Brain Tumour Classification using Deep learning methods

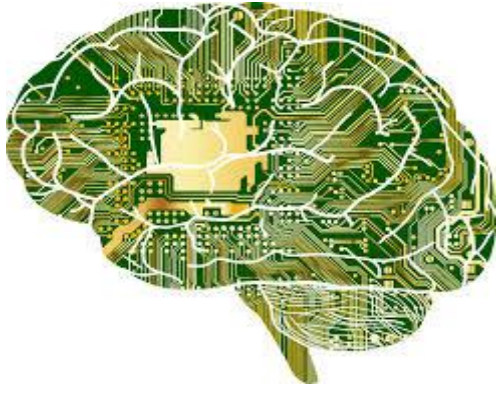
ED6001 Term Project

Shital Yelne (EE20S052) | Piyush Nehete(CL21M011)

Overall Problem Statement

We are having brain data containing different types of tumours and normal tumour data. There are 3 folders of different types of tumours i.e glioma_tumor, meningioma_tumor and pituitary_tumor. Other folder has normal brain data(no_tumour). Training and Testing samples has been provided. No ground truth images are given.

So we pose this as a Multiclass Classification Problem which we tried of solving using different deep learning approaches.



Dataset

Dataset is having four different classes i.e. 'glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor'

Training :

glioma_tumor= 826 samples

meningioma_tumor= 822 samples

no_tumor= 395 samples

pituitary_tumor= 827 samples

Test:

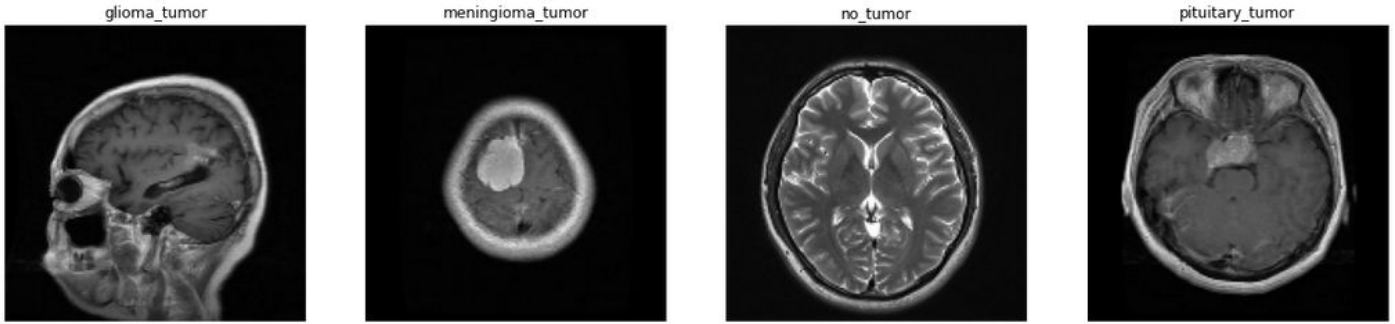
glioma_tumor= 100 samples

meningioma_tumor= 115 samples

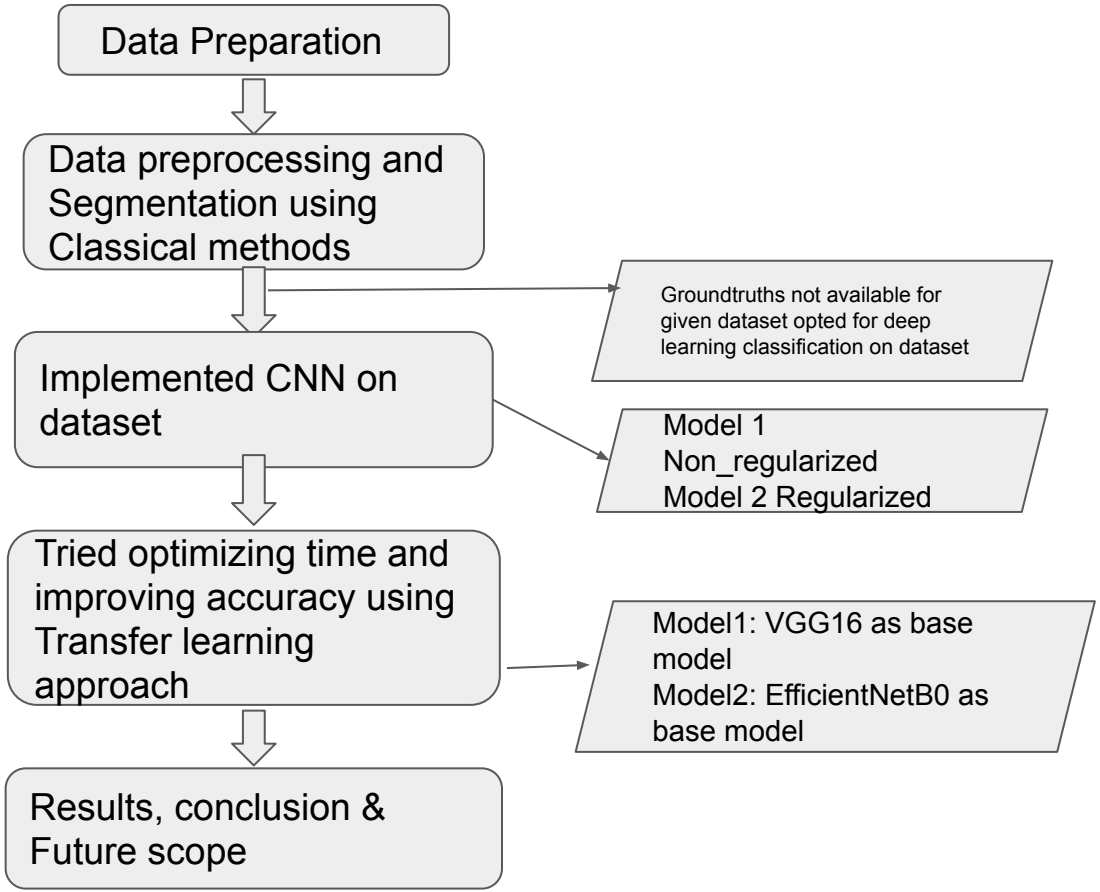
no_tumor= 105 samples

pituitary_tumor= 74 samples

Images one of each type:



WorkFlow

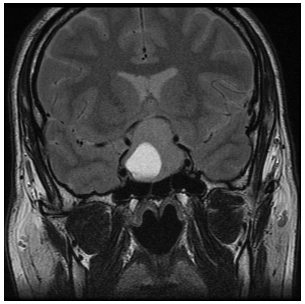

Data Preparation
Data preprocessing and Segmentation using Classical methods
Groundtruths not available for given dataset opted for deep learning classification on dataset
Implemented CNN on dataset
Model 1
Non_regularized
Model 2 Regularized
Tried optimizing time and improving accuracy using Transfer learning approach
Model1: VGG16 as base model
Model2: EfficientNetB0 as base model
Results, conclusion & Future scope

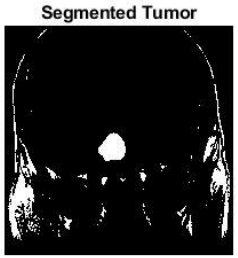
Classical Method Segmentation

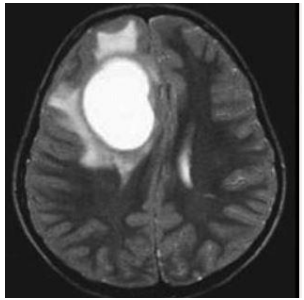
Process of segmentation

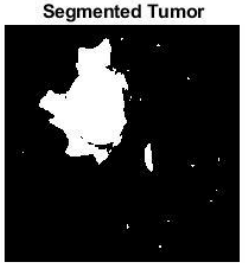
- Image Preprocessing
- Otsu Binarization for segmentation
- K-means clustering
- Applying colorform
- Created clusters
- Created blank cell array to store score
- Created Label

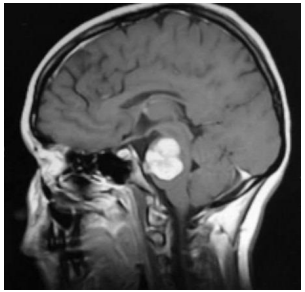
Results of Segmentation

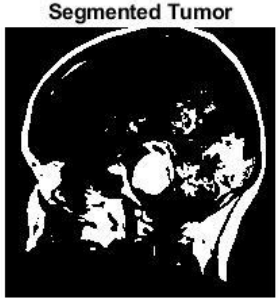


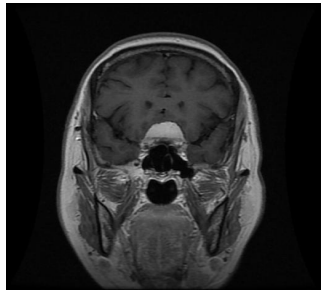


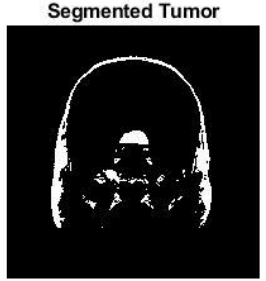




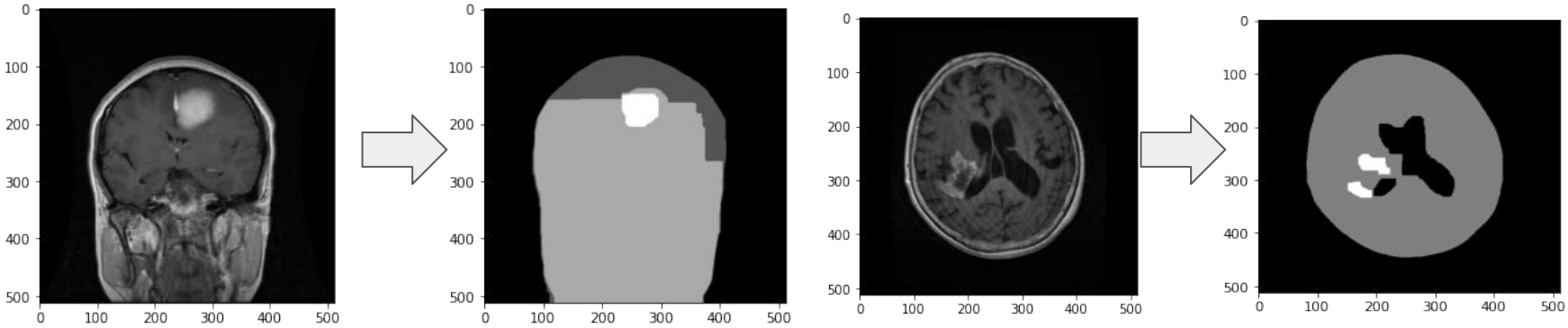








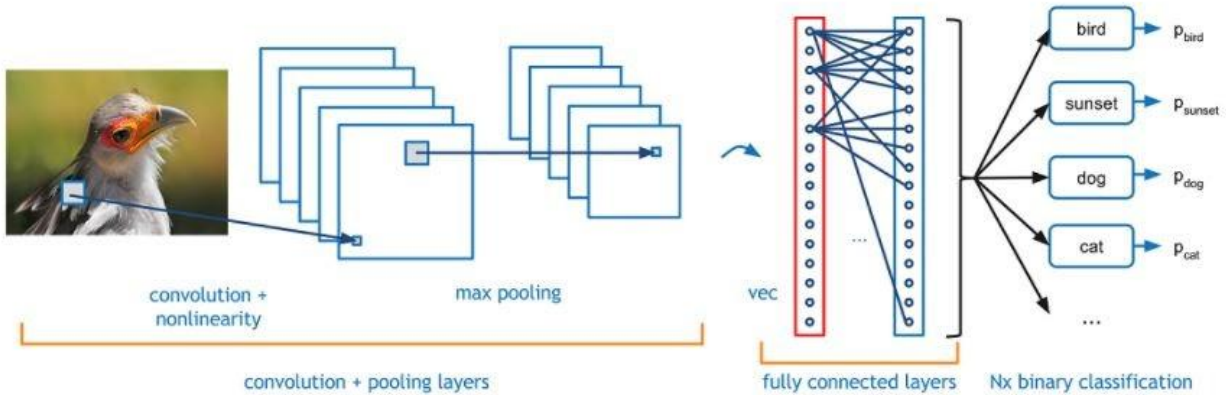
Results of Segmentation(Graph Cut)



- Ground truth is not available for this dataset for validation of segmentation
- Decided to opt deep learning methods to classify tumor

CNN

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of artificial neural network, most commonly applied to analyze visual imagery.



source: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

CNN

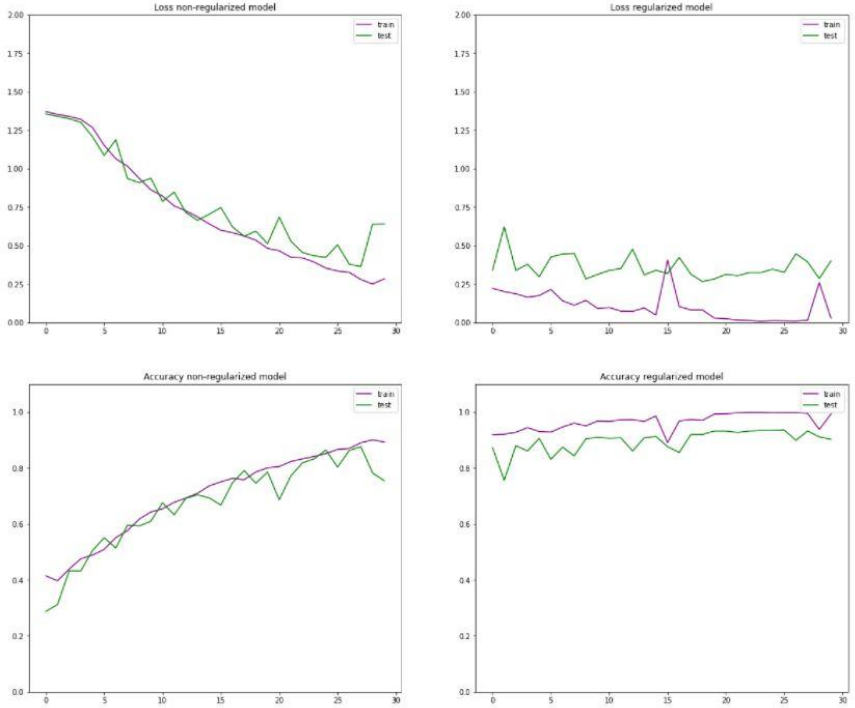
	Model 1	**Model 2**
Batch size	32	32
epochs	30	30
Normalization	Not used	After each convolution layer
Number of Conv. Layers	6	6
Last activation Function	sigmoid	softmax
Optimizer used	SGD	SGD

CNN Model

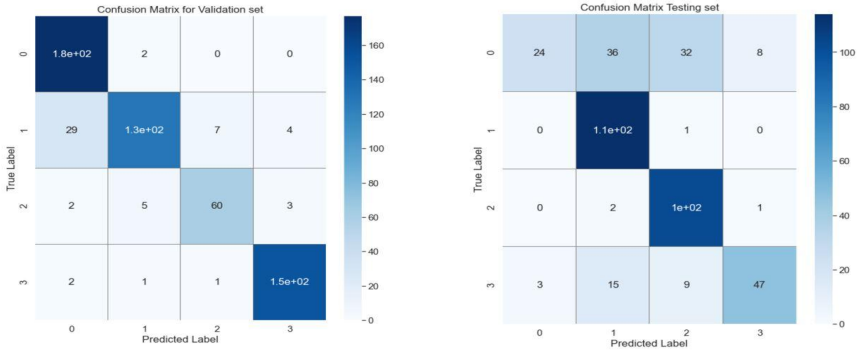
```
model1 = Sequential()
model1.add(Conv2D(filters=64, kernel_size=3, padding= 'Same', activation='relu', input_shape=(X_train.shape[1],X_train.shape[2],
model1.add(MaxPool2D(pool_size=(2,2)))
model1.add(BatchNormalization())

model1.add(Conv2D(filters=128, kernel_size=3, padding= 'Same', activation='relu'))
model1.add(MaxPool2D(pool_size=(2,2)))
model1.add(BatchNormalization())

model1.add(Conv2D(filters=128, kernel_size=3, padding= 'Same', activation='relu'))
model1.add(MaxPool2D(pool_size=(2,2)))
model1.add(BatchNormalization())

model1.add(Conv2D(filters=256, kernel_size=3, padding= 'Same', activation='relu'))
model1.add(MaxPool2D(pool_size=(2,2)))
model1.add(BatchNormalization())

model1.add(Conv2D(filters=256, kernel_size=3, padding= 'Same', activation='relu'))
model1.add(MaxPool2D(pool_size=(2,2)))
model1.add(BatchNormalization())

model1.add(Conv2D(filters=512, kernel_size=3, padding= 'Same', activation='relu'))
model1.add(MaxPool2D(pool_size=(2,2)))
model1.add(BatchNormalization())

model1.add(Flatten())
model1.add(Dense(1024, activation = "relu"))
model1.add(Dropout(0.2))

model1.add(Dense(512, activation = "relu"))
model1.add(Dropout(0.2))

model1.add(Dense(4, activation = "softmax"))

model1.compile(optimizer = 'SGD' , loss = "categorical_crossentropy", metrics=["accuracy"])
```

CNN Results

Accuracy and Loss analysis after each epochs



Confusion Matrix Model 2



	Model 1	**Model 2**
Validation Accuracy @epoch30	0.78	0.9
Validation loss @epoch30	0.63	0.39

CNN Results

Conclusion for CNN:

- Model 2 is giving high accuracy
- Support values are also good for model 2

So, model 2 (Regularize model) is better compare to model 1(non Regularize)

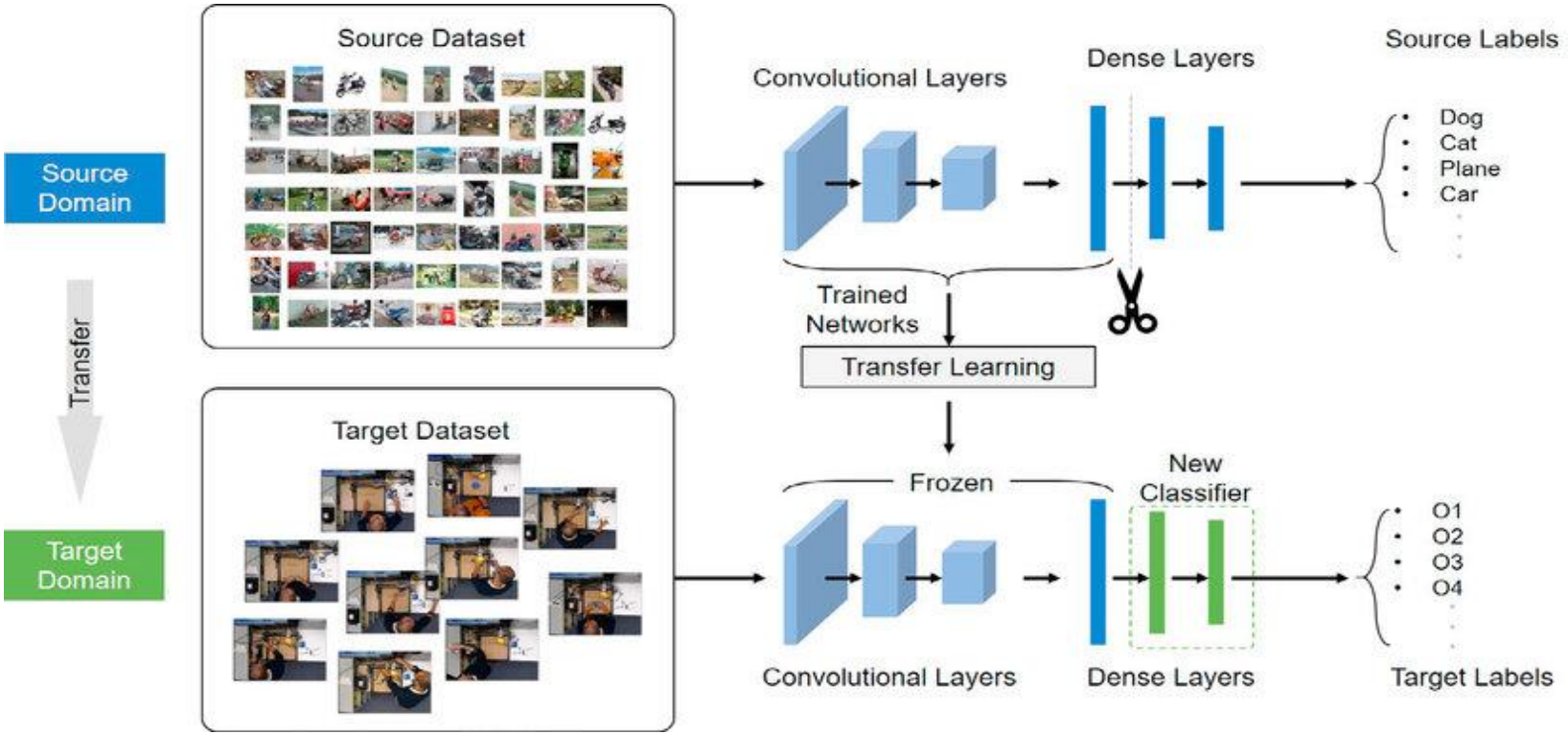
Drawbacks of these models:

- Computationally very heavy
- Classification accuracy for some class is very low

To Overcome these flaws in these models & to increase accuracy further decided to go for transfer learning

Transfer Learning

Transfer learning generally refers to a process where a model trained on one problem is used in some way on a second related problem.



VGG16

Models implemented for Transfer learning approach:

Model 1:

```
vgg_model = VGG16(weights='imagenet',include_top=False)
x=vgg_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(128,activation='relu')(x)
x=Dropout(0.30)(x)
output=Dense(4,activation='softmax')(x)
model3=Model(inputs=vgg_model.input,outputs=output)#.to(device)
model3.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

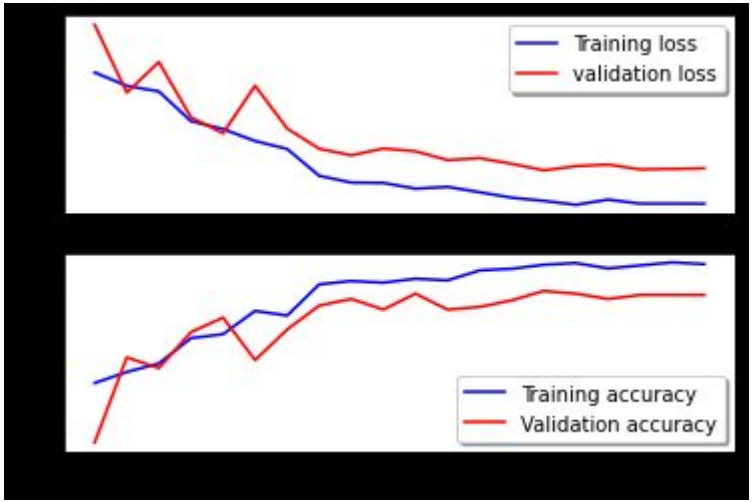
Model 2:

```
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(image_size, image_size, 3))
model = base_model.output
model = GlobalAveragePooling2D()(model)
model = Dense(4, activation='softmax')(model)
model = Model(inputs = base_model.input, outputs=model)
model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
```

VGG16

Best results were obtained for model 2:

No. of epochs	20
Training Accuracy	92.38%
Training Loss	0.2009
Validation Accuracy	88.33%
Validation Loss	0.2996
Accuracy on Testing dataset	58%



	precision	recall	f1-score	support
0	0.94	0.17	0.29	100
1	0.69	0.71	0.70	115
2	0.45	0.90	0.60	105
3	0.76	0.46	0.57	74
accuracy			0.58	394
macro avg	0.71	0.56	0.54	394
weighted avg	0.70	0.58	0.54	394

EfficientNetB0

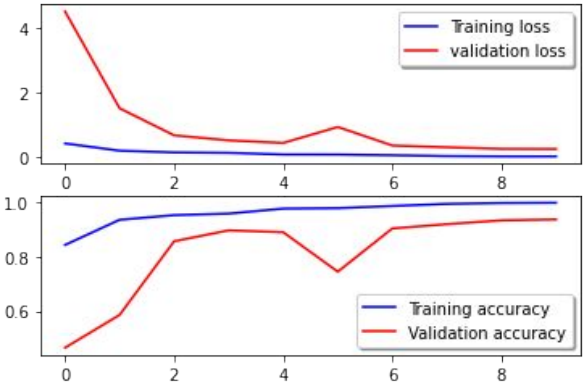
With considerably fewer numbers of parameters, the family of models are efficient and also provide better results.EfficientNet is being used as standard pretrained model for image classification. We are using EfficientNetB0.

Model used for transfer learning approach:

```
base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(image_size, image_size, 3))
model = base_model.output
model = GlobalAveragePooling2D()(model)
model = Dense(4, activation='softmax')(model)
model = Model(inputs = base_model.input, outputs=model)
model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
```

EfficientNetB0 Results

No. of epochs	10
Training Accuracy	99.65%
Training Loss	0.0123
Validation Accuracy	93.55%
Validation Loss	0.2447
Accuracy on Testing dataset	80%



	precision	recall	f1-score	support
0	1.00	0.38	0.55	100
1	0.72	0.97	0.83	115
2	0.75	0.98	0.85	105
3	0.98	0.82	0.90	74
accuracy			0.80	394
macro avg	0.86	0.79	0.78	394
weighted avg	0.85	0.80	0.78	394

Conclusion

- Training and validation accuracy is far better in case of Transfer Learning approach 99.65% and 93.55% respectively. One reason can be for less data, using pretrained weights through transfer learning helps network learn more.
- Time required for Transfer Learning models is considerably low.
- Our transfer learning model is working better on test dataset also with accuracy around 80% which was not the case in CNN.
- In both transfer learning approaches EfficientNetB0 worked better for us as we were getting less testing accuracy in case of VGG16.

Future Scope

- One thing which can be implemented further is getting survival rate of patient by size or other features of tumour.
- Early detection of brain tumor is possible by implementing accurate model at the time of MR Imaging.
- Also improving accuracy and optimizing network for time reduction can be done.

THANK YOU