Data preparation

The ultra sound scan data

- 163 scans total, clinically confirmed as having either bening or malignant (cancerous) lesions
- 100 scans for training, 63 for testing
- Training data will be passed through 7 transformations to give us 800 training images total
- Testing images will not be transformed
- All images will be converted from grayscale to RGB
- All images will be resized to 224X224
- Images will then be converted to numpy arrays and saved

In [1]:

```
# some setup code for this notebook
import numpy as np
import matplotlib.pyplot as plt
from functions import data, Timer
timer = Timer()

# This makes matplotlib figures appear inline in the notebook
# rather than in a new window.
%matplotlib inline
plt.rcParams['figure.figsize'] = (10.0, 8.0) # set default size of plots
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'gray'

# Make the notebook reload external python modules;
# see http://stackoverflow.com/questions/1907993/autoreload-of-modules-in-ipython
%load_ext autoreload
%autoreload 2
```

The functions we will be using. Code is in the transform_images.py files

In [2]:

```
#functions
from functions.transform_images import performTransformationSetWithBlurAndSharpen, \
    resize, getFileAddresses, convertToRGB, toNpArray
```

In [3]:

```
# training data locations

training_base = "J:\\final year project\\code and models\\data\\augmented\\training\\"

benign_training_folder = training_base + "benign"
malignant_training_folder = training_base + "malignant"

benign_training_image_files = getFileAddresses(benign_training_folder)
malignant_training_image_files = getFileAddresses(malignant_training_folder)

benign_training_images_np_file = benign_training_folder + "_training.npy"
malignant_training_images_np_file = malignant_training_folder + "_training.npy"

benign_training_labels_np_file = benign_training_folder + "_training_labels.npy"
malignant_training_labels_np_file = malignant_training_folder + "_training_labels.npy"
```

In [4]:

```
# testing data locations

testing_base = "J:\\final year project\\code and models\\data\\original\\testing\\"

benign_testing_folder = testing_base + "benign"
malignant_testing_folder = testing_base + "malignant"

benign_testing_image_files = getFileAddresses(benign_testing_folder)
malignant_testing_image_files = getFileAddresses(malignant_testing_folder)

benign_testing_images_np_file = benign_testing_folder + "_testing.npy"
malignant_testing_images_np_file = malignant_testing_folder + "_testing.npy"

benign_testing_labels_np_file = benign_testing_folder + "_testing_labels.npy"
malignant_testing_labels_np_file = malignant_testing_folder + "_testing_labels.npy"
```

Functions we will be using

In [5]:

```
def transformImages(files):
    timer.start()
    print('transforming %d files' % len(files))
    for file in files:
        performTransformationSetWithBlurAndSharpen(file)
    timer.stop('transforming %d files' % len(files))
```

In [6]:

```
def convertImagesToRGB(files):
    timer.start()
    print('converting %d files to rgb' % len(files))
    for file in files:
        convertToRGB(file)
    timer.stop('converting %d files to rgb' % len(files))
```

In [7]:

```
def resizeImages(files):
    timer.start()
    print('resizing %d files' % len(files))
    for file in files:
        resize(file, (224, 224))
    timer.stop('resizing %d files' % len(files))
```

In [8]:

```
# augment, convert to RGB and resize
def augmentAndPreprocessTrainingImages():
    transformImages(benign_training_image_files)
    transformImages(malignant_training_image_files)

    convertImagesToRGB(getFileAddresses(benign_training_folder))
    convertImagesToRGB(getFileAddresses(malignant_training_folder))

    resizeImages(getFileAddresses(benign_training_folder))
    resizeImages(getFileAddresses(malignant_training_folder))
```

In [9]:

```
# convert to rgb and resize, no transformations this time
def preprocessTestingImages():
    convertImagesToRGB(getFileAddresses(benign_testing_folder))
    convertImagesToRGB(getFileAddresses(malignant_testing_folder))

    resizeImages(getFileAddresses(benign_testing_folder))
    resizeImages(getFileAddresses(malignant_testing_folder))
```

In [10]:

```
# convert image files to a numpy array
def convertImageFilesToNpArray(files):
    first_image = toNpArray(files[0])
    try:
        color_channels = first_image.shape[2]
    except Exception:
        color_channels = 1

    dimension1 = len(files)
    dimension2 = first_image.shape[0]
    dimension3 = first_image.shape[1]

    if color_channels == 3:
        dimension4 = 3
        image_array = np.zeros((dimension1, dimension2, dimension3, dimension4))
    else:
        image_array = np.zeros((dimension1, dimension2, dimension3))

    labels_array = np.zeros((dimension1, 1), dtype=np.int32)
    for i in range(dimension1):
        image_array[i] = toNpArray(files[i])

    if files == getFileAddresses(benign_training_folder) or files == getFileAddresses(benig
        for i in range(dimension1):
            labels_array[i] = 0
    elif files == getFileAddresses(malignant_training_folder) or files == getFileAddresses(
        for i in range(dimension1):
            labels_array[i] = 1

    if files == getFileAddresses(benign_training_folder):
        np.save(benign_training_images_np_file, image_array)
        np.save(benign_training_labels_np_file, labels_array)
    elif files == getFileAddresses(benign_testing_folder):
        np.save(benign_testing_images_np_file, image_array)
        np.save(benign_testing_labels_np_file, labels_array)
    elif files == getFileAddresses(malignant_training_folder):
        np.save(malignant_training_images_np_file, image_array)
        np.save(malignant_training_labels_np_file, labels_array)
    elif files == getFileAddresses(malignant_testing_folder):
        np.save(malignant_testing_images_np_file, image_array)
        np.save(malignant_testing_labels_np_file, labels_array)

    return image_array, labels_array
```

In [11]:

```
# helper function for getting training data
def getTrainingDataNp(reload=False):
    timer.start()
    if reload:
        x_benign, y_benign = convertImageFilesToNpArray(getFileAddresses(benign_training_fo
        x_malignant, y_malignant = convertImageFilesToNpArray(getFileAddresses(malignant_tr

        x = np.concatenate((x_benign, x_malignant))
        y = np.concatenate((y_benign, y_malignant))

        timer.stop("getting training data")
        return x, y
    else:
        x_benign = np.load(benign_training_images_np_file)
        y_benign = np.load(benign_training_labels_np_file)

        x_malignant = np.load(malignant_training_images_np_file)
        y_malignant = np.load(malignant_training_labels_np_file)

        x = np.concatenate((x_benign, x_malignant))
        y = np.concatenate((y_benign, y_malignant))

        timer.stop("getting training data")
        return x, y
```

In [12]:

```
# helper function for getting testing data
def getTestingDataNp(reload=False):
    timer.start()
    if reload:
        x_benign, y_benign = convertImageFilesToNpArray(getFileAddresses(benign_testing_fol
        x_malignant, y_malignant = convertImageFilesToNpArray(getFileAddresses(malignant_te

        x = np.concatenate((x_benign, x_malignant))
        y = np.concatenate((y_benign, y_malignant))

        timer.stop("getting testing data")
        return x, y
    else:
        x_benign = np.load(benign_testing_images_np_file)
        y_benign = np.load(benign_testing_labels_np_file)

        x_malignant = np.load(malignant_testing_images_np_file)
        y_malignant = np.load(malignant_testing_labels_np_file)

        x = np.concatenate((x_benign, x_malignant))
        y = np.concatenate((y_benign, y_malignant))

        timer.stop("getting testing data")
        return x, y
```

In [13]:

```
#helper function for loading all the data
def getData():
    x, y = getTrainingDataNp(reload=True)

    x_test, y_test = getTestingDataNp(reload=True)

    return x, y, x_test, y_test
```

In [14]:

```
# preparation done, now we do the transformations
augmentAndPreprocessTrainingImages()
```

```
transforming 67 files
Timing:: took 49 second(s) transforming 67 files
----------------------------------
transforming 33 files
Timing:: took 16 second(s) transforming 33 files
----------------------------------
converting 536 files to rgb
Timing:: took 77 second(s) converting 536 files to rgb
----------------------------------
converting 264 files to rgb
Timing:: took 59 second(s) converting 264 files to rgb
----------------------------------
resizing 536 files
Timing:: took 45 second(s) resizing 536 files
----------------------------------
resizing 264 files
Timing:: took 17 second(s) resizing 264 files
----------------------------------
```

In [15]:

```
# for test images
preprocessTestingImages()
```

```
converting 42 files to rgb
Timing:: took 10 second(s) converting 42 files to rgb
----------------------------------
converting 21 files to rgb
Timing:: took 5 second(s) converting 21 files to rgb
----------------------------------
resizing 42 files
Timing:: took 2 second(s) resizing 42 files
----------------------------------
resizing 21 files
Timing:: took 0 second(s) resizing 21 files
----------------------------------
```

In [16]:

```
# we confirm data was pre processed correctly
x_train, y_train, x_test, y_test = getData()
print('Training data shape: ', x_train.shape)
print('Training labels shape: ', y_train.shape)
print('Test data shape: ', x_test.shape)
print('Test labels shape: ', y_test.shape)
```

```
Timing:: took 18 second(s) getting training data
----------------------------------
Timing:: took 1 second(s) getting testing data
----------------------------------
Training data shape:  (800, 224, 224, 3)
Training labels shape:  (800, 1)
Test data shape:  (63, 224, 224, 3)
Test labels shape:  (63, 1)
```

In [ ]:

```

```