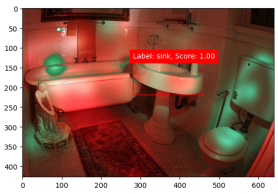
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

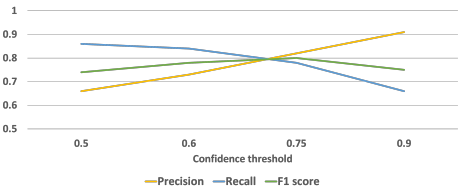
http://fit.vut.cz

Project practice
2023/2024

Search for the Specific Object in the Image

Sandis Studers





Abstract

This paper details the exploration of pre-existing tools to solve a given problem of recognizing objects in a room setting. It explores existing datasets and pre-trained models and attempts to obtain a better performance for the task by fine-tuning a chosen pre-trained model on a prepared subset of room images from the pre-existing dataset.

A subset of 1008 room images is obtained from Common Objects in Context 2017 dataset. The subset contains 3215 object instances belonging to 10 different classes. Various pre-trained models are evaluated on the subset in accordance to obtained F1 scores and recall. Torchvision's offered pre-trained detection model based on Faster R-CNN architecture is chosen as the best performing and a fine-tuning attempt is made on it with the same subset. Fine-tuning results in a slightly decreased overall performance of the model while also showing slight performance improvements for some of the classes leaving the possibility that an improved dataset could lead to improvements in other classes as well.

Keywords: Object detection — Fine tuning — Pre-trained model

Supplementary Material: *N/A*

1. Introduction

The problem that serves as the motivation for this paper is the organization of photographs in a forensic setting, particularly photographs that document rooms of crime scenes. When the photographs are taken, afterwards they manually need to be organized: pictures that belong to the same scene or the same room within the same scene can be grouped together while others are sorted elsewhere. While this process is manual, it could be automated.

It is the aim to solve this organization problem through the usage of a neural network model which would be able to identify objects appearing within the photographs of the room. If each photograph has a certain set of objects appearing in it, then photographs that share exactly the same set of objects are likely to be depicting the same room. In this way photographs can be automatically tied to certain rooms and photographs that depict the same rooms can be tied together. This paper seeks to deal with the creation of a neural network model that can recognize objects commonly found in a room setting.

The approach the paper takes is to use already existing datasets and pre-trained models as the basis and then seek to improve upon them if possible. With the difficulty of creating new models from scratch – that includes the creation of a comprehensive, annotated dataset, specification and implementation of a model architecture, hyperparameter tuning and training – there can be great worth in looking at the increasing abundance of already available tools, datasets and pre-trained models that can greatly speed up the process and perhaps yield a better performance than

a model created with limited resources. This paper seeks to explore the capabilities of existing tools and resources and to see, how they can be used together and applied to this specific problem and whether they can be improved upon with further fine-tuning.

The main tasks set out for this project are:

1. Use existing datasets to obtain a set of annotated room images that includes 10 different object classes and consists of at least 1000 images;
2. Find a pre-trained detection model which yields the best performance on the obtained dataset in terms of F1 score and recall;
3. Attempt to fine-tune the selected detection model with the obtained dataset to observe whether the model's performance can be further improved in the context of object detection in a room setting.

2. Dataset

2.1 Selection of an external dataset

The prefered requirements for the dataset was that it would include enough classes of objects commonly found within room setting and that the images where these classes were detected would also themselves be taken in ordinary room setting.

In the end, the dataset chosen for this task was Microsoft's Common Objects in Context (COCO) dataset. This dataset includes more than 200 000 labeled images and has 91 object categories. The categories consist of commonly encountered objects such as transportation vehicles (such as bike, car), road infrastructure elements (such as traffic lights, street signs), animals (such as cat, elephant), clothing items and accessories (such as umbrella, shoe), kitchen utensils (such as fork, knife), food items (such as apple, donut), furniture (such as chair, couch), electronics (such as laptop, toaster) and various other items (such as book, teddy bear). [2]

The COCO dataset exists in multiple different versions and the version used for this project was COCO 2017 which consists of 123 287 images and 886 284 object instances. [2]

2.2 Selection of classes

Several criteria were taken into account while selecting the classes: most importantly the objects had to be commonly found in room setting, objects should be of various sizes and should appear in various room settings.

The size variability helps to mitigate the influence that size can have on the model's performance. Smaller objects may have worse performance than large objects, for example, cups can be quite uniform in generic in shape and if they are sufficiently far from the perspective of the photo they can be quite easily mistaken for another object, or another object can be easily mistaken for a cup.

Objects were also chosen from various room settings just to make the model usable in different rooms. The room settings accounted for are:

- Living room/Bedroom (Bed, chair, TV, couch, laptop, clock, cup);
- Kitchen (Chair, dining table, cup, sink, refrigerator);
- Bathroom (Chair, cup, sink).

Based on these criteria the selected classes from COCO dataset were:

- Bed;
- Chair;
- TV;
- Dining table;
- Couch;
- Laptop;
- Clock;
- Cup;
- Sink;
- Refrigerator.

2.3 Selection of a subset

The COCO dataset was handled through the FiftyOne library which is also the official partner of COCO and allows to easily download and visualize their datasets. [3]

Using the library 5000 samples were loaded from COCO 2017 dataset's training split that included the previously selected classes. The visual interface offered by FiftyOne library was utilized to view the obtained dataset and pick out a suitable subset of photos that would consist of at least 1000 images and include each of the 10 chosen classes in at least 100 images of the subset. An example image can be seen in Figure 1. [3]

Figure 1. An example of an image and its labels that were included in the subset.

The images of the subset were also selected according to additional criteria:

- Images are taken within a room and of the room: this criteria is the most important and is basis for how the obtained dataset can emulate the images that the model can encounter in forensic setting. Choosing such images also excludes instances of objects that are not likely to be observed in room setting but may be observed in other settings, such as beach chairs for chair class or clock towers for clock. Focusing on objects within room setting can help the model obtain a better performance.
- Images are not a close-up of any object within the room: as the images encountered in forensic setting will likely strive to encompass the room as a whole, pictures that are close ups may confuse the model. For example, if there is a picture taken of some objects in a bed, the bed takes up the whole frame and is labeled, then such appearance of the bed does not represent how the bed would look in full room photos.
- Preferably images include multiple instances of the sought objects and multiple classes appear in the same image: picking such images leads to higher total instances in the dataset and a better representation of each of the classes.
- Images don't include humans: the pictures in forensic setting likely would not include any humans to give a clearer perspective of the room, therefore the subset also doesn't include humans not to confuse the model.

In some cases the second, third and fourth criteria may be disregarded to obtain more varied representations of a class. Such decisions were made subjectively based on the previous selections.

The final obtained dataset consisted of 1008 images, their representations can be seen in Table 1.

The final dataset manages to include each class in at least 100 images. The multiple instance criteria has been successful in providing class representation in more images and more instances for each class with the most often encountered class "Chair" appearing in 341 images and having 839 instances in the dataset, therefore being represented in 241% more images than minimum requirement. Meanwhile the least encountered object refrigerator appears in 125 images and has 130 instances, therefore still being represented in 25% more images than minimum requirement.

Altogether dataset includes 3215 object instances.

The visual interface of FiftyOne allows to generate a JSON file that contains the IDs of the selected images.

Table 1. Class statistics in subset

Class	Represented in no of images	Instances in dataset
Bed	194	222
Chair	341	839
TV	274	327
Dining table	172	211
Couch	211	292
Laptop	155	196
Clock	137	188
Cup	216	435
Sink	312	375
Refrigerator	125	130

These IDs were afterwards obtained from the JSON file and passed to the dataset to obtain a subset of the images. The subset was exported to the local drive for further modifications. [3]

The exported dataset consisted of the dataset's image files and JSON file titled "Labels" which includes data about the classes, images and annotations where each annotation has "`image_id`" (linking it to a specific image), "`category_id`" (linking it to a specific class), and bounding box coordinates. [3]

Before proceeding the "Labels" file was modificed in certain ways. Firstly, in addition to our chosen classes the file also included additional classes and annotations for them that appeared in the selected images. As these are not valid targets for fine-tuning or evaluation for our model which is meant to focus only on our selected classes, these classes and annotations were removed from the "Labels" file, leaving only our categories and annotations for them. [3]

Secondly, the "`category_id`" which identifies each class was modified to be equal to the id that this specific class has within the COCO dataset. In this way, if a model is pre-trained on the COCO dataset, it will already be predicting in accordance with those ids and it will not need to re-learn them. [3]

3. Pre-trained models

3.1 Pre-trained model selection

It was chosen to obtain a pretrained model through PyTorch's Torchvision library. Obtaining a model this way had certain benefits, as all detection models were already pre-trained on the COCO dataset, meaning that the model can evaluate the chosen classes straight out of the box and fine-tuning the model does not require radical changes in the model's head. These models are also directly integrated with the rest of the PyTorch ecosystem making the model's fine-tuning via PyTorch a lot easier to manage. [1]

Table 2. Model evaluation at 0.5 confidence threshold

Model	Precision	Recall	F1 score
FASTERRCNN_RESNET50_FPN_V2	0.66	0.86	0.74
RETINANET_RESNET50_FPN_V2	0.91	0.64	0.75
FCOS_RESNET50_FPN	0.80	0.78	0.79
FASTERRCNN_RESNET50_FPN	0.65	0.83	0.73
RETINANET_RESNET50_FPN	0.90	0.63	0.74

Table 3. Model evaluation at varying confidence thresholds

Model	Confidence threshold	Precision	Recall	F1 score
FASTERRCNN_RESNET50_FPN_V2	0.6	0.73	0.84	0.78
	0.75	0.82	0.78	0.80
	0.9	0.91	0.66	0.76
FCOS_RESNET50_FPN	0.6	0.94	0.62	0.74
	0.75	1.00	0.23	0.38
	0.9	1.00	0.00	0.00

Torchvision offers five different architectures for pre-trained detection models:

- Faster R-CNN;
- FCOS;
- RetinaNet;
- SSD;
- SSDlite.

[1]

The architectures can be offered in different version with either different backbones such as ResNet or MobileNet or different implementations in accordance with certain scientific papers. [1]

Torchvision also offers statistics about the performance and statistics of these models which include their box mean average precision, parameter count and GFLOPS. For this evaluation task five detection models with the highest MAP performance were chosen for further evaluation. Those are:

- FASTERRCNN_RESNET50_FPN_V2;
- RETINANET_RESNET50_FPN_V2;
- FCOS_RESNET50_FPN;
- FASTERRCNN_RESNET50_FPN;
- RETINANET_RESNET50_FPN.

[1]

The chosen models represent Faster R-CNN, RetinaNet and FCOS architectures. All of the models have a 50 layer residual network and feature pyramid network as their backbone. Some of the models are represented in multiple versions (Faster R-CNN and RetinaNet) which means that there are multiple implementations of these architectures according to different scientific papers. Higher versions generally mean a more modern and a better performing implementation. [1]

3.2 Pre-trained model evaluation

The models were evaluated using FiftyOne library's evaluation tool which calculates average precision, recall and f1 score for each class and total. The obtained evaluation results for all classes at 0.5 confidence threshold can be seen in Table 2. [1]

Based on these results two models were selected for further evaluation at higher confidence threshold levels: FASTERRCNN_RESNET50_FPN_V2 since it provided the best recall and FCOS_RESNET50_FPN since it provided the best F1 score.

Further evaluations on the two selected models were done at confidence thresholds 0.6, 0.75 and 0.9 and provided the results seen in Table 3.

Based on these evaluations FASTERRCNN_RESN ET50_FPN_V2 was chosen as the model to be used, since it provided the best recall and F1 score.

A confidence threshold of 0.6 was chosen as it provides a high recall while also having a good precision and resulting in a good F1 score. The impact on confidence threshold on the performance metrics of the model can be seen in Figure 2

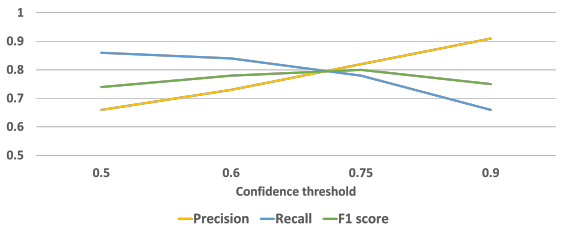


Figure 2. Torchvision's offered pre-trained FASTERRCNN_RESN ET50_FPN_V2 model's performance metrics on the selected dataset depending on the selected confidence threshold.

Using this model and a confidence threshold of 0.6 the model provided performance results that can be seen in Table 4.

Table 4. Precision, recall, and F1 score breakdown per class

Class	Precision	Recall	F1 Score
Bed	0.87	0.96	0.91
Chair	0.72	0.83	0.77
TV	0.83	0.91	0.87
Dining table	0.53	0.74	0.62
Couch	0.82	0.79	0.81
Laptop	0.89	0.89	0.89
Clock	0.68	0.81	0.74
Cup	0.66	0.71	0.68
Sink	0.65	0.88	0.75
Refrigerator	0.77	0.95	0.85

The best precision at 0.89 is for laptop, the worst at 0.53 is for dining table. The best recall at 0.96 is for bed, the worst at 0.71 is for cup. Altogether, the best F1 score at 0.91 is for bed and the worst at 0.62 is for dining table.

4. Fine-tuning

4.1 Fine-tuning setup

The dataset was split into training, validation and test splits with 70% of the images going to the training split, 10% to the validation split and 20% to the test split.

The split was done semi-randomly with FiftyOne library's offered split tool. As the images had different distributions of classes the split possibly couldn't be perfect on class-basis for each class but after the split it was verified that in each class the training set contains 70%±5% of the instances, the validation set contains 10%±5% of the instances and the test set contains 20%±5% of the instances.

The training of the model had quite few tunable hyperparameters, since many of them were already fixed in the architecture of the model. From the tunable parameters several variations were tried experimentally to gauge which had the better performance. In the end, the learning rate was chosen to be 1e-4 and momentum to be 0.9 which are quite standard training values.

Torchvision also allows to have a variable amount of trainable backbone layers for its pre-trained models. Generally, this amount ranges from 0 layers in which case only the head can be trained up to 6 trainable layers. More layers allow the model to increasingly change how it interprets the images and objects which can lead to a better performance with a good dataset but in case of a smaller dataset that may have some gaps it can lead to worsening results. [1]

The number of trainable backbone layers was also tuned experimentally. An amount of two to four trainable backbone layers provided the best results which were very similar in performance. The final number of trainable backbone layers was chosen to be three.

The model was trained in total for 10 epochs. The training was done using a 15GB Nvidia T4 GPU through Google Colab. Each epoch took approximately 5 minutes to train and the total training time was approximately 50 minutes.

Table 5. Fine-tuning hyperparameters

Hyperparameter	Value
Learning rate	1×10^{-4}
Momentum	0.9
Trainable backbone layers	3
Epochs	10

4.2 Fine-tuning results

An example of how model's predictions changes due to the fine-tuning can be seen in Figure 3. This example also illustrates that fine-tuning can be quite useful for removing unnecessary classes, if we wish to focus only on a select few: before fine-tuning the model was making predictions for such classes as "person" and "toilet" in addition to our classes of interest but after fine-tuning there were no more predictions for those classes that were above the confidence threshold of 0.6. Also in this case the prediction of "sink" objects is more precise: previously the bathtub was erroneously classified as a sink but after fine-tuning that is no longer the case.

The evaluated F1 scores breakdown by class on the test dataset before training and after training can be seen in Table 6.

Table 6. Comparison of F1 Scores Before and After Training

Class	Pre-training F1 score	Post-training F1 score
Bed	0.91	0.87
Chair	0.79	0.75
TV	0.89	0.88
Dining table	0.60	0.67
Couch	0.80	0.75
Laptop	0.87	0.90
Clock	0.73	0.74
Cup	0.63	0.52
Sink	0.76	0.71
Refrigerator	0.80	0.77
Weighted average	**0.78**	**0.74**

Overall, the finetuning has slightly decreased the weighted average F1 score of the dataset. Inspecting the results on individual classes it can be seen that F1 score has decreased for most of the classes but not for all of them. While the F1 score decreased for seven

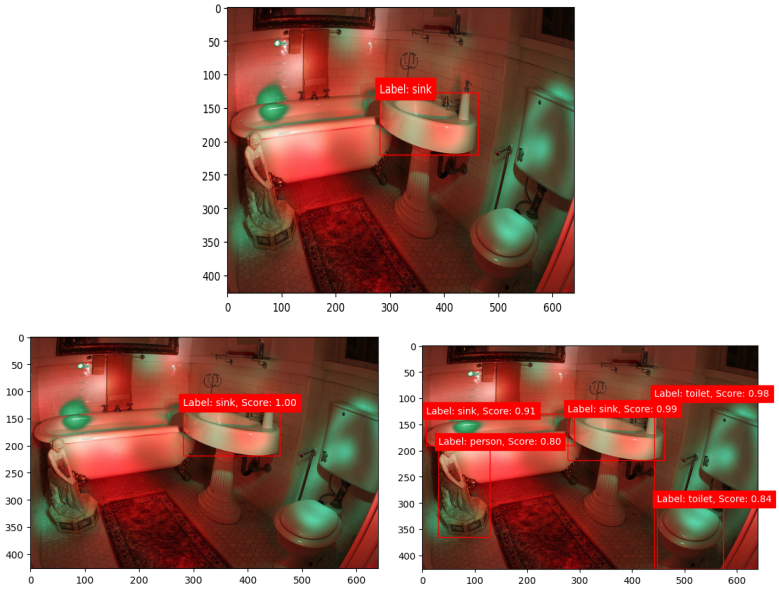


Figure 3. An example of image targets and model's predictions. Clockwise from the top: target, model's prediction before fine-tuning, model's prediction after fine-tuning.

classes, the remaining three classes experienced F1 score improvements: dining table class experienced the most notable improved of F1 score from 0.6 to 0.67, meanwhile laptop and clock classes experienced mild improvements from 0.87 to 0.9 and from 0.73 to 0.74 respectively.

It is quite probable that in the cases where F1 score decreased, the training set was not able to comprehensively provide all the variations and representations of those classes, causing a performance decrease on the test set when such missing representations were found in it. Meanwhile in the cases where F1 score increased, it seems that the representation of those classes in the training set was good enough for the instances encountered in the test set and the narrowed focus on the room context (for example outdoors dining tables, clock towers etc. being emmitted) allowed to achieve a better performance.

In that case, a larger dataset with more varied representations of the classes could lead to an F1 score increase in the other classes as well and therefore to an overall improvement of the model's performance.

5. Conclusions

In summary, this paper explored the possibilities of using existing solutions – annotated datasets and pre-trained neural network models – to obtain and evaluate a neural model which would be capable of solving the given task: identifying objects in a room setting. An attempt was made afterwards to further improve the performance of the model by fine-tuning it on the obtained dataset of annotated objects in room context.

It was observed, there are existing datasets and pre-trained models which fit and can solve the given problem: Microsoft's Common Objects in Context dataset contains many classes of objects very commonly found in room settings and annotated images which depict those objects in a room setting, meanwhile Torchvision's detection models are pre-trained to identify those same classes and yield quite good results out-of-the-box.

A subset of images in room context containing 10 different object classes was obtained and was used as basis for pre-trained model evaluation and fine-tuning. After comparing the different models offered by Torchvision the best model was based on a specific implementation of Faster R-CNN architecture with a 50 layer residual network and FPN backbone.

An attempt was made to further fine-tune the model with the created subset. This resulted in an overall slight decrease in the performance of the model in terms of the obtained F1 scores, however certain classes showed an improvement in performance after fine-tuning, raising a possibility that further expanding and curating the dataset could lead to an improvement for other classes as well and thus also for the model as a whole.

The work can be further continued in several ways. One way is to keep the focus on the same classes and attempt to increase the model's performance by improving the dataset to better cover the possible representations of the classes and performing fine-tuning again with the improved dataset. Another way is to expand to other classes within the COCO dataset and re-evaluate model's overall performance and its performance on the newly included classes. A third way

is to expand to new classes that are not included in the COCO dataset. This would require to change the model's head to a new one that can accommodate additional classes and to prepare a new dataset which depicts and identifies these classes in the room setting, so that the model can be trained to recognize them.

References

[1] PyTorch Developers. *PyTorch: Models and pre-trained weights*. Available at: https://pytorch.org/vision/stable/models.html. Accessed: 2023-02-01.

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, ... C. L. Zitnick. *Common Objects in Context (COCO)*. Available at: https://cocodataset.org/#home. Accessed: 2023-02-01.

[3] Voxel51. *FiftyOne Documentation*. Available at: https://docs.voxel51.com/. Accessed: 2023-02-01.

[4] D. Acharya, K. Khoshelham. *Smart parking in fast-growing cities: Challenges and Solutions*. Vienna: TU Wien Academic Press, 2021.