Inspect Data Notebook:

In this notebook, we looked at the data images and labels to get to know them better and see their special characteristics. We used the nibabel library to load given files and numpy, pandas, and other Python libraries for our work here. We have already created a CSV file for labels from the Word file. We can easily see that the dataset, which has 4 classes, is completely balanced and has 10 data samples for each class. So we don't need to use approaches and methods for unbalanced datasets. The only issue here is that classes start from 1 instead of zero, so for some deep learning methods, we may need to adjust them.

Our data inspection was thorough and meticulous. We examined rCBF and DAT images for the training dataset and even created animations to enhance our understanding. We also comprehensively analyzed the minimum and maximum values in all volumes of the dataset, plotting a histogram for visual clarity. This process was repeated for the test dataset, ensuring that the orientations and shapes of the data matched. We noted a difference in the max number, leading us to decide against dividing intensities by the max number to avoid potential preprocessing errors. A simple normalization was deemed sufficient.

We also looked at the VOI template and checked its shape and orientation. We plotted it and even made an interactive animation to examine structures in the brain template. We counted 53 structures labeled for us in this template. Some of them have one voxel, and some have more than a thousand.

SVM using VoI Notebook:

The idea behind this method is to use the given VoI template for feature engineering; then, we can use these generated features to train an SVM classifier. We chose the SVM classifier because we only have 40 training data and it is so little that we can't train and evaluate a good deep model. However, in this situation, the SVM model will perform better because of its nature; as we will see in a bit, we achieved 100 percent accuracy in our experiments. We were able to generate features for data based on the given VoI template because you already made all images and the VoI template spatially normalized (MNI space). So, based on the template, we extracted mean values for each region in the brain.

First, we loaded NIfTI files using the nibabel library. Then, using the VoI template, we calculated regional means for each brain structure. We obtained 106 features per subject by analyzing both rCBF and DAT images of the 53 brain structures. If the images were not in MNI space, we could use brain segmentation models to predict structures for each image and then calculate the mean for each brain segment. There are pre-trained models that can segment the brain for us, but we didn't need them here.

Because each structure has different sizes and, accordingly, different numbers of corresponding voxels (from 1 to a few thousand), there could be wrong information in these features. Also, because the mean values could be very different from each other, the importance of each feature can be different. We decided to normalize each feature based on its own mean and standard deviation to bring their values closer together. It is good to note that mean and std should be calculated only on data on which we want to train our model, not validation or test.

We split our training dataset to train and validation as 80 percent would be the training set. In order to test the effect of feature normalization we trained and tested two models. We got 75 percent accuracy without normalization, but when we used normalized features, we got 100 percent accuracy. We could remove some features, like the ones with less than 5 voxels because they are small parts of brains, and values could be very different and create mistakes, but since we already got maximum accuracy, we couldn't test this theory to see the effect.

Now that we have found the best solution, we can train an SVM classifier on all of our training data and make predictions for test data. We loaded test files like before, then used our function to generate features based on the VoI template for both train and test datasets. Then, we calculated the mean and std for the training dataset and normalized both datasets using them. Using the training dataset, we trained an SVM model and predicted labels for the test dataset. The results were then saved to a CSV file.

The predicted labels corresponding to this method are placed under the column named "SVM_VoI", in the provided Excel file.

ResNet with GradCAM Notebook:

We decided to use a deep method for classification. ResNet is one of the best and most powerful models for feature extraction and classification. The original ResNet was designed for 2D images (with 3 color channels), so we had to design our own 3D Residual Network (ResNet). We put rCBF and DAT images together like image channels. The model we designed can process 3D data with 4 dimensions: 3 for volume and one for channels. Channels have to be 2.

We split our dataset into training, validation, and test sets, with 80 percent for training, 10 percent for validation, and 10 percent for testing. We normalized our data using the mean and standard deviation of the train set. We used data augmentation for the training dataset to make our model more robust and generalizable. Suitable data augmentation methods for brain images are random rotation, translation, and scale.

We trained our model using an SGD optimizer and CyclicLR Learning rate scheduler. We plotted training and validation losses for better understanding. We used validation loss to save the best model during training. Our best model got 100 percent accuracy on the test set. Because we had a small amount of data, the model may not be this good on your test dataset, especially since we only had 4 subjects as test data, one for each class. This is a problem with deep learning models. We used this model to make predictions on your test subjects, and the results are available in the provided Excel file under the column “ResNet”.

For this model, we also wanted to see which parts of the input regions contributed more to the final prediction. We used the idea of Grad-CAM paper, one of the best approaches in Explainable AI. Grad-CAM (Gradient-weighted Class Activation Mapping) is a deep learning technique for visualizing the regions of an image that contribute most to a model's prediction. It generates heatmaps by leveraging the gradients of the target class with respect to the desired convolutional layer of the neural network. This approach was also on 2D images, so we modified it and designed our own 3D GradCAM. The heat is upscaled and it shows which parts of the input layer had the most effect on predicting the output class. It is different class by class and subject by subject. In Figures 1 you can see heatmaps of different train and test subjects belonging to class 1.

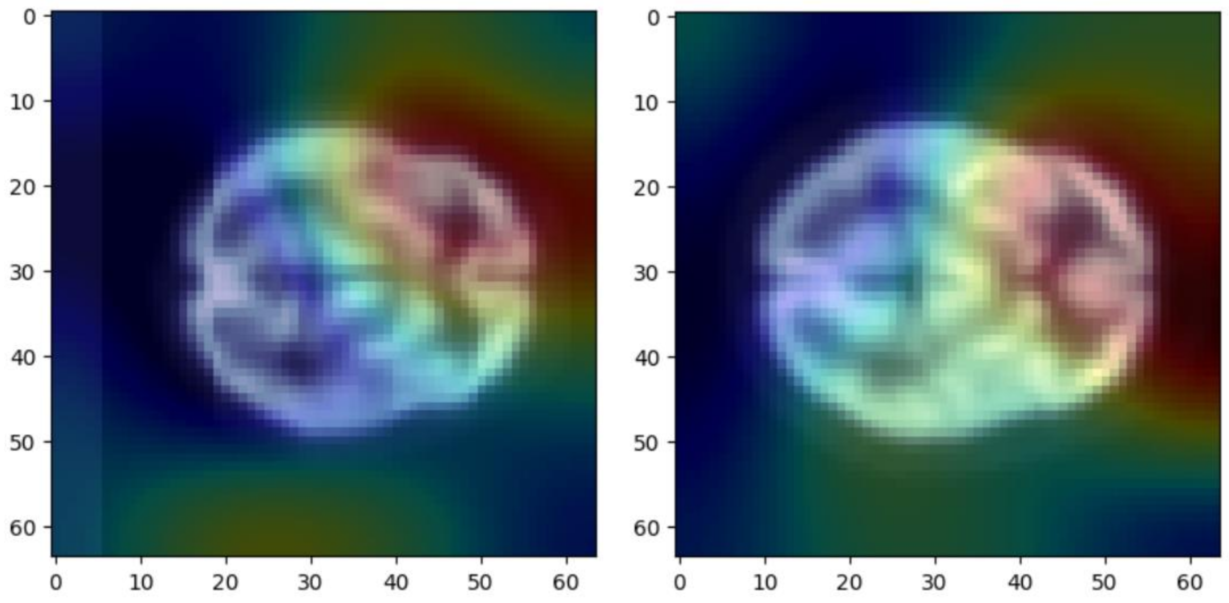

Figure 1 Heatmaps of Train and Test samples belonging to Class 1

A solution for making our model better when we have a small amount of data is transfer learning. We can use another dataset with a similar context, for example, an MRI brain dataset for tumor detection or segmentation, and train our model first on that dataset. Then, we fine-tune the model on our own dataset and for our own purpose (the last layers will be trained from scratch). Similarly, we can use a pre-trained model and change the last layers for our task. Anyway, even if we found such a dataset, we didn’t have enough time and resources to test this approach. Also, even if we did it, we couldn’t tell the effect of using transfer learning because we had already achieved 100 percent accuracy.

Branch ResNet Notebook:

We used the same idea of the ResNet model again. This time, instead of concatenating the rCBF and DAT images at the beginning of the model, we decided to have two different branches of ResNet that work on each image separately. We will concatenate the features from the last convolutional layer (512 features) and then use three fully connected layers for classification. This is an approach for combining information from multiple sources or modalities which is called “late fusion”. In this model, we used the same normalization and augmentation as before.

We split our dataset to train, validation, and test sets like we did before. We used the idea of cross validation for testing two optimizers. We trained our model with SGD and then ADAM optimizer, both with a learning rate scheduler. Using the validation set, we selected the best model for each experiment, and we also observed that SGD had better results. This model gave us 100 percent accuracy on the test set, but again, because we have only four subjects for testing, our model may not work this well for your test subjects.

The predicted labels corresponding to this method are placed under the column “Branch_ResNet”, in the provided Excel file.

SVM_ResNet Notebook:

Because SVM works very well on small datasets, we decided to have another one without using VoIs. For feature extraction, we used the ResNet model that we had already trained. We extracted the features with the ResNet model for all training data. Then, we split it into train and validation sets with a ratio of 80 percent for train and 20 percent for validation. We trained an SVM classifier on the training set and evaluated our model using validation; we got 100 percent accuracy.

To make sure that we were taking the correct approach, we used k_fold cross-validation. In this method, we divided our 40 training subjects into 5 folds of 8, each containing 2 subjects for validation per class. Each time, we used 1 fold for validation and the rest for training. We trained 5 SVM models and, each time, got 100 percent accuracy on the validation fold.

Using the idea of ensemble learning, we used 5 SVMs that were trained in k-fold experiments to predict labels for the test subjects. Then we used a majority voting system to predict the final results.

In another approach, we extracted features from test subjects using the ResNet model. Then, we used the whole training data to train the SVM classifier and predicted labels for the test dataset with it. Surprisingly, the results were exactly the same.

The predicted labels corresponding to this method are placed under the column named “SVM_ResNet”, in the provided Excel file.