Assignment 1:

In this assignment, we construct **dense neural networks** for classifying images from the MNIST dataset: http://yann.lecun.com/exdb/mnist/. The MNIST database consists of a set of 70,000 small (28x28 pixel) grayscale images of digits handwritten by high school students and employees of the US Census Bureau. Each image is labeled with the digit it represents.

There are two main goals for this assignment:

(1) To get experience building and testing `dense neural network` models using the Keras module in TensorFlow (tf.Keras).
(2) To also explore what the node(s) in the hidden layer of our models are "detecting" and what their outputs (i.e. `activation values`) contribute to the final classification of an image.

For the first goal, we use tf.Keras to load the MNIST dataset and split it into training and testing sets (of 60,000 and 10,000 images, respectively). We also use tf.Keras to build the models which we then train and test on the MNIST data. We try to optimize our model by experimenting with different hyperparameters such as the number of layers (say one or two), the number of nodes in each layer, the optimizer used, etc. in order to find the "best" `dense neural network` model for the data.

For the second goal, we want to find examples of intrinsic `global explanations` and `local explanations` , respectively, as described in this recent survey article:

https://cacm.acm.org/magazines/2020/1/241703-techniques-for-interpretable-machine-learning/fulltext

In the sample code provided, we illustrate the process of loading and analyzing the MNIST data, creating a dense sequential model, and training and testing on the MNIST data we loaded. We display the training and validation loss (resp. accuracy) for each (training) epoch side by side and we use confusion matrices to get insights on which of the MNIST digits were misclassified and why.

We also explore the correlation between the outputs (activation values) of the hidden nodes, thought of as features, and the classification classes. Since our model consists of one layer of 128 hidden nodes, there are 128 features (activation values) for each training image that determine the classification class. We can similarly think of each of the 784 (=28x28) pixels of each image as a features that determine the classification class as well. We perform the following experiments on each of these sets of features:

1) For a fixed hidden node, we look at the 60,000 `activation values` produced by the 60,000 training images. We group these values by their corresponding predicted classes—the digits 0 to 9—and visualize this grouping using a `boxplot`. Do you see much overlap in the range of values in each grouping? What if there was only one node? Would you see much overlap then?

2) We then focus on two of the pixel locations (near the center of the images). We look at the values of the two pixels for each of the 60,000 training images to see how well they predict the image classes. We plot each pair of pixels in a scatter plot using different colors for each of the 10 predicted classes.
3) We use PCA decomposition to reduce the number of dimensions from 784 pixel dimensions to 2 dimensions. We look at the two PCA values for each of the 60,000 training images to see how well they predict the image classes. We plot each pair of values in a scatter plot using different colors for each of the 10 predicted classes.
4) We also apply PCA decomposition to the `activation values` of the 128 hidden nodes to reduce the number of dimensions from 128 to 2 once again. We look at the two PCA values for each of the 60,000 training image to see how well they predict the image classes. We plot each pair of values in a scatter plot using different colors for each of the 10 predicted classes.
5) We repeat last experiment but using t-Distributed Stochastic Neighbor Embedding (t-SNE) to reduce the number of dimensions in instead.
6) Finally, we use a Random Forest classifier to get the relative importance of the 784 features (pixels) of the training and select the top 70 features (pixels). We select those 70 pixels from each of the training and images. How well those our model perform on these new data sets?

Your goal is to continue this line of exploration and visualization by varying both the architecture of the models, as described above, and the visualization experiments you perform on your models. In your report you want to make sure to clearly state both the results of your experiments and your interpretations of the results. Be creative!

7)

training and testing it on the data we loaded. We also use visual the performance of

In **EXPERIMENT 4** we use PCA decomposition to reduce the number of dimensions of our training set of 28x28 dimensional MNIST images from 784 to 154 (with 95% of training images variance lying along these components). We also reduce the number of dimensions of 'best' model from `Experiment 3` to 154 inputs nodes and train it on the new lower dimensional data.

In **EXPERIMENT 5** we use a Random Forest classifier to get the relative importance of the 784 features (pixels) of the 28x28 dimensional images in training set of MNIST images and select the top 70 features (pixels). We train our 'best' `dense neural network` using these 70 features and compare its performance to the the `dense neural network` models from EXPERIMENTS 3 and 4.

Here are more details for the first `three` experiments:

- **EXPERIMENT 1**: Our `dense neural network` will consist of 784 input nodes, a hidden layer with `1 node` and 10 output nodes (corresponding to the 10 digits). We use `mnist.load_data()` to get the 70,000 images divided into a set of 60,000 training images and 10,000 test images. We hold back 5,000 of the 60,000 training images for validation. After training the model, we group the 60,000 `activation values` of the hidden node for the (original) set of training images by the 10 predicted classes and visualize these sets of values using a `boxplot`. We expect the overlap between the range of values in the "boxes" to be minimal. In addition, we find the pattern that maximally activates the hidden node as a "warm up" exercise for similar analysis we will perform on `CNN` models in `Assignment 2`.
- **EXPERIMENT 2**: This time our `dense neural network` will have 784 input nodes, a hidden layer with `2 nodes` and 10 output nodes (corresponding to the 10 digits). For each of the 60,000 images, the output of the two hidden nodes are plotted using a `scatterplot`. We color code the points according to which of the 10 classes the the output of the two nodes predicts. Ideally, just like in `EXPERIMENT 1`, the color clusters should have very little overlap.

NOTE: For EXPERIMENTS 1 & 2 we also perform the following additional tasks:

1. We use Matplotlib to create 2 plots--displaying the training and validation loss (resp. accuracy) for each (training) epoch side by side.
2. For each model we obtain the confusion matrix and use it to display sample images of true vs false positives and negatives.

- **EXPERIMENT 3**: Students can experiment with more hidden nodes (in the hidden layer) to obtain the `best` model. This `final` model will be used in EXPERIMENTS 4 & 5.

References: https://github.com/fchollet/deep-learning-with-python-notebooks (2.1 & 5.4)