Brain MRI Classification for Tumour Detection

by

Rebecca Holland

24/25 Methods 4: Deep Learning and Big Data Integration: Technologies and Tools

18th June 2025

Abstract

This project explores the use of AI in MRI image-based brain tumour diagnosis. To determine if an MRI image had a tumour or not, a convolutional neural network built on the ResNet18 architecture was trained. The system, which was created in Google Colab and built in PyTorch, was eventually made available via a simple Streamlit interface that allowed anyone to input photographs and get predictions and confidence scores. The motivation behind this project is personal, inspired by the loss of a loved one to a brain tumour that was detected too late. The technical implementation, decision-making procedure, and practical utility of the system are described in this report. As an experienced user of LaTeX and Overleaf, I have formatted this report accordingly to maintain clarity, structure, and academic rigour.

Contents

0.1 Project Idea

The project's goal is to create a binary image classifier that can recognise brain cancer in MRI pictures. It utilises transfer learning with the ResNet18 architecture (Contributors 2025) to classify images into two categories: "Tumour" or "No Tumour". The technology supports early diagnosis choices, especially for non-specialist evaluation or in low-resource situations, offering a useful application of AI in healthcare (Esteva et al. 2019).

0.2 What I Made

The system developed includes the following components:

- A fully commented and functional Python codebase written in Google Colab
- A custom PyTorch Dataset class for loading and transforming images
- A transfer learning setup using pretrained ResNet18
- Data augmentation strategies (rotation, flipping) to enhance generalisability (Shorten & Khoshgoftaar 2019)
- A training and evaluation pipeline with confusion matrix and classification report
- A prediction function for newly uploaded MRI images
- An interactive Streamlit app that allows real-time image upload and classification
- Output of predictions alongside a confidence score to increase interpretability

The model achieved around 74% accuracy and showed reliable performance across both tumour and non-tumour classifications, this was calculated using the classification_report function from scikit-learn, which compares the predicted labels against the true labels in the test set. This means that out of all test images, the model correctly classified about three out of every four MRI scans.

0.3 How I Made It

0.3.1 Realisation (Scope, Feasibility, Tools)

I used a publically accessible dataset from Kaggle to choose a project that strikes a mix between practicality and scope (Chakrabarty 2019). I used transfer learning using the ResNet18 architecture (He et al. 2016), pretrained on ImageNet, rather than creating a neural network from the ground up. For cloud-based GPU access, Google Colab was used for all development, and local preparations were made for further testing and deployment.

0.3.2 Process (Software Engineering Decisions)

I followed a modular and accessible approach from the standpoint of software engineering. I used PyTorch DataLoaders for batching, made a special dataset class to control the loading of MRI images, and organised the model architecture by freezing early layers and redefining the last fully linked layer (Selokar 2024). The Adam optimiser with cross-entropy loss was employed by the system (Kingma & Ba 2015). The robustness of the model was enhanced by data augmentation.

A real-time diagnostic interface was constructed using the trained model, which was saved as a '.pth' file and used in a different context. Because of its simplicity and speed, Streamlit was selected. In order to load the model, convert incoming user photos, and display the outcome along with the confidence score, I wrote an 'app.py' script. The tool is intended for non-technical users and can be accessed through a local browser.

0.4 Code Explanation

0.4.1 Google Colab Training Pipeline

The project begins with mounting Google Drive to ensure access to the image dataset stored in a specific directory. All required Python packages such as PyTorch, torchvision, scikit-learn, and matplotlib are installed and imported.

Data Preparation The images are separated: tumour-positive scans and non-tumour scans. The Python script recursively loads the file paths and assigns class labels based on the folder name ('yes' = tumour, 'no' = no tumour). The dataset is split into training and testing sets using train_test_split from scikit-learn, stratified by label for balance.

A custom PyTorch Dataset class (BrainMRIDataset) is implemented to streamline loading and transforming images. The transforms applied include resizing to 128×128 pixels, random horizontal flipping, slight random rotations, and normalisation. These augmentations improve the model's ability to generalise by introducing variability during training.

Model Architecture The model I used is ResNet18, a CNN architecture pretrained on ImageNet. To fine-tune it for binary classification, the base layers was frozen (preventing their weights from updating during training), and the final fully connected layer was replaced with a new classifier consisting of:

- A linear layer reducing features to 128 units
- ReLU activation
- Dropout for regularisation

- A final linear layer mapping to two output classes

This configuration preserves general image recognition features while allowing training of the task-specific classifier.

Training and Evaluation The model is trained over 10 epochs using cross-entropy loss and the Adam optimiser. During each epoch, the model processes images in mini-batches, updating only the new classifier layers. After training, the model is evaluated on the test set using metrics like accuracy, precision, recall, and confusion matrix- all computed using classification_report and confusion_matrix from scikit-learn.

A custom function predict_image() allows testing individual images by displaying them alongside their prediction and confidence score.

The trained model's weights are saved to a .pth file using torch.save() for later deployment.

0.4.2 Streamlit Deployment App

I also created a web-based deployment using Streamlit, allowing non-technical users to interact with the trained model.

Interface Design The Streamlit script begins by importing the necessary libraries including Streamlit, PIL for image handling, and PyTorch modules.

Model Loading The function load_model() reconstructs the same ResNet18 architecture used during training and loads the saved weights from model.pth. It is wrapped with @st.cache_resource to prevent reloading on every interaction. If the model fails to load, an error message would show.

Image Input and Prediction Users can upload an image, then it is converted to RGB format and displayed. The image is transformed using the same preprocessing pipeline as the training data to ensure consistency.

The image is passed through the model, and a probability distribution over the two classes is produced using softmax. The highest value determines the prediction, and the associated confidence score is shown.

Output Streamlit displays the uploaded image, the prediction (e.g., "Tumour"), and the model's confidence (e.g., "95.32%"). This real-time feedback makes the model accessible and interpretable for a broader audience.

0.5 Why I Made It

This project has personal significance in addition to being technically intriguing. My experience of losing a family member to a brain tumour motivated me to create a technology that aids early diagnosis. My goal was to turn classroom learning into something impactful. The availability of free tools like Google Colab and Streamlit for the prototyping and deployment of such models demonstrates how accessible AI has become for tackling significant medical issues (Esteva et al. 2019, Beam & Kohane 2018).

Several existing tools and frameworks, such as Aidoc (Aidoc 2025), Qure.ai (Qure.ai 2025), and DeepMedic (Kamnitsas et al. 2017), offer AI-driven analysis of brain images, but are designed for use in clinical environments with specialist oversight and focus on segmentation rather than binary classification. Many of these platforms require enterprise-level infrastructure and regulatory approval, making them less accessible for prototyping or deployment in low-resource settings. The system I developed in this project is lightweight, interpretable, and intentionally built for accessibility-using free tools like Google Colab, Streamlit, and open-source datasets. It enables real-time prediction through a simple web interface tailored for non-specialists, aiming to support early screening or educational purposes rather than clinical diagnosis. This emphasis on accessibility, simplicity, and educational impact distinguishes the project from more complex or regulated solutions.

0.6 Impact and Real-World Use Case

This model could help physicians, particularly in places where radiologists are limited (Topol 2019). It's intended to be user-friendly, lightweight, and interpretable. An MRI scan can be uploaded, and a prediction and confidence level could be obtained instantly by a technician or physician. This is useful for early screening in community clinics or for triage in crowded hospitals. The Streamlit interface's ease of use guarantees that the tool can be utilised as an instructional tool or linked into more extensive healthcare platforms (Community 2020).

Bibliography

Aidoc (2025), 'Aidoc: Always-on ai for medical imaging', https://www.aidoc.com.

Beam, A. L. & Kohane, I. S. (2018), 'Clinical artificial intelligence requires evaluation on realistic data', *Nature Biomedical Engineering* **2**(10), 845–847.

Chakrabarty, N. (2019), 'Brain mri images for brain tumor detection', https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection.

Community, S. (2020), 'Deploying a pytorch model - community cloud', https://discuss.streamlit.io/t/deploying-a-pytorch-model/4198.

Contributors, T. (2025), 'Resnet-18 pretrained model', https://pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html.

Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S. & Dean, J. (2019), 'A guide to deep learning in healthcare', *Nature Medicine* **25**(1), 24–29.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 770–778.

Kamnitsas, K., Ledig, C., Newcombe, V. F., Simpson, J. P., Kane, A. D., Menon, D. K., Rueckert, D. & Glocker, B. (2017), Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation, *in* 'Medical Image Analysis', Vol. 36, Elsevier, pp. 61–78.

Kingma, D. P. & Ba, J. (2015), Adam: A method for stochastic optimization, *in* 'Proceedings of the 3rd International Conference on Learning Representations (ICLR)'.

Qure.ai (2025), 'Qure.ai - ai in radiology', https://www.qure.ai.

Selokar, A. (2024), 'Fine-tuning a pre-trained resnet-18 model for image classification on custom dataset with pytorch', https://medium.com/@imabhi1216/fine-tuning-a-pre-trained-resnet-18-model-for-image-classification-on-custom-dataset-with-pytorch-02df12e83c2c.

Shorten, C. & Khoshgoftaar, T. M. (2019), 'A survey on image data augmentation for deep learning', *Journal of Big Data* **6**(1), 1–48.

Topol, E. J. (2019), 'High-performance medicine: the convergence of human and artificial intelligence', *Nature Medicine* **25**(1), 44–56.