Novel Deep Learning Based Stacking Models for Multiclass Object Recognition in Drone Images

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

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

Abstract: Object detection models might not be as accurate as they could be when trained on particular datasets. Learning from the predictions of multiple object detection models can help improve their detection accuracy. The learning of predictions to create a new set of predictions is called meta-learning, and the method to meta-learn is Stacking Generalization, or just stacking. This paper proposed Stacking models to improve several multiclass object detectors trained the Visdrone2019-DET dataset and find the most accurate model. Four models were trained –YOLOv5x, YOLOv7x, YOLOv8x, and EVA02L. The models trained on one variation of the dataset. The meta-models trained from the object detection predictions were XGBoost and Multilayer Perceptron (MLP). Nearly all the Stacking models using either XGBoost or MLP meta-models improved the mAP of the object detection models trained on VisDrone-split. The most improved Stacking models in both sets used MLP meta-models. The most accurate model was YOLOv8x + EVA02L MLP on VisDrone-split with an mAP50 of 0.609, with an mAP of 0.389, and whose mAP improved from EVA02L by 4.01%.

Keywords—Deep learning; Stacking Generalization; Meta-learning; VisDrone2019-DET YOLOv5; YOLOv7; YOLOv8; EVA02

I. INTRODUCTION

Multiclass object detection in drone images can learn to perform a wide range of tasks including creating a map of an area, predicting traffic, and monitoring wildlife. Drone images include real-world data, as opposed to synthetic data, which has it challenges. Real-world data can create a more realistic distribution of the objects to be detected, resulting in a distribution that can be highly imbalanced. This means some objects may be detected with high accuracy while others are not. Ensemble methods are often being improved to increase accuracy of multiple machine learning models without increasing the cost of time and effort significantly. There are ensemble methods to combine multiple object detection, such as the Affirmative method, the Unanimous method, and the Consensus method [1]. The Affirmative method may have too many false positive predictions, but the other two methods cannot be sure take advantage of every true positive prediction from each model. So, the goal of this research was to improve state-of-the-art object detection models to find the most accurate model using meta-learning.

Stacking Generalization, or stacking, is an ensemble method that creates a new set of predictions from a set of models as defined in [2]. Stacking uses meta-learning which doesn't use a simple formula to create a new set of final predictions. Meta-learning, or learning to learn, is used to improve an already trained machine learning model. The inputs of the meta-learning model are the predictions or metadata of a trained model, then the meta-learner learns from the given predictions or metadata to make more accurate predictions. Stacking uses a meta-learner by training on predictions of at least two machine learning model, also known as base or level-0 models. The meta-model, or level-1 model, creates a new set of predictions from the predictions of multiple base models.

There is more research available using stacking to perform classification and regression tasks than object detection tasks. So, this research was not as straightforward as finding an API or library to automate the process. There are APIs and libraries which can be used to train a meta-classifier or meta-regressor. Scikit-Learn, for example, has a stacking classifier method and a stacking regressor method in its ensemble module. These methods take a list of level-0 models and a level-1 model as input and output a trained stacking model. The level-1 models used in this research were XGBoost regressors and Multilayer Perceptron (MLP) regressors to stack multiple object detection models. Although a regression model was used as a level-1 model, the stacking regressor method in Scikit-Learn could not be used. The output of the level-0 models needed to be processed so that a regressor could be trained as a level-1 model.

The rest of the paper first discusses related works using similar level-0 models, the same dataset, or level-1 models in Chapter 2. Next, datasets used to train four level-0 models and test the Stacking models are explained in Chapter 3. The models were YOLOv5x [3], YOLOv7x [4], YOLOv8x [5], and EVA02L [6]. Chapter 4 describes the methodology and architecture of the level-0 models, Stacking models, and level-1 models. Chapters 5 explains the setup of the experiments, and 6 explains the results of the level-0 and Stacking models. Finally, Chapter 7 contains the conclusion and the focus of future research.

II. Related Work

Meta-learning for image classification has been more thoroughly researched than meta-learning for object detection. Since image classification and object detection are related tasks, it's important to discuss meta-learning for image classification before meta-learning for object detection. An article published in 2023 in the MDPI Journal proposed four decomposition-based meta-learning methods in four different datasets with between 3 and 8 classes as described in [7]. Two of the four methods improved the baseline multiclass classifier the most. The first of the best methods divide the multiclass classification problem into different classifiers for each class in a dataset. Each classifier was trained as a binary classifier, then combined into the final predictions using a generalizer. The second of the best methods used Stacking Generalization. Binary classifiers were trained for each class independently and one multiclass classifier was trained for the level-0 classifiers. The level-1 classifier is a generalizer. So, stacking for classification tasks perform well.

Meta-learning is not only used for classification, but also regression. A Multi-target Stacked Generalisation (MTSG) method was proposed in an article published in the Chemometrics and Intelligent Laboratory Systems Journal in 2021 [8]. The level-0 regression models were trained to predict ten soil fertility parameters. MTSG is similar to stacking, except instead of a meta-model predicting one target, multiple meta-models predict multiple targets. For d targets and j level-0 models, each level-0 model will predict d targets for an input set X. These j*d predictions will go through a filter F to preserve only the best predictions from each target. For the level-0 models, there is one meta-model for each of the d targets. The predictions after passing through F will be inputted into these meta-models to get the final output predictions. The next paper uses a similar method of MTSG to predict multiple targets but applies to single class object detection.

A paper published in 2022 in the Scientific Reports Journal provided a stacking-based framework for improving the detection of a single class of polyps, called StackBox [9]. Five models, including YOLOv5 [3], were combined through their final predictions to create a new set of final predictions. For each level-1 model, there were four datasets for each coordinate. The best regression models that were used as level-1 models were Linear Regression, Random Forest, Gradient Boosting, and XGBoost. Even though the article does not use VisDrone2019, it explains which learners are best for meta-learning. StackBox was able to significantly improve the average precision from the level-0 models. The next paper expands upon the meta-learning task that StackBox completes by detecting objects in the multiclass dataset, VisDrone2019.

In 2022, an article published in the International Journal of Computational Intelligence Systems explained how a global and local ensemble network was used to increase the accuracy of object detection model predictions [10]. Three models were combined using a global and local ensemble network (GLE-NET). In general, the global ensemble network combines detection boxes, and the local ensembled network creates a new confidence score from these combined detection boxes. This helps to visualize the process of creating a new prediction box from base prediction boxes as separate from the process of creating a new confidence score. GLE-NET improved the accuracy by combining two or more models together over what those selected models can do alone. This research merges GLE-Net and StackBox by using a simpler algorithm to improve upon multiple models trained on a dense multiclass dataset, while also using higher average precision models as the level-0 model

III. Dataset Description

A. *VisDrone*

Visdrone2019-DET, or VisDrone for short, a drone image dataset created at Tianjin University in China as explained in [11] were used to train the object detection models. The annotation format of the VisDrone dataset was that for each image, the coordinates in pixels for each detection are "X min, Y min, width, height." The images had a wide range of sizes, up to 1500 by 2000 pixels. So, they were resized to 1500 by 1500 pixels. An example from the resized dataset is shown Figure 1. The image shown was originally 1080 by 1920 pixels, so the width got smaller, and the height was increased. This explains why the objects look compressed or stretched. The number of images used for training was 6,471 and the number of images used for testing was 548.

Figure 1. A resized 1500 by 1500 pixel image from the VisDrone *[11]* dataset. The ground truths in the image are pedestrian, people, bicycle, car, and motor.

There are ten main classifications– pedestrian, people, bicycle, car, van, truck, tricycle, awning-tricycle, bus, and motor. There are a total of 343,205 training annotations in VisDrone. There are also some inconsistencies with ground truth detections. There are objects that are a part of the ground truth, but other objects of the same size and class are which are not a part of the ground truth. For example, in the upper left-hand corner of Figure 1, the cars parked in the parking lot and those parked on the street are about the same size and almost as crowded. However, some cars in the parking lot are not a part of the ground truth, while cars parked on the street are.

B. VisDrone-split

To increase accuracy and remove some inconsistencies in the ground truth, each 1500 by 1500 pixel image from VisDrone was later cropped to nine equal sized sub-images. The images in VisDrone-split do not overlap with their adjacent images. This means the number of training annotations in the new dataset, VisDrone-split, is 295,677. Since the only objects that were trained to be identified are objects that are completely in the frame, the annotations for cropped objects were removed. The images without annotations from both image inconsistencies and removed cropped annotations were also removed. This means there are fewer instances of each class due to all the cropped objects in each image.

IV. Rearch Methodology

A. Base Models (Level-0): YOLOv5, YOLOv7, YOLOv8

The level-0 object detection models were trained on the VisDrone dataset variation, VisDrone-split. The four level-0 models are YOLOv5x, YOLOv7x, YOLOv8x, and EVA02L. YOLO models can be characterized by their backbone, neck, and head. YOLOv5 [3] has the same backbone as YOLOv4. The neck is Spatial Pyramid Pooling Fusion (SPPF) [12] and CSP-Path Aggregation Network (CSP-PAN) [13], [14]. CSP-PAN uses the features that were learned in the backbone and preserves the information into the lower layers. The activation function is Sigmoid Linear Units (SiLU), which is defined as the following:

$$silu(x) = x/(1 + e^{-x}). \quad (1)$$

By default, the optimizer is Stochastic Gradient Descent (SGD). The model head is the same as YOLOv3 [15] with has three output heads, which correspond to bounding boxes, objectness, and class. The three output heads and loss functions correspond to bounding boxes, objectness, and class. Location loss uses Complete Intersection over Union (CIOU) loss defined in [16]. Objectness and class loss both use Binary Cross Entropy (BCE) loss. 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:

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

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.

YOLOv7 and YOLOv8 architectures are similar to YOLOv5 with a few differences. YOLOv7 [4] modified Efficient Layer Aggregation Network (ELAN) [17] as the backbone, called Extended ELAN. CSPSPP and PAN form the neck, followed by YOLOR [18] as the head. The SiLU activate function is used and, by default, SGD with momentum 0.937 is used for the optimizer. The image augmentation that is used includes mosaic, where portions of images are combined into one. There are three losses from the output head for bounding boxes, objectness, and class. YOLOv7 used the same losses for bounding boxes, objectness, and class as YOLOv5. The backbone for YOLOv8 [5] is a modified YOLOv4 backbone with a Feature Pyramid Network (FPN) [19], and the neck is PAN. One output head identifies the bounding boxes, objectness, and class. The loss functions are CIOU with Distribution Focal Loss (DFL) [20] for bounding box loss and BCE for class loss. DFL is a specific form of Focal Loss which is calculated using a weighted cross entropy where there is more weight on predictions closer to the ground truths.

B. Base Models (Level-0): EVA02L

One of the two EVA methods is EVA02 [6]. EVA02L uses Transform Vision (TrV), a type of Vision Transformer (ViT) [21]. TrV follows Contrastive Language-Image Pre-training (CLIP) [22] to pretrain the TrV. The improved ViT include multi-head self-attention (MHSA) layers + 2D rotary position embedding (RoPE) [23], random weight initialization in a

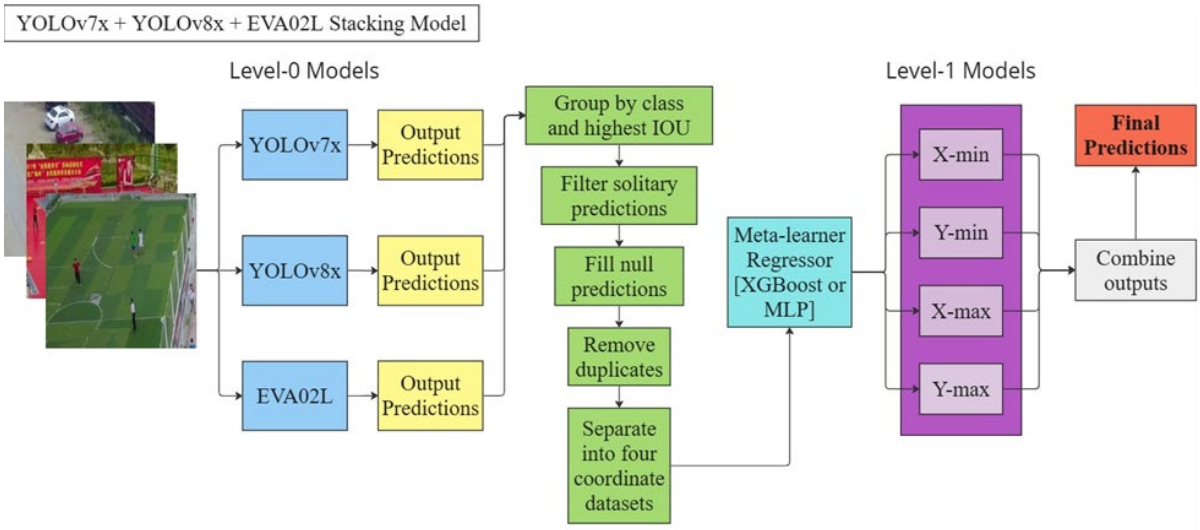

Figure 2. One of the proposed Stacking models- YOLOv7x + YOLOv8x + EVA02L. For an input image dataset, the output of YOLOv7x, YOLOv8x, and EVA02L level-0 models are used as the dataset to train the meta-learner regressors. But first, the data must be combined and processed so that predictions with the highest IOU from each level-0 model are grouped with the same class, any ungrouped predictions are removed, and incomplete groups are filled. Once the data is separated based on the four coordinates, each dataset can be used to train an XGBoost or MLP Meta-learner regressor. Each of the four resulting level-1 models provide a single coordinate for a new set of final predictions. Combining the coordinates provides the final output.

feedforward network (FFN) with a SiLU activation function, and a normalization layer further explained in [6]. Before the inputs can be used in the MSHA layer, they need to be separated into tokens, where input embeddings convert them into vectors. The position of each token is important to the relationship between the elements, so positional embeddings include information about each position of the tokens to create the query, key, and value vectors. So, RoPE modifies only the query and key vectors.

C. Proposed Stacking Models

After training four level-0 models, the stacking ensemble method was implemented to combine multiple object detectors. The meta-learning models learned from the output of several combinations of the level-0 models on the VisDrone datasets. The outputs are all combinations of YOLOv5x, YOLOv7x, YOLOv8x, and EVA02L trained on VisDrone-split. The meta-regressors trained as level-1 models were XGBoost and Multilayer Perceptron (MLP). Four datasets were created from the output of the level-0 models, one for each coordinate to meta-learn the detection boxes, like StackBox [9]. Even though the datasets used in this research were multiclass, meta-learning the class label may have created a problem in the end results since the dataset is imbalanced. So, each class prediction was not used in each of the meta-learner datasets. To create the datasets to train the level-1 models, the data was first prepared.

The first step was converting the images and annotations into tables for each image in the dataset. The features of the table are the bounding box, confidence score, class label, image number, and model name. Next, ground truths were paired with predictions of the same class with the highest Intersection over Union (IOU) from each level-0 model. If there were no predictions from a model to be paired with a ground truth, then a null value is used for that prediction. The third step was to drop a ground truth from training if only null predictions were paired with it. For each ground truth, if there were still any null predictions, it would be filled with the prediction from another model with the highest IOU. The last step was to create the four datasets for each meta-learner, which contains the detection box coordinates of each level-0 model, their confidences, and the IOUs between each other.

The learners trained as meta-learners were XGBoost and MLP. Since each level-1 regressor needed to learn the coordinates of the detection box, four models were trained after the dataset was separated into four datasets for each coordinate of a detection box - X min, Y min, X max, and Y max. To train each XGBoost, four-fold cross-validation was used to find the optimal set of hyperparameters. To train each MLP, the number of layers and the number of nodes had to be determined. Each MLP had two deeply connected layers because having one or three layers did not improve the results. However, the number of nodes in each layer of each MLP was determined by the number of level-0 models that were being combined. This is because more level-0 models mean more features. The number of nodes in the first layer was number of features plus one and output layer had one node. The number of nodes in the second layer was between the number of nodes in the first and last layer. There were no signs of significant overfitting, so there was no need for a dropout layer or any other techniques to reduce overfitting.

The level-0 output predictions from the test dataset were grouped differently than the level-0 output predictions from the training dataset. Each level-0 model was paired with the other level-0 models instead of the ground truth being paired with the level-0 models. Any prediction without a group was removed from testing. Any other missing predictions left were filled in the same manner as for the training dataset. The next step was to remove the test dataset's duplicate rows. The same predictions were not always paired together so, rows with repeated predictions were also removed. Next, the test dataset was split into the four coordinate datasets as inputs to the four separate level-1 models. Lastly, the outputs were combined to get a finalized set of predictions to calculate the metrics to compare the results of the level-0 and level-1 models. Figure 2 shows one of the Stacking models.

V. Experiment Setup

A. Hardware and Software

There were two GPUs used in this research. All the level-1 models were trained using the NVIDIA GeForce RTX 3060 with 6 GB. The level-0 models that were trained using the NVIDIA Titan XP with 12 GB were YOLOv7x and YOLOv8x. The cloud computing platform, Amazon Web Services (AWS), was used to train the largest models – YOLOv5x and EVA02L. Two instances of Amazon Sagemaker were created with a NVIDIA A10G Tensor Core GPU with 24 GB. YOLOv5x trained with one GPU and EVA02L trained with four GPUs. Even though the level-0 and level-1 models were trained using GPUs, the meta-data was preprocessed using the CPU.

B. Model Evaluation Metrics

An object detection metric is Average precision (AP), which is the area under the precision-recall curve [24]. The precision and recall are calculated as the following:

$$Precision = \frac{TP}{FP + FP} = \frac{TP}{all\ detections} \quad (4)$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths} \quad (5)$$

where TP is true positive, FP is false positive, and FN is false negative. A prediction is considered TP if its Intersection over Union (IOU) with a ground truth is greater than given threshold. The mean AP (mAP) is the average of each class AP. The most used IOU thresholds mAP uses are 50%, 75%, and 50:5:95%. The first two mAPs with IOU thresholds 50% and 75% are labeled as mAP50 and mAP75, respectively. mAP50:5:95 is the average of the ten mAPs from an IOU of 50% to 95% with increments of 5%.

TABLE I. Stacking Model Runtime in Minutes Based on the Number of Level-0 Models

Number of Level-0 Models	2	3	4
XGBoost Stacking Runtime (min)	51	93	155
MLP Stacking Runtime (min)	60	100	171

TABLE II. Proposed XGBoost Stacking Models mAP per Class and mAP50 With VisDrone-Split

Method	mAP50	mAP	pedes-trian	people	bicycle	car	van	truck	tricycle	awning-tricycle	bus	motor
Level-0 Models												
YOLOv5x-s	0.560	0.337	0.340	0.236	0.195	0.631	0.426	0.300	0.242	0.170	0.508	0.326
YOLOv7x-s	0.563	0.341	0.337	0.227	0.206	0.629	0.428	0.316	0.262	0.147	0.528	0.330
YOLOv8x-s	0.579	0.362	0.367	0.262	0.215	0.656	0.453	0.337	0.285	0.172	0.514	0.357
EVA02L-s	0.587	0.374	0.358	0.281	0.247	0.604	0.437	0.395	0.309	0.213	0.544	0.352
Proposed XGBoost Stacking models With Only YOLOs												
YOLOv5x + YOLOv7x-s	0.577	0.353	0.344	0.241	0.214	0.630	0.442	0.331	0.267	**0.180**	0.539	0.342
YOLOv5x + YOLOv8x-s	0.586	0.361	0.355	0.257	0.221	0.635	0.452	0.337	0.279	**0.189**	0.533	0.351
YOLOv7x + YOLOv8x-s	0.587	0.364	0.355	0.254	**0.228**	0.639	0.450	0.351	0.290	0.176	0.542	0.358
YOLOv5x + YOLOv7x + YOLOv8x-s	0.590	0.366	0.358	0.256	0.227	0.639	0.456	0.347	0.285	**0.188**	0.544	0.357
Proposed XGBoost Stacking models With EVA02L												
YOLOv5x + EVA02L-s	0.603	0.377	0.354	0.277	0.247	0.626	**0.456**	0.377	0.306	0.213	0.555	0.359
YOLOv7x + EVA02L-s	0.605	0.382	0.361	0.277	0.256	0.629	0.453	0.383	**0.324**	0.209	0.563	0.366
YOLOv5x + YOLOv7x + EVA02L-s	0.605	0.382	0.365	0.277	0.253	0.634	**0.460**	0.378	0.317	0.212	0.555	0.369
YOLOv5x + YOLOv8x + EVA02L-s	0.605	0.383	0.370	0.283	0.252	0.637	0.461	0.382	0.312	0.211	0.550	**0.370**
YOLOv5x + YOLOv7x + YOLOv8x + EVA02L-s	0.604	0.384	0.371	0.281	0.252	0.640	0.464	0.378	0.316	0.211	0.553	**0.374**
YOLOv7x + YOLOv8x + EVA02L-s	0.606	0.385	0.373	0.283	0.254	0.640	0.459	0.386	0.322	0.208	0.553	**0.373**
YOLOv8x + EVA02L-s	0.608	0.385	0.365	0.282	0.254	0.634	0.461	0.394	**0.324**	0.209	0.553	0.370

Note: The "-s" means the 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 in the table. The green and blue cells show the highest metrics.

C. *Base Models (Level-0) Parameter Setups*

The four base models which were trained were YOLOv5x [3], YOLOv7x [4], YOLOv8x [5], and EVA02L [6]. Due to time constraints and computational limitations for the larger models, the four models were only trained on the VisDrone-split dataset. They all used the default parameters. The batch size and number workers were based on the space available. The main complication was overfitting. YOLOv8x had a 20% dropout rate but, adding more image augmentation helped slow done overfitting the most across all models. The augmentations that were applied were image rotation, translation, scale, flip across both x and y axes, mosaic, and mixup. The training stopped when the training and testing mAP50 results differed more than 0.05.

D. *Meta Model (Level-1) Parameter Setups*

Before training and testing XGBoost and Multilayer Perceptron (MLP) level-0 models, the final predictions for the training and testing datasets were saved from all four level-0 models. An IOU threshold of 0.0 or 0.1 was put on the overlap of each pair of base models. This was because some of the objects in the images were close together, so the IOU threshold attempts to make sure that the paired predictions were detecting the same object. Although the training predictions were paired using the ground truth, that was not the case for the testing predictions. So, the meta-learners were trained and tested using predictions that have some overlap between them. Table I shows the time to preprocess the data, train, and test was based on the number of level-0 models for Stacking models. The average runtime of the Stacking models with two and three level-0 models including EVA02L was calculated to find the runtime for two and three level-0 models, respectively. The runtimes for YOLOv5x + YOLOv7x + YOLOv8x + EVA02L XGBoost and MLP are the runtimes for four level-0 models. Each runtime is rounded to the nearest whole minute.

As mentioned in the previous chapter, four-fold cross-validation was used to find the optimal set of hyperparameters to train each XGBoost. The hyperparameters that were tested to find the optimal set of measurements were n estimators for the number of trees, max depth for the maximum depth of each tree, and learning rate. The default values for each hyperparameter were 100 estimators, a max depth of 6, and a learning rate of 0.3. All the models for any combination of level-0 models used a

Figure 3. Metric comparison between EVA02L and proposed XGBoost Stacking models. The green boxes show the positive percent changes between each Stacking model and EVA02L. The percent changes shown are all from the metric scores of EVA02L

max depth of either 4, 6, or 8, and they all used a learning rate of 0.05. The number of estimators for most models for all combinations of level-0 models used 200 estimators.

To train and to more efficiently find the optimal MLP, 0.25 of the training was used as a validation dataset to find the models with the lowest mean squared error. Each MLP used the Adam optimizer, a learning rate of 0.001, the number of features plus one as the number of nodes in the first layer and one node for the output layer. The number of nodes in the second layer was between the number of nodes first layer and output layer. Each model ran for a maximum of 100 epochs with early stopping and a batch size of 128 with eight workers. Early stopping monitored the validation loss with a patience of ten epochs as a stopping point.

VI. Results and Discussion

The results of Stacking models with level-1 models trained on the level-0 models are analyzed. Between the level-1 models, Multilayer Perceptron (MLP) provides the highest average precision results.

A. Level-0 Models: YOLOv5x, YOLOv7x, YOLOv8x, EVA02L

The metric results of YOLOv5x, YOLOv7x, YOLOv8x, and EVA02L are shown in Table II, where "-s" is for models trained on VisDrone-split. EVA02L had the highest mAP50 and mAP scores of the level-0 models. Class 3 (car) had the highest mAP class score for every level-0 model. However, EVA02L had the lowest scoring car class, 0.052 below YOLOv8x's car mAP. EVA02L had the highest scoring class 5 (truck), whose mAP is 0.058 above YOLOv8x's truck mAP. Given the differences, the car and truck mAPs between models affected some Stacking models.

B. XGBoost Stacking Models

The set of base models were stacked to create eleven combinations of models with XGBoost. The XGBoost Stacking models that trained on the output of level-0 models were tested on VisDrone-split. Table II show the average mAP50, average mAP, and mAP per class of all the Stacking models with XGBoost trained on each combination of level-0 models. The XGBoost Stacking models tested on VisDrone-split improved on nearly all the combinations of the level-0 models. The most improved XGBoost Stacking model in terms of mAP was YOLOv5x-s + YOLOv7x-s which had both their mAP50 and mAP increase from YOLOv7x. The most improved class was the least accurate class, awning-tricycle. The only negative percent change was the most accurate class, car, but the car class for nearly every Stacking model had no improvement. Figure 3 shows percent change from EVA02L's metrics to their XGBoost Stacking model's metrics in the green boxes. The most accurate XGBoost Stacking model is YOLOv8x-s + EVA02L-s since it has the highest mAP50 and mAP. The most improved class was tricycle but, the class with the smallest percent change was car. The class improvements of YOLOv8-s + EVA02L-s XGBoost were less significant than YOLOv5x-s + YOLOv7x-s XGBoost, and the class deteriorations of YOLOv8-s + EVA02L-s XGBoost were more significant than YOLOv5x-s + YOLOv7x-s XGBoost. This is why the best XGBoost model in terms of mAP was also not the most improved.

C. MLP Stacking Models

Next, the set of four level-0 models were stacked to create eleven combinations of models with MLP, the same as with XGBoost. The MLP Stacking models that trained on the output of level-0 models were tested on VisDrone-split. Table III show the average mAP50, average mAP, and mAP per class of the most improved and best performing MLP Stacking models trained on each set of level-0 models trained on VisDrone-split. The best performing model overall in the table according to mAP50 and mAP came from YOLOv8x + EVA02L-s MLP, the same as with an XGBoost meta-model.

These MLP Stacking models tested on VisDrone-split improved on all the combinations of the level-0 models when measuring both mAP50 and mAP. The most improved model in terms of mAP in Table III is YOLOv5x-s + YOLOv7x-s MLP. The most improved class was awning tricycle. This model was one of the few Stacking models that experienced an increase in the car class from their level-0 models. Every class from the YOLOv5x-s + YOLOv7x-s MLP model improved more than every class from the YOLOv5x-s + YOLOv7x-s XGBoost model. Figure 4 shows percent change from EVA02L's metrics to their MLP Stacking model's metrics. Each green box shows a

TABLE III. Proposed MLP Stacking Models mAP per Class and mAP50 With VisDrone-Split

Method	mAP50	mAP	pedes-trian	people	bicycle	car	van	truck	tricycle	awning-tricycle	bus	motor
Level-0 Models												
YOLOv5x-s	0.560	0.337	0.340	0.236	0.195	0.631	0.426	0.300	0.242	0.170	0.508	0.326
YOLOv7x-s	0.563	0.341	0.337	0.227	0.206	0.629	0.428	0.316	0.262	0.147	0.528	0.330
YOLOv8x-s	0.579	0.362	0.367	0.262	0.215	0.656	0.453	0.337	0.285	0.172	0.514	0.357
EVA02L-s	0.587	0.374	0.358	0.281	0.247	0.604	0.437	0.395	0.309	0.213	0.544	0.352
Proposed MLP Stacking Models With Only YOLOs												
YOLOv5x + YOLOv7x-s	0.578	0.358	0.356	0.247	0.219	0.637	0.447	0.334	0.271	**0.182**	0.544	0.348
YOLOv5x + YOLOv8x-s	0.587	0.366	0.367	0.263	0.223	0.644	0.456	0.339	0.284	**0.189**	0.541	0.358
YOLOv7x + YOLOv8x-s	0.588	0.369	0.366	0.262	**0.230**	0.648	0.455	0.352	0.294	0.178	0.543	0.365
YOLOv5x + YOLOv7x + YOLOv8x-s	0.591	0.369	0.366	0.262	0.230	0.643	0.456	0.347	0.289	**0.188**	0.547	0.361
Proposed MLP Stacking Models With EVA02L												
YOLOv5 + EVA02L-s	0.608	0.384	0.369	0.284	0.253	0.633	**0.461**	0.383	0.312	0.216	0.562	0.367
YOLOv5x + YOLOv7x + YOLOv8x + EVA02L-s	0.605	0.386	0.380	0.284	0.257	0.642	0.463	0.380	0.318	0.210	0.550	**0.378**
YOLOv7x + YOLOv8x + EVA02L-s	0.606	0.386	0.377	0.283	0.256	0.642	0.462	0.385	0.322	0.209	0.549	**0.374**
YOLOv5x + YOLOv8x + EVA02L-s	0.607	0.387	0.380	0.288	0.254	0.642	0.464	0.384	0.316	0.212	0.555	**0.376**
YOLOv5 + YOLOv7x + EVA02L-s	0.608	0.387	0.377	0.285	0.256	0.639	0.461	0.381	0.319	0.212	0.556	**0.375**
YOLOv7x + EVA02L-s	0.608	0.388	0.375	0.286	0.261	0.637	0.460	0.389	**0.329**	0.212	0.563	0.373
YOLOv8x + EVA02L-s	0.609	0.389	0.375	0.290	0.257	0.640	0.463	0.394	**0.324**	0.211	0.559	0.374

Note: The "-s" means the 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 in the table. The green and blue cells show the highest metrics.

positive percent change. The most efficient model with the most improved mAP50 metric was YOLOv8x + EVA02L-s MLP. The YOLOv8x + EVA02L-s MLP model had all but three of their classes improved from its level-0 models, including the car and truck classes. However, each class in the YOLOv8x + EVA02L-s MLP model improved at least as much as its XGBoost counterpart. The YOLOv8x + EVA02L-s MLP model's most improved class was tricycle and the smallest percent change was car. The decrease in the car class from YOLOv8x was expected since YOLOv8x had a significantly better car mAP than EVA02L.

VII. Conclusion and Future Work

The proposed Stacking models used XGBoost and Multilayer Perceptron (MLP) meta-models to learn from and improve multiple multiclass object detectors. The XGBoost and MLP level-1 models were trained on the predictions of two sets of level-0 models each. Improvements were seen in nearly all XGBoost and all MLP Stacking models. The most improved Stacking model was the YOLOv5x-s + YOLOv7x-s with an MLP level-1 model whose mAP improved 4.99% from YOLOv7x. The model with the highest mAP between both level-0 sets was the YOLOv8x-s + EVA02L-s MLP model with an mAP50 of 0.609, with an mAP of 0.389, and whose mAP improved 4.01% from EVA02L. Therefore, the YOLOv8x-s + EVA02L-s MLP Stacking model is the best performing models of all proposed models.

In order to further refine and examine these models, future work should include updating the process for the Stacking and selecting other metrics. First, the dataset used could be another version of the VisDrone dataset to improve the imbalanced number of objects per class. Second, new data preprocess techniques should be researched to better minimize the number of false positives. Next, other models could also be used as meta-learners trained from object detection predictions, such as Support Vector Machine or Cascade Forest [25]. Lastly, other metrics should be included in the results to get a better picture of how the Stacking models improved from their level-0 models. These metrics could be based on the size of the object, such as mAP-large, mAP-medium, and mAP-small. So, there are many paths on which to move forward to improve on multiclass object detection models.

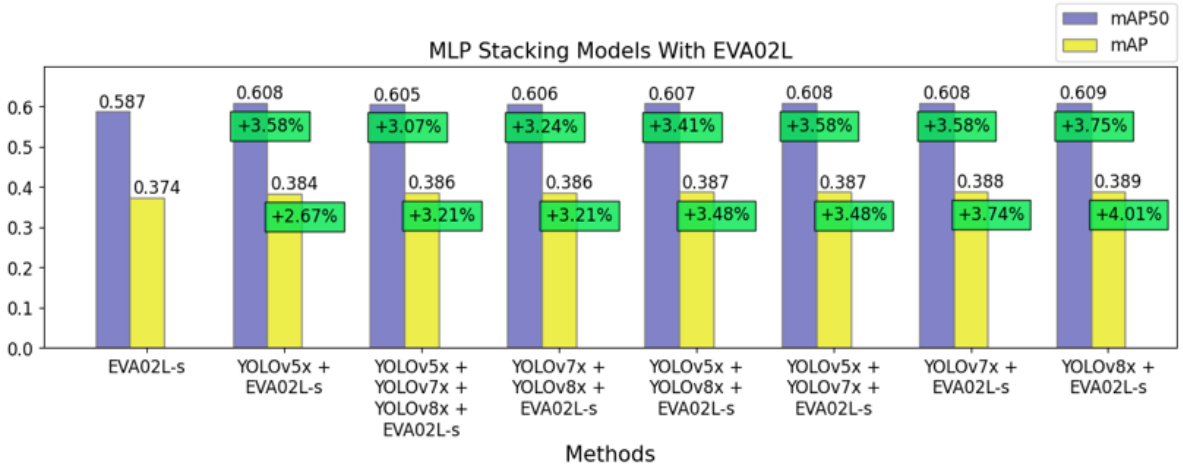

Figure 4. Metric comparison between EVA02L and proposed MLP Stacking models. The green boxes show the positive percent changes between each Stacking model and EVA02L. The percent changes shown are all from the metric scores of EVA02L.

REFERENCES

[1] A. Casado-Garcia and J. Heras, "Ensemble Methods for Object Detection," in *Frontiers in Artificial Intelligence and Applications (ECAI 2020)*, vol. 325, G. D. Giacomo, A. Catala, B. Dilkina, M. Milano, S. Barro, A. Bugarín and J. Lang, Eds., Logrono, La Rioja: IOS Press Ebooks, 2020, pp. 2688-2695.

[2] K. M. Ting and I. H. Witten, "Issues in Stacked Generalization," *Journal of Articial Intelligence Research,* vol. 10, pp. 271-289, 1999.

[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] C.-Y. Wang, A. Bochkovskiy and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA, 2023.

[5] D. Reis, J. Kupec, J. Hong and A. Daoudi, *Real-Time Flying Object Detection with YOLOv8,* arXiv:2305.09972, 2023.

[6] Y. Fang, Q. Sun, X. Wang, T. Huang, X. Wang and Y. Cao, *EVA-02: A Visual Representation for Neon Genesis,* arXiv:2303.11331, 2023.

[7] A. Vogiatzis, S. Orfanoudakis, G. Chalkiadakis, K. Moirogiorgou and M. Zervakis, "Novel Meta-Learning Techniques for the Multiclass Image Classification Problem," *Sensors,* vol. 23, no. 1, 2023.

[8] E. J. Santana, F. Rodrigues dos Santos, S. M. Mastelini, F. L. Melquiades and S. Barbon Jr, "Improved prediction of soil properties with multi-target stacked generalisation on EDXRF spectra," *Chemometrics and Intelligent Laboratory Systems,* vol. 209, p. 104231, 2021.

[9] C. Albuquerque, R. Henriques and M. Castelli , "A stacking-based artificial intelligence framework for an effective detection and localization of colon polyps," *Scientific Reports,* vol. 12, 2022.

[10] J. Liao, Y. Liu, Y. Piao, J. Su, G. Cai and Y. Wu, "GLE-Net: A Global and Local Ensemble Network for Aerial Object Detection," *International Journal of Computational Intelligence Systems,* vol. 15, no. 2, 2022.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] C.-Y. Wang, H.-Y. M. Liao and I.-H. Yeh, *Designing Network Design Strategies Through Gradient Path Analysis,* arXiv:2211.04800, 2022.

[18] C.-Y. Wang, I.-H. Yeh and H.-Y. M. Liao, *You Only Learn One Representation: Unified Network for Multiple Tasks,* arXiv:2105.04206, 2021.

[19] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[20] 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.

[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai and T. Unterthiner, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,* arXiv:2010.11929, 2021.

[22] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal and e. al, "Learning Transferable Visual Models From Natural Language Supervision," in *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[23] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen and Y. Liu, *RoFormer: Enhanced Transformer with Rotary Position Embedding,* arXiv:2104.09864, 2022, pp. 418-434.

[24] 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.

[25] Z.-H. Zhou and J. Feng, "Deep Forest," *National Science Review,* vol. 6, no. 1, pp. 74-86, 2019.