Predicting saliency by learning inpainting error

by

Arezoo Abdollahi

A thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Computer Science
University of Manitoba
Winnipeg, MB, Canada

Copyright © 2019 by Arezoo Abdollahi

Thesis advisor

Dr. Neil Bruce

Author

Arezoo Abdollahi

Predicting saliency by learning inpainting error

Abstract

Understanding which parts of an image are most salient is an ongoing research problem in the field of computer vision. In this thesis, we consider the problem of saliency prediction insofar as this can benefit from learning image inpainting error. We address this problem by presenting a novel approach to predict saliency maps. The first model learns which parts of the image are most difficult to predict by applying an inpainting algorithm on a regular grid and measuring the error in the resulting prediction. A convolutional neural network is trained to predict the degree of error subject to this inpainting process. A second network uses transfer learning from the first network to predict saliency maps by taking advantage of image inpainting error. We demonstrate that saliency prediction can benefit from first learning image inpainting error. We evaluate and compare our results both by considering image inpainting error, and also through an ablation study we consider a comparison to direct training on the saliency data without transfer learning. We then evaluate our results with the previous state of the art models. We test our networks on two well-known saliency datasets including CAT2000, and SALICON.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank you, my thesis advisor, Dr. Neil Bruce, who has the substance of genius: he guided and encouraged me during my master studies. Without his help, the goal of this project would not have been realized.

I wish to express my sincere appreciation to my committee member Dr. Yang Wang and Dr. Shahin Kamali for generously offering their time, support, and their useful feedback.

I would like to extend my express to my lab mates and friends for all their emotional support.

A special thanks to my family for their unconditional love and support during this time. Words cannot express how grateful I am to my family for all of the sacrifices that youve made on my behalf. They kept me going and this work would not have been possible without their input.

Chapter 1

Introduction

1.1 General Introduction

When looking at an image, humans tend to look at the important parts of the image based on the key attentional mechanism of the human visual system. This mechanism is called saliency detection, which helps humans and animals to survive and learn faster by focusing their attention on the most important perceptual and cognitive parts of the data they receive through sensation. There are different definitions for saliency that are related to each other but in general we can say that saliency refers to the parts of the scene that stand out from the rest of the image that might include unique structure and patterns, different colors or shapes, discontinuities in structure, the contrasting parts of the scene, or objects in front of the background.

Detecting dominant patterns and textures in an image has been demanded in image tampering detection [1], copy-move forgery detection [2], image inpainting [3], or saliency detection [4]. Definition of dominant patterns or textures is application

dependent. For instance, a dominant texture in detecting the tampered region of an image refers to an anonymous altered texture in the image while in inpainting task this texture is a heterogeneous one (i.e., damaged) which must be repaired and reconstructed in a way that a skilled art conservator does. However, this definition in saliency detection [5] is more adaptable with the human vision system where the algorithm detects those regions on which human usually focus to narrow down to the important parts he sees.

Saliency prediction has many applications; for instance, one can present testable predictions that can be utilized for understanding human attention mechanisms at behavioral and neural levels. Indeed, a large number of cognitive studies have utilized saliency models for model-based hypothesis testing. Second, predicting where people look in images and videos is useful in a wide variety of applications across several domains (e.g., computer vision, robotics, neuroscience, medicine, assistive systems, healthcare, human-computer interaction, and defense). Saliency detection can be used in any area in which understanding what stands out in an image is going to be automated. If a UX designer decides to understand which parts of a design are useful or an advertiser wants to catch the eyes with a single glance in posters, saliency detection can provide the feedback loop. Image saliency detection deals with different problems, including generating a saliency map that simplifies the image representation in a way that is easier to analyze [6]. Image features and statistics (e.g., striking patterns, color contrast, discontinuities in structure), motion, and objectness are usually used in constructing saliency maps[7].

In the last few decades, several improvements have happened in the area of saliency

prediction. New data-sets, including Cat2000 and SALICON have helped to build models that address saliency problems better. With these new datasets, many new models were proposed that help to better predict the corresponding saliency map. Additionally, many metrics have been proposed for the task of saliency in ranking predictions of models that attempt to characterize where a human tends to look in an image.

In addition, With the development of the Convolutional Neural Networks(CNN) and large datasets, many models have been proposed for the task of visual saliency prediction. CNN's allows us to extract better visual features in an image in an end-to-end design. Many models have been developed based on shallow networks such as VGG-16 or Resnet-50, or even just having 3 to 5 layers. Many models for saliency prediction have been proposed based on learned features from the task of image classification. Most of the saliency prediction networks [8; 9; 10; 11; 12; 13; 14; 15; 16; 17; 18; 19] are based on a network that is pre-trained on the ImageNet dataset [20], and it seems that saliency prediction can benefit from the previously learned features. Most of the CNN models that are trained to predict saliency gain the advantage of prior knowledge about objects. However, a network that is pre-trained on image classification only considers the 1000 categories defined in ImageNet, while they are missing other categories, which may correspond to important parts of the image. For instance, there are some images in the Cat2000 data-set [21] including art, jumbled, inverted, and noisy categories that are not in the ImageNet dataset [20] or it is not common categories. Therefore, it leads saliency models which are trained on ImageNet [20] to predict lower score for these categories. It is possible that saliency

can benefit from other feature domains or a combination of different domains adapted to the problem of saliency prediction. The choice of data is very important for training image saliency models and in determining how the saliency prediction model performs at the end.

Following this idea, we contribute towards saliency map generation by proposing a collaborative learning approach in which a consensus of two different domains could provide supplementary information for the target saliency detector. We introduce image inpainting in the context of the saliency prediction problem. The image inpainting task refers to repairing parts of the image that are masked, or altered in a way that must be repaired and reconstructed in a fashion similar to what a skilled art conservator does. Image inpainting uses the semantic information with neighboring information in the image to fill in missing regions while it fails to fill in regions when there is an object that stands out from the rest of the image, or the color or shape of it is different from the background. Therefore, *the difference between the original image and the inpainted image can reveal which parts of the image are difficult to predict*. Our contributions can be summarized as follows:

- We create a dataset from the CAT2000 images [21] that shows which parts of the image are hard to inpaint if we have a mask with varying size obscuring different parts of each image. We select all twenty categories of the CAT2000 [21] dataset including 100 images from 20 categories with the total number of 2000 images. Then, we applied the image inpainting network proposed by Liu et al [22] on each image on the data-set considering different mask sizes. Each image in the dataset is inpainted by different sizes of masks along a regular grid to capture

irregularity in the image subject to different spatial scales acting as context. We calculate the difference between the original image with the inpainted image using the SSIM score [23], and we generate a heatmap as ground-truth for each scale based on the aggregation of all SSIM differences for each image.

- We use two networks to predict saliency. The first network learns to predict the degree of error for each pixel location assuming it was masked and subsequently inpainted. We trained our model with three different networks including Resnet-101 [24], Deeplab-V2 [25], and Nas-Net [26]. For this step, we use Cat2000 images as the input of our model and we use 4-scale image inpainting error heatmap generated based on Cat2000 [21] as the ground-truth. Then, the network predicts image inpainting error at 4 different spatial scales. We evaluate the results to see how the network can predict saliency after inpainting.

- A second network will learn to predict saliency. We use transfer learning to use the weights from the image inpainting network and fine-tune the network for the task of predicting saliency. The fine-tuning network also learns the weights for the newly added layers. In fine-tuning step, we fine-tune the model on Cat2000 [21], SALICON [4], and MIT dataset [27] as well to learn to predict saliency.

- We evaluate our results on Cat2000 [21], MIT [27], and SALICON [4] datasets. We compare the results for transfer learning from the inpainting task with networks trained directly on saliency ground truth data and without using image inpainting error.

1.2 Thesis Outline

The rest of this thesis is organized as follows. Section 2 lays out the background for the thesis and reviews relevant literature. Section 3 describes methodology. Dataset and evaluation methods are addressed in Section 4. Experimental results are discussed in section 5. Discussion and future work are presented in section 6. Conclusions of the thesis are presented in Section 6.1.

Chapter 2

Background and Related Works

Studying the human visual system has demonstrated that humans tend to look at areas or objects that stand out from the rest of the image. The ability of the human visual system to recognize eye-catching parts of the scene has been studied more than 80 years among neuroscientists [28]. This ability inspired computer vision scientists to investigate the human visual system more and model human gaze behaviour.

Biological studies in human and animal visual systems have inspired prior works in saliency predictions. Analyzing what grabs human attention when observing a scene, how information is processed to detect the salient parts of their environment, how animals detect predators, prey, find a mate, and camouflage help scientists to model which parts of the image stand out from the surroundings. Prior works in saliency investigate the effect of low-level, mid-level, and high-level features such as similarity, contrast, different shapes, color, and texture, or the combination of them; also, some of the approaches consider faces, animals, humans as salient targets. The advent of CNNs alongside larger scale saliency datasets has helped to capture a more

meaningful representation of images and in developing new methods for predicting image saliency. In what follows, we will investigate both traditional approaches and deep learning approaches to predict saliency.

2.1 Saliency Models

2.1.1 Classic Saliency Models

Earlier works in image saliency detect local low-level features, including colors, textures, edges, contrast, text, or different categories such as faces, animals, and some semantic information. One of the prior works inspired by the human visual system is proposed by Treisman et al [29] in 1980. Treisman et al.'s work is inspired by the ability of the brain to process different types of information, including shape, color, curves, location, texture, and objects in different parts of the brain efficiently. Treisman et al. [29] integrated the initial features at various levels to build up feature maps. Generated feature maps help them to recognize objects better.

Koch et al. [30] also simulated the focus of attention mechanism in humans by employing a neural network to capture the representation of low-level features.

Itti et al. [31], inspired by the human visual system and the role of the feature maps in recognizing objects, proposed the first bottom-up computational model. Their model extracts 42 bottom-up prior features maps made up of color, intensity, and orientation. These features are combined using a neural network to generate a two-dimensional saliency map. However, their model could not perform well on difficult scenes and structures since the model was trained on primitive features and didn't

account for spatial bias in observers toward the centre of the image.

Harel et al. [32] proposed a model based on a theoretical point of view. They proposed a computational model based on a graph-based formulation. Zhang et al. [33] also proposed a model in a theoretical framework based on Bayesian assumptions. Valenti et al. [34] considered the role of the color and edges in saliency prediction. Erdem et al. [35] explored the effect of each individual feature on saliency prediction and proposed the combination of non-linear features using region covariance matrices. Another work that builds on low-level features is a work by Judd et al. [36] and Borji et al. [37]. They first extracted low level and top-down features from images. Then, they learned a classifier in a network to determine which parts of the image are salient or not. Another approach for image saliency prediction, which takes low-level features into account, lies on a contextually adaptive representation by considering contrast and rarity for saliency [38]. In this work, the hierarchical adaptive whitening of color and feature spaces is generated. Adaptation happens by decorrelation and normalizing contrast. Saliency then is calculated by the square of the vector norm, and it measures how far the vector features reside from the center of the data. Apart from contrast and rarity, Kootestra et al. [39] used symmetry as a measure for saliency prediction. Kootstra et al. [39] were inspired by the human visual system that tends to be attracted to symmetry in a scene. Their model extracts symmetry instead of contrast and rarity in an image. They proved that the performance of their model based on the symmetry is higher than the previous models based on the contrast [31]. Liang and Hu [40] extracted middle-level features instead of low-level features, and they combined this with object detection features to predict saliency.

While prior works in saliency were based on the low-level and middle-level features, some methods considered the whole image to predict saliency, and they made their model based on global context and features.

Moreover, some of the methods combined low and high level features together, such as [36]. Zhang and Sclaroff used a simple heuristic [41]. Hou and Zhang [42] used another method to predict the saliency maps by transforming the whole image into the frequency domain. They used a frequency domain to extract spectral residuals, and then convert the result to the spatial domain to get the saliency map.

Some works predict saliency based on an Information theoretic perspective. Oliva et al [43] proposed a model to detect saliency using statistical rarity of the local features across the scenes. Bruce et al [44] also presented work to model the human attention mechanism based on maximizing sampled information from a scene. In this theory, they also get the advantage of the human visual processing system by taking the self-information of local image content into account. Bruce and Tsotsos [45] proposed a saliency attention model based on the Information Maximization theory. Their intuition implied that a region is salient if it significantly differs from its surroundings.

Zhang et al. [41] presented a saliency prediction method inspired by the human visual system that captures the global perceptual cues. The model is grounded in a boolean map(BMS), which addresses the problem of previous works that they captured edges as local low-level features which imply salient regions [46][45]. BMS is a topographical representation of a saliency map that calculates randomly thresholded boolean maps. BMS is not based on center-surround filtering and statistical analysis

of features in contrast to other models.

2.1.2 Saliency prediction based on deep learning models

The advent of the Convolutional Neural Networks (CNNs) had a significant impact in improving performance for many different tasks by providing more a powerful feature representation. Many CNN architectures have been proposed in the area of object detection, and image classification [24; 47; 48; 26] that show impressive results compared to the previous methods. In addition, deep learning models have proven to generalize to other tasks [49]. Therefore, a model that is trained for a specific task such as object detection can be extended to learn another related task such as scene recognition or image classification. A model can use the previous model to learn another similar task, or it can use the same model to learn the same task on a different dataset by starting with the weights derived from the original training pass. Therefore, the use of CNNs has proven to be a powerful tool for saliency prediction as well. Deep learning methods, along with large scale datasets for gaze prediction, can benefit from learned visual features derived from training on the ImageNet dataset [50]. Many approaches are proposed for saliency prediction, which are based on learned features from other tasks, including [8; 9; 10; 11; 12; 13; 14; 15; 16; 17; 18; 19]. In what follows, different methods are described and discussed.

eDN: The first model for saliency prediction using CNNs was proposed by Vig et al [8]. Training a model to predict saliency helped to consider different features without any assumption about which of them attract attention most. Their network has three layers, which generates lots of parameters. Inspired by biological studies, their

model searched for lots of optimal features using automated hyper-parameter optimization, and the resulting model benefits from this richly-parameterized search in predicting saliency. They combine features with a linear classifier to predict saliency. Their model was the first model that selectively chose learned features instead of combining different hand-crafted features, but their model failed to capture objects as salient regions because their network was shallow.

DeepGaze 1: Inspired by the usage of the pre-trained model on different tasks, DeepGaze 1[16], and DeepGaze 2 [13] used a pre-trained classification model to predict saliency. Kummerer proposed a method named Deep Gaze1[16] for saliency prediction, which benefits from trained features of Image-Net [51]. The baseline of DeepGaze 1 [16] is Alex-Net [20], having five convolutional layers that comprise a deeper network compared to the eDN [8] network, which has 1 to 3 layers. They designed a linear network which joins the output from each CNN layer to learn and predict saliency. Some of the output layers of this network are used as input to the readout network. They converted the output layer to a probability distribution using a softmax activation function. However, the output of the readout network is blurred and center-biased.

DeepGaze 2 and ICF: In another work, Kummerer et al [13] proposed a saliency prediction network based on convolutional neural networks (CNNs) with pre-trained weights on object recognition. While most of the proposed Convolutional Neural Network methods are based on the features trained on object recognition with high-level contrast, Kummerer considered the relevance of low-level and high-level features. They trained their network with a VGG-19 baseline [48]. DeepGaze 2 uses transfer

learning from VGG-19 [48] with some modifications to have a spatial resolution suitable for predicting the saliency map. Their model is based on high-level features such as objects, faces, and text. The output of this network is used as the input for their readout network, which has four layers of 1*1 convolutions followed by the ReLu nonlinearities. The readout network cannot learn new spatial features; however, it learns the interactions between existing features across channels instead of pixels. They convolve the output of the last layer of the readout network with a Gaussian to regularize the prediction. They also added prior distribution to model the center bias fixation maps. In the end, they converted the probability distribution over the image by using a Softmax. They also presented ICF, and they compared their results with the ICF model as well. In ICF, they used low-level features such as intensity and contrast instead of using pre-trained features from object recognition. They subtract each level of the Gaussian pyramid from the input channel to compute five levels of residuals. Then, to compute contrast in pixels, they squared the residuals, and then they blurred the output with a Gaussian kernel. However, DeepGaze 2 achieved better results compared to the ICF model, but the contribution of ICF for predicting low-level feature contrast and the effectiveness of it remains significant.

Mr-CNN: Liu et al. [52] proposed the multiresolution convolutional neural network named Mr-CNN based on the capacity of the human visual systems to search for bottom-up and top-down visual features. The architecture of their model has three different CNNs with two fully connected layers at the end. Their model learns top-down and bottom-up visual features from the multi-resolution convolutional neural network. The input of their network is image pixels, and the ground-truth is eye fix-

ation attributes that determine the fixated and non-fixated image pixels. Top-down visual features can be learned in higher layers, and bottom-up visual features can be learned by combining features from multiple resolutions. At the end, top-down and bottom-up visual features will be integrated. Their methods demonstrated that humans tend to look at both low-level and high-level visual features to extract salient regions of a natural scene.

SALICON: Huang et al. [11] proposed a method named SALICON to solve the problem of previous papers that are unable to capture the semantic content in an image as a salient region. Their proposed network is a combination of two pre-trained CNNs with coarse and fine inputs that are based on the AlexNet [20], VGG-16 [48], and GoogLeNet [53] networks. At the end, the two CNNs are concatenated to generate the final saliency map. They mention that they got better results by using VGG compared to the AlexNet [20] or GoogleNet [53].

DeepFix: Kruthiventi [19] presented an end-to-end approach based on a very deep VGGNet [48] named Deepfix, which models saliency prediction. DeepFix consists of five convolution blocks that are followed by the two new layers called Location Biased Convolutions, which are designed to capture the semantic information of the saliency prediction. In Deepfix, we have convolutional kernels with holes to enable convolutional layers to have a multiple scales and large receptive fields without increasing the memory footprint. Having a receptive field with a bigger size enabled them to model the center-bias in predicting human eye fixations. Utilizing large and multi-scale receptive fields enabled DeepFix to exploit the interrelated context of the image, and it improved the results significantly. Adding Gaussian priors improved the

results by helping to learn weights better. DeepFix improved the performance over DeepGaze I in the MIT Saliency Benchmark [27]. The results of DeepFix also suggest that the VGG features are more powerful to extract features suitable for predicting saliency compared to AlexNet [20]. However, one downside of the Deepfix model is that using a Convolutional Neural Network with a large receptive field cannot help to predict saliency in all parts of the image and the model tends to perform well in the area near to the center of the image.

ML-Net: Cornia et al. [9] proposed an architecture that combines features extracted from different levels of a convolutional neural network instead of just keeping the features from the last layer. Their network also learns Gaussian parameters in an end-to-end manner, which is different from the previous methods where they had a fixed set of Gaussians, or they feed Gaussian parameters into the network. Learning Gaussian parameters during training can help to model the center-prior. Their end-to-end network encodes features extracted from multiple levels, and then they will have a prior learning network at the end instead of a pre-trained network. Inspired by the problems in previous saliency detection works, they proposed a new loss function. Their loss function measures the similarity with the ground truth, solves the problem of variance in the saliency map, and highlights the pixels with high ground-truth fixation probability.

Shallow and Deep Network: Junting Pan et al. [54] proposed the first data-driven end-to-end saliency prediction approach. They trained two separate networks, one of which is a shallow network that they trained from scratch. The other one is a deeper network that uses three layers from the AlexNet [20], which is trained on

image classification. The network learns to reduce the gap between the predicted saliency map and the ground-truth by measuring the Euclidean distance as a loss function.

PDP: Jetley et al. [12] consider the saliency problem as a Bernoulli distribution. Then, they designed a deep end-to-end network to learn this probability distribution with a new loss function. Instead of using loss functions that are designed for regression or classification tasks, they employed a softmax activation function with a measure designed for calculating distances between probability distributions. Their evaluation results demonstrate that their proposed loss function works better than Euclidean distance and Huber loss for the saliency prediction problem.

DSCLRCN: Liu et al [18] proposed a multi-resolution convolutional neural network for saliency prediction that outperforms DeepFix [19] which used larger receptive fields in convolutional networks to combine contextual texture and information. Both DSCLRCN and SAM-Resnet [10] use ResNet and long-short term memory (LSTM) to achieve high accuracy in predicting saliency. SAM-Resnet uses LSTM to generate an attention map to refine the predicted saliency map iteratively. In DSCLRCN, they first extract local features by the model pre-trained for scene recognition to predict saliency in parallel, and then they imitate the human cortical lateral inhibition mechanism by using LSTM which sweeps horizontally and vertically through the image to infer global context and accounts for this towards a more accurate saliency prediction. DSCLRCN can predict accurate detection compare to the DeepFix, which demonstrates the ability of their proposed model in incorporating global and scene context compared to the Deepfix [19] model. Their model also predicts less false posi-

tives compare to Deepfix [19], ML-Net [9], Deep Convnet [54] and eDN [8]. DSCLRCN also can predict challenging scenarios for cluttered scenes and places when there are no obvious salient regions.

SALGAN: Junting Pan et al. [14] were the first to take advantage of adversarial networks in predicting saliency. Their architectures incorporate two networks, one to predict saliency using a VGG-16 based encoder-decoder, and the other a discriminator network, which compares the output with the ground truth and decides how close is the prediction to the ground-truth based on the loss function. While previous methods tend to use a loss function based on the single saliency metrics, they considered that the evaluation results would be lower on other saliency metrics. They choose BCE-based content adversarial loss function, and they proved the effectiveness of it both for initializing the saliency prediction networks and for regularizing the adversarial network using the simple cross-entropy loss function. Taking adversarial BCE loss function improved the results of saliency metrics compared to using cross-entropy in their loss function. They also proved the effect of adversarial training on generated saliency maps compared to training with cross-entropy loss. Training with only BCE causes some visual artifacts and does not seem to be smooth in the output produced; adversarial training helped them to have a more integrated and smoother image.

iSEEL: In iSEEL [15], they proposed a method based on the similarities among two images. The intuition for their model is that because people might have similar fixation patterns when looking at two similar images, the fixation prediction for an image can benefit the saliency prediction of another similar image. They investigated their idea of how well the saliency map of an image can benefit from predicting the

fixation map of another similar image. They chose some similar and dissimilar pairs from a dataset. They found that the evaluation score of the similar pairs is high, which demonstrates two fixation maps are similar, and for the different images, the evaluation results show that the fixation map of two images is not similar together. Their network for learning human eye fixations was based on the Extreme Learning Mechanism (ELM) [55]. They defined similar image sets for each image and the network train to learn saliency for each image was based on similar images.

EML-Net: In this paper, Jia et al. [56] addressed the problem of using shallow network in considering the large computational space needed for deep CNN models. Their proposed model is Expandable Multi-Layer NETwork (EML-NET), which is a scalable system for saliency prediction. Training splits to smaller modules for more scalability. For instance, encoder and decoder in a Fully Connected Neural Network can be trained separately instead of training together. Aside from training encoder and decoder separately, each CNN model in the encoder step is trained separately as well. This scalability allows EML-Net to have different CNN models in the encoder stage that can be combined to use prior knowledge from different but related tasks such as object detection [50] and scene [57] recognition for the task of saliency prediction. They also can combine this structure with recurrent and multi-resolution models. The decoding stage requires training from scratch for new added encoding modules; however, it has less computational costs. They also presented a new loss function in their network by modifying Pearson's Correlation Coefficient (CC) and Normalized Scanpath Saliency (NSS) to represent dissimilarity, and they combined it with Kullback-Leibler Divergence (KLD) to compute an overall loss function.

DVA: Wang et al. [17] proposed deep supervised learning to predict saliency based on skip-layer networks with various receptive field sizes and an encoder-decoder network. Their model captures both local and global features hierarchically. Unlike previous methods that have supervision for their last layer, they use deep supervision for all the layers. The encoder consists of multiple convolutional layers, and three decoders are designed to aggregate the encoder's inputs in three different stages of the encoding process. Their proposed network extracts saliency features at different layers, and in the end, they aggregate the output of these three decoders to generate a saliency map.

SAM-Net: Cornia et al. [10] incorporate an LSTM-based attentive convolutional neural network into their network. Like the DSCLRCN network [18], they also employed Resnet [24] in their network. Cornia et al. [10] combined LSTM with both Resnet-50. [48] and VGG-16. [24] networks to iteratively make better saliency prediction by refining the saliency map. Their method outperforms the previous Resnet-based model named DSCLrscn with an improvement of 1.5% in NSS metric, 1.3% in CC, and 0.4 in sAUC metric. In their method, they designed an LSTM network for spatial features instead of using a sequence of time. They obtained this by replacing convolutional operations in the LSTM equations with dot products. The input to the LSTM network is the features extracted from the Dilated Convolutional Network that can preserve scale better, and this reduces the unfortunate effect of rescaling. The output of the LSTM Network is a refined stack of feature maps that are used as input to the prior learning module. In learning the prior module, their model learns center bias by learning the parameters of Gaussian functions that modulate

the output. Their network also can predict the center-bias in eye fixation prediction. At the end, a Convolutional Neural Network with 512 filters is applied on top of the prior learning module to change the size of the tensor channel while adding more non-linearity to the model. They also presented a novel loss function based on the saliency evaluation metric in which they use the linear combination of the Normalized Scanpath Saliency (NSS), the Linear Correlation Coefficient (CC), and the Kullback-Leibler Divergence (KL-Div). They proved that the combination of three evaluation metrics works better than each of them individually.

FUCOS: Bruce et al. [58] presented a fully convolutional neural network for both gaze prediction and salient object recognition. They train their model based on the pre-trained layers of CNNs, and then they fine-tune it on the PASCAL-Context dataset [59].

DenseSal: Taiki Oyama et al [60] was the first that proved the relationship between the higher accuracy in image classification, and it's effectiveness in estimating image saliency. They also employed an effective architecture that has fewer parameters with upsampling layers and based on multi-scale images to improve the resolution of the saliency map. They choose their initialization weights based on pre-trained ImageNet classification.

MSI-Net: Kroner et al [61] proposed a method based on the pre-trained convolutional neural networks on the image classification task. Their goal was to predict fixations in natural scenes, and they incorporate high-level features with contextual information to develop a robust representation. They propose an architecture consisting of an encoder-decoder with large receptive fields in different dilation factors

to capture the salient regions in complex scenes. They outperform other methods for saliency prediction, which were based on a pre-trained VGG16 network.

While most of the presented approaches incorporate object recognition information to predict salient region in the scene, none of them investigate the effect of image inpainting on predicting saliency. A brief summary of the saliency prediction model is in table 2.2. Here, we prove that image inpainting can help us to better predict the salient regions in an image, and in the rest of the related works, we discuss some of the inpainting models.

2.2 Image Inpainting

Image inpainting refers to the process of completing deteriorated or missing parts of the image. Different types of image inpainting methods, including structural inpainting and textural inpainting, are used for different objectives. In structural inpainting, geometric approaches are employed to fill in the missing region of the image. Since structural inpainting methods are incapable of solving the problem of texture restoration; textural inpainting methods can restore the texture in the missing area. A combination of structural and textural image inpainting can produce more satisfying results to fill in the missing texture of the image with the textural method and maintaining boundaries of the image with the structural method.

Classic Approaches

In some of the classical methods, a missing region is filled with the neighbors' information. Telea et al. [62] proposed a method based on the Fast Marching algorithm

to fill in the missing region. The Fast Marching algorithm starts from the boundary of the inpainted region to the interior section. During this marching, it approximates the distance map from the boundaries while it propagates the locally weighted averages of already traced pixels. Therefore, Fast Marching estimates the value of a pixel by calculating the weighted means of the previously calculated pixels. This model failed to replace the big holes, and it produces artifacts, such as over-smoothing.

Some of the previous models were based on Partial Differential Equations with filling in the missing region with isophotes line. Bertalmio et al. [63] proposed a method that was based on the Partial Differential Equations (PDE) in image processing. The basic idea of their work was propagating information in the isophotes direction from neighbors to fill in the missing region. However, their algorithm could produce natural results in some cases, but it fails to regenerate the large textured missing parts. It also takes a few minutes to generate the image, which was not suitable for the application. Ballester et al. [64] also proposed a method that is was based on PDEs. Their algorithm fills in missing regions by both the pixels gray value and their related gradient directions. This algorithm also fails to fill in the textured regions in an image.

PatchMatch [65] is one of the methods for reproducing images based on filling the patch, and it is not based on deep learning. The reproduced images are smooth, but because it does not follow any semantics in filing the patch, sometimes the results are not meaningful. Also, they did not provide their method with enough statistics to make it comparable to other methods. However, Patch-Match has a higher speed compared to the previous methods based on filling the patch.

Deep Learning based Approaches

Recently, deep learning approaches for image inpainting have evolved that generate visually more plausible results compared to previous approaches by considering semantic knowledge. Some of the proposed deep learning approaches heavily depend on the size and shape of the masks, or depend on post-processing steps, which will create blurriness, color-discrepancies, and artifact effects. To remove the artifact's effect and generate a smooth image, they employ different post-processing strategies.

Fast Marching [62] and Poisson blending [66] is used in work by Iizuka et al [67] to remove the effect of centered masks. Another network that needs refinement for the prediction is a work by [3] in which they used a network with contextual attention to enhance the sharpness of the texture. Iizuka et al [67] and Yuet al. [3] solved the problem of depending on central masks in image inpainting; however, they did not provide their test on a large scale of images.

Many models based on unsupervised and supervised learning have been proposed. One of the first unsupervised deep learning methods for image inpainting is proposed by Pathak et al [68], which is based on an Encoder-Decoder framework. Their encoders are based on AlexNet [20], and map the image to a lower-dimensional feature space. Their model is trained from scratch with randomly initialized weights, and it is trained to predict context instead of predicting image classification. Encoding features are used to generate the image in the decoding step through the channel-wise, fully connected layer. In the decoding step, they used upsampled convolutional layers with the ReLU activation function to generate the image. They also used two loss functions, one of which was based on the reconstruction loss, and the other the

combination of reconstruction loss with adversarial loss. Their model can capture semantic information of an image; however, the re-generated image contains visual artifacts often, and they seem blurry because of the information bottleneck in the channel-wise fully connected layer.

Iizuka et al. [67] addressed some of the problems of the Context-Encoder-Decoder network [68] by having less down-sampling layers and consequently less up-sampling layers. They have two discriminator networks one of which searches globally for the coherent pattern, and the other one searches locally for the detailed information. Then, the image completion network tries to fool both discriminators. In their completion network, they used dilated convolution instead of a fully connected channel-wise layer to make better predictions compared to this work [65]. However, they used less down-sampling layers, but they had heavy dilated convolutional layers, which increased the training time significantly.

Yang et al. [69] also proposed another solution to improve both the results of context encoder-decoder by using a VGG-network [48] to address the problem of inpainting high-resolution images. They designed a joint optimization convolutional neural network to generate missing regions of an image by considering global context and local textures. They also proposed a patch synthesis approach to inpaint high-resolution image inpainting. Their approach increases the computational costs in inference time, which is a negative of their work.

Liu et al. [22] proposed a deep learning method that can fill in irregular holes in the image. Instead of using U-Net CNNs [70] with Skip layers like previous works, they proposed to use a Partial Convolutional Neural Network instead of typical convolu-

tions to address the problem of previous networks. They added a mask updates step in their network to get better results. The results of their method are semantically meaningful while having very good results both with irregular and regular masks.

Nazeri et al. [71] proposed a method for image inpainting inspired by how artists first draw lines and then complete the image. They proposed a network consists of two steps where, in the first steps, they force the network to generate edges in the missing region and then an image completion network completes the image with suitable texture and color in the second step. In the end, an end-to-end network combines the generated edges with the image completion network.

While all of these different strategies tried to reconstruct destroyed parts of the image, it often leads to some artifacts. If we consider the difference between the original image and the inpainted image, this difference is higher in some particular areas. For instance, the error is higher when there is an object, or when there is a different pattern, color, or a different object that stands out from the rest of the image. We take advantage of this property of inpainting error correlating with potentially interesting content in learning saliency.

Table 2.1: Summary of deep learning based models for saliency prediction.

Model	**Description and Data-set**
eDN[8]	The first model selectively chooses features instead of combining different features. It uses a 3 layer network with a linear classifier to predict saliency. This model failed to capture objects as salient regions because their network was shallow. **DataSet:** MIT1003, Toronto, NUSEF, MIT300.
DeepGaze 1[16]	Baseline: AlexNet with pre-trained weight on ImageNet. The output is blurred and center-biased Trained on ImageNet features **DataSet:** MIT1003, MIT300
DeepGaze 2 [13]	They presented one model based on VGG-19 with pre-trained weights from object recognition to capture high-level features. They also presented ICF to capture low-level features. **DataSet:** MIT1003
Mr-CNN [52]	Proposed a multiresolution CNNs named Mr-CNN consisting of three different CNNs with two fully connected layers. It learns top-down visual features in higher layers and bottom-up visual features by combining features from multiple resolutions. **DataSet:** MIT, Toronto, NUSEF

Continued on next page

Table 2.1 – continued from previous page

Model	Description and Data-set
SALICON [11]	Their method is based on the combination of AlexNet, VGGNet and GoogLeNet which is pre-trained both on coarse and fine input to capture the semantic content better **DataSet:** MIT300, Toronto, OSIE, MIT1003, NUSEF, FIFA, PASCAL-S
DeepFix [19]	The model is based on VGGNet with a large sized receptive field. Their model mostly predicts center-biased saliency information. **DataSet:** SALICON, CAT2000, MIT1003, MIT300
ML-Net [9]	This model Combines features from different layers instead of just using the last layer while it learns Gaussian parameters during training. It also uses prior learning instead of using the pre-trained network. **DataSet:** SALICON, MIT300
Deep and Shallow Network [54]	They trained two networks separately. One of them is a shallow network trained from scratch and the other one uses AlexNet pre-trained on image classification **DataSet:** SALICON, MIT1003, MIT300, iSUN
PDP [12]	They proposed a network to learn a Bernoulli distribution with a new loss function. Their proposed loss function works better than Euclidean distance and Huber loss. **DataSet:** SALICON, MIT1003, MIT300, OSIE, VOCA-2012

Continued on next page

Table 2.1 – continued from previous page

Model	Description and Data-set
DSCLRCN [18]	Proposed a multi-resolution CNN based on Resnet with a larger receptive field to combine contextual texture and other information. **DataSet:** SALICON, MIT300, MIT1003
SALGAN [14]	They proposed the first adversarial network with a BCE adversarial loss function for learning saliency. **DataSet:** SALICON, MIT300
iSEEL [15]	They investigate how the saliency map of one image can help to predict the saliency map for a similar image. Their network is based on the Extreme Learning Mechanism(EML) which learns saliency for an image from a saliency map of another similar image. **DataSet:** MIT, MIT300, CAT2000, OSIE
EML-Net [56]	They trained encoder and decoder separately and also each CNN model in the encoder is trained separately to address the problem of computational space. They also presented a new loss function based on the saliency evaluation metrics. **DataSet:** SALICON, MIT1003, MIT300, CAT2000
DVA [17]	Presented a supervised model to capture global and local features with a skip-layer network and various receptive fields that supervise all layers. **DataSet:** MIT300, MIT1003, TORONTO, PASLCAL-S, DUT-OMRON

Continued on next page

Table 2.1 – continued from previous page

Model	Description and Data-set
SAM-Net [10]	Combined LSTM with Resnet and VGGNet. They replaced the convolution operation with dot product in LSTM. **DataSet:** SALICON, MIT1003, MIT300, Cat2000
Focus [58]	They present a fully connected neural network for both gaze prediction and object recognition
DenseSal [60]	They investigate the effect of higher accuracy in image classification and saliency prediction. **DataSet:** SALICON, OSIE, PASCAL-S, MIT1003, MIT300
MSI-Net [61]	They proposed a model based on learned features of image classification to capture salient regions in complex scenes by incorporating high-level features and contextual information. **DataSet:** SALICON, Cat2000, MIT1003

Chapter 3

Methodology

In this section, we describe how we selected our dataset and the network for creating our unique ground-truth based on image inpainting error. Further, we will discuss the architecture of the first network, which learns image inpainting error and the second network that uses transfer learning to predict saliency.

3.1 Generating Ground Truth

The output of the network is heavily dependent on the data we choose. Therefore, it is essential to understand the ground-truth of the network and how we generate these data. The steps for generating ground truth are depicted in Figure 3.1.

3.1.1 Image Inpainting with partial convolution

We used the image inpainting network that is proposed by Liu et al. [22] to inpaint our images. This paper is the first approach that uses a neural network with irregular

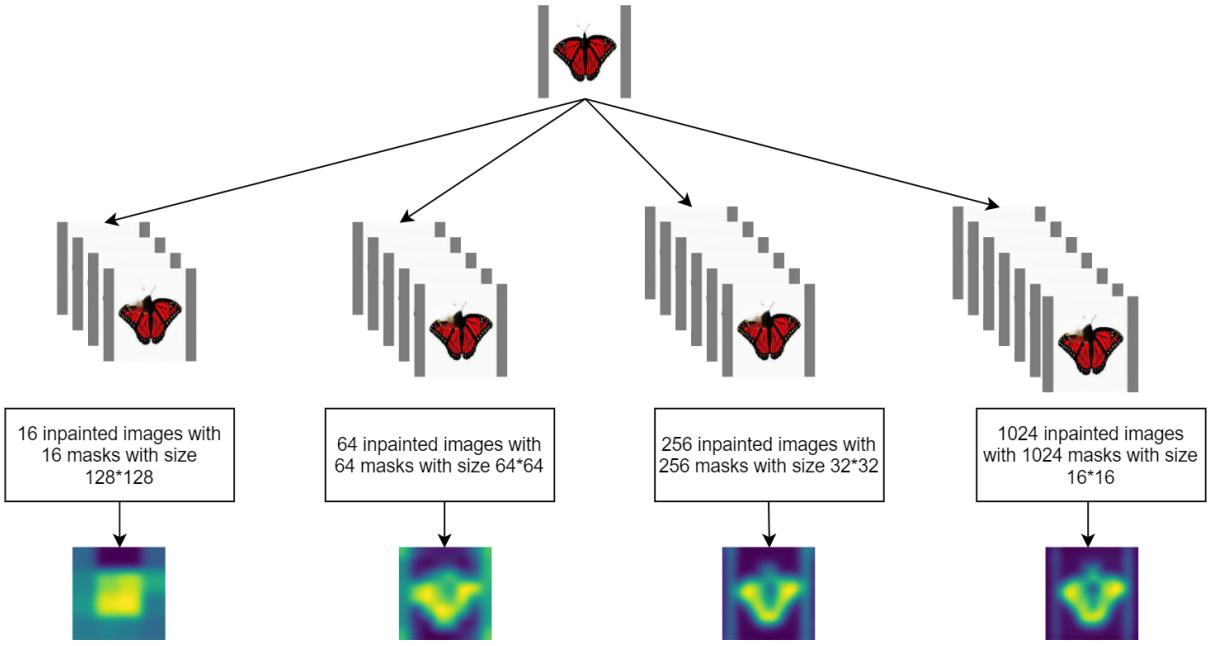

Figure 3.1: Generating 4 scale ground-truth from differences of image inpainting and the original image for a deep learning model

masks to inpaint images and it has the best performance both qualitatively and quantitatively compared to the previous state-of-the-art models [67][3] [68] [69] [72]. This model can semantically fill in the missed regions while it does not depend on the centered rectangular mask to avoid over-fitting. They employed Partial Convolutional Neural Networks instead of typical convolutional layers to overcome the previous shortcomings in the inpainting methods. Also, they added a mask update step to remove the effect of the mask by doing partial convolutional layers on the unmasked area.

In a partial convolutional neural network, each convolution window is calculated based on the element-wise multiplication of feature values of the current window and

the binary mask. Then, the result will be divided by some of the masks to adjust the amount of unmasked inputs. The equation 3.1 will demonstrate this better if we consider W as the weight for convolution filter, M as the mask, and X as the feature pixels in the current window. In the partial convolution layer, each time the mask will be updated based on the equation 3.2. If the convolution was able to change the input value in at least one input and the sum of the mask is more than zero, then the new mask would be 1; If not, it would be zero.

$$x^{'} = \begin{cases} W^T(X \odot M)\frac{1}{sum(M)} + b, & if some(M) > 0 \\ 0, & otherwise \end{cases} \tag{3.1}$$

$$m^{'} = \begin{cases} 1, & if sum(M) > 0 \\ 0, & otherwise \end{cases} \tag{3.2}$$

The partial convolutional neural network is developed based on this U-Net network [73] by replacing typical convolutional layers with partial convolutional layers. There are eight partial convolutional layers in the encoding step, and ReLU is used between all the layers. The encoding step uses a stride of size 2, one kernel of size 7, two kernels of size two, and the last five layers use a kernel of size 3. The channel sizes are 64, 128, 256, 512, 512, 512, 512, and 512. The decoding step uses eight upsampling layers by nearest neighbor and each with a factor of two followed by a partial convolutional layer in the upsampling layer and with a LeakyReLu with alpha of 0.2. Batch-Normalization is also used between each layer. Both feature maps and binary masks are concatenated channel-wise before being fed to the decoder stage through a skip link. The last layer of the network will copy the non-hole pixels from

the image by having the original image with the hole and the mask as input to the last layer.

We inpaint all the images in the CAT2000 dataset [21] with partial convolutional network image inpainting models [22]. The network is trained on the places365 dataset [23], and it is fine-tuned on the MS-COCO dataset [74] with the batch-size of 4 for both training and fine-tuning, and a learning rate of 0.002 for training and 0.0002 for fine-tuning. The network is trained on the images with size $224 * 224$. Other configurations of the network are the same as the paper. We test this network on the Cat2000 dataset [21], and we resized images to a size of $512 * 512$. We had four different sized square masks, and the size of the masks is $16 * 16$, $32 * 32$, $64 * 64$, and $128 * 128$. Respectively, We inpaint each image 1024 times by masks of size $16 * 16$, 256 times by masks of size $32 * 32$, 64 times by masks of size $64 * 64$ and 16 times by masks of size $128 * 128$.

3.1.2 Measuring the difference between original image and the inpainted image

We choose the image inpainting network with the highest accuracy to inpaint images, with the different sizes of masks placed in different parts of the image to capture content at different spatial scales. The reason that we choose the best model among other models is that we want to just have a high error for the areas that there is something that stands out from the rest of the image. The difference between the original image and the inpainted image in other models such as [67] are higher in even simple areas of the image such as sky compare to other parts of the image.

In addition, the lower performance of other image inpainting network is due to the effect of mask in the re-produced image [67] [68]. Therefore, to prevent both the effect of masks in the reproduced image and the high error in the usual parts of the image, we choose to work with the model with the highest performance. Even the best inpainting network cannot refill all masked parts of the image the same as their initial pattern. While image inpainting uses semantic information to fill in the missing region, it cannot reproduce images when there is something that stands out from the rest of the image. Also, the reproduced images are most likely to be different for refilling edges of objects, and outliers in luminance, texture, or the color of an image. We calculate the difference between the original image and the inpainted image based on the structural similarity metric (SSIM)[75]. SSIM compares two images using information about luminance, contrast and structure. SSIM is calculated using following formula:

$$(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.3}$$

3.1.3 Generating Heat-maps as ground-truth

We calculate the SSIM difference [75] between each original image and the corresponding inpainted images. For instance, for an image inpainted with mask 16*16, we have 1024 inpainted images. We have 1024 scores for the difference between each inpainted image and the original image, which identifies how similar are the original image and the inpainted image. After having all these scores, we can generate a positional heat-map for them. Then we subtract the score from 1 to get the dissim-

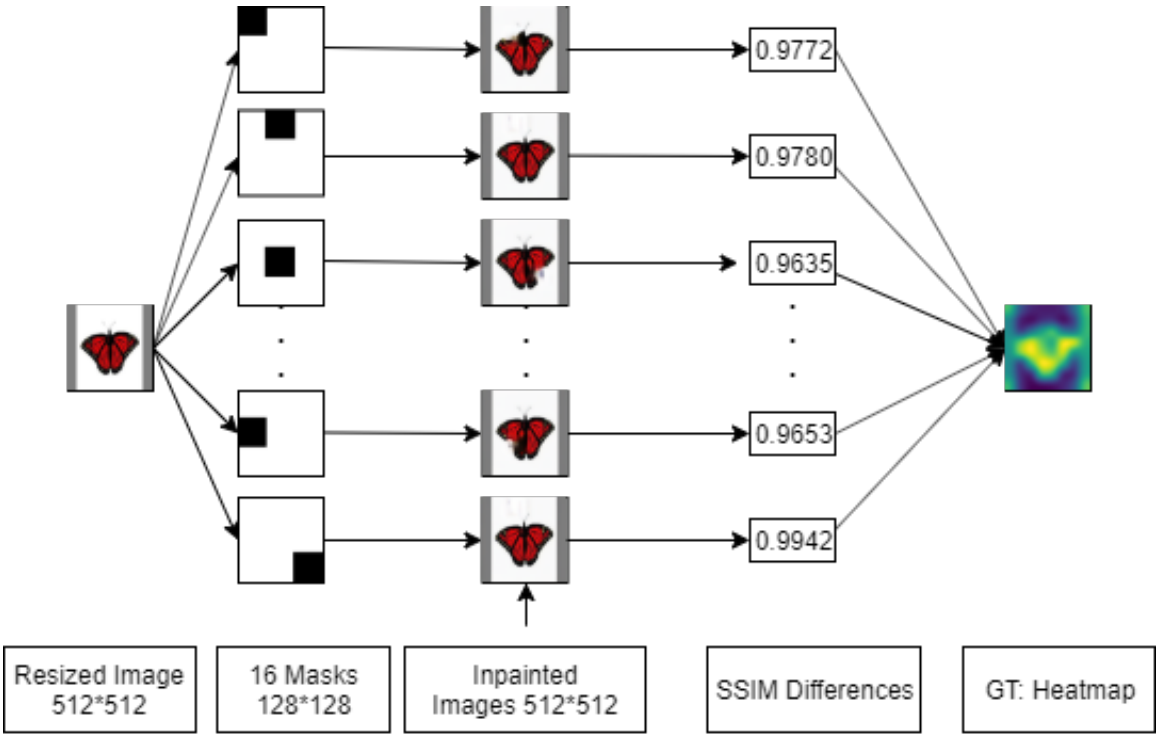

Figure 3.2: Generating a heat-map for the images inpainted with 16 masks of size 128*128 as ground-truth for the deep learning model

ilarity of the original image and the inpainted image. Therefore, we know what is the dissimilarity of each part of the image with the inpainted image. We produce the final image using a Gaussian with sigma = 20, and we normalize the image to have values between 0 and one. We experimented with generating heat-maps with different sigmas for the Gaussian. We found that a Gaussian with a standard deviation of 20 works well in avoiding grid artifacts of the masks, and the output is more coherent compared to the smaller sigma. We repeat this step for different sizes of masks. Therefore, we have four heat-maps for each image at the end corresponding to different scales of inpainted regions. In figure 3.2, generating ground-truth for the

inpainted images with 16 masks of size $128 * 128$ is depicted. We use the images from this step as the ground-truth for the CNN model that captures inpainting error.

The effect of sigma on Heat-map

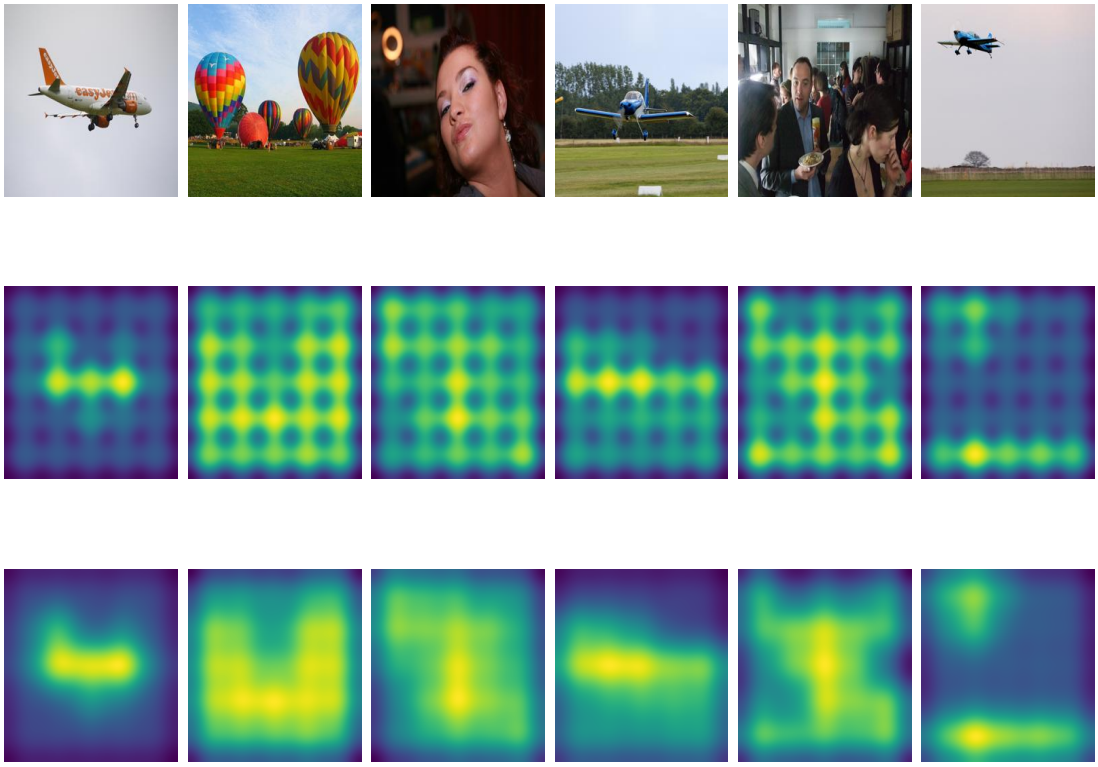

Figure 3.3: Qualitative examples of using different sigma for generating heat-map from 25 inpainted images with circular masks.

In Figure 3.3, the first row shows the original image, the second line shows the heat-map generated by 25 masks and sigma of 10, the second row shows the heat-map generated from 25 inpainted error images and with the sigma of 20. We choose to work with the sigma of 20 because it removes the effect of mask locations while it

gives a clear representation of which parts of the image are more salient compared to the rest of the image.

How is image inpainting error related to saliency?

As it is shown in Figure 3.4, the generated heat-map by using image inpainting error, and having different sizes for the mask, can produce an image that is similar to the saliency map. We observed that the heat-map could generate the border of an object in an image since the error is higher in the edge areas. Also, it includes information from other parts of the image, such as variation in color or texture that was difficult to inpaint.

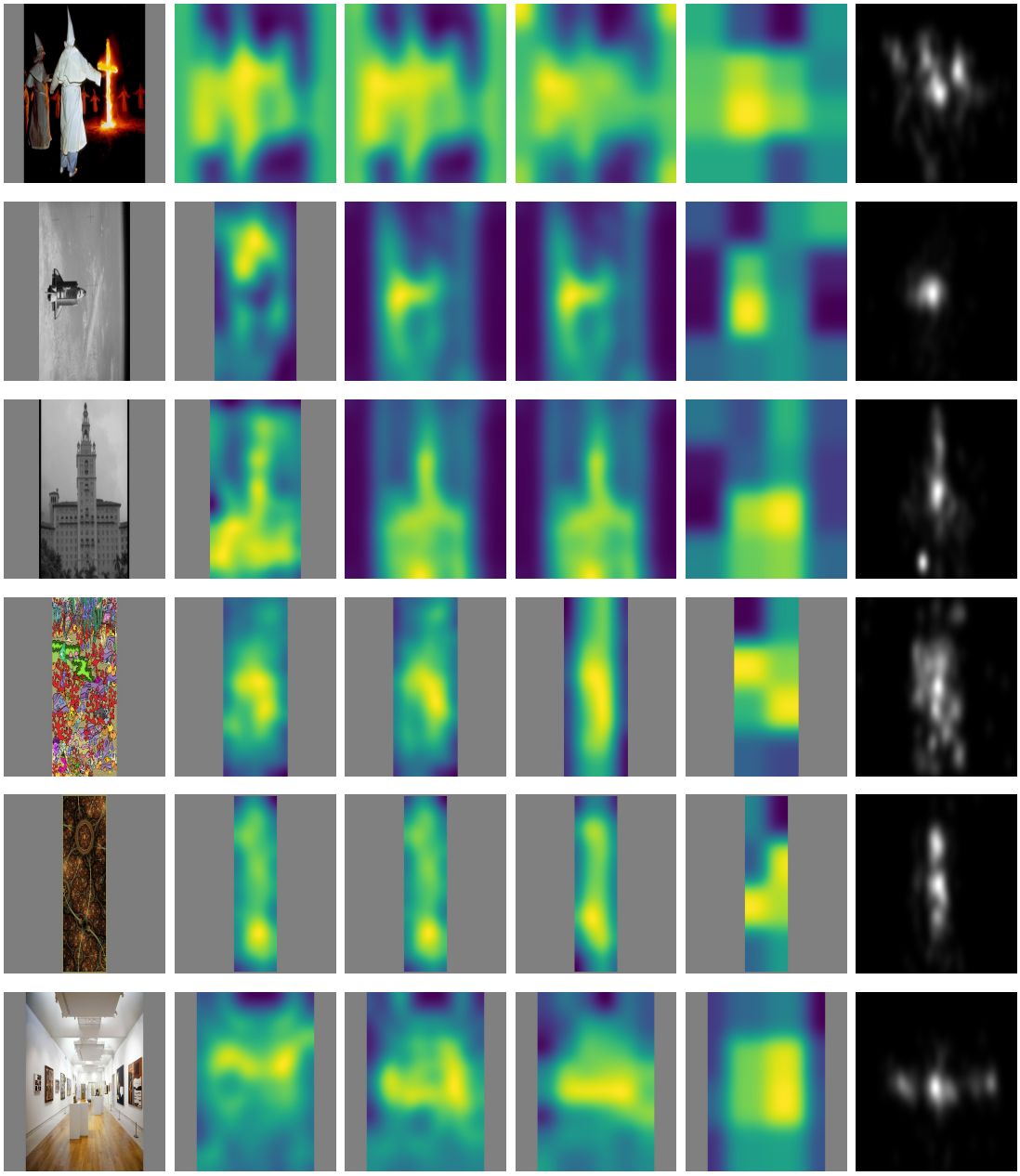

Figure 3.4: Comparison of heat-map and saliency map. The first column shows the original image, the second column is the result of heatmap with mask16, the third column is heatmap with mask32, the forth one is the hetamap with mask64, and fifth column is the heatmap with mask128, and the last one is saliency map from the Cat2000 dataset.

3.2 Proposed Architecture

We have two steps to learn and predict the saliency regions of an image. The first network will learn which parts of the image have a higher inpainting error. The second network uses transfer learning to learn which parts of the image are more salient. In the previous methods, Resnet-50 [24] and VGGNet-16 [48] are used in most of the saliency prediction models including [10; 19; 13; 18; 17]. It is also proven that Resnet-50 outperforms VGGNet-16 both in the image classification task and in the saliency prediction task [10; 18]. Here, we choose to work with deeper networks, one of them is based on Resnet-101 [24], one is based on Deeplab-V2 [25], and the last one is based on NassNet [26]. Further, we learn to predict the saliency map by transfer learning on each of these networks. In what follows, the architecture of the network with the applied change on them is also described.

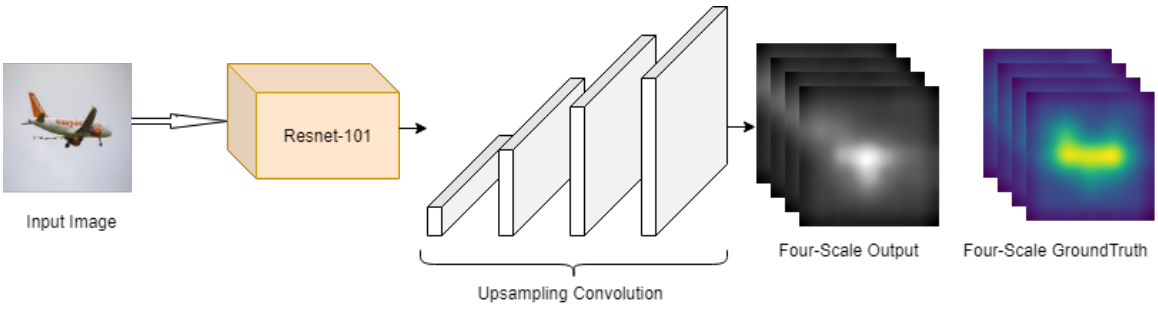

Figure 3.5: A high level overview of the proposed method

- In the first step, we trained our model with Resnet-101, Deeplab-V2, and NasNet. The first network learns the image inpainting error. We have original images as input and four heatmaps related to each image as the ground-truth.

- In the second step, we used transfer learning to predict saliency by having an original image as input and saliency map as ground-truth. The second network uses the pre-trained weight of the first network, and it learns to predict saliency.

3.2.1 Training with Resnet-101

Resnet [24] is a convolutional neural network proposed by Microsoft Research Asia trained on millions of images from the ImageNet dataset [50]. It has three variants named Resnet-50, Resnet-101, and Resnet-152. Resnet achieved outstanding results for image classification, detection, and localization in ILSVRC 2015, it was also the winner of image classification, detection, and localization in MS-COCO competition in 2015. Since image saliency tends to correspond to salient parts of the image, object locations, and segmentation we choose to use Resnet-101 [24] to predict saliency. Resnet [24] addressed the problem of vanishing or exploding gradients in previous models [76; 77] by introducing skip connections that previously were addressed by using intermediate layers as normalization. Resnet also addressed the problem of complexity from the previous models such as [48].

However, previous models introduced networks with too many layers, increasing numbers of layers leading to having more training error. Resnet used residual connections which allowed for 152 or more layers while keeping error small. The core idea behind Resnet [24] is proposing residual connections that allow having a deeper network that that can be optimized more easily. Residual connections can connect the input of the previous layer to the next layer. The skip or shortcut connection in the Residual network adds the input value to the output after some layers to avoid

vanishing gradients. Therefore, if the value of the weight layers after some layers becomes small, it will add it up with the input, and it prevents from vanishing the gradient. The weight layer then is learning the difference between the summation of input and the weight and the input itself.

Resnet is inspired by the VGG networks [48]. The resnet convolutional layers mostly have 3*3 filters, and follows two rules. If the input size is equal to the output size, the layers have an equal number of filters as the input size. If the input dimension is smaller than the output dimension, the number of filters will double to preserve the time complexity per layer. Convolutional layers with stride 2 will use down-sampling. An average pooling layer and a fully connected layer with softmax are applied on top of it. VGG [48] has fewer filters overall, and the complexity of Resnet is less than VGG nets [48].

Resnet addressed the problem of time complexity by designing a Bottleneck architecture. Inspired by the technique used in Network In Network [78] and GooLENet [53], using 1*1 convolution can reduce the number of parameters while not affecting the performance of the network in a negative way. Therefore, they modified the residual blocks as each residual block uses a stack of 3 layers instead of 2 layers. Each stack of 3 layers includes 1*1, 3*3, and 1*1 convolutions, which reduces the dimension of the input/output. Batch-Normalization is also used in each stack inspired by Inception-v1 [79]. They increased the number of layers by adding the 1*1 convolution while preserving time complexity. They changed 34-layer Resnet to 50 layers, and they also used more bottleneck design to change it to Resnet-101, and Resnet-152.

We used Resnet-101 [24] to train our model. We keep all the stacked layers of the

Resnet, and we omit the average pooling and fully connected layers. We added four upsampling layers with learned filters that upsample the image by a factor of 2 in each layer. Each of the upsampling layers consists of the convolution layer with input size of 2048, 1025, 512, and 256, and the output channel is 1024, 512, 256, and 4. The last convolutional layer has a stack of four images in training related to each scale of the heatmap. We used a kernel of size 1, a stride of size 1, no padding, and dilation of 2. Batch Normalization [79] is used in all the upsampling layers except the last layer. For the newly added layer, we initialized the weight using Kaiming normalization. We used a ReLU after the first upsampling layers, and we used sigmoid for the last layer.

3.2.2 Training with Deeplab-V2

Deeplab [80] is a convolutional neural network that addresses the task of semantic image segmentation to label each pixel as a class of an object. Semantic segmentation can learn and predict the boundaries of objects which has an application in autonomous vehicles. Deeplab-V1 [80] is built on top of the pre-trained VGG-16 [48], and it modifies it by adding fully connected conditional random field (CRFs), and atrous convolution.

Deeplab-V2 [25] is on top of the pre-trained ResNet [24] trained on ImageNet [20] and it use Resnet as the main feature extraction step but Deeplab modified the ResNet Residual block to capture multi-scale features. Deeplab-V2 [25] improves Deeplab-V1 by adding Atrous Spatial Pyramid Pooling. DeepLab-V2 has three main contributions to the ResNet network. First, it uses the atrous convolution in the

last block of ResNet, and all the convolutions in this new block employ different dilation rates to capture multi-level feature extraction. Second, it uses the Atrous Spatial Pyramid Pooling named ASPP on top of the last convolution block. ASPP uses different rates for dilated convolution to extract multi-scale features. Third, they added a connected Conditional Random Field (CRF) at the end of the fully connected layer to improve the localization performance. Here, we describe all three steps.

Atrous Convolution for Dense Feature Extraction And Field-of-View Enlargement: One advantage of deeplab comes from so-called 'atrous convolution' instead of vanilla convolution filters. Previous deep CNNs use the convolution with a kernel size of 3*3 to keep both the number of the parameters and computational cost in check. Atrous convolution changes the size of the filter by adding zeros. If we have filters of size $k * k$, and the rate r, then the size of the kernel can be enlarged by $k + (k - 1) * (r - 1)$. Atrous convolution help to increase the receptive field of viewed filters while it keeps the number of parameters and the computational complexity the same as before. in addition, atrous convolution helps to control the resolution at places where features are used in convolutional neural networks. Therefore, atrous convolution is by design, a way to increase the receptive field quickly and efficiently.

Multiscale Image Representations using Atrous Spatial Pyramid Pooling: A second advantage involves a method to improve the scalability of the CNNs. While previous methods could capture features in different scales by having both small and large size images, they proposed an approach to handle various scales in deep learning. They employ atrous spatial pyramid pooling (ASPP) on top of the ResNet with dilated convolution and with different rates to segment objects in various

scales. This helps the network to both capture objects and image context at multiple scales.

Structured Prediction with Fully-Connected Conditional Random Fields for Accurate Boundary Recovery

At the end, probabilistic graphical models are combined with previous deep CNNs to improve the localization of object boundaries. They applied Conditional Random Fields (CRFs) at the last layer of DCNN, which helped them to improve the performance of object localization both qualitatively and quantitatively.

In our work, we used DeepLab-V2 because image saliency and image segmentation can be related to each other. We used all the layers of DeepLab-V2 except the last layer which sums up the previous layer. We added a layer at the end to have four scale outputs for the network to generate the four-scale heatmap. We also ignore using CRFs as a post-processing step on the images since we need to use the weights from training to fine-tune the model for the next step and this prediction is not the end product of the model.

3.2.3 Training with NasNet

NasNet network [26] searches for the best convolutional layer or cell on a small data-set to generalize to another data-set. Despite other methods that are designing blocks in a hand-crafted fashion, NasNet finds the best combinations of the possible set of operations, that leads to the highest performance. This is done using a controller in the form of a recurrent neural network to form a cell. In NasNet, they first found the best units and cells on the CIFAR data-set, and then they applied these cells

on the ImageNet data-set [51] by copying more of these cells. ScheduledDropPath is also used in the NasNet network to generalize the model better. NasNet achieved state-of-the-art results with smaller model size and with a lower complexity among other models. We applied the NasNet architecture to learn saliency features. We changed the last layer of the NasNet network to generate four scale output instead of one scale output.

3.2.4 Fine-tuning by transfer learning

Humans have an ability to learn a new task from skills or knowledge acquired from previously learned tasks, and we do not learn everything from scratch. Humans can use from their previous knowledge and experience to analyze new problems or tasks and learn them faster. Previous methods in machine learning are designed to solve a specific task without having the ability to generalize to other tasks. Transfer learning gives us this ability to learn a new task from a previously learned task with the same data-set, to learn the same task with a different data-set, or to learn a new task with a new data-set. The primary motivation behind transfer learning is that we need to have a vast amount of data for some tasks to be able to solve them with deep learning, and the data should be labeled. For most supervised models, we need a large amount of data to learn from, which can pose a challenge given the cost and complexity of making a data-set or labeling all the images in a data-set from scratch. Transfer learning enables us to train a model even with a small amount of data, and it also helps us to increase our capability for learning by using pre-trained networks and using features and weights derived from a previously trained-on task. Transfer

learning enable us to learn a new task from a previously similar learned task. If the two tasks are more related to each other, it is easier for the network to learn the new task.

Using pre-trained models as feature extractors:

Convolutional neural networks consist of different layers such that each layer learns different features. All these layers ultimately connect to the final layer of the network at the end. If we use this network without the last layer, we can have a pre-trained network as feature extractors. Ultimately the final layer of the network is involved in making a decision and has an associated loss. However, this final layer can be redefined with a different objective while retaining all of the value in the representation that corresponds to features learned for an earlier task. For example, features that contributed to a network for image classification can be re-purposed for semantic segmentation by changing the structure of the final (or last few) layers in the network to predict an alternative objective.

Fine-tuning the pre-trained models:

In this method, we can train some layers of the previous model selectively and update the weights for these layers again. During fine-tuning, one can freeze some layers and use the general extracted features from the training step while not training some of the previous layers again. In addition, more layers can be added during fine-tuning such that these layers have learnable parameters with pre-trained features either being adaptable or frozen.

Using transfer learning to predict saliency from the learned image inpainting error:

We use transfer learning in this thesis to predict saliency by using image inpainting error. While previous models use a pre-trained object detection network to predict saliency, we investigated the effects of first pre-training to predict image inpainting error as follows.

1. All the images in Cat2000 [21] data-set are inpainted with 4-scale masks to inpaint all parts of the image. The difference between the original image and the inpainted image is calculated by SSIM metrics [75]. We observed that places where there is an object present, or where there is something that stands out from the rest of the image are more difficult to inpaint. We found these two domains close to each other, and we used the first network to learn image inpainting error, and then we use transfer learning to learn to predict saliency.

2. We generate inpainted error images, and we used them as ground-truth in our model. We inpainted each image 1360 times with different sizes of masks, and then we generated heat-maps in 4 distinct spatial scales. We choose to use the images in the Cat2000 data-set [21] for inpainting and for training the network. Inpainting 2000 images with each inpainted 1360 times requires both time and memory, which proved prohibitive if we also wanted to also inpaint the SALICON dataset [4] images, which has 10000 training samples.

3. The goal of the first network is to learn which parts of the image are more difficult to inpaint if we were to mask any given region of the image, i.e. the network predicts which parts of the image are unique and are difficult for the inpainting network to predict what lies in the masked section of the image. The input of the first network is based on resized images from the CAT2000

data-set [21], and the ground-truth is heatmaps for four spatial scales generated from image inpainting error. Then, we use the learned features from the first network to train the second network.

4. We freeze most of the layers from the first network to prevent over-fitting because the number of CAT2000 categories is limited to 20, and the number of images was just 2000 images. Then, we added some learned layers on top of the second network, and we fine-tuned the network. The input of the second network is the same as the trained network, and we use the original Cat2000 [21] images from the data-set, while the ground-truth of this network is a saliency map from the CAT2000 data-set [21]. The second network learns which parts of the image are more salient by taking advantage of knowing which parts of the image stand out and are more difficult to inpaint.

The chart below demonstrates training and fine-tuning steps more clearly.

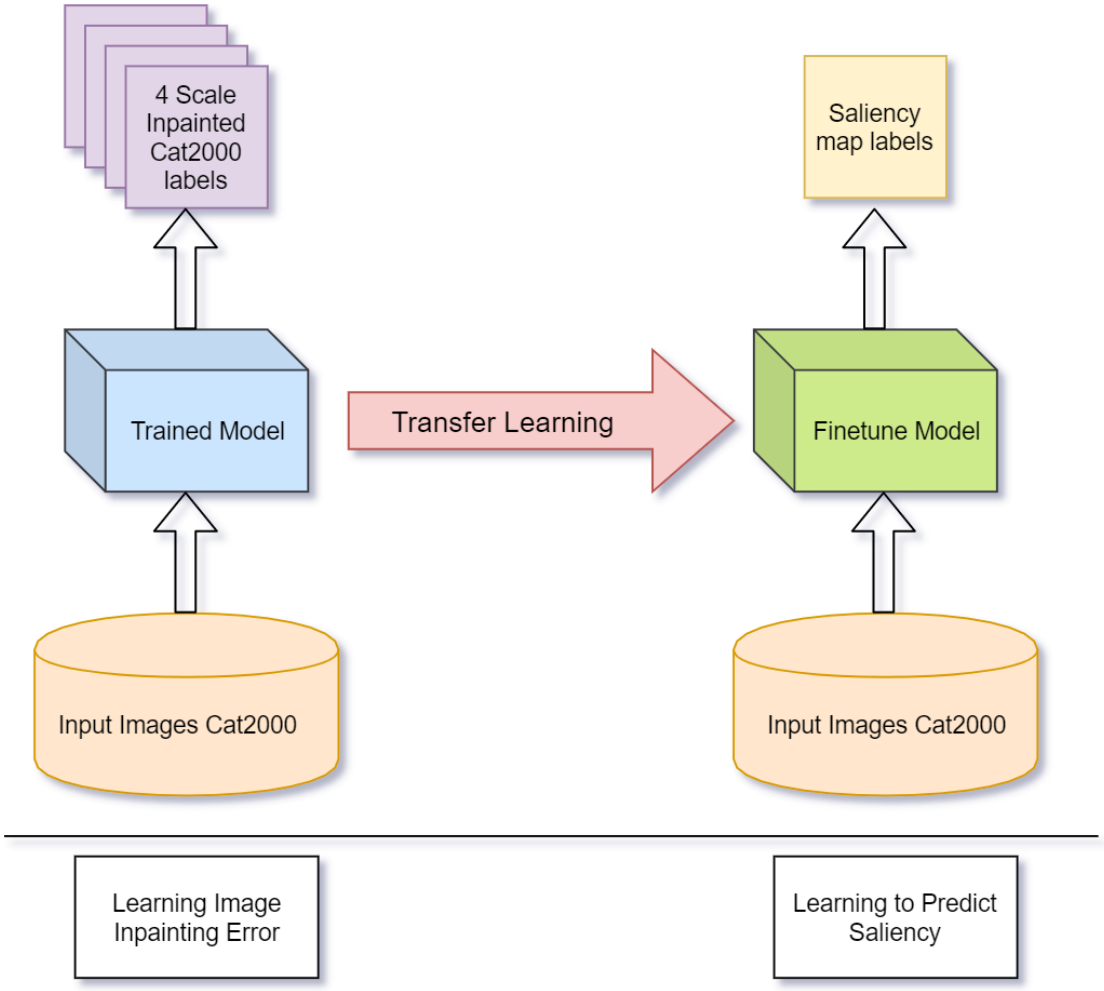

Figure 3.6: Saliency prediction by learning image inpainting error. In the first step the network learns to predict image inpainting error by having images from Cat2000 data-set as input and 4-scale heatmaps as ground-truth. In the second step, the network learns to predict saliency by fine-tuning on SALICON or Cat2000 data-set with image and label both from the saliency data-set.

Chapter 4

Evaluation Metrics and Dataset

We evaluated the performance of our network on the Cat2000 [21]and SALICON [4] datasets. We provide performance according to both the evaluation metrics on the training-validation set and the test set. Since recent studies by Bylinskii et al [81], and Kummerer [82] demonstrate that evaluating the performance of a model based on one of these metrics does not give a fair comparison, we evaluated our results on a variety of Metrics. We used different evaluation metrics, including AUC-Judd, AUC-Borji, shuffled-AUC, Normalized Scanpath Saliency (NSS), Similarity, and Linear Correlation Coefficient(CC). In what follows, we will describe the evaluation metrics we used here. We divide the evaluation metrics into two categories such that one of them is based on measuring the similarity between two images, and the other one is based on the dissimilarity of them.

4.1 Metrics based on Similarity

4.1.1 Area Under Curve (AUC) :

AUC is among the similarity-based evaluation metrics, and it is a widely used evaluation metric to measure the accuracy of the model on different datasets. AUC is refered to as the area under the Receiver Operating Characteristic (ROC) curve, which is a fundamental measuring scope to distinguish two different groups in a signal detection theoretic sense. AUC can measure how well a parameter can discriminate two different groups. Considering the saliency map prediction as a binary classification problem such that each pixel can be a fixated pixel or not, an ROC curve is plotted which captures the true-positive rate vs. false-positive curve in different level sets. AUC considers the ground-truth, which are human eye fixations as a positive set, and it chooses some other points for the negative dataset as non-fixation points. Depending on the distribution of the negative points, AUC can be divided into AUC-Judd and AUC-Borji in which both of have a uniform distribution of negative points as non-fixation points and shuffled-AUC uses human fixations of other images as the non-fixation distribution.

AUC-Judd [36] is one of the common metrics in evaluating the predicted saliency map. For each threshold given, AUC-Judd measures the number of true positives to the total fixated pixels as the TP rate, and it measures the number of total pixels in each threshold set minus the fixated pixels which gives false positives. Therefore, FP rates would be calculated by the division of false-positive by the total number of saliency map pixels that are 'on' for a specific distribution. Borji et al. [27] proposed

another variant of AUC named AUC-Borji, which selects the negative points subject to a uniform distribution.

4.1.2 Shuffled-AUC :

Shuffled-AUC [33] samples the negative fixation point locations from other images instead of picking fixation location at random. The s-AUC mostly selects negative samples from the center of the image because if we average fixations of many images, the results would tend to lie in the center of the image. Therefore, if a model predicts human eye fixations in the center of an image, it is more probable that it leads to a lower score for s-AUC.

4.1.3 Normalized Scanpath Saliency (NSS):

Normalized Scanpath Saliency is introduced by [83] and measures the relation between saliency maps and eye fixation ground-truth. It measures the average of unit normalized saliency map values at human eye fixation pixels with unit standard deviation and mean of zero.

Unlike AUC that receives a high score if we have high-valued predictions at fixated locations with many low-valued false positives, NSS is sensitive to all false positive points, and relative differences in saliency across the image. Having false positives leads to a lower normalized saliency value at each fixation location, and it reduces the overall NSS score.

NSS is calculated through the following formula where P is the saliency map, F the binary fixation points, and N indicates the number of fixation points $N = \sum_i F_i$,

and i iterates through all the fixation points. NSS is calculated by equation 4.1 by the average sum of multiplying the saliency map and fixation points at each point. The mean saliency map is subtracted from the saliency map for each pixel through equation 4.2 and this makes it invariant to linear transformation.

$$NSS(P, F) = \frac{1}{N} \sum_{i}^{N} \overline{P_i} \times F_i \tag{4.1}$$

$$\overline{P} = \frac{P - \mu(P)}{\sigma(P)} \tag{4.2}$$

4.1.4 Information Gain (IG):

Kummerer et al [82] proposed a saliency evaluation metric based on an information-theoretic method and inspired by the idea that saliency maps are regularized and optimized to be center-biased [82; 84]. IG compares the predicted fixation map and the ground-truth to determine how well the model can predict fixation locations beyond dataset biases rather than the center bias prior. If IG is above zero this means that it can predict fixation points better than the center-prior image, and if it comes below zero, it means that it mostly predicts center-biased image saliency.

4.1.5 Similarity Metric (SIM):

Similarity metric [85] is the summation of minimum values for each pixel between the two distributions such that one is the ground truth and the other one is the saliency map. Both of these distributions should be normalized and the summation of them for calculating SIM should be one. The closer the score to 1, the more similar

are these distributions and the value is 0, it means that two distribution do not have any overlap. In the following formula, G is the ground-truth and P is the prediction.

$$S_{im} = \sum_{i=1}^{N} min(P_N(i), G_N(i)) \tag{4.3}$$

One of the negative points of SIM is that the SIM score depends highly on the Gaussian sigma value. For instance, if a prediction map selects the correct location but with higher sigma or less sigma, it will affect the SIM score negatively. SIM only reaches its maximal value if the sigma value for both distribution maps is the same. SIM is more sensitive to false positives rather than false negatives.

4.1.6 Linear Correlation Coefficient(CC):

The Correlation Coefficient metric measures the linear relationship between two saliency maps. To calculate CC, both saliency maps should have unit variance and a mean of zero. The closer the score to +1, the more similar are the two saliency maps. Where both saliency map and ground-truth fixation map have the similar magnitude values, the CC will reach its highest score. CC treats false positives and false negatives equally unlike SIM. Correlation Coefficient is calculated using this equation in which G indicates ground truth saliency map and P is the predicted saliency map.

$$CC = \frac{cov(G, P)}{\sigma_G * \sigma_P} \tag{4.4}$$

4.2 Metrics based on Dissimilarity

4.2.1 Earth Movers Distance (EMD):

EMD measures the spatial distance between two 2D maps, ground-truth, and prediction map; specifically, how far is the prediction map from the ground-truth. If the two maps are similar (identical), then EMD would be zero. However, if the difference between the two maps is high, then EMD would be higher. EMD measures the cost of transforming the probability distribution of the predicted saliency map P to the ground truth map G. A low EMD score shows that there is a small difference between the predicted saliency map and the ground truth. Therefore, the lower the EMD score, the better the prediction. A higher EMD score means that the predicted map spreads all over the image. Therefore, false positives cause a higher EMD as EMD needs to move one density map to another. Even though SIM considers the alignment of two distribution in calculating SIM score, EMD prefers a sparse prediction, and it has a higher score in sparse predictions where there is a good match.

EMD is calculated based on four constraints and through the following formula in which f_{ij} indicates how much density we need to transfer from the ith distribution to the jth distribution. d is the distance between ground-truths from distribution i to distribution j:

$$EMD(P, Q^D) = min_{f_{ij}} \sum_{i,j} f_{ij} d_{ij} + \mid \sum_{i} P_i - \sum_{j} Q_j^D \mid max_{i,j} d_{i,j} \tag{4.5}$$

The first constraint is that distribution should move from the P to Q while having as a constraint $f_{ij} \geq 0$. EMD also checks the amount of density that is needed to move from P to Q by checking $\sum_j f_{ij} \leq P_i$. EMD also prevents depositing of more

density in location Q than it should be by having $\sum_i f_{ij} \leq Q_j$ as a constraint. The last constraint checks whether the amount of density moved is not more than the density in both P, and Q by checking $\sum_{i,j} f_{i,j} = min(\sum_i P_i, \sum_j Q_j)$.

4.2.2 KL Divergence

The KL-Div metric calculates the Kullback-Leibler divergence between the ground truth and predicted saliency after normalizing both maps with unit variance. In KL-Div, the lower the values, the better is the prediction as it calculates the divergence between the predicted-map and the ground-truth saliency map. KL divergence computes the dissimilarity between the predicted saliency map and the ground-truth. Therefore, if we consider the ground-truth as Q and saliency map as P, a large number is added to the loss where the ground-truth is not zero but the predicted map is zero or close to zero. KL-divergence penalizes zero values based on the KL-divergence loss. The KL divergence will calculated through the following formula:

$$KL(P, Q) = \sum_i Q_i \log(\epsilon + \frac{Q_i}{\epsilon + P_i}) \tag{4.6}$$

4.3 Datasets

CAT 2000: This dataset [21] contains two sets of training images and test images. Images are gathered from different categories, such as Action, Art, Pattern, Indoor, Jumbled, and Low-resolution images. In total, the training set images have 20 different categories that each of consists of 100 images. There are also 2000 saliency map images related to each of the images respectively. In each data-set, there are

also fixation maps and fixation locations for each image in the data-set. The test set consists of 20 categories with a saliency map related to each image as well. However, it doesn't have a fixation map and fixation locations that are publicly available. From the 2000 images in the training set, we used four categories consisting of Action, Pattern, Indoor, and social. We test our network on two categories of the images, including pattern and Action.

SALICON : The SALIency in CONtext (SALICON) data-set [4] gathers fixation points of different categories of images from the MS-COCO data-set [74] using mouse-contingent-tracking for multi-resolution images instead of eye-contingent-tracking. SALICON consists of 10000 training images, 5000 validation images, and 5000 test images. Training and validation images include saliency maps and fixation points, but for the test set, the saliency maps and fixation maps are not provided.

Chapter 5

Experimental Results

In this section, we present the results of Resnet-101 [24], Deeplab-V2 [25], and NASNet [26] for both training, and fine-tuning results. In the end, we add the test results from the SALICON data-set.

5.1 Training Resnet-101

We trained Resnet-101 by having Cat2000 images [21] as input, and having four scale heatmaps as ground-truth. The output produces four heatmaps, one for each scale. In this step, the network does not consider the saliency map from the Cat2000 dataset and depends only on the Heatmap from the image inpainting error stage. We then evaluate different saliency metrics for each scale of the output by comparing the output with the saliency map from Cat2000 dataset.

The results shown in Table 5.1, Table 5.2 and corresponding Tables that show the training accuracy for each category in Tables 5.3, and 5.4 are from the Resnet-101

network. We run the training network for 50 epochs, and we had validation every five epochs. We also consider the value of AUC-Borji, AUC-Judd, CC, and SIM in our training. We run our model until both validation and training loss were decreasing, but validation accuracy starts to decrease. We run Resnet-101 with a learning rate of the $1e-4$, and batch-size of 8. We used Adam optimization for the network. We tried different loss functions, including L1, MSE, MS-SSIM, KL-Divergence, and the combination of NSS with CC, but we found that the model works best with MSE as the loss function.

As is shown in the tables, the output of the network after training on Cat2000 and with the heatmap as ground-truth demonstrates that the network can predict the saliency map by just learning image inpainting error. Also, from the training Table 5.1 and validation Table 5.2, we can see that the output of the network for heatmap 32 seems predict the results better for both validation and training set than other scales. For training, we selected all 20 categories of Cat2000, we choose 90 images randomly from each category, and we reserve 10 images for the validation set. In total, we have 1800 images for training and 200 images for validation.

We also demonstrate the results for Resnet-101 after training images with the inpainting error heatmap as their ground-truths. As is shown, from the results of heatmap with masks16, and 32 in Table 5.3, we can see that heat-map-16 and heat-map-32 have AUC more than 0.82 for Sketch, Object, Random, and LineDrawing categories, while having a lower score for Satelite, Art, and OutdoorManMade. However, in Heatmap 64, and 128 as we can see in the Table 5.4, that LineDrawing, Satellite, and Sketch do not have high scores since having mask 64, and 128 leads

more error in the generated heatmap but it is still the case that Object, Sketch and Random have a higher score compared to the other categories.

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
Output-Train-Masks16	0.7788	0.7672	0.9405	0.6301	0.6787
Output-Train-Masks32	0.8041	0.7940	1.1067	0.7207	0.7032
Output-Train-Masks64	0.7898	0.7811	1.0698	0.6665	0.6939
Output-Train-Masks128	0.7804	0.7722	1.0390	0.5440	0.6592

Table 5.1: Result of training multi-scale Resnet-101 with 4-scale heatmap as ground-truth on the training set of Cat2000

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
Output-Val-Masks16	0.7696	0.7573	0.9123	0.6235	0.6749
Output-Val-Masks32	0.7957	0.7841	1.0570	0.7142	0.6961
Output-Val-Masks64	0.7814	0.7730	1.0132	0.6609	0.6942
Output-Val-Masks128	0.7770	0.7673	0.9988	0.5193	0.6522

Table 5.2: Result of training multi-scale Resnet-101 with 4-scale heatmap as ground-truth on the training set of Cat2000

Metric / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
Action	0.772(0.809)	0.762(0.798)	0.8316(1.090)	0.560(0.692)	0.662(0.696)
Affective	0.762(0.834)	0.748(0.822)	0.823(1.186)	0.615(0.719)	0.709(0.731)
Art	0.670(0.694)	0.668(0.687)	0.622(0.827)	0.544(0.773)	0.667(0.727)
BlackWhite	0.810(0.830)	0.797(0.817)	0.899(1.050)	0.704(0.828)	0.730(0.766)
Cartoon	0.780(0.831)	0.744(0.813)	0.832(1.099)	0.462(0.655)	0.671(0.713)
Fractal	0.737(0.793)	0.725(0.784)	0.885(1.005)	0.689(0.716)	0.727(0.735)
Indoor	0.773(0.817)	0.764(0.799)	0.825(0.989)	0.666(0.751)	0.747(0.771)
Inverted	0.760(0.791)	0.752(0.783)	0.772(1.040)	0.604(0.753)	0.699 (0.742)
Jumbled	0.760(0.782)	0.746(0.772)	0.835(0.913)	0.711(0.775)	0.740(0.755)
LineDrawing	0.825(0.748)	0.810(0.737)	1.101(0.809)	0.787(0.608)	0.667(0.606)
LowResolution	0.747(0.818)	0.738(0.804)	0.793(1.002)	0.635(0.711)	0.699(0.708)
Noisy	0.846(0.830)	0.836(0.819)	1.185(1.156)	0.788(0.809)	0.774(0.782)
Object	0.898(0.932)	0.875(0.915)	1.245(1.609)	0.626(0.728)	0.610(0.631)
OutdoorManMade	0.676(0.736)	0.672(0.728)	0.619(0.830)	0.500(0.652)	0.676(0.705)
OutdoorNatural	0.723(0.749)	0.707(0.740)	0.703(0.793)	0.541(0.672)	0.712(0.748)
Pattern	0.678(0.702)	0.670(0.691)	0.661(0.649)	0.571(0.583)	0.606(0.608)
Random	0.829(0.861)	0.823(0.845)	1.197(1.346)	0.626(0.817)	0.582(0.630)
Satelite	0.660(0.647)	0.651(0.642)	0.564(0.530)	0.495(0.564)	0.701(0.720)
Sketch	0.947(0.946)	0.934(0.937)	2.133(2.346)	0.670(0.708)	0.351(0.355)
Social	0.730(0.756)	0.714(0.743)	0.712(0.861)	0.667(0.763)	0.758(0.784)

Table 5.3: Validation Results on the output-heatmap 16(32) of training Resnet-101 for each category of Cat2000

5.1.1 Fine tuning Resnet-101 on SALICON

In fine-tuning Resnet-101 for the SALICON dataset, we used images from the SALICON dataset [4] as input to the network, and we have the fixation map as ground-truth for the network while the network uses image inpainting error information. Then, we evaluate the network by comparing the output of the network for training and validation sets with a fixation map, for which results are shown in the table 5.5. We run the network for 120 epochs. We measure the training loss every epoch and validation loss every five epochs, and we stopped our training when both training and validation loss was still decreasing, but the evaluation metrics, including AUC, CC, and SIM, start to increase. We fine-tuned Resnet-101 by keeping all the layers of the training network, and we load the model for its learned weights, then we added another convolution layer having sigmoid as its activation function to change the output of the model from 4 scales to 1. We used the learning rate of $1e-4$, Adam optimization, and MSE as the loss function. We use the weights of epoch 119 to test the model.

5.1.2 Fine tuning Resnet-101 on CAT2000

By having weights from the training step, we fine-tune our model with images from Cat2000 as input and the saliency map from Cat2000 as ground-truth. We run our model for 80 epochs until the training loss was decreasing, but validation loss starts to increase. We used MSE as a loss function and Adam as an optimizer, and we used the learning rate of $1e-4$.

Metric / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
Action	0.808(0.835)	0.798(0.825)	1.161(1.255)	0.680(0.573)	0.717(0.692)
Affective	0.839(0.825)	0.828(0.814)	1.212(1.140)	0.688(0.639)	0.733(0.722)
Art	0.688(0.742)	0.688(0.737)	0.770(0.971)	0.757(0.655)	0.740(0.709)
BlackWhite	0.819(0.798)	0.808(0.790)	1.060(1.132)	0.803(0.655)	0.771(0.722)
Cartoon	0.842(0.863)	0.823(0.845)	1.128(1.176)	0.651(0.553)	0.718(0.684)
Fractal	0.797(0.849)	0.795(0.836)	1.039(1.196)	0.687(0.594)	0.738(0.688)
Indoor	0.830(0.889)	0.816(0.878)	1.076(1.482)	0.733(0.569)	0.776(0.731)
Inverted	0.792(0.864)	0.788(0.853)	1.074(1.433)	0.739(0.623)	0.748(0.721)
Jumbled	0.789(0.809)	0.782(0.804)	0.969(1.059)	0.761(0.633)	0.755(0.716)
LineDrawing	0.539(0.754)	0.538(0.743)	0.170(0.786)	0.150(0.420)	0.510(0.552)
LowResolution	0.823(0.863)	0.814(0.849)	1.088(1.339)	0.728(0.615)	0.722(0.698)
Noisy	0.835(0.858)	0.822(0.849)	1.176(1.270)	0.690(0.599)	0.769(0.729)
Object	0.919(0.900)	0.906(0.887)	1.610(1.515)	0.729(0.636)	0.648(0.614)
OutdooorManMade	0.737(0.771)	0.729(0.766)	0.864 (0.988)	0.649 (0.570)	0.710(0.692)
OutdoorNatural	0.763(0.796)	0.751(0.788)	0.883(1.032)	0.646(0.541)	0.753(0.723)
Pattern	0.647(0.480)	0.648(0.470)	0.536(-0.15)	0.514(-0.16)	0.589(0.428)
Random	0.848(0.850)	0.838(0.833)	1.302(1.242)	0.812(0.774)	0.662(0.599)
Satelite	0.608(0.590)	0.609(0.588)	0.425(0.338)	0.540(0.468)	0.720(0.694)
Sketch	0.921(0.397)	0.908(0.394)	1.751(-0.35)	0.506(-0.22)	0.311(0.163)
Social	0.775(0.799)	0.761(0.791)	0.963(1.123)	0.746(0.660)	0.786(0.757)

Table 5.4: Validation Results on the output-heatmap 64(128) of training Resnet-101 for each category of Cat2000

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
TrainingSet	0.8834	0.8788	2.560	0.8921	0.794
ValidationSet	0.8764	0.8633	2.498	0.8832	0.793

Table 5.5: Fine-tunning Resnet on Salicon. The pre-trained Resnet-101 first trained on Cat2000 with heatmap as GT

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
TrainingSet	0.9278	0.9111	5.5248	0.2615	0.1884
ValidationSet	0.9016	0.84288	3.5842	0.1381	0.2326

Table 5.6: Fine-tunning Resnet on CAT2000. The pre-trained Resnet-101 first trained on Cat2000 with heatmap as GT

5.1.3 Directly training on Cat2000

In directly training Resnet-101 for Cat2000, we just have one step in which we train Resnet-101 by having the input of Cat2000, and Cat2000 fixation maps as the ground-truth with no intermediate inpainting step. We used the same architecture that we used for training except for the last layer, where instead of generating 4 scale outputs, it just generates one output related to each input. We run Resnet for 20

epochs before overfitting begins to occur on the data with a learning rate of $1e-5$, MSE loss, and Adam optimization.

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
TrainingSet	0.8956	0.9078	3.8652	0.4223	0.3999
ValidationSet	0.8741	0.8389	3.39134	0.4383	0.4397

Table 5.7: Directly training Resnet on CAT2000

5.1.4 Directly training on SALICON

We used Resnet-101 to train the Salicon dataset directly without using image inpainting information. We had SalICon images as input and saliency maps as ground-truth instead of having a 4-scale heatmap. We run the model for 25 epochs with the learning rate of $1e-5$ and Adam optimizer.

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
TrainingSet	0.7534	0.7428	2.562	0.791	0.682
ValidationSet	0.7423	0.7320	2.482	0.789	0.673

Table 5.8: Directly training Resnet-101 on the SALICON dataset

5.2 Training with Deeplab-V2

We trained DeepLab-V2 to learn inpainting error by having images from Cat2000 as input and having the 4-scale heatmap as ground-truths. We run deeplab-v2 for 120 epochs with the learning rate of $1e-4$. We used Adam as an optimizer, and we used MSE as our loss function.

The results of the Deeplab-v2 after training and learning for both the training set and validation set are in table 5.9, and 5.10. We compare the output of the network, which had four scale outputs with the fixation map ground-truths. The generated output by learning image inpainting error only, can predict fixation locations more than 78%. In Deeplab-V2, the accuracy of the output for the heatmap with mask 128 seems to be higher compared to other scales.

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
Output-Train-Masks16	0.8147	0.8004	1.0100	0.6366	0.6787
Output-Train-Masks32	0.8258	0.8152	1.2018	0.6872	0.7014
Output-Train-Masks64	0.7809	0.7714	1.0254	0.6459	0.6767
Output-Train-Masks128	0.8316	0.8205	1.1966	0.6978	0.7026

Table 5.9: Evaluation metrics on the output of training multi-scale Deeplab-V2 on the training set of Cat2000

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
Output-Val-Masks16	0.7892	0.7755	0.9350	0.6095	0.6725
Output-Val-Masks32	0.8351	0.8221	1.2036	0.6712	0.6968
Output-Val-Masks64	0.7873	0.7760	1.0113	0.6392	0.6767
Output-Val-Masks128	0.8398	0.8269	1.1901	0.6840	0.6976

Table 5.10: Evaluation metrics on the output of training multi-scale Deeplab-V2 on the validation set of Cat2000

Fine tuning DeepLab-V2 on SALICON

We fine-tune DeepLab-V2 for the SALICON data-set by having the original images from the SALICON data-set as input and fixation map as ground-truth. We used the same architecture, and we load the weights from training DeepLab-V2 on Cat2000 [] images with the inpainting error heatmaps. We add another layer on top of the network to change the output channels from 4 to 1. The learning rate of $1e-5$ worked better for fine-tuning on SALICON, and with the Adam optimizer. We run the model for 25 epochs, where training loss was decreasing, but validation loss starts to increase.

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
TrainingSet	0.9223	0.9451	2.588	0.4335	0.5851
ValidationSet	0.9153	0.8793	2.180	0.4127	0.5671

Table 5.11: Fine tuning DeepLab-V2 on SALICON dataset with pre-trained weight of DeepLabV2 trained on Cat2000 dataset with heatmap

Fine tuning DeepLab-V2 on Cat2000

We fine-tune DeepLab-V2 for the Cat2000 data-set by having weights from training DeepLab-V2 on Cat2000 inpainting error heatmaps. For fine-tuning, we passed the original Cat2000 images as input and Cat2000 saliency maps as ground-truth. We train the network for 50 epochs where both validation and training loss were still decreasing, and evaluation metrics, including AUC, NSS, and cc began increasing in training, and validation.

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
TrainingSet	0.9675	0.9469	4.4308	0.4634	0.6331
ValidationSet	0.9479	0.9332	3.4863	0.4995	0.6424

Table 5.12: Fine tuning DeepLab-V2 on Cat2000 dataset with pre-trained weight of DeepLabV2 trained on Cat2000 dataset with Heatmap

5.3 Training with NasNet

We trained NasNet to learn image inpainting error without using pre-trained weights on ImageNet. We keep the architecture of NasNet as it is, and we just changed the last layer of NasNet to produce 4-scale heatmaps instead of a single output. We train NassNet for epochs with the Adam optimizer, and MSE loss function, and with the learning rate of .

The result of the training is shown in tables 5.13, and 5.14. We compare each scale of the NASNet output with the saliency map and the fixation map of the Cat2000 data-set. As is shown in the table, the network is able to predict saliency more than 77%. Also, it seems that the heatmap generated with mask scale 32 works better compare to the other heatmap outputs both in training and validation.

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
Output-Train-Masks16	0.7576	0.7502	0.9094	0.6172	0.6935
Output-Train-Masks32	0.8004	0.7912	1.0970	0.7148	0.7190
Output-Train-Masks64	0.7768	0.7695	1.0234	0.6808	0.7038
Output-Train-Masks128	0.7949	0.7859	1.0783	0.6369	0.6897

Table 5.13: Evaluation metrics on the output of NasNet on the training set of Cat2000

Metrics / Category	AUC-Judd	AUC-Borji	NSS	CC	SIM
Output-Val-Masks16	0.7626	0.7533	0.8763	0.6058	0.6812
Output-Val-Masks32	0.7821	0.7742	1.02894	0.7124	0.7180
Output-Val-Masks64	0.7743	0.7656	1.0015	0.6999	0.7069
Output-Val-Masks128	0.7907	0.7822	1.0418	0.6316	0.6872

Table 5.14: Evaluation metrics on the output of NasNet on the training set of Cat2000

5.4 Test

To further validate our test, we submitted our results on the SALICON [4] dataset, where we do not have access to the ground-truth for evaluation.

Test Resnet-101 on Salicon Dataset

In table 5.15, we can see the results of Resnet-101 on four different cases.

- First, we trained Resnet-101 with pre-trained weights on ImageNet with images from Cat2000 as input and heatmap as ground-truth to learn which parts of the image are more difficult to predict and have high inpainting error. Then, we fine-tune our model by having both input and ground-truth from the SALICON data-set and by loading the learned weights from the inpainting error training step. In the end, we pick the weights from the fine-tuning step when we have the best accuracy, and we test this model on the Salicon data-set, and we sent our results to get the saliency evaluation metrics from the test server.

Metrics / Category	AUC	CC	KL	sAUC	IG	NSS	SIM
F:Resnet+Inpainting+ImageNet	0.855	0.873	1.192	0.733	-0.158	1.863	0.763
F: Resnet+Inpainting-ImageNet	0.834	0.781	1.607	0.695	-0.724	1.641	0.639
D: Resnet-Inpainting+ImageNet	0.800	0.639	0.931	0.669	-0.108	1.305	0.607
D: Resnet-Inpainting-ImageNet	0.662	0.425	11.619	0.543	-15.332	0.872	0.360

Table 5.15: Results of Resnet-101 on Salicon test-set, F means fine-tuning Resnet-101 on Salicon dataset by having pre-trained weight derived from training Resnet-101 with Cat2000 images and image inpainting error as a heatmap. D means training directly on salicon dataset and saliency ground truth.

- Second, we trained Resnet-101 from scratch without using pre-trained ImageNet weights. In training, Resnet-101 learns to predict inpainting error by having Cat2000 images as input and inpainting error heatmaps as ground-truth. Then, we fine-tune the model with the weights from this training step and also without using the pre-trained weights from ImageNet, and we used the SALICON data-set for both input and ground-truth. We then test this model on the Salicon data-set with the best weights from the fine-tuning step.

- In the third case, we train Resnet-101 directly on the SALICON data-set. We used Resnet-101 with pre-trained weights from ImageNet to learn saliency. We used both input and Ground-truth from the SALICON data-set.

- In the last case, we train Resnet-101 directly on the SALICON data-set. We used both input and ground-truth from the SALICON data-set without using

the inpainting error heatmaps and compared with using the image inpainting error information. We then test the network on the SALICON test-set by picking the best epoch performance on the training part for each case.

As is shown in table 5.15, if we train Resnet-101 with image inpainting error information and by using ImageNet as pre-trained weight, we can get a better score compared to other scenarios. We can conclude that learning Image Inpainting error can help us to predict saliency better compared to the case where we didn't first train the network for the inpainting error heat map, and we just fine-tune the network to learn the saliency map directly.

Test NasNet on Salicon Dataset

In table 5.16, we test NasNet on Salicon data-set in two scenarios.

Metrics / Category	AUC	CC	KL	sAUC	IG	NSS	SIM
F:NasNet+Inpainting-ImageNet	0.851	0.803	0.386	0.705	0.565	1.675	0.686
D:NasNet-Inpainting-ImageNet	0.850	0.802	0.397	0.705	0.545	1.677	0.680

Table 5.16: Results of NasNet on Salicon test-set. F means fine-tuning NASNet on Salicon dataset by having pre-trained weights from training NASNet on Cat2000 with inpainting error heatmap as GT. D means directly training NASNet on Salicon dataset.

In the first scenario, we train NasNet without having pre-trained weights and with input from Cat2000 images and inpainting error heatmaps as ground-truth. Then,

we used the weights from the training step, and we fine-tune NasNet by having both input and ground-truth from the SALICON dataset.

In the second scenario, we train NasNet directly on the SALICON dataset without using the intermediate step of an inpainting error heatmap as ground-truth. We use both input and ground-truth from the SALICON dataset, and then we test it on the test-set SALICON dataset.

Test DeepLab-v2 on Salicon Dataset

In table 5.17, we test DeepLab-V2 on the Salicon data-set in two scenarios.

In the first scenario, we train DeepLab-V2 with pre-trained weights and with input from Cat2000 images and with heatmaps as ground-truths. Then, we used the weights from the training step, and we fine-tune DeepLab-V2 by having both input and ground-truth from the SALICON dataset.

In the second scenario, we train DeepLab-V2 directly on the SALICON dataset without using a heatmap as ground-truth. We use both input and ground-truth from the SALICON dataset, and then we test it on the test-set SALICON dataset.

Metrics / Category	AUC	CC	KL	sAUC	IG	NSS	SIM
F: Deeplab-V2+Inpainting	0.818	0.710	0.641	0.659	0.116	1.450	0.568
D: Deeplab-V2-Without-Inpainting	0.774	0.669	0.654	0.601	0.119	1.511	0.578

Table 5.17: Results of DeepLab-V2 on Salicon test-set. F means fine-tuning DeepLabV2 on Salicon dataset by having pre-trained weights from training Resnet-101 on Cat2000 inpainting error heatmaps. D means training directly on SALICON dataset without having heatmap as GT and going directly to fixations

Metrics / Category	AUC-Judd	sAUC	CC	NSS
Resnet+Inpainting	0.855	0.733	0.873	1.863
NASNet+Inpainting	0.851	0.705	0.803	1.675
DeepLabV2+Inpainting	0.818	0.659	0.710	1.450
EML-NET[56]	0.866	0.746	**0.886**	2.050
SAM-ResNet [10]	0.883	**0.779**	0.842	**3.204**
DSCLRCN [18]	0.884	0.776	0.831	3.157
SAM-VGG [10]	0.881	0.774	0.825	3.143
ML-Net [9]	0.866	0.768	0.743	2.789
MixNet [86]	0.861	0.771	0.730	2.767
Kruthiventi et al [87]	0.880	0.760	0.780	2.610
SALGAN [14]	0.781	0.772	0.781	2.459
SalNet [54]	0.858	0.724	0.622	1.859
DeepGaze 2 [13]	0.885	0.761	0.509	1.336
Human Observer [13]	0.89	-	-	-

Table 5.18: State of The art results on SALICON

Chapter 6

Discussion

Predicting saliency not only depends on the architecture we are using but also depends on the type of data we are training the model on. Recent advances and development of new architectures alongside with large available data-sets for human gaze prediction [21] and mouse tracking [11] has led to some improvement in the task of saliency prediction. However, there is still a gap between what saliency models predict and what grabs human eyes when looking at a scene.

Bruce [88] pointed out that the manner in which we select data, and the generated ground-truth have a crucial impact on model performance. Large scale datasets such as SALICON [11] helped to better model saliency predictions; however, there is a gap between choosing a fixation map with mouse clicking and with eye movements. Therefore, refining previous data-sets or introducing a new data-set for different types of categories would be rewarding. In addition, the work of [16] proved that saliency could benefit from features learned from object recognition [20]. Saliency might also benefit from domains other than object detection and image classification.

In this work, we investigate the effect of image inpainting as a new domain to the set of tasks that might transfer to saliency prediction tasks. Since object recognition only provides information for a limited number of categories, we consider to provide a new data-set for the task of saliency predicting made of inpainting error images, and investigate the effect of it on the saliency prediction. Our goal to use image inpainting is that image inpainting fails to fill in a region when there is something that is different from its surroundings or perhaps that attracts human attention since image inpainting looks for similar patterns and structures in the unmasked parts of the image. With the same network architecture and with adding image inpainting error as a middle step, we got better results with inpainting 5.15 compare to the case that we did not use image inpainting as a middle step. Also, in training, we found that image inpainting can help to capture objects, sketch, and line-drawing patterns better. Also, image inpainting as an intermediate step can help to predict noisy images and low-resolution images better.

How can we apply these features to learn other tasks? One can fine tune objectness for specific categories. Therefore, it is useful also as a prior stage to any object recognition task regardless of what the end stage or application is. Inpainting allows for feature learning and the goal is really feature learning with the possibility of transferring this to other tasks and testing them. Saliency is one such task and is especially suitable given its affinity for content that stands out or is surprising.

While our results are close to the state of the art, our model does not obtain the best performance on the SALICON data-set. Since a heat-map that is generated from image inpainting error can distinguish where there is an object, it also considers the

boundary of the object as something important while fixation data-sets tend to just focus on the center of the object. For a metric like AUC, our Resnet-101 model can achieve the score of 0.85 while the first state-of-the-art model named EML-Net [56] achieved the score of 0.86. Also, we achieved a score of 0.733 for sAUC, and EML-Net achieved a score of 0.746 for sAUC. AUC is mostly based on true positives while it does not significantly penalize false positives, the close AUC score that we achieved shows that our model can predict most of the true positives of fixation points. Also, our CC is 0.87 which is very close to the EML-Net, which has 0.886. In addition, the second rank for the saliency is SAM-ResNet [9], which has the AUC of 0.88, sAUC of 0.779, and CC of 0.842 while our model achieved higher CC 0.87 compared to the SAM-Resnet which achieved the score of 0.84 and lying in second place.

Kummerer et al [13] proposed that sometimes a model with a lower accuracy can predict saliency for some images for which another model with higher accuracy cannot predict correctly. As we could see in Figure 6.4, a model with having image inpainting as a middle step can predict saliency better compare to the model without using image inpainting. Here, the salient part of the image is the region when there is something that stands out from the rest of the image.

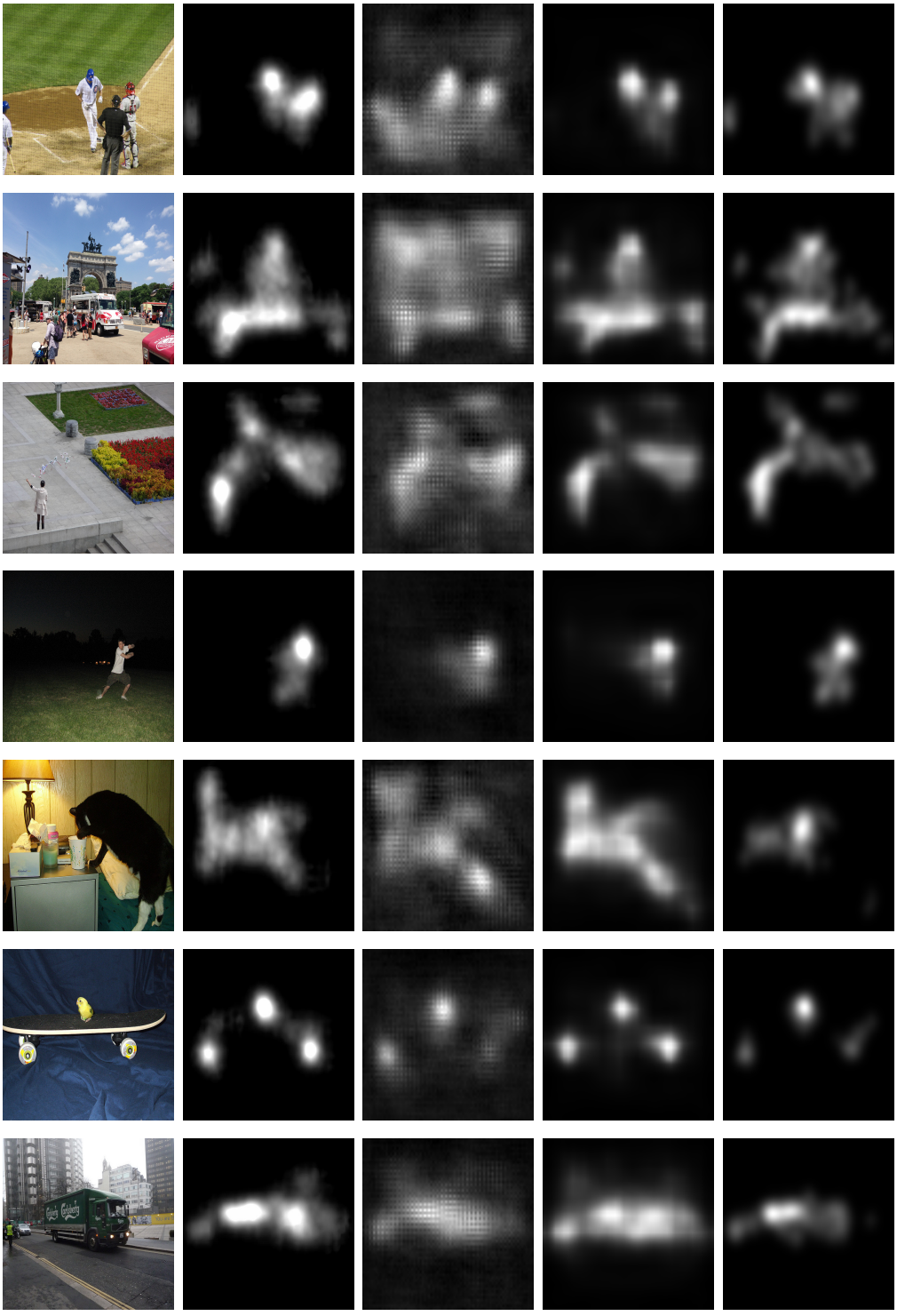

Figure 6.1: The result of fine-tuning on Salicon validation set. The first column is the original image, the second one is the output of Resnet-101, the third one is the output of Deeplab-V2, the fourth one is the output of the Nasnet and the last one is the ground-truth

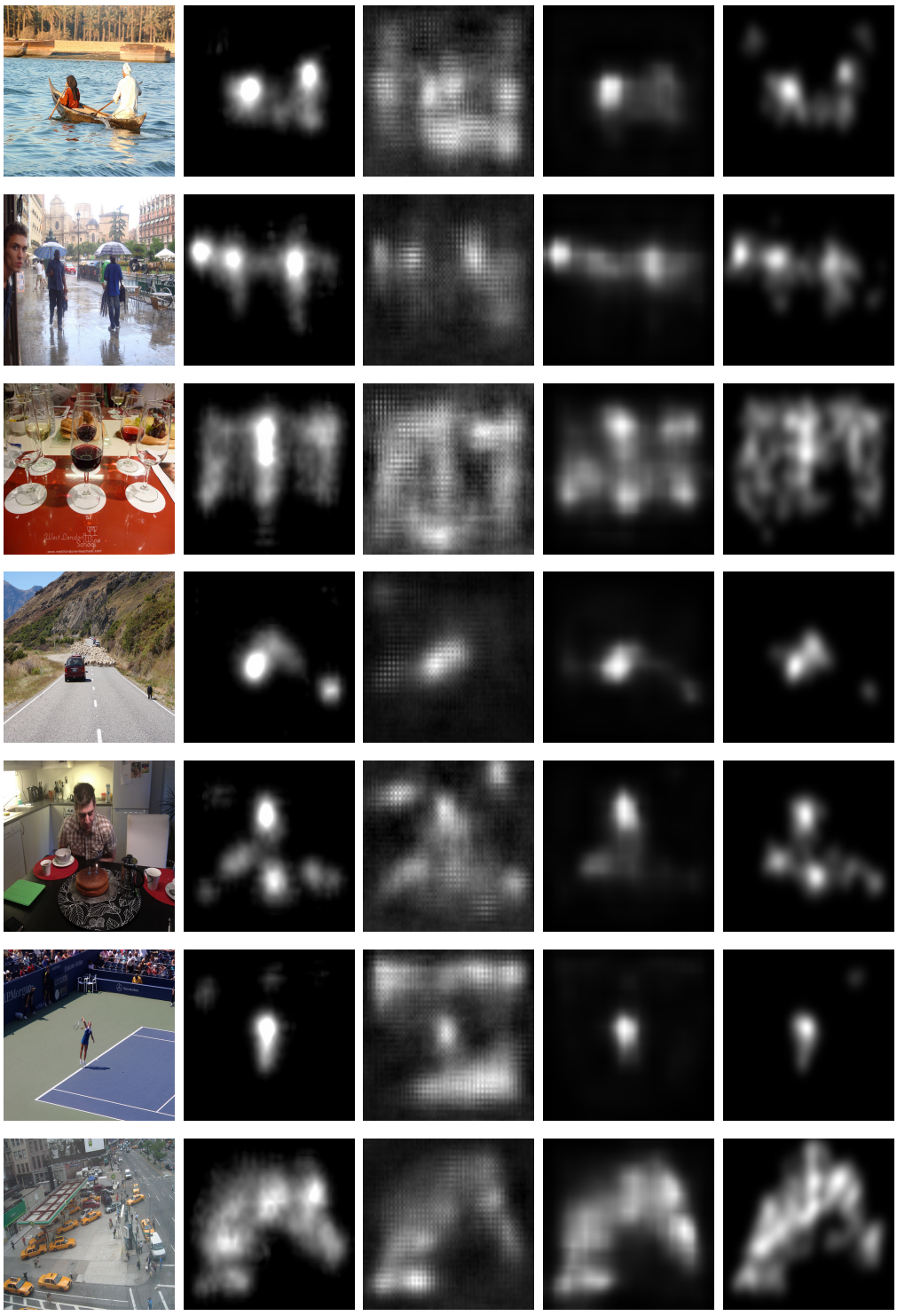

Figure 6.2: The result of fine-tuning on Salicon validation set. The first column is the original image, the second one is the output of Resnet-101, the third one is the output of Deeplab-V2, the fourth one is the output of the Nasnet and the last one is the ground-truth

Figure 6.3: The effect of learning image inpainting in predicting standout patterns. The first and forth columns are input, the second and the fifth column is the output of the Resnet-101 trained on heatmap and fine-tuned on salicon. The third and the sixth columns are the output of Resnet-101 without training on inpainting error heat-maps

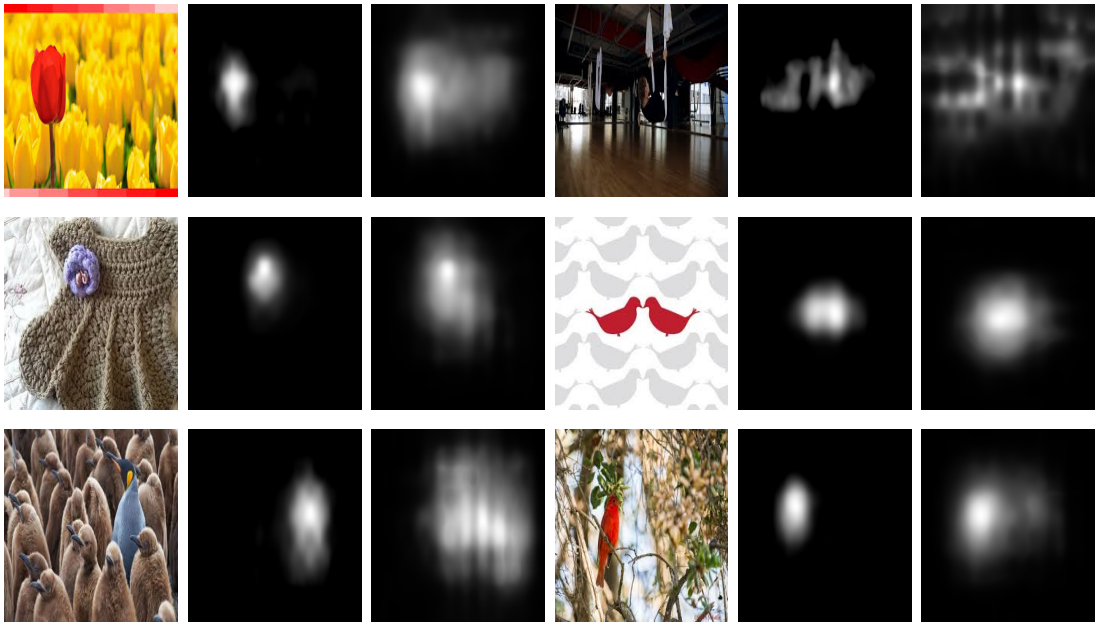

Figure 6.4: The effect of learning image inpainting in predicting standout patterns. The first and forth columns are input, the second and the fifth column is the output of the Resnet-101 trained on heatmap and fine-tuned on Salicon. The third and the sixth columns are the output of Resnet-101 without training on inpainting error heat-maps

6.1 Conclusion

In this work, we presented an approach to predict saliency based on image inpainting error. Our proposed approach consists of three main streams. First, we generated ground-truths for our model based on the difference between an original image and the inpainted image. Second, we trained three different networks including Resnet-101, Deeplab-V2, and NasNet to learn the heatmap for each image in the Cat2000 dataset. Third, we used transfer learning to learn saliency by getting help from learning image inpainting error. Overall this methods appears to be successful and especially reveals interesting properties that emerge in the qualitative results. This marks the first effort to consider inpainting as a precursor to saliency prediction and more exploration is warranted.

Bibliography

[1] L. Zheng, Y. Zhang, and V. L. Thing, "A survey on image tampering and its detection in real-world photos," *Journal of Visual Communication and Image Representation*, vol. 58, pp. 380–399, 2019.

[2] E. Ardizzone, A. Bruno, and G. Mazzola, "Copy–move forgery detection by matching triangles of keypoints," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2084–2094, 2015.

[3] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5505–5514.

[4] 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*, 2015, pp. 1072–1080.

[5] S. Rahman and N. Bruce, "Saliency, scale and information: Towards a unifying theory," in *Advances in Neural Information Processing Systems*, 2015, pp. 2188–2196.

[6] A. Borji, M.-M. Cheng, H. Jiang, and J. Li, "Salient object detection: A benchmark," *IEEE transactions on image processing*, vol. 24, no. 12, pp. 5706–5722, 2015.

[7] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu, "Global contrast based salient region detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 569–582, 2015.

[8] E. Vig, M. Dorr, and D. 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*, 2014, pp. 2798–2805.

[9] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara, "A deep multi-level network for saliency prediction," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 3488–3493.

[10] ——, "Predicting human eye fixations via an lstm-based saliency attentive model," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5142–5154, 2018.

[11] X. Huang, C. Shen, X. Boix, and Q. Zhao, "Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 262–270.

[12] S. Jetley, N. Murray, and E. Vig, "End-to-end saliency mapping via probability distribution prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5753–5761.

[13] M. Kummerer, T. S. Wallis, L. A. Gatys, and M. Bethge, "Understanding low-and high-level contributions to fixation prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4789–4798.

[14] J. Pan, C. C. Ferrer, K. McGuinness, N. E. O'Connor, J. Torres, E. Sayrol, and X. Giro-i Nieto, "Salgan: Visual saliency prediction with generative adversarial networks," *arXiv preprint arXiv:1701.01081*, 2017.

[15] H. R. Tavakoli, A. Borji, J. Laaksonen, and E. Rahtu, "Exploiting inter-image similarity and ensemble of extreme learners for fixation prediction using deep features," *Neurocomputing*, vol. 244, pp. 10–18, 2017.

[16] M. Kümmerer, L. Theis, and M. Bethge, "Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet," *arXiv preprint arXiv:1411.1045*, 2014.

[17] W. Wang and J. Shen, "Deep visual attention prediction," *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2368–2378, 2017.

[18] N. Liu and J. Han, "A deep spatial contextual long-term recurrent convolutional network for saliency detection," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3264–3274, 2018.

[19] S. S. Kruthiventi, K. Ayush, and R. V. Babu, "Deepfix: A fully convolutional neural network for predicting human eye fixations," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4446–4456, 2017.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[21] A. Borji and L. Itti, "Cat2000: A large scale fixation dataset for boosting saliency research," *arXiv preprint arXiv:1505.03581*, 2015.

[22] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 85–100.

[23] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2018.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[26] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[27] Z. Bylinskii, T. Judd, A. Borji, L. Itti, F. Durand, A. Oliva, and A. Torralba, "Mit saliency benchmark."

[28] G. T. Buswell, "How people look at pictures: a study of the psychology and perception in art." 1935.

[29] A. M. Treisman and G. Gelade, "A feature-integration theory of attention," *Cognitive psychology*, vol. 12, no. 1, pp. 97–136, 1980.

[30] C. Koch and S. Ullman, "Shifts in selective visual attention: towards the underlying neural circuitry," in *Matters of intelligence*. Springer, 1987, pp. 115–141.

[31] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 11, pp. 1254–1259, 1998.

[32] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," in *Advances in neural information processing systems*, 2007, pp. 545–552.

[33] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "Sun: A bayesian framework for saliency using natural statistics," *Journal of vision*, vol. 8, no. 7, pp. 32–32, 2008.

[34] R. Valenti, N. Sebe, T. Gevers *et al.*, "Image saliency by isocentric curvedness and color." in *ICCV*, vol. 1, no. 3, 2009, p. 6.

[35] E. Erdem and A. Erdem, "Visual saliency estimation by nonlinearly integrating features using region covariances," *Journal of vision*, vol. 13, no. 4, pp. 11–11, 2013.

[36] T. Judd, K. Ehinger, F. Durand, and A. Torralba, "Learning to predict where humans look," in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 2106–2113.

[37] A. Borji, "Boosting bottom-up and top-down visual features for saliency estimation," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 438–445.

[38] A. Garcia-Diaz, X. R. Fdez-Vidal, X. M. Pardo, and R. Dosil, "Saliency from hierarchical adaptation through decorrelation and variance normalization," *Image and Vision Computing*, vol. 30, no. 1, pp. 51–64, 2012.

[39] G. Kootstra and L. R. Schomaker, "Prediction of human eye fixations using symmetry," in *The 31st Annual Conference of the Cognitive Science Society (CogSci09)*. Cognitive Science Society, 2009, pp. 56–61.

[40] M. Liang and X. Hu, "Predicting eye fixations with higher-level visual features," *IEEE Transactions on Image Processing*, vol. 24, no. 3, pp. 1178–1189, 2015.

[41] J. Zhang and S. Sclaroff, "Saliency detection: A boolean map approach," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 153–160.

[42] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2007, pp. 1–8.

[43] A. Oliva, A. Torralba, M. S. Castelhano, and J. M. Henderson, "Top-down control of visual attention in object detection," in *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, vol. 1. IEEE, 2003, pp. I–253.

[44] N. Bruce and J. Tsotsos, "Saliency based on information maximization," in *Advances in neural information processing systems*, 2006, pp. 155–162.

[45] N. D. Bruce and J. K. Tsotsos, "Saliency, attention, and visual search: An information theoretic approach," *Journal of vision*, vol. 9, no. 3, pp. 5–5, 2009.

[46] A. Borji and L. Itti, "Exploiting local and global patch rarities for saliency detection," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 478–485.

[47] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[49] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*, 2014, pp. 647–655.

[50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[51] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[52] N. Liu, J. Han, D. Zhang, S. Wen, and T. Liu, "Predicting eye fixations using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 362–370.

[53] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[54] J. Pan, E. Sayrol, X. Giro-i Nieto, K. McGuinness, and N. E. O'Connor, "Shallow and deep convolutional networks for saliency prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 598–606.

[55] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew *et al.*, "Extreme learning machine: a new learning scheme of feedforward neural networks," *Neural networks*, vol. 2, pp. 985–990, 2004.

[56] S. Jia and N. D. Bruce, "Eml-net: An expandable multi-layer network for saliency prediction," *arXiv preprint arXiv:1805.01047*, 2018.

[57] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[58] N. D. Bruce, C. Catton, and S. Janjic, "A deeper look at saliency: Feature contrast, semantics, and beyond," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 516–524.

[59] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 891–898.

[60] T. Oyama and T. Yamanaka, "Influence of image classification accuracy on saliency map estimation," *CAAI Transactions on Intelligence Technology*, vol. 3, no. 3, pp. 140–152, 2018.

[61] A. Kroner, M. Senden, K. Driessens, and R. Goebel, "Contextual encoder-decoder network for visual saliency prediction," *arXiv preprint arXiv:1902.06634*, 2019.

[62] A. Telea, "An image inpainting technique based on the fast marching method," *Journal of graphics tools*, vol. 9, no. 1, pp. 23–34, 2004.

[63] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[64] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE transactions on image processing*, vol. 10, no. 8, pp. 1200–1211, 2001.

[65] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," in *ACM Transactions on Graphics (ToG)*, vol. 28, no. 3. ACM, 2009, p. 24.

[66] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on graphics (TOG)*, vol. 22, no. 3, pp. 313–318, 2003.

[67] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 107, 2017.

[68] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[69] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6721–6729.

[70] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9446–9454.

[71] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, "Edgeconnect: Generative image inpainting with adversarial edge learning," *arXiv preprint arXiv:1901.00212*, 2019.

[72] Y. Song, C. Yang, Z. Lin, X. Liu, Q. Huang, H. Li, and C.-C. Jay Kuo, "Contextual-based image inpainting: Infer, match, and translate," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[73] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[74] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[75] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[76] Y. Bengio, P. Simard, P. Frasconi *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[77] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[78] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[79] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[80] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[81] Z. Bylinskii, A. Recasens, A. Borji, A. Oliva, A. Torralba, and F. Durand, "Where should saliency models look next?" in *European Conference on Computer Vision*. Springer, 2016, pp. 809–824.

[82] M. Kümmerer, T. Wallis, and M. Bethge, "How close are we to understanding image-based saliency?" *arXiv preprint arXiv:1409.7686*, 2014.

[83] R. J. Peters, A. Iyer, L. Itti, and C. Koch, "Components of bottom-up gaze allocation in natural images," *Vision research*, vol. 45, no. 18, pp. 2397–2416, 2005.

[84] M. Kümmerer, T. S. Wallis, and M. Bethge, "Information-theoretic model comparison unifies saliency metrics," *Proceedings of the National Academy of Sciences*, vol. 112, no. 52, pp. 16 054–16 059, 2015.

[85] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[86] S. F. Dodge and L. J. Karam, "Visual saliency prediction using a mixture of deep neural networks," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4080–4090, 2018.

[87] S. S. Kruthiventi, V. Gudisa, J. H. Dholakiya, and R. Venkatesh Babu, "Saliency unified: A deep architecture for simultaneous eye fixation prediction and salient object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5781–5790.

[88] N. D. Bruce, C. Wloka, N. Frosst, S. Rahman, and J. K. Tsotsos, "On computational modeling of visual saliency: Examining whats right, and whats left," *Vision research*, vol. 116, pp. 95–112, 2015.