In [1]:
```
# Import libraries and functions

from __future__ import print_function
import numpy as np
np.random.seed(1337)  # for reproducibility

import csv
import os,random

from keras.models import Sequential
from keras.layers import Dense
from keras.layers.core import Dropout, Activation, Flatten
from keras.layers import LSTM, SimpleRNN, Bidirectional
from keras.layers.advanced_activations import LeakyReLU
from keras.optimizers import *
from keras.callbacks import EarlyStopping

import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.metrics import mean_squared_error
from math import sqrt

import datetime
```

```
C:\Users\mm1656.UNT\.conda\envs\py35\lib\site-packages\h5py\__init__.py:36: Future
Warning: Conversion of the second argument of issubdtype from `float` to `np.float
ing` is deprecated. In future, it will be treated as `np.float64 == np.dtype(floa
t).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [2]:
```
timesteps = 240 # Number of timesteps
nr = 90   # Number of repetitions
n_dim = 117  # Dimension of data sequences
```

In [3]:
```
# Function to load the data
def load_data(nr, n_dim, timesteps):
    f = open('../../Data/Data_Correct_m02.csv')
    csv_f = csv.reader(f)
    Correct_X = list(csv_f)

    # Convert the input sequences into numpy arrays
    train_input1 = np.asarray(Correct_X, dtype = float)
    correct_input = np.zeros((nr,timesteps,n_dim))
    for i in range(len(train_input1)//n_dim):
           correct_input[i,:,:] = np.transpose(train_input1[n_dim*i:n_dim*(i+1),:])

    f = open('../../Data/Labels_Correct.csv')
    csv_f = csv.reader(f)
    Correct_Y = list(csv_f)

    # Convert the input labels into numpy arrays
    correct_label = np.asarray(Correct_Y, dtype = float)

    f = open('../../Data/Data_Incorrect_m02.csv')
    csv_f = csv.reader(f)
    Incorrect_X = list(csv_f)

    # Convert the input sequences into numpy arrays
    test_input1 = np.asarray(Incorrect_X)
    n_dim = 117
    incorrect_input = np.zeros((nr,timesteps,n_dim))
    for i in range(len(test_input1)//n_dim):
          incorrect_input[i,:,:] = np.transpose(test_input1[n_dim*i:n_dim*(i+1),:])

    f = open('../../Data/Labels_Incorrect.csv')
    csv_f = csv.reader(f)
    Incorrect_Y = list(csv_f)

    # Convert the input labels into numpy arrays
    incorrect_label = np.asarray(Incorrect_Y, dtype = float)
```

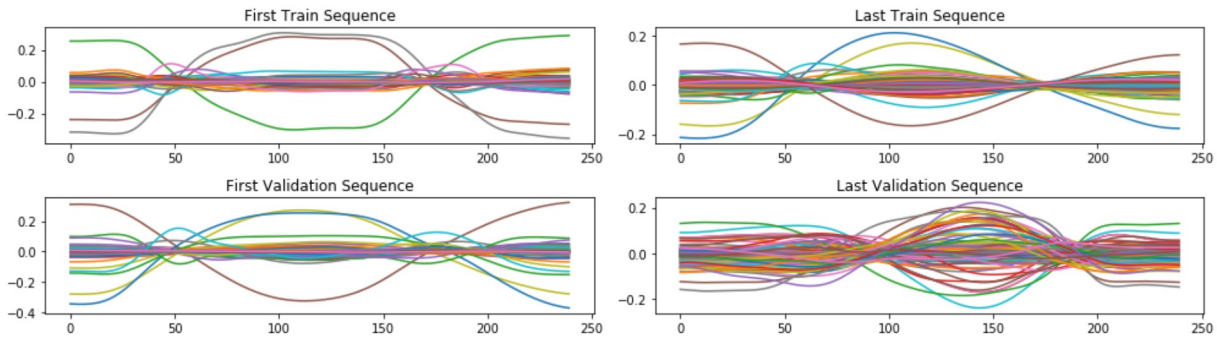
In [4]:
```
# Load the data
Correct_data, Correct_label, Incorrect_data, Incorrect_label = load_data(nr, n_dim, ti

# Print the size of the data
print(Correct_data.shape, 'correct sequences')
print(Correct_label.shape, 'correct labels')
print(Incorrect_data.shape, 'incorrect sequences')
```

```
(90, 240, 117) correct sequences
(90, 1) correct labels
(90, 240, 117) incorrect sequences
(90, 1) incorrect labels
```

In [5]:
```
# Split the data into training and validation sets
# Training set: 70%
# Validation set: 30%

# Sample random indices
trainidx1 = random.sample(range(0,Correct_data.shape[0]),int(nr*0.7))
trainidx2 = random.sample(range(0,Incorrect_data.shape[0]),int(nr*0.7))
valididx1 = np.setdiff1d(np.arange(0,nr,1),trainidx1)
valididx2 = np.setdiff1d(np.arange(0,nr,1),trainidx2)

# Training set: data and labels
train_x = np.concatenate((Correct_data[trainidx1,:,:],Incorrect_data[trainidx2,:,:]))
print(train_x.shape, 'training data')
train_y = np.concatenate((np.squeeze(Correct_label[trainidx1]),np.squeeze(Incorrect_la
print(train_y.shape, 'training labels')

# Validation set: data and labels
valid_x = np.concatenate((Correct_data[valididx1,:,:],Incorrect_data[valididx2,:,:]))
print(valid_x.shape, 'validation data')
valid_y = np.concatenate((np.squeeze(Correct_label[valididx1]),np.squeeze(Incorrect_la
```

```
(124, 240, 117) training data
(124,) training labels
(56, 240, 117) validation data
(56,) validation labels
```

In [6]:
```
# Plot the first and last sequence in the training and validation data
plt.figure(figsize = (14,4))
plt.subplot(2,2,1)
plt.plot(train_x[0])
plt.title('First Train Sequence')
plt.subplot(2,2,2)
plt.plot(train_x[-1])
plt.title('Last Train Sequence')
plt.subplot(2,2,3)
plt.plot(valid_x[0])
plt.title('First Validation Sequence')
plt.subplot(2,2,4)
plt.plot(valid_x[-1])
plt.title('Last Validation Sequence')
plt.tight_layout()
```



In [7]:
```
# Build RNN model ...
def Network():
    model = Sequential()

    model.add(Bidirectional(LSTM(20, recurrent_dropout = 0.5, return_sequences = True)
    model.add(Dropout(0.25))

    model.add(Dense(30, activation = 'tanh'))
    model.add(Dropout(0.5))

    model.add(Bidirectional(LSTM(10, recurrent_dropout = 0.5)))
    model.add(Dropout(0.25))

    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer=Adam())
    #model.summary()

    # Measure the training time and implement early stoping
    t = now()

    early_stopping = EarlyStopping(monitor='val_loss', patience = 100)

    history = model.fit(train_x, train_y, batch_size=10, epochs=5000, verbose=0,
                        validation_data=(valid_x, valid_y),
                        callbacks = [early_stopping])
    print('Training time: %s' % (now() - t))

    # Plot the results
    plt.figure(1)
    plt.subplot(221)
    plt.plot(history.history['loss'])
    plt.title('Training Loss')
    plt.subplot(222)
    plt.plot(history.history['val_loss'])
    plt.title('Validation Loss')
    plt.tight_layout()
    plt.show()

    # Plot the prediction of the RNN model for the training and validation sets
    pred_train = model.predict(train_x)
    pred_test = model.predict(valid_x)

    plt.figure(figsize = (8,8))
    plt.subplot(2,1,1)
    plt.plot(pred_train,'s', color='red', label='Prediction', linestyle='None', alpha
    plt.plot(train_y,'o', color='green',label='Quality Score', alpha = 0.4, markersize
    plt.ylim([-0.1,1.1])
    plt.title('Training Set',fontsize=18)
    plt.xlabel('Sequence Number',fontsize=16)
    plt.ylabel('Quality Scale',fontsize=16)
    plt.legend(loc=3, prop={'size':14}) # loc:position
    plt.subplot(2,1,2)
    plt.plot(pred_test,'s', color='red', label='Prediction', linestyle='None', alpha =
    plt.plot(valid_y,'o', color='green',label='Quality Score', alpha = 0.4, markersize
    plt.title('Testing Set',fontsize=18)
    plt.ylim([-0.1,1.1])
    plt.xlabel('Sequence Number',fontsize=16)
    plt.ylabel('Quality Scale',fontsize=16)
    plt.legend(loc=3, prop={'size':14}) # loc:position
    plt.tight_layout()
    plt.savefig('../../Results/RNN_Vicon_Scores.png', dpi=300)
    plt.show()
```

In [8]:

```
Training time: 0:11:04.495134
```