CS230

The benefits of transfer learning for predicting the presence and severity of dementia from cross-sectional MRIs

Constantine Athanitis
costa99@stanford.edu

Elizabeth Fitzgerald
elizfitz@stanford.edu

Abstract

Alzheimer's disease is a debilitating condition that affects millions of people every year. Although there is no cure, it's important to catch the condition early for the sake of the patient and their families. One way of doing so is to identify symptoms, such as the onset of dementia. We aim to identify the presence and severity of dementia from cross-sectional MRI images of the human brain. Within this context, we wanted to explore the benefits of transfer learning. We ran two separate investigations of transfer learning. In the first, we created both a standard neural network and a convolutional neural network (CNN) for binary classification. We also created a standard neural network and a CNN for multi-class classification. We then used transfer learning on the networks for binary classification and compared the results to those of the direct multi-class networks. We found that this transfer learning did in fact improve model performance. In the second investigation, we trained a ResNet network on our own data and compared the results to those of a pretrained ResNet network. We found that transfer learning again improved performance.

1 Introduction

For our final project, we want to create a model that can accurately predict not only whether a patient has dementia, but also the severity of their condition. Specifically, we want to create a binary neural network classifier that determines if a patient has dementia, as well as a multi-class neural network classifier that is able to categorize the severity of dementia in an afflicted patient, based only on cross-sectional MRI images of the patient's brain. This project is personally important, as Elizabeth's grandmother suffers from dementia and it affects her family deeply. Being able to catch signs of Alzheimer's early is very important for both the patient and their families, so this project hopes to create a system that accurately identifies when a patient has dementia. However, it primarily aims to explore the potential benefits of transfer learning. While prior research has attempted to use deep learning to solve this problem (such as studies done by Stamate et. al [1]), we are more focused on the benefits of transfer learning as a novel approach for the classification of symptoms.

2 Related work

Normally, the process of diagnosing dementia and Alzheimer's is done by medical professionals. Diagnoses are based on a variety of patient details, including MRI scans that are reviewed manually by a doctor. The task of using deep learning to identify dementia from MRIs is not a novel topic. However, our project is more focused on the benefits of transfer learning within this context. To construct the baseline binary CNN model, we used the same architecture as seen in the Lin paper

CS230: Deep Learning, Fall 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

[2]. We found that this approach, while a good starting point, did not work for our particular data. Section 4.1 briefly describes the process of improving upon this model. Another paper we looked at by Sergey Korolev and others was also tackling the problem of dementia classification with the ADNI data [3]. Their research did implement ResNet, although they chose to derive it from VoxResNet, which is an architecture trained for state-of-the-art performance on 2D image recognition tasks [4]. Another paper focused more of the use of CNNs with a combined dataset of the ADNI dataset and a separate dataset known colloquially as the "Milan" dataset [5]. Finally, we did an investigation of other research with transfer learning, as that is the primary goal of this project. For example, we found a paper entitled "Classification of Alzheimer Disease on Imaging Modalities with Deep CNNs Using Cross-Modal Transfer Learning" [6]. The researchers used a novel approach of fusing classifier results with majority vote to create more accurate, augmented scores. We find this to be a rather creative approach, even if we do not implement it ourselves. Lastly, we found a paper that used transfer learning to improve model performance on 3D images after being trained on 2D images [7]. We found this to be an interesting example of future work for our own project.

3 Dataset and Features

Our dataset is actually a combination of two datasets. The first is the OASIS 1 dataset [8]. While it is comprised of multiple images per patient, we have extracted only the unmasked cross-sectional MRIs from the dataset, leaving us with exactly 235 images, as seen in Figure A1. The second dataset comes from the ADNI database [9]. This data consists of 5,831 3D brain scans of patients with varying degrees of dementia. From these scans, we extract the 96th slice, as this was the closest to matching the longitudinal images from the OASIS data set, as seen in Figure A2. Once all the images were extracted, we then standardized them. First, we converted the images to a .jpg format, and then rotated them to be vertically-oriented. Next, we renamed each individual image by its corresponding patient ID, and then rescaled them to be 208 x 160 pixels (as this was the dimensionality of the smallest image from the full dataset). We finally split the processed images into train/dev/test sets with a split percentage of 70/15/15 using scikit-learn's built-in functionality [10]. These splits (as well as the examples in each) remained constant for every model we experimented on. The only data difference between models is the label: either a binary label or a multi-class label. The binary labels are 0 and 1, where 0 means the patient does not have dementia, and 1 indicates dementia. The multi-class labels are 0, 1, and 2. 0 means the patient does not have dementia, 1 indicates mild dementia, and 2 indicates moderate to severe dementia (see figure A9 for label frequencies). Finally, as a note, all of this data preprocessing was done through an efficient pipeline that we wrote that generates each of the above sets for both binary and multi-class classification. This pipeline parses through a given .csv file (for train, dev, or test set) and then creates numpy arrays from the images as well as their labels. With the pre-processing done, we created all eight of our models with the Keras framework [11].

4 Methods

4.1 Baseline Binary Models: Standard Neural Network and CNN

We first created Model 1, which is a standard neural network meant for binary classification. We used a basic architecture with an input dimension of 33280 (i.e. the dimensionality of the flattened images), six hidden layers each with 416, 208, 104, 52, 26, and 13 neurons (respectively), and all of them using the ReLU activation function with He initialization. The output layer consists of a single neuron using the sigmoid activation function. The model uses the binary cross entropy loss function (as shown below) and the Adam optimizer.

$$\text{Loss}(y, \widehat{y}) = -\left(y \cdot \log \widehat{y} + \log(1-y) \cdot \log(1-\widehat{y})\right)$$

For Model 2, our initial binary convolutional neural network model was copied from the Lin paper [2]. This model used three convolutional layers paired with average pooling layers, a flatten layer, and a final dense layer with size 1024 for the output layer. Overall, this model resulted in an accuracy of 71.1%. Unsatisfied with this result, we added another dense layer to match the number of outputs from the prior flatten layer, but this resulted in an Out of Memory error. At this point, we experimented with adding additional Conv2d layers and modifying the dense layer. However, during this investigation, we ran into an issue with stagnant validation loss. This was subsequently solved with the use of batch

normalization. In the end, our final model included five main layers, each consisting of a convolution, a batch normalization, and a max pooling layer, followed by a flatten layer, and two final dense layers with 5120 and 1 node respectively. Each Conv2D layer uses a kernel size of 5, same padding, and a ReLU activation. Each max pooling layer uses a pool size of 3, valid padding, and a stride of 2x2. While the second to last dense layer uses a ReLU activation, our final dense layer uses a sigmoid activation. Just like in Model 1, we use a binary cross entropy loss and an Adam optimizer.

4.2 Baseline Multi-class Models: Standard Neural Network and CNN

We then created Model 3, which is a standard neural network meant for multi-class classification. We had first used the same architecture as in Model 1 (with the exception of the last layer having three neurons instead of one, and having a softmax activation instead of a sigmoid activation), but we saw that for all combinations of hyperparameters, the model severely overfit the training data. As a result, we decided to cut the number of layers. Thus, the final architecture for Model 3 consists of 3 hidden layers with 416, 26, and 13 neurons, respectively. Each of these hidden layers uses a ReLU activation function with He initialization. The last layer consists of three neurons and uses a softmax activation function. The loss used in this model is a categorical cross entropy loss function (as shown below) and the optimizer is Adam.

$$\text{Loss}(y, \widehat{y}) = -\sum_{i=1}^{3} y_i \log(\widehat{y}_i)$$

For Model 4, we used the same architecture as for Model 2. The only difference was that the output layer for this model uses a softmax activation.

4.3 Transfer Learning for Models 1 and 2

Model 5 is our first transfer learning model. It takes the weights learned by the baseline neural network in Model 1 and applies them to the multi-class problem. To do this, we first loaded in the architecture and saved weights from Model 1. We then removed the last output layer (which contained a single node and used a sigmoid activation) and inserted a dense layer with 3 nodes with a softmax activation. At first, we experimented with how many layers we wanted to freeze. We tried both the first 4 layers and the first 2 layers (2/3 and 1/3, respectively). Once we determined that freezing only the first 2 layers was more effective, we added batch normalization in order to combat the recurring issue of stagnant validation loss.
Model 6 is our second transfer learning model. It takes the weights learned by the CNN model in Model 2 and applies them to the multi-class problem. Once again, this means that we had to swap out the last original output layer with a new dense layer containing 3 nodes and a softmax activation. Based on our experimentation done with Model 5, we froze the first 1/3 full layers. It is worth noting that this is technically the first 6 layers when you consider that each full layer consists of a convolutional layer, a batch normalization layer, and a max pooling layer.

4.4 Standard ResNet and Transfer Learning with ResNet

For our Model 7, we wanted to determine how useful the architecture (and not the saved weights) of ResNet would be in our specific multi-class problem involving MRI images [12]. As a result, we loaded in the ResNet architecture from the Keras library, but with setting the number of classes to 3. For model 8, our final transfer learning model, we essentially used Model 7's architecture again (i.e. ResNet with 3 output classes), but this time with pre-loaded weights generated from training the ResNet model on the ImageNet dataset. Based on our previous experiments with freezing layers, we froze the first 59 layers of the model, corresponding to the first 1/3 of the layers. For both models, we reshaped our images to be 224 x 224 to ensure that we did not modify the ResNet architecture too much (as this was the standard input dimensionality used in the original architecture).

5 Results

5.1 Hyperparameter Searches

We performed a standard grid search over the mini-batch size and a linear search over the number of epochs, using our validation split to determine the optimal hyperparameters. For our binary classifiers

(Models 1 and 2), we ran through mini-batch sizes of 5, 10, 50, 100, and 200. For the rest of our models (the multi-class classifiers), we saw that small batch sizes didn't perform well, so we decided to run through the mini-batch size grid search over sizes of 100, 200, and 300. However, due to the large size of the ResNet model, using large batch sizes in our hyperparameter search for Models 7 and 8 led to Out of Memory errors. As a result, we ran through a mini-batch size grid search over sizes of 32, 64, and 100. Furthermore, for the models that used a standard neural network architecture (Models 1, 3, and 5), we set the maximum number of epochs to search through to 30, as we saw severe overfitting (i.e. train accuracy of 100%) past this number. Differently, for the models that used a CNN architecture (Models 2, 4, and 6) and the ResNet models (Models 7 and 8), we increased the maximum number of epochs to search through to 50, as we believed using batch normalization within the models would reduce the risk of overfitting and thus wanted to get better results.

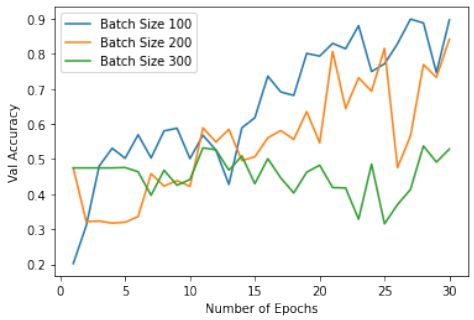

Figure 1: Model 4 Validation Accuracy vs. Epochs across Batch Sizes

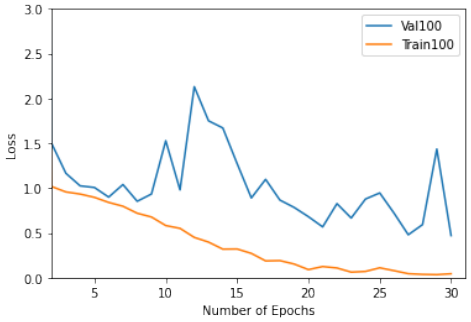

Figure 2: Model 4 Validation and Train Accuracy vs. Epochs for Batch Size 100

To determine the optimal hyperparameters, we first observed the validation accuracy across the number of epochs for all batch sizes. An example of this plot for Model 4 is shown in Figure 1. From this plot, we can determine which batch size has the highest overall validation accuracy. Upon choosing the optimal mini-batch size, we observed the validation and train loss across the number of epochs. An example of this plot for Model 4 (zoomed in) is shown in Figure 2. From these two plots, we chose an epoch number where the validation and train loss are close to each other while also maximizing the validation accuracy. The final, optimal hyperparameters for each of our models are listed in the table below:

Model #	1	2	3	4	5	6	7	8
Batch Size	50	5	300	100	100	200	64	100
Epochs	27	13	49	26	10	9	20	20

5.2 Accuracy Results for Binary Classification

To evaluate how well our binary classification models perform, we use a standard accuracy metric. Here, we have the accuracy results for Models 1 and 2. We see that, as expected, the CNN binary classification model performed better than our baseline binary classification neural network.

Model #	1	2
Accuracy	80.77%	86.81%

5.3 F1-Scores for Multi-class Classification and Transfer Learning

To evaluate how well our multi-class classification models are predicting and determine overall classifier performance, we use macro-averaged F1 scores. Additionally, we use confusion matrices to visualize their performances over individual classes. Below are the F1 scores for all the models that performed multi-class classification, including those which used transfer learning. First, we once again see that the CNN multi-class model (Model 4) performed better than our standard multi-class neural network (Model 3). We also see that Models 5 and 6 both performed better than their respective

counterparts (3 & 4), which affirms our hypothesis that transfer learning would improve our results. The confusion matrices for each result can be seen in Figures A3 through A6.

Model #	3	4	5	6	7	8
F1-Score	0.739	0.779	0.783	0.906	0.873	0.890

Finally, we completed our investigation of ResNet architecture. The results for Models 7 & 8 can be seen above. We found that the ResNet model trained only on our own data performed worse than the model that was pre-trained on ImageNet. This also falls in line with our expectations of transfer learning. The respective confusion matrices can be seen in Figures A7 and A8. Interestingly, the best performing model on the multi-class problem was our standard CNN that transfer learned from the binary problem. This could indicate that added complexity (i.e. the ResNet model) doesn't necessarily equate to better performance.

5.4 Analysis

Looking at our models' confusion matrices, of the three multi-class labels, we found that label 1 was classified fairly well. It had an average error rate of 8.2% over all 8 models. However, labels 0 and 2 had more issues, with error rates of 21.2% and 20.5%, respectively. With label 0, we believe the issue stems from image inconsistency from the pre-processing phase. As it turns out, images resulting from slice 96 of the ADNI images can be very different depending upon the individual patient and imaging procedure. This means that some brains that were healthy were likely identified as unique and unhealthy. With label 2, we believe the errors are caused by a simple lack of data. With more data for patients with moderate to severe dementia, we believe predictions on 2 would be more accurate. In Figure 3, we see the Model 6 confusion matrix as an example. It is worth noting that the classifier does well in not misidentifying unhealthy brains as healthy ones. In terms of real world application this is extremely important, as it's better to be misdiagnosed as a false positive, than a false negative.

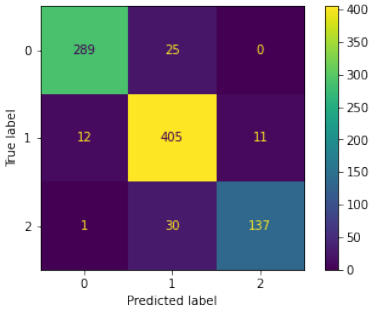

Figure 3: Model 6 Confusion Matrix

6 Conclusion and Future Work

Overall, our project's results matched our expectations. It's clear that transfer learning improved model performance across all architectures. This would suggest that the optimal approach to dementia classification from MRIs involves transfer learning. This has several logistical benefits. For example, transfer learning tends to improve a model's runtime since frozen layers need not be retrained. However, more pertinently, transfer learning improves model performance when the amount of data is small. For this problem in particular, gathering data is difficult because it requires a great deal of resources to collect MRI images from real dementia patients. Transfer learning gives researchers and medical professionals a method by which to sidestep this issue.

In terms of future work, the first obvious thing to do is to try different ratios of freezing layers for models 5, 6, and 8 to see if F1 scores can be increased. Additionally, inspiration can be found in the Ebrahimi-Ghahnavieh paper [7]. It would be a reasonable next step to expand this project to the 3D images provided by ADNI, although it would require us to remove the OASIS data from our project. Using the 3D MRI scans as provided would somewhat alleviate the data quality problem mentioned in 5.4 (although our models still performed well despite this sticking point). We also think future work could include creating a multi-modal architecture to include other structured patient data to make more educated predictions on whether a patient has dementia and its severity.

7 Contributions

Overall, we shared the load of this project fairly equally. We came up with the concept together and defined the project plan as a team. Once we started coding, Constantine completed image preprocessing on all the data, while Elizabeth created pipelines to rename data when needed and created the train/dev/test splits for both the midpoint evaluation and the final datasets. We usually worked on the code together and discussed what model architectures would be best. However, due to the logistical difficulties of Google Colab, Constantine wrote most of the code seen in the final project github. In turn, Elizabeth wrote most of the final paper/presentation.

References

[1] Stamate, D., Smith, R., Tsygancov, R., Vorobev, R., Langham, J., Stahl, D., Reeves, D. (2020). Applying Deep Learning to Predicting Dementia and Mild Cognitive Impairment. Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part II, 584, 308–319. https://doi.org/10.1007/978-3-030-49186-4$_2$6

[2] Lin, W., Tong, T., Gao, Q., Guo, D., Du, X., Yang, Y., ...Initiative, T. A. D. N. (2018). Convolutional Neural Networks-Based MRI Image Analysis for the Alzheimer's Disease Prediction From Mild Cognitive Impairment. Front. Neurosci., 0. doi: 10.3389/fnins.2018.00777

[3] Korolev, S., Safiullin, A., Belyaev, M., Dodonova, Y. (2017, April). Residual and plain convolutional neural networks for 3D brain MRI classification. In 2017 IEEE 14th international symposium on biomedical imaging (ISBI 2017) (pp. 835-838). IEEE.

[4] Chen, H., Dou, Q., Yu, L., Heng, P.-A. (2016). VoxResNet: Deep Voxelwise Residual Networks for Volumetric Brain Segmentation. arXiv, 1608.05895. Retrieved from https://arxiv.org/abs/1608.05895v1

[5] Basaia, S., Agosta, F., Wagner, L., Canu, E., Magnani, G., Santangelo, R., Filippi, M. (2019). Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks. NeuroImage: Clinical, 21, 101645. doi: 10.1016/j.nicl.2018.101645

[6] Aderghal, K., Khvostikov, A., Krylov, A., Benois-Pineau, J., Afdel, K., Catheline, G. . Classification of Alzheimer Disease on Imaging Modalities with Deep CNNs Using Cross-Modal Transfer Learning. 2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS). IEEE. doi: 10.1109/CBMS.2018.00067

[7] Ebrahimi-Ghahnavieh, A., Luo, S., Chiong, R. (2019). Transfer Learning for Alzheimer's Disease Detection on MRI Images. 2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT). IEEE. doi: 10.1109/ICIAICT.2019.8784845

[8] Marcus, D. S., Wang, T. H., Parker, J., Csernansky, J. G., Morris, J. C., amp; Buckner, R. L. (2007, September 1). Open access series of imaging studies (OASIS): Cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. Journal of Cognitive Neuroscience. Retrieved November 19, 2021, from https://direct.mit.edu/jocn/article/19/9/1498/4427/Open-Access-Series-of-Imaging-Studies-OASIS-Cross.

[9] Petersen, R. C., Aisen, P. S., Beckett, L. A., Donohue, M. C., Gamst, A. C., Harvey, D. J., Jack, C. R., Jr, Jagust, W. J., Shaw, L. M., Toga, A. W., Trojanowski, J. Q., Weiner, M. W. (2010). Alzheimer's Disease Neuroimaging Initiative (ADNI): clinical characterization. Neurology, 74(3), 201–209. https://doi.org/10.1212/WNL.0b013e3181cb3e25

[10] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[11] Chollet, F., others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[12] He, K., Zhang, X., Ren, S., Sun, J. (2015). Deep Residual Learning for Image Recognition. arXiv, 1512.03385. Retrieved from https://arxiv.org/abs/1512.03385v1

Appendix

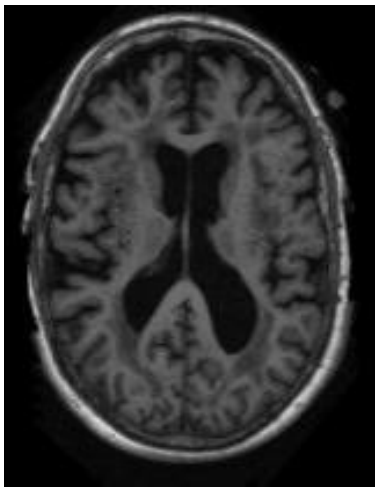

Figure A1: OASIS Example

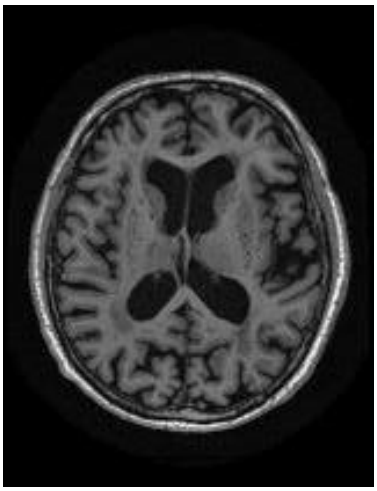

Figure A2: ADNI Example

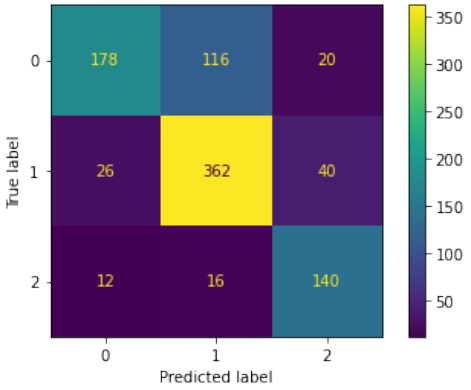

Figure A3: Model 3 Confusion Matrix

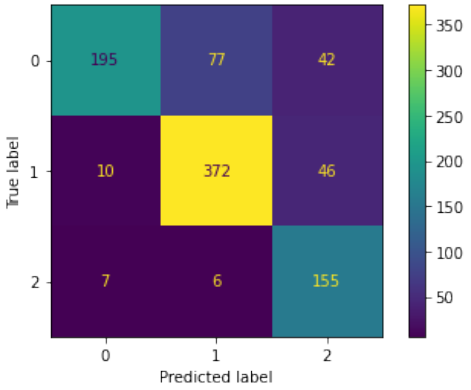

Figure A4: Model 4 Confusion Matrix

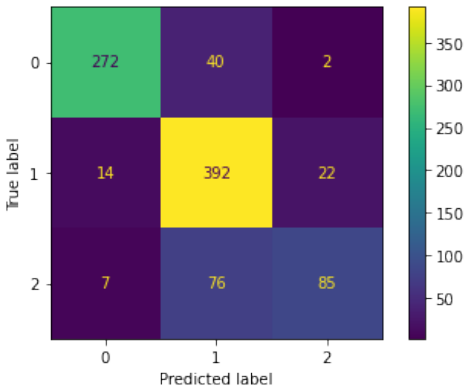

Figure A5: Model 5 Confusion Matrix

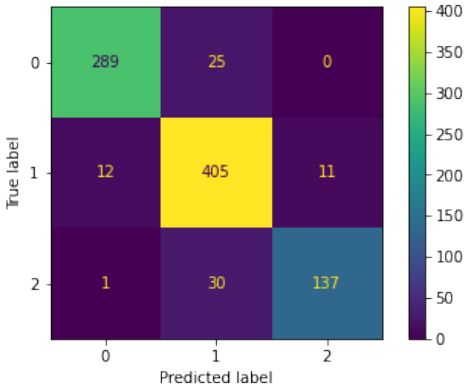

Figure A6: Model 6 Confusion Matrix

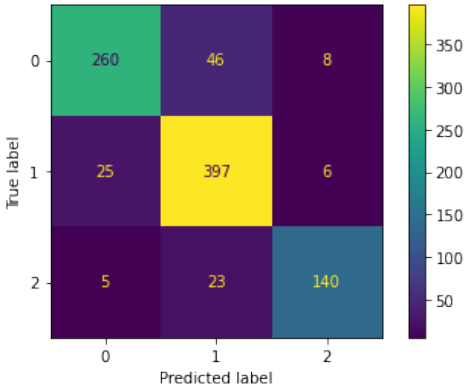

Figure A7: Model 7 Confusion Matrix

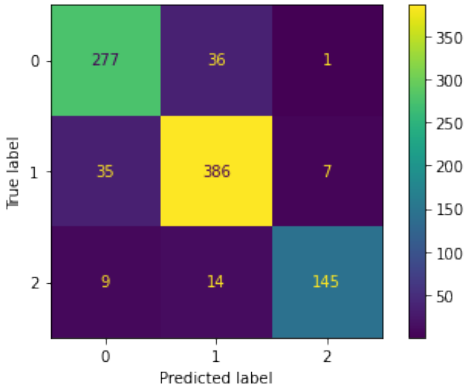

Figure A8: Model 8 Confusion Matrix

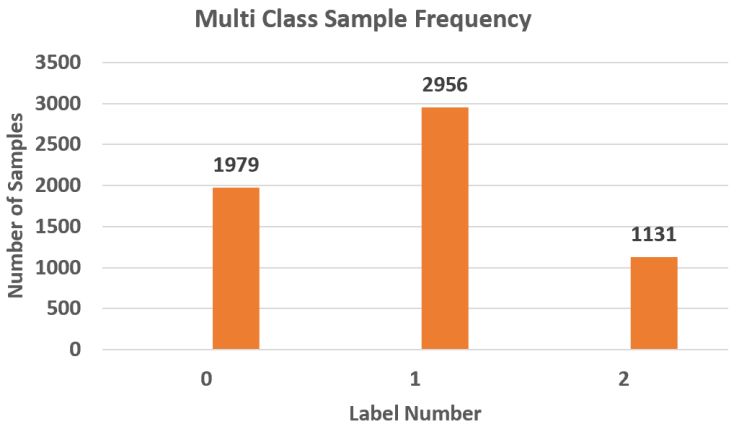

Figure A9: Label Frequencies