Course CSL7590 Assignment 04

Deep Learning

Ashish Kumar(M24AIR003)

Mihir Tomar(M24AIR006)

Shubham Khushwaha (M24AIR011)

April 15, 2025

Contents

List of Tables

List of Figures

Objective

Implement Data-Free Adversarial Distillation.

1 Method

Knowledge distillation is a model compression technique wherein a compact student model is trained to replicate the behavior of a larger, more expressive teacher model. Traditionally, this process requires access to the original training data used to train the teacher model. However, in real-world applications, training data may be proprietary, privacy-sensitive, or unavailable. To address this limitation, Data-Free Knowledge Distillation (DFKD) techniques aim to train student models without access to real data.

A prominent approach within DFKD is Adversarial Knowledge Distillation (AKD), which introduces a generative model that learns to synthesize input data that elicits strong responses from the teacher network. These synthetic images, although not grounded in any specific dataset, are rich enough to carry the semantic cues necessary for the student model to learn from the teacher.

In our work, we implement and analyze a data-free adversarial knowledge distillation framework using PyTorch, focusing on CIFAR-100 classification. We use ResNet-34 as the teacher model and two scaled student models based on ResNet architecture, one with approximately 20% and another with 50% of the teacher's parameters. A carefully constructed deconvolutional generator (GeneratorDeconv) is used to create synthetic data that drives the knowledge transfer process. The student is then trained to match the teacher's outputs on these generated samples using an L2-based loss function. This report details the model architectures, training dynamics, evaluation procedures, and insights derived from our experiments.

Let $T(x)$ be the teacher network trained on a dataset $\mathcal{D}$, and $S(x)$ be the student network that must learn from T without access to $\mathcal{D}$. The central challenge is to simulate a meaningful learning signal using a generator $G(z)$, where $z \sim \mathcal{N}(0, I)$ is a noise vector.

To achieve this, the training pipeline consists of two alternating phases:

1. **Imitation (Student) Stage** – Fix the generator and update the student to match the teacher's output on the generated samples.

2. **Generation (Adversarial) Stage** – Fix the student and update the generator to maximize the disagreement between teacher and student, thus driving generation of informative samples.

This adversarial feedback loop encourages the generator to explore the data space that challenges the student while being semantically aligned with the teacher's learned distribution.

The generator model plays a pivotal role in the AKD framework, as it acts as a proxy for the unavailable real dataset. The design of the generator must therefore enable rich and diverse synthesis of high-dimensional visual data, here tailored to mimic natural images in the CIFAR-100 domain.

The generator, named *GeneratorDeconv*, is a deep deconvolutional neural network (also referred to as a transposed convolutional network). It begins with a dense projection layer that transforms a latent vector $z \in \mathbb{R}^{100}$ into a high-dimensional tensor, which is then upsampled through a sequence of deconvolutional blocks to produce an image of resolution 224×224.

1.1 Network Architecture

- **Latent Vector Projection**
  - Input: $z \in \mathbb{R}^{100}$
  - Layer: $Linear(nz, ngf * 8 * init\_size^2)$, where $init\_size = \lfloor \frac{final\_img\_size}{16} \rfloor$, nz is length of vector z and ngf is a hyperparameter.
  - Activation: ReLU
- **Deconvolutional Stack**
  - **Block 1**: ConvTranspose2d (ngf*8 $\rightarrow$ ngf*4) $\rightarrow$ BatchNorm $\rightarrow$ ReLU
  - **Block 2**: ConvTranspose2d (ngf*4 $\rightarrow$ ngf*2) $\rightarrow$ BatchNorm $\rightarrow$ ReLU
  - **Block 3**: ConvTranspose2d (ngf*2 $\rightarrow$ ngf) $\rightarrow$ BatchNorm $\rightarrow$ ReLU
  - **Block 4**: ConvTranspose2d (ngf $\rightarrow$ nc) $\rightarrow$ Tanh
- **Output**
  - Synthetic image tensor with shape $[B, 3, 224, 224]$
  - Pixel values scaled to $[-1, 1]$ via Tanh activation

1.2 Initialization Scheme

All convolutional and linear layers are initialized using Gaussian initialization:

- $\mathcal{N}(0, 0.02)$ for weights
- Zeros for biases
- BatchNorm weights initialized to 1.0, biases to 0

This follows the standard DCGAN-style initialization to stabilize training and improve convergence.

1.3 Student Models: Scaled ResNet Architectures

To assess the effectiveness of knowledge distillation, we evaluate two student architectures:

1. **20% parameter student**: Compact ResNet-34 variant with about one-fifth of the full model's capacity

2. **50% parameter student**: Intermediate-capacity student to evaluate the scalability of learning

Both models are created by scaling down the number of channels in each convolutional block while retaining the overall topology of ResNet-34, including identity mappings and bottleneck structures.

The forward path remains unchanged:

- Convolutional stem (7×7 kernel)
- Residual blocks with skip connections
- Global average pooling
- Fully connected output with 100 classes

These scaled models offer a balance between capacity and efficiency, ensuring compatibility with CIFAR-100's complexity without overfitting.

1.4 Teacher Model: ResNet-34

The teacher model is a standard ResNet-34 network pre-trained on CIFAR-100. It serves as the source of knowledge for training the student via feature-level and prediction-level guidance. The teacher is frozen during AKD and used solely for generating soft targets.

- Depth: 34 layers
- Output classes: 100
- Pretrained accuracy: 85.1%
- Input size: Adjusted to match generator output (224×224)

The teacher encapsulates rich semantic representations learned from the real data distribution. Its response to synthetic inputs becomes the ground truth for student training.

1.5 Training Procedure

The training proceeds in epochs, each consisting of multiple iterations alternating between student updates and generator updates.

1.5.1 Student (Imitation) Stage

1. Sample noise: $z \sim \mathcal{N}(0, I)$
2. Generate image: $x = G(z)$
3. Compute teacher output: $T(x)$
4. Compute student output: $S(x)$

5. Compute loss: $\mathcal{L}_S = \|S(x) - T(x)\|_2^2$ (L2 loss)

6. Backpropagate and update student parameters

This encourages the student to mimic the teacher's predictions for synthetic inputs. The loss captures fine-grained class probabilities rather than hard labels.

1.5.2 Generator (Adversarial) Stage

- Sample new $z \sim \mathcal{N}(0, I)$
- Generate image: $x = G(z)$
- Get outputs $T(x)$ and $S(x)$
- Maximize student-teacher disagreement:
- $\mathcal{L}_G = -\log(\|S(x) - T(x)\|_2^2 + \epsilon)$

This adversarial loss drives the generator to produce inputs where the student still diverges from the teacher, maximizing the information gain for the student.

The two losses are optimized using:

- SGD for student (with momentum and weight decay)
- Adam for generator (with fixed learning rate)

A learning rate scheduler (multi-step decay) is additionally employed to ensure convergence over epochs.

1.6 Evaluation and Visualization

After training, the student is evaluated on the CIFAR-100 test set (real images) using classification accuracy and cross-entropy loss. This assesses whether the student has learned a generalizable mapping from the synthetic training signal.

1.6.1 Evaluation Metrics

- Test Accuracy: Percentage of correct predictions
- Test Loss: Average cross-entropy loss over test samples
- Confusion Matrix: Qualitative analysis of per-class performance

1.6.2 Generated Image Visualization

The generator's outputs are visualized periodically during training. A batch of noise vectors is passed to the generator, and the resulting synthetic images are:

- Assembled into a grid
- Normalized to [0,1] for display
- Saved using *matplotlib* and *torchvision.utils.make_grid*

This provides a visual cue on the diversity and realism of the generated data.

2 Results

2.1 Reporting parameters

Table 1: Trainable and Non-Trainable parameter counts for ResNet-34 student models

Model	Trainable Parameters	Non-Trainable Parameters
ResNet-34 (20% parameters)	4,227,747	0
ResNet-34 (50% parameters)	10,640,858	0

2.2 Reporting on 20% Test data

- Test Accuracy - Resnet with 20% parameters: 15.62 % . The training was carried out for 300 epochs, which took six hours.

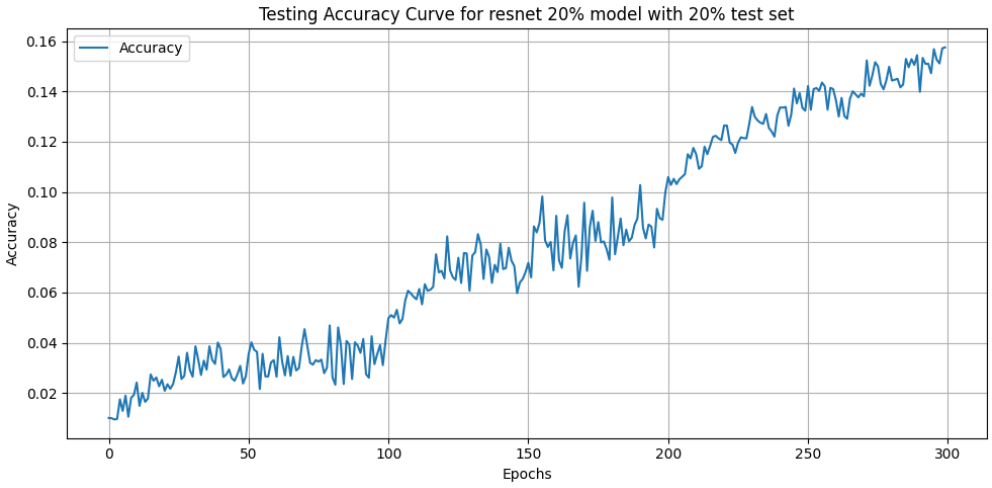


Figure 1: Model was trained on MAE loss

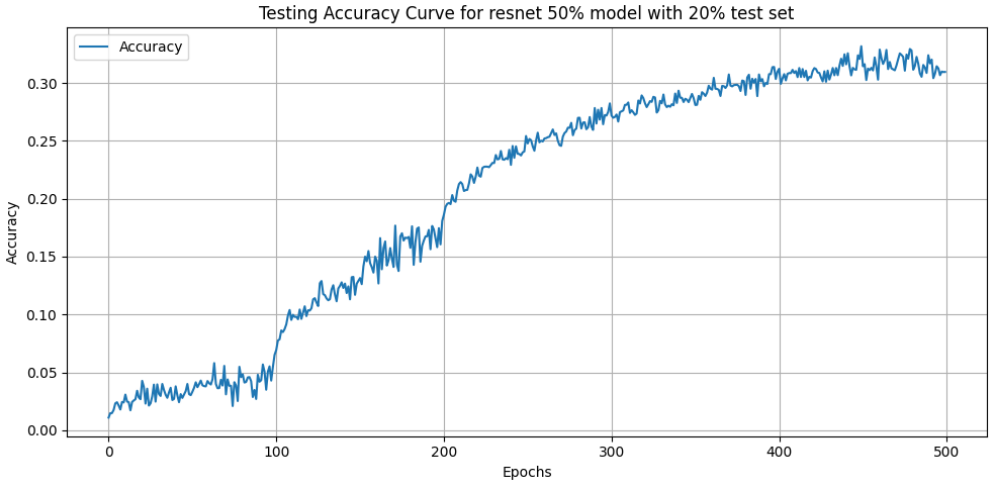
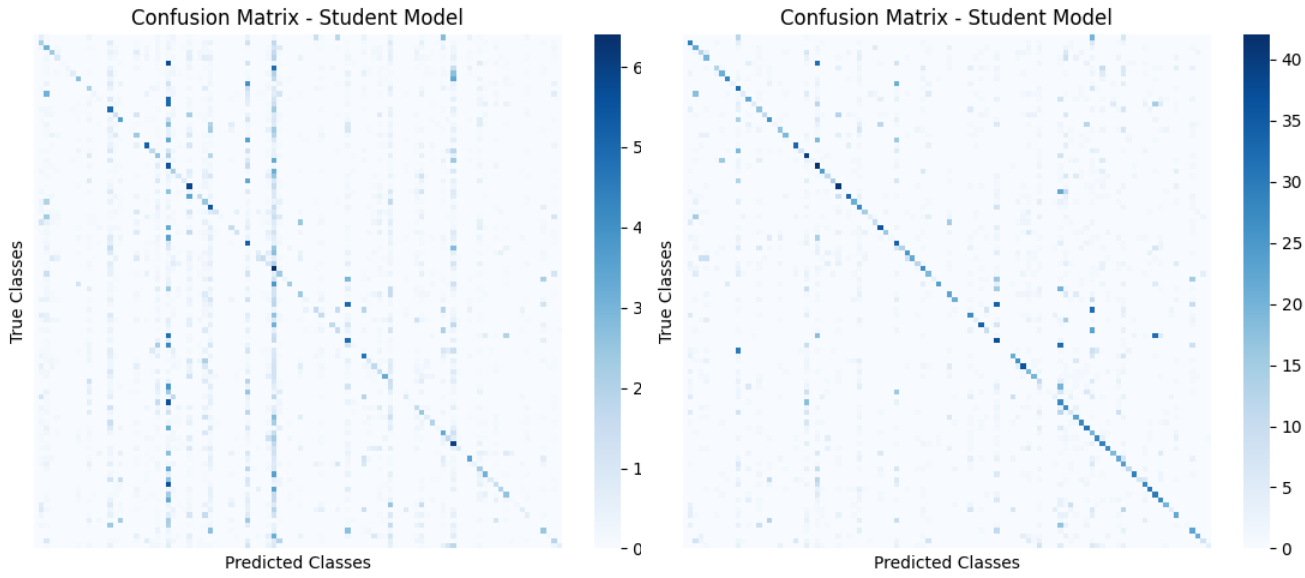


Figure 2: MSE loss is used to train this model only

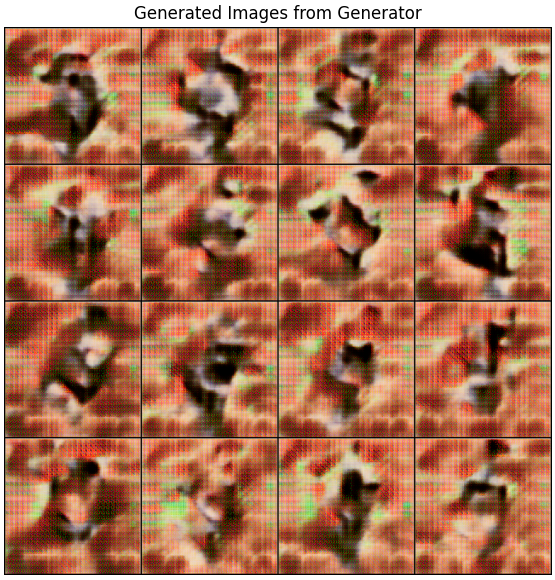
- Test Accuracy - Resnet with 50% parameters: 34.66 % . The training was carried out for 500 epochs, which took twelve hours.
- Confusion Matrix - Resnet with 20% and 50% parameters:



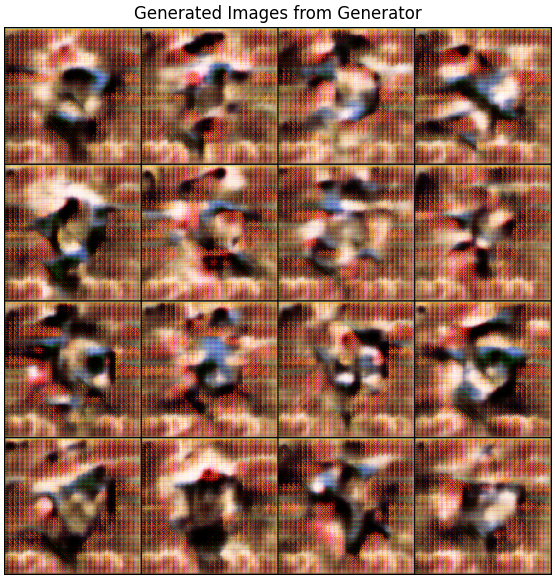
(a) ResNet-34 student with 20% parameters (b) ResNet-34 student with 50% parameters

Figure 3: Confusion matrices for student models with different parameter budgets.

- Images generated by Generator trained with Resnet34-20 and Resnet34-50 discriminators.



(a) ResNet-34 with 20% parameters



(b) ResNet-34 with 50% parameters

Figure 4: Generated images from the generator model trained with different student capacities.

2.3 Reporting on 10% Test data

- Test Accuracy - Resnet with 20% parameters: 23.98 % . The training was carried out for 500 epochs, which took twelve hours.

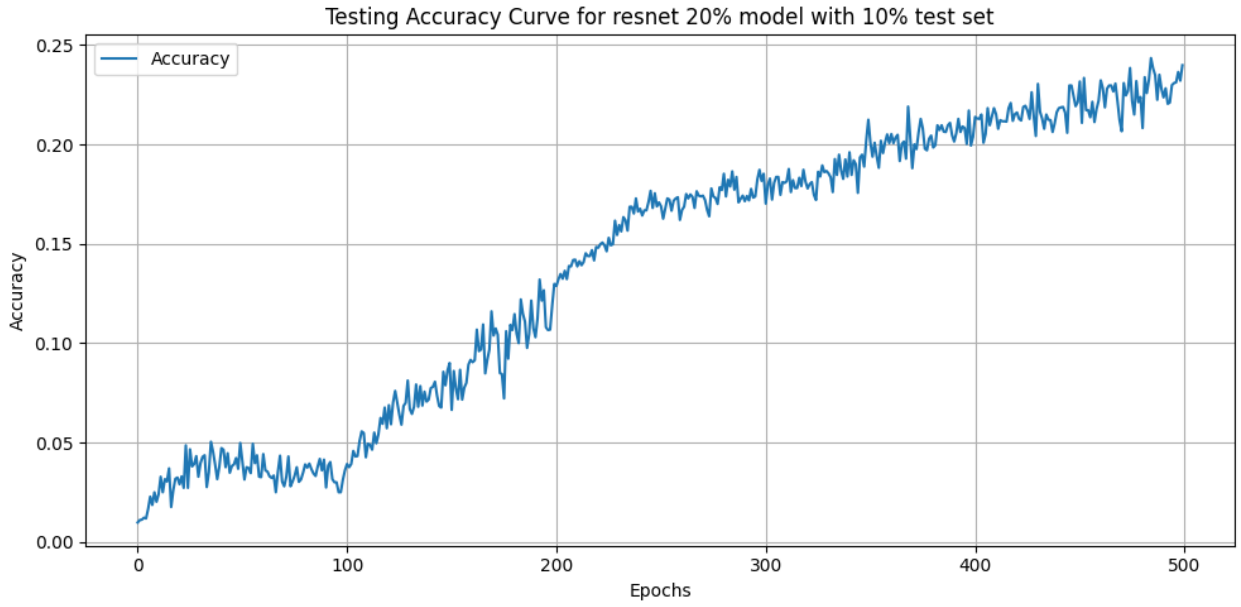


Figure 5: Used MAE loss

- Test Accuracy - Resnet with 50% parameters: 30.18 % . The training was carried out for 500 epochs, which took twelve hours.

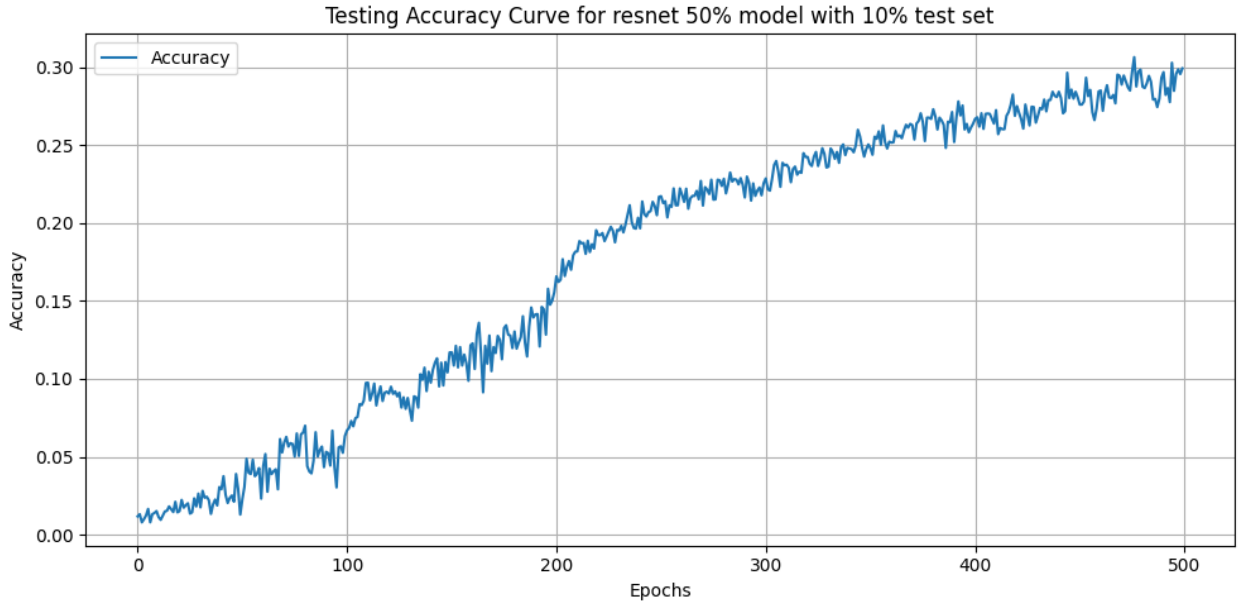
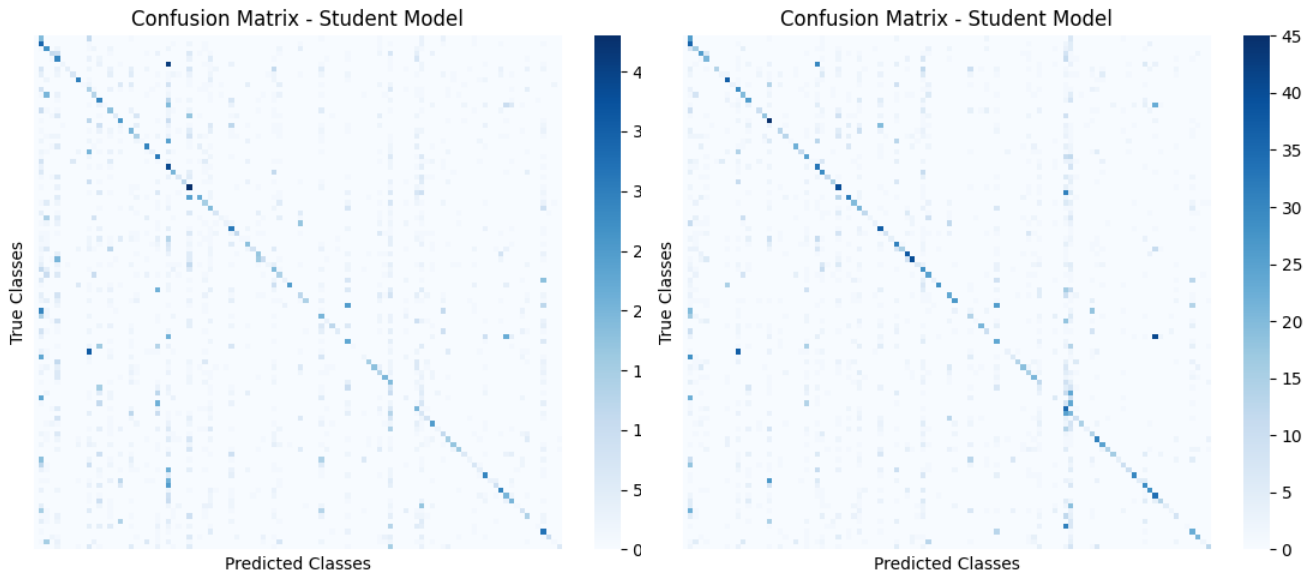


Figure 6: Used MAE loss

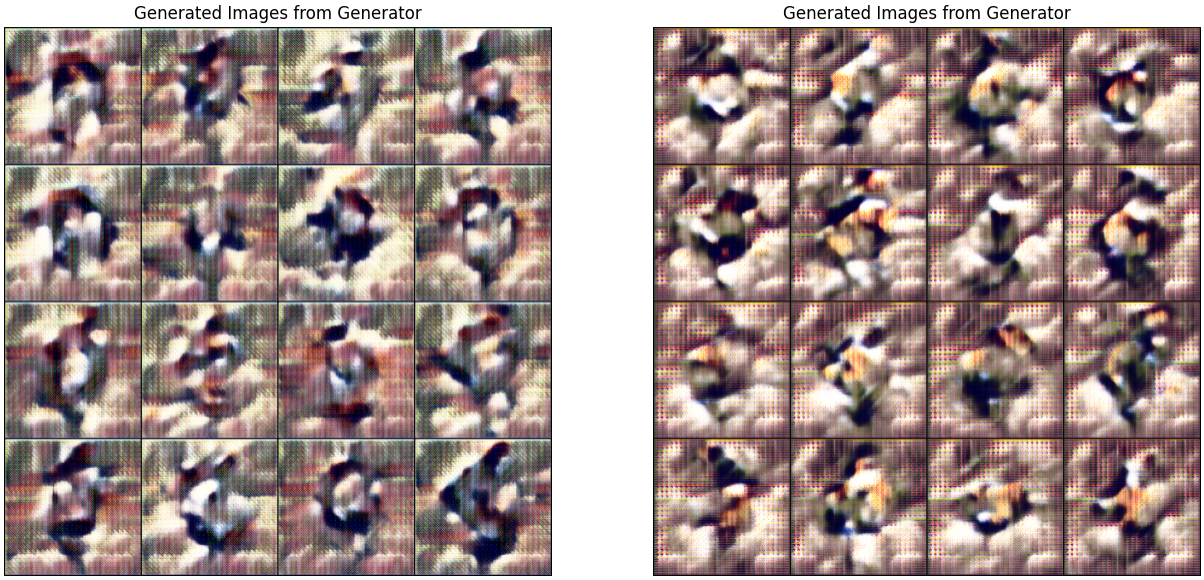
- Confusion Matrix - Resnet with 20% and 50% parameters:



(a) ResNet-34 student with 20% parameters (b) ResNet-34 student with 50% parameters

Figure 7: Confusion matrices for student models with different parameter budgets.

- Images generated by Generator trained with Resnet34-20 and Resnet34-50 discriminators.



(a) ResNet-34 with 20% parameters

(b) ResNet-34 with 50% parameters

Figure 8: Generated images from the generator model trained with different student capacities.

3 Observations

- MSE (or L2) loss performed better than MAE (or L1) loss. This can be due to the fact that the MSE amplifies the effect of outliers due to the squaring of error terms, affecting the model's error estimation more substantially, as evidenced by the figures 2 and 6.
- More parameters showed a better result, which can be seen by comparing figures 5 and 6.
- One may conclude that a more parametric student model acts as a better adversary for the generator by just observing the figures 8a and 8b. Observing figures 4a and 4b, the generator learns better by a better discriminator, which fits the results for both cases and not merely by a heavier discriminator.

4 Discussion

This AKD framework demonstrates the feasibility of compressing models in the absence of real data. Key observations include:

- **Effective Knowledge Transfer**: Students trained with adversarial synthetic data reach impressive accuracies (**24%** with 20% parameters and **35%** with 50% parameters).
- **Generator Expressiveness**: A well-structured deconvolutional generator can explore semantically meaningful regions of the input space even without labels.
- **Loss Function Impact**: L2 loss provides a little smoother gradients and better alignment with the teacher's feature space than L1 loss.
- **Parameter Scaling**: Larger student models (50%) benefit more from the richer synthetic data, while smaller models exhibit faster convergence but lower ceilings.
- **Data Diversity**: The adversarial loop ensures that the generator does not collapse to a limited mode but instead explores areas that help student generalization.

5 Conclusion

We presented a comprehensive implementation of adversarial knowledge distillation using a generator-student-teacher triad, trained without access to original data. Through adversarial feedback and distillation losses, the student effectively learns from the teacher's responses to synthetically generated inputs.

This framework has broad applications in privacy-sensitive domains where sharing training data is not feasible, such as medical imaging, finance, and defense. The success of our experiments with CIFAR-100 validates the potential of data-free methods to democratize deep learning model deployment.

Future work may explore more expressive generative models (e.g., GANs with discriminators, diffusion models), extend the framework to different modalities (e.g., NLP), or integrate unlabeled real-world data into the loop for hybrid distillation strategies.

Click here to explore full code