**How to Create Custom Model With Transfer Learning**

**Objective**: If you are here, that means you have successfully created model and prediction using Microsoft Custom Vision Service (CVS). This time, we are going to create our own custom model trying to beat the accuracy of CVS.

To create a custom model and validate its accuracy, we need to use four scripts below to achieve the results. This document will describe each script in details and explain how to use them and what they output.

* custom\_model\_img\_file\_moving.py
* tensorFlow\_karas\_custom\_model.py
* custom\_model\_evaluation.py
* data\_analysis.py

**Steps**:

1. **Run custom\_model\_img\_file\_moving.py**

To begin with, user needs to run ‘custom\_model\_img\_file\_moving.py’ to separate images into appropriate folders by each classifier. For instance, if user is trying to create a custm\_model for glass\_damage, the images need to be separated based on labels for glass\_damage.

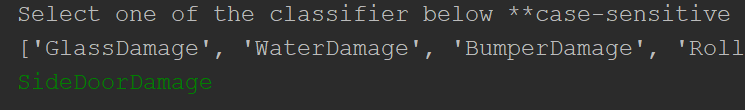
* + Before you run the script, please make sure to go to image folder and delete the previous or existing images in folders.



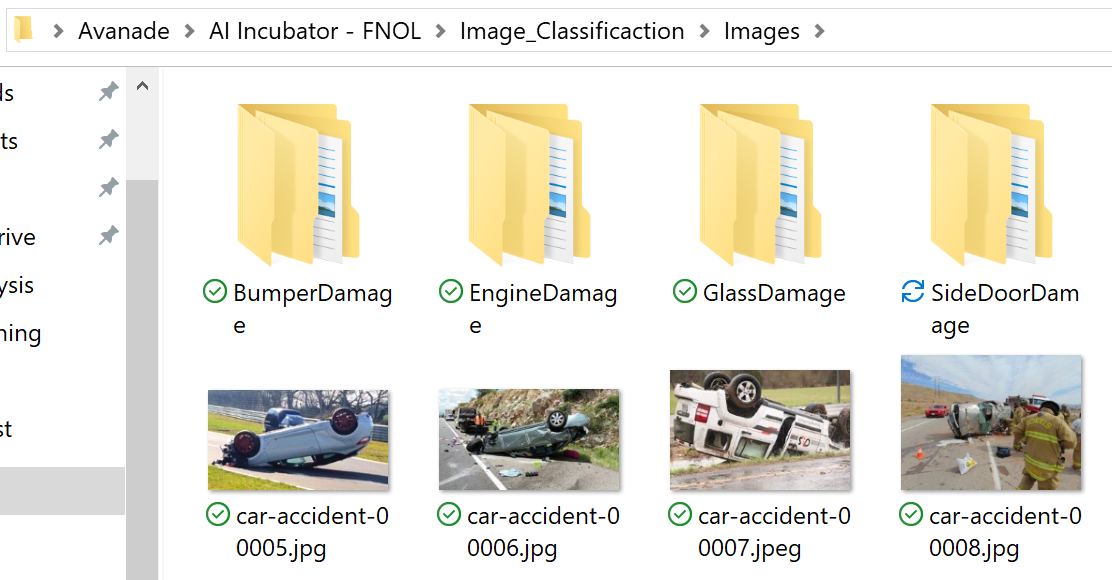
In this case, you already have ‘GlassDamage’ folder, hence, you can delete the folder then run the script.

Running custom\_model\_img\_file\_moving.py

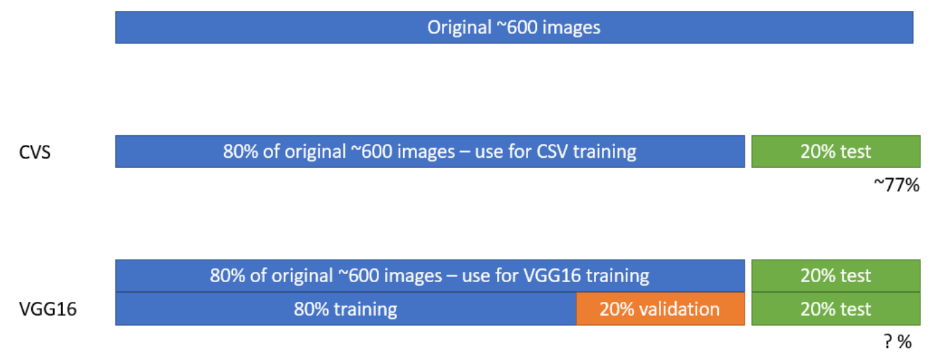
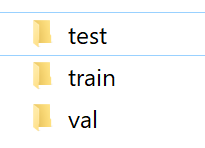
* + 1. Please provide all required paths and variables
       - **Training\_csv\_src**: provide the path to training\_car.csv, which is the product of process\_data.py
       - **Testing\_csv\_src**: provide the path to testing\_car.csv, which is the product of process\_data.py
       - **Src\_folder\_path**: provide the directory path where all the images are stored. It should be in Avanade/AI Incubator – FNOL/Image\_Classification/Images
    2. Run the script and answer the console prompt
       - Once you run the script, it asks user to select one of the classifiers so that it can split images.



* + - * Once the script runs successfully, the SideDoorDamage folder will be created in the directory where all the images are stored.



Inside of the folder consists of three folders – training, validation, and testing folder. This is because we are trying to split the data into 80% training and 20% testing data. Within training data, we conduct another 80:20 split to create training data and validation data.

1. **Run tensorFlow\_karas\_custom\_model.py**

Once the images have successfully divided, it’s time to create and train a custom model.

If you have not ran ‘custom\_model\_img\_file\_moving.py’ in step 1, please do it before running this script.

Running tensorFlow\_karas\_custom\_model.py

* + 1. Please provide all required paths and variables
       - **Training\_folder\_dir**: provide the directory path to training image folder that was created by ‘custom\_model\_img\_file\_moving.py’
       - **Testing\_folder\_dir**: provide the directory path to testing image folder that was created by ‘custom\_model\_img\_file\_moving.py’
       - **new\_folder\_path**: provide a name for the model (ex. Vgg16.h5)
       - **new\_model\_save\_dir**: provide the directory path of the new model you want to save
       - **num\_classes**: provide the output number. For instance, glassDamage can only be yes or no; thus, the num\_classes should be 2. For location, it will be 3 since there are three options (urban, suburban, and rural)
    2. Select a CNN pre-trained model
       - To select an CNN pre-trained model, user needs to import it from tensorflow.pthon.keras.applications as the image indicated below.

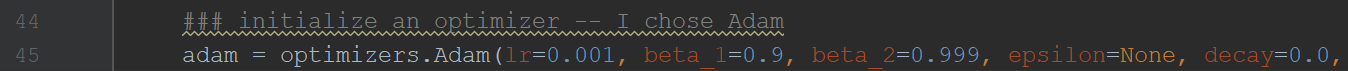
The default is set to VGG16, but user can import other CNN models such as ResNet50, VGG19, and etc.

* + - * Add the model imported to Sequential model of Keras.



If you import other models, you can change the name and parameters here.

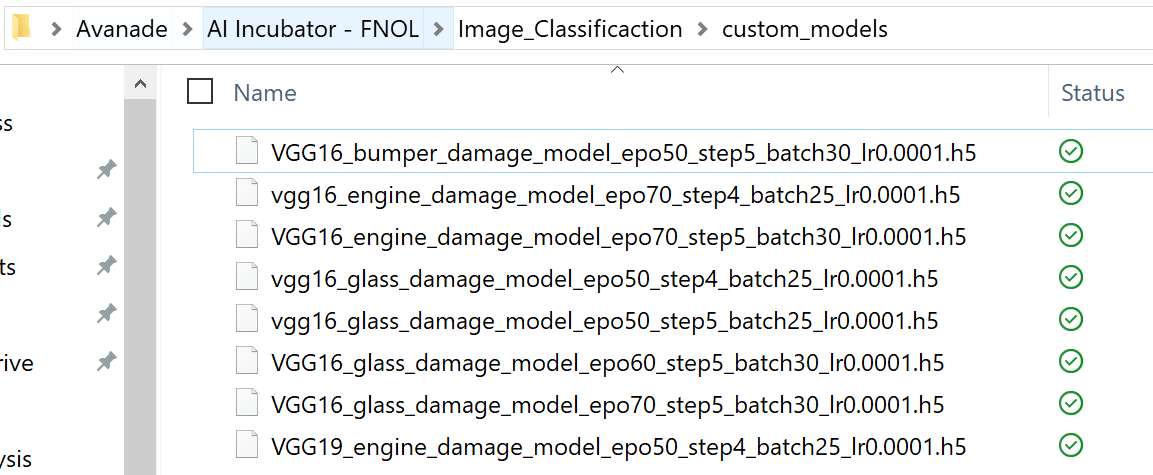
* + 1. Select an optimization algorithm
       - Optimization algorithm changes weights and bias values as it calculates cost of the network and conduct backpropagation. The default is set to Adam, and user can change the optimization algorithm in the line below.



* + - * To select another optimization algorithm, user can choose one by typing ‘optimizers.’ Ex. Optimizers.sgd
      * When selecting an optimization algorithm, user can also conduct fine-tuning by selecting learning rate, beta\_1, beta\_2, and amsgrad.
    1. Data Augmentation
* Data augmentation helps users with small size of data. For car image classification, roughly 700 images are not enough. However, we can augment it by flipping the images. Under ‘ImageDataGenerator,’ Keras does not only import and modify the images but it also augments the images. The default is set to horizontal\_flip.
  + 1. Batch, Steps\_per\_epoch, and Epochs
* When it comes to training, user can adjust the number of times images get trained.
  + Batch\_size: this refers to how many images user wants to train at one Steps\_per\_epoch.
  + Step\_per\_epoch: number of rounds that selected\_batches get processed
  + Epoch: number of times that selected step\_per\_epoch get processed.

For instance, batch\_size=30, Step\_per\_epoch=5, epochs=50, this results in 30 images \* 5 step\_per\_epoch \* 50 epochs

* + 1. Output – this script output two plots and a custom model
  + **Training plot and cost plot**
    1. Because each epoch training data and cost data are kept, there will be two plots indicated at the end.
       - Accuracy plot: it indicates training and validation accuracy throughout the training
       - Cost/Loss plot: it indicates cost value for training and validation throughout the training
  + **Custom\_model.h5**
    1. The end-result of custom model gets saved in AI Incubator – FNOL -> Image\_Classification -> custom\_models folder.



1. **Custom\_model\_evaluation.py**

Once the model has successfully created, the next step is to run custom\_model\_evaluation.py script to produce result using the test data that we have not used yet.

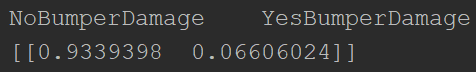
Running custom\_model\_evaluation.py

* + 1. Please provide all required paths and variables
       - **Model\_name**: custom model’s name and the custom model is the product of tensorFlow\_keras\_custom\_model.py’s product.
       - **Model\_dir**: provide the directory path to the model’s folder. It should reside in “Avanade\AI Incubator – FNOL\Image\_Classification\custom\_models
       - **Img\_folder\_path**: provide the directory path to where all images are resided
       - **Testing\_df**: provide the path to where testing\_car.csv file is located
       - **Classifier**: select which classifier to classify using the custom\_model.

**\*\*Note**: custom\_model is trained for classifier; hence, please select an appropriate classifier. If the cutom\_model is for glass damage, please assign ‘GlassDamage’ to the classifier variable

* + - * **Output\_csv\_name:** provide a name for output csv file name
    1. Run the script

Upon successful run, the console will display each image’s classification result.



All the results displayed in console will be outputted as one csv file.

1. **Data\_analysis.py**

The final step is comparing the output from custom\_model\_evaluation.py with original testing data. To do so, we run data\_analysis.py. Even though we used data\_analysis.py for CVS analysis, we use it again because the code can be pretty much reused.

Running custom\_model\_evaluation.py

* + 1. Please provide all required paths and variables
       - **Testing\_car\_df\_path**: provide the path to testing\_car.csv
       - **Custom\_model\_result\_df\_location**: provide the path to custom\_model\_result csv file. This should be the product of custom\_model\_evaluation.py.
    2. Run the script

As user runs the script, it asks user to either enter ‘custom\_vision’ or ‘custom\_model.’ Because we are trying to see the accuracy of custom\_model, type ‘custom\_model’ into the prompt.

\*\*Note: In case encountering permission errors, please check and make sure you don’t have the file opened. If you have the csv file open, please close it and type ‘yes’ to prompt.

Once it succeeds, it displays ‘Successfully Written to a CSV file.’

* + 1. Output

**Csv file**: the output is the accuracy result csv file.

**What now?**

Congratulations! Now we can compare the accuracy results from Custom Vision Service against custom model.

