Parameter Efficient Discriminative and Generative Modelling

Ben Hughes - krcc82

Abstract

This paper proposes using a MobileNet [3] architecture to classify images from the AddNIST [1] dataset and a Spectral-Normalized [5] Deep Convolutional Generative Adversarial Network [6] to generate images from the CIFAR-100 [4] dataset. The classifier achieves 95.2% training accuracy and also 90.4% testing accuracy with 97,687 parameters and 10,000 training steps. The Generator achieves realistic textured images with 56.63 FID[2] score with 973,936 parameters and 50,000 training steps.

Part 1: Classification

1 Methodology

This section outlines the methodology employed to train a MobileNet[3] classifier on the AddNIST[1] dataset. The approach includes a detailed explanation of the model architecture and the training process.

1.1 Model Architecture

The proposed classifier, referred to as **Model A**, utilizes MobileNet blocks adapted from the design outlined in [3]. MobileNet blocks use depthwise and pointwise convolutions to significantly reduce the number of parameters compared to traditional convolutional layers, whilst maintaining similar performance, making them particularly effective under the given parameter constraints.

The model takes AddNIST input images of dimensions $28 \times 28 \times 3$, for each of the RGB channels. Each channel is processed independently through a sequence of six MobileNet blocks, followed by recombination into a single tensor. The architecture is:

- **Per-channel blocks:** Due to the nature of the dataset, with the character of each channel being independent of each other, each RGB channel is independently processed through six MobileNet blocks. These blocks progressively expand the number of channels while maintaining spatial dimensions, followed by downsampling in the sixth block. The final per-channel output dimensions are $14 \times 14 \times 24$.
- **Combined blocks:** The outputs of each channel from the previous blocks are concatenated along the channel dimension, resulting in a tensor of dimensions $14 \times 14 \times 72$. Then, this tensor is passed through four more MobileNet blocks, where their dimensions are reduced to 1×1, with channel dimensions expanded to 196.
- **Global Average Pooling:** A global average pooling layer reduces the tensor to a $1 \times 1 \times 196$ feature vector.
- **Classification Head:** The feature vector is flattened and passed through two fully connected layers, first mapping $196 \rightarrow 64$, followed by $64 \rightarrow 20$, the number of classes.

The complete architecture is illustrated in Fig 1.

1.2 Training

The model was trained on the AddNIST dataset using the Adam optimizer with a learning rate of 0.003. The training process was conducted for a maximum of 10,000 optimization steps, ensuring compliance with the given constraints. Training and test accuracy were monitored during the process. Cross-entropy loss was used to optimize the model.

To optimize the model the number of layers was changed, with a deeper, more narrow architecture being more optimal than a shallow wide one. The learning rate was adjusted with the value of 0.003 being optimal out of 0.001, 0.005 & 0.008. LeakyReLU, GELU and ReLU were all tested as activation functions with LeakyReLU being best. Experimentation was also done with the batch size, as the batch size increased the training accuracy would increase but the gap between training and testing increased aswell, as the model began to overfit. A batch size of 96 was used as this gave the best balance.

2 Results

The total number of parameters in the model is 97687.

It attains 95.2% training accuracy and also 90.4% testing accuracy at 10,000 optimisation steps. (train loss: 0.155, train acc: 0.952±0.022, test acc: 0.904±0.037). It is very confident with its predictions, with the classification category often having over 90%, however it is occasionally wrong despite this.

The training graph is shown in Fig. 2

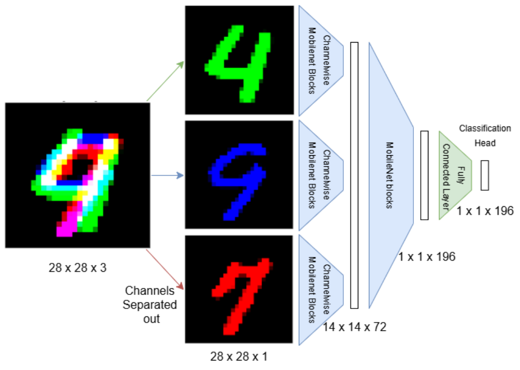

Figure 1: Classifier Diagram

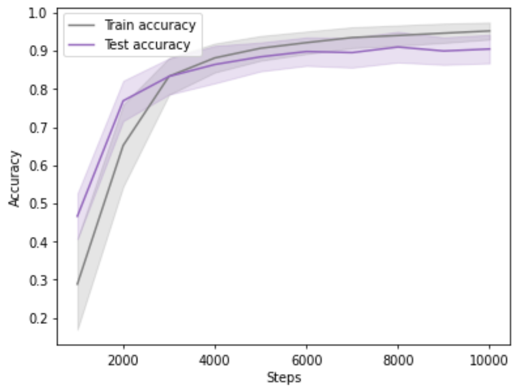

Figure 2: Training Graph

Fig 2. shows how the model begins to over-fit towards the training data around the 3,000th training step. The over-fit by the end of training is only around 5% and the test accuracy is still over 90% so the model generalizes quite well to the test data.

The channel-wise MobileNet blocks contribute a lot to the performance, as they extract details from each channel independently before recombining them - helping the model learn each digit in each channel quicker. without this separation and with just a MobileNet architecture the accuracy was around 80% train and 70% test so they provide a big improvement.

3 Limitations

The results are very good with very high test and train accuracy and only small difference between the 2 suggesting the model has not been over fit to the training data. The architecture may not generalize very well to other problems, as it splits the channels up which is very specific to this task. The training graph suggests that with more training steps the train accuracy may still improve, so using more training steps may improve performance. The channel wise classification blocks could potentially be reused for each channel, this would give them much more ability to learn the meaning of each digit whilst also reducing the number of parameters needed for that part of the model by 200%

Part 2: Generative model

4 Methodology

The method for generation involves training a Spectral Normalization[5] Deep Convolutional Generative Adversarial Network[6] (SN-DCGAN) on the CIFAR-100[4] dataset.

4.1 Model Architecture

The SN-DCGAN consists of two neural networks: a generator G and a discriminator D. Both have deep convolutional architectures and spectral normalization [5] is used in the discriminator to stabilize training and improve performance.

The generator takes a latent vector $\boldsymbol{z} \sim p_{\boldsymbol{z}}$ as input and transforms it into an image through convolutional layers with batch normalization and ReLU activation. The discriminator, works as a binary classifier, with deep convolutional layers, spectral normalization and LeakyReLU activation to tell apart the real and generated images.

Generator: The generator takes a Latent vector $\boldsymbol{z} \sim \mathcal{N}(0,1)$ with 100 dimensions as input. It applies 4 convolutional layers to the tensor, reducing the size $184 \to 128 \to 64 \to 3$. Batch normalisation is applied between all layers but the output, with ReLU activation in hidden layers and Tanh in the output layer. This results in a final output of a 32×32 RGB image.

Discriminator: The discriminator takes a 32×32 RGB image and returns a probability $0 \to 1$ of the image being real. It consists of 4 convolutional layers with spectral normalization: $3 \to 32 \to 64 \to 128 \to 1$, to reduce the chance of mode collapse during training. Batch normalization in all but first and last layers, along with LeakyReLU in hidden layers and Sigmoid activation in the output layer. Its output is a probability that the image is real.

4.2 Training

The training process uses an adversarial mini-max optimization objective to find the equilibrium point between the real and generated images, where the discriminator is no longer able to distinguish between.

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

Spectral normalization is applied to the discriminator's weight matrices to enforce Lipschitz continuity.

$$\hat{W} = \frac{W}{\sigma(W)}, \quad \sigma(W) \approx \mathbf{u}^T W \mathbf{v} \tag{2}$$

The Adam optimizer is used with a learning rate of 0.0005 for both the generator and the discriminator, with momentum parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. It uses a batch size of 64 and trains for 50,000 optimization steps.

The training loop consists of:

- Sample real images from the training data $\boldsymbol{x} \sim p_{\text{data}}$ and generate fake images $G(\boldsymbol{z})$.
- Compute discriminator loss:
$$\mathcal{L}_D = -\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))] \tag{3}$$
- Compute generator loss:
$$\mathcal{L}_G = -\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log D(G(\boldsymbol{z}))] \tag{4}$$

- Optimize D and G using backpropagation.

Fréchet Inception Distance (FID) [2] is used to evaluate the images:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}) \tag{5}$$

where μ_r, Σ_r and μ_g, Σ_g represent the mean and covariance of features from real and generated images, respectively.

The Model was optimized to have a low FID score as well as having visually good images. To optimize the model, experiments were done with changing the balance between model parameters of the discriminator and generator, if the discriminator was larger than the generator the model would collapse, whereas having a larger generator than discriminator improved performance. Layers were added and adjusted to maximize the parameters used, as more parameters increased performance. Experimentation was also done with the learning rate and momentum parameters, a grid search was done over LR 0.001 0.0005, $\beta_1 = 0.5, 0.8$ and $\beta_2 = 0.999, 0.99$, with the optimal values in FID score being used.

5 Results

The network has 973,936 parameters and achieves an FID score of 56.63 against the CIFAR-100 test dataset after 50,000 optimisation steps.

The results display a diverse range of images with different styles, colours and textures, showing the model has captured a range of patterns from the data. The objects have basic structure where you can see that it is trying to generate animals, objects and landscapes. The images have a good level of texture and some fine detail.

The interpolations have smooth transitions between images, suggesting there is a continuos latent space representation. There is a diversity in the variation across the grid. There is a morphing between images where features are preserved and they do not just look like alpha blending.

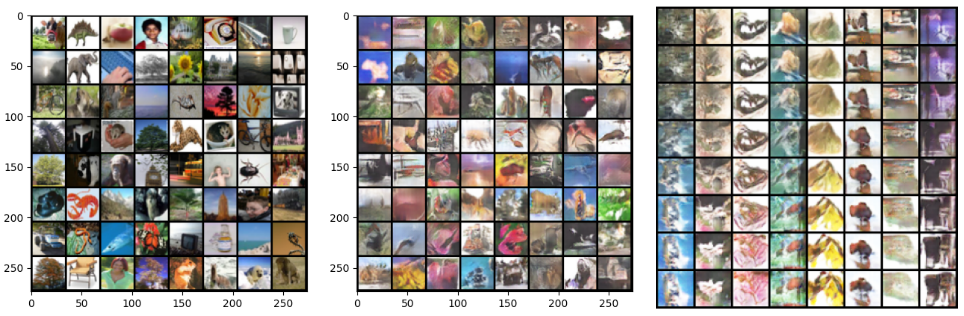

Figure 3: Comparison of different image representations. From left to right: Ground truth image, generated image, and interpolated image.

6 Limitations

There is a lack of realism in the objects, they do not look like anything recognisable. Some images are dull and grey in colour in comparison with the vibrant images from the ground truth. The interpolations lack gradual shifts in some parts and parts of the images come from nowhere. In the future, a different generative model with a better architecture could be used to obtain lower FIDs, such as diffusion. More experimentation could be done with the adjustment of the learning rate and betas for each generator and discriminator individually.

References

[1] Rob Geada et al. *Insights from the Use of Previously Unseen Neural Architecture Search Datasets*. 2024. arXiv: `2404.02189 [cs.LG]`. URL: `https://arxiv.org/abs/2404.02189`.

[2] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: `https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf`.

[3] Andrew G. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *CoRR* abs/1704.04861 (2017). arXiv: `1704.04861`. URL: `http://arxiv.org/abs/1704.04861`.

[4] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-100 (Canadian Institute for Advanced Research)". In: (). URL: `http://www.cs.toronto.edu/~kriz/cifar.html`.

[5] Takeru Miyato et al. *Spectral Normalization for Generative Adversarial Networks*. 2018. arXiv: `1802.05957 [cs.LG]`. URL: `https://arxiv.org/abs/1802.05957`.

[6] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: `1511.06434 [cs.LG]`. URL: `https://arxiv.org/abs/1511.06434`.