In order to train the model, follow the following procedure:

- First import MNIST dataset and USPS dataset and resize them to 32x32x3 to make them compatible with the model.
- Import *Model* from DALNModel.py and initialize it.
- Import *train* from DALNtrain.py and initialize a train object (trainer) with the arguments *X_source, y_source, model*(model object), *batch_size, X_target, y_target, epochs, source_only*(boolean).
- Setting *source_only* argument as True sets the trainer to train the model without domain adaptation and setting it as False sets the trainer to train the model with domain adaptation. *X_target, y_target* are optional arguments depending upon your application. For example, if you are training the model without domain adaptation, you don't need to give *X_target* and *y_target.*
- The training process will display source accuracy and target accuracy or source accuracy only depending on if you've provided *X_target* and *y_target* or not.
- In order to predict labels from the model, you can use *predict_label* method of model object.

One point to note is that the model is designed in such a way that equal number of samples of both source and target data is required. So, USPS data has to be stacked to get equal number of samples as MNIST dataset.

Calculating accuracy:

- Since the training process will display source and target accuracy, it is easy to keep track of it.
- If you still want to calculate accuracy by yourself, the first method is of-course predicting the labels using the model and then using the *accuracy_score* function.
- You can also import *display_logs* from DisplayLogs.py if you don't want to manually calculating accuracies for source and target data. You can initialize *display_logs* object by giving arguments such as *X_source, y_source, model*(model object), *X_target, y_target, epochs*(only for logging purpose), and *source_only*(boolean). Then you can call *accuracy* method of *display_logs* object that returns the dictionary having source accuracy and target accuracy or only source accuracy depending on the arguments provided.
- There are also *plot_tsne* and *plot_pca* methods for plotting the t-SNE, and PCA projections of the features.

Calculating determinacy and diversity:

- For calculating determinacy and diversity, one can simply make use of the above methods to find out what the determinacy and diversity as explained in the paper.

Originality of the codes:

- The codes in nwd.py and grl.py has been adapted from the repository by the authors of the original paper on DALN. The code provided was written for PyTorch and has been adapted for TensorFlow in this project. The link for the paper and the github repository is given below:
  https://openaccess.thecvf.com/content/CVPR2022/papers/Chen_Reusing_the_Task-Specific_Classifier_as_a_Discriminator_Discriminator-Free_Adversarial_Domain_CVPR_2022_paper.pdf
  https://github.com/xiaoachen98/DALN
- The architecture of the model has been fine-tuned based on several architectures adapted for several purposes. The architecture was made to be simple in order to cater for the low computational availability for the project.
- The rest of the codes have been created by the author.

Datasets

MNIST and USPS dataset have been used for this project. They both consist of a set of handwritten digits from 0 to 9. They are of small sizes and have enough samples to carry out this experiment. Also, they have considerable differences in the samples, that makes them suited for this project. MNIST dataset was imported from tensorflow.keras.datasets.mnist and USPS dataset was imported from extra_keras.usps.