Project Report

WAYLA - What Are You Looking At

Jin Peng Zhou, Hengrui Jia

University of Toronto
Department of Engineering Science
Option of Machine Intelligence

Word Count: 1999

1 Goal and Motivation

The goal of this project is to predict what image features (i.e. objects) a normal person would look at on a given image, as indicated by the project name.

Nowadays, eye tracking technique is commonly used in visual neuroscience and cognitive science to answer visual attention related questions [1] since some studies have shown that human gaze behavior is able to reflect cognitive processes of the mind [2], such as intention [3, 4, 5], and is influenced by the user's task [6]. Therefore, when an image containing many different image features is provided, it is useful to know how attention of a normal person is biased towards the image features for some studies related to cognitive process. This motivates us to work on this project, which is predicting maps for images indicating attention bias of people (i.e. saliency map [7]) by using neural network.

Another motivation is that we believe it is interesting to show that there can be artificial intelligence (AI) similar to human intelligence in terms of having attention bias when inspecting images. The term attention bias means people would spend different amount of attention/time on objects in an image. For example, when a picture of kids on playground is shown, people would pay more attention to the kids rather than the playground. If the same image is provided to the AI and it also "focuses" more on the kids, then we can conclude they have similar attention bias on this image. By doing so, we can show that it is possible for AI to be as smart as human brains in this area.

2 Description of Overall Software Structure

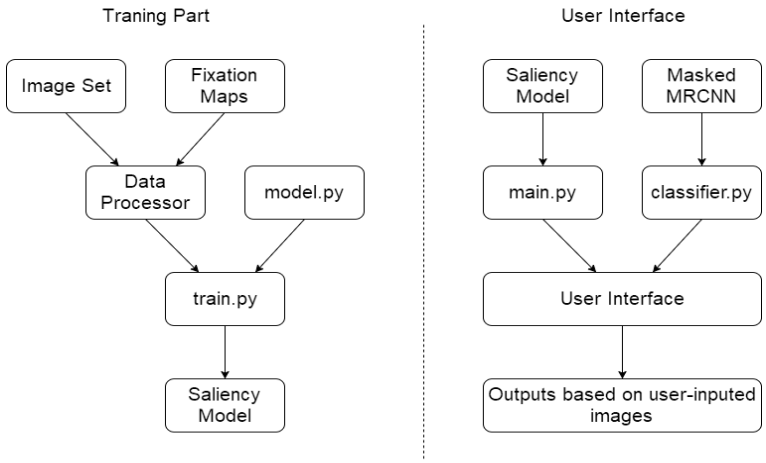


Figure 1: Overall Software Structure

As seen from Figure 1, this software contains 2 parts: training and user interface. The training part processes the data, defines the architecture of the neural networks, and trains the model. In details, there is a data processing function which loads a batch of 640 x 480 x 3 images and their corresponding fixation points in the form of lists of

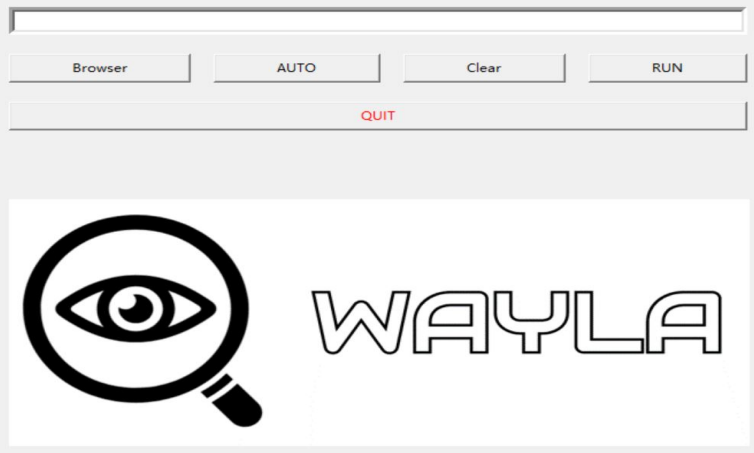


Figure 2: User Interface

(x, y) coordinates and converts them into `numpy` arrays. Then based on the neural network architecture defined in the `model.py` file, we train the saliency model.

The user interface part consists of two trained neural network models and python scripts which convert each model into a stand-by-itself software. By using the software, user can generate prediction of fixation point maps (i.e. locations of gaze) of given images, lists of objects in the images, and prediction of attention bias of normal people on each image, without knowing anything about the code and knowledge of programming. In details, the saliency model takes input of a 640 x 480 x 3 image and generates a predicted heatmap of fixation points (similar to saliency maps, which is the reason why named as saliency model) for each image. Then the images are fed into the pre-trained Mask R-CNN [8] (i.e. an object localization neural network) to produce bounding box for each object in the image. By combining the map of fixation points with bounding boxes, we can calculate the attention bias of a person on a given image by counting the proportion of fixation points within each bounding box.

As shown in the Figure 2, the user interface consists of a text field where the user can input the directory of his/her data (this is the only input required by the software), 5 buttons, and the logo of this project. The five buttons are "Browser", which lets the user browses his/her computer; "AUTO", which automatically inputs the directory of the software into the text field; "Clear", which clears the text field content; "RUN", which generates heatmaps of fixation points by saliency model or attention bias predictions by Mask R-CNN model depending on which model we are running; "QUIT", which terminates this program. It is designed for usability. For example, when one uses this software, they only need to put appropriate data in the same folder as the software, and then click "AUTO" and "RUN" to get the output.

3 Source of Data

There are two sources of data: SALICON [1] and Microsoft COCO [9]. SALICON is an online dataset consisting of 15,000 640 x 480 x 3 images with corresponding fixation points in the form of lists of (x, y) coordinates [1]. All images were taken from Microsoft COCO, which is another online dataset that was originally used for image classification. According to SALICON, the saliency maps are generated by a mouse-contingent multi-resolutional paradigm (i.e. people use mouse to indicate where they are looking at) based on neurophysiological and psychophysical studies and that is proven to be similar to fixation points generated by eye-tracking systems [10].

Since the size of this dataset is very large and downsizing the images may affect the accuracy of the trained model, we process the data as the training goes on and free the data from memory immediately after finishing using them.

Besides, since the fixation points are (x, y) coordinates, we converted them into a matrix of size 640 x 480, and each entry of the matrix is either 0 or 1, where 0 means there is no fixation point on that pixel, and 1 means there is a fixation point on that pixel.

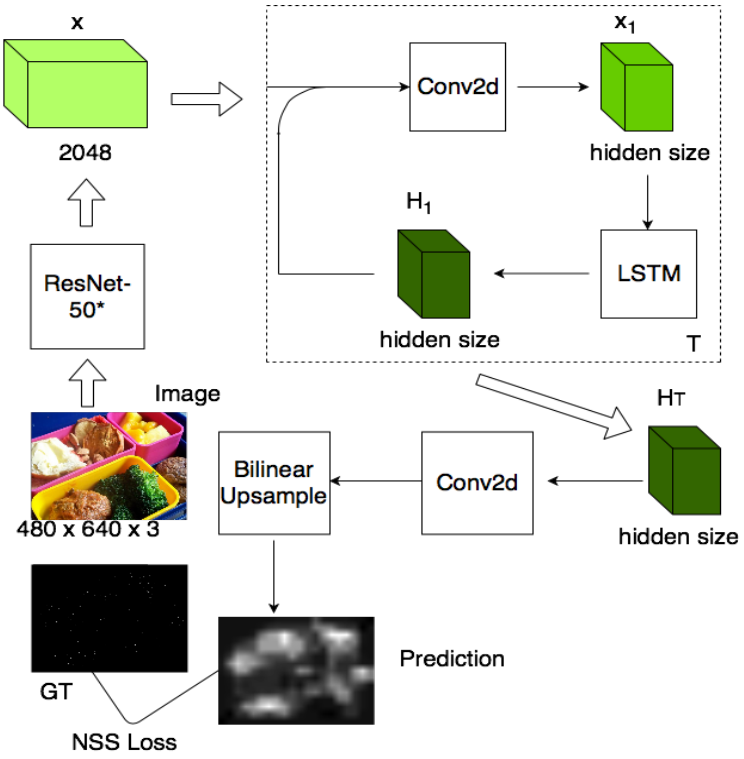


Figure 3: Model Structure

4 Description of the Trained Model

The overall structure of our model consists of four layers: a pretrained ResNet-50 [11] on ImageNet [12] with average pooling and fully connected layer removed followed by a convolutional LSTM followed by an upsampling layer followed by a pretrained Mask R-CNN network trained on Microsoft COCO to make final prediction. The model takes a 640 x 480 x 3 image as input and outputs an attention bias prediction, which is a vector of class attention probability distribution that sums to 1, along with bounding boxes that represent the object instances and the most probable locations people will be looking at. The first three layers are depicted in Figure 3 and an example of final output from the last layer is illustrated in Figure 4. The following subsections are dedicated to a detailed description of each layer of the model.

4.1 ResNet-50

We employ a ResNet-50 model pretrained on ImageNet as our feature extraction/representation layer. We remove the average pooling and fully connected layer since we are not interested in object classification in this part but rather feature maps. The reason why we intend to use a pretrained ResNet-50 is because we believe it is good at image feature extraction and representation, and it is very general purposed in tasks related to images so that we can use transfer learning to reduce the amount of computation and time needed in the training process. This part of the network is kept fixed during backpropogation. This layer takes a 640 x 480 x 3 image as input and outputs a feature map that serves as input of the Convolutional LSTM layer.

4.2 Convolutional LSTM

LSTM has shown promising results on time-series prediction and attention-based tasks. Since we are predicting where people will be looking at for an image, our intuition suggests that we should allow the network to continuously refine its prediction rather than generate the prediction in one step. Consequently, we feed our feature map obtained from ResNet-50 layer into a Convolutional LSTM and obtain a hidden state at each time step. The hidden state and the original feature map are then concatenated together for input at next time step. This allows our model to continuously refine its fixation heatmap prediction as time step increases. This is illustrated in Figure 5 and as time step increases, the prediction gets more granulated and refined, and empirically this confirms our intuition. The

Image	bench	bus	motorcycle	person
1	16.32 %	38.85 %	15.17 %	29.65 %

Figure 4: Sample Output

hidden state at last time step serves as input of the upsampling layer.

4.3 Upsampling

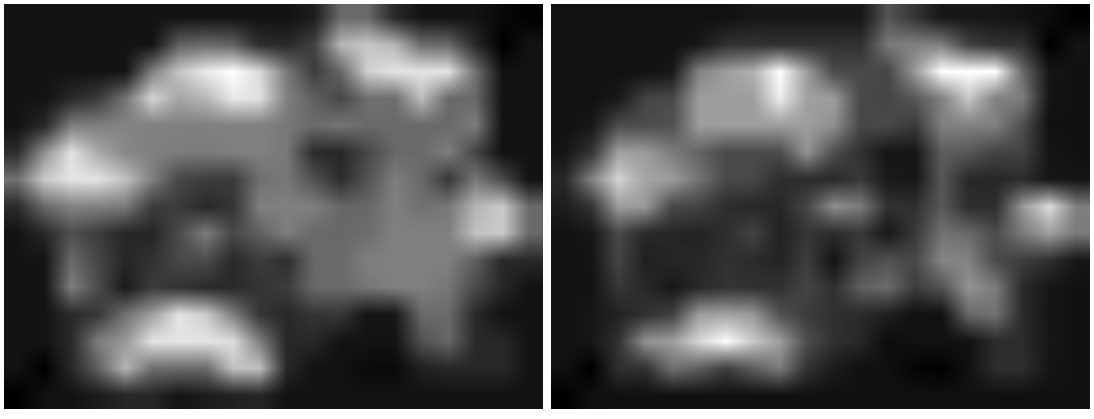
The upsampling layer uses a 2D convolution that makes each feature map two dimensional (i.e. with one color channel corresponding to a grey scale fixation heatmap prediction). The feature map is then upsampled through a bilinear upsampling process to reshape feature map to 640 x 480. The intuition behind this upsampling layer is that the 2D convolution utilizes hidden state information and compresses it to a compact feature map before bilinear upsampling in order to obtain better results.

4.4 Mask R-CNN

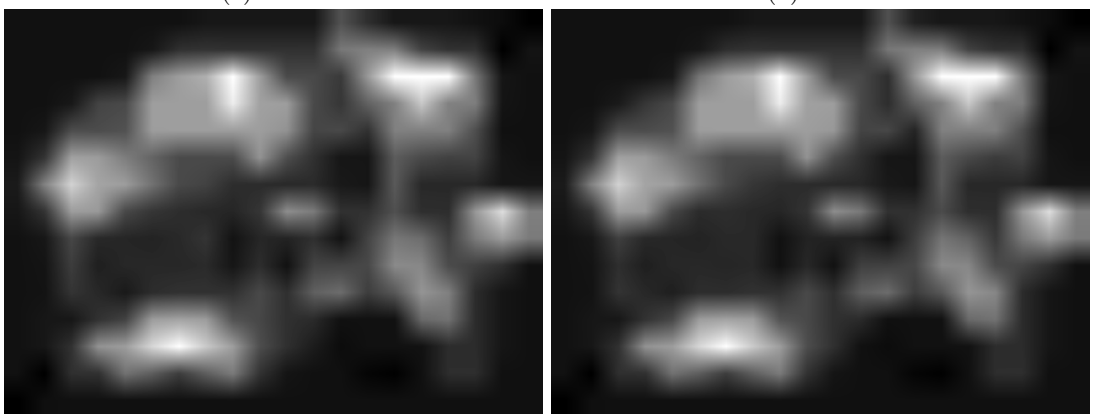
The final layer is a pretrained Mask R-CNN model on Microsoft COCO dataset. This part of the network is also kept fixed and does not go through any parameter updating. Because we need to know what classes our fixation prediction points belong to, we simply use Mask R-CNN to make an objection localization and detection prediction. The prediction generates bounding boxes around each class instance and we utilize this information to categorize our predicted fixation points and obtain our final output.

5 Training, Validation and Test

As discussed in the above section, Mask R-CNN does not need to update its parameters throughout entire training process. After conducting some research on saliency evaluation metrics and losses, We employed a standard saliency comparison loss: Normalized Scanpath Saliency (NSS) [13] on the output of third layer with ground truth fixation maps. We initially allowed ResNet-50 to update its parameters but very quickly found out that it required too much computation and the loss was even higher than the loss with ResNet-50 parameters fixed. Therefore, both the first and last layer were kept fixed during training process. We used Adam optimizer to update weights of the network and tuned various hyperparameters including learning rate, kernel sizes of convolution layers, LSTM hidden size, LSTM time steps and number of epochs with grid search.

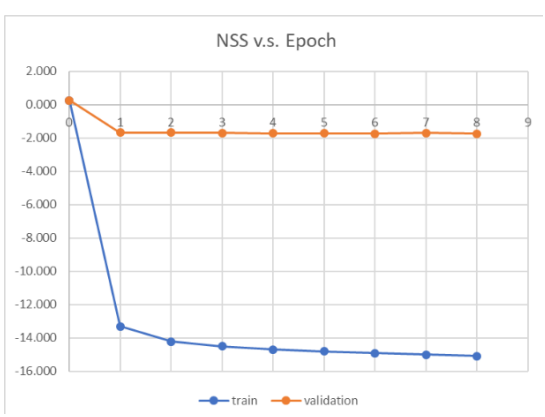


(a) T = 1 (b) T = 4

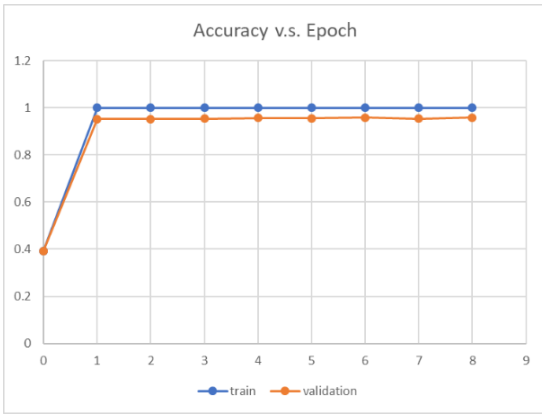


(c) T = 16 (d) T = 64

Figure 5: Fixation Prediction at Different Time Steps



(a) Training and Validation Loss



(b) Training and Validation Accuracy

Figure 6: Results

5.1 Evaluation

NSS can both be a loss and an evaluation metric with the difference of a minus sign.

$$-NSS = L_{NSS}(\tilde{y}_i, y_i^{fix}) = -\frac{1}{N}\sum_i \frac{\tilde{y}_i - \mu(\tilde{y})}{\sigma(\tilde{y})} \cdot y_i^{fix} \tag{1}$$

The formula for NSS loss and evaluation is given in equation 1 where $\tilde{y}_i$, $\mu(\tilde{y})$, $\sigma(\tilde{y})$, y_i^{fix} and N are the value of each pixel, average value of pixels, standard deviation of pixels of prediction, corresponding pixel value of ground truth

and number of fixation points in ground truth respectively. A positive NSS value indicates a positive correspondence between prediction and ground truth and negative value means an anti-correspondence. The training and validation NSS loss values are illustrated in Figure 6 (a). Since NSS is essentially a z score in normal distribution [13], the accuracy of our network is computed by finding the area under the curve from negative infinity to the z value. This is depicted in Figure 6 (b) and our training and validation accuracy are shown to be above 95%. In addition, we also calculated cosine similarity between predicted and ground truth attention bias vector obtained from their respective fixation maps. We consider this as our model final accuracy since it is directly computed between final output and ground truth. The training, validation and test accuracy are 99.9%, 97.2% and 95.1% respectively.

6 Ethnical Issues

- As mentioned in the Source of Data section, the data (i.e. saliency maps) are collected by SALICON. If participants who generated these data are baised (e.g. many participants of a certain age/race/gender group, etc.), the model trained on these data may also be biased.
- Since image classification is involved in this project, and we know image classification may take color as a parameter to classify objects, there may exist some ethnical issues related to skin color of different races.
- Since twp pre-trained models are used, any ethnical issues due to that model may also affect this project.

7 Key Learning and Conclusion

In general, this project meets our expectation. However, due to the restricted amount of time and lack of experience on this kind of machine learning project, there are some drawbacks in the software. For example, since we trained the model on GPU, we did not recognize how slow the model runs on a 4-core CPU. Hence, our software turned out to be relatively slow: it requires about two minutes to process an image.

If we started again, we would definitely do many things differently. First of all, we would find a way to downsize/simplify the model to make it faster, so that our software could be more efficient when dealing with larger datasets. Besides, we would spend some time learning `TensorFlow` so that we may understand the pre-trained Mask R-CNN model and modify it to make it fits our software better.

References

[1] Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. Salicon: Saliency in context. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[2] Hosnieh Sattar, Andreas Bulling, and Mario Fritz. Predicting the category and attributes of mental pictures using deep gaze pooling. *CoRR*, abs/1611.10162, 2016.

[3] Päivi Majaranta and Andreas Bulling. Eye tracking and eye-based human–computer interaction. In *Advances in physiological computing*, pages 39–65. Springer, 2014.

[4] Chris L Kleinke. Gaze and eye contact: a research review. *Psychological bulletin*, 100(1):78, 1986.

[5] Michael F Land and Sophie Furneaux. The knowledge base of the oculomotor system. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 352(1358):1231–1239, 1997.

[6] Alfred L Yarbus. Eye movements during perception of complex objects. In *Eye movements and vision*, pages 171–211. Springer, 1967.

[7] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. In *Advances in neural information processing systems*, pages 545–552, 2007.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[10] Vidhya Navalpakkam, LaDawn Jentzsch, Rory Sayres, Sujith Ravi, Amr Ahmed, and Alex Smola. Measurement and modeling of eye-mouse behavior in the presence of nonlinear page layouts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 953–964. ACM, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[13] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. What do different evaluation metrics tell us about saliency models? *IEEE transactions on pattern analysis and machine intelligence*, 2018.