Github repository of project:
https://github.com/COMP6248-Reproducability-Challenge/Topological-CNN

REPRODUCING TOPOLOGICAL CONVOLUTIONAL NEURAL NETWORKS

Gopika Bejoy, Palak Jain, Benjamin Lellouch & Gabija Petroskeviciute
University of Southampton
Southampton, SO17 1BJ, UK

ABSTRACT

This paper attempts to reproduce the findings in the paper "Topological Convolutional Neural Networks" submitted in the NeurIPS 2020 Workshop on Topological Data Analysis and Beyond. The original paper introduced a new CNN framework called Topological CNN (TCNN) which derived new topological convolutional layers from CNN weights through topological data analysis (TDA). This paper attempts to implement the TCNNs and replicate the experiments stated within the original paper to comment on its reproducibility. Our results show that most of the experiments and TCNNs were not reproducible.

1 INTRODUCTION

Love et al. (2020) carried out four experiments, comparing TCNNs to conventional CNNs in different situations to show the former's greater generalisability, interpretability, and robustness to noise. To reproduce these findings, TCNNS and experiments were implemented on the basis of the information provided in the original paper. This allowed for investigation on which parts of the paper are suitable to reproduce, the essential/missing information for reproducing implementing the paper, the cost in terms of time and resources required for the reproduction, and the original paper results and claims compared to the reproduced results.

The paper being reproduced introduced TCNNs which utilised theoretical topological knowledge of natural images and the topology obtained from trained convolution filters which can be used in image classification. These TCNNS are created by combining custom convolutional layers introduced in the original paper: Type 1 is called the Circle One Layer (*COL*) or Klein One Layer (*KOL*) and Type 2 is Circle Filters Layer (*CF*) or Klein Filters Layer (*KF*). Love et al. (2020) define a standard convolutional layer as a Normal One Layer (*NOL*).

2 METHODOLOGY

The TCNN models were implemented from scratch based on the information provided in the original paper Love et al. (2020). We also referred to the additional paper Love et al. (2021) published by the authors on the topological deep learning to understand the mathematical formulation of the topological layers. Our work focused on the implementation of two TCNN layers, namely *KF* and *CF*, as these were described in more detail. Furthermore, only experiments that test *KF* and *CF* layers were conducted. We first compared our reproduced filters with the filters provided in the original paper during the interpretability experiment. We then conducted experiments to see if models containing these new layers are more robust to noise and able to generalise better on digit and binary classify problems.

3 IMPLEMENTATION

The main challenge was to reproduce the layers *CF* and *KF* as it required substantial background knowledge in TDA, an area that the team members were not familiar with. We aimed to generate *CF* and *KF* filters based on the equations **1** and **2** found in Love et al. (2020) as well as **Definition.**

9 found in Love et al. (2021) . *KF* filters were generated based on two angles in the Klein bottle and *CF* filters were generated based on angles on a primary circle. As mentioned in Love et al. (2020) *CF* and *KF* layers are fixed, thus during the training these layers were frozen. The TCNN models were created by combining *CF* and *KF* layers with a conventional convolutional layer (*CF+NOL*, *KF+NOL*), or relying only on *KF* layers (*KF+KF*).

The implementation was primarily done using the PyTorch framework. Our implemented *CF* and *KF* layers extend the torch.nn.Module making them reusable in any other PyTorch model. We then used these layers to generate the TCNN models which are also children of the torch.nn.Module class. These models consist of two layers with ReLU activations followed by a flattening layer at the end. Training details such as a loss function and a choice of the optimiser were not given. Thus, the categorical cross-entropy loss was chosen as it is a common loss function for multi-class classification. The ADAM optimiser was also adopted because of its popularity and easy of use. Although the links to the datasets were given in the original paper, most of the datasets were loaded using the torchvision package. The interpretability and generalisability experiments were carried out on a GTX1050 with 4GB of VRAM while the synthetic experiments were carried out on an RTX2060 with 6GB of VRAM.

3.1 Reproducing the Experiments

The meta-parameters such as kernel size, batch size, learning rate used for the following experiments, as well as train and test size splits were the same as those given in Sections 4.2 and 4.3 of the Supplementary Material section of the original paper.

3.1.1 Interprebability

In this experiment, the weights of three convolutional layers (*NOL*, *CF*, *KF*) were visualised using heatmaps and their activations were applied to the MINST image for digit five. The CF filters were generated by specifying 16 evenly spaced angles around the circle and 16 KF filters were created by specifying 4 evenly spaced values for each of 2 angles on the Klein bottle. Training of the *CF* and *KF* layers was not required as the weights of these layers are always fixed. In this experiment, we investigate whether our filters, created using the limited amount of information provided, match the filters represented in Love et al. (2020). We also aimed to observe whether activations of *CF* and *KF* filers are easier to interpret than the conventional trained CNN layer (NOL).

3.1.2 Synthetic Experiments

The motivation behind this experiment is to test the robustness to noise of *KF* and *CF*. As the *KF* and *CF* layers are pre-trained layers, we expect them to provide meaningful features to the model and smooth the noise out prior to the learnable layers. To test this hypothesised two scenarios were envisioned: one where the models are trained on a noisy trainset and evaluated on a "clean" testset (without global noise added) and one where the models are trained on a "clean" trainset and evaluated on a noisy testset. This experiment was carried out using the MNIST dataset. It was divided according to a 85%/15% train-test split and global noise was applied according to the parameters specified in Love et al. (2020). We trained the following models: *KF+NOL*, *CF+NOL*, *NOL+NOL*.

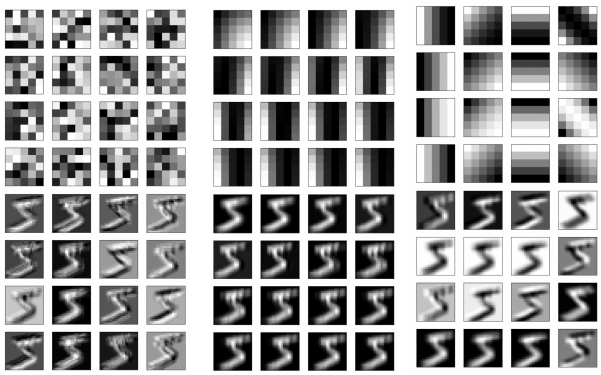
3.1.3 Generalizability

The aim of this experiment was to investigate TCNN models' ability to generalize on unseen datasets. Specifically, the accuracy of predictions was compared between the models when trained on one dataset then tested on a similar one. The digit datasets, MNIST and SVHN, as well as the cats and dogs dataset from Kaggle and CIFAR-10 were used. All datasets were loaded using torchvision except for the Kaggle dataset, which was downloaded from the link provided in the paper. The entire datasets were used when training and testing (except for the Kaggle data which had corrupted images). For training and testing datasets pairs, higher resolution images were down-resolved to have the same image resolutions.

4 RESULTS

4.1 INTERPREBABILITY

Figure 1 shows the weights of the first layers of NOL, *CF* and *KF* models and corresponding activations on the image containing digit five from the MINST dataset. The testing accuracy of 94% has been obtained for the trained NOL model compared to 99% in Love et al. (2020). It can be seen from Figure 1, that *KF* and *CF* are easier to interpret as stated in the original paper. A clear pattern can be seen in their heat maps as opposed to the heat maps of the NOL filter. Although similar, the patterns shown for reproduced filters differ from that of the original paper.



((a)) ((b)) ((c))

Figure 1: Visualisation of weights (top row) and activations (bottom row) for three convolutional layers, namely NOL (a), CF (b), and KF (c). *Original paper:page 3 Figure 1* (Love et al. (2020))

4.1.1 SYNTHETIC EXPERIMENTS

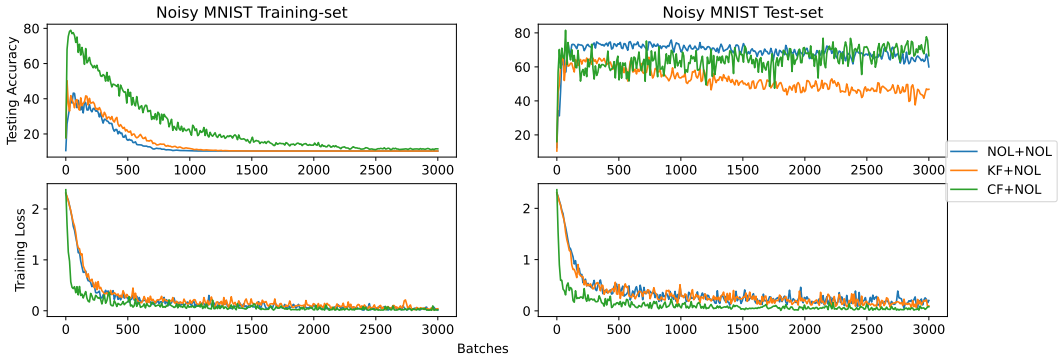


Figure 2: Reproduced results. Authors' results are in *page 4 Figure 3* (Love et al. (2020))

Overall, our results tell a very different story compared to the original paper's results. The most noticeable difference is how well our baseline model *NOL+NOL* performs in relation to the TCNN models in both scenarios. In some cases, *NOL+NOL* actually outperforms topological models. This observation makes it difficult for us to argue that these topological layers add robustness to noise. In the "Noisy MNIST Training-set" scenario, the different models exhibit similar behaviours: their testing accuracy peaks very early on and then starts catastrophically fitting to the noise. We observe that *CF + NOL* peaks much higher ($\approx 80\%$). This suggests that the *CF* layer provided meaningful features from the beginning but that the learnable parameters started to fit to the noise in the dataset. This behaviour implies that the topological layers failed to smooth noise out for the learnable layers.

The overfitting of the TCNNs in the author's results is present although a lot less pronounced. In the "Noisy MNIST Test-set" scenario, both *NOL+NOL*s' testing accuracy stabilises but ours stabilises at around 70% whilst theirs stabilises at around 20%. With our *NOL+NOL* performance on par if not better than the TCNNs, it is difficult to arrive to similar conclusions as the original authors did.

4.1.2 GENERALIZABILITY

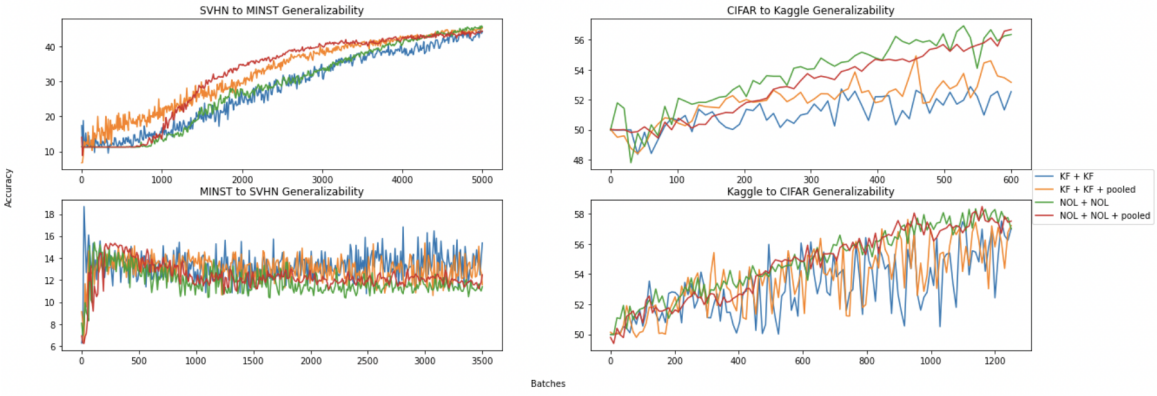


Figure 3: Reproduced results. Authors' results are in *page 5 Figure 5* (Love et al. (2020))

In general, the results show that none of the models involving *KF* layers outperform the conventional CNN models as stated in the original paper. For some datasets, models with *KF* layers achieve testing accuracies close to those of the *NOL* models at the end of the training. This is visible when models have been trained on SVHN and tested on MINST, as well as when the models have been trained on Kaggle and tested on CIFAR. Moreover, for *KF* models the addition of pooling layers gives an improvement on the generalizability for cats and dogs datasets. The significant improvement can be seen when *KF + KF* pooled model has been trained on CIFAR and tested on Kaggle dataset. This observation was also made in the original paper. For time and memory costs, CIFAR and Kaggle were trained on CPU and MNIST and SVHN on GPU; all training was done on 16GB RAM. Training times for generalizability dataset pairings in hours and minutes are, MNIST-SVHN - 08:32, SVHN-MNIST - 04:52, CIFAR-KAGGLE - 04:38, KAGGLE-CIFAR - 00:43.

5 CONCLUSION AND REFLECTION

The reproduced filters and experiments differed significantly from the original paper. The baseline model performed significantly better in our implementation than theirs. This was slightly suspicious, as if similar behaviour was observed in their experiments; it could negate their claims that these topological layers add generalisability potential and robustness to noise to CNNs. This could be due to mistakes in the formulas provided or missing information needed for implementation which resulted in our filters exhibiting different patterns than theirs in the interpretability experiment. Difference in experimental set up such as the loss function used may also have contributed to this difference in performance. The most expensive experiment in terms of time and resources was generalisability, which took a total training time of 18 hours; the other experiments were much quicker. It should also be noted that this paper was written from the point of view of TDA which makes it difficult for novice Deep Learning practitioners such as ourselves to interpret and implement.

REFERENCES

Ephy Love, Benjamin Filippenko, Vasileios Maroulas, and Gunnar E Carlsson. Topological convolutional neural networks. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020.

Ephy R. Love, Benjamin Filippenko, Vasileios Maroulas, and Gunnar Carlsson. Topological deep learning. *CoRR*, abs/2101.05778, 2021. URL `https://arxiv.org/abs/2101.05778`.