Efficient CIFAR-100 Modelling

Fener Ahmed

Abstract

This paper explores the development of a compact, efficient classifier for the CIFAR-100 dataset using a convolutional neural network (CNN) and the creation of a generative model using a Deep Convolutional Generative Adversarial Network (DCGAN). Aimed at balancing model complexity with computational efficiency, this study adheres to strict parameter and optimisation step constraints.

Part 1: Classification

1 Methodology

This project uses a Convolutional Neural Network (CNN) inspired by a minimalist design for the classification of images from the CIFAR-100 dataset. CNN's are particularly designed for processing data such as images. Our CNN architecture is constructed to reduce the number of trainable parameters while maintaining or enhancing the model's predictive accuracy on the CIFAR-100 dataset.

1.1 CNN Architecture

Our CNN model integrates several convolutional layers, batch normalisation [3], ReLU activation functions [4], and residual blocks [2] to efficiently learn from the CIFAR-100 dataset with a reduced parameter count.

The convolutional operation can be described as: $F_{out}(x) = ReLU(W * x + b)$ where x is the input, W represents the convolutional weights, b is the bias, and F_{out} denotes the output feature map. This operation is fundamental in extracting hierarchical features from the input images.

Residual learning frameworks are incorporated to mitigate the vanishing gradient problem, enabling the training of deeper networks. The identity mapping in residual blocks is defined as: $y = F(x, \{W_i\}) + x$ where x and y are the input and output of the layers encompassed by the function F, respectively, and $\{W_i\}$ are the weights [2].

1.2 Training Strategy

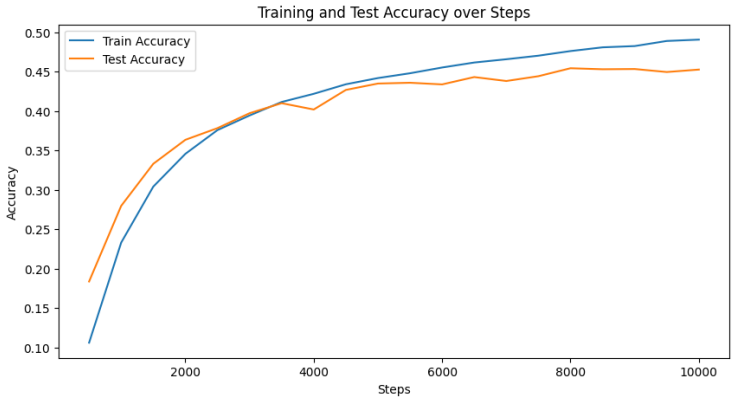
The Cross-Entropy Loss function is utilized to measure the model's prediction accuracy, facilitating multi-class classification [1]: $L = -\sum_{c=1}^{M} y_{o,c} log\left(p_{o,c}\right)$ where M represents the number of classes, y is a binary indicator if class label c is the correct classification for observation o, and p is the predicted probability that observation o is of class c.

2 Results

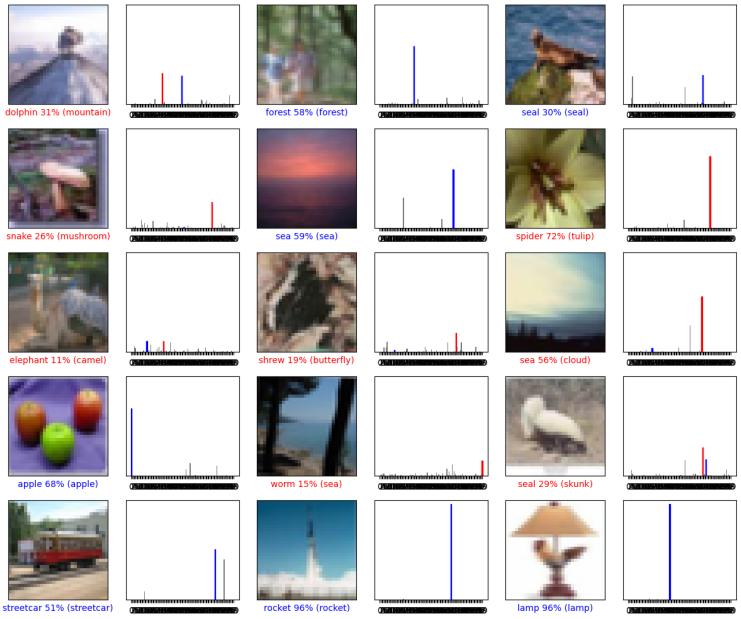
The network has 97,748 parameters.

The classifier attains a final training accuracy of 49% and a final testing accuracy of 45% at 10,000 optimisation steps, which is not the best and can surely be improved (Train Loss: 1.8797%, Train Acc: 0.4906%, Test Acc: 0.4526%)

Here is the training graph:



Here are some non cherry-picked examples of classifications done:



3 Limitations

While these results demonstrate some capability to generalise from the CIFAR-100 training data, there remains a substantial margin for improvement. The discrepancy between training and testing accuracy indicates overfitting, despite efforts to regularise the model. Future improvements could explore more sophisticated data augmentation techniques or deeper, more complex network architectures. Additionally, refining the balance between model complexity and the risk of overfitting will be crucial. Achieving these improvements, however, may increase the computational demand, suggesting a need for more extensive training periods.

Part 2: Generative model

4 Methodology

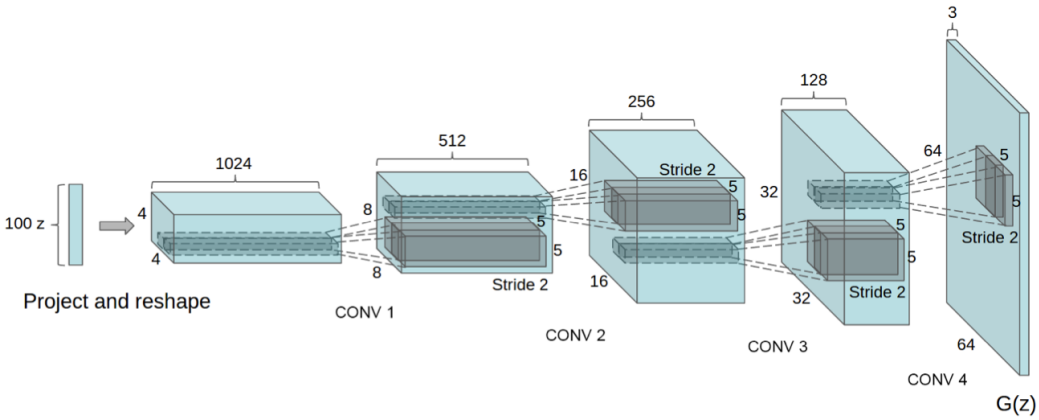
The DCGAN framework is a pivotal advancement in unsupervised learning with CNNs, introducing architectural constraints that have shown to stabilize training and facilitate the learning of rich representations.

The Generator ($\boldsymbol{G}$) architecture utilises transposed convolutions for upsampling, moving from a latent space vector ($\boldsymbol{z}$) to a full resolution image. Specifically, $\boldsymbol{G}$ starts with a latent vector $\boldsymbol{z} \in \mathbb{R}^{\mathbf{80}}$, mapped through a series of strided transposed convolutions—effectively learning its spatial upsampling to generate an image $x^{'}$. The discriminator ($\boldsymbol{D}$), conversely, employs strided convolutions to downsample the input image to a single scalar output indicating the authenticity of the input image. [5]

Following the DCGAN paper, our model adopts several key architectural features:

- **Strided and Fractionally-Strided Convolutions**: In $\boldsymbol{D}$, strided convolutions reduce spatial dimensions, while in $\boldsymbol{G}$, fractionally-strided convolutions (also known as transposed convolutions) increase spatial dimensions.
- **Batch Normalisation**: Applied in both $\boldsymbol{G}$ and $\boldsymbol{D}$ to stabilise training by normalizing the input to each unit to have zero mean and unit variance.
- **Activation Functions**: $\boldsymbol{G}$ employs ReLU activation for all layers except the output, which uses Tanh. $\boldsymbol{D}$ utilises LeakyReLU for all layers to aid in higher resolution modeling and stability.

DCGAN generator architecture:

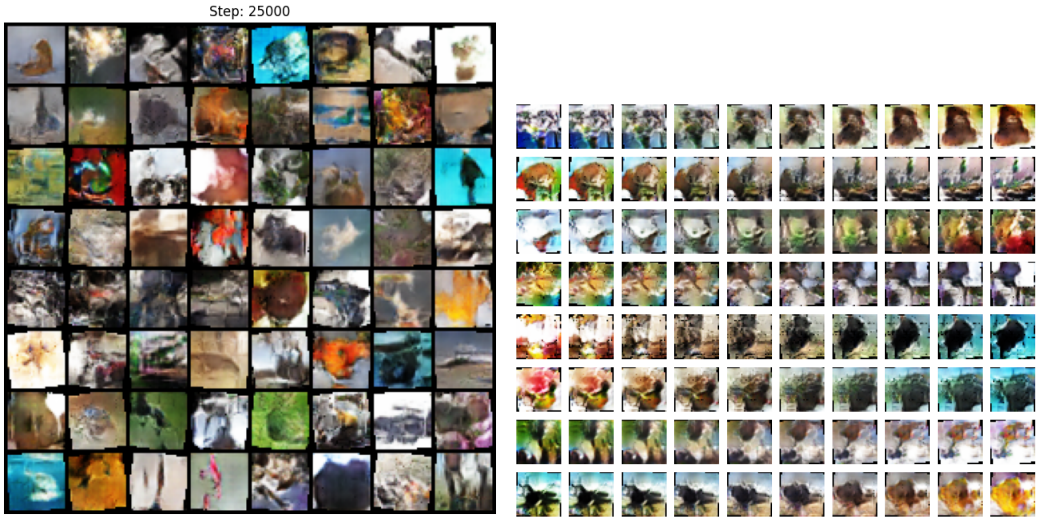


In the referenced architecture, a 100-dimensional noise vector, denoted as $\boldsymbol{z}$, undergoes a series of upscaling transformations through the "ConvTranspose2D" operation in PyTorch. Initially, $\boldsymbol{z}$ is upscaled to a 4x4 spatial dimension featuring 1024 channels. Subsequent upscaling steps further expand $\boldsymbol{z}$ to 8x8 dimensions with 512 channels, progressing until the final output, $\boldsymbol{G}(\boldsymbol{z})$, achieves a resolution of 64x64 pixels with 3 channels, corresponding to the RGB colour space.

5 Results

The network consisting of the generator and the discriminator has 617,376 and 374,592 parameters respectively, totaling to 991,968 parameters. The DCGAN achieves a FID score of 53.36 against the CIFAR-100 test dataset. The DCGAN was trained for the maximum 50,000 optimisation steps.

The results of the DCGAN look a bit blurry, however, there are realistic shapes and textures visible in the non-cherry picked sample images (left). The interpolations between the points in the latent space look like this (right):



6 Limitations

One potential limitation lies in the model's architecture or training steps, which may benefit from further optimisation to enhance the fidelity and diversity of the generated images. The balance between the generator and discriminator during training —a critical aspect of GAN performance— might also contribute to the FID score.

Future work could explore alternative architectures, loss functions, or more advanced regularisation techniques to refine the quality of generated images. Adjusting the training strategy, such as by employing different learning rates for the generator and discriminator or integrating recent advancements in GAN training methodologies, may also yield improvements.

7 Note on References

In the preparation of this document, specific challenges were encountered in accurately citing certain sources due to technical constraints. Particularly, the citation for one online resource could not be formatted in congruence with the rest of the references without causing disruptions. The reference is "https://towardsdatascience.com/how-to-reduce-training-parameters-in-cnns-while-keeping-accuracy-99-a213034a9777" which inspired my classifier model.

References

[1] Pieter-Tjerk de Boer et al. "A Tutorial on the Cross-Entropy Method". English. In: *Annals of operations research* 134.1 (Jan. 2005), pp. 19–67. ISSN: 0254-5330. DOI: 10.1007/s10479-005-5724-z.

[2] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[3] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[4] Vinod Nair and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[5] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].