Advertisement Extraction Using Deep Learning

Boraq Madi(✉), Reem Alaasam(✉), Ahmad Droby(✉), and Jihad El-Sana(✉)

Ben-Gurion University of the Negev, Beersheba, Israel
{borak,rym,drobya}@post.bgu.ac.il, el-sana@cs.bgu.ac.il

Abstract. This paper presents a novel deep learning model for extracting advertisements in images, PTPNet, and multiple loss functions that capture the extracted object's shape. The PTPNet model extracts features using Convolutional Neural Network (CNN), feeds them to a regression model to predict polygon vertices, which are passed to a rendering model to generate a mask corresponding to the predicted polygon. The loss function takes into account the predicted vertices and the generated mask. In addition, this paper presents a new dataset, AD dataset, that includes annotated advertisement images, which could be used for training and testing deep learning models. In our current implementation, we focus on quadrilateral advertisements. We conducted an extensive experimental study to evaluate the performance of common deep learning models in extracting advertisement from images and compare their performance with our proposed model. We show that our model manages to extract advertisements at high accuracy and outperforms other deep learning models.

Keywords: Ads extraction · Loss function · Segmentation model · Regression model

1 Introduction

Advertisements play a significant role in various aspects of our daily life. Commercial organizations advertise their products and services for the general public, and governments utilize the advertisement framework to deliver messages and announcements for educating the public on a wide range of issues. As a result, they occupy none trivial portions of our buildings, streets, and media (printed and digital). Detecting advertisements in a view, an image, or a document has numerous applications. For example, local municipalities are interested in detecting advertisements on the streets for taxing issues, and advertisements agencies want to measure the exposure of their posted advertisements in various media.

Advertisement extraction can be viewed as an image segmentation problem, which is the task of assigning the pixels of each object, in the image, the same label. The segmentation output is usually represented as a pixel map, a graph, or a list of polygons that form the boundaries of these segments. Many recent

E. H. Barney Smith and U. Pal (Eds.): ICDAR 2021 Workshops, LNCS 12917, pp. 81–97, 2021.
https://doi.org/10.1007/978-3-030-86159-9_6

research [1,2,7] focus on predicting polygons instead of explicit pixel-wise labeling, due to their simple representation. In this work, we adopt this scheme and focus on extracting the boundary of objects in a polygon representation.

Object extraction research has made impressive progress over the recent year [14]. However, they are far from obtaining pixel-level accuracy. In this work, we limit our work to convex quadrilateral (Tetragon) objects and obtain high pixel-level accuracy. These objects are very common in our daily views and include advertisements, billboards, frames, paintings, screens, etc. We show that by narrowing the search domain, we can obtain more accurate results than the state-of-the-art general segmentation methods.

This paper explores extracting advertisements from images. It introduces a new dataset (AD dataset), presents a novel deep learning model for advertisement extraction, and develops several geometry-based loss functions for the presented model. Our experimental study shows that our model outperforms other general segmentation methods.

We introduce a new dataset for advertisement extraction, AD dataset. It includes around six thousand images containing various forms of advertisements. The images were collected from the internet and manually annotated by determining the boundary of each advertisement. The label of each advertisement includes a boundary polygon and a mask. In addition, we present and evaluate a novel deep learning model called PTPNet, which is trained using a loss function that considers the polygon and mask labels in the AD dataset. The PTPNet is composed of a regression model and a mask generator model. The regression model extracts CNN features, which are fed to fully connected regression layers to output the vertices of a polygon. These vertices are passed to the generator model to produce a mask, which is used by the loss function. The current PTPNet model outputs the vertices and the mask of a quadrilateral.

The vertex-based loss function is the vertex-wise L_1 distance between the predicted and the ground-truth polygon. The drawback of these loss functions is the assumption that the object's vertices are independent variables. However, these vertices are highly correlated. Considering the area of the polygon reduces the dependency and strengthens the loss function. To overcome this limitation, we introduce a novel loss function that takes into account the polygon and its mask representations. The mask loss function is the Dice distance between the ground-truth mask and the mask generated from the predicted polygon. The final PTPNet loss function combines these two loss functions. According to our experimental study, the loss function contributes to higher accuracy and accelerates the model convergence; these findings align with this work [23].

To summarize, the main contributions of this paper are:

- A new dataset of quadrilateral objects, AD dataset, which includes advertisements images labeled by boundary contours and binary mask.
- Novel loss functions for regression models that take into account the geometric shape and the distance between the vertices of quadrilateral objects.
- A novel regression model, PTPNet, that utilizes boundary contours and binary masks of elements in the AD dataset to learn predicting the boundary polygon of an advertisement.

The rest of the paper is organized as follows: we review the related work in Sect. 2, then describe the new dataset in Sect. 3. In Sect. 4 we present our method and in Sect. 5 we describe the different loss functions. Section 6 presents our experiments and results. Finally, we draw concluding remarks in Sect. 7.

2 Related Work

Extracting the polygon representation of an object can be done by first segmenting the object in the image and extracting its contour, which is often simplified to form a compact polygon. Classic object contour extraction methods are based on superpixel grouping [15], Grabcut [20], and saliency detection [4,22].

One of the popular methods for polygon extraction using deep learning is Polygon-RNN [3] and its improved version [1]. This method provides semi-automatic object annotation using polygons. The RNN's decoder considers only three preceding vertices when predicting a vertex at each step which may produce polygons with self-intersections and overlaps. PolyCNN [8] extracts rectangular building roofs from images. However, this method does not handle perspective transformations and its accuracy does not reach pixel level. PolyMapper [16] presents a more advanced solution by using CNNs and RNNs with convolutional long-short term memory modules. It provides good results for aerial images of residential buildings.

In recent years, many deep learning methods and architectures have been proposed for semantic segmentation and they lead to outstanding progress in semantic segmentation. The most two relative groups of methods for our work are Regional proposal and Upsampling/Deconvolution models.

Regional proposal models detect regions according to similarity metrics, determine whether an object is present in the region or not, and applying segmentation methods for positive regions. He *et al.* [9] proposed a Mask Regional Convolutional Neural Network (Mask-RCNN). It extends Faster R-CNN [19] abilities for object detection and segmentation.

The deconvolution models focus on extracting high-level features via layer-to-layer propagation and obtain segmentation by upsampling and deconvolution. The reconstruction techniques for obtaining a segmentation map include refinement methods that fuse low and high-level features of convolutional layers. Long *et al.* [18] proposed the first Fully Convolutional Network (FCN), which constructs the segmentation by adding skip architecture that combines the semantic and appearance for precise segmentation results.

Many other segmentation models [14] are used to provide benchmark results for new datasets, Such as Xception [5], Resenet [10], MobileNet [11] and DenseNet [12]. These models are regression models which output polygon, while Mask-RCNN [9] provides an object's mask. In this paper, we apply these models to test various loss functions and compare their performance with our proposed model.

3 Dataset

This section presents a new dataset of advertisement images called AD dataset[1]. It includes about six thousand RGB-images where each image contains at least one quadrilateral advertisement. The AD dataset includes a wide range of advertisements types and sizes, starting from billboards on the highway to advertisements in malls. Figure 1 shows example images from the dataset.

(a) (b)

Fig. 1. Example of images in AD dataset.

3.1 Collecting and Labeling Images

The images of the AD dataset were collected automatically from the internet and labeled manually. We collected images that include advertisements using a python script called 'Google images download'[2] which downloads images from google using given keywords. The resolution of the images varies from 256×144 to 2048×1080. The advertisements within these images have diverse perspective views.

We label the images using Labelbox[3], the annotators went over each of the downloaded images and traced the boundary of each advertisement in the image. We define the boundary of an advertisement by its vertices in a counter-clockwise (CCW) order, as shown in Fig. 2b. The labeling task was conducted by several undergraduate and graduate students in our lab. In addition, we generate a binary mask (See Fig. 2c) using the annotated polygon for each image, where the pixels of the advertisement are marked by 1 and the background by 0. Hence, we have two types of labels for each image.

[1] https://github.com/BorakMadi/Ads-Extraction-using-Deep-learning.
[2] https://github.com/hardikvasa/google-images-download.
[3] https://labelbox.com/.

(a) (b) (c)

Fig. 2. Element example of AD dataset: (a) the original image, (b) the vector of vertexes label, and (c) the binary mask label

4 The PTPNet Model

Yu *et al.* [23] introduced Intersection-Over-Union (IOU) loss function to regress four vertices of a box. This loss function considers the four vertices as a whole unit, i.e. the bounding box. Their trained model manages to obtain higher accuracy than those using L_2 (Euclidean) distance loss function. Their IOU loss function assumes axis aligned bounding boxes, which are calculated directly from the predicted and ground-truth boxes. This calculation scheme of the IOU does not suit our task, as the perspective view of the quadrilateral is not expected to be axis aligned. In addition, the scalar difference between the size of the predicted and the ground-truth polygons is an inappropriate approximation for the IOU loss.

To overcome these limitations, we constructed a novel learning model called *PTPNet*, which enables applying advanced geometrical loss functions to optimize the prediction of the vertices of a polygon. The PTPNet outputs a polygon and its mask representation and manages to combine classical regression and advanced geometrical loss functions, such as IOU.

4.1 PTPNet Architecture

PTPNet network includes two sub-networks: Regressor and Renderer. The Regressor predicts the four vertices of a quadrilateral. It utilizes the Xception model as its backbone. The classification component of the Xception model is removed and replaced by a regression component that outputs a vector of eight scalars that represent the four vertices. The regression component includes Global Average Pooling layers followed by 4-Fully-Connected layers with different sizes, as shown in Fig. 9.

The Renderer (rendering model) generates a binary mask corresponding to the Regressor's predicted polygon. It is trained separately from the regression model using the quadrilaterals' contours from the AD dataset (see Render Datasets). The trained model is concatenated to the regression model and its weights are frozen during the training of the regression model (Fig. 3).

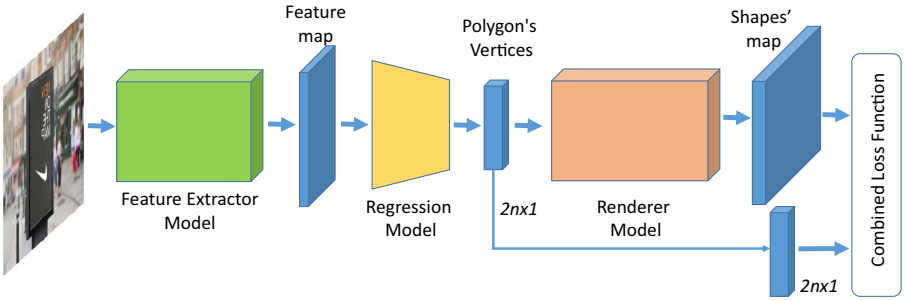
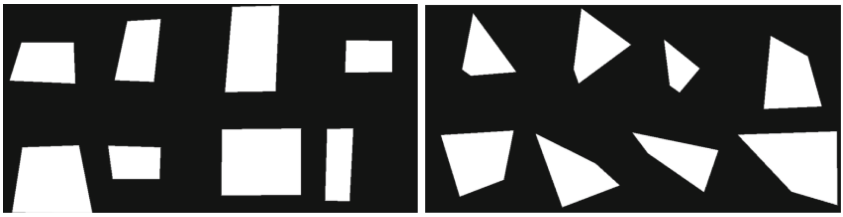


Fig. 3. The architecture of PTPNet

4.2 Rendering Models

We experimented with various rendering decoder networks, which are based on fully connected and deconvolution networks. The results obtained using the fully connected decoders were inadequate, while the deconvolutional decoders produce promising results (see Sect. 6). Therefore, we developed a novel rendering model based on the Progressive Growing of GANs [13], which is a form of deconvolutional decoders. We removed the discriminator network and modified the generator network (the deconvolutional decoder) to act as a supervised based learning decoder that accepts a polygon representation and outputs its corresponding mask. We shall refer to this decoder as *GenPTP*.

To train and test the rendering models we build two datasets, *synthesized-quads* and *ads-quads*. The synthesized-quads dataset was constructed by randomly sampling convex quadrilaterals, i.e. sampling four vertices. In typical images, the convex quadrilaterals are the perspective transformation of rectangular advertisements. Figure 4(b) shows an example of such quadrilaterals. The ads-quads dataset was generated by considering only the contours of the advertisements in the AD-Dataset (the manually annotated quadrilaterals), as shown in Fig. 4(a). The labels of these contours are the binary masks of the corresponding advertisement, which are part of the AD-Dataset. We evaluated the performance of the Rendering models using the two datasets, i.e. synthesized-quads and ads-quads. We trained two different instances of each rendering model, one for each dataset and compare their performances. For faithful comparison, we evaluate the performance of these instances using test sets from the ads-quads dataset.



(a) ads-quads (b) synthesized-quads

Fig. 4. Sample from ads-quads and synthesized-quads.

5 Loss Functions

In this section, we present several loss functions for optimizing our models. These loss functions are categorized into vertex-based regression functions, areas-based loss functions, and hybrid methods.

5.1 Vertex-Based Loss Functions

These loss functions compute the sum of the distances between the corresponding vertices of predicted and ground-truth polygons. To determine the corresponding among the vertices, we compute all the possible circular shifts of vertices of the predicted polygon and choose the shift with minimum distance with respect to the ground truth vertices. The distance between vertices is calculated using L_1 and L_2 metric. In this approach, the model learns to regress to polygon vertices independent of their order, similar to [8]. Equation 1 and Eq. 2 formulate the loss functions with respect to L_1 and L_2, receptively, where n is the number of vertices, V_{gt} and V_{pred} represent vertices of the ground truth and predicted polygons, and R is the circular shift function applied to the V_p with step r. Since we limit our domain to quadrilaterals, n is equal to four.

$$MinR_{L_2} = \min_{\forall r \in [0,3]} \frac{1}{n} \sum_{i=1}^{n} \|V_g - R(V_p, 2 * r)\|_2 \tag{1}$$

$$MinR_{L_1} = \min_{\forall r \in [0,3]} \frac{1}{n} \sum_{i=1}^{n} \|V_g - R(V_p, 2 * r)\|_1 \tag{2}$$

5.2 Area Loss Functions

Area based loss functions, such as the difference between the area of prediction and ground-truth polygons, are insufficient. These functions motivate the model to learn to regress to four vertices that have the same area as the ground-truth, but without considering the location of these vertices, which is incorrect. We propose an alternative approach that considers the locations of the vertices

and treats them as representatives of geometric shapes rather than independent vertices, similar to IOU in [23].

Theoretically, calculating the similarity between two quadrilateral can be performed using advanced geometrical loss functions such as IOU. However, calculating such similarity based only on polygon representation is complicated. Thus, we approximate the difference between the intersection and the union between the two polygons using two trapezoids. Among possible shifts of the predicted vertices, we choose the permutation with minimum distance with respect to the ground truth according to L_1 criterion (see Eq. 3). Let us denote this permutation, by r_{min}. We calculate the two trapezoid areas by utilizing four vertices, two from the r_{min} permutation and two from the ground truth. The final loss value is the sum of these two trapezoid areas (see Eq. 4 and Fig. 5c).

$$r_{min} = \min_{\forall r \in [0, \frac{n}{2}-1]} \frac{1}{n} \sum_{i=1}^{n} \|V_g - R(V_p, 2 * r)\|_1 \tag{3}$$

$$Trapezoids_{loss} = TrapezoidArea(p1, p2, g1, g2) + TrapezoidArea(p3, p4, g3, g4) \tag{4}$$

5.3 Hybrid Loss Functions

The vertex-based and area-based approaches suffer from vertex-independent optimization and overlook the importance of boundary vertices, respectively. Since the vertices of a quadrilateral are correlated, the independent optimization can not provide the best results. The distance between vertices does not describe the difference in the induced areas faithfully. For example, the ground-truth (See Fig. 6a) and predicted (See Fig. 6b) share the same three vertices and differ in one (See Fig. 6c). As seen, the loss is the green region (See Fig. 6d),

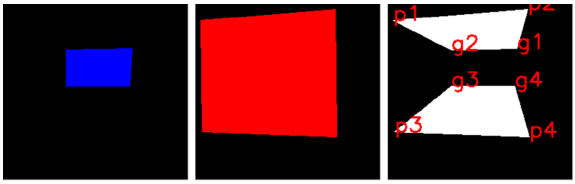


Fig. 5. The trapezoids between the two polygons

the loss based only on the vertices is small and gives insufficient feedback. The area-based loss functions provide uniform weights for all quadrilateral points and neglect the importance of vertices and edges.

To overcome these limitations, we combine independent vertex regression and geometrical loss functions to take into account the appropriate weight of the vertices and the correlation between them. Practically, we define the loss function as a sum of the quadrilateral area and vertex-distance loss functions.

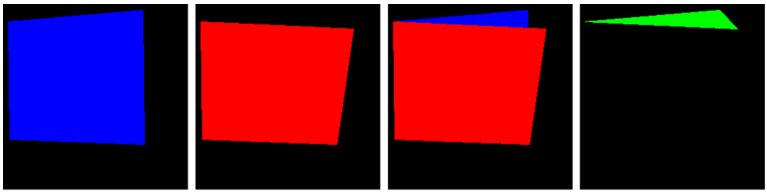


Fig. 6. Intersection and loss regions

Vertex and area difference based loss functions combine the distance between the corresponding vertices using L_1 or L_2 and the difference between the size of the quadrilaterals, i.e. subtracting the two scalars. The size difference is not an adequate optimization parameter in our case (see the discussion earlier). Nevertheless, it improves performance when combined with vertex distance loss components, i.e. this combination guides the loss to choose the nearest points with the same area. Practically, we combine Eq. 1 or Eq. 2 with area difference. The distance between ground-truth and predicted vertices is performed using Eq. 1 or Eq. 2. We calculate the area based on Shoelace or Gauss's area formula. The area difference is computed by subtracting the areas of polygons spanned by the ground truth V_{gt} and the predicted V_{psh} vertices. This combination is formally expressed by Eq. 5 and Eq. 6

$$Area\_MinR_{L_2} = MinR_{L2} + AreaSub(V_{psh}, V_{gt}) \tag{5}$$

$$Area\_MinR_{L_1} = MinR_{L_1} + AreaSub(V_{psh}, V_{gt}) \tag{6}$$

5.4 Loss Function for PTPNet

Above we have shown how to integrate vertex-based loss function with simple geometrical loss, such as areas. The geometrical element in previous hybrid-loss functions is limited to scalar representation. This limitation prohibits using sophisticated loss functions that accurately describe the geometrical relation between the two quadrilateral shapes. The PTPNet model overcomes this by integrating a rendering component, which generates a binary mask that resembles the quadrilateral corresponding to the predicted vertices.

The PTPNet loss function, Eq. 7, combines vertex-regression, Eq. 2, with Sørensen–Dice coefficient (Dice) loss function. The Dice is applied to the generated and ground-truth mask.

$$PTPNet\_loss = MinR_{L_1} + (1 - Dice) \tag{7}$$

6 Experiments

In this section, we overview data preparation, models construction, and evaluation results.

6.1 Dataset Preparation

In Sect. 3 we discussed collecting and annotating the dataset. Next, we overview preparing the train and test dataset, the augmentation process, and the sanity-check dataset.

The train and test sets are selected from the AD-dataset (see Sect. 3). In this study, we focus on images that include one quadrilateral advertisement and subdivide them to 70% and 30% for training and testing, respectively. The train and test images are resized to 256×256. The annotations are modified according to the resized images. Recall that the annotation of each advertisement includes its boundary contours and a binary mask. The contour (polygon) is represented by a normalized vector.

We augment the training dataset by applying the basic geometric transformation, i.e. rotation, scale, and sheer. We apply random rotations between 0° to 90°, scale between 0.8 to 1.2, and shear from 0° to 20°. The same transformation is applied to the image and its labels, i.e. contour and mask.

We build a sanity-check dataset for the initial evaluation of our regression methods. This dataset is generated from the contours of advertisements within the images of the AD-dataset, i.e. a contour (quadrilateral polygon) is embedded within a black image creating a binary image with a quadrilateral polygon (see Fig. 7) (Fig. 8).

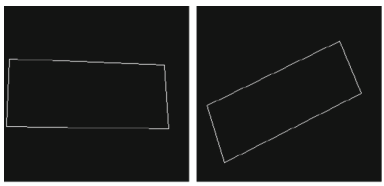


Fig. 7. Samples from sanity-check dataset.

Fig. 8. Examples of augmented images

6.2 Study Models

Toward our evaluation study, we experimented with various types of deep learning architectures, which we discuss next.

We explored applying Mask-RCNN [9] to detect and localize advertisements within images and outputting a binary mask. The MaskRCNN was pre-trained on COCO-dataset [17]. We define the performance of the pre-trained MaskRCNN as the baseline for our experiments.

Our regression networks accept a color image that contains quadrilateral advertisement and predict the coordinates of its four vertices. These neural networks consist of a features extractor and a regression model. The model regresses to four vertices (vector) using the latent vector of the feature extractor, similar to PolyCNN [8], but with a different truncation style and model architecture.

We choose a set of *study models* that includes 13 network architecture, which are variations[4] of Xception [5], Resenet [10], MobileNet [11] and DenseNet [12]. We refer to these 13 models as the *study models*, which are pre-trained on ImageNet [6] and truncated to act as feature extractors. To build a regression model, we append to each feature extractor a component composed of Global Average Pooling layers followed by 4-Fully-Connected layers with different sizes as shown in Fig. 9.

We trained the study models on $10k$ synthesized images using the Mean Square Error (MSE) loss function. As shown in Table 1 the top five models are the modified (replacing the fully connected component with a regression model)

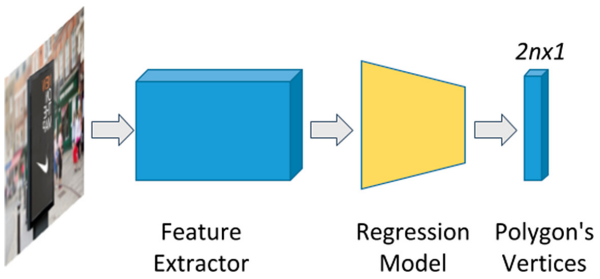


Fig. 9. The modified regression models network

[4] The full list appears in Table 1.

versions of Xception [5], ResNet50 [10], ResNet101 [10], DensNet121 [12] and MobileNetV2 [21]. Therefore, we adopted these models to study the influence of the various loss functions, i.e. we explore the performance of these models using the loss functions we discussed in Sect. 5.

Table 1. The accuracy of the regression models on the sanity-check dataset

Regression model	Loss function	Accuracy	Regression model	Loss function	Accuracy
Xception	MSE	**0.961**	InceptionV3	MSE	0.937
InceptionResNetV2	MSE	0.927	VGG16	MSE	0.947
ResNet50	MSE	**0.965**	ResNet50V2	MSE	0.949
ResNet101	MSE	**0.964**	ResNet101V2	MSE	0.951
ResNet152V2	MSE	0.951	MobileNet	MSE	0.936
MobileNetV2	MSE	**0.964**	DenseNet121	MSE	**0.955**
DenseNet201	MSE	0.943			

In this experiment we compare the performance of the modified regression models using Eq. 1 and Eq. 2 as mentioned in Sect. 5. The training of the models utilizes the polygon label only.

Table 2. The regression models results using $MinR_{L_2}$ and $MinR_{L_1}$ loss functions on AD dataset.

Regression model	Loss function	Accuracy	Regression model	Loss function	Accuracy
Xception	$MinR_{L_1}$	0.843	Xception	$MinR_{L_2}$	0.822
ResNet50	$MinR_{L_1}$	0.814	ResNet50	$MinR_{L_2}$	0.775
ResNet101	$MinR_{L_1}$	0.837	ResNet101	$MinR_{L_2}$	0.751
MobileNetV2	$MinR_{L_1}$	0.839	MobileNetV2	$MinR_{L_2}$	0.808
DenseNet121	$MinR_{L_1}$	0.850	DenseNet121	$MinR_{L_2}$	0.78

The results are shown in Table 2. As seen, Eq. 2 gives better results than Eq. 1 for the five models. It outperforms Eq. 1 by at least 2.5% for all the models. However, these two loss functions consider the four vertices independently, i.e. do not take into account the correlation among the vertices. To overcome this limitation, we integrate area difference in the loss functions.

The area difference, d_{area}, measures the size difference between the area of the ground truth and predicated quadrilaterals. We add d_{area} to the $MinR_{L_2}$ and $MinR_{L_1}$ and get hybrid loss functions that consider the vertices independently and takes into account the area of the polygon they define. The loss functions we use are $Area\_MinR_{L_2}$ and $Area\_MinR_{L_1}$.

Table 3. The performance of regression models using $Area\_MinR_{L_2}$ and $Area\_MinR_{L_1}$

Regression model	Loss function	Accuracy	Regression model	Loss function	Accuracy
Xception	$Area\_MinR_{L_1}$	0.863	Xception	$Area\_MinR_{L_2}$	0.8303
ResNet50	$Area\_MinR_{L_1}$	0.825	ResNet50	$Area\_MinR_{L_2}$	0.816
ResNet101	$Area\_MinR_{L_1}$	0.840	ResNet101	$Area\_MinR_{L_2}$	0.818
MobileNetV2	$Area\_MinR_{L_1}$	0.8366	MobileNetV2	$Area\_MinR_{L_2}$	0.801
DenseNet121	$Area\_MinR_{L_1}$	0.8578	DenseNet121	$Area\_MinR_{L_2}$	0.8177

As shown in Table 3 adding the area difference to $MinR_{L_2}$ and $MinR_{L_1}$ improves the performance of all the models by 2% on average, expect MobileNetV2. The accuracy of MobileNetV2 deteriorates by less than 0.05%. We believe this is due to the lack of training data, as our dataset is not big enough.

$Area\_MinR_{L_2}$ and $Area\_MinR_{L_1}$ loss functions consider the vertices and the area separately. They capture the correlation between the vertices as a geometric shape; i.e. the predicted vertices aim at producing a shape, which area is equal to that of the ground truth.

The $Trapezoids_{loss}$ loss function aims at capturing the geometric correlation between the predicted vertices and the shape they form without separating the two, as we explained in Sect. 5. As shown in Table 4 the trapezoid loss function gives better results than $MinR_{L_2}$ for all the study models. In addition, it gives better results than $MinR_{L_1}$ for both Xception and MobileNetV2 models. This indicates its usability for extracting objects in a similar way to $MinR_{L_2}$ and $MinR_{L_1}$ with a focus on the object as a whole.

The $Trapezoids_{loss}$ loss function considers the shape as one entity, vertices and area together, thus restricting the independent movement of the vertices. To evaluate the role of $MinR_{L_1}$ in the independent movement of the vertices and the overall performance, we added the $MinR_{L_1}$ to the $Trapezoids_{loss}$ loss function. We refer to this loss function, as $Trapezoids\_MinR_{L_1}$.

As shown in Table 5, using $Trapezoids\_MinR_{L_1}$ outperform $Trapezoids_{loss}$ for all the study models, expect ResNet101. The accuracy of the remaining study

Table 4. The performance of the regression models using $Trapezoids_{loss}$

Regression model	Loss function	Accuracy
Xception	$Trapezoids_{loss}$	0.8671
ResNet50	$Trapezoids_{loss}$	0.7949
ResNet101	$Trapezoids_{loss}$	0.7923
MobileNetV2	$Trapezoids_{loss}$	0.8461
DenseNet121	$Trapezoids_{loss}$	0.8068

Table 5. The performance of the regression models using $Trapezoids\_MinR_{L_1}$

Regression model	Loss function	Accuracy
Xception	$Trapezoids\_MinR_{L_1}$	0.8674
ResNet50	$Trapezoids\_MinR_{L_1}$	0.8349
ResNet101	$Trapezoids\_MinR_{L_1}$	0.7852
MobileNetV2	$Trapezoids\_MinR_{L_1}$	0.840
DenseNet121	$Trapezoids\_MinR_{L_1}$	0.820

models increased by an average of 2%, which indicates that combing the vertices with the geometric shape loss function improves performance.

6.3 PTPNet

We refer to the rendering model composed of Fully-Connected (FC) layers as n-FCGenNet, where n refers to the number of FC layers in the network. In this experiment we compare the performance of n-FCGenNet and GenPTP (see Sect. 4). We experiment with two n-FCGenNet instance models: 3-FCGenNet and 6-FCGenNet. The two models input a vertex vector and output a pixel vector, which represents a 256×256 mask. We choose the Dice coefficient loss function to train the rendering models, in which accuracy is measured using IOU and DICE.

Table 6 summarized the comparison of the three rendering models. As seen, the GenPTP outperforms the 3-FCGenNet and 6-FCGenNet models. The GenPTP trained using ads-quads outperforms the same model trained on the synthesized-quads dataset by %12 and %7 using IOU and DICE metrics, respectively. We train GenPTP separately and combine it with the Regressor.

Table 6. The performance of different rendering models

Render model	Dataset	IOU	DICE
GenPTP	ads-quads	91.18	95.33
GenPTP	synthesized-quads	79.64	88.21
3-FCGenNet	ads-quads	59.73	73.3
3-FCGenNet	synthesized-quads	49.23	63.38
6-FCGenNet	ads-quads	75.59	85.95
6-FCGenNet	synthesized-quads	65.4	78.15

Table 7. Comparing the accuracy of PTPNet and MaskRCNN

Model	Loss function	Accuracy
PTPNet	$PTPNet\_loss$	0.851
MaskRCNN	$MaskRCNNLossfunction$	0.810

We decided to compare the performances of PTPNet with MaskRCNN. As shown in Table 7, PTPNet outperforms MaskRCNN by 4%. The performance of PTPNet is similar to the top regression model, which is the Xception architecture with $Trapezoids\_MinR_{L_1}$ loss function. However, PTPNet is easier to generalize for any polygon since its loss function does not assume a prior geometric shape. In addition, having the mask and the polygon in the training phase enables handling more complex tasks.

7 Conclusion

In this paper, we presented various deep learning models with different loss functions for advertisement extraction. We introduce AD dataset, which is a dataset of quadrilateral advertisements. Modified versions of several regression models are explored and their performances are studied using various loss functions. We use L_2 and L_1 loss functions and add the area difference to improve performance. We introduce the trapezoid loss function that considers the vertices as representatives of a shape instead of focusing on the predicted vertices independently and show that adding L_1 loss to trapezoid loss gives the best results in most of the modified regression models. In addition, we introduce the PTPNet model with its own loss function that combines the results of a rendering model and a regression model. We conduct an extensive experimental study to evaluate the performance of common deep learning models in extracting advertisements from images and compare their performance with our proposed model. We show that our proposed model manages to extract advertisements at high accuracy and outperforms common deep learning models. The scope of future work includes extending our approach to handle general polygons.

Acknowledgment. This research was supported in part by Frankel Center for Computer Science at Ben-Gurion University of the Negev. One of the authors, Reem Alaasam, is a fellow of the Ariane de Rothschild Women Doctoral Program, and would like to thank them for their support.

References

1. Acuna, D., Ling, H., Kar, A., Fidler, S.: Efficient interactive annotation of segmentation datasets with polygon-RNN++. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 859–868 (2018)

2. Bauchet, J.P., Lafarge, F.: KIPPI: kinetic polygonal partitioning of images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3146–3154 (2018)
3. Castrejon, L., Kundu, K., Urtasun, R., Fidler, S.: Annotating object instances with a polygon-RNN. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5230–5238 (2017)
4. Cheng, M.M., Mitra, N.J., Huang, X., Torr, P.H., Hu, S.M.: Global contrast based salient region detection. IEEE Trans. Pattern Anal. Mach. Intell. **37**(3), 569–582 (2014)
5. Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258 (2017)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)
7. Duan, L., Lafarge, F.: Image partitioning into convex polygons. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3119–3127 (2015)
8. Girard, N., Tarabalka, Y.: End-to-end learning of polygons for remote sensing image classification. In: IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium, pp. 2083–2086. IEEE (2018)
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969 (2017)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
11. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
12. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)
13. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
14. Lateef, F., Ruichek, Y.: Survey on semantic segmentation using deep learning techniques. Neurocomputing **338**, 321–348 (2019)
15. Levinshtein, A., Sminchisescu, C., Dickinson, S.: Optimal contour closure by superpixel grouping. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6312, pp. 480–493. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15552-9_35
16. Li, Z., Wegner, J.D., Lucchi, A.: Topological map extraction from overhead images. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1715–1724 (2019)
17. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
18. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. CoRR abs/1411.4038 (2014). http://arxiv.org/abs/1411.4038
19. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR abs/1506.01497 (2015). http://arxiv.org/abs/1506.01497

20. Rother, C., Kolmogorov, V., Blake, A.: "GrabCut" interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. (TOG) **23**(3), 309–314 (2004)
21. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
22. Wang, L., Wang, L., Lu, H., Zhang, P., Ruan, X.: Saliency detection with recurrent fully convolutional networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 825–841. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_50
23. Yu, J., Jiang, Y., Wang, Z., Cao, Z., Huang, T.: UnitBox: an advanced object detection network. In: Proceedings of the 24th ACM International Conference on Multimedia, pp. 516–520 (2016)