Graph Plot for GRU Models

December 12, 2018

```
In [ ]: # Depression Analysis in Bangla with GRU-RNN RESULTS
        # copyright (c) ABDUL HASIB UDDIN <abdulhasibuddin@gmail.com>
        # LICENSE: GNU General Public License v3.0

In [1]: import matplotlib.pyplot as plt
        from scipy.interpolate import spline
        import numpy as np

In [2]: # GRU MODEL STATISTICS:

In [3]: # gru validation accuracies:
        gru_unique_1 = [0.427,0.318,0.300,0.545,0.500,0.400,0.509,0.673,0.464,0.436,0.545,0.59
        gru_unique_2 = [0.436,0.391,0.309,0.709,0.445,0.427,0.500,0.709,0.391,0.409,0.627,0.73
        gru_unique_3 = [0.355,0.355,0.309,0.700,0.364,0.473,0.509,0.618,0.427,0.436,0.718,0.51
        gru_unique_4 = [0.291,0.273,0.318,0.582,0.455,0.418,0.664,0.736,0.464,0.445,0.664,0.48
        gru_unique_5 = [0.336,0.318,0.336,0.318,0.500,0.455,0.682,0.509,0.464,0.500,0.745,0.51
        gru_unique_6 = [0.490,0.410,0.680]
        gru_unique_7 = [0.560,0.400,0.610,0.580,0.620,0.590,0.560]
        gru_unique_8 = [0.435,0.296,0.348,0.470,0.409,0.409,0.730,0.539,0.374,0.426,0.426,0.51
        gru_unique_9 = [0.432,0.534,0.551,0.441,0.398,0.415,0.458,0.356,0.398,0.364,0.356,0.40
        gru_unique_10 = [0.420,0.340,0.510,0.390,0.430,0.530,0.550,0.620,0.520,0.480,0.600,0.5
        gru_unique_11 = [0.420,0.360,0.650,0.390,0.460,0.760,0.510]
        gru_unique_12 = [0.470,0.383,0.365,0.409,0.461,0.417,0.713,0.696,0.522,0.478,0.452,0.6
        gru_unique_13 = [0.504,0.452,0.443,0.287,0.391,0.496,0.670,0.504,0.504,0.470,0.348,0.4
        gru_unique_14 = [0.504,0.330,0.330,0.435,0.443,0.435,0.722,0.391,0.365,0.470,0.417,0.4

        print(len(gru_unique_9))

75

In [4]: # iterations (x-axis):

        x_axis_1 = []
        for iter_no in range(1,470+1):
            if iter_no%25 == 0:
                x_axis_1.append(iter_no)
        x_axis_2 = []
```

```
for iter_no in range(1,470+1):
    if iter_no%25 == 0:
        x_axis_2.append(iter_no)
x_axis_3 = []
for iter_no in range(1,470+1):
    if iter_no%25 == 0:
        x_axis_3.append(iter_no)
x_axis_4 = []
for iter_no in range(1,470+1):
    if iter_no%25 == 0:
        x_axis_4.append(iter_no)
x_axis_5 = []
for iter_no in range(1,470+1):
    if iter_no%25 == 0:
        x_axis_5.append(iter_no)
x_axis_6 = []
for iter_no in range(1,90+1):
    if iter_no%25 == 0:
        x_axis_6.append(iter_no)
x_axis_7 = []
for iter_no in range(1,180+1):
    if iter_no%25 == 0:
        x_axis_7.append(iter_no)
x_axis_8 = []
for iter_no in range(1,560+1):
    if iter_no%25 == 0:
        x_axis_8.append(iter_no)
x_axis_9 = []
for iter_no in range(1,1880+1):
    if iter_no%25 == 0:
        x_axis_9.append(iter_no)
x_axis_10 = []
for iter_no in range(1,370+1):
    if iter_no%25 == 0:
        x_axis_10.append(iter_no)
x_axis_11 = []
for iter_no in range(1,185+1):
    if iter_no%25 == 0:
        x_axis_11.append(iter_no)
x_axis_12 = []
for iter_no in range(1,560+1):
    if iter_no%25 == 0:
        x_axis_12.append(iter_no)
x_axis_13 = []
for iter_no in range(1,560+1):
    if iter_no%25 == 0:
        x_axis_13.append(iter_no)
x_axis_14 = []
```

```
        for iter_no in range(1,1880+1):
            if iter_no%25 == 0:
                x_axis_14.append(iter_no)
```

```
In [5]: x_list = [x_axis_1,x_axis_2,x_axis_3,x_axis_4,x_axis_5,x_axis_6,x_axis_7,x_axis_8,x_axi
        model_list = [gru_unique_1,gru_unique_2,gru_unique_3,gru_unique_4,gru_unique_5,gru_uniq
        required_iteration_list = [470,470,470,470,470,90,180,560,1880,370,185,560,560,1880]
        required_epoch_list = [5,5,5,5,5,5,10,3,2,10,5,3,3,10]
        test_acc_list = [59.1,70.0,67.3,74.5,69.1,52.0,61.0,75.7,70.3,57.0,61.0,74.8,69.6,56.5]

        print(len(x_list))
        print(len(model_list))
        print(len(required_iteration_list))
        print(len(required_epoch_list))
        print(len(test_acc_list))
```

```
14
14
14
14
14
```

```
In [6]: # average validation accuracies:
        avg_val_acc_list = []
        for model in model_list:
            avg_val_acc_list.append(sum(model)/len(model))
        print(len(avg_val_acc_list))
        print(avg_val_acc_list)
```

```
14
[0.49994444444444436, 0.5054444444444444, 0.4828333333333334, 0.47883333333333344, 0.48577777777
```

```
In [7]: best_model_val_acc = gru_unique_8
        best_model_val_loss = [0.247,0.182,0.202,0.240,0.205,0.265,0.066,0.272,0.192,0.190,0.1

        print(len(best_model_val_acc))
        print(len(best_model_val_loss))
```

```
22
22
```

```
In [ ]:
```

```
In [8]: for i in range(0,len(x_list)):
            x_axis_name = 'len(x_axis_'+str(i+1)+') ='
            model_name = '; len(gru_unique_'+str(i+1)+') ='
            x_axis_length = len(x_list[i])
            model_length = len(model_list[i])
            print(x_axis_name,x_axis_length, model_name,model_length, '; status =',x_axis_leng
```

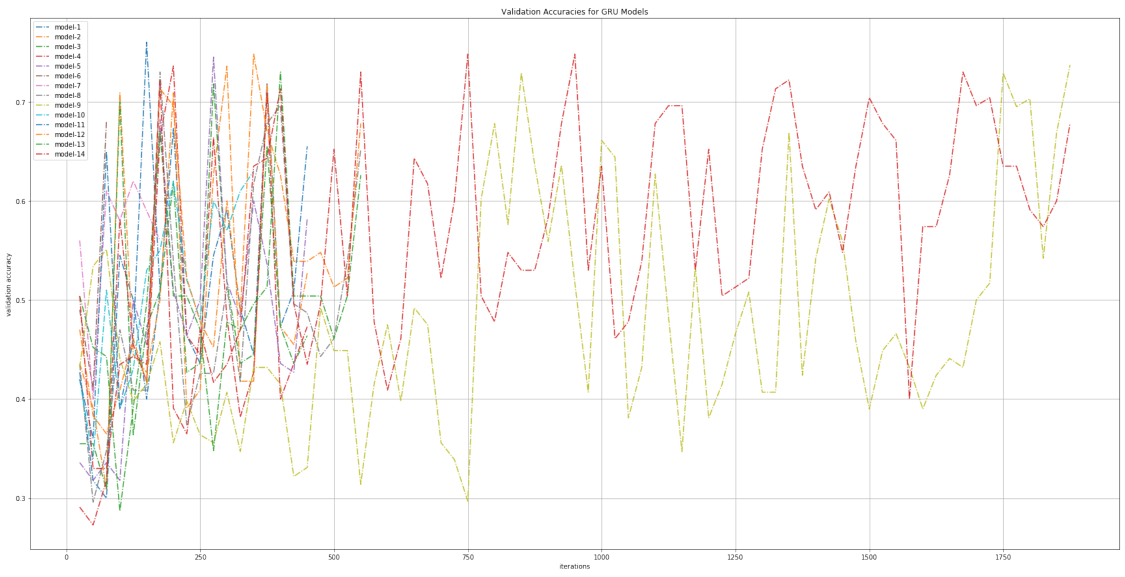
```
len(x_axis_1) = 18 ; len(gru_unique_1) = 18 ; status = True
len(x_axis_2) = 18 ; len(gru_unique_2) = 18 ; status = True
len(x_axis_3) = 18 ; len(gru_unique_3) = 18 ; status = True
len(x_axis_4) = 18 ; len(gru_unique_4) = 18 ; status = True
len(x_axis_5) = 18 ; len(gru_unique_5) = 18 ; status = True
len(x_axis_6) = 3 ; len(gru_unique_6) = 3 ; status = True
len(x_axis_7) = 7 ; len(gru_unique_7) = 7 ; status = True
len(x_axis_8) = 22 ; len(gru_unique_8) = 22 ; status = True
len(x_axis_9) = 75 ; len(gru_unique_9) = 75 ; status = True
len(x_axis_10) = 14 ; len(gru_unique_10) = 14 ; status = True
len(x_axis_11) = 7 ; len(gru_unique_11) = 7 ; status = True
len(x_axis_12) = 22 ; len(gru_unique_12) = 22 ; status = True
len(x_axis_13) = 22 ; len(gru_unique_13) = 22 ; status = True
len(x_axis_14) = 75 ; len(gru_unique_14) = 75 ; status = True
```

```
In [9]: '''
        x_axis = x_axis_9
        y_axis = []
        for i in range(10+1):
            y_axis.append(i/10)

        print(y_axis)
        print(len(x_axis))
        print(len(y_axis))
        '''
```

```
Out[9]: '\nx_axis = x_axis_9\ny_axis = []\nfor i in range(10+1):\n    y_axis.append(i/10)\n
```

```
In [10]: # gru plotting validation accuracies against iterations:

         linestyle='-.'
         linewidth = 2.5

         plt.figure(figsize=(30,15))
         plt.title('Validation Accuracies for GRU Models')
         plt.xlabel('iterations')
         plt.ylabel('validation accuracy')
         #plt.plot(x_axis, y_axis_dummy)
         #plt.plot(x_axis_dummy, y_axis)
         #plt.xticks(x_axis)
         #plt.yticks(y_axis)

         for i in range(0,len(x_list)):
             label = "model-"+str(i+1)
```

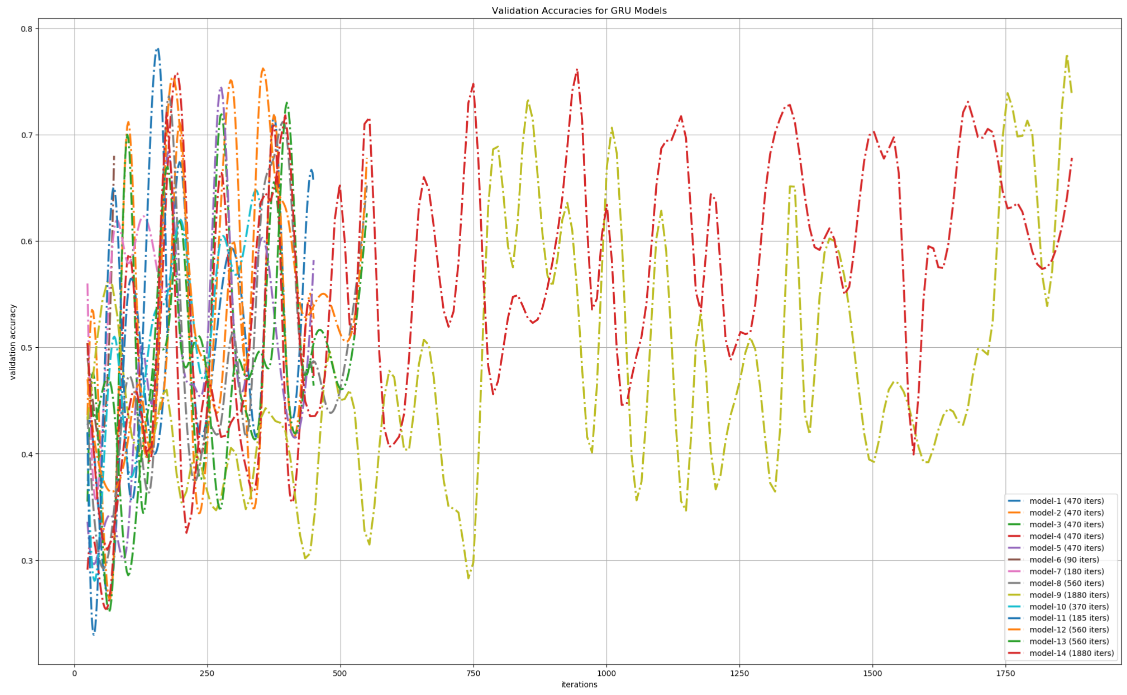
```
    x = x_list[i]
    y = model_list[i]
    plt.plot(x, y, linestyle=linestyle, label=label)
    #plt.plot(x, y, marker='o', markersize=5, linestyle=linestyle, label=label)

#plt.axis([min(gru_x_axis),max(gru_x_axis),0,1])
plt.grid(True)
plt.legend()
plt.show()
```



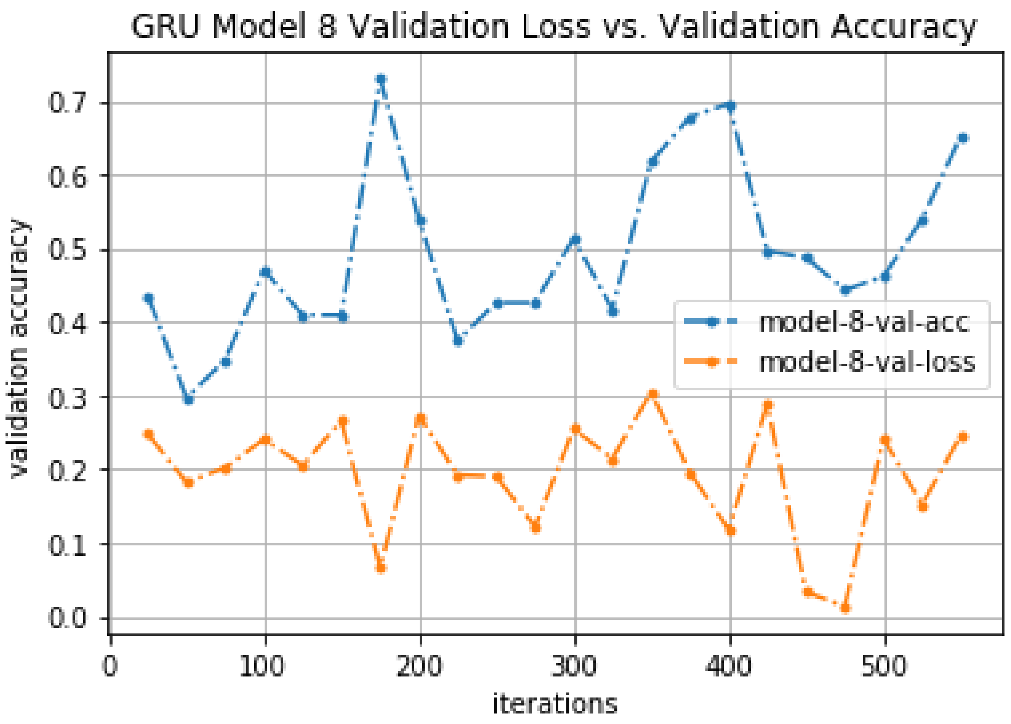
```
In [20]: # gru plotting validation accuracies against iterations:
         smoothing_factor = 200
         linestyle='-.'
         linewidth = 2.5
         legend_properties = {'weight':'bold'}

         #plt.savefig('gru_plotting _validation _accuracies _against _iterations.png')
         #plt.figure(figsize=(13,8))
         plt.figure(figsize=(25,15))
         plt.title('Validation Accuracies for GRU Models')
         plt.xlabel('iterations')
         plt.ylabel('validation accuracy')
         '''
         x = x_axis_1
         y = gru_unique_1
         x_sm = np.array(x)
         y_sm = np.array(y)
         x_smooth = np.linspace(x_sm.min(), x_sm.max(), smoothing_factor)
```

```
        y_smooth = spline(x, y, x_smooth)
        plt.plot(x_smooth, y_smooth, marker='o', linestyle=linestyle, markersize=3, label="gr
        #plt.plot(x_axis_1, gru_unique_1, marker='o',  markersize=5, label="gru model 1")
        '''
        #x_list = [x_axis_1,x_axis_2,x_axis_3,x_axis_4,x_axis_5,x_axis_6,x_axis_7,x_axis_8,x_
        #y_list = [gru_unique_1,gru_unique_2,gru_unique_3,gru_unique_4,gru_unique_5,gru_uniqu

        for i in range(0,len(x_list)):
            label = "model-"+str(i+1)+" ("+str(required_iteration_list[i])+" iters)"
            x = x_list[i]
            y = model_list[i]
            x_sm = np.array(x)
            y_sm = np.array(y)
            x_smooth = np.linspace(x_sm.min(), x_sm.max(), smoothing_factor)
            y_smooth = spline(x, y, x_smooth)
            plt.plot(x_smooth, y_smooth, linestyle=linestyle, linewidth=linewidth, label=label
            #plt.plot(x, y, marker='o', markersize=5, linestyle=linestyle, label=label)

        #plt.axis([min(gru_x_axis),max(gru_x_axis),0,1])
        plt.grid(True)
        #plt.legend(prop=legend_properties)
        plt.legend()
        plt.savefig('images\gru_image_1_plotting _validation _accuracies _against _iterations
        plt.show()
        #plt.savefig('gru_plotting _validation _accuracies _against _iterations.png', bbox_in
```

```
c:\python36\lib\site-packages\ipykernel_launcher.py:33: DeprecationWarning: `spline` is depreca
spline is deprecated in scipy 0.19.0, use Bspline class instead.
```

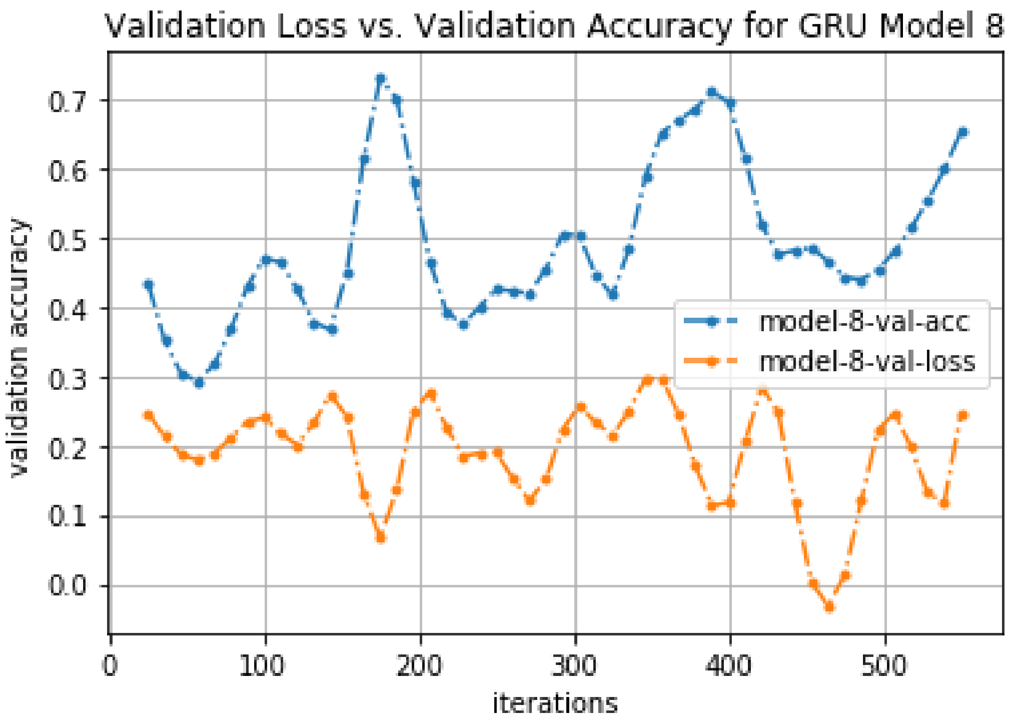


```
In [ ]:

In [12]: # gru best model validation loss vs validation accuracy:

         #plt.figure(figsize=(30,15))
         plt.title('GRU Model 8 Validation Loss vs. Validation Accuracy')
         plt.xlabel('iterations')
         plt.ylabel('validation accuracy')
         #plt.plot(x_axis, y_axis_dummy)
         #plt.plot(x_axis_dummy, y_axis)
         #plt.xticks(x_axis)
         #plt.yticks(y_axis)

         plt.plot(x_axis_8, best_model_val_acc, marker='o', markersize=3, linestyle=linestyle,
         plt.plot(x_axis_8, best_model_val_loss, marker='o', markersize=3, linestyle=linestyle
         #plt.plot(x, y, marker='o', markersize=5, linestyle=linestyle, label=label)

         #plt.axis([min(gru_x_axis),max(gru_x_axis),0,1])
         plt.grid(True)
         plt.legend()
         #plt.savefig('images\gru_plotting _validation _accuracies _against _iterations.png',
         plt.show()
```



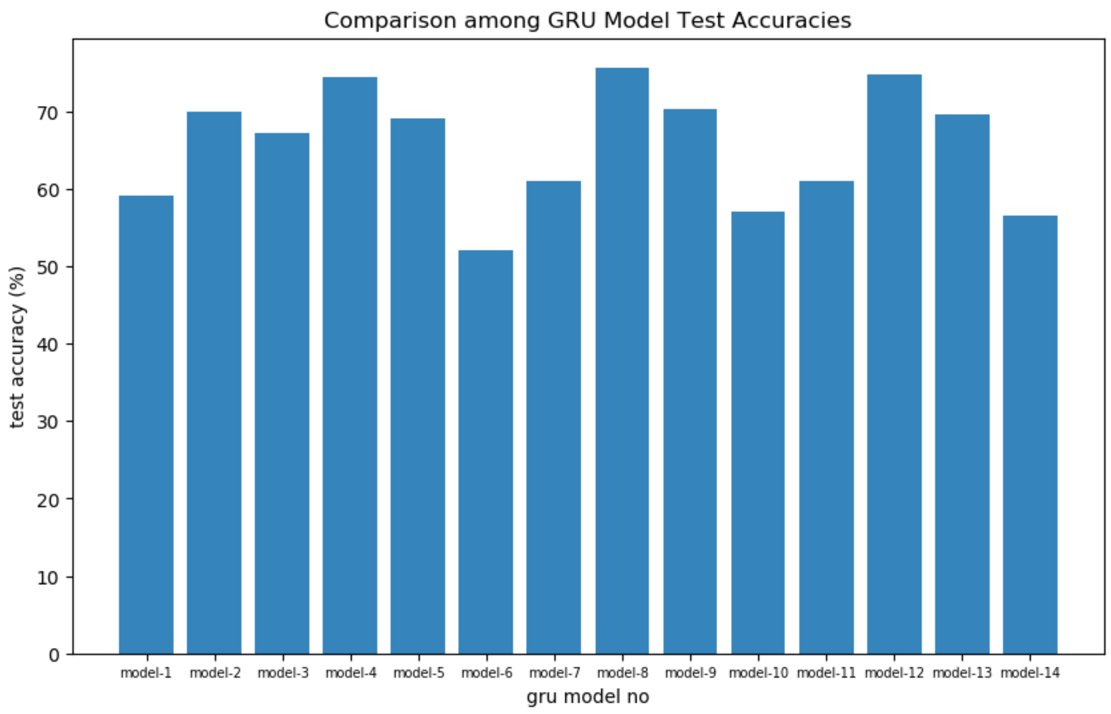
```
In [13]: smoothing_factor = 50
         linestyle='-.'

         #plt.figure(figsize=(30,15))
         plt.title('Validation Loss vs. Validation Accuracy for GRU Model 8')
         plt.xlabel('iterations')
         plt.ylabel('validation accuracy')
         x = x_axis_8

         y = best_model_val_acc
         x_sm = np.array(x)
         y_sm = np.array(y)
         x_smooth = np.linspace(x_sm.min(), x_sm.max(), smoothing_factor)
         y_smooth = spline(x, y, x_smooth)
         plt.plot(x_smooth, y_smooth, marker='o', markersize=3, linestyle=linestyle, label='mod

         y = best_model_val_loss
         x_sm = np.array(x)
         y_sm = np.array(y)
         x_smooth = np.linspace(x_sm.min(), x_sm.max(), smoothing_factor)
         y_smooth = spline(x, y, x_smooth)
         plt.plot(x_smooth, y_smooth, marker='o', markersize=3, linestyle=linestyle, label='mod

         #plt.plot(x, y, marker='o', markersize=5, linestyle=linestyle, label=label)
```

```
        plt.grid(True)
        plt.legend()
        plt.savefig('images\gru_image_2_best_model_validation_loss_vs_validation _accuracy', 
        plt.show()

c:\python36\lib\site-packages\ipykernel_launcher.py:14: DeprecationWarning: `spline` is depreca
spline is deprecated in scipy 0.19.0, use Bspline class instead.

c:\python36\lib\site-packages\ipykernel_launcher.py:21: DeprecationWarning: `spline` is depreca
spline is deprecated in scipy 0.19.0, use Bspline class instead.
```



```
In [ ]:

In [ ]:

In [14]: # comparing gru model test accuracies (bar chart):

        plt.rcdefaults()
        '''
        objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
        y_pos = np.arange(len(objects))
        performance = [10,8,6,4,2,1]
```

```
'''
plt.figure(figsize=(10,6))
objects = []
for i in range(0,len(model_list)):
    object_name = "model-"+str(i+1)
    objects.append(object_name)
y_pos = np.arange(len(objects))
performance = test_acc_list ###

plt.bar(y_pos, performance, align='center', alpha=0.9)
plt.xticks(y_pos, objects)
#plt.tick_params(axis='both', which='major', labelsize=10)
plt.tick_params(axis='x', which='major', labelsize=7)
plt.xlabel('gru model no')
plt.ylabel('test accuracy (%)')
plt.title('Comparison among GRU Model Test Accuracies')
plt.savefig('images\gru_image_3_comparing_gru_model_test_accuracies_bar_chart.png', bl
plt.show()
```



```
In [ ]:
```

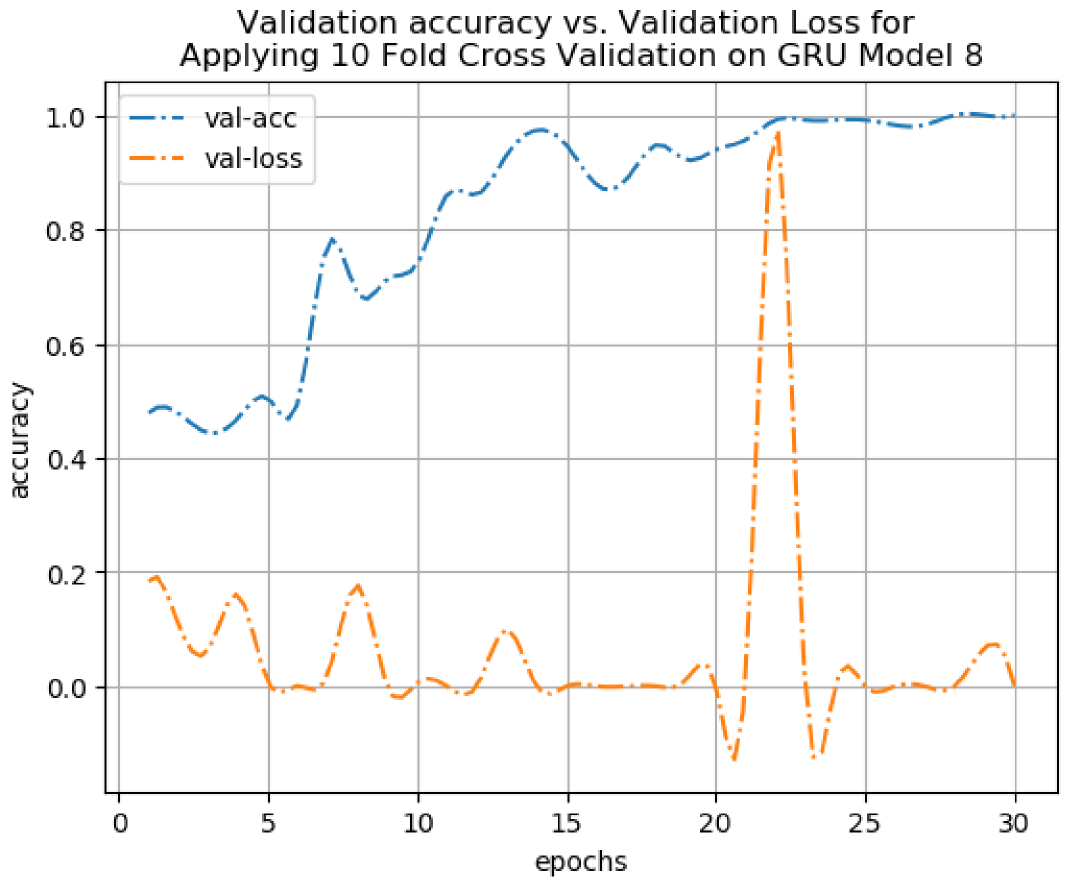
```
In [15]: # gru_10_fold_cross_validation_on_model_8:
         gru_10_fold_cross_val_acc_list = [0.4783,0.4783,0.4435,0.4696,0.5043,0.4957,0.7739,0.6
         gru_10_fold_cross_val_loss_list = [0.1843,0.1090,0.0634,0.1604,0.0079,0.0009,0.0202,0
```

```
        print(len(gru_10_fold_cross_val_acc_list))
        print(len(gru_10_fold_cross_val_loss_list))
```

```
30
30
```

```
In [16]: smoothing_factor = 100
        linestyle='-.'

        #plt.figure(figsize=(30,15))
        plt.title('Validation accuracy vs. Validation Loss for \nApplying 10 Fold Cross Valida
        plt.xlabel('epochs')
        plt.ylabel('accuracy')
        x = [i for i in range (1,len(gru_10_fold_cross_val_acc_list)+1)]

        y = gru_10_fold_cross_val_acc_list
        x_sm = np.array(x)
        y_sm = np.array(y)
        x_smooth = np.linspace(x_sm.min(), x_sm.max(), smoothing_factor)
        y_smooth = spline(x, y, x_smooth)
        plt.plot(x_smooth, y_smooth, marker='', markersize=3, linestyle=linestyle, label='val

        y = gru_10_fold_cross_val_loss_list
        x_sm = np.array(x)
        y_sm = np.array(y)
        x_smooth = np.linspace(x_sm.min(), x_sm.max(), smoothing_factor)
        y_smooth = spline(x, y, x_smooth)
        plt.plot(x_smooth, y_smooth, marker='', markersize=3, linestyle=linestyle, label='val

        plt.grid(True)
        plt.legend()
        plt.savefig('images\gru_image_4_accuracy_vs_loss_for_10_fold_cross_validation.png', bl
        plt.show()
```

```
c:\python36\lib\site-packages\ipykernel_launcher.py:14: DeprecationWarning: `spline` is depreca
spline is deprecated in scipy 0.19.0, use Bspline class instead.

c:\python36\lib\site-packages\ipykernel_launcher.py:21: DeprecationWarning: `spline` is depreca
spline is deprecated in scipy 0.19.0, use Bspline class instead.
```



```
In [17]: # GRU 10 FOLD CROSS VALIDATION MODEL ACCURACY::
         gru_10_fold_cross_val_folds_acc_list = [0.4435,0.4957,0.7130,0.8609,0.9478,0.9478,0.95
         gru_10_fold_cross_val_model_acc = sum(gru_10_fold_cross_val_folds_acc_list)/len(gru_10
         print('GRY 10 FOLD CROSS VALIDATION MODEL ACCURACY =',gru_10_fold_cross_val_model_acc

GRY 10 FOLD CROSS VALIDATION MODEL ACCURACY = 0.8339099999999998


In [ ]:
```