Detecting and Categorizing Brain Tumors With Python Using a CNN

CS 122

Dr. Wendy Lee

Team Eagle - Friend Dickinson, Matthew Fu, Bella Wei

Introduction/Overview

Purpose

- Many have lost loved ones to brain cancer.
- Use the Brain Tumor **MRI**(Magnetic resonance imaging) dataset to categorize tumors.

Goal

- Develop a Convolutional Neural Network (CNN).
- Accurately detect the type of tumor.

Medical Context

What are tumors?

- Abnormal masses of tissue that form when cells divide more times than they should.
- Not all cancerous!
- Needs to be detected and classified ASAP

- The methods that doctors use
  - Magnetic Resonance Imaging **(MRI)**
  - Computed Tomography **(CT)**
- Problems:
  - Can make mistakes
  - Need years and years of training

Data Selection

Source: Kaggle

Contains: 7022 Images

Training: 78% of Data

Testing: 22% of Data

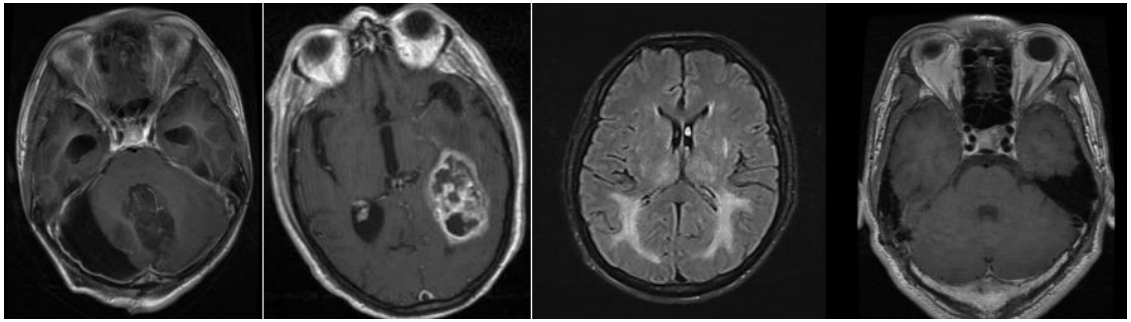


Fig 1: Sample Data (left to right) Glioma, Meningioma, No tumor, Pituitary

Methods(Design Neural Network)

- Two layers of Conv2D for entry block with ReLU activation function
- Two layers of SeparableConv2D, followed by MaxPooling2D layer and residual layer for four sizes
- One last layer of SeparableConv2D, followed by one layer of GlobalAveragePooling2D
- Set model activation to softmax
- Add a Dropout layer and condense all layers with Dense layer
- Train model with training images.

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 64, 64, 1)]	0	
sequential (Sequential)	(None, 64, 64, 1)	0	input_1[0][0]
rescaling (Rescaling)	(None, 64, 64, 1)	0	sequential[0][0]
conv2d (Conv2D)	(None, 32, 32, 32)	320	rescaling[0][0]
batch_normalization (BatchNorma	(None, 32, 32, 32)	128	conv2d[0][0]
activation (Activation)	(None, 32, 32, 32)	0	batch_normalization[0][0]
conv2d_1 (Conv2D)	(None, 32, 32, 64)	18496	activation[0][0]
batch_normalization_1 (BatchNor	(None, 32, 32, 64)	256	conv2d_1[0][0]
activation_1 (Activation)	(None, 32, 32, 64)	0	batch_normalization_1[0][0]
activation_2 (Activation)	(None, 32, 32, 64)	0	activation_1[0][0]
separable_conv2d (SeparableConv	(None, 32, 32, 128)	8896	activation_2[0][0]
batch_normalization_2 (BatchNor	(None, 32, 32, 128)	512	separable_conv2d[0][0]
activation_3 (Activation)	(None, 32, 32, 128)	0	batch_normalization_2[0][0]
separable_conv2d_1 (SeparableCo	(None, 32, 32, 128)	17664	activation_3[0][0]
batch_normalization_3 (BatchNor	(None, 32, 32, 128)	512	separable_conv2d_1[0][0]
max_pooling2d (MaxPooling2D)	(None, 16, 16, 128)	0	batch_normalization_3[0][0]

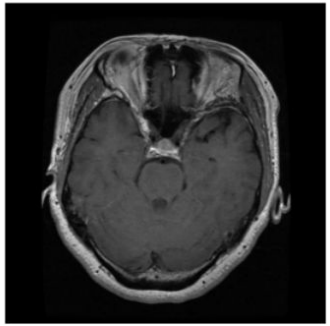
Train Neural network

- Trained over 50 epochs
- Optimizer: Adam Algorithm
- Loss function: Categorical Crossentropy
- Metrics: Accuracy
- Predicted image with a 99.17% accuracy

```
# test_img_filename = './braintumors/Testing/meningioma/Te-me_0068.jpg'
# test_img_filename = './braintumors/Testing/notumor/Te-no_0224.jpg'
test_img_filename = './braintumors/Testing/pituitary/Te-pi_0023.jpg'
# test_img_filename = './braintumors/Testing/glioma/Te-gl_0028.jpg'

img = io.imread(test_img_filename)
plt.figure(figsize=(8, 5))
plt.axis("off")
plt.imshow(img, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x282db227780>
```



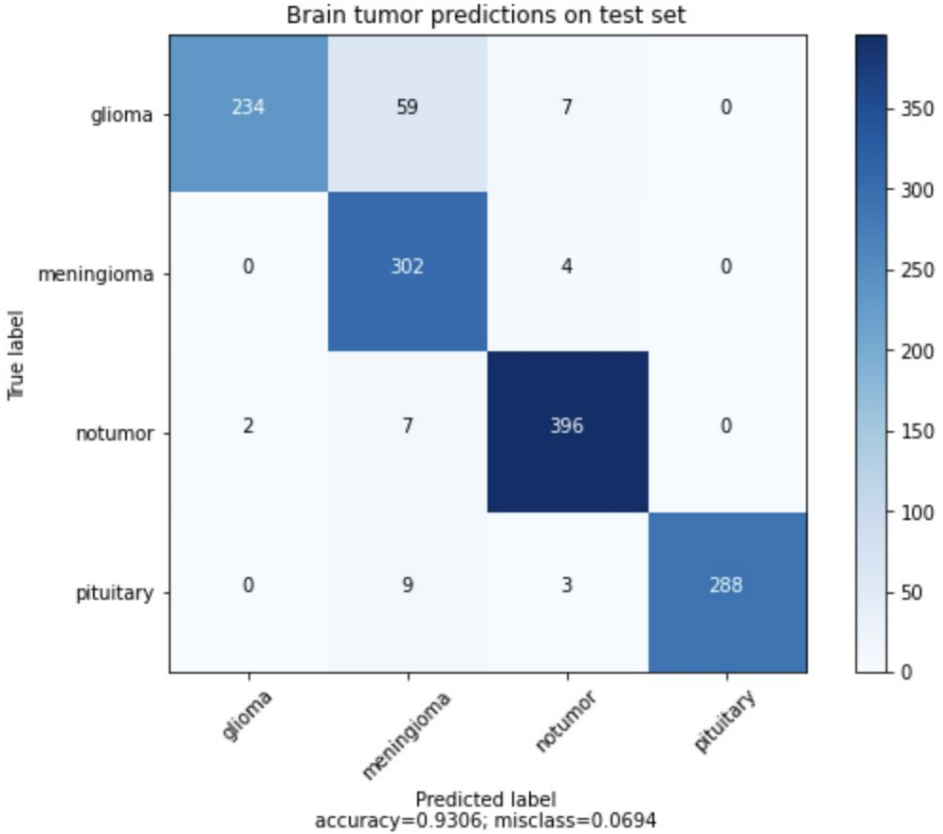
```
[ ] def predict_image(img_array):
        predictions = model.predict(img_array)
        return predictions[0].tolist()

    test_img = keras.preprocessing.image.load_img(test_img_filename, color_mode="grayscale", target_size=(64,64),)
    test_img_array = keras.preprocessing.image.img_to_array(test_img)
    test_img_array = np.array([test_img_array])  # Create batch axis
    predictions = predict_image(test_img_array)
    results_str = ""
    for idx, val in enumerate(class_names):
        score = predictions[idx]
        results_str += f'{(100 * score):.2f}% a {class_names[idx]}'
        if(idx != len(class_names)-1): results_str += " and "
    print(f"This image is {results_str}.")
```

```
This image is 0.00% a glioma and 0.79% a meningioma and 0.04% a notumor and 99.17% a pituitary.
```

Results

- Tested with images from Testing folder
- Unseen by CNN before classifying
- Model predicted with 93% accuracy!
- Generated a confusion matrix w/ TF
  - X-axis: predicted label
  - Y-axis: true label
  - # of images in each cell
  - Hue: darker = more images
- Glioma classified as meningioma was most common misclassification



Discussion

Challenges:

- Neural network design
  - Very abstract
  - Seems like something you need LOTS of experience with to better intuit design principles
- Tensorflow GPU support

Thankfully, neither added much delay!

Next Steps/Improvements:

- Because we ended up with a fast training model, we didn't bother training it for very long
  - Could spend hours/days training
- After finding the limit of our current model, could tweak some layers and try to optimize
- Try on newly generated tumor pictures!

Thanks for listening!

How to run (1/2)

Tested on Windows 10 with an Nvidia GPU:

- Download the term project notebook
- Download the kaggle brain tumor dataset: https://www.kaggle.com/masoudnickparvar/brain-tumor-mri-dataset
    - Put the folder in the same folder as project notebook
- Download python 3.7
- Download pip (python package manager)
    - In terminal, do “pip install [package name]” for every top level package in the imports cell of the project notebook
        - tensorflow, os, matplotlib, etc
    - (there might be some way to make this faster with a file with all the imports needed - but in this case there’s not too many)
    - If you want increased speed by making tensorflow use your GPU, you need to follow these steps: https://www.tensorflow.org/install/gpu

How to run (2/2)

- Download jupyter-lab (or some other way to run python notebooks)
- To start jupyter lab, go in the project folder in a terminal and run “jupyter-lab”
    - To run by training the CNN, leave the training cell (look for the cell with “model.fit“ in it) uncommented and the “start from a checkpoint” cell commented and click Run->Run All Cells.
    - To start from a checkpoint, make sure you have the checkpoint file in a model_checkpoints directory and comment out the training cell, uncomment the checkpoint cell. (I set it to look for the checkpoint with the same number as the set number of epochs - by default 50. You can customize the filename if that isn’t wanted)

Google Colab:

- Should be similar to above. Don’t have to download python/pip or install packages, but may be slower. Epochs took about 5 seconds locally with GPU support enabled for comparison - if it turns out it’s not TOO much slower Colab would definitely be the way to go.