COMSM0045: Applied Deep Learning Coursework

Saif Anwar
University of Bristol
Department of Computer Science
United Kingdom, Bristol
sa17826

I. Introduction

Visual saliency is the property of prominence through differentiation between an object and its surroundings. Humans are naturally fast at observing an image and fixating on the most important regions to interpret more information in a shorter period of time. This helps to develop an understanding of an environment and allows for enhancement of spacial awareness towards objects of proposed significance without wasting cognitive resources. Contrary to object detection, this is a problem associated with the location of an object within an image rather than the what the object is.

Through training a convolutional neural network (CNN), saliency within images can be predicted. This report presents a replication of the work proposed by Pan et al. [1] where they train a CNN and quantify the accuracy of its produced saliency predictions. They have designed 2 of the first end-to-end CNNs for this task, of which the 5-layer Shallow CNN (JuntingNet) has been replicated.

CNNs are deep learning architectures aiming to simulate the behaviour of the primate visual cortex. These differ from multilayer perceptrons (MLPs) since MLPs contain fully connected layers where every neuron from one layer is connected to every other neuron in its preceding and succeeding layers. Given these neural networks (NNs) can contain a substantial number of neurons, the greater complexity of MLPs tends to lead to overfitting of the training data [2], thus weaker generalisation. CNNs on the other hand are much more sparsely connected and closer match the work produced by Hubel and Wiesel [3] who concluded that the visual cortex contains multiple receptive fields. Neurons, in accordance to their receptive fields, will respond to independent stimuli where combining receptive fields of all neurons covers the entire visual field. This means that not all neurons will need to connect with each other and can instead focus on identifying their particular stimulus.

One of the key limiting factors towards the progress of saliency prediction is the associated difficulty and cost with regards to collecting suitable training data. Rather than just a general label for an entire image, saliency must be defined for each pixel in the image. To create training data that satisfies this, human fixation points must be captured by observers through eye tracking data.

Within recent years, through increased availability and establishment of large-scale crowdsourcing data collection, saliency prediction has become a much more active research area within computer vision. SALICON [4] is the largest saliency prediction dataset which was used by Pan et al to develop JuntingNet so it will also be used in this replication.

II. Related Work

Similar to object detection and segmentation, early approaches for predicting saliency within images were based on low-level features such as contrast, edges and colour. However, using only low-level features does not differentiate between importance of different detected regions of significance. These predictions therefore, could include background objects that would not generally be fixation points from a human visual perspective. Also with features such as contrast, an object may not differ with regards to its surroundings so would be missed entirely by such an approach.

After the introduction of new datasets, such as SALICON and CAT2000 [5], more data-driven approaches have been developed using supervised learning methods. These allow us to learn higher level features and overcome the difficulties presented by previous techniques. In this section we will explore recent works succeeding that of Pan et al.

Borji et al. [6] focus on similarity between images when training a saliency prediction framework. They combine studies exploring the role of memory in guiding eye movements to dictate fixation points within an image. They find that images with similar scenes will have similar saliency maps. Using this they can train a model to detect a scene by aggregating saliency maps produced by similar training images.

Furthering their previous work, Pan et al [7] developed SalGAN; a dual NN architecture to more accurately predict salient areas within images. The first network produces saliency maps from a raw image (such as JuntingNet). The second network takes the first's output and predicts whether the saliency map shown is a prediction or ground truth. This discriminator network acts as a classifier whose loss is penalised through misclassification. General adversarial networks (GANs) aim to replicate a probability distribution relationship between the ground truth and the prediction such that it minimises their difference.

As well as producing saliency predictions for images, Zheng et al [8] developed a CNN architecture to predict saliency of webpages when different tasks are assigned to the user.

The intuition behind this being that depending on the task assigned, users will vary their area of focus when viewing a webpage. These task-based predictions can then be used to optimise webpage design. They combine 2 subnets: task-specific attention shift prediction and task-free saliency prediction. They then fuse the outputs of the two subnets to create a final prediction.

III. Dataset

JuntingNet was trained using various datasets however the SALICON representation will be replicated. The dataset contains a total of 20,000 images making it the largest available dataset for saliency prediction. This is split up with 10,000 images assigned for training; 5,000 for testing; and the remaining 5,000 for testing. It is built upon images from the Microsoft CoCo: Common Objects in context [9] dataset.

To measure visual attention on images, eye tracking can be used to monitor fixation points. Collecting this sort of data however, is complex and costly due to the custom hardware required to do so. SALICON's ground truths were instead labelled through a crowdsourcing scheme where a general purpose mouse was used, rather than eye tracking hardware, to record visual attention. By aggregating multiple users trajectories, the probability of salient areas within an image can be computed. Through validation, this technique was shown to be highly similar compared to using eye tracking hardware.

Each of the subsets of the dataset are stored as pickle files. The python 'pickle' library allows you to serialise a python object such that the character stream can be used to reconstruct the object at another location.

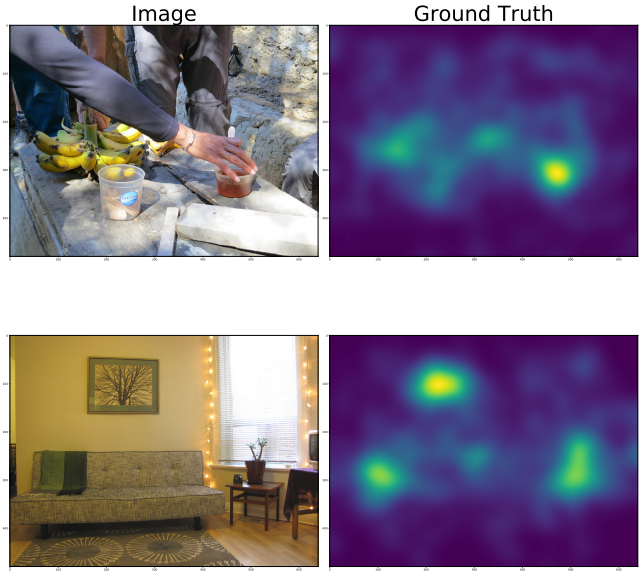

Fig. 1: Example images and their corresponding ground truths from the SALICON dataset

IV. Input

So far we have not discussed the form of the inputs to train the NN or the saliency prediction outputs. Each image from the dataset is an image from the MS:CoCo dataset such as shown in Figure 1a. A saliency map is a pixelwise representation of the saliency probability across the image. In a greyscale saliency map, the brightness of each pixel is directly proportional to its saliency. Each image in the SALICON dataset has its respective ground truth (or label) which is a saliency map. Examples of these labels are shown in Figure 1b, whose corresponding images are in Figure 1a. Our model produces predictions of the same format as the SALICON ground truths.

V. Shallow Architecture

The developed replication of the JuntingNet architecture contains 5 hidden layers. The input to the network will consist of a SALICON input image resized from its original (640×800) down to $(96 \times 96 \times 3)$ as it is an RGB image. Table **??** shows the layers within the CNN. A visual representation of the dimensions and arrangement of each of the layers is shown in Figure 2 with colours corresponding to Table **??**.

Input size $(96 \times 96 \times 3)$
Convolution 1 $(5 \times 5 \times 32)$
Max Pooling 1 $(kernel(2 \times 2) stride = 2)$
Convolution 2 $(3 \times 3 \times 364)$
Max Pooling 2 $(kernel(3 \times 3) stride = 2)$
Convolution 3 $(96 \times 96 \times 3)$
Max Pooling 3 $(kernel(3 \times 3) stride = 2)$
Fully Connected 1 4608
Slice 1 — Slice 2
Max out
Fully Connected 2 2304
Output

TABLE I: Architecture of the developed CNN

The received input goes through 3 sets of convolutional and max pool layers. After each convolutional layer there exists a linear ReLU activation layer. The ReLU activation function will return 0 for any negative input x, or will return x if input x is positive. The max pooling layers reduce the dimensions from (96×96) to (10×10). The first max pool layer will take the maximum value in every (2×2) region whilst taking strides of 2 pixels.

The output from this layer is of size $(46 \times 46 \times 32)$. After another convolutional layer, the second max pooling layer reduces the size down to $(22 \times 22 \times 64)$. The final max pool layer provides an output of size $(10 \times 10 \times 128)$ to the first fully connected layer containing 4608 neurons. This layer splits the network in two and performs the maxout activation function on the network. The maxout function takes a maximum value from n linear functions where $n = 2$ in this case. This halves the total number of neurons to 2304. The output vector can be reshaped to a (48×48) saliency map. A Gaussian filter of standard deviation 2 is applied to resize the saliency map to match that of the original image.

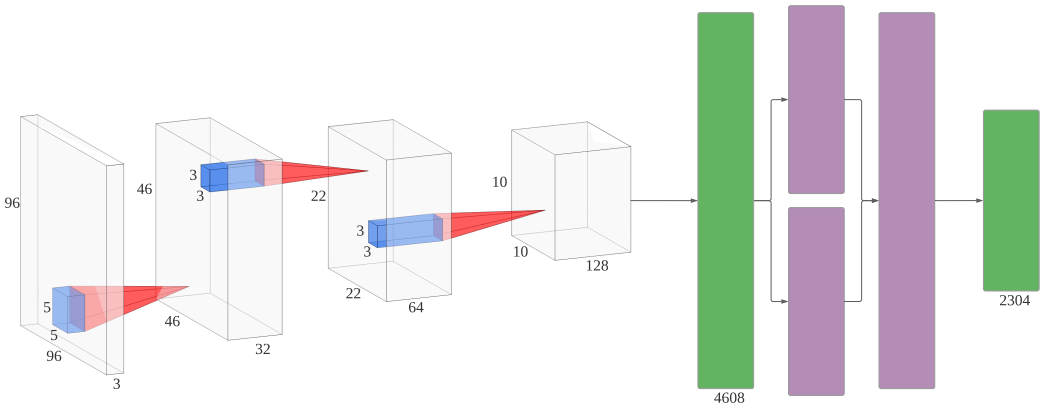

Fig. 2: Developed NN architecture visualising dimensions and structure corresponding to Table 1.

VI. Implementation Details

The CNN was developed in Python using the PyTorch library [10].

A. Training

The model weights were initialised using the following distribution

$$X \sim \mathcal{N}(0,\, 0.001^2) \tag{1}$$

The model performance is quantified through an L_2 Euclidean Loss, also known as Mean Squared Error (MSE). The MSE of a model in iteration θ can be calculated as

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2 \tag{2}$$

for a batch containing N samples, where $\hat{\boldsymbol{Y}}$ is the batch of predictions from the CNN output layer and $\boldsymbol{Y}$ are the corresponding ground truths.

Gradient Descent (GD) is an iterative approach to minimise the loss function by taking steps proportional to the negative gradient of the function at different points [11]. For a NN, the minimum of the loss function signifies the optimal values of its weights. By computing the forward activity of a model in a training iteration, the loss of a given model is calculated. GD uses the backpropagation algorithm to calculate which direction to adjust the model weights. After forward activity has been computed, the observed loss can be propagated back through the network to compute the gradient of the loss curve with respect to each weight. This is then used to update the weight value. Stochastic gradient descent takes this further by introducing batches of training data for each iteration. This allows for generalisation as well as efficiency with regards to speed. The proportion of the gradient to take as an update step is decided by the learning rate, η. The new value of weight i in layer k is calculated as

$$w_i^k(\theta+1) = w_i^k(\theta) - \eta\frac{\partial J}{\partial w_i^k} \tag{3}$$

The value of the learning rate is critical on the competence of SGD. If the chosen η is too small (Figure 3a), the weight will take too long to converge to its optimum, or it could get stuck in local optima.. If η is too large (Figure 3c), an adjusted weight will overshoot its optimum value.

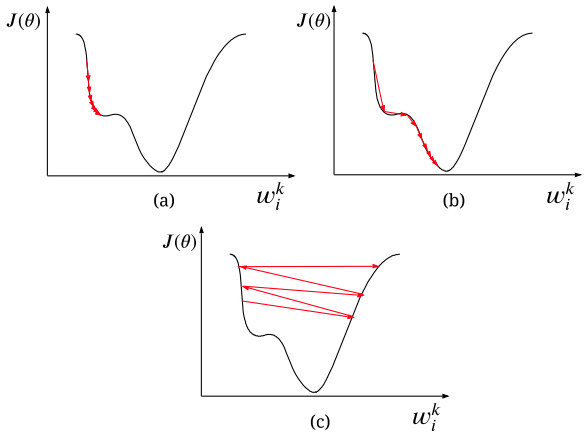

Fig. 3: Instances of Gradient Descent shown for different values of the learning rate η.

B. Hyperparameters

The model was trained for 1,000 epochs using batches of 128 random images from the available 10,000 images in the SALICON training dataset. In line with the training method used by Pan et al., the learning rate was varied throughout training with η decreasing linearly from 0.3 to 0.0001.

C. Acquiring Results

Training of the CNN was performed on BlueCrystal Phase 4, using an NVIDIA P100 GPU with 16GB of VRAM and 3584 CUDA cores. Throughout training, the MSE of the model was recorded after training each batch of training data. The model's accuracy was tested using the test set at a frequency of 10 epochs. The total training time for the model was approximately 2.5 hours.

VII. Replicating Quantitative Results

After training for 1000 epochs, the performance of the model is quantified through the following 3 metrics.

- **Pearson's Corellation Coefficient (CC):** A statistical method for measuring how correlated two variables are; in this case the prediction saliency map and corresponding ground truth. CC values per pixel which are highly positive, occur where the prediction and ground truth have similar values. Values close to -1 and 1 indicate an almost perfectly linear relationship.
- **AUC Borji:** Area under curve (AUC) metrics are most common for evaluating saliency predictions. Fixation points are defined in the ground truths. A uniform random sample of pixels is taken from the saliency map as negatives. Binary classification comparisons are made between the prediction and ground truth. If pixel values are above the threshold value then they are classed as false positives.
- **AUC Shuffled:** Another AUC metric where instead of a random uniform sample of negatives, samples are taken from fixation points of other images. As before, values falling above the threshold value are classed as false positives.

Evaluation Metric:	CC	AUC Shuffled	AUC Borji
Pan et al.:	0.58	0.67	0.83
Replication:	0.66	0.55	0.71

TABLE II: Training evaluation metrics

For CC, the replication shows noticeable improvements over the results obtained from JuntingNet. The AUC shuffled metric exhibits the lowest performance out of the 3 evaluated. Given the model uses a Gaussian filter on the saliency maps, there is a central bias. Due to this, fixation points selected randomly by AUC Shuffled which may lie out of the centre, will not be predicted correctly as consistently. AUC Borji, on the other hand, exhibits a higher performance due to its fixed image random uniform selection. This is expected as AUC Shuffled is known to discriminate against models with a centre bias [12]. However, a central bias is not necessarily a negative attribute of the model. According to a study conducted by Tatler [13], there is a central bias on fixation points from humans when making observations within images. Therefore, our model would more closely resemble characteristics observed within natural human activity.

VIII. Training Curves

The MSE recorded throughout the training epochs is shown in Figure 4 for both the training and test datasets. As can be noticed, the MSE observed on the training dataset is consistently lower than the testing counterpart. This could be due to the model being trained only using batches from the training dataset therefore would overfit to features observed in the seen data.

Figure 5 shows the convolutional filters learned by the first convolutional layer in the network. These represent features that the model would learn to detect.

IX. Qualitative Results

Figure 6 shows 3 examples of the saliency maps produced by the CNN compared to their original image and the corresponding ground truths.

Figure 6a) is an example where the saliency map shows noticeable resemblance to the ground truth. As seen in the ground truth, there are 3 fixation regions with the predicted saliency map showing prominence in the same 3 regions.

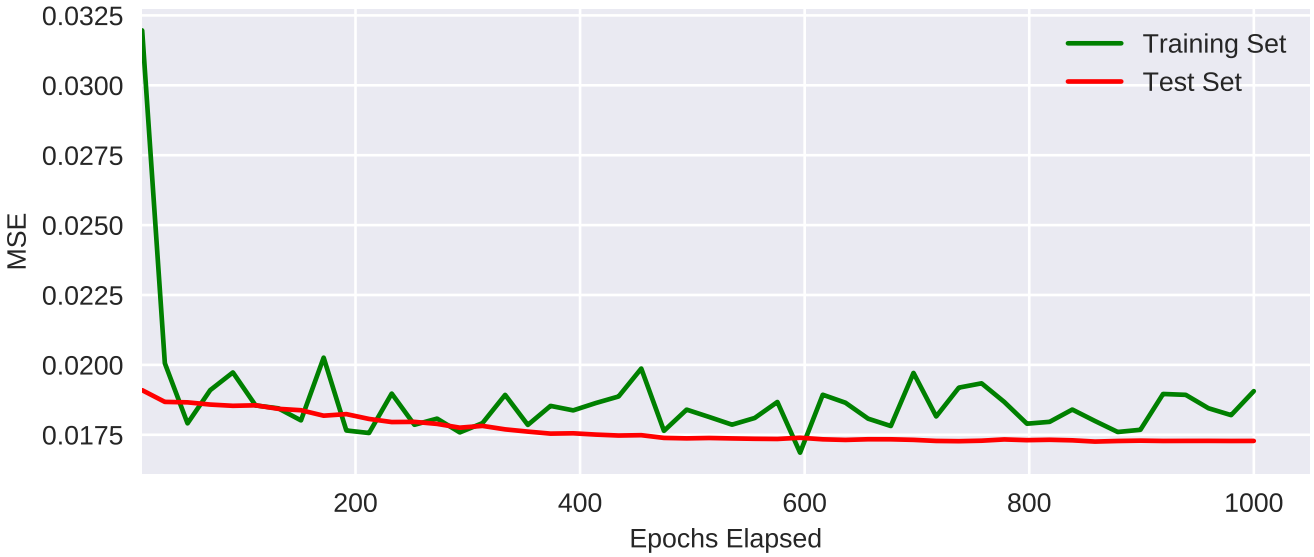

Fig. 4: Error when training the developed CNN for 1000 epochs and linearly interpolated across 50 intervals.

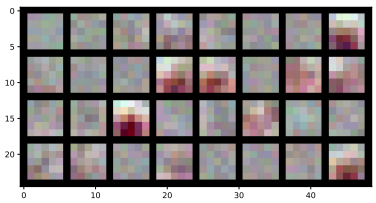

Fig. 5: Convolution filter features learnt by the first convolutional layer.

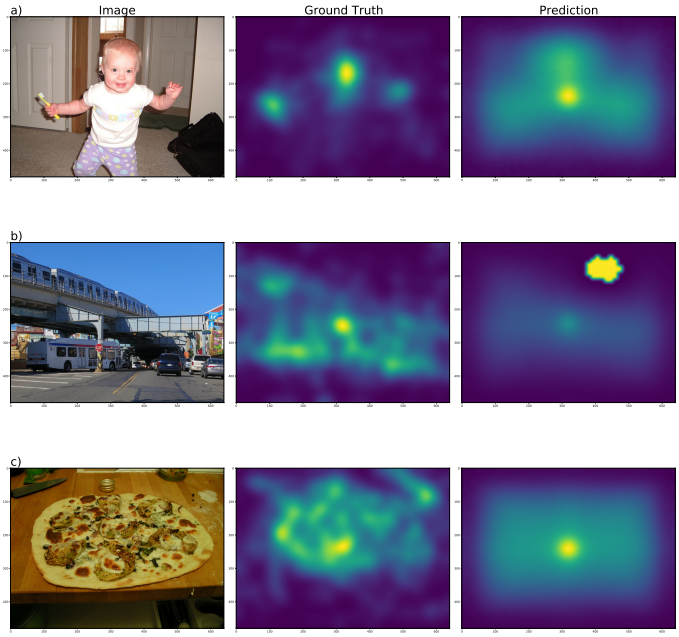

Fig. 6: Predictions produced by the developed CNN compared to the corresponding ground truths and images from the SALICON dataset

The prediction in Figure 6b), on the other hand is, far from an accurate depiction of its ground truth. There is a bright spot in the top of the prediction which may be due to the resizing of the image. There is a significant contrast between the sky and other objects and when processing from a smaller (48×48) image to the full resolution, this variation in brightness and colour contrast resorts to a fixation point prediction. Finally for Figure 6c), looking at the ground truth, the fixation points are quite scattered. Also, there is not much variation in brightness and contrast across the image which lead to no noticeable fixation point predictions.

X. Conclusion and Future Work

This report contains a brief literature review followed by a replication of the shallow architecture proposed by Pan et al in [1]. The architecture and training parameters were reproduced, albeit using contrasting hardware for the training process. From the observed results, it can be deduced that the replicated network shows similar results to those achieved by Pan et al through JuntingNet. Although there are slight improvements in the CC metric, there are also shortcomings within others. This may be due to the differences in computing hardware used.

To improve upon the architecture and results, there a various approaches which can be taken. Firstly, further regularisation techniques can be explored which go beyond those described by Pan et al. Also, different loss metrics can be used in the training process. Each of these would alter the behaviour of the backpropagation algorithm and essentially the training behaviour of the network. Finally, the model can be evaluated using the other proposed datasets by Pan et al (e.g. iSUN). This would allow for further comparison, of the replication with the original, and potentially give light to any key differences not yet addressed.

References

[1] Junting Pan, Elisa Sayrol, Xavier Giro-I-Nieto, Kevin McGuinness, and Noel E. O'connor. Shallow and deep convolutional networks for saliency prediction. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:598–606, 2016.

[2] Rich Caruana, Steve Lawrence, and Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in Neural Information Processing Systems*, 2001.

[3] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 1962.

[4] Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. Salicon: Saliency in context. June 2015.

[5] Ali Borji and Laurent Itti. CAT2000: A large scale fixation dataset for boosting saliency research. *CoRR*, abs/1505.03581, 2015.

[6] Hamed R. Tavakoli, Ali Borji, Jorma Laaksonen, and Esa Rahtu. Exploiting inter-image similarity and ensemble of extreme learners for fixation prediction using deep features. *Neurocomputing*, 244:10–18, 2017.

[7] Junting Pan, Cristian Canton-Ferrer, Kevin McGuinness, Noel E. O'Connor, Jordi Torres, Elisa Sayrol, and Xavier Giro-I-Nieto. SalGAN: Visual saliency prediction with adversarial networks. *arXiv*, pages 1–9, 2017.

[8] Quanlong Zheng, Jianbo Jiao, Ying Cao, and Rynson W.H. Lau. Task-driven webpage saliency. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11218 LNCS:300–316, 2018.

[9] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8693 LNCS(PART 5):740–755, 2014.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[11] D. P. Mandic. A generalized normalized gradient descent algorithm. *IEEE Signal Processing Letters*, 11(2):115–118, 2004.

[12] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Fredo Durand. What Do Different Evaluation Metrics Tell Us about Saliency Models? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(3):740–757, 2019.

[13] Benjamin W. 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):1–17, 2007.