Classification of MRI Images by Extent of Dementia

Jessica Hernandez & John Fortner

College of Computing, Georgia Institute of Technology

CS 4641: Machine Learning

Dr. Gerandy Britito

27 July, 2022

Introduction

With the advent of MRI technology, researchers are becoming better and better at understanding how the human brain functions, and what kinds of damage are associated with what physical symptoms. Using machine learning techniques, we intend to use MRI images of a variety of patients with different extents of dementia in order to train a classifier that can estimate the extent of dementia progression in a given patient using an MRI scan.

We will attempt to classify images into four categories, associated with one the following states: NotDemented, VeryMildDemented, MildDemented, or ModerateDemented. Using this classifier, we can predict, given an MRI scan, to what extent an individual is likely affected by Alzheimer's, making accurate prediction of onset and access to appropriate treatments easier to obtain.

Data Source

The dataset was found on Kaggle under the name "Alzheimer's Dataset (4 class of Images)". Sarvesh Dubey, a data engineer at Accenture AI, was the uploader of the dataset. The data was hand collected from various websites, with each label verified. Dubey states that the inspiration behind sharing the dataset was to "make a very highly accurate model predict the stages of Alzheimer's".

Data Format

Our data consists of 6400 MRI images of size 208 x 176. Although these images seemed to be in grayscale with one color channel at first glance, upon further inspection, the images had 3 channels, thus making the size of each image 176 x 208 x 3. This was discovered after multiple attempts of outputting the image, only to get an "image size" error because we were . To fix this, we modified the creation of the ImageFolder dataset to include a transform to grayscale. Cutting the channels of the images from 3 to 1 would in theory result in better runtime performance by reducing total computations involved in kernel applications during convolution.

The images are each associated with one of four classes of dementia: ModerateDemented, MildDemented, VeryMildDemented, and NotDemented. We downloaded the dataset directly from the Kaggle API, converted the training and test folders into ImageFolder datasets, and combined these datasets with a customized sampler to create DataLoader objects for training and testing.

Method Documentation

Our model was implemented using a convolutional neural network created using the pytorch library. The model has the following structure:

- Convolutional layer with one input channels, six output channels, and a kernel of size five
- ReLU activation
- Max Pooling layer with a square 2x2 window and a stride of two
- Convolutional layer with six input channels, fifteen output channels, and a kernel of size five
- ReLU activation

- Max Pooling layer with a square 2x2 window and a stride of two
- Flattening
- Dropout layer with dropout rate 0.8
- Linear layer with 30135 inputs and 2340 outputs
- ReLU activation
- Dropout layer with dropout rate 0.45
- Linear layer with 2340 inputs and 420 outputs
- ReLU activation
- Linear layer with 420 inputs and four outputs

We used the Adamax optimizer, common for image classification, and for our loss function, we used Cross Entropy Loss. Our learning rate was 0.003, and our batch size for both training and testing was 12. For training, we created a custom weighted random sampler that adjusts the probability of each image in the training dataset being sampled such that each class has an equal likelihood of being selected. For testing, we used a standard random sampler on the existing data.

Method Reasoning

Convolution

Our model includes two convolutional layers, each followed by max pooling. Convolution is a common technique used in image classification due to its effectiveness at identifying features in images (Sultana et al., 2019). We opted to use two convolutional layers, with six and 15 channels respectively.

We mimicked common Convolutional Neural Network paradigms, beginning with convolutional layers followed by pooling, and ending with a fully-connected linear component. This is a common strategy, with the idea that convolutional layers locate various significant features which linear layers then use to perform classifications. We believe that our model represents an informed, sensible approach to this image classification task.

Dropout layer

After running the dropping the batch size from 32 to 12, and decreasing the learning rate from 0.3 to 0.003, our model started to overfit. For example, for a run of 20 epochs, the training accuracy plateaued around 97% while the testing accuracy remained around 60%. This meant that our model was not able to generalize well from the training data to the new testing data; in other words, the model fit too closely to the training data and therefore was unable to accurately classify the degree of Alzheimer's in new images of MRI scans.

In order to fix this, we introduced dropout layers following two of our linear layers. Research has shown that while the inclusion of a dropout layer with a dropout probability as high as 0.5 can increase the time taken to converge, dropout can significantly decrease overfitting by reducing complex co-adaptations of neurons since a neuron cannot rely entirely on the presence of other individual neurons (Krizhevsky et al., 2012).

In our model, we introduced a dropout layer after each of the first two layers in the fully-connected linear component, each with a dropout rate of 0.5, similar to that used in the Krizhevsky paper. This change resulted in significant experimentation, and eventually led to a variety of different structurings of our dropout layers. We had significant issues obtaining good results from these layers due to our model's high instability. Overall, the dropout layers were either not effective or too effective depending on the assigned dropout rate across a variety of implementations and trials. However, our final implementation included two dropout layers with dropout rates 0.8 and 0.45, and although our model continued to overfit in later epochs, we did observe a small increase in testing accuracy.

Findings Recording

Our first attempt at the model resulted in a consistent 50% accuracy, regardless of the characteristics of the data or the number of epochs for which the model trained. This led us to change the optimizer from Adam to SGD in an attempt to find the problem. However, the optimizer was not the problem; the consistent 50% accuracy was a result of imbalance in the amount of data. Since half of the training images had no dementia, the model predicted that a patient had no dementia in every circumstance in order to achieve a reliable 50% accuracy. After investigation of the problem, this led us to implement the WeightedRandomSampler for the training data in order to equalize the proportion of each class trained on by the model.

After the implementation of the sampler, our model began to suffer from extremely low accuracy on both training and testing data. One way we were able to resolve this was by reducing our batch size from 32 to 12 and by lowering our learning rate from 0.03 to 0.003.

After instituting these changes, we were able to obtain very high training accuracy, peaking at 99% after 20 epochs, but with testing accuracy of only 60%. At this point, we decided that given the disparity in training and testing accuracy that we were experiencing overfitting. This was the biggest issue with our model.

To resolve the overfitting, we began to experiment with dropout layers. This led us to include two dropout layers in the fully-connected portion of the model, each with a dropout rate of 0.5. In addition, we switched the optimizer back from SGD to Adam. This optimizer was choses because of evidence suggesting that in multi-layer neural networks, Adam consistently outperforms other methods (Kingma & Ba, 2015). After these changes, we continued to experience significant overfitting.

After further investigation, we found research indicating that for a certain image classification problem trained with a variety of architectures and optimizers the Adamax optimizer consistently outperformed Adam at moderate learning rates (Krizhevsky et al., 2012). With these results in mind, we experimented with introducing the Adamax optimizer, and observed few differences. Additionally, we continued to observe overfitting at later epochs, even with dropout rates as high as 0.5. To combat this, we tried further increasing our dropout rates to 0.7, which also failed to combat overfitting. At this point, we checked to see whether our dropout layers were correctly implemented with some sanity checks, which seemed to indicate that the layers were indeed implemented correctly. Finally, we attempted to use a dropout rate of 0.9, which failed to converge to a solution.

Next, we decided to change where our dropout layers occured; instead of taking place on the second two layers, we decided to implement dropout immediately after the flattening,

and after the first linear layer. Our first draft of this strategy had a dropout rate of 0.6. We tried several different values in this model, but always obtained either overfitting or a model that failed to converge at all.

Other things we tried were adding a higher dropout rate prior to the first linear layer, and a smaller dropout rate after the layer. We tried this with a variety of values, with every choice either leading to severe overfitting or lack of convergence.

Finally, we decided that our model was too unstable to handle significant dropout. Continued improvement would require expansion of our architecture. Here, we decided to expand the number of nodes of our last two linear layers, to 2340 and 420 nodes respectively. We kept a dropout rate of 0.8 prior to the larger layer, and maintained a dropout rate of 0.6 after the layer. This came with the cost of significantly increased runtime. After some experimentation and some failure to converge, we reduced the second dropout rate to 0.45. This resulted in the following metrics.

```
epoch number 22 complete
Training accuracy: 5121/5121 (100%)

Testing accuracy: 829/1279 (65%)


epoch number 23 complete
Training accuracy: 5120/5121 (100%)

Testing accuracy: 821/1279 (64%)


epoch number 24 complete
Training accuracy: 5121/5121 (100%)

Testing accuracy: 849/1279 (66%)


epoch number 25 complete
Training accuracy: 5121/5121 (100%)

Testing accuracy: 826/1279 (65%)


epoch number 26 complete
Training accuracy: 5121/5121 (100%)

Testing accuracy: 825/1279 (65%)
```

Our final, best results consisted of:

- Model with two Conv2d layers with Max Pooling, two dropout layers with 0.8 and 0.45 dropout rates respectively, and Adamax optimizer
- Training accuracy of 100%
- Testing accuracy of 65%
- Runtime of 4 hours 53 minutes across 26 epochs

Findings Analysis

Our final testing accuracy was 65%. One attempt online was able to achieve a 95% testing accuracy using different neural architectures. However, the convolutional neural network that was used in the paper –VGG16– was 16 layers deep and the model was left

running over 4 weeks over 4 GPUs. Given these conditions, our accuracy seems reasonable in comparison.

One thing that stuck out most in our results was the consistent struggle with model overfitting. Even with multiple dropout layers with high dropout rates, we continued to struggle with overfitting of data. Even in our final, best attempt, we continued to experience significant overfitting, with exceptionally high training accuracy and relatively low test accuracy. In the future, other methods that could be attempted to counteract this problem would be acquiring access to more data, or using data augmentation techniques to artificially increase the size of the data set and discourage memorization of the images. That said, we considered artificial data augmentation inappropriate for this task given the high level of standardization of the images and consistent method of collection.

Another key thing we found about our model was that it was highly unstable. Adding a little too much dropout could result in complete failure to converge, while a bit too little resulted in severe overfitting with at best 60% testing accuracy. We believe that adding larger layers in our linear components could have made the model more stable, at the cost of a much longer runtime, which was not feasible for this project. We did attempt increasing the size of the second two linear layers, but it began taking prohibitively long to train the model. However, with access to more time and computing power, this would be a natural approach that we believe could definitely improve results.

Future Work

To increase efficacy of the model, future work could entail the use of more complex neural networks. Currently, work is being done with this dataset in that direction by making use of highly sophisticated network architectures, higher computational power, and longer periods of training time. Use of these models has the potential to significantly increase the accuracy of classification on this set; these developments are even more significant given the sensitive medical nature of the data involved.

Furthermore, a highly accurate model trained on this dataset could be extremely important in real-world applications. A machine learning model with higher accuracy than human medical professionals could increase the likelihood that each patient gets the most appropriate treatment for them, as well as increasing the efficiency of delivering such diagnoses.This can result in making a higher quality of healthcare more accessible to a wider audience. Machine learning models in this vein, as well as in other medical fields where diagnoses are highly based on classification of images and scans, are extremely important tools with the potential to contribute significantly to the accuracy of medical diagnoses.

References

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting.. *Journal of Machine Learning Research*, 15, 1929-1958.

Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

F. Sultana, A. Sufian and P. Dutta, "Advancements in Image Classification using Convolutional Neural Network," *2018 Fourth International Conference on Research*

in Computational Intelligence and Communication Networks (ICRCICN), 2018, pp. 122-129, doi: 10.1109/ICRCICN.2018.8718718.