Brain Tumor Detection

Detecting Brain Tumors in Patients Given Brain Scan Imagery

Team: Amit Gattadahalli, Arindam Sett, Beijing Wu, Neil Bhatia
Class: MIDS W207 - Nedelina Teneva
Term: Spring 2022

Berkeley
UNIVERSITY OF CALIFORNIA

Motivation

- Brain tumors are the **2nd leading cause of cancer death** in children under age 15 and **2nd fastest growing cause of cancer death** among those over age 65
- A brain tumor is a mass or growth of abnormal cells in the brain
  - Non-cancerous (benign) vs. cancerous (malignant)
  - Primary (begin in brain) vs. secondary (metastasize to brain)
  - More than 120 different types of brain tumors
- Benign brain tumors can still be dangerous
  - Damage and compress parts of the brain → severe dysfunction
- Risk factors: exposure to radiation or chemical, age, family history of brain tumors
- Symptoms: depend on size and location in the brain
  - Damage to vital tissue & pressure on the brain or swelling and a buildup of fluid around the tumor
  - Headaches, seizures, change in personality/memory, change in speech, etc.
- Diagnosis: neurological exam → imaging tests (CT/MRI/X-rays) → biopsy
  - Imaging techniques: identify tumor, pinpoint tumor location, evaluate tumor & plan treatment

Introduction to Dataset and EDA

- Relatively balanced binary classification problem

Label	# of Images	Percentage
Brain Tumor	2513	54.6%
Healthy	2087	45.4%
Total	4600	100.0%

- Variety of formats (.jpeg, .png, .tif), image sizes (459 unique image sizes), and color schemes (RGB, greyscale)
- The variability in image specifications highlights the need to standardize and preprocess the images so our machine learning model's can learn from data being expressed on a common scale.

Preprocessing

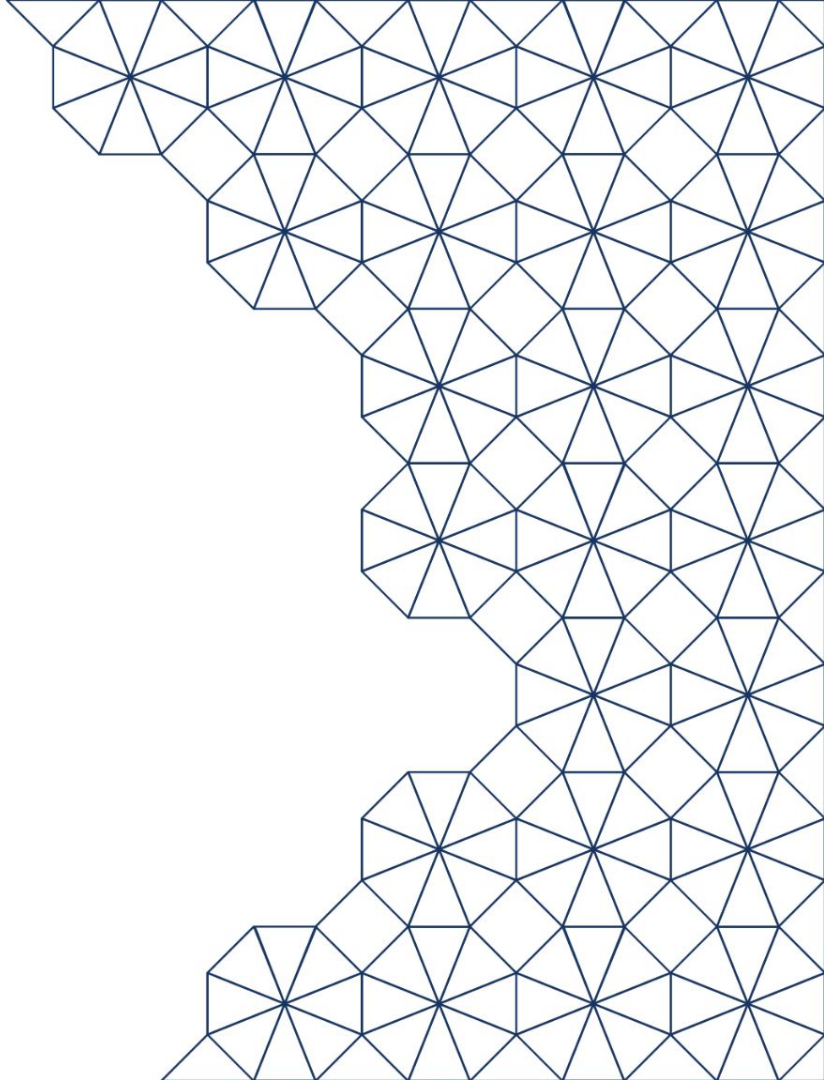


Image Preprocessing Functions

Goal: preserve/maximize useful features & decrease computational costs

	Speed Test (sec/img)	
Standardization		To modify images to 2D 256x256 in size and in greyscale
Blurring	0.003	To reduce image noise and details using custom filter (Gaussian, box)
Pooling	0.014	To apply an aggregation function (max, min, avg) to a chunk of an image
Binarization	0.09	To convert grayscale images into black and white (sharpen)
Cropping	0.136	To remove unnecessary padding and minimize sensitivity to relative scan size
Resizing		To further reduce image size using pillow; single step version of pooling, blurring, and binarization

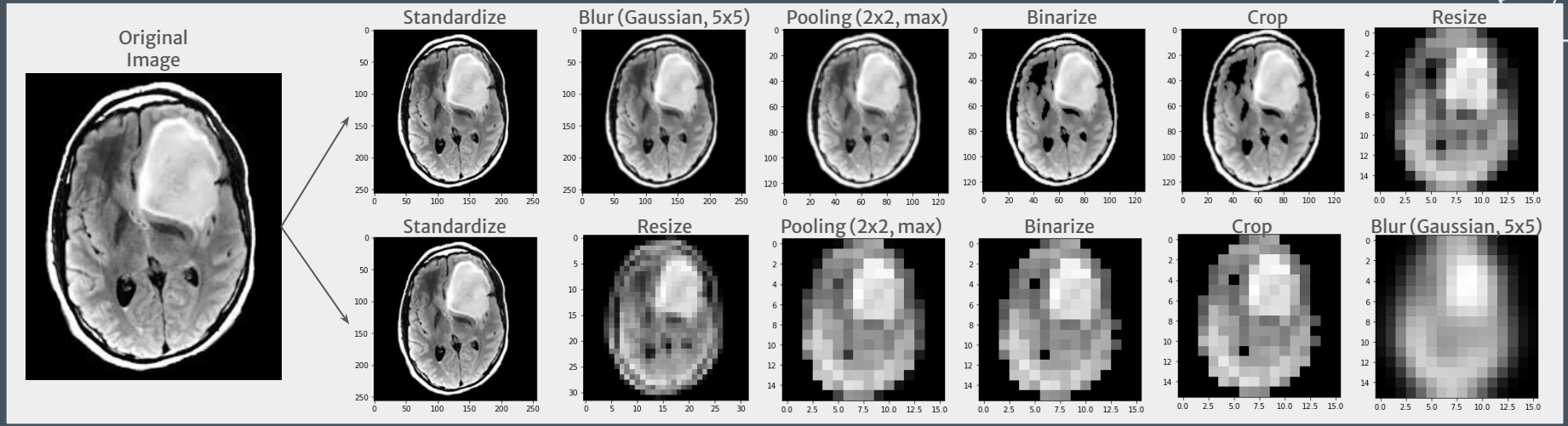
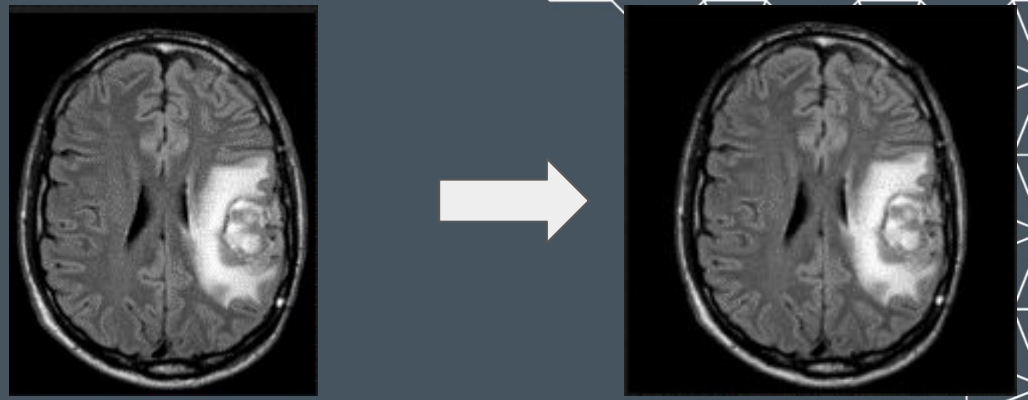


Image Standardization and Cropping

Standardization

- To modify our images, and associated vector representations, to be the same size (150 x 150 2D or 22500 1D) and color scheme (greyscale)



Cropping

- To remove variable and noisy/unnecessary padding around the brain image. This helps to maximize the amount of useful features while making models insensitive to original scan size.

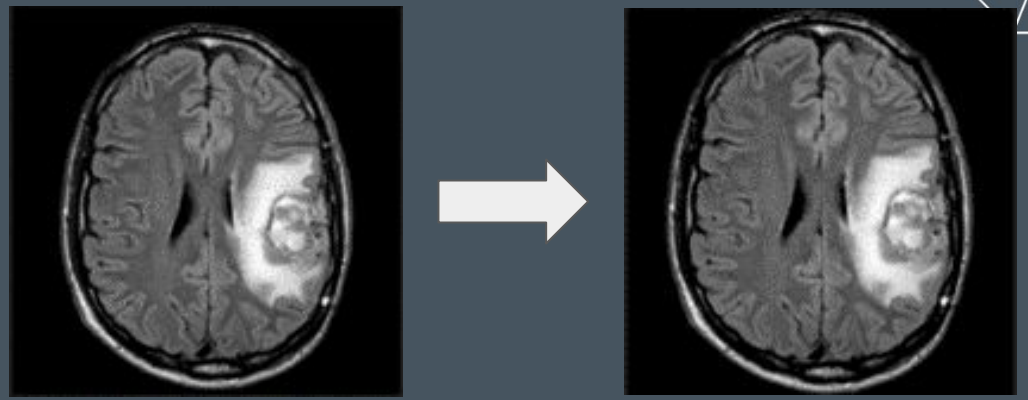
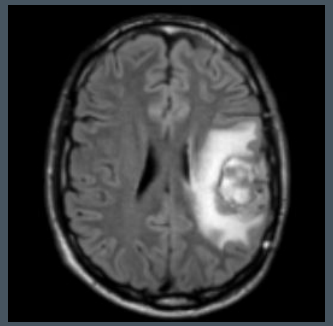
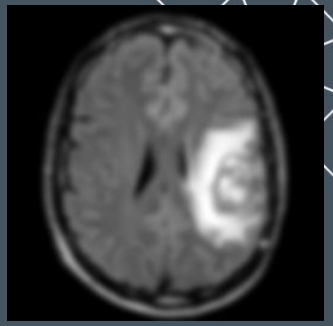


Image Blurring

In image processing, a Blur is utilized to reduce image noise and details



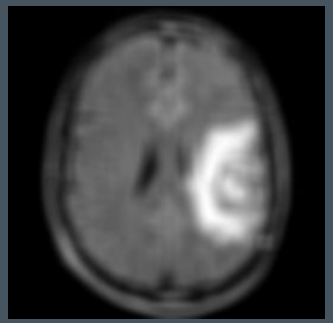
Original Image



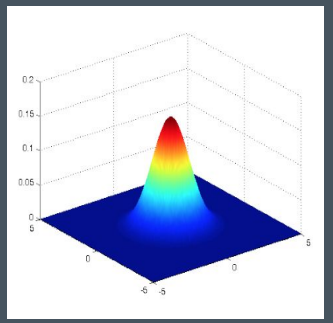
Gaussian Filter: k=(7,7), σ_x=0, σ_y=0

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Normalized Box
Example: k=(3,3)



Normalized Box Filter: k=(7,7)



Gaussian Function

Image Binarization

- Avoid background noise generated in images
- Convert grayscale images into
  - Binarized: black (0) and white (1)
- Provide sharper and clearer contours of objects in the image
- Selection of threshold
  - Manual
  - Otsu's method: threshold is determined by minimizing intra-class intensity variance (maximizing inter-class variance)
    - `skimage.filters.threshold_otsu()` - returns a single value that separate pixels into foreground and background

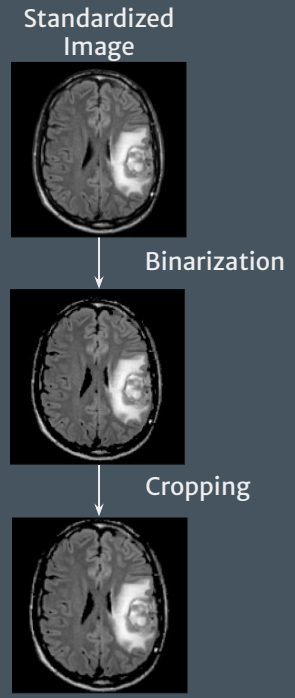
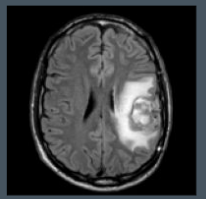


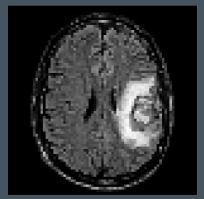
Image Pooling

- **What is Pooling?**
  - Pooling takes a chunk (a pool) of an image and applies an aggregation function like max, min, or avg. The aggregate value of the chunk is stored, creating a new image representative of the old image.
- **Benefits of Pooling**
  - Intensifies important features while discarding irrelevant information -> smaller image size with the "same idea," leading to decreased computation costs.
- **Pooling operations:**
  - **Main:** Max Pooling, Min Pooling, Mean Pooling
  - **Others:** median, sum, custom
- **Pooling Function**
  - skimage.measure.block_reduce()
  - We specify an image, "block size" (pool/chunk), padding values, and pooling function.

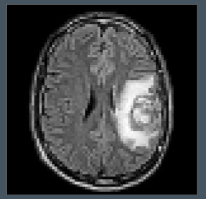
Standardized Image



Min Pooling



Mean Pooling



Max Pooling

Pixels pooled with block size of 2x2
- *The **image size** for each is reduced by ~50%*

Model Development

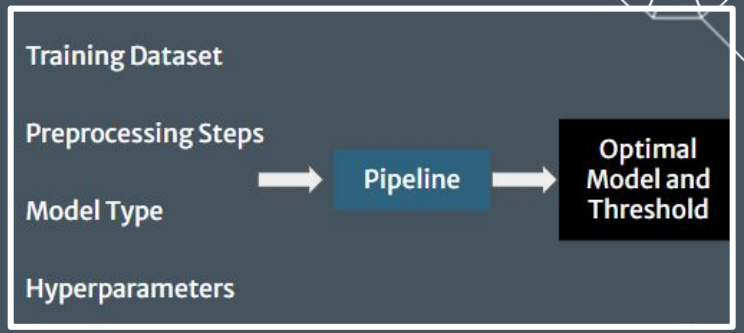
Optimal Model Development

Key Consideration:

- Given preprocessed labeled data, what is an optimal model and associated hyperparameter settings that will generalize best to out of sample data (measured via F1 score)

Solution:

- Develop pipeline that applies specified preprocessing steps to original data
- Pipeline executes grid search with a specified model and hyperparameter options to identify model / hyperparameter combo that maximizes F1 score out of sample
- Pipeline conducts a threshold analysis to find predicted probability threshold that maximizes optimal model performance



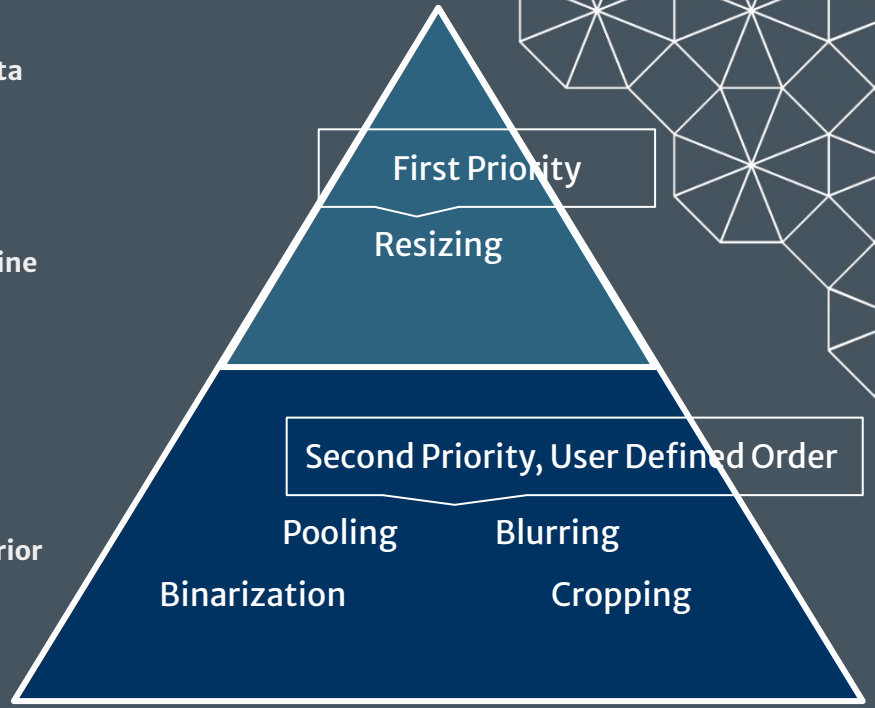
Identifying Optimal Preprocessing Steps

Key Consideration:

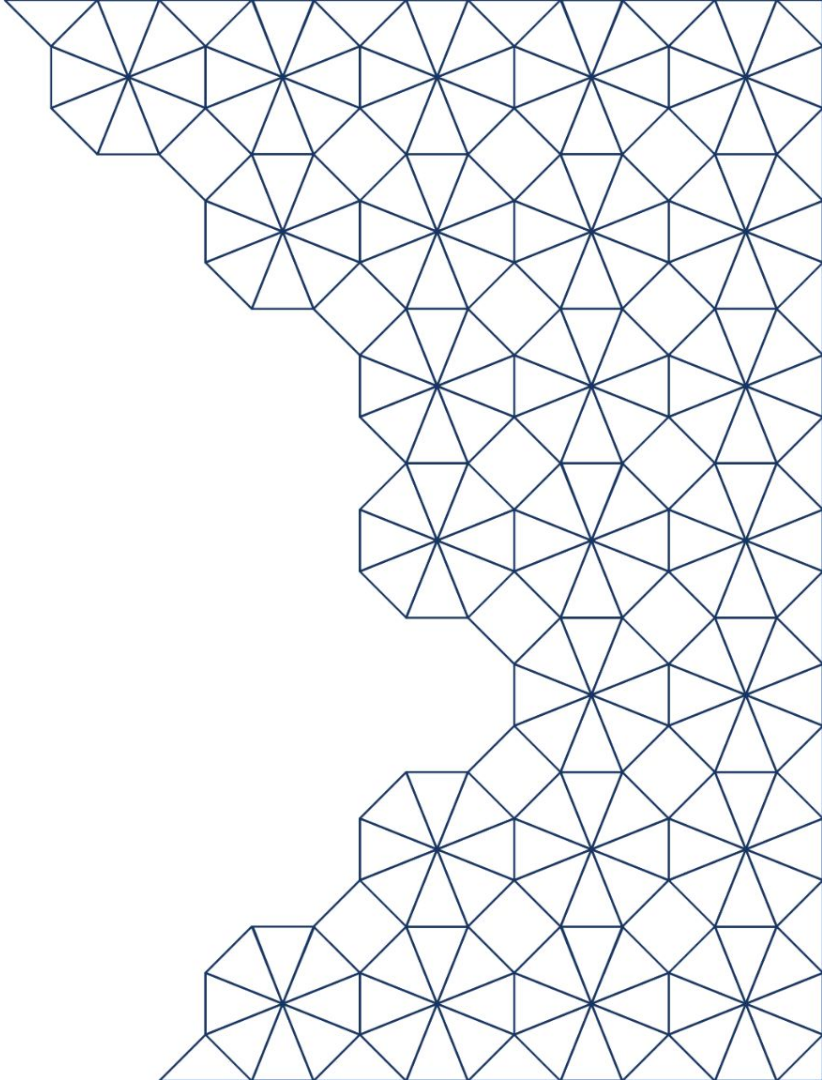
- What sequence of preprocessing steps enable an optimal model to generalize best to out of sample data (measured via F1 score)

Solution:

- Develop a wrapper around model development pipeline that evaluates the impact of different preprocessing steps on optimal model performance.
- Greedy approach, evaluates the impact of various preprocessing methods sequentially
- Only executes a specific preprocessing step if it improves the performance of optimal model given prior preprocessing decisions



Model Evaluation

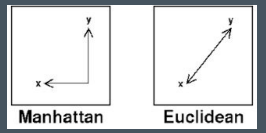


Distance Based Models

K-Nearest Neighbors Algorithm (KNN)

- Supervised learning method
- Assumes that similar things exist in close proximity → calculation of distance
- Hyperparameters:
  - K: # of neighbors
  - Distance metric: Minkowski distance - Lp norm of two vectors
    - Euclidean distance (L2 norm)
    - Manhattan distance (L1 norm)
  - Weights
    - Uniform: each neighbor carries the same weight
    - Distance-based

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^c \right)^{\frac{1}{c}}$$

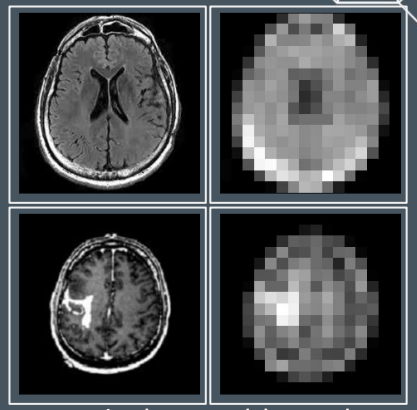


	Optimal KNN
Hyperparameters	n_neighbor=1 p=1 weights=uniform
Preprocessing	Resize to 16x16
Cross Validated F1 Score	0.9742

KNN Evaluation on Test Data →

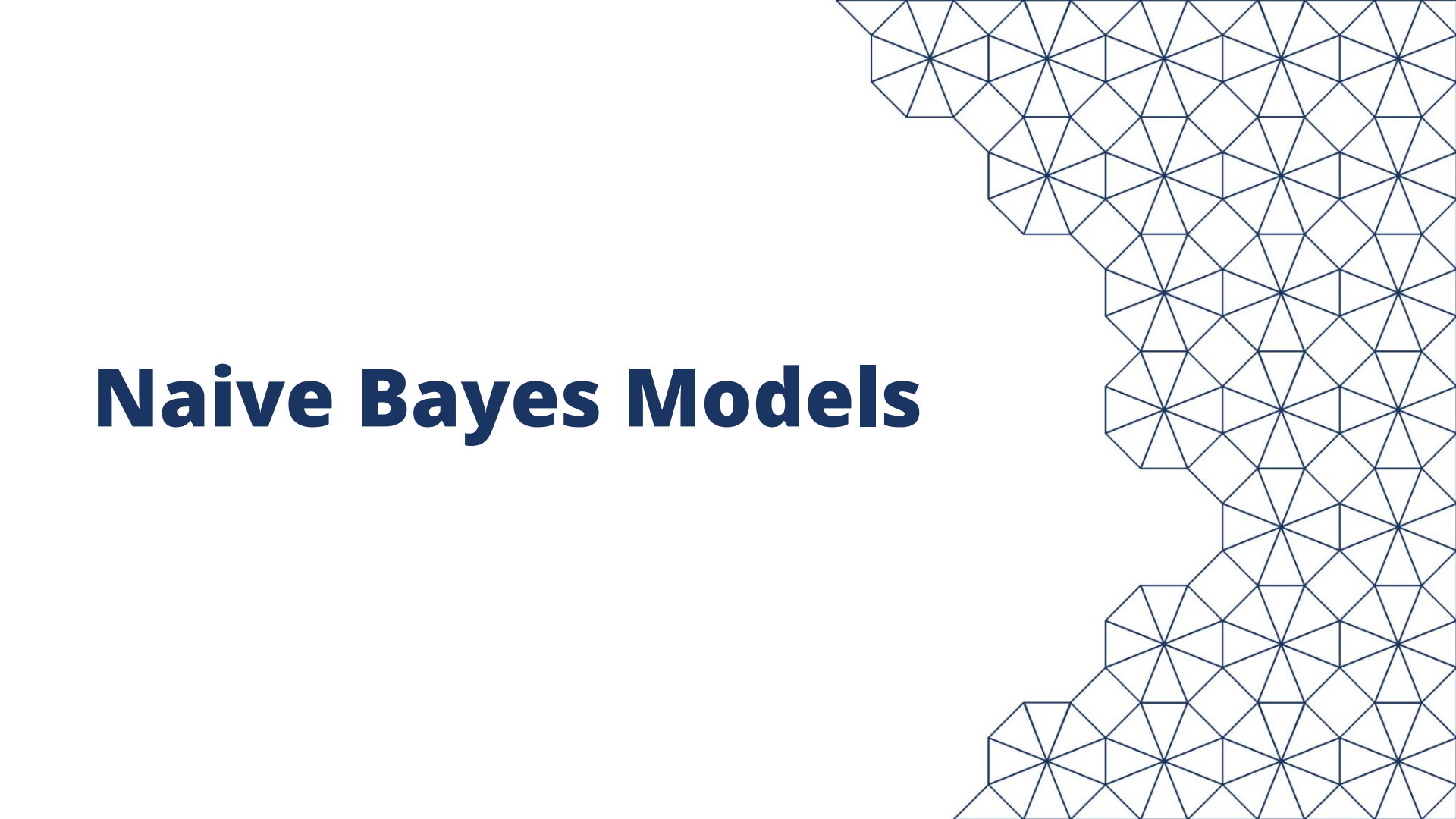
	Pred Healthy	Pred Cancer
True Healthy	628	18
True Cancer	11	723

F1 Score: 0.9803
Accuracy: 0.9790
True Cancer Detection Rate: 0.985
False Positive Rate: 0.028



Optimal KNN model correctly classified these test images

Naive Bayes Models



Bernoulli Model Approach

- **We applied our preprocessing optimization pipeline, finding the best scoring model using these steps:**
  1. Initial Standardization + Resizing of Images, from 256x256 to 128x128.
  2. Normalized Box Filter Blurring with Filter Kernel of (3,3)
  3. Maximum pooling with a pool size of (8, 8)

- **Model Results as a result of these steps:**
  - **F1 Score: .716**
  - **Accuracy: .618**
- **Confusion Matrix Breakdown**

True Negatives *False Positives*

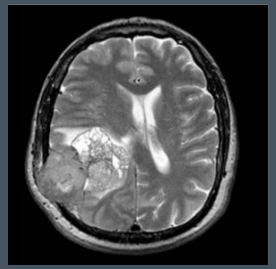
188	458
69	665

False Negatives *True Positives*

- Total Samples = 1380
- (188+ 665)/1380 = .618 = Accuracy
- (69 + 458)/1380 = .382 = Error Rate

Our view vs model view

No Preprocessing



256x256

All 3 Preprocessing Steps

16x16

***This model correctly classified this image as Cancer**

Gaussian Model Approach

- **We applied our preprocessing optimization pipeline, finding the best scoring model using these steps:**
  1. Initial Standardization + Resizing of Images, from 256x256 to 128x128.
  2. Maximum pooling with a pool size of (16, 16)

- **Model Results as a result of these steps:**
  - **F1 Score: .729**
  - **Accuracy: .657**

- **Confusion Matrix Breakdown**

True Negatives *False Positives*

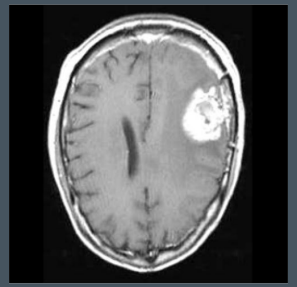
272	374
99	635

False Negatives *True Positives*

- Total Samples = 1380
- (272+ 635)/1380 = .657 = Accuracy
- (99 + 374)/1380 = .343 = Error Rate

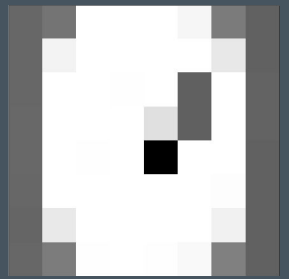
Our view vs model view

No Preprocessing



256x256

Both 2 Preprocessing Steps



8x8

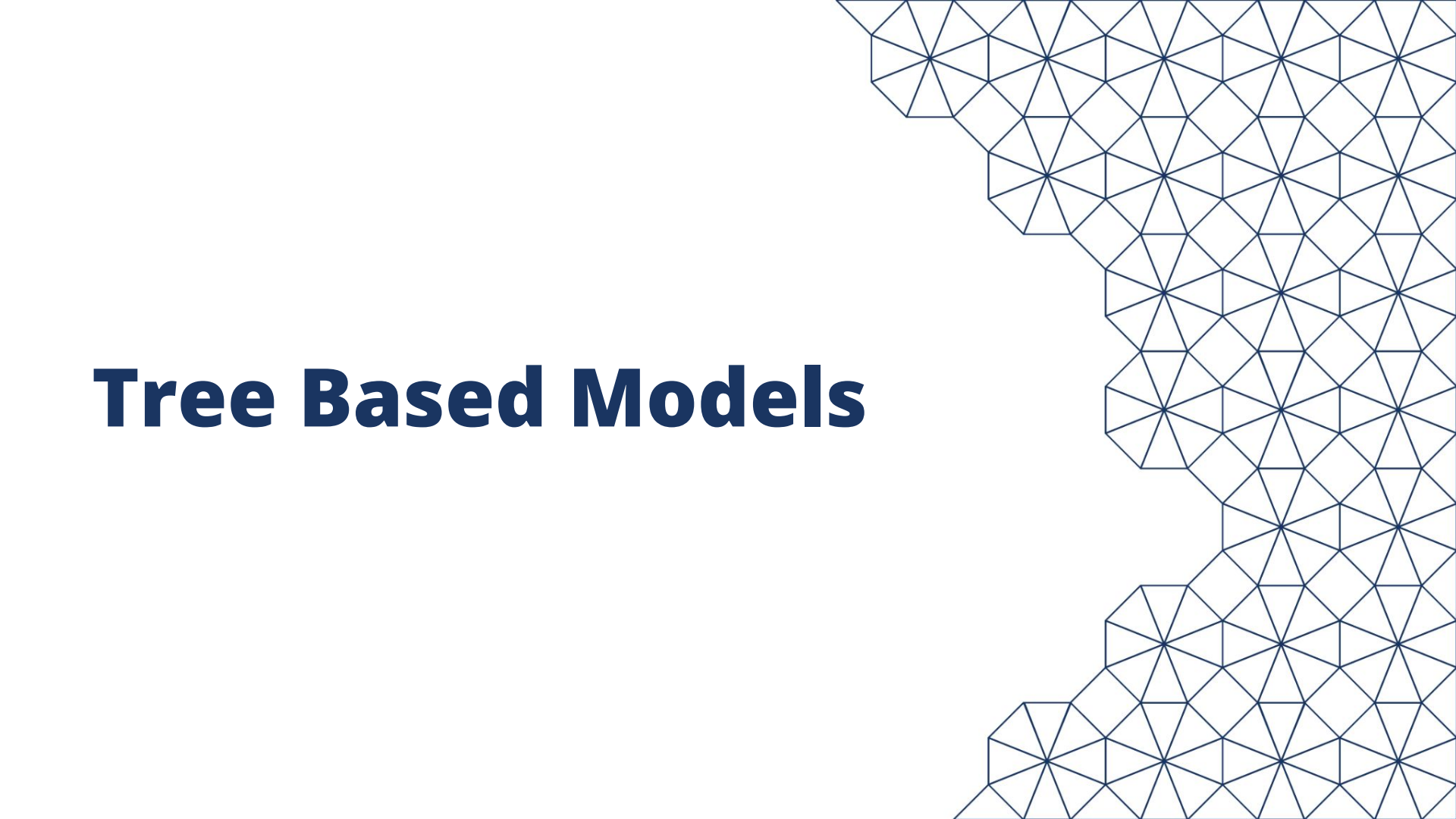
***This model correctly classified this image as Cancer**

Final Verdict and Naive Bayes Limitations

Performance	Bernoulli	Gaussian
F1 Score	.716	.729
Accuracy	.618	.657

- **Best Model:** Gaussian NB was an overall better performing model
- **Limitations:** Naive Bayes produces unreliable classification accuracy, but decent class label association probability estimates.

Tree Based Models



Random Forest vs Gradient Boosted Classification Tree (GBCT)

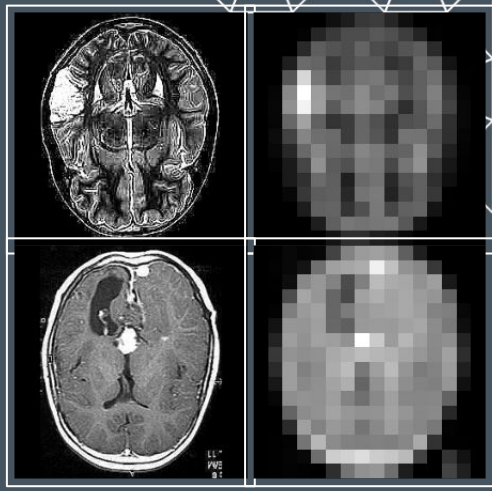
- During training, both methods learn what combination of grayscale intensities in different locations of an image result in a cancerous vs non-cancerous image

	Optimal Random Forest	Optimal GBCT
Hyperparameters	Min 3 Samples/Leaf, 500 Trees	Max Depth = 4, 500 Trees
Threshold	0.56	0.51
Preprocessing	Resize to 32x32, Gaussian Blur	Resize to 16x16
Cross Validated F1 Score	0.9615	0.9703
Model Size	11.07 MB	1.04 MB

GBCT Evaluation on Test Data

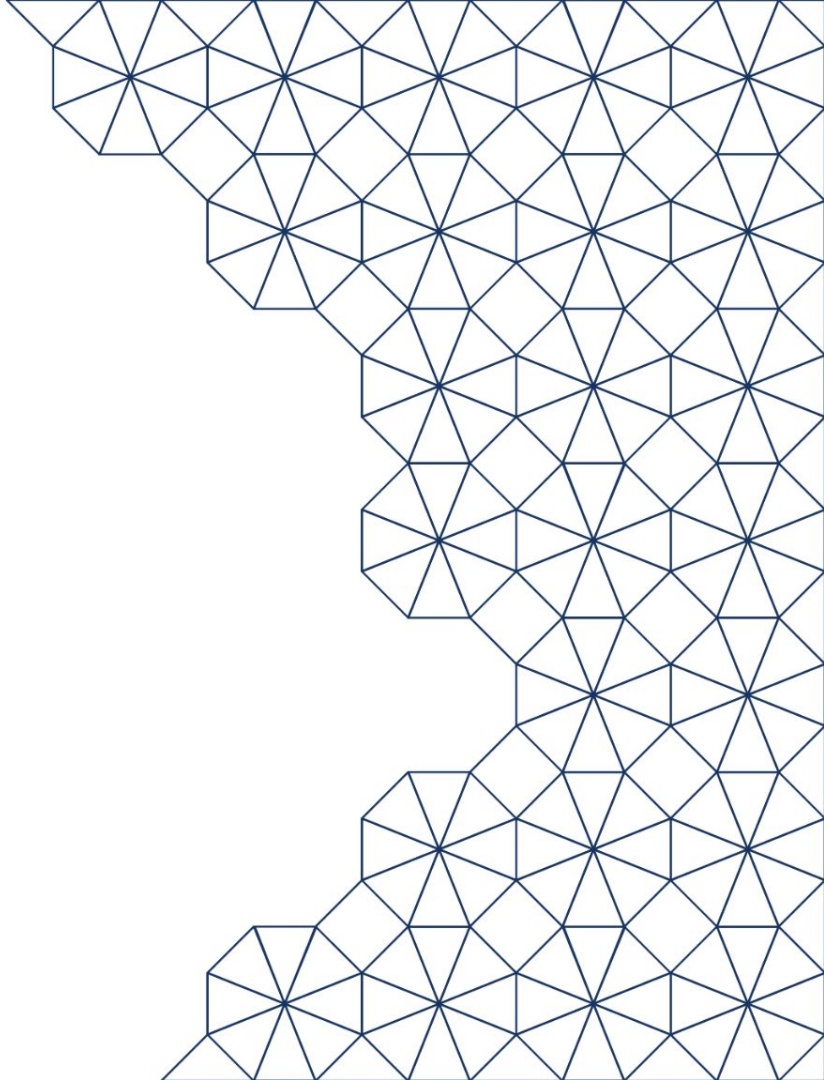
	Pred Healthy	Pred Cancer
True Healthy	612	34
True Cancer	16	718

F1 Score: 0.966
Accuracy: 0.964
True Cancer Detection Rate: 0.978
Predicted Cancer Precision: 0.955
Predicted Healthy Precision: 0.975
False Positive Rate: 0.053
Preprocessing and Prediction Time: 38 s



Test Images Before and After Preprocessing

Linear Models



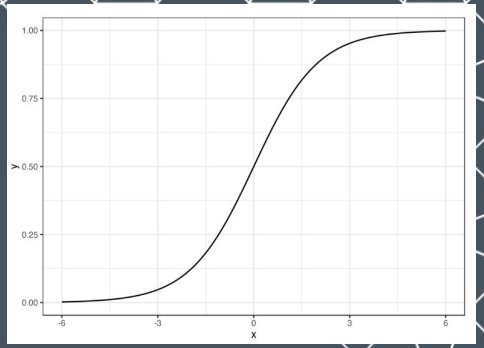
Logistic Regression

Fundamentals: Supervised ML algorithm used for classification problems. Models the probability of possible outcomes.

	Logistic Regression
Hyperparameters	C=0.001, l2, lbfgs
Preprocessing	Standardize(256,256) Max pool (2,2)
Cross Validated F1 Score	0.945
Model Size	132k

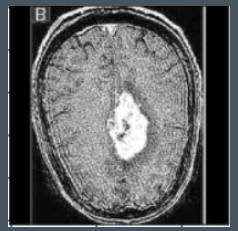
$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$

Logistic Function

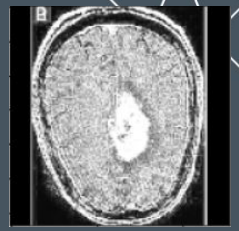


Logistic Regression Evaluation on Test Data

	Pred Healthy	Pred Cancer
True Healthy	596	50
True Cancer	23	711



Original



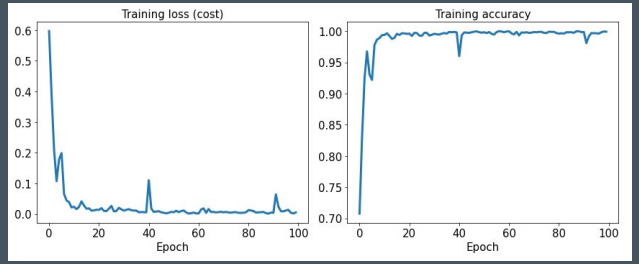
post-preprocessing

F1 Score: 0.947
Accuracy: 0.951
True Cancer Detection Rate: 0.969
Predicted Cancer Precision: 0.934
Predicted Healthy Precision: 0.963
False Positive Rate: 0.077

Multilayer Perceptron NN

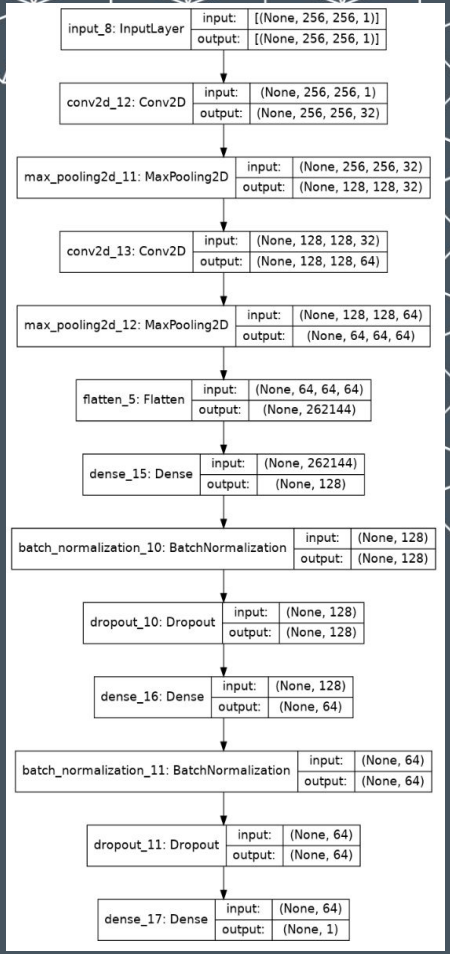
Model 1: Tensorflow, Keras

	MLP NN
Optimizer Epochs Loss	Adam(LR=0.001) 100 BinaryCrossEntropy
Preprocessing	Standardize(256,256)
Activation Functions	RELU, Sigmoid
Dropout Rate	0.3
Training Accuracy Score	0.989
Training Time	1h 49 mins



Evaluation on Test Data

Accuracy: 0.961
Prediction Time: 6s

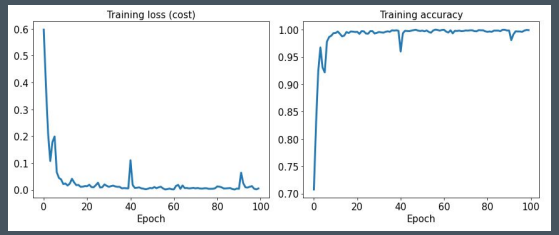


Reference: https://www.kaggle.com/code/boneacrabonjac/brain-tumor-classification-with-simple-cnn

Multilayer Perceptron NN

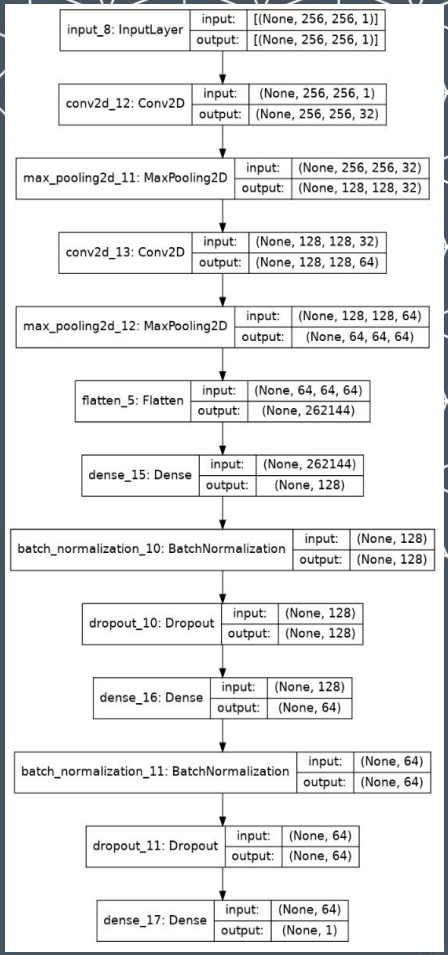
Model 1: Tensorflow, Keras

Optimizer	Adam(0.001)	SGD
Epochs	100	100
Loss	BinaryCrossEntropy	BinaryCrossEntropy
Preprocessing	Standardize(256, 256)	Standardize(256, 256)
Activation Functions	RELU, Sigmoid	RELU, Sigmoid
Dropout Rate	0.3	0.3
Avg. Training Accuracy	0.989	
Training Time	1h 49 mins	
Test Accuracy	0.961	



Adam

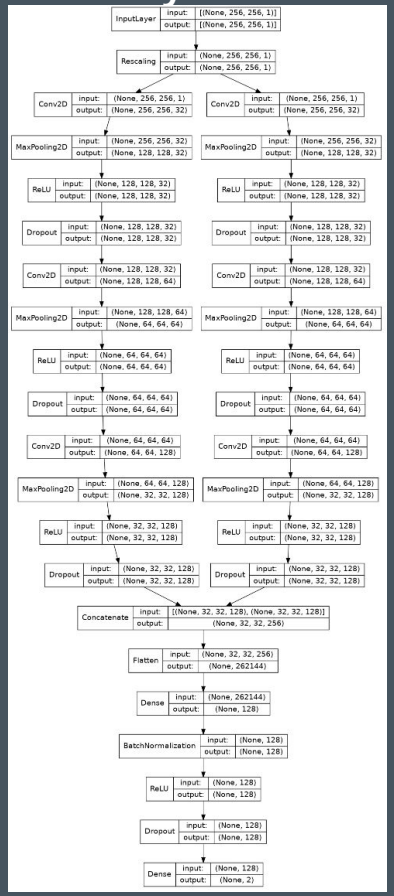
SGD

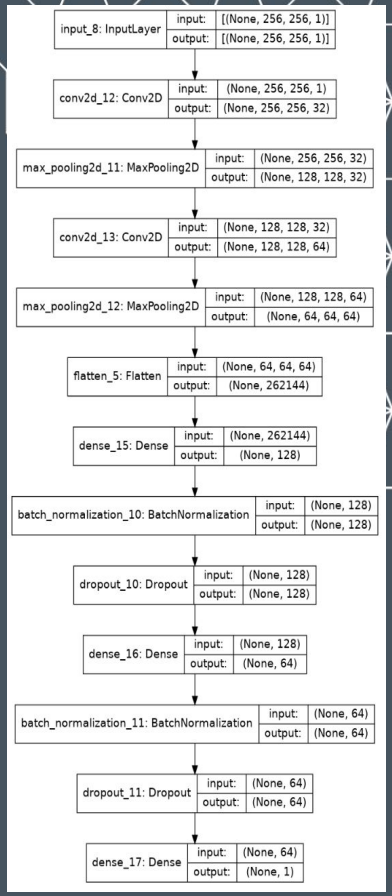


Reference: https://www.kaggle.com/code/boneacrabonjac/brain-tumor-classification-with-simple-cnn

Multilayer NN

Two CNN Models constructed and tested





Reference: https://www.kaggle.com/code/fconcas/cnn-for-the-brain-tumor-dataset, https://www.kaggle.com/code/boneacrabonjac/brain-tumor-classification-with-simple-cnn

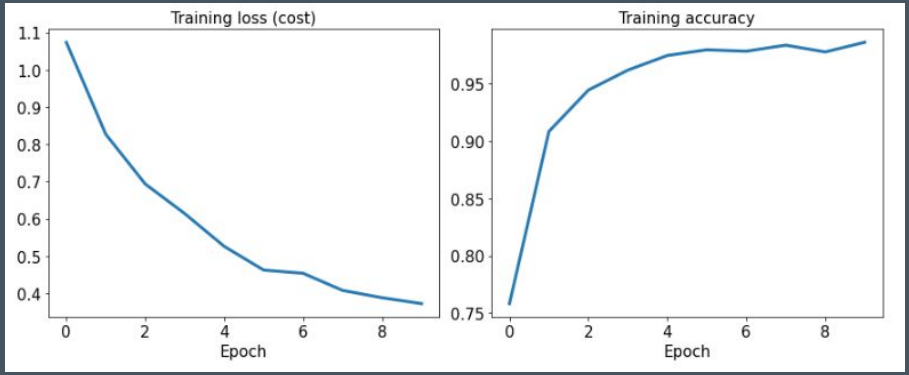
Multilayer NN

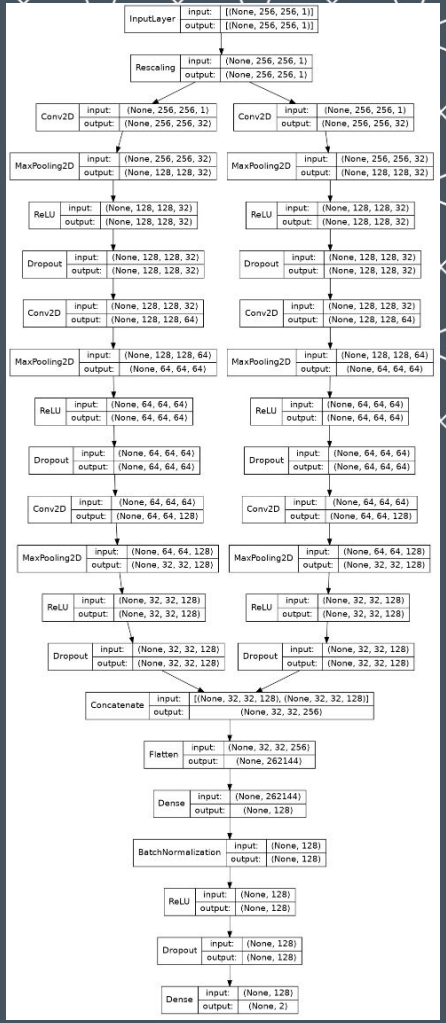
Model 1:
Optimizer: Adam (learning rate: 5e-4)
Loss: SparseCategoricalCrossentropy

Epoch: 10

Avg. Training Accuracy: 0.986

Test Accuracy: 0.875





Reference: https://www.kaggle.com/code/fconcas/cnn-for-the-brain-tumor-dataset

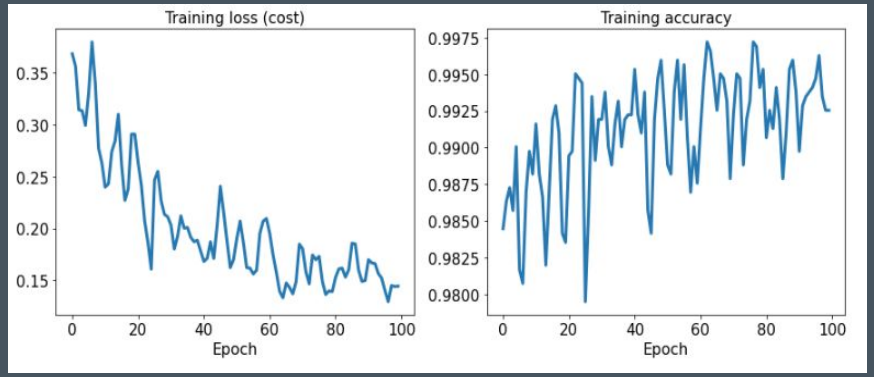
Multilayer NN

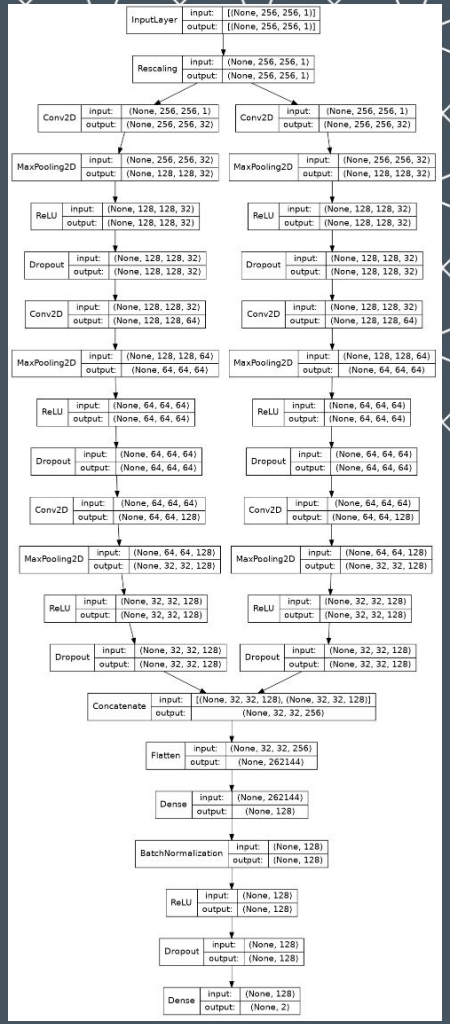
Model 1:
Optimizer: Adam (learning rate: 5e-4)
Loss: SparseCategoricalCrossentropy

Epoch: 100

Avg. Training Accuracy: 0.993

Test Accuracy: 0.968





Reference: https://www.kaggle.com/code/fconcas/cnn-for-the-brain-tumor-dataset

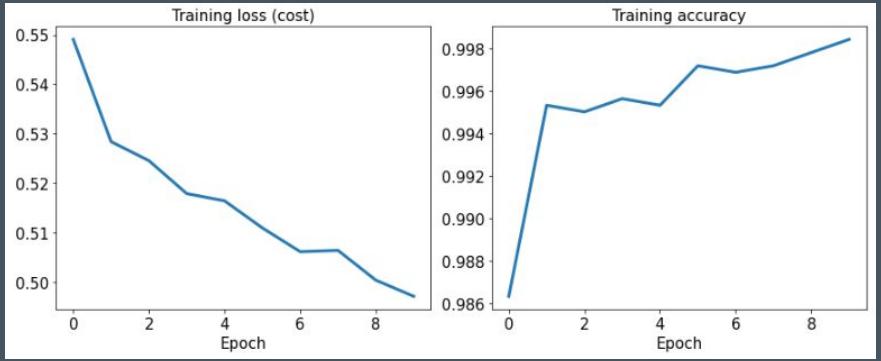
Multilayer NN

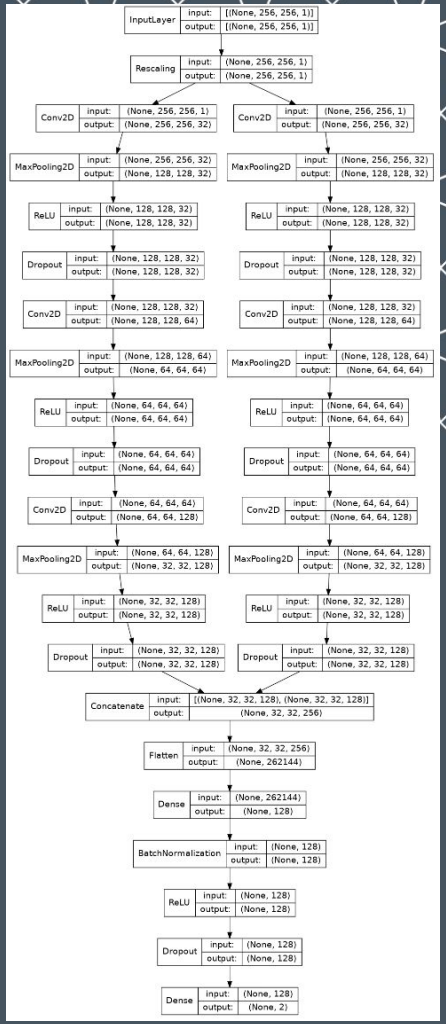
Model 1:
Optimizer: SGD
Loss: SparseCategoricalCrossentropy

Epoch: 10

Avg. Training Accuracy: 0.998

Test Accuracy: 0.938





Reference: https://www.kaggle.com/code/fconcas/cnn-for-the-brain-tumor-dataset

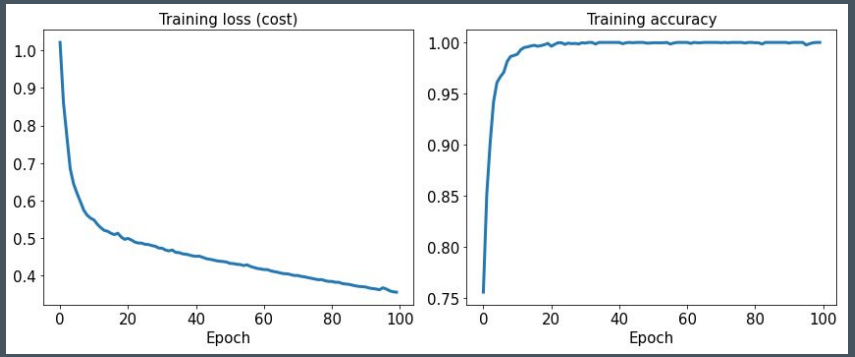
Multilayer NN

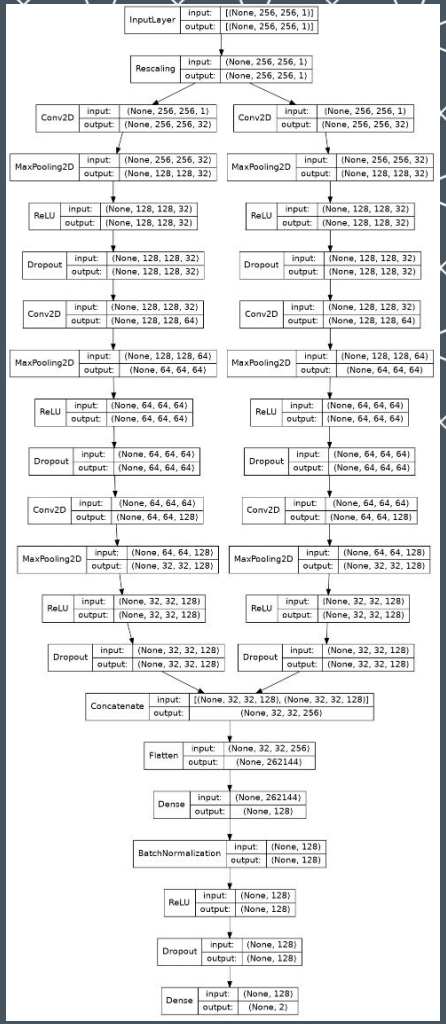
Model 1:
Optimizer: SGD
Loss: SparseCategoricalCrossentropy

Epoch: 100

Avg. Training Accuracy: 0.992

Test Accuracy: 0.953





Reference: https://www.kaggle.com/code/fconcas/cnn-for-the-brain-tumor-dataset

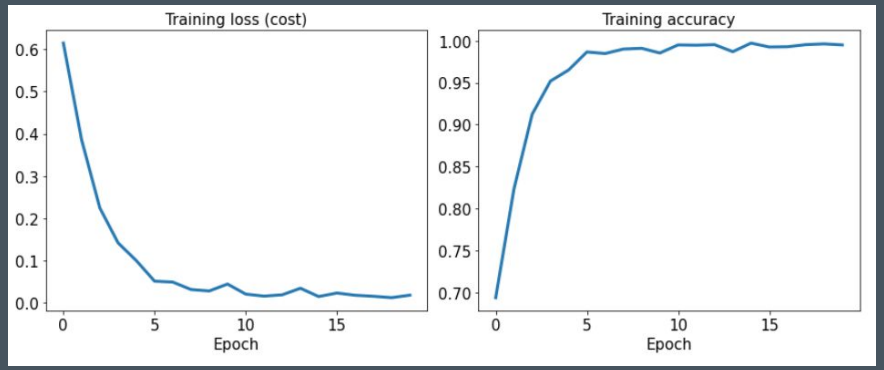
Multilayer NN

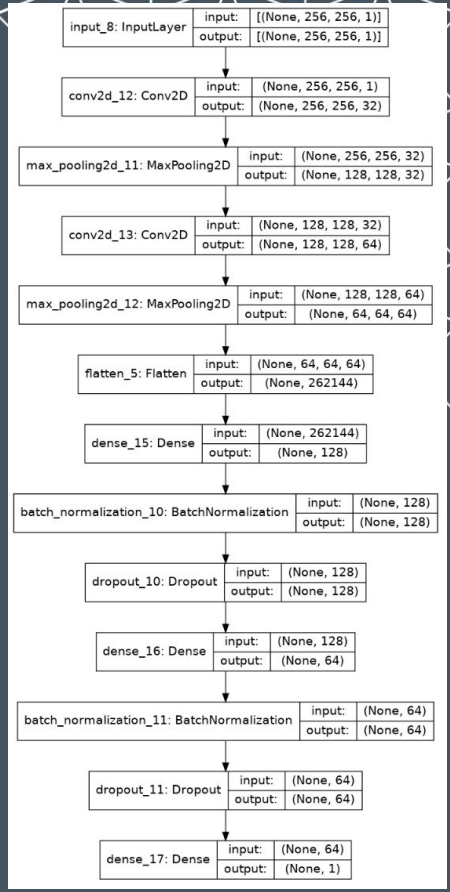
Model 2:
Optimizer: Adam (Learning Rate=0.01)
Loss: BinaryCrossentropy

Epoch: 20

Avg. Training Accuracy: 0.961

Test Accuracy: 0.958



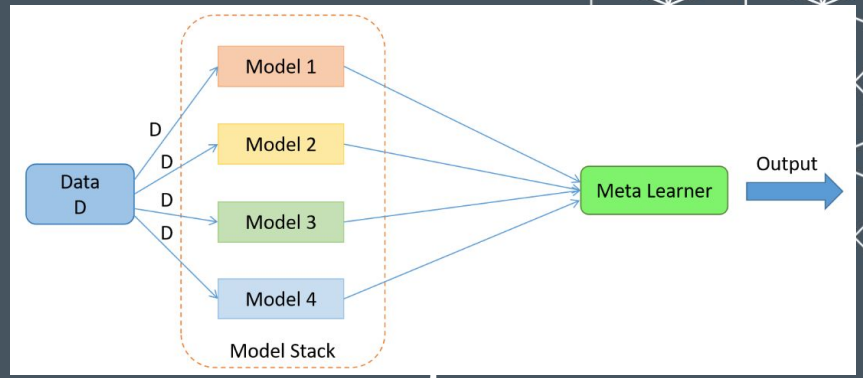


Reference: https://www.kaggle.com/code/boneacrabonjac/brain-tumor-classification-with-simple-cnn

Stacked Ensemble

Stacked Ensemble Model

- Each of our optimal model's learned different decision boundaries in the dataset. While some models generalized better than others, that does not make learned boundaries and performance of other models unusable
- This introduces the question of how to best combine the predictions of multiple models together to deliver even more optimized results



<u>Option 1: Voting Classifier</u>

- Classifications are made according to the majority vote of multiple classifier
- Issue: Weights have to be specified manually if different models should have different priority towards determining final classification

<u>Option 2: Stacked Ensemble</u>

- Train a machine learning model that learns how to best combine predictions from previous layer models
- "Predict labels given predictions from prior models"

Our Optimal Stack

- Training Data for Final Layer Model:
  - Features = Out of sample predicted probabilities of tumor class for each observation in our training dataset, obtained via 5 fold Cross Validation, 1 column per model, 4 columns total
  - Labels = Original Training Labels
- Captures general behavior of each model (what each model would have predicted for our training dataset) alongside the true labels
- Used 5 fold Cross Validation to develop an optimal model for predicting labels given prior predictions
  - Final Layer Model: Gaussian Naive Bayes
  - Directly learned the probability that each model made certain predictions given each label

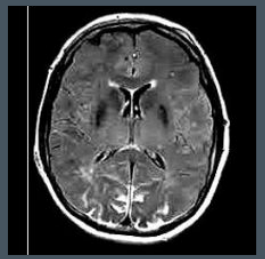
	Final Layer Gaussian NB
Hyperparameters	Var_smoothing = 0.1
Threshold	0.26
Cross Validated F1 Score	0.98

Stacked Model Evaluation on Test Data →

	Pred Healthy	Pred Cancer
True Healthy	626	20
True Cancer	8	726

F1 Score: 0.981
Accuracy: 0.98
True Cancer Detection Rate: 0.989
Predicted Cancer Precision: 0.973
Predicted Healthy Precision: 0.987
False Positive Rate: 0.031

Visual Representation of Stacked Model



Execute optimal preprocessing for each model

Layer 1 Gaussian NB

Layer 1 KNN

Layer 1 GBCT

Layer 1 Logistic Regression

0.99

0.00

0.24

0.00

P(Y = Cancer | Image Vector X) for Each Model

Final Layer Gaussian NB

0.00

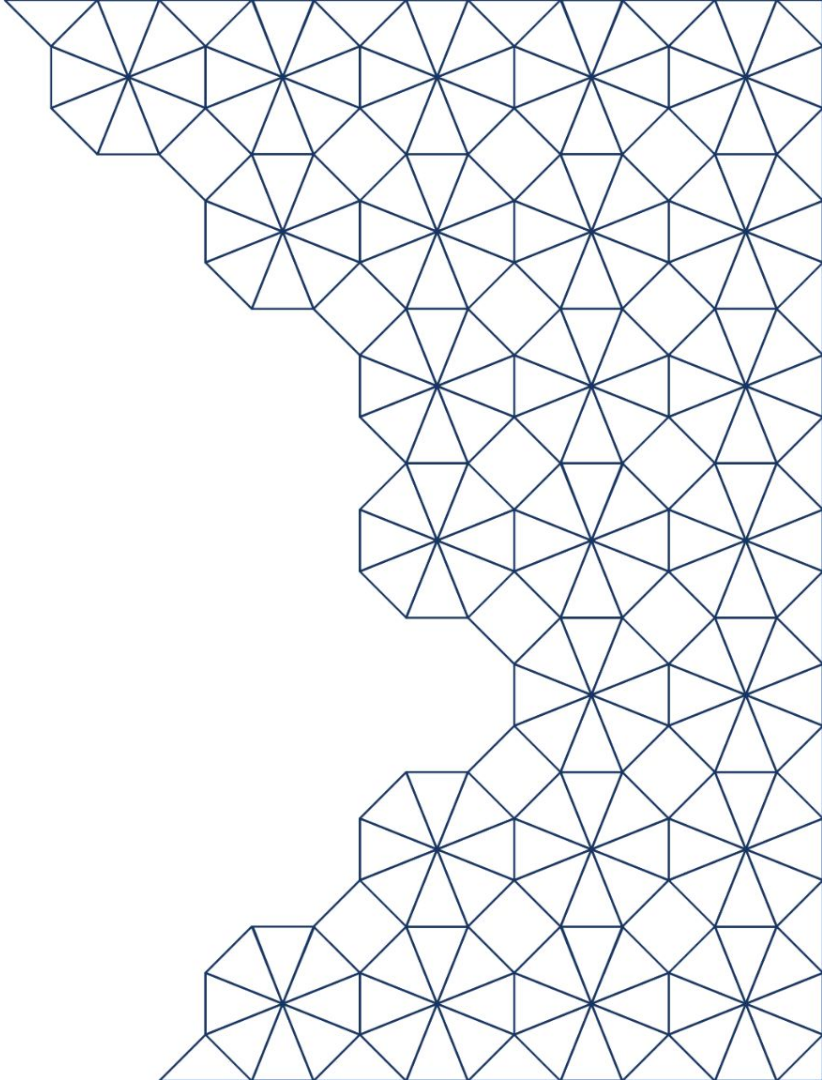
P(Y = Cancer | Predictions from Layer 1 Model)

>= 0.26

False

Healthy

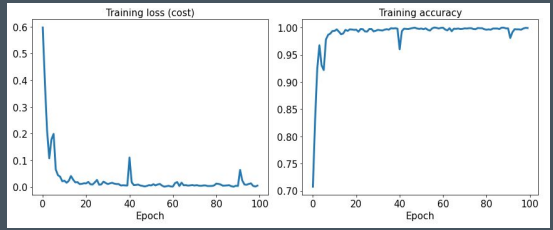
Network Models



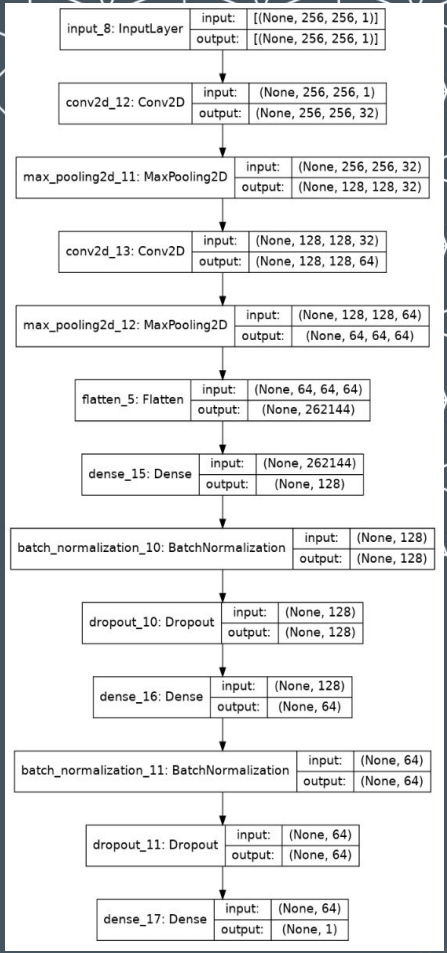
Multilayer Perceptron NN

Model 1: Tensorflow, Keras

Optimizer	Adam(0.001)
Epochs	100
Loss	BinaryCrossEntropy
Preprocessing	Standardize(256,256)
Activation Functions	RELU, Sigmoid
Dropout Rate	0.3
Avg. Training Accuracy	0.989
Training Time	1h 49 mins
Test Accuracy	0.961



Training Loss/Accuracy over Epochs

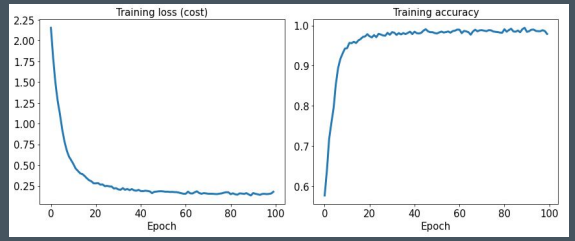


Reference: https://www.kaggle.com/code/boneacrabonjac/brain-tumor-classification-with-simple-cnn

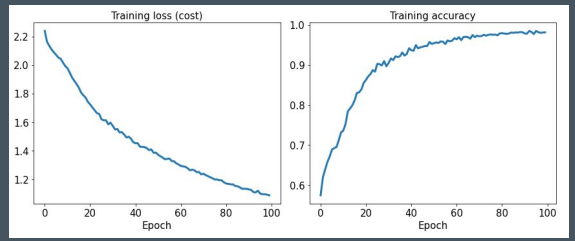
Multilayer Perceptron NN

Model 2: Tensorflow, Keras

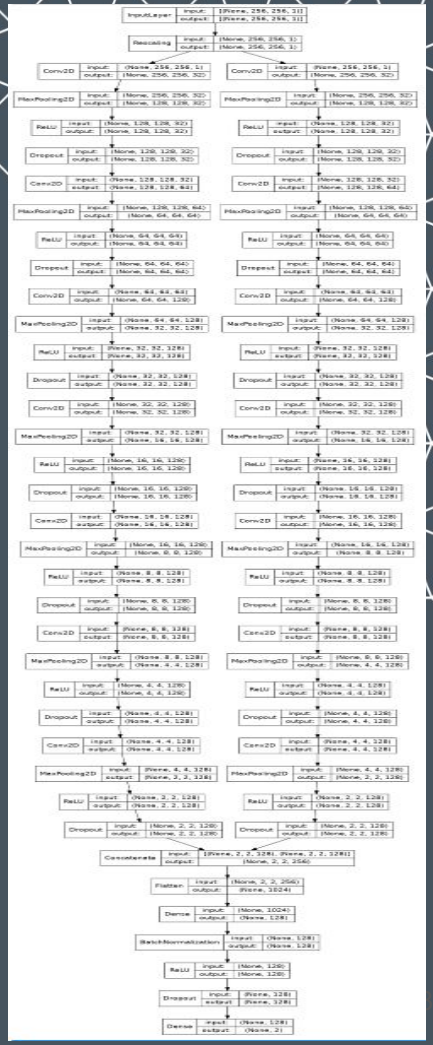
Optimizer	Adam(0.0005)	SGD
Epochs	100	100
Loss	SparseCategorical Crossentropy	SparseCategorical Crossentropy
Preprocessing	Standardize(256, 256)	Standardize(256, 256)
Activation Functions	RELU	RELU
Dropout Rate	0.2	0.2
Avg. Training Accuracy	0.963	0.91
Training Time	3h 35 mins	3h 37 mins
Test Accuracy	0.944	0.941



Training Loss/Accuracy over Epochs - Adam

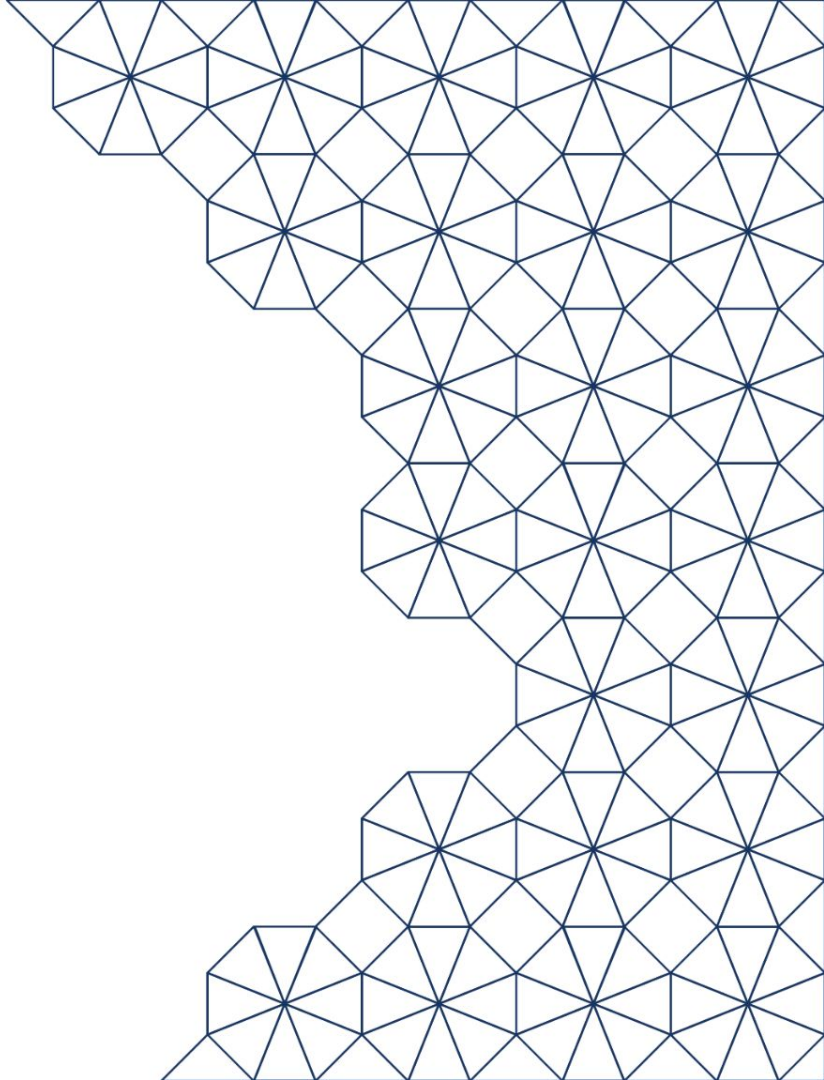


Training Loss/Accuracy over Epochs - SGD



Reference: https://www.kaggle.com/code/fconcas/cnn-for-the-brain-tumor-dataset

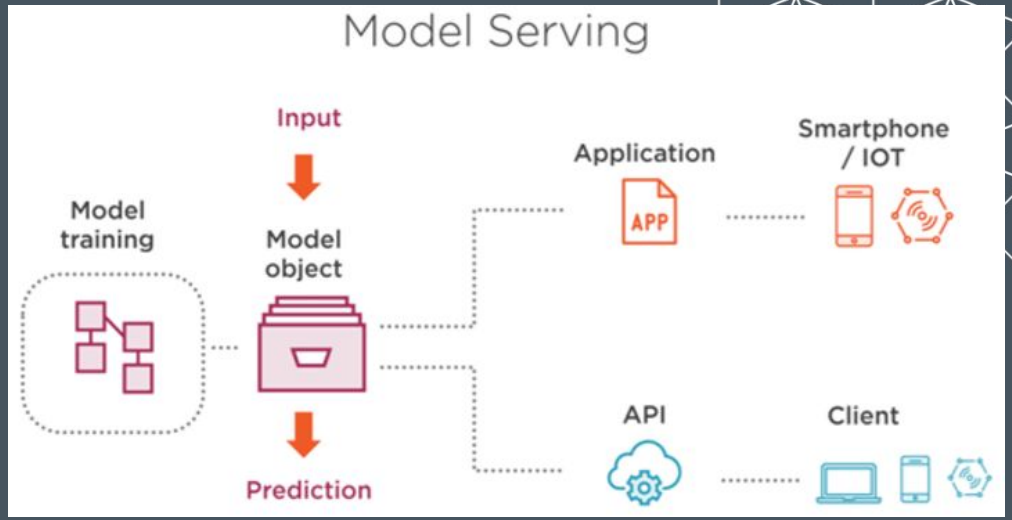
Future Work



Going Forward

Productionizing the Model

- Model Deployment
- Model Re-Training
- Model Maintenance
  - Ongoing updates
  - Experiments
- Auditing
- Versioning
- Monitoring



Business Cases

- Front End Drag and Drop (Open to the public)
- Model as a Service (SaaS -API Call)
- White Label Tool (Build and customize tool for healthcare provider)

References:

https://www.kaggle.com/datasets/preetviradiya/brian-tumor-dataset
https://www.kaggle.com/code/boneacrabonjac/brain-tumor-classification-with-simple-cnn
https://www.kaggle.com/code/fconcas/cnn-for-the-brain-tumor-dataset
https://christophm.github.io/interpretable-ml-book/logistic.html