Detecting Brain Tumors

W207 Final Project
Fall 2022

By Meng-Kang Kao, Zachary Galante, Milan Dean and Kevin Cahillane

GitHub Repository: https://github.com/mkkaoMIDS/w207_brain_tumor

Berkeley
UNIVERSITY OF CALIFORNIA

Motivation

- Brain cancer remains one of the most deadly cancers with a 12.8% 5-year survival rate (“Cancer Survival Rates”).
    - Early and accurate diagnosis is crucial

700,000
AMERICANS
are living with a primary brain tumor

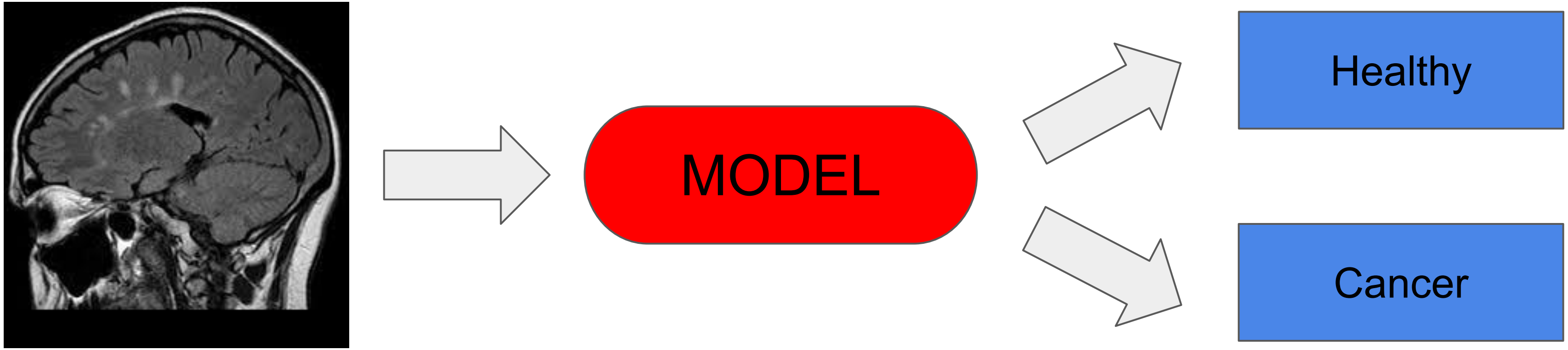
88,970
AMERICANS
will receive a primary brain tumor diagnosis in 2022

36%
RELATIVE SURVIVAL RATE
for all patients with a malignant brain tumor

18,200
AMERICANS
will die from a malignant brain tumor in 2022

Research Question

Can we use machine learning to accurately detect brain tumors in MRI brain scans?



Overall Plan

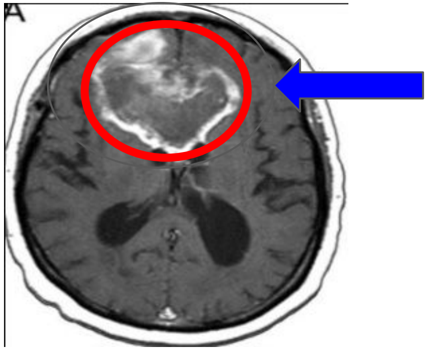
- Use Logistic Regression as baseline model
- Use CNN to Improve prediction accuracy
- While achieving highest possible accuracy, in medical field, recall is equally important. The goal is to find the balance between accuracy and recall.
- Final model performance
  - 96.1% accuracy
  - 99% recall

Berkeley
UNIVERSITY OF CALIFORNIA

Data

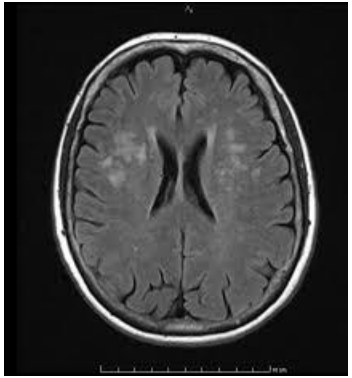
- Kaggle Dataset including 4600 brain images with the binary labels "Healthy" or "Cancer"
  - Original dataset included duplicates
  - Not all images were consistent in size, angle, format

Cancer



Tumor

Healthy



EDA

- Images include JPEG (98%), TIF (1.9%), and PNG (0.4%).
- Image modes include RGB(97%), greyscale (2.9%), and other (< 0.1%).
- Classification balance of images are satisfactory (54% tumor vs. 46% healthy).

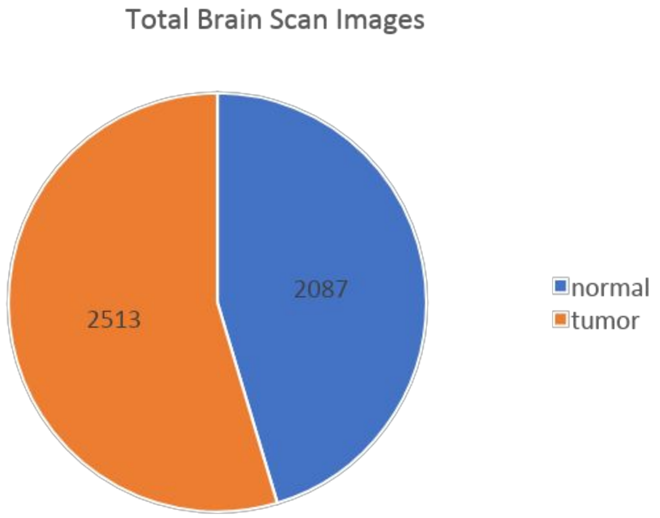
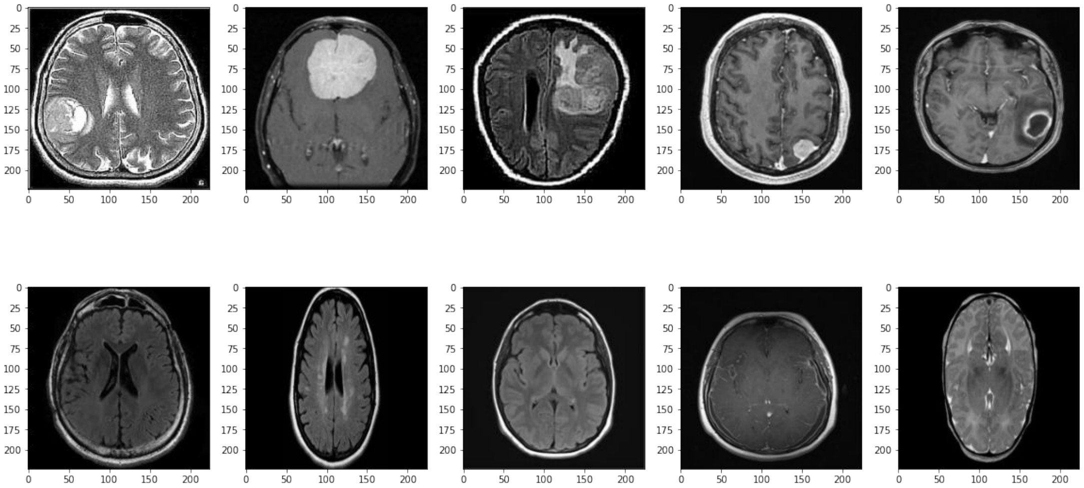


Image Preprocessing

- Convert images from RGB to gray-scale
- Resize all images to 224x224



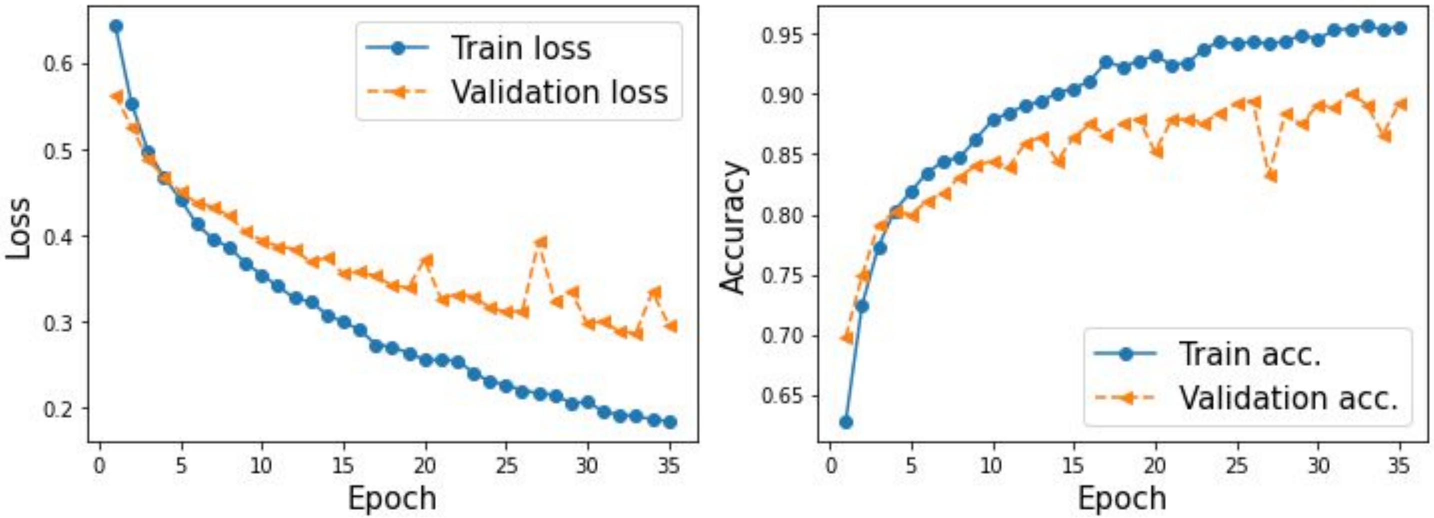
Baseline Logistic Regression Model

```
model.add(tf.keras.layers.Dense(
  units=1,                          # output dim (for binary classification)
  use_bias=True,                    # use a bias param
  activation="sigmoid"
))

# Use the Adam optimizer.
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
```

- 35 Epochs
- Batch Size: 32
- Sigmoid Activation
- .5 threshold classification threshold

Berkeley
UNIVERSITY OF CALIFORNIA

Baseline Results



Baseline Results

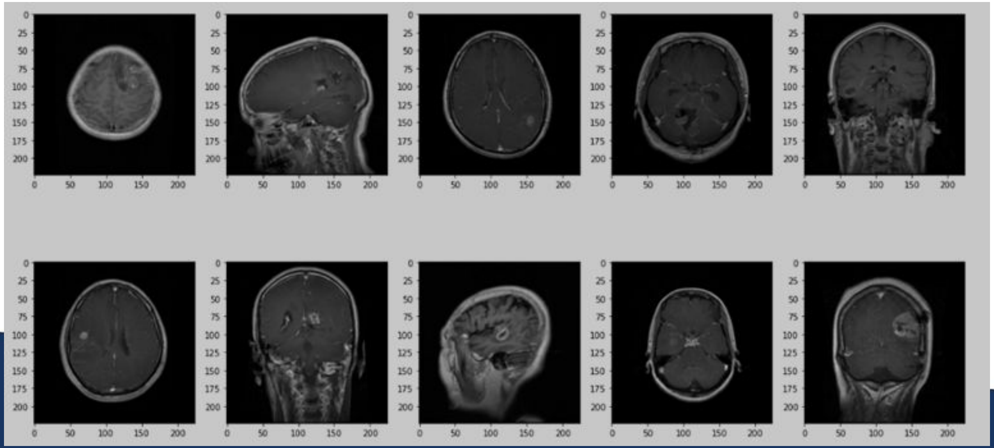
- Optimized learning rate: .0001
- Struggles with generalization
- Epoch 15: 86.41% validation accuracy
- Epoch 35: 89.24% validation accuracy

	loss	binary_accuracy	val_loss	val_binary_accuracy
25	0.219473	0.943478	0.311209	0.894565
26	0.215995	0.942029	0.391359	0.832609
27	0.214230	0.943841	0.323699	0.883696
28	0.204672	0.949275	0.334616	0.875000
29	0.206021	0.945652	0.296731	0.891304
30	0.194942	0.953261	0.299855	0.889130
31	0.190990	0.954348	0.287026	0.901087
32	0.189671	0.956884	0.286809	0.891304
33	0.185663	0.953623	0.333660	0.866304
34	0.183789	0.956159	0.295949	0.892391

Berkeley
UNIVERSITY OF CALIFORNIA

Attempted Deduplication using Unsupervised Learning

- Dataset included different angles of brain images
    - Tried to use kmeans to differentiate between angles
- Original dataset had unknown number of duplicate images
- Ran image dataset through DBScan Clustering algorithms
    - Both had clusters with non-identical images



Berkeley
UNIVERSITY OF CALIFORNIA

Deduplication with Image Hashing

```
# Generate hash for each image.
hashed_images = []

for i in range(len(total_images)):
    hashed_images.append((str(average_hash(array_to_img(total_images[i]), hash_size=64)), i))
```

```
# Load hash code into a dictionary and find unique values.
unique_items = list(dict(hashed_images).values())

len(unique_items)
```

```
3985
```

```
# Filtering the images and labels list containing only the unique items
total_images_deduped = [total_images[idx] for idx in unique_items]
total_labels_deduped = [total_labels[idx] for idx in unique_items]

total_images_deduped = np.array(total_images_deduped)
total_labels_deduped = np.array(total_labels_deduped)
```

- Generate hash code for each image and load hash and index to a dictionary
- Filter image and label datasets to index within dictionary
- Scalable for larger datasets

Berkeley
UNIVERSITY OF CALIFORNIA

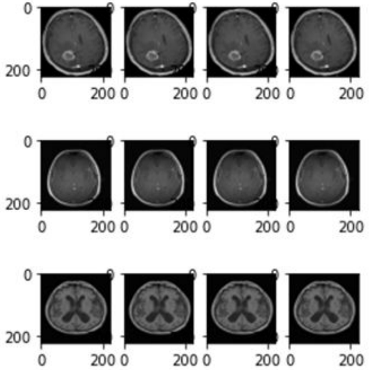
Validate deduplication with DBScan

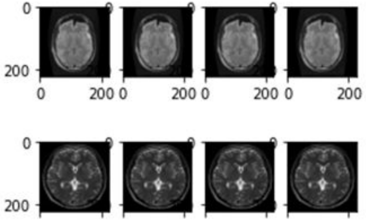
- Used DBScan to verify all detected duplicates were the same image

- **Motivation:** Should return 1 cluster for each group of images

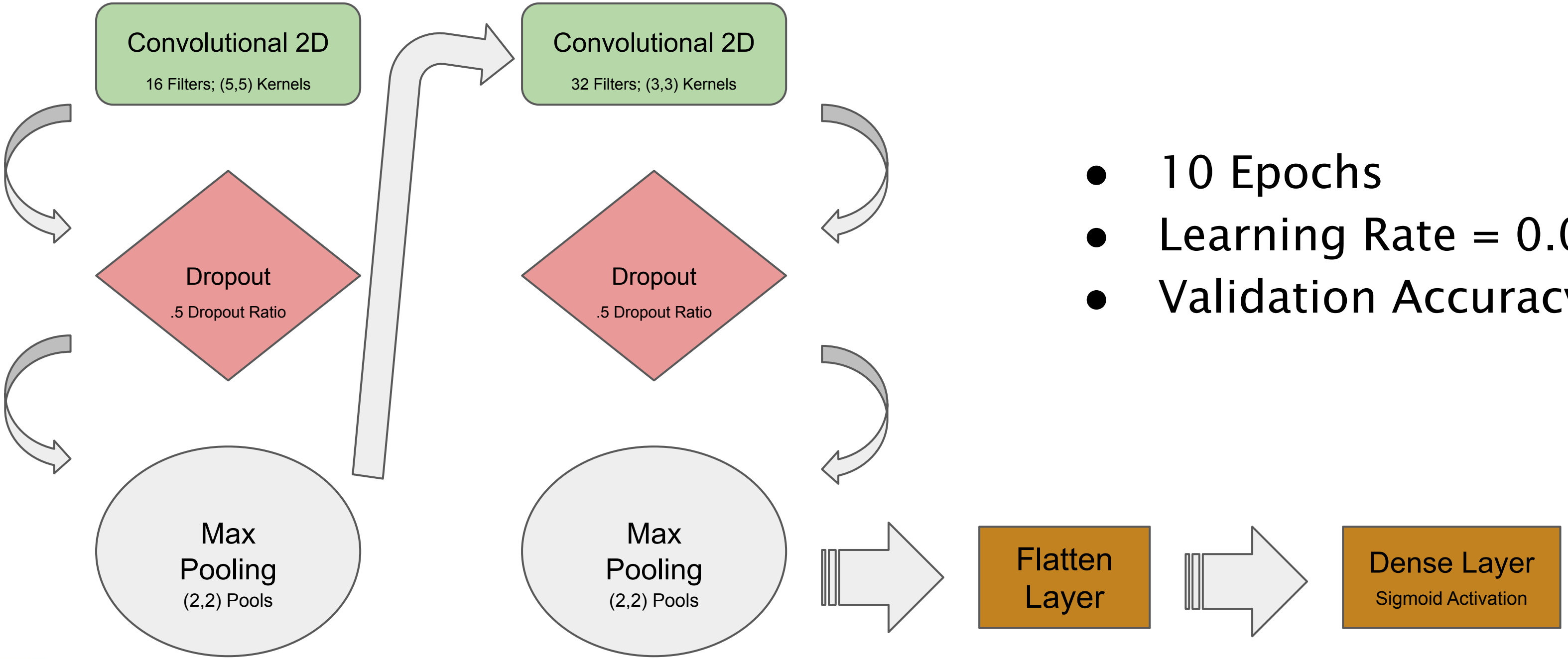
- Minimum of 1 sample for each cluster

```
db = DBSCAN(eps=0.2, min_samples=1, metric='euclidean')
num_of_clusters = []
```



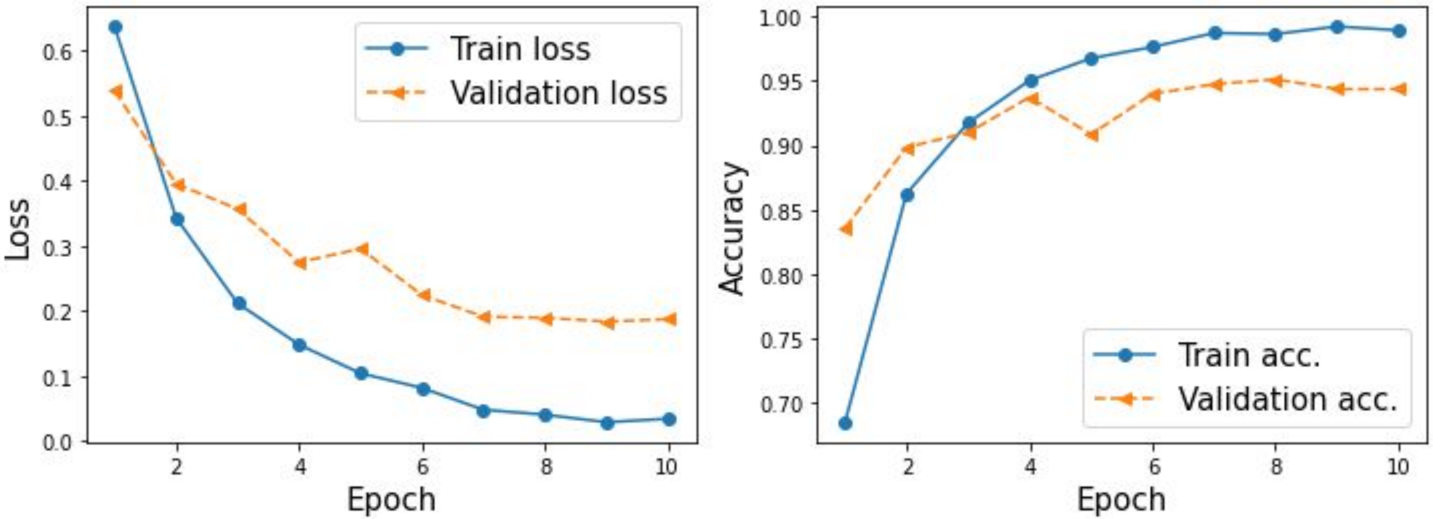


Initial CNN Model



- 10 Epochs
- Learning Rate = 0.001
- Validation Accuracy: 94.3%

Initial CNN Model Results



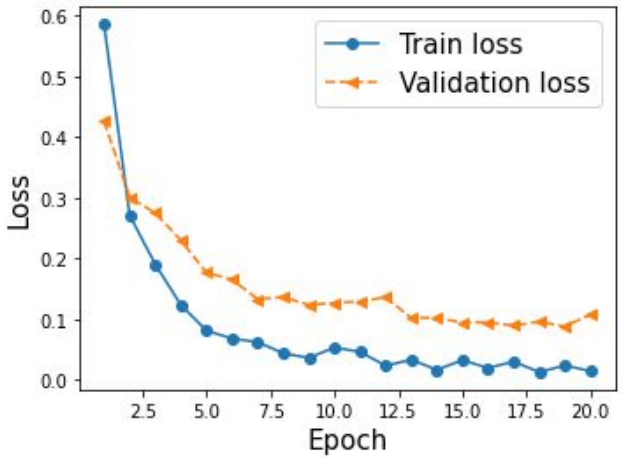
Data Augmentation

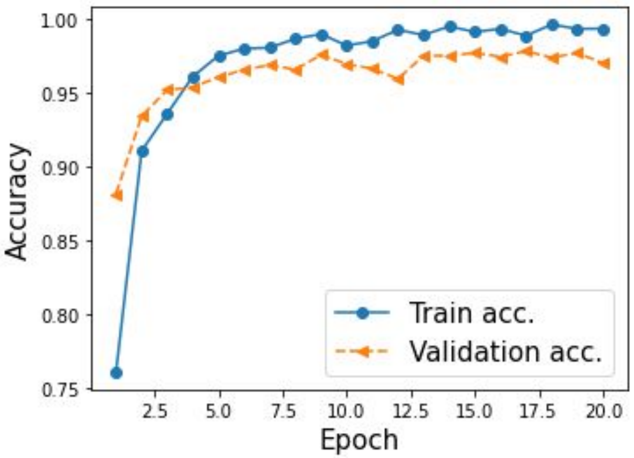
```
# adjust brightness
X_train_augm = tf.image.adjust_brightness(X_train, delta=DELTA)

# adjust contrast
X_train_augm = tf.image.adjust_contrast(X_train_augm, contrast_factor=CONTRAST_FACTOR)

# random flip
X_train_augm = tf.image.random_flip_left_right(X_train_augm)
```

- Duplicates images, transforming brightness, contrast, and orientation
- Training set doubles from 3,188 to 6,376 images

Berkeley
UNIVERSITY OF CALIFORNIA

CNN Results After Augmentation





- Increased to 20 epochs

- 97.05% validation accuracy
- Shrunk gap between train and val loss curves

Berkeley
UNIVERSITY OF CALIFORNIA

Optimized CNN Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 16)      416

 max_pooling2d (MaxPooling2D  (None, 112, 112, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 32)        9248

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 28, 28, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 conv2d_4 (Conv2D)           (None, 14, 14, 128)       73856

 max_pooling2d_4 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 dropout (Dropout)           (None, 7, 7, 128)         0

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 512)               3211776

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 1)                 129

=================================================================
Total params: 3,482,785
Trainable params: 3,482,785
```

- 3.48M Parameters
- 10 Epochs
- Modified Dropout and added 64/128 Conv2D Layers
- Inspired by AlexNet and VGG-16
- Learning Rate = 0.001
- Validation Accuracy: 96.42%

Hyper Parameter Tuning

- Unsupervised Learning - Not used in final model
- Image Deduplication
  - Hash size from 16 to 96, finalized at 64
- CNN Model
  - Learning rate, kernel size, max pooling size, model
  - Epochs finalized at 10
  - Training / Validation Split from 90/10 to 75/25
- Data Augmentation
  - Contrast (3), Delta (0.2)

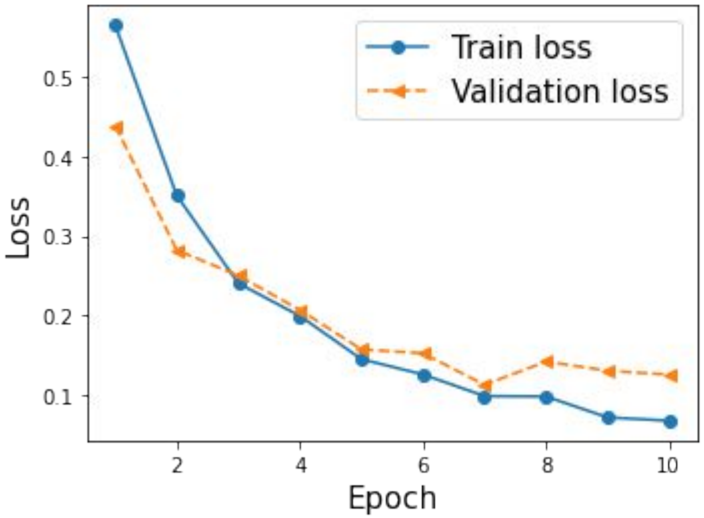
Epoch Tuning

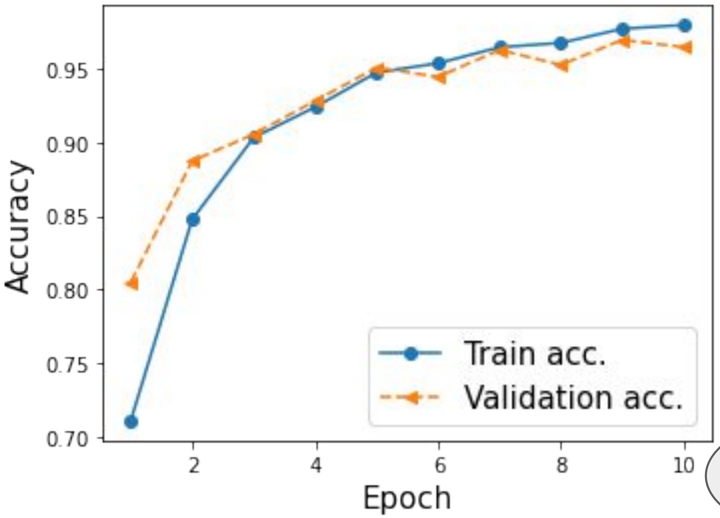
- Reduced epochs to 10
  - Based on where loss curve flattened

	loss	binary_accuracy	val_loss	val_binary_accuracy
0	0.564886	0.711000	0.438439	0.804893
1	0.350659	0.847553	0.282156	0.887077
2	0.240362	0.902969	0.250086	0.905270
3	0.199348	0.923672	0.206845	0.927854
4	0.145057	0.947093	0.157314	0.950439
5	0.125804	0.953367	0.152374	0.944166
6	0.098666	0.964241	0.112832	0.962359
7	0.098107	0.967169	0.141990	0.952321
8	0.071861	0.976788	0.130332	0.969260
9	0.067658	0.979297	0.125660	0.964241

Berkeley
UNIVERSITY OF CALIFORNIA

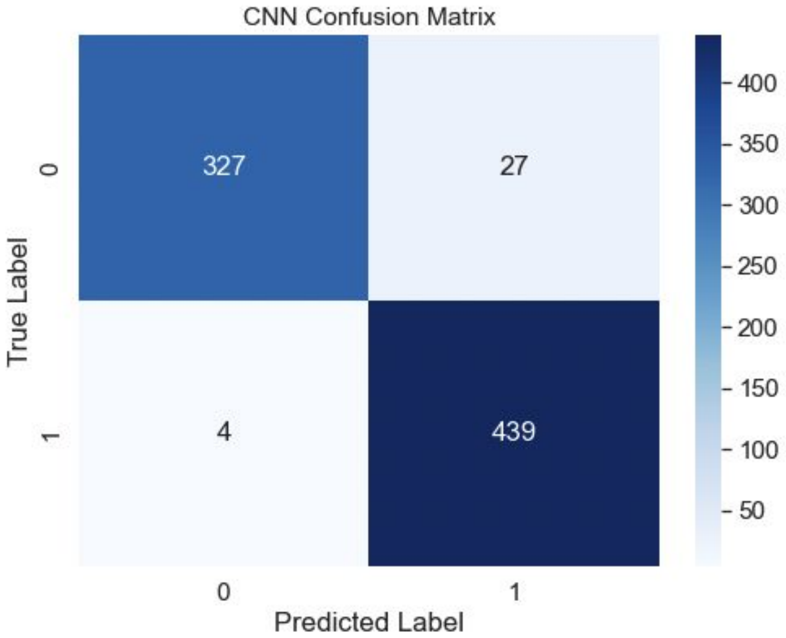
Final model results



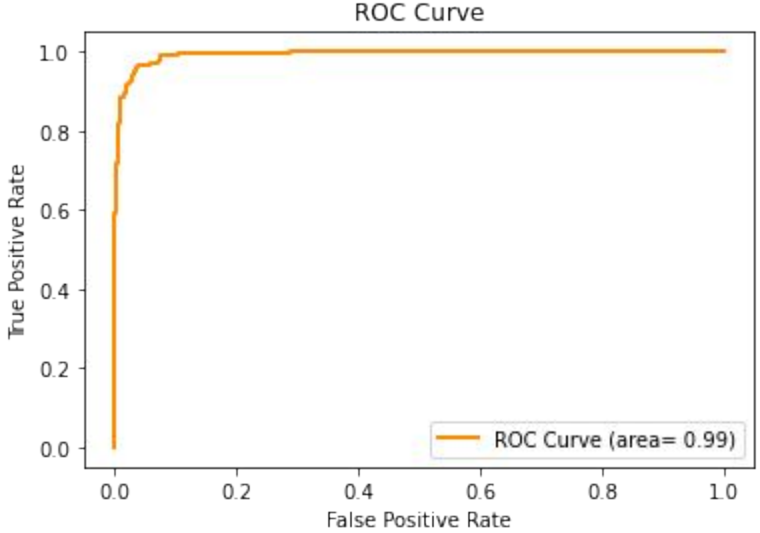


Nice little happy curves :)

Model Evaluation - Test Data



96.1% Accuracy
99% Recall

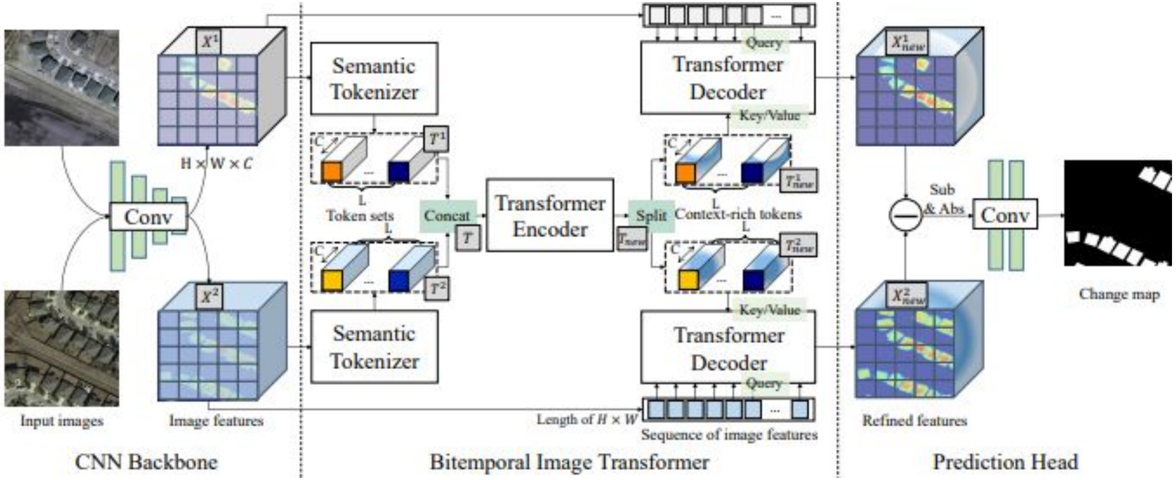


Road Map

Model	# Parameters	# Epochs	Validation Accuracy	Recall
Logistic Regression (Baseline)	50,177	35	89.24%	
Logistic Regression After Deduplication	50,177	35	90.72%	
Initial CNN Model	105,409	10	94.35%	
Initial CNN Model with Data Augmentation	105,409	9	97.61%	94%
Hyper Parameter and Model Tuning	3,482,785	10	96.42%	99%

Future Implementations

- Potential for future tumor detection models to apply transformers to CNN
- BIT could be used for time series analysis of tumor growth and remission
- Opportunities to use transfer learning



Example of a Bitemporal Image Transformer used to detect changes in satellite images (Hao, Zipeng, Zhenwei).

Berkeley
UNIVERSITY OF CALIFORNIA

Conclusion

- Despite a relatively small dataset, the supervised learning models proved effective at identifying brain tumors
- Our most basic logistic regression model had over 89% accuracy
- Final CNN model performed at 96.1% test accuracy
  - Although the final accuracy is slightly lower than previous model, but we believe by getting recall at 99% is more important.
  - And there are pathways to improve even further
- These kinds of models that detect tumors with accuracy can take pressure off the healthcare system, enable those in need to receive treatment, and help save lives.

Berkeley
UNIVERSITY OF CALIFORNIA

Works Cited

“Cancer Survival Rates.” *Nuffield Trust*, www.nuffieldtrust.org.uk/resource/cancer-survival-rates, 11/10/22.

Hao Chen and Zipeng Qi and Zhenwei Shi. “Remote Sensing Image Change Detection With Transformers.” *IEEE,* Transactions on Geoscience and Remote Sensing, 2022, 1 - 14.

Viradiya, Preet. “Brian Tumor Dataset,” https://www.kaggle.com/datasets/preetviradiya/brian-tumor-dataset, 09/04/22.

All images from Google Images

Berkeley
UNIVERSITY OF CALIFORNIA

Thank you!

Team Contribution

	Research Question	Data Pre-processing	Baseline Model	Unsupervised Learning	Data Augmentation	CNN Architecture	Hyper Parameter Tuning	Presentation Slides
Meng-Kang Kao	X	X	X	X	X	X	X	X
Zachary Galante	X		X	X		X	X	X
Milan Dean	X	X	X			X	X	X
Kevin Cahillane	X		X	X			X	X

Berkeley
UNIVERSITY OF CALIFORNIA