model-zoo Documentation

Release 0.0.1

Lambda Labs

Aug 23, 2018

Contents

Lambda Model Zoo is a curate of easy-to-use, high performance, open-source machine learning solutions:

- We understand machine learning boilplate can be boring to write and hard to maintain. For this reason we've isoloated the core algrithms from the rest of the code so you can focus on designing the best machine leanring models.
- We understand keeping up with the rapidly-evolving machine learning frameworks and APIs can be difficult. So we've designed very carefully a set of resuable, maintainable modules as the building blocks for machine learning solutions. You can create novel solutions by simply composing these building blocks.
- We believe transfer learning is the key to efficient research and development. That is why we focus on models that have stood the test of time and make it extremely easy to apply these models to your own problems.

CHAPTER 1

Tutorials

1.1 Getting Started

- *Create a machine learning application*
- *Run the applicaiton*

1.1.1 Create a machine learning application

Model zoo creates machine learning solutions by composing reusable building blocks:

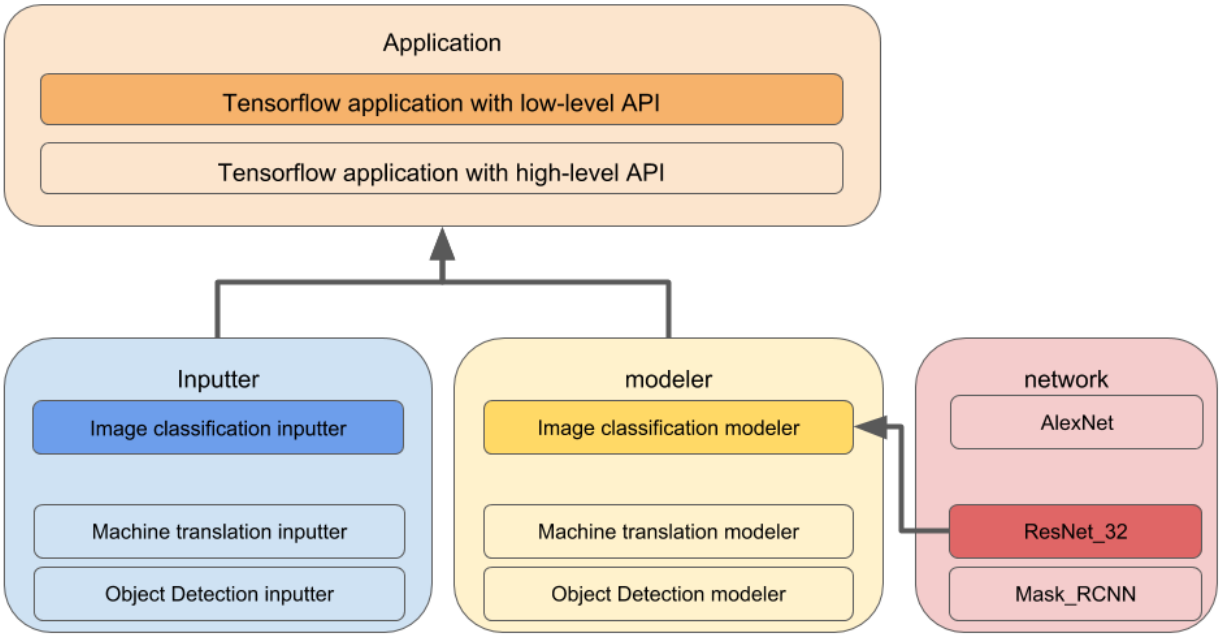
- **Modeler**: The model pipeline. It encapsulates the forward pass, the computation of loss, the evaluation metric. For reusability, the forward pass is a placeholder that can be dynamically configured to different network architectures. This allows a modeler to be applicable to a general problem. For example, we have image_classification_modeler, machine translation_modeler, object_detection_modeler ... etc.
- **Inputter**: The data pipeline. It reads data from the disk, shuffles and preprocesses the data, creates batches, and does prefetch. Like the modeler, an inputter is applicable to a general problem. For example, we have image_classification_inputter, machine_translation_inputter, object_detection_inputter ... etc.
- **Application**: The excuter. It orchestrates the excution of an inputter and an modeler, distributes the workload across multiple hardware devices, logs the statistics of the job and saves the trained model to disk.

The value of having the above building blocks is they can be pre-built and re-used in many tasks – an image_classification_modeler is meant to work with all image classification tasks. The same applies to the inputter. Application is even more general – it is applicable to many different problems and only varies by the selection of framework and level of APIs.

Now, we can add the core algorithm and complete a machine learning solution:

- **network**: Creates a particular network architecture, such as AlexNet, VGG19, ResNet32 ... etc. It completes the solution by replacing the modeler's placeholder for forward pass.

Fig. 1.1.1 illustrates the composition of a machine learning application.



1.1.2 Run the applicaiton

blah, blah

1.2 Overview

1.3 Applications

1.4 Write your own

CHAPTER 2

Documentation

2.1 API documentation