# Lateralization CNN with Identification

**Junxian Rao**
University of California, San Diego
rao.jun.xian@gmail.com

## Abstract

Convolutional neural networks have gained quick popularity in the last decade for image classification because they are uniquely well suited to parse out features and weight the values into a classification system. One domain that has seen particular benefit is the medical imaging community. Radiologists are hard to train, sometimes produce errors, and have a limited ability to detect subtle features of the image, as is to be expected from the limits of human physiology, so convolutional neural networks have proven a promising solution. I trained a convolutional neural network to differentiate between MRI images of right-side and left-side mesial temporal lobe epilepsy patients as well as from a healthy control population and show the potential of applications of artificial intelligence in this field.

## 1 Introduction

### 1.1 Background

The term 'Machine learning' has been around since IBM employee Arthur Samuel coined it in 1959[6] and artificial neural networks (ANN) have played an important role since the beginning. Frank Rosenblatt created the first ANN, a 'perceptron' in 1958 [5] which can be thought of as a computational unit that resembles the functionality of a neuron. When these units are connected and layered, they become known as multi-layer perceptrons (MLP), a type of feedforward processing. Subsequently, convolutional neural networks (CNN), a regularized version of MLPs, began emerging in the mid-1980s but became more prominent in the late 2000s as higher quantities of training data and better hardware became available, two important components for successful CNNs. The design of MLPs and thus CNNs were originally modeled from processes in the visual cortex of mammals, elucidated by neurobiological research,[2][4] and as a natural consequence, they are particularly well suited for image analysis. They have become fundamental in the domain of computer vision.

## 2 Prior Work

### 2.1 Radiological identification of temporal lobe epilepsy using deep learning

Zhang, et al. conceived of a CNN in 1991 for medical imaging analysis in the detection of breast cancer using mammograms[7]. Eventually, CNNs made their way toward disease diagnosis in structural Magnetic Resonance Imaging (MRI) of brain tissue. Recently in 2021, research was published assessing whether convolutional neural networks are a viable method for the diagnosis of Medial Temporal Lobe Epilepsy (MTLE).[1] There exists a significant correlation between MTLE and Mesial Temporal Sclerosis (MTS), a histological deformity characterized by cell loss and glial reactivity in the hippocampus. Using T1-weighted images, reduced volume can be quantified, and gliosis can be measured with T2 values.[3] The 2021 Gleichgerrcht, et al. paper had both the raw and smoothed gray matter segmentation of coronal slices of healthy control and TLE MRI scans fed into the convolutional neural network (CNN). They started with an input image of a native $N \times N$ coronal slice.

Preprint. Under review.

The first convolutional layer is a subtle $N/2 \times N/2$ layer, the second is a multi-scale $N/4 \times N/4$ layer followed by a coarse $N/8 \times N/8$ layer. Afterward, there was a fully connected layer defined by layer properties of gray matter tissue patterns and several features. Finally, there was a fully connected softmax classification layer splitting the images into 'control' or 'TLE'. It works by convolving the image (applying a function to the pixels) and extracting the most important features from each layer, then passing it on to the next. This way, even subtle changes can be detected by the algorithm. This methodology is typically adopted for image analysis because the differences are more complicated and non-linear, thus a traditional machine learning approach like regression and Support Vector Machines (SVM) are not the best fit for the task.

### 2.2 Analysis Workflow

In this study, I want to replicate a previous study that developed a deep learning method to detect epilepsy in neuroimaging data, and I want to expand on that study by also lateralizing the epilepsy group. I have a copy of the original dataset as well as a new group of data (Right TLE patients). I first replicated the results by adopting their model into Python. After I have shown that I have successfully replicated their results, I will add my new dataset and see how well their model is performing with the new task. I will also try to implement changes to their deep learning model to see if I can improve upon their model for my current task.

## 3 Replication

### 3.1 Data Preprocessing

The original study has over 200 healthy controls and about 100 left TLE patients. They are processed from raw dicom images from a scanner using SPM to spatially register the images to standard space and extract a grey matter mask from the image (as well as smoothing in certain cases). A copy of the original dataset is obtained but preprocessing steps are not able to be completely replicated due to updated software and missing preprocessing parameters. The dataset used in this current study is processed similarly to the original study with registration to standard space and grey matter extraction but the specific steps and processes are replicated to my best efforts. Figure **??** is an example of the input image that is produced from my preprocessing steps and as inputs to the model.

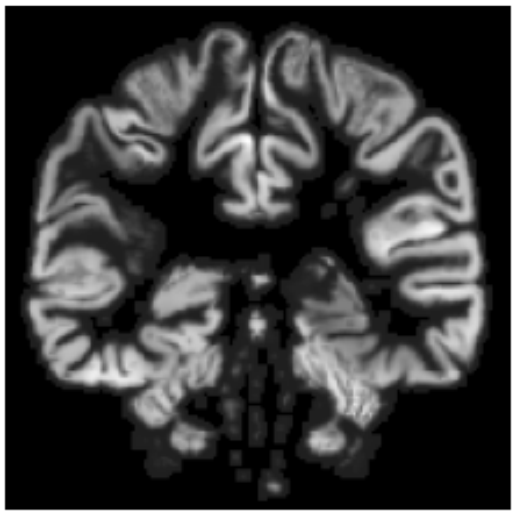

Figure 1: Sample coronal slice output after spatial registration and grey matter extraction.

### 3.2 Analysis Pipeline

How this pipeline was the same as the paper will be stated first, and then my adjustments will be addressed. The pipelines consist of a stratified K-fold cross-validation to make the best use of the small number of participants and check for overfitting. MRI scans are costly, time-consuming, and require technical expertise so typically human imaging processing pipelines need to account for low sample sizes; therefore the data was split, trained, tested, and validated in multiple iterations slicing up the data differently each time. The CNN was built using Keras and TensorFlow, using a similar architecture to the previous study with 3 convolutional layers with roughly the same elements: a relu activation function, a batch normalization layer after all three, and a pooling layer for the first two. The relu activation (rectified linear unit: max(0,x)) is a common function in neural networks where the data either outputs as a 0 (if the input is negative) or the original value (1, 2, 3… n), this functions similar to a biological neuron in that it uses the all or nothing principle but slightly different in that it doesn't have an upper limit like a biological neuron does. Lastly, there's a final dense layer with a 'SoftMax' activation function which converts the values to a probability distribution according to the likelihood of each category, either left TLE or healthy control. I decided to augment some of the code to optimize certain elements. Eventually, there is a step to set up the optimizer with learning rate and loss function. Figure 2 shows the overall architecture of the model I adopted from the previous study.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv_1 (Conv2D) | (None, 160, 160, 8) | 80 |
| BN_1 (BatchNormalization) | (None, 160, 160, 8) | 32 |
| max_pooling2d_327 (MaxPoolin | (None, 80, 80, 8) | 0 |
| conv_2 (Conv2D) | (None, 80, 80, 16) | 1168 |
| BN_2 (BatchNormalization) | (None, 80, 80, 16) | 64 |
| max_pooling2d_328 (MaxPoolin | (None, 40, 40, 16) | 0 |
| conv_3 (Conv2D) | (None, 40, 40, 32) | 4640 |
| BN_3 (BatchNormalization) | (None, 40, 40, 32) | 128 |
| flatten_78 (Flatten) | (None, 51200) | 0 |
| dense (Dense) | (None, 2) | 102402 |

Total params: 108,514
Trainable params: 108,402
Non-trainable params: 112

Figure 2: Model architecture. I adopted the same model architecture as the previous study.

### 3.3 Replication Results

After replicating the input and the model from the previous study, I successfully run the model. This is shown in Figure 3. I can see that the training set loss (blue), as well as the validation set loss (orange), are both going down generally. In some splits, I can see that the validation set loss ended with it going up but it could easily be due to not stopping training early enough and the model started to over-fit to the training set. But in general, I can see that the model is running and running correctly.

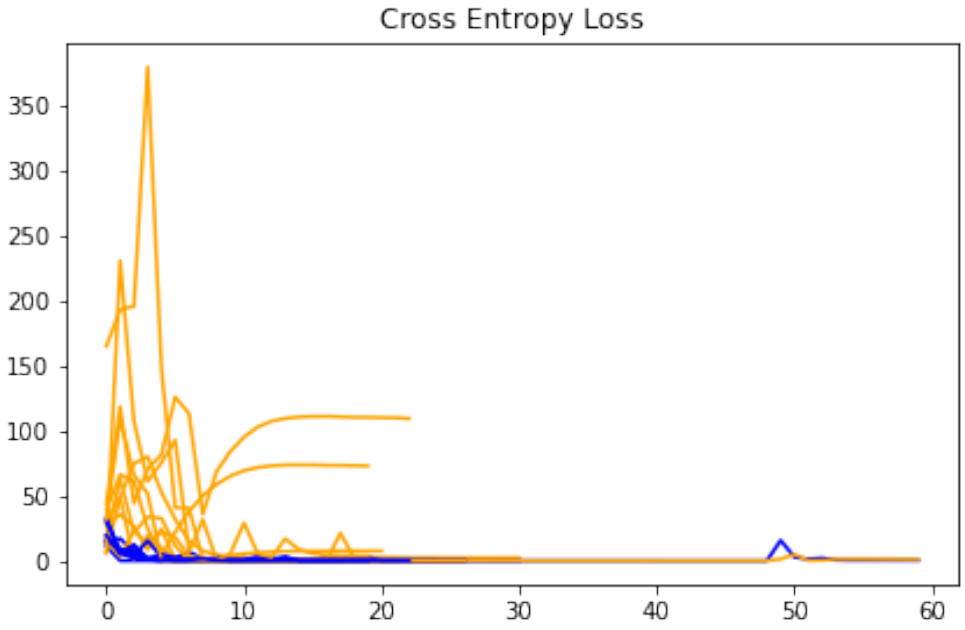


Figure 3: Replicated Model Training Loss. Loss of the replicated model showing that the model has been successfully implemented and is training correctly. Training set loss is in blue and validation set loss is in orange.

Next, I look at the model performance shown in Figure 4. I can see that accuracy is higher than in the study at 94.71% compared to the 85% of in the previous study. This could be likely due to us not needing to SMOTE the data (create artificial data) and have balanced classes. I can see that sensitivity (how accurate are the model when it labels a subject as a patient) is similar to the previous study, where they both sometimes mislabeled patients as controls. But here, I can see that the specificity of the current model (how accurate is the model when it labels a subject as control) is 100% meaning that the model does not mislabel patients as controls. I can also see that PPV is also 100% meaning that if the model labels someone as a patient, they will be a patient. NPV is less than 100% like the previous study meaning that if the model labels someone as a control, they may not be a control. AUC is around 0.95 which is slightly higher than the previous study which is understandable as this model has higher performance than the one from the previous study.

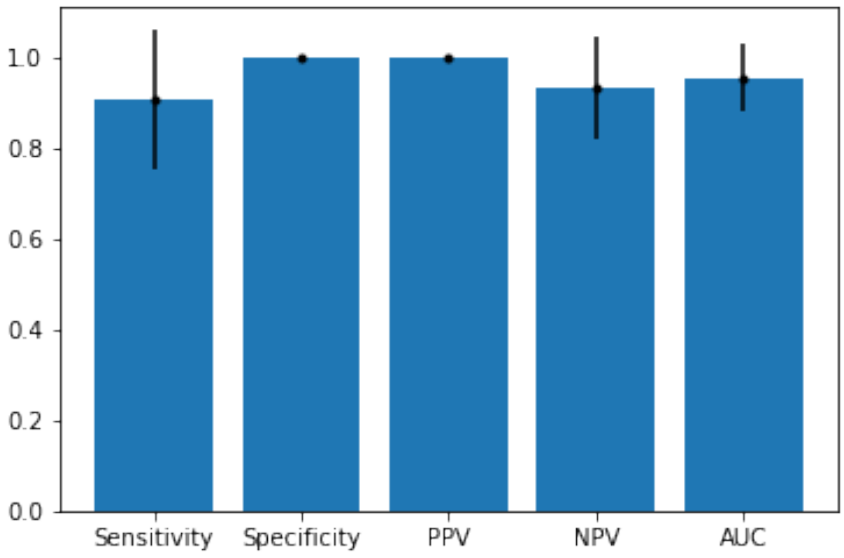


Figure 4: Replicated Model Performance. The sensitivity, specificity, PPV, NPV, and AUC of the replicated model and error across the splits.

In Figure 5, I show the feature importance map of each of the three convolutional layers. From the top figure, I can see that the model is focusing on most of the grey matter with extra emphasis on the grey matter of the temporal lobe. The second convolutional layer, the model mainly focuses on the contours of the brain. The third convolutional layer is back to focusing on mainly the temporal lobe of the brain but some other extra-temporal regions also became important to the model. These differences in the feature importance map of the model and the previous model but I believe that Figure 5 still shows that the model still learned the correct regions of the brain to identify epilepsy, it is just doing it differently than the model from the previous study.

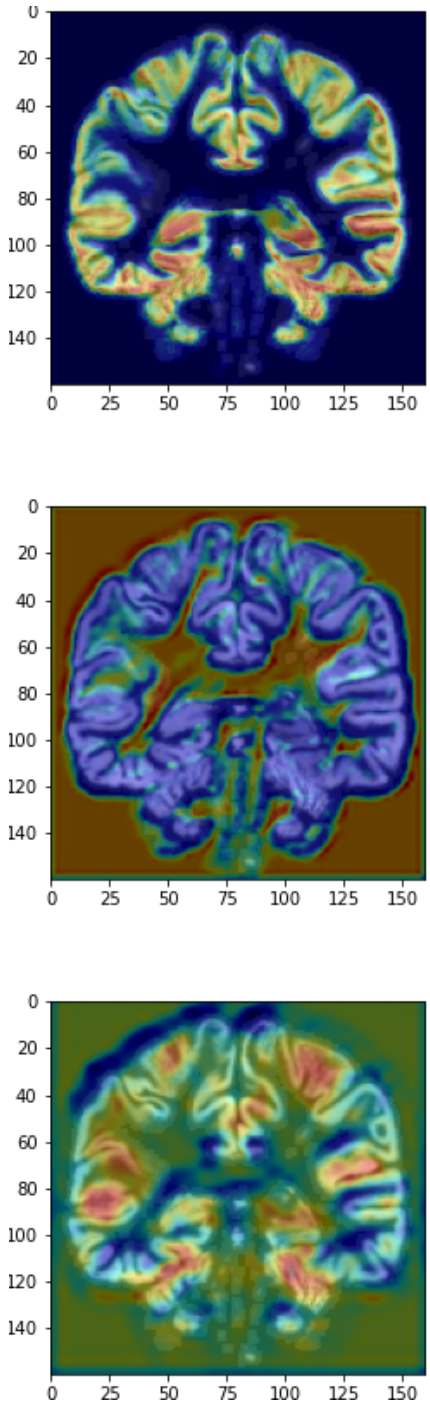


Figure 5: Feature importance map of the replicated model. The top figure is from the first convolutional layer, the second figure is from the second convolutional layer, and the third figure is from the third convolutional layer.

## 4 TLE identification and lateralization

After showing that I have correctly implemented the convolutional neural network from the previous study, I want to test how well it can perform with the extra task of lateralizing the side of epilepsy onset. I will be adding Right TLE patients with a new class label and seeing how well the model performs and if there are any improvements to be made.

### 4.1 Additional Dataset

I combined the previous dataset with more left-side TLE patients as well as added a new group of right-side TLE. In total, I had 170 control patients 170 left TLE, and 160 right side TLE. Everything about preprocessing was the same except for the addition of a new group. This new group is important because it adds a whole new patient population who needs diagnosis. Until now the CNN is only capable of classifying between left-side TLE and control groups.

### 4.2 Validation

I show that the model ran correctly by showing that the model has been learning the new dataset (Figure 6). I can see that just like any proper model, the training and validation loss are both going down. In Figure 7, I show that the model on the new task has significantly lower results, with an accuracy of around 66% from 95%. Next, I show in Figure 8 that the model is performing much worse in differentiating left and right patients but still does fairly well in differentiating control from the rest. I can see that most of the controls are correctly labeled, only a few times are they labeled as Right TLEs. In the case of Left TLEs, they are the group that is most often mislabeled, often mislabeled as Right TLE and a few times as controls. In the case of Right TLEs, they are also mislabeled, but they are mainly mixed up with controls. I can also see that most of the time, the models prefer binary labels which will often mislabel one of the patient groups as the other.

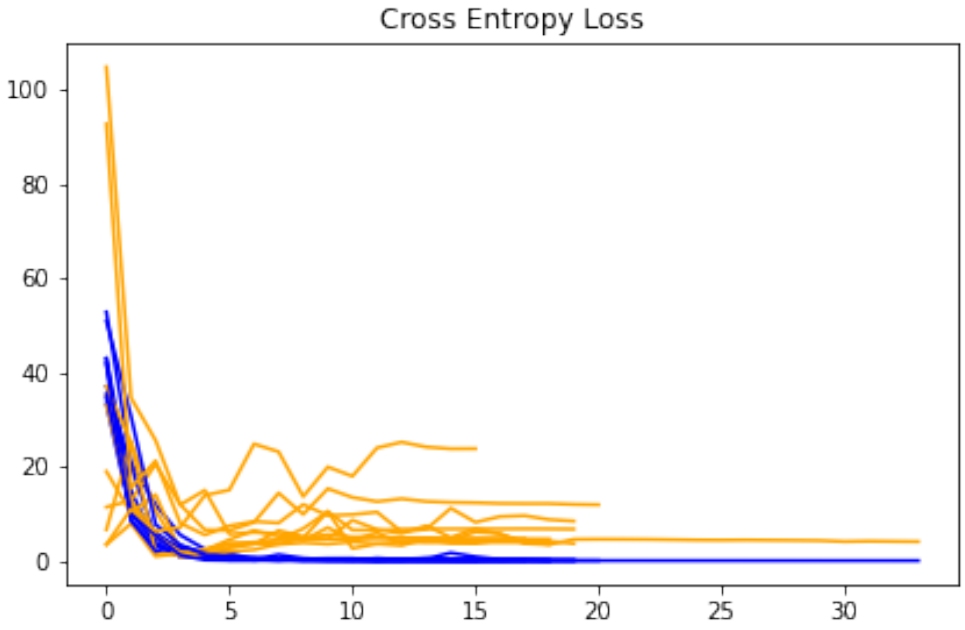


Figure 6: Validation model training loss. Loss of the replicated model showing that the model has been successfully implemented and is training correctly. Training set loss is in blue and validation set loss is in orange.

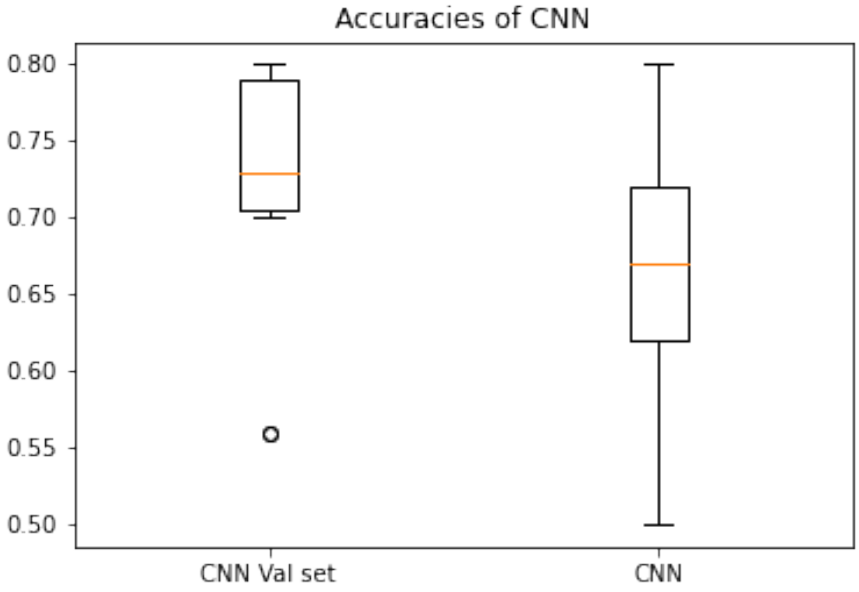


Figure 7: Validation model accuracy box plot. Box plots of the accuracies of the model on the new dataset. The left box plot is the accuracy of the validation set, and the right box plot is the accuracy of the testing set.

| split | Controls labeled Left TLE | Correctly labeled Controls | Controls labeled Right TLE | Left TLE labeled Controls | Correctly labeled Left TLE | Left TLE labeled Right TLE | Right TLE labeled Controls | Correctly labeled Right TLE | Right TLE labeled Left TLE |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 16.0 | 1.0 | 9.0 | 6.0 | 2.0 | 6.0 | 10.0 | 0.0 |
| 1 | 0.0 | 16.0 | 1.0 | 4.0 | 7.0 | 6.0 | 0.0 | 16.0 | 0.0 |
| 2 | 0.0 | 17.0 | 0.0 | 0.0 | 15.0 | 2.0 | 1.0 | 15.0 | 0.0 |
| 3 | 0.0 | 17.0 | 0.0 | 0.0 | 7.0 | 10.0 | 3.0 | 13.0 | 0.0 |
| 4 | 0.0 | 17.0 | 0.0 | 0.0 | 4.0 | 13.0 | 12.0 | 4.0 | 0.0 |
| 5 | 0.0 | 15.0 | 2.0 | 0.0 | 7.0 | 10.0 | 0.0 | 16.0 | 0.0 |
| 6 | 0.0 | 17.0 | 0.0 | 0.0 | 15.0 | 2.0 | 0.0 | 16.0 | 0.0 |
| 7 | 0.0 | 17.0 | 0.0 | 1.0 | 14.0 | 2.0 | 0.0 | 16.0 | 0.0 |
| 8 | 0.0 | 16.0 | 1.0 | 0.0 | 11.0 | 6.0 | 0.0 | 16.0 | 0.0 |
| 9 | 0.0 | 17.0 | 0.0 | 0.0 | 13.0 | 4.0 | 0.0 | 16.0 | 0.0 |

Figure 8: Validation model confusion matrix. There are 17 controls, 17 Left TLE, and 16 Right TLE per split

## 5 Novel model for Epilepsy detection and side of onset lateralization

With the results from the previous section, I concluded that the model is sufficient in separating controls and patients but fails to extract the features that can help it differentiate between the two patient subgroups. This is understandable as disease detection is a much easier task than epilepsy lateralization, there is just a more apparent difference in the brains of controls and patients than between different patient subpopulations. I think that the model will be able to perform better if the model is deeper. A deeper network will let it perform more feature extraction and be able to perform more tasks. The figure shows the new model architecture with more convolutional layers. Class weights are also changed, to help the model put more emphasis on differentiating between the patient subpopulations than between all the different classes.

### 5.1 Results

I show Figure 9 the new model architecture. In Figure 10, I show that this new model is also able to train correctly and minimize its loss function. Even though the validation loss does not change

as much, there is still a general downward trend. Figure 11 shows the box plots of the accuracies of this novel model on the task, it has an average test set accuracy of 71%.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv_1 (Conv2D) | (None, 160, 160, 8) | 80 |
| BN_1 (BatchNormalization) | (None, 160, 160, 8) | 32 |
| max_pooling2d_210 (MaxPoolin | (None, 80, 80, 8) | 0 |
| conv_2 (Conv2D) | (None, 80, 80, 8) | 584 |
| BN_2 (BatchNormalization) | (None, 80, 80, 8) | 32 |
| max_pooling2d_211 (MaxPoolin | (None, 40, 40, 8) | 0 |
| conv_3 (Conv2D) | (None, 40, 40, 16) | 1168 |
| BN_3 (BatchNormalization) | (None, 40, 40, 16) | 64 |
| max_pooling2d_212 (MaxPoolin | (None, 20, 20, 16) | 0 |
| conv_4 (Conv2D) | (None, 20, 20, 16) | 2320 |
| BN_4 (BatchNormalization) | (None, 20, 20, 16) | 64 |
| max_pooling2d_213 (MaxPoolin | (None, 10, 10, 16) | 0 |
| conv_5 (Conv2D) | (None, 10, 10, 32) | 4640 |
| BN_5 (BatchNormalization) | (None, 10, 10, 32) | 128 |
| max_pooling2d_214 (MaxPoolin | (None, 5, 5, 32) | 0 |
| conv_6 (Conv2D) | (None, 5, 5, 32) | 9248 |
| BN_6 (BatchNormalization) | (None, 5, 5, 32) | 128 |
| flatten_42 (Flatten) | (None, 800) | 0 |
| dropout_42 (Dropout) | (None, 800) | 0 |
| dense (Dense) | (None, 3) | 2403 |

Total params: 20,891
Trainable params: 20,667
Non-trainable params: 224

Figure 9: Model Architecture of the Novel CNN model. This model now has more convolutional layers, regularization is implemented on all layers due to the increase in layer number to control for overfitting.

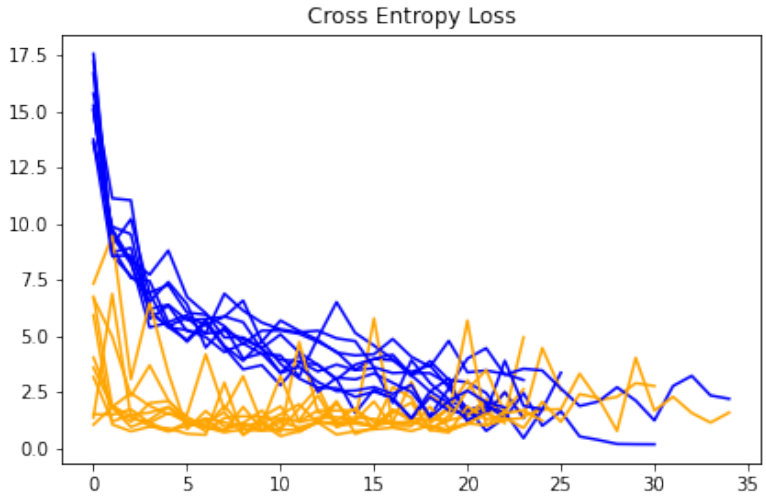


Figure 10: Novel model training loss. Loss of the replicated model showing that the model has been successfully implemented and is training correctly. Training set loss is in blue and validation set loss is in orange.

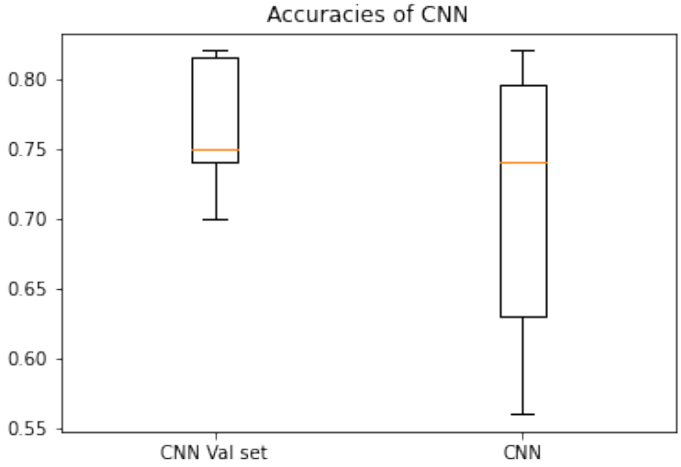


Figure 11: Novel model accuracy box plot. Box plots of the accuracies of the model on the new dataset. The left box plot is the accuracy of the validation set, and the right box plot is the accuracy of the testing set.

This new model is having improvements compared to the previous model with a performance of 71%, improved from 66%. The figure shows the accuracy box plot of the new novel CNN model. I then looked into the confusion matrix of this model, shown in Figure 12. I can see that it is still performing the task of labeling controls well. But here, the novel model is classifying left TLE much better than previously. I also see improvements in classifying Right TLE patients.

| split | Controls labeled Left TLE | Correctly labeled Controls | Controls labeled Right TLE | Left TLE labeled Controls | Correctly labeled Left TLE | Left TLE labeled Right TLE | Right TLE labeled Controls | Correctly labeled Right TLE | Right TLE labeled Left TLE |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 17.0 | 0.0 | 2.0 | 13.0 | 2.0 | 3.0 | 13.0 | 0.0 |
| 1 | 0.0 | 17.0 | 0.0 | 2.0 | 11.0 | 4.0 | 0.0 | 16.0 | 0.0 |
| 2 | 0.0 | 17.0 | 0.0 | 0.0 | 13.0 | 4.0 | 2.0 | 14.0 | 0.0 |
| 3 | 0.0 | 17.0 | 0.0 | 0.0 | 10.0 | 7.0 | 2.0 | 14.0 | 0.0 |
| 4 | 0.0 | 14.0 | 3.0 | 1.0 | 6.0 | 10.0 | 8.0 | 8.0 | 0.0 |
| 5 | 0.0 | 17.0 | 0.0 | 0.0 | 16.0 | 1.0 | 0.0 | 16.0 | 0.0 |
| 6 | 0.0 | 16.0 | 1.0 | 0.0 | 9.0 | 8.0 | 0.0 | 16.0 | 0.0 |
| 7 | 0.0 | 17.0 | 0.0 | 1.0 | 6.0 | 10.0 | 0.0 | 16.0 | 0.0 |
| 8 | 0.0 | 14.0 | 3.0 | 0.0 | 15.0 | 2.0 | 0.0 | 16.0 | 0.0 |
| 9 | 0.0 | 15.0 | 2.0 | 0.0 | 11.0 | 6.0 | 0.0 | 16.0 | 0.0 |

Figure 12: Novel model confusion matrix. There are 17 controls, 17 Left TLE, and 16 Right TLE per split

## 6 Discussion

In this study, I have correctly replicated a CNN model from a previous study to detect epilepsy from neuroimaging. I also added in a new patient subpopulation and tested the ability of that model to differentiate patient subpopulations as well as patients and controls. I see that the model is not well adapted for that task because of its low accuracy due to not being able to differentiate between the subpopulations. It mostly outputs two labels, one for controls and another for the general patient population, making it perform horribly in classifying the two patient subpopulations. I therefore modified the model to have more layers, theorizing that this will help the model extract the finer features that differentiate between the patient subpopulations but is not needed for classifying between patients and controls. The results of the new model supported my hypothesis that a deeper model can better differentiate between the patient subpopulations. The biggest improvement is in the fact that there is a more equal spread of prediction labels and not just outputting two labels means that the model is improving and is starting to differentiate between left and right TLE patients. There is still room for improvement in my model, it is still not differentiating right and left TLE patients as well as it can between controls and the rest. A future direction can be to explore a Multi-Task Learning model where it learns to first differentiate between patients and controls and then, differentiate between the patient subpopulations so that it will reduce the errors of miss-classifying patients as controls.

## References


[1] et al. E. Gleichgerrcht. “Radiological identification of temporal lobe epilepsy using artificial intelligence: a feasibility study”. In: *Brain Communications* (2021), p. 284. DOI: 10.1093/brain-comms/fcab284.

[2] Kunihiko Fukushima. “Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position”. In: *Biological Cybernetics* (1980), pp. 193–202. DOI: 10.1007/BF00344251.

[3] et al. G. D. Jackson. “Hippocampal sclerosis can be reliably detected by magnetic resonance imaging”. In: *Neurology* (1990), pp. 1869–1875. DOI: https://doi.org/10.1212/WNL.40.12.1869.

[4] D. H. Hubel and T. N. Wiesel. “Receptive fields and functional ar- chitecture of monkey striate cortex”. In: *The Journal of Physiology* (1968), pp. 215–243. DOI: 10.1113/jphysiol.1968.sp008455.

[5] Frank Rosenblatt. “The Perceptron: A Probabilistic Model For Information Storage And Organization in the Brain”. In: *Psychological Review* (1958), pp. 386–408. DOI: 10.1037/h0042519.

[6] Arthur Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* (1959), pp. 210–229. DOI: 10.1147/rd.33.0210.

[7] Wei Zhang. “Computerized detection of clustered microcalcifications in digital mam- mograms using a shift-invariant artificial neural network”. In: *Medical Physics* (1994), pp. 517–524. DOI: 10.1118/1.597177.