National College of Ireland

Brain Tumour Prediction Using CNN Algorithm

Configuration Manual

MSc Research Project

MSc in Data Analytics

Susanth Kumar Athanti Gurunathan

Student ID: 23269138

School of Computing

National College of Ireland

Supervisor: Dr. Christian Horn

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Susanth Kumar Athanti Gurunathan
Student ID:	23269138
Programme:	MSc in Data Analytics (MSCDAD_C) **Year:** 2024 - 2025
Module:	MSc Research Practicum Part 2
Lecturer:	Dr. Christian Horn
Submission Due Date:	15-09-2025
Project Title:	Brain Tumour Prediction Using CNN Algorithm
Word Count:	1344 **Page Count:** 8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Susanth Kumar Athanti Gurunathan
Date:	15-09-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Susanth Kumar Athanti Gurunathan
Student ID: 23269138

1 System Requirements and Environment Setup

1.1 Hardware Specifications

Specification	Minimum Requirements
CPU	Intel Core i5- 11400H
RAM	16GB DDR4-2666 MHz
GPU	NVIDIA GeForce RTX 2050
Storage	200 GB SSD with 100GB free space
Network	Broadband internet connection for package downloads

1.2 Operating System Requirements

Primary Development Environment: Windows 10/11 Professional (Build 19041+)

1.3 Python Environment Configuration

Core Python Setup:

- Version: 3.13
- Pip: 21.3+ or conda 4.12+
- virtual Environment: venv, virtualenv, or conda environment

Python Installation Commands:

```
“python -m venv brain_cancer_env
brain_cancer_env\Scripts\activate
”
```

2 Dependencies and Library Versions

2.1 Core Deep Learning Framework

- TensorFlow: 2.14.0
- Keras: 2.12.0 (included with TensorFlow)
- CUDA Toolkit: 11.8 (to be used when NVIDIA GPUs need to be supported)

Installation Commands:

```
“pip install tensorflow==2.14.0
pip install tensorflow-gpu==2.12.0
”
```

2.2 Data Processing Libraries

NumPy and Scientific Computing:

- numpy -2.3.2
- pandas -2.3.1
- tensorflow==2.14.0
- protobuf>=6.0.0

Computer Vision Libraries:

- pillow -11.3.0
- opencv-python -4.12.0.88
- scikit-image: 0.21.0

Installation Commands:

```
“pip install numpy==2.3.2
pip install scipy==1.10.1
pip install pandas==2.3.1
pip install opencv-python==4.12.0.88
pip install Pillow==11.3.0
pip install scikit-image==0.21.0
”
```

2.3 Machine Learning Libraries

Scikit-learn Ecosystem:

- scikit-learn: 1.3.0
- scikit-metrics: 0.3.3

Model Evaluation Tools:

- classification-report: Integrated with scikit-learn
- confusion-matrix: Integrated with scikit-learn
- precision-recall-fscore: Integrated with scikit-learn

2.4 Visualization Libraries

Plotting and Visualization:

- matplotlib -3.10.5
- seaborn - 0.13.2

Installation Commands:

```
“pip install matplotlib==3.10.5
pip install seaborn==0.13.2
”
```

3 Dataset Configuration and Setup

3.1 Dataset Structure Requirements

```
DATASET_PATH = "../Data/Brain_Cancer mendeley"

# Define the categories
CATEGORIES = ['brain_tumor', 'brain_menin', 'brain_glioma']

def load_dataset_info(dataset_path):
    dataset_info = {
        'category': [],
        'image_count': [],
        'sample_paths': []
    }

    for category in CATEGORIES:
        category_path = os.path.join(dataset_path, category)
        if os.path.exists(category_path):
            image_files = [f for f in os.listdir(category_path)
                           if f.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.tiff'))]

            dataset_info['category'].append(category)
            dataset_info['image_count'].append(len(image_files))
            dataset_info['sample_paths'].append([os.path.join(category_path, f) for f in image_files[:5]])
        else:
            print(f"Warning: Category path {category_path} does not exist")

    return pd.DataFrame(dataset_info)

# Load dataset information
try:
    df_info = load_dataset_info(DATASET_PATH)
    print("Dataset Overview:")
    print(df_info[['category', 'image_count']])
    print(f"\nTotal images: {df_info['image_count'].sum()}")
except Exception as e:
    print(f"Error loading dataset: {e}")
    print("Please ensure the dataset path is correct and the folders exist")
```

Figure 1: Dataset Loading as per the structure

(Jupyter Notebook)

The data are placed in the subdirectory Brain_Cancer_mendeley/, which represents the following three different tumor classes, planned as the key subfolders: brain tumor/ brain menin/ and brain glioma/. All the image files in each subfolder are numbered (image_001.jpg, image_002.jpg), the number of files in brain_tumor/ amounts to 2,048 files, in brain_menin/ it is 2,004 files, and 2,004 images are in brain_glioma/. This organized structure supports loading and processing of classes such that the dataset is balanced and usable to train, validate and test.

3.2 Image Specifications

The dataset allows not only one but several image formats with JPEG (.jpg, .jpeg) being a primary format and PNG (.png), BMP (.bmp), and TIFF (.tiff) being secondary. Original pictures are also 512x512 pixels in size which are converted to the size 224x224 pixels during processing. Images that have been RGBA (the four color channels) are converted into three channels (RGB) to be standard. The set contains 6, 056 images that are split into 3600 images (60 %) out of which 1200 images (20 %) are used to test and validate the images and 1,200 images (20 %) to train the images. The proportion of the classes is even, with about 33.3 percent of the contained images being of each category.

Data Preprocessing Pipeline

All the images are standardized to a set of 224x224 pixels as the preprocessing pipeline. Values of pixels would be scaled down into [0, 1] to introduce stability in the training process and all pictures would be converted into RGB color space into RGBA in relevant cases. The data processed is already saved into the float32 data type so that they can integrate well with the deep learning framework. To achieve a balance between faster computing and low memory, a batch size of 32 samples per batch is applied in training the model.

4 Model Architecture Configurations

4.1 CNN Model Configuration

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	1,184
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_2 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 128)	0
conv2d_3 (Conv2D)	(None, 24, 24, 128)	147,584
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 128)	0
flatten (Flatten)	(None, 18432)	0
dropout (Dropout)	(None, 18432)	0
dense (Dense)	(None, 512)	9,437,696
dropout_1 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 3)	1,539

Total params: 9,680,355 (36.93 MB)
Trainable params: 9,680,355 (36.93 MB)
Non-trainable params: 0 (0.00 B)

Figure 2: Normal CNN model architecture

(Jupyter Notebook)

CNN Model Configuration The configuration of the developed CNN was thoroughly analyzed, as well as how it performed on the different data sets. The used library was Keras which is a python library developed on top of Tensorflow2. The standard CNN architecture accepts (224,224,4) size input and is reduced to (224,224,3) channels. It includes four Conv2D layers with 32, 64, 128 and 128 filters with a kernel size of (3, 3) and a MaxPooling2D layer with (2, 2). There is a thick layer of 512 neurons after which there is an output layer of three neurons that are used to classify the result using ReLU and softmax activations respectively. The dropout with 0.5 rate counteracts an overfit. The architecture has a total of 9,680,355 parameters (36.93 MB), which is highly lightweight and still effective in this case of medical image classification.

4.2 Improved CNN with Batch Normalization

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 222, 222, 32)	1,184
batch_normalization (BatchNormalization)	(None, 222, 222, 32)	128
max_pooling2d_4 (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_5 (Conv2D)	(None, 109, 109, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 109, 109, 64)	256
max_pooling2d_5 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_6 (Conv2D)	(None, 52, 52, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 52, 52, 128)	512
max_pooling2d_6 (MaxPooling2D)	(None, 26, 26, 128)	0
conv2d_7 (Conv2D)	(None, 24, 24, 256)	295,168
batch_normalization_3 (BatchNormalization)	(None, 24, 24, 256)	1,024
max_pooling2d_7 (MaxPooling2D)	(None, 12, 12, 256)	0
flatten_1 (Flatten)	(None, 36864)	0
dropout_2 (Dropout)	(None, 36864)	0
dense_2 (Dense)	(None, 512)	18,874,880
batch_normalization_4 (BatchNormalization)	(None, 512)	2,048
dropout_3 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 3)	1,539

Total params: 19,269,091 (73.51 MB)
Trainable params: 19,267,107 (73.50 MB)
Non-trainable params: 1,984 (7.75 KB)

Figure 3: Enhanced CNN model architecture (Batch Normalization)
(Jupyter Notebook)

Such an enhanced version of CNN involves the addition of an extra Conv2D layer to repeat the ability to extract features with 256 filters. To make the training more stable and converge, Batch Normalization is used against each convolutional layer. The thicker layer with 512 neurons also involves the use of BatchNorm and combines it with drop out to make it more regularized. Such an arrangement has major benefits in enhancing performance as it causes less internal covariate shift and prevents overfitting. It is an architecture of 19,269,091 parameters (73.51 MB) by setting up deeper layers and the use of more effective normalizations, and thus performance seems to be improved in terms of accuracy in classification task, but by a penalty factor of around two of parameters.

4.3 ResNet50 Transfer Learning Configuration

Pre-trained Model Setup:

Model: "functional_3"

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 224, 224, 4)	0
lambda (Lambda)	(None, 224, 224, 3)	0
lambda_1 (Lambda)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense_4 (Dense)	(None, 512)	1,049,088
dropout_4 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 256)	131,328
dropout_5 (Dropout)	(None, 256)	0
dense_6 (Dense)	(None, 3)	771

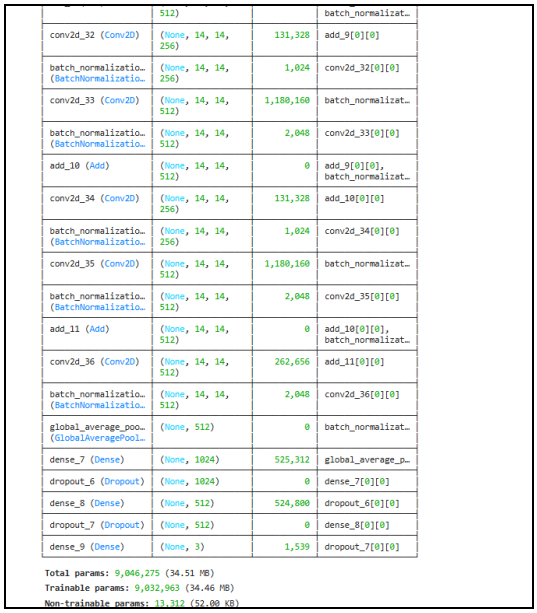
Total params: 24,768,899 (94.49 MB)
Trainable params: 1,181,187 (4.51 MB)
Non-trainable params: 23,587,712 (89.98 MB)

Figure 4: ResNet50 Transfer Learning architecture
(Jupyter Notebook)

This is an architecture with an immobile feature extractor that is ResNet50 pre-trained on ImageNet. The base model accepts input pre-processed into three channel-wise and all 23,587, 712 parameters were frozen to preserve learned weights. It has a custom classification head appended to it which contains a GlobalAveragePooling2D, Dense layer of 512 and 256 neurons and a Dense layer of 3 neurons which perform the classification. The number of trainable parameters is only 1,181,187 (4.51 MB), but the overall number of parameters is 24,768,899 (94.49 MB). The configuration enjoys the strength of deep feature extraction ResNet 50 but runs short potentially given by domain shift of medical images over those of the natural ones.

4.4 YOLOv8-Inspired CNN Architecture

Advanced Architecture Features:



	512)		batch_normalizat…
conv2d_32 (Conv2D)	(None, 14, 14, 256)	131,328	add_9[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv2d_32[0][0]
conv2d_33 (Conv2D)	(None, 14, 14, 512)	1,180,160	batch_normalizat…
batch_normalizatio… (BatchNormalizatio…	(None, 14, 14, 512)	2,048	conv2d_33[0][0]
add_10 (Add)	(None, 14, 14, 512)	0	add_9[0][0], batch_normalizat…
conv2d_34 (Conv2D)	(None, 14, 14, 256)	131,328	add_10[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 14, 14, 256)	1,024	conv2d_34[0][0]
conv2d_35 (Conv2D)	(None, 14, 14, 512)	1,180,160	batch_normalizat…
batch_normalizatio… (BatchNormalizatio…	(None, 14, 14, 512)	2,048	conv2d_35[0][0]
add_11 (Add)	(None, 14, 14, 512)	0	add_10[0][0], batch_normalizat…
conv2d_36 (Conv2D)	(None, 14, 14, 512)	262,656	add_11[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 14, 14, 512)	2,048	conv2d_36[0][0]
global_average_poo… (GlobalAveragePool…	(None, 512)	0	batch_normalizat…
dense_7 (Dense)	(None, 1024)	525,312	global_average_p…
dropout_6 (Dropout)	(None, 1024)	0	dense_7[0][0]
dense_8 (Dense)	(None, 512)	524,800	dropout_6[0][0]
dropout_7 (Dropout)	(None, 512)	0	dense_8[0][0]
dense_9 (Dense)	(None, 3)	1,539	dropout_7[0][0]

Total params: 9,046,275 (34.51 MB)
Trainable params: 9,032,963 (34.46 MB)
Non-trainable params: 13,312 (52.00 KB)

Figure 5: YOLOv8-Inspired CNN architecture

(Jupyter Notebook)

This further CNN is based on the idea of Yolov8; the SiLU (Swish) activation instead of ReLU is used due to its easier gradients and a higher degree of precision. It uses residual connections and bottleneck blocks combined as 1x1 and 3x3 convolution strengths in order to increase the efficiency of feature learning. The feature processing of multi-scale feature brings down the feature maps iteratively providing enhancement in location of objects in medical settings. The dense head comprises of 1024, 512 and three-neuroned layers. The skip connections also enhance flow of gradient in the direction in which the network is easier to train. It contains 9 046 275 parameters (34.51 MB), perfroming well balanced efficiency and classifications power and achieving higher accuracy percentages in neural brain tumors classifications.

5 Training Configuration and Hyperparameters

5.1 Optimization Settings

- The training step uses the Adam optimiser with a learning rate of 0.001, the default Beta1 of 0.9 and Beta2 of 0.999 and default Epsilon of 1e-07.
- Loss functions The general loss to be used is Categorical Crossentropy that is effective in multi-class classification. The Softmax is used as an activation of the top layer of the model that provides class probability distributions.

5.2 Training Parameters

Epoch and Batch Configuration:

The training of the model is done using a batch size of 32 samples and 20 to 100 epochs depending on the architecture and convergence behavior. A validation split of 20% of the training data is used and dataset shuffling is enforced, in order to provide randomness in the training. EarlyStopping and ReduceLROnPlateau are incorporated to reduce training time, with the patient of 8 to 10 epochs and reduction factor of 0.2 and patient of 5 epochs on EarlyStopping and ReduceLROnlPlateau respectively. As a last point, when you set the RestoreBestWeights parameter to True, the model saves the best-performing weights as determined by the validation performance.

5.3 Memory Management

Memory Usage Estimates:

Model Name	Approximate GPU Memory Requirement
Basic CNN	4 GB
Improved CNN	6 GB
ResNet50	~8 GB
YOLOv8-inspired CNN	~8 GB

6 References

Wong, Y., Su, E.L.M., Yeong, C.F., Holderbaum, W. and Yang, C., 2025. Brain tumor classification using MRI images and deep learning techniques. PloS one, 20(5), p.e0322624.

Masood, M., Nazir, T., Nawaz, M., Mehmood, A., Rashid, J., Kwon, H.Y., Mahmood, T. and Hussain, A., 2021. A novel deep learning method for recognition and classification of brain tumors from MRI images. Diagnostics, 11(5), p.744.

TensorFlow (2025) TensorFlow. Available at: https://www.tensorflow.org/ (Accessed: 1 August 2025).

Keras (2025) Keras. Available at: https://keras.io/ (Accessed: 1 August 2025).

Python Software Foundation (2025) Python. Available at: https://www.python.org/ (Accessed: 1 August 2025).

NumPy (2025) NumPy. Available at: https://numpy.org/ (Accessed: 1 August 2025).

SciPy (2025) SciPy. Available at: https://scipy.org/ (Accessed: 1 August 2025).

Pandas (2025) Pandas. Available at: https://pandas.pydata.org/ (Accessed: 1 August 2025).

PyPI (2025) OpenCV-Python. Available at: https://pypi.org/project/opencv-python/ (Accessed: 1 August 2025).

PyPI (2025) Pillow. Available at: https://pypi.org/project/pillow/ (Accessed: 1 August 2025).

Liu, X. and Wang, Z., 2024, July. Deep learning in medical image classification from mri-based brain tumor images. In 2024 IEEE 6th International Conference on Power, Intelligent Computing and Systems (ICPICS) (pp. 840-844). IEEE.

Khan, M.A. and Auvee, R.B.Z., 2024, December. Comparative Analysis of Resource-Efficient CNN Architectures for Brain Tumor Classification. In 2024 27th International Conference on Computer and Information Technology (ICCIT) (pp. 639-644). IEEE.

scikit-image (2025) scikit-image 0.21 release notes. Available at: https://scikit-image.org/docs/0.25.x/release_notes/release_0.21.html (Accessed: 1 August 2025).