Masking your problems away

Showing the effect of adding a masking layer on out of distribution accuracy

Quinten Nouwens[1]

Supervisor: Wendelin Böhmer[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 25, 2026

Name of the student: Quinten Nouwens
Final project course: CSE3000 Research Project
Thesis committee: Wendelin Böhmer, David Tax

Abstract

Spurious correlation can be detrimental to performance of machine learning solutions on data that it has never seen before. This could be disastrous in situations where the prediction is important and the situation is ever changing. This paper investigates whether adding a masking layer can improve the accuracy of machine learning models on visual data, where the background is different during training and testing. A new dataset is created that overlays MNIST numbers on CIFAR-10 images, where during training only a distinct subset of background images for every label is presented to the model. The performance is measured by supplying the model with an unseen foreground sample on a background that was not encountered during training. This paper then shows that adding a masking layer can improve the performance of a model. This improvement depends on the implementation and the dataset on which the model is trained and tested.

1 Introduction

Machine learning is used for all kinds of disciplines. It is used in robotics, medicine, computer vision and many more [1]. Machine learning can recognise patterns in data by training on training data. It can use this learned pattern recognition to recognise similar patterns in never before seen data.

Machine Learning solutions are currently still black boxes. This means it is impossible to tell how they made their decisions. They are also prone to overfitting [8] and spurious correlation [7]. Both of these result in the fact that the machine learning solution performs well on data similar to the data seen during training. However, when they encounter never before seen data they perform worse. Because it is impossible to tell how the machine learning models made there decisions, it is often hard to detect whether they were influenced by spurious correlations. Spurious correlations have been found in data ranging from celebrity gender classification [7] to classifying whether a skin cancer is benign or not [6].

Since machine learning solutions are deployed in situations where an incorrect result could be disastrous, correctness is important. They are deployed in real life situations. Real life situations are ever changing meaning that data that has not been encountered in training data will occur. Therefore, being able to adapt to these changes is important. The ability to classify never before seen data correctly is called generalization.

A masking layer has been promising in generalizing complex visual tasks with distractions [2]. This masking layer added a small percentage of parameters to the model and significantly increased performance. This masking layer consists of several convolution layers that return a value between 0 and 1 for every pixel and multiplies this output with the original image to scale the pixel value. The masking layer does not require its own loss function and is trained together with the rest of the machine learning solution.

This paper evaluates whether a masking layer is also suitable for a stronger correlation between distractions and desired patterns. It also investigates whether multiplying the mask with the original image is the best way to apply the mask. This is done because multiplying the mask can only bring the values closer to 0. It is therefore investigated whether bringing the values closer to 1 or masking towards a random value is better.

The main research question of this research is: How do masking solutions influence the performance of a machine learning solution on out of distribution test data? To answer this research question several subquestions are answered:

- How does scaling the image towards a random value influence the performance on out of distribution test data?
- Does a masking layer improve out of distribution accuracy?
- Does the way in which the mask is applied change the accuracy?

These questions are answered by first establishing a baseline which trains the LeNet architecture on a dataset that contains a strong spurious correlation. Then noise is added during training, after which several masking layer solution are tested and evaluated.

2 Dataset Description

In this paper, MNIST digits [5] are classified on top of CIFAR-10 images [4]. MNIST is a 28x28x1 grayscale image dataset that contains handwritten digits. CIFAR-10 is a dataset containing 32x32x3 RGB images showing 10 classes of images, such as aeroplanes and deer. They both contain integer values between 0 and 255.

For every class of MNIST, so for every digit from 0 to 9, a certain number of random CIFAR images are selected. These sets of images are distinct from each other. The size of these sets is called the class set size. During training, the machine learning solution sees MNIST samples overlaid on just the CIFAR images from these sets.

To overlay the MNIST sample on top of CIFAR images several operations are performed. Firstly, the MNIST sample is padded on all sides with zeros to make every image the same size as the CIFAR sample (28x28x1 $\rightarrow$ 32x32x1). Then the pixel value of the MNIST sample is repeated over the three color channels (32x32x1 $\rightarrow$ 32x32x3). Both the MNIST and CIFAR sample are now scaled to the range 0-1 so divided by 255. Now the two are combined using Equation 1. Where M_{value} is the value of the MNIST sample and C_{value} is the pixel value of the CIFAR sample. The $target_{color}$ specifies the color that the MNIST is when overlaid on the CIFAR sample. Unless otherwise specified this target color is black (0, 0, 0). An example image with different target colors can be seen in Figure 1

$$pixel_{value} = (1 - M_{value}) * C_{value} + M_{value} * target_{color} \quad (1)$$

Testing can be done both in distribution (id) and out of distribution (ood). For in distribution testing MNIST samples

Figure 1: Examples images contained within the dataset. This shows the same CIFAR image with the same MNIST sample, but a different target color.

unseen during training are overlaid on CIFAR images drawn from the subsets used during training for the corresponding MNIST class. For out of distribution testing MNIST samples are overlaid on CIFAR images, where both the MNIST samples and the CIFAR images are unseen during training. So the CIFAR images do not get chosen from any of the class sets.

The accuracy of the machine learning solution is determined by the ability to classify the MNIST digit. There is an expected difference between in distribution and out of distribution accuracy for small class sizes. When the class size is 1 every MNIST sample from a certain class is overlaid on the same background for training. For the machine learning algorithm learning this background is easier than learning the digit on top of it and will always be accurate. So when encountering a totally new background, therefore an out of distribution test, it is almost a random selection between MNIST classes and so the accuracy is expected to be low. For in distribution testing the accuracy is close to the training accuracy since it has the same background as the training data.

For a large amount of backgrounds this discrepancy is less. This is because when every sample has a different background, the background is no longer viable data for classifying the MNIST class. Of course there is always a risk of overfitting to the data, but this is no longer due to the background.

Therefore, there is a gap between in distribution testing and out of distribution testing and this gap goes away for higher class sizes. Decreasing the size of this gap is the aim of this paper.

3 Architectures and Noise Procedure

3.1 Baseline

The baseline compares the accuracy on in distribution data and the accuracy on out of distribution data for different class sizes without using masks or noise. The LeNet-5 architecture is chosen for the baseline [5]. The architecture is shown in Table 1. This architecture is chosen because of its high accuracy on MNIST samples. It is also relatively small, which means that it should not be able to learn a large amount of background images. This is used to show whether or not the created solutions in this paper have an effect on decreasing the discrepancy between the accuracy on in distribution and out of distribution data.

Baseline
Convolution (5x5) 6 channels
Pool 2x2 kernel 2 stride
Convolution (5x5) 16 channels
Pool 2x2 kernel 2 stride
120 fully connected neurons
84 fully connected neurons
10 fully connected neurons

Table 1: Architecture of the baseline: LeNet-5

3.2 Noise

The effect of noise on the ability to generalise is investigated. This noise will be uniform and will affect a certain percentage of pixels. This is done to see whether the masker performs better than random noise and whether random noise can be used to enhance the masker. It uses the same model as the one that was used for generating the baseline.

The noise is applied only during training and is generated just before the data is passed through the model. The noise is also different every epoch. This is done because this way the model cannot rely on the noise at all. The percentage of pixels affected and the intensity of the noise is varied.

How the noise is applied is shown in Equation 2. Where original is the original pixel value, aon is the amount of noise or intensity, and noise is a random pixel. Whether a pixel is affected is determined by the percentage affected value. From this equation, it can be seen that if the amount of noise is equal to 1, the output does not depend on the original pixel. This means that when the amount of noise is 1, this method is equivalent to dropout [9].

$$pixel = \begin{cases} original * (1 - aon) + aon * noise & \textit{if affected} \\ original & else \end{cases} \tag{2}$$

3.3 Masking

The masking layer consists of two convolution layers both with 6 out channels with ReLu activations in between. After this a final convolution layer is used to bring it down to one channel only and a sigmoid activation is used to generate a value between 0 and 1 for every pixel in the original image. These values are multiplied with the original image. This scaled image is then supplied to the rest of the model, which is the same model as used for the baseline. This full process is illustrated in Figure 2.

There are several ways in which the masker is applied to the image. Firstly the masker described above, where the mask is multiplied with the image. Therefore, the output is between 0 and the original pixel value. This is called the black masker, because it lowers the pixel value of all the pixels and hence darkens the image. Secondly, a masker where the output value lies between the original pixel value and 1. This is called the white masker, since it brightens all pixels. This masker does not simply multiply the mask together with the original image, it uses Equation 3. Thirdly a masker is considered, where a constant is added to the original pixel values and the output of the masker therefore is between 0 and

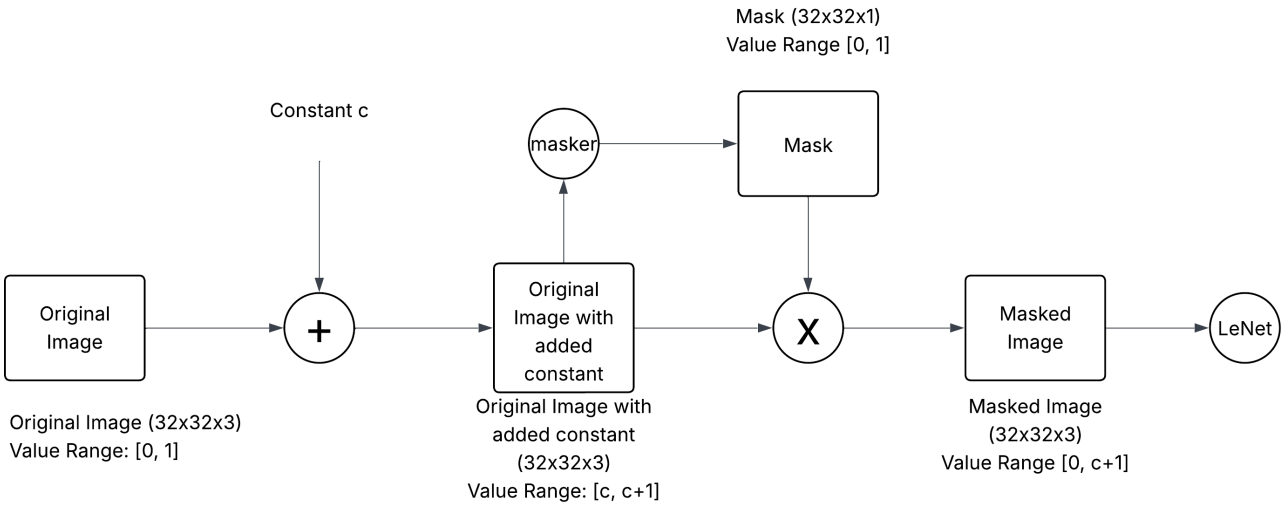


Figure 2: Flowchart showing the masking classification process. The value range of the masked image can be different if the constant is negative

Masker
Convolution (3x3) 6 channels with padding
Convolution (3x3) 6 channels with padding
Convolution (3x3) 1 channel with padding

Table 2: Masker architecture

the original pixel value plus the constant. The first masker is equivalent to the third if the constant is 0. These are named according to their constant. The architecture of the masker is found in Table 2.

$$pixel = original + (1 - original) * mask \quad (3)$$

This masking layer is implemented to see whether such a layer is able to differentiate between the background and foreground and therefore be able to isolate the foreground and improve out of distribution accuracy. This layer is not trained separately, but together with the rest of the machine learning solution and no additional loss function is used.

The different constants are added to see whether that makes a difference. The mask outputs a value between 0 and 1 and multiplies this with the image, which darkens it. If the foreground is black and there is no constant added, this would mean that the masker can only make the background more similar to the foreground. However if a constant is added, the masker could make the background stand out more.

The effect of the masking layers on the difference of the accuracies is investigated. It is interesting to see whether only this masking layer has an effect on the accuracy at all. This is because the masking layer does not destroy information, which for example a dropout layer would do. The dropout layer removes the information from a node entirely. The masking layer only changes the intensity of the information. The machine learning solution could change its weights, such that there is no change in the overall outcome.

4 Experimental Setup and Results

All the shown results are from models trained and tested on the same 10 datasets. They are all trained using the same hyperparameters, same optimizer and same loss function. These are shown in Table 3. The datasets are generated randomly using randomly chosen seeds. This is done to ensure a fair comparison between different methods. The accuracy graphs contain a non-transparent line showing the mean accuracy and unless otherwise specified the semi-transparent area around this line is the area that is within one standard deviation from the mean.

Parameter	Value
Loss function	Cross Entropy Loss
Optimiser	Adam
learning rate	10^{-3}
Adam β_1, β_2	0.9, 0.999
Scheduler	MultiStepLR
Scheduler gamma	0.1
Scheduler milestones	5 epochs and 7 epochs
batch size	64
number of epochs	10

Table 3: Hyperparamers

4.1 Baseline

The baseline accuracy can be seen in Figure 3. It shows that training and in distribution accuracy are almost 100% for all class sizes and closely aligned. The out of distribution accuracy is low for small class sizes. For a class size of 1, it is 15%. This is slightly above random, which would be an accuracy of 10%. So even when the machine learning solution can classify the digit based on the background only, it still performs slightly better than random on unseen backgrounds. For high class sizes, the out of distribution accuracy

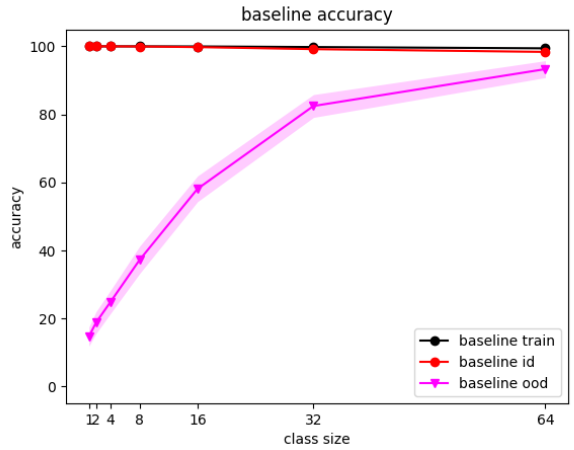


Figure 3: Baseline accuracy, showing the training accuracy in black, the in distribution accuracy (IDD) in red and the out of distribution (OOD) accuracy in magenta

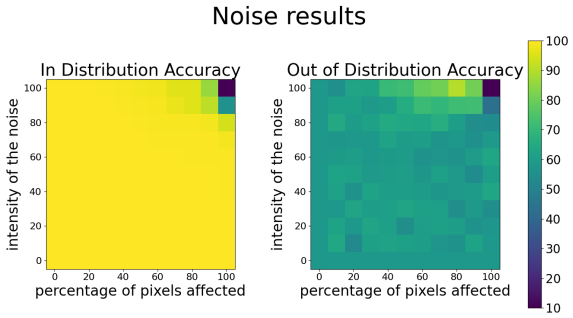


Figure 4: 2D grids showing the in distribution accuracy and out of distribution accuracy for models, which have noise induced during training at varying percentage effected and noise intensity. These results are obtained with a class size of 16

gets closer to the in distribution accuracy, with the gap almost disappearing at a class size of 64.

4.2 Noise

Figure 4 shows two grids that show the average in distribution accuracy and out of distribution accuracy for different percentage affected and intensity of the noise values. These values were obtained by training 5 models on different datasets all with a class size of 16 with the target color being black. The percentage affected and the intensity of noise ranged from 10% until and including 100% with 10% increments. The value for 0% percentage affected or 0% intensity is set to the mean baseline accuracy for the same class size.

It shows that most values are similar for the entire domain, except when the intensity of the noise is 100%. The in distribution accuracy is nearly 100% for the enitre domain, except when both the intensity and the percentage of noise reach 100% at which point the entire image would be noise during training. The out of distribution accuracy stays within one standard deviation of the baseline for most of the domain.

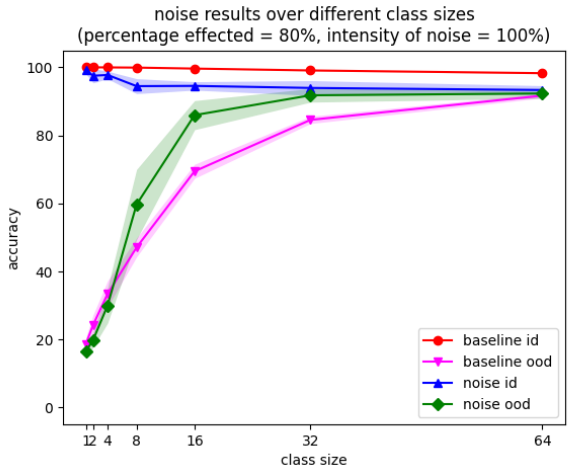


Figure 5: Accuracy graph of models with a percentage affected of 80% and a noise intensity of 100%

However, when the intensity of the noise reaches 100%, the mean accuracy increases. This means that the out of distribution accuracy only increases when no part of the original signal is going through in the affected pixels. This would be similar to dropout [9], which is less computationally complex.

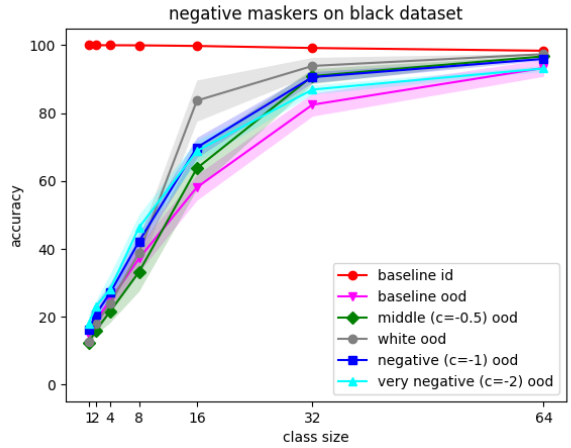
Figure 5 shows the accuracy graph when 80% of the pixels are complete noise. It shows the same mean as Figure 4 for n=16. It also shows that these results are significant. The out of distribution accuracy with the noise is higher than the baseline. It also shows that the in distribution accuracy converges with the out of distribution accuracy for high class sizes.

Because the noise shows very little difference over the full domain, it was decided to not incorporate noise into the masking solution.

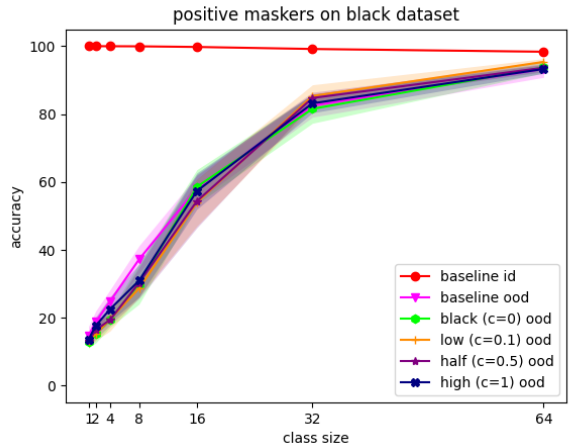
4.3 Masker

Figure 6a and Figure 6b show the results of several maskers. Figure 6a shows maskers with a negative constant and the white masker. For these, it can be seen that they outperform the baseline after n=8. Figure 6b show maskers with non-negative constant and shows that they do not differ from the baseline accuracy at all. It also shows that the maskers almost reach the same accuracy on out of distribution tests as in distribution tests.

Figure 7 shows masks generated by the masking layers of several models. The models were trained on a dataset, which class sizes are shown above the masks. The added constant is shown to the left of the masks. The image is out of distribution for all of them. It shows that for class sizes until 4 the masks do not do much at all. Most of the values are close to each other and it highlights random artifacts. From class sizes of 8 the models with the nonzero constants clearly show the foreground number in the mask. The masker without any added constant, however, shows fully yellow masks. This indicates that the masks are close to 1 for the whole image. Which means that the original image is not altered at all. This gives an indication why the black (c=0) masker does not



(a) Accuracy of several maskers with a negative constant and the white masker



(b) Accuracy of several maskers all with a non negative constant

Figure 6: Accuracy of maskers with negative and non negative constants on MNIST samples with a black color

improve over the baseline.

The difference between the white masker and the negative masker is surprising. When the white masker is applied to the image, a value between the original pixel value and 1 will be returned, where 1 represents the value of a white pixel. The negative masker first adds a constant of negative 1 to the original image and then multiplies this with a value between 0 and 1. This means that this masker also returns a value between the original pixel value and the value associated with white in this case 0. This should result in similar performance.

None of the maskers with a non negative constant outperform the baseline, while the ones with a negative constant do. This might be due to the fact that the foreground color of the MNIST digit is black. This would mean that when the color would be white, it would be the other way around.

The maskers add only a small amount of parameters to the network. Models with a mask have 62.56k parameters and models without a mask have 62.01k parameters. So the masker increases the amount of parameters by 0.9%.

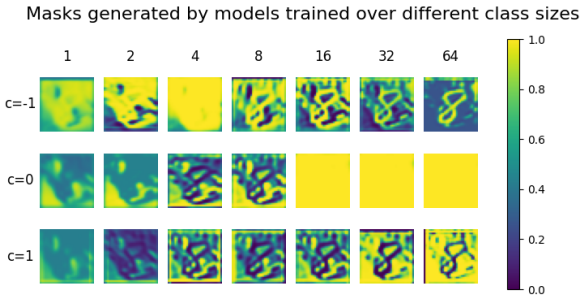


Figure 7: Masks generated by models trained with different class sizes and a different constant. The original image is the same and out of distribution for all of them

4.4 Different color datasets

Figure 8a and Figure 8b show the accuracy results when models are trained on the same datasets as previous results, but with the target color set to white (1, 1, 1). The baseline is also trained and evaluated on this white dataset.

The models trained on the white dataset show the opposite behavior to the ones trained on the black dataset. The maskers with a negative added constant are close to the baseline, while maskers with a positive constant show improved performance compared to the baseline. The only constant that shows improvement for both of them is -0.5.

Figure 9a and Figure 9b show the accuracy results with the target color set to magenta (0.99, 0.24, 0.71). This color was chosen because it does not lie at the extremes of the color spectrum, which white and black do. Besides this it appears less in the cifar dataset and is therefore more distinct than black and white.

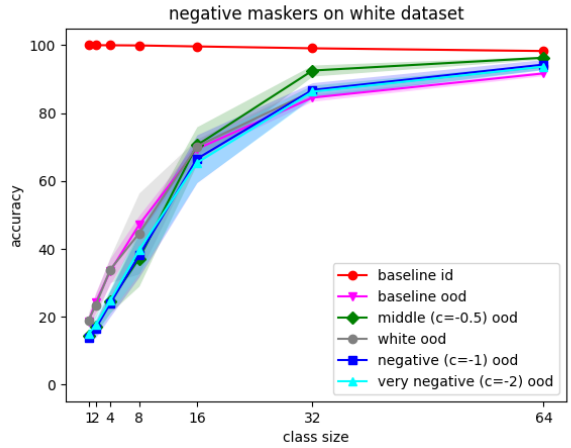
The models generalize sooner on the magenta dataset than on the other colors. Whereas the baseline only has about 80% out of distribution accuracy for the black and white dataset at a class size of 32, the models achieve close to in distribution accuracy for the magneta dataset at that class size.

Because the models generalise faster, significant results are harder to obtain. There is simply less space to improve. The only significant result is the high masker (c=1), which performs better than the baseline. The means for every masker are higher than the baseline, except for the middle masker.

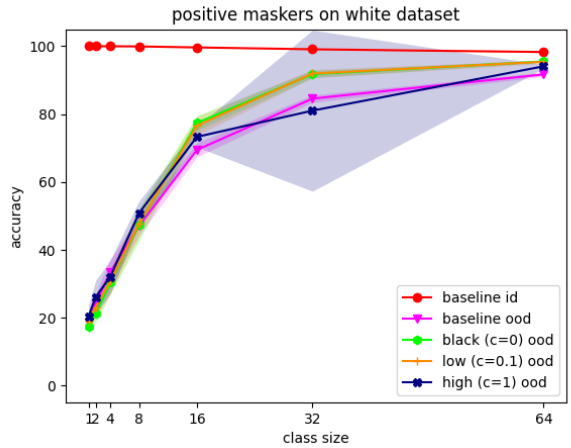
5 Responsible Research

This paper has been written with reproducability in mind. Because the dataset is not widely known the complete creation process has been highlighted. The two datasets that make up the new dataset are widely known and therefore not explained in detail. The specific implementation of the maskers and LeNet are not included, but there should be enough information to implement a similar model. The results directly come from that implementation and should therefore be reproducible.

All results obtained during this research with the same hyperparameter are included in the paper. In the beginning of

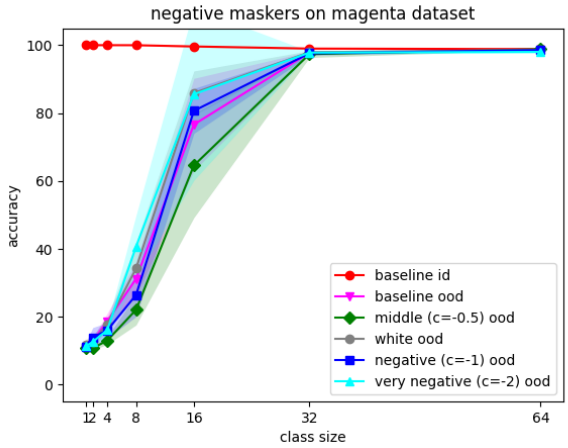


(a) Accuracy of several maskers with a negative constant for the white dataset

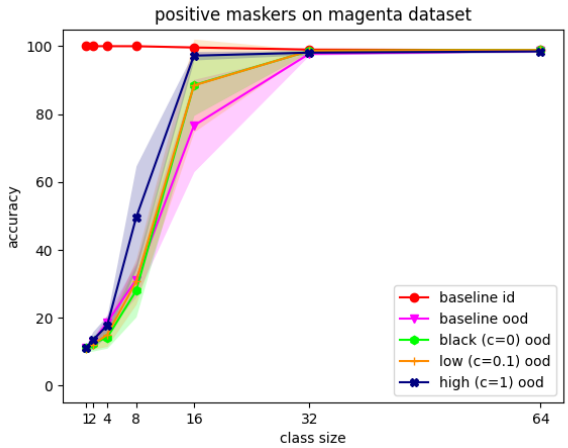


(b) Accuracy of several maskers with a non negative constant for the white dataset

Figure 8: accuracy of negative and non negative maskers on MNIST samples with a white color



(a) Accuracy of several maskers with a negative constant for the magenta dataset



(b) Accuracy of several maskers with a non negative constant for the magenta dataset

Figure 9: accuracy of maskers with a negative and non negative constant on MNIST samples with a magenta color

this research there was a bug in the code, which resulted in the fact that some models were trained with different hyperparameters, these results have not been included in the research. This was decided because these results could not be fairly compared to the other results and would make the paper more unclear. Every other result has been included in a graph.

During this research random seeds were used to generate the datasets. The datasets were also not selected based on whether they provided good results.

This research contributes to the field of machine learning, which already consumes a non negligible amount of energy and is expected to double by 2030 [3]. This means that the footprint of this field is increasing massively and care should be taken before adding to this field. However, this research aims to increase the accuracy of machine learning solutions with a small amount of extra overhead. Machine learning solutions are already widespread and deployed in critical situations. Therefore, increasing accuracy was deemed important.

6 Discussion and Conclusion

Maskers can be used to increase out of distribution accuracy. On every dataset there is a masker that significantly outperformed the baseline. This replicates the effect shown in previous research [2].

Which masker works is dataset dependent. Non negative maskers do not work for the black dataset, but do for the white dataset. With negative maskers it is the other way around. Only the middle masker works for both of them, which makes sense, since the distance from the target color to 0.5 is the same for both of them. The middle masker does not seem to work for the magenta dataset, however, where only the high masker got significant results.

Noise is only effective when the intensity of the noise reaches 100%. At that point it is equivalent to dropout [9] and more computationally expensive. It was therefore decided not to implement hybrid models with noise and masking, where the mask would add noise to the image instead of just multiplying itself with the image.

For future research, it is interesting to find out why some maskers work better than others. Additionally, it can be investigated whether adding a loss function that incentives the mask to mask out more of the original image increases the performance of masks that currently show no improvement. It can also be seen whether the accuracy can be improved by letting the model change the constant during training.

References

[1] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1):53, Mar 2021.

[2] Bram Grooten, Tristan Tomilin, Gautham Vasan, Matthew E. Taylor, A. Rupam Mahmood, Meng Fang, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Madi: Learning to mask distractions for generalization in visual deep reinforcement learning, 2023. https://arxiv.org/abs/2312.15339.

[3] IEA. Energy and ai, 2025. https://www.iea.org/reports/energy-and-ai.

[4] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Meike Nauta, Ricky Walsh, Adam Dubowski, and Christin Seifert. Uncovering and correcting shortcut learning in machine learning models for skin cancer diagnosis. *Diagnostics*, 12(1), 2022.

[7] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *CoRR*, abs/1911.08731, 2019.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

A Usage of AI

None of the writing in this project was generated by AI. Some suggestions were accepted that were posed by the AI Assist of overleaf, which is based on TeXGPT. AI has helped with the coding process, mainly to aid in understanding the machine learning framework and the graphing framework. These were some of the prompts:

- I am trying to get a graph in matplotlib that has subplots of 3x3 where in the top row two text elements are displayed and in the first column two text elements are displayed and in the last two rows and last two columns four colorplots are shown, how would i do that
- I have a pytorch tensor that is shaped (B, 3, 32, 32), where B is the batch size and it represents a 32 by 32 image with 3 color channels. I now want to create a boolean tensor that returns true for all completely black pixels, so 0.0 for every color channel
- Hey I am looking to combine images from the CIFAR10 and MNIST datasets into one dataset in pytorch, how may I do that