Performance Comparison of Deep Learning Techniques for Recognizing Birds in Aerial Images

Yang Liu[1], Peng Sun[1], Max R. Highsmith[1], Nickolas M. Wergeles[1], Joel Sartwell[2], Andy Raedeke[2], Mary Mitchell[3], Heath Hagy[3], Andrew D. Gilbert[4], Brian Lubinski[3], and Yi Shang[1]
[1]*Department of Electrical Engineering and Computer Science (EECS), University of Missouri, Columbia, MO, USA*
[2]*Missouri Department of Conservation, Columbia, MO, USA* , [3]*U.S. Fish & Wildlife Service, Bloomington, MN, USA*
[4]*Illinois Natural History Survey, Forbes Biological Station, University of Illinois, Champaign, IL, USA*
{ylt5b, ps793, mrh8x5}@mail.missouri.edu, {Joel.Sartwell, Andrew.Raedeke}@mdc.mo.gov,
{mary_mitchell, heath_hagy, Brian_Lubinski}@fws.gov, agilb849@illinois.edu, {shangy, wergelesn}@missouri.edu

Abstract— **In computer vision, significant advances have been made in recent years on object recognition and detection with the rapid development of deep learning, especially deep convolutional neural networks (CNN). The majority of deep learning methods for object detection have been developed for large objects and their performances on small-object detection are not very good. This paper contributes to research in low-resolution small-object detection by evaluating the performances of leading deep learning methods for object detection using a common dataset, which is a new dataset for bird detection, called Little Birds in Aerial Imagery (LBAI), created from real-life aerial imagery data. LBAI contains birds with sizes ranging from 10px to 40px. In our experiments, some of the best deep learning architectures were implemented and applied to LBAI, which include object detection techniques such as YOLOv2, SSH, and Tiny Face, in addition to small instance segmentation techniques including U-Net and Mask R-CNN. Among the object detection methods, experimental results demonstrated that SSH performed the best for easy cases, whereas Tiny Face performed the best for hard cases, i.e. where a cluttered background makes detecting birds difficult. Among small instance segmentation methods, experimental results revealed U-Net achieved slightly better performance than Mask R-CNN.**

Keywords—small-object detection, instance segmentation, convolutional neural networks, deep learning, aerial image dataset

I. INTRODUCTION

Object detection is one of the crucial tasks in computer vision. In the past few years, the performance of object detection [1-14] has dramatically improved due to the success of deep convolutional neural networks (CNN). Typically, object detection and recognition involve two steps: first, deep neural networks are used to localize the potential location of each target object; then, objects are classified into appropriate classes. If the first step can effectively localize the potential object, the second step will be easier. Even though the two-step approach achieved state-of-the-art performance, the running times are usually slow [11]. Therefore, one-stage detectors have been developed to improve the speed.

Small-object detection remains challenging because small objects usually have lower resolution and less context information. Finding a 20 × 20 size object located in a 5000 × 5000 image is a difficult task, even for humans. As described in the literature, state-of-the-art methods for object detection usually performed poorly on small objects [11]. Recent research has shown the importance of context information and scale for small-object recognition [8][9]. In addition, it has been reported that lower-layer features extracted from CNNs are very useful for small-object detection and segmentation [8-11].

The work presented in this paper focuses on low-resolution small-object detection by evaluating the performances of leading deep learning methods using a common dataset, which is a new dataset for bird detection, called Little Birds in Aerial Imagery (LBAI). This dataset was created from real-life aerial imagery data, provided by the Illinois Natural History Survey at the University of Illinois at Urbana-Champaign. LBAI contains images of waterfowl and other water birds in shallow lakes within the Illinois River Valley. LBAI includes different colors, shapes, poses, resolutions, and bird sizes range from 10px to 40px. The dataset contains different backgrounds of rivers, vegetation, land, and mixtures between each type of background. Overall, LBAI captures the diversity of real-life situations for bird detection in shallow lake and wet lands across the Midwest. Some of the birds have larger sizes, in higher resolutions and homogenous backgrounds, which make them easier to be identified. While others have smaller sizes, in lower resolutions with blurry contours, making them hard to be detected. LBAI is designed to identify the difficulties and improve existing methods on small object detection.

Using the LBAI dataset, we compared a wide-range of representative state-of-the-art deep learning methods. The results shed a light on the strengths and weaknesses of different deep neural network architectures for small object detection. The contributions of this research include applying and adapting leading deep-learning methods to the LBAI dataset, evaluating performances of these methods on a common benchmark dataset for small-object detection and segmentation, and automating the time-consuming process of manual image processing from waterfowl surveys.

The rest of the paper is organized as follows. Section II introduces deep learning methods on object detection, with an emphasis on the techniques related to small-size object detection. Section III presents the state-of-the-art object-detection methods applied to LBAI for performance evaluation. Section IV describes the properties of LBAI in detail. Section V

978-1-5386-4210-8/18/$31.00 ©2018 IEEE
DOI 10.1109/DSC.2018.00052

presents experimental results. Last, Section VI summarizes this research.

II. Related Work

There are two major approaches for object detection and recognition. The first detection-based approach is the traditional one that generates a bounding box of the detected objects and then identifies the type of objects. The second approach, which is a segmentation-based approach, can also be used for object detection. This approach first generates labelling at the pixel level and then tries to identify the class of the objects to which each pixel belongs.

A. Object detection methods

Existing deep learning algorithms for object detection falls into two categories: one-stage detectors and two-stage detectors [1-9]. At first, two-stage detectors generate many region proposals, which may potentially contain the objects. Then, these sparse proposals are further classified into different object categories. In general, two-stage detectors are more accurate, but slow when compared to one-stage detectors. In one-stage detectors, the bounding boxes proposal step is eliminated and both, object localization and classification, are done in one pass. This strategy significantly improves the speed of detection when compared with two-stage detectors.

In a two-stage detector, the regions that potentially contain objects are first proposed. Then, detection refinement is applied to classify proposed regions and regress the bounding box location. For example, the Selective Search method [1] is used in R-CNN [2] to generate category-independent region proposals to localize the regions that may contain the target objects. R-CNN then uses a convolution neural network to refine regions. Each region proposal is fed into the CNN independently, which is a slow process. Fast R-CNN [3] addressed these issues by only computing the convolutional feature map once. Therefore, each region proposal shares the computation of the same feature map. The region proposals are generated in a Region of Interest (RoI) pooling format to feed into fully connected layers [3].

Faster R-CNN [4] further improved the detection speed by using a fully convolutional network, called Region Proposal Networks (RPNs), to generate the region proposals, replacing the Selective Search method used in previous methods. In the second stage, a CNN is used for proposal refinement and object classification. The main benefit of this design is that the RPN shares the same convolutional layers with the object detection network, which reduces the detection time [4]. Furthermore, FPN was proposed to improve Faster R-CNN [12]. FPN used the concept of a feature pyramid. Instead of applying a pyramid on the input images, it used the feature map pyramid, since CNN already provide a hierarchy between the different feature layers. The idea was implemented in a bottom-up and top-down path with lateral connections. FPN utilized a lower, high-resolution feature layer, compared to other algorithms, which dramatically improve detection accuracy, especially for small objects.

In one-stage detectors, the proposal generation stage is removed, resulting with localization and classification being performed in one stage. Recently, YOLO [5][6], SSD [7], and their variations achieved promising results. YOLO [5] divided input images into 7×7 cells and each cell predicts two bounding boxes. The network has convolutional layers followed by fully connected layers. Even though YOLO achieved a 45 frame per second detection speed, which is extremely fast when compared to other algorithms, the main drawbacks result from localization prediction errors and low object detection recall [5].

DSSD [10] is a variation of SSD [7]. It improved the performance of SSD, especially for small objects, by using a larger network as well as adding additional context information with de-convolutional neural networks. DSSD achieved higher accuracy, especially for small objects. Recently, RetinaNet [13] achieved state-of-the-art results for one-stage detection. It outperformed existing two-stage detectors while maintaining a fast detection time. The work in [13], found that the accuracy gap between one-stage detectors and two-stage detectors was mainly due to the positive and negative examples being highly unbalanced, since there are extremely large amounts of background examples overwhelming the process. Even though each loss of the background examples is small, the large number of the background cases result in dominating the total loss, which results in a degenerated model. This problem was solved by introducing a new loss function, called focal loss, to change the weights between positive examples and negative examples, so they cannot affect the loss function dramatically. Huang et.al. in [11] compared the detection accuracy and detection time between two-stage detectors and one-stage detectors. They concluded that, on average, one-stage detectors are faster than two-stage detectors, while two-stage detectors tend to be more accurate than one-stage detectors. The performance of most detection algorithms dropped dramatically when applied to small-object detection. In addition, several one-stage detectors were developed for small face detection, including Tiny Face [8] and SSH [9].

B. Instance segmentation methods

FCN [15] is one of the first methods that use CNNs in the semantic segmentation area. FCN employs CNNs without fully connected layers, which allows the input image to have an arbitrary size. This method laid the foundation for later methods.

A key issue of segmentation methods is the pooling layers. Adding pooling layers can reduce the computation time and increase the reception field size. U-Net is based on FCN [15], with the encoder-decoder architecture to address the issue of determining the appropriate numbers of pooling layers. It has a U-shape architecture to balance the trade-off between good localization accuracy and efficient context information. Therefore, it only needs a small number of training images. In the encoder stage, it uses pooling layers to gradually reduce the layer size, whereas, in the decoder stage, it uses up-convolution to gradually increase the layer size. Moreover, U-Net uses the short-cut connection from encoder to decoder to help the decoder recover fine-grain information. Regarding the trade-off between reception field and localization accuracy, large reception fields lead to lower localization accuracy. On the other hand, when the reception field is too small, the localization accuracy may also decrease due to the lack of context information.

Mask R-CNN [14] is a recent work based on Faster R-CNN and FCN. Faster R-CNN already provides two predictions:

bounding box localization and recognition. Mask R-CNN added the third output on top of Faster R-CNN, which is the instance mask prediction for segmentation. The Mask R-CNN architecture can output bounding box localization, classification, and segmentation at the same time. The improvement of Mask R-CNN from FCN comes from the new ROIAlign layer, multitask training, and a better backbone network [14] [15].

III. Deep Learning Methods Applied to LBAI

In this section, the representative state-of-the-art object-detection methods are presented. In our experiments, they were applied to the LBAI dataset to evaluate their performances on low-resolution small-object detection.

A. Object detection methods

(1) Single Shot MultiBox Detector (SSD) [7] is a one-stage detector that performs object localization and classification in a single forward pass of its CNN. SSD's network is built on the VGG-16 architecture, with the fully connected layer removed. Instead of using a fully connected layer, several small convolutional feature maps are added on top of VGG-16 to predict the target objects. Moreover, to capture different object scales, SSD generates different scales of feature maps for detection. This will result with two predictions being generated, one predicts the bounding box category and the other predicts the location of the bounding box. At the end, non-maximum suppression (NMS) is used to generate the final detection results. SSD achieved good accuracy, comparable to two-stage detectors, but much faster. However, SSD's performance on smaller objects was much worse. The reason is that small objects may not appear on higher-level feature maps. Even though increasing the input image size can help slightly, SSD cannot address the problem well.

In our experiments on the new LBAI dataset, we used the source code of SSD built on the Caffe [17] framework with a VGG-16 architecture as the backbone network. VGG-16 is pretrained on ImageNet [21] for image classification and fine-tuned on our LBAI dataset. We used the same data augmentation and hard negative mining as SSD: batch size set to 16, input image size of 512 × 512, and a really small learning rate of 1e-7. The results were poor and thus not report in the experimental results section.

(2) Single Stage Headless (SSH) [9] is one of the variations of SSD, specifically designed for face detection. Like SSD, SSH is also a single-stage detector and the architecture of SSH is built on a VGG-16 network without fully connected layers. The convolutional detection module is applied on top of three different feature maps, corresponding to different strides, in order to handle scales of variation. The detection module consists of several convolutional layers and eventually generates two results, classification prediction and bounding box localization. Unlike SSD, SSH focuses on adding more context information, especially for small objects. A context module is designed inside of the detection module to increase the receptive field size. With VGG-16 as a base-net, SSH outperformed the ResNet-101 based methods. SSH achieved state-of-the-art results on both the WIDER FACE dataset [18] and the FDDB dataset [19]. Moreover, the inference time is fast. SSH achieved 21 FPS with an image resolution of 400 × 800 and 13 FPS with a resolution of 600 × 1000, which is much faster than Tiny Face [8].

In our experiments on the LBAI dataset, we used the source code of SSH built on the Caffe [17] framework with a VGG-16 architecture as the backbone network. VGG-16 is pretrained on ImageNet [21], for the task of image classification, and fine-tuned on our LBAI dataset. We changed the number of output classes to two, which is bird or not bird (i.e. background in our case). The network was trained with the batch size equal to 128 and the input size equal to 512 × 512.We set the learning rate to 0.0004 for easy cases and to 0.004 for hard cases. The momentum is equal to 0.9 and the weight decay was set to 0.0005. It took approximately two hours to train 10,000 iterations on a GTX 980M graphics processing unit (GPU) with 8GB memory. Both classification loss and regression loss decreased during training.

(3) You Only Look Once (YOLO) v2 [5] is an improved version of the original YOLO network with several adjustments. Batch normalization on the convolutional layers is used to stabilize network training. The performance is increased by approximately 2% mAP with batch normalization. As found by other research, higher resolution can capture more information, especially for small objects [5]. This strategy increases the performance by approximately 4% mAP. In YOLOv2, the fully connected layers are removed. Instead of directly predicting the location of bounding boxes, YOLOv2 adopted an anchor box strategy similar to that used by Faster R-CNN. This can improve the recall by a large margin while only slightly lowering precision. A dimension clustering algorithm is used to find the starting anchor box dimensions based on the data from the training set. With dimension clustering and direct location prediction, the location accuracy is improved by over 4%. Finally, for improving performance on small objects, the lower feature map is concatenated with the higher feature map.

In our experiments on LBAI, we used the source code for YOLOv2 built on the Darknet framework. We loaded weights from darknet53.conv.74 and fine-tuned them on the LBAI dataset for 16,000 batches. We changed the number of output classes to one and adjusted the last convolutional filter to 30. The network was trained with a batch size of 64 and subdivisions set to 8. We applied a jitter of .4 to the training set and used a resolution of 448 × 448. We set the learning rate to 0.0001 with a decay of 0.0005.

(4) Tiny Face [8] is one of the face detectors specifically designed to detect small objects. In [8], experiments were conducted to investigate the importance of scale, image resolution, and contextual information. Separate detectors were trained over different feature layers of a single CNN to tackle the scale variant problem. The work in [8] claimed that context is the key to finding small objects, especially for low-resolution faces, and having a large reception field is good to detect small faces. More context was introduced by combing feature maps from multiple CNN layers. In addition, a two-times larger resolution was found to be effective to find smaller faces. The work showed that Tiny Face achieved state-of-the-art results on both WIDER FACE and FDDB datasets. More importantly, it achieved 82% mAP accuracy for small faces, whereas previous methods only achieved 29-64% mAP. However, the image

pyramid pipeline slows down the method. The running time was 1.4 FPS on 1080 resolution images and 3.1 FPS on 720 resolution images, which is much slower than other one-stage detectors.

In our experiments on LBAI, the source code of Tiny Face was built on the Matconvent framework written in Matlab. The original input size is 500 × 500. After we changed the input size to 512 × 512, the algorithm randomly cropped a 500 × 500 image region. We used a batch size of 12 and a learning rate equal to 1e-5 for easy cases and a learning rate equal to 1e-4 for hard cases. The momentum was set to 0.9 and the weight decay set to 0.0005. We trained Tiny Face with ResNet-101 as the base architecture for 50 epochs, which took approximately 5 hours.

B. Instance segmentation methods

(1) U-Net [15] is built on fully convolutional networks, specifically designed for biomedical image segmentation. In the contracting path, the convolutional layers are applied with pooling layers to extract context features. In the expanding path, the up-sampling layers are added to increase the localization accuracy. More importantly, the feature maps from the contracting path are concatenated with the up-sampling layers to improve localization. In addition, elastic deformations are applied as data augmentations during training. U-Net is the winner of the ISBI challenge for segmentation and the ISBI cell tracking challenge in 2015. With a 512 × 512 input image, the inference time is less than one second.

In our experiments, the basic U-Net architecture was used to train on the LBAI dataset. However, because of the significant difference between the natural images in LBAI and bio-cell images, we added zero-padding after each convolution operation block, instead of cropping the reception field as in Isola [24] and Zhu's work [25]. This prevented the network from losing too much pixel label information, which was needed because objects in LBAI are very small. With padding, the U-Net architecture would have the same size of input and output.

In order to apply the segmentation method for object detection, instance segmentation labels were prepared as the ground truth. However, it is time and labor consuming to generate segmentations for every target object in LBAI. So, instead of using object contours as labels, we used a 20 × 20 square as a ground truth mask, centered at the coordinate of each object. After fixing the network architecture, specifically the inputs and outputs, we fed 512 × 512 images into U-Net and trained the network. In the training phase, we used the VGG-16 pretrained weights on ImageNet [21] as initial weights in all encoder blocks and the Xavier initializer in all decoder blocks. The learning rate was set to 0.001 with a learning rate decay equal to 0.1 for every 7 epochs. The batch size was set to 2 in the training phase and since we were using a GTX 980M GPU with 8GB memory, the Adam optimizer was used. In the inference phase, blob detection on the final output was used to calculate the coordinates after running through the segmentation network.

(2) Mask R-CNN [14] is a recent work for segmentation and object detection, as explained in the related work section. The major change in Mask R-CNN is that it solves the ROI pooling problem that causes the feature maps and original image not to be aligned. Mask R-CNN uses the ROIAlign layer to replace the ROI pooling layer. In the ROIAlign layer, the rounding for boundaries or bins are removed and bilinear interpolation is applied to compute the exact values for the feature maps. Moreover, Mask R-CNN uses a binary cross-entropy loss for the instance mask, which avoids the competition among all classes. Mask R-CNN achieved state-of-the-art results of instance segmentation on the COCO [22] test dataset with a running time of 5 FPS.

When implementing Mask R-CNN on the LBAI dataset, we used the same input and output described in the U-Net implementation. In the training phase, we froze the weights of ResNet-101 and trained all the other weights in the original Mask R-CNN architecture. For the hyper-parameters, we used the Adam optimizer with the value of 0.0001 as the learning rate until the loss curve converged. Other implementation details for the training and inference phases were the same as U-Net for a direct comparison between these two methods, e.g. batch sizes and blob detection.

IV. LBAI DATASET

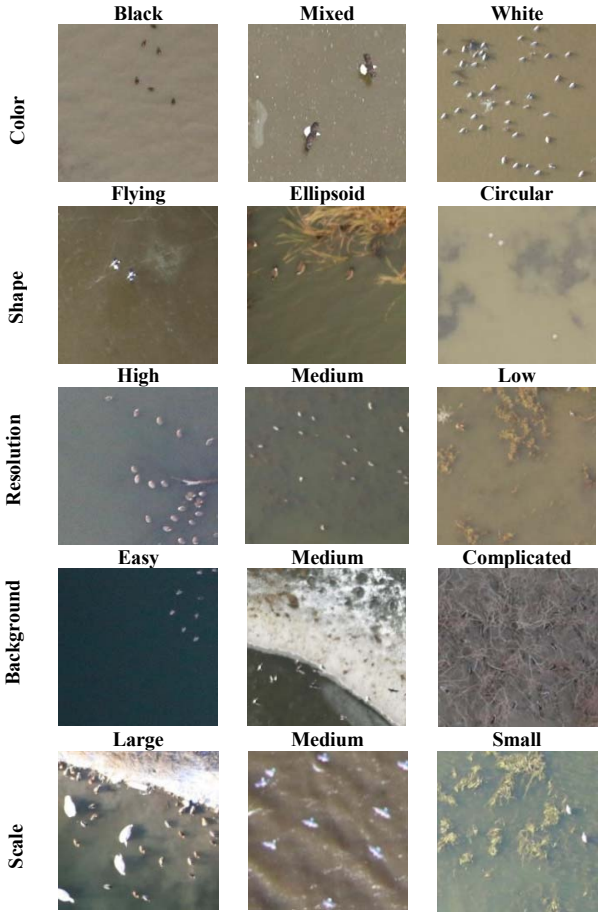

Fig. 1. Examples of the new LBAI dataset for small object detection and instance segmentation. Cropped images with different color, shape, resolution, background, and scale are shown.

A. Dataset overview

The LBAI dataset was provided by the Illinois Natural History Survey at the University of Illinois at Urbana-Champaign. The total dataset has 230GB of data, with 440 high-resolution images that have a resolution of 5760 × 3840, and an altitude value of approximately 90 meters above ground level (AGL). Fig. 1 shows examples of the cropped images, with different color, shape, resolution, background, and scale. Due to the large size of images, it is difficult to train CNNs directly on the original images. From the original dataset, we created two sub-datasets, LBAI-A and LBAI-B, with different cropping scenarios as stated below.

LBAI-A: There were 336 images with high-resolution that were used, and this dataset was divided into the training, validation, and test sets based on these images. For each set, we take the original image and crop it into the small patches with a size of 512 × 512, without overlapping the patches. This will insure that the original image does not get put into different sets (e.g. a patch from the original image gets put into the training set and another patch from the same original image gets put into the validation set). The incomplete boundary regions were discarded after cropping, since resizing may change the ratio and shape of the birds. For the training set, there are a total of 3,158 cropped images with 24,836 birds. We only keep the small patches with birds in the training and validation set. However, the test set contains all the cropped images, both with birds and without birds. After applying the various object-detection methods on the cropped images to detect the birds, the detection results from the patches were merged back into the original images. In our experimental results on this dataset, the performance comparisons of the various methods are based on the merged patches which form the original images.

LBAI-B: We use a total of 336 images in dataset-B as well. We used the same strategy as for dataset-A, i.e., no patch from the same original image will be used across the training, validation, and testing set. The original images were cropped into small patches of size 512×512 without overlap. Incomplete boundary images were not included. In the training, validation, and test set, only the cropped images which contain birds are used. In our experimental results on this dataset, performance comparisons of the various methods are based on the cropped image set, which only contains birds.

B. Dataset labelling

When we received this dataset, it contained the bird counting labels, i.e. the number of birds per image, from the Illinois Natural History Survey at the University of Illinois at Urbana-Champaign. However, it did not contain the bounding box locations for the birds, which is the labelling needed for detection. We generated the annotations for the birds' location, so that the number of birds would match the total number of birds received from the expert annotations. A labelling tool, called Sloth, was used to label the images. For each image, a dot was put at the center of each visible bird for all of the birds. This dot label was used for blob detection to generate the bounding box and pixel level labels. Next, we used image processing techniques to find the contour of the labeled birds. A bounding box was drawn around the bird's contour to generate bounding box labels. All labeled results are saved in an xml file. These labels were created from multiple observers with varying levels of training and experience.

C. Dataset seperation based on difficulty levels

The backgrounds of the LBAI images are very different, which have a significant impact on the bird detection results. Some images have clear backgrounds with uniform colors, which usually correspond to rivers and water. In this case, the main problem is to identify the birds among different colors, shapes, and resolution situations. On the other hand, in the images with backgrounds of land, trees, or vegetation, detection of birds is much harder, even for humans with great eyes. It is hard to distinguish emergent vegetation and submersed aquatic vegetation from birds. Therefore, following ideas from other datasets [16], we split each dataset into easy and hard cases based on the background. In LBAI-A, 3,158 images are categorized as easy cases, which contributed 52% of our labeled data, and 2,907 images as hard cases. In LBAI-B, there are 2,416 easy case images and 2,056 hard case images. The proportions of easy and hard cases are 54% and 46%, respectively.

D. Image diversity

As shown in Fig. 1, LBAI contains images of birds of various colors, shapes, poses, resolutions, scales, and backgrounds. The properties are discussed as follows.

Color. For different species, birds have different features. Some birds are all white, while others are all gray or blackish. Some of the birds have darker stripes on their back, while others may have darker dots around their heads. Different color features and weather conditions add more variety to the dataset.

Shape and Pose. Birds have different shapes in our dataset. Most of the birds have ellipsoidal shapes, i.e., the head and tail are narrow, and the middle of the body is wide. However, this is not true for all cases. Some of the birds are partially covered by the vegetation area, which leads to circular shapes. Different activities of the birds also led to different shapes and poses. For example, most birds are swimming or floating, whereas others are flying.

Resolution. LBAI contains images taken under different image resolutions. In some images, the bird's shape, color and angle are clear to see. On the other hand, some birds are blurry and hard to distinguish in both color and shape. This is due to the variation of airplane altitude, weather conditions, and image resolution from the camera when the images were taken.

Scale. LBAI contains birds ranging from the size of 10px to 40px, due to the different species and activities. Most of the birds are in the range of 10px to 20px. Some flying birds or birds of bigger species (i.e., ducks and geese) have larger sizes of 30px to 40px.

Background. LBAI covers diverse background information, including calm water, water waves, vegetation region, forest, and many others. Background clutter is one of the major issues for correctly identifying birds.

V. Experiment Results

In this research, we evaluate the detection results and the counting results of various deep learning methods on a common

dataset, the LBAI dataset. For detection, the performance metrics include precision, recall, and F1 score. Precision is the percentage of correctly predicted instances over the total number of predictions, while recall is the percentage of correctly predicted instances over the total amount of instances, defined as follows:

$$Precision = \frac{tp}{tp + fp} \quad (1)$$

$$Recall = \frac{tp}{tp + fn} \quad (2)$$

where tp is true positive, fp is false positive, and fn is the false negative instances.

F1 is the harmonic mean of precision and recall:

$$F1 = 2 * \frac{precision * recall}{precision + recall} \quad (3)$$

For counting results, the performance metric is the mean absolute error (MAE), i.e., the difference between the predicted count of birds in an image and the true count based off the labels described in the previous section.

A. Performance comparion using LBAI-A

Using LBAI-A, we compared the performance of five representative state-of-the-art deep learning methods, including YOLOv2, SSH, Tiny Face, Mask R-CNN, and U-Net.

TABLE I. PERFORMANCES OF DIFFERENT DEEP LEARNING TECHNIQUES ON THE **EASY CASES** IN THE **LBAI-A** DATASET.

Methods	Precision	Recall	F1 Score	MAE
YOLOv2	0.831	0.745	0.785	49.3
SSH	0.801	0.836	0.818	46.4
Tiny Face	0.613	0.809	0.697	103.2
Mask R-CNN	0.772	**0.842**	0.805	49.0
U-Net	**0.861**	0.781	**0.819**	**38.5**

TABLE II. PERFORMANCES OF DIFFERENT DEEP LEARNING TECHNIQUES ON THE **HARD CASES** IN THE **LBAI-A** DATASET.

Methods	Precision	Recall	F1 Score	MAE
YOLOv2	**0.568**	0.238	0.335	22.0
SSH	0.46	0.62	0.53	20.3
Tiny Face	0.552	0.542	**0.547**	21.2
Mask R-CNN	0.193	**0.659**	0.299	89.5
U-Net	0.55	0.51	0.53	**20.1**

Fig. 2 shows the examples of bird detection results of the five deep learning methods on the three different types of LBAI images. The green bounding boxes are the prediction results and the red bounding boxes are the ground truth. In Image 1, the background is uniform, with calm water surfaces, which is an easy case. Looking at Image 2, which is still an easy case, there are vegetation areas, making it harder to identify the birds. Image 3 represents an example of land as the background, although there are only four birds in the water, a significant number of false positives were generated by all methods in the land regions, which is why this is a hard case.

The results on the test cases are shown in Table I and II. There is a total of 944 test images for the easy cases and 944 images for the hard cases in LBAI-A. As shown in Table I, on the easy cases in LBAI-A, U-Net and SSH obtained the highest F1 score, 81.9% and 81.8%, respectively, which were much higher than the other three methods. In terms of precision, U-Net was the best, while YOLOv2 was second. In terms of recall, Mask R-CNN was the best, while SSH was second. In terms of MAE, U-Net was much better than the other methods, outperforming them by at least 17%.

As shown in Table II, focusing on the hard cases in LBAI-A, the precision, recall, and F1 scores of all methods were much worse than their corresponding results on the easy cases in Table I. The best F1 score on the hard cases in LBAI-A was 54.7%, generated by Tiny Face. U-Net and SSH had results similar to Tiny Face. However, YOLOv2 and Mask R-CNN had significantly lower F1 scores. Mask R-CNN had the highest recall rate, but lowest precision because it generated too many false positives. In terms of MAE metrics, U-Net and SSH performed the best, slightly better than Tiny Face and YOLOv2. Mask R-CNN performed very poorly.

B. Performance comparion using LBAI-B

TABLE III. PERFORMANCES OF DIFFERENT DEEP LEARNING TECHNIQUES ON THE **EASY CASES** IN THE **LBAI-B** DATASET.

Methods	Precision	Recall	F1 Score
YOLOv2	0.886	0.878	0.882
SSH	0.909	0.941	**0.925**
Tiny Face	**0.936**	0.879	0.907
Mask R-CNN	0.845	**0.956**	0.902
U-Net	0.921	0.897	0.909

TABLE IV. PERFORMANCES OF DIFFERENT DEEP LEARNING TECHNIQUES ON THE **HARD CASES** IN THE **LBAI-B** DATASET.

Methods	Precision	Recall	F1 Score
YOLOv2	0.742	0.698	0.720
SSH	0.766	**0.810**	0.787
Tiny Face	**0.903**	0.722	**0.802**
Mask R-CNN	0.656	0.721	0.687
U-Net	0.867	0.592	0.704

Using LBAI-B, we compared the performance of the same five deep learning methods on the cropped images. The results on the test cases are shown in Table III and IV. There are 202 and 93 testing images for easy and hard cases, respectively. The number of testing images are much smaller than LBAI-A

because we only included the patches containing birds. The patches that did not contain birds were discarded in LBAI-B. Therefore, in our results, we didn't calculate MAE for LBAI-B because MAE is only calculated for the original sized images. However, with only patches containing birds, we would not be able to reconstruct the original images.

As shown in Table III, focusing on the easy cases in LBAI-B, SSH obtained the highest F1 score of 92.5%. This is significantly higher than the other methods. Whereas, YOLOv2 was the worst out of all the methods. In terms of precision, Tiny Face was the best, while YOLOv2 was the worst. In terms of recall, Mask R-CNN was the best, while SSH was a close second.

As shown in Table IV, focusing on the hard cases, the precision, recall, and F1 scores of all methods were again much worse than their corresponding results on the easy cases in Table III, resulting in a greater than 10% difference. The best F1 score on the hard cases was 80.2%, generated by Tiny Face. SSH was a close second. SSH had the highest recall rate, whereas Tiny Face had the highest precision.

C. *Running time comparision*

The execution times of the methods were measured as frames per second (FPS) to predict the results. All of the FPS were generated on a GTX 980M GPU with 8GB memory. Given the input size of 512 × 512, SSH was the fastest, reaching 14 FPS. The running time for U-Net and Mask R-CNN are both 5 FPS. TinyFace runs the slowest with 1.4 FPS, due to its image pyramid multi-scale architecture.

VI. SUMMARY

In this paper, we have presented a new aerial imagery dataset based on real-life images including waterfowl and other water birds in wetlands around the Midwest. Different from most of the existing datasets, the new LBAI dataset contains small birds of sizes ranging from 10px to 40px. Several state-of-the-art deep learning object detection and instance segmentation techniques have been applied to the LBAI database and obtained a range of performance results. Among object detection methods, SSH performed the best on the easy cases, while Tiny Face achieved the best accuracy on the hard cases. Between instance segmentation methods, U-Net achieved better performance than Mask R-CNN. These results are useful for identifying the strengths and weaknesses of existing methods and the development of future methods with improved performance.

REFERENCES

[1] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," In *IJCV*, 2013.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," In *CVPR*, 2014.

[3] R. Girshick, "Fast r-cnn," In *ICCV*, 2015.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards realtime object detection with region proposal networks," In *NIPS*, 2015.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," In *CVPR*, 2016.

[6] J. Redmon and A. Farhadi. "YOLO9000: Better, faster, stronger," In *CVPR*, 2017.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," In *ECCV*, 2016.

[8] Hu, Peiyun and Ramanan, Deva, "Finding Tiny Faces," In *CVPR*, 2017

[9] Najibi, Mahyar and Samangouei, Pouya and Chellappa, Rama and Davis, Larry, "SSH: Single Stage Headless Face Detector," In *ICCV* 2017.

[10] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. "DSSD: Deconvolutional single shot detector," arXiv:1701.06659, 2016.

[11] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama et al., "Speed/accuracy trade-offs for modern convolutional object detectors," In *CVPR*, 2017.

[12] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection," In *CVPR*, 2017.

[13] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. "Focal loss for dense object detection," arXiv preprint arXiv:1708.02002, 2017.

[14] .K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," arXiv:1703.06870, 2017.

[15] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation," In *MICCAI*, pages 234–241. Springer, 2015.

[16] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," In *CVPR*, 2015.

[17] C. Zitnick and P. Dollar, " Edge boxes: Locating object proposals from edges," In *ECCV*, 2014.

[18] S. Yang, P. Luo, C.-C. Loy, and X. Tang. "Wider face: A face detection benchmark," In *ICCV*, June 2016.

[19] V. Jain and E. Learned-Miller. "Fddb: A benchmark for face detection in unconstrained settings," Technical Report UMCS-2010-009, University of Massachusetts, Amherst, 2010.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," Proceedings of the 22nd *ACM international conference on Multimedia*, 675-678, 2014.

[21] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition," In *ICLR*, 2015.

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. "Microsoft COCO: Common objects in context," In *ECCV*. 2014.

[23] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes (VOC) Challenge," *IJCV*, pages 303–338, 2010.

[24] Isola, P., Zhu, J.Y., Zhou, T. and Efros, A.A., 2017. "Image-to-image translation with conditional adversarial networks," arXiv preprint.

[25] Zhu, J.Y., Park, T., Isola, P. and Efros, A.A., 2017. "Unpaired image-to-image translation using cycle-consistent adversarial networks," arXiv preprint arXiv:1703.10593.

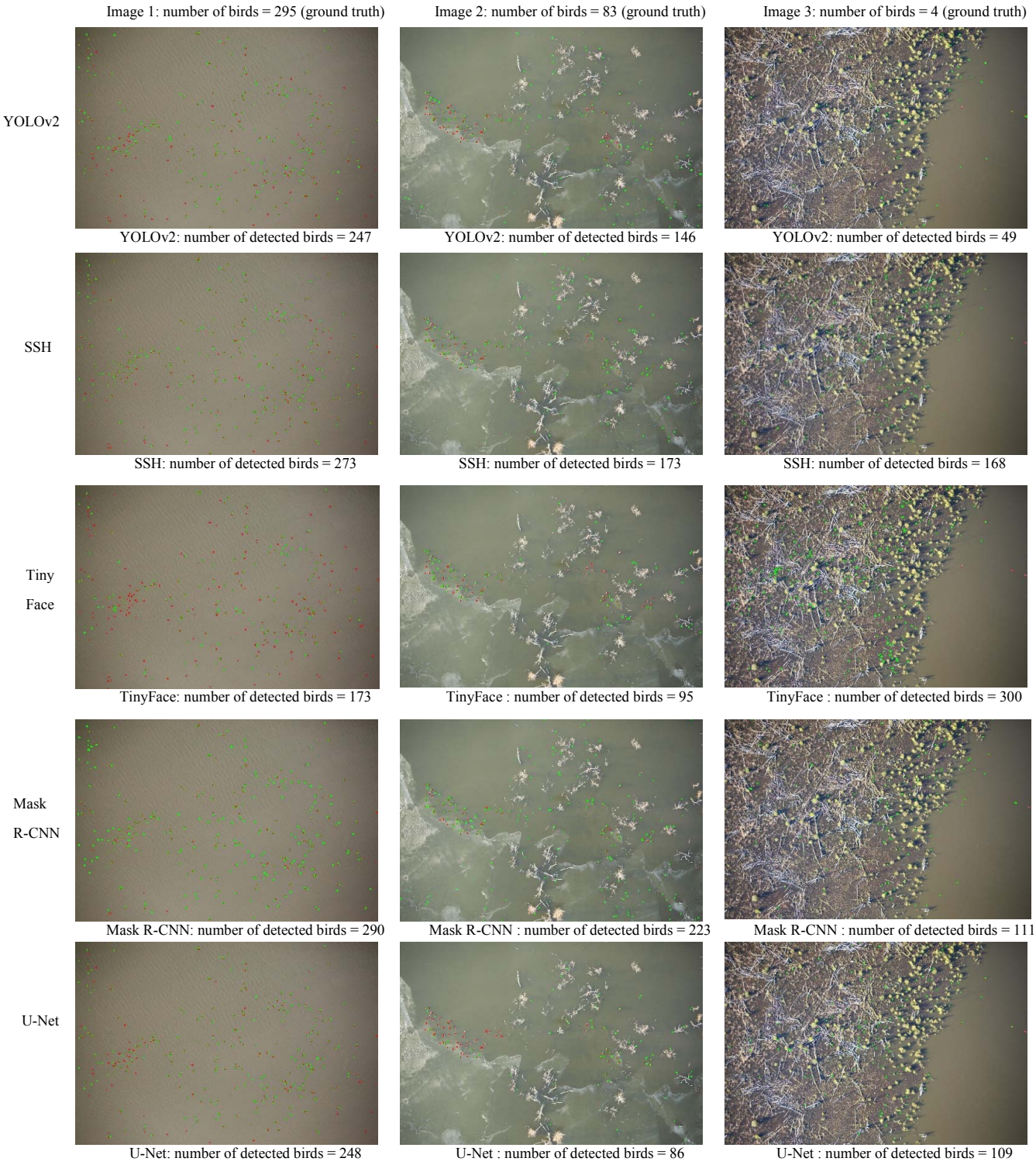

Fig. 2. Examples of bird detection results of the 5 deep learning methods using 3 different types of LBAI images. The green bounding boxes are the prediction results. The red bounding boxes are the ground truth. In Image 1, the background is uniform, calm water surfaces. In Image 2, there are ice in the bottom half of the image and vegetation scattered in the image, making it harder to identify the birds. In Image 3, as an example of land background, although there are only 4 birds in the water, a lot of false positives are generated by all the methods in the land regions.