Legal Information

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is a subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications. Current characterized errata are available on request.

Cilk, Intel, the Intel logo, Intel Atom, Intel Core, Intel Inside, Intel NetBurst, Intel SpeedStep, Intel vPro, Intel Xeon Phi, Intel XScale, Itanium, MMX, Pentium, Thunderbolt, Ultrabook, VTune and Xeon are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.

© 2016, Intel Corporation.

Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Description</th>
<th>Revision Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Initial release.</td>
<td>October 2016</td>
</tr>
<tr>
<td>002</td>
<td>Updated for Beta.</td>
<td>December 2016</td>
</tr>
</tbody>
</table>
Contents

1 Introduction ............................................................................................................. 5
  1.1 Release Notes .............................................................................................................................. 5
  1.2 Installing Intel® Deep Learning SDK Training Tool ................................................................. 5
  1.3 Introducing the Intel® Deep Learning SDK Training Tool ........................................................... 6
    1.3.1 Home Tab ................................................................................................... 6
    1.3.2 Uploads Tab ................................................................................................ 7
    1.3.3 Datasets Tab ............................................................................................... 7
    1.3.4 Models Tab ................................................................................................. 7
    1.3.5 Advanced tab .............................................................................................. 8

2 Uploading Images ..................................................................................................... 9

3 Creating a Training Dataset ...................................................................................... 11
  3.1 Adding Source Images to a Dataset .......................................................................................... 12
  3.2 Data augmentation .................................................................................................................. 14
  3.3 Preprocessing Input Image Data .............................................................................................. 16
    3.3.1 Selecting Image Chroma type ...................................................................... 16
    3.3.2 Resizing Images ......................................................................................... 17
  3.4 Setting Database options ......................................................................................................... 18
  3.5 Generating a Dataset ................................................................................................................ 19

4 Saving, Editing and Reviewing a Dataset .................................................................... 20

5 Creating a Model ..................................................................................................... 22
  5.1 Assigning a Dataset .................................................................................................................. 23
  5.2 Configuring Model Topology .................................................................................................... 24
    5.2.1 Creating custom topology ............................................................................ 26
  5.3 Transforming Input Images ...................................................................................................... 27
  5.4 Configuring Training Parameters ............................................................................................. 28
  5.5 Running the Training Process ................................................................................................... 30

6 Saving, Editing and Reviewing a Model ....................................................................... 32

7 Advanced Features .................................................................................................. 39
  7.1 Jupyter Notebook* editor ........................................................................................................ 39
  7.2 Using Sample Routines .......................................................................................................... 40
  7.3 Using Jupyter Notebook* GUI Editor ....................................................................................... 40
  7.4 Using Jupyter Notebook* Terminal Console ............................................................................ 41
  7.5 NNCompressor Tool ................................................................................................................. 42

8 Additional Resources ................................................................................................ 43
1 Introduction

1.1 Release Notes

1.2 Installing Intel® Deep Learning SDK Training Tool
For installation steps please see the Intel® Deep Learning SDK Training Tool Installation Guide.
1.3 Introducing the Intel® Deep Learning SDK Training Tool

The Intel® Deep Learning SDK Training Tool is a feature of the Intel Deep Learning SDK, which is a free set of tools for data scientists, researchers and software developers to develop, train, and deploy deep learning solutions. The Deployment Tool User Guide can be found here.

With the Intel Deep Learning SDK Training Tool, you can:

- Easily prepare training data, design models, and train models with automated experiments and advanced visualizations
- Simplify the installation and usage of popular deep learning frameworks optimized for Intel platforms

The Training Tool is a web application running on a Linux* server or Apple macOS* machine and provides a user-friendly, intuitive interface for building and training deep learning models.

When you start the Training Tool and login, you are presented with a workspace displaying the home page and a series of the main tabs in the blue panel on the left side. These tabs provide access to a set of features that enable you to upload source images, create training datasets, and build and train your deep learning models.

1.3.1 Home Tab

Use the Home tab for quick access to the wizards, recent events and currently active processes.
Quick Links panel lets you upload image files, create datasets and new models or get additional information directly from the Home tab directly.

Active panel tracks the statuses of currently running processes. The panel displays the completeness progress bars and indicators updating in the real-time mode. Clicking on an active process in this panel takes you to an overview page with the process details.

Notifications panel shows the notification messages about processes completeness.

History panel shows the stack of the latest events.

Recent panel lists the recently created objects: uploads, datasets and models. Clicking on a list item takes you to the object overview page.

1.3.2 Uploads Tab

Before you build and train your model, use the Uploads tab to upload an archive of images that will form a dataset for training the model. For details see the Uploading Images topic.

1.3.3 Datasets Tab

Create datasets from previously uploaded images using the Datasets panel. A dataset is not just a bunch of images, but it is a database of a specific format that holds all images arranged in categories.

While creating a dataset you can change image color mode (i.e. grayscale or RGB color), encoding format, or multiply the initial image set by data augmentation, i.e. applying different transformations to original images to create modified versions. All changes are stored in the database and do not physically affect the original files.

You can use the entire image set for training, validation and testing (by assigning a percentage for each subset) or use separate folders for each procedure.

For more information, see the Creating a Training Dataset section.

1.3.4 Models Tab

Use this tab to create and train your model with an existing dataset. There are three available pre-defined topologies you can use for your model as well as the ability to upload a custom topology:

- LeNet
- AlexNet
- GoogLeNet

While configuring the model, you can apply transformations to images from the selected dataset without changing the original image files.

While each of pre-defined models already have a set of default parameters values that are optimal for general use, the Models tab enables you to configure the learning process for specific use cases.

For more information, see the Creating a Model section.
1.3.5 Advanced tab

You can use the Advanced tab to get access to the embedded Jupyter Notebook tool – a web application for creating documents that can contain and run code, visualization and text. This feature allows you to implement your own Python scripts for training deep learning models in the Command Line Interface (CLI) mode.

For more information, see the Advanced Features chapter.
# Uploading Images

Before creating training datasets, you need to upload images you intend to use for training your model. Use the **Uploads** tab to upload input images as an archive with the strictly predefined structure.

All images inside the uploaded archive must be divided into separate subfolders which are named after desired training labels/categories. For example, the structure of a sample archive for 0-9 digits, which could be used for training a LeNet model, may look like the following:

```
digits.rar/
  0/
    0_01.png
    0_02.png
    ...
  1/
    1_01.png
    1_02.png
    ...
  9/
    9_01.png
    9_02.png
    ...
```

Choose the archive located on your computer or on the web and specify the root directory that will hold all extracted images.

The directory path is relative to the Docker installation directory you specified while installing the Training Tool. For the installation steps, see [Installing Intel® Deep Learning SDK Training Tool](#).

The table under the **Upload** button provides the information about completed and currently running uploads:
<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Size</th>
<th>Started</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>hands.rar</td>
<td>/workspace/dfsdk/uploads/hands</td>
<td>194.838 MB</td>
<td>Thu Dec 8 02:39:05 AM</td>
<td>00h 02m 41s</td>
</tr>
<tr>
<td>constraints.zip</td>
<td>/workspace/dfsdk/uploads/tes148113651</td>
<td>13.485 MB</td>
<td>Wed Dec 7 18:31:20 PM</td>
<td>00h 00m 04s</td>
</tr>
<tr>
<td>constraints.zip</td>
<td>/workspace/dfsdk/uploads/tes148113651</td>
<td>13.485 MB</td>
<td>Wed Dec 7 18:31:07 PM</td>
<td>00h 00m 05s</td>
</tr>
<tr>
<td>constraints.zip</td>
<td>/workspace/dfsdk/uploads/tes148113651</td>
<td>13.485 MB</td>
<td>Wed Dec 7 18:30:53 PM</td>
<td>00h 00m 04s</td>
</tr>
<tr>
<td>constraints.zip</td>
<td>/workspace/dfsdk/uploads/tes148113651</td>
<td>13.485 MB</td>
<td>Wed Dec 7 18:30:39 PM</td>
<td>00h 00m 04s</td>
</tr>
<tr>
<td>constraints.zip</td>
<td>/workspace/dfsdk/uploads/tes148113641</td>
<td>13.485 MB</td>
<td>Wed Dec 7 18:28:44 PM</td>
<td>00h 00m 04s</td>
</tr>
<tr>
<td>constraints.zip</td>
<td>/workspace/dfsdk/uploads/tes148113211</td>
<td>13.485 MB</td>
<td>Wed Dec 7 17:16:40 PM</td>
<td>00h 00m 11s</td>
</tr>
</tbody>
</table>
3 Creating a Training Dataset

You can easily create training datasets using the Datasets tab. Once you click the tab, the panel comes up with the New Dataset icon and a list of previously saved datasets:

You can look up saved datasets by searching them via the name, edit, rename, delete them or complete their generation process. For more information, see Saving, Editing and Reviewing a Dataset.

To start creating a dataset, click New Dataset that launches the wizard. A wizard screen contains the following elements:
1. Dataset Name field – Sets the name of the dataset
2. Dataset Description field – Sets the description for the dataset
3. Dataset Manage panel – Enables saving, running or deleting the current dataset at any step
4. Navigation panel – indicates the current step and switches between dataset creation steps.

The wizard divides the workflow of creating training image dataset into three separate steps indicated as tabs on the navigation bar in the wizard screen:

1. Define the folder that contains source images, number of files to use for training and validation and other settings in the **Data folder** tab.
2. Make preprocessing settings in the **Image Preprocessing** tab for input images.
3. Choose the image database options in the **Database option** tab.

Whenever you need to modify the settings you can switch over the steps using the **Next** and **Back** buttons or by clicking a tab on the navigation bar directly.

To abort creating the dataset, click the **Delete** icon in the toolbar in the upper right corner.

### 3.1 Adding Source Images to a Dataset

Start creating a dataset with setting its name and the source folder.
Set the dataset name using the **Dataset Name** field to identify the dataset in the dataset collection. Using meaningful names can help you find the dataset in the list when you are creating a model.

You can add annotations for the dataset if needed using the **Description** field.

Specify an upload to be added to the dataset by using either of two options:

- Use the **Server folder where images were uploaded to** field to specify the path to the root folder that holds contents of the extracted archive that you previously uploaded to the system. If you have not completed this step, use the **New Upload** button to create a new upload, or see the **Uploading Images** section to learn about image archives used for datasets.

- Click the arrow button to choose one of existing uploads from the drop-down list.

From the entire set of training images you can define the image groups for each phase of the model training process:

- Training – trains the model using a set of image samples.
- Validation – could be used for model selection and hyper-parameter fine-tuning.
- Testing – used for the final testing of the model.

To define the validation subset, choose a percentage of images for validation in the **Validation percentage** field. The default value is 10%.
Alternatively, you can also use separate folder for validation. You can specify this folder once you select the **Use other folder** option.

**NOTE:** If you are using other folder for validation, the respective percentage field resets to zero value. Analogously, you can define the testing subset by choosing a percentage in the **Testing percentage** field.

**NOTE:** If you are using other folder for testing, the respective percentage field resets to zero value.

### 3.2 Data augmentation

You can extend your dataset using the Training Tool augmentation feature. It enables you to enlarge the set by creating copies of existing images and applying a number of transformations such as rotating, shifting, zooming and reflecting.
You can simply specify the maximum number of transformations to be applied to each image in the dataset using the **Max number of transformations per image** field.

Alternatively you can use the **Advanced** section to additionally define which types of transformations to apply, transformation parameters and weights. Weight here is the percentage of the selected augmentation type in the total number of performed augmentations, in percent. The higher the specified weight, the more augmentations of the selected type are performed. The total weights of all selected augmentation types must be 100%.

Sometimes transformations result in exposing undefined parts of the image. For example, after zooming out, an image might have blank areas in the border area. Choose a method to fill those blank areas in augmented images using the **Fill Method** section:

- **Constant** - Fills the missing pixels with a certain hexadecimal color code value in RGB format
- **Nearest** - Fills the missing pixels with the values of neighboring pixels
3.3 Preprocessing Input Image Data

You can pre-process images included in a dataset using the Image Processing tab.

3.3.1 Selecting Image Chroma type

The tool enables you to use color or grayscale modes for images. Choose the desired option in the Type group.

**NOTE:** If you select the grayscale option in creating a dataset with color images in RGB format, the tool automatically perform the pixel wise RGB-to-grayscale conversion according to the formula:

\[
Y = 0.299 \times \text{Red} + 0.587 \times \text{Green} + 0.114 \times \text{Blue},
\]

where \( Y \) is the intensity/grayscale value.

**NOTE:** If you use the Color option for grayscale images, the algorithm uses the intensity value as the values of red, green, blue channels: \( R = Y, \ G = Y, \ B = Y \).
3.3.2 Resizing Images

By default, the resize dimensions are set to 28x28 pixels but you can resize the images to arbitrary size with one of the available resize options:

- Squash
- Crop
- Fill
- Half crop, half fill.

The table below demonstrates the results of resizing an example image of original size 128x174 pixels, to a 100x100 pixels square image using each resizing method.

<table>
<thead>
<tr>
<th>Original image, 128x174 pixels</th>
<th>Squash transforms original image by upsampling or downsampling pixels using bi-cubic interpolation to fill new width and length without keeping the aspect ratio.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Squash" /></td>
<td><img src="image2.png" alt="Squash" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Crop option resizes the image while maintaining the aspect ratio. The image is first resized such that the smaller image dimension fits the corresponding target dimension. Then the larger dimension is cropped equal amounts from either sides to fit the corresponding target.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Crop" /></td>
<td><img src="image2.png" alt="Crop" /></td>
</tr>
</tbody>
</table>


### Fill option

The image is first resized such that the larger image dimension fits the corresponding target dimension. Then the resultant image is centered in the smaller dimension and white noise strips of equal width are inserted to both sides to make that dimension equal to the target.

### Half crop, half fill

The image is first resized using the fill option halfway and then the Crop option is applied. For a transformation with the dimensions of the original image and the target image $w$ (width), $h$ (height) and $W$, $H$ respectively, it means that the original image is resized to the dimensions $W+(w-W)/2$, $H+(h-H)/2$ with the Fill transformation first and then the resulting image is resized to the target dimensions $W$, $H$ using the Crop transformation.

#### 3.4 Setting Database options

The Training Tool stores all input images in a database file. At this step you can set desired database options such as database type and image encoding.

You can choose a certain type of the database using the DB backend drop-down list:

- LMDB
- LevelDB

To find more information on both types, please see LMDB and LevelDB home pages.
Choose an image encoding format from the **Image Encoding** drop-down list if needed. Using PNG or JPEG formats can save disk space, but may increase the training time.

### 3.5 Generating a Dataset

After you complete configuring the dataset, you can launch the dataset generation process by clicking the **Run** icon in the dataset manage panel. The Training Tool starts the process and shows the progress bar and the details of the dataset generation process:

Once the generating completes the dataset changes its status in the right tab list to **Complete**.

For more information about saving datasets and dataset statuses, see *Saving, Editing and Reviewing a Dataset*. 
4 Saving, Editing and Reviewing a Dataset

When you click the Dataset tab, the Dataset panel comes up with the New Dataset icon and the list of previously saved datasets. Datasets in the list can be in one of three states: Draft, Ready or Completed.

You can save a dataset as a Draft at any moment before all of its mandatory fields are set. The Ready status indicates that you have set all mandatory fields for the saved dataset and it is ready to be generated. The Completed status identifies already generated datasets.

To find a dataset by its unique name, use the Search field.

You can rename, edit or delete a dataset in the Draft state using the toolbar in the right upper corner.

For a dataset in the Ready state, the Run operation is additionally available.

To view or edit datasets in Draft or Ready state, select it from the list or click the Edit icon in the toolbar.

To view the details of a Completed dataset, select it from the list.

In the Dataset Info tab you can find creation date, physical path to the dataset in the Docker* image and its size.
The **Data folder** tab contains the graphs for training, validation and testing datasets that show number of images in each category.

In the **Image processing** tab you can find information about the images type, their size and crop type. In the **Database options** tab you can see the type of the dataset configuration (LMDB, LevelDB) and the compression type.
5 Creating a Model

After images were uploaded and a dataset was generated using those images, you are ready to create and train a deep learning model. To begin the model creation process, choose the Models tab in the vertical blue panel. The panel comes up with the New Model icon and the list of previously trained and drafted models displayed under the Search text field.

You can look up existing models by searching them by the given model name and rename, edit, build or delete them. For more information, see Saving, Editing and Reviewing a Model.

To create a new model, use the New Model icon. Once you click it, you are presented with a wizard screen.
Wizard screen contains the following elements:

1. Model Name field – A mandatory field that sets the name of the model
2. Model description field – Adds an optional description about the model
3. Model manage panel – Enables saving, running training or deleting the current model at any step
4. Navigation panel – indicates the current step and switches between model creation steps.

The new model creation process consists of four stages as illustrated in the navigation pane.

1. Select a dataset from the list of generated datasets in the Dataset Selection tab.
2. Choose and tune a pre-defined model topology in the Topology tab.
3. Transform images from the dataset if needed in the Data Transformation tab.
4. Configure default parameters to tune the learning process in the Parameters tab.

5.1 Assigning a Dataset

The first stage of creating a model is the dataset selection stage.
As the first step in this stage, enter a unique name for the model in the **Model Name** text field. Using meaningful names can help you find the model in the model list.

In order to quickly recognize the model among other existing models in the future, the Training Tool provides an option of entering a descriptive text about the model in the **Description** field.

Every model should be linked to a particular dataset at any given time. The next step is choosing a dataset that provides the model with training and validation images. Select an existing dataset from the listed datasets or search for one by the name and select it. Press **Next** to move on to the second stage of the process.

### 5.2 Configuring Model Topology

In the second stage you need to configure the topology of the model.
First step is to select a specific model topology from the pre-loaded three topologies listed in the **Topology name** list:

- LeNet
- AlexNet
- GoogLeNet

These pre-loaded topologies come configured with the optimal training/validation settings for that specific topology under general use conditions. However, you can customize it to match specific requirements via checking the **Fine tune topology** check-box. There are two levels of fine tuning available - light and medium - you may pick one as desired. Or you can choose the option not to tune the original weights. Configuration options in the following stages will change upon selecting the fine tuning option and the level of fine tuning chosen.

**Back/Next** buttons in the bottom blue pane allows you to move between the four stages as needed.
5.2.1 Creating custom topology
You can change the structure of a common pre-loaded topology or create a custom one.

In order to create custom topology based on an existing, click the **Edit** button opposite the topology name.

This opens a panel with two fields:
- **View/Modify topology** - Contains the .prototxt file that describes the network topology (for example, see https://github.com/BVLC/caffe/blob/master/examples/mnist/lenet.prototxt)
- **Save as custom topology** – Specifies a name of the new topology

After editing is complete, save the topology and it will appear in the list of custom topologies.
To create a new custom topology from scratch, click the **New custom topology** icon and define the structure in the .prototxt file.

**NOTE:** Note, the tool validates the topology structure you are editing. Only valid topologies can be saved. Now you can use the newly created custom topology for training your model.

**NOTE:** Note, that for custom topologies (as well as for pre-loaded ones) you can use the **Parameters** tab to configure network solver.

**NOTE:** Currently, running testing for the model based on a custom topology is unavailable.

You can also delete a custom topology by clicking the **Delete** button.

### 5.3 Transforming Input Images

The third stage allows you to add pre-processing to the images in before they are fed to the model for training or validation.

You may add three optional pre-processing operations to the training data. Two of them, cropping and horizontal mirroring, add some degree of randomness to the training process, by applying those operations to randomly chosen training images. In image classification tasks with large datasets, these
types of random pre-processing are used to enhance the performance of the learned model by making it robust to deviations of input images which may not be covered in the training set. The mean subtracting, if selected will be done for each and every image, and there are two options: subtract the mean image or pixel.

5.4 Configuring Training Parameters

In the fourth and final stage, training parameters (i.e. hyper-parameters) are configured to tune the training process. Pre-loaded models in the Training Tool come with a set of default values for each of the parameter fields. These values are the optimal parameter values for the given module in its general use case.

Typical training of a deep learning module involves hundreds of thousands of parameters (aka weights), hence a module is trained over-and-over with a given training set. One complete pass of the total
training dataset is called an epoch. At the end of one epoch every image in the training dataset has passed exactly once through the module. You can adjust the number of epochs using the Training epochs field. This number depends on the module topology, parameter estimation algorithm (solver type), initial learning rate and the learning rate decay curve, required final accuracy, and the size of the training dataset.

Within an epoch, images in the training dataset are partitioned into batches and the module is trained with one batch at a time. Once a batch of images passes through the module, the parameters of the module are updated, and then a next batch is used. One such pass is called an iteration. In general, a larger batch size reduces the variance in the parameter update process and may lead to a faster convergence. However, larger the batch size higher the memory usage will be during the training.

By specifying the Validation interval value, you can define how often validations should take place in number of epochs. For example setting the value to 1 will lead to validations taking place at the end of each epoch. Use the Validation batch size value to define the size of the batch of validation images.

Training a deep learning module is a lengthy and complex task, therefore the tool regularly takes snapshots to backup the status of the module being trained and the status of the solver. To set the frequency of backups, use the Snapshot intervals field.

Parameter (or weight) estimation is not only about optimizing the loss/error function for the training datasets the estimated weights should be able to generalize the model to new unseen data. Using the Weight decay setting, you can adjust the regularization term of the model to avoid overfitting.

Learning rate determines the degree an update step influences the current values of the weights of the module. Larger learning rates will cause drastic changes at updates and could lead to either oscillations around the minima or missing the minima all together, while unreasonably smaller rate would lead to very slow convergence. The Base learning rate is the initial learning rate at the start of the learning process.

Momentum captures the direction of the last weight update and helps to reduce oscillations and the possibility of getting stuck in a local minima. Momentum ranges from 0 to 1 and typically higher value such as 0.9 is used. However, it is important to use a lower learning rate when using a higher momentum to avoid drastic weight updates.

You may choose a solver type from a list of available types, the default one is the stochastic gradient descent.

Use Advanced learning rate options to further specify how the learning rate changes during the training.
There are several learning rate update policies (or curves) to choose from.

**Step size** determines how often the learning rate should be adjusted (in number of iterations).

**Gamma** controls the amount of change in the learning rate (determines the learning rate function shape) at every adjustment step.

By checking the **Visualize LR** box you can visualize the learning rate change as a curve.

### 5.5 Running the Training Process

After you complete configuring the model, you can launch the model training process by clicking the **Run** icon in the model manage panel. The Training Tool starts the process and shows the progress bar and the status of the model as it is being trained:
Once the training completes the model changes its status in the list to **Training completed**.

For more information about saving models and model statuses, see *Saving, Editing and Reviewing a Model*. 
6 Saving, Editing and Reviewing a Model

When you click the **Models**, the **Models** panel comes up with the **New Model** icon and the list of previously saved models. A model in the list can be in one of three states: **Draft**, **Ready** or **Test Completed**.

You can only save the model as a **Draft** at any moment prior to setting all mandatory fields. The **Ready** status indicates that you have set all mandatory fields for the saved model and it is ready to be trained. The **Train Completed** status is achieved when the model has completed training with the associated dataset.

To find a model by the given unique name, use the **Search** field.

You can **Rename**, **Edit** or **Delete** a model in the **Draft state** using the toolbar in the right upper corner for that model.

For a model which is in the **Ready state** the **Run** operation is additionally available.

To view or edit a model in either **Draft** or **Ready** state, select it from the list or click the **Edit** icon in the toolbar.
For a model in the Training Completed state, Rename, Duplicate and Delete operations are available.

To view the details of a completed model, select it from the list. Once you choose the model, you are redirected to the View model page.

In the Information tab you can find the data about training duration, number of epochs, also training, validation and testing accuracy. Below that you can see the graph showing the dynamically changing values of error, loss and learning rate.

To download the model, use the link Download in the top right corner of the Information tab. After clicking it, you get the browser dialog suggesting you to accept the download of the archive that contains a model snapshot at the last training epoch.

In the Model Analysis tab you can review results of the model classification for validation and testing datasets (if specified when configuring the model).
Clicking the **Show more** button opens the **Confusion Matrix**.
If the model has just started training, the confusion matrix is not available until at least one training epoch is finished. Once it ready you get appropriate notification at the top of the screen. The confusion matrix contains information about the general model performance on the particular dataset (testing or validation) and includes data on hits and misses, accuracy, F1 score etc.

In the Dataset tab you can see the table which includes images from the given dataset (testing or validation) and Top-5 predictions for each of them.
This table is scrollable, so that you can review each image one-by-one.

In the **Testing** tab you can test the trained model on the single image. Provide the input image, preview it and run testing. Once the testing is complete you can see Top-5 predictions for the given image, so that you can manually check the model correctness on the arbitrary image.
The Details tab shows the snapshot of model configuration parameters specified before training.
## View model

### Mnist Model

<table>
<thead>
<tr>
<th>Information</th>
<th>Details</th>
<th>Parameters</th>
<th>Image</th>
<th>Advanced options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology:</td>
<td>Mnist Model</td>
<td>Training epochs: 30</td>
<td>Dimensions: 28 x 28</td>
<td>Step Size: 33 (no information)</td>
</tr>
<tr>
<td>Dataset:</td>
<td>LeNet</td>
<td>Validation interval: 1</td>
<td>Type: Grayscale</td>
<td>Gamma: 0.1 (no information)</td>
</tr>
<tr>
<td>Name: BBD Dataset</td>
<td>Batch size: 100</td>
<td>Batch size: 100</td>
<td>Type: Grayscale</td>
<td>Power: 0.75</td>
</tr>
<tr>
<td>Number of Categories: 10</td>
<td>Solver type: SGD</td>
<td>Learning Rate: 0.01</td>
<td>Encoded: PNG</td>
<td>Solver type: SGD</td>
</tr>
<tr>
<td>Time Submitted: 13:12:20</td>
<td>Policy: step</td>
<td>Weight Decay: 0.0005</td>
<td></td>
<td>momentum: 0.9</td>
</tr>
<tr>
<td>Dataset size: 79.774 MB</td>
<td>Crop Size (no information)</td>
<td>Preize transformation: Squash</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backend: LMDB</td>
<td>Subtract mean (no information)</td>
<td>Regularization type: (no information)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7 Advanced Features

7.1 Jupyter Notebook* editor

As an advanced feature the Training Tool provides the embedded Jupyter Notebook* editor with access to the installed Docker* container with the Caffe* framework. It allows you to develop and run your own Python scripts for reproducing the whole workflow available through the Training Tool user interface including:

- Uploading image files
- Generating datasets
- Creating and training your deep learning models.

For developing your scripts, you can use the GUI-based editor or the Jupyter terminal CLI console. Also using the terminal gets you access to uploads, datasets and model files.

To find more information about the Jupyter Notebook, see this Quick Start Guide.

**NOTE:** Note that all objects (image uploads, datasets, models) created using CLI are not accessible though the user interface.

Once you click the Advanced tab, the Jupyter dashboard page comes up:

In this page you can find the provided sample notebook files, run the GUI editor or open the terminal console to create your own Python scripts.
7.2 Using Sample Routines

There are two notebook files that provide step-by-step instructions for implementing the training process and performing image classification.

The sample files contain explanatory text followed by executable Python code fragments for each step.

To run a certain code fragment, select the fragment and click the button \( \text{Shift} + \text{Enter} \) in the toolbar.

To stop the code execution, press the button \( \text{ } \text{I} \text{I} \).

**00-classification.ipynb**

Use this sample to go through the image classification process with the bundled pre-trained CaffeNet* model that is deployed with the Caffe* framework. The sample provides descriptions of all required steps from loading to testing the model with the custom image.

**01-learning-lenet.ipynb**

This sample describes the steps for training the sample LeNet-based model.

7.3 Using Jupyter Notebook* GUI Editor

You can create your custom Python scripts using the Jupyter Notebook* GUI. To launch the editor, click the **New** button and choose the **Python 2** item from the drop-down list:
7.4 Using Jupyter Notebook* Terminal Console

As an alternative to GUI editor, you can use the Terminal Console to create your Python scripts.

To open the terminal, click the **New** button and choose the **Terminal** item from the drop-down list:

Besides developing scripts, the terminal gives you access to all workflow objects: uploads, datasets and model files.

During installation you have specified the file system path which is to be mounted as a volume inside the Docker* container (the **Mount file system path** setting). Inside that directory the installer creates the /dlsdk directory that keeps all upload, dataset and model objects This directory matches to the /workspace/dlsdk directory in the Docker context and has the following structure:

```
/workspace/dlsdk/
    custom-topologies/
    jobs/
    datasets/
```
NOTE: Note, that the Jupyter default directory is /opt/jupyter. That causes files removal after the Docker is updated to a new version. To save the files in the server file system, use the /workspace/ directory as a work directory.

7.5 NNCompressor Tool

The Intel® Deep Learning SDK Training Tool also provides the NNCompressor tool that enables reducing complexity of a convolutional neural network. The tool implements the Low Rank Approximation algorithm (LRA) and deployed as the nncompress Python module that can be used in the Jupyter® terminal console.

For more information about the tool, refer to the NNCompressor-README.html file delivered within the release package.
8 Additional Resources

To ask questions and share information with other users of the Intel® Deep Learning SDK Training Tool, visit Intel® Deep Learning SDK forum.