Challenge 4: Deep Learning

Background

The Data Science team at AdventureWorks and your team have now learned a great deal about classical Machine Learning and have put it to practical use. If the accuracy was acceptable, AdventureWorks would likely stick with a simple, easy-to-use frameworks like scikit-learn.

However, the gear retailer wants to try deep learning algorithms on the data to see if the accuracy of gear classification improves. AdventureWorks also anticipates an influx of catalog and outdoor pictures soon and deep learning shines when the data size gets bigger. Also, the Data Science team is eager to learn more about deep learning and Convolutional Neural Networks (CNNs) for image analysis because CNNs naturally lend themselves to performing well on complex data such as images.

What differentiates Deep Learning from the more general Artificial Neural Networks is the hidden layers in its architecture which help to better “learn” features in complex data. Neural Networks are the network architectures made up of simple neurons and Deep Learning is the implementation that can solve complex problems. Also, Deep learning solutions can require less preprocessing and feature engineering.

Prerequisites

* Team decides upon a deep learning framework to use in this Challenge, after reading about several different frameworks (see [References](https://openhacks.azurewebsites.net/labs/player/microsoft-open-hack-ai#references))
* Team performs any installs or updates to the latest versions of frameworks in their chosen setup

Challenge

Use the team setup and expertise to do the following tasks.

**NOTE:** Training a Deep Learning model is faster on a GPU machine. If the team setup does not already include GPU, consider working with your coach to adjust the setup before proceeding.

Create a Convolutional Neural Network (a deep learning architecture) to classify the gear data. The architecture or design should contain the following types of layers. See [References](https://openhacks.azurewebsites.net/labs/player/microsoft-open-hack-ai#references) for resources and more information.

Suggested architecture:

1. Input Layer (3 channel image input layer)
2. Convolutional (2D)
3. Max Pooling
4. Convolutional (2D)
5. Max Pooling
6. Dense (Output layer)

There are plenty of examples online in the documentation of each framework to get the team started. Check the [References](https://openhacks.azurewebsites.net/labs/player/microsoft-open-hack-ai#references) for more.

There is architecture information in a CNTK [Tutorial](https://cntk.ai/pythondocs/CNTK_103D_MNIST_ConvolutionalNeuralNetwork.html#CNN-Model-Creation) that is helpful in understanding these concepts and implementation.

Train a model on the training dataset using the suggested architecture or an equivalent that the team wishes to try. The team will utilize a portion of their training dataset as a validation dataset. The team may have to iterate on the architecture. Make sure the best trained model is saved to disk.

Success Criteria

* Team will run a code cell in a Jupyter notebook for the coach that shows the model accuracy is 90% or greater (using the test data set from Challenge 3)
* Team will show logging that demonstrates validation accuracy that reaches 90% (using the validation dataset created in this challenge)

References

Read this first

* What is a convolutional neural net [Ref](https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/) or [Video](https://www.youtube.com/watch?v=FmpDIaiMIeA)
* High level overview of Machine Learning and CNNs [Video](https://youtu.be/k-K3g4FKS_c)

Deep Learning Frameworks

* Keras (an abstraction layer that uses TensorFlow or CNTK on the backend)
  + [Docs](https://keras.io/) And [Tutorials](https://github.com/fchollet/keras-resources)
* TensorFlow
  + [Docs](https://www.tensorflow.org/) And [Tutorials](https://www.tensorflow.org/tutorials/)
  + Suggested starting point is a CNN from a Tutorial with the Layers API
* CNTK
  + [Docs](https://www.microsoft.com/en-us/cognitive-toolkit/) And [Tutorials](https://cntk.ai/pythondocs/tutorials.html)
  + Suggested starting point is a CNN from a Tutorial with the Layer API