Parallel Backpropagation for Inverse of a Convolution with Application to Normalizing Flows: Supplementary Materials

We provide a comprehensive extension to the main paper, offering in-depth insights into the experimental setup, additional experimental results, and rigorous mathematical proofs. The Supplementary begins with experimental specifications Section 7, including information about model architecture, training parameters, and hardware used. In the next section 8, we present an interesting application of inverse convolution layers in image classification, demonstrating high accuracy on the MNIST dataset with a remarkably small model. Section 9 presents thorough proofs of two theorems related to the backpropagation algorithm for the inverse of convolution layers. These proofs, presented with clear mathematical notation and step-by-step derivations, establish a theoretical foundation for computing input gradients and weight gradients in the context of inverse convolution operations.

7 Experimental Details

The architecture of SNF is the starting point for Inverse-Flow architecture and all our experiments. All models are trained using the Adam optimizer. We evaluate our Inverse-Flow model for density estimation (BPD, NLL), Sampling time (ST), and Forward time (FT) with a batch size of 100 for all experiments. For MNIST, we use an initial learning rate of $1e-3$, scheduled to decrease by one order of magnitude after 50 epochs for all datasets but CIFAR10, which is decreased every 25 epochs. All the experiments were run on NVIDIA GeForce RTX 2080 Ti GPU. For MaCow, SNF, MaCow, and SNF, we use the official code released by the authors. Emerging was implemented in PyTorch by the authors of SNF, we make use of that. We have implemented CInC Flow on PyTorch to get the results.

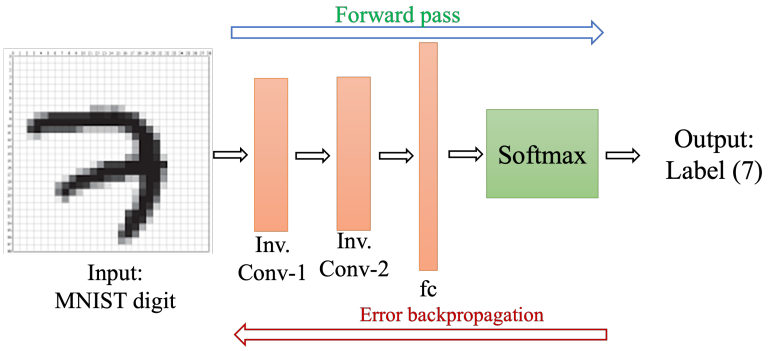

Figure 4: Overview of a small image classification model with two inverse of convolution (3×3 inv-con) layers with 97.6% accuracy on MNIST dataset.

8 Image Classification using Inverse of Convolution Layers:

For MNIST digits image classification using the inverse of convolution (inv-conv) layers and proposed its backpropagation algorithm, we trained a two inv-conv layer and one fully connected layer model with 16 learnable parameters for inv-conv layers. See Figure 4; this simple and small two inv-conv and one fully connected (FC) layers model gives 97.6% classification accuracy after training for 50 epochs.

9 Detailed Proofs

In this section, we provide the proofs relating to the proposed backpropagation algorithm for the inverse of the convolution layer. First, we provide the following notation and the equation for the gradients.

Notation: We will follow the notation used in the main paper.
We will denote input to the inverse of convolution (inv-conv) by $y \in \mathbb{R}^{m^2}$ and output to be $x \in \mathbb{R}^{m^2}$. We will be indexing x, y using $p = (p_1, p_2) \in \{1, \cdots, n\} \times \{1, \cdots, n\}$. We define

$$\Delta(p) = \{(i, j) : 0 \leq p_1 - i, p_2 - j < k\} \setminus \{p\}.$$

$\Delta(p)$ informally is set of all pixels except p which depend on p, when convolution is applied with top, left padding. We also define a partial ordering $\leq$ on pixels as follows

$$p \leq q \quad \Leftrightarrow \quad p_1 \leq q_1 \text{ and } p_2 \leq q_2.$$

The kernel of $k \times k$ convolution is given by matrix $W \in \mathbb{R}^{k \times k}$. For the backpropagation algorithm for inv-conv, the input is

$$x \in \mathbb{R}^{m^2} \text{ and } \frac{\partial L}{\partial x} \in \mathbb{R}^{m^2},$$

where L is the loss function. We can compute y on $O(mk^2)$ time using the parallel forward pass algorithm Aaditya et. al. The output of backpropagation algorithm is

$$\frac{\partial L}{\partial y} \in \mathbb{R}^{m^2} \text{ and } \frac{\partial L}{\partial W} \in \mathbb{R}^{k^2}$$

which we call input and weight gradient, respectively. We provide the algorithm for computing these in the next 2 subsections.

9.1 Proof of Theorem 1

Computing Input Gradients Since y is input to inv-conv and x is output, $y = \text{conv}_W(x)$ and we get the following m^2 equations by definition of the convolution operation.

$$y_p = xp + \sum_{q \in \Delta(p)} W_{(k,k)-p+q} \cdot x_q \tag{7}$$

Using chain rule of differentiation, we get that

$$\frac{\partial L}{\partial y_p} = \sum_q \frac{\partial L}{\partial x_q} \times \frac{\partial x_q}{\partial y_p}. \tag{8}$$

Hence if we find $\frac{\partial x_q}{\partial y_p}$ for every pixels p, q, we can compute $\frac{\partial L}{\partial y_p}$ for every pixel p.

Theorem 1: Input y gradients

$$\frac{\partial x_q}{\partial y_p} = \begin{cases} 1 & \text{if } p = 1 \\ 0 & \text{if } q \not\leq p \\ -\sum_{r \in \Delta(p)} W_{(k,k)-r} \frac{\partial x_{p-r'}}{\partial y_p} & \text{otherwise.} \end{cases} \tag{9}$$

Proof. We will prove Theorem 1 by induction on the partial ordering of pixels.

Base Case: For $p = (1, 1)$, which is the smallest element in our partial ordering:

From Equation (7), we have: $y_{(1,1)} = x_{(1,1)}$. This implies: $\frac{\partial x_{(1,1)}}{\partial y_{(1,1)}} = 1$ and for any $q \neq (1, 1)$: $\frac{\partial x_q}{\partial y_{(1,1)}} = 0$. This satisfies the theorem for the base case.

Inductive Step: Assume the theorem holds for all pixels less than p in the partial ordering. We will prove it holds for p.

1. For $q \not\leq p$, x_q does not depend on y_p due to the structure of the convolution operation. Therefore, $\frac{\partial x_q}{\partial y_p} = 0$.

2. For $q \leq p$, we differentiate Equation (7) with respect to y_p:

$$\frac{\partial x_p}{\partial y_p} = \frac{\partial x_p}{\partial y_p} + \sum_{r \in \Delta(p)} W_{(k,k)-p+r} \cdot \frac{\partial x_r}{\partial y_p}$$

$$1 = \frac{\partial x_p}{\partial y_p} + \sum_{r \in \Delta(p)} W_{(k,k)-p+r} \cdot \frac{\partial x_r}{\partial y_p} \tag{10}$$

Rearranging 10:

$$\frac{\partial x_p}{\partial y_p} = 1 - \sum_{r \in \Delta(p)} W_{(k,k)-p+r} \cdot \frac{\partial x_r}{\partial y_p} \tag{11}$$

This is equivalent to the third case in the theorem, with $q = p$.

3. For $q < p$, we can write:

$$x_q = y_q - \sum_{r \in \Delta(q)} W_{(k,k)-q+r} \cdot x_r$$

Differentiating with respect to y_p:

$$\frac{\partial x_q}{\partial y_p} = \frac{\partial y_q}{\partial y_p} - \sum_{r \in \Delta(q)} W_{(k,k)-q+r} \cdot \frac{\partial x_r}{\partial y_p}$$

Since $q < p$, $\frac{\partial y_q}{\partial y_p} = 0$. Therefore:

$$\frac{\partial x_q}{\partial y_p} = - \sum_{r \in \Delta(q)} W_{(k,k)-q+r} \cdot \frac{\partial x_r}{\partial y_p} \tag{12}$$

This is equivalent to the third case in the theorem.

Thus, by induction, the theorem holds for all pixels p. □

9.2 Proof of Theorem 2

Computing Weight Gradients From Equation 7, we can say computing gradient of loss L with respect to weights W involves two key factors. Direct influence: how a specific weight W in convolution kernel directly affects output x pixels, and Recursive Influence: how neighboring pixels, weighted by kernel, indirectly influence output x during inverse of convolution operation. Similarly, to compute gradient of loss L w.r.t filter weights W, we apply chain rule:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial x} \times \frac{\partial x}{\partial W} \tag{13}$$

where: $\frac{\partial L}{\partial x}$ is gradient of loss with respect to output x and inverse of convolution operation is applied between $\frac{\partial L}{\partial x}$ and output x. Computing the gradient of loss L with respect to convolution filter weights W is important in backpropagation when updating the convolution kernel during training. Similarly, $\partial L/\partial W$ can be calculated as Equation 13 and $\partial x/\partial W$ can be calculated as (Equation 13) for each $k_{i,j}$ parameter by differentiating Equation 7 w.r.t W:

$$\frac{\partial L}{\partial W_a} = \sum \frac{\partial L}{\partial x_q} \times \frac{\partial x_q}{\partial W_a} \tag{14}$$

Equation 14 states that to compute the gradient of the loss with respect to each weight W_a, we need to:

- Compute how loss L changes with respect to each output pixel x_q (denoted by $\frac{\partial L}{\partial x_q}$).
- Multiply this by gradient of each output pixel x_q with respect to weight W_a (denoted by $\frac{\partial x_q}{\partial W_a}$)

We then sum over all output pixels x_q.

Theorem 2: Weights W gradients

$$\frac{\partial x_q}{\partial W_a} = \begin{cases} 0 & \text{if } q \leq a \\ -\sum_{q' \in \Delta q(a)} W_{q'-a} \times \frac{\partial x_{q-q'}}{\partial W_a} - x_{q-a} & \text{if } q > a \end{cases} \tag{15}$$

Proof. We will prove Theorem 2 by induction on the partial ordering of pixels.

Base Case:

For $q \leq a$, we have $\frac{\partial x_q}{\partial W_a} = 0$.

This is because in the inverse of convolution operation, x_q does not directly depend on W_a. The weight W_a only affects pixels that come after q in the partial ordering.

Inductive Step: Assume the theorem holds for all pixels less than q in the partial ordering. We will prove it holds for $q > a$.

From Equation (7), we have:

$$y_q = x_q + \sum_{r \in \Delta(q)} W_{(k,k)-q+r} \cdot x_r \tag{16}$$

Rearranging this equation 16:

$$x_q = y_q - \sum_{r \in \Delta(q)} W_{(k,k)-q+r} \cdot x_r \tag{17}$$

Now, let's differentiate both sides of 17with respect to W_a:

$$\frac{\partial x_q}{\partial W_a} = \frac{\partial y_q}{\partial W_a} - \sum_{r \in \Delta(q)} \left(\frac{\partial W_{(k,k)-q+r}}{\partial W_a} \cdot x_r + W_{(k,k)-q+r} \cdot \frac{\partial x_r}{\partial W_a} \right) \tag{18}$$

Note that $\frac{\partial y_q}{\partial W_a} = 0$ because y is the input to the inverse convolution and doesn't depend on W.

Also, $\frac{\partial W_{(k,k)-q+r}}{\partial W_a} = 1$ if $(k,k) - q + r = a$, and 0 otherwise.

Let $\Delta q(a) = \{r \in \Delta(q) : (k,k) - q + r = a\}$. Then we can rewrite the equation 18 as 19:

$$\frac{\partial x_q}{\partial W_a} = -\sum_{r \in \Delta q(a)} x_r - \sum_{r \in \Delta(q)} W_{(k,k)-q+r} \cdot \frac{\partial x_r}{\partial W_a} \tag{19}$$

The first sum simplifies to $-x_{q-a}$ because $r = q - (k,k) + a$ for $r \in \Delta q(a)$.

In the second sum, we can use the inductive hypothesis for $\frac{\partial x_r}{\partial W_a}$ because $r < q$.

Therefore:

$$\frac{\partial x_q}{\partial W_a} = -x_{q-a} - \sum_{r \in \Delta(q)} W_{(k,k)-q+r} \cdot \frac{\partial x_r}{\partial W_a} \tag{20}$$

The right side of 20 is equivalent to the second case in the theorem.

Thus, by induction, the theorem holds for all pixels q. □