Data Generation Methods for Multi-Label Images of LEGO Bricks

[1]**Berend Kam** , **Responsible Professor:** [1]**Dr. J.C. van Gemert** , **Supervisor:** [1]**Attila Lengyel**
[1]TU Delft
bkam@student.tudelft.nl, {J.C.vanGemert, A.Lengyel}@tudelft.nl

Abstract

Data collection and annotation have proven to be a bottleneck for computer vision applications. When faced with the task of data creation, alternative methods to traditional data collection should be considered, as time and cost may be reduced significantly. We introduce three novel datasets for multi-label classification purposes on LEGO bricks: a traditionally collected dataset, a rendered dataset, and a dataset with pasted cutouts of LEGO bricks. We investigate the accuracy of a ResNet classifier tested on real data, but trained on the different datasets. This research seeks to provide both insight into future dataset creation of LEGO bricks, as well as act as an advisor for general multi-label dataset creation. Our findings indicate that the traditionally collected dataset is prone to overfitting due to speedups used during collection, with 90% accuracy during training but 19% during testing. Furthermore, synthetic data techniques are applicable to multi-label LEGO classification but need improvement, with accuracy ranging from 34% to 45%.

(a) Real data (b) Rendered data (c) Cut&paste data

Figure 1: We introduce three novel multi-label datasets of LEGO bricks. (a) was collected 'traditionally' by manually shooting pictures of LEGO bricks against various backgrounds. (b) was generated through rendering software. (c) used cut out images of LEGO bricks and posted them against random backgrounds. While data in (a) is realistic, it is also most costly to produce.

1 Introduction

Data collection and annotation are a bottleneck for computer vision applications. With computer vision applications becoming more commonplace and also more complex, there is also an increased need for annotated datasets however. Hence, data collection is an active research topic [1, 2]. Researchers are looking for ways to produce datasets that are more diverse [3], faster [4], and cheaper [5]. According to a 2018 survey [6], in case there is not enough quality data readily available, the only means to collect data is through generation. The two main methods of data generation presented in this survey are data generation through traditional methods, and synthetic data generation.

Traditional data is collected and annotated manually, and although this produces realistic data, creating a dataset by hand is a task of great proportion [6]. Crowdsourcing is often used to divide manual labour more effectively, but this produces its own challenges. Uncertainty can arise from workers incorrectly following instruction, costs are often high as workers need to be compensated, and latency is significant as human workers tend to work much slower than machines [7]. To limit costs, data collection can also be partially automated. One study used a car to collect data of traffic situations by driving with camera equipment on board [1]. However, this data still had to be annotated manually.

Synthetic data generation, wherein artificial images are created, is considerably faster and cheaper than traditional data collection, but can have trouble bridging the gap to reality. This gap is referred to as the domain shift. There are techniques in place that can reduce the effects of this domain shift. Recent works have had success by training with rendered images through domain randomization, wherein textures, lighting, and backgrounds were randomized [4]. Other advances have been made by generating semi-synthetic data, wherein images of real objects were cut and paste onto random backgrounds to increase training diversity [5]. Both of these methods allowed for cheap and fast collection and annotation, and produced results deemed competitive. Alternatively, real data can be sampled for synthetic data generation [8]. Generative Adversarial Nets were able to generate diverse synthetic im-

ages by learning features of small real-life datasets [9].

To our knowledge, a dataset of multi-label images of LEGO bricks is missing; although some have trained using synthetic images of single bricks [10], well labelled pictures of multiple real LEGO bricks do not exist. A dataset of such format is the first step in obtaining networks that can find a specific LEGO brick in a large pile of bricks, or immediately tell the composition of a pile of bricks. Data collection and annotation can be a time consuming task however. For novel applications, such as in the case of creating a dataset of images of LEGO bricks, it is often unclear if synthetic data generation techniques translate well. A preliminary investigation into the effectiveness of synthetic datasets can thus have great payoff when those datasets are deemed competitive with traditional data. Particularly, two different approaches to synthetic data generation are investigated, and their effectiveness is compared to a traditional dataset. Since our approach aims to replicate or surpass traditional dataset generation, an image classification network is trained on different types of datasets, but finally tested on a traditional dataset. Specifically, our main contributions can be summarized as follows:

- We present three novel datasets of multi-label images LEGO bricks, to be used for computer vision purposes, of which example images are seen in Figure 1. These datasets consist of:
  1. a 'real' dataset consisting of manually collected pictures of LEGO bricks in various scenes
  2. a synthetic dataset of rendered images using structured domain randomization
  3. a synthetic dataset of images generated through cut-and-paste strategies
- We evaluate the domain shift between the three datasets by using a classifying network as a baseline comparator. Performance is measured in F1 scores, a metric whose output ranges from 0 to 1 and which combines precision and recall.
- We explore possible speedups for generating the real dataset.

We find that for multi-label classification on images with 1-13 bricks and 85 different classes, F1 scores of up to 0.955 may be achieved when a network trained on synthetic data is tested on synthetic data. This F1 score drops significantly when these networks are then used to classify real images: 0.406 for rendered data and 0.447 for cut-and-paste data in the best case. The traditionally collected 'real data' appears to be too unbalanced; while each image is different, similarities in labels and background cause the classifier to achieve high scores during training, but low scores on a selected test set, where an F1 accuracy of 0.192 is achieved. In the following sections we showcase our findings in detail. Section 2 shows related works and provides background information to common methods to create multi-label datasets. Section 3 details the methods used to create data. Section 4 showcases the experimental setup, section 5 discusses the results, followed by section 6 in which we discuss possible fallacies and ethical implications of our research. In section 7, a conclusion is drawn.

2 Related Works

2.1 Existing LEGO datasets

There are other works that have looked at creating a dataset of pictures of LEGO bricks. In particular, software engineer Daniel West created a LEGO sorter along with a dataset of 100,000 images of individual LEGO bricks [11]. During his project, West trained an image classifier using rendered images of LEGO bricks, randomizing textures, lighting, and noise. Afterwards the sorter was able to correctly identify single bricks as they were fed to it, including bricks the classifier had never seen. The sorter photographed bricks as they passed through the machine, creating a dataset of labelled images of single bricks. This dataset does not contain varied backgrounds or multiple LEGO bricks per image however, making it unable to serve as a training dataset for multi-label classification networks or object detection. So while it is suitable for sorting, for purposes such as finding a certain type of brick in a large pile of bricks or seeing if a brick is present in a pile of bricks, this dataset falls short.

A 2018 study [12] also created a LEGO sorter, only this time available data from an online LEGO retail store was used, in which users upload pictures of single bricks they aim to sell. While this data offers large variation in backgrounds, lighting, and picture quality, there are significant caveats. For one, for most bricks few images are available, giving low class representations for most bricks. Second, images of multiple bricks are uncommon as most pictures contain 1 or 2 bricks. As a result, this sorter performed worse on images containing multiple bricks, and hence this data is also unsuitable for multi-label classification. Other online image databases containing LEGO bricks such as Shutterstock [13] more often contain completed sets or LEGO art, and cannot be effectively filtered to contain pictures of only piles: a dataset created from such a site would have to be handpicked, as well as annotated manually. Additionally, bricks contained within these images may be of many distinct classes, leading to a large classification domain and again low class representation. To our knowledge, no labeled dataset exists of multiple LEGO bricks.

2.2 Traditional data generation

The most common way of traditional data generation is through the use of existing unlabelled databases, such as picture sharing sites. In such a process, ensuring quality, diversity, and accuracy in labelling of images is a costly and time consuming endeavor [6]. If databases fail to meet these requirements, they may be diversified further through data augmentation, whose common techniques include applying filters to, cropping, mirroring, and rotating existing images [14, 15]. If this still produces insufficient data, data must be created, which is a costly task. Although data collection may be automated [1], annotations are mostly created manually. Our research presents a labelled sample dataset generated from scratch, without the use of crowdsourcing during collection.

Data must meet specific requirements to accurately train computer vision applications. First, to allow for preservation of physical properties that may be used in certain com-

puter vision applications [16], and to preserve images in high quality as low quality images may reduce performance [17], past research has created datasets using DSLR cameras [18]. Other Large databases, such as the VOC2007 database, have variation in object size, orientation, pose, illumination, position and occlusion [19]. In the case that data is not available through existing databases, it is necessary to simulate these factors. Failure to do so can result in under representative datasets, such as in the case a research where video data of traffic scenes in Germany did not translate well to traffic scenes from other countries [1]. In summary, variation in backgrounds, camera angle, camera zoom (size), illumination, occlusion, orientation of bricks and position are factors to look out for when collecting LEGO images. All images are shot using a DSLR camera.

2.3 Cut-and-paste dataset generation

Cut-and-paste strategies are a state-of-the-art method for synthesizing datasets [5] , greatly reducing production costs as images are not collected or annotated manually. Compared to rendered images, cut-and-paste images instinctively suffer less from physical limitations as they are more 'real', but may suffer from lack of 3D orientation coverage as well as accurate lighting and shadows. A quick summary of the cut-and-paste method follows below:

1. Images of objects to be pasted are collected.
2. Images of scenes are collected.
3. Objects are isolated from their backgrounds.
4. Objects are pasted into the background, making use of both blending and data augmentation

Region-based features play a significant role in computer vision [20], and this is a double edged sword. For one, it allows us to ignore physical limitations of brick placement in real life, for example, a brick may be pasted on the side of a cupboard without significantly increasing error rates. Second, blending must be used to remove boundary artifacts that arise from pasting images, as seen in Figure 2. The two modes of blending proposed are Gaussian blur and Poisson blending, with best results achieved by applying to each configured image both types of blending and no blending separately. This means for each configuration of pasted objects, 3 images are created. Data augmentation is used to further make images robust. Random 2D rotations of pasted objects, as well as using multiple pictures of the same object from different angles throughout the dataset, ensuring angle coverage. Further augmentations are partially overlapping objects in a process called occlusion, and objects placed partially out of bounds, called truncation. Finally, non-classified 'distractor' objects are also pasted into the scene. While our research aims to replicate this method of data generation, no distractor objects are added as these are not present in the traditionally collected data. Additionally, given the size of the dataset, brick configurations across images are not repeated with aim to increase data diversity. Lastly, to further reduce boundary artifacts an alpha channel is added instead of an object mask.

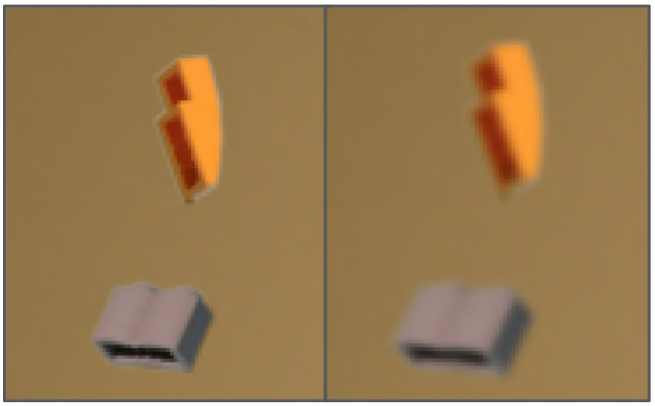


Figure 2: Two similar cut-and-paste images with varying levels of Gaussian blur. Blur removes the boundary artifact present in the first image, but may reduce artifact overall appearance

2.4 Rendered data generation

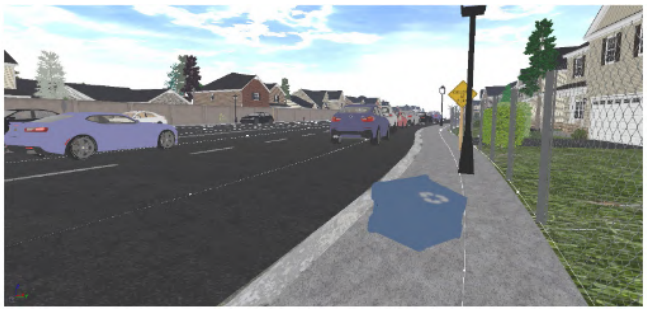
Rendered images pose a viable solution to both the high cost and latency of traditional data creation [2, 4], yet other problems arise for such databases. Although variation in controllable factors can be synthesized, the gap to reality is large. Besides the obvious difference in realism of a rendered image as compared to a real image, unforeseen factors and distractors in real data often derail networks trained on synthesized data. Domain randomization [4] must be applied to overcome these factors. In short, domain randomization includes randomly placing distractor objects in rendered scenes, as well as strongly randomizing lighting, camera angles, camera zoom, and textures. Structured domain randomization limits the randomness of these factors, by placing strong emphasis on realistic random factors. For example, instead of randomly placing distractor objects in scenes, high probability is given to contextually sound locations. See Figure 3. Our research implements structured domain randomization instead of domain randomization as LEGO bricks may be distinct only on texture, hence full randomization may make some classes indistinguishable. As no distractor objects are present in the traditionally collected data, the rendered data will also not contain any distractor objects.

2.5 Classification network

Image classification is a common application for image datasets, and both single label classification and multi-label classification [21] are common research topics. Since LEGO bricks are often found together, the data collected is geared towards multi-label classification. A popular class of image clasifier is the residual network classifier, or ResNet in short. It is similar to a Convolutional Neural Network with up to one thousand layers and may be used a standard comparator for new networks [22]. Alternatives to ResNet may include classification networks such as ResNext [23] and WideResNet [24]. We implement a ResNet-50 network and apply thresholding to turn it into a multi-label classifier [25].

On a more general note, all types of classification networks benefit greatly from balanced classes [26], as well as large amounts of class representation. ImageNet, a commonly used

(a) Domain Randomization. Img source: [4]



(b) Structured Domain Randomization. Img source: [2]

Figure 3: Two differently rendered images. Note the difference in realism of car placement, textures, and distractor objects

database for computer vision applications, has around 650 images per class, as per June, 2020 [27]. While the amount of data is often targeted as most important metric, as a lack of data often leads to reduced accuracy and increased error rates, aspects such as class type and amount of features of objects should not be overlooked. For similar objects that rely on many features for distinction classification accuracy may decrease drastically [28]. LEGO bricks may be prone to errors of this type, as bricks may vary only on one relatively small detail, such as color, amount of eyes, or small changes in shape. Lastly, the asymmetric shape of some bricks may lead to a large amount of possible 2D representations, which could also lead to reduced classification rates.

3 Methodology

To compare several data generation techniques, we generate 3 different datasets. These are: a traditional 'real' dataset that acts as a baseline comparator, a cut-and-paste dataset, and a rendered dataset. Subsection 3.1, 3.2 and 3.3 describe the generation methods for these datasets.

3.1 Realdata

A dataset is created by shooting around 3000 images of LEGO bricks, with the main focus on class balance. There are 85 different classes of bricks[1], each picture contains a random selection of 1-13 LEGO bricks. Combinations of more bricks are not generated due to annotation costs [29]. To speed up the collection process, several speedups are introduced. For one, all labels for pictures are produced randomly before collection, so that no time is wasted recording labels manually. Second, each combination of LEGO bricks is photographed three times, rearranging the bricks in a random manner in between pictures, and varying camera angle and zoom. Both occlusion and truncation occur naturally during this random rearrangement and are not prevented. Third, a random brick is taken away or added to the combination to create a new combination, and this is repeated every 3 pictures for up to 10 times. This depends on if a boundary for the amount of bricks per image is reached beforehand, so either 1 or 13 bricks. Lastly, to maintain class balance with these speedups, the lowest represented classes may be given priority when creating random combinations. Figure 4 showcases the class balance of the real dataset. It is noted that such speedups may cause imbalances as classes of bricks may appear together relatively frequently, as well as combinations of bricks and backgrounds. This is not a certainty however, as this relation decreases relative with size.

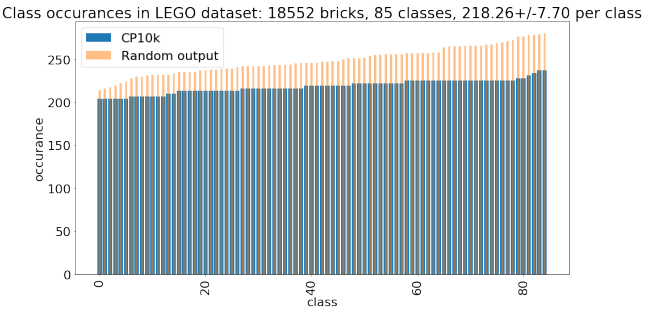


Figure 4: Class balance of the traditionally collected data and of a uniform randomly drawn dataset, which acts as comparator. While the rendered and cut-and-paste data are also uniform randomly drawn, the real data is actively balanced due to speedups in data collection.

Bricks are photographed from different camera angles, as well as shuffled around in between pictures to ensure that bricks can be recognized from a variety of viewpoints. Furthermore, backgrounds and lighting are varied every around 70 pictures to prevent the model from over-fitting on one type of background or lighting. Figure 5 shows the difference between various pictures. The images are saved in a HQ RAW format, which are unfiltered images that preserve physical qualities which are important in physics-based computer vision systems [16].

3.2 Cut-and-paste data

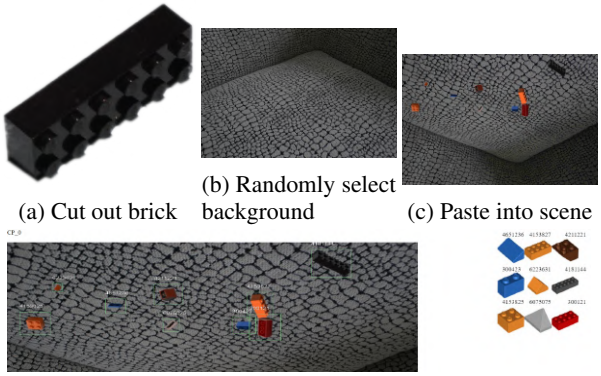
Similar to the baseline dataset, the cut-and-paste dataset contains a random 1-13 bricks per image. A full overview of the cut-and-paste data creation process can be found in Figure 6. Original images for each class of brick were photographed from 4 different angles, ensuring that all sides are photographed at least once. They are shot against a distinct color background to allow for facilitated edge detection. To create photoshopped cutouts for each brick, backgrounds

[1] a selection is made out of the LEGO Classic set 110002

(a) Parquet floor, natural lighting (b) Wooden desk, artificial light

Figure 5: Two examples of variations between traditionally collected pictures. Lighting, background, camera angle, camera zoom, and brick orientation may be varied between pictures.

were manually removed using photoshop, and an alpha channel was added to help smooth out any pasting discrepancies. Images were resized to tightly fit the brick. This image size also acts as a bounding box for annotation.



(a) Cut out brick (b) Randomly select background (c) Paste into scene

(d) Label brick

Figure 6: 4 step process for creating a cut-and-paste image

Each background from the baseline dataset is photographed without bricks. This gives 124 different pictures that can be used as background. To further diversify the background, they are randomly mirrored or flipped. Pasted bricks have a 50% chance of being mirrored, and have a random rotation of 0-359 degrees. To prevent boundary artifacts created by pasting [20], each image is blended in by using random varying radii of Gaussian blur of 0-1. No further blur was applied some research suggests that a blur greater than 2 might significantly reduce accuracy [17].

3.3 Rendered data

We synthesize 5000 images of 1-13 LEGO bricks per picture, classes remain the same. 3D models of bricks are downloaded from the Rebrickable [30]. All bricks are fitted with a tight bounding box relative to the camera angle, as seen in Figure 7.

We apply structured domain randomization [2] on the following factors: to increase robustness to varying backgrounds, the floor of the rendered model is fitted with one of the 124 backgrounds from the baseline dataset. To ensure varying camera angles and to increase robustness to relative brick size, both angle and zoom are randomized for each image. A single light source is also randomly placed within the scene at varying brightness levels. Lastly, to accurately portray varying brick orientation, a physics simulation 'drops' bricks onto the surface, after which the render is captured and data augmentation is applied; to ensure that downsampling images to training size does not compromise recognizability of bricks in pictures that are zoomed out, pictures are cropped to annotation [25].

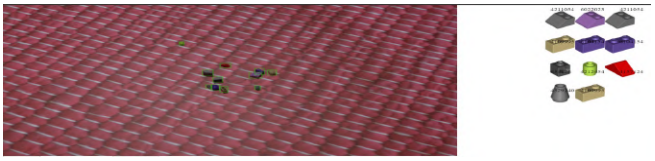


Figure 7: An annotated rendered image produced through structured domain randomization. Camera angle and zoom, backgrounds, lighting and brick combinations are varied between pictures. A physics simulation 'drops' bricks to produce random brick orientation.

4 Experimental Setup

An F1 score calculation is used to compare classification rates across datasets [31]. As F1 scores are a combination of the precision and recall, they give insight into false positive and false negative rates simultaneously. We implement two methods of calculating F1 scores: micro-averaged and macro-averaged. Micro-averaging takes the average over all labels, whereas macro-averaging calculates scores per instance, and then takes the average over all instances. All formulas are given in (1-4).

$$\text{micro-average precision} = \frac{\sum_{c \in C} TP_c}{\sum_{c \in C} TP_c + \sum_{c \in C} FP_c} \quad (1)$$

$$\text{micra-average precision} = \sum_{c \in C} \frac{TP_c}{TP_c + FP_c} \quad (2)$$

$$\text{micro-average recall} = \frac{\sum_{c \in C} TP_c}{\sum_{c \in C} TP_c + \sum_{c \in C} FN_c} \quad (3)$$

$$\text{macro-average recall} = \sum_{c \in C} \frac{TP_c}{TP_c + FN_c} \quad (4)$$

To calculate the F1 score, the harmonic mean is taken of the precision and recall. See (5). This will give scores of between 0 up to 1. Both the micro and macro scores may be used in the results depending on the applicability[2], and tables and figures will indicate which metric is being used.

$$F1 = 2 * \frac{precision * recall}{precision + recall} \quad (5)$$

In overview, each dataset is sampled for 3000 images, and then split up into a training, validation, and test set. Then, a

[2]Micro-average scores are insightful to overall performance while macro-average scores are useful for measuring performance per instance

classification network is trained seperately for each dataset, and scores are determined on their respective test set. For the rendered and cut-and-paste datasets, scores are also determined on the test set of the real dataset. Next, 5000 images of the synthetic datasets are taken and split again into training, validation, and test sets. Again networks are trained separately and scores are calculated on their respective test sets and on the test set of the real dataset. Lastly, 10,000 images cut-and-paste images will be used to train a final network. As a baseline comparator, a score for a random label predictor is also given.

To further enhance understanding of dataset fallacies, F1 scores are also calculated per brick class, color, or shape. For example, while an average F1 score on all images may give an overview of the overall performance, average F1 scores for each brick shape may reveal that 2 by 4 bricks are most often misclassified. Additionaly, we present F1 scores that are 'relaxed' on shape or color. A classifier that is "relaxed on shape" may output a brick of the right color but of the wrong shape, but it will still be counted as a true positive. Similarly, "relaxed on color" means a wrong color but a right shape will result in a true positive.

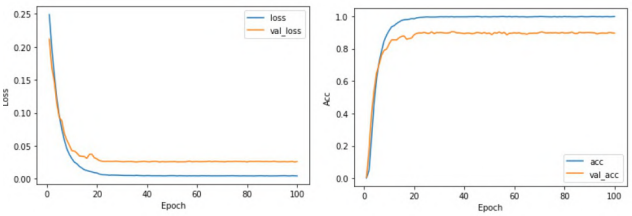
We use PyTorch's implementation of ResNet-50 as it is most popular and using a BCEWithLogitsLoss [32] and a threshold, we convert it to multi-label classification. We shuffle each dataset and subdivide them into a test (10%), validation (10%) and training (80%) set. For the real dataset, a second test (Real3000s) is run wherein both the validation and test set were selected to contain unseen combinations or backgrounds. Each dataset is trained for 100 epochs using ResNet-50, with a learning rate of 0.1, batch size 16. Images are also resized to 384*256 to speed up training. A threshold of 0.5 is used for classification. Full details can be found at [25]. The networks are tested on their respective training types and additionally tested on 3000 real test images. Note that the rendered data contains only 83 possible classes due to 2 3D models not being available, hence the real test set for these networks do not contain any instances of these classes. The F1 accuracy is calculated according to formulas (1-5). The results can be seen in Table 1.

5 Results

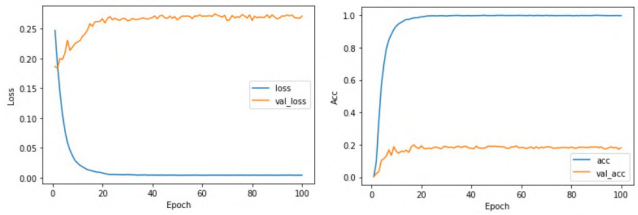
Dataset type	Test	Real Test	Rel. on shape	Rel. on color
Real3000	896	N/A	901	898
Real3000s	192	N/A	297	275
Rendered3000	804	340	468	453
Rendered5000	905	406	512	484
CP3000	851	374	486	480
CP5000	926	447	550	558
CP10k	955	440	555	551

Table 1: F1 scores for all datasets, scaled by 10^3

The results show that for the synthetic data trained networks are able to sufficiently learn their respective dataset. However, the Real3000s dataset which contains selected validation and test sets does not seem to approach the same F1 accuracy as the other networks. Further inspection of the loss and accuracy during training of both real datasets can be seen in Figure 8. When fully trained on synthetic data, Resnet-50 classification networks have an F1 accuracy of around 0.4 for both renderend and cut-and-paste data. Further details of the results of running the real test set on each classifier can be found in appendix A, Figures 13-16. The most notable results are summarized here: for one, both the cut-and-paste and rendered trained networks were able to recognise certain classes, colors, and shapes of the real data at near perfect rates, while others were not recognised at all. Second, all classifiers seemed to be more color sensitive than shape sensitive, as average 'relaxed on shape' scores were always higher than 'relaxed on color' scores. Lastly, Figure 16 indicates that for these datasets, F1 accuracy does not decrease when more bricks are present in an image.



(a) Real3000



(b) Real3000s

Figure 8: The training loss and macro-average F1 accuracy of the real dataset compared with random test and validation set and selected test and validation set, which contain unseen background and combinations.

5.1 How do speedups in traditional data collection affect performance?

Table 1 indicates that when trained on Real3000 F1 accuracy is around 0.9, but this accuracy is skewed: a handpicked testset that contains pictures with completely new combinations and backgrounds performs worse. Moreover, the loss and accuracy rates during training in Figure 8 indicates that the classifier is not able to learn features of a handpicked validation set, but can learn features of a randomized validation set. A possible explanation for this phenomenon is as follows. The speedups that were used to create the dataset causes some bricks to appear together frequently. A combination of bricks appears together three times, and while the camera angle changes and the bricks are shuffled around, the background remains the same. This may cause the classifier to associate backgrounds with certain combinations of bricks, and thus overfit. Furthermore, since a combination of bricks is turned into another combination by simply removing or

adding a brick, and this may be done up to 10 times, again certain bricks appear together more frequently. This further causes the model to overfit on brick combinations. These two types of imbalances in the dataset could explain why the model performs relatively well on a random validation and test set; the classification problem is reduced to recognizing the combination and background used, after which a prediction of this combination will yield an F1 score comparable to training data. See Figure 9.

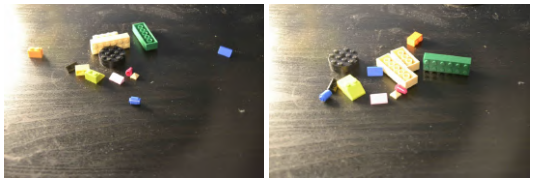


Figure 9: While both images contain different arrangements of bricks from different angles, the combination remains the same. This could explain why a network is able to recognize a randomized test set, but not one containing unseen combinations.

When tested on combinations and backgrounds it has never seen before, the model cannot identify most bricks. Figure 10 shows that in this test set, few classes receive F1 scores between 0.6 and 0.8, but many rates are much lower than that. This could be an indication that the model only needs to identify a few bricks in a picture to predict a combination.

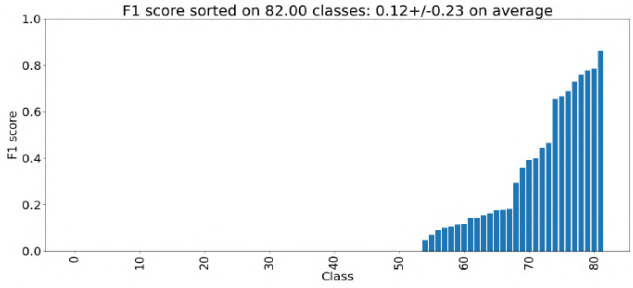
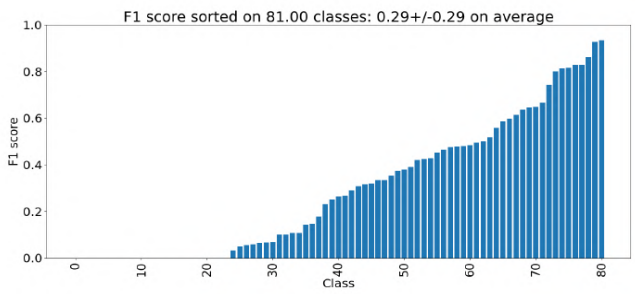


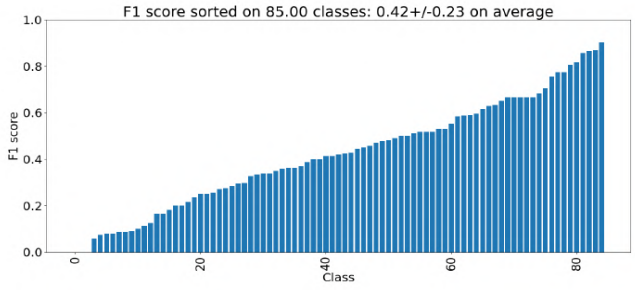
Figure 10: Micro average F1 score per class: Real3000s. While some classes are recognized at training rates, most classes are not recognized at all.

5.2 How do the different methods of data generation compare in domain?

The networks trained on rendered data achieve F1 rates of between 0.8 and 0.9 on test set, comparable to training rates. When looking at Figure 13 it appears classes in the real dataset are recognized at varying F1 rates, from 0 to 0.9 or even 0.95. This indicates that the domain is greater for some bricks than others. Compared to cut-and-paste data, a larger portion of classes do not get recognised at all and the overall macro-average F1 score is lower, namely 0.29 compared to 0.42 for 5000 images, as shown in Figure 11. This trend upholds in for colors and shapes in Figure 14 and Figure 15, though in the latter the difference is smaller.



(a) Rendered5000



(b) CP5000

Figure 11: Micro average F1 score per class. Note how rendered data is misclassifies a larger portion of classes than cut-and-paste data, possibly indicating a larger domain shift.

Cut-and-paste data exhibits similar behaviour to rendered data, in that by Figure 13, there are varying F1 rates for different classes, indicating that the domain shift affects some bricks more than others. Furthermore, Figure 14 indicates that certain colors such as grey and beige are recognized at low rates across all cut-and-paste datasets, whereas dark pink and dark purple are consistently in the top 3 most recognized classes, possibly indicating that standout colors are recognized easier. This trend is also present in shapes as per Figure 15. Lastly, it appears that for this problem, 10k images do not translate to better recognition of real data, as no improvements were made in any regard.

6 Discussion

Results indicate that although synthetic data can be used to train real data classifiers, more adjustments to the data will have to be made before rates approach rates of tests on synthetic data. Some classes are more sensitive to this domain shift than others, and further investigation into why this is might prove useful in further understanding of classification networks. Lastly, with all datasets, it appears that the number of bricks per image does not affect F1 scores. It remains to be investigated whether this trend upholds with significantly more bricks per image as increased truncation and occlusion might increase error rates.

6.1 Fallacies

We suspect the real data collection method suffers primarily from a lack of diverse data. Although performance is known

to suffer under multi-label classification [21], results from Table 1 indicate that at least for synthetically generated data, F1 scores of up to 0.955 can be attained. If this translates to real data, we suspect that this decrease in performance is caused by lack of diversity in the data. Speedups used to collect this data reduce the variation in background and appearance of different combinations of bricks. To combat this, more images would need to be created so that networks better learn to categorize individual bricks in different surroundings, or no speedups should be implemented to make data more diverse. A lack of data is a common problem for most computer vision applications as data creation and annotation are massive and costly undertakings [6]. For practical applications, such as checking if a LEGO brick is present in a pile of bricks, we suspect traditional dataset collection methods may not be sufficient. Although collecting data for such large piles would not be much harder than for 1-13 bricks, due to the amount of occlusions, truncation, and sheer number of different LEGO bricks, annotation would be prone to error, not to mention costly and time consuming. For testing for such purposes we recommend using a synthetic dataset.

Synthetic datasets are known to suffer from bridging the gap to reality, and techniques such as domain randomization [4] and structured domain randomization [2] are aimed especially at solving this. Although some domain randomization is applied to the rendered dataset, more variation in lighting intensity and color might produce better results, as not all colors were recognized at equal rates per Figure 13. The results show that shapes are also recognized at varying rates, possibly indicating that some 3D models were not accurate representations of their real life counterparts an example of which is shown in Figure 12. It should be noted that the nature of the LEGO classifying problem might hinder domain randomization, as domain randomization depends on randomizing textures and thus brick colors, an essential feature for distinguishing bricks. The cut-and-paste dataset suffers least

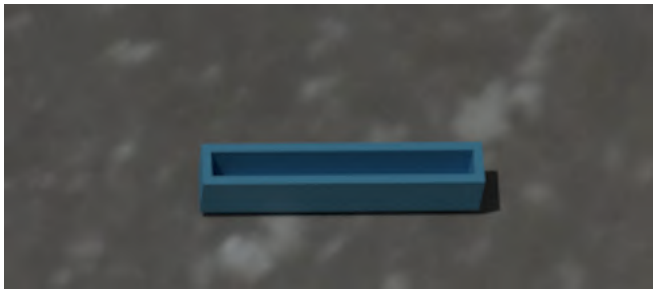


Figure 12: A faulty 3D brick model. Connective parts on the bottom are missing and the texture is slightly off, possibly hindering a domain shift.

from a lack of data and instinctively less of the domain shift than rendered data, however, other fallacies arise. A lack of 3D-orientation coverage of bricks may cause the network to overfit on certain viewpoints, and pasted images may not blend seamlessly into scenes. To better these results, a more diverse 3D coverage should be made and Poisson blending must be applied. Furthermore, edge detection using neural networks could further improve both 3D coverage as well as blending, for it increases the efficiency by which new images can be photoshopped into scenes. Lastly, a drawback as seen in Figure 6d is that rotation causes bounding boxes to not tightly fit. This could be removed by switching from an alpha channel to a layer mask, though this would increase boundary artifacts. Alternatively, oriented bounding box annotation could be used [33].

6.2 Ethics and Reproducibility

Besides the annotation of the real data which was done through crowdsourcing [29], there are no large ethical concerns as no human interaction was needed for the completion of the data collection.

In light of a ongoing reproducibility crisis in the scientific community [34], we will shortly discuss the reproducibility of this research. All synthetic data creation protocols, as well as the ResNet-50 classifier are available for use on Gitlab [35–37]. All created datasets are available on TU Delft's WebDrive. In case the real dataset is to be replicated, it should be noted that any pictures taken may produce somewhat different results than the current datasets: while lighting, background, camera angle, camera zoom, and brick orientation were randomized, they were not controlled. These factors may differ slightly between datasets. The cut-and-paste data suffers from these same random factors: different backgrounds and different brick orientations in cut-outs may produce different dataset characteristics. The rendered data may differ in textures used for backgrounds and different 3D models. Lastly, all classification scores present in section 6 and appendix A are taken from a single test run and may differ from run to run.

7 Conclusion and Future Work

We presented 3 novel datasets for computer vision purposes on multi-label images of LEGO bricks. After, we investigated the domain shift between these datasets by comparing F1 scores of a ResNet-50 classifier. Lastly, we investigated the use of speedups in traditional data collection. We find that networks trained on synthetic data are able to achieve F1 scores of 0.804 to 0.955, but these scores drop to 0.340 to 0.447 depending on the type and amount of synthetic data used. The use of speedups during traditional data collection appears to make real data prone to overfitting.

For future works, we do not consider improvements to networks as this is not the focus of the research, rather we recommend to further remove fallacies in data creation protocols. To further improve rendered data generation, we recommend using improved domain randomization. Using more detailed 3D models and textures, as well as varying lighting conditions more and randomly placing distractor objects could improve the domain shift. For cut-and-paste data, we suspect improving bounding boxes as well as 3D orientation coverage would increase scores the most. Finally, to prevent overfitting in the real dataset, additional data should be made with more emphasis variation of backgrounds and brick combinations in pictures.

References

[1] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila, "The eurocity persons dataset: A novel benchmark for object detection," *CoRR*, vol. abs/1805.07193, 2018.

[2] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield, "Structured domain randomization: Bridging the reality gap by context-aware synthetic data," *CoRR*, vol. abs/1810.10093, 2018.

[3] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[4] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," *CoRR*, vol. abs/1804.06516, 2018.

[5] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," *CoRR*, vol. abs/1708.01642, 2017.

[6] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: a big data - AI integration perspective," *CoRR*, vol. abs/1811.03402, 2018.

[7] H. Garcia-Molina, M. Joglekar, A. Marcus, A. Parameswaran, and V. Verroios, "Challenges in data crowdsourcing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 4, pp. 901–911, 2016.

[8] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 399–410, 2016.

[9] S. Gurumurthy, R. K. Sarvadevabhatla, and V. B. Radhakrishnan, "Deligan : Generative adversarial networks for diverse and limited data," *CoRR*, vol. abs/1706.02071, 2017.

[10] K. Quach, "You looking for an ai project? you love lego? look no further than this reg reader's machine-learning lego sorter," Dec 2019.

[11] D. West, "How i created over 100,000 labeled lego training images," Mar 2019.

[12] C. ZEH and S. BABOVIC, "Lego sorting tool," 1957.

[13] "shutterstock." https://www.shutterstock.com/.

[14] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *CoRR*, vol. abs/1712.04621, 2017.

[15] Y. Xu, R. Jia, L. Mou, G. Li, Y. Chen, Y. Lu, and Z. Jin, "Improved relation classification by deep recurrent neural networks with data augmentation," *CoRR*, vol. abs/1601.03651, 2016.

[16] B. A. Maxwell, C. A. Smith, M. Qraitem, R. Messing, S. Whitt, N. Thien, and R. M. Friedhoff, "Real-time physics-based removal of shadows and shading from road surfaces," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1277–1285, 2019.

[17] S. F. Dodge and L. J. Karam, "Understanding how image quality affects deep neural networks," *CoRR*, vol. abs/1604.04004, 2016.

[18] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 509–516, 2014.

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.

[21] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "CNN-RNN: A unified framework for multi-label image classification," *CoRR*, vol. abs/1604.04573, 2016.

[22] H. Jung, M. Choi, J. Jung, J. Lee, S. Kwon, and W. Y. Jung, "Resnet-based vehicle classification and localization in traffic surveillance systems," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 934–940, 2017.

[23] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *CoRR*, vol. abs/1611.05431, 2016.

[24] S. Zagoruyko and N. Komodakis, "Wide residual networks," *CoRR*, vol. abs/1605.07146, 2016.

[25] N. Tahur, "The current state of the art in image classification applied on multi-label images of lego bricks," 2020.

[26] N. Japkowicz *et al.*, "Learning from imbalanced data sets: a comparison of various strategies," in *AAAI workshop on learning from imbalanced data sets*, vol. 68, pp. 10–15, Menlo Park, CA, 2000.

[27] "Summary and statistics." http://image-net.org/about-stats. Accessed: 2020-05-30.

[28] O. Kwon and J. M. Sim, "Effects of data set features on the performances of classification algorithms," Sep 2012.

[29] R. Oltmans, A. Lengyel, and J. van Gemert, "Optimal methods for crowdsourcing image annotation of multiple lego's," 2020.

[30] "rebrickable." https://rebrickable.com/.

[31] K. Nooney, "Deep dive into multi-label classification..! (with detailed case study)," Feb 2019.

[32] "Pytorch." https://pytorch.org/docs/master/generated/torch.nn.BCEWithLogitsLoss.html.

[33] J. O. du Terrail and F. Jurie, "Faster RER-CNN: application to the detection of vehicles in aerial images," *CoRR*, vol. abs/1809.07628, 2018.

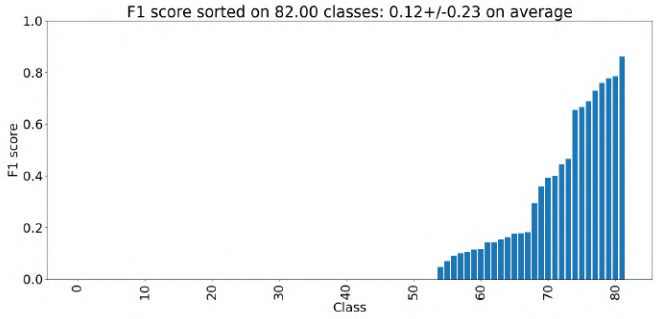
[34] M. Baker, "Is there a reproducibility crisis?," *Nature*, vol. 533, pp. 452–454, 05 2016.

[35] B. Kam, A. Lengyel, and J. van Gemert, "Lego data collection." https://gitlab.com/lego-project-group/lego-data-collection, 2020.

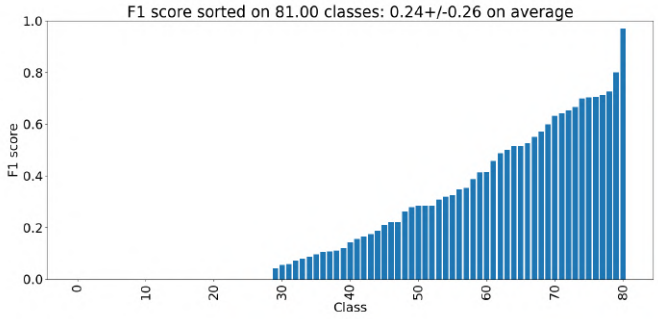
[36] B. Kam, R. Oltmans, H. Abderrazik, N. Tahur, D. Cian, A. Lengyel, and J. van Gemert, "Synthetic lego generation." https://gitlab.com/lego-project-group/lego, 2020.

[37] N. Tahur, A. Lengyel, and J. van Gemert, "Imageclass." https://gitlab.com/lego-project-group/ImageClass, 2020.

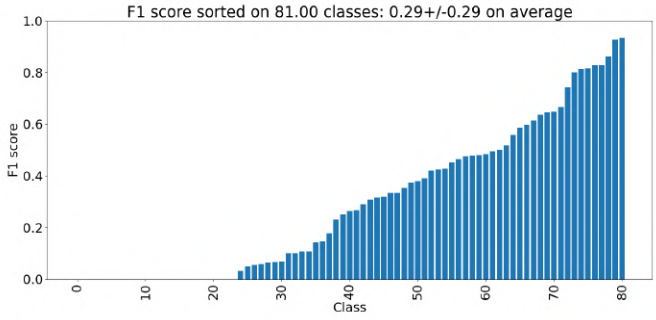
A Results



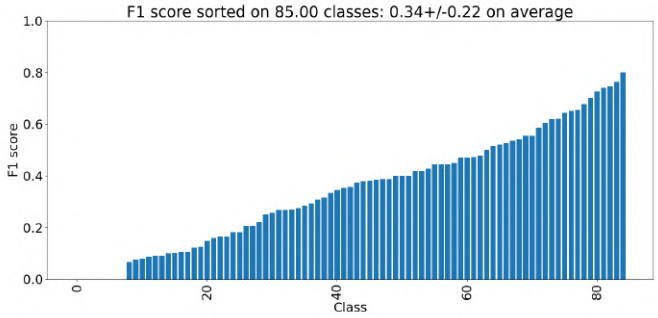
(a) Real3000s



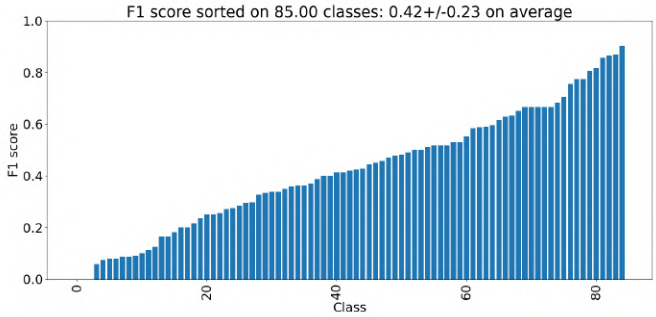
(b) Rendered3000



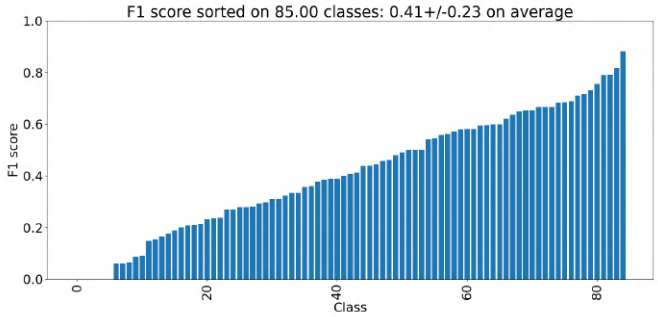
(c) Rendered5000



(d) CP3000

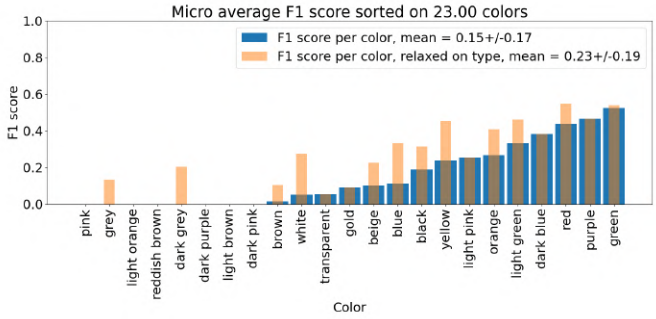


(e) CP5000

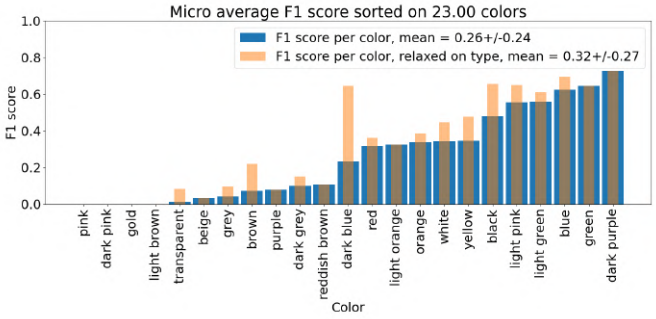


(f) CP10k

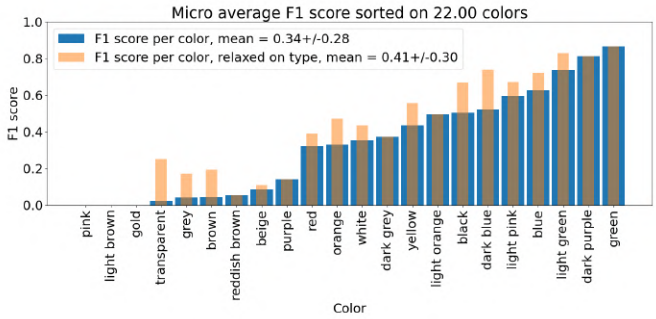
Figure 13: Micro average F1 score per class



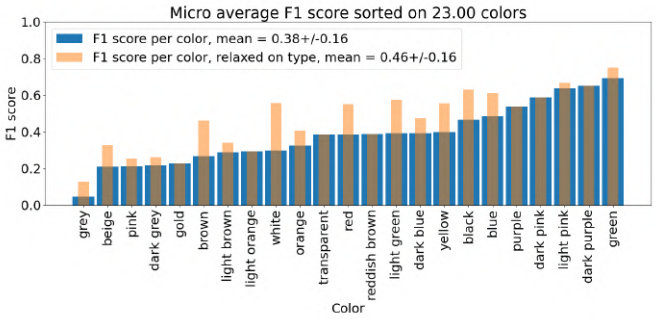
(a) Real3000s



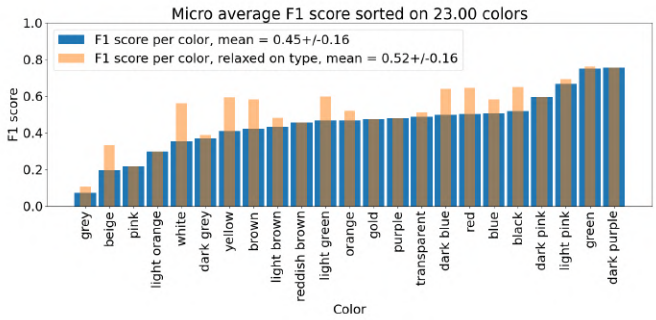
(b) Rendered3000



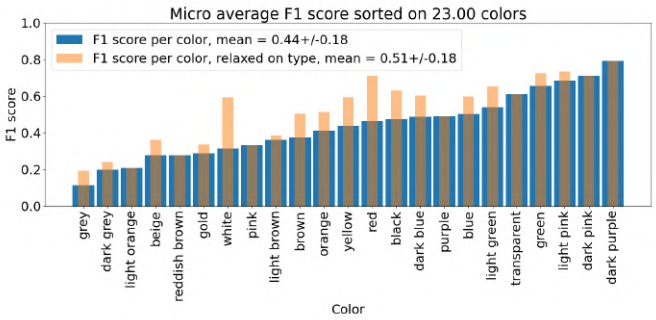
(c) Rendered5000



(d) CP3000

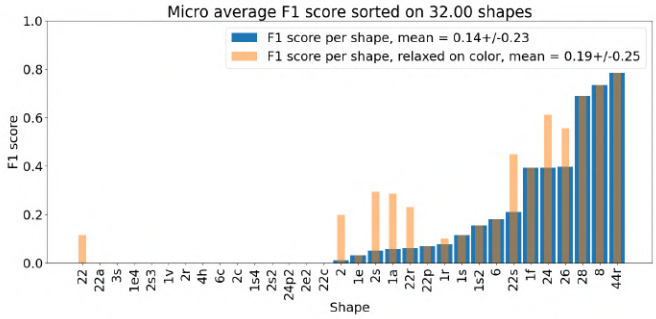


(e) CP5000

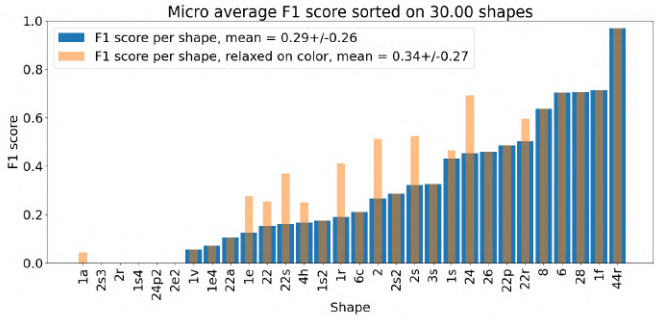


(f) CP10k

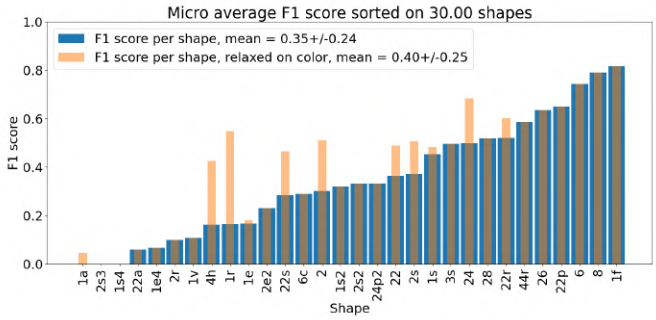
Figure 14: Micro average F1 score per color



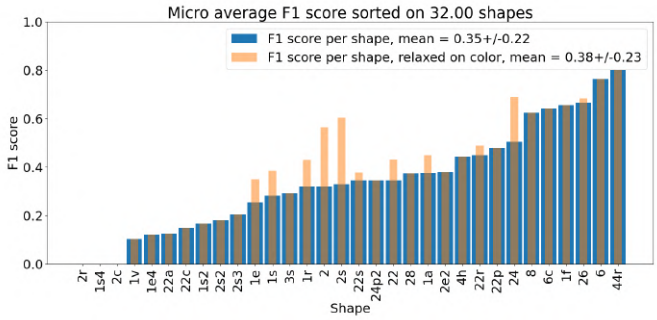
(a) Real3000s



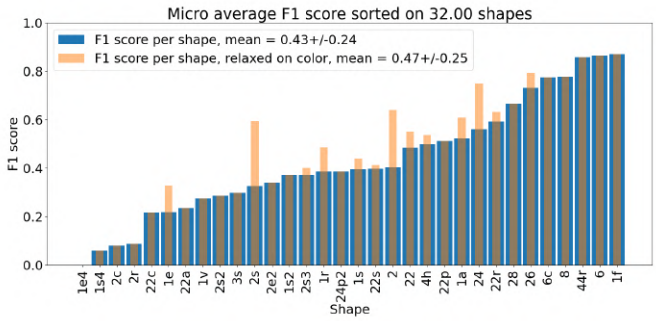
(b) Rendered3000



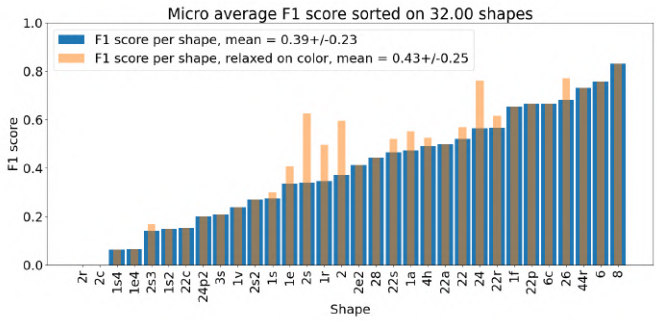
(c) Rendered5000



(d) CP3000

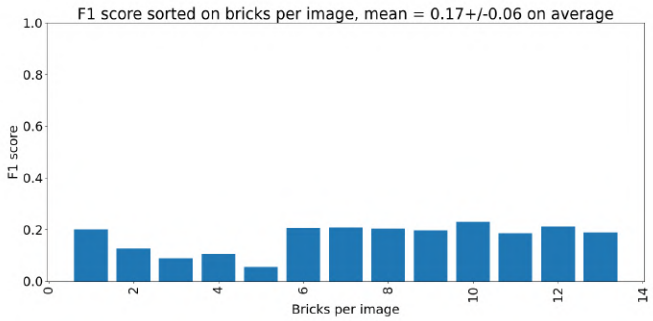


(e) CP5000

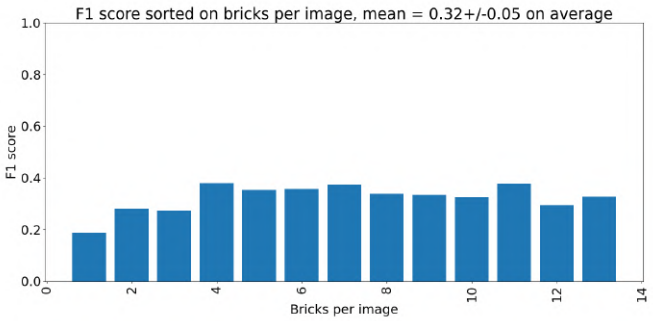


(f) CP10k

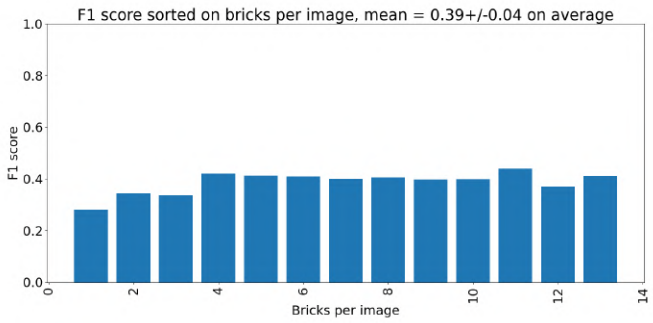
Figure 15: Micro average F1 score per shape



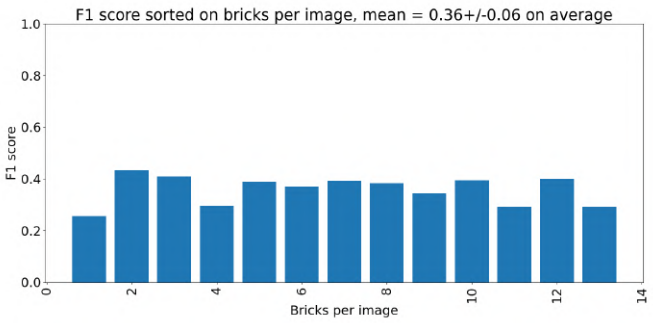
(a) Real3000s



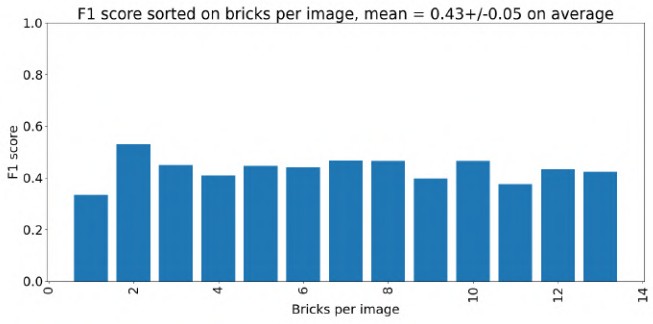
(b) Rendered3000



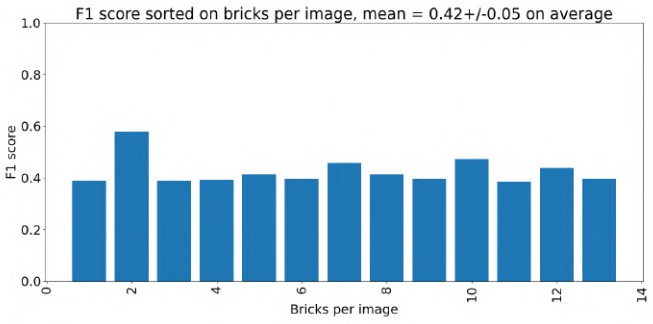
(c) Rendered5000



(d) CP3000



(e) CP5000



(f) CP10k

Figure 16: Micro average F1 score sorted on bricks per image