U^2^2 Net - Project Outline

Team **w or w/o u**

Jiaju Ma (jma14), Carrie Zhuang (yzhuang6), Shuyan Wang (swang75)

Introduction

We are reimplementing the CVPR 2020 paper “U^2-Net: Going Deeper with Nested U-Structure for Salient Object Detection” (paper link). The paper presents a powerful deep learning architecture for salient object detection (SOD), a classic problem in Computer Vision. In plain words, the model proposed by this paper can detect regions within an image that are more attentive than its surrounding areas and separate them via saliency mapping (see examples below). Generating the map is similar to a per-pixel classification problem (whether or not a pixel should be part of the salient object or not) and the model tackles the problem via supervised learning as both images (inputs) and ground truth masks (labels) are fed in as training data. We chose this paper because it is the technology powering the viral AR Cut & Paste demo on Twitter, in which objects in the real world can be “cut ” and pasted without their backgrounds into a desktop software for further editing. This demo is a great example of real-world applications of SOD and Augmented Reality. Moreover, this paper has a simpler model architecture than most other existing saliency detection models but still achieves state-of-the-art performance, allowing us to re-implement and train it from scratch without requiring a fleet of powerful GPUs.

Related Work

The same first author of the paper published another deep learning architecture for saliency detection in CVPR 2019: "BASNet: Boundary-Aware Salient Object Detection" (paper link). The model proposed in that paper consists of two major parts - a Predict Module and a Residual Refinement Module (RRM). The Predict Module has an Encoder-Decoder structure for saliency detection, and the RRM is in charge of saliency map refinement. The model was able to achieve state-of-the-art performances in terms of regional and evaluation boundary measures.

Another paper we are looking at is "End-to-end Animal Image Matting" by Jizhizi et al. (paper link). This paper focuses on salient object detection for animal images. They proposed a new model architecture called the Glance and Focus Matting network (GFM), which has one encoder connected to two separate decoders (one for "glance" and the other one for "focus"). They also contributed a saliency detection dataset named AM-2k, which consists of 2,000 natural animal images and is available upon request.

Public Implementations

- Author's Official GitHub Repo: https://github.com/NathanUA/U-2-Net (Written in PyTorch)

Data

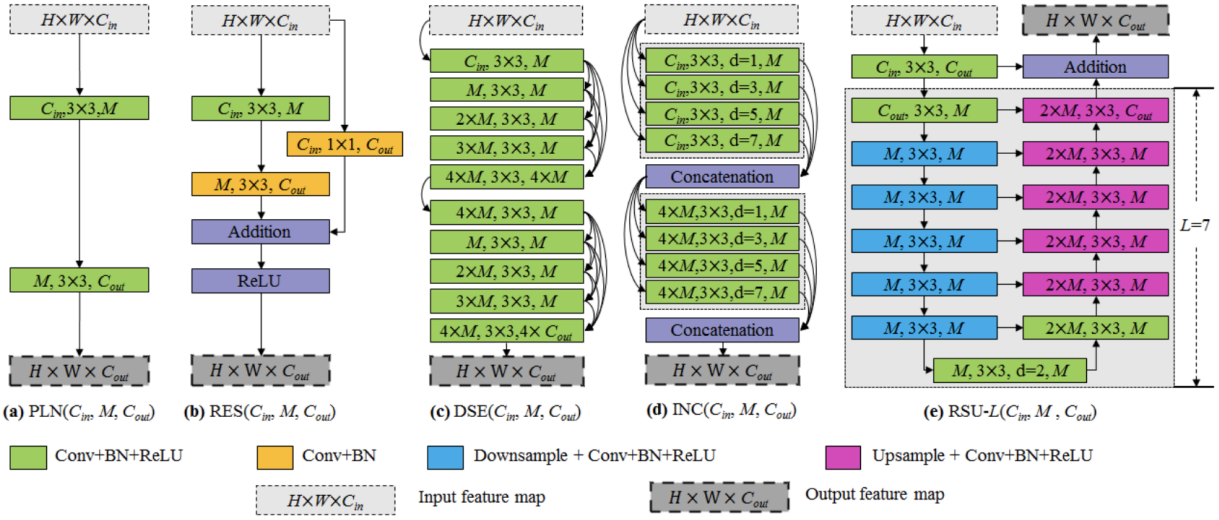
The paper used the DUTS-TR, a part of the DUTS dataset, as training data. Many other saliency detection datasets are being used for evaluation purposes in the paper: DUTOMRON, DUTS-TE, HKU-IS, ECSSD, PASCAL-S, and SOD. These datasets together contain a considerable amount of image data (DUTS itself has 10,553 training images and 5,019 test images). We will not be using the DUTS-TR dataset as the original paper but we have plenty of options to choose from - we might go with another subset in DUTS or try out any dataset used for evaluation in the paper. Moreover, we could also use the AM-2k dataset created by Jizhizi et al. as mentioned in the Related Work section. The only major

preprocessing we need is to crop the images to square and downsample them to smaller sizes (the original paper used 320x320).

Methodology

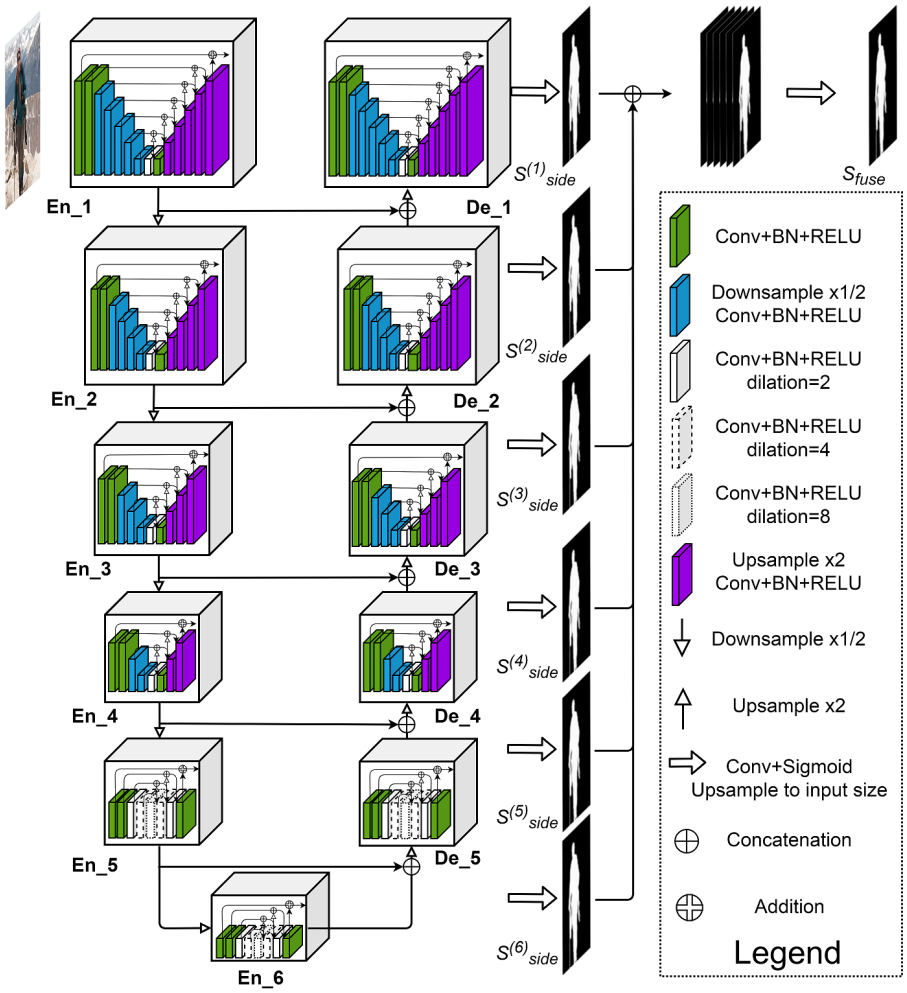
First, U-Net architecture stems from a “fully convolutional network”, and one important modification is that there are a large number of feature channels in the upsampling part, which allow the network to propagate context information to higher resolution layers . The paper proposed a novel residual U-block (RSU) based on U-Net. The motivation of designing such RSU is to achieve more global information at high resolution feature maps from shallow layers. The general structure of RSU proposed in the paper is shown in figure (e) below. It mainly consists of the following three components:

1. An input convolution layer, which transforms the input feature map χ ($H \times W \times C_{in}$) to an intermediate map $F_1(\chi)$ with channel of C_{out}. This is essentially a plain convolutional layer for local feature extraction.
2. A U-Net like symmetric encoder-decoder structure with height of L (L represents the number of layers in the encoder) which takes the intermediate feature map $F_1(\chi)$ as input and learns to extract and encode the multi-scale contextual information $U(F_1(\chi))$. Urepresents the U-Net like structure as shown in figure (e), and larger L leads to deeper RSU.
3. A residual connection which fuses local features and the multi-scale features by the summation: $F_1(\chi) + U(F_1(\chi))$.



The main design difference between RSU and the original residual block is that RSU replaces the plain, single-stream convolution with a U-Net like structure, and replaces the original feature with the local feature transformed by a weight layer, such that the network is empowered to extract features from multiple scales directly from each residual block with smaller computation overhead. Note that in RSU, we are downsampling the vectors for greater efficiency in convolution, and we can safely do so without losing critical information because after convolving, we would concatenate the prior information back to the output vector for proper preservation of information.

For our model, the general idea is to build a U^n-Net by stacking RSUs in a nested structure . It is improved from the frequently explored approach to stack U-Nets sequentially to build cascaded models, where the magnification of computation and the memory costs by n is removed. In our model, we plan to set n to 2 to build a U^2-Net, which is a two-level nested U-structure. The main overall architecture of our model is illustrated in the figure below: it is a U-Net like Encoder-Decoder, where each stage consists of the newly proposed RSUs.



Our U^2-Net will mainly be consisted of three parts:

1. A six stages encoder: In encoder stages En_1, En_2, En_3, En_4, we use RSU-7, RSU-6, RSU-5, and RSU-4 respectively (the “7”, “6”, “5”, “4” denote the heights (L) of RSU blocks). For En_5 and En_6, we use RSU-4F where “F” means that the RSU is a dilated version, in which we replace the pooling and upsampling operations with dilated convolutions. It means all of the intermediate feature maps of RSU-4F have the same resolution with its input feature maps.
2. A five stages decoder: We use similar structures to their symmetrical encoder stages. Note that each decoder stage takes the concatenation of the upsampled feature maps from its previous stage and those from its symmetrical encoder stage as the input.

3. A saliency map fusion module attached with the decoder stages and the last encoder stage: We first generate six side output saliency probability maps $S_{side}^{(6)}, S_{side}^{(5)}, S_{side}^{(4)}, S_{side}^{(3)}, S_{side}^{(2)}, S_{side}^{(1)}$ from stages En_6, De_5, De_4, De_3, De_2, De_1 by a 3x3 convolution layer and a sigmoid function. We then upsample the logits of the side output saliency maps to the input image size, fuse them (concatenate), and apply a 1x1 convolution layer and a sigmoid function to generate the final saliency probability map S_{fuse} .
    a. Upsampling to ensure that all the side output saliency maps are of the input image size
    b. Concatenating along the depth dimension
    c. Applying a 1x1 convolution layer to adjust the depth of the output saliency map after fusing all side saliency maps together
    d. Applying a sigmoid function to map the results to the range [-1,1]
    e. Output image should have 0 at pixels with negative values and 255 at pixels with positive values in output saliency map (thresholding)

For the training process, we use deep supervision and our training loss is defined as:

$$L = \sum_{m=1}^{M} w_{side}^{(m)} l_{side}^{(m)} + w_{fuse} l_{fuse}$$

For each term l, we use the standard binary cross-entropy to calculate the loss:

$$l = \sum_{(r,c)}^{(H,W)} [P_{G\,(r,c)} log P_{S\,(r,c)} + (1 - P_{G\,(r,c)}) log(1 - P_{S\,(r,c)})]$$

where (r, c) is the pixel coordinates and (H, W) is image size: height and width. $P_{G\,(r,c)}$ and $P_{S\,(r,c)}$ denote the pixel values of the ground truth and the predicted saliency probability map, respectively. We will be minimizing the overall loss L during the training process, and during the testing process, we will choose the fusion output l_{fuse} as our final saliency map.

The hardest part about implementing the model introduced in the paper might be realizing the novel RSU block proposed in the paper, which is the fundamental building blocks of our nested U^2-Net model.

Metrics

To understand what metrics can best evaluate the effectiveness of the salient object detection model, we first have to understand what the output of our model is as well as what the “ground truth” looks like.

The outputs of the deep salient object methods are usually probability maps that have the same spatial resolution with the input images. Each pixel of the predicted saliency maps has a value within the range of 0 and 1 (or [0, 255]). The ground truth are usually binary masks, in which each pixel is either 0 or 1 (or 0 or 255) where 0 indicates the background pixels and 1 indicates the foreground salient object pixels.

The traditional “accuracy” notion also applies to our model, just in a slightly different form. The authors of the original U2 net paper proposed six measures to evaluate the quality of the probability maps against the ground truth, including (1) Precision-Recall (PR) curves, (2) maximal F-measure, (3) Mean Absolute Error (MAE), (4) weighted F-measure, (5) structure measure and (6) relaxed F-measure of boundary. In our experiment, we will be mainly using the maximal F-measure and the Mean Absolute Error (MAE) to evaluate the accuracy of our outputs. We chose these two key metrics because they comprehensively evaluate the quality of the results both from a “precision-recall” framework, as well as the traditional error perspective. Furthermore, the paper provided specific data on how well the U2 net is performing on these 2 metrics, so we can leverage that available data to serve as a source of comparison.

1. Maximal F-Measure

In order to measure the accuracy of our model using maximal F-measure, we need to first calculate the precision-recall pairs. Given a predicted saliency probability map, its precision and recall scores are computed by comparing its thresholded binary mask against the ground truth mask. The precision and recall of a dataset are computed by averaging the precision and recall scores of those saliency maps. By varying the thresholds from 0 to 1, we can obtain a set of average precision-recall pairs of the dataset. From there, we can calculate the F-measure of the model, which is defined as:

$$F_\beta = \frac{(1+\beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall}.$$

2. Mean Absolute Error (MAE)

Then, we can also calculate MAE, the Mean Absolute Error which denotes the average per-pixel difference between a predicted saliency map and its ground truth mask. It is defined as the following:

$$MAE = \frac{1}{H \times W} \sum_{r=1}^{H} \sum_{c=1}^{W} |P(r,c) - G(r,c)|$$

where P and G are the probability map of the salient object detection and the corresponding ground truth respectively, (H, W) and (r, c) are the (height, width) and the pixel coordinates.

With these two key metrics determined, we plan to run our model using the data and methodology outlined in the previous sections. Our base goal, which we definitely can achieve by the final due date, is to have a working salient object detection model that follows the U2 net logic. Our target goal is to achieve a similar level of accuracy as defined by the two key metrics above, compared to the original paper. The following chart shows the model accuracy performances for a list of state-of-the-art SOD models, and our target metrics should be close to the last two rows.

Configuration	DUT-OMRON		ECSSD		Time (ms)
	$maxF_\beta$	MAE	$maxF_\beta$	MAE	
Baseline U-Net	0.725	0.082	0.896	0.066	**14**
PLN U-Net	0.782	0.062	0.928	0.043	**16**
RES U-Net	0.781	0.065	0.933	0.042	19
DSE U-Net	0.790	0.067	0.927	0.046	70
INC U-Net	0.777	0.069	0.921	0.047	57
PPM U-Net	0.792	0.062	0.928	0.049	105
Stacked HourglassNet [31]	0.756	0.073	0.905	0.059	103
CU-NET [37]	0.767	0.072	0.913	0.061	50
NIV U^2-Net	0.803	0.061	0.938	0.085	30
U^2-Net w/ VGG-16 backbone	0.808	0.063	0.942	0.038	23
U^2-Net w/ ResNet-50 backbone	0.813	0.058	0.937	0.041	41
(Ours) RSU U^2-Net	**0.823**	**0.054**	**0.951**	**0.033**	33
(Ours†) RSU U^2-Net†	**0.813**	**0.060**	**0.943**	**0.041**	25

If we have time, our stretch goal would be to fine tune the model and achieve a similar level of accuracy performance training and testing with other datasets.

Ethics

What broader societal issues are relevant to your chosen problem space?

Our model will be primarily used for salient object detection. That is, we can use our model to detect and cut out an object from the surrounding background areas on an image. While it can be widely used in the field of augmented reality, such as the “AR cut and paste” demo on Twitter, it might also have unintended negative societal impacts which could potentially lead to serious consequences. If such an SOD model is released online and made easily accessible to the general public, some malicious users might try to use it to synthesize fake photos and conduct fraudulent activities. For example, with the use of the model, a person’s face/body can potentially be cut out from the original photograph and synthesized into a new one in a completely different scene with completely different settings. Then the new synthesized photo can be used to spread rumors, or in fake news to create public tension. Especially with more and more advanced SOD technology, the synthesized photo will not be easily identified as fake and it will gain much more credibility as it is published and seen by the general public, making the issue even harder to solve.

Why is deep learning a good approach to this problem?

First, it is fairly easy to acquire a large amount of image data for the training process. There are already plenty of datasets in the field of deep learning that are specifically designed for salient object detection training. Moreover, as long as the images have objects more attentive than the surrounding areas on the scene, it can potentially be used for training. Given such a simple requirement, it is thus very easy to collect, or even create, data for training the model. In addition, since the image data used for training SOD models normally do not have any people in it, nor does it have any personal information associated with it, there is not much ethical concern related to the collection or the usage of the data. That is, we can in most cases safely use the open-source image dataset online without worrying about or being hindered by the potential ethical implications. Therefore, the requirement of training on enough data can be easily satisfied.

On the other hand, we also need to think about model interpretability - whether we care about understanding why the model makes certain decisions. For SOD models, the answer really depends on the actual application and where we want to use our model. Take the “AR cut and paste” application as an example, as long as we can accurately contour the detected object and smoothly cut out the object from the background scene, it does not really matter if that detection/cutout is explainable. Especially because we are not performing any identification or classification procedures, we do not really need any explanation for the “decision” our model is making as long as a fairly good accuracy can be guaranteed.

With the two conditions satisfied, we can claim that deep learning seems to be a good approach to the problem of salient object detection; hence we could proceed with compiling data and training a neural network model.

Division of labor

Jiaju Ma

- Training the model on GCP or on a local windows machine with GPU
- Fine-tune model and experiment with different hyperparameters

Carrie Zhuang

- Implementing the model in Tensorflow
- Debugging and verifying the correctness of our implementation

Shuyan Wang

- Prepare training, testing, and evaluations datasets
- Evaluate model performance by calculating metrics