Network Inversion of Convolutional Neural Nets (Student Abstract)

Pirzada Suhail, Amit Sethi

IIT Bombay
Mumbai, IN
psuhail, asethi@iitb.ac.in

Abstract

Neural networks have emerged as powerful tools across various applications, yet their decision-making process often remains opaque, leading to them being perceived as "black boxes." This opacity raises concerns about their interpretability and reliability, especially in safety-critical scenarios. Network inversion techniques offer a solution by allowing us to peek inside these black boxes, revealing the features and patterns learned by the networks behind their decision-making processes and thereby provide valuable insights into how neural networks arrive at their conclusions, making them more interpretable and trustworthy. This paper presents a simple yet effective approach to network inversion using a meticulously conditioned generator that learns the data distribution in the input space of the trained neural network, enabling the reconstruction of inputs that would most likely lead to the desired outputs. To capture the diversity in the input space for a given output, instead of simply revealing the conditioning labels to the generator, we encode the conditioning label information into vectors and intermediate matrices and further minimize the cosine similarity between features of the generated images.

1 Introduction

Neural networks have become indispensable in a wide array of applications, ranging from image recognition and natural language processing to autonomous driving and medical diagnostics. Despite their remarkable performance, the decision-making processes within these networks often remain elusive, earning them the moniker "black boxes." This opacity poses significant challenges, particularly in scenarios where interpretability and reliability are paramount, such as in safety-critical applications. Network inversion offers a solution by allowing us to inspect and reconstruct inputs that trigger specific outputs. By inverting the network, we can reconstruct inputs that are likely to produce specific outputs, thereby gaining insights into the network's learned data distribution and feature extraction processes.

Network Inversion has been studied in different cases as in (Saad and Wunsch 2007),(Jensen et al. 1999), (Wong 2017), (Yang et al. 2019),(Kumar and Levine 2020) and (Ansari et al. 2022). Early research on inversion for multilayer perceptrons in (Jensen et al. 1999), (Kindermann and Linden 1990),(Saad and Wunsch 2007) derived from the backpropagation algorithm, demonstrates the utility of this method in applications like digit recognition. Subsequently multi-element evolutionary inversion procedures were introduced, that unlike gradient descent stand out for their ability to simultaneously discover multiple inversion points. Recent work by (Liu et al. 2022) proposes learning a loss landscape where gradient descent becomes efficient, thus significantly improving the speed and stability of the inversion process. Later in (Suhail 2024) Network Inversion is performed by encoding the neural network into a Conjuctive Normal Form (CNF) Propositinal Formula and then looking for satisfying assignments to the constrained CNF formula using SAT Solvers.

In this paper, we present a simple yet effective approach to network inversion that aims to strike a balance between computational efficiency and the diversity of generated inputs by using a carefully conditioned generator trained to learn the data distribution in the input space of a classifier. To ensure the generator learns a diverse set of inputs, we alter the conditioning from simple labels to vectors and matrices that encode the label information within and discourage easy shortcut solutions. This diversity is further reinforced through the application of heavy dropout during the generation process, specifically during up-convolution, and by minimizing the cosine similarity between the features of the generated images as returned by the classifier which helps in achieving a more varied and representative set of generated images, each corresponding to different conditioning vectors. By revealing the hidden patterns and features that influence network predictions, we gain a more comprehensive understanding of neural network behavior which is crucial in improving their interpretability.

2 Methodology

Our approach to Network Inversion uses a conditioned generator that learns the data distributions in the input space of the trained classifier by simple modification of the training objectives as shown in Figure 1. Inversion is performed on a classifier which includes convolution and fully connected layers, appropriate to the classification task along with Leaky-ReLU layers(Xu et al. 2015) and Dropout layers (Srivastava et al. 2014).

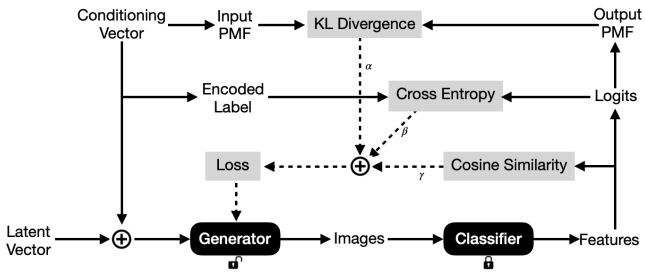

Figure 1: Schematic Representation of the Inversion Process

The images in the input space are generated by a conditioned generator that builds up from a latent vector through up-convolution operations to the required size. While generators are typically conditioned on an embedding learned from a label for generative tasks, we observe its ineffectiveness for network inversion. Instead, we propose a more intense conditioning mechanism using vectors by encoding the label information into an n-dimensional vector for an n-class classification task that are randomly generated from a normal distribution and soft-maxed to represent the input distribution for the generated images. A linear layer maps the n-dimensional vector to the hidden dimensions of the generator before concatenation with the latent vector. The arg-max index of the soft-maxed vectors serves as the conditioning label in the cross-entropy loss function, promoting greater diversity in the generated images by varying the arg-max element intensity. The generator includes two up-sampling stages of the linear vector, the vector is upsampled into nxn dimensions followed by concatenation with an nxn hot matrix and further upsampling to the image size. The nxn hot matrix has the elements at arg-max index across both rows and columns set to 1 while others to 0.

The main objective of Network Inversion is to generate images that when passed through the classifier will elicit the same label as the generator was conditioned to. Achieving this objective through a straightforward cross-entropy loss between the conditioning label and the classifier's output leads to mode collapse. Hence we propose Inversion loss $\mathcal{L}_{\text{Inv}}$ as a collection of losses defined as:

$$\mathcal{L}_{\text{Inv}} = \alpha \cdot \mathcal{L}_{\text{KL}} + \beta \cdot \mathcal{L}_{\text{CE}} + \gamma \cdot \mathcal{L}_{\text{Cosine}}$$

Here, $\mathcal{L}_{\text{KL}}$ is the KL Divergence loss, $\mathcal{L}_{\text{CE}}$ is the Cross Entropy loss, and $\mathcal{L}_{\text{Cosine}}$ is the Cosine Similarity loss. The hyperparameters α, β, γ control the contribution of each individual loss term. They are defined as:

$$\mathcal{L}_{\text{KL}} = \sum_i P(i) \log \frac{P(i)}{Q(i)}, \quad \mathcal{L}_{\text{CE}} = -\sum_i y_i \log(\hat{y}_i),$$

$$\mathcal{L}_{\text{Cosine}} = \frac{1}{N(N-1)} \sum_{i \neq j} \cos(\theta_{ij})$$

where $\mathcal{L}_{\text{KL}}$ represents the KL Divergence between the input distribution P and the output distribution Q, y_i is the set encoded label, $\hat{y}_i$ is the predicted label from the classifier, and $\cos(\theta_{ij})$ is the cosine similarity between the features of generated images i and j in a batch of N.

3 Results

The Network Inversion Technique was evaluated on classifiers trained on MNIST, FashionMNIST, SVHN and CIFAR10 datasets by training a generator to produce images that, when passed through a classifier, elicit the desired labels. While the generator is based on Vector-Matrix Conditioning followed by multiple layers of transposed convolutions, batch normalization (Ioffe and Szegedy 2015) and dropout layers (Srivastava et al. 2014) to encourage diversity in the generated images.

The resulting generated images are visualized to assess the quality and diversity of the generated samples in Figure 2 for all 10 classes of MNIST and FashionMNIST, SVHN and CIFAR10 respectively. While each row corresponds to a different class each column corresponds to a different generator and as can be observed the images within each row represent the diversity of samples generated for that class. It is also observed that high weightage to cosine similarity increases both the inter-class and the intra-class diversity in the generated samples of a single generator.

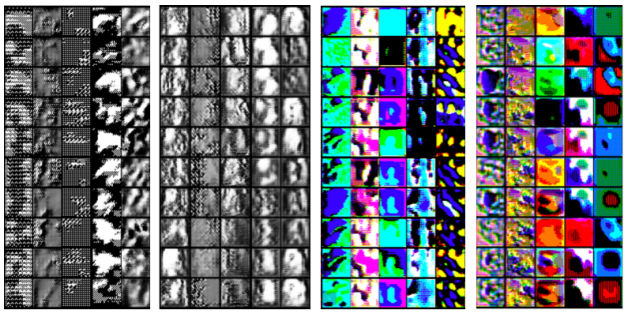

Figure 2: Inverted Samples for MNIST, FashionMNIST, SVHN and CIFAR10. Each row represents a different class.

Network inversion may occasionally generate samples that resemble the training data. However, to further encourage the generation of training-like data, we exploit the robustness properties of the classifiers. Typically, classifiers are not very robust to perturbations in randomly generated images but they exhibit some degree of robustness around the training examples. Consequently, by leveraging this robustness, network inversion can be guided to generate and reconstruct data that resembles the training set by encouraging the generator to produce robust inverted samples such that the perturbations of the generated images within a certain bound will still result in the same desired label.

4 Conclusion & Future Work

This paper introduced a novel approach to network inversion, utilizing a single vector-matrix conditioned generator to generate a diverse set of inputs with desired output labels.

Future work will aim to quantify the aspects of the inversion technique and explore its potential in interpretability using sparse auto-encoders on the distinct set of generated images and in training data reconstruction by encouraging the generation of robust inverted samples.

References

Ansari, N.; Seidel, H.-P.; Ferdowsi, N. V.; and Babaei, V. 2022. Autoinverse: Uncertainty Aware Inversion of Neural Networks. arXiv:2208.13780.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 448–456. Lille, France: PMLR.

Jensen, C.; Reed, R.; Marks, R.; El-Sharkawi, M.; Jung, J.-B.; Miyamoto, R.; Anderson, G.; and Eggen, C. 1999. Inversion of feedforward neural networks: algorithms and applications. *Proceedings of the IEEE*, 87(9): 1536–1549.

Kindermann, J.; and Linden, A. 1990. Inversion of neural networks by gradient descent. *Parallel Computing*, 14(3): 277–286.

Kumar, A.; and Levine, S. 2020. Model Inversion Networks for Model-Based Optimization. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 5126–5137. Curran Associates, Inc.

Liu, R.; Mao, C.; Tendulkar, P.; Wang, H.; and Vondrick, C. 2022. Landscape Learning for Neural Network Inversion. arXiv:2206.09027.

Saad, E. W.; and Wunsch, D. C. 2007. Neural network explanation using inversion. *Neural Networks*, 20(1): 78–93.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958.

Suhail, P. 2024. Network Inversion of Binarised Neural Nets. In *The Second Tiny Papers Track at ICLR 2024*.

Wong, E. 2017. Neural network inversion beyond gradient descent. In *WOML NIPS*.

Xu, B.; Wang, N.; Chen, T.; and Li, M. 2015. Empirical Evaluation of Rectified Activations in Convolutional Network. arXiv:1505.00853.

Yang, Z.; Zhang, J.; Chang, E.-C.; and Liang, Z. 2019. Neural Network Inversion in Adversarial Setting via Background Knowledge Alignment. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, 225–240. New York, NY, USA: Association for Computing Machinery. ISBN 9781450367479.