

Data & AI Tech Immersion Workshop – Product Review Guide and Lab Instructions

Day 2, Experience 4 - Making deep learning portable with ONNX

- [Data & AI Tech Immersion Workshop – Product Review Guide and Lab Instructions](#)
 - [Day 2, Experience 4 - Making deep learning portable with ONNX](#)
 - [Technology overview](#)
 - [Scenario overview](#)
 - [Task 1: Train and deploy a deep learning model](#)
 - [Wrap-up](#)
 - [Additional resources and more information](#)

Technology overview

Using the [main Python SDK](#) and the [Data Prep SDK](#) for Azure Machine Learning as well as open-source Python packages, you can build and train highly accurate machine learning and deep-learning models yourself in an Azure Machine Learning service Workspace. You can choose from many machine learning components available in open-source Python packages, such as the following examples:

- [Scikit-learn](#)
- [Tensorflow](#)
- [PyTorch](#)
- [CNTK](#)
- [MXNet](#)

After you have a model, you use it to create a container, such as Docker, that can be deployed locally for testing. After testing is done, you can deploy the model as a production web service in either Azure Container Instances or Azure Kubernetes Service. For more information, see the article on [how to deploy and where](#).

Then you can manage your deployed models by using the [Azure Machine Learning SDK for Python](#) or the [Azure portal](#). You can evaluate model metrics, retrain, and redeploy new versions of the model, all while tracking the model's experiments.

For deep neural network (DNN) training using TensorFlow, Azure Machine Learning provides a custom TensorFlow class of the Estimator. The Azure SDK's [TensorFlow estimator](#) (not to be conflated with the `tf.estimator.Estimator` class) enables you to easily submit TensorFlow training jobs for both single-node and distributed runs on Azure compute.

The TensorFlow Estimator also enables you to train your models at scale across CPU and GPU clusters of Azure VMs. You can easily run distributed TensorFlow training with a few API calls, while Azure Machine Learning will manage behind the scenes all the infrastructure and orchestration needed to carry out these workloads.

Azure Machine Learning supports two methods of distributed training in TensorFlow:

- MPI-based distributed training using the [Horovod](#) framework
- Native [distributed TensorFlow](#) via the parameter server method

The [Open Neural Network Exchange](#) (ONNX) format is an open standard for representing machine learning models. ONNX is supported by a [community of partners](#), including Microsoft, who create compatible frameworks and tools. Microsoft is committed to open and interoperable AI so that data scientists and developers can:

- Use the framework of their choice to create and train models
- Deploy models cross-platform with minimal integration work

Microsoft supports ONNX across its products including Azure and Windows to help you achieve these goals.

The interoperability you get with ONNX makes it possible to get great ideas into production faster. With ONNX, data scientists can choose their preferred framework for the job. Similarly, developers can spend less time getting models ready for production, and deploy across the cloud and edge.

You can create ONNX models from many frameworks, including PyTorch, Chainer, Microsoft Cognitive Toolkit (CNTK), MXNet, ML.Net, TensorFlow, Keras, SciKit-Learn, and more. There is also an ecosystem of tools for visualizing and accelerating ONNX models. A number of pre-trained ONNX models are also available for common scenarios. [ONNX models can be deployed](#) to the cloud using Azure Machine Learning and ONNX Runtime. They can also be deployed to Windows 10 devices using [Windows ML](#). They can even be deployed to other platforms using converters that are available from the ONNX community.

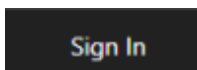
Scenario overview

In this experience you will learn how Contoso Auto can leverage Deep Learning technologies to scan through their vehicle specification documents to find compliance issues with new regulations. Then they will deploy this model, standardizing operationalization with ONNX. You will see how this simplifies inference runtime code, enabling pluggability of different models and targeting a broad range of runtime environments from Linux based web services to Windows/.NET based apps.

Task 1: Train and deploy a deep learning model

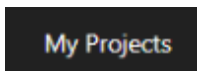
In this task, you will train a deep learning model to classify the descriptions of car components provided by technicians as compliant or non-compliant, convert it to ONNX, and deploy it as a web service. To accomplish this, you will use an Azure Notebook and Azure Machine Learning.

1. To start, open a new web browser window and navigate to <https://notebooks.azure.com>.
2. Select **Sign In** and then use your Microsoft Account to complete the sign in process.



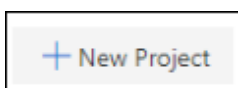
The Sign In button

3. Dismiss the dialog to create the user ID (you will not need this). Within the Microsoft Azure Notebooks portal, select **My Projects** from the menu at the top.



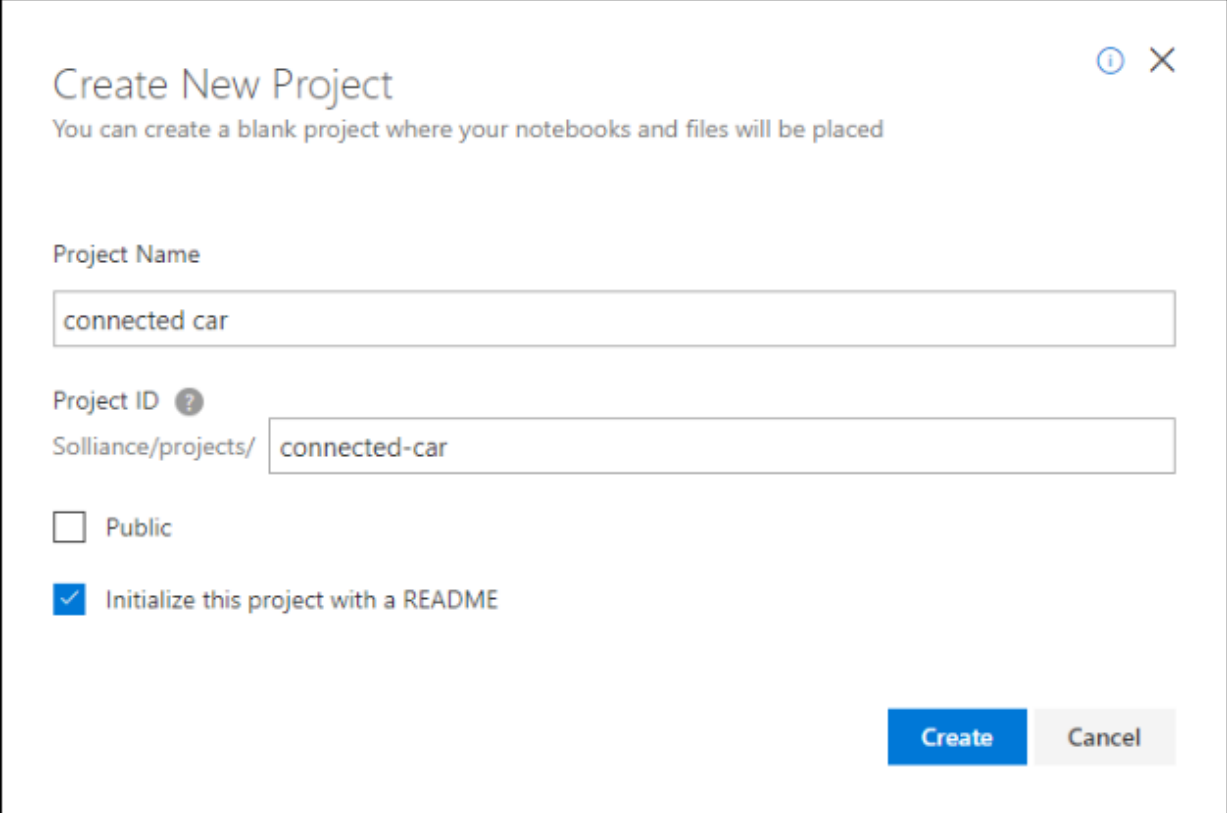
The My Projects button

4. Then select **New Project**.



The New Project button

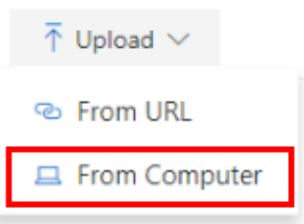
5. On the Create New Project dialog, provide a Project Name (this should be a user friendly description) and Project ID (this will form a part of the URL used to access this project in the browser) and uncheck Public. Select **Create**.



The image shows a 'Create New Project' dialog box. At the top, it says 'Create New Project' and 'You can create a blank project where your notebooks and files will be placed'. There are two input fields: 'Project Name' with the text 'connected car' and 'Project ID' with a help icon and the text 'Solliance/projects/connected-car'. Below these are two checkboxes: 'Public' (unchecked) and 'Initialize this project with a README' (checked). At the bottom right are 'Create' and 'Cancel' buttons.

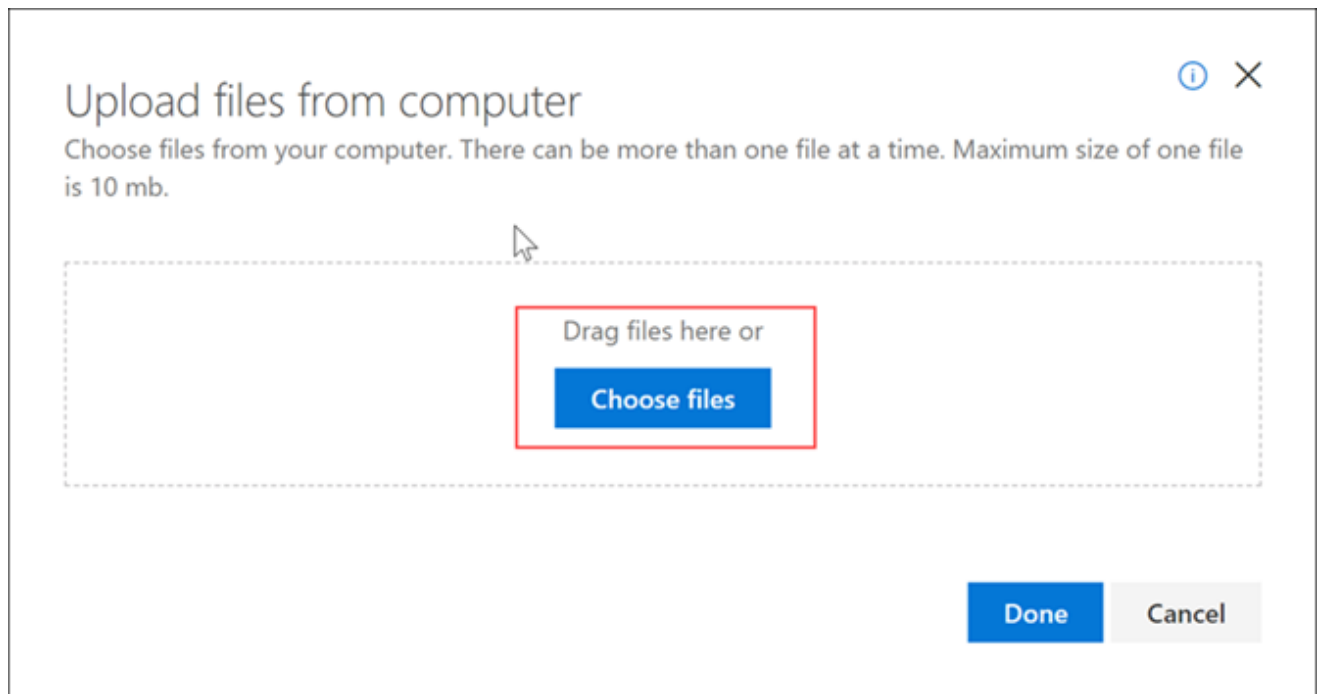
The Create New Project dialog

6. Select the **Upload** menu and then choose **From Computer**.



The Upload menu

7. In the Upload files from Computer dialog, select **Choose Files** and then select the `deep-learning.ipynb` from lab starter files (C:-files\4). Select **Upload** to upload the notebook and then select **Done** to dismiss the dialog.



The Upload files from Computer dialog

8. In the listing, select the Notebook you just uploaded (deep-learning.ipynb) to open it.
9. Follow the instructions within the notebook to complete the experience.

Wrap-up

Congratulations on completing the deep learning with ONNX experience. In this experience you completed an end-to-end process for training a deep learning model, converting it to ONNX and deploying the model into production, all from within an Azure Notebook.

To recap, you experienced:

1. Using [Keras](#) to create and train a deep learning model for classifying text data on a GPU enabled cluster provided by Azure Databricks.
2. Converting the Keras model to an ONNX model.
3. Using the ONNX model to classify text data.
4. Creating and deploying a web service to [Azure Container Instances](#) that uses the ONNX model for classification.

Additional resources and more information

To learn more about the Azure Machine Learning service, visit the [documentation](#)