Saliency Prediction via Convolutional Networks

Gregg Keanu Nurick
GN17738
University of Bristol

Abstract

This paper is a written as a submission for a coursework at the University of Bristol. The cohort was tasked with reading the paper "Shallow and Deep Convolutional Networks for Saliency Prediction" [1] by *Pan et al*, and replicating a shallow convolutional architecture described in said paper. The paper takes on a end-to-end neural network driven approach to the task of predicting a saliency map of an image; a task that is traditionally done using hand made filters. Some basic level of understanding of convolutional neural networks, and neural nets as a whole will be assumed. For the extent of this paper, the term "*Pan et al*" will be used to refer to the original paper that this coursework is based on, and the term "the examiners" will be used as a reference to the university when describing the criteria expressed in the coursework description.

I. Introduction

Convolutional neural networks are very popular and effective for extracting information from data where it is not just individual values that are of concern, but rather some scope of continuous values in an input, such as images and audio series [2]. The aim of *Pan et al* was to take an end-to-end convolutional network approach to saliency prediction (see section IV for definition of salience and saliency maps). This problem is one that beforehand was mostly resolved using hand made filters to detect features of an image at a local level, and successively using features learned to gain some understanding of an image's potential saliency map. This approach, however, gives rise to some issues. Firstly, using filters for feature detection has its limitations; for example, an edge in an image may be poorly defined with respect to pixel values if the difference in light level of the object from its background is low, or if the edge is blurry. Both of these cases may hinder the detection of some feature, while that said feature may still be of paramount importance its image's saliency map. Furthermore, even given perfect feature detection, this will give us information on a very local level within areas of the image, and it is then left to interpretation to derive some useful output. For saliency prediction, we need a broad scope of image information, as it will be a compilation of characteristics of an image that ultimately influence the attention of a human eye.

The need for a broad scope of image information, however, bring about its own set of problems. Understanding a vast multitude of characteristics of an image effectively becomes the process of trying to approximate a very complex function. To do this, we generally need a (relatively) high number of layers and parameters in a network [3]. This unfortunately can leave a network prone to over-fitting due to its ability to replicated functions with high levels of oscillation, and as described in *Pan et al*, the kind of data needed for training in this case is rather hard to come by. This means there is a limited supply of training data we can use, making over-fitting a real concern, and resulting in measures being taken to reduce to complexity of the model's architecture.

Another concern brought up in *Pan et al* is the need for a regressive model rather than a classification model; The resulting output image must be spatially coherent, with a smooth transition between neighbouring pixels. This adds further measures that must be taken when retrieving the output of the network.

II. Related Work

Since the publication of *Pan et al*, there have been multiple related works exploring the intricacies of this task.

The paper *Deep Saliency Models: The Quest For The Loss Function* [4] explores in more detail potential loss functions to be used as a base for stochastic gradient descent, and their effect on the performance of convolutional models for saliency prediction. it explores pixel based loss functions (such as the one used in this replication), distribution based loss functions, saliency inspired loss functions and perceptual based loss function. It also attemts to address the issue of centre bias, a problem which seemingly inherent in the data. The paper gave extensive results on performance of a network with not only different loss functions, but also vast combinations of loss functions.

The paper *Tidying Deep Saliency Prediction Architectures* [5], as the title suggests, works on tidying the saliency predictions given by convolutional models. It does this not by increasing the complexity of the model, but rather by analysing the architecture through methods

of modular analysis and adjusting each component according, usually for a simpler alternative.

Finally, *End-to-End Saliency Mapping via Probability Distribution Prediction* [6] takes a more in depth look as using probability distributions to find measures of success of a prediction. It does this by looking at a saliency map as a generalised Bernoulli distribution, then explores multiple loss functions that return some measure of distance between probability distributions.

III. Dataset

The data used for training was taken from the Salicon (Saliency in Context) data-set. This data was gathered using participation from a crowd, utilising a mouse to indicate points of interest for a given image. An aggregation of mouse activity from multiple participant was used to create each image's saliency map.

Each point in the training data is a dictionary with the keys 'X', 'y', and 'file_name'. 'X' holds the image used for training, downscaled to a size of 96x96x3 from a size of 640x480x3, and 'y' holds the ground truth saliency map (created through physical measurement of human activity) for the image's output to be compared to, downsized to 48x48 from 640x480. Stored in 'file_name' is simply the name of the file that the image is stored in.

I see three potential reasons why the image and ground truth values have been downsized for training purposed. Firstly, it reduces the complexity needs of the model; our input layer is over 30 times smaller than if we did not take this step, significantly decreasing the training time and memory required to train. Secondly, this reduction in complexity may help the model's tendency to overfit. Thirdly, while the downsized version of the input image will most likely have no impact on the true salience of that image, working with a resulting saliency prediction that is downsized allows for a reversed Gaussian filter to be applied to get a correctly scaled version of the saliency prediction, which will satisfy the requirement that the saliency map has smooth transitions of pixel values.

The training data was preprocessed by the examiners and provided to us as a set consisting of not only the original Salicon data, but also augmented versions of the data that had been mirrored horizontally. Data augmentation was used by *Pan et al* presumably in attempts to increase the performance of the model, a technique that many architectures working with continuous data make use of to overcome data scarcity [7].

The validation data provided was formatted similarly to the training data, but with two extra keys in each dictionary. 'X_original', whose value contained to originally sized image, and 'y_original', whose value contains the originally sized saliency map. These extra keys have been provided for visualisation of the images and resulting saliency maps.

Both datasets were provided as pickle files. The training set consisted of 20000 examples (including augmentations), and the validation set consisted of 500 examples. This is far from the normal training/test split which usually lies between 70/30 and 80/20, but with limited data, some compromises must be made.

IV. Input

Two types of input are given to the model during training. Firstly, the downsized images that are to be passed through the network, and secondly, the downsized ground truth saliency maps which are used for calculating the loss to be utilised in the process of stochastic gradient descent gradient descent of the loss function via backpropagration [8].

The saliency map of an image is a visual representation of its respective image's salience. Salience is defined by the quality of being particularly noticeable or important. So, in the context of the salience of an image, a saliency map is a representation of the image (with aligning dimensionality) showing the points and areas which are most noticeable to the human eye, in other words, the most "eye-catching" parts of the image.

Examples of ground truth saliency maps of images can be found in figure 1.

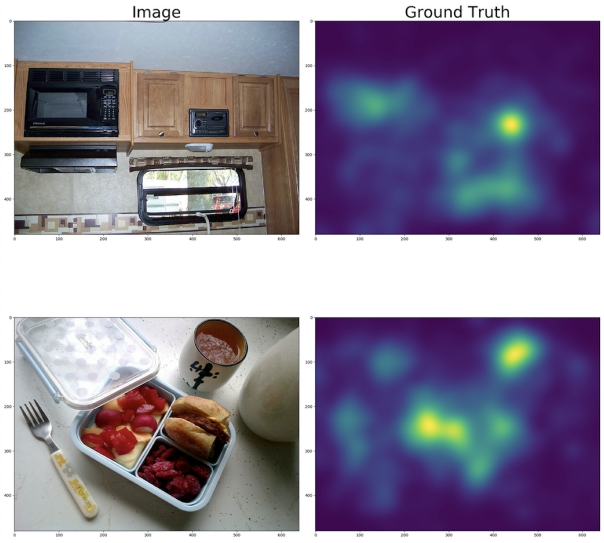

Figure 1. Saliency maps of two example images

V. Shallow Architecture (*Pan et al*)

A full description of the architecture can be found in figure 2.

A. Convolutional layers

As mentioned in the introduction, convolution layers in a neural network are very useful for capturing information in continuous data such as images. The kernel allows for learning of features that range over multiple pixels, something that is clearly important for a use case such as this.

Each of the convolutional layers are passed through a rectifying linear unit (ReLu) function. The ReLu function is one of the most common activation function used in neural networks, it sets negative values to 0, thus generating a higher level of sparsity on the layer and also reducing the risk of vanishing gradients. It is unclear from *Pan et al* what the original usage of activation functions were in their architecture, but activation functions are commonly used in neural net layers due to their ability to improve the performance a network in many cases [9], so it has been used here.

Shallow Architecture		
Layer	Layer type	Details
1	Convolutional	Size: 96x96, Depth: 32, Kernel size: 5x5, Activation: ReLu
2	Max-pool	Kernel size: 2x2, Stride: 2
3	Convolutional	Size: 48x48, Depth: 64, Kernel size: 3x3, Activation: ReLu
4	Max-pool	Kernel size: 3x3, Stride: 2
5	Convolutional	Size: 32x32, Depth: 128, Kernel size: 3x3, Activation: ReLu
6	Max-pool	Kernel size: 3x3, Stride: 2
7	Fully connected	Size: 15488, Activation: None
8	Max-out	Slices: 2
9	Fully connected	Size: 2305, Activation: None

Figure 2. Architecture description

B. Maxpooling layers

Pooling layers have been used in between convolutional layers as a measure to reduce complexity of the model as the model's layers abstract further from the input. This can be seen as a mitigation against overfitting, and also as a way to improve to compute time of training. Due to the nature of saliency maps, it wouldn't seem that much critical information would be loss through maxpooling as is sometimes the case [10], making the implementation of such layers in this architecture a net positive for performance.

C. Maxout layer

The max-out layer works in place of an activation function after the first fully connected layer. Max-out layers effectively take arbitrarily complex function representations and provide an approximate linear representation in place of them [11]. This is a method of generalising the model to help further protect from overfitting. Also, adding some higher level of simplicity to the output is desirable in the context of saliency map outputs.

D. Fully connected layers

The architecture makes use of two fully connected layers which together downsize the information to the dimensionality required for the output. These add complexity to the model, and their respective parameters will surely be highly important in the learning process. No activation function was used on either of these layers. We were instructed by the examiners to not use one after the first layer, as they noted that this helped with performance. None was used after the final layer as this is the output layer, so we would like to maintain its information for its use in calculating the loss.

VI. Implementation Details

The architecture described in *Pan et al* was replicated in Pytorch, as instructed by the examiners. The model instance is derived from the nn.module class. Upon initialisation of the model, all layers are defined, once again using instances of Pytorch classes. For the convolutional layers, we were instructed by the examiners to use specific values for padding. This is fact led to some discrepancy with the paper. *Pan et al* states that the original image is downsized to a size of 10x10 by the last max-pooling layer. This differs from reality when using the specifications required by the examiners, as it in fact downsizes to a dimension of 11x11, so this is the size that was used. All weights were initialized using a Gaussian distribution, and biases set so 0.1, as in *Pan et al*. Predefined Pytorch classes were used again for describing the loss function and the optimiser, which allowed easy implementation of Nestarov momentum, initialisation of learning rate, and steps of stochastic gradient descent.

While *Pan et al* makes no mention of weight decay for the shallow architecture, we were instructed by the examiners to include L_2 norm weight decay with a value of 0.0005, so this use. The eximiners also instructed us to not include momentum decay, despite it being used in *Pan et al*.

The Dataloader class was made use of for processing batches of the data. *Pan et al* makes light of the ability to train with "very large" batches on the shallow architecture due to its lack of need for extremely high memory requirements. The examiners clarified that a batch size of 128 was sufficient.

The native Python function 'reshape' was used for slicing the layer for the max-out method, and a predefined ReLu function was implemented from Pytorch.

To deal with the degrading learning rate mentioned in *Pan et al*, every 100 epochs I set the gradient to a factor of 0.55 of itself, after having checked it not be below the minimum learning rate described in *Pan et al*. To do

this, I looped through the attribute 'param_groups' in the optimiser, adjusting the value stored in key 'lr'.

A validation method was called every 10 epochs, which ran the validation set through the model, finding their average loss. After 1000 epochs, the validation set was run through the model again. This time, however, each prediction was kept as an element in a list that was then dumped as a pickle file to be used to obtain accuracy metrics.

VII. Replicating Quantitative Results

Test Accuracies			
SALICON (val)	CC	AUC Shuffled	AUC Borji
Pan et al [2016]	0.58	0.67	0.83
Examiners' replication	0.65	0.55	0.71
My replication	0.65	0.55	0.71

VIII. Training Curves

Presented here in figure 3 is the loss curves for the train data during training, and validation data that was periodically tested throughout training. See that after around 60000 batches, the test loss starts to plateau, and our model would appear to halt learning. Meanwhile, the train loss also seems to plateau in a similar place. Since our test loss never starts to actually increase, our model has not necessarily overfit to the training data. It seems that the mitigations to prevent overfitting have worked to some extend, however those same mitigation (such as learning rate decay and L_2 normalisation) may also play a role in hindering to model from further learning.

The training curve is extremely 'jumpy', and the jumpiness does not seem to improve over training. This suggests that the model consistently works better on images with certain qualities and worse on images with certain other qualities, and moreover does not seem to learn from its mistakes. More detail can be found in section IX.

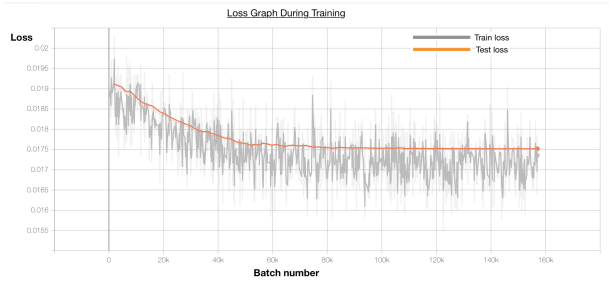

Figure 3. The train and test loss curves acquired during training

Figure 4 shows the filter learned by the first convolutional layer after training. The filters show little variance between them, besides some more blank filters, and it is difficult to see how these would be very useful in distinguishing information of the image.

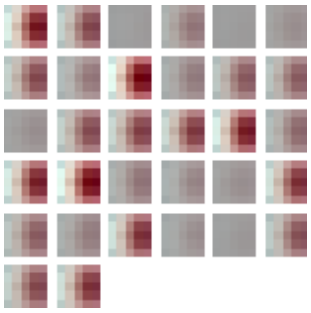

Figure 4. Filters learned by the first layer after training

IX. Qualitative Results

Shown here are saliency predictions output by the model. As suggested by the loss curve, the model varies noticeably in the level at which it performs. There is a strong centre bias, meaning pixels closer to the centre of the image will likely be of relatively high salience compared to the rest of the image according to the model. This is most likely due to an disproportionately high number of images in the dataset having higher salience closer to the centre. Shown in figure 5 is an example of where the model seems to perform decently. Notably, in this example, the point of highest salience in the image is right in the centre, which clearly takes advantage of the centre bias regarding accuracy; it is somewhat unclear whether the model has actually done a good job in this case, or just gotten lucky so to speak.

Shown in figure 6 is an example where the model appears to have not performed so well. While there is some correlation between the predicted saliency map and its ground truth, the output is for the most part dominated by the centre bias. This renders the prediction rather useless, as the consistency of the prediction with its ground truth is barely noticeable compared to the misinformation shown by the bias.

Figure 7 shows another example of failure, although one that tells a very different story. See there is an area of extremely bright pixels in a place where the ground truth shows very little to no salience. This is possibly due to exploding gradients conjured in the networks architecture [12].

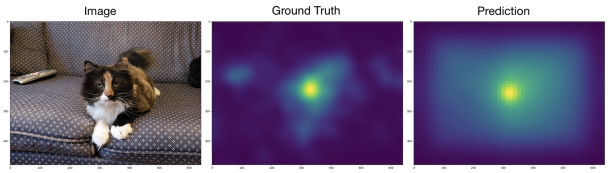

Figure 5. Example of acceptable prediction

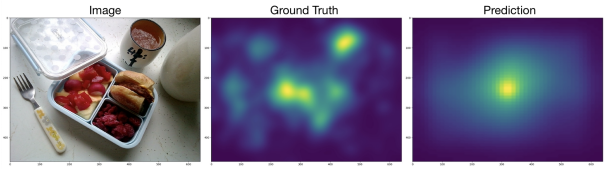

Figure 6. Example of failure due to strong centre bias

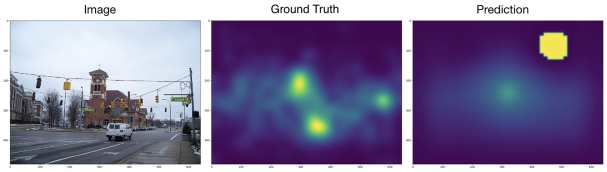

Figure 7. Example of failure due to exploding gradients

X. Improvements

As an improvement on this architecture, I implemented two dimensional batch-norm layers after each of the convolutional layers. Batch-norm is a method of reducing internal covariant shift [13], meaning reducing the change in distribution of the network's neuron's activation due to changing parameters. This acts as a regulariser, which could further help with overfitting and problems such as exploding gradients. This improvement seemed fitting for the task at hand for these reasons, as it will potentially stop some inputs from resulting in wild fluctuations in saliency prediction, such as the case with figure 7.

To implement this, I simply defined the three layers using the Pytorch library in the initialisation function of the model, and utilised them during the forward pass.

This ended up in notable improvements from the original architecture, increasing each measure of accuracy by around 4%.

Note, however, that this does not address the issue of centre bias. My initial attempt at improvement was one to address this problem, and was inspired by the regularisation term on the loss function used in *Bruckert et al* [4]. Unfortunately, having implemented this measure, accuracy went down very marginally, which also seems to be the case in *Bruckert et al*'s results.

Test Accuracies with Improvement			
SALICON (val)	CC	AUC Shuffled	AUC Borji
Pan et al [2016]	0.58	0.67	0.83
Examiners' replication	0.65	0.55	0.71
My replication	0.65	0.55	0.71
With improvement	0.69	0.59	0.73

XI. Conclusion and Future Work

Pan et al proposed a solution for predicting saliency maps based on an end-to-end approach using convolutional networks. While convolutional networks have been shown in a vast number of cases to produce excellent results, it would seem based on the results shown in the original paper, and the results produced in this replication, that there is still a long way to go from here in applying them to saliency map prediction. This is for a number of reason, first and foremost of which seems to be a lack of data. Data is the driving force of regressive algorithms, and to perform such a complex task without sufficient data is bound to show troubles.

This paper only looks at a shallow architecture; expanding our scope of potential architectures to deeper models, and more complex initialisation of the model to introduce some prior knowledge could go a long way in terms of performance [14].

An idea of implementing prior knowledge could be to first train the network for a simpler, but highly related task such as edge detection or corner detection. This could do great things in priming the weights for saliency prediction to be more effective, due to them being trained to find features that could be of high consequence to an image's salience.

References

[1] "Shallow and Deep Convolutional Networks for Saliency Prediction", Junting Pan and Kevin McGuinness and Elisa Sayrol and Noel O'Connor and Xavier Giro-i-Nieto, 2016.
[2] "The handbook of brain theory and neural networks", Chapter: "Convolutional Networks for Images, Speech, and Time-Series", Yann LeCun and Yoshua Bengio, 1998.
[3] "Benefits of Depth in Neural Networks", Matus Telgarsky, 2016.
[4] "Deep Saliency Models : The Quest For The Loss Function", Alexandre Bruckert and Hamed R. Tavakoli and Zhi Liu and Marc Christie and Olivier Le Meur, 2019.
[5] "Tidying Deep Saliency Prediction Architectures", Navyasri Reddy and Samyak Jain and Pradeep Yarlagadda and Vineet Gandhi, 2020.
[6] "End-to-End Saliency Mapping via Probability Distribution Prediction", Saumya Jetley, Naila Murray, Eleonora Vig, 2018.
[7] "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification", J. Salamon and J. P. Bello, 2017.
[8] "Backpropagation and stochastic gradient descent method", Shun-ichi Amari, 1993.
[9] "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning", Chigozie Enyinna Nwankpa and Winifred Ijomah and Anthony Gachagan and Stephen Marshall, 2018.
[10] "Implications of Pooling Strategies in Convolutional Neural Networks: A Deep Insight", Shallu Sharma and Rajesh Mehra, 2019.
[11] "Maxout Networks", Ian J. Goodfellow and David Warde-Farley and Mehdi Mirza and Aaron Courville and Yoshua Bengio, 2013.
[12] "On the difficulty of training Recurrent Neural Networks", Razvan Pascanu and Tomas Mikolov and Yoshua Bengio, 2013.
[13] "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", Sergey Ioffe and Christian Szegedy, 2015.
[14] "Predicting Human Eye Fixations via an LSTM-based Saliency Attentive Model", Marcella Cornia and Lorenzo Baraldi and Giuseppe Serra and Rita Cucchiara, 2018.