Is the batch size affecting the performance of Regression CNNs ?

LAMON Julien[1] , **KATO Yuko**[1] , **TURAN Taylan**[1] , **VIERING Tom**[1] , **WANG Ziqi**[1] , **LOOG Marco**[1] , **TAX David**[1]

[1]Delft University of Technology

J.A.D.Lamon@student.tudelft.nl,{Y.Kato, O.T.Turan, T.J.Viering, Z.Wang-8, M.Loog, D.M.J.Tax}@tudelft.nl

Abstract

With an expectation of 8.3 trillion photos stored in 2021 [1], convolutional neural networks (CNN) are beginning to be preeminent in the field of image recognition. However, with this deep neural network (DNN) still being seen as a black box, it is hard to fully employ its capabilities. A need to tune hyperparameters is needed to have a robust CNN that can more accurately do its task. In this study, the batch size, being one of the most important hyperparameters, is our main concern. Moreover, we show how sensitive CNNs are to the following regression tasks: the mean, median, standard deviation (std) and variance. A sensitivity analysis is conducted by analyzing how well regression CNNs converge, giving different batch sizes and a fixed learning rate. Additionally, the sensitiveness is analyzed by comparing the final mean squared error given by all different batch sizes. At the end of the research, our findings concluded that ...

to be continued

Keywords— Deep Learning, Convolutional Neural Network, Regression, Sensitivity Analysis, Batch Size

1 Introduction

A Convolutional Neural Network (CNN) is a "deep learning neural network designed for processing structured arrays of data such as images"[2]. The breakthrough of CNNs [3] was mostly seen in the field of image recognition. Its benefit and innovation come from the fact that it can efficiently detect features without any human supervision. Whilst CNNs have first been discussed in the 1980s [2], the arrival of autonomous cars and much more makes it the preeminence in its favourable field. However, while this method is very helpful, it is yet treated as a black box: studying its structure will not give any judgment on the structure being approximated. Although some scientists such as Zeiler and Fergus [4] came up with solutions to visualize the work of a CNN, various sections of it are still not fully understood. Furthermore, image recognition is a subject that can be treated in two different ways: classification and regression tasks. Recent interest has been growing in the classification alternative, such as Ye Zhang and Byron Wallace [5] where they evaluate the sensitivity of a CNN to its "input vector representations; filter region size(s); the number of feature maps; the activation functions; the pooling strategy; and regularization terms". However, until now, there have been only a few amounts of studies on the sensitivity of CNNs in a regression task [6].

As demonstrated by Satya Mallick and Sunita Nayak [7], the number of parameters a CNN contains is vast: as an example, the first successful CNN architecture that proved its usefulness - AlexNet [8] - contains over 60 million parameters in total. In consequence, a significant amount of research is being produced around the optimization of hyperparameters [9]. However, before diving into the optimization, there is a need to first improve our know-how of the network's sensitivity to specific hyperparameters. The aim of this work is therefore to get a more profound understanding of how sensitive CNNs are to the batch size, as well as why they are sensitive to them. More precisely, the analysis will be made over CNNs training on a regression task, namely: the mean, standard deviation and median. All experiments will be conducted on a baseline model, with the assumption that all hyperparameters - except for the batch size - do not have any impact on the model's performance.

The paper is structured as follows. To first have a broader understanding of the problem, section 2 will be charged of that. That is, it gives a better definition of a convolutional neural network, as well as sensitivity analysis. Then, in section 4, a detailed description of the methods is given. This encircles the dataset used with the preprocessing made on it, with a high end description of the baseline model chosen, followed by the training and performance evaluation. In section 5, the experiments that are done as well as their results is described. The responsible research paragraph is given in section 6. A discussion on our results compared to the results of related works is in section 7. Finally, section 8 concludes the paper and also give use some ideas of future work.

2 Problem Description

This section develops to a greater extent the problem we are trying to answer, namely, why is a regression CNN sensitive (or not) to different batch sizes. To fully understand the design and context there is first a need to have a more in-depth explanation of a Convolutional Neural Network. Then, the section ends with a detailed description of what is a sensitivity analysis.

2.1 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a class of Deep Learning Methods. While an Artificial Neural Network (ANN) would need to flatten the data – being an image in our research context – and feed it in a network of the same size as the vector, a CNN can learn, and extract patterns present in an image by running a filter on an image. This gives the advantage of it being able to autonomously detect features without the need for human supervision. Moreover,

CNNs are composed of multiple blocks, namely: convolution layers, pooling and fully connected layers. More generally, the model can be divided into two distinctive parts: feature extraction and the mapping of these extracted features to final output, being in our case the mean, standard deviation, or median.

Feature Extraction

This is the major part of the model. It is made out of a sequence of one or more convolutional and pooling layers.

The role of a convolutional layer is to extract features of a picture by the use of a kernel. That is, this kernel – also known as a filter – is being dragged on the picture, left to right, top to bottom, and a matrix multiplication between the kernel and each portion of the scanned image is made. The output matrix is then summed to obtain the output value at the position of the filter. The result of all output given by the kernel is called a feature map (as seen in figure 1). That feature map is then passed through an activation function, such as the ReLu, tanh, etc.

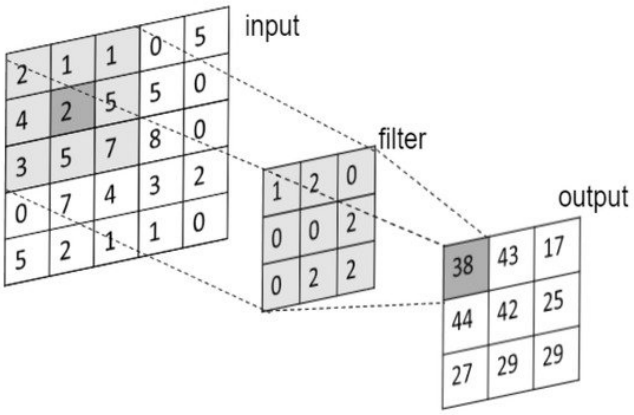


Figure 1: Outline of the Convolutional Layer [10]

Pooling layers are used to reduce the special size given by the convolutional layer. This way, the number of learnable parameters is reduced. A vast amount of different pooling layers exists, but the common pooling layers are the average and max pooling layers. The latter is used in our research project due to its noise suppression ability. The max pooling function, as the convolutional layer, will "drag" a window (i.e. a filter/kernel) through the feature map returned by the ReLu function, and return the biggest digit in that window, as shown in figure 2.

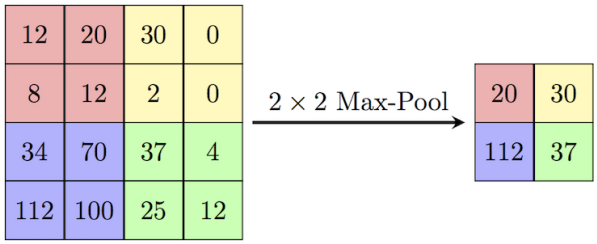


Figure 2: Pictoral representation of a Max Pooling layer [11]

Mapping features to an output

At the end of the aforementioned step, the output result of the last convolution or pooling layer is flattened to be fed to the Fully Connected Layer(s). The fully connected (FC) layer is a layer where ingoing/outgoing edges are equal to the number of neurons on the previous/next layer. The last layer then maps this flattened vector to final outputs, which can be the probabilities of the input image belonging to a specific label, or its computed mean.

2.2 Sensitivity Analysis

In order to effectively answer our reqearch question, there is first a need to understand what is sensitivity, and most importantly how can it be measured. As said explained in by Maosen Cao, Nizar F. Alkayem, Lixia Pan and Drahomir Novak (October 19th 2016), "the key point behind sensitivity analysis is that by slightly varying each explicative input parameter and registering the response in the output, the explicative parameters with the highest sensitivity values gain the greatest importance"[12].

To be continued... (one piece)

3 Related Work

TODO

4 Methods

Following a more profound description of the research problem, there is now a need to get a more detailed description of the method. The section first starts with a description of the dataset used during the research, as well as the pre-processing conducted before feeding it to the network. Then, as a second subsection, a more detailed description of the model is given, with explanations on why such a custom model was created. In section 4.4, details on the training is given. That is, details on parameters that is kept constant for the whole research process. And finally, to end the section, an explanation of how the network is evaluated is explained.

4.1 Experimental Environment

In order to be able to fully replicate all experiments, there is a need to detail the experimental environment. Then, it should be noted that all experiments have been conducted on Windows 10 installed on a machine with an Intel Core i5-10400F 2.90GHz and 16GB RAM. In order to significantly improve the speed of training, a single RTX2060 GPU is used in our experiments, with the use of CUDA.

Considering the implementation of the CNN, the whole network is created using the python package named PyTorch. PyTorch is one of the strongest Deep Learning libraries, with Tensorflow on its side. With its support for C, C++ and Tensor computing, it permits to have a much faster model while still being easy to use [13]. All experiments are publicly available in the following GitHub repository: github.com/Jlamon/sensitivity_batch_size_regression_cnn.

4.2 Dataset

All experiments will be operated on the MNIST dataset [14], being a collection of images representing handwritten digits. This specific dataset contains 60,000 training images, with an additional 10,000 images for the test set. All images have already been preprocessed by these researchers [14], resulting in size-normalized and centred 28x28 images.

Since the project is focussing on the computation of regression tasks, labels given with the MNIST dataset are of no help to us. Therefore, a need to preprocess the dataset by ourselves was needed. Therefore, to approximate the mean, median, or standard deviation, these values were computed for all images.

4.3 Baseline Model

A baseline model was implemented following a tutorial called "MNIST Handwritten Digit Recognition in PyTorch"[15]. A modification in the code was needed to fit our needs.

The model is constructed out of two sequential layers and then a set of fully connected layers (FC), as seen in figure 3. The two sequential layers are similar; that is, they both have one 2D Convolutional Layer with a kernel size of 5x5, following by a 2D max-pooling layer with a 2x2 sized kernel, and then a ReLu activation function. The only difference between the two sequential layers are the input and output channels: the first layer has one input and 10 output channels while the second layer has 10 input and 20 output channels. The first input channel is of size one because the MNIST dataset is grayscaled. Finally, a set of fully-connected layers is created. This set is comprised of two FC: the first contains 320 nodes and is connected to the second layer, which has 50 nodes and finished with only one, being the mean or std or, lastly, the median. It should be noted that right after the two sequential layers, the output is first flattened in order to be fed to the set of fully connected layer.

It should be noted that no dropout layer is being added to the model since it is being deactivated while the model is evaluating.

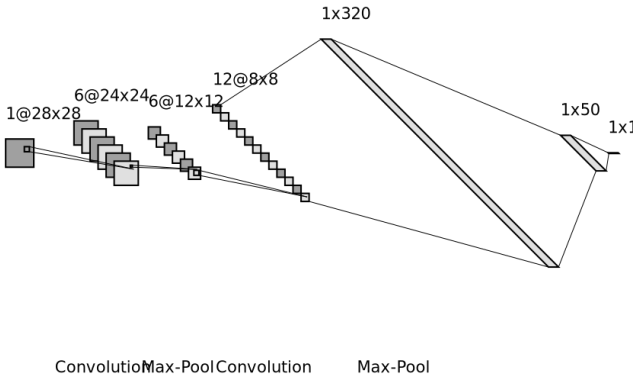


Figure 3: Outline of CNN model

4.4 Training

With the help of PyTorch, discussed in the earlier "Experimental Environment" section, most of the hyperparameters are kept as default. In other words, the number of hyperparameters being set is kept to a minimum to concentrate the most on the research question, and not the fine tune of the model. As an example, the 2D Convolutional Layer class provided by PyTorch [16] requires the following parameters:

torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')

However, we are only modifying the first three parameters, the rest is kept as default, as seen in the class above. This also applies to the pooling layers, fully connected layers, ReLu activation function, and the optimizer used.

Concerning the optimizer, the choice was being considered between the Stochastic Gradient Descent (SGD) and the Adaptive Moment Estimation (Adam). To make up our mind on which optimizer to choose, we considered the results of each of them while computing the mean of images in the MNIST dataset, as seen in fig

ADD FIGURE

. It can therefore be seen that the Adam optimizer is achieving a lower Mean Squared Error loss, compared to the SGD. We, therefore, chose to continue with the Adam optimizer.

It should be noted that no validation set has been used in the training of the model. The reason is that the use of a validation set is to evaluate different models on it to choose the best performing one. However, due to our research question a baseline model is needed to have its performance differing only due to the parameters manually changed. Therefore, since the model is decided beforehand, a validation set is not needed.

4.5 Performance Evaluation

While testing the accuracy in classification tasks can only be evaluated by checking if the predicted label corresponds to the targeted label, regression tasks need a loss function. As the no free lunch theorem states: there is no single best optimization algorithm [17]. The choice to use either the Mean Squared Error (MSE) or Mean Absolute Error (MAE) loss function is then being judged in figure 4. This figure indicates that the MSE performs better than the MAE for our model. This can be supported by the fact that the MAE at the last epoch is greater than the MSE results by a factor of two. It can also be indicated that there are fewer fluctuations in the MSE curve.

The MSE is the commonly used regression function, being the sum of squared distances between the targeted and predicted values: $\frac{1}{n}\sum_{t=1}^{n} e_t^2$. On the other side, the MAE is the sum of absolute differences between the targeted and predicted values: $\frac{1}{n}\sum_{t=1}^{n} |e_t|$.

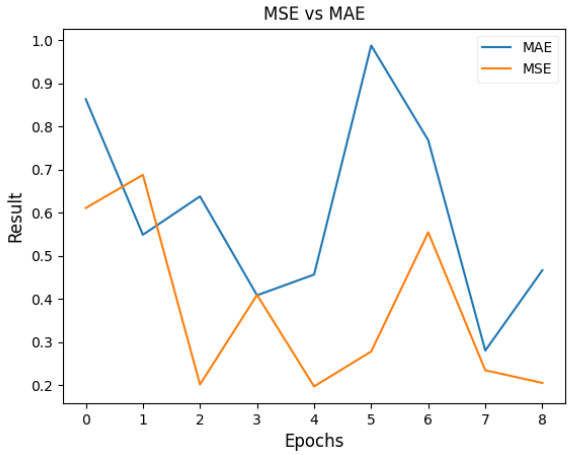


Figure 4: Differences between MSE and MAE

5 Results

This is yet a WIP: do not take into the grammar, tone, etc.

In the interest of answering the research question, we executed several experiments. It is in this section that we present the multiple evaluations performed to determine how sensitive are regression CNNs to the batch size. All results given in this section are created using the methods described in section 4.

For each regression task (mean, median, std and variance), the learning rate of the baseline model is tuned considering a batch size of 32. The learning rates first tested are on the log-scale curve:

[0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1]

Then, considering the best result found on the log-scale, values around it are tested as well. As an example, if the model trains better with an initial learning rate of 0.01, these following values are then tested:

$$[0.0025, 0.005, 0.0075, 0.01, 0.015, 0.0175, 0.02]$$

As in the paper of Pavlo M. Radiuk [18], the different batch sizes that will be tested are all on the power of 2:

$$[16, 32, 64, 128, 256, 512, 1024]$$

For each one of them, the model is trained from scratch and the accuracy is tested at the end of each epoch, as well as before any prior training.

In the end, a total amount of four experiments will be performed, with each of them being conducted on the four regression tasks. For each experiment, a different set of graphs is given to compare the results. For each experiment and batch size, the baseline model is trained for 10 epochs. A higher number of epochs would result in the model overfitting.

5.1 Experiment 1: How well does they converge ?

In this experiment, graphs a created to see if there are any bad fluctuations in the test loss. We are doing these experiments because the sensitivity can be observed if there are significant differences in the convergence of the model's loss between different batch size. That specific convergence is then represented in graphs, where the curve represents the mean squared error (MSE) loss being computed on the test set, and at the end of each epoch.

As seen in figure 5, we can already draw some conclusions, we can see that the convergence in the mean and variance is quite normal, but for the standard deviation, we see bigger fluctuations for the high batch sizes, with a relatively normal convergence for the smaller ones. The difference between the worst MSE result and the best one is 1300. This is a difference to be analyzed.

Then, concerning the median, we can see that, again, the higher batch size perform worse than the smaller ones. Conclusions can be drawn from the distribution of the synthetic datasets, as seen in appendix A. But this will all be discussed in the discussion section.

5.2 Experiment 2: The time it takes to converge

The graph below is not the final graph

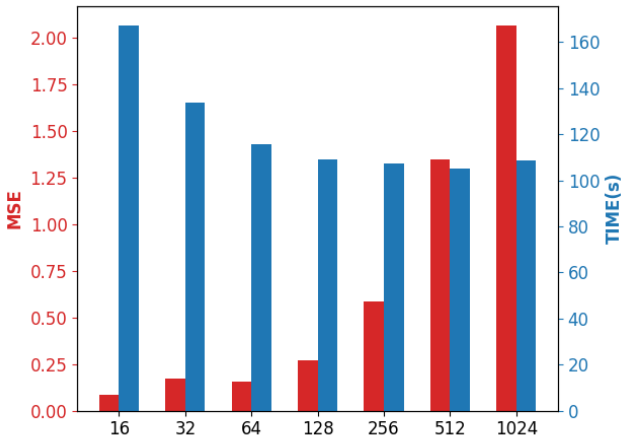


Figure 6: Barchart with results from the mean regression task. The MSE is represented in red while the time it takes to train the model is represented in blue. The time is represented in seconds.

5.3 Experiment 3: Sensitivity of CNN to different batch sizes

The third experiment, being the most important one, represents the last MSE loss registered for each different batch size. It can be seen that two graphs (figure 7 and 8) are being used, with the variance being separated from the other regression tasks (the Mean, Standard Deviation and Median). The reason behind is the great difference in the MSE results, which makes it impossible to analyze the differences.

As a first look, it can be seen that the MSE loss gradually increases just as an exponential curve, except for the median. Again, the median might behave this way do to its distribution, as seen in appendix A.

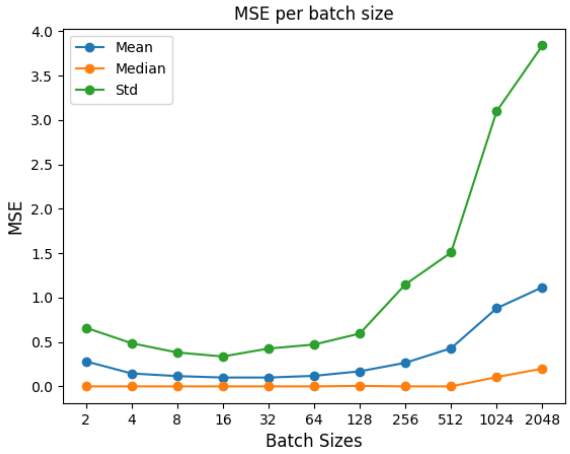


Figure 7: Sensitivities

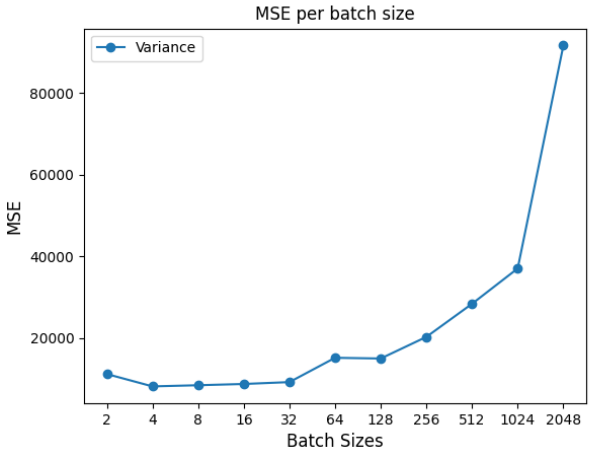
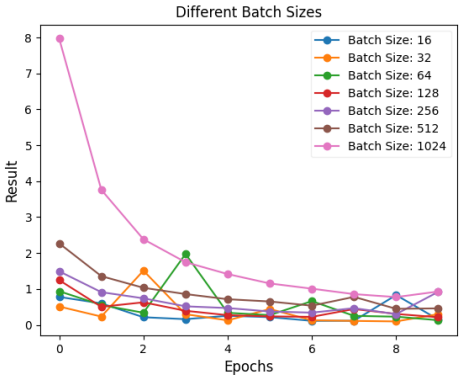


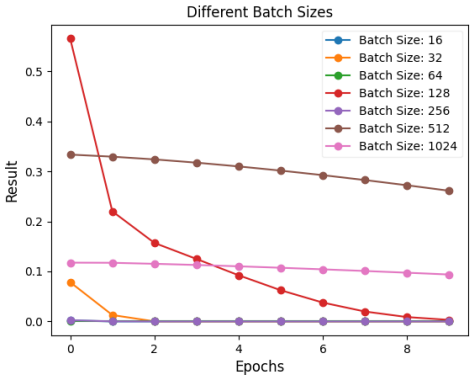
Figure 8: Sensitivity of Variance

5.4 Experiment 4: Error difference in 10 different runs

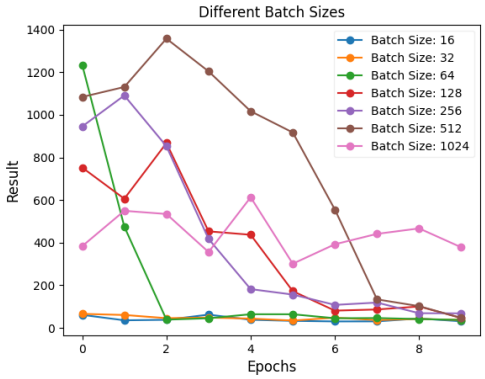
Sensitivity can also be measured by how different results are on multiple runs. That is, the graphs seen in figure 9 are representing the maximum, minimum and mean results given by the last epoch of



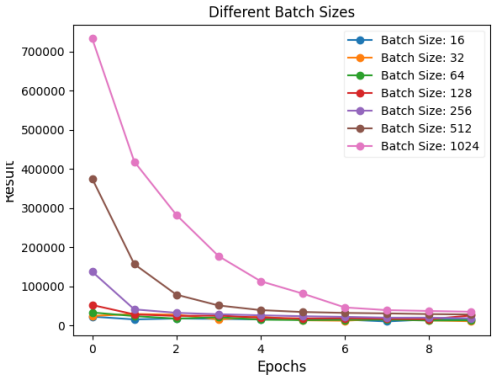
(a) Convergence Mean



(b) Convergence Median



(c) Convergence Standard Deviation



(d) Convergence Variance

Figure 5: Different Convergences

the model. Moreover, the baseline model is being trained ten times from scratch, and only the MSE of the last layer is kept. Then, the maximum, minimum and mean of each batch size is computed and represented on the graph. Therefore the x-axis represents the different batch sizes (with their size gradually increasing), and the y-axis represents the MSE loss. It should be noted that for each run, a new random seed is being created. The digit used to set the random seed is computed as follows: current batch size + current run index.

From the graphs, it can be seen that there are similarities to the graphs in the first experiments, such as the high MSE for a batch size 1024 in the median, etc. But, also, we can see that the difference between the high MSE loss and the mean is higher than the difference between the minimum MSE and the mean. This can then be used to show some sensitivities to specific batch sizes. The understanding of why this is happening will however be discussed in the upcoming discussion section.

6 Discussion

6.1 Adam vs. SGD

TODO

6.2 Observed Results

From the above graphs, it can be concluded that the higher the batch size:

- The higher the Mean Squared Error
- The less time it takes to train
- The slower it converges

Some subjective results can also be observed, such as:

- We can clearly see the fluctuations in the convergence of models in the std regression task.
- Some models computing the median does not even converge.

6.3 Why do the smaller batch sizes perform better ?

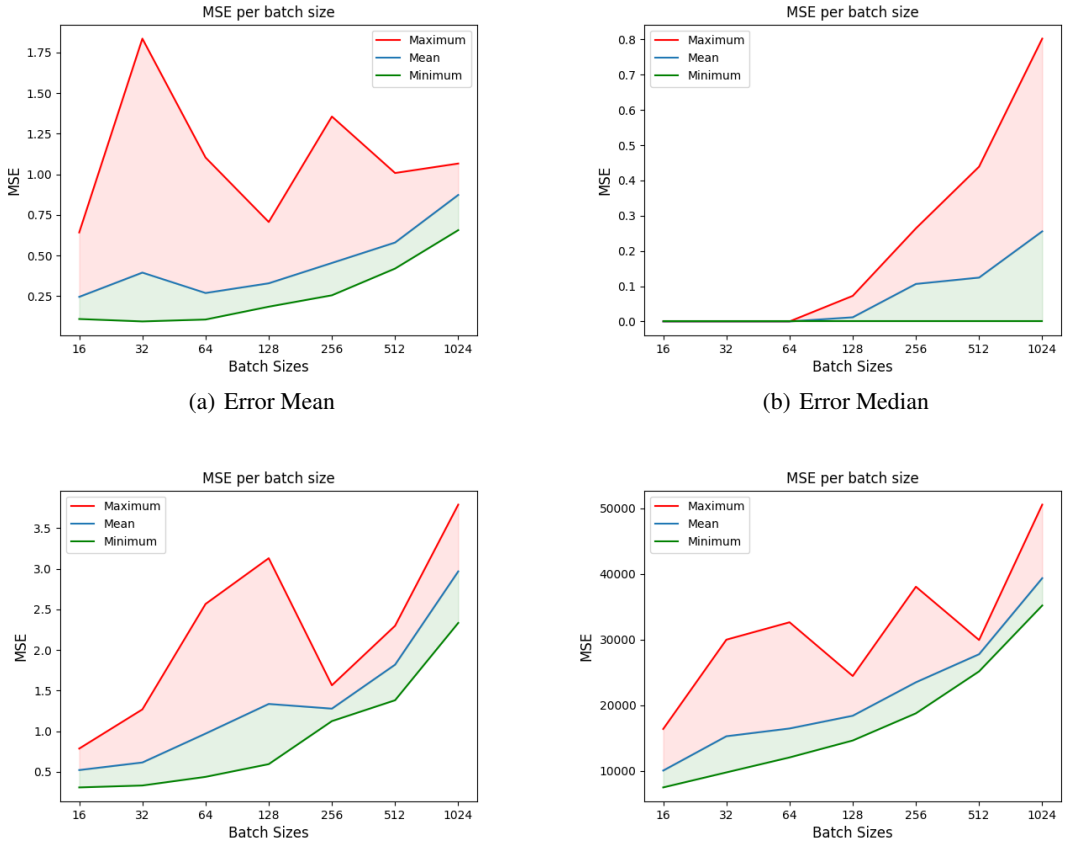
An extended discussion between the related papers and the results found in this research will be given.

6.4 Limitations

TODO

7 Responsible Research

In addition to the time taken to represent and discuss the results found during the research, an appropriate amount of time was taken to represent the truth to its fullest, and to cite the work and code established by others. That is, all information is given to prove that



(a) Error Mean

(b) Error Median

(c) Error Standard Deviation

(d) Error Variance

Figure 9: Different Errors

results and representations can be reproducible on any other machine if the conditions are met.

Concerning the code, it was already noted that the code was taken from a blog, namely "MNIST Handwritten Digit Recognition in PyTorch" [15]. It was then changed, with the help of Julian Biesheuvel, Remco den Heijer and Ratish Thakoersingh, the teammates with whom I indirectly worked during the whole research project. As the baseline was created, each teammate forked from it, and work on his specific research question. It is then that the code, followed with its experiments, plots and saved states of the model, can be publicly found on a GitHub repository [source], all sorted by different weeks. Additionally, a profound explanation of experiments, with the hyperparameters used and why can be found in the previous sections 4.

Concerning the main python package, PyTorch, with its strong community behind it, makes the source code easy to learn and understand. This advantage, therefore, enables anybody to reproduce the results if they have a powerful enough machine.

With all the conditions met, this is research can be fully reproduced with similar results. A minor difference in the results can be found if another GPU is used, with mostly a change in the time taken to train the model. Also, due to rounding errors, minor differences can be found in the MSE/MAE results, but yet having a global similarity to the results shown in this research paper. Finally, some results might also change due to the random seeds.

Furthermore, the experiment does not involve human subjects, thus unethical human research cannot occur.

8 Conclusions and Future Work

TODO

References

[1] David Carrington. How many photos will be taken in 2021?, Mar 2021.

[2] Thomas Wood. Convolutional neural network, May 2019.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdelrahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[4] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Computer Vision – ECCV 2014 Lecture Notes in Computer Science*, page 818–833, 2014.

[5] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification, 2016.

[6] Stephane Lathuiliere, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. A comprehensive analysis of deep regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9):2065–2081, 2020.

[7] Satya Mallick and Sunita Nayak. Number of parameters and tensor sizes in a convolutional neural network (cnn): Learn opencv, Apr 2021.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[9] Erik Bochinski, Tobias Senst, and Thomas Sikora. Hyperparameter optimization for convolutional neural network committees based on evolutionary algorithms. 09 2017.

[10] Saifullahi Aminu Bello, Shangshu Yu, Cheng Wang, Jibril Muhmmad Adam, and Jonathan Li. Review: Deep learning on 3d point clouds. *Remote Sensing*, 12(11), 2020.

[11] Max-pooling / pooling.

[12] Maosen Cao, Nizar F. Alkayem, Lixia Pan, and Drahomír Novak. Advanced methods in neural networks-based sensitivity analysis with their applications in civil engineering. *Artificial Neural Networks - Models and Applications*, 2016.

[13] Stephen Welch. Pytorch vs. tensorflow: What you need to know, May 2020.

[14] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

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

[16] Conv2.

[17] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.

[18] Pavlo M. Radiuk. Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Information Technology and Management Science*, 20, 1 2018.

A Boxplot Distribution

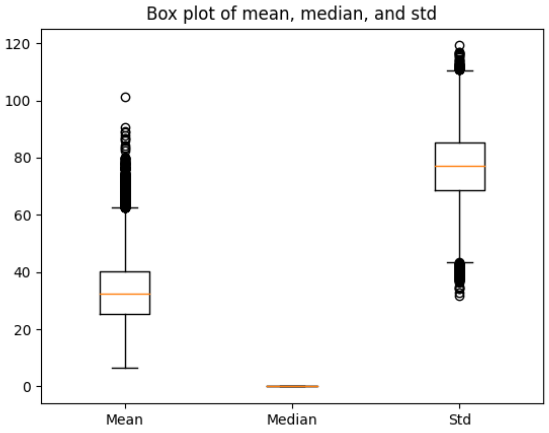


Figure 10: Distribution of Mean, Median and Standard Deviation

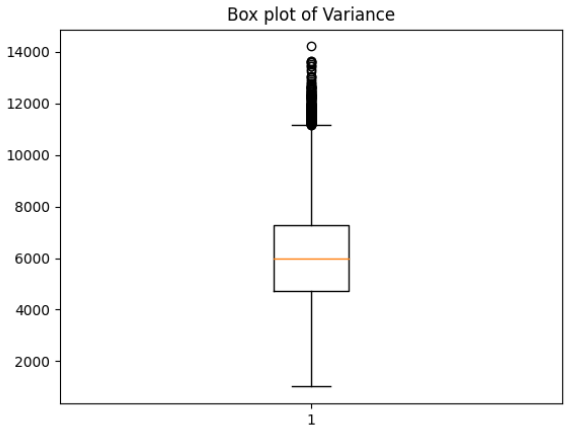
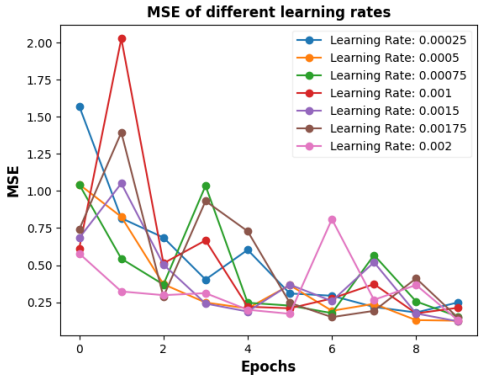
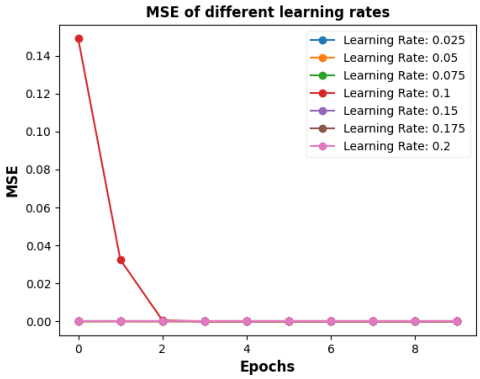


Figure 11: Distribution of Variance

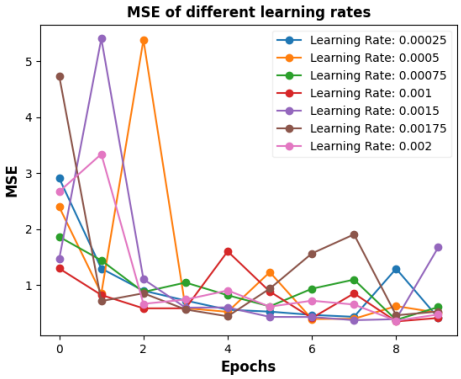
B Learning rate results



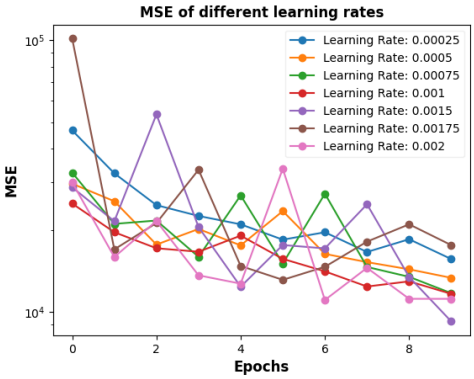
(a) Learning Rates Mean



(b) Learning Rates Mean



(c) Learning Rates Median



(d) Learning Rates Variance

Figure 12: Learning rates per different regressions