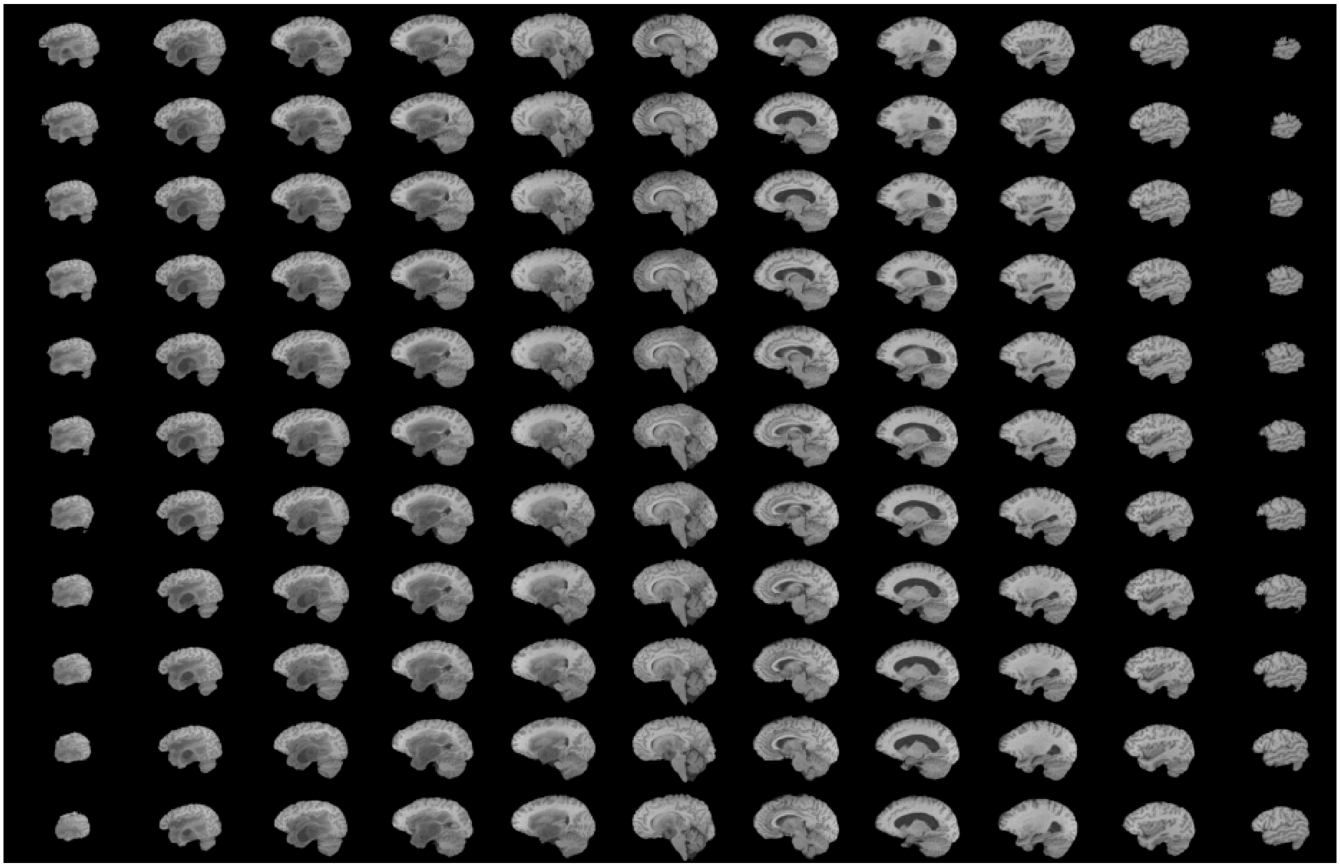
Brain Tumors & Computer Vision

Jonah Maroszek



This image is from my project. Can you see the tumor? Me neither — but my models can.

Introduction

Personal motivation

I will never forget the first time I held a human brain. I was 16, attending a neuroscience camp at UW Madison. On my fourth week, my teacher, a neurosurgeon of 30 years, brought us to a lab. After we took our seats, he walked to a mysterious door in the back of the room covered with ominous labels. As soon as he opened it, the putrid stench of formaldehyde ran into the lab. Not long after, he emerged with a *severed human head*. There was still skin, hair, and everything else you would expect a head to have. It was preserved almost perfectly, as if you could just reattach it to a body, and your only suspicion thereafter would be that the man slept poorly last night. I will spare you the details of what happened next, but soon, we had a human brain before us — the real thing — not a diagram.

When it was my turn to hold the brain, I was astonished at how light it was. I had read the brain is three pounds at least one hundred times, but that fact didn’t sink in until I held one. What I had never read about before was how soft, smooth, and squishy brains are; though I should’ve guessed that too, because they're made of mostly water and fat. As I held this brain, the word that came to mine was *delicate.* I cradled it with careful respect, because in some real sense, I was holding someone’s entire life in my hands. This structure — this remarkably fragile structure — was at the center of every thought, action, and emotion this person experienced. As I set the brain down, I wondered if there was still a person in there, waiting to be rescued.

Since then, I’ve wanted to help humans by helping their brains. I’ve been obsessed with knowing how the brain can function at its best, and just as importantly, how it can malfunction. This matters — perhaps more than anything else — because when something goes wrong, when the parts don’t function together as they should, a person’s fundamental mode of being is altered. A brain malfunction is not simply a physical ailment: it changes a person's contact with reality, and degrades their ability to engage with the good parts of it. As this happens, the person we know and love becomes harder to find. Anyone who’s known someone with Alzheimers is keenly aware of this fact. (Alzheimer’s was my research project at this camp). Recently, I was reminded of the delicacy of the human brain in another form: a family friend of mine was diagnosed with a brain tumor. One month ago, at age six, he died.

This project is dedicated to him.

How computer vision can help

Diagnosing brain tumors fast and accurately is required for effective treatment. It determines what interventions are applied, patience survival rate, and tumor recurrence rate (Gao & Jiang, 2013). Despite radiologists’ best efforts, 3-5% of scans are misdiagnosed, leading to inferior patient outcomes. There are approximately 1 billion radiological scans conducted every year. That equates to 30-50 million annual misdiagnoses (Brady, 2016). These errors can take the form of missing visible anomalies, or failing to find additional anomalies after finding one (termed “search satisfaction”). Errors occur because diagnosis is just as much an art as a science. Much like a machine learning model, a radiologist has to learn what to look for through experience, and each may develop a different unconscious search strategy. As a result, when given the same brain scan, different radiologists may give different diagnoses. That is not surprising. But what is surprising is that the same radiologist may give a different diagnosis when shown the same scan on different days (Al-Khawari et al., 2010). This is likely due to a high workload, long hours, and other contextual variables. We can not blame the radiologists, they are human, and humans make mistakes — and that is why we need computers to help.

Computers do not tire, work as long as they remain plugged in, and give their answers in milliseconds. Importantly, given the same MRI, they will always give the same diagnosis. This standardization is essential for improving treatment over time. Errors due to computer algorithms are scrutinized heavily, and anything less than the best will not be tolerated. I am sure you want the best for you and your loved ones, but here is a noteworthy fact that says you probably aren't getting it: not every radiologist is in the top 5% of radiologists. That is true by definition. Just how big of a difference is there between the best and the rest? It turns out, there is a huge difference. Radiological skill follows the pareto distribution, meaning that a small number have the most skill — by a lot — and therefore provide the best patient outcomes (Brady, 2016). A computer vision model trained to absorb the expertise of the best radiologists can be distributed to every hospital, giving every doctor the resources they need to save human lives. This is especially important for under-served communities.

In the near future, computer vision models will provide a fast, effective, and scalable solution to an impactful problem: patients are not getting the care they deserve. This project shows why that is possible.

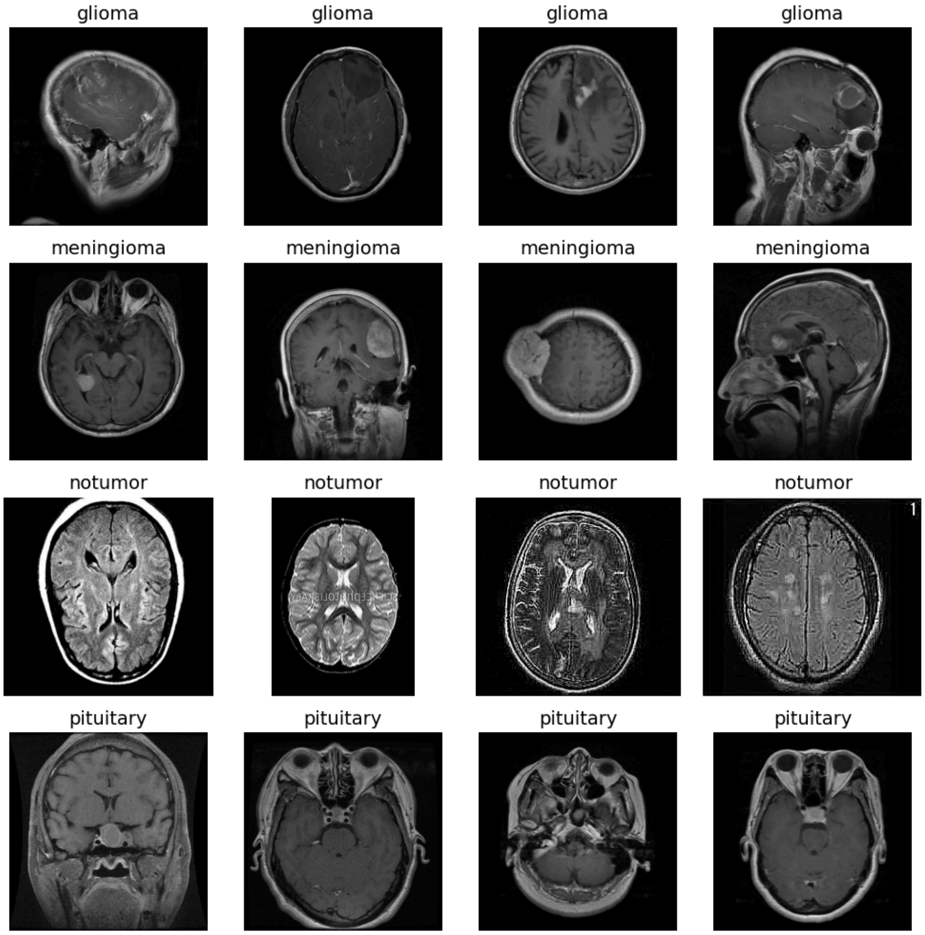
Initial plan and development of this project

At the start of this project, I planned to use transfer learning with three CNNs for brain tumor classification: VGG16, Xception, and EfficientNet, then compare its performance with AnomalyGPT, a system for anomaly detection and image segmentation that is integrated with a large language model. I will tell you more about the first part in the remainder of the report, but for the second part, I had to change strategies; I successfully reproduced the results of AnomalyGPT on the datasets it was trained on (see appendix), however, I had trouble adapting it to my needs. Towards the end of the project, with advisor and TA permission, I decided to focus exclusively on CNN computer vision models. Therefore, I implemented a custom object segmentation algorithm based on the U-net architecture to complement my classification networks.

Tumor classification

The data

The dataset used to train tumor classification is from Kaggle. It is small by big data standards: a mere seven thousand images, totalling 160 MB. It consists of brain MRIs that have four possible classes: three represent a tumor (glioma, meningioma, pituitary), and the fourth is benign, or no tumor. Four examples of each class are shown below.



As you can see, the scans are taken from different angles and deal with different cross sections of brain tissue. The location of the brain tumor affects the presence of surrounding tissue. For example, take the pituitary row. The three rightward images contain two small black dots, which are eyes, not tumors. The model has to learn to distinguish decoys like this from real tumors — and only from 7k images. Luckily, there are three factors that allowed me to achieve strong performance: I artificially increased the training data size with data augmentation, the dataset represents each class well, and I built upon already high-performing neural networks.

Data preprocessing

Preprocessing for this dataset was simple and straightforward. I created synthetic examples by applying the following transformations to the training set: rotation, shear, zoom, inversion, pixel standardization, and horizontal and vertical shifts. These changes resulted in robust training that had strong generalizable performance.

The models

I planned to explore three architectures for tumor classification, but after my second attempt, the performance was strong enough to make that unnecessary. As a result, I have two classification networks: one based on VGG16, and the other based on Xception. Out of the box, neither performed well on my brain scans. For each network, I added about 8 layers to them, then fine-tuned them with my data. My intention was to achieve the best performance I could, not to compare the base architectures, so the layers I added to each were different. Although my version of Xception is the clear winner, it is not necessarily because that is the better model. I did that one second, and learned more about deep learning and computer vision since my first attempt with VGG16.

VGG16

VGG16 was introduced in 2014 by the Visual Geometry Group at the University of Oxford. It is primarily composed of standard convolutional layers. Although it's an older architecture, I thought it would be a great base model given its continued success on many image classification tasks. After freezing the initial network, I experimented with adding other layers. I let each configuration run for a small number of epochs, then let the best one train for 50. The final architecture is shown in the appendix. In the end, the model performs well, but not exceptionally so: the overall accuracy is 89%. More detail about the performance will be given after a brief introduction to Xception and my training strategy with that.

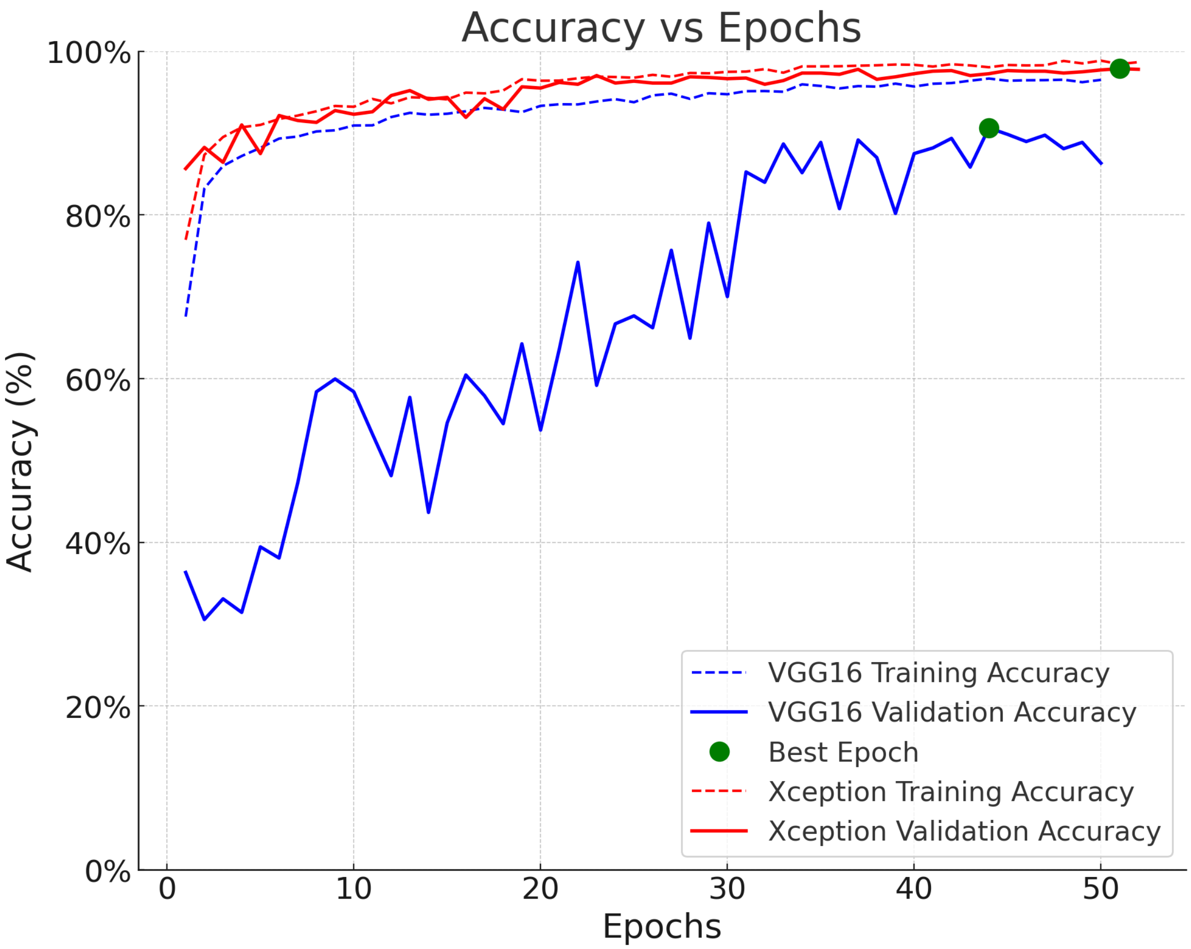
Xception

Xception uses separable convolutional layers that often increases the accuracy and lowers the required number of parameters. I froze the base model, then experimented with adding my own layers. My best performing network had three convolutional layers, followed by a GlobalAveragePooling layer, then two Dense layers interspersed with Dropouts (see the appendix). In my mind, this was a much better architecture design than my first attempt with VGG16. In the end, I achieved 98% accuracy here.

I believe another component of my success with this model was due to the training strategy. I used an adaptive learning rate that became progressively lower if the validation loss stopped improving after 5 epochs. Before this strategy, Xception was oscillating around 91%, which is about the level of my VGG performance. On the final run, I noticed that every time the learning rate decreased, the model's validation loss improved after having been stuck. Since this model is more computational efficient, I was going to let it train for 100 epochs, but it hit my early stopping condition and ended at 52 epochs.

Comparing the tumor classification networks

Performance on the training data was similar for both networks, however, Xception had better generalizable performance. Additionally, it converges faster and has about 5x fewer parameters.



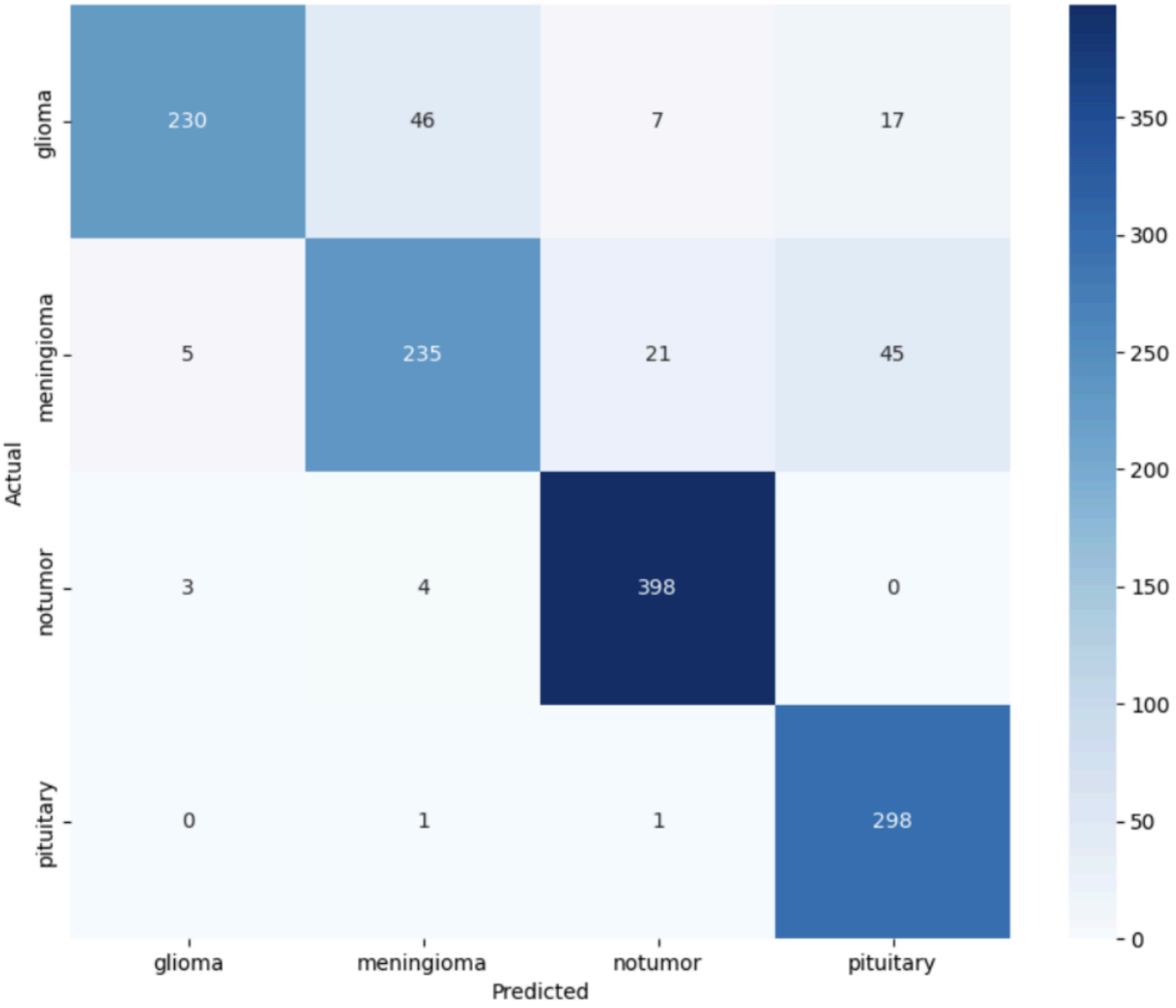
Confusion matrices

VGG16

From the confusion matrix, we can see there are two problematic scenarios with the VGG model:

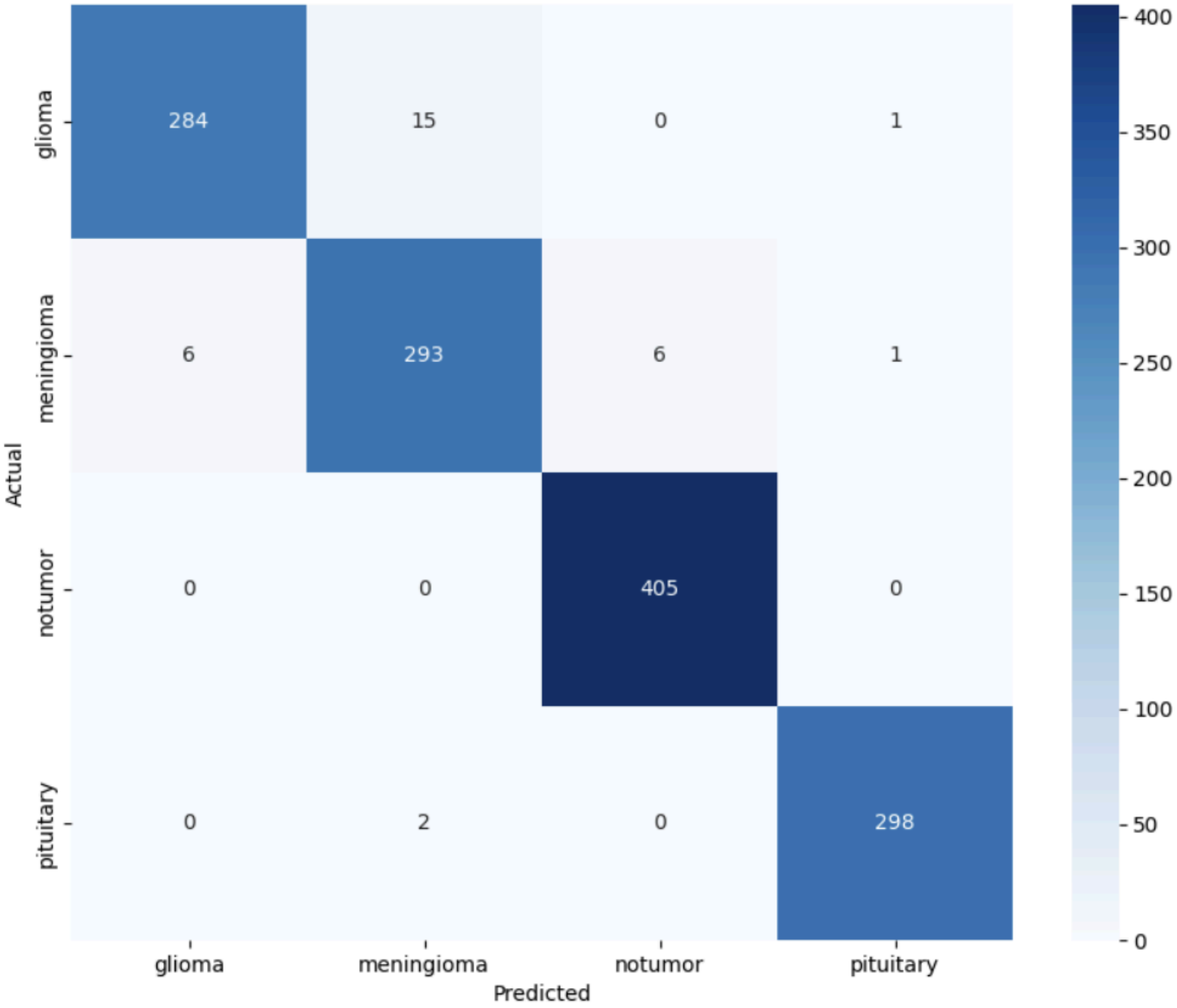
1. The actual diagnosis is meningioma, but the model predicts a pituitary tumor
2. The actual diagnosis is glioma, but the model predicts it is a meningioma

As of right now, I am not sure why that is. If this were the best performing model, I would explore ways to improve prediction in these subcases.



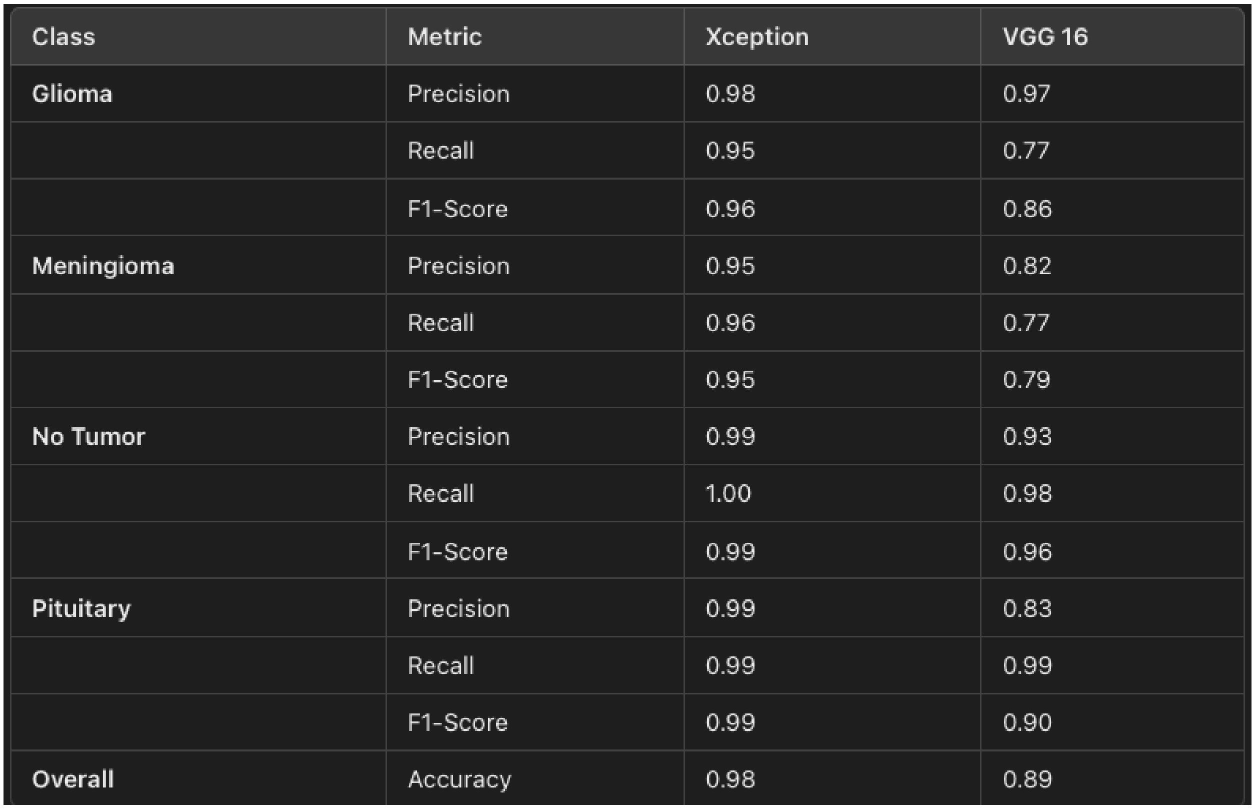
Xception

Overall, the performance is excellent, but the most problematic scenario for Xception is mistaking a glioma for a meningioma. Notice it has perfect recall for the no tumor case. It is also extremely good at identifying pituitary tumors.



Classification report

The Xception model is universally better.



Class	Metric	Xception	VGG 16
Glioma	Precision	0.98	0.97
	Recall	0.95	0.77
	F1-Score	0.96	0.86
Meningioma	Precision	0.95	0.82
	Recall	0.96	0.77
	F1-Score	0.95	0.79
No Tumor	Precision	0.99	0.93
	Recall	1.00	0.98
	F1-Score	0.99	0.96
Pituitary	Precision	0.99	0.83
	Recall	0.99	0.99
	F1-Score	0.99	0.90
Overall	Accuracy	0.98	0.89

Tumor Segmentation

With my Xception network, I can identify tumors and their types with a high degree of confidence. This is surely helpful, however, there are two drastic limitations with the previous approach. First, the model doesn't indicate what features of the image result in a particular diagnosis. Second, the dataset only included 2D images that were already deemed relevant for diagnosis by an experienced radiologist. Ideally, our model would take in all data given by an MRI machine, then highlight potentially problematic regions. I focused on solving these problems with the techniques I'll share next.

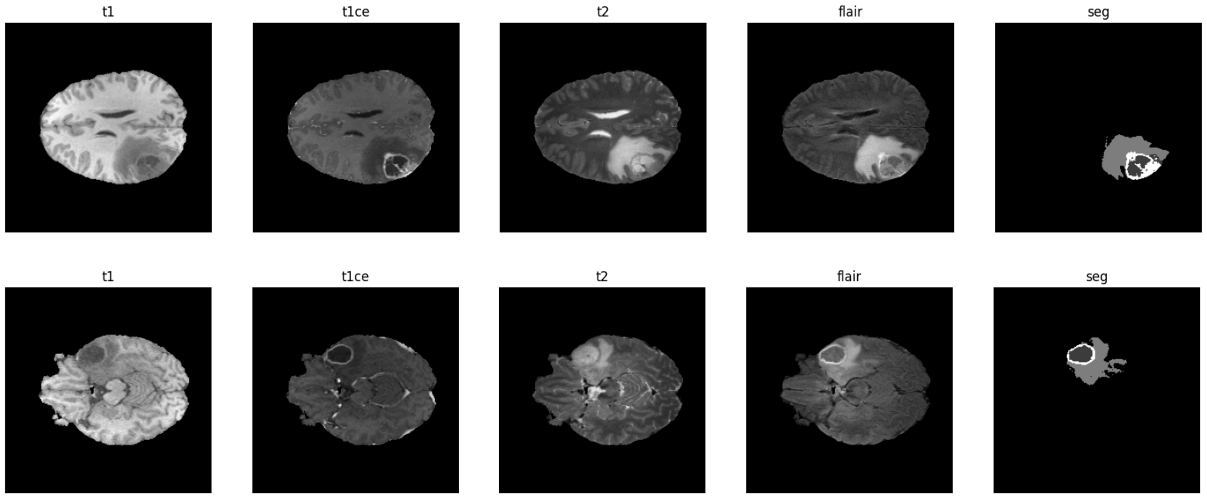
The data

I used the BraTS2020 Dataset for tumor segmentation. Compared to the first dataset, it is more complex. First of all, it's 40 GB, but the enormous size doesn't come from more brains: there are actually only 369 training and 125 validation examples. The size and complexity comes from the high dimensional information given by the MRI machine. The first dataset only included one 2D slice from each brain, but here, we have all the slices — some relevant and some not — and in addition, each slice has more channels to consider. The most important ones are:

1. T1
2. T1 post contrast
3. T2 weighted
4. T2 flair
5. Image segmentation mask

It is not important to understand in depth what these channels mean, but the basic idea is that an MRI technician can change properties of the machine to focus on certain tissues: fatty, water-based, and so on. These correspond to items one through four. The image segmentation mask is not an output of the machine, but a pixel-by-pixel description of brain tissue produced by an experienced radiologist (and vetted by others). The mask is used to distinguish normal from diseased tissue, then further classify diseased tissue. Since all tumors in this dataset are gliomas, classification means identifying subregions of the glioma (swelling, dead tissue, active tissue, etc).

Below are relevant slices from two brains showing different channels. For each brain, these channels are combined into one tensor in the data preprocessing.



Data preprocessing

The dataset is composed of .nii files, which is the standard format for neuroimaging. I defined a data generator to efficiently load and preprocess these files. The generator converts them to numpy arrays, rescales all pixels to be in the range of 0 to 1, and concatenates each channel into a single tensor. For computational reasons, I ignored the T1 slice, as most people say it’s redundant if you have a T1 contrast scan. In addition, the generator crops the spatial dimensions, focusing on the interior of the brain (again, for computational efficiency). Finally, it prepares the tensors in batches for efficient training. The end result is a 5D tensor with shape (batch_size, 128, 128, 128, 4). The “128”s are the spatial dimensions, and the 4 represents the three scans (t1 contrast, t2, and flair) plus the segmentation mask.

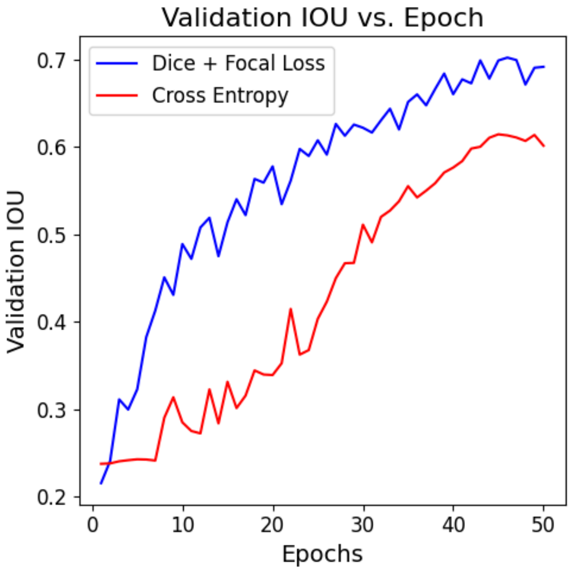
Although the dataset comes with a validation folder, there aren’t ground truth masks in it. I used 20% of the original training folder as a validation set, so I could calculate assessment metrics like the IOU score from the ground truth mask.

The model: 3D Unet

U-net is widely recognized for its utility in biological image segmentation. It is composed of an encoder where the spatial channels get progressively smaller, a decoder where they get upsampled to the original size, and skip connections between each that preserve precise spatial relationships from the original image. I extended the 2D U-net architecture to work with the tensors I prepared in data preprocessing with a custom implementation in Keras.

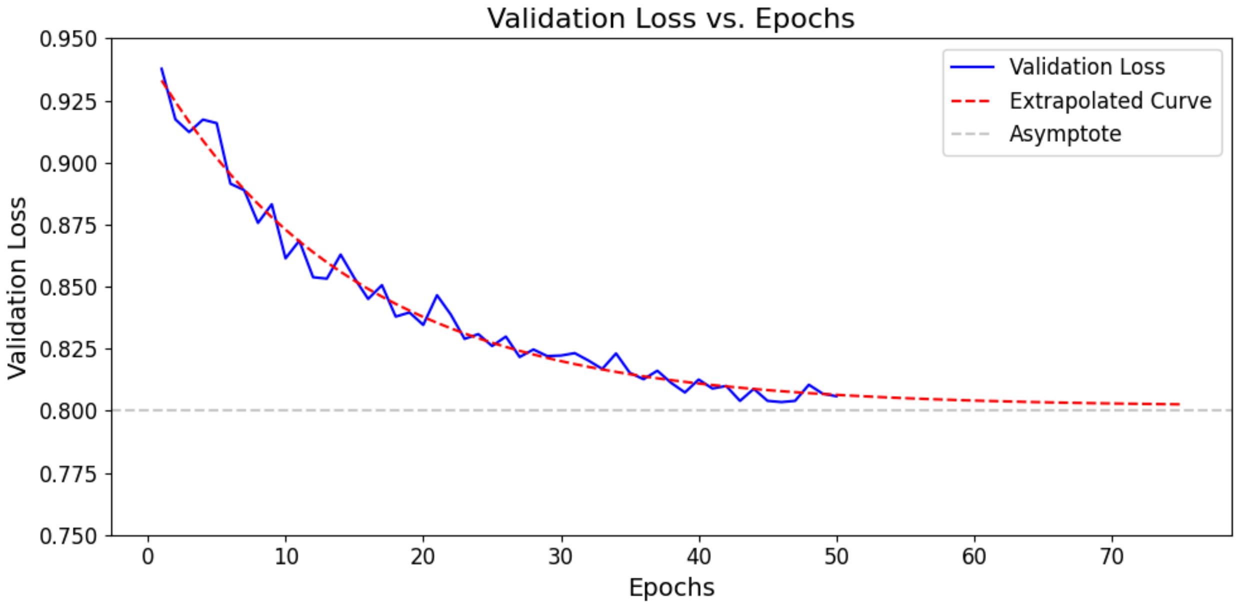
Training

The choice that seemed to have the most impact on training was the loss function. I started with categorical cross entropy, but then discovered others that might perform better in my case: dice loss and focal loss. Focal loss modifies the standard cross entropy loss function by downweighting well-classified examples, focusing more on hard-to-classify examples as the model trains. Dice loss is based on the dice coefficient, and is useful when there is a class imbalance. That is the case here, because most brain tissue is healthy even for people with aggressive tumors. To leverage the strengths of each, I created a custom loss function that is the sum of the dice and focal loss. The following chart compares the performance of cross entropy with this custom loss function.[1]



[1] The dice and focal loss was trained with a batch size of 16, because I had access to an A100 GPU. The cross entropy loss was trained with a batch size of 8, because I could only connect to an L4 GPU. Otherwise, all other hyperparameters were the same.

Based on the chart above, it looks reasonable that the IOU could continue to improve with more epochs for either loss function. I was running out computational credits, so I wondered if I should buy more. To test if it would be worth it, I plotted the validation loss of my best model to predict how performance would scale with epochs.

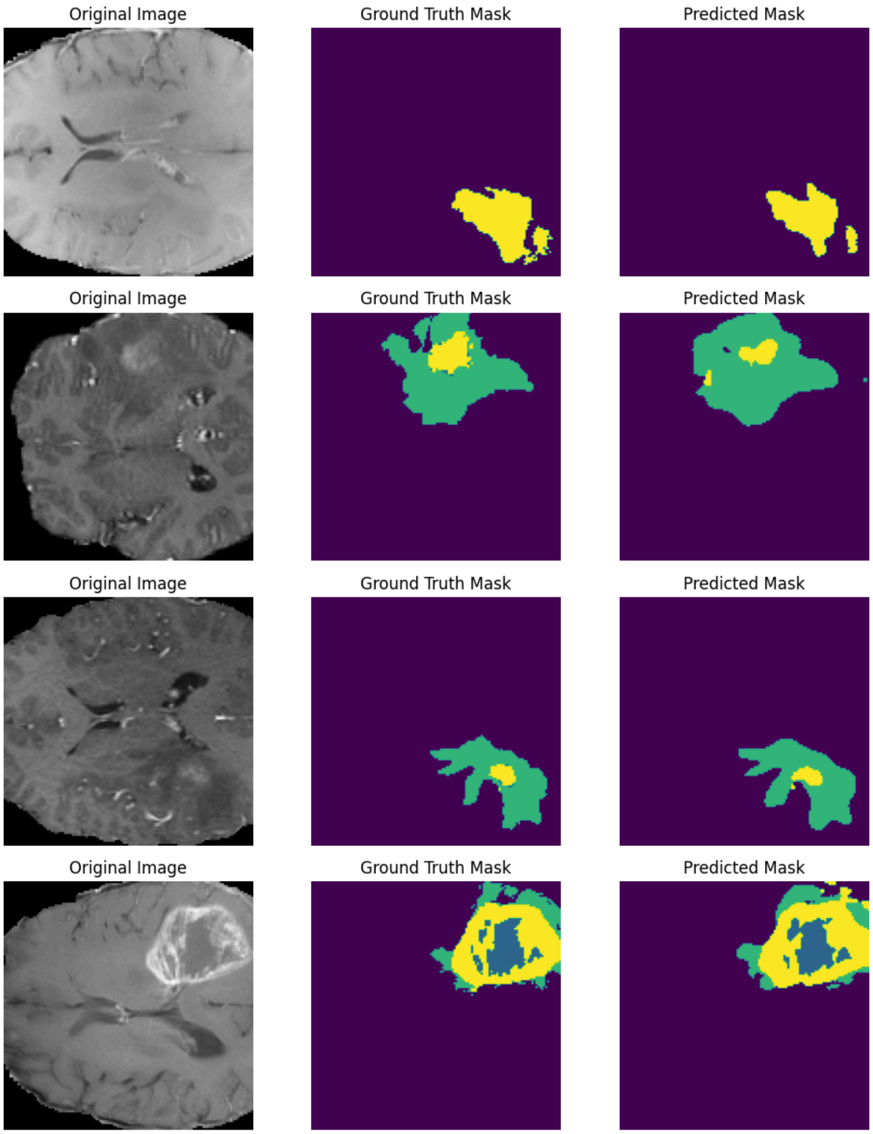


I did not buy more credits.

Results

My best model achieves an accuracy, precision, recall, and f1 score of 0.98 for all metrics just listed. Precision, recall, and the F1 score are still useful in an image segmentation context, however, accuracy is often substituted for the IOU score or the dice coefficient, because imbalanced classes on the segmentation mask make it less meaningful. I used the IOU score, which measures the quality of the overlap between predicted and ground truth masks. My model has an IOU score of 0.7, which is pretty good given the complexity of the task.

Below, you can see the first four samples in a batch from my test generator. In the first row, I can't even tell there is a tumor there after knowing there is one, but apparently the model can. In all cases, the masks line up well with the radiologist's mask.



Although the masks are not perfect, they are surely good enough to counteract the two most common sources of error made by radiologists: failure to find an anomaly when there is one, and failure to identify multiple anomalies when they are present.

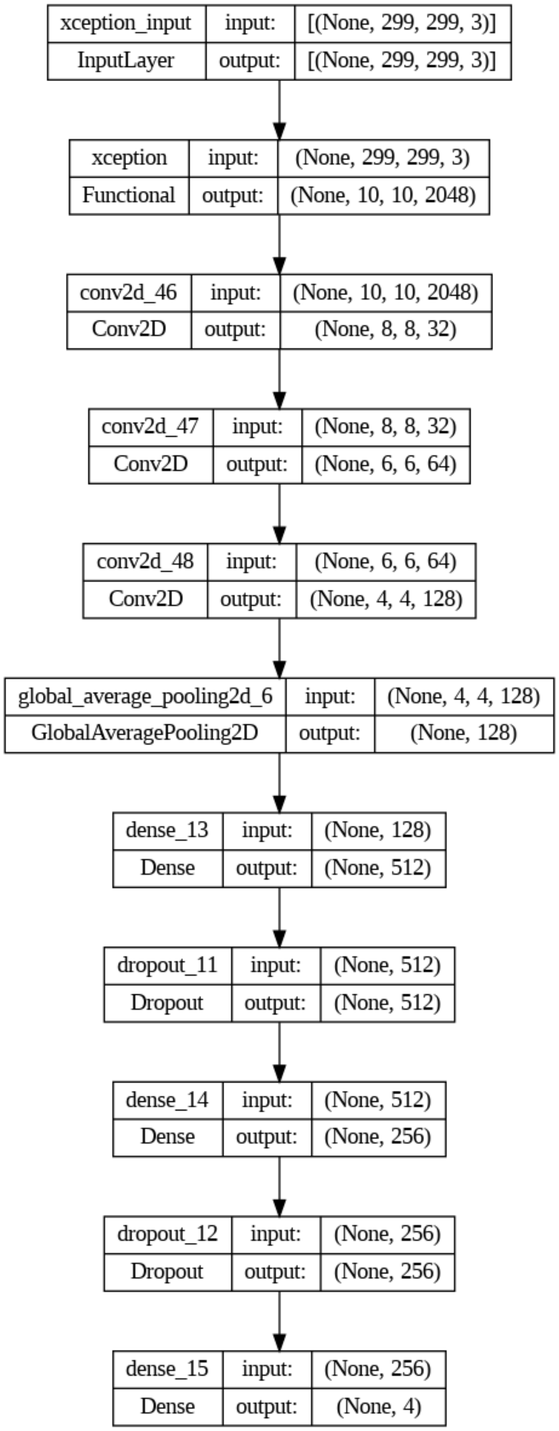
Conclusion

Through this project, I hope I've effectively added to the growing number of voices that say computer vision will revolutionize health care. With Xception, we have seen it's possible to accurately diagnose 98% of brain scans, which is better than even the most optimistic estimate of correctly diagnosed medical images in the US. With 3D U-net, we've seen it's possible to take it a step further by pointing out which parts of the scan are problematic after ingesting all data given by an MRI machine. Integrating existing human expertise with technological innovations that enable rapid, consistent, and robust diagnoses will allow doctors to provide effective care to everyone who needs it.

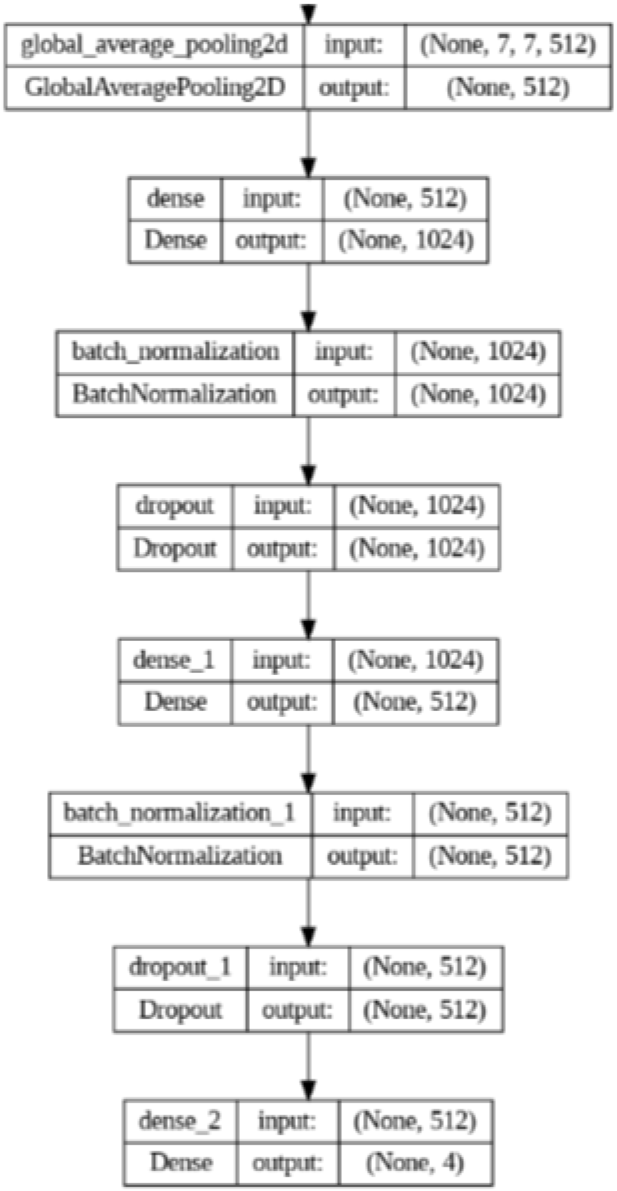
Appendix

CNN architectures

Layers added to Xception


xception_input | input: | [(None, 299, 299, 3)]
InputLayer | output: | [(None, 299, 299, 3)]
xception | input: | (None, 299, 299, 3)
Functional | output: | (None, 10, 10, 2048)
conv2d_46 | input: | (None, 10, 10, 2048)
Conv2D | output: | (None, 8, 8, 32)
conv2d_47 | input: | (None, 8, 8, 32)
Conv2D | output: | (None, 6, 6, 64)
conv2d_48 | input: | (None, 6, 6, 64)
Conv2D | output: | (None, 4, 4, 128)
global_average_pooling2d_6 | input: | (None, 4, 4, 128)
GlobalAveragePooling2D | output: | (None, 128)
dense_13 | input: | (None, 128)
Dense | output: | (None, 512)
dropout_11 | input: | (None, 512)
Dropout | output: | (None, 512)
dense_14 | input: | (None, 512)
Dense | output: | (None, 256)
dropout_12 | input: | (None, 256)
Dropout | output: | (None, 256)
dense_15 | input: | (None, 256)
Dense | output: | (None, 4)

Layers added to VGG16



global_average_pooling2d | input: | (None, 7, 7, 512)
GlobalAveragePooling2D | output: | (None, 512)

dense | input: | (None, 512)
Dense | output: | (None, 1024)

batch_normalization | input: | (None, 1024)
BatchNormalization | output: | (None, 1024)

dropout | input: | (None, 1024)
Dropout | output: | (None, 1024)

dense_1 | input: | (None, 1024)
Dense | output: | (None, 512)

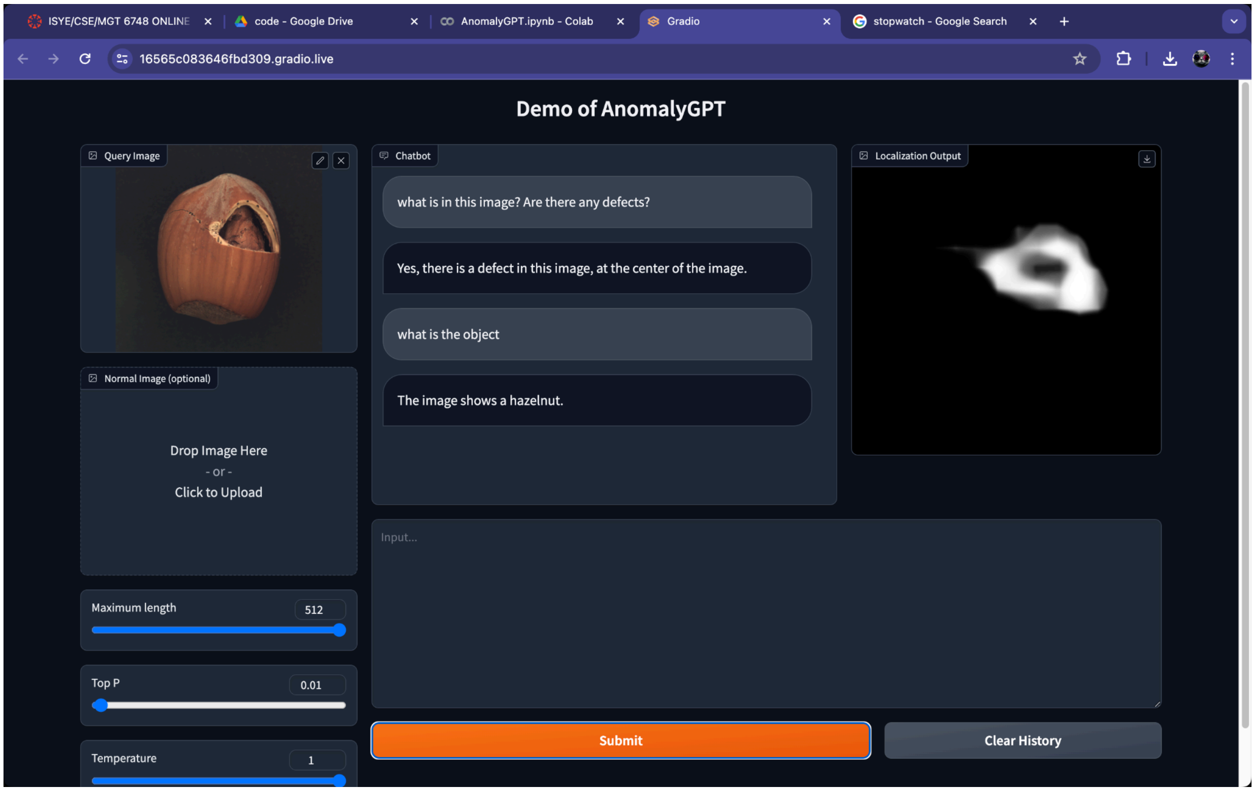
batch_normalization_1 | input: | (None, 512)
BatchNormalization | output: | (None, 512)

dropout_1 | input: | (None, 512)
Dropout | output: | (None, 512)

dense_2 | input: | (None, 512)
Dense | output: | (None, 4)

Reproduced results of AnomalyGPT

Showing that I got AnomalyGPT working on the traditional image dataset



what is in this image? Are there any defects?

Yes, there is a defect in this image, at the center of the image.

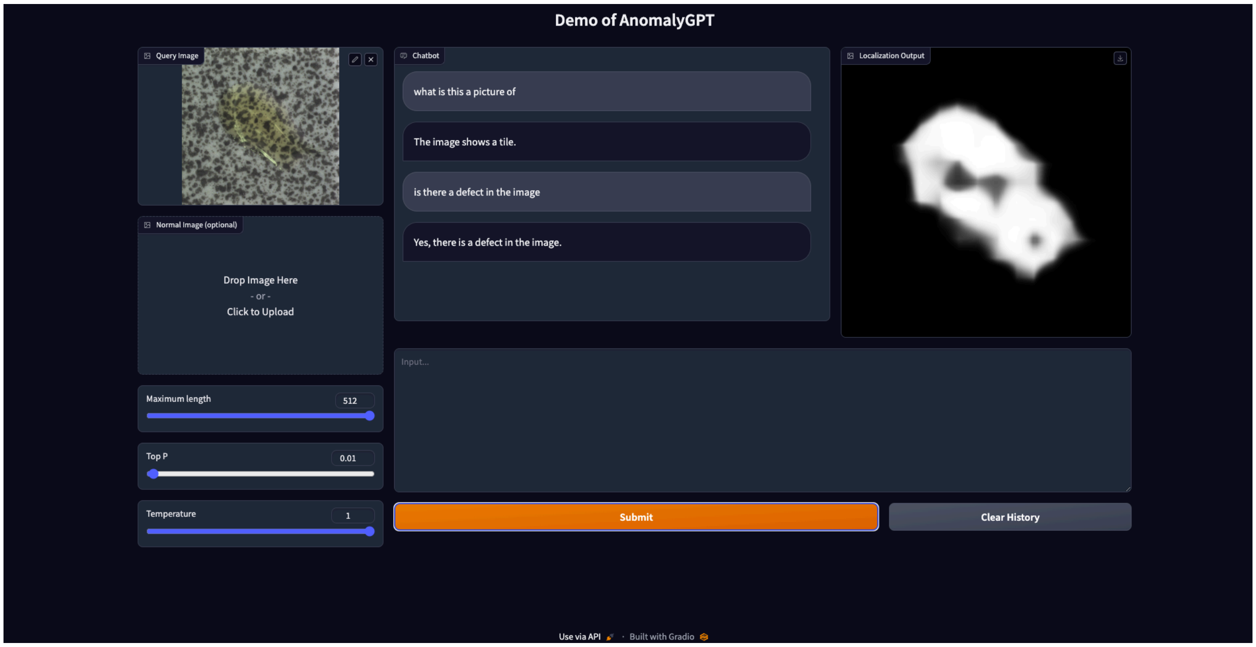
what is the object

The image shows a hazelnut.

Maximum length 512

Top P 0.01

Temperature 1



what is this a picture of

The image shows a tile.

is there a defect in the image

Yes, there is a defect in the image.

Maximum length 512

Top P 0.01

Temperature 1

References

Al-Khawari, H., Athyal, R., & Sada, P. (2010). Inter- and intraobserver variation between radiologists in the detection of abnormal parenchymal lung changes on high-resolution computed tomography. *NCBI*. Retrieved June 25, 2024, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2855063/

Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J. S., et al. (2017). Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. *Nature Scientific Data*, 4, 170117. DOI: 10.1038/sdata.2017.117

Bakas, S., Reyes, M., Jakab, A., Bauer, S., Rempfler, M., Crimi, A., et al. (2018). Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge. *arXiv preprint*, arXiv:1811.02629.

Brady, A. P. (2016, December 7). Error and discrepancy in radiology: Inevitable or avoidable? *NCBI*. Retrieved June 25, 2024, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5265198/

Gao, H., & Jiang, X. (2013). Progress on the diagnosis and evaluation of brain tumors. *NCBI*. Retrieved June 25, 2024, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3864167/

Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., et al. (2015). The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging*, 34(10), 1993-2024. DOI: 10.1109/TMI.2014.2377694