# BRAIN TUMOR CLASSIFICATION USING HIT-OR-MISS CAPSULE LAYERS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Spencer Chang

June 2019

## COMMITTEE MEMBERSHIP

| | |
|---|---|
| TITLE: | Brain Tumor Classification Using Hit-or-Miss Capsule Layers |
| AUTHOR: | Spencer Chang |
| DATE SUBMITTED: | June 2019 |
| COMMITTEE CHAIR: | John Seng, Ph.D. Professor of Computer Science |
| COMMITTEE MEMBER: | Maria Pantoja, Ph.D. Professor of Computer Science |
| COMMITTEE MEMBER: | Phillip Nico, Ph.D. Professor of Computer Science |

## ABSTRACT

Brain Tumor Classification Using Hit-or-Miss Capsule Layers

Spencer Chang

The job of classifying or annotating brain tumors from MRI images can be time-consuming and difficult, even for radiologists. To increase the survival chances of a patient, medical practitioners desire a means for quick and accurate diagnosis. While datasets like CIFAR, ImageNet, and SVHN have tens of thousands, hundreds of thousands, or millions of samples, an MRI dataset may not have the same luxury of receiving accurate labels for each image containing a tumor. This work covers three models that classify brain tumors using a combination of convolutional neural networks and of the concept of capsule layers. Each network utilizes a hit-or-miss capsule layer to relate classes to capsule vectors in a one-to-one relationship. Additionally, this work proposes the use of deep active learning for picking the samples that can give the best model, PSP-HitNet, the most information when adding mini-batches of unlabeled data into the master, labeled training dataset. By using an uncertainty estimated querying strategy, PSP-HitNet approaches the best validation accuracy possible within the first 12-24% of added data from the unlabeled dataset, whereas random choosing takes until 30-50% of the unlabeled to reach the same performance.

## ACKNOWLEDGMENTS

Thanks to:

- Andrew Guenther, for uploading this template
- My parents for supporting me and providing for me through my 6 years at Cal Poly.
- All the people who made LaTeX the amazing thing it is today.
- God and the Christian community for providing constant love and support during hard times.
- My friends who helped me by letting me bounce ideas and concepts off of them.
- All of the professors from whom I learned about computer engineering concepts, electrical engineering concepts, mathematics, life lessons, and graduate school.

TABLE OF CONTENTS

Page

LIST OF TABLES . . . . . viii

LIST OF FIGURES . . . . . ix

CHAPTER

1 Introduction . . . . . 1

1.1 Brain Tumors . . . . . 1

1.2 Machine Learning and Dataset Techniques . . . . . 2

1.3 Contribution . . . . . 3

1.4 Thesis Overview . . . . . 4

2 Background . . . . . 5

2.1 Brain Tumors . . . . . 5

2.2 Artificial Neural Networks . . . . . 6

2.3 Convolutional Neural Networks . . . . . 8

2.3.1 Residual Shortcut Connections . . . . . 10

2.3.2 Pyramid Scene Parsing Network . . . . . 11

2.3.3 Drawback of Pooling . . . . . 12

2.4 Capsule Networks . . . . . 13

2.4.1 Recent Adaptations . . . . . 17

2.4.2 Hit-or-Miss Transforms and HitNet . . . . . 19

2.5 Deep Active Learning . . . . . 20

3 Related Work . . . . . 24

3.1 Machine Learning Techniques for Tumor Classification . . . . . 24

3.2 CNNs for Tumor Classification . . . . . 25

3.3 Capsule Network Approaches . . . . . . . . . . . . . . . . . . . . . . 26
3.4 (Deep) Active Learning Techniques . . . . . . . . . . . . . . . . . . . 27
4 Deep Learning Model Design & Implementation . . . . . . . . . . . . . . . . 29
4.1 Original HitNet . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
4.2 Baseline HitNet . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
4.3 Residual HitNet (Res-HitNet) . . . . . . . . . . . . . . . . . . . . . 32
4.4 Pyramid Scene Parsing HitNet (PSP-HitNet) . . . . . . . . . . . . . . 33
5 Research Validation . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
5.1 Tumor Dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
5.2 5-Fold Cross-Validation . . . . . . . . . . . . . . . . . . . . . . . . 34
5.3 Equalized 5-Fold Cross-Validation . . . . . . . . . . . . . . . . . . . 35
5.4 Deep Active Learning Cross-Validation . . . . . . . . . . . . . . . . . 37
6 Results & Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . 40
6.1 Cross-Validation Results of Proposed Models . . . . . . . . . . . . . . 40
6.1.1 5-Fold Cross-Validation . . . . . . . . . . . . . . . . . . . . . . 40
6.1.2 Architecture Modifications . . . . . . . . . . . . . . . . . . . . 41
6.1.3 Equalized 5-Fold Cross-Validation . . . . . . . . . . . . . . . . . 42
6.2 Applying Deep Active Learning to Best Model . . . . . . . . . . . . . 49
7 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
8 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
BIBLIOGRAPHY . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

LIST OF TABLES

Table Page

1 Encoding Layers for Original HitNet [17] and Implemented Versions 32

2 Original Distribution of Tumor Slices in Dataset . . . . . . . . . . . 35

3 Equalized Distribution of Tumor Slices in Dataset . . . . . . . . . . 36

4 Image Split Per Class for 20% Unlabeled for the Fifth Fold . . . . . 38

5 Average Precision, Recall, and F1-Score for All Models after Training on the Original Dataset for 30 Epochs . . . . . . . . . . . . . . . . 43

6 Validation Accuracy Results for All Modifications to Base-HitNet, Res-HitNet, and PSP-HitNet . . . . . . . . . . . . . . . . . . . . . 44

7 Average Precision, Recall, and F1-Score for All Models after Training on the Equalized Dataset for 30 Epochs . . . . . . . . . . . . . . . 46

8 Average Precision, Recall, and F1-Score for PSP-HitNet after Training on the Data-Augmented Equalized Dataset for a Maximum of 400 Epochs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

9 Average Validation Accuracy per Active Learning Iteration Using SGD Optimizer without Regularization . . . . . . . . . . . . . . . . 49

10 Average Validation Accuracy per Active Learning Iteration Using SGD Optimizer with L2-Regularization . . . . . . . . . . . . . . . . 50

11 Results for 100 Highest-Confidence, 100 Random Choose, and 100 Lowest-Confidence per Active Learning Iteration . . . . . . . . . . 52

LIST OF FIGURES

Figure Page

1 Sampler of Brain MRIs with Tumors . . . . . . . . . . . . . . . . . 2

2 Example of Artificial Neural Network with Four Fully-Connected Layers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

3 Calculating the Output of a Neuron in an ANN Using the Activation Function $f$ Evaluated on Its Weighted Sum [2] . . . . . . . . . . . . 7

4 Convolutional Layer Passing a Filter Over an RGB Image [31] . . . 9

5 Max Pooling Layer Downsampling a Matrix of Values [31] . . . . . 10

6 Residual Learning Building Block [22] . . . . . . . . . . . . . . . . 11

7 Overview of PSPNet . . . . . . . . . . . . . . . . . . . . . . . . . . 12

8 Overview of the Dynamic Routing Algorithm . . . . . . . . . . . . 14

9 Original HitNet Architecture [17] . . . . . . . . . . . . . . . . . . . 19

10 Deep Active Learning for This Work . . . . . . . . . . . . . . . . . 22

11 Graphical Representation of HitNet [17] (top) and Baseline HitNet (bottom) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30

12 Implementation of Res-HitNet . . . . . . . . . . . . . . . . . . . . . 31

13 Comparison of PSPNet [61] (top) and PSP-HitNet (bottom) . . . . 31

14 Example of HitNet Architecture Modifications . . . . . . . . . . . . 36

15 Training and Validation Accuracy Results for All Models Using the Original Dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41

16 Results for Base-HitNet when Removing Pieces from the Architecture 43

17 Results for Res-HitNet when Removing Pieces from the Architecture 45

18 Results for PSP-HitNet when Removing Pieces from the Architecture 45

19 Training and Validation Accuracy Results for All Models Using the Equalized Dataset. . . . . . . . . . . . . . . . . . . . . . . . . . . 46

20 Results for PSP-HitNet after Training for 101 Epochs with Data Augmentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

21 Active Learning Results for Both Random and Uncertainty-Estimated Queries Using SGD without Regularization . . . . . . . . . . . . . . 50

22 Active Learning Results for Both Random and Uncertainty-Estimated Queries Using SGD with L2-Regularization . . . . . . . . . . . . . . 51

23 Resulting Predictions for MRI Scans and Corresponding Uncertainty Calculations . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

24 Active Learning Results for 100 Highest-Confidence, 100 Lowest-Confidence, and Random Sampling . . . . . . . . . . . . . . . . . . . 54

Chapter 1

# INTRODUCTION

The survival of patients with brain tumors is influenced by the speed of diagnosis. However, given an MRI image, it can be time-consuming even for neurologists to correctly classify the brain tumor and suggest a treatment plan. Computer aided diagnosis has become more popular in the research field not as a way of replacing the medical experts but as a way of providing second opinions on the diagnosis of a patient's condition. Various machine learning techniques have been applied to classifying the brain tumors from MRI images, including the use of Convolutional Neural Networks (CNN). If successful, deep learning can speed up diagnosis and help increase the survival chance for patients.

In this section, a quick overview is presented on brain tumors, deep learning networks, dataset techniques for improved performance, and this work's contribution.

## 1.1 Brain Tumors

A brain tumor is abnormal growth of cells in brain tissue. These tumors may be diagnosed as either benign or malignant, where malignant tumors grow quickly [41, 43]. While the benign tumor does not invade the surrounding tissue of a person's body, the malignant tumor will spread as such. A brain tumor is considered primary if it begins in the brain. On the other hand, secondary or metastatic tumors are from a primary tumor elsewhere in the body and move to the brain.

There exist more than 120 types of brain and central nervous system tumors [42]. Some of the most common diagnosed tumors are acoustic neuroma, astrocytoma, chordoma, glioma, medulloblastoma, meningioma, and pituitary tumors. To identify a patient's tumor, medical institutions use the World Health Organization (WHO) classification system [39, 42]. Doctors typically rely on techniques such as magnetic

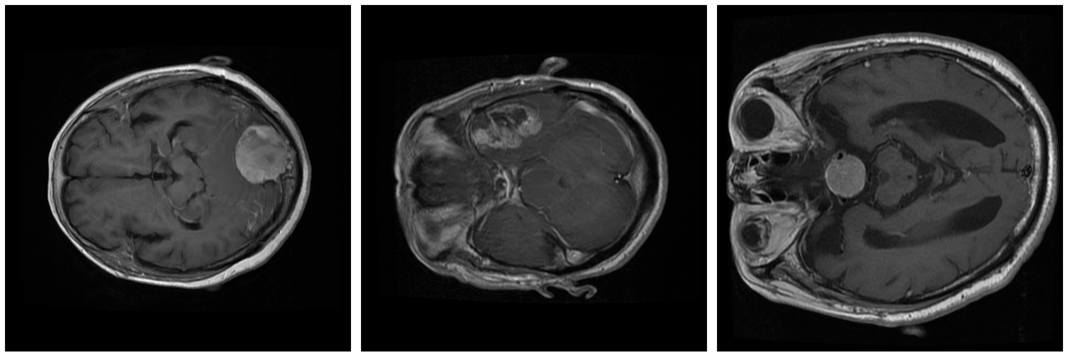


**Figure 1: Sampler of Brain MRIs with Tumors.** From left to right, tumors shown are classified as meningioma, glioma, and pituitary. Images are from the dataset by Cheng et al. [10].

resonance imaging (MRI), computed tomography (CT) scans, and biopsies to determine the existence and classification of a brain tumor. The particular dataset used in this work cover only meningioma, glioma, and pituitary tumors. Example brain MRI slices can be seen in Figure 1.

## 1.2 Machine Learning and Dataset Techniques

Multiple proposed solutions to computer aided diagnosis of brain tumors have come from a field of artificial intelligence called machine learning. A couple notable approaches try augmenting the tumor region of interest [10] or try combining multi-modal wavelet features with random forest classifiers [54]. Within the field of machine learning is deep learning, mainly represented by a set of learning models called artificial neural networks.

Among these networks, the CNN has become the most popular choice for computer vision tasks today [37]. However, with the way a CNN works, the global understanding of a single image can be lost with layer upon layer of convolution and pooling [48]. Toward the end of 2017, Sabour et al. presented a paper that formalized a new kind of artificial neural network based on a concept called capsules that was to improve upon the CNN by increasing robustness versus translational and rotational variances

[48]. Using the common MNIST data set, they demonstrated that their new network, dubbed CapsNet, could perform on-par with state-of-the-art CNNs and outperform them in recognizing overlapping digits, all while keeping a smaller network parameter space.

Since its introduction, multiple versions of the CapsNet have been created. These range from a focus on object segmentation [36] to a focus on natural language processing [7] and are covered in Chapter 3 on Related Works. In particular, Deliège et al. combined the encoding of input data using convolutional layers with a classifier made of vectors, or capsules, named the Hit-or-Miss layer [17].

Besides the introduction of new models, other well-known techniques like data augmentation and equalization of training data are used to improve the performance of proposed models. Other researchers, like Chitta et al. [11], focus on the active learning technique to modify how data is fed into the model. Active learning considers a model's uncertainty for predictions on data and feeds iterations or mini-batches of low-confidence samples into the master training dataset to train the network.

## 1.3 Contribution

Using the brain tumor dataset collected by Cheng et al. [10], three models are presented that have an average 5-fold cross-validation accuracy close to 85% on the tumor data without masking the tumors. Additionally, it is shown that when parts are removed within each of the networks, their performance remains similar to the unmodified version. Finally, this work will show that the active learning approach causes the best model to train the fastest when picking the lowest confidence samples as compared to picking randomly or the highest confidence samples.

## 1.4 Thesis Overview

In Chapter 2, this work will cover the background required for the tools and experiments that follow. Chapter 4 will cover the architecture of the network model and the routing algorithm used for the capsules. Then, evaluation is discussed in Chapter 5 with discussion and results following in Chapter 6. Finally, the work concludes with Chapters 8 and 7.

# Chapter 2
# BACKGROUND

In this section, basic background information is discussed concerning what is necessary for understanding the system implementation and the work's contribution to the field. Topics covered include a brief overview of the National Institute of Health's classification of brain tumors, discrete wavelet transforms, artificial neural networks, convolutional neural networks, capsule networks, and different implementations of capsule networks.

## 2.1 Brain Tumors

A tumor is an abnormal growth of cells within exterior or interior body organs. It is created when cells in a human body do not grow, divide, or die as they should in normal cells. A tumor may be classified as either benign or malignant. While benign tumors are noncancerous, they still require treatment to prevent them from spreading into other parts of the original organ. Malignant tumors are carcinogenic and might still reoccur after treatments [8].

Brain tumors that originate within the brain are termed primary. Other tumors that spread to the brain from other organs in the body are termed secondary. Primary brain tumors may be either malignant or benign, but secondary brain tumors are always malignant [18, 46, 55]. Brain tumors have multiple classes assigned by the World Health Organization and can be considered a meningioma, glioma, pituitary, or other kind of tumor [39, 42]. According to the American Cancer Society, in 2016, brain and spinal cord malignant tumors affected 23,800 adults and 100,000 children in the US [18].

To diagnose brain tumors, doctors will call for neurological tests like computerized tomography (CT) scans, magnetic resonance imaging (MRI), or even biopsy.

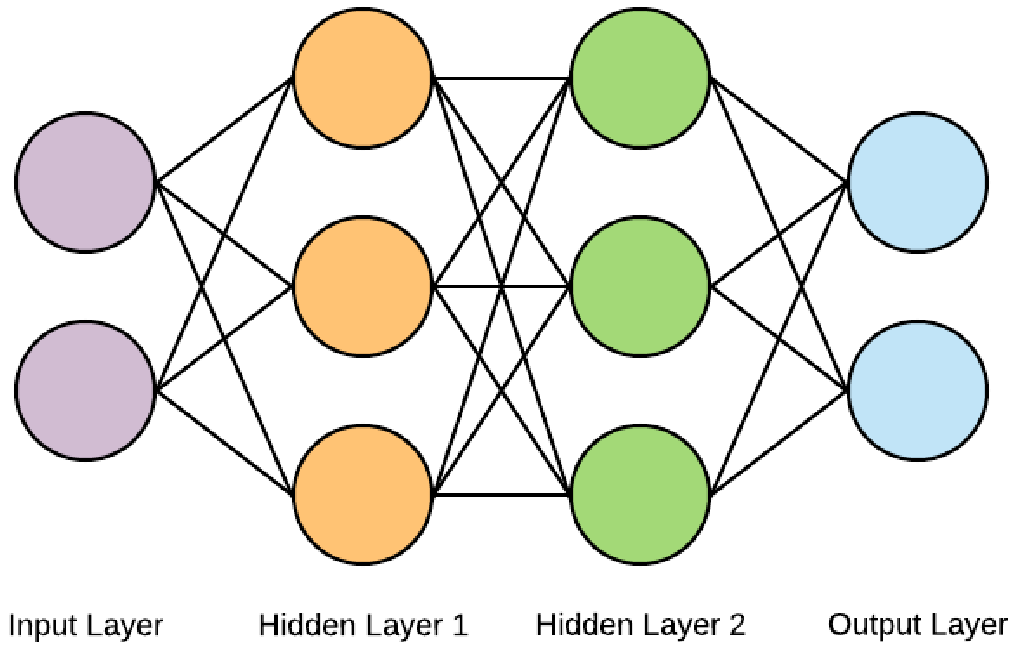


**Figure 2: Example of Artificial Neural Network with Four Fully-Connected Layers**

Following diagnosis, they will suggest a treatment plan for the patient, which could be anything from surgery to chemotherapy [41, 43].

## 2.2 Artificial Neural Networks

In the field of artificial intelligence, there is a subset of learning models categorized as deep learning models, which involve the progressive learning of increasingly layered representations of given data [12]. Among these advancements, artificial neural networks were some of the first models to be created and trained on training data for problems ranging from speech recognition to computer vision.

Artificial neural networks (ANN) are inspired by and loosely based on what a brain might look like. Both a brain and ANNs contain networks of neurons hooked to each other through synapses, or connections. However, this is as far as the similarities go between the two neural networks. ANNs are made up of many layers of neurons. An

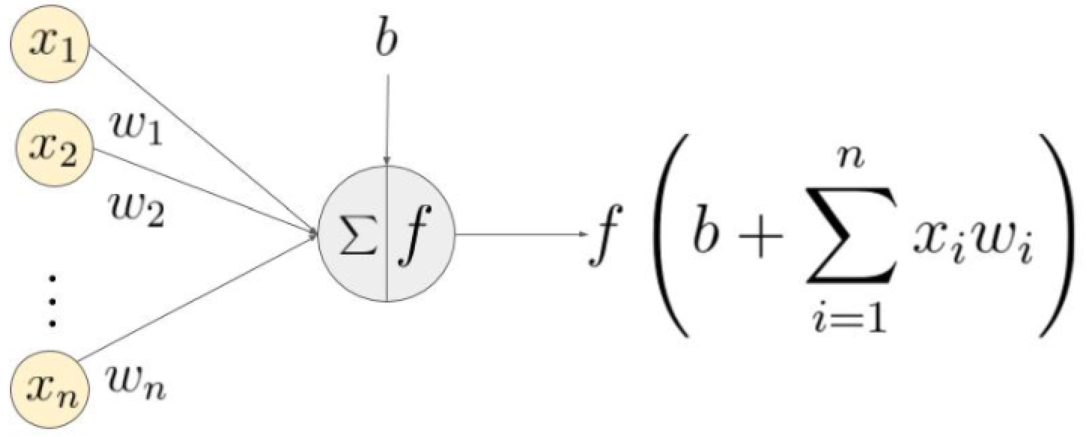


**Figure 3: Calculating the Output of a Neuron in an ANN Using the Activation Function $f$ Evaluated on Its Weighted Sum [2]**

example is seen in Figure 2. Connections from one layer to the next have associated weights that tell the network how heavily a neuron's output should be considered, as seen in Figure 3.

A neuron's output is determined by the incoming weighted sum and the response of its activation function $f$ to that weighted sum. The complete equation per neuron can be seen in Figure 3. At the initialization of an ANN, the weights $w_i$ are assigned random values or all 1's. After the input has passed through the ANN's activations, a loss function calculates the error between the predicted output and the desired output. Examples of loss functions include mean squared error, mean absolute error, and cross-entropy (Kullback-Leibler Divergence). The calculated loss then informs the ANN how its weights should be adjusted based on an averaged gradient per input batch for each neuron. This process of calculating the gradients is called backpropagation. The method of using incremental batches of training data to calculate gradients and find loss minima is called stochastic gradient descent (SGD). Over many passes of the training data through the ANN, backpropagation and SGD train the ANN weights to better represent data in such a way that its output matches the correct label for each training sample.

In Figure 2, a fully-connected neural network is shown. However, it is not the best type of model for solving computer vision problems, such as semantic segmentation or object recognition. To have a decent test accuracy, multiple fully-connected layers are needed, which results in tens of millions of parameters. A huge number of parameters increases the amount of information a network can memorize about its training data, but it does so at the cost of overtraining and not being able to generalize well to new, unseen test images. Thus, for computer vision problems such as object recognition, the most common network model used is the convolutional neural network.

## 2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) have become popular in the artificial intelligence industry for their ability to help solve a wide range of computer vision problems. The CNN uses a stack of convolutional and pooling layers to represent the input image with multiple channels, also called feature maps. These feature maps are then fed through the final section of the network called a classifier, which is most often a couple of fully-connected layers like those used in the example from Section 2.2. While a CNN also makes use of fully-connected layers like ANNs, the CNN has fewer trainable parameters and typically achieves better performance.

In a convolutional layer, a filter is applied to subsections in the image based on the filter's size, which can be 3x3, 5x5, etc. pixel grids. For each channel of a convolutional layer, there is a separate filter applied. In Figure 4, the layer convolves the RGB image to five separate feature maps. This means that one channel filter will have different values from another channel's. Convolutional layers can vary in the size of the filter used and the number of strides (pixels) between consecutive convolutions in a channel. These hyperparameters of the network will determine the kind of features learned from an image. The image may also be slightly subsampled by a few pixels depending on each layer's hyperparameters and whether the channels have a padding

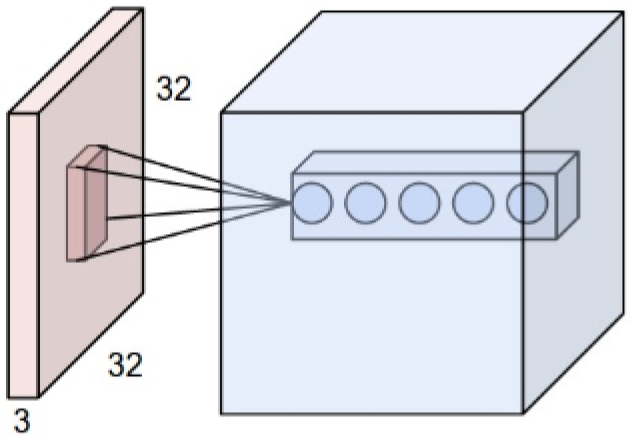


**Figure 4: Convolutional Layer Passing a Filter Over an RGB Image [31]**

of zeroes at their borders. After the convolutions are applied to a number of channels in a convolutional layer, the layer's output may feed into a pooling layer.

Like the convolutional layer, a pooling layer will focus on consecutive subregions of each feature map or channel, applying either an averaging or maximum function. Typically, max pooling is used to draw out the most prominent features in an image, which would be given by larger values in channels [37]. For instance, in Figure 5, max pooling is applied to bring out the largest values that correspond to the features a CNN may focus on to classify an input image. Compared to a convolutional layer, pooling layers do not have trainable parameters or weights. They only apply a function on each subregion as output to the next layer.

After the CNN has applied an arbitrary number of convolution and pooling operations on the input data, the neurons feed into a classifier. The classifier predicts the class of the input data. Often, the classifier is made up of fully-connected layers of neurons that take as input the final set of feature maps from the many convolutional and pooling operations applied to the original image.

Over the past decade, multiple versions of the CNN have been created. From those, a group of researchers has proposed residual connections called skip connections

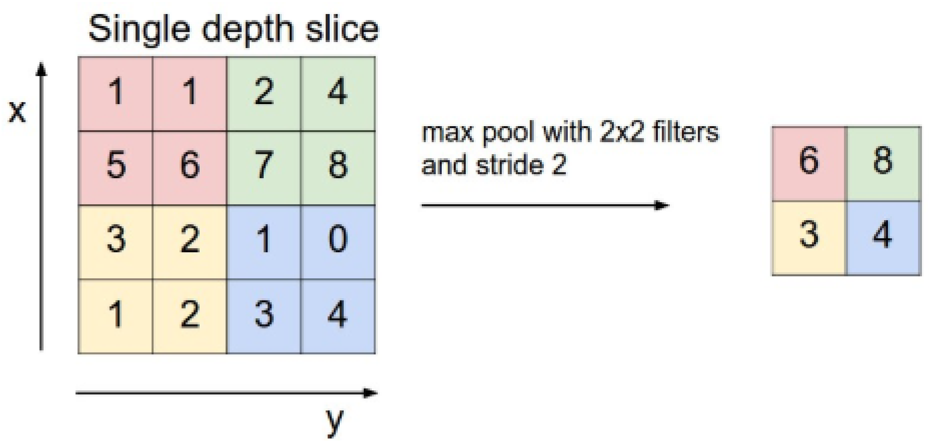


**Figure 5: Max Pooling Layer Downsampling a Matrix of Values [31]**

be included in a network to improve upon what kind of information is passed from the input to the classifier [22]. Another CNN version pertinent to this thesis creates an ensemble of branches that vote on pixel labels in cityscape and indoor scene images [61].

For future reference, CNNs in this thesis will be seen as going from lower layers to higher layers. For instance, the input layer and first convolutional layer are seen as the lowest two layers and the classifier is the highest layer of the network.

### 2.3.1 Residual Shortcut Connections

When deeper neural networks converge to their peak performance, a degradation problem can be exposed. If the network is continually trained past this point, the training accuracy becomes saturated and performance begins to drop off rapidly. According to sources cited by He et al. [22], the problem is not due to overfitting but that deeper models cause higher training error. To solve the problem of high training error from increasingly deep networks, He et al. propose the use of identity mappings from previous layers in a network [22]. These mappings are achieved by shortcut connections that skip one or more layers in the network and are added back into the mix by calculating $\mathcal{F}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x}$. By creating the shortcuts, the training error in

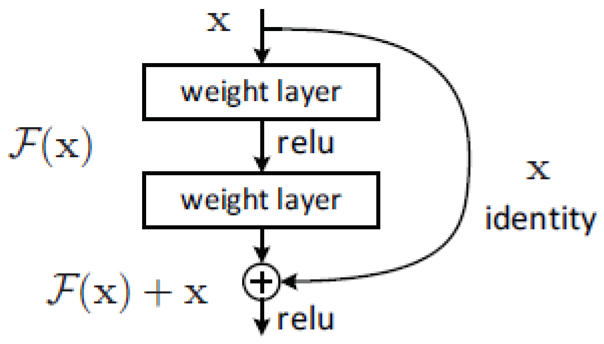


**Figure 6: Residual Learning Building Block [22]**

the higher layer should be no greater than in lower layers that map to it. A residual building block demonstrating this can be seen in Figure 6.

Residual networks are able to increase the depth of a CNN while keeping the computational complexity low. This is helpful when trying to compare plain and residual networks since the main difference lies in the element-wise addition from shortcuts. To deal with size differences between layers added together, 1x1 convolutions are passed over the lower layer at a given stride that will subsample its size to the same as the higher layer that receives the shortcut. Additionally, these shortcut connections can still be trained end-to-end by SGD with backpropagation. These shortcuts have been implemented in Python libraries such as Keras [33].

### 2.3.2 Pyramid Scene Parsing Network

Zhao et al. [61] tackled the problem of semantic segmentation of pictured scenes [61]. Labeled objects in these images range from pillows and beds to people and buildings. To get down to the specific pixels of each image, the researchers created what is called a pyramid scene parsing network (PSPNet) that took from a single pooling layer. After an initial convolutional layer, the network splits into four different convolutional branches with filter sizes of 1x1, 2x2, 3x3, and 6x6, respectively. Refer to Figure 7 for a depiction of their proposed module. After the branches are

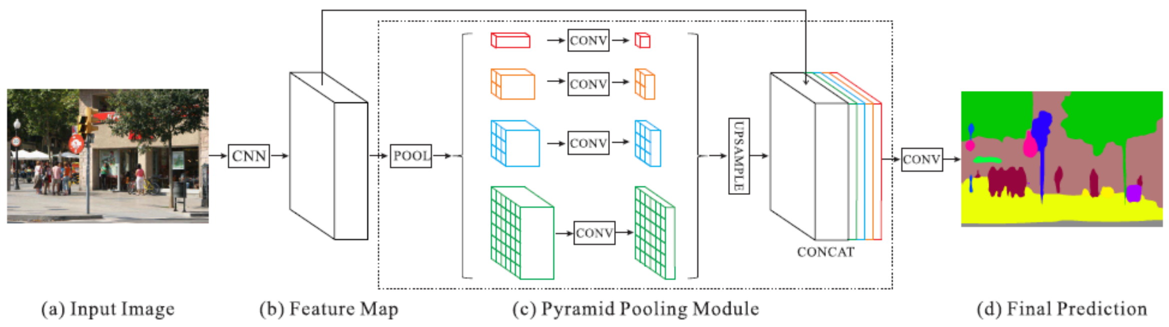


**Figure 7: Overview of PSPNet.** Given an input image (a), the CNN is used to get the feature map of the last convolutional layer (b), then a pyramid parsing module is applied to harvest different sub-region representations, followed by upsampling and concatenation layers to form the final feature representation, which carries both local and global context information in (c). Finally, the representation is fed into a convolution layer to get the final per-pixel prediction (d) [61].

computed, results are upsampled to the same size as the initial convolutional layer and concatenated together.

With the added shortcut connection from the first convolutional layer, PSPNet can factor in a global contextual prior when the four branches combine into the output segmented image. In total, the contextual prior and the four branches in the pyramid pooling module allow the network to learn representations at different levels by looking at coarse and fine sub-regions within the image. Zhao et al. [61] used the PSP module for semantic segmentation, but by subsampling the initial layer with 1x1 convolutions, it can be adapted toward object recognition and will be detailed further in Chapter 4.

### 2.3.3 Drawback of Pooling

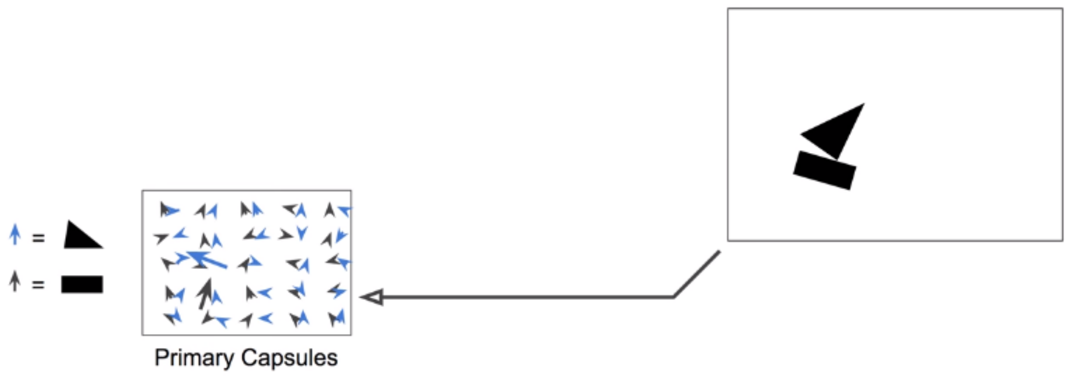
While CNNs have made great progress in solving computer vision problems for robot navigation and in facial recognition, there are a couple issues that make it unsuitable in going forward with CNNs as the main ANN applied to computer vision. One major issue is that pooling layers cause a loss of global connectivity throughout the CNN [48]. This means that a CNN may see separate parts in an image, but it will not be able to tell how the two parts are spatially connected. While the aforementioned versions [22, 61] can alleviate this problem, the CNN also suffers from rotational and translational variations in an image. Some more CNN versions have attempted to create more robust models [58]. In their response to these issues, Hinton et al.[25] mentioned a new kind of network based on 'capsules', which were formally presented and implemented by Sabour et al.[48].

## 2.4 Capsule Networks

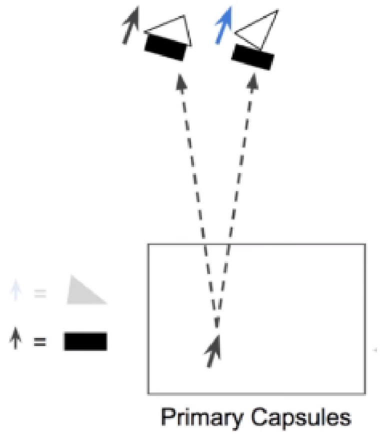
The first idea of a capsule was presented by Hinton et al. in a paper about transforming autoencoders [25], but the idea was not formally implemented and tested until a paper by Sabour et al. [48] at the end of 2017. A capsule is a group of neurons whose activity vector represents the instantiated parameters of a specific type of entity, such as an object or object part. According to Sabour et al., in the future, capsule networks (CapsNet) will replace CNNs, which are the current state-of-the-art in computer vision problems.

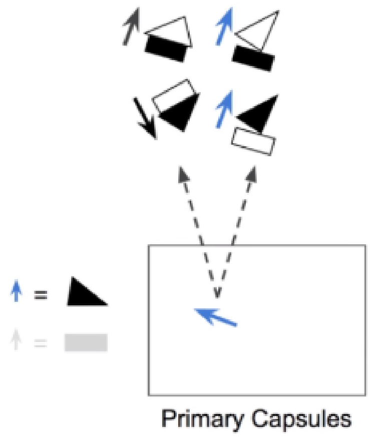
At a high-level, capsules represent an entity's properties using a vector as opposed to a scalar. This vector has properties such as the entity's pose, and its length corresponds to the probability that the entity exists in any given input. When presented by Sabour et al. [48], CapsNets consist of a convolutional layer leading into two capsule layers, what are called primary and 'digit' capsules. The second layer of capsules could be generalized to be called 'class' capsules, as this is what the particular layer represents. To classify an image, the primary capsule layer is trying to predict the output of the class capsule layer. To train these predictions, the capsule layers use an algorithm called "routing by agreement", where primary capsules will only route to parent capsules in the class capsule layer that agree with its own output.

In Figure 8, three main portions of the routing algorithm are pictured. When first entering the primary capsule layer, vectors are calculated to check for any existing



(a)





(b) (c)

**Figure 8: Overview of the Dynamic Routing Algorithm.** [20] (a) how the primary capsules look when computed (b) the between-capsule agreement for the rectangle portion (c) the between-capsule agreement for the triangle portion. The poses of either entity are pictured in a-c.

entities, as seen in (a). There might be a little bit of noise within the image, but the largest pose and length shown are that for the triangle and rectangle. Suppose the CapsNet is trying to figure out if the picture is an image of a sailboat or a house. Going from primary capsules to class capsules, each capsule is checked to see if they could signal the existence of either class. In (b), the rectangle is seen routing to both the house and sailboat class. In (c), the triangle is also routed to the house and sailboat class. However, the triangle's pose does not agree with the rectangle's pose when routing to the house class. On the other hand, the triangle and rectangle have strong agreement for the sailboat class. So, the routing algorithm will simultaneously weaken the routing connection from the primary capsules to the house class capsule and strengthen the routing connection from the primary capsules to the sailboat class capsule.

In the work by Sabour et al. [48], the following equations are presented as means of mathematically implementing the routing by agreement algorithm, particularly using a straightforward implementation called dynamic routing. As noted in their paper, this implementation is only a way to show that capsules do work fairly well.

For both capsule layers, the length of the output vector represents the probability that the entity represented by a particular capsule is present in the input. To do this in a non-linear fashion while preventing large values from dominating the capsules, a "squashing" function is created. As seen in Equation 2.1, vectors are squashed such that short vectors are shrunk down to nearly zero length while long vectors are shrunk down to a vector length slightly below one.

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1+||\mathbf{s}_j||^2}\frac{\mathbf{s}_j}{||\mathbf{s}_j||} \tag{2.1}$$

$\mathbf{v}_j$ is the vector output of capsule $j$ and $\mathbf{s}_j$ is its total input, given by the weighted sum in Equation 2.2. Excluding the first layer of capsules, the total input to a capsule $\mathbf{s}_j$ is calculated from a weighted sum over all "prediction vectors" $\hat{\mathbf{u}}_{j|i}$ from capsules in the previous layer multiplied by corresponding "coupling coefficients" $c_{ij}$ between the capsule layers. $\hat{\mathbf{u}}_{j|i}$ is created from multiplying the output $\mathbf{u}_i$ of capsules in the previous layer by a weight matrix $\mathbf{W}_{ij}$ going from capsule $i$ to capsule $j$ in the next layer. The first layer of capsules gets its input from a previous non-capsule layer, or a convolutional layer [48].

$$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}, \quad \hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i \tag{2.2}$$

In Equation 2.3, the coupling coefficients $c_{ij}$ between the capsule layers sum to 1 using a "routing softmax", where the $b_{ij}$ are log prior probabilities that capsule $i$ should be coupled to capsule $j$.

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{2.3}$$

In the original CapsNet paper, a margin loss, seen in Equation 2.4 was implemented to have differing losses based on whether the particular MNIST digit exists in the input or not. For the loss $L_k$ below, $T_k = 1$ iff a digit of class $k$ is present, and $m^+ = 0.9$, $m^- = 0.1$, and $\lambda = 0.5$ for the implementation. The margin loss is calculated for each class capsule, resulting in a total loss that is a sum of all class capsules' margin losses.

$$L_k = T_k \max(0, m^+ - ||\mathbf{v}_k||)^2 + \lambda(1 - T_k)\max(0, ||\mathbf{v}_k|| - m^-)^2 \tag{2.4}$$

The agreement between capsules is the scalar product $a_{ij} = \mathbf{v}_j \cdot \hat{\mathbf{u}}_{j|i}$. The resulting product is treated as if it was a log likelihood and is added to $b_{ij}$ before computing new values for all coupling coefficients between capsule $i$ and the next layer's capsules.

In Algorithm 1, the dynamic routing algorithm is presented. As seen, routing by agreement involves several loops that will repeatedly strengthen couplings for correct capsule links and weaken couplings for incorrect capsule links. Lines 4 and 6 correspond to Equation 2.3 and 2.1, respectively. Compared to CNNs, CapsNets are meant to take the entire input into account when computing capsule outputs. In the original work [48], the input for primary capsules was 6x6 since a larger size

**Algorithm 1** Dynamic Routing Algorithm [48]

```
1: procedure ROUTING($\hat{\mathbf{u}}_{j|i}$, $r$, $l$)
2:     for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3:     for $r$ iterations do
4:         for all capsule $i$ in layer $l$: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
5:         for all capsule $j$ in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
6:         for all capsule $j$ in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
7:         for all capsule $j$ in layer $l$ and capsule $j$ in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$
8:     end for
9:     return $\mathbf{v}_j$
10: end procedure
```

would result in a substantial increase in trainable parameters, causing the CapsNet to quickly become unwieldy in size.

A more recent paper was written by Hinton et al. [24] in 2018 that trained and tested on the smallNORB dataset. In particular, the routing mechanism instead used expectation-maximization (EM) to strengthen or weaken couplings. However, other implementations of CapsNets have surfaced recently that help demonstrate that CapsNets are a promising future area to research in place of CNNs.

### 2.4.1 Recent Adaptations

Due to the recent introduction of CapsNets, good results on large datasets like ImageNet [35] and Pascal VOC2012 [40] have yet to appear in the research community. However, some researchers working on medical imaging problems have adapted the capsule networks or created their own kind of capsules. There are two main cases in which capsules have started to be adapted to new problems other than recognizing handwritten digits. Additionally, there is one network that has adapted the idea of capsules into a new kind of layer meant for any kind of classification network.

A naïve and simple adaptation of the CapsNet was implemented by Afshar et al. in classifying brain tumor types between meningioma, glioma, and pituitary tumors [4]. The original CapsNet was slightly modified to better accommodate their use case. Overall, the architecture used was practically similar to the CapsNet presented by Sabour et al. [48]. The authors demonstrated that the new architecture shows promise in increasing accuracy in medical imaging problems.

In LaLonde and Bagci's object segmentation [36], the concept of deconvolutional capsules were introduced. Along with skip connections as implemented by He et al. in ResNet [22], LaLonde and Bagci were able to create a kind of CapsNet that was at least four times deeper than the original CapsNet. Compared to convolutional capsules that may downsample the image going from one capsule layer to another, deconvolutional capsules will upsample and eventually, with the help of shortcut connections, return an output that is the same size as the input images. These images would contain information for the CapsNet's classifier to successfully segment out pathological lungs from a CT scan. Additionally, the idea of locally-constrained capsules was introduced which allowed their CapsNet to take input images that were 512x512 and yet remain lightweight in trainable parameter count.

Most importantly, the HitNet architecture proposed by Delíege et al. [17] uses the idea of capsules as vector-based neurons to create a learned Hit-or-Miss (HoM) layer, placing it as the decoder for the feature maps from the convolutional layers. Their objective is that by checking each vector against a preset center of the 'target', the network will learn to represent the data in a better way. To elaborate, each vector in the HoM layer corresponds to a single class, and the closest HoM capsule vector to the 'target' is the predicted class. Figure 9 shows an illustration of the original HitNet architecture.

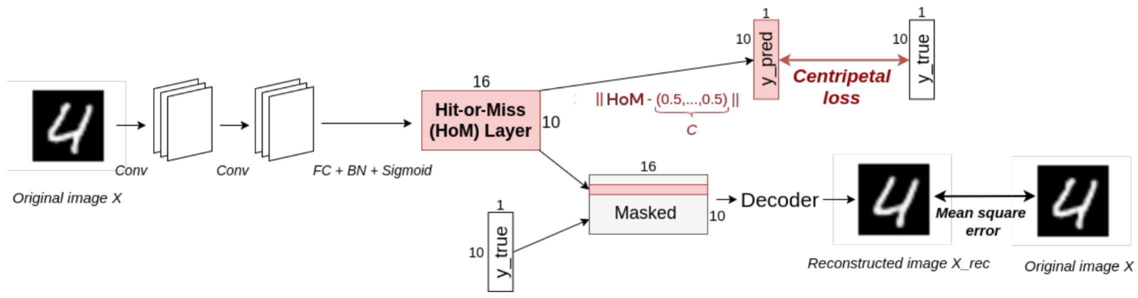


**Figure 9: Original HitNet Architecture [17]**

### 2.4.2 Hit-or-Miss Transforms and HitNet

While Delíege stated that they had not known about Hit-or-Miss Transforms [15, 19], it is worth discussing the image processing transform as another possible explanation for how their network's Hit-or-Miss capsule layer works. Since it compares with an arbitrary 'target' vector, namely $\mathbf{C} = [0.5, 0.5, 0.5, \ldots, 0.5]$, the weights between the final convolutional layer and the HoM layer are being modified such that the encoded feature maps conform to the set Hit-or-Miss pattern, or the 'target' vector. At the same time, the weights would be modified such that these feature maps would lack conformity to the other capsule vectors.

Each vector in the HoM layer corresponds to a distinct class in our dataset. Thus, by modifying the weights, it is possible to filter out unwanted patterns from the convolutional encoding layers. The resulting vector would be recognized as being close to the target since its pattern is recognized by the weights between the feature maps and the HoM layer. At the same time, all other vectors are trained to have vector lengths that are far from the center. That is, if a vector is meant to recognize a 1 from the MNIST dataset, then vectors that are supposed to recognize 2, 3, 4, 5, 6, 7, 8, 9, and 0 should have vector lengths that are much greater.

## 2.5 Deep Active Learning

Although many approaches to solving deep learning problems have involved creating new architectures, active learning is another way to improve a model by allowing the model to choose what data it wishes to train on. The key idea is that a machine learning algorithm can achieve greater accuracy when it is allowed to choose the data from which it learns [50]. A comprehensive review on active learning can be found in Settles's survey [50]. Chitta et al. [11] have proposed Deep Probabilistic Ensembles (DPE) that use regularized ensembles to approximate a Bayesian Neural Network. Although one of many active learning approaches, they demonstrate that such approaches can allow models to achieve competitive performance while requiring less training data. Empirically, Algorithm 2 shows that after the initial training cycle, the system injects mini-batches into the labeled dataset and trains the ensemble of models on the mini-batch. This process is repeated until we have reached $B$ total added samples to the set. The algorithm is performed for every model in the ensemble, where:

- $U = \{\mathbf{x}_u^j\}_{j=1}^{N_l}$ is a large **unlabeled set** consisting of $N_u$ samples, where each $\mathbf{x}^j \in X$ is an input data point.
- $L = \{(\mathbf{x}_l^j, y_l^j)\}_{j=1}^{N_l}$ is a **labeled set** consisting of $N_l$ labeled pairs, where each $\mathbf{x}^j \in X$ is a data point and each $y^j \in Y$ is its corresponding label. Thus, for a classification problem having $K$ labels, the label space $Y$ would be $\{1, \ldots, K\}$.
- $\mathcal{A} : X \rightarrow Y$ is an **annotator** that can be queried to map data points to labels.
- $\{\mathcal{M}_e : X \rightarrow Y\}_{e=1}^{E}$ is the set of models making up the **DPE**, where each is trained to estimate the label of a given data point.
- $\Omega$ is a single regularization term that is added into each model's weights and scaled by a factor $\beta$.

This particular active learning setup is commonly referred to as *batch mode active learning* [11], where every iteration involves 2 stages, as shown in Algorithm 2. In the first stage, the ensemble $\mathcal{M}^{(i)}$ is trained to convergence with the current labeled set $L^{(i)}$. In the second stage, the trained ensemble evaluates the existing unlabeled pool $U^{(i)}$ to select a subset of $b$ samples for labeling performed by the annotator $\mathcal{A}$. This subset is then moved from $U^{(i)}$ to $L^{(i+1)}$ before the next iteration. Training will proceed until the budget of $B$ total label queries is met.

To rank all of the available unlabeled data using the acquisition function $\alpha$, predictive entropy of the ensemble is used. This predictive entropy is given by

$$H_{ens} = -\left(\sum_{e\in E} \mathbf{p}^{(e)}\right)^T \log\left(\sum_{e\in E} \mathbf{p}^{(e)}\right) \tag{2.5}$$

where $\mathbf{p}$ refers to the normalized model prediction vector. Contrary to the above explanation, no annotators are needed in the experimental loop for this algorithm to proceed. Instead, it is sufficient to take a completely labeled dataset and hide a subset of the class labels from the model until this subset of 'unlabeled' data is transferred to the labeled dataset.

Additionally, as it pertains to this work, a simplified view of deep active learning can be found from Figure 10. In this work, the unlabeled dataset makes up 80% of the total training dataset, and the data budget is only limited by the size of the latter set.

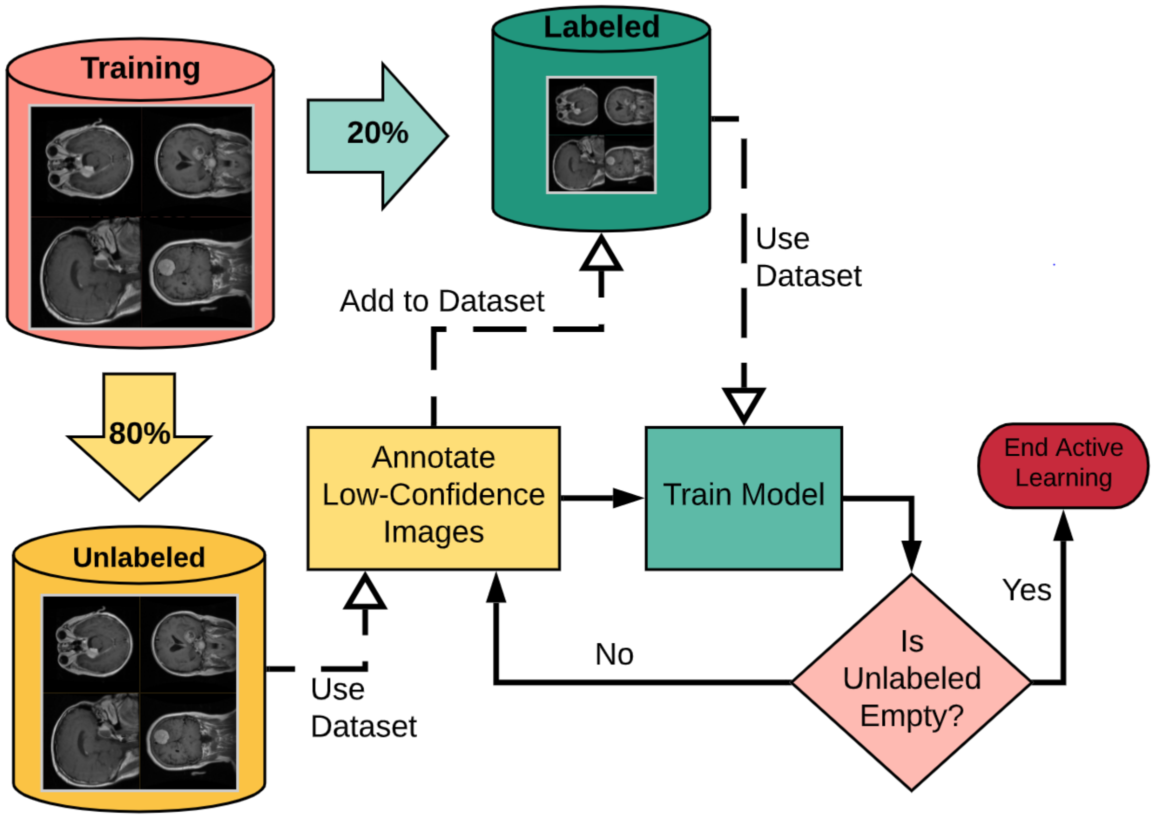


**Figure 10: Deep Active Learning for This Work**

**Algorithm 2** Active Learning with DPEs

```
$i \leftarrow 0$
Randomly sample $b$ points $\{\mathbf{x}^j\}_{j=1}^{b}$ from unlabeled set $U^{(0)}$
Annotate this data, $\{y^j = \mathcal{A}(\mathbf{x}^j)\}_{j=1}^{b}$
Move $\{(\mathbf{x}^j, y^j)\}_{j=1}^{b}$ to initial labeled set $L^{(0)}$
total_added_samples $\leftarrow b$
**while** total_added_samples $\leq B$ **do**
    **while** model_not_converged **do**                                    ▷ **stage (i)**
        Forward pass a mini-batch from $L^{(i)}$, $\hat{y} = \mathcal{M}^{(i)}(x)$
        Compute gradients $\delta$ of loss $l(y, \hat{y}) + \beta\Omega$
        Update DPE, $\mathcal{M}^{(i)} \leftarrow \mathcal{M}^{(i)} + \delta$
    **end while**
    **for** num_iteration_samples = 1 to $b$ **do**                         ▷ **stage (ii)**
        **for** $\mathbf{x} \in U^{(i)}$ **do**
            $\alpha(\mathbf{x}) \leftarrow H_{ens}(\mathbf{x})$
        **end for**
        $choice = \arg\max_{\mathbf{x}} \alpha(\mathbf{x})$
        $y^{choice} = \mathcal{A}(\mathbf{x}^{choice})$
        Move $(\mathbf{x}^{choice}, y^{choice})$ from $U^{(i)}$ to $L^{(i+1)}$
        total_added_samples += 1
    **end for**
    Update $b$
    $i$ += 1
**end while**
```

# Chapter 3
# RELATED WORK

In this chapter, research related to this work are briefly covered. Included in this chapter are machine learning algorithms, convolutional neural networks, capsule networks, and deep active learning techniques. Within each section, research that is more relevant to this work is discussed in more detail.

## 3.1 Machine Learning Techniques for Tumor Classification

Multiple brain tumor segmentation and classification solutions have been proposed using machine learning approaches. Abbadi et al. [1] have used gray level co-occurrence matrices (GLCM) to extract features and classified tumors based on those extractions using ANNs. Some other researchers have focused on large portions of or the entire pipeline of selecting abnormal MRI slices, segmenting, and classifying brain tumors. In particular, Deepak et al. [14] segment the sampled slice using K-means and classify the segmented partition using decision trees and a Support Vector Machine (SVM). Gupta et al. [21] went further and created a system that would automatically detect tumors in MRI slices by first identifying the tumors then segmenting the tumors in those slices. Their system used discrete wavelet transforms (DWT) and an SVM or random forest classifier. Another group of researchers led by Usman [54] processed multi-modal data by using DWTs and classified each slice's pixels with random forest classification, resulting in an image that would be segmented into five pixel classes. In another work [46], the researchers focused on comparing feature selection and extraction capabilities between Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) when applied to segment approximately 140 brain tumors in MRIs. They then classified pixels using an SVM. Also in the paper by Rathi et al. [46], they use an Artificial Network Fuzzy Inference System, which has also been used by Sharma and Mukharjee [52] for MRI brain tumor classification.

A couple of other notable approaches include the origin of the dataset used in this work, first presented by Cheng et al. [10]. They proposed three models that utilized intensity histograms, GLCMs, or a bag-of-words to classify tumor regions. Additionally, their approach attempted to enhance the performance of all three models by augmenting (i.e. enlarging) the region-of-interest, which was the tumor itself. Finally, Huang et al. [27] created a content-based image retrieval approach that kept an image database and used a mean average precision metric to calculate a sample image's similarity to a stored image. The closest similarity then determined the sample's given label.

## 3.2 CNNs for Tumor Classification

In general, there is a basic overview of advancements in deep learning written by LeCun, Bengio, and Hinton [37] that covers CNNs and recurrent neural networks, where the latter is commonly used in natural language processing. Other quick and general reviews on automated techniques for segmenting or classifying brain tumors in MRIs are those by Hani et al. [18] and by Vidyarthi and Mittal [55]. At least one paper calls out a weakness due to the pooling operation in CNNs, which in excess causes the network to lack the translational invariance that is desired in computer vision applications [6]. Additionally, Azulay and Weiss [6] demonstrate that modern CNNs are unable to be invariant to translations but that they can also mitigate such problems by introducing a decent amount of data augmentation.

Some approaches use CNNs and various architectural versions that specifically address the problem of classifying brain tumors from MRIs. Korolev et al. [32] briefly demonstrate the application of plain and residual CNNs on 3D brain MRIs for classifying between brains displaying Alzheimer's Disease, early and late mild cognitive impairment, or normal control. In considering CNNs for this problem, Abiwinanda et al. [3] create seven simple CNN models that are trained for approximately 10 epochs on the dataset provided by Cheng et al. [10]. Their validation accuracy results do not appear to improve past 85 – 90%. Balasooriya et al. [8] also created a CNN for MRI brain tumor classification. However, their training was conducted over close to 96k samples grabbed and combined from multiple different datasets. They applied their system to five different classes, with four of them tumorous and one of them a normal brain.

## 3.3 Capsule Network Approaches

Before the initial empirical results were created for the first capsule network (CapsNet [48]), Hinton published a paper on transforming autoencoders that hinted at the use of capsules to represent data [25]. Following the introduction of CapsNet by Sabour et al. [48], Hinton et al. published a paper on a different way of modifying routing connections between capsules by expectation-maximization [24]. Later in the same year, a group of researchers conducted tests using CapsNet on data sets with higher dimensionality [59], such as CIFAR10 [34]. They found CapsNets hard to train in higher dimensionality and attempted to find the best configuration for a CapsNet applied to high dimension problems. Wang and Liu [57] present the dynamic routing algorithm proposed by Sabour et al. [48] as an optimization problem for a number of heuristics then propose their own simple routing approach that is shown to outperform the original paper's results.

After the introduction of CapsNets, a few papers were written that propose sparse, unsupervised capsules for better generalization [47], CapsNets that use group theory to create equivariance of output pose vectors [38] or modifications that will let capsules scale for complex data [45]. Nguyen et al. also applied CapsNets to detecting images and videos that may have been forged by trying to solve inverse graphics problems. While many of the aforementioned CapsNets are geared toward computer vision, Bahadori applied CapsNets to features extracted to a one-dimensional linear subspace, which shows CapsNets have applications in natural language processing [7]. Additionally, Shahrodnejad et al. attempted to better explain how capsules work by considering the routing between capsule networks as a relevance path-by-agreement [51].

In reference to medical computer vision problems, Afshar et al. [4, 5] took the original CapsNet [48] and applied it to the brain tumors in the dataset given by Cheng et al. [10]. Without image masks, their first network achieved approximately 78% validation accuracy while their second proposed network achieved approximately 90% validation accuracy. LaLonde and Bagci [36] propose a network that makes use of shortcuts and is modeled after the well-known U-Net, traditionally used for many computer vision applications. Their network, which output a segmented image of CT lung scans, contained a deep network of convolutional and capsule layers that continuously downsampled the image then upsampled the image with the help of shortcut connections. It is important to note that LaLonde and Bagci's proposed network could be adapted for any kind of image segmentation. Additional applications to computer vision problems include segmenting breast histology images [28] and reconstructiong functional MRIs with CapsNets [44].

## 3.4 (Deep) Active Learning Techniques

A comprehensive review of active learning was written by Settles [50]. In the review, Settles covers the iterative and batch-mode active learning approaches and multiple types of query methods, such as uncertainty sampling or query-by-committee. Many of these approaches use a confusion metric to determine what samples will maximize information gain for a network. For the purposes of this thesis, the focus will be on batch-mode approaches.

Chitta et al. [11] worked on what is known as batch-mode active learning, where mini-batches are added from the unlabeled dataset to the labeled dataset up to a certain data budget. Other versions of batch-mode active learning have been used for medical image applications [26] or multiple classes in a large dataset [29, 30]. In the paper by Hoi et al. [26], the problem addressed was data correlation, and the proposed solution was a Fisher information matrix. Joshi et al. [29] proposed a margin-based uncertainty measure that was easy to compute so that the active learning could be efficiently scaled to a large dataset and a large number of classes.

# Chapter 4
# DEEP LEARNING MODEL DESIGN & IMPLEMENTATION

In this chapter, the design and implementation of three deep learning models is covered. In summary, the original HitNet architecture [17] will be covered and two additional architectures will be proposed. These models build off the original HitNet as a template in different ways that are described below. In particular, there is a basic HitNet, residual HitNet, and pyramid spatial pooling HitNet.

## 4.1 Original HitNet

In the original HitNet architecture [17], there are two convolutional layers followed by the Hit-or-Miss (HoM) layer. The HoM layer leads to a reconstruction sub-network or a prediction vector, seen at the top of Figure 11. In addition, only 28-by-28 grayscale images were input to their proposed architecture throughout the extent of their work. The HoM layer is made of ten classes with 16 values per vector. Each of the ten classes corresponds to one digit in the MNIST dataset, what Delíege et al. [17] originally used to validate their model.

The reconstruction sub-network takes as input the HoM layer's output and the true class label for the given sample. It uses these inputs to reconstruct the original image and checks the mean-squared error between the reconstructed image and the original. A low error value implies that this model and the following models have encoded the data well.

## 4.2 Baseline HitNet

For the baseline, there are three convolutional layers encoding the input. The specifications for each layer can be seen in Table 1. Additionally, the bottom of Figure 11 shows the proposed baseline architecture. This particular set-up was chosen

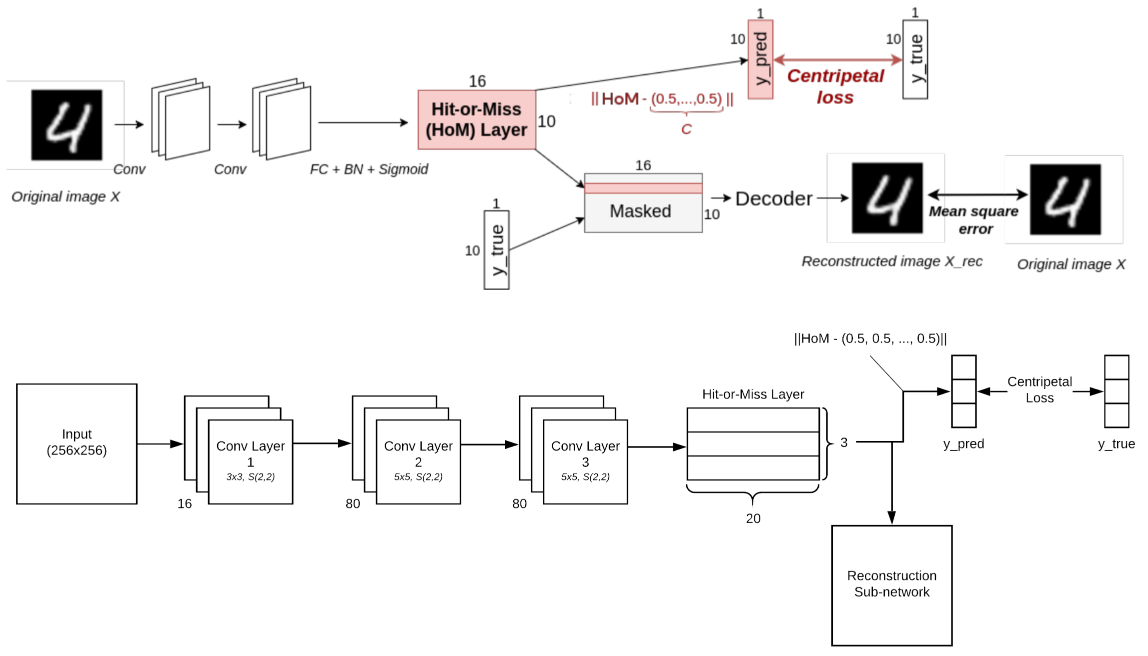


**Figure 11: Graphical Representation of HitNet [17] (top) and Baseline HitNet (bottom)**

to match the original HitNet as close as possible. As this network takes 256-by-256 images as input, the architecture contains another convolutional layer to further encode and reduce the image. Additionally, the baseline has only three classes, where each class vector comes with 20 values. The HoM vectors were arbitrarily given 20 elements for the baseline as well as the following two architectures.

For the baseline and following models, the reconstruction network has a total of 33,767,168 parameters. This is due largely in part to reconstructing a 256-by-256 image, which by itself is 65,536 pixels. Preceding the reconstructed image, there are two fully-connected layers of size 256 then 512 neurons. Thus, even with only the fully-connected layer of 512 neurons, the reconstruction sub-network would reach approximately 33.7 million parameters. The provided parameter numbers in Table 1 are the result of subtracting away the reconstruction sub-network from the full model, leaving us with the encoding convolutional layers and the HoM layer.

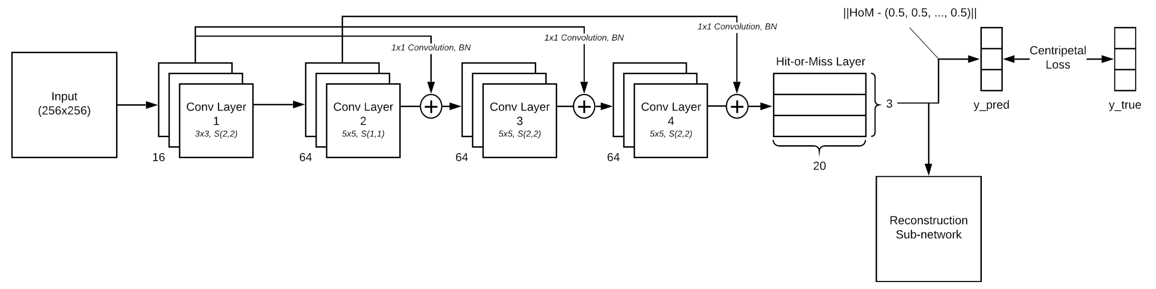


**Figure 12: Implementation of Res-HitNet.** For each shortcut, the result of the 1x1 convolution is normalized across the batch then added into the output of the given layer.

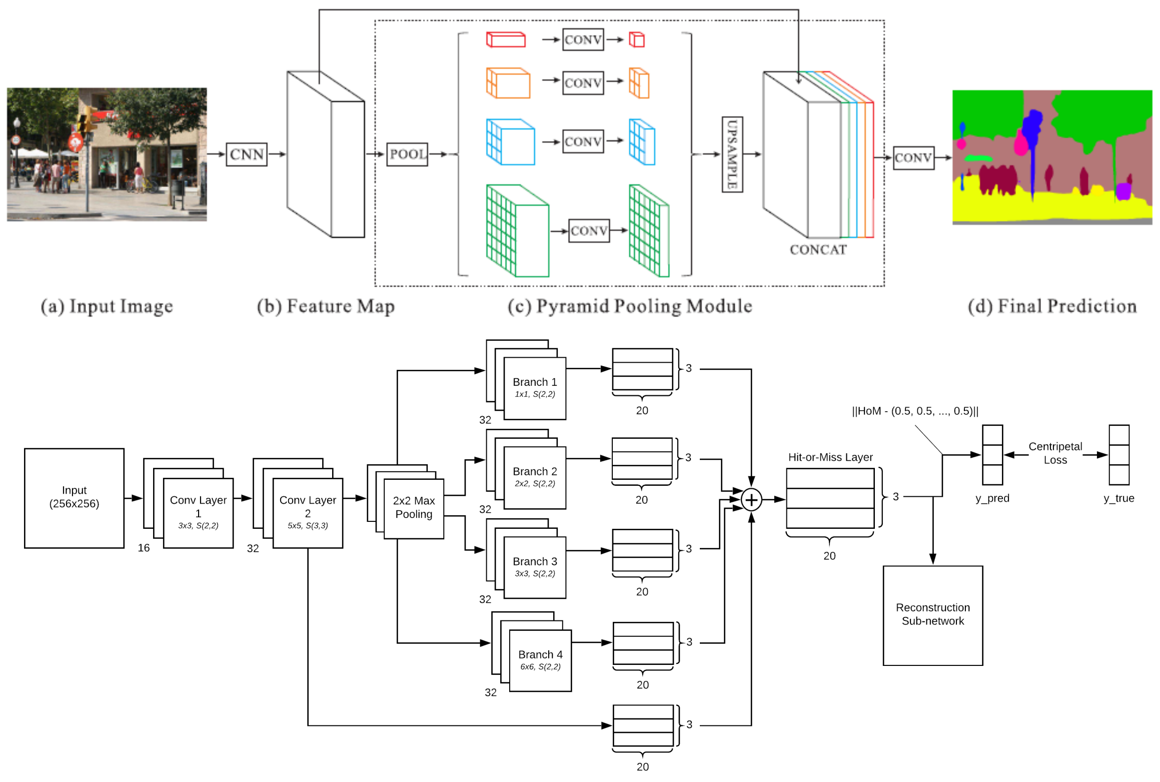


**Figure 13: Comparison of PSPNet [61] (top) and PSP-HitNet (bottom)**

**Table 1: Encoding Layers for Original HitNet [17] and Implemented Versions.** The filter specifications detail how to apply a convolution on the previous layer to get the listed layer (i.e. for the Baseline Conv. Layer 1, perform a 3x3 convolution with vertical and horizontal stride (2,2) on the input layer).

| **Model** | Layer | Feature Maps | Filter Spec. | # Parameters |
|---|---|---|---|---|
| HitNet [16, 17] | Conv. Layer 1 | 42 | 5x5 S(2,2) | — |
| | Conv. Layer 2 | 42 | 5x5 S(2,2) | |
| Baseline HitNet | Conv. Layer 1 | 16 | 3x3 S(2,2) | 4,190,844 |
| | Conv. Layer 2 | 80 | 5x5 S(2,2) | |
| | Conv. Layer 3 | 80 | 5x5 S(2,2) | |
| Residual HitNet | Conv. Layer 1 | 16 | 3x3 S(2,2) | 4,170,156 |
| | Conv. Layer 2 | 64 | 5x5 S(1,1) | |
| | Conv. Layer 3 | 64 | 5x5 S(2,2) | |
| | Conv. Layer 4 | 64 | 5x5 S(2,2) | |
| Pyramid Scene Parsing HitNet | Conv. Layer 1 | 16 | 3x3 S(2,2) | 4,358,040 |
| | Conv. Layer 2 | 32 | 5x5 S(1,1) | |
| | Max Pooling | 32 | 2x2 S(2,2) | |
| | Branch 1 | 32 | 1x1 S(2,2) | |
| | Branch 2 | 32 | 2x2 S(2,2) | |
| | Branch 3 | 32 | 3x3 S(2,2) | |
| | Branch 4 | 32 | 6x6 S(2,2) | |

### 4.3 Residual HitNet (Res-HitNet)

The residual version of HitNet makes use of shortcuts [22, 23] in the attempt to connect global information with higher layers. The main rationale for using this architecture is that both local and global information is desired when attempting to predict the correct tumor label for any given sample image. By using the shortcuts, the architecture can keep that global information and add it in along with local information. With this, the hope is that residual shortcuts will help the network achieve higher performance than the baseline. To get more variety in the shortcut connections, another convolutional layer was added before the HoM layer. More information about the network can be seen in Table 1 and Figure 12.

## 4.4 Pyramid Scene Parsing HitNet (PSP-HitNet)

PSP-HitNet was inspired by the semantic segmentation work done by Zhao et al. for parsing cityscape and room scenes [61]. A comparison between the original and the implemented versions can be seen in Figure 13. The original PSPNet architecture looks like an ensemble of four different networks, each using a different filter size for the convolution. The filter sizes are 1x1, 2x2, 3x3, and 6x6. This allows the model to acquire encodings for different sizes of objects as they appear in the image. In turn, the encodings will help the network classify each pixel based on an increasing area of influence, hopefully resulting in successful segmentation of all objects in an image. In addition to the four ensemble branches, a global prior connection brings information from the first convolutional layer to the ensemble branches' output.

While the original PSPNet upsampled from the ensemble of convolutional branches, the implemented PSP-HitNet does not upsample. The implemented version does not segment the image and does not need to classify each pixel like the original network. Thus, the outputs of all convolutions output reduced forms of their input. With exception of the first convolutional layer, all of the convolutional layers and branches connect to HoM layers specific for each branch. The final stage exists as a kind of voting scheme that adds together the results from the five separate HoM branches to a single HoM layer that will determine each sample's class label.

# Chapter 5
# RESEARCH VALIDATION

This section will cover the experiments performed to validate the models mentioned in Chapter 4 using only the data mentioned by Cheng et al. [10]. There are three main tests, where each one is conducted using a dataset-determined 5-fold cross-validation. That is, the five folds are given by the dataset in a provided MATLAB file.

The first test uses the original cross-validation dataset. Then, removing random samples from each fold, the models train on an equalized dataset where each tumor class has the same number of samples to train from for each epoch. The final test takes the best of all three models and trains it using active learning.

## 5.1 Tumor Dataset

The data used is first referenced by Cheng et al. [10] and can be found at [9]. The MRI slices are T1-weighted and contrast-enhanced covering three brain tumor classes. These slices were anonymized and acquired from 2005 to 2010 at Nanfang Hospital in Guangzhou, China and General Hospital at the Tianjing Medical University in China. In total, there were 233 patients, with the entire dataset containing 708 meningioma, 1426 glioma, and 930 pituitary tumor slices for a sum of 3064 images. The tumor borders were manually delineated by three experienced radiologists, and all tumor slices come with a provided mask [10]. The distribution of slices per fold can be seen in Table 2.

## 5.2 5-Fold Cross-Validation

In the first set of tests, each model was trained using K-fold cross-validation, where K=5. There was no data augmentation involved, and all folds were run for

**Table 2: Original Distribution of Tumor Slices in Dataset**

| Class | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Total |
|---|---|---|---|---|---|---|
| Meningioma | 112 | 167 | 139 | 124 | 166 | 708 |
| Glioma | 245 | 338 | 231 | 325 | 287 | 1426 |
| Pituitary | 185 | 174 | 202 | 179 | 190 | 930 |

30 epochs. For each fold, only the best weights were saved. The models were then modified by removing certain parts of their architectures to see what pieces could have the greatest or least impact on their performance.

In particular, for the baseline, each convolutional layer would be taken out one at a time while keeping the other two in the model. For Res-HitNet, shortcuts are the modified pieces, and for PSP-HitNet, each of the branches are considered pieces. Examples of one modification for all three models can be found in Figure 14.

## 5.3 Equalized 5-Fold Cross-Validation

In the next set of tests, the dataset is equalized across all tumor classes, as seen in Table 3. Based on the class with the least number of samples, the other two tumor classes have a random set of images removed from the folds such that the size of the respective set is equivalent. All three models are trained with 5-fold cross-validation again without any data augmentation for 30 epochs. Then, the best model was trained again with data augmentation on the equalized dataset. For each sample, the data augmentation was allowed a rotation range of 10 degrees, horizontal and vertical flip, and width and height shifts of 0.1. Any pixels that needed filling were done so with a zero-value.

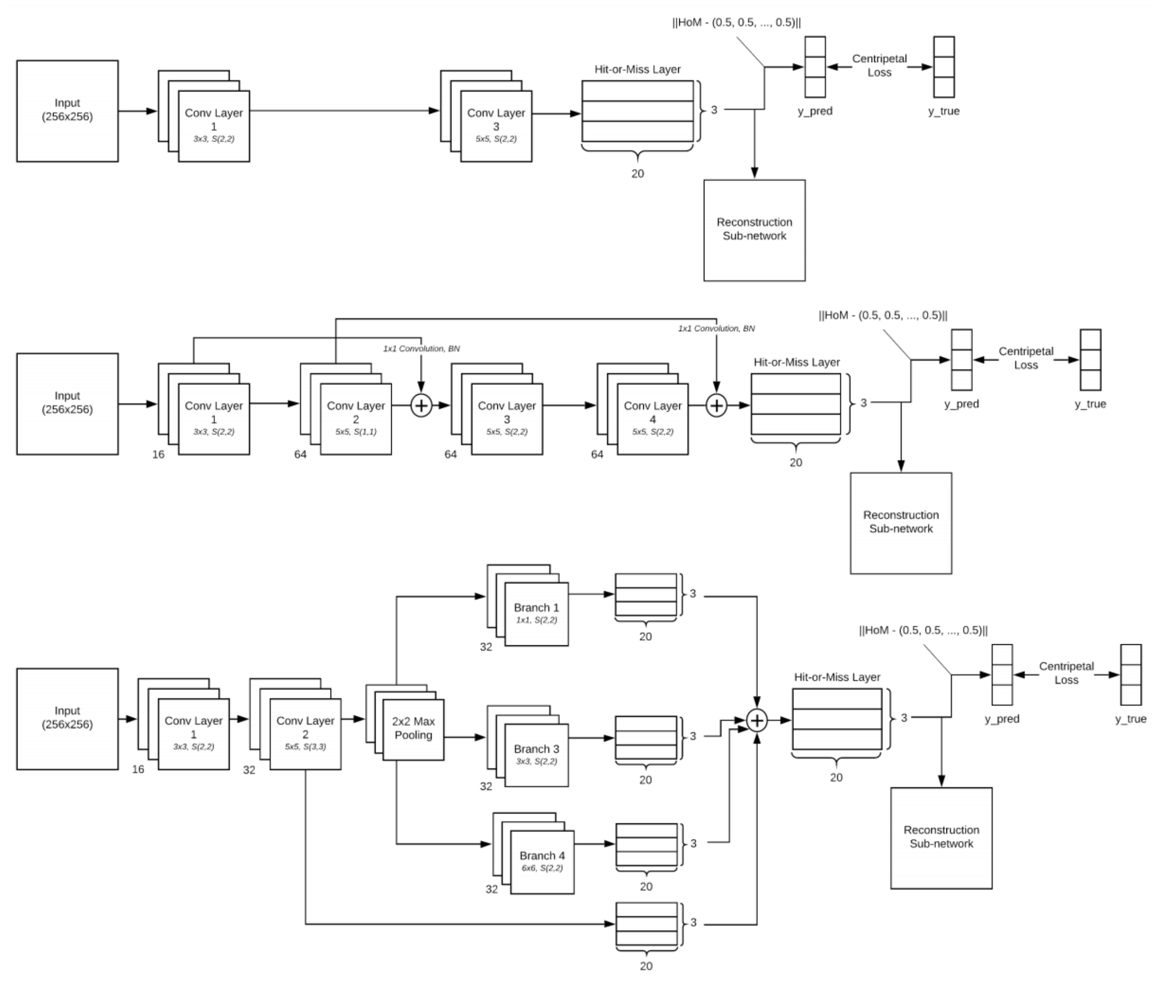


**Figure 14: Example of HitNet Architecture Modifications.** From top to bottom, the baseline has the second convolutional layer removed, Res-HitNet does not have a second shortcut, and PSP-HitNet does not have the branch 2x2 convolutions.

**Table 3: Equalized Distribution of Tumor Slices in Dataset**

| Class | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Total |
|---|---|---|---|---|---|---|
| Meningioma | 112 | 167 | 139 | 124 | 166 | 708 |
| Glioma | 112 | 167 | 139 | 124 | 166 | 708 |
| Pituitary | 112 | 167 | 139 | 124 | 166 | 708 |

## 5.4 Deep Active Learning Cross-Validation

In the final set of tests, the network with best validation performance in the previous two cross-validation tests is trained again with the 5-fold cross-validation for 40 epochs. Then, the network is trained using active learning on each fold. Two different active learning tests were conducted using 80% of the training data in the equalized dataset.

In the first test, the network is trained for approximately five or six active learning iterations. Each iteration moves about 20% of the unlabeled data to the labeled data that is used for training the network. Each portion of unlabeled data is pulled either with the uncertainty estimate or through random choosing. Given the flooring of values when dividing the unlabeled data by five, there can be a sixth iteration that moves the rest of the samples from the unlabeled to the labeled.

In the second test, the network undergoes active learning where the system annotates either the 100 lowest-confidence, 100 highest-confidence, or 100 random samples from the unlabeled dataset. Thus, this particular test may have the network train from scratch for between 13 and 16 active learning iterations. For the purposes of this thesis, the model's predictive confidence indirectly relates to uncertainty. That is, a prediction with high confidence has low uncertainty and vice versa.

When pulling the unlabeled data from the training data, the folds are weighted depending on how many samples they contribute to the entire equalized training dataset. For instance, if the model is training its fifth fold, the training set takes from folds one to four. The second fold has the most samples and will have the most labeled images moved to the unlabeled dataset. In the test for 20% unlabeled data, Table 4 shows how the sample counts are decided.

In work by Chitta et al. [11], the active learning test is conducted for a maximum of 400 epochs with an early stopping patience parameter of 25 epochs without im-

**Table 4: Image Split Per Class for 20% Unlabeled for the Fifth Fold.** All results from floating-point operations are floored to get integer numbers for images per fold and per class.

| | **Fold 1** | **Fold 2** | **Fold 3** | **Fold 4** | Total |
|---|---|---|---|---|---|
| Equalized Total | 336 | 501 | 417 | 372 | 1626 |
| Fraction of Training | 0.2066 | 0.3081 | 0.2565 | 0.2288 | 1.0 |
| Maximum Images per Fold | 268 | 400 | 333 | 297 | 1298 |
| Maximum Images per Class | 89 | 133 | 111 | 99 | 432 |

provement in validation accuracy. They use the stochastic gradient descent (SGD) optimizer with a learning rate of 0.1 and momentum of 0.9. All active learning tests run for the 80% unlabeled case used the SGD optimizer with the same learning rate and momentum. In the DPE proposed by Chitta et al. , a custom regularization term was calculated, but this work proposes using a single model for the active learning. Thus, the active learning phase uses an L2-regularization of 0.00001 on all convolutional layers of the best network. However, compared to the paper, where the learning rate is decreased and the training is continued, this work's model will only be trained a maximum of 200 epochs and a patience of 25 when there is not at least an improvement of 0.01% validation accuracy. There was no decrease in learning rate as done by Chitta et al. [11]. For a baseline comparison, the best model was trained on the entire equalized training data for a maximum of 200 epochs. As will be seen in the following chapter, the best cross-validated model did not improve its validation accuracy after approximately 165 epochs. Thus, when conducting the active learning for 80% unlabeled, the model is trained for a maximum of 200 epochs.

$$H = -\left(1 - \frac{\mathbf{p}}{||\mathbf{p}||}\right)^T \log_{10}\left(1 - \frac{\mathbf{p}}{||\mathbf{p}||}\right) \tag{5.1}$$

To calculate a prediction's uncertainty, Equation 5.1 is used. $\mathbf{p}$ stands for the prediction vector for a single label coming out of the derived and proposed HitNet models. Due to the vector values representing capsule lengths in reference to the center, the smallest value corresponds to the predicted label for an input sample. Additionally, the vector lengths may exceed 1.0. Thus, the prediction vectors are length-normalized and subtracted from one. In general, this calculation should result in lower confidence – a high $H$ – when two or three prediction values are similar. By contrast, if the prediction vector contains two similar large values and one small value, higher confidence – a low $H$ – should result.

# Chapter 6
# RESULTS & DISCUSSION

Below, results are presented for all tests mentioned in Chapter 5. In addition to the validation accuracy, precision, recall, and F1-Score are given for each class. Precision is used for checking how often the model gives false positives, while recall will check how often the model gives false negatives. A high F1-Score means that the model has both a low number of false positives and false negatives. Thus, it is a value to maximize when looking at the results. The equations for precision, recall, and F1-Score are given below. TP, FP, TN, FN stand for true positive, false positive, true negative, and false negative, respectively.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$\text{F1} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

## 6.1 Cross-Validation Results of Proposed Models

Using 5-Fold cross-validation with the regular training dataset and the equalized dataset, results for all three proposed models are presented, and their performances are compared. Additionally, results are given for when each model's architecture has parts removed one-at-a-time. From the mentioned results, a best model is chosen for deep active learning testing, where its performance is further detailed in Section 6.2.

### 6.1.1 5-Fold Cross-Validation

After the first cross-validation test on the original dataset without equalizing the tumor classes, validation accuracy reached a maximum average validation accuracy

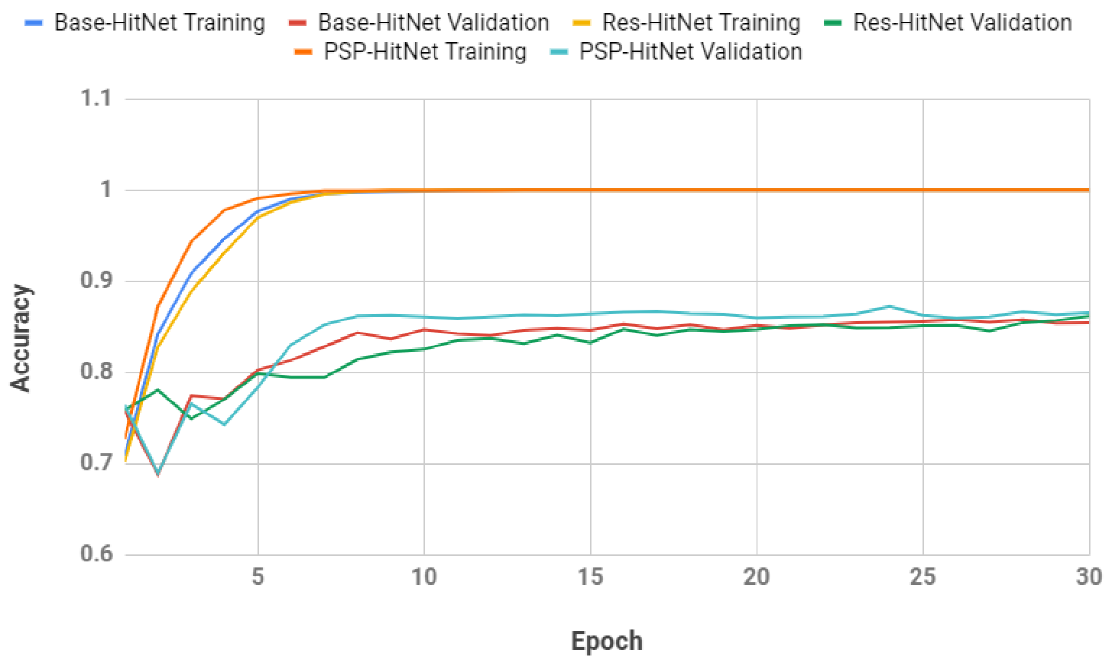


**Figure 15: Training and Validation Accuracy Results for all models using the original dataset.** Training results are those curves that converge to an accuracy value of 1.

of 85.79%, 86.18%, and 87.23% for Base-HitNet, Res-HitNet, and PSP-HitNet, respectively. The precision, recall, and F1-Score for all three are presented in Table 5. As can be seen, PSP-HitNet had the best F1-Score for classifying meningioma and glioma tumors and held comparable performance for pituitary tumors versus Res-HitNet. PSP-HitNet was also the quickest model to converge during the 30 epochs of training, as shown in Figure 15.

### 6.1.2 Architecture Modifications

When removing the convolutional layers one at a time for Base-HitNet, the resulting performance was similar to the unmodified Base-HitNet. Specifically, the average validation accuracy results for the Base-HitNet with layer 1 removed and with layer 2 or 3 removed were 84.59% and 84.52%, respectively. Additional results are not necessary for removing layer 3 as both layer 2 and layer 3 have the same filter specifications, seen in Table 1 of Chapter 4. These results are close to the unmodified Base-HitNet result of 85.79%.

For Res-HitNet, the removal of shortcut connections had greater influence on the overall performance of Res-HitNet's original model. The removal of the second or third shortcut connection resulted in an overall decrease of approximately 2% from the original's results. Given the more dramatic drop in validation accuracy, shortcut connections that came into play later in the network appeared to have a greater impact on the model's performance.

Similar to the Base-HitNet results, modifications made to PSP-HitNet had little to no impact. Perhaps due to the random weight initialization, some of the modified architectures performed slightly better than the original PSP-HitNet. This is the case with removing the branch containing 3x3 convolutions, which achieved an average validation accuracy of 86.63% versus the original's 86.55%. Otherwise, the results show no sign of there being a decrease in performance.

A side-by-side comparison between this models' architecture modifications can be found in Table 6. Additionally, each architecture's training results can be seen in Figures 16, 17, and 18.

#### 6.1.3 Equalized 5-Fold Cross-Validation

When equalizing the dataset, Base-HitNet, Res-HitNet, and PSP-HitNet achieved a maximum validation accuracy of 85.57%, 84.70%, and 87.04%, respectively. From Figure 19, it can be seen that all models had similar convergence rates but PSP-HitNet came out with better validation results than the other two tested models. In Table 7, it can be seen that equalizing the dataset allowed each model to improve their F1-Score for classifying meningioma tumors while trading off decreases in the F1-Scores for classifying glioma and pituitary tumors. The results in Table 7 also show that PSP-HitNet had better overall precision, recall, and F1-Score when trained using the

**Table 5: Average Precision, Recall, and F1-Score for All Models after Training on the Original Dataset for 30 Epochs.** Bolded entries signify the best performance for precision, recall, or F1-Score.

| Class | HitNet Model | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Meningioma | Base | 75.421 | 74.448 | 74.929 |
| | Res | **76.822** | 71.916 | 74.288 |
| | PSP | 76.412 | **77.178** | **76.793** |
| Glioma | Base | 89.008 | 87.454 | 88.224 |
| | Res | 88.108 | **88.335** | 88.221 |
| | PSP | **90.461** | 87.775 | **89.098** |
| Pituitary | Base | 90.544 | 93.591 | 92.043 |
| | Res | **92.908** | 94.636 | **93.764** |
| | PSP | 91.935 | **95.293** | 93.584 |

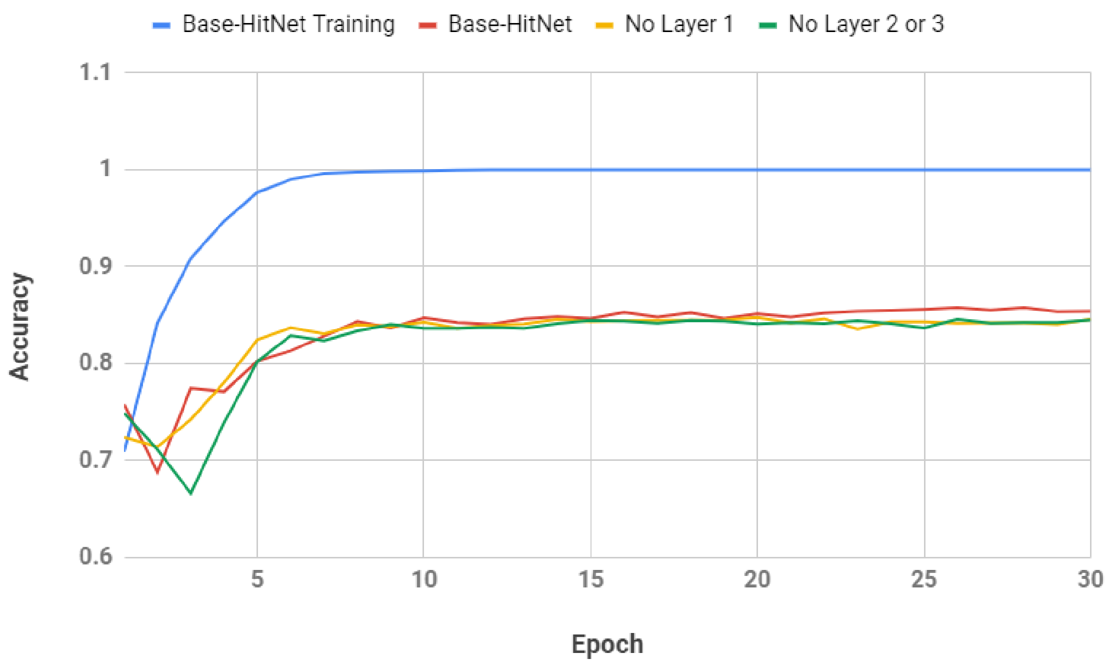


**Figure 16: Results for Base-HitNet when Removing Pieces from the Architecture.** The model's original results are shown by the red line.

**Table 6: Validation Accuracy Results for All Modifications to Base-HitNet, Res-HitNet, and PSP-HitNet.** To observe how the change in parameters might have affected the average validation accuracy for the networks, the number of parameters were included as well. All of the parameter counts are completed without counting the reconstruction sub-network.

| Model Type | Modification | # Parameters | Validation Accuracy |
|---|---|---|---|
| Base-HitNet | Original | 4,190,844 | 85.42 |
| | No Layer 1 | 18,023,100 | 84.59 |
| | No Layer 2 or 3 | 18,483,580 | 84.52 |
| Res-HitNet | Original | 4,170,156 | 86.18 |
| | No Shortcut 1 | 4,168,812 | 85.14 |
| | No Shortcut 2 | 4,168,812 | 83.99 |
| | No Shortcut 3 | 4,165,740 | 83.92 |
| PSP-HitNet | Original | 4,358,040 | 86.55 |
| | No 1x1 | 4,124,524 | 85.65 |
| | No 2x2 | 4,161,772 | 86.22 |
| | No 3x3 | 4,156,652 | 86.63 |
| | No 6x6 | 4,198,124 | 85.41 |
| | No Res | 807,820 | 86.24 |

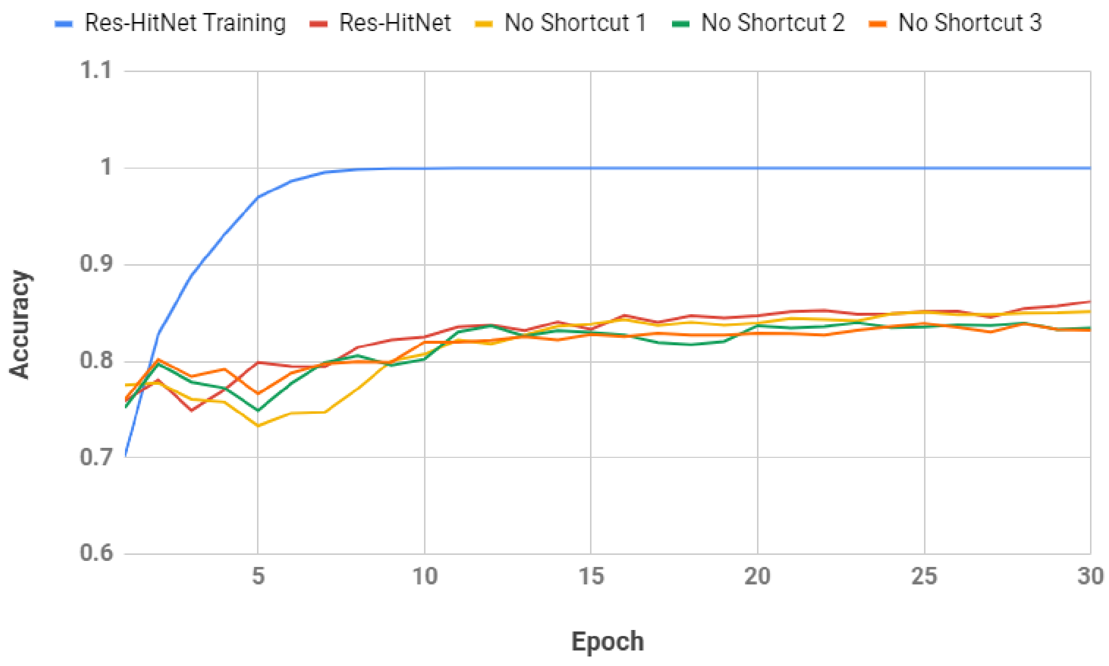


**Figure 17: Results for Res-HitNet when Removing Pieces from the Architecture.** The model's original results are shown by the red line.

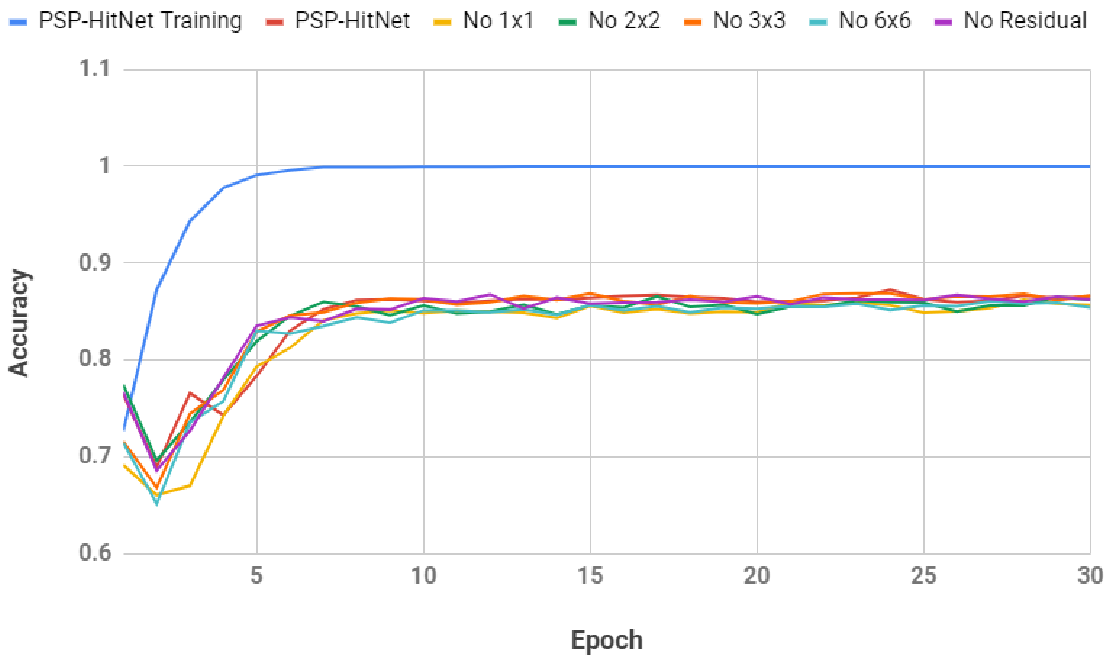


**Figure 18: Results for PSP-HitNet when Removing Pieces from the Architecture.** The model's original results are shown by the red line.

**Table 7: Average Precision, Recall, and F1-Score for All Models after Training on the Equalized Dataset for 30 Epochs.** Bolded entries signify the best performance for precision, recall, and F1-Score.

| Class | HitNet-Model | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Meningioma | Base | **83.307** | 81.304 | 82.293 |
| | Res | 79.667 | 81.988 | 80.811 |
| | PSP | 81.164 | **86.555** | **83.773** |
| Glioma | Base | 86.136 | 82.439 | 84.247 |
| | Res | 86.753 | 80.433 | 83.474 |
| | PSP | **90.455** | **83.066** | **86.603** |
| Pituitary | Base | 90.428 | **94.828** | 92.576 |
| | Res | 91.235 | 93.597 | 92.401 |
| | PSP | **92.482** | 93.179 | **92.829** |

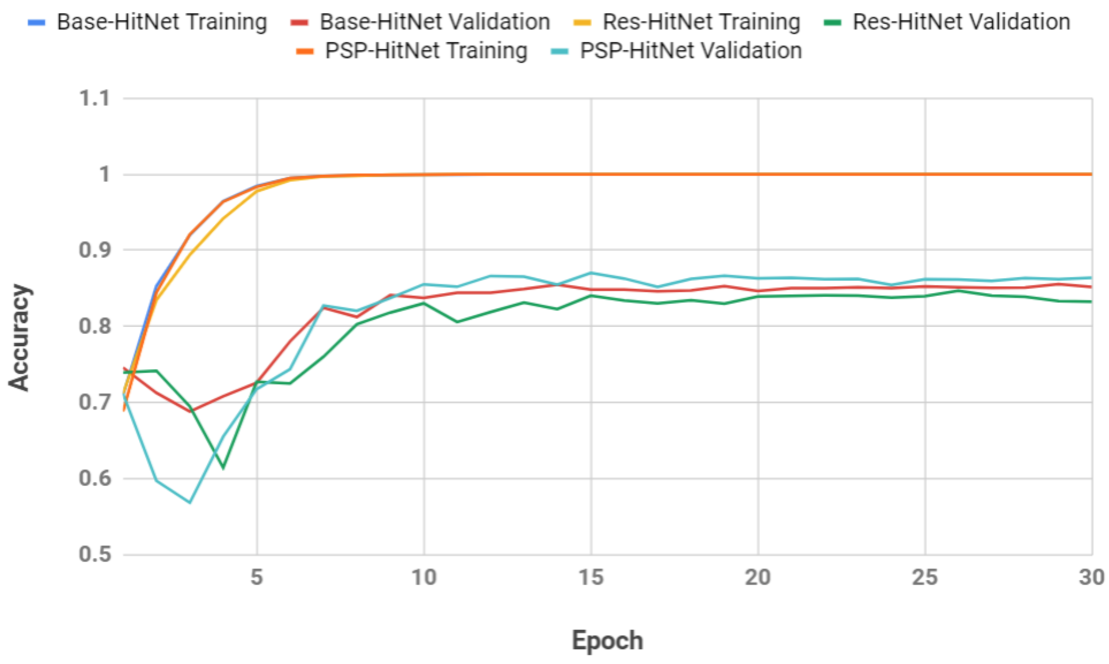


**Figure 19: Training and Validation Accuracy Results for All Models Using the Equalized Dataset.**

**Table 8: Average Precision, Recall, and F1-Score for PSP-HitNet after Training on the Data-Augmented Equalized Dataset for a Maximum of 400 Epochs.** Due to early stopping based on the validation accuracy, none of the training per fold exceeded 165 epochs.

| Class | Precision | Recall | F1-Score |
|---|---|---|---|
| Meningioma | 87.761 | 90.447 | 89.084 |
| Glioma | 95.988 | 87.026 | 91.288 |
| Pituitary | 92.600 | 97.943 | 95.197 |

equalized dataset. From the validation accuracy, precision, recall, and F1-Score, it was determined that PSP-HitNet was the best model of the three and was chosen for moving forward with active learning. From this point in the work onward, PSP-HitNet will be the only 'model' being trained and validated with additional tests.

After training the model for a maximum of 400 epochs using data augmentation on the equalized dataset, it was discovered that training on all folds was stopped early by the $165^{th}$ epoch and resulted in an average validation accuracy of between 89.3% and 93%. Results for training PSP-HitNet for this extended period of time can be seen in Figure 20. As four of five folds did not train past 102 epochs, data is truncated once the average accuracy contains only one fold in the calculation. In particular, the yellow line in the graph represents the average across all five folds while red and blue represent the averages across three and two folds, respectively. The maximum average validation accuracies were 89.3%, 92.71%, and 93%. The yellow line ran for 56 epochs, the red for 94 epochs, and the blue for 102 epochs. After 102 epochs of training, only training for the fourth fold continued until epoch 165. Precision, recall, and F1-Score values can be seen in Table 8.

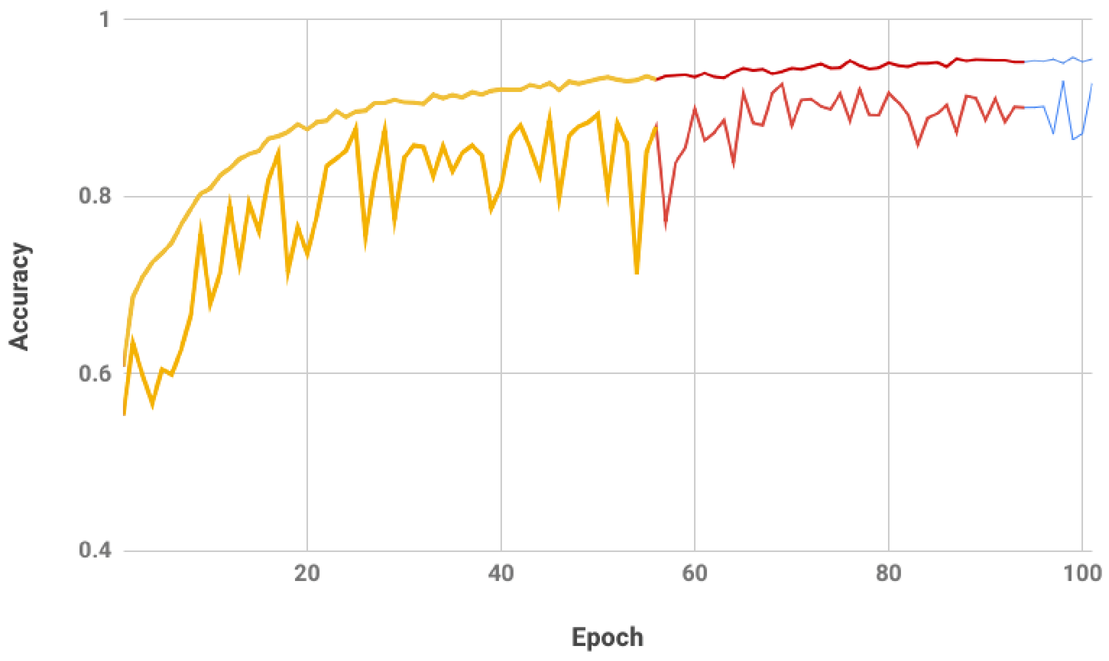


**Figure 20: Results for PSP-HitNet after Training for 101 Epochs with Data Augmentation.** Yellow contains a 5-fold average, Red a 3-fold average, and Blue a 2-fold average. Respectively, their maximum average validation accuracies were 89.3%, 92.71%, and 93%.

**Table 9: Average Validation Accuracy per Active Learning Iteration Using SGD Optimizer without Regularization.** The zeroth iteration represents results after training on the starting labeled data.

| Iteration | 80% Uncertainty | 80% Random |
|---|---|---|
| 0 | 77.063 | 76.926 |
| 1 | 87.558 | 82.472 |
| 2 | 89.653 | 88.334 |
| 3 | 90.720 | 88.756 |
| 4 | 89.927 | 89.817 |
| 5 | 91.003 | 90.471 |

## 6.2 Applying Deep Active Learning to Best Model

The best model, PSP-HitNet, was then run using the batch-mode active learning method. In particular, for each deep active learning iteration, the model was trained for a maximum of 200 epochs, stopping early if the validation accuracy did not improve after 25 consecutive epochs. When comparing the uncertainty estimation with the random choosing, there is a marginal performance improvement. By pulling in the lowest-confidence samples from the unlabeled dataset, the network improves faster than if it chose random samples. Results for no regularization and for regularization can be seen in Tables 9 and 10 and in Figures 21 and 22. Note that although the two querying strategies begin at the same general validation accuracy for each starting point, the uncertainty querying strategy generally trains the model faster than the random strategy.

The differences between querying strategies are demonstrated when the model is configured to pick either 100 lowest-confidence, 100 random, or 100 highest-confidence samples each iteration from the unlabeled dataset. Since picking 100 samples per iteration involves a minimum of 13 iterations, approximately 6% of the complete training set is annotated at a time. Both Table 11 and Figure 24 display results for

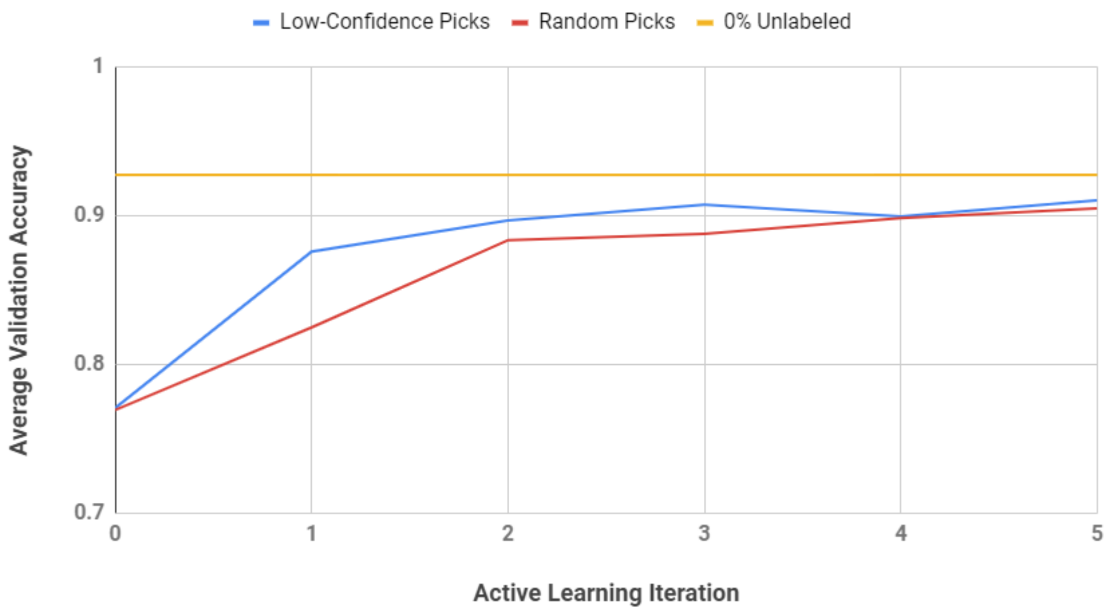


**Figure 21: Active Learning Results for Both Random and Uncertainty-Estimated Queries Using SGD without Regularization**

**Table 10: Average Validation Accuracy per Active Learning Iteration Using SGD Optimizer with L2-Regularization.**

| Iteration | 80% Uncertainty | 80% Random |
|---|---|---|
| 0 | 74.503 | 74.793 |
| 1 | 86.670 | 82.302 |
| 2 | 87.027 | 87.993 |
| 3 | 89.512 | 90.051 |
| 4 | 90.901 | 90.048 |
| 5 | 90.597 | 89.686 |

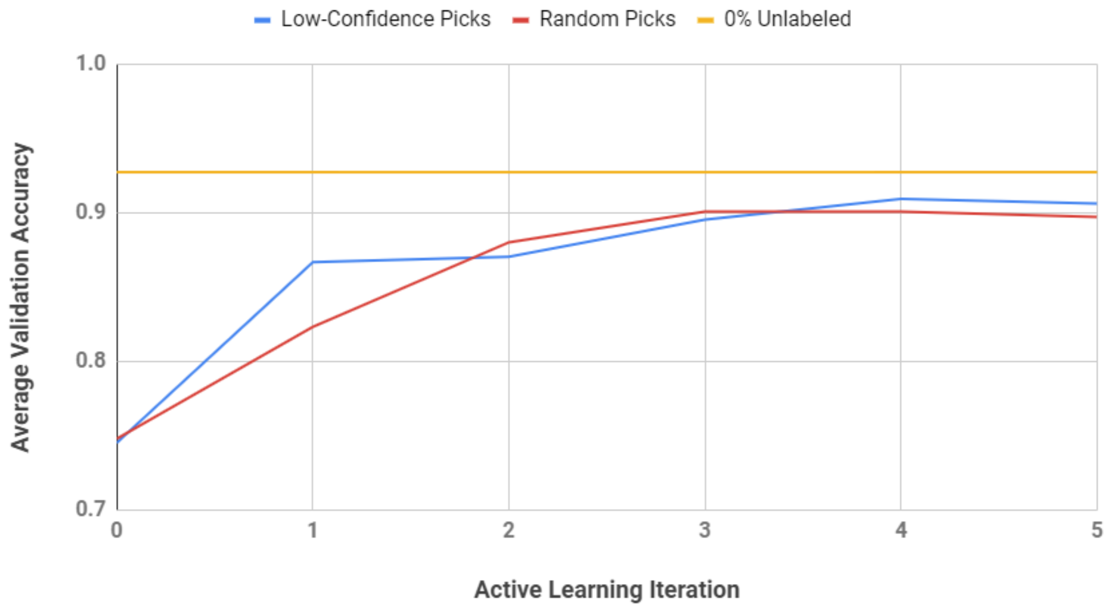


**Figure 22: Active Learning Results for Both Random and Uncertainty-Estimated Queries Using SGD with L2-Regularization**

an average of two runs per querying strategy. It can be seen in Table 11 and Figure 24 that the first additional 12 – 24% of training data shows quicker improvement by picking the 100 lowest-confidence samples per iteration.

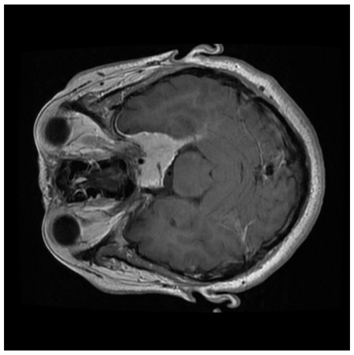
In the first active learning iteration, picking the 100 lowest-confidence samples showed less improvement than the other strategies, perhaps due to random initialization that occurs when the model is trained from scratch for each active learning iteration. In addition, due to the SGD optimization algorithm, some active learning iterations show querying strategies where the model may have become stuck in a local minimum. However, during iterations two, three, and four, annotating the samples with the lowest confidence leads to better performance compared to either picking random samples or picking samples with highest confidence. This strategy led to bringing in the most information for PSP-HitNet to train. As the training set is filled with images that bring in less information for the model, picking low-confidence images converges to an average validation accuracy of approximately 90%. Broadly

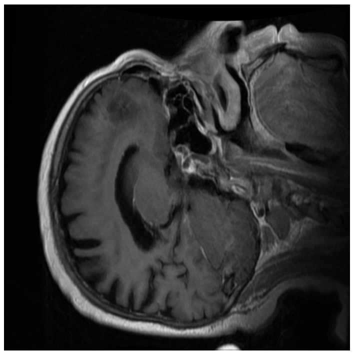
**Table 11: Results for 100 Highest-Confidence, 100 Random Choose, and 100 Lowest-Confidence per Active Learning Iteration.** The results presented are averaged across two runs for each strategy.

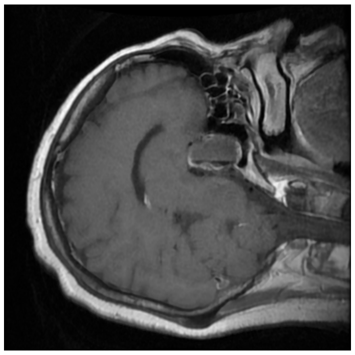
| Iteration | High Confidence Picks | Random Picks | Low Confidence Picks |
|---|---|---|---|
| 0 | 75.021 | 75.458 | 75.573 |
| 1 | 80.028 | 82.595 | 78.999 |
| 2 | 81.124 | 80.722 | 87.258 |
| 3 | 83.064 | 82.392 | 83.589 |
| 4 | 81.474 | 86.634 | 88.615 |
| 5 | 84.884 | 88.891 | 89.075 |
| 6 | 85.982 | 89.115 | 88.845 |
| 7 | 84.066 | 88.479 | 89.797 |
| 8 | 87.958 | 88.717 | 89.863 |
| 9 | 88.343 | 90.529 | 90.201 |
| 10 | 89.212 | 90.406 | 89.844 |
| 11 | 90.068 | 90.769 | 90.667 |
| 12 | 90.819 | 90.026 | 90.391 |
| 13 | 90.291 | 91.079 | 90.639 |

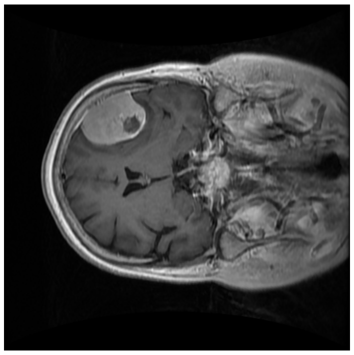
speaking, as the training set is filled with more annotated samples, all three querying strategies converge to the same approximate average validation accuracy.

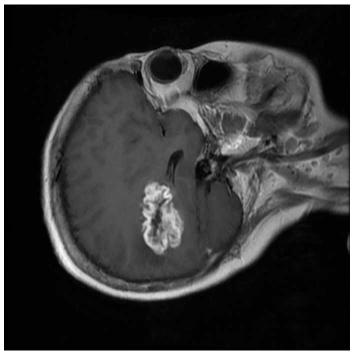
Lastly, Figure 23 shows a sample of some high-confidence and low-confidence samples. These samples were determined by the model when it is first trained from scratch using Folds two to five for training and Fold one for validation.

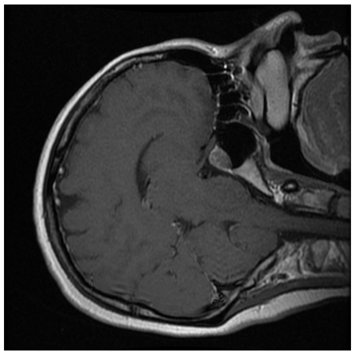

(a) (M, P) - 0.27346858

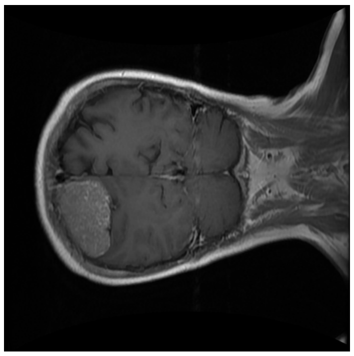

(b) (G, P) - 0.27386144

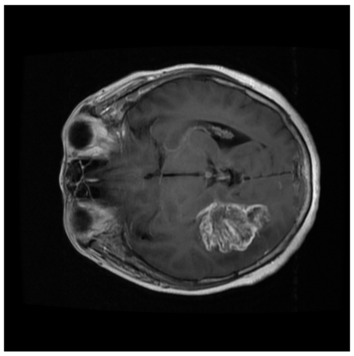

(c) (P, P) - 0.26106316


(d) (M, M) - 0.36038724


(e) (G, M) - 0.3601587


(f) (P, P) - 0.3603682


(g) (M, P) - 0.47303647


(h) (G, M) - 0.47384858

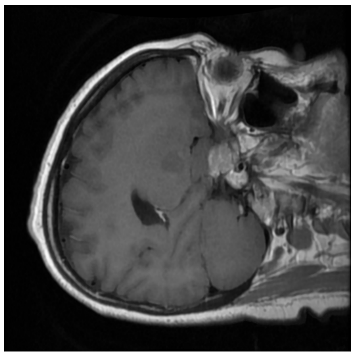

(i) (P, P) - 0.47279787

**Figure 23: Resulting Predictions for MRI Scans and Corresponding Uncertainty Calculations.** Pictured MRI scans were determined after training with 20% unlabeled using Fold 1 as the validation set. The letters in the parentheses are true label and predicted label, respectively. They are followed by their calculated uncertainty value. A higher uncertainty value implies low-confidence for the model's prediction while a lower value implies high-confidence.

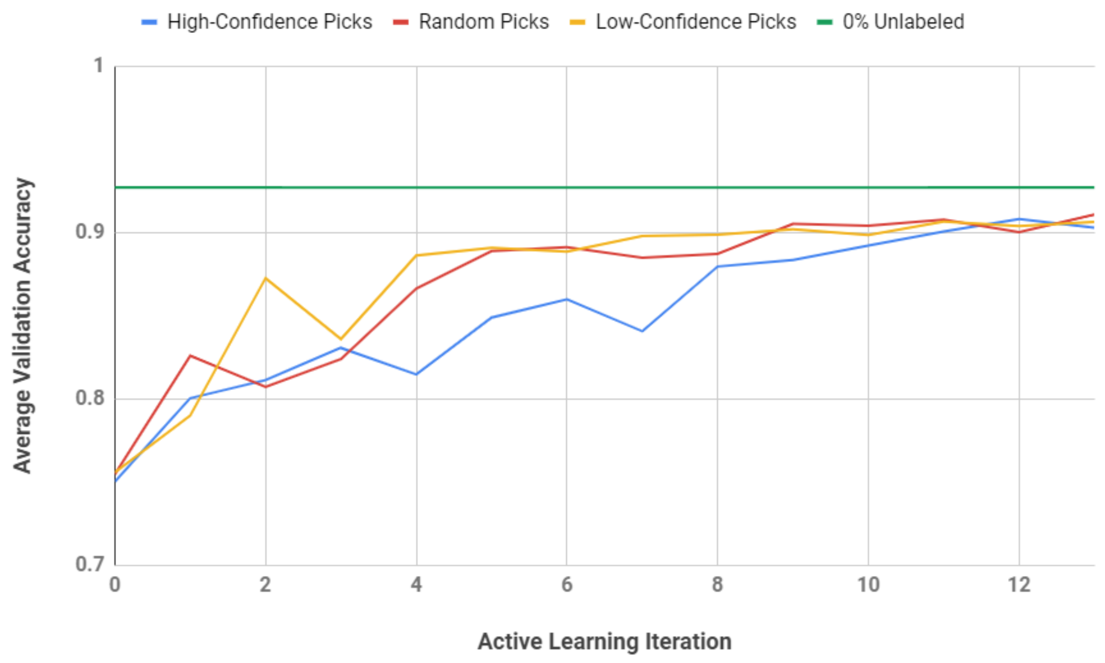


**Figure 24: Active Learning Results for 100 Highest-Confidence, 100 Lowest-Confidence, and Random Sampling.** The results presented are averaged across two runs for each strategy.

# Chapter 7
# FUTURE WORK

To extend this work and improve upon the presented active learning technique, a couple different paths are possible. First, in reference to work by Hoi et al. [26], medical images may have information overlap between each other, making it difficult to choose a subset of unlabeled data based on uncertainty estimates. The researchers propose a Fisher information matrix and an efficient way of reducing the matrix to pick out the most informative examples from the unlabeled data.

A second approach would be to extract some features from the MRI slices using wavelet transforms [13, 60]. Wavelet transforms have also been applied to ultrasound images to diagnose cancer [53]. Either discrete or stationary wavelets could help filter out unnecessary information in the MRIs, allowing the network to focus on the brain tumor features that might remain. Some introductions and studies can be found by Walker [56] and Schmeelk [49].

In reference to deep active learning, another appraoch would be to change the annotated set size to smaller, more fine-grained portions to better understand the behavior of the model when adding each sample. For example, instead of annotating 100 samples from the unlabeled dataset to the labeled dataset, the system would annotate 50, 25, 10, or as little as 5 samples per each deep active learning iteration.

A final proposed extension would be to re-implement the segmentation portion of the PSP module in PSP-HitNet. This would allow the model to not only classify between tumors but also segment exactly where the tumor is in the brain. In turn, the model would become more powerful than it currently is.

# Chapter 8
# CONCLUSION

A quick diagnosis of brain tumors using MRI scans can help increase the survival chances for patients. Some researchers use trained machine learning models as second opinions to aid in the diagnosis. These techniques can include augmenting the tumor region-of-interest [10], capsule networks [4, 5], and CNNs [3, 8]. In this work, variants of HitNet [17] have been proposed and trained on a small dataset of brain MRI scans containing meningioma, glioma, and pituitary tumors. Through tests on the original and equalized datasets, it was found that PSP-HitNet had the best performance compared to Base-HitNet and Res-HitNet, gaining a maximum validation accuracy of approximately 87% and improving by at most 2% over the Base-HitNet. Training the best model with data augmentation and for a maximum of 400 epochs revealed that PSP-HitNet reached at most 93% validation accuracy for any single training fold.

While modifying and making new deep learning architectures is one way of increasing the overall validation and test accuracy, other techniques like deep active learning work with the dataset to increase the validation accuracy. It is not always possible to acquire large amounts of data, and deep active learning helps train models faster with less data. This work has proposed using PSP-HitNet with deep active learning that allowed it to train the fastest by choosing the lowest-confidence, or highest-uncertainty, samples. In using this technique, the model was able to achieve better performance than randomly choosing within the first 12-24% of annotated unlabeled data. Even with a small dataset, the network was able to quickly achieve performance by annotating the lowest-confidence unlabeled samples and adding them into the master training dataset.

# BIBLIOGRAPHY

[1] N. K. E. Abbadi and N. E. Kadhim. Brain cancer classification based on features and artificial neural network. *International Journal of Advanced Research in Computer and Communication Engineering*, 6, Jan 2017.

[2] abhigoku10. `https://medium.com/@abhigoku10/activation-functions-and-its-types-in-artifical-neural-network-14511f3080a8`.

[3] N. Abiwinanda, M. Hanif, S. T. Hesaputra, A. Handayani, and T. R. Mengko. Brain tumor classification using convolutional neural network. In *World Congress on Medical Physics and Biomedical Engineering*, volume 3, 2018.

[4] P. Afshar, A. Mohammadi, and K. N. Plataniotis. Brain tumor type classification via capsule networks. *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3129–3133, 2018.

[5] P. Afshar, K. N. Plataniotis, and A. Mohammadi. Capsule networks for brain tumor classification based on mri images and course tumor boundaries. *CoRR*, abs/1811.00597, 2018.

[6] A. Azulay and Y. Weiss. Why do deep convolutional networks generalize so poorly to small image transformations?, May 2018. arXiv:1805.12177v1.

[7] M. T. Bahadori. Spectral capsule networks. *The Sixth International Conference on Learning Representations*, pages 0–1, Feb 2018. `https://openreview.net/pdf?id=HJuMvYPaM`.

[8] N. M. Balasooriya and R. D. Nawarathna. A sophisticated convolutional neural network model for brain tumor classification. In *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, pages 1–5, Dec 2017.

[9] J. Cheng. brain tumor dataset. Aug 2015.
`https://figshare.com/articles/brain_tumor_dataset/1512427`.

[10] J. Cheng, W. Huang, S. Cao, R. Yang, W. Yang, Z. Yun, Z. Wang, and Q. Feng. Enhanced performance of brain tumor classification via tumor region augmentation and partition. *PLoS ONE*, 10, Oct 2015.

[11] K. Chitta, J. M. Alvarez, and A. Lesnikowski. Large-scale visual active learning with deep probabilistic ensembles. *CoRR*, abs/1811.03575, 2018.

[12] F. Chollet. Before we begin: the mathematical building blocks of neural networks. In *Deep Learning with Python*, chapter 2, pages 25–55. Manning Publications Co., 2018.

[13] Da-Zeng Tian and Ming-Hu Ha. Applications of wavelet transform in medical image processing. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, volume 3, pages 1816–1821 vol.3, Aug 2004.

[14] K. S. Deepak, K. Gokul, R. Hinduja, and S. Rajkumar. An efficient approach to predict tumor in 2d brain image using classification techniques. In *2013 International Conference on Information Communication and Embedded Systems (ICICES)*, pages 559–564, Feb 2013.

[15] A. Deliège. Hitnet: Clarification questions. private, Jan 2019.

[16] A. Deliège, A. Cioppa, and M. V. Droogenbroeck.
`http://www.telecom.ulg.ac.be/hitnet/`.

[17] A. Deliège, A. Cioppa, and M. Van Droogenbroeck. HitNet: a neural network with capsules embedded in a Hit-or-Miss layer, extended with hybrid data augmentation and ghost capsules. *ArXiv*, abs/1806.06519, June 2018.

[18] U. e Hani, S. Naz, and I. A. Hameed. Automated techniques for brain tumor segmentation and detection: A review study. In *2017 International Conference on Behavioral, Economic, Socio-cultural Computing*, pages 1–6, Oct 2017.

[19] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. Hit-and-miss transform, 2004. `https://homepages.inf.ed.ac.uk/rbf/HIPR2/hitmiss.htm`.

[20] A. Géron. Capsule networks (capsnets) tutorial. `https://www.youtube.com/watch?v=pPN8d0E3900`.

[21] T. Gupta, P. Manocha, T. K. Gandhi, R. K. Gupta, and B. K. Panigrahi. Tumor classification and segmentation of mr brain images. *CoRR*, abs/1710.11309, 2017.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[24] G. Hinton, S. Sabour, and N. Frosst. Matrix capsules with em routing. *International Conference on Learning Representations 2018*, Nov 2017. `https://openreview.net/pdf?id=HJWLfGWRb`.

[25] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. *International Conference on Artificial Neural Networks*, pages 44–51, 2011. `http://www.cs.toronto.edu/~fritz/absps/transauto6.pdf`.

[26] S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, 2006.

[27] M. Huang, W. Yang, Y. Wu, J. Jiang, Y. Gao, Y. Chen, Q. Feng, W. Chen, and Z. Lu. Content-based image retrieval using spatial layout information in brain tumor t1-weighted contrast-enhanced mr images. In *PloS one*, 2014.

[28] T. Iesmantas and R. Alzbutas. Convolutional capsule network for classification of breast cancer histology images. In *ICIAR*, 2018.

[29] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379, June 2009.

[30] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class batch-mode active learning for image classification. In *2010 IEEE International Conference on Robotics and Automation*, pages 1873–1878, May 2010.

[31] A. Karpathy. Cs231n convolutional neural networks for visual recognition. `http://cs231n.github.io/convolutional-networks/`.

[32] S. Korolev, A. Safiullin, M. Belyaev, and Y. Dodonova. Residual and plain convolutional neural networks for 3d brain mri classification. *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 835–838, 2017.

[33] R. Kotikalapudi. `https://github.com/raghakot/keras-resnet`.

[34] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009. `https://www.cs.toronto.edu/~kriz/cifar.html`.

[35] S. Lab. `http://www.image-net.org/`.

[36] R. LaLonde and U. Bagei. Capsules for object segmentation. *Conference on Medical Imaging with Deep Learning*, Apr 2018. `https://arxiv.org/pdf/1804.04241.pdf`.

[37] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, May 2015. `http://dx.doi.org/10.1038/nature14539`.

[38] J. E. Lenssen, M. Fey, and P. Libuschewski. Group equivariant capsule networks. In *NeurIPS*, 2018.

[39] D. N. Louis and et al. The 2007 who classification of tumours of the central nervous system. *Acta neuropathologica*, 114(2):97–109, Jul 2007.

[40] e. a. Mark Everingham. `http://host.robots.ox.ac.uk/pascal/VOC/voc2012/`.

[41] MedlinePlus. Brain tumors, 2018.

[42] National Brain Tumor Society. Tumor types: Understanding brain tumors, 2016.

[43] National Institutes of Health - National Institute of Neurological Disorders and Stroke. Brain and spinal tumors information page, 2018.

[44] K. Qiao, C. Zhang, L. Wang, B. Yan, J. Chen, L. Zeng, and L. Tong. Accurate reconstruction of image stimuli from human fmri based on the decoding model with capsule network architecture. *CoRR*, abs/1801.00602, 2018.

[45] S. Ramasinghe, C. D. Athuraliya, and S. H. Khan. A context-aware capsule network for multi-label classification. In *ECCV Workshops*, 2018.

[46] V. P. G. P. Rathi and S. Palani. Brain tumor mri image classification with feature selection and extraction using linear discriminant analysis. *CoRR*, abs/1208.2128, 2012.

[47] D. Rawlinson, A. Ahmed, and G. Kowadlo. Sparse unsupervised capsules generalize better, Apr 2018. `https://arxiv.org/pdf/1804.06094.pdf`.

[48] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. *Conference on Neural Information Processing Systems*, pages 3859–3869, Nov 2017. `https://arxiv.org/pdf/1710.09829.pdf`.

[49] J. Schmeelk. Wavelet transforms on two-dimensional images. *Math. Comput. Model.*, 36(7-8):939–948, Nov. 2002.

[50] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison, 2010.

[51] A. Shahroudnejad, A. Mohammadi, and K. N. Plataniotis. Improved explainability of capsule networks: Relevance path by agreement. *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 549–553, 2018.

[52] M. Sharma. Artificial neural network fuzzy inference system (anfis) for brain tumor detection. *CoRR*, abs/1212.0059, 2012.

[53] V. K. Sudarshan, M. R. K. Mookiah, U. R. Acharya, V. Chandran, F. Molinari, H. Fujita, and K. H. Ng. Application of wavelet techniques for cancer diagnosis using ultrasound images: A review. *Computers in Biology and Medicine*, 69:97 – 111, 2016.

[54] K. Usman and K. Rajpoot. Brain tumor classification from multi-modality mri using wavelets and machine learning. *Pattern Analysis and Application*, 20:871–881, Feb 2017.

[55] A. Vidyarthi and N. Mittal. Comparative study for brain tumor classification on mr/ct images. In *Proceedings of the Third International Conference on Soft Computing for Problem Solving, Advances in Intelligent Systems and Computing*. Springer India, 2014.

[56] J. S. Walker. Daubechies wavelets. In *A Primer on Wavelets and Their Scientific Applications*, chapter 2. CRC Press, 1999.

[57] D. Wang and Q. Liu. An optimization view on dynamic routing between capsules. *The Sixth International Conference on Learning Representations*, pages 0–1, Feb 2018. `https://openreview.net/pdf?id=HJjtFYJDf`.

[58] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Browstow. Harmonic networks: Deep translation and rotation equivariance, Apr 2017. arXiv:1612.04642v2.

[59] E. Xi, S. Bing, and Y. Jin. Capsule network performance on complex data. *The Sixth International Conference on Learning Representations*, pages 42–52, Dec 2017. arXiv:1712.03480v1.

[60] Y. Zhang, Z. Dong, L. Wu, S. Wang, and Z. Zhou. Feature extraction of brain mri by stationary wavelet transform. In *2010 International Conference on Biomedical Engineering and Computer Science*, pages 1–4, April 2010.

[61] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239, 2017.