2024 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI) | 979-8-3503-6011-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/SSIAI59505.2024.10508687

A low-rank tensor reconstruction and denoising method for enhancing CNN performance

Rohin Harikumar
Department of Mathematical Sciences
The University of Texas at Dallas
Richardson, TX, USA
rxh210003@utdallas.edu

Susan E. Minkoff
Department of Mathematical Sciences
The University of Texas at Dallas
Richardson, TX, USA
sminkoff@utdallas.edu

Yifei Lou
Department of Mathematics
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
yflou@unc.edu

***Abstract*—Neural network training data is often corrupted by equipment malfunction or noise leading to red blurry and incomplete data. This paper proposes a combination of a reconstruction technique and a neural network to deal with data corruption in a machine vision task. Specifically, we consider minimizing the tensor nuclear norm for low-rank data completion and denoising and demonstrate the method's effectiveness using a convolutional neural network (CNN) for image classification. We conduct classification experiments on 3 datasets, showing consistently that training on reconstructed images achieves improved accuracy ranging from 7-25% over training using corrupted data.**

***Index Terms*—Low-rank tensor completion, tensor nuclear norm, convolutional neural network, alternating direction method of multipliers, image classification**

I. Introduction

Data corruption refers to incomplete or blurry images from equipment failure, decimation, or noise. Most machine vision systems are sensitive to data corruption. For example, convolutional neural networks (CNNs) are primarily used for image classification tasks and are typically trained and tested on images with negligible levels of corruption. Dodge et al. [4] demonstrate that augmenting the training dataset with noise can negatively affect the accuracy of large-scale and state-of-the-art CNNs. Furthermore, they show that the reduced performance under noise is not limited to a particular CNN architecture. As CNNs struggle to make satisfactory associations of corrupted image patterns to the correct labels, an image reconstruction algorithm may be employed to reconstruct the patterns necessary for the CNN to make accurate classification.

Low-rank reconstruction techniques are commonly used for completing and denoising data with high inherent redundancy such as seismic data [5], [16]. Ely et al. [5] introduce a relaxation to tensor rank called the tensor nuclear norm (TNN) and demonstrate that the alternating direction method of multipliers (ADMM) [2] can be used to minimize the TNN to reconstruct the (uncorrupted) seismic data tensor. Popa et al. (see [14], [15], [16], [13]) investigate this TNN-ADMM reconstruction procedure which we also use in this work due to its ease of implementation, effectiveness, and generalizability to higher-dimensional data tensors.

In this paper, we demonstrate the effectiveness of TNN-ADMM in completing and denoising data which we then use to train a CNN for image classification. Experimentally, we compare the performance of CNNs trained on images that are clean, images that are noisy and incomplete, and finally, images that have been completed and denoised using TNN-ADMM. We show that the CNNs trained on the TNN-ADMM reconstructed images exhibit higher accuracy compared to those trained on corrupted images, particularly when evaluated against ground truth and reconstructed testing sets. Across three datasets, we see an increase in accuracy of 7–24% for CNNs trained on reconstructed images compared to CNNs trained on corrupted images when the neural network is used to classify clean images.

II. Background

A. Tensor Algebra

In this work we consider color images stored as 3D tensors. An image is a two-dimensional array of colored squares called pixels. A pixel is a vector in R^3 with the three components representing the intensities of red, green, and blue. For example, a square color image can be stored as a 3D tensor of dimension $n \times n \times 3$.

Unlike for matrices, there is not a unique definition of tensor rank. We use tensor tubal rank, obtained from the tensor singular value decomposition (t-SVD) [5], as the definition of tensor rank. As direct minimization of tensor rank is computationally intractable [3], we use an approximation to tensor tubal rank called the tensor nuclear norm (TNN) [5].

In this paper tensors are denoted by script letters, e.g. $\mathcal{X}$, and matrices by capital letters, e.g. X. Let $\mathcal{A}$ be a 3D tensor of dimension $n \times n \times 3$. The k^{th} frontal slice of $\mathcal{A}$ is the n by n matrix $\mathcal{A}[:,:,k]$ where $\mathcal{A}[:,:,k](i,j) = \mathcal{A}(i,j,k)$. The singular value decomposition of the k^{th} frontal slice of $\mathcal{A}$ is $U_k \times S_k \times V_k^\tau$, where U_k and V_k are orthogonal matrices ($\mathcal{U}[:,:,k] = U_k$, $\mathcal{V}[:,:,k] = V_k$), $\mathcal{S}[:,:,k] = S_k$ is diagonal, and τ denotes the matrix transpose [9]. The t-SVD of the tensor $\mathcal{A}$ is a factorization of $\mathcal{A}$ into three tensors $\mathcal{U}, \mathcal{S}$ and $\mathcal{V}$ such that,

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^\top.$$

where $\top$ denotes tensor transpose [5]. Since S_k are diagonal matrices, all frontal slices of $\mathcal{S}$ are diagonal. We denote the Fourier transform of $\mathcal{A}$ along the 3^{rd} dimension as fft($\mathcal{A}$) and the inverse Fourier transform along the 3^{rd} dimension as ifft($\mathcal{A}$). The TNN of $\mathcal{A}$, denoted $||\mathcal{A}||_{\text{TNN}}$, is defined to be the sum of all diagonal entries of fft($\mathcal{S}$). The (i, i, k) entries in fft($\mathcal{S}$) for all $1 \leq i \leq n; 1 \leq k \leq 3$ are defined to be the singular values of $\mathcal{A}$.

B. Low-rank minimization

Images may be corrupted due to a lack of pixel values (decimation) and noise. Let $\mathcal{X}$ denote the ground truth image tensor. We define a decimation operator $\mathcal{A}$ of the same dimension as $\mathcal{X}$ with zeros for missing pixel values and ones elsewhere. Then $\mathcal{A}(\mathcal{X})$, the entry-wise product of $\mathcal{A}$ and $\mathcal{X}$, represents the decimated image. If $\mathcal{Y}$ is an image corrupted by both missing data and noise then,

$$\mathcal{Y} = \mathcal{A}(\mathcal{X}) + \mathcal{N},$$

where $\mathcal{N}$ is a noise term. Reconstruction of $\mathcal{X}$ given $\mathcal{A}$ and $\mathcal{Y}$ is ill-posed. Here we assume the ground truth data $\mathcal{X}$ is low rank. As rank minimization is computationally intractable, we impose the constraint that $\mathcal{X}$ minimizes an approximation to rank, in our case the tensor nuclear norm or TNN. We solve the following optimization problem [13],

$$\min_{\mathcal{X}} \lambda||\mathcal{X}||_{\text{TNN}} + \frac{1}{2}||\mathcal{Y} - \mathcal{A}(\mathcal{X})||_F^2, \tag{1}$$

where $\lambda > 0$ is a parameter that attempts to balance these two terms.

We apply the alternating direction method of multipliers (ADMM) [2] introducing the variable $\mathcal{Z}$ to decouple the TNN regularization and data fidelity terms in (1), i.e.,

$$\min_{\mathcal{X},\mathcal{Z}}\{\ \lambda||\mathcal{Z}||_{\text{TNN}} + \frac{1}{2}||\mathcal{Y} - \mathcal{A}(\mathcal{X})||_F^2\} \text{ such that } \mathcal{Z} = \mathcal{X}. \tag{2}$$

Its corresponding augmented Lagrangian is given by,

$$\begin{aligned} L_\rho(\mathcal{Z}, \mathcal{X}; \mathcal{B}) = \lambda||\mathcal{Z}||_{\text{TNN}} + \frac{1}{2}||\mathcal{Y} - \mathcal{A}(\mathcal{X})||_F^2 \\ + \rho\langle\mathcal{B}, \mathcal{X} - \mathcal{Z}\rangle_F + \frac{\rho}{2}||\mathcal{X} - \mathcal{Z}||_F^2, \end{aligned} \tag{3}$$

where $\mathcal{B}$ is a dual variable and $\rho > 0$ is the step size. ADMM iterates are as follows:

$$\mathcal{Z}^{m+1} = \arg\min_{\mathcal{Z}} L_\rho(\mathcal{Z}, \mathcal{X}^m; \mathcal{B}^m) \tag{4}$$

$$\mathcal{X}^{m+1} = \arg\min_{\mathcal{X}} L_\rho(\mathcal{Z}^{m+1}, \mathcal{X}; \mathcal{B}^m) \tag{5}$$

$$\mathcal{B}^{m+1} = \mathcal{B}^m + \mathcal{X}^{m+1} - \mathcal{Z}^{m+1}, \tag{6}$$

where m is iteration number. It is shown in Popa et al. [13] that if $\rho > 1$, the iterations (4)-(6) converge to a local minimizer of (1).

C. Convolutional Neural Networks (CNN)

Consider a set of N images denoted by $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N\}$ which is referred to as the training set. Each image $\mathcal{X}_i$ is associated with a unique integer $y_i \in \{1, 2, \cdots, C\}$, known as its label [8], where C denotes the total number of classes. The objective of image classification is to find a function F such that $F(\mathcal{X}_i) = \mathbf{y}_i$ for all $1 \leq i \leq N$, where $\mathbf{y}_i$ is the one-hot vector corresponding to the integer label y_i.

Deep neural networks (DNNs) are specialized algorithms that act as universal function approximations [7]. A typical DNN consists of arrays of neurons, called layers. Given an input vector $\mathbf{x}$, each layer evaluates a nonlinear function

$$\phi(W\mathbf{x} + \mathbf{b}), \tag{7}$$

where the matrix W is an array of numbers known as weights, the vector $\mathbf{b}$ is a bias term, and ϕ is an activation function. The nonlinearity introduced by ϕ enables the DNN to approximate any arbitrary function F. The layers are arranged sequentially so that the output from one layer becomes the input to the next. The first layer accepts the input data and the last layer is the output layer with all the layers in between called hidden layers. For example, we can write a simple two-layer DNN as

$$F(\mathcal{X}; \{W_j\}, \{\mathbf{b}_j\}) = \text{Softmax}(W_2(\text{ReLu}(W_1\mathbf{x} + \mathbf{b}_1)) + \mathbf{b}_2),$$

where $\mathbf{x}$ is an input array (for instance a frontal slice of an image $\mathcal{X}$). Softmax and ReLu [7] are standard activation functions used in the last layer and in the hidden layers, respectively.

The trainable parameters $\{W_j\}$ and $\{\mathbf{b}_j\}$ are iteratively updated using stochastic gradient descent-based algorithms (see [10], [12]) to minimize a loss function defined by

$$J(\{W_j\}, \{\mathbf{b}_j\}) = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{y}_i - \hat{\mathbf{y}}_i||^2, \tag{8}$$

where $\hat{y}_i = F(\mathcal{X}_i; \{W_j\}, \{\mathbf{b}_j\})$, and $\{\mathcal{X}_i, \mathbf{y}_i\}_{i=1}^N$ are training images with one-hot labels $\{\mathbf{y}_i\}$.

Convolutional neural networks (CNNs) [11] are a class of DNNs primarily used for processing images. In CNNs, 2D arrays of weights known as kernels capture dominant spatial patterns, referred to as features, in the training set. Entrywise product of the kernels with all patches of an image generates the feature map. The feature map is a 2D array having higher response at entries where the kernel and the patch represent similar spatial patterns. CNNs detect features in an input image using this technique. Multiple kernels are utilized to detect multiple features, and feature detection occurs across each frontal slice of an image $\mathcal{X}$.

III. Numerical Results

All experiments presented in this paper were carried out using Tensorflow [1]. The results are presented as three separate experiments with different types of color images.

1) Experiment A: Classifying pictures of 5 different bird species with approximately 640 images in the training set, 80 images in the validation set, and 80 images in the testing set[1]. We use the Adagrad algorithm [12] which works well for this small dataset to compute the gradient and minimize the loss.
2) Experiment B: Classifying pictures of cats, dogs, and wild animals with approximately 12,000 images in the training set, 1500 images in the validation set, and 1500

[1]https://www.kaggle.com/datasets/gpiosenka/100-bird-species

images in the testing set[2]. For the optimization We used the Adam algorithm [10] which performs better for larger datasets than Adagrad.

3) Experiment C: Classifying pictures of 6 different landscapes with approximately 10,080 images in the training set, 1260 images in the validation set, and 1260 images in the testing set[3]. We use the Adam [10] optimization algorithm in the CNN.

We partition the images among the training, validation, and testing sets in an 80-10-10 ratio, respectively, and initialize the trainable parameters of the CNN by values drawn from the Glorot uniform distribution [6]. Validation sets are employed during training to evaluate CNN performance and thereby assist in hyperparameter tuning. In our experiments, we build CNNs by stacking three pairs of 2D convolutional and max-pooling layers, followed by a dense layer and an output layer with the Softmax activation function. We use ReLU activation in all the hidden layers.

For each dataset, we train three separate CNNs: one using the original (or ground truth) data, a second CNN trained on corrupted images, and a third on TNN-ADMM reconstructed images. Some examples of the ground truth, corrupted, and reconstructed images are shown in Figure 1. Subsequently, we record the accuracy of the CNNs in classifying ground truth, corrupted, and reconstructed testing sets. The accuracy is defined to be the ratio of correct predicted labels to the total predictions over a set of images. We show that the CNNs trained on reconstructed images exhibit higher accuracy compared to those trained on corrupted images, particularly when evaluated against ground truth and reconstructed testing sets.

A. Workflow for each experiment

We first train a CNN using the ground truth training set. During this phase, the network's hyperparameters, such as the number of training iterations, number of kernels, etc., are tuned to maximize the accuracy with respect to ground truth in the validation set. Note that no further changes to the network's hyperparameters are made beyond this stage. The CNN trained on the ground truth training set is then saved (referred to as Ground truth CNN).

We generate a corrupted version of the ground truth dataset by applying a low-rank approximation, followed by decimation and addition of noise to each image sample. Note that decimation is carried out independently across each color channel. For the "corrupted dataset" we decimate 50% of the data and add Gaussian noise with a signal-to-noise ratio (SNR) of 10:2. The CNN is trained using the corrupted training set, and the trained CNN is saved (referred to as Corrupted CNN). Reconstruction of each image in the corrupted dataset is carried out by the TNN-ADMM algorithm. The network trained using the reconstructed images is referred to as Reconstructed CNN.

[2]https://www.kaggle.com/datasets/andrewmvd/animal-faces

[3]https://www.kaggle.com/datasets/puneet6060/intel-image-classification

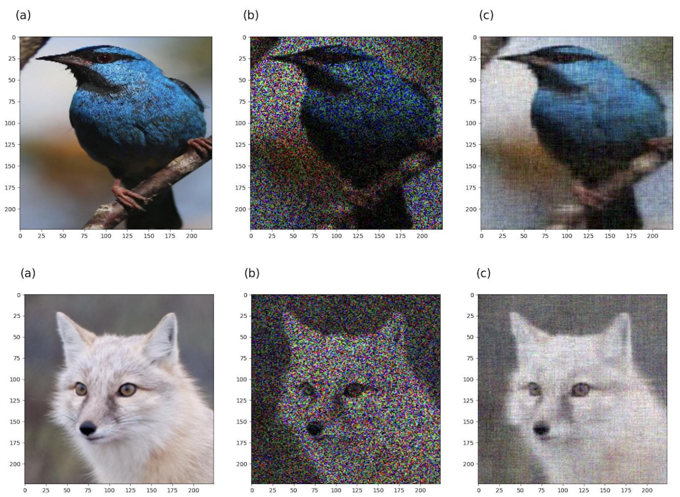

Fig. 1: From left to right, sample images from (a) ground truth, (b) corrupted, and (c) reconstructed images respectively. Top: a bird in Experiment A and bottom: a fox in Experiment B.

B. Testing results

Each of the 3 saved CNNs just described is loaded and tested on ground truth, corrupted, and reconstructed testing sets. We document the accuracy of each CNN across three different testing sets in Tables I, II, and III, based on datasets used in Experiments A, B, and C, respectively. In each table, the training set is designated by the row label, while the testing set utilized is indicated by the column label. We observe the following:

- All three tables (Tables I-III) consistently illustrate a maximum accuracy in each row along the diagonal of the table, which implies that the CNN performs best when evaluated on the same type of data on which it was trained.
- CNNs trained and tested on reconstructed images perform better than CNNs trained and tested on corrupted images.
- CNNs trained on uncorrupted or reconstructed images perform poorly when tested on corrupted images and ones trained on corrupted images perform poorly when tested on uncorrupted or reconstructed images.

More importantly, a consistent trend emerges: CNNs trained on reconstructed images consistently yield higher accuracy in classifying ground truth and reconstructed images compared to those trained on corrupted images.

	Ground truth	Corrupted	Reconstructed
Ground truth CNN	0.96	0.36	0.88
Corrupted CNN	0.56	0.80	0.56
Reconstructed CNN	0.80	0.52	0.88

TABLE I: Testing accuracy for Experiment A (birds). CNNs are trained/tested on the ground truth, corrupted, and reconstructed training/testing sets. The row labels denote the training set used and the column labels denote the testing set.

	Ground truth	Corrupted	Reconstructed
Ground truth CNN	0.96	0.37	0.80
Corrupted CNN	0.86	0.90	0.82
Mixture CNN	0.90	0.88	0.89
Reconstructed CNN	0.93	0.46	0.93

TABLE II: Testing accuracy for Experiment B (cats, dogs and wild animals). In Table II the row labeled "Mixture CNN" indicates a training set made up of 50% clean and 50% corrupted samples.

	Ground truth	Corrupted	Reconstructed
Ground truth CNN	0.81	0.18	0.44
Corrupted CNN	0.53	0.70	0.50
Reconstructed CNN	0.69	0.34	0.75

TABLE III: Testing accuracy of Experiment C (landscapes).

Specifically in Table I it is evident that the Corrupted CNN exhibits lower accuracy on both the ground truth and the reconstructed testing sets, whereas the Reconstructed CNN exhibits higher accuracy. There is an increase in accuracy of 24% when using the Reconstructed CNN on the ground truth testing set and a 32% increase on the reconstructed testing set. In Table II, we see that the Reconstructed CNN outperforms Corrupted CNN on both ground truth and reconstructed testing sets with an increase in accuracy of 7% and 11% respectively. Furthermore, the Mixture CNN yields a 4% improvement in accuracy compared to the Corrupted CNN when tested against the ground truth. The Reconstructed CNN still outperforms the Mixture CNN by 3%. Unsurprisingly, training data with lower levels of corruption show less difference between the Corrupted and Reconstructed CNNs. Table III shows that the Reconstructed CNN is more accurate than the Corrupted CNN on both ground truth and reconstructed testing sets with an increase in accuracy of 16% and 25% respectively.

IV. Discussion

In this work, we demonstrate that CNNs trained on corrupted images consistently exhibit low accuracy when classifying ground truth and reconstructed images. This observation suggests that CNNs struggle to learn the inherent features present in uncorrupted images from corrupted training sets. On the other hand, CNNs trained on images reconstructed using the TNN-ADMM algorithm consistently show higher accuracy for both ground truth and reconstructed testing sets. This finding indicates that the TNN-ADMM algorithm can effectively restore essential features that are representative of the ground truth dataset.

In the case of images with very simple features, such as the MNIST handwritten digits dataset, no significant degradation in accuracy was observed for CNNs trained on corrupted images and tested on uncorrupted images. One possible explanation is that noise and decimation-induced corruption may not considerably affect the boundaries between different colors in these cases. Consequently, the CNNs are still capable of distinguishing between various boundaries and making correct predictions. Investigating the underlying reasons for this behavior would be of great interest.

Furthermore, CNNs are widely applied in classification problems involving tensors of higher dimensions, such as seismic data tensors. The TNN-ADMM algorithm can be readily extended to these higher dimensions and has proven to be particularly effective and efficient in reconstructing low-rank tensors. Many real-world data tensors are naturally of higher dimension, are often corrupted, and are of low rank. It would be worthwhile to explore the potential enhancement in CNN accuracy that arises from employing higher order tensors, reconstructed using the TNN-ADMM algorithm, as training sets.

V. Conclusion

We demonstrate in this paper that TNN-ADMM is a viable image reconstruction procedure that produces better-quality training sets for CNNs in scenarios where high-quality datasets are difficult to obtain. Furthermore, the proposed combination of TNN-ADMM and CNNs offers an effective workflow to deal with corrupted datasets in classification tasks.

Acknowledgments

This research is partially supported by the sponsors of the UT Dallas "3D+4D Seismic FWI" research consortium. YL is partially supported by NSF CAREER DMS-1846690.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, and et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2015.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn., 3(1):1–122, 2011.

[3] E. Candes and B. Recht. Exact matrix completion via convex optimization. Communications of the ACM, 55(6):111–119, 2012.

[4] S. F. Dodge and L. J. Karam. Understanding how image quality affects deep neural networks. CoRR, abs/1604.04004, 2016.

[5] G. Ely, S. Aeron, N. Hao, and M. E. Kilmer. 5d seismic data completion and denoising using a novel class of tensor decompositions. Geophysics, 80(4):V83–V95, 2015.

[6] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[7] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[8] G. James, D. Witten, T. Hastie, R. Tibshirani, et al. An introduction to statistical learning, volume 112. Springer, 2013.

[9] M. E. Kilmer and C. D. Martin. Factorization strategies for third-order tensors. Linear Algebra and its Applications, 435(3):641–658, 2011.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[11] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10):1995, 1995.

[12] A. Lydia and S. Francis. Adagrad—an optimizer for stochastic gradient descent. Int. J. Inf. Comput. Sci, 6(5):566–568, 2019.

[13] J. Popa, Y. Lou, and S. E. Minkoff. Low-rank tensor data reconstruction and denoising via admm: Algorithm and convergence analysis. J. Sci. Comput., 97(2):49, 2023.

[14] J. Popa, S. E. Minkoff, and Y. Lou. Improving seismic data completion via low-rank tensor optimization. In SEG International Exposition and Annual Meeting, page D041S075R003, 2020.

[15] J. Popa, S. E. Minkoff, and Y. Lou. An improved seismic data completion algorithm using low-rank tensor optimization: Cost reduction and optimal data orientation. Geophysics, 86(3):V219–V232, 2021.

[16] J. Popa, S. E. Minkoff, and Y. Lou. Tensor-based reconstruction applied to regularized time-lapse data. Geophys. J. Int., 231(1):638–649, 2022.