WHERE’S YOUR FOCUS: PERSONALIZED ATTENTION

by

SIKUN LIN

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science and Engineering

June 2017, Hong Kong

Copyright © by Sikun Lin 2017
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

______________________________
SIKUN LIN
WHERE’S YOUR FOCUS: PERSONALIZED ATTENTION

by

SIKUN LIN

This is to certify that I have examined the above M.Phil. thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

__________________________
DR. PAN HUI, THESIS SUPERVISOR

__________________________
PROF. HUAMIN QU, DEPARTMENT HEAD

Department of Computer Science and Engineering

29 June 2017
ACKNOWLEDGMENTS

I would never have completed this work without the help from many people.

First of all, I thank my advisor, Dr. Pan Hui, for his years of mentoring, advice, and encouragement. I have learned from him how to develop, evaluate, express, and defend my ideas. These skills are important for my research in the future.

I thank my colleagues of HKUST-DT System and Media Lab. In daily life, we have been good friends. Without them, my graduate study at HKUST would not be so colorful.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Page</td>
<td>i</td>
</tr>
<tr>
<td>Authorization Page</td>
<td>ii</td>
</tr>
<tr>
<td>Signature Page</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>Abstract</td>
<td>ix</td>
</tr>
<tr>
<td><strong>Chapter 1</strong> Introduction</td>
<td>1</td>
</tr>
<tr>
<td><strong>Chapter 2</strong> Background and Related Works</td>
<td>4</td>
</tr>
<tr>
<td>2.1 DNNs for object detection</td>
<td>4</td>
</tr>
<tr>
<td>2.2 DNNs for Saliency Prediction</td>
<td>6</td>
</tr>
<tr>
<td><strong>Chapter 3</strong> Design</td>
<td>8</td>
</tr>
<tr>
<td>3.1 Preference Identification</td>
<td>8</td>
</tr>
<tr>
<td>3.2 Model Architecture</td>
<td>10</td>
</tr>
<tr>
<td>3.2.1 Saliency prediction part</td>
<td>11</td>
</tr>
<tr>
<td>3.2.2 Detection part</td>
<td>11</td>
</tr>
<tr>
<td>3.2.3 Non-Maximum Suppression (NMS) operation</td>
<td>12</td>
</tr>
<tr>
<td>3.2.4 Mapping operation</td>
<td>14</td>
</tr>
<tr>
<td>3.2.5 Merging two streams</td>
<td>14</td>
</tr>
</tbody>
</table>
Chapter 4 Implementation

4.1 Data Collection
4.2 Data Generation
4.3 Data Augmentation
4.4 Training
  4.4.1 Pretraining: Training a SSD model
  4.4.2 Pretraining: Training saliency layers
  4.4.3 Training a PANet

Chapter 5 Evaluation

5.1 Quantitative Results
  5.1.1 Datasets
  5.1.2 Metrics
  5.1.3 SALICON Results
  5.1.4 PASCAL-S Results
5.2 Qualitative Results
5.3 Same model, different preference
5.4 Comparison with unbiased models

Chapter 6 Discussion and Future Work

Chapter 7 Conclusion

References
## LIST OF FIGURES

2.1 Grid cell segmentation in detection networks.  

3.1 PANet architecture. The model takes three inputs: original image, personal preference vector $pvec$, and detailed class to super category mapping. The upper stream is responsible for saliency detection, and the lower stream is responsible for fitting user preference. Final output is personalized saliency prediction of the input image.  

3.2 SSD Architecture  

3.3 NMS Operation Layer  

3.4 Mapping Operation Layer  

4.1 Matlab GUI for collecting user labels.  

4.2 Choosing weights for ground truth generation.  

4.3 COCO image, labeled ground truth by subject A, our generated ground truth.  

4.4 COCO image, saliency ground truth, generated personal saliency ground truth with preference vector:  

```plaintext  
[ outdoor, food, indoor, appliance, sports, person, animal, vehicle, furniture, accessory, electronic, kitchen ] = [0.833, 0.346, 0.189, 0.098, 0.934, 0.679, 0.481, 0.875, 0.081, 0.579, 0.901, 0.223].  
```

4.5 Model architecture for pretraining layers responsible for saliency prediction. Weights of layers in red will be used in the next phase.  

4.6 COCO image, model predictions with preference vector:  

```plaintext  
[pvec1 = [ outdoor, food, indoor, appliance, sports, person, animal, vehicle, furniture, accessory, electronic, kitchen ] = [0.833, 0.346, 0.189, 0.098, 0.934, 0.679, 0.481, 0.875, 0.081, 0.579, 0.901, 0.223] and pvec2 = [person, non-human] = [1.0, 0.05].  
```

4.7 Effect of fine-tuning. Images are: COCO image, prediction for $pvec_1$ with original model, prediction for $pvec_2$ with original model, prediction for $pvec_2$ with fine-tuned model.  

4.8 Performance of original model and fine-tuned model.
LIST OF TABLES

3.1 Default mapping from detailed classes to super categories in MS COCO. 9
4.1 Subject preference vectors in our collected dataset. 16
5.1 Pascal mapping and preference setup in our experiment. 27
5.2 Average scores. 27
5.3 Performance of original model and fine-tuned model on different preference vectors. 32
5.4 Salicon model tested on the ground truth generated with \( pvec_1 \) and \( pvec_2 \). 33
WHERE’S YOUR FOCUS: PERSONALIZED ATTENTION

by

SIKUN LIN

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Human visual attention is subjective and biased according to the personal preference of the viewer, however, current works of saliency detection are general and objective, without counting the factor of the observer. This will make the attention prediction for a particular person not accurate enough. In this work, we propose PANet, a convolutional network that predicts saliency in images with personal preference. The model consists of two streams which share common feature extraction layers, and one stream is responsible for saliency prediction, while the other is adapted from the detection model and used to fit user preference. Experimental results on augmented PASCAL-S and SALICON dataset confirm that PANet can predict saliency areas according to input preference vectors. Compared with other general saliency prediction models, the model with ability of fitting user preference will provide more benefits to either augmented reality (AR) or recommendation applications.
CHAPTER 1

INTRODUCTION

Attention is a very personalized experience, different people may focus on totally different regions even they are facing at a same scene: imagine a view in a park containing both entertainment equipment and children playing around; in the view of parents, the attention will be focused on children, whereas in the view of children, the attention will be on the equipment. Predicting correctly where the attention is for each user is crucial for a Human-Computer-Interaction (HCI) application. This is the primary motivation of our work. With recent advances in deep learning and the improvement of computation powers, vision tasks such as object detection and classification, saliency prediction are getting higher and higher accuracy, and at the same time being executed at a faster speed. It is feasible now to create a model for predicting user attention, which is not objective but has the ability to fit user preferences differently. Preference here means the different tendencies to focus on various objects, and is inherently consistent for one particular user.

Such a model is useful in various situations: it can be a part of an augmented reality (AR) recommendation system that retrieves and displays only information that the user is in favour of, focusing the post processing only on useful regions instead of the whole image. In [38], egocentric videos can be summarized to create a diary-like record for daily life, with personalized attention model, customized scene description or video summarization for different individuals can be achieved. The model can also be used in the completely opposite way: Autism patients tend to focus only on a particular set of things around them, which is the main reason that makes them disconnect with the outside world. The model may help in providing information that they usually do not notice, thus offering the chance to know different aspects of the world.

Recent development of HCI applications increase the demand of providing different user experience to fit individual needs, and for AR applications, processing input images differently is the first step to achieve this. However, the state-of-the-art saliency prediction works are all trained and evaluated on objective saliency datasets [24, 5, 10], in which the saliency labels are collected through crowd sourcing and averaging. There exists the need to build a model that can predict the attention for a particular user with his/her own preference. To fulfill this need, we propose in this
work, Personalized Attention Network (PANet), a deep architecture that combines object detection and saliency prediction techniques to predict personalized saliency areas.

PANet consists of two streams of convolutional neural networks (CNNs), which share common feature extraction layers. The model takes three inputs: raw image to be processed, user-defined detailed class to super category mapping, and user preference vector on super categories (detailed explanations will be presented in Chapter 3.1). Given the input image, PANet will extract its deep features at multiple scales and pass them to two streams: the saliency prediction stream will generate a saliency map without the influence of user preference, and the preference fitting stream will utilize object detection model architecture, in our case the SSD model [36], to generate a preference map according to the input preference. After combining the results gotten from two streams, post-processing including adding a central prior will be done and the prediction result will be given as a pixel-level saliency map that fits this particular user. To train a PANet model, we directly minimize the objective function that represent pixel-wise divergence between the ground truth personalized saliency map and the one predicted by our model, both in the form of probability distribution. Therefore, ground truth saliency maps are needed for training and validation purpose, which should be paired with a particular preference vector: different input preferences will result in separate sets of ground truth maps. We solve this by defining the ground truth dynamically in the training generator.

In summary, the key contributions of our work are:

- We present the novel idea of personalized attention prediction and discuss its importance over existing generalized attention models. To accomplish this task, we develop PANet, a deep learning model utilizing both object detection and saliency prediction techniques to predict where is the user’s attention according to the collected preference.

- We propose an approach for automatically collecting personal preference from user’s album. This approach leaves the freedom for users to define how many categories his/her preference is divided into, and what these categories are.

- We dynamically generate ground truth preference-influenced saliency maps for training the model, which are built upon existing object detection dataset and saliency detection dateset. The parameters for ground truth generation are based upon true collected labels.
We evaluate the model with saliency prediction metrics and test the trained model on different preference vectors. The results have shown that our system is efficient to use for different preferences with little fine-tuning.

The rest of the thesis is organized as follows. The related work is introduced in Chapter 2. The model design, implementation and evaluation are presented in Chapter 3, 4 and 5 respectively. After that, potential improvements and future work are discussed in Chapter 6 before concluding the thesis in Chapter 7.
CHAPTER 2

BACKGROUND AND RELATED WORKS

PANet requires techniques of both object detection and saliency prediction. In general, it is a saliency prediction model, but it also requires object detection technique to get the knowledge of object categories and positions, so as to make the model have the ability to fit personal preference. In this chapter, we first review recent works in object detection, and then discuss the techniques for saliency prediction, both in virtue of the advance in deep neural networks (DNNs).

2.1 DNNs for object detection

Object detection is a core computer vision problem and has many previous works covered. Traditional detection pipeline typically consists of multiple steps: feature detection and descriptor extraction, optional feature representation (Bag-of-Visual-Words, Fisher Vector), and classifiers applied in a sliding window manner (e.g. Deformable Part Model) or in selected image sub-regions (e.g. Selective Search region proposal). Here we mainly focus on the works that utilize recent advance in CNNs.

After CNN gains its popularity, tried to solve detection problem in a regression manner, however the performance was not satisfactory: only 30.5% mAP on PASCAL VOC2007. As CNN has good performance in classification tasks, R-CNN combines region proposal and CNN-based classification, becoming the pioneering work in this track. Its good performance also made region proposal deep detection systems popular and several works later tried to improve R-CNN in terms of prediction accuracy and speed. Original R-CNN requires feature extraction and classification of many image sub-regions that are overlapped with each other, thus containing many repeated calculation. SPP-net solves this issue by putting the feature extraction convolutional layers at the beginning and introducing the spatial-pyramid pooling layer, making later classification layers share the calculated features. This one-time calculation speeds up the original version notably. Fast R-CNN borrows the idea from SPP-net and refines the ROI pooling layer that
generates potential bounding boxes, further making the whole model end-to-end trainable. In addition, the multi-task loss that firstly introduced in the Multibox model \cite{12} is used here, adding bounding box regression directly into the network. However, selective search based region proposal is still not fast enough for real-time predictions. Faster R-CNN \cite{46} tries to tackle this bottleneck by proposing Region Proposal Network (RPN), letting the network itself learn to generate region proposals. It then combines RPN into the overall network, making RPN share the computation with fast R-CNN convolutional layers.

To achieve even faster implementations, some works omit region proposals completely and replace them with default grid cells and anchor boxes, modeling the detection as regression for each cell or anchor box. OverFeat \cite{51} implements an efficient deep version of sliding window and predicts object location information for each window. But the lack of global context information makes it require other pipeline components to fulfill a reasonable detection task. Some more recent works do not fit into larger pipelines that have several components, with each needs its own optimization. Instead, they are complete systems that can simultaneously learn multiple targets: object class, bounding box position, and prediction confidence. This is also where faster R-CNN and the following models separate: although all of them generate a set of bounding boxes with network layers, region information and class information in faster R-CNN are not simultaneously dealt with. YOLO \cite{44} segments input image into $7 \times 7$ grid cells as in Figure 2.1a, and each cell is responsible for predicting 2 bounding boxes and one object class. Although this results in a faster speed, at the same time it imposes the limitation to detect small or overlapped objects. SSD \cite{36} performs better as it generates more default anchor boxes in different aspect ratios for each cell and generates at different feature map scales (Figure 2.1b). Class prediction is not bound with cells but with anchor boxes, making SSD more flexible in predicting multiple objects within
a small sub-region. It also replaces fully connected prediction layers in YOLO with convolutional ones, further increasing the speed. To solve the detection problem with small objects, YOLO v2 [45] fine-tunes the classifier network with higher input resolution, making high-resolution detection part adjust better. This improved YOLO version also removes the fully connected layers for bounding box prediction and use the idea of anchor boxes as raised in faster R-CNN and used in SSD. Its customized network structure, learnt bounding box dimensions through clustering, and carefully designed training techniques altogether make a good end performance.

To fit the purpose of both object detection and saliency prediction, and have an acceptable prediction speed, our work uses the network structure of SSD, which serves as the detection stream in our whole network, as well as the provider of feature maps at different scales for the saliency prediction stream.

2.2 DNNs for Saliency Prediction

Saliency prediction can be categorized into three general categories: bottom-up approaches based on low-level features such as color, contrastness, orientation, texture, etc. [22, 39, 20, 15, 1, 27, 35, 42, 63, 64, 25, 65, 10]; top-down approaches based on high-level image features, typically incorporating object knowledge [26, 8, 18, 52, 23]; and the combination of the two [41, 62, 9]. Development in deep learning also boosts the performance of saliency prediction. Meanwhile, saliency datasets used for training and benchmark (SALICON [24], MIT300 [5], etc.) make important contributions, providing enough data to train new deep models.

Recent works using deep networks in saliency detection obtain good performance, as the networks can extract more robust features than handcrafted low-level features. [21, 34] convert the input image into different resolutions by down-sampling, and feed them into multiple CNN streams to extract multi-scale features, and then concatenate the features together, passing the combined feature to post-processing layers to get the final saliency map. [32] combines deep features and handcrafted low-level features to increase prediction accuracy. It also uses three deep CNN streams for multi-scale feature extraction, but unlike previous works that directly pass the complete image, it segments the image into non-overlapping regions, and for each region, extracts features from three nested bounding boxes. However, this leads to many repeated calculation during the feature extraction stage. [28] directly uses truncated VGG-16 [54] layers for feature extraction, and add
Inception modules to capture the multi-scale information. [29, 31, 30] use only one CNN stream, but get features in different scales through different layers, and rescale all these feature maps to a same size for further feature merging and saliency map generation. For the saliency detection part in our work, we use the deep features extracted from VGG-16 trained on ImageNet [49]. Meanwhile, the feature maps at different scales extracted from detection layers are merged in saliency prediction stream for further processing.

Including saliency prior is also an noticeable factor that improves the final estimation. Several works [26, 35, 30] include center prior as human generally focus their attention on the center of their eyesights. The priors are added either as a Gaussian function of the pixel distance to the center, or gotten from labeled saliency datasets. Adding center prior may have a negative influence on the result depending on the task, such as predicting eye fixations on a webpage, where human may tend to look for information on the edge menus. Our work considers only general scenes, thus center prior is added and can make positive contribution to the final result.

Although the state-of-the-art saliency prediction models can perform rather well in terms of evaluation metrics (F-score, NSS, sAUC, etc.), they are all measured on benchmark datasets considering no personal preference factor. On the other hand, a subjective version of saliency prediction is needed in the real world for HCI applications. Therefore, in our work, we combine object detection and saliency prediction techniques to build a model for predicting personalized attention. The detailed design will be discussed in Chapter 3.
CHAPTER 3

DESIGN

In this chapter, we will first propose methods to collect the user preference, which serves as an extra input to the model. After that, we will give an overview of our designed model PANet, as well as introducing each part of it in more details.

3.1 Preference Identification

Objects can be categorized into both detailed class and super categories: for example, an apple has a detailed class name “apple”, and also belongs to the super category “fruit”. We train the SSD model on detailed classes, but build user preference vector upon super categories, and this leaves the freedom of defining super categories by users.

Our system gives users the opportunity to define their own mappings between detailed classes and super categories. Table 3.1 lists the default mapping used as category labels in MS COCO. However, this mapping is not perfect in several aspects: some super categories contain classes that are highly likely to co-occur, such as objects belong to “appliance” and objects belong to “kitchen”, and their preference scores prone to be similar to each other for a fixed user. A more meaningful mapping should merge the super categories that tend to occur in a same environment or have a similar usage. In addition, this default mapping may not fit user needs well, one user may want a mapping points to only two super categories: “out-door” and “in-door”, whereas another user that pay more attention to people rather than objects may want “human” and “non-human” categories to represent his preference. These limitations make a general default mapping not suitable for our personalized attention system, so we ask users to define the mapping by themselves and provide them with detailed class names. The operation of mapping definition is done only once. Users can decide how many super categories they want to have, and which detailed classes belongs to those categories correspondingly, so as to describe their preferences in the best way. For implementations in Chapter 4 we use the default mapping in MS COCO. However, we have further experiments on another different mapping in Chapter 5.3.
| traffic light, fire hydrant, stop sign, parking meter, bench | outdoor  |
| banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake | food     |
| book, clock, vase, scissors, teddy bear, hair drier, toothbrush | indoor   |
| microwave, oven, toaster, sink, refrigerator | appliance |
| frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket | sports   |
| person | person |
| bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe | animal   |
| bicycle, car, motorcycle, airplane, bus, train, truck, boat | vehicle  |
| chair, couch, potted plant, bed, dining table, toilet | furniture |
| backpack, umbrella, handbag, tie, suitcase | accessory |
| tv, laptop, mouse, remote, keyboard, cell phone | electronic |
| bottle, wine glass, cup, fork, knife, spoon, bowl | kitchen  |

Table 3.1: Default mapping from detailed classes to super categories in MS COCO.

There are several ways of collecting a user’s preference. We first describe our proposed method, and then compare with other possible approaches. To collect the preference of a particular user, we pass images in the user’s smart device into a trained detection model, which in our case is the SSD model that we are going to integrate in our PANet. As user’s preference may change over time, we only select the image files that have last modified dates within three months, so that the preference vector we get can reflect the user’s current state. Output of SSD model contains object categories, corresponding prediction confidence, and object locations, and here we use the former two to define user preference. By counting the occurrence of objects falling into different categories throughout all the images we looped over, and taking into account the prediction confidence, we can roughly know this user’s preference: the preference towards a particular super category $SCat_i$ is defined as:

$$Pref_i = \sum_{x \in SCat_i} Conf_x,$$

in which $x$ is the object belongs to this category, and $Conf_x$ is the confidence score of that prediction. In this manner, more occurrence of the objects in category $SCat_i$ will lead to a higher preference of it. Meanwhile, adding up the confidence score instead of counting only occurrence number will make sure detections with low confidence can not influence the preference much. After iterating over all the images, the preference vector can be obtained by normalizing the preference values to between 0 and 1:

$$Pref_i = \frac{Pref_i}{\max_j Pref_j}.$$
then we can get the final preference vector of the user: \( pvec = [Pref_1, ..., Pref_n] \), where \( n \) is the total number of super categories.

We considered several other approaches for preference collection. The first one is taking videos with wearable glasses, and analyzing the object occurrence time for different categories. This will results in more accurate preference if we can collect several days of videos. However, wearable glasses have battery life lower than one hour if the camera is in use, which makes this method extremely impractical. A similar method is using Spectacles (snap glass) or GoPro to take pieces of short videos, but this requires too much user control. Rating the preference towards super categories by user themselves is also a choice, but this will lead to discontinuous rating (integer rating from 1 to 10). Our proposed preference identification approach requires the least user effort, but may suffer from accuracy problem when trained detection model covers not enough categories and cannot detect all daily objects. Thus we offer users the chance to rate preference by themselves: we first calculate \( pvec \) through above mentioned approach, and present it to the user. If he thinks this \( pvec \) cannot truly reflect his preference, he can then rate each super category respectively.

### 3.2 Model Architecture

![PANet architecture](image)

Figure 3.1: PANet architecture. The model takes three inputs: original image, personal preference vector \( pvec \), and detailed class to super category mapping. The upper stream is responsible for saliency detection, and the lower stream is responsible for fitting user preference. Final output is personalized saliency prediction of the input image.

As shown in Figure 3.1 our model architecture contains two basic streams: the upper one is for predicting the general saliency information, and the lower one is for fitting user preference upon the information gotten from detection layers. Image features are extracted by shared VGG-16 layers.
without its final classification layers, as well as the convolutional layers in the detection part. This shared feature extraction will make the model prediction run faster than directly combining the results getting from two separate detection and saliency prediction models.

### 3.2.1 Saliency prediction part

Saliency prediction part is the upper stream in Figure 3.1. In order to combine multi-scale features of the input image for saliency prediction, but without an extra feature extraction stream as in the [21, 32], our model uses the features extracted in different layers from VGG-net and SSD customized layers. Features from three different scales are chosen: in our implementation are conv4_3, conv5_3, conv6_2 (layer names are consistent with those in [50]), with size $38 \times 38$, $19 \times 19$, $10 \times 10$ respectively. The features extracted from the latter two are upsampled to the same size as the one extracted from the first layer. Combining features from multiple layers can increase saliency prediction accuracy compared with using a single layer feature. Although [21] uses features only from two scales and claims that adding more layers will not further improve the result, we found in our implementation that adding the features extracted from the third scale still can further minimize training loss and improve saliency prediction accuracy.

After rescaling, the feature maps are combined as a 3-dimensional tensor, with size $38 \times 38 \times 3$ and 512 channels. This combined tensor is then passed through four 3-dimensional convolutional layers with kernel size 1, with 64, 128, 4, 1 feature channels respectively. Then the network re-shapes the tensor back to 2 dimension with size $38 \times 38$ and 3 channels, and passes it through one more $1 \times 1$ convolutional layer, from which the network outputs the final result with a single feature channel for the saliency detection stream.

### 3.2.2 Detection part

Detection part contains truncated VGG16 layers and the layers labeled as SSD ADD-ONs in Figure 3.1, where the latter are the same as those customized detection layers in SSD model shown in Figure 3.2. The add-on part contains its featured anchor box generation layers and produces feature maps at multiple scales, two of which are shared with saliency prediction stream as mentioned. The final output result of this part is a concatenation of object category, confidence and coordinate information, which needs an additional non-maximum-suppression (NMS) step (see Chapter 3.2.3).
with a confidence threshold to filter out the predictions with low confidence, and convert the result back to a normal image-shaped tensor for further processing.

### 3.2.3 Non-Maximum Suppression (NMS) operation

To keep only predictions with high confidence and collapse overlapped predictions to one, NMS operation is performed on the output of above mentioned object detection part. For each detection, given a confidence score of that particular prediction as one of the input, NMS operation will choose whether to keep this detection or not depending on whether its confidence score is above a certain threshold. Choosing the value of this threshold has an important influence on the final result. The general target is to increase recall, and at the same time decrease the influence of false detection. Different datasets may require different thresholds to achieve this goal. For PASCAL VOC2007, threshold 0.6 is enough to get most of the true positives kept, and eliminate most false positives. However, for MS COCO dataset that contains many small objects, the threshold needs to have a smaller value so as to get a satisfactory recall. In our implementation, the confidence threshold is set to 0.5, which will detect most of the small objects, and have a reasonable false positive rate. We will further minimize the influence of low-confidence predictions described as follows.

The results gotten from NMS operation cannot be directly used by the following layers, as they are information alone out of an image context. Therefore, the information representing those predictions with confidence score high enough is translated back into a 2-dimensional tensor in our model, as shown in Figure 3.3a. To merge with the saliency prediction stream later, the created tensor served as the output of this layer is set to have a size of $38 \times 38$, and its channel number $N$ is the same as the number of detailed classes. Each channel represents the prediction of one
particular class. For an input image, if there exists predictions of objects in class $Cat_i$, then the $i^{th}$ channel of the tensor will have non-zero pixels according to the position of predicted object and the prediction confidence. To lower the influence of low confidence predictions, and strengthen the influence of high confidence ones, the value at each pixel location $(x, y)$ is calculated as follows, assuming there exists an object at $(x, y)$ with prediction confidence $Conf$:

$$Val(Conf) = \frac{1}{1 + e^{-12 \times (Conf - 0.6)}}$$

which has a shape shown in Figure 3.3b. Tensor value at which there exists no predicted object, namely the prediction confidence is below 0.5 and has been filtered out, is set to 0, the same as assuming a 0 confidence for that pixel position. If the detection stream predicts multiple objects in the same class $Cat_i$ and the objects have overlapped bounding boxes, then the pixel value at $(x, y)$ at $i^{th}$ channel inside the overlapped region will have a value $\max[Conf_1, \ldots, Conf_k]$, where $Conf_1, \ldots, Conf_k$ are the confidence of predictions that have bounding boxes enclosing pixel $(x, y)$. The conversion from prediction information to the tensor directly related to the image space is done for all valid predictions, and then the converted tensor is output to the next mapping operation layer.
3.2.4 Mapping operation

(a) Combine multiple channels given user-defined mapping between super categories to detailed classes.

(b) An example of a combined channel from two channels.

Figure 3.4: Mapping Operation Layer

For mapping operation, the user-defined preference vector and the mapping between super categories to detailed classes are passed to the model as extra inputs. The results coming from previous NMS operation are in detailed classes, while this mapping operation will combine them into general categories. Given a super category $SCat_i$ to detailed class $Cat_{ij}$ mapping:

$$\{SCat_1 \rightarrow Cat_{11}, \ldots, Cat_{1k_1};$$

$$\vdots$$

$$SCat_n \rightarrow Cat_{n1}, \ldots, Cat_{nk_n}\}$$

tensor channels representing $Cat_{ij}(\forall j)$ will be merged into a single channel $SCat_i$: Figure 3.4a shows for an particular mapping, how multiple channels representing different detailed classes are combined to channels that represent super categories. The pixel-wise value of the new channel representing $SCat_i$ is: $SCat_i = \max_j[Cat_{ij} \times pvec[SCat_Id_i]]$, where $pvec[SCat_Id_i]$ represents the preference towards category $SCat_i$. The process of this mapping operation is summarized as in Figure 3.4b and it is the key operation that makes the model take in and fit user preferences.

3.2.5 Merging two streams

The model merges two streams together by tensor concatenation, and two $1 \times 1$ convolutional layers with channel number being 8 and 1 are added to learn the combination way of saliency information.
and preference information. Furthermore, as people tend to focus on the central part of their eyesight, we add a center prior to our model before the final activation layer, and we generate this prior map from saliency labels in SALICON dataset by summing up all the saliency ground truth \( \text{SAL}_g \) in the dataset, and then normalize the prior map to \([0, 1]\):

\[
prior = \sum \text{SAL}_g,
\]

\[
prior = \frac{\text{prior} - \min[\text{prior}]}{\max[\text{prior}] - \min[\text{prior}]}
\]

Finally, a softmax activation layer is added to output the final prediction as a probability map.
CHAPTER 4

IMPLEMENTATION

In this chapter the training procedure of PANet will be described. We will first discuss how to get the ground truth data for training and validation, and then talk about different phases to train the model.

4.1 Data Collection

Currently there is no personalized attention annotation available. To train our model, we need ground truth data labeled with consistent preference. In our implementation, we first collect a small dataset labeled by two subjects with their own preferences. Then we generate more ground truth data according to our collected labels.

To collect this dataset, we asked two subjects A and B to firstly define their preference super categories, then rate each category in the scale of 0 to 10, the higher rating means the subjects is more interested in that category. We re-scale their rating to 0 to 1 to get their corresponding $pvec$, listed in Table 4.1. After this, we select a subset of images from SALICON dataset that consists of 15k images. The selected images contain at least one object that belongs to the categories rated larger than 5. For example, images selected for subject A should contain at least one cat or one car, as labeling images having both preferred objects and other objects can make subject label more effective. This turned out to be 3126 images for subject A and images 3974 for subject B: we randomly choose 500 images each for subjects to label.

<table>
<thead>
<tr>
<th></th>
<th>[cat, car, others] = [1.0, 0.8, 0.2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>[keyboard, food, person, others] = [1.0, 0.8, 0.5, 0.3]</td>
</tr>
</tbody>
</table>

Table 4.1: Subject preference vectors in our collected dataset.

The label collection is done through mouse tracking and has a similar procedure as [24]. Each image is presented to the subject and the subject is asked to move the mouse cursor to where
they want to look at. A mouse click will trigger the recording process as well as finishing current recording. The presentation of image is done in Matlab R2015b, and has a GUI as shown in figure 4.1, in which the mouse cursor is represented as a red circle. When the subject finishes current image labeling, he will need to hit Next button for the next image to show up. There will be 2 seconds waiting interval showing a blank figure between two images. As in [24], 50 images form a block, and subjects are allowed to take a short break between blocks. To make our result stable, we ask each subject label their 500 images five times, in five separate days. The final fixation label for each image is averaged from those five labels. Figure 4.3 shows some visualized results.

4.2 Data Generation

Taking into account that preference is unique, and it is impractical to build a dataset with preferences that cover all typical users, it is better to use ground truth generated dynamically from objective ones. Using generated ground truth can make us train PANet with more data, and make the training flexible on new preferences, so that we do not need to ask each new user using our model to label thousands of images with their preferences. We generate our training ground truth data upon different preference vectors by utilizing currently available detection labels and attention annotations, in our case the MS COCO dataset [33] and its attention annotation dataset SALICON [24].
Given a particular \textit{pvec}, we first generate the preference map \textit{pMap} of all images, using the ground truth bounding box positions and categorical information of the objects. The preference value at each pixel position in an image is:

\[ pMap[x, y] = \max(pvec[SCatId[x, y]]) , \]

where \textit{SCatId[x, y]} are the super category IDs of the objects at position \([x,y]\). There might be multiple objects covering \([x,y]\), leading to multiple preference values at a single pixel, and the final value of \textit{pMap}[x, y] is the maximum among all preference values. Then we use the saliency ground truth \textit{SAL}\textit{gt} in SALICON dataset as prior, redistributing the attention according to \textit{pMap}. In addition, \textit{SAL}\textit{gt} is added with a weight to make sure salient area receives fair amount of attention even it is not the preferred type of object by the user. There are also regions containing objects in the preferred categories, but have low saliency probability in SALICON annotations, these regions will not get attention if we only consider \textit{SAL}\textit{gt} as prior. To give a chance to these areas, \textit{pMap} itself is also added:

\[ PSAL_{gt} = \alpha \textit{SAL}_{gt} + \beta \textit{SAL}_{gt} \cdot pMap + \gamma pMap , \]

where \textit{PSAL}\textit{gt} is the generated personal saliency ground truth with the preference \textit{pvec}. To choose the values of weights, we first fix \( \beta : \gamma \) ratio to be 0.8 : 0.2, and find \( \alpha = 0.06 \) will give the best performance, as shown in Figure 4.2a in terms of average CC and SIM scores. These two metrics measure the relationship and similarity between ground truth labeled by two subjects and our corresponding generated ground truth, and a higher score means the generated ground truth is...
more similar to the labeled one. Their mathematical meaning will be explained in detail in Chapter 5.1. After determining the most fitting $\alpha$, we change the ratio between $\beta$ and $\gamma$, the generation performance is shown in Figure 4.2b. $\beta : \gamma = 0.8 : 0.2$ getting the best score. Under the constraint $\alpha + \beta + \gamma = 1$, the final weights are 0.06, 0.752 and 0.188 respectively.

Generated ground truth are then normalized and saved as a probability distribution by going through a softmax:

$$PSAL_{gt} = \frac{PSAL_{gt} - \min[PSAL_{gt}]}{\max[PSAL_{gt}] - \min[PSAL_{gt}]},$$

$$x_i = \frac{e^{x_i}}{\sum_j e^{x_j}},$$

where $x_i$ denotes for every pixel value in each generated PSAL$_{gt}$. We show the comparison of labeled ground truth and our generated one in Figure 4.3.

We also test our generation approach on unlabeled images. To imitate the preference of a person, we randomly generate a preference vector $\text{pvec}$ with a fixed length. In the context of MS COCO dataset, the length is fixed to 12, which is the number of super categories of the dataset. Each element in the $\text{pvec}$ has the range from 0 to 1, and higher value indicates the person is more likely to pay attention to objects belong to this super category. Figure 4.4 shows some results generated upon the preference vector listed in the caption.
(a) Vehicle gets new attention.  (b) Vehicle keeps the attention while the house does not.

(c) Electronics instead of animals gets the most attention.  (d) Electronics keeps the attention while the others not.

(e) Attention shifts to person.  (f) Similar attention, as no particular object is more preferred.

Figure 4.4: COCO image, saliency ground truth, generated personal saliency ground truth with preference vector:

\[
\text{[outdoor, food, indoor, appliance, sports, person, animal, vehicle, furniture, accessory, electronic, kitchen]}
\]

\[= [0.833, 0.346, 0.189, 0.098, 0.934, 0.679, 0.481, 0.875, 0.081, 0.579, 0.901, 0.223].\]

## 4.3 Data Augmentation

Augmenting input image data will help the model be more robust to various inputs and make it less overfit during training. We augment our data in several options:

- **Random Crop**: Each input image has 0.5 probability to be cropped. The size of cropped patch is random between $[0.75, 1.0]$ of the original image, and the aspect ratio is between $[0.8, 1.25]$.

- **Flip**: Each input image might be flipped horizontally and vertically, each with 0.5 probability.

- **Color**: Image saturation, brightness, contrast and lighting might change, each with probability 0.5. The parameters needed for changing a particular color option are generated randomly.

For the former two options, if the input image is augmented, the ground truth needs to be cropped or flipped correspondingly.
4.4 Training

We train PANet in three phases. Input images fed to the whole model are rescaled to $300 \times 300$. For all training phases, we stop training when the model starts to overfit, namely when the validation loss starts to increase. We do this by keeping a track of the minimum validation loss, if the validation loss is smaller than the minimum, we update current loss as the minimum. If the validation loss of three continuous epochs are all larger than the minimum one, the training will be stopped.

4.4.1 Pretraining: Training a SSD model

Our model will use the weights of SSD300 layers. To get the weights trained on our desired data, we pretrain a SSD300 model on the MS COCO dataset which contains 80k training images and 40k validation images. Its feature extraction layers, namely VGG-16 without the final classification layers, have the weights pretrained on ImageNet. The training process is adapted from [50]. According to MS COCO section in [36], the scale of the default boxes are set to be smaller than normal in order to fit small objects in this particular dataset. As for the default box aspect ratio, we don't use the generation method described in [36], instead, we first get a prior box distribution from the dataset ground truth bounding boxes, and then use this prior distribution for the default box generation. The training objective is the combination of confidence loss and localization loss:

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + L_{loc}(x, l, g)),$$

where confidence loss is the cross entropy loss between predicted confidence and ground truth class label (0 and 1), and localization loss is smooth L1 loss between the predicted bounding box and the ground truth one. We train the model for 180k iterations with batch size 20, and the learning rate starting from $3 \times 10^{-4}$, decreasing every epoch with a rate of 0.9.

4.4.2 Pretraining: Training saliency layers

Without merging the result from the preference fitting stream, the weights of those layers responsible for saliency prediction are pretrained. This pretraining process will improve model performance compared with directly training the entire model. We show the layers that we want to learn the weights of in Figure 4.5. We import and freeze the weights from previous phase into this phase.
in order to extract image features. The model is trained on the original SALICON dataset, which contains 10k training images and 5k validation images. The ground truth label is unbiased saliency area by averaging labels from different workers. We directly regress the model output, the predicted saliency in probability distribution, to the ground truth distribution pixel-wisely. Training objective for this phase is KL-divergence:

\[ D_{KL}(p|q) = \sum_i p_i \log \frac{p_i}{q_i} \]

which is a common metric for saliency prediction models. We trained this part for 30k iterations with batch size 20, and the learning rate starts from \(10^{-3}\), decreasing by 0.9 per epoch.

### 4.4.3 Training a PANet

After pre-training phases, the entire model is trained on our dynamically generated ground truth data (Chapter 4.2) containing 10k training images and 5k validation images. For our training, randomly generated user preference and default MS COCO category mapping are served as extra inputs to the model, where the former one is also used in ground truth generation. In a real use case, these inputs should be collected user preference and user-defined mapping. We load and freeze the weights of VGG part and SSD add-on layers, while the weights learned for saliency prediction layers are loaded and going to be fine-tuned. Layers without pretrained weights are initialized with random weights. The training objective is again KL-divergence, and we train the model for 50k iterations with batch size 20 and the same learning rate as in the previous phase.
CHAPTER 5

EVALUATION

In this chapter, we will first describe our results in the quantitative manner, and then show its prediction results qualitatively. As we do not want to retrain our model every time when it is applied to a new user, or when the user updates his preference, we will present the experimental results of using a trained model to a new preference vector and user-defined mapping in Chapter 5.3. At the end, we will also show the effect of biased factor by comparing our model with a standard saliency prediction model.

5.1 Quantitative Results

5.1.1 Datasets

Different from unbiased models, our model requires to be tested on ground truth that generated according to input preference. Saliency benchmark datasets including MIT300, MSRA-1000 [40], ECSSD [53], DUT-OMRON [64] provide pixel-wise saliency information, and the last one with additional fixation and bounding box labels. Nonetheless, they all lack of the object category information required by the ground truth generation process as described in Chapter 4.2. We therefore investigate current detection and classification datasets that provide object class and location label, including PASCAL VOC, MS COCO, ILSVRC. The former two have corresponding extended sub-datasets with fixation or saliency labels: PASCAL-S [56] and SALICON. PASCAL-S contains 850 images taken from the validation set of PASCAL VOC 2010 segmentation challenge, with an extra file indicating corresponding image ID for each selected image. SALICON dataset directly extends the original MS COCO dataset structure, also retaining the original image ID which can be used to get object category and location information. However, we need to re-split it as it only provides train and validation sets without the test set.
5.1.2 Metrics

There are several works \[47\][58] comparing different models under different metrics, and works \[6\][61] that illustrate the use case and particular features of each metric in detail. In general, there are two metric categories when it comes to evaluate a saliency prediction model. One category needs the ground truth to be binary fixation maps, which is the direct averaged labeled data by human workers: if the number of workers labeling a pixel as salient is greater than a threshold, then the value of this pixel is set to one, otherwise the pixel value is zero. Metrics such as Normalized Scanpath Saliency (NSS), Information Gain (IG), Area Under ROC Curve (AUC), and AUC variants including shuffled AUC (sAUC) are used to measure the performance of models given the binary ground truth, in which sAUC specially discounts the influence of center bias incorporated in saliency prediction models.

Another type of metrics work on continuous ground truth. Unlike binary fixation maps, this type of ground truth is not location based that can be interpreted as a sample from a distribution, instead, it is distribution based that directly represent the underlying distribution. Metrics used to measure upon this kind of ground truth directly compare the difference or similarity between two distributions: Similarity (SIM) measures the intersection between two distributions that are interpreted as two histograms, Pearson’s Correlation Coefficient (CC) evaluates the linear relationship between two distributions, Kullback-Leibler divergence (KL-divergence), which is used as the loss function for our model regression, measures the difference between two probability distributions, and Earth Mover’s Distance (EMD), originally used for image feature matching, measures the spatial distance between two histograms.

Traditionally, the distribution based ground truth is blurred and post-processed from the original fixation map, during which the blur parameters (e.g. Gaussian sigma) has influence on the performance results and needs to be carefully tuned. On the other hand, to incorporate the preference factor, our generated ground truth was initially blurred (the sigma value is pre-defined in the SALICON API) and processed into probability distributions. It is natural to evaluate our model using metrics that measure the divergence between distributions with continuous values. We will show the performance of our model on SALICON and PASCAL-S with CC, SIM, KL-divergence and EMD measures in Chapter 5.1.3 and Chapter 5.1.4.
5.1.3 SALICON Results

(a) PANet. (b) Baseline1: center prior baseline. (c) Baseline2: detection baseline.

Figure 5.1: CC score histograms. Vertical axis is the number of testing images.

SALICON is more suitable for testing our model, as images in MS COCO dataset generally contains more objects and covers different super categories compared with PASCAL images. We randomly re-split the 15K images in SALICON into 7K training images, 3k validation images and 5k test images, and then train the model following the same procedures as in Chapter 4.4. The preference vector are the same as in Figure 4.4.

Here we compare our model with two baselines. The first is center prior, which sets the saliency baseline. The second is detection baseline: for each image, we highlight detected objects with detection confidence $\times$ object preference. One problem of the second baseline is that there exist images that no object is detected with confidence higher than the 0.5 threshold as mentioned in Chapter 3.2.3. This will result in zero matrices as detection baselines for these images. To solve this, we replace these zero matrices with randomly generated matrices with pixel values between 0 and 0.01, and then do a normalization to make sure all pixels sum to 1.

We calculate CC:

$$CC(p, q) = \frac{\sigma(p, q)}{\sigma(p) \times \sigma(q)}$$

where p,q are the predicted saliency map and dynamically generated ground truth respectively, and $\sigma(p, q)$ is the covariance of p and q. The result is shown in Figure 5.1a, with most images have high CC scores, and the average is 0.725. The CC of center bias and generated ground truth has an average 0.420, and the distribution is shown in Figure 5.1b. Detection baseline has a average CC 0.493 and distribution as shown in Figure 5.1c. These indicate that PANet predicts much better than the two baselines.
Apart from the linear relationship, we also measure the similarity between predicted saliency map and our ground truth. The similarity metric, SIM, is calculated as follows,

$$\text{SIM}(p, q) = \sum_{i} \min[p, q],$$

where $$\sum_{i} p_i = \sum_{i} q_i = 1$$

Two same saliency maps will have $$\text{SIM}(p, q) = 1$$. Different from CC that is calculated symmetrically and measures false positive and false negative equally, SIM metric penalizes false positives less than false negatives, making the center prior baseline scores higher under this metric. Figure 5.2a shows an example of SIM measure, and Figure 5.2b shows the SIM score distribution of our model. The average score on the test set is 0.742 and the center prior baseline has SIM value 0.622 comparing with the ground truth, and the detection baseline scores 0.587. Intersection ratio of predicted distribution and ground truth falls mostly between 0.7 to 0.9: in the case of our model, the predicted saliency area usually has a larger coverage thus values less at each salient pixel, lowering the summation of $$\min[p, q]$$.

KL-divergence measures the difference between two saliency maps that are viewed as distributions, and we used KL-Judd variant in our measurement:

$$KL(p, q) = \sum_{i} q_i \log(\epsilon + \frac{q_i}{\epsilon + p_i}),$$

where $$\epsilon = 2.2204 \times 10^{-16}$$ is a regularization constant. KL-divergence is non-symmetric, and penalizes the pixel predictions where $$q_i$$ are much larger than $$p_i$$. For our experiment on MS COCO with
this particular preference vector, the average KLD is 0.159, and the center prior baseline is 0.316. Detection baseline has a much higher average KLD score 11.222 as its distribution is much different from a saliency map. We will also see in Chapter 5.3 that the variance of preference vector will change KLD score significantly, making it less suitable for measuring our model performance.

EMD measures the spatial distance between distributions by computing the minimum cost of moving densities from one distribution into another. The version we use in our experiment is its linear variant as in [5]:

$$EMD(p, q) = \min_{\{f_{ij}\}} \sum_{i,j} f_{ij}d_{ij} + \sum_{i} p_i - \sum_{j} q_j \max_{ij} d_{ij},$$

under constraints: (1) $f_{ij} \geq 0$  (2) $\sum_j f_{ij} \leq p_i$

(3) $\sum_i f_{ij} \leq q_j$  (4) $\sum_{i,j} f_{ij} = \min(\sum_i p_i, \sum_j q_j),$

where $d_{ij}$ is the ground distance between $i^{th}$ and $j^{th}$ bin of the distributions, and $f_{ij}$ is the density flow between them. The average EMD score on the test set for this particular preference vector is 0.685, with center bias scoring 3.668 and detection baseline scoring 5.116.

### 5.1.4 PASCAL-S Results

<table>
<thead>
<tr>
<th>pvec</th>
<th>aeroplane, bicycle, boat, bus, car, moterbike, train</th>
<th>vehicle</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bird, cat, cow, dog, horse, sheep</td>
<td>animal</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>bottle, chair, dining table, potted plant, sofa, TV monitor</td>
<td>indoor</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>person</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 5.1: Pascal mapping and preference setup in our experiment.

<table>
<thead>
<tr>
<th></th>
<th>CC</th>
<th>SIM</th>
<th>KLD</th>
<th>EMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>0.664</td>
<td>0.689</td>
<td>0.0116</td>
<td>0.622</td>
</tr>
<tr>
<td>baseline1</td>
<td>0.491</td>
<td>0.610</td>
<td>0.318</td>
<td>3.609</td>
</tr>
<tr>
<td>baseline2</td>
<td>0.493</td>
<td>0.570</td>
<td>13.102</td>
<td>4.453</td>
</tr>
</tbody>
</table>

Table 5.2: Average scores.

We setup our experiment for PASCAL-S as in Table 5.1 we define four super categories for the 20 classes in PASCAL VOC, and the corresponding preference vector is set to be [0.1, 1, 0.3, 0.5].
Figure 5.3: Performance on PASCAL-S.

There are only 850 images available with both detection labels and fixation labels, and we randomly split them into training set with 400 images, validation set with 200 images, and test set with 250 images. We train the model in three phases as in Chapter 4.4, however the iteration numbers are much lower when training on augmented PASCAL-S (1K for the second phase and 2K for the final phase) as we have much fewer training images.

The results are shown in Table 5.2 and Figure 5.3 comparing model prediction with center prior baseline and detection baseline. The difference of PASCAL-S and SALICON can be reflected in the results. Although the prediction accuracy of the detection layers is higher for PASCAL images, the whole model performance is lower. The reason is that PASCAL dataset mostly contains very few objects inside one image, and the objects tend to be in the same class, whereas MS COCO images contain more objects in more categories, in which case the model is more useful as it will have the chance to treat each object differently according to the preference.
5.2 Qualitative Results

We show our model predictions in Figure 5.4, which are tested upon MS COCO images with the preference vector:

\[
\text{[outdoor, food, indoor, appliance, sports, person, animal, vehicle, furniture, accessory, electronic, kitchen]} = [0.833, 0.346, 0.189, 0.098, 0.934, 0.679, 0.481, 0.875, 0.081, 0.579, 0.901, 0.223].
\]

The performance bottleneck is the accuracy of detection layers. When the detection stream cannot recognize an object (no corresponding bounding box) or recognize it wrongly, false negative and false positive will occur. The predicted saliency map either pays not enough attention to correct areas, or pays attention to wrong areas. For example, in Figure 5.5, the foreground and background chair belong to the same category, but the detection stream wrongly predicts the foreground one as an object with high preference, attracts most attention to it. The detection confidence might also
Figure 5.5: Problematic predictions.

have negative influence on the result. Figure 5.5b has three cars in it, but the foreground one has a low prediction confidence thus receives much less attention than the two at the back. Another problem of the model is that when objects in the preferred category fill the image, the model will make the whole image get attention and cannot distinguish the important part inside an object bounding box, as shown in Figure 5.5d.

5.3 Same model, different preference

Figure 5.6: COCO image, model predictions with different preference vectors:

$pvec_1 = [\text{outdoor}, \text{food}, \text{indoor}, \text{appliance}, \text{sports}, \text{person}, \text{animal}, \text{vehicle}, \text{furniture}, \text{accessory}, 
\text{electronic}, \text{kitchen}]$

$= [0.833, 0.346, 0.189, 0.098, 0.934, 0.481, 0.875, 0.081, 0.579, 0.901, 0.223]$

and $pvec_2 = [\text{person, non-human}] = [1.0, 0.05]$. 
We allow users to define their own preference mapping, and each mapping may points to different number of super categories. To fit different mappings without retraining the whole model each time, we fix the output channel of the Mapping layer (see Figure 3.1) to 20, which sets the maximum limit on allowed number of super categories. Now the preference vector always has length 20. If the number of user-defined super categories is less than 20, we automatically fill the remained entries by zero.

In our experiment, we test the model with two preference vectors of different lengths on the SALICON test set (split as in Chapter 5.1.3): $pvec_1$ and $pvec_2$ listed in caption of Figure 5.6. First, we directly pass the new preference vector $pvec_2$ and new mapping $[\text{person} \rightarrow \text{person}; \text{others} \rightarrow \text{non-human}]$ into the model, which is trained with the ground truth generated upon $pvec_1$. The results are shown in Figure 5.7a3 and 5.7b3. Comparing with 5.7a2 and 5.7b2, they can reflect the difference of saliency areas with $pvec_1$ and $pvec_2$, as the “person” area in the image gets more attention in the predicted saliency map. However, taking into account the new preference vector prefers person to non-human objects with a very strong bias, these results are not good enough to reflect $pvec_2$. We then fine-tune the model to see the results. Loading the weights trained with $pvec_1$, and fine-tuning the model with ground truth generated by $pvec_2$ as well as changing inputs to $pvec_2$ and the new mapping, PANet converges very fast. Only three epochs (1050 iterations) are needed to make the objective function reach its lowest point, which is much less than our original training that takes 50K iterations. Figure 5.7a4 and 5.7b4 show the results predicted by the fine-tuned model, and comparing with the ones gotten from the original model without fine-tuning, they reflect $pvec_2$ much better. More results with different preference vectors are shown in Figure 5.6.

We also measure the performance quantitatively. Figure 5.8 shows the CC, SIM, KLD and EMD score distributions before and after fine-tuning. Table 5.3 shows the average scores of the models: model1 is the model trained on ground truth generated from $pvec_1$, and model2 is the fine-tuned model with ground truth generated from $pvec_2$. When trying to predict the saliency map for a
new preference vector, a little fine-tuning will increase the performance significantly than directly using the old model. The result also indicates that for different preferences, model performances are different on a same metric, which might be influenced by the variance of preference scores. KL-divergence score is largely determined by it, as an even preference will likely result in a dispersive saliency map as each object is getting comparable amount of attention, similar to Figure 5.5c3 and 5.5d3, and such maps have smaller value for each pixel and are more likely to be penalized by KL-divergence measure.

### 5.4 Comparison with unbiased models

To see the effect of shifting attention according to personal preference, we compare our model with general saliency prediction models, test upon the ground truth generated according to different preferences. In this experiment, we keep using MS COCO images, and generate personalized saliency ground truth with $pvec_1$ and $pvec_2$ as in Chapter 5.3. The model we use to compare is Salicon model [21], and we use their codes [57] to generate fixation maps of the images in
SALICON test set without any personal bias. The results are shown in table 5.4.

<table>
<thead>
<tr>
<th>pvec</th>
<th>CC</th>
<th>SIM</th>
<th>KLD</th>
<th>EMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>pvec1</td>
<td>0.527</td>
<td>0.621</td>
<td>0.332</td>
<td>4.605</td>
</tr>
<tr>
<td>pvec2</td>
<td>0.525</td>
<td>0.594</td>
<td>0.337</td>
<td>4.656</td>
</tr>
</tbody>
</table>

Table 5.4: Salicon model tested on the ground truth generated with pvec1 and pvec2.

Comparing the predictions of our PANet and generalized models such as Salicon model, PANet performs much better in fitting user preference (results shown in Table 5.3). The saliency maps predicted by a general model such as Salicon will score similarly to the center prior. For different preferences, the performances will be different but stay in the same level. A more biased preference will lead to a slightly worse performance when using the general saliency model.
CHAPTER 6

DISCUSSION AND FUTURE WORK

Our ground truth generation is based on object bounding box location, which is not accurate enough compared with instance-level segmentation. However, there is no available segmentation dataset labeled with object category information at present. As a future work, a dataset on instance-level segmented saliency map with categorical information about each object might be collected. The starting point can be the segmentation dataset such as PASCAL-S \cite{56}, and ask workers to label each segmented objects with their class information.

Current model has non-differentiable operation layers, making the execution time depends on the object numbers in the input image. Revising the model to a differentiable one will be the major future work for us. The model can also be extended into focusing on the familiar person when combined with face recognition models. For example, in an image containing multiple people, the attention will be focused more on the acquaintance even if that person is farther away than others. Another possible direction is extending current convolutional network into a recurrent version, thus it can find the personalized pattern about how the viewer changes the attention areas through a time series. The order of attention change can be used to generate personalized description of a scene.

As for identifying the user preference, currently our work is one-time identification. A more advanced online updating approach can be explored to adapt the preference for current context, thus making the model always up-to-date. How to keep user inputs as few as possible, and at the same time increase the identification accuracy can be a side project to work on.
CHAPTER 7

CONCLUSION

In this work, we design, implement, and evaluate a personalized attention prediction model PANet, which can predict the saliency area in an image according to individual user preference. This work presents the novel approach of including user preference in the vision model, allowing more efficient post-processing steps in various HCI applications. The shared feature extraction layers can efficiently provide multi-scale features for both saliency prediction stream and object detection stream, and the latter one will be used to fit individual preference through our customized non-maximum-suppression layer and mapping layer given a preference vector as the extra input. In order to train the model to fit input preference, we collect saliency labels from two subjects together with their preferences. For new input preferences, we dynamically generate ground truth from established datasets during training and validation, with parameters based on collected labels. We experimentally validate that our model can predict saliency areas according to the input preference, and it can work for different preferences and user-defined class to super category mappings with little fine-tuning. Compared with general saliency prediction models, it fits much better to the saliency maps biased with personal tendencies. The more biased a preference is, the more performance gain provided by PANet.

Apart from the model itself, the concept of considering user experience is important for vision works designed for HCI scenarios. Many improvements can be done on personalized vision, in terms of efficiently incorporating user preference, as well as increasing the execution speed, so as to achieve fitting and smooth user experience. As individual differences are being appreciated more and more, we believe this is a promising direction to explore as a part of larger application systems.
REFERENCES


