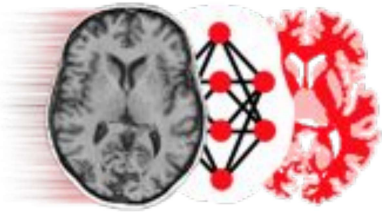
NiftyNet Tutorial
-- for Developers



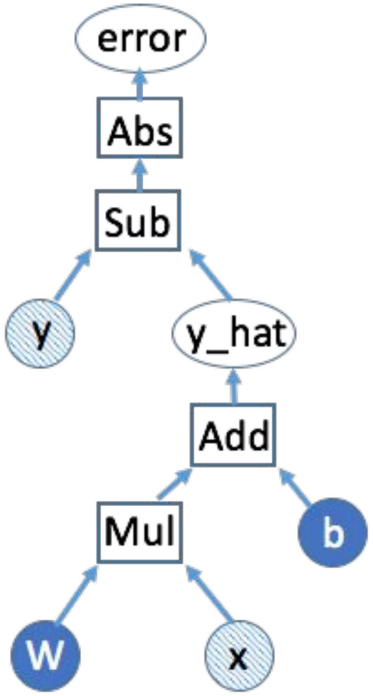
Initial document 18-Oct-2017

Let’s start from training a linear model with TensorFlow APIs...

A TensorFlow minimal working example

A linear model

$$\hat{y} \approx Wx + b$$




```python
import tensorflow as tf

demo_graph = tf.Graph()
with demo_graph.as_default():
    x = tf.placeholder(tf.float32)
    y = tf.placeholder(tf.float32)

    W = tf.Variable([2.0], dtype=tf.float32)
    b = tf.Variable([-4.0], dtype=tf.float32)

    y_hat = W * x + b
    error = tf.abs(y_hat - y)

with tf.Session(graph=demo_graph) as sess:
    sess.run(tf.global_variables_initializer())
    [y_pred, loss] = sess.run(
        [y_hat, error], feed_dict={x: 10.0, y: 15.0})
    # y_pred [16.], loss [1.]
```

Deep learning application

- Inputs (observations)
  - Load image volumes
  - Augmentation
  - Sample
- Network model
  - Params. management
  - Layer operations
  - Loss functions
- Model sharing
- Outputs aggregation

```python
import tensorflow as tf

demo_graph = tf.Graph()
with demo_graph.as_default():
    x = tf.placeholder(tf.float32)
    y = tf.placeholder(tf.float32)

    W = tf.Variable([2.0], dtype=tf.float32)
    b = tf.Variable([-4.0], dtype=tf.float32)

    y_hat = W * x + b
    error = tf.abs(y_hat - y)

with tf.Session(graph=demo_graph) as sess:
    sess.run(tf.global_variables_initializer())
    [y_pred, loss] = sess.run(
        [y_hat, error],
        feed_dict={x: 10.0, y: 15.0})
    # y_pred [16.], loss [1.]
```

Let’s take a look at NiftyNet APIs...

Application interfaces

Deep learning application

- Inputs (observations)
  - Load image volumes
  - Augmentation
  - Sample
- Network model
  - Params. management
  - Layer operations
  - Loss functions
- Model sharing
- Outputs aggregation

```
Class BaseApplication(object):

    # Input data
    def initialise_dataset_loader()

    def initialise_sampler()

    # Network model (and sharing)
    def initialise_network()

    def connect_data_and_network()

    # Outputs aggregation
    def interpret_output()
```

An application demo

```
Class DemoApplication(BaseApplication):
    def initialise_dataset_loader():
        self.reader = ImageReader(...)

    def initialise_sampler():
        self.sampler = UniformSampler(reader=self.reader, ...)

    def initialise_network():
        self.net = ApplicationNetFactory.create('toynet')(...)

    def connect_data_and_network(variable_collector):
        window = self.sampler.pop_batch_op()
        net_output = self.net(window['image'])
        if is_training:
            loss_layer = LossFunction(net_output, window['label'])
            gradients = compute_gradients(loss_layer(net_output))
        variable_collector.add(net_output)

    def interpret_output(output_numpy_array):
        self.decoder(output_numpy_array)
```

```
# In application engine
For each iteration:
    numpy_output = tf.Session.run(
        variable_collector.variables())
```

Commands for `pip install` user

```
pip install NiftyNet
net_run train -a apps.demo_app.DemoApplication \
              -c my_config.ini
```

Command for `git clone` user

```
cd NiftyNet/
python net_run.py train -a apps.demo_app.DemoApplication \
                        -c my_config.ini
```

App. file layout

apps/demo_app.py	Class **DemoApplication**(BaseApplication): . . .

```
# Make sure my_folder/apps is a Python module available for `import`
export $PYTHONPATH=$PYTHONPATH:my_folder/apps
```

Configuration file parsing

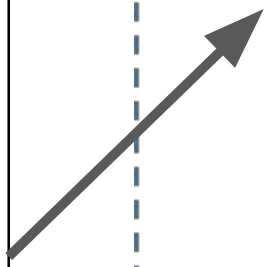
.ini configuration file

```
[SYSTEM]
```
Low-level TF API configurations

```
[NETWORK]
```
Hyperparameters of networks

```
[TRAINING]
```
Describes a training process

```
[INFERENCE]
```
Describes a inference process



```
Class BaseApplication(object):
    self.net_param
    self.action_param

    # Input data
    def initialise_dataset_loader()
    def initialise_sampler()
    # Network model (and sharing)
    def initialise_network()
    def connect_data_and_network()
    # Outputs aggregation
    def interpret_output()
```

Configuration file parsing

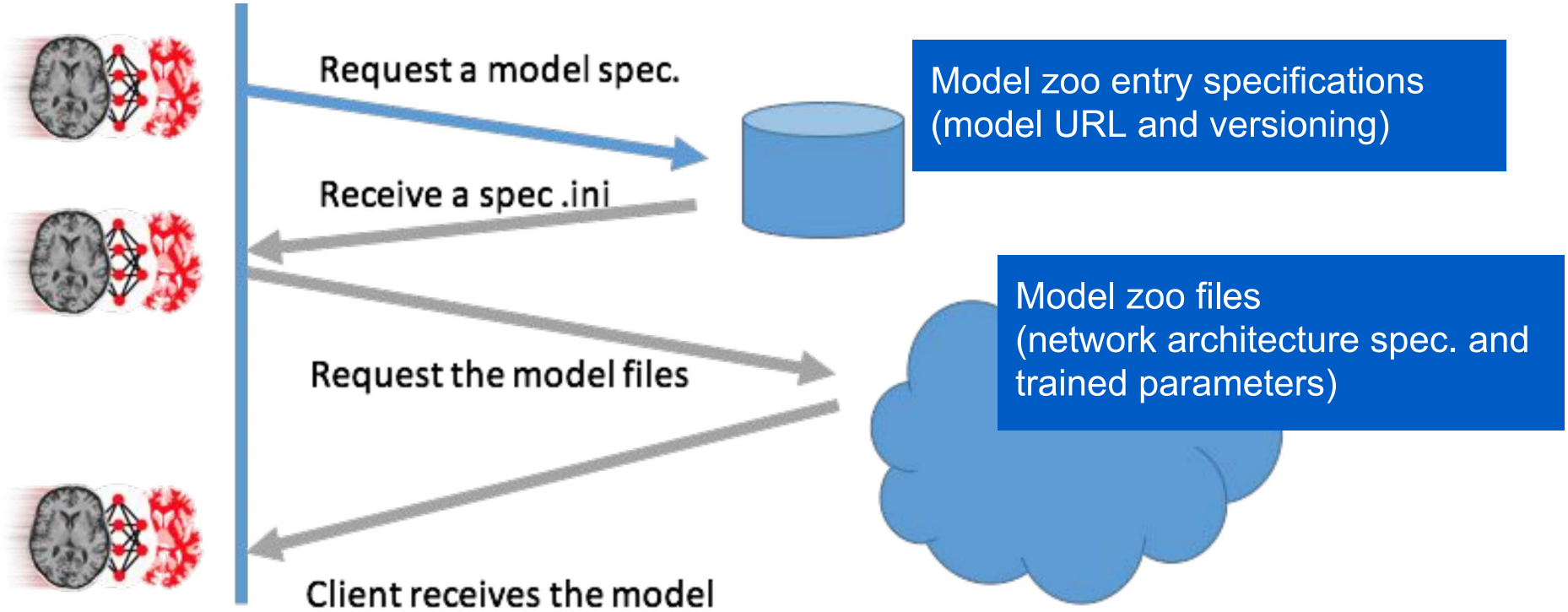
.ini configuration file

```
# Task-specific parameters
[TASK_SECTION]
image = T1MR, T2MR

# Multi-modal input sources
[T1MR]
path_to_search = /folder/T1
filename_contains = T1
[T2MR]
path_to_search = /folder/T2
filename_contains = T2
```

```
Class BaseApplication(object):
    self.task_param
    self.data_param

    # Input data
    def initialise_dataset_loader()
    def initialise_sampler()
    # Network model (and sharing)
    def initialise_network()
    def connect_data_and_network()
    # Outputs aggregation
    def interpret_output()
```

Model zoo

(Under construction as of 23-Nov-2017)




```
pip install NiftyNet
net_download dense_vnet_abdominal_ct_model_zoo
```

For more info...

Commands and configurations:

https://github.com/NifTK/NiftyNet/blob/dev/config/README.md

Demos:

https://github.com/NifTK/NiftyNet/tree/dev/demos

Contributing

https://github.com/NifTK/NiftyNet/blob/dev/CONTRIBUTING.md