TorchExpo

Release 1.1.0a4

Omkar Prabhu

Oct 18, 2020

CONTENTS

1 Get Started with TorchExpo 3

2 FAQ 5

3 TorchExpo Android 7

4 TorchExpo iOS (Coming Soon) 9

5 torchexpo.modules 11

6 torchexpo.vision 13

7 torchexpo.nlp 19

8 Persons Of Interest 21

Python Module Index 23

Index 25

TorchExpo is a collection of models and extensions for mobile deployment built on top of PyTorch.

CHAPTER

ONE

GET STARTED WITH TORCHEXPO

1.1 Prerequisites

1.1.1 Python

It is recommended that you use Python 3.5 or greater, which can be installed from Python website.

1.1.2 Package Manager

To install the TorchExpo, you will need to use one of the supported package managers: pip. pip is the recommended package manager as it will provide you all of the dependencies in one, sandboxed install, including Python.

pip

Python 3

If you installed Python via the Python website, `pip` was installed with it. If you installed Python 3.x, then you will be using the command `pip3`.

Anaconda

Coming Soon

1.2 Installation

1.2.1 pip

To install TorchExpo via pip, use following command:

```
# Python 3.x
pip3 install torchexpo
```

1.2.2 Anaconda

To install TorchExpo via Anaconda, use following command:

Coming Soon

1.3 Building from Source

For the majority of TorchExpo users, installing from a pre-built binary via a package manager will provide the best experience. However, there are times when you may want to install the nightly TorchExpo code, whether for testing or actual development. To install the latest TorchExpo code, you will need to build TorchExpo from source.

```
git clone https://github.com/torchexpo/torchexpo.git
cd torchexpo
python3 setup.py install
```

CHAPTER

TWO

FAQ

Q: Do I need to have a GPU machine for extraction of output in torchscript or ONNX format? No, as long as you have a reliable internet connection, you'll be able to download the pre-trained models and run the conversion smoothly.

Q: Does the TorchExpo project support models developed by individuals? No, not at this point. We are however looking at ways to better support the community. If you have suggestions or inputs, please reach out to the maintainers.

Q: How do I contribute code to the project? Please also see the TorchExpo Contributor Guide for contribution guidelines.

Q: What if i would like to deliver a PyTorch tutorial at a conference or otherwise? We encourage community members to showcase their work wherever and whenever they can. Please reach out to torchexpo@gmail.com for more technical support.

Q: What if I am a company looking to use TorchExpo for development, can I be granted or purchase a seat to drive the project direction? No, the TorchExpo project is strictly driven by the maintainer-driven project philosophy and does not have a board or vehicle to take financial contributions relating to gaining influence over technical direction. However you can contribute to the technical improvments like others.

CHAPTER

THREE

TORCHEXPO ANDROID

3.1 Prerequisites

- Kotlin
- Android Studio

3.2 Installation

- Git clone this/forked repository

```
git clone https://github.com/torchexpo/android.git
OR
git clone https://github.com/<YOUR_GITHUB_USERNAME>/android.git
```

- Open the folder as an existing project in Android Studio
- Build the application

CHAPTER

FOUR

TORCHEXPO IOS (COMING SOON)

4.1 Prerequisites

TBD

4.2 Installation

TBD

CHAPTER

FIVE

TORCHEXPO.MODULES

class torchexpo.modules.**ImageClassificationModule**(*model*, *model_name*, *model_example*)

Image Classification module for all vision models under vision/image_classification

Parameters

- **model** – Pretrained model which is used for conversion
- **model_name** – Name of the model
- **model_example** – Example tensor used as example input while extraction

extract_onnx(*opset_version=None*)

Extract onnx from image classification model

Parameters **opset_version** – Opset version used while ONNX conversion

extract_torchscript()

Extract torchscript module from image classification model

class torchexpo.modules.**ImageSegmentationModule**(*model*, *model_name*, *model_example*)

Image Segmentation module for all vision models under vision/image_segmentation

Parameters

- **model** – Pretrained model which is used for conversion
- **model_name** – Name of the model
- **model_example** – Example tensor used as example input while extraction

extract_onnx(*opset_version=11*)

Extract onnx from image segmentation model

Parameters **opset_version** – Opset version used while ONNX conversion

extract_torchscript()

Extract torchscript module from image segmentation model

class torchexpo.modules.**SentimentAnalysisModule**(*model*, *model_name*, *model_example*)

Sentiment Analysis module for all nlp models under nlp/sentiment_analysis

Parameters

- **model** – Pretrained model which is used for conversion
- **model_name** – Name of the model
- **model_example** – Example tensor used as example input while extraction

extract_onnx(*opset_version=None*)

Extract onnx from sentiment analysis model

Parameters **opset_version** – Opset version used while ONNX conversion

extract_torchscript()
Extract torchscript module from sentiment analysis model

class torchexpo.modules.**TorchExpoModule**(*model*, *model_name*, *model_example*)
Base class for all PyTorch models

Parameters

- **model** – Pretrained model which is used for conversion
- **model_name** – Name of the model
- **model_example** – Example tensor used as example input while extraction

extract_caffe2_mobile()
Extracts model in Caffe2 Mobile format

extract_onnx(*opset_version=None*)
Extracts model in ONNX format

Parameters **opset_version** – Opset version used while ONNX conversion

extract_torchscript()
Extracts model in TorchScript format

get_extracted_file_name()
Returns file name for output

print_message(*output_type*)
Prints message

Parameters **output_type** – Name of the output format used for printing

CHAPTER

SIX

TORCHEXPO.VISION

6.1 Image Classification

Image Classification is a fundamental task that attempts to comprehend an entire image as a whole. The goal is to classify the image by assigning it to a specific label. It refers to images in which only one object appears and is analyzed. In contrast, object detection involves both classification and localization tasks, and is used to analyze more realistic cases in which multiple objects may exist in an image.

Example:

```
>>> from torchexpo.vision import image_classification
>>>
>>> model = image_classification.squeezenet1_0()
>>> model.extract_torchscript()
>>> model.extract_onnx()
```

6.1.1 AlexNet

torchexpo.vision.image_classification.**alexnet**()

AlexNet Model pre-trained on ImageNet

6.1.2 VGG

torchexpo.vision.image_classification.**vgg11**()
VGG11 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**vgg11_bn**()
VGG11_BN Model pre-trained on ImageNet

torchexpo.vision.image_classification.**vgg13**()
VGG13 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**vgg13_bn**()
VGG13_BN Model pre-trained on ImageNet

torchexpo.vision.image_classification.**vgg16**()
VGG16 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**vgg16_bn**()
VGG16_BN Model pre-trained on ImageNet

torchexpo.vision.image_classification.**vgg19**()
VGG19 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**vgg19_bn**()
VGG19_BN Model pre-trained on ImageNet

6.1.3 ResNet

torchexpo.vision.image_classification.**resnet18**()
ResNet18 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**resnet34**()
ResNet34 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**resnet50**()
ResNet50 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**resnet101**()
ResNet101 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**resnet152**()
ResNet152 Model pre-trained on ImageNet

6.1.4 SqueezeNet

torchexpo.vision.image_classification.**squeezenet1_0**()
SqueezeNet 1.0 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**squeezenet1_1**()
SqueezeNet 1.1 Model pre-trained on ImageNet

6.1.5 DenseNet

torchexpo.vision.image_classification.**densenet121**()
DenseNet-121 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**densenet169**()
DenseNet-169 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**densenet161**()
DenseNet-161 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**densenet201**()
DenseNet-201 Model pre-trained on ImageNet

6.1.6 Inception v3

torchexpo.vision.image_classification.**inceptionv3**()
Inception v3 Model pre-trained on ImageNet

6.1.7 GoogLeNet

torchexpo.vision.image_classification.**googlenet**()
GoogLeNet (Inception v1) Model pre-trained on ImageNet

6.1.8 ShuffleNet v2

torchexpo.vision.image_classification.**shufflenet_v2_x0_5**()
ShuffleNet V2 0.5x Model pre-trained on ImageNet

torchexpo.vision.image_classification.**shufflenet_v2_x1_0**()
ShuffleNet V2 1.0x Model pre-trained on ImageNet

6.1.9 MobileNet v2

torchexpo.vision.image_classification.**mobilenet_v2**()
MobileNet V2 Model pre-trained on ImageNet

6.1.10 ResNext

torchexpo.vision.image_classification.**resnext50_32x4d**()
ResNext-50 32x4d Model pre-trained on ImageNet

torchexpo.vision.image_classification.**resnext101_32x8d**()
ResNext-101 32x8d Model pre-trained on ImageNet

6.1.11 Wide ResNet

torchexpo.vision.image_classification.**wide_resnet50_2**()
Wide ResNet-50-2 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**wide_resnet101_2**()
Wide ResNet-101-2 Model pre-trained on ImageNet

6.1.12 MNASNet

torchexpo.vision.image_classification.**mnasnet0_5**()
MNASNet 0.5 Model pre-trained on ImageNet

torchexpo.vision.image_classification.**mnasnet1_0**()
MNASNet 1.0 Model pre-trained on ImageNet

6.2 Image Segmentation

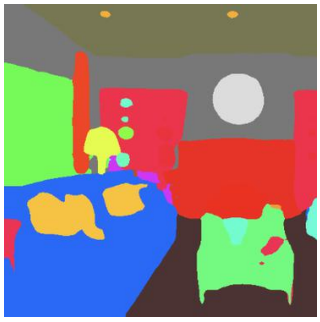

Image Segmentation (or Semantic Segmentation) is the task of clustering parts of an image together which belong to the same object class. It is a form of pixel-level prediction because each pixel in an image is classified according to a category. Some example benchmarks for this task are Cityscapes, PASCAL VOC and ADE20K. Models are usually evaluated with the Mean Intersection-Over-Union (Mean IoU) and Pixel Accuracy metrics.

Example:

```
>>> from torchexpo.vision import image_segmentation
>>>
>>> model = image_segmentation.fcn_resnet50()
>>> model.extract_torchscript()
>>> model.extract_onnx()
```

6.2.1 FCN-ResNet

torchexpo.vision.image_segmentation.**fcn_resnet50**()
FCN-ResNet50 Model pre-trained on COCO train2017

torchexpo.vision.image_segmentation.**fcn_resnet101**()
FCN-ResNet101 Model pre-trained on COCO train2017

6.2.2 DeepLabV3-ResNet

torchexpo.vision.image_segmentation.**deeplabv3_resnet50**()
DeepLabV3-ResNet50 Model pre-trained on COCO train2017

torchexpo.vision.image_segmentation.**deeplabv3_resnet101**()
DeepLabV3-ResNet101 Model pre-trained on COCO train2017

CHAPTER

SEVEN

TORCHEXPO.NLP

7.1 Sentiment Analysis

Sentiment analysis is the task of classifying the polarity of a given text.

Example:

```
>>> from torchexpo.nlp import sentiment_analysis
>>>
>>> model = sentiment_analysis.electra_imdb()
>>> model.extract_torchscript()
>>> model.extract_onnx()
```

7.1.1 DistilBERT (IMDb)

torchexpo.nlp.sentiment_analysis.**distilbert_imdb**()
DistilBERT Model pre-trained on IMDb

7.1.2 ELECTRA (IMDb)

torchexpo.nlp.sentiment_analysis.**electra_imdb**()
Electra Model pre-trained on IMDb

7.2 Text Classification (Coming Soon)

Coming Soon

7.3 Question Answering (Coming Soon)

Coming Soon

CHAPTER

EIGHT

PERSONS OF INTEREST

8.1 Project Maintainers

- Omkar Prabhu (prabhuomkar)

8.2 Module-level Maintainers

8.2.1 modules.*

- Omkar Prabhu (prabhuomkar)

8.2.2 vision.*

Looking for core developers

8.2.3 nlp.*

- Omkar Prabhu (prabhuomkar)

Looking for core developers

PYTHON MODULE INDEX

t

torchexpo.modules, 11
torchexpo.nlp, 19
torchexpo.nlp.sentiment_analysis, 19
torchexpo.vision, 13
torchexpo.vision.image_classification, 13
torchexpo.vision.image_segmentation, 16

INDEX

A

alexnet() (*in module torchexpo.vision.image_classification*), 13

D

deeplabv3_resnet101() (*in module torchexpo.vision.image_segmentation*), 17
deeplabv3_resnet50() (*in module torchexpo.vision.image_segmentation*), 17
densenet121() (*in module torchexpo.vision.image_classification*), 15
densenet161() (*in module torchexpo.vision.image_classification*), 15
densenet169() (*in module torchexpo.vision.image_classification*), 15
densenet201() (*in module torchexpo.vision.image_classification*), 15
distilbert_imdb() (*in module torchexpo.nlp.sentiment_analysis*), 19

E

electra_imdb() (*in module torchexpo.nlp.sentiment_analysis*), 20
extract_caffe2_mobile() (*torchexpo.modules.TorchExpoModule method*), 12
extract_onnx() (*torchexpo.modules.ImageClassificationModule method*), 11
extract_onnx() (*torchexpo.modules.ImageSegmentationModule method*), 11
extract_onnx() (*torchexpo.modules.SentimentAnalysisModule method*), 11
extract_onnx() (*torchexpo.modules.TorchExpoModule method*), 12
extract_torchscript() (*torchexpo.modules.ImageClassificationModule method*), 11
extract_torchscript() (*torchexpo.modules.ImageSegmentationModule method*), 11
extract_torchscript() (*torchexpo.modules.SentimentAnalysisModule method*), 12
extract_torchscript() (*torchexpo.modules.TorchExpoModule method*), 12

F

fcn_resnet101() (*in module torchexpo.vision.image_segmentation*), 17
fcn_resnet50() (*in module torchexpo.vision.image_segmentation*), 17

G

get_extracted_file_name() (*torchexpo.modules.TorchExpoModule method*), 12
googlenet() (*in module torchexpo.vision.image_classification*), 15

I

ImageClassificationModule (*class in torchexpo.modules*), 11
ImageSegmentationModule (*class in torchexpo.modules*), 11
inceptionv3() (*in module torchexpo.vision.image_classification*), 15

M

mnasnet0_5() (*in module torchexpo.vision.image_classification*), 16
mnasnet1_0() (*in module torchexpo.vision.image_classification*), 16
mobilenet_v2() (*in module torchexpo.vision.image_classification*), 15
module
    torchexpo.modules, 11
    torchexpo.nlp, 19
    torchexpo.nlp.sentiment_analysis, 19

torchexpo.vision, 13
torchexpo.vision.image_classification, 13
torchexpo.vision.image_segmentation, 16

P

print_message() (*torchexpo.modules.TorchExpoModule method*), 12

R

resnet101() (*in module torchexpo.vision.image_classification*), 14
resnet152() (*in module torchexpo.vision.image_classification*), 14
resnet18() (*in module torchexpo.vision.image_classification*), 14
resnet34() (*in module torchexpo.vision.image_classification*), 14
resnet50() (*in module torchexpo.vision.image_classification*), 14
resnext101_32x8d() (*in module torchexpo.vision.image_classification*), 15
resnext50_32x4d() (*in module torchexpo.vision.image_classification*), 15

S

SentimentAnalysisModule (*class in torchexpo.modules*), 11
shufflenet_v2_x0_5() (*in module torchexpo.vision.image_classification*), 15
shufflenet_v2_x1_0() (*in module torchexpo.vision.image_classification*), 15
squeezenet1_0() (*in module torchexpo.vision.image_classification*), 14
squeezenet1_1() (*in module torchexpo.vision.image_classification*), 14

T

torchexpo.modules
module, 11
torchexpo.nlp
module, 19
torchexpo.nlp.sentiment_analysis
module, 19
torchexpo.vision
module, 13
torchexpo.vision.image_classification
module, 13
torchexpo.vision.image_segmentation
module, 16
TorchExpoModule (*class in torchexpo.modules*), 12

V

vgg11() (*in module torchexpo.vision.image_classification*), 14
vgg11_bn() (*in module torchexpo.vision.image_classification*), 14
vgg13() (*in module torchexpo.vision.image_classification*), 14
vgg13_bn() (*in module torchexpo.vision.image_classification*), 14
vgg16() (*in module torchexpo.vision.image_classification*), 14
vgg16_bn() (*in module torchexpo.vision.image_classification*), 14
vgg19() (*in module torchexpo.vision.image_classification*), 14
vgg19_bn() (*in module torchexpo.vision.image_classification*), 14

W

wide_resnet101_2() (*in module torchexpo.vision.image_classification*), 16
wide_resnet50_2() (*in module torchexpo.vision.image_classification*), 16