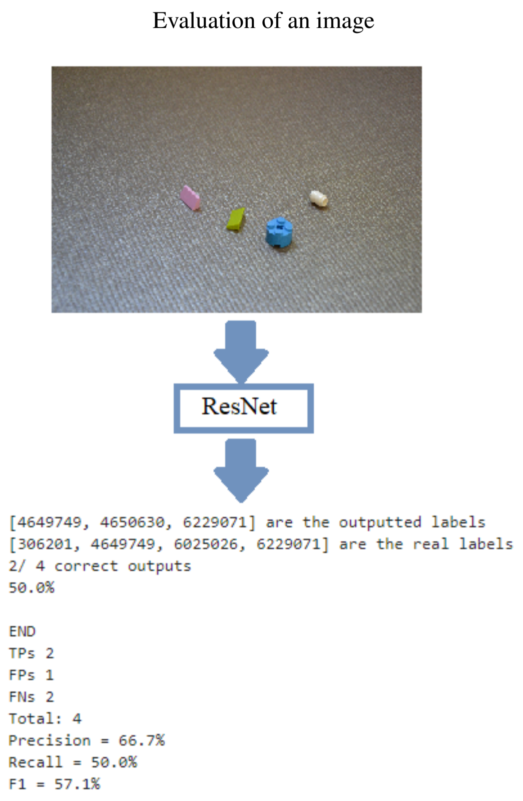
The Current State of the Art in Multi-Label Image Classification Applied on LEGO Bricks

Nishad Tahur, **Attila Lengyel**, **Jan van Gemert**
TU Delft
i.h.n.tahur@student.tudelft.nl, {A.Lengyel, J.C.vanGemert}@tudelft.nl

Abstract

This paper shows how the current state of the art in image classification performs on LEGO bricks. Currently the standard image classification models with deep learning are single label image classifiers. In this paper we will convert them to work on multi-label images and subsequently evaluate how well they perform. We show how well the classifiers will work on three different types of datasets. Experiments will be conducted on these three types of datasets to compare the performance of three different multi-label image classifiers. The main research question accompanying this paper is “How well does the state of the art in image classification work on LEGO bricks?”. Three subquestions are set up to answer this question. The first will regard the existence of the image classifiers. The second subquestion will regard how big the influence is of real life aspects, such as deterioration of the LEGO bricks. The final subquestion will be about the performance on the datasets. After answering these questions and conducting the experiments, we came to the conclusion that the ResNext model performed the best on almost all of the categories. Based on the numbers of the results we can also conclude that the models should perform well with multi-label images of LEGO bricks.



```
[4649749, 4650630, 6229071] are the outputted labels
[306201, 4649749, 6025026, 6229071] are the real labels
2/ 4 correct outputs
50.0%

END
TPs 2
FPs 1
FNs 2
Total: 4
Precision = 66.7%
Recall = 50.0%
F1 = 57.1%
```

Figure 1: The process of evaluating an image with ResNet representing a trained model.

1 Introduction

Image classification is a fundamental problem when it comes down to computer vision [14]. Other computer vision tasks such as object detection and localization heavily depend on image classification as it forms their basis [21]. Currently there are many image classification models making use of deep learning. Deep learning is the process in which training data, which is similar to another problem, is given to the system. The goal is to train the system automatically so that it can solve future problems without needing any extra information [17]. There are multiple settings when it comes to deep learning for image classification, these are single label classification and multi-label classification for example. When given an image of a cat, it is the task of the single label image classifier to return the result “cat”. However, when a picture with multiple objects in it is presented for a multi-label image classifier, the classifier should return every object that is present in the image. Great success has been achieved within the field of single label image classification [25], however, multi-label classification still seems to be quite the challenge [30].

An interesting application for this multi-label image classification is on LEGO bricks. The reason is that not only can this be applied on real world situations but also due to the fact that this research can be taken as a basis for other research. In the real world for example, there are companies that scan satellite images for cars to predict serveral things such mall sales [19]. We will be researching three of the top image classifiers on a set of LEGO bricks. The problem will be as follows. The models will be fed images which have up to thirteen of those LEGO bricks in it. The images will be process and at the end, the model will have to return the subset of bricks present in the image. An example of the input and output can be found in figure 1This will be done for three different types of datasets. One is a real dataset and the other two are synthetic datasets. The real dataset will be images which are taken by hand with a camera. The synthetic datasets will be divided into a dataset which is generated with a modeling program while the other will be created by cutting LEGO bricks from real images and pasting it on random backgrounds. The former we will refer to as the synthetic dataset while the latter we will refer to as the cut and paste dataset. With these results we will be able to evaluate the performances of the modified multi-label image classifiers.

We will research how good the current state of the art in image classification works on LEGO bricks. To this end, three subquestions have been set up to help answer the main question. The first research question is “What image classifiers currently exist?”. Currently there are many image classifiers available on the official website of PyTorch, however, due to time constraints, only three image classification models will be chosen for this research. The three chosen ones will be the top three available on their website. The second subquestion will be, “How well does real data perform against synthetic and cut and paste data?”. This will be done by comparing the results of the different datasets. The final subquestion is “How well does a model perform when it is trained on synthetic or cut and paste data and evaluated on real data?”.

The main contribution of this paper will be the analysis and modification of different multi-label image classifiers applied on LEGO bricks. This will provide the information about whether the models currently work on multi-labeled LEGO brick images in the first place. This paper will also provide the results of the performances on different datasets with the purpose of showing how the models perform on the different circumstances, such as the difference in lighting, details of background and reflection. Finally, this paper will also provide the comparison of the performance of the multi-label image classifiers on near perfect data and real life data. The near perfect data is synthetic data in which the objects are perfectly clean, have a consistent color and reflect light perfectly.

To report our findings as efficiently as possible, first a description of the models will be given together with related work. The training and evaluation process will then be explained after which the experiments follow. With the experiments there will be conclusions and an overall result and discussion of the results is given afterwards. Finally, the conclusion and the future work will be discussed.

2 Related Work

2.1 Feature Extraction

Image classification is a task in the computer vision world that refers to the process of classifying an image based on the contents of the image [12]. The input for such a classifier is the image, the features of the image and what the image represents. Image classification with deep learning is the process of training an image classifier with input data that only consists of the image and what the image represents. The difference between classical image classifiers and image classifiers based on deep learning, is that the latter learns to extract the features of an image itself [3]. There are many different methods to extract features from an image. An image can have color, texture and shape features for example [16]. ResNet down samples the images to extract the features. While down sampling there are different combinations of average pooling and max pooling. That is a layer that goes over all of the pixels of an image and creates groups of pixels. Next it converts those groups of pixels into one pixel with a certain value. For an average pooling the value of the new pixel will be the average value of all the pixels in that group. For max pooling the new value will be the maximum value of the group of pixels [1]. ResNet and variations of ResNet will mostly be used in this research.

2.2 LEGO Classification

There have been people who have tried to implement LEGO classification before. Daniel West, for example, created a LEGO sorter [20]. This application had the limitation that the bricks had to be analyzed one by one before sorting them. This reduced the problem to a single label classification problem. Another group of students tried to create a machine that does this. However, they realized that this is a hard task and limited themselves to five pieces. Even with that limitation, they did not manage to finish the research in the time given [29]. To summarize, both multi-label classification and LEGO brick classification are implemented, however, the application of the first on the second is missing. In this research a machine will not be created, however, we will attempt to do what they did not, evaluate multiple pieces at once.

2.3 ResNet Signature

A popular method for these multi-label image classification problems is the use of residual networks (ResNet) [11]. ResNet introduced a network which is similar to a plain convolutional neural network, however, with the addition of shortcut connections between layers [15]. These shortcut connections have as a consequence that if the output of a layer corresponds with the input of two layers further on, an identity mapping is executed [9]. This reduces the training difficulty and the error rate of the system. Because of this implementation, a convolutional neural network of up to a thousand layers could be realized [10]. This was a great achievement in the world of image classification with deep learning.

2.4 Single Label vs Multi-Label Evaluation

ResNet is one of the image classifiers available right now on PyTorch that is tested on the large dataset of ImageNet [6]. Alongside ResNet, there are twelve other image classifiers available. They are tested in [22] which is a challenge to help set a benchmark for new computer vision image classifiers. However, these classifiers have been trained on single label images. The difference between single label and multi label image classifiers is the way they evaluate the output. A softmax layer is usually applied to a single label image classifier [8]. A softmax layer balances the output. That means that if the model most likely expects the image to contain a cat, all other classes are less likely to appear in the image. The output will be in percentages for each class such that at the end, all percentages add up to a 100%. Each class is represented as a certain percentage to be the description of the image. For a multi-label image classifier, however, every class has a random chance to appear in the image and is mostly not dependent on the other labels. The total amount of percentages could thus be higher than 100%. Since this research will be a multi-label image classification problem, the standard models will be used, however, the softmax layer will be changed.

2.5 Loss Function and Optimizer

Another important feature of the image classifier is which loss function is being used. This is necessary such that during the training, the loss can be calculated and parameters within the model can be tuned for better results [4]. It is used to measure how well it can predict the expected outcome. Together with an optimizer, the loss function makes it so that the model can train to improve. The job of the loss function is to get to a loss as low as possible, however, throughout the learning process there are multiple local minima. To make sure that the model does not get stuck at such a local minimum, the optimizer is there to look at the bigger picture. In this research we will be using both a loss function and an optimizer, however, they will not be the standard ones that PyTorch provides with their models.

2.6 Classification Networks

For the models themselves, the ResNet, WideResnet and the ResNext models are currently the top 3 models on PyTorch. These three models have different signatures. ResNet [9] uses shortcut connections in the convolutional layers of the network. This makes it such that network could become deeper than before. WideResNet [28] on the other hand, increases the amount of features in a convolutional layer in the act of reducing the depth of the network but increasing the width. ResNext [27], however, has shown that cardinality is more important than the width and the depth of a convolutional neural network. With cardinality, the layer does not have just one transformation, but instead has an aggregration of transformations based on the cardinality. The cardinality was compared to both the width and the depth in [27], and the results showed that it outperformed both. We will be using these three models to evaluate the performances.

3 Method

An important part of image classification with deep learning is the start where the model is trained with thousands of images with their corresponding labels. It will be important to provide the right labels accompanying these images. Currently the models take in images in batches of 16 for example accompanied by a vector of length 16, representing the labels. If there are 5 classes, each class will be tied to a number. The values in the vector of 16 will then each be the number representing the image. In our application we change this the following way. The input would still be in a batch of 16 images, however, each image is represented with a vector which is 85 long. Each number representing a unique class. The vector consists of 0's and 1's depending on whether the class was present in the picture or not. A 0 stands for false while a 1 stands for true.

We did this because we had to change the loss function. The loss function that is currently being used by the PyTorch image classifiers is the *CrossEntropy* loss function. This function applies a softmax layer at the end. Since we are tackling a multi-label problem, we do not want this. The loss function we will use is *BCEWithLogitsLoss* because this binary loss function works well on multi-label image classification problems. This loss function required the labels to be in a vector of 0's and 1's of length equal to the total amount of classes. The optimizer accompanying this was the *SGD* because the *SGD* does not apply a softmax layer at the end and thus the representation of each class will not be dependent on the other classes.

3.1 Training

The models used in this research were all implemented with PyTorch [18]. To create these models, all of the same modifications have been made. Due to performance issues, the ResNet, ResNext and WideResnet of 50 layers have been used. While training all of the models will use the same loss function, *BCEWithLogitsLoss*, and the same optimizer, *SGD*. The optimizer will have an initial learning rate of 0.1 which will be divided by 10 for every 20 epochs. The optimizer will have a momentum of 0.9 and a weight decay of 1e-5. Before processing the images, the images will be normalized according to the normalization used for all the PyTorch models. To calculate the accuracy and validation accuracy during the training, the F1-score will be used. A sigmoid will be applied to the output such that the 85 classes' appearance are represented with a percentage. The threshold applied to the models will be 50%. This means that if the number after the application of the sigmoid is bigger than 0.5, it has a bigger than 50% chance to appear in the scene and will thus be seen as one of the present objects. Furthermore, all of this will be done on the pretrained models. The only modification made to the models themselves are that the final layers are changed to a layer that has 85 output nodes. All of the models will be implemented in PyTorch and the same tweaks will be made for each of the models currently available on the official website of PyTorch.

3.2 Evaluation

To evaluate the results, a convention similar to what was used in [22] will be adhered to, the top-1 error. [9; 27; 28] also used this metric to evaluate their models. The top-1 error means the percentage of the results in which the top prediction was not the correct result. To transform this to a multi-label image classification metric, the exact match ratio metric will be used to evaluate the models. This is comparable to the single label classification version because the difference between the evaluation of single label and multi-label image classifiers is that the answer for the latter can also be partially correct. To circumvent this problem to decide how much must be correct to consider the final result to be correct, the exact match ratio (MR) is a metric that only counts the result as correct if every prediction is correct. It is calculated with the following formula:

$$MR = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i). \tag{1}$$

The n represents the amount of images, the Y_i represents the predicted vector of labels of the i^{th} image and Z_i represents the ground truth for the i^{th} image. The I is the indicator function. Accompanying this metric, the precision, recall and F1-score will also be used. The same symbols are used as in the MR and the symbols have the same meaning. The precision is used to calculate which percentage of the correct answers were returned and is calculated the following way,

$$P = \frac{1}{n}\sum_{i=1}^{n} \frac{|Y_i \cap Z_i|}{|Z_i|}. \tag{2}$$

The recall calculates which percentage of the predicted answers were correct and is calculated with

$$R = \frac{1}{n}\sum_{i=1}^{n} \frac{|Y_i \cap Z_i|}{|Y_i|}. \tag{3}$$

The final metric used, the F1-score

$$F_1 = \frac{1}{n}\sum_{i=1}^{n} \frac{2|Y_i \cap Z_i|}{|Y_i| + |Z_i|}, \tag{4}$$

is the harmonic mean between the two meaning it takes into account both the precision and the recall scores [23]. These metrics combined, give a clear overview of how the models perform, which is why we chose for these four. These metrics also give an easy overview in the context of LEGO bricks. These four measures will be applied to the three types of datasets individually and then the results of the datasets will be compared to each other.

4 Experimental Setup and Results

We will perform experiments on three custom-made datasets: the synthetic dataset, cut and paste dataset and the real dataset. These datasets will each have 85 classes, each representing a certain LEGO brick and the datasets will be as balanced as possible. We will conduct a total of five experiments and each experiment will be performed on three classification models to compare the results. For all five experiments, the models will be trained on the same circumstances with a minor change for every experiment. For these experiments there were a total of three models trained for every dataset so 18 training sessions. Judging by the accuracies, losses, validation accuracies and validation losses in figures 2 and 3, the models did not improve that much after 20 epochs. Thus we chose to train the models for 40 epochs and use those states to evaluate the performance of the models.

Datasets

There will be a total of six different datasets out of the three types of datasets. Due to the time constraint of this problem, only 3000 real images can be taken by hand. The process of how these images have been taken can be found in [13]. Since this was a lengthy process however, another way of testing the models was necessary. That is why synthetic data had to be generated to train the models while the real data was in the process of being made.

The synthetic data was made with the help of a 3D modelling program called Blender [5]. The goal of the synthetic data was to be as realistic as possible compared to the real data. To this end, the same constraints and settings were adapted. A constraint of the real data was that there could only be up to 13 LEGO bricks in a single image. The same constraint was put upon the synthetic data. A second point that was taken into account was that the same backgrounds were used as in the real data. This was done by taking real images of just the backgrounds and using them as the background for the synthetic data. For the synthetic dataset, there was no real constraint as to how much data could be generated except for the time constraint. This resulted in the generation of 5000 synthetic images. Since the real data was only 3000 images big, two synthetic datasets had been created. One was to evaluate the performance of the models on the synthetic dataset. This dataset was 5000 large. The other one was 3000 big to try to keep the comparison with the real data as even as possible.

The third dataset was the cut and paste dataset. This is another method for creating a synthetic dataset. The creation of this data can also be found in [13] Since this data is created by cutting out LEGO bricks from real images and pasting them on real background, this dataset also has the real world effects applied to the images. This means that the deterioration of the bricks for example can also be seen in this type of data. Other aspects like the reflection of the light for example is less realistic, this is an aspect that is handled well in the synthetic dataset. The cut and paste dataset will have three datasets in total. One will be 3000 big to compare with the real data. The other two will be of size 5000 and 10000.

All the dataset will be split into 80% for the training, 10% for the validation and 10% for the test dataset. This is a common division so as to maintain a large set to train on while still having enough data to validate and evaluate on [7]. While training, the training dataset will be fed to the models in batches of 16 images, however, every time the the whole training set is traversed, the batches will be randomized.

Experiment 1: Synthetic Dataset

The first experiment will have a synthetic dataset of 5000 images. In this experiment we will be comparing the performance of the models in a synthetic data environment. The synthetic data will be generated with Blender and with the use of the physics engine random bricks of up to 13 will be generated into a scene. These will be dropped onto a plane with a random background and a render will be taken from a random angle facing the bricks.

The constraints of this dataset will be as follows: all of the images used will be cropped to 384x256 (width x height) pixels, such that the aspect ratio of the LEGO bricks can be maintained. Furthermore, the data fed to the models while training will be in batches of 16 and the labels accompanying

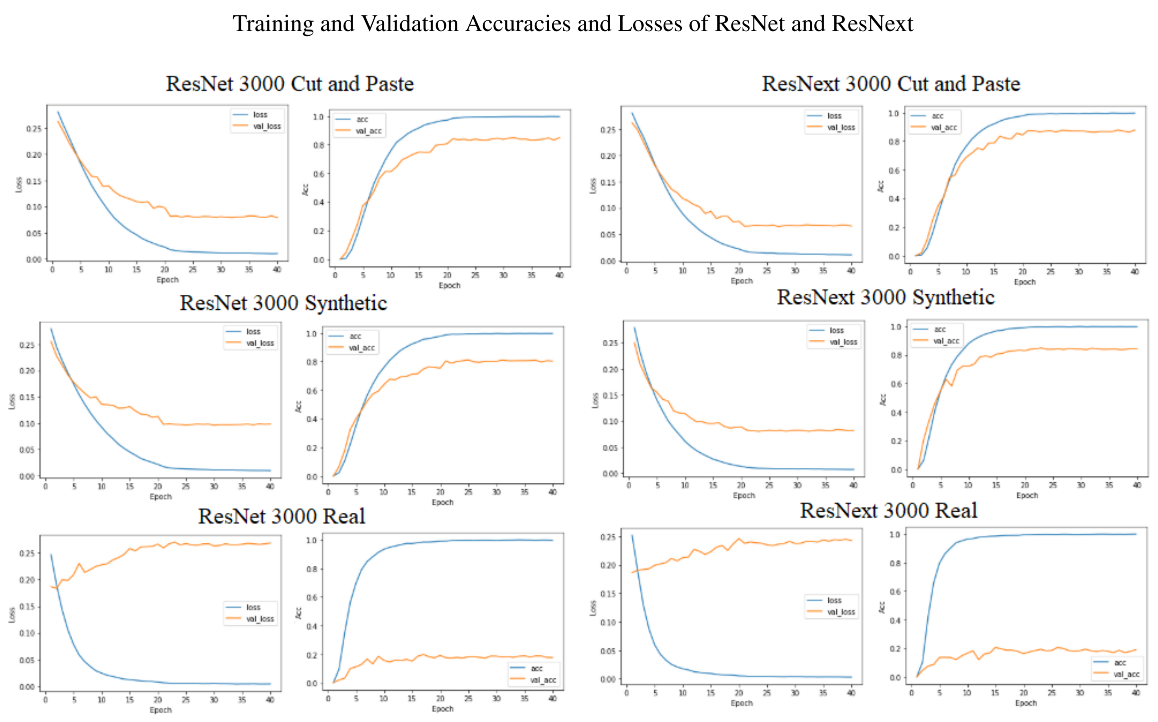


Figure 2: The losses, accuracies, validation losses and validation accuracies of the ResNext and ResNext model on the three different types of datasets. The acc stands for accuracy and for the synthetic and cut and paste data, the graphs seem to behave as they should, however, for the real data this does not seem to be the case.

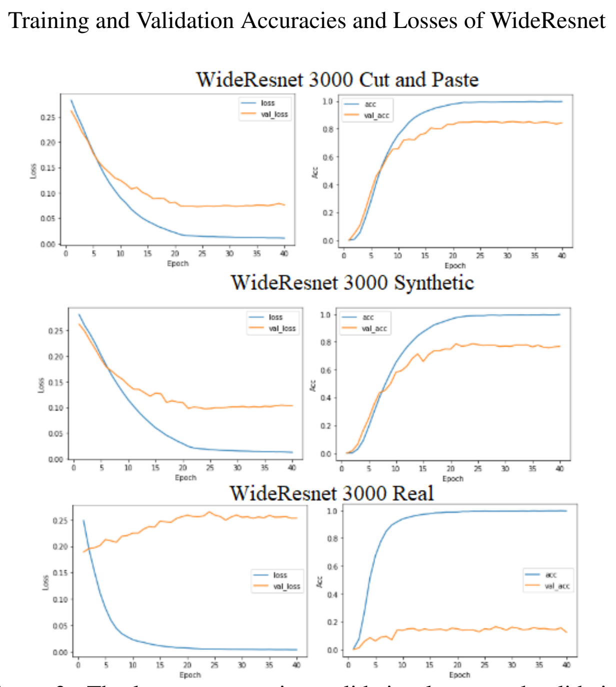


Figure 3: The losses, accuracies, validation losses and validation accuracies of the ResNext and ResNext model on the three different types of datasets. The accuracy rises up to 80% for the synthetic data and cut and paste data, but does not rise that much for the real data. The same goes for the loss, the loss drops by much for the synthetic data and cut and paste data, but not as much for the real data.

them will be a batch of 16 vectors of 0's and 1's with a length of 85, representing all classes. If the brick is in the image, the accompanying label will be a 1, else it will be a 0. Finally, the model will be trained for 40 epochs meaning that every image will be trained on 40 times. The purpose of this experiment is to research how well multi-label classifiers work on synthetic images when the situation is perfect. That means when the bricks are in perfect shape and the colors are consistent. The question accompanying this experiment will be "How well do multi-label image classifiers perform on near perfect data?"

In Table 1 we can see the results of the three chosen models. In [26] for example, multiple different multi-label image classifiers have been tested on three benchmark datasets. The F1-scores of the classifiers in [26] were approximately between 40% and 60%. Compared to these numbers, the three models modified for this research perform very well on LEGO bricks with an approximate F1-score of 91%. In particular, the models are performing well when having to return most of the results, but in the process also return wrong predictions. This can be seen due to the fact that the precision is higher than the recall. The exact match ratio also shows that the models do not perform that well yet on returning exactly the correct results. The exact match ratio for the modified models, however, is approximately 50%. The pretrained versions on PyTorch have a top-1 error of approximately 20% meaning an approximate accuracy of 80%. Compared to this, the models do not perform that well for the exact match ratio. A note on this however is that the exact match ratio is very strict. This means that if an image would have predicted

Experiment 1: Evaluation on 500 synthetic images

	ResNet	ResNext	WideResnet
Total images	500	500	500
Total labels	3403	3403	3403
True Positives	2860	2999	2869
False Positives	177	107	127
False Negatives	543	404	534
Exact Match Ratio	45.4%	54.6%	44.4%
Precision	94.5%	96.4%	95.7%
Recall	87%	90.4%	87.4%
F1-score	90.1%	92.9%	90.9%

Table 1: The results of the three modified classifiers ResNet, ResNext and WideResnet on 5000 synthetic images. There are a total of 500 synthetic images which are tested on and together they have 3403 labels. Out of those the true positives amount to the correctly predicted results, the false positives amount to the labels predicted as correct but are in fact not correct and the false negatives amount to the ground truth labels which were not predicted. From the three results it is clear that the ResNext model performs better than the ResNet and WideResnet. However, judging by the numbers themselves, the number are really high so that means that the models perform well on synthetic data where the state of the bricks, lighting, etc. are "perfect".

9 out of 10 labels correctly, it would have still been seen as incorrect. From this experiment we can deduct that the modified image classifiers do perform well on LEGO bricks when the bricks are in near perfect condition. This means when the bricks have their original shape, color, and reflection of light.

Experiment 2: Cut and Paste Dataset

The second experiment that will be conducted is on the cut and paste dataset of 5000 images. Once again all of the images will be resized to 384 by 256 pixels and the models will be fed images in batches of 16. The accompanying labels will be in a vector of 85 and the model will be trained for 40 epochs. The purpose of this experiment is to test how the models work on another type of synthetic data, namely synthetic data in which the LEGO bricks are more realistic. Since the cut and paste data consists of bricks which are cut from real images, the deterioration and recoloring of the bricks are also taken with them to the new images. The question we will try to answer is "How do multi-label image classifiers perform on somewhat realistic LEGO bricks?"

The results of the modified ResNet, ResNext and WideResnet models can be seen in Table 2. The results are once again really good and the precisions seem to be higher than the recalls. The ResNext models performs better than the other models which also corresponds to the fact that the ResNext model is the best performing model on PyTorch. The exact match ratio, however, is approximately the same as for the evaluation on the synthetic data, which was not that high. The results are even higher when compared to the synthetic data. Since the test set consists of images that have never been seen

Experiment 2: Evaluation on 5000 cut and paste images

	ResNet	ResNext	WideResnet
Total images	500	500	500
Total labels	3421	3421	3421
True Positives	2957	3052	2995
False Positives	98	54	70
False Negatives	464	369	426
Exact Match Ratio	47.4%	52.8%	49.4%
Precision	97%	98.4%	97.6%
Recall	88.7%	91.1%	89.7%
F1-score	92.2%	94.3%	93.1%

Table 2: The results of the three classifiers ResNet, ResNext and WideResnet on 5000 cut and paste images. The precision, recall and F1-score for the three classifiers are really high meaning that in almost all of the images, the classifiers are able to classify most of the bricks present. From the exact match ratio, however, it can be concluded that the models do not work that well yet on predicting the exact correct bricks.

before by the model, this means that the models are able to perform at least just as well on synthetic LEGO bricks as on LEGO bricks which have a more realistic condition regarding the state of the bricks.

Experiment 3: Evaluation of 300 real data images

	ResNet	ResNext	WideResnet
Total images	300	300	300
Total labels	2076	2076	2076
True Positives	285	323	219
False Positives	398	356	436
False Negatives	1791	1753	1857
Exact Match Ratio	1.8%	1.8%	1.8%
Precision	35%	42.5%	30%
Recall	14.5%	16.4%	11.8%
F1-score	18.8%	21.4%	14.8%

Table 3: The evaluation of the three modified image classifier on real data. There are a total of 300 real images and the true positives are really low when compared to the total amount of labels. The results are between the 14 and 22%. This means that results from this experiment were low. The exact match ratio is also really low. Only 1.8% images out of the 300 images had the exact ground truth as the prediction.

Experiment 3: Real data

The third experiment will be with real data. The real dataset will be 3000 images big. To keep the models from returning overfitted results, we made sure to only include images in the validation and test set that are not trained on. The reason we handpicked the validation and test dataset is because the data itself was be shot by hand following a certain script [13]. This measure was taken so that we could make sure that

Experiment 4: Evaluation on real, trained on synthetic and cut and paste

	Synthetic vs Real			**Cut and Paste vs Real**		
Models	ResNet	ResNext	WideResnet	ResNet	ResNext	WideResnet
Total images	300	300	300	300	300	300
Total labels	2076	2076	2076	2076	2076	2076
True Positives	625	691	519	805	737	763
False Positives	985	867	996	629	566	664
False Negatives	1451	1385	1557	1271	1339	1313
Exact Match Ratio	1.4%	3.6%	1.2%	4%	3%	4.4%
Precision	37.8%	44.8%	32.4%	59.2%	55.7%	56.4%
Recall	29.4%	36%	25.4%	42.7%	37.8%	41.5%
F1-score	32.2%	38.7%	27.5%	48.3%	43.9%	46.8%

Table 4: Evaluations on a real dataset when trained on a synthetic or cut and paste dataset. For both situations the evaluation is done with 300 real images with a total of 2076 labels. On average the evaluations when trained on cut and paste is higher when compared to the synthetic dataset. TThe percentages for the precision, recall and F1-score are higher. This means that on average when the model is trained on cut and paste data, that is is able to return more of the correct answers and more correct answers compared to wrong answers.

the data was balanced at the end. The balance of data is important because if a class is overrepresented while training, the model is more likely to output that class. This experiment will be the most meaningful for this research, since this experiment will be conducted on real data. All of the aforementioned constraints for the synthetic and cut and paste data also apply for the real data. The results of this experiment will show whether the current state of the art actually performs well enough. The question for this experiment will be "How well does the current state of the art in image classification perform on LEGO bricks?"

The results as can be seen in Table 3 are not that great. When compared to the models in [26], the F1-scores are approximately twice as low. The exact math ratio is also low which means that either the models are not good enough yet to evaluate real data or there was an external factor influencing the results.

Experiment 4: Synthetic & Cut and Paste Data vs Real Data

The fourth experiment will be a combination of the synthetic data, cut and paste data and the real data. Each dataset will consists of 2400 training images, 300 validation images and 300 test images. In this case the models will be trained on synthetic data and cut and paste data and tested with real data. This is to test whether the real life situations have a big impact on the performance of the model. Real life situations can be the state of the LEGO bricks for example. Another reason to test this is to see if the synthetic and cut and paste data are representative for the real data. In the synthetic data, each brick will be perfect in size shape and color. In real life, however, it could be the case that the quality the brick has deteriorated meaning that the shape has somewhat changed or that the color has changed. For cut and paste data the lighting will be really unrealistic and this might also have an effect on the performance. The question accompanying this research will be "Are the results from the synthetic and cut and paste data representative for the performance on real data?"

The results in Table 4 show that the models perform better on real data when trained on cut and paste data than when trained on synthetic data. The exact match ratio is also really low with a maximum of 4% meaning that in the best case, only labels from 12 out of 300 images were predicted exactly. Overall the ResNext model seems to perform the best when trained on synthetic and tested on real, however, for the cut and paste it is performing the worst. The modified models when trained on cut and paste, however, have scores of approximately 40 to 50%. When comparing these numbers to the numbers in [26], they almost fall in the same range since the scores in the paper are approximately 40% to 60%. The results when trained on synthetic data, however, do perform worse when compared to the models in the paper. Judging by these results, the modified models perform decently well when trained on synthetic or cut and paste data and evaluated on real data.

From experiment 4 we can also conclude that experiment 3 had an external factor that influenced the performance of the modified models. Due to time constraints, the real data was shot according to a script. To save time, the script gave the same combination of bricks and backgrounds 3 times. This caused the models to overfit on the data and perform not that well the evaluation.

Experiment 5: Cut and Paste Data 10.000 Images

It is generally known for image classification that more data is better [2]. With the following experiment we want to investigate whether the performance on real data becomes better when it is trained on twice as much cut and paste data. This dataset will contain 10.000 cut and paste images of which 8000 will be used for training and 1000 for validation. With this experiment we will be able to find out if the models do indeed work better when more data is present. The question accompanying this experiment will be, "Do models trained on more cut and paste data also perform better on real data?"

We can see in Table 5 that the models do perform relatively

Experiment 5: Trained on 10.000 cut and paste images, evaluated on 300 real images

	ResNet	ResNext	WideResnet
Total images	300	300	300
Total labels	2076	2076	2076
True Positives	993	947	942
False Positives	778	857	775
False Negatives	1083	1129	1134
Exact Match Ratio	7.2%	7%	8%
Precision	59.7%	55.9%	57.5%
Recall	52.5%	50.2%	51.6%
F1-score	54.9%	52.2%	53.4%

Table 5: Evaluation on 300 real images when trained on a dataset of 10.000 cut and paste images. Generally the precision, recall and F1-scores seem reasonably high. The models are able to predict almost half of the labels correctly but in the process also predict almost the same amount incorrectly. The exact match ratio is low when it is compared to the top-1 error of the standard models on PyTorch.

well when compared to the models in [26]. The exact match ratio, however, is relatively low. The models also output more correct results than wrong results, however, the numbers are quite close. The models also not predict correct labels more often than prediction the correct ones. This causes the precision to be higher than the recall. Overall the F1-score of the models when trained with more data cut and paste data, performs better on real data. There is an approximate increase of 6 to 8%.

Results and Discussion

Overall the modified models seem to perform really well on the synthetic data and cut and paste data. The models perform less well on real data, however, from the experiments and figures 2 and 3 we can conclude there there was another factor influencing the performance of the models, namely the data. With these results, we can say that the current state of the art in multi-label image classification applied on LEGO bricks performs decently well but there is room for improvement.

A limitation of the models, as well as a limitation for image classification in general, is the data used for the models. The experiment with the real data was not that big. Not only was it not big, every image also appears three times with a slight difference. This causes the models to overfit, giving really bad statistics while evaluating.

5 Responsible Research

All the data used for this research can be ethically justified. The real data that is used to test the models is shot with a camera by other students researching a similar topic. In this research I have indirectly worked together with four other students, Berend Kam, Hiba Abderrazik, Rembrandt Oltmans and David Cian. We worked together with Berend in the sense that my model was tested on real data which was part of his research. However, thousands of images had to be taken by hand so he got help from Hiba and David who also needed it for their research. The cut and paste data has also been generated by Berend Kam. The synthetic data was initially developed by Rembrandt Oltmans but the improved by David with Hiba's and my help. David and Hiba also needed the synthetic data for their own research.

This research is reproducible in the sense that all of the code and saved states of the models are available on [24]. The data used for this research is available on the storage servers of the TU Delft. All other resources like the models used are available for the public on the official website of PyTorch. To mimic the model We have used for this research, the PyTorch models have to be tweaked according to the things mentioned in this paper. If all of these conditions are met, this research can be reproduced with similar results.

6 Conclusion and Future Work

We modified three of the current best performing single label image classifiers to work on multi-label problems. We changed the output layers of those to work on LEGO brick instances. We also used a different loss function and this enabled us to properly evaluate the performances of the models on LEGO brick images.

In this research we used the the pretrained models ResNet, ResNext and WideResnet of only 50 layers. The official website of PyTorch had models with more layers available, however during the research we got poor results because the models would overfit on the type of input data and underperform on the other data. To overcome this problem, the pretrained models with fewer layers were chosen. Future work could instead work on finding the ideal values for the optimizer and the learning rate or even find another way of implementing these models.

We evaluated the current state of the art in image classification on LEGO bricks. Three types of datasets were created for this purpose, a synthetic dataset, a cut and paste dataset and a real dataset. The models modified for this research performed really well on the synthetic and cut and paste dataset, however, performed poorly on the real dataset. When the models were trained on the synthetic or cut and paste dataset and evaluated on the real dataset, the latter also outperformed the former. Finally, the modified models also showed that training on more data, gives better results at the end.

References

[1] Madhushree Basavarajaiah. Maxpooling vs minpooling vs average pooling. *Medium*, Februari 2019.

[2] Aleksey Bilogur. Boost your cnn image classifier performance with progressive resizing in keras. *Towards Data Science*, April 2019.

[3] Anne Bonner. The complete beginner's guide to deep learning: Convolutional neural networks and image classification. *Towards Data Science*, February 2019.

[4] Jason Brownlee. How to choose loss functions when training deep learning neural networks. *Machine Learning Mastery*, January 2019.

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Rachel Draelos. Best use of train/val/test splits, with tips for medical data. *Glass Box*, September 2019.

[8] Rachel Draelos. Multi-label vs. multi-class classification: Sigmoid vs. softmax. *Glass Box*, May 2019.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[11] Heechul Jung, Min-Kook Choi, Jihun Jung, Jin-Hee Lee, Soon Kwon, and Woo Young Jung. Resnet-based vehicle classification and localization in traffic surveillance systems. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 61–67, 2017.

[12] David Kaeli, Perhaad Mistry, Dana Schaa, and Dong Ping Zhang. Chapter 9 - case study: Image clustering. In David Kaeli, Perhaad Mistry, Dana Schaa, and Dong Ping Zhang, editors, *Heterogeneous Computing with OpenCL 2.0*, pages 213 – 228. Morgan Kaufmann, Boston, 2015.

[13] Berend Kam. Dataset generation methods for multi-label images of lego bricks. 2020.

[14] Andrej Karpathy et al. Cs231n convolutional neural networks for visual recognition. *Neural networks*, 1:1, 2016.

[15] Sihan Li, Jiantao Jiao, Yanjun Han, and Tsachy Weissman. Demystifying resnet, 2016.

[16] Seyyid Ahmed Medjahed. A comparative study of feature extraction methods in images classification. *International journal of image, graphics and signal processing*, 7(3):16, 2015.

[17] Michael A Nielsen. *Neural networks and deep learning*, volume 2018. Determination press San Francisco, CA, USA:, 2015.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[19] Josh Patterson and Kirit Basu. Solving real-world business problems with computer vision. applications of cnns for real-time image classification in the enterprise., September 2017.

[20] Katyanna Quach. You looking for an ai project? you love lego? look no further than this reg reader's machine-learning lego sorter. *The Register*, December 2019.

[21] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[23] Mohammad S Sorower. A literature survey on algorithms for multi-label learning.

[24] Nishad Tahur. Imageclass. https://gitlab.com/lego-project-group/ImageClass, 2020.

[25] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[26] Qian Wang, Ning Jia, and Toby P Breckon. A baseline for multi-label image classification using an ensemble of deep convolutional neural networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 644–648. IEEE, 2019.

[27] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[28] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016.

[29] Christine Zeh and Simon Babovic. Lego recognition tool, March 2018. https://robo4you.at/publications/Lego.pdf.

[30] Feng Zhu, Hongsheng Li, Wanli Ouyang, Nenghai Yu, and Xiaogang Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.