Image Analysis Project – Mathias Walter Nilsen

Project description:

Object detection often requires large amounts of pre annotated data. Something that is usually challenging to find in addition to it being of high quality. Furthermore, annotating such object in images manually often require large amounts of time as this process is both tedious and slow. Especially when annotating is performed using polygons instead of setting bounding boxes over an area. Thus, creating a synthetic dataset from a smaller sample dataset would alleviate the required time to manually annotating the images.

The main goal of this project is to explore the possibility of utilizing synthetic datasets in training object detection. A synthetic dataset is a dataset that is automatically generated based on another dataset as a sample. Creating such a dataset could be achieved by first cropping an object, augmenting this, and implanting the result into a background. Another way is to use the Generative Adversarial Network(GAN) to mimic the original dataset. However, the latter method is more time-consuming, and as such, this project will use the first method mentioned. The object detection algorithm used a variant of instance segmentation called mask R-CNN R101-FPN, made available by the Facebook detectron2 project, link in the footer.

Dataset:

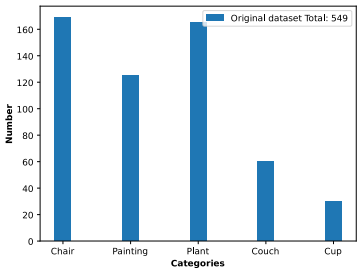

Figure 1 - Class distribution in original dataset

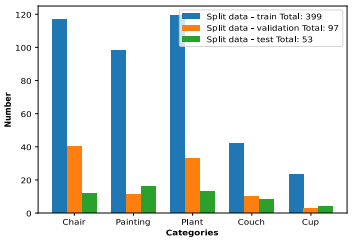

Figure 2 - Class distribution split original dataset

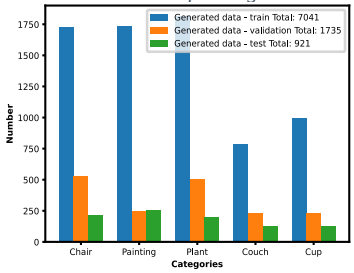

Figure 3 – Class distribution split generated dataset

The original dataset was created by first procuring 173 open-source images that contained the desired target objects. These images show a variety of different brightness, object colors, and shapes. Hence, making this a balanced dataset. Figure 1 displays the class distribution in the procured dataset. Annotating these were performed using the online tool, MakeSense. The process for splitting is done by first randomly splitting the original dataset into a training, test, and validation set, 70-20-10, respectively. In addition to this, the same dataset split is used throughout the testing and contains only real images.

Thus, showing the impact of each change on the training process.

The process for generating the synthetic data is done by first cropping the annotated foreground objects from the original dataset using masks. The next step consists of augmenting this object using several types of randomized augmentations. This augmented object is stored on disk and is the object mask used for embedding the image into a background. The following augmentations are performed on the foreground objects, flipping horizontally and vertically, adjusting the colors, rotating, and skewing the image. The foreground object is then supplanted into a random background, belonging to

Detectron2: https://github.com/facebookresearch/detectron2

the corresponding data split. Additional augmented objects are embedded on a random basis, chosen from the same split dataset. This method prevents data leakages from occurring. Furthermore, all images are resized to either a max width or height of 1000, which is executed using the script called "resize_images.py." In addition to this, the process for generating the synthetic data is executed using the script, "preprocess.py." Figures 2 and 3, display the class distribution in both the original split dataset and the generated split dataset respectively. Furthermore, both the validation and test set will only consist of real images.

Training with synthesized data:

The baseline model of which to test against was trained using only real images and using the same amount of iterations. It is observed that even without any alterations, the scored themselves were fairly high, as seen in figure 4, given the limited real dataset. Training with synthesized data was done using three different configurations, one with color alterations, no color alterations, and scaling the objects with no color alterations. However, augmentations like flipping, rotation, and skewing are applied to all configurations on default. The results from each of these alterations are depicted in figure 4. Overall it is observed that training using synthesized data yielded significantly lower results on all types of alterations. Possible reasons behind this are discussed later in this report. However, there were fascinating results observed in these training runs. An observation is that using color alterations yielded decent scores in the following classes, "chair," "cup," and "couch." The aforementioned is expected and is probably due to those classes having ranging colors in real life. However, the following targets, "plant," and "painting," received poor accuracy. The reason behind this is inferred based on a manual inspection of the augmented objects. The color alterations make the objects themself challenging to identify due to the concentrated color alterations. Thus, making it difficult to find and extrapolate the required patterns from these objects.

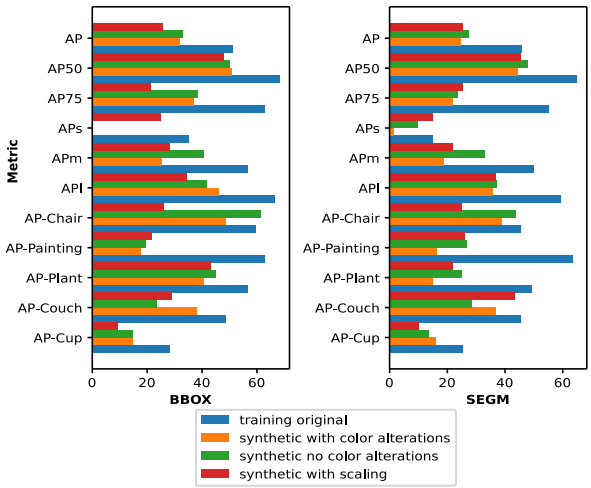

Figure 4 – Results from training with varuious configurations

Removing the color alterations caused higher accuracy to the classes, "painting" and "plant." However, the class "cup" received the same score, which indicates that this class is not affected by the color alteration. Furthermore, the class chair received a significantly higher accuracy score than that of using only real images. Hence, suggesting that removing the color alteration led to an increased score for this class. Running the training using random scaling and no color alteration yielded a significant increase in score for class "couch." Running the training using random scaling and no color alteration produced a significant increase for only one class, "couch." The resulting score is comparable to that of the non-synthetic training data. Furthermore, the class "painting," also had an increase. However,

this is an insignificant increase compared to training with original data. In addition to this, the accuracy for small objects also increased significantly. Thus, making this alteration valuable for learning targets of different sizes. However, this further suggests that the main problem lies not with the generation and applying the various alterations, but with the background to foreground size ratio and how foreground objects are embedded.

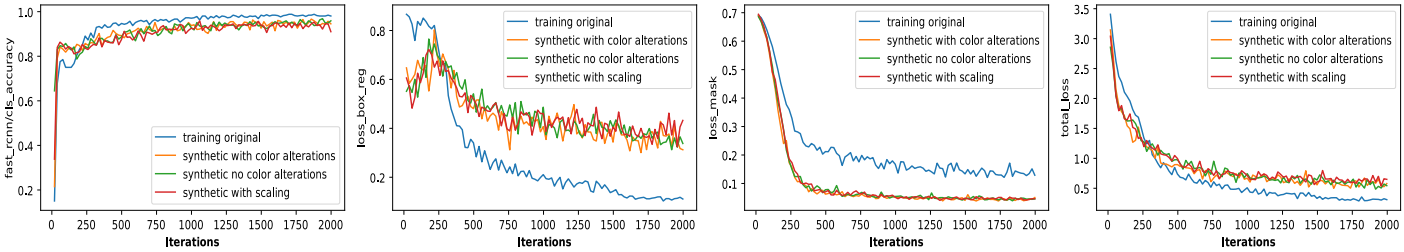

Figure 5 – Showcases loss and accuracy corresponding to the varying training iterations.

Additional results

Figure 5 depicts loss for both bounding box and segmentation, total loss, and accuracy, corresponding to the different training iterations during the training process. Using this figure reveals some deviations when training with synthesized data. One observed deviation is that the bounding box loss converges to a significantly higher point than that of the real training data. Furthermore, the mask loss is converging to a significantly lower point than that of the real training data. Hence, suggesting that the model is struggling to detect the foreground objects. However, when it does, the model seems to overfit on some of the images, which causes an abnormally high accuracy on the validating set. Thus, leading to a lowered accuracy on the testing data.

A manual inspection of the inferred images reveals that the model is overly optimistic in categorizing other objects that are not one of the target labels. Thus, causing the model to receive less accuracy across all metrics. However, there are very few of the target objects that do not get labeled. The aforementioned is also the case for the segmentation masks. Additionally, the model is highly accurate whenever the test image contains only target objects or few non-target objects. The reason behind this probably stems from the that the majority of the synthesized training data includes only target objects.

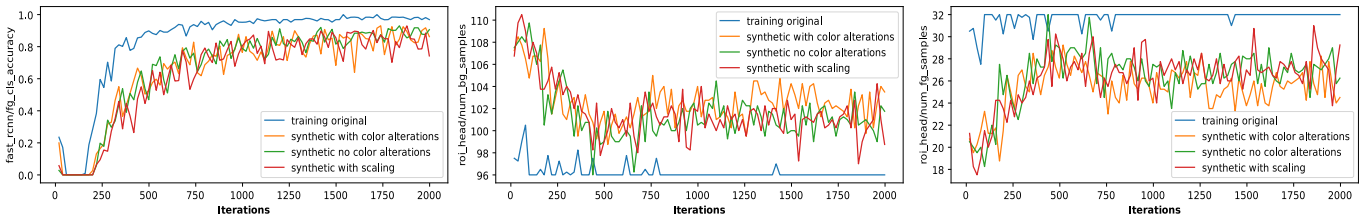

Figure 6 – showcases foreground accuracy, background and foreground samples corresponding to the varyious training runs.

Figure 6 depicts the foreground accuracy, number of generated background, and foreground samples. This figure contains some deviations from the synthetic training data set in contrast to the real training data. A difference is that training with synthesized data causes the model to struggle to separate the foreground from the background, as inferred from the high discrepancy in the foreground accuracy. Something that is further suggested by the disparity in the number of foreground and background samples, in contrast to training with the original data. It is further worth mentioning that decreasing the

background image sizes to max 700 width and height caused a significant increase in overall accuracy and foreground accuracy. Thus, suggesting that the foreground background-size difference has a high significance when training concerning this test set.

Conclusion

There are several ways of improving the performance of synthetic datasets. Customizing the augmentations, and their corresponding hyperparameters, such that they benefit the class they augment, is one way. An example of this is to perform both the color and scaling alterations to the target class "chair." In addition to also tune the hyperparameters of the alterations. Another way of improving the performance is to vary the size foreground to background-size in the synthetic dataset. Thus, allowing the model to detect additional foreground objects while reducing overfitting.

Furthermore, reducing the over-optimistic detection of foreground objects in the test data could be done by embedding additional non-target objects into the foreground in the synthetic dataset. Using supplementary backgrounds should also increase the foreground detection. These added foreground objects should mimic the non-target objects that are also present in the test data. Hence, allowing the model to train under similar enough context and potentially increase the number of detected foreground samples. Increasing the number of ROIs could also potentially solve this. However, this would also potentially increase the accuracy of the real training dataset. Furthermore, reducing both image and background sizes would be more beneficial than increasing the number of ROIs. The benefit comes from that increasing the number of ROIs also requires prolonged computation.

Another way of increasing the foreground detection while decreasing overfitting is to use a Poisson embedding instead of naively cropping and embedding the object. This method allows the foreground to have its pixels transformed with respect to the target domain. Something that is performed by mixing the source and target gradients. However, this might generate strange embeddings when the difference between the target and source is very high.

Improving embedding of foreground images could be performed with GANs. These would be able to generate similar pictures using the original dataset. However, this method might require additional images and is computationally expensive, in contrast to the naive embedding or Poisson blending. Hence, making alternative unsuited for this project, given the imposed time constraint.