Long Term Short Term Convolutional Neural Network Classifier to Diagnosis Alzheimer's Disease

William Floyd Jones Georgia State University

***Keywords*—Long Short Term Memory, Convolutional Neural Network, Magnetic Resonance Imaging, Longitudinal Images, Alzheimer's Disease, Computer Aided Diagnosis,**

I. Introduction

Alzheimer's is one of the most prevalent types of dementia and makes up 60-80 percent of dementia cases [1]. The disease brings about memory loss which gradually worsen over time. With no cure, Alzheimer's can be detrimental to day life. Early detection is necessary to help mitigate symptoms and slow progression. On of the ways doctors can help detect Alzheimer's is by conducting scans such as MRI. The Magnetic Resonance Imaging scan can create images of the brain and detect abnormalities. Some abnormalities in the brain can be classified as mild cognitive impairment (MCI) that can then be further classified if its associated with Alzheimer's [2]. MRI is one of the modalities that make up Computer Assisted Diagnosis (CAD) to aid neurologist in brain detection.

CAD employ machine learning methods in brain abnormality detection. Since these models start of as images there needs to be prior work done to to create features for the model to look for. Prior studies have conducted different methods such as feature extraction, feature selection, noise reduction, and classification techniques to detect Alzheimer's [3].

One of the most common machine learning methods used on images is Convolutional Neural Network (CNN). Using CNN we can create a model to classify MCI as Alzheimer's or non-Alzheimer's. Previous papers have used CNN to predict the disease[3][4]. CNN is deep learning algorithm that is used to learn to detect features on images. They are most used in computer vision and classification. The benefit of a CNN is it learns the features own and reduces the time to classify a image. It is comprised of many different layers being convolutional, pooling, and fully connected layer. In the convolutional layer the image is comprised of a matrix of numbers , a smaller matrix called a kernel/filter moves across the matrix checking if there are any features in the image matrix. The filter is a matrix of weights that when interacting with the image matrix performs a dot product to each area of the image. the output of this is a smaller activation map. The activation map is then moved to the pooling layer where it is reduces the number of parameters in the map. This is done by conducting a filter similar to that in the convolutional layer but it takes the maximum value of the selected map and sends it to another much smaller map. After that in the fully connected layer the classification is conducted there are layers such as flatten, dense, and activation that are located in the fully connected layer. The CNN is able to back propagate meaning feed the final output back into the the beginning of neural network where it is done for a select number of epochs.

The long short term memory (LSTM) is mostly used in recurrent neural networks. It is able to help avoid long term dependency that happens when training a neural network for too long. This works by switching between long term information(previous run) and short term information(current run) by way of its cell state. The reasoning for combining with the CNN model is to help better detect features in the MRI since every brain is slightly unique in small aspects compared to others a new approach is needed.

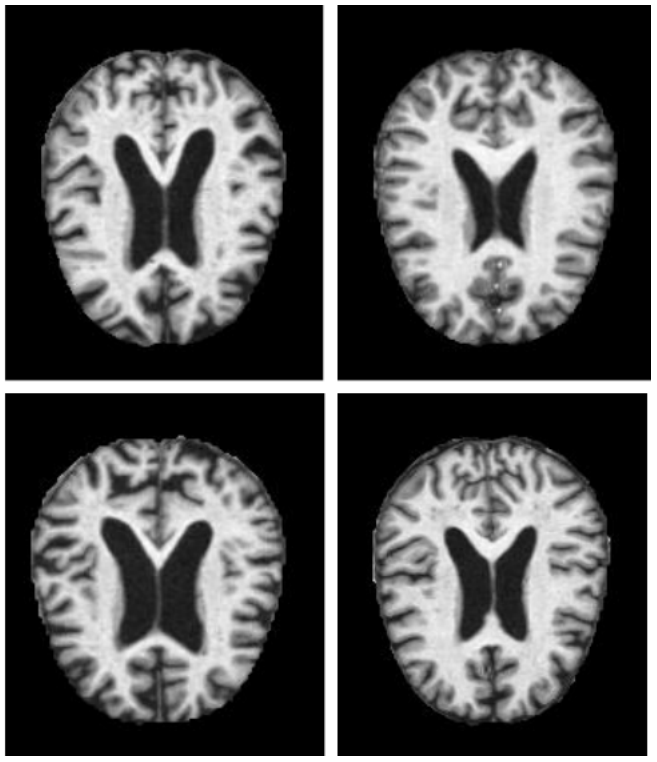
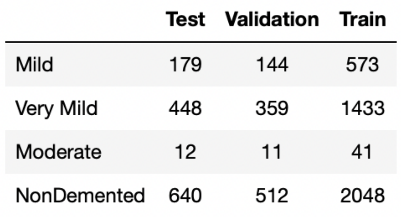


Fig. 1. Example of images used in the LSTM-CNN. The images are top left: Mild Demented, top right: NonDemented, bottom left: Moderate Demented, and bottom right: Very Mild Demented.

II. Data Collection

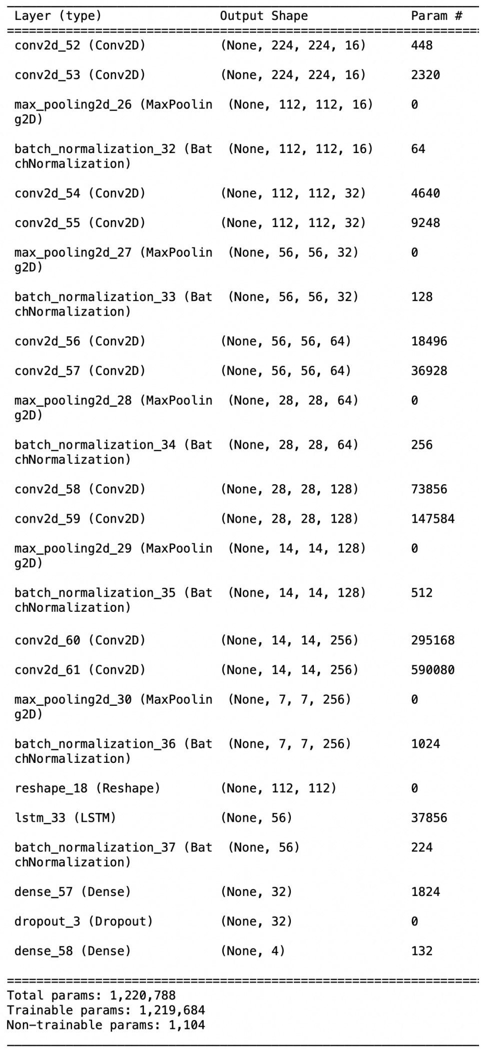
The data used was provided from kaggle, it consists of several thousand images collected from MRI scans. The images are broken up into the categories of non demented, mild demented, very mild demented, and moderate demented. An example of these images are shown in figure 1. The sets of images are separated into categories of train, validation, and test as shown in figure 2. They are used to train the neural network, validate the parameters, and then test the metrics of the network respectively.



	Test	Validation	Train
Mild	179	144	573
Very Mild	448	359	1433
Moderate	12	11	41
NonDemented	640	512	2048

Fig. 2. A breakdown of images categorized in data sets and per demented level.

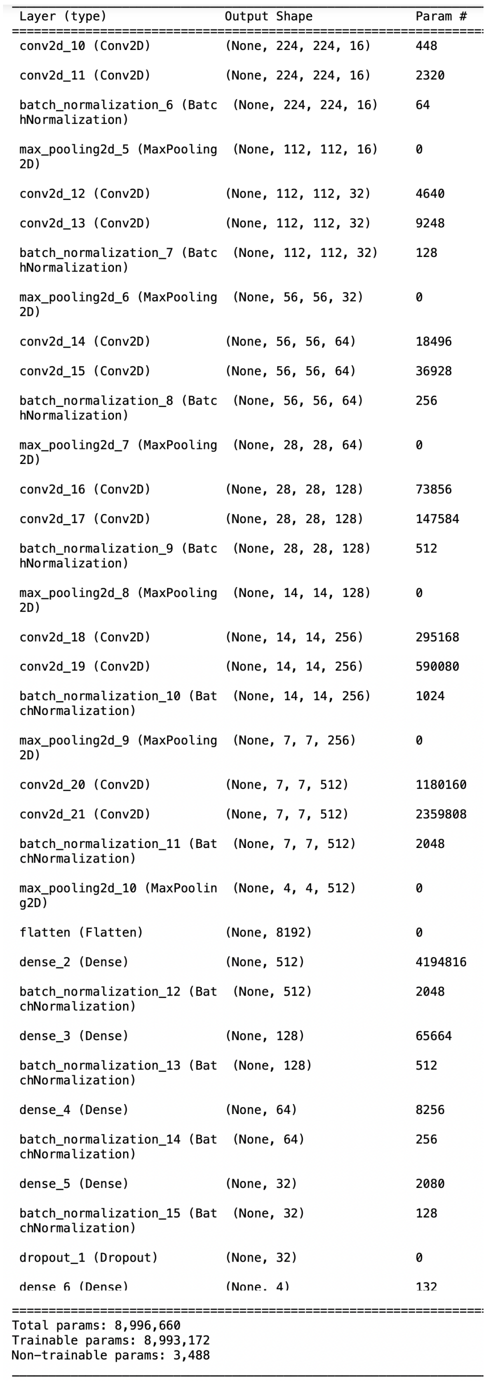
III. Methodology



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_52 (Conv2D)           (None, 224, 224, 16)      448

conv2d_53 (Conv2D)           (None, 224, 224, 16)      2320

max_pooling2d_26 (MaxPoolin  (None, 112, 112, 16)      0
g2D)

batch_normalization_32 (Bat  (None, 112, 112, 16)      64
chNormalization)

conv2d_54 (Conv2D)           (None, 112, 112, 32)      4640

conv2d_55 (Conv2D)           (None, 112, 112, 32)      9248

max_pooling2d_27 (MaxPoolin  (None, 56, 56, 32)        0
g2D)

batch_normalization_33 (Bat  (None, 56, 56, 32)        128
chNormalization)

conv2d_56 (Conv2D)           (None, 56, 56, 64)        18496

conv2d_57 (Conv2D)           (None, 56, 56, 64)        36928

max_pooling2d_28 (MaxPoolin  (None, 28, 28, 64)        0
g2D)

batch_normalization_34 (Bat  (None, 28, 28, 64)        256
chNormalization)

conv2d_58 (Conv2D)           (None, 28, 28, 128)       73856

conv2d_59 (Conv2D)           (None, 28, 28, 128)       147584

max_pooling2d_29 (MaxPoolin  (None, 14, 14, 128)       0
g2D)

batch_normalization_35 (Bat  (None, 14, 14, 128)       512
chNormalization)

conv2d_60 (Conv2D)           (None, 14, 14, 256)       295168

conv2d_61 (Conv2D)           (None, 14, 14, 256)       590080

max_pooling2d_30 (MaxPoolin  (None, 7, 7, 256)         0
g2D)

batch_normalization_36 (Bat  (None, 7, 7, 256)         1024
chNormalization)

reshape_18 (Reshape)         (None, 112, 112)          0

lstm_33 (LSTM)               (None, 56)                37856

batch_normalization_37 (Bat  (None, 56)                224
chNormalization)

dense_57 (Dense)             (None, 32)                1824

dropout_3 (Dropout)          (None, 32)                0

dense_58 (Dense)             (None, 4)                 132

=================================================================
Total params: 1,220,788
Trainable params: 1,219,684
Non-trainable params: 1,104
_________________________________________________________________
```

Fig. 3. A model summary of the LSTM-CNN with each layer created.

For the LSTM-CNN model creation a neural network was created with 5 of the following 2 convolutional layers, max pooling layer, and followed by a batch normalization. after those blocks a reshaping had to be conducted to provide the correct input dimensions to the lstm layer, once that occurred a two dense layers were used to connect layers. A detailed summary of each layer is shown in figure 3. To test the model against conventional methods of classifying images another CNN was created with another convolutional block consisting of 2 convolutional layers, batch normalization, and max pooling layer that replaced the original lstm layer. It also contained a couple more dense layers to better connect everything, this is also detailed in a summary in figure 4. Both of the models used a learning rate scheduler called exponential decay which lowers the learning rate of the model as the training progresses. This helps with optimization and generalization.



```
Layer (type)                Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)          (None, 224, 224, 16)      448

conv2d_11 (Conv2D)          (None, 224, 224, 16)      2320

batch_normalization_6 (Batc  (None, 224, 224, 16)     64
hNormalization)

max_pooling2d_5 (MaxPooling  (None, 112, 112, 16)     0
2D)

conv2d_12 (Conv2D)          (None, 112, 112, 32)      4640

conv2d_13 (Conv2D)          (None, 112, 112, 32)      9248

batch_normalization_7 (Batc  (None, 112, 112, 32)     128
hNormalization)

max_pooling2d_6 (MaxPooling  (None, 56, 56, 32)       0
2D)

conv2d_14 (Conv2D)          (None, 56, 56, 64)        18496

conv2d_15 (Conv2D)          (None, 56, 56, 64)        36928

batch_normalization_8 (Batc  (None, 56, 56, 64)       256
hNormalization)

max_pooling2d_7 (MaxPooling  (None, 28, 28, 64)       0
2D)

conv2d_16 (Conv2D)          (None, 28, 28, 128)       73856

conv2d_17 (Conv2D)          (None, 28, 28, 128)       147584

batch_normalization_9 (Batc  (None, 28, 28, 128)      512
hNormalization)

max_pooling2d_8 (MaxPooling  (None, 14, 14, 128)      0
2D)

conv2d_18 (Conv2D)          (None, 14, 14, 256)       295168

conv2d_19 (Conv2D)          (None, 14, 14, 256)       590080

batch_normalization_10 (Bat  (None, 14, 14, 256)      1024
chNormalization)

max_pooling2d_9 (MaxPooling  (None, 7, 7, 256)        0
2D)

conv2d_20 (Conv2D)          (None, 7, 7, 512)         1180160

conv2d_21 (Conv2D)          (None, 7, 7, 512)         2359808

batch_normalization_11 (Bat  (None, 7, 7, 512)        2048
chNormalization)

max_pooling2d_10 (MaxPoolin  (None, 4, 4, 512)        0
g2D)

flatten (Flatten)           (None, 8192)              0

dense_2 (Dense)             (None, 512)               4194816

batch_normalization_12 (Bat  (None, 512)              2048
chNormalization)

dense_3 (Dense)             (None, 128)               65664

batch_normalization_13 (Bat  (None, 128)              512
chNormalization)

dense_4 (Dense)             (None, 64)                8256

batch_normalization_14 (Bat  (None, 64)               256
chNormalization)

dense_5 (Dense)             (None, 32)                2080

batch_normalization_15 (Bat  (None, 32)               128
chNormalization)

dropout_1 (Dropout)         (None, 32)                0

dense_6 (Dense)             (None, 4)                 132

=================================================================
Total params: 8,996,660
Trainable params: 8,993,172
Non-trainable params: 3,488
_________________________________________________________________
```

Fig. 4. A model summary of the CNN with each layer created.

IV. Results

To evaluate the models against each other several metrics were used. The first being loss which was measured using the categorical crosss entropy loss as shown in figure 5. Both of the models have a low training loss which shows that the errors in countered while training are low. However, the validation data for the CNN model is extremely high the in the first epoch which means the CNN model in countered lots of errors in the first run. But with the loss minimizing as the epoch increases means the model is learning.

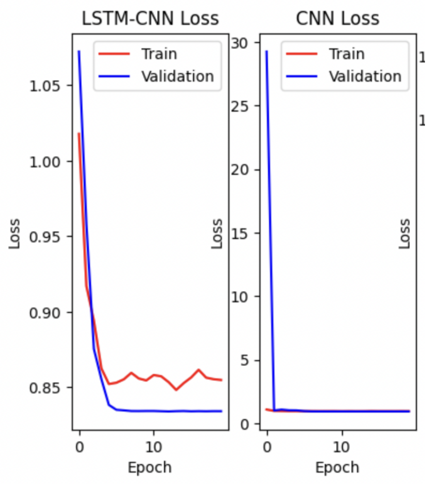


Fig. 5. the categorical cross entropy loss of the models.

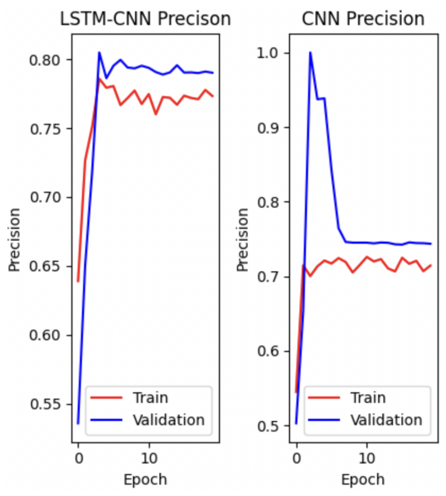


Fig. 6. The precision of the models.

In figure 6, the precision of each model is shown. The LSTM-CNN has a slightly higher precision in the both the train and validation. Both of the models has a precision of .7-.8 meaning that they have a high degree of getting true positives out of false positives. Now when looking at the area under the curve (auc) of the models in figure 7 it shows the ability of the neural network to distinguish between positive classes (true positive and true negative) and negative classes (false positive and false negative). A value of 1 would indicate the model is perfectly able to distinguish between the two while a value of 0 would mean it would be predicting positives as negatives and vice versa. looking at both the models they are able to distinguish between the two classes to a high degree with the LSTM-CNN performing better than the CNN .

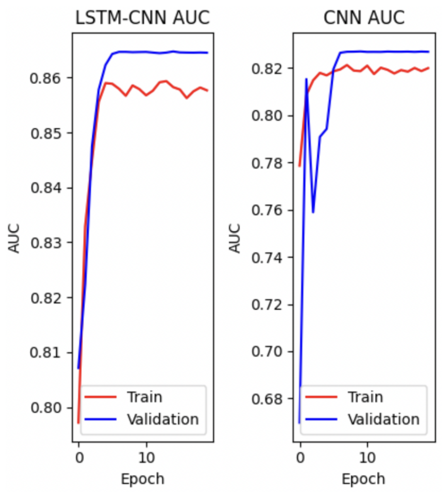
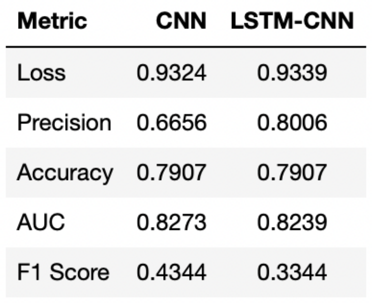


Fig. 7. The area under the curve of the models.



Metric	CNN	LSTM-CNN
Loss	0.9324	0.9339
Precision	0.6656	0.8006
Accuracy	0.7907	0.7907
AUC	0.8273	0.8239
F1 Score	0.4344	0.3344

Fig. 8. The area under the curve of the models.

To better understand our outputs a f1 score was done. It is the mean of the precision and recall of the models. It is more of a true accuracy of a neural network. In figure 8 this is displayed. It is shown that the LSTM-CNN model has a higher f1 score at about 0.45 than the CNN model of about 0.35. While being below 0.5 is not the best for a f1 score this means that the recall is low and there is a high number of false positives. To see how these models perform when testing, a test data set was evaluated for each model consisting of the same images. The outputs are shown in figure 9, while both seem to be on par with each other the LSTM-CNN model has a higher precision but a lower f1 score. This means that the model labeled a lot more images incorrectly than the CNN model.

V. Conclusion

In finding ways to stop a deadly disease that affect so many humans. A deep learning model was proposed to predict the demented cases of thousands of MRI imaging. The LSTM-CNN model was able to predict cases well but also in the process incorrectly labeled more images than a typical CNN model. This shows that is possible to use an LSTM layer combined with a CNN model to predict demented cases. In the future more available imaging would be necessary to improve the model over time.

References

[1] "What is Alzheimer's?," Alzheimer's Disease and Dementia. [Online]. Available: https://www.alz.org/alzheimers-dementia/what-is-alzheimers. [Accessed: 05-Sep-2022].

[2] Radiological Society of North America (RSNA) and American College of Radiology (ACR), "Alzheimer's disease," Radiologyinfo.org. [Online]. Available: https://www.radiologyinfo.org/en/info/alzheimers. [Accessed: 05-Sep-2022].

[3] N. Vinutha, S. Pattar, C. Kumar, A. Agarwal, P. D. Shenoy and K. Venugopal, "A Convolution Neural Network based Classifier for Diagnosis of Alzheimer's Disease," 2018 Fourteenth International Conference on Information Processing (ICINPRO), 2018, pp. 1-6, doi: 10.1109/ICINPRO43533.2018.9096819.

[4] B. Khagi, B. Lee, J. -Y. Pyun and G. -R. Kwon, "CNN Models Performance Analysis on MRI images of OASIS dataset for distinction between Healthy and Alzheimer's patient," 2019 International Conference on Electronics, Information, and Communication (ICEIC), 2019, pp. 1-4, doi: 10.23919/ELINFOCOM.2019.8706339.