How does imbalanced data affect performance of regression CNNs?

Ratish Thakoersingh[1]
Supervisors: Yuko Kato[1], Tom Viering[1]
Responsible Professors: David Tax[1], Marco Loog[1]
[1]Delft University of Technology

Abstract

This research provides an overview on how training Convolutional Neural Networks (CNNs) on imbalanced datasets affect the performance of the CNNs. Datasets could be imbalanced as a result of several reasons. There are for example naturally less samples of rare diseases. Since the network is trained less on those instances, this might lead to worse performance on those cases. However, it might be more crucial to identify those cases properly. Furthermore, it is non-trivial to check whether real-time generated data is balanced. The networks in this research are trained on three different types of synthetic datasets. Balanced datasets, datasets with missing targets and datasets that have normally distributed targets. The task of the network is to find the standard deviation of the pixel intensity of the input. The results show that it is best to train the network on balanced datasets, however training networks on datasets with normally distributed targets does not result in a big loss. Furthermore, in this case the CNNs were still able to learn the task with decent performance if the training set missed targets.

1 Introduction

For the last decade machine learning has been becoming more and more popular. This is due to the vast advancement of state-of-the-art performance on several fundamental computer tasks as a result of machine learning. There are various machine learning approaches, and deep neural networks are widely used machine learning techniques [1][2]. Deep learning algorithms are based on the human brain structure, imitating how it perceives and handles data. Convolutional Neural Network (CNN) is a spatial invariant class of deep neural networks and it has been proven to be a successful state-of-the-art algorithm for image recognition [3]. Although CNNs are popular, they are still widely regarded as black boxes. Treating these networks as black boxes can lead to unexpected and possibly unfavorable behavior. There are still various aspects of these networks that are not well understood, e.g. how training on different datasets affect the performance of these networks.

Real-life datasets could be imbalanced because of various practical reasons. For instance when training a network for disease checking, there are naturally less samples of rare diseases available. This might lead to worse performance on those cases, as the network is less trained on those instances. Nonetheless, it could be more vital to properly identify those cases [4]. Furthermore, it is not uncommon to use real-time generated data in machine learning. It is complex to find out whether real-time data is balanced. The imbalanced problem has been studied mostly in the context of classification tasks. Studies show that training CNNs on imbalanced datasets result in bad performance for classification tasks [5]. There are multiple ways to reduce the negative effects of imbalanced datasets, e.g., over/down-sampling, feature selection, cost sensitive learning [6] [4] and cross validation [7]. While these examples refer to the effect of imbalanced datasets on CNNs, they solely focus on image classification tasks. There are studies about what data imbalance in regression entails [8] and how to make those datasets suitable for training [9][10][11]. However these studies are very generalized, and not directed towards data imbalance for CNNs with a specific regression task.

This paper aims to understand the influence of imbalanced training datasets on the performance of CNNs. The CNNs are constructed to perform a regression task, meaning that they take images as input and return continuous values as outputs. In this research the regression task the networks have to perform is finding the standard deviation of the pixel intensities of the input image. Synthetic datasets with controlled distinct target distributions were used to train, validate and test the network. The datasets are either balanced, missing targets or have normally distributed targets. A baseline was used to put the performance of the networks in context. By gaining a better understanding of how and why the distributions of targets impact the performance of a CNN, the importance of the target distribution is established.

From the experiments can be derived that the networks trained on balanced datasets have the best performance. While the networks trained on datasets with normally distributed targets had a lower performance, they still performed quite well. The networks trained on datasets with missing targets were able to perform decently, however the performance was inferior to that of the other networks.

The organization of this paper is as follows. Section 2

elaborates on the research question and the relevance of it. Furthermore, it introduces a hypothesis of the research question. In section 3 a detailed desciption of the method is given. It includes a description of the convolutional neural network and datasets considered in this research. Additionally, an overview is provided regarding the training method and the performance metrics. The experiments and their results are shown in section 4. These results are then discussed and compared to the results of related work in section 5. A reflection of reproducability of this research is given in section 6. Finally, the research is concluded in section 7.

2 Imbalanced Regression Problems

The research question of this paper is as follows: "How do imbalanced training datasets affect the performance of CNNs?". The networks were trained to predict the standard deviation of the input images. An imbalanced dataset is defined as a dataset that has either a non-uniform distribution of the target domain, or poorly presents part of the range compared to the test dataset [9].

2.1 Subquestions

In order to achieve an answer to the research question, it has been divided into subquestions.
The first question answers how a network trained on a balanced dataset performs on a balanced dataset. The results of this question can be used to put the performance of networks trained on imbalanced datasets into perspective.
How well a network trained on a dataset with missing targets performs is researched in the second question.
The third research question pursues to answer how a network trained on a dataset with a normal target distribution performs. The dataset with normally distributed targets is imbalanced as the test dataset has a uniform target distribution.
With the results of these sub-questions, the main research question was answered.

2.2 Hypothesis

By looking at the performance of networks trained on imbalanced datasets for classification tasks, it can be expected that networks trained on balanced datasets generally perform significantly better than networks trained on imbalanced datasets [5]. Studies about regression networks trained on imbalanced datasets further support this claim [9][10][8][11]. It is uncertain whether this claim holds for this research, as these studies used different imbalanced datasets and networks, furthermore the tasks the networks had to perform in those studies is also different.

3 Methods

In this section detailed descriptions of the methods are given. In subsection 3.1 an overview of the convolutional neural network is presented. The datasets considered in this research, with information on how they are synthesized is outlined in subsection 3.3. Additionally, in subsection 3.2 a discussion is provided regarding the hyperparameters. Finally, the optimizer and the performance metrics are presented in subsection 3.4.

3.1 Convolutional Neural Networks

Implemented Network

The network used in this research has been constructed based on an article[12] using Python. The base network was then further adjusted to make it suitable for the regression task and images that are used in this paper. A visualisation of the network can be found in Figure 1. In the figure can be seen that the network has two 2D convolutional layers, both have a filter size of 5. The convolutional layers are followed by two fully-connected/linear layers. The activation functions are rectified linear units (ReLUs), these were chosen as ReLU has high speed and accuracy [13]. Furthermore, two dropout layers are used for regularization [14]. The network also has two average-based pooling layers to reduce the computational load. It was decided to keep the network fairly shallow, as shallow networks have been proven to give a good performance [15], furthermore this will also keep the research less convoluted. The code of the network can be found in the GitLab repository [16].

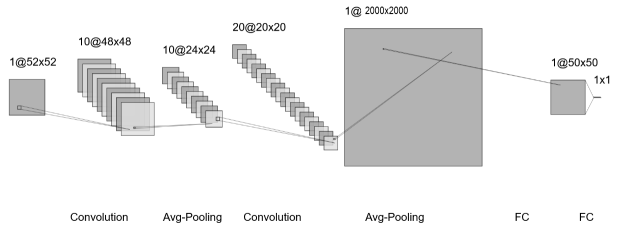

Figure 1: Visualisation of the used network with the format (depth)@(width)x(height)

Network task

The task of the network is to find the standard deviation of the pixel intensity of the input. Therefore the targets of the datasamples are their standard deviations. The formula for standard deviation is:

$$std = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(pixel_i - mean)^2}$$

N = Size of the image
$pixel_i$ = Intensity of pixel i of the image
$mean$ = The mean of the pixel intensity of the image

3.2 Hyper parameters

During the training of the networks several hyper parameters needed to be configured. One of these parameters is the batch size. For a good trade-off between train speed and fluctuation sensitivity of the network, a batch size of 50 was chosen.
Another hyper parameter is the number of epochs. The number of epochs has been chosen to be 15, as the validation loss seemed to converge before that. Furthermore, by increasing the epochs the validation loss occasionally diverged.
The learning rate is another hyper parameter that can be

tweaked. During tweaking with the validation sets, a learning rate of $5 \cdot 10^{-7}$ was observed to be the most effective value. By using a larger learning rate, the model learned faster, at the cost of converging on a sub-optimal validation error. By using a smaller learning rate than the chosen learning rate the training error finalised at a lower value, however the validation error was higher. This is an indication of overfitting. A significantly lower learning rate caused the loss never to converge. An important remark is that the hyperparameters were tweaked concurrently, as they are interdependent.

3.3 Datasets

In this research three different target distributions are considered. The used datasets are all synthetic and created for this specific research. There has been chosen for synthetic datasets as for those, it is easier to reason about the results. In addition, by using synthetic datasets it was easier to make small alterations and change parameters to satisfy the needs of the research. As mentioned in the introduction the datasets are tailored for the standard deviation predicting task. Therefore, the images in the datasets are distinguishable by their standard deviation. Furthermore, the datasets can be identified by the range and distribution of the standard deviations (note that these are also the targets) of the samples in the datasets. Every dataset has a training, validation and test set. For the training sets, a sample size of 50,000 was chosen. The validation set and test sets all have a sample size of 5,000.

Samples

For every dataset, the samples have been drawn from an $N(0, std^2)$ distribution. This std (standard deviation) is generated from a distribution dependant on the dataset the sample is in. Here the assumption is made that the mean of the images do not affect the relative performances of the trained networks as all the images are drawn from distributions with zero means. Therefore we can identify the datasets solely on their distribution of targets. The images are all 52 by 52 pixels. Visual representations of a few samples are shown in Figure 2. The most important thing to note from this figure is the variation in the images which correlates to the standard deviation of the image.

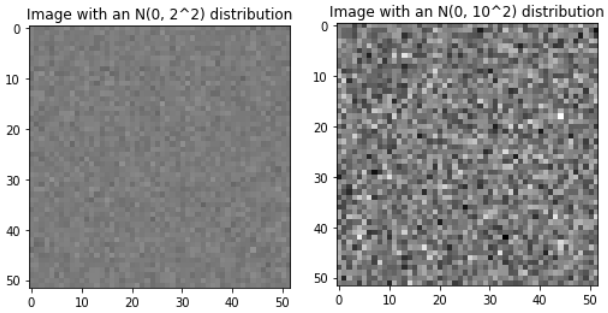

Figure 2: Visualisation of $N(0, 2^2)$ and $N(0, 10^2)$ data samples

U(1, 10) dataset

The first dataset has a $U(1, 10)$ target distribution, this implies that the targets of all the images in the dataset are continuous values between 1 and 10, uniformly distributed. A visualisation of the target distribution can be found in Figure 3. This dataset is used as the balanced and reference dataset as the targets of the images are equally distributed in the full range that is considered.

U(1, 5) dataset

The second dataset has a $U(1, 5)$ target distribution, this can be seen in Figure 3. The targets are, similar to the targets of the reference dataset, uniformly distributed and continuous. However, the range of targets in this dataset is from 1 to 5, while the reference dataset (the dataset with a $U(1, 10)$ target distribution) has targets between 1 and 10. Therefore, this dataset is imbalanced as it is missing targets.

N(5.5, 1) dataset

The last dataset has a normal target distribution, with a 5.5 mean and 1 as standard deviation $\rightarrow N(5.5, 1)$. A visualisation of this dataset can be found in Figure 3. Since the dataset has normally distributed targets, while the reference dataset has uniformly distributed targets, this dataset is imbalanced.

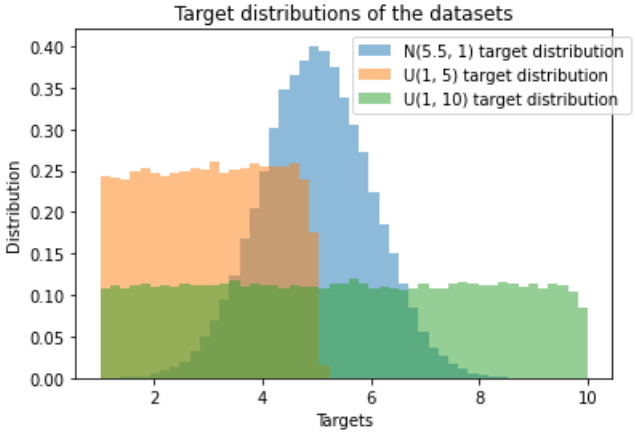

Figure 3: A graph of the target distributions per dataset

3.4 Performance

Optimizer

The optimizer updates the network properly based on the training results. This is done by adjusting the internal network parameters in response to the loss function output. The Adam optimizer was chosen, as it generally has a good performance [17] [18].

Loss Function

There are several metrics that can be used to measure the performance of a CNN, this is generally calculated with a loss function. The loss function is a function that maps the expected and actual outputs of a network onto a number that represents the loss associated with the task. To optimize the performance, the loss needs to be minimized. During training the optimizer uses the outcome of the loss function to adjust

the network in a favourable manner, therefore it is important to select the proper loss function. The network from the article uses the negative log likelihood, however according to studies MSE performs better and is the most common loss function for regression tasks [19]. The formula for MSE is:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(predicted_i - actual_i)^2$$

N = Size of the dataset
i = Sample of the dataset
$predicted_i$ = The value predicted by the network of sample i
$actual_i$ = The target of sample i

Furthermore, the percentage loss is used to account for the difference in scales between the datasets. The formula for percentage loss is:

$$percLoss = \frac{1}{N}\sum_{i=1}^{N}(\frac{predicted_i - actual_i}{actual_i}) \times 100\%$$

(see the legend of MSE for variable explanations)

Baseline

The reference datasets have uniformly distributed target in the range [1, 10). Therefore the baseline randomly guesses a continuous number between 1 and 10. The baseline is used to compare the performances of the trained networks. Networks that perform better than the baseline are considered as having a good performance.

4 Results

Every experiment has been done 10 times for generalization and reduction of the influence of random and measurement errors. The first three experiments examine the performance on the reference dataset (dataset with the $U(1, 10)$ target distribution) per network. Then, the aggregated results of the first three experiments are shown to compare the performances of the networks on the reference dataset. Lastly, the performances of every network on all the target distributions are shown. From this can be concluded which test distribution gives the best performance generally.

4.1 Performance of networks trained on balanced datasets

To answer this question, networks were trained on datasets with a $U(1, 10)$ target distribution, and then tested on datasets with $U(1, 10)$ distributions. Since the train and test sets both have the same target distributions, this is seen as balanced. In Figure 4 and Table 1 can be seen that the network performs significantly better than the baseline, as the MSE and percentage loss are lower.

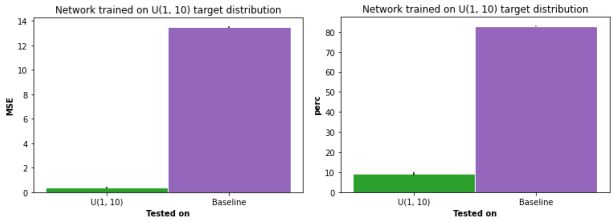

Figure 4: The performances of networks trained on datasets with a $U(1, 10)$ target distribution, tested on $U(1, 10)$ target distributions

Table 1: The performances of networks trained on datasets with a $U(1, 10)$ target distribution, tested on $U(1, 10)$ target distributions

	MSE	percentage
U(1, 10)	0.375 ± 0.057	9.175 ± 0.978
Baseline	13.495 ± 0.08	82.899 ± 0.351

4.2 Performance of networks trained on datasets with missing targets

For this question, the networks were trained on datasets with a $U(1, 5)$ target distribution. The networks were then tested on datasets with a $U(1, 10)$ target distribution. The results of this were compared to the baseline and the performance of the networks tested on datasets with a $U(1, 5)$ target distribution. From Figure 5 can be derived that the network outperforms the baseline by a large margin. Furthermore, the network performs better on datasets with the same target distribution as it was trained on. We can see that in Table 2, as the test performance on the $U(1, 5)$ datasets is higher than on the $U(1, 10)$ datasets.

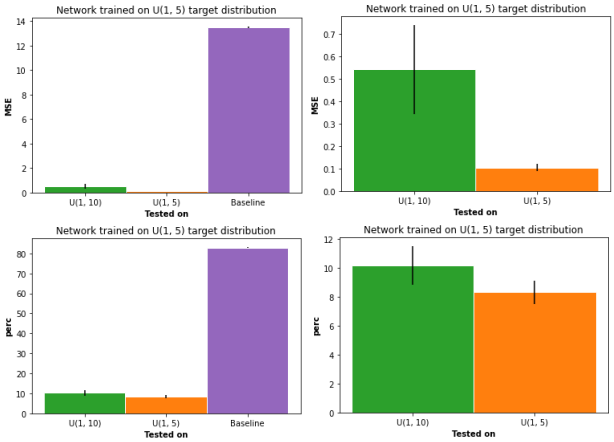

Figure 5: The performances of networks trained on datasets with a $U(1, 5)$ target distribution

Table 2: The performances of networks trained on datasets with a $U(1, 5)$ target distribution

	MSE	percentage
U(1, 10)	0.541 ± 0.199	8.626 ± 2.248
U(1, 5)	0.104 ± 0.016	9.175 ± 0.978
Baseline	13.495 ± 0.08	82.899 ± 0.351

4.3 Performance of networks trained on normally distributed targets and tested on datasets with uniformly distributed targets

The networks that were used for answering this question were trained on datasets with a $N(5.5, 1)$ target distribution. These networks were then tested on datasets with a $U(1, 10)$ target distribution. To put the performance into perspective, the results also contain the performance of the networks on datasets with a $N(5.5, 1)$ target distribution and the performance of the baseline. The networks performed better than the baseline, as can be seen in Figure 6. By only looking at the MSE one would conclude that the networks are performing better on the datasets with the $N(5.5, 1)$ target distribution. However since the percentage loss is seemingly equal, the difference in MSE mostly comes from imagines in the $U(1, 10)$ distribution having higher pixel intensities.

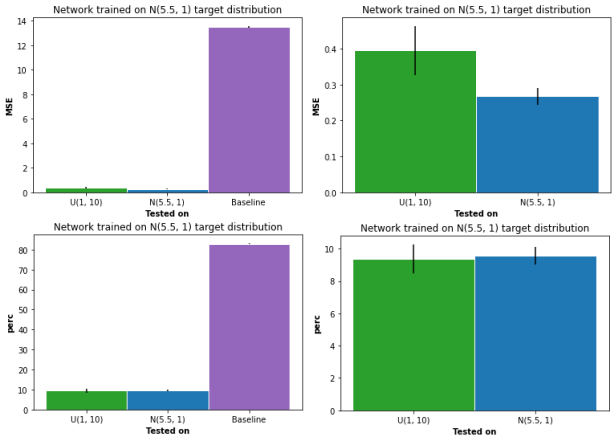

Figure 6: The performances of networks trained on datasets with a $N(5.5, 1)$ target distribution

Table 3: The performances of networks trained on datasets with a $N(5.5, 1)$ target distribution

	MSE	percentage
U(1, 10)	0.395 ± 0.069	9.371 ± 0.901
N(5.5, 1)	0.268 ± 0.023	9.567 ± 0.54
Baseline	13.495 ± 0.08	82.899 ± 0.351

4.4 Comparison of the performance of the networks trained on different target distributions

This question is divided into two experiments. For the first experiment the networks have been trained on all the target distributions and tested on the reference datasets. To compare the performances of the networks more easily, the results have then been aggregated per training distribution in Figure 7 and Table 4. From this can be seen that all the networks perform relatively well compared to the baseline. The networks trained on the datasets with the $U(1, 10)$ distribution gave the best performance. This was expected, as the test and train datasets had the same target distributions. The second best performing network group was the group trained on datasets with the $N(5.5, 1)$ target distribution. The worst performing network group was the group trained on datasets with the $U(1, 5)$ target distribution.

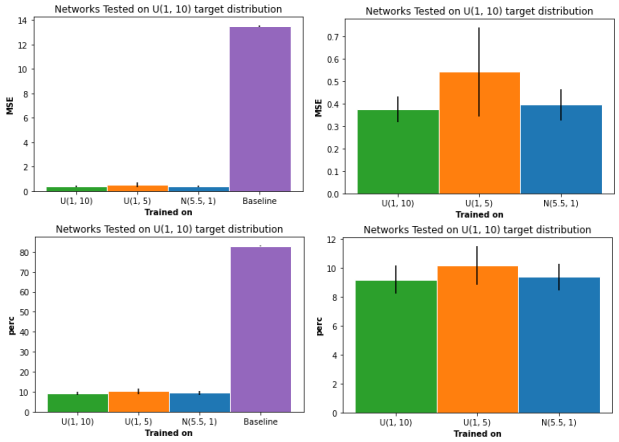

Figure 7: The performances of networks tested on datasets with a $U(1, 10)$ target distribution

Table 4: The performances of networks tested on datasets with a $U(1, 10)$ target distribution

	MSE	percentage
U(1, 10)	0.375 ± 0.057	9.175 ± 0.978
U(1, 5)	0.541 ± 0.199	10.189 ± 1.331
N(5.5, 1)	0.395 ± 0.069	9.371 ± 0.901
Baseline	13.495 ± 0.08	82.899 ± 0.351

To get an idea of how the networks perform on all target distributions that are considered in this research, the networks have been tested on all the target distributions. From these results, the average per train distribution has then been taken. The results of this can be found in Figure 8 and Table 5. Similarly to the previous experiment, the networks trained on the $U(1, 10)$ target distribution performed the best, followed by the networks trained on the $N(5.5, 1)$ target distribution. The networks trained on the $U(1, 5)$ target distribution had the

worst performance. The standard deviation of the results is higher than in the previous experiment, as multiple test distributions were used.

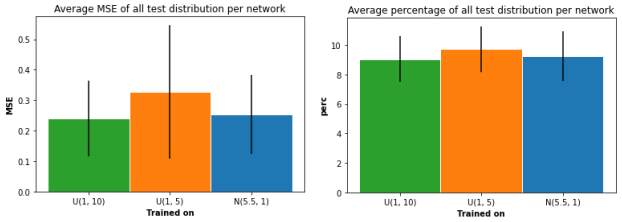

Figure 8: The performances of every network group tested on all target distributions

Table 5: The performances of every network group tested on all target distributions

	MSE	percentage
U(1, 10)	0.24 ± 0.124	9.043 ± 1.543
U(1, 5)	0.328 ± 0.219	9.729 ± 1.549
N(5.5, 1)	0.253 ± 0.13	9.242 ± 1.692

5 Discussion

The experiment in subsection 4.1 showed that the networks trained on datasets with a $U(1, 10)$ target distribution had a significantly higher performance than the baseline. From this can be concluded that the networks performs well if they are trained on balanced datasets.
The networks trained on the datasets with a $U(1, 5)$ target distribution also outperformed the baseline by a great margin, as the results from subsection 4.2 established. This implies that networks trained on the datasets with missing had a relatively good performance. Furthermore, the network performs better on datasets with the same target distribution as it was trained on. This is logical, as for those datasets it does not have to predict targets that were not in the scope of the training set.
From the experiment in subsection 4.3 can be derived that the networks trained on datasets with normally distributed targets had a good performance on the reference datasets. This conclusion was made as those networks had a much higher performance than the baseline.
The hypothesis in subsection 2.2 theorized that the networks trained on balanced datasets would perform better than the datasets trained on imbalanced datasets. From the the results in subsection 4.4 can be seen that this is indeed the case, as the networks trained on the balanced datasets had a better performance. Furthermore, the experiments showed that the networks trained on normally distributed datasets performed the second best. The reasoning for this can be that some of the targets were underrepresented, therefore the network was not able to predict those as effectively. The networks trained on datasets with missing targets had the worst performance. As those training sets did not include all of the targets of the test sets, the networks were unable to perform as well as the networks from the other groups. In addition, the results showed that the networks trained on the dataset with uniformly distributed targets covering the full target space had the highest overall performance on all of the test target distributions.
The findings of this research can be put into perspective by comparing it to the results of similar research. According to studies on classification tasks, CNNs trained on datasets with balanced distributions perform significantly better than CNNs trained on imbalanced datasets [5]. This is in line with the results of this research, as the networks trained on the complete uniform range always performed better than the other networks. The experiments showed that this is also the case in the setting of this research, however the differences between the performances were not as significant. The reasoning for this can be that for regression tasks the network not only learns about the target of a given training sample, but also about other close targets. For classification, this is not the case as labels are not continuous and do not have an order. Studies involving imbalanced datasets with regression, but not CNNs suggest that networks trained on imbalanced datasets have a lower performance [9][10][8][11]. This is in line with the results of the experiments.

6 Responsible Research

Many steps were taken to make this research and the experiments reproducable. The code can be found in on the github repository [16]. Although many datasets have been used for the experiments, these have not been included in the repository. Instead, the code which created these dataset has been included. The algorithm creates semi-random datasets with defined parameters, the used parameters can be found in section 3. Since this research is about the distribution type of targets and not about specific datasets, the research is general enough to be reproducible with these randomly generated datasets as long as the same parameters are used. Furthermore, the code for the experiments has been provided, included with the parameters that were used for this research. There was no involvement of sensitive data in this research, as all the data was generated from scratch.

7 Conclusions and Future Work

In this paper research was conducted on how imbalanced datasets affect the performance of a regression CNNs using synthetic datasets. This is done by comparing the performances of a baseline and networks trained on datasets that are balanced, have missing targets and have normally distributed targets. The networks are tailored for finding the standard deviation of the pixel intensity of the input. From the results can be seen that all the networks significantly outperformed the baseline. The networks trained on the balanced datasets had the best performances, followed by the networks trained on the datasets with normal targets. The networks trained on the datasets with the missing targets performed the worst. Therefore can be concluded that it is better to train the network on balanced datasets, however training networks on datasets with normally distributed targets does not result in a big loss. There is still plenty of room for future work. One could adjust how many of the targets are missing in the missing targets datasets, different amounts of missing targets could lead to

different outcomes. Furthermore, only a normal target distribution with a standard deviation of 1 was used, adjusting the standard deviation could result in different behaviour. There can also be conducted research about distributions that were not included in this paper, for example an inverse triangle distribution. The task considered in this research was finding the standard deviation, using a different task like finding the median could result in different findings. Lastly as the dropout layers provides more generalization for a network, it could be compelling to check whether that influences the flexibility of networks trained on a specific distribution.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[2] Ida Bagus Krishna Yoga Utama, Akhmad Faqih, and Benyamin Kusumoputro. Three mixture of odor classification using convolutional neural network. In *2019 16th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering*, pages 1–4, 2019.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–11052, 2012.

[4] Aida Ali, Siti Mariyam Shamsuddin, and Anca L Ralescu. Classification with class imbalance problem. *Int. J. Advance Soft Compu. Appl*, 5(3), 2013.

[5] Paulina Hensman and David Masko. The impact of imbalanced training data for convolutional neural networks. *Degree Project in Computer Science, KTH Royal Institute of Technology*, 2015.

[6] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] Claudia Beleites, Richard Baumgartner, Christopher Bowman, Ray Somorjai, Gerald Steiner, Reiner Salzer, and Michael G Sowa. Variance reduction in estimating classification error using sparse datasets. *Chemometrics and intelligent laboratory systems*, 79(1-2):91–100, 2005.

[8] Yuzhe Yang, Kaiwen Zha, Ying-Cong Chen, Hao Wang, and Dina Katabi. Delving into deep imbalanced regression. *CoRR*, abs/2102.09554, 2021.

[9] Paula Branco, Luís Torgo, and Rita Ribeiro. Smogn: a pre-processing approach for imbalanced regression. 09 2017.

[10] Paula Branco, Luís Torgo, and Rita Ribeiro. Rebagg: Resampled bagging for imbalanced regression. 09 2018.

[11] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[12] Gregor Koehler. Mnist handwritten digit recognition in pytorch, Feb 2020.

[13] Maria Pavlova. Comparison of activation functions in convolution neural network. In *2020 28th National Conference with International Participation (TELECOM)*, pages 65–67, 2020.

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[15] Daniel E. Kim and Mikhail Gofman. Comparison of shallow and deep neural networks for network intrusion detection. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 204–208, 2018.

[16] Ratish Thakoersingh. Gitlab repository with the code. https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-52/rp-group-52-rthakoersingh/-/tree/dev.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[18] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[19] Arash Sangari and William Sethares. Convergence analysis of two loss functions in soft-max regression. *IEEE Transactions on Signal Processing*, 64(5):1280–1288, 2016.