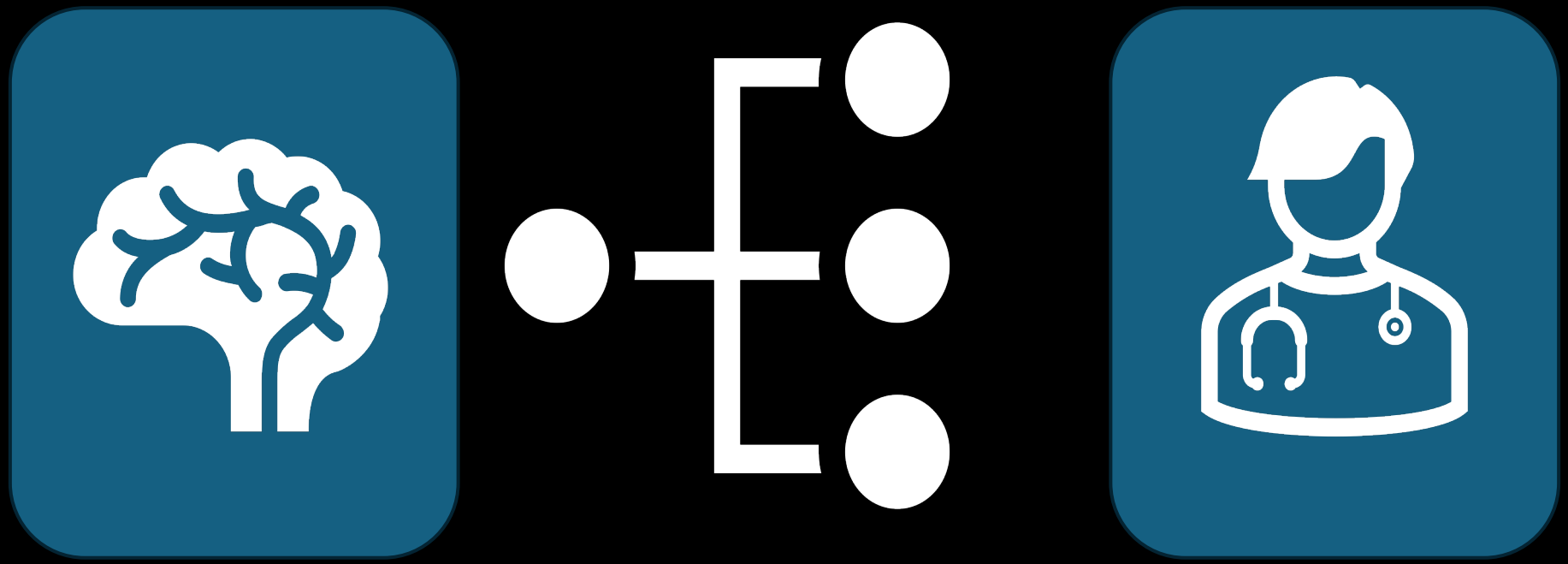
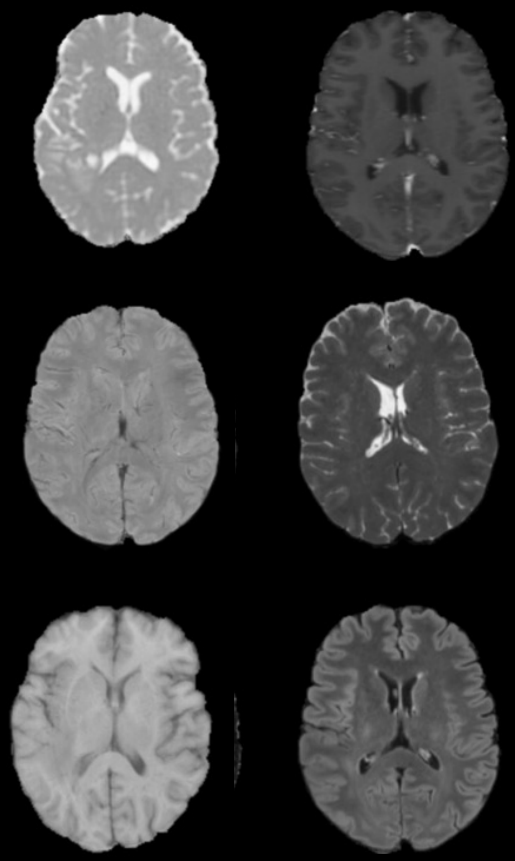
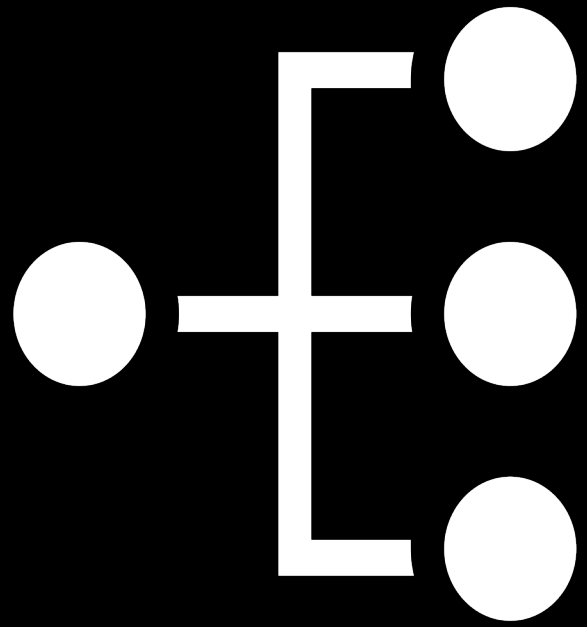
Classifying Multimodal Brain MRI Using a 2D Convolutional Neural Network

James M. Holcomb
University of Texas at Austin, University of Texas Southwestern Medical Center



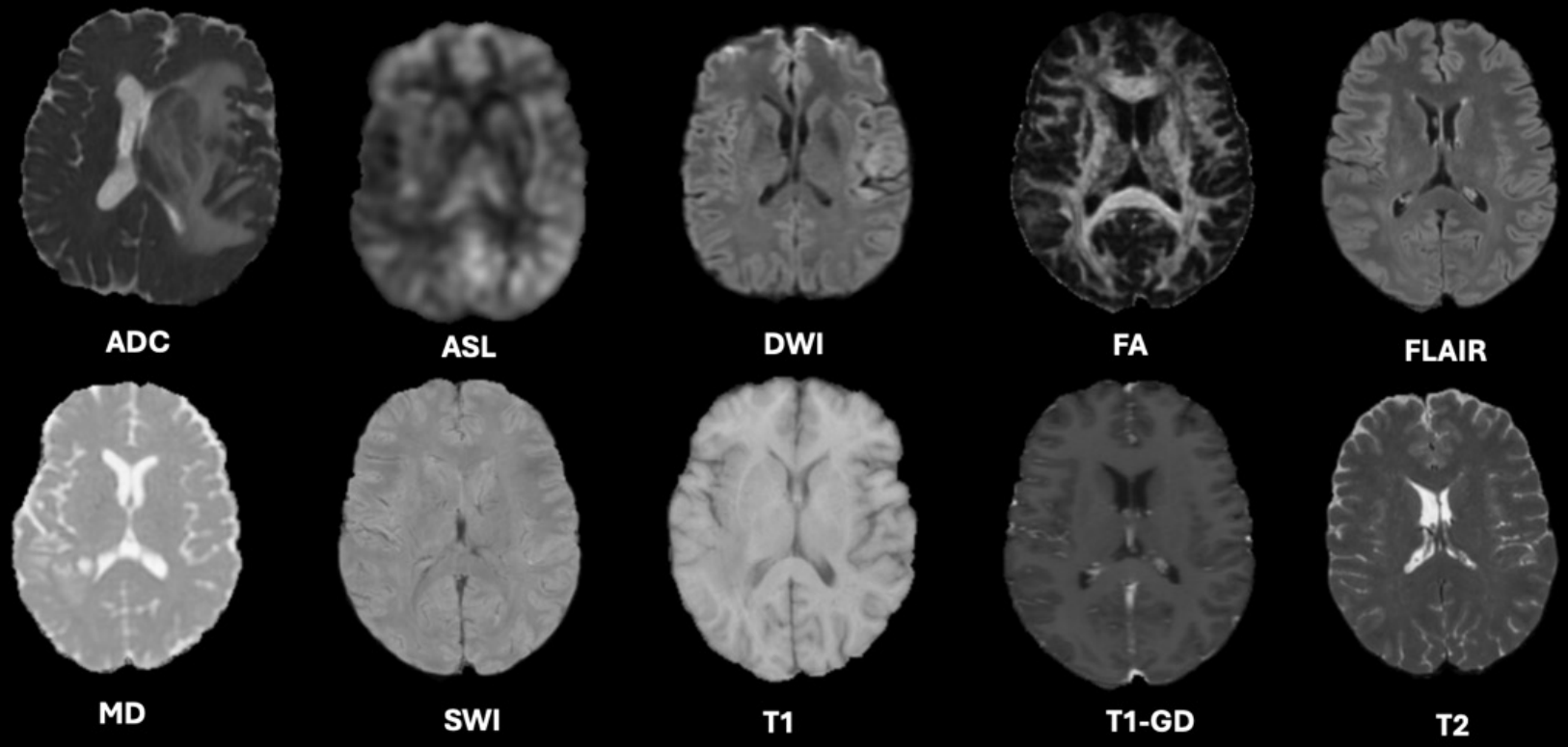
Introduction:





Purpose:

The purpose of this study was to develop a CNN that would automatically classify and label **10** common brain MRI sequences using single slice 2D images.



Methods - Dataset:

The data used was was comprised of **5,010** multiparametric images collected from **501** patients with glioblastoma. The data comes from The Cancer Imaging Archive (TCIA) University of California San Francisco Preoperative Diffuse Glioma MRI (UCSF-PDGM) dataset. Link provided below.

https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=119705830

Methods:

```
# Make all directories and subdirectories

import os
import numpy as np
import nibabel as nib
import pandas as pd
from sklearn.model_selection import StratifiedShuffleSplit
basedir = '/project/radiology/ANSIR_lab/shared/s175064_workspace/UCSF_IDH_Trial/image_picker'

os.mkdir(str(basedir) + str('/train'))
os.mkdir(str(basedir) + str('/val'))
os.mkdir(str(basedir) + str('/test'))

image_list = ['t1', 't1gd', 't2', 'flair', 'swi', 'adc', 'fa', 'md', 'asl', 'dwi']

for i in range(len(image_list)):
    os.mkdir(str(basedir) + str('/train/') + str(image_list[i]))
    os.mkdir(str(basedir) + str('/val/') + str(image_list[i]))
    os.mkdir(str(basedir) + str('/test/') + str(image_list[i]))
```

Methods:

```
# read in all image paths from csv
base = '/project/radiology/ANSIR_lab/shared/s175064_workspace/UCSF_IDH_Trial/Results'
ADC = pd.read_csv(str(base) + str('/ADC.csv'))
ASL = pd.read_csv(str(base) + str('/ASL.csv'))
FA = pd.read_csv(str(base) + str('/DTI_eddy_FA.csv'))
MD = pd.read_csv(str(base) + str('/DTI_eddy_MD.csv'))
DWI = pd.read_csv(str(base) + str('/DWI_bias.csv'))
FLAIR = pd.read_csv(str(base) + str('/FLAIR_bias.csv'))
SWI = pd.read_csv(str(base) + str('/SWI_bias.csv'))
T1 = pd.read_csv(str(base) + str('/T1_bias.csv'))
T1GD = pd.read_csv(str(base) + str('/T1gad_bias.csv'))
T2 = pd.read_csv(str(base) + str('/T2_bias.csv'))
# get brain mask paths so each image will be completely dark in non-brain areas
mask = ADC['Brain_Mask'].to_list()
ADC = ADC['Image'].to_list()
ASL = ASL['Image'].to_list()
FA = FA['Image'].to_list()
MD = MD['Image'].to_list()
DWI = DWI['Image'].to_list()
FLAIR = FLAIR['Image'].to_list()
SWI = SWI['Image'].to_list()
T1 = T1['Image'].to_list()
T1GD = T1GD['Image'].to_list()
T2 = T2['Image'].to_list()
```

Methods:

```
# manually define cutoffs for training, validation, and testing
num_subs = len(ADC)
# where validation indices start - use first 80% of data for training
val_cutoff = round(num_subs*0.8)
# where test indices start
test_cutoff = val_cutoff + round((num_subs-val_cutoff)/2)
print(val_cutoff)
print(test_cutoff)
```

```
401
451
```

Methods:

```
# process all images - read in a nifti file and convert to an anxial slice png
def preprocess_img(img, i, val_cutoff, test_cutoff, subfolder, mask):
    from PIL import Image as im
    outdir = '/project/radiology/ANSIR_lab/shared/s175064_workspace/UCSF_IDH_Trial/image_picker'
    sub_id = img.split('/')[-7]
    mask = nib.load(mask[:])
    mask = np.array(mask.dataobj)
    mask = np.rot90(mask[:, :, 95], 3)
    img = nib.load(img[:])
    img = np.array(img.dataobj)
    # normalize image intensities
    _min = np.amin(img[:, :, :])
    _max = np.amax(img[:, :, :])
    img = ((img[:, :, :] - _min) * (1/(_max - _min) * 255.0)).astype('uint8')
    img = np.rot90(img[:, :, 95], 3)
    img = im.fromarray(np.multiply(mask, img))
    img = img.convert("L")
    if i <= val_cutoff -1:
        end_path = str('/train/') + str(subfolder) + str('/') + str(sub_id) + str('_') + str(subfolder) + str('.png')
    elif i > test_cutoff-1:
        end_path = str('/test/') + str(subfolder) + str('/') + str(sub_id) + str('_') + str(subfolder) + str('.png')
    else:
        end_path = str('/val/') + str(subfolder) + str('/') + str(sub_id) + str('_') + str(subfolder) + str('.png')
    outpath = str(outdir) + str(end_path)
    img.save(outpath)

import nibabel as nib
from PIL import Image as im
for i in range(num_subs):
    preprocess_img(ADC[i], i, val_cutoff, test_cutoff, 'adc', mask[i])
    preprocess_img(ASL[i], i, val_cutoff, test_cutoff, 'asl', mask[i])
    preprocess_img(FA[i], i, val_cutoff, test_cutoff, 'fa', mask[i])
    preprocess_img(MD[i], i, val_cutoff, test_cutoff, 'md', mask[i])
    preprocess_img(DWI[i], i, val_cutoff, test_cutoff, 'dwi', mask[i])
    preprocess_img(FLAIR[i], i, val_cutoff, test_cutoff, 'flair', mask[i])
    preprocess_img(SWI[i], i, val_cutoff, test_cutoff, 'swi', mask[i])
    preprocess_img(T1[i], i, val_cutoff, test_cutoff, 't1', mask[i])
    preprocess_img(T1GD[i], i, val_cutoff, test_cutoff, 't1gd', mask[i])
    preprocess_img(T2[i], i, val_cutoff, test_cutoff, 't2', mask[i])
```

Methods:

```
# Build model and create training/validation batches
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping
outdir = '/project/radiology/ANSIR_lab/shared/s175064_workspace/UCSF_IDH_Trial/image_picker'
train_path = str(outdir) + str("/train")
val_path = str(outdir) + str("/val")

img_datagen = ImageDataGenerator(rescale=1./255,
                                 fill_mode="nearest",
                                 rotation_range=180,
                                 width_shift_range=0.1,
                                 height_shift_range=0.1,
                                 shear_range=0.0,
                                 zoom_range=0.8,
                                 vertical_flip=1,
                                 horizontal_flip=1,
                                 validation_split=0.2
                                 )
train_batches = img_datagen.flow_from_directory(directory=train_path, target_size=(224,224),
                    classes=['adc', 'asl', 'dwi', 'fa', 'flair', 'md', 'swi', 't1', 't1gd', 't2'], batch_size=128)
img_datagen2 = ImageDataGenerator(rescale=1./255, fill_mode="nearest")
val_batches = img_datagen2.flow_from_directory(directory=val_path, target_size=(224, 224),
                    classes=['adc', 'asl', 'dwi', 'fa', 'flair', 'md', 'swi', 't1', 't1gd', 't2'], batch_size=128)

vgg16_model = tf.keras.applications.vgg16.VGG16()
model = Sequential()

for layer in vgg16_model.layers[:-1]:
    model.add(layer)

for layer in model.layers:
    layer.trainable = True
model.add(Dense(10, activation='softmax'))
# View final model architecture
print(model.summary())
```

Methods:

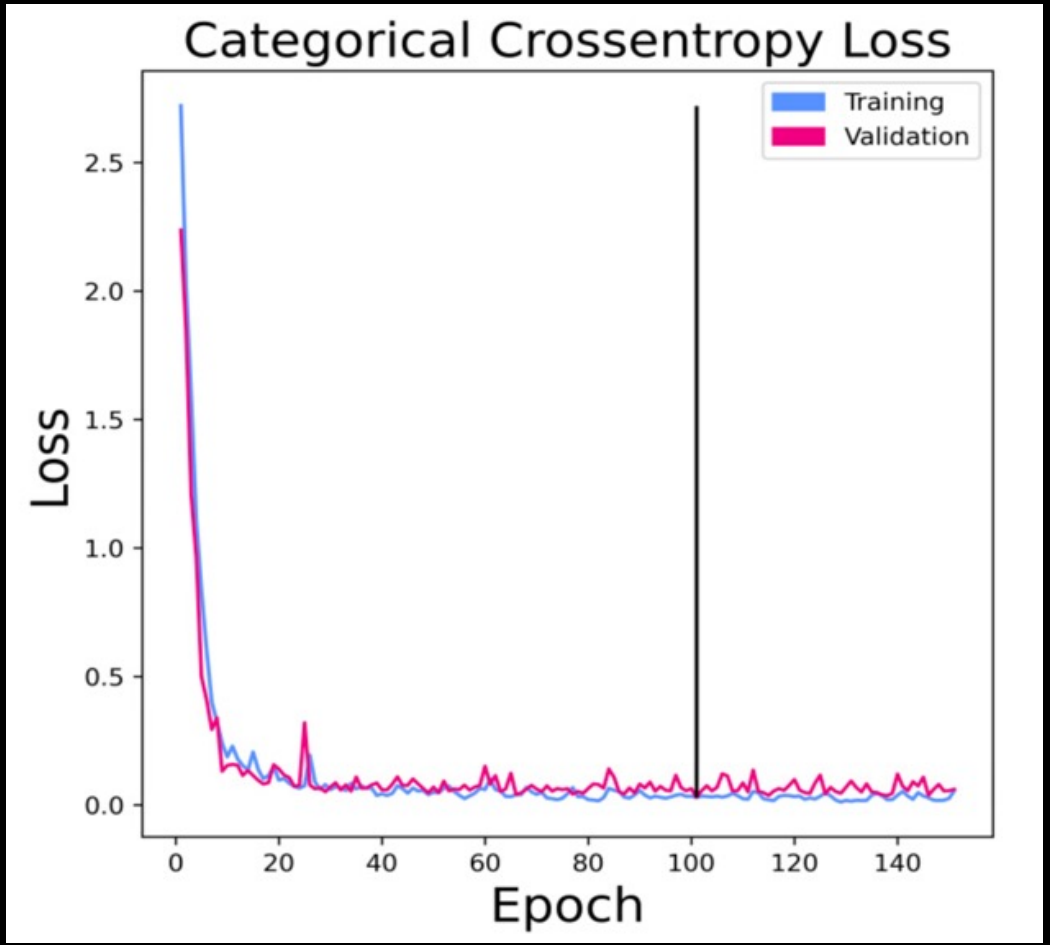
```
# set optimization and how to monitor training
# fit model, save losses, and save model
opt = Adam(learning_rate=1e-4, decay=1e-4 / 500)
model.compile(optimizer=opt, loss='categorical_crossentropy',  metrics = [tf.keras.metrics.CategoricalCrossentropy()]
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-4, patience=50, verbose=1, mode='auto',
                        restore_best_weights=True)
history = model.fit(x=train_batches,
            steps_per_epoch=len(train_batches),
            validation_data=val_batches,
            validation_steps=len(val_batches),
            callbacks = monitor,
            epochs=500,
            verbose=2)
model_path = str(outdir) + str("/image_picker") + str(".h5")
model.save(model_path)
hist_df = pd.DataFrame(history.history)
csv_path = str(outdir) + str("/image_picker") + str(".csv")
hist_df.to_csv(csv_path)
```

Results:

```
# view the losses to make sure the model is learning
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.patches as mpatches
f = plt.figure(figsize=(6, 6))
losses = pd.read_csv('/project/radiology/ANSIR_lab/shared/s175064_workspace/UCSF_IDH_Trial/image_picker/image_picker.
color_palette = ['#648fff', '#dc267f']
sns.set_palette(color_palette)

a = sns.lineplot(y=losses['categorical_crossentropy'], x=range(1, len(losses)+1))
a.axes.set_title("Categorical Crossentropy Loss", fontsize=20)
a = sns.lineplot(y=losses['val_categorical_crossentropy'], x=range(1, len(losses)+1))
a.set_xlabel('Epoch', fontsize=20)
a.set_ylabel('Loss', fontsize=20)
plt.vlines(len(losses)-50, min(losses['val_categorical_crossentropy']),
           max(losses['categorical_crossentropy']), color='k')
train = mpatches.Patch(color='#648fff', label='Training')
val = mpatches.Patch(color='#dc267f', label='Validation')
plt.legend(handles=[train, val])
plt.savefig('losses.tiff', dpi=600, format="tiff")
```

Results:


Categorical Crossentropy Loss
Training
Validation
Loss
2.5
2.0
1.5
1.0
0.5
0.0
0
20
40
60
80
100
120
140
Epoch

Results:

```
# reload the model if needed
import keras
model = keras.models.load_model("/project/radiology/ANSIR_lab/shared/s175064_workspace/UCSF_IDH_Trial/image_picker/im
```

```
# Get all test image paths and store in test_imagePaths list
import os
test_imagePaths = []
# traverse whole directory
for root, dirs, files in os.walk(r'/project/radiology/ANSIR_lab/shared/s175064_workspace/UCSF_IDH_Trial/image_picker/
    # select file name
    for file in files:
        # check the extension of files
        if file.endswith('.png'):
            test_imagePaths.append((os.path.join(root, file)))
```

```
# sort the list for easy interpretation
test_imagePaths.sort()
```

```
# Check image paths to make sure they make sense
print(test_imagePaths)
```

Results:

```
import tensorflow as tf

test_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255, fill_mode = "nearest")

import keras
test_images = test_generator.flow_from_dataframe(
    dataframe=test_df,
    x_col='id',
    y_col='score',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='raw',
    batch_size=1,
    shuffle=False)
```

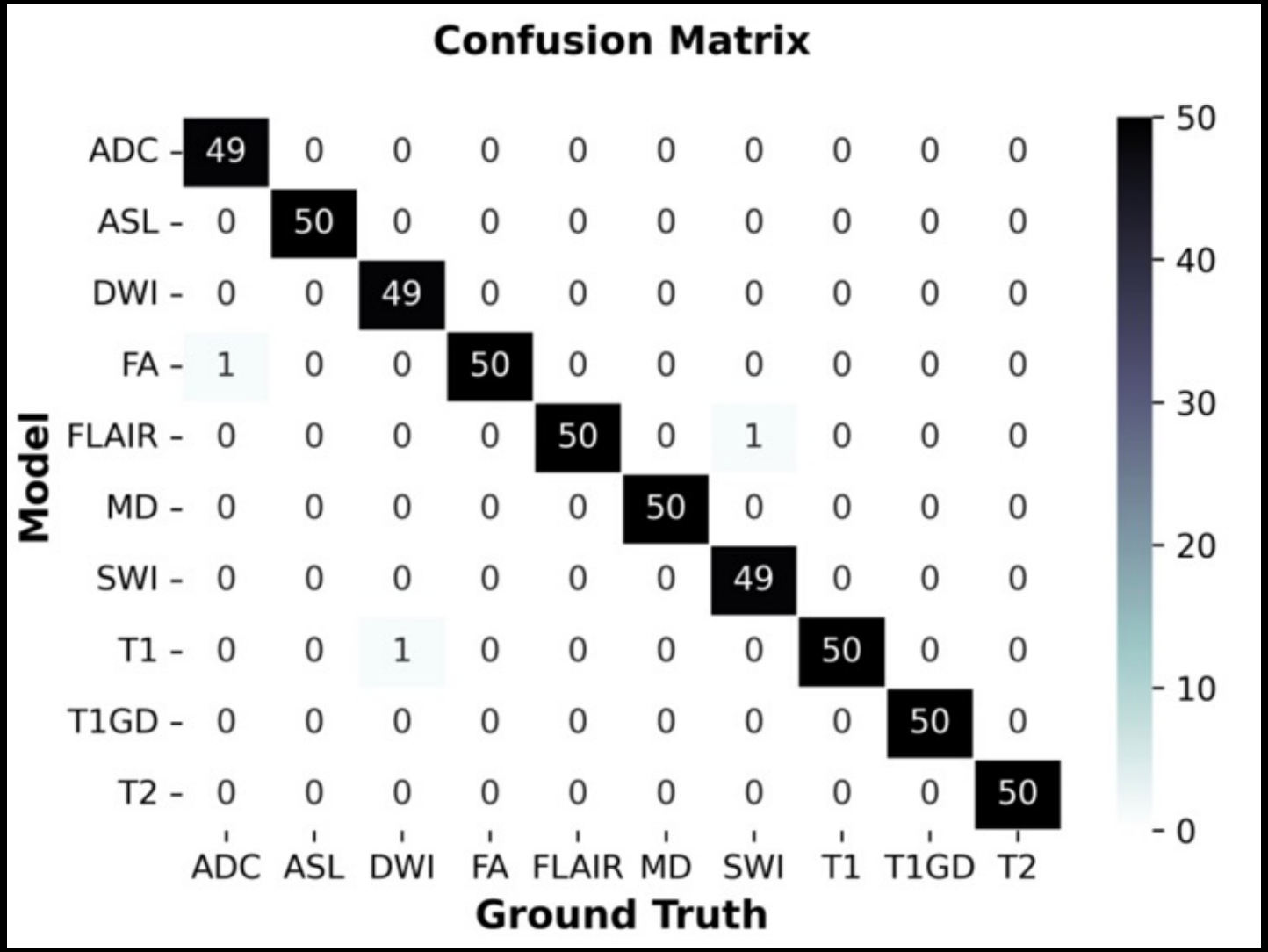
```
Found 500 validated image filenames.
```

```
# Get probabilities for each class and predictions
probs = model.predict(test_images).tolist()
preds = np.argmax(probs, axis=1)
print(preds)
```

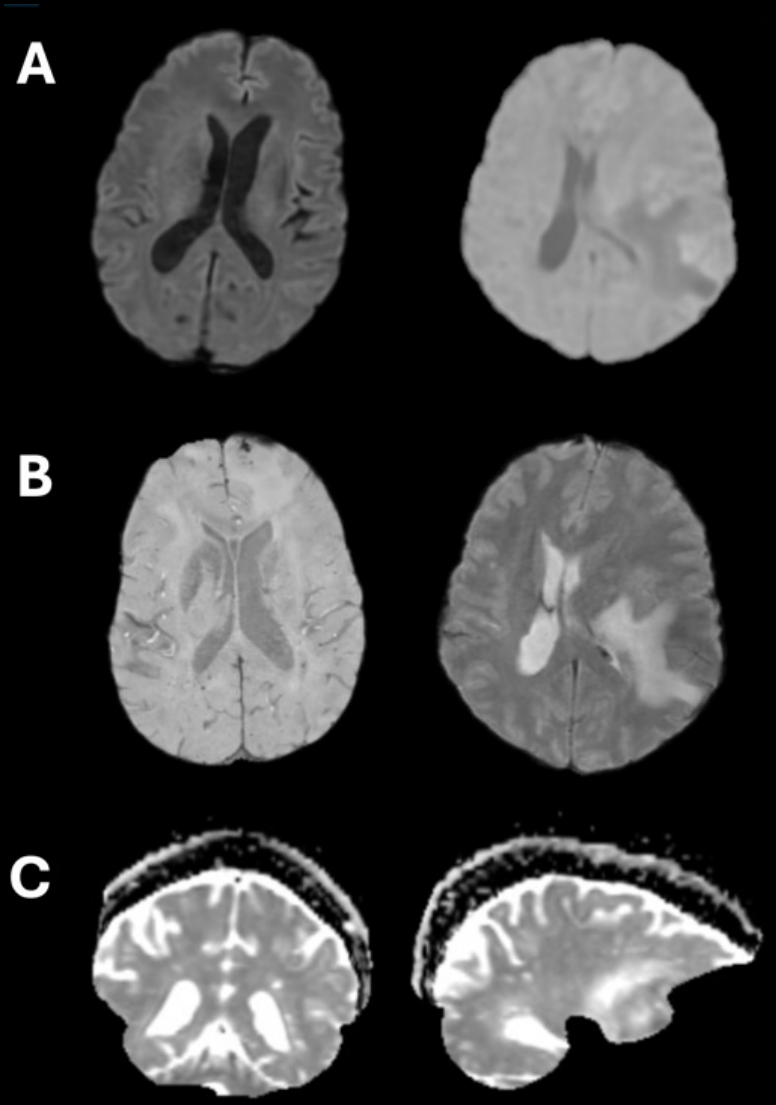
```
# Calculate Accuracy
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(np.array(preds), np.array(test_y))
print("Overall Accuracy: ", accuracy)
```

```
Overall Accuracy:  0.994
```

Results:


Confusion Matrix
Model
ADC
ASL
DWI
FA
FLAIR
MD
SWI
T1
T1GD
T2
Ground Truth
ADC 49 0 0 0 0 0 0 0 0 0
ASL 0 50 0 0 0 0 0 0 0 0
DWI 0 0 49 0 0 0 0 0 0 0
FA 1 0 0 50 0 0 0 0 0 0
FLAIR 0 0 0 0 50 0 1 0 0 0
MD 0 0 0 0 0 50 0 0 0 0
SWI 0 0 0 0 0 0 49 0 0 0
T1 0 0 1 0 0 0 0 50 0 0
T1GD 0 0 0 0 0 0 0 0 50 0
T2 0 0 0 0 0 0 0 0 0 50
50
40
30
20
10
0

Results:



Incorrect DWI Image

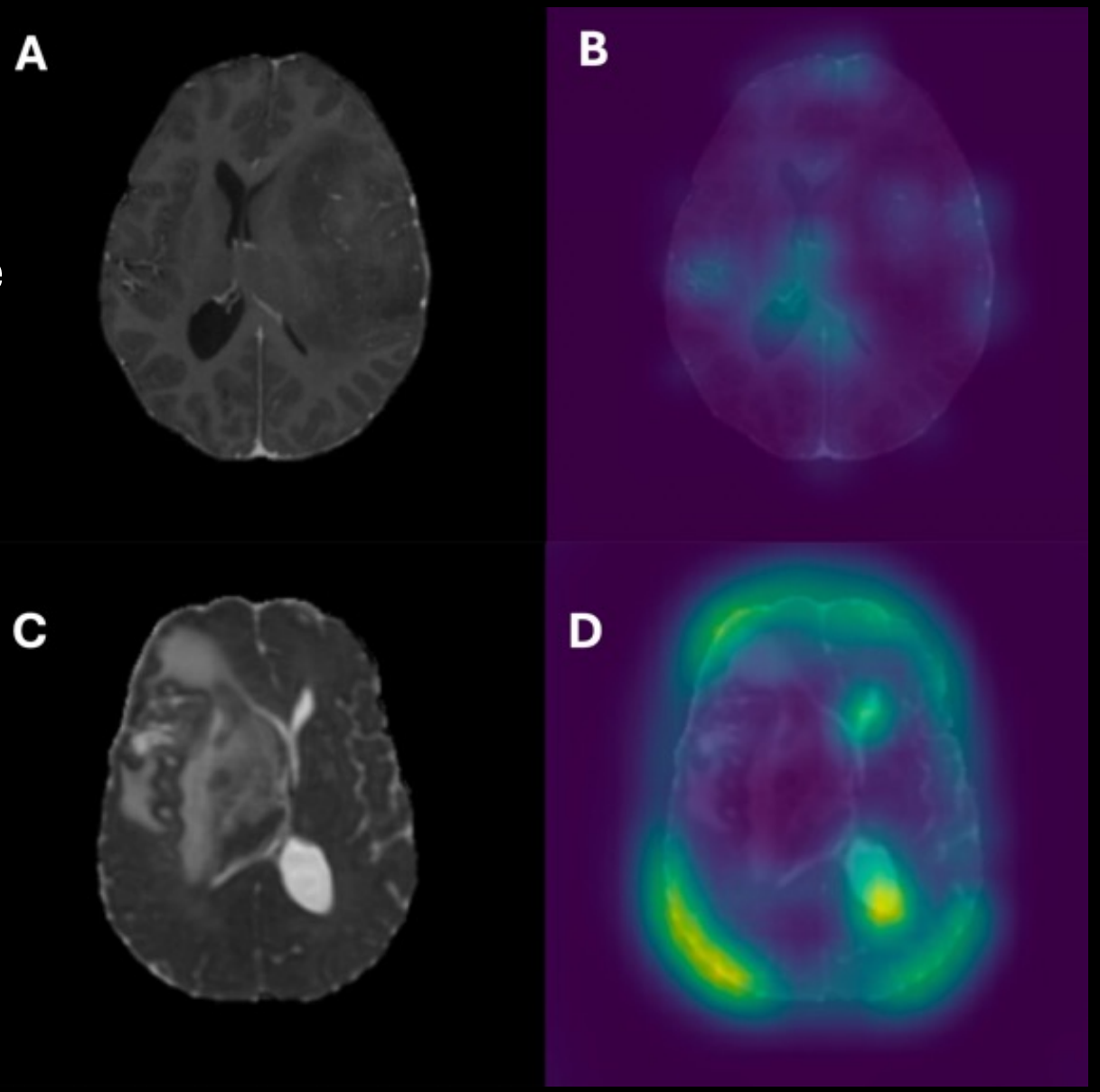
Incorrect SWI Image

Poor Preprocessing

Results GradCAM:

Vessels Used to Classify T1GD Image

Tumor ignored on ADC image



Conclusion:

- When including poorly curated data, my model is **99.4%** accurate.
- When excluding poorly curated data, this model is **100%** accurate.
- Both accuracies > all CNN brain MRI sequence classification results currently reported in the literature.
- Image categories are also **greater** than anything currently published in the literature.

Limitations and Future Work:

- My model was trained on imaging from one scanner at one hospital. Future work should compile images from multiple scanners and institutions for training.
- My model is dependent on high quality data free of artifact. A motion correction algorithm or a quality filter may improve model performance in practice.
- My model was only trained for ten categories of structural images. An eleventh “Other” category would need to be added and appropriate examples would need to be included in the training.
- Lastly, some image types cannot be identified on appearance alone. My model would thus benefit by incorporating large language models to read DICOM information.

Thank you!

James.Holcomb@UTSouthwestern.edu