# Assignment 2 Report: Text Aligned SalGAN

Wu Shuwen(521030910087)

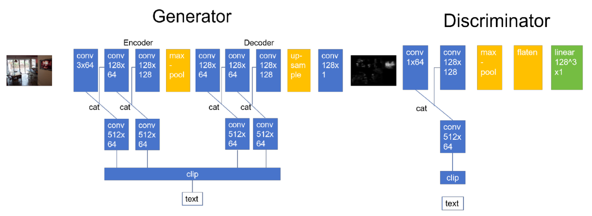

Fig. 1. Structure of my Text Aligned SalGAN

***Abstract*—This report is about my saliency model design based on SalGAN [1] and CLIP. As is shown in the reference paper, SalGAN [1] is the SOTA model on given text-guided saliency dataset, so I chose SalGAN as the base model of my proposed model. For text alignment, I utilized pretrained CLIP model (ViT-B/32 in my experiment) and added it to both the generator and the discriminator.**

## I. Introduction

In my report, I'd like to introduce my idea of my model Text Aligned SalGAN. To make the model more trainable, I simplified original SalGAN. Also, to make a deep alignment between text and saliency, I added text features to multiple layers in both generator and discriminator, both the encoder and the decoder in the generator.

## II. Approach

### *A. Generator*

The generator is basically a classic vgg-like encoder-decoder structure. I simplified the model so that there's only one maxpoolling layer and one up-sampling layer instead of three. For two convolution layers in the encoder and decoder respectively, I concatenated the image features and text features as the input, so that there will be a deep alignment between text and image.

### *B. Discriminator*

I also simplified the original discriminator of SalGAN. As shown in the teaser, there are only two convolution layers, including one for reading the input fake or real image. Also, there's a linear layer for getting the final fake/real score for the input picture. In the discriminator, I also concatenated text and image features, so that the text-image alignment is also checked in the discriminator.

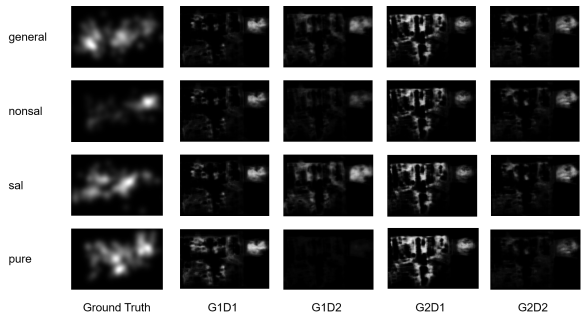

Fig. 2. Output Pictures of My Models

## III. Implementation

I implemented my Text Aligned SalGAN model based on a variety of github repositories. For the basic SalGAN model, I refered to the official implementation in keras and a faulty pytorch implemented version. When I came up with this idea of Text Aligned SalGAN, I first tried to implement a full SalGAN with text features added to it, which however, is too large too train and suffer from gradient vanishing. Then I tried to simplified the original SalGAN and turned it to this version. Also, for data processing, I refered to repository CLIP-SalGAN.

### *A. Generator*

For the generator, I implemented two versions. The Generator in my code is exactly the one shown in the Fig.2. The Generator2 in my code however, doesn't read the text features as input of the encoder. In other words, I'm trying to compare two generators, Generator with text features read in both encoder and decoder, and Generator2 only in decoder.

### *B. Discriminator*

For the discriminator, I also compared two different versions. Similarly, Discriminator in my code is exactly the one in Fig.3 and Discriminator2 in my code is with no text feature input.

## IV. Experiment

I conducted my experiment on my Text Aligned SalGAN using the two generators and the two discriminators mentioned above. In this part, I'd like to refer to the four modules as G1, G2, D1, and D2 respectively. But firstly there's more training details about my experiment.

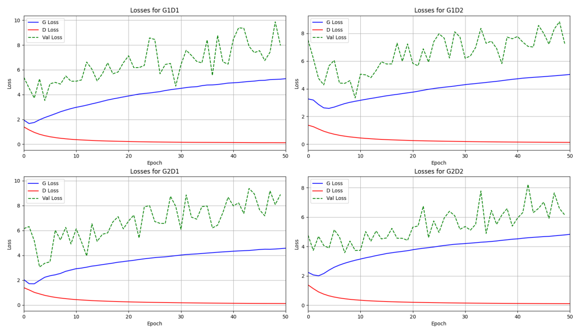

Fig. 3. Loss for my models

### A. *BCELoss*

I used BCELoss to calculate different between generated picture and real picture in my training:

$$L_B = -\frac{1}{N}(\sum Nj = 1(S_j log(\hat{S}_j) + (1 - S_j) log(1 - \hat{S}_j)))$$
.

### B. *Adversarial Loss*

The loss of discriminator:

$$L_D = L(D(I, S), 1) + L(D(I, \hat{S}, 0)$$

The Adversarial Loss:

$$L_A = \alpha L_B + L(D(I, \hat{S}, 1)$$

### C. *Optimizer*

For the optimizer, I utilized the Adam optimizer. Hyper-parameters are as follows:

TABLE I
HYPER PARAMTERS

| **Batch** | **Epochs** | **LR** | **Adam Beta** |
|---|---|---|---|
| 16 | 50 | 0.00002 | (0.5,0.999) |

## V. RESULTS AND ANALYSIS

As is shown in the table, the models have fine performance. In comparison, G1D2, whose discriminator has no text features as inputs and generator has text features inputs in both encoders and decoders performs best.

## REFERENCES

[1] SalGAN: Visual Saliency Prediction with Generative Adversarial Networks CVPR2017
[2] Yinan Sun, Xiongkuo Min, Huiyu Duan, and Guangtao Zhai. How is Visual Attention Influenced by Text Guidance? Database and Model.

TABLE II
RESULTS OF DIFFERENT MODELS ON DIFFERENT DATASETS

| **Model** | **Data** | **AUC** | **sAUC** | **CC** | **NSS** |
|---|---|---|---|---|---|
| G1D1 | total | 0.6155 | 0.5577 | 0.2376 | 0.2115 |
| G1D1 | pure | 0.6326 | 0.5665 | 0.3248 | 0.1545 |
| G1D1 | sal | 0.6292 | 0.5646 | 0.2857 | 0.2224 |
| G1D1 | nonsal | 0.6129 | 0.5563 | 0.2229 | 0.2241 |
| G1D1 | general | 0.6292 | 0.5646 | 0.2857 | 0.2224 |
| | | | | | |
| G1D2 | total | 0.6339 | 0.5670 | 0.3222 | 0.2020 |
| G1D2 | pure | 0.6377 | 0.5689 | 0.3559 | 0.1677 |
| G1D2 | sal | 0.6292 | 0.6413 | 0.3218 | 0.2369 |
| G1D2 | nonsal | 0.6105 | 0.5552 | 0.3415 | 0.2470 |
| G1D2 | general | 0.6466 | 0.5735 | 0.2857 | 0.2143 |
| | | | | | |
| G2D1 | total | 0.6185 | 0.5591 | 0.2354 | 0.1374 |
| G2D1 | pure | 0.6294 | 0.5645 | 0.2816 | 0.1249 |
| G2D1 | sal | 0.6181 | 0.5589 | 0.2297 | 0.1458 |
| G2D1 | nonsal | 0.5963 | 0.5482 | 0.1471 | 0.1652 |
| G2D1 | general | 0.6284 | 0.5639 | 0.2392 | 0.1496 |
| | | | | | |
| G2D2 | total | 0.6218 | 0.5609 | 0.2684 | 0.1584 |
| G2D2 | pure | 0.6168 | 0.5584 | 0.2705 | 0.1224 |
| G2D2 | sal | 0.6327 | 0.5666 | 0.2961 | 0.1878 |
| G2D2 | nonsal | 0.6049 | 0.5523 | 0.2015 | 0.1919 |
| G2D2 | general | 0.6412 | 0.5703 | 0.2992 | 0.1741 |