12-2021

Developing Deep-Learning Methods for Diagnosis and Prognosis of Pediatric Progressive Diseases Using Modern Imaging Techniques

Mahdieh Shabanian
University of Tennessee Health Science Center

Developing Deep-Learning Methods for Diagnosis and Prognosis of Pediatric Progressive Diseases Using Modern Imaging Techniques

Abstract

Purpose and Rationale. Central nervous system manifestations form a significant burden of disease in young children. There have been efforts to correlate the neurological disease state in tuberous sclerosis complex (TSC) neurological disease state with imaging findings is a standard part of patient care. However, such analysis of neuroimaging is time- and labor-intensive. Automated approaches to these tasks are needed to improve speed, accuracy, and availability. Automated medical image analysis tools based on 3D/2D deep learning algorithms can help improve the quality and consistency of image diagnosis and interpretation for cognitive disorders in infants. We propose to automate neuroimaging analysis with artificial intelligence algorithms. This novel approach can be used to improve the accuracy of TSC diagnosis and treatment. Deep learning (DL) is among the most successful types of machine learning and utilizes deep artificial neural networks (ANNs), which can determine efficient feature representations of input data. DL algorithms have created new opportunities in medical image analysis. Applications of DL, specifically convolutional neural networks (CNNs), in medical image analysis, cover a broad spectrum of tasks, including risk prediction/estimation with a machine learning system trained on these classification tasks.

Study population. We reviewed an NIMH Data Archive (NDA) dataset that was collected in 2010. We also reviewed imaging data from patients and normal cases from birth to 8 years of age acquired at Le Bonheur Children's Hospital from 2014 to 2020. The University of Tennessee Health Science Center Institutional Review Board (IRB) approved this study.

Research Design and Study Procedures. Following Institutional Review Board (IRB) approval, this thesis: 1) Presents the first 2D/3D fusion CNN models to estimate the age of infants from birth to 3 years of age. 2) Presents the first work to look at whole-brain network to automatically distinguish TSC brain structural pathology from normal cases using a 3DCNN model.

Conclusions. The study findings indicate that deep neural networks tackle the problem of early prediction of cognitive and neurodevelopmental disorders and structural brain pathology based on MRI automatically in TSC children. It is the hope of the author that analysis of MRI images via methods of deep learning will have a positive impact on healthcare for infants and children at risk of rare diseases.

Document Type

Thesis

Degree Name

Master of Science (MS)

Program

Biomedical Engineering

Research Advisor

John J. Bissler, MD

Keywords

2D CNN, 3D CNN, Brain disease, deep learning, Infant, MRI

UNIVERSITY OF TENNESSEE HEALTH SCIENCE CENTER

MASTER OF SCIENCE THESIS

Developing Deep-Learning Methods for Diagnosis and Prognosis of Pediatric Progressive Diseases Using Modern Imaging Techniques

---

Author:
Mahdieh Shabanian

Advisor:
John J. Bissler, M.D.

A Thesis Presented for The Graduate Studies Council of
The University of Tennessee Health Science Center
in Partial Fulfillment of Requirements for the Master of Science Degree
In the Joint Graduate Program in Biomedical Engineering
From The University of Tennessee
and The University of Memphis

Biomedical Engineering
College of Graduate Health Sciences

December 2021

This work is dedicated to the practicing and aspiring healthcare providers who support children and families in some of their hardest days at Le Bonheur Children's Hospital.

Acknowledgements

First, I would like to thank my mentor, Dr. John Bissler, who always listened and entertained my ideas and questions with enthusiasm and grace. He was essential in helping me strive for excellence while maintaining work-life balance. He guided me in conducting strong research that was meaningful and clinically relevant. I would also like to thank Dr. John DeVincenzo for his 2 years of support and funding for this research. Thanks to Dr. Courtney Bricker-Anthony for editorial assistance.

Thanks to my committee members, Drs. Curry and Eckstein, for their guidance and advice on my research, writing, and so much more. They were a well-balanced team who supported me through research setbacks, teaching woes, and life changes. I am a more well-rounded and capable individual for their time, attention, and input throughout my doctoral work.

Finally, thanks to all of my family and friends for all the love and support they have given to me over the years. First, I would like to thank my parents for teaching me to always to work to the best of my abilities and never give up on my dreams. They are the reason I still have joy in this work. Their dedication and hard work that I have gotten to witness firsthand, in clinical work and research, gives me strength to continue fighting for our children. We got each other through graduate school and I could not have had a better time.

Abstract

Mahdieh Shabanian

Developing Deep-Learning Methods for Diagnosis and Prognosis of Pediatric Progressive Diseases Using Modern Imaging Techniques

Purpose and Rationale. Central nervous system manifestations form a significant burden of disease in young children. There have been efforts to correlate the neurological disease state in tuberous sclerosis complex (TSC) with imaging findings as a standard part of patient care. However, such analysis of neuroimaging is time- and labor-intensive. Automated approaches to these tasks are needed to improve speed, accuracy, and availability. Automated medical image analysis tools based on 3D/2D deep-learning algorithms can help improve the quality and consistency of image diagnosis and interpretation of cognitive disorders in infants. We propose to automate neuroimaging analysis with artificial intelligence algorithms. This novel approach can be used to improve the accuracy of TSC diagnosis and treatment. Deep learning (DL) is among the most successful types of machine learning and utilizes deep artificial neural networks (ANNs), which can determine efficient feature representations of input data. DL algorithms have created new opportunities in medical image analysis Applications of DL, specifically convolutional neural networks (CNNs) in medical image analysis, cover a broad spectrum of tasks, including risk prediction/estimation with a machine learning system trained on these classification tasks.

Study Population. We reviewed an NIMH Data Archive (NDA) dataset that was collected in 2010. We also reviewed imaging data from patients and normal cases from birth to 8 years of age acquired at Le Bonheur Children's Hospital from 2014 to 2020. The University of Tennessee Health Science Center Institutional Review Board (IRB) approved this study.

Research Design and Study Procedures. This thesis 1) Presents the first 2D/3D fusion CNN models to estimate brain age of infants from birth to 3 years of age. It also 2) Presents the first work to look at the whole-brain network to automatically distinguish TSC brain structural pathology from normal cases using a 3D CNN model.

Conclusions. The study findings indicate that deep neural networks tackle the problem of early prediction of cognitive and neurodevelopmental disorders and structural brain pathology based on MRI automatically in TSC children. It is the hope of the author that

analysis of MRI images via methods of deep learning will have a positive impact on healthcare for infants and children at risk of rare diseases.

Contents

List of Tables

List of Figures

List of Abbreviations

ACC	accuracy
ANN	artificial neural network
ASD	autism spectrum disorder
cCMV	congenital cytomegalovirus
CNN	Convoutional Neural Network
CP	cerebral palsy
DTI	diffusion tensor imaging
DL	Deep Learning
FC	fully connected
FSPGR	spoiled gradient-echo
HIE	hypoxic ischemic encephalopathy
HSVE	herpes simplex virus encephalitis
IRB	Institutional Review Board
MRI	magnetic resonance imaging
NDDs	Neurodevelopmental disorders
NIfTI	Neuroimaging Informatics Technology Initiative
PDW	PD-weighted
PPV	Positive predictive value
RT	Repetition time
TNR	Negative rate
TSC	Tuberous sclerosis complex
TPR	True positive
T1W	T1-weighted
T2W	T2-weighted
XAI	EXplainable Artificial Intelligence

Chapter 1

Deep Neural Networks in Medical Imaging

1.1 Introduction

Traditionally, machine learning models are trained to perform useful tasks based on features manually extracted from raw data or are engineered using other classical machine learning models. In deep learning (DL) networks, computers learn high-level features automatically, directly from raw data, bypassing an often difficult feature engineering step. Automatic feature detectors represent a significant difference between DL algorithms and more classical machine learning. Yann et al. published a review paper titled "deep learning" in Nature to help readers understand the fundamental strengths of deep learning algorithms (LeCun, Bengio, and G. Hinton, 2015; Litjens et al., 2017), and also provided a general overview of deep learning algorithms in radiology (Litjens et al., 2017). Since 2019 DL algorithms, specifically convolutional neural networks (CNNs), have become widely used for analyzing medical images. In medical imaging, the interest in DL is mostly driven by CNNs that can uncover non-obvious, useful features of images. With a deep neural network, many image features are typically preserved in image analysis. Some potent preferences are embedded in CNNs: W. Wang et al., 2019; Yamashita et al., 2018, explain how CNNs provide powerful image analysis while lessening pre- and post-processing tasks needed to extract higher level features that reduce an image to its key features, thus enabling easier classification (Shabanian et al., 2019). Various types of CNNs were developed recently, and all have the potential to contribute to the accuracy and speed of image classification or automatic segmentation. CNNs can learn high-level hierarchies of features automatically by backpropagating errors through multiple blocks and layers, such as convolutional layers, max-pooling layers, and fully connected layers used for image classification tasks (W. Wang et al., 2019).

1.2 Structure of Convolutional Neural Networks

CNNs are able to learn spatial hierarchies of features automatically and adaptively using multiple building blocks, such as convolutional layers, pooling layers, and fully connected layers. In fact, CNN is trained through backpropagation and gradient descent to generate an error signal that measures the difference between the estimation of the network and the

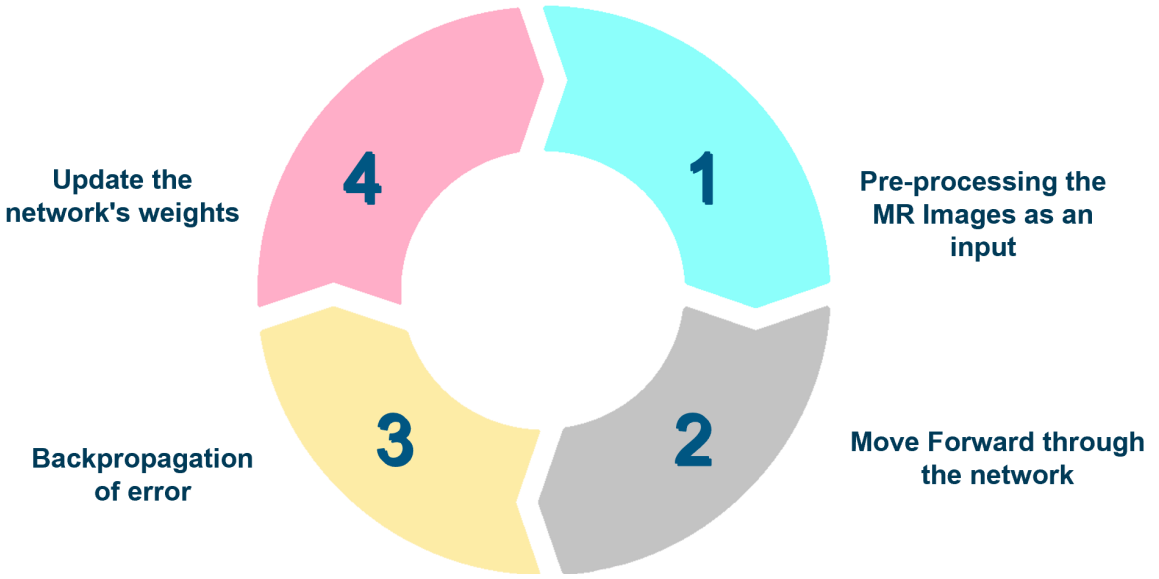


Figure 1.1: CNN architecture in simple word.

target value (**Figure**. 1.1). A CNN uses this error signal to change the weights or parameters for more accurate classification or segmentation (Nielsen, 2015). CNNs typically have several convolution layers, pooling layers, and fully connected (FC) layers at the end, which compute the final outputs (**Figure**. 1.2). The performance of the model is calculated under weights and kernels with a loss function through forward propagation on a training dataset. All learnable parameters, such as weights and kernels are updated based on the loss value through backpropagation with a gradient descent optimization algorithm.

1.2.1 Hardware and Software

Training CNNs usually requires a graphic processing unit (GPU) with its memory matched to the number of parameters in the network. GPUs led to the remarkable rise of DL. Technically, GPUs increase speed by 10 to 50 times of CPU-based work for training deep neural networks. The AlexNet developed by Krizhevsky et al. owes its success to its architecture (Krizhevsky, Sutskever, and G. E. Hinton, 2017), in which the power of calculation is based on the GPU. Several open-source DL libraries provide efficient GPU implementations of deep neural networks, including one focused on convolutions (LeCun, Bengio, and G. Hinton, 2015). Other distinct deep learning frameworks include ones by Google Research which released TensorFlow in 2015 (Abadi et al., 2015); it provides C++ and Python interfaces and is used by Google AI. Google's engineers developed Keras (Chollet, 2015) , a popular framework that provided the Python interface in 2017. Facebook's AI Research lab (FAIR) developed and released PyTorch (Paszke et al., 2019) for the first time in 2016.

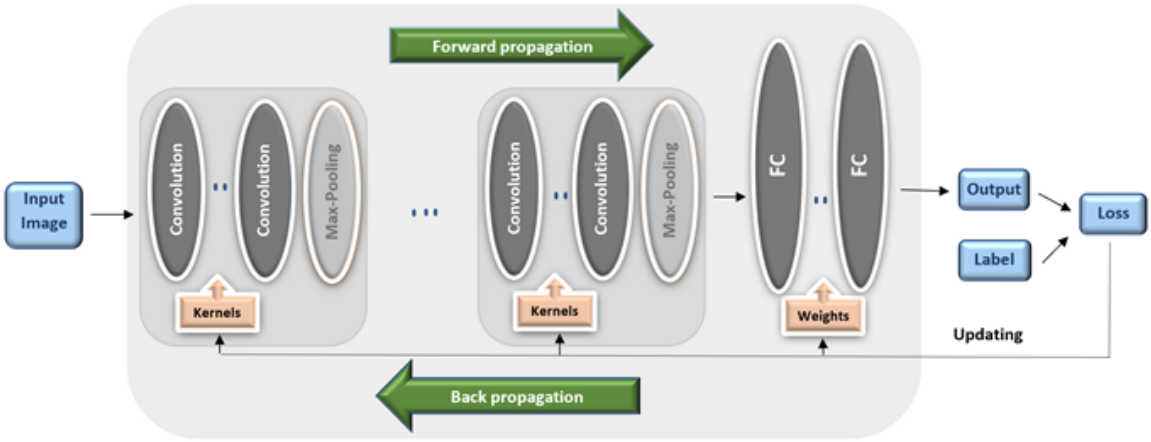


Figure 1.2: CNN architecture in the training process.

Figure. 1.3 shows the popularity between Tensorflow and PyTorch as two excellent frameworks for developing and researching deep learning applications in 5 past years through Google Trends. TensorFlow framework is a powerful, mature and popular deep learning library in part because it has strong visualization capabilities. TensorFlow framework has several options for high-level model development in both classification and segmentation tasks. PyTorch is new framework that is very popular in medical imaging fields and is gaining momentum fast. We worked with TensorFlow(Keras) and PyTorch in this research.

PyTorch and Keras are both excellent for deep neural networks applied for imaging tasks. Both frameworks extract top features of image sets, thereby making it challenging to select which one is better in medical tasks. Indeed, Keras consist of more mature library but PyTorch has powerful libraries especially relevant in medical imaging field.

1.2.2 Deep-Learning Algorithms in Biomedical Image Analysis

Many image diagnosis tasks require an initial search to identify abnormalities. Measurement of lesions is often done, a task that can have significant variability between radiologists. If there are prior images, then any changes over time are closely examined. Automated medical image analysis tools based on DL algorithms could be helpful tools in adding objective, algorithmic elements to image interpretation. This approach has opened new doors in medical image analysis. DL applications in medical image analysis are available for many imaging modalities, including X-Ray, CT and MRI. Especially comprehensive, useful surveys of deep learning in medical imaging include these: (LeCun, Bengio, and G. Hinton, 2015; Yamashita et al., 2018; Mazurowski et al., 2019).

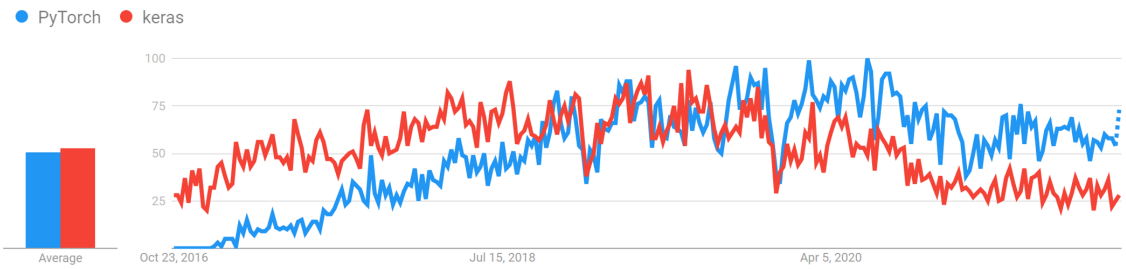


Figure 1.3: Popularity of Keras versus Pytorch in the 5 past years. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. A score of 0 means there was not enough data for this term (*Google Trends - Interest in Pytorch Verses Keras Over Five Years* 2021). Data source: Google Trends (`https://www.google.com/trends`)

1.2.3 2D versus 3D CNN Model

The development of 3D CNNs remains at an early stage because of their complexity, especially in medical image interpretation. Around 2018-2019, exponential growth was seen in using 3D deep learning models in medical imaging (Singh et al., 2020). Their review paper also described the development of 3D CNN from machine learning roots in detail.

Convolutions of 2D CNNs are used to derive features only from spatial dimensions in 2D feature maps. 2D CNNs take a 2D matrix as an input, such as image slices. Therefore, information from adjacent slices is unavailable. With this approach, information is more likely to be lost from interest regions of the brain in specific diseases because the brain is a 3D structure. **Figure**. 1.4 *a* shows the 2D filter (kernel) in 2D CNN used to extract spatial-spectral features.

The 3D CNN architectures using volumetric convolutions are very useful DL methods for analyzing volumetric imaging data. The 3D CNN mathematical model is very similar to 2D CNN and is obtained by adding one extra dimension. Volumetric image analysis is a time-consuming process that currently requires expert knowledge, especially for medical images such as MRI images that use the Neuroimaging Informatics Technology Initiative (NIfTI) format (Ueda et al., 2019; Cole et al., 2017). For the first time, Cole et al. showed that 3D-CNN could accurately estimate developmental brain age from MRI data using healthy adult samples as control images. Additionally, the authors demonstrated 3D CNNs are more effective and less likely to miss regions of interest in medical images of the brain.

3D convolution applies a 3D filter (kernel) to the dataset, and this filter moves in 3-directions (x, y, z) to calculate low-level feature representations at particular sites.

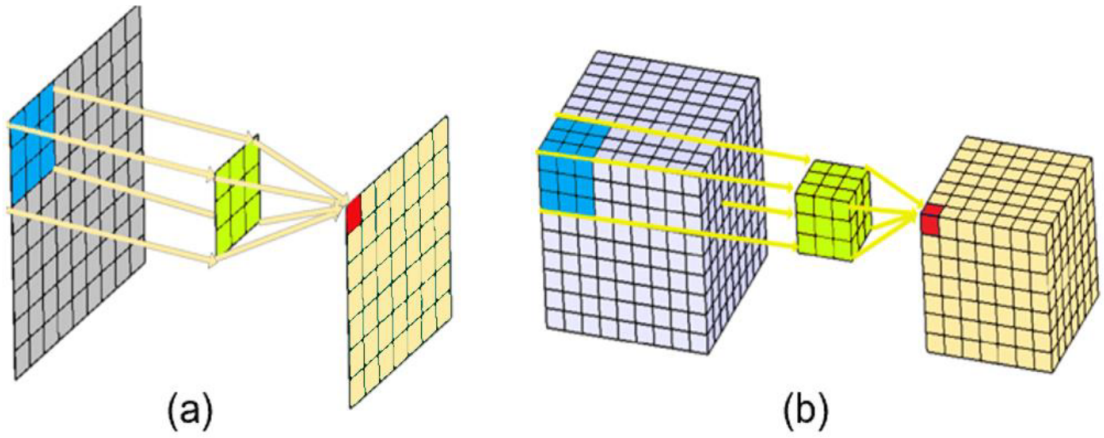


Figure 1.4: Illustration of convolution. (a) illustration of a 2D filter to extract spatial features and (b) illustration of a 3D filter to extract spatial-spectral features.

Figure. 1.4 *b* shows that 3D CNNs filters help prevent loss of the region of interest that occurs in 2D medical images. CNNs apply a filter to the input to create a feature map to summarize detected features in the input. Filters, such as line detectors, edge locations, and curved lines as sections of surfaces, are portions of CNN-innovations that are learned during training in the context of a multi classification problem. Statistical tests indicated that the 3D CNN model was able to classify 2D MRIs more accurately than the 2D CNN model. 3D CNNs can use entire image volume rather than individual slices such as 2D CNNs; this feature includes more information (e.g., relationships between consecutive slices). Implementation of 3D models can also be more difficult on commonly available GPU cards due to their large memory consumption. A 3D model has better performance when sufficient training data and powerful computational hardware are available. Several articles demonstrated the performance of 3D CNN on MRI brain scans (Jnawali et al., 2018; Gao, Hui, and Tian, 2017). Using 3D CNN models, we can now observe subtle micro- and macro-structural changes, as well as aberrant connectivity variances that may go undetected at an age where early intervention could have impacted patient care. In addition, these findings may be undetectable by the human eye until much later in the child's development, preventing implementation of an early treatment plan.

1.3 Specific Aims and Hypotheses

The primary purpose of this research was to develop novel deep learning models in medical imaging analysis based on MRIs to classify neurodevelopmental age and detect brain abnormalities in children. We showed in this research how to use the novel deep learning

models on a small number of samples, and we compared 2D and 3D models in the following chapters.

We were guided by the following:

Aim 1. We hypothesize that a custom fusion deep learning approach (a 3D Convolutional Neural Network, 3D CNN) could be used to differentiate age-appropriate brain development from retarded development based on multi-modal MRI sequences. As it will be shown, this is not the case, and we examine possible reasons by providing different types of reduced information to the DL approach: 3D data, 2.5D data, and 2D data. The ultimate aim of this work is to approach a continuous (rather than age-group stratified) estimation of the developmental brain age from MRI. But even a classification into age groups will help identify patients with developmental delay and serve as an objective finding that could help lessen inter-observer and intra-observer variability. More objective findings could also be used to risk-stratify different patient populations.

Aim 2. We hypothesize that custom 3D deep learning models can help in the early identification of TSC anatomical abnormalities and tuber features using standard brain MRI sequences for TSC cases at risk of developing epilepsy (and subsequent neurodevelopmental delay). Neuroimaging is important for detecting TSC brain lesions that can contribute to the neurological disease process, even during fetal development. Although neuroimaging analysis is time-consuming and labor-intensive, automated deep learning approaches to neuroimaging analysis could improve detection of TSC structural brain pathologies by comparison with MRI from normal subjects. Accurately detecting epileptogenic zones in TSC cases may assist physicians with more rapid and reliable identification of TSC cases at high risk of epilepsy.

Chapter 2

Neurodevelopmental Disorders

2.1 Introduction

Determining if the brain is developing normally is a key component of pediatric neuroradiology and neurology. Brain magnetic resonance imaging (MRI) of infants demonstrates a specific pattern of development beyond simply myelination. While radiologists have used myelination patterns, brain morphology and size characteristics to determine age-adequate brain maturity, this requires years of experience in pediatric neuroradiology. With no standardized criteria, visual estimation of the structural maturity of the brain from MRI before three years of age remains dominated by inter-observer and intra-observer variability. A more objective estimation of brain developmental age could help physicians identify many neurodevelopmental conditions and diseases earlier and more reliably. Such data, however, is naturally hard to obtain, and the observer ground truth is not much of a gold standard due to subjectivity of assessment.

Neurodevelopmental disorders (NDDs) are a diverse group of conditions characterized by delayed milestones involving cognition, communication, behavior, and motor skills. Infancy and early childhood are characterized by rapid cognitive development, especially in the first three years of life. This cognitive development is mirrored by changing brain structure, function, and connectivity. Many pediatric diseases impair this development. Therefore, brain-developmental-age estimation is crucial to determine if a child's brain is developing normally. Magnetic resonance imaging (MRI) based neuroimaging of infants offers qualitative information such as myelination patterns and brain morphology. MRI also offers quantitative information such as head circumference, brain volume, and water content. Neuroimaging is critical for determining the impact of pediatric brain diseases, in particular when patient images are compared to a healthy clinical sample of normal brain development, to help categorize the patient with many infant diseases, including: a) prematurity Van Bel, Vaes, and Groenendaal, 2019, b) hypoxic ischemic encephalopathy (HIE) of the newborn Shetty et al., 2019, c) congenital cytomegalovirus (cCMV) infection Grinberg et al., 2019, d) bacterial meningitis, e) herpes simplex virus encephalitis (HSVE) Ramirez et al., 2018, f) pediatric epilepsy,Gupta et al., 2020 g) cerebral palsy (CP), h) tuberous sclerosis complex (TSC) Sánchez Fernández et al., 2020 and many other genetic and

non-genetic disorders.

2.2 Neurodevelopmental Disorders in Children

Newborns with brain disorders are at increased risk of severe long-term motor deficit and cognitive delay. Detecting these conditions earlier, even before birth, could help health care professionals and prevent mitigate morbidity. The developing brain's distinct morphological changes are the basis of image-based brain age determination. In light of observer variability and the complex information for humans must assess, computer-based automated brain-developmental age estimation (BDAE) could offer an important automated second opinion in patients with mild cognitive impairment and/or a tool to monitor neurodevelopment in patients with known chronic neurologic diseases.

For infants, both imaging and computational approaches to BDAE have yet to achieve an acceptable level of consensus. At 38-40-weeks of normal in utero development, normal newborn features are found on MRI, the brain's pattern has nearly normal adult sulcal pattern. Corticospinal tracts are hyperintense on T1w compared to the rest of the brain. On T2w images, we have hopointensity in the dorsal brain stem. this results come from high water content and lack of myelination, is significantly different from older infants. In the first year of life, myelination occurs very rapidly with gray matter migrating from its origin in the periventricular region to the cortex. Anatomical, histological, and functional properties change rapidly thereafter and lead to many computational challenges not seen in adult models. Neonatal brain development, as observed by MRI, can be roughly divided into four distinct temporal stages Paus et al., 2001:

1. Newborn pattern; myelination only present along the posterior limbs of the internal capsule and perirolandic regions. The major sulci and gyri are well developed. Myelination involves the genu of the corpus callosum, although a majority of the T1w and T2w signal remains switched around 3 months. The majority of the white matter is hyperintense in T2w, making it difficult to differentiate from cerebral edema.
2. 12 months pattern; Characterized by further progression of myelination anteriorly (5-9 months of age), superiorly, and laterally. The splenium of the corpus callosum becomes myelinated during this time. Around 12 months, the rate of myelination drops off, with tissue contrast more closely resembling a fully developed brain on T1-weighted imaging.
3. 24 months pattern; After 2 years, essentially the adult myelination pattern is seen on T1-weighted imaging, however T2-weighted sequences will still have areas of bright signal that are not found in an adult.
4. 36 months pattern; T2 hyperintense signals consistent with white matter tracts that have yet to be myelinated can be found in the young adult, especially in the frontal lobes.

Without biomarkers to diagnose NDD or quantitative assessment of neurological pathology in infants, these NDDs are identified through physical exams and imaging tests. Then improvements in the identification of MRI abnormalities is needed in infants to predict long term damage and to assign earlier interventions. Automated BDAE algorithms applied to infants who were imaged with MRI in clinical care could thus provide a reproducible, automated second opinion to the clinicians' assessment. Current pediatric neuroradiology assesses the age of the patient predominantly from myelination of the brain and brain volume. Such an approach suffers from interobserver and intraobserver variability. Artificial intelligence algorithms may make it possible to evaluate early signs of delayed age more objectively and quantitatively for each patient. Deep learning (DL) is one of the most successful variants of machine learning, utilizing deep artificial neural networks (ANNs). These hierarchically organized ANNs are able to learn efficient feature representations of input data Mostapha and Styner, 2019; Jahangard, Zangooei, and Shahedi, 2020; Hosseini, Chen, and Jablonski, 2020; Cui et al., 2020. State-of-the-art BDAE approaches use manually extracted features, preventing subsequent machine learning approaches from fully exploiting the content inherent in sequences of MR images J. Wang et al., 2014; Franke et al., 2010; Lao et al., 2004.

We hypothesize that a custom 3D deep learning approach (a 3D Convolutional Neural Network, 3D CNN) could be used to differentiate age-appropriate brain development from retarded development based on multi-modal MRI sequences. As it will be shown, this is not the case, and we examine possible reasons by providing different types of reduced information to the DL approach: 3D data, 2.5D data, and 2D data. The ultimate aim of this work is to approach a continuous (rather than age-group stratified) estimation of the developmental brain age from MRI. But even a classification into age groups will help to identify patients with developmental delay and serve as an objective finding that could help eliminate interobserver and intraobserver variability. It could also be used to risk-stratify different patient populations.

2.3 Deep Learning Based Modeling of Brain Age

The fundamental strengths of deep learning algorithms LeCun, Bengio, and G. Hinton, 2015, as well as their utility in radiology Litjens et al., 2017 are undisputed. deep learning can be considered the state of the art methodology for image classification, which in this work we want to explore for the task at hand. DL algorithms, specifically convolutional neural networks (CNNs), have become widely used for analyzing medical images. The inferential bias of CNNs explains how CNNs provide powerful image analysis, W. Wang et al., 2019; Yamashita et al., 2018, while lessening pre- and post-processing tasks, thus enabling easier classification. Various architectures of CNNs were developed over the past 10 years, offering potential to increase the accuracy and speed of image classification or automatic segmentation (Sarvamangala and Kulkarni, 2021).

Though conceptually convincing, the use of 3D CNNs for medical image analysis tasks is challenging because of their computational complexity. Nevertheless, 3D deep learning models are being used increasingly in medical imaging Singh et al., 2020, which is particularly relevant for classification tasks. Otherwise, the per-slice classification information derived from a 2D CNN would need to be integrated secondarily, or even be disregarded entirely by only classifying a predetermined single slice. A 3D CNN architecture, on the other hand, is the natural choice for analyzing volumetric imaging data. 3D CNN have been shown to accurately estimate developmental brain age from MRI data using healthy adult samples Cole et al., 2017. Statistical tests and multiple publications indicate that 3D CNN models were able to classify MRIs more accurately than the 2D CNN models in sufficiently large datasets. Hong et al., 2020; Shabanian et al., 2019; Jnawali et al., 2018; Gao, Hui, and Tian, 2017. In this chapter, however, we show that a 2D CNN achieves a performance superior to that of the best 3D network we designed for the available cases of rare genetic and neurological disorders and the concomitant images. In the meantime, we work with radiology to obtain images in similar cases meeting similar medical classifications, even though they are rare.

2.4 Materials and Methods

2.4.1 Dataset

We obtained 552 normal MRI scans (equaling 184 sets of T1w, T2w, and PDw sequences) of 84 infants from the NIMH Data Archive (NDA). Some records were for infants who had several exams at different ages. In all training-validation splits, we ensured that these patients were always assigned to training or validation sets only. The patients ranged in age from 8 days to 3 years, spanning a critical developmental period to enable early diagnosis of neurological sequelae. The infants were scanned with 1.5T MRI while awake or during natural sleep without sedation. MRI acquisition generally lasted 30–45 minutes on a 1.5T scanner with a 2D sequence that minimized scan duration for the newborn to 3 years group. The axial scans consisted of a 2D T1-weighted (T1W) spin echo and a T2-weighted (T2W) 2D turbo spin-echo sequence. The T1W sequences utilized repetition time (TR) 500, echo time (TE) 12, 90-degree flip angle, and 3-mm slice thickness. The T2W and PD-weighted (PDW) scans utilized TR 3500, TE 15-17 (115-119), and 3-mm slice thickness. The T1W and T2W scans were nominally 1×1×3 resolution (1×1×3 or .97×.97×3) with a matrix of 256 x 192 mm. Most scans were obtained using a Siemens Medical Systems (Sonata, Magnetom) scanner, another site used a GE Signa Excite scanner (GE Healthcare, Chicago, IL) to obtain less than half of the scans. Sanchez, Richards, and Almli, 2012.

All our models were trained on a personal computer (NVIDIA TITAN RTX GPU, Python 3.7.9, TensorFlow 2.1.0). [1]

[1] Codes are available in Appendix C

2.4.2 Model Details

For our estimation of brain developmental age, we designed custom 2D, and 3D fusion CNN architectures. We slightly downsampled all MRIs to 250x250x40 voxels and we cropped a center region to 150×150×20 voxels for 3D model and to 150×150 for 2D model to reduce computational complexity while maximizing the amount of information retained from the original resolution.

Our 3D proposed architecture has four scale level blocks consisting of following parts:

a. 2-2-2-1 (3x3x3) convolutional layers followed by

b. 2x2x2 max pooling layers,and a global average pooling layer before

c. three fully-connected (FC) layers that lead into

d. the SoftMax four-class output layer

The convolutional blocks feature 32, 64, 96, and 128 channels, respectively. We used two blocks of 2x2 convolutional layers, four 64 channels and two 128 channels for our 2D proposed model with 2x2 max pooling and three FC layers as a simple model. Following best practices, cross-entropy divergence loss and the Adam optimization algorithm (lr=0.001) were used for the loss function and optimizer, respectively in both the 2D (368,580 parameters) and the 3D model(392,516 parameters). The model weights were randomly initialized. Since no separate test data was available, we used a stratified 5-fold cross-validation scheme to train and validate all CNN models. In 84 patients with partially multiple visits, this resulted in 139-154 images in the training set and 30-45 images in the validation set. We performed data augmentation to reduce potential overfitting and monitored training and validation loss accordingly. Each image in the training set of both models was augmented by L/R flipping, static rotations around the z axis by $\pm$15 degree, and random rotations of $\pm$15-45 degree. **Table** 2.1 and **Table** 2.2 show the 2D and 3D fusion CNN model in detail.

2.5 Exprimental Evaluations

To measure the performance of classification of MRIs in each category, we calculated six metrics in the entire validation dataset: positive predictive value (PPV), true positive/negative rate (TPR, TNR), F1-score (FS), accuracy (ACC), area under the receiver operating characteristic curve (AUROC) and Matthew's Correlation Coefficient (MCC). These metrics are useful to characterize the performance of models in multi-classification tasks on imbalanced datasets, where accuracy is a misleading metric if reported alone (Vidiyala, 2020). We reported the metrics according to the four classes of brain developmental age and the classification errors in a normalized confusion matrix for the most accessible visualization.

Table 2.1: 2D CNN model for early fusion.

Layer (type)	Output Shape	Param #
conv2d_18 (Conv2D)	(None, 148, 148, 64)	1792
conv2d_19 (Conv2D)	(None, 146, 146, 64)	36928
batch_normalization_9 (Batch	(None, 146, 146, 64)	256
max_pooling2d_9 (MaxPooling2	(None, 73, 73, 64)	0
dropout_15 (Dropout)	(None, 73, 73, 64)	0
conv2d_20 (Conv2D)	(None, 71, 71, 64)	36928
conv2d_21 (Conv2D)	(None, 69, 69, 64)	36928
batch_normalization_10 (Batc	(None, 69, 69, 64)	256
max_pooling2d_10 (MaxPooling	(None, 34, 34, 64)	0
dropout_16 (Dropout)	(None, 34, 34, 64)	0
conv2d_22 (Conv2D)	(None, 32, 32, 128)	73856
conv2d_23 (Conv2D)	(None, 30, 30, 128)	147584
batch_normalization_11 (Batc	(None, 30, 30, 128)	512
max_pooling2d_11 (MaxPooling	(None, 15, 15, 128)	0
dropout_17 (Dropout)	(None, 15, 15, 128)	0
global_average_pooling2d_3 (	(None, 128)	0
dense_9 (Dense)	(None, 128)	16512
dropout_18 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 128)	16512
dropout_19 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 4)	516

Table generated by Python.

Table 2.2: 3D CNN model for early fusion.

Layer (type)	Output Shape	Param #
conv3d_29 (Conv3D)	(None, 148, 148, 18, 32)	2624
conv3d_30 (Conv3D)	(None, 146, 146, 16, 32)	27680
batch_normalization_20 (Batc	(None, 146, 146, 16, 32)	128
max_pooling3d_18 (MaxPooling	(None, 73, 73, 16, 32)	0
max_pooling3d_19 (MaxPooling	(None, 36, 36, 8, 32)	0
dropout_24 (Dropout)	(None, 36, 36, 8, 32)	0
conv3d_31 (Conv3D)	(None, 34, 34, 6, 64)	55360
conv3d_32 (Conv3D)	(None, 32, 32, 4, 64)	110656
batch_normalization_21 (Batc	(None, 32, 32, 4, 64)	256
max_pooling3d_20 (MaxPooling	(None, 16, 16, 4, 64)	0
dropout_25 (Dropout)	(None, 16, 16, 4, 64)	0
conv3d_33 (Conv3D)	(None, 14, 14, 2, 96)	165984
batch_normalization_22 (Batc	(None, 14, 14, 2, 96)	384
max_pooling3d_21 (MaxPooling	(None, 7, 7, 2, 96)	0
dropout_26 (Dropout)	(None, 7, 7, 2, 96)	0
global_average_pooling3d_4 (	(None, 96)	0
dense_15 (Dense)	(None, 128)	12416
dropout_27 (Dropout)	(None, 128)	0
dense_16 (Dense)	(None, 128)	16512
dropout_28 (Dropout)	(None, 128)	0
dense_17 (Dense)	(None, 4)	516

Table generated by Python.

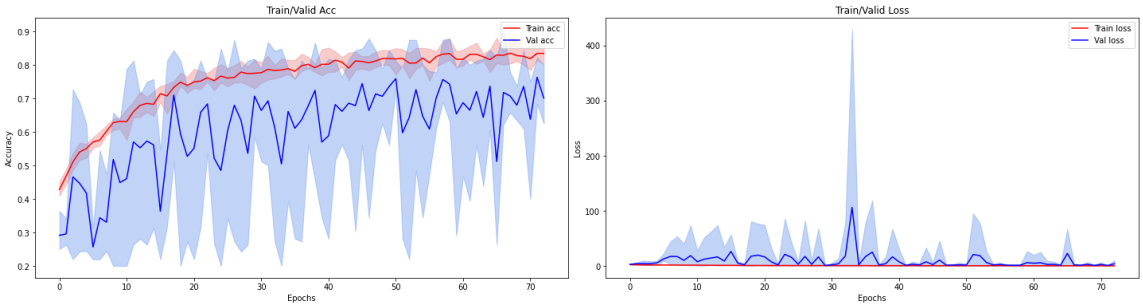


Figure 2.1: Accuracy/loss in 2D CNN in T1w, averaged for 5 folds.

Table 2.3: Overall statistical results in T1w.

	Overall Statistical Results/Metrics					
CNN Model	**Acc**	**TPR**	**PPV**	**TNR**	**F1**	**MCC**
2D CNN	0.81	0.81	0.79	0.94	0.80	0.74
2.5D CNN	0.58	0.58	0.63	0.85	0.59	0.43
3D CNN	0.77	0.77	0.78	0.92	0.78	0.69

2.6 Results

2.6.1 Baseline Experiment

Several studies used only T1-weighted MRI volumes to estimate brain age using CNN Cole et al., 2017; Jnawali et al., 2018; Ueda et al., 2019. Therefore, we initially performed a corresponding experiment using only the T1w scans in our proposed CNN models. We compared them through 5-fold cross-validation similar to the methodology used when evaluating the fusion models. Dropout (0.7) and (0.2) used respectively in first and second FC layers and batch normalization further help to improve model convergence.

The training progress of this model is pictured in **Figure** 2.1 in terms of training/validation accuracy and loss averaged over the five folds. Training resulted in a micro-averaged 81% validation accuracy. **Table** 2.3 provides the accuracy (Acc), recall (TPR), precision (PPV), specificity (TNR), F1-score and MCC for this and the compared inferior 2.5D and 3D CNNs. We observe that in our setting, the 2D CNN model outperforms the more complex architectures. In small validation test datasets, we hypothesize that complex architectures like 3D CNN models are not an appropriate option.

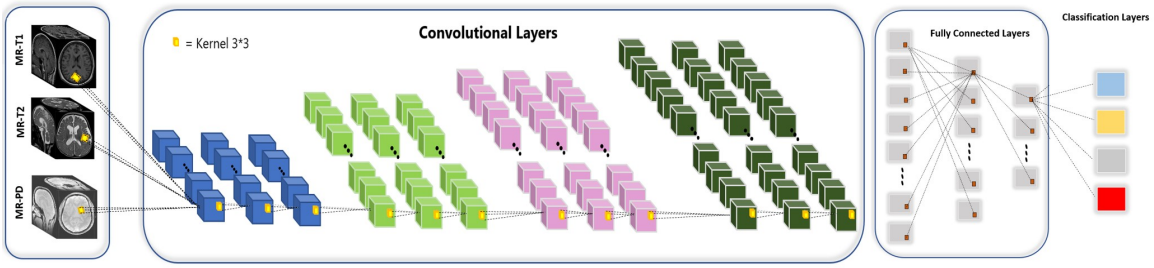


Figure 2.2: 3D CNN architecture for early fusion.

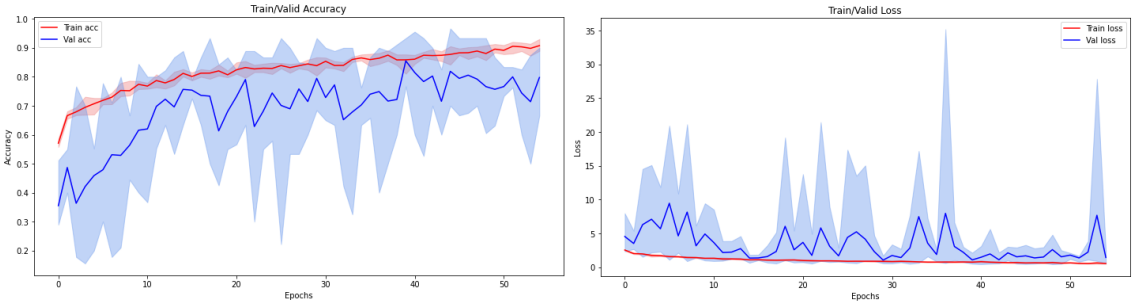


Figure 2.3: Accuracy/loss in 2D fusion CNN, averaged for 5 folds.

2.6.2 3D Fusion CNN versus 2D Fusion CNN in Brain Age Classification

The early fusion of this data was to concatenate the three MRI sequences into the channel dimension of the input image. We trained our proposed fusion CNN models multiple times before selecting a set of hyperparameters that we used in all subsequent experiments. **Table** 2.4 shows the detailed metrics for this and the compared 3D in fusion models.The proposed 2D fusion model outperformed the 3D model at a 90% accuracy across the cross-validated validation.

Figure 2.2 shows the 3D CNN architecture using the early fusion strategy on sets of 184 fusion inputs consisting of MR-T1w, T2w and PDw MRI sequences. Our approach to early fusion of this data is to concatenate the three MRI sequences into the channel dimension of the input image. Note that there is considerable motion between the contrast which we didn't account for through image registration. We trained our proposed fusion CNN models multiple times before selecting a set of hyperparameters (epochs to train, learning rate schedule, early stopping criteria etc.) that were used in all subsequent experiments.

The averaged training/validation accuracy and loss of the 2D fusion CNN model is depicted in **Figure** 2.3, and **Table** 2.4 shows the detailed metrics for this and the compared

Table 2.4: Overall statistical results in the fusion CNN models.

CNN Model	Overall Statistical Results/Metrics							
	Acc	**TPR**	**PPV**	**TNR**	**AUROC**	**F1**	**MCC**	**Kappa 95% CI**
2D CNN	0.90	0.90	0.90	0.97	0.99	0.90	0.86	(0.79-0.92)
2.5D CNN	0.87	0.87	0.86	0.92	0.98	0.87	0.83	(0.74-0.90)
3D CNN	0.86	0.86	0.85	0.91	0.98	0.86	0.83	(0.75-0.89)

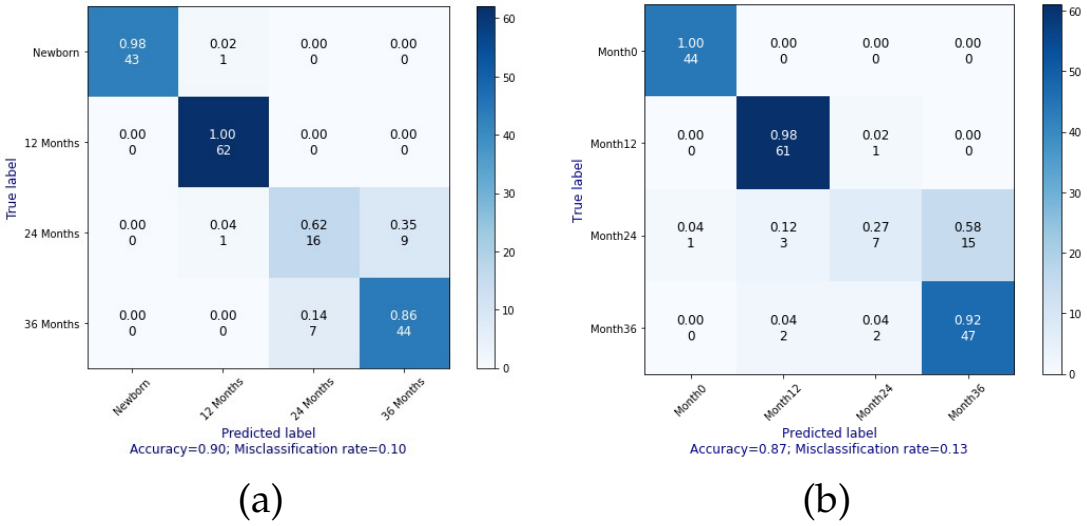


Figure 2.4: Confusion matrices for (a) 2D and (b) 2.5D fusion CNN.

2.5D and 3D in fusion models, followed by the normalized confusion matrices in **Figure** 2.4 and **Figure** 2.5. Consistent with the results on the T1w data alone, the proposed 2D fusion model outperformed the other architectures at a 90% accuracy across the cross-validated validation.

We observed that using the fusion of common three MRI sequences improved performance in estimating brain developmental age. Despite the small dataset, the proposed simple 2D fusion CNN model achieved high accuracy in estimating brain developmental stage accurately. The fusion of multiple MRI sequences helped to extract more hierarchical features from the images. In infants, T1w, T2w and PD sequences were crucial to estimation of brain development. We compared the overall statistical results of the proposed 2D and 3D CNN models using fusion of MRI sequences. While 3D CNNs are supposed to yield higher performance for analyzing volumetric data, we achieved better performance using

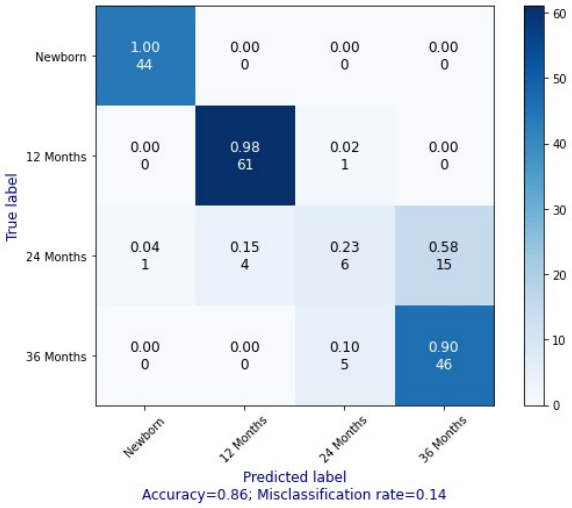


Figure 2.5: Confusion matrix for 3D fusion CNN.

the 2D fusion CNN model. We hypothesize that in cases where very little data is available, it is not possible to extract predictive information from the convolutional layers in the 3D model, since the variability of the ”normal” is too high to be captured in the limited dataset. Therefore, the use of the 2D model provides an advantage, although it is likely that in a scenario with more data the 2D model might be surpassed by a 3D model. CNN fusion models might be able to detect subtle micro- and macro-structural changes and aberrant connectivity variances that might go undetected at an age where early intervention could impactpatient care. In addition, these findings may be undetectable by the human eye until much later in the child's development, preventing implementation of an early treatment plan. This paper illustrated that better performance could be obtained by passing the whole MRI volume from three MRI sequences into the 2D fusion CNN model.

2.7 Summary of Results

We compared the overall statistical results of the proposed 2D, 2.5D, and 3D CNN models using T1w and fusion of MRI sequences. While analysis of volumetric data using 3D CNNs would be expected to perform better, we achieved better performance using the 2D fusion CNN model. We hypothesize, also based on the additional 2.5D experiment, that in a setting with only few data, the extraction of predictive information in the convolutional layers is not possible in the 3D model, since the variability of the "normal" is too large to be captured in the limited data. Therefore, the 2D model has an advantage, though it is likely that in a scenario with small data, it might be surpassed by a 3D model. Further statistical

Table 2.5: Overall statistics in T1w/fusion MRI using 2D CNN model.

	Overall Statistical Results/Metrics						
CNN Model	**Acc**	**TPR**	**PPV**	**TNR**	**AUROC**	**F1**	**MCC**
2D CNN /T1w	0.81	0.81	0.79	0.94	0.98	0.80	0.74
2D Fusion CNN	0.90	0.90	0.90	0.97	0.99	0.90	0.86

results showed – much in line with the clinical hypothesis – that the fusion of multiple MR sequences improves the performance of age estimation in infants (**Table** 2.5).

Stacking multiple MRI sequences might help to find high-level features associated with age from different MRI sequences. Therefore, CNN fusion models might be able to detect subtle micro- and macro-structural changes, as well as aberrant connectivity variances that may go undetected at an age where early intervention could have impacted patient care. In addition, these findings may be undetectable by the human eye until much later in the child's development, preventing implementation of an early treatment plan. This research showed better performance by passing in the whole MRI volume from three MRI sequences into the 2D fusion CNN model.

2.8 Discussion

Our counterintuitive finding was that a small model (in terms of depth and number of trainable parameters) outperformed more complex models with higher numbers of levels and parameters. Our result needs to be replicated on a larger, less redacted dataset in future research and the special value of working with 2D images needs to be elaborated upon with respect to voxel size and volume of tissue in the voxels. Ablation studies may provide exemplary methods for selecting optimal model complexity with regard to the classification task and the available dataset(s). The major goal of this work is to show that 2D CNN works better in the small number of samples. The original dataset has four age classes, even though the number of cases was too small specifically in 24 months to achieve reliable performances. Despite the small dataset, our method achieved high accuracy, and therefore suggests that further work is warranted to test whether BDAE using 2D fusion CNN might have a role in patient care.

Based on these results, 2D fusion CNNs could be used to determine the trajectory of normal brain development and neurodevelopmental age within the first three years of life. This approach could also prove useful of fusion multiple MR sequences in identifying otherwise undetectable abnormalities in brain development to improve performance.

Chapter 3

TSC Structural Brain Pathology Detection

3.1 Introduction

Of people born with mutations causing tuberous sclerosis complex (TSC) and demonstrating symptoms in childhood, neurological involvement is a leading cause of death. Such neurological involvement, including epilepsy, can cause significant long-term sequelae in children (Jóźwiak, 2021; Hulshof, Benova, et al., 2021). Recent research demonstrates a correlation between tuber load and outcome. Although this relationship is complex, tubers are associated with epilepsy (Cohen et al., 2021), affecting more than 90% of patients with TSC and may become intractable. Brain involvement in TSC can be detected by magnetic resonance imaging (MRI). Still, neuroimaging analysis is time- and labor-intensive, begging the need for automated approaches to these tasks to improve speed, accuracy, and availability. In this light, we explored the general feasibility of using three-dimensional convolutional neural networks (CNNs) to automatically enhance image diagnosis quality and consistency to identify anatomical abnormalities in TSC children. We trained the 3D CNN on axial T1-weighted, axial T2-weighted FLAIR, and 3D T1-FSPGR weighted images from 296 TSC and 245 Normal cases from birth to 8 years of age, acquired at Le Bonheur Children's Hospital, were acquired with IRB permission. In the best performing DL approach, our model had an accuracy of 0.82 [95% CI:0.75-0.90] with 0.90% AUC.

3.2 Tuberous Sclerosis Complex in Children

Tuberous sclerosis complex (TSC), a rare autosomal dominant multisystem disorder, exhibits a phenotype that is both age-dependent and highly variable. Central nervous system manifestations form a significant burden of disease in young children with TSC (Henske et al., 2016; Curatolo, Moavero, and Vries, 2015). Epilepsy affects 90% of TSC patients, and 70% experience their first seizure during the first years of life (Gupta et al., 2020). Up to 60% of TSC patients develop drug-resistant epilepsy (Gupta et al., 2020; Jeong, Nakagawa, and Wong, 2017), and 50% of patients exhibit intellectual disability, behavioral problems, and autism spectrum disorder (ASD) (Curatolo, Moavero, and Vries, 2015; Jeste et al., 2016; Torre-Ubieta et al., 2016; Capal et al., 2017). Clearly, intervention to limit the neurological issues for patients with TSC represents a significant unmet need. Early identification and treatment

of infants with TSC at risk of developing epilepsy (and subsequent neurodevelopmental delay) greatly improve outcomes. The international TSC guidelines call for MRI evaluation of the child's brain at diagnosis (Northrup et al., 2013). Neuroimaging is important for detecting TSC brain lesions that can contribute to the neurological disease process, even during fetal development (Hulshof, Slot, et al., 2021). Although neuroimaging analysis is time-consuming and labor-intensive, automated approaches to neuroimaging analysis could improve detection of lesions that can lead to poor outcomes.

We propose to automate neuroimaging analysis with artificial intelligence algorithms. This novel approach can be used to improve the accuracy of TSC diagnosis and treatment. Deep learning (DL) is among the most successful types of machine learning and utilizes deep artificial neural networks (ANNs), which can learn to efficiently extract and classify feature representations of input data (Mostapha and Styner, 2019; Jahangard, Zangooei, and Shahedi, 2020; Cui et al., 2020). DL algorithms have created new opportunities in medical image analysis. Applications of DL, specifically convolutional neural networks (CNNs), in medical image analysis cover a broad spectrum of tasks, including risk prediction/estimation. With a machine learning system trained on this classification task, TSC patients at risk for epilepsy might receive more tailored or earlier intervention based on an imaging study included in their existing diagnostic work-up.

3.3 Materials and Methods

3.3.1 Data

We received Institutional Review Board (IRB) approval for this study, which was conducted at the TSC Center of Excellence at Le Bonheur Children's Hospital. We collected existing longitudinal data from 98 TSC patients with multiple visits (2 to 11 visits) and 245 control subjects with one visit. In total, we reviewed multiple MRI sequences from 296 TSC cases with epilepsy and 245 controls from birth to 8 years of age who were imaged between 2014 and 2020. TSC patients and control subjects were imaged on a 1.5T GE scanner, 1.5T Toshiba Scanner, 3T GE scanner, or 3T Siemens Scanner. The inclusion criteria for the control group were children who had one brain MRI scan performed at Le Bonheur Children's Hospital which was interpreted as normal interpretation by a pediatric neuroradiologist .

We reviewed one 3D and two 2D sequences available in Neuroimaging Informatics Technology Initiative (NIfTI) image volume format. The 3D images were obtained with a sagittal T1-fast spoiled gradient-echo (FSPGR) sequence (TR/TE: 9-12/3-4.5; acquisition matrix: 256 × 256; field of view: 22 mm^2; inversion time: 450.00 milliseconds; flip angle: 13°; slice thickness: 1.2-0.8 mm with no space). The first 2D acquisition was an axial fast spin-echo T2-weighted fluid attenuated inversion recovery (FLAIR) sequence (TR/TE: 8,000-10000/120-130; acquisition matrix, 512 × 512; field of view: 21 mm^2; slice thickness: 4.0–5.0 mm with 1.0-mm space). A T1-weighted sequence completed the protocol (TR/TE: variable/8-10; inversion time: 1,111 milliseconds; acquisition matrix: 256 × 256; field of view: 21 mm^2; slice thickness: 4.0 mm with 1.0-mm space).

Table 3.1: Number of TSC and Normal Cases in each MRI sequence.

MRI Sequences	#TSC Cases	#Normal Cases
AX T2-Flair	266	246
AX T1	186	149
Sagittal T1-FSPGR	266	238

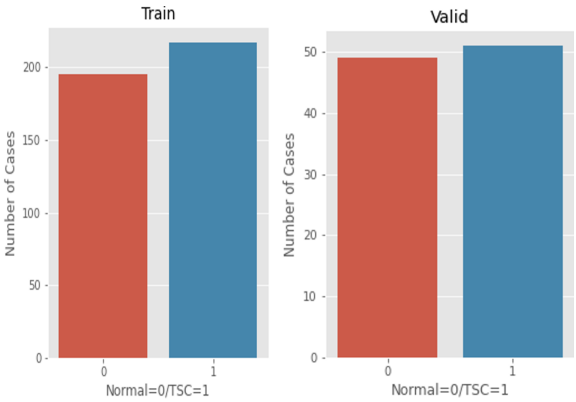


Figure 3.1: TSC/Normal balanced Train and valid dataset in AX T2 Flair.

The numbers of TSC and Normal cases that we reviewed in these three common MRI sequences are different. **Table** 3.1 and **Figure** 3.1 show the exact number of TSC and Normal cases that had these three common MRI sequences in this research.

3.3.2 Model Details

We made all MRIs isotropic (1.0, 1.0, 1.0) and same size (200x200x100) for our model. Our proposed 3D architecture has four scale-level blocks comprising 2-1-1-1 (3x3x3) convolutional layers, 2x2x2 max-pooling layers, a global average pooling layer, and three fully-connected (FC) layers that lead into SoftMax. The convolutional blocks have 32, 64, 128, and 215 channels. We used batch normalization, dropout, regularization and global average pooling and three FC layers. Following best practices, binary cross-entropy divergence loss and the SDG optimization algorithm (lr=0.0001) were used for the loss function and optimizer, respectively in this proposed 3D CNN architecture.

Table 3.2 and **Figure** 3.2 and show the proposed 3D CNN model, which has 2,068,117 trainable parameters. The model weights were randomly initialized. We had no separate test data available, so we divided 296 TSC and 245 Normal cases into training (80%) and validation (20%) data sets in each class. Then, we trained the same 3D CNN model for each

Table 3.2: 3D CNN model (TS3DCNN) in binary TSC task.

Layer (type)	Output Shape	Param #
conv3d (Conv3D)	(None, 200, 200, 100, 32)	896
conv3d_1 (Conv3D)	(None, 198, 198, 98, 32)	27680
batch_normalization (BatchNo	(None, 198, 198, 98, 32)	128
max_pooling3d (MaxPooling3D)	(None, 99, 99, 49, 32)	0
conv3d_2 (Conv3D)	(None, 97, 97, 47, 64)	55360
batch_normalization_1 (Batch	(None, 97, 97, 47, 64)	256
max_pooling3d_1 (MaxPooling3	(None, 48, 48, 23, 64)	0
conv3d_3 (Conv3D)	(None, 46, 46, 21, 128)	221312
batch_normalization_2 (Batch	(None, 46, 46, 21, 128)	512
max_pooling3d_2 (MaxPooling3	(None, 23, 23, 10, 128)	0
conv3d_4 (Conv3D)	(None, 21, 21, 8, 215)	743255
batch_normalization_3 (Batch	(None, 21, 21, 8, 215)	860
max_pooling3d_3 (MaxPooling3	(None, 10, 10, 4, 215)	0
global_average_pooling3d (Gl	(None, 215)	0
dense (Dense)	(None, 1024)	221184
batch_normalization_4 (Batch	(None, 1024)	4096
dense_1 (Dense)	(None, 512)	524800
batch_normalization_5 (Batch	(None, 512)	2048
dropout (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 512)	262656
batch_normalization_6 (Batch	(None, 512)	2048
dropout_1 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 2)	1026

2,068,117 total parameters. Table generated by Python.

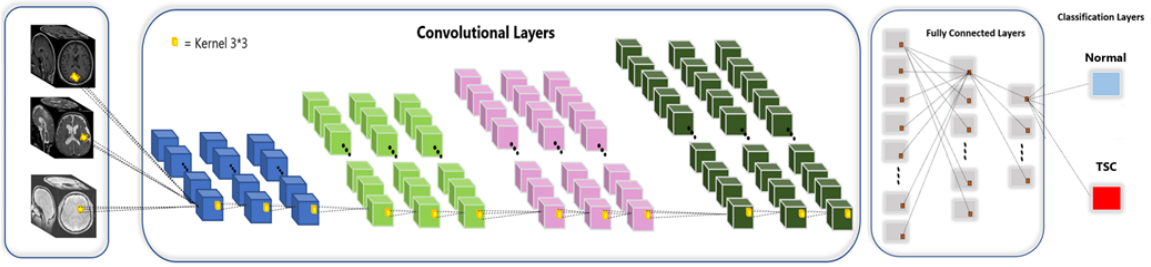


Figure 3.2: 3D CNN model in binary task.

MRI sequence and compared their results to identify TSC structural pathology like tubers and nodules (Shahid, 2013).

3.4 Experimental Evaluations

To measure the model's performance in MRI classification in each category, we calculated five metrics in the entire validation data set: positive predictive value (PPV), true positive/negative rate (TPR, TNR), F1-score (FS), accuracy (ACC), and area under the receiver operating characteristic curve (AUROC). These metrics are useful to characterize the performance of models in classification tasks on imbalanced data sets where accuracy is a misleading metric if reported alone (Vidiyala, 2020). We reported the metrics according to the four brain developmental age classes. Classification errors are shown in a normalized confusion matrix. All models were implemented on an NVIDIA TITAN RTX GPU using Python 3.7.9 and TensorFlow 1.15.0. [1]

3.5 Results

Based on cortical and subcortical tubers that will appear based on MRI signal intensity in the frontal, parietal, temporal, and occipital cortex, we have examined the 3D CNN model to identify anatomical abnormalities in TSC children from Normal using only one MRI sequence. We compared the performance of three effective MRI sequences, such as T1-weighted, T2-weighted FLAIR, and T1-FSPGR (1-mm isotropic voxels), which are the best sequences for visualizing cortical tubers and detecting abnormal cortical development in volumetric analyses. Some types of tubers are isointense on T1w and hyperintense on T2w Flair. And other types of tubers are hypointense on T1w and homogenously hyperintense or heterogenous on T2w Flair. Furthermore, the recent progress in 3D T1 FSPGR sequences makes them valuable to identify anatomical abnormalities in TSC children.

[1] Codes are available in Appendix D.

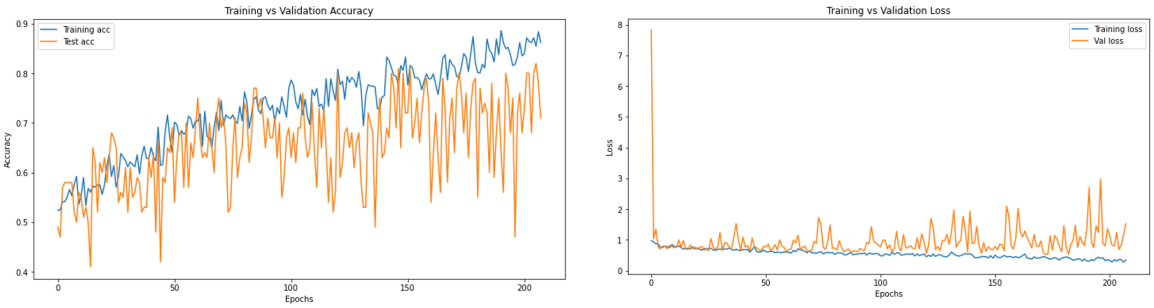


Figure 3.3: Accuracy/loss in 3D CNN using T2w Flair.

Table 3.3: Overall statistical results in TS3DCNN models in each MRI sequence.

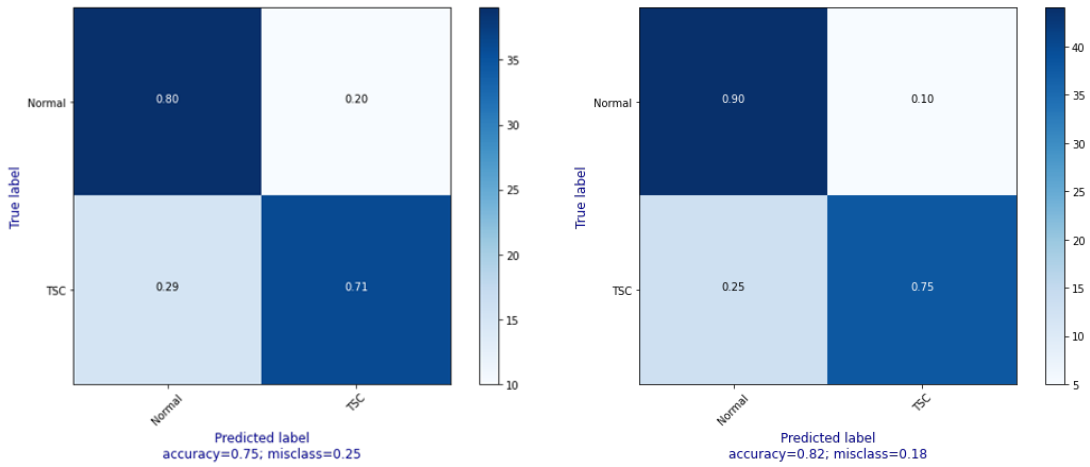
	Overall Statistical Results/Metrics						
CNN Model	**Acc**	**TPR**	**PPV**	**TNR**	**AUROC**	**F1**	**Kappa 95% CI**
AX T2-FLAIR	0.82	0.82	0.82	0.80	0.90	0.82	(0.50-0.80)
AX T1	0.75	0.75	0.75	0.73	0.86	0.75	(0.75-0.89)
T1-FSPGR	0.71	0.71	0.71	0.70	0.77	0.72	(0.10-0.76)

The training/validation accuracy and loss for T2w is depicted in **Figure** 3.3. **Table** 3.3 shows the detailed metrics for comparing the three sequences that we reviewed to distinguish TSC anatomical abnormalities from normal, followed by the normalized confusion matrices in **Figure** 3.4 and **Figure** 3.5. T2-FLAIR showed higher accuracy in diagnosing TSC cases than T1-weighted and T1-FSPGR.

3.6 Comparison with BCH 2D TSCCNN Model

We compared our proposed TS3DCNN model with TSCCNN model from Boston Children's Hospital (BCH) (Sánchez Fernández et al., 2020). They used some 2D slices from T2 and FLAIR axial MRIs for training the 2D CNN model. Their 2D CNN model consisted of: 2-2-2-6 (2x2) convolutional layers followed by 2x2x2 max pooling layers that convolutional blocks feature 64, 128, 256, and 512 channels, respectively; it included a GlobalAveragePooling layer to flatten the input. They trained their 2D neural network with a batch size of 64 on 5634 samples in training,and validate on 248 samples. They achieved 83% accuracy in validation dataset **Figure** 3.6.

Compared to our proposed TS3DCNN model, they used a deeper model and more samples in training and validation, because they used two MRI sequences. We observe that



(a) (b)

Figure 3.4: Confusion matrices for (a) T1-weighted, (b) T2-weighted FLAIR.

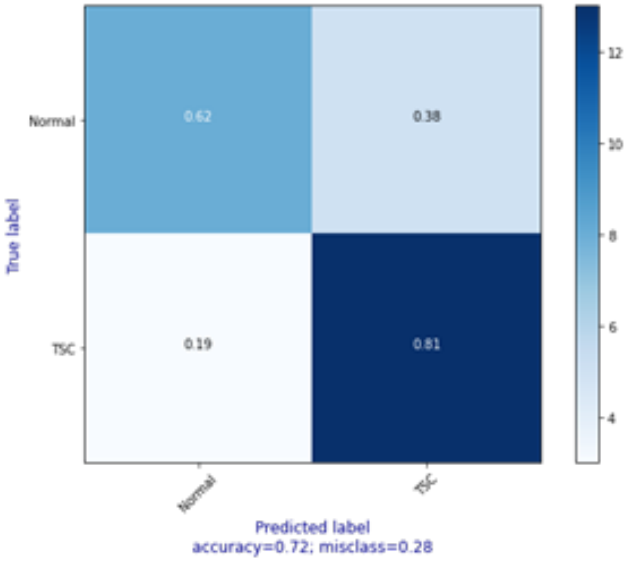


Figure 3.5: T1 FSPGR confusion matrix.

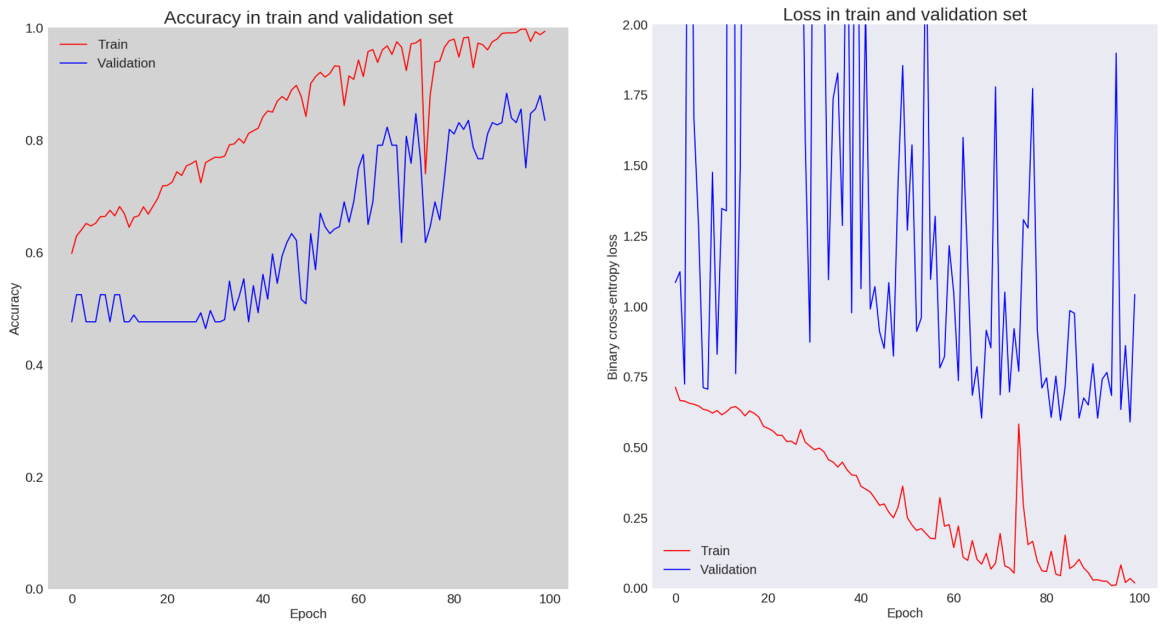


Figure 3.6: Accuracy/Loss in TSCCNN model.

---

the 3D CNN models improves performance in estimating over using one MRI sequence. 3D models help to extract more hierarchical features from images.

3.7 Discussion

To our best knowledge, this study is the first to demonstrate that 3D CNNs can be used to identify TSC anatomical abnormalities and tuber features using standard brain MRI sequences for TSC cases. An automatic 3D deep neural model helps to enhance image diagnosis quality to detect anatomical abnormalities related to tubers in TSC children. Our approach provides results from T1-weighted, T2-weighted FLAIR, and 3D T1-FSPGR brain MRI to detect TSC cases from normals using the proposed 3D CNN model. Accurately detecting likely epileptogenic tuber zone in TSC cases may assist physicians with more rapid and reliable identification of TSC cases at high risk of epilepsy.

Chapter 4

Conclusions, Discussion and Future Directions

4.1 Conclusions

Neuroimaging is vital for detecting brain lesions that may contribute to the neurologic disease process, especially during infant brain development (Hulshof, Slot, et al., 2021). There is an unmet need to optimize the anatomical identification of lesions, leading to deleterious neurological outcomes and seizures in children. Prediction of anatomical abnormalities at birth is still very challenging because of confounding factors in infant brain development, including genetic and epileptogenic. It is especially difficult in infants due to the small brain sizes, low image resolution and, artifacts due to motion, and especially rapid structural changes during the first 3 months of age. Delayed brain development in infants causes long-term neurological disorders, the impact of which can be lessened by early intervention, especially during the first 3 years of age. Analysis of neuroimages is time- and labor-intensive and in addition requires extensive training. In this situation effective, automated approaches to the analysis tasks can improve the clinicians speed, accuracy, and ability to bring the information inherent in MRI images to the diagnostic and prescriptive aspects of providing patient care.

To approach this problem, we propose to leverage the high-dimensional data and computing capabilities using machine learning. This approach is emerging to better understand pathways to diagnose and treat disease accurately. We used deep learning to develop an approach to optimize the anatomical pathology-neurological outcome relationship. Deep learning has gained an increasingly critical role in brain MRI (LeCun, Bengio, and G. Hinton, 2015; Yamashita et al., 2018; Mazurowski et al., 2019). Through neuroimaging, deep learning can assist in identifying disease markers and translate them into windows of opportunities for interventions in diagnosis, treatment, and management of neurological disorders in children with TSC. Convolutional neural networks (CNNs) have shown promise in a variety of applications, including automatic diagnosis, biomarker identification, early detection, and risk assessment for neurological conditions in children. CNNs are a type of deep learning approach that automatically learns to detect patterns of interest in images. Convolutional Neural Network is an excellent model for medical tasks when data are limited. We addressed this critical need by leveraging the information contained in the high-dimensional

data using the Fusion CNN model. This approach has rapidly emerged as a new trend to better understand pathways and accurately diagnose and treat disease.

Prediction of anatomical abnormalities at birth is still very challenging because of some confounding factors in infant brain development, including genetic and epileptogenic. It is especially difficult in infants due to the small brain sizes, low image resolution and, artifacts due to motion, and also rapid structural change across first 1-3 months of age (discussed in Chapter 2). Despite these challenges, tuberous sclerosis complex is a multisystem genetic disorder with a high prevalence of neurodevelopmental comorbidities and a high risk of early-onset epilepsy in children. Approximately 90% of patients with TSC develop medically refractory epilepsy. Refractory epilepsy can increase the risk of impaired neurodevelopment and early death. Epilepsy affects 90% of TSC patients and about 70% experience their first seizure during the first years of life. Up to 60% of TSC patients develop drug-resistant epilepsy, and intellectual disability, behavioral problems, and autism spectrum disorder are reported to affect 50% of patients. Prediction of long-term neurological disorders and epileptogenic abnormality within the first few years of age especially remains a desirable goal that would be helpful to obtain better treatment planning and more informed assessment of patient outcomes. CNN models are useful when important features are too complex to be detected directly, e.g., epileptogenic abnormality.

Our most astonishing finding is perhaps that a small fusion deep learning model (in terms of depth and number of trainable parameters) performs much better than attempts with deeper and complex models. Based on these results, Simple fusion CNNs using stacking effective whole MRI volume from multiple MRI sequences could be used to determine the trajectory of identifying brain development disorders in rare genetic, neurological disorders in small datasets (chapter 2). Usually, each subject has multiple MRI sequences that our proposed model can detect effective sequences to diagnose (chapter 3) and then fuse them as an input. Fusion is an intuitive approach for linking and integrating relevant and complementary information from multiple MRI sequences into one fused body of data to accurately diagnose in the early stage of neurodevelopmental disorders. Therefore, with fused volumetric information, we seek via DL to obtain models of lesser complexity and depth. In this light, we explored the general feasibility of using fusion CNN model to automatically enhance image diagnosis quality and consistency to identify anatomical abnormalities in children in the early stages.

Also, this approach could prove that fusion of multiple MR sequences is useful in efforts to identify undetectable abnormalities in brain development in children. In addition, these findings may be undetectable by the human eye until much later in the child's development, preventing implementation of an early treatment plan. CNN fusion models might be able to detect subtle micro- and macro-structural changes and aberrant connectivity variances that might go undetected at an age where early intervention could impact patient care in early stages. Early identification and treatment of infants at risk of developing neurodevelopmental delay greatly improve outcomes. While these findings

offer amazing promise and hope for better patient outcomes, the logistics of implementing this approach to a world-wide population of infants with a rare disease are complicated.

4.2 Recent Deep-Learning Methods in Medical Imaging

Deep learning applications have received much attention because they can surpass humans in many tasks, especially in image analysis. However, deep neural networks lack reliability and explainability. Without these properties, few would comfortably trust them in medical imaging tasks. And so, deep learning algorithms are considered as imperfect black boxes. The complex underlying mechanism of deep learning models is difficult to understand. Methods categorized as explainable artificial intelligence (XAI) methods could tackle these aforementioned issues. Therefore, explainable artificial intelligence has become a hot research topic these days in deep learning models, especially in medical tasks. Providing explanations is the aim of XAI methods in deep learning models to be rational and understandable by humans, physicians, and radiologists in medical imaging tasks. Papastratis, 2021 provided a survey from recent methods, applications and frameworks.

Explainable artificial intelligence systems have been interested for other medical tasks. Deep learning models have shown significant results especially in medical imaging tasks. Recently, researchers have focused on explainable medical systems to assist medical experts and provide useful explanations so that any physician can understand the predictions of a system. In Brunese et al., 2020, the authors focused on coronavirus detection through x-ray images. They developed a deep convolutional model for extracting features from images to determine if the patient is healthy or affected by coronavirus. Then they use Grad-CAM (Xu et al., 2020; Selvaraju et al., 2017) to provide visual explanations and mark the areas of the x-ray that are affected by coronavirus.

ExplAIner framework (Spinner et al., 2019) helps physicians to understand machine and deep- learning models in medical classification and segmentation tasks. In addition, this framework consists tools for analyzing models by using explainable techniques, which can help to monitor the optimization process for building better models. The explAIner can provide interactive graph visualization of a model, performance metrics and integrate high-level explainable methods to interpret it.

4.3 Future Directions

Following the work in this thesis, there are some exciting possible directions of continued exploration. For instance, the collection of a larger, less redacted dataset to continue this research is currently underway at our hospital. Comparing the reported results to new ones on the new data will confirm our hypotheses and show more definitely how precisely age estimation and TSC anatomical abnormalities detection can become automatically obtained for children at risk of long-term neurological disorders.

Further, it would be helpful to have the opportunity to discuss the misclassified cases with the radiology personnel who collect the data and the observers who interpret it to better understand the failure modes of our models. Such work would help address reproducibility and validity of the models.

Explainable methods (XAI) are another interesting direction of work. With XAI applied to TSC structural brain pathology based on MRI sequences, a pediatrician might have rapid and reliable identification of TSC cases at high risk of epilepsy upon examination of the interpreted images. 3D visualization of important features in gradient-weighted Grad-CAM and saliency maps using deep convolutional networks to visualize the seizure onset zone from normal brain MRIs. These maps show where a CNN model focuses within an MRI image to place the image into the epilepsy category. Such methods may be coupled to the electrophysiological studies of epileptic regions of the brain.

Finally, establishing a precise 3D segmentation of childrens' brain tissues for segmentation of cortical tubers, subependymal nodules (SENs), and subependymal giant cell astrocytomas (SEGAs) would be helpful in TSC diagnosis. So-called "panoptic" segmentation (where each foreground voxel is assigned one class rather than detecting and segmenting one target structure only) can be an essential step towards comprehensive volumetric studies and especially, quantitative analysis of early brain development in TSC patients with epilepsy risk. 3D CNNs will likely be used for automatic TSC MR image segmentation. The project will provide methods to aid the clinicians that interpret structural and functional changes of infant brains that lead to elevated risk for future impairments in cognitive function and behavior(e.g., autism spectrum disorders). Continuing to develop advanced predictive and analytical models to better understand the child's brain for predicting neurological disease states will be a major effort in the coming decades. These AI methods for estimating the BDA and determining locations of tuberous structures are a sound place to begin that program of work.

Appendix A

NIMH Agreement

OMB Control Number: 0925-0667
Expiration Date: 11/30/2020

NIMH Data Archive

Data Use Certification

Last updated: August 2019

OMB Control Number: 0925-0667
Expiration Date: 11/30/2020

NIMH Data Archive Data Use Certification

Introduction

The National Institute of Mental Health (NIMH) Data Archive (NDA) is an NIH-funded collaborative resource that contains human subjects research data from multiple research data repositories.

The NIMH Data Archive Data Use Certification (DUC) is used to request access to shared research data in a data repository within the NIMH Data Archive. Shared data are available with either an Institutional sponsorship or an Individual sponsorship. All data access requests require acceptance of the Data Use Terms and Conditions contained in this DUC. (See the *NIMH Data Archive Recipient Information and Certifications* form in this document for available data and associated sponsorship types.)

- Institutional sponsorship requires Recipients to be affiliated with an NIH recognized institution (foreign or domestic), based upon registration in the NIH's eRA Commons system, with an active Federal Wide Assurance (FWA) issued by the Department of Health and Human Services, Office for Human Research Protections (OHRP). The signature of an Authorized Institutional Business Official is also required on this DUC.
- Individual sponsorship may be requested by a Recipient without the need for sponsorship by or affiliation with an NIH recognized institution and, therefore, the signature of an Authorized Institutional Business Official or an active institutional FWA is not required.

A Data Access Committee(s) (DAC) will objectively review a data access request sponsored by an Institution. Individual sponsorships do not require DAC review. To submit data to the NIMH Data Archive, the NIMH Data Archive Data Submission Agreement (DSA) must be completed, which is a separate document.

The NIMH Data Archive (NDA)

The National Institutes of Health (NIH) and NIMH have developed a data infrastructure to store the collection of data from participants in research studies, regardless of the source of funding. The extensive information collected by these studies is harmonized and subsequently stored in one of several data repositories within the NIMH Data Archive (NDA) data infrastructure, providing a rare and valuable scientific resource. A current list of all NDA data repositories and links to their websites is available at https://nda.nih.gov/about/about-us.html.

The NIH and NIMH seek to encourage the use of these resources to achieve rapid scientific progress. Moreover, NIMH has made data sharing a requirement for all clinical research it funds (see NOT-MH-19-033). In order to take full advantage of such resources and maximize their research value, it is important that data are made **broadly available**, on appropriate terms and conditions, to the largest possible number of qualified investigators in a timely manner.

Data collected by the Submitters have been stripped of all individual identifiers, but the unique and intrinsically personal nature of genomics data, brain imaging, and other derivative data of which are included in these repositories, combined with the recent increase in the accessibility of conducting genotype and other sequence analyses (in terms of technological capacity and cost), has altered the framework through which "identify-ability" can be defined. To protect and assure the confidentiality and privacy of all participants, the Recipient who is granted access to these data is expected to adhere to the specifications of this DUC. Failure to do so could result in denial of further access to data.

OMB Control Number: 0925-0667
Expiration Date: 11/30/2020

Data Use Terms and Conditions

I request access to shared data from the NIMH Data Archive for the purpose of scientific investigation, scholarship or teaching, or other forms of research and research development as described in the following NIMH Data Archive Data Use Certification (DUC). I, and any Other Recipients listed in this DUC, agree to the following terms:

1. Research Data Use Statement

Generally, these data will be used by the Recipient in connection with the purpose indicated and described in the *Research Data Use Statement* on the DUC. Recipients are encouraged to explore shared data in the NIMH Data Archive for a variety of purposes including secondary analysis, hypothesis generation, and replication regardless of whether said exploration leads to analysis in support of a question beyond the scope of the originally identified purpose described in the *Research Data Use Statement*.

2. Non-transferability of Agreement

This DUC is not transferable. If a Recipient changes institution and wishes to retain access to the NIMH Data Archive, a new DUC is required.

3. Non-Identification of Subjects

Recipients agree that data will not be used to establish the individual identities of any of the study participants from whom data were obtained (or their relatives) and/or contact the individual study participant, except as permitted by law (e.g., in connection with a separately negotiated collaboration with the original research team or the enrollment of the consented subject in the Recipient's study). Recipients agree to not publish or disseminate any derived data that could aid in the re-identification of any of the study participants (or their relatives). Recipients agree to notify the NIH at NDAHelp@mail.nih.gov as soon as possible if, upon use of NIMH Data Archive data, identifying information is discovered.

4. Use of the NIH Global Unique Identifier (GUID)

The Global Unique Identifier (GUID) is a computer-generated alphanumeric code that is unique to each research participant. The GUID allows the NIMH Data Archive to link together all submitted information on a single participant, giving researchers access to information even if the data were collected at different locations or through different studies. If Recipients access data on individuals for whom they, themselves, have previously submitted data to the NIMH Data Archive, Recipients may gain access to more data about an individual participant than they, themselves, collected. Consequently, these research activities may be considered "human subjects research" within the scope of 45 C.F.R. 46. Recipients must comply with the requirements contained in 45 C.F.R. 46, as applicable, which may require Institutional Review Board (IRB) approval of the Research Data Use Statement.

5. Data Disclaimers

Recipients acknowledge that the NIH does not and cannot warrant the results that may be obtained by using any data or data analysis tools included in the NIMH Data Archive. The NIH disclaims all warranties as to the accuracy of the data in the NIMH Data Archive or the performance or fitness of the data or data analysis tools for any particular purpose.

6. Data Access for Research

Data and Supporting Documentation in the NIMH Data Archive are eligible for access by qualified researchers, pursuant to the terms set forth in this DUC. Recipients acknowledge that other researchers have access to the data and that downloading, and duplication of research is a distinct possibility,

OMB Control Number: 0925-0667
Expiration Date: 11/30/2020

thereby decreasing subject data protections. Raw or nearly raw research data files (e.g. fastq, bam, MRI, and EEG recordings) are made available for just in time computation, regardless of where the computational resources may reside. Therefore, data copied shall not be persisted (i.e., stored) beyond the time necessary for computation and shall be expunged once computation has been completed. Recipients are encouraged to utilize the NIMH Data Archive computational capabilities described at https://nda.nih.gov/tools/nda-tools.html#cloud.

7. Supporting Documentation

Recipients agree to review the supporting information, materials, and documentation ("Supporting Documentation") for the data accessed in the NIMH Data Archive to enable efficient use of the submitted data by Recipients unfamiliar with the data or the research project. Examples of supporting documentation include:

- Research protocol(s)
- Questionnaire(s)
- Study manuals

8. Sharing of a NIMH Data Archive Study/Acknowledgements

Recipients agree to create and share an NIMH Data Archive Study (https://nda.nih.gov/get/manuscript-preparation.html) for each publication, computational pipeline, or other public disclosure of results from the analysis of data accessed in the NIMH Data Archive, whether reporting positive or negative results, thereby linking it to the underlying data. Recipients agree to create the NIMH Data Archive Study when a manuscript is submitted for review and share the Study when the publication is released. Recipients agree to acknowledge the appropriate NIMH Data Archive data repository and the relevant Digital Object Identifier(s) (DOI), which will be created by NIMH Data Archive staff, in any and all oral and written presentations, disclosures, and publications (including abstracts, as space allows) resulting from any and all analyses of data, whether or not the Recipient is collaborating with Submitter(s). The oral or written presentation, disclosure, or publication should include an acknowledgement statement, which includes a disclaimer of NIH endorsement, as appropriate. Acknowledgements specific to each NDA data repository are maintained at https://nda.nih.gov/get/manuscript-preparation.html.

If the Research Project involves collaboration with Submitters or NIH staff (as indicated in the DUC), then Recipient will acknowledge Submitters or NIH staff as co-authors, if appropriate, on any presentation, disclosure, or publication.

9. No Distribution of Data

Recipients agree to retain control over data from the NIMH Data Archive, and further agree not to transfer or sell data, with or without charge, in any form, to any other entity or any individual or to distribute the data to anyone other than the Other Recipients listed on this DUC who also agree to the terms in this DUC. This includes any data derived from the data in the NIMH Data Archive if the associated GUID is distributed with that derived data or if the derived data can aid in the re-identification of a research participant.

10. Non-Governmental Endorsement; Liability

Recipients agree not to claim, infer, or imply endorsement of the research project described in the *Research Data Use Statement*, the entity, or personnel conducting the research project or any resulting commercial product(s) by the United States Government, the Department of Health & Human Services, the National Institutes of Health, or the National Institute of Mental Health. The United States Government assumes no liability except to the extent provided under the Federal Tort Claims Act (28

OMB Control Number: 0925-0667
Expiration Date: 11/30/2020

U.S.C. § 2671-2680).

11. Recipient's Compliance with Institutional Requirements
Recipients with Institutional sponsorship acknowledge that access, if provided, is for research that is approved by the Institution with which they are affiliated, which must be operating under an active Federal Wide Assurance (FWA) issued by the Department of Health & Human Services, Office for Human Research Protections (OHRP). Furthermore, Recipients agree to comply with all applicable rules for the protection of human subjects, which may include Department of Health and Human Services regulations at 45 C.F.R. Part 46, and other federal and state laws for the use of this data. Recipients agree to report promptly to the NIH any unanticipated problems involving risks to subjects or others. This DUC is made in addition to, and does not supersede, any of Recipient's institutional policies or any local, State, and/or Federal laws and regulations that provide additional protections for human subjects.

12. Recipient's Permission to Post Information Publicly
Recipient agrees to permit the NIMH Data Archive to publicly summarize the Recipient's research use of data along with the Recipient's name and organizational/institutional affiliation.

13. Privacy Act Notification
Recipients agree that information collected by the NIH from a Recipient, as part of the DUC, may be made public in part or in whole for tracking and reporting purposes. This Privacy Act Notification is provided pursuant to Public Law 93-579, Privacy Act of 1974, 5 U.S.C. Section 552a. Authority for the collection of the information requested below from Recipients comes from the authorities regarding the establishment of the National Institutes of Health, its general authority to conduct and fund research and to provide training assistance, and its general authority to maintain records in connection with these and its other functions (42 U.S.C. 203, 241, 289l-1 and 44 U.S.C. 3101), and Sections 301 and 493 of the Public Health Service Act. These records will be maintained in accordance with the Privacy Act System of Record Notice 09-25-0156 (https://oma.od.nih.gov/forms/Privacy%20Documents/Documents/Privacy%20Act%20Systems%20of%20Records%20Notices%20(SORNs)%205-1-15.pdf) covering "Records of Participants in Programs and Respondents in Surveys Used to Evaluate Programs of the Public Health Service, HHS/PHS/NIH/OD." The primary uses of this information are to document, track, monitor, and evaluate the use of NIMH Data Archive datasets, as well as to notify interested Recipients of updates, corrections or other changes to the database.

The Federal Privacy Act protects the confidentiality of some NIH records. The NIH will use the information collected for the purposes described above. In addition, the Act allows the release of some information in the Recipient's records without the Recipient's permission; for example, if it is requested by members of Congress or other authorized individuals. The information requested in this DUC is voluntary, but necessary for obtaining access to data in the NIMH Data Archive.

14. Security
Recipients acknowledge that the data being made available were made available for researcher use with the expectation that the data will be protected in a manner consistent with security best practices. Such practices include, but are not limited to, the following:

- Accounts and passwords will not be shared.
- Data are protected from anonymous access. Any data transferred or stored outside of the NIMH Data Archive will be protected using standard encryption protocols and/or strong password protection.
- When finished using the data, the data will be expunged, as permitted by law.

OMB Control Number: 0925-0667
Expiration Date: 11/30/2020

15. Annual Update/Research Use Reporting
Recipients will provide to NDAHelp@mail.nih.gov an annual summary of research accomplishments from using the NIMH Data Archive and agree to create and share an NIMH Data Archive Study for each public disclosure of results pursuant to the Sharing of an NIMH Data Archive Study/Acknowledgements term in this DUC. The NIH encourages Recipients who publish manuscripts based on a combination of data from the NIMH Data Archive data, data derived from NDA data, and data collected independent of the NIMH Data Archive to consider submitting the complete analyzed dataset to the NIMH Data Archive.

16. Amendments
Amendments to this DUC must be in writing and signed by authorized representatives of all parties.

17. Termination
Either party may terminate this DUC, without cause, provided 30 days' advanced written notice to the other party. Recipients agree to immediately report violations of this agreement to the appropriate NIMH Data Archive Data Access Committee. Additionally, the NIH may terminate this agreement with 5 days' advanced written notice if the NIH determines, in its sole discretion, that a Recipient has committed a material breach of this DUC. The NIH may, in its sole discretion, provide a Recipient with 30 days' advanced written notice to remedy a breach before termination.

18. Term and Access Period
Recipients are granted permission to access requested and approved data from the NIMH Data Archive for a period of one year and this DUC will automatically terminate at that time. Data access may be renewed upon certification of a new DUC.

19. Accurate Representations
Recipients expressly certify that the contents of any statements made or reflected in this document are truthful and accurate.

OMB Control Number: 0925-0667
Expiration Date: 11/30/2020

Burden Disclosure Statement

Public reporting burden for this collection of information is estimated to vary from 15 min to 1.5 hours per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. **An agency may not conduct or sponsor, and a person is not required to respond to, a collection of information unless it displays a currently valid OMB control number.** Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to: NIH, Project Clearance Branch, 6705 Rockledge Drive, MSC 7974, Bethesda, MD 20892-7974, ATTN: PRA (0925-0667). Do not return the completed form to this address.

OMB Control Number: 0925-0667
Expiration Date: 11/30/2020

NIMH Data Archive Recipient Information and Certifications

1. Access Request:

Application Type		Data Requested	Recipient Sponsor*
NEW ☐	RENEWAL ☐	NIMH Data Archive (NDA)	Institutional
NEW ☐	RENEWAL ☑	Adolescent Brain Cognitive Development Study (ABCD)	Institutional
NEW ☐	RENEWAL ☑	Connectome Coordination Facility (CCF)	Institutional
NEW ☐	RENEWAL ☐	Osteoarthritis Initiative (OAI)	Individual
NEW ☐	RENEWAL ☐	NIAAA Data Archive ($NIAAA_{DA}$)	Institutional
NEW ☐	RENEWAL ☐		

**Institutional sponsorship requires the signature of an Authorized Institutional Business Official and an active Federal Wide Assurance (FWA) number in the Signatures section below. See the "Introduction" section on page 1 for more information.*
See https://nda.nih.gov/about/about-us.html for a current list of all NDA Data Repositories.

2. Lead Recipient:

First Name: Mahdieh Last Name: shabanian Degree:

Institution: Le Bonheur Children's Hospital
City: memphis State/Province: TN - Tennessee Country: UNITED STATES
Phone: 9012875387 E-mail Address: mshabani@uthsc.edu

3. Research Data Use Statement: Describe the purpose of the scientific investigation, scholarship or teaching, or other form of research and research development for which you are requesting access to the NIMH Data Archive.

I am conducting the research involving central nervous system infection in infants from birth to 5 years to predict any neurodevelopmental delay after birth immediately using deep learning algorithms as a Ph.D. student in UTHSC. I need the normal and healthy MRIs and DTIs for training the deep learning model to predict any CNS infections after birth. This research is necessary for the radiology department in Le Bonheur. BCP dataset would be a great dataset for this project from birth to 5 years. Thanks

0MB Control Number: 0925-0667
Expiration Date: 11/30/2020

4. Renewal Applicants Only:

Has a publication, computational pipeline, or other public disclosure of results from the analysis of data accessed in the NIMH Data Archive resulted from a Recipient's previous access period? ☐ Yes ☐ No

If Yes, has an NOA Study been created? ☐ Yes List the NOA Study number(s): _____
☐ No* List the PubMed ID(s) orcitation(s): _____
arXiv: 1910.1259 2019 Oct 27

Classification of Neurodevelopmental Age in Normal Infants Using 3D-CNN based on Brain MRI

*See 8. Sharing of a NIMH Data Archive Study/Acknowledgements above. Contact the NOA Help Desk (NDAHelp@mail.nih.gov) to create an NOA Study.

5. Other Recipient(s): List all individuals who will access, use, or analyze the data regardless of position title or data use role. Use additional sheets as needed.

First Name: Mahdieh Last Name: Shabanian Degree: Ph.D
Institution: University of Tennessee Health Science Center
City: Memphis State/Province: TN Country: USA
Phone: 9012875384 E-mail Address: mshabani@uthsc.edu

First Name: _____ Last Name: _____ Degree: _____
Institution: _____
City: _____ State/Province: _____ Country: _____
Phone: _____ E-mail Address: _____

First Name: _____ Last Name: _____ Degree: _____
Institution: _____
City: _____ State/Province: _____ Country: _____
Phone: _____ E-mail Address: _____

First Name: _____ Last Name: _____ Degree: _____
Institution: _____
City: _____ State/Province: _____ Country: _____
Phone: _____ E-mail Address: _____

First Name: _____ Last Name: _____ Degree: _____
Institution: _____
City: _____ State/Province: _____ Country: _____
Phone: _____ E-mail Address: _____

OMB Control Number: 0925-0667
Expiration Date: 11/30/2020

6. **Authorized Institutional Business Official**: Requests to access data requiring an Institutional sponsorship must list an individual with an "SO" role as defined in the NIH eRA Commons - *https://commons.era.nih.gov/commons*

Name: Sarah White Email Address: swhite82@uthsc.edu

7. **Signatures:** By signing and dating this DUC to request access to data in the NIMH Data Archive, I and my Institutional Official *(if required)* certify that we will abide by the Data Use Terms and Conditions defined in this DUC. I further acknowledge that I have shared this document with any Other Recipients who will participate in the use of data from the NIMH Data Archive. My Institutional Business Official *(if required)* also acknowledges that they have shared this document with appropriate institutional organizations.

[redacted] Signature — 3/10/20 Date

Authorized Institutional Business Official Signature *(if required)* — 3/10/2020 Date

Inquiries and requests to access data in the NIMH Data Archive should be sent to:
Office of Technology Development and Coordination (OTDC), Program Director
National Institute of Mental Health | National Institutes of Health
6001 Executive Boulevard, Room 8125, MSC 9640 Bethesda, MD 20892-9640
Telephone: 301-443-3265 | Email: NDAHelp@mail.nih.gov

August 2019

Appendix B

IRB Approval Letter

Institutional Review Board
910 Madison Avenue, Suite 600
Memphis, TN 38163
Tel: (901) 448-4824

May 21, 2021

John Joseph Bissler, M.D.
UTHSC - Peds-Nephrology
49 North Dunlap Street Room 323
Memphis, TN 38163-2242

Re: 21-08082-XM
Study Title: Deep Neural Network Model Detection of Anatomical Biomarkers for Epilepsy in Tuberous Sclerosis Complex (TSC) in pediatric patients based on MRI and CT

Dear Dr. Bissler:

The Administrative Section of the UTHSC Institutional Review Board (IRB) reviewed your application for revision of your previously approved project, referenced above.

The IRB determined that your revision application is eligible for expedited review under 45 CFR 46.110(b)(2) and that your study remains eligible for exempt status under 45 CFR 46.104(d) (4) (iii). The attached revisions to your project were approved as complying with proper consideration of the rights and welfare of human subjects. The use of children as subjects is approved under 45 CFR 46.404.

The revisions to this study may not be instituted until you receive approval from the institution(s) where the research is being conducted.

In the event that subjects are to be recruited using solicitation materials, such as brochures, posters, web-based advertisements, etc., these materials must receive prior approval of the IRB. Any revisions in the approved application must also be submitted to and approved by the IRB prior to implementation. In addition, you are responsible for reporting any unanticipated problems, including reportable adverse events, involving risks to subjects or others in the manner required by the local IRB policy. Lastly, you must request to close your project when you have completed data analysis by submitting a study closure form to the IRB.

Sincerely,

Signature applied by Lisa Robin Hagen on 05/21/2021 07:25:40 AM CDT

Lisa Hagen BA
IRB Administrative Research Specialist
UTHSC IRB

Attachment: Revisions

1. The study application was updated to Version 1.3 to incorporate:

   a) The addition of Harris Cohen as co-investigator
   b) The addition of Mohammed Abdeen as research support staff
   c) Updated KSP contact information and;
   d) Converted to the latest application format

2. In accord with 45 CFR 46.104(d)(4), informed consent remains not required

Please note that while the IRB is still processing IRB submissions during the COVID-19 pandemic, you must follow the UTHSC IRB's COVID 19 policy located on our website here: https://www.uthsc.edu/research/compliance/irb/covid-19.php You must review the policy and adhere to it as it relate to any and each of your UTHSC IRB studies.

Appendix C

MRI Classification_5 fold_ NIfTI (2D CNN & 3D CNN)

MRI Classification_5 fold_ NIfTI (2D CNN & 3D CNN)

Mahdieh Shabanian

October 1, 2021

[ ]:
```
# Under Review

# 2D and 3D CNN models were trained on a personal computer (NVIDIA TITAN RTX␣
 ↪GPU, Python 3.7.9, TensorFlow 2.1.0)
```

[ ]:
```
from __future__ import division, print_function, absolute_import

import math
import time
import sys
import cv2
import os
import random
import itertools
from pathlib import Path

import numpy as np
from PIL import Image
from os import listdir
from os.path import join, basename, isdir
from scikitplot.metrics import plot_confusion_matrix, plot_roc
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score,␣
 ↪classification_report
from sklearn.model_selection import KFold, StratifiedKFold, train_test_split
from sklearn.preprocessing import normalize
from sklearn.utils import shuffle
from skimage.color import gray2rgb
from tqdm.notebook import trange, tqdm, tqdm_notebook
import scipy
from scipy import ndimage
import h5py
import zipfile
import pandas as pd

import matplotlib.pyplot as plt
import matplotlib.image as im
import matplotlib.image as img
```

```
import nibabel as nb
from nibabel.testing import data_path
import SimpleITK as sitk

%matplotlib inline
```

[ ]:
```
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow import keras
from tensorflow.keras import optimizers
from tensorflow.keras.optimizers import schedules

# If you use a recent tensorflow, you HAVE TO use the tensorflow.keras␣
 ↪exclusively, or you can get weird results from version mismatches.

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model, save_model, Sequential  ␣
 ↪#to creat hirachical layers
from tensorflow.keras.layers import Dense, Activation, Dropout,SpatialDropout2D,␣
 ↪Flatten, Conv3D, Conv2D, MaxPool3D, MaxPool2D, GlobalAveragePooling3D,␣
 ↪GlobalAveragePooling2D, Flatten, Input, BatchNormalization, GRU,␣
 ↪Bidirectional,Reshape
from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard, Callback
from tensorflow.keras.losses import categorical_crossentropy, binary_crossentropy
from tensorflow.keras.optimizers import Adadelta, Adam, SGD
from tensorflow.keras.regularizers import l1, l2
from tensorflow.keras.utils import to_categorical
from matplotlib.pyplot import cm

# more info on callbakcs: https://keras.io/callbacks/ model saver is cool too.
```

[ ]:
```
# Confusion matrix
class_labels = ['Newborn', '12 Months', '24 Months', '36 Months']

def plot_confusion_matrix(cm,
                          target_names=class_labels,
                          title='Confusion matrix',
                          cmap=None,
                          normalize=True):
    """
    given a sklearn confusion matrix (cm), make a nice plot

    Arguments
    ---------
    cm:           confusion matrix from sklearn.metrics.confusion_matrix

    target_names: given classification classes such as [0, 1, 2]
```

```
                the class names, for example: ['high', 'medium', 'low']

   title:       the text to display at the top of the matrix

   cmap:        the gradient of the values displayed from matplotlib.pyplot.cm
                see http://matplotlib.org/examples/color/colormaps_reference.
↪html
                plt.get_cmap('jet') or plt.cm.Blues

   normalize:   If False, plot the raw numbers
                If True, plot the proportions

   Usage
   -----
   plot_confusion_matrix(cm           = cm,                  # confusion matrix␣
↪created by
                                                             # sklearn.metrics.
↪confusion_matrix
                         normalize    = True,                # show proportions
                         target_names = y_labels_vals,       # list of names of␣
↪the classes
                         title        = best_estimator_name) # title of graph

   Citiation
   ---------
   http://scikit-learn.org/stable/auto_examples/model_selection/
↪plot_confusion_matrix.html

   """
   import matplotlib.pyplot as plt
   import numpy as np
   import itertools

   accuracy = np.trace(cm) / float(np.sum(cm))
   misclass = 1 - accuracy

   if cmap is None:
       cmap = plt.get_cmap('Blues')

   plt.figure(figsize=(8, 6))
   plt.imshow(cm, interpolation='nearest', cmap=cmap)
   plt.title(title)
   plt.colorbar()

   if target_names is not None:
       #plt.colorbar()
       #plt.imshow(data, cmap=cmap, vmin=0, vmax=1)
```

```
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)

    cm_norm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis] # [mwen]

    thresh = cm.max() / 2 # [mwen]
    thresh_norm = cm_norm.max() / 1.5 # [mwen]
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.2f}".format(cm_norm[i, j]) + "\n{:,}".
↪format(cm[i, j]), # [mwen]
                     horizontalalignment="center", verticalalignment="center",␣
↪fontsize=12,
                     color="white" if cm_norm[i, j] > thresh_norm else "black")
        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")


    plt.tight_layout()
    plt.ylabel('True label', fontsize=12, color='darkblue') # fontweight='bold'
    plt.xlabel('Predicted label\nAccuracy={:0.2f}; Misclassification rate={:0.
↪2f}'.format(accuracy, misclass), fontsize=12, color='darkblue')
    plt.show()
```

[ ]:
```
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
  try:
    # Currently, memory growth needs to be the same across GPUs
    for gpu in gpus:
      tf.config.experimental.set_memory_growth(gpu, True)
    logical_gpus = tf.config.experimental.list_logical_devices('GPU')
    print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
  except RuntimeError as e:
    # Memory growth must be set before GPUs have been initialized
    print(e)
```

[ ]:
```
# Global definitions
DO_3D = False

if DO_3D:
    INPUT_SHAPE = (150,150,20,3)
else:
    INPUT_SHAPE = (150,150,3)
```

```
NUM_CLASSES = -1 # will be determined from the number of folders later.
```

[ ]:
```
DATA_DIR = "F:/Python/Dataset/NIMH/Fusion/dev"  # Directory of dataset
```

[ ]:
```
os.listdir(DATA_DIR)
```

[ ]:
```
# These are needed to decide based on the file path if an image belongs to train␣
 ↪or valid
TRAIN_DIR_NAME = 'train_part'
VALID_DIR_NAME = 'valid_part'
```

0.0.1 Create a dataframe to hold the training/validation data information.

[ ]:
```
source_files = []
for current_file in Path(DATA_DIR).rglob('*.gz'):
    source_files.append(str(current_file).split('\\')[-3:])
source_files_df = pd.DataFrame(source_files)

print(source_files_df.head()) # Add this line to receive the output. It has to␣
 ↪have three columns, or the format has to be fixed.

#source_files_df.columns = ['a', 'b', 'c', 'd', 'set', 'class', 'filename']
source_files_df.columns = ['set', 'class', 'filename']
filename_parts = source_files_df['filename'].str.split('_', expand=True) # Note␣
 ↪that this is NOT the str.split() method, but a Pandas version! expand=True␣
 ↪yields dataframe columns rather than a list.
source_files_df['patient_id'] = filename_parts[1]
source_files_df['visit'] = filename_parts[2]
ending = filename_parts[3].str.split('.', expand=True)
source_files_df['contrast'] = ending[0]
source_files_df.head()

CATEGORIES = list(source_files_df['class'].unique())
NUM_CLASSES = len(CATEGORIES)

# Create dictionary of target classes
label_dict = {
 0: 'Month0',
 1: 'Month12',
 2: 'Month24',
 3: 'Month36',
}
```

[ ]:
```
FOLDS = 5 # How many folds to split into

patient_ids = source_files_df.patient_id.unique()
```

```
# For stratified folds, we need to figure out the class for each patient. We␣
 ↪assign it to the first, if imaged multiple times.
patient_classes = []
for patID in patient_ids:
    patient_classes.append(source_files_df.
 ↪loc[source_files_df['patient_id']==patID]['class'].unique()[0])
#print(list(zip(patient_ids, patient_classes)))

kf = StratifiedKFold(n_splits=FOLDS, shuffle=True, random_state=4711)

# Save the folds yielded by the generator method into an array, in order not to␣
 ↪need to wrap the entire training into the folds.
folds = []
for train_indices, val_indices in kf.split(patient_ids, patient_classes):
    folds.append([patient_ids[train_indices], patient_ids[val_indices]])
```

1 Create 2D and 3D dataset

[ ]:
```
pred = []
history = []

for TRAIN_ON_FOLD in range(FOLDS):

    if DO_3D:
        RESCALE_SHAPE = (250,250,30) # From this size, we will discard 50,50,5␣
 ↪from each border to yield the 150x150x20 size.
    else:
        RESCALE_SHAPE = (250,250,50) # Don't throw away so much information for␣
 ↪2D
    INTERPOL_ORDER = 2

    train_data = []
    train_labels = []
    valid_data = []
    valid_labels = []

    # loop over patients.
    for patient in tqdm_notebook(patient_ids[:], desc='Patient'): # First, go by␣
 ↪patients.
        images = source_files_df.loc[source_files_df['patient_id']==patient]
        # print(patient, '\n', images)
        classes = images['class'].unique()

        # loop over classes this patient is in.
```

```
        for the_class in classes: # Then by class (age group)
            class_images = images.loc[images['class']==the_class]
            #print(the_class, '\n', class_images)
            visits = class_images['visit'].unique()

            # Make sure there aren't multiple visits.
            for visit in visits: # Then by visits...
                visit_images = class_images.loc[class_images['visit']==visit]
                #print(visit, '\n', class_images)

                # For the fusion data, we expect 3 contrasts (which will end up␣
↪in the channel dimension)
                if len(visit_images)!=3:
                    print('should always have three contrasts, but have %d for␣
↪patient %s, class %s' %(len(visit_images),patient,the_class))
                    continue;

                if DO_3D:
                    image_out_size = (3,150,150,20)
                else:
                    image_out_size = (3,150,150) # We will discard the border to␣
↪save space.

                multichannel_image = np.zeros(image_out_size)
                multichannel_image_flipped = np.zeros(image_out_size)
                multichannel_image_rotL = np.zeros(image_out_size)
                multichannel_image_rotR = np.zeros(image_out_size)
                multichannel_image_rand_rotL = np.zeros(image_out_size)
                multichannel_image_rand_rotR = np.zeros(image_out_size)

                # now read, preprocess, cut, average, and concatenate the images␣
↪of one visit
                for idx, (index,image) in enumerate(visit_images.iterrows()):
                    #print(os.path.
↪join(DATA_DIR,image['set'],image['class'],image['filename']))
                    nii_data = nb.load(os.path.
↪join(DATA_DIR,image['set'],image['class'],image['filename'])) # compose the␣
↪filename from the current dataframe entry
                    np_data = np.asarray(nii_data.dataobj)

                    # resize data using ndimage from scipy (https://docs.scipy.
↪org/doc/scipy/reference/ndimage.html)
                    original_shape = np_data.shape
                    scale = [(RESCALE_SHAPE[i] + 0.0)/original_shape[i] for i in␣
↪range(len(original_shape))]
```

```
                    out_data = ndimage.interpolation.zoom(np_data, scale, mode="nearest", order = INTERPOL_ORDER)
                    out_data_mean = np.mean(out_data[50:200,50:200,int(out_data.shape[2]/2-2):int(out_data.shape[2]/2+2)], axis=2) # Select some slices in the middle and average.
                    rand_rotL = np.random.randint(10,90)
                    rand_rotR = np.random.randint(10,90)
                    if DO_3D:
                        multichannel_image[idx] = cv2.normalize(out_data[50:200,50:200,5:25], None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
                        if patient in folds[TRAIN_ON_FOLD][0]: # Only augment training data
                            multichannel_image_flipped[idx] = cv2.normalize(out_data[50:200,50:200,5:25], None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
                            multichannel_image_rotL[idx] = cv2.normalize(scipy.ndimage.rotate(out_data[50:200,50:200,5:25],-10, reshape=False), None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
                            multichannel_image_rotR[idx] = cv2.normalize(scipy.ndimage.rotate(out_data[50:200,50:200,5:25], 10, reshape=False), None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
                            multichannel_image_rand_rotL[idx] = cv2.normalize(scipy.ndimage.rotate(out_data[50:200,50:200,5:25],-rand_rotL, reshape=False), None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
                            multichannel_image_rand_rotR[idx] = cv2.normalize(scipy.ndimage.rotate(out_data[50:200,50:200,5:25], rand_rotR, reshape=False), None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
                    else:
                        multichannel_image[idx] = cv2.normalize(out_data_mean, None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
                        if patient in folds[TRAIN_ON_FOLD][0]: # Only augment training data
                            multichannel_image_flipped[idx] = cv2.normalize(out_data_mean[::-1], None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
                            multichannel_image_rotL[idx] = cv2.normalize(scipy.ndimage.rotate(out_data_mean, -10, reshape=False), None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
                            multichannel_image_rotR[idx] = cv2.normalize(scipy.ndimage.rotate(out_data_mean, 10, reshape=False), None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
```

```
                            multichannel_image_rand_rotL[idx] = cv2.
↪normalize(scipy.ndimage.rotate(out_data_mean, -rand_rotL, reshape=False),␣
↪None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
                            multichannel_image_rand_rotR[idx] = cv2.
↪normalize(scipy.ndimage.rotate(out_data_mean, rand_rotR, reshape=False), None,␣
↪alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)

                # Assign to either train or valid data
                if patient in folds[TRAIN_ON_FOLD][0]:
                    train_data.append([np.moveaxis(multichannel_image,0,3 if␣
↪DO_3D else 2), CATEGORIES.index(the_class)])  # add this to our train_data    ␣
↪
                    train_data.append([np.
↪moveaxis(multichannel_image_flipped,0,3 if DO_3D else 2), CATEGORIES.
↪index(the_class)])
                    train_data.append([np.moveaxis(multichannel_image_rotL,0,3␣
↪if DO_3D else 2), CATEGORIES.index(the_class)])
                    train_data.append([np.moveaxis(multichannel_image_rotR,0,3␣
↪if DO_3D else 2), CATEGORIES.index(the_class)])
                    train_data.append([np.
↪moveaxis(multichannel_image_rand_rotL,0,3 if DO_3D else 2), CATEGORIES.
↪index(the_class)])
                    train_data.append([np.
↪moveaxis(multichannel_image_rand_rotR,0,3 if DO_3D else 2), CATEGORIES.
↪index(the_class)])
                else:
                    valid_data.append([np.moveaxis(multichannel_image,0,3 if␣
↪DO_3D else 2), CATEGORIES.index(the_class)])



    # Create traain and valid
    X_train = []
    y_train = []

    for features,label in train_data:
        X_train.append(features)
        y_train.append(label)

    if DO_3D:
        X_train = np.array(X_train).reshape(-1, INPUT_SHAPE_3D[0],␣
↪INPUT_SHAPE_3D[1], INPUT_SHAPE_3D[2], INPUT_SHAPE_3D[3])
    else:
        X_train = np.array(X_train).reshape(-1, INPUT_SHAPE[0], INPUT_SHAPE[1],␣
↪INPUT_SHAPE[2])
    y_train = to_categorical(y_train, num_classes=NUM_CLASSES)
```

```
    print ("x_train shape: ", X_train.shape)
    print ("y_train shape: ", y_train.shape)

    X_valid = []
    y_valid = []

    for features,label in valid_data:
        X_valid.append(features)
        y_valid.append(label)

    if DO_3D:
        X_valid = np.array(X_valid).reshape(-1, INPUT_SHAPE_3D[0],␣
↪INPUT_SHAPE_3D[1], INPUT_SHAPE_3D[2], INPUT_SHAPE_3D[3])
    else:
        X_valid = np.array(X_valid).reshape(-1, INPUT_SHAPE[0], INPUT_SHAPE[1],␣
↪INPUT_SHAPE[2])
    y_valid = to_categorical(y_valid, num_classes=NUM_CLASSES)
    print ("x_test shape: ", X_valid.shape)
    print ("y_test shape: ", y_valid.shape)


    np.save('xtrain-aug_%d.npy'%TRAIN_ON_FOLD, X_train)
    np.save('xvalid_%d.npy'%TRAIN_ON_FOLD, X_valid)
    np.save('ytrain-aug_%d.npy'%TRAIN_ON_FOLD, y_train)
    np.save('yvalid_%d.npy'%TRAIN_ON_FOLD, y_valid)

##############################################################################################
↪
    #CNN Model
    from keras.constraints import max_norm
    from tensorflow.keras.layers import Dense, Activation,␣
↪Dropout,SpatialDropout2D, SpatialDropout3D

    num_classes=4



    def Conv(filters=16, kernel_size=(3,3), activation='relu', input_shape=None):
        if input_shape:
            return Conv2D(filters=filters, kernel_size=kernel_size,
                          activation=activation, input_shape=input_shape)
        else:
            return Conv2D(filters=filters, kernel_size=kernel_size,
                          activation=activation)

    def Conv_3d(filters=16, kernel_size=(3,3,3), activation='relu',␣
↪input_shape=None):
```

```
        if input_shape:
            return Conv3D(filters=filters, kernel_size=kernel_size,
                          activation=activation, input_shape=input_shape)
        else:
            return Conv3D(filters=filters, kernel_size=kernel_size,
                          activation=activation)


    # Define 2D Model
    def CNN(input_dim, num_classes):
        model = Sequential()

        model.add(Conv(64, (3,3), input_shape=(150,150,3)))
        model.add(Conv(64, (3,3)))
        model.add(BatchNormalization())
        model.add(MaxPool2D(pool_size=(2,2)))
        model.add(Dropout(0.2))


        model.add(Conv(64, (3,3)))
        model.add(Conv(64, (3,3)))
        model.add(BatchNormalization())
        model.add(MaxPool2D(pool_size=(2,2)))
        model.add(Dropout(0.2))


        model.add(Conv(128, (3,3)))
        model.add(Conv(128, (3,3)))
        model.add(BatchNormalization())
        model.add(MaxPool2D(pool_size=(2,2)))
        model.add(Dropout(0.2))

        model.add(GlobalAveragePooling2D())
        #model.add(Flatten())

        model.add(Dense(128, activation='relu',  kernel_constraint=max_norm(2.)))
        model.add(Dropout(rate=0.7))
        model.add(Dense(128, activation='relu',  kernel_constraint=max_norm(2.)))
        model.add(Dropout(0.2))

        model.add(Dense(num_classes, activation='softmax'))

        return model

##########################################################################################
 ↪
    # Define 3D Model
```

```
    def CNN_3d(input_shape = (150,150,20,3), num_classes=4):
        model = Sequential()

        model.add(Conv_3d(32, input_shape=(150,150,20,3)))
        model.add(Conv_3d(32))
        model.add(BatchNormalization())
        model.add(MaxPool3D(pool_size=(2,2,2)))
        #model.add(MaxPool3D(pool_size=(2)))
        model.add(MaxPool3D())
        model.add(Dropout(0.2))

        model.add(Conv_3d(64))
        model.add(Conv_3d(64))
        model.add(BatchNormalization())
        model.add(MaxPool3D(pool_size=(2,2,2)))
        model.add(Dropout(0.2))

        model.add(Conv_3d(96))
        model.add(Conv_3d(96))
        model.add(BatchNormalization())
        model.add(MaxPool3D(pool_size=(2,2,2)))
        model.add(Dropout(0.2))

        model.add(Conv_3d(128))
        #model.add(BatchNormalization())
        #model.add(MaxPool3D())
        #model.add(Dropout(0.2))


        model.add(GlobalAveragePooling3D())
        #model.add(Flatten())

        model.add(Dense(128, activation='relu'))
        model.add(Dropout(0.7))
        model.add(Dense(128, activation='relu'))
        model.add(Dropout(0.2))


        model.add(Dense(num_classes, activation='softmax'))

        return model

##############################################################################################


##############################################################################################
    # 2D variants
```

```
    model = CNN(INPUT_SHAPE, NUM_CLASSES)

    # 3D variants
    #model = CNN_3d(INPUT_SHAPE_3D, NUM_CLASSES)

    trainable_count = np.sum([K.count_params(w) for w in model.
↪trainable_weights])
    non_trainable_count = np.sum([K.count_params(w) for w in model.
↪non_trainable_weights])
    print('Total params: {:,}'.format(trainable_count + non_trainable_count))
    print('Trainable params: {:,}'.format(trainable_count))
    print('Non-trainable params: {:,}'.format(non_trainable_count))

###############################################################################################
    optimizer = tf.keras.optimizers.Adam(lr=0.001)

    # Make loss more expensive for the tougher classes.

    earlystop_cb = tf.keras.callbacks.EarlyStopping(monitor='val_loss',␣
↪patience=40, restore_best_weights=True)
    model.compile(loss=categorical_crossentropy,  optimizer= optimizer ,␣
↪metrics=['acc'])

    history.append(model.fit(X_train, y_train, batch_size=8, epochs=300,␣
↪validation_data=(X_valid,y_valid), class_weight=class_weight,␣
↪callbacks=[earlystop_cb]))

###############################################################################################


    #Confusion Matrix and Classification Report

    pred.append(model.predict(X_valid, batch_size=8))
```

Apart from now postprocessing the predictions for the desired fold, nothing has to change...

[ ]:
```
# Confusion Matrix and Classification Report for individual folds
y_true_all = []
y_pred_all = []
for WHICH_FOLD in range(FOLDS):

    y_pred = np.argmax(pred[WHICH_FOLD], axis=1)
    y_pred_all.append(y_pred)
    # Load corresponding y_valid data
    y_true = np.load(str('yvalid_%d.npy'%WHICH_FOLD))

    y_true = np.argmax(y_true, axis=1)
```

```
    y_true_all.append(y_true)

    class_labels = ['Newborn', '12 Months','24 Months', '36 Months']
    print(classification_report(y_true, y_pred, target_names=class_labels))
    print(confusion_matrix(y_true, y_pred))
```

[ ]:
```
# Calculate summary statistics from all validation runs
flat_y_pred_all = [item for sublist in y_pred_all for item in sublist]
flat_y_true_all = [item for sublist in y_true_all for item in sublist]
cm = confusion_matrix(flat_y_true_all, flat_y_pred_all)
class_labels = ['Newborn', '12 Months','24 Months', '36 Months']
print("Summed Metrics:")
print(classification_report(flat_y_true_all, flat_y_pred_all,␣
 ↪target_names=class_labels))
print(cm)
```

[ ]:
```
plot_confusion_matrix(cm, normalize=True)
```

[ ]:
```
from pycm import *

cm = ConfusionMatrix(flat_y_true_all, flat_y_pred_all)
cm.table
print(cm)
```

[ ]:
```
#Plot Accuracy and los for 5fold
losses = []
val_losses   = []
accs = []
val_accs   = []
for hist in history:
    losses.append(hist.history['loss'])
    val_losses.append(hist.history['val_loss'])
    accs.append(hist.history['acc'])
    val_accs.append(hist.history['val_acc'])

shortest = min([len(l) for l in losses])
print ('Shortest training:', shortest)

nlosses = [l[:shortest] for l in losses]
nval_losses = [l[:shortest] for l in val_losses]
naccs = [l[:shortest] for l in accs]
nval_accs = [l[:shortest] for l in val_accs]

mean_l = np.mean(nlosses, axis=0)
mean_vl = np.mean(nval_losses, axis=0)
mean_a = np.mean(naccs, axis=0)
mean_va = np.mean(nval_accs, axis=0)
```

```
min_l = np.min(nlosses, axis=0)
min_vl = np.min(nval_losses, axis=0)
min_a = np.min(naccs, axis=0)
min_va = np.min(nval_accs, axis=0)
max_l = np.max(nlosses, axis=0)
max_vl = np.max(nval_losses, axis=0)
max_a = np.max(naccs, axis=0)
max_va = np.max(nval_accs, axis=0)

# Visualize the result
plt.figure(figsize=(12, 6))
plt.plot(mean_l, '-r', label="Train loss")
plt.fill_between(range(shortest), min_l, max_l,
                 color='lightcoral', alpha=0.4)
plt.plot(mean_vl, '-b', label="Val loss")
plt.fill_between(range(shortest), min_vl, max_vl,
                 color='cornflowerblue', alpha=0.4)
plt.legend()
plt.title("Train/Valid Loss")
plt.xlabel('Epochs')
plt.ylabel('Loss')


plt.figure(figsize=(12, 6))
plt.plot(mean_a, '-r', label="Train acc")
plt.fill_between(range(shortest), min_a, max_a,
                 color='lightcoral', alpha=0.4)
plt.plot(mean_va, '-b', label="Val acc")
plt.fill_between(range(shortest), min_va, max_va,
                 color='cornflowerblue', alpha=0.4)
plt.legend()
plt.title("Train/Valid Accuracy")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

[ ]:
```
# If you want to plot one specific fold
# plot Loss
WHICH_FOLD=1
plt.figure(figsize=(12, 6))
plt.plot(history[WHICH_FOLD].history['loss'], label="Train loss")
plt.plot(history[WHICH_FOLD].history['val_loss'], label="Val loss")
plt.legend()
plt.title("Train/Valid Loss (fold %d)"%WHICH_FOLD)
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```

[ ]:
```
# plot accuracy for one specific fold
WHICH_FOLD=1
plt.figure(figsize=(12, 6))
plt.plot(history[WHICH_FOLD].history['acc'], label="Train acc")
plt.plot(history[WHICH_FOLD].history['val_acc'], label="Val acc")
plt.legend()
plt.title("Train/Valid Acc (fold %d)"%WHICH_FOLD)
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.show()
```

[ ]:
```
# Statistical results
from pycm import *

cm = ConfusionMatrix(flat_y_true_all, flat_y_pred_all)
cm.table
print(cm)
```

Appendix D

Effective_MRI Sequences_ Selection

Effective_MRI_sequences_Keras

December 5, 2021

1 Selecting Effective MRI sequences

[ ]:
```
 import math
import time
import sys
import cv2
import os
import h5py
import zipfile
import pandas as pd
import random
import itertools
import scipy
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as img
import nibabel as nb
from nibabel.testing import data_path
import SimpleITK as sitk
from PIL import Image
from os import listdir
from os.path import join, basename, isdir
from scikitplot.metrics import plot_confusion_matrix, plot_roc
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score
from sklearn.model_selection import KFold, StratifiedKFold, train_test_split
from sklearn.preprocessing import normalize
from sklearn.utils import shuffle
from tqdm.notebook import trange, tqdm, tqdm_notebook
from pathlib import Path
from scipy import ndimage
%matplotlib inline
```

[ ]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import optimizers
from tensorflow.keras.optimizers import schedules
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
from tensorflow.keras.models import Model, load_model, save_model, Sequential  ␣
 ↪#to creat hirachical layers
from tensorflow.keras.layers import Dense, Activation, Dropout,SpatialDropout2D,␣
 ↪Flatten, Conv3D, Conv2D, MaxPool3D, MaxPool2D, GlobalAveragePooling3D,␣
 ↪GlobalAveragePooling2D, Flatten, Input, BatchNormalization, GRU,␣
 ↪Bidirectional,Reshape
from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard, Callback
from tensorflow.keras.losses import categorical_crossentropy, binary_crossentropy
from tensorflow.keras.optimizers import Adadelta, Adam, SGD
from tensorflow.keras.regularizers import l1, l2
from tensorflow.keras.utils import to_categorical
from matplotlib.pyplot import cm
```

2 Isotropic function

[ ]:
```
#https://www.programcreek.com/python/example/123390/SimpleITK.ResampleImageFilter
# Apply resample (To the all dims same as isotropic)

def resample_img(itk_image, out_spacing=[2.0, 2.0, 2.0]):

    # Resample images to 2mm spacing with SimpleITK
    original_spacing = itk_image.GetSpacing()
    original_size = itk_image.GetSize()

    out_size = [
        int(np.round(original_size[0] * (original_spacing[0] / out_spacing[0]))),
        int(np.round(original_size[1] * (original_spacing[1] / out_spacing[1]))),
        int(np.round(original_size[2] * (original_spacing[2] / out_spacing[2])))]

    resample = sitk.ResampleImageFilter()
    resample.SetOutputSpacing(out_spacing)
    resample.SetSize(out_size)
    resample.SetOutputDirection(itk_image.GetDirection())
    resample.SetOutputOrigin(itk_image.GetOrigin())
    resample.SetTransform(sitk.Transform())
    resample.SetDefaultPixelValue(itk_image.GetPixelIDValue())
    resample.SetInterpolator(sitk.sitkNearestNeighbor)


    return resample.Execute(itk_image)
```

[ ]:
```
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
  try:
    # Currently, memory growth needs to be the same across GPUs
    for gpu in gpus:
```

```
      tf.config.experimental.set_memory_growth(gpu, True)
    logical_gpus = tf.config.experimental.list_logical_devices('GPU')
    print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
  except RuntimeError as e:
    # Memory growth must be set before GPUs have been initialized
    print(e)
```

[ ]:
```
# Dataset

DATA_DIR = "E:"
TRAIN_DIR_NAME = 'E:/Train'
```

[ ]:
```
os.listdir(TRAIN_DIR_NAME)
```

3 Create a data frame to hold the training/validation data information.

[ ]:
```
# This is not entirely robust! The paths below DATA_DIR need to be like mine␣
 ↪above.
#******************Train dataset*****************

train_source_files = []
#print("PATH:", DATA_DIR)
for current_file in Path(TRAIN_DIR_NAME).rglob('*.gz'):
    #print(current_file)
    train_source_files.append(str(current_file).split('\\')[-4:])
    #print(source_files)
train_source_files_df = pd.DataFrame(train_source_files)
#source_files_df.columns = ['a', 'b', 'c', 'd', 'set', 'class', 'filename']
train_source_files_df.columns = ['set','class', 'MRID', 'Sequences']

print(train_source_files_df.head())

#****************Validation dataset**************
val_source_files = []
#print("PATH:", DATA_DIR)
for current_file in Path(VALID_DIR_NAME).rglob('*.gz'):
    #print(current_file)
    val_source_files.append(str(current_file).split('\\')[-4:])
    #print(source_files)
val_source_files_df = pd.DataFrame(val_source_files)
#source_files_df.columns = ['a', 'b', 'c', 'd', 'set', 'class', 'filename']
val_source_files_df.columns = ['set','class', 'MRID', 'Sequences']

print(val_source_files_df.head())
```

3.1 Counting the number of MRIs

```
[ ]: print("TSC :",len(train_source_files_df.loc[(train_source_files_df['class'] ==␣
      ↪'TSC')]["MRID"].unique()),
     "Normal :",len(train_source_files_df.loc[(train_source_files_df['class'] ==␣
      ↪'Normal')]["MRID"].unique()))
```

```
[ ]: print("TSC :",len(val_source_files_df.loc[(val_source_files_df['class'] ==␣
      ↪'TSC')]["MRID"].unique()),
     "Normal :",len(val_source_files_df.loc[(val_source_files_df['class'] ==␣
      ↪'Normal')]["MRID"].unique()))
```

```
[ ]: def normalize(volume):
         """Normalize the volume"""
         min = -1000
         max = 400
         volume[volume < min] = min
         volume[volume > max] = max
         volume = (volume - min) / (max - min)
         volume = volume.astype("float32")
         return volume
```

```
[ ]: interpolation_order = 2
     out_shape=(100,200,200)
     out_spacing=[1.0, 1.0, 1.0]
     mode = 'nearest'
```

```
[ ]: # what kind of MRI sequesces we want!

     effective_sequences_TSC = train_source_files_df.
      ↪loc[(train_source_files_df['Sequences'] == '.nii.gz')]
     effective_sequences_Normal = train_source_files_df.
      ↪loc[(train_source_files_df['Sequences'] == '.nii.gz')]


     import os
     from tqdm import tqdm
     #  read all train seqs
     Train_seqs=[]
     Train_labels=[]

     classes={"TSC":1, "Normal":0}

     patient_id=np.array(effective_sequences_TSC["MRID"])
     seq_name=np.array(effective_sequences_TSC["Sequences"])

     for i in tqdm(range(len(effective_sequences_TSC["Sequences"]))):
```

```
    sequence=sitk.ReadImage(os.path.join(r"E:
 ↪\Train\TSC",patient_id[i],seq_name[i]))


################################### isotropic ###################
    # isotropic
    resampled_sitk_img = resample_img(sequence, out_spacing)

#################################################################
    read_sitk_t2=sitk.GetArrayFromImage(resampled_sitk_img)
    original_shape =read_sitk_t2.shape
    assert(len(original_shape) == len(out_shape))

    scale = [(out_shape[i] + 0.0)/original_shape[i] for i in␣
 ↪range(len(original_shape))]

    read_sitk_t2 = ndimage.interpolation.zoom(read_sitk_t2, scale, mode = mode,␣
 ↪order = interpolation_order)
    #normalize
    read_sitk_t2=normalize(read_sitk_t2)
    Train_seqs.append(read_sitk_t2)
    Train_labels.append(classes["TSC"])

############################### Normal ########################

patient_id=np.array(effective_sequences_Normal["MRID"])
seq_name=np.array(effective_sequences_Normal["Sequences"])


for i in tqdm(range(len(effective_sequences_Normal["Sequences"]))):
    sequence=sitk.ReadImage(os.path.join(r"E:
 ↪\Train\Normal",patient_id[i],seq_name[i]))


################################### isotropic ###################

resampled_sitk_img = resample_img(sequence, out_spacing)

#################################################################

    read_sitk_t2=sitk.GetArrayFromImage(resampled_sitk_img)
    original_shape =read_sitk_t2.shape
    assert(len(original_shape) == len(out_shape))

    scale = [(out_shape[i] + 0.0)/original_shape[i] for i in␣
 ↪range(len(original_shape))]
```

```
    read_sitk_t2 =ndimage.interpolation.zoom(read_sitk_t2, scale, mode = mode,␣
 ↪order = interpolation_order)

    #normalize
    read_sitk_t2=normalize(read_sitk_t2)

    Train_seqs.append(read_sitk_t2)
    Train_labels.append(classes["Normal"])

Train_seqs=np.array(Train_seqs)
Train_labels=np.array(Train_labels)
```

3.2 Counting the number of MRI sequesnces in VALID

[ ]:
```
# Checking the number of MRI sequesnces in valid

effective_sequences_TSC = val_source_files_df.
 ↪loc[(val_source_files_df['Sequences'] == '.nii.gz')]
effective_sequences_Normal = val_source_files_df.
 ↪loc[(val_source_files_df['Sequences'] == '.nii.gz')]
```

[ ]:
```
print("TSC",val_source_files_df.loc[(val_source_files_df['Sequences'] == 'AX T2␣
 ↪FLAIR.nii.gz')].nunique())
```

[ ]:
```
print("Normal",val_source_files_df.loc[(val_source_files_df['Sequences'] == 'MR␣
 ↪AX T2 FLAIR.nii.gz')].nunique())
```

[ ]:
```
# Checking the number of MRI sequesnces


effective_sequences_TSC = val_source_files_df.
 ↪loc[(val_source_files_df['Sequences'] == '.nii.gz')]
effective_sequences_Normal = val_source_files_df.
 ↪loc[(val_source_files_df['Sequences'] == '.nii.gz')]


############  read all Valid seqs ##############
Valid_seqs=[]
Valid_labels=[]

############################TSC###################


patient_id=np.array(effective_sequences_TSC["MRID"])
seq_name=np.array(effective_sequences_TSC["Sequences"])

for i in tqdm(range(len(effective_sequences_TSC["Sequences"]))):
```

```
    sequence=sitk.ReadImage(os.path.join(r"E:
 ↪\Valid\TSC",patient_id[i],seq_name[i]))


###################### isotropic ###############
    # isotropic
    print('Old shape: ', sitk.GetArrayFromImage(sequence).shape)
    plt.imshow(sitk.GetArrayFromImage(sequence)[5,:,:])
    plt.title("before isotropic")

    plt.show()
    print ("spacing before isotropic " ,sequence.GetSpacing())
    resampled_sitk_img = resample_img(sequence, out_spacing)
    print("Spacing after isotropic",resampled_sitk_img.GetSpacing())

################################################
read_sitk_t2=sitk.GetArrayFromImage(resampled_sitk_img)
    plt.imshow(read_sitk_t2[5,:,:])
    plt.title("after isotropic")

    plt.show()
    print('New shape for isotropic: ', read_sitk_t2.shape)
    print('Old shape for resize: ', read_sitk_t2.shape)
    original_shape =read_sitk_t2.shape
    assert(len(original_shape) == len(out_shape))

    scale = [(out_shape[i] + 0.0)/original_shape[i] for i in␣
 ↪range(len(original_shape))]

    read_sitk_t2 = ndimage.interpolation.zoom(read_sitk_t2, scale, mode = mode,␣
 ↪order = interpolation_order)
    print('New shape: ', read_sitk_t2.shape)

    plt.imshow(read_sitk_t2[5,:,:])
    plt.title("after resize")

    plt.show()
    # Normalize
    read_sitk_t2=normalize(read_sitk_t2)

    Valid_seqs.append(read_sitk_t2)
    Valid_labels.append(classes["TSC"])


################### Normal ######################
patient_id=np.array(effective_sequences_Normal["MRID"])
seq_name=np.array(effective_sequences_Normal["Sequences"])
```

```
for i in tqdm(range(len(effective_sequences_Normal["Sequences"]))):

    sequence=sitk.ReadImage(os.path.join(r"E:\Madi\TSC_project\80-20␣
 ↪class\Valid\Normal",patient_id[i],seq_name[i]))


#################### Isotropic ##################
    # Isotropic
    print('Old shape: ', sitk.GetArrayFromImage(sequence).shape)

    print ("spacing before isotropic " ,sequence.GetSpacing())
    resampled_sitk_img = resample_img(sequence, out_spacing)
    print("Spacing after isotropic",resampled_sitk_img.GetSpacing())


#################################################
    read_sitk_t2=sitk.GetArrayFromImage(resampled_sitk_img)
    print('New shape for isotropic : ', read_sitk_t2.shape)
    print('old shape for resize: ', read_sitk_t2.shape)

    original_shape =read_sitk_t2.shape
    assert(len(original_shape) == len(out_shape))

    scale = [(out_shape[i] + 0.0)/original_shape[i] for i in␣
 ↪range(len(original_shape))]

    # Resize
    read_sitk_t2 =ndimage.interpolation.zoom(read_sitk_t2, scale, mode = mode,␣
 ↪order = interpolation_order)
    #read_sitk_t2 = read_sitk_t2.crop((0, top, right, bottom))  #(left, top,␣
 ↪right, bottom)
    print('New shape: ', read_sitk_t2.shape)

    # Normalize
    read_sitk_t2=normalize(read_sitk_t2)
    Valid_seqs.append(read_sitk_t2)
    Valid_labels.append(classes["Normal"])

Valid_seqs=np.array(Valid_seqs)
Valid_labels=np.array(Valid_labels)
```

[ ]:
```
print("Valid seqs shape",Valid_seqs.shape,"\nValid label shape", Valid_labels.
 ↪shape)
```

4 Making Isotropic

5 Prepare train and valid dataset

[ ]:
```
#convert the label To categorical
NUM_CLASSES=2
Train_labels = to_categorical(Train_labels, num_classes=NUM_CLASSES)

# print("the label before convert",Train_labels[-1])
print("the label after convert",Train_labels[-1])

Valid_labels = to_categorical(Valid_labels, num_classes=NUM_CLASSES)
Valid_labels.shape
```

[ ]:
```
Train_seqs=Train_seqs.reshape(Train_seqs.shape[0],200,200,100)
Valid_seqs=Valid_seqs.reshape(Valid_seqs.shape[0],200,200,100)

Train_seqs=np.expand_dims(Train_seqs,axis=-1)
Valid_seqs=np.expand_dims(Valid_seqs,axis=-1)

Train_seqs.shape ,Valid_seqs.shape
```

6 CNN Models

[ ]:
```
#CNN Model
from keras.constraints import max_norm
from tensorflow.keras.layers import Dense, Activation, Dropout,SpatialDropout2D,␣
 ↪SpatialDropout3D
NUM_CLASSES=2

INPUT_SHAPE=(200,200,100,1)


def Conv_3d(filters=16, kernel_size=(3,3,3), activation='relu', padding= 'same',␣
 ↪kernel_regularizer=l2(0.001), kernel_constraint=max_norm(2.),␣
 ↪input_shape=None):
    if input_shape:
        return Conv3D(filters=filters, kernel_size=kernel_size,
                      activation=activation, padding=padding,␣
 ↪kernel_regularizer=kernel_regularizer,input_shape=input_shape,␣
 ↪kernel_constraint=kernel_constraint)
    else:
        return Conv3D(filters=filters, kernel_size=kernel_size,
                      activation=activation)

######################################################################
```

```
# Define 3D Model
def CNN_3d(input_shape = INPUT_SHAPE, num_classes=NUM_CLASSES):
    model = Sequential()

    #model.add(Conv_3d(32, input_shape=(1,150,150,20,1)))
    model.add(Conv_3d(32,(3,3,3) , padding= 'same', activation = 'relu' ,
                      kernel_regularizer=l2(0.001) , input_shape = INPUT_SHAPE))␣
 ↪#kernel_regularizer=l2(0.001)
    model.add(Conv_3d(32))
    model.add(BatchNormalization())
    model.add(MaxPool3D(pool_size=(2,2,2)))
    model.add(Dropout(0.15))

    model.add(Conv_3d(64))
    model.add(BatchNormalization())
    model.add(MaxPool3D(pool_size=(2,2,2)))
    model.add(Dropout(0.15))

    model.add(Conv_3d(96))
    model.add(BatchNormalization())
    model.add(MaxPool3D())
    model.add(MaxPool3D(pool_size=(2,2,2)))


    model.add(Conv_3d(128))
    model.add(BatchNormalization())
    model.add(MaxPool3D(pool_size=(2,2,2)))
    model.add(Dropout(0.15))

    model.add(GlobalAveragePooling3D())
    #model.add(Flatten())


    model.add(Dense(1024, activation='relu', kernel_constraint=max_norm(2.)))
    model.add(BatchNormalization())
    model.add(Dropout(0.2))

    model.add(Dense(512, activation='relu', kernel_constraint=max_norm(2.)))
    model.add(BatchNormalization())
    model.add(Dropout(0.25))

    model.add(Dense(512, activation='relu', kernel_constraint=max_norm(2.)))
    model.add(BatchNormalization())
    model.add(Dropout(0.15))


    model.add(Dense(num_classes, activation='softmax'))
```

```
    return model
```

[ ]:
```
# 3D variants
model = CNN_3d(INPUT_SHAPE, NUM_CLASSES)
```

[ ]:
```
model.summary()
```

[ ]:
```
# Define class weights for imbalacned data
from sklearn.utils import class_weight
class_weights = class_weight.compute_class_weight('balanced', np.unique(np.
 ↪argmax(Train_labels, axis=1)), np.argmax(Train_labels, axis=1))
# class_weights = class_weight.compute_class_weight('balanced', np.
 ↪unique(Train_labels), Train_labels)

print("Class weights:", class_weights)
```

[ ]:
```
optimizer = keras.optimizers.SGD(learning_rate=0.001 , momentum=0.96)

model.compile(loss="binary_crossentropy",  optimizer= optimizer ,␣
 ↪metrics=['acc']) #
```

[ ]:
```
history = model.fit(Train_seqs, Train_labels, batch_size=2, epochs=100,␣
 ↪validation_data=(Valid_seqs,Valid_labels))
```

[ ]:
```
model.save("TSC.h5")
```

[ ]:
```
from keras.models import load_model
model = load_model('TSC.h5')
model.summary()
```

[ ]:
```
model.evaluate(Train_seqs, Train_labels, batch_size=2)
```

[ ]:
```
model.evaluate(Valid_seqs, Valid_labels, batch_size=4)
```

[ ]:
```
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label="Training loss")
plt.plot(history.history['val_loss'], label="Val loss")
plt.legend()
plt.title("Training vs Validation Loss")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```

[ ]:
```
plt.figure(figsize=(12, 6))
plt.plot(history.history['acc'], label="Training acc")
plt.plot(history.history['val_acc'], label="Test acc")
plt.legend()
plt.title("Training vs Validation Accuracy")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.show()
```

[ ]:
```
# Confusion matrix
from sklearn.metrics import confusion_matrix, classification_report

#Confusion Matrix and Classification Report
pred = model.predict(Valid_seqs, batch_size=2)
pred = np.argmax(pred, axis=1)
y = np.argmax(Valid_labels, axis=1)

#pred=model.predict_classes(X_valid)
cm= confusion_matrix(y, pred)

class_labels = ['Normal', 'TSC']
print(classification_report(y, pred, target_names=class_labels))

print(confusion_matrix(y, pred))
```

[ ]:
```
def plot_confusion_matrix(cm,
                          target_names=class_labels,
                          title='T2w Flair_Confusion matrix',
                          cmap=None,
                          normalize=True):
    """
    given a sklearn confusion matrix (cm), make a nice plot

    Arguments
    ---------
    cm:           confusion matrix from sklearn.metrics.confusion_matrix

    target_names: given classification classes such as [0, 1, 2]
                  the class names, for example: ['high', 'medium', 'low']

    title:        the text to display at the top of the matrix

    cmap:         the gradient of the values displayed from matplotlib.pyplot.cm
                  see http://matplotlib.org/examples/color/colormaps_reference.
 ↪html
                  plt.get_cmap('jet') or plt.cm.Blues
```

```
    normalize:    If False, plot the raw numbers
                  If True, plot the proportions

    Usage
    -----
    plot_confusion_matrix(cm           = cm,                  # confusion matrix␣
↪created by
                                                              # sklearn.metrics.
↪confusion_matrix
                          normalize    = True,                # show proportions
                          target_names = y_labels_vals,       # list of names of␣
↪the classes
                          title        = best_estimator_name) # title of graph

    Citiation
    ---------
    http://scikit-learn.org/stable/auto_examples/model_selection/
↪plot_confusion_matrix.html

    """
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools

    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy

    if cmap is None:
        cmap = plt.get_cmap('Blues')

    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

    if target_names is not None:
        #plt.colorbar()
        #plt.imshow(data, cmap=cmap, vmin=0, vmax=1)
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]


    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
```

```
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.2f}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")


    plt.tight_layout()
    plt.ylabel('True label', fontsize=12, color='darkblue') # fontweight='bold'
    plt.xlabel('Predicted label\naccuracy={:0.2f}; misclass={:0.2f}'.
 ↪format(accuracy, misclass), fontsize=12, color='darkblue')
    plt.show()
```

[ ]:
```
plot_confusion_matrix(cm, normalize=True)
```

[ ]:
```
# ROC curve

from sklearn.metrics import roc_curve, auc


nn_fpr_keras, nn_tpr_keras, nn_thresholds_keras = roc_curve(y, pred)
auc_keras = auc(nn_fpr_keras, nn_tpr_keras)
plt.plot(nn_fpr_keras, nn_tpr_keras, marker='.', label='3D CNN (auc = %0.3f)' %␣
 ↪auc_keras)

plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='best')
plt.show()
```

[ ]:
```
from sklearn.metrics import roc_auc_score, roc_curve, auc
print('Area under ROC curve : ', roc_auc_score(y, pred) *100 )
```

[ ]:
```
from pycm import *

cm = ConfusionMatrix(y, pred)
cm.table
print(cm)
```

List of References

Abadi, Martin et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Brunese, Luca et al. (Nov. 2020). "Explainable Deep Learning for Pulmonary Disease and Coronavirus COVID-19 Detection from X-rays". In: *Computer Methods and Programs in Biomedicine* 196, p. 105608. ISSN: 0169-2607. DOI: 10.1016/j.cmpb.2020.105608. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7831868/ (visited on 10/13/2021).

Capal, Jamie K et al. (May 2017). "Influence of Seizures on Early Development in Tuberous Sclerosis Complex". In: *Epilepsy & behavior : E&B* 70 (Pt A), pp. 245–252. ISSN: 1525-5050. DOI: 10.1016/j.yebeh.2017.02.007. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5497719/ (visited on 10/13/2021).

Chollet, François (2015). *keras*. https://github.com/fchollet/keras.

Cohen, Alexander L et al. (2021). "Tuber Locations Associated with Infantile Spasms Map to a Common Brain Network". In: *Annals of Neurology* 89.4, pp. 726–739. ISSN: 1531-8249 0364-5134. DOI: 10.1002/ana.26015. URL: https://pubmed.ncbi.nlm.nih.gov/33410532/.

Cole, James H et al. (2017). "Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker". In: *NeuroImage* 163, pp. 115–124. DOI: 10.1016/j.neuroimage.2017.07.059. URL: https://pubmed.ncbi.nlm.nih.gov/28765056/.

Cui, Sunan et al. (2020). "Introduction to machine and deep learning for medical physicists". In: *Medical physics* 47.5, e127–e147. DOI: 10.1002/mp.14140. URL: https://pubmed.ncbi.nlm.nih.gov/32418339/.

Curatolo, Paolo, Romina Moavero, and Petrus J de Vries (2015). "Neurological and neuropsychiatric aspects of tuberous sclerosis complex". In: *The Lancet Neurology* 14.7, pp. 733–745. DOI: 10.1016/S1474-4422(15)00069-1. URL: https://pubmed.ncbi.nlm.nih.gov/26067126/.

Franke, Katja et al. (2010). "Estimating the age of healthy subjects from T1-weighted MRI scans using kernel methods: exploring the influence of various parameters". In: *Neuroimage* 50.3, pp. 883–892. DOI: 10.1016/j.neuroimage.2010.01.005. URL: https://pubmed.ncbi.nlm.nih.gov/20070949/.

Gao, Xiaohong W, Rui Hui, and Zengmin Tian (2017). "Classification of CT brain images based on deep learning networks". In: *Computer methods and programs in biomedicine* 138, pp. 49–56. DOI: 10.1016/j.cmpb.2016.10.007. URL: https://pubmed.ncbi.nlm.nih.gov/27886714/.

Google Trends - Interest in Pytorch Verses Keras Over Five Years (2021). URL: https://trends.google.com/trends/explore?date=today%205-y&geo=US&q=%2Fg%2F11gd3905v1,Keras (visited on 10/17/2021).

Grinberg, A et al. (2019). "Volumetric MRI Study of the Brain in Fetuses with Intrauterine Cytomegalovirus Infection and Its Correlation to Neurodevelopmental Outcome". In: *American Journal of Neuroradiology* 40.2, pp. 353–358. DOI: 10.3174/ajnr.A5948. URL: https://pubmed.ncbi.nlm.nih.gov/30760462/.

Gupta, Ajay et al. (2020). "Epilepsy and neurodevelopmental comorbidities in tuberous sclerosis complex: a natural history study". In: *Pediatric neurology* 106, pp. 10–16. DOI: 10.1016/j.pediatrneurol.2019.12.016. URL: https://pubmed.ncbi.nlm.nih.gov/32139167/.

Henske, Elizabeth P et al. (2016). "Tuberous sclerosis complex". In: *Nature reviews Disease primers* 2.1, pp. 1–18. DOI: 10.1038/nrdp.2016.35. URL: https://pubmed.ncbi.nlm.nih.gov/27226234/.

Hong, Jin et al. (2020). "Brain Age Prediction of Children Using Routine Brain MR Images via Deep Learning". In: *Frontiers in Neurology* 11. DOI: 10.3389/fneur.2020.584682. URL: https://pubmed.ncbi.nlm.nih.gov/33193046/.

Hosseini, SM Hadi, Hao Chen, and Monica M Jablonski (2020). "Automatic detection and counting of retina cell nuclei using deep learning". In: *Medical Imaging 2020: Biomedical Applications in Molecular, Structural, and Functional Imaging*. Vol. 11317. International Society for Optics and Photonics, p. 113172I. DOI: 10.1117/12.2567454. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11317/113172I/Automatic-detection-and-counting-of-retina-cell-nuclei-using-deep/10.1117/12.2567454.short?SSO=1.

Hulshof, Hanna M, Barbora Benova, et al. (2021). "The epileptogenic zone in children with tuberous sclerosis complex is characterized by prominent features of focal cortical dysplasia". In: *Epilepsia Open*. DOI: 10.1002/epi4.12529. URL: https://pubmed.ncbi.nlm.nih.gov/34328682/.

Hulshof, Hanna M, Emma MH Slot, et al. (2021). "Fetal Brain Magnetic Resonance Imaging Findings Predict Neurodevelopment in Children with Tuberous Sclerosis Complex". In: *The Journal of Pediatrics* 233, pp. 156–162. DOI: 10.1016/j.jpeds.2021.02.060. URL: https://pubmed.ncbi.nlm.nih.gov/33640330/.

Jahangard, Simindokht, Mohammad Hossein Zangooei, and Maysam Shahedi (July 17, 2020). "U-Net Based Architecture for an Improved Multiresolution Segmentation in Medical Images". In: *arXiv:2007.08238 [cs, eess]*. arXiv: 2007.08238. URL: http://arxiv.org/abs/2007.08238 (visited on 12/05/2021).

Jeong, Anna, Jo Anne Nakagawa, and Michael Wong (2017). "Predictors of drug-resistant epilepsy in tuberous sclerosis complex". In: *Journal of child neurology* 32.14, pp. 1092–1098. DOI: 10.1177/0883073817737446. URL: https://pubmed.ncbi.nlm.nih.gov/29129154/.

Jeste, Shafali S et al. (2016). "Symptom profiles of autism spectrum disorder in tuberous sclerosis complex". In: *Neurology* 87.8, pp. 766–772. DOI: 10.1212/WNL.0000000000003002. URL: https://pubmed.ncbi.nlm.nih.gov/27440144/.

Jnawali, Kamal et al. (2018). "Deep 3D convolution neural network for CT brain hemorrhage classification". In: *Medical Imaging 2018: Computer-Aided Diagnosis*. Vol. 10575. International Society for Optics and Photonics, p. 105751C. DOI: 10.1117/12.2293725. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10575/105751C/Deep-3D-convolution-neural-network-for-CT-brain-hemorrhage-classification/10.1117/12.2293725.short?SSO=1.

Jóźwiak, Sergiusz (2021). "Can we prevent epilepsy? Yes, we can!" In: URL: https://depot.ceon.pl/handle/123456789/20199.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2017). "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6, pp. 84–90. DOI: 10.1145/3065386. URL: https://dl.acm.org/doi/abs/10.1145/3065386.

Lao, Zhiqiang et al. (2004). "Morphological classification of brains via high-dimensional shape transformations and machine learning methods". In: *Neuroimage* 21.1, pp. 46–57. DOI: 10.1016/j.neuroimage.2003.09.027. URL: https://pubmed.ncbi.nlm.nih.gov/14741641/.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539.

Litjens, Geert et al. (2017). "A survey on deep learning in medical image analysis". In: *Medical image analysis* 42, pp. 60–88. DOI: 10.1016/j.media.2017.07.005. URL: https://pubmed.ncbi.nlm.nih.gov/28778026/.

Mazurowski, Maciej A et al. (2019). "Deep learning in radiology: An overview of the concepts and a survey of the state of the art with focus on MRI". In: *Journal of magnetic resonance imaging* 49.4, pp. 939–954. DOI: 10.1002/jmri.26534. URL: https://pubmed.ncbi.nlm.nih.gov/30575178/.

Mostapha, Mahmoud and Martin Styner (2019). "Role of deep learning in infant brain MRI analysis". In: *Magnetic resonance imaging* 64, pp. 171–189. DOI: 10.1016/j.mri.2019.06.009. URL: https://pubmed.ncbi.nlm.nih.gov/31229667/.

Nielsen, Michael A (2015). *Neural networks and deep learning*. Vol. 25. Determination press San Francisco, CA. URL: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=ZP0eZ94AAAAJ&citation_for_view=ZP0eZ94AAAAJ:j8SEvjWlNXcC.

Northrup, Hope et al. (2013). "Tuberous sclerosis complex diagnostic criteria update: recommendations of the 2012 International Tuberous Sclerosis Complex Consensus Conference". In: *Pediatric neurology* 49.4, pp. 243–254. DOI: 10.1016/j.pediatrneurol.2013.08.001. URL: https://pubmed.ncbi.nlm.nih.gov/24053982/.

Papastratis, Ilias (2021). "Introduction to Explainable Artificial Intelligence (XAI)". In: *https://theaisummer.com/*. URL: https://theaisummer.com/xai/.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Paus, Tomas et al. (2001). "Maturation of white matter in the human brain: a review of magnetic resonance studies". In: *Brain research bulletin* 54.3, pp. 255–266. DOI: 10.1016/s0361-9230(00)00434-2. URL: https://pubmed.ncbi.nlm.nih.gov/11287130/.

Ramirez, Kacy A et al. (2018). "Comparing molecular quantification of herpes simplex virus (HSV) in cerebrospinal fluid (CSF) with quantitative structural and functional disease severity in patients with HSV encephalitis (HSVE): implications for improved therapeutic approaches". In: *Journal of Clinical Virology* 107, pp. 29–37. DOI: 10.1016/j.jcv.2018.08.005. URL: https://pubmed.ncbi.nlm.nih.gov/30170224/.

Sanchez, Carmen E, John E Richards, and C Robert Almli (2012). "Neurodevelopmental MRI brain templates for children from 2 weeks to 4 years of age". In: *Developmental psychobiology* 54.1, pp. 77–91. DOI: 10.1002/dev.20579. URL: https://pubmed.ncbi.nlm.nih.gov/21688258/.

Sánchez Fernández, Iván et al. (2020). "Deep learning in rare disease. Detection of tubers in tuberous sclerosis complex". In: *PloS one* 15.4, e0232376. DOI: 10.1371/journal.pone.0232376. URL: https://pubmed.ncbi.nlm.nih.gov/32348367/.

Sarvamangala, DR and Raghavendra V Kulkarni (2021). "Convolutional neural networks in medical image understanding: a survey". In: *Evolutionary intelligence*, pp. 1–22. DOI: 10.1007/s12065-020-00540-3. URL: https://pubmed.ncbi.nlm.nih.gov/33425040/.

Selvaraju, Ramprasaath R et al. (2017). "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*, pp. 618–626. URL: https://openaccess.thecvf.com/content_iccv_2017/html/Selvaraju_Grad-CAM_Visual_Explanations_ICCV_2017_paper.html.

Shabanian, Mahdieh et al. (2019). "Classification of neurodevelopmental age in normal infants using 3D-CNN based on brain MRI". In: *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, pp. 2373–2378. DOI: 10.1109/BIBM47256.2019.8983399. URL: https://ieeexplore.ieee.org/abstract/document/8983399.

Shahid, Asim (2013). "Resecting the epileptogenic tuber: what happens in the long term?" In: *Epilepsia* 54, pp. 135–138. DOI: 10.1111/epi.12458. URL: https://pubmed.ncbi.nlm.nih.gov/24328887/.

Shetty, Anil N et al. (2019). "Cerebral oxygen metabolism during and after therapeutic hypothermia in neonatal hypoxic–ischemic encephalopathy: a feasibility study using magnetic resonance imaging". In: *Pediatric radiology* 49.2, pp. 224–233. DOI: 10.1007/s00247-018-4283-9. URL: https://pubmed.ncbi.nlm.nih.gov/30402807/.

Singh, Satya P et al. (2020). "3D deep learning on medical images: a review". In: *Sensors* 20.18, p. 5097. DOI: 10.3390/s20185097. URL: https://www.mdpi.com/1424-8220/20/18/5097.

Spinner, Thilo et al. (2019). "explAIner: A visual analytics framework for interactive and explainable machine learning". In: *IEEE transactions on visualization and computer graphics* 26.1, pp. 1064–1074. DOI: 10.1109/TVCG.2019.2934629. URL: https://ieeexplore.ieee.org/abstract/document/8807299.

Torre-Ubieta, Luis de la et al. (2016). "Advancing the understanding of autism disease mechanisms through genetics". In: *Nature medicine* 22.4, pp. 345–361. DOI: 10.1038/nm.4071. URL: https://pubmed.ncbi.nlm.nih.gov/27050589/.

Ueda, Masaru et al. (2019). "An age estimation method using 3D-CNN from brain MRI images". In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE, pp. 380–383. DOI: 10.1109/ISBI.2019.8759392. URL: https://ieeexplore.ieee.org/abstract/document/8759392.

Van Bel, Frank, Josine Vaes, and Floris Groenendaal (2019). "Prevention, reduction and repair of brain injury of the preterm infant". In: *Frontiers in physiology* 10, p. 181. DOI: 10.3389/fphys.2019.00181. URL: https://pubmed.ncbi.nlm.nih.gov/30949060/.

Vidiyala, Ramya (July 26, 2020). "Performance Metrics for Classification Machine Learning Problems". In: *toward data science*. URL: https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007.

Wang, Jieqiong et al. (2014). "Age estimation using cortical surface pattern combining thickness with curvatures". In: *Medical & biological engineering & computing* 52.4, pp. 331–341. DOI: 10.1007/s11517-013-1131-9. URL: https://pubmed.ncbi.nlm.nih.gov/24395657/.

Wang, Wei et al. (2019). "Development of convolutional neural network and its application in image classification: a survey". In: *Optical Engineering* 58.4, p. 040901. DOI: 10.1117/1.OE.58.4.040901. URL: https://www.spiedigitallibrary.org/journals/optical-engineering/volume-58/issue-4/040901/Development-of-convolutional-neural-network-and-its-application-in-image/10.1117/1.OE.58.4.040901.full?SSO=1.

Xu, Yiran et al. (2020). "Explainable object-induced action decision for autonomous vehicles". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9523–9532. URL: https://openaccess.thecvf.com/content_CVPR_2020/html/Xu_Explainable_Object-Induced_Action_Decision_for_Autonomous_Vehicles_CVPR_2020_paper.html.

Yamashita, Rikiya et al. (2018). "Convolutional neural networks: an overview and application in radiology". In: *Insights into imaging* 9.4, pp. 611–629. DOI: 10.1007/s13244-018-0639-9. URL: https://link.springer.com/article/10.1007/s13244-018-0639-9.

Vita

Mahdieh Shabanian was born in Iran in 1981. For more than 10 years, she has focused on artificial intelligence and machine learning models as an Electrical and Computer Engineer. In 2017 Mahdieh began her graduate career, pursuing a second Masters in Biomedical Engineering at the University of Tennessee Health Science Center (UTHSC). She has started implementing deep learning models for biomedical-related problems. As a data scientist, she started her collaboration with Le Bonheur Children's Hospital as an intern at the Center for Biomedical Informatics in the summer of 2018. Her research interests include combining statistical learning and machine/deep learning to obtain powerful computational models. In 2019, Mahdieh was invited to join the Children's Foundation Research Institute where she was able to continue her research on brain anomalies in infants. The focus of her work was using deep neural network models and machine learning applications to aid clinical decision-making before the onset of neurological sequelae. She has been analyzing neuroimaging data using deep learning methods to predict epilepsy in patients with tuberous sclerosis complex (TSC) at the TSC Center of Excellence, which is housed in Le Bonheur Children's Hospital. Mahdieh expects to receive her M.S. degree in December 2021.