A Multi-Resolution CNN for Visual Saliency Prediction

We agree that all members have contributed to this project (both code and report) in an approximately equal manner.

vn29614
Harleen Gulati

qz21935
Deepkumar Pawar

tm21064
Edward Su

I. Introduction

Computational visual saliency models predict where the human eye attends to in an image using computer vision techniques [1]. Intuitively, certain regions in an image may have a higher probability of attention assigned to them as they contain features more salient upon first glance by a person, such as contrast, colour, and complexity.

Traditional approaches for saliency modelling follow three steps: early feature extraction, feature contrast inference and contrast integration [2]. However, these approaches only consider bottom-up visual saliency (i.e. referring to colour, contrast) and neglects top down factors such as objects, actions and events (e.g. faces, vehicles) [2]. It is believed that upon looking at an image, the eye is directed by bottom-up visual saliency and then by top down factors [3]. Thus, models were introduced to combine both bottom-up visual saliency and top down factors to predict eye fixation. These models showed better performance compared to relying solely on bottom-up visual saliency [2]. However, these solutions only considered a small number of hand-tuned factors [2]. For instance in some models, only human faces were recognised where animals could also be a plausible top down factor.

To counter this, some machine learning methods are integrated into some models to learn bottom-up saliency and top down factors, such as individual component analysis [4]. However, it is still hard to learn high-level information of complex images due to the limited representational capacity of their shallow networks [2].

"Predicting Eye Fixations using Convolutional Neural Networks" by Liu et al. [2] proposes a novel computational model based on a multi-resolution convolutional neural network (Mr-CNN) which targets the problems above: the Mr-CNN simultaneously learns early features, bottom-up saliency, top down factors and their integration from raw image data to enhance the current methods mentioned above in predicting eye fixation. The model Liu proposes learns information with hierarchically increasing complexity in the convolutional layers. This approach is automated, hence stripping away the need for handcrafted features, human designed mathematical tools and predefined top down factors, thus dealing with the limitations mentioned earlier. We replicate the architecture implemented in this paper.

II. Related Work

A. Multiscale Deep CNN Features

In 2016, Guanbin et al. proposed "Visual Saliency Detection Based on Multiscale Deep CNN Features" [5]. This proposal targets the same problem from the original paper. Guanbin et al. follow a similar methodology by introducing a network which has fully connected layers on top of a convolutional neural network.

The implemented network requires the input image to be decomposed into a set of non-overlapping regions each of which has almost uniform saliency values internally (i.e., for each region, the pixels within the region have similar saliency values). Guanbin et al. applies their model to three nested and increasingly larger rectangular windows which are: the bounding box of the region of interest, the bounding box of the immediate neighbouring region, and the entire image respectively.

Guanbin et al. use three convolutional neural network streams for each of their respective scale region inputs, which are responsible for multi-scale feature extraction. Each stream has five convolutional layers. The final convolutional layer is followed by a fully connected layer for each stream. The fully connected layers are then fused together to learn multi-scale features. In the model described, the fused fully connected layer is followed by another fully connected layer aiming to learn deep contrast features (i.e. complex and abstract patterns). The output of this layer is concatenated with handcrafted low-level features to boost saliency detection performance. The result of this concatenation is fed into a random forest regressor which holds the purpose of mapping the feature vector of each region to a saliency score.

The model in Guanbin's paper was not trained on the MIT1003 dataset [6] and hence we could not offer a direct comparison to the results our paper produced, as our paper was trained on this dataset. The model was tested on the DUT-OMRON dataset [7]. Guanbin et al. observed this dataset was difficult in the past for many to achieve high accuracy on, as the saliency annotations were controversial amongst human observers. Despite this, their model performed on this dataset with an AUC of 93.5%.

B. Contextual Encoder-Decoder Network

Kroner et al. proposed a "Contextual Encoder-Decoder Network for Visual Saliency Prediction" [8] in September 2020 to solve the same problem of predicting visual saliency. We discuss the approach taken by Kroner, the architecture utilized and the results achieved.

Kroner et al. focus on the extraction of high-level visual features (e.g. objects, faces) at multiple spatial scales, and this information is then augmented with context to predict visual saliency. He argues this is a focus of the paper as existing models do not incorporate such a mechanism explicitly. For example, Liu et al. focus on the extraction and augmentation of top-down factors and low-level features at multiple spatial scales to predict visual saliency. However they do not augment the context of an image as Kroner et al. do. Here, augmenting the context of an image refers to adding additional information about the broader scene of the image (e.g., positioning of objects relative to each other, whether the image is taken indoors or outdoors) to predict visual saliency.

Kroner et al. use an encoder-decoder architecture for their prediction. The encoder is a convolutional neural network, which is then followed by the ASPP module. This module involves three convolutional layers with kernel sizes 3 x 3 and dilation rates 4, 8 and 12 in parallel. Each convolutional layer learns the global context of the image. A larger dilation rate implies a larger receptive field and hence more global context being learnt. In parallel, there is also a 1

x 1 convolutional layer which non-linearly combines existing feature maps as opposed to learning new information, and this convolution layer has no dilation. The three former convolutional layers have the same weights to learn. The information learnt from the parallel convolutional layers is then concatenated to understand multi-scale features i.e. both local details and global context. Finally, the decoder reconstructs the saliency map. This is achieved by upsampling the information learnt by the ASPP (i.e. reconstructing the information to the original image size) and concatenating the information learnt by the ASPP module (i.e. global context) and the information learnt by the encoder (i.e. high-level visual features).

The architecture was trained on the SALICON dataset [9] and the weights learnt were used as pretrained weights for training on other datasets, such as the MIT1003 dataset [6] (whereas Liu et al. trained from scratch on the MIT1003 dataset). For the MIT1003 dataset, Kroner et al. rescaled all the images and padded them to be 360 x 360 pixels. Unlike Liu et al., they did not perform any data augmentation even though they briefly mentioned it as an avenue for future work. For the shuffled AUC metric, Kroner et al. achieved 72%, compared to Liu et al. achieving 71.9%.

C. Deeply-Supervised Recurrent Convolutional Neural Network

In 2016, Tang et al. proposed a "Deeply-Supervised Recurrent Convolutional Neural Network for Saliency Detection" [10] for visual saliency prediction. This method considers local, global and contextual information of an image to obtain a high quality salient map.

Tang et al. uses a deeply-supervised recurrent convolutional neural network for full image saliency prediction (i.e. the output of the network is a map of the size of the original image showing the saliency at each pixel location). This neural network constitutes of two components: the first being the recurrent convolutional neural network (RCNN) and the second being the deep supervised network (DSN). The RCNN learns contextual features and the DSN learns discriminative features from a local and global view (i.e. the colours, contrasts, high-level objects such as faces). For this network, VGGNet-16 is used as a pre-trained model. VGGNet-16 has six blocks where the first five blocks contain convolutional layers and pooling layers and the last block contains pooling layers and fully connected layers. The output of one block becomes the input to the next block with the input to the first block being the raw image. Tang et al. removes the last block of VGGNet-16 because the pooling operation in this block makes the feature maps too small (1/32 of the input image size) and the fully connected layers in this block are time and memory consuming. This portion of the network focuses on learning contextual features. The portion to focus on learning local and global views involves generating a side-output from each last convolutional layer of each block (so i.e. 5 side-outputs). This side-output in summary is a convolutional layer followed by a deconvolutional (upsampling) layer. The convolutional layer is a 1 x 1 convolutional kernel which converts feature maps to a saliency map. The deconvolutional layer converts the saliency map to have the same size as the input image.

The final saliency map is constructed by fusing the local and global information learnt by each side-output (for robustness to variation within each side-output). This concatenation is then converted to the final saliency map with the use of a 1 x 1 convolutional kernel. The deconvolutional layer and the last convolutional layer is followed by the sigmoid activation function (for the first and final generation of the saliency maps). As Liu et al. did, Tang et al. applies a dropout of ratio 0.5 to alleviate over-fitting for each block.

As with Guanblin et al., Tang et al. did not evaluate their model performance on the MIT1003 dataset nor with the AUC metric. However for other metrics and datasets they used (e.g. the mean absolute error metric) they observed a significant outperformance compared to state-of-the-art saliency detection approaches.

III. Dataset

We used the MIT dataset [6] to train our model to predict visual saliency. The MIT dataset consists of 1003 images collected from Flicker and LabelMe datasets, with resolutions of the images ranging from 405 x 1024 to 1024 x 1024. The dataset consists of 779 landscape, 228 portrait and several synthetic images. Each image was viewed by 15 human subjects. For each image, the dataset also contains the corresponding raw eye fixation point map (i.e., a map showing how much the human subjects attended to each pixel). Gaussian blur is applied to the raw eye fixation point maps to generate ground truth density maps [2] (this is now a map with a saliency value for each pixel, where the larger the saliency value of a pixel, the more the human subjects tend to focus on that given pixel).

We extract regions of size 42 x 42 from our images to pass into our model. We extract regions whose central pixel has a saliency value of either greater than 0.9 or a saliency value of less than 0.1. We then assign each region a binary value of 0 or 1. The value 0 is assigned if the central pixel saliency value of the region is less than 0.1, and if greater than 0.9, the value 1 is assigned. This binary value becomes the label of the corresponding region. The binary values capture the fixation and non-fixation property of the region (i.e., a label of 0 implies the region is a non-fixation region and so the human subjects attended less to this region and likewise for the value of 1).

For training, we extract 30 regions per training dataset image (10 fixation regions and 20 non-fixation regions). We have 24090 regions in the training dataset (hence we use 24090/30 = 803 images in the training dataset). For testing and validation, we extract 2500 regions per image and have 250000 regions in each set, hence for testing and validation we use 250000/2500 = 100 images in each of these sets respectively). Thus, the training, validation and testing split is: 803, 100 and 100 images respectively.

IV. MrCNN Architecture

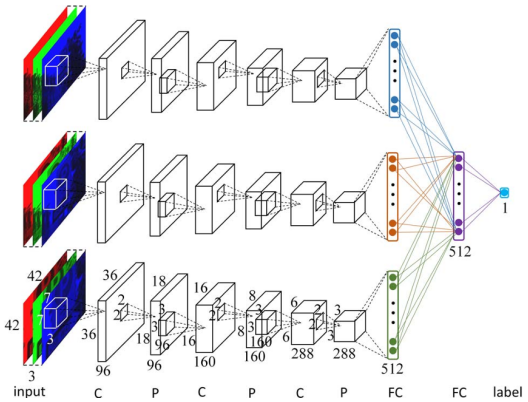

Fig. 1: Liu et al: Mr-CNN Architecture

We discuss the architecture as we have replicated from Liu et al.

For each image in the training dataset, the image is resized to three resolutions with the three resolutions chosen by Liu et al. to be 150 x 150, 250 x 250 and 400 x 400 respectively. For each resolution of the image, a 42 x 42 region is extracted where the centre location of

each region is chosen to be the same across all three resolutions of the image.

The Mr-CNN starts from three streams and takes three inputs (each input is passed through each stream). The inputs are precisely the three 42 x 42 image regions extracted from the three different resolutions of the image.

Each stream is composed of three convolutional (C) layers, three pooling (P) layers and a fully connected (FC) layer. The three streams are then fused together using another FC layer. This fusion allows the learning of bottom-up saliency across multi-resolution inputs. The last fully connected layer is then followed by a logistic regression layer at the end to perform classification. This layer computes a sigmoid activation function outputting a value between 0 and 1. We then classify the region as a fixation or non-fixation region dependent on this value (if the value returned is greater than 0.5 then the region is classified as a fixation region, otherwise is classified as a non-fixation region).

The parameters are shared in each C layer across the three streams. The first C layer has 96 filters and a filter size of 7 x 7. The next two C layers have a filter size of 3 x 3 and 168 and 288 filters respectively. The convolutional stride is set to 1 and there is no padding for any of the C layers. In all the P layers, 2 x 2 max-pooling windows are used.

For all the FC layers, 512 neurons are used and 1 neuron is used in the output layer (i.e. the classification layer). Except for the last layer which uses the sigmoid activation function, the ReLU activation function is utilized for all the C layers and FC layers.

V. Implementation Details

Below we discuss the implementation as we have replicated from Liu et al.

As Liu et al. did, we performed data augmentation on the training set by horizontally flipping each rescaled and cropped image region within this set.

While the original paper trained and tested the model using 10 fold cross-validation, we used a preselected training and validation split described earlier. Hence, the data was divided into training, testing and validation sets with each element of each set being three 42 x 42 regions centred at the same location at different resolutions.

Liu et al. mentions before training, each dimension in the training image regions was mean-centred and normalised to unit variance over the training set. We replicate this as follows: for all the image regions in our training dataset, we find the mean and standard deviation of each dimension i.e. each colour channel red, blue and green. For each image region in the training dataset, we then subtract the mean colour channel from its current colour channel and divide this value by the standard deviation of the colour channel to obtain the normalised colour channel. We do this for all three colour channels. Since Liu et al. applies this normalisation process for the testing stage, we repeat it for both our validation and testing sets.

We replicate the hyper-parameters Liu et al. used for training, where we had a weight constraint of 0.1 to the convolutional kernels of the first convolutional layer. Likewise, we used a weight decay of 0.0002, a linearly increasing momentum from 0.9 to 0.99 along with the training process and a learning rate of 0.002.

As Liu et al. did, we alleviate over-fitting in training by implementing dropout with a corruption probability of 0.5 for the third convolutional layer and subsequent two fully connected layers of each stream.

We train the model on the normalised training dataset for 30 epochs and a mini-batch of size 256. We evaluate the performance of our network on the validation dataset every 2 epochs. For each evaluation, we implemented model checkpointing such that if the AUC is higher than the AUC of a previous evaluation, we save this network and AUC score. Finally, once all epochs have been iterated through, we take the saved network with the highest score as our final network and test this network with our normalised testing dataset.

Additionally, we have also trained a model that extends on the architecture. We train this model for 60 epochs (around 11k training steps) and validate every 8 epochs. We discuss it in detail in section VIII but evaluate it alongside our base implementation below. We refer to it as 'the extension model' or 'model with CBAM modules' and refer to the base implementation simply as 'the base model'.

We trained our model on an Nvidia GeForce RTX 3080 with 8960 cuda cores. Training time was approximately 22 seconds and data load time was approximately 11 seconds per epoch or 18 training steps. Finally, we calculated a total of 2,685,827 and 2,700,841 weights in our base and extension models respectively.

VI. Replicating Quantitative Results

Metric\Model	Our Base	Our Ext.	Liu et al.	Baseline
AUC	**81.90**	**84.48**	N/A	**71 ± 1**
sAUC	**72.77**	**69.23**	**71.9**	N/A

Table 1: Test accuracy using AUC and Shuffled AUC (%)

We tested our models on the MIT1003 training set. Table 1 shows the AUC score and Shuffled AUC (sAUC) score produced by evaluating both our base model and extension on the testing dataset. For the base model, we see that when evaluated with sAUC, we produced a similar score to the original paper. However, our AUC was much higher than the baseline. We can justify this result by describing the drawbacks of the AUC evaluation metric (see section VIII). Our extension model improves on AUC but is slightly worse for sAUC. However, we argue that it performs better than the base model when comparing qualitative results and could be improved on more with further hyperparameter tuning than what we have done due to time constraints.

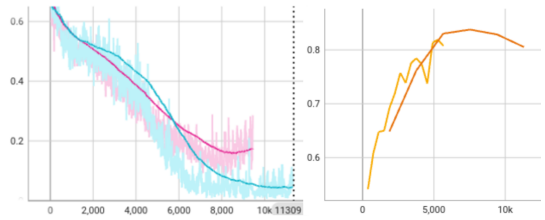

Fig. 2: Training Loss of Base (Pink) and Extension (Blue), Validation AUC of Base (Yellow) and Extension (Orange)

The pink curve in Fig. 2 shows the training loss of the model, which appears to reach a minimum after 8000 training steps and starts to increase after. Note that this is not overfitting since this metric is measured on the training set. We believe this is likely due to the saturated momentum value of 0.99 causing the model to move towards a higher minimum. Observing the yellow validation curve in Fig. 2, we see that the validation AUC value has some minor dips while training and starts to overfit towards the end.

The blue curve shows the training loss of the extension model which performs much better than the base model. However, looking at the validation AUC curve in orange, we see that it start to overfit very strongly towards the end. We used model checkpointing to save the model with the highest AUC.

VII. Qualitative Results

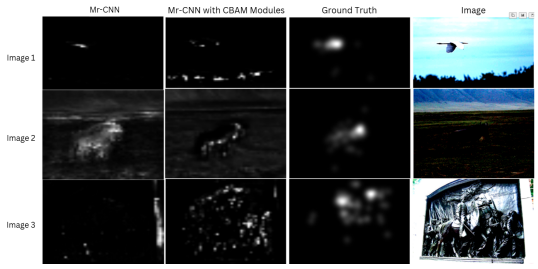

Fig. 3: Prediction Maps of our Base (Mr-CNN) and Extension (Mr-CNN with CBAM layers) alongside Ground Truths and Images

Image 1 of 3 shows an example of our base model performing well. The predicted fixation map and the ground truth show visual saliency in approximately the same location. This suggests our base model performs well for identifying high-level factors (e.g., in this scenario, a bird). Our extension model predicts more saliency towards the bottom and this example is one of the few ones where it performs poorer than the base.

Image 2 of Figure 3 shows a prediction where the base model does well in some areas and struggles in others. The ground truth shows visual saliency on the the central region between the blue sky and the dark green grass whereas our prediction shows visual saliency along the entire horizontal patch where the blue sky and the dark green grass meet as well as at the centre of the dark green grass. Predictions like this suggest that our model is biased towards contrast in images and has a slight central bias (e.g. the visual saliency along the centre of the dark green grass). It also shows a significant 'overprediction' of saliency, covering a larger area than the ground truth. The trend shown in this example is what we found in the majority of qualitative results we evaluated for the base model. On the other hand, our extension model is an improvement as it curbs the 'overprediction' observed in the base model whilst retaining good accuracy.

Image 3 of Fig. 3 is an example where our base model performs a poor prediction of the ground truth. The prediction seems to grasp visual saliency of the lightest and darkest region of the image, reinforcing the idea that the model focuses greatly on contrast when predicting visual saliency. Perhaps introducing more convolutional layers to learn more hierarchical abstract features could lessen the impact contrast has in predicting visual saliency and provide an opportunity for more complex high-level features to be learnt to improve the prediction. Our extension model does better but also struggles for this particular image, likely due to similar reasons.

Summarising, our evaluation of the qualitative results of the base model suggests that it is biased towards contrast and slightly towards the centre. We also observed that it struggles to confine it's predictions to the salient regions in the ground truth. Our extension performs better than the base and while it still has a bias towards contrast in images (exemplified in Image 1), it minimises 'overprediction' and is better at concentrating predicted saliency on regions closer to the ground truth.

VIII. Improvements

A. Shuffled AUC

Our default evaluation metric represents the Area Under ROC Curve (AUC) metric [11], which quantifies the performance of our model by measuring the area under its Receiver Operating Characteristic (ROC) curve. This is found by plotting the True Positive Rate against the False Positive Rate at various classification thresholds for both the associated predicted saliency map and the ground truth saliency map for each image. However, the reliability of the AUC metric is affected by centre bias [12] and border cut [13], leading to prediction maps that may exhibit features of a central Gaussian blob to be classified as excessively accurate.

In order to mitigate this bias, we introduce Shuffled AUC (sAUC) [13] as an additional evaluation metric. Our sAUC implementation adopts thresholded ground truth fixation points above a certain value for our positive set, and randomly samples the size of the positive set from the thresholded fixation points from all other ground truth images as our negative set. This ensures that the negative set consists of points which represent the places in the image which are more relevant to where we wish to define wrong classifications, as the other existing fixation points are more likely to be located towards the centre. By penalising predictions which coincide in these points (false positives) we are able more accurately evaluate models that simply predict unrelated Gaussian-like distributions.

B. Convolutional Block Attention Module

In 2018, Woo et al. proposed the Convolutional Block Attention Module (CBAM) [14], an attention module for feed-forward convolutional neural networks. The module infers attention maps across the channel and spatial dimensions and multiplies the original input feature by these attention maps. The channel and spatial attention maps represent the importance of each channel and spatial dimension respectively. By multiplying these maps with the input feature map, the model refines the features before passing them through subsequent layers, leading to improved focus on more relevant features.

Our justification for using this module is based on our observations about the base model predictions as well as the findings of the paper, which showed consistent improvements in detection performances with various CNN models, which in turn demonstrated the wide applicability of CBAM in CNN-based detection tasks. As discussed earlier in section VII, we found that the base model tends to predict a larger region than that covered in the ground truth saliency maps, and sometimes struggles to find the right region at all. We argued that the multiple resolutions passed through as separate channels in our existing CNN could benefit significantly from the extraction of informative features by using the channel attention module to selectively amplify important channels. Similarly, the spatial attention module allows networks to dynamically focus on more specific parts of the each image rather than the entire scene. We initially implemented a CBAM module only for the final convolutional layer and upon observing positive results extended this to all the convolutional layers to find a significant increase in performance as well as encouraging qualitative results. Our justification is helped by the detailed explanation of the impact of the CBAM modules presented further.

The channel module consists of an average pooling layer which pools the features in the spatial dimension down to a Cx1x1 channel vector, and a max pooling layer which uses max pool logic to process the original input feature in the same way. We process the input feature separately into two vectors using average and max pooling. We pass both these vectors as inputs to a shared MLP with one hidden layer, in order to perform our attention calculation. We add these two resulting vectors using element-wise summation and perform sigmoid activation on it, producing our C-length channel attention map that ranges from 0 to 1, reflecting the importance of each channel feature after attention. We then multiply this map across our input feature, effectively scaling down the individual channels that have been determined to have less significance.

The spatial module is applied to the input feature after it has been multiplied by the channel attention map. Applying spatial attention involves first applying an average function across the channel dimension. This creates a WxH 2D tensor where W and H are the width and height of the input feature. The same is done using the max function across the channel dimension to produce two WxH 2D tensors. We concatenated these tensors and perform a simple convolution to reduce the channel size from 2 to 1, representing the attention calculation between the two tensors. This results in a 1xWxH tensor on which we perform a sigmoid activation to once again produce an attention map representing the relative importance of each feature in the spatial domain. By multiplying this map with our channel-attended input feature, we scale down the less significant parts of the image itself, allowing for attention to be placed on the important areas in the image.

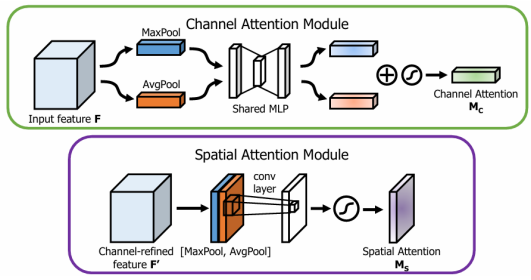

Fig. 4: Woo, Park, Lee, Kweon: Channel and Spatial Attention Modules(taken from [14])

We end up with an input feature that has been scaled with respect to attention in the channel and spatial dimensions, which we then continue to pass through the FC layers. We see a significant qualitative change when we generate prediction maps for the model. We see that in cases where the base model failed to find a salient region, the model integrated with CBAM was successful.

We found that applying the CBAM module after each convolutional layer resulted in a higher AUC score compared to a model with only the final convolutional layer integrated with CBAM.

We conducted hyper-parameter tuning on the extension model with CBAM over the parameters of learning rate, batch size and dropout corruption probability as shown in Fig. 5 by running and evaluating each configuration. We can see that AUC and sAUC is not always correlated and it was was difficult to objectively select the best model. In the end, we selected the model that showcased the best qualitative results out of the top performing models. This model had learning rate of 0.002, batch size of 128 and a dropout probability of 0.5.

Learning Rate	Dropout: 0.2	Dropout: 0.3	Dropout: 0.4	Dropout: 0.5
0.0002	AUC: 80.93% sAUC: 66.73%	AUC: 83.54% sAUC: 67.02%	AUC: 83.68% sAUC: 65.52%	AUC: 83.00% sAUC: 70.21%
0.0011	AUC: 86.73% sAUC: 60.31%	AUC: 85.37% sAUC: 62.67%	AUC: 87.74% sAUC: 65.35%	AUC: 83.35% sAUC: 65.79%
0.002	AUC: 88.14% sAUC: 65.88%	AUC: 87.26% sAUC: 65.36%	AUC: 86.03% sAUC: 66.04%	AUC: 84.48% sAUC: 69.23%

Batch size: 128

Learning Rate	Dropout: 0.2	Dropout: 0.3	Dropout: 0.4	Dropout: 0.5
0.0002	AUC: 79.35% sAUC: 73.56%	AUC: 81.47% sAUC: 72.08%	AUC: 75.95% sAUC: 76.34%	AUC: 79.05% sAUC: 73.72%
0.0011	AUC: 86.15% sAUC: 63.60%	AUC: 86.22% sAUC: 60.88%	AUC: 66.25% sAUC: 59.32%	AUC: 86.49% sAUC: 62.36%
0.002	AUC: 85.28% sAUC: 65.87%	AUC: 86.33% sAUC: 64.24%	AUC: 84.61% sAUC: 62.48%	AUC: 80.74% sAUC: 64.07%

Batch size: 256

Fig. 5: Hyperparameter Tuning

IX. Conclusion and Future Work

In this report, we detail our implementation of the paper "Predicting eye fixations using convolutional neural networks" [2]. We evaluate how our model performs on the MIT1003 visual saliency dataset, and provide quantitative analyses using both the AUC and sAUC metric. We find that, similar to the saliency maps produced by the model from the original paper, our model can effectively identify most salient points but can also tend to predict more background points as false positives. We extend the base architecture in order to mitigate this behaviour by introducing Convolutional Block Attention modules to the last CNN layer. This allows the model to gain more valuable information about the channel features, i.e. the resolutions, as well as to focus on more important spatial information present in the images.

In future work, more rigorous hyper-parameter tuning could be leveraged to find an optimal combination of factors such as learning rate, momentum, dropout factor, and weight decay, particularly for the extension model to combat noisy test metrics. In this way, the optimal balance between different factors such as batch normalisation and dropout, which both contribute to regularisation, could be found, and the risk of having both factors contribute towards under-generalisation could be mitigated. Additionally, in order to truly evaluate the model's generalisation ability, other datasets such as the Toronto visual saliency set could have been used to test the model. This could give us valuable insight into what factors might cause over-fitting in the model.

References

[1] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[2] N. Liu, J. Han, D. Zhang, S. Wen, and T. Liu, "Predicting eye fixations using convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 362–370, 2015.

[3] W. Einhäuser, M. Spain, and P. Perona, "Objects predict fixations better than early saliency," *Journal of vision*, vol. 8, no. 14, pp. 18–18, 2008.

[4] N. Bruce and J. Tsotsos, "Saliency based on information maximization," *Advances in neural information processing systems*, vol. 18, 2005.

[5] G. Li and Y. Yu, "Visual saliency detection based on multiscale deep cnn features," *IEEE Transactions on Image Processing*, vol. 25, p. 5012–5024, Nov. 2016.

[6] T. Judd, K. Ehinger, F. Durand, and A. Torralba, "Learning to predict where humans look," in *2009 IEEE 12th international conference on computer vision*, pp. 2106–2113, IEEE, 2009.

[7] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3166–3173, 2013.

[8] A. Kroner, M. Senden, K. Driessens, and R. Goebel, "Contextual encoder–decoder network for visual saliency prediction," *Neural Networks*, vol. 129, pp. 261–270, 2020.

[9] X. Huang, C. Shen, X. Boix, and Q. Zhao, "Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 262–270, 2015.

[10] Y. Tang, X. Wu, and W. Bu, "Deeply-supervised recurrent convolutional neural network for saliency detection," 2016.

[11] N. D. B. Bruce and J. K. Tsotsos, "Saliency based on information maximization," in *Proceedings of the 18th International Conference on Neural Information Processing Systems (NIPS'05)*, (Cambridge, MA, USA), pp. 155–162, MIT Press, 2005.

[12] B. W. Tatler, R. J. Baddeley, and I. D. Gilchrist, "Visual correlates of fixation selection: effects of scale and time," *Vision Research*, vol. 45, pp. 643–659, Mar 2005.

[13] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. Cottrell, "Sun: A bayesian framework for saliency using natural statistics.," *Journal of vision*, vol. 8 7, pp. 32.1–20, 2008.

[14] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," *arXiv preprint arXiv:1807.06521*, 2018.