university of groningen

faculty of science and engineering

The Effect of Shape Selective Kernels in Convolutional Neural Networks

Stijn de Vries

university of groningen

faculty of science and engineering

University of Groningen

Constrained ConvNets and Their Effect on In- and Out of Distribution Shifts

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at University of Groningen under the supervision of
Dr. M. Valdenegro Toro (Artificial Intelligence, University of Groningen)
and
Dr. G. Azzopardi (Computer Science, University of Groningen)
and
G.S. Bennabhaktula (Information Systems group, University of Groningen)

Stijn de Vries (s3447146)

July 17, 2024

Contents

Acknowledgments

I would like to thank my supervisors George, Guru and Matias for their guidance throughout my final project and their insightful feedback and ideas. They have provided me with great suggestions and allowed me to explore a lot of different directions.
I also want to thank my family and friends for their continued interest and support during the past year. In particular, I want to thank my parents for their unconditional support throughout my whole academic career.
Lastly, I thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Hábrók high performance computing cluster.

Abstract

State of the art Convolutional Neural Networks (CNNs) can outperform humans in classification tasks, but are still not as robust and reliable as humans. Moreover, recent research suggests that CNNs have a strong texture bias, whereas humans develop a strong shape bias for object recognition. If CNNs were made to process visual data more like humans, an improvement in robustness and reliability of the visual models might be observed. The goal of this research is firstly to develop kernels that respond strongly to shapes in images, using biological visual cells for inspiration. Secondly, the effect that the shape selective kernels have on the performance of CNNs is investigated. Shape selective kernels were employed in CNNs which were trained in three different settings. The performances of the models were measured and the models were analyzed to reveal how the data was processed. Results show small improvements in some models, but a large decrease in both clean accuracy and robustness for most models using shape selective kernels. However, improvements in model certainty can be observed, and further analysis indicates that models with shape selective kernels process data differently than conventional CNNs.

1 Introduction

Over the last two decades, research in artificial intelligence (AI) has gained much attention due to the increasing performance of new models across a range of different tasks. With the arrival of Large Language Models (LLMs) especially, AI has entered the public domain as a tool for an increasing amount of tasks, like creation, identification or explanation.
As these models are released and are being used by an increasing amount of users, dealing with data from the real, uncontrolled world becomes inevitable. This data is not only extremely diverse but also noisy and can contain countless different corruptions or errors which degrade the quality of the data. Consequently, models that will be deployed in the real world should not only perform well on the training data, but should also be robust and consistently able to deal with diverse data that can be of lower quality. Especially as humans start to rely on the output that these models generate, the performance should not degrade significantly when these models are presented with data containing corruptions or data from unknown distributions.

Preventing the degradation of performance on corrupted data is especially important in applications where the degradation can have disastrous consequences, such as in self-driving cars. During driving, various corruptions of visual images can occur naturally from a host of different weather conditions like rain, sunshine or snow. Ideally, models should be able to avoid misclassifcations like the one shown in Figure 1 as choices by both models and humans might rely on these classifications when making a decision.

Moreover, the models that the car relies upon should be able to drive safely across all imaginable

Figure 1: **Left**: image of a vulture from the Imagenet dataset. **Right**: the same image of the vulture but with glass blur applied to it. A Resnet50 classified the left image correctly as a vulture, but the same model classified the right image incorrectly as a harvester. If decisions are made based on the output of these models, they should be robust against corruptions or alterations as the consequences of misclassifcations could be dangerous.

weather conditions, and furthermore in any new environment, even if the specific landscape or weather condition was not part of the models training data. Thus, it is paramount that the models that the car uses to perceive its surroundings are robust against corruptions and are general enough to deal with any imaginable visual situation.

Early Convolutional Neural Networks (CNNs) were usually trained on clean and processed data, which led to models that were not robust and could not generalise well. Though contemporary, state of the art vision models are often trained on data that includes corruptions or augmentations, robustness and generalisability is still not as good as that of humans [1].
Some state of the art models are still not robust [2], leading to severely degraded performance on data that is not part of the training distribution. Before such models can be used beneficially in day to day life, it is important that they become robust enough such that they can be relied upon in any circumstance, known or unknown. To achieve this, a general approach should be developed in which the model can deal with both in and out of distribution shifts. Furthermore, the performance of the model should be assessed extensively to make sure that the model is indeed robust.

One of the most robust and well-known models of vision is the human visual system [3, 4]. Humans are able to deal with corruptions very well and can classify objects in never seen before settings, and with great accuracy even when these objects have been seen only once or twice.
To create a robust and general CNN, it could be beneficial to study the different structures in the brain and the way that the brain processes visual data. Models that contain mechanics that try to imitate the way the human brain processes data can provide valuable information about what makes the human brain robust.

One well studied mechanism in the visual pathway of humans are the cells that specialise for certain stimuli, like movement, orientation or frequency [5]. These stimuli are called the preferred stimulus of a cell, referring to the fact that the associated stimulus produces the highest response in that specific cell. As the stimulus becomes more dissimilar to the preferred stimulus of the cell, the response of the cell to the stimulus quickly becomes smaller.
Similarly, kernels in CNNs filter for a pattern that can be found in the image data that they process. However, these kernels do not necessarily obtain the highest response for their associated pattern. This discrepancy will be the focus of the research, with the goal of finding out what the effect would be when the kernels of CNNs are constrained in such a way that they do obtain the highest response for their associated pattern, making them more like biological visual cells.

1.1 Research Questions

To summarize, this thesis focuses on the following problems:

Q1. How can kernels be created such that they obtain the highest response for their preferred stimulus?

Q2. What is the effect of filters that obtain the highest response for their preferred stimulus on the performance on clean data?

Q3. What is the effect of filters that obtain the highest response for their preferred stimulus on out-of-distribution shifts of data?

1.2 Thesis Outline

The contribution of this thesis is twofold: first, a method is proposed that creates kernels that always obtain the highest response for their associated pattern, making them process data more similar to the way biological visual systems process visual data. Secondly, this research shows that these kernels can slightly improve the performance of some models, in terms of training accuracy, robustness and model certainty. Furthermore, the thesis shows that this class of kernels changes the way that CNNs process image data.

The rest of the paper is structured as follows: First, an overview of the background literature will be presented. Then, the methods will be discussed which will be used to answer the proposed research questions. Afterwards, for each set of experiments, the experimental setup will be laid out, the results shown and a discussion provided to gain insight in the obtained information. Lastly the conclusion will present the main results and summarise the contributions of this research.

2 Background Literature

Robustness of vision models Robustness of vision models is a large area of research with several different topics. One of the most researched topics is adversarial robustness, which deals with changes to the inputs of a network that are deliberately designed to lead the network into making the wrong decision.
One famous example is the addition of a vector to a picture of a panda, which was invisible to humans but made the vision network classify the image as a gibbon instead of a panda with high certainty [6]. Although the model switched its prediction with high certainty, for humans the change was imperceptible, highlighting that humans and models classify images very differently. Adversarial robustness is important because models should be able to resist deliberate attacks that cause severely degraded performance or erroneous predictions.

This paper, however, focuses on the robustness where vision models learn to deal with corruptions like weather conditions and different types of noise or blur. A popular approach to measuring the robustness of models uses the performance on datasets that have images which are corrupted, distorted or changed in some way. Both Imagenet-C (INC) and Imagenet-P (INP) [7] are popular benchmarks that are used to measure robustness. Images from Imagenet (IN) [8] are used to create INC and INP by applying corruptions and perturbations respectively to the images of the Imagenet dataset.
These distorted or corrupted datasets are often referred to as the corrupted data, whereas the data which is used for training and which is unaltered is often referred to as the clean data. The accuracy obtained on the validation set is referred to as the clean accuracy, whereas the accuracy obtained on the corrupted data is referred to as the robustness of the model.

Contemporary vision models are not robust against natural variations, which include changes in pose or size [2]. These models required a sizeable amount of data with each specific variation to become robust against that variation, and robustness against one variation did not extent to robustness to other variations or classes.
Similarly, other work found that robustness of CNNs against a corruption occurs when the corruption is included in the training data, but this robustness to one class of corruptions generally does not extend to other classes [4].

As convolutional models have increased in size, performance on clean data has improved significantly but the relative robustness of these models has not improved as much [7].
More recent research suggests that robustness decreases as model size grows for both CNNs and transformer-based vision models [9]. The performances of models were compared against a baseline Resnet18 model on IN and several derivative datasets, and larger models generally showed a worse relative robustness.
Transformer-based vision models [10, 11, 12] seem to have a greater robustness than CNNs do. This results was also found in other studies [13, 14], and robustness for transformer-based vision models seems to scale better than CNNs with both number of parameters and size of training data [15]. However, one study obtained accuracies for IN and nine IN-derivative robustness benchmarks and averaged performance over the ten datasets. They found that some CNNs were able to outperform transformers in terms of mCE, conflicting with the previously discussed research [16].

Increasing robustness Data augmentation has become a standard method for increasing accuracy of models. Augmentation ranges from simple operations like mirroring, rotating or adding noise [17, 18]

to complex augmentation strategies like CutMix [19] or AutoAugment[20].
Data augmentation helps prevent overfitting and can improve performance of models significantly [21]. Furthermore, data augmentation can improve robustness of models [22, 23], especially against corruptions that were introduced by data augmentation [4, 24]. However, distortions that are included in the training data do not increase robustness against corruptions that were not included [4, 24].

Increased robustness can also be achieved by limiting the frequencies that the model relies on. For example, robust vision transformers seem to rely on low frequency features more than CNNs [13]. Although vision transformers might naturally rely on low-frequency components in the data, CNNs can be forced to rely on them as well. Increasing the reliance of CNNs on low-frequency components seems to increase the robustness of these models [25, 26].
Similarly, filtering of high-frequency features can improve robustness [27, 28], and by combining a method that can deal with low frequency data and a high-frequency filter, significant increases in robustness can be achieved [29, 28]. Moreover, the techniques used did not degrade the performance on the clean data.

In contrast, it is often observed that techniques that increase robustness lower the performance on the clean data slightly. This problem has been referred to as a trade-off between robustness and accuracy, and there has been a debate whether this trade-off is inherent to classification or that the existence of the trade-off does not necessarily exist [30, 31].
For example, introducing random patches that have Gaussian patch augmentation improves the robustness of models, but does not degrade the accuracy on clean data [32]. However, most mechanics that improve robustness generally have a small degraded performance on clean data [29].

Shape or texture bias Human object recognition is very robust for a range of different corruptions and perturbations [4], especially when compared to vision models [3, 23]. Humans develop a strong shape bias as they age from early childhood to adulthood [33], which means that shape is a very important factor feature for humans when classifying objects.

Originally, it was thought that CNNs processed images in a similar fashion [34], with filters in later layers progressively filtering for more complex shapes. However, recent research suggests that CNNs are more biased towards texture than towards shape [35, 36]. By creating images that have conflicting stimuli, the research found that CNNs mainly classified images according to the texture that was present, whereas humans mainly classified according to the shape.

A shape bias can be induced by creating a dataset that removes the texture cues, which forces the models to rely on shapes in order to classify images [35]. Not only did this induce a shape bias, but it also increased robustness against several different corruptions. However, creating stylized training data is costly, and requires more training time for the models as well since there is more data.
This shape-based approach to visual classification that humans use could be the reason why the human visual system is so robust. Different research however suggests that the robustness of models in the previously discussed research did not come from the shape bias but rather from the additional (augmented) data that the model was trained on [37].
They trained several models and assessed their shape bias and robustness, and found that a higher shape bias did not cause more robustness. Instead, they found that the increased robustness that was associated with shape bias instead was likely due to the extra training data was used, which was also augmented.

Biologically inspired vision Although it seems that a higher shape bias does not necessarily increase the robustness of a model, implementation of subsystems of the human visual system can still provide techniques that can produce robust classification models. Several projects have attempted to mimic parts of biological visual systems to observe their effect on performance and robustness.
For example, one research created kernels according to the McCulloch-Pitts neuron model and was able to identify the orientation of objects in images precisely and reliably [38]. Even for distorted images the model performed better than previous attempts, and furthermore used considerably less computation time and computational resources.

A different research designed a module based on the primate primary visual cortex (V1) and used it to process images before they were fed into the model [39]. This led to an increase in robustness across several corruptions, perturbations and adversarial attacks. Furthermore, this was achieved without data augmentation and was thus a general approach to creating a robust model.
Similarly, the push-pull inhibition phenomenon which occurs in the human visual pathway [40] was recreated in vision models and the enhanced models obtained both state-of-the-art accuracy on clean datasets, generalised better and were more robust against noise and corruptions. This body of research shows that the mechanisms in biological visual systems can provide a valuable source of robustness, efficiency and inspiration.

3 Methods

The goal of the current research is to ascertain the effect that kernels that obtain the highest response for the stimuli that they are filtering for (shape-selective kernels) have on both the performance of the model and the manner in which the model processes the image data. Specifically, the performance was measured using the clean accuracy and the robustness of the models on a corrupted dataset.
To find differences in the way the models processed the data, the uncertainty of the model was assessed using the Expected Calibration Error (ECE) metric, and the shape/texture bias was assessed for some models.

The methods section will first define shape selectivity and what a shape selective kernel is. Secondly, it will discuss how the shape selectivity was implemented in neural networks. Third, the datasets and architectures that were used will be discussed and lastly the evaluation of the performance of the models is explained.

3.1 Shape selectivity

In images, shapes and more fundamentally edges, occur due to a difference in values of neighbouring pixels. In a grayscale picture, when one half of the image is black and the other half of the image is white, humans see a line in the middle of the picture. Similarly, a white box in the middle of an otherwise black image consists of several straight lines and corners, with the middle of the box being a homogeneous white region.
As a shape consists of one or more edges in some configuration, for a shape to exists it is required that there exists some difference in magnitude between pixels in a certain region. The higher the difference between pixels, the more pronounced a shape occurs to humans. Thus, when a region in an image solely contains pixels with the same values, it cannot contain any shapes since there is no edge in that region to create any shape.

This definition of shapes in images will allow us to define what shape selectivity is, together with the biological visual cells of cats that were found to respond to specific stimuli [5]. In biological visual systems, there are visual cells that are shape selective, meaning that they produce a high responses for one specific shape, but very little for any other shape. The specific shape for which they produce a high responses will be referred to as their associated pattern. Thus, the associated pattern of some cells can be a horizontal line whereas for others it might be a vertical line.

Similarly, kernels in CNNs filter for a specific pattern, which can be called the associated pattern of that kernel. Because convolutions in CNNs are actually implemented as the cross correlation, the convolution of a CNN provides an output map of similarity scores between the pattern at some location in the image and the kernel.
When the pattern and kernel are similar, the cross correlation obtains a relatively large, positive number. On the other hand when a pattern and a kernel are not at all similar the cross correlation obtains a relatively small or negative number. Because the output of the cross correlation function increases with similarity and decreases with dissimilarity, the associated pattern of a kernel will be defined as the kernel itself. A kernel that filters for a shape and obtains its highest response for its associated pattern will be referred to as a shape selective kernel.

1	0.5	-0.1
1	0.5	-0.1
1	0.5	-0.1

(A)

1	0.5	-0.1
1	0.5	-0.1
1	0.5	-0.1

(b)

1	1	-1
1	1	-1
1	1	-1

(c)

Figure 2: From left to right: (A) kernel A, (b) kernel A's associated pattern and (c) the pattern for which kernel A obtains the highest response.

Figure 2 shows an example of a kernel A, the kernel's associated pattern b, and a different pattern c. The response that a kernel obtains for some pattern is easy to calculate, and shown in Equation 1:

$$CC(A,b) = \sum_{n=0}^{i} \sum_{m=0}^{j} A_{n,m} * b_{n,m} \tag{1}$$

Here, CC stands for the cross-correlation of a kernel and a pattern, where A is the kernel and b the pattern, and i and j refer to the width and height respectively of the kernel and the pattern. Furthermore, we have $i = j$. Lastly, $A_{n,m}$ indicates the value of kernel A at row n and column m. Similarly, $b_{n,m}$ indicates the value a pattern b at row n and column m.
The cross correlation of kernel A and its associated pattern b obtains a response of $CC(A,b) = 3.78$. However, the cross correlation of kernel A and pattern c obtains a response of $CC(A,c) = 4.2$. Thus, kernel A does not obtain its highest response for its associated pattern, and is not considered a shape-selective kernel.

-1	1	-1
-1	1	-1
-1	1	-1

(D)

0	1	0
0	1	0
0	1	0

(e)

-0.5	1	-1
-1	1	-1
-1	0.5	-1

(F)

0	1	0
0	1	0
0	0.5	0

(g)

Figure 3: From left to right: (D) kernel D, (e) kernel D's associated pattern, (F) kernel F and (g) kernel F's associated pattern.

Although the range of input values is theoretically unbounded, in practice this is often not the case due to for example batch normalization layers or the ReLU activation function. Thus, suppose that the input values for convolutional layers are bounded by the range $[0,a]$, where a is a positive number. Furthermore, suppose a is set to 1. Strictly, this would mean that any shape selective kernel would only contain 0s and positive numbers, since there exists no associated patterns with negative values. Furthermore, if a kernel does not have an associated pattern, it cannot be shape selective. Consequently, there would not be any shape-selective kernels since a kernel whose values are all positive would obtain the highest response for a pattern whose values are all one, which is not to be regarded as a pattern, as discussed earlier.

Instead, the values of the associated patterns will be bounded by the defined range, but the pattern will still be regarded as the associated pattern as it is the closest pattern that can occur in practice. Then, kernels are allowed to contain negative weights, which is necessary for shape selectivity to occur.
Figure 3 shows two kernels D and F and their associated patterns e and g. The response of kernel D for its associated pattern is $CC(D,e) = 3$. Since the cross-correlation is just the summation of each term, the only way to increase the response is to increase any of the terms. Furthermore, influencing the values of one term of the equation does not influence any other terms in the equation. There are 2 unique terms in the equation for $CC(D,e)$:

- $1 * 1 = 1$ (for the terms $D_{1,0} * e_{1,0}$, $D_{1,1} * e_{1,1}$ and $D_{1,2} * e_{1,2}$)
- $-1 * 0 = 0$ (for the terms $D_{0,0} * e_{0,0}$, $D_{0,1} * e_{0,1}$, $D_{0,2} * e_{0,2}$, $D_{2,0} * e_{2,0}$, $D_{2,1} * e_{2,1}$ and $D_{2,2} * e_{2,2}$)

To increase the response for pattern e, either the 0 values should be decreased or the 1 values should be increased. However, since these values are bounded by the range $[0,1]$ as stated before, the response of kernel D cannot be increased by changing pattern e. Thus, D obtains the highest response for e and since e is the associated pattern of D, and as such D is shape selective.
On the other hand, performing the same steps for kernel F, the response of kernel F for pattern g can be increased by increasing the value for $g_{1,2}$ from 0.5 to 1. The pattern that is obtained by increasing the value for $g_{1,2}$ will be referred to as the pattern g'. This results in an increases in response from $CC(F,g) = 2.25$ to $CC(F,g') = 2.5$. Thus, F does not obtain the highest response for its associated pattern g, and so F is not shape selective.

From this example, it is clear that for a kernel to be shape selective, the positive weights in the kernel should all have the same value. This characteristic makes it so that each value in the pattern is treated exactly the same, and so each part of the shape or edge in the pattern contributes the same amount.
Similarly, each negative weight in the kernel should have the same value since each pixel that is not part of the shape should discount the magnitude of the response equally.
When the limit a in the range as described previously is 1, kernel D from Figure 3 is shape selective. However, for $a = 2$, it is not, but the kernel $2 * D$ would be shape selective. The range of values that the pixels in the input image can have is not explicitly known and depends on factors like the type of task, the images in the dataset, the preprocessing methods and the location of the convolutional layer within the architecture of the model, and can furthermore be influenced by corruptions. Instead of a data-dependent definition, a general definition of shape-selectivity for kernels in CNNs is introduced.

Thus, a kernel is shape selective kernel if it has the following three characteristics:

1. The kernel contains no weights that are exactly 0
2. All positive weights in the kernel are the same
3. All negative weights in the kernel are the same

This definition is simple both in theory and in practice, and is shape-oriented since the positive and negative values in the kernels create edges which are all equally distinctive.

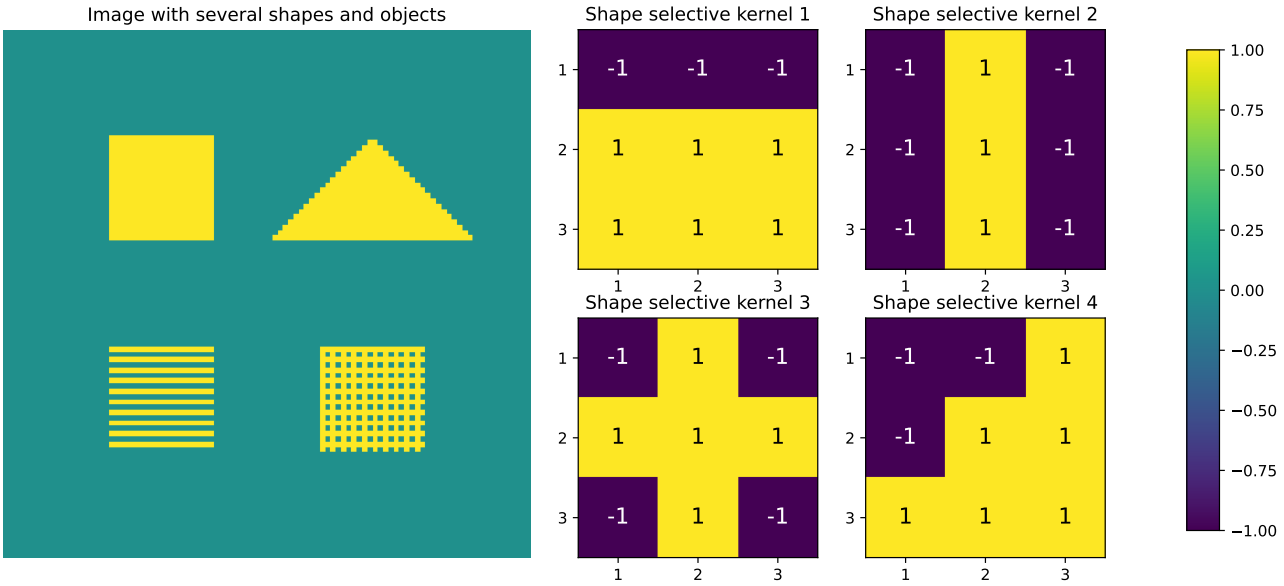

Figure 4: An image with several shapes and objects, and four shape selective kernels. Values in the image range from 0 to 1. Values in the kernels range from -1 to 1.

3.2 Shape selective kernels

Figure 4 shows an input image containing several shapes, and four different shape selective kernels. These kernels all filter for their associated pattern, which is the pattern that is the kernel itself.

Figure 5 shows the activation maps that were obtained by convolving the image and each of the kernels from Figure 4.

As can be seen by the activation maps for kernels 1, 3 and 4, the patterns that they filter for are present in the image, and the kernels obtain the highest response at the locations in which their associated pattern is present. Notice that the response that kernel 3 obtains for its associated pattern is lower than the response that kernels 1 and 4 obtain for their associated responses. This is due to the fact that kernel 3 has five positive weights whereas kernels 1 and 4 have six positive weights. This means that the highest response kernel 3 can obtain is 5, whereas the highest response for kernels 1 and 4 is 6, provided the pixel values in the image are bounded by the range $[0, 1]$.
The activation map of kernel 2 shows almost no positive responses. This is due to the fact that the associated pattern of kernel 2 is not present in the image. Furthermore, it also a result of the fact that the kernel contains six negative weights, which means that it will obtain mostly negative response unless the pattern closely resembles the associated pattern of the kernel.

The discrepancy between the magnitudes of the 'highest response' between kernels can be counteracted by using a bias in the convolutional layer, or alternatively by using batch normalization layers, as they are applied in the Resnet models. Secondly, the magnitude of the positive and negative weights need not be the same across kernels in a convolutional layer, meaning that a kernel with six negative weights can have a smaller value for the negative weights than that of a kernel with only three negative weights.

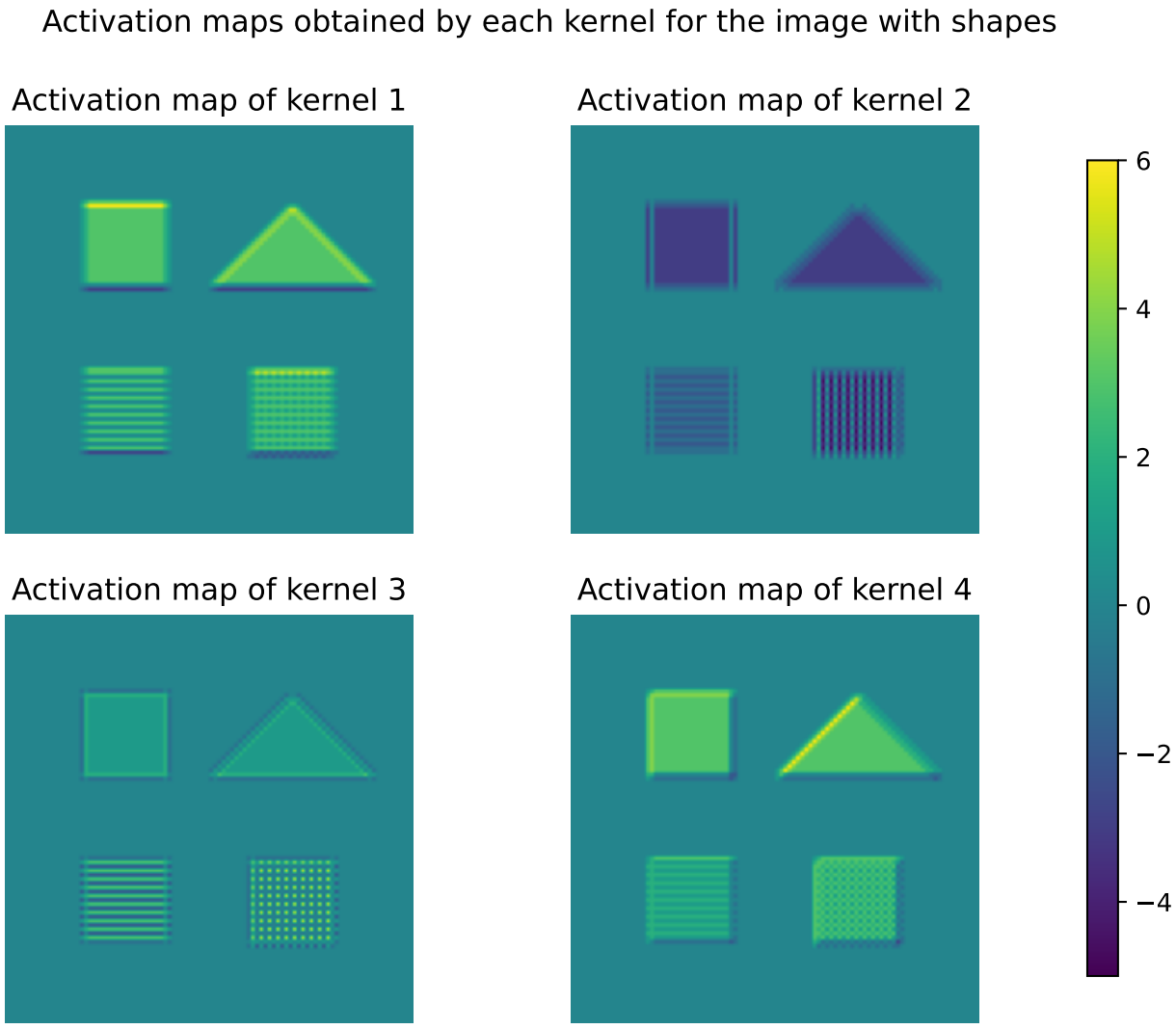

Figure 5: The activation maps that each kernel from Figure 4 obtains for the image in the same figure.

3.3 Current approach

Shape selective kernels can be employed in conventional CNNs in various different ways. In the current research, three different approaches will be implemented and tested. Two of these approaches include different types of shape selectivity, and the last approach proposes a new type of convolutional layer, the Polar Orientation (PO) layer.

3.3.1 Shape selectivity in networks

Shape selectivity was introduced in convolutional layers using parametrization. Parametrization uses the original weights in the network as parameters for the actual weights that will be used during inference and training. This is achieved by using a parametrization function that transforms the weights before they are used during inference. The gradients however will be applied to the original weights, updating them and thus changing the parametrized weights with them.
Equation 2 shows the parametrization for the shape selective (SS) parametrization, the first type of shape selectivity.

$$p(x) = \begin{cases} a & \text{if } x < 0 \\ b & \text{if } x >= 0 \end{cases} \tag{2}$$

Here x represents the weight before parametrization and $p(x)$ the weight after parametrization. Parameters a and b are learnable, with a in the range $[-\infty, 0)$ and b in the range $(0, \infty]$, to ensure that the negative weights stay negative and the positive weights stay positive.
This parametrization ensures that all negative weights have the same magnitude and that all positive weights have the same magnitude. Furthermore, it ensures that there is no weight that has the exact value of 0. Thus, the parametrization creates kernels that are shape selective as defined in Section 3.1.

The SS parametrization is applied to the weight of a convolutional layer as a whole. However, each convolutional layer consists of several kernels. Thus, this parametrization can also be applied to each kernel individually instead of the convolutional layer as a whole. Equation 3 shows the parametrization when it is applied to each kernel independently, and this type of shape selectivity is referred to as independently shape selective (ISS).

$$\forall k \in K : p(x^k) = \begin{cases} a_k & \text{if } x^k < 0 \\ b_k & \text{if } x^k >= 0 \end{cases} \tag{3}$$

Here, parameters a and b are vectors of size K where K indicates the amount of kernels in a convolutional layer. The values in a are bound by the range $[-\infty, 0)$ and the values in b are bound by the range $(0, \infty]$. The weight x^k is a weight in the k-th kernel. This parametrization ensures that values with the same sign are the same within a kernel but are not necessarily the same as the values for other kernels.

Implementation of either SS or ISS kernels in convolutional layers of a CNN can be achieved easily by parametrization of weights of convolutional layers in existing architectures.

3.3.2 Polar Orientation Layer

Suppose a convolutional layer has a kernel of size 64x3x3, e.g. 64 kernels of size 3x3, and the SS parametrization is applied to it. Then, there are only two values in this kernel; one negative and one positive. Because the kernels are shape selective and they have the same values for their negative and positive weights, the responses of each kernel can be compared since the magnitudes of the responses will be the same.
Out of all the kernels, there is one that will obtain the highest response. Since each kernel filters for a specific shape (their associated pattern), the associated pattern of the kernel that achieves the highest response is the best approximation to the actual pattern that is present. Instead of feeding forward all outputs of all kernels, only the magnitude of the response and the information about what kernel obtained that response are fed forward to the next layer.
This design only conveys the most important shape information to the next layers, as it detects what pattern is most present at a location, and feeds only that information to the next layer. Since such a layer detects shapes and only conveys shape information, it should be sensitive to shape information and furthermore rely on shape information.

To achieve such a layer, the Polar Orientation (PO) layer is proposed, named after the polar coordinates of vectors. The output of this layer consists of an activation map for each filter and one additional

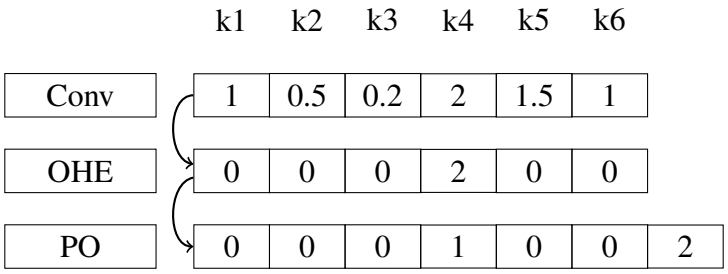

Figure 6: Visualization of the calculations of the PO layer. The "Conv" vector is a depth slice over the output of a shape selective convolutional layer at some location. Thus, each value is the response of a kernel at that location, here indicated by the six kernels k1 through k6. In the "OHE" step only the outputs of the kernels that obtained the highest response are retained. In the "PO" step, a 1 for some kernels indicates that the kernel obtained the highest response, and the size of the highest response (here the number 2) is added as an extra output map.

activation map. The added activation map contains information about the size of the highest response at each location.
For each activation map of the N filters, for each location, if some kernel obtains the highest response out of all kernels, a 1 is placed in that location for that kernel. If the kernel did not obtain the highest response, a 0 is placed in that location for that kernel. When multiple kernels obtain a response of the same magnitude, the output will contain multiple 1s since none of the responses is more pronounced than any other.
Then, for each location in the image there is a vector of size N which is almost always one-hot encoded, except for when 2 responses are the same. Each of these vectors provides information about what kernel obtained the highest response, and how high the response is. Since shape selective kernels will be used, this equates to providing information about what shape is most present at some location of the image. Figure 6 shows a visualization of the calculations performed in the PO layer.

Equation 4 shows the one-hot encoding (OHE) of the output of the convolutional layer.

$$OHE(x) = \forall h \in H, \forall w \in W, \forall k \in K: \ \mathrm{f(x)}_{h,w,k} \begin{cases} 1 & \text{if } f(x)_{h,w,k} = \max\limits_{n \in N} f(x)^n_{h,w,k} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Here, x is the input tensor to the PO layer. $f(x)$ represents a standard convolution and H, W and N represent the height, width and the activation map respectively. Equation 5 shows the output of the PO layer by concatenating the one-hot encoding and the size-response activation map.

$$PO(x) = OHE(x) \oplus \left[\forall h \in H, \forall w \in W \max_{n \in N} f(x)^n_{h,w} \right] \tag{5}$$

For a standard convolutional layer with N filters, batch size B and height and width H and W respectively, the output would be of size $B \times N \times H \times W$. However, since the PO layer adds an extra output map, the output size would be $B \times (N+1) \times H \times W$ instead. However, the PO layer will replace standard convolutional layers and thus its output would not match the input size of following layers. Instead, since the added output map has the same size as the output map of a single filter, one filter is removed from the convolutional layer, which means that the output size of the layer stays the same. This allows the PO layer to be swapped in for a normal convolutional layer seamlessly.

3.4 Evaluation metrics

Several evaluation metrics were used to gain insight in how the models process the input data and predict the class it belongs to. Models were evaluated on several datasets using accuracies and the Expected Calibration Error (ECE) [41], which is a measure of model uncertainty. Furthermore, important features in the input data were extracted using a method called occlusion sensitivity [42].

3.4.1 Accuracy

Models were evaluated on two versions of a datasets: a clean dataset and its corrupted version. The clean datasets used in the current research are MNIST, Tiny Imagenet and Imagenet, and their corrupted versions are MNIST-C, Tiny Imagenet-C and Imagenet-C. Performance on the clean datasets was measured using the validation accuracy in the last epoch of training, as in Equation 6.

$$\text{Accuracy} = \frac{1}{|N|}\sum_{i=1}^{N}(\hat{y}_i = y_i) \tag{6}$$

Here, N is the dataset containing image-label pairs where the image is represented by x_i and the label by y_i. The output of the model to image x_i is represented by $\hat{y}_i$. Since the model trains on each image in the dataset once in every epoch, we obtain an average accuracy over the whole dataset.
For the MNIST-C dataset, performance of models was measured as the average accuracy of the model over all corruptions, as shown in Equation 7.

$$\text{Corrupted Accuracy (MINST)} = \frac{1}{|C|}\sum_{i=1}^{C}\frac{1}{|C_i|}\sum_{j=1}^{C_i}\hat{y}_{i,j} = y_{i,j} \tag{7}$$

Here, C is the total corrupted dataset, containing i different corruptions applied to the MNIST dataset. Then, C_i represents the dataset for one specific corruption, containing image-label pairs $x_{i,j}$ and $y_{i,j}$. Since the corrupted datasets for Tiny Imagenet and Imagenet contain 5 severities per corruption, the accuracy is calculated as shown in Equation 8:

$$\text{Corrupted Accuracy (TINC/INC)} = \frac{1}{|C|}\sum_{i=1}^{C}\frac{1}{|C_i|}\frac{1}{5}\sum_{s=1}^{5}\sum_{j=1}^{C_i}\hat{y}_{i,j,s} = y_{i,j,s} \tag{8}$$

where s indicates the level of severity of the corruption and the dataset of corruption i and severity j contains the image-label pairs $x_{i,j,s}$ and $y_{i,j,s}$. Details about performance per corruption was included in the Appendices.

The Imagenet models were furthermore evaluated on 3 different modified datasets to asses their shape and texture bias, and their performance on datasets containing solely shape or edge information. These datasets are the edge, silhouette and cue-conflict datasets from a research by Geirhos et al. [35].
Figure 7 shows an example from each dataset. For the edge and silhouette datasets, the texture information is completely removed, and only the shape remains. The accuracy of the models for these datasets can provide insight into the way that models process the data, since models require a shape representation of the classes to be able to classify the images.
For the cue-conflict dataset, there are two different classes present in each image. The shape information in the image is from one class, whereas the texture information is taken from another class. For example, the right image in Figure 7 shows the shape information of a plane but the texture information of an elephant. The cue conflict dataset provides insight in the preferences or biases of models,

Figure 7: An example image from three different datasets. From left to right, the images are from the edge, silhouette and cue conflict dataset. For the right image, the shape class is ”airplane” whereas the texture class is ”elephant”. Images were obtained from work by Geirhos et al. [35]

since they will classify the images according to either the shape or the texture information, and if they consistently prefer one approach to another, they likely are biased towards one class of stimuli, e.g. either shape or texture.

3.4.2 Model uncertainty

To asses the model calibration, the Expected Calibration Error (ECE) [41] will be used. The ECE measures the expected difference between the confidence of the model and its accuracy. This is achieved by binning predictions and obtaining the difference between the accuracy and confidence in each bin. The weighted average over all the bins then obtains the ECE score, as shown in Equation 9.

$$\mathrm{ECE} = \sum_{i=1}^{I} \frac{|B_i|}{n} \mathrm{acc}(B_i) - \mathrm{conf}(B_i)| \tag{9}$$

Here, I is the number of bins and n the number of samples in bin B_i. The accuracy $\mathrm{acc}(B_i)$ for bin B_i is calculated according to Equation 10,

$$\mathrm{acc}(B_i) = \frac{1}{|B_i|} \sum_{j \in B_m} (\hat{y}_j = y_j) \tag{10}$$

where $\hat{y}_j$ is the predicted class label and y_j is the true class label for sample j. The confidence $\mathrm{conf}(B_i)$ of bin B_i is calculated according to Equation 11,

$$\mathrm{conf}(B_i) = \frac{1}{|B_i|} \sum_{j \in B_m} \hat{p}_j \tag{11}$$

where $\hat{p}_j$ is the confidence for sample j, which is calculated as the euclidean distance between the softmax output for sample j and the one-hot encoded true class label. For calculating the ECE scores, $I = 15$ bins were used.

3.4.3 Occlusion sensitivity

Occlusion sensitivity was introduced by Fergus & Zeiler [42] and aims to discover what features of the input data are important for trained models. This is achieved by masking the input at several

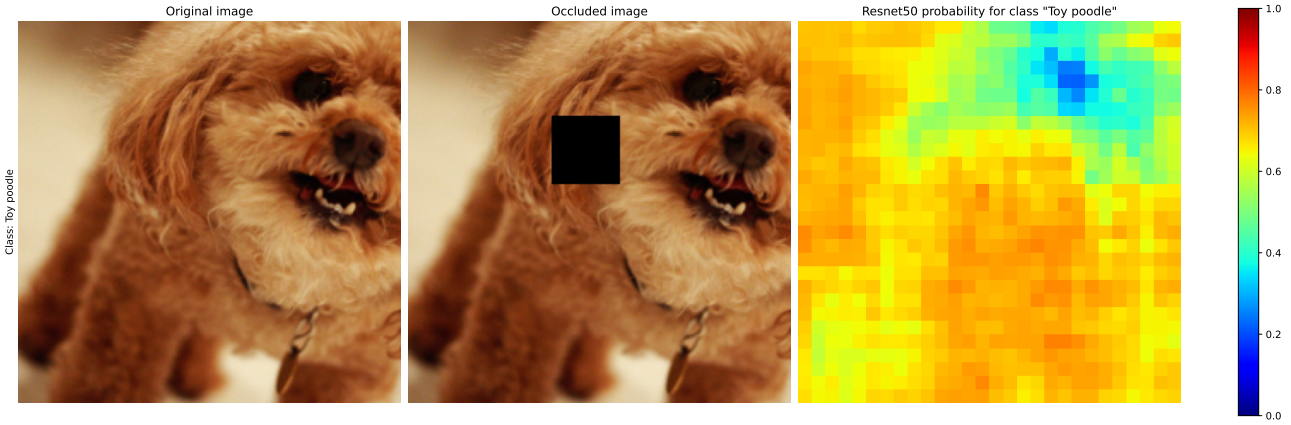

Figure 8: Left: An example of an image from the Imagenet dataset. Middle: the image with a mask applied to a small region. Right: the probability of a trained Resnet50 model predicting the occluded image as the class ”Toy poodle”, as a function of the mask at different locations in the image.

locations and having the model predict each masked image. Figure 8 shows an example of an image, a mask applied to the image at one location, and a plot of the (Softmax) probability for the class ”Toy poodle” that a Resnet50 assigns to the image.
The plot (right image in Figure 8) is obtained by moving the mask over the original image and obtaining the output probabilities for the class ”Toy poodle” from a trained Resnet50. The size of the mask is 40 by 40 pixels and a stride of 7 in both directions was used.
Figure 8 shows that a mask applied to the top left of the image does not impact the probability for the class ”Toy poodle” much, as the probability is still high. However, when the mask occludes the face and specifically the eye of the toy poodle in the image, the probability drops significantly. Thus, the face of the toy poodle is an important feature for the model to classify the image as ”Toy poodle”, the correct class.

4 Experiments and results

The effect of shape selective kernels will be evaluated on three clean datasets and their corrupted versions: MNIST and MNIST-C, Tiny Imagenet (TIN) and Tiny Imagenet-C (TINC), and Imagenet (IN) and Imagenet-C (INC). For each dataset, first the dataset and its corrupted version will be discussed. Then, the experimental setup will be explained. Afterwards, the results of the performed experiments are presented and lastly the results will be discussed.
The first section will cover the MNIST dataset, the second the Tiny Imagenet dataset and the third the Imagenet dataset. As larger CNNs are used for the larger datasets, the results of the experiments performed in the smaller datasets (MNIST and Tiny-Imagenet) were used to determine what experiments were to be performed for the larger datasets (Tiny-Imagenet and Imagenet) as training takes longer and there exists too many configurations to perform exhaustive testing.
There are 4 types of modifications that were tested for all models: shape selective (SS) parametrization, independent shape selective (ISS) parametrization, a polar orientation layer (PO) with shape selective kernels (the PO/SS modification), and a PO layer with standard convolutional kernels. The PO modification is included as an ablation experiment and allows for observing whether a change in performance for the PO/SS modification is due to the SS parametrization or the PO modification.

4.1 MNIST

MNIST is a dataset that contains 60000 handwritten digits with 10 classes, numbers 0 through 9. The images in MNIST are grayscale and have a resolution of 28 by 28 pixels. The MNIST dataset is considered as an easy task as most models can obtain close to a 100% accuracy. MNIST-C [43] contains images from the MNIST dataset, but with corruptions applied to each image. In total, there are 15 different corruptions, with each corruption applied separately to each image of the clean dataset. A detailed overview of the corrupted MNIST dataset and the corruptions that are included in them is shown in Appendix A.

4.1.1 Experimental setup

For MNIST, a small custom convolutional network was used, which will be referred to as the SmallNet. The structure of SmallNet is shown in Table 1.
Images were preprocessed by normalizing them with $\mu = 0.5$ and $\sigma = 0.25$. No further data augmentation was used.
Models were trained for 30 epochs with Stochastic Gradient Descent (SGD), using a cosine learning scheduler [44], a starting learning rate of *0.01*, momentum of *0.9* and weight decay with value 0.0001. After the model was trained, it was evaluated on the corrupted dataset. For each model 5 runs were performed, and the obtained accuracies and scores were averaged over the 5 runs.
A baseline SmallNet model was trained with the base architecture as shown in Table 1 to compare the obtained results.

4.1.2 Experiments

Since the SmallNet model contains only two convolutional layers, exhaustive experiments were performed, resulting in 3 experiments for every modification available.

Layer type	Number of kernels	Output size	Kernel size	Other
Input layer		1x28x28		
Convolutional layer	64	64x28x28	3x3	
ReLU				
Max pooling layer		64x14x14	2x2	
Convolutional layer	64	64x14x14	3x3	
ReLU				
Max pooling layer		64x7x7	2x2	
Flatten		3136		
Dropout				$p = 0.5$
Fully connected layer		128		
ReLU				
Dropout				$p = 0.25$
Output layer		10		

Table 1: The architecture of the SmallNet, used for the MNIST dataset.

For each modification, three experiments were performed. One with the modification applied to the first convolutional layer (Conv1), one with the modification applied to the second convolutional layer (Conv2), and one where the modification was applied to both layers. Thus, for MNIST 12 experiments were performed in total.

The trained models will be named according to their architecture. Any change in architecture will be included in the name of the model. The base model will be referred to as SmallNet$_{base}$. Parametrization of the first convolutional layer with either SS or ISS is indicated by the subscript of $L1_{SS}$ or $L1_{ISS}$ respectively, resulting in the name SmallNet-$L1_{SS}$. The use of a PO layer is indicated using a superscript, as in SmallNet-$L1^{PO}$.

4.1.3 Results

The average performance of the models for the clean data and the corrupted data and the average ECE value over all 5 runs are shown in Table 2. The accuracies for each model for every corruption can be found in Appendix B.1. Model SmallNet-$L1_{ISS}$ obtained the highest clean accuracy. Model SmallNet-$L1^{PO}$ obtained the highest robustness and the lowest ECE score.
Out of the 13 models trained, 7 models learned to perform the task in all 5 runs. The accuracies obtained by these models for each corruption are shown in Figure 9.
Figure 10 shows the occlusion sensitivity of the base model and four other models.

4.1.4 Discussion

Several models (models SmallNet-$L2_{SS}$, SmallNet-$L12_{SS}$, SmallNet-$L12_{ISS}$, SmallNet-$L12^{PO}_{ISS}$ and SmallNet-$L12^{PO}_{ISS}$) were not, or only sometimes were, able to perform the task. This is indicated by the degraded performance[1] of these models, for both the clean accuracy and robustness.

[1]The performance of modified models are always compared to the base model, unless stated otherwise.

Model	MNIST (↑)	MNIST-C (↑)	ECE (↓)
SmallNet$_{base}$	98.35%	86.69%	0.0937
SmallNet-$L1_{SS}$	98.34%	85.13%	0.1073
SmallNet-$L2_{SS}$	11.22%	11.10%	0.0205
SmallNet-$L12_{SS}$	36.67%	26.91%	0.0729
SmallNet-$L1_{ISS}$	**98.37%**	86.01%	0.1035
SmallNet-$L2_{ISS}$	45.77%	38.59%	0.0579
SmallNet-$L12_{ISS}$	11.22%	11.03%	0.0544
SmallNet-$L1_{SS}^{PO}$	97.94%	84.90%	0.0808
SmallNet-$L2_{SS}^{PO}$	11.22%	11.10%	0.0205
SmallNet-$L12_{SS}^{PO}$	11.22%	11.10%	0.0205
SmallNet-$L1^{PO}$	98.11%	**86.78%**	**0.0773**
SmallNet-$L2^{PO}$	96.66%	66.37%	0.1679
SmallNet-$L12^{PO}$	95.47%	71.60%	0.1317

Table 2: Performance of models for the clean data (MNIST), the corrupted data (MNIST-C) and the ECE. Performance is expressed as accuracy. For MNIST, the accuracy is over the validation set, obtained during the last epoch of training. For MNIST-C, the accuracy is the mean accuracy over all corruptions in the MNIST-C dataset. Values in **bold** indicate the best performing model for that dataset or for the ECE score. Arrows indicates whether higher is better (↑), or lower is better (↓)

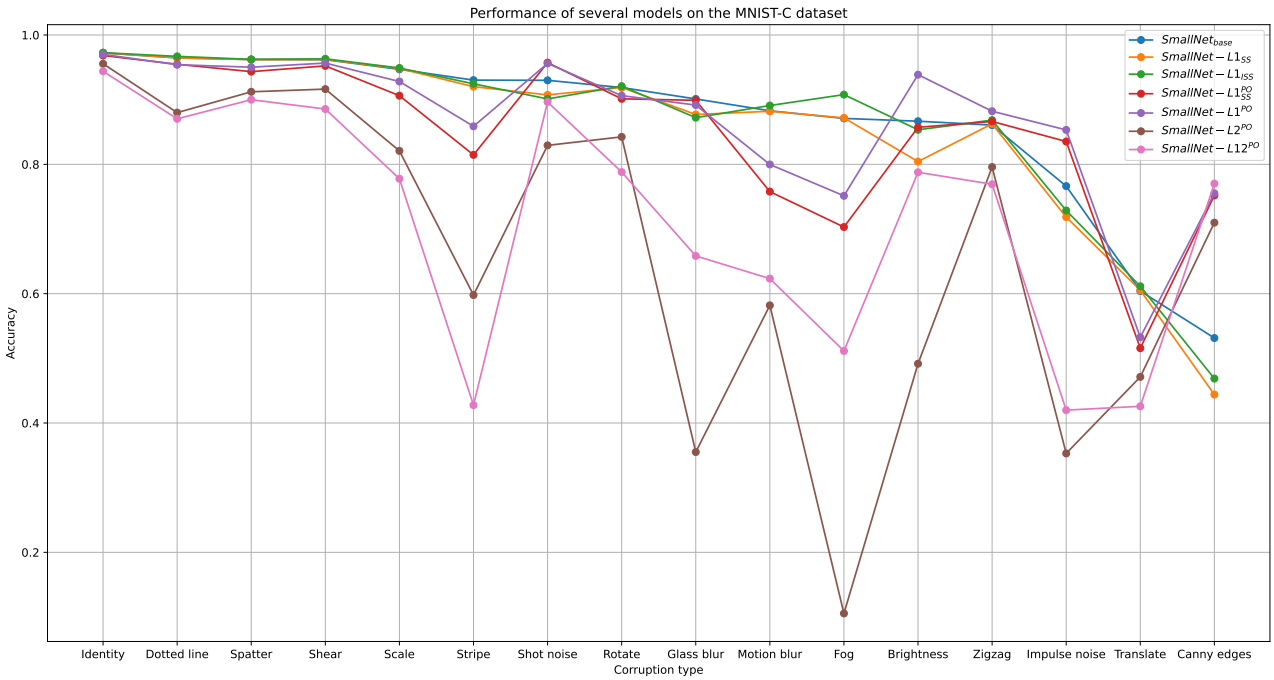

Figure 9: The accuracy for every corruption for the SmallNet models. Corruption are sorted according to the performance of the base model, from highest to lowest obtained accuracy.

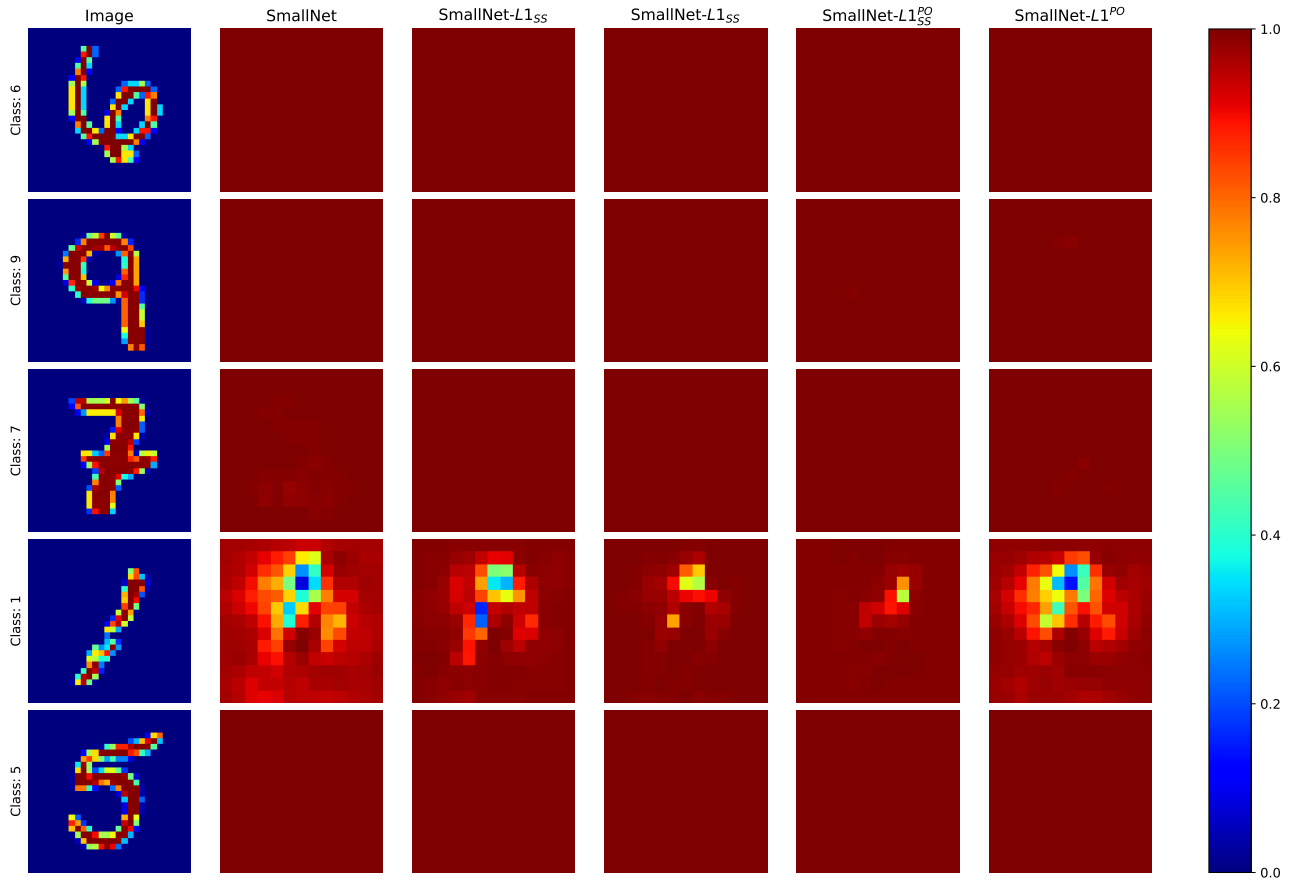

Figure 10: The occlusion sensitivity (Section 3.4.3) for five different images from the MNIST dataset. The first column shows the original image and the label. Columns two through six show maps of the Softmax probabilities that the models assigned to the image as a function of the location of the mask. The models from left to right are SmallNet$_{base}$, SmallNet-$L1_{SS}$, SmallNet-$L1_{ISS}$, SmallNet-$L1^{PO}_{SS}$ and SmallNet-$L1^{PO}$. The mask size was 4 by 4 pixels and the stride was set to 2.

Although models SmallNet-$L2^{ISS}$ and SmallNet-$L12^{SS}$ seem to be able to distinguish between some classes, this is due to the fact that the values are the average over 5 runs. In some runs, the model was able to learn to perform the task but in some others it was not. The ECE score of models also decreases drastically when models are not able to perform the task. However, this does not indicate that these models are certain about their decisions, but rather that they are as uncertain as they should be since they are just randomly guessing.

The models that used SS or ISS parametrization in either the second layer or both layers have severely degraded performance with respect to the base model. This decline in performance could be due to a decrease in representational power of the kernels. The weights can effectively only take two values, those being either a single negative value or a single positive value. This can make it difficult for the network to delineate between classes that might only differ a small amount, hurting the classification capabilities significantly.
A different reason for the drastic decline in performance for parametrization in the second convolutional layer but not the first could also be the amount of weights in the layer. The second convolutional layer contains 64 times the amount of weights that the first has, but the weights can still take only 2 values. It could be the case that as the representations in the later layer grow more complex, it is much

harder or impossible to capture these representations with just 2 values. Thus, this seems to indicate that the representational power of shape selective kernels is not as large as that of conventional kernels.

While models SmallNet-$L1_{SS}$ and SmallNet-$L1_{ISS}$ have increased ECE scores, models SmallNet-$L1_{SS}^{PO}$ and SmallNet-$L1^{PO}$ have lower ECE scores, suggesting that the PO layers are responsible for this decrease. The cause of this reduction in model uncertainty could be due to the type of information available to the models. Though the models with PO layers only have a small amount of information available, the information is about notable shapes and features that are present in the images. Since every location only contains one detected shape, there is no conflicting information that impacts the certainty of the models decision. Thus, it could be that the model effectively removes one source of uncertainty which is uncertainty from background noise, resulting in a better calibrated model.

Model SmallNet-$L1^{PO}$ performs almost up to par with the base model in terms of clean accuracy and very slightly outperforms in terms of corrupted accuracy. This is surprising, due small amount of information in the output of the first convolutional layer, caused by the one hot encoding of the PO modification.
For a random image, about 13.4% of the values in the output of the first convolutional layer of the base model are zeroes, but for the output of the first convolutional layer in model SmallNet-$L1^{PO}$ (which is a PO layer), about 96.9% of the values are zeroes. This illustrates that the amount of information in the modified model is a lot smaller, but that the quality of the information is higher, since the model is still able to compete with the performance of the base model.

A clear discrepancy between models with and without the PO modification can be seen in Figure 9 for the ”Canny edges” corruption. This corruption removes all information from the image but retains the contour lines that are visible. Thus, the models have only shape information to rely on for their prediction. It seems that the models with PO layers are better able to deal with this type information, and rely on it more than models without such a layer.

The occlusion sensitivities in Figure 10 show that there are some small differences in processing, specifically for image which is class ”1”. However, there is no clear pattern and for all other examples the Softmax probabilities do not change at all, as all models are confident in their predictions regardless of the position of the mask.

4.2 Tiny Imagenet

Tiny Imagenet is a subset of Imagenet which consists 200 classes with 500 images per class. The images are in color, and are obtained by taking images from IN and downsizing them to a resolution of 64 by 64.
Tiny Imagenet-C contains the images of the validation set of TIN, but have 19 different corruptions applied to them. In the experiments performed in the current paper, only 17 out of the 19 corruptions were included due to limited storage space. The corruptions that were not included were ”Zoom blur” and ”Contrast”. Detailed information about the types of corruptions in the TINC dataset is shown in Appendix A. More information about the IN dataset can be found in Section 4.3.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 11: Architecture details of several models from the Resnet family. The column with the red box indicates the architecture of the Resnet18 model. The table was obtained from the original paper [45].

4.2.1 Experimental setup

For Tiny-Imagenet Resnet18 was used. The structure of various Resnet models can be found in Table 11. The structure of Resnet18 is indicated with a red box.
The Resnet18 architecture was slightly adapted to accommodate for the input size of images in the Tiny Imagenet dataset. Instead of an input layer with kernels of size 7x7, the size was reduced to 3x3. Secondly, the max-pooling layer after the input layer was removed. Lastly, the output layer contains 200 nodes since there are 200 categories in the Tiny Imagenet dataset.

Models for Tiny Imagenet were trained for 150 epochs with SGD, using a cosine learning rate scheduler with warm restarts, using $T_0 = 10$ and $T_{mult} = 2$ [44]. The starting learning rate was set at *0.01*, using a momentum of *0.9* and weight decay with value 0.0001. The starting weights were the pre-trained weights provided by torch, unless otherwise specified. For each model 5 runs were performed, and the obtained accuracies and scores were averaged over the 5 runs.
Images were normalized using $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$, for the red, green and blue channel respectively. During training data augmentation was used by flipping the the image along its horizontal axis with a probability of 50%.

4.2.2 Experiments

A baseline Resnet18 model was trained with the architecture as shown in Table 11 to compare the obtained results.
The Resnet18 model has 17 convolutional layers, and so exhaustive experimentation with every possible configuration is not feasible. Therefore, most experiments were performed in terms of Resnet blocks. Resnet18 consists of 4 blocks containing 4 convolutional layers each. Modifications were applied to the blocks as a whole, meaning to every convolutional layer in the block.
For each modification, one experiment was performed where the modification was applied to the input

layer. The effect of the modifications in the input layer is important since the rest of the network uses the outputs of the input layer for further processing.
Similarly, for each modification one experiment was performed where the modification was applied to all convolutional layers in the network.
For the first block, one experiment was performed for each convolutional layer in the block, where the layer was parametrized by the SS parametrization. These experiments were performed to see whether single-layer modification can influence performance, and whether the location of the parametrized layer within the block is important.
Lastly, for each block and each modification, an experiment was performed where the modification was applied to all layers of that block. These experiments can provide insight into the influence that the modifications have on the performance of the whole network, and whether the location of the modification changes the performances of the models.

Names of models are similar to the names from Section 4.1. However, blocks are used instead of layers to refer to the modifications. The number or text in the parentheses following the block indicates to which layers the modifications are applied. For example, B1(1) refers to the first layer of block 1, and B2(all) refers to all four layers of block 2. SS and ISS modifications are indicated with the subscript SS or ISS respectively (e.g. Resnet18-$B1(all)_{SS}$ and Resnet18-$B1(all)_{ISS}$). PO modifications are indicated the PO superscript (e.g. Resnet18-$B1(all)^{PO}$). Furthermore, the input layer will be referred to as layer 0 ($L0$). Lastly, Resnet18-ALL refers to all convolutional layers in the network, excluding the down sampling layers.

4.2.3 Results

Table 3 shows the clean and corrupted accuracy, and the ECE scores of the trained models.
Figure 12 shows a plot of the obtained corruption accuracies per corruption, for the base model and six other selected models. These six models were selected because they performed best and provide the most insight about the internal processes of the models. The corruption accuracies per corruption for every model are shown in Appendix B.2.
Figure 13 shows a plot of the ECE scores obtained by the same six models as shown in Figure 12. The ECE score plots of every model are shown in Appendix C.1.
Figure 14 shows the occlusion sensitivity for the base model and four selected models.

4.2.4 Discussion

The performance of models which have all their layers modified degrades drastically. Especially those containing the PO layers, indicating that these layers do not allow enough information to feed forward for the network to distinguish between the classes. Although the model with only PO layers in the MNIST task (Section 4.1) did not degrade as much, this is likely due to the differences between the two datasets. In the MNIST task there is little to no variation in textures, as the images are of handwritten digits in black and white. For Tiny Imagenet however, the images are often in a natural setting in color, and textures are much more abundant. Thus, small differences in illumination and activation are much more important, and since the PO layer only allows the highest activation at some location in the image to feed forward, too much texture information is lost for the models to learn to perform the task.

Model	Tiny Imagenet (↑)	Tiny Imagenet-C (↑)	ECE (↓)
Resnet18	57.49%	35.66%	0.2614
Resnet18-$L0_{SS}$	57.52%	35.92%	0.2617
Resnet18-$L0_{ISS}$	57.56%	35.65%	0.2611
Resnet18-$L0_{SS}^{PO}$	53.95%	34.47%	0.2637
Resnet18-$L0^{PO}$	54.71%	33.19%	0.2657
Resnet18-ALL_{SS}	30.49%	19.14%	0.1185
Resnet18-ALL_{ISS}	33.48%	21.46%	0.1250
Resnet18-ALL_{SS}^{PO}	9.11%	6.17%	0.0656
Resnet18-ALL^{PO}	13.41%	8.61%	0.0777
Resnet18-$B1(1)_{SS}$	57.75%	36.00%	0.2622
Resnet18-$B1(2)_{SS}$	57.72%	35.90%	0.2626
Resnet18-$B1(3)_{SS}$	57.71%	36.08%	0.2619
Resnet18-$B1(4)_{SS}$	**57.78**%	36.13%	0.2590
Resnet18-$B1(all)_{SS}$	57.34%	36.71%	0.2575
Resnet18-$B2(all)_{SS}$	56.79%	35.71%	0.2601
Resnet18-$B3(all)_{SS}$	55.06%	32.99%	0.2841
Resnet18-$B4(all)_{SS}$	56.34%	33.65%	0.2276
Resnet18-$B1(all)_{ISS}$	57.26%	36.68%	0.2564
Resnet18-$B2(all)_{ISS}$	56.58%	35.39%	0.2602
Resnet18-$B3(all)_{ISS}$	55.44%	33.40%	0.2708
Resnet18-$B4(all)_{ISS}$	56.28%	33.59%	**0.2261**
Resnet18-$B1(all)_{SS}^{PO}$	57.26%	**36.84**%	0.2551
Resnet18-$B2(all)_{SS}^{PO}$	56.02%	34.41%	0.2730
Resnet18-$B3(all)_{SS}^{PO}$	55.16%	32.75%	0.2840
Resnet18-$B4(all)_{SS}^{PO}$	56.49%	33.55%	0.2292
Resnet18-$B1(all)^{PO}$	57.13%	36.65%	0.2571
Resnet18-$B2(all)^{PO}$	56.03%	34.51%	0.2705
Resnet18-$B3(all)^{PO}$	54.99%	32.60%	0.2899
Resnet18-$B4(all)^{PO}$	56.55%	33.68%	0.2270

Table 3: Performance of models for the clean data (Tiny Imagenet), the corrupted data (Tiny Imagenet-C) and the ECE score. Performance over the clean and corrupted data is expressed as accuracy. For Tiny Imagenet-C, the accuracy is the mean accuracy over all corruptions in the Tiny Imagenet-C dataset. Values in **bold** indicate the best performing model for the clean data, the corrupted data or for the ECE score. Arrows indicates whether higher is better (↑), or lower is better (↓)

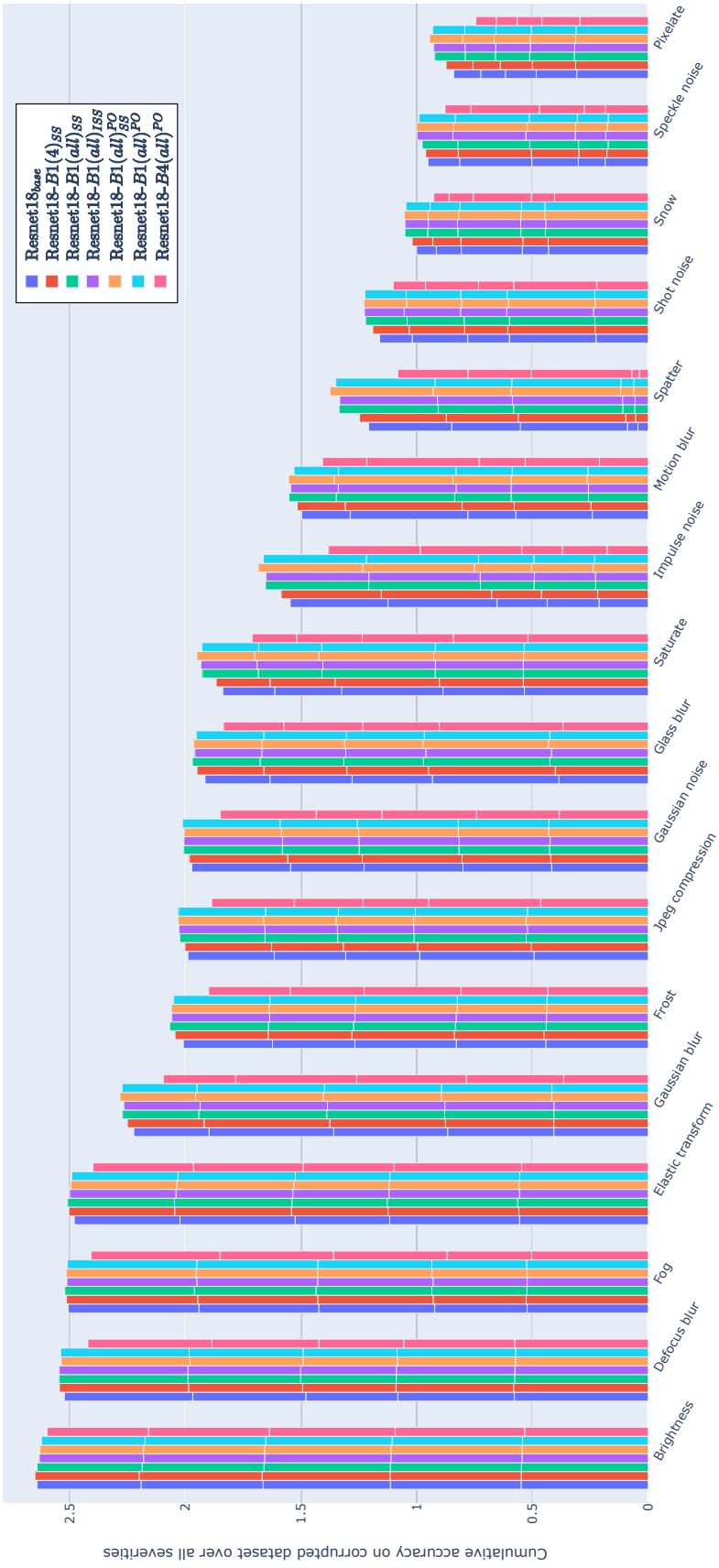

Figure 12: A barplot of the accuracies obtained per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in blue for reference. The other barplots are for models Resnet18-$B1(4)_{SS}$ (red), Resnet18-$B1(all)_{SS}$ (green), Resnet18-$B1(all)_{ISS}$ (purple), Resnet18-$B1(all)^{PO}_{SS}$ (orange), Resnet18-$B1(all)^{PO}$ (light blue) and Resnet18-$B4(all)^{PO}$ (pink).

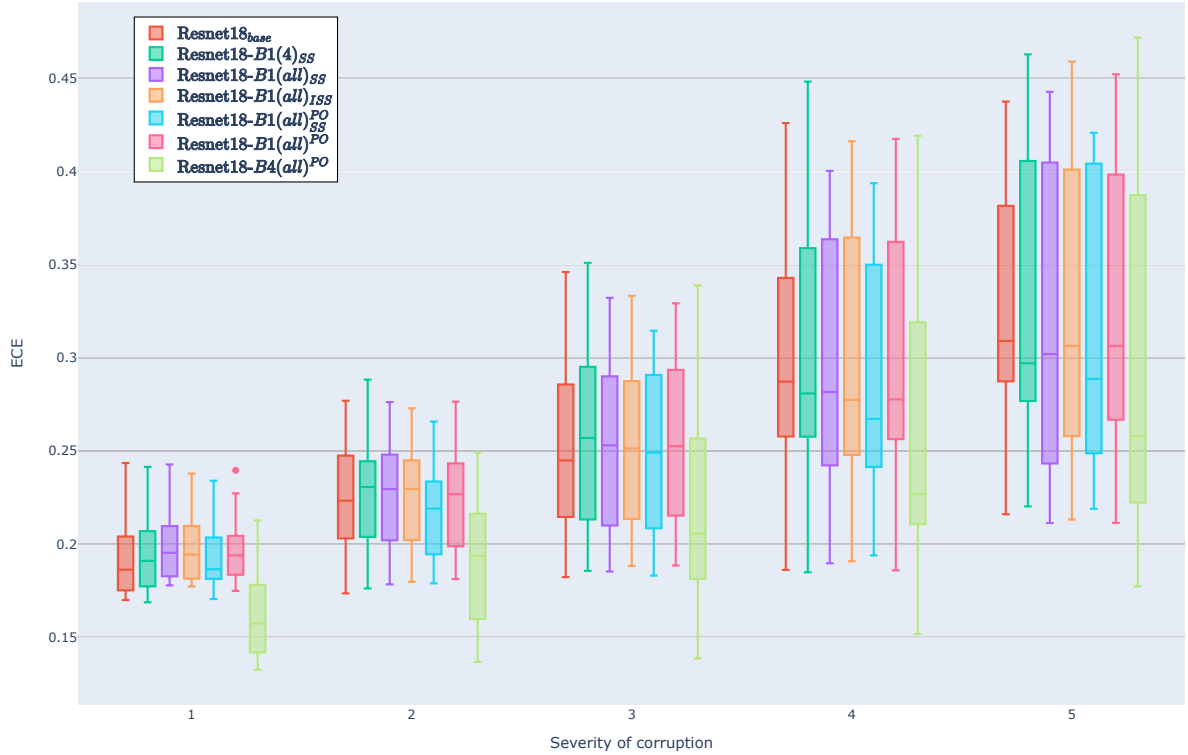

Figure 13: A boxplot of the ECE scores per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in red for reference. The other boxplots are for models Resnet18-$B1(4)_{SS}$ (green), Resnet18-$B1(all)_{SS}$ (purple), Resnet18-$B1(all)_{ISS}$ (orange), Resnet18-$B1(all)^{PO}_{SS}$ (light blue), Resnet18-$B1(all)^{PO}$ (pink) and Resnet18-$B4(all)^{PO}$ (lime).

When the modifications were not applied to the whole network but to just a single block, all modifications seem to have a similar effect on clean accuracy, robustness and the ECE score. Similar trends can be observed in Table 3 within each modification. For example, the ECE score of any modification increases as it is applied to blocks 1 through 3, but the lowest score is obtained for block 4. Similar patterns can be observed for the clean accuracy and robustness, though some anomalies exist. Thus, it seems that each modification achieves a similar way of processing the data.

Models that have just one layer parametrized in the first block perform slightly better for both the clean performance and the robustness. Out of the 4 models, model Resnet18-$B1(4)_{SS}$ performs best in all three metrics. Furthermore, it slightly performs better in all three categories in comparison to the base model. Thus, it can be beneficial for a shape selective layer to be included in a network. However, it is not clear why a single layer modification can improve the performance of the model, and more single layer modification experiments are required for a thorough understanding of the effect that this small change in architecture can have on the processing of the model.

Models with modifications in the first block, models Resnet18-$B1(all)_{SS}$, Resnet18-$B1(all)_{ISS}$, Resnet18-$B1(all)^{PO}_{SS}$ and Resnet18-$B1(all)^{PO}$, perform the best with respect to the other modifications in other blocks, regardless of the type of modification. Furthermore, the robustness of these models is slightly increased. Because all modifications obtain increased robustness, it is not obvious what the cause of this increase is. However, a common goal of the three of the modifications (SS, ISS and PO/SS) is to detect what shape is most present at some location. Though this is not true for the PO modification, as it does not have shape selective filters, the modification still detects what pattern is most present in

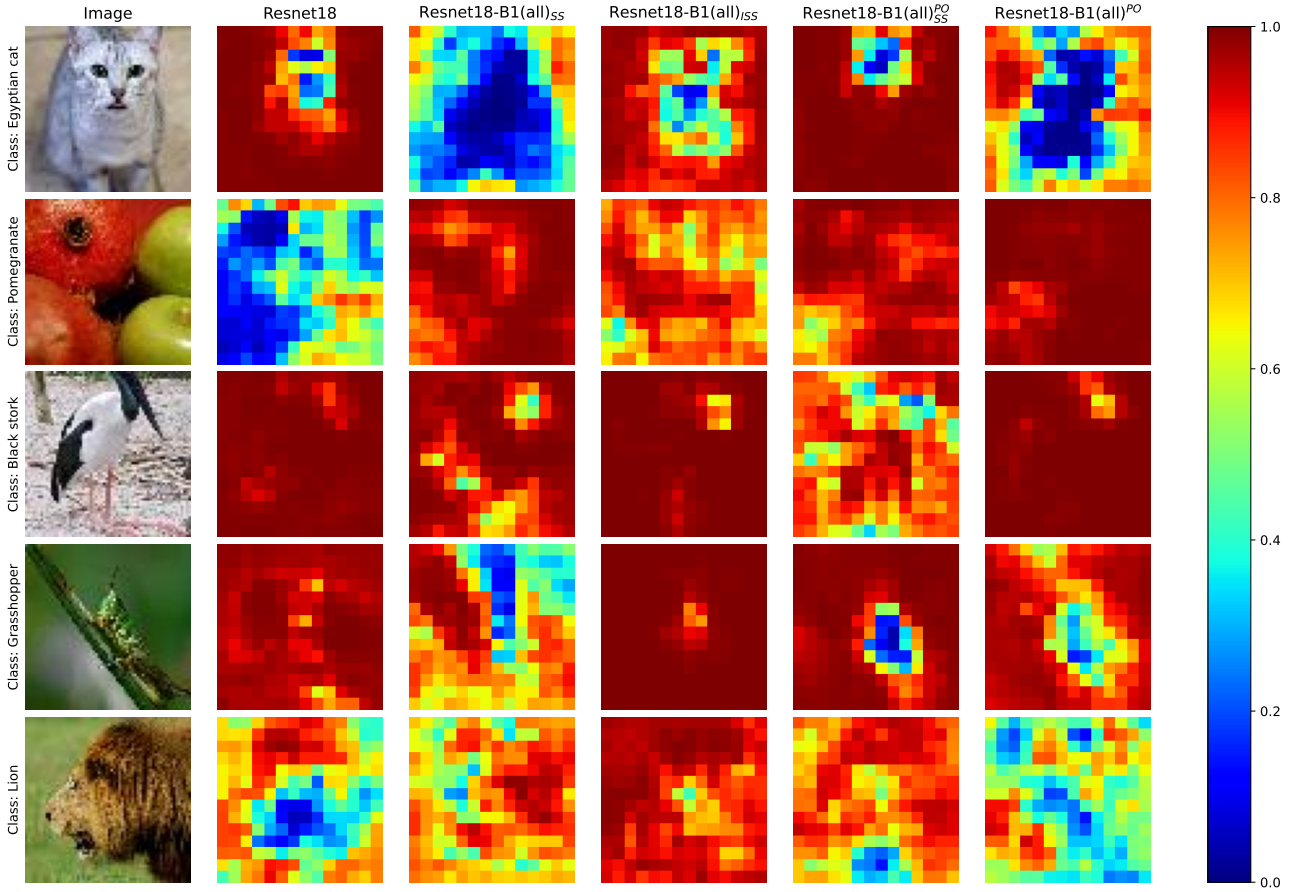

Figure 14: The occlusion sensitivity (Section 3.4.3) for five different images from the Tiny Imagenet dataset. The first column shows the original image and the label. Columns two through six show maps of the Softmax probabilities that the models assigned to the image as a function of the location of the mask. The models from left to right are Resnet18, Resnet18-B1(all)$_{SS}$, Resnet18-B1(all)$_{ISS}$, Resnet18-B1(all)$^{PO}_{SS}$ and Resnet18-B1(all)^{PO}. The mask size was 12 by 12 pixels and the stride was set to 4.

the network. This finding suggests that detection of basic shapes or features is important in the first block for classification.

Parametrization in the third block performs the worst out of parametrization in any block. Recent research suggests that representational complexity in a convolutional layer increases with layer depth, up until about 75-80% of the total depth of the network, after which representational complexity drops sharply [46, 47].
These two results together seem to suggest that the representational power of shape selective kernels is restricted by the modifications. This characteristic would explain that the largest drop in performance is observed in the third block, and that modifications in the fourth block perform similarly on the clean dataset as modifications in the second block.
Furthermore, for the modified models, clean accuracy for modifications in the fourth block is slightly lower (0.6% - 1%) compared to modifications in the first block. Moreover, robustness increases for modifications in the first block, whereas robustness decreases for modifications in the fourth block. This decrease is also larger than the decrease in clean accuracy. Thus, restriction of representational power seems to be beneficial for robustness at the beginning of the network, possibly because the

shape selective kernels are able to create more robust representations of simple shapes and edges. On the other hand, towards the end of the network shape selective kernels are not able to model the complex representations effectively, causing a decrease in performance.

For models with modification in the fourth block, ECE values decreases with respect to the base model, but so do clean accuracy and robustness. This indicates that the modifications in the fourth block create better calibrated models.
Figure 13 furthermore shows that this is true for each severity, although the spread of the values becomes comparable to the other models for severities 3, 4 and 5.
For models Resnet18-$B4(all)^{PO}$ and Resnet18-$B4(all)^{PO}_{SS}$ the reason could be due to the fact that they have limited information to work with. At the end of the network, when a class prediction needs to be made, these models only have access to the most expressive features in the data.
Moreover, most background noise is filtered, which means that the models only make predictions based on the features that contain the most information. This characteristic could improve the models certainty by allowing the model to discard features that are likely not important for the final decision.
For models Resnet18-$B4(all)_{SS}$ and Resnet18-$B4(all)_{ISS}$ it is not clear why the ECE score improves. However, these results do reinforce the idea that each modification create similar behaviour, even if the mechanisms employed are different.

As can be seen in Figure 13, models Resnet18-$B1(4)_{SS}$, Resnet18-$B1(all)_{ISS}$, Resnet18-$B1(all)^{PO}_{SS}$ and Resnet18-$B1(all)^{PO}$ obtain a small increase in accuracy for almost all corruptions. This results suggests that the general robustness of the models is affected as opposed to robustness against a single robustness or type of robustness. Although the differences are small, a method that improves general robustness is desirable because it is more likely to transfer its robustness to other unknown corruptions.
This furthermore reinforces the idea that all modifications achieve a similar effect on the processing that happens within the models, since the general robustness seems to increase similarly for each model, and there is little variation between performance of models on individual corruptions.

The occlusion sensitivity of the models shows some similarities but also some differences between the way that the models process the data. For the picture of the ”Egyptian cat”, the face is an important feature for the base model, but the whole body is important for Resnet18-B1(all)$_{SS}$ and Resnet18-B1(all)^{PO}. However, for other images the whole body or shape of the object in the picture is not necessarily important for those models.
There is not a clear pattern for any of the occlusion sensitivity maps. Furthermore, although all models predicted the true label for the original image correctly, there can be differences in how confident the model was. If the model was not very confident, a mask at any location in the image might change the decision, whereas when the model is very confident, the mask will only change the models prediction when it covers the features in the image that the mask classifies as the true label.

4.3 Imagenet

ImageNet is a large database with over 20 million colored images across more than 20000 classes. Several subsets of this dataset exist, and the most used and well known subset is usually referred to as ”Imagenet” (IN). This subset is the ”ImageNet Large Scale Visual Recognition Challenge (ILSVRC)” subset, which was used in the challenge during years 2012 to 2017. The task that the dataset is best known for is the classification task, where a model is tasked with classifying the class of the object

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 15: Architecture details of several models from the Resnet family. The column with the red box indicates the architecture of the Resnet50 model. The table was obtained from the original paper [45].

in the image. This subset is used to train models and the performance on the validation data is an important benchmark for assessing the performance of a model, and to compare different models' performances. The dataset contains just under 1.3 million images across a 1000 classes. Resolution of images varies wildly, but most early vision models rescale images to a size of 224x224.
Imagenet-C (INC) contains 19 different corruptions at 5 different severity levels. The corruptions were applied to the validation images to create the dataset. Detailed information about the types of corruptions in the INC dataset is shown in Appendix A.

4.3.1 Experimental setup

For Imagenet Resnet50 was used. The structure of various resnet models can be found in Table 15. The structure of Resnet50 is indicated with a red box.
Models for Imagenet were trained from scratch for 30 epochs with SGD using a using a cosine learning rate scheduler. The starting learning rate was set to 0.001, using a momentum of 0.9 and a weight decay of 0.00001. Due to timely and computational restrictions, only one model per modification was trained for the Imagenet dataset. Furthermore, models were not trained for more than 30 epochs since this would exceed the maximum amount of time that can be reserved for a job on the Hábrók cluster. Images were normalized using $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$, for the red, green and blue channel respectively. During training data augmentation was used by flipping the image along its horizontal and vertical axis, both with a probability of 50%. The images were then resized to a 256 by 256 pixels and cropped to the center to obtain a 224 by 224 pixel images.

4.3.2 Experiments

A baseline model was trained with the base Resnet50 architecture as shown in Table 15 to compare the obtained results.

For Resnet50, each block consists of a number of bottleneck blocks. The first block contains three bottleneck blocks, the second block four, the third block six and the fourth block 3. Each bottleneck block contains one up-sample, one standard convolutional and one down-sample layer. Both the up and down-sample layers consist of filter with the size $1 \times 1 \times N_{in}$ where N_{in} refers to the amount of channels that the layer takes in. Because of this configuration, shape-selectivity can be applied in several ways. First, parametrization of only the middle convolutional layer of each bottleneck block is most logical since these layers are the only ones that are able to discern some sort of shape, as they have a receptive field that is larger than one pixel. Since the receptive field of the up and down-sample layers are only one pixel, there cannot exist a shape in their receptive field, and so it does not make sense to talk about shape selectivity in those layers. Thus, when a bottleneck block is modified, only the middle layer will be modified, and the up and down sample layers will stay as they are.
Similar experiments as those performed for Resnet18 were performed for Resnet50. However, modifying all layers in the network was not tested since the performance for Resnet18 decreased significantly for those models. Furthermore, modifications in the input layer was not tested since there was no improvement for Resnet18.

Again, experiments were performed for single-layer modification with the SS parametrization in the first block. Similarly to the experiments from Section 4.2, the experiments were performed to see whether single-layer parametrization can influence performance, and whether the location of the parametrized layer within the block is important.
Furthermore, for each block and each modification an experiment was performed where the modification was applied to all layers of that block. These experiments can provide insight into the influence that the modifications have on the performance of the whole network, and whether the location of the modification changes the performances of the models.
Since the first block contains three bottleneck blocks, layers 2, 5 and 8 are the layers that were modified. For single-layer SS parametrization, this is indicated by the block and the layer number in parantheses (e.g. Resnet50-B1$(5)_{SS}$). When all middle layers of all bottleneck blocks in some block were modified, it is indicated by the text "All-L2" in parantheses following the block number (Resnet50-B1$(\text{All-L2})^{PO}_{SS}$). All further naming conventions are the same as in previous experiments.

4.3.3 Results

Table 4 shows the clean and corrupted accuracy, and the ECE scores of the trained models.
Figure 16 shows a plot of the obtained corruption accuracies per corruption, for the base model and six other selected models.
Figure 17 shows a plot of the ECE scores obtained by the same seven models as shown in Figure 16.
Figure 20 shows the occlusion sensitivity for the base model and four selected models.
Figure 18 shows the fraction of decisions that the Resnet50 models made when tested on the cue-conflict dataset.
Figure 19 shows the obtained accuracies for the Resnet50 models for the edge, silhouette and cue-conflict datasets.

4.3.4 Discussion

None of the models obtained a higher robustness than the base model, and only models Resnet50-B1$(2)_{SS}$ and Resnet50-B1$(8)_{SS}$ obtained a higher clean accuracy. Comparing it to the results for Tiny

[2]The ECE score is the lowest, although a lower clean accuracy also leads to lower ECE scores.

Model	Imagenet (↑)	Imagenet-C (↑)	ECE (↓)
Resnet50	54.93%	**18.43**%	0.1446
Resnet50-B1(2)$_{SS}$	54.96%	18.37%	0.1409
Resnet50-B1(5)$_{SS}$	54.88%	18.12%	0.1319
Resnet50-B1(8)$_{SS}$	**55.06**%	18.33%	0.1357
Resnet50-B1(All-L2)$_{SS}$	54.15%	18.18%	0.1396
Resnet50-B2(All-L2)$_{SS}$	53.81%	17.98%	0.1399
Resnet50-B3(All-L2)$_{SS}$	51.94%	16.17%	0.1338
Resnet50-B4(All-L2)$_{SS}$	52.42%	16.40%	0.1374
Resnet50-B1(All-L2)$_{ISS}$	54.07%	17.91%	0.1429
Resnet50-B2(All-L2)$_{ISS}$	53.79%	17.45%	0.1355
Resnet50-B3(All-L2)$_{ISS}$	51.89%	15.93%	0.1359
Resnet50-B4(All-L2)$_{ISS}$	51.91%	16.24%	0.1344
Resnet50-B1(All-L2)$^{PO}_{SS}$	53.24%	17.36%	0.1363
Resnet50-B2(All-L2)$^{PO}_{SS}$	52.47%	15.93%	0.1380
Resnet50-B3(All-L2)$^{PO}_{SS}$	47.97%	13.22%	0.1282
Resnet50-B4(All-L2)$^{PO}_{SS}$	46.53%	13.44%	**0.1098**[2]
Resnet50-B1(All-L2)^{PO}	53.42%	17.42%	0.1361
Resnet50-B2(All-L2)^{PO}	52.41%	15.99%	0.1348
Resnet50-B3(All-L2)^{PO}	48.49%	13.68%	0.1344
Resnet50-B4(All-L2)^{PO}	46.37%	13.48%	0.1206

Table 4: Performance of models for the clean data (Imagenet), the corrupted data (Imagenet-C) and the ECE score. Performance over the clean and corrupted data is expressed as accuracy. For Imagenet-C, the accuracy is the mean accuracy over all corruptions in the Imagenet-C dataset. Values in **bold** indicate the best performing model for the clean data, the corrupted data or for the ECE score. Arrows indicates whether higher is better (↑), or lower is better (↓)

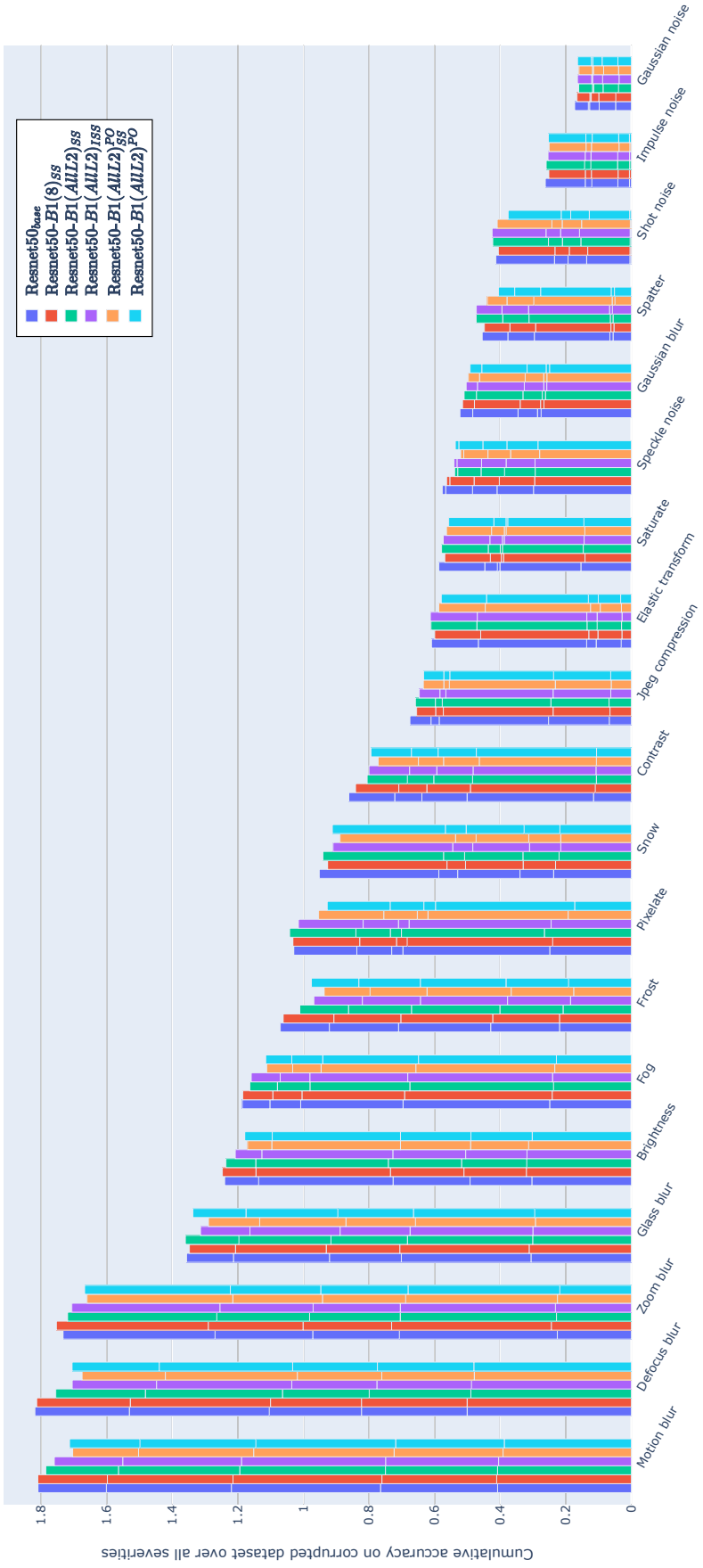

Figure 16: A barplot of the accuracies obtained per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in blue for reference. The other barplots are for models Resnet50-$B1(8)_{SS}$ (red), Resnet50-$B1(AllL2)_{SS}$ (green), Resnet50-$B1(AllL2)_{1SS}$ (purple), Resnet50-$B1(AllL2)^{PO}_{SS}$ (orange) and Resnet50-$B1(AllL2)^{PO}$ (light blue).

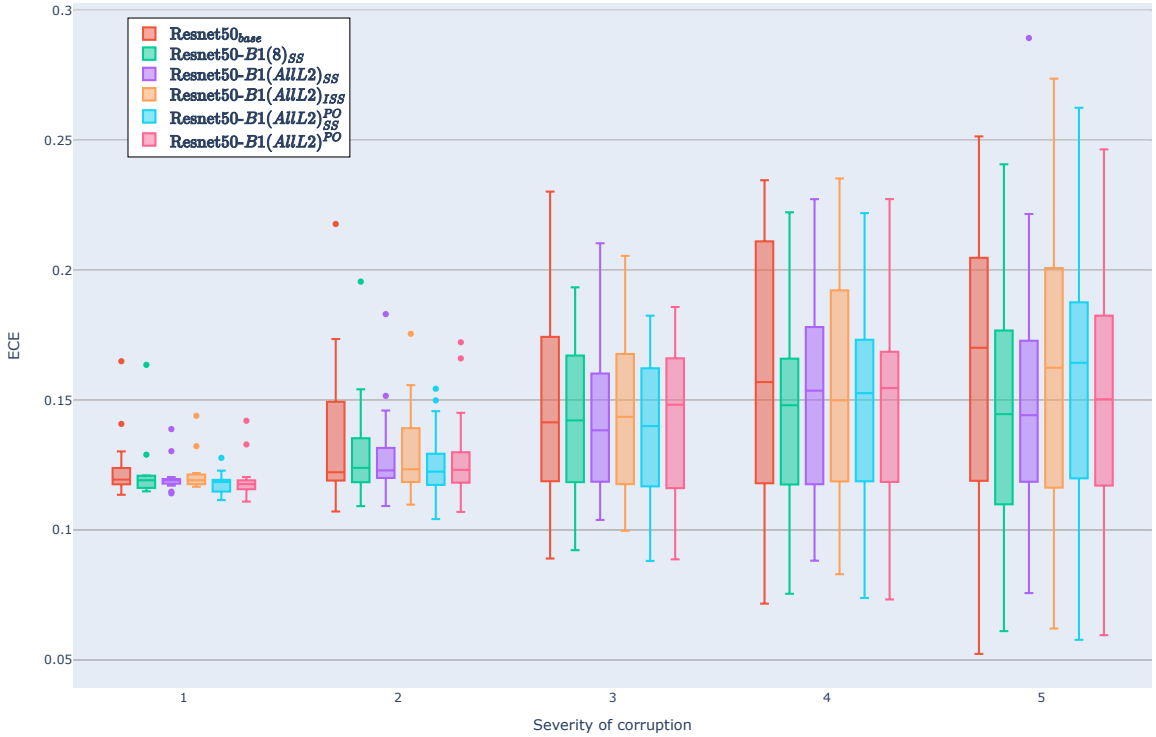

Figure 17: A boxplot of the ECE scores per corruption in the Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in red for reference. The other boxplots are for models The other barplots are for models Resnet50-$B1(8)_{SS}$ (green), Resnet50-$B1(AllL2)_{SS}$ (purple), Resnet50-$B1(AllL2)_{ISS}$ (orange), Resnet50-$B1(AllL2)^{PO}_{SS}$ (light blue) and Resnet50-$B1(AllL2)^{PO}$ (pink).

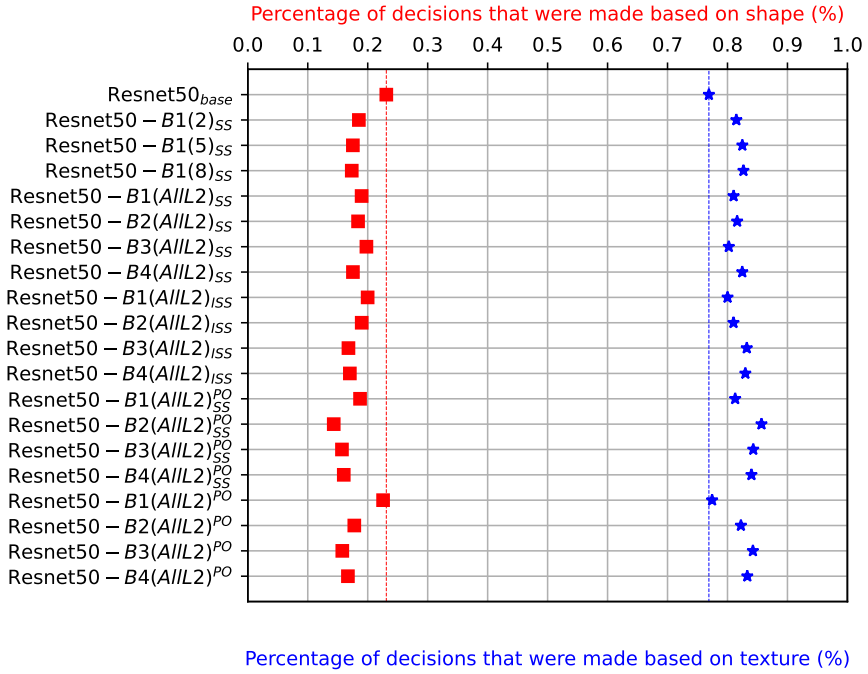

Figure 18: The fractions of texture and shape decisions that Resnet50 models made when tested on the cue conflict dataset. The red and blue dotted lines indicate the fraction of decisions of the base Resnet50 model, Resnet50$_{base}$, for the shape and texture decisions respectively.

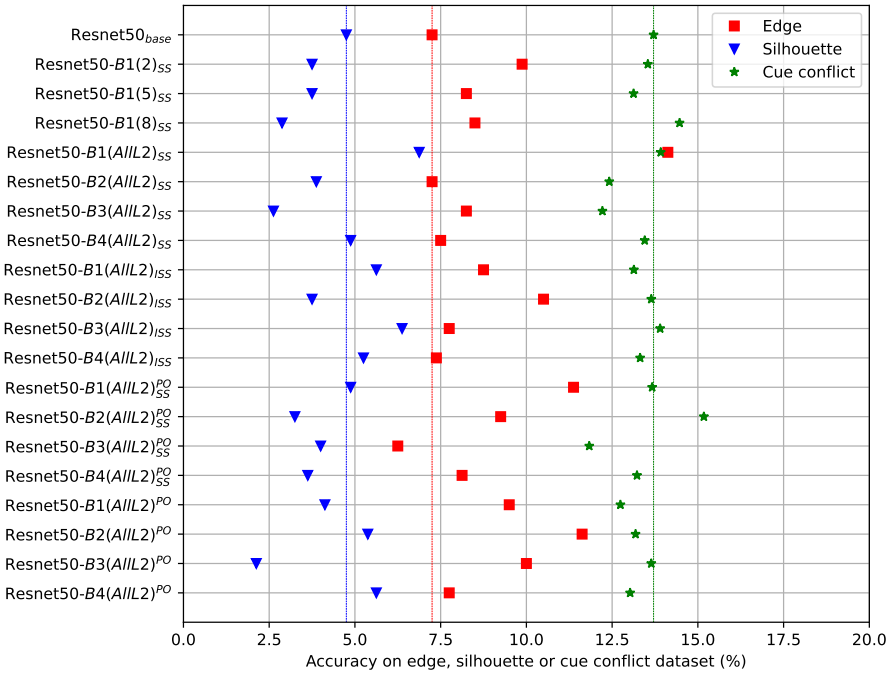

Figure 19: The accuracies on the Edge, Silhouette and Cue-Conflict datasets of all the Resnet50 models. The red, blue and green dotted lines indicate the performance of the base Resnet50 model, Resnet50$_{base}$, for the edge, silhouette and cue-conflict datasets respectively.

Imagenet (Section 4.2), this could be due to the fact that the models were not trained long enough for the loss to minimize.

For SS and ISS layers in blocks 1 through 4 a similar pattern can be seen as was found for Resnet18, where clean accuracy and robustness decreases as parametrization is introduced in layers in blocks 1 through 3, but increases again for block 4. However, as opposed to the results for Tiny-Imagenet (4.2), there are differences between the effects that the modifications have on the performance of the models.
For models with the PO layers a different pattern is found, as clean accuracy and robustness continue to decrease as PO layers are introduced into the later blocks. This large decrease in performance is likely due to the interaction between the PO layer and the bottleneck blocks. As the middle layer of each bottleneck block is modified with the PO layer, the output of the middle layer contains depth-wise one-hot encoded, which contains a lot of 0s.
Furthermore, the up-sample layers at the end of the block will compound this, as it will multiply the depth of the output by 4.
For the first out of 4 blocks, this sparsity seems to be manageable, however as the amount of filters grows, the output becomes to scarce. Furthermore, the output of the previous block will be added to the output of the block containing the PO layers. If the output is very sparse, adding it to the previous output which was not modified will impact the output map after addition very little. Thus, PO layers in later bottleneck blocks provide increasingly less information, resulting in a drop of performance as the network effectively loses one block of computation.

Models with the SS parametrization perform slightly better than models with the ISS parametrization do. For the Tiny Imagenet (Section 4.2) models a similar pattern can be observed.

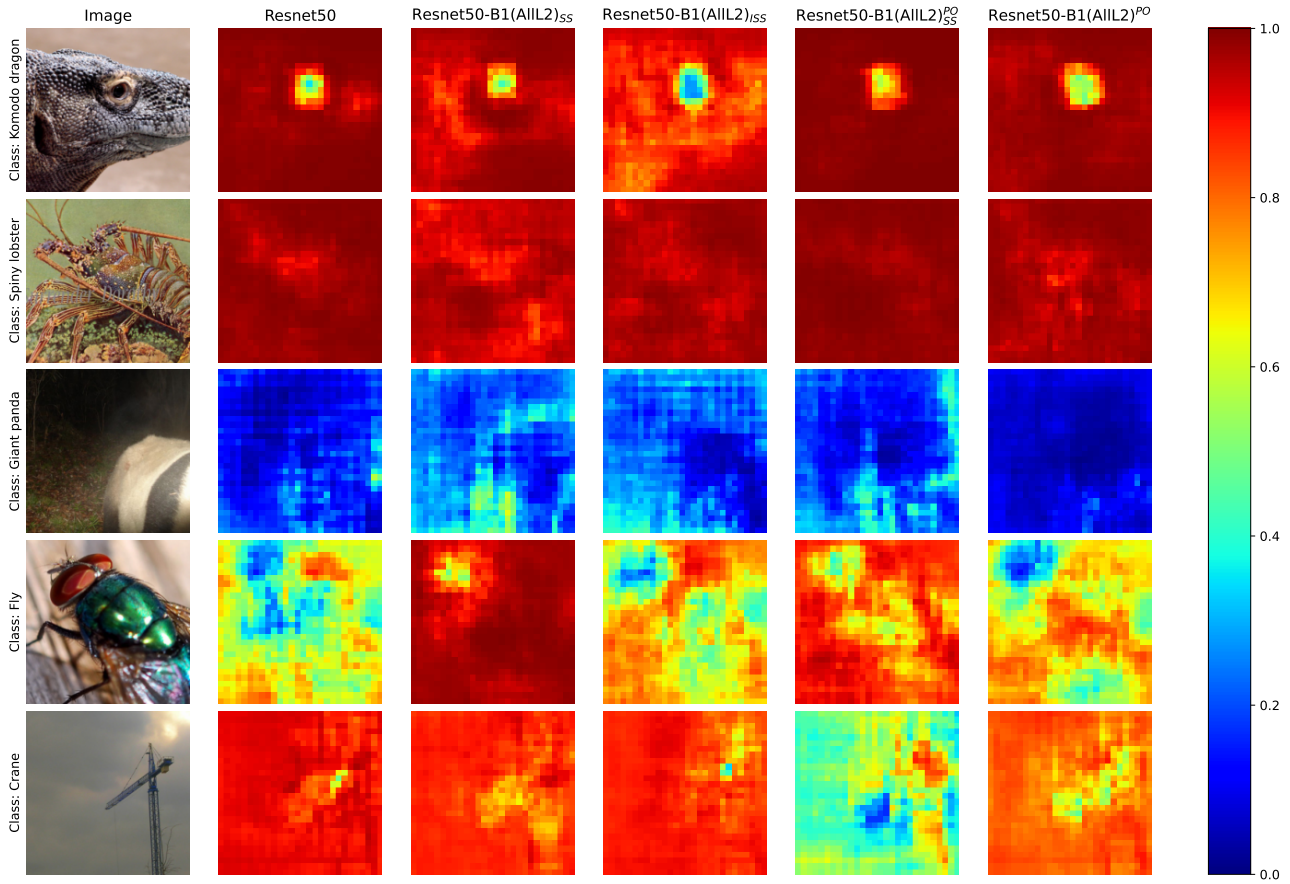

Figure 20: The occlusion sensitivity (Section 3.4.3) for five different images from the Imagenet dataset. The first column shows the original image and the label. Columns two through six show maps of the Softmax probabilities that the models assigned to the image as a function of the location of the mask. The models from left to right are Resnet50, Resnet50-B1(AllL2)$_{SS}$, Resnet50-B1(AllL2)$_{ISS}$, Resnet50-B1(AllL2)$^{PO}_{SS}$ and Resnet50-B1(AllL2)^{PO}. The mask size was 40 by 40 pixels and the stride was set to 7.

Even though the ISS parametrization restricts the parameter space of the weights less than the SS parametrization does, it does not obtain a better performance. This result indicates that the values of the positive and negative weights in the shape selective layer are not important, since a less restrictive parametrization does not improve performance.

All modified models obtained a better ECE score than the base model. Although for some models this is likely the results of a degraded clean accuracy and robustness (e.g. Resnet50-B4(All-L2)^{PO}), models Resnet50-B1(2)$_{SS}$ and Resnet50-B1(8)$_{SS}$ obtained similar accuracies as the base model while having lower ECE scores.

This results is different than the one obtained for the Tiny Imagenet models, for which only fourth block modification resulted in a large improvement of ECE scores. For Resnet50, only models with PO layers see a large improvement in ECE scores. Resnet50-B4(All-L2)$^{PO}_{SS}$, especially obtains a low score, which furthermore is not solely due to a degraded clean accuracy, as becomes obvious when the performance is compared to that of model Resnet50-B3(All-L2)$^{PO}_{SS}$.
Improvements in ECE score for the PO layers are likely due similar reasons as those discussed in

Section 4.2, namely that by selecting only the most prominent features background noise is filtered out and thus does not impact the decision made the models as much.

For model Resnet50-$B1(AllL2)^{PO}_{SS}$ and less so for model Resnet50-$B1(AllL2)^{PO}$, the performance on all blur corruptions and other corruptions like brightness, fog, frost and contrast is decreased, as can be seen in Figure 16. This makes sense as these models only feed forward the most explicit pattern, and more specifically for model Resnet50-$B1(AllL2)^{PO}_{SS}$, the most explicit shape. The previously mentioned corruptions make edges and shapes less clear, and so the recognition of a shape is more difficult. Furthermore, the difference between shapes becomes smaller and the wrong shape might get fed forward, resulting in decreased performance.

The occlusion sensitivity for all models in Figure 20 display similar patterns for all classes. Although some differences exist, there is not clear difference that can be attributed to a modification. Similar results were observed for the occlusion gradients in Sections 4.1 and 4.2.

Surprisingly, the base Resnet50 model made the most decisions based on shape out of all the models, meaning that any modified model had a higher texture bias than the base model. This is counter intuitive, since the modified models contain filters that have been specifically engineered such that they respond proportionally to the amount of shape that is present in an image. Thus, it seems that shape-selective filters do not introduce a shape bias, but rather a texture bias.
On the other hand, Figure 19 shows that only 1 model, Resnet50-$B3(AllL2)^{PO}_{SS}$, performs worse on the edge dataset than the base model. This indicates that the models are better able to deal with shape information, since the edge dataset contains images that solely contain black lines and a white background.
Thus, it seems that these modified models prefer to classify according to texture, but that they also have improved representations of classes based on shapes and edges. Though not immediately apparent, this results seems to suggest that shape selective kernels do create solid shape representations in neural networks, even when the models prefers classification based on texture representations.

5 Conclusion

The goal of the current research was to create shape selective kernels and observe what effect shape selectivity has on the performance of models, which was measured in terms of clean accuracy, robustness and uncertainty. Two types of shape selectivity were theorized and a new type of layer was proposed, the PO layer. Both shape selectivity and the PO layer impacted the performance of models in diverse ways. The change in performance depended on the dataset, the architecture used and the location of the modification within the network. Although changes in performance were relatively small, the current research showed that models with shape selective kernels achieve a different way of processing data when compared to conventional CNNs.

5.1 Future Work

Future research should focus mainly on the implementation of shape selectivity in state-of-the-art models and on complete and optimal training of modified models. Due to computational restrictions, models were not trained for the optimal amount of time, especially models trained for the Imagenet dataset.
Furthermore, the current research showed that interactions of the shape selective kernels differ depending on architecture, dataset and training recipe. Shape selectivity is easily applied to convolutional layers via parametrization, and provided that enough compute is available, future research could focus on finding the effects of shape selectivity for a range of datasets, tasks and architectures, including Vision Transformers that use convolutions.

The results regarding the PO layer show that models can work with very limited information, as long as the information is of high quality. Optimizations and simplifications can be made to the layer, which can help decrease the overhead that the layer introduces. Removing the additional activation map in favour of using the softmax or simply feeding forward just the highest activation simplifies the layer significantly, and reduces the complexity of the structure of the output, in turn possibly improving performance of the model it is implemented in.
Alternatively, research could focus on the effects limiting the amount of information that is available at each location in the output maps, similarly to what the PO layer tries to achieve, especially in the earlier layers of CNNs. The results of the current research show that the PO layer obtains a slightly worse performance but also obtains increased model calibration, even when the design of the PO layer is not optimal.

Although the texture bias was increased in the modified models, performance on images containing solely edges (the ”Edge” dataset) improved for all modified models. Architectures combining shape selective kernels with traditional ones in more sophisticated ways could produce interesting models, for example by having two different streams, one using conventional kernels and one using shape selective ones.
However, the main pursuit should lie in understanding the differences in processing between shape selective kernels and conventional ones. Both the directions of how shape selective and conventional kernels differ in their processing and what the effects of these differences in processing are are valid questions.

Bibliography

[1] R. Geirhos, K. Narayanappa, B. Mitzkus, T. Thieringer, M. Bethge, F. A. Wichmann, and W. Brendel, "Partial success in closing the gap between human and machine vision," *CoRR*, vol. abs/2106.07411, 2021.

[2] M. Ibrahim, Q. Garrido, A. Morcos, and D. Bouchacourt, "The robustness limits of sota vision models to natural variation," 2022.

[3] R. Geirhos, D. H. J. Janssen, H. H. Schütt, J. Rauber, M. Bethge, and F. A. Wichmann, "Comparing deep neural networks against humans: object recognition when the signal gets weaker," *CoRR*, vol. abs/1706.06969, 2017.

[4] R. Geirhos, C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge, and F. A. Wichmann, "Generalisation in humans and deep neural networks," *CoRR*, vol. abs/1808.08750, 2018.

[5] D. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, 1962.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.

[7] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *CoRR*, vol. abs/1903.12261, 2019.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[9] S. Wang, R. Veldhuis, C. Brune, and N. Strisciuglio, "Larger is not better: A survey on the robustness of computer vision models against common corruptions," 2023.

[10] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," 2021.

[11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *CoRR*, vol. abs/2103.14030, 2021.

[12] H. Bao, L. Dong, and F. Wei, "Beit: BERT pre-training of image transformers," *CoRR*, vol. abs/2106.08254, 2021.

[13] P. Benz, S. Ham, C. Zhang, A. Karjauv, and I. S. Kweon, "Adversarial robustness comparison of vision transformer and mlp-mixer to cnns," *CoRR*, vol. abs/2110.02797, 2021.

[14] Y. Bai, J. Mei, A. L. Yuille, and C. Xie, "Are transformers more robust than cnns?," *CoRR*, vol. abs/2111.05464, 2021.

[15] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit, "Understanding robustness of transformers for image classification," *CoRR*, vol. abs/2103.14586, 2021.

[16] C. Liu, Y. Dong, W. Xiang, X. Yang, H. Su, J. Zhu, Y. Chen, Y. He, H. Xue, and S. Zheng, "A comprehensive study on robustness of image classification models: Benchmarking and rethinking," 2023.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[18] S. Zheng, Y. Song, T. Leung, and I. J. Goodfellow, "Improving the robustness of deep neural networks via stability training," *CoRR*, vol. abs/1604.04326, 2016.

[19] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," *CoRR*, vol. abs/1905.04899, 2019.

[20] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *CoRR*, vol. abs/1805.09501, 2018.

[21] R. Poojary, R. Raina, and A. K. Mondal, "Effect of data-augmentation on fine-tuned cnn model performance," *IAES International Journal of Artificial Intelligence*, vol. 10, no. 1, p. 84, 2021.

[22] S. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. A. Mann, "Data augmentation can improve robustness," *CoRR*, vol. abs/2111.05328, 2021.

[23] S. F. Dodge and L. J. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," *CoRR*, vol. abs/1705.02498, 2017.

[24] S. F. Dodge and L. J. Karam, "Quality resilient deep neural networks," *CoRR*, vol. abs/1703.08119, 2017.

[25] Z. Wang, Y. Yang, A. Shrivastava, V. Rawal, and Z. Ding, "Towards frequency-based explanation for robust CNN," *CoRR*, vol. abs/2005.03141, 2020.

[26] J. Lukasik, P. Gavrikov, J. Keuper, and M. Keuper, "Improving native CNN robustness with filter frequency regularization," *Transactions on Machine Learning Research*, 2023.

[27] P. Roy, S. Ghosh, S. Bhattacharya, and U. Pal, "Effects of degradations on deep neural network architectures," *CoRR*, vol. abs/1807.10108, 2018.

[28] M. T. Hossain, S. W. Teng, F. Sohel, and G. Lu, "Robust image classification using a low-pass activation function and dct augmentation," *CoRR*, vol. abs/2007.09453, 2020.

[29] M. T. Hossain, S. W. Teng, D. Zhang, S. Lim, and G. Lu, "Distortion robust image classification using deep convolutional neural network with discrete cosine transform," *CoRR*, vol. abs/1811.05819, 2018.

[30] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," 2019.

[31] Y. Yang, C. Rashtchian, H. Zhang, R. Salakhutdinov, and K. Chaudhuri, "Adversarial robustness through local lipschitzness," *CoRR*, vol. abs/2003.02460, 2020.

[32] R. G. Lopes, D. Yin, B. Poole, J. Gilmer, and E. D. Cubuk, "Improving robustness without sacrificing accuracy with patch gaussian augmentation," 2019.

[33] B. Landau, L. B. Smith, and S. S. Jones, "The importance of shape in early lexical learning," *Cognitive Development*, vol. 3, no. 3, pp. 299–321, 1988.

[34] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013.

[35] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *CoRR*, vol. abs/1811.12231, 2018.

[36] N. Baker, H. Lu, G. Erlikhman, and P. J. Kellman, "Deep convolutional networks do not classify based on global object shape," *PLOS Computational Biology*, vol. 14, pp. 1–43, 12 2018.

[37] C. K. Mummadi, R. Subramaniam, R. Hutmacher, J. Vitay, V. Fischer, and J. H. Metzen, "Does enhanced shape bias improve neural network robustness to common corruptions?," *CoRR*, vol. abs/2104.09789, 2021.

[38] T. Chen, B. Li, and Y. Todo, "Orientation detection system based on edge-orientation selective neurons," *Electronics*, vol. 11, no. 23, 2022.

[39] J. Dapello, T. Marques, M. Schrimpf, F. Geiger, D. Cox, and J. J. DiCarlo, "Simulating a primary visual cortex at the front of cnns improves robustness to image perturbations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13073–13087, 2020.

[40] M. M. Taylor, M. Sedigh-Sarvestani, L. Vigeland, L. A. Palmer, and D. Contreras, "Inhibition in simple cell receptive fields is broad and off-subregion biased," *The Journal of Neuroscience : The Official Journal of the Society for Neuroscience*, vol. 38, no. 3, pp. 595–612, 2018.

[41] M. Pakdaman Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," vol. 29, Feb. 2015.

[42] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013.

[43] N. Mu and J. Gilmer, "MNIST-C: A robustness benchmark for computer vision," *CoRR*, vol. abs/1906.02337, 2019.

[44] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with restarts," *CoRR*, vol. abs/1608.03983, 2016.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[46] R. A. Janik and P. Witaszczyk, "Complexity for deep neural networks and other characteristics of deep feature representations," 2021.

[47] M. Jamroż, M. Kurdziel, and M. Opala, "A bayesian nonparametrics view into deep representations," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 1440–1450, Curran Associates, Inc., 2020.

Appendices

A Corruption Types Across Datasets

Corruption	Dataset		
	MNIST-C	Tiny Imagenet-C	Imagenet-C
Brightness	✓	✓	✓
Canny Edges	✓	✗	✗
Contrast	✗	✓	✓
Defocus Blur	✗	✓	✓
Dotted Line	✓	✗	✗
Elastic Transform	✗	✓	✓
Fog	✓	✓	✓
Frost	✗	✓	✓
Gaussian Blur	✗	✓	✓
Gaussian Noise	✗	✓	✓
Glass Blur	✓	✓	✓
Identity	✓	✗	✗
Impulse Noise	✓	✓	✓
JPEG Compression	✗	✓	✓
Motion Blur	✓	✓	✓
Pixelate	✗	✓	✓
Rotate	✓	✗	✗
Saturate	✗	✓	✓
Scale	✓	✗	✗
Shear	✓	✗	✗
Shot Noise	✓	✓	✓
Snow	✗	✓	✓
Spatter	✓	✓	✓
Speckle Noise	✗	✓	✓
Stripe	✓	✗	✗
Translate	✓	✗	✗
Zigzag	✓	✗	✗
Zoom Blur	✗	✓	✓

Table 5: A table showing which corruptions are present in which datasets.

Table 5 shows which corruption are present in the MNIST-C, Tiny Imagenet-C and Imagenet-C datasets. Corruptions in the MNIST-C dataset have just one severity, and are obtained according to the proceedings of the original paper [43]. Tiny Imagenet-C and Imagenet-C have 5 different severity

levels for each corruption. A detailed explanation of corruptions and severity levels, and the code to create both datasets can be found in the original paper [7].

B Performances per Corruption

B.1 MNIST

Table 6 shows the obtained accuracies of every model that was trained for the MNIST dataset, for every corruption. The performances shown are averages of 5 runs.

Model	Brightness	Canny Edges	Dotted Line	Fog	Glass Blur	Identity	Impulse Noise	Motion Blur	Rotate	Scale	Shear	Shot Noise	Spatter	Stripe	Translate	Zigzag
SmallNet	***86.67***	53.14	96.46	87.10	**90.11**	***97.24***	76.63	***88.28***	***91.88***	94.69	96.18	92.99	***96.21***	**93.02**	60.42	86.07
SmallNet-$L1_{SS}$	80.43	44.40	***96.47***	***87.19***	87.71	***97.24***	71.85	88.21	***91.88***	***94.82***	***96.20***	90.74	96.19	92.02	***60.53***	86.26
SmallNet-$L2_{SS}$	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10
SmallNet-$L12_{SS}$	20.15	30.33	30.65	20.54	32.55	36.24	22.15	28.65	27.42	18.68	31.99	35.84	35.15	19.50	15.23	25.53
SmallNet-$L1_{ISS}$	85.34	46.87	**96.70**	**90.78**	87.22	**97.26**	72.86	**89.10**	**92.09**	**94.90**	**96.32**	90.12	**96.23**	***92.45***	**61.15**	***86.84***
SmallNet-$L2_{ISS}$	42.36	35.76	42.36	21.71	39.57	45.25	27.95	36.90	42.30	41.97	44.48	44.15	44.37	42.81	27.29	38.14
SmallNet-$L12_{ISS}$	10.09	11.10	11.10	11.09	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10
SmallNet-$L1_{SS}^{PO}$	85.72	75.19	95.47	70.30	***89.90***	96.83	***83.53***	75.78	90.13	90.62	95.23	**95.73**	94.33	81.46	51.57	86.63
SmallNet-$L2_{SS}^{PO}$	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10
SmallNet-$L12_{SS}^{PO}$	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10	11.10
SmallNet-$L1^{PO}$	**93.87**	***75.52***	95.41	75.13	89.17	97.00	**85.32**	79.98	90.62	92.81	95.66	***95.61***	95.02	85.88	53.26	**88.21**
SmallNet-$L2^{PO}$	49.17	70.98	88.00	10.55	35.51	95.56	35.30	58.19	84.25	82.08	91.64	82.94	91.22	59.79	47.13	79.58
SmallNet-$L12^{PO}$	78.77	**77.02**	87.04	51.14	65.84	94.39	41.99	62.33	78.81	77.77	88.55	89.65	89.99	42.77	42.58	76.92

Table 6: Performance of every model for every corruption for the MNIST-C dataset. Best accuracies per corruption are indicated with **bold**, second best with ***bold italics*** and third best are underlined.

B.2 Tiny Imagenet

Table 7 shows the obtained accuracies of every model that was trained for the Tiny Imagenet dataset, for every corruption, averaged over all severities.
Figures 21, 22, 23, 24, 25, 26 and 27 show the accuracies that each model obtained for each corruption.

Model	Brightness	Defocus blur	Elastic transform	Fog	Frost	Gaussian blur	Gaussian noise	Glass blur	Impulse noise	JPEG compression	Motion blur	Pixelate	Saturate	Shot noise	Snow	Spatter	Speckle noise
Resnet18	40.75%	36.56%	51.96%	***33.18%***	35.21%	32.21%	25.53%	23.18%	25.67%	49.18%	43.05%	46.64%	34.17%	31.13%	31.56%	39.76%	25.39%
Resnet18-$L0_{SS}$	40.75%	37.47%	52.23%	32.49%	35.34%	32.59%	26.45%	24.17%	26.27%	49.73%	42.27%	46.81%	34.05%	31.99%	31.51%	40.47%	26.14%
Resnet18-$L0_{ISS}$	40.07%	36.30%	51.24%	32.12%	35.51%	31.61%	24.96%	22.93%	24.95%	49.70%	40.74%	48.83%	32.90%	30.41%	31.96%	40.32%	23.84%
Resnet18-$L0_{SS}^{PO}$	36.64%	36.83%	49.48%	27.96%	27.12%	32.44%	26.75%	25.38%	**27.75%**	49.26%	42.48%	50.99%	32.62%	31.47%	25.21%	37.31%	**27.53%**
Resnet18-$L0^{PO}$	38.82%	33.28%	48.93%	29.42%	30.01%	31.56%	25.23%	20.42%	***27.55%***	46.99%	37.70%	44.93%	**37.69%**	29.76%	27.21%	36.40%	25.24%
Resnet18-$L0_{SS}$-$B(all)(all)_{SS}$	20.47%	24.64%	30.96%	14.25%	14.67%	19.39%	13.35%	14.95%	13.61%	27.26%	25.83%	31.13%	15.54%	16.22%	13.72%	20.81%	13.40%
Resnet18-$L0_{ISS}$-$B(all)(all)_{ISS}$	22.75%	26.90%	33.38%	15.26%	16.77%	21.22%	15.90%	16.88%	15.99%	30.88%	28.77%	34.02%	16.67%	19.22%	16.22%	24.09%	16.17%
Resnet18-$L0_{SS}^{PO}$-$B(all)(all)_{SS}^{PO}$	5.48%	8.67%	9.89%	3.99%	3.46%	5.63%	4.55%	5.05%	4.45%	8.14%	9.16%	9.54%	4.32%	5.12%	3.36%	6.24%	4.11%
Resnet18-$L0^{PO}B(all)(all)^{PO}$	10.31%	11.70%	14.49%	7.59%	6.62%	8.62%	5.32%	7.38%	5.27%	12.71%	11.93%	14.50%	6.64%	6.55%	5.93%	8.94%	5.05%
Resnet18-$B1(1)_{SS}$	40.83%	37.13%	***52.50%***	32.49%	35.15%	32.22%	26.02%	23.32%	25.93%	49.46%	43.67%	47.37%	***34.43%***	31.26%	30.71%	40.01%	25.42%
Resnet18-$B1(2)_{SS}$	41.31%	36.95%	52.25%	32.46%	35.79%	32.46%	26.55%	24.03%	26.71%	50.25%	43.41%	48.71%	34.18%	32.11%	31.79%	***41.02%***	25.99%
Resnet18-$B1(3)_{SS}$	***41.37%***	38.30%	52.38%	32.99%	35.67%	32.80%	26.75%	24.92%	25.74%	50.30%	43.83%	48.62%	34.22%	31.94%	32.56%	40.39%	25.87%
Resnet18-$B1(4)_{SS}$	40.92%	36.94%	52.21%	32.43%	35.64%	32.16%	26.49%	24.27%	25.11%	50.25%	42.66%	49.62%	34.24%	32.11%	32.36%	40.35%	26.28%
Resnet18-$B1(all)_{SS}$	**41.41%**	***39.04%***	52.40%	33.05%	**36.35%**	**32.93%**	26.68%	***26.07%***	25.91%	50.10%	44.36%	50.55%	33.66%	32.06%	***32.65%***	39.70%	26.47%
Resnet18-$B2(all)_{SS}$	40.29%	37.65%	50.92%	32.11%	35.14%	31.90%	26.34%	24.45%	26.58%	49.03%	42.41%	46.58%	33.34%	31.95%	31.31%	40.17%	26.53%
Resnet18-$B3(all)_{SS}$	38.29%	34.14%	49.94%	29.44%	31.63%	29.27%	24.10%	21.55%	23.99%	47.50%	39.46%	44.92%	32.21%	29.38%	28.77%	37.30%	23.71%
Resnet18-$B4(all)_{SS}$	39.95%	34.83%	49.53%	31.40%	33.91%	29.38%	22.44%	20.05%	22.33%	46.51%	39.06%	46.46%	32.83%	27.53%	29.78%	38.28%	21.97%
Resnet18-$B1(all)_{ISS}$	40.54%	38.44%	52.15%	32.86%	***36.19%***	32.64%	***26.80%***	**26.22%**	26.05%	***50.42%***	43.99%	**51.62%**	34.12%	32.55%	**32.67%**	40.20%	26.80%
Resnet18-$B2(all)_{ISS}$	40.05%	36.87%	51.23%	31.44%	34.68%	31.07%	26.49%	23.12%	26.59%	49.05%	42.42%	48.05%	32.94%	32.34%	30.49%	40.51%	26.97%
Resnet18-$B3(all)_{ISS}$	37.50%	35.05%	50.19%	29.78%	31.95%	30.69%	24.65%	23.19%	23.90%	47.53%	40.59%	45.76%	30.99%	29.89%	28.83%	38.07%	23.78%
Resnet18-$B4(all)_{ISS}$	39.61%	35.61%	49.88%	32.63%	34.42%	31.27%	22.97%	20.98%	21.92%	46.86%	41.01%	44.06%	33.42%	28.56%	30.12%	37.82%	22.58%
Resnet18-$B1(all)_{SS}^{PO}$	40.48%	**39.72%**	52.37%	31.29%	35.36%	***32.82%***	**27.35%**	25.96%	26.93%	50.00%	**45.09%**	50.94%	33.22%	***32.64%***	31.38%	40.19%	***27.10%***
Resnet18-$B2(all)_{SS}^{PO}$	39.45%	36.09%	51.34%	30.86%	33.50%	30.85%	25.11%	22.32%	24.34%	48.49%	42.17%	46.08%	32.68%	30.32%	30.21%	39.69%	24.10%
Resnet18-$B3(all)_{SS}^{PO}$	37.65%	34.86%	49.71%	28.63%	31.13%	30.49%	23.09%	21.39%	22.89%	47.06%	40.15%	44.40%	31.65%	28.42%	28.22%	36.80%	21.92%
Resnet18-$B4(all)_{SS}^{PO}$	39.61%	35.65%	50.59%	32.00%	34.15%	30.64%	22.24%	20.94%	22.64%	46.85%	40.35%	45.35%	32.51%	27.69%	30.18%	39.16%	21.85%
Resnet18-$B1(all)^{PO}$	40.77%	38.61%	**52.52%**	**33.20%**	35.38%	32.57%	**27.35%**	24.84%	26.77%	**50.66%**	***44.35%***	***51.31%***	33.28%	**32.99%**	32.54%	**41.33%**	26.93%
Resnet18-$B2(all)^{PO}$	39.22%	36.50%	51.18%	31.13%	33.84%	31.02%	24.20%	22.74%	24.38%	47.92%	42.10%	45.52%	32.27%	29.36%	30.20%	39.11%	23.53%
Resnet18-$B3(all)^{PO}$	37.86%	33.67%	49.06%	29.66%	31.67%	29.06%	22.72%	20.63%	22.00%	47.68%	39.03%	45.32%	31.61%	28.18%	28.44%	37.49%	21.97%
Resnet18-$B4(all)^{PO}$	40.06%	35.22%	50.08%	31.75%	34.45%	30.49%	22.35%	20.75%	22.62%	47.25%	39.77%	45.88%	33.06%	28.02%	29.76%	38.98%	21.97%

Table 7: Performance of every model for every corruption for the Tiny Imagenet-C dataset. Best accuracies per corruption are indicated with **bold**, second best with ***bold italics*** and third best are underlined.

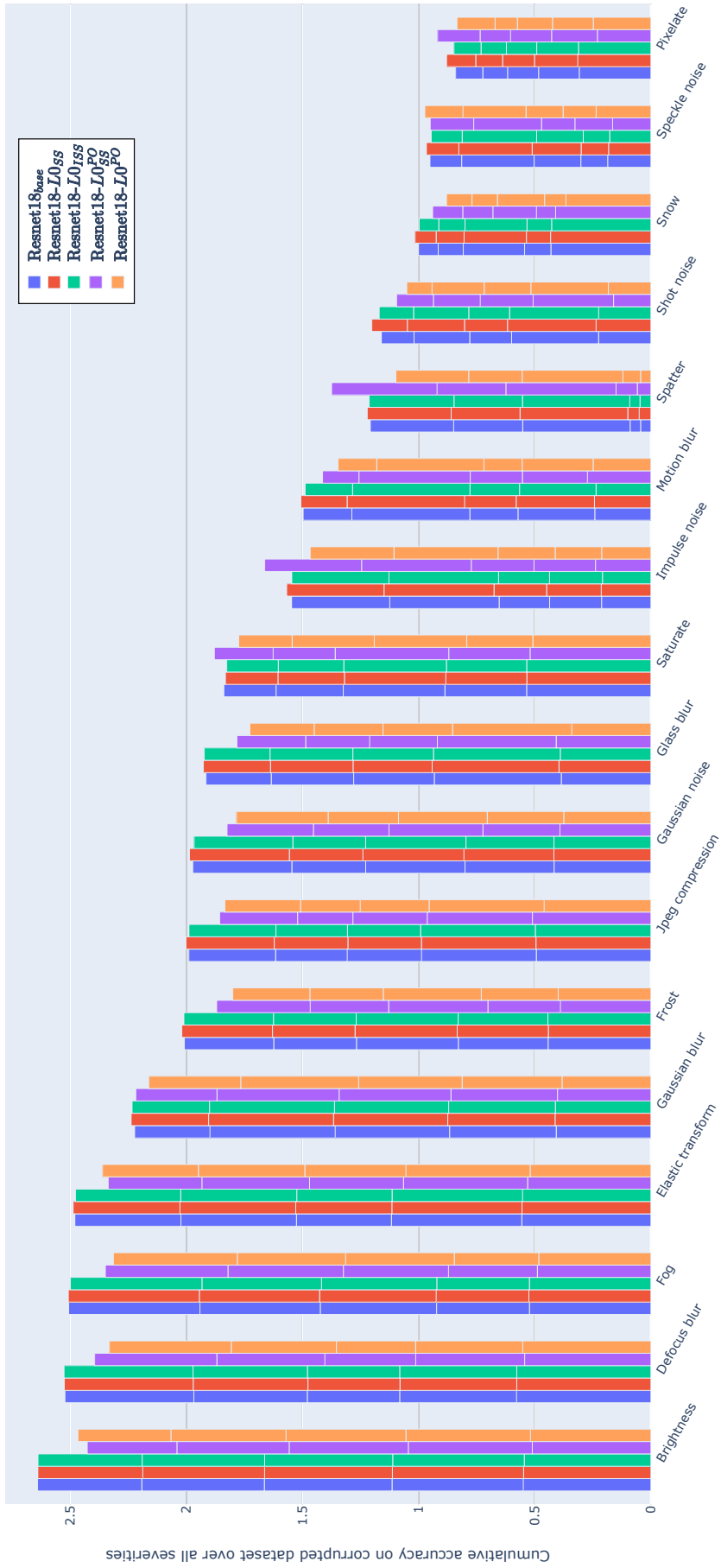

Figure 21: A barplot of the accuracies obtained per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in blue for reference. The other barplots are for models Resnet18-$I0_{SS}$ (red), Resnet18-$I0_{ISS}$ (green), Resnet18-$I0_{SS}^{PO}$ (purple) and Resnet18-$I0^{PO}$ (yellow).

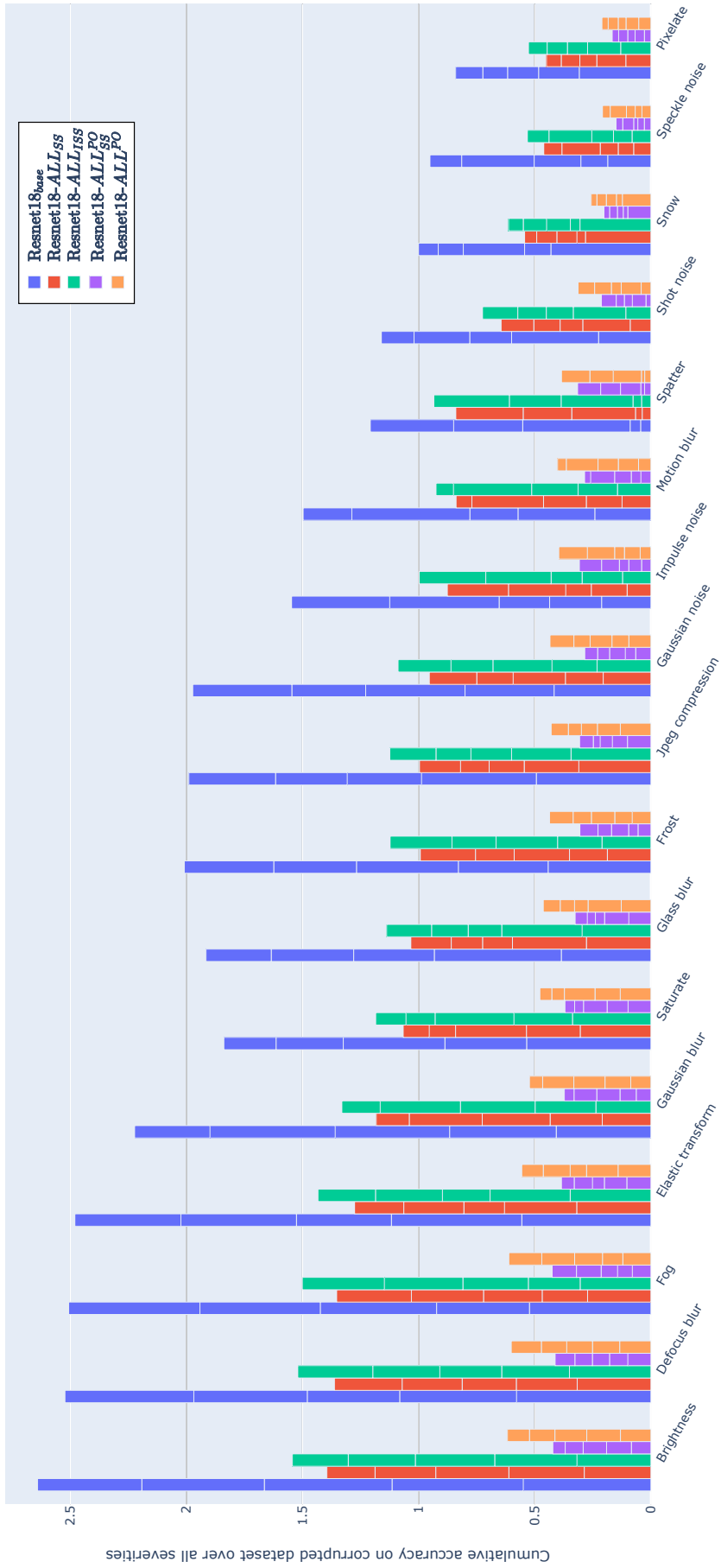

Figure 22: A barplot of the accuracies obtained per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in blue for reference. The other barplots are for models Resnet18-ALL$_{SS}$ (red), Resnet18-ALL$_{ISS}$ (green), Resnet18-ALL$^{PO}_{SS}$ (purple) and Resnet18-ALL^{PO} (yellow).

Figure 23: A barplot of the accuracies obtained per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in blue for reference. The other barplots are for models Resnet18-$B1(1)_{SS}$ (red), Resnet18-$B1(2)_{SS}$ (green), Resnet18-$B1(3)_{SS}$ (purple) and Resnet18-$B1(4)_{SS}$ (yellow).

Figure 24: A barplot of the accuracies obtained per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in blue for reference. The other barplots are for models Resnet18-$B1(all)_{SS}$ (red), Resnet18-$B2(all)_{SS}$ (green), Resnet18-$B3(all)_{SS}$ (purple) and Resnet18-$B4(all)_{SS}$ (yellow).

Figure 25: A barplot of the accuracies obtained per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in blue for reference. The other barplots are for models Resnet18-$B1(all)_{ISS}$ (red), Resnet18-$B2(all)_{ISS}$ (green), Resnet18-$B3(all)_{ISS}$ (purple) and Resnet18-$B4(all)_{ISS}$ (yellow).

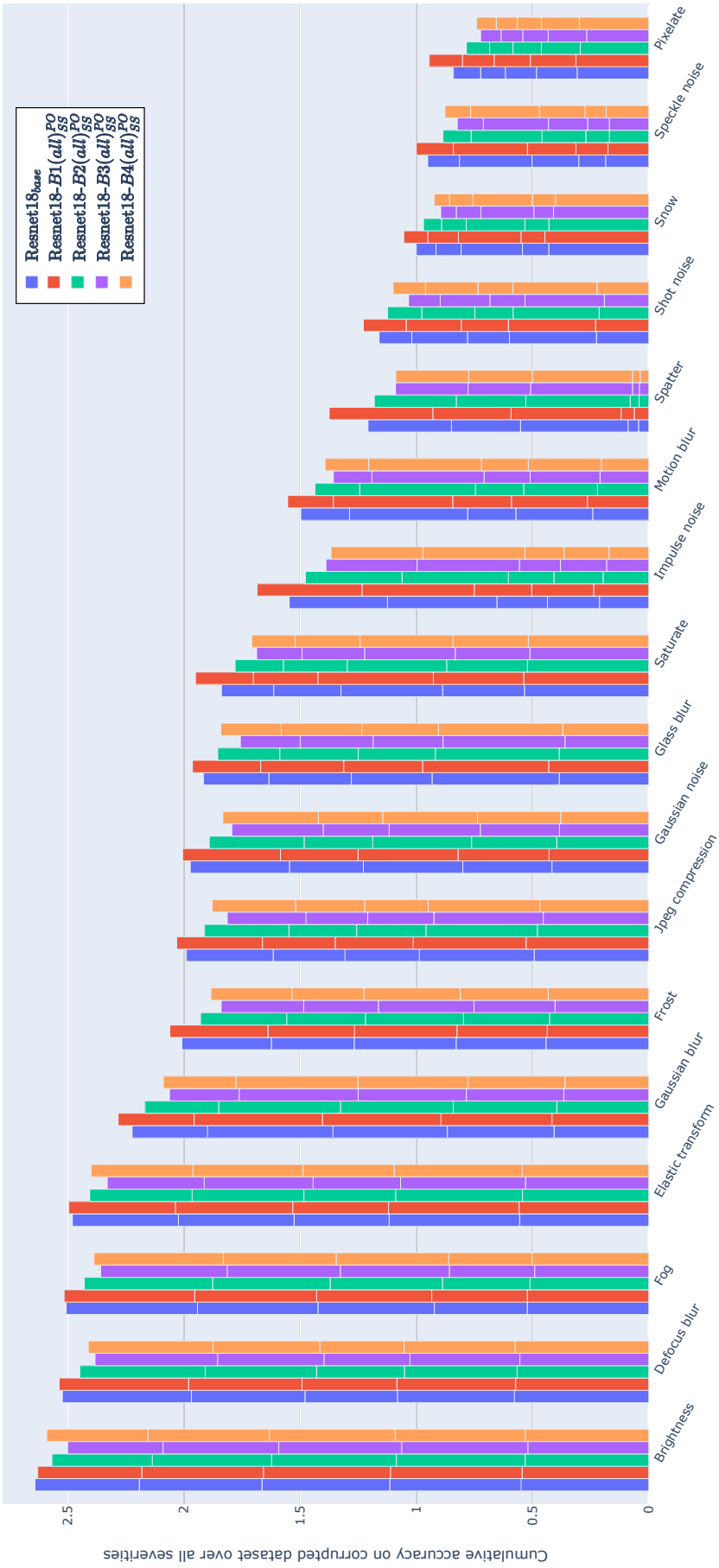

Figure 26: A barplot of the accuracies obtained per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in blue for reference. The other barplots are for models Resnet18-$B1(all)^{PO}_{SS}$ (red), Resnet18-$B2(all)^{PO}_{SS}$ (green), Resnet18-$B3(all)^{PO}_{SS}$ (purple) and Resnet18-$B4(all)^{PO}_{SS}$ (yellow).

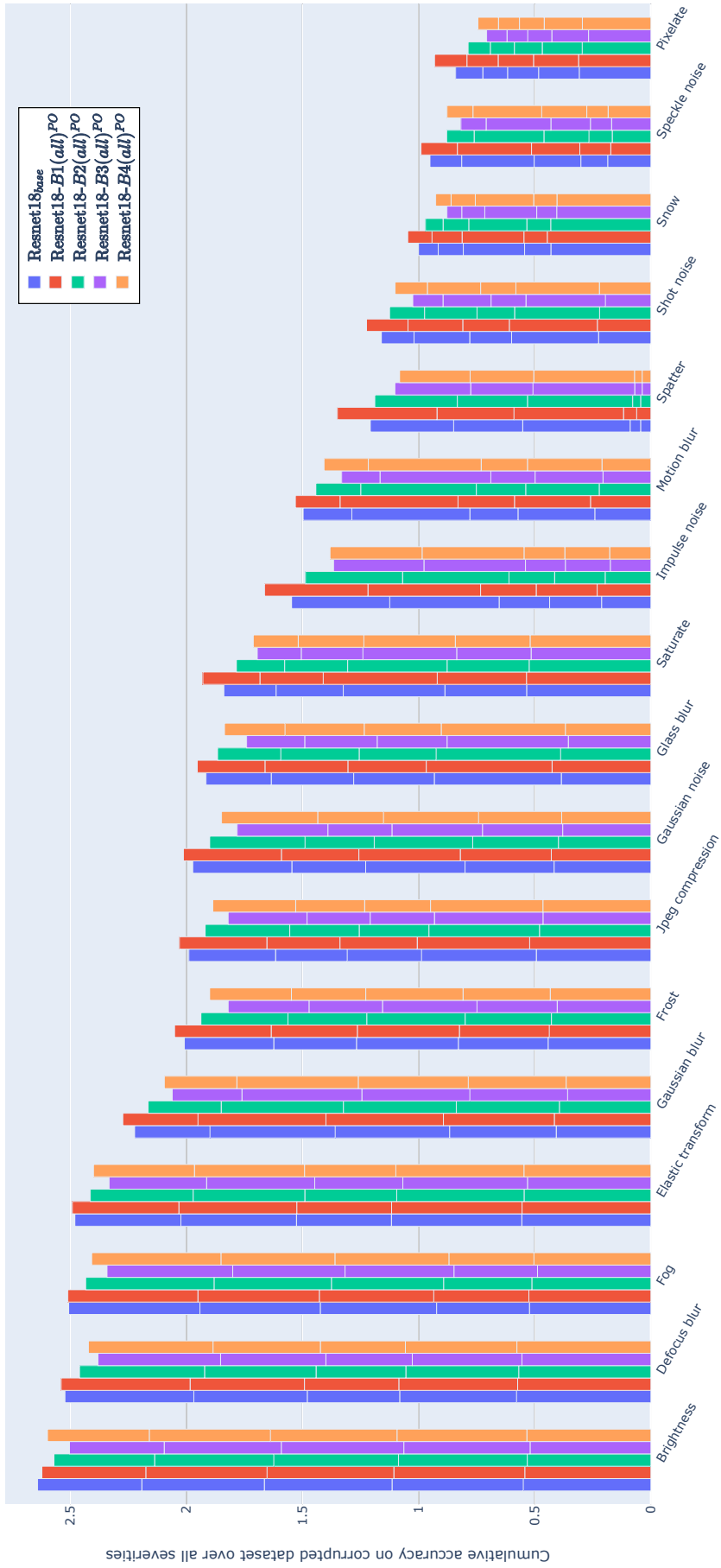

Figure 27: A barplot of the accuracies obtained per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in blue for reference. The other barplots are for models Resnet18-$B1(all)^{PO}$ (red), Resnet18-$B2(all)^{PO}$ (green), Resnet18-$B3(all)^{PO}$ (purple) and Resnet18-$B4(all)^{PO}$ (yellow).

B.3 Imagenet

Table 8 shows the accuracies obtained per corruption, for every model.
Figures 28, 29, 30, 31 and 32 show the accuracies that each model obtained for each corruption.

Model	Defocus blur	Glass blur	Motion blur	Zoom blur	Frost	Snow	Fog	Brightness	Contrast	Elastic transform	Pixelate	JPEG compression	Speckle noise	Spatter	Gaussian blur	Saturate	Gaussian noise	Shot noise	Impulse noise
Resnet50	**14.18**%	**13.13**%	***17.73***%	**19.87**%	13.27%	10.22%	**15.92**%	39.10%	**13.56**%	**30.64**%	25.41%	29.30%	11.59%	23.90%	**17.67**%	**31.90**%	7.34%	7.07%	4.88%
Resnet50-B1(2)$_{SS}$	13.74%	***13.05***%	**17.85**%	***19.68***%	***13.60***%	10.18%	15.46%	**39.21**%	13.14%	30.11%	***25.78***%	28.91%	11.54%	**24.64**%	17.19%	***31.75***%	7.38%	7.07%	4.87%
Resnet50-B1(5)$_{SS}$	13.08%	11.64%	16.84%	18.11%	12.84%	10.42%	14.83%	38.95%	12.99%	29.56%	**25.94**%	29.08%	**12.02**%	***24.46***%	16.53%	31.44%	***7.74***%	**7.42**%	5.29%
Resnet50-B1(8)$_{SS}$	13.65%	12.36%	17.33%	18.79%	**13.61**%	**10.90**%	***15.55***%	***39.14***%	***13.48***%	***30.45***%	25.31%	29.26%	11.61%	**24.64**%	17.09%	31.57%	7.31%	7.00%	4.77%
Resnet50-B1(All-L2)$_{SS}$	13.23%	12.62%	17.11%	18.96%	12.97%	10.27%	14.33%	38.53%	12.74%	30.02%	25.43%	**30.20**%	***12.00***%	24.42%	16.49%	30.85%	7.73%	***7.39***%	***5.41***%
Resnet50-B2(All-L2)$_{SS}$	***14.05***%	12.73%	17.57%	18.65%	13.01%	9.81%	14.95%	38.60%	13.07%	29.96%	25.02%	29.03%	10.91%	23.99%	***17.37***%	30.98%	6.85%	6.49%	4.41%
Resnet50-B3(All-L2)$_{SS}$	11.30%	11.05%	14.50%	17.31%	11.84%	8.63%	13.15%	35.71%	11.55%	26.66%	24.21%	26.06%	10.05%	21.88%	14.57%	28.35%	6.37%	6.01%	4.59%
Resnet50-B4(All-L2)$_{SS}$	12.08%	11.14%	15.37%	17.48%	12.08%	9.37%	13.44%	36.41%	11.99%	26.91%	22.06%	25.88%	10.59%	22.16%	15.38%	29.28%	6.36%	6.14%	4.11%
Resnet50-B1(All-L2)$_{ISS}$	13.21%	12.50%	17.33%	18.80%	13.12%	***10.46***%	13.41%	38.18%	11.58%	29.78%	24.97%	***30.12***%	11.35%	24.28%	16.38%	30.32%	6.99%	6.92%	5.11%
Resnet50-B2(All-L2)$_{ISS}$	13.03%	12.04%	16.63%	17.95%	12.63%	9.89%	14.36%	37.74%	12.42%	29.04%	24.25%	28.11%	10.92%	23.26%	16.36%	30.61%	6.85%	6.62%	4.30%
Resnet50-B3(All-L2)$_{ISS}$	11.60%	10.28%	14.42%	16.66%	11.30%	8.43%	12.92%	35.77%	11.62%	26.21%	23.64%	25.75%	10.05%	21.47%	14.89%	28.01%	6.10%	5.85%	4.04%
Resnet50-B4(All-L2)$_{ISS}$	11.46%	11.08%	15.18%	17.37%	11.23%	8.53%	12.79%	35.57%	11.14%	27.54%	23.55%	26.62%	10.37%	22.01%	14.71%	28.83%	6.45%	6.18%	4.04%
Resnet50-B1(All-L2)$^{PO}_{SS}$	13.13%	11.59%	16.97%	18.24%	12.47%	9.93%	13.29%	37.58%	11.09%	28.67%	22.72%	29.04%	10.85%	23.97%	16.16%	29.84%	7.06%	6.69%	5.21%
Resnet50-B2(All-L2)$^{PO}_{SS}$	11.89%	10.51%	14.42%	16.91%	11.10%	8.26%	12.61%	35.74%	10.57%	25.87%	22.31%	25.85%	10.55%	21.75%	15.02%	28.63%	6.29%	6.18%	3.86%
Resnet50-B3(All-L2)$^{PO}_{SS}$	8.68%	8.60%	11.48%	14.07%	9.61%	6.68%	10.68%	31.04%	9.35%	22.00%	18.29%	21.74%	8.60%	18.60%	11.64%	23.94%	4.77%	4.85%	3.47%
Resnet50-B4(All-L2)$^{PO}_{SS}$	9.15%	8.60%	11.55%	13.83%	9.16%	7.06%	10.88%	31.44%	9.39%	21.86%	19.11%	22.54%	8.54%	19.04%	12.03%	24.43%	4.75%	4.77%	3.65%
Resnet50-B1(All-L2)^{PO}	13.08%	12.38%	16.67%	18.48%	12.84%	9.83%	14.12%	37.30%	11.86%	29.15%	21.97%	27.63%	11.42%	23.52%	16.24%	29.62%	**7.76**%	7.22%	**5.42**%
Resnet50-B2(All-L2)^{PO}	11.88%	10.48%	15.05%	16.77%	11.31%	8.78%	14.08%	35.87%	11.40%	27.11%	21.56%	26.05%	9.54%	21.77%	15.09%	28.84%	5.60%	5.48%	3.41%
Resnet50-B3(All-L2)^{PO}	9.39%	8.97%	11.95%	14.88%	9.21%	7.17%	10.44%	32.20%	9.43%	23.04%	19.50%	23.48%	8.28%	19.05%	12.51%	25.37%	4.72%	4.58%	2.75%
Resnet50-B4(All-L2)^{PO}	9.48%	8.58%	12.41%	14.59%	9.57%	7.06%	10.93%	31.09%	9.83%	22.27%	19.79%	22.34%	8.07%	18.82%	12.18%	24.36%	4.56%	4.49%	3.14%

Table 8: Performance of every model for every corruption for the Imagenet-C dataset. Best accuracies per corruption are indicated with **bold**, second best with ***bold italics*** and third best are underlined.

Figure 28: A barplot of the accuracies obtained per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in blue for reference. The other barplots are for models Resnet50-$B1(2)_{SS}$ (red), Resnet50-$B1(5)_{SS}$ (green) and Resnet50-$B1(8)_{SS}$ (purple).

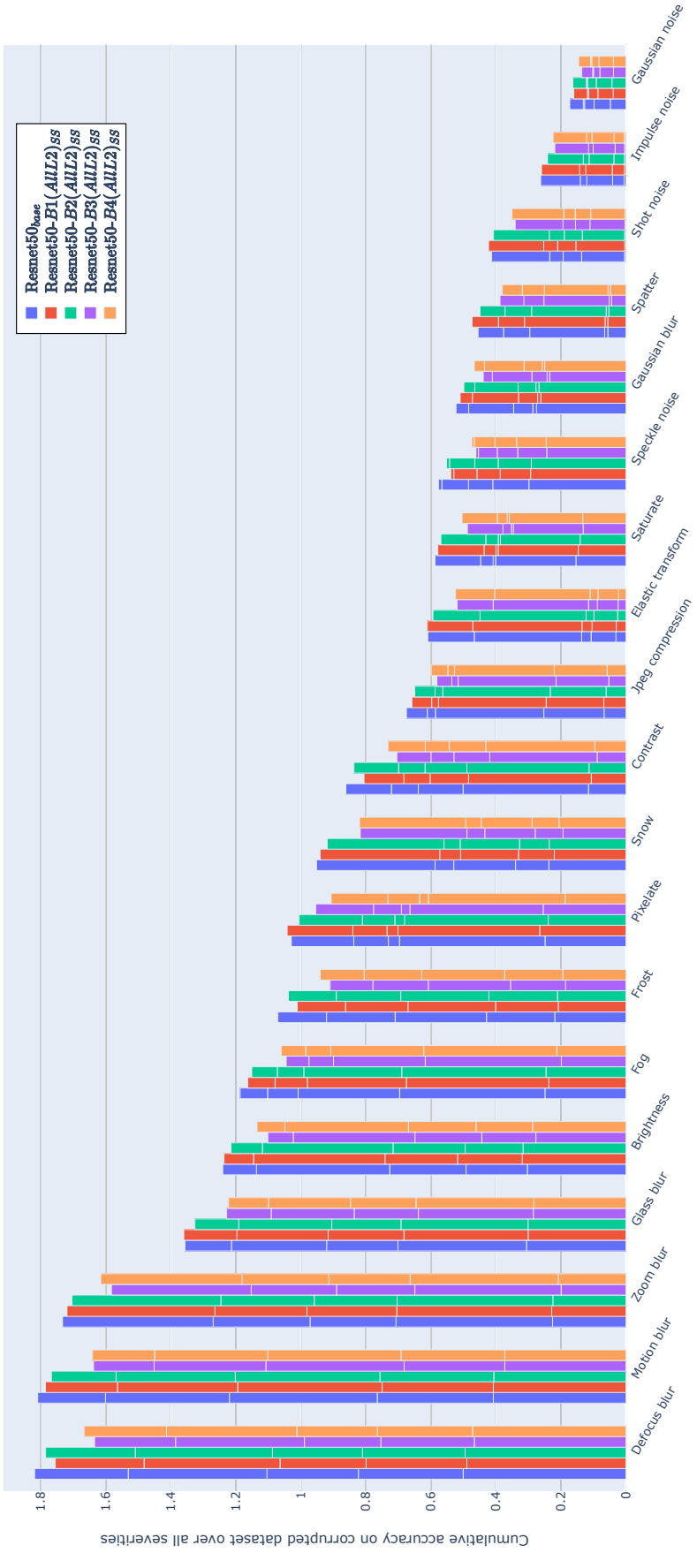

Figure 29: A barplot of the accuracies obtained per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in blue for reference. The other barplots are for models Resnet50-$B1(AllL2)_{SS}$ (red), Resnet50-$B2(AllL2)_{SS}$ (green), Resnet50-$B3(AllL2)_{SS}$ (purple) and Resnet50-$B4(AllL2)_{SS}$ (yellow).

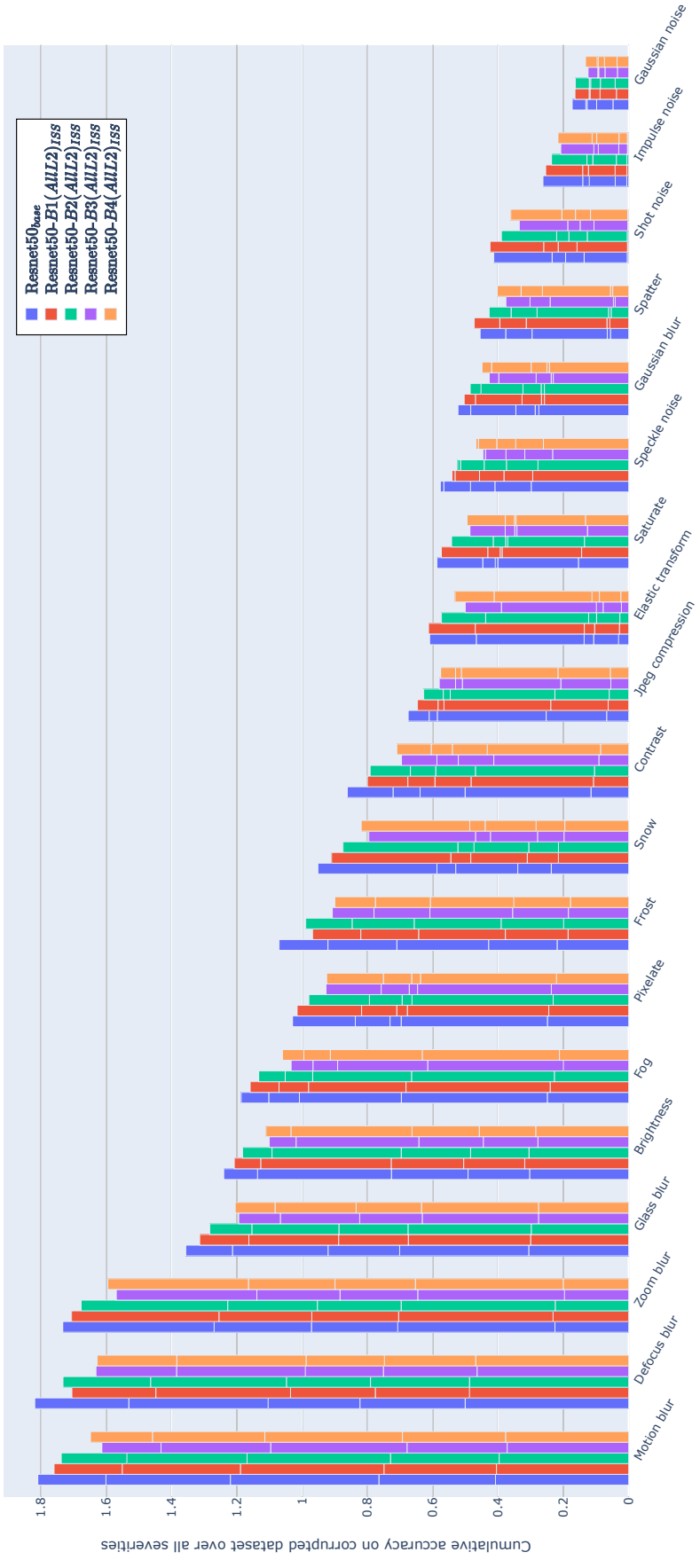

Figure 30: A barplot of the accuracies obtained per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in blue for reference. The other barplots are for models Resnet50-$B1(AllL2)_{ISS}$ (red), Resnet50-$B2(AllL2)_{ISS}$ (green), Resnet50-$B3(AllL2)_{ISS}$ (purple) and Resnet50-$B4(AllL2)_{ISS}$ (yellow).

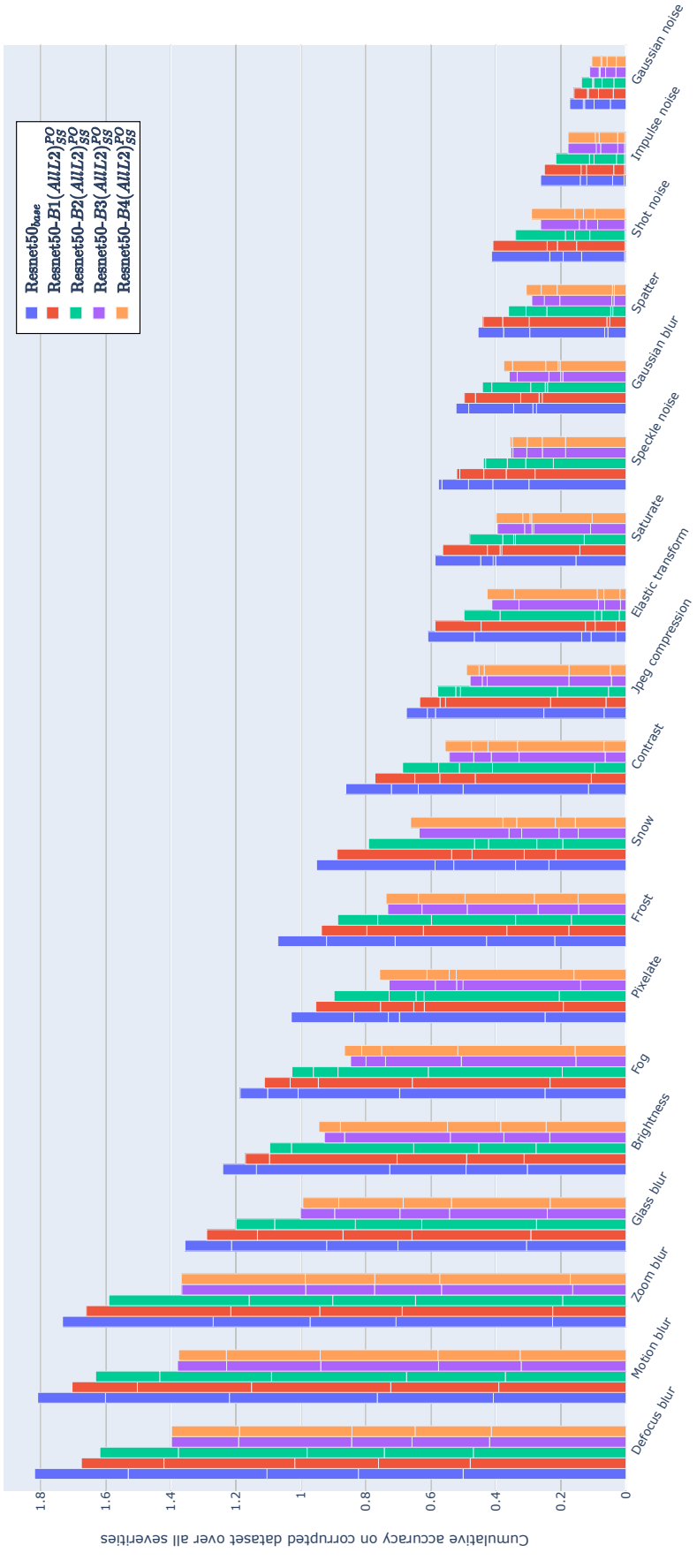

Figure 31: A barplot of the accuracies obtained per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in blue for reference. The other barplots are for models Resnet50-$B1(AllL2)^{PO}_{SS}$ (red), Resnet50-$B2(AllL2)^{PO}_{SS}$ (green), Resnet50-$B3(AllL2)^{PO}_{SS}$ (purple) and Resnet50-$B4(AllL2)^{PO}_{SS}$ (yellow).

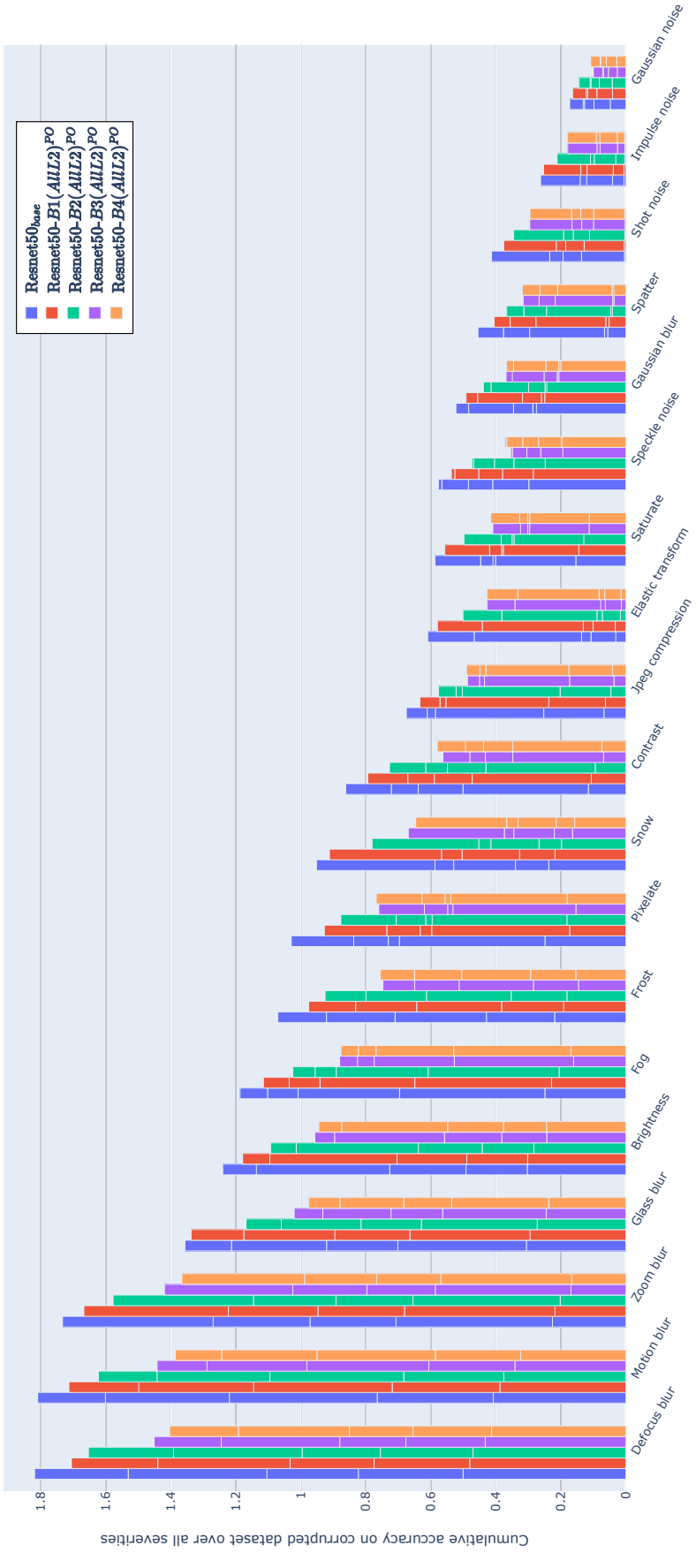

Figure 32: A barplot of the accuracies obtained per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in blue for reference. The other barplots are for models Resnet50-$B1(AllL2)^{PO}$ (red), Resnet50-$B2(AllL2)^{PO}$ (green), Resnet50-$B3(AllL2)^{PO}$ (purple) and Resnet50-$B4(AllL2)^{PO}$ (yellow).

C ECE scores

C.1 Tiny Imagenet

Figures 33, 34, 35, 36, 37, 38, 39 show the ECE scores that each model obtained for each severity in the Tiny Imagenet-C dataset.

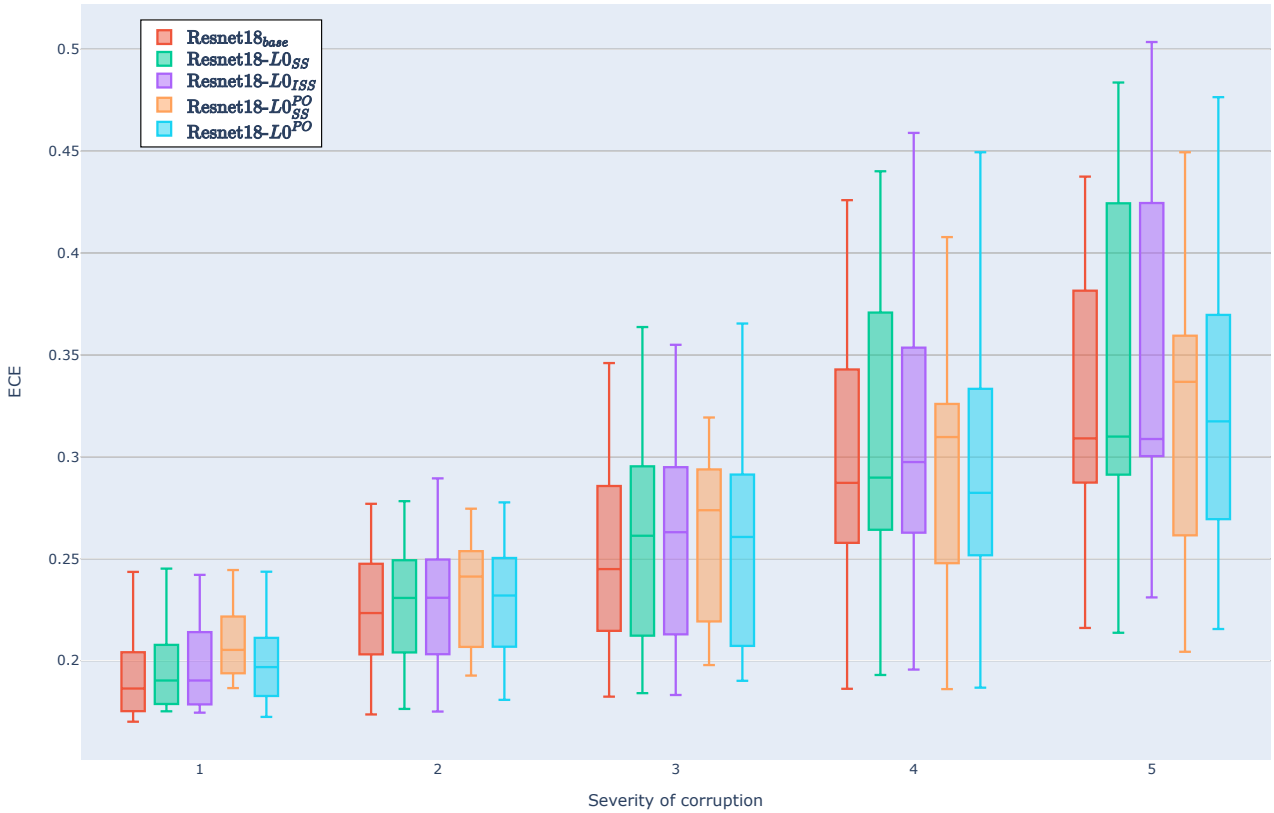

Figure 33: A boxplot of the ECE scores per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in red for reference. The other boxplots are for models Resnet18-$l0_{SS}$ (green), Resnet18-$l0_{ISS}$ (purple), Resnet18-$l0^{PO}_{SS}$ (yellow) and Resnet18-$l0^{PO}$ (light blue).

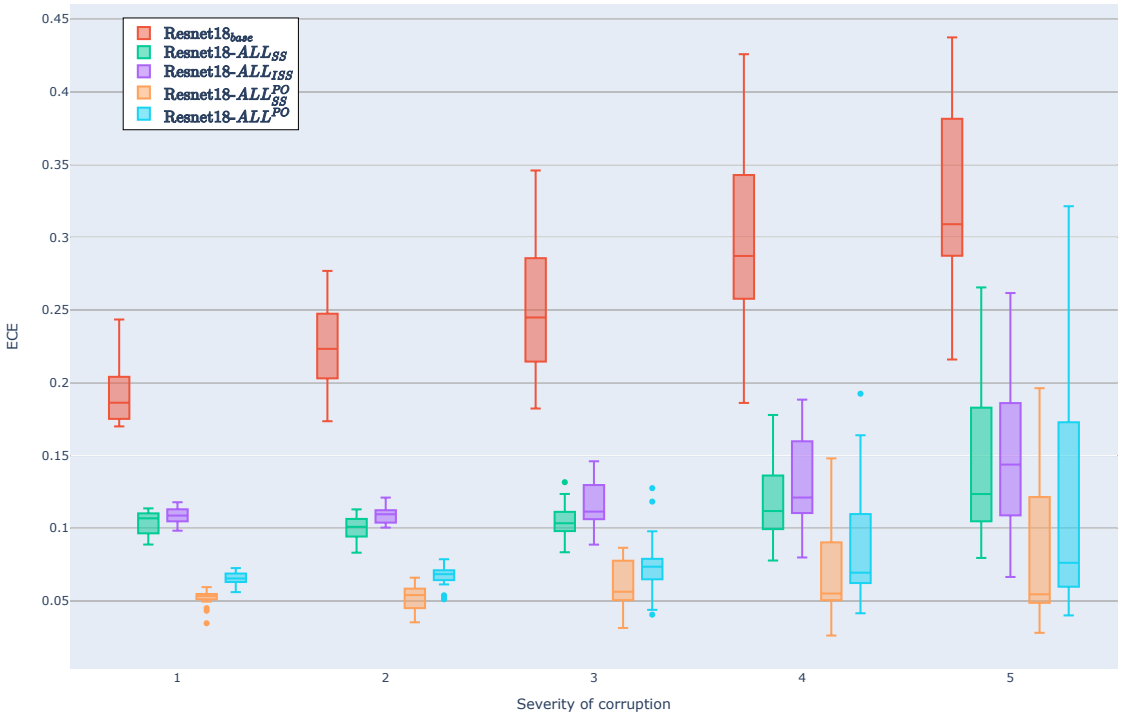

Figure 34: A boxplot of the ECE scores per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in red for reference. The other boxplots are for models Resnet18-ALL$_{SS}$ (green), Resnet18-ALL$_{ISS}$ (purple), Resnet18-ALL$^{PO}_{SS}$ (yellow) and Resnet18-ALL^{PO} (light blue).

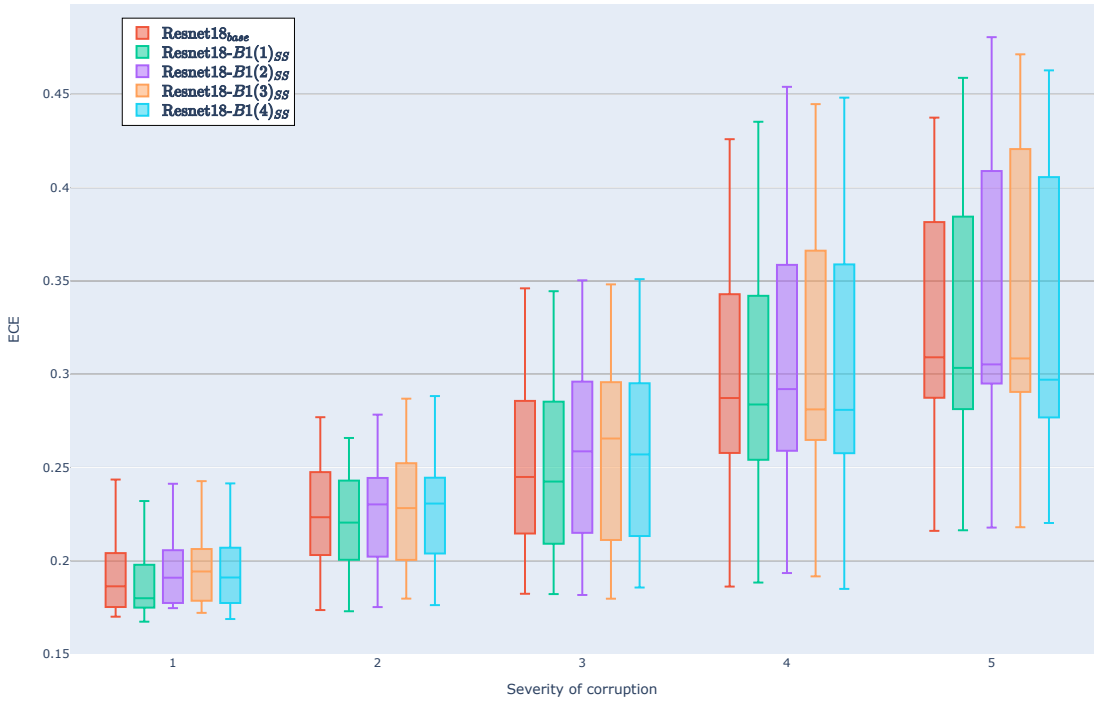

Figure 35: A boxplot of the ECE scores per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in red for reference. The other boxplots are for models Resnet18-$B1(1)_{SS}$ (green), Resnet18-$B1(2)_{SS}$ (purple), Resnet18-$B1(3)_{SS}$ (yellow) and Resnet18-$B1(4)_{SS}$ (light blue).

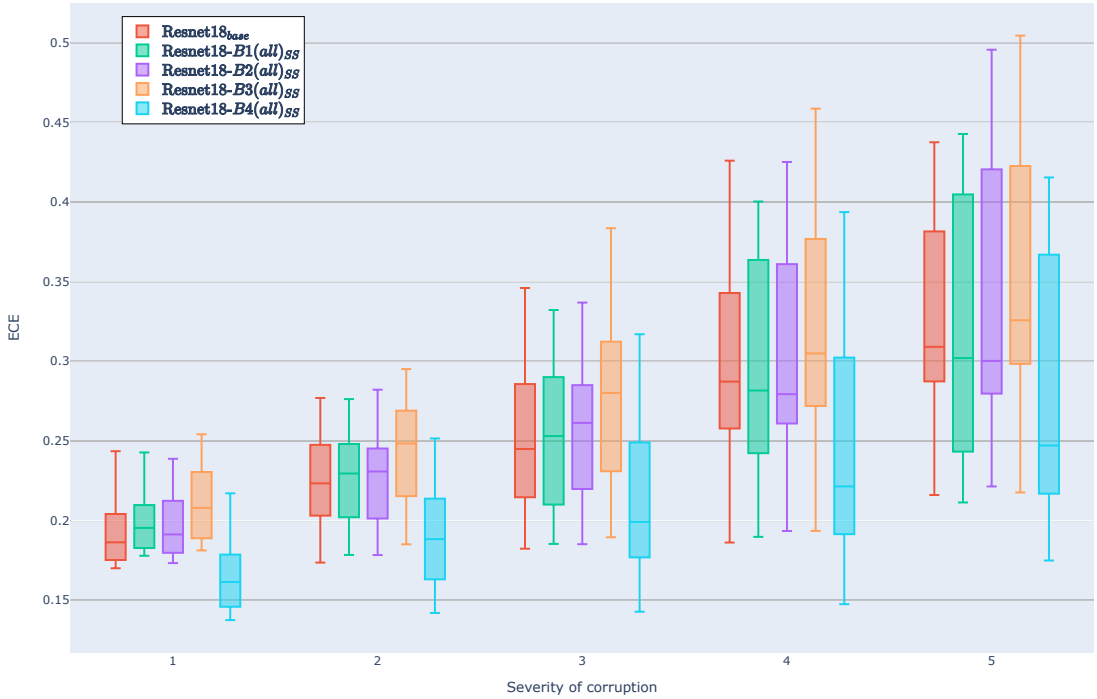

Figure 36: A boxplot of the ECE scores per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in red for reference. The other boxplots are for models Resnet18-$B1(all)_{SS}$ (green), Resnet18-$B2(all)_{SS}$ (purple), Resnet18-$B3(all)_{SS}$ (yellow) and Resnet18-$B4(all)_{SS}$ (light blue).

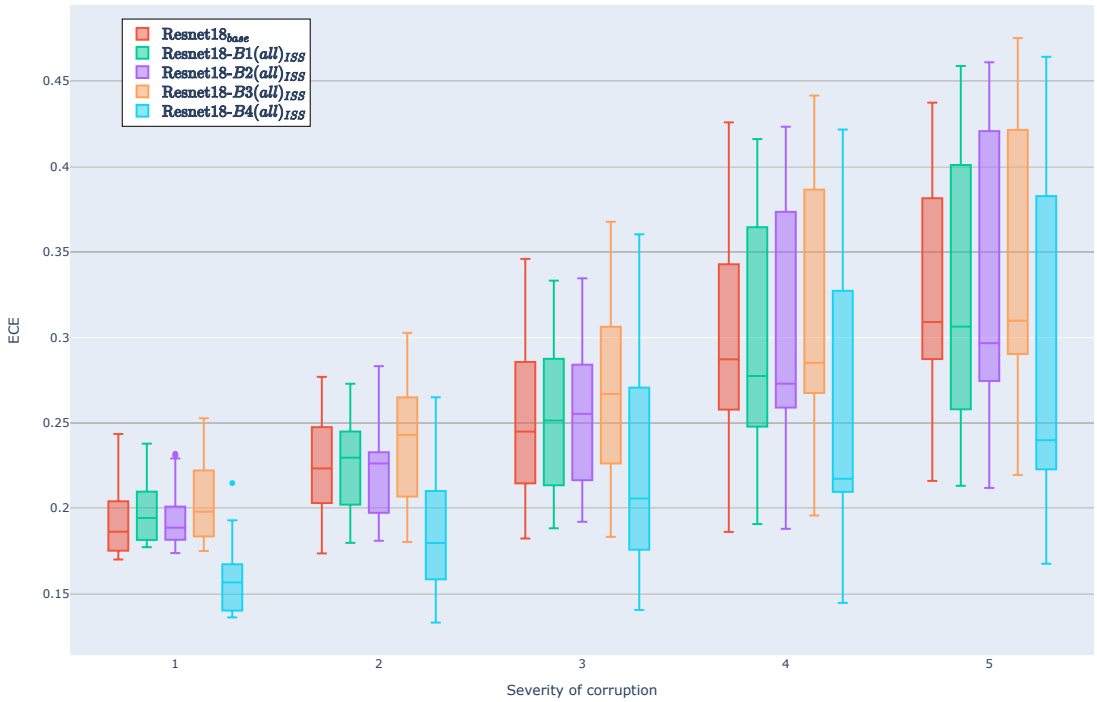

Figure 37: A boxplot of the ECE scores per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in red for reference. The other boxplots are for models Resnet18-$B1(all)_{ISS}$ (green), Resnet18-$B2(all)_{ISS}$ (purple), Resnet18-$B3(all)_{ISS}$ (yellow) and Resnet18-$B4(all)_{ISS}$ (light blue).

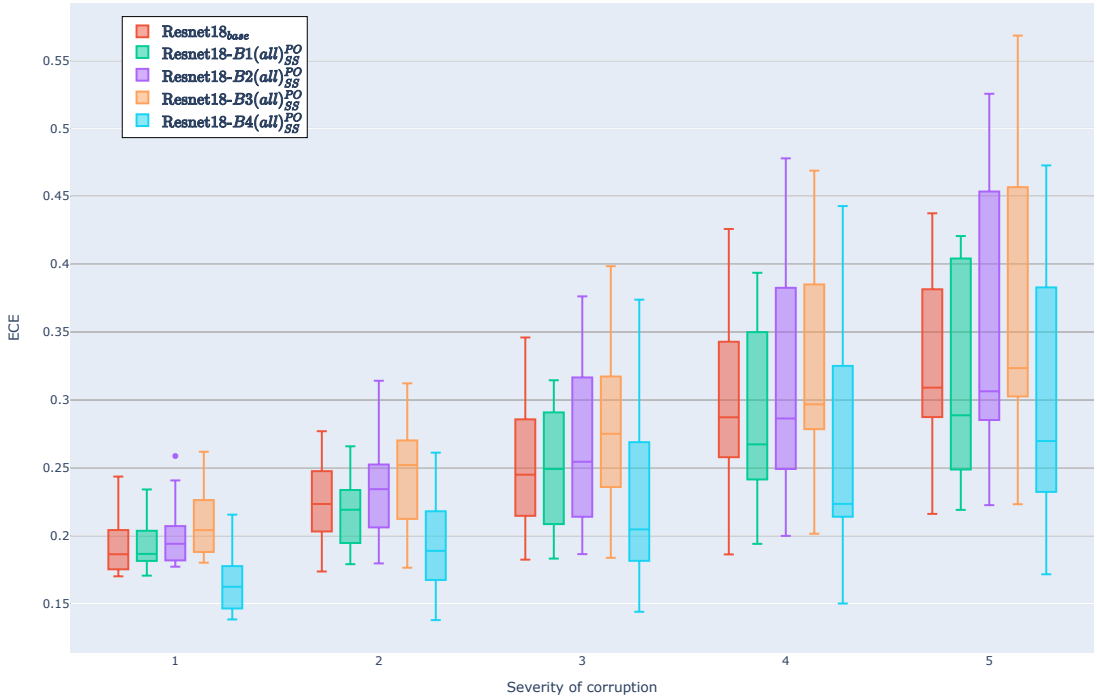

Figure 38: A boxplot of the ECE scores per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in red for reference. The other boxplots are for models Resnet18-$B1(all)^{PO}_{SS}$ (green), Resnet18-$B2(all)^{PO}_{SS}$ (purple), Resnet18-$B3(all)^{PO}_{SS}$ (yellow) and Resnet18-$B4(all)^{PO}_{SS}$ (light blue).

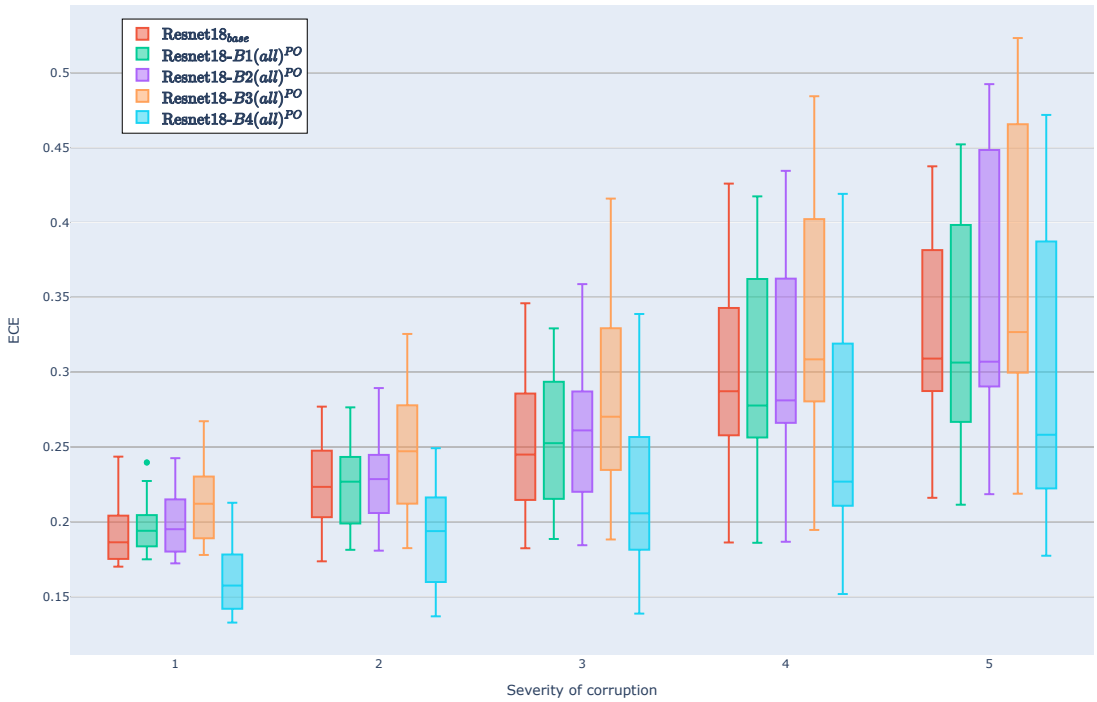

Figure 39: A boxplot of the ECE scores per corruption in the Tiny Imagenet-C dataset. The base model, Resnet18$_{base}$, is shown in red for reference. The other boxplots are for models Resnet18-$B1(all)^{PO}$ (green), Resnet18-$B2(all)^{PO}$ (purple), Resnet18-$B3(all)^{PO}$ (yellow) and Resnet18-$B4(all)^{PO}$ (light blue).

C.2 Imagenet

Figures 40, 41, 42, 43 and 44 show the ECE scores that each model obtained for each severity in the Imagenet-C dataset.

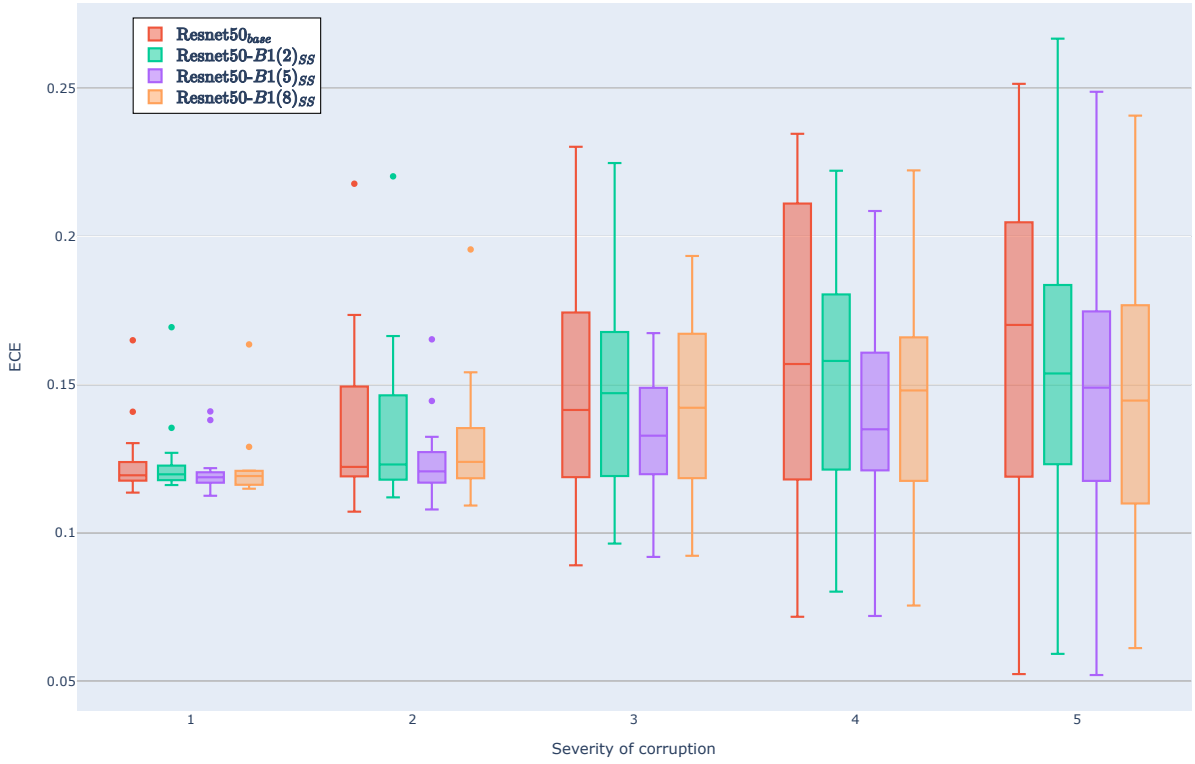

Figure 40: A boxplot of the ECE scores per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in red for reference. The other boxplots are for models Resnet50-$B1(2)_{SS}$ (green), Resnet50-$B1(5)_{SS}$ (purple) and Resnet50-$B1(8)_{SS}$ (yellow).

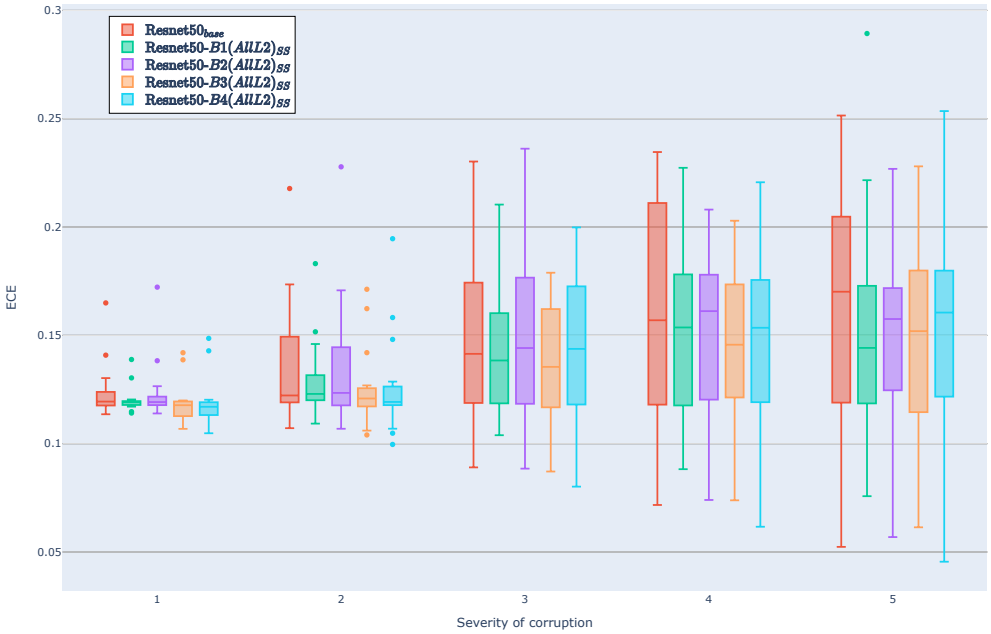

Figure 41: A boxplot of the ECE scores per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in red for reference. The other boxplots are for models Resnet50-$B1(AllL2)_{SS}$ (green), Resnet50-$B2(AllL2)_{SS}$ (purple), Resnet50-$B3(AllL2)_{SS}$ (yellow) and Resnet50-$B4(AllL2)_{SS}$ (light blue).

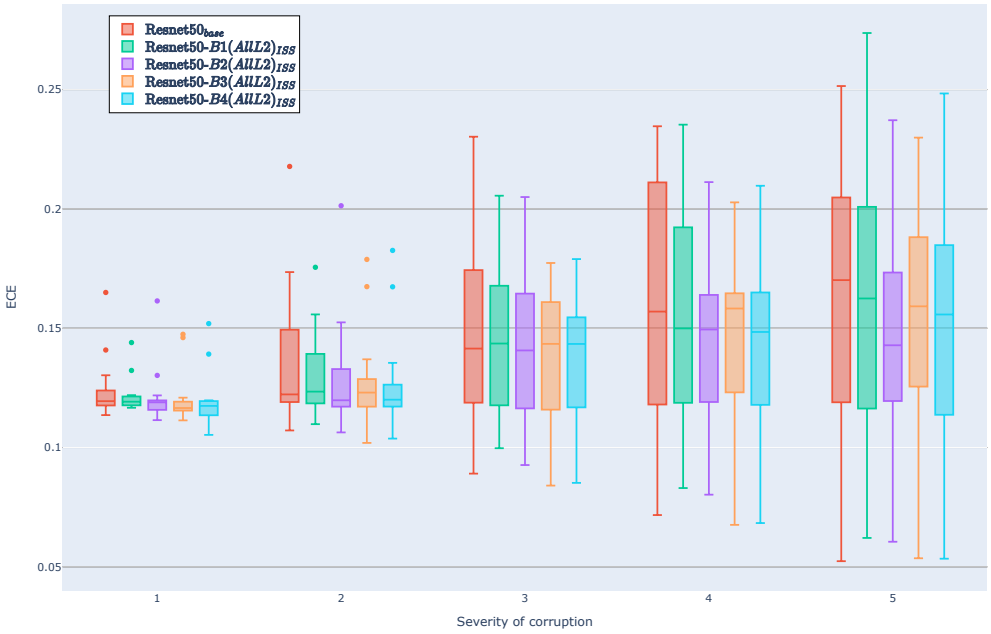

Figure 42: A boxplot of the ECE scores per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in red for reference. The other boxplots are for models Resnet50-$B1(AllL2)_{ISS}$ (green), Resnet50-$B2(AllL2)_{ISS}$ (purple), Resnet50-$B3(AllL2)_{ISS}$ (yellow) and Resnet50-$B4(AllL2)_{ISS}$ (light blue).

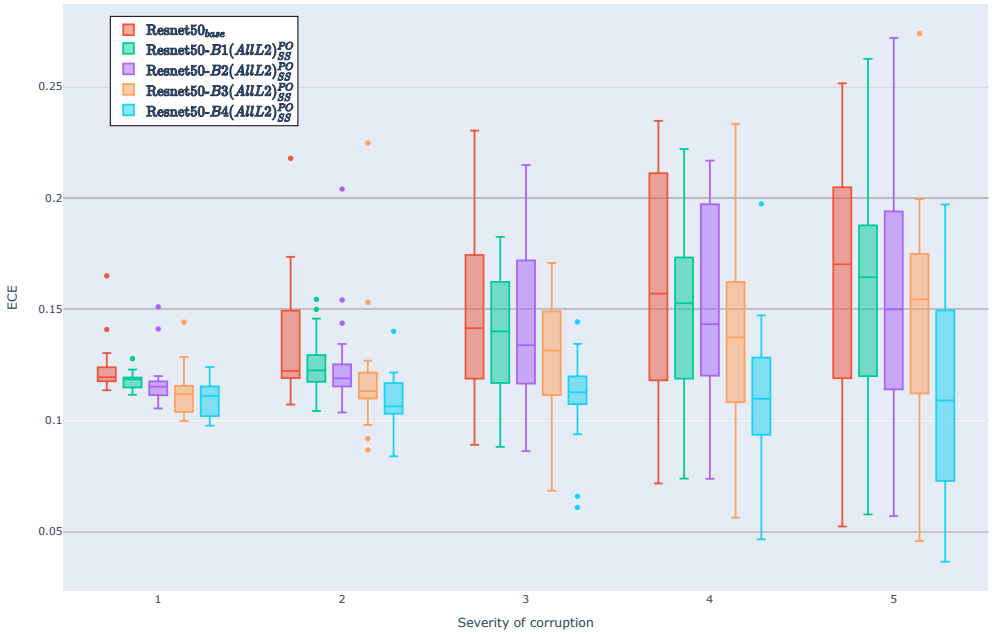

Figure 43: A boxplot of the ECE scores per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in red for reference. The other boxplots are for models Resnet50-$B1(AllL2)^{PO}_{SS}$ (green), Resnet50-$B2(AllL2)^{PO}_{SS}$ (purple), Resnet50-$B3(AllL2)^{PO}_{SS}$ (yellow) and Resnet50-$B4(AllL2)^{PO}_{SS}$ (light blue).

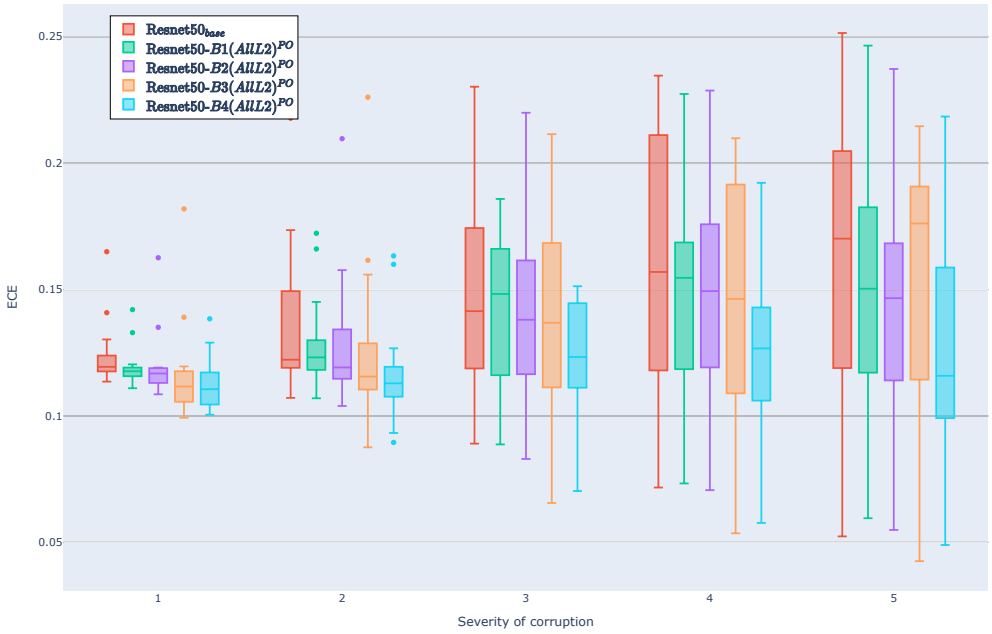

Figure 44: A boxplot of the ECE scores per corruption in the Imagenet-C dataset. The base model, Resnet50$_{base}$, is shown in red for reference. The other boxplots are for models Resnet50-$B1(AllL2)^{PO}$ (green), Resnet50-$B2(AllL2)^{PO}$ (purple), Resnet50-$B3(AllL2)^{PO}$ (yellow) and Resnet50-$B4(AllL2)^{PO}$ (light blue).