Extended convolutional layers

Ali Siahkoohi, Mathias Louboutin, and Felix J. Herrmann
School of Computational Science and Engineering,
Georgia Institute of Technology

Summary

We consider extending convolutional layers in convolutional neural networks (CNNs) to convolutions with auxiliary variables. These auxiliary variables amount to non-stationary convolutional weights. As a result, this type of convolutional layers, referred to as extended convolutions, have two sets of variables: primary and auxiliary. The primary variables are the typical weights and biases of conventional convolutional layers in commonly used in CNNs. The auxiliary variables provide an independent convolutional weight for each pixel in the image or in the intermediate feature spaces. The forward propagation of the network involves computing non-stationary convolutions via the auxiliary weights. To train the extended convolutional layers we propose a variable projection scheme where we first solve for auxiliary variables by minimizing a loss function consisting of a conventional loss functions and a penalty term enforcing the non-stationary weights to be close to the primary variable. Next, we take a gradient step for the primary variable by minimizing the penalty term with respect to the primary variable. Through numerical examples applied to a toy neural network with four extended convolutional layers we observe that this extension broadens the valley of global minimizer. This could potentially enhance the ability of networks to generalize and provide opportunity for faster training of network with second order optimization methods. Code to reproduce our results are made available on GitHub.

Method

Here we first describe conventional convolutional layers in a mathematical framework. Next, we introduce our extended convolutional layers. Finally we introduce our variable-projection based approach for training extended convolutional layers.

Conventional convolutional layers

Consider a conventional convolutional layer denoted by $\mathbf{y} = \text{conv}(\mathbf{x}; \mathbf{w})$ where $\mathbf{x} \in \mathbb{R}^n$ is the input, $\mathbf{w} \in \mathbb{R}^{n_w}$ the convolutional weights of size n_w, and $\mathbf{y} \in \mathbb{R}^m$ is the convolution output. For simplicity, we are ignoring the bias term and we only concentrate on the convolutional weights. Since this convolution operator is linear, it can be equivalently represented by a matrix-vector product:

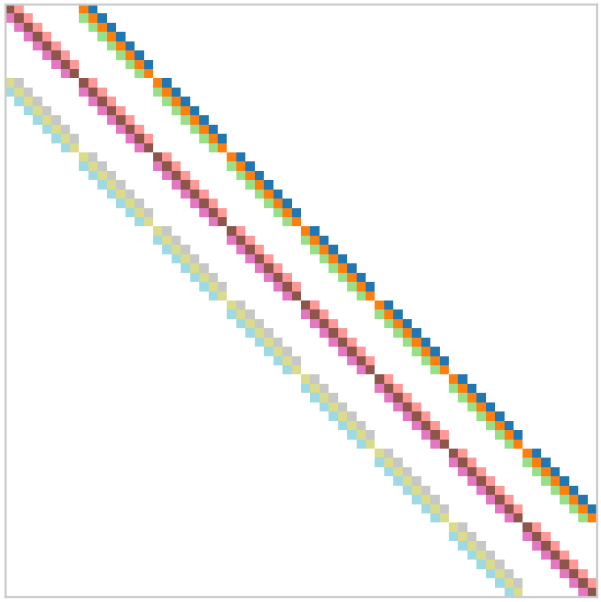
$$\text{conv}(\mathbf{x}; \mathbf{w}) = \mathbf{W}\mathbf{x}. \tag{1}$$

In the above expression, matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ represents the convolution operator. This matrix can be explicitly written as follows:

$$\mathbf{W} = \sum_{i=1}^{n_w} w_i \mathbb{I}\mathbf{T}_{t(i)} \tag{2}$$

where w_i's are elements of $\mathbf{w}$ and $\mathbf{T}_{t(i)}$ is a matrix that is zero everywhere except for the entires on the $t(i)^{\text{th}}$ diagonal. Figure 1a provides an schematic to the weights $\mathbf{w}$ where each color represent a different element w_i, and Figure 1b illustrates the matrix $\mathbf{W}$ with colors correspond to different element w_i of $\mathbf{W}$. Each color in matrix $\mathbf{W}$ corresponds to an individual term of the summation in Equation 2.

(a)



(b)

Figure 1: Construction of a conventional convolutional matrix $\mathbf{W}$ for a 3×3 convolutional layer with one input and output channel, stride one, and no padding. (a) Convolution weight $\mathbf{w}$. (b) Convolutional matrix $\mathbf{W}$ (cf. Equation 2).

Extended convolutional layers

Extended convolution operators are different from conventional convolutional layers in that they have an extra set of variables referred to as auxiliary variables. These auxiliary variables correspond to having a different convolutional weight. Given the definitions in the previous section, there are $n \times n_w$ unknown in the auxiliary variables. We denote these unknowns by $\{\underline{\mathbf{w}}_i\}_{i=1}^{n}$, where $\underline{\mathbf{w}}_i \in \mathbb{R}^{n_w}$. Similar to conventional convolution, the extended convolution, denoted by $\underline{\text{conv}}$, can also be shown as a matrix-vector product:

$$\underline{\text{conv}}(\mathbf{x}; \{\underline{\mathbf{w}}_i\}_{i=1}^{n}) = \underline{\mathbf{W}}\mathbf{x}. \tag{3}$$

Matrix $\underline{\mathbf{W}} \in \mathbb{R}^{m \times n}$ can be written as:

$$\underline{\mathbf{W}} = \sum_{i=1}^{n_w} \text{diag}(\underline{\mathbf{w}}_i)\mathbf{T}_{t(i)}. \tag{4}$$

Figure 2 provides a schematic for $\underline{\mathbf{W}}$. Comparing Figures 1b and 2 summarizes the difference between the two convolutional layers.

Figure 2: Extended convolutional matrix.

Training extended convolutional layers

The overall training loss function for extended convolutional layers is brought below:

$$\min_{\{w_i, \underline{\mathbf{w}}_i\}_{i=1}^{n_w}} \mathcal{L}(\underline{\mathbf{W}}) + \frac{\lambda^2}{2} \left\| \mathbf{W} - \widehat{\mathbf{W}} \right\|_F^2. \tag{5}$$

In the equation above, $\mathcal{L}$ denotes any typical loss function—e.g., ℓ_2 norm for regression, crops-entropy for classification, or any divergence between two probability distributions for training generative models. The penalty term ensure all the non-stationary convolutional weights—i.e., rows in Figure 2, are close to the primary convolutional weight, $\mathbf{w} = \{w_i\}_{i=1}^{n_w}$

We use variable projection [Aravkin et al., 2013, van Leeuwen et al., 2014] to train the extended convolutional layers. That is to say we first minimize the loss function in Equation 5 with respect to $\{\underline{\mathbf{w}}_i\}_{i=1}^{n_w}$:

$$\{\widehat{\underline{\mathbf{w}}}_i\}_{i=1}^{n_w} := \arg, \min_{\{\underline{\mathbf{w}}_i\}_{i=1}^{n_w}} \mathcal{L}(\underline{\mathbf{W}}) + \frac{\lambda^2}{2} \left\| \mathbf{W} - \widehat{\mathbf{W}} \right\|_F^2. \tag{6}$$

Next, we take a gradient step for $\{w_i\}_{i=1}^{n_w}$ via the following loss function:

$$\min_{\{w_i\}_{i=1}^{n_w}} \left\| \mathbf{W} - \widehat{\mathbf{W}} \right\|_F^2, \tag{7}$$

where $\widehat{\mathbf{W}}$ is the extended convolutional matrix (cf. Equation 4) constructed via to $\{\widehat{\underline{\mathbf{w}}}_i\}_{i=1}^{n_w}$ obtained in Equation 6.

Numerical example

We showcase the effect of the additional auxiliary variables in the extended convolutional layers on the loss landscape via a toy example.

We construct a four-layer CNN with conventional and extended convolutional layers. All the layers have kernel size 3×3, one input and output channel, each with stride one and no padding. We use the sigmoid function after each (extended) convolutional layer. We denote the network with conventional convolutional layers as $G(\,\cdot\,;\mathbf{w})$ and the network with extended convolutional layers as $G^{\text{ext}}(\,\cdot\,;\mathbf{w},\{\underline{\mathbf{w}}_i\}_{i=1}^{n})$. Note that here for notational simplicity $\mathbf{w}$ and $\{\underline{\mathbf{w}}_i\}_{i=1}^{n}$ denote all the primary and auxiliary weights for the four layers combined, respectively. We simulate a training pair by first creating a random fixed input $\mathbf{x} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$. Next, we pass it thorough the conventional network, initialized by an artificial global minimum $\mathbf{w}^*$ to obtain a artificial output pair:

$$\mathbf{y} = G(\mathbf{z}; \mathbf{w}^*). \tag{8}$$

We visualize the loss landscape around $\mathbf{w}^*$ via the approach described in Li et al. [2018]. In this approach, we randomly draw two random vectors, $\boldsymbol{\delta}$ and $\boldsymbol{\eta}$, with the same size as $\mathbf{w}^*$ where the filters in each layer are normalized to have the same mean and standard deviation as the corresponding filter in $\mathbf{w}^*$. For more details we refer interested readers to Li et al. [2018].

To visualize the loss landscape for the the network with conventional convolutional layers, we compute the following loss function and we make a 2D visualization of the obtained values as a function of α and β,

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\left\| \mathbf{y} - G(\mathbf{x}, ; \mathbf{w}) \right\|_2^2 \quad \mathbf{w} = \mathbf{w}^* + \alpha\boldsymbol{\delta} + \beta\boldsymbol{\eta}. \tag{9}$$

Throughout all examples we choose $\alpha \in [-4, 4]$ and $\beta \in [-4, 4]$. We illustrate the above loss landscape in the top row of Figure 3. Each Figure on the top row is indicating the same landscape, but visualized with different random vectors $\boldsymbol{\delta}$ and $\boldsymbol{\eta}$. The dark red star indicates the location of $\mathbf{w}^*$.

Visualizing the loss landscape of the network with extended convolutional layers amounts to solving the optimization problem 6 for all primary variables $\mathbf{w} = \mathbf{w}^* + \alpha\boldsymbol{\delta} + \beta\boldsymbol{\eta}$ and visualizing the value of that objective at the minimizer. This quantity has formulated below:

$$\mathcal{L}^{\text{ext}}(\mathbf{w}) = \min_{\{\underline{\mathbf{w}}_i\}_{i=1}^{n_w}} \left\| \mathbf{y} - G^{\text{ext}}(\mathbf{x}; \mathbf{w}, \{\underline{\mathbf{w}}_i\}_{i=1}^{n}) \right\|_2^2 + \frac{\lambda^2}{2}\left\| \mathbf{W} - \widehat{\mathbf{W}} \right\|_F^2, \quad \mathbf{w} = \mathbf{w}^* + \alpha\boldsymbol{\delta} + \beta\boldsymbol{\eta}. \tag{10}$$

The second row of Figure 3 indicates these loss landscapes. Each column in Figure 3 compares the loss landscape for networks with conventional (top) and extended (bottom) convolutional layers. We consistently observe that utilizing extended convolutional layers is widening the global minima valley. This is of great importance in the context of deep networks and their ability to generalize [Chaudhari et al., 2019, Foret et al., 2020].

We also demonstrate the enhanced ability of extended convolutions to fit the data and bring down the loss function by plotting normalized loss functions and optimization trajectories of the networks with conventional and extended convolutions layers when initialized with different weights. To achieve this, we solve objectives 9

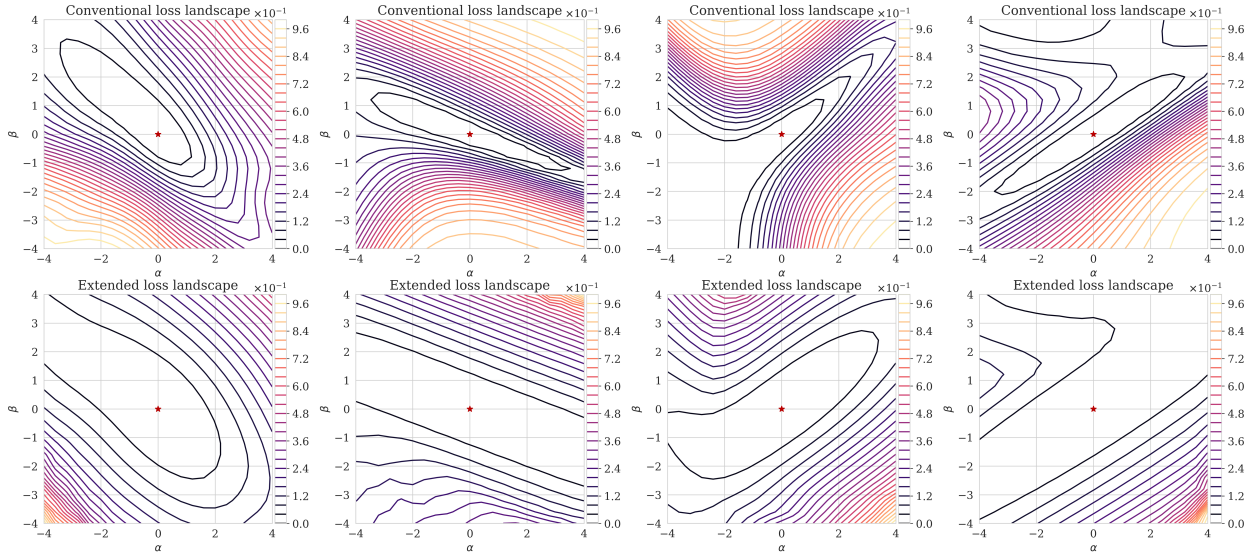


Figure 3: Normalized loss landscape. Dark red star indicates the location of $\mathbf{w}^*$. Top rows show the values obtain via Equation 9 for different random vectors $\boldsymbol{\delta}$ and $\boldsymbol{\eta}$. Bottom rows are visualizing the network with extended convolutions with the same set of random vectors as the top row.

and 10 for three initial primary weights corresponding to $(\alpha, \beta) = (-2, 2),\ (2, 2),\ (2, -2)$. We always initial auxiliary weights randomly draw from standard normal distribution—i.e., $\{\underline{\mathbf{w}}_i\}_{i=1}^{n_w} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$. In each figure, we solve the optimization problems 9 and 10 with Adam [Kingma and Ba, 2014] and L-BFGS [Liu and Nocedal, 1989] optimization algorithms for 200 iterations.

Figures 4a – 4c show the loss function during optimization of the conventional and extended networks for starting primary weights corresponding to $(\alpha, \beta) = (-2, 2),\ (2, 2),\ (2, -2)$, respectively.

We make the following observations. The extended objective (cf. Equation 10) solved with L-BFGS algorithm brings down the residual more the all the other alternatives for most of the initial primary weights and iterations. When solved with Adam, the extended objective often brings down the loss more than the conventional objective (Equation 9).

References

Aleksandr Y. Aravkin, Tristan van Leeuwen, and Ning Tu. Sparse seismic imaging using variable projection. In *ICASSP*, 05 2013. URL https://slim.gatech.edu/Publications/Public/Conferences/ICASSP/2013/aravkin2013ICASSPssi/aravkin2013ICASSPssi.pdf.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-Aware Minimization for Efficiently Improving Generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

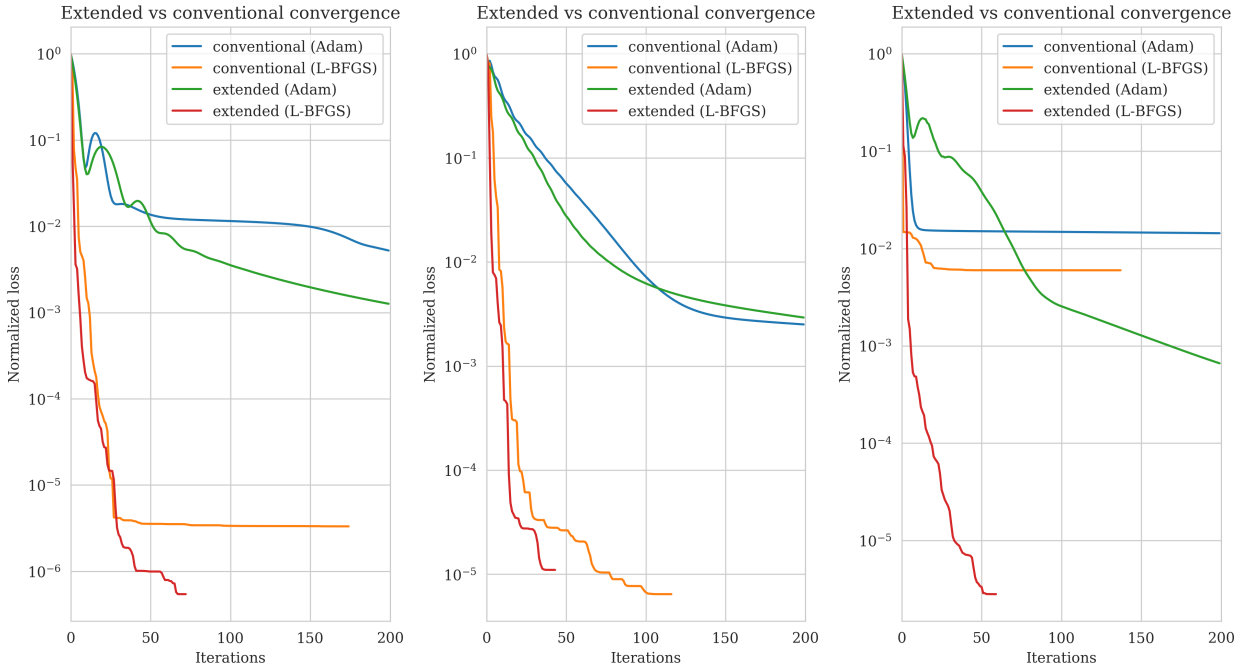


Figure 4: Optimization loss comparison.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems*, 2018.

Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Tristan van Leeuwen, Aleksandr Y. Aravkin, and Felix J. Herrmann. Comment on: "application of the variable projection scheme for frequency-domain full-waveform inversion" (m. li, j. rickett, and a. abubakar, geophysics, 78, no. 6, r249–r257). *Geophysics*, 79(3):X11–X17, 05 2014. doi: 10.1190/geo2013-0466.1. URL https://slim.gatech.edu/Publications/Public/Journals/Geophysics/2014/vanLeeuwen2014GEOPcav/vanLeeuwen2014GEOPcav.pdf. (discussion by Tristan van Leeuwen, Aleksandr Y. Aravkin, and Felix J. Herrmann).