On the Regularization of Convolutional Neural Networks and Transformers under Distribution Shifts

Leo Zi-You Assini
Supervisor: Wendelin Böhmer
EEMCS, Delft University of Technology, The Netherlands
22-06-2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

The use of Transformers outside the realm of natural language processing is becoming more and more prevalent. Already in the classification of data sets such as CIFAR-100 it has shown to be able to perform just as well as the much more established Convolutional Neural Network. This paper investigates the possible out-of-distribution capabilities of the multi-head attention mechanism, through the classification of the MNIST data set with added backgrounds. Additionally, various regularization techniques are applied to increase the generalization capabilities even more. Regularization is shown to be an important tool to improve out-of-distribution accuracy, though it might imply some trade offs for in-distribution settings.

1 Introduction

Image classification is a hugely important subdomain of Computer Vision which has given rise to many applications previously considered impossible, ranging from the detection of diseases in medical images to self-driving cars.

Great success in this field has already been achieved with highly structured models such as Convolutional Neural Networks (CNNs) [LeCun et al., 1989]. However, these models are usually trained and tested on images from the same distribution. Accuracy drops rapidly when the test images have the same type of information as the training data but with systematic modifications, known as distribution shift. We say that these images are *out-of-distribution*.

The ability to recognise images on which a system has not been trained is obviously essential for practical applications. A potential solution to this generalisation problem is the introduction of Transformers [Vaswani et al., 2017]. They have revolutionised the field of natural language processing and show great promise for image classification.

Of particular interest is the multi-head attention (MHA) mechanism found in the Transformer. The use of the MHA in image classification is however still limited. In [Dosovitskiy et al., 2020] the authors utilize the encoder from the Transformer architecture, that includes the MHA, achieving excellent accuracy in large datasets such as ImageNet. The images used for testing were however still in-distribution, so the question of the degree of generalization achieved is still unanswered.

The contributions of this paper to this line of research are the following:

- Investigate the MHA module, rather than the full Transformer encoder, as an image classification mechanism.
- Investigate the effects of regularization schemes on CNNs and MHAs under distribution shifts.

To establish a baseline, the accuracy of basic MHA and CNN on out-of-distribution (OOD) images is measured. Then a variety of regularization schemes, such as weight normalization and dropout [Srivastava et al., 2014], are added to both architectures to determine their effect on accuracy.

2 Background: Convolutional Neural Network

The primary operation utilized in Convolutional Neural Networks is the mathematical operation "convolution". The arguments of this operation are an **input** (the function x below) and a **kernel** (the function w below). The output is usually referred to as a **feature map**. As defined in the book "Deep Learning" [Goodfellow et al., 2016] the equation below is the convolution operation.

$$s(t) = \int x(a)w(t-a)\,da \tag{1}$$

This operation represents a weighted average of the values of the function x, giving a more smoothed estimate of the results of x. Though the weighting function w could be replaced with any function, this definition is the most common for machine learning contexts. Additionally, the discrete case is more common, where the kernel and input are tensors, so multidimensional arrays.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2}$$

The asterisk is the symbol used to denote the convolution between two functions. The convolution will often be utilized with multiple axes at the same time, for example a 2-dimensional image will require a 2-dimensional kernel. The equation below allows for this and is called "cross-correlation", a term that in machine learning contexts is often interchanged with the term convolution.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3}$$

In the context of a 2-D image, a convolution is the act of placing the kernel in the top left region of the image, computing the weighted sum and then shifting the kernel to the right. This operation is done row by row until the feature map is completed.

Multiple feature maps can be computed in parallel to be able to detect different features with each kernel.

A traditional layer of a CNN can be split into 3 sublayers. The first is a series of convolutions performed in parallel to produce a set of linear activations. The second is a nonlinear activation such as a rectified linear unit (ReLU), often referred to as the detector stage. Thirdly, a pooling function which is defined in section 2.3. This building block can then be repeated or combined freely with other functions and layers to create deep learning models.

CNNs have three important properties which will be explained in the following sections.

2.1 Sparse interactions

When working on images, the input is usually composed by many thousands of pixels. Feeding it to a traditional neural network would require a model with just as many parameters. However, meaningful information is likely found in just a small subset of the image, so most of the weights would be redundant.

The use of just a few kernels, each one much smaller than the input image, greatly reduces the amount of parameters needed by the CNN.

One might suppose that if not all nodes between layers are fully connected then information could be lost. Fortunately, when used in *deep* networks with many layers the nodes will *indirectly* interact, allowing for both efficiency and expressiveness.

2.2 Parameter Sharing and Equivariant representations

The use of kernels is a form of parameter sharing, as a single set of weights are applied to every single input. This not only drastically reduces the total number of model parameters, as already mentioned, but also causes the layers to have *equivariance* to translation.

Mathematically a function f is equivariant to a function g if $f(g(x)) = g(f(x))$. In the case of images g could, for example, shift all images to the right. This would confuse a traditional neural network but not a CNN. A particular feature, for example a straight line, can be at any location in the image and the network will still be able to recognise it.

2.3 Multiple Layers and Pooling

A pooling layer replaces areas of the input with a statistical summary. A classic example is that of max-pooling [Zhou and Chellappa, 1988], where an area is summarized by its maximum value.

The goal is to make the model invariant to meaningless variations or noise in the input as well as to provide local translation invariance, as we are usually not interested in the *precise* location of features but in their presence and *approximate* position.

Combining multiple layers of convolutions and pooling creates a recognition hierarchy where higher levels detect increasingly more complex and global features [LeCun et al., 1989].

3 Background: Transformer

Originally introduced in [Vaswani et al., 2017], the transformer architecture is a complex model following the encoder decoder pattern. This model has had great success in the field of natural language processing (NLP). The most prevalent example of the Transformer architecture applied to image classification is in [Dosovitskiy et al., 2020]. In this paper only the encoder component is used with an additional Multilayer Perceptron (MLP) layer for final classification.

3.1 Multi-Head Attention

The Multi-Head Attention (MHA) takes as input a sequence of elements, often referred to as **tokens** and returns the tokens

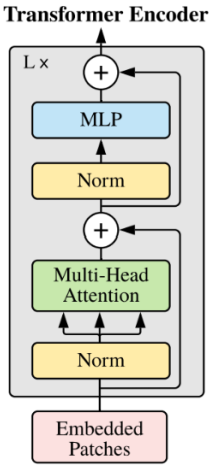

Figure 1: Encoder architecture used in [Dosovitskiy et al., 2020]

modified to take in account the similarity between each token and every other token in the sequence. In NLP, this is equivalent to contextualising the meaning of a word with respect to the surrounding sentence.

To do this the attention mechanism requires a key, query and value. Denoted as K, Q and V. In the case of self-attention all three of these will be created from the same input tensor.

The input is defined as, $I = [\boldsymbol{x}_1 \quad \dots \quad \boldsymbol{x}_n]^T \in R^{n \times d}$, where n is the length of the sequence, d is the number of features per token and $\boldsymbol{x}_i$ is a column vector representing a token of the sequence.

The K, Q and V are created by projecting I using learnable weights, $W_k, W_q, W_v \in R^{d \times d_k}$. Where d_k is the embedding dimension. To calculate the similarity between pairwise tokens the dot product is used between K and Q, to form an attention score matrix, $QK^T \in R^{n \times n}$. This matrix contains the pairwise attention values each token should have towards other tokens. These values are multiplied with V to allows the tokens to attend to various parts of the sequence. Additionally, 2 normalization techniques are used on the attention values in the form of a division by $\sqrt{d_k}$ and a row wise softmax function. The full attention formula can be found below.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \qquad (4)$$

The computation above produces an output matrix $O \in R^{n \times d_k}$. This output is also called a *head* and can be computed multiple times in parallel $\{O_1 \dots O_H\}$ before being concatenated and passed through a final linear layer to produce the final result.

There have also been attempts to describe the relationship between self-attention and Convolutional layers. *"The class of functions expressed by a layer of self-attention encloses all convolutional filters"*, [Cordonnier et al., 2019]. The authors prove that given enough heads the self-attention layer is at least as expressive as any Convolutional layer. To

be noted however that this does not mean that self-attention is actually acting like a CNN, simply that it could.

3.2 Positional Encoding

Unlike Recurrent Neural Networks inputs are passed as a *bag of tokens* to the self-attention mechanism. In the setting of NLP this can be problematic as the meaning of sentences depends greatly on the order of words.

In [Vaswani et al., 2017] a sinusoidal positional encoding is utilized. It is also possible to use a learned positional encoding as done in [Dosovitskiy et al., 2020]. This consists of a set of learnable parameters which are summed to the input.

The two aforementioned papers note that positional encodings improve performance but the type of encoding (either learned or fixed) seems to have no significant impact on final accuracy.

4 Background: Regularization

As defined in [Goodfellow et al., 2016] regularization is any modification made to a learning algorithm to reduce its generalization error but not its training error. This can be done in a number of ways, from adding parameter norm penalties to the objective function, to dropping neurons or even modifying the input.

4.1 Parameter Norm Penalties

Parameter Norm Penalties aim at reducing the model complexity by restricting or modifying the values of its parameters. This is done by adding a term to the objective function J of the model.

$$\hat{J}(\theta, X, y) = J(\theta, X, y) + \alpha\Omega(\theta) \tag{5}$$

With $\hat{J}$ as the regularized version of the objective function. The α term is a hyperparameter to regulate the impact of the regularization, values can range between $[0, \infty)$.

One of the most common penalties chosen is L_2, also known as weight decay. The regularization term is half the sum of squared values of the weights, $\Omega(\theta) = \frac{1}{2}||w||^2$. The goal is to drive the values of the weights closer to 0.

Disagreement regularization terms [Li et al., 2018] have also been defined to improve the accuracy of the MHA. The goal of the MHA is to have each head attend to different parts of the input. There is however no part of the model which ensures this diversity in the heads.

The authors of [Li et al., 2018] attempt to solve this by increasing the cosine distance between the output heads. In the contexts of matrices this can be defined as the dot product between two matrices, normalized by the product of their norms. This similarity is computed pairwise over all the heads and then subtracted from the objective function so that the average distance between heads is increased. H is the number of output heads.

$$D_{output} = -\frac{1}{H^2}\sum_{i=1}^{H}\sum_{j=1}^{H}\frac{O_i O_j^T}{||O_i||\,||O_j||} \tag{6}$$

4.2 Dropout

Dropout [Srivastava et al., 2014] is the process of removing, for every training round, some hidden units of the neural network with probability p. In practice this means setting random values of the output tensor of a layer to 0.

The motivation behind this is that it allows the model to test a great range of different architectures without additional computational costs. In addition it has shown to reduce co-adaptation of the neurons [Hinton et al., 2012], as the neurons cannot rely on the other neurons being there.

Similar techniques can be used on the input, for example with MHAs it could be possible to drop certain tokens randomly.

4.3 Layer Normalization

In layer normalization [Ba et al., 2016] the outputs of the previous layer are normalized using their $E[x]$ and $Var[x]$ as shown in the equation below.

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} * \gamma + \beta \tag{7}$$

The γ and β are learned parameters which are there to replace the lost expressiveness of the model during regularization. It may seem counter intuitive to allow the model to potentially return to the same values as before regularization. However, as explained in [Goodfellow et al., 2016], the mean of this new parametrization is determined completely by β and not by complicated interactions occurring in previous layers. This makes the parametrization much easier to learn with gradient descent.

5 Methodology

The research question addressed by this paper can be answered more easily by dividing it into smaller hypotheses.

Hypothesis 1: *"An unregularized MHA performs better than an unregularized CNN under distribution shift."*

The self-attention mechanism has proven successful in other domains and has interesting properties including long range dependencies and low inductive priors which should help it perform well in OOD tasks.

Measuring the baseline performance of the two architectures is also required to assess the effect of the regularization schemes.

Hypothesis 2: *"Regularization schemes, with tuned hyperparameters, improve the accuracy of the CNN and MHA architectures under distribution shift."*

Regularization schemes are well established techniques which are known to improve model performance for in-distribution tasks. These improvements should be transferable to the OOD scenario.

Hypothesis 3: *"A regularized MHA has a better accuracy than a regularized CNN under distribution shift."*

The greater variety of regularization techniques available for MHA should give it an edge over the CNN.

5.1 Data Set

To test the accuracy of the models in in-distribution settings against out-of-distribution two data sets are necessary, MNIST and CIFAR-10. Both contain thousands of images and are split into a training set and a test set. MNIST contains black and white hand drawn digits, CIFAR-10 coloured images of ten different classes (airplane, automobile, bird, etc...). These data sets are combined by superimposing (technically, subtracting) a number from MNIST to an image from CIFAR-10 so that visually the number appears as the foreground and the image as the background.

The models were trained on a varying number of background images, respectively $[1, 2, 4, 8, 16, 32, 64]$ images, for each digit class. This was done to observe how the models handle variations in the correlation between the foreground and background. In the case in which there are few different background images it is likely that the model will simply focus on the background as it has more characteristics compared to the visually simpler digits. If we move onto the other extreme of 64 background images then there will be almost no correlation between the foreground and background making it more likely for the model to concentrate on the foreground. This 64 background case will be considered similar to an in-distribution case during the experimental phase.

During training, both foreground and background come from the training sets of the MNIST and CIFAR-10 data sets. The validation set foregrounds come from the MNIST test set and the backgrounds from the CIFAR-10 training set, this is used to calculate in-distribution accuracy. The test set foregrounds and backgrounds come from the respective test sets, this is used to calculate the out-of-distribution accuracy.

Figure 2: Representation of how the data sets were combined.

5.2 Architecture

To understand the out-of-distribution capabilities of the MHA module, not to be confused with the complete transformer, attempts have been made to replicate the architecture of a traditional CNN while replacing only the convolutional layer with the MHA.

The first task is to simulate the moving kernel of the CNN moving over the input image with a stride of one. This was done by utilizing the PyTorch unfold method [Paszke et al., 2019].

The architecture is then composed of a layer of MHA, a max pool, then again the same 2 layers followed by an MLP head for classification.

The positional encoding is not part of the baseline architecture but has been added to clarify where it is positioned when the corresponding component is tested.

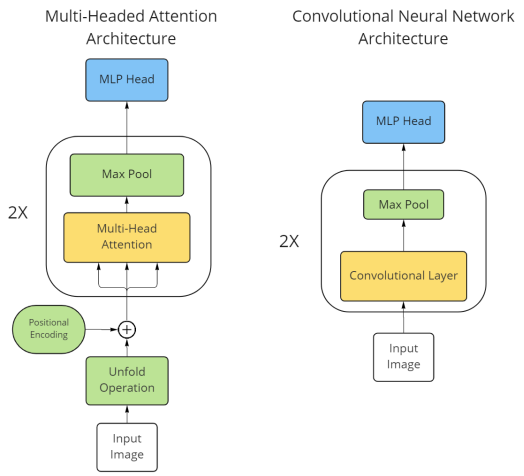

Figure 3: Overview of model architectures. The illustration is inspired by [Vaswani et al., 2017]

6 Experimental Setup

To run the experiments, the two architectures have been implemented using the PyTorch framework.

Models were trained over 10 epochs of the data, using the Adam optimizer, batch size of 128, fixed learning rate of 0.001 and cross entropy as the loss function. To reliably calculate the mean and standard deviation of the accuracy of each experimental setup they where each run 10 times. If the standard deviations of different experiments do not overlap we can be confident that there is a statistically significant difference between their accuracy. Training, in-distribution and out-of-distribution (OOD) accuracies were calculated but in the plots below often only the OOD case is show as that is the value of interest. The x axis indicates the number of backgrounds used.

Initial experiments were run with the baseline CNN and MHA architectures found in figure 3. Once these baselines were established the architectures where run with regularization techniques applied one at a time to observe their effects.

In the CNN the following techniques were tried:

- Normalization layers before and after each convolutional layer.
- Dropout layers after each convolutional layer.
- L_2 norm applied to the loss function.

In the MHA the following techniques were tried:

- Normalization layers after each MHA layer. The layer was also tried before each layer, inspired by the work of [Xiong et al., 2020].
- Dropout layers after each MHA layer.
- L_2 norm applied to the loss function.
- Learned positional encoding.
- Disagreement regularization, see equation 6.

Hyperparameter tuning was conducted on dropout (probability of dropout p) and L_2 norms (α) to verify hypotheses 2 and 3. The values tried can be found in the graphs of section 7.

During hyperparameter training it is important to have a validation set to find the best hyperparameter not just for the test set used but test sets in general. If the test set were used for hyperparameter tuning then overfitting would likely occur on the test set. The in-distribution data set is used as a validation set.

However during the experiments it became clear that the ideal values for the in-distribution set do not achieve even similar performance in the out-of-distribution settings.

A possible explanation is that the background images have already been seen and therefore can not be considered OOD. Meaning that it makes sense for the best hyperparameters for in-distribution to not be the best for the OOD data set.

Therefore in the results section a greater focus is placed on the hyperparameters and regularization schemes which work best for the out-of-distribution data set.

The regularization techniques which showed the best performance in isolation were also tried together on their respective architecture.

7 Results

In this section, we test our hypotheses against the experimental results. Particular attention is paid to the cases of 4, 8 and 16 backgrounds because they are considered the most difficult for the models to understand correctly. Furthermore the 64 background case can be considered a representation of in-distribution performance, as explained in section 5.1.

7.1 Unregularized Architectures

Figure 4 below compares the two unregularized, baseline, architectures. Accuracies are similar for all numbers of background images, except for the extreme case of 64 backgrounds where the CNN outperforms the MHA by a significant margin. MHA outperforms the CNN on the 8 backgrounds case but there is a strong overlap of the standard deviations so no statistically significant difference can be observed. This disproves Hypothesis 1.

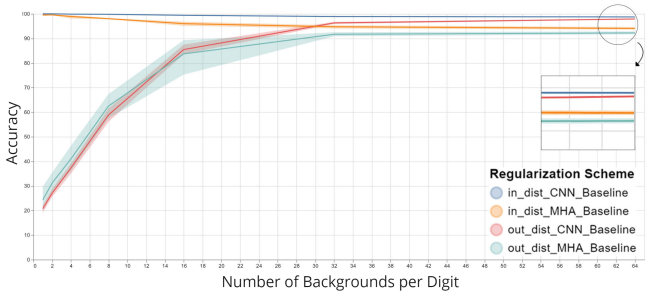

Figure 4: Performance of baseline CNN and MHA

7.2 Regularized CNN

Normalization Figure 8 in the Appendix shows that layer normalization before convolution has little effect on improving OOD accuracy, only for 16 background images is the accuracy improved, however not by a significant margin. For all other numbers of backgrounds, performance is decreased.

Layer normalization after the convolution is even worse, reducing accuracy with respect to the baseline architecture for all values of backgrounds images.

Dropout

As shown in Figure 9, the optimal dropout hyperparameter determined using the validation set is $p = 0.3$.

However when observing the test set in figure 10, $p = 0.7$ greatly outperforms the other values and 0.3 has the worse performance.

L2 Norm

As shown in figure 12, L_2 has no impact on performance for most values of its α hyperparameter. In most cases accuracy improves only modestly and with high amounts of overlap with respect to the mean of the baseline architecture.

The only α value with a statistically significant difference is 0.01. This value achieves a strong improvement with both 8 and 16 backgrounds, with no overlap present. The trade-off with for this improvement is a drop in accuracy for 64 backgrounds.

Combined Regularization

By combining the best techniques for CNNs which were dropout ($p = 0.7$) and L_2 ($\alpha = 0.01$) accuracy was significantly improved for 16 backgrounds but not for other number of backgrounds.

Overall, the techniques used in isolation had higher accuracies than the combined version, see figure 22.

Summary of CNN Regularization

Figure 5 shows the best CNN regularization schemes. Both dropout and L_2, with appropriate hyperparameters, provide strong improvements compared to the baseline architecture, with dropout performing slightly better than normalization with 8 background but no clear winner present. An unexpectedly high value of dropout ($p = 0.7$) is found to be required for the model to not overfit on the training set, generalize better and therefore provide higher accuracy in OOD settings.

The pattern of regularization helping in OOD but not in in-distribution can be noted for both dropout and normalization. This reinforces the idea stated in [Torralba and Efros, 2011] that in-distribution accuracy is no guarantee of out-of-distribution accuracy.

These results are strong evidence for Hypothesis 2, with regards to the CNN. Regularization does seem to improve the accuracy of the model under distribution shift, with the trade off in high background cases.

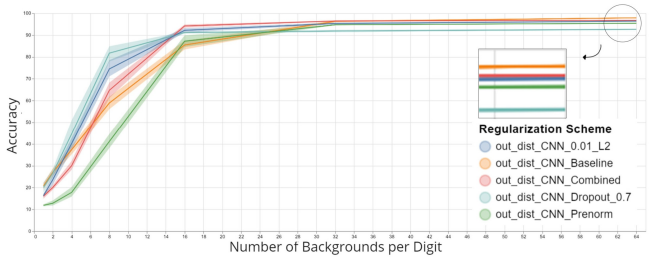

Figure 5: Overview of Regularization schemes on CNNs

7.3 Regularized MHA

Normalization

Figure 13 shows that neither placing the normalization before or after the MHA layer provides significant improvements for a number of background images under 32.

However, unlike the regularization for the CNN, both normalization techniques improve the accuracy of the model in high background cases.

Dropout

Compared to the CNN architecture, dropout for MHAs has less promising results in OOD settings. As shown in figure 15 no matter the value of the dropout, accuracy is similar up to 8 backgrounds and then falls greatly for higher numbers of background images.

L2 Norm

Similarly to dropout, L_2 has limited effect on the accuracy of the model. For all values tried there was overlap of the standard distributions for all number of backgrounds, with the best performing value being $\alpha = 0.1$, which improved both 8 and 16 images though not by large margins.

It should be noted that unlike dropout, performance for high backgrounds is not reduced. Accuracies vary only by a few points.

Positional Encoding

Positional encoding, figure 19, improves accuracy for 2, 4, 8 and 16 backgrounds but with overlaps in the standard deviations. Almost no difference is made for 64 backgrounds.

The unfold operation creates overlapping patches, this could explain the poor performance of the absolute positional encoding tested.

Further research might look at different kinds of positional encodings. For example [Shaw et al., 2018] utilizes a relative positional encoding.

Disagreement Regularization

Disagreement Regualization, equation 6, had no significant impact on the accuracy of the model.

Tests conducted by another team member of the Research Project concluded that the number of heads utilized had no impact on performance. Seeing as the penalty norm used has the goal of making each head as different as possible this could explain why it does not impact performance almost at all.

Combined Regularization

An MHA with positional encoding, post layer normalization and L_2 ($\alpha = 0.1$) did result in an architecture with significant accuracy improvement for 16 backgrounds and barely any overlap in standard deviation for 8 backgrounds. Accuracy does drop however for low backgrounds and 64 backgrounds.

This architecture resembles the traditional transformer much more than the architectures tested in the previous experiments. This result suggests that the MHA module in isolation is less effective than a Convolutional layer, however with the addition of other modules can become very effective.

Summary of MHA Regularization

The full comparison of the best regularization techniques can be found in figure 6, disagreement regularization and positional encoding have been removed to reduce clutter.

No single technique in isolation was able to improve accuracy in a statistically significant way. Only when combined, did regularization improve accuracy meaningfully.

Therefore Hypothesis 2 for MHAs is likely to be true for combined regularization techniques, however more research needs to be done to determine which combinations provide the best results.

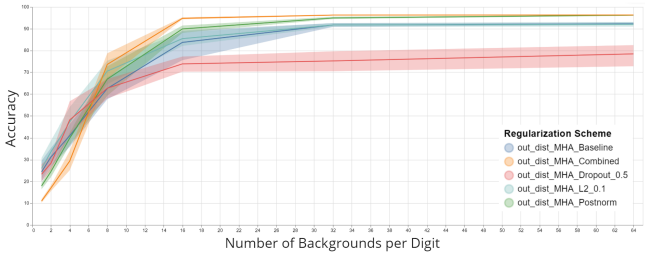

Figure 6: Overview of Regularization schemes on MHAs

7.4 Comparison between Regularized Architectures

The best regularization scheme for CNNs was the L_2 norm and for MHA it was the combined architecture described in section 7.3. Both schemes improve the architectures significantly above their baseline counterparts, figure 4.

In the case of 64 backgrounds the baseline CNN still has the the highest performance, followed by the combined MHA.

Neither of the two regularized architectures shows consistently higher results than the other. The MHA performs better for 16 backgrounds but worse for 8.

Therefore Hypothesis 3 cannot be confirmed or rejected. More work on different types of regularizations and their combinations would be required.

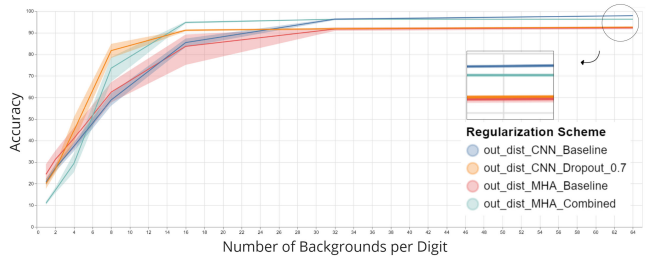

Figure 7: Overview of Best Regularization scheme for both architectures.

8 Related Works

The majority of work done in supervised learning is in settings where training and testing data are considered to be independent and identically distributed (i.i.d), meaning that they are all taken from the same distribution. This would not be an issue if the distributions of commonly used training data sets faithfully reflected the distribution of real world data.

[Torralba and Efros, 2011] shows that data sets have, on the contrary, strong in-build biases. A support vector machine (SVM) trained to identify which of the twelve most popular data sets in computer vision an image belongs to, achieves a significantly more than random accuracy. The same paper also shows the lack of *cross-dataset generalization*, the average accuracy of models drops considerably when tested on a separate data set containing images of the same class. They also note that researchers concentrate too much on improving benchmark scores on specific data sets and forget the final objective of applying these models to the real world.

Some studies have tried to tackle this issue. [Dollar et al., 2010] uses a data set orders of magnitude greater than usual of images from surveillance systems to avoid selection bias.

[Hendrycks et al., 2020] creates 4 new data sets with systematic distribution shifts on which researchers can test the OOD robustness of their models. ImageNet-9, [Xiao et al., 2020] is a data set that uses images with different amounts of background and foreground signal. This also alleviates the tendency of some models to focus on the background rather than the foreground of images, evidence of this behaviour is demonstrated in [Rosenfeld et al., 2018].

The domain of computer vision has long been dominated by the CNN [LeCun et al., 1989], not surprising due the their impressive results [Krizhevsky et al., 2012]. However, CNNs have several limitations which have come to light in OOD settings. [Geirhos et al., 2018] showed that CNNs trained on ImageNet had a strong bias towards recognizing textures instead of shapes. This contradicts the long standing hypothesis that CNNs combine low-level features into increasingly complex shapes. They managed to mitigate this issue by using a stylized version of ImageNet.

[Hendrycks et al., 2020] showed the CNN's difficulties in dealing with orientation, zoom and scale in test data. Benchmarks on corruption robustness and surface variation robustness have also been conducted by [Hendrycks and Dietterich, 2019].

A possible solution to this issues could be the Transformer architecture. Transformers first appear in 2017 with [Vaswani et al., 2017], and they have since reached impressive performance in tasks such as machine translation [Ott et al., 2018]. Much of their success has been due to their scalability and parallelisation capabilities leading to high performing models such as BERT [Devlin et al., 2018] and GPT [Radford et al., 2018] which have millions of parameters.

Their potential in computer vision became quickly clear with [Dosovitskiy et al., 2020] and [Touvron et al., 2020]. The fact that Transformers have weaker biases towards textures and stronger biases for shapes and structures has already been shown in [Zhang et al., 2021]. They have also shown great performance when dealing with occlusion and information loss [Naseer et al.,]. This might be due to their ability to learn global relationships within the input [Ramachandran et al., 2019].

This body of research has inspired the idea examined in this paper of replacing the CNN's convolutional layer with self-attention. Other examples of augmented CNNs can be found in various papers including [Bello et al., 2019] where they combine convolution and self-attention.

Regularization techniques have long been used in CNNs to improve performance and generalization capabilities. The most common examples are those of L_2 norm, normalization and dropout [Srivastava et al., 2014]. More complex and recent attempts to reduce overfitting and improve generalization are in [Zheng et al., 2018] which uses a two phase training method.

Regularization techniques are also commonly used with Transformers with new methods continuously being introduced. Examples of specific dropout strategies applied to transformers are found in [Zhou et al., 2020] where instead of dropping connections entire attention heads are dropped with a certain probability. This is to avoid any heads from dominating the predictions and to encourage a wider range of information to be encoded in the other heads.

Three different disagreement regularizations are introduced in [Li et al., 2018], which act on the subspaces, attended positions and output heads. Other examples of penalty norms can be found in [Lin et al., 2017], which has the goal of reducing redundancy between the attention weights of the various heads.

9 Responsible Research

9.1 Ethics

The data used comes from the MNIST and CIFAR-10 data sets. These are well known and publicly available data sets which have been used in countless experiments and research in the field of computer vision. MNIST contains hand written digits. CIFAR-10 contains images of 10 classes of animals and vehicles. Therefore there are no ethical concerns concerning private or personal information being used.

This research aims to further the understanding of machine learning models in the field of computer vision. This is a field whose final applications are often in human centered environments. The classical example of self driving cars is one where the accuracy of these models is paramount and errors can lead to damage to both humans and property. It is therefore essential for all results to be accurate and well documented.

To ensure that no data trimming has occurred the results of all experiments conducted can be found in the appendix.

9.2 Reproducibility

All the code utilized to run the experiments is contained in a single repository which will be publicly released so that it can be independently examined and executed [1].

The execution time of the models is significant. The experiments were run on the TU-Delft clusters [Delft High Performance Computing Centre (DHPC), 2022] which greatly reduced processing times.

[1] https://github.com/LeoAssini/research-project-2022

Given the random initialization of the weights in the models slight variations will be of course observed by other researchers when trying to replicate results. However this should not affect the results exposed in this paper as random variations have been mitigated a much as possible by running all experiments 10 times. All results reported are the mean and standard deviation of those multiple runs.

10 Conclusion and Future Work

Regularization has proven to be an effective tool for improving the generalization capabilities of neural networks, significantly increasing out-of-distribution accuracy (with the important trade off that in-distribution accuracy must be partially sacrificed).

There was no clear winner between CNN and MHA with both regularized architectures showing similar final performance.

In the case of CNNs, accuracy in out-of-distribution settings increases greatly using the regularization techniques of dropout with probability of 0.7 and L_2 norm with $\alpha = 0.01$. Combining multiple regularization techniques does not result in significant additional improvement.

Regularization is less effective for the MHA architecture. Applying regularization in isolation does not show statistically significant improvements over an unregularized MHA. However, when combining the best regularization techniques of positional encoding, post layer normalization and L_2 with $\alpha = 0.1$, there is a significant additional improvement in accuracy.

For future work, there are many avenues that might be explored.

Using much larger data sets, for example ImageNet, as due to the low inductive priors of MHAs they might outperform CNNs in this scenario.

Using other regularization techniques such as drophead, specific to MHAs, or the many different positional encodings that have been proposed for Transformers.

Using different combinations of regularization techniques as, due to time constraints, only the most obvious one was tested for each architecture. The promising results for MHAs suggests that this could result in further improvements.

References

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization.

[Bello et al., 2019] Bello, I., Zoph, B., Vaswani, A., Shlens, J., and Le, Q. V. (2019). Attention augmented convolutional networks.

[Cordonnier et al., 2019] Cordonnier, J.-B., Loukas, A., and Jaggi, M. (2019). On the relationship between self-attention and convolutional layers.

[Delft High Performance Computing Centre (DHPC), 2022] Delft High Performance Computing Centre (DHPC) (2022). DelftBlue Supercomputer (Phase 1). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

[Dollar et al., 2010] Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2010). Pedestrian detection: A benchmark. pages 304–311. Institute of Electrical and Electronics Engineers (IEEE).

[Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale.

[Geirhos et al., 2018] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. (2018). Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Hendrycks et al., 2020] Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., and Gilmer, J. (2020). The many faces of robustness: A critical analysis of out-of-distribution generalization.

[Hendrycks and Dietterich, 2019] Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations.

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

[Li et al., 2018] Li, J., Tu, Z., Yang, B., Lyu, M. R., and Zhang, T. (2018). Multi-head attention with disagreement regularization.

[Lin et al., 2017] Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding.

[Naseer et al.,] Naseer, M., Ranasinghe, K., Khan, S., Hayat, M., Khan, F. S., and Yang, M.-H. Intriguing properties of vision transformers.

[Ott et al., 2018] Ott, M., Edunov, S., Grangier, D., and Auli, M. (2018). Scaling neural machine translation.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A.,

Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

[Ramachandran et al., 2019] Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. (2019). Stand-alone self-attention in vision models.

[Rosenfeld et al., 2018] Rosenfeld, A., Zemel, R., and Tsotsos, J. K. (2018). The elephant in the room.

[Shaw et al., 2018] Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. volume 2.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[Torralba and Efros, 2011] Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias.

[Touvron et al., 2020] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2020). Training data-efficient image transformers amp; distillation through attention.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

[Xiao et al., 2020] Xiao, K., Engstrom, L., Ilyas, A., and Madry, A. (2020). Noise or signal: The role of image backgrounds in object recognition.

[Xiong et al., 2020] Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. (2020). On layer normalization in the transformer architecture.

[Zhang et al., 2021] Zhang, C., Zhang, M., Zhang, S., Jin, D., Zhou, Q., Cai, Z., Zhao, H., Liu, X., and Liu, Z. (2021). Delving deep into the generalization of vision transformers under distribution shifts.

[Zheng et al., 2018] Zheng, Q., Yang, M., Yang, J., Zhang, Q., and Zhang, X. (2018). Improvement of generalization ability of deep cnn via implicit regularization in two-stage training process. *IEEE Access*, 6:15844–15869.

[Zhou et al., 2020] Zhou, W., Ge, T., Xu, K., Wei, F., and Zhou, M. (2020). Scheduled drophead: A regularization method for transformer models.

[Zhou and Chellappa, 1988] Zhou, Y. and Chellappa, R. (1988). Ieee 1988 international conference on neural networks.

A Graphs for CNNs and MHAs

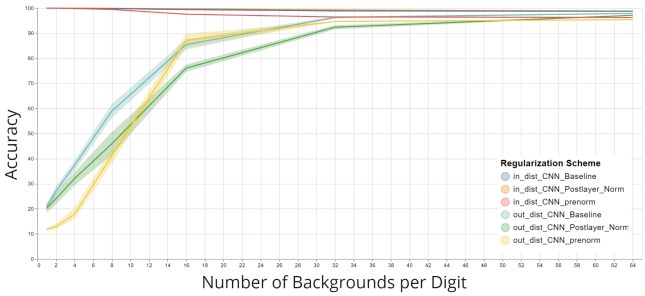

Figure 8: Comparison of pre and post layer normalization for CNNs.

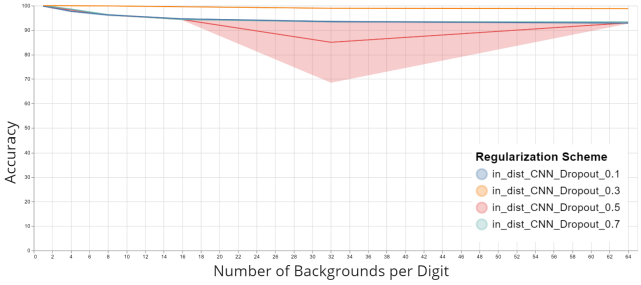

Figure 9: In-distribution of CNN with Dropout.

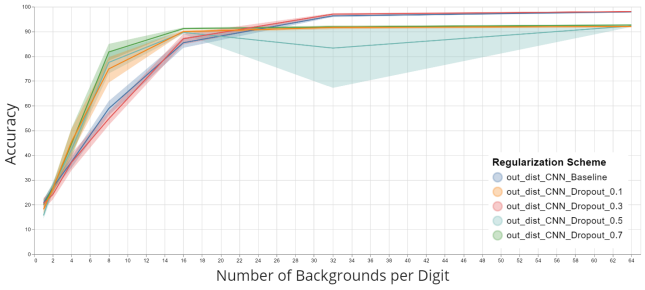

Figure 10: Out-distribution of CNN with Dropout.

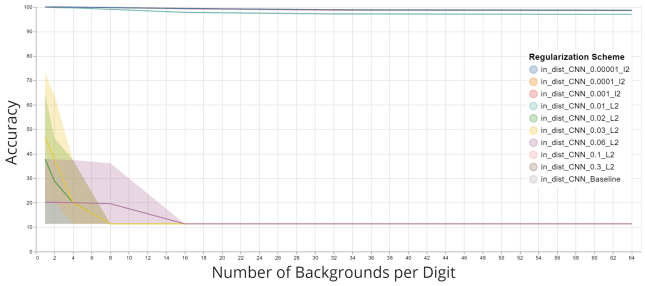

Figure 11: In-distribution of CNN with L2 Norm.

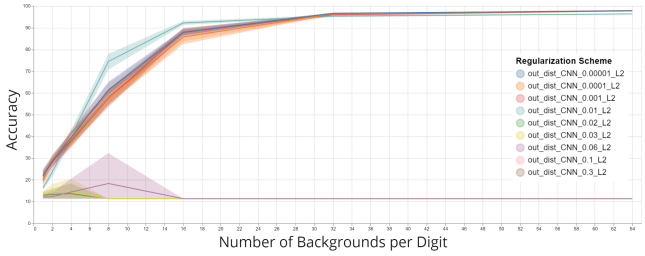

Figure 12: Out-distribution of CNN with L2 Norm.

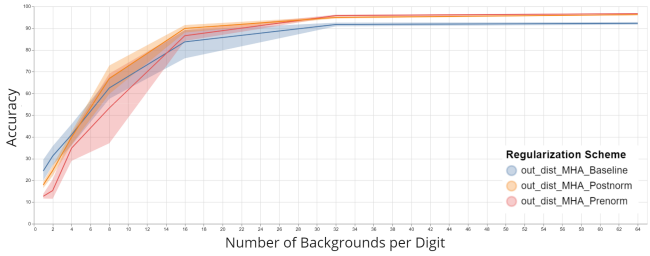

Figure 13: Out-distribution of MHA with Normalization.

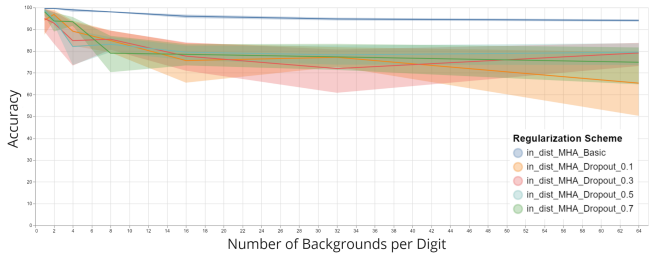

Figure 14: In-distribution of MHA with Dropout.

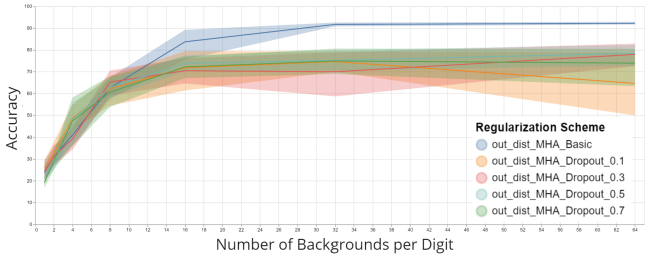

Figure 15: Out-distribution of MHA with Dropout.

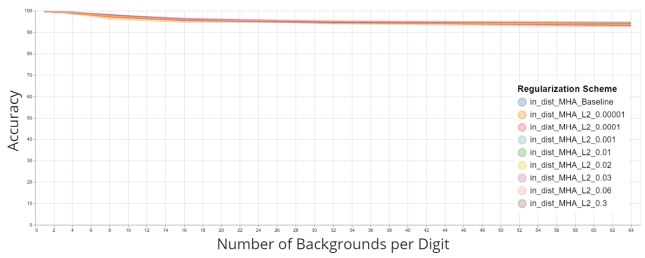

Figure 16: In-distribution of MHA with L2 Norm.

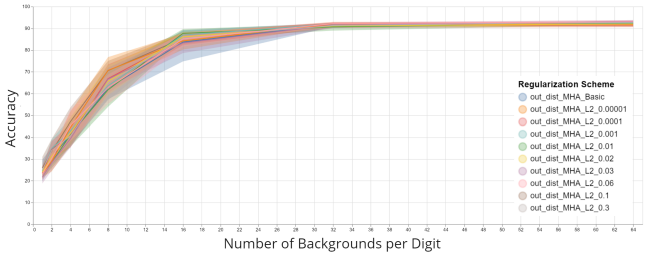

Figure 17: Out-distribution of MHA with L2 Norm.

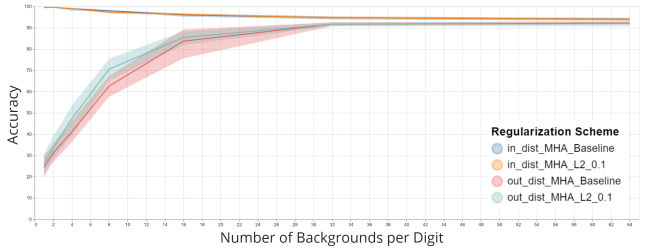

Figure 18: Out-distribution of MHA with L2 Norm, $\alpha = 0.1$.

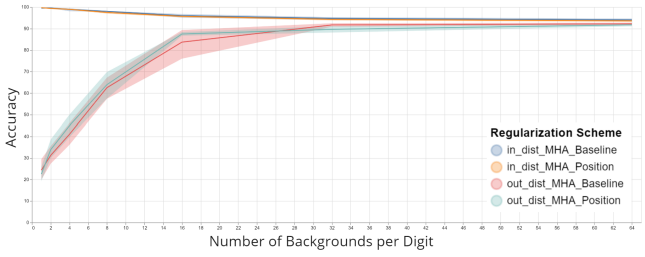

Figure 19: MHA with Learnable positional encoding.

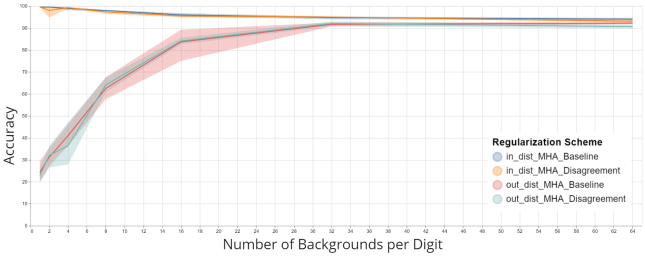

Figure 20: MHA with Disagreement Regularization.

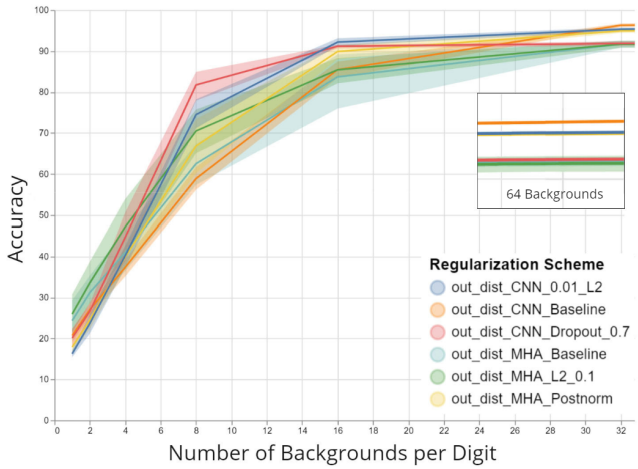

Figure 21: Overview of Best Regularization schemes in isolation for both architectures.

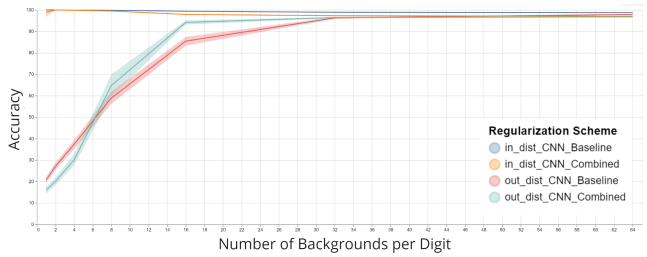

Figure 22: CNN architecture with dropout ($p = 0.7$) and L_2 ($\alpha = 0.01$).