Table 3: Key differences between Gradient Inversion and Feature Inversion (Ours)

| | **Gradient Inversion** [50, 57, 60, 91, 98] | **Feature Inversion (Ours)** |
|---|---|---|
| **Scenario** | Federated learning (***training***) | Split computing (***inference***) |
| **Input** | Weights + Gradients | Weights + Features |
| **Goal** | Exact image recovery from ***gradients*** | Exact image recovery from ***features*** |
| **Insight** | Gradients encode private information via inversion | Features encode private information via *even faster* inversion |

# A Additional Related Work

**Autoencoder.** Model inversion is also related to autoencoders [44, 84] given a similar functionality of target and inversion models, to an encoder-decoder pair. However, the encoder and decoder in an autoencoder are trained jointly with a vast amount of data in an end-to-end manner. In our case, only an individually pre-trained target model is accessible. Aligning with the previous finding of layer-wise training of autoencoder (*e.g.*, deep belief network [9, 43]), we also find layer-wise training beneficial for inversion model training as detailed in Sec. 3.1.

**Invertible Neural Networks.** Invertible neural networks (INNs) [8, 21, 50] are a family of neural networks which is theoretically invertible due to their special architectures and weights. For instance, i-RevNet [46] uses carefully planned convolution, reshuffling, and partitioning procedures to ensure inversion. i-ResNet [7] constrains weights with Lipschitz condition. In conclusion, INNs are inapplicable to standard model architectures since they depend on uncommon architectures and regularizations.

# B Implementation Details

## B.1 Training Strategy

DCI divides the computation of a feed-forward neural network into multiple parts and inverts the computational flow progressively. One straightforward strategy is to sequentially optimize each individual inversion module $\mathcal{F}_k^{-1}$, beginning with the first layer $\mathcal{F}_1^{-1}$. We observe that as we progress deeper into the model, the inversion error will increase. To address this accumulation issue, we use a more potent training strategy. After optimizing a particular inversion module $\mathcal{F}_k^{-1}$, we take into account all previous inversion modules and fine-tune all layers up to $k$, *i.e.*, $\mathcal{F}_1^{-1} \circ \cdots \circ \mathcal{F}_{k-1}^{-1} \circ \mathcal{F}_k^{-1}$, with the same loss $\mathcal{L}_k$ to reduce the accumulated inversion error. When $k = 1$, this fine-tuning is skipped because there is no inversion model prior to $\mathcal{F}_1^{-1}$.

## B.2 Inversion Models

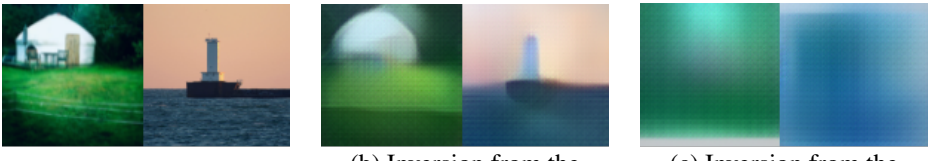
**ResNet-18.** The first four sequential layers in ResNet models are convolution, BN, ReLU, and max pooling layers, and we invert the sequence of the first four layers as a whole block. Each inversion module mimics the architecture of its corresponding target counterpart but with reversed input and output dimensions. For a `BasicBlock` (left) in ResNet-18, its inversion module (right) is shown as below in Fig. 10.

**ResNet-50.** We break the overall ResNet-50 architecture into five sub-networks and invert one sub-network each time. Similar to ResNet-18, the first sub-network is the initial block. The other four sub-networks consist of $3, 4, 6, 3$ repeats of `Bottleneck` respectively [38].

Figure 10: (Left) A `BasicBlock` in ResNet-18 (Right) Corresponding inversion module for the `BasicaBlock`.

# C Additional Inversion Results

**Lossy Final Fully-Connected Layer.** In the main manuscript, we show that the recovered images from RepVGG feature embeddings after 21 convolution layers preserve original semantic and visual attributes. However, we find that information in the feature embeddings decays rapidly through the last two layers in RepVGG. This may indicate that class-invariant information is quickly filtered out towards the end of the model while the initial stages focus on feature extraction, which is aligned with prior observations in transfer learning [22, 57, 62, 92, 100] and self-supervised learning [16, 33, 39]. We visualize the above findings in Fig. 11. One can still recognize the class of inverted images after the penultimate (*i.e.*, the 22-th convolution) layer. However, if we invert features after the final (*i.e.*, the fully-connected) layer, only the predominant color is recognizable.



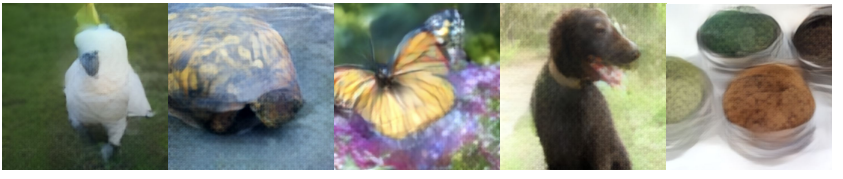
(a) Original images (b) Inversion from the last convolution layer (c) Inversion from the final full-connected layer

Figure 11: Results of inversion from (b) the penultimate layer (*i.e.*, the last convolution layer), and (c) the final layer (*i.e.*, the final full-connected layer)

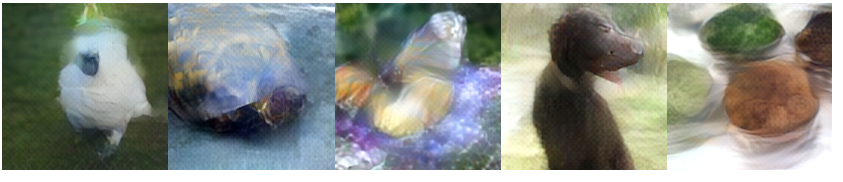
# D Ablation Studies

**Effectiveness of Synthetic Data.** We study how well synthetic data may be used to optimize our inversion modules. For a sanity check, we re-perform the experiment in Fig. 3 using real data from ImageNet training set rather than the synthetic data. We also vary the number of synthetic data used for optimization. In Fig. 12, we show that using more (real or synthetic) data samples improves the inversion quality. For instance, in the first column of Fig. 12, the inverted images can reveal the parrot's tiny eyes when 10K (real or synthetic) images are used for the optimization of inversion modules. However, we also discover that when there are more than 10K (real or synthetic) samples, the improvement of inversion quality is limited. In addition, we demonstrate that DCI achieves a comparable inversion quality when using synthetic data as opposed to using real data. However, due to their sensitivity to data distribution change, as illustrated in Fig. 8, prior generative inversion approaches [69, 94] cannot efficiently use synthetic data.

Real images **x** from the ImageNet validation set.



Inverted images by inversion model trained on 1K real data. (LPIPS ↓ = 0.462)



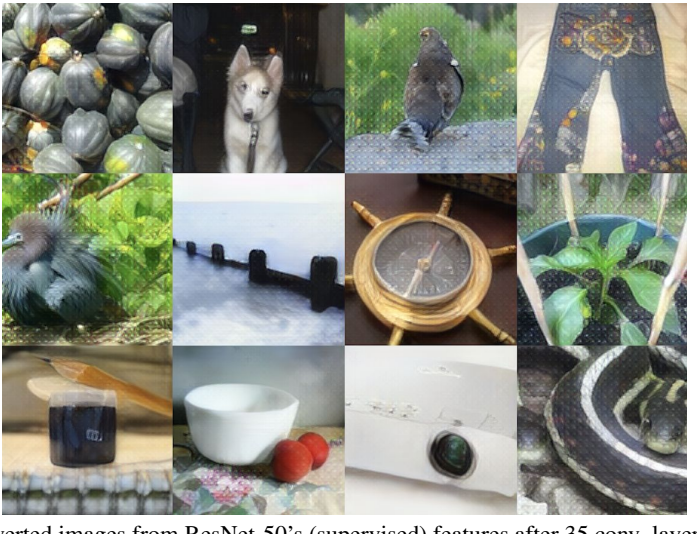
Inverted images by inversion model trained on 1K synthetic data. (LPIPS ↓ = 0.481)

Inverted images by inversion model trained on 10K real data. (LPIPS ↓ = 0.430)

Inverted images by inversion model trained on 10K synthetic data. (LPIPS ↓ = 0.443)

Figure 12: Comparison of different data settings. Our method achieves a comparable inversion quality using synthetic data compared to using real data.

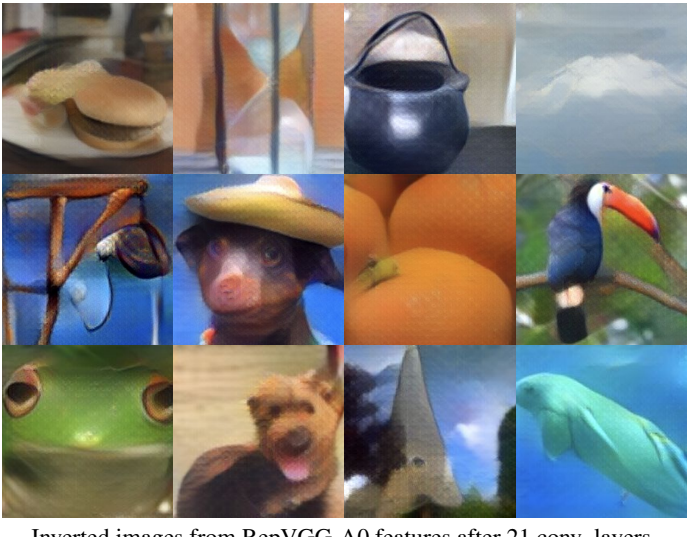
Real images **x** from the ImageNet validation set.



Inverted images from ResNet-50's (supervised) features after 35 conv. layers.

Figure 13: More inversion results on ResNet-50 without any real data. Our method generalizes well on different types of images.

Real images **x** from the ImageNet validation set.



Inverted images from RepVGG-A0 features after 21 conv. layers.

Figure 14: More inversion results on RepVGG-A0 without any real data. Our method generalizes well on different types of images.