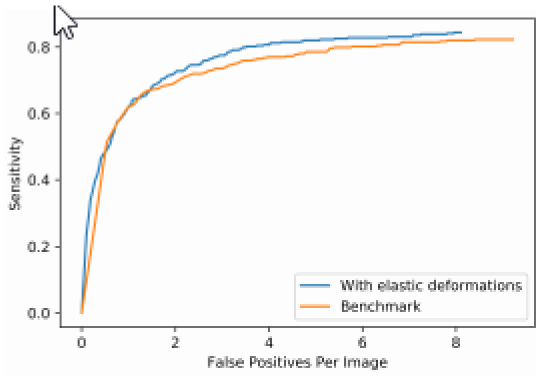
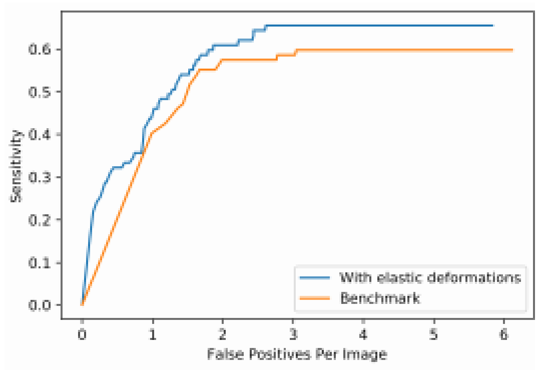
Data augmentation

Data augmentation is an effective method to reduce the “overfitting” of deep CNN caused by limited training samples, which approximates the data probability space by manipulating input samples, such as horizontal flipping, random crop, scale transformation, and noise disturbance. In general, as long as the quantity, quality, and diversity of the data in the dataset are increased, the effectiveness of the model can be improved.

Employed Elastic Deformation in Breast Cancer classification

In a continuous body, a deformation results from a stress field induced by applied forces. If the deformation recovers after the stress field has been removed the deformations are called elastic. This phenomena can realistically occur in breast cancer screening. For each exam, a slightly different stretch is applied due to the position of the breast and the strength used in the compression. As such, the same breast observed in two different screenings may have different appearances, which should not influence the decision of whether a lesion is present or not. Elastic deformations as a data augmentation technique, therefore, comes naturally as a method to model these variations.





Elastic deformation is one of the more extreme forms of augmentation. It is similar to stretching, however, with more freedom. Elastic deformation allows you to change the image almost like kneading a stress ball. See Figure 3 for a visual illustration. One should be careful when applying this strategy as it is easy to overdo it, resulting in almost unrecognizable training images.

Employed Elastic Deformation Modules

2D Architecture:

Kaggle: *https://www.kaggle.com/ori226/data-augmentation-with-elastic-deformations*

Proposed Implementation library:

Kaggle: *https://www.kaggle.com/ori226/data-augmentation-with-elastic-deformations*

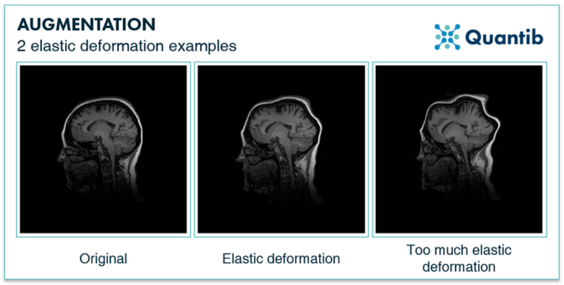


Figure 3: Augmentation can also be accomplished by applying elastic deformation.

```
import Augmentor
p = Augmentor.Pipeline('./pathToImages')
```

```
pip install Augmentor
```

```
p.rotate(probability=0.7, max_left_rotation=10,
max_right_rotation=10)
p.zoom(probability=0.5, min_factor=1.1, max_factor=1.5)
```

```
p.random_distortion(probability=1, grid_width=3, grid_height=3,
magnitude=5)
p.gaussian_distortion(probability=1, grid_width=3, grid_height=3,
magnitude=5, corner='bell', method='in')
```

```
p.sample(10000)
```

```
p.process()
```

```
p.shear(0.5, 10, 10)  #Size Preserving Shearing
p.skew_tilt(0.5, magnitude=0.2) #Perspective Transforms
```

```
def remove_corners(image):
    img=cv2.imread(image,0)
    corners = cv2.goodFeaturesToTrack(img, 100, 0.01, 10)
    corners = np.int0(corners)

    for corner in corners:
        n = random.randint(1,4)
        if(n != 3):
            x,y = corner.ravel()
            cv2.circle(img,(x,y),3,255,-1)
    return img
```