HW3 Explore Notebook (2470 ONLY)

In this explore notebook, you will be picking a dataset of choice and training a CNN Model on it!

This is an open-resource assignment with some restrictions on the final deliverable. Specifically, you are required to:

- Explore and visualize the dataset you choice. Some ideas include visualizing sample images, plotting the distribution of the input dataset, etc.
- Include a preprocessing/data augmentation component (you can use Keras for this). You should NOT be simply downloading the data and using it as is.
- Make at least two interesting visualizations relating to the model and explain why it is good to consider.
  - Some valid ones include performance testing for a specific use case, visualization/analysis of latent-space outputs, or model interpretation.
  - If the resource you use already has some downstream analysis, you can replicate two of them in your own style as one of your visualizations.
- Create a modular system (feel free to use the classes you've already implemented in HW 3 or create new ones based on those). Modular means that related code is grouped into a separate class and it's easy to swap out layers and play around with hyperparameters. You may use Keras Subclassing of SequentialModel subclassing, but must take advantage of model.compile and model.fit with custom hyperparameters.
- **YOU MUST CITE YOUR SOURCE(S).**

For All Requirements:

- **Use and briefly describe the dataset. Visualize entries and get some simple statistics:** Feel free to select a dataset from the [Tensorflow Dataset (https://www.tensorflow.org/datasets/catalog/overview)](https://www.tensorflow.org/datasets/catalog/overview) list! The following datasets **CANNOT** be used:
  - MNIST and variants (including Fashion-MNIST).
  - CIFAR and variants.
- Make it easy to customize the model inside a notebook cell.
- If you would like, feel free to use transfer learning on an existing architecture.
- Try to limit time spend on this notebook to around 2-4 hours.

In [1]:
```
!python3 -VV
```

```
Python 3.9.12 (main, Apr  5 2022, 01:52:34) 
[Clang 12.0.0 ]
```

In [2]:
```
from types import SimpleNamespace

import numpy as np
import tensorflow as tf
```

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import tensorflow as tf
import tensorflow_datasets as tfds
```

In [4]:
```
%load_ext autoreload
%autoreload 2
import   assignment, conv_model, layers_keras, layers_manual
%aimport assignment, conv_model, layers_keras, layers_manual
```

In [5]:
```
from types import SimpleNamespace

import numpy as np
import tensorflow as tf
from conv_model import CustomSequential

## Run functions eagerly to allow numpy conversions.
## Enable experimental debug mode to suppress warning (feel free to remove
tf.config.run_functions_eagerly(True)
tf.data.experimental.enable_debug_mode()
```

@ONLINE {beansdata, author="Makerere AI Lab", title="Bean disease dataset", month="January", year="2020", url="[https://github.com/AI-Lab-Makerere/ibean/" (https://github.com/AI-Lab-Makerere/ibean/")](https://github.com/AI-Lab-Makerere/ibean/) }

In [6]:
```
def get_data():
    """
    Loads CIFAR10 training and testing datasets

    :return X0: training images,
            Y0: training labels,
            X1: testing images,
            Y1: testing labels
            D0: TF Dataset training subset
            D1: TF Dataset testing subset
        D_info: TF Dataset metadata
    """

    ## This process may take a bit to load the first time; should get much
    import tensorflow_datasets as tfds

    ## Overview of dataset downloading: https://www.tensorflow.org/datasets
    ## CIFAR-10 Dataset https://www.tensorflow.org/datasets/catalog/cifar10
    (D0, D1), D_info = tfds.load(
        "beans", as_supervised=True, split=["train[:50%]", "test"], with_in
    )

    X0, X1 = [np.array([r[0] for r in tfds.as_numpy(D)]) for D in (D0, D1)]
    Y0, Y1 = [np.array([r[1] for r in tfds.as_numpy(D)]) for D in (D0, D1)]

    return X0, Y0, X1, Y1, D0, D1, D_info
```

In [7]:
```
data = get_data()
```

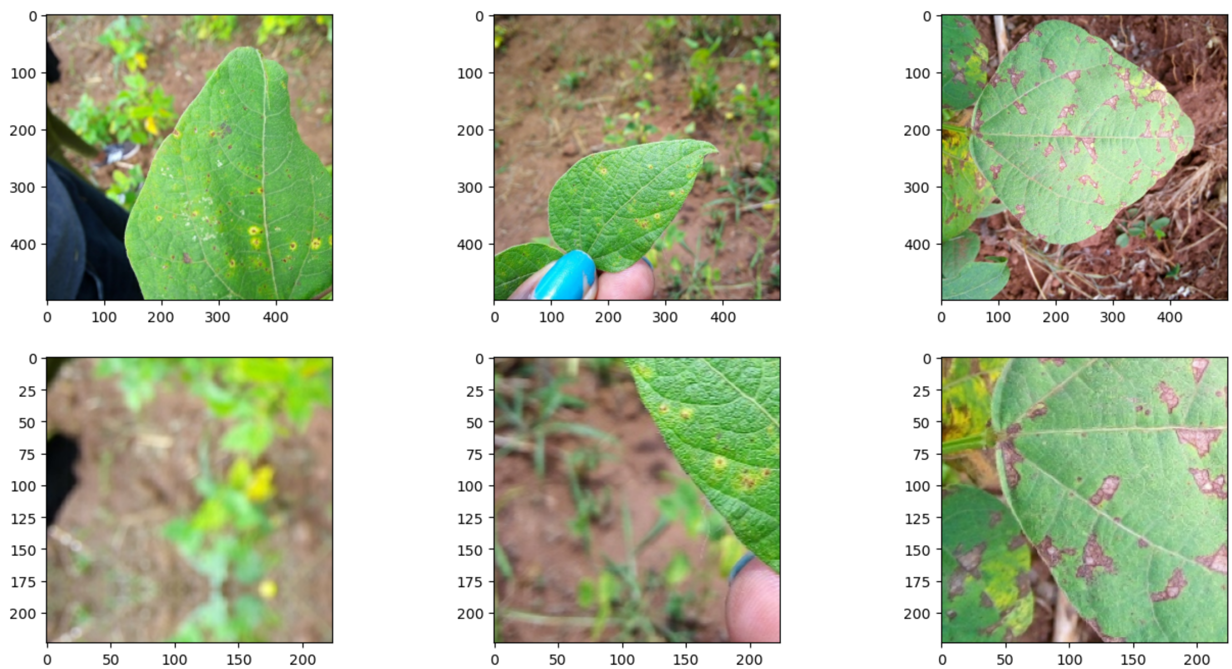
```
Metal device set to: Apple M1 Pro

2022-10-25 23:08:39.145795: I tensorflow/core/common_runtime/pluggable_de
vice/pluggable_device_factory.cc:305] Could not identify NUMA node of pla
tform GPU ID 0, defaulting to 0. Your kernel may not have been built with
NUMA support.
2022-10-25 23:08:39.145932: I tensorflow/core/common_runtime/pluggable_de
vice/pluggable_device_factory.cc:271] Created TensorFlow device (/job:loc
alhost/replica:0/task:0/device:GPU:0 with 0 MB memory) -> physical Plugga
bleDevice (device: 0, name: METAL, pci bus id: <undefined>)
2022-10-25 23:08:39.189320: W tensorflow/core/platform/profile_utils/cpu_
utils.cc:128] Failed to get CPU frequency: 0 Hz
```

In [11]:
```
X0, Y0, X1, Y1, D0, D1, D_info = data
```

I used RandomTranslation in case that the accuracy has increased.

In [64]:

```python
import explore_model #a new explore model.py

## You can use any list of 10 indices
sample_image_indices = [0, 1, 2]
sample_images = tf.cast(tf.gather(X0, sample_image_indices), tf.float32)
sample_labels = tf.gather(Y0, sample_image_indices)



args = explore_model.get_default_CNN_model()

preprocessed_images = args.model.input_prep_fn(sample_images)
augmented_images = args.model.augment_fn(sample_images)

fig, ax = plt.subplots(2, 3)
fig.set_size_inches(16, 8)
for i in range(3):
    ax[0][i].imshow(sample_images[i]/255., cmap = "Greys")
    #ax[1][i].imshow(preprocessed_images[i], cmap = "Greys")
    ax[1][i].imshow(augmented_images[i]/255, cmap = "Greys")
```



In [56]:
```
def run_task(data, task, subtask="all", epochs=None, batch_size=None):

    ## Retrieve data from tuple
    X0, Y0, X1, Y1, D0, D1, D_info = data

    # Training model
    print("Starting Model Training")
    history = args.model.fit(
        X0, Y0,
        epochs          = epochs,
        batch_size      = batch_size,
        validation_data = (X1, Y1),
    )

    return args.model
```

In [61]:
```
cnn_model = run_task(data, 3, epochs=6, batch_size=64)
```

```
Starting Model Training
Epoch 1/6
9/9 [==============================] - 5s 561ms/step - loss: 0.8236 - cat
egorical_accuracy: 0.6673 - val_loss: 0.7829 - val_categorical_accuracy:
0.7188
Epoch 2/6
9/9 [==============================] - 4s 490ms/step - loss: 0.7631 - cat
egorical_accuracy: 0.6925 - val_loss: 0.7340 - val_categorical_accuracy:
0.6953
Epoch 3/6
9/9 [==============================] - 4s 484ms/step - loss: 0.8470 - cat
egorical_accuracy: 0.6731 - val_loss: 0.6658 - val_categorical_accuracy:
0.7188
Epoch 4/6
9/9 [==============================] - 4s 482ms/step - loss: 0.7123 - cat
egorical_accuracy: 0.7099 - val_loss: 0.6973 - val_categorical_accuracy:
0.7188
Epoch 5/6
9/9 [==============================] - 4s 482ms/step - loss: 0.7261 - cat
egorical_accuracy: 0.7021 - val_loss: 0.8500 - val_categorical_accuracy:
0.6875
Epoch 6/6
9/9 [==============================] - 4s 479ms/step - loss: 0.7239 - cat
egorical_accuracy: 0.7099 - val_loss: 0.7230 - val_categorical_accuracy:
0.7188
```