# Deep Learning for Brain Tumor Classification

By

Justin Paul

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

May 2016

Nashville, Tennessee

Approved:

Daniel Fabbri, Ph.D.

Bennett Landman, Ph.D.

To my friends and family,

whose support and advice has made all of this possible

## ACKNOWLEDGEMENTS

This work would not have been possible without the mentorship and support from friends, family, and professors. I am especially thankful to Professor Daniel Fabbri, whose guidance and advisership has not only bolstered my career goals but also provided the academic setting to allow me to learn. He is an example of both what it means to be a good researcher and a good person. I additionally owe my gratitude to my professors in general, who have taught me the fundamentals needed to achieve higher learning. Lastly, I am indebted to my family and friends who have supported me throughout my time in academia. Their inspiration and encouragement has been at the center of my drive to learn and create.

TABLE OF CONTENTS

## LIST OF TABLES

LIST OF FIGURES

# Deep Learning for Brain Tumor Classification

**Justin Paul** JUSTIN.S.PAUL@VANDERBILT.EDU
Vanderbilt University

## Abstract

Deep learning has been used successfully in supervised classification tasks in order to learn complex patterns. The purpose of the study is to apply this machine learning technique to classifying images of brains with different types of tumors: meningioma, glioma, and pituitary. The image dataset contains 233 patients with a total of 3064 brain images with either meningioma, glioma, or pituitary tumors. The images are T1-weighted contrast enhanced MRI (CE-MRI) images of either axial (transverse plane), coronal (frontal plane), or sagittal (lateral plane) planes. This research focuses on the axial images, and expands upon this dataset with the addition of axial images of brains without tumors in order to increase the number of images provided to the neural network. Training neural networks over this data has proven to be accurate in its classifications an average five-fold cross validation of 91.43%.

## 1. Introduction

Patient diagnosis relies on a doctor's manual evaluation of a patient and his or her test results. With no automated tools to help with a doctor's diagnosis and limited number of available doctors, not only is there a higher risk of misdiagnosis but also an increase in wait time for patients to be seen. Doctors must take the time to manually review test results and images rather than spending time with the patient. In order to improve patient care, enhanced medical technology in the form of automated tools is necessary to increase doctor efficiency and decrease patient time in hospitals and time toward recovery.

The purpose of this research is to develop automated methods to aid doctors in diagnosis in order to prevent misdiagnosis and decrease patient wait time. In particular, this research achieves this automation through the classification of brain tumor types from patient brain images. Images require a doctor to examine multiple image slices to determine health issues which takes time away from more complex diagnoses. Our goal is to confidentally identify brain cancer types to reduce doctor burden, leaving the most complex diagnoses to them.

Previous research has developed specialized methods for automated brain tumor classification. Cheng et. al.[1] has created a brain tumor dataset containing T1-weighted contrast-enhanced images from 233 patients with three brain tumor types: meningioma, glioma, and pituitary. Additionally, the dataset has a variation of types of images: axial, coronal, and sagittal. Examples of these images can be seen in Figure 1. In their work, Cheng et. al. used image dilation and ring-forming subregions on tumor regions to increase accuracies of classifying brain tumors to up to 91.28% using a Bag of Words (BoW) model. In addition to BoW they also applied intensity histogram and gray level co-occurrence matrix (GLCM) with less accurate results[1]. This research improves on previously presented results using a more general method of neural networks and by adding images of brains without tumors.

Three main types of NNs have been researched: fully connected NNs (FCNNs), convolutional NNs (CNNs), and recurrent NNs (RNNs). For this study, CNNs are primarily used given that the inputs are images, though FCNNs are also examined. Though there have been prior attempts to apply machine learning to medical data such as the example above, there are a lack of tools utilizing modern advances in neural networks (NNs). While extensive research has successfully applied these techniques to recognizing patterns in non-medical images[2], the proposed research applies them to medical images which there is a lack of available datasets. Furthermore, applying neural networks to medical images has implications of faster and more precise diagnoses.

By including images of brains without tumors, neural networks can better learn the structure of a brain and take steps towards differentiating brains with and without tumors. More generally, this differentiates physiological structures through deep learning. Applying neural networks to medical images has implications of faster and more precise diagnoses automatically, and this research introduces neu-

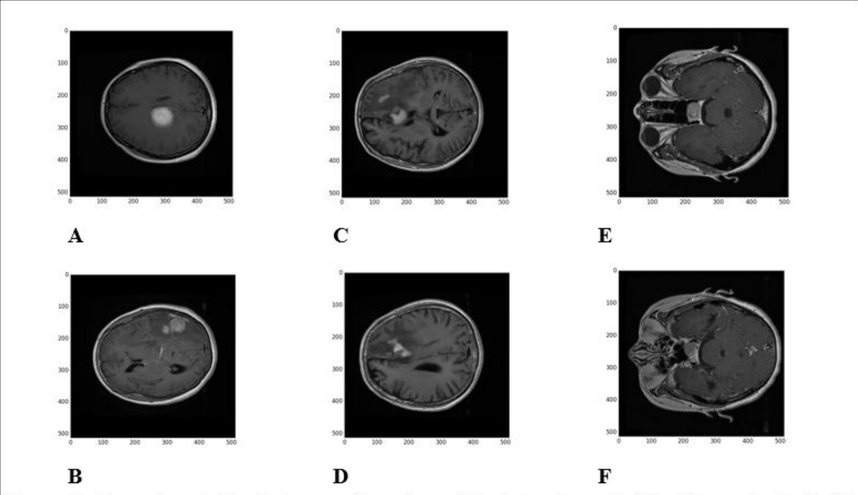


**Figure 1:** Example axial brain images from the public dataset provided by Cheng et. al. (**A, B**) Meningioma, (**C, D**) glioma, (**E, F**) and pituitary tumors show the variation in the dataset.

ral networks into the medical field where it has little current use. The main contributions of this paper are as follows:

- Create a more generalized method for brain tumor classification using deep learning
- Analyze the application of tumorless brain images on brain tumor classification
- Empirically evaluate neural networks on the given datasets with per image accuracy and per patient accuracy.

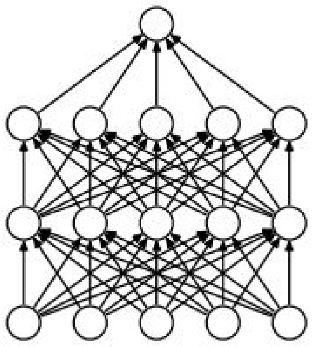
## 2. Related Work

A public brain tumor dataset was created from Nanfang Hospital, Guangzhou, China, and General Hospital, Tianjing Medical University, China from 2005 to 2012 and was used in Cheng et. al. 2015[1] to classify brain tumors in these images. Three previous approaches were used to analyze this dataset: intensity histogram, gray level co-occurence matrix (GLCM), and bag-of-words (BoW). Rather than only using the tumor region, Cheng et. al. augmented the tumor region by image dilation in order to enhance the surrounding tissue which could offer insights into the tumor type. Augmentation continued by using increasing ring formations helped by the image dilation and created by common normalized Euclidean distances in order to use spatial pyramid matching (SPM) to discover local features through computing histograms. In BoW, the local features are then extracted through dictionary construction and historgram representation, which are then fed into a feature vector to be trained on a classifier. Out of all three methods, BoW gave the highest classification accuracy with 91.28%. Yet, this classification method is highly specialized, requiring zooming into the tumor or region of interest and knowledge of tumor existence. On the contrary, neural networks are generalizable and can discover local features from image input alone.

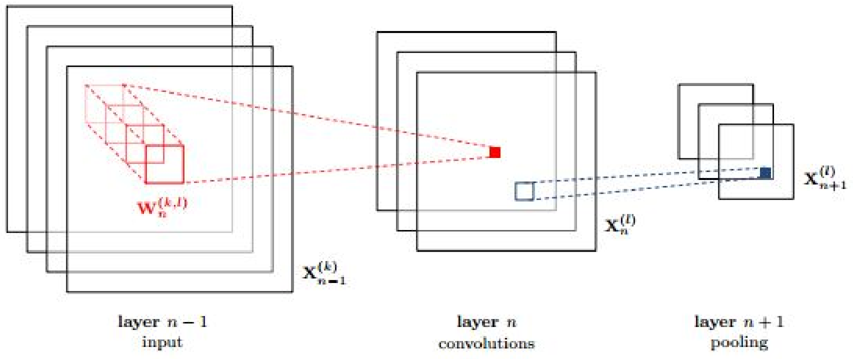
Neural networks and its generalizability has only appeared in recent years. After its rejection in the 1990's, deep learning came back into favor when Hinton et. al.[8] in 2006 introduced the method of pre-training hidden layers one at a time through unsupervised learning of restricted Boltzmann machines (RBMs). This demonstrated an effective method of training neural networks through greedily stacking RBMs. Since then the field of deep learning has expanded and produced more efficient methods of training neural networks and quickly became the state of the art. Examples of modern neural networks are shown in Figure 2.

While originally introduced into the public in 1998 by LeCun et. al.[3], convolutional neural networks gained popularity when in 2012 Krizhevsky et. al.[2] designed a winning convolutional neural network for the ImageNet competition and performed considerably better than the previous state of the art model. The computer vision community adopted neural networks as the state of the art after this competition, realizing the potential convolutional neural networks have on classification of images. Since 2012, convolutional neural networks have dominated other classification competitions, including the Galaxy Zoo Challenge that occurred from 2013 to 2014. Dieleman et. al.[7] introduced how data augmentation can greatly increase dataset size through transformations, rotations, and translations of images. This in turn prevented overfitting and more generalized learning.

Preventing overfitting in neural networks has been a main



**(A)**



**(B)**

**Figure 2. (A)** A standard fully connected neural network where each layer's node is connected to each node from the previous layer[5]. **(B)** A convolutional neural network connecting a covolutional layer to a pooling layer[7].

focus for much research, and in 2014 Srivastava et. al.[5] introduced dropout as a simple way to prevent co-adaptation of neurons. Dropout randomly drops neuron connections with a given probability, causing neuron units to become more independent rather than relying on other neurons to detect features. Similar in nature, maxout layers were designed to work in conjunction with dropout. Created by Goodfellow et. al.[9], maxout layers are equivalent to the standard feed-forward multilayer perceptron neural network, yet it uses a new activation function named the maxout unit. This unit takes the maximum linear activation for that unit.

In addition to overfitting, deep learning research has created faster ways to train neural networks. Glorot et. al.[12] revealed rectified linear units (ReLUs) performed much faster in supervised training of deep neural networks as compared to logistic sigmoid neurons and performed equal if not better than the hyperbolic tangent. This is due to ReLUs nonlinear nature where it creates sparse representations which work well for naturally sparse data. While ReLUs represent a change in nonlinearty application to improve learning, Nesterov's momentum[13] is a form of momentum update that has been adapted to neural networks. Nesterov's momentum takes the gradient at a future location following the the momentum from previous updates that has directed updates in a particular direction. This differs from standard momentum where the gradient is taken at the current location.

Neural network research has recently started to combine with medical research though still in its infancy stages. While prior research has shown promising results[10,14], only recently has access to large quantities of medical data begun to surface. Many of the concepts and ideas from past research in neural networks are directly applied to this study.

## 3. Model

Convolutional neural networks is a type of neural network designed specifically for images and have been shown to perform well in classifying various supervised learning tasks[3]. There have been several variations of convolutional neural networks created with commonalities in structure, including the use and ordering of convolutional, pooling, and dense layers.

### 3.1. Convolutional Neural Network

Convolutional neural networks were created with the assumption that nearby inputs are highly related to one another. In the case with images, the values are pixels, and pixels next to each other in images have a strong correlation with each rather than pixels further away in distance. With this assumption in mind, convolutional neural networks focus on local regions of the images in order to extrapolate local features from these subregions. This extrapolation of local features is performed in the convolutional layers.

As there is an increase in convolutional layers, these local features build upon one another to form higher-order features, combining to understand what the image is in its entirety. Extrapolating local features to higher order features is called using local receptive fields where a neuron in the convolutional layer takes in a particular k x j subregion of the image. In order for each neuron in the convolutional layer to take in various blocks of k x j pixels, convolutional layers can add in stride, which will shift the k x j pixels over by some given stride. For the best convolutional neural networks used in this research, a stride of 1 is used and values for k and j are 5 and 5 respectively. This implies that k x j pixel subregions can overlap with each other, which depending on the size of the stride, can typically help since nearby pixels are related in value.

Max-pooling layers are often paired with convolutional layers in order to reduce dimensionality in the neural network and augment pixel values slightly so as to make the layer insensitive to small changes in pixel values. Max-pooling is a type of subsampling or pooling layers which produce smaller images from there inputs by taking the max value over a k x j subregion to represent the entire subregion in the newly produced image. Common values of k and j are 2 and 3, and the best convolutional neural network from this research currently use k = 2 and j = 2 in order to cut the dimensionality to one-fourth of the size at each use. While averaging is another function used in pooling layers, each pooling layer used in this research applies the max function.

Convolutional and max-pooling layers in neural networks are often fed into fully connected or dense layers (i.e. all neurons in a layer are connected to each neuron in the following layer). Since fully connected layers have full connections to all activations in the previous layer, fully connected layers perform high-level reasoning in the neural network. For each neuron in every layer besides pooling layers, a nonlinearity function is applied in the convolutional neural network, otherwise layers could be collapsed into one since applying linear functions can be substituted with applying just one linear function. The nonlinear function used in this case is the rectified linear units which have proven to increase performance in neural networks[4].

The last layer in the neural network is a softmax layer of size 3 or 4 neurons which is used to determine classification probabilites. These neurons represent the probabilities of an image belonging to a particular category; three neurons are for the three types of brain tumors and the fourth optional neuron is for brains without tumors.

## 4. Algorithms and Implementation

Several algorithms are utilized in the construction of a neural network ranging from updating weights to calculating loss or error. This section will review the various algorithms incorporated into convolutional neural network and the specifics on implementing the layers mentioned in the previous section.

### 4.1. Forward Pass

Convolutional neural networks is a type of feedforward neural networks in which a forward pass of training is computed with no loops in neuron connections; the next layers must only be connected to previous layers. When moving to a convolutional or fully connected layer, a set of weights and bias is applied to all of the connected neurons from the previous layer in order to sum them together. This can be seen as applying a certain weight to a certain pixel and adding a bias. This formula can be seen below for a certain neuron i for a certain convolutional or fully connected layer $l$ receiving input.

$$a_i^l = \sum_{j=1}^{n} W_{ij}^l x_j + b_i$$

In this formula, j represents the certain input into neuron i. The nonlinearity ReLU is then applied to this sum $a_i^l$ to give neuron i in layer $l$ its new value of $z_i^l$.

$$z_i^l = max(0, a_i^l)$$

These two formulas are applied to every neuron in a convolutional or fully connected layer in order to obtain each neuron's value respectively. For max-pooling layers, the max function is applied over certain k x j subregions in order to get the max value as an output, and this is applied over the entire input keeping note of the given stride. The last layer contains the softmax function instead of the ReLU function in order to assign probabilities of the image being a certain type of tumor.

$$z_i^l = \frac{e^{a_i^l}}{\sum_k e^{a_k^l}}$$

The denominator represents the sum of all output neurons. This will give the predictions for an image by choosing the highest probability. In order to learn from these probabilities, first the loss or error of the predictions is calculated. To calculate loss, these convolutional neural networks use the categorical cross-entropy loss function.

$$L = \sum_j t_j \log(p_j)$$

In the above formula, t represents the target label, and p represents the prediction probability for the target label from our calculated predictions from the neural network. Given this summed error, an average categorical cross-entropy loss is calculated by dividing by the total number of examples in training m.

$$\frac{1}{m}L$$

In addition to categorical cross-entropy, it is common to add regularization to the loss in order to prevent weights from increasing too far in magnitude which is prone to overfitting. In this neural network, weight decay uses L1 normalization.

$$R = \frac{\lambda}{m}\sum_{w}|w|$$

In the above formula, w represents the weights in the neural network, m is the number of training examples, and lambda is the regularization contant / strength. The regularization constant is a hyperparameter which can vary based on the design of the convolutional neural network. In this convolutional neural network, a regularization strength of $10^{-4}$ is currently used. This regularization is combined with the categorical cross-entropy to give the overall cost function.

$$C = \frac{1}{m}L + R = \frac{1}{m}\sum_{j} t_j \log(p_j) + \frac{\lambda}{m}\sum_{w}|w|$$

### 4.2. Backwards pass

Neural networks now can use backpropagation to update weights and biases by propagating this error backwards through the neural network. This is propagated back until the inputs are reached, and the backpropagation has reached all parameter weights W and biases b, during which they are updated in order to minimize the overall cost. In order to change the parameters in a direction that minimizes the cost function, partial derivatives are used with respect to each parameter, starting with the partial derivative of the cost function with respect to weights and bias.

$$\frac{\partial C}{\partial W^l}, \frac{\partial C}{\partial b^l}$$

In the above formula, $l$ is the current layer (the last layer). The partial derivatives are used to update the weights connected to the last layer containing the softmax function. In order to continue to update previous layers weights and biases, the chain rule is applied from the current layer to the previous layer. This is done by finding the partial derivative with respect to the current layer z$^l$. Multipling this by the partial derivative of the previous layer with respect to its weights, biases, and its previous layer is shown below.

$$\frac{\partial C}{\partial W^{l-1}} = \frac{\partial C}{\partial z^l}\frac{\partial z^l}{\partial W^{l-1}}$$
$$\frac{\partial C}{\partial b^{l-1}} = \frac{\partial C}{\partial z^l}\frac{\partial z^l}{\partial b^{l-1}}$$
$$\frac{\partial C}{\partial z^{l-1}} = \frac{\partial C}{\partial z^l}\frac{\partial z^l}{\partial z^{l-1}}$$

This can be computed for any layer $l$ by continuation of backpropagation. Now the gradient for each parameter can be used to update the parameters by using Nesterov's momentum[6].

$$\hat{p}^l = p^l + \mu v^l$$
$$v^l = \mu v^l - \lambda\frac{\partial C}{\partial p^l}$$
$$p^l + = v^l$$

In the above equations, p represents a parameter, l is the layer, $\hat{p}$ is the look ahead in Nesterov's momentum, and $\mu$ is a hyperparameter momentum constant whose common values include [0.5, 0.9, 0.95, 0.99] (in this research $\mu$ is 0.9). With this new set of weights and biases that were just updated, the neural network has just completed one epoch, which consists of one forward and one backward iteration. Neural networks train through multiple epochs, and, for this research, 100 and 500 epochs are used to train the neural networks.

## 5. Approach

In this section, we describe the various brain image datasets and our approach to practically training our convolutional neural networks. We first describe the images in the brain tumor dataset and the brains without tumors dataset. We then describe processing and augmentation of images in order to gain more training data. Lastly, we discuss how the transformation of images is played into the software development.

### 5.1. Data

As stated previously, the brain tumor dataset contains 3064 T1-weighted contrast-enhanced images with images categorized into three sets: axial, coronal, and sagittal images. These represent the various planes images of the brain are scanned; they correlate with the transverse plane, frontal plane, or lateral plane planes respectively. There are 994 axial images, 1045 coronal images, and 1025 sagittal images where each image contained an original size of 512 x 512 in pixels. Furthermore out of all of the images, there were 708 slices of meningioma, 1425 slices of glioma, and 930 slices of pituitary tumors. These images originated from 233 patients, so many of the images are from the same patient.

In order to avoid confusing the convolutional neural network with three different planes of the brain that could have the same label, the images were separated into the three planes, and this paper focuses on the axial images due to the availability of tumorless brain images that were in the axial plane. This left us with a final brain tumor dataset of 191 patients and 989 images. Out of these images, we have 208 meningioma, 492 glioma, and 289 pituitary tumor images. The tumorless brains dataset contains 3120 axial 256 x 256 images from 625 patients where each patient is represented by 5 images that were randomly selected across the splices from their brain scans.

## 5.2. Preprocessing

Image data was preprocessed in several forms. Preprocessing included imitating previous research using a neural network or typical image preprocessing for neural networks which focuses only on the images themselves. Each form of preprocessing is described below.

### 5.2.1. Vanilla Data Preprocessing

Brain tumor images were originally formatted as int16, and tumorless brain images were formatted as float32. In order to compare the two, each was scaled to a range of 0 to 255.

### 5.2.2. Image and Location

This preprocessing applies only to the brain tumor dataset. The brain tumor dataset provided the tumor location for each image as a set of points that described the tumor boundaries. In order to provide this to a neural network, the maximum and minimum boundary point in the x and y directions were determined.

### 5.2.3. Tumor Zoom

Rather than provide the neural network the maximum and minimum boundary points in the x and y directions, these values were used in order to zoom into the tumor region of each brain scan. In order for each image to have a consistent size, the minimum box needed to contain every tumor was determined. To find this box, we found the mimum width and height needed to contain each tumor. The width was determined via the difference between the minimum x and maximum x, and the height was determiend via the difference between the minimum y and maximum y. This preprocessing was based on the note from Cheng et. al.[1], stating how the tissue surrounding the tumor can give insight into tumor type.

## 5.3. Image Transformation

Large image sizes has implications towards not only neural network training time but also memory issues. Original brain tumor images are 512 x 512 which creates memory problems when loading all of the images into the neural network. To prevent this issue, images were downsized to various sizes. While several image sizes were tested, we mainly discuss the polar ends of the downsizing since they performed the best in areas regarding accuracy and training time performance.

### 5.3.1. Large Image Size

Large image sizes consisted of images scaled to 256 x 256. This required brain tumor images to downsize from 512 x 512 while tumorless brain tumors remained the same size.

### 5.3.2. Small Image Size

Similarly to Dieleman et. al.[7], brain tumor and tumorless brain images were given as a large size of 512 x 512 and 256 x 256 respectively. While 512 x 512 images were too large of a memory requirement for our neural networks to handle, 256 x 256 images were small enough to run tests, though the speed of training was very high. In order to speed up training, images were cropped and shrunk. All images were resized to 512 x 512 and subsequently cropped to 412 x 412 by removing 50 pixels from each side. These pixels were majority unimportant, not withholding any information regarding the brain itself and holding constant pixel values of 0. Since all brain images were equally centered in their images, only minor portions of the edges of the brain images were affected. Images were then reduced to 69 x 69 in size by downscaling. This increased training speed by a factor of 10. A small image size of 64 x 64 was created as well by downsizing from the original size of 512 x 512 with similar increases in training speed.

## 5.4. Counteracting Overfitting

Convolutional neural networks have a high number of learnable parameters; the cutting edge neural networks have millions of learnable parameters relying on a large number of images in order to train. With the limited dataset from the brain tumor images, our neural networks were at a high risk of overfitting. Overfitting can occur when neural networks' weights memorize training data rather than generalize the input to learn patterns in the data. This can often happen due to small datasets. We applied several methods that prevent overfitting including data augmentation, regularization through dropout, and parameter sharing implied through rotations and transformations of images mentioned below.

Like many images, brain tumor image classifications are invariant under translations, transformations, scaling, and rotations. This allows for several forms of data augmentation to be exploited. Data augmentation has proven useful in expanding small datasets[7] to prevent overfitting. In a set of the tests run on the images, several forms of data augmentation were applied.

1. **Rotation**: Images were rotated with an angle between 0° and 360° that was randomly taken from a normal distribution.
2. **Shift**: Images were randomly shifted -4 to 4 pixels left or right and up or down[7]. These minor shifts were taken from a normal distribute and kept brains in the center of the image but changed the location of the brains enough to avoid memorization of location in an image rather than relative to the brain itself.
3. **Scaling**: Images were randomly rescaled using the scaling between $1.3^{-1}$ and 1.3 from Dieleman et. a.l.[7]
4. **Mirror**: Each image was mirrored across its y-axis (horizontally) with a probability of 0.5.

After these initial transformations, further augmentation was performed in order to increase the size of the training set each round. Each image was rotated 0° and 45° and flipped horizontally to create four images. These four images were then cropped to a size of 45 x 45 taking the four corners of the images as edges to produce 16 different images. The above data augmentation was run on the training data every epoch of training in order to constantly introduce new images to the neural network every iteration. This augmentation affected training time very little. We will call this processing step CO for counteracting overfitting.

#### 5.4.1. Crop Averaging

Following Krizhevsky et. al.[2], another form of data augmentation was implemented during training for 256 x 256 images, in which images were downscaled to 224 x 224 and five random patches of 196 x 196 were extracted from each training of these images in order to increase training data. When testing occurred, five 196 x 196 patches of each test image downscaled to 224 x 224 were extracted, one for each corner of the image and one for the center. The softmax probabilities for each of these images were then averaged together to give averaged softmax probabilties.

### 5.5. Network construction

A variety of neural networks were constructed based on the preprocessing of image data. Each is described in detail in this section.

#### 5.5.1. Convolutional Neural Network

This neural network represents taking only images as input. While many combinations of layers were tested, the best combination for this neural network was the following.

- Convolutional Layer with 64 filters of size 5 x 5 and stride of 1
- Max-pooling Layer with pool and stride size 2 x 2
- Convolutional Layer with 64 filters of size 5 x 5 and stride of 1
- Max-pooling Layer with pool and stride size 2 x 2
- Fully Connected Layer with 800 neurons
- Fully Connected Layer with 800 neurons
- Softmax Layer with 3 or 4 neurons depending on brain tumor only in training or tumorless brain inclusion in training respectively

Each layer besides max-pooling applied the nonlinearity ReLU, and each of the last three layers applied dropout to help in regularization and overfitting. We will refer to this neural network as CNN from now on in this paper.

#### 5.5.2. Fully Connected Neural Network

This neural network represents taking only images as input as well, but it does not utilize any convolutional or max-pooling layers. This network consisted of the following layers.

- Fully Connected Layer with 800 neurons
- Fully Connected Layer with 800 neurons
- Softmax Layer with 3 or 4 neurons depending on brain tumor only in training or tumorless brain inclusion in training respectively

Dropout and ReLUs were applied to each of these layers as well. We will refer to this neural network as FCNN from now on in this paper.

#### 5.5.3. Concatenation of Convolutional and Fully Connected Input Layers

This neural network represents providing more information than one image input. There are two version of the neural network. Each version has a neural network synonymous to CNN from above. However, a second input layer exists representing the same image input or the maximum and minimum x and y to represent the location of the tumor. These have their own neural network path that eventually concatenates with the CNN from before. This second neural network path consists of the following layers:

- Fully Connected Layer with 800 neurons
- Fully Connected Layer with 800 neurons

**Table 1.** Average Five-Fold Cross Validation Test Accuracies with Brain Tumor Images Only

| Model Details | | | | Per Image Accuracy | | | Per Patient Accuracy | | |
|---|---|---|---|---|---|---|---|---|---|
| Image Size | Preprocessing | Network | Epochs | Last | Best | Best-PD | Last | Best | Best-PD |
| 256 x 256 | Vanilla | CNN | 100 | 89.95 | 90.26 | 89.69 | **91.43** | 89.52 | **91.43** |
| 256 x 256 | Vanilla | FCNN | 100 | 87.30 | 87.32 | 87.46 | 86.67 | 85.71 | 86.67 |
| 256 x 256 | Vanilla | ConcatNN | 100 | 84.62 | 86.09 | 84.30 | 86.67 | 86.67 | 87.62 |
| 256 x 256 | Tumor Locations | ConcatNN | 100 | 85.66 | 85.96 | 85.80 | 87.62 | 87.62 | 89.52 |
| 207 x 312 | Tumor Zoomed | CNN | 100 | 88.99 | 88.16 | 88.99 | 88.57 | 88.57 | 88.57 |
| 69 x 69 | Vanilla | CNN | 100 | 81.70 | 82.46 | 81.44 | 79.05 | 82.86 | 79.05 |
| 45 x 45 | CO | CNN | 500 | 83.67 | 81.72 | 82.75 | 81.90 | 84.76 | 85.71 |
| 64 x 64 | Vanilla | CNN | 100 | 83.83 | 84.52 | 82.10 | 82.86 | 82.86 | 82.86 |
| 64 x 64 | Vanilla | FCNN | 100 | 80.86 | 80.43 | 81.30 | 77.14 | 76.19 | 77.14 |
| 196 x 196 | Crop Averaging | CNN | 100 | 86.77 | 87.65 | 88.16 | 82.86 | 83.81 | 84.76 |

**Table 2.** Average Five-Fold Cross Validation Test Accuracies with Brain Tumor and Tumorless Brain Images

| Model Details | | | | Per Image Accuracy | | | Per Patient Accuracy | | |
|---|---|---|---|---|---|---|---|---|---|
| Image Size | Preprocessing | Network | Epochs | Last | Best | Best-PD | Last | Best | Best-PD |
| 256 x 256 | Vanilla | CNN | 100 | 88.59 | **89.13** | 88.78 | 85.71 | 89.52 | 88.57 |
| 64 x 64 | Vanilla | CNN | 100 | 85.06 | 82.69 | 83.21 | 84.76 | 82.86 | 84.76 |
| 64 x 64 | Vanilla | FCNN | 100 | 84.51 | 86.30 | 84.05 | 82.86 | 84.76 | 81.90 |

The last layer of this path and the last fully connected layer from CNN were then concatenated together and connected to one last fully connected layer with 800 neurons before reaching the softmax layer from CNN. We will refer to this neural network as ConcatNN from now on in this paper.

## 5.6. Random Forests

Random Forests were created in 2001 by Breiman[11], and they are a combination of tree predictors where trees are dependent on randomly sampled independent vectors. Each tree is given features with minor amounts of perturbation in order inject noise in the data, and noise is further injected in the model level through randomization of attributes to split decisions on. While random forests are not neural networks, they have become a common technique in machine learning research in the medical field. Two separate prediction tests were conducted, one using only the brain tumor dataset and another using both the brain tumor datset and the tumorless brain dataset.

## 5.7. Training

For each of the preprocessed datasets, patients were randomly placed into three sets for training, validation, and test with 149, 21, and 21 patients respectively. A patient represents all of a patients images; this avoids mixing patient data in both training and test which allows for easier predictions since patient images are similar in structure. The mean picture from training was subtracted from train, validation, and test in order to centralize the data. An example mean picture can be seen in Figure 3. This was found to produce higher accuracies than cases without subtraction of the mean picture. Training data was used during the training of the neural networks in which train data was used for updating weights while validation data gave a glimpse into how the neural network was improving over time. After the training phase was completed, the test data was then used to see how well the neural networks predicted types of tumors from new images.

A variety of hyperparameters are available to alter. We list the hyperparameters that produced the highest accuracies.

- **Regularization constant:** 0.014
- **Learning rate:** 0.0001
- **Momentum constant:** 0.9
- **Batch size:** 4 for non-augmented datasets, 128 for augmented datasets
- **Epochs:** 100 (and one 500) which was compensation between accuracy and training time

### 5.7.1. Decaying Learning Rate

Rather than maintain a constant learning rate, a decaying learning rate was attempted in order to increase accuracies by decreasing the learning rate over time. However, each case of the decaying learning rate had significantly worse accuracies than without them.

### 5.7.2. Accuracy Metrics

Three different models were computed during validation in order to evaluate model performance on test data.

- Last: The trained model after the last epoch.
- Best: The model at the point of the best validation accuracy calculating per image accuracies.

- Best-PD: The model at the point of the best patient diagnosis validation accuracy calculating per patient accuracies.

For each of the above models, per image and per patient accuracies were applied to evaluate test performance. The results from the test evaluations are shown in Table 1 and 2.

## 6. Results

The accuracies for the conducted tests can be seen in Table 1 and Table 2. From these accuracies we can see the Vanilla CNN with image size 256 x 256 using the tumor brain dataset only has the highest accuracy at 91.43%. Furthermore, per patient accuracies proved consistent with per image accuracies, implying consistent predictions across patient images. Even with the extra compute time through the increase in epochs seen in CO FCNN with image size 45 x 45, the larger size images trained neural networks more accurately, producing 8% higher results. The weights from the top neural network's first convolutional layer can be seen in Figure 4. Minor structures representing low level features can be seen from each of these 5 x 5 weight regions. In order to further compare these models, the loss and accuracy history for training and validation sets were plotted for each model with five-fold cross validation (Figures 11 - 20). While the loss histories show the 256 x 256 images had overfitting over time due to lack of examples, their accuracies showed to consistantly be in a higher range than smaller images.

When looking specifically at the precision at k for these models (Figures 5 - 10), nearly all models remained above a 90%. Precision at k uses the top k predictions with the highest probabilities over all images. This represents the images neural networks were most confident in classifying. Having 90% accuracy consistently implies the predictions with the highest probabilities were often correct for any neural network. A particular note is that any model using 256 x 256 images had a precision of 1.0 from k = 1 to 20. Feeding larger images into the neural network ensured higher sucess of classification for a model's top predictions.

When comparing neural networks that used tumorless brain datasets and neural networks that did not, there were mixed results. For images of smaller size, adding tumorless brains tied or increased accuracies up to 2%. However for images of larger size, tumorless brains appeared to produce slightly less accuracies.

Analyzing the Vanilla CNN 256 x 256 for Brain Tumors Only neural network which performed the best overall in per image accuracy and in per patient accuracy, we look into the confidence, precision, and sensitivity in Tables 3 - 6. As seen from Figure 5, Vanilla CNN 256 x 256 earned a perfect score for average precision at k for k equals 1 to 20. In Table 3 and 4 we continue to increase k for each cross validation until the neural network has an incorrect prediction for per image and per patient accuracies respectively. For per image accuracy, the neural network averages reaching well over half of the test images before predicting an incorrect tumor type, with the best cross validation reaching 90% of images. For per patient accuracy, the neural network averages reaching over half of test patients as well, with the best cross validation predicting 100% of the patients correctly.

To see how the Vanilla CNN 256 x 256 for Brain Tumors Only neural network performs on each tumor type, we break down the tumor types into meningioma, glioma, and pituitary to evaluate the precision and recall for the Best model for per image accuracy and the Last model for per patient accuracy since these performed the best in their respective accuracy measure. These results can be seen in Tables 5 and 6 respectively. These two models performed the best in their respective accuracies. In Table 5, we can see meningioma tumors were the most difficult to predict with an average of 0.84 precesion and 0.74 recall, while glioma and pituitary had precision and recall in the mid-90%s. In Table 6, tumor type precision and recall is approximately equal for averages with 93%, 93%, and 91% for meningioma, glioma, and pituitary tumors respectively.

Lastly, random forest was run on both the brain tumor dataset only and with tumorless brain images. The former and latter gained averages close to 90% consistantly with consderable speed up as compared to training neural networks. Using tumorless brain images did not affect the accuracies.

## 7. Conclusion and Future Work

Convolutional neural networks are the state of the art in computer vision, and introducing them into the medical field could greatly improve current practices of diagnosing patients. Training convolutional neural networks to detect types of tumors in brain images improves classification accuracy and provides initial steps into introducing deep learning into medicine. Not only does this method produce equal and better results when compared to Cheng et. al.'s initial work, but neural networks also utilize a more general methodology requiring only an image to understand brain tumor types. Furthermore, the accuracy per patient metric consistently remained at the levels of per image accuracy results, implying the neural network is providing consistent predictions for patient images.

Future work can add upon this research by exploring neural networks that train on coronal and sagittal images. Furthermore, combining patient images across planes can not only increase dataset size but also provide further insights into

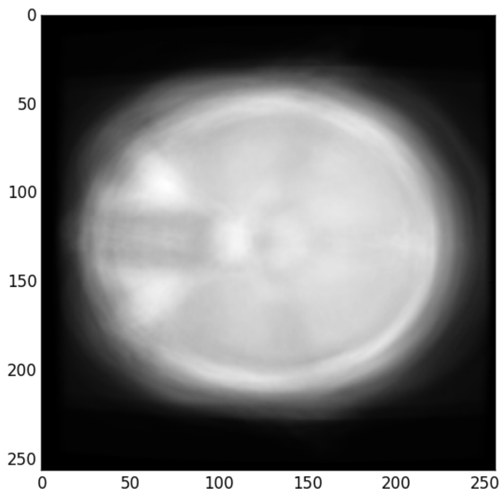


**Figure 3.** The average axial image across a training set.

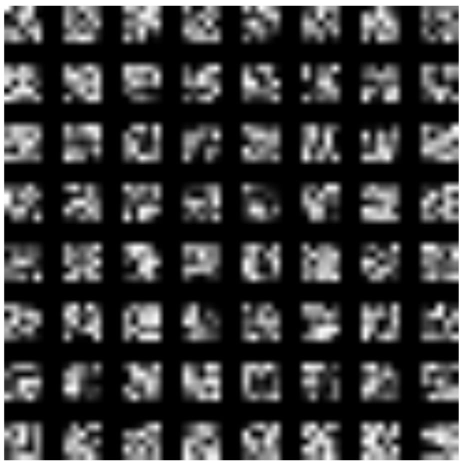


**Figure 4.** The 64 filters learned in the first convolutional layer of the best-performing neural network.

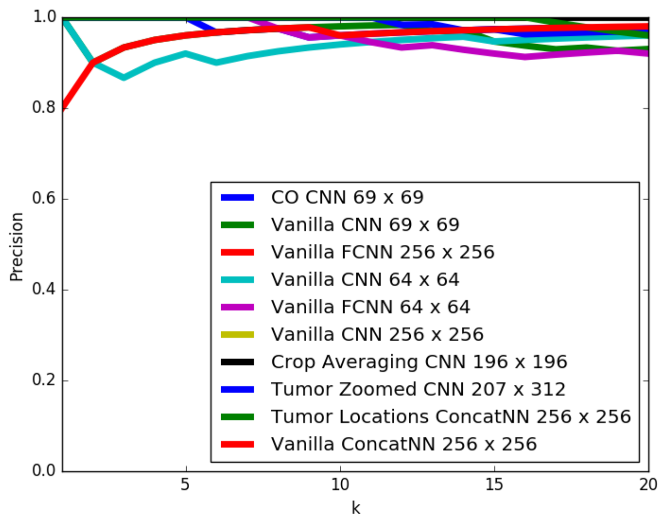


**Figure 5.** Average precision at k, where k ranges from 1 to 20, for Last models for per image accuracy. Vanilla CNN and Vanilla FCNN for 256 x 256 images and Crop Averaging for 196 x 196 images received perfect scores for k = 1 to 20.

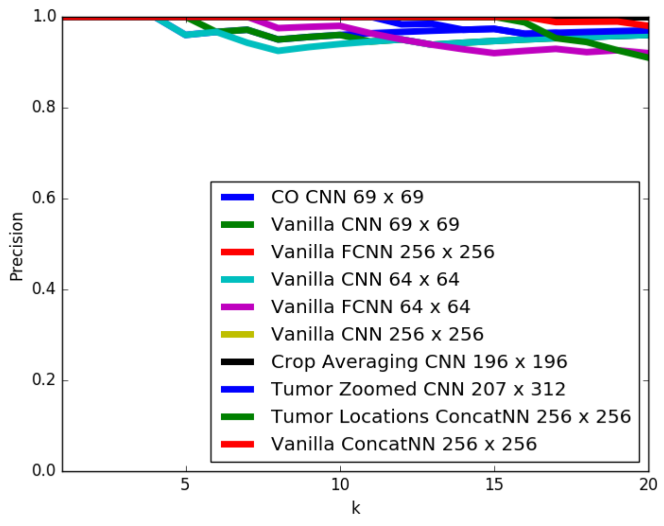


**Figure 6.** Average precision at k, where k ranges from 1 to 20, for Best models for per image accuracy. Vanilla CNN and Vanilla FCNN for 256 x 256 images and Crop Averaging for 196 x 196 images received perfect scores for k = 1 to 20.

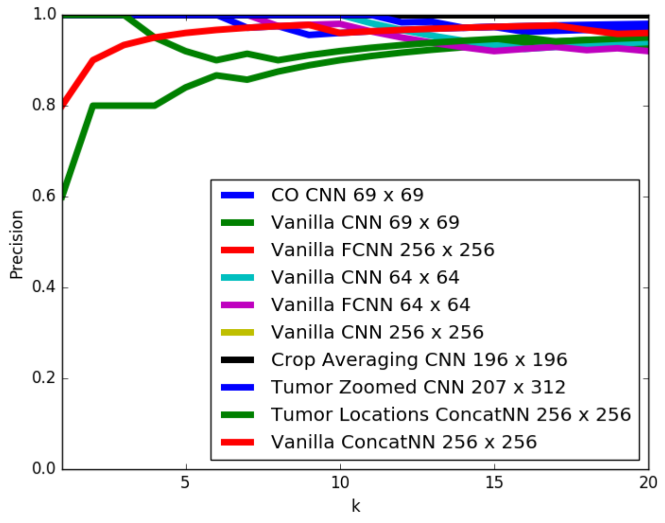


**Figure 7.** Average precision at k, where k ranges from 1 to 20, for Best-PD models for per image accuracy. Vanilla CNN and Vanilla FCNN for 256 x 256 images and Crop Averaging for 196 x 196 images received perfect scores for k = 1 to 20.

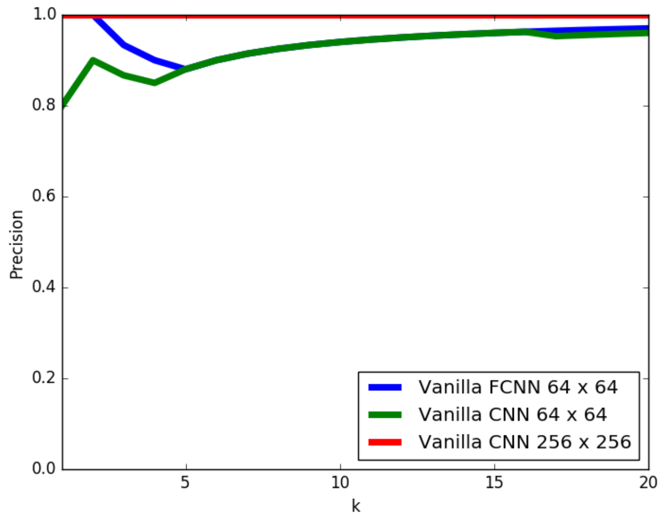


**Figure 8.** Average precision at k, where k ranges from 1 to 20, for Last models with tumorless brain images for per image accuracy. Vanilla CNN for 256 x 256 images received a perfect score for k = 1 to 20.

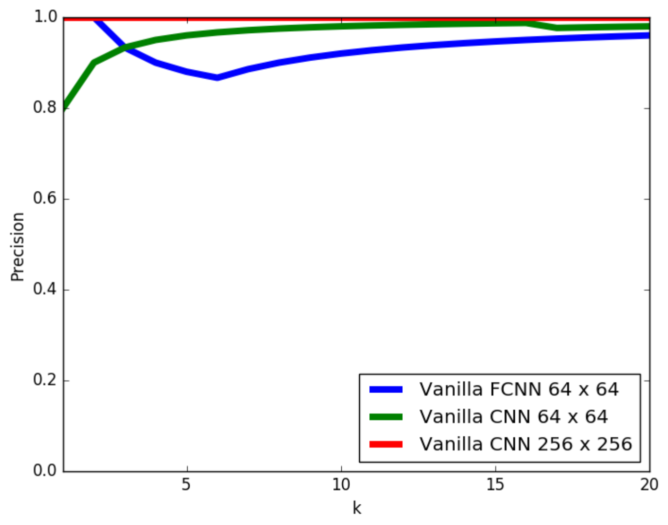


**Figure 9.** Average precision at k, where k ranges from 1 to 20, for Best models with tumorless brain images for per image accuracy. Vanilla CNN for 256 x 256 images received a perfect score for k = 1 to 20.

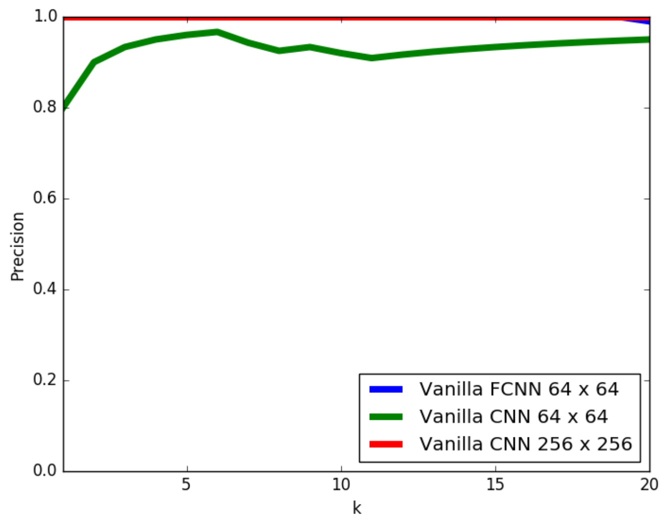


**Figure 10.** Average precision at k, where k ranges from 1 to 20, for Best-PD models with tumorless brain images for per image accuracy. Vanilla CNN for 256 x 256 images received a perfect score for k = 1 to 20.

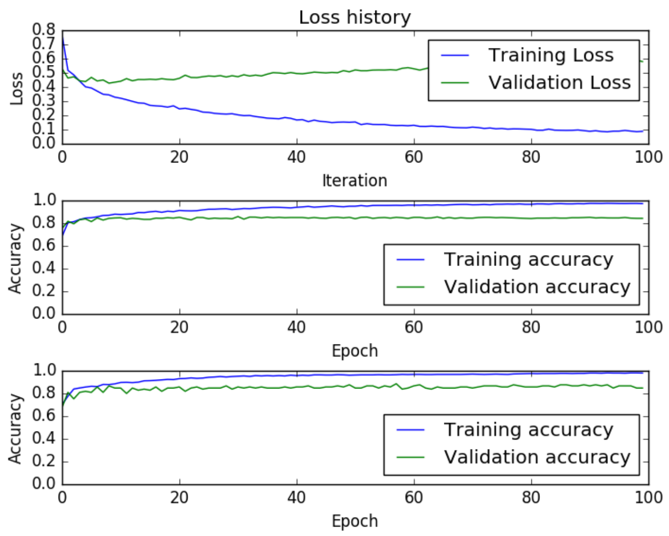


**Figure 11.** Loss and accuracy history for Vanilla FCNN with 256 x 256 images. The top graph represents loss over time, the middle graph represents per image accuracy over time, and the bottom graph represents per patient accuracy over time.

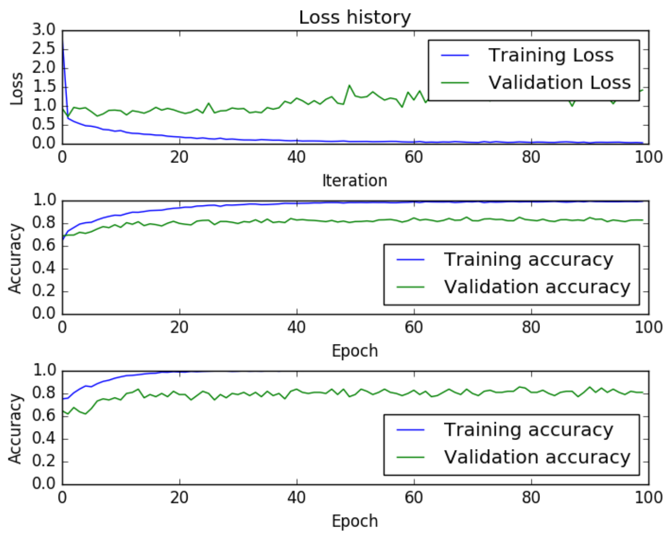


**Figure 12.** Loss and accuracy history for Vanilla CNN with 69 x 69 images. The top graph represents loss over time, the middle graph represents per image accuracy over time, and the bottom graph represents per patient accuracy over time.

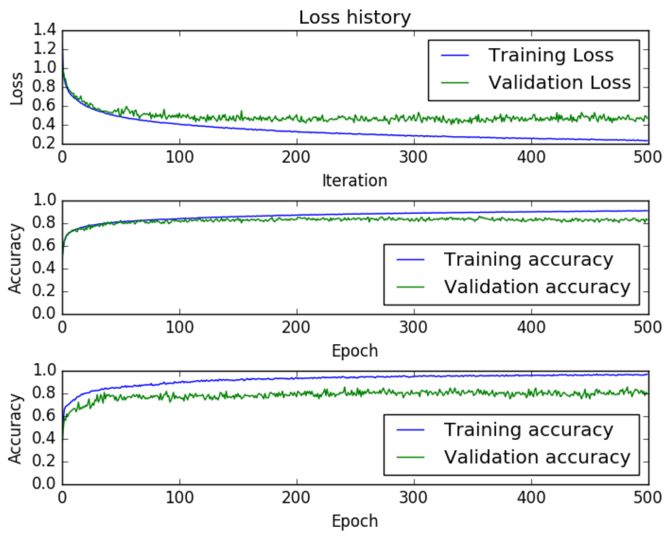


**Figure 13.** Loss and accuracy history for CO CNN with 45 x 45 images. The top graph represents loss over time, the middle graph represents per image accuracy over time, and the bottom graph represents per patient accuracy over time.

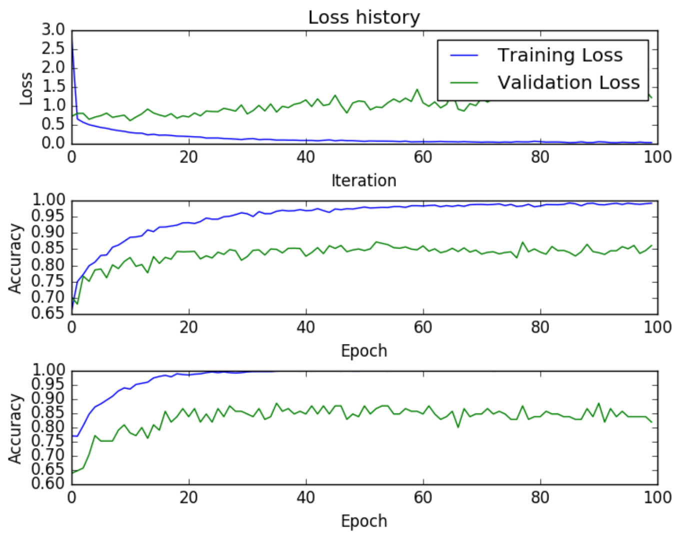


**Figure 14.** Loss and accuracy history for Vanilla CNN with 64 x 64 images. The top graph represents loss over time, the middle graph represents per image accuracy over time, and the bottom graph represents per patient accuracy over time.

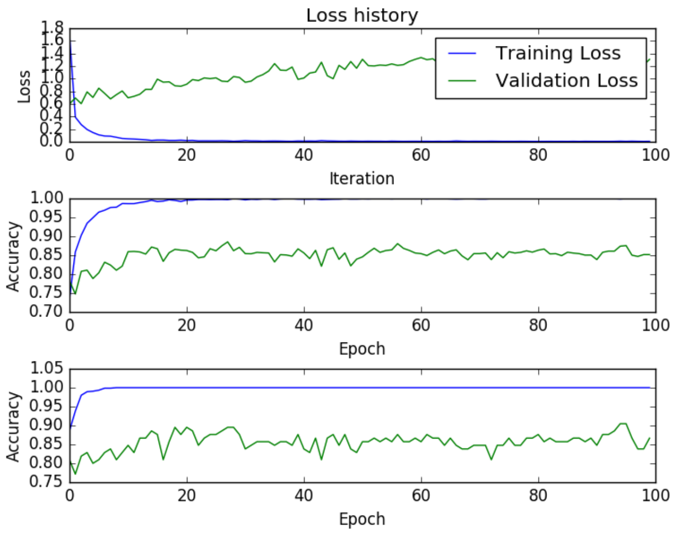


**Figure 15.** Loss and accuracy history for Vanilla CNN with 256 x 256 images. The top graph represents loss over time, the middle graph represents per image accuracy over time, and the bottom graph represents per patient accuracy over time.

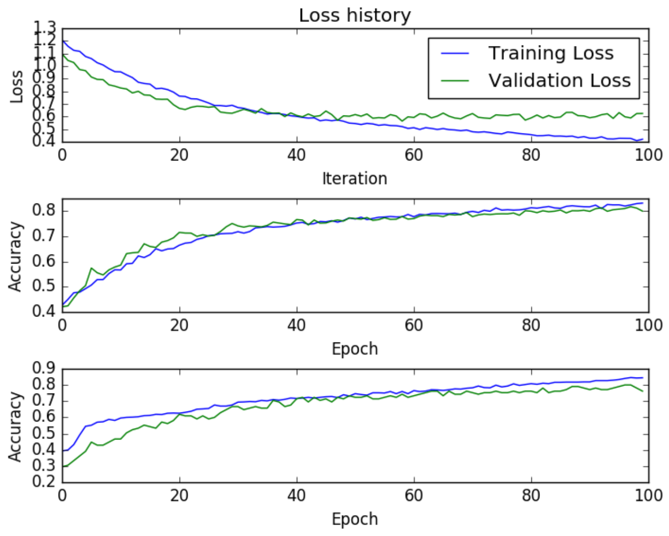


**Figure 16.** Loss and accuracy history for Vanilla FCNN with 64 x 64 images. The top graph represents loss over time, the middle graph represents per image accuracy over time, and the bottom graph represents per patient accuracy over time.

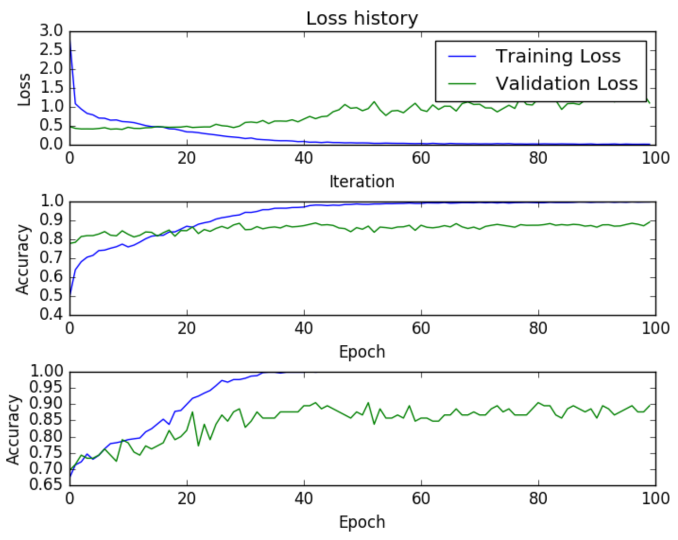


**Figure 17.** Loss and accuracy history for Vanilla ConcatCNN with 256 x 256 images. The top graph represents loss over time, the middle graph represents per image accuracy over time, and the bottom graph represents per patient accuracy over time.

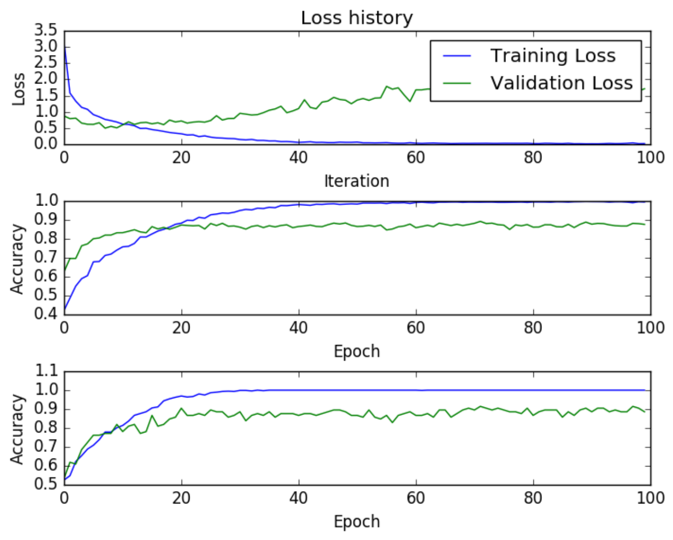


**Figure 18.** Loss and accuracy history for Tumor Locations ConcatCNN with 256 x 256 images. The top graph represents loss over time, the middle graph represents per image accuracy over time, and the bottom graph represents per patient accuracy over time.

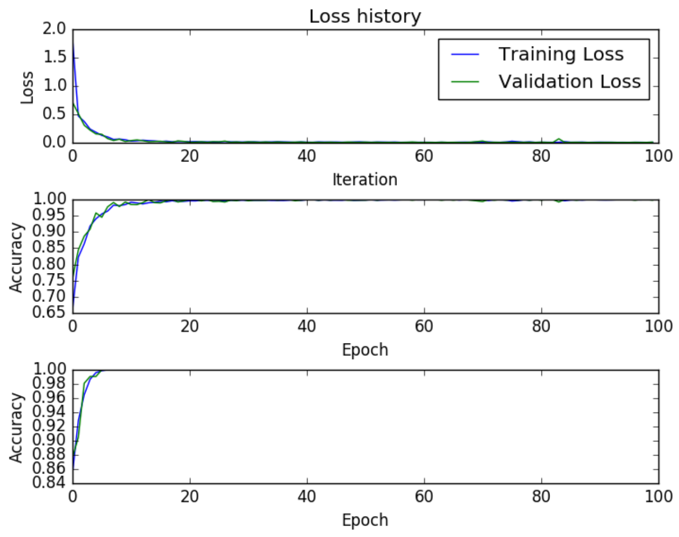


**Figure 19.** Loss and accuracy history for Tumor Zoomed CNN with 207 x 312 images. The top graph represents loss over time, the middle graph represents per image accuracy over time, and the bottom graph represents per patient accuracy over time. This model was trained on validation data.

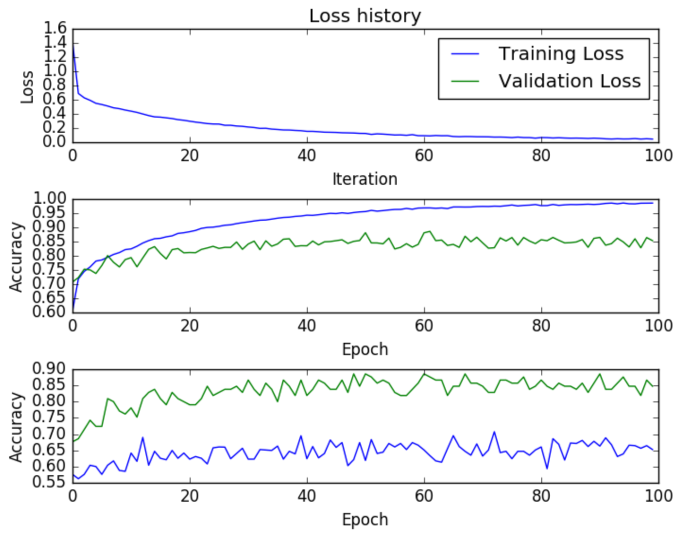


**Figure 20.** Loss and accuracy history for Crop Averaging CNN with 196 x 196 images. The top graph represents loss over time, the middle graph represents per image accuracy over time, and the bottom graph represents per patient accuracy over time.

**Table 3.**
Number Correctly Predicted Per Image in Order of Confidence Until Incorect Prediction for 256 x 256 Vanilla CNN

| Cross Validations | Number of Images | Last | Best | Best-PD |
|---|---|---|---|---|
| 1 | 82 | 53 | 56 | 35 |
| 2 | 111 | 23 | 22 | 23 |
| 3 | 126 | 81 | 75 | 84 |
| 4 | 119 | 107 | 110 | 106 |
| 5 | 110 | 86 | 66 | 63 |
| Average[1] | 109.6 | 70 | 65.8 | 61.8 |

[1]Averaged over Five-Fold Cross Validation.

**Table 4.**
Number Correctly Predicted Per Patient in Order of Confidence Until Incorect Prediction for 256 x 256 Vanilla CNN

| Cross Validations | Samples | Last | Best | Best-PD |
|---|---|---|---|---|
| 1 | 21 | 11 | 12 | 8 |
| 2 | 21 | 1 | 9 | 1 |
| 3 | 21 | 18 | 18 | 18 |
| 4 | 21 | **21** | **21** | **21** |
| 5 | 21 | 10 | 7 | 9 |
| Average[1] | 21 | 13.4 | 12.2 | 11.4 |

[1]Averaged over Five-Fold Cross Validation.

**Table 5.** Precision and Recall for Vanilla CNN 256 x 256 Best Per Image Model[2]

| Tumor Type | Cross Validations | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|---|
| Meningioma | 1 | 1.00 | 0.75 | 0.86 | 20 |
| | 2 | 0.84 | 0.62 | 0.71 | 34 |
| | 3 | 0.57 | 0.62 | 0.59 | 13 |
| | 4 | 1.00 | 0.88 | 0.94 | 26 |
| | 5 | 0.57 | 0.86 | 0.69 | 14 |
| | Avg / Total[1] | 0.84 | 0.74 | 0.78 | 107 |
| Glioma | 1 | 1.00 | 1.00 | 1.00 | 23 |
| | 2 | 0.80 | 0.93 | 0.86 | 55 |
| | 3 | 1.00 | 0.92 | 0.96 | 78 |
| | 4 | 0.93 | 1.00 | 0.96 | 41 |
| | 5 | 0.92 | 0.88 | 0.90 | 76 |
| | Avg / Total[1] | 0.93 | 0.93 | 0.93 | 273 |
| Pituitary | 1 | 0.89 | 1.00 | 0.94 | 39 |
| | 2 | 1.00 | 1.00 | 1.00 | 22 |
| | 3 | 0.82 | 0.91 | 0.86 | 35 |
| | 4 | 1.00 | 1.00 | 1.00 | 52 |
| | 5 | 1.00 | 0.80 | 0.89 | 20 |
| | Avg / Total[1] | 0.94 | 0.96 | 0.94 | 168 |

[1]Averaged over Five-Fold Cross Validation. [2] Best Per Image Model represents the Best model using the per image accuracy.

**Table 6.** Precision and Recall for Vanilla CNN 256 x 256 Last Per Patient Model[2]

| Tumor Type | Cross Validations | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|---|
| Meningioma | 1 | 1.00 | 0.57 | 0.73 | 7 |
| | 2 | 0.88 | 0.88 | 0.88 | 8 |
| | 3 | 0.75 | 0.75 | 0.75 | 4 |
| | 4 | 1.00 | 1.00 | 1.00 | 5 |
| | 5 | 1.00 | 0.83 | 0.91 | 6 |
| | Average[1] | 0.93 | 0.80 | 0.85 | 30 |
| Glioma | 1 | 1.00 | 1.00 | 1.00 | 4 |
| | 2 | 0.88 | 0.88 | 0.88 | 8 |
| | 3 | 1.00 | 1.00 | 1.00 | 10 |
| | 4 | 1.00 | 1.00 | 1.00 | 6 |
| | 5 | 0.85 | 1.00 | 0.92 | 11 |
| | Average[1] | 0.93 | 0.98 | 0.95 | 39 |
| Pituitary | 1 | 0.77 | 1.00 | 0.87 | 10 |
| | 2 | 1.00 | 1.00 | 1.00 | 5 |
| | 3 | 0.86 | 0.86 | 0.86 | 7 |
| | 4 | 1.00 | 1.00 | 1.00 | 10 |
| | 5 | 1.00 | 0.75 | 0.86 | 4 |
| | Average[1] | 0.91 | 0.94 | 0.92 | 35 |

[1]Averaged over Five-Fold Cross Validation. [2] Last Per Patient Model represents the Last model using the per patient accuracy.

tumor type that is difficult to view from only one plane. This can particularly improve meningioma tumors which caused neural networks the most difficult in classifying. Lastly, decreasing image size improved efficiency of training neural networks greatly. Improving performance on smaller images can have great benefits in training and assisting doctors in treatment of patients. Dealing with noisy, smaller images can help generalize neural networks to understand more complex brain images which in turn can help doctors in their diagnosis.

## 8. References

1. Cheng J, Huang W, Cao S, Yang R, Yang W, et al. (2015) Correction: Enhanced Performance of Brain Tumor Classification via Tumor Region Augmentation and Partition. PLOS ONE 10(12): e0144479. doi: 10.1371/journal.pone.0144479

2. A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, Proc. Neural Information and Processing Systems, 2012.

3. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. Proc. IEEE 86, 22782324 (1998).

4. Glorot, X., Bordes, A. & Bengio. Y. Deep sparse rectifier neural networks. In Proc. 14th International Conference on Artificial Intelligence and Statistics 315323 (2011).

5. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. J. Machine Learning Res. 15, 19291958 (2014).

6. Nesterov, Y. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). Soviet Mathematics Doklady, 27:372376, 1983.

7. Dieleman, S., Willett K., & Dambre J. Rotation-invariant convolutional neural networks for galaxy morphology prediction MNRAS;450:1441-1459 (2015).

8. Hinton, G. E., Osindero, S. & Teh, Y.-W. A fast learning algorithm for deep belief nets. Neural Comp. 18, 15271554 (2006).

9. I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In Proceedings of the 30th International Conference on Machine Learning, pages 1319 1327. ACM, 2013.

10. Toward content based image retrieval with deep convolutional neural networks JES Sklan, AJ Plassard, D Fabbri, BA Landman SPIE Medical Imaging, 94172C-94172C-6

11. Breiman, L. (2001). Random forests, Machine Learning 45: 532.

12. Glorot, X., Bordes, A. & Bengio. Y. Deep sparse rectifier neural networks. In Proc. 14th International Conference on Artificial Intelligence and Statistics 315323 (2011).

13. Nesterov, Y. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). Soviet Mathematics Doklady, 27:372376, 1983.

14. Lasko, T. et al. (2013) PLOS ONE;8:6