Open for Innovation
KNIME

KNIME Deep Learning Integration Installation Guide

KNIME AG, Zurich, Switzerland
Version 5.3 (last updated on 2024-01-10)

Table of Contents

Introduction

This document describes how to install the KNIME Deep Learning Integrations. The following deep learning libraries have been integrated:

- The KNIME Keras Integration based on Keras deep learning framework.
- The KNIME TensorFlow 1 Integration based on TensorFlow 1* .
- The KNIME TensorFlow 2 Integration based on TensorFlow 2* .

  * TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc.
- The KNIME ONNX Integration based on ONNX.
- The KNIME Deeplearning4j Integration based on Deeplearning4j.

The **KNIME Keras Integration**, the **KNIME TensorFlow 1 Integration**, the **KNIME TensorFlow 2 Integration**, and the **KNIME ONNX Integration** depend on an existing Python installation, which requires certain Python dependencies to be installed. In order to set up Python for the Deep Learning Integrations, follow the instructions in the Setting up Python for KNIME Deep Learning section.

For detailed instructions how to manually install and setup the individual KNIME Deep Learning Integrations, please refer to the corresponding sections below.

On Apple computers with the Apple Silicon Chips (M1, M2,…) since 2020, these extensions might not work anymore.

Setting up Python for KNIME Deep Learning

This section describes how to set up Python for the **KNIME Keras Integration**, the **KNIME TensorFlow 1 Integration**, the **KNIME TensorFlow 2 Integration**, and the **KNIME ONNX Integration**.

Anaconda Setup

Similar to the KNIME Python Integration, KNIME Deep Learning uses Anaconda to manage Python environments (for a short introduction to Anaconda and how it is used in KNIME, see this section). If you already installed Anaconda, e.g. for the KNIME Python Integration, skip this step.

Get and install the latest Anaconda version (`Anaconda ≥ 2019.03, conda ≥ 4.6.2`) from

here. On the Anaconda download page you can choose between Anaconda with Python 3.x or Python 2.x, however this only affects the root `conda` environment, which we will not use (as we are creating our own). Therefore, you can choose either one (if you're not sure, we suggest selecting Python 3).

Installation

Next, install the desired deep learning integrations using the KNIME Analytics Platform Update Site. In KNIME Analytics Platform, go to *File → Install KNIME Extensions*. The integrations can be found under *KNIME Labs Extensions* or by entering Deep Learning into the search box.

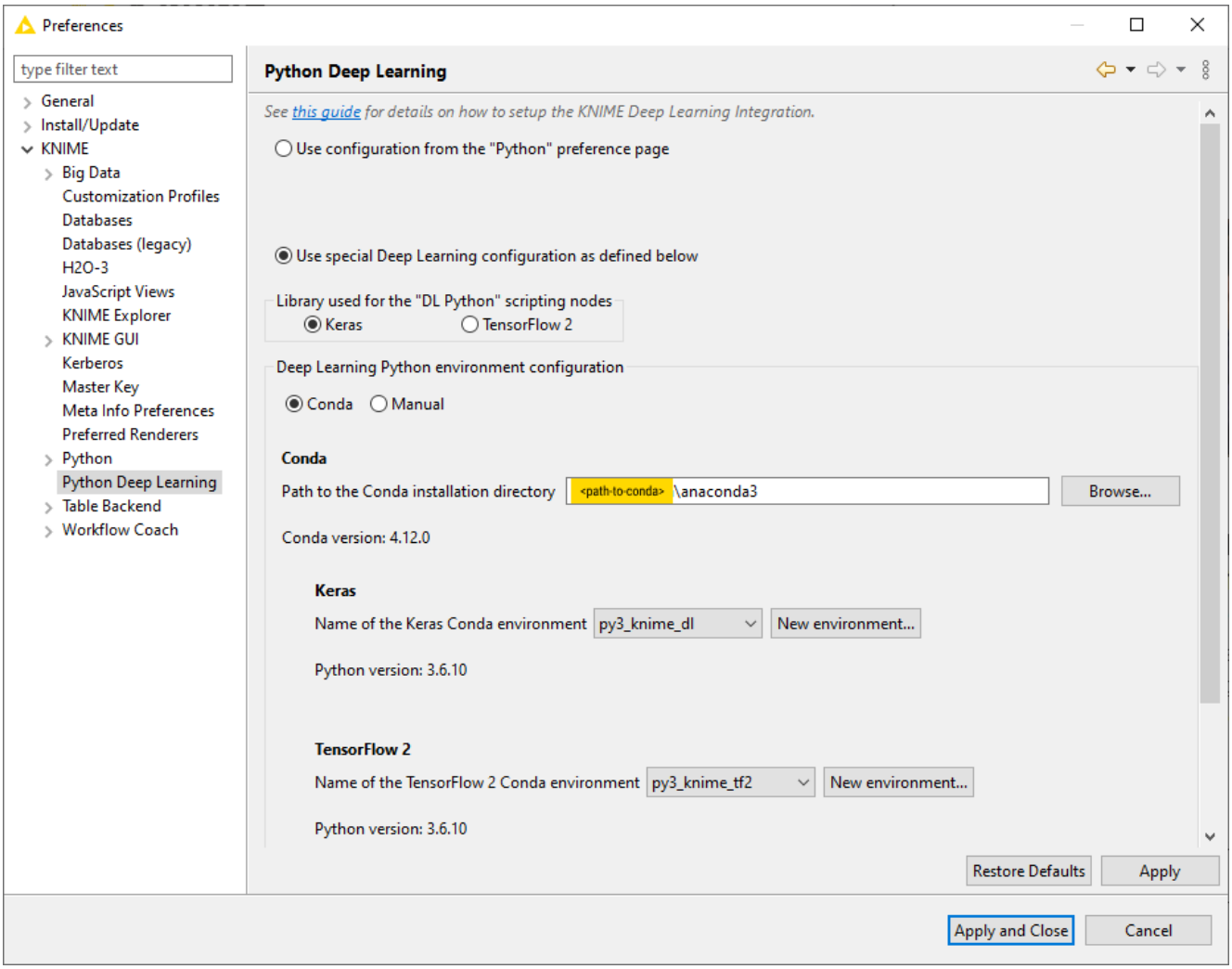
Configure Python for KNIME Deep Learning

After the integrations have been installed, go to Python Deep Learning Preference page located at *File → Preferences*, then select *KNIME → Python Deep Learning* from the list on the left. A dialog opens giving you two options for configuring the Python Deep Learning environment:

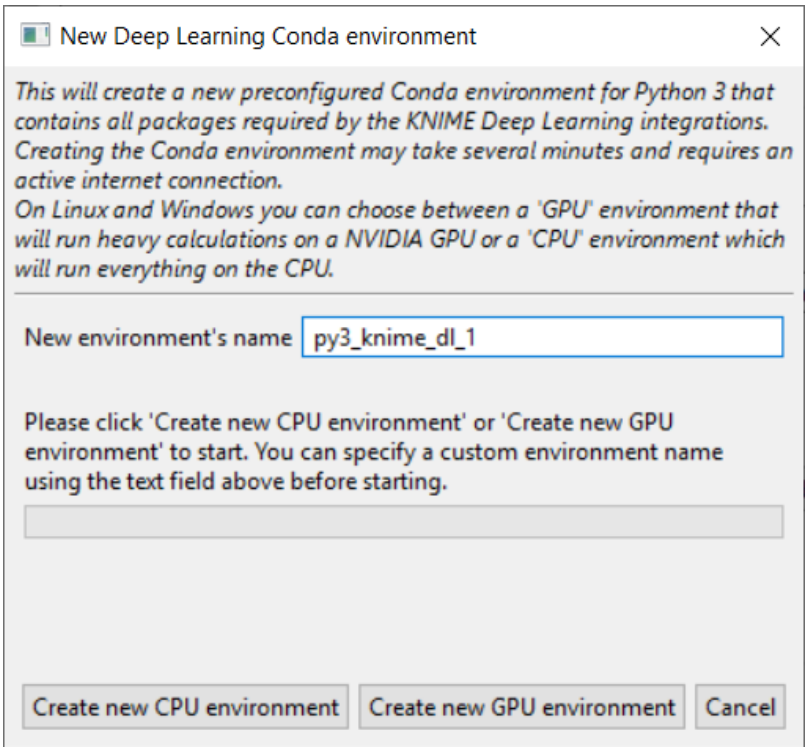
Option 1: Use special Deep Learning configuration (recommended)

By selecting this option, KNIME Deep Learning will use a Python 3 environment different from the one configured in the Python Preference page.

Similar to the Python Preference page, you can either automatically create Python 3 environments containing all required packages (by selecting the **Conda** sub option), or point to Python start scripts that activate a suitable environment you created manually (by selecting the **Manual** sub option). We recommend to choose **Conda**. If you do so, the dialog should look like the screenshot shown below.



In this dialog, provide the path to the folder containing your Anaconda installation (the default installation path is documented here). Once you have entered a valid path, the installed conda version is displayed and KNIME automatically checks for all available conda environments. Underneath the conda version number, you can choose which conda environment should be used for the KNIME Keras nodes and for the KNIME TensorFlow 2 nodes by selecting it from the combo boxes. If you have already set up a Python Deep Learning environment containing all the necessary dependencies for KNIME Deep Learning, just select it from the list and you are ready to go. If you do not have a suitable environment available, click the *New environment…* button. This opens the following dialog:



First, provide a name for the new environment. Next, choose if you want to create a new CPU or GPU environment and click the corresponding button. This creates a new `conda` environment containing all required Python dependencies for Keras or for TensorFlow 2.

Only choose GPU if you have a TensorFlow compatible GPU available. More information about Keras GPU support can be found here. More information about TensorFlow 2 GPU support can be found here.

Depending on your internet connection, the environment creation may take a while as all packages need to be downloaded and extracted.

Once the environment is successfully created, the dialog closes and the new environment is selected automatically. If everything worked out fine, the Python version is now shown below the environment selection.

Selecting the environment for the DL Python nodes

Above the *Deep Learning Python environment configuration*, you can select which library should be used for the *"DL Python" scripting nodes*. If *Keras* is selected the environment selected for the Keras nodes will be used and the Keras Python library (and the TensorFlow 1 Python library) will be available when using *"DL Python" scripting nodes*. If *TensorFlow 2* is selected the environment selected for the TensorFlow 2 nodes will be used and the TensorFlow 2 Python library will be available when using *"DL Python" scripting nodes*.

Option 2: Use Python configuration

If you choose this option, KNIME Deep Learning will use the same Python 3 environment as configured in the Python Preference page. This assumes that this Python 3 environment already contains all required Python Deep Learning packages (which have to be installed manually). A list of the required Python Deep Learning packages for the Keras Integration is documented in the KNIME Keras Integration Installation section. This option is intended for advanced users, we recommend to choose **Option 1**.

KNIME Keras Integration installation

This section describes how to install the KNIME Keras Integration.

Similar to the KNIME Python Integration, the KNIME Keras Integration uses an existing Python installation, which is installed alongside of KNIME and depends on certain Python packages that have to be installed.

You will need to:

1. Install the KNIME Keras Integration in KNIME Analytics Platform
2. Set up Python. We **recommend** you set up Python using the Python Deep Learning Preference page. To do so follow the instructions in the Setting up Python for KNIME Deep Learning section.

   However it is also possible to manually set up the Python installation, by following the instructions given in the Manual Python Installation section.

Installing the KNIME Keras Integration

The KNIME Keras Integration extension can be installed using the KNIME Analytics Platform Update Site. In KNIME Analytics Platform, go to *File* → *Install KNIME Extensions*. The integration can be found under *KNIME Labs Extensions* or by entering Keras into the search box.

If Keras is not displayed as an available deep learning back end, you may need to restart KNIME Analytics Platform.

After you installed the KNIME Keras Integration and set up Python from the Python Deep Learning Preference page (see Setting up Python for KNIME Deep Learning section) you are ready to go, and you **do not** need to read through the next section.

Manual Python Installation

Only follow the instructions given in this section if you want to set up Python for the KNIME Keras Integration manually. We recommend instead to set up Python using the Python Deep Learning Preference page. To do so, please go to the Setting up Python for KNIME Deep Learning.

1. KNIME executes Keras in a local Python installation, which has to be set up manually. If you have already set up your Python installation, you can skip this step. We **highly recommend** setting up a `conda` environment as described in our Python Installation Guide. In this guide you can either follow the Quickstart Section, or follow the Full Installation Section if you'd like to have a detailed walkthrough.

   The KNIME Keras Integration only supports Python 3, i.e. it is necessary to set up Python 3.

2. Additionally to the Python packages installed in the previous step, the KNIME Keras Integration depends on further packages. If you are using Anaconda, they can be installed with a single command. The required packages are `h5py=2.8`, `tensorflow-mkl=1.12` (`tensorflow=1.12` for GPU), and `keras=2.2.4` (`keras-gpu=2.2.4` for GPU). For the CPU version use the command:

   ```
   conda install --name <your_env_name> h5py=2.8 tensorflow-mkl=1.12 keras=2.2.4
   ```

   and for the GPU version the command:

   ```
   conda install --name <your_env_name> h5py=2.8 tensorflow=1.12 keras-gpu=2.2.4
   ```

   where `<your_env_name>` is the name of your conda environment (e.g. `py3_knime`).

   A general description about how to install further Python packages using Anaconda can be found here.

The packages `keras` and `keras-gpu` are only available with newer conda versions (minimum conda version: 4.3.30). If you have an older version, you can update `conda` using the command `conda update conda`.

In case you run into problems within KNIME due to missing Keras dependencies or if you want to double check that everything was set up correctly, here is a list of Keras dependencies (these should have been installed automatically):

- h5py (version: 2.8)
- numpy (version: 1.15)
- pyyaml (version: 3.13)
- scipy (version: 1.1)
- six (minimum version: 1.11)
- tensorflow or tensorflow-gpu (version: 1.12)

If you are using Anaconda, you can check whether these dependencies are installed by running:

```
conda list -n <your_env_name>
```

where `<your_env_name>` is the name of your conda environment.

Additional extensions

For the KNIME Deep Learning Integration the extension *KNIME Image Processing - Deep Learning Extension* is available. It provides support for images from *KNIME Image Processing*.

GPU support

Keras is able to accelerate deep learning models using a compatible NVIDIA® GPU via TensorFlow. Most of the required dependencies for GPU (i.e. CUDA® and cuDNN) will be automatically installed by Anaconda when installing the `conda` packages `tensorflow=1.12` and `keras-gpu=2.2.4`. The only additional requirement that needs to be installed manually is the latest version of the NVIDIA® GPU driver.

Tensorflow `1.12` requires a NVIDIA GPU card with CUDA Compute Capability 3.5 or higher. See the list of CUDA-enabled GPU cards to check if your GPU card meets the requirements.

KNIME TensorFlow 1 Integration Installation

This section describes how to install the KNIME TensorFlow 1 Integration for use with KNIME Analytics Platform. With the KNIME TensorFlow 1 Integration it is possible to read and execute TensorFlow 1 Saved Models using the TensorFlow 1 Java API, which is independent of Python.

Installation

The KNIME TensorFlow 1 Integration can be installed using the KNIME Analytics Platform Update Site. In KNIME Analytics Platform, go to *File* → *Install KNIME Extensions*. The integration can be found under *KNIME Labs Extensions* or by entering TensorFlow 1 into the search box.

If TensorFlow 1 is not displayed as an available deep learning back end, you may need to restart KNIME Analytics Platform.

Advanced

TensorFlow 1 Saved Models can be also executed via Python. If you want to use TensorFlow 1 models in DL Python nodes with custom Python scripts, you need a Python installation alongside KNIME. This Python installation has the same requirements as the KNIME Keras Integration. In order to install Python and the necessary dependencies, please refer to the Python Installation Section.

GPU Support

This section describes how to set up GPU support for the KNIME TensorFlow 1 Integration.

GPU support for the KNIME TensorFlow 1 Integration (which uses the TensorFlow 1 Java API) is generally independent of the GPU support the KNIME Keras Integration (which uses Python). Hence, they have to be set up individually.

Due to limitations by TensorFlow, the GPU support for the KNIME TensorFlow 1 Integration can't be used on Mac. Only Linux and Windows are supported.

The KNIME TensorFlow 1 Integration uses TensorFlow version `1.13.1`, which requires the following NVIDIA® software to be installed on your system:

- NVIDIA® GPU drivers - CUDA® `10.0` requires `410.x` or higher.
- CUDA® Toolkit - TensorFlow (≥ `1.13.0`) supports CUDA® `10.0`. Detailed installation instructions can be found here.

Make sure to install CUDA `10.0`. CUDA `10.1` will **not** work.

- cuDNN - (version ≥ `7.4.1`) - select cuDNN `v7.6.0` (May 20, 2019), for CUDA® `10.0`. Detailed installation instructions can be found here.

KNIME TensorFlow 2 Integration Installation

This section describes how to install the KNIME TensorFlow 2 Integration for use with KNIME Analytics Platform. With the KNIME TensorFlow 2 Integration it is possible to read, modify, execute, and train TensorFlow 2 Keras Models using the TensorFlow 2 Python API.

Similar to the KNIME Python Integration, the KNIME TensorFlow 2 Integration internally uses an existing Python installation, which is installed alongside KNIME and depends on certain Python packages that have to be installed. Please follow the instructions in the Setting up Python for KNIME Deep Learning section to setup Python to be used with TensorFlow 2.

Installation

The KNIME TensorFlow 2 Integration can be installed using the KNIME Analytics Platform Update Site. In KNIME Analytics Platform, go to *File* → *Install KNIME Extensions*. The integration can be found under *KNIME Labs Extensions* or by entering TensorFlow 2 into the search box.

If TensorFlow 2 is not displayed as an available deep learning back end, you may need to restart KNIME Analytics Platform.

GPU Support

TensorFlow 2 can accelerate deep learning models using a compatible NVIDIA® GPU. Most of the required dependencies for GPU (i.e. CUDA® and cuDNN) will be automatically installed by Anaconda when creating the Python environment on the `Python Deep Learning` preference page. The only additional requirement that needs to be installed manually is the latest version of the NVIDIA® GPU driver.

Tensorflow 2 requires an NVIDIA GPU card with CUDA Compute Capability 3.5 or higher. See the list of CUDA-enabled GPU cards to check if your GPU card meets the requirements.

KNIME ONNX Integration Installation

This section explains how to install the KNIME ONNX Integration to be used with KNIME

Analytics Platform.

Installation

The KNIME ONNX Integration can be installed using the KNIME Analytics Platform Update Site. In KNIME Analytics Platform, go to *File* → *Install KNIME Extensions*. The integration can be found under *KNIME Labs Extensions* or by entering ONNX into the search box.

ONNX Dependencies

Like the KNIME Keras Integration, the KNIME ONNX Integration runs using the KNIME Python Integration and depends on additional Python packages. We **recommend** to set up Python using the Python Deep Learning Preference page. This will set up Python for all KNIME Deep Learning Integrations at once including all ONNX dependencies. To do so, please go to the Setting up Python for KNIME Deep Learning section.

If you want to set up Python manually, the KNIME ONNX Integration depends on the following `pip` Python packages:

- `onnx==1.4.1`
- `onnx-tf==1.2.1`

The packages listed above are not available through `conda`. However, they can be easily installed into an existing `conda` environment using `pip`. To do so, just activate the `conda` environment which you want to add the packages to and run a `pip install` command, e.g. `pip install onnx==1.4.1`.

KNIME Deeplearning4j Installation

This section explains how to install KNIME Deeplearning4j Integration to be used with KNIME Analytics Platform.

Installation

The KNIME Deeplearning4j Integration can be installed using the KNIME Analytics Platform Update Site. In KNIME Analytics Platform, go to *File* → *Install KNIME Extensions*. The integration can be found under *KNIME Labs Extensions* or by entering Deeplearning4j into the search box.

GPU Support

The KNIME Deeplearning4j Integration supports acceleration of deep learning models using a compatible NVIDIA® GPU. For the GPU support, CUDA® Toolkit 8.0 must be installed on your system. Detailed installation instructions can be found here.

Known Issues

Older versions of KNIME Analytics Platform, used together with older versions of the KNIME Deep Learning Integrations dependencies, may lead to errors. These errors can be fixed by updating KNIME Analytics Platform and the KNIME Deep Learning Integrations to the latest versions. Furthermore, if you are using Python, please make sure to use the recommended conda package versions. The following issues are currently known:

- Node fails with error **AttributeError: '[..]' object has no attribute 'inbound_nodes'** at the bottom of a Python traceback in the KNIME log (KNIME 3.5.x only).

  Keras version 2.1.3 introduced breaking changes that were adapted in KNIME version 3.6.0. Please upgrade KNIME to version 3.6.0 or downgrade Keras to version 2.1.2 or below (minimum version: 2.0.7).

- Node fails with error **UnicodeEncodeError: 'ascii' codec can't encode character [..] in position [..]: ordinal not in range(128)** at the bottom of a Python traceback in the KNIME log.

  This error may occur when using Keras version 2.1.2 to load a Keras network that was saved using an older Keras version. Make sure not to use Keras 2.1.2 in such cases.

- Node fails with both an error **SystemError: unknown opcode** and a warning **XXX lineno: [..], opcode: [..]** in the KNIME log.

  This is a Python related error that occurs when loading a Keras network containing a Lambda expression (e.g. within a Lambda layer) that was saved using a different Python version. Make sure you use the same Python version for saving and loading the same network.

- DL Keras Network Learner fails with error **AttributeError: 'int' object has no attribute 'dtype'** at the bottom of a Python traceback in the KNIME log when the Clip norm option is enabled in the node dialog and Keras (TensorFlow) is the selected back end.

This is a TensorFlow related error that only occurs in very specific situations. Try to use a different Keras back end to work around this issue.

- `DL Python Network Executor scripting node outputs wrong numerical values when using Flatbuffers serialization library (KNIME 3.6.1 and older).`

  Flatbuffers in KNIME does not support float32 data at the moment. We recommend using Apache Arrow as serialization library instead. This option can be changed in KNIME via *File → Preferences → KNIME → Python → Serialization library*.

- `TensorFlow 1 and 2 on Windows: DL Python scripting nodes fails while saving the model with the error "The system cannot find the path specified."`

  This error can appear due to the path length limit of 260 characters in Windows. A guide on how to disable this limit for advanced users can be found here. The related TensorFlow issue can be found here.

- `TensorFlow 2: Cannot load a model downloaded from the TensorFlow Hub.`

  TensorFlow Hub models do not always contain all information necessary to run them in KNIME. The example workflow "02_Use_a_TFHub_Model" shows how a TensorFlow Hub model can be used by using a simple component.

- `Installation test for Python back end ⋯ timed out. Please make sure your Python environment is properly set up and consider increasing the timeout (currently 25000 ms) using the VM option '-Dknime.dl.installationtesttimeout=<value-in-ms>'.`

  Before using the Python environment, the Deep Learning nodes check if all required Python packages can be imported. If the import does not finish after the default timeout of 25 seconds this is considered a failure. Please make sure that your environment is set up correctly by checking for errors on the "Python Deep Learning" preference page (see Setting up Python for KNIME Deep Learning). On some systems importing the required Python packages (especially `tensorflow`) can take longer than 25s. In this case, the timeout for the installation tests can be increased by setting the VM option `-Dknime.dl.installationtesttimeout=<value-in-ms>`. Do this by adding the line to the bottom of the `knime.ini` file.

- Networks containing an LSTM layer don't work on macOS

  This error is likely caused by problems with the MKL library. You can install the nomkl package to you conda environment via `conda install nomkl` to uninstall MKL. For more details see https://docs.anaconda.com/mkl-optimizations/index.html#uninstalling-mkl.

KNIME AG
Talacker 50
8001 Zurich, Switzerland
www.knime.com
info@knime.com