Reproducing CNN^2: Viewpoint Generalization via a Binocular Vision

Manoosh Samiei
McGill University

Rezvan Sherkati
McGill University

Aarash Feizi
McGill University

Abstract

In this project, we replicate the results of the paper "CNN^2: Viewpoint Generalization via a Binocular Vision" for two datasets SmallNORB and ModelNet2D. We realize that, CNN^2 gives results close to the original CNN model, referred to as "vanilla CNN" by the authors. We also perform an ablation study, by changing the network's architecture, multiple setups, and hyperparameters in the proposed method. We observe that some modifications such as changing the CM pooling to max pooling, lower the accuracy as they have also claimed. However, there are some modifications that appear to be improving the accuracy, in contrast to what the authors claim in the paper.

1 Introduction

In the original paper, "CNN^2: Viewpoint Generalization via a Binocular Vision", the authors propose the model CNN^2, which is a convolutional neural network that takes two binocular images of an object as input and is able to recognize that object at unseen viewpoints at test time. The authors claim that their model has improved performance in 3D viewpoint generalizability using 2D images. A picture of their model can be seen in Figure 1.

CNN^2 which models some priors from binocular vision, applies contrastive channel augmentation to the respective images so they are scanned by filters in two parallel, complementary feedforward pathways. At each layer, the feature maps are combined and then split by following the dual parallax augmentation procedure, which is done as follow: First, given a pair of left and right images: H_L and H_R. $H_R - H_L$ is added to H_L, and $H_L - H_R$ is added to H_R as a series of new channels. These two tensors are fed into each of the two parallel pathways. Then after each layer, the output feature maps of the two pathways (which resembles the left and right eye visual pathways) are concatenated in their channel dimension and are fed into the next layers through each pathway.

Besides the augmentation, CNN^2 employs concentric multi-scale (CM) pooling layers that are applied before the convolutional layers to learn the in-focus and out-of-focus features. The proposed CM pooling does not change the width and height of the input image or feature map. These pooling layers perform average or max pooling across different kernel sizes and then the results, each called one scale, are stacked up across channel dimension of the input tensor. The authors claim that this pooling method introduces translation invariance to the model, since large scales detect blurry features in the background and small scale detects clear features in the foreground. Figure 2 indicates the CM pooling operation.

As authors claim, this model has improved accuracy in 3D viewpoint generalizability comparing to its previous models, such as vanilla CNN [6, 7], capsule networks [3, 4, 11] and voxel discretization [10, 12, 14].

The authors goal is to capture generic patterns, such as depth, stereoscopic edges, and foreground and background that are obtained by human visual system using binocular images. However, the idea

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

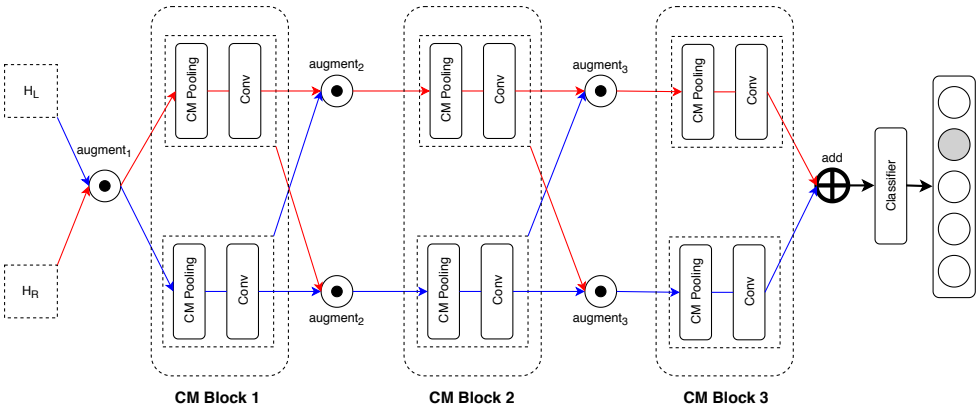


Figure 1: The architecture of CNN^2 model

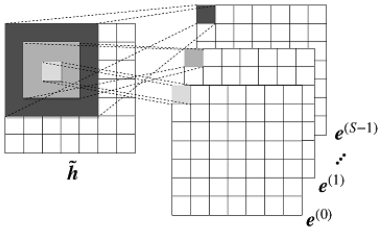


Figure 2: Concentric multi-scale pooling. Image is taken from the original paper [2]

of binocularity is loosely implemented in their paper and its definition defers between their three datasets. We will elaborate on this idea further in Section 3.

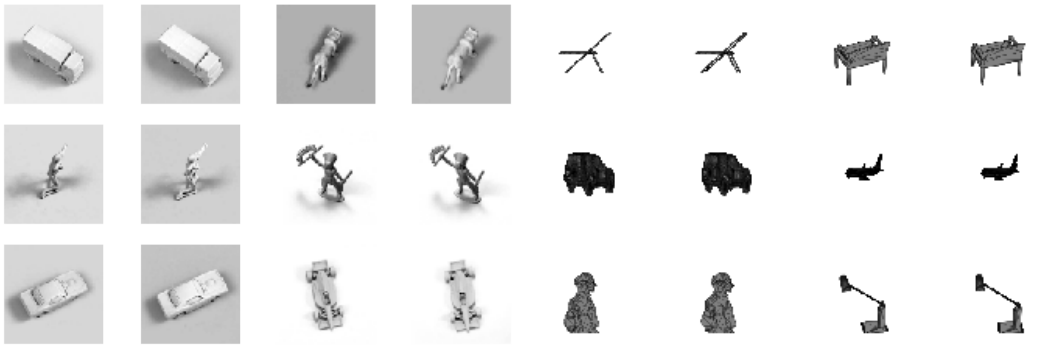
In this project, we perform an ablation study on the mehtod proposed by this paper. we implement different modifications on their model and experiment them on the ModelNet2D and SmallNORB datasets.

In Section 2, we discuss some of the related works regarding this method. Then, in Section 3 we describe the datasets used in the project. Finally, in Section 4 we present our results and in Section 5 we discuss them.

2 Related Work

Viewpoint generalization has been an important problem over time. Related to the 3D viewpoint generalizability from 2D images, Hinton et al. [3] introduce the idea of capsule and use a non-convolutional network to learn group-invariant features. After this, Sabour et al. [11] introduce the concept of capsule networks which encodes an entity in a capsule that outputs a vector rather than a scalar. The authors of Hinton et al. [4] propose the notion of matrix capsule that can learn to represent the relationship between the entity and the viewer (the pose) and it uses an expectation-maximization-based routing algorithm to train the capsule network. Capsule networks have higher performance compared to CNNs, but they also take a longer time to train because of the greater number of weights and the iterative routing algorithm.

Also there has been many studies on viewpoint generalization by voxel discretization or by reconstructing manifold (and non-manifold) surfaces in 3D space from point clouds using voxels as an intermediate representation [10, 9, 12]. A downside to this approach is that the models that have voxel outputs require either the voxel-level supervision or omnidirectional images as input, which are both expensive to get.



(a) Binocular images in SmallNORB dataset (b) Binocular images in ModelNet2D dataset

Figure 3: Several samples of the two datasets, displayed in pairs of binocular images.

3 Dataset and Setup

In the original paper the authors experiment their model on 3 different datasets, SmallNORB [8], ModelNet2D [13] and RGB-D Object [5]. However, the authors did not include the RGB-D Objects' dataset in their provided datasets. Therefore, we were only able to use the first two datasets in our ablation study.

The ModelNet40 dataset contains about 12,311 CAD models in 40 classes. The authors picked 5 classes (chairs, human figures, cars, airplanes, and lamps) and each of these classes has 15 model instances. They have rendered each 3D model into 648 grayscale images with 72 azimuth angles (with ticks at each 5 degrees), and 9 elevations (30 to 70 degrees with 5-degree ticks) under the fixed lighting conditions. Following these steps, they obtain 48,600 images, which is called ModelNet2D. These images are converted to binocular pairs. Then 10,800 image pairs are used for training set, 32,400 pairs for testing set, and 5,400 image pairs for validation set.

In the main paper, the authors claim that the ModelNet2D dataset does not include binocular images, and they use images with successive azimuth angles (5 degree difference in view point) to simulate binocular images. However, in the supplementary materials (Datasets and Preprocessing Section) they state that the dataset is binocular, which has a conflict with their previous claim [1]. Furthermore, simulating binocular images is not implemented correctly in their code and random images with different elevations and azimuths are chosen as binocular pairs. Even if successive azimuths were chosen as binocular pairs, there was an extra rotation between the two binocular images' viewpoints in addition to a translation, which is different from the binocularity of the other dataset which is captured by binocular cameras. (there is only a translation between the lenses of the two cameras.)

The other dataset, SmallNORB, consists of 48,600 grayscale binocular images of size 96×96 pixels in 5 classes (four-legged animals, human figures, airplanes, trucks, and cars). Each class has 10 object instances, and each instance has 972 binocular images taken by two cameras under 6 lighting conditions, 9 elevations (30 to 70 degrees, with ticks at each 5 degrees), and 18 azimuths (0 to 340, with ticks at every 20 degrees). [8]

Among these 48,600 pair of binocular images, 16,200 pairs are used for training, and 32,400 pairs are used for testing the model. The authors did not include validation set in their published code. However, we extracted 5400 images from the training data as validation set, which was further removed as it decreased the accuracy on test set.

In SmallNORB datasets, in contrast to some classes in ModelNet40, classes are selected in a way that objects in different classes can be distinguished by humans from each other in every view angle. In ModelNet2D authors select 5 classes, "chair," "human," "airplane," "car," and "lamp" from ModelNet40, which can be easily distinguished by humans at every viewpoint angle [1]. In Figure 3 you can see several samples of binocular images in these datasets.

Table 1: Test accuracy in different modifications in the two datasets.

	Original model	Max Pooling	Reverse Pooling
ModelNet2D	99.57%	99.38%	**99.74%**
SmallNORB	**89.50%**	86.60%	88.97%

Table 2: Test accuracy in different modifications in the two datasets

	Orig. model	Vanilla CNN	Orig. w/o 3 extra conv	Vanilla w/o 3 extra conv
ModelNet2D	99.57%	**99.66%**	99.46%	99.45%
SmallNORB	89.50%	88.25%	**91.09%**	88.71%

4 Results

In our ablation study, we tested the authors' implementations of the original model and also implemented the modifications of the original model for the two datasets, ModelNet2D and SmallNORB, and measured the model's performance in each modification. To speed up the training process, the authors downsample SmallNORB and ModelNet2D images to 48×48 pixels and normalize the values of the pixels. They train each model using the Adam optimizer with mini-batches of size 32 and an adaptive learning rate. They also split the data such that the size of test set is approximately twice the size of training set. In order to compare the results of different modifications, a certain number of seeds were assigned to the different random number generators and data-shufflers.

In each of the following subsections, the details and results of each modification is demonstrated.

4.1 Comparing the Model with Vanilla CNN

We compared CNN^2 with a vanilla CNN model(with the same number of filters and kernel size as CNN^2 in each layer + max polling layers with the same strides as CM pooling layers), on both datasets. For ModelNet2D dataset, CNN^2 performance is approximately the same as Vanilla CNN. While for SmallNORB, CNN^2 performs slightly better than Vanilla CNN. Also, we observed that authors have placed three extra convolutional layers at the end of the parallel pathways in their implementation of the network. We considered these three extra layers as the classifier of the model as they are not present in the authors' network blocks in Figure 1. We investigated the effect of these three convolution layers, by removing them from the original implementation; however, we did not observe significant change in the accuracy of the network. Even for the SmallNORB dataset, the accuracy was improved after removing these three layers. We also compared the performance of the original model with out three extra convolution layers with a vanilla CNN with out three extra convolution layers. The results of all these experiments are presented in table 2.

4.2 Changing Hyper-parameters

We further investigated the effect of some hyper parameters of the model. In particular, we changed the number of pooling scales and batch size of data. The original model has three pooling scales:[1,3,5], which are the size of the kernels of max pooling. We tested scales=[1,3], [3,5], and [1,3,5,7]. We realized that scales [1,3] perform the best on both datasets. The batch size in the original model is 32. We tested the model with batch size 16 and 8 on ModelNet2D dataset, and realized that batch size 32 works the best. (batch size of 8 had a comparable performance as well)

Table 3: Test accuracy in different modifications in the two datasets.

	Original Model	S=[1,3]	S=[3,5]	S=[1,3,5,7]	Batch=16	Batch=8
ModelNet2D	**99.57%**	**99.60%**	99.01%	99.44%	99.42%	**99.59%**
SmallNORB	89.50%	**91.47%**	-	89.36%	-	-

Table 4: Test accuracy with various numbers of CM blocks on 2 different datasets. The * indicates the original model.

	2 CM Blocks	3 CM Blocks *	4 CM Blocks
ModelNet2D	99.39%	**99.57**%	98.31%
SmallNorb	87.01%	**89.50**%	83.51%

Table 5: Accuracy for different levels of augmentations. Each column is a variation of the model where the numbers represent the active augmentation cells. E.g. $\{1, 2\}$ means $augmentation_1$ and $augmentation_2$ are active, but $augmentation_3$ is not active according to Figure 1.

Augmentations	None	$\{1\}$	$\{2\}$	$\{3\}$	$\{1, 2\}$	$\{1, 3\}$	$\{2, 3\}$	$\{1, 2, 3\}$*
ModelNet2D	98.31%	98.95%	98.31%	98.67%	99.28%	99.50%	99.39%	**99.57**%
SmallNORB	77.77%	85.65%	79.44%	78.55%	86.67%	87.66%	80.43%	**89.50**%

4.3 Placing Pooling layers after convolutional layers

Here for each of the two datasets we reversed the order of the CM pooling and the convolutional layer in each of the blocks to see its effect on the accuracy of the prediction. The authors claim that placing the CM Pooling before the convolution layer aids the filter in the next layer to easily detect stereoscopic patterns by contrasting blurry features with clear features and they also claim that this model produces a higher accuracy than the model with CM pooling after the convolutional layer.

Trying to evaluate this claim, as we can see in Table 1, for ModelNet2D dataset we observed the opposite effect; meaning the accuracy of original method is lower than the modified version.

4.4 Substituting CM Pooling

In this part, instead of using the CM pooling suggested by the authors of the paper, we use the max pooling layer in the network. As we see in Table 1, the accuracy of original model is higher in this case which verifies the claim of authors. This proved the advantage of using CM pooling over max pooling.

4.5 Altering Number of CM Blocks

In this section, we explore the effect of the number of the CM blocks (CM pooling + convolution layer) on the accuracy on the test set. The original model proposed by the authors has three CM blocks. We look into the results for 2 and 4 CM blocks and compare them to the original model. The results can be seen in Table 4. It can be concluded that the model with three CM blocks performs the best for both datasets.

4.6 Altering Augmentation in CM Blocks

To investigate the effect of augmenting the left and right inputs in each block, we remove the concatenations in each block of the original model (with three CM blocks) and compare the test accuracies of each modification. More specifically, according to Figure 1, we train the model with/without different augmentation steps. The results can be seen in Table 5. Removing an augmentation step means not using the output of the parallel network for the input to the next layer. If the inputs are not concatenated, since the number of channels that are fed to the next block are halved, the number of filters used for the convolution in the next block are also halved to follow the model's structure. As it can be seen, the augmentation in the first layer is the most important one and has the most effect on the accuracy of the models.

4.7 Altering Size of CM Block Filters

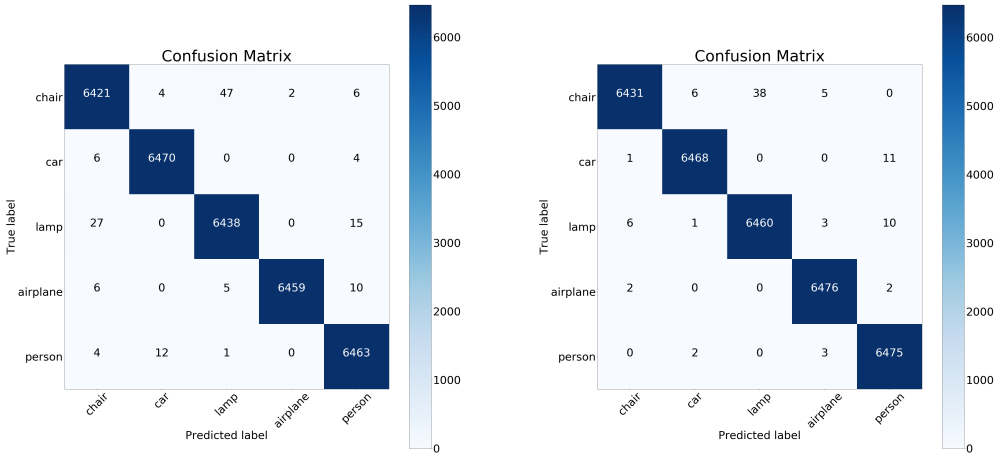
Another factor for each convolution layer is the number of filters used in the CM blocks. Different number of filters well used for the original architecture and also the two-layered version. The original

Table 6: Accuracy for different number of filters in CM blocks. The two-layered network was used for the columns with two filter sizes. The * represents the original architecture.

Number of Filters	{6, 12, 32}*	{6, 18, 40}	{6, 24, 40}	{6, 12}	{6, 18}	{6, 24}
ModelNet2D	99.57%	**99.72%**	99.62%	99.39%	99.58%	99.59%
SmallNORB	**89.50%**	89.41%	88.69%	87.01%	87.37%	86.81%

number of filters are 6, 12, and 32 for the first, second and third layer, respectively. The different variations of the networks are shown in Table 6.

Fortunately both datasets are balanced and contain the same number of samples for each class. Hence, training on these datasets will not directly encourage a bias towards a certain class. Furthermore, since each class has the same number of samples, the accuracy on the test set is a reasonable metric for measuring the performance of the trained network. However, for further investigation, we also plot the confusion matrix for a few of the models to identify the misclassified classes. As it can be seen in Figure 4, the network is more confused with objects of lamp and chair classes.



(a) The original network.

(b) The network with 6, 18, and 40 filters.

Figure 4: Confusion matrices of two architectures.

5 Discussion and Conclusion

We investigated different modifications on the authors proposed model, and figured out that even though their model has better performance in SmallNORB dataset, for ModelNet2D dataset the original CNN model has better accuracy. So based on this, we cannot compare which model has better performance in general.

Also, the authors claim that their model is a simplified model of how human's visual system works. However, the characteristics of binocular images captured by human eyes are much more complex and not fully discovered, yet. For instance when two images are captured using two horizontally placed binocular cameras, the position of cameras' lenses are fixed. However, when humans look at an object, both of their eye balls move toward that object. These extra movements affect the disparity between the two images. Hence, the stereo vision of human eyes is much more complex, compared to binocular cameras. For future investigations, we suggest that two eyeball cameras (which can imitate the eye's movement) be placed at a fixed distance from each other, to capture binocular images.

6 Statement of Contributions

All members of the project contributed equally to all the parts of the project.

References

[1] Wei-Da Chen and Shan-Hung Wu. Cnn2: Supplementary materials.

[2] Wei-Da Chen and Shan-Hung Wu. Cnn2: Viewpoint generalization via a binocular vision.

[3] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011.

[4] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. 2018.

[5] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE international conference on robotics and automation*, pages 1817–1824. IEEE, 2011.

[6] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:II–104 Vol.2, 2004.

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[10] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.

[11] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

[12] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[13] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[14] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 1696–1704, 2016.