TUMOR IDENTIFICATION: EXPERIMENTAL DESIGN FOR MACHINE LEARNING ON BIG DATA

Yi Wang
Undergraduate in EECS
University of California, Berkeley
wangyi1022@berkeley.edu

Daniel Lin
Master of Science in EECS
University of California, Berkeley
pandan@berkeley.edu

Chen Wang
Master of Engineering in EECS
University of California, Berkeley
chw@berkeley.edu

May, 2022

Contents

1 Introduction

1.1 Motivation

In the medical society, tumors often come up in diagnoses. It is often necessary to determine if the tumor is benign or malignant. In Figure 1, we see examples of tumors and their corresponding labels. The applications of a successful learner on these tumors would greatly increase processing time, and we could diagnose tumors a lot faster and with high accuracy.

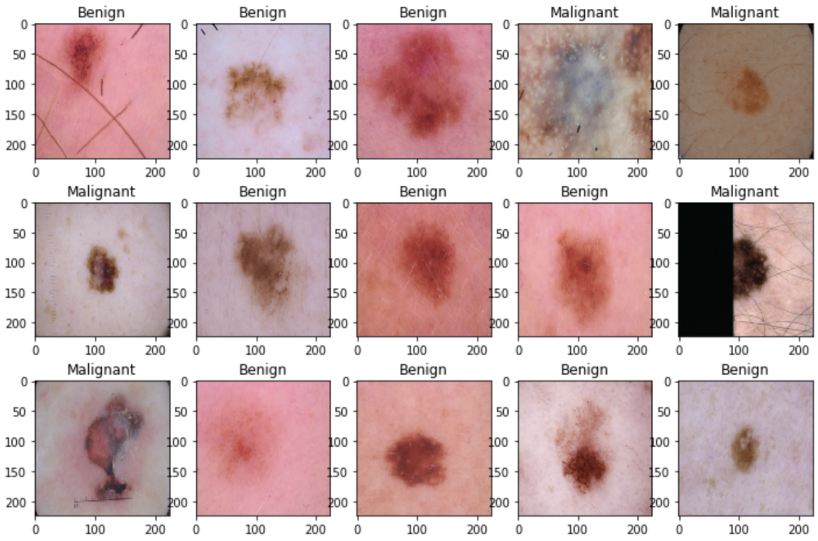

Figure 1: Benign and Malignant Tumor Examples

1.2 Background Work

Most of the work done in image classifcation use a Convolutional Neural Network (CNN). CNNs, with a general model shown in 2 are useful because they focus on certain areas of images, making it easier to recognize pattern. A standard network used is the ResNet [1], which uses skip connections to help the vanishing gradient problem when learning. We use a pretrained ResNet50 to generate a feature map.

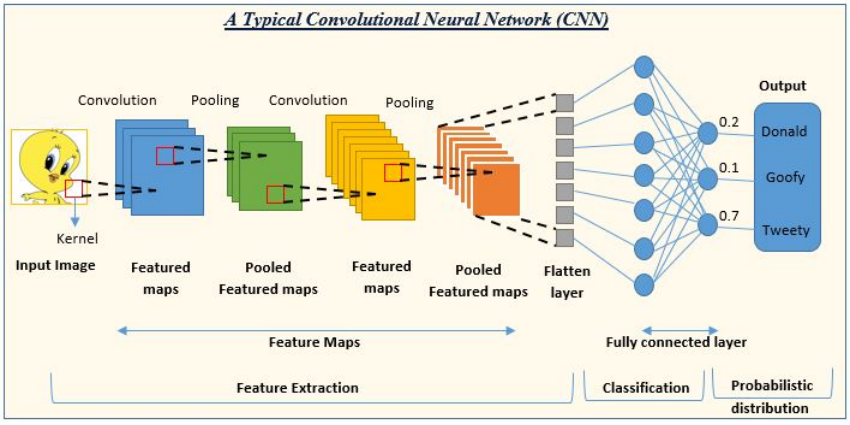

Figure 2: CNN architecture

2 Model Architecture

2.1 Data-set Labeling

The machine learner is to solve a binary classification problem. It should predict whether the image corresponds to a tumor, with a benign tumor being labeled as 0 and a malignant tumor labeled as 1. The labeling information can be found from the following summarization figure below 3 and 4. Because we are given these labels as ground truth, we have to assume that it is fully accurate for our analysis. Each image has a corresponding ground truth, there is no notion of an annotator agreement. Thus, the agreement value and the Cohen Kappa score are both 1.

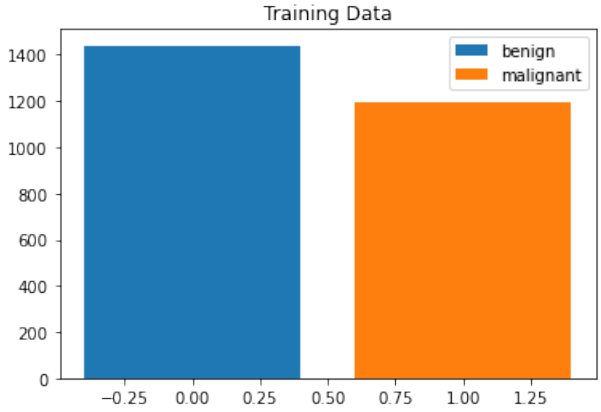

Figure 3: Train set label

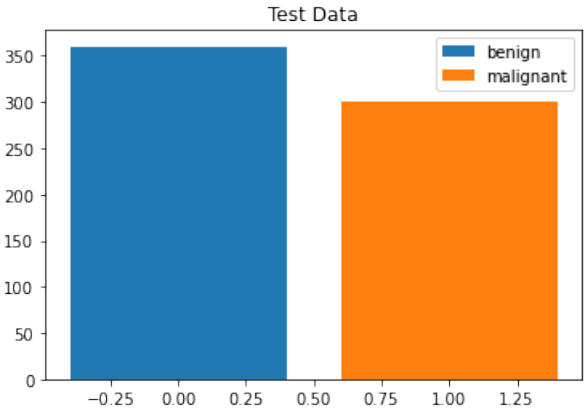

Figure 4: Test set label

To summarize and to make sure that the instructors can find the answers more clearly, we re-iterate and provide a clearer answer to following questions regarding our research topic:

- What is the variable the machine learner is supposed to predict?

We want to predict the label for each image. In our research project, we work on the medical image data-set where each image is a picture of the tissue. The machine learner should predict whether the tissue in the picture has a `benign` tumor or a `malignant` tumor.

- How accurate is the labeling?

In our data-set, the images are classified by the professional doctors. In this case, we assume that the labeling is 100% accurate.

- What is the annotator agreement (measured)?

In our example, each image is classified as either `benign` or `malignant`. The classification is clear and accurate in the data-set. Therefore, we have a annotator agreement 1 and the Cohen Kappa score 1.

2.2 Project Context and Data Characteristics Analysis

For success, we need to develop a classifier that is better than random guessing. Randomly guessing would give us an accuracy in a binary classification problem of $\frac{1}{2}$. Without any additional training or knowledge of the dataset, each image can be of any of the classes, so this is a suitable metric to see if our classifier performs better than random.

The training dataset has 2637 examples, while the test dataset has 660 samples. In the train dataset, the dataset is divided into 1440 benign samples and 1197 malignant samples. In the test dataset, the dataset is divided into 360 benign samples and 300 malignant samples. The ratios of benign to malignant are both about 6:5, showing that our test set has a similar data distribution as the train dataset.

Each image is an RGB image of input size of 224x224x3, but we randomly crop the image so that it is 128 x 128 x 3. After putting the image through a ResNet50 encoder, we get a feature vector of size 1x2048 before the final classification layer. There are multiple channels, but there is overall only one single image, so we only use one mode. These images are all static, so there is no concept of temporal information. Because the images are manually labeled and randomly cropped, there is going to inherently be noise involved in the labeling and the dataset. There should not be too much bias overall outside of the class imbalance.

To summarize and to make sure that the instructors can find the answers more clearly, we re-iterate and provide a clearer answer to following questions regarding our research topic:

- What is the required accuracy metric for success?

We need to develop a classifier that is better than random guessing to succeed in this project.

- How much data do we have to train the prediction of the variable?

We are going to train on 2637 samples.

- Are the classes balanced?

The classes are mostly balanced. The class distribution can be referred from the figures above 3 and 4.

- How many modalities could be exploited in the data?

Our image data-set has 3 modalities.

- Is there temporal information?

summer@berkeley.edu

No.

- How much noise are we expecting?

Since we are working on the image data-set, we are mostly dealing with image signal capturing noises.

- Do we expect bias?

Because our class distribution is balanced, we won't have much bias.

3 Memory Equivalent Capacity and Data Sufficiency Analysis

3.1 Memory Equivalent Capacity Analysis

There are 1237 thresholds in the dataset. This translates to a dictionary dataset MEC as $\log_2(\text{thresholds}) * (\text{number of features}) = \log_2(1237) * 2048 = 20138.345$ bits MEC for the data. We take the ceiling function to get that it would take 20139 bits to classsify the dataset as a dictionary. However, the MEC for a neural network to classify this dataset would be 153 bits according to Brainome.

3.2 Edge Cases

The following numbers are generated from a Brainome Neural Network. The expected generalization of a neural network is 0.24 bits / bit. The actual generalization is 10.85 bits/bit. The resilience to noise ratio is -1.32 dB. The resilience corresponding to the actual generalization is -1.04 dB. To convert this value to bits/bit of resilience, we simply multiply by 20. This gives a resilience of -20.8 bits/bit. There are a total of 2048 (feature vector size) * 32 (32-bit float) = 65536 bits of signal. We multiply the signal by the resilience to get 65536 bits * -20.8 bits / bit = -1363148.8 bits of noise. Our input data is an RGB image of size 224 x 224 x 3. Because each element is a 32-bit float, the original data contains 224 x 224 x 3 x 32 = 4816896 bits.

We realize that the amount of noise in the data is greater than the resilience in the model, so maybe the resilience is not enough to fully solve this task. Our validation accuracies also aren't the highest, so maybe we realize there is too much noise in the dataset and not enough resilience.

There could some adversarial examples, but for the most part they would be in the labeling and inputs. We should view most of these examples as noise, as sometime there would be a mislabeling. Similarly, from a visual standpoint, it sometimes isn't immediately obvious the difference between the two samples.

For the most part, we do not expect too much data drift because the basics of what makes a tumor malignant and benign should not change too much. Even though circumstances may change and tumors may evolve, the base analysis of what to look for should still look somewhat similar.

3.3 Data Sufficiency

After we input the corresponding data into `brainome` using the command

```
ubuntu@ip-10-21-3-156: /brainome$ brainome result_data_train.csv
```

, we can get the following prompt, which means that we possibly have the enough data to generalize

```
Data Sufficiency:  Maybe enough data to generalize.  [yellow]
```

Then we analyzed the capacity progression by using brainome, the capacity progression chart according to our data-set is shown in the Table 1 and Figure 5

Fraction	5%	10%	20%	40%	80%	100%
Capacity Progression	8	9	10	11	11	12

Table 1: Capacity Progression Analysis Report

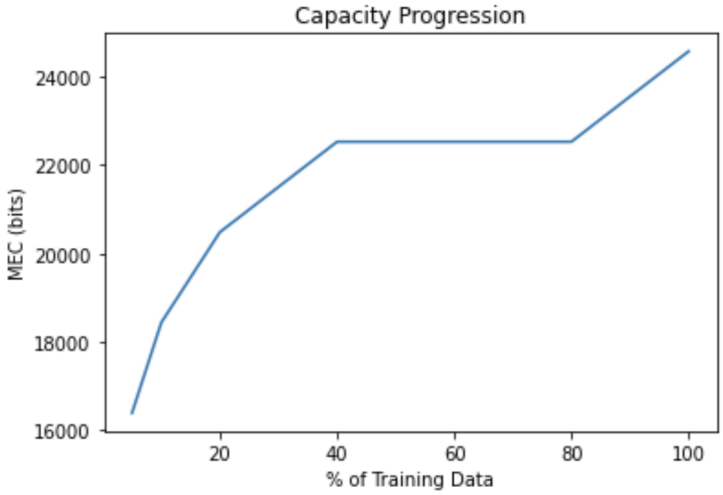

Figure 5: Capacity Progression Chart

4 Machine Learner Memorization

We are not able to reach near 100% memorization. The accurancy we get is abot 65.79%. The reason we cannot achieve close to that is likely because of the loss of information and the somewhat random nature when we crop and downscale the feature vector for our analysis. The feature vector from the ResNet 50 also may not be fine-tuned on our specific use case, so the data might not be perfect.

5 Machine Learner Generalization

Our generalization graph is shown below in Figure 6.

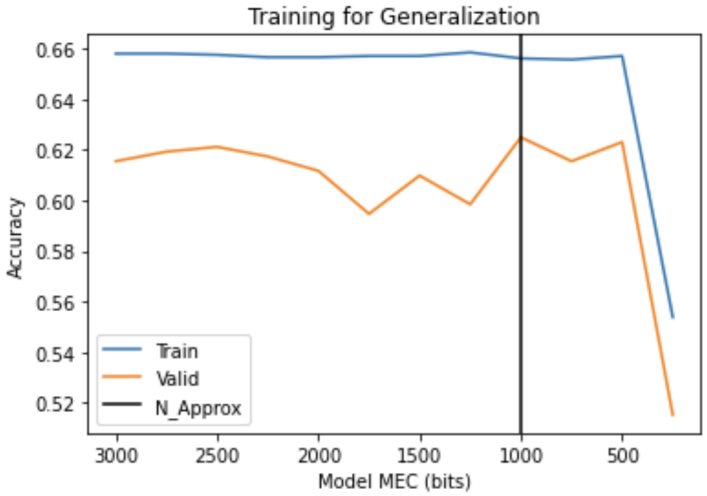

Figure 6: Generalization Chart

We decided that a hidden layer of size 4 was good enough for our project. Here, our training accuracy was 0.657, and the generalization is 0.282. We get that the MEC of our model can be defined as (# of inputs) * (hidden layer size + hidden layer size + hidden layer size + min(hidden layer size, hidden layer size * 2 + 2)) = 2048 * hidden layer size + hidden layer size + hidden layer size = 2050 * 4 = 8200.

Generalization = # correctly classified instances / MEC of model = (0.657 * 2637) / (2050 * 4) = 0.211 bits/bit

If we could use a machine learner with a smaller MEC with similar accuracy, that would be ideal. Currently, we are using an MLP with hidden layers, in which each hidden layer adds 2050 bits to the total capacity. If we look at the other potential options that Brainome provides, we see that we can use a random forest. The odel achieves the same traininig accuracy with only a slightly lower validation accuracy of 58.89% rather than 61.5%. The Brainome random forest has an MEC of 60 bits, while the Brainome neural net has a MEC of 153 bits. There is a tradeoff between MEC and accuracy that we could fine-tune.

On an independent test data, our predictor does relatively similarly to the training set. We get an accuracy of 61.5%. The corresponding generalization is then (0.615 * 660) / (2050 * 4) = 0.050 bits/bit.

We see that our network cannot fit to the training dataset that well, but there is also not too big of a dip from the training to validation results. We are confident in our results that the network can generalize, but the overall accuracy is not the greatest.

6 Quality Assurance

We can use another similar dataset to validate the findings of this network. For example, this network could be used to look into other domains that also involve binary classification that have similar characteristics. For example, benign and malignant tumors are both subclasses of tumors, so maybe we take a look into two specific types of cancer cells to distinguish. If the two classes are very different, it may be too easy for the network to distinguish between them and the results would be less similar to the ones we found. To validate the machine learner model, we have used other retinal data sets and

	TP	FP	TN	FN
value	902	0	1440	1440

Another additional quality assurance measure that we can try is to use other networks to generate our initial feature maps, i.e AlexNet [2] or VGG [3]. In addition, we can also include more augmentations to help our network generalize to more unseen data. Standard data augmentations may include shifting color, saturation, or hue, or more flipping, cropping, and rotation.

When curating the dataset, we can help normalize the images so that the distribution of pixels as the same throughout. By normalizing the values, it allows the machine learner to more accurately determine which features are important. It also allows for the learner to not be over-reliant on intensity, which can be caused by different image-taking processes.

There are no time present, so there is no need to worry about temporal information in this experiment.

7 Repeatability and Reproducibility

To ensure repeatability, we have also submitted the dataset and the calcutions done in a standard Jupyter notebook. A user can simply rerun the cells to achieve similar results. Results may be slightly different because of inherent random cropping and initializations of weights, but the numbers should be in the same ballpark.

To ensure reproducibility, we have submitted the code that we used. We use our findings from Brainome, which anybody can use, and take calculations from the algorithms learned in class.

These results should overall be repeatable because the task of binary classification is not unique to this dataset. A user should be able to simply substitute a different dataset and get similar results.

References

[1] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CVPR*, 2016.

[2] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *NeurIps*, 2012.

[3] Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. 2015.