OBBStacking: An Ensemble Method for Remote Sensing Object Detection
Haoning Lin, Changhao Sun, and Yunpeng Liu

Abstract—Ensemble methods are a reliable way to combine several models to achieve superior performance. However, research on the application of ensemble methods in the remote sensing object detection scenario is mostly overlooked. Two problems arise. First, one unique characteristic of remote sensing object detection is the oriented bounding boxes (OBB) of the objects and the fusion of multiple OBBs requires further research attention. Second, the widely used deep learning object detectors provide a score for each detected object as an indicator of confidence, but how to use these indicators effectively in an ensemble method remains a problem. Trying to address these problems, this article proposes OBBStacking, an ensemble method that is compatible with OBBs and combines the detection results in a learned fashion. This ensemble method helps take first place in the Challenge Track Fine-Grained Object Recognition in High-Resolution Optical Images, which was featured in 2021 Gaofen Challenge on Automated High-Resolution Earth Observation Image Interpretation. The experiments on DOTA dataset and FAIR1M dataset demonstrate the improved performance of OBBStacking and the features of OBBStacking are analyzed.

Index Terms—Ensemble, object detection, oriented bounding box, remote sensing, stacking.

I. INTRODUCTION

With deep learning, researchers can design arbitrarily structured models as they see fit for a specific problem, which in turn leads to a wide range of off-the-shelf deep learning models. Ensemble methods are a reliable way to combine these models and achieve stronger performance. However, in the remote sensing scenario, while the object detection methods are seeing significant progress [1], [2], [3], [4], [5], [6], the potential of ensemble methods is rarely exploited.

Nonmaximum suppression (NMS) [7] is a widely used method to suppress redundant detection bounding boxes in a close neighborhood, by clustering the overlapped bounding boxes (BBs) and eliminating the nonconfidence-maximum BBs in each cluster. Beyond its wide application in single object detectors, it can also be used as a simple ensemble method. However, NMS adopts an affirmative voting strategy, and thus, assumes all of the detection results are true positives, and favors the models that vote for a detected object over those that vote against it.

Weighted boxes fusion (WBF) [8] aims to alleviate the weakness of NMS, by taking into account all the confidence scores of the to-be-fused bounding boxes and assigning an average confidence score to the resulting bounding boxes.

This method, however, leaves two problems unaddressed. First, WBF treats the confidence scores from different models equally and takes the nonweighted mean value as the fused confidence score, disregarding the following three facts. 1) Some models may perform better than other models and their scores should have more weight. 2) Some models may share a similar neural network structure and produce similar results, so the ensembled result may bias toward a group of similar-structured models. 3) Deep learning models are poorly calibrated and different models will be overconfident to different extents, so a simple ensemble method may favor the more overconfident models.

Second, WBF is only compatible with horizontal bounding boxes (see Fig. 1).

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
When deep learning was first introduced into the remote sensing object detection problem, the position of a detected object was initially encoded in the same format as those in the other scenarios, i.e., a nonoriented rectangular bounding box with its sides always horizontal to either one of the side of the image coordinate grids. This format soon posed a problem. Due to the high altitude viewpoint and the steep viewing angle of the remote sensing images, the presented objects can have arbitrary orientations. Some types of objects, such as large ships, buses, buildings, and airport runways, have a large length-to-width ratio and are poorly represented by horizontal bounding boxes, especially when the objects are at a roughly ±45° angle to the image axes.

Oriented bounding box (OBB) was proposed to address this problem. OBB keeps the rectangular form but obtains orientation as a new degree of freedom (DoF), the other existing DoFs being the position of its center, length, and width. OBB introduces finer labels to the objects in the remote sensing images and a better data format for the detection accuracy criteria. However, the existing ensemble methods are not compatible with OBB.

In this article, to address the first problem, a stacking ensemble method is proposed. The stacking model is trained to best combine the member models, while simultaneously considering three factors, model calibration, model redundancy, and the performance gap between the models. For the second problem, a new bounding box fusion method is proposed for the oriented bounding boxes. The bounding boxes are parameterized with orientation, position, width, and height, and each parameter is fused separately. The combined method, OBBStacking, helps take first place in the Challenge Track of the 2021 Gaofen Challenge on Automated High-Resolution Earth Observation Image Interpretation.

This article is structured as follows. Related work will be discussed in Section II. The proposed ensemble method is introduced in Section III. The experiment setup and the quantitative results are described in Section IV. We also provide some analysis of OBBStacking in Section V. The conclusion is given at the end.

II. RELATED WORK

A. Remote Sensing Object Detection

Quite a few deep neural network detectors are proposed in recent years. Notably, Liu et al. [9] are among the earliest to utilize oriented bounding boxes (OBB) for object detection in remote sensing images. The method is built upon faster RCNN [10] and proposes a rotated region of interest (RROI) pooling layer for accurate feature extraction; and an OBB regression model for precise object positioning. Later methods [11], [12], [13] adopt oriented anchors for a better formulation of the bounding box that is easier to learn for the neural networks, but at the cost of relying on a redundant number of rotated anchors. Ding et al. [14] propose ROI Transformer to alleviate the problem by formulating RROI as offset parameters relative to only nonoriented ROIs. Han et al. [15] build upon general rotation equivariant CNNs [16] and ROI transformer to create an oriented object detection model (ReDet) with rotation equivariant features. Xie et al. [17] further simplify the OBB inference process of the ROI transformer with 1/3000 number of parameters used and propose a new model, oriented R-CNN, which is currently state-of-the-art on Dota [18] Dataset.

ReDet and Oriented R-CNN are two of the models we select to generate the detection results for our ensemble method. This is due to their recognized performance on similar problems and their large backbone network difference, where ReDet uses rotation equivariant CNN and Oriented R-CNN uses the more traditional ResNet [19] architecture. The intrinsic difference in their backbone will help increase the model diversity, and in turn, increase the effectiveness of the ensemble process.

B. Transformer

Transformer is another neural network structure we take interest in, due to its structural difference from CNN. It was first introduced by Vaswani et al. [20] for the natural language processing (NLP) problem. It is designed for sequential data and is effective at modeling long-distance dependencies, which is typical in language data. Its success motivated its adaptation to the computer vision domain, with the major hurdle being the difference in the structuring of data (one dimension versus two/three dimensions) and the increased data length at each dimension.

ViT [21] by Dosovitskiy et al. was one of the notable transformer models for computer vision problems. ViT divides one full image into several small patches to be treated as tokens, like the words in NLP, and proposes large-scale pretraining to compensate transformer’s lack of intrinsic properties for image data, such as translation equivariance and feature locality. Based on ViT, DeIT [22] introduces some training strategies that allow a smaller training dataset, and there are some improvements on ViT with conditional positional encoders [23] and more efficient backbone [24].

Feature pyramid is another way to improve ViT. Wang et al. [25] propose to utilize multiresolution feature maps but its complexity is quadratic to image size. Swin Transformer [26] while proposes to boost its efficiency by utilizing the locality characteristic of the images and increasing the scale of features step-by-step through a hierarchical design. Swin Transformer will also be one of the backbones for our member neural network detectors.

C. Calibration of the Neural Networks

A well-calibrated model can produce the probability of correctness for each prediction. Guo et al. [27] show that while modern neural networks excel at making correct predictions, their level of calibration degrades. This hinders the attempt to effectively combine different neural networks and their application in critical scenarios. Guo et al. propose to calibrate the models in a postprocessing manner and train a simple parametric model (Temperature Scaling) [28] to map the confidence scores of the models to the probabilities of correctness. Wenger et al. [29] propose a latent Gaussian process to correct the model output. Zhang et al. [30] propose an ensemble of postprocessing methods that is data efficient and with high generalizability.
The aforementioned methods are postprocessing calibration methods that are most related to our work. There are also calibration methods such as Bayesian neural network methods [31] and neural network regulation methods [32] that change the design philosophy or the objective functions to achieve more calibrated neural networks.

### D. Bounding Box Postprocessing Methods

Object detection methods, along with other vision-related algorithms, may produce redundant activations in a close spatial neighborhood. NMS has been used in such scenarios for over half a century [33] and to this day, is still being used in the deep neural network pipelines. Specifically, modern neural network detectors generate redundant results for a single object and NMS postprocesses the results by checking the spatial overlaps of the results and keeping the ones with the highest confidence scores.

NMS eliminates the redundant bounding boxes completely, which may lead to false negatives when there are overlaps between the ground-truth bounding boxes. Soft-NMS [34] alleviates the problem by keeping all the bounding boxes and only mapping the confidence scores of the to-be-suppressed bounding boxes to a lower value. However, this approach may not fit ensemble methods, where the redundant bounding boxes across different models are common.

Recently, Li et al. [35] propose a new bounding box fusion method (G-NMS) for ship targets. It demonstrates that many of the poor fusion results are due to the lack of consideration of the geometry features of the ship targets, which often have an elongated shape. G-NMS introduces the center distance factor into the fusion process to alleviate the problem. On general objects, especially in an ensemble process, however, G-NMS may perform poorly.

WBF [8] specifically aims at postprocessing the bounding boxes from different models. Instead of selecting one best bounding box (NMS) or keeping all of the bounding boxes (Soft-NMS), WBF produces a weighted average of the bounding boxes in terms of position and size, so all of the to-be-fused bounding boxes can contribute to the final bounding box and no redundant bounding boxes are introduced.

### III. METHODS

OBBStacking is a stacking ensemble method that is compatible with OBBs. In a stacking method, a new model called a metalearner, is trained to best combine the results of multiple existing models. OBBStacking has two stages (see Fig. 2), training the metalearner, and applying the metalearner to the member models. First, we will introduce the metalearner proposed in our method. Then, we will be discussing the key processes that constitute the two stages—namely bounding box clustering, metalearner parameter optimization and bounding box fusion.

#### A. Metalearner

In a stacking method, every member model makes an independent prediction based on a data sample, and the metalearner combines the predictions to form a more accurate one. In OBB-Stacking, we choose a simple model, logistic regression, as the metalearner. The model takes the form

$$\sigma_{WA}(z) = \sigma(zw + b)$$  \hspace{1cm} (1)

where $z = [z_1, z_2, \ldots, z_M] \in \mathbb{R}^{2 \times M}$ is the concatenation of the logit output from $M$ member models, $\sigma(z) = \frac{1}{1 + \exp(-z)}$ is the logistic function, and $w \in \mathbb{R}^M$ and $b \in \mathbb{R}$ are the weight and the intercept parameter of the metalearner, respectively.

Note that logit $z \in \mathbb{R}^2$ is the nonprobabilistic output of the member models and the two dimensions correspond to the tendency of refusing a target and accepting a target, respectively. In the context of deep learning, logits are often converted to probabilistic output through the logistic function, but here the logits are used because of their amenity to (1).

Later in Section V, we will show how this simple form of the metalearner can simultaneously consider model calibration, model redundancy, and the performance gap between the models.

#### B. Bounding Box Clustering

Under the OBB detection setting, each member model produces a set of OBBs, but the correspondence of OBBs between different sets is unknown. Therefore, the first goal is to collect output $z$ on the same object from the different member models. We assume the OBBs are relatively accurate in terms of position and shape such that OBBs generated from the same object but different models have a significant spatial overlap. Therefore, an OBB spatial clustering method is used to assign OBBs from different models. Instead of selecting one best bounding box (NMS) or keeping all of the bounding boxes (Soft-NMS), WBF produces a weighted average of the bounding boxes in terms of position and size, so all of the to-be-fused bounding boxes can contribute to the final bounding box and no redundant bounding boxes are introduced.

### III. METHODS

OBBStacking is a stacking ensemble method that is compatible with OBBs. In a stacking method, a new model called a metalearner, is trained to best combine the results of multiple existing models. OBBStacking has two stages (see Fig. 2), training the metalearner, and applying the metalearner to the member models. First, we will introduce the metalearner proposed in our method. Then, we will be discussing the key processes that constitute the two stages—namely bounding box clustering, metalearner parameter optimization and bounding box fusion.

#### A. Metalearner

In a stacking method, every member model makes an independent prediction based on a data sample, and the metalearner combines the predictions to form a more accurate one. In OBB-Stacking, we choose a simple model, logistic regression, as the metalearner. The model takes the form

$$\sigma_{WA}(z) = \sigma(zw + b)$$  \hspace{1cm} (1)

where $z = [z_1, z_2, \ldots, z_M] \in \mathbb{R}^{2 \times M}$ is the concatenation of the logit output from $M$ member models, $\sigma(z) = \frac{1}{1 + \exp(-z)}$ is the logistic function, and $w \in \mathbb{R}^M$ and $b \in \mathbb{R}$ are the weight and the intercept parameter of the metalearner, respectively.

Note that logit $z \in \mathbb{R}^2$ is the nonprobabilistic output of the member models and the two dimensions correspond to the tendency of refusing a target and accepting a target, respectively. In the context of deep learning, logits are often converted to probabilistic output through the logistic function, but here the logits are used because of their amenity to (1).

Later in Section V, we will show how this simple form of the metalearner can simultaneously consider model calibration, model redundancy, and the performance gap between the models.

#### B. Bounding Box Clustering

Under the OBB detection setting, each member model produces a set of OBBs, but the correspondence of OBBs between different sets is unknown. Therefore, the first goal is to collect output $z$ on the same object from the different member models. We assume the OBBs are relatively accurate in terms of position and shape such that OBBs generated from the same object but different models have a significant spatial overlap. Therefore, an OBB spatial clustering method is used to assign OBBs from the same object into the same cluster.

The clustering method has the following steps.

1. Aggregates all the OBBs from the member models into a list $S$, sorted by their bounding box scores $s$ in descending order.
2. Create an empty list $C$ for the resulting clusters.
3) Pop the first OBB from \( S \) as a new cluster center, and push the cluster into \( C \).
4) Iterate through \( S \) and find OBBs from other member models and have an overlap greater than \( \text{IoU}_{\text{thresh}} \) with the cluster center and move them from \( S \) to the new cluster.
5) Go back to Step 3 and repeat until \( S \) is empty.

Note that although both stages of OBBStacking include bounding box clustering, the method is applied to different sets of data. The whole scheme separates the data into three sets, the training set, the validation set, and the test set. \textit{Training set} is used to train the meta-learner (Stage 1 of OBBStacking). \textit{Validation set} is used for measuring the final performance of OBBStacking (Stage 2). The ground-truth labels for the testing set are usually not publicly accessible in the benchmark datasets, but are provided in the form of testing servers that are accessible online. Member models and the meta-learner are trained on separate datasets to prevent the meta-learner from favoring the member models that overfit the training set.

C. Metalearner Parameter Optimization

After the member models are trained on the \textit{training set} and produce \( M \) sets of detection OBBs on the \textit{validation set}, the bounding box clustering method is applied to acquire the clustered OBBs \( C_{\text{val}} = \{c_i | i = 1, 2, \ldots n \} \). Each OBB in a cluster \( c_i \) represents the prediction of a member model from one data sample \( x_i \).

Here, the major role of the metalearner is to fuse the bounding box scores \( s \) in the same clusters. Note that we use the logit output \( z \) in (1). In most detectors, \( s \) and \( z \) can be acquired by keeping both outputs before and after the last logistic function. Additionally, in most clusters, one or more member models will be absent when they predict the probability is lower than a threshold. We set \( z \) for these cases to a fixed negative value to keep the optimization simple.

We use negative log likelihood (NLL) as the objective function, which can be formulated as

\[
L = -\sum_{i=1}^{n} \log(\sigma_{WA}(z_i)(y_i)) \\
= -\sum_{i=1}^{n} \log(\sigma(z_i w + b)(y_i))
\]  

(3)

where \( y_i \) is the ground-truth label of each cluster. To determine \( y_i \), we calculate intersection over union (IOU) between the cluster center OBB and all the ground-truth OBBs in the validation set. A cluster is marked as a true positive (\( y = 1 \)) if it has an overlapped ground-truth OBB, and a false positive (\( y = 0 \)) otherwise.

Equation (3) is a convex function regarding to \( w \) and \( b \), and can be easily optimized.

D. Oriented Bounding Box Fusion

Before this step, the trained member models produce \( M \) sets of OBBs from the \textit{test set}, which are then clustered into \( C_{\text{test}} \) with the bounding box clustering method. This step aims to fuse the OBBs \( O = \{o_1, \ldots , o_K \} \) that belong to the same cluster into one OBB. We represent an OBB with a 7-tuple:

\[
o = (x, y, w, h, \theta, z, l)
\]  

(4)

where, \( x, y, w, h, \) and \( z \) represent the center coordinates on the x–y axis, width, height, and logit score, respectively. \( l \in \{1, 2, \ldots , M \} \) is the index of its source model. Orientation \( \theta \in [0, \pi) \) represents the angle between the longest axis of the bounding box and the x-axis.

The fusion process needs to derive the first five elements in \( o \) to acquire the final OBB, and these elements will be fused separately. With regard to the first four elements, the fusion process can be formulated as

\[
o_{\text{fused}}(j) = \frac{\sum_{p=1}^{n} o_p(j) s_p^*}{\sum_{p=1}^{n} s_p^*}, j = 1, 2, 3, 4
\]

(5)

where \( j \) is the index of the element in \( o \), \( p \) is the index of the OBB in the cluster, and \( o_p \) is the fused OBB. \( s^* \) is the calibrated score derived from OBB’s logit score and the weight parameters in (1) as follows:

\[
s_p^* = \sigma(z^{(1)} w^{(l_p)} + b)
\]

(6)

\( s^* \) acts like an improved weight for each OBB that addresses the output calibration and the redundancy in the member models.

Orientation parameter \( \theta \) receives special treatment due to its cyclic property. First, the orientation of the bounding box with the largest score \( s^* \) is designated as the major orientation \( \theta_{\text{MJ}} \) of the cluster. Then, the fused orientation is determined by averaging the relative orientations to \( \theta_{\text{MJ}} \) as follows:

\[
\theta_f = \frac{\sum_{p=1}^{n} r(\theta_p, \theta_{\text{MJ}}) s_p^*}{\sum_{p=1}^{n} s_p^*} + \theta_{\text{MJ}}
\]

(7)

where \( r \) is a bivariate function that calculates the relative difference between the two angles while considering their cyclic property:

\[
r(\theta_1, \theta_2) = \begin{cases} 
\theta_1 - \theta_2, & \text{for } \theta_1 - \theta_2 \leq \frac{\pi}{2} \\
\theta_1 - \theta_2 + \pi, & \text{for } \theta_1 - \theta_2 < -\frac{\pi}{2}
\end{cases}
\]

(8)

Note that here we assume \( \theta \in [0, \pi) \) since we do not discriminate between the head and the tail of an OBB.

We further illustrate how the relative angles are determined in Fig. 3. Here, the green OBB is designated as the source of the major orientation. The relative angles of the yellow OBB and the red OBB are marked as \( r_0 \) and \( r_1 \), respectively. Note how we use the smallest angles between two lines and since \( r_0 \) is on the clockwise side of \( \text{MJ} \), we have \( r_0 > 0 \), while in contrast \( r_1 < 0 \) since it is on the counter-clockwise side.

Finally, the score of the fused bounding box is determined with (1) with the learned metalearner.

IV. RESULTS

A. Datasets

Two datasets are used to validate our method, FAIR1M 1.0 dataset [36] and DOTA 1.0 dataset [18] (see Fig. 4). Both datasets have an evaluation server that evaluates the detection results on a
TABLE I
QUANTITATIVE RESULTS ON DOTA DATASET, TRAINED WITH THE TRAINING SET ONLY

| Methods      | PL | BD | BR | GTF | SV | LV | SH | TC | BC | ST | SBF | RA | HA | SP | HC | mAP |
|--------------|----|----|----|-----|----|----|----|----|----|----|-----|----|----|----|----|-----|
| Individual   |    |    |    |     |    |    |    |    |    |    |     |    |    |    |    |     |
| Oriented R-CNN | 89.84 | 85.16 | 60.99 | 79.57 | 79.75 | 84.92 | 88.44 | 90.88 | 84.43 | 87.56 | 70.39 | 68.38 | 81.51 | 77.81 | 68.35 | 79.86 |
| ReDet        | 88.20 | 84.25 | 56.05 | 79.95 | 76.97 | 85.82 | 88.39 | 90.90 | 87.39 | 86.24 | 67.27 | 63.32 | 77.68 | 74.89 | 71.12 | 78.56 |
| Swin Det     | 88.77 | 81.99 | 57.59 | 76.63 | 65.26 | 84.24 | 87.96 | 90.83 | 84.49 | 87.24 | 63.36 | 64.45 | 80.74 | 67.34 | 65.82 | 76.58 |
| Ensemble     |    |    |    |     |    |    |    |    |    |    |     |    |    |    |    |     |
| G-NMS        | 79.79 | 85.46 | 56.62 | 81.56 | 73.75 | 85.15 | 87.39 | 90.77 | 84.16 | 85.77 | 66.79 | 68.86 | 78.20 | 68.59 | 63.88 | 77.12 |
| Soft-NMS     | 87.65 | 85.65 | 57.56 | 81.91 | 74.38 | 85.60 | 88.14 | 90.81 | 86.47 | 87.47 | 69.20 | 69.75 | 81.79 | 71.48 | 69.77 | 79.18 |
| NMS          | 89.49 | 84.84 | 60.18 | 80.94 | 78.91 | 86.26 | 88.90 | 90.90 | 87.43 | 87.59 | 72.93 | 69.35 | 82.12 | 77.34 | 75.24 | 80.83 |
| WBF          | 89.49 | 84.94 | 60.20 | 80.94 | 78.99 | 86.25 | 88.90 | 90.90 | 87.43 | 87.84 | 73.06 | 70.62 | 82.45 | 76.13 | 75.24 | 80.89 |
| Ours         | 89.31 | 85.66 | 61.76 | 81.47 | 79.29 | 86.45 | 88.87 | 90.89 | 87.68 | 88.50 | 73.02 | 72.47 | 83.06 | 78.49 | 75.53 | 81.50 |

Fig. 3. Illustration of the relative orientation.

The Swin detector in our experiment is a simple modification to the original one [26] for its compatibility with OBB detection. Both Swin backbone and the recent CNN backbone produce a feature pyramid [38], consisting of layers of image features with different spatial resolutions and semantic depths, so their outputs have a similar structure and they can share the same types of detectors. We keep the original backbone and replace the original detector head with the one from Oriented R-CNN, since its OBB detector structure is elegant and concise.

For Oriented R-CNN and ReDet, we follow the experiment setups in the original papers, except for those that can be limited by the GPU specifications. We use a similar setting in Swin Detector to the ones in Oriented R-CNN since they share the same type of detectors. We use 2 GTX 3080 Ti for training and inference. The images are cropped into 1024 × 1024 patches and the batch size is set to 2, 2, and 1 per GPU for Oriented R-CNN, ReDet, and Swin Detector, respectively, due to the limit of GPU memory. Multiscale training and testing are also used because they are often used in combination with ensemble methods to achieve the highest performance possible.

C. Quantitative Comparison

First, for a fair comparison, we augment the original Soft-NMS, G-NMS, NMS, and WBF with OBB compatibility by including our proposed bounding box fusion method; and evaluate the performance of the member models and the selected ensemble methods on DOTA dataset. Since most of the experiments in the literature [15], [17] combine the training and the validation sets to train their models to achieve maximum performance, and our ensemble model needs a separate validation set to learn the parameters of the metalearner, we do two separate experiments to verify the effectiveness of our method.

1) We follow the original scheme of our method, and train all the member models on the training set only, leaving the validation set for the parameter training of the metalearner.

2) We follow the training scheme of other methods and train the member models with data from both the training set and the validation set, and use the trained metalearner from Experiment (1).

In the following tables on DOTA dataset, the names of the categories are abbreviated to conserve space. The categories, in order, are plane, baseball-diamond, bridge, ground-track-field, small-vehicle, large-vehicle, ship, tennis-court,
Fig. 4. Showcase of the ensemble results of OBBStacking on DOTA dataset and FAIR1M dataset. Only the objects with a confidence score larger than 0.2 are shown.

The quantitative results of Experiment (1) are listed in Table I. Oriented R-CNN achieves the best performance among the member models and our method achieves the top score with 81.50% mAP, 0.61% over WBF. G-NMS and Soft-NMS perform poorly, probably because the assumptions they are based on are not compatible with the ensemble process, as discussed in Section II-D.

For Experiment (2), we assume the performance gap, the calibration, and the redundancy of the member models do not drift too much from Experiment (1), and we could reuse the metalearner for the ensemble. The results are shown in Table II. The results are generally similar to the previous one, with a slight overall performance increase of 1% mAP among the member models and 0.1–0.4% mAP increase among the ensemble methods. Our method, with the metalearner from Experiment (1), still outperforms WBF by 0.24% mAP. This shows that our assumption holds when the training data expands, and even though our method requires a separate validation set, it still outperforms the existing ensemble methods.

Next, we evaluate the member models and the ensemble models on FAIR1M dataset using Experiment (1) setup and show the results in Table III. Among the member models, Oriented R-CNN still achieves the best performance with 47.77% mAP. Composed to the individual methods, the ensemble models obtain a huge performance increase by around 4% mAP, where our method achieves the best score with 52.42% mAP, a 4.65% increase over Oriented R-CNN, a 0.57% mAP increase over WBF.
We notice there is a large performance gap between Dota and FAIR1M dataset. This is mainly due to the larger number of fine categories in FAIR1M, which raises the difficulty level for the detection methods. Many fine-grained categories are difficult to distinguish, even for humans, such as Boeing 737, A220, and A330, tractors, dump trucks, etc.

Finally, we want to mention that the only difference between the augmented WBF and OBBStacking is the involvement of a metalearner. So the results in this comparison can also act as an ablation experiment on the metalearner component.

V. DISCUSSION

In this section, we discuss how OBBStacking addresses the three problems that arise during an ensemble process on deep learning models—namely model calibration, the performance gap between the models, and model redundancy.

A. Model Calibration

Deep learning models tend to overfit the training data and are overconfident about their predictions. When the member models are overconfident to different degrees, their predictions are on different measurements and do not indicate true probability values. Therefore, the ensemble methods may not work well on these models as intended, and a model calibration process is needed.

In this section, we show that one of the calibration methods, temperature scaling (TS) [27], can be regarded as a special form of our metalearner, indicating that OBBStacking includes the feature of model calibration.

TS attempts to map the nonaccurate predictions to the real probability of correctness, by "softening" the final logistic layer in the neural networks and introducing a temperature parameter $T > 1$. The "softened" logistic layer is

$$\sigma_{TS}(z) = \sigma(z/T + t)$$

When $T \to \infty$, all results of $\sigma_{TS}$ approach $\frac{1}{2}$ and indicate maximum uncertainty.

The inference of parameter $T$ also uses NLL as the objective function, since NLL is a standard measure of a probabilistic model's quality [39]. Here, the objective function can be defined as

$$L = -\sum_{i=1}^{n} \log(\sigma(z_i/T + t)^{(y_i)})$$
As can be seen, our metalearner, (1) becomes (10), when the number of the member models is 1, and thus, can calibrate models in the same fashion.

B. Performance Gap

In this section, we experiment to try to demonstrate how OBBStacking adjusts the weights when there is a performance gap between the models.

Our model tackles three problems simultaneously, model calibration, redundancy, and performance gap. We assume these three problems can be disentangled and thus the factorization of the parameter exists, \( w = p \odot r \odot g \), where the operator \( \odot \) is the elementwise multiplication, \( p, r, \) and \( g \) are the weight vectors for the model calibration, model redundancy, and the performance gap, respectively.

We want to minimize the effect of the first two factors and see how OBBStacking handles the performance gap between the models. Along with the Swin detector used in our previous experiment, three additional Swin detectors are added to the Swin detector family. The only difference between these Swin detectors is the total number of epochs used in training, which are 12, 9, 16, and 18 epochs, respectively. At different epochs during the training with stochastic gradient descent, the neural networks may randomly lean toward more accuracy on some categories instead of others, and rely upon different features, thus creating a sequence of different models with relatively high redundancy.

We first run OBBStacking on the Swin family and acquire \( w \) for later comparison. Then, to show the factor of redundancy among the Swin families, their weights decrease drastically, with a sum value of 0.36, in between the weights of Oriented R-CNN and ReDet. The weights of Oriented R-CNN and ReDet decrease slightly because the Swin family improves its performance with the increase of its members.

C. Model Redundancy

In this part, we build upon the previous experiments to show how OBBStacking handles model redundancy. Two collections of models are included. Collection 1 consists of Oriented R-CNN, ReDet, and Swin 12. Collection 2 includes all the models in Collection 1 and the additional Swin 9, Swin 16, and Swin 18, adding up to six models in total.

The correlation coefficient between all the models is shown in Fig. 5 and the weight parameters \( w \) of the metalearner are shown in Table V. We notice that in Collection 2, because of the redundancy among the Swin families, their weights decrease drastically, with a sum value of 0.36, in between the weights of Oriented R-CNN and ReDet. The weights of Oriented R-CNN and ReDet decrease slightly because the Swin family improves its performance with the increase of its members.

VI. Conclusion

We propose an ensemble method, OBBStacking, that is compatible with the oriented bounding box (OBB), which is widely used in object detection in the remote sensing field. OBBStacking consists of a metalearner that can address the problems in the ensemble process of the deep neural network detectors, namely the model calibration, the redundancy between the models and the performance gap between the models. OBBStacking outperforms other ensemble methods in the DOTA dataset and the FAIR1M dataset and helps us win first place in the Challenge Track Fine-grained Object Recognition in High-Resolution Optical Images featured in 2021 Gaofen Challenge on Automated High-Resolution Earth Observation Image Interpretation.
REFERENCES

[1] G. Cheng et al., “SPNet: Siamese prototype network for few-shot remote sensing image scene classification,” IEEE Trans. Geosci. Remote Sens., vol. 60, Jul. 2022, Art. no. 5608011.

[2] E. Liu, Y. Zheng, B. Pan, X. Xu, and Z. Shi, “DCL-Net: Augmenting the capability of classification and localization for remote sensing object detection,” IEEE Trans. Geosci. Remote Sens., vol. 59, no. 9, pp. 7933–7944, Sep. 2021.

[3] A. Qin, S. Lin, G. Cheng, X. Yao, H. Ren, and W. Wang, “Object detection in remote sensing images based on improved bounding box regression and multi-level features fusion,” Remote Sens., vol. 12, no. 1, Jan. 2020, Art. no. 143.

[4] L. Shi, L. Kuang, X. Xu, B. Pan, and Z. Shi, “CA-Net: Centerness-aware network for object detection in remote sensing images,” IEEE Trans. Geosci. Remote Sens., vol. 60, Apr. 2022, Art. no. 563613.

[5] X. Yao, H. Shen, X. Feng, G. Cheng, and J. Han, “R2Points: Pursuing rotation-insensitive point representation for aerial object detection,” IEEE Trans. Geosci. Remote Sens., vol. 60, May 2022, Art. no. 5623512.

[6] J. Zhang, R. Huang, Y. Li, and B. Pan, “Oriented ship detection based on intersecting circle and deformable RoI in remote sensing images,” Remote Sens., vol. 14, no. 19, Jan. 2022, Art. no. 4749.

[7] F. Elzehr, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part based models,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[8] R. Soloyev, W. Wang, and T. Gaburseva, “Weighted boxes fusion: Ensemble boxes from different object detection models,” Image Comput., vol. 107, Mar. 2021, Art. no. 104117.

[9] Z. Liu, J. Hu, L. Weng, and Y. Yang, “Rotated region based CNN for ship detection,” in Proc. IEEE Int. Conf. Image Process., 2017, pp. 900–904.

[10] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Toward real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[11] Z. Zhang, W. Guo, S. Zhu, and W. Yu, “Toward arbitrary-oriented ship detection with rotated region proposal and discrimination networks,” IEEE Geosci. Remote Sens. Lett., vol. 15, no. 11, pp. 1745–1749, Nov. 2018.

[12] X. Yang et al., “Automatic ship detection in remote sensing images from Google earth of complex scenes based on multiscale rotation dense feature pyramid networks,” Remote Sens., vol. 10, no. 1, 2018, Art. no. 132.

[13] S. M. Azimi, E. Vig, R. Bahmanyar, M. Körner, and P. Reimartz, “Towards multi-class object detection in unconstrained remote sensing imagery,” in Proc. Asian Conf. Comput. Vis., 2018, pp. 150–165.

[14] J. Ding, N. Xue, Y. Long, G.-S. Xia, and Q. Lu, “Learning ROI transformer for oriented object detection in aerial images,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Piscataway, NJ, USA, 2019, pp. 2849–2858.

[15] J. Han, J. Ding, N. Xue, and G.-S. Xia, “Redet: A rotation-equivariant detector for aerial object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 2786–2795.

[16] M. Weiler and G. Cesa, “General E (2)-equivariant steerable CNNs,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 14334–14345.

[17] X. Qian, G. Cheng, J. Wang, X. Yao, and J. Han, “Oriented R-CNN for object detection,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 3520–3529.

[18] G.-S. Xia et al., “DOTA: A large-scale dataset for object detection in aerial images,” in Proc. IEEE Comput. Vis. Pattern Recognit., 2018, pp. 3974–3983.

[19] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Piscataway, NJ, USA, Dec. 2016, pp. 770–778.

[20] A. Vaswani et al., “Attention is all you need,” in Proc. 29th Int. Conf. Neural Inf. Process. Syst., 2017, pp. 1982–1990.

[21] A. Dosovitskiy et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” in Proc. Int. Conf. Learn. Representations, 2021.

[22] H. Tourvnon, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, “Training data-efficient image transformers & distillation through attention,” in Proc. 38th Int. Conf. Mach. Learn., Jul. 2021, pp. 10347–10357.

[23] X. Chu et al., “Conditional positional encoding for vision transformers,” in Proc. Int. Conf. Learn. Representations, 2021.

[24] L. Yuan et al., “Tokens-to-Token ViT: Training vision transformers from scratch on ImageNet,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021.

[25] W. Wang et al., “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 548–558.