GluonCV and GluonNLP: Deep Learning in Computer Vision and Natural Language Processing

Jian Guo  
University of Michigan MI, USA

He He  
Tong He  
Leonard Lausen  
Mu Li  
Haibin Lin  
Xingjian Shi  
Chenguang Wang  
Junyuan Xie  
Sheng Zha  
Aston Zhang  
Hang Zhang  
Zhi Zhang  
Zhongyue Zhang  
Shuai Zheng  
Yi Zhu  
Amazon Web Services, CA, USA

Abstract

We present GluonCV and GluonNLP, the deep learning toolkits for computer vision and natural language processing based on Apache MXNet (incubating). These toolkits provide state-of-the-art pre-trained models, training scripts, and training logs, to facilitate rapid prototyping and promote reproducible research. We also provide modular APIs with flexible building blocks to enable efficient customization. Leveraging the MXNet ecosystem, the deep learning models in GluonCV and GluonNLP can be deployed onto a variety of platforms with different programming languages. The Apache 2.0 license has been adopted by GluonCV and GluonNLP to allow for software distribution, modification, and usage.

Keywords: Machine Learning, Deep Learning, Apache MXNet, Computer Vision, Natural Language Processing

1. Introduction

Deep learning, a sub-field of machine learning research, has driven the rapid progress in artificial intelligence research, leading to astonishing breakthroughs on long-standing problems in a plethora of fields such as computer vision and natural language processing. Tools powered by deep learning are changing the way movies are made (Knorr et al., 2018), dis-

* Mu Li is the corresponding author.


License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/ Attribution requirements are provided at http://jmlr.org/papers/v21/19-429.html.
Such development is made possible by deep learning frameworks, such as Caffe (Jia et al., 2014), Chainer (Tokui et al., 2015), CNTK (Seide and Agarwal, 2016), Apache (incubating) MXNet (Chen et al., 2015), PyTorch (Paszke et al., 2017), TensorFlow (Abadi et al., 2016), and Theano (Bastien et al., 2012). These frameworks have been crucial in disseminating ideas in the field. Specifically, imperative tools, arguably spearheaded by Chainer, are easy to learn, read, and debug. Such benefits make imperative programming interface quickly adopted by the Gluon API of MXNet (though it can be seamlessly switched to symbolic programming for high performance), PyTorch, and TensorFlow Eager.

Leveraging the imperative Gluon API of MXNet, we design and develop the GluonCV and GluonNLP (referred to as GluonCV/NLP hereinafter) toolkits for deep learning in computer vision and natural language processing. GluonCV/NLP simultaneously i) provide modular APIs to allow customization by re-using efficient building blocks; ii) provide pre-trained state-of-the-art models, training scripts, and training logs to enable fast prototyping and promote reproducible research; iii) provide models that can be deployed in a wide variety of programming languages including C++, Clojure, Java, Julia, Perl, Python, R, and Scala (via the MXNet ecosystem).

Sharing the same MXNet backend and enjoying its benefits such as multiple language bindings, which are not offered by other frameworks, the frontend of GluonCV/NLP are separated to allow users of either computer vision or natural language processing to download and install either GluonCV or GluonNLP of a smaller size than that of their combination.

2. Design and Features

In the following, we describe the design and features of GluonCV/NLP.

2.1. Modular APIs

GluonCV/NLP provide access to modular APIs to allow users to customize their model design, training, and inference by re-using efficient components across different models. Such common components include (but are not limited to) data processing utilities, models with individual components, initialization methods, and loss functions.

To elucidate how the modular API facilitates efficient implementation, let us take the data API of GluonCV/NLP as an example, which is used to build efficient data pipelines with popular benchmark data sets or those supplied by users. In computer vision and natural language processing tasks, inputs or labels often come in with different shapes, such as images with a varying number of objects and sentences of different lengths. Thus, the data API provides a collection of utilities to sample inputs or labels then transform them into mini-batches to be efficiently computed.

The following code snippet shows an example of using fixed bucketing to sample mini-batches more efficiently. In the first assignment statement, the batchify utilities provided by the data API specifies that inputs of different shapes will be padded to have the same length then be stacked as a mini-batch. The following FixedBucketSampler class of the data API will group inputs of more similar shapes into the same mini-batch so more computation is saved due to less padding. In the end, we pass the aforementioned specifications,
together with the actual data set \texttt{train\_data}, via arguments of the \texttt{DataLoader} class of the Gluon API of MXNet, so it will return efficient mini-batches of \texttt{train\_data}.

```python
import gluon, gluonnlp
batchify_fn = gluonnlp.data.batchify.Tuple(
    gluonnlp.data.batchify.Pad(), gluonnlp.data.batchify.Stack())
train_sampler = gluonnlp.data.FixedBucketSampler(
    lengths=train_data.transform(lambda x: len(x[0])),
    batch_size=batch_size, shuffle=True)
train_iter = gluon.data.DataLoader(
    train_data, batchify_fn=batchify_fn, batch_sampler=train_sampler)
```

Besides, users can access a wide range of popular data sets via the \texttt{data} API, including (but are not limited to) ImageNet of image classification, VOC of object detection, COCO of instance segmentation, SST of sentiment analysis, IWSLT of machine translation, SQuAD of question answering, and WikiText of language modeling. The code snippet below shows that users can access training sets of IWSLT2015 English-Vietnamese and SQuAD 2.0, and the test set of WikiText103 with just one line of code via the \texttt{data} API.

```python
import gluonnlp
iwslt15 = gluonnlp.data.IWSLT2015('train', src_lang='en', tgt_lang='vi')
squad = gluonnlp.data.SQuAD('train', '2.0')
wikitext103 = gluonnlp.data.WikiText103('test')
```

### 2.2. Model Zoo

Building upon those modular APIs, GluonCV/NLP provide pre-trained state-of-the-art models, training scripts, and training logs via the model zoo to enable fast prototyping and promote reproducible research. As of the time of writing, GluonCV/NLP have provided over 200 models for common computer vision and natural language processing tasks, such as image classification, object detection, semantic segmentation, instance segmentation, pose estimation, video action recognition, word embedding, language model, machine translation, sentiment analysis, natural language inference, dependency parsing, and question answering. Figure 1 visualizes inference throughputs (on 1 NVIDIA TESLA V100) vs. validation accuracy of ImageNet pre-trained models on image classification.

### 2.3. Leveraging the MXNet Ecosystem

GluonCV/NLP have benefitted from the MXNet ecosystem through use of MXNet. At the lowest level, MXNet provides high-performance C++ implementations of operators that are
leveraged by GluonCV/NLP; thus, improvements in low-level components of MXNet often result in performance gains in GluonCV/NLP. Same as any other model implemented with MXNet, GluonCV/NLP can be used to train models on CPU, GPU (single or multiple), and multiple machines. In sharp contrast to building upon other deep learning frameworks, through the unique hybridizing mechanism by MXNet (Zhang et al., 2020), usually GluonCV/NLP models can be deployed with no or minimal configuration in a wide spectrum of programming languages including C++, Clojure, Java, Julia, Perl, Python, R, and Scala. There are also ongoing efforts to bring more quantization (int8 and float16 inference) benefits from MXNet to GluonCV/NLP to further accelerate model inference, e.g., comparing with the float32 inference, our sentence classification result on the MRPC dataset shows that the int8 inference reduces latency of the BERT\textsubscript{BASE} model by 59.6% on Intel CLX-8280.

The documentation https://gluon-cv.mxnet.io/ and http://gluon-nlp.mxnet.io/ of GluonCV/NLP include installation instructions, contribution instructions, open source repositories, extensive API reference, and comprehensive tutorials. As another benefit of leveraging the MXNet ecosystem, the GluonCV/NLP documentation is supplemented by the interactive open source book *Dive into Deep Learning* (Zhang et al., 2020), which provides sufficient background knowledge about GluonCV/NLP tasks, models, and building blocks. Notably, some users of *Dive into Deep Learning* have later become contributors of GluonCV/NLP.

### 2.4. Requirement, Availability, and Community

GluonCV/NLP are implemented in Python and are available for systems running Linux, macOS, and Windows since Python is platform agnostic. The minimum and open source package (e.g., MXNet) requirements are specified in the documentation. As of the time of writing, GluonCV/NLP have reached version 0.6 and 0.8 respectively, and have been open sourced under the Apache 2.0 license. The rapid growth in the vibrant open source community has spurred active development of new features in the toolkits. Each pull request (code contribution) will trigger the continuous integration server to run all the test cases in the code repositories. All the changes based on community feedback can be found through the merged pull requests. For example, the open-source community has enriched the data API by contributing more utilities, and added implementations, training scripts, or training logs of a variety of models, such as textCNN and CycleGAN.

### 3. Performance

We demonstrate the performance of GluonCV/NLP models in various computer vision and natural language processing tasks. Specifically, we evaluate popular or state-of-the-art models on standard benchmark data sets. In the experiments, we compare model performance between GluonCV/NLP and other open source implementations with Caffe, Caffe2, Theano, and TensorFlow, including ResNet (He et al., 2016) and MobileNet (Howard et al., 2017) for image classification (ImageNet), Faster R-CNN (Girshick, 2015) for object detection (COCO), Mask R-CNN (He et al., 2017) for instance segmentation, Simple Pose (Xiao et al., 2018) for pose estimation (COCO), Inflated 3D networks (I3D) (Carreira and Zisserman, 2017) for video action recognition, textCNN (Kim, 2014) for sentiment analysis (TREC), and BERT (Devlin et al., 2018) for question answering (SQuAD 1.1), sentiment analysis
Table 1: Comparison of model performance (in percentage) on the validation data sets between GluonCV/NLP and other open source implementations (OOSI) across popular computer vision and natural language processing tasks and data sets.

<table>
<thead>
<tr>
<th>Task</th>
<th>Data set</th>
<th>Model</th>
<th>Measure</th>
<th>GluonCV/NLP</th>
<th>OOSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Classification</td>
<td>ImageNet</td>
<td>ResNet-50</td>
<td>top-1 acc.</td>
<td>79.2</td>
<td>75.3[a]</td>
</tr>
<tr>
<td>Image Classification</td>
<td>ImageNet</td>
<td>ResNet-101</td>
<td>top-1 acc.</td>
<td>80.5</td>
<td>76.4[a]</td>
</tr>
<tr>
<td>Image Classification</td>
<td>ImageNet</td>
<td>MobileNet 1.0</td>
<td>top-1 acc.</td>
<td>73.3</td>
<td>70.9[b]</td>
</tr>
<tr>
<td>Object Detection</td>
<td>COCO</td>
<td>Faster R-CNN</td>
<td>mAP</td>
<td>40.1</td>
<td>39.6[c]</td>
</tr>
<tr>
<td>Instance Segmentation</td>
<td>COCO</td>
<td>Mask R-CNN</td>
<td>mask AP</td>
<td>33.1</td>
<td>32.8[c]</td>
</tr>
<tr>
<td>Pose Estimation</td>
<td>COCO</td>
<td>Simple Pose (f)</td>
<td>OKS AP</td>
<td>74.2</td>
<td>N.A.</td>
</tr>
<tr>
<td>Action Recognition</td>
<td>Kinetics400</td>
<td>3D ResNet-50</td>
<td>top-1 acc.</td>
<td>74.0</td>
<td>72.9[f]</td>
</tr>
<tr>
<td>Sentiment Analysis</td>
<td>TREC</td>
<td>textCNN</td>
<td>acc.</td>
<td>92.8</td>
<td>92.2[e]</td>
</tr>
<tr>
<td>Question Answering</td>
<td>SQuAD 1.1</td>
<td>BERT_BASE</td>
<td>F1/EM</td>
<td>88.5/81.0</td>
<td>88.5/80.8[c]</td>
</tr>
<tr>
<td>Natural Language Inference</td>
<td>MNLI-m</td>
<td>BERT_BASE</td>
<td>acc.</td>
<td>84.6</td>
<td>84.4[e]</td>
</tr>
<tr>
<td>Paraphrasing</td>
<td>MRPC</td>
<td>BERT_BASE</td>
<td>acc.</td>
<td>88.7</td>
<td>86.7[e]</td>
</tr>
</tbody>
</table>

[a] https://github.com/KaimingHe/deep-residual-networks (in Caffe)
[c] https://github.com/facebookresearch/Detectron (in Caffe2)
[e] https://github.com/google-research/bert (in TensorFlow)
[f] https://github.com/open-mmlab/mmaction (in PyTorch)

(SST-2), natural language inference (MNLI-m), and paraphrasing (MRPC). Table 1 shows that the GluonCV/GluonNLP implementation matches or outperforms the compared open source implementation for the same model evaluated on the same data set. In some cases, such as image classification with ResNet-50 on the ImageNet data set, our implementation has significantly outperformed implementations in other frameworks. We highlight that the outperformance can be attributed to “minor” refinements such as in data augmentations and optimization methods in the training procedure (He et al., 2019).

4. Conclusion

GluonCV/NLP provide modular APIs and the model zoo to allow users to rapidly try out new ideas or develop downstream applications in computer vision and natural language processing. GluonCV/NLP are in active development and our future works include further enriching the API and the model zoo, accelerating model inference, improving compatibility with the NumPy interface, and supporting deployment in more scenarios.

Acknowledgments

We would like to thank all the contributors of GluonCV and GluonNLP (the git log command can be used to list all the contributors). Specifically, we thank Xiaoting He, Heewon Jeon, Kangjian Wu, and Luyu Xia for providing part of results in Table 1. We would also like to thank the entire MXNet community for their foundational contributions.
References


