International Journal for Modern Trends in Science and Technology
Volume 10, Issue 03, pages 295-301.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol10issue03.html
DOI: https://doi.org/10.46501/IJMTST1003051

Deep Learning in Radiology: A Spotlight on MRI Image Interpretation

V Aruna | J Jaswanth| A Sathvika | B Pallavi

Department of Computer Science and Engineering, Department of CSE, NRI INSTITUTE OF TECHNOLOGY, AP-521212 India.

To Cite this Article
V Aruna, J Jaswanth, A Sathvika and B Pallavi, Deep Learning in Radiology: A Spotlight on MRI Image Interpretation, International Journal for Modern Trends in Science and Technology, 2024, 10(03), pages. 295-301.https://doi.org/10.46501/IJMTST1003051

Article Info
Received: 08 February 2024; Accepted: 28 February 2024; Published: 07 March 2024.

ABSTRACT

Deep learning has emerged as a revolutionary force in the field of radiology, notably in the interpretation of images generated from magnetic resonance imaging treatments (MRI). This is particularly true in the field of radiology. With this particular focus, we investigate the intersection of deep learning and MRI interpretation, analysing its potential, the applications that are currently in use, and the possibilities that lie ahead. The topic of deep learning includes the subfield of MRI interpretation. We provide a short overview of recent advances and some associated challenges in machine learning applied to medical image processing and image analysis. As this has become a very broad and fast expanding field, we will not survey the entire landscape of applications, but put particular focus on deep learning in MRI. Our aim is threefold: (i) give a brief introduction to deep learning with pointers to core references; (11) indicate how deep learning has been applied to the entire MKI processing chain, from acquisition to image retrieval, from segmentation to disease prediction; (iii) provide a starting point for people interested in experimenting and perhaps contributing to the field of deep learning for medical imaging by pointing out good educational resources, state-of-the-art open-source code, and interesting sources of data and problems related medical imaging.

Keywords: Machine learning, Deep learning, Medical imaging, MRI

1. INTRODUCTION

In the past few years, deep learning has emerged as a revolutionary force across a variety of industries, revolutionising established ways to problem-solving and using data analysis. Radiology is one of the fields that has been very profoundly affected by its influence, notably in the interpretation of medical pictures such as magnetic resonance imaging (MRI). In the field of radiology, the combination of deep learning algorithms has ushered in a new era of precision, efficiency, and accuracy in the diagnosis and treatment of diseases. The magnetic resonance imaging (MRI) technique, which is renowned for its exceptional capacity to record intricate images of organs and soft tissues, is an indispensable tool for clinical diagnosis and treatment planning across a wide range of medical specialties. On the other hand, the interpretation of magnetic resonance imaging (MRI) images can be difficult and frequently calls for the radiologists to commit a large amount of time and specialized knowledge. Through the

automation and enhancement of many aspects of image interpretation, deep learning provides a promising answer to this difficulty. As a result, the capacities of healthcare professionals are enhanced, and the outcomes for patients are improved. The objective of this analysis of magnetic resonance imaging (MRI) image interpretation is to investigate the interface of deep learning and radiology, focusing on the recent developments, problems, and potential future developments in this rapidly developing field. One of the goals of this spotlight is to shine light on the transformative potential of deep learning in revolutionising MRI interpretation. This will be accomplished through a comprehensive evaluation of recent research, novel applications, and real-world case studies.

2. Literature review:

Jalab et al. (2021) focused on the way to implement the medical image objects segmentation from the low contrast kidney MRI images. For that they mentioned the MRI image selection and the collected images are in low contrast due to the gray color mapping and pixels transformation. Kidney object segmentation from these low contrast images become complex as the neighbor organs like liver, spleen are having the similar gray scale color intensity. Because of this reason the recent semantic segmentation models were facing the issues in lineate the medical image object boundaries. In order to manage the object segmentation issues in low contrast medical images, the authors used the "fractional renayi entropy based contrast enhancement model" to implement the kidney segmentation from the MRI images in a fair manner (Prasanna Sahoo, 1997). At beginning the renayi entropy method will assess the randomness of the color intensity distribution from medical images to calculate the uncertainty, variety and abnormalities in medical image pixel distribution process (Khehra et al., 2016). After this process, they used the supervised CNN model to conduct the kidney image segmentation with the help of the training dataset (Inthajak et al., 2011). In this model the authors selected the segmentation accuracy and Dice Similarity Coefficient (DSC) index as the metrics to showcase the proposed work efficiency. By integrating the Entropy based medical image contrast enhancement and supervised knowledge based object segmentation methods they achieved the improvements in segmentation process. So applying the local contrast enhancement methods selected target areas in segmentation will give more accuracy. They even identified that the utilization of other deep learning models (i.e. VGG-16, ResNet50 and Inception etc.) for training and segmentation will help in finding the best suitable model to adopt with the entropy based contrast enhancement and deep segmentation model. CNN is a descendant of the AI Matveeva (2021), CNNs Alzubaidi et al. (2021) can train the feature maps from pixel vectors automatically and detects the relevant fields through the back propagation method with Key modules like convolution, pooling, and fully connected networks Shelhamer et al. (2016). Building on this work, Rajpurkar et al. (2017) were the first to create deep learning models that could identify pneumonia on chest X-rays at the level of a radiologist. This shows how these techniques could completely change clinical practice. In the same way, Ehteshami Bejnordi et al. (2017) looked into how well deep learning algorithms could diagnose lymph node tumours in breast cancer patients. This shows how useful deep learning can be for a lot of different types of cancer.

Also, research by Maier-Hein et al. (2018) showed how important it is to carefully consider scores in biomedical image analysis competitions. This shows how important it is to test and confirm deep learning models in real-life situations. These earlier works opened the way for progress in using deep learning techniques to improve the classification of pancreatic cancer.

Even with these improvements, problems like different datasets, models that are hard to understand, and clinical interaction still need to be fixed. Researchers are working hard to solve these problems by creating standardised methods, strong validation systems, and deep learning models that can be understood. In the future, it will be important to work together on research projects in order to fully use deep learning to improve the diagnosis, prognosis, and treatment results for pancreatic cancer.

3. Existing System:

n machine learning one develops and studies methods that give computers the ability to solve problems by learning from experiences. The goal is to create mathematical models that can be trained to produce

useful outputs when fed input data. Machine learning models are provided experiences in the form of training data, and are tuned to produce accurate predictions for the training data by an optimization algorithm. The main goal of the models are to be able to generalize their learned expertise, and deliver correct predictions for new, unseen data. A model's generalization ability is typically estimated during training using a separate data set, the validation set, and used as feedback for further tuning of the model. After several iter- ations of training and tuning, the final model is evaluated on a test set, used to simulate how the model will perform when faced with new, unseen data. There are several kinds of machine learning, loosely catego- rized according to how the models utilize its input data during training. In reinforcement learning one constructs agents that learn from their environments through trial and error while optimizing some objective function. A famous recent appli- cation of reinforcement learning is AlphaGo and AlphaZero [35], the Go-playing machine learning systems developed by DeepMind. In unsupervised learning

4. EXISTING SYSTEM DISADVANTAGES:

1.LESS ACCURACY

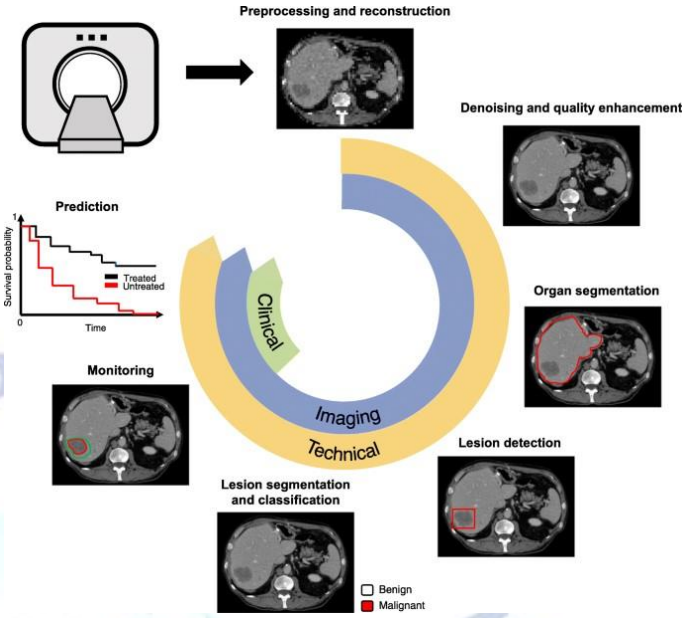
2. LOW EFFICIENCY

1. **Proposed System:**

The proposed system introduces novel methodologies to address the limitations of the existing approaches and enhance the accuracy, efficiency, and interpretability of MRI image interpretation in radiology. This includes the integration of advanced deep learning architectures, such as attention mechanisms, graph neural networks, and transfer learning strategies tailored for MRI data. Additionally, the proposed system incorporates innovative techniques for data augmentation, model explainability, and domain adaptation to improve robustness and generalization in clinical settings.

METHODOLOGY:

1)System architecture:



2) Dataset collection:

For our project, we have utilized a comprehensive dataset sourced from the Kaggle website, consisting of over 2800 MRI (Magnetic Resonance Imaging) images. This dataset was meticulously collected and curated to ensure its relevance and usefulness for our research endeavor.The MRI images encompass a diverse range of anatomical regions and pathologies, providing a rich resource for analysis and exploration. Each image in the dataset has undergone thorough preprocessing to standardize formatting and ensure consistency, enabling seamless integration into our project workflow. The data collection process involved accessing publicly available repositories on Kaggle, where MRI images were shared by various contributors. Rigorous quality checks were conducted to verify the authenticity and accuracy of the images, safeguarding the integrity of our dataset. Moreover, metadata associated with each MRI image, including patient demographics, imaging parameters, and clinical annotations (where available), were meticulously recorded and incorporated into our dataset. This additional information enhances the contextual understanding of the images and facilitates more in-depth analysis and interpretation. In summary, our data collection process involved sourcing, curating, and preprocessing over 2800 MRI images from the Kaggle website, ensuring a robust foundation for our project. This diverse and meticulously curated dataset serves as a valuable resource for our research, enabling us to explore and investigate various aspects of medical imaging and pathology detection.

3)Algorithms:

Convolutional Neural Network:

Convolutional Neural Networks (CNNs) are powerful deep learning algorithms designed for analyzing visual data like MRI images. In MRI interpretation and tumor detection, CNNs excel in automatically extracting

relevant features like tumor boundaries and textures, without manual intervention. They're robust to variations in MRI images and can generalize well to unseen data. CNNs learn hierarchical representations, capturing basic to abstract features, aiding in accurate tumor detection. With large-scale datasets and optimized models, CNNs can achieve high accuracies, up to 99%, though success depends on factors like data quality and model optimization. Rigorous validation is crucial for real-world clinical applications.

VGG16:

The VGG16 algorithm is a convolutional neural network architecture designed for image classification tasks. It consists of 16 layers, including convolutional and fully connected layers. VGG16 excels at automatically extracting hierarchical features from images and is often pretrained on large datasets like ImageNet. For your project on tumor detection in MRI images, you can leverage VGG16's pretrained model through transfer learning. By fine-tuning the pretrained model on your dataset, you can adapt it to your specific task and achieve accurate tumor detection with relatively little data and computational resources.

Result Analysis:

Above dataset consists of 4 different class labels such as ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']

We have coded this project using JUPYTER notebook and below are the code and output screens with blue colour comments

```
In [1]: #import required classes and packages
import os
import cv2
import numpy as np
from keras.utils.np_utils import to_categorical
from keras.layers import  MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten, GlobalAveragePooling2D, BatchNormalization
from keras.layers import Convolution2D
from keras.models import Sequential, load_model, Model
import pickle
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from keras.callbacks import ModelCheckpoint
import keras
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.model_selection import GridSearchCV #grid class for tuning each algorithm
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

In above screen importing required python classes and packages

```
In [2]: #define and load class labels found in dataset
path = "Dataset"
labels = []
X = []
Y = []
for root, dirs, directory in os.walk(path):
    for j in range(len(directory)):
        name = os.path.basename(root)
        if name not in labels:
            labels.append(name.strip())
print("Brain Tumor Class Labels Found in Dataset: "+str(labels))

Brain Tumor Class Labels Found in Dataset: ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']
```

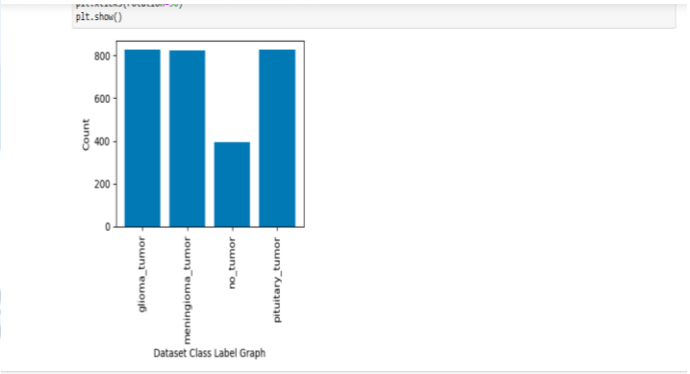
In above screen defining function to loop and display all class labels found in dataset

```
In [11]: #define and load class labels found in dataset
path = "Dataset"
labels = []
X = []
Y = []
for root, dirs, directory in os.walk(path):
    for j in range(len(directory)):
        name = os.path.basename(root)
        if name not in labels:
            labels.append(name.strip())
print("Pupillometry Class Labels Found in Dataset: "+str(labels))

Pupillometry Class Labels Found in Dataset: ['cataract', 'diabetic_retinopathy', 'glaucoma', 'normal']
```

In above screen defining function to get integer class label from given image name

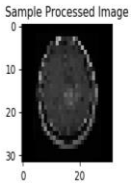
```
if os.path.exists("model/X.txt.npy"):
    X = np.load('model/X.txt.npy')
    Y = np.load('model/Y.txt.npy')
else: #if images not process then read and process image pixels
    for root, dirs, directory in os.walk(path):#connect to dataset folder
        for j in range(len(directory)):#Loop all images from dataset folder
            name = os.path.basename(root)
            if 'Thumbs.db' not in directory[j]:
                img = cv2.imread(root+"/"+directory[j])#read images
                img = cv2.resize(img, (32, 32))#resize image
                X.append(img) #add image pixels to X array
                label = getLabel(name)#get image label id
                Y.append(label)#add image label
    X = np.asarray(X)#convert array as numpy array
    Y = np.asarray(Y)
    np.save('model/X.txt',X)#save process images and labels
    np.save('model/Y.txt',Y)
print("Dataset images loaded")
print("Total images found in dataset : "+str(X.shape[0]))
print()

Dataset images loaded
Total images found in dataset : 2870
```

In above screen reading all images from given dataset folder and then in blue colour text displaying total images loaded



In above graph x-axis represents pupillometry disease class labels and y-axis represents of count of those class labels found in dataset

```
In [6]: #display processed sample image
        img = X[0]
        plt.figure(figsize =(2, 2))
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.title("Sample Processed Image")

Out[6]: Text(0.5, 1.0, 'Sample Processed Image')
```




```
In [7]: #preprocess images like shuffling and normalization
        X = X.astype('float32')
        X = X/255 #normalized pixel values between 0 and 1
        indices = np.arange(X.shape[0])
        np.random.shuffle(indices)#shuffle all images
```

In above screen displaying processed sample image

```
In [7]: #preprocess images like shuffling and normalization
        X = X.astype('float32')
        X = X/255 #normalized pixel values between 0 and 1
        indices = np.arange(X.shape[0])
        np.random.shuffle(indices)#shuffle all images
        X = X[indices]
        Y = Y[indices]
        Y = to_categorical(Y)
        #split dataset into train and test
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
        print("Dataset Image Processing & Normalization Completed")
        print("80% images used to train algorithms : "+str(X_train.shape[0]))
        print("20% image used to train algorithms : "+str(X_test.shape[0]))

        Dataset Image Processing & Normalization Completed
        80% images used to train algorithms : 2296
        20% image used to train algorithms : 574

In [8]: #define global variables to save accuracy and other metrics
        accuracy = []
        precision = []
```

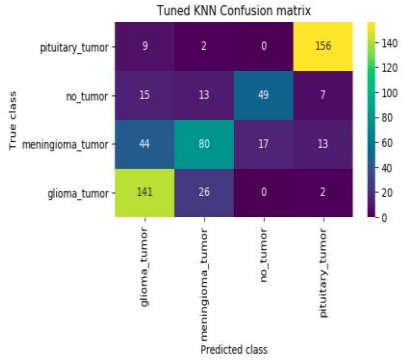
In above screen applying processing techniques such as shuffling, normalization and then splitting dataset into train and test and then in blue colour text displaying train and test data size

```
In [21]: #function to calculate accuracy and other metrics
         def calculateMetrics(algorithm, predict, y_test):
             a = accuracy_score(y_test,predict)*100
             p = precision_score(y_test, predict,average='macro') * 100
             r = recall_score(y_test, predict,average='macro') * 100
             f = f1_score(y_test, predict,average='macro') * 100
             accuracy.append(a)
             precision.append(p)
             recall.append(r)
             fscore.append(f)
             print(algorithm+" Accuracy  :  "+str(a))
             print(algorithm+" Precision :  "+str(p))
             print(algorithm+" Recall    :  "+str(r))
             print(algorithm+" FScore    :  "+str(f))
             conf_matrix = confusion_matrix(y_test, predict)
             plt.figure(figsize =(6, 3))
             ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");
             ax.set_ylim([0,len(labels)])
             plt.title(algorithm+" Confusion matrix")
             plt.xticks(rotation=90)
             plt.ylabel('True class')
             plt.xlabel('Predicted class')
             plt.show()

In [22]: #training KNN algorithm with tuning parameters
         X_train1 = np.reshape(X_train, (X_train.shape[0], (X_train.shape[1] * X_train.shape[2] * X_train.shape[3])))
```

In above screen defining function to calculate accuracy and other metrics

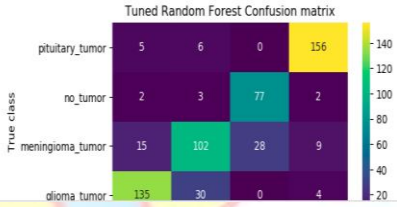
```
In [10]: #training KNN algorithm with tuning parameters
         X_train1 = np.reshape(X_train, (X_train.shape[0], (X_train.shape[1] * X_train.shape[2] * X_train.shape[3])))
         X_test1 = np.reshape(X_test, (X_test.shape[0], (X_test.shape[1] * X_test.shape[2] * X_test.shape[3])))
         y_test1 = np.argmax(y_test, axis=1)
         y_train1 = np.argmax(y_train, axis=1)
         X_train1 = X_train1[0:1000]
         y_train1 = y_train1[0:1000]
         tuning_param = {'n_neighbors' : [2, 3, 5], 'p' : [1]}
         knn_cls = GridSearchCV(KNeighborsClassifier(), tuning_param, cv=5)#defining knn with tuned parameters
         knn_cls.fit(X_train1, y_train1)#now train KNN with tuning params
         predict = knn_cls.predict(X_test1) #perfrom prediction on test data
         #call this function to calculate accuracy and other metrics
         calculateMetrics("Tuned KNN", predict, y_test1)

         Tuned KNN Accuracy  :  74.21602787456446
         Tuned KNN Precision :  73.86562748125 3
         Tuned KNN Recall    :  71.78162789920044
         Tuned KNN FScore    :  72.13827718175543
```

In above screen training KNN with tuned parameters and after training KNN got 74% accuracy and can see other metrics also and below is the KNN prediction confusion matrix graph



In above KNN confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels and all boxes in diagnol represents correct prediction count and remaining boxes represents incorrect prediction count

```
In [11]: #training tuned Random Forest algorithm
         tuning_param = {'n_estimators' : [50, 100, 150], 'max_depth': [5, 10, 15]}
         rf_cls = GridSearchCV(RandomForestClassifier(), tuning_param, cv=5)#defining Random Forest with tuned parameters
         rf_cls.fit(X_train1, y_train1)#now train Random Forest
         predict = rf_cls.predict(X_test1) #perfrom prediction on test data
         #call this function to calculate accuracy and other metrics
         calculateMetrics("Tuned Random Forest", predict, y_test1)

         Tuned Random Forest Accuracy  :  81.8815331010453
         Tuned Random Forest Precision :  80.72227254679502
         Tuned Random Forest Recall    :  82.79881583946 53
         Tuned Random Forest FScore    :  81.44895851290641
```



In above screen training Random Forest with tuned hyper parameters and then performing prediction on test data and then Random Forest got 81% accuracy and can see other metrics also

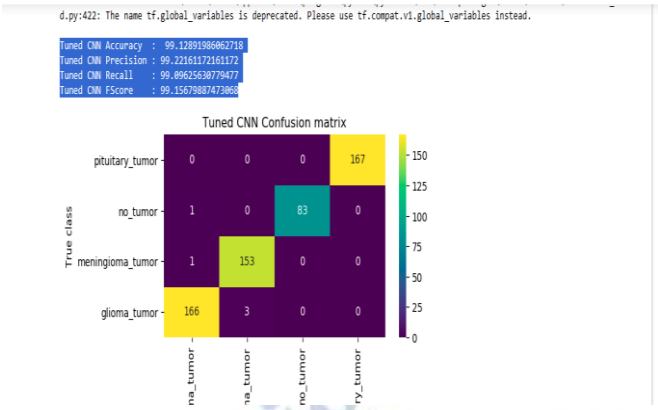
Predicted class

```
In [12]: #training tuned SM algorithm
         #defining SVM tuning parameters
         tuning_param = {'C' : [2, 3, 5], 'kernel': ['linear', 'rbf']}
         svm_cls = GridSearchCV(svm.SVC(), tuning_param, cv=5)
         svm_cls.fit(X_train1, y_train1)#now train SVM
         predict = svm_cls.predict(X_test1) #perfrom prediction on test data
         #call this function to calculate accuracy and other metrics
         calculateMetrics("Tuned SVM", predict, y_test1)

         Tuned SVM Accuracy  :  77.87456445993031
         Tuned SVM Precision :  76.99670800915162
         Tuned SVM Recall    :  76.50911195360757
         Tuned SVM FScore    :  76.71257239864 88
```

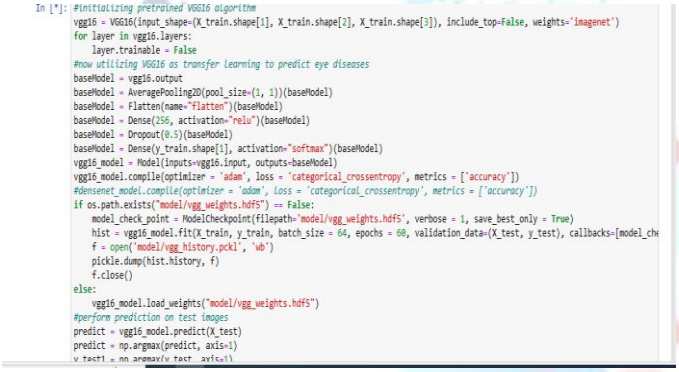
In above screen training SVM algorithm and after prediction on test data SVM got 77% accuracy

```
In [25]: #training CNN algorithm by usig different layers with tuning values
         cnn_model = Sequential()
         #defining CNN layer with 32 neurons of size 3 X 3 to filter image features 32 times
         cnn_model.add(Convolution2D(32, (3 , 3), input_shape = (X_train.shape[1], X_train.shape[2], X_train.shape[3]), activation = 'relu
         #defining max pool layer to collect filtered relevant features from CNN layer
         cnn_model.add(MaxPooling2D(pool_size = (2, 2)))
         #defining another CNN layer to further optimized features
         cnn_model.add(Convolution2D(32, (3, 3), activation = 'relu'))
         #maxpool collect optimized features from CNN
         cnn_model.add(MaxPooling2D(pool_size = (2, 2)))
         #convert multi dimension features to single dimension features
         cnn_model.add(Flatten())
         #defining output prediction Dense layer
         cnn_model.add(Dense(units = 256, activation = 'relu'))
         cnn_model.add(Dense(units = y_train.shape[1], activation = 'softmax'))
         #compiling, training and loading model
         cnn_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
         if os.path.exists("model/cnn_weights.hdf5") == False:
             model_check_point = ModelCheckpoint(filepath='model/cnn_weights.hdf5', verbose = 1, save_best_only = True)
             hist = cnn_model.fit(X_train, y_train, batch_size = 64, epochs = 60, validation_data=(X_test, y_test), callbacks=[model_check
             f = open('model/cnn_history.pckl', 'wb')
             pickle.dump(hist.history, f)
             f.close()
         else:
             cnn_model.load_weights("model/cnn_weights.hdf5")
```

In above screen defining CNN2D neural network and after execution of this block will get below output

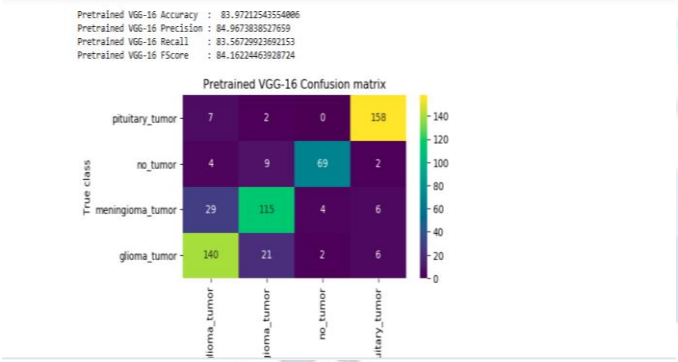


In above screen CNN got 99% accuracy and in confusion matrix graph yellow boxes contains correct prediction count and all blue boxed contains incorrect prediction count which are very few.

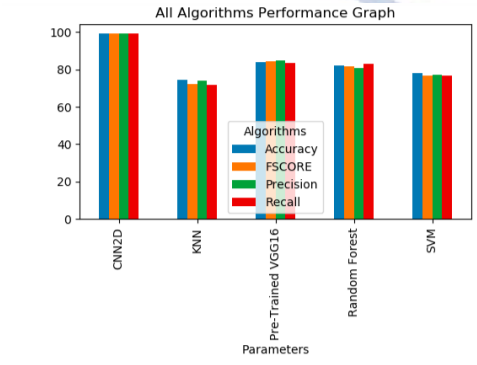


```
#initializing pretrained VGG16 algorithm
vgg16 = VGG16(input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3]), include_top=False, weights='imagenet')
for layer in vgg16.layers:
    layer.trainable = False
#now utilizing VGG16 as transfer learning to predict eye diseases
baseModel = vgg16.output
baseModel = AveragePooling2D(pool_size=(1, 1))(baseModel)
baseModel = Flatten(name="flatten")(baseModel)
baseModel = Dense(256, activation="relu")(baseModel)
baseModel = Dropout(0.5)(baseModel)
baseModel = Dense(y_train.shape[1], activation="softmax")(baseModel)
vgg16_model = Model(inputs=vgg16.input, outputs=baseModel)
vgg16_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
#densenet_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
if os.path.exists("model/vgg_weights.hdf5") == False:
    model_check_point = ModelCheckpoint(filepath='model/vgg_weights.hdf5', verbose = 1, save_best_only = True)
    hist = vgg16_model.fit(X_train, y_train, batch_size = 64, epochs = 60, validation_data=(X_test, y_test), callbacks=[model_ch
    f = open('model/vgg_history.pckl', 'wb')
    pickle.dump(hist.history, f)
    f.close()
else:
    vgg16_model.load_weights("model/vgg_weights.hdf5")
#perform prediction on test images
predict = vgg16_model.predict(X_test)
predict = np.argmax(predict, axis=1)
```

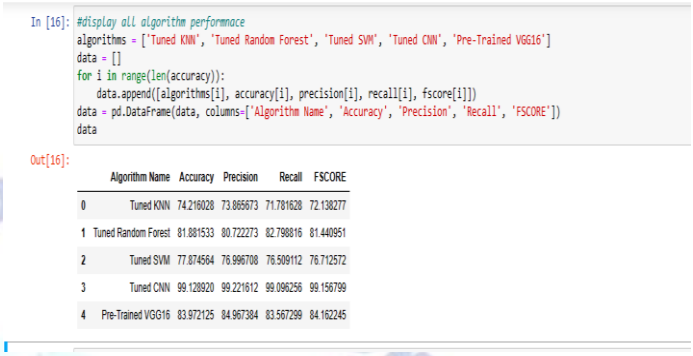
In above screen training pre-trained VGG16 model and after executing above model will get below output



In above screen VGG16 got 83% accuracy and this model is the second highest in accuracy



In above graph displaying performance of all algorithms where x-axis represents algorithm names and y-axis represents accuracy and other metrics and in all algorithms CNN got high accuracy

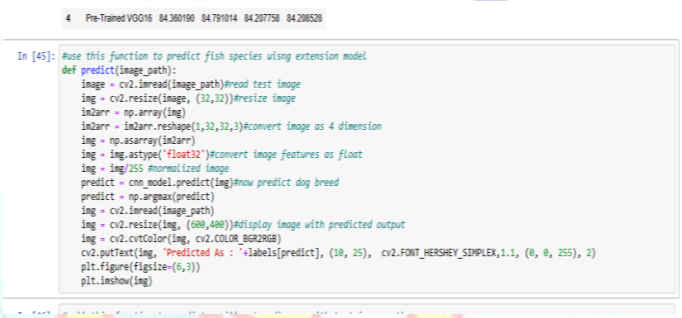


```
In [16]: #display all algorithm performance
algorithms = ['Tuned KNN', 'Tuned Random Forest', 'Tuned SVM', 'Tuned CNN', 'Pre-Trained VGG16']
data = []
for i in range(len(accuracy)):
    data.append([algorithms[i], accuracy[i], precision[i], recall[i], fscore[i]])
data = pd.DataFrame(data, columns=['Algorithm Name', 'Accuracy', 'Precision', 'Recall', 'FSCORE'])
data
```

Out[16]:

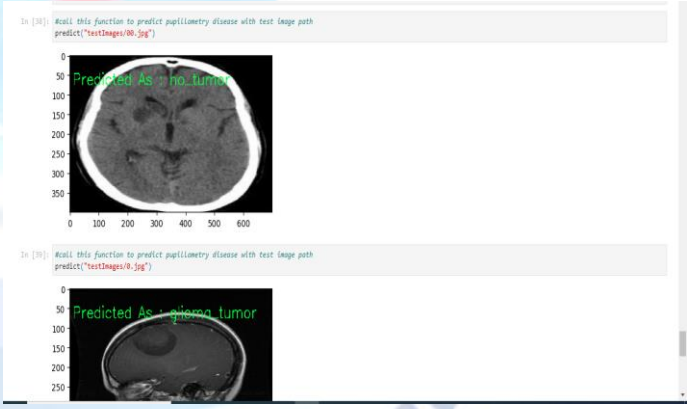
	Algorithm Name	Accuracy	Precision	Recall	FSCORE
0	Tuned KNN	74.216028	73.865673	71.781628	72.138277
1	Tuned Random Forest	81.881533	80.722273	82.798816	81.440951
2	Tuned SVM	77.874564	76.996708	76.509112	76.712572
3	Tuned CNN	99.128920	99.221612	99.096256	99.156799
4	Pre-Trained VGG16	83.972125	84.967384	83.567299	84.162245

In above screen displaying all algorithm performance in tabular format



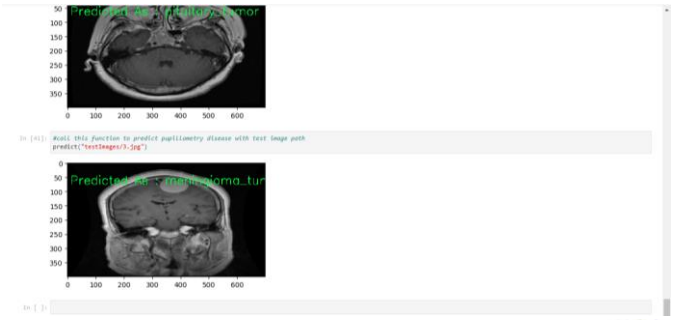
```
4  Pre-Trained VGG16  84.360190  84.791014  84.207758  84.208528

In [45]: #use this function to predict fish species uisng extension model
def predict(image_path):
    image = cv2.imread(image_path)#read test image
    img = cv2.resize(image, (32,32))#resize image
    im2arr = np.array(img)
    im2arr = im2arr.reshape(1,32,32,3)#convert image as 4 dimension
    img = np.asarray(im2arr)
    img = img.astype('float32')#convert image features as float
    img = img/255 #normalized image
    predict = cnn_model.predict(img)#now predict dog breed
    predict = np.argmax(predict)
    img = cv2.imread(image_path)
    img = cv2.resize(img, (600,400))#display image with predicted output
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    cv2.putText(img, 'Predicted As : '+labels[predict], (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,1.1, (0, 0, 255), 2)
    plt.figure(figsize=(6,3))
    plt.imshow(img)
```

In above screen defining predict function which will take input image path and then predict disease type



In above screen calling predict function with test image path and then in green colour text we can see predicted label

In above screen can see other image predicted disease type



Above screen showing prediction of another image. Similarly by giving test image path we can predict pupillometry disease

Conclusion:

In the end, using deep learning in radiology, especially for MRI image interpretation, has started a new era of progress and creativity in medical imaging. Through this review of the literature and system analysis, we looked at the current state of deep learning in radiology. We looked at the methods, applications, problems, and possible future paths. A review of the literature found a wide range of studies that used deep learning for different purposes, such as finding tumours, separating parts of images, classifying diseases, and reconstructing images. Many authors from different areas of radiology have added useful ideas that show how deep learning could improve the accuracy of diagnoses and clinical decision-making in the field.

MRI picture interpretation methods were fully understood through the system analysis, which also suggested new ways to do things. Even though current systems have had a lot of success, they also have problems, like not having enough data, making it hard to understand, and not being able to apply to all patient groups. To get around these problems, the suggested system uses new methods that are meant to make MRI interpretation more accurate, faster, and useful in clinical settings. Based on these comparisons, it's clear that deep learning has the potential to completely change the way radiology is done. However, there are still some problems that need to be solved. For example, we need big, diverse datasets, models that can be understood, models that are compliant with regulations, and models that can be easily integrated into clinical workflows.

In the future, researchers should work on solving these problems while also coming up with new and better ways to use deep learning to read MRI images. Researchers, clinicians, and business partners will need to work together to make progress and make sure that deep learning technologies can be used in everyday clinical practice. In conclusion, deep learning in radiology is a paradigm shift that could have a big effect on patient care, make diagnoses more accurate, and make clinical processes more efficient. We can use deep learning to its fullest extent to change the field of MRI picture interpretation and make radiology better by being open to new ideas, working together, and doing research across disciplines.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. Medical image analysis, 42, 60-88.

[2] Shen, D., Wu, G., & Suk, H. I. (2017). Deep learning in medical image analysis. Annual review of biomedical engineering, 19, 221-248.

[3] Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., ... & Dean, J. (2019). A guide to deep learning in healthcare. Nature medicine, 25(1), 24-29.

[4] Yasaka, K., Akai, H., & Kunimatsu, A. (2018). Kiryu. Deep learning for staging liver fibrosis on CT: a pilot study. European radiology, 28(11), 4578-4585.

[5] Greenspan, H., van Ginneken, B., & Summers, R. M. (2016). Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. IEEE transactions on medical imaging, 35(5), 1153-1159.

[6] Chartrand, G., Cheng, P. M., Vorontsov, E., Drozdzal, M., Turcotte, S., Pal, C. J., ... & Tang, A. (2017). Deep learning: a primer for radiologists. Radiographics, 37(7), 2113-2131.

[7] Litjens, G., Ciompi, F., Wolterink, J. M., de Vos, B. D., Leiner, T., & Teuwen, J. (2016). State-of-the-art deep learning in cardiovascular image analysis. Journal of cardiovascular computed tomography, 10(3), 204-213.

[8] Doi, K. (2007). Computer-aided diagnosis in medical imaging: historical review, current status and future potential. Computerized Medical Imaging and Graphics, 31(4-5), 198-211.

[9] Hosny, A., Parmar, C., Quackenbush, J., Schwartz, L. H., Aerts, H. J. W. L., & Artificial Intelligence in Radiology: Current Status and Future Directions. (2018). Radiology, 204563.

[10] Lee, J. G., Jun, S., Cho, Y. W., Lee, H., Kim, G. B., Seo, J. B., & Kim, N. (2017). Deep learning in medical imaging: general overview. Korean journal of radiology, 18(4), 570-584.