Machine Learning Project:

Refinement of a Deep Neural Network (DNN) Model to Detect Camouflaged Objects

Sponsor: Dr. Jay Hegde,
Department of Neuroscience

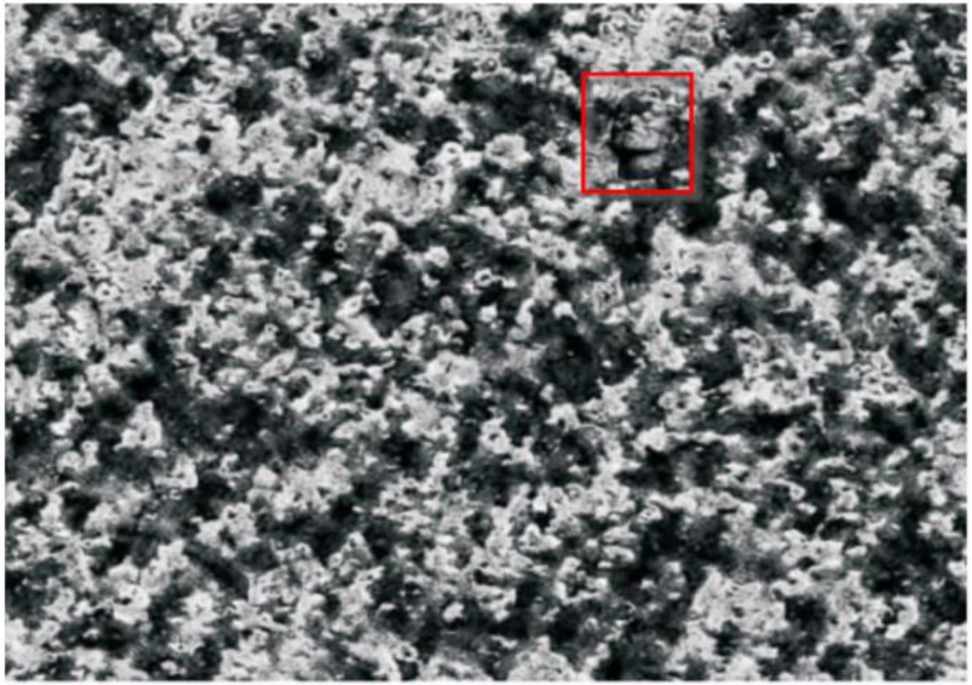
Members:
Atabey Abbasi, Elisha Ebanks, Jacob Doby, Martin Navarrete, and Kaney Nguyen

CamoTect Functional Requirements

1) Image classification: Whereby the DNN classifies a given image as a “positive image” that contains a human head, or a “negative image” that contains no target.

1) Target segmentation, Whereby the DNN outlines the image region containing the head target.

The Goal



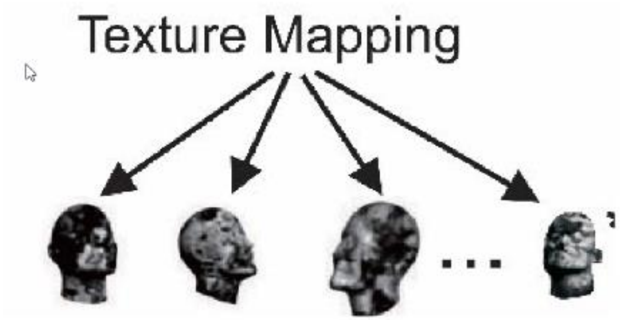


Image Classification

- Image classification is the task of identifying what is in an image. There are multiple types of image classification. The type of image classification used in CamoTect is Binary classification.

- Image classification in CamoTect classifies whether the target image has the camouflaged object head or no head.

- If the target image contains 1 head it is labeled pos img, if 0 head neg img.

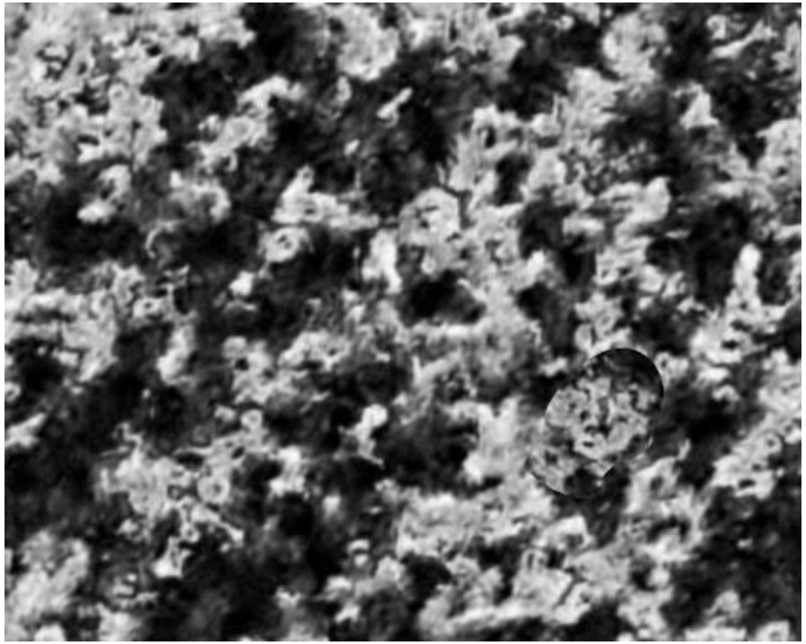
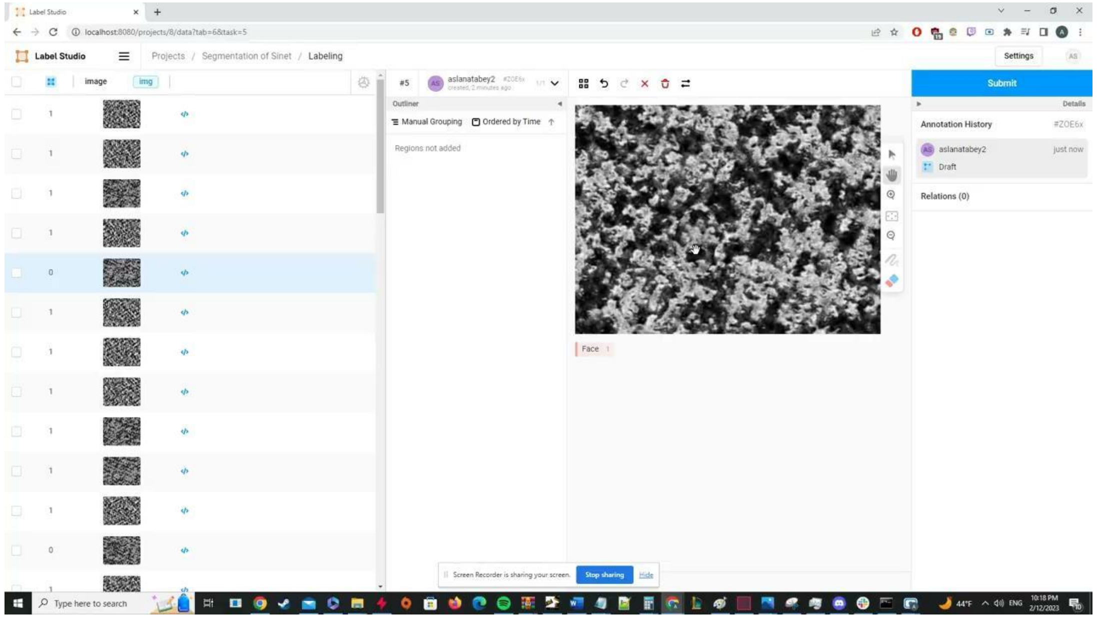


Image Segmentation

Image segmentation divides an image into multiple segments or regions each corresponding to the part of the original image.

Image segmentation is used to change the representation of an image into something easier to analyze.

This process is automated through our image generator.



Prior Work Data

- The group from the previous year only had image classification.
- Q1 = 66.16%
- Q2 = 68.8%
- Q3 = 68.912%
- Q4 = 68.2%
- Average = 68.02%
- Min = 56.4%
- Max = 70%

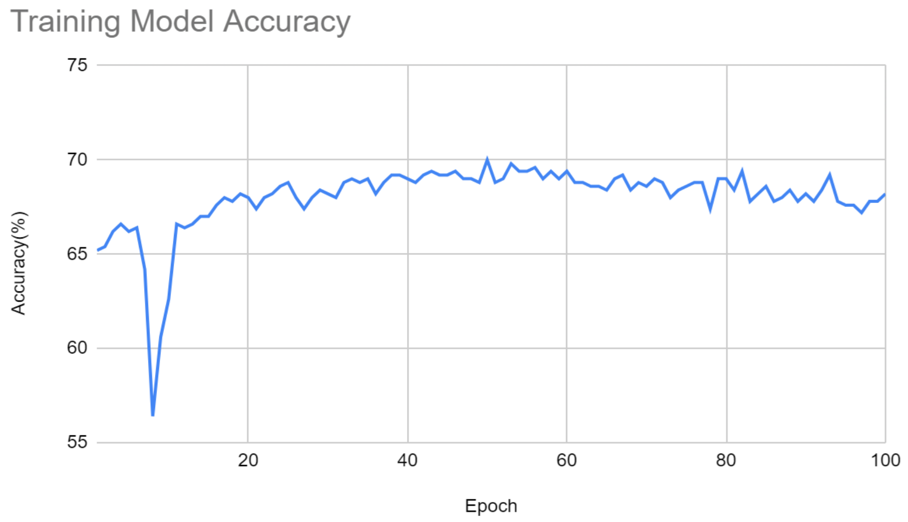


Image Generator - Image Augmentations

- Blender is a 3D computer graphics software and has a powerful Python API.

- Augmentations introduce noise and variation in the training dataset. Helps in making the model more robust.

- Improves the model's generalization ability.

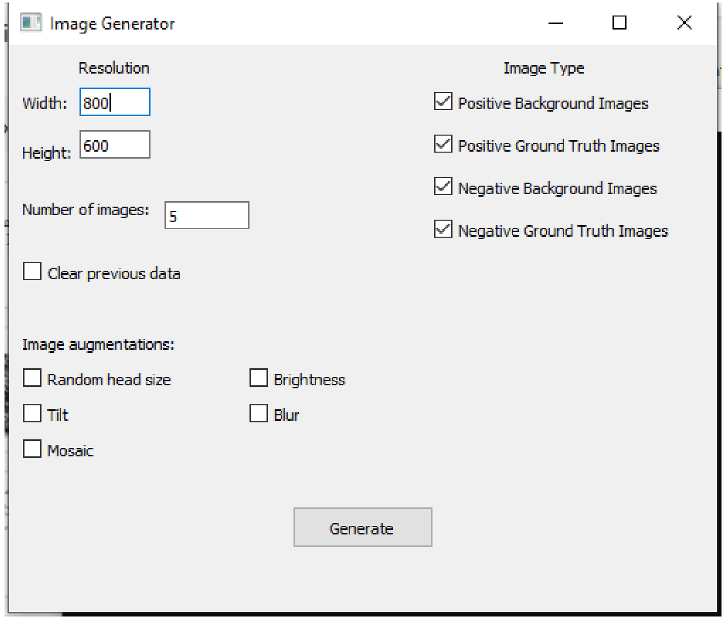


Image Generator

Resolution

Width: 800

Height: 600

Number of images: 5

☐ Clear previous data

Image Type

☑ Positive Background Images

☑ Positive Ground Truth Images

☑ Negative Background Images

☑ Negative Ground Truth Images

Image augmentations:

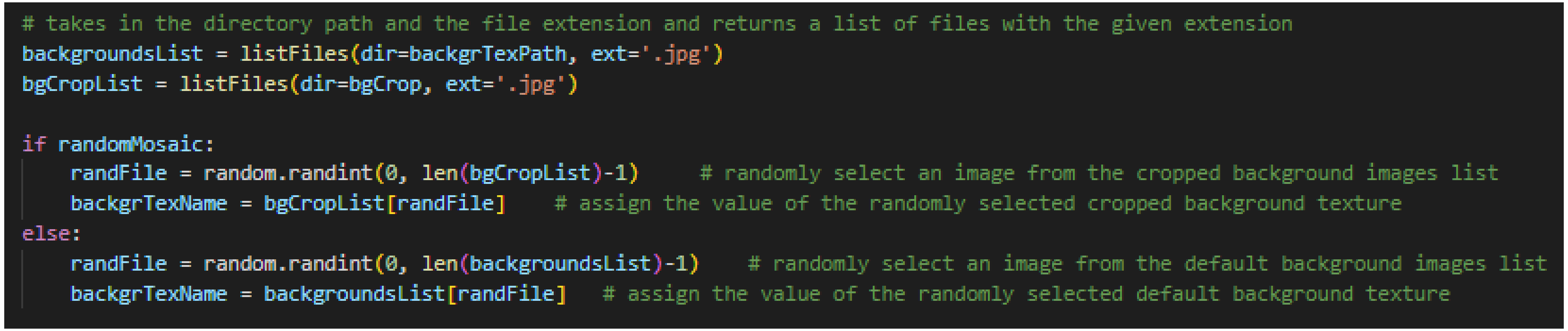
☐ Random head size ☐ Brightness

☐ Tilt ☐ Blur

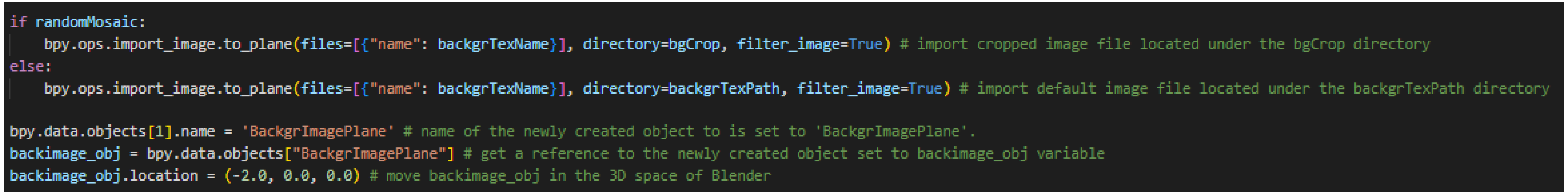
☐ Mosaic

Image Generator GUI

Image Generator - Mosaic Augmentations

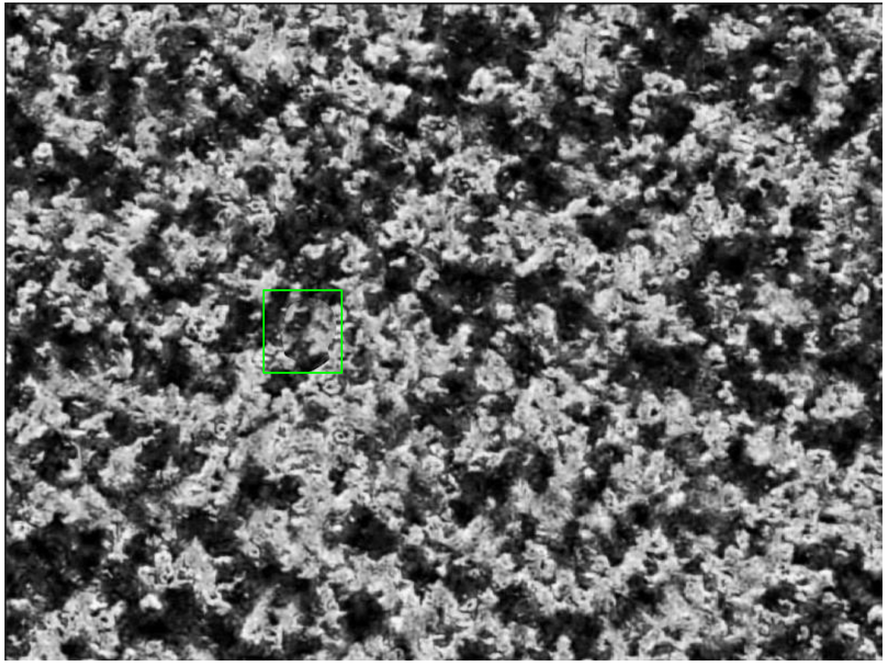


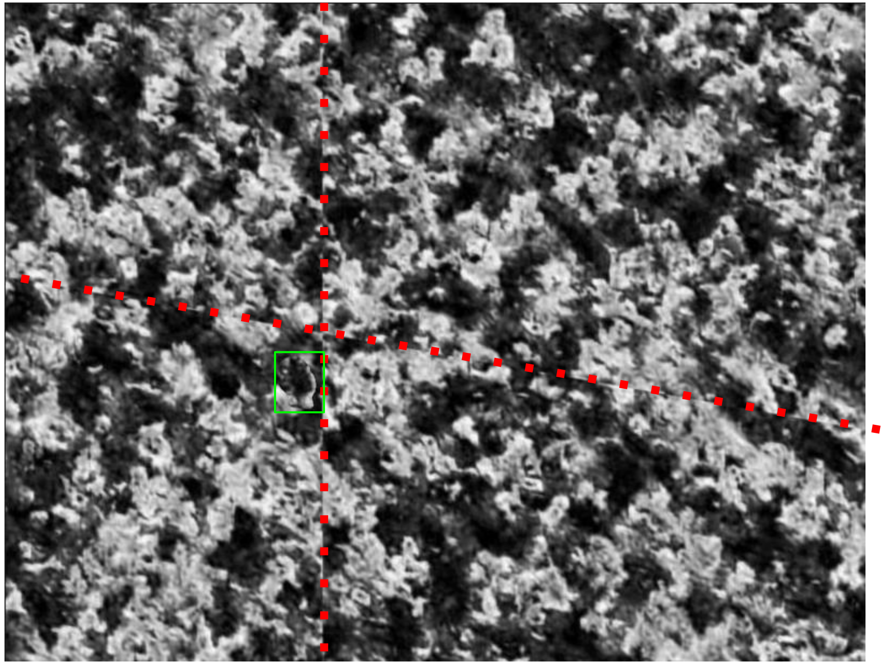
```python
# takes in the directory path and the file extension and returns a list of files with the given extension
backgroundsList = listFiles(dir=backgrTexPath, ext='.jpg')
bgCropList = listFiles(dir=bgCrop, ext='.jpg')

if randomMosaic:
    randFile = random.randint(0, len(bgCropList)-1)     # randomly select an image from the cropped background images list
    backgrTexName = bgCropList[randFile]    # assign the value of the randomly selected cropped background texture
else:
    randFile = random.randint(0, len(backgroundsList)-1)    # randomly select an image from the default background images list
    backgrTexName = backgroundsList[randFile]   # assign the value of the randomly selected default background texture
```



```python
if randomMosaic:
    bpy.ops.import_image.to_plane(files=[{"name": backgrTexName}], directory=bgCrop, filter_image=True) # import cropped image file located under the bgCrop directory
else:
    bpy.ops.import_image.to_plane(files=[{"name": backgrTexName}], directory=backgrTexPath, filter_image=True) # import default image file located under the backgrTexPath directory

bpy.data.objects[1].name = 'BackgrImagePlane' # name of the newly created object to is set to 'BackgrImagePlane'.
backimage_obj = bpy.data.objects["BackgrImagePlane"] # get a reference to the newly created object set to backimage_obj variable
backimage_obj.location = (-2.0, 0.0, 0.0) # move backimage_obj in the 3D space of Blender
```

Image Augmentation Example


Default Positive Image


Background Image With Tilt and Mosaic Augmentations

Challenges

1) Getting the right image augmentations in order for model to perform well to unseen data. Image augmentations such as tilt and mosaic effect were added to make the DNN perform more efficiently upon tested images with variance.

1) Finding the right balance of positive and negative images with the appropriate amount of variance. This step is vital for the DNN to perform efficiently. If there is too much variance in either positive/negatives image might create bias where the DNN classifies every image either as negative or positive regardless of whether the object of interest is present.

K-Fold Cross-Validation

- K = 5(split into 5 subsets)
- Total samples: 3000 images
- Train 2400 Images
- Test 600 Images

Output

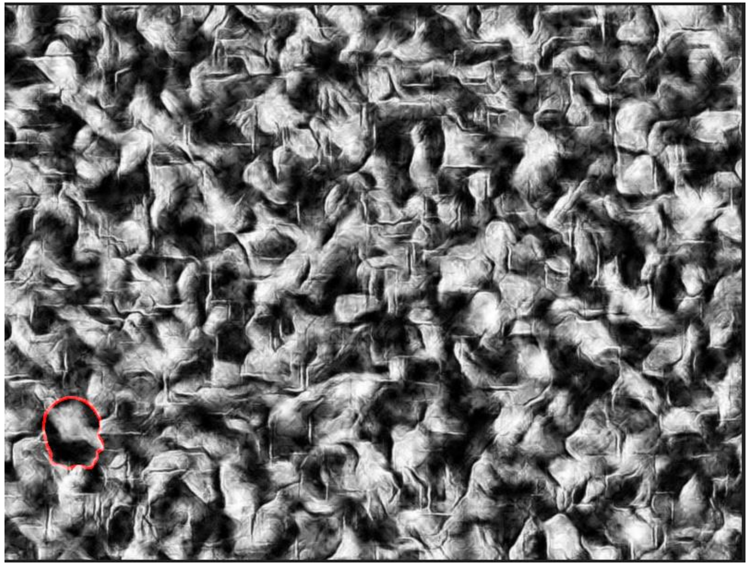
GT

Result(avg of all)

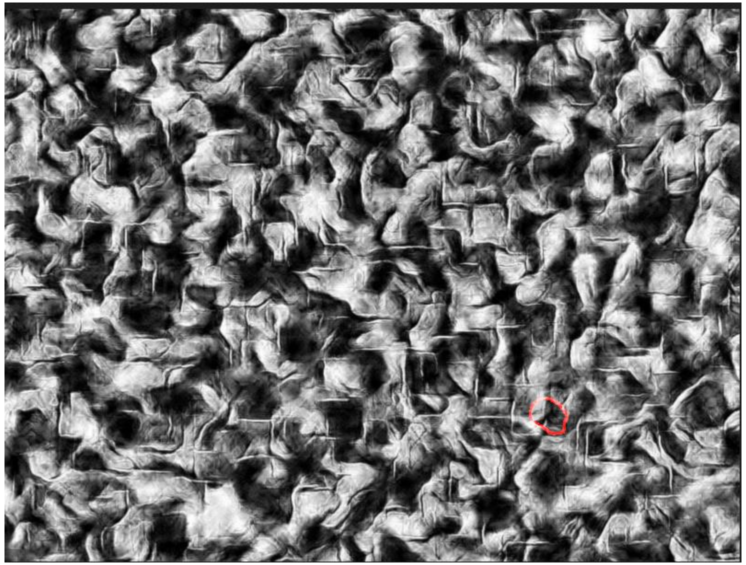
- True Positive: 49.9556%
- True Negative: 49.9562%
- False Positive: 0.0624%
- False Negative: 0.0438%

- Accuracy: 99.91%
- Target Segmentation Accuracy: 96.76%

Output from the DNN



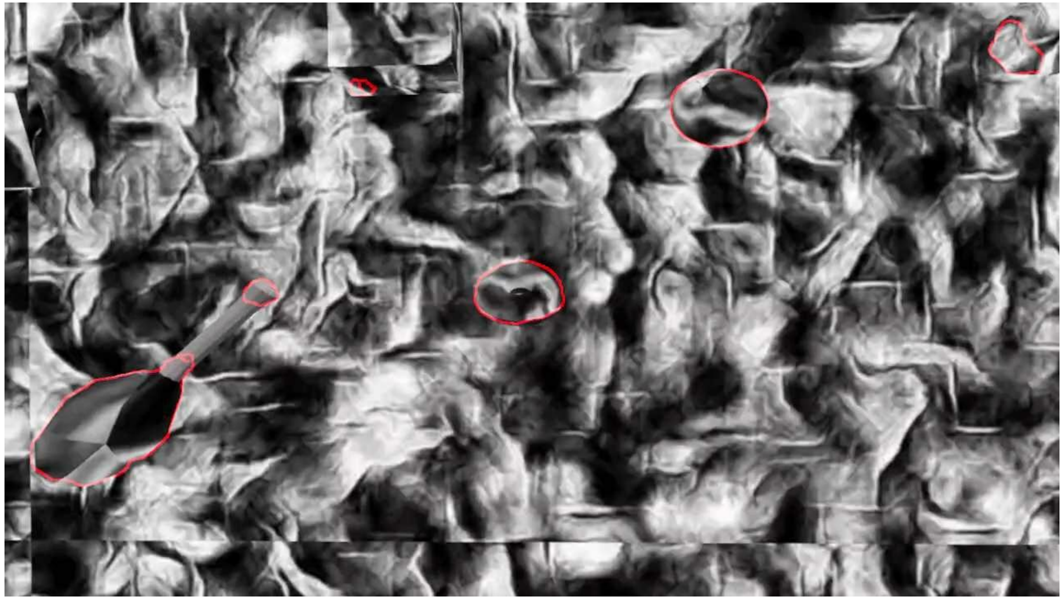
Large Head



Small Head

Easy

Hard(Overlearning/Overfitting)



- **Overlearning/Overfitting** - The system performed really well on trained object but it did not perform well on untrained object due to overlearning or overfitting.

- In the video the system detected other objects that has similar shape to the head.

Conclusion

- Understand the system is overlearning.
- Build a scalable software.
- Able to detect the camouflage object.
- Able to generate good target segmentation.

Future Work

- Solving the head detection problem by adding a confidential threshold for the head.
- Solving the problem by tweaking the DNN.
- Train with objects similar trait as the target object.
- Find the midpoint for training when it is overfitting and underfitting.
- Implement early stop.
- Release as a package
- https://github.com/MagicKey23/PFNet_Ease

Live Demo