

# CS598 - Deep Learning for Healthcare

## Final Report: Reproducing Alzheimer's Disease Classification

Tim Crosling  
Bhavuk Jain

University of Illinois Urbana-Champaign  
tgc3@illinois.edu, bhavukj2@illinois.edu

### Abstract

This report details the approach we took to reproduce the paper along with an evaluation of the results of our reproduced code. This also includes an extension to assess three different pipeline model approaches: 1) CNN only, 2) CNN with GNN, 3) CNN with GNN including persistent homography features. Finally, our reproduction also extends the original concept by wrapping our code in a pyHealth code hosted on GitHub and requested to fold into the core pyHealth library. We reach similar conclusions to the paper, notably that CNN with GNN is sufficient to identify and classify Alzheimer's Disease on standard high resolution MRI images without the need for more computationally intensive processing such as Topological Data Analysis (TDA).

**Paper Selected:** "Back to the basics with inclusion of clinical domain knowledge - A simple, scalable and effective model of Alzheimer's Disease classification"; Bruning S., Hensel F., Lukas L., Kuijs M., Jutzeler C., Rieck B.; Proceedings of Machine Learning Research; 2021. (1)

## 1. INTRODUCTION

Around 1 in 9 people (11.3%) aged 65+ in the US have Alzheimer's Disease (AD). MRI image diagnosis with existing machine learning models relies on complex architectures that are hard to interpret and costly to scale - limiting clinical availability. Down-sampling data has been considered to increase availability - but risks loss of key features.

We reproduced the paper "Back to the basics with inclusion of clinical domain knowledge - A simple, scalable and effective model of Alzheimer's Disease classification" by Bruning et al:

1. Applying a 3D Convolutional Neural Network (CNN) on a subset of full-resolution MRI images to measure brain tissue atrophy, particularly in the hippocampus,
2. Employing Topological Data Analysis (TDA) to assess anatomical brain areas and measure loss of whole-brain tissue connectivity

3. Extension: Comparing Grad-CAM visualization for the CNN vs. GNN embeddings to understand how the different models assessed the MRI images
4. Extension: Comparison of 3 different pipeline models (CNN only, CNN+GNN, CNN+GNN with persistent homography features)
5. Extension: Expose our code as a pyHealth task for future reuse

Our implementation is available on [HERE](https://github.com/bozlingo/cs598-DLH-Project) <https://github.com/bozlingo/cs598-DLH-Project> with source data available on request (due to ADNI patient confidentiality requirements). We tested our reproduction on a subset of 20 Alzheimer's positive and 20 clinically normal MRI subjects (downscaled from the 358 used in the paper for practicality reasons).

Implementation note: While the paper source code was provided (1) we felt that there would be more learning in reproducing from scratch. The description below is 100% our own reproduction albeit heavily leaning on chatGPT to support implementation.

A 4 minute video summary of our reproduction is available at <https://github.com/bozlingo/cs598-DLH-Project/blob/main/Project>

## 2. SOURCE DATA

Image data was sourced from the Alzheimer's Disease Neuroimaging Institute (ADNI) study Image and Data Archive (IDA) which is managed by the University of Southern California. MRI images were downloaded for 20 Alzheimer's Diagnosed (AD) and 20 Clinically Normal (CN) patients according to the following criteria:

- ADNI 1, ADNI2, ADNI GO, ADNI 3 studies
- Subject Age: 77 - 78 years
- Modality: T-1 weighted MRI
- Diagnosis: AD and CN subjects
- Original (unprocessed file types)
- Image type: MPRAGE (high resolution T1 weighted structural image); used for brain volume, segmentation and atrophy detection.

Image Type	Purpose	Usage in ADNI	Key Features
<b>MPRAGE</b>	High resolution structural MRI	Brain volume, segmentation, atrophy detection	3d T-1 weighted, strong grey/white matter contrast
<b>Localizer / 3-Plane Localizer</b>	Quick positioning scan	Aligns patient for later scans	Fast, low resolution, 3 plane acquisition
<b>B1-Calibration Head / Body</b>	RF field uniformity correction	Optimizes image quality and signal homogeneity	Measures B1-field inhomogeneities
<b>Double_TSE</b>	T2-weighted image	White matter hyperintensities, lesions	High SNR, pathology detection
<b>Circle Scout</b>	Positioning scan	Confirms field of view	Circular FOV, low resolution
<b>Axial PD-T2 TSE</b>	Proton density and T2 contrast	White matter integrity, lesion detection	PD and T2 contrast, axial slices

Figure 1: ADNI MRI Image types - MPRAGE was selected

Images were manually filtered using Description notes as we found that there was low adherence to the image type definitions below. Images with description MPRAGE or MPRAGE-REPEAT only were selected.

### 3. GENERATING CODE FOR DATA PROCESSING

#### (3.1) Processing MRI images

Images downloaded from IDA were initially in ADNI DICOM (DCM) format. To allow input to clinica, the images were reformatted into NIfTI format using the dcm2niix tool:

```
1 pip install nibabel
2 dcm2niix -z y -o /path/to/output/
  -f %p_%s /path/to/dicom/
```

Resulting .nii.gz formatted files were moved into a single "ADNI.Input" folder along with the labelled ADNI\_output.csv file. These files were then opened sequentially using the nibabel (NeuroImaging in Python) library. Each image was then intensity normalized into a range [0, 1] and then center cropped into shape (120, 144, 120).

These cropped images were then broken into 4 non-overlapping 3D patches of size (30, 36, 30) representing different segments of the cropped images. These patches are used as input to the 3D CNN model (model 1) to identify areas of lower voxel intensity corresponding to areas of the brain that atrophy with Alzheimer's Disease.

#### (3.2) Left and Right Hippocampus Labelling

Alzheimer's Disease (AD) is known to affect the left hippocampus to a greater degree with asymmetrical atrophy of the left vs. right hippocampus being a known indicator typical AD and limbic-predominant AD (two other AD sub-types do not exhibit this trend). To provide this view we developed a left vs. right hippocampus labelling for each normalized and cropped image using the Cerebrum Atlas Mask from the McConnell Brain Imaging Center.

We sourced the symmetric MNI-ICBM152 template was used which provided cortical and subcortical labels in NIFTI format (10). This mask was provided in original MNI space with shape (394, 466, 378), so we used scipy.ndimage.zoom to sample down to (120, 144, 120) space which aligned with

our normalized cropped images. Simply multiplying the atlas mask by the normalized cropped images provided a left vs. right hippocampus label within each image.

#### (3.3) Persistent Homography Features for Topological Data Analysis (TDA)

Topological Data Analysis (TDA) is used by the paper for model 2. In this approach, we employ a mathematical model to extract a graph-analog structure from each of the normalized, cropped MRI images. In this approach, we select only voxels with a minimal intensity level  $R$ , allowing low intensity 'noise' voxels to be filtered and the remaining 'high intensity' voxels to be intensity scaled across the full [0, 1] range.

Topological features are then identified using a three dimensional vector where  $d = 0$  represents 'connected components',  $d = 1$  represents 'cycles or tunnels' and  $d = 2$  represents 'voids'. By looking at changes in these features as the intensity level  $R$  changes, the model is able to identify 'persistent homology features'. This method allows us to detect latent structures in the brain with a hypothesis that looking at changes in atrophy in these structures using GNN will identify deeper underlying indications of Alzheimer's Disease.

To develop the persistent homography features we first flattened the normalized cropped images to 1D with `np.flatten()`. These images were then loaded into the python Gudhi (Geometry Understanding in Higher Dimensions) library, creating a complex cubical form of the image. The paper refers to DIPHA to compute persistent homology; but we found that this was incompatible with Google Colab (must be run as a command line utility) and so used the equivalent `Gudhi.persistance()` method to extract persistent homography features.

### 4. GENERATING CODE FOR ALGORITHM IMPLEMENTATIONS

For model training, we split the input data into training and test sets using `sklearn.stratifiedKFold` with `n_splits = 2` and `random_state=shuffling`. Using these splits we executed two sequences

#### (4.1) 3D CNN model development (for cropped, normalized image patches)

Model 1 of the paper involves a standard 3D Convolutional Neural Network (CNN) applied to the (30, 36, 30) shape image patches established in section 3.1. Because full 3D MRI scans can be large and noisy, we pass the patches into the 3D CNN individually to extract deep embeddings and then train a logistic regression function on these patch embeddings.

To build the 3D CNN we established a `build_3d_cnn` function which is then trained over 50 epochs with batch size=8 to generate patch embeddings for both trainset and testsets:

- Layer 1: tensorflow.keras.layers.Conv3D with 32 filters on a (3, 3, 3) 3D kernel size and initialized using HeUniform for weight initialization. Followed by batch normalization, ReLu activation and then 30
- Layer 2: tensorflow.keras.layers.Conv3D with 64 filters on a (3, 3, 3) 3D kernel size and initialized, batch normalized as above. ReLu activation and 30
- Layer 3: tensorflow.keras.layers.Conv3D with 128 filters on (3, 3, 3) 3D kernel size and initialized, batch normalized as above. ReLu activation and 30
- GlobalAveragePooling3D followed by a fully connected layer with 128 neurons and ReLu activation and 40
- The model is then wrapped into a Keras functional API model with Adam optimizer (adaptive learning rate), Binary cross-entropy loss (ideal for binary classification) and AUC as the evaluation metric

To apply logistic regression on the patch embeddings we create a sklearn LogisticRegression model with balanced class weight and liblinear solver. We train the Logistic Regression with the patch embeddings from the trainset, and then use the test embeddings to develop a prediction.

#### (4.3) GNN model development

In section 3.3 we developed persistent homography features based on the TDA method with the hypothesis that this would provide more precision around atrophy within the underlying brain structures. To incorporate these features in our analysis we performed two additional steps:

1. Concatenate persistent homography features with the trained patch CNN encodings. This was performed with a simple np.concatenate call, applying a max tensor length of 5,000 elements for scalability.
2. Perform Global/Topological Context Modelling with a Graph Neural Network (GNN) to balance interactions within each patch (CNN) and interactions across patches (GNN). To create the GNN network we created GNNPatchAggregator class which established the network and allowed us to train over 50 epochs:
  - Layer 1: torch\_geometric.nn.GCNConv using the CNN patch embeddings shape as input and outputting a 64 channel hidden layer with ReLu activation.
  - Layer 2: torch\_geometric.nn.GCNConv using the resulting hidden layer as input and outputting a 64 channel hidden layer
  - GlobalMaxPooling to input into a linear regression classifier using binary cross entropy loss function.

## 5. GENERATING CODE FOR EXTENSIONS

### (5.1) Adapting Grad-CAM to identify which brain areas affect the GNN decision

We asked ChatGPT to suggest potential extensions for the paper - and opted to support the following suggestion:

*Try adapting Grad-CAM or attention-based explanations to visualize which graph nodes (i.e., brain regions or patches) influenced the GNN decision the*

	3D CNN + Logistic Regression	3D CNN $\rightarrow$ GNN	CNN + Persistent Homography $\rightarrow$ GNN
Accuracy	0.57	0.43	0.43
AUC	0.67	0.66	0.54
Precision	-	0.43	0.43
Recall	-	1.0	1.0

Figure 2: Model performance comparison

*most. Compare CNN Grad-CAM vs. GNN attention: Do CNN-based saliency maps highlight the same regions as what the GNN ‘cares about’? This could be very revealing about the synergy between CNNs and GNNs.*

This suggestion led us to create three separate model outputs:

1. CNN + Logistic Regression: Classify AD based on local patterns within each image patch. i.e., visualize predictions per subject and overlay with ground truth
2. CNN  $\rightarrow$  GNN: Combine local prediction from each patch with overall pattern recognition across the entire image
3. CNN + Persistence  $\rightarrow$  GNN: Incorporating the TDA output to incorporating structural elements

As described in section 6.2, these models were visualized using Grad-Cam (model 1), GNNExplainer (models 2 and 3).

### (5.2) pyHealth Extension

As a further extension we wrapped our code into a pyHealth task to aid future development. To do this we first defined a custom ADNI dataset class for loading and processing ADNI data, including label extraction. We submitted this as a pyHealth pull request available via <https://github.com/sunlabuiuc/PyHealth/pull/353>.

We also completed a pyHealth task wrapper to run an end-to-end pipeline given inputs of ADNI image data, label csv and (optionally) hippocampus atlas mask. Details can be found on [https://github.com/bozlingo/cs598-DLH-Project/blob/main/CS598\\_DLHProject\\_code\\_Final.ipynb](https://github.com/bozlingo/cs598-DLH-Project/blob/main/CS598_DLHProject_code_Final.ipynb)

## 6. ANALYZING AND VISUALIZING REPRODUCED RESULTS

### (6.1) Evaluating Performance

The paper evaluated performance based on 358 subjects on a high performance GPU. For our purposes, we evaluated performance on a scaled down dataset of 20 positive (AD) and 20 negative (CN) sample MRI images. Code was executed on Google Colab (T4 GPU with 16 GB of VRAM with 1 hr limit) with source data hosted on our local Google Drive for fast data access. The code executed end to end in 52 minutes, 43 seconds.

Model training was undertaken using 4 splits of the trainset using random selection without replacement. Results of these splits are shown in Figure 2.

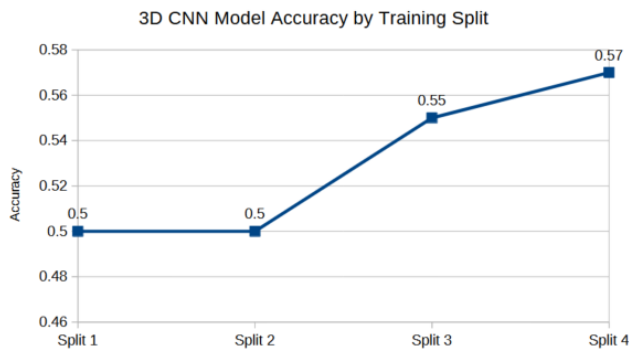


Figure 3: 3D CNN Model Training Accuracy Progression

Similar to the paper, we found the simplest 3D Convolutional Neural Network (3D CNN) model performed the best with 0.57 accuracy and 0.66 AUC. However, Precision and Recall measures were deemed unreliable due to likely lack of balanced AD positive samples in the splits.

Furthermore, we found that applying GNN to the CNN embeddings delivered reduced performance - suggesting that the classification is predominately local within each image patch (rather than requiring pattern recognition across multiple patches).

Similar to the paper, we also found the Topological Data Analysis generated lower results than the simple 3D CNN suggesting that this method overly simplified the topology of the brain structure and therefore did not detect underlying patterns that would improve diagnosis.

Overall, our results were directionally correct - but fell short of the 86% accuracy presented in the paper. We believe this is due primarily to the reduced training dataset (238 subjects with multiple images over 10 years vs. our reduced 38 subject trainset with a single MRI image per subject). This is illustrated in figure 3 the learning rate of the model - where we saw an increasing rate of accuracy once the model saw two splits of data ( 20 images) - but also illustrating that the gradient of improvement was yet to flatten.

We also believe that the model results can be improved by using the Atlas Cerebral Mask to separately train the left and right hemispheres - supporting the observation that Alzheimer's disease tends to affect the left hippocampus. This is within the capabilities of our code - but required additional computational time outside of the Colab 1 hour allowance and so we do not publish results here.

## (6.2) Visualizing results for Interpretability

As with the paper, we leveraged Grad-CAM to visualize our results. To do this we had to first get the last 3d Convolution layer of the model and flatten to 2D. We then computed a heatmap that could be inputted into a matplotlib pyplot. As shown in Figure 4 below we found clear markers of Alzheimer's in the MRI patches - with a hypothesis that these would become sharper with larger training set and potentially focused training on the left hemisphere images.

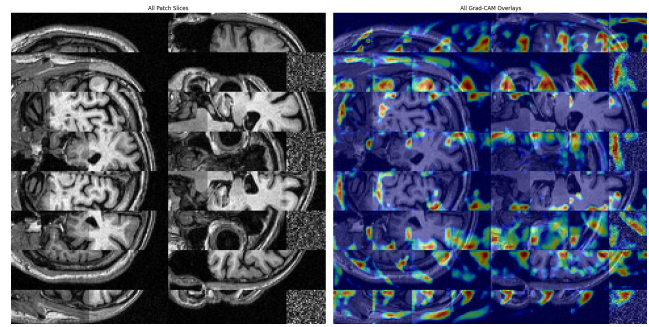


Figure 4: Original MRI image patches vs. GRAD-CAM categorization heatmap

## 7. CONCLUSION

While this project was challenging, we are happy with our paper selection on the basis that it tested multiple of the advanced concepts covered in this course. While Neuroimaging is certainly a complex field, we found a practical way to source real world MRI images from ADNI and many freely available tools to process and manipulate this data. ChatGPT was an invaluable guide in providing both initial source code generation and trouble shooting. However, we did also find that we needed to prompt ChatGPT on specific methodologies (e.g., we got into a multi-week black hole due to an initial prompt implementing the 3D CNN with a Sequence object, which was then incompatible with CAM-GRAD). From an output perspective, our project was limited by our decision to process only 40 subjects (10% of the volume covered by the paper). This was a necessary decision due to our resource constraints, but clearly has impacted the accuracy of our findings. However, our code modularity made it quite straightforward to implement our extensions and wrap the code in a pyHealth task - and we would suggest this is an improvement over the papers' authors who used a procedurally monolithic approach in their publicly available code. This is a rich topic and we see many potential extensions including: (1) Further visualization of the TDA structures detected to assess clinical applicability of the topological framework used (which would provide root understanding of the causes of Alzheimer's Disease), (2) Specialized extensions to provide diagnosis for other forms of Alzheimer's Disease which affect 14% of cases (e.g., hippocampal sparing subtype)

## References

- [1] Paper Source Code, [https://github.com/BorgwardtLab/ADNI3DCNN\\_vsT\\_DA](https://github.com/BorgwardtLab/ADNI3DCNN_vsT_DA)
- [2] ADNI, <https://adni.loni.usc.edu>
- [3] Clinica, <https://github.com/aramis-lab/clinica>
- [4] fMRIPrep, <https://fmriprep.org/en/stable/>
- [5] Cerebrum Atlas, <https://www.nature.com/articles/s41597-020-0557-9>
- [6] DIPHA, <https://github.com/DIPHA/dipha>
- [7] Persim, <https://persim.scikit-tda.org/en/latest/index.html>

- [8] GradCam and Guided GradCAM, [https://github.com/nguyenhoa93/GradCAM\\_and\\_GuidedGradCAM\\_tf2](https://github.com/nguyenhoa93/GradCAM_and_GuidedGradCAM_tf2)
- [9] Cerebrum Atlas Mask, McGill University, <https://www.bic.mni.mcgill.ca/ServicesAtlases/ICBM152NLin2009>
- [10] McConnell Brain Imaging Center Cerebrum Atlas Mask <https://www.bic.mni.mcgill.ca/ServicesAtlases/ICBM152NLin2009>
- [11] dcm2niix <https://github.com/rordenlab/dcm2niix>