805675/tensorflow-tf-layers-vs-tf-contrib-layers

TensorFlow-Slim

TF-Slim is a lightweight library for defining, training and evaluating complex models in TensorFlow. Components of tf-slim can be freely mixed with native tensorflow, as well as other frameworks, such as tf.contrib.learn.

Usage

```
import tensorflow.contrib.slim as slim
```

Why TF-Slim?

TF-Slim is a library that makes building, training and evaluation neural networks simple:

- Allows the user to define models much more compactly by eliminating boilerplate code. This is accomplished through the use of argument scoping and numerous high level layers and variables. These tools increase readability and maintainability, reduce the likelihood of an error from copy-and-pasting hyperparameter values and simplifies hyperparameter tuning.
- Makes developing models simple by providing commonly used regularizers.
- Several widely used computer vision models (e.g., VGG, AlexNet) have been developed in slim, and are available to users. These can either be used as black boxes, or can be extended in various ways, e.g., by adding "multiple heads" to different internal layers.
- Slim makes it easy to extend complex models, and to warm start training algorithms by using pieces of pre-existing model checkpoints.

Follow TF's contribution guideline

- Tensorflow.contrib 패키지 구조 파악
  (contrib.gan, contrib.layers, contrib.rnn, contrib.slim)

- Git basic usage (Issues, PR, ..)

- TEST code writing

- Markdown

- Styleguide – pylint

https://github.com/tensorflow/tensorflow/blob/master/CONTRIBUTING.md

다음 시간까지:

- Python Pacaking
  => python packaging, from * import *, …

- Contrib package 구조 분석
  (contrib.gan, contrib.layers, contrib.rnn, contrib.slim)

- TEST code analysis(tf.contrib.slim)

- Pylint Styleguide

- 공통: Markdown, Git basic usage (Issues, PR, ..)

- 개인별로 Tutorial 하나씩 작성 -> PR

다음 시간에는,

- Fork EpisysScience/SafeAI-exp to your own repository
- Tensorflow contribute guideline를 따르도록
  간단한 모델을 models/에 구현하고, example/에 사용 예제 작성
  => `pip3 uninstall -y safeai; pip3 install .` 로 확인
- github.com/episysscience/safeai-exp에 Pull Request 날리기!

https://github.com/tensorflow/tensorflow/blob/master/CONTRIBUTING.md