Network Inversion of Convolutional Neural Nets

Pirzada Suhail*
Department of Electrical Engineering
IIT Bombay
Mumbai, IN 400076
psuhail@iitb.ac.in

Amit Sethi
Department of Electrical Engineering
IIT Bombay
Mumbai, IN 400076
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. Additionally, we incorporate feature orthogonality as a regularization term to boost image diversity which penalises the deviations of the Gram matrix of the features from the identity matrix, ensuring orthogonality and promoting distinct, non-redundant representations for each label.

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 techniques provide a mechanism to reveal the internal representations and decision-making pathways of neural networks. 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. This capability is crucial for enhancing the interpretability and transparency of neural networks, making them more trustworthy and reliable.

The concept of neural network inversion has garnered significant attention as a method for visualizing and understanding the internal mechanisms of neural networks. Inversion seeks to identify input patterns that closely approximate a given output target, thereby revealing the information processing capabilities embedded within the network's weights. Early research on inversion for multi-layer

*Corresponding Author

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

perceptrons in [Kindermann and Linden, 1990], derived from the back-propagation algorithm, demonstrates the utility of this method in applications like digit recognition highlighting that while multi-layer perceptrons exhibit strong generalization capabilities—successfully classifying untrained digits—they often falter in rejecting counterexamples, such as random patterns.

Subsequently [Jensen et al., 1999] expanded on this idea by proposing evolutionary inversion procedures for feed-forward networks that stands out for its ability to identify multiple inversion points simultaneously, providing a more comprehensive view of the network's input-output relationships. The paper [Saad and Wunsch, 2007] explores the lack of explanation capability in artificial neural networks (ANNs) and introduces an inversion-based method for rule extraction to calculate the input patterns that correspond to specific output targets. [Wong, 2017] addresses the problem of inverting deep networks to find inputs that minimize certain output criteria by reformulating network propagation as a constrained optimization problem. Model Inversion attacks in adversarial settings are studied in [Yang et al., 2019], where an attacker aims to infer training data from a model's predictions by training a secondary neural network to perform the inversion.

The paper [Kumar and Levine, 2020] presents a method for tackling data-driven optimization problems, where the goal is to find inputs that maximize an unknown score function by proposing Model Inversion Networks (MINs), which learn an inverse mapping from scores to inputs. While [Ansari et al., 2022] introduces an automated method for inversion by focusing on the reliability of inverse solutions by seeking inverse solutions near reliable data points that are sampled from the forward process and used for training the surrogate model. The traditional methods for network inversion often rely on gradient descent through a highly non-convex loss landscape, leading to slow and unstable optimization processes. To address these challenges, 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. Similarly Suhail [2024] proposes an alternate approach to inversion by encoding the network into a Conjunctive Normal Form (CNF) propositional formula and using SAT solvers and samplers to find satisfying assignments for the constrained CNF formula.

In this paper, we present a network inversion technique that learns the input space corresponding to different classes within a classifier using a single conditioned generator trained to generate a diverse set of samples from the input space with desired labels guided by a combination of losses including cross-entropy, KL Divergence, cosine similarity and feature orthogonality. 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. This diversity is reinforced through the application of heavy dropout during the generation process and by minimizing the cosine similarity between the features of the generated images. Additionally, we incorporate feature orthogonality as a regularization term, by penalizing deviations of the Gram matrix of the features from the identity matrix. The orthogonality loss combined with cosine similarity helps achieve a more varied set of generated images, each corresponding to different conditioning vectors.

2 Methodology & Implementation

Our approach to Network Inversion and subsequent training data reconstruction uses a carefully conditioned generator that learns diverse data distributions in the input space of the trained classifier.

In this paper inversion and reconstruction is performed on a classifier which includes convolution and fully connected layers as appropriate to the classification task. We use standard non-linearity layers like Leaky-ReLU [Xu et al., 2015] and Dropout layers [Srivastava et al., 2014] in the classifier for regularisation purposes to discourage memorisation.

The images in the input space of the classifier will be generated by an appropriately conditioned generator. The generator builds up from a latent vector by up-convolution operations to generate the image of the given size. While generators are conventionally conditioned on an embedding learnt of a label for generative modelling tasks, we given its simplicity, observe its ineffectiveness in network inversion and instead propose more intense conditioning mechanism using vectors and matrices. While Label Conditioning can be used for inversion, the inverted samples do not seem to have the diversity that is expected of the inversion process due to the simplicity and varying confidence behind the same label.

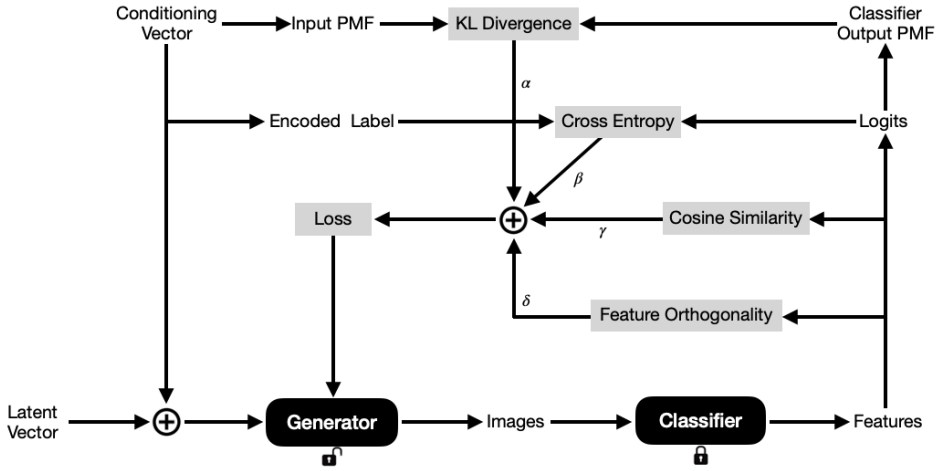

Figure 1: Proposed Approach to Network Inversion

In order to achieve more diversity in the generated images, the conditioning mechanism of the generator is altered by encoding the label information into an N-dimensional vector for an N-class classification task. The argmax index of the soft-maxed vectors now serves as the de facto conditioning label, which can be used in the cross-entropy loss function without being explicitly revealed to the generator. This can be further extended into Matrix Conditioning which apparently serves as a better prior in case of generating images and allows for more ways to encode the label information for a better capture of the diversity in the inversion process. In its simplest form we use a Hot Conditioning Matrix in which an NXN dimensional matrix is defined such that all the elements in a given row and column (same index) across the matrix are set to one while the rest all entries are zeroes. The index of the row or column set to 1 now serves as the label for the conditioning purposes. The conditioning matrix is concatenated with the latent vector intermediately after up-sampling it to NXN spatial dimensions, while the generation upto this point remains unconditioned. Since the generation is initially unconditioned in Intermediate Matrix Conditioning, we combine both vector and matrix conditioning, in which vectors are used for early conditioning of the generator upto NXN spatial dimensions followed by concatenation of the conditioning matrix for subsequent generation. The argmax index of the vector, which is the same as the row or column index set to high in the matrix, now serves as the conditioning label.

With the classifier trained, the inversion is performed by training the generator to learn the data distribution for different classes in the input space of the classifier as shown schematically in Figure 1 using a combined loss function $\mathcal{L}_{\text{Inv}}$ defined as:

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

where $\mathcal{L}_{\text{KL}}$ is the KL Divergence loss, $\mathcal{L}_{\text{CE}}$ is the Cross Entropy loss, $\mathcal{L}_{\text{Cosine}}$ is the Cosine Similarity loss, and $\mathcal{L}_{\text{Ortho}}$ is the Feature Orthogonality loss. The hyperparameters $\alpha, \beta, \gamma, \delta$ control the contribution of each individual loss term defined as:

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

$$\mathcal{L}_{\text{CE}} = -\sum_i y_i \log(\hat{y}_i)$$

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

$$\mathcal{L}_{\text{Ortho}} = \frac{1}{N^2} \sum_{i,j} (G_{ij} - \delta_{ij})^2$$

where D_{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, $\cos(\theta_{ij})$ represents the cosine similarity between features of generated images i and j, G_{ij} is the element of the Gram matrix, and δ_{ij} is the Kronecker delta function. N is the number of feature vectors in the batch.

3 Experiments & Results

In this section, we present the experimental results obtained by applying our network inversion and reconstruction technique on the MNIST [Deng, 2012], FashionMNIST [Xiao et al., 2017], SVHN and CIFAR-10 [Krizhevsky et al.] datasets by training a generator to produce images with desired labels. The classifier is initially normally trained on a dataset and then held in evaluation for the purpose of inversion and reconstruction. The images generated by the conditioned generator corresponding to the latent and the conditioning vectors are then passed through the classifier.

The classifier is a simple multi-layer convolutional neural network consisting of convolutional layers, dropout layers, batch normalization, and leaky-relu activation followed by fully connected layers and softmax for classification. While the generator is based on Vector-Matrix Conditioning in which the class labels are encoded into random softmaxed vectors concatenated with the latent vector 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. Once the vectors are upsampled to NXN spatial dimensions for an N class classification task they are concatenated with a conditioning matrix for subsequent generation upto the required image size of 28X28 or 32X32.

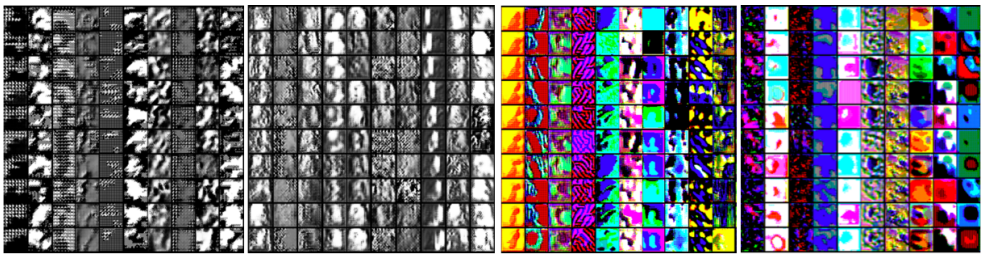

Figure 2: Inverted Images for all 10 classes in MNIST, FashionMNIST, SVHN & CIFAR-10.

The inverted images are visualized to assess the diversity of the generated samples in Figure 2 for MNIST, FashionMNIST, SVHN and CIFAR-10 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 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.

4 Conclusion & Future Work

This paper introduced a novel approach to network inversion, utilizing a conditioned generator to generate a diverse set of inputs with desired output labels. By shifting from simple label conditioning to vector encoding and incorporating heavy dropout during the generation process, our method complicates the conditioning mechanism, encouraging the generator to explore a more extensive range of the data distribution. Additionally, by minimizing cosine similarity and using feature orthogonality between features of generated images, we ensure that the generator learns a broad representation of the input space for any given output.

Our approach is useful in improving the interpretability of neural networks by revealing the internal patterns and decision-making processes. This enhancement contributes to increased safety and robustness against adversarial perturbations and counterexamples. In Future we would like to explore the applications of our inversion technique in generating minimally adversarial examples from the input space, effective reconstruction of training-like data, and iterative refinement of the classifier using inverted samples enhancing the classifier's ability to handle counterexamples. We would

also like to quantify the aspects of the inversion technique and explore its potential in enhancing interpretability and robustness across various real-world tasks.

Additionally, we would like to investigate applications of network inversion in out-of-distribution (OOD) detection, domain generalization, and uncertainty estimation to improve the safety and reliability of neural networks in diverse and challenging environments.

References

Navid Ansari, Hans-Peter Seidel, Nima Vahidi Ferdowsi, and Vahid Babaei. Autoinverse: Uncertainty aware inversion of neural networks, 2022. URL https://arxiv.org/abs/2208.13780.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

C.A. Jensen, R.D. Reed, R.J. Marks, M.A. El-Sharkawi, Jae-Byung Jung, R.T. Miyamoto, G.M. Anderson, and C.J. Eggen. Inversion of feedforward neural networks: algorithms and applications. *Proceedings of the IEEE*, 87(9):1536–1549, 1999. doi: 10.1109/5.784232.

J Kindermann and A Linden. Inversion of neural networks by gradient descent. *Parallel Computing*, 14(3):277–286, 1990. ISSN 0167-8191. doi: https://doi.org/10.1016/0167-8191(90)90081-J. URL https://www.sciencedirect.com/science/article/pii/016781919090081J.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5126–5137. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/373e4c5d8edfa8b74fd4b6791d0cf6dc-Paper.pdf.

Ruoshi Liu, Chengzhi Mao, Purva Tendulkar, Hao Wang, and Carl Vondrick. Landscape learning for neural network inversion, 2022. URL https://arxiv.org/abs/2206.09027.

Emad W. Saad and Donald C. Wunsch. Neural network explanation using inversion. *Neural Networks*, 20(1):78–93, 2007. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2006.07.005. URL https://www.sciencedirect.com/science/article/pii/S0893608006001730.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Pirzada Suhail. Network inversion of binarised neural nets. In *The Second Tiny Papers Track at ICLR 2024*, 2024. URL https://openreview.net/forum?id=zKcB0vb7qd.

Eric Wong. Neural network inversion beyond gradient descent. In *WOML NIPS*, 2017. URL https://api.semanticscholar.org/CorpusID:208231247.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015. URL https://arxiv.org/abs/1505.00853.

Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. 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, page 225–240, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. doi: 10.1145/3319535.3354261. URL https://doi.org/10.1145/3319535.3354261.