

The background features a complex network of thin grey lines and dots, forming a web-like structure. Scattered throughout are various triangles of different sizes and orientations, some with solid black dots at their vertices. The overall aesthetic is minimalist and technical.

sumMRI

Chris Kang, Jiayan Luo, Lena Sauter,
Ryan Newkirk, Stephen Savas

Table of Contents



01

Problem

The current state and implications of tumor diagnosis from Brain MRIs

02


Product Design

Introducing the sumMRI platform for enhanced tumor diagnosis

03

Technical Implementation

Deep dive into front-end, back-end, and ML model implementation



01

Problem


Where are we now?





250,000 Deaths Annually

Brain tumors are the 10th leading cause of death for humans, with over 250,000 dying from brain tumors annually ^[1]



Diagnosis Inefficiencies

Research

Diagnostic
error rates are
stuck at 5%^[3]

**Early assisted diagnosis
can improve recovery
rates by 400%^[2]**

Errors

40 million
diagnostic
errors annually^[4]

Burnout

Radiologists
spend 3-4
seconds/image^[5]



02

sumMRI

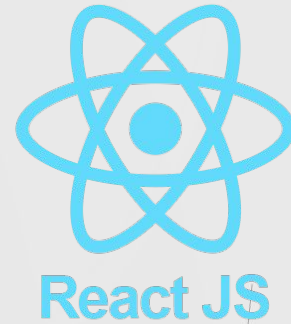
Full-stack ML platform to diagnose brain tumors

Demo: Tumor Found



Video Demo goes here

Front-End

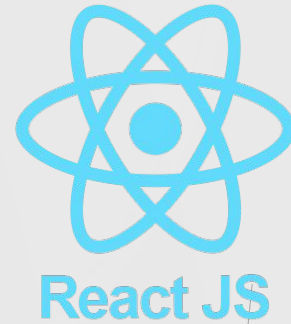


Demo: Tumor Not Found



Video Demo goes here

Front-End



Model Performance

126

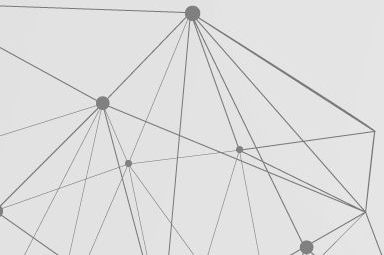
Images Tested On

0.65

Loss

83.33%

Accuracy



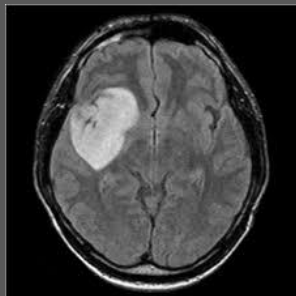


03

CNN CLASSIFICATION MODEL

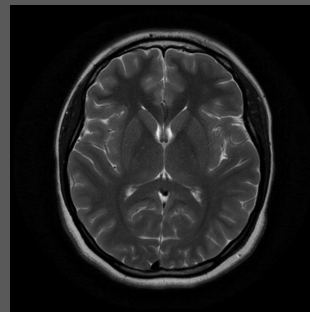
CLASSIFICATION

Input:



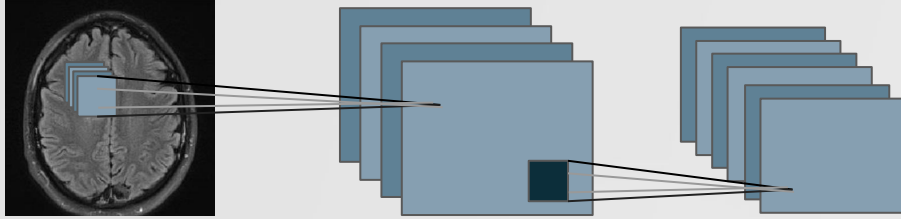
Output: 1

Input:



Output: 0

Convolutional Neural Network



INPUT IMAGE

Takes an MRI scan of a brain as input

FEATURE MAPS

Identify relevant features



POOLING

Abstract away irrelevant features



1 0

FLATTENING

Produce a 1d array of numbers

VALIDATION

Check if model works for new, unseen data

CLASSIFICATION

Outputs 1 if tumor is present, 0 if not

Building and Training a Neural Net

```
# creates model for exactly one input and one output
model = Sequential()
# layer that normalizes inputs
model.add(BatchNormalization(input_shape = (28,28,3)))
# convolution layer - forms a representation of part of an image:
# inputs:
# 32 - filters: the number of output filters in the convolution
# (3, 3) - kernel_size: height, width of convolution window
# activation function. relu = rectified linear unit
model.add(Convolution2D(32, (3,3), activation='relu', input_shape = (28, 28, 3)))
# pooling layer - abstracts away irrelevant parts of the image
# pool_size = size over which to take the max value (2x2)
model.add(MaxPooling2D(pool_size=2))
# padding = 'same' -> even padding to the left/right/up/down of input such that
# the output has the same dimensions as input
model.add(Convolution2D(filters=64, kernel_size=4, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Convolution2D(filters=128, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Convolution2D(filters=128, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
# dropout layer - randomly sets input units to 0 w frequency of rate (0.25)
# at each step during training time
model.add(Dropout(0.25))
# flattening layer - values are compressed into a vector to be processed
model.add(Flatten())
# densely connected NN layer
# units = dimensionality of output space
model.add(Dense(units=128,activation = 'relu'))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dense(units = 32, activation = 'relu'))
model.add(Dense(units = 2, activation = 'softmax'))
```

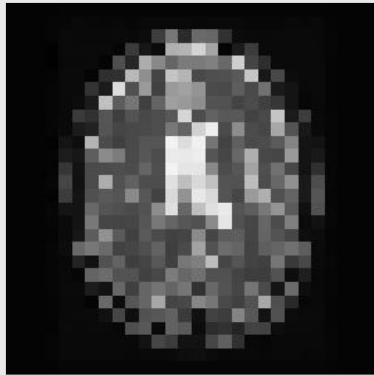
```
model.compile(optimizer='adam', loss = 'categorical_crossentropy',metrics = ['accuracy'])
```

```
history = model.fit(X_train,y_train,
                    epochs=50,
                    validation_data=(X_test,y_test),
                    verbose = 1,
                    initial_epoch=0)

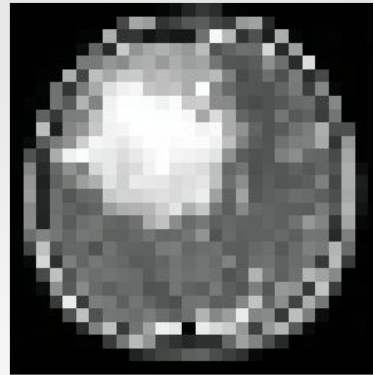
# tensorflow save model
```

CNN Model Output

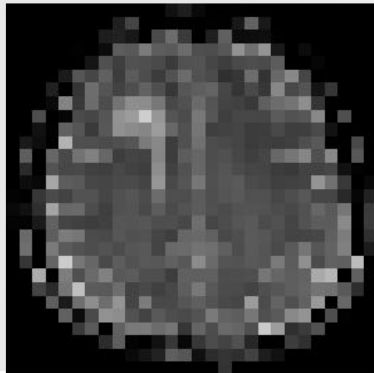
Prediction Class = 0.0
Actual Label = 0.0



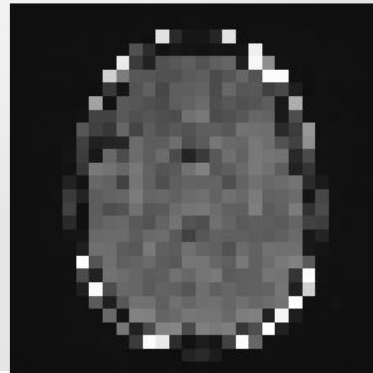
Prediction Class = 1.0
Actual Label = 1.0



Prediction Class = 1.0
Actual Label = 1.0



Prediction Class = 0.0
Actual Label = 0.0



03

Image Segmentation Model



Semantic Segmentation

Use a CNN to classify each pixel in a grid.

Input Image



Semantic Segmentation

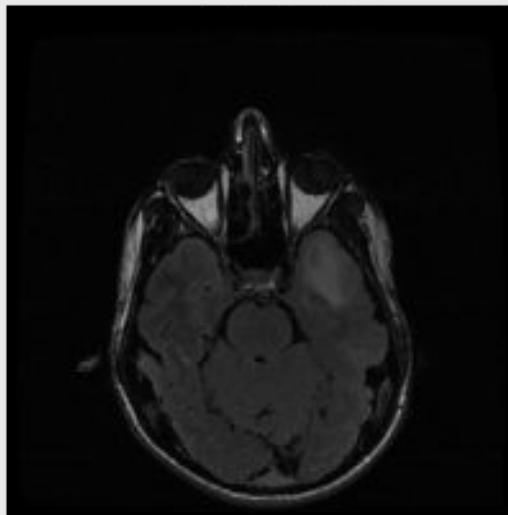


Our Semantic Segmentation

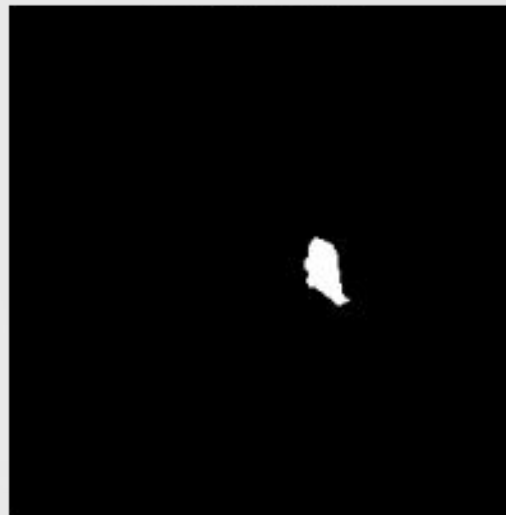
Issue #1: Not a lot of Training Data (relative to Population Needs)

Issue #2: Small Details need to be Segmented

Original Image



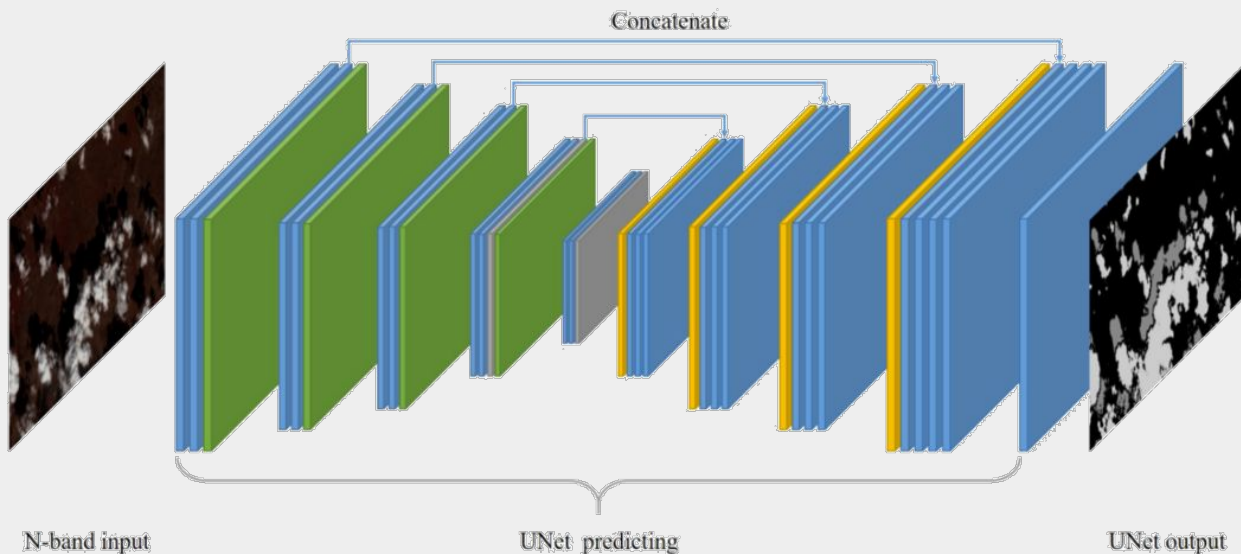
Original Mask



U-Net

Convolutional Network for Biomedical Image Segmentation

- Detects structures with low contrast
- Fast (Model can process an image in 1 second)



U-Net

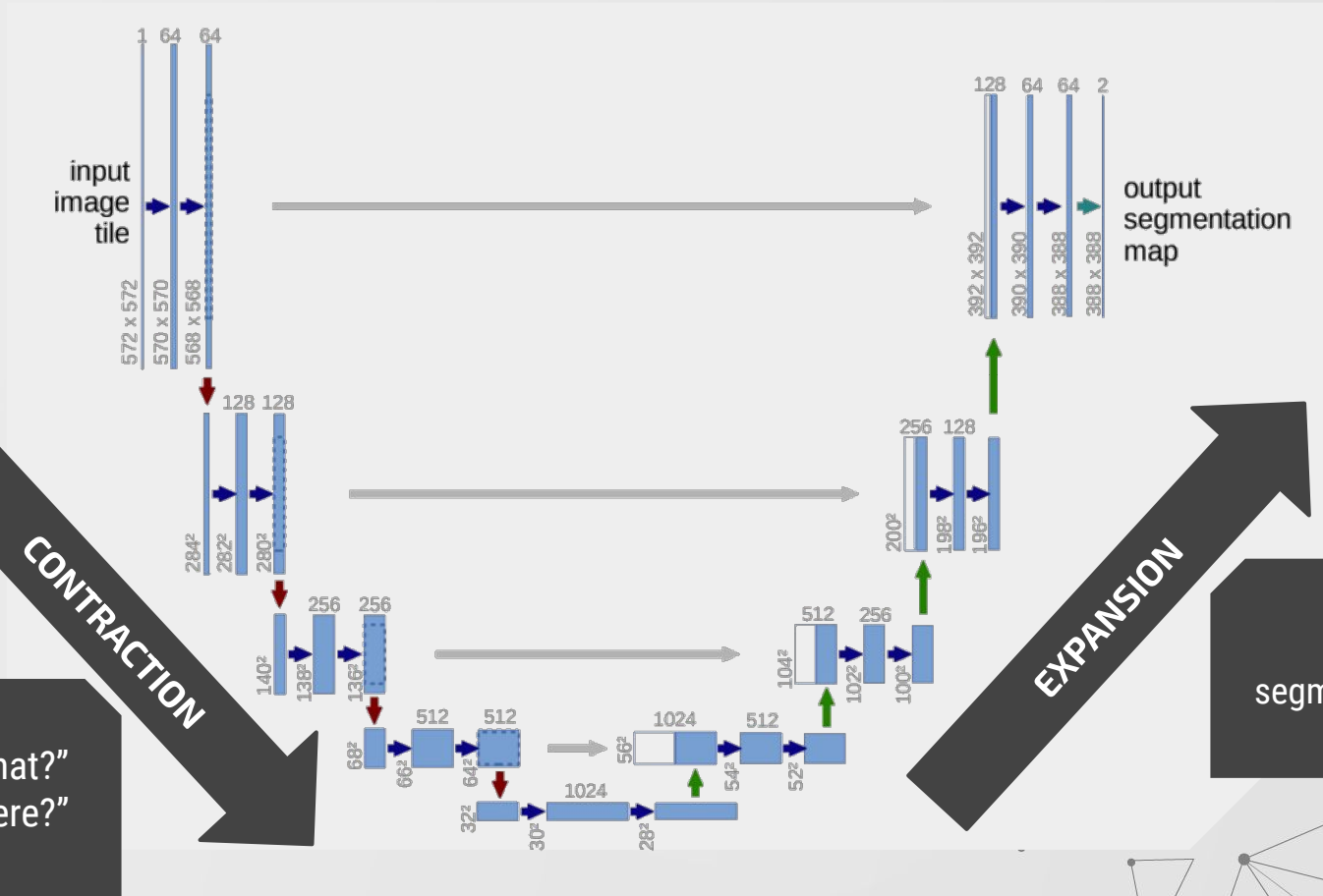
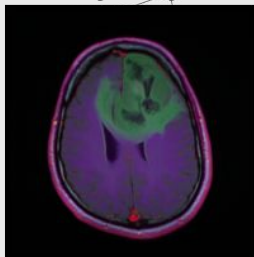


Image Segmentation Model Output

Original Image



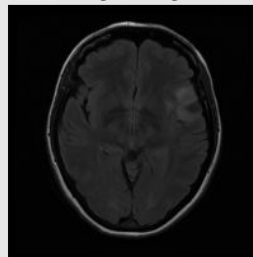
Original Mask



Prediction



Original Image



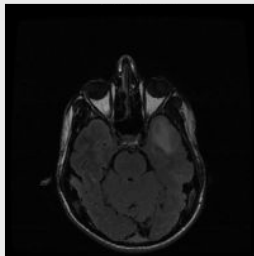
Original Mask



Prediction



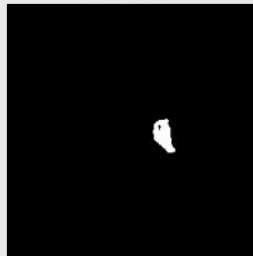
Original Image



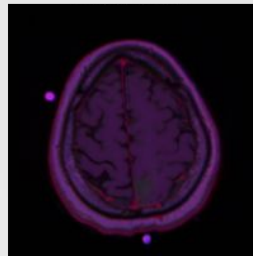
Original Mask



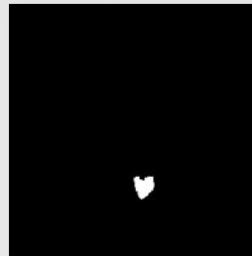
Prediction



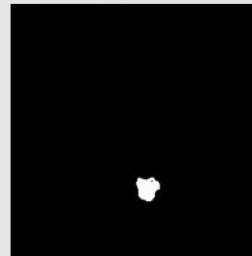
Original Image



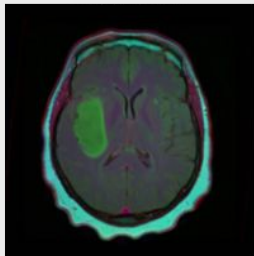
Original Mask



Prediction



Original Image



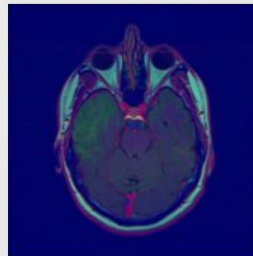
Original Mask



Prediction



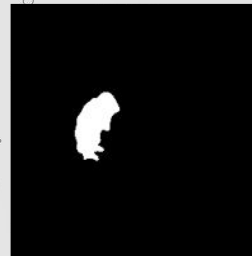
Original Image



Original Mask



Prediction





THANKS

We now open the floor to any questions

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**.

Please keep this slide for attribution.