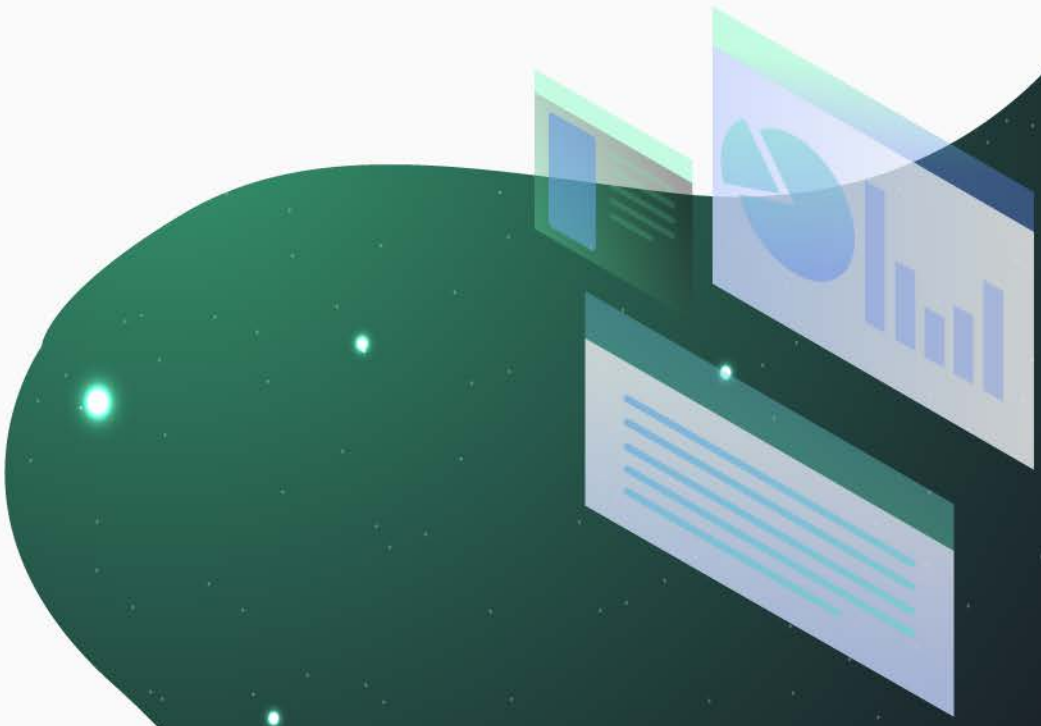
PerceptiLabs

Technical Overview



PerceptiLabs

Technical Overview of PerceptiLabs

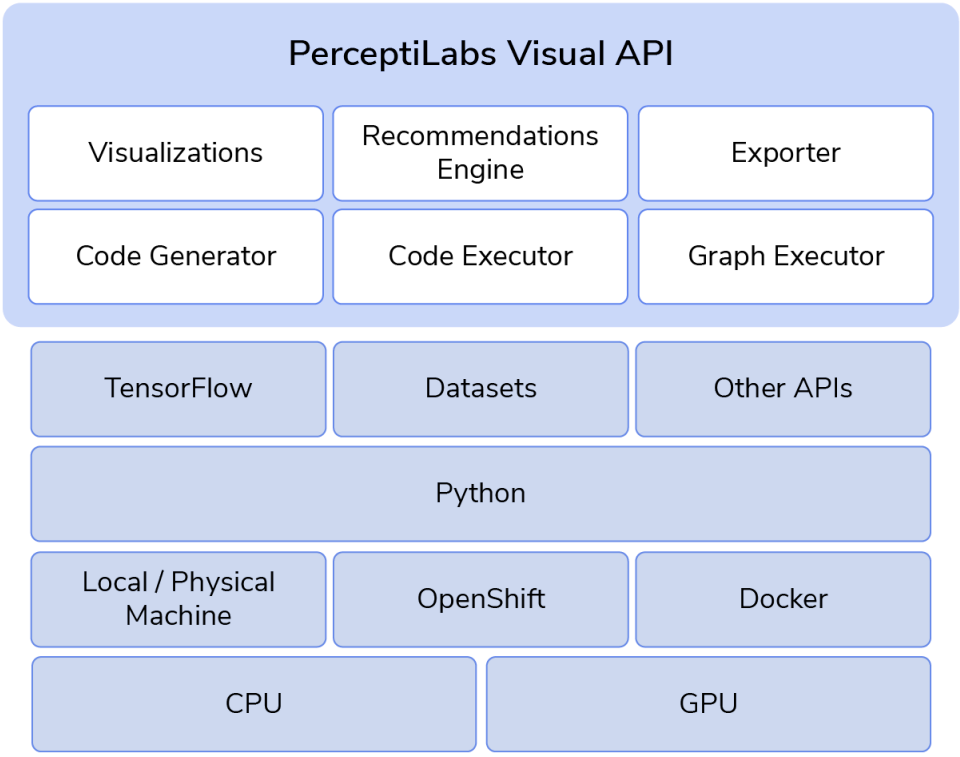
Abstract

PerceptiLabs enables machine learning practitioners to work more efficiently with machine learning models and to gain more insight into their models. PerceptiLabs' visual approach lowers the barrier of entry for beginners while providing researchers and advanced users with code-level access to their models.

Technology

PerceptiLabs is a dataflow driven, visual API for TensorFlow, distributed as a free Python package (hosted on PyPI) for everyone to use. PerceptiLabs wraps low-level TensorFlow code to create visual components, which allows users to visualize the model architecture as the model is being built.

As a visual API, PerceptiLabs sits on top of the various TensorFlow API layers including tf.Keras:



How it works

The settings/hyperparameters of each component (layer) can be set and tuned with the visual interface. Since these high-level settings generate low-level code, we can provide lots of support to the user,

which is the key behind the benefits PerceptiLabs provides. This allows us to auto-generate granular visualizations for each component and to suggest settings (auto-configs) for the user in each component. PerceptiLabs gives the user warnings, errors, and tips during the modeling process to guide them towards building better models. When training starts, PerceptiLabs auto-generates visualizations for every single underlying variable in the model, which can then be seen and analyzed in a statistics view. When the training is complete, the user can automatically test and validate the model, before exporting the model (i.e., push to production or share on GitHub).

Design Principles

Each component acts as a template for generating low-level TensorFlow code based on the settings, the user can work in a code view as well. We have made this easy by allowing the user to toggle between the visual interface and the code. By keeping this level of transparency and flexibility, we hope to bridge the gap between beginners and researchers, where we have designed the UX in a similar way to Keras. Instead of using a high-level code API as Keras, PerceptiLabs is a visual API where each component can be configured at both a high level (i.e., via the UI) and at a low level (e.g., via code) right away.

PerceptiLabs is designed so that the user drags and drops components onto a workspace for each layer they want to include in their model. To complete and run the model, a Training component is connected at the end of the model's graph. It's designed in a similar way to Keras, where the user writes one-liners of code for each layer they want their model to include, and to wrap up and train the model, a .compile() and a .fit() method are invoked. However, the Training

component in PerceptiLabs comes with the benefit of making it easier to build complex models and to use different machine learning techniques in an easy way. Due to the Training component's unique design, PerceptiLabs supports any sort of novel model types and techniques. For example, if the user wants to use reinforcement learning or object detection, they will connect the respective components at the end of the model.

The Python code being generated in each component is fetched from Jinja files, where our goal is to have the community contributing to these templates through our GitHub repository.

Comparison to TensorFlow and Keras

While PerceptiLabs is built on TensorFlow and other APIs, PerceptiLabs' visual API provides immediate benefits over pure code:

While users can work directly with the raw code for each component in their model, most users will enjoy the ability to configure each

component via the user interface and will see an immediate preview of what the component outputs.

Key Benefits

There are several benefits of taking a visual approach as it enables us to:

- Show the model architecture with output visualizations for each component
- Show granular visualizations during the modeling phase, run-time, and testing
- Show warnings and recommendations for debugging and model building
- Automatically suggest configs/settings and hyperparameters

On top of that, PerceptiLabs also provides:

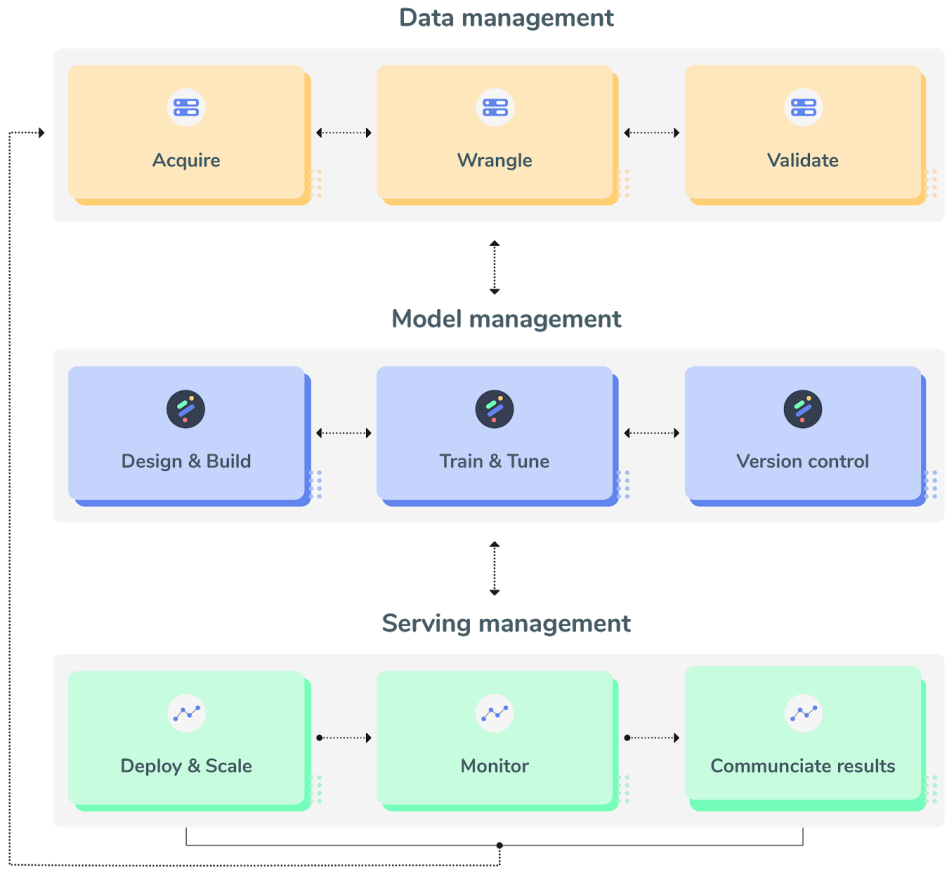
- Model templates for common machine learning problems
- Dimensions and I/O shape fitting
- A model registry to easily keep track of models and experiments
- Data and model version control in order to reproduce experiments and go back in time
- Distributed training over all available GPUs
- Different tests to try out the model before pushing it to production

Integration/Workflow/Architecture

The process for building machine learning models involves numerous steps.

A machine learning workflow usually begins with a “Data Management” phase, where training data is acquired, organized into a useful form, and may be labeled (e.g., for supervised learning) so that it can be classified by an ML system. Following this, a “Model Management” phase takes place, where an ML model is built, evaluated and tuned for behaviour and performance, and a version of the trained model is stored and made ready for deployment. Finally, “Serving Management” involves the deployment of that trained model into the real world, where it’s made available for inference, continually monitored, and its results are communicated to the business owner and back to the ML team.

Machine Learning workflow



PerceptiLabs helps with the "Model management" phase while providing the flexibility for users to choose Data management and Serving management solutions that work best for them.

In our free version, users host their data and model on their local machine, and then share them via GitHub. Models exported by the free version of PerceptiLabs can then be hosted either locally or on a server.

For enterprise, users can choose between our Docker version or our Red Hat OpenShift version. The Docker version can distribute processing across multiple GPUs on a single machine, while the Red Hat OpenShift version can distribute processing on multiple GPUs across several machines, and offers an integrated, high-performance platform on which to piece together components for designing and hosting models.

PerceptiLabs' visual interface helps a wide range of users from developers to project managers, collaborate during the “Design and Build” phase, while its rich statistical view helps users “Train & Tune” their model. PerceptiLabs also offers features for "Version Control" such as the ability to view the history of a model.