Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 8856789, 9 pages
https://doi.org/10.1155/2022/8856789

Research Article

An Efficient CNN for Radiogenomic Classification of Low-Grade Gliomas on MRI in a Small Dataset

Jun Liu[1], **Feng Deng**[2], **Geng Yuan**[3], **Changdi Yang**[3], **Houbing Song**[4] and **Liang Luo**[5]

[1]*Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA*
[2]*Beijing Key Laboratory of High Dynamic Navigation Technology, Beijing Information Science & Technology University, Beijing, China*
[3]*Department of Electrical & Computer Engineering, College of Engineering, Northeastern University, Boston, MA, USA*
[4]*Security and Optimization for Networked Globe Laboratory (SONG Lab), Embry-Riddle Aeronautical University, Daytona Beach, FL, USA*
[5]*School of Computer Science and Engineering, University of Electronic Science and Technology, Chengdu, China*

Correspondence should be addressed to Liang Luo; luoliang@uestc.edu.cn

Received 1 April 2022; Accepted 28 April 2022; Published 6 June 2022

Academic Editor: Maode Ma

Gliomas, often known as low-grade gliomas, are malignant brain tumors. Codeletion of chromosomal arms 1p/19q has been connected with a good response to treatment in low-grade gliomas (LGG) in several studies. For treatment planning, the ability to anticipate 1p19q status is crucial. This research's purpose is to develop a noninvasive approach based on MR images using our efficient CNNs. While public networks like VGGNet, GoogleNet, and other well-known public networks can use transfer learning to identify brain cancer on MRI, the model contains a large number of components that are unrelated to brain tumors. We build a model from the bottom-up, rather than relying on transfer learning. Our network structure flexibly uses a deep convolution stack mixed with dropout and dense operation, which reduces overfitting and enhances performance. We increase the number of samples by augmenting the dataset. The Gaussian noise is introduced during the model training. To address the issue of data imbalance, we use stratified k-fold cross-validation during training to find the best model. Our proposed model is compared with models fine-tuned through transfer learning, such as MobileNetV2, InceptionResNetV2, and VGG16. Our model achieves better results than these models on the same small dataset. In the test set, when deciding whether or not an image should be 1p/19q codeleted, the proposed architecture achieved an F1-score of 96.50%, precision of 96.50%, recall of 96.49%, and accuracy of 96.50%. By comparing with the transfer model, we found that transfer learning does not outperform CNN on a small dataset.

1. Introduction

Low-grade gliomas (LLG) [1] are brain tumors that arise from astrocytes and oligodendrocytes, which are two separate types of brain cells [1]. Low-grade gliomas can cause a variety of symptoms depending on where they are in the brain. The tumor in the area of the brain that governs language may prevent the patient from speaking or understanding. A brain tumor diagnosis can be devastating for patients. The majority of tumors are discovered as a result of a symptom that prompts doctors to perform a brain MRI or CT scan.

MRI is the most effective method for detecting brain malignancies. The scans provide a massive amount of image data. The radiologist examines these images. Tumors of the brain are difficult to diagnose and treat. The sizes and locations of brain tumors vary dramatically. As a result, fully comprehending the nature of the tumor is quite challenging. For MRI analysis, a qualified neurosurgeon is required. The

absence of skilled doctors and a lack of information regarding tumors can make generating reports from MRIs extremely difficult and time-consuming. A manual inspection may be susceptible to errors due to the complexities involved in brain tumors and their characteristics. Machine learning-based automated classification systems have consistently outperformed manual classification.

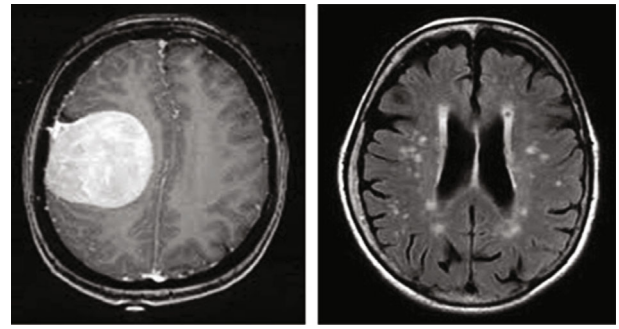
The study of the relationship between cancer imaging features and gene expression is known as radiogenomics. Biomarkers that determine the genetics of a disease without the use of an intrusive biopsy can be created using radiogenomics. A biomarker is a biological indicator of some state or condition. The presence or lack of biomarkers is important in avoiding intrusive biopsies because certain treatments for brain tumors are more successful in the presence or absence of a biomarker. The detection of biomarkers can ensure that patients receive the most effective treatment for their specific situation [2].

Low-grade gliomas (LLG) [2–4] are tumors that are considered formed from glial cells, have infiltrative development, and lack malignant histopathological characteristics. One of the biomarkers that appear to be essential in low-grade gliomas is 1p/19q chromosomal codeletion. When 1p/19q codeletion is discovered in low-grade gliomas, studies demonstrate that they respond better to chemotherapy and radiotherapy. The novelty and promising results of combining deep learning with radiogenomics are what make this study noteworthy. The detection of 1p/19q codeletion using deep learning works better with T2 images than with T1 postcontrast images [2].

In 2017, deep learning was firstly used by Akkus et al. [2] to predict 1p19q from LGG MRI; tumor segmentation, image registration, and CNN-based 1p/19q status classification are the three primary steps of their method. When data augmentation is not performed, their multiscale CNNs overfit the original training data. Lombardi et al. [4] used popular public networks, including AlexNet, VGG19, and GoogleNet, for 1p19q categorization through transfer learning [5–7]. According to their description, even with limited datasets, the results offered by transfer learning are robust. Abiwinanda et al. [8] used five different CNN designs, with the second design with two convolutional layers, one maxpool layer, and one ReLU layer, then come 64 hidden neurons, achieving the highest accuracy.

Why are there just thousands of training examples? Maithra Raghu et al. [9] wondered. They looked upon transfer learning in small data settings. They discovered that there was a significant performance difference between transfer learning and training from scratch for a big model (ResNet), but not for a smaller model. For a little amount of data, the large model built by ImageNet can have too many parameters. They discovered that transfer learning provides limited performance increases for the evaluated medical imaging tasks after a rigorous performance evaluation and examination of hidden representations of neural networks. Transfer learning had little effect on the performance of medical imaging tasks, and the model trained from the ground up was near as well as the ImageNet transfer model.

The following are our main contributions:



Yes No

Figure 1: Brain MRI.

(i) Using the 3×3 convolution and LeakyReLU, we create a dedicated convolutional neural network for detecting brain tumors on MRI images

(ii) During training, we use a customizable combination of dropout and Gaussian noise to reduce overfitting and increase performance

(iii) Stratified k-fold is used to correct problems in training induced by data imbalance

(iv) Our proposed model is compared to MobileNetV2, InceptionResNetV2, and VGG16 that have been fine-tuned through transfer learning

2. Materials and Methods

We use the provided dataset to train our planned network. Meanwhile, on the same dataset, we compare the performance of MobileNetV2, InceptionResNetV2, VGG16, etc., which were fine-tuned using transfer learning approaches.

2.1. Experimental Data. The Kaggle small brain tumor dataset [10] provided the brain MRI dataset that was utilized to evaluate the planned study. The dataset contains 253 brain MRI images in two folders: yes and no. There are 155 tumorous brain MRI images in folder yes, and there are 98 nontumorous brain MRI images in folder no.

Figure 1(a) is a brain with a tumor, and Figure 1(b) is a brain tumor.

2.2. Network Architectures. In Figure 1, there are 14 layers in the model. Convolutional kernels with smaller convolutions—3×3—were found to produce positive outcomes, as these smaller convolutions may capture some of the finer characteristics of the edges. This network's convolutional layers all employ 3×3 kernels. It is starting with 16 kernels per layer; the architecture progresses to 32 kernels per layer, 64 kernels per layer, and finally 128 kernels per layer.

This network depicted in Figure 2 is made up of the convolution layer, pooling layer, dropout layer, LeakyReLU layer, dense layer [11], flatten layer, and softmax layer, with the input picture.

Table 1 specifies the network activities utilized by each layer, as well as the size of the convolution kernel and the size of the input.

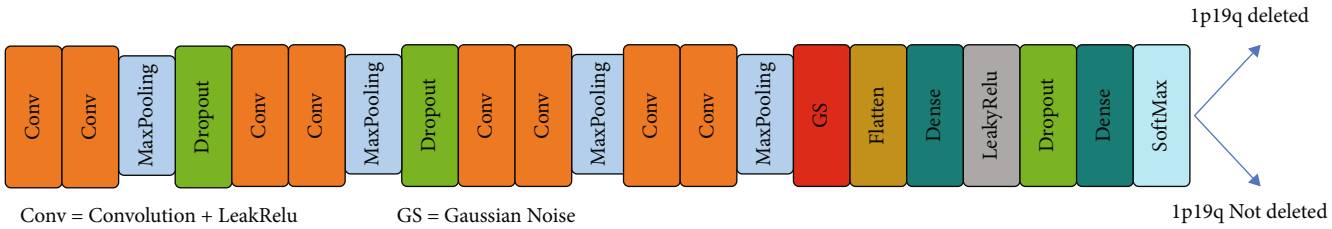


Figure 2: Network architecture.

Table 1: Network layer.

Type	Filter shape	Input size
Input layer	N/A	256 × 256 × 1
Conv2D	16 × 3 × 3	254 × 254 × 16
LeakyReLU	N/A	254 × 254 × 16
Conv2D	16 × 3 × 3	252 × 252 × 16
LeakyReLU	N/A	252 × 252 × 16
Maxpooling	2 × 2	126 × 126 × 16
Dropout	N/A	126 × 126 × 16
Conv2D	32 × 3 × 3	124 × 124 × 32
LeakyReLU	N/A	124 × 124 × 32
Conv2D	32 × 3 × 3	122 × 122 × 32
LeakyReLU	N/A	122 × 122 × 32
Maxpooling	2 × 2	61 × 61 × 32
Dropout	N/A	61 × 61 × 32
Conv2D	64 × 3 × 3	59 × 59 × 64
LeakyReLU	N/A	59 × 59 × 64
Conv2D	64 × 3 × 3	57 × 57 × 64
LeakyReLU	N/A	57 × 57 × 64
Maxpooling	2 × 2	28 × 28 × 64
Conv2D	128 × 3 × 3	26 × 26 × 128
LeakyReLU	N/A	26 × 26 × 128
Conv2D	128 × 3 × 3	24 × 24 × 128
LeakyReLU	N/A	24 × 24 × 128
Maxpooling	5 × 5	4 × 4 × 128
Gaussian noise	N/A	4 × 4 × 128
Flatten	N/A	2048
Dense	N/A	1024
LeakyReLU	N/A	1024
Dropout	N/A	1024
Dense	N/A	2
Softmax	N/A	2

(i) All images of brain tumors that are fed into the network are scaled to 256 × 256

(ii) This network uses 3 × 3 kernels for all convolutional layers. The architecture starts with 16 kernels per layer, then 32, 64, and finally 128 kernels per layer

In Figure 3, we incorporate more hidden layers and therefore more nonlinear functions, enhancing the decision function capabilities and introducing fewer parameters, inspired by a VGGNet stack of three 3 × 3 convolutional layers, instead of a single 7 × 7 layer.

However, our network differs from the VGGNet structure, which is made up of a stack of 3 × 3 convolutions and ReLUs. Our network is made up of 3 × 3 convolutions and LeakyReLUs.

(i) Because negative values are kept and saturation concerns are avoided when employing tanh, LeakyReLU was chosen as the activation function

(ii) This model employs 2 × 2 maxpooling at first, then 7 × 7 maxpooling later. The number of neurons in a layer is reduced when it is dense

(iii) Dense [11] (fully connected) is used twice just before the softmax layer to reduce the number of neurons to two, reflecting the binary prediction of either "codeletion" or "no codeletion"

(iv) Gaussian noise was purposely supplied to the training data to minimize overfitting, which can think of it as a form of random data augmentation. For corrosion processes with genuine inputs, Gaussian noise (GS) is a natural choice. During training, the error is reduced, and at the same time, the interference items generated by noise are penalized to achieve the purpose of reducing the square of the weight. The noise distribution's standard deviation was set to 0.5

The following formula can be used to compute the probability density of a Gaussian distribution:

$$F(x;\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)dx. \quad (1)$$

Assume we introduce Gaussian noise to the inputs in Figure 4. Before moving on to the next layer, the squared weight amplifies the noise variation. The squared error increases as a result of this. When the input is noisy, minimizing the squared error tends to minimize the square of the weights. Let us assume y^{noisy} is output by GS at one time:

$$y^{\text{noisy}}=\Sigma_i w_i x_i+\Sigma_i w_i \varepsilon_i, \quad (2)$$

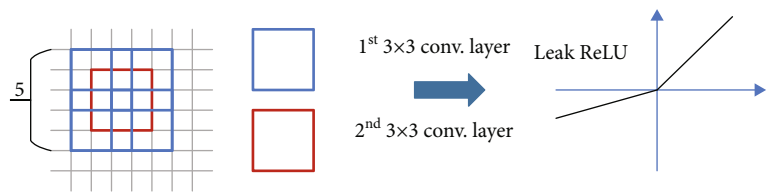


Figure 3: 3 × 3 convolutions and LeakyReLU.

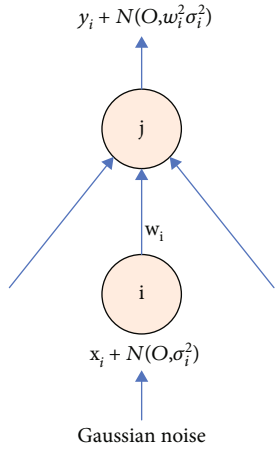


Figure 4: Introduce Gaussian noise to the inputs.

where ε_i is sampled from $N(0,\sigma_i^2)$:

$$\begin{aligned} E\left[\left(y^{\text{noisy}}-t\right)^2\right] &= E\left[\left(y+\Sigma_i w_i\varepsilon_i - t\right)^2\right] \\ &= E\left[\left((y-t)+\Sigma_i w_i\varepsilon_i\right)^2\right] \\ &= (y-t)^2 + E\left[2(y-t)\Sigma_i w_i\varepsilon_i\right] + E\left[\left(\Sigma_i w_i\varepsilon_i\right)^2\right] \\ &= (y-t)^2 + E\left[\Sigma_i w_i^2\varepsilon_i^2\right] \\ &= (y-t)^2 + \Sigma_i w_i^2\sigma_i^2. \end{aligned} \tag{3}$$

Because ε_i is independent of ε_i and ε_i is independent of $(y-t)$, σ_i^2 is equivalent to L2 penalty [12]. The error is minimized during the training process, and the noise-induced interference items are penalized in order to reduce the square of the weight and achieve a comparable result to L2 regularization.

(i) The goal of flatten is to one-dimensionalize the multidimensional input, which is accomplished by transitioning from convolutional to fully connected layers

(ii) During the forward and backward propagation phases, dropout avoids neurons at random. The number of neurons that are not updated is determined by the dropout value. The dropout rate was set to 0.3

(iii) The probability of each of the binary outcomes—"codeletion" and "no codeletion"—is included in the output layer using a softmax classifier

2.3. Hyperparameters. Several hyperparameter values were explored in this study.

2.3.1. Learning Rate. The learning rate is the amount of time we spend moving in a particular direction to find the global minima. Starting with a greater learning rate usually works fine because the initial weight values are rather random. We typically grow closer and closer to either the global or local minima as we proceed through the training phase. Because we do not want to overshoot the minima, annealing, the learning rate is a typical method. To put it in another way, as the training phase progresses, we begin to take smaller and smaller moves in a specific direction. When there is no change in the loss value, we will continue to reduce the learning rate by the square root of 0.1 until it reaches a reduction of $0.5e-6$.

2.3.2. Early Stopping. Overfitting models to training data can be prevented or limited by early stopping techniques. Moreover, when the findings are static, early halting procedures prevent needless computations. If there has not been a change of at least 0.001 in 10 (epochs), the model provided here will terminate training. We usually start with smaller weights when we begin the network. The partial network weights may grow in size as the training time increases. We can limit the network's capabilities to a specific range by stopping training at the appropriate time. The steps are as follows:

(i) The validation set is used to collect test results after every 5 epochs. Stop training if the test error on the validation set increases as the epoch increases

(ii) After stopping, use the weights as the network's final parameters

2.3.3. Batch Size. The batch size specifies how many photos are handled during forward propagation to produce a loss value for backpropagation. Batch size is typically set to a power of two and is restricted by the available memory. Furthermore, while a bigger batch size allows for faster training, weights update less frequently and may not deliver the greatest outcomes. The batch size was set to 16.

2.3.4. Number of Epochs. When training your model, the number of epochs denotes the number of times the complete training dataset is iterated over. Validation determines how

well the model generalizes to new data at the end of each epoch. The number of epochs was set to 32.

2.4. Stratified k-Fold Cross-Validation. k-fold cross-validation is an excellent technique to examine the bias-variance tradeoff and guarantee that the model has low bias and variance. The following is a summary of the testing procedure:

(a) The data from the original is split into k sections at random by unrepeat sampling

(b) Each time, as the test set, one of these is chosen. The other $k-1$, on the other hand, is employed as a model training set

(c) Repeat the second step k times more to give a probability for each subgroup and the rest to be the training set

(d) After each training set, obtain a model

(e) Use this model to test on the relevant test set, as well as to calculate and save the model's evaluation metrics

(f) Using the current k-fold cross-validation procedure, calculate the average of all the test results from the k sets as a measure of the model's accuracy and performance

Choosing a good "k" number ensures that the testing procedure provides the most accurate assessment of the performance of our model. When increasing the number of splits k, the variance increases while the bias decreases. Lowering k, on the other hand, increases the bias while decreasing the variance. The tradeoff between bias and variance is a difficult problem.

The predictors X and response Y can both be written as variables:

$$Y = f(X) + \epsilon \sim \mathcal{N}(0, \sigma_\epsilon). \tag{4}$$

The quadratic error's expected value can then be represented as

$$\mathrm{SE}(x) = E\left[(Y - f(x))^2\right]. \tag{5}$$

After some arithmetic, we get

$$\mathrm{SE}(x) = (E[f(x)] - f(x))^2 + E\left[(f(x) - E[f(x)])^2\right] + \sigma_e^2. \tag{6}$$

In the formula above, $(E[f(x)] - f(x))^2$ is bias^2, $E[(f(x) - E[f(x)])^2$ is variance, and σ_e^2 is irreducible error.

The square of a random variable X should have the following expected value:

$$\begin{aligned} E\left[X^2\right] &= \mathrm{Var}[X] + E[X]^2, \\ E[y] &= E[f + \epsilon] = E[f] = f. \end{aligned} \tag{7}$$

This is how it works:

$$\begin{aligned} E\left[(y-f)^2\right] &= E\left[y^2 + f^2 - 2yf\right] \\ &= E\left[y^2\right] + E\left[f^2\right] - E[2yf] \\ &= \mathrm{Var}[y] + E[y]^2 + \mathrm{Var}[f] + E[f]^2 - 2fE[f] \\ &= \mathrm{Var}[y] + Var[f] + (f - E[f])^2 \\ &= \mathrm{Var}[y] + \mathrm{Var}[f] + E[f-f]^2 \\ &= \sigma^2 + Var[f] + \mathrm{Bias}[f]^2. \end{aligned} \tag{8}$$

The relationship between mistake and the bias-variance tradeoff is depicted in Figure 5. The best model is one that minimizes both bias and variance at the same time, resulting in the lowest error rate. The vertical dashed line represents a model with just the correct level of complexity. This model will have high accuracy ratings on both train and test data, indicating that it is generalizable. We hope that the model's bias and variance are very low, but this is not always possible. We must weigh the benefits and drawbacks and strike a balance. In practice, we use $k = 3$. The most significant benefit of k-fold cross-validation is that all data is used in training and prediction, thereby avoiding overfitting and accurately reflecting the concept of crossover.

Because of the disparity in the number of photos of brain tumors versus healthy brains in the dataset, if we use k-fold on an unbalanced dataset, we may end up with no or very few minority classes in our training data. We utilize stratified k-fold to avoid this issue. Stratified k-fold is a k-fold variant that produces hierarchical folds: each set has nearly the same percentage of samples for each target class as the entire set. If the dataset is divided into four categories and the ratio is 2 : 3 : 3 : 2, the divided sample ratio is approximately 2 : 3 : 3 : 2.

2.5. Experiments. For the experiments, we use TensorFlow as the backend Keras Python package on an Ubuntu 18.04 X86_64 server. One NVIDIA 2080ti GPU is used.

2.5.1. Data Preparation. Data preparation steps are included deleting a third class, standardizing the data, and implementing cross-validation [12], to shuffle the training data. Because this is a small dataset, there were insufficient examples to train the neural network. In addition, data augmentation was useful in addressing the data imbalance issue.

The image is preprocessed before being processed into the proposed structure. The original MR image is scaled to 225×2251 pixels in the first step. Image augmentation techniques such as flipping, mirroring, and rotating are used to generate redundant data for the network, which is frequently used to avoid network overfitting and improve system resilience.

ImageDataGenerator is a Keras class that describes the image data preparation and augmentation setup. We can rotate the image at any angle between 0 and 360 degrees using the ImageDataGenerator class. For flipping along the vertical or horizontal axis, the ImageDataGenerator class has options for horizontal flip and vertical flip. The key advantage of utilizing the Keras ImageDataGenerator class

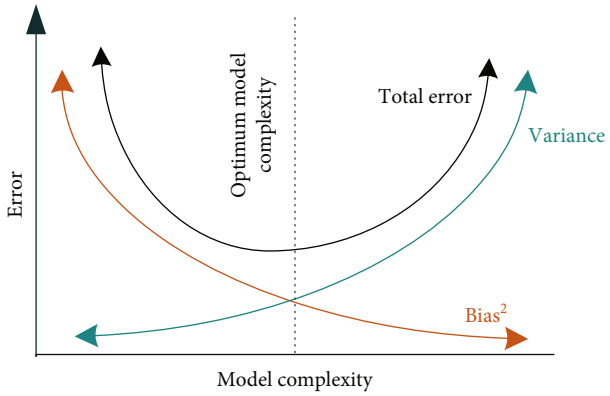


Figure 5: Bias and variance relationship.

is that it is intended for real-time data enhancement. The model generates augmented images on the fly while it is still being trained.

2.5.2. Proposed Workflow. We build, evaluate, and train our model to improve performance and use stratified *k*-fold cross-validation in model training as depicted in Figure 6.

(i) We divided the data into training and testing datasets at random and built the model using the training set and estimated its accuracy using the test set

(ii) Then, we acquire the best quality model by fine-tuning the model via 3-fold cross-validation to enhance the estimate's accuracy

(iii) On the test set, evaluate the model's expected accuracy

(iv) Output evaluation statistics include precision, recall, F1-score, and confusion matrix

2.5.3. Evaluation Method

(1) Confusion Matrix. A confusion matrix is a technique for determining whether or not a classification method is effective. If your dataset has more than two classes or an uneven number of observations in each, classification accuracy alone can be deceiving.

In Table 2, we can clearly see the number of correct identifications and the number of incorrect identifications for each category.

(2) Precision. The model properly predicted the percentage of patients with 1p/19q codeletion based on the total number of patients with 1p/19q codeletion referred to as precision. It has the following formula:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{9}$$

(3) Recall. The fraction of 1p/19q codeleted patients recognized by the model is divided by the total number of 1p/19q codeleted and 1p/19q nondeleted patients to compute recall. It has the following formula:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{10}$$

(4) F1-Score. The purpose of the F1-score was to combine precision and recall measurements into a single value. It is an important metric for class imbalance problems; due to an imbalance in the number of brain and nonbrain tumors in this brain MRI dataset, the F1-score was created to operate effectively with data that is unbalanced. It has the following formula:

$$\text{F1-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \tag{11}$$

3. Results

The harmonic mean of precision and recall is calculated using the F1-score. In relation to all other classes, the scores for each class indicate how accurate the classifier was in classifying the data points in that class. The number of samples of the real answers that fall into that group is the support.

We train the model using stratified 3-fold cross-validation. F1-score, precision, and recall are all factors that must be considered. Table 3 demonstrates that the model achieves good values.

In the test set, we employed 171 photos, 86 of which are 1p19q deleted and 85 of which are 1p19q not deleted; the suggested architecture received an F1-score of 0.9650 in Table 4.

Figure 7 shows the confusion matrix for the classification of 1p19q status on the test set. We can be certain that all 125 1p19q deleted pictures were detected accurately.

We compare pretrained MobileNetV2 [13], Inception-ResNetV2 [14], VGG16 [15], etc., which fine-tuned using the transfer learning approach and other approaches.

Table 5 demonstrates that for classification on small datasets, transfer learning is not superior to ordinary CNN. This is due to the insufficient number of training samples in small datasets to learn complex sets of deep feature sets. With reasonable design, CNNs without transfer learning can attain and surpass transfer learning. Our method yields the best outcomes. Simultaneously, we examine the indicators listed in the above table and discover that the deep learning approach outperforms the machine learning SVM method by a wide margin.

4. Discussion

We provide a reliable and noninvasive approach for predicting 1p/19q chromosomal arm deletion in this work. Having a sufficient amount of datasets is a significant difficulty when applying deep learning approaches to medical imaging. Despite the fact that the initial data amount was limited, our data volume expanded as a result of data augmentation approaches. With larger patient populations and more

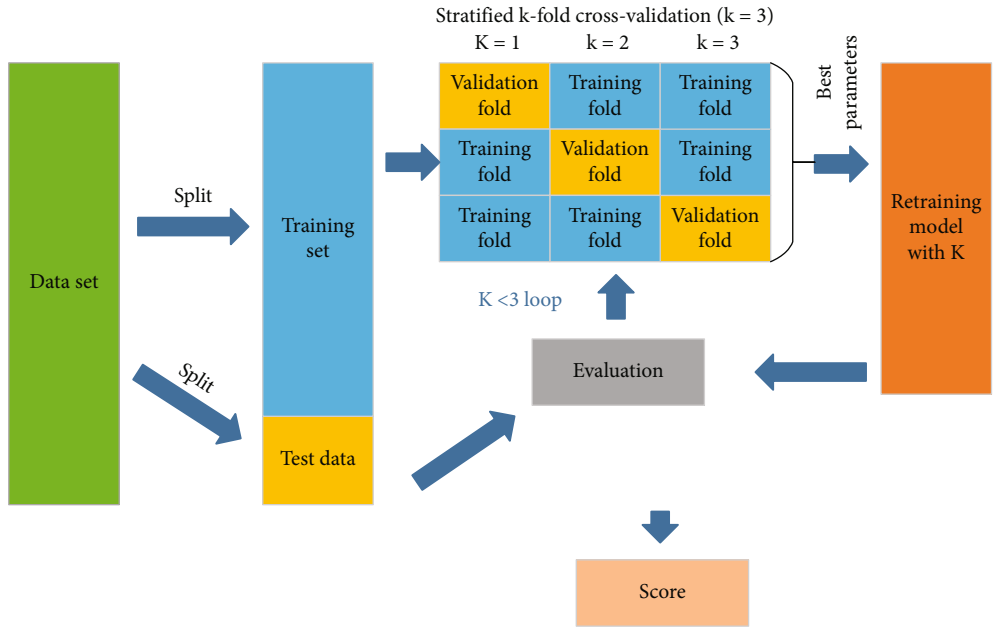


Figure 6: Proposed workflow.

Table 2: Confusion matrix.

	1p19q deleted	1p19q not deleted
1p19q deleted	TP	FP
1p19q not deleted	FN	TN

TP = true positive; FN = false negative; FP = false positive; TN = true negative.

Table 3: Classification performance.

Fold	Train/test	Precision	Recall	F1-score	Support
Fold-1	Train	0.9850	0.9562	0.9704	274
	Test	0.9517	0.9841	0.9598	477
Fold-2	Train	0.9881	0.9679	0.9241	274
	Test	0.9851	0.9635	0.9742	477
Fold-3	Train	0.9886	0.9526	0.9703	274
	Test	0.9517	0.9841	0.9598	477

Table 4: Statistics for test set.

	Precision	Recall	F1-score	Support
1p19q deleted	0.9761	0.9535	0.960	86
1p19q not deleted	0.9540	0.9764	0.970	85
Avg/total	0.9650	0.9649	0.9650	171

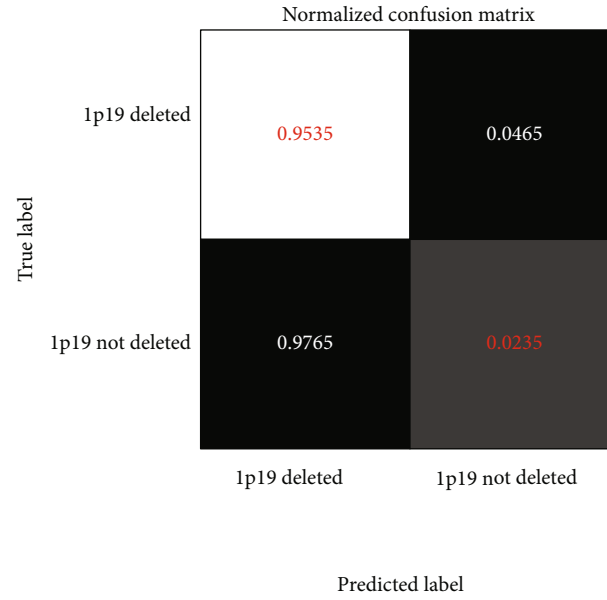


Figure 7: Confusion matrix.

varied data, it is possible that additional performance gains will be gained.

As large convolution kernels are inefficient in terms of cost. We are reducing the number of irrelevant features conceivable by restricting the number of parameters. This drives the deep learning algorithm to learn traits that are common to a variety of scenarios, allowing it to generalize more effectively. Smaller odd-sized kernel filters would be preferable. However, 1×1 is removed from the list of possible ideal filter sizes since the features recovered would be fine-grained and local, with no information from nearby pixels. Furthermore, it does not extract any useful features. Through experiments, we found that although VGG16 also uses a 3×3 convolution kernel, it is prone to overfitting due to the complexity of the network, and the dataset is small. As a result, VGG16 categorization precision and recall of 1p/19q chromosomal arm deletion are not very good.

Table 5: Performance comparison.

Model	Precision	Recall	F1-score	Accuracy
Ours	0.9650	0.9649	0.9650	0.9650
MobileNetV2 [14]	1.0	0.8709	0.9200	0.9200
InceptionV3 [15]	0.923076	1	0.9600	0.9428
InceptionResNetV2 [14]	0.90625	0.9354	0.9153	0.90
AlexNet [16]	0.93620	0.95650	0.9462	0.9483
VGG16 [16]	0.89660	0.9286	0.9123	0.9138
Shwetha and Madhavi [17]	0.89	0.92	0.90	0.88
DenseNet-169 [10]	—	—	—	0.9412
DenseNet-SVM [10]	—	—	—	0.9412
CNNs [18]	—	—	—	0.9500
SVM [19]	0.70	0.71	0.70	0.71

Because of the deep architecture of current networks like GoogleNet and ResNet, feature maps from these networks frequently have a very large receptive field. However, studies [20] reveal that the network gathers information from a considerably narrower portion of the receptive field, which is referred to as the valid receptive field in this research. In this experiment, we found that the recall rate was not high by using InceptionResNetV2 and VGG16. As a result, a large receptive field does not increase the performance of medical images on small datasets considerably.

We discovered that MobileNetV2 is significantly higher than InceptionResNetV2 and VGG16 in the fields of precision. It employs depth-wise separable convolutions and divides an ordinary 3×3 convolution into two convolutions, which is the same as the 3×3 convolution we employ. It makes use of ReLU6. ReLU6 is a standard ReLU with a maximum output limit of 6, allowing for high numerical resolution even when the mobile device's float16/int8 accuracy is low. However, ReLU6 is not as accurate to the server as the LeakyReLU we used.

The model's capacity to learn mapping rules from the input space can be increased by adding Gaussian noise during training, as can the model's generalization ability and fault tolerance. Because the training samples change frequently, adding noise to the network can lead it to lose track of them, resulting in smaller network weights and a more robust network while lowering the generalization error. Since new samples are selected from the domain adjacent to known samples, the structure of the input space is smoothed. This smoothing may make the learning mapping function easier for the network, leading to better and faster learning. After adding Gaussian noise to our model training, we can see significant improvement in performance.

Since medical imaging data is scarce, transfer learning approaches are used to fine-tune medical imaging models using popular public models (e.g., VGGNet and GoogleNet) generated from large public ImageNet datasets. However, these models create a large number of characteristics that are unrelated to medical imaging, jeopardizing the accuracy of medical diagnosis [21]. Our model does not involve transfer learning, and the parameters it generates are specific to the medical imaging dataset that was used. As a result, the reliability of brain tumor diagnosis has substantially improved. Simultaneously, we discover that our method beats transfer learning on small datasets but that transfer learning performs better on large datasets.

5. Conclusion

The results of our CNN approach for 1p/19q codeletion status classification noninvasively are promising. We create a brain tumor detection model that does not rely on transfer learning. Our network structure employs a deep convolution stack strategy when training with Gaussian noise, reducing overfitting and improving performance. Compared to transfer learning models, our model gives more accurate findings. With basic, lightweight models equivalent to ImageNet topology, we discovered that transfer learning offered no performance benefit in small datasets. By properly designing the network and optimizing the hyperparameters during training, CNNs without transfer learning can reach and surpass transfer learning.

Data Availability

This study makes use of datasets that are freely available to the public. This dataset can be found at the following link: https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection.

Conflicts of Interest

There are no conflicts of interest declared by the authors.

Acknowledgments

The National Natural Science Foundation of China (62176048), The National Natural Science Foundation of China (41871348), Beijing Information Science & Technology University (2020KYNH224), and Beijing Key Laboratory of High Dynamic Navigation Technology (HDN2019002) all contributed to this work.

References

[1] J. Amin, M. Sharif, A. Haldorai, M. Yasmin, and R. S. Nayak, "Brain tumor detection and classification using machine learning: a comprehensive survey," *Complex & Intelligent Systems*, 2021.

[2] Z. Akkus, I. Ali, J. Sedlář et al., "Predicting deletion of chromosomal arms 1p/19q in low-grade gliomas from MR images using machine intelligence," *Journal of Digital Imaging*, vol. 30, no. 4, pp. 469–476, 2017.

[3] D. Bhattacharya, N. Sinha, and J. Saini, "Determining chromosomal arms 1p/19q co-deletion status in low graded glioma by cross correlation-periodogram pattern analysis," *Scientific Reports*, vol. 11, no. 1, p. 23866, 2021.

[4] G. Lombardi, V. Barresi, A. Castellano et al., "Clinical management of diffuse low-grade gliomas," *Cancers*, vol. 12, no. 10, article 3008, 2020.

[5] R. Chelghoum, A. Ikhlef, A. Hameurlaine, and S. Jacquir, "Transfer learning using convolutional neural network architectures for brain tumor classification from MRI images," in *IFIP Advances in Information and Communication Technology*, Springer, 2020.

[6] M. F. Alanazi, M. U. Ali, S. J. Hussain et al., "Brain tumor/mass classification framework using magnetic-resonance-imaging-based isolated and developed transfer deep-learning model," *Sensors*, vol. 22, no. 1, p. 372, 2022.

[7] Z. N. K. Swati, Q. Zhao, M. Kabir et al., "Brain tumor classification for MR images using transfer learning and fine- tuning," *Computerized Medical Imaging and Graphics*, vol. 75, pp. 34–46, 2019.

[8] N. Abiwinanda, M. Hanif, S. T. Hesaputra, A. Handayani, and T. R. Mengko, *Brain Tumor Classification Using Convolutional Neural Network World Congress on Medical Physics and Biomedical Engineering 2018*, Springer, Singapore, 2019.

[9] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, "Transfusion: understanding transfer learning for medical imaging," *Advances in neural information processing systems*, vol. 32, 2019.

[10] J. Kang, Z. Ullah, and J. Gwak, "MRI-based brain tumor classification using ensemble of deep features and machine learning classifiers," *Sensors*, vol. 21, no. 6, article 2222, 2021.

[11] *Keras API Reference / Layers API / Core Layers / Dense Layer*- March 2022, https://keras.io/api/layers/core_layers/dense/.

[12] A. Panigrahi and M. R. Patra, "Chapter 6 - network intrusion detection model based on fuzzy-rough classifiers," in *Handbook of Neural Computation*, P. Samui, S. Sekhar, and V. E. Balas, Eds., pp. 109–125, Academic Press, 2017.

[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, Salt Lake City, UT, USA, 2018.

[14] F. Taghiyev, *Brain Tumor Detection Using TensorFlow 2.x*, 2022, March 2022, https://www.kaggle.com/code/faridtaghiyev/brain-tumor-detection-using-tensorflow-2-x/notebook.

[15] M. F. I. Soumik, *Brain Tumor Detection InceptionV3*, 2022, March 2022, https://www.kaggle.com/code/mdfarhanisraksoumik/brain-tumor-detection-inceptionv3-auc-99-84.

[16] S. Kuraparthi, M. K. Reddy, C. N. Sujatha et al., "Brain tumor classification of MRI images using deep convolutional neural network," *Traitement du Signal*, vol. 38, no. 4, pp. 1171–1179, 2021.

[17] V. Shwetha and C. H. R. Madhavi, "Classification of brain tumors using hybridized convolutional neural network in brain MRI images," *International Journal of Circuits, Systems and Signal Processing*, vol. 16, pp. 561–570, 2022.

[18] A. Sinha, *Brain Tumour Detection with CNN 96% Accuracy*, 2022, March 2022, https://www.kaggle.com/code/ethernext/brain-tumour-detection-with-cnn-96-accuracy/notebook.

[19] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters ——improve semantic segmentation by global convolutional network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4353–4361, Honolulu, Hawaii, USA, 2017.

[20] Brendon Im, 2022, March 2022, https://www.kaggle.com/code/brendonim/brain-mri-tumor-detection-using-svm.

[21] J. Liu, F. Deng, G. Yuan, X. Lin, H. Song, and Y. Wang, "An explainable convolutional neural networks for automatic segmentation of the left ventricle in cardiac MRI," in *Proceedings of the CECNet 2021*, Beijing, China, 2021.