Deep Visual Attention Prediction

Suraj Maniyar
Electrical and Computer Engineering
North Carolina State University
Raleigh, NC
spmaniya@ncsu.edu

Hardi Desai
Electrical and Computer Engineering
North Carolina State University
Raleigh, NC
hdesai2@ncsu.edu

Akshay Kalkunte Suresh
Electrical and Computer Engineering
North Carolina State University
Raleigh, NC
akalkun@ncsu.edu

Abstract— **In this report, we discuss about our attempt to replicate the implementation of Deep Visual Attention Prediction in a different framework as adopted in [1]. Authors have implemented using Caffe and we have built the architecture in TensorFlow and Keras(Using TensorFlow as backend). We discuss the motivation and related work done for predicting the eye gaze. System architecture adopted for learning is discussed in detail explaining the reasons for adopting it. Major emphasis of this report is in the implementation section where we have discussed the challenges faced in porting the model from one framework to another. We have obtained comparable results as the original paper considering the constraints in time and hardware resources.**

I. INTRODUCTION

Study of human eye fixations has been of long interest among the research community of computer vision and neuroscience. Humans have the ability to quickly gaze to important parts form the massive visual information received by eyes. This selective behavior of Human Visual System is represented by saliency maps (or attention maps) where brighter pixel indicates higher probability of human eye fixation. Predicting the attention map would have a broad application in image segmentation, object recognition, visual tracking, scene understanding to name a few.

Traditional methods for predicting the saliency includes two approaches: top-down approach and bottom-up approach. While bottom-up approach focuses more on low level features like color,texture or brightness, the top-up approach focuses more on the entities or objects present in the image. It can be noted that top-down approach requires contextual awareness about the image.

In this project we aim to predict saliency map using the bottom-up approach as discussed by [3]. With Convolutional Neural Networks(CNN) for feature extraction as well learning the human visual attention model, we have implemented a skip architecture based deep learning model. We extract the multilevel saliency response, ranging from the local to the global, using shallow to deep convolutional layers with small to large receptive fields.

We focused on replicating the results obtained by [1]. The authors have obtained the state-of-art results using the Caffe Framework in MATLAB. Using their openly available source code as a reference, we implemented the model in two different Framework: TensorFlow and Keras(using TensorFlow as backend) using Python.

Further part of this report is structured as follows: Section II describes the architecture adopted for implementation. Section III elucidates on the dataset used for training and testing the Convolutional Neural Network. Section IV explains the implementation details with focus on the challenges faced and its corresponding solution for porting the architecture in different framework. Results are discussed in section V. Section VI summarizes the report with conclusion.

II. RELATED WORK

The existing deep saliency models can be classified into following major categories:-

A. *Single-stream network*

It is the standard architecture of CNN based attention models, which is opted in by many saliency detection models. All other kinds of saliency detection can be viewed as variations of single-stream network. Describing the saliency cues at different levels by representing the input with different degrees of details from coarser to finer representations plays a key role in saliency detection Thus incorporating multi-scale feature representations into the attention model is usually opted in for saliency prediction.

B. *Multi-stream network*

This type of network learns multi-scale saliency information by training multiple networks with multi-scale inputs. The multiple network streams are parallel and may have different architectures, corresponding to multiple scales. The input data are simultaneously fed into parallel multiple streams. The feature responses from these streams are concatenated and given into one common layer which produces the final saliency. In this architecture, the multi-scale learning happens outside of the network. In the current architecture, the multi-level learning happens inside the network, via combining hierarchical features from multiple convolutional layers.

C. *Skip-layer network*

In CNNs, a skip architecture skips some layers in the network and feeds the output of one layer as the input to the next layer as well as some other layers. This allows for flow of information that was captured in the initial layers to the subsequent layers that might use it for reconstruction. The skip-layer learns multi-scale features ”inside” a primary stream.

D. *Bottom-up/top-down network*

A bottom-up approach appends together systems to give rise to more complex systems. A top-down approach essentially breaks down a system into compositional sub-systems in a reverse engineering fashion. The segmentation features are first generated via traditional bottom-up approach using convolution layers. Then the top-down refinement is proceeded for merging the information from deep to shallow layers into segmentation mask. The intuition behind this architecture is to generate high-fidelity object mask without loosing image information. This network can be seen as a kind of skip layer network, as different layers are linked together.

III. Architecture

The architecture proposed by the original paper is shown in figure 2. The model works in an encoder-decoder format. This is done to preserve the spatial information which is destroyed by the pooling operation after convolution. Initially, the encoder part captures high and low level features while convolving and down-sampling the low-level feature map, which decreases the size of feature maps. The decoder part up-samples the feature maps and constructs the output maintaining the same resolution as input.

A. *Proposed Model*

The proposed model is a full convolutional neural network, which aims at predicting pixel-wise saliency values. The encoder part is very much similar to the architecture of VGG-16 Network without the fully connected layers. The convolution layers are passed through down-sampling operation (max pooling) and then through point-wise non-linearity (ReLU). Because of the several convolutional layers that are stacked alternately in depth, hierarchical features are extracted from the image with increasingly large receptive fields.

Due to the spatial information that is lost during convolution, the coarse features are upsampled using deconvolution (transposed convolution).

To best capture the saliency, features are considered at multiple scales. The paper selects 3 layers from the encoder network for explaining the saliency at multiple levels. Each of the selected layer is then passed through a separate decoder network. This gives us 3 different attention prediction maps with the same size as the input image. Theses attention maps are then fused together before passing it through another convolution layer. The final output is used as the prediction which is compared with the ground truth to calculate the loss.

As described in Fig 1, our model consists of five convolutional layers acting as encoder, three deconvolutional layers as part of decoder.

IV. Dataset Description

It is well known that Convolutional Neural Networks are data hungry. For a convolutional neural network correctly predict, it is important that the data provided for training is sufficient for learning. We have used Saliency in Context (SALICON) openly available dataset for training and validating our model. It offers saliency annotations on the popular Microsoft Common Objects in Context (MS COCO) image database. We obtained 10,000 images for training and 5,000 images for validation. In this datase, the eye fixation

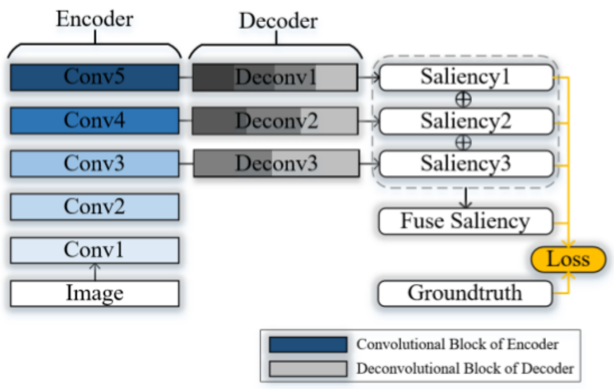

Fig. 1. Model Architecture Block Diagram

annotations are simulated through mouse movements of users on blurred images. The authors of [2] show that the mouse-contingent saliency annotations strongly correlate with actual eye-tracker annotations.

V. Implementation

A. *Data Preprocessing*

The SALICON dataset is composed of a folder with all the input images and a json file with the saliency map for each image. We extract the corresponding saliency maps for each of the images using the SALICON API. The SALICON API provides a tool to visualize the saliency map. We modify the scripts to save them as images instead after converting them to grayscale with pixel intensities between 0 to 255. We then resize all the images and their corresponding saliency maps to a size of 224 x 224 pixels.

B. *Caffe v/s TensorFlow*

The authors of the original paper have used Caffe Framework which offered them certain functionality while building the model that are not directly available in Keras and TensorFlow. In this section we discuss about the major difference we observed while porting the implementation from one framework to other.

1) Crop Layer: Caffe has a inbuild function of crop layer. This layer crops the elements of given layer in order to match the dimensions of given layer with the next specified layer except in one axis. In our implementation crop layer played an crucial role in fusing the multiple attention maps. The saliency maps generated from the three different deconvolutional layers are not equivalent to the dimension of ground truth attention map. Hence, we crop the generated attention map from the center to match its dimension the ground truth attention map. While keras had the functionality of crop layer, Tensor Flow Framework does not have this layer as on day writing this report. We had to build our own crop layer for Tensor Flow implementation.

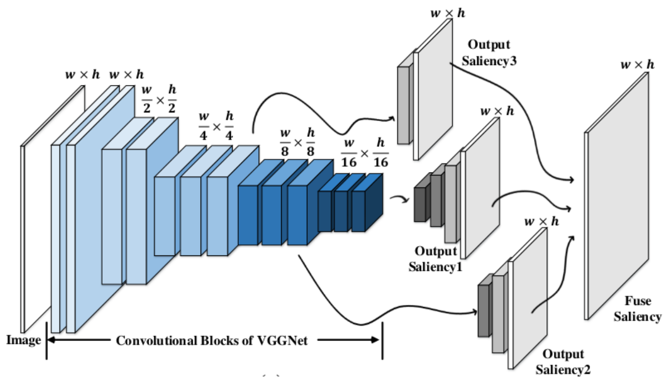

Fig. 2. Model Architecture

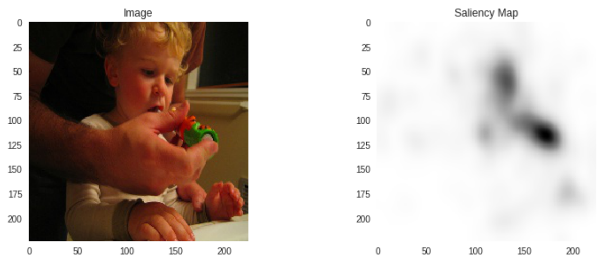

Fig. 3. Dataset

2) Padding: In order to maintain the dimension of the image after each convolutional layer, we pad the image with zeros. In caffe, one is allowed to to explicitly define the the number of pixels to be padded on each side of the input image. on the other hand, Tensor Flow and Keras only have two options for padding. First is valid, where the frame work understands that input image is already padded and no padding is required, and second one is same where the framework ensures that dimension of output image is same as input image given the filter length and stride. In our implementation, the authors have padded the input image to the model by 35 pixels on each side. This is an odd number of pixels and neither Tensor Flow nor Keras provide this functionality directly. In order to implement this, we have made extra layers where we manually pad zero to the image and change the input image before passing it to the convolutional layer.

C. Experimentation

In order for implementation to provide results as discussed in the original paper in Keras Frame Work (using Tensor Flow backend) , we made certain changes in while building the model. In the following section we discuss each of the changes in detail.

1) Binning of classes: Pixel wise prediction of saliency maps could be understood as a classification problem with number of classes equal to 255 as we are predicting a gray scale image. With lack of hardware resources as discussed in following section, we simplified our problem statement by binning the the ground truth images used for training into 10 different classes. This helped us in providing the proof of concept that the model built is able to learn the features for predicting attention map.

2) Loss: As observed in caffe implementation, authors have implemented sigmoid cross entropy for predicting the attention map after the deconvolutional layers. However, in our implementation, using the sigmoid cross entropy for calculation of loss did not yield good results. This could be attributed to back-end implementation of function which could different in both the frame work. So instead of using sigmoid cross entropy loss which is in essence activation layer using sigmoid function and calculation of cross entropy over it, we have used binary cross entropy. As it is visible in Fig 1, we have calculated the loss for each of the attention layer and averaged it out for back propagation.

3) Batch Normalization: As discussed in above Loss function, we removed the sigmoid activation function before predicting the attention map. We added a batch normalization layer after cropping the attention maps to the size of ground truth attention map. This helped in removing co-variance shift and improving our results.

D. Hardware Resources

Authors of the original implementation have achieved processing speed as little as 0.1 seconds on PC with a TITANX GPU and 32G RAM for the given model. We have implemented using the free on line resource Google Colaboratory which provides K80 GPU with 12GB ram for 12 hours. However, while using the Google Colaboratory we realized that the resource is provided on sharing bases and it often disconnects. As it is shared, we might get as low as 256 MB of RAM which would lead to resource exhaust error for our model. With multiple trials, we were able to run our model when the allocated RAM for a given runtime on Google Colaboratory was sufficient.

E. *Hyper Parameter Selection*

Due to constraints in hardware resources and time as discussed in above section, we have kept the batch size as 16, learning rate of 0.0001, number of epochs as 10 trained on 1000 train images with 100 images for testing. For Keras implementation we have Stochastic Gradient Descent for optimizing the loss function calculated using the cross entropy.

VI. Results

The original paper reports several error metrics for model evaluation. In this implementation we restrict our model to two error metrics - Earth Mover's Distance and Linear Correlation Coefficient.

A. *Earth Mover's Distance*

The Earth Mover's Distance is a measure of the distance between two probability distribution over a region. It is also the Wasserstein metric. Intuitively, given two distributions, one can be seen as a mass of earth properly spread in space, the other as a collection of holes in that same space. Then, the EMD measures the least amount of work needed to fill the holes with earth. Here, a unit of work corresponds to transporting a unit of earth by a unit of ground distance. The EMD has following advantages:

1) Naturally extends the notion of a distance between single elements to that of a distance between sets, or distributions, of elements.
2) Can be applied to the more general variable-size signatures, which subsume histograms. Signatures are more compact, and the cost of moving "earth" reflects the notion of nearness properly, without the quantization problems of most other measures.
3) Allows for partial matches in a very natural way. This is important, for instance, for image retrieval and in order to deal with occlusions and clutter.
4) Is a true metric if the ground distance is metric and if the total weights of two signatures are equal. This allows endowing image spaces with a metric structure.
5) Is bounded from below by the distance between the centers of mass of the two signatures when the ground distance is induced by a norm. Using this lower bound in retrieval systems significantly reduced the number of EMD computations.
6) Matches perceptual similarity better than other measures, when the ground distance is perceptually meaningful. This was shown by [2] for color- and texture-based image retrieval.

B. *Linear Correlation Coefficient*

The Correlation Coefficient is a statistical method that measures how correlated or dependent the two variables are. It can be used to interpret saliency and fixation maps, G and S as as random variables to measure the linear relationship between them:

$$CC = \frac{cov(S,G)}{\sigma(S)\sigma(G)} \qquad (1)$$

The value of coefficient ranges from -1 to +1 and a score close to -1 or +1 indicates a perfect alignment between the two maps.

C. *Obtained Results*

To infer the effectiveness of our implementation we evaluated the above given metrics during training. All the subsequent results were obtained from the model implemented in Keras. The obtained results are as follows:

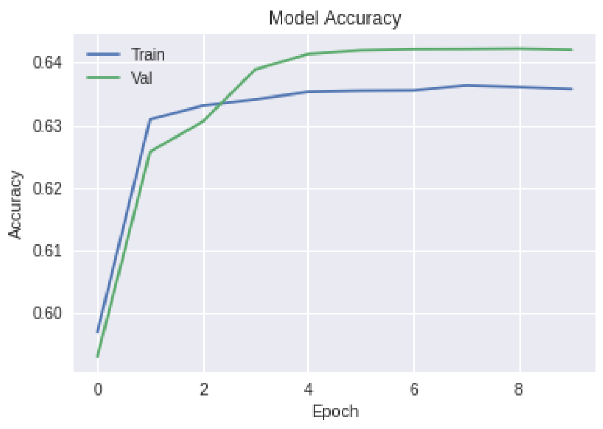

Fig. 4. Accuracy

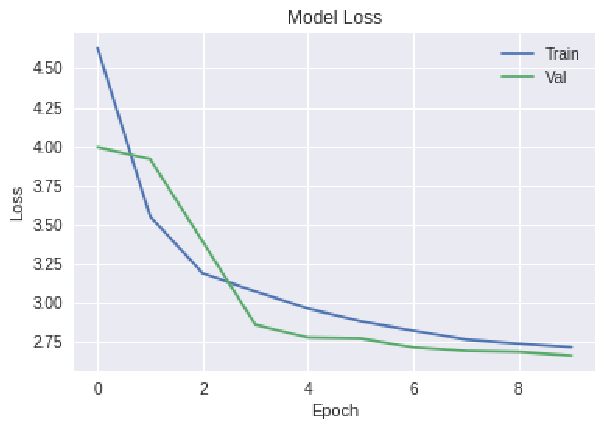

Fig. 5. Loss

The results show that the model is consistent as the validation accuracy increases and the loss decreases after each epoch as observed in Fig 4. and Fig 5. respectively. The Earth Mover's Distance decreases which shows that the predicted saliency map approaches closer to the target saliency map which is evident in Fig 6. Also the loss is seen as decreasing with each iteration suggesting that the weights are learned. The accuracy increases with the train accuracy greater than the validation accuracy. Linear Correlation Coefficient increases showing the high correlation between the predicted attention map and ground truth attention map as depicted in Fig 7. Figure 8 to 13 actually show are results for a given input image. The image displayed in Fig 8 is the

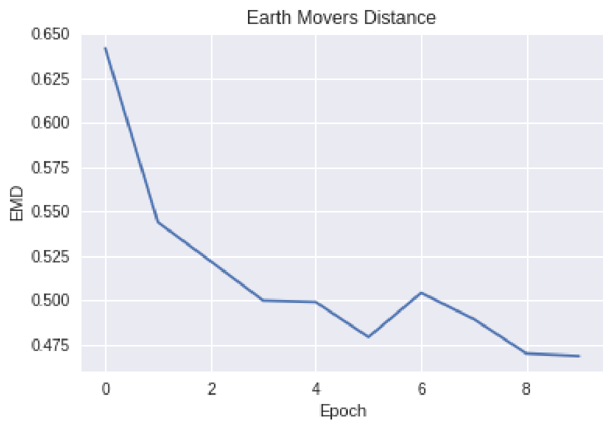

Fig. 6. Earth Mover's Distance

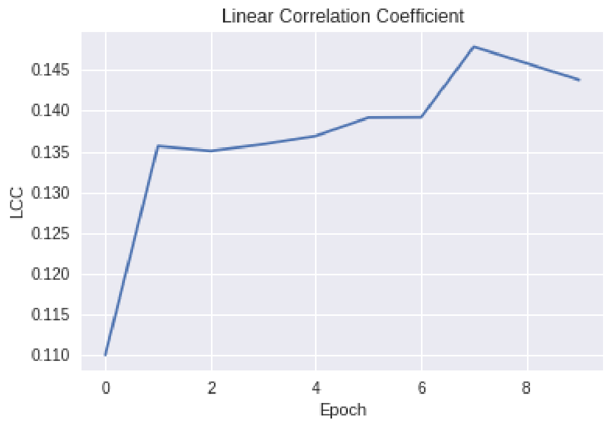

Fig. 7. Linear Correlation Coefficient

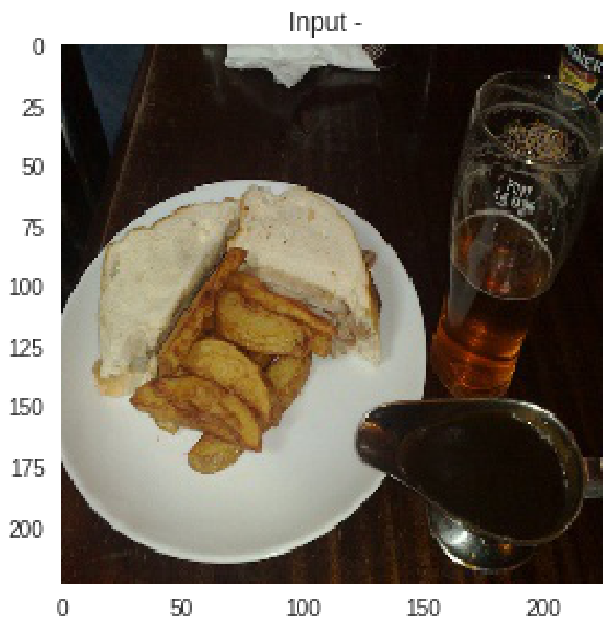

Fig. 8. Original Image

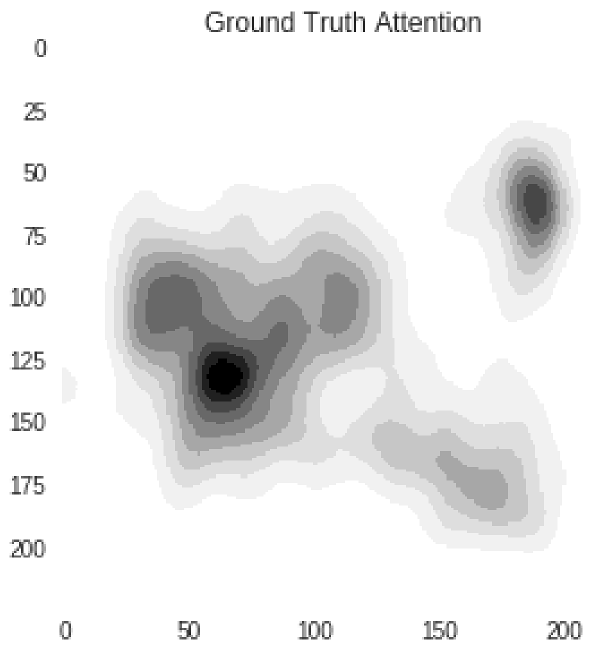

Fig. 9. Ground Truth Attention

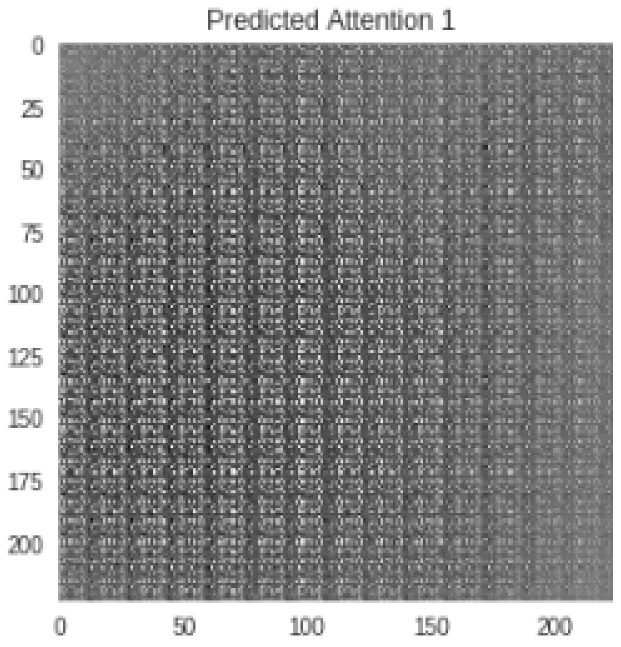

Fig. 10. Attention Layer-1

input image. It is obvious that the human attention is going to focus on the plate. Figure 9 is binned ground truth saliency map used by the network for learning. Figure 10 is attention map predicted by the first decoder network. Similarly, Figure 11 and 12 are the attention map predicted by second and third decoder network. The fused saliency map is depicted by Figure 13. Here, we can observe that the model tries to predict at attention as circle which is correct as the input had

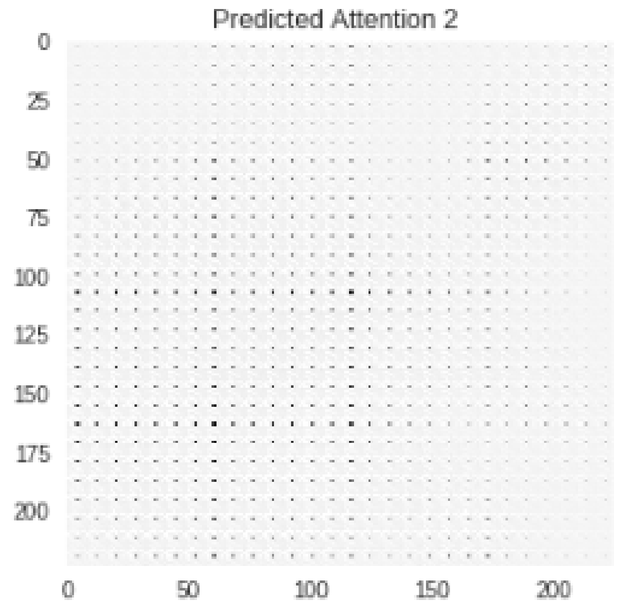

Fig. 11. Attention Layer-2

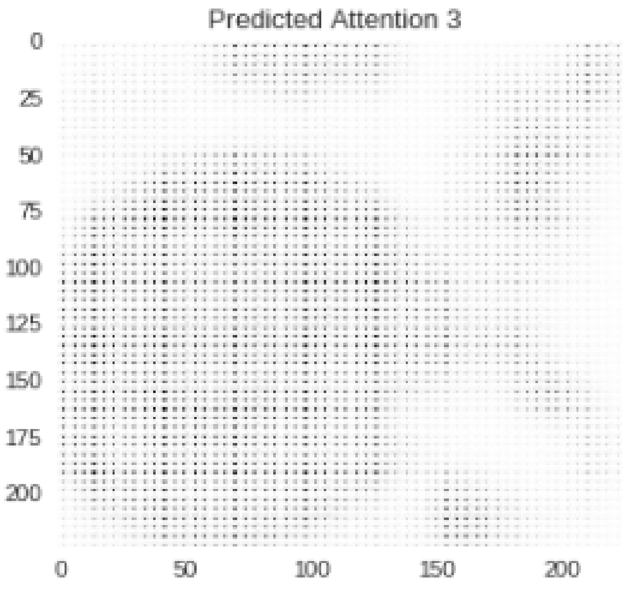

Fig. 12. Attention Layer-3

a plate.

VII. Conclusion and Future Scope

Observing the results, it can be concluded the we were successfully able to port the deep learning model from caffe to TensorFlow. We believe that if our model is provided with entire salicon dataset and trained on more number of epochs as discussed in original paper we could have obtained higher accuracy.

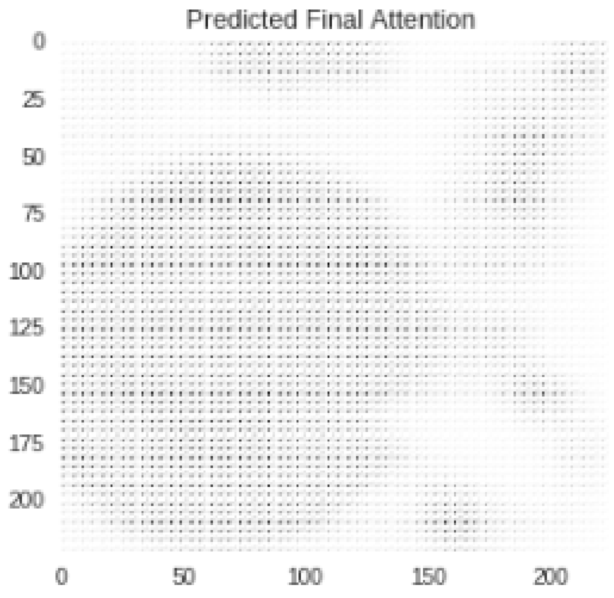

Fig. 13. Final Predicted Attention

Acknowledgment

We would like to acknowledge Dr. Tianfu Wu for providing us an opportunity for working on this project. We are thankful to him for guiding us in implementation and helping us understand the possible mistakes in our code when we were obtaining erroneous results.

References

[1] W. Wang and J. Shen. Deep visual attention prediction. IEEE TIP, 27(5):23682378, 2018.
[2] M. Jiang, S. Huang, J. Duan, and Q. Zhao, Salicon: Saliency in context, in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 10721080, 2015
[3] Y. Lu, W. Zhang, C. Jin, and X. Xue, Learning attention map from images, in CVPR, 2012, pp. 10671074.

Individual Contributions

The majority of our time was spent formulating the implementation. We had many brainstorming sessions after we obtained erroneous result on the first attempt of the implementation. Each one of us opined our interpretation to the rest of the group to figure out possible mistakes or reason for such results.

As we were working on google colaboratory, we were able to work on the same piece of code and debug the errors simultaneously. We divided our work by developing different functionality of the implementation. We attempted to replicate the model in TensorFlow as well as Keras.