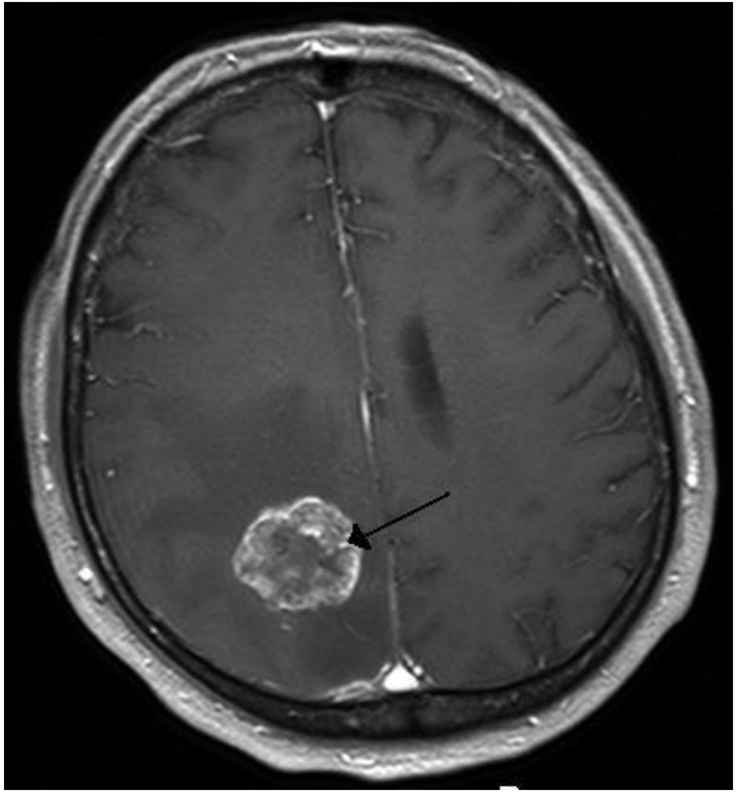
Brain Tumor detection and localization

What is brain tumor?

A brain tumor occurs when abnormal cells form within the brain. There are two main types of tumors: cancerous (malignant) tumors and benign tumors. Cancerous tumors can be divided into primary tumors, which start within the brain, and secondary tumors, which have spread from elsewhere, known as brain metastasis tumors. headaches, seizures, problems with vision, vomiting and mental changes.



Dataset description

The image data that was used for this problem is Brain MRI Images for Brain Tumor Detection. It consists of MRI scans of two classes:

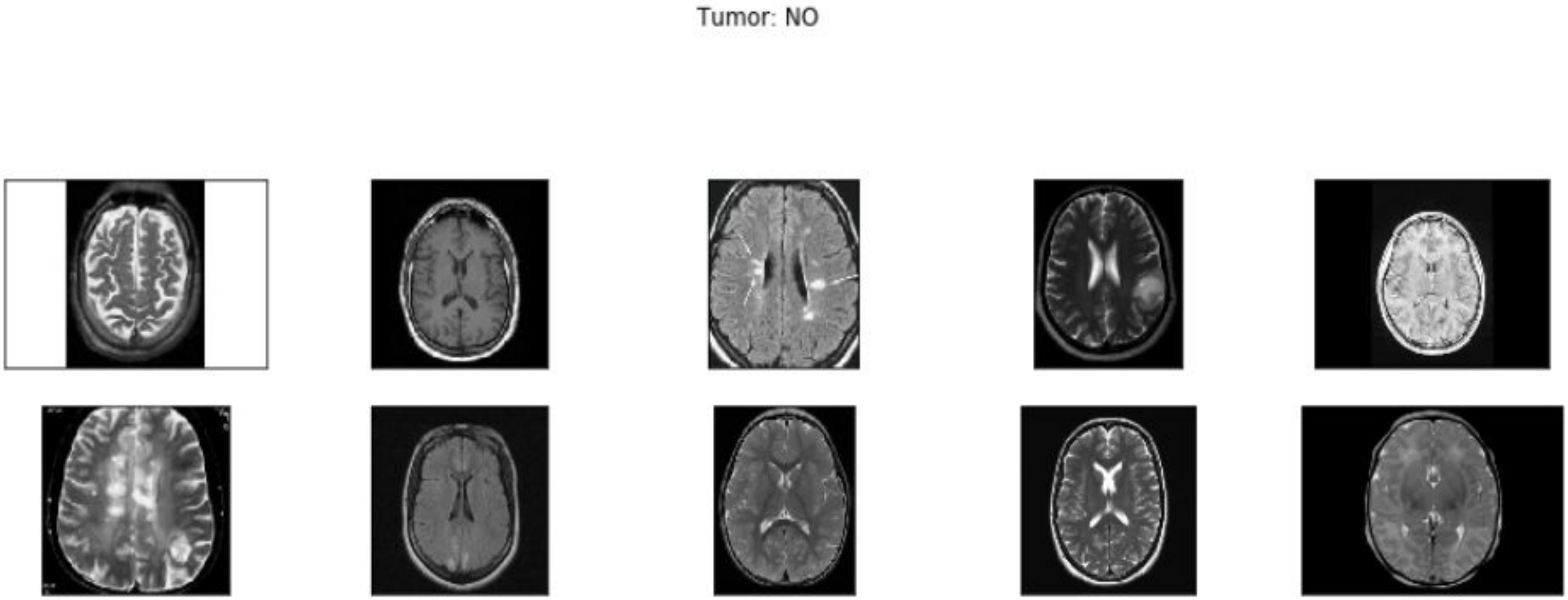
- **NO** - no tumor, encoded as **0**
- **YES** - tumor, encoded as **1**

Overall there are **98** images of non-tumor and **155** images with tumor

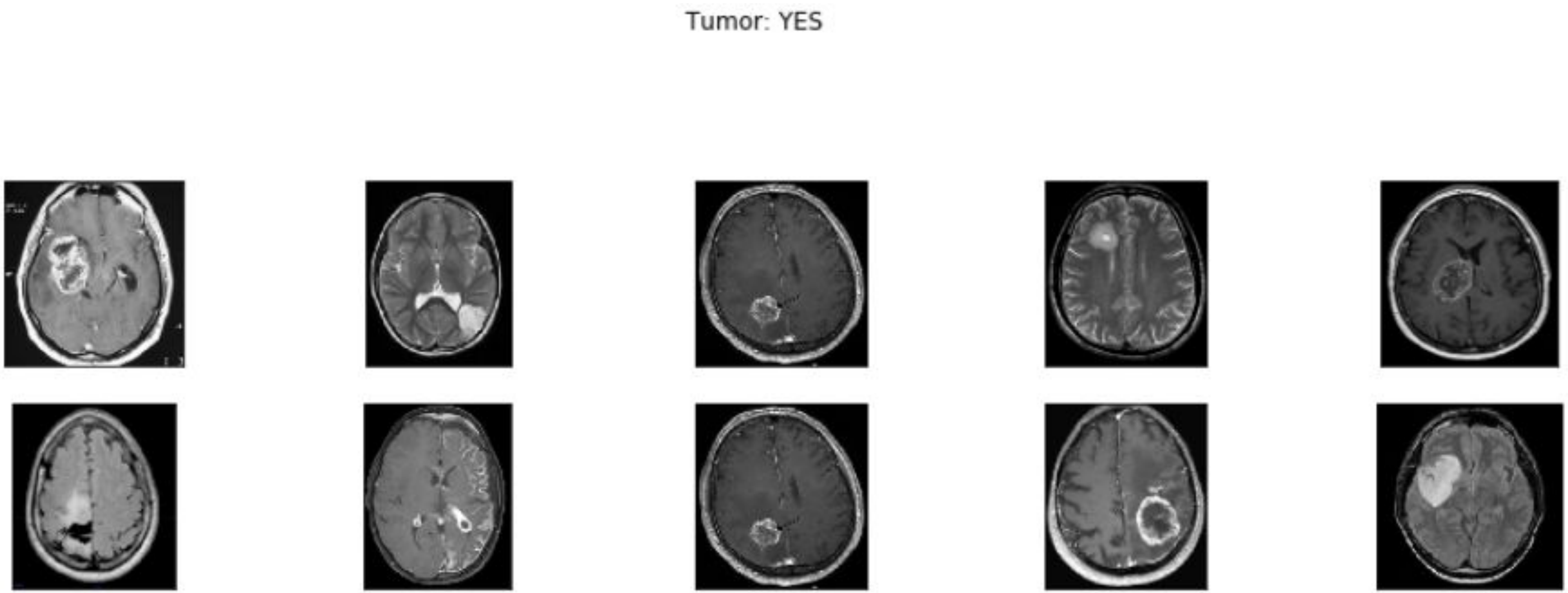
Data import and preprocessing

- Train - 193 images
- Validation - 50 images
- Test - 10 images

Samples without tumor



Samples with tumor

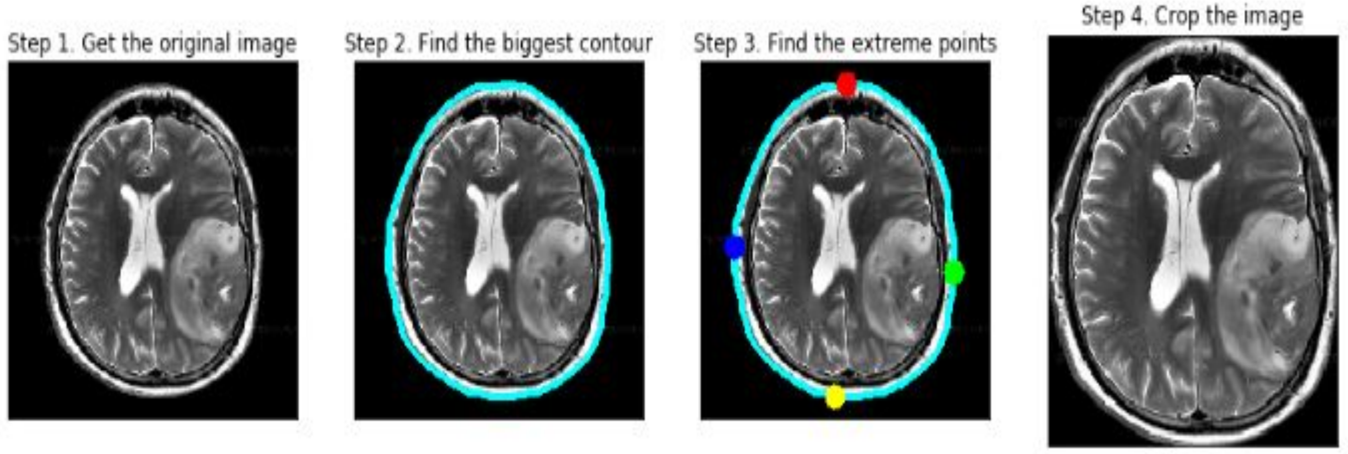


Normalization

As you can see, images have different width and height and different size of "black corners". Since the image size for e.g. VGG-16 input layer is (224, 224) some wide images may look weird after resizing.

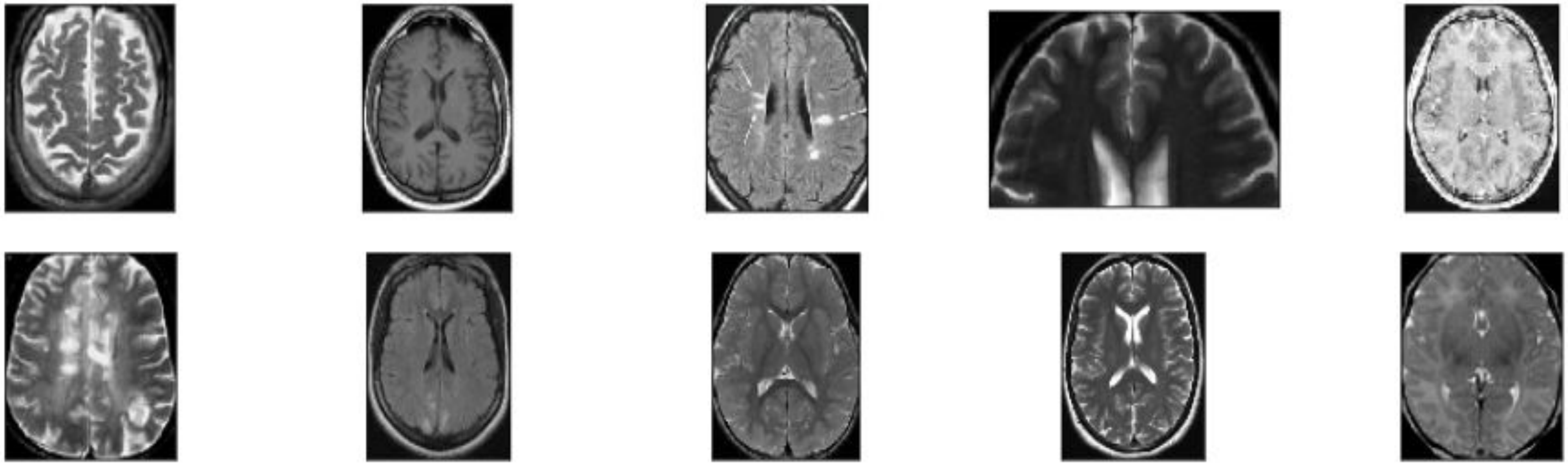
The first step of "normalization" would be to crop the brain out of the images. I used technique which was perfectly described in pyimagesearch blog https://www.pyimagesearch.com/2016/04/11/finding-extreme-points-in-contours-with-opencv/

Normalization - crop algorithm



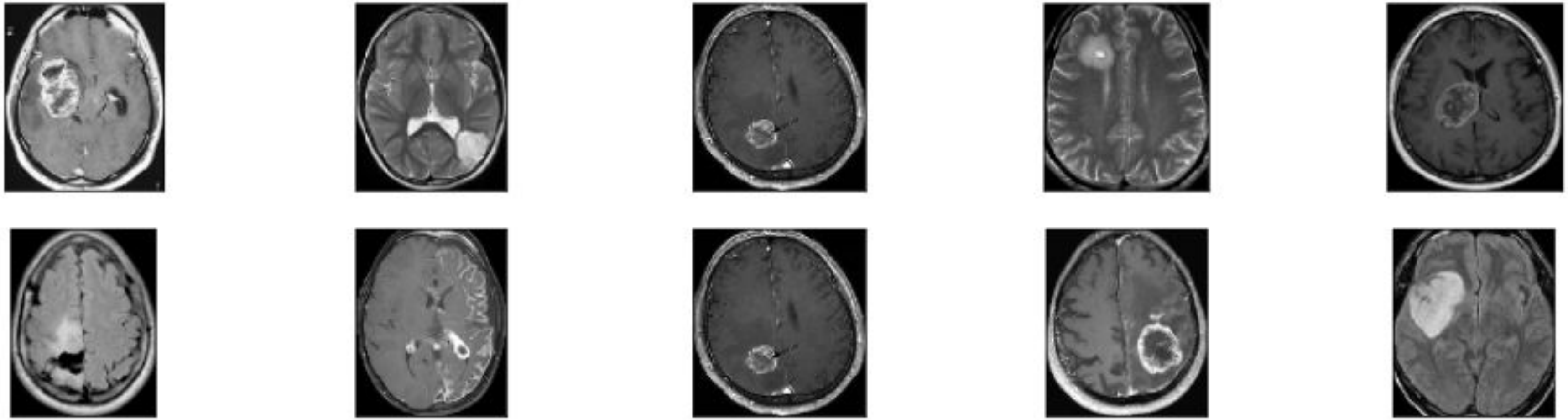
Samples without tumor - cropped

Tumor: NO

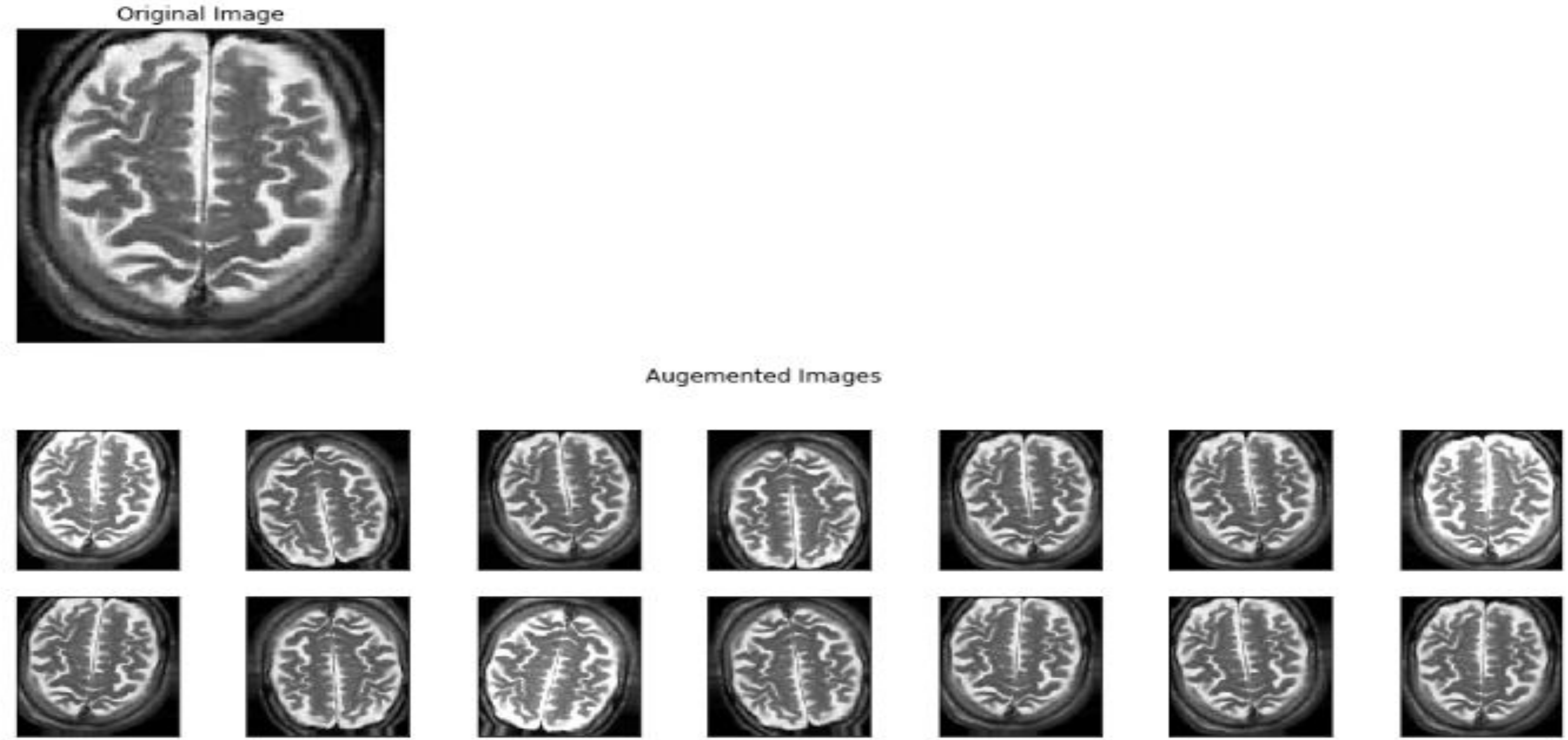


Samples with tumor - cropped

Tumor: YES



Augmentation

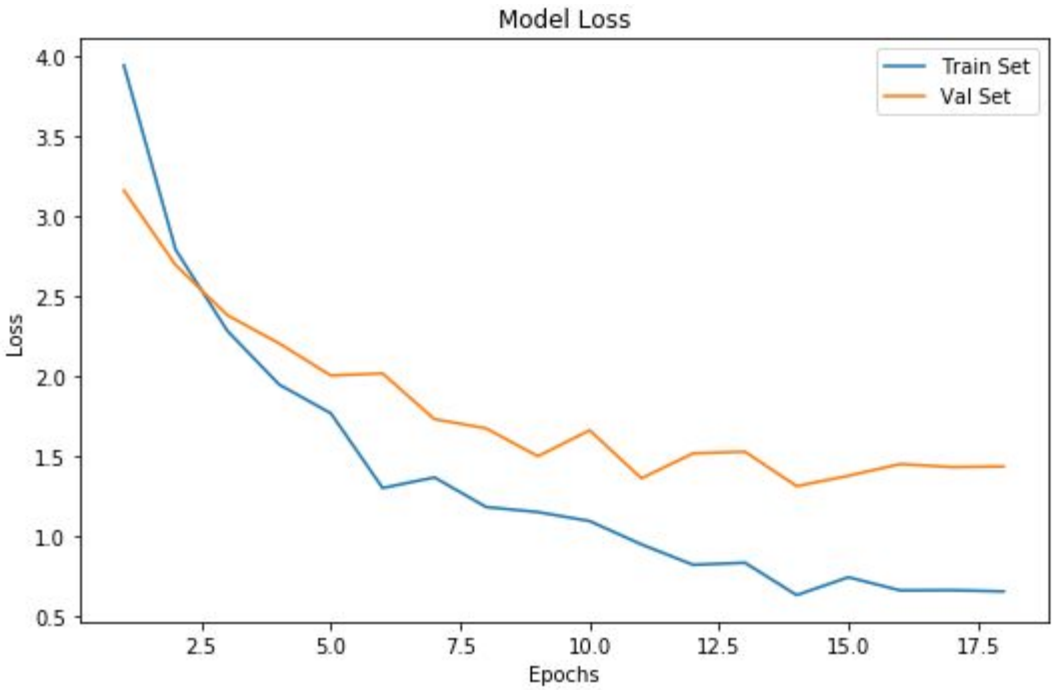


VGG16 model transfer learning

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 25089
=================================================================
Total params: 14,739,777
Trainable params: 25,089
Non-trainable params: 14,714,688
_________________________________________________________________
```

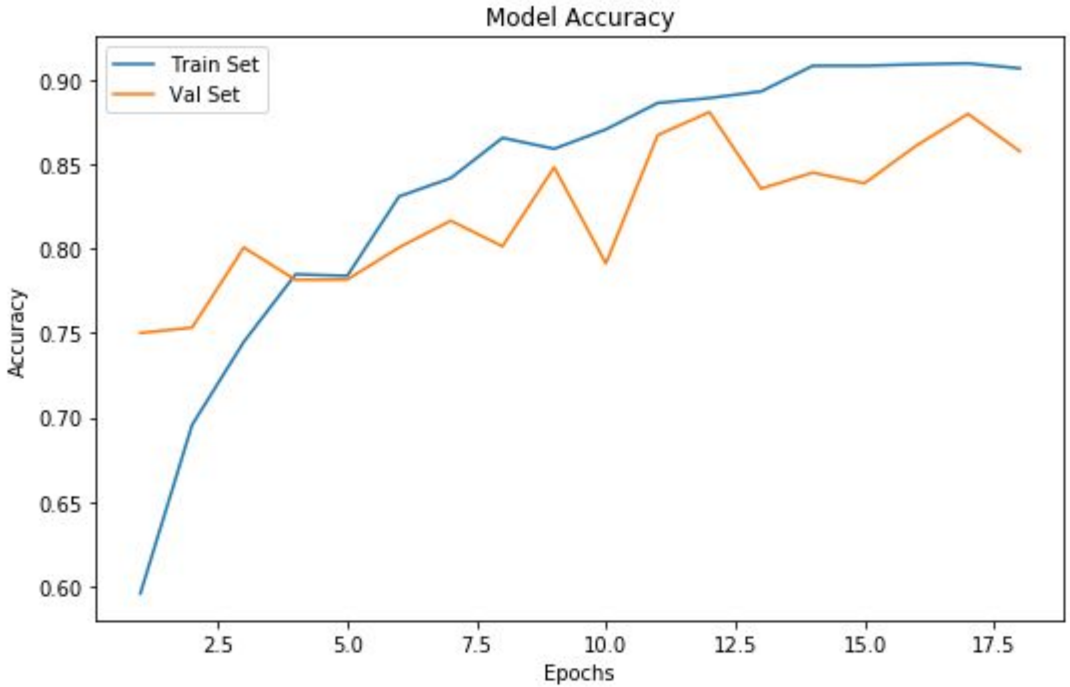
VGG16 model transfer learning

- Loss - binary cross-entropy
- Optimizer - RMSProp
- Learning rate - 1e-4
- Metrics - accuracy
- Epochs - 16



VGG16 model transfer learning

- Validation accuracy - 0.86 - best, test accuracy - 0.9
- Validation loss - 1.2926, test loss - 0.5571



VGG19 model transfer learning

- Loss - binary cross-entropy
- Optimizer - Adam
- Learning rate - 1e-4
- Metrics - accuracy
- Epochs - 30

- Validation accuracy - 0.8576
- Test accuracy - 0.9436
- Validation loss - 1.1591
- Test loss - 0.3835

ResNet-50 model transfer learning

- Loss - binary cross-entropy
- Optimizer - Adam
- Learning rate - 1e-4
- Metrics - accuracy
- Epochs - 11

- Validation accuracy - 0.5981
- Test accuracy - 0.9390
- Validation loss - 2.4702
- Test loss - 0.1750

ResNet-50 model transfer learning

- Loss - binary cross-entropy
- Optimizer - RMSProp
- Learning rate - 1e-4
- Metrics - accuracy
- Epochs - 7

- Validation accuracy - 0.7595
- Test accuracy - 0.9542
- Validation loss - 1.1659
- Test loss - 0.1377

ResNet-101V2 model transfer learning

- Loss - binary cross-entropy
- Optimizer - RMSProp
- Learning rate - 1e-4
- Metrics - accuracy
- Epochs - 12

- Validation accuracy - 0.8013
- Test accuracy - 0.9402
- Validation loss - 0.8676 - best
- Test loss - 0.1797

MobileNet-V2 model transfer learning

- Loss - binary cross-entropy
- Optimizer - RMSProp
- Learning rate - 1e-3
- Metrics - accuracy
- Epochs - 17

- Validation accuracy - 0.7437
- Test accuracy - 0.9118
- Validation loss - 2.6682
- Test loss - 0.9332

Localization

Now we want to build a detector which will point out on the location of the tumor on the scan.

But wait, we need annotations for image localization.

Used the following github repo (data-cleaned) folder, where on each folder (train, test, val) there is also corresponding annotations file.

Also in the new data deleted some duplicated scans.

And used matterplot Mask-RCNN method for localization.

Loss metrics of Mask-RCNN

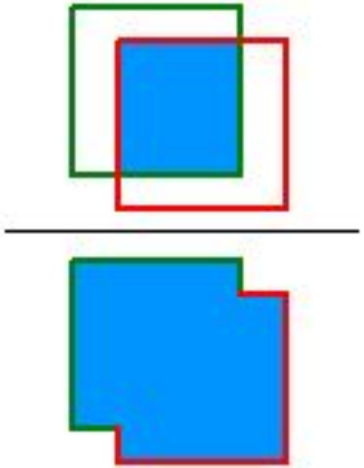
- **rpn_class_loss** : How well the Region Proposal Network separates background with objetcs
- **rpn_bbox_loss**: How well the RPN localize objects
- **mrcnn_bbox_loss**: How well the Mask RCNN localize objects
- **mrcnn_class_loss**: How well the Mask RCNN recognize each class of object
- **mrcnn_mask_loss** : How well the Mask RCNN segment objects
- **loss**: A combination (surely an addition) of all the smaller losses.

All of those losses are calculated on the training dataset. The losses for the validation dataset are those starting with '**val**'

IOU - Intersection Over Union

Calculated also mean IOU score for validation and test sets.

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$



MRCNN - MS COCO pretrained model

- Pretrained model - MS COCO
- Learning rate - 1e-3
- Epochs - 25

Results:

- **loss: 0.1754**
- rpn_class_loss: 0.0028
- rpn_bbox_loss: 0.0423
- mrcnn_class_loss: 0.0118
- mrcnn_bbox_loss: 0.0352
- mrcnn_mask_loss: 0.0832

Val mean iou score - 0.42223

Results:

- **val_loss: 1.5171**
- val_rpn_class_loss: 0.0427
- val_rpn_bbox_loss: 0.5718
- val_mrcnn_class_loss: 0.0908
- val_mrcnn_bbox_loss: 0.3488
- val_mrcnn_mask_loss: 0.4631

MRCNN - Nucleus pretrained model

- Pretrained model - Nucleus
- Learning rate - 1e-3
- Epochs - 25

Results:

- **loss: 0.2739**
- rpn_class_loss: 0.0044
- rpn_bbox_loss: 0.1092
- mrcnn_class_loss: 0.0141
- mrcnn_bbox_loss: 0.0515
- mrcnn_mask_loss: 0.0947

Val mean iou score - 0.4571

Results:

- **val_loss: 1.6718**
- val_rpn_class_loss: 0.0425
- val_rpn_bbox_loss: 0.8242
- val_mrcnn_class_loss: 0.0703
- val_mrcnn_bbox_loss: 0.3212
- val_mrcnn_mask_loss: 0.4135

MRCNN - Balloon pretrained model

- Pretrained model - Balloon
- Learning rate - 1e-3
- Epochs - 25

Val mean iou score - 0.4366

Results:

- **loss: 0.1508**
- rpn_class_loss: 0.0021
- rpn_bbox_loss: 0.0298
- mrcnn_class_loss: 0.0119
- mrcnn_bbox_loss: 0.0300
- mrcnn_mask_loss: 0.0770

Results:

- **val_loss: 1.4284 - best**
- val_rpn_class_loss: 0.0335
- val_rpn_bbox_loss: 0.4494
- val_mrcnn_class_loss: 0.1022
- val_mrcnn_bbox_loss: 0.3291
- val_mrcnn_mask_loss: 0.5141

MRCNN - Shapes pretrained model

- Pretrained model - Shapes
- Learning rate - 1e-3
- Epochs - 25

Val mean iou score - 0.47776

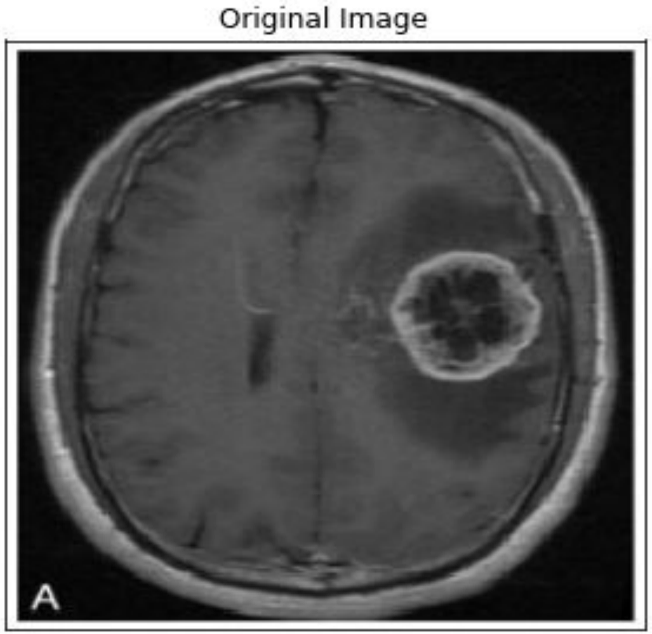
Results:

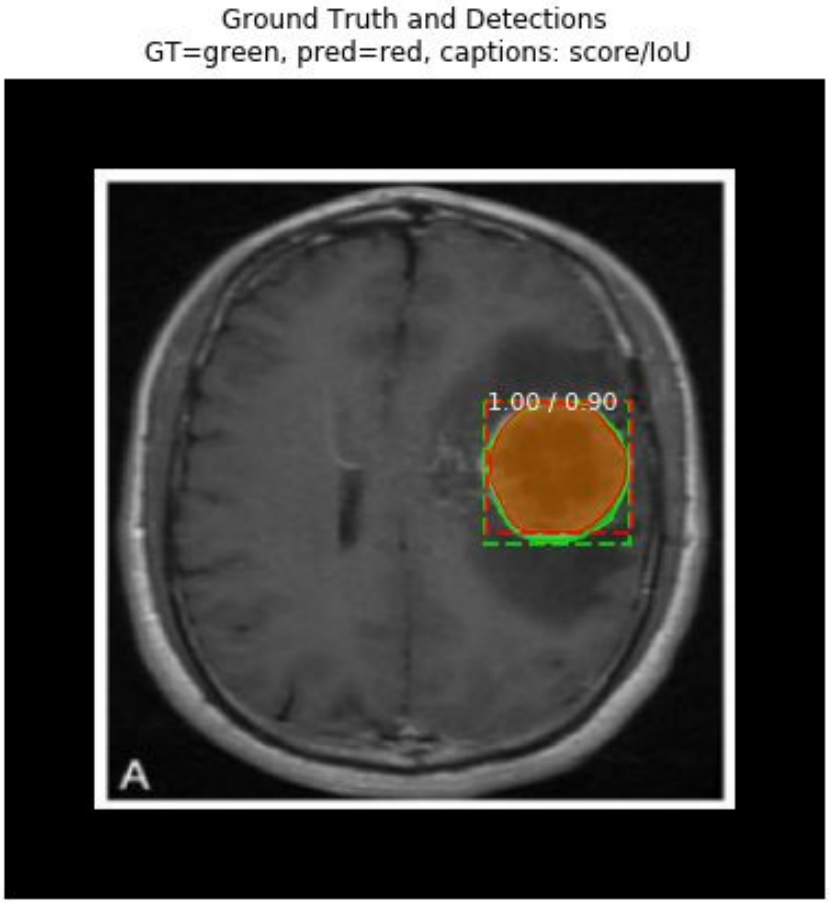
- **loss: 0.2144**
- rpn_class_loss: 0.0032
- rpn_bbox_loss: 0.0770
- mrcnn_class_loss: 0.0118
- mrcnn_bbox_loss: 0.0401
- mrcnn_mask_loss: 0.0823

Results:

- **val_loss: 1.5864**
- val_rpn_class_loss: 0.0472
- val_rpn_bbox_loss: 0.7531
- val_mrcnn_class_loss: 0.0572
- val_mrcnn_bbox_loss: 0.3073
- val_mrcnn_mask_loss: 0.4217

MRCNN results example





Accuracy metrics for Mask-RCNN - MAP

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} \mathrm{AveP(q)}}{Q}$$

Mean average precision formula given provided by Wikipedia

where Q is the number of queries in the set and AveP(q) is the average precision (AP) for a given query, q.

MAP - Mean Average Precision

- True Positive - IoU > 0.5
- False Positive - IoU <= 0.5 or Duplicated BB
- False Negative - IoU > 0.5 but has the wrong classification
- Precision/Recall Curve (PR Curve)
- Interpolated precision

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r})$$

Interpolated Precision for a given Recall Value (r)

MAP - Mean Average Precision

- The AP is then calculated by taking the area under the PR curve.
- The mAP for object detection is the average of the AP calculated for all the classes.

https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52

AP - drawbacks

- not confidence-score sensitive
- does not suggest a confidence score threshold for the best setting of the object detector
- uses interpolation between neighboring recall values

Localization Recall Precision (LRP)

- X - the set of ground truth boxes
- Y - the set of boxes returned by an object detector
- S - score threshold
- Tau - IoU threshold
- Y_s - only the detections that pass the threshold s
- N_{TP} the number of true positives
- N_{FP} the number of false positives
- N_{FN} the number of false negatives

https://arxiv.org/pdf/1807.01696.pdf

LRP error

$$\mathrm{LRP}(X, Y_s) := \frac{1}{Z}\left(w_{IoU}\mathrm{LRP}_{IoU}(X, Y_s) + w_{FP}\mathrm{LRP}_{FP}(X, Y_s) + w_{FN}\mathrm{LRP}_{FN}(X, Y_s)\right), \tag{1}$$

Where $Z = (N_{TP} + N_{FP} + N_{FN})$ is the normalization constant. and the weights $w_{IoU} = N_{TP} / (1 - tau)$, $w_{FP} = |Y_s|$, $w_{FN} = |X|$ control the contributions of the terms.

$$\mathrm{LRP}_{IoU}(X, Y_s) := \frac{1}{N_{TP}}\sum_{i=1}^{N_{TP}}(1 - IoU(x_i, y_{x_i})), \tag{2}$$

Mean bounding box Localization Error.

Another interpretation is that $1 - \mathrm{LRP}_{IoU}$ is the average IoU of the valid detections.

LRP errors

$$\mathrm{LRP}_{FP}(X, Y_s) := 1 - Precision = 1 - \frac{N_{TP}}{|Y_s|} = \frac{N_{FP}}{|Y_s|}, \tag{3}$$

$$\mathrm{LRP}_{FN}(X, Y_s) := 1 - Recall = 1 - \frac{N_{TP}}{|X|} = \frac{N_{FN}}{|X|}. \tag{4}$$

$$\mathrm{LRP}(X, Y_s) := \left(\sum_{i=1}^{N_{TP}} \frac{1 - IoU(x_i, y_{x_i})}{1 - \tau} + N_{FP} + N_{FN} \right) / (N_{TP} + N_{FP} + N_{FN}). \tag{5}$$

Optimal LRP

$$\mathrm{oLRP} := \min_{s} \mathrm{LRP}(X, Y_s). \tag{6}$$

Mean optimal LRP:

$$\mathrm{moLRP} := \frac{1}{|C|} \sum_{c \in C} \mathrm{oLRP}_c. \tag{7}$$

References

- https://www.kaggle.com/ruslankl/brain-tumor-detection-v1-0-cnn-vgg-16
- https://www.kaggle.com/ruslankl/brain-tumor-detection-v2-0-mask-r-cnn
- https://www.pyimagesearch.com/2016/04/11/finding-extreme-points-in-contours-with-opencv/
- https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/
- https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52
- https://arxiv.org/pdf/1807.01696.pdf


THANK YOU
FOR LISTENING!