ProjectF2

Team 27

Brain Tumor Detection using Convolutional Neural Networks

Britanya Wright

Arun Gaonkar

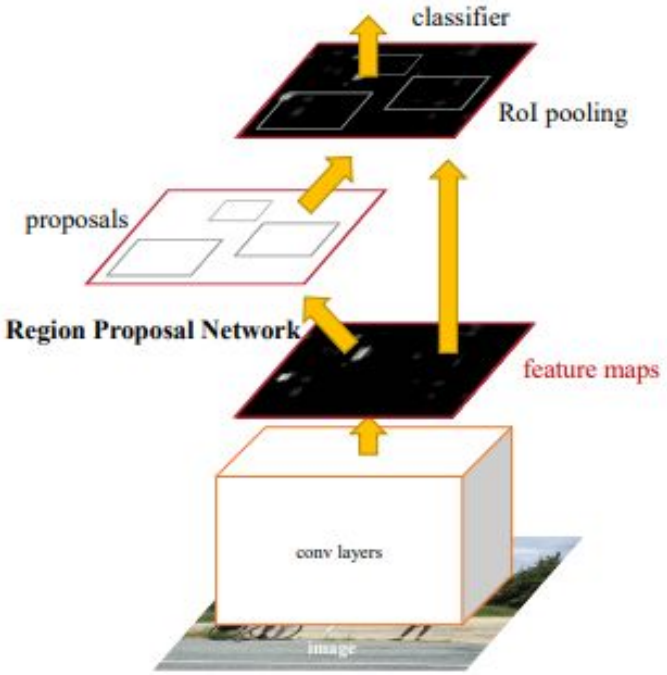
Sai Harsha Nadendla

Motivation/Problem

- Affects over 700,000 people within the United States
- Caution and care has to be taken in the accuracy of the diagnosis, treatment, and planning to increase the life expectancy of that patient.
- Radiologists and clinical experts are tasked with detection and identification.
- To reduce the introduction of human error we use Machine Learning for automatic classification.

Task

- Implement CNN model with the ability to detect brain tumors in Images.
- Implement Faster R-CNN for Image segmentation to identify the location of the tumor.



Faster R-CNN *

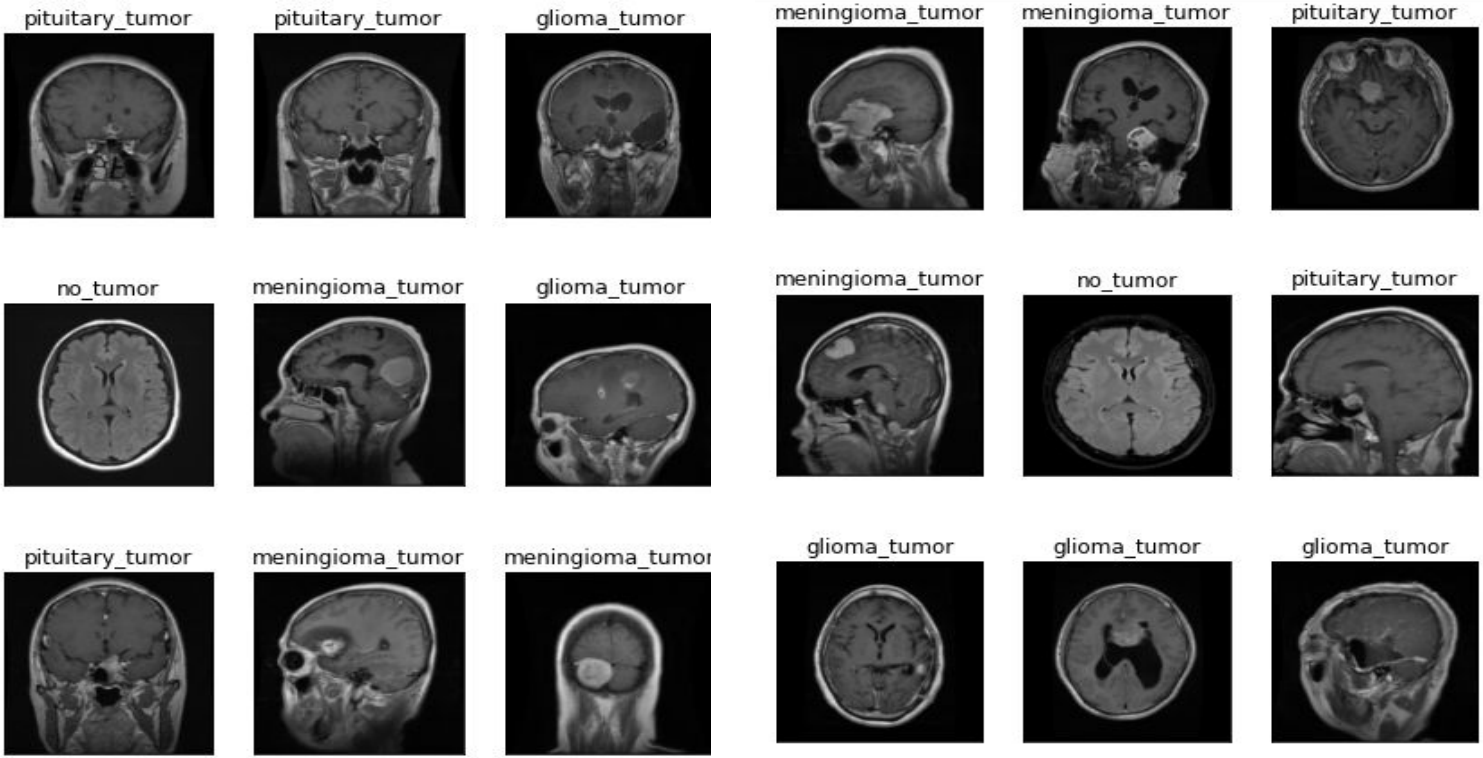
* source - S Ren, K He, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”
https://arxiv.org/pdf/1506.01497.pdf

Dataset

- 3,565 raw MRI images found in Kaggle*.
- Each Image is of different sizes.
- Varying tumor location with size and different types.
- 3 different tumor types : Glioma, Meningioma, and pituitary; + 1 for No tumor.
- Resized image shape 3 x 150 x 150

*** source :** https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri

Sample Images

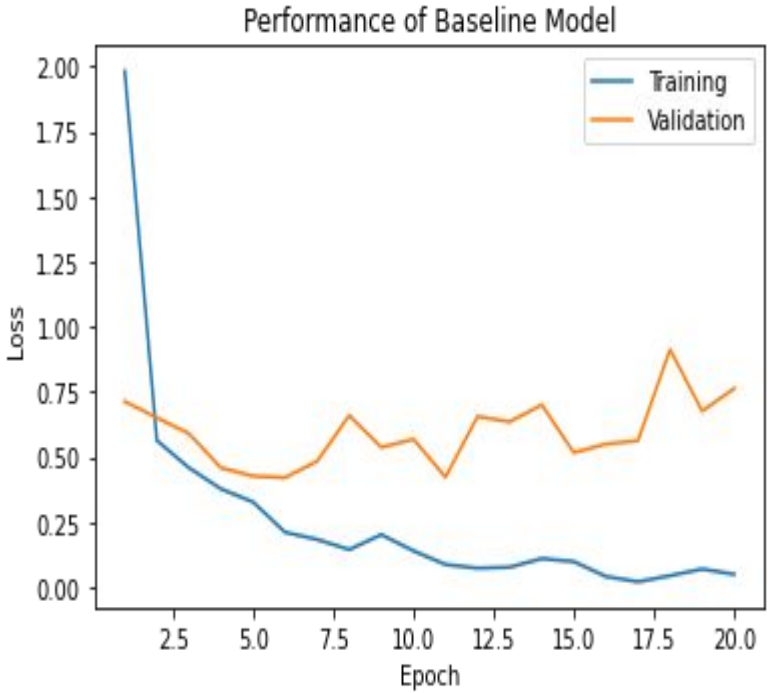


Baseline Model : CNN Parameters

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 12, 150, 150]             336
       BatchNorm2d-2         [-1, 12, 150, 150]              24
         MaxPool2d-3           [-1, 12, 75, 75]               0
            Conv2d-4           [-1, 20, 75, 75]           2,180
       BatchNorm2d-5           [-1, 20, 75, 75]              40
         MaxPool2d-6           [-1, 20, 37, 37]               0
            Conv2d-7           [-1, 32, 37, 37]           5,792
       BatchNorm2d-8           [-1, 32, 37, 37]              64
         MaxPool2d-9           [-1, 32, 18, 18]               0
           Linear-10                 [-1, 5184]      53,752,896
           Linear-11                 [-1, 2592]      13,439,520
           Linear-12                    [-1, 4]          10,372
================================================================
Total params: 67,211,224
Trainable params: 67,211,224
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.26
Forward/backward pass size (MB): 7.37
Params size (MB): 256.39
Estimated Total Size (MB): 264.02
----------------------------------------------------------------
```

```python
class Net(nn.Module):
  def __init__(self, num_classes=4):
    super(Net, self).__init__()
    self.conv1 = nn.Conv2d(in_channels=3, out_channels=12, kernel_size=3, stride=1, padding=1)
    self.bn_1 = nn.BatchNorm2d(num_features=12)
    self.pool = nn.MaxPool2d(kernel_size=2)
    self.conv2 = nn.Conv2d(in_channels=12,out_channels=20,kernel_size=3,stride=1,padding=1)
    self.conv3 = nn.Conv2d(in_channels=20,out_channels=32,kernel_size=3,stride=1,padding=1)
    self.bn_2 = nn.BatchNorm2d(num_features=20)
    self.bn_3 = nn.BatchNorm2d(num_features=32)
    self.fc1 = nn.Linear(in_features=32*18*18,out_features=16*18*18)
    self.fc2 = nn.Linear(in_features=16*18*18,out_features=8*18*18)
    self.fc3 = nn.Linear(in_features=8*18*18,out_features=num_classes)

def forward(self, x):
    x = self.pool(F.relu(self.bn_1(self.conv1(x))))
    x = self.pool(F.relu(self.bn_2(self.conv2(x))))
    x = self.pool(F.relu(self.bn_3(self.conv3(x))))
    #print(x.shape)
    x = x.view(-1, 32*18*18)
    x = F.relu(self.fc1(x))
    x = F.relu(self.fc2(x))
    x = self.fc3(x)
    return x
```

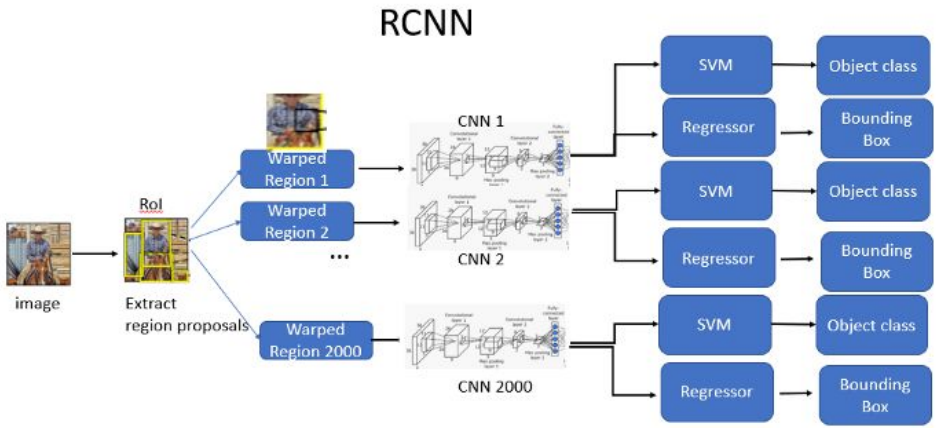
CNN - Baseline Performance




```
Test Loss: 2.753260

Test Accuracy of glioma_tumor: 28% (28/100)
Test Accuracy of meningioma_tumor: 83% (96/115)
Test Accuracy of   no_tumor: 96% (101/105)
Test Accuracy of pituitary_tumor: 62% (46/74)

Test Accuracy (Overall): 68% (271/394)
```

CNN to Faster R-CNN (1)

- Convolutional Neural Network - Image classification and object detection.
  - Multiple filters and feature decode layers for Image Classification.
- Regional -CNN (R-CNN) is used for object detection with bounding boxes.
  - Selective Search -> Generate Region Proposals -> CNN -> SVM -> Regressor.

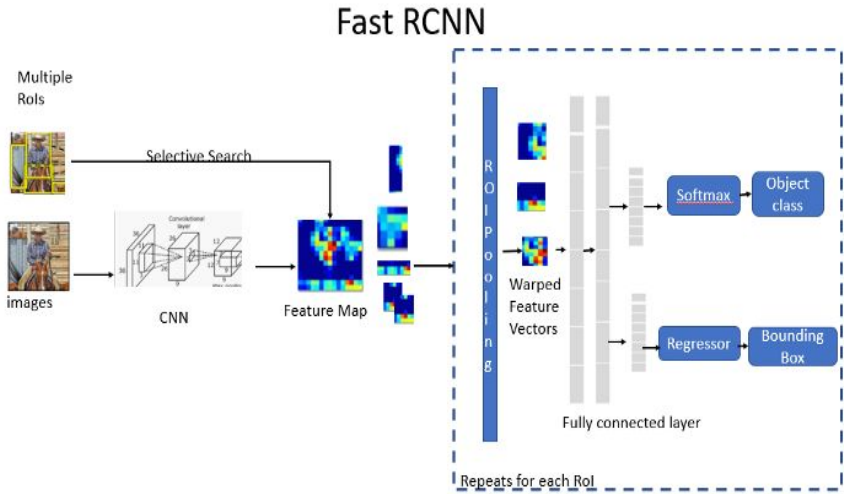


CNN to Faster R-CNN (2)

Disadvantages of R-CNN

- R-CNN slow because of Selective search.
- Region extraction for every region in the image is slow.

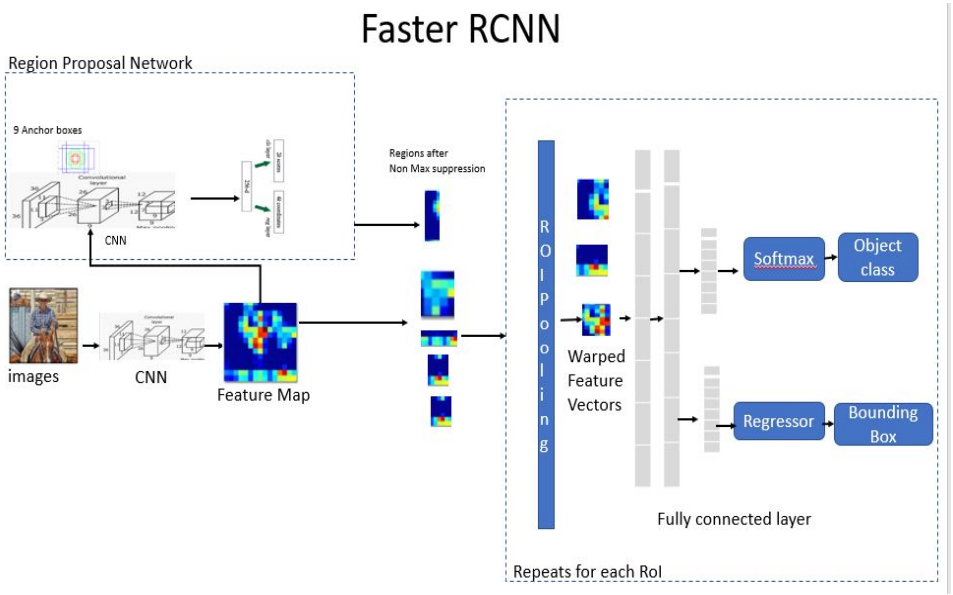
- Fast R-CNN uses one ConvNet to extract features for entire image.
- ‘RoI Pooling layer’ to extract interested regions from feature maps using selective search.
- FC with 2 output layers
  - Softmax for Object Classification.
  - Regressor for defining bounding boxes.



CNN to Faster R-CNN (3)

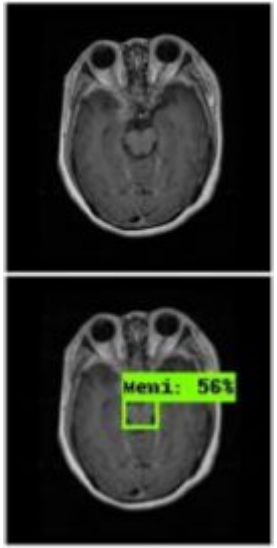
- Use Region Proposal Network instead of selective search, to generate RoI's.

  - Image -> Deep CNN -> Feature Maps
  - Feature Maps -> RPN -> RoI Bounding Boxes
  - Feature Maps & RoI -> Fast R-CNN Detection -> Classification and Bounding Boxes.



Proposed Model : Faster R-CNN

```
#Prediction Function
def predict(classes, data, target, model):
  output = model(data) # get the predictions on the image

  pred_score, pred = torch.max(output, 1)
  amount = torch.max(pred_score)
  pred_class = [classes[i] for i in list(pred.cpu().numpy())]


  print(amount)
  #print(pred_class)
  return pred_class, pred_score

#Different color for each class
colors = np.random.uniform(0, 255, size=(len(classes), 3))

# Setting model to evaluate
#device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.eval()

#Test data
batch_size = 197
test = DataLoader(test_set, batch_size = batch_size)

# Iterating over batches of test data
for data, target in test:
  if flag_cuda:
    data, target = data.cuda(), target.cuda()
  pred_class, pred_score = predict(classes, data, target, model)
```



Our Goal

Thank you...