COMSM0045: Predicting Eye Fixation using Convolutional Neural Networks

David Hao Yan
Faculty of Science and Engineering
University of Bristol
lj21689@bristol.ac.uk, 2101428
https://github.com/davidhaoyan/ADL-VisualSaliency

I. Introduction

The visual saliency problem bridges the fields of computer vision and cognitive science. Visual saliency refers to the distinct subjective perceptual quality which makes an aspect of an image stand out amongst its surroundings. It is believed to be composed of bottom-up visual saliency as well as top-down visual saliency. Bottom-up visual saliency refers to low-level, stimulus-driven features such as intensity, colour, or orientation which can be strongly affected or even overridden by top-down visual saliency which refers to high-level, user-driven factors such as expectation, knowledge or goals.

Liu et al. [1] proposed a multi-resolution convolutional neural network (Mr-CNN) capable of predicting eye fixations by learning these two types of visual features. The Mr-CNN is trained on image crops that are centered on either fixation or non-fixation points across multiple resolutions. Top-down visual features can be learned in higher layers whilst bottom-up visual features can be produced by considering the information across multiple resolutions together. Lastly, the balance between these two can be learned in the final logistic regression layer.

This paper aims to replicate the architecture and results produced by the paper. Extensions to the paper such as data augmentation and batch normalization are also explored.

II. Related Works

A. GAN Approach

Pan et al. [2] introduced a novel data-driven metric-based saliency prediction method called SalGAN. As a generative adversarial network (GAN), it is trained using an adversarial loss function and is composed of two competing networks. One, the generator, predicts saliency maps in an attempt to fool the other network, the discriminator, which learns to distinguish between the predicted saliency maps and ground truth. The two networks are trained in tandem and by being able to generate predicted saliency maps which are indistinguishable from the ground truth, SalGAN is able to generate saliency maps which resemble the actual saliency maps. Pan et al. [2] compare the performance of SalGAN with other state-of-the-art solutions and highlights the significance of introducing adversarial training on improving the quality of predictions, as well as being able to capture high-salient regions that may be missed by other models.

Despite its impressive performance, SalGAN is prone to the inherent challenges associated with training GANs. For instance, non-convergence arises because the generator and discriminator do not have a deterministic convergence path. In this case, training may not reach an equilibrium point where both the generator and discriminator are adequately trained and for this reason, it is common for GANs to oscillate, with neither network making consistent improvements. Another common pitfall is mode collapse, in which the generator may produce a trivial solution that appears optimal but fails to represent the features of the target distribution.

B. Attention Is All You Need

Liu et al. [3] also proposed a new model based on the state-of-the-art transformer architecture, the Visual Saliency Transformer (VST). The model provides an alternative to existing saliency detection methods which rely heavily on CNN-based architectures by providing a convolution-free sequence-to-sequence approach to predicting saliency. Characteristic of a transformer, the network is able to model long-range dependencies effectively. The paper highlights the limitations of CNNs when it comes to learning global contexts and the importance of being able to do so. By using self-attention mechanisms, the VST model demonstrates a significant improvement compared to existing methods.

III. Dataset

Liu et al. [1] trained and evaluated their model on four widely used datasets: MIT, Toronto, Cerf and NUSEF. The replication was based solely on the MIT dataset which contains 1003 images collected from Flickr and LabelMe datasets, with resolutions ranging from 405x1024 to 1024x1024 pixels. It is the largest dataset amongst the four and consists of 779 landscapes, 228 portraits and several synthetic images. Each image has a corresponding ground truth fixation point image, as well as a fixation map image which is produced by applying a Gaussian blur to the fixation points.

Of the 1003 images in the MIT dataset, 100 images are held aside for validation, another 100 for testing, and the remaining 803 are used for training. During training, 10 crops centered around non-fixation points and 20 crops centered around fixation points are sampled - a fixation point is labelled by a saliency value above 0.9 and conversely, a non-fixation

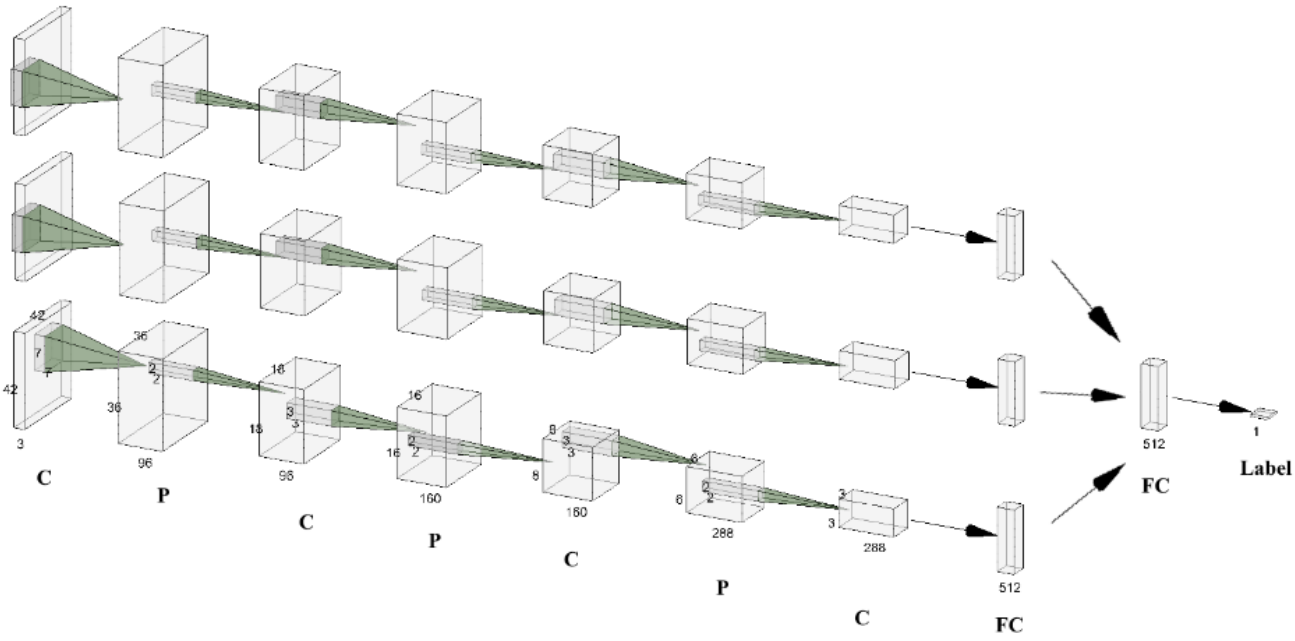

Fig. 1: Network architecture of the model proposed by Liu et al. [1]. There are three streams for the three scales ie. 150x150, 250x250 and 400x400. From these three resolutions, 42x42 sized image regions are cropped with the same centre locations and passed as input to the network. Convolutional, pooling and fully-connected layers are represented by C, P and FC respectively. Note that the parameters are shared for the three C layers between each stream. The three FC layers at the end of each stream are concatenated and fed into the last FC layer which uses logistic regression to output a visual saliency score - this can be thresholded into a binary classification (fixation or non-fixation point).

point is labelled by a saliency value below 0.1. This amounts to a total of 803*30 = 24090 different crops. Note that each crop has three resolutions as well. During evaluation, 50x50 linearly interpolated centres are extracted and classified as fixation or non-fixation points, the raw saliency map is produced by upscaling to the image size, which were then processed by thresholding and applying a Gaussian blur.

IV. CNN Architecture

A. Conventional CNN

Conventional convolutional neural networks (CNNs) are composed of alternating convolution layers and pooling layers, typically max-pooling. They are especially effective when it comes to dealing with grid-like data such as images, because the learnable parameters are confined to the kernels (or filters) applied at each layer. In this sense, the learnable parameters are shared spatially, which enables the model to learn features that are translation invariant, such as edges, textures and shapes.

B. Mr-CNN

My model replicates the model proposed by Liu et al. [1], a multi-resolution CNN (Mr-CNN) which extends the capability of a conventional CNN by simultaneously processing input data at multiple resolutions. This allows the network to extract features that are invariant to the resolution, which proves to be significant when it comes to visual saliency problems - a bright, red object will stand out regardless of the resolution.

The Mr-CNN consists of three input streams: 150x150, 250x250 and 400x400, from which 42x42 sized crops with the same centre location are extracted (paired with their corresponding labels: 1=fixation point; 0=non-fixation point).

Each stream is composed of three shared C layers (convolution), three P layers (pooling), and a FC layer (fully-connected). The three streams are concatenated and then followed by one more FC layer, then one logistic regression layer which outputs a visual saliency score from which binary classification of the point can be performed. The three C layers are shared between the three streams to allow the network to learn scale-invariant features. The properties of the three C layers are as follows:

1) 97 kernels: size=7x7, stride=1, padding=0
2) 160 kernels: size=3x3, stride=1, padding=0
3) 288 kernels: size=3x3, stride=1, padding=0

The P layers implement max-pooling with filter size 2x2 and each FC layer is composed of 512 neurons, and 1 neuron in the output layer.

V. Implementation Details

A. Optimiser and Loss Function

The model implements stochastic gradient descent as the optimizer used for back propagation and binary cross entropy between the predicted labels and the ground truth labels in the lasy layer as the loss function.

B. Hyperparameters

In the original paper, the model was trained in 5,000 steps, where one mini-batch (stochastic gradient descent) is processed per step. The learning rate specified was 0.002, however after fine-tuning, I discovered an increase in performance with smaller learning rates trained for an increased number of steps. For this reason, my replication strays from the original implementation in regards to hyperparameters, namely the steps, learning-rate and batch-size.

Staying true to the original, the model implements a weight decay of 0.0002 and momentum linearly increasing from 0.9 to 0.99. Dropout with corruption probability of 0.5 was used in the original. However, I discovered adverse impacts to performance when implementing dropout for my model. Even with smaller corruption probabilities, training accuracy and validation AUC were hindered. I suspect that this may be because the original is trained on four datasets, whilst my model is only trained on one - the features that my model would be able to extract would not be as intricate and diverse compared to the original, and so culling nodes in the forward process may detract from the information learned. It is for this reason that I decided to omit dropout entirely.

Learning Rate	Test AUC
0.002	0.8395
0.001	0.8211
0.0005	0.8118
0.0001	**0.8588**

TABLE I: AUC against learning rate.

I experimented with four different learning rates and found that the lowest learning rate of 0.0001 performed the best. To adjust for a small learning rate, I opted to train the model for an increased number of steps and converged on 19,000 as the final number of steps trained for optimal performance.

To account for the increased number of training steps, I also opted to double the batch size from 256 in the original to 512 in my replication. My intention was to speed up the training time by making the most out of GPU parallel processing. However, this may have caused my model to under fit as each mini-batch was more generalized, shown in Fig. 6.

VI. Replicating Quantitative Results

MIT	AUC (non-shuffled)
Mr-CNN	0.8588

TABLE II: Replicated MrCNN Performance

The replicated model produced a non-shuffled AUC of 0.8588. This is considerably higher than the original result produced in the paper of 0.7190. The reason for the disparity is due to the difference in metric used. Whilst normal AUC was used for our evaluation metric, the original used shuffled AUC which tends to be more discriminative and is regarded as tighter, and more relevant visual saliency metric. A common pitfall of various saliency metrics is the centre bias, where a dense distribution around the centre tends to score highly due to the tendency for viewers to fixate on an image's centre [4]. Instead, with shuffled AUC, a centered Gaussian distribution which could outperform a well-designed system on most of its metrics [5], can now only achieve a shuffled AUC score of 0.5 [6].

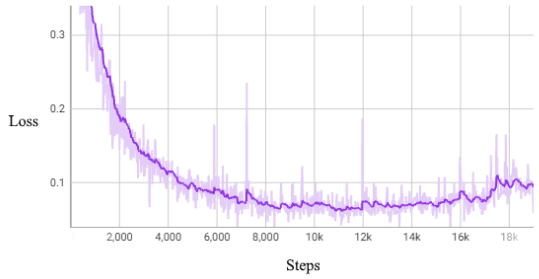

Fig. 2: Training loss against steps.

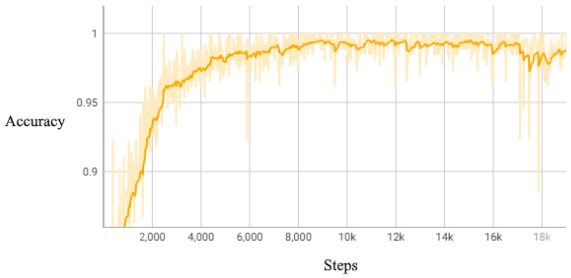

Fig. 3: Training accuracy against steps.

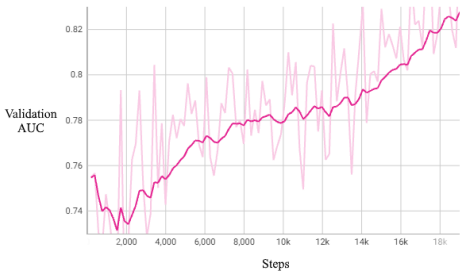

Fig. 4: Validation AUC against steps.

VII. Training Curves

The training curves [Fig. 2, 3, 4] show the loss, accuracy and validation AUC for my model over 19,000 steps. The bulk

of the learning is completed within 6000 steps, where you can see the loss and accuracy start to plateau. However, despite the fact the model loss and accuracy of the model is not increasing during training, the validation AUC continues to rise, leading to an overall better performance.

Towards the end, at around 17,000 steps, my model starts to overfit to the training data. This can be seen by the peak in the loss curve, and conversely, by the trough in the accuracy curve. This information guided me when it came to choosing a suitable number of steps for my model to train. My intention was to find a compromise which maximised the model's performance on the unseen validation data, whilst not overfitting to the training data. As a result, I chose to stop the training process at 19,000 steps.

VIII. Qualitative Results

Fig. 5 shows the qualitative results of two MrCNN models (trained with learning rate = 0.001 and 0.0001) after thresholding the raw saliency maps and applying a Gaussian blur filter. The model performs well against the first and second photo, picking up the sign of the shop and the three points of the swan. Interestingly, the model trained with learning rate 0.001 with a lower test AUC produced a more accurate saliency map, minimising false positive values, whilst the model trained with learning rate 0.0001 with a higher test AUC, despite also picking up the main saliency region,

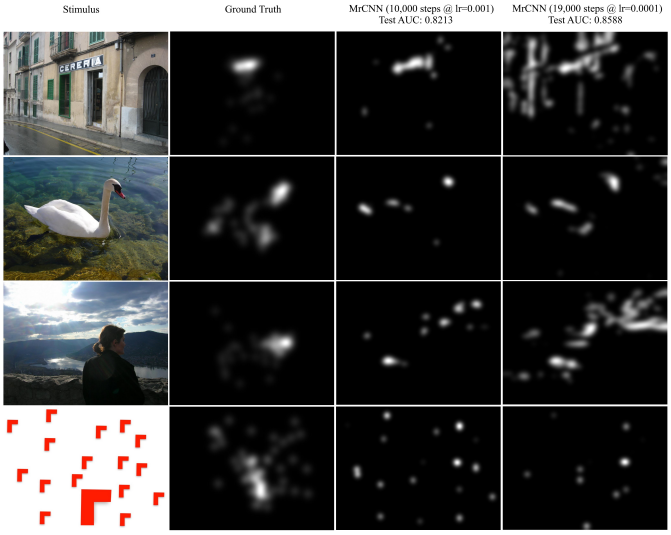

Fig. 5: Comparison of stimuli, ground truth saliency maps, MrCNN (10,000 steps @ lr=0.001; Test AUC=0.8211), MrCNN (19,000 steps @ lr=0.0001; Test AUC = 0.8588) respectively.

produced significantly more false positives. This reinforces the unreliability of AUC as a visual saliency metric.

Taking a look at the third image, the model seems to do well in predicting the bottom-up saliency features i.e. the sharp contrast from the glare of the sun emerging from behind the clouds as well as the reflection in the lake. The model falls short when it comes to incorporating more abstract top-down visual saliency features. For instance, the ground truth demonstrates that humans tend to fixate on the woman's head. Despite being a darker region of the photo, our conditioning and intuition tell us that the woman is the subject. In contrast, the model is unable to discern this and treats the subject as if it is part of the background, avoiding the region entirely.

Similarly in the fourth photo, from a bottom-up perspective, the uniform shapes are all equal - they have the same colour, the same edges, the same contrast etc. The ground truth demonstrates that humans tend to fixate on the larger object, but this proves to be difficult for the model, which treats each shape as if they were equal.

Fig. 6 shows the average distribution of generated prediction maps and ground truth maps. Centre bias can be seen very clearly in the mean ground truth map and it is captured to some extent by the model. However, the model tends to predict a lot more erratically, which drastically increases the false positive rate and this is reflected in the predicted maps which have a much lower precision.

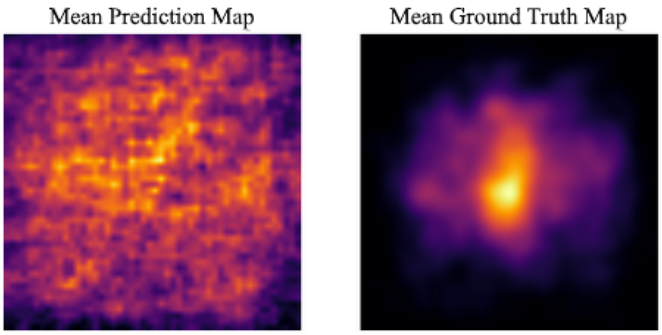

Fig. 6: Averaged prediction map compared with averaged ground truth map. AUC: 0.6786

IX. Improvements

In order to extend the model proposed by Liu et al. [1], I explored both batch normalization and data augmentation. In the original implementation, data augmentation is limited to horizontal flipping. In my implementation, the transforms are composed horizontal flipping, vertical flipping, color jittering and affine translations.

Feature	Test AUC
Batch Normalization	**0.8970**
Augment	0.8769
Baseline	0.8588

TABLE III: Test AUC of improvements, all models trained for 19,000 steps @ lr=0.0001

A. Data Augmentation

Data augmentation in deep learning refers to random transformations applied to the training sample in each epoch. In each epoch, the transformations for each image are unique, and this leads to better results through generalization.

Specifically, the random transforms composed of horizontal and vertical flips with a probability of 0.5, colour jitter with a brightness, contrast and saturation factor of 0.8 to 1.2, a hue factor of 0.9 to 1.1, and affine transformation of +- 1°, vertical and horizontal translation of +- 0.1*height and 0.1*width respectively, and scale factor from 0.9 to 1.1.

Fig. 7 demonstrates an increase in precision, only predicting true fixation points whereas the original model tends to overestimate fixation points, leading to many false positives.

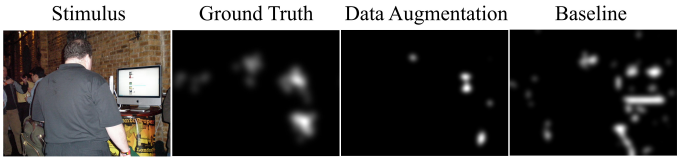

Fig. 7: Comparion of stimulus, ground truth saliency map, MrCNN (19,000 steps @ lr=0.0001 w/ data augmentation), MrCNN (19,000 steps @ lr=0.0001) respectively.

B. Batch Normalization

Batch normalization is a technique used to stabilize and speed up the training process of deep neural networks by normalizing the input to each layer [7]. It addresses a phenomenon known as internal covariate shift, where the distribution of each layer's input changes during training. By normalizing the layer inputs, it acts as a form of regularization which prevents overfitting.

Fig. 8 demonstrates how the model trained with batch normalization outperforms the base model which has failed to predict the true fixation points.

Batch normalization also allows you to use larger learning rates, and to achieve the same efficacy with fewer training steps [7]. This could be explored in the future.

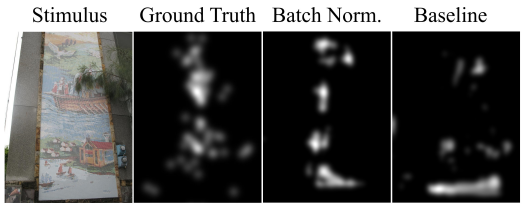

Fig. 8: Comparison of stimulus, ground truth saliency map, MrCNN (19,000 steps @ lr=0.0001 w/ batch normalization), MrCNN (19,000 steps @ lr=0.0001) respectively.

X. Conclusion and Future Work

In this paper, I presented a replication of the MrCNN model for visual saliency proposed by Liu et al. [1]. I demonstrated quantitatively and qualitatively the efficacy of such an architecture, as well as addressing the challenges the model may face. In particular, the model falls short when it comes to evaluating top-down semantics which may not be readily apparent from simply evaluating the bottom-up features i.e. intensity, colour and orientation.

Today, the problem of visual saliency has developed from static images to dynamic videos [8]. Extending the problem by a spatial dimension compounds the challenge of being able to detect the regions that humans will fixate on. In addition, the subjectiveness of top-down saliency features plays a crucial role in being able to accurately depict visual saliency. These human factors, which are based on expectation, knowledge or goals, are incredibly diverse and subject to the viewer. In this way, the two fields of computer vision and cognitive science that are connected by the visual saliency problem will likely progress together, with both fields working in tandem.

References

[1] Nian Liu et al. "Predicting Eye Fixations Using Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 362–370.

[2] Junting Pan et al. *SalGAN: Visual Saliency Prediction with Generative Adversarial Networks*. arXiv:1701.01081. 2017. URL: https://arxiv.org/abs/1701.01081.

[3] Nian Liu et al. *Visual Saliency Transformer*. arXiv:2104.12099. 2021. URL: https://arxiv.org/abs/2104.12099.

[4] Candace E. Peacock, Taylor R. Hayes, and John M. Henderson. "Center Bias Does Not Account for the Advantage of Meaning Over Salience in Attentional Guidance During Scene Viewing". In: *Frontiers in Psychology* 11 (2020). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2020.01877. URL: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2020.01877.

[5] Lingyun Zhang et al. "SUN: A Bayesian framework for saliency using natural statistics". In: *Journal of Vision* 8.7 (Dec. 2008), pp. 32–32. ISSN: 1534-7362. DOI: 10.1167/8.7.32. eprint: https://arvojournals.org/arvo/content_public/journal/jov/933536/jov-8-7-32.pdf. URL: https://doi.org/10.1167/8.7.32.

[6] Zoya Bylinskii et al. *What do different evaluation metrics tell us about saliency models?* 2017. arXiv: 1604.03605. URL: https://arxiv.org/abs/1604.03605.

[7] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167. URL: https://arxiv.org/abs/1502.03167.

[8] Andrey Moskalenko et al. "AIM 2024 Challenge on Video Saliency Prediction: Methods and Results". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2024.