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

Jian Guo GJIAN@UMICH.EDU
University of Michigan MI, USA

He He HEHEA@AMAZON.COM
Tong He HTONG@AMAZON.COM
Leonard Lausen LAUSEN@AMAZON.COM
***Mu Li** MLI@AMAZON.COM
Haibin Lin HAIBILIN@AMAZON.COM
Xingjian Shi XJSHI@AMAZON.COM
Chenguang Wang CHGWANG@AMAZON.COM
Junyuan Xie ERIC.JY.XIE@GMAIL.COM
Sheng Zha ZHASHENG@AMAZON.COM
Aston Zhang ASTONZ@AMAZON.COM
Hang Zhang HZAWS@AMAZON.COM
Zhi Zhang ZHIZ@AMAZON.COM
Zhongyue Zhang ZHONGYUE@AMAZON.COM
Shuai Zheng SHZHENG@AMAZON.COM
Yi Zhu YZAWS@AMAZON.COM
Amazon Web Services, CA, USA

Editor: Antti Honkela

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.

eases are diagnosed (Lipton et al., 2015), and play a growing role in understanding and communicating with humans (Zhu et al., 2018).

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 langauge 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 train_data, via arguments of the DataLoader class of the Gluon API of MXNet, so it will return efficient mini-batches of train_data.

```
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 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 data API.

```
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.

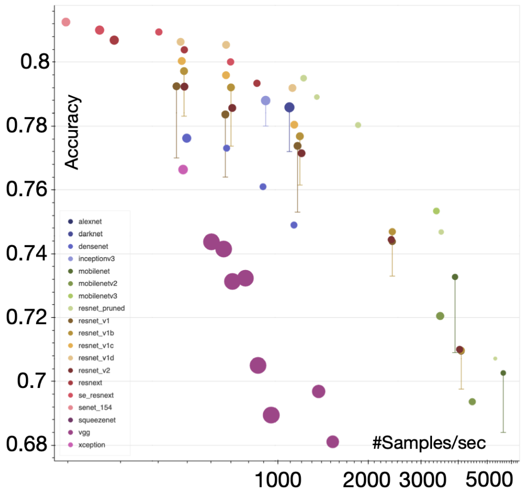

Figure 1: GluonCV's inference throughputs vs. validation accuracy. Circle Area is proportional to device memory requirement.

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 $\text{BERT}_{\text{BASE}}$ 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.

Task	Data set	Model	Measure	GluonCV/NLP	OOSI
Image Classification	ImageNet	ResNet-50	top-1 acc.	79.2	75.3[a]
Image Classification	ImageNet	ResNet-101	top-1 acc.	80.5	76.4[a]
Image Classification	ImageNet	MobileNet 1.0	top-1 acc.	73.3	70.9[b]
Object Detection	COCO	Faster R-CNN	mAP	40.1	39.6[c]
Instance Segmentation	COCO	Mask R-CNN	mask AP	33.1	32.8[c]
Pose Estimation	COCO	Simple Pose (f)	OKS AP	74.2	N.A.
Action Recognition	Kinetics400	I3D ResNet-50	top-1 acc.	74.0	72.9[f]
Sentiment Analysis	TREC	textCNN	acc.	92.8	92.2[e]
Sentiment Analysis	SST-2	BERT$_{\text{BASE}}$	acc.	93.0	92.7[e]
Question Answering	SQuAD 1.1	BERT$_{\text{BASE}}$	F1/EM	88.5/81.0	88.5/80.8[e]
Question Answering	SQuAD 1.1	BERT$_{\text{LARGE}}$	F1/EM	91.0/84.1	90.9/84.1[e]
Natural Language Inference	MNLI-m	BERT$_{\text{BASE}}$	acc.	84.6	84.4[e]
Paraphrasing	MRPC	BERT$_{\text{BASE}}$	acc.	88.7	86.7[e]

[a] `https://github.com/KaimingHe/deep-residual-networks` (in Caffe)
[b] `https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md` (in TensorFlow)
[c] `https://github.com/facebookresearch/Detectron` (in Caffe2)
[d] `https://github.com/yoonkim/CNN_sentence` (in Theano)
[e] `https://github.com/google-research/bert` (in TensorFlow)
[f] `https://github.com/open-mmlab/mmaction` (in PyTorch)

(SST-2), natural langauge 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

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6299–6308, 2017.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Sebastian Knorr, Matis Hudon, Julian Cabrera, Thomas Sikora, and Aljosa Smolic. Deepstereobrush: Interactive depth map creation. In *2018 International Conference on 3D Immersion (IC3D)*, pages 1–8. IEEE, 2018.

Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Frank Seide and Amit Agarwal. CNTK: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135. ACM, 2016.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, volume 5, pages 1–6, 2015.

Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 466–481, 2018.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2020. `http://www.d2l.ai`.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. SDNet: Contextualized attention-based deep network for conversational question answering. *arXiv preprint arXiv:1812.03593*, 2018.