Introduction

The base-model selector for the TANGO framework currently iterates through a list of object-detection models and picks the ‘best’ model. The current method of picking the best model involves first creating a small subset of images from the COCO dataset. The subset of images is run on all candidate models to obtain latency and accuracy metrics. Together, these metrics create a new evaluation metric (accuracy divided by latency) to guess at a model that presents the best tradeoff between accuracy and latency. The base model is selected only once and may be susceptible to differences between the sample set and the testing set. It is expected that a dynamic selector since it is being used many times, will be able to select the model that is right for the image, which can lead to improvements in both performance as well as latency. It will also help that an incorrect choice can easily be rectified in the next pick. Picking the selector only once, at the start of the program, creates the risk that a sub-par choice can affect all downstream tasks.

Recent Upgrades

The base-model selector was upgraded in the summer of 2023. Instead of choosing between the Yolov5 family of models, the selector now chooses between models in the yolov7 family. The most significant difference is that Yolov7 models, on average, possess lower latency and higher accuracy while simultaneously using fewer parameters (see Image 4). While upgrading the model went smoothly, the largest problem came in the form of slow inference speeds, unfamiliarity with docker, and broken sections of the codebase. The slow inference speeds were resolved once GPUs were acquired; however, fixing the broken docker orchestration turned out

to be more difficult than expected. The upgrade was made and tested outside of any containers, and eventually, the broken interfaces to other parts of the TANGO framework were repaired.

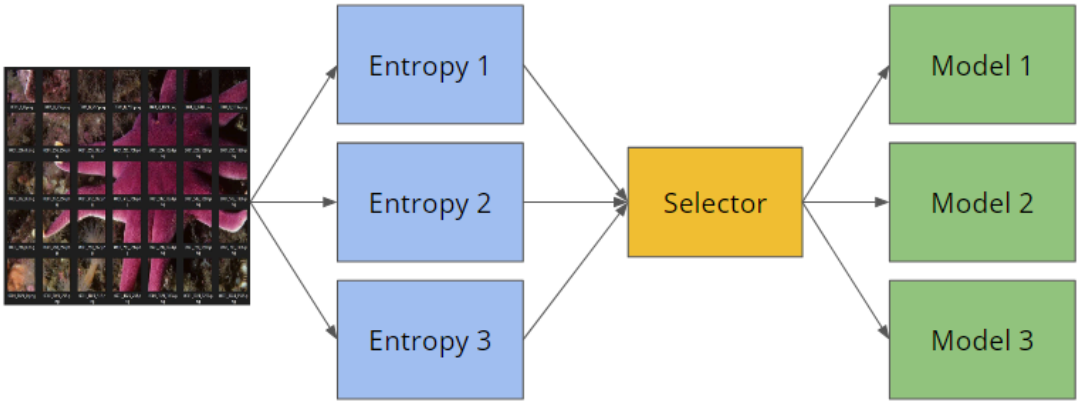

Image 1: Inference Architecture with Selector

Selector

Instead of making one prediction for the entire dataset, a new selector mechanism was created to allow for the dynamic selection of models. A complex image should be run on a slow and complex model, and a simple image should be run on a smaller but faster model. This dynamic routing can allow the users to obtain a better accuracy/latency metric. This way unexpected variations in the dataset, can still be handled gracefully. To create such an evaluator, the resulting mechanism must be trivially fast compared to the cost of running the selected object-detection model, otherwise, it would simply be easier to run the largest model every time.

The selector, under the hood, is simply an SVM. As an input, it takes in an array of Shannon entropy values and outputs the model that it believes is able to provide the best accuracy/latency ratio. For a given dataset, the Shannon entropy has been shown to correlate with the performance

of semantic segmentation networks and as such, was chosen to be an indicator of image complexity to feed into the selector for the object detection models. The image is split into several sections, and an entropy value is calculated for each section, creating an array of entropies as input. To train the SVM, each image was labeled with the smallest model that was able to detect all the objects in the scene.

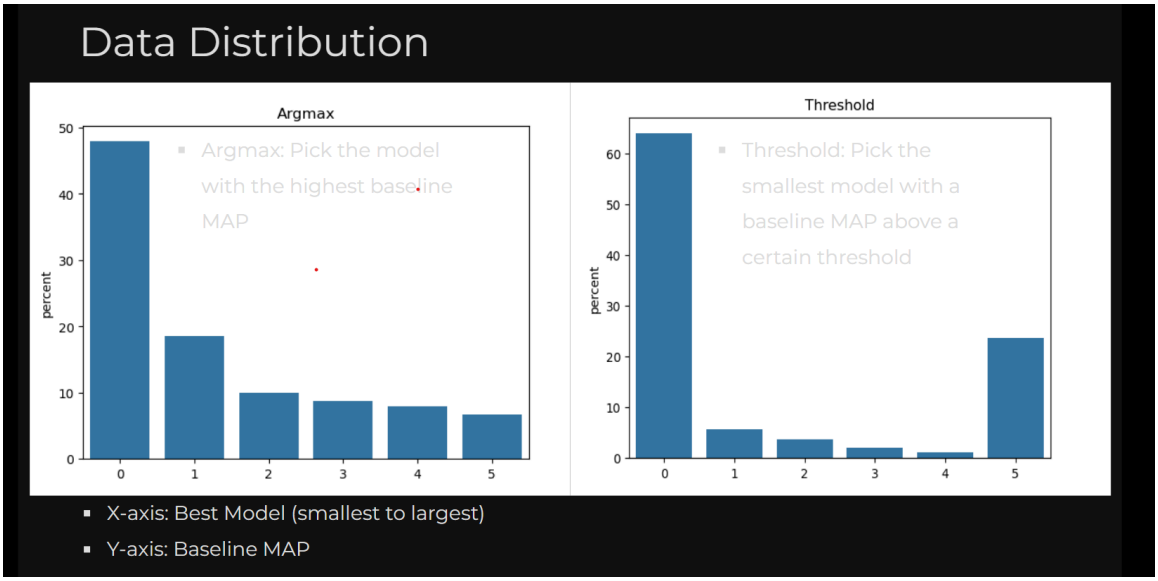

Image 2: Plot showing the most frequently chosen models by the selector

Conclusion

It was noticed that regardless of the training setup, the selector strongly preferred picking the smallest model over others. Additionally, it was seen that the accuracy between models could differ only by a hundredth or thousandth of a percent. It was also noticed that the Yolov7 family of models in general did not vary significantly in mAP. With the current architecture setup to optimize accuracy divided by latency, the selector will almost always choose the fastest model. The different Yolov7 versions are meant to service different types of GPUs rather than the

accuracy or latency metric. Additionally, the yolov7 models are also specialized for certain types of inputs/use cases. For example, the Yolov7-X may be good for handling images of size 640x640, but may fail when exposed to 1280x1280 size. Another of the same family, such as Yolov7-w6, would perform better here. Note that for a group of object-detection models that have a significantly larger accuracy and latency distribution, a selector may be the correct approach. For the Yolov7 family, the accuracies are simply too close to where differences in model selection are more likely to be perpetuated by training randomness rather than any meaningful information.

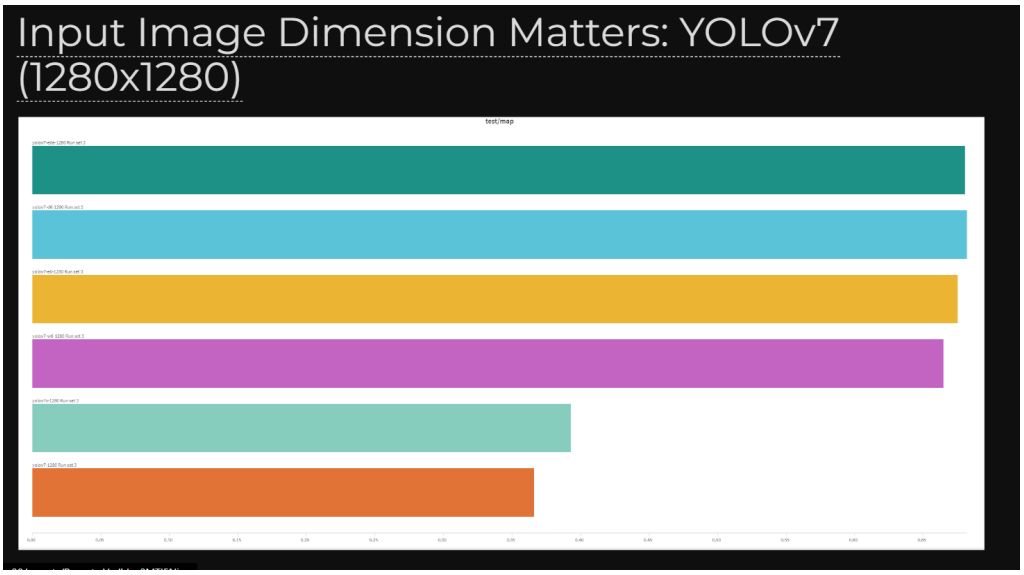

Image 3: Graph showing the accuracy of Yolov7 models with a 1280x1280 size input image

Table 2: Comparison of state-of-the-art real-time object detectors.

Model	#Param.	FLOPs	Size	FPS	AP^{test} / AP^{val}	AP^{test}_{50}	AP^{test}_{75}	AP^{test}_{S}	AP^{test}_{M}	AP^{test}_{L}
YOLOX-S [21]	9.0M	26.8G	640	102	40.5% / 40.5%	-	-	-	-	-
YOLOX-M [21]	25.3M	73.8G	640	81	47.2% / 46.9%	-	-	-	-	-
YOLOX-L [21]	54.2M	155.6G	640	69	50.1% / 49.7%	-	-	-	-	-
YOLOX-X [21]	99.1M	281.9G	640	58	51.5% / 51.1%	-	-	-	-	-
PPYOLOE-S [85]	7.9M	17.4G	640	208	43.1% / 42.7%	60.5%	46.6%	23.2%	46.4%	56.9%
PPYOLOE-M [85]	23.4M	49.9G	640	123	48.9% / 48.6%	66.5%	53.0%	28.6%	52.9%	63.8%
PPYOLOE-L [85]	52.2M	110.1G	640	78	51.4% / 50.9%	68.9%	55.6%	31.4%	55.3%	66.1%
PPYOLOE-X [85]	98.4M	206.6G	640	45	52.2% / 51.9%	69.9%	56.5%	33.3%	56.3%	66.4%
YOLOv5-N (r6.1) [23]	1.9M	4.5G	640	159	- / 28.0%	-	-	-	-	-
YOLOv5-S (r6.1) [23]	7.2M	16.5G	640	156	- / 37.4%	-	-	-	-	-
YOLOv5-M (r6.1) [23]	21.2M	49.0G	640	122	- / 45.4%	-	-	-	-	-
YOLOv5-L (r6.1) [23]	46.5M	109.1G	640	99	- / 49.0%	-	-	-	-	-
YOLOv5-X (r6.1) [23]	86.7M	205.7G	640	83	- / 50.7%	-	-	-	-	-
YOLOR-CSP [81]	52.9M	120.4G	640	106	51.1% / 50.8%	69.6%	55.7%	31.7%	55.3%	64.7%
YOLOR-CSP-X [81]	96.9M	226.8G	640	87	53.0% / 52.7%	71.4%	57.9%	33.7%	57.1%	66.8%
YOLOv7-tiny-SiLU	6.2M	13.8G	640	286	38.7% / 38.7%	56.7%	41.7%	18.8%	42.4%	51.9%
YOLOv7	36.9M	104.7G	640	161	51.4% / 51.2%	69.7%	55.9%	31.8%	55.5%	65.0%
YOLOv7-X	71.3M	189.9G	640	114	53.1% / 52.9%	71.2%	57.8%	33.8%	57.1%	67.4%
YOLOv5-N6 (r6.1) [23]	3.2M	18.4G	1280	123	- / 36.0%	-	-	-	-	-
YOLOv5-S6 (r6.1) [23]	12.6M	67.2G	1280	122	- / 44.8%	-	-	-	-	-
YOLOv5-M6 (r6.1) [23]	35.7M	200.0G	1280	90	- / 51.3%	-	-	-	-	-
YOLOv5-L6 (r6.1) [23]	76.8M	445.6G	1280	63	- / 53.7%	-	-	-	-	-
YOLOv5-X6 (r6.1) [23]	140.7M	839.2G	1280	38	- / 55.0%	-	-	-	-	-
YOLOR-P6 [81]	37.2M	325.6G	1280	76	53.9% / 53.5%	71.4%	58.9%	36.1%	57.7%	65.6%
YOLOR-W6 [81]	79.8G	453.2G	1280	66	55.2% / 54.8%	72.7%	60.5%	37.7%	59.1%	67.1%
YOLOR-E6 [81]	115.8M	683.2G	1280	45	55.8% / 55.7%	73.4%	61.1%	38.4%	59.7%	67.7%
YOLOR-D6 [81]	151.7M	935.6G	1280	34	56.5% / 56.1%	74.1%	61.9%	38.9%	60.4%	68.7%
YOLOv7-W6	70.4M	360.0G	1280	84	54.9% / 54.6%	72.6%	60.1%	37.3%	58.7%	67.1%
YOLOv7-E6	97.2M	515.2G	1280	56	56.0% / 55.9%	73.5%	61.2%	38.0%	59.9%	68.4%
YOLOv7-D6	154.7M	806.8G	1280	44	56.6% / 56.3%	74.0%	61.8%	38.8%	60.1%	69.5%
YOLOv7-E6E	151.7M	843.2G	1280	36	56.8% / 56.8%	74.4%	62.1%	39.3%	60.5%	69.0%

[1] Our FLOPs is calaculated by rectangle input resolution like 640 × 640 or 1280 × 1280.
[2] Our inference time is estimated by using letterbox resize input image to make its long side equals to 640 or 1280.

Image 4: Yolo Family Models and Statistics

For further proof, a t-SNE plot was created to investigate any potential clustering. As seen below, it is quite dominated by purple (the smallest model). It is expected the clusters that do form, relay the semantic information (if there is a bike, car, etc.) of the image rather than any supposed complexity. Since no clear correlation was found and for most purposes, the models perform very similarly, there is no significant advantage to using a model selector for the yolov7 family of models.

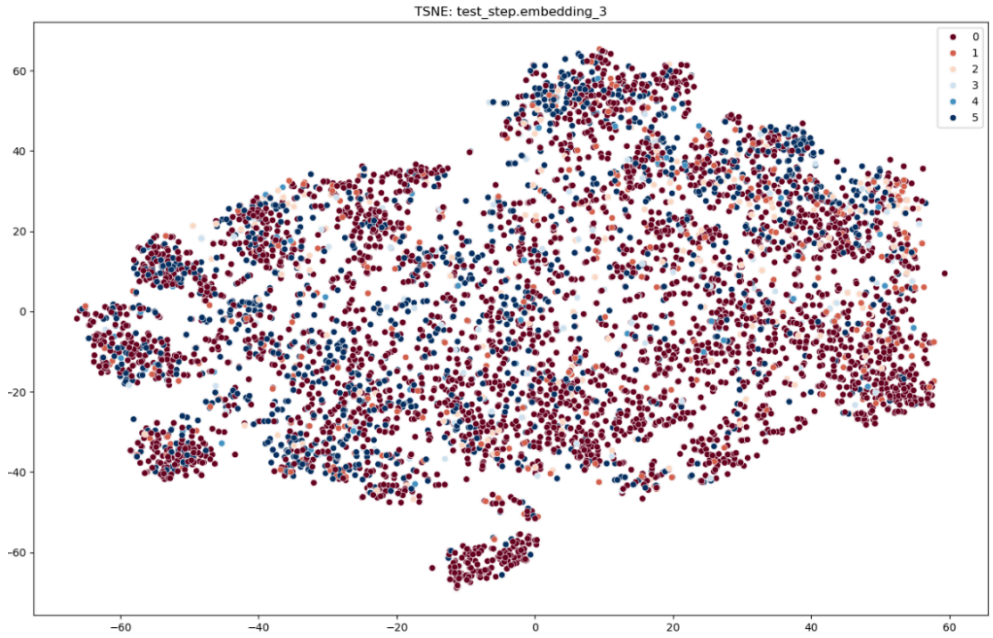

Image 5: t-SNE distribution dominated by points of the smallest model

Code Design

The base model selector in TANGO will be upgraded again to incorporate the research presented above. The first goal is to use an entropy-based model selection mechanism. This shall come in the manner of a pre-trained SVM, especially since it would be simply too time-consuming to train an SVM during run-time. A new mechanism will be created to have an SVM trained (perhaps with a dataset as an input parameter), which will not only output the trained .pt file, but also create a report of the SVM statistics. The second goal is to support dataset-generalized selection. Our design above allows the user to select a model for each image. However, such a design may not integrate well into the current needs of the project and will need to support the previous approach of picking one model for the entire dataset. It is expected that using a pre-trained SVM will be significantly faster than running inference on all Yolo models against a subset of images since calculating the entropy of an image only needs to be done once and is

computationally cheaper than running inference for all models. A large subset of images will simply be passed onto the selector and the most frequently picked model shall be chosen as the most efficient. The increased speed can allow for either a more complex predictor or for the selector to take in a larger subset before making its decision. Implementing the image-specific design can prove to be a challenging task and will not be prematurely added to reduce code bloat.

References

[1] 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,” arXiv:2207.02696 [cs], Jul. 2022, Available: https://arxiv.org/abs/2207.02696

[2] “Measures of Complexity for Large Scale Image Datasets | IEEE Conference Publication | IEEE Xplore,” ieeexplore.ieee.org. https://ieeexplore.ieee.org/document/9065274 (accessed Dec. 13, 2023).

[3] Y. J. Cruz, M. Rivas, R. Quiza, R. E. Haber, F. Castaño, and A. Villalonga, “A two-step machine learning approach for dynamic model selection: A case study on a micro milling process,” Computers in Industry, vol. 143, p. 103764, Dec. 2022, doi: https://doi.org/10.1016/j.compind.2022.103764.