MiniProject4: CNN2: Viewpoint Generalization via a Binocular Vision

Arneet Singh Kalra
McGill University
110 Chateau Kirkland
arneet.kalra@mail.mcgill.ca

Behnaz Ghefati Feizabadi
Concordia University
2175 Maisonneuve Boulevard West
behnaz.gheflatifeizabadi@mail.mcgill.ca

Hussein Lakkis
McGill University
367 Aylmer Street
hussein.lakkis@mail.mcgill.ca

Reproducibility Summary

Scope of Reproducibility

The original paper attempts to improve 3D viewpoint generalizability of CNNs based on the idea of human binocular vision. The paper proposes CNN^2, a CNN which uses novel augmentation, CM Pooling, and convolutional layers to learn a sense of 3-dimensions to predict images from an unseen viewpoint. They claimed the proposed solution is more accurate and takes less time to train compared to other techniques such as Vanilla (unaltered) CNN.

Methodology

The author's original code and dataset were attached to their Github repository (https://github.com/wdchenxyz/CNN2). The code was refined and further experiments were added and run using Google Colab, with a GPU hardware accelerator. The experiments altogether took 50 and 30 minutes for ModelNet2D and SmallNORB datasets respectively.

Results

Our results also indicated that the proposed CNN^2's superiority over other methods was not as clear as the paper. Firstly, our results for the CNN^2 models were generally 3 to 4% higher than in the paper. However, for the ModelNet2D dataset, we observe a less than 1% testing accuracy difference between CNN^2 and Vanilla CNN, whereas the paper sees a substantial difference. For the SmallNORB dataset, we do observe a substantial difference of 10% between the 2 models similar to the paper. Furthermore, for the ModelNet2D set, we do not reproduce the same results regarding the additional experiments with CM pooling and feedforward pathways, but we do for the SmallNORB set.

What was easy

Reproducing the paper results on the ModelNet2D and SmallNORB dataset was easy to do. The provided Github code worked and was relatively straightforward to implement for data pre-processing, construction the model, and running the experiments, and required no debugging. Adding additional experiments was also simple to do using tensorflow.

What was difficult

Since the RGB-D Object Dataset was unavailable, we were not able to validate the accuracy of the provided results for the set. Understanding the code took time as it was not well documented or commented. The authors had pretrained models that were missing from the data available online. Also, we did not test the Bl-Net, Monodepth, PTN, CapsuleNet, and CNN^2 + BL models as those were based on other papers and were not included in the given code. Furthermore, the validation sets for the smallNORB dataset were not provided. We attempted to create one splitting the test dataset but that further decreased performance and hence we only used the test dataset provided as a validation dataset.

Communication with original authors

We contacted the authors regarding the "cp.ckpt" file which was used in their code, but was not provided in their Github repository. According to a reply we received from Wei-Da Che, we trained the model by ourself without using that file.

1 Introduction

Although Convolutional Neural Networks (CNNs) are great models for many tasks, they are still behind human performance when it comes to the 3D viewpoint generalizability. A proposed solution to the improving these models is CNN^2, which aims to use binocular vision. The idea comes from human anatomy: humans have a left and right eye which make images separately and are then combined together. Furthermore, studies show that human's visual systems can detect stereoscopic edges, foreground and background, and contours of objects extrapolated from already seen angles [4]. The CNN^2 model aims to capture these patterns. It does so by employing dual feedforward pathways, recursive parallax augmentation, and concentric multi-scale pooling (CM Pooling). The proposed model supposedly works better than conventional CNNs as it is more accurate and converges faster during training[2].

2 Scope of reproducibility

The paper makes the following claims which we are reproducing to check the validity of the following claims:

- CNN^2 has improved 3D generalizability compared to Vanilla CNN.
- The learning speed of CNN^2 is faster than Vanilla CNN for 3D generalizability.
- Using Concentric Multi-Scale pooling in CNN^2 is more accurate than the conventional Maxpooling.
- Having two feedforward pathways result in a more accurate model compared to one feedforward pathway.
- The parallax augmentation can improve the model generalizability at challenging view angles.
- Placing the pooling layers after the convolution layers in CNN^2 reduces accuracy compared to placing it before.
- Fusing the two feedforward pathways at each layer results in a more accurate model when compared to performing fusion in the only the first and last layer.

3 Methodology

In order to reproduce the results, we used the original source code provided by the the authors of the paper, which was found on their GitHub repository. The code is split into different Jupiter Notebook files for each dataset. There are also accompanying python files with helper functions. The associated documentation is the README file found in their repository containing information of how to run their models. Unfortunately, the code is not very well explained and lacks comments. Furthermore, the data they used was provided in a ZIP file in the repository. However, we were unable to obtain access to the RGB-D dataset. When reproducing the experiments, we ran the dedicated models on Google Colab instead of locally on our machines. This helped standardize the testing among our team members. We chose to check CNN^2 and Vanilla CNN models only, since the other models were based on other papers and implementing them needed more time. We also ran the experiments found in the paper, and added few more as described later in the report.

3.1 Model descriptions

3.1.1 CNN^2

We begin with tests of the proposed CNN^2 model. Figure 1 showcases the model architecture of the solution provided by the paper. It includes two feedforward pathways which provide the dual parallax augmentation at varying abstraction levels. There is also concentric multi-scale pooling. This is used to mimic the way humans experience the reflection of light, blurring our objects as they are out of focus. This pooling is placed before a convolution in CNN^2. Essentially, it applies a filter to easily detect stereoscopic patterns as it contrasts in-focus versus out-of-focus features [2]. There are a few drawbacks with the model. A feature map at a hidden layer will have the same large dimensions as the input image which could slow down computations. Furthermore, there are more number of filter weights due to a larger number of channels.

3.2 Datasets

The authors of the paper provide a ZIP file with the data that they used. However, the RGB-D dataset is unavailable to us. Due to this, we only tested the ModelNet2D and SmallNORB datasets.

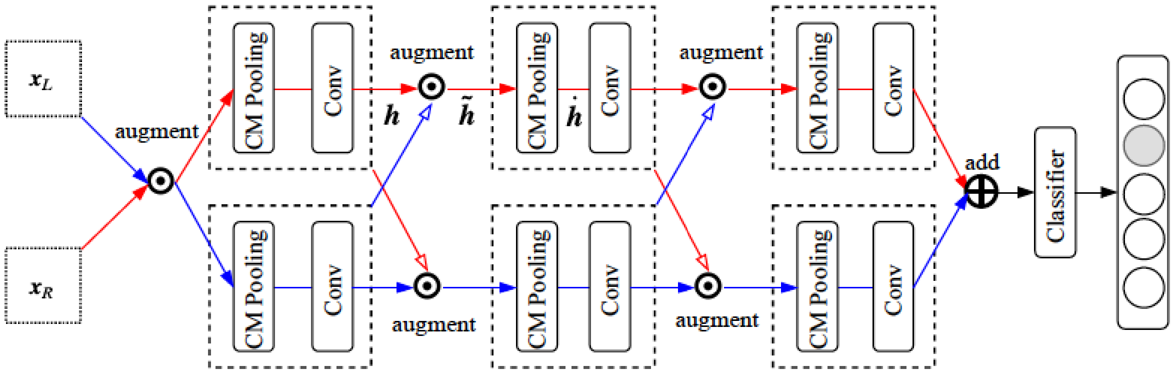

Figure 1: CNN^2 Model Architecture

3.2.1 ModelNet2D

The ModelNet2D dataset is prepared by the authors based off Princeton University's ModelNet40 set. The original has about 12 300 CAD models in 40 classes. Each class has 15 model instances. This dataset does not natively provide binocular images so preprocesssing was required. Essentially, pairs of images with successive azimuths degrees were used to simulate binocular images (Figure 2). Each model is rendered into 648 grayscale binocular images with 72 azimuth angles at 5 degree intervals, and 9 elevations under fixed lighting conditions. The distance between the two lens in the binocular camera is 7.5cm, and the images themselves are 96 x 96 pixels. The resulting dataset is the ModelNet2D which contains 48 600 binocular images, out of which 10800 are for training, 5400 for validation, and 32400 for testing. The selected classes found in this dataset are 'chair', 'car', 'lamp', 'airplane' and 'person'.

3.2.2 SmallNORB

The SmallNORB dataset consists of 48 600 grayscale binocular images, each of size 96 x 96. There are 5 classes for this set: 'four-legged animals', 'human figures', 'airplanes', 'trucks', and 'cars'. Each class has 10 object instances, and each instance has 972 binocular images taken 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) (Figure 3). The SmallNORB dataset did not provide a validation set, so instead we chose to use the test set for validation. There are 16200 training images and 32400 testing images.

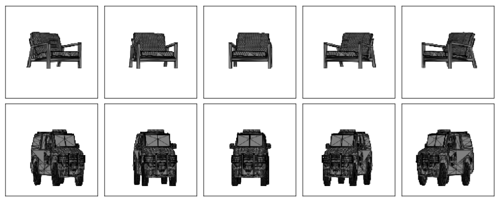

Figure 2: ModelNet2D Sample Images

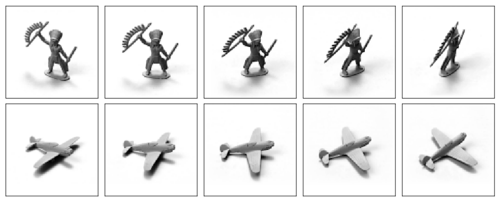

Figure 3: SmallNORB Sample Images

3.3 Hyperparameters

The authors of the paper downsampled the ModelNet2D and SmallNORB images to 48 x 48 in order to increase training time. Additionally, they chose to use a mini batch size of 32 and an early stopping strategy for all experiments. Each model is trained using the Adam algorithm. We will be using the same strategies and running each experiment for 10 epochs, since we did not observed significant changes in the accuracy after 10 epochs.

3.3.1 CNN^2

The proposed CNN^2 consists of 3 blocks, each with a CM pooling layer and a convolution layer. 3 scales (s = 0, 1, 2) are used in a CM pooling layer. The first, second, and third convolution layers have 6, 12, and 32 filters with kernel sizes 5×5, 5×5, and 3×3 respectively. All filters use stride of 1, and each unit uses the ReLU non-linearity.

3.3.2 Vanilla CNN

The hyperparameters for Vanilla CNN are based off an original paper by LeCun et al [6]. The Vanilla CNN model has three convolutional layers with 16, 32 and 64 filters respectively. Each filter has a size 5 x 5 with stride 1 and is then followed by a 2 by 2 max pooling with stride 2. The task model has 1024 units and is connected to the 5-way softmax output layer. All units use ReLU activations. No modifications were made to the hyperparameters.

3.4 Experimental setup and code

The source code was downloaded from the GitHub repository provided by the authors of the paper. The files were added to a Google Drive folder from which we would run the Google Colab Notebooks. The dataset was also placed in this folder. Then, in Google Colab, we ran the given Jupiter Notebook files for each dataset in a GPU hardware accelerated runtime environment. We then ran experiments specifically testing the accuracy of CNN^2, Vanilla CNN, and a few of the additional experiments highlighted in the paper. Specifically, we tested pooling after convolution instead of before, ablation study, and fusion of the two feedforward pathways. Additionally, we also tested a different number of scales in CM pooling, which was not present in the paper. To test the effectiveness of the models, we used the same evaluate function provided by the authors. We returned the loss and accuracy results. Our code can be found here: https://github.com/arneetsinghkalra/Reproducibility_Challenge_ML.

3.5 Computational requirements

A GPU hardware boost was selected for the runtime within Google Colab. Unfortunately, with Google Colab there is no way to select the type of GPU. The most likely options for our runtime are Nvidia K80s, T4s, P4s and P100s, as described in the documentation of Google Colab. The experiments on ModelNet2D took 50 minutes to run, and 30 minutes for SmallNORB.

4 Results

Our results do not produce the claims to the same extent as the paper. For the SmallNORB dataset, we do see a substantial difference between CNN^2 and Vanilla CNN in favour of the former model. However, the same results are not seen for the ModelNet2D dataset, while in the paper CNN^2 is more accurate. The results of the experiments included in the paper are summarized in table 1, and our additional experiments are found in table 2.

4.1 Results reproducing original paper

4.1.1 CNN^2 versus Vanilla CNN

This experiment was run to support the claim that CNN^2 has improved 3D generalizability compared to Vanilla CNN, and that the learning speed is also faster. Our results concluded that for the SmallNORB dataset, there is a 10.90% accuracy difference in favour of CNN^2. However, for the ModelNet2D dataset, we do not see a significant difference. Our results show a negligible 0.2% testing accuracy in favour of Vanilla CNN, whereas in the paper CNN^2 is 3.4% more accurate.

4.1.2 CNN^2 with CM Pooling After Convolution

The proposed CNN^2 model performs pooling before the convolution layer since it supposedly gives better performance. However, for both the ModelNet2D and SmallNORB datasets, we see accuracy results within 1% whether we perform pooling before or after the convolution layer. Therefore, our results do not support the claim that pooling before gives better performance.

4.1.3 CM Pooling versus Maxpooling

This experiment supports the claim that in CNN^2, the CM pooling is more effective than the conventional max pooling. Our results indicate that for the ModelNet2D dataset, we see a negligible 0.2% decrease in accuracy with max pooling. For the SmallNORB dataset, however, we do see a 4.44% decrease with max pooling. Then, we tested the effect of using CM Pooling on the accuracy for Vanilla CNN. For ModelNet2D and SmallNORB, we receive accuracies of 98.5% and 88.1%. There is a 1% decrease for ModelNet2D dataset, while the accuracy for SmallNORB dataset increase substantially in comparison to the Vanilla CNN results, 79.6%.

4.1.4 One versus Two Feedforward Pathways

This experiment verifies the claim that two feedforward pathways result in a more accurate model compared to one feedforward pathway. Our results show that for the ModelNet2D dataset, there is no significant difference between the two. However, for SmallNORB, we do see 7.88% decrease in accuracy when using only one feedforward path. Since ModelNet2D does not show the same results as in the paper, we were unable to completly reproduce the results.

4.1.5 CNN^2 Without Parallax Augmentation

The experiment checks to see if the claim that parallax augmentation in CNN^2 results in a more accurate result. For the ModelNet2D, we see a 0.2% increase without augmentation, but this is likely due to chance and also negligible. On the SmallNORB dataset, we see a 7.27% decrease when not using augmentation. We were not able to successfully reproduce the complete results showed in the report.

4.1.6 Fusion of two Feedforward Pathways

This experiment checks to see if fusing the two feedforward pathways at each layer actually results in a more accurate model when compared to fusing in only the first and last layers. We do not see a substantial difference between the two for the ModelNet2D dataset. However, we do see a 6.1% decrease when not fusing at each layer in the SmallNORB dataset. Since we did not see adequate results in ModelNet2D, we were unable to completly reproduce the paper's results.

4.2 Results beyond original paper

4.2.1 Varying Number of Scales in CM Pooling

In the paper, the authors use the same 3 scales in the CM pooling layer. We ran additional experiments to test the effect of less and more scales on the accuracy. We tested using scales [1,3] and [1,3,5,7]. Our results showed that for the ModelNet2D dataset as well as SmallNORB, the original 3 scales performed best. ModelNet2D had accuracies of 98.65% and 99.00% for 2 and 4 scales respectively. SmallNORB resulted in 85.51% and 87.28% for 2 and 4 scales respectively.

4.2.2 CM Pooling with Average Pooling instead of Max Poolin

In addition to testing max pooling as a replacement for CM Pooling, we also test using average pooling instead of max pooling in CM Pooling layer. For the ModelNet2D, we received an identical testing accuracy as CM Pooling at 99.4%, showing no real difference. For the SmallNORB dataset, we also receive an almost identical testing accuracy at 90.52%.

4.2.3 Varying number of CM Pooling and Convolutional layers

Lastly, we test to see the effect of having a differing number of CM Pooling + Convolutional layers. We test both datasets with 2 and 4 layers respectively. For the ModelNet2D set, we receive accuracies within 0.3% of the original CNN^2 model, therefore no substantial difference. However, for the SmallNORB dataset, we receive accuracies of 87.45% and 86.05% for 2 and 4 layers respectively. Comparing this to the original 90.5% testing accuracy, we observe less accurate models with these varying layers. This shows 3 CM Pooling + Convolutional layers chosen in the paper is optimal.

5 Discussion

Based off our results, we do not completely reproduce the results we verified reported in the paper. While the SmallNORB dataset, does support all the claims of the paper, we do not see any substantial changes between the models for the ModelNet2D dataset. It is important to note that we were unable to check other models which the paper also looked at. The original authors compared their proposed CNN^2 model not only with Vanilla CNN, but also with BL-Net[1], Monodepth [3], PTN [7], CapsuleNet[5], and CNN^2+BL, which we were unable to test due to time constraints. Furthermore, the paper also tested the CNN^2 performance with different backbone architectures which we did not verify. Instead, we ran a few extra expriments checking the validity of certain decisions made by the authors. We varied the number of scales in CM Pooling to check if the selected number was optimal, which our results showed to be true. We then tested using max pooling instead of CM Pooling and see the CM Pooling results in a better performance. We also investigated the effect of using average pooling instead of max pooling in the CM Pooling layer on the CNN2 model accuracy. We see in the CM Pooling layer, Using max pooling is more accurate than average pooling. Lastly,

Experiment	Dataset	Test Accuracy(%)
CNN^2	ModelNet2D	99.2
CNN^2	SmallNORB	90.5
Vanilla CNN	ModelNet2D	99.6
Vanilla CNN	SmallNORB	79.6
Pooling After Convolution	ModelNet2D	98.7
Pooling After Convolution	SmallNORB	89.6
Max Pooling CNN^2	ModelNet2D	99.4
Max Pooling CNN^2	SmallNORB	86.1
CM Pooling Vanilla CNN	ModelNet2D	98.5
CM Pooling Vanilla CNN	SmallNORB	88.1
One Feedforward Pathway	ModelNet2D	99.4
One Feedforward Pathway	SmallNORB	82.6
CNN^2 w/o Augmentation	ModelNet2D	99.5
CNN^2 w/o Augmentation	SmallNORB	83.2
Fusing First and Last Layer	ModelNet2D	99.2
Fusing First and Last Layer	SmallNORB	84.4

Table 1: Testing Accuracies of Various Experiments

Experiment	Dataset	Test Accuracy(%)
(1,3) Scales in CM Pooling	ModelNet2D	98.7
(1,3) Scales in CM Pooling	SmallNORB	85.5
(1,3,5,7) Scales in CM Pooling	ModelNet2D	99.1
(1,3,5,7) Scales in CM Pooling	SmallNORB	87.3
Average Pooling in CM Pooling	ModelNet2D	99.4
Average Pooling in CM Pooling	SmallNORB	90.5
2 CM Pooling + Conv. Layers	ModelNet2D	99.3
2 CM Pooling + Conv. Layers	SmallNORB	99.3
4 CM Pooling + Conv. Layers	ModelNet2D	87.5
4 CM Pooling + Conv. Layers	SmallNORB	86.1

Table 2: Additional Experiments

we tested a varying number of CM Pooling and convolutional layers. Our results showed no real difference on the ModelNet2D set, but showed a decline in accuracy for the SmallNORB set.

5.1 What was easy

The authors of the original paper gave a clear explanation of what CNN^2 is as a concept, allowing us to have a good general idea of what to expect in the code. The source code itself was bug-free and so we were able to pull it from GitHub to run. Furthermore, the datasets they used were also stored so we did not have to go and find themselves, or manipulate the existing set as in the case of ModelNet2D.

5.2 What was difficult

The original source code was unfortunately not well commented to explain the development of CNN^2. Therefore, we had to spend some extra time trying to understand the code. Furthermore, in the provided dataset, the RGB-D Data was unavailable so we were not be able to reproduce those test results.

5.3 Communication with original authors

We sent email to the authors regarding the “cp.ckpt” file which is the model weights stored in TensorFlow format, as they have not provided this file in their Github repository. According to a reply we received from Wei-Da Che, they didn’t attach the checkpoints on the GitHub due to the limitation of theworkspace, so we could train the model by ourself.

References

[1] Chun-Fu Chen et al. *Big-Little Net: An Efficient Multi-Scale Feature Representation for Visual and Speech Recognition*. 2019. arXiv: 1807.03848 [cs.CV].

[2] Wei-Da Chen and Shan-Hung (Brandon) Wu. "CNN^{2} $: ViewpointGeneralizationviaaBinocularVision$". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https : / / proceedings . neurips . cc / paper / 2019 / file / dabd8d2ce74e782c65a973ef76fd540b-Paper.pdf.

[3] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. *Unsupervised Monocular Depth Estimation with Left-Right Consistency*. 2017. arXiv: 1609.03677 [cs.CV].

[4] R von der Heydt, E Peterhans, and G Baumgartner. "Illusory contours and cortical neuron responses". In: *Science* 224.4654 (1984), pp. 1260–1262. ISSN: 0036-8075. DOI: 10.1126/science.6539501. eprint: https://science.sciencemag.org/content/224/4654/1260.full.pdf. URL: https://science.sciencemag.org/content/224/4654/1260.

[5] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. "Matrix capsules with EM routing". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=HJWLfGWRb.

[6] Yann Lecun, Fu Huang, and L. Bottou. "Learning methods for generic object recognition with invariance to pose and lighting". In: vol. 2. Jan. 2004, pp. II–97. ISBN: 0-7695-2158-4. DOI: 10.1109/CVPR.2004.1315150.

[7] Xinchen Yan et al. *Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision*. 2017. arXiv: 1612.00814 [cs.CV].