INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 6.078
(Volume 6, Issue 6)
Available online at: https://www.ijariit.com

Smart image enhancement technique by removal of undesirable objects/background from the image

Pankaj Tanwar
p.tanwar@samsung.com
Samsung Research and Development Institute, Noida, Uttar Pradesh

Karishma Kumari
k6.kumari@samsung.com
Samsung Research and Development Institute, Noida, Uttar Pradesh

Saqib Kamal
saqib.kamal@samsung.com
Samsung Research and Development Institute, Noida, Uttar Pradesh

ABSTRACT

The devil is always in your picture. You always miss the best shot and that particular tiny object in your photo can worsen the entire composition and result in something opposite from what you hoped for. The proposed AI based image enhancement model will first detect unwanted objects/background based on the training of large set of already edited images by users then intelligently reconstructs the image without those objects. We have prepared a dataset with distracting elements in the images and used it to train our predictor model which predicts the distracting regions and thereafter used image-inpainting to remove those areas which results in a standalone system for distractor removal with no user input. In the proposed method of this paper, the image is first segmented using Convolutional networks for semantic segmentations and then each segment is classified in terms of the score of distractors on the basis of various features which almost covers all types of distractors in an image. Our main focus was to collect the data which contains all kinds of distractors and then deciding the features which classify an object as a distractor in an image. Detection and removal of distracting regions helps to enhance the beauty and visual quality of the image which can be fulfilled by using our model.

Keywords: *Image Segmentation, Visual Distractor, Salient Regions, Visual Attention, Image Enhancement, Random Forest*

1. INTRODUCTION

Capturing an image is too easy, but capturing a perfect image is never. Furthermore, editing the image after capturing it has also never been a simple job. It takes much effort and extreme care to capture or edit an image to make it exquisite and elegant. A common saying is "Beauty is in the eye of the beholder," which means beauty doesn't exist independently but is created by observers. To be a beholder, you have to pay attention to it.

Nevertheless, making people pay attention to an image for its beauty is not easy. Many image processing tools are currently available in the market, providing users with many options and controls to edit an image. However, those tools are complicated to use and require a lot of effort and knowledge to use them.

Tools like Adobe Photoshop provide professional photographers with many functions and manipulation techniques to modify and create high effect images with aesthetically pleasing features. However, this tool is not that easy to use for everyone; one needs to have a high level of experience and expertise.

Ninety-five million photos and videos are shared on Instagram per day. Most of the Instagram influencers and bloggers want their posted pictures to be perfect, so they may be benefited from a semi or fully automated system which can automatically predict the distracting objects or regions from an image and remove it without manual interference. Instagram and other image sharing platforms offer features like Editing and applying filters, but they are mainly focused on manipulating global properties such as exposure and tone, which enhances the texture of the image but does not cover the manipulation of the image's content. There can be many examples where we can see that an image has consisted of other insignificant information that does not contribute to the main subject of an image; instead distracts the viewer from focusing on the image. For example, trash on the ground, a person photobombing in the image, any object partially captured or out of frame.

Previous works have been done to address this problem, but most of them mainly focus on predicting certain kinds of distractors, whereas our work is focused on generalizing the concept of the distractor in an image. Despite limiting the absolute no. of possible distractors an image can contain [1, 2, 3, and 4], we tried to cover every kind of distractor by selecting specific features that any possible distractor can have in the image. To address this challenge, we first segmented [5] the image to get all the possible distinct objects or regions and then labeled each of them as distractors or non-distractor. For each segment, a feature vector is generated and then trained with multiple Machine Learning Classifiers [6] to map each segment with 1(distractor) and 0 (for non-distractor) value. Our main contribution is:

a) Collection of images containing various kinds of distractors to create a dataset.
b) Developing a Web-Tool to enable the user to annotate distracting regions of an image.
c) Distinctive feature selection containing distractor properties.
d) Training a machine learning model to produce a distractor prediction model for new images
e) Removal of predicted regions from the inputted image and regeneration of the image using Image-inpainting.

In the following sections, we describe related work (Section 2), describe the technical aspects of our proposed method (Section 3), Performance Evaluation, and results on different models (Section 5) and Conclusion and Future work of our proposed method (Section 6).

2. RELATED WORK

2.1 Finding Distractors in Images

Our work has been inspired by [1], which presents a system to improve an image by removing the region, which draws attention away from the main subject in the image. Their work is focused on building a model for automatic distractor removal from images. Given input images and user labels for distracting areas in an image, they have constructed a model for learning distractor maps. First, the images are segmented using multi-scale combinatorial grouping (MCG) [7]. For training, the segments are manually classified as one of the predefined distractor classes. Based on the dataset and annotations, they have calculated each segment's features, which contributed to marking a region as a distractor. Based on the dataset and annotations, they have calculated each segment's features, which contributed to marking a region as a distractor. Lastly, they use LASSO [8] to train a mapping from segment features to a distractor score — the average number of distractor annotations per pixel[1]. Although their system is quite promising, there were many areas for improvement in their model. They have created a class for a few fixed objects that appeared majorly as distractors in their dataset, but it can be widened for other distractors in the images with an improved segmentation method and dataset.

2.2 Fully Convolutional Network for Semantic Segmentation

Semantic image segmentation is the process of grouping each pixel in an image from a predefined set of classes. Pixel-wise image segmentation is a well-known problem in computer vision. Most of the traditional methods cannot tell different objects. Long, Shelhamer and Darrel [5] present a system that builds “fully convolutional” networks that produce a correspondingly-sized output with learning and efficient inference from an arbitrary size input. They have adapted networks like AlexNet, VGGNet, and GoogleNet into fully convolutional networks and transferred by fine-tuning their learned representation into a segmentation job. Finally, they were able to define an architecture that combined the appearance information from the shallow, fine layer with semantic information from a deep coarse layer to produce detailed and precise segmentation. The output of this neural network is used to create a hard segmentation using a threshold of 0.6.

2.3 Detecting and Removing Visual Distractors for Video Aesthetic Enhancement

A similar task has also been done for removing visual distractors for video [3]. Fang-Lue Zhang, Xian Wu, Rui-Long Li, Jue Wang, Zhao-Heng Zheng, and Shi-Min Hu have proposed a machine learning-based approach for localizing and detecting visual distractors in the video. They proposed an approach to use a manually labeled dataset to learn a distractor detector. Their work is also similar to the work of Fried at al. [1], where LASSO based learning system was proposed to detect visual distractors. As there is high inter-frame consistency in Video content, so they extracted the motion, Spatio-temporal distribution, and color features by decomposing a video into Temporal-Superpixels(TSPs)[9]. Then using those features, they trained the SVM classifier. As we know, applying image distractor detection on the frame by frame or a few frames does not produce good video distractor detection, given that temporal continuity and motion play a critical role in saliency analysis in video. Experimental results on SVM and end-to-end deep neural network Model SegNet [10] show similar results, and the best F-score is 83.8% on the test dataset. They also demonstrate three different approaches for removing the video's distractors, including video frame replacement, camera path replanning, and video hole filling.

2.4 Image Inpainting for Irregular Holes Using Partial Convolutions

NVIDIA Corporation has proposed a model for image inpainting that has operated robustly on irregular hole patterns and produced semantically meaningful predictions that incorporated smoothly with the rest of the image without needing for any additional post-processing or blending operation [11]. They proposed using a Partial Convolutional Layer to properly handle irregular masks, which comprised a masked and re-normalized convolution operation following a mask-update step. This concept of a masked and re-normalized convolution is also referred to as segmentation-aware convolutions for the image segmentation task; however, they did not modify the input mask. Their use of partial convolutions is such that our convolutional results depend only on the non-hole regions at every layer given a binary mask. Their main extension is the automatic mask update step, which removes any masking where the partial convolution can operate on an unmasked value. Given sufficient layers of subsequent updates, even the most massive masked holes will eventually shrink away, leaving only valid responses in the feature map. The partial convolutional layer ultimately makes our model agnostic to placeholder hole values. After predicting the distractors, we have used this model to remove them and regenerate the image without those regions. This shows the promising result of the small region and fails in a larger area, which is a limitation of our system.

2.5 Visual Distractors Detecting in Images Using TPTSSR

Fardin Sabouri and Farzin Yaghmaee have done a similar work of detecting distractors from the image [4]. In this paper, they first extracted the pixel-based features from different approaches for training and test images, and once the segmentation of the images is done, segment-based features are generated. They manually classify each segment of the training image in a particular class according to the corresponding masks, and then the test image segments are classified according to the TPTSSR method [12] based on sparse coding and representation system in terms of the severity of the distractor in the nine different classes. For the evaluation, they used the MSE metric in the first place. Since the MSE metric leads to unreliable results in conditions of class unbalancing, a new metric has been introduced to evaluate the results. The validation of the results based on the Mean Square Error metric is done which is more valid than the Accuracy and AUC metrics.

3. THE PROPOSED METHOD

The first and essential part of our research was to collect the data on which we would train our model. For this, we used two different datasets. The first one was collected from Amazon Mechanical Turk containing user annotations, which was freely available. However, we discarded the annotations from the dataset and only used the original images from the dataset. The dataset contains various previous saliency literature [13, 14, and 15] datasets of about 1000 images.

We manually checked each image to see if they could be used by us in this project and filtered out those images that do not contain distractor or relevant information regarding our research.

The next dataset we created was collecting random images from google by using a simple web scraping tool. We handpicked images from there, which had some distracting parts in it. We collected this dataset to add a variation on the Mechanical Turk dataset so that our final dataset does not be biased.

We collected all of these images to form a 1700 images dataset containing various images of different types. Now we created a primary drawing tool for the user, which allows them to mark distracting objects/segments in the image. The users were given proper instruction –" For each image in the dataset, mark the area or region which is distracting or disturbing from the main subject in the image, using a marker provided in our tool."We survey this among nearly 100 peoples, and each person was told to mark on 30-40 images. A single image was also shown to multiple users to get a different perspective of distractors by different persons. After the survey, we were able to get 3500 different annotations on the images. On a single image, a lot of different people marked different objects as distractors. Here, we used two different approaches to merge the annotations of different peoples on a single image.

We followed both the approaches parallel for our dataset generation and marking an object or segment in an image as a distractor or not. In the first approach, we only considered those pixels as distractors, marked by everyone as a distractor who filled out the survey, i.e., we took the intersection of all the annotations done by everyone in a single sample image. For the latter approach, we considered all the image pixels as a distractor, which has been marked by anyone in the survey. I.e., we took the mathematical union of all the pixels of all the annotation made by surveyors on a single image. This resulted in two different datasets on which we can train the model and compare our results for better accuracy and prediction of our model.

3.1 Segmentation

We computed pixel-wise features for each image, but that can only apprehend the properties of an isolated pixel or some neighboring pixels around it. Still, we aim to compute the features of a region which form any kind of distractor in the image. So we first segment the image and the cumulative aggregate features across regions. To fulfill this purpose, we use semantic segmentation using a fully convolutional neural network .In particular; our goal is to take an image of size W x H x 3 and generate a W x H matrix containing the predicted class ID's corresponding to all the pixels.

Usually, in an image with various entities, we want to know which pixel belongs to which entity. For example, in the above image, we can segment the person, plant/grass, sidewalk, buildings/structure, etc. To perform the segmentation of the image, a higher-level understanding of the image is required as it has to figure out the objects present in the image and which pixel belongs to which object.

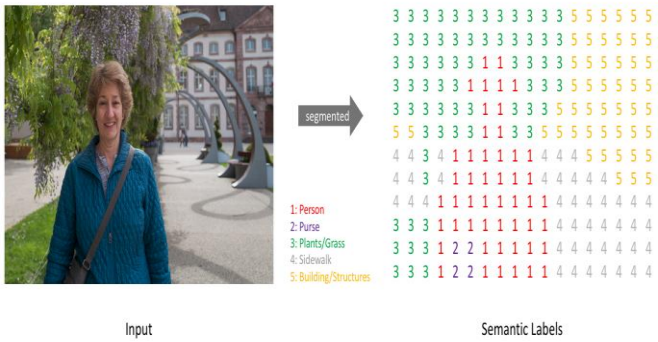

Fig. 1: Image Segmentation

Convolutional networks are powerful visual models that yield hierarchies of features[5]. Thus using CNN for semantic segmentation has been the obvious choice. Naturally, the CNN architecture contains many convolutional layers, batch normalization, pooling layers, and nonlinear activations. The starting layers of this architecture learn low-level concepts such as edges and colors, and higher levels learn concepts such as different objects. Thus the higher levels neurons contain information for a large region of the image. Adding more layers increases the number of channels and decreases the size of the image.

To do image Classification, the mapping of spatial tensor from the convolutional layers to a fixed-length vector is needed, which is achieved using fully connected layers, destroying all the spatial information. To do the semantic segmentation, we need to retain. Hence no fully connected layers are used.

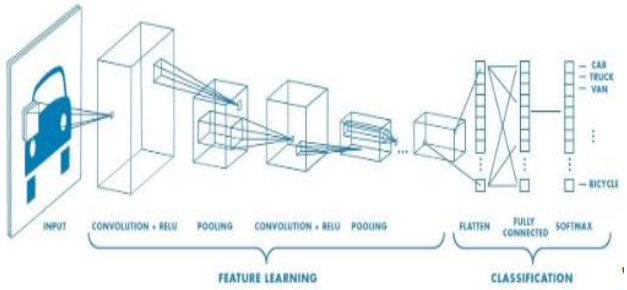

Fig. 2: Spatial tensor is down sampled and converted to a vector

That is why they are called fully convolutional networks. The convolutional layers, coupled with down sampling layers, produce a low-resolution tensor containing the high-level information.

Semantic segmentation is different from object detection as it does not distinguish between different instances of the same object. However, in our scenario, we want the segmentation done so that the same objects are also classified as different segments. For example, a person present in the center of the image might not be a distractor, but a person present in the corner can be a potential distractor for our input image. However, as per the above segmentation's output, all persons present in an image will be segmented as a single entity. To get the required segments, we first calculated all the segments in an image and then treated each connected segment as a single

entity, which means two people present in different regions will be treated as two segments.

3.2. Feature Extraction

From the given images in the dataset we extracted the list of features. Some of them where pixel wise and some of them where segment wise features.

3.2.1. Feature Description

Table 1: Pixel Specific features

S no.	Feature Name	Feature Description
1.	Saliency Detection	Image saliency detection is about identifying the exciting parts of an image, the parts of the image human eyes would fix on. The saliency features depict the visually alluring locations in an image. Here, we create a saliency map of the input image where a pixel is denoted by one if it is salient and 0 if not.
2.	Edge box detection	Edge Boxes is a method to generate object bounding box proposals directly from edges. Edges provide a simplified but informative representation of an image. Line drawing of an image can accurately convey the high-level information in an image using only a small fraction of the information. The use of edges offers many computational advantages since they may be efficiently computed, and the resultant edge maps are sparse. In this research, we investigate how to relate distracting objects from edge-maps.
3.	EdgeBoxesTop20	Along with edge boxes in the image, we also calculate the score of each edge boxes. The score for a box is computed by summing the edge strength of all the edge groups within, minus the strength of edge groups that are part of a contour that straddles the box's boundary. We sort all the boxes accordingly to their respective score, and for this feature, we only consider the top 20 boxes in the image with the highest score and discard others.
4.	EdgeBoxesNotInCenter	We compute all the edges boxes in the images and discard those present in/near the center of the image and are present on the boundary or near the image's boundary. We considered a threshold value of 20%, i.e., boxes which are in the distance within 20% of the radius of the image from the center are discarded, and others are considered here for feature extraction and edge map creation. We did this considering that in most cases, the center of the image contains the salient objects of the image, and the distracting object is present near the edges or far from the center in the image.
5.	Find Horizon	Horizon line detection, also known as segmentation, is a problem of identifying a boundary between sky and non-sky regions(ground, water, or mountains) given a gray scale or color image. For this, we used Hough Line Transform i.e., classifying each pixel as horizon or non-horizon and applying Dynamic Programming on the classification map to extract the horizon line.
6.	Find distance to Center	Here we calculate the Euclidean distance of each pixel from the center of the image. I.e., for each pixel img(i,j) in the image with the center of the image being (x,y), the distance is calculated as : Dis = sqrt(pow(x-i,2)+pow(y-j,2)) This helps to estimate how far a pixel is from the center. The farther the pixel is, the more chance it will be a distractor in the image.
7.	Find distance to edge	Here we calculate the Euclidean distance of the pixel to the closest out of the four image borders, i.e., the distance to the nearest among all the images' sides.
8.	Color Histogram	Histogram of an image presents a graphical representation of the distribution of the pixel's intensities in an image. Histogram of an image gives an intuition regarding the image's properties such as brightness, the tonal range, and the contrast of the image.
9.	Text Feature	Using Optical Character recognition, we extracted the pixels in an image that contains text as sometimes texts in an image can also be considered as distracting. So we used Tesseract to extract the segment in the image which contained text. We marked the pixels that contain text as one and those pixels which contain no texts as 0.
10.	Image Subband Features	We calculate the coefficients of the subbands of the steerable pyramid of the image. A steerable pyramid is a multi-scale linear, multi-orientation image decomposition that presents a useful front-end for image-processing.

Table 2: Segment Specific Features

S no.	Feature Name	Feature Description
1.	Euler number	Number of objects in the region subtracted from the number of holes in those objects and then returned as a scalar. Only 2-D label matrices support this property. The Euler number is computed by regionprops using 8-connectivity(also known as the Euler characteristic).
2.	Filled Area	A number of on pixels in Filled Image, returned as a scalar.
3.	Convex area	A number of pixels in 'ConvexImage', returned as a scalar.

4.	Area	The no of pixels actually present in the region is returned as a scalar. (This value might differ slightly from the value returned by bwarea, which weighs different patterns of pixels differently.)
5.	Eccentricity	Eccentricity of the ellipse that has the same second-moments as the region, returned as a scalar. The eccentricity is the ratio of the distance between the foci of the ellipse and its major axis length. The value is between 0 and 1. (0 and 1 are degenerate cases. An ellipse whose eccentricity is 0 is actually a circle, while an ellipse whose eccentricity is 1 is a line segment.)
6.	Major Axis length	Length (in pixels) of the ellipse's major axis that has the same normalized second central moments as the region returned as a scalar.
7.	Minor Axis Length	Length (in pixels) of the ellipse's minor axis that has the same normalized second central moments as the region, returned as a scalar.
8.	Perimeter	The distance around the boundary of the region returned as a scalar. Regionprops computes the perimeter by calculating the distance between each adjoining pair of pixels around the region's border. If the image contains discontinuous regions, regionprops returns unexpected results.
9.	Solidity	The proportion of the pixels present in the convex hull that are also in the region, returned as a scalar—computed as Area / Convex Area.
10.	Orientation	The angle between the ellipse's major axis and the x-axis that has the same second-moments as the region, returned as a scalar. The value is in degrees, ranging from -90 degrees to 90 degrees.
11.	Equivalent diameter	Diameter of a circle with same area as the region, returned as a scalar. Computed as sqrt(4*Area/pi).
12.	Extent	The ratio of pixels in the total bounding box to the pixels in the region is returned as a scalar. It is computed as the area divided by the area of the bounding box.

3.3. Feature Set generation

For each image in our collected dataset we have segmented the images and we also have the pixel wise and segment wise data. We grouped the pixelwise together of each segments to create a complete feature vector of each segment. For grouping together each pixelwise feature to per-segment features we calculate the mean, median and max for each of the pixel features, resulting three different dataset for each segment of the image. We combined these with the segment specific features. Also for each segment we also determined whether it is a distractor or not in the by the results of the survey which we did with different user. For a segment to be a distractor we considered if the user thinks more than 75% of the pixels in the segment are distractor i.e. they have marked it as a distractor through the survey tool. For others we considered them as a non-distractor. So in the feature vector we added this information along with each segment i.e. it is a distractor or not. So in the end we have six different vector set combination for the whole image dataset. They are:

1. Union min
2. Union median
3. Union max
4. Intersection min
5. Intersection max
6. Intersection median

3.4. Model Learning

Now after the completed dataset generation part, it was the time to train our model using different machine learning techniques and compare our results among them. For each image in our dataset we now have segments and a feature vector per segment. For this we used three different Machine Learning Algorithms. We used LASSO (Least Absolute Shrinkage and Selection operator), SVM (Support Vector Machine) and Random Forest to learn a mapping between segment features and a segment's distractor score (0 or 1). All results in this paper are using all these three mentioned models. Using LASSO allows us to learn the importance of various features and perform feature selection. But in our case, random forests algorithm produced better results.

3.5. Feature Ranking and Optimization

The prepared dataset contained nearly about 8K segments that are labelled as non-distractors and 1k segments as distractors. We shuffled the dataset and chose randomly training and testing feature set. We calculated feature weights from all leave-one-out experiment using LASSO. We then arranged the features according to their feature importance to the resultant model.

Using the feature ranking algorithm we then discarded some of the features from the dataset to make the data more comprehensive and precise. And to avoid overfitting or under fitting of data while training our model. We found out the following features to be redundant in our feature vector. Redundant Features:

1. DisToCenter
2. AttenuatedSaliencyUsingDistance
3. AttenuatedEdgeboxUsingDistance
4. subBand0
5. colorBlue
6. colorProbGreen
7. eulerNumber
8. Solidity
9. Extent

We computed redundant features using different algorithms and all of them presented a different set of redundant features depending on the algorithm and classification they used. Here we took only those features as redundant which was considered redundant by various algorithm.

4. PERFORMANCE EVALUATION AND RESULTS

Experimentations has been performed on a machine having a core i5 processor with a clock speed of 3.4 GHz, 8 GB RAM, and 2 GB graphics card. Python has been used for the complete implementation of the proposed approach. In recent times, python has been very popular for machine learning problems due to the availability of its large amount of libraries, such as Scikitlearn, OpenCV, keras, numpy, and flexibility of language itself. We have used the following Python library to implement our distractor prediction method.

● Scikit-learn: It is used to implement different machine learning algorithms such as SVM, random forest, decision tree.
● OpenCV: It is used to implement different image processing and manipulation techniques.
● Matplotlib: It is used to find the data distribution and plots the results in pictorial forms.
● Pandas: It is used to perform data analysis and for data manipulation.

The results are calculated by a confusion matrix. A confusion matrix is a summary of correctly and incorrectly predicted instances of a class in a classification problem. We have used 70% of dataset as training data, whereas the remaining 30% as testing data. Table 3-8 presents all the results generated from different classifiers on different variations of dataset (mean, median & max of pixels wise features & union and intersection of distracting objects) .Here we can see that the highest precision was offered by Random Forest in every variation of dataset.

Table 3: Median Union Result

Algo.	FPR	Recall	Precision	F1 score	Accuracy
SVM	38.65	--	0	0	61.78
Random Forest	20.64	83.16	61.99	71.03	81.03
Lasso	38.65	--	0	0	61.78

Table 4: Median Intersection Results

Algo.	FPR	Recall	Precision	F1 score	Accuracy
SVM	39.24	--	0	--	62.07
Random Forest	21.66	86.08	59.85	70.60	82.18
Lasso	39.15	--	0	--	62.07

Table 5: Mean Uniėon Results

Algo.	FPR	Recall	Precision	F1 score	Accuracy
SVM	39.51	--	0	--	60.91
Random Forest	26.05	78.77	50.9	61.84	75.71
Lasso	39.51	--	0	--	60.92

Table 6: Mean Intersection Results

Algo.	FPR	Recall	Precision	F1 score	Accuracy
SVM	38.11	--	0	--	63.21
Random Forest	22.31	74.87	57.09	64.78	77.44
Lasso	38.11	--	0	--	63.21

Table 7: Max Union Results

Algo.	FPR	Recall	Precision	F1 score	Accuracy
SVM	38.21	50	0.38	75.42	60.77
Random Forest	21.49	85.95	58.39	69.80	79.17
Lasso	38.30	0	0	--	60.63

Table 8: Max Intersection

Algo.	FPR	Recall	Precision	F1 score	Accuracy
SVM	39.79	--	0	--	60.20
Random Forest	20.80	86.57	62.81	72.80	81.32
Lasso	39.94	20	0.36	70.72	59.77

From the result presented in the table it is clear that Random Forest showed better precision in every variations of dataset among which also it performed the best on the Median Intersection dataset attaining 82.18% accuracy with a recall percentage of 86.08% and F1 score of 70.6 %. We also expected the result to be better in the case of Intersection dataset because the common region was selected as distractor from the various inputs for a single image by different surveyors. For further evaluation we also plotted a receiver operating characteristic (ROC) curve (true positive rate vs. false positive rate) and also calculated the area under the curve (AUC) of the plot.

For performance evaluation, we have to analyze true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), false-negative rate (FNR) and accuracy. We further have calculated precision score, recall score, and f1 score. To gain insight into how precision is varying with recall we have also build the precision recall curve. We have used one more metric i.e. receiver operation characteristics (ROC) curve.

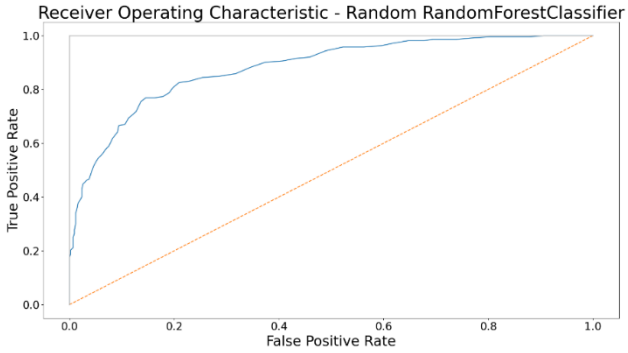

Fig. 3: ROC Curve for Random Forest Classifier

The receiver operating characteristic (ROC) curve is another metric to measure the correctness of a classifier. ROC curve plots the true positive (TPR) rate against the false positive rate (FPR). Figure 3 presents the ROC curve of different classifiers. If any algorithm covers the maximum area under the ROC curve then it is called the best classifier.

In figure 4 recall is plotted along the X-axis and precision is plotted along the Y-axis. The graph depicts that how precision is decreasing when there is an increase in recall score. Figure 4 shows that initially, Random Forest classifier has a precision of 100%, however, it started decreasing after a certain threshold value.

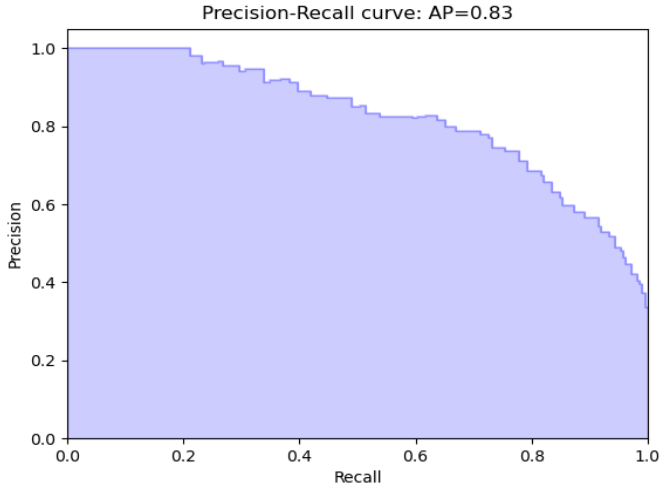

Fig. 4: Precision Recall Curve of Random Forest Classifier

Our proposed features are very accurate and give better accuracy. However, we apply three different algorithms to measure the importance of features. We have used LASSO, K best, and Random Forest, three different feature ranking algorithm. Table 10 represents the feature AUC scores of the different datasets.

Table 10: Different Dataset AUC Scores

Dataset Variation	Total features	Features used	AUC score
Median Intersection	42	35	0.87
Median Union	42	34	0.85
Mean Union	42	33	0.78
Mean Intersection	42	35	0.84
Max Intersection	42	37	0.86
Max Union	42	40	0.84

Total response time is the time between the image inputted and the result outputted by our model after identifying and removing the distractors in the image. During response time, several processes are carried out, such as segmentation of the image, feature extraction for each pixel in the image, segment-wise feature extraction and distribution, preparation of dataset loading the model, and predicting the distracting segments in and removing them from the image. The minimal amount of response time is also very important along with the accuracy. In our analysis, we obtained the response time by testing solution with some random Images. Table 11 described the obtained response time for training and prediction on various models are described below. However, this runtime may very on other systems since runtime is dependent on the configuration of the system.

Table 11: Training and Prediction time of various models

Algorithm	Model Training Time (in sec.)	Model Predication Time (in sec.)
SVM	8.92	0.078
Random Forest	10.26	0.046
LASSO	10.19	0.0156

Fig. 5: Examples of distractor removal results. Each quadruplet shows (from left to right) 1. Original Image 2. Segmented Image 3.Predicated Distractor in the Image

5. CONCLUSION AND FUTURE WORK

In proposed method for detecting the regions with distractors along with overcoming previous challenges, we have prepared a dataset with distracting elements in the images and used it to train our predictor model which predicts the distracting regions and thereafter used image-inpainting to remove those areas which results in a standalone system for distractor removal with no user input.

The main feature of the distractor regions is their saliency so that the visual attention of the viewer (the audience) of the image is attracted to them. But they do not transmit any useful information to the observer and merely distract the observer's senses from the subject and the main message of the image. But when we look closely, the presence of these regions somehow hampers the detection of salient regions because the main supposition of these algorithms is the presence of useful and important information in salient regions. In the proposed method of this paper, the image is first segmented using Convolutional networks for semantic segmentation, which overcomes the challenge of predicting only limited classes of distractors. Then each segment is classified in terms of the score of distractors on the basis of various features, which almost covers all types of distractors in an image. Our main focus was to collect the data which contains all kinds of distractors and then deciding the features which classify an object as a distractor in an image. Detection and removal of distracting regions help to enhance the beauty and visual quality of the image, which can be fulfilled by using our model.

Although our system shows great promise, a few shortcomings can be overcome as the extension of this project. The segmentation part is taking a bit much time, which can be optimized further to improve for faster execution. Image inpainting on our data fails somewhere and does not result in a satisfactory result. As an extension of this project, we can also train and tune the model of image inpainting in order to get better results with our dataset. Further studies of various features are also required, which can contribute to the better prediction of the distractors and cover more distracting elements in the image. One of our major objective is to implement the whole system on Smart Phones which can be a very useful tool for users get a parallel suggestion for their photos in their gallery with modified content.

6. REFERENCES

[1] Fried, Ohad, et al. "Finding distractors in images." Proceedings of the IEEE Conference on Computer Vision and pattern Recognition. 2015.

[2] Shan, Ning, et al. "Photobomb defusal expert: Automatically remove distracting people from photos." IEEE Transactions on Emerging Topics in Computational Intelligence (2018).

[3] Zhang, Fang-Lue, et al. "Detecting and removing visual distractors for video aesthetic enhancement." IEEE Transactions on Multimedia 20.8 (2018): 1987-1999.

[4] Sabouri, Fardin, and Farzin Yaghmaee. "Visual distractors detecting in images using TPTSSR." 2017 10th Iranian Conference on Machine Vision and Image Processing (MVIP). IEEE, 2017.

[5] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[6] Osisanwo, F. Y., et al. "Supervised machine learning algorithms: classification and comparison." International Journal of Computer Trends and Technology (IJCTT) 48.3 (2017): 128-138.

[7] Arbeláez, Pablo, et al. "Multiscale combinatorial grouping." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[8] Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society: Series B (Methodological) 58.1 (1996): 267-288.

[9] Chang, Jason, Donglai Wei, and John W. Fisher. "A video representation using temporal superpixels." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013. Luo, Yiwen, and Xiaoou Tang. "Photo and video quality evaluation: Focusing on the subject." European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2008.

[10] Badrinarayanan, Vijay, Ankur Handa, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling." arXiv preprint arXiv:1505.07293 (2015).

[11] Liu, Guilin, et al. "Image inpainting for irregular holes using partial convolutions." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[12] Xu, Yong, et al. "A two-phase test sample sparse representation method for use with face recognition." IEEE Transactions on circuits and systems for video technology 21.9 (2011): 1255-1262.

[13] Liu, Hantao, and Ingrid Heynderickx. "Studying the added value of visual attention in objective image quality metrics based on eye movement data." 2009 16th IEEE international conference on image processing (ICIP). IEEE, 2009.

[14] Judd, Tilke, et al. "Learning to predict where humans look." 2009 IEEE 12th international conference on computer vision. IEEE, 2009.

[15] Alers, Hani, et al. "Studying the effect of optimizing the image quality in saliency regions at the expense of background content." Image Quality and System Performance VII. Vol. 7529. International Society for Optics and Photonics, 2010.

[16] Yu, Jin-Gang, et al. "A computational model for object-based visual saliency: Spreading attention along gestalt cues." IEEE Transactions on Multimedia 18.2 (2015): 273-286.

[17] Borji, Ali, and Laurent Itti. "State-of-the-art in visual attention modeling." IEEE transactions on pattern analysis and machine intelligence 35.1 (2012): 185-207.

[18] Goferman, Stas, Lihi Zelnik-Manor, and Ayellet Tal. "Context-aware saliency detection." IEEE transactions on pattern analysis and machine intelligence 34.10 (2011): 1915-1926.

[19] Zitnick, C. Lawrence, and Piotr Dollár. "Edge boxes: Locating object proposals from edges." European conference on computer vision. Springer, Cham, 2014.

[20] Luo, Wei, Xiaogang Wang, and Xiaoou Tang. "Content-based photo quality assessment." 2011 International Conference on Computer Vision. IEEE, 2011.

[21] Rahtu, Esa, et al. "Segmenting salient objects from images and videos." European conference on computer vision. Springer, Berlin, Heidelberg, 2010.

[22] Smith, Ray. "An overview of the Tesseract OCR engine." Ninth international conference on document analysis and recognition (ICDAR 2007). Vol. 2. IEEE, 2007.

[23] Neumann, Lukáš, and Jiří Matas. "Real-time scene text localization and recognition." 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012.

[24] Simoncelli, Eero P., and William T. Freeman. "The steerable pyramid: A flexible architecture for multi-scale derivative computation." Proceedings., International Conference on Image Processing. Vol. 3. IEEE, 1995.

[25] Rubinstein, Michael, et al. "A comparative study of image retargeting." ACM SIGGRAPH Asia 2010 papers. 2010. 1-10.

[26] Oliva, Aude, and Antonio Torralba. "Modeling the shape of the scene: A holistic representation of the spatial envelope." International journal of computer vision 42.3 (2001): 145-175.

[27] Murray, Naila, Luca Marchesotti, and Florent Perronnin. "AVA: A large-scale database for aesthetic visual analysis." 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012.

[28] Yan, Jianzhou, et al. "Learning the change for automatic image cropping." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013.

[29] Ahmad, Touqeer, et al. "An edge-less approach to horizon line detection." 2015 IEEE 14th international conference on machine learning and applications (ICMLA). IEEE, 2015.

[30] Newson, Alasdair, et al. "Video inpainting of complex scenes." Siam journal on imaging sciences 7.4 (2014): 1993-2019.

[31] Modha, Dharmendra S., and W. Scott Spangler. "Feature weighting in k-means clustering." Machine learning 52.3 (2003): 217-237.

[32] Menze, Bjoern H., et al. "A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data." BMC bioinformatics 10.1 (2009): 213.

