This tutorial shows how to create a very basic neural network design and train a model that you can use to classify [MNIST data - hand written digits](http://yann.lecun.com/exdb/mnist/). The data set consists of 60,000 training and 10,000 test examples of grayscale images that measure 28x28 pixels.

Prerequisites

* Prepare an IBM Cloud Object Storage bucket with MNIST data in the format acceptable by the Neural Network Modeler.
  + Use data sets that are stored as pickle files. Each data set file (pickle) consist of the pair (tuple):
  + numpy array with data in a dimention specific for TensorFlow (channels last): [N, 28, 28]
  + labels array in a shape [N]

Prepare the data sets

Prepare the following data sets:

1. training data set
2. test data set
3. validation data set

You can generate the test files by using keras tools:

import keras

from keras.datasets import mnist

import pickle, numpy

from sklearn.model\_selection import train\_test\_split

*# The data, split between train and test sets:*

(X, y), (X\_test, y\_test) = mnist.load\_data()

*# Make additional split between training and validation sets:*

X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(X, y, test\_size=0.166, random\_state=42)

with open('mnist-tf-train.pkl', 'wb') as f:

pickle.dump((X\_train, y\_train), f, protocol=pickle.HIGHEST\_PROTOCOL)

with open('mnist-tf-valid.pkl', 'wb') as f:

pickle.dump((X\_valid, y\_valid), f, protocol=pickle.HIGHEST\_PROTOCOL)

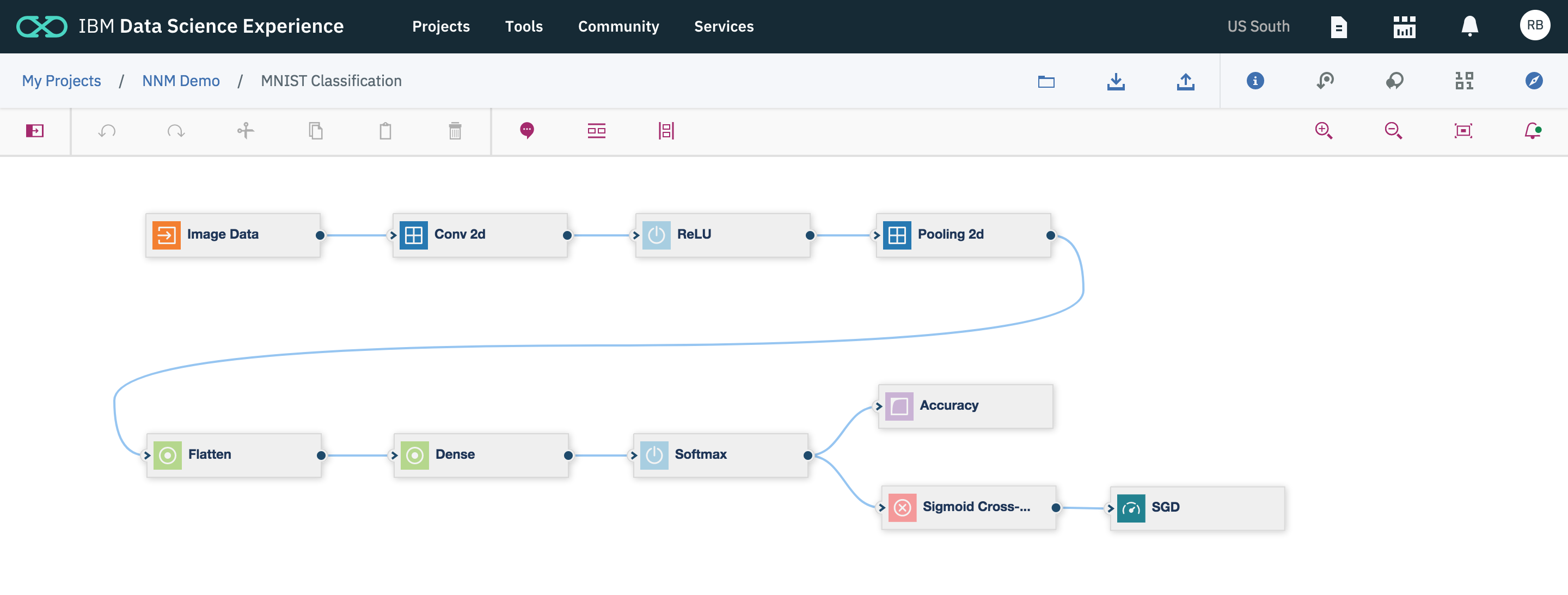
with open('mnist-tf-test.pkl', 'wb') as f:

pickle.dump((X\_test, y\_test), f, protocol=pickle.HIGHEST\_PROTOCOL)

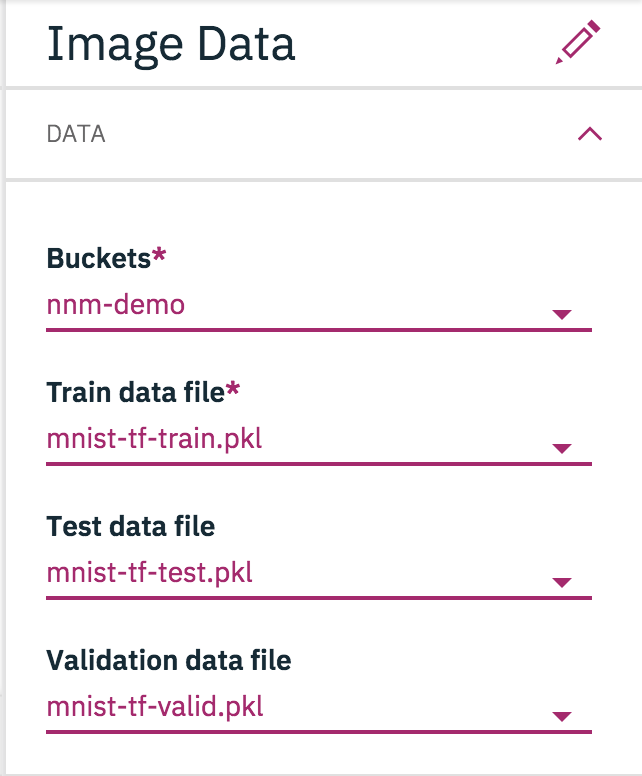
For this example you can copy the following files to your IBM Cloud Object Storage bucket:

* mnist-tf-train.pkl
* mnist-tf-test.pkl
* mnist-tf-valid.pkl

Now, from the Watson Studio new modeler flow window, you can create a new modeler flow from the sample that is named: **Single Convolution layer on MNIST**.



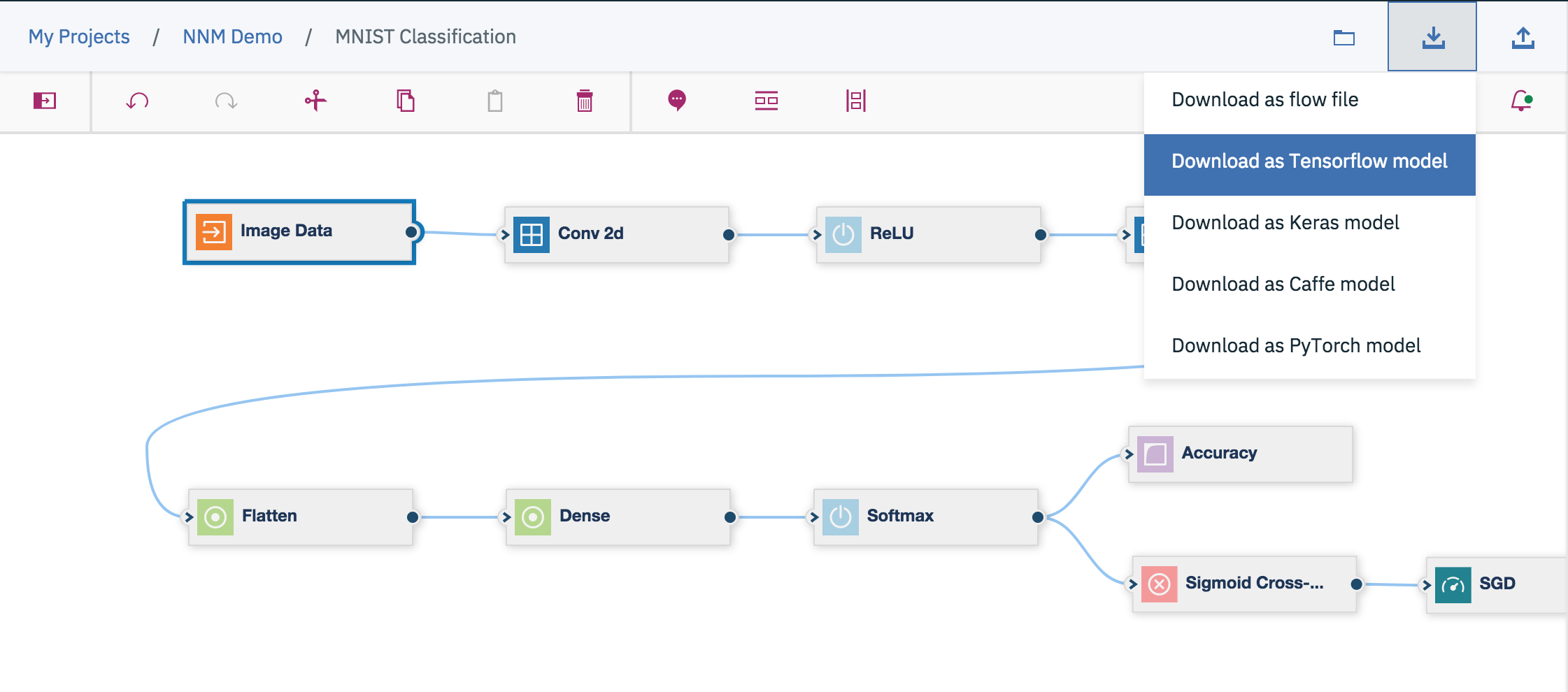
The next step requires pointing the source data. Please click on ImageData node, select your COS bucket and data sets (train, test and validation):



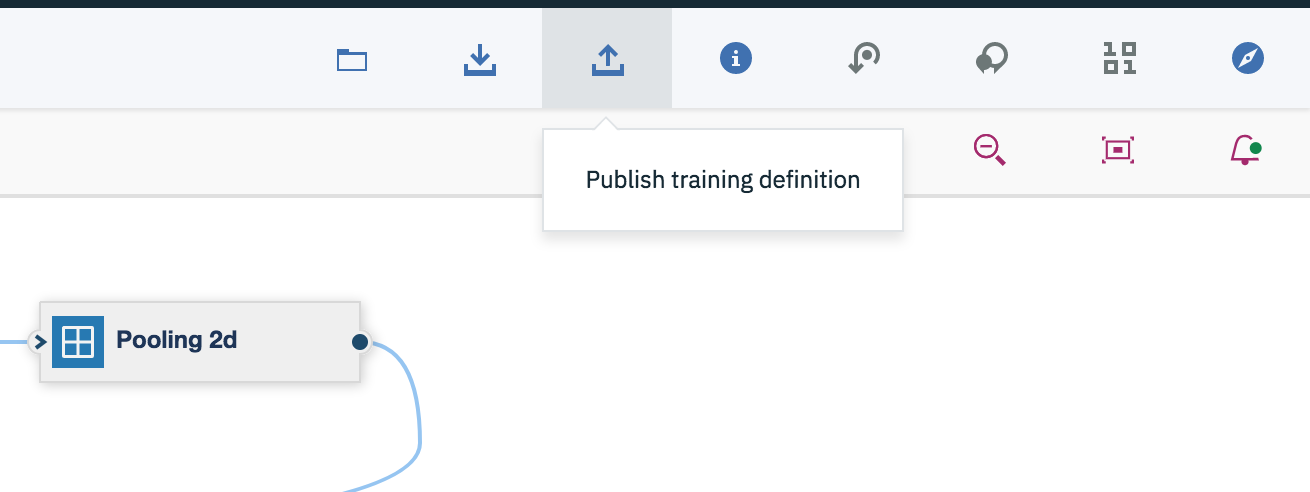
Your first design is ready now. It's indicated by the green dot on the notification icon (right top corner). You can download the source code or start training by publishing training definition.

NNM supports code generation in all famous deep learning frameworks:

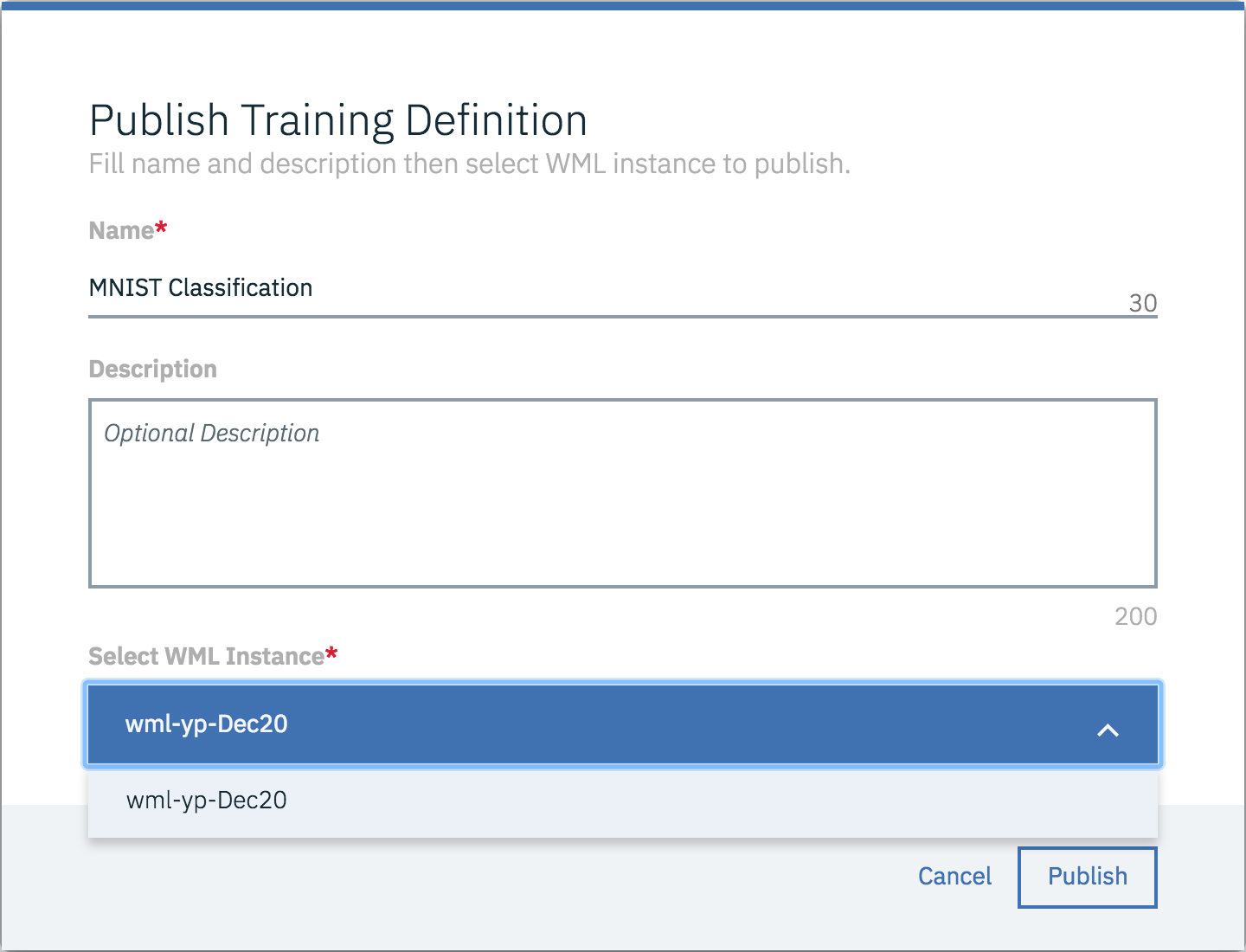
* Tensorflow (1.5) with embedded Keras
* Keras (2.1) with Tensorflow backend
* caffe (1.0)
* pytorch (0.3)



Finally you can publish a training definition and work with it in [Experiment Builder](https://datascience.ibm.com/docs/content/analyze-data/xs-overview.html).



The new training definition is stored is a selected Machine Learning instance's repository with the given name.



This tutorial guides you through using the MNIST computer vision data set to train a TensorFlow model to recognize hand-written digits. In this tutorial, you will train, test, and deploy the model in IBM Watson Studio with experiment builder.

Prerequisite

* [Create a deep learning project in Watson Machine Learning](https://9.30.108.105:9443/docs/content/analyze-data/wml-setup.html)

Steps overview

This tutorial presents the basic steps for training a deep learning model with expriment builder in Watson Studio:

1. [Set up data files in IBM Cloud Object Storage](https://9.30.108.105:9443/docs/content/analyze-data/ml_dlaas_tutorial_tensorflow_experiment-builder.html#step1)
2. [Download sample code](https://9.30.108.105:9443/docs/content/analyze-data/ml_dlaas_tutorial_tensorflow_experiment-builder.html#step2)
3. [Train the model](https://9.30.108.105:9443/docs/content/analyze-data/ml_dlaas_tutorial_tensorflow_experiment-builder.html#step3)
4. [Monitor training progress and results](https://9.30.108.105:9443/docs/content/analyze-data/ml_dlaas_tutorial_tensorflow_experiment-builder.html#step4)
5. [Deploy the trained model](https://9.30.108.105:9443/docs/content/analyze-data/ml_dlaas_tutorial_tensorflow_experiment-builder.html#step5)
6. [Test the deployed model](https://9.30.108.105:9443/docs/content/analyze-data/ml_dlaas_tutorial_tensorflow_experiment-builder.html#step6)

This tutorial does not demonstrate distributed deep learning, or using the Watson Machine Learning hyperparameter optimization feature.

Step 1: Set up data files in Cloud Object Storage

Training a deep learning model using Watson Machine Learning relies on using Cloud Object Storage for reading input (such as training data) as well as for storing results (such as log files.)

1. Download MNIST sample data files to your local computer from here: [MNIST sample files](http://yann.lecun.com/exdb/mnist)  
   Note: Some browsers automatically uncompress the sample data files, which causes errors later in this tutorial. Follow instructions on the MNIST download page for verifying how your browser handled the files.
2. From the **Services** menu in the top menu bar of Watson Studio, choose "Data Services".
3. Select "Manage in IBM Cloud" from the ACTIONS menu beside the service instance of Cloud Object Storage that is associated with your deep learning project. (This opens the service details page for the Cloud Object Storage service instance.)
4. Create two buckets: one for storing training data, and one for storing training results.  
   See: [Creating a Cloud Object Storage bucket](https://9.30.108.105:9443/docs/content/analyze-data/ml_dlaas_object_store.html#createbucket)
5. Upload all of the MNIST sample data files to the training data bucket.  
   See: [Uploading data to Cloud Object Storage](https://9.30.108.105:9443/docs/content/analyze-data/ml_dlaas_object_store.html#uploaddata)

Step 2: Download sample code

Download sample TensorFlow model-building Python code from here: [tf-model.zip](https://github.com/pmservice/wml-sample-models/blob/master/tensorflow/hand-written-digit-recognition/definition/tf-model.zip) .

tf-model.zip contains two files:

1. input\_data.py - A "helper" file for reading the MNIST data files
2. convolutional\_network.py - The model-building code

*Point of interest:* The sample file convolutional\_network.py demonstrates using the environment variable $RESULT\_DIR to cause extra output to be sent to the Cloud Object Storage results bucket:

model\_path = os.environ["RESULT\_DIR"]+"/model"

...

builder = tf.saved\_model.builder.SavedModelBuilder(model\_path)

In this case, the trained model is saved in protobuf format to the results bucket. You could send any output to the results bucket using the $RESULT\_DIRvariable like this.

Step 3: Train the model

In this tutorial, we'll train the model using experiment builder in Watson Studio.

1. From the **Assets** page of your project in Watson Studio, click **New experiment**.
2. Specify a name for the experiment.
3. In the **Machine Learning Service** drop-down, select the Watson Machine Learning service instance that is associated with the project.
4. Configure Cloud Object Storage for the experiment:
   1. In the area for Cloud Object Storage click **Select**.
   2. Click the **New connection** tab.
   3. In the Cloud Object Storage instance drop-down list, select the instance of Cloud Object Storage where you created the training data bucket and the training results bucket.
   4. From the drop-down lists, specify the training data and results buckets that you created before.
   5. Click **Create**
5. Add a training definition.
   1. Click **Add training definition**.
   2. Give the training definition a name.
   3. Upload the sample code, tf-model.zip, where prompted.
   4. Specify the framework that is used in the model-building code: tensorflow 1.5.
   5. In the **Execution command** box, specify this command for running the model-building code:
   6. python3 convolutional\_network.py --trainImagesFile ${DATA\_DIR}/train-images-idx3-ubyte.gz --trainLabelsFile ${DATA\_DIR}/train-labels-idx1-ubyte.gz --testImagesFile ${DATA\_DIR}/t10k-images-idx3-ubyte.gz --testLabelsFile ${DATA\_DIR}/t10k-labels-idx1-ubyte.gz --learningRate 0.001 --trainingIters 20000

*Point of interest:* This sample execution command demonstrates using the environment variable $DATA\_DIR to cause data to be read from the Cloud Object Storage training data bucket.

* 1. Select "1/2 x NVIDIA Tesla K80 (1 GPU)" for the compute plan.
  2. Select "none" for the hyperparameter optimization method.

1. Click **Create and run**.

Step 4: Monitor training progress and results

* You can monitor the progress of a training run in the **Training Runs** tab of experiment builder.
* When the training run is complete, click on the training definition to view details of its training results, including logs and other output.

Step 5: Deploy the trained model

You can't use your trained model to analyze new data until the model has been deployed.

1. In the **Training Runs** tab of experiment builder, under the ACTIONS menu, select "Save model". Then give the model a name and click **Save**. This stores the model in the Watson Machine Learning repository.
2. In the **Assets** page of your project in Watson Studio, click the new model in the **Models** section.
3. Click the **Deploymnts** tab and then click **Add Deployment**.
4. Choose "Web Service" as the deployment type, specify a name for the deployment, and then click **Save**.
5. Click the new deployment to view the model details page.

Step 6: Test the deployed model

You can quickly test your deployed model from the deployment details page.

1. On your local computer, download this sample payload JSON file with input data corresponding to the hand-written digits "5" and "4": [tf-mnist-test-payload.json](https://github.com/pmservice/wml-sample-models/blob/master/tensorflow/hand-written-digit-recognition/test-data" \t "_other)
2. In the **Test** area of the deployment details page in Watson Studio, paste the value of the payload field from tf-mnist-test-payload.json. Then click **Predict**.

Sample output:

{

"values": [

5,

4

]

}

In this output, we can see: the first input data was correctly classified as belonging to the class "5", and the second input data was correctly classified as belonging to the class "4".

