Global Pooling, More than Meets the Eye: Position Information is Encoded Channel-Wise in CNNs

Md Amirul Islam* [1,6] Matthew Kowal* [2,6] Sen Jia[4] Konstantinos G. Derpanis[2,5,6] Neil D. B. Bruce[3,6]
[1]Ryerson University, Canada [2]York University, Canada [3]University of Guelph, Canada
[4]Toronto AI Lab, LG Electronics [5]Samsung AI Centre Toronto, Canada [6]Vector Institute for AI, Canada
amirul@cs.ryerson.ca, {m2kowal,kosta}@eecs.yorku.ca, sen.jia@lge.com, brucen@uoguelph.ca

Abstract

In this paper, we challenge the common assumption that collapsing the spatial dimensions of a 3D (spatial-channel) tensor in a convolutional neural network (CNN) into a vector via global pooling removes all spatial information. Specifically, we demonstrate that positional information is encoded based on the ordering of the channel dimensions, while semantic information is largely not. Following this demonstration, we show the real world impact of these findings by applying them to two applications. First, we propose a simple yet effective data augmentation strategy and loss function which improves the translation invariance of a CNN's output. Second, we propose a method to efficiently determine which channels in the latent representation are responsible for (i) encoding overall position information or (ii) region-specific positions. We first show that semantic segmentation has a significant reliance on the overall position channels to make predictions. We then show for the first time that it is possible to perform a 'region-specific' attack, and degrade a network's performance in a particular part of the input. We believe our findings and demonstrated applications will benefit research areas concerned with understanding the characteristics of CNNs. Code is available at: https://github.com/islamamirul/PermuteNet.

1. Introduction

One of the fundamental ideas behind different neural network architectures [10, 23, 24, 2, 9] is the idea of *invariance*. Given an input signal, an X-invariant operation is one that produces the same output regardless of any change (of some type X) to the input. This property is desirable in a multitude of applications in computer vision, the most obvious being object recognition [17, 10, 23]; the goal being to assign the corresponding image-level label (e.g., *dog*) regardless of where the object is located in the image. This is referred to as *translation* invariance. Another property of operations that is intimately related to translation invariance is that of translation *equivariance*: shifting the input and then passing it through the operation is equivalent to passing the input through the operation, and then shifting the signal.

To achieve invariant neural networks, a common strategy is to use *equivariant* operations on a per-layer basis [5], which then culminates in an *invariant* output. One of the best examples of this can be found in convolutional neural networks (CNNs) for the task of image classification. Following a hierarchy of translation equivariant convolutional layers, CNNs use a global pooling layer to transform the 3D (spatial-channel) tensor into a 1D vector, which is then fed into a fully connected layer to produce the classification logits. One would therefore (intuitively) assume that after the spatial dimensions are collapsed due to the global pooling operation [18, 24], spatial information should be removed while translation invariance is produced. However, previous work has shown that absolute position information exists both in the latent representations [11] as well as the output of the network [14, 12].

None of these previous works have answered the critical question: How can a CNN contain positional information in the representations if there exists a *global pooling layer* in the forward pass? In this paper, we provide an answer to this question, and demonstrate through rigorous quantitative experiments that CNNs do this by encoding the position information along the *channel* dimension even though the spatial dimensions are collapsed. Moreover, we show that the *position* information is encoded based on the *ordering of the channel dimensions*, while the *semantic* information is largely invariant to this ordering. We argue that these findings are important to better understand the properties of CNNs and to guide their future design.

To demonstrate the real-world impact of these findings, we leverage the fact that position information is encoded channel-wise in a number of domains and applications. First, we tackle the lack of translation invariance of CNNs. We propose a simple yet effective loss function which minimizes

*Equal Contribution

the distance between the encodings of translated images to achieve higher translation invariance. Second, we propose an efficient way to identify which channels in the latent representation are responsible for encoding both (i) position information in the entire image and (ii) 'region-specific' position information (e.g., channels which activate for the left part of the image). We show quantitative and qualitative evidence that networks have a significant reliance on these channels when making predictions compared with randomly sampled channels. Finally, we show it is possible to target region-specific neurons, and impair the performance in a particular part of the image. To summarize, our main contributions are as follows:

- We reveal how global pooling layers admit spatial information via an *ordered* coding along the *channel* dimension. We then demonstrate the real-world applicability of this finding by applying it to the problem domains that follow in this list.
- We propose a simple data-augmentation strategy to improve the translation invariance of CNNs by minimizing the distance between the encodings of translated images.
- We propose a simple and intuitive technique to identify the position-specific neurons in a network's latent representation. We show that multiple complex networks contain a significant reliance on these position-encoding neurons to make correct predictions.
- We show it is possible to attack a network's predictions in a region-specific manner, and demonstrate the efficacy of this approach on a standard self-driving semantic segmentation dataset.

2. Related Work

Islam et al. [11] first demonstrated that *absolute* positional information is captured in a CNN's latent representations. More specifically, they took a frozen pre-trained CNN, and then trained a read-out module to predict a gradient-like position map. They showed absolute position information (e.g., pixel coordinates) could be extracted from many stages of the CNN and that *zero padding* was a significant cause of this information to be encoded. The ability of zero padding to inject position information has been an area of focus in multiple follow-up works [14, 1, 12, 19, 20]. Kayhan et al. [14] demonstrated a number of properties concerning zero padding and positional information, including sample efficiency and generalization to out-of-distribution locations. Mind the pad [1] explored zero padding with respect to object recognition, and demonstrated that it causes severe artifacts in intermediate representation. Finally, Islam et al. [12] performed a large-scale case study on border heuristics, including padding and canvas colors, and showed that positional information can hurt (e.g., texture recognition [4]) or help (e.g., semantic segmentation and instance segmentation [25]) performance depending on the task. *Relative* position information refers to the position of features in relation to one another within an image. One example of how relative position is used in the design of computer vision algorithms is in capsule networks [22], which encode every object with a pose (i.e., translation and rotation). This encoding results in later layer features which activate only for specific relative positions from earlier layers. We view relative position information as a different line of research compared with absolute position and thus focus our analysis on solely the latter in this paper.

Gatys et al. [8] first mentioned absolute position in the latent representations of CNNs in a footnote which comments on how zero padding effects generative modelling. A more recent study explored zero padding in generative models in a much deeper and systematic way and showed that CNNs use it to produce more robust spatial structure [26].

A handful of recent studies showed that CNNs are not fully translation invariant. For example, BlurPool [27] demonstrated that small pixel-level shifts in the input produce large fluctuations in the output classification probabilities. To make CNNs more invariant to translations, they proposed adding a Gaussian blurring layer after each max-pooling layer within a CNN, which significantly improved the translation invariance of the network. Zou et al. [28] extended this work by using content-aware learned filter weights, which predict separate filter weights for each spatial and channel location in the input. While the proposed solution is effective, BlurPool [27] does not explain the underlying mechanisms of *why* CNNs are not translation invariant in the first place. In our paper, we extend beyond the previous works which showed the existence of position information in CNNs, by explaining for the first time the precise mechanism which allows CNNs to encode position information despite the global pooling layer.

3. Channel-wise Position Encoding in CNNs

Recent works [11, 14, 1, 12] showed that CNNs exploit absolute position information. However, no efforts have identified the mechanism in which position information is encoded after global average pooling (GAP) layers. Given the apparent importance of position information, one might raise the question of whether spatial information is being retained by some means. In this section, we aim to answer this interesting question through a series of experiments. We show that absolute position information, despite the spatial dimensions being collapsed, can encode channel-wise after a global pooling layer, in the $1 \times 1 \times C$ latent representation.

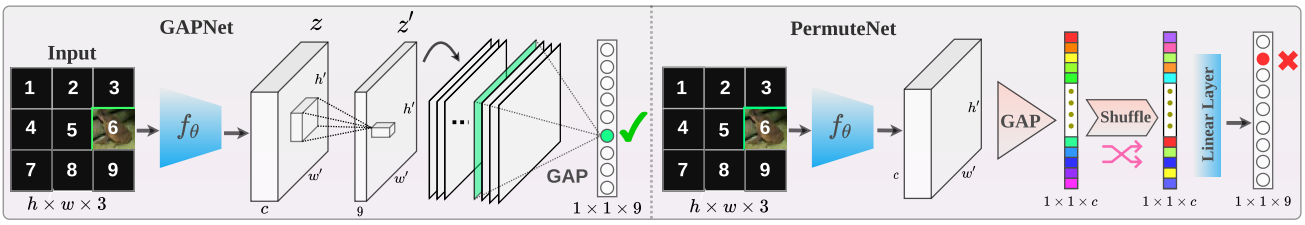

Figure 1. An illustration of our GAPNet (left) and PermuteNet (right) architectures used to determine the existence of channel-wise positional encodings in CNNs. **Left:** We feed a grid based input image to the encoder, f_θ, of standard CNNs (e.g., ResNet-18 [10]) to obtain a latent representation, z. z is then transformed to a representation, z', through the last convolutional layer which has the output channel dimension set to the number of locations in the input grid (e.g., 9 in the above example). This enforces the global average pooling (GAP) layer to output the number of locations. The network is then trained to predict the location of the image patch. **Right:** PermuteNet follows the same structure of a standard CNN except we shuffle the dimensions of the latent representation to verify whether obfuscating the channel ordering hurts the positional encoding capacity.

Padding	Network	Loc. Cls. Acc (%)			Image Cls. Acc (%)		
		3×3	5×5	7×7	3×3	5×5	7×7
Zero	GAPNet	100	100	100	82.6	82.4	82.1
	PermuteNet	78.8	37.8	21.4	73.6	72.2	69.9
Reflect	GAPNet	100	100	100	83.8	83.4	82.9
	PermuteNet	78.3	36.3	23.2	71.1	71.4	65.7
Replicate	GAPNet	100	100	100	83.1	82.9	82.8
	PermuteNet	78.3	40.1	23.6	72.0	71.5	71.4

Table 1. Performance comparison between *GAPNet* and *PermuteNet* in terms of absolute location classification and object recognition under various padding types. Regardless of the padding, it is clear that *GAPNet* achieves 100% positional encoding accuracy while *PermuteNet* fails to correctly classify locations. This demonstrates that positions are encoded channel-wise in the latent representation. Object recognition accuracy further confirms that position information depends on the ordering of the channels while semantic information does not heavily rely on ordering.

3.1. Learning Positions with a GAP Layer

Existing *K-class* object recognition networks [18, 24, 10] largely follow a similar structure including a feature encoder network followed by a classifier. Given an input image, $I \in \mathbb{R}^{h \times w \times 3}$, the encoder network, f_θ, maps the $h \times w \times 3$ input to a latent representation, $z \in h' \times w' \times c$ where $h' < h$, $w' < w$, and $c \gg 3$. The latent representation, z, is then passed to a global average pooling [18] operator which collapses the spatial dimensions of z and outputs $\hat{z} \in 1 \times 1 \times c$ representation. This 1-D representation is finally passed to a linear classification layer which produces the K categorical logits, a set of un-normalized scores representing the likelihood of the object class, $k \in K$ being the label of the current image. As the 1-D representation, $\hat{z}$, has no spatial dimension, the common assumption is that position information is certainly lost.

To demonstrate that the GAP operation can retain absolute positional information of an object, we design two network architectures which we term GAPNet and PermuteNet (see Fig. 1). GAPNet follows a similar structure to a standard CNN (e.g., ResNet-18 [10] and NIN [18]) for object recognition, except we remove the final fully connected layer, such that the last layer of the network is the GAP layer. To ensure the GAP layer outputs the correct number of classes, K, we set the number of output channels of the last convolutional layer (i.e., the layer before the GAP layer) to K. Formally, the last convolutional layer takes the latent representation, $z \in h' \times w' \times c$ as input and outputs a representation, $z' \in h' \times w' \times K$. The intuition of removing the fully connected layer is to enable the GAP layer to predict the class logits. The output of the GAP layer will thus be the same size as the categorical logits and can be used as the last layer of the network (see Fig. 1 left).

PermuteNet also follows the structure of a standard object classification network, except for a single *shuffle* operation which occurs between the GAP layer and the penultimate linear layer. This operation randomly shuffles the channel indices of the representation of the GAP layer, which is then passed to the linear layer for classification (see Fig. 1 right). Note that we construct GAPNet to clearly demonstrate that the output of a GAP layer can directly map to specific absolute positions in the input. We design PermuteNet to show that randomly shuffling the channel ordering hinders the network's ability to encode position information.

3.2. Evaluation of Channel-wise Position Encoding

To validate the existence of a channel-wise position encoding, we design a simple location-dependant task using GAPNet and PermuteNet, such that the output logits can be directly mapped to specific locations in the input image. Inspired by previous works [14, 12], we first conduct a location classification experiment, where each input is a CIFAR-10 [16] image placed on a $n \times n$ grid, where each of the pixels not containing the image is set to *zero* (see *input* in Fig. 1). The target for each input is the location where the image patch is placed (e.g., the target for the given input in Fig. 1 is 6). We use a ResNet-18 [10] architecture to report experimental results for GAPNet and PermuteNet under

Methods	CIFAR-100 [16]		CIFAR-10 [16]		ImageNet [7]		
	Top-1	Cons.8	Top-1	Cons.8	Top-1	Cons.8	Cons.16
ResNet-18 [10]	72.6	70.1	93.1	90.8	69.7	89.5	87.4
+AugShift (ours)	72.6	85.6	92.1	94.8	70.1	90.2	88.2
Blurpool [27]	72.4	78.2	92.5	92.5	71.4	90.5	88.8

Table 2. Comparison of overall classification accuracy and shifting consistency (Cons.) for different networks. Interestingly, our simple *AugShift* technique outperforms the baselines on a smaller dataset (e.g., CIFAR-10 and CIFAR-100) while achieving competitive performance on the large-scale ImageNet dataset. Cons.8 and Cons.16 refer to randomly shifting the input horizontally and vertically by up to 8 and 16 pixels, respectively.

three different padding types. For the location classification task, we train GAPNet and PermuteNet for 20 *epochs* with a learning rate of 0.001, and use the ADAM optimizer [15]. The number of output logits is set to the number of input locations (e.g., 81 for a 9×9 grid). We also train the networks for object recognition using grid based data settings by changing the number of output logits to 10 for both GAPNet and PermuteNet. For the location-dependent object classification task, we train GAPNet and PermuteNet for 100 *epochs* with a learning rate of 0.01.

Results. We present the location classification and object recognition results of GAPNet and PermuteNet in Table 1. For the location classification task, GAPNet achieves a counterintuitive *100% accuracy for all tested grid sizes*. It is clear that the GAP layer can admit robust positional information which directly represents absolute locations on the input image. In contrast, while PermuteNet can learn to identify a marginal number of positions for small gridsizes (e.g., 3×3), the shuffling of the channel dimension significantly degrades the ability of the network to perform location classification as the grid size increases. This provides direct evidence that the *order of the channel dimensions* is the main representational capacity which allows for the GAP layer to admit absolute position information. We further evaluate GAPNet and PermuteNet for an image recognition task using similar data settings and report the results in Table 1 (right). Interestingly, unlike the location classification task, PermuteNet can achieve classification performance close to GAPNet. This reveals an interesting dichotomy between the type of coding CNNs use for *positional* and *semantic* representations: position information depends mainly on the *ordering* of the channels, while semantic information does not.

We show in these experiments that GAP layers can admit position information through the ordering of the channel dimensions. Another interesting question we explore is how much position information can be decoded from *pre-trained* models which are not trained explicitly for location classification. Towards answering this question, we provide more experiments in the supplementary and evaluate the amount of positional information contained in the channel-wise ordering of networks trained for various tasks, such as image classification [10] and semantic segmentation [21]. In further support of our hypothesis, these results also demonstrate unequivocally that GAPNet can recover spatial position information while PermuteNet can not.

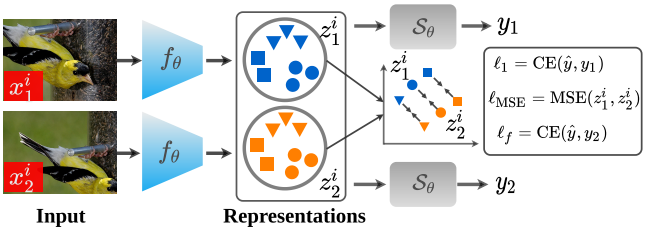

Figure 2. Illustration of the overall training pipeline of our proposed translation invariant model. We use two different crops of the input image, x^i to generate x_1^i and x_2^i, which are then both passed to a convolutional encoder network, f_θ, to obtain latent representations, z_1^i and z_2^i, respectively. We minimize the distance between the two latent representations, z_1^i and z_2^i, through the mean squared error loss which enforces the model to reduce positional bias. The latent representations are also passed to a classification head, $\mathcal{S}_\theta$, which produces the class logits, y_1 and y_2, respectively.

4. Applicability of Channel-Wise Positional Encoding

We now demonstrate multiple ways in which our finding that positional information is encoded channel-wise can be leveraged. To ensure that the channel-wise encoding is the source of improvement for each of these applications, we use the representation after passing it through a GAP layer in each case. First, we propose a simple loss function to improve translation invariance in CNNs. Next, we explore the robustness of object recognition networks. We demonstrate different ways to *target* these models using the implicit positional encodings (i.e., as an adversary) both for overall performance and for a region-specific attack.

4.1. Translation Invariance for Object Recognition

We first tackle the issue of CNNs not being translation invariant. More specifically, recent work [27] showed that small shifts in the input can cause large shifts in the output class likelihoods and proposed BlurPool [27] which applies a Gaussian kernel with every pooling layer of a network.

Learning Translation Invariant Representations. A truly shift invariant network should produce the same output logits, regardless of its shift (i.e., a *cat* is a *cat*, regardless of its *position* in the image). Given that positional information is encoded in the latent representation *preceding* the output logits (i.e., the representation after the GAP layer but before the linear layer), we propose to minimize the difference in this representation between different shifts of the same image. The overall training pipeline is illustrated in Fig. 2. Given two different crops of the same image, denoted as x_1^i

and x_2^i, where i is the image index and the subscript denotes the different crops, we feed these images through the same encoder network, f_θ, and obtain the latent representations, z_1^i and z_2^i, after the *GAP* layer. We enforce them to be shift invariant by minimizing the distance between the latent representations using the mean squared error (MSE) as a loss term. The latent representations are then passed through the prediction layer, $\mathcal{S}_\theta$, which generates the class logits, y_1 and y_2, respectively. We apply two cross entropy losses, ℓ_1 and ℓ_2 between $\{y_1, \hat{y}\}$ and $\{y_2, \hat{y}\}$, respectively where $\hat{y}$ is the object class. Our overall loss function, L_{AugShift}, is then defined as:

$$L_{\text{AugShift}} = \ell_1 + \ell_2 + \lambda \cdot \ell_{\text{MSE}}(z_1^i, z_2^i) \,, \tag{1}$$

where λ refers to the loss weight which controls the contribution of MSE loss.

Shift Invariance and Accuracy. To validate the effectiveness of our proposed training strategy, we show overall performance and shift consistency results on the CIFAR-10 [16], CIFAR-100 [16], and ImageNet [7] datasets. We adopt the same consistency (Cons.) metric as proposed previously [27] to measure the translation invariance. More specifically, we measure how often a network predicts the same category after the input image is vertically and horizontally shifted a random number of pixels: up to 8 pixels (Cons.8) for CIFAR, and 8 or 16 (Cons.16) pixels for ImageNet (results for additional shift magnitudes are reported in the supplementary.

The classification and consistency results are presented in Table 2. Our approach achieves competitive Top-1 accuracy on CIFAR-10 compared to the baseline but significantly outperforms in terms of shifting consistency (94.8% vs. 90.8%). Our approach matches the overall classification accuracy on CIFAR-100 while significantly improving the shifting consistency (85.6% vs. 70.1%). Note that our approach admits greater shift consistency than the BlurPool on the CIFAR datasets. Our method improves the overall performance and the shift consistency on ImageNet compared to the ResNet baseline and remains competitive in terms of performance and consistency with [27]. We believe the performance gap is less (which is true for AugShift and BlurPool) for larger datasets with high resolution (e.g., ImageNet) because the network can learn some degree of translation invariance from the data without any invariance-specific regularization, as proposed. Note that [27] must modify the majority of layers in the CNN architecture by adding another filter at each pooling layer, which in turn adds additional computations at inference time. In contrast, ours is architecture agnostic and adds no computational overhead during inference. Also our motivation for achieving translation invariance differs from Blurpool (i.e., enforcing the similarity of channel-wise position information vs. anti-aliasing) and thus the two techniques may be complementary.

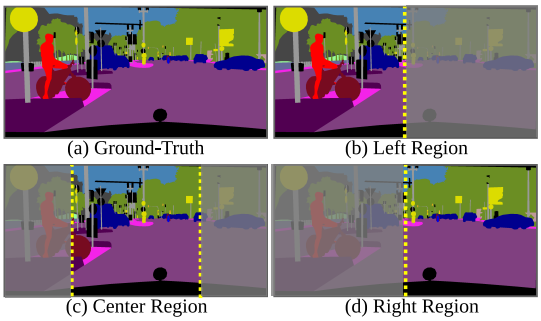
(a) Ground-Truth (b) Left Region (c) Center Region (d) Right Region

Figure 3. Ground-truth samples with different evaluation regions (not highlighted) used in our experiments to demonstrate the degree of performance drop in specific image regions.

4.2. Attacking the Position-Encoding Channels

We now aim to demonstrate that complex networks trained for position-based tasks, such as semantic segmentation [2, 13], have a significant reliance on the position information encoded channel-wise in their latent representations. To perform this type of attack, we first propose a simple and intuitive technique to estimate the position encoding neurons in a CNN's latent representations. A simple modification of the technique allows us to identify the channels which largely encode: (i) overall position or (ii) region-specific position. Then, we evaluate the network's reliance on them for making predictions by simply turning them off during inference. We show that the removal of these neurons causes significantly more harm to the performance than randomly sampled neurons. This suggests a significant encoding of positional information is contained within them. We additionally show the feasibility of performing region-specific attacks on networks trained for autonomous driving tasks. These results demonstrates that channel-wise positional encodings exist in more complex networks and tasks as well as revealing an interesting future direction for adversarial attacks and defences based on position information.

4.2.1 Identifying Position-Specific Neurons

Overall Position-Encoding Channels. Our first goal is to identify the channels in the latent representation of a network which encode the overall position of objects within an image. A simple and intuitive way to estimate the neurons which encode overall position information is by computing the *absolute difference* between the activations of two latent representations of horizontally flipped image pairs. The key intuition of our approach is, given a network which encodes two images where the semantics are identical but the object's position is different, the channels which have small changes in activation are invariant to position, while those that have large changes in activation encode positional information. More formally, given a pretrained encoder for a pixel-wise task (e.g., semantic segmentation), $f(I) = z$,

where $z \in \mathbb{R}^{1\times1\times C}$ is a latent representation after passing it through a GAP layer, our goal is to estimate and rank the most position-specific neurons in z. For a given image, I^a, we simply apply horizontal flipping on I^a to obtain the flipped image, I^b. Note that the only semantic factor that differs between this image pair is the *absolute position* of the objects. We then feed these images to a pretrained network to obtain the latent representations, $\{z^a, z^b\}$. Finally, we compute the *absolute* difference, $|\Delta z_i| = |z_i^a - z_i^b|$ between these two latent representations, where i denotes the sample index from dataset. We calculate $|\Delta z_i|$ for all images, i, in the Cityscapes [6] validation set and average the differences to obtain a position encoding score for each neuron. Finally, we rank the neurons in descending order to obtain an ordered list of the overall position-specific neurons, $\hat{z} \in \mathbb{R}^{1\times1\times C}$, where the first element has the highest amount of *positional encoding* and the final element has the largest *positional invariance*. We can formalize $\hat{z}$ as follows:

$$\hat{z} = \texttt{argsort}_{j\in C}\Big[\frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}|\Delta z_i|\Big]\,. \tag{2}$$

One might argue that a horizontally flipped input will introduce differences to the network even without considering the positional information because convolutional kernels can be asymmetric [1]. To further justify our ranking strategy we also rank overall position encoding channels under two settings. Kernel-Flip*: We pass an image and its flipped version (Eq. 2) to a pretrained model that has flipped kernels, Kernel-Flip**: We pass the original image to two pretrained models, with and without flipped kernels, and use the activation differences to rank the neurons using Eq. 2.

Region-Specific Channels. Contrasting Eq. 2 which treats any change in a channel's activation subject to horizontally flipping the image as an indication it encodes positional information, we now aim to identify and rank the channels based on encoding position for a *specific* image region. Thus, we are interested in channels which are *highly activated when an object is in a particular location, and have a low activation otherwise*. To this end, we create two subsets of data out of the Cityperson [6] validation set: images where the pedestrians are only located on the left half of the image (val-left), and pedestrians that are only located in the right half (val-right (see Fig. 3). If we want to identify the channels which encode objects on the left half of the image, we pass an image from *val-left*, I^a, and its corresponding flipped image, I^b, through the network and take the *signed difference* of the two latent representations after the GAP layer. After ranking the channels based on the average activation difference over the val-left dataset, we can now identify the channels which particularly activate for pedestrians on the left side. More specifically, we can calculate an ordered list of the left region-specific neurons, $\hat{z}^l \in \mathbb{R}^{1\times1\times C}$ with the following equation:

$$\hat{z}^l = \texttt{argsort}_{j\in C}\Big[\frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\Delta z_i\Big]\,. \tag{3}$$

We can easily calculate an ordered list of the *right* region-specific neurons $\hat{z}^r$ by simply passing val-right through the network and then follow the same procedure outlined above. Note that both of these procedures use the assumption that the ordering of the channels after a GAP layer admits positional information, both overall and region specific.

4.2.2 Targeting Overall Position-Encoding Neurons

Semantic Segmentation. We first validate how the top N overall position-encoding channels affect the performance of a state-of-the-art semantic segmentation network, DeepLabv3-ResNet-50 [3], trained on the Cityscapes [6] dataset. The goal of this evaluation is to determine how much DeepLabv3-ResNet-50 relies on these global position-encoding channels for semantic segmentation by measuring the difference in validation performance, measured by the mean intersection over union (mIoU), after *removing* the top N position-specific channels. We remove these N neurons from the latent representation by simply setting the feature activations at these channel dimensions to *zero*. In addition to the standard mIoU, we aim to evaluate whether these neurons affect all locations equally. To this end, we also assess the performance on three input regions: the *left*, *center* or *right* region of the image. To perform this assessment, this we simply compute the mean intersection over union (mIoU) only on the *left*, *center*, or *right* regions by setting the other pixels to the *unlabelled value* (see Fig. 3 for an example). Note that each region (i.e., left, right, and center) has a spatial resolution of 1024×1024 pixels as the width of the Cityscapes images are 2048 pixels (note that the center overlaps with the left and right sides equally). In these experiments, we perform validation on the *val* split of the Cityscapes dataset.

Results. Figure 4 presents the semantic segmentation results of *DeepLabv3-ResNet-50* on Cityscapes in terms of mIoU when the top N overall position-specific channels are set to *zero*. Note that for this experiment, we do not fine-tune the pretrained segmentation network. Interestingly, we observe that gradually removing the position-specific neurons from Cityscapes pretrained DeepLabv3-ResNet-50 (with a baseline performance of 74.0% mIoU) model hurts the overall mIoU (Fig. 4 (a)) significantly more than removing randomly selected neurons (note that 'Normal' refers to neurons selected using Eq. 2 without flipping the kernels). For the position-specific neurons, removing the top 100 channels results in a performance of 71.9%, while 100 random neurons drops the performance to only 73.4%. When removing

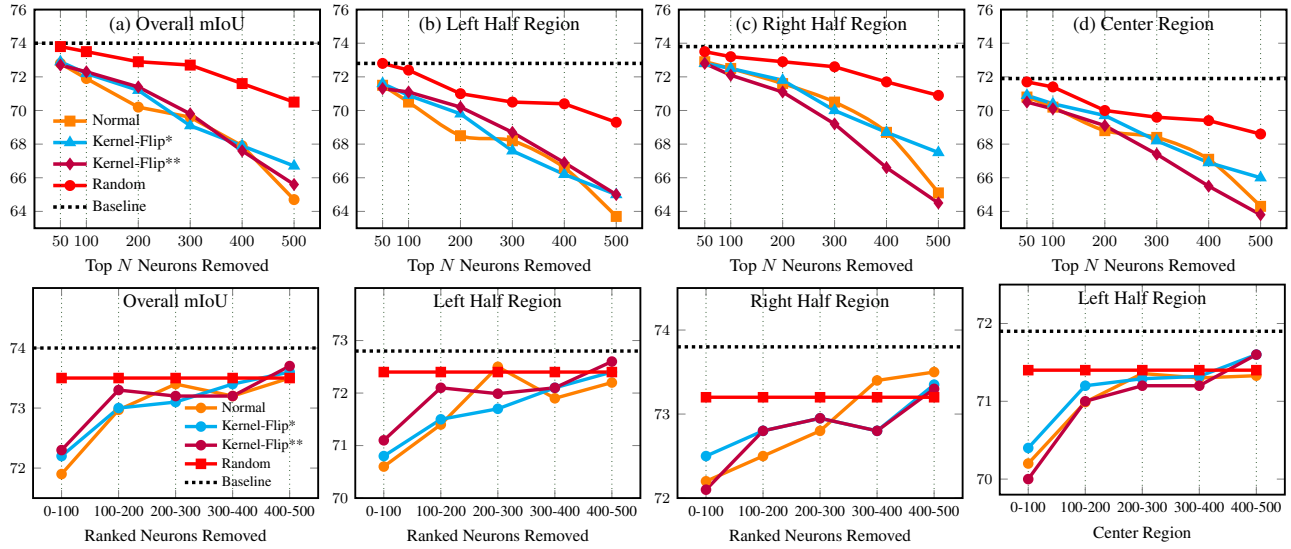

Figure 4. Semantic segmentation results for DeepLabv3-ResNet-50 trained on Cityscapes. **Top Row:** The top N overall *position*-specific channels are removed from the latent representation during inference. Removing the top N position-specific channels from the *DeepLabv3-ResNet-50* impairs the network's recognition ability more severely compared to the removal of the same number of *random* channels. **Bottom Row:** Same settings as the top row, but only channels in the ranges specified (i.e., every 100 channels) are removed.

even more neurons, the difference is more significant. For example, removing 500 position-specific channels results in a performance of 64.7% mIoU, a 9.3% drop from the baseline of 74.0%, while 500 random channels only drops the performance by 3.5% to 70.5% mIoU. When removing neurons using the kernel-flipping rankings a similar pattern in seen as removing these neurons degrades the performance more severely than removing random ones.

These results clearly demonstrate the network's reliance on the channel-wise positional encoding in the latent representations to make accurate semantic segmentation predictions. Note that the performance drop by removing 100 position encoding neurons (a drop of 2.1%) is quite significant as the size of the latent dimension, z, is 2048 for the *DeepLabv3-ResNet-50* network. This finding is *consistent for all locations* in the image. Figure 4 (b, c, d) shows the mIoU for the *left*, *right*, and *center* regions (as seen in Fig. 3). For each region, the performance drops more when targeting the position neurons obtained through the channel-wise ranking strategy than compared with targeting random neurons. Figure 4 (bottom row) plots the results for each portion of the image when we remove only 100 neurons in the ranking. These results also show that, in general, removing higher ranked neurons lowers the mIoU more than the ones at lower rankings. Collectively, these results provide strong evidence that the ranked channels using Eq. 2 encode overall image position. Note that the center region has a lower baseline accuracy, and therefore a lower drop in performance, due to pixels in the center region of autonomous driving datasets being associated with objects which are a greater distance from the camera and therefore harder to accurately classify.

Figure 5 provides qualitative results on the Cityscapes validation images while removing N specific neurons. It is clear that the segmentation quality gradually degrades with the increase of N. Note that the failure of segmenting the *smaller* and *thinner* objects in the left or right *periphery* are particularly pronounced in our neuron specific targeting. We also validate how the top N overall position-encoding channels affect the performance of a pedestrian detection network and report the results in the supplementary.

4.2.3 Targeting Region-Specific Channels

Semantic Segmentation. We now provide evidence that it is possible to harm the performance in a specific input region, for a fully convolutional neural network. We choose again the DeepLabv3-ResNet-50 [3] semantic segmentation network, trained on the Cityscapes [6] dataset. We use the *signed* ranking from Eq. 3 along with the *val-left* and *val-right* subsets to obtain the channel ranking in terms of region specific positional encoding. As motivation, we take the role of an adversary intent on causing a high-speed collision. As the Cityscapes [6] dataset was collected in Germany, cars drive on the right side of the road. Therefore *oncoming traffic appears on the left side of the images.* Our main motivation is to *expose the possibility* of a more fine-grained region-specific attack for which we use a driving dataset as a test-case. There may be cases where adversaries want to impair performance in regions of interest, while minimizing the chance of being exposed. We target the semantic segmentation performance in the *left-half of the image* using our channel targeting technique described in Sec. 4.2.1.

Figure 6 shows the validation results in terms of mIoU

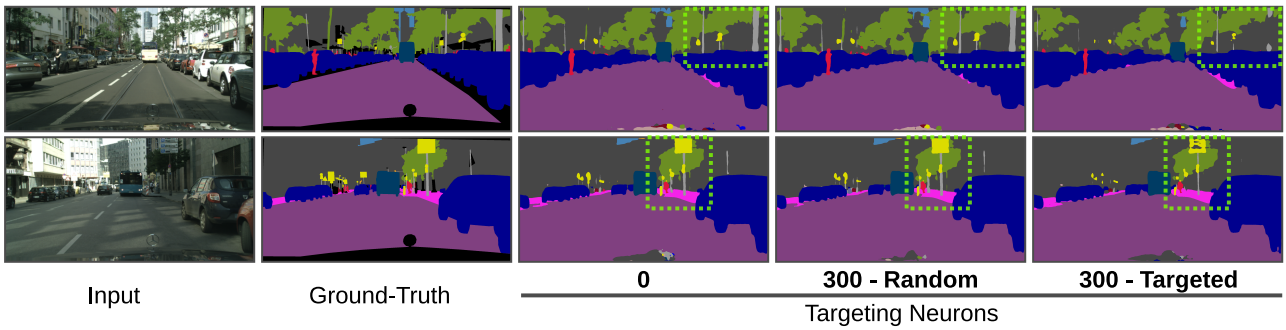

Figure 5. Qualitative comparisons between the *position-specific* and *random* neuron removal on the Cityscapes val set. Note the performance drop on objects near the periphery (highlighted in dotted box) are particularly pronounced for our position specific neuron targeting.

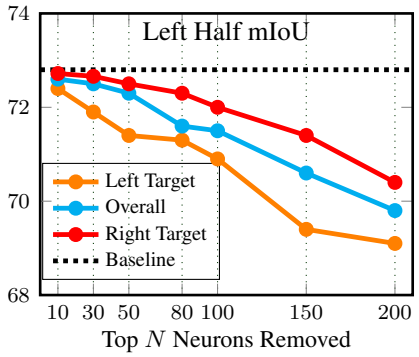

Figure 6. Semantic segmentation results (mIoU) evaluated on only the *left half of the image* for DeepLabv3's trained on Cityscapes when the top N *region*-specific channels are removed from the latent representation during inference. Removing the top N left-specific channels from the *DeepLabv3-ResNet-50* impairs the network's recognition ability more severely compared to the corresponding removal of overall position or right-specific channels.

where only the left half of the image is considered during evaluation (see Fig. 3 for example ground-truth segmentation maps) and the top N channels are set to *zero*. Our hypothesis predicts that the left target curve should be lower than the right target curve as the left encoding channels should harm the mIoU on the left half more than the right encoding channels. As expected, the channels which we identified as encoding 'Left' are predictive of a larger performance drop in the left-half mIoU calculation compared with the other channels. Interestingly, the 'Overall' position channels have the next largest performance drop, followed by the 'Right' channels. This provides further evidence that these channels do capture objects in specific regions in the image, as one would expect less overlap between the 'Left' and 'Right' channels compared with the 'Left' and 'Overall' position channels. Also note that (although small) a discrepancy in mIoU between the Left and Right channels is even observed after 10 neurons are zeroed out with performance drops of 0.6% and 0.08%, respectively. This discrepancy grows as more of the channels that capture objects on the left are removed. Results for an equivalent attack on the right half of the image are included in the supplementary and are consistent with findings presented here on the left half.

5. Conclusion

We have shown for the first time how CNNs with global average pooling layers, which collapse the spatial dimensions, admit absolute positional information. Moreover, we showed that position information is encoded based on the *ordering* of the channels while semantic information is largely not. We then applied these findings to various real-world applications. We proposed an objective function to improve the translation invariance of CNNs trained for object recognition. We introduced a simple and intuitive technique to identify and rank the position-encoding neurons in a CNN's latent representation. We showed this technique could identify the channels which largely encode either (i) global position or (ii) region-specific positions in the input. Regarding the global position channels, we show that suppressing their responses in networks trained for semantic segmentation and object recognition results in a greater drop in performance compared to other baselines, suggesting that these CNNs rely significantly on the channel-wise positional encoding. Finally, we demonstrated the possibility of a fine-grained adversarial attack which aims to harm the performance of a network in a *specific location*. All of these experiments were performed through the manipulation of the latent representations *after a global average pooling layer* demonstrating the rich positional information contained in the ordering of the channels in a number of neural network architectures. We believe these findings and associated applications can help guide the future design of neural networks to contain the right inductive biases for the task at hand.

Acknowledgements. We gratefully acknowledge financial support from the Canadian NSERC Discovery Grants and Vector Institute Post-graduate Affiliation award. K.G.D. contributed to this work in his personal capacity as an Associate Professor at York University. We thank the NVIDIA Corporation for providing GPUs through their academic program.

References

[1] Bilal Alsallakh, Narine Kokhlikyan, Vivek Miglani, Jun Yuan, and Orion Reblitz-Richardson. Mind the pad–CNNs can develop blind spots. In *ICLR*, 2020. 2, 6

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *TPAMI*, 2018. 1, 5

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017. 6, 7

[4] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 2

[5] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *ICML*, 2016. 1

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 6, 7

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 4, 5

[8] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *NeurIPS*, 2015. 2

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 3, 4

[11] Md Amirul Islam, Sen Jia, and Neil DB Bruce. How much position information do convolutional neural networks encode? In *ICLR*, 2020. 1, 2

[12] Md Amirul Islam, Matthew Kowal, Sen Jia, Konstantinos G Derpanis, and Neil DB Bruce. Position, padding and predictions: A deeper look at position information in CNNs. *arXiv preprint arXiv:2101.12322*, 2021. 1, 2, 3

[13] Md Amirul Islam, Mrigank Rochan, Neil DB Bruce, and Yang Wang. Gated feedback refinement network for dense image labeling. In *CVPR*, 2017. 5

[14] Osman Semih Kayhan and Jan C van Gemert. On translation invariance in CNNs: Convolutional layers can exploit absolute spatial location. In *CVPR*, 2020. 1, 2, 3

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 4

[16] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset. *online: http://www. cs. toronto. edu/kriz/-cifar. html*, 2014. 3, 4, 5

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[18] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 1, 3

[19] Guilin Liu, Kevin J Shih, Ting-Chun Wang, Fitsum A Reda, Karan Sapra, Zhiding Yu, Andrew Tao, and Bryan Catanzaro. Partial convolution based padding. *arXiv:1811.11718*, 2018. 2

[20] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *NeurIPS*, 2018. 2

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 4

[22] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *NeurIPS*, 2017. 2

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 3

[25] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. SOLO: Segmenting objects by locations. In *ECCV*, 2020. 2

[26] Rui Xu, Xintao Wang, Kai Chen, Bolei Zhou, and Chen Change Loy. Positional encoding as spatial inductive bias in gans. In *CVPR*, 2021. 2

[27] Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, 2019. 2, 4, 5

[28] Xueyan Zou, Fanyi Xiao, Zhiding Yu, and Yong Jae Lee. Delving deeper into anti-aliasing in ConvNets. In *BMVC*, 2020. 2