university of groningen

faculty of science and engineering

Capture The Flag - Developing a Pipeline for Binary Flag Detection

Bachelor's Project Computing Science

June 2024

Author: Stefan Malinovski
Student Number: s4790804
First supervisor: dr. George Azzopardi
Second supervisor: Jelle Visser, Msc

Abstract

Flag detection systems can have many useful applications in the world of safety and security. Recent advances in machine learning enable the development of these systems using deep learning approaches. However, the effectiveness of such methods heavily depends on the availability of large amounts of data.

This bachelor's thesis presents the development of a flag detection system using a YOLOv9 object detection model. Annotated flag images from the Open Images V7 dataset serve as the initial training data. To further address the lack of large-scale annotated datasets, various data enhancement methods are investigated with the goal of improving the model's performance and robustness.

Synthetic data is produced by a Stable Diffusion XL image generation model, as well as by programmaticaly altering stock photos of flags and pasting them onto natural image backgrounds. Augmented data is created by taking samples from the base dataset and applying transformations to them to create new images.

The results shown in this paper confirm that these methods are viable answers to the lack of data and can be used to train a fast and accurate flag detection model.

Contents

List of Figures

List of Tables

1 Introduction

The detection of symbols that indicate national, regional, political or ideological affiliation is essential in early detection of threats and establishing of patterns within these threats. This information is beneficial for police, national security or military purposes. For broad use of the tool, an algorithm detecting any flag or similar affiliation symbol allows generalized detection of even previously unseen indicators.

Deep learning based object detection methods can be used to achieve reliable and accurate results [10]. These methods exhibit complex computer vision capabilities by training multilayered neural networks to recognize patterns and make predictions. A critical component of this is a large dataset from which the network can learn. The performance of the resulting model is directly influenced by the quality of the data, necessitating a diverse and comprehensive dataset. If the dataset is biased or incomplete, the model will inherit these flaws, leading to inaccurate predictions and an ineffective tool.

There is a lack of a open source large-scale annotated dataset for training flag detection models. Additionally, flags have non-uniform shapes and sizes and they can be mistaken for other objects such as posters or banners. In [10], the authors address these problems by creating a comprehensive dataset with synthetic and augmented images. They train a pipeline based on Mask-RCNN and PointRend to recognize and segment flags from up to 225 countries. Data is synthesized by altering stock photos of flags and pasting them onto natural image backgrounds, as well as by transforming samples from their dataset.

This project focuses solely on the binary detection of flags in images, separating their detection from their recognition. The research goal is to investigate the effectiveness of data enhancement methods on model performance. YOLOv9 was chosen because it is a single-shot detector [11], resulting in much faster inference times when compared to two-shot detectors like Mask-RCNN. This makes it more suitable for use-cases which may require real time detection. The model was trained on datasets with varying distributions of natural, synthetic, and augmented data, and conclusions will be drawn by comparing the results.

Taking all of this into account, the research questions this paper seeks to answer are "*What is the performance of YOLOv9 when used for binary flag detection in real world images?*" and "*Which data enhancement methods best increase the performance of the model?*"

Two directions for data enhancement were explored, those being data augmentation and data synthesis. With data augmentation, samples are taken from a preexisting dataset and transformed in some way to generate new samples. Depending on their effect on the bounding boxes, the data augmentations can be split into pixel transformations and spatial transformations. The latter changes the bounding boxes, while the former keeps them the same. With data synthesis new samples are generated using some method. Two approaches were explored for this, programmatic data synthesis and generative AI data synthesis. For programmatic data synthesis, stock images of flags were altered to look more lifelike and then pasted onto natural image background. For generative AI data synthesis, a Stable Diffusion XL text-to-image model was used to synthesize images of flags.

Chapter 2 gives further background information about the methods used and the reasoning behind them. Chapter 3 delves into the setup of the experiments and analyzes their results, with the paper concluding in chapter 4.

2 Methods

2.1 YOLOv9

At the core of the project is the YOLOv9 object detection model. As previously stated, the YOLO series of models are all single-shot detectors [4]. Instead of detecting possible regions of interest using a region proposal network and then performing recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer.

YOLOv9 introduced 2 new features that improved it's performance when compared to previous iterations, those being *programmable gradient information* (PGI) and a new lightweight network architecture - *Generalized Efficient Layer Aggregation Network* (GELAN). The features were added to address the challenges posed by information loss in deep neural networks. PGI helps to combat the information bottleneck problem, ensuring the preservation of essential data across deep network layers, while GELAN is a architectural advancement, enabling YOLOv9 to achieve superior parameter utilization and computational efficiency.

Since the goal of the project is to test performance of the model, the YOLOv9e version was chosen. It is the largest of all of the YOLOv9 versions, with the "e" standing for extended. This choice was made with the performance metrics in mind, since the aim of the project is to optimize for model accuracy and not inference time.

2.1.1 Hyper-parameters

Hyper-parameters are external values in machine learning models that are set before the training process begins. They are not learned from the data but are crucial for guiding the training process and determining the performance of the model. All the hyper-parameters that are relevant for the conducted experiments can be seen in Table 1

epochs represents the number of training iterations that the model undergoes, while the value of *patience* controls the early stopping for the model. If the model doesn't improve in a set amount of epoch, training stops early. The *batch* hyper-parameter dictates the number of training examples processed in a single forward and backward pass through the network during one iteration of training. lr_0 and lr_f are the starting and final learning rate for the model. The learning rate is the amount by which the model updates itself on every iteration. Most commonly *lr0* and *lrf* have the same value. *momentum* controls how much the previous update influences the current one. *weight_decay* is a penalty added to the loss function to avoid having large weights.

The *warmup_epochs*, *warmup_momentum* and *warmup_bias_lr* all have to do with the warmup period of the training. The warmup period is a sort of initialization phase that ramps up the model at the beginning of the training. *warmup_epochs* controls the number of epochs, that the warmup lasts for, while *warmup_momentum* and *warmup_bias_lr* are the momentum and learning rate respectively during warmup.

The last 3 weights, *box*, *cls* and *dfl* all represent loss function gains, for the box, cls and dfl loss functions respectively. These values control the weight assigned to each loss function. Loss functions will be further explored in 2.1.2

Name	Value
epochs	200
patience	30
batch	16
lr0	0.01
lrf	0.01
momentum	0.937
weight_decay	0.0005
warmup_epochs	3.0
warmup_momentum	0.8
warmup_bias_lr	0.1
box	7.5
cls	0.5
dfl	1.5

Table 1: Hyper-parameters

The values for *epochs* and *patience* were chose by running experiments with the YOLOv9 model. 200 epochs provided enough time for the model to converge, while a patience of 30 prevented it from over-fitting. For the rest of the hyper-parameters, the out of the box values were used, as the model performed well and had good results.

2.1.2 Training Results

An understanding of the outputted results is needed to reason on and analyze the effect of the different datasets on the model. Figure 1 shows an example of this. It depicts the change in the loss values and metrics as they change after each epoch throughout the course of the training. This is useful for understanding how the training is going and reasoning about when the model reaches convergence.

Loss Functions On the left side of the figure the *training* and *validation* loss are shown. The loss function is the value that the AI model is aiming to optimize while training or validation. The *box_loss* value represents the loss associated with the bounding box predictions. It measures how accurately the model can predict the location and size of the bounding boxes for detected objects. *cls_loss* represents the classification loss, which measures the accuracy with which the model can classify the objects within the bounding boxes as belonging to a particular class. In single-class classification, the cls loss directly measures how accurately the model is able to distinguish between the presence and absence of the class of interest. *dfl_loss*, which stands for Distribution Focal Loss, is used to handle class imbalance in the object detection process.

Through comparing the training loss with the validation loss, it can be inferred whether the model is over-fitting on the data or not. If the validation loss starts to increase, while the training loss continues to decrease, this typically indicates that the model is over-fitting. This means it is learning the data too well, capturing noise and details that are specific to

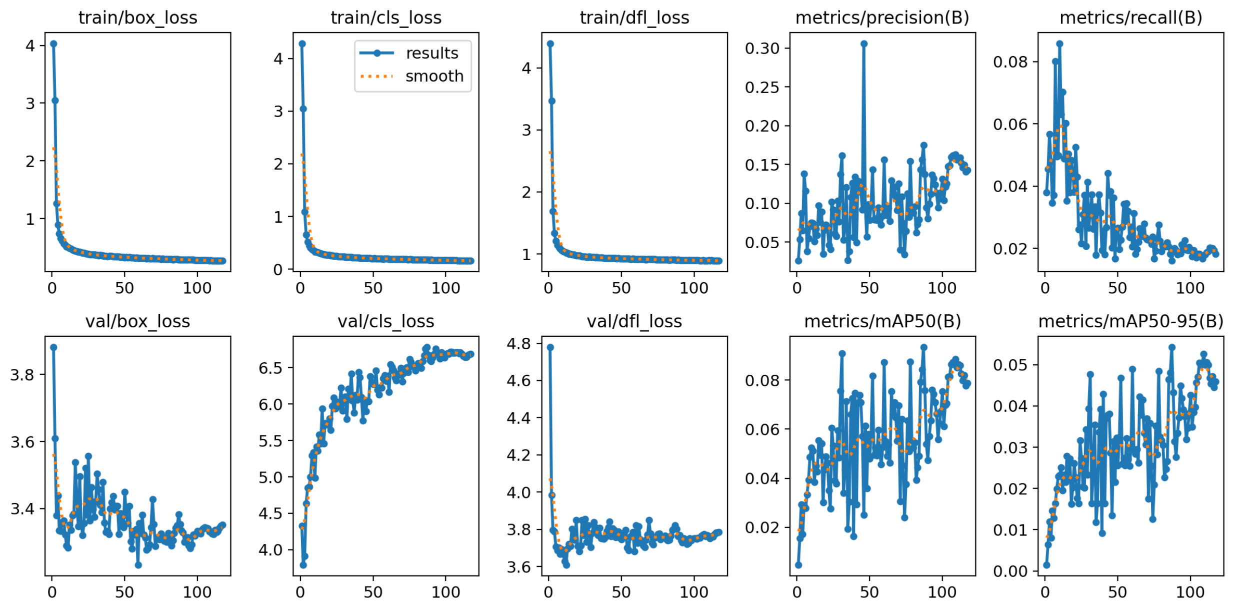


Figure 1: Example YOLOv9 training results

the training dataset and will not generalize well on novel unseen data.

Metrics The model also provides the following training metrics: *precision*, *recall*, *mAP50*, *mAP50-95*. Precision is the ratio of true positives to all positives, which for detecting flags would mean the ratio of flags correctly detected out of all of the predictions.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Recall is the measure of how often the model identifies true positive instances out of all of the positive samples in the set. Following the flag detection example, recall would be the number of flags correctly detected divided by the the total number of flags in the image.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

These metrics mustn't be taken individually as they can be misleading. It's possible to have very high recall by just simply detecting every single object or to have great precision by only labeling a small selection of object with high likelihood of belonging to the correct class. Because of this it is useful to look at how both variable interact with each other. This is given by the model in the form of an Precision-Recall curve, as can be seen in Figure 2.

Average Precision (AP) summarizes the PR curve into a single value. It is calculated by integrating the precision over all recall levels, essentially computing the area under the PR curve. mAP extends the concept of AP to multiple classes and multiple Intersection over Union (IoU) thresholds. IoU measures the overlap between the predicted bounding box and the ground truth bounding box. An IoU threshold determines what is considered a true positive detection. mAP50 uses a IoU threshold of 50, while mAP50-95 uses multiple

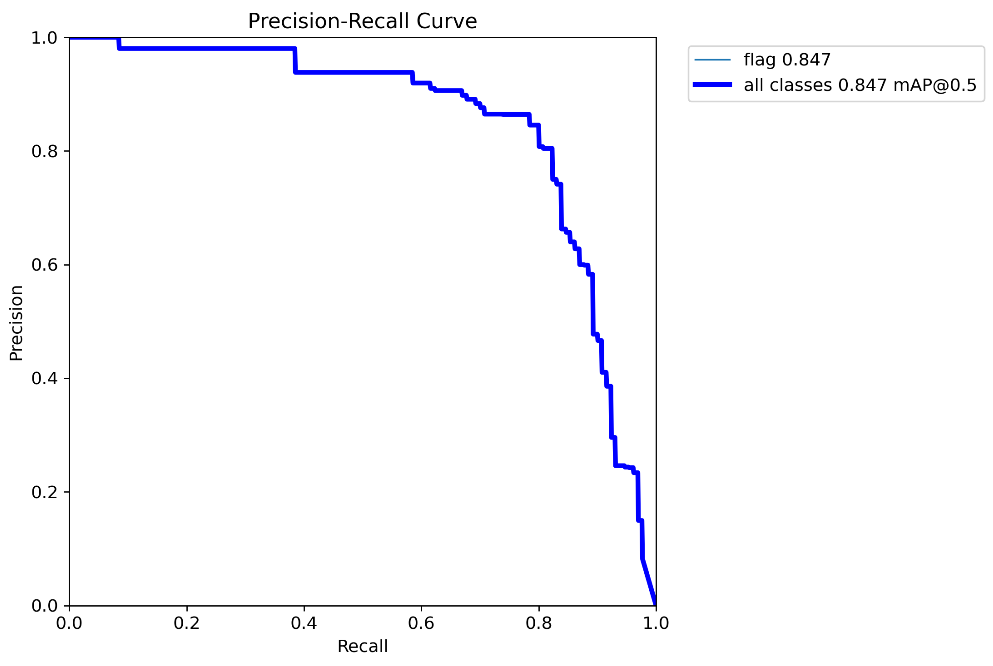


Figure 2: Precision Recall curve

thresholds ranging from 50-95. Since the goal of this project is binary detection, which applied means there is only one class, the mean average precision equals the average precision.

An additional metric which shows the interaction between *Precision* and *Recall* is *F1-Score*. The F1 score is the harmonic mean of precision and recall, providing a single metric that balances the two. It is especially useful for dealing with both false positives and false negatives in datasets with imbalanced classes.

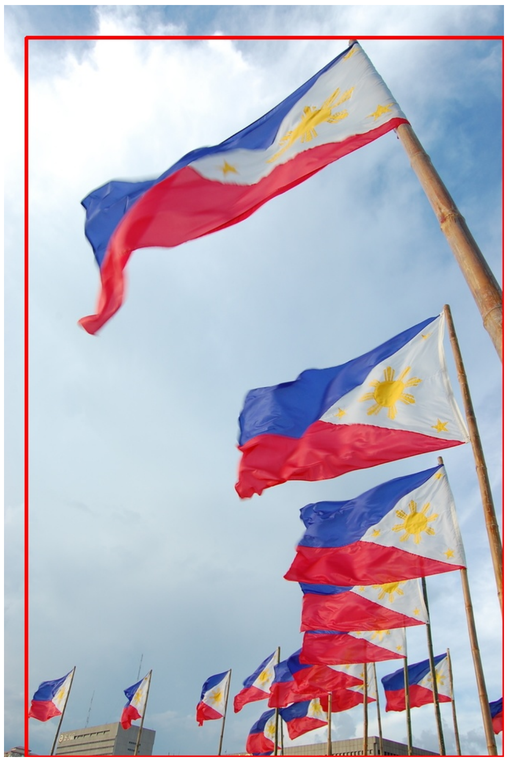
$$Recall = 2 * \frac{Precision * Recall}{Precision + Recall}$$

After training them, all the different models will be tested on the same "ground truth" base set. The resulting metrics will be used to reason about the effectiveness of the data enhancement methods that were used to construct the datasets that were used for training. The main metric that will be given the most weight is mAP50-95.

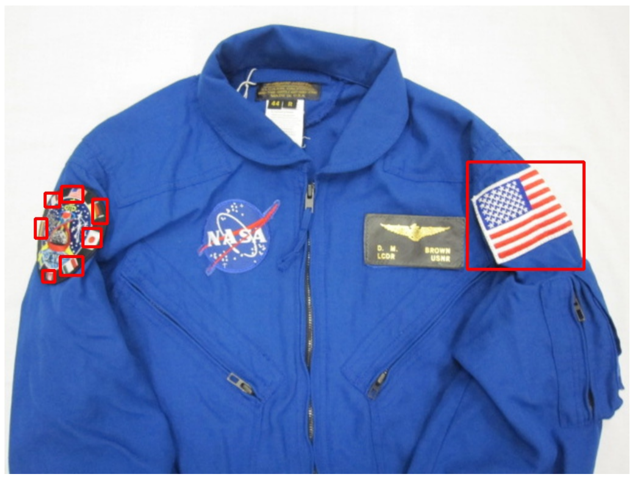
2.2 Base Dataset

One of the main challenges with this project, is the lack of large scale annotated training data. As a starting point, the Open Images V7 Dataset [3] was chosen (OIV7D). After filtering for images with flags, 8691 samples were left in the dataset.

Figure 3 is an example of the images in the OIV7D dataset. In Figure 3a not all flags are labeled individually, but instead are put together under one annotation. This is not ideal as it can reduce accuracy of the model, since we want to train it to detect all flags separately. Figure 3b on the other hand is an example of precise labeling, as not only the flag of the United States of America (USA) is annotated, but all of the individual flags on



(a) Example of imprecise labeling.



(b) Example of precise labeling.

Figure 3: Examples of Open Images Dataset V7

the arm-patch on the left arm. Additionally, we have to be aware of biases in the data. One of the more prominent ones is the over-representation of the USA's flag, which can lead to the model being less accurate on non USA flags.

To account for the lack of annotated data, Data Augmentation (2.3) and Data Syntheses (2.4) were used as methods for enriching the dataset.

2.3 Data Augmentation

All the images available in the base dataset were run through a data augmentation pipeline to create new samples that can be used for training the model. As stated in Section 1, the Python library Albumentations [2] was used to accomplish this. It supports fast and flexible image transformations and is widely used in the Computer Vision community. YOLOv9 has image augmentation capabilities built in, automatically applying a number of transformations to the dataset before training the model, however these will be disabled. For these experiments, all augmentations will be manually applied by a Python script before feeding the dataset to the model.

The transformations can be divided based on whether they change the bounding boxes in the image. Those that do not alter the bounding box fall into the *Pixel-Level* transforms category, while the ones that do are a part of the *Spatial-Level* transforms.

2.3.1 Pixel-Level Transformations

The complete 40 pixel level transformations can be found in appendix A. The augmentations that were picked, were chosen with the goal of broadening data diversity and creating a more encompassing dataset.

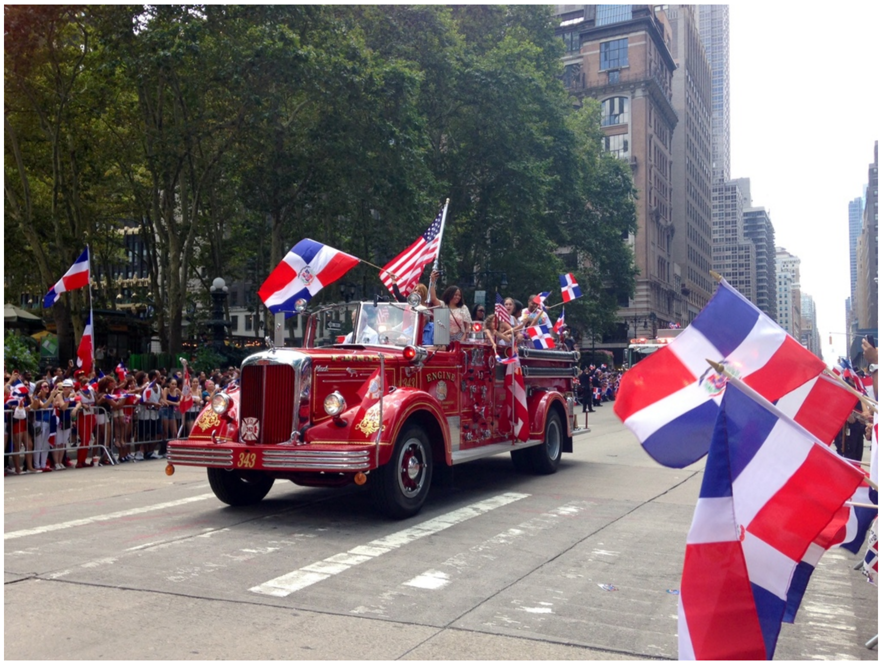


Figure 4: Example Image without any transformations

Figure 4 shows an unaltered image. It depicts an older fire truck driving as a part of some parade with people waving numerous flags around. After sending this image through the pipeline, 40 new images are produced. Figure 5 depicts the image as transformed by 6 different augmentations.

All the transformations applied depict common scenarios that might occur when taking photographs of flags. Figure 5a has the *Defocus* transformation applied and similarly Figure 5f has the *ZoomBlur* transformation applied. Both of these transformations are relevant, since when taking photos in real life it is very easy to lose focus on your camera or blur the photography by zooming. Likewise, Figures 5c and 5d show the image under 30% and 50% compression. Even though at a first glance their effect may seem minimal, since many popular image sharing platforms implement some form of image compression, these transformations are critical in training the model to effectively run on real world data. The *RandomRain* transformation, shown in Figure 5e, could help train the model for what otherwise might be an underrepresented condition, as any training dataset might be lacking in images depicting flags in the rain. Finally, the *RGBShift* transformation, adds more variety to the dataset and lowers the models bias towards real world flags, by switching the colour channels of the image and creating new flags with different colors.

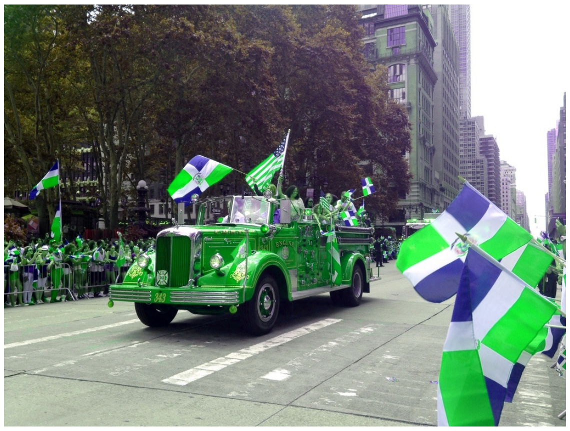
Following this, there are more transformations, such as *Sharpen*, *Blur*, *RandomFog* or *HueSaturationValue* that are added to account for real world situations and boost dataset diversity.

(a) *Defocus* Transformation

(b) *RandomRain* Transformation

(c) *ImageCompression30* Transformation

(d) *ImageCompression50* Transformation


(e) *ChannelShuffle* Transformation

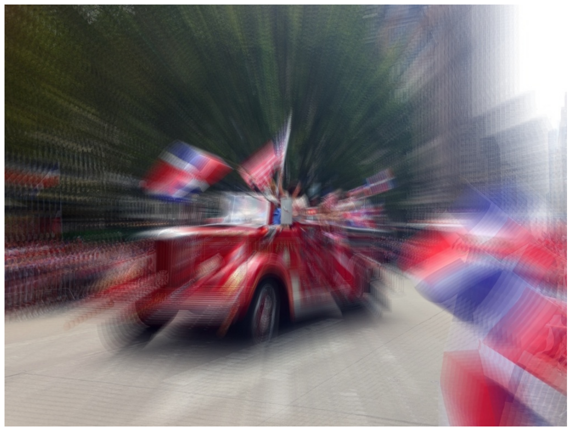
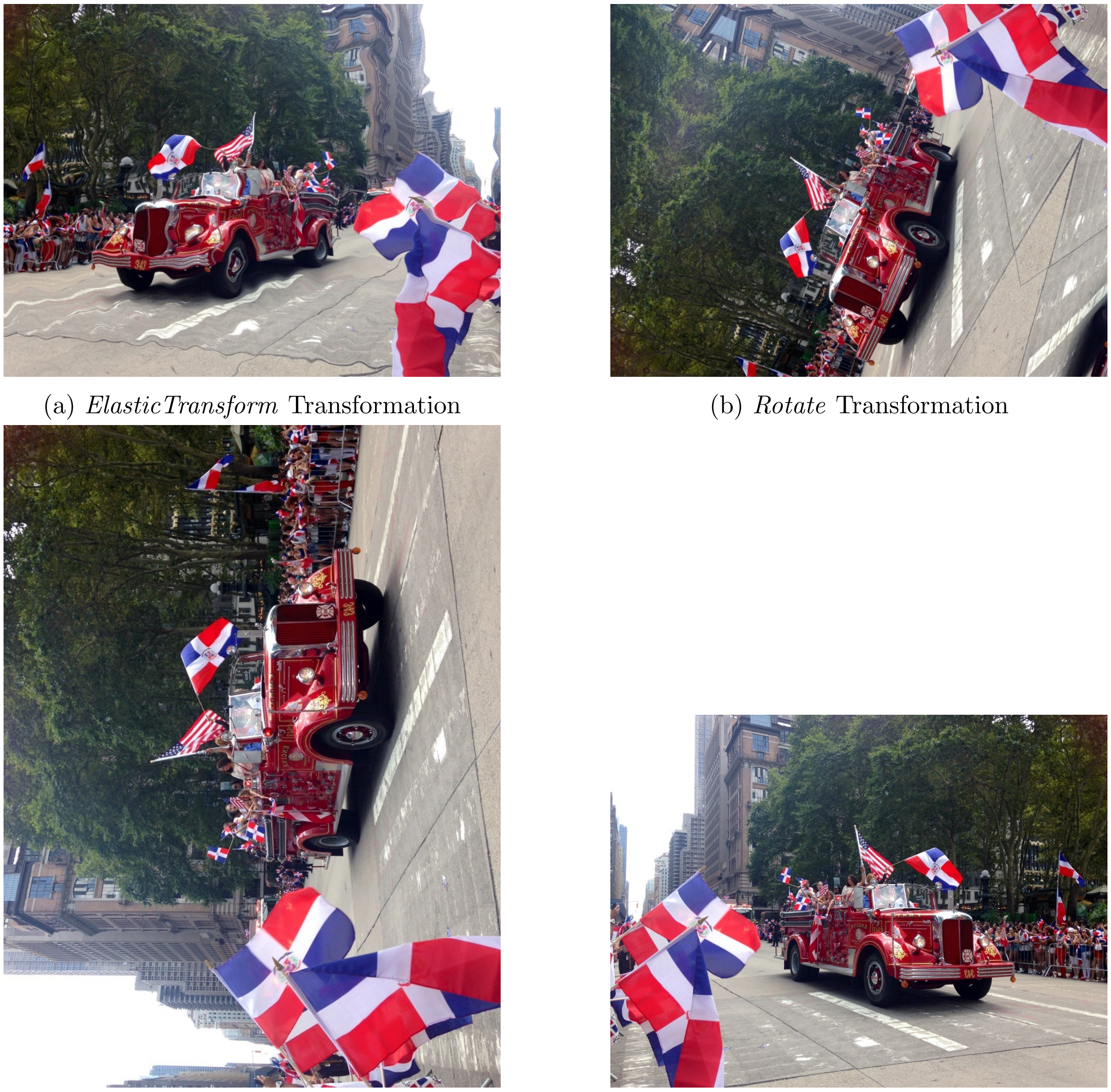

(f) *ZoomBlur* Transformation

Figure 5: Examples of Albumentations Pixel Transformations

2.3.2 Spatial-Level Transformations

Due to some of the transformations not working with bounding boxes or being functional copies of each other, the number of spatial transformations applied is smaller than pixel, with there being only 12 transformations. The full list of transformations can be found in appendix B.



(a) *ElasticTransform* Transformation

(b) *Rotate* Transformation

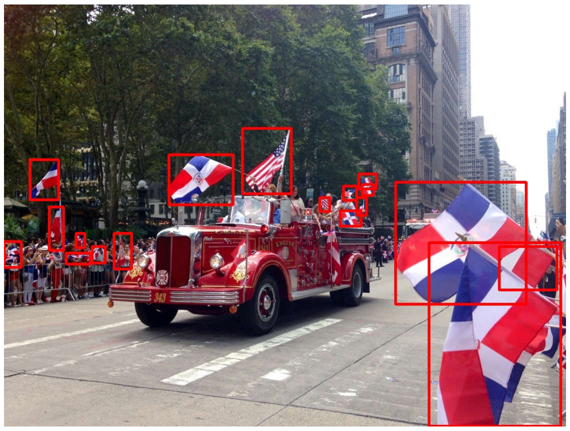
(c) *Transpose* Transformation

(d) *Flip* Transformation

Figure 6: Examples of Albumentations Spatial Transformations

Taking Figure 4 as an example again, the images in Figure 6 showcase some of the transformations that are applied to the image. An important part of the spatial transformations is preserving or altering the bounding boxes in the images. By rotating the image like in

Figure 6b the leftmost flag in the image is lost and the pipeline needs to properly account for this and change the annotation. Exactly this can be noticed in Figure 7, with image 7a showcasing the bounding boxes before the rotation and image 7b showing them afterwards.


(a) Base image with bounding boxes.

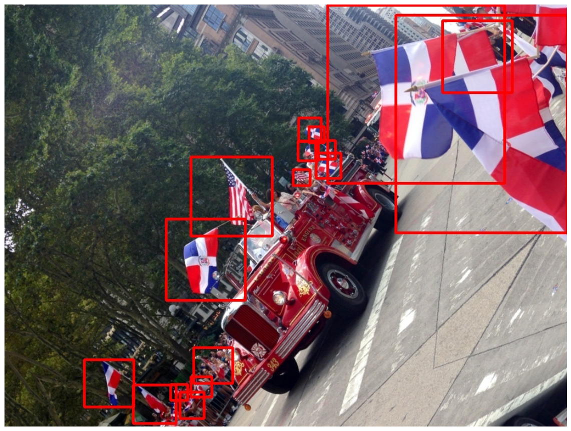

(b) Rotated image with bounding boxes.

Figure 7: Examples of how spatial transformations alter bounding boxes.

2.4 Data Synthesis

Another answer to the lack of data is to generate the data needed for the model. Doing this allows for greater control over the images that are fed to the model, which can be used to fix certain biases the are present in the model from the base dataset. 2 methods for synthesizing data were explored, programmatically creating the data by transforming stock images and pasting them onto real world backgrounds and using a generative AI model to synthesize the images.

2.4.1 Programmatic Data Synthesis

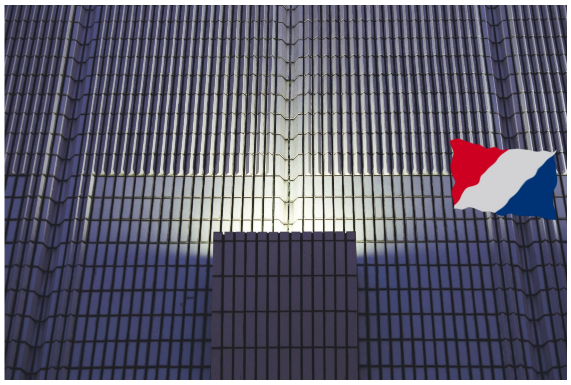
Stock images of flags were obtained from [9] after which they were put through a image augmentation pipeline that transforms them into data usable for the model. The first step of the pipeline is to make the flags more "wavy" and lifelike by applying several transformations to the stock images. This is done in 2 steps by firstly applying transformations like *RandomBrightnessContrast*, *HueSaturationValue* and *Rotate*, followed by putting a translucent mask around the flag and applying *OpticalDistortion* and *ElasticTransform* to the image. Afterwards, the transformed image is copied onto a background image. The background images were obtained from the *BG-20K* dataset [6, 7, 5]. An example of the entire pipeline can be seen in Figure 8.

The benefit to this approach is that control over where the flag is pasted in the background image is maintained, making labeling the samples part of the generative process and trivializing it. This way, thousands of samples can easily be generated.

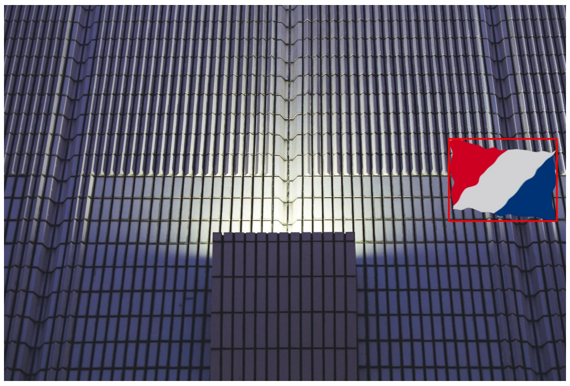
(a) Stock image of the Dutch flag.

(b) Step 1 in synthesizing the sample.

(c) Step 2 in synthesizing the sample.



(d) Final sample.



(e) Final sample with bounding box.

Figure 8: Examples of data augmentation pipeline.

2.4.2 Generative AI-Based Data Synthesis

With the recent growth and improvement of generative AI, using a text-to-image model to create a synthetic dataset is a valid option. Stable Diffusion XL (SDXL), which is an open source model that can be run locally, was utilized to generate the images. The prompts used can be found in Appendix C.

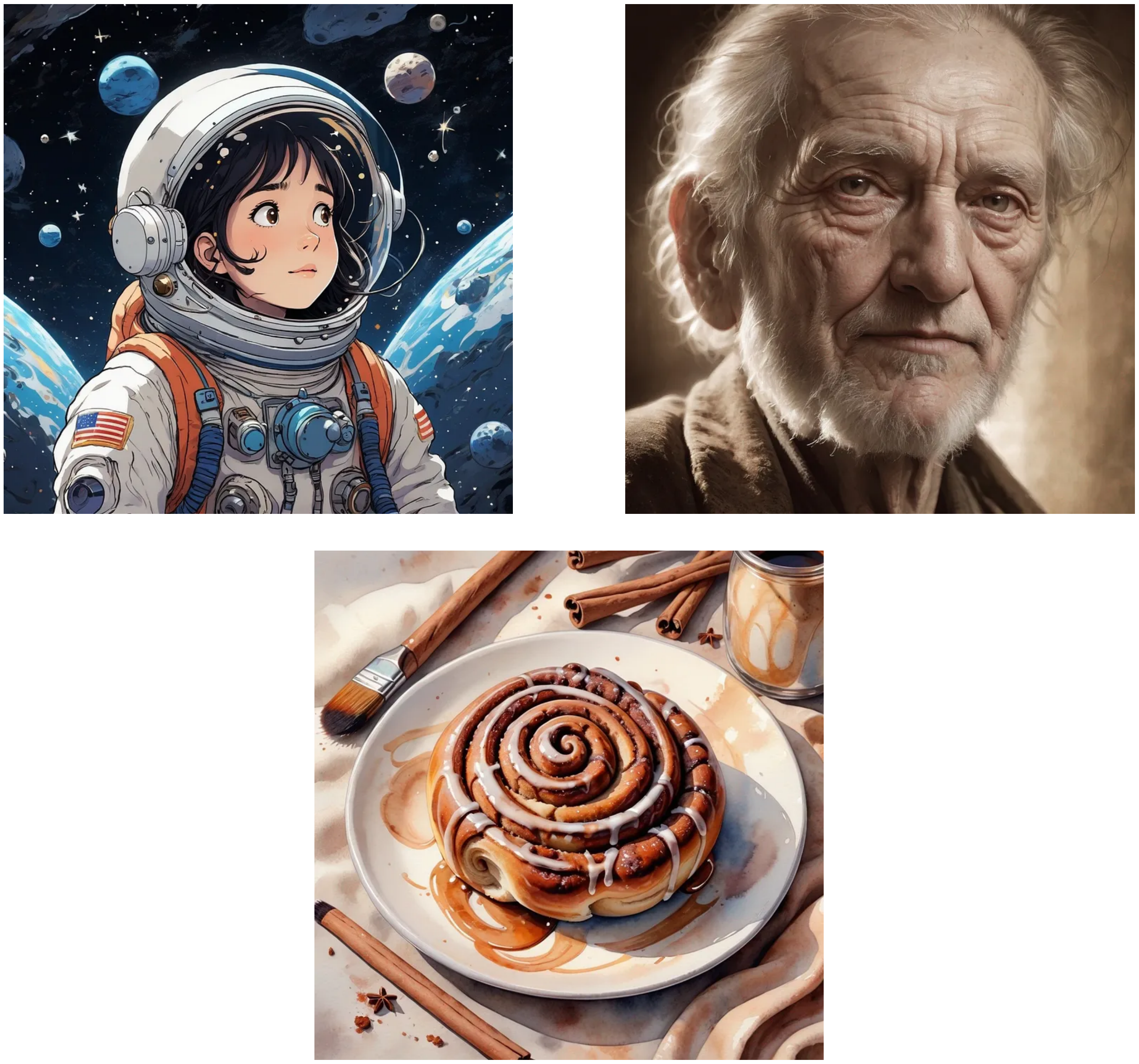
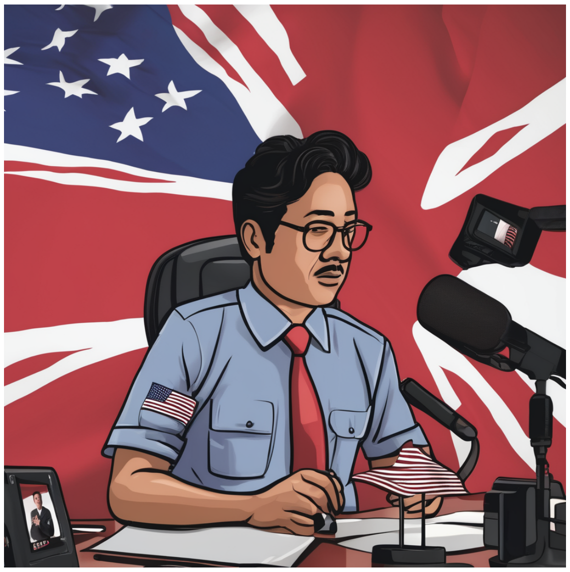


Figure 9: Examples of images generated by SDXL.

Prompt Engineering To effectively utilize the capabilities of SDXL, experiments were conducted to gain insight into proper prompt writing. Small differences in prompts can cause big differences in the output of the model. Since the goal is photo-realism, adding words and phrases like "photo, CANON EOS R3, 80mm, 1/125 Sec shutter speed" associates the model towards photography instead of art and paintings and results in more realistic images. Another way to avoid generating cartoon-ish images is using negative prompts. Negative prompts are very important since they explicitly tell the model what things to avoid. For example, keeping with the goal of more realistic imagery, we can add "art, painting, drawing, anime, cartoon, low poly, fantasy art" in the negative prompt. Likewise if the model is generating humans with distorted bodies, adding "disfigured, ugly, bad" helps to mitigate that. All of this helps with avoiding photos like like Figure 10a and have the generated data resemble Figure 10b.


(a) Image generated with improper prompt.

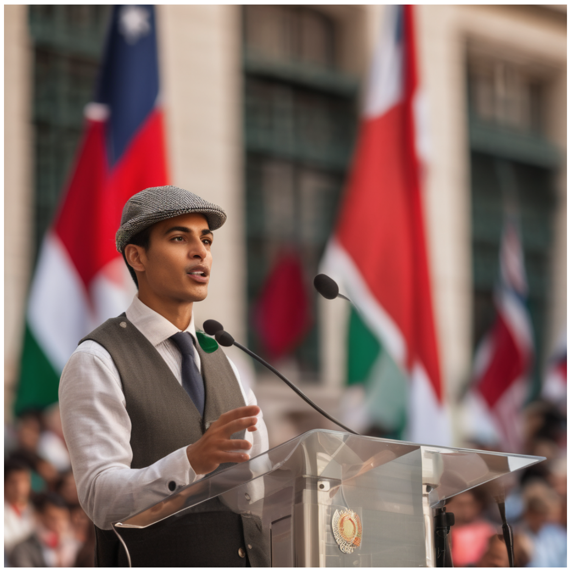
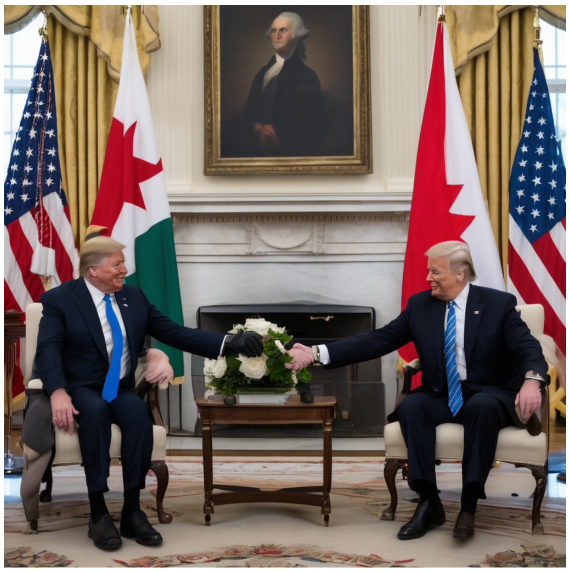

(b) Image generated with proper prompt.

Figure 10: Differences between proper and improper prompt writing.

It is crucial to carefully consider the choice of words when constructing prompts, as specific terms can have significant implications. Using the prompt "*Two presidents shaking hands with their nations flags behind them, photography, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed*" with the negative prompt "*art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor*" results in figures such as 11a, where both of the people in the image resemble former US president Donald Trump. By simply changing the word president to "*country leaders*" with no other changes, the model produces images like Figure 11b, where the people depicted have a wider variety appearances.

Stable Diffusion could be used as a tool to "patch up" holes that are left by the base model. If a user has a specific use case for the model, that features flags in a situation that is under-represented in the base dataset, targeted prompts can be written to specifically depict that scenario which would improve the performance of the model when applied to that use case. The prompts that were written for the experiments are meant to test this proposal as well as depict common images where flags can appear. An example of this is the prompt "*A photograph of soldiers on a base saluting a flag...*" which is a specific scenario that is unrepresented in the base dataset. Additionally the prompt "*A photograph of a*


(a) Generated image with "*presidents*" in the prompts.

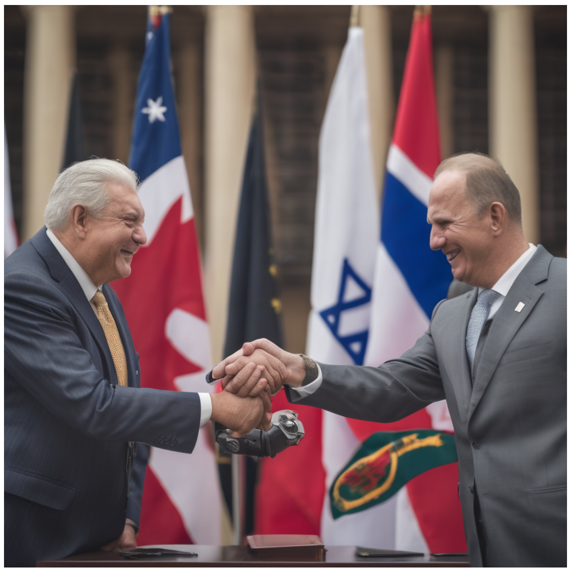

(b) Generated image with "*country leaders*" in the prompts.

Figure 11: Two images generated with the same prompt, with only a difference between the phrases "presidents" and "country leaders".

country's national flag waving in the wind on a blue sky..." is a very common image of a flag that appears in the base dataset and should be included the generated images to test if the generated data lowers the performance of the model.

Model Biases When making use of AI models, it is important to be mindful about potential biases inherent in the models. One of the prompts that was utilized focused on generating protesters with flags. The prompt contained the sentence "*A protester carrying his country's flag...*". After preliminary testing the model seemed to generate images depicting solely people with darker skin tones. This resulted in different phrases and words being added to the prompt to increase the diversity of the subjects depicted. However after generating 100 images for the dataset, the results still were still heavily biased and skewed towards people of darker skin color. Aside from diluting the predictive capability of the model, as researchers we should always strive for equality and fairness in the field and be mindful about biases in our work.

2.5 Final Datasets

To answer the research questions posed in Section 1, several different datasets were made using the methods described in Section 2.3 and 2.4. These datasets were be used to train different YOLOv9 models, that were then be tested on the same set of "test" images. The performance metrics of each of these models was then be compared with the goal of drawing meaningful conclusions.

The list of all datasets and their descriptions can be found in can be found in Table 2.

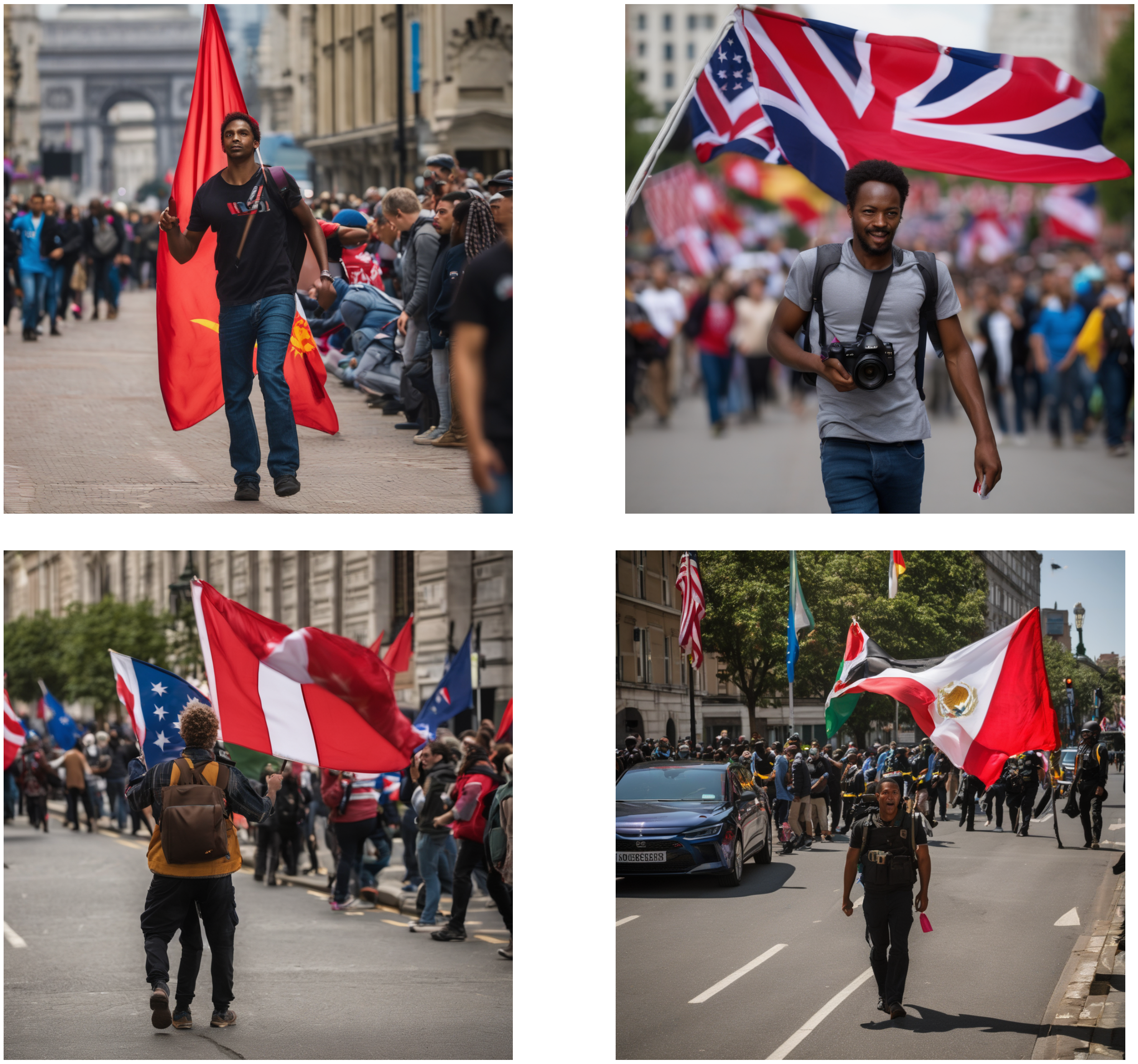


Figure 12: Examples of images generated with a biased model.

Dataset	Description	Number of Samples
Base	The base dataset containing images from the Open Images v7 Dataset, as described in section 2.2.	8476
Base + Stable Diffusion	As the title eludes to, this dataset would be the "Base" set with 1258 images generated using Stable Diffusion added onto the train split of the dataset. This would test if the images generated with the targeted prompts improved the performance of the model on a specific use case as well as the test set.	9734
Synthetic	This dataset would be constructed using programmatic data synthesis. It would have the same number of images in the test and validation splits as the "Base" dataset, with both splits being entirely composed of generated images. The aim of it is to test the viability of programmatic synthesis as the only source of data.	8476
Base Synthetic 50/50	This dataset takes 50% of the train split from the "Base" dataset and mixes it with 50% of the train split from the "Synthetic" dataset, while keeping the validation set from the "Base" dataset.	8476
Base + Synthetic	This dataset combines the samples in train split from the "Base" and "Synthetic" datasets. The total number of training samples is 13562. The aim is to test if adding the images generated with programmatic synthesis would improve the performance of the model.	15257
Spatial	This dataset would be the samples from the "Base" dataset, with only spatial transformations (see B) applied to the images. It's purpose is to test the effect of the spatial transformations on the performance of the model.	83067
Pixel	This dataset would be constructed by applying pixel transformations (see A) to the "Base" dataset. It' purpose is to test whether the pixel transformations has a positive effect on the performance of the model.	272935
Full	This dataset combines the "Spatial" and "Pixel" datasets and represents applying all possible transformations to the "Base" dataset. This would test if applying all of the transformations would improve the performance of the model.	354307

Base + Stable Diffusion Full	After some exploratory testing it was shown that training on the “Base + StableDiffusion” dataset was successful in improving the model’s performance on specific use cases, which prompted further testing and building this dataset by applying all the transformations to the “Base + StableDiffusion” dataset, since the “Full” dataset also showed positive results.	419723

Table 2: All the different datasets that YOLOv9 was trained on.

The train/validation split for the ”Base”, ”Synthetic” and ”Base Synthetic 50/50” dataset is 80%/20% respectively. For all of the datasets that have augmented data, the augmentations are applied only to the samples in the train split. The validation split only has real, unaltered data which ensures that the model’s performance metrics reflect its ability to generalize to real world data, not just the augmented variations it was trained on. All of the models trained on these datasets will be tested on the same set of 265 test images obtained from the Open Images v7 dataset.

2.5.1 Transfer Learning

Another important point of consideration is whether to use pretrained weights or to start from scratch. The developers of YOLOv9 released a series of YOLOv9 weights trained on the MS COCO [8] dataset ranging from YOLOv9-t (tiny) to YOLOv9-e (extended). While MS COCO does not contain annotations for flags, starting from these weights can leave the model with some basic ideas of patterns and features which can be helpful for further training. Computer Vision models work by first learning more basic shapes like lines in the first layers of the neural network, before learning more abstract ideas like flags in the deeper layers. Starting training from pretrained weights means that the model already comes with the lower layers pretrained. This technique is called *Transfer Learning*. To add more diversity to the experiments, the model was trained on each dataset twice, once starting from scratch and once starting from pretrained weights, specifically the YOLOv9-e weights.

3 Results

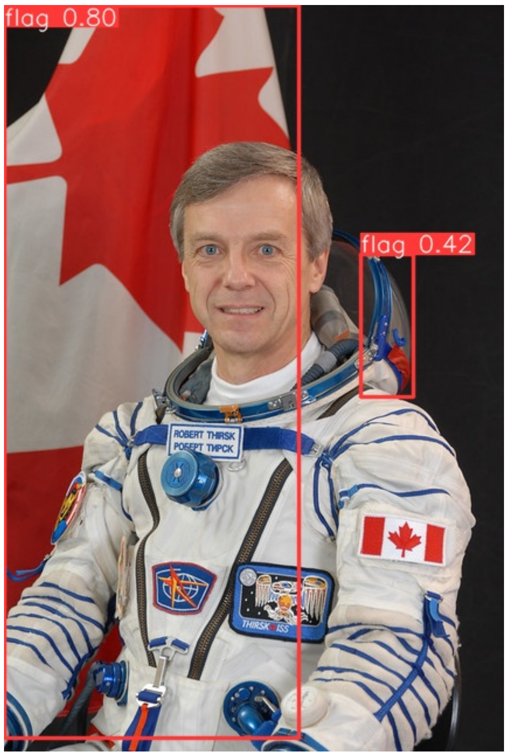
3.1 Experimental Outcomes

The outcomes of all the test runs can be seen in Table 3. The most important metric for analyzing the results is *mAP50-95*, since it takes into account both the *Precision* and *Accuracy* of the model and the overlap between the predicted bounding box and the ground truth bounding box. It is also more strict than *mAP50*, by evaluating the mean average precision on multiple IoU thresholds.

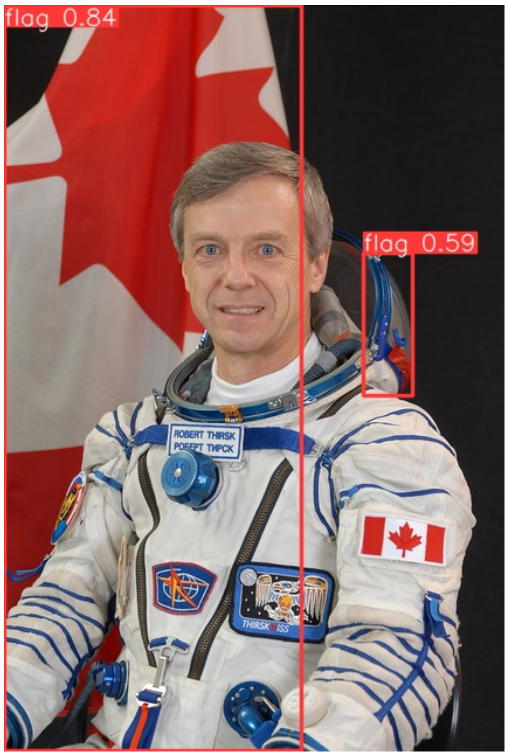
Model	Precision	Recall	F1-Score	mAP50	mAP50-95
Base Scratch	0.7948	0.7500	0.7718	0.7970	0.6351
Base Pretrained	0.7563	0.7846	0.7702	0.7824	0.6214
Base + StableDiffusion Scratch	0.7583	0.8025	0.7798	0.7957	0.6395
Base + StableDiffusion Pretrained	0.7618	0.7718	0.7668	0.7814	0.6391
Synthetic Scratch	0.1111	0.0144	0.0255	0.0604	0.0411
Synthetic Pretrained	0.0969	0.3247	0.1493	0.1140	0.0737
Base Synthetic 50/50 Scratch	0.7597	0.7730	0.7663	0.7507	0.5619
Base Synthetic 50/50 Pretrained	0.7608	0.7769	0.7688	0.7560	0.5819
Base + Synthetic Scratch	0.7877	0.7845	0.7861	0.7773	0.6077
Base + Synthetic Pretrained	0.7635	0.7792	0.7713	0.7745	0.6251
Spatial Scratch	0.7784	0.7960	0.7871	0.7800	0.5972
Spatial Pretrained	0.7256	0.8436	0.7802	0.7927	0.6279
Pixel Scratch	0.7268	0.7413	0.7340	0.7230	0.5235
Pixel Pretrained	0.7601	0.8013	0.7802	0.8053	0.6475
Full Scratch	**0.8103**	0.7443	0.7759	0.7857	0.6006
Full Pretrained	0.7937	0.8075	**0.8005**	**0.8207**	0.6537
Base + Stable Diffusion Full Scratch	0.7797	0.7931	0.7863	0.8048	0.6328
Base + Stable Diffusion Full Pretrained	0.7222	**0.8514**	0.7815	0.7998	**0.6761**

Table 3: Results from different models on the train set.
Numbers in **bold** represent maximum values.

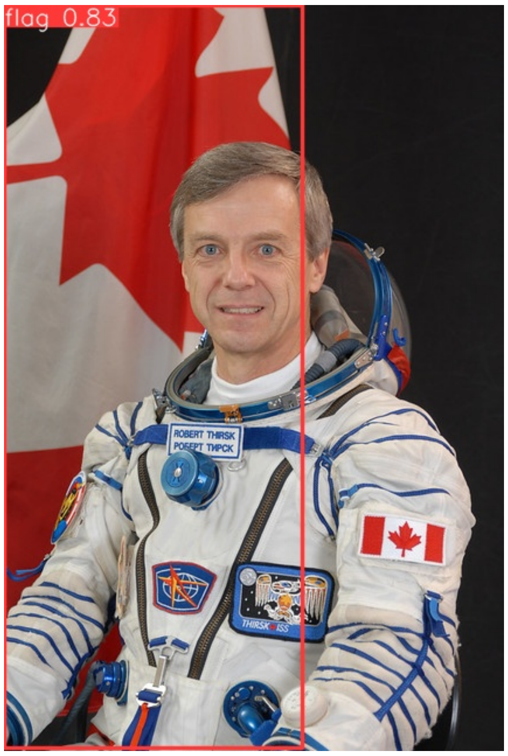
From the Table 3, it is recognizable that the "Base + Stable Diffusion Full Pretrained" model performed the best, scoring the highest mAP50-95 and Recall with a consistently high score in Precision and mAP50. Many of the other models scored high in either Precision *or* Recall leading to their mAP50-95 score not being as high. "Full Pretrained" and "Pixel Pretrained" both resulted in higher mAP50-95 than the "Base Pretrained" showing that adding transformations to the pretrained model can improve the performance. After them are ranked the "Base + StableDiffusion Scratch" and "Base + StableDiffusion Pretrained" models, showing that the the artificial data had a positive effect on the models predictive capability.



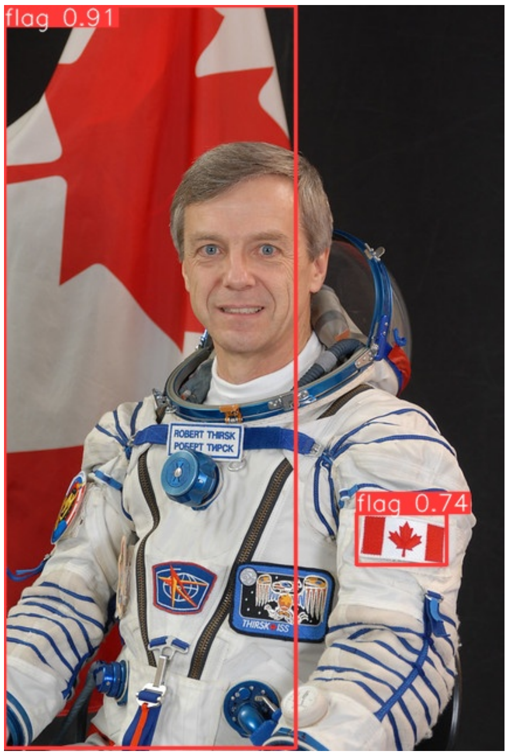
(a) Result from "Base Pretrained"



(b) Result from "Base + Stable Diffusion Pretrained"



(c) Result from "Full Pretrained"



(d) Result from "Base + Stable Diffusion Full Pretrained"

Figure 13: Results from different models on a test image of an astronaut.

3.1.1 Individual Prediction Tests

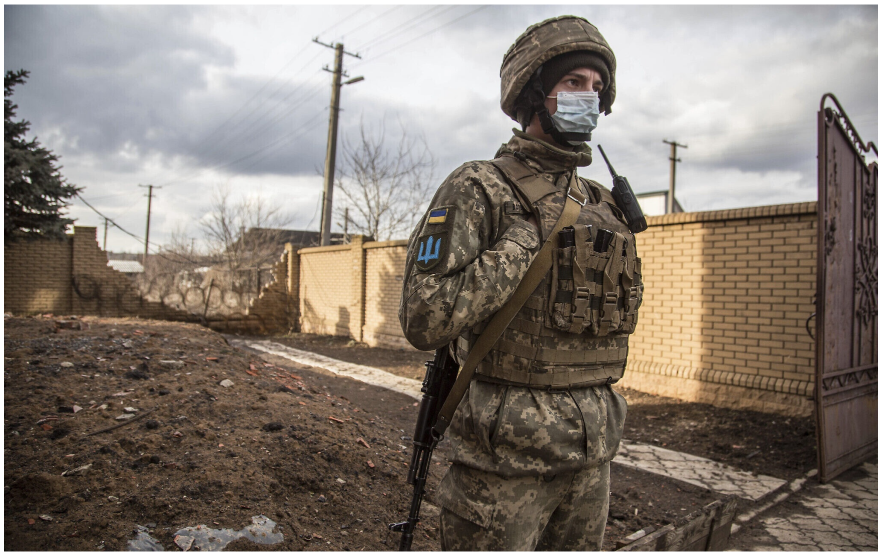
Figure 13 depicts how some of the models performed on one of the images in the test set. The "Base Pretrained" and "Base + Stable Diffusion Pretrained" models both mislabeled part of the astronaut's helmet as a flag, while the model "Full Pretrained" didn't, which is a good example of it's higher Recall and mAP50. However, the only model that correctly labeled the astronauts arm-patch as a flag was "Base + Stable Diffusion Full Pretrained". Some of the images generated with Stable Diffusion depicted soldiers with flags as arm-patches on their shoulders, which is why the model correctly labeled this image. Additionally, the transformations applied to the dataset increased the number of those images, which could be why it performed better than "Base + Stable Diffusion Pretrained". To further test this particular case, the models were tested again on a similar image depicted in Figure 14.

The image features a Ukrainian soldier and the only flag present is the one on his shoulder patch. In this example the models "Base Pretrained" and "Full Pretrained" failed to detect the flag on the soldier's shoulder, while the 2 models trained with the additional Stable Diffusion data, "Base + Stable Diffusion Pretrained" and "Base + Stable Diffusion Full Pretrained", correctly labeled the sample. While anecdotal this example and similar other tests did show promising results that Stable Diffusion and generative AI can be used to create large amounts of training data while having positive results.

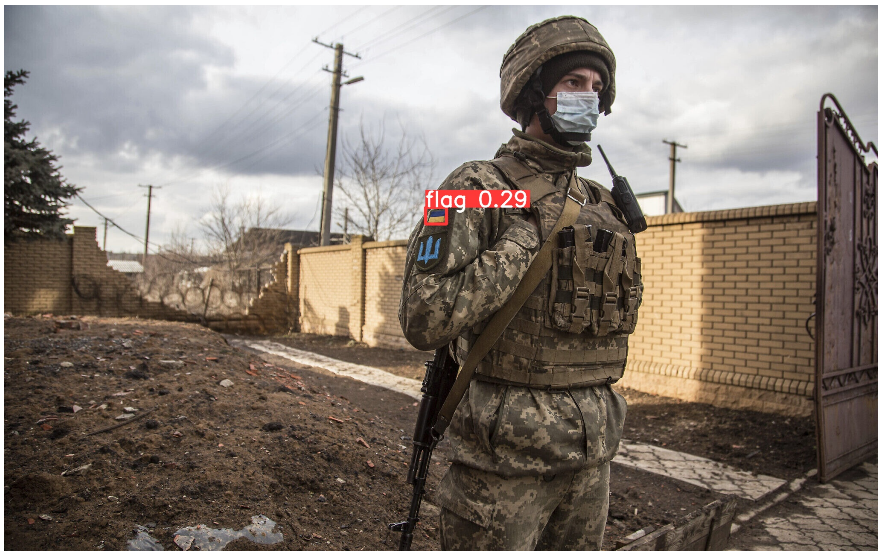
Another interesting example can be seen in Figure 15. The image depicts a balcony with a red and white stripped chair that has a pillow on it. The chair can easily be mistaken as the flag of the United States of America, which is exactly what all of the models did. The chair was labeled as a flag in every test. One of the reasons for this may be the over-representation of the USA's flag in the dataset, with it being depicted in the largest amount of images.

To test this idea, the models were tested again on Figure 16. It features an image similar to Figure 15, however the colors have been shifted so the chair appears to be blue with white stripes. The "Base Pretrained" and "Base + Stable Diffusion Pretrained" models didn't label the chair as a flag, however the former did label part of the railing on the balcony. On the other side, the models "Full Pretrained" and "Base + Stable Diffusion Full Pretrained" did label the chair as a flag. This behaviour makes sense, since these models were trained on color shifted instances of the USA's flag, so they would still detect it in this situation. The models that were trained on a dataset without the image transformations weren't exposed to flags that looked similar to the chair, so those models performed better in this instance.

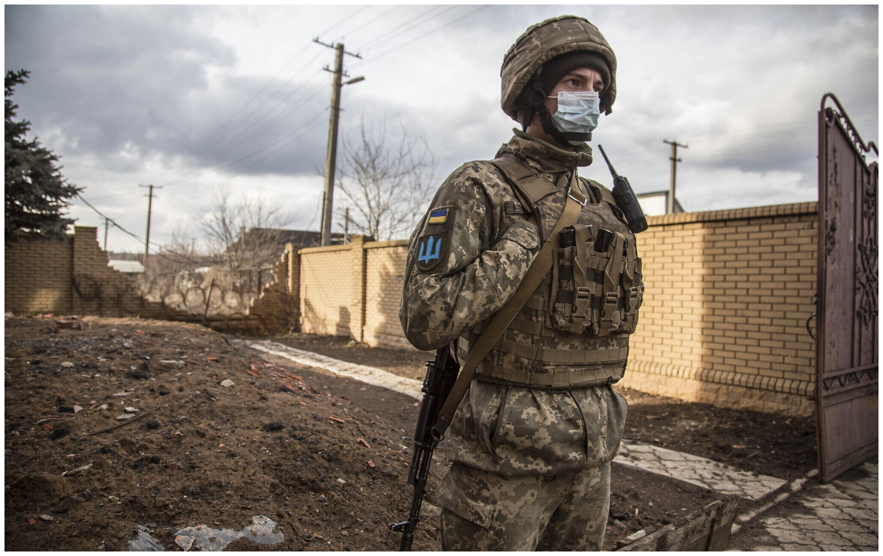
Figure 17 features 2 people parachuting through the air with a flag being carried behind them. This is another example where the model might have problems with false positives, as it could detect the parachutes as a flag. The model that performed the best for this test is "Base Pretrained", since the other models labeled either one or both of the parachutes. This can be attributed to how colorful the parachutes are, which increases their similarity to flags. When tested on a similar image with a gray parachute all the models performed positively and only labeled the flag. This can be seen in Figure 18.



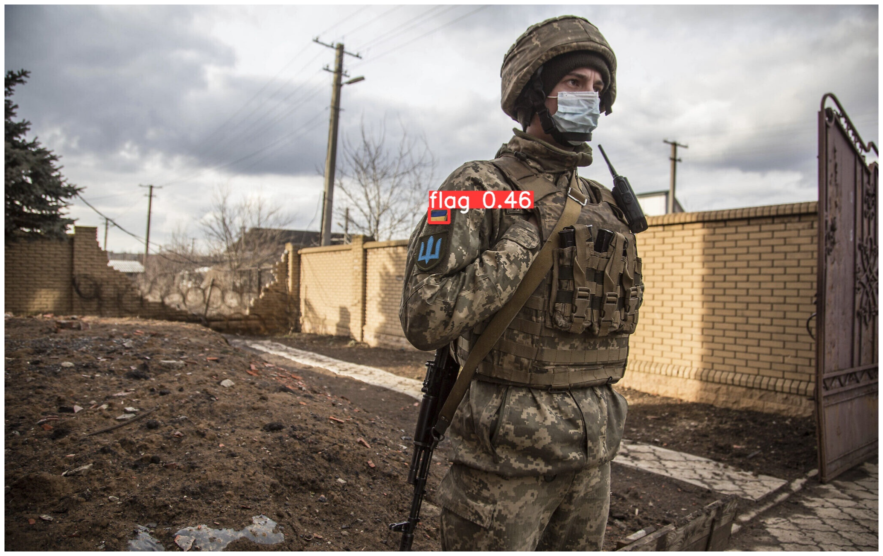
(a) Result from "Base Pretrained"



(b) Result from "Base + Stable Diffusion Pretrained"

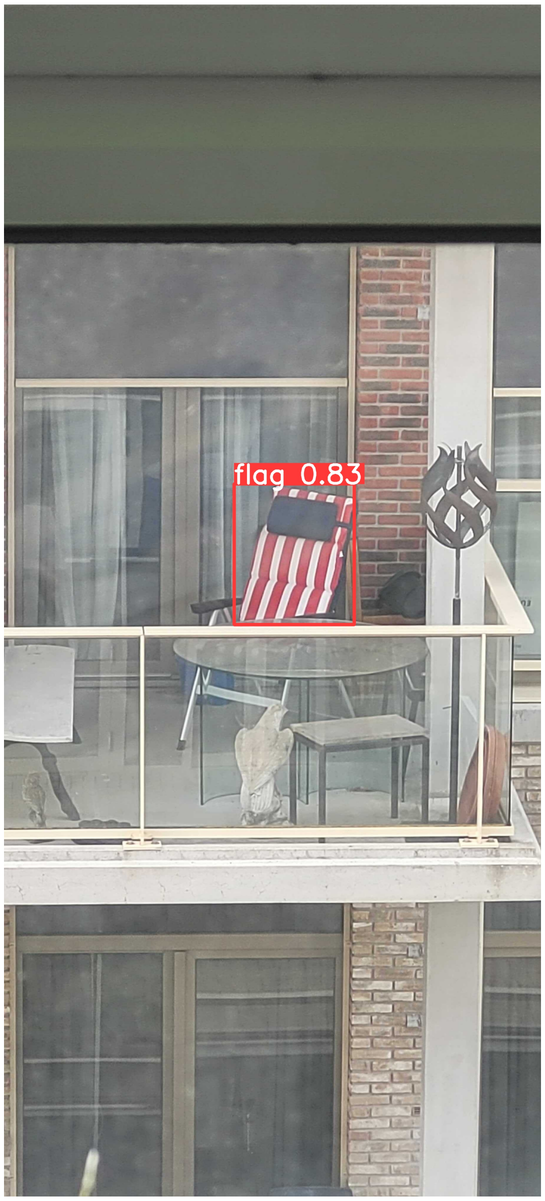


(c) Result from “Full Pretrained”

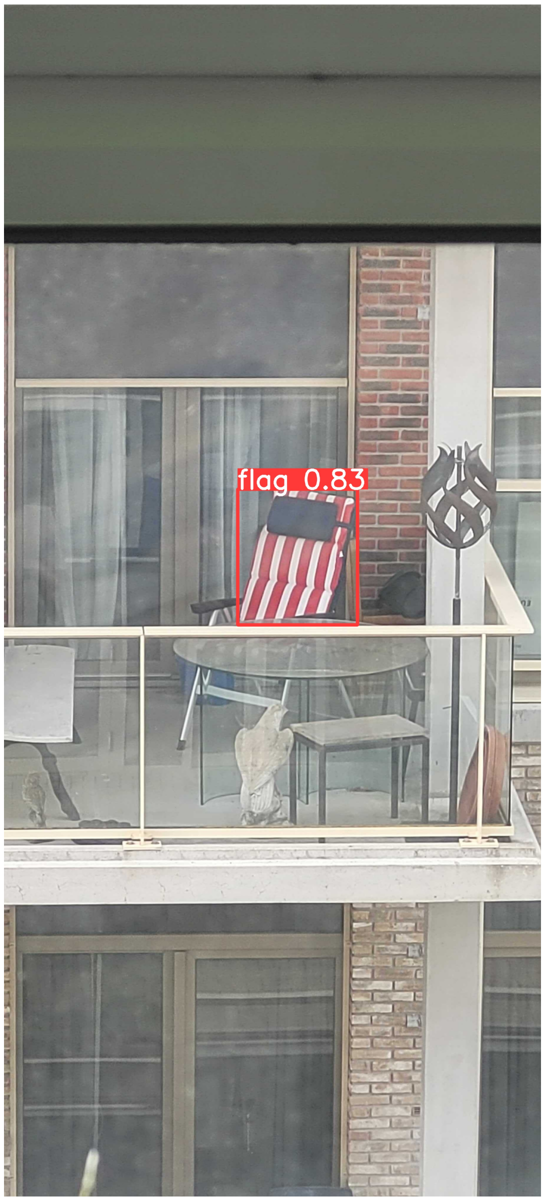


(d) Result from “Base + Stable Diffusion Full Pretrained”

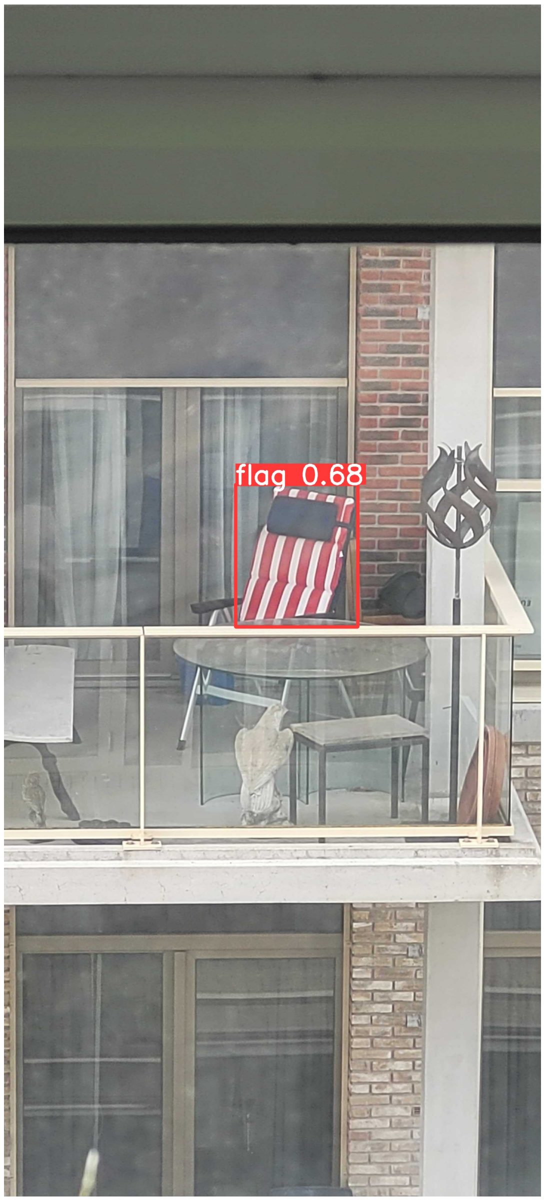
Figure 14: Results from different models on an image that that features a Ukrainian soldier.



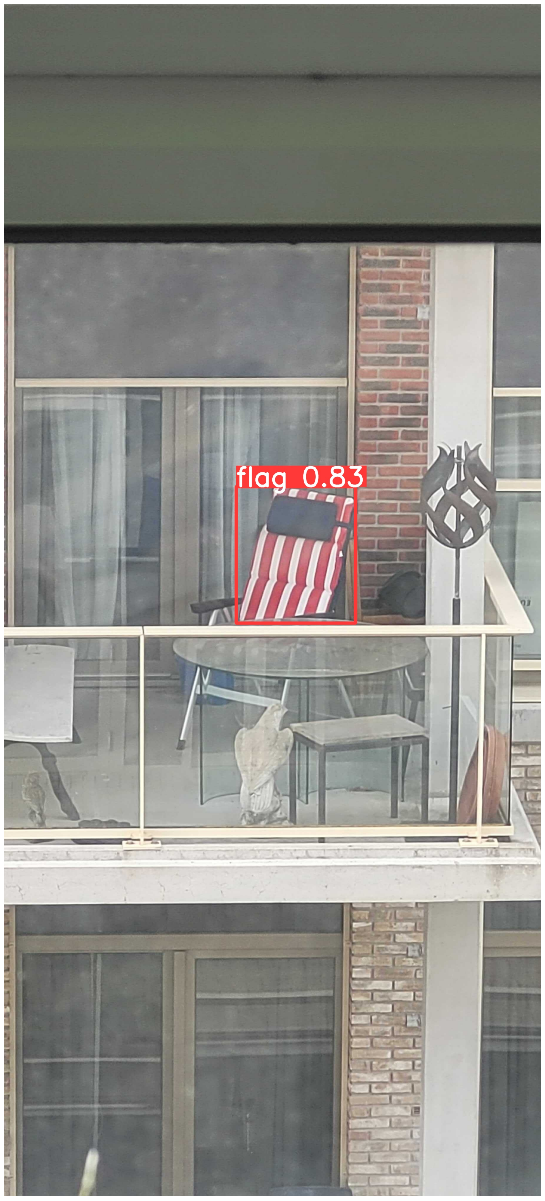
(a) Result from "Base Pretrained"



(b) Result from "Base + Stable Diffusion Pretrained"

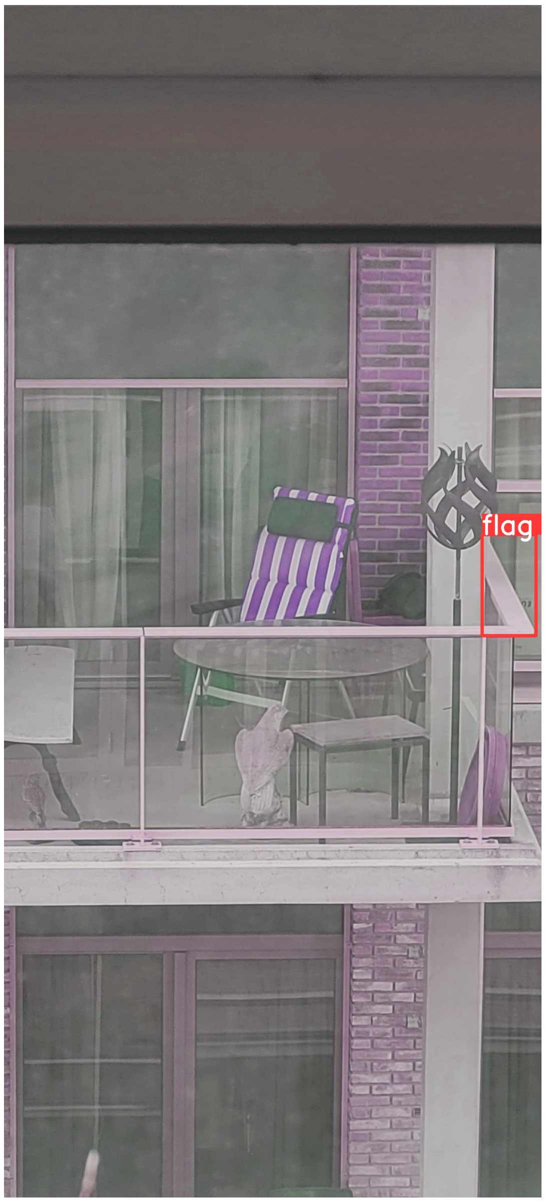


(c) Result from "Full Pretrained"

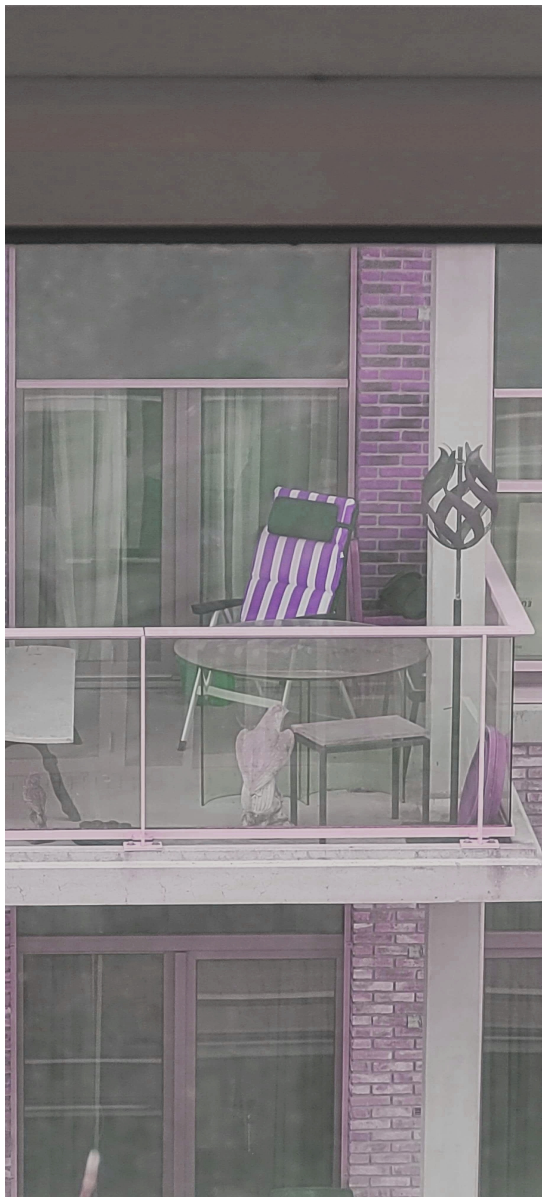


(d) Result from "Base + Stable Diffusion Full Pretrained"

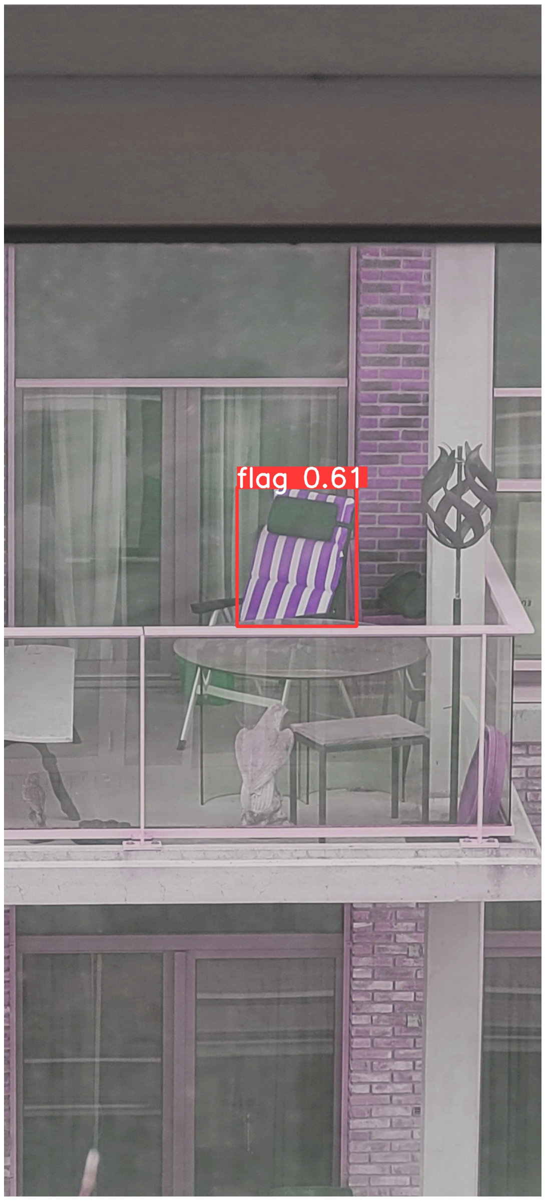
Figure 15: Results from different models on an image that features a chair resembling the USA's flag.



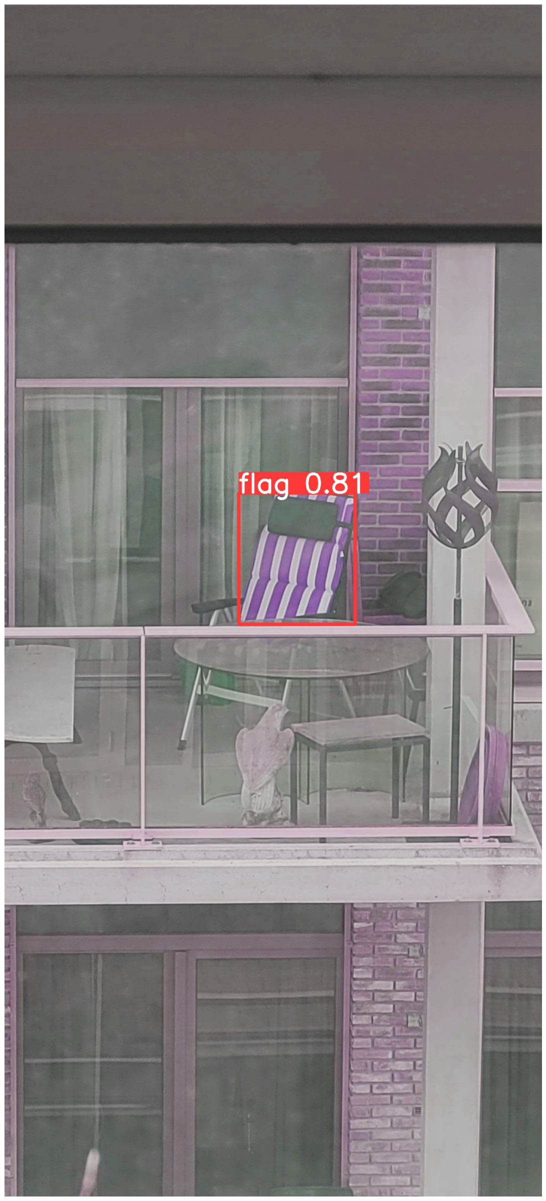
(a) Result from "Base Pretrained"



(b) Result from "Base + Stable Diffusion Pretrained"

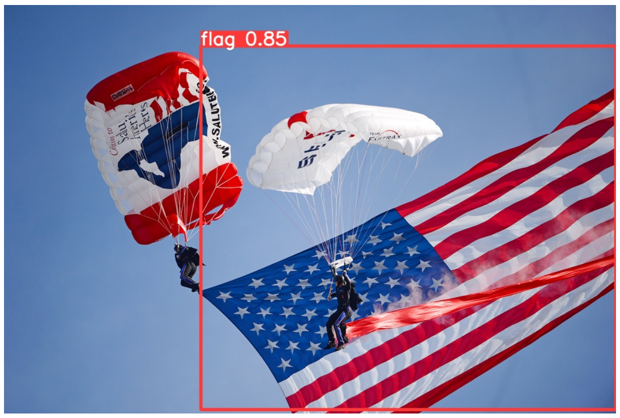


(c) Result from "Full Pretrained"

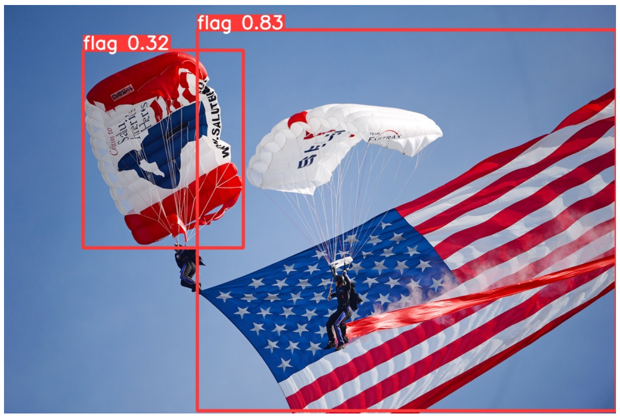


(d) Result from "Base + Stable Diffusion Full Pretrained"

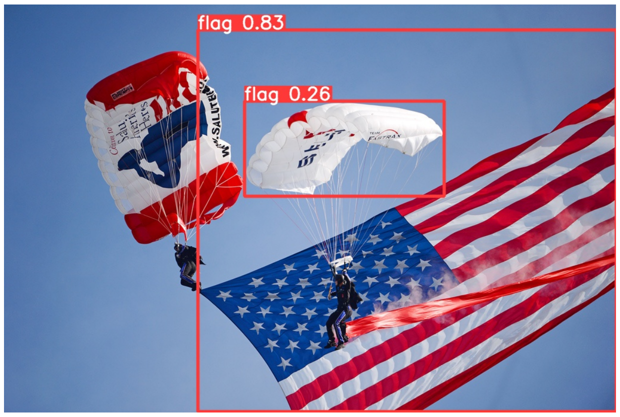
Figure 16: Results from different models on the image from Figure 15, but with a *ColorJitter* transformation applied to it that makes the chair look blue.



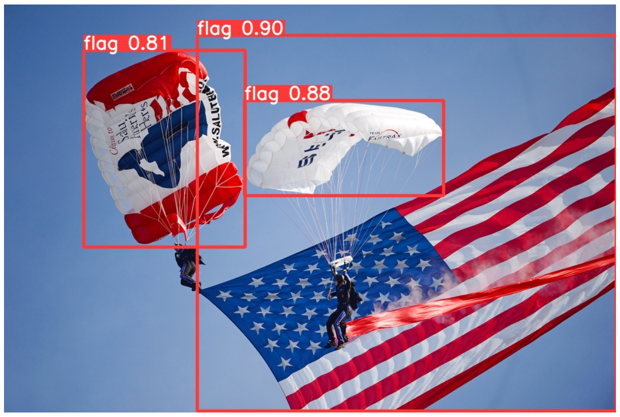
(a) Result from "Base Pretrained"



(b) Result from "Base + Stable Diffusion Pretrained"

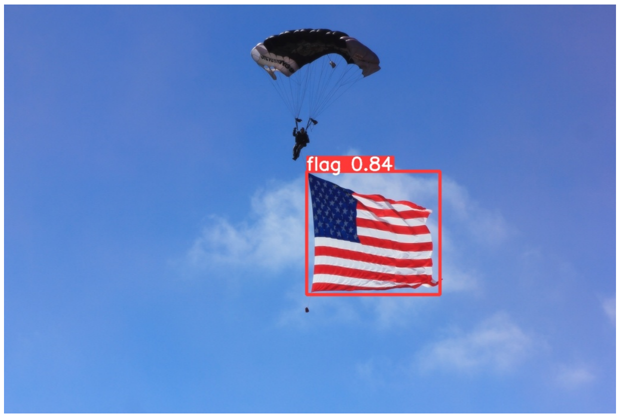


(c) Result from "Full Pretrained"

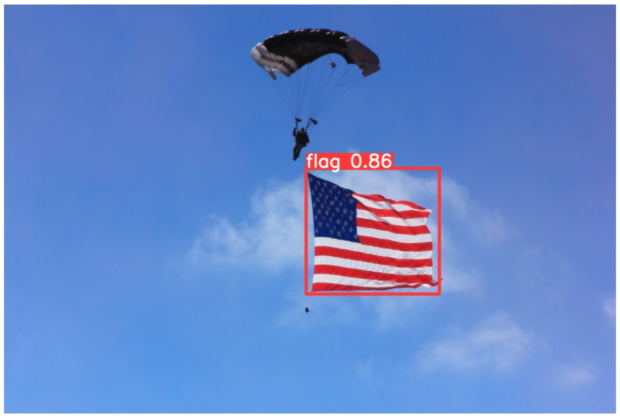


(d) Result from "Base + Stable Diffusion Full Pretrained"

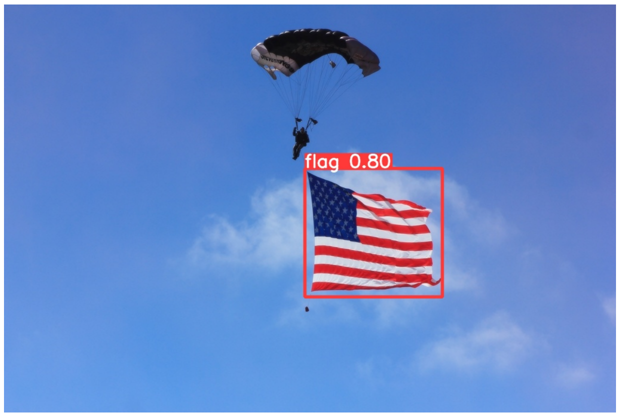
Figure 17: Results from different models on an image featuring people parachuting with a flag behind them.



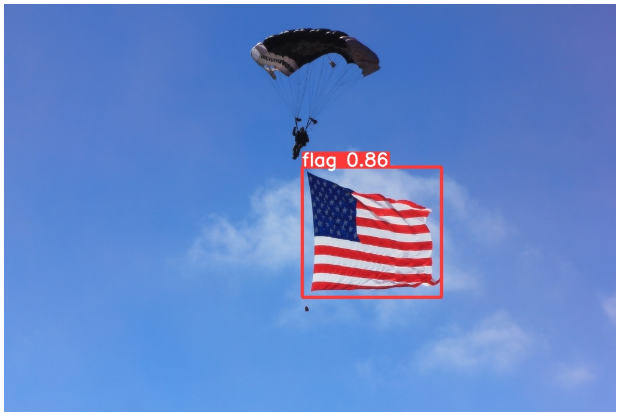
(a) Result from "Base Pretrained"



(b) Result from "Base + Stable Diffusion Pretrained"



(c) Result from "Full Pretrained"



(d) Result from "Base + Stable Diffusion Full Pretrained"

Figure 18: Results from different models on an image featuring people parachuting with a flag behind them.

3.1.2 Benchmark tests

To test whether the resulting model was suitable for commercial use, the “Base + Stable Diffusion Full Pretrained” model was benchmark tested on a 1000 random images from its train set. The goal of this was to calculate the average inference time, to see how quickly the model could process requests. The resulting average inference time was *1.705* seconds with a standard deviation of *0.208* seconds.

This inference time leaves room for improvement. A possible cause of the slower performance could be the use of YOLOv9e. Switching to a smaller version like YOLOv9c (compact) or YOLOv9m (medium) could possibly improve this.
The benchmark tests were run on a system with the following specifications:

- CPU Model name: AMD Ryzen 7 6800HS Creator Edition
- Max CPU clock speed: 4785,0000
- RAM: 13976696 kB
- Type of hard-drive: HDD
- OS: Ubuntu 22.04.4 LTS

4 Conclusions

Going back to the research questions ,"*What is the performance of YOLOv9 when used for binary flag detection in real world images?*" and "*Which data enhancement methods best increase the performance of the model?*", we can now provide answers based on the conducted experiments.

YOLOv9's performance can be seen in table 3, where the models trained on datasets constructed with the different data aggregation methods, worked on par with and sometimes improved on the performance of the models trained on just the base dataset. The data augmentation methods showed the best results, with the "Full Pretrained" model scoring the best *mAP50* of 0.8207 and the "Pixel Pretrained" model scoring second best with 0.8053. The generative AI data synthesis showed promising results, scoring on par with the "Base" models, while also performing positively on individual prediction tests as shown in 3.1.1, indicating that Generative AI may be used as an additional source of data for flag detection. The programmatic data synthesis under-performed, scoring lower in all tests than the "Base" models indicating that it isn't as suitable for large scale data generation, like the generative AI, at least in it's current state.

The end product of this project is a model that is capable of producing outputs like the one in figure 19.

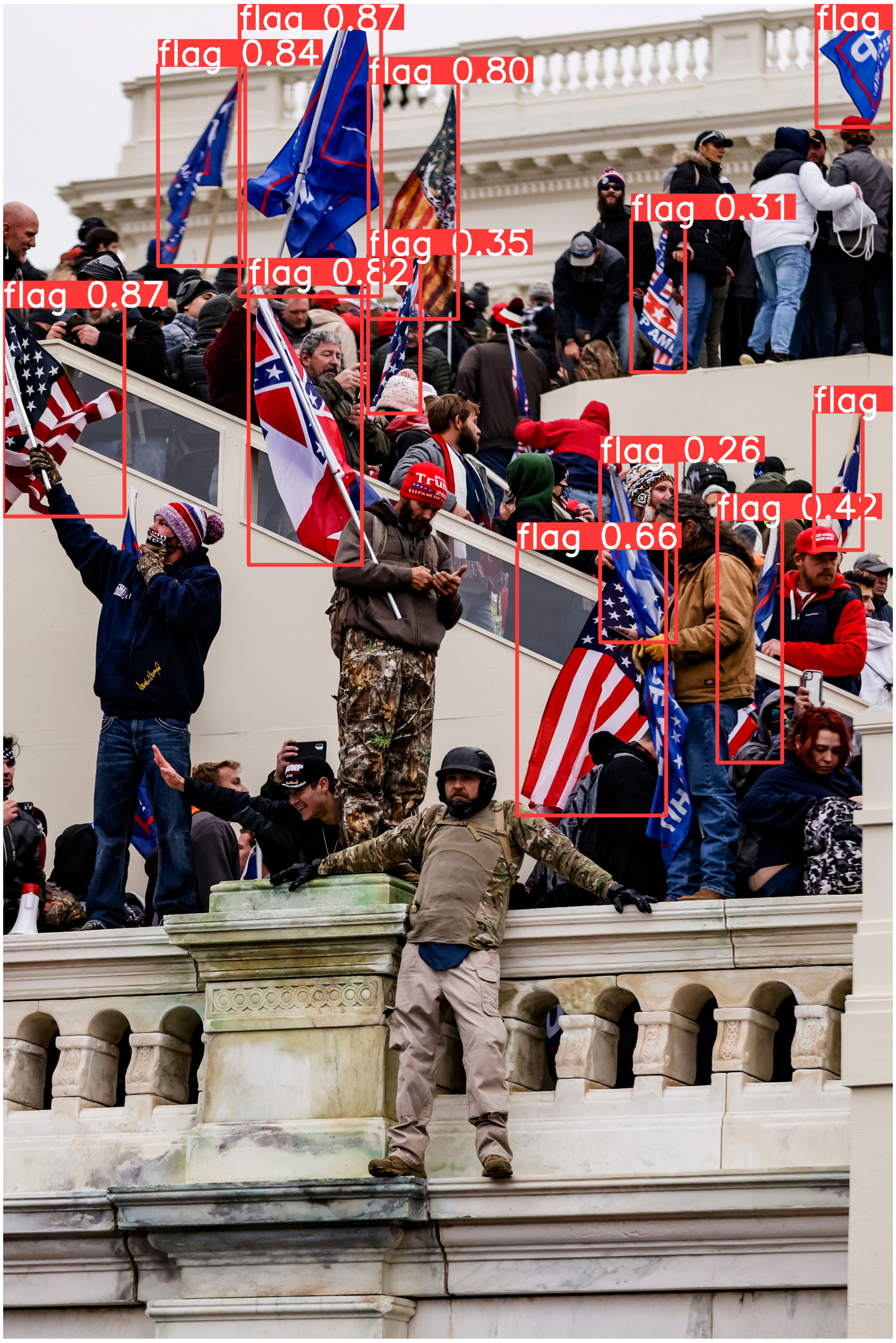


Figure 19: Results from "Base + Stable Diffusion Full Pretrained" on an image of the January 6th Capitol Riots

5 Future Work

The quality of the programmatic synthetic data can be improved by using more complex methods involving 3D computer graphics to make the flags more lifelike. Additionally, color matching/blending, by analyzing the colors in the background and adjusting the colors of the pasted flag, can be used so the flags fit in better with the background image. This would also increase the quality of the data, which should in turn result in better performance of the models trained on that data.

Further research into using generative AI along with targeted prompts to generate training data for specific use cases, since initial testing showed positive results. This could be used to generate large scale data or address class imbalances in datasets. Additionally at the time of writing there hasn't been a wide release of Stable Diffusion 3, however the model seems promising and should perform better than Stable Diffusion XL.

As work on this project was concluding, YOLOv10 was released which has better performance metrics and faster inference times when compared to YOLOv9, which should make it more suitable for commercial use. Repeating some of the most promising experiments for YOLOv10 should result in even better values.

The "Full Pretrained" model, when bench-marked on 1000 images, had an average inference time of 1.659 seconds per image. While adequate, this can certainly be improved, especially if the model is to be intended in a commercial setting. Switching to one of the smaller YOLOv9 models, like YOLOv9C or YOLOv9M should improve this.

References

[1] Albumentations. Albumentations full api reference, 2024.

[2] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[3] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.

[4] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei. Yolov6: A single-stage object detection framework for industrial applications, 2022.

[5] Jizhizi Li, Sihan Ma, Jing Zhang, and Dacheng Tao. Privacy-preserving portrait matting, 2021.

[6] Jizhizi Li, Jing Zhang, Stephen J. Maybank, and Dacheng Tao. Bridging composite and real: Towards end-to-end deep image matting, 2021.

[7] Jizhizi Li, Jing Zhang, and Dacheng Tao. Deep automatic natural image matting, 2021.

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[9] Dynamo Spanish. Dynamo spanish, 2024.

[10] Shou-Fang Wu, Ming-Ching Chang, Siwei Lyu, Cheng-Shih Wong, Abhineet Kumar Pandey, and Po-Chi Su. Flagdetseg: Multi-nation flag detection and segmentation in the wild. In *2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8, 2021.

[11] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276, 2023.

A Pixel Transformations

All pixel-level transformations applied to the base dataset. Further explanation for each transformation can be found at [1].

Blur	CLAHE	ChannelDropout
ChannelShuffle	ChromaticAberration	ColorJitter
Defocus	Downscale	Equalize
Downscale	Equalize	GaussNoise
GaussianBlur	GlassBlur	FancyPCA
HueSaturationValue	ISONoise	ImageCompression30
ImageCompression50	InvertImg	MedianBlur
MotionBlur	Posterize	RGBShift
RandomBrightnessContrast	RandomFog	RandomGamma
RandomGravel	RandomRain	RandomShadow
RandomSnow	RandomToneCurve	RingingOvershoot
Sharpen	Solarize	Spatter
ToSepia	ToGray	UnsharpMask
ZoomBlur		

Table 4: All Pixel-Level transformations

B Spatial Transformations

All spatial-level transformations applied to the base dataset. Further explanation for each transformation can be found at [1].

ElasticTransform	BBoxSafeRandomCrop	GridDistortion
Flip	HorizontalFlip	LongestMaxSize
OpticalDistortion	Perspective	RandomRotate90
Transpose	VerticalFlip	Rotate

Table 5: All Pixel-Level transformations

C Stable Diffusion prompts

Amount of Images Generated	Prompt	Negative Prompt

100	A Photogarph of a goverment individual with a flag behind them, photography for an identification card, medium shot, portrait, photo, CANON EOS R3, 80mm, 1/125 Sec shutter speed	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor
120	A Photograph of a country's national flag waving in the wind on a blue sky, high angle, photo, CANON EOS R3, 80mm, 1/125 Sec shutter speed	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor
90	A photograph of a goverment building with the national flag infront of it, wide shot, photography, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor
80	A photography of a city with the flag of the coutry the city is in, wide shot, photography, flag, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor
80	A photography of a landscape of a country with the countrys flag, photography, flag, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor
120	A conference room with national flags set on the wall, two country leaders shaking hands over the table, photography, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed, high detailed skin	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor, disfigured, ugly, bad, b&w, hands
120	A person giving a speech with their country's flag behind them, photography, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed, high detailed skin	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor, disfigured, ugly, bad, b&w
120	A photography of soldiers on a base saluting a flag, photography, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed, high detailed skin	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor, disfigured, ugly, bad, b&w
80	A ship on the sea with it's country's flag on it, photography, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor, disfigured, ugly, bad, b&w
110	A photo of a politician giving a talk with his country's flag behind him, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed, high detailed skin	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor, disfigured, ugly

80	A busy city street with a flag on a building, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed, high detailed skin	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor
100	A protester carrying his country's flag, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed, diverse people	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor, b&w
100	A soldier having his picture taken with his country's flag behind him, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed, high detailed skin	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor, b&w
100	An id photograph of a military peronnel wih a flag behind them, photography, camera, CANON EOS R3, 80mm, 1/125 Sec shutter speed, high detailed skin	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor, disfigured, ugly, bad, b&w, hands
70	A flag hoisted bove a town square, the flag is colorfully waving in the wind, full shot, high angle, the anthosphere is relaxed imitation a cozy saturday sunset, photography, photo, camera image, CANON EOS R3, 80mm, DSLR, 1/125 Sec shutter speed, golden hour	art, painting, drawing, anime, cartoon, low poly, fantasy art, watercolor

Table 6: All stable diffusion prompts used to generate the dataset.