Galaxy Jupyterlab for AI

Anup Kumar

Freiburg Galaxy team, Bioinformatics group, University of Freiburg, Germany
Jupyterlab

- Jupyter notebooks - popular editor
  - Data science
  - Scientific computing
  - Machine learning
  - Learn to code. E.g Python
- Simple and fast way to create prototypes
- No need for any package installation
- Easy to share an analysis
- Run on web
Interactive Galaxy tool
Remote training jobs
Remote training jobs
Interactive Galaxy tool
Features

- Faster computations using GPU
- Ready to use, pre-installed packages
  - ML: Scikit-learn, Tensorflow, CUDA, OpenCV, ONNX AI models
  - Data manipulation: Pandas, H5py, NumPy, Scipy, Nibabel, …
  - Visualizations: Matplotlib, Seaborn
- Git integration
- Workflows of notebooks (Elyra AI)
- Communicate with Galaxy (Bioblend)
- Remote training (using a separate Galaxy tool)
- Miscellaneous - resource dashboards, collapsible headers …

Base container from Jupyter Docker stacks: jupyter/tensorflow-notebook:tensorflow-2.6.0
### Comparison with other notebook infrastructures

<table>
<thead>
<tr>
<th></th>
<th>Google Colab</th>
<th>Kaggle Kernel</th>
<th>Galaxy Jupyterlab</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory/disk space</strong></td>
<td>~ 12 GB/70 GB</td>
<td>~ 16 GB/73 GB</td>
<td>~ 20 GB/1 TB</td>
</tr>
<tr>
<td><strong>GPU/TPU</strong></td>
<td>Yes/Yes</td>
<td>Yes/Yes</td>
<td>Yes/No</td>
</tr>
<tr>
<td><strong>Max usage time</strong></td>
<td>12 hrs</td>
<td>12 hrs/session, 30 hrs of GPU/week, 20 hrs of TPU/week</td>
<td>No time restriction on GPU usage, notebook and job execution</td>
</tr>
<tr>
<td><strong>Dynamic resources</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Fixed and guaranteed</td>
</tr>
<tr>
<td><strong>Remote model training</strong></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Use-case 1: COVID-19 CT scan image segmentation

- Reproduce results from published work: "COVID TV-Unet: Segmenting COVID-19 chest CT images using connectivity imposed Unet" [1]
- Entire analysis in Galaxy Jupyterlab notebooks
- Save model as ONNX
- Fetch trained model (as ONNX file) from Galaxy and make predictions
- Remote model training or in notebooks using GPU
- For remote training: convert datasets to H5 (save as matrices)
1. **Script to run**

```python
def dice_loss(y_true, y_pred):
    return 1 - dice_coef(y_true, y_pred)

def custom_loss(y_true, y_pred):
    layer_names = [layer.name for layer in model.layers]
    for l in layer_names:
        if l == layer_names[-1]:
            value = TV_bin_loss(y_true, y_pred)
        else:
            value = binary_crossentropy(K.flatten(y_true), K.flatten(y_pred))
    return value

combined_data = hSpy.File("h5_datasets/combined_CT_datasets.h5", "r")
X_train = np.array(combined_data["X_train")
X_valid = np.array(combined_data["X_valid")
y_train = np.array(combined_data["y_train")
y_valid = np.array(combined_data["y_valid")
```

2. **Send script to run remotely**

```python
import os
API_KEY = os.environ.get('API_KEY', None)
GALAXY_URL = os.environ.get('GALAXY_URL', None)

script_path = "4_create_model_and_train_remote.ipynb"
data_list = ["h5_datasets/combined_CT_datasets.h5"]
tool_output = run_script_job(script_path, data_dict=data_list, server=GALAXY_URL, key=API_KEY, new_history_name="CT_segmentation_march_18")
```

3. **History**

- **Epoch 1/10**
- 1/41 [........................] - ETA: 27:08 - loss: 1.0769 - accuracy: 0.1384 - dice_loss: 0.5996 - recall_1: 0.3317 - pre_1: 0.2842
- Compressed zip file
- 5: Zipped files
- 131.9 MB
- format: zip, database: ?
Use-case 2: Predict protein 3D structures using Colabfold

- Colabfold: Predict 3D structures of proteins using only sequences [2]
- Less memory intensive than Alphafold2, faster prediction
- Use Alphafold2 weights
- Addition of only two packages in Galaxy Jupyterlab - colabfold and JAX
- Acceleration of the prediction of 3D conformation via GPU
- Notebook
- Next slide: Predicted 3D structure of 300 amino-acid long spike protein of SARS-CoV-2
Welcome to the Galaxy’s GPU enabled Interactive Jupyterlab for Artificial intelligence (AI).

Jupyter notebook is powered by the latest JupyterLab, Tensorflow, Scikit-learn, Pandas, Numpy, Scipy, Seaborn, Matplotlib and many more which can be used to prototype and develop machine learning and deep learning solutions executing on Galaxy’s NVIDIA GPUs. The docker container used can be found at docker image.

Core features

- Run AI programs on GPUs
- Pre-installed packages for AI
- Integrated with Git version control
- Elyra AI workflow of notebooks
- Shareable AI models via Open Neural Network Exchange (ONNX)
- Run Galaxy tools using BioBlend APIs

Introduction

Jupyterlab notebooks are extremely popular with data scientists and researchers to explore datasets from multiple fields of studies and develop prototypes. The notebooks come integrated with a lots of packages such as NumPy, Statsmodel, Pandas, Scikit-learn, Tensorflow, Matplotlib which expedite prototyping and provide useful insights into the datasets. In a notebook, each rectangular box is known as a cell which executes Python code written in it.
Running instance, GTN tutorial.

- Running instance
- GTN tutorial
- Preprint
- Dockerfile
- Submitted to GigaScience and is under review
- Thanks to Gianmauro, Bjoern and Rolf
Thank you for your attention!

Questions?
References

2. Colabfold: https://www.nature.com/articles/s41592-022-01488-1
3. Docker image: https://github.com/anuprulez/ml-jupyter-notebook
4. https://www.docker.com/
6. https://jupyter.org/
7. https://www.tensorflow.org/
10. https://github.com/
11. https://pandas.pydata.org/
12. https://keras.io/examples/nlp/text_classification_with_transformer/