Item 291
**git_comments:**

**git_commits:**

1. **summary:** Tensor Inspector Tutorial (#15517)
   **message:** Tensor Inspector Tutorial (#15517) * add tensor inspector tutorial * link docs * link docs * add license * Revert "add license" This reverts commit 32881e5acff6a0dc833e52f9c82c4967df40f006. * Revert "link docs" This reverts commit f93ae219262513e8ce52c0d68abd8eb3f40b2ed5. * Revert "link docs" This reverts commit 160b8912e14746f2de1b660c6b24a24197ffcb46. * Revert "add tensor inspector tutorial" This reverts commit 3b53981a4fe932e8ae80e4ea1ab5cd0260a12574. * add tensor inspector doc * fix api name * add new test and limitations section * fix * update urls and other fixes * fix urls * fix

**github_issues:**

**github_issues_comments:**

**github_pulls:**

1. **title:** Tensor Inspector Tutorial
   **body:** ## Description ## Tutorial for this PR: https://github.com/apache/incubator-mxnet/pull/15490 Need images uploaded here: https://github.com/dmlc/web-data/pull/194 ## Checklist ## ### Essentials ### Please feel free to remove inapplicable items for your PR. - [ ] The PR title starts with [MXNET-$JIRA_ID], where $JIRA_ID refers to the relevant [JIRA issue] (https://issues.apache.org/jira/projects/MXNET/issues) created (except PRs with tiny changes) - [ ] Changes are complete (i.e. I finished coding on this PR) - [ ] All changes have test coverage: - Unit tests are added for small changes to verify correctness (e.g. adding a new operator) - Nightly tests are added for complicated/long-running ones (e.g. changing distributed kvstore) - Build tests will be added for build configuration changes (e.g. adding a new build option with NCCL) - [ ] Code is well-documented: - For user-facing API changes, API doc string has been updated. - For new C++ functions in header files, their functionalities and arguments are documented. - For new examples, README.md is added to explain the what the example does, the source of the dataset, expected performance on test set and reference to the original paper if applicable - Check the API doc at http://mxnet-ci-doc.s3-accelerate.dualstack.amazonaws.com/PR-$PR_ID/$BUILD_ID/index.html - [ ] To the my best knowledge, examples are either not affected by this change, or have been fixed to be compatible with this change ### Changes ### - [ ] Feature1, tests, (and when applicable, API doc) - [ ] Feature2, tests, (and when applicable, API doc) ## Comments ## - If this change is a backward incompatible change, why must this change be made. - Interesting edge cases to note here

**github_pulls_comments:**

1. @ChaiBapchya Thanks for the thoughtful review!
2. @mxnet-label-bot add [Operator, NDArray, Visualization, pr-work-in-progress]
3. @access2rohit @sandeep-krishnamurthy The tutorial is ready for review. After applying your suggestions I will replace the image url to the ones I uploaded to dmlc/web-data so don't worry about image names or urls for now
4. @aaronmarkham - Can you please help review? Thanks!
5. @ChaiBapchya @IvyBazan Thanks for the very detailed revision suggestions! I've made the changes locally, and after my images are ready, I will change the image url's in the doc as well. Very looking forward to having it posted so that developers can start using!
6. Haha you guys are BREATHTAKING
7. Thanks @Zha0q1 @IvyBazan @ChaiBapchya @larroy @apeforest @marcoabreu

**github_pulls_reviews:**

1. ```suggestion This utility is located in `src/common/tensor_inspector.h`. To use it in any operator code, just include `tensor_inspector`, construct an `TensorInspector` object, and call the APIs on that object. You can run any script that uses the operator you just modified then. ```

2. ```suggestion When developing new operators, developers need to deal with tensor objects extensively. This new utility, Tensor Inspector, mainly aims to help developers debug by providing unified interfaces to print, check, and dump the tensor value. To developers' convenience, this utility works for all the three data types: Tensors, TBlobs, and NDArrays. Also, it supports both CPU and GPU tensors. ```
3. Would be great if you could give the image a recognizable name
4. ```suggestion Essentially, `TensorInspector` can be understood as a wrapper class around `TBlob`. Internally, the `Tensor`, `Tbob`, or `NDArray` object that you passed in will be converted to a `TBlob` object. The `RunContext` object is used when the the tensor is a GPU tensor; in such a case, we need to use the context information to copy the data from GPU memory to CPU/main memory. ```
5. likewise
6. ```suggestion This API will create a file with name "{tag}_{visit_count}.npy", where tag is the name that we give to the call, and visit is the visit count, should the operator be called more than once. ```
7. ```suggestion Notice: in `interactive_print()`, you could also do value dumping with command "d". You will be prompted to enter the `tag` value: ```
8. object -> objects
9. JASON -> JSON?
10. Haha you r right!
11. it's "the object", referring to the same object that the user has passed as described in line 36
12. Also please host it somewhere properly
13. Proper host
14. Proper host
15. Proper host
16. Yeah, I have them uploaded in another PR. Will update the url after taking in the comments
17. Yeah, I have them uploaded in another PR. Will update the url after taking in the comments
18. ok
19. Duplicate "the". Corrected sentence: "The `RunContext` object is used when the tensor is a GPU tensor"
20. "Following are the three constructors:"
21. Suggested edit for clarity and flow: "You can find some useful information about the tensor on the last line of the output."
22. Edit for clarity: "When debugging, situations might occur at compilation time, so you do not know which part of a tensor to inspect. Sometimes, it may be nice to pause the operator control flow to “zoom into” a specific, erroneous part of a tensor multiple times until you are satisfied."
23. "This API will set a "break point" in your code. When used, you will enter a loop that will keep asking you for further command input."
24. "A visit count will tell you how many times you stepped into this particular "break point", should this operator be called more than once."
25. "There are many useful commands available, as described in the previous screenshot"
26. "Or, you might do that simply because a binary dump has better precision and is faster to load than output copy-pasted from `print_string()` and loaded as a `JASON` string."
27. "Let's see how it runs:"
28. nitpick: ```suggestion Let's see how it runs: ```
29. Any command for the same? How to include it?
30. nitpick: ```suggestion To print out the tensor value in a nicely structured way, you can use this API: ```
31. ```suggestion NegativeChecker, // check if negative ``` or ```suggestion NegativeChecker, // check if it is negative ```
32. ```suggestion PositiveChecker, // check if positive ``` or ```suggestion PositiveChecker, // check if it is positive ``` So on and so forth for the rest

**jira_issues:**

**jira_issues_comments:**