Predicting where humans look in images by probabilistically modeling the fixations density using Convolutional Neural Networks

Manoosh Samiei
McGill University
Montreal, Canada
manoosh.samiei@mail.mcgill.ca

Abstract

In this project, I implement a high-level saliency prediction model, termed DeepGaze II, proposed by paper [19]. This model extracts high-level features in images using VGG19 convolutional neural network pretrained for an object recognition task. DeepGaze II is trained using a log-likelihood learning framework, and aims to predict where humans look while free-viewing a set of images using high-level features of images such as faces and texts.

1. Introduction

The attention mechanisms of human brain have been studied extensively over the past few decades. Attention plays an important role in visual perception of the world and enables humans and animals to focus their fovea on a part of visual data that is important to them. This is a crucial task as only a small central part of the retina has high density of receptors and allows for high-resolution perception. [17]

Studying how humans selectively choose important parts of the scene can be studied via their eye movements' trajectories. Being able to predict eye trajectories of humans enables us to study human's attention behavior. Furthermore, we might be able to equip different object detectors with the ability to efficiently detect, and localize salient objects in the surrounding environment.

Attention mechanisms are divided into two categories: bottom-up and top-down. Bottom-up (BU) component draws the eyes to places in the scene that contain discontinuities in image features such as a high contrast location (e.g. a red circle among a set of green circles) or are biologically preferred by the observers such as faces. On the other hand, top-down (TD) component guides attention and gaze in a task-dependent and goal-directed manner, and involves paying attention to those objects that are relevant for observer's task. In natural scenarios, these two components are combined in the control of gaze. [2]

To study each of these components separately, scientists study human's eye movement patterns under two main categories: looking at a scene without performing any particular task, which is called free-viewing, or when performing a task such as searching for an object, which is called goal-directed or task-related.

In [19] authors introduce a new model DeepGaze II that can predict where humans look in images while performing a free-viewing task. Thus their predictions mainly involve bottom-up attention. Salient objects can refer to low-level image features such as local contrast or high-level features such as faces, objects and texts. As the authors suggest, these high-level features can be encoded by a deep convolutional neural network pretrained for object recognition task on a different dataset such as ImageNet [4]. DeepGaze II is a probabilistic saliency model, and generates the probability distribution of fixations over the pixels of the input image. This model is trained on SALICON dataset [10] and fine-tuned (cross-validated) on MIT1003 [11] in a probabilistic framework by minimizing negative log-likelihood as its loss function. The authors also introduce another model called ICF (Intensity Contrast Feature) which can predict fixations that occur due to discontinuities in low-level features: local intensity and contrast. They compare the performance of ICF and DeepGaze II models to further analyze how detecting low-level vs. high-level features contribute to fixation predictions. They argue that while DeepGaze II outperforms ICF in most cases, in 10% of images ICF model have better predictions.

In this project, I implement DeepGaze II model, and try to address some of the weaknesses of the model in detecting low-level features. In Section 2, I discuss some of the related works regarding saliency models. In Section 3, I describe the model architecture, the technical details of my implementation, and the theories behind the probabilistic framework used for training and evaluating the model. In Section 4, I present my results, and finally in Section 5, I include some discussions and concluding remarks.

2. Related Work

In computer vision, research on visual attention has been primarily focused on the Bottom-Up (task-independent) component. Early studies were directly influenced by Feature Integration Theory (FIT). [24] Inspired by this theory, Koch and Ullman introduced the idea of ”saliency map” as a topographic map with retinotopic organization where locations that stand out in an image (e.g., because of distinctive low-level features such as color, or texture) are highlighted. [12] The first complete implementation of this architecture was then presented by Itti et al [7]. This was the first saliency model that tried to predict where humans look in images. Since Itti and Koch, many models have been proposed for predicting fixations from image features, and recently many deep-learning based models have been proposed. The first model to use deep features was called eDN [25] which trained convolutional neural networks from scratch to predict saliency. Then DeepGaze I model used transfer learning method by exploiting DNN features that were trained on object recognition (AlexNet trained on the ImageNet dataset), which significantly outperformed training from scratch. This improvement showed a close relationship between high-level features such as objects in the scene and fixation locations, which led many successive saliency models to leverage transfer-learning from object recognition task in their predictions. For example, SALICON model [6] fine tunes a mixture of deep features from AlexNet [14] , VGG-16 [20] and GoogLeNet[21] for saliency prediction. DeepFix [16] and PDP [8] fine-tune features from the VGG-19 network [20].

When authors published their paper in 2017 their model achieved the state of the art performance in MIT saliency benchmark [3]. Currently, their model is ranked 4th (excluding the first two models which are gold standards) after UNISAL [5], EML-NET[9], and MSI-Net [15] in MIT saliency leader board available at `https://saliency.tuebingen.ai/results.html`. UNISAL is currently the state of the art model for predicting saliency both in videos and in images. This model is composed of an encoder-RNN-decoder-style network, and is trained jointly with image and video saliency data. The two other models MSI-Net and EML-NET are both composed of an encoder-decoder architecture and leverage transfer learning in their features extraction stage, similar to DeepGaze II.

3. Theory and Model

As discussed earlier, DeepGaze II models the fixation probability distribution. In other words, for each image in the dataset, it generates a density map which indicates the probability distribution of eye fixations over each pixel of the image. I call the predictions of the model density map, not to be confused with saliency map. Previously when density maps were also regarded as saliency maps, there was inconsistencies among commonly used metrics (AUC, sAUC, NSS, CC, SIM, KL-Div), and it was impossible to rank models appropriately. In papers [17] and [18], authors propose a bench-marking framework to resolve these inconsistencies. They defined a saliency map as a metric-specific prediction (e.g. AUC saliency map for AUC metric) derived from the model density which maximizes the expected performance on that metric given the model density.

They also use log-likelihood framework for model training and evaluation, particularly reported as “information gain”. Later, I will discuss how the two terms information gain and log-likelihood are equivalent in the current context.

3.1. Architecture

The authors use a VGG19 convolutional neural network [20] pretrained on ImageNet dataset to extract high level features (such as objects) from training images. In particular, the authors use a normalized VGG19 network for which all filters have been rescaled to yield feature maps with unit mean over the ImageNet dataset. The weights for this normalized VGG19 network are available for download at authors lab (bethgelab) website. This model is written in caffe framework and since I am using tensorflow, I needed to convert the model to tensorflow checkpoints.

The input image to this network is first downsampled by a factor of two, and then is passed to through the VGG19 network, in which all final fully connected and softmax layers are discarded. Then the output of five high-level layers in the last (fifth) convolutional block namely (conv5_1, relu5_1, relu5_2, conv5_3, relu5_4) are upsampled by a factor of 16 and concatenated to build an array with 5x512 channels (each layer output has 512 channels). The original paper upsamples the layers' outputs by 8, however in order to have a one-to-one correspondence between the pixels of density map with input image, I upsampled the outputs by 16 to match the dimensions of the input image. Note that all convolutional layers are zero padded and do not change the image size, only the 4 pooling layers (before the fifth colvolution block) in VGG19 reduce the dimensions of input image by a factor 2^4. VGG19 architecture is shown in figure 1. Then, the concatenated output array enter the final network called readout network to output a prediction density map for fixation locations. Readout network is made of four layers of 1×1 convolutions with 32, 16, 2, and 1 channels respectively, followed by ReLu nonlinearities [19]. The readout network can only compute a point-wise non-linearity in VGG features across channels. It cannot learn spatial relationships in features as all the convolution layers are 1x1. The architecture of the model can be seen in figure 2.

The architecture of ICF model which detects low-level driven fixations is similar to DeepGaze II, except that instead of extracting features via a pretrained VGG, two low-

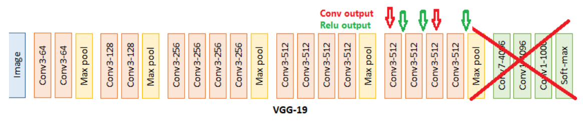

Figure 1. VGG19 architecture used for feature extraction. The red arrows show the output of convolutional layers before applying relu activation, and green arrows show the output of relu activations. The output of five layers conv5_1, relu5_1, relu5_2, conv5_3, and relu5_4 are extracted.

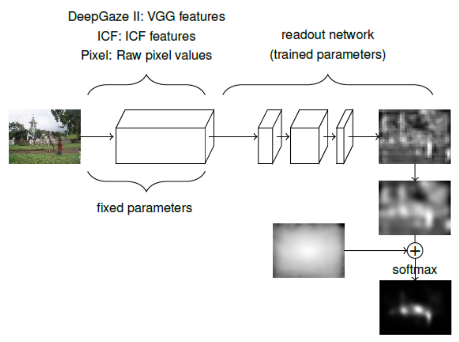

Figure 2. The architecture of the proposed models. Image is from paper [19].

level image features namely intensity and intensity contrast are extracted from images. I will not elaborate on the technical details of ICF implementation as I did not have enough time to implement it along with DeepGaze II.

3.2. Computing the log density distribution

The output of the readout network, referred to as $O(x, y)$ by the authors, is then convolved with a Gaussian to regularize the predictions as shown in equation 1.

$$S(x, y) = O(x, y) * G_\sigma \tag{1}$$

This gaussian filter has a sigma (standard deviation) and a kernel size. In my implementation I assigned a sigma of 7, and a kernel size of 21 for my initialization. Then during training, we can learn the best kernel size by considering it as one of the learnable parameters. However, we should note that by increasing the number of parameters we increase the training time.

As discussed in [22] humans fixations favor the center parts of images which is dataset-dependent. To obtain an unbiased image-based prediction, the authors subtract the log density of this center prior, referred to as $p_{baseline}$, from their predictions S, as shown in equation 3.3. In their paper they state that they add prior log density to the predictions, because they compute the negative log density of the prior. In my implementation code, I have computed the center prior for both training and evaluation datasets, using pysaliency library that is created by the authors. (by printing the values of center bias density, I noticed that they are all negative.)

$$S'(x, y) = S(x, y) - \log p_{baseline}(x, y) \tag{2}$$

According to the bayes rule, we in fact would like to obtain the likelihood from the multiplication of prior by likelihood. In other words, we at first have a biased estimate ($likelihood * prior$) and then divide out the prior to obtain an unbiased estimate ($likelihood$). But here as we are computing everything in the log scale, instead of division we have subtraction.

This center bias distribution is referred to as $p_{baseline}$ as it can predict many fixations correctly, and outperform some early saliency models. Therefore, it is used as a baseline model in MIT saliency benchmark. To compute the center$p_{baseline}$, the authors use a Gaussian Kernel density estimate over all fixations from the dataset. In dataset subsection I will display the center bias density for the two datasets MIT1003 and SALICON.

In the paper, the authors use a softmax function to convert the predictions to a probability distributions, as shown in equation 3.

$$p(x, y) = \frac{\exp(s'(x, y))}{\sum_{x,y} \exp(s'(x, y))} \tag{3}$$

However, due to some technical difficulties (encountering undefined/nan values) I used normalization shown in equation 4, which was also used by [15].

$$p(x, y) = \frac{s'(x, y) - \min[s'(x, y)]}{\max[s'(x, y) - \min[s'(x, y)]]} \tag{4}$$

3.3. Training

The objective function that we try to maximize during training of DeepGaze II is the log likelihood. However to use gradient descent,we need to minimize a loss function; thus we define the negative log likelihood as our loss function.

in [19], the authors find the values of their log density predictions (before softmax) which correspond to the location of the ground truth fixation points in each image, and sum over those values. They aim to maximize their predicted log probabilities in the place of ground truth fixations, for each image. The loss function that the authors use is shown in equation 3.3.

$$Loss = -\frac{1}{N} \sum_{i}^{N} \log p(x_i, y_i | I_i) \tag{5}$$

Where $\log p(x, y|I)$ is the log density (log likelihood) prediction of the model over pixel coordinates x and y for image I, i is the index of fixation in dataset, (x_i, y_i) is the coordinates of fixation location, and I_i refers to the image containing that fixation.

Note that the predicted log density $\log p(x, y|I)$ was previously denoted as $S'(x, y)$ in equation . By substituting in , we will obtain 10.

$$Loss = -\frac{1}{N}\sum_{i}^{N} \log \hat{p}(x_i, y_i|I_i) - \log p_{baseline}(x_i, y_i) \tag{6}$$

Where $\log \hat{p}(x_i, y_i|I_i)$ is the biased log density predictions previously referred to as $S(x, y)$. Also note that center bias density is image independent as is computed over all images in the whole dataset, hence we dont have the conditional notation on I_i. We can relate the above formula to information theory. We previously discussed that entropy is the average number of bits required to describe a random variable, and can be obtained using equation 7.

$$H(x) = -\sum_{x} p(x) \log p(x) = -E_p[\log p(x)] \tag{7}$$

Then, by estimating the expectation with average, we we can write the entropy of image I_i as:

$$H(x, y|I) \approx -\frac{1}{N}\sum_{i}^{N} \log \hat{p}(x_i, y_i|I_i) \tag{8}$$

$$H(x, y) \approx -\frac{1}{N}\sum_{i}^{N} \log p_{baseline}(x_i, y_i) \tag{9}$$

The above equation will converge to the true value of entropy in the limit as the number of fixations increases. By substituting above values in equation , we will have:

$$\begin{aligned} Loss = -\frac{1}{N}\sum_{i}^{N} \log \hat{p}(x_i, y_i|I_i) - \log p_{baseline}(x_i, y_i) \\ = H(x, y|I) - H(x, y) = -IG(\hat{p}||p_{baseline}) \end{aligned} \tag{10}$$

Where IG is the information gain and calculates is the reduction in entropy after a transformation. Information gain has the exact same formula as mutual information, but it has a different interpretation. Mutual information calculates the statistical dependence between two variables, and is a measure of the amount of information that one random variable contains about another; while information gain indicates how much information we gain by transforming the data in a certain way. Here our transformation is constraining the predictions of fixations to one image, and "information gain tells us what the model knows about the data beyond a given baseline model for which we use the image-independent center bias, expressed in bits / fixation" [19]

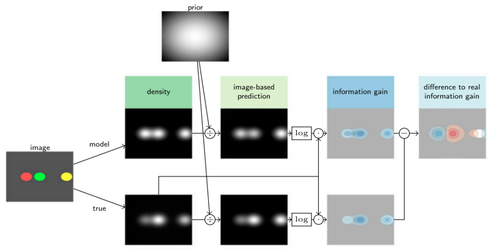

Figure 3. KL divergence loss computation pipeline.Image is taken from [17].

Therefore as seen in equation 10, by minimizing the loss we are trying to maximize the information we know about the data beyond the center bias; or in other words we try to maximize the image-dependent information.

Unfortunately, I failed to compute the aforementioned loss function in practice, due to running out of RAM memory in google colab. I used authors' pysaliency library to extract the fixations locations but then could not calculate loss on the large fixation arrays due to limited RAM (even with a batch size of 1 and using tensorflow efficient iterator functions). Hence, I used KL-divergence between the ground truth density(saliency) map and predicted density map as my loss function during training, as displayed in equation 11.

$$KL(q||p) = \sum_{x} q(x) \log \frac{p(x)}{q(x)} \tag{11}$$

Where q(x) is the grand-truth density(saliency) map, and p(x) is the predicted density map. Where both the ground-truth and predicted density maps are divided by the center baseline prior. This loss function is very similar to the previous loss function, with a slight difference that we compute kl divergence over all pixels of the predicted map and ground truth map, instead of finding the values of predicted maps only in fixation locations.

This new loss function is also related to information gain, which can be proved using the same procedure we took previously. The pipeline for computing this new loss function is provided in authors' previous paper [17], and is displayed in figure 3.

During training, we aim to minimize this loss function with respect to parameters of the read out network and the kernel size of the Gaussian which was used for regularizing the output of readout network. We do not train the parameters of VGG19. The authors use SFO (Sum of Functions Optimizer) to optimize their gradient descent, however I used Adam optimizer.

The approach that authors use for training the readout network is composed of a pre-training stage and a fine-tuning stage. During pretraining, the readout network parameters are trained from scratch on SALICON dataset, and validated on MIT1003 dataset to determine when to stop the training process. MIT1003 is used as the validation here because it is more similar to the final test data, MIT300, in MIT saliency benchmark. I also once validated the model using SALICON validation data, instead of MIT1003. I will present the results of both cases in the results section. Due to limited gpu allocation in colab, I trained the model for 15 epochs with a batch size of 1, and saved the model with highest performance (lowest KL-loss) among all training epochs. The authors train their model for minimum of 20 epochs and maximum of 800 epochs and stop training when the model starts to overfit the data.

After the pre-training stage is over, the authors cross validated the model over MIT1003 dataset. Meaning that they split the dataset images into 10 parts of equal size. Then ten models are trained starting from the result of the pretraining, each one using 8 of the 10 parts for training, one part for validation and one held-out part for testing. This final fine tuning stage was done by the authors to familiarize the network with the final test model. However, I did not implement fine-tuning stage due to time constraint and limited memory resources.

3.4. Evaluation

Many different metrics have been proposed to evaluate the saliency models performance, such as: Area under curve type metrics (AUC-type), Linear correlation coefficient (CC), Normalized scanpath saliency (NSS), Similarity (SIM), Information gain (IG), and Kullback-Leibler (KL) divergence. The authors additionally use another evaluation metric termed information gain explained (IGE). “This relates the model’s performance to the performance of a gold standard model that predicts one subject’s fixations for a given image from the fixations of all other subjects using a Gaussian kernel density estimate.” [19] The formula for this metric is as follows:

$$IGE = \frac{IG(\hat{p}||p_{baseline})}{IG(\hat{p}_{gold}||p_{baseline})} \tag{12}$$

where p_{gold} is the density of the gold standard model, and information gain can be computed using equation 10.

In my implementation, I could not compute IGE for not having computed the loss function in equation 10. Therefore, I use KL-divergence loss as my evaluation metric.

3.5. Dataset

In this project, two eye-tracking datasets namely, SALICON and MIT1003 were used. Each of these datasets has three types of data, first the images (stimuli) that are free-viewed by observers, second the fixation data, which contain the image location of the raw gaze points [x,y], the timestamp (millisecond) of each raw gaze point, and the fixation points, [x,y]. Fixation data are normally stored as matlab files. Third, fixation density maps, which present the normalized ground truth probability density over each image. These density maps are also blurred by a gaussian filter. In calculating KL-divergence loss, I use these probability density maps as my ground truth distribution.

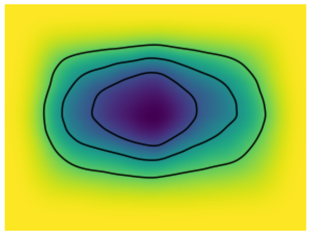

Figure 4. The center bias density over images of SALICON dataset.

I will elaborate more on each of these datasets in the following.

3.5.1 SALICON dataset

SALICON dataset [10] consists of 10000 training and 5000 validation images. The corresponding mouse tracking maps for each image were acquired from 16 observers during free-viewing of images for 5 seconds. Hence, instead of using an eye tracker to record observers’ viewing behaviors, a general-purpose mouse was used by the observers. Then, the mouse trajectories from different viewers were aggregated. To acquire a more eye-like fixations, they deployed a new algorithm based on cluster analysis (Cluster Fix).[13] All stimuli are of size 480 by 640 pixels (height by width), and were down-sampled by a factor of 2 (240x320) before being used as an input to the model.

To extract the fixations data from matlab m-files for different observers and overlay them on dataset images, I used the scripts of pysaliency library, created by authors of the paper [19]. This was a challenging task in google colab platform, as MATLAB is not installed in colab environment, and I needed to use “Octave”, as an alternative software. I have provided a readme file in my project folder with detailed instructions on how to install octave and all required statistical packages in colab environment to be able to extract fixation data. A display of center bias density in SALICON dataset (computed with the help of pysaliency library scripts) is shown in figure 4.

I also display gold standard kernel density estimate for one image in salicon dataset along with its ground truth fixations in figure 5.

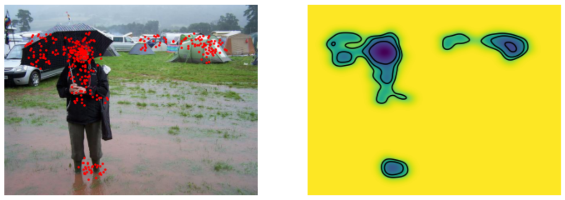
Figure 5. The left figure shows an stimulus in SALICON dataset and its ground truth fixation locations. The right image shows the gold standard kernel density estimate for the same stimuli.

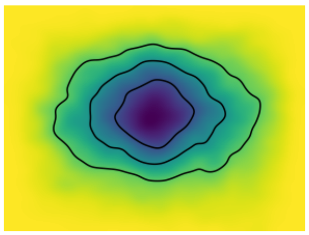
Figure 6. The center bias density over images of SALICON dataset.

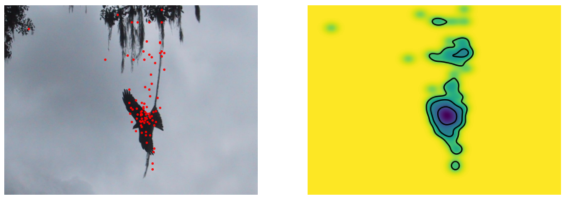
Figure 7. The left figure shows an stimulus in MIT1003 dataset and its ground truth fixation locations. The right image shows the gold standard kernel density estimate for the same stimuli.

3.5.2 MIT1003 dataset

This dataset consists of 1003 natural images. The corresponding fixation maps were acquired from 15 observers during free-viewing of images for 3 seconds. The largest dimension found among all stimuli of this dataset is 1024 pixels. Although images of this dataset have various shapes, a large number of them are either of size 1024 × 768 or 768 × 1024. Hence, as suggested by authors, in my implementation I also resize all images to either 1024 × 768 or 768 × 1024 depending on their aspect ratio, before downsampling by a factor of two. A display of center bias density in MIT1003 dataset is shown in figure 6.

I also display gold standard kernel density estimate for one image in MIT1003 dataset along with its ground truth fixations in figure 7.

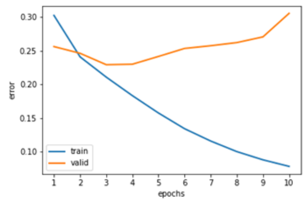

Figure 8. The training curve when using SALICON dataset as the training and validation sets.

4. Results

I performed two sets of experiments. First I trained and validated deep gaze 2 on SALICON training and validation sets respectively, and then tested the model predictions on MIT1003 dataset.(note that I do not perform the final cross validation step that authors perform.). My training curve for this experiment is shown in figure 8. As displayed in the figure, model overfits really fast. The lowest loss is achieved in the third epoch, with training loss of 0.210874, and validation loss of 0.229348. Each epoch of training takes approximately 50 minutes , and validation takes 3 minutes.

Second, I used SALICON dataset as the training set and MIT1003 as the validation set. Then I tested the model on MIT1003 dataset. It is not a common practice in deep learning literature to have the same validation and test sets, as it is probable that the model become familiarized with the test data during the validation, and our evaluations would be biased. However, in the paper the authors use this method, and report their performance both on MIT1003 dataset (which is also their validation data) and MIT300 held-out test set (used in MIT sliency benchmark). Due to time constraint, I did not test model's performance on MIT300 held-out test set. The reason the authors use MIT1003 as the validation data instead of SALICON validation set, is to save (pick) the trained model that has better performance on a dataset similar to the test set. My training curve for this experiment is shown in figure 9. As displayed in the figure, model performance plateaus after epoch 5. In this case, the lowest training loss was 0.175 and the lowest validation loss was 0.985. Each epoch of training takes approximately 50 minutes , and validation takes 1.5 minutes. (Here the validation time decreases as we are validating over 1003 images; while previously we validated the model on 5000 images of SALICON validation data.)

As it is clear from the second training curve in in figure 9, the model performs poorly on MIT1003 dataset. Its loss is never less than 1. Therefore, it is expected that without doing a fine-tuning (cross validation) step, the results on MIT1003 would not be that desirable. However there were still some cases which network performed quite well on them. I have included one example of successful predic-

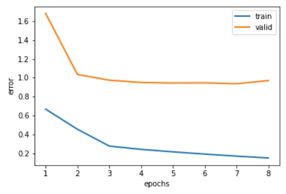

Figure 9. The training curve when using SALICON dataset as the training and MIT1003 as the validation set.

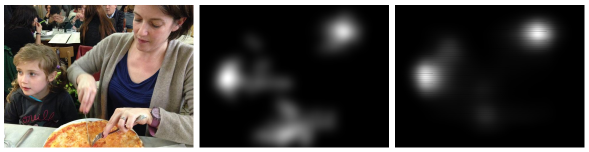

Figure 10. The leftmost image is an stimuli from MIT1003. The middle image is the ground truth density map of the image, and the rightmost image is the predicted density map by DeepGaze II. The fixations are mostly located on the faces, and also the place where the woman is looking i.e. her food. The network correctly predicts the fixations on the faces; however, it fails to do higher level gaze-following and predict fixations on the food.

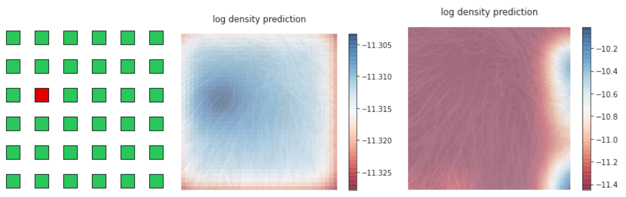

Figure 11. The leftmost image is an arbitrary stimuli (not part of any of the aforementioned datasets) with low-level features. The middle image is the log density prediction by authors' checkpointed ICF model, and the rightmost image is the log density prediction by authors' checkpointed DeepGaze II model. It is clear that DeepGaze II model fails to predict the low-level pop-out effect; while ICF model succeeds in predicting the salient part.

tion of the network in figure 10.

One of the weird results that I obtained was that without Gaussian blurring the networks sometimes gains a higher performance. This is unexpected as the ground truth density maps are also blurred by a Gaussian. Therefore it should be more accurate to match a blurred predicted density map with a blurred ground truth map. This counter-intuitive result might be due to networks failure in learning the correct kernel size for the Gaussian, or a fallacy in my implementation of Gaussian blurring.

I also loaded the authors checkpointed DeepGaze II and ICF models in a separate notebook and checked their performance on two images, one image has a low-level pop-out effect and directs eyes via low-level contrast; and the other contains a human face which directs fixations via high-level features. The results are shown in figures 11 and 12.

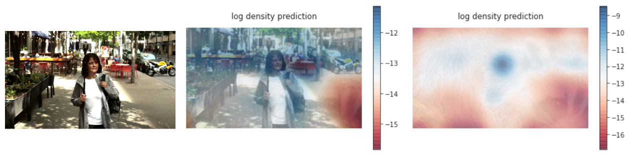

Figure 12. The leftmost image is an arbitrary stimuli (not part of any of the aforementioned datasets) with high-level features. The middle image is the log density prediction by authors' checkpointed ICF model, and the rightmost image is the log density prediction by authors' checkpointed DeepGaze II model. In this case, ICF fails to detect the face of the woman as the salient region while DeepGaze II succeeds in predicting it.

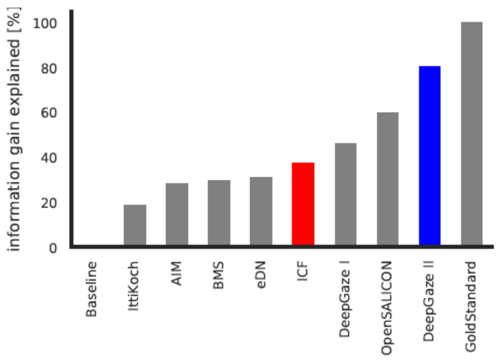

Figure 13. Ranking of the models according to information gain explained. DeepGaze II and ICF are marked by the colored bars. All models to the right of ICF use pre-trained deep features. Image and caption credit [19].

In the paper, the authors also compare the performance of DeepGaze II, and ICF models with other saliency models: Itti and Koch, AIM [1], eDN, DeepGaze I, OpenSALICON [23], and also computed baseline models: Centerbias, and Gold Standard. A ranking of these models based on information gain explained can be seen in figure 13.

In general, DeepGaze II can correctly predict fixations that occur due to high-level features such as text and faces, in accordance with its status as a far more powerful model than ICF (more parameters with pre-trained features). "On the MIT1003 dataset as a whole, authors find that there is a substantial subset of images (94 of 1003) for which the ICF model produces better predictions than DeepGaze II." [19].

Because DeepGaze II should also have access to low-level features, this result suggests that DeepGaze II may be giving a low weight to the low-level features while predicting fixations.

5. Discussion and Conclusion

In the implemented paper, the authors introduce two saliency models DeepGaze II, which predicts fixations due

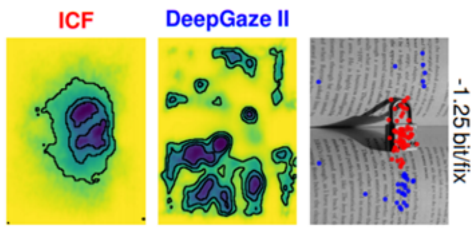

Figure 14. An example where ICF model outperforms DeepGaze II. DeepGaze II incorrectly predicts the texts are fixated by subjects; while ICF correctly predicts the lock is fixated by observers. Image is taken from paper [19].

to high-level features in an image and ICF which predicts fixations caused by low level intensity and contrast features. These two models have the same readout architecture on top of their feature extraction block. DeepGaze II uses transfer learning from the VGG-19 deep neural network to extract high-level features, while ICF only extracts local contrast and intensity from images.

DeepGaze II is a powerful saliency model, it is ranked 4th in MIT saliency benchmark with information gain = 0.9247, AUC= 0.8733, sAUC= 0.7759, NSS= 2.3371, CC= 0.7703, kl-Div=0.4239 and SIM= 0.6636. While ICF is ranked 14th in this benchmark. This indicates the importance of high-level features in directing humans' fixations. On the other hand, DeepGaze II performs poorly in cases where fixations are directed by low-level image features; which are not insignificant.

My initial idea for solving this issue was to consider explicitly modelling low-level features as a kind of prior and add this prior to the predictions of the network. Unfortunately this idea is not feasible as low-level features in images are not shared among all images of the dataset in contrast to center bias densities that are shared among all images of the dataset. In other words, each image has a different density of low-level features over its pixels. Computing these separate priors for each image, is not better than having a separate low-level ICF model.

A more plausible idea would be to consider adding more low-level driven fixations among the training set of DeepGaze II. Because this model as many other deep learning models, is sensitive to its training set, and suffers from the ability to generalize to new datasets. I would like to continue my research on investigating new ways to address this problem in deep learning-based saliency models such as DeepGaze II.

Also, DeepGaze II fails to correctly predict fixations when high-level features (texts) are present in the image but are not fixated. One example of this case is displayed in figure 14.

DeepGaze II also fails in other scenarios, such as occluded objects, faint images, and small texts/objects. Furthermore, it does not capture higher level concepts such as gaze following, i.e. when a group of people are looking at an object in an image, we unconsciously fixate at that object; while DeepGaze II fails at this, probably due to low exposure to these kinds of data in its training set.

Additionally, most contemporary saliency models, such as DeepGaze II do not consider the temporal aspect of fixations. They do not contain any information regarding how long each point was fixated in the image, and in what order they were fixated. These are useful information that should be considered in visual attention studies.

Besides, it is possible to extend the application of DeepGaze II model to predict fixations during a task-based scenario, such as when searching for an object in a scene.

To conclude, DeepGaze II was a great breakthrough in predicting fixations by using transfer learning in neural networks. However, this model and its successors still have weaknesses in modeling human's fixation behavior, which are remained to be addressed in future research.

References

[1]

[2] A. Borji, D. N. Sihite, and L. Itti. Probabilistic learning of task-specific visual attention. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, Rhode Island*, pages 1–8, Jun 2012.

[3] Zoya Bylinskii, Tilke Judd, Ali Borji, Laurent Itti, Frédo Durand, Aude Oliva, and Antonio Torralba. Mit saliency benchmark.

[4] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Richard Droste, Jianbo Jiao, and J. Alison Noble. Unified image and video saliency modeling. *Lecture Notes in Computer Science*, page 419–435, 2020.

[6] X. Huang, C. Shen, X. Boix, and Q. Zhao. Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 262–270, 2015.

[7] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:1254–1259, 2009.

[8] Saumya Jetley, N. Murray, and E. Vig. End-to-end saliency mapping via probability distribution prediction. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5753–5761, 2016.

[9] Sen Jia and Neil D. B. Bruce. Eml-net:an expandable multi-layer network for saliency prediction, 2019.

[10] M. Jiang, S. Huang, J. Duan, and Q. Zhao. Salicon: Saliency in context. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1072–1080, 2015.

[11] Tilke Judd, Krista A. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. *2009 IEEE 12th*

International Conference on Computer Vision, pages 2106–2113, 2009.

[12] C Koch and S Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. *Human neurobiology*, 4(4):219—227, 1985.

[13] Seth D. König and E. Buffalo. A nonparametric method for detecting fixations and saccades using cluster analysis: Removing the need for arbitrary thresholds. *Journal of Neuroscience Methods*, 227:121–131, 2014.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[15] Alexander Kroner, Mario Senden, Kurt Driessens, and Rainer Goebel. Contextual encoder–decoder network for visual saliency prediction. *Neural Networks*, 129:261 – 270, 2020.

[16] S. S. S. Kruthiventi, K. Ayush, and R. V. Babu. Deepfix: A fully convolutional neural network for predicting human eye fixations. *IEEE Transactions on Image Processing*, 26(9):4446–4456, 2017.

[17] Matthias Kümmerer, Thomas S. A. Wallis, and M. Bethge. Information-theoretic model comparison unifies saliency metrics. *Proceedings of the National Academy of Sciences*, 112:16054 – 16059, 2015.

[18] Matthias Kummerer, Thomas S. A. Wallis, and Matthias Bethge. Saliency benchmarking made easy: Separating models, maps and metrics. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[19] Matthias Kümmerer, Thomas S. A. Wallis, Leon A. Gatys, and M. Bethge. Understanding low- and high-level contributions to fixation prediction. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4799–4808, 2017.

[20] K. Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[21] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[22] B. Tatler. The central fixation bias in scene viewing: selecting an optimal viewing position independently of motor biases and image feature distributions. *Journal of vision*, 7 14:4.1–17, 2007.

[23] Christopher Thomas. Opensalicon: An open source implementation of the salicon saliency model. *ArXiv*, abs/1606.00110, 2016.

[24] Anne M. Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12(1):97 – 136, 1980.

[25] E. Vig, M. Dorr, and D. Cox. Large-scale optimization of hierarchical features for saliency prediction in natural images. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2798–2805, 2014.