

Deep Learning- CSE641

Assignment - 4

Name: Akanksha Shrimal
Name: Vaibhav Goswami
Name: Shivam Sharma

Roll No: MT20055
Roll No: MT20018
Roll No: MT20121

FILES SUBMITTED : submitted .py file , .ipynb file and readme.pdf and output.pdf

References:

1. <https://github.com/AaronCCWong>Show-Attend-and-Tell>
2. <https://github.com/jiangqn/IAN-pytorch>

UTILITY FUNCTIONS :

1. Save and Load models using Pickle

```
# Saving and Loading models using pickle

def save(filename, obj):
    with open(filename, 'wb') as handle:
        pickle.dump(obj, handle, protocol=pickle.HIGHEST_PROTOCOL)

def load(filename):
    with open(filename, 'rb') as handle:
        return pickle.load(handle)
```

2. Pre Processing data - Normalization and One Hot Encoding

```
# Utility function to normalize the data and one hot encode the
labels

def pre_process_data(train_x, train_y, test_x, test_y):
    # Normalize
    train_x = train_x / 255.
    test_x = test_x / 255.
    enc = OneHotEncoder(sparse=False, categories='auto')
    train_y = enc.fit_transform(train_y.reshape(len(train_y), -1))
    test_y = enc.transform(test_y.reshape(len(test_y), -1))
    return train_x, train_y, test_x, test_y
```

3. Plotting functions

```
# function to plot double line graph
# Plot double line using X1 , Y1 and X2 , Y2
def plot_double_line_graph(X1,Y1,label1 ,X2 ,Y2,label2
, title,y_name):
    fig = plt.figure(figsize=(7,5))
    plt.subplot(111)
    plt.plot(X1,Y1 ,label=label1 ,marker = "x" , color="blue")
    plt.plot(X2, Y2 , label=label2 ,marker = "x" , color="red")
    plt.title(title)
    plt.ylabel(y_name)
```

```

plt.xlabel('Epochs')
plt.legend( loc='upper left',prop={'size': 13})
plt.show()

# Plot single line using X1 , Y1
def plot_single_line_graph(X1,Y1,label1, title,name_y):
    fig = plt.figure(figsize=(7,5))
    plt.subplot(111)
    plt.plot(X1,Y1 ,label=label1 ,marker = "x" , color="blue")
    plt.title(title)

    plt.ylabel(name_y)
    plt.xlabel('Epochs')
    plt.legend( loc='lower right',prop={'size': 13})
    plt.show()

```

4. Plotting ROC Curve

```

# (7,7)
#https://www.dlogy.com/blog/simple-guide-on-how-to-generate-roc-
plot-for-keras-classifier/
def plot_roc(classes, y_test, y_score, figsize=(7,7)):
    n_classes = len(classes)
    # Plot linewidth.
    lw = 2

    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    # Compute micro-average ROC curve and ROC area
    fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(),
y_score.ravel())
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

```

```

# Compute macro-average ROC curve and ROC area

# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in
range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure(1)
plt.figure(figsize=figsize)
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})' +
               '\n'.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})' +
               '\n'.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})' +
                   '\n'.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([0.0, 1.0])

```

```

plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to
multi-class')
plt.legend(loc="lower right")
plt.show()

```

6. Plotting Confusion Matrix

```

def confusion_matrix_find(y, y_hat, nclasses):

    y = y.astype(np.int64)
    y_hat = y_hat.astype(np.int64)

    conf_mat = np.zeros((nclasses, nclasses))

    for i in range(y_hat.shape[0]):
        true, pred = y[i], y_hat[i]
        conf_mat[true, pred] += 1
    return conf_mat

# Plotting confusion matrix
def confusion_matrix_plot(cm, classes, title='Confusion matrix',
cmap=plt.cm.Blues, figsize=(7, 7), path=None, filename=None):

    cm = cm.astype(np.int64)
    plt.figure(figsize=figsize)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.

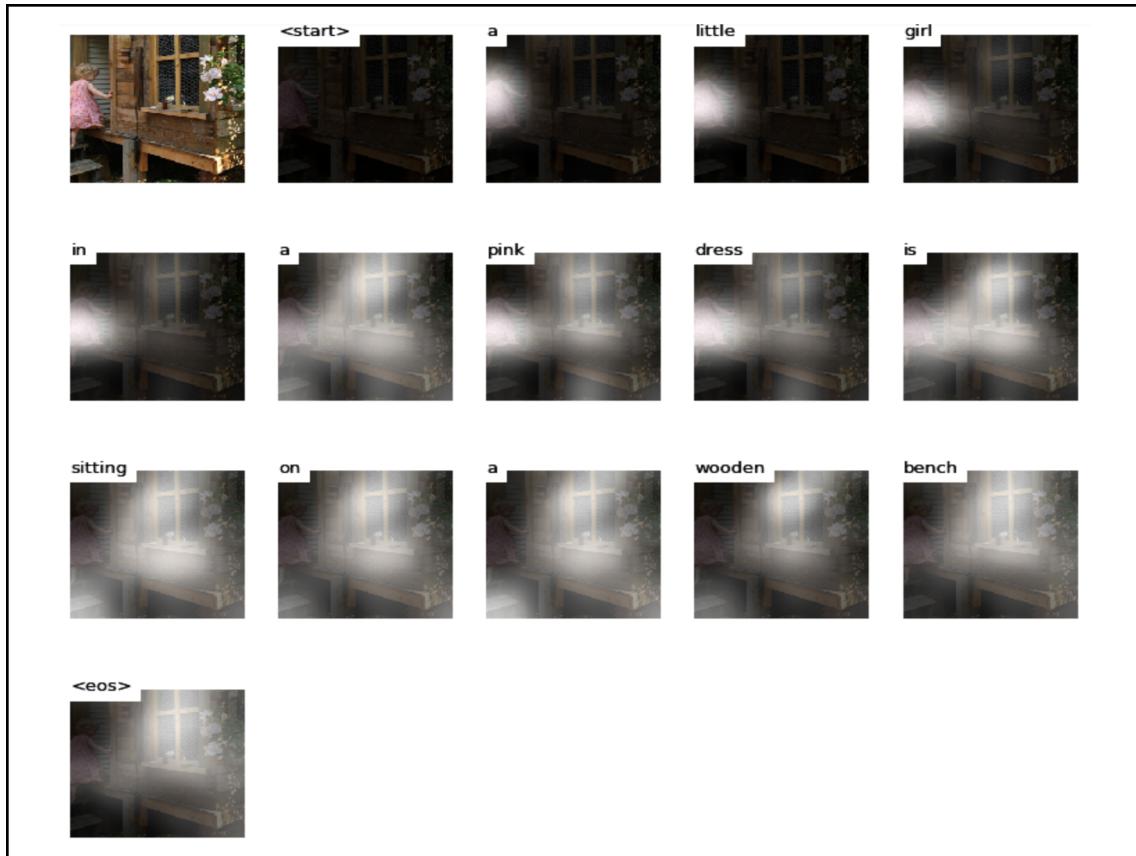
```

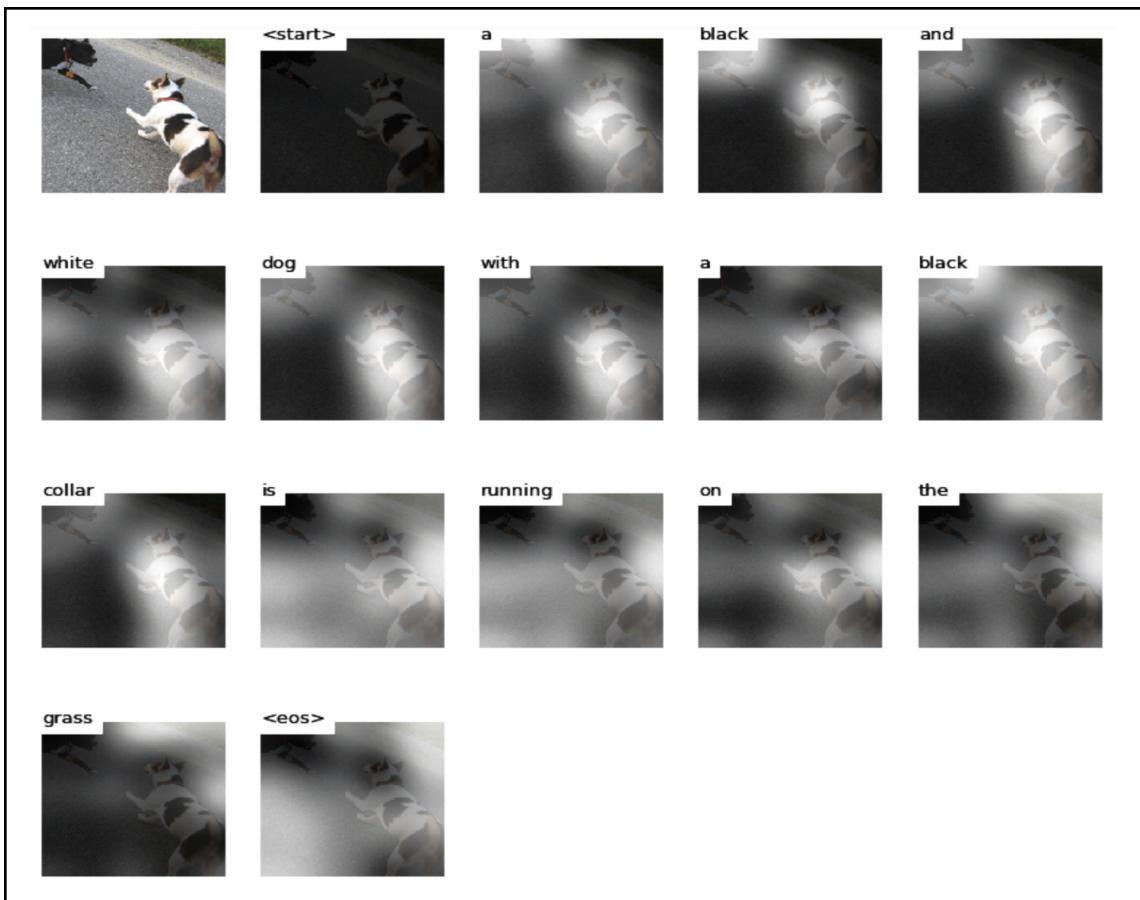
```
for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], fmt),
              horizontalalignment="center",
              color="white" if cm[i, j] > thresh else "black")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.tight_layout()
```

1. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention using soft attention

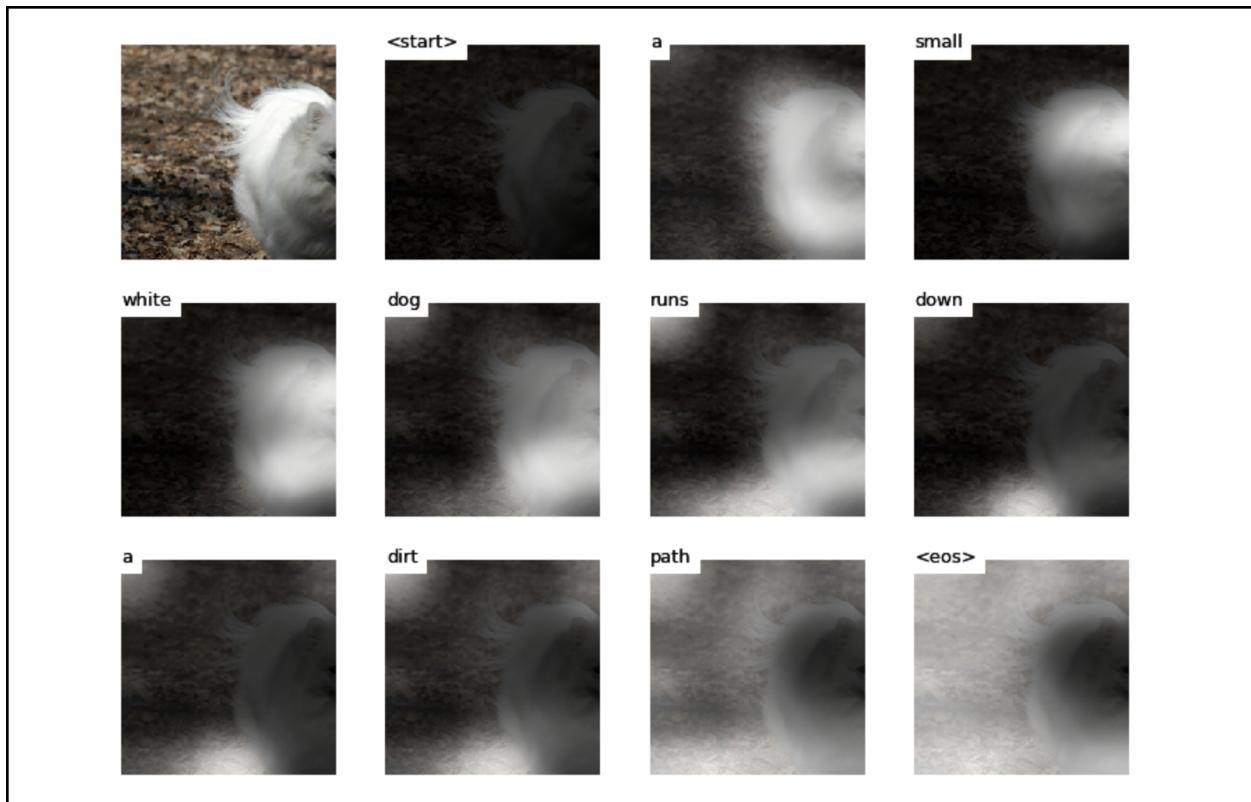
1.8 Attention Weights

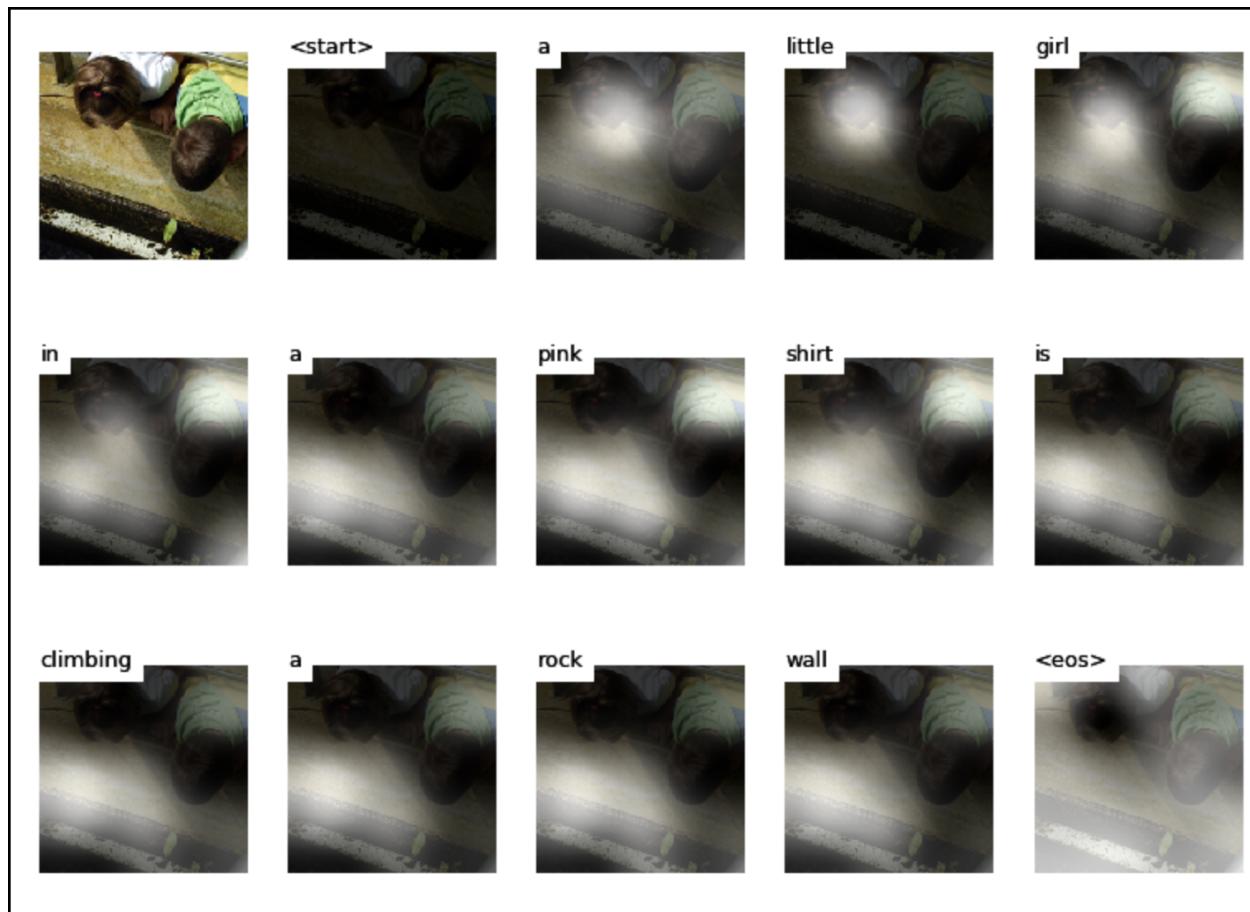
1.8.1 Train Dataset





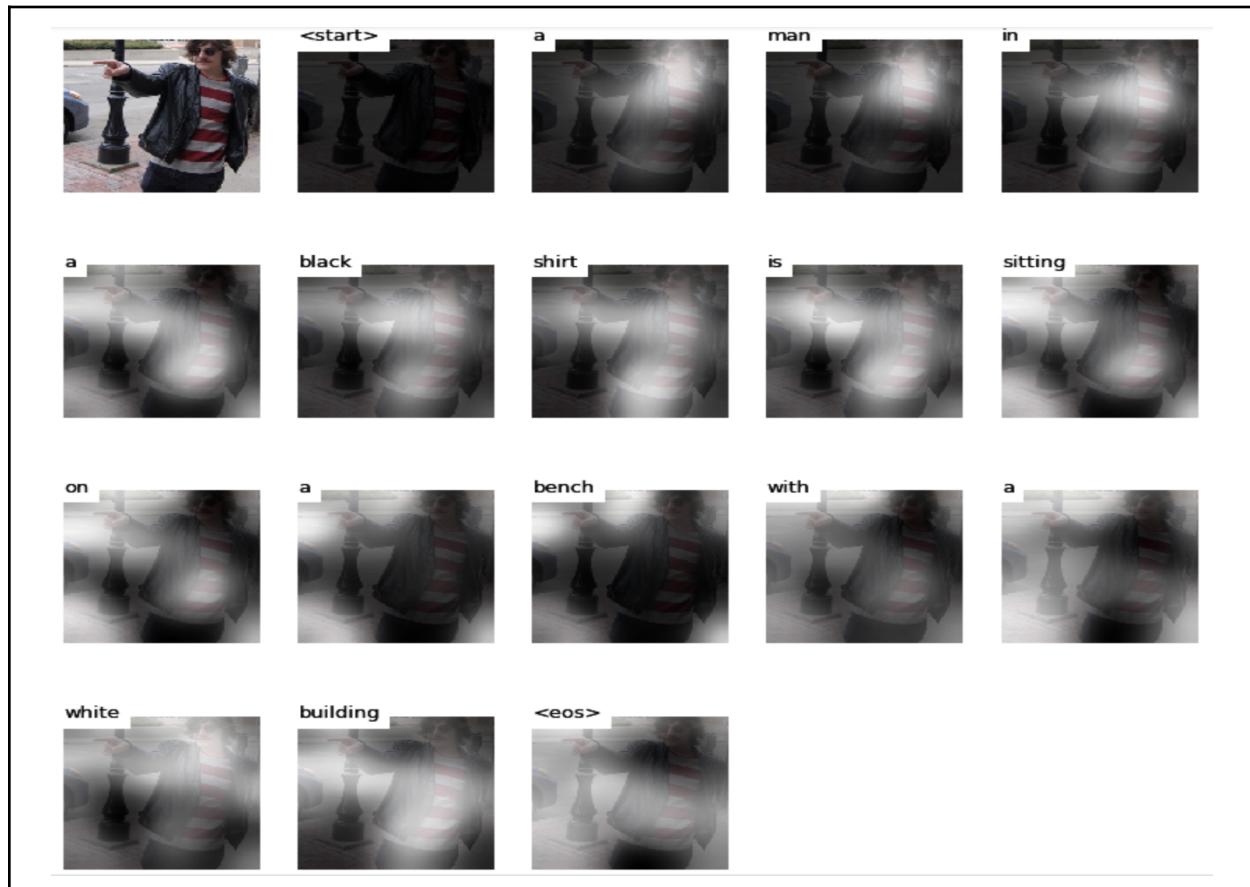
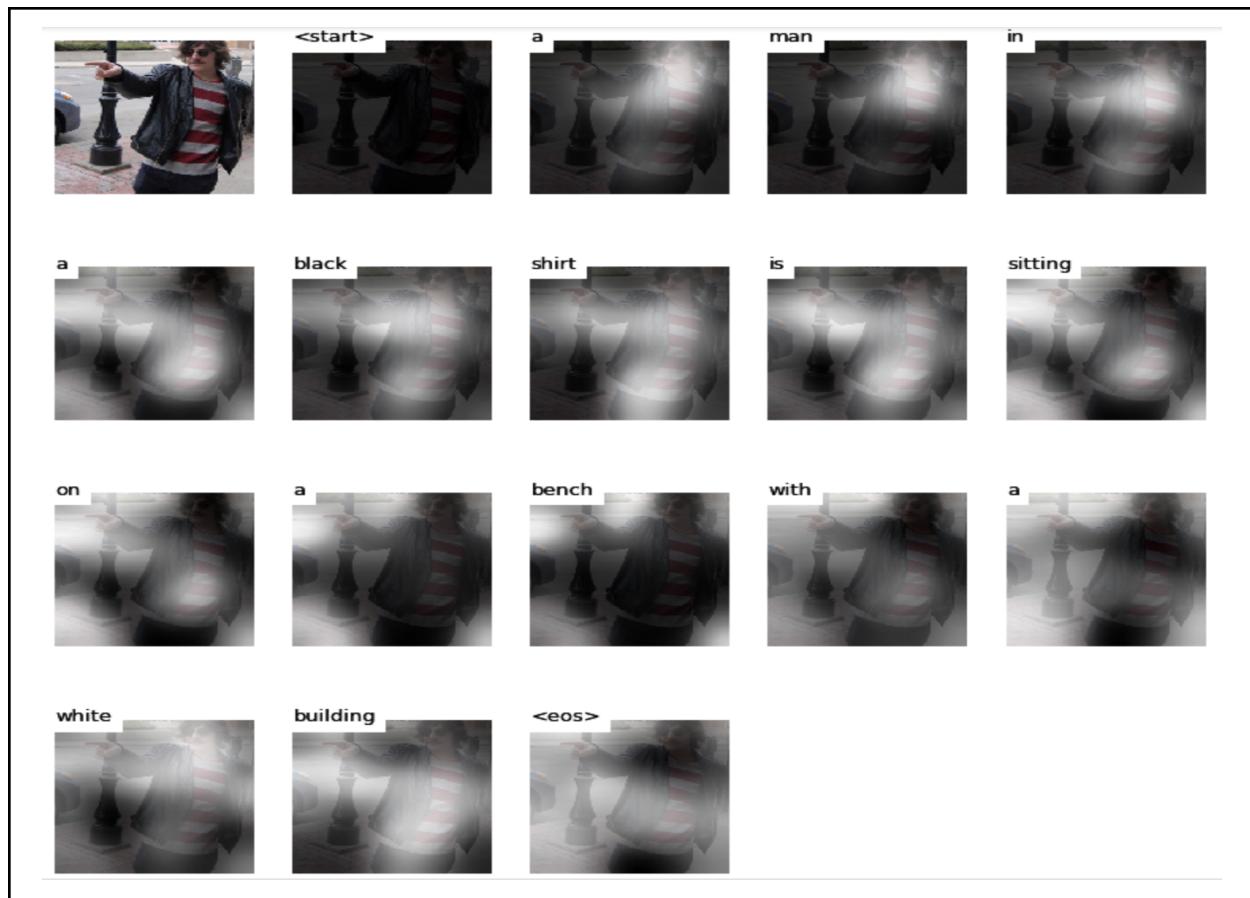
1.8.2 Validation Dataset

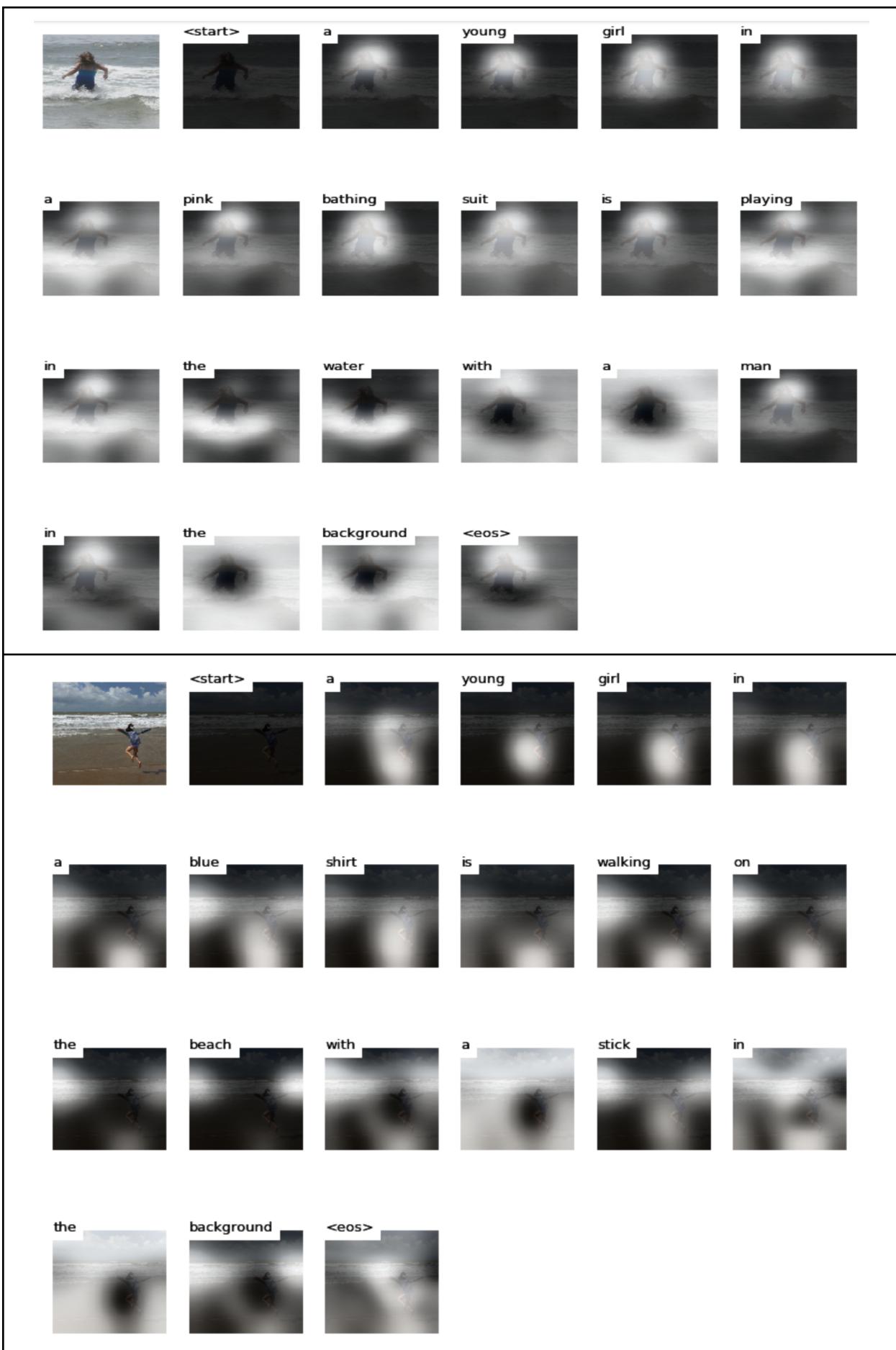




1.8.3 Test Dataset







1.9 Results

1.9.1 Validation Dataset

- Number of epochs - 25
- Validation Loss - 2.8601
- Accuracy:

Metric	Accuracy (%)
BLEU-1	58.2
BLEU-2	37.9
BLEU-3	24.9
BLEU-4	15.9
METEOR	48.2

1.9.2 Test Dataset

- Number of epochs - 25
- Test Loss - 2.5919
- Accuracy:

Metric	Accuracy (%)
BLEU-1	59.6
BLEU-2	38.9
BLEU-3	25.5
BLEU-4	16.3
METEOR	53.1

Contribution of Each Member

1. Shivam Sharma

Implemented (2) and report

2. Akanksha Shrimal

Implemented (1) and report

3. Vaibhav Goswami

Implemented (2) and report.