Stacked Generalization Models for Improving Multiclass Object Detection in Drone Images

Allison Nicole Paxton
Department of Computer Science
California State University
Northridge, CA, USA
allison.paxton.505@my.csun.edu

Abhishek Verma
Department of Computer Science
California State University
Northridge, CA, USA
abhishek.verma@csun.edu

Abstract*— Drones have become increasingly popular due to their many real-world applications such as military, surveillance, agriculture. Automatic detection of objects in drone camera footage poses challenges due to varied angle, heights, occlusion, and illumination. Training a machine learning model on other machine learning model predictions is called meta-learning, and meta-learning from multiple machine learning models is called Stacked Generalization. This research used this technique to propose eight Stacked Generalization models to improve on the three object detection models – Faster R-CNN, RetinaNet, and YOLOv5s. The regression models trained as meta-models were XGBoost and Multilayer Perceptron (MLP). Experiments were performed on the challenging VisDrone image dataset. All the proposed Stacked Generalization models improved from the object detection models they learned from, thereby performing better than the state-of-the-art base models. The best regression model as a meta-model was MLP. The best Stacked Generalization model in terms of accuracy was FRCNN-s + RetinaNet-s + YOLOv5s-s with MLP meta-models, which scored an mAP50 of 0.421. Its mAP was 0.244, which improved 4.72% from YOLOv5s.

Keywords—Deep learning; Stacked Generalization; Meta-learning; VisDrone2019-DET; Faster RCNN, RetinaNet, YOLOv5

I. Introduction

Autonomous multiclass object detection in drone images is a challenging area of research. Already trained machine learning models can increase their prediction accuracy by using their predictions or meta-data to train another model, or meta-model. The idea of meta-models further learning from what the base models have already learned is called meta-learning. Our research focus is on meta-learning from multiple object detection models to increase the accuracy of the object detection models and create the best performing model.

Meta-learning from multiple machine learning models is called Stacked Generalization (SG). Once an input dataset has been created from the combined predictions of the trained base machine learning models, the dataset can be used to train a classification or regression meta-model. SG models have two layers. The first are the level-0, or base, machine learning models and the second are the meta-models, or level-1 models. Since our research is meta-learning object detection, the level-1 models were regression models to learn from the predicted detection boxes. The level-1 regression models used in our experiments were XGBoost and Multilayer Perceptron (MLP). With four combinations of level-0 models and two different level-1 models, we propose eight new SG models, all of which improved their corresponding level-0 models. So, the best performing model was found to be an SG model.

Related works, dataset description, methodology, experiment setup, and experiment results are examined to go over the research. First, works that use the same dataset or complete similar tasks such as object detection or meta-learning and state-of-the-art approaches are discussed in section II. Next, the VisDrone2019-DET and VisDrone-split drone image datasets are explained in Section III. Third, the methodology of not only the proposed SG models, but also the level-0 object detection models are presented in section IV. These level-0 models are Faster R-CNN (FRCNN) [1], RetinaNet [2], and YOLOv5s [3]. Section V briefly explains the hardware and software used, the metrics to compare models, and how the level-0 and level-1 models were trained. Section VI provides and discusses the results of the level-0 and SG models using XGBoost and MLP level-1 regression models. Lastly, section VII is the conclusion and future work.

II. Related Work

The following articles documented the research on object detection methods and meta-learning for object detection. The first article published in 2021 compared object detection methods Faster R-CNN [1], YOLOv3 [4], RetinaNet [2], CenterNet [5], YOLOv5 [3], etc. The methods were trained on VisDrone2019 [6] and VisDrone-split as explained in [6]. VisDrone-split is the same dataset as VisDrone2019 except it splits each image into 600 by 600 pixel sub-images with a 150 pixel overlap between adjacent sub-images. The researchers also proposed their own method, called UCGNet-o and UCGNet, which was also trained on VisDrone2019 and VisDrone-split, respectively. UCGNet uses a Local Location Module (LLM) to localize the objects with a binary map, an Unsupervised Clustering Module (UCM), and the Farthest Point Sampling (FPS) method to improve clustering efficiency. The results also improved the proposed method after using VisDrone-split. Splitting the dataset is used in our research as well because training models on VisDrone-split provided us better results than training on VisDrone2019.

The final two articles detailed the work done to improve the detection of small objects and objects within a dense image in

[7], [8]. The resulting methods are called YOLOv3_ReSAM and AdaZoom. YOLOv3_ReSAM improves small object detection by first improving on the YOLOv3-tiny backbone by modifying the top-level image feature pyramid so that there is a small-scale layer, mesoscale layer, and large-scale layer depending on the size of the target. This is because if there is a single scale, then the features of small targets might not get noticed. AdaZoom improves small object detection by finding the regions with a dense amount of small object, then uses reinforcement learning to determine the magnification of the zoom based on the size of the objects. YOLOv3_ReSAM and AdaZoom defines small objects as having areas less than 0.1% of the whole image area. With these methods, both YOLOv3_ReSAM and AdaZoom improved on the YOLOv3-tiny. So, the article explains why models on highly dense datasets have room for improvement and thereby provided us in our research to explore improved approaches such as meta learning.

III. Dataset Description

A. VisDrone2019-DET

VisDrone2019 dataset comes from the AISKYEYE team at the Tianjin University in China [6], [9]. One of four tasks can be completed with this dataset. Our research used VisDrone2019-DET for the object detection task. The ten classes are pedestrian, people, bicycle, car, van, truck, tricycle, awning-tricycle, bus, and motor. There are 7,019 images between the training and validation datasets. The validation dataset was used as the test dataset in this research. There was a wide range of image sizes, so they were resized to 1,500 by 1,500 pixels to have the same dimensions.

B. VisDrone-split

Figure 1 shows an image from the VisDrone2019 training dataset. Every class except truck and bus are shown. On the left side of the image there are three cars hidden behind trees within yellow boxes showing that they are a part of the ground truth. There is a car and a tricycle in the top of the image that are also hidden behind trees, but they are not a part of the ground truth. To handle the errors in the ground truth, each image was split into equal sized sub-images with a length and height of 500 pixels. The images without objects in the ground truth were removed. This new dataset was named VisDrone-split, which has a total of 39,557 images in the training and test datasets.

IV. Research Methodology

The object detection models that were trained on the VisDrone-split dataset were Faster R-CNN, RetinaNet, and YOLOv5s. The predictions of these three models trained the proposed Stacked Generalization models, so this section goes over the methodology of various models.

A. Faster RCNN

Faster R-CNN (FRCNN) [1] has a ResNet + Region Proposal Network (RPN) backbone. The model in our research specifically used ResNet50. With an input image of any size, RPN uses anchor boxes and a neural network to generate a set of regions and locate potential objects. The class loss is Log loss of the object and not object classes. The regression loss is a robust loss function. The class and regression losses are labeled

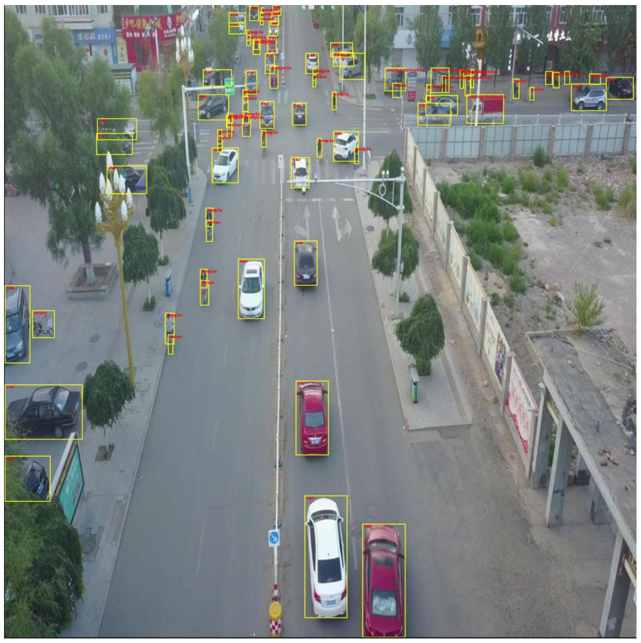
Figure 1. An annotated 1,500 pixel squared training image from VisDrone2019 [6] dataset. Examples from eight of the ten classes are shown in the yellow bounding box – pedestrian, bicycle, can, van, tricycle, awning-tricycle, and motor.

as L_{cls} and L_{reg}, respectively. The loss function for anchor is the following:

$$L = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \qquad (1)$$

where p_i and $p_i^* \epsilon \{0,1\}$ are the probabilities of the i^{th} anchor and the ground truth, respectively. Ground truth p_i^* is only 1 if the i^{th} anchor is positive. The anchor and ground truth bounding box coordinates are represented by the t_i and t_i^*, respectively. L_{cls} and L_{reg} are normalized with N values. Once the objects were located, then they were classified after applying ROI Pooling. This is faster than using a Selective Search approach to locate potential objects.

B. RetinaNet

RetinaNet [2] has a ResNet backbone followed by a Feature Pyramid Network (FPN). The model in our research specifically used ResNet50. FPN is used to extract features by creating feature maps at several scales. After FPN is applied, RetinaNet uses subnetworks to locate and classify predictions of anchor boxes. RetinaNet uses the focal loss function, which uses a focusing parameter to increase loss for predictions with a smaller probability [10].

C. YOLOv5

YOLOv5 [3] can be characterized by its backbone, neck, and head. The backbone is CSPDarknet53 [11] with Cross Stage Partial connections (CSP) [12], [13]. The neck is Spatial Pyramid Pooling Fusion (SPPF) [14] and CSP-Path Aggregation Network (CSP-PAN) [15], [12]. SPPF comes from Spatial Pyramid Pooling (SPP) [16], which removes the need for fixed input image sizes. However, SPP is slower than SPPF. Individually, CSP reduces expensive computations and PAN preserves localization information to improve instance

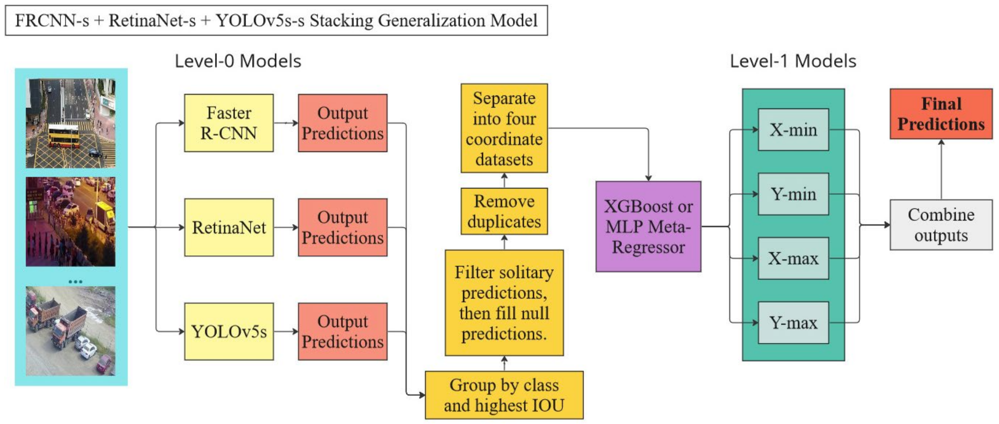
FRCNN-s + RetinaNet-s + YOLOv5s-s Stacking Generalization Model
Level-0 Models
Level-1 Models
Faster R-CNN
RetinaNet
YOLOv5s
Output Predictions
Output Predictions
Output Predictions
Group by class and highest IOU
Filter solitary predictions, then fill null predictions.
Remove duplicates
Separate into four coordinate datasets
XGBoost or MLP Meta-Regressor
X-min
Y-min
X-max
Y-max
Combine outputs
Final Predictions

Figure 2. System pipeline of one of the eight proposed Stacked Generalization Model that combined FRCNN-s + RetinaNet-s + YOLOv5s-s, where "-s" stands for the VisDrone-split dataset for which the level-0 models are trained on. The objects in the input images are predicted by the level-0 models as shown in the blue, light yellow, and light red text boxes. The predictions then grouped and used to create four meta-datasets as shown in the deeper yellow text boxes. These four meta-datasets are used to train four level-1 meta-regressors to predict one bounding box coordinate each, which is shown in the purple and teal text boxes. The last two text boxes show combining the coordinate prediction to get the final set of predictions.

segmentation. There is an output head and loss function for bounding boxes, objectness, and class separately. Objectness loss uses Binary Cross Entropy (BCE) loss, class loss also uses BCE loss, and location loss uses Complete Intersection over Union (CIOU) loss [17]. BCE for *m* classes and *n* predictions is:

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} * log(P(y_{ij})), \quad (2)$$

where $P(y_{ij})$ is the probability of class *j*. CIOU for two detection boxes *a* and *b* is:

$$CIOU(a,b) = 1 - IOU(a,b) + dist(\text{a},\text{b})^2/c^2 + \alpha \quad (3)$$

where *dist(a,b)* is the Euclidean distance, *c* is the length of the diagonal of the smallest box covering both *a* and *b*, and *α* is the aspect ratio.

D. *Proposed Stacked Generalization Models*

1) Creating the Meta-dataset

The predictions from level-0 object detection models were used to train the meta-models and test the proposed Stacked Generalization models. First, datasets to train meta-regression models were created, which is shown in the orange text boxes in Figure 2. The predictions from the level-0 models and the ground truths of the training dataset were first converted into tables. The level-0 tables contained the four coordinates of bounding boxes, the classification, the image number, the confidence, and the model as its features. Next, each ground truth in each image was paired with the closest prediction from each level-0 model. To select the closest prediction to a ground truth, the prediction had the same classification and the highest Intersection over Union (IOU) with that ground truth. IOU is a ratio between two bounding boxes that measures their overlapping area to their total area. If there was no prediction from any level-0 model that was paired with a ground truth, then that ground truth was not used in training. If there was at least one prediction paired with a ground truth, then a null prediction was used as any missing prediction from the other level-0 models. Each ground truth is left with a group of predictions from each level-0 model. To replace the null(s) in each group, the prediction closest to its ground truth is used. After removing the duplicates, a meta-dataset is created with the four coordinates of the ground truth and each level-0 bounding box, the confidence of each level-0 prediction, and IOU between every two level-0 predictions to measure how close the predictions are to each other.

2) Training the Level-1 models

The meta-dataset was used to train level-1 models to predict each coordinate of a bounding box, which is why regressors were trained. However, the distribution of objects between the ten classes is highly skewed, so the predicted class of each bounding box is not predicted. To predict the coordinates, four level-1 models were used to predict one coordinate each. This means the meta-dataset was split into four meta-datasets. For example, the meta-dataset to train a level-1 model to predict the Y-min coordinate used the Y-min coordinates of each level-0 model, the confidence scores, and the IOUs as its features.

These four meta-datasets are used to train four XGBoost level-1 models or four MLP level-1 models, which is shown in the purple and teal text boxes of Figure 2. Four-fold cross validation was used to find the number of trees, the depth of each tree, and the learning rate to provide the best results of the XGBoost level-1 models. Each MLP had two fully connected layers and an output layer of one node. The number of nodes in the first layer was one plus the number of features in the meta-dataset since the more level-0 models there were, the more features there were. The Mean Squared Error (MSE) was used to determine the best time to stop training, which is further explained in the next section. The MSE for n datapoints is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2, \quad (4)$$

where Y are the ground truths and Y hat are the predicted coordinates.

3) Testing the Stacked Generalization Models

To test the resulting SG models after training, the test meta-dataset is created by pairing each level-0 prediction to every other level-0 prediction instead of the ground truth. The resulting meta-dataset was split similarly to the training meta-dataset. After the coordinate outputs were combined, duplicates were removed since many of the same predictions were in multiple groups. The final set of predictions could then be used to calculate the metrics. Figure 2 shows the specific SG, FRCNN-s + RetinaNet-s + YOLOv5s-s, which is one of the eight proposed SG models.

V. Experiment Setup

To perform the SG experiments, certain hardware and software was required, object detection metrics needed to be understood, and the level-0 and level-1 model parameters needed to be set up to be trained.

A. *Hardware and Software*

The three level-0 models and all the SG experiments were trained using a NVIDIA GeForce RTX 3060 with 6 GB of dedicated memory. The application to run the experiments was Jupyter Notebook. To train the level-0 models on a custom dataset required an API or set of libraries, a dataset format, and file type for the dataset. This also includes a preferred or required operating system. The two file types for object detection used were text for YOLOv5s and JSON for FRCNN and RetinaNet. The format for the annotations in JSON files is given at the official COCO website [18]. The coordinate format that VisDrone provided was "*X center, Y center, width, height*." However, the level-0 models required the "*X min, Y min, X max, Y max*," coordinate format. These box coordinates were in pixels for FRCNN and RetinaNet and normalized for YOLOv5s. The meta-datasets to train the level-1 models were set up using a CPU.

B. *Metrics*

The metric, average precision (AP), measures the prediction accuracy of a single class object detection model. After ordering each prediction by its confidence score, the precision and recall values are calculated after each prediction. Each set of values creates an (x, y) point to create the precision-recall curve, and the area under the precision-recall curve is the AP, which is detailed in [19]. The mean AP (mAP) is the result of finding the AP for each class separately in a multiclass model, then averaging the results. One of the metrics used in this research was mAP with a 50% Intersection over Union (IOU) threshold (mAP50), which means that the IOU of a true positive prediction and a ground truth is more than 50%. The other metric used was mAP with a 50:5:95% IOU threshold (mAP). This metric is calculated by averaging all the mAPs from an IOU of 50% to 95% with increments of 5%. The COCO website [20] states that mAP is the best metric to determine performance on COCO, so mAP was the priority over mAP50 to determine the best model in this research.

C. *Level-0 Model Parameter Setups*

The set of object detection models that were trained were YOLOv5s [3], Faster R-CNN (FRCNN) [1], and RetinaNet [2]. Training the YOLOs required annotations in text files as in the original dataset, but each box coordinate was normalized. Training YOLOv5s on VisDrone-split used the Adam optimizer and a learning rate of 0.005 or 0.0025. FRCNN and RetinaNet both models required the annotations in JSON files. They trained for at least 10,000 iterations each once loss seemed to flatten sufficiently. Before training on a custom dataset, the dataset was registered to be accessed, the model weights were initialized, and the parameters were set. Training FRCNN and RetinaNet on VisDrone-split used default parameters with a learning rate of 0.002 or 0.001. After training, a 0.1 confidence threshold provided optimum results.

D. *Level-1 Model Parameter Setups*

As was explained in part C of Section IV, fourfold cross validation was used to train XGBoost meta-regressors and two fully connected layers were used to create the MLP meta-regressors. For XGBoost, the hyperparameters to generate the best results are about 150 to 250 trees, a max depth of four, five, or eight for each tree, and a learning rate of 0.05 for each coordinate level-1 model. To find the best combination of three hyperparameters, GPU was used to speed up training. For each MLP level-1 model, an Adam optimizer, a learning rate of 0.001, and a batch size of 128 was used. Each MLP ran for a maximum of 100 with ten epochs of early stopping based on the lowest MSE. To determine the best time to stop, one fourth of the training dataset was used as a validation dataset. Once MSE of the validation dataset did not decrease after ten epochs, then the epoch with the lowest validation MSE was used as the final MLP coordinate model. To run for a maximum of 100 epochs, a GPU was also used to speed up training.

TABLE I. SG Model Average Runtime in Minutes Based on the Level-1 Model and Number of Level-0 Models

Number of Level-0 Models	**2**	**3**
XGBoost SG Runtime (min)	39	66
MLP SG Runtime (min)	45	71

Table 1 shows the average runtime of combining two and three level-0 models based on what meta-regressor was used in the SG model. The time is from grouping the level-0 predictions to getting the results to an SG model. There is only one SG model with three level-0 models using either XGBoost or MLP, so its runtime is what is shown in the table. However, the average runtime of all the of two level-0 models for each meta-model is taken. Clearly there is a longer runtime for more level-0 models being combined. There is also a longer runtime from using an XGBoost to MLP level-1 model. The next section shows the SGs with MLP results are worth the additional runtime.

VI. Results and Discussion

The eight experiments done included four SG experiments using XGBoost level-1 models and four SG experiments using MLP level-1 models. The four experiments using each type of level-1 model came from the four combinations of FRNN, RetinaNet, and YOLOv5s. The MLP SG models are consistently more accurate than their XGBoost counterparts.

A. *Level-0 Models*

The set of level-0 models are Faster R-CNN (FRCNN), RetinaNet, and YOLOv5s trained on VisDrone-split shown in both Table II, where "-s" is for models trained on VisDrone-split. The metric results of YOLOv5s trained on VisDrone and VisDrone-split were similar and the results of FRCNN trained on VisDrone-split were significantly better than trained on VisDrone. So, experiments were done with level-0 models

TABLE II. Level-0 Models and Proposed MLP and XGBoost based SG MODELS' MAP PER CLASS AND MAP50

Method	**mAP50**	**mAP**	**pedes-trian**	**people**	**bicycle**	**car**	**van**	**truck**	**tricycle**	**awning-tricycle**	**bus**	**motor**
Level-0 Models												
FRCNN	0.304	0.168	0.133	0.087	0.023	0.479	0.237	0.173	0.107	0.057	0.257	0.127
FRCNN-s	0.380	0.209	0.245	0.177	0.095	0.513	0.247	0.148	0.135	0.051	0.265	0.216
RetinaNet-s	0.382	0.214	0.215	0.161	0.106	0.534	0.280	0.165	0.144	0.059	0.259	0.221
YOLOv5s	0.413	0.240	0.259	0.161	0.086	0.624	0.327	0.174	0.152	0.077	0.307	0.236
YOLOv5s-s	0.415	0.233	0.237	0.159	0.094	0.569	0.343	0.167	0.136	0.072	0.324	0.233
Proposed XGBoost SG Models												
FRCNN-s + RetinaNet-s	0.398	0.227	0.239	0.178	0.113	0.541	0.289	0.178	0.152	0.063	**0.287**	0.233
RetinaNet-s + YOLOv5s-s	0.412	0.238	0.233	0.165	0.110	0.563	0.341	**0.186**	0.147	0.069	0.325	0.240
FRCNN-s + YOLOv5s-s	0.423	0.241	0.252	0.181	**0.115**	0.554	0.325	0.182	0.153	0.074	0.324	0.246
FRCNN-s + RetinaNet-s + YOLOv5s-s	0.420	0.242	0.250	0.183	0.115	0.557	0.329	**0.188**	0.160	0.069	0.320	0.249
Proposed MLP SG Models												
FRCNN-s + RetinaNet-s	0.398	0.230	0.244	0.181	0.115	0.545	0.291	0.178	0.153	0.064	**0.289**	0.237
RetinaNet-s + YOLOv5s-s	0.413	0.240	0.240	0.169	0.111	0.566	0.343	**0.187**	0.149	0.069	0.324	0.243
FRCNN-s + YOLOv5s-s	0.423	0.242	0.255	0.184	**0.115**	0.558	0.326	0.183	0.153	0.073	0.322	0.248
FRCNN-s + RetinaNet-s + YOLOv5s-s	0.421	0.244	0.255	0.187	0.117	0.559	0.332	**0.187**	0.161	0.069	0.325	0.251

Note: The "-s" means which level-0 models were trained on VisDrone-split. Bold values are the most improved class per method and the bold method is the best performing model. The green and blue cells show the highest metrics.

trained on VisDrone-split only. Since the same level-0 model predictions are used in both XGBoost and MLP SG models, the results of the level-0 models are shown in Table II.

B. *XBGoost Stacked Generalization Models*

The set of three level-0 models were stacked to create four combinations of input to XGBoost. The XGBoost meta-learners that trained on the output of level-0 models were tested on VisDrone-split. Table II shows the average mAP50, average mAP, and mAP per class of the SG models with XGBoost level-1 models trained on each set of level-0 models and trained on VisDrone-split. The most accurate XGBoost SG model from Table II according to mAP of 0.242 came from FRCNN-s + RetinaNet-s + YOLOv5s-s.

All XGBoost SG models on VisDrone-split improved their corresponding level-0 models when measuring mAP, but the SG model that improved its level-0 models the most was FRCNN-s + RetinaNet-s. FRCNN-s + RetinaNet-s may have improved the most because the level-0 metrics were so close. However, it

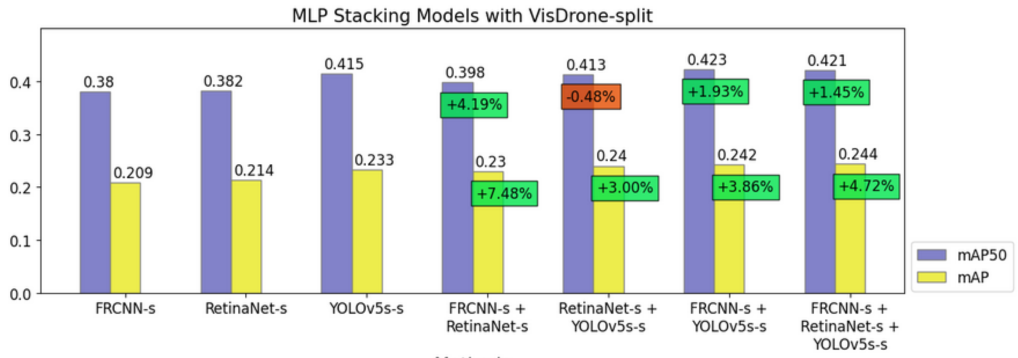

Figure 3. mAP50 and mAP metric comparison between the level-0 models and proposed MLP SG models. The green and red boxes show the positive and negative percent changes between each SG model and their corresponding level-0 model with the highest metric. For example, in case of FRCNN-s + RetinaNet-s, the mAP50 percent change is from 0.382 for RetinaNet-s to 0.398 and the mAP percent change is from 0.214 to 0.230, we take the higher of the two individual corresponding models the RetinaNet-s and the FRCNN-s results to compute the percentage change in SG model.

wasn't always the case that the farther apart the metrics between the level-0 models, the worse the percent change in the level-1 model metrics. When measuring mAP50, a prediction is a true positive if it has an IOU with a ground truth of at least 50%. The true positives must have sufficiently high IOUs to keep up with the increasing IOU thresholds when measuring mAP. The RetinaNet-s + YOLOv5s-s mAP50 metric may not have improved from the mAP50 metrics of its level-0 models because the level-0 true positive predictions were removed during the preprocessing phase and the level-1 model decreased the prediction IOUs with the ground truth from above 50% to below 50%.

C. *Multilayer Perceptron Stacked Generalization Models*

The set of three level-0 models were stacked to create four combinations of input to Multilayer Perceptron (MLP); the same four as with XGBoost. The MLP SG models that trained on the level-0 models were tested on VisDrone-split. Table II shows the average mAP50, average mAP, and mAP per class of the SG models with MLP level-1 models trained on each set of level-0 models trained on VisDrone-split. The best MLP SG model results from Table II according to mAP of 0.244 again came from FRCNN-s + RetinaNet-s + YOLOv5s-s.

Similar to the XGBoost SG models, these MLP SG models tested on VisDrone-split improved on all the combinations of their corresponding level-0 models. The most improved level-1 model was FRCNN-s + RetinaNet-s as shown in Figure 3, whose every class mAP, besides pedestrian, had a positive percent change. The percent change for each class was calculated from the best class metric from its level-0 models.

VII. Conclusion and Future Work

Eight SG models were proposed in this paper. The results of the previous section showed that every proposed SG model improved upon their corresponding state-of-the-art level-0 models, so an SG model was the best performing model using either XGBoost or MLP level-1 models. The most improved SG model was FRCNN-s + RetinaNet-s with MLP, whose mAP improved 7.48% from RetinaNet. FRCNN-s + RetinaNet-s + YOLOv5s-s with MLP was the model with the highest mAP, whose mAP50 and mAP were 0.421 and 0.244, respectively. Furthermore, their mAP improved 4.72% from YOLOv5s. Thus, FRCNN-s + RetinaNet-s + YOLOv5s-s is the best performing of all the proposed SG models.

Since meta-learning improved every object detection model using SG, future work would include using meta-learning with SG with larger and more accurate object detection models to test if better performing models can still be improved upon. Other methods can also be used to improve object detection models by combining their predictions without meta-learning. The same steps can be used to create the meta-datasets of predictions, but instead of meta-learning, a non-meta-learning method can be applied. One such method could be Decision Fusion [8], which combines the predictions of multiple models to create a subset of predictions more accurate than the individual models. This can be used to test how meta-learning performs compared to non-meta-learning methods.

References

[1] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, 2015.

[2] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 42, no. 2, pp. 318-327, 2020.

[3] G. Jocher, A. Chaurasia, J. Borovec, A. Stoken, Y. Kwon, J. Fang and e. al, "yolov5," [Online]. Available: https://github.com/ultralytics/yolov5. [Accessed Nov 2022].

[4] J. Redmon and A. Farhadi, *YOLOv3: An Incremental Improvement,* arXiv:1804.02767, 2018.

[5] X. Zhou, D. Wang and P. Krähenbühl, *Objects as Points,* arXiv:1904.07850, 2019.

[6] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu and e. al, "Detection and tracking meet drones challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 44, no. 11, pp. 7380-7399, 2022.

[7] B. Liu, H. Luo, H. Wang and S. Wang, "YOLOv3_ReSAM: A Small-Target Detection Method," *Electronics,* vol. 11, no. 10, 2022.

[8] J. Xu, Y. Li and S. Wang, "AdaZoom: Towards Scale-Aware Large Scene Object Detection," *IEEE Transactions on Multimedia,* 2022.

[9] D. Du, P. Zhu, L. Wen, X. Bian, H. Lin, Q. Hu and e. al, "VisDrone-DET2019: The Vision Meets Drone Object Detection in Image Challenge Results," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019.

[10] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li and e. al, "Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection," in *Advances in Neural Information Processing Systems*, 2020.

[11] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, *YOLOv4: Optimal Speed and Accuracy of Object Detection,* arXiv:2004.10934, 2020.

[12] C.-Y. Wang, L. H.-Y. Mark, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh and I.-H. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

[13] J. Terven and D. Cordova-Esparza, *A Comprehensive Review of YOLO: From YOLOv1 and Beyond,* arXiv:2304.00501, 2023.

[14] "Ultralytics YOLOv5 Architecture," 2023. [Online]. Available: https://docs.ultralytics.com/yolov5/tutorials/architecture_description/. [Accessed Sep 2023].

[15] S. Liu, L. Qi, Q. Haifang, J. Shi and J. Jiaya, "Path Aggregation Network for Instance Segmentation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[16] K. He, X. Zhang, S. Ren and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in *Computer Vision – ECCV 2014*, 2014.

[17] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye and D. Ren, "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression," in *The AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[18] "Data format," [Online]. Available: https://cocodataset.org/#format-data. [Accessed Jan 2023].

[19] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto and E. A. B. da Silva, "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit," *Electronics,* vol. 10, no. 3, 2021.

[20] "Detection Evaluation," [Online]. Available: https://cocodataset.org/#detection-eval. [Accessed Jan 2023].