Are CNNs that Learn to Predict Image Statistics Invariant to Domain Shifts?

Julian Biesheuvel[1] , **Tom Viering**[1] **, Ziqi Wang**[1] , **David Tax**[1]**, Marco Loog**[1]
[1]Delft University of Technology

Abstract

Yes, convolutional neural networks are domain-invariant, albeit to some limited extent. We explored the performance impact of domain shift for convolutional neural networks. We did this by designing new synthetic tasks, for which the network's task was to map images to their mean, median, standard deviation, and variance pixel intensities. We find that the performance drop due to domain shift is related to the shift in pixel values between source and target domain. Colour space transformations seemed to notably impact the network's performance, opposed to geometric transformations. For the last domain shift we find that the network manages to beat a baseline, from which we can conclude the domain shift is not too severe. Additionally, the findings reveal a less dominant role for feature transferability, for our synthetic regression tasks.

1 Introduction

Convolutional Neural Networks (CNN) are the backbone of computer vision tasks and have proved their practicality amongst others in image recognition, classification, and segmentation. 2012 marks the year in which CNNs gained traction, as a result of performance advancement in both image classification and speech recognition [1, 2, 3]. Although deep learning practices are well established in the machine learning domain, interpretability and transparency continue to remain a challenging subject [4]. For illustration, deep neural networks (DNN) are susceptible to adversarial examples [5]. As a result, the network is likely to be flawed [5]. An example like this stresses the need of attaining a better understanding of the potential capabilities and limitations of deep learning applications.

Deep learning architectures like DNNs are not omniscient and do not have unlimited capabilities [6]. Such as the network's capacity to generalise well on unseen data (target domain) with a different distribution compared to that of the training data (source domain) [7]. This is called domain shift. As CNNs are a class of DNNs, the same principles apply. Domain adaptation is the ongoing study to mitigate these effects [8, 9, 10]. It keeps, however, an ongoing search to have a complete and effective domain adaptation strategy. After all, there are always new poses, lighting, and angles that are unfamiliar to a CNN when encountered in the target domain.

Image classification tasks are known to poorly perform as a result of domain shift [7, 11]. In the former studies, the networks were exclusively trained for image classification tasks. However, for this setting, we will utilise CNNs to solve regression tasks. For regression tasks, we expect to see similar to different results. The difference between image classification and regression in this setting is the effect domain shift will have on either one or both of the domains. We will see that for regression tasks the distribution of the features and labels is different for both the source and target domain. Whereas, an image displaying a traffic sign of a roundabout after inverting to its negative still depicts a traffic sign of a roundabout [12] i.e. only the distribution of the features is different for both domains. Aside from performance, feature transferability can be studied. As is reported by Yosinski et al. [13], the transferability of features decreases as the gap of domain discrepancy increases. As the study was conducted for image classification tasks, it is worth exploring what can be said about feature transferability for regression tasks.

Newly designed synthetic tasks will be developed that are used in all of the experiments. The network's task is to predict the correct, or within a small error margin, a continuous value. Images in the dataset are mapped to their mean, median, variance, and standard deviation pixel intensities. These tasks are considered easy for humans and are common image processing operations, but can they be learned by the network? CNNs have been chosen as means of regressor since they are a popular approach.

To date, a limited number of studies have investigated the capabilities and applications of CNNs in combination with regression tasks [14, 15, 16]. In [16], the network's task was to estimate the age of a person given a picture of their face. It is worth exploring more in-depth, how the performance of CNNs is impacted by domain shift in the realm of regression.

The objective is to study the effect of domain shift on the performance of CNNs using regression tasks. The work will explore to what extent CNNs are capable of learning these newly designed synthetic tasks. Is the network able to learn to predict the correct image statistics? How is the network's performance impacted as a result of domain shift in this particular setting? And, what can be said about feature transfer-

ability for the proposed tasks? For the first 2 questions we will see that 3 out of the 4 image statistics can be sufficiently learned by the network (i.e. beat the set baseline). As for the third question, it may seem like feature transferability might not be as important in the setting of regression tasks.

Together with our peers, we will contribute, albeit limited, to the understanding of how CNNs behave to regression tasks. Hence no new techniques or solutions will be recommended.

2 Background Information

In the section, we briefly discuss the relevant background information. Starting with the topic of CNNs. Followed by the theory on domain shifts. And ending with the related work.

2.1 Convolutional Neural Networks

CNNs are specific types of deep learning models that act on data from a regular grid (e.g. images), with the goal to detect and learn low-level patterns, such as edges and shapes, to higher-level patterns like objects that are embedded as features in the spatial structure of the input [17]. Here, we will solely focus on vanilla, shallow CNNs. For more detailed information about the inner workings and computational processes of CNNs please consult one or more of the following [18, 19].

2.2 Domain Shift

An example of domain shift, for a supervised task, is to correctly identify exhibits in museum spaces [20]. Images in the source domain are centered and photographed in non-adverse conditions [20]. Subsequently, images in the target domain display the same exhibits, only this time the images were captured in an egocentric fashion, resulting in new angles, viewpoints, lighting, additional glare, transparency, and clutter [20]. The distribution shift is likely to cause the network to preform bad on the target domain.

Domain Shift is a recurring generalisation problem in machine learning [21]. The nature of the existence of this problem arises from a discrepancy between the source domain and the target domain. Both domains are labeled, these labels we will call from hereinafter: targets. Due to a different distribution of the target domain, compared to that of the source domain, the network may not be able to perform well on the test dataset. While both domains capture the same kinds of objects.

The domains each consist of a feature space $\mathcal{X}$ and target space $\mathcal{Y}$. Sample instances of these spaces are respectively denoted as: $\mathcal{X} = \{x_1, ..., x_n\}$ and $\mathcal{Y} = \{y_1, ..., y_n\}$ [10]. Now, the goal for the network is to learn the correct mapping function $f(\cdot)$, such that: such that $f : \mathcal{X} \rightarrow \mathcal{Y}$ [10]. Next, the correct model is learned by feeding the network samples from the source domain denoted by: $S = \{(x_i, y_i)\}_{i=1}^{n}$, which are the product of $\mathcal{X} \times \mathcal{Y}$ [22]. After training the model is evaluated on the samples of the target domain, denoted by: $T = \{(x_i, y_i)\}_{i=1}^{n}$ [22].

2.3 Related Work

To the best of our knowledge, the performance impact of domain shifts for CNNs, is not studied for these research-specific synthetic tasks (i.e. map images to their mean, variance, standard deviation, and median value). The aforementioned works [7, 11], we deem the most related to this specific work. The work that has been done in both studies was under the same denominator: measure the performance impact of domain shifts in CNNs. Both studies have been conducted for medical datasets, and equal performance drops have been reported.

Hosseini et al. [12] explored the limits of CNNs, and whether these would be capable of capturing the semantics of the training data. This was done by training the network on regular, non-transformed images, and evaluating the model on negative images. Despite the images retaining their initial structure and semantics after transformation, the model's accuracy showed a decrease of up to 60% [12].

In the three studies mentioned [7, 11, 12], image classification tasks were utilised. Even though loss and accuracy are not a one-to-one relationship, for our regression tasks, we expect to see similar results.

Because of domain discrepancy, the transferability of features in higher layers decreases [23]. Yosinski et al. [13] discussed the strong dependence between the chosen dataset and task, and the features computed in the higher layers of the trained network. Again, the experiments in [13, 23] involved image classification. In the setting of regression tasks, the semantics of an image, and the ordering of the pixels are not of importance anymore. This will only hold if the pixel values are not changed by transforming the images or reshuffling the pixels. Thus, we expect domain discrepancy to be alleviated in case of regression, due to the absence of those two factors.

3 Experimental Setup

In this section, we briefly go over the experimental setup. We will talk about the datasets used, the synthetic tasks, domain shifts, baselines, the architecture and parameters, and lastly training and evaluation. In that specific order.

3.1 Datasets

The MNIST database [24] was used as a dataset in all experiments. This dataset was chosen since we know what the images are. The database consists of 28×28 grayscale images of handwritten digits. The 60.000 training images were split into 40.000 training images, and 20.000 validation images. These two datasets are the source domain. 10.000 images were used for testing the model, the target domain. All images were normalised before the new targets (mean, median, variance, standard deviation) were calculated and used in the experiments. Normalisation makes sure the data have a zero mean and unit variance [25].

In addition, artificial images, consisting of just noise were created. These images were used in experiment 5. The images had a mean, median, variance, and standard deviation that was within the range for each of the statistics of the original images in the test dataset.

3.2 Synthetic tasks

Newly created synthetic tasks were used in the experiments, these were: mapping images to their mean, median, variance and standard deviation, pixel intensities. These statistics are easy to compute, and so to control for. For all of the statistics, the following notations hold:

- n : the number of total pixels in the image: $28 \cdot 28 = 784$
- y_i : the target (mean, median, etc.) for the image i in the dataset
- x_i : the image i in the dataset

Hereafter, a mathematical notation of all four statistics is provided:

- **Mean:** $y_i = mean(x_i) = \frac{1}{n}\sum_{j=1}^{n} z_j$, such that: z_j is pixel j in image x_i.
- **Median:** $y_i = median(x_i) = \{(n+1)/2\}^{th}$value in the list of all pixel values.
- **Variance:** $y_i = var(x_i) = \frac{1}{n}\sum_{j=1}^{n}(X - z_j)^2$, such that: X is the mean pixel intensity of image x_i.
- **Standard Deviation:** $y_i = std(x_i) = \sqrt{V}$, such that: V is the variance pixel intensity of image x_i.

3.3 Domain Shift

For this instance, data transformations were applied to both the training dataset and the test dataset. Transformations were used as a tool to create different distributions of the features and targets for both domains. 3 different kinds of transformations were individually applied, either on the training data or on the testing data, see figure1 for visuals of the transformations. These transformations were arbitrary chosen. It was important to have at least two different types of transformations e.g. one related to geometry transformations, and one to the image pixel values. Evaluation of the tasks; how well they performed, are measured by the Mean Absolute Error (L1) loss function. In (1), n is the total number of examples in the test dataset. The training, validation, and test loss are calculated by the L1 loss.

$$\text{L1} = \frac{\sum_{i=1}^{n} |f(x_i) - y_i|}{n} \tag{1}$$

3.3.1 Rotation

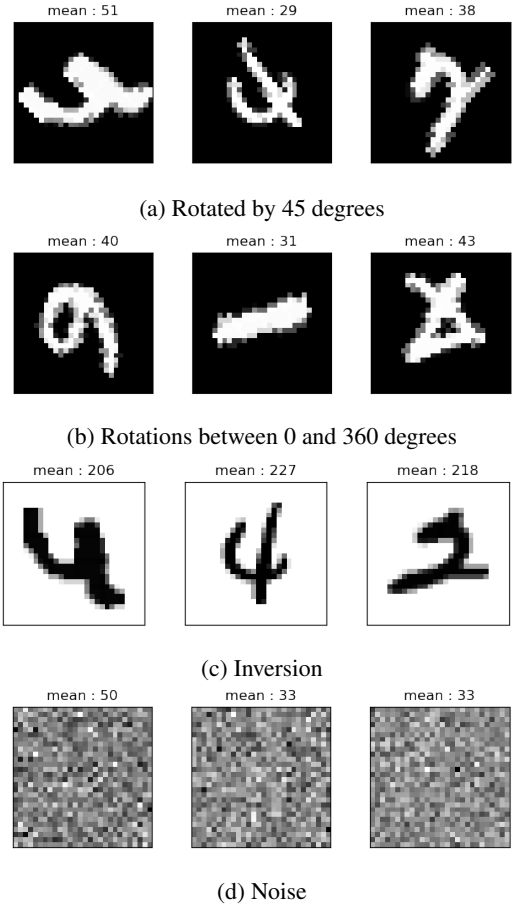
In experiments 1, 2, and 3, the images of either the source domain or the target domain were rotated. In experiments 1 and 2, 45 and 180 degrees rotations respectively, were applied to images in the test dataset. In experiment 3, random angels in the range from 0 to 360, were applied to the images in the training dataset. The applied angles fit to a uniform distribution between 0 and 360 degrees. To explore if it would make any difference to which domain the rotation would be applied, experiment 3 was designed. No other angels, such as 10 or 70 degrees were picked as individual experiments. Since those angles would be too close to 0 or 90 degrees rotations, and thus would not contribute to more insights.

3.3.2 Inversion

Experiment 4, was created as a follow-up to the work of Hosseini et al. [12]. The images of the test dataset were inverted to their negatives: $p' = 255 - p$, also used by [12]. Due to this colour space transformation, the statistics of the images changed accordingly. See figure 2, for the mean pixel intensity distribution shift. Additionally, this experiment would give us more insights into the correlation between the applied image transformation and the performance of the network. Since images retained their original shape and semantics after the image transformation had been applied.

3.3.3 Noise

For experiment 5, we applied the same rotations as we did in experiment 3 on the source domain, for the MNIST dataset. The network was evaluated on the target domain containing noisy images, as described in the section 3.1. This experiment had as purpose to tell us more about the importance of structure in the image, and subsequently the feature transferability within the networks for the synthetic tasks.



(a) Rotated by 45 degrees

(b) Rotations between 0 and 360 degrees

(c) Inversion

(d) Noise

Figure 1: From top to bottom: first we see (a), images of the target domain, used in experiment 1; second, we have (b), depicting the images of the source domain, used in experiment 3; third, we have (c), images of the target domain, used in experiment 4; and lastly we have (d), images of the target domain, used in experiment 5. 180 degrees rotated images have been omitted in this visual.

3.4 Baseline

A baseline was set in all experiments to compare the network's performance and draw conclusions. First, we took a prediction constant $\hat{y}_{const} = median(y_1^{test}, ..., y_n^{test})$. In experiments 1, 2, 3, and 5, the prediction constant was calculated for the targets in the test dataset. In experiment 4, we took the targets in the training dataset $(y_1^{train}, ..., y_n^{train})$ to calculate the prediction constant. Since we work with the L1 loss, we took the median value of each of the statistics. Next, in all 5 experiments, for each of the statistics, the baseline was calculated before each batch of runs, with randomly initialised weights in the network (denoted as rw) by: $y_{rw} = L1(y_i^{test}, \hat{y}_{const})$, for $i = 1, ..., n$ and y_i as targets of the test dataset. Lastly, we calculated the test loss, on the trained network, for each task, with: $\hat{y}_{train} = L1(y_i^{test}, \hat{y}_{pred})$. If $\hat{y}_{train} < \hat{y}_{rw}$, the network had sufficiently learned the task by outperforming the baseline.

3.5 Architecture and Parameters

We used the network architecture[1] of [26] but only used one output node as we have a regression task. No additional modifications were made. For more information about the network and settings, see Appendix A.

The overall aim was to keep the model small and concise for two reasons. First, we work with a relatively small dataset and tasks that are not too demanding for the network. Second, by keeping the model small, it should make it more straightforward to reason about the network's abilities and therefore to comprehend the results.

The (hyper)parameters that were used can be found in Appendix A. Before the actual experiments were carried out, the parameters and settings were fine-tuned through trial-and-error. The Adaptive Momentum Estimation (Adam) optimizer was used and is known to be robust and give good results quickly [27].

3.6 Training and Testing

To measure the impact of domain shift on the performance of the network, 5 different experiments were conducted. During all experiments, the training and testing setup remained the same. Each experiment, in turn, consisted of 4 sub-experiments, one for each of synthetic tasks respectively. The sub-experiments were each executed 10 times, for each run a unique random seed. For run *i*, the random seed was set to *i*. This way we could examine how consistent the network would be for the returned test losses. The training of the network was done for a maximum of 40 epochs. This was an arbitrary chose, keeping in mind the computational limits of the machine. To prevent the network from overfitting the training data, early stopping was implemented. The network would stop training after registering 5 times an increased validation loss. This number was arbitrarily chosen. In all the experiments, the training batch size was set to 64. For predicting the mean, standard deviation, and median pixel intensities of the images, a learning rate of 5e-07 was used throughout all of the experiments. In the case of the variance, the learning rate was set to 3.7e-06 instead and again used for all of the 5 experiments.

[1]The code that was used for training, testing, and plotting is available on https://github.com/JulianBiesheuvel/Research_Project_CSE3000

4 Results

The results of each individual experiment will be presented in this section. Table 3 and Table 2 are a compression of multiple tables, see for more information Appendix B.

4.1 Experiment 1: Trained on non-transformed images, tested on 45 degrees rotated images.

For this experiment, images in the target domain were 45 degrees rotated, no transformations were applied on the images in the source domain. As can be seen in Table 3, with exception of the median pixel intensity, the network managed to learn to predict the mean, standard deviation, and variance pixel intensity, as for all three the average test loss outperforms the baseline. From Table 3, it can be concluded that the network was the most consistent for predicting the standard deviation pixel intensities, and the least consistent for the mean pixel intensities, over all the 10 runs. In addition, this shows that the network is not biased towards smaller numbers, as the variance, with a significantly higher average test loss, is more consistent.

4.2 Experiment 2: Trained on non-transformed images, tested on 180 degrees rotated images.

For this experiment, images in the target domain were 180 degrees rotated, no transformations were applied on the images in the source domain. As it was the case for experiment 1, the mean, standard deviation and the variance are here once again outperforming the set baseline (Table 3). Also in Table 3, it can be seen that all the average test losses for all four statistics have outperformed the test losses for the statistics in experiment 1. This observation can be explained by the applied rotation. Images containing a digit 8, will after rotation resemble the digit 8, the same goes for the digit 1. However, compared to experiment 1, the coefficient of variation for the standard deviation is lower, suggesting it was even easier for the network to be consistent for this task.

4.3 Experiment 3: Trained on randomly rotated images, tested on non-transformed images.

For this experiment, images in the source domain were rotated between a random angle of 0 and 360 degrees, no transformations were applied to the images in the target domain. As can be concluded from Table 3, we see little to no difference compared to the results observed in experiment 1 and 2. For the mean, standard deviation and median task, the network turned out to be less consistent. This supports the idea that it indeed matters to what domain the rotation transformation is applied. In learning to predict the median pixel intensity, it did not make any difference to what domain the rotation was applied, as the average test loss did not outperform the baseline.

4.4 Experiment 4: Trained on non-transformed images, tested on negative images.

For this experiment, images in the target domain were inverted to their negatives, no transformations were applied to the images in the source domain. As a product of the applied colour space transformation, the pixel values of the image changed accordingly. Whereas this was not the case in experiment 1, 2, and 3. See figure 2 for the mean pixel intensity distribution shift, for the the targets in the test dataset. Most notable depicted by Table 3, is that only the mean task outperformed the set baseline. However, in Table 1 we see that the network was actually capable of outperforming the baseline for the median pixel intensities a few times. This is interesting, since the network was not capable of doing so for the median pixel intensities in experiment 1, 2 and 3. If we compare these results found by Hosseini et al. [12], keeping in mind that the study entailed an image classification task, we can generally conclude that regression tasks show similar performance drops.

4.5 Experiment 5: Trained on randomly rotated images, tested on noisy images.

For this experiment, images in the source domain were rotated between a random angle of 0 and 360 degrees, and images in the target domain were artificially created consisting of just noise. Table 3, shows that none of the sub-experiments has outperformed the baseline for that task. However, if we take a look at Table 4, we see that for the mean, standard deviation and variance the network was actually capable of outperforming the baseline in nearly half of the runs. Consequently, the coefficients of variation turn out to be a lot higher. Nevertheless, it is fascinating to see that the spatial structure, such as edges and shapes, seem to be of lesser importance in these synthetic tasks in this setting.

4.6 Convergences & Performance non-shifted task

Table 2 depicts the average number of epochs and standard deviation that was needed for the network to converge, according to our set early stopping criteria. Generally speaking, the network had a not direct one-to-one interrelationship between Table 2 and Table 3. This is supported by values shown for experiment 4 in Table 2. These are nearly similar to those of experiments 1, 2 and 3, for which the sub-experiments indeed outperformed the baseline, whereas this is not the case for experiment 4. Overall, we assume that predicting the median pixel intensity, in case of experiment 1, 2, and 3, would not benefit from more epochs, to get test losses that beat the baselines. As this would potentially lead to overfitting the network.

After each epoch, the learned model was evaluated on the validation dataset. Since this dataset originates from the training dataset, there were no distribution shifts. Therefore, validating the network can be considered as a non-shifted task. By judging from the plotted training, validation and test losses (Appendix C), we found about equal performance between the validation loss and the training loss. This suggests the network has learned the right model for the task.

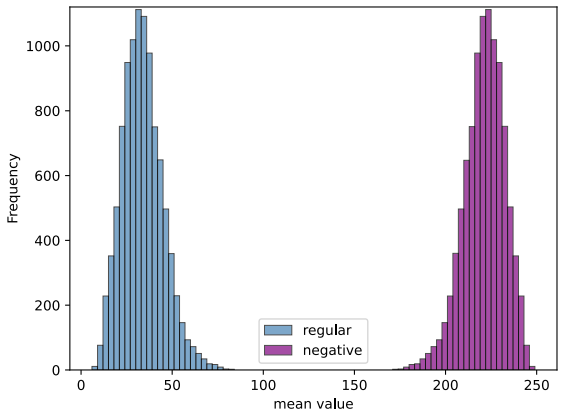


Figure 2: This plot depicts the distribution shift for the targets in the test dataset. On the left we have the original mean pixel intensity distribution of the non-transformed images in the target domain. On the right we have the new distribution of the targets in the test dataset, after the images were inverted.

Median		
$\hat{y}_{rw}$: 255		
Run	$\hat{y}_{train}$	#Epoch
1	259.13	29
2	258.46	22
3	253.54	27
4	250.48	11
5	256.23	15
6	259.27	23
7	260.68	38
8	263.98	33
9	251.14	26
10	252.30	17

Table 1: The test losses ($\hat{y}_{train}$), and the baseline ($\hat{y_{rw}}$), are depicted for each image statistic in experiment 4. For the runs in which: $\hat{y}_{train} < \hat{y_{rw}}$, the test losses are highlighted. This table is part of a larger table that can be found in B

	Number of Epoch Before Halt							
	Mean		**Standard Deviation**		**Median**		**Variance**	
Experiment	μ	σ	μ	σ	μ	σ	μ	σ
1	30	10	30	6	24	8	32	10
2	30	10	30	6	24	8	32	10
3	29	10	29	7	29	9	26	9
4	30	10	30	6	24	8	32	10
5	31	8	31	9	27	8	30	9

Table 2: For each experiment and all image statistics, the average (μ) and standard deviation (σ) number of epochs before the network halts is depicted. This table is a compression of the 5 tables, one for each experiment, that can be found in Appendix B.

	Test Loss (L1)															
	Mean				**Standard Deviation**				**Median**				**Variance**			
Experiment	$\hat{y}_{train}^{avg}$	σ	C_v	$\hat{y}_{rw}$	$\hat{y}_{train}^{avg}$	σ	C_v	$\hat{y}_{rw}$	$\hat{y}_{train}^{avg}$	σ	C_v	$\hat{y}_{rw}$	$\hat{y}_{train}^{avg}$	σ	C_v	$\hat{y}_{rw}$
1	3.6	1.0	0.28	8.8	5.4	1.0	0.18	9.6	1.8	0.3	0.19	0.0	472	113	0.24	1485
2	2.8	1.0	0.34	8.8	3.8	0.6	0.16	9.7	1.7	0.4	0.23	0.0	359	118	0.33	1498
3	2.9	1.0	0.35	8.8	5.0	1.8	0.35	9.7	1.6	0.5	0.29	0.0	438	101	0.23	1498
4	157.3	11.5	0.07	189.2	37.4	12.4	0.33	9.7	256.5	4.5	0.02	255.0	7291	998	0.14	1498
5	10.2	3.8	0.37	9.1	10.3	8.1	0.79	9.8	1.3	0.4	0.31	0.0	1621	1100	0.68	1518

Table 3: This table is a compression of the 5 tables, one for each experiment, that can be found in Appendix B. The average test losses ($\hat{y}_{train}^{avg}$), over 10 runs, are depicted together with the standard deviation (σ) of the test losses. The *coefficient of variation* is calculated as such: $C_v = \frac{\sigma}{\hat{y}_{train}^{avg}}$. The coefficient of variation tells us how consistent the network was i.e. was there a lot of variability within the 10 test losses? As a fourth measure, the baseline ($\hat{y}_{rw}$) for each sub-experiment was given, to compare with the average test loss. If $\hat{y}_{train}^{avg} < \hat{y}_{rw}$, we can conclude the network outperformed the baseline.

Mean			**Standard Deviation**			**Median**			**Variance**		
$\hat{y}_{rw}$: 9.12			$\hat{y}_{rw}$: 9.83			$\hat{y}_{rw}$: 0.0			$\hat{y}_{rw}$: 1517.64		
Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch
1	12.61	40	1	10.34	32	1	1.39	37	1	813.64	26
2	14.53	40	2	19.21	40	2	0.93	39	2	2346.90	40
3	9.94	40	3	26.74	25	3	1.34	30	3	736.67	25
4	15.94	31	4	3.38	23	4	0.96	23	4	3106.85	40
5	6.92	20	5	4.75	40	5	1.29	15	5	792.86	26
6	6.93	19	6	3.65	16	6	2.21	15	6	2595.30	40
7	14.30	30	7	17.36	31	7	1.37	32	7	3376.52	40
8	7.92	28	8	3.60	21	8	1.02	27	8	985.66	18
9	7.71	40	9	6.32	39	9	1.66	27	9	778.08	25
10	5.37	26	10	7.17	38	10	0.91	22	10	672.69	19

Table 4: Depicted are: the test losses ($\hat{y}_{train}$), the number of epochs till convergence, and the baseline ($\hat{y_{rw}}$), for each image statistic in experiment 5. For the runs in which: $\hat{y}_{train} < \hat{y_{rw}}$, the test losses are highlighted.

5 Discussion & Limitations

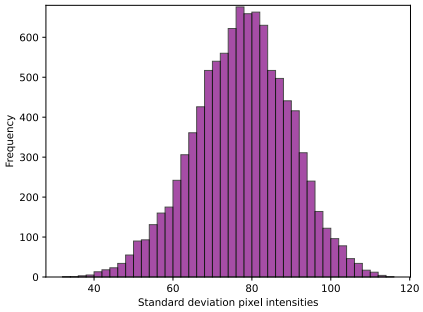
The goal was to determine whether CNNs were invariant to domain shifts, while learning different kinds of image statistics. We will highlight the most notable findings and point out the limitations.

First of all, the common denominator in experiments 1, 2, 3, and 5 was the inability of the network to outperform the baseline in predicting the median pixel intensity. Why the network is not capable of learning the median value in these cases, is not clear to us. Experiment 4, was the only experiment in which there was a distribution shift in both the source and target domain. As said earlier, experiment 4 had some successes predicting the median pixel intensity by beating the baseline. Therefore it might be the case that the network's filters are more favoured towards higher pixel values for predicting the median pixel intensity.

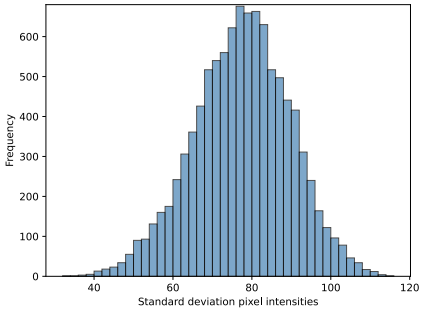
Second, the inability of the network to outperform the baseline for the standard deviation and variance pixel intensities in experiment 4 is worth mentioning (Table 3). The poor performance of the standard deviation can be due to the poor performance of the variance. Since the variance is used in calculating the standard deviation. Based on the baselines for each statistic in experiment 4, we can only speculate about the recorded test losses. One guess is that the network with the randomly initialised weights has a bias towards predicting smaller values in case of this experiment.

Third, in experiment 5 the importance of spatial structure, such as edges, shapes, and objects were studied. As the results showed us, given the setting in which the experiments were conducted, the structure and ordering of the pixels in the image seem not to be dominant in how well the network performs. The importance of spatial structure and ordering seems to be curbed for these tasks. This is backed by the results obtained in experiment 4. Since the distributions for the standard deviation and variance did not shift by inverting the images, see figure 3, where it was the case for the mean distribution (2), we can compare these statistics to the standard deviation and variance in experiment 5. From this we can see that while the images in experiment 4 retain their original semantics, shapes and structure, the network performed on average worse, compared to the average test losses for the standard deviation and variance in experiment 5. Even though, the network was more inconsistent in experiment 5, as opposed to experiment 4 (Table 3).

Lastly, the importance of the type of transformation applied to the images are discussed. As the targets are a direct product of the pixel intensities, it matters what kind of application is applied to the images. We have seen that rotation transformations have little to no effect on the initial distributions of the targets (mean, median, etc. pixel intensity). Colour space transformations, such as inverting, do have a significant impact on the distributions of the targets. Not only the feature distributions are shifted, but also the target distributions, in whatever domain we are operating. This is why we might have seen bad results in case of experiment 4, since the distributions of the features and targets are different in both domains. This statement can be supported by the nature of experiment 5. In this experiment, only the feature space encountered a big distribution shift: from rotated images to noisy images, but the target distribution between the two domains were nearly identical. Still, this yielded satisfactory results in experiment 5.



(a) Distribution inverted images



(b) Distribution non-transformed images

Figure 3: Both plots show the distribution of the standard deviation pixel intensity for images in the test dataset. In (a) we see the distribution for negative images (inverted). In (b) we see the original distribution of the test dataset, before any transformation was applied to it. (a) and (b) are of the same shape. As the standard deviation is calculated by the variance, we therefore can say, without showing a plot, that the distributions are equal for the variance as well.

5.1 Limitation

Although the MNIST database is easy and fast to use in projects, and it is easy to see what the data is (e.g. digits), it also comes with the drawback of interpretability of the results. Instead of 2D data, we could have opted for 1D data instead. In addition, the data could be optionally crafted by ourselves, to have more control over it. It is worth recreating the experiments in similar fashion, but without using the MNIST database, and just use our own data and see if we obtain similar results.

For experiment 5 we would have liked to conduct one additional experiment in which we would train the network on noisy images, and evaluate the network on non-transformed MNIST images. To have more certainty about our outcomes and conclusions.

Moreover, just one dataset and one network were used in this work. This is in itself already a limitation. Network architectures that might be simpler or more complex, with for example dropout in order to generalise better, have an advantage over the network used here.

Since we worked with synthetic tasks, the results found here do not say anything about other regression related tasks i.e. it does not say something about it's generalisation to other tasks.

6 Conclusion

The goal of this work was to investigate if convolutional neural networks were invariant to domain shifts, while their task was to learn to predict image statistics. We measured this by the network's performance. First, we applied different kinds of transformations (e.g. geometric and colour space transformations) to either the source domain or target domain, or we created artificial data ourselves to work with. These data transformations and new images were utilised to achieve a distribution shift between the source and target domain. Integral to this research question, was the question if the network would be able to learn the synthetic tasks we newly designed for this work. The tasks encompassed predicting the mean, median, standard deviation and variance pixel intensities of the images in the dataset. We found that not all statistics could be sufficiently learned by the network. As an additional subquestion, we explored the importance of spatial structure and ordering of the images to these synthetic tasks. The findings suggest that the importance of spatial structure is questionable.

Lastly, the type of image transformation applied to the images, does seem to matter to the performance of the network. Geometric transformations had less notable impact than colour space transformations had.

Preferably more work is needed to investigate and understand how convolutional neural networks behave to regression tasks, and how the results compare to what we found here. Since the scope of our work has been limited, we can not say anything about how well this concept translates to other kinds of regression tasks.

7 Responsible Research

This research had been conducted for the Delft University of Technology, for the Course CSE3000. The code and data are available online for reproducibility purposes hereby following the guidelines recommended by the Netherlands Code of Conduct for Research Integrity [28]. In addition, the code that was used as sample code for the research, originating from (personal)blogs, have been cited and people have been credited for their work. The MNIST database is an open online database publicly available on the Internet. The database does not consist of any harmful or privacy-sensitive data. We do not see any potential application to use our contributions that might be harmful to others, or negatively impact their lives.

References

[1] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[4] A. H. Jackson, "Machine learning," *Expert Systems*, vol. 5, no. 2, pp. 132–150, 1988.

[5] D. Heaven, "Deep trouble for deep learning. Artificial-Intelligence researchers are trying to fix the flaws of neural networks," *Nature*, vol. 574, pp. 163–166, 2019. [Online]. Available: https://www.nature.com/articles/d41586-019-03013-5

[6] M. M. Waldrop, "News feature: What are the limits of deep learning?" *Proceedings of the National Academy of Sciences*, vol. 116, no. 4, pp. 1074–1077, 2019. [Online]. Available: https://www.pnas.org/content/116/4/1074

[7] K. Stacke, G. Eilertsen, J. Unger, and C. Lundström, "A closer look at domain shift for deep learning in histopathology," 2019.

[8] P. Haeusser, T. Frerix, A. Mordvintsev, and D. Cremers, "Associative Domain Adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, 2017, pp. 2784–2792. [Online]. Available: https://git.io/vyzrl

[9] W. M. Kouw and M. Loog, "A Review of Domain Adaptation without Target Labels," pp. 766–785, 2021. [Online]. Available: https://doi.org/10.1109/TPAMI.2019.2945942

[10] H. Venkateswara, S. Chakraborty, and S. Panchanathan, "Deep-learning systems for domain adaptation in computer vision: Learning transferable feature representations," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 117–129, 2017.

[11] E. H. P. Pooch, P. L. Ballester, and R. C. Barros, "Can we trust deep learning models diagnosis? the impact of domain shift in chest radiograph classification," 2020.

[12] H. Hosseini, B. Xiao, M. Jaiswal, and R. Poovendran, "On the limitation of convolutional neural networks in recognizing negative images," in *Proceedings - 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017*, vol. 2017-Decem, mar 2017, pp. 352–358. [Online]. Available: http://arxiv.org/abs/1703.06857

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" 2014.

[14] heofanis Kalampokas, leni Vrochidou, G. A. Papakostas, T. Pachidis, and V. G. Kaburlasos, "Grape stem detection using regression convolutional neural networks," *Computers and Electronics in Agriculture*, vol. 186, p. 106220, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169921002374

[15] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," 2017.

[16] R. Rothe, R. Timofte, and L. Van Gool, "Deep Expectation of Real and Apparent Age from a Single Image Without Facial Landmarks," *International Journal of Computer Vision*, vol. 126, no. 2-4, pp. 144–157, 2018.

[17] S. Sharma and S. Sharma, "Activation functions in neural networks," *Towards Data Science*, vol. 6, no. 12, pp. 310–316, 2017.

[18] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320317304120

[19] "Cs231n convolutional neural networks for visual recognition," cs231n.github.io, accessed on 2021, June 7. [Online]. Available: https://cs231n.github.io/convolutional-networks/#pool

[20] P. Koniusz, Y. Tas, H. Zhang, M. Harandi, F. Porikli, and R. Zhang, "Museum exhibit identification challenge for the supervised domain adaptation and beyond," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[21] W. M. Kouw and M. Loog, "A review of domain adaptation without target labels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 3, pp. 766–785, 2021.

[22] T. N. Mateiu, A.-J. Gallego, and J. Calvo-Zaragoza, "Domain adaptation for handwritten symbol recognition: A case of study in old music manuscripts," in *Pattern Recognition and Image Analysis*, A. Morales, J. Fierrez, J. S. Sánchez, and B. Ribeiro, Eds. Cham: Springer International Publishing, 2019, pp. 135–146.

[23] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*. PMLR, 2015, pp. 97–105.

[24] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[25] "Mnist normalization," PyTorch Forums, 06 2019. [Online]. Available: https://discuss.pytorch.org/t/mnist-normalization/49080

[26] C. H. M. Chan, "Pytorch: Real step by step implementation of cnn on mnist," Medium, 04 2020. [Online]. Available: https://medium.com/swlh/pytorch-real-step-by-step-implementation-of-cnn-on-mnist-304b7140605a

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[28] DANS, "Nederlandse gedragscode wetenschappelijke integriteit," Data Archiving and Networked Services (DANS), 06 2020. [Online]. Available: https://www.fbi.gov/news/stories/forging-papers-to-sell-fake-art

Appendix A Model Architecture Convolutional Neural Network

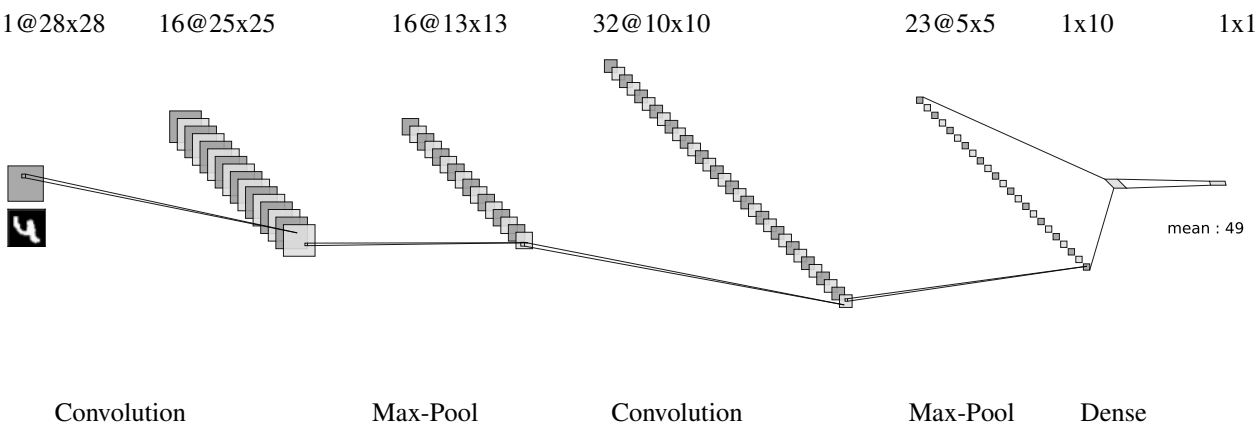


Figure 4: The model architecture that was used in all experiments.

```
Net(
  (cnn1): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1))
  (relu1): ReLU()
  (maxpool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (cnn2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
  (relu2): ReLU()
  (maxpool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=800, out_features=10, bias=True)
  (fc2): Linear(in_features=10, out_features=1, bias=True)
)
```

Figure 5: The (hyper)parameters used in training and testing of the network.

Appendix B Losses & Convergences

For all of the following tables hold: $\hat{y}_{rw}$ is the set baseline, and $\hat{y}_{train}$ the returned test loss (L1) while evaluated the trained network.

Mean			**Standard Deviation**			**Median**			**Variance**		
$\hat{y}_{rw}$: 8.75			$\hat{y}_{rw}$: 9.62			$\hat{y}_{rw}$: 0.0			$\hat{y}_{rw}$: 1484.73		
Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch
1	2.79	40	1	4.40	37	1	1.76	29	1	498.69	33
2	2.95	40	2	4.53	31	2	1.64	22	2	562.91	22
3	4.28	32	3	5.58	40	3	1.64	27	3	666.30	28
4	2.34	29	4	5.47	29	4	1.76	11	4	359.88	40
5	4.69	40	5	7.29	34	5	2.14	15	5	420.15	40
6	2.85	23	6	4.38	23	6	2.11	23	6	582.61	18
7	5.28	18	7	6.40	22	7	2.19	38	7	511.57	18
8	3.12	26	8	4.51	26	8	1.29	33	8	300.90	40
9	3.44	40	9	5.96	32	9	2.39	26	9	372.44	40
10	4.71	15	10	5.70	28	10	1.54	17	10	447.22	40

Table 5: Experiment 1: Trained on non-transformed images, tested on 45-degrees rotated images.

Mean			**Standard Deviation**			**Median**			**Variance**		
$\hat{y}_{rw}$: 8.82			$\hat{y}_{rw}$: 9.72			$\hat{y}_{rw}$: 0.0			$\hat{y}_{rw}$: 1498.11		
Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch
1	2.15	40	1	3.04	37	1	1.36	29	1	368.19	33
2	1.82	40	2	3.29	31	2	1.42	22	2	418.02	22
3	4.52	32	3	4.71	40	3	1.85	27	3	586.96	28
4	1.79	29	4	3.11	29	4	1.33	11	4	234.61	40
5	2.29	40	5	4.20	34	5	1.79	15	5	255.15	40
6	2.80	23	6	4.39	23	6	2.32	23	6	486.08	18
7	4.17	18	7	4.57	22	7	2.20	38	7	413.08	18
8	2.47	26	8	3.42	26	8	1.18	33	8	254.55	40
9	2.62	40	9	3.67	32	9	2.05	26	9	254.29	40
10	3.55	15	10	3.57	28	10	1.55	17	10	319.80	40

Table 6: Experiment 2: Trained on non-transformed images, tested on 90-degrees rotated images.

Mean			**Standard Deviation**			**Median**			**Variance**		
$\hat{y}_{rw}$: 8.82			$\hat{y}_{rw}$: 9.72			$\hat{y}_{rw}$: 0.0			$\hat{y}_{rw}$: 1498.02		
Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch
1	2.11	40	1	3.69	40	1	1.64	23	1	375.49	33
2	2.11	40	2	4.19	23	2	0.98	40	2	420.62	22
3	5.41	28	3	9.41	19	3	1.62	32	3	583.68	21
4	1.96	26	4	4.40	32	4	0.95	40	4	425.89	28
5	3.03	40	5	6.19	37	5	1.89	19	5	336.75	40
6	2.65	27	6	3.73	34	6	1.74	33	6	462.79	23
7	3.35	17	7	4.24	34	7	2.38	18	7	439.92	18
8	2.36	21	8	3.70	21	8	1.17	40	8	263.84	40
9	2.66	38	9	5.30	23	9	1.98	29	9	590.11	24
10	3.53	14	10	5.38	26	10	1.39	17	10	485.49	14

Table 7: Experiment 3: Trained on randomly rotated images, tested on non-transformed images.

Mean			**Standard Deviation**			**Median**			**Variance**		
$\hat{y}_{rw}$: 189.20			$\hat{y}_{rw}$: 9.72			$\hat{y}_{rw}$: 255			$\hat{y}_{rw}$: 1498.11		
Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch
1	156.90	40	1	33.47	37	1	259.13	29	1	7052.71	33
2	150.89	40	2	34.76	31	2	258.46	22	2	6748.63	22
3	179.47	32	3	16.95	40	3	253.54	27	3	5781.65	28
4	137.79	29	4	57.40	29	4	250.48	11	4	8279.89	40
5	160.22	40	5	35.52	34	5	256.23	15	5	8311.17	40
6	153.41	23	6	45.08	23	6	259.27	23	6	6655.73	18
7	167.79	18	7	23.47	22	7	260.68	38	7	6019.74	18
8	160.55	26	8	30.48	26	8	263.98	33	8	8105.52	40
9	146.41	40	9	47.76	32	9	251.14	26	9	8565.63	40
10	159.97	15	10	49.24	28	10	252.30	17	10	7388.10	40

Table 8: Experiment 4: Trained on non-transformed images, and tested on negative images.

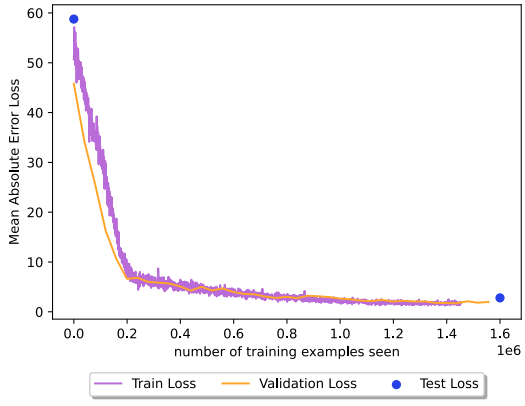
Mean			**Standard Deviation**			**Median**			**Variance**		
$\hat{y}_{rw}$: 9.12			$\hat{y}_{rw}$: 9.83			$\hat{y}_{rw}$: 0.0			$\hat{y}_{rw}$:1517.64		
Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch	Run	$\hat{y}_{train}$	#Epoch
1	12.61	40	1	10.34	32	1	1.39	37	1	813.64	26
2	14.53	40	2	19.21	40	2	0.93	39	2	2346.90	40
3	9.94	40	3	26.74	25	3	1.34	30	3	736.67	25
4	15.94	31	4	3.38	23	4	0.96	23	4	3106.85	40
5	6.92	20	5	4.75	40	5	1.29	15	5	792.86	26
6	6.93	19	6	3.65	16	6	2.21	15	6	2595.30	40
7	14.30	30	7	17.36	31	7	1.37	32	7	3376.52	40
8	7.92	28	8	3.60	21	8	1.02	27	8	985.66	18
9	7.71	40	9	6.32	39	9	1.66	27	9	778.08	25
10	5.37	26	10	7.17	38	10	0.91	22	10	672.69	19

Table 9: Experiment 5: Trained on randomly rotated images, tested on noisy images.

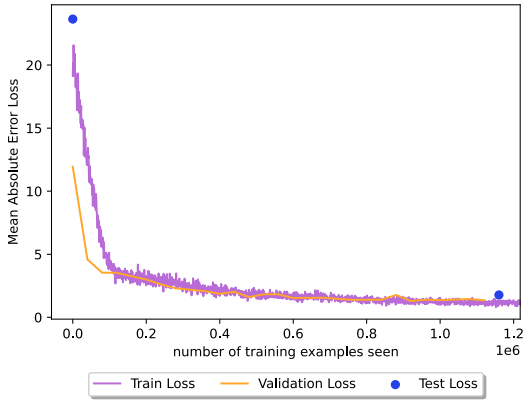
Appendix C Training, validation, and test-losses

The graphs are captured at the first run, out of 10, for each sub-experiment individually.

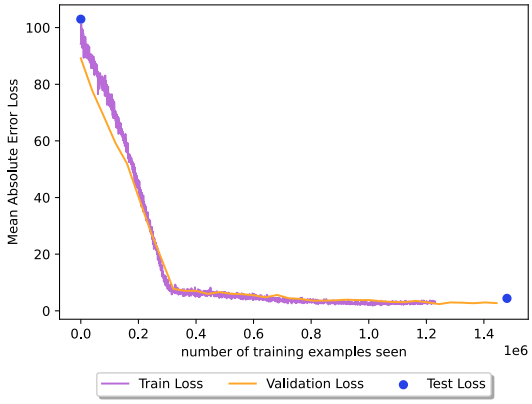
C.1 Plots Experiment 1



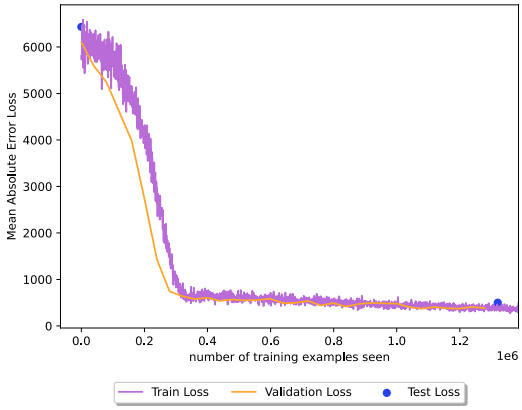
(a) Predicting the mean pixel intensity



(b) Predicting the median pixel intensity



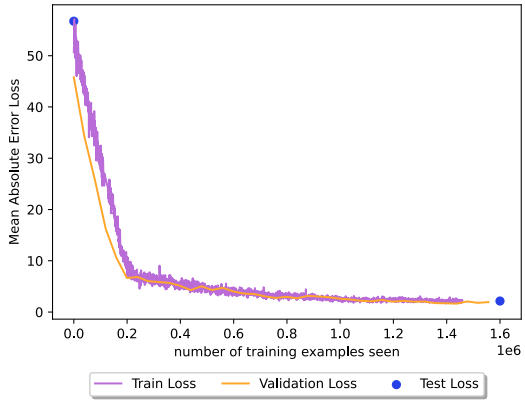
(c) Predicting the standard deviation pixel intensity



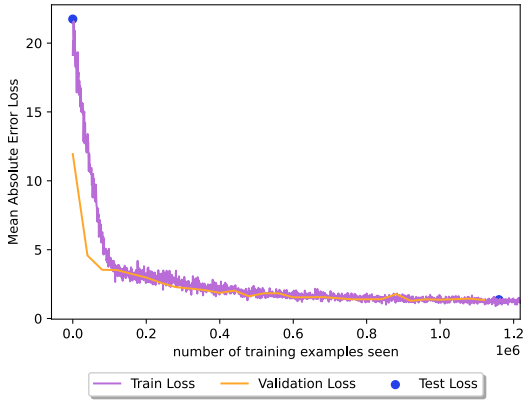
(d) Predicting the variance pixel intensity

Figure 6: For this experiment, images in the target domain were 45 degrees rotated, no transformations were applied on the images in the source domain.

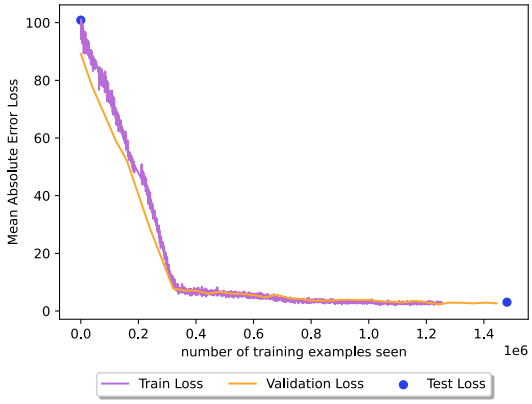
C.2 Plots Experiment 2



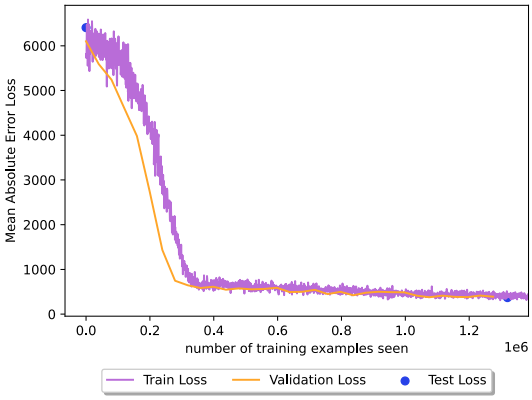
(a) Predicting the mean pixel intensity



(b) Predicting the median pixel intensity



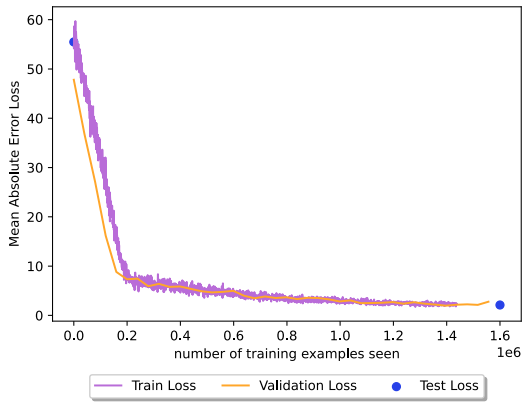
(c) Predicting the standard deviation pixel intensity



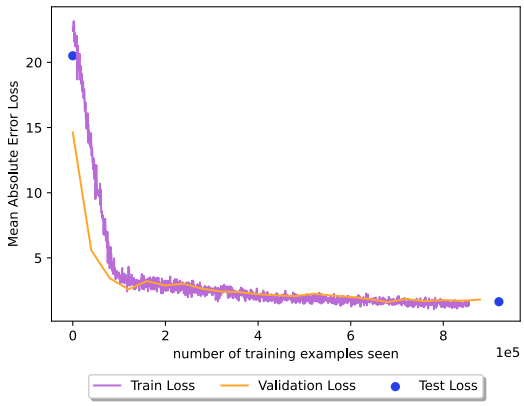
(d) Predicting the variance pixel intensity

Figure 7: For this experiment, images in the target domain were 180 degrees rotated, no transformations were applied on the images in the source domain.

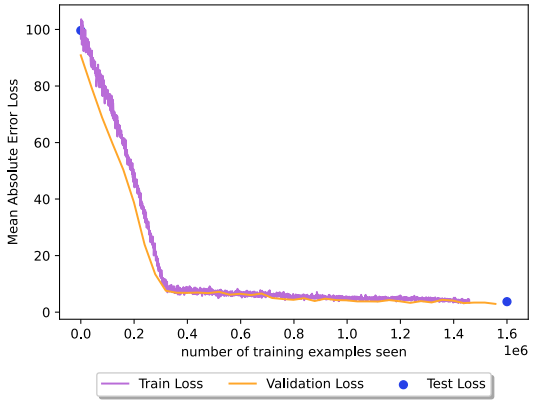
C.3 Plots Experiment 3



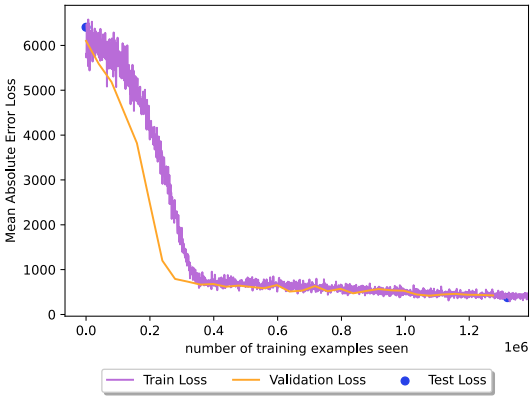
(a) Predicting the mean pixel intensity



(b) Predicting the median pixel intensity



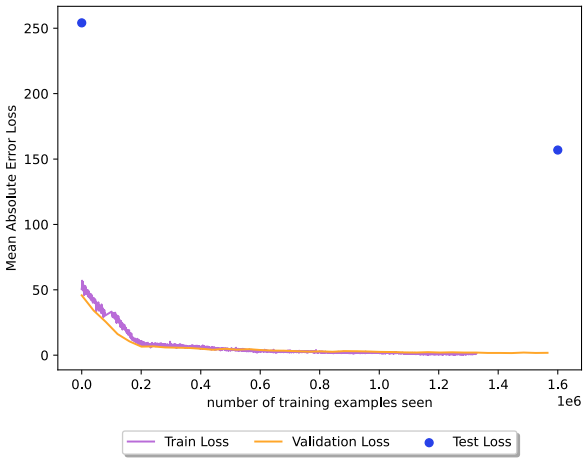
(c) Predicting the standard deviation pixel intensity



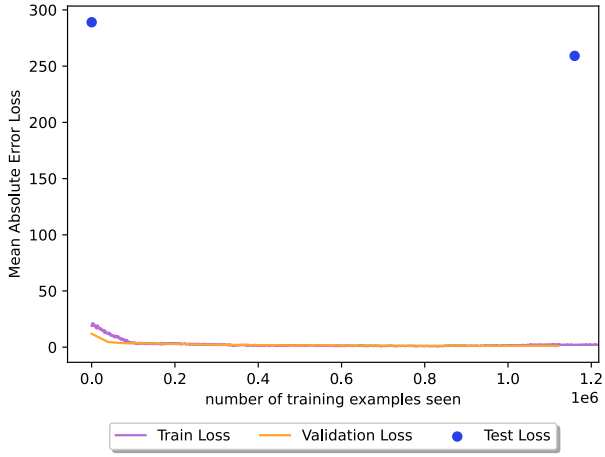
(d) Predicting the variance pixel intensity

Figure 8: For this experiment, images in the source domain were rotated between a random angle of 0 and 360 degrees, no transformations were applied to the images in the target domain.

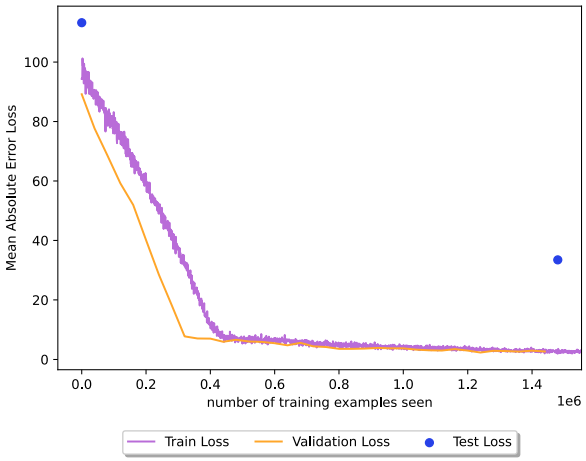
C.4 Plots Experiment 4



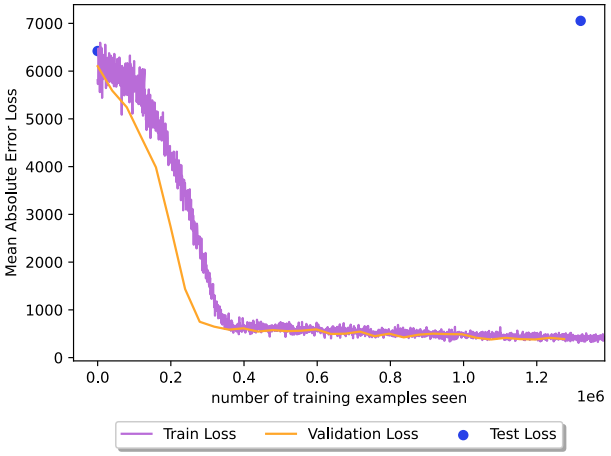
(a) Predicting the mean pixel intensity



(b) Predicting the median pixel intensity



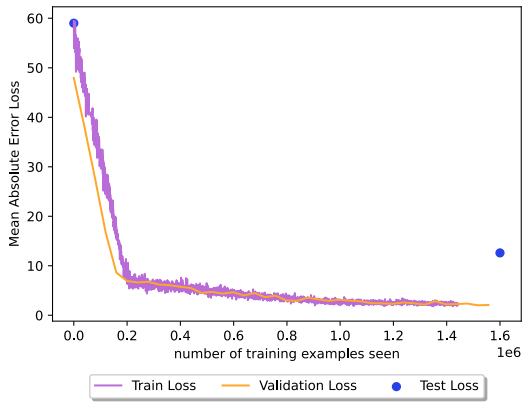
(c) Predicting the standard deviation pixel intensity



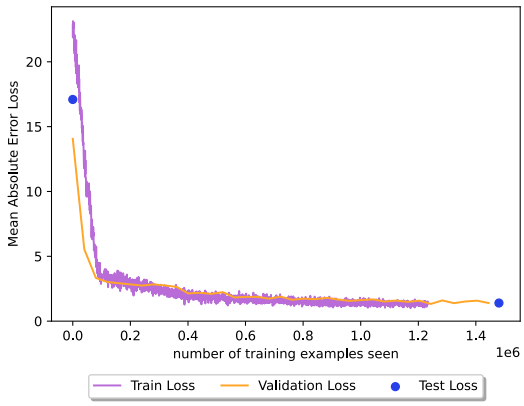
(d) Predicting the variance pixel intensity

Figure 9: For this experiment, images in the target domain were inverted to their negatives, no transformations were applied to the images in the source domain.

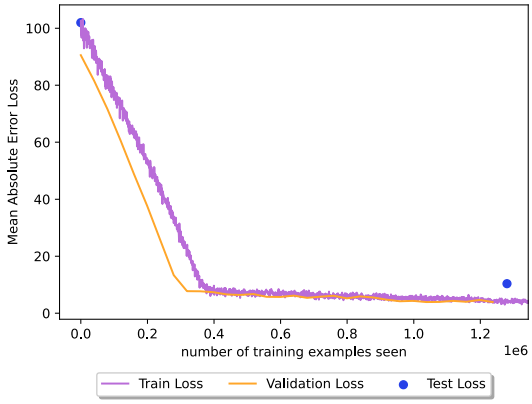
C.5 Plots Experiment 5



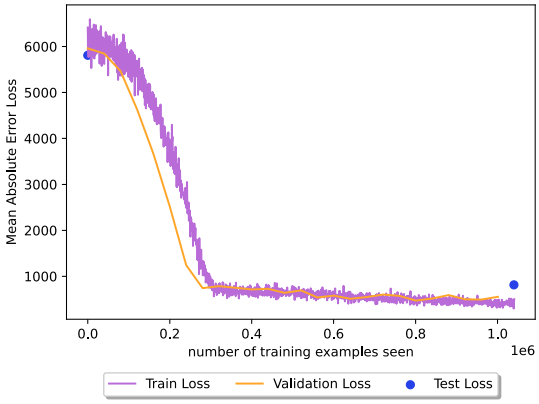
(a) Predicting the mean pixel intensity



(b) Predicting the median pixel intensity



(c) Predicting the standard deviation pixel intensity



(d) Predicting the variance pixel intensity

Figure 10: For this experiment, images in the source domain were rotated between a random angle of 0 and 360 degrees, and images in the target domain were artificially created consisting of just noise.