Final Project Report

Computer Graphics

Shallow and Deep Convolutional Networks for Saliency Prediction

Submission Date: 6th, January, 2017

Group Members:

Name	Student ID
Lantao Yu	5140219071
Lingkun Kong	5140219016
Xiaotian Qi	5140219220

Table of Contents

Group Members:

Name	Student ID	Tasks Assigned
Lantao Yu	5140219071	Understanding the given Research Paper Coding of the proposed idea Report Documentation Presentation
Lingkun Kong	5140219016	Understanding the given Research Paper Coding of the proposed idea Implementation Report Documentation
Xiaotian Qi	5140219220	Understanding the given Research Paper Report Documentation Coding of the proposed idea Implementation

1. Introduction

The ultimate goal of computer vision research is to endow computers with the ability of modeling and understanding the physical world through the images and videos. One of the most interesting topics in this field is to teach computer to look like a human. More specifically, we want to predict the probability of visual attention for specific pixels on an image, which is also called saliency prediction. The traditional method of saliency prediction is to design a lot of hand-crafted features inspired by neurology and morphology studies, which requires a lot of prior and professional knowledge and the quality of features plays a crucial role.

Recently deep learning has been a great success and convolutional neural network has been widely used in visual pattern recognition, ranging from global image classification to local object detection. One of the most attracting advantages of convolutional neural networks is that we no longer need to design plenty of hand-crafted manually. Instead, with the hierarchical structure of Convnets, we can directly utilize an end-to-end learning framework to extract different levels of features automatically. Compared to the traditional feature-based methods, this approach is more general and outperforms the baseline significantly. However, one big challenge of this completely data-driven methods is that deep neural networks always require huge amount of data to train a satisfying model. Recently, two large datasets released from saliency prediction competitions have enabled deep learning methods to be applied to the saliency prediction task.

In conclusion, as the first proposal of using deep learning for saliency prediction, this work successfully formulates the saliency prediction problem to be an end-to-end regression problem and design a hierarchical neural network structure, which outperforms the traditional methods significantly.

1.1. Related Work

Traditional methods of saliency prediction widely use the hand-tuned features which have a loose connection to the biological visual systems. Nevertheless, this kind of features is always too simple to catch the complex patterns of human visual attention. For more complex and biologically plausible features, a broad class of bio-inspired hierarchical models were proposed, which has achieved better performance in a wide variety of pattern recognition tasks.

These kinds of hierarchical models actually belong to the more general class of hierarchical convolutional neural networks. *Eleonora* first propose a framework called *ensembles of Deep networks (eDN)* to blend the feature maps from three different convnets and ensemble those feature maps with a simple linear classifier. The biggest problem with this method is that it only performs a classification to every pixel (whether it is the fixation point or not) rather than a probabilistic prediction. Inspired by *eDN*, *Matthias* proposed a deeper network structure called *DeepGaze*, which uses deep convolutional layers of *AlexNet* network and replaces the fully connected layer with a linear parameterized model. The main contribution of *DeepGaze* is it is the first model that uses transfer learning, which is being proved as efficient, to transfer the knowledge learned from image classification to the saliency prediction task. Another architecture proposed by Liu focuses on combining the fully connected layer of three different parallel convnets

with a single layer. Some similar works mainly explore the possibility of using different convnets working at different resolutions to capture both the local and global saliency pattern.Other related works proposed some novel architectures for salient object detection.

1.2. Proposed Work in Assigned Research Paper

The research paper mainly proposed two neural network architectures. The lightweight shallow convent has more simple structures and less trainable parameters, which can be trained from scratch. The detailed structure of the proposed shallow convent is shown in Figure 1.

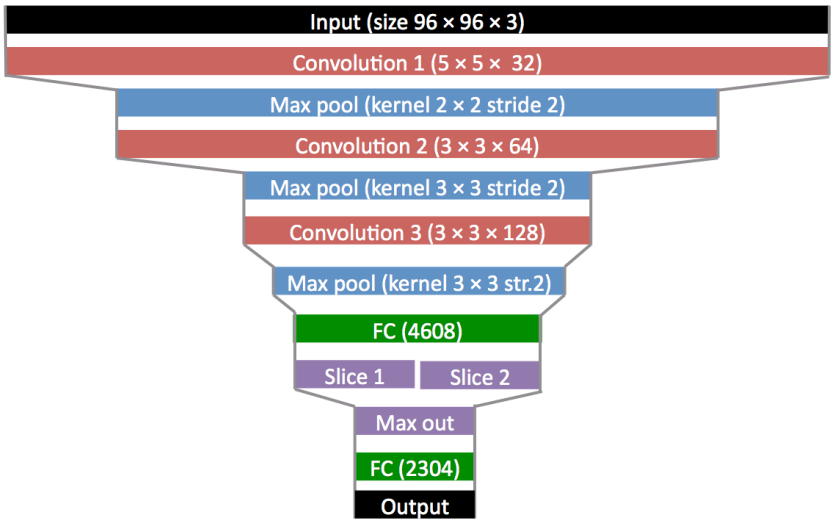


Figure 1. Illustration of Shallow Convnet for saliency prediction

There are five layers with trainable parameters, which are three convolutional layers and two fully connected layers. In these five layers, there are 64.4 million parameters in total. Each convolutional layer is followed by a rectified linear unit (ReLU) and a max pooling layer. The ReLU will add the non-linear characteristics into the model, which will increase the capacity a lot. The max pooling layer will reduce the parameters and keep the most relevant information forward. Since the model needs to be trained from scratch, a lot of strategies are used to avoid overfitting. Unlike the structure in *AlexNet*, it only uses three convolutional layers to do the feature extraction rather than five layers. Furthermore, it resizes the input picture size to [96 x 96] rather than [227 x 227]. More importantly, norm constraint regularization in the max out layer is essential to avoid overfitting.

Another proposed architecture, which has much better performance than the shallow one is the deep convent. The detailed structure of deep convent is shown in Figure 2. This approach follows the *DeepGaze* and also uses transfer learning methodology to utilize an existing very deep Convnet trained for image classification. The motivation of using parameters from other deep architectures is that a lot of work has found the parameters in the lowest level of a deep convolutional neural network always converge in a few epochs, which suggests that those layers perform low-level visual feature extraction such as the color or textures, which can be used in a wide range of visual pattern recognition task. Thus this work proposes a second Convnet which builds on the

pre-trained filters and combines them with new layers trained for saliency prediction. Since the low-level features can be commonly used for various tasks, this kind of transfer is reasonable and effective.

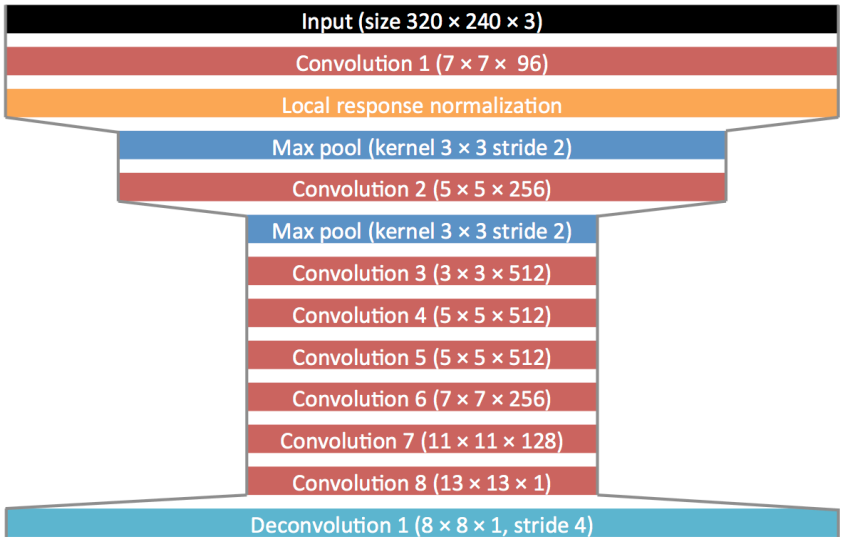


Figure 2. Illustration of the deep convolutional network for saliency prediction

1.3. Limitations of Proposed Work

As an initial proposal of applying end-to-end training strategy for saliency prediction, there are some limitations in the proposed framework. The detailed analysis is in the critical review subsection.

1.3.1 Critical Review:

Firstly, this work formulates the saliency prediction to be a regression problem and the loss is just the pixel-wise Euclidean loss. Since this kind of loss is built on every pixel, it only takes the local performance into consideration. As the human's fixation area must be smooth, the output saliency map must present a spatial coherence and a smooth transition between neighboring pixels. To model this characteristic and produce more natural saliency map, we not only need to consider the Euclidean distance between the predict value and ground truth on every local pixel, but also need to build a loss from a global perspective. For example, in generative adversarial nets, a discriminator learns to discriminate an input image from the real ones. In the GANs framework, we use another deep Convnets to extract high level features to learn to diagnose the loss from a global perspective. In conclusion, it is not enough that only using a Euclidean distance on every pixel to get a natural, smooth and accurate saliency map.

Secondly, given an input image, the proposed Convnets just perform a set of convolution and pooling operations, which are deterministic. This kind of deterministic transformation betrays the real pattern of human visual systems. For one reason, different people have different attention mechanisms. Specifically, given a same image, different people will absolutely have different saliency answers and the datasets were collected with crowdsourcing approaches, which means there are many people engaged in annotating the images saliency map and there are many random factors in the ground truth. For another reason, even for the same person, the saliency area could change from time to

time due to different environments and physical conditions. Consequently, we should start from an input sampled from some noise distribution and perform a set of convolution and deconvolution to get a saliency map with robustness and diversity.

2. Proposed Research Work

During the implementation of this project, we have overcome many limitations of the framework in the assigned paper and we have developed some novel ideas to improve the existing architecture. Firstly, the goal of saliency prediction research is to model the human visual attention mechanism and understand the logic of the computer vision. That is to say, we want to understand the magic behind the amazing visual pattern recognizing ability of the artificial intelligence system. For example, in human face recognition, computer has achieved better accuracy performance than the humans. Thus, to get a better understanding of the saliency prediction, we propose a novel method to visualize the intermediate layers' results and understand what happens inside the amazing convolutional neural networks, which has been such a success in computer vision. The specific methods we use to visualize the intermediate layers is to output the weights of the intermediate layer, normalize the value, perform a set of mathematical transformation and then tile the filters into an image. The detailed implementation will be introduced in the implementation section.

Secondly, as argued above, the current framework proposed by the paper only considers the Euclidean loss on every pixel and for saliency prediction. To produce a smooth and natural saliency map, we need to design a global loss based on the whole picture. To address this issue, we plan to utilize the adversarial training method to construct a discriminator to build a global loss on the whole output saliency map. Then, since the transformation in the Convnets is deterministic, we can directly perform parameters updates from the output of the discriminator to the saliency generator model.

Thirdly, as argued that given an input mage, the output saliency map should be stochastic rather than deterministic, we propose a framework which the input to the saliency generative model is sampled from a predefined noise distribution, which models the random factors in the real world. Another advantage of using random noise as an input is that now that we have random factors at the input stage, we can then use deterministic transformations, which enables the directly gradient updates from the discriminator in GANs framework.

Another ambitious idea is inspired by variational autoencoder (VAE). In VAE, the output is sampled from a distribution and in the learning stage, we aim to learn the parameters of the output distribution. For example, to get a saliency value of a pixel, we sample from a Gaussian distribution, where the two parameters' mean and variance are the parameters we need to learn. Due to the way the datasets collected, there are a lot of noise included in the ground truth. Instead of directly optimizing the Euclidean distance, we believe optimizing the likelihood of the ground truth in the output distribution is a more decent and smooth way.

Last but not the least, since the performance of the shallow Convnet is not satisfying, we decide to choose the deep Convnet as our saliency predicting model. However, although the low-level layers are transferred from Alexnet, training such a deep architecture still needs a large amount of data points. Thus we decide to train our model on a wide range of databases including SALICON, iSUN, MIT300 ad MSRA.

3. Methodology

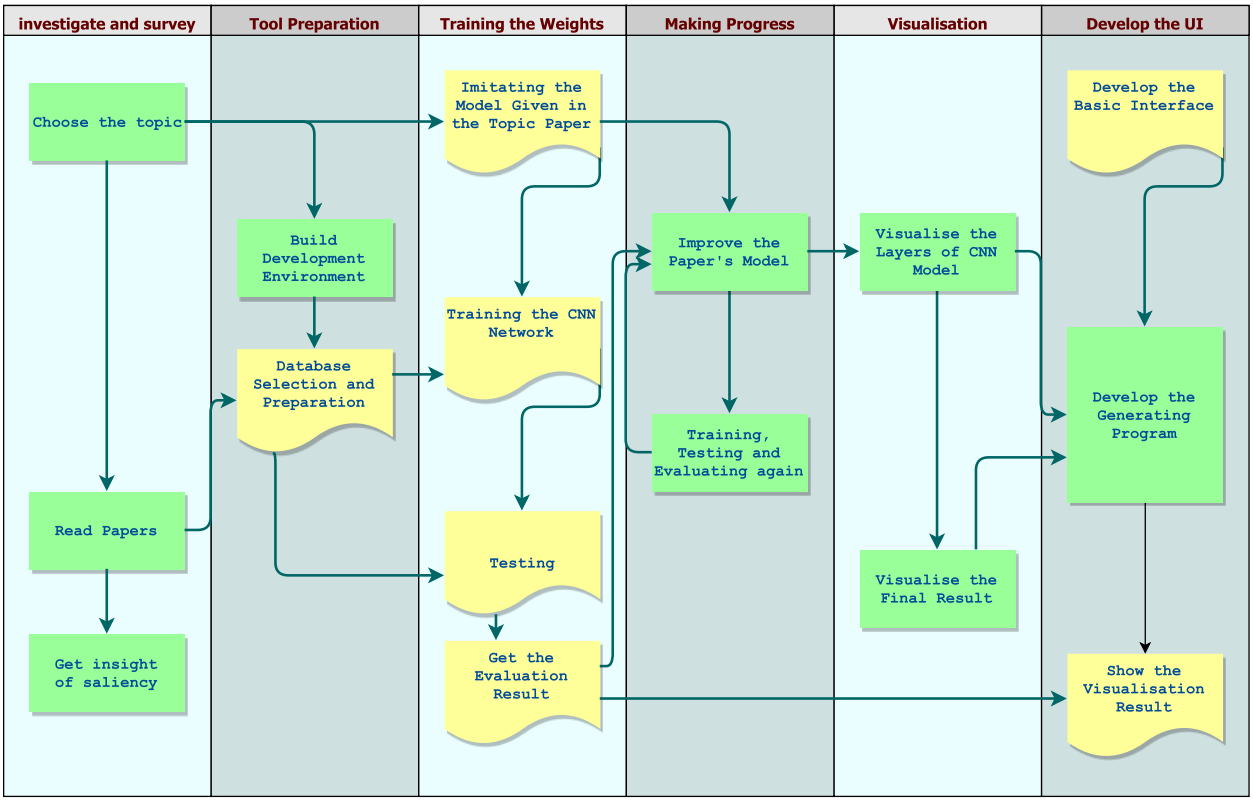


Figure 3. The flow chart of the different steps of the proposed work

As illustrated in Figure 3, we present a detailed flow chart to explicitly show the whole process of implementing the proposed work. In the investigate and survey stage, we began our journey from choosing the most interesting and attractive topic, which is the saliency prediction task. Then we made detailed survey of existing work on saliency prediction and summarized the advantages and disadvantages of those works. With the knowledge and background got from the survey, we read through the assigned paper: Shallow and Deep Convolutional Networks for Saliency Prediction and hold meetings to share the insight and ideas for saliency prediction.

Then we move on to the tool preparation stage. In this stage we mainly decided which deep learning framework to use and which database is suitable for our task. We investigated the basic information and characteristics of currently prevalent deep learning frameworks including Tensorflow, Keras, Caffe and MXNet. Since the saliency prediction task mainly involves the convolutional neural network, we chose the widely used Caffe to be our developing environment. After building the develoing environment, we downloaded

the universally used saliency prediction datasets such as SALICON, iSUN, MIT300, MIT1003 etc.

In the third stage, we first implemented the deep Convnets model proposed by the paper and trained it in a variety of datasets. After training the deep Convnets model, we conducted a lot of tests including human perceptual quality evaluation and some metrics such as Euclidean distance similarity, CC and AUC based methods.

In the fourth stage, after getting the performance of the proposed deep convnets model, we came to the improving stage, where we analyzed the limitations of current model, as argued in the Limitations of Proposed Work section and design and implement some improving methods. After implementing the improved model, we conducted evaluation again to test the efficiency of the improved model.

In the fifth stage, in order to have a better understanding of the mysterious convolutional neural network, we not only got the final saliency map output, but also managed to visualize every intermediate layer output, the implementation details will be introduced in the next section.

4. Implementation

In this section, we will introduce the methods of getting the corresponding saliency map for an input image.

First, we need to import the python packages as follows:

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from pylab import *
import os
```

Then, to visualize the intermediate layer filters, we define a function as follows:

```
def vis_square(data):
    data = (data - data.min()) / (data.max() - data.min())

    n = int(np.ceil(np.sqrt(data.shape[0])))
    padding = (((0, n ** 2 - data.shape[0]),
               (0, 1), (0, 1))                 # add some space between filters
               + ((0, 0),) * (data.ndim - 3))  # don't pad the last dimension (if there is one)
    data = np.pad(data, padding, mode='constant', constant_values=1)  # pad with ones (white)
    # tile the filters into an image
    data = data.reshape((n, n) + data.shape[1:]).transpose((0, 2, 1, 3) + tuple(range(4, data.ndim + 1)))
    data = data.reshape((n * data.shape[1], n * data.shape[3]) + data.shape[4:])
    plt.imshow(data); plt.axis('off')
```

After setting up running environment and defining the visualization function, we need to load the trained model as follows:

```
net = caffe.Net(model_def, model_weights, caffe.TEST)
```

There are three parameters here. “ model_def ” is the model building protocol, which includes all the architecture information needed to build a deep Convnets.

“ model_weights” is the file path where we save the trained model and “ caffe.TEST” disables the dropout regularization, which is utilized to avoid overfitting in the learning stage.

Before feeding an image to the net, we need to perform a set of preprocess to the image as follows:

```
mu = np.load(caffe_root + 'python/caffe/imagenet/ilsvrc_2012_mean.npy')
mu = mu.mean(1).mean(1)  # average over pixels to obtain the mean (BGR) pixel values
print 'mean-subtracted values:', zip('BGR', mu)

transformer = caffe.io.Transformer({'data': net.blobs['data1'].data.shape})
transformer.set_transpose('data', (2,0,1))  # move image channels to outermost dimension
transformer.set_mean('data', mu)            # subtract the dataset-mean value in each channel
transformer.set_raw_scale('data', 255)      # rescale from [0, 1] to [0, 255]
transformer.set_channel_swap('data', (2,1,0))  # swap channels from RGB to BGR
```

There are many preprocessing methods that can be used and here we choose the caffe.io.Transformer to do the job.

With the defined visualization function, we can see an intermediate layer activation like this:

```
filters = net.params['conv1'][0].data
vis_square(filters.transpose(0, 2, 3, 1))
```

Finally, to get the saliency map, we start from preprocessing an input image and then perform a forward propagation in the convolutional neural network as:

```
picture = net.blobs['deconv1'].data[0].transpose(1, 2, 0)
x,y,z = picture.shape
save_pic = np.zeros((x,y))
for i in range(x):
    for j in range(y):
        save_pic[x-i-1][y-j-1] = picture[i][j][0]
plt.imshow(save_pic)
plt.imsave('path',save_pic)
```

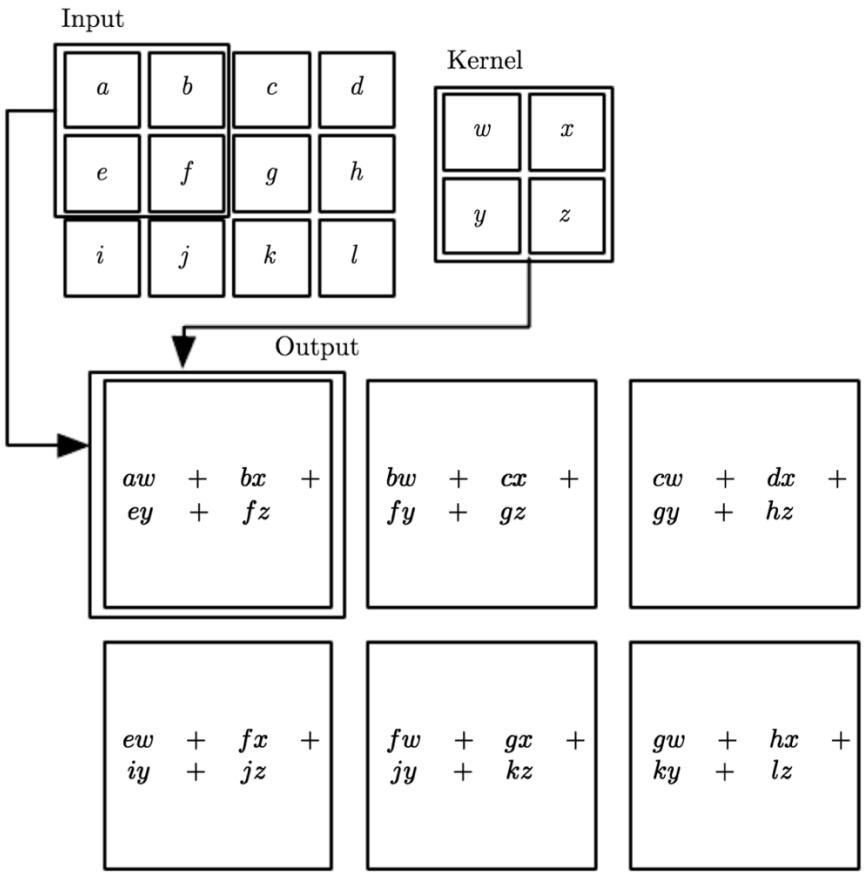
5. Results

5.1. Understanding CNNs by visualization

Before moving on to the results part, there is a need to formally define the convolution operation and its principle. If we use a two-dimensional image ***I*** as the input and also use a two-dimensional filter ***K*** correspondingly, then a convolution operation is defined as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n).$$

To make it easier to understand, we have an example here:



Since we have known what is an convolutional operation, a convolutional layer is just using different filters to extract different aspects of features and to increase the capacity of the model, we normally use a non-linear activation function after convolution, such as ReLU, tanh or Sigmoid.

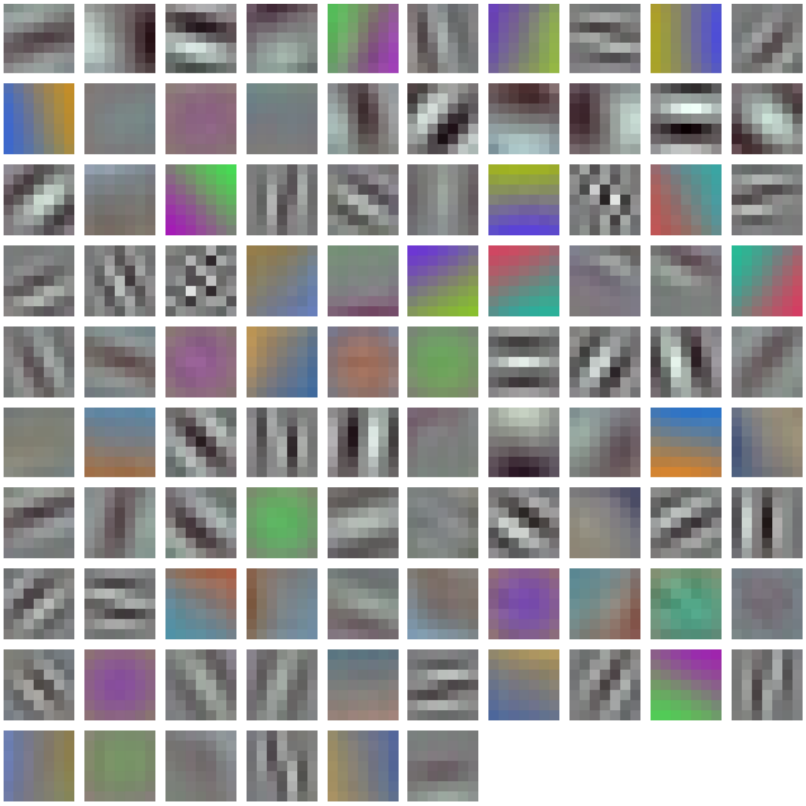
Now, let's have a look at the architecture of the implemented model:

In [12]:
```
# for each layer, show the output shape
for layer_name, blob in net.blobs.iteritems():
    print layer_name + '\t' + str(blob.data.shape)
```

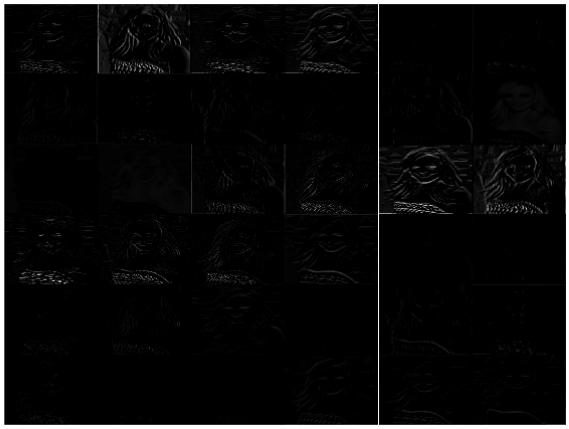
```
data1   (10, 3, 240, 320)
conv1   (10, 96, 240, 320)
norm1   (10, 96, 240, 320)
pool1   (10, 96, 120, 160)
conv2   (10, 256, 120, 160)
pool2   (10, 256, 60, 80)
conv3   (10, 512, 60, 80)
conv4   (10, 512, 60, 80)
conv5   (10, 512, 60, 80)
conv6   (10, 256, 60, 80)
conv7   (10, 128, 60, 80)
conv8   (10, 32, 60, 80)
conv9   (10, 1, 60, 80)
deconv1 (10, 1, 240, 320)
```

Then to have a better understanding of the proposed convolutional neural network, we visualize every layer of the deep Convnets.

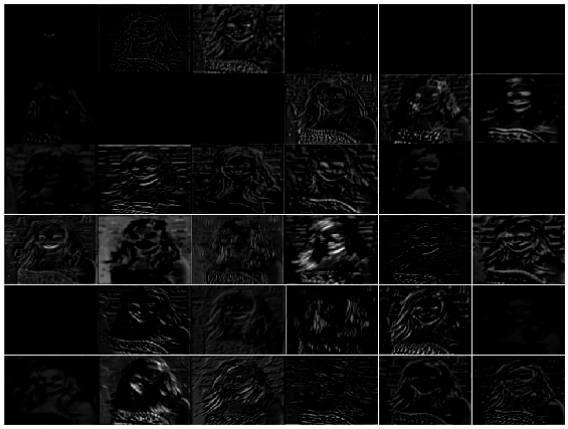
For the first convolutional layer, as discussed above, we use many different filters sliding on an image to extract different features. With different initialization, each filter will try to find some kind of features respectively during the learning process. Here are some visualization results of the filters in the first convolutional layer:



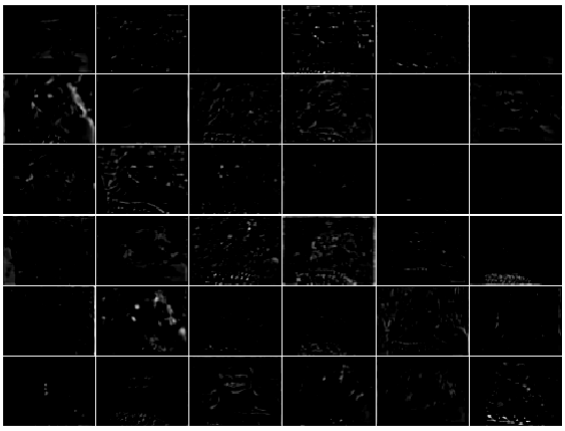
Then let's have a look at the activation results of each convolutional layer:



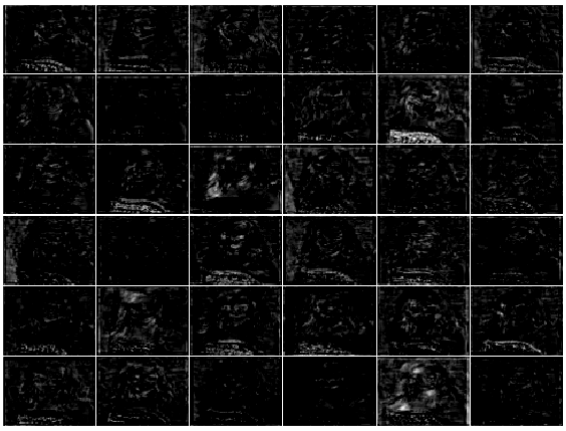
Convolutional layer 1



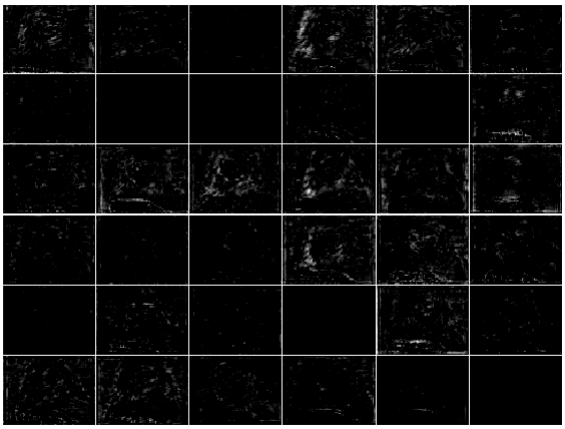
Convolutional layer 2



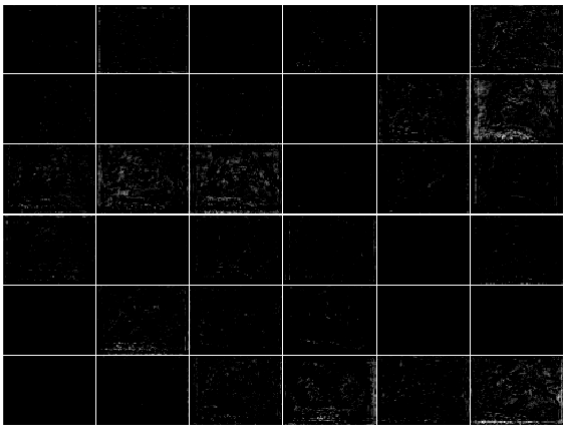
Convolutional layer 3



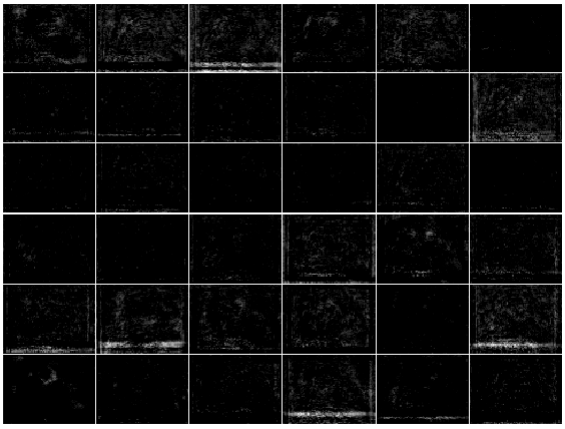
Convolutional layer 4



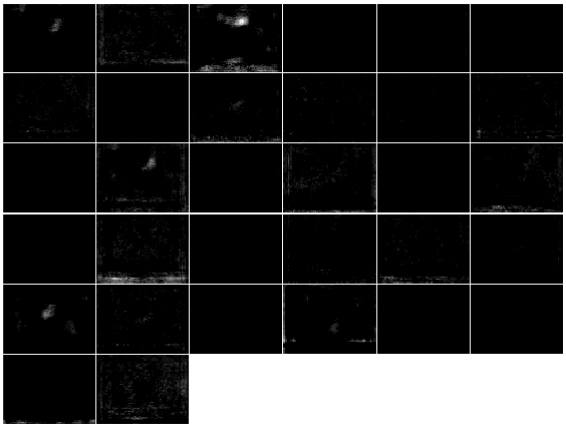
Convolutional layer 5



Convolutional layer 6



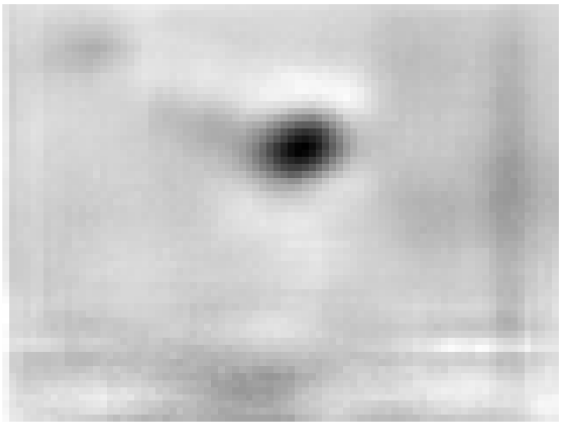
Convolutional layer 7



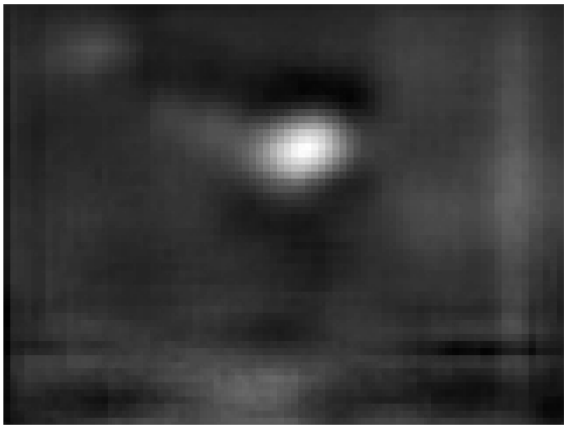
Convolutional layer 8

Those pictures show the latent representation of the input image, which can not be interpreted easily by human. But if you feed in different images, which each has its own characteristic, the activation record is different.

The ninth convolutional layer is very close to the final output saliency map. Now let's have a look at the activation record of the ninth layer and the output of the deconvolutional layer:



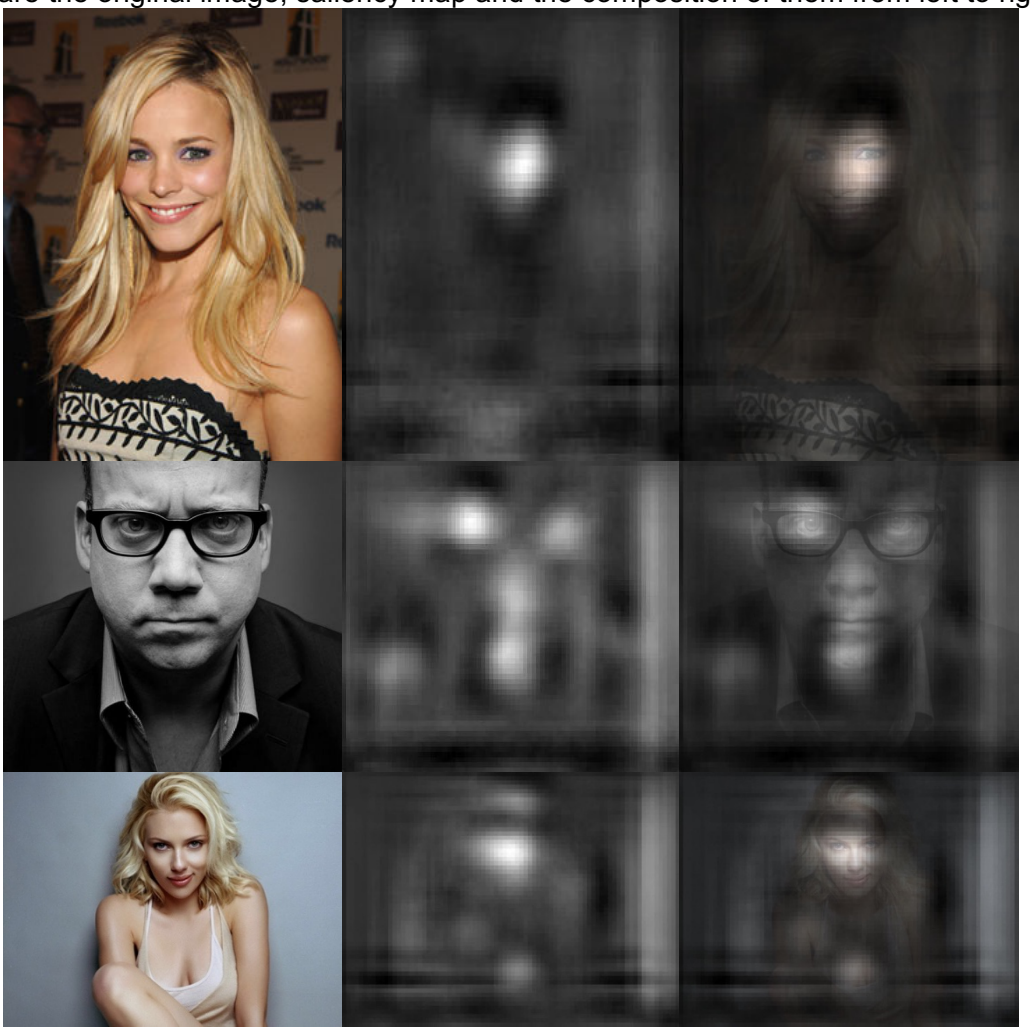
Convolutional layer 9



Output deconvolutional layer

5.2. Saliency map prediction results

Here, we will show some specific case of the saliency map prediction task. In each row, the pictures are the original image, saliency map and the composition of them from left to right.



Besides, we also implement our model to different datasets. And tables below present our accuracy performance, in which our model precedes the original model in several evaluating indicators, while because our special loss function, our model get comparatively weak performance in AUC proposed by Borji and Judd.

	SIMILARITY	CC	AUC SHUFFLED	AUC BORJI	AUC JUDD
OUR MODEL	**0.6902**	**0.8332**	**0.6867**	**0.7834**	**0.8356**
SHALLOW CONVNET(ISUN)	**0.6833**	**0.8230**	**0.6650**	**0.8463**	**0.8693**
WHU IIP	0.5593	0.6263	0.6307	0.7960	0.8197
LCYLAB	0.5474	0.5699	0.6283	0.7582	0.7846
BASELINE: BMS	0.5026	0.3465	0.5885	0.6560	0.6914
BASELINE: GBVS	0.4798	0.5087	0.6208	0.7913	0.8115
BASELINE: ITTI	0.4251	0.3728	0.6024	0.7262	0.7489

Table1: Results for the iSUN test set, according to the LSUN Challenge 2015.

	SIMILARITY	CC	AUC SHUFFLED	AUC BORJI	AUC JUDD
OUR MODEL	**0.5203**	**0.6340**	**0.6823**	**0.7923**	**0.7915**
SHALLOW CONVNET	**0.5198**	**0.5957**	**0.6698**	**0.8291**	**0.8364**
WHU IIP	0.4908	0.4569	0.6064	0.7759	0.7923
RARE 2012 IMPROVED	0.5017	0.5108	0.6644	0.8047	0.8148
BASELINE: BMS	0.4542	0.4268	0.6935	0.7699	0.7899
BASELINE: GBVS	0.4460	0.4212	0.6303	0.7816	0.7899
BASELINE: ITTI	0.3777	0.2046	0.6101	0.6603	0.6669

Table2: Results for the SALICON test set, according to the LSUN Challenge 2015.

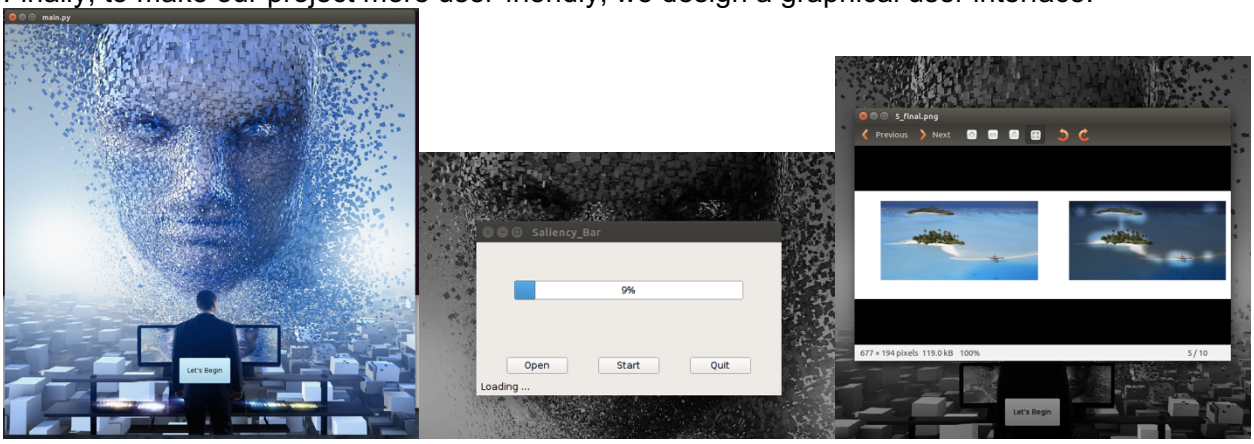
	SIMILARITY	CC	AUC SHUFFLED	AUC BORJI	AUC JUDD
BASELINE: INFINITE HUMANS	1.00	1.00	0.80	0.87	0.91
DEEP GRAZE	0.39	0.48	0.69	0.82	0.83
OUR MODEL	**0.54**	**0.59**	**0.74**	**0.78**	**0.78**
DEEP CONVNET	**0.52**	**0.58**	**0.69**	**0.82**	**0.83**

SHALLOW CONVNET	**0.46**	**0.53**	**0.64**	**0.78**	**0.80**
RARE 2012 IMPROVED	0.46	0.42	0.67	0.75	0.79
BASELINE: ONE HUMAN	0.38-0.46	0.52-0.65	0.63-0.67	0.66-0.71	0.80-0.83

Table3: Results of the MIT300 dataset.

5.3. Graphical User Interface

Finally, to make our project more user-friendly, we design a graphical user interface:



With these wrappers, the complicated details of the deep convolutional neural network are transparent to the users.

6. Conclusion & Future directions

In this work, we first implement the proposed saliency prediction framework proposed by the assigned paper. After developing a sound test platform, we use some really novel techniques to improve the existing model and solid experiments have shown the efficacy of the improved model. The limitations of current model mainly lie in the efficiency. Such deep architecture always need days of GPU training and even after we get the trained weights, predicting a saliency map of an image still requires 2 to 3 minutes. So in future, our research will focus on more simple yet effective model structure for the saliency prediction.

7. Conclusion & Future directions

[1] Bylinskii Z, Judd T, Durand F, et al. Mit saliency benchmark[J]. 2015.
[2] Everingham M, Eslami S M A, Van Gool L, et al. The pascal visual object classes challenge: A retrospective[J]. International Journal of Computer Vision, 2015, 111(1): 98-136.
[3] Harel J, Koch C, Perona P. Graph-based visual saliency[C]//Advances in neural information processing systems. 2006: 545-552.
[4] Huang X, Shen C, Boix X, et al. Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks[C]//Proceedings of the IEEE International Conference on Computer Vision. 2015: 262-270.
[5] Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding[C]//Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014: 675-678.
[6] Jiang M, Huang S, Duan J, et al. SALICON: Saliency in context[C]//2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2015: 1072-1080.
[7] Judd T, Ehinger K, Durand F, et al. Learning to predict where humans look[C]//2009 IEEE 12th International Conference on Computer Vision. IEEE, 2009: 2106-2113.
[8] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.
[9] Kümmerer M, Theis L, Bethge M. Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet[J]. arXiv preprint arXiv:1411.1045, 2014.
[10] Li G, Yu Y. Visual saliency based on multiscale deep features[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 5455-5463.
[11] Liu N, Han J, Zhang D, et al. Predicting eye fixations using convolutional neural networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 362-370.
[12] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in Neural Information Processing Systems. 2014: 2672-2680.
[13] Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps[J]. arXiv preprint arXiv:1312.6034, 2013.
[14] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
[15] Wang L, Lu H, Ruan X, et al. Deep networks for saliency detection via local estimation and global search[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 3183-3192.
[16] Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[C]//European Conference on Computer Vision. Springer International Publishing, 2014: 818-833.
[17] Zhao R, Ouyang W, Li H, et al. Saliency detection by multi-context deep learning[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 1265-1274.