Gaze Following Saliency Model on Ad-Based Eye-Tracking Dataset

CSP354

William Webb
950011@swansea.ac.uk

October 2019

Abstract

Gaze fixation prediction is a rapidly developing field of research that has benefited tremendously from a combination of deep saliency models and expansive datasets. However, these improvements to models still fall short of the goal of consistent and accurate prediction performance, and the gaze fixation data are limited in scope in comparison to general image datasets. In this paper, I aim to close the gap between model predictions and ground truths, by creating a novel dataset for the development of more complex models, along with providing a public implementation of a top-down saliency model to test the dataset with.

1 Introduction

Figure 1: Gaze fixation heatmaps collected from a test run of this project's eye tracking experiment.

Visual attention is a powerful tool that allows a human to ignore unimportant information and dedicate their focus on only the regions of greatest interest. Even in images of large crowds or densely detailed objects, humans will be drawn to the most salient information. The past few years have seen great success in predicting what regions a human would be most drawn to, thanks to advances in model architectures and a prevalence of large datasets[1, 2].

The functionality of saliency models in predicting a human's gaze fixations would be helpful in tasks related to ads and visual design. Ads are designed with their ability to attract attention and convey a certain message in mind, which makes it hard to judge an ad's effectiveness without focus groups, A/B testing, or other market research[3]. By developing new datasets and models, we can more effectively predict how humans perceive the world, and use that information to design better interfaces and adverts.

Saliency models can be categorised into "bottom-up" and "top-down". Bottom-up (*a.k.a* task-agnostic) saliency models find objects and regions in an image that by some measure are more salient than other parts[1, 2]. Many current state-of-the-art models accomplish this by building and improving upon image recognition models [4, 5]. This format has proven effective at creating models that can predict human fixations across many datasets[6, 7, 8]. Top-down (*a.k.a* task-specific) models expand upon this approach by considering how more active attention might influence saliency, e.g how objects within an image can influence the saliency of other objects in the scene. One such model is [9], which more accurately predicts how a person in the image can 'push' attention to other objects in the scene, as the viewer of the image follows their gaze.

The datasets themselves have also advanced over the past few years, thanks to a new abundance of data to work with. Previously, visual attention and saliency datasets were orders of magnitude smaller than comparable sets for other fields, due to the cost of collecting annotations and eye tracking data [1]. New datasets have been able to overcome this problem by using cheaper hardware like webcams instead of dedicated hardware[10], or by using analogues such as mouse movement to provide similar ground truth fixations to eyetrackers[11, 12].

There are a few issues with the publicly available saliency models, and the datasets that exist to train those models on. Even with recent eye-tracking datasets and new approaches on how to collect fixation data, there is still a large gap in size between those and general image datasets such as MS COCO and SUN[13, 14]. This restricts how effective models can be trained to be. Top-down models are themselves limited, either by how well a dataset has been annotated (if it makes use of extra labels to make predictions), or by how the model architecture cannot predict general object interactions beyond the scope that it has been trained for. Finally, many of these models do not provide source code, or what is available is built on a variety of different platforms, which makes it much harder to confirm and compare results.

If more complex top-down models are to be developed that are capable of interpreting object interactions beyond a narrow subset, then more datasets are necessary. The development of models will also depend on the ease of which works can be replicated and reviewed,

In this paper, I propose the following:

- To design an experiment to collect eye-tracking data for a novel dataset,
- To collect gaze-fixations over a full run-through of the novel dataset,
- To evaluate the novel dataset against existing works,
- To implement and publicly release Gorji and Clark's Attentional Push, and
- To evaluate the predictive performance of my Attentional Push implementation.

The rest of the paper is organised as such:

In 2, I present the related work on datasets and on saliency models, alongside the methodology and tools that I use to plan and implement the model and eye-tracking experiment,

In 3, I cover similar efforts in the field of visual attention,

4 is about the components that make up the project,

In 5, I cover my planning and management of this project,

6 discusses the components of the project that I have completed, and details how I have done so,

In 7, I consider some final thoughts,

And in 8, I conclude.

2 Background

In this section, I will cover what some of the terms that I use in this paper, alongside the software that I use. This will be divided into Terms and Meanings and Tools.

2.1 Terms and Meanings

Saliency, as a term, is only a small part of the science of visual attention. Saliency refers to the features that makes an object stand out from its neighbours. In an image, the most salient object will be the one that the viewer will pay attention to first, regardless of whether they have consciously chosen to do so. Attention itself includes both this instinctive response to salient objects, along with more conscious mechanisms, where the viewer is actively paying attention to an image and its contents.

A way to differentiate between the two of these types of attention is through the terms "bottom-up" and "top-down". Bottom-up saliency refers to the low-level features of an image, such as the colour, brightness, shape, and other characteristics that might define it from the rest of the image. For humans, these cues are responded to instinctively without conscious thought, and is referred to as pre-attentive processing[15].

Top-down saliency depends on the viewer's recognition of objects and their context in the image. This might also be referred to as "task-dependent", as the viewer's attention might be influenced by something they are doing; if one is driving, then the viewer will naturally be alert to road hazards and signals over other stimuli[16].

In the context of this research, we can determine saliency from the gaze of the viewer; by using an eye-tracker to capture where a viewer looks on a screen (referred to as a gaze fixation), we can determine what regions in an image that the viewer was looking at are most salient. By capturing this data, we can use it to train deep neural networks to respond similarly.

A model's final estimate of what is most salient in an image is represented by a saliency map. This is similar to a heat map, representing regions in an image which the model finds most salient. By comparing this map against the regions recorded from a viewer looking at an image, we can determine how accurate a model is at predicting an image's saliency.

2.2 Tools

Each part of my project required several other pieces of software to accomplish. For the design and creation of the experiment, I went with OpenSesame, an open-source experiment builder [17]. It was a very simple program that allowed quick prototyping of various experimental setups, and also supported eye-tracking data collection through PyGaze[18].

Part of the data that the experiment collects is given in the form of a log file. The .tsv logfile can contain several batches of images, depending on when the participant wishes to take a break. These are saved alongside the a .csv and calibration image detailing the session. The .tsv is what contains the gaze fixation information, which is used with PyGazeAnalyser[18] to create visualisations of the fixation data.

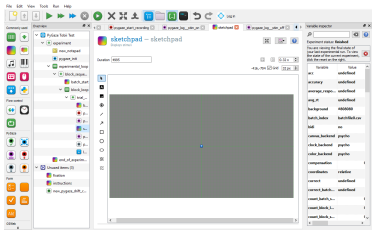

Figure 2: The Open Sesame behavioural experiment builder.

The original implementation of the Attentional Push[9] network was claimed to have been done in Caffe, a C++ ML library[19]. However, as no source code was published alongside the paper, I was free to choose another platform for my own implementation. As such, I chose Tensorflow [20], due to some of my past familiarity with Python gained from working on the experimental part of the project, and because of the integration of the Keras[21] neural-network library, which makes the implementation of DNNs much easier.

3 Related Works

In this section, I cover the current state of the field, in both datasets and models. These are split into their respective subsections.

3.1 Datasets

	MIT300	MIT1003	CAT2000	OSIE	POET	Turker Gaze	SALICON
Image Count	300	1003	4000	700	6270	20608	10000
Image Types	Natural	Natural	Mixed	Natural	Natural	Natural	Natural
Observers (per image)	39	15	24	15	5	3	60
Annotations	Gaze Fixation (Eye Tracker)	Gaze Fixation (Eye Tracker)	Gaze Fixation (Eye Tracker)	Gaze Fixation (Web Camera), Segmented Objects, Semantic Attributes	Gaze Fixation (Natural), Object Class	Gaze Fixation (Web Camera)	Gaze Fixation (Mouse), Category

Figure 3: Datasets built using eyetrackers must trade off either number of images or number of observers per image. Analogues for conventional eye tracking makes it easier and cheaper to collect fixation data.

There is a variety of datasets available for saliency models to train and test upon. Previously, datasets for attention research were significantly smaller than equivalents in other fields, due to the cost and difficulty in annotating images and collecting attention and saliency data[10]. Recent datasets have been able to resolve this issue by using different data collection methods, such as using mouse-based input as an analogue for eye tracking[11, 12], or by substituting dedicated hardware with common webcams[10]. This increase in size of public datasets is a necessity for deep saliency models, which can have trouble generalising themselves beyond a training dataset if it is too small(referred to as overfitting).

As such, saliency models like [7, 6, 8] were initially trained on SALICON, before fine tuning on smaller datasets with real eye-tracking data[22, 23].

This example of transfer learning, where a model trains on larger, more general datasets, before specialising on the niche eye-tracking datasets, is how nearly all saliency models are trained. Transfer learning can also be of use for training other aspects of saliency. The Gazefollow dataset[24] is made up of over 120,000 images that are annotated with the locations of both the location of the eyes of a person in the scene, along with whatever they are looking at. This can be used to train top-down gaze-following saliency models like *Gorji et al.*'s [9].

Despite the advantages of transfer learning, the datasets that models must fine tune on leave something to be desired. Most datasets that make use of true eye trackers to collect gaze fixation annotations are generally small, and increasing their size ends up reducing the number of observers per image, which makes the dataset more vulnerable to outliers. Even CAT2000, which has both a large number of images and observers (4000 and 24 per image, respectively) is limited by its image types; It only has 200 images in any particular category. For models focused on predicting the saliency of natural images alone, this is not useful. My dataset of 4000 natural images would be suitable for this situation.

3.2 Saliency Models

There is a variety of different types of saliency models. Depending on whether these models use a bottom-up, top-down, or mixed approach, they might predict the most salient regions in an image by judging the inherent saliency of objects in those regions, or they might use the context of the scene (such as what people in the scene are looking at) to determine the most salient regions. Bottom-up DNNs work well, and recent models [7, 6, 8, 11, 25, 26] are sometimes indistinguishable from ground truth fixations. However, these models are not totally accurate; while they are certainly capable at determining the saliency of individual components of an image, they do not account for how those components might interact with each other.

As such, these models might be augmented with a top-down approach, to better recognise components of an image that a bottom-up approach might not catch, such as text or object interactions[2]. Many models have had success in implementing these "top-down" approaches in saliency models. **Attentional Push** [9] achieves this by augmenting a standard saliency model such as [26, 27, 28, 29, 30] with a separate network based off the VGG-16 architecture [4]. As a viewer's attention is affected by what people within an image might be looking at, this model takes that into account by creating a saliency map based upon that interaction. This map will then get recombined with the output of the standard saliency model to create an enhanced saliency map. This leads to an improvement in evaluation scores, regardless of which standard model Attentional Push augments.

Beyond mere augmentation, recent papers have also been able to propose linking bottom-up and top-down attention together for better performance in

object detection and segmentation[31, 32, 33]. **ZigZagNet**, for example, creates many top-down and bottom-up networks that exchange feature maps between each other. These fused feature maps are fed through the many stages with an end result of state-of-the-art accuracy in segmentation tasks. This opens some new opportunities in cooperative saliency models that can directly predict gaze fixations.

Even as more papers experiment with the uses of top-down models to enhance the results of saliency-related tasks, there remains a very large gap that saliency models must close if they wish to consistently achieve human-level accuracy in predicting saliency maps. If anything, the complexity of these models and their fine-tuning reveals that some saliency features are lost through the process[34]. It is clear that without the foundation of a large dataset with rich categories, the gap will remain there, no matter the model.

4 Methods

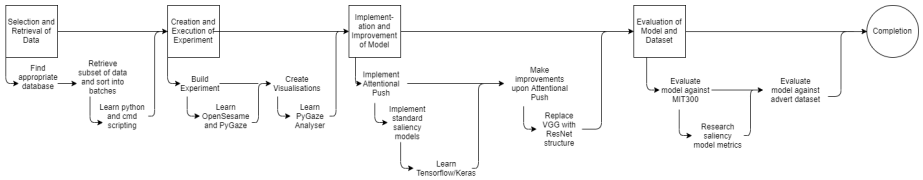

Figure 4: Workflow of project.

In this section, I will cover the components of the dissertation, and the steps I take for each one. These components will be split into the subsections

4.1 Dataset Selection and Preparation

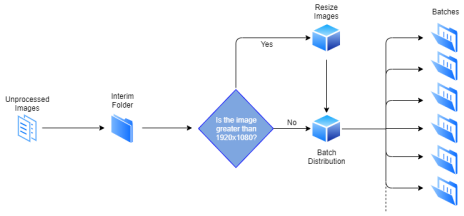

Figure 5: The image resizing pipeline

The first major component of the entire project is in selecting the images that will be used to construct the dataset. To select and organise the data, several python scripts had to be written. The images chosen from MS COCO [13] were 4,000 various images, including a mix of both human subjects and other

objects suitable to generalise Attentional Push's capabilities to. The selected images were taken from their source folder and placed in an interim selection folder, where they were resized for the experiment. This was a necessary step, as PyGaze's visualisation tool[18] could not handle image files that had a higher resolution than the dimensions of our monitor (1920x1080). As such, any images with a resolution greater than 1920x1080 were resized to equal or under those dimensions. Please note that any images already smaller than the resolution of 1920x1080 were left as is, and all images preserved their original aspect ratio. Once this resizing step was complete, images were moved to equally sized batches of 50.

4.2 Experiment Creation and Execution

The design of this experiment will have to take into account proper procedures in capturing fixation data, and so an analysis of the protocols and methods of other eye-tracking datasets[23, 22, 35] will be done.

The defining similarity between eye-tracking datasets is the time that each viewer is presented the image. This is often set to three seconds, which allows the viewer to instinctively look at the components of the image that are most salient. It is also enough time for the viewer to make connections between objects, such as a viewer following the direction of an actor's gaze to another object in the scene. For our own data collection, we will use a similar timing, with a short pause between images.

A participant sits down at the eyetracking station, approximately 65cm away from a monitor with a size of 48cm by 38cm, a resolution of 1920 x 1080 pixels, and refresh rate of 60Hz. The participant goes through a gaze calibration step before beginning a set of trials. Each trial begins with a centred dot on a grey background, with a message asking the participant to fix their gaze there. This continues for 5 seconds, before the scene transitions to a randomly selected image from a batch, which they will look at freely for 3 seconds. During this time, the participant's gaze is tracked with a Tobii X3-120 Eyetracker, which will record their eye movements at a sample rate of 120Hz and gaze accuracy within 0.4°[36]. This process repeats for the rest of the batch of 50 images. Saccades were classified as events where eye velocity exceeded 35°/s and eye acceleration exceeded 9500°/s^2. Each batch takes approximately 6 minutes to complete, with each participant completing 4 batches of images.

Figure 6: The experiment in use.

The data from the experiments is processed using PyGaze Analyser[18] to create visualisations of the gaze fixations. The data from this is an output

.tsv log file which contains 360 entries for each image, which detail whether the eyes were fixating on an area, saccading to another region, or invalid (due to a blink or looking away from the monitor). This can be used alongside the original image to create a variety of different graphs, such as fixation heatmaps and gaze scanpaths.

4.3 Model Implementation

For Attentional Push itself, the model is modelled as closely to Gorji and Clark's own work as possible, but the transfer to a new machine learning platform means that some aspects of the work are adapted. In the case that some aspect of the original model does not transfer over, then I present a working alternative.

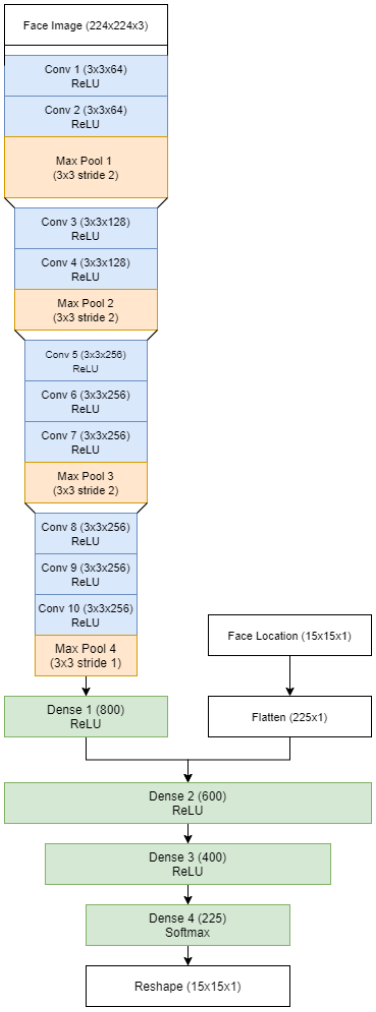

Figure 7: My implementation of the Attentional Push model

My implementation is more or less as similar as I can manage to make it, within the limits that Tensorflow has. I initialise the model first as VGG16 with the ImageNet[37] weights. Then, I create a shortened version by removing the last iteration of convolution/maxpool layers. I replace these with a new MaxPool with a stride of 1, before leading into a fully connected layer with batch normalisation on its output. I then reload the weights for the model. The output of this model is concatenated with the face location input, before it is run through several fully connected layers before ending in a soft-max layer to compute the logistic loss during training.

Just like Gorji and Clark's implementation, the input to the short VGG network is a cropped region of a person's face within the image. The bounds of the cropped region are defined as:

$$F = I(X_h - sW : x_h + sW, y_h - sH : y_h + sH, :)$$

The x and y values are provided with the annotations of the GazeFollow dataset, and the width and height values of the image are easily retrievable. s itself is the scale of the crop, which I have set to 0.25 as in the

original paper. F is then resized to 224×224 pixels before being fed to the network.

The face location input is given as a zero-initialised image $L \in R^{M \times M \times 1}$, with $L(\frac{x_h}{W}M, \frac{y_h}{H}M) = 1$. As M is equal to 15, this will be interpreted as a 15x15 blank image, with one region in that image active and representing where the person's head is in the image.

With the model implemented, the next phase was to begin training it. Gorji and Clark trained their model using a mini-batch stochastic gradient descent, with a batch size of 2, a momentum of 0.9, and a weight decay of 0.0005. All of these parametres were replicated in the optimiser I used to compile the model.

A change that did have to be made was the learning rates for the model. Gorji and Clark were able to set different learning rates for each layer in their model, which tensorflow didn't natively support. As such, I set all layers to have a learning rate of 1×10^{-5}, and disabled learning on the first two layers of the model, which was the most similar I could make the model.

To implement the mini-batching, I made use of the ImageDataGenerator class. This was the best method that allowed multiple inputs, both image and label, and a single output, being another label. The generator class I wrote randomly selected two images from the given directory (train or test), and returned it along with the appropriate head and gaze locations. Using this generator, the model trained for 1000 steps per epoch, over approximately 119 epochs.

Once the training finished, the model was saved and put aside, ready to be used to augment any given model.

5 Software Methodology

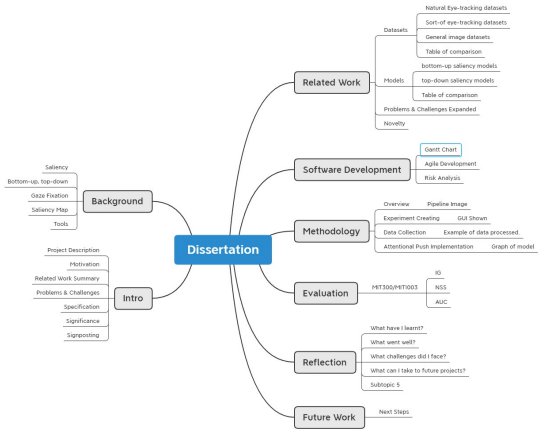

Figure 8: Project Mindmap

The structure I have used to plan out and manage my project revolves around the completion of small tasks from each of my components. I had originally thought to make use of a simple, "waterfall" cycle, but I rejected this for a few reasons. While waterfall development models are effective for small teams or individuals that have a sequence of development goals that are unlikely to change, I would not be able to effectively complete and evaluate each stage without the presence of other components. The waterfall model would have also required a clear outline of each of the components ahead of time. As I was not aware of what tools or libraries I would need to build each of the components, I would have had to spend a lot of time researching options and alternatives.

As such, I decided to adopt a more agile methodology instead. This approach would allow me to break each component up into smaller sub-tasks, that I could complete when I needed them. I would also be able to rapidly prototype and test solutions for each task in a smaller time-frame than a cycle in a waterfall model would have allowed. I created a workflow chart that represents each of the components, alongside the various subtasks each are composed of, and how they flow into one another.

Component	Point of Failure	Likelihood	Impact	Description	Remedy
Data Collection	Wrong Format	3	1	The images that we'll be collecting for the dataset may well be corrupted or malformed in some way that they'll require further processing before they can be used in the experiment	Python provides a powerful image manipulation library that will make it easy to handle images that don't fit our requirements. As the ADVISE dataset's images aren't outright broken, we shouldn't expect to deal with anything more than changing .png to .jpg where necessary.
Experiment Design and Deployment	Broken Visualisation Tools	2	2	It's rare, but not totally out of the question, that the output we receive from the experiment's eye-tracking will be in some way incompatible with the PyGaze Analyser tool.	It would be wise to test the visualisation tools with some sample experiment output to ensure it does work. If it does happen to break later, then the experiment output will need to be examined for obvious errors. Contacting the developers of the visualisation tool to report this error may also help in resolving the issue.
	Small Sample of Participants	3	5	There is a chance that the experiment study may not get a large sample of participants to collect eye-tracking data. A small number of participants would be incapable of gathering data on all 4,000 images in the proposed dataset, resulting in its incompletion.	It may be worth gathering a number of friends or associates who can run through at least 1 batch in the experiment to pad the sample size with extra observers. Better incentives for participants may also be necessary.
Model Implementation and Refining	Errors in Training/ Testing Model	1	3	It's highly unlikely that, if I follow the architecture laid out by Gorji and Clark, my own implementation will run into errors during training or testing. If it does happen, however, then any further evaluation might be difficult to justify.	Prototyping the Attentional Push model with smaller training datasets might be worth it, to ensure that basic functionality can be achieved by my implementation. It may also be worth it to inspect my standard model implementations for errors.
	Issues in Refining Model	5	2	I'm no expert in machine learning, so it's likely that I'll run into difficulties when I begin implementing the new ResNet architecture to replace Attentional Push's original model. If the issues persist, I may find myself unable to complete this goal.	Despite the high likelihood of this problem, it should be easily resolved. There is a wealth of information on machine learning available to me, through online resources and knowledgeable associates.
Model & Dataset Evaluation	Inconsistencies in Evaluation	2	2	The final step of the project could find difficulties in accurately evaluating the scores of my model implementations against other models. This would make it difficult to come to a conclusion on the performance on my own models.	The MIT300 benchmark is simply another dataset that we'll be running our models on. Assuming that the models are running in the first place, they shouldn't face any troubles here. Test runs with standard saliency models that can be compared with published results might be worth it to confirm if we're at fault.

Figure 9: Risk Analysis.

As part of my project management, I have also created an analysis on the likelihood and impact of failure of each of the main components. This is not a rating of which of these stages is the most important, as no stage in this project can function without the others, but instead what the effect is on the project if one stage does happen to fail in some regard. Of all these risk factors, only one is highly likely, and its impact on the project is relatively low and easily resolvable.

The project timeline itself stretched over the course of several months, beginning with the design and implementation of the original experiment. Each component of the project was made up of several sub-components for the prerequisites to accomplish each one; as mentioned before, some sub-components were completed out of order, to better develop other aspects of the dissertation.

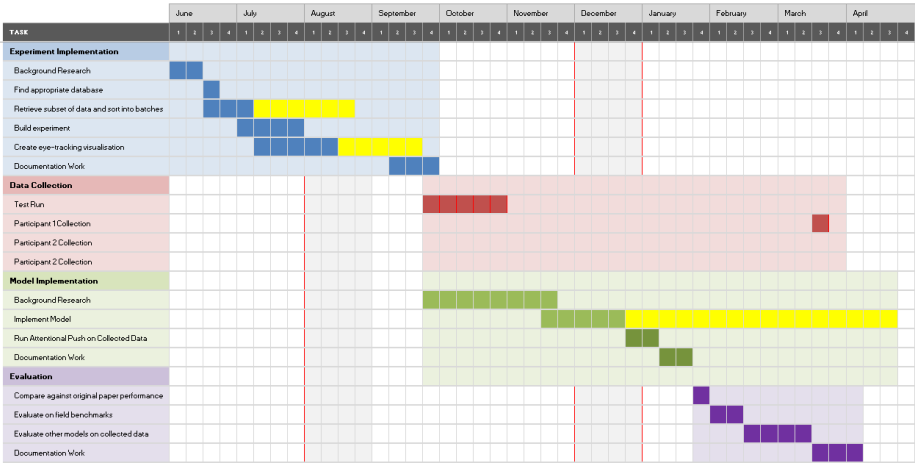

Figure 10: Gantt Chart. Yellow represents time past milestone.

6 Results and Evaluation

In this section, I will cover the results for each component of my project.

6.1 Experiment Implementation

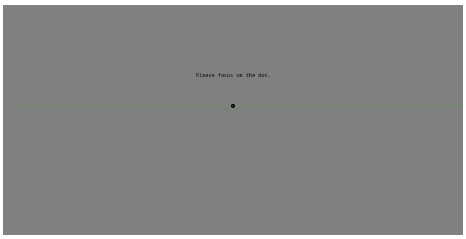

Figure 11: A prompt screen from the experiment

The experiment's final form was remarkably simple. To the participant, the only screens that ever ended up being shown was calibration, the button prompt screen to begin the experiment, and the image screen. The only occasions of failure due to the experimental software was a compatibility error with

OpenSesame's version of python compared to what the eyetracker was running on: this error was solved simply by starting the experiment again, and only occurred very rarely during the participant's sessions.

For the purpose of future data collection, this experiment is an excellent tool, and I plan on using it for future participants in expanding this dataset.

6.2 Annotation Collection

The end result of the experiment was good. The size of the dataset is very large, requiring hours of work, and so breaks were given in between larger batches of images, to prevent fatigue. Overall, the final experiment took approximately ten hours to complete over the course of two days. The participant was paid commensurately for their efforts.

Visualising the dataset presents some of the advantages that this dataset will have over existing works. The dataset, having been pulled from MS COCO, contains a variety of different image subjects. Unlike CAT2000, which categorises images based on an attribute of the image itself (cartoonish, abstract, natural), this dataset is entirely natural, and instead can categorise based on objects within the image.

Figure 12: Gaze fixation heatmaps from the participant's first session of the experiment.

The advantages of such a dataset are in being able to develop new top-down models that can generalise their predictions beyond specific object interactions. One of the requirements of Attentional Push is annotated inputs that include the locations of heads in the image. If Attentional Push can be generalised to all object interactions, then a dataset that already contains a variety of subjects in entirely natural settings would be a boon to training.

6.3 Attentional Push Model

Evaluating the Attentional Push model on its own was a little harder than one might think. Without a saliency model to plug into, the best way to evaluate it was simply against the validation images in the GazeFollow dataset. This revealed another error with my model that I hadn't previously seen, to do with the unlikelihood of the results:

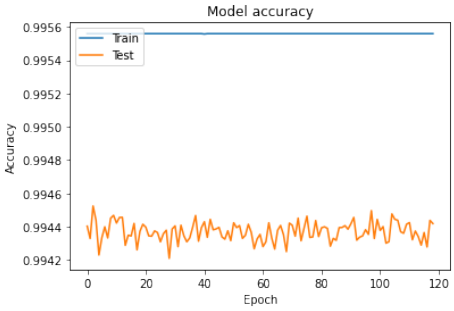

Figure 13: Model Accuracy.

Simply put, the model is incredibly accurate. This can be explained by the VGG's imagenet weights already encoding saliency information for the purpose of image recognition, such as what He et al suggest in [34].

Beyond this, I cannot evaluate the Attentional Push model further.

7 Reflection and Future Work

My experience in developing each of the components of this dissertation varied greatly. Developing the experiment and collecting the data afterwards proved to be the simplest part of the process, whereas the development of the models proved to be the most difficult and time consuming.

Over the course of this project, several components needed to be changed to better fit my results. The most obvious change that I had to take into account when working on this project was the COVID-19 outbreak. My original risk analysis chart did not take into account a global epidemic and national lockdown, and as such I was unprepared when such an event occurred. While I was able to finish off several components of this project in advance, several key aspects of my project needed to be adapted. My data collection also needed to be cut short when the University went into a dormant state, as I no longer had access to the lab facilities and the eye tracking equipment, but not before I was able to collect one entire sample of the 4,000 images from a participant.

The most disappointing part of the project was my inability to truly present an evaluation of the performance of my model, even with all I have put into it. As the Attentional Push model took so much time to work with and develop, it ate into the time to implement the saliency models that Attentional Push

was meant to augment. Those saliency models themselves presented a similar problem as Attentional Push did, being hard to convert to Tensorflow if they provided source code, or being difficult to implement outright if no code was given. In this case, my original estimate of the risks of problems with tensorflow, and how easy it would be to resolve such issues, was massively off. As it stands, I was unable to meet my initial expectations of work for this project.

However, I have still been able to make good progress in laying a foundation for a future project. The dataset I have begun to collect has a legitimate use in developing more advanced saliency models, and my Attentional Push implementation will be useful as a baseline for how I can augment bottom-up saliency models to better understand task-specific mechanisms in visual attention. In the future, I can continue to expand upon my dataset with more observers until it is truly ready, and then make use of it in the development of new general top-down models.

8 Conclusion

In conclusion, this dissertation presents a novel eye-tracking dataset for use in developing generalised top-down saliency models. I have laid the foundation for future evaluation of the dataset with an implementation of a top-down saliency model for use in augmenting other models, for which I can publicly provide the source code for others to work on.

References

[1] Ali Borji. Saliency prediction in the deep learning era: Successes and limitations. *IEEE transactions on pattern analysis and machine intelligence*, PP, 08 2019.

[2] Zoya Bylinskii, Adrià Recasens, Ali Borji, Aude Oliva, Antonio Torralba, and Frédo Durand. Where should saliency models look next? volume 9909, pages 809–824, 10 2016.

[3] Young. *The Advertising Research Handbook*. Ideas In Flight, 2 edition, 2008.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

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

[7] Matthias Kümmerer, Thomas S. A. Wallis, and Matthias Bethge. Deepgaze ii: Reading fixations from deep features trained on object recognition, 2016.

[8] Sen Jia and Neil D. B. Bruce. Eml-net:an expandable multi-layer network for saliency prediction, 2018.

[9] Siavash Gorji and James J. Clark. Attentional push: Augmenting salience with shared attention modeling, 2016.

[10] Pingmei Xu, Krista Ehinger, Yinda Zhang, Adam Finkelstein, Sanjeev Kulkarni, and Jianxiong Xiao. Turkergaze: Crowdsourcing saliency with webcam based eye tracking. 04 2015.

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

[12] Nam Wook Kim, Zoya Bylinskii, Michelle A Borkin, Krzysztof Z Gajos, Aude Oliva, Fredo Durand, and Hanspeter Pfister. Bubbleview: an interface for crowdsourcing image importance maps and tracking visual attention. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 24(5):36, 2017.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Zitnick. Microsoft coco: Common objects in context. 05 2014.

[14] Jianxiong Xiao, James Hays, Krista Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. pages 3485–3492, 06 2010.

[15] Christopher G Healey et al. Perception in visualization. *Retrieved February*, 10:2008, 2007.

[16] Laurent Itti and Ali Borji. Computational models: Bottom-up and top-down aspects, 2015.

[17] Sebastiaan Mathôt, Daniel Schreij, and Jan Theeuwes. Opensesame: An open-source, graphical experiment builder for the social sciences. *Behavior research methods*, 44(2):314–324, 2012.

[18] Edwin S Dalmaijer, Sebastiaan Mathôt, and Stefan Van der Stigchel. Pygaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments. *Behavior research methods*, 46(4):913–921, 2014.

[19] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[20] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016.

[21] François Chollet et al. Keras. https://keras.io, 2015.

[22] Ali Borji and Laurent Itti. Cat2000: A large scale fixation dataset for boosting saliency research, 2015.

[23] Tilke Judd, Krista Ehinger, Frédo Durand, and Antonio Torralba. Learning to predict where humans look. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[24] Adria Recasens*, Aditya Khosla*, Carl Vondrick, and Antonio Torralba. Where are they looking? In *Advances in Neural Information Processing Systems (NIPS)*, 2015. * indicates equal contribution.

[25] Nian Liu, Junwei Han, Dingwen Zhang, Shifeng Wen, and Tianming Liu. Predicting eye fixations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 362–370, 2015.

[26] Eleonora Vig, Michael Dorr, and David Cox. Large-scale optimization of hierarchical features for saliency prediction in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2798–2805, 2014.

[27] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. A deep multi-level network for saliency prediction. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3488–3493. IEEE, 2016.

[28] Junting Pan, Elisa Sayrol, Xavier Giro-i Nieto, Kevin McGuinness, and Noel E O'Connor. Shallow and deep convolutional networks for saliency prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 598–606, 2016.

[29] Nicolas Riche, Matei Mancas, Matthieu Duvinage, Makiese Mibulumukini, Bernard Gosselin, and Thierry Dutoit. Rare2012: A multi-scale rarity-based saliency detection with its comparative statistical analysis. *Signal Processing: Image Communication*, 28(6):642–658, 2013.

[30] Jianming Zhang and Stan Sclaroff. Exploiting surroundedness for saliency detection: a boolean map approach. *IEEE transactions on pattern analysis and machine intelligence*, 38(5):889–902, 2015.

[31] Di Lin, Dingguo Shen, Siting Shen, Yuanfeng Ji, Dani Lischinskiand Daniel Cohen-Or, and Hui Huang. Zigzagnet: Fusing top-down and bottom-up context for object segmentation. *Conference on Computer Vision and Pattern Recognition (Proceedings of CVPR 2019)*, pages 7490–7499, 2019.

[32] Wenguan Wang, Jianbing Shen, Ming-Ming Cheng, and Ling Shao. An iterative and cooperative top-down and bottom-up inference network for salient object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[33] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[34] Sen He, Hamed R. Tavakoli, Ali Borji, Yang Mi, and Nicolas Pugeault. Understanding and visualizing deep visual saliency models, 2019.

[35] Tilke Judd, Frédo Durand, and Antonio Torralba. A benchmark of computational models of saliency to predict human fixations. 2012.

[36] Accuracy and precision test report. https://www.tobiipro.com/siteassets/tobii-pro/accuracy-and-precision-tests/tobii-pro-x3-120-accuracy-and-precision-test-report.pdf, 2015.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.