We wish to thank our reviewers for their insightful feedback that helped us improve the clarity and overall quality of our work. We have revised the paper as suggested by the reviewers. The corresponding changes are:

- As mentioned by reviewer #1, the hardware requirements of PyTorch were not clearly explained. We therefore updated section 5.1 of our paper to state that PyTorch can run in a CPU only environment but will happily take advantage of GPUs if such devices are available.
- To better support our performance claims, we compared the speed of PyTorch against that of 3 additional libraries (namely Chainer, CNTK and PaddlePaddle) as requested by reviewer #1. We included in our new comparison matrix the tools that Amazon or NVidia support officially. In essence, we relied on these companies to perform a market research study on our behalf and identify the best solutions currently available. However, we excluded Caffe and Theano from the final list since they are no longer actively maintained. We also excluded Keras since it is essentially a frontend running on top of TensorFlow, CNTK, or Theano. We reproduced the updated version of Table 1 below. We believe that these additional results confirm that the performance of PyTorch is indeed very competitive.
- Reviewer #1 wondered about attribution. Since more than 1000 people have contributed to PyTorch, we can't list every single individual. Instead we've highlighted the people who had a profound impact on the library and have significantly helped shape its development. We've also acknowledged the impact of the community as a whole in section 8.
- We fixed the typographical error line 87 pointed out by reviewer #1.
- We realized that didn't cite the "Automatic differentiation in PyTorch" work from Paszke, Gross, Chintala, Chanan, Yang, DeVito, Lin, Desmaison, Antiga and Lerer submitted at the 2017 NIPS autodiff workshop. We added the missing reference in section 4.3.
- Reviewer #2 mentioned that the last paragraph of Section 5.2 is confusing. We agree, and rewrote the paragraph to emphasize that we keep the execution of tensor operators on CPU synchronous since, unlike on GPU, the benefits of an asynchronous execution model are overshadowed by the overhead of cross-thread communication and synchronization.
- We rewrote section 5.4 to explicitly state that the PyTorch multiprocessing module builds upon the native Python multiprocessing library. We emphasized that while the PyTorch version of the module optimizes the transfer of large tensors between processes, it leverages the Python implementation whenever possible.
- We updated Listing 1 to refer to the 'nn.functional' module instead of 'F' to increase the clarity of the code as suggested by reviewer #2
- We reordered the list of programming languages to match the order in which array libraries are listed line 37-38 to improve clarity as suggested by reviewer #3.

Framework	*Throughput (higher is better)*					
	AlexNet	VGG-19	ResNet-50	MobileNet	GNMTv2	NCF
Chainer	778 ± 15	N/A	$\mathbf{219} \pm 1$	N/A	N/A	N/A
CNTK	845 ± 8	84 ± 3	210 ± 1	N/A	N/A	N/A
MXNet	$\mathbf{1554} \pm 22$	113 ± 1	218 ± 2	444 ± 2	N/A	N/A
PaddlePaddle	933 ± 123	112 ± 2	192 ± 4	$\mathbf{557} \pm 24$	N/A	N/A
TensorFlow	1422 ± 27	66 ± 2	200 ± 1	216 ± 15	$9631 \pm 1.3\%$	$4.8e6 \pm 2.9\%$
PyTorch	1547 ± 316	$\mathbf{119} \pm 1$	212 ± 2	463 ± 17	$\mathbf{15512} \pm 4.8\%$	$\mathbf{5.4e6} \pm 3.4\%$

Table 1: Training speed for 6 models using 32bit floats. Throughput is measured in images per second for the AlexNet, VGG-19, ResNet-50, and MobileNet models, in tokens per second for the GNMTv2 model, and in samples per second for the NCF model. The fastest speed for each model is shown in bold.