Efficient CIFAR-100 Classification and Modelling using DenseNet and DCGAN

Yosef Berezovskiy

Abstract

This paper proposes use of deep dense convolutional network (DenseNet) to classify images, and using Deep Convolutional Generative Adversarial Network (DCGAN) to generate images on CIFAR-100 dataset of resolution 32x32. The paper discusses utilised methodologies and presents, and analyses yielded results.

Part 1: Classification

1 Methodology

The method involves training deep DenseNet, introduced in 2018 [1], with 4 dense blocks, each containing 5 layers. The equipped dense connectivity pattern in the network strengthens feature propagation and encourages feature reuse, improving parameter efficiency - making network a suitable choice given the 100,000 parameter constraint. Method connects all layers directly to each other such that the ℓ^{th} layer receives the feature-maps of all preceding layers, $x_0, \ldots, x_{\ell-1}$, as input:

$$x_\ell = H_\ell([x_0, x_1, \ldots, x_{\ell-1}]) \tag{1}$$

where x_0 is a single image passed through the network of L layers, $H_\ell(\cdot)$ is a non-linear transformation applied at each layer, x_ℓ is an output of the ℓ^{th} layer, and $[x_0, x_1, \ldots, x_{\ell-1}]$ refers to the concatenation of the feature-maps produced in layers $0, \ldots, \ell - 1$. $H_\ell(\cdot)$ is defined to be a combination of batch normalisation (BN)[2], rectified linear unit (ReLU)[3] and a 3x3 convolution (Conv) applied consecutively, as motivated by [4]. To enable concatenation in Eq.(1) given changing feature-map sizes down-sampling is performed through transition layers between densely connected blocks (Fig. 1) that perform 1x1 Conv and 2x2 average pooling. In DenseNet, each function $H_\ell(\cdot)$ generates k feature maps, meaning the ℓ^{th} layer receives $k_0 + k(\ell - 1)$ input feature maps, with k_0 being the initial channel count, , with k defining the network's growth rate. The example in Figure 1 has $k = 4$ [1]. The

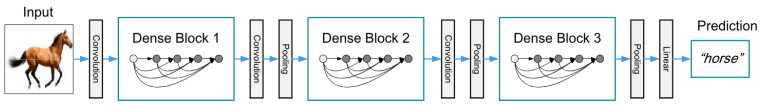


Figure 1: An example of deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling[1].

pre-processing steps for training the model involved augmenting and normalizing the data to prevent overfiting. Techniques include random cropping with reflection padding, horizontal flipping, and AugMix — a method that mixes multiple augmented images [5]. These steps were determined through extensive experimentation to introduce beneficial variability and noise into the training process. Although large batch sizes can cause poor model generalisation, batch size of 512 was determined experimentally to improve the performance of the model without causing overfiting - this can be attributed to the increase of noise and data augmentation of the training data.

2 Results

The network has 99,735 parameters. It attains 66.4% training accuracy and also 63.1% testing accuracy at 10,000 optimisation steps, which is a good result implying that model outperforms significantly multiple benchmarks provided by [6] (train loss: 1.144, train acc: 0.6640.021, test acc: 0.631±0.016).

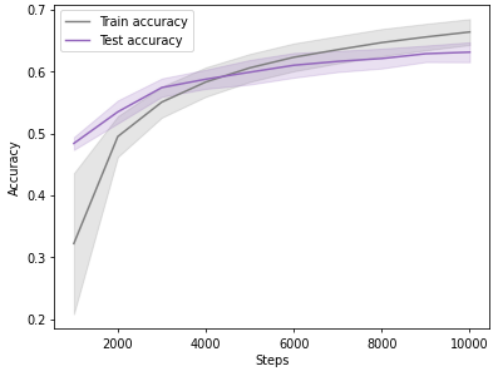


Figure 2: DenseNet training graph with batch size 512, learning rate 0.0075, growth rate 8, 4 dense blocks containing 5 layers

3 Limitations

Model has shown comparatively high accuracy [5] and low overfiting, hence it is a useful classifier. However, constraining parameter counts limits DenseNet's depth and density, detracting from its potential performance [1]. Easing these constraints may boost its efficacy, however, given the network's design to concatenate feature maps from each layer with those from all preceding layers, leading to parameter count growth and, consequentially, exponential rises in computational and memory requirements, scalability challenges are present. An analysis of individual image classifications (Fig. 3) reveals that the model's confidence diminishes in inherently ambiguous categories, particularly in images with overlapping categories (e.g., sea and clouds) or where low resolution and visual similarities impede accurate classification. However, a notable number of sampled images achieve high classification accuracy (90%-100%), underscoring the trained model's high performance metrics (Fig. 2).

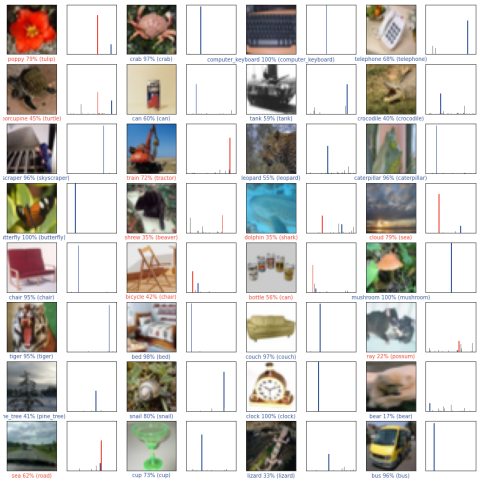


Figure 3: Inference of trained classifier on dataset

Part 2: Generative model

4 Methodology

Basic GAN Properties

Generative Adversarial Networks (GANs) comprise two neural networks: the Generator (G) and the Discriminator (D). They engage in a min-max game where G generates data mimicking real data, and D distinguishes between actual and generated data. The core objective function of a GAN is given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2}$$

where x is real data from distribution p_{data}, and z is a noise vector from distribution p_z. Training involves alternating updates: enhancing D to accurately identify real versus fake data, and refining G to deceive D. This competition improves G's data realism. In the implementation, both networks use Binary Cross Entropy Loss (BCELoss) to optimize performance, defined as:

$$BCELoss = -\frac{1}{N}\sum_{i=1}^{N} \omega_i \left(y_i \log x_i + (1 - y_i) \log(1 - x_i)\right) \tag{3}$$

Deep Convolutions

Deep Convolutional Generative Adversarial Networks (DCGANs) utilize deep convolutional layers to improve Generative Adversarial Networks (GANs) in image synthesis. DCGANs incorporate convolutional layers in the Discriminator for downsampling and feature extraction, and transposed convolutional layers in the Generator for upsampling from latent space. Enhancements include the use of strided and fractional-strided convolutions, batch normalization to mitigate internal covariate shift, and LeakyReLU activations to avoid the "dying ReLU" problem, thereby enhancing training stability and mitigating issues such as mode collapse and vanishing gradients. The symmetrical architecture of DCGANs, comprising five convolutional layers with specific stride, padding, and kernel size parameters, facilitates effective dimensionality reduction and expansion for complex representation learning. The Discriminator employs a sigmoid output for classification, while the Generator expands a 128-dimensional latent vector into images. Labeling strategies for real and fake images are adjusted to encourage diversity and prevent Discriminator overconfidence. Additionally, data augmentation and normalization techniques are employed to improve dataset variability and model efficacy.

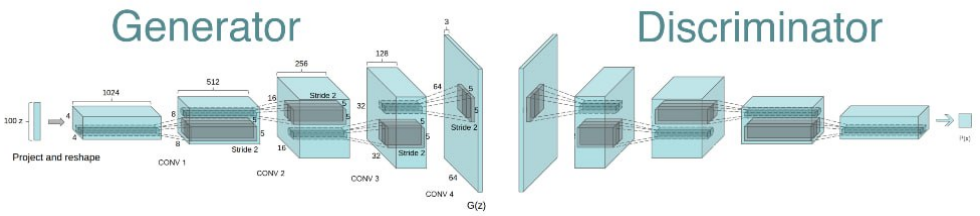
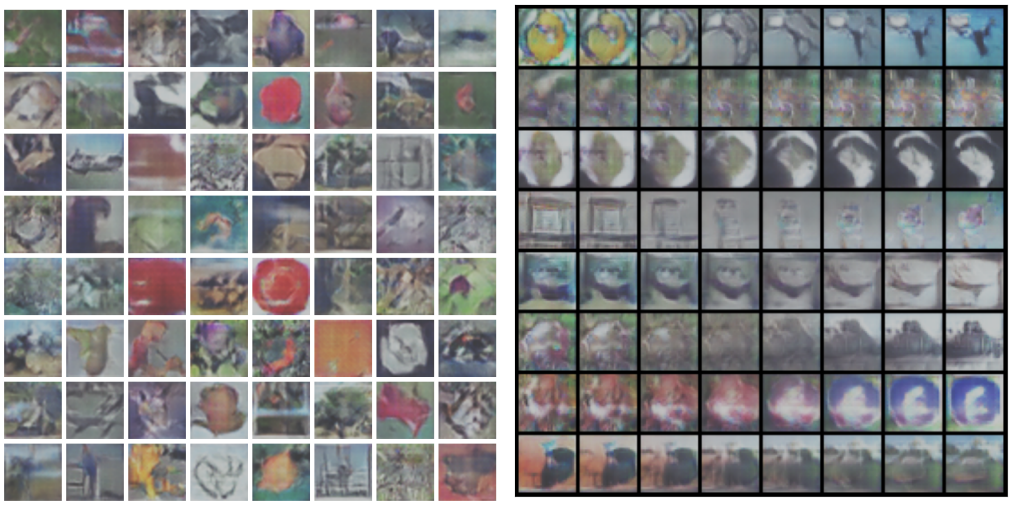


Figure 4: DCGAN Architecture

5 Results

The network has 595,840 parameters and achieves an FID of 61.047 against the CIFAR-100 test dataset. It was trained for 49,998 optimisation steps.

The results look blurry and majority of shapes are difficult to recognise, however images look diverse and have relatively realistic colors and textures. Human observer can distinguish silhouettes of animals, flowers and other objects (e.g. plate). A random batch of non-cherry picked samples is presented on the left. A set of linear interpolants between points in the latent space is presented on the right.



6 Limitations

As you can see, the images generated by the algorithm are not very similar to the pictures from the dataset. However, it can be seen that in the pictures you can clearly distinguish the outlines and boundaries of some objects, as well as the background on which they are located. Thus, we come to the conclusion that the neural network is learning, but for more detailed generation it lacks depth and more layers that could highlight more characteristic and clear features, as well as, quite possibly, the number of iterations in training, which could also affect the clarity of details and their drawing. It is worth mentioning that we also tried more advanced neural networks such as WGAN, LSGAN and cGAN, however, due to the fact that their parameters and the number of iterations had to be significantly reduced, they could not generate images similar to real ones, so the simplest model showed the best results.

7 References

[1] Huang, G. et al. (2018) 'Densely Connected Convolutional Networks'. arXiv. Available at: http://arxiv.org/abs/1608.06993

[2] Ioffe, S. and Szegedy, C. (2015) 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'. arXiv. Available at: https://doi.org/10.48550/arXiv.1502.03167.

[3] Glorot, X., Bordes, A. and Bengio, Y. (2011) 'Deep Sparse Rectifier Neural Networks', in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, pp. 315–323. Available at: https://proceedings.mlr.press/v15/glorot11a.html

[4] He, K. et al. (2016) 'Identity Mappings in Deep Residual Networks'. arXiv. Available at: https://doi.org/10.48550/arXiv.1603.05027.

[5] Hendrycks, D. et al. (2020) 'AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty'. arXiv. Available at: http://arxiv.org/abs/1912.02781

[6] Papers with Code - CIFAR-100 Benchmark (Image Classification) (no date). Available at: https://paperswithcode.com/sota/image-classification-on-cifar-100