(/)

LOGOUT

OPEN HACK ENVIRONMENT OVERVIEW

OPEN HACK GUIDE PROVIDE FEEDBACK

1 2 3
4 5 6

Mark Complete

# Challenge 3: Deep in the Woods

## Background

The Adventure Works data science team wants to use *deep learning* techniques to build an image classification model that can identify each class of product the company sells. Specifically, the team wants to develop a *convolutional neural network* (CNN) model. You can use your choice of deep learning framework from *PyTorch*, *Keras*, *TensorFlow*, or *Cognitive Toolkit (CNTK)*.

(/)

LOGOUT

> **Note**: If you are new to deep learning and convolutional neural networks, you should review the **CNN Concepts** resources in the **References** section below before starting this challenge. If anything is unclear, ask your coach!

# Prerequisites

- A Data Science Virtual Machine (DSVM)
- The resized ***gear*** image data fom the previous challenges.
- An installation of the latest version of your chosen deep learning framework(s) based on the **References** section below.

# Challenge

There are two tasks in this challenge:

1. Create and train a convolutional neural network (CNN) model
2. Use your model with new data

## 1. Create and train a convolutional neural network (CNN) model

Create a CNN that predicts the class of an image based on the *gear* dataset.

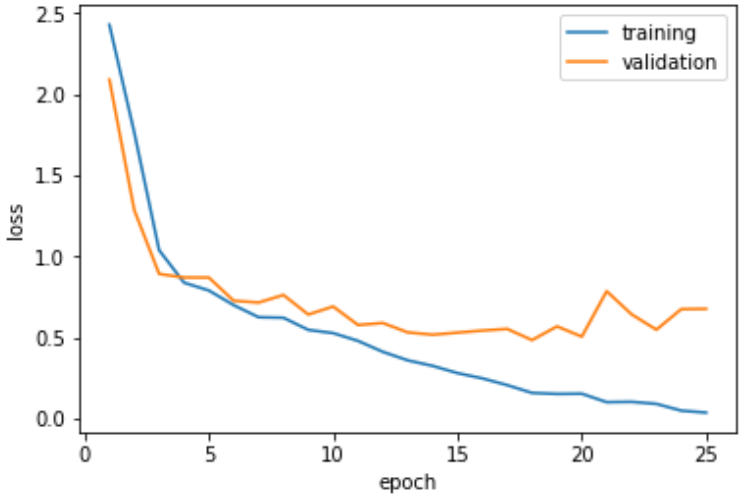
- The architecture of your model should consist of a series of *convolutional*, *pooling*, *drop*, and *fully-connected* layers that you define.

(/)

LOGOUT

- The input layer of your model must match the size and shape of the training image arrays.
- The output layer of your model must include an output for each class the model is designed to predict.
- Randomly split the data into training and validation subsets with which to train and validate the model.

- For each epoch in your training process, you should record the average *loss* for both the training and validation data; and when training is complete you should plot the training and loss values like this:




*Training and Validation Loss*

**Hints**

- Training a deep learning model is faster on a GPU machine. If the team environment does not already include GPU, consider working with your coach to adjust the environment before proceeding. The N-series virtual machine sizes in Azure support GPUs.
- As you iterate on the model architecture, save the best trained models to disk.

- Consider the following suggested starting point 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)

LOGOUT

- Try to avoid *overfitting* your model to the training data. One sign of this is that after your training and validation loss metrics converge, the training loss continues to drop but your validation loss stays the same or rises (as shown in the image above). The end result is a model that performs well when predicting the classes of images that it has been trained on, but which does not generalize well to new images.
- Techniques to help avoid overfitting include:
  - Including *drop* layers to randomly remove some features from the model.
  - Augmenting the data with re-oriented, skewed, or otherwise altered versions of training images.

## 2. Use your model with new data

Use the model to predict the class of at least five relevant images that are not included in the ***gear*** dataset. You can find these images by using Bing to search for appropriate terms, for example:

- https://www.bing.com/images/search?q=ski+helmet (https://www.bing.com/images/search?q=ski+helmet)
- https://www.bing.com/images/search?q=climbing+axe (https://www.bing.com/images/search?q=climbing+axe)
- https://www.bing.com/images/search?q=tent (https://www.bing.com/images/search?q=tent)

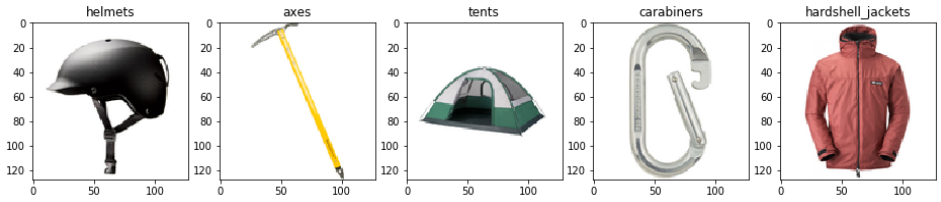
(/)

LOGOUT

- https://www.bing.com/images/search?q=carabiner (https://www.bing.com/images/search?q=carabiner)
- https://www.bing.com/images/search?q=insulated+jacket (https://www.bing.com/images/search?q=insulated+jacket)

# Success Criteria

- Successfully train a convolutional neural network model
- Plot the average training and validation loss observed when training your model
- Achieve model accuracy of **0.9** (90%) or greater using your test data set.
- Show predictions for the five images you identified in the **Challenge** section, like this:




*Gear predictions*

*(Note: Your model is not required to predict the correct class for all of the images, but it would be good if it does!)*

# References

## CNN Concepts

- An Intuitive Explanation of Convolutional Neural Networks (https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/)
- How Convolutional Neural Networks work (https://www.youtube.com/watch?v=FmpDIaiMIeA) (video)

(/)

- Demystifying AI (https://youtu.be/k-K3g4FKS_c) (video)

LOGOUT

## Deep Learning Frameworks

- **PyTorch (https://pytorch.org/)**
  - Documentation (https://pytorch.org/docs/stable/index.html)
  - Tutorials (https://pytorch.org/tutorials/)
- **Keras (https://keras.io/)** (an abstraction layer that uses a TensorFlow or CNTK backend)
  - Documentation (https://keras.io/)
  - Tutorials (https://github.com/fchollet/keras-resources)
- **TensorFlow (https://www.tensorflow.org/)**
  - Documentation (https://www.tensorflow.org/guide/)
  - Tutorials (https://www.tensorflow.org/tutorials/)
- **Cognitive Toolkit (CNTK) (https://www.microsoft.com/en-us/cognitive-toolkit/)**
  - Documentation (https://docs.microsoft.com/en-us/cognitive-toolkit/)
  - Tutorials (https://cntk.ai/pythondocs/tutorials.html)