We thank all the reviewers for providing constructive feedback. First, we hope to bring a new theoretical result to the reviewers' attention. We can now prove that **Algorithm 1 is guaranteed to converge locally** whenever $0 < \alpha < 2$. We will include this new result in the revision.

To Reviewer #1

Q1: *Difference from emerging convolutions.*

We thank R1 for pointing out a missing reference. We developed our approach without knowing the work of emerging convolutions. We will include a discussion of it in Section 4.3. We believe that our approach has *critical advantages over emerging convolutions*, which can be detailed as follows:

- **The inversion is much faster to compute.** Emerging convolutions are inverted using forward/back substitutions designed for inverting triangular matrices, which requires the same number of iterations as the input dimension. In stark contrast, we use a fixed-point iteration method (Algorithm 1) for inversion, which is similar to i-ResNet and requires substantially fewer iterations than the input dimension. For example, our inversion takes 120 iterations to converge on CIFAR-10, while emerging convolutions will need 3072 iterations. In other words, our inversion is roughly 25 times faster than emerging convolutions on powerful GPUs.
- **No bottleneck in the architecture.** The input and output of an emerging convolution must have the same dimensions, and therefore emerging convolutions cannot be used to expand the hidden dimensions of a neural network. As a result, if a flow model is built using only emerging convolutions, the largest number of hidden units for a layer will be the input dimension, which becomes a bottleneck in expressive power. In stark contrast, we can use masked convolutions to expand the hidden dimensions (See Eq. (2)) and eliminate the bottleneck. This is why we can build MintNet *using only masked convolutions*, while the emerging convolution paper modified the Glow architecture by replacing 1×1 convolutions with emerging convolutions.

Q2: *Strengths over i-ResNet, ResNet, Flow++, etc.*

We would like to remind the reviewer that we use Section 4.3 to compare MintNet with many popular paradigms of flow models, *e.g.*, free-form invertible models (including i-ResNet) and dimension partitioning models (including Flow++). We will clarify Section 4.3 more to incorporate the following information.

Unlike ResNet, MintNet is an invertible architecture that can be used directly as a flow generative model for density estimation and sample generation. Unlike i-ResNet, MintNet provides exact likelihood values and better empirical performance (see Table 1), while having a comparable cost of sample generation. Unlike Flow++, MintNet is a free-form invertible model that does not rely on dimension partitioning and coupling layers. We did not compare results with Flow++, as it uses variational dequantization, while all models we compare (Glow, i-ResNet, and other models in Table 1) use uniform dequantization. For the same model, the bpd values resulting from variational dequantization are **lower bounds** to the bpd values of uniform dequantization, and therefore will always appear to be "better" (hence not comparable).

To Reviewer #2

Q1: *Code and experiments on ImageNet* 64×64.

We will put the code and checkpoints online once the paper is accepted. We agree that results on more datasets are helpful. However, our resources are too limited to finish experiments on ImageNet 64×64, and we believe that our state-of-the-art results across all three datasets (MNIST, CIFAR-10, and ImageNet 32×32) are sufficient to demonstrate the advantages of MintNet.

To Reviewer #3

Q1: *The initial choice of* $\mathbf{x}_0$.

When inverting Mint layers, we always choose $\mathbf{x}_0 = \mathbf{z} \odot \frac{1}{\mathbf{t}}$ because we empirically observe $\mathbf{z} = \mathfrak{L}(\mathbf{x}) \approx \mathbf{t} \odot \mathbf{x}$ for deeper layers after training. We will elaborate more on this in Section 4.1.

Q2: *More explanations on Paired Mint layers*.

A Paired Mint layer is a lower triangular Mint layer followed by an upper triangular Mint layer. The results of the lower triangular layer are fed to the upper one as the input. There are clearer descriptions on Paired Mint layers in the network architecture tables in Appendix D (see, *e.g.*, Table 3). We will clarify this more in Section 4.2.