

Deep Generative Models (AI314)

Master of Science in AI



Dr. Nibras Abdullah

*Assistant professor
Faculty of Informatics
Midocean University*



Introduction



ProGAN



Progressive Training



Minibatch standard deviation



Pixelwise normalization



Introduction

Generative Adversarial Networks (GANs) are a powerful class of machine learning models that learn to generate new data. They are known for their ability to create realistic and diverse data, making them useful for a wide range of applications.



Introduction to ProGAN

ProGAN is a powerful technique developed by NVIDIA Labs in 2017 to train generative adversarial networks (GANs) more effectively. It introduces a progressive approach to training, gradually increasing the resolution of generated images to improve both speed and stability.



Challenges in GAN Training

1

Instability

Traditional GAN training can be unstable, leading to poor convergence and low-quality generated images.

2

Slow Convergence

GAN training often requires significant time and computational resources to reach high resolutions.

3

Mode Collapse

GANs can suffer from mode collapse, where the generator learns to produce limited, repetitive outputs.



Concept of Progressive Training

1

Start Small

Train the GAN on low-resolution images, e.g., 4x4 pixels.

2

Gradually Grow

Incrementally add layers to the generator and discriminator, increasing the resolution over time.

3

Reach High Res

Ultimately, the GAN is trained on high-resolution images, e.g., 1024x1024 pixels.



Incremental Layer Addition

Generator

The generator model is progressively expanded by adding new convolutional layers and upsampling blocks.

Discriminator

The discriminator model also grows in complexity, with new convolutional layers and downsampling blocks added.

Smooth Transition

The new layers are blended in gradually, ensuring a smooth transition between resolutions.

Starting with Low-Resolution Images

Stable Training

Training on low-res images is more stable, as the models have fewer parameters to learn.

Faster Convergence

Low-res training requires fewer iterations to reach convergence, speeding up the overall process.

Easier Learning

The generator and discriminator can focus on learning the basic structure and features before scaling up.

Reduced Memory

Working with lower-resolution images requires less GPU memory, allowing for larger batch sizes.

Starting with Low Resolutions

1

4x4 Images

The ProGAN is trained in stages, starting with a training set that has been condensed down to 4×4 -pixel images using interpolation. The generator is initially trained to transform a latent input noise vector into an image of shape $4 \times 4 \times 3$, while the discriminator is trained to transform an input image of size $4 \times 4 \times 3$ into a single scalar prediction.

2

Gradual Expansion

To expand the generator and discriminator to work with 8×8 -pixel images, new upsampling and convolutional layers are appended to the existing networks. This is managed in two phases: transition and stabilization.

3

Transition Phase

During the transition phase, the new layers are initially masked using a parameter α that is gradually increased from 0 to 1 throughout the transition phase to allow more of the new toRGB output through and less of the existing toRGB layer. This is to avoid a shock to the network as the new layers take over.



Gradual Increase in Resolution

1

4x4

Start with a 4x4 pixel
image resolution.

2

8x8

Incrementally add layers
to reach 8x8 pixels.

3

16x16

Continue growing the
models to 16x16 pixels.

4

32x32

Finally, reach high-
resolution 32x32 pixel
images.

Improved Speed and Stability

Traditional GAN

ProGAN

Slow convergence

Faster convergence

Unstable training

Stable training

Limited resolution

High-resolution outputs

High memory usage

Lower memory usage



Comparison to Traditional GAN Training

Traditional GAN

Trains the generator and discriminator on full-resolution images from the start, leading to instability and slow convergence.

ProGAN

Starts with low-resolution images and gradually increases the model complexity and resolution, resulting in faster training and higher-quality outputs.

Key Advantages

ProGAN's progressive approach addresses the core challenges of traditional GAN training, enabling more robust and efficient learning.



Potential Applications of ProGAN

Image Generation

ProGAN can be used to generate high-quality, photorealistic images for a variety of applications, such as art, design, and entertainment.

Video Generation

The progressive training approach can be extended to generate high-resolution videos, opening up new possibilities in areas like animation and visual effects.

Text-to-Image

ProGAN can be combined with language models to enable text-to-image generation, allowing users to create visual content from textual descriptions.

Super-Resolution

The progressive training technique can be applied to image super-resolution, upscaling low-resolution images to high-quality, high-resolution outputs.

Progressive Training with GANs

As always with GANs, we build two independent networks, the generator and discriminator, with a fight for dominance taking place during the training process. However, the ProGAN paper introduces a novel approach called "progressive training" that aims to improve the training process by starting with low-resolution images and gradually increasing the resolution over time.

Transitioning to Higher Resolutions

Transition Phase

In the transition phase, new upsampling and convolutional layers are appended to the existing generator and discriminator networks. A residual connection is set up to maintain the output from the existing trained toRGB layer.

Gradual Blending

The new layers are initially masked using a parameter α that is gradually increased from 0 to 1 throughout the transition phase. This allows more of the new toRGB output through and less of the existing toRGB layer, avoiding a shock to the network as the new layers take over.

Stabilization Phase

Eventually, there is no flow through the old toRGB layer and the network enters the stabilization phase - a further period of training where the network can fine-tune the output, without any flow through the old toRGB layer.

Benefits of Progressive Training



Improved Stability

The gradual approach helps stabilize GAN training, reducing mode collapse and oscillations.



Faster Convergence

ProGAN converges to high-resolution outputs much more quickly than traditional GAN training.



Higher Quality

The incremental layer addition leads to better-quality generated images with more realistic details.



Reduced Memory

The progressive approach requires less GPU memory, allowing for larger batch sizes.

Figure 1: Performance of ProGAN

Comparison of ProGAN and traditional GAN training

Comparison of ProGAN and traditional GAN training

Comparison of ProGAN and traditional GAN training

Comparison of ProGAN and traditional GAN training

Comparison of ProGAN and traditional GAN training

Comparison of ProGAN and traditional GAN training

Comparison of ProGAN and traditional GAN training

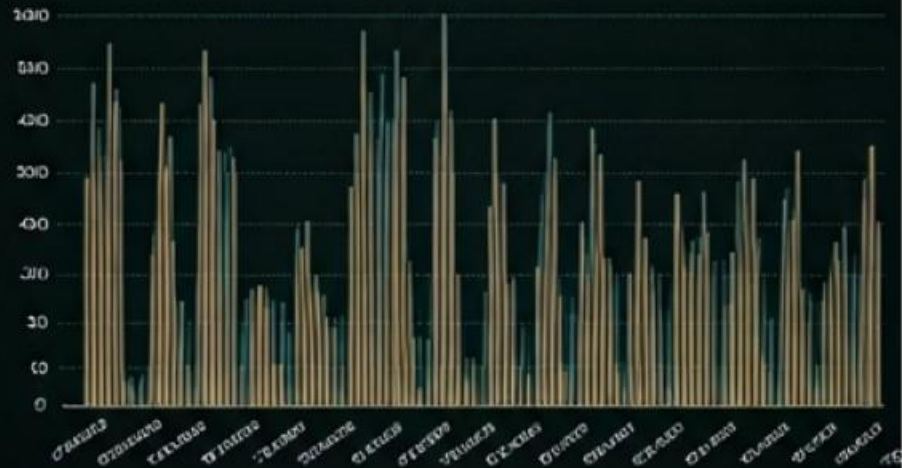
Comparison of ProGAN and traditional GAN training

Comparison of ProGAN and traditional GAN training

Comparison of ProGAN and traditional GAN training

Comparison of ProGAN and traditional GAN training

Comparison of ProGAN and traditional GAN training



Benefits of Progressive Training

1 Improved Learning

The generator might be slow to learn high-level structures in the early stages of training when operating over complex, high-resolution images. Starting with low-resolution images allows the generator to first learn the basic structures before gradually increasing the resolution.

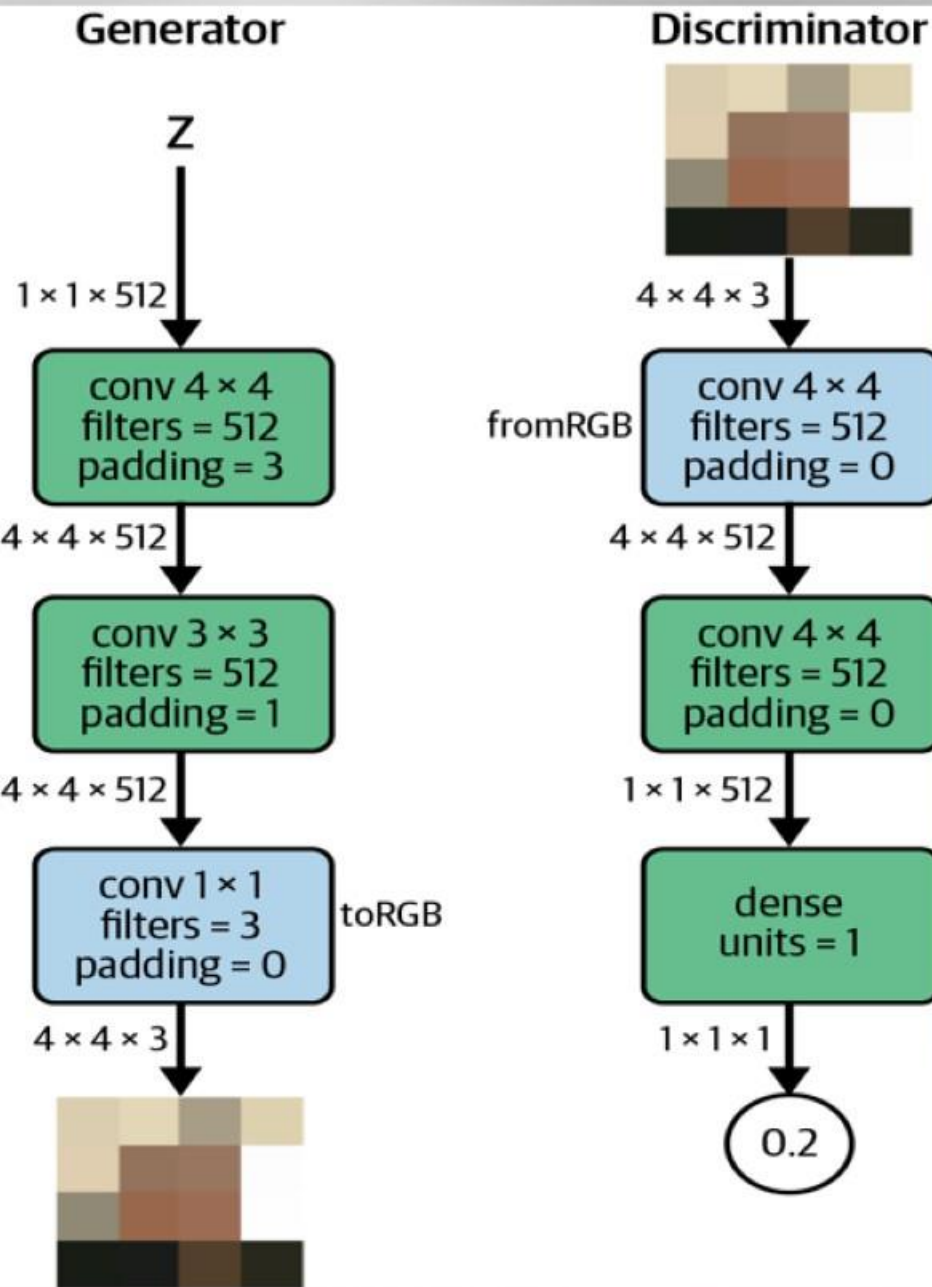
2 Gradual Refinement

By building on the knowledge gained from the low-resolution training, the generator can gradually refine its output, adding more detail and complexity as the resolution increases. This can lead to more stable and higher-quality results.

3 Efficient Training

Training on low-resolution images is computationally less expensive, allowing for faster training and experimentation. The progressive approach can make the overall training process more efficient.

The Generator and Discriminator Architectures



Generator Architecture

The generator architecture starts with a convolutional layer that converts the set of feature maps into an RGB image (toRGB). As the resolution increases, new upsampling and convolutional layers are appended to the existing network, with a residual connection to maintain the output from the existing trained toRGB layer.

Discriminator Architecture

The discriminator architecture starts with a convolutional layer that converts the RGB images into a set of feature maps (fromRGB). As the resolution increases, new convolutional layers are added to the existing network to handle the higher-resolution inputs.

Gradual Transition

The new layers in both the generator and discriminator are initially masked using a parameter α that is gradually increased from 0 to 1 throughout the transition phase. This allows for a smooth transition as the new layers take over.

The Training Process

Existing 4×4 generator

Transition

Stabilization

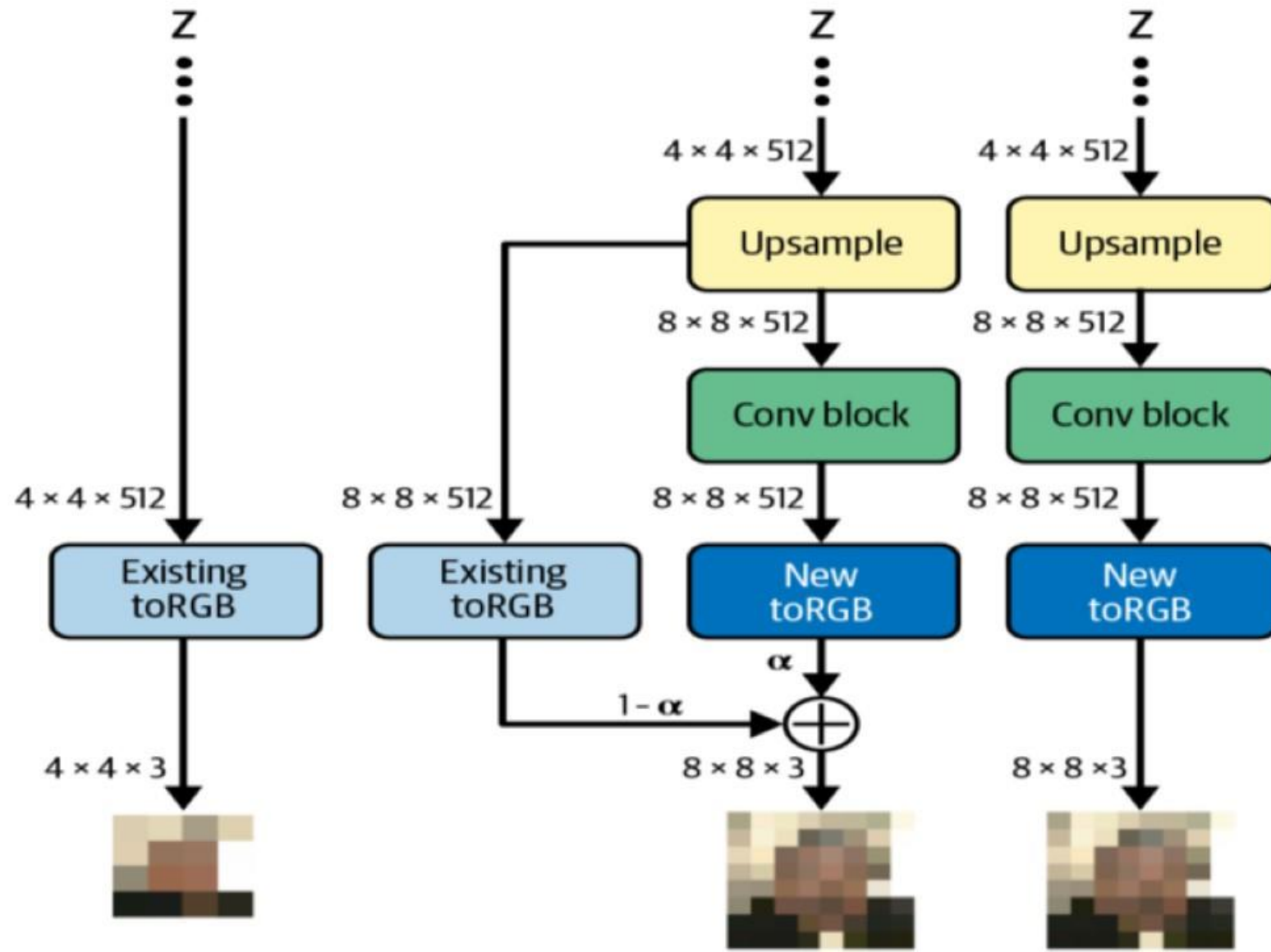


Figure 10-3. The ProGAN generator training process, expanding the network from 4×4 images to 8×8 (dotted lines represent the rest of the network, not shown)

The Training Process

Existing 4×4 discriminator

Transition

Stabilization

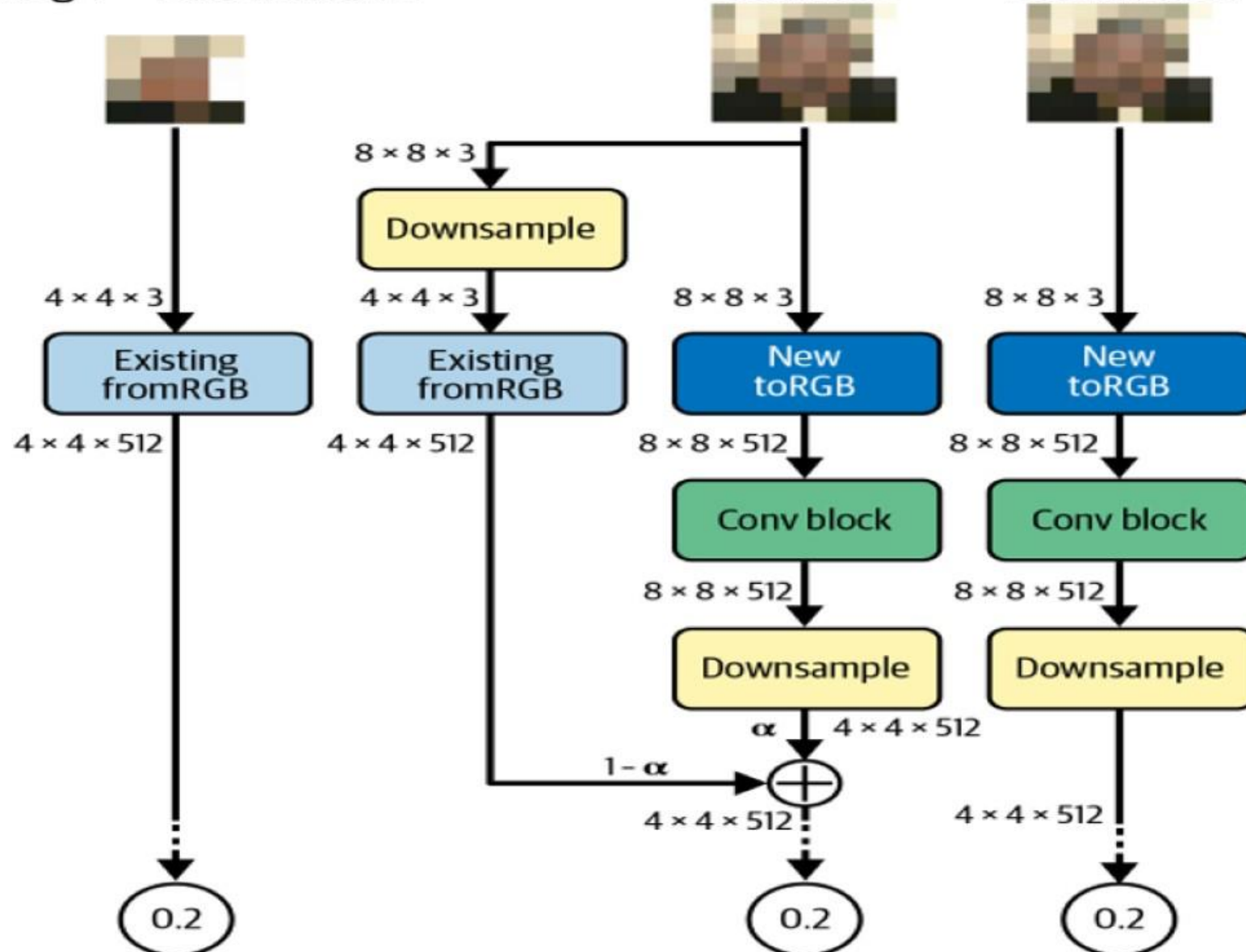


Figure 10-4. The ProGAN discriminator training process, expanding the network from 4×4 images to 8×8 (dotted lines represent the rest of the network, not shown)

The Training Process

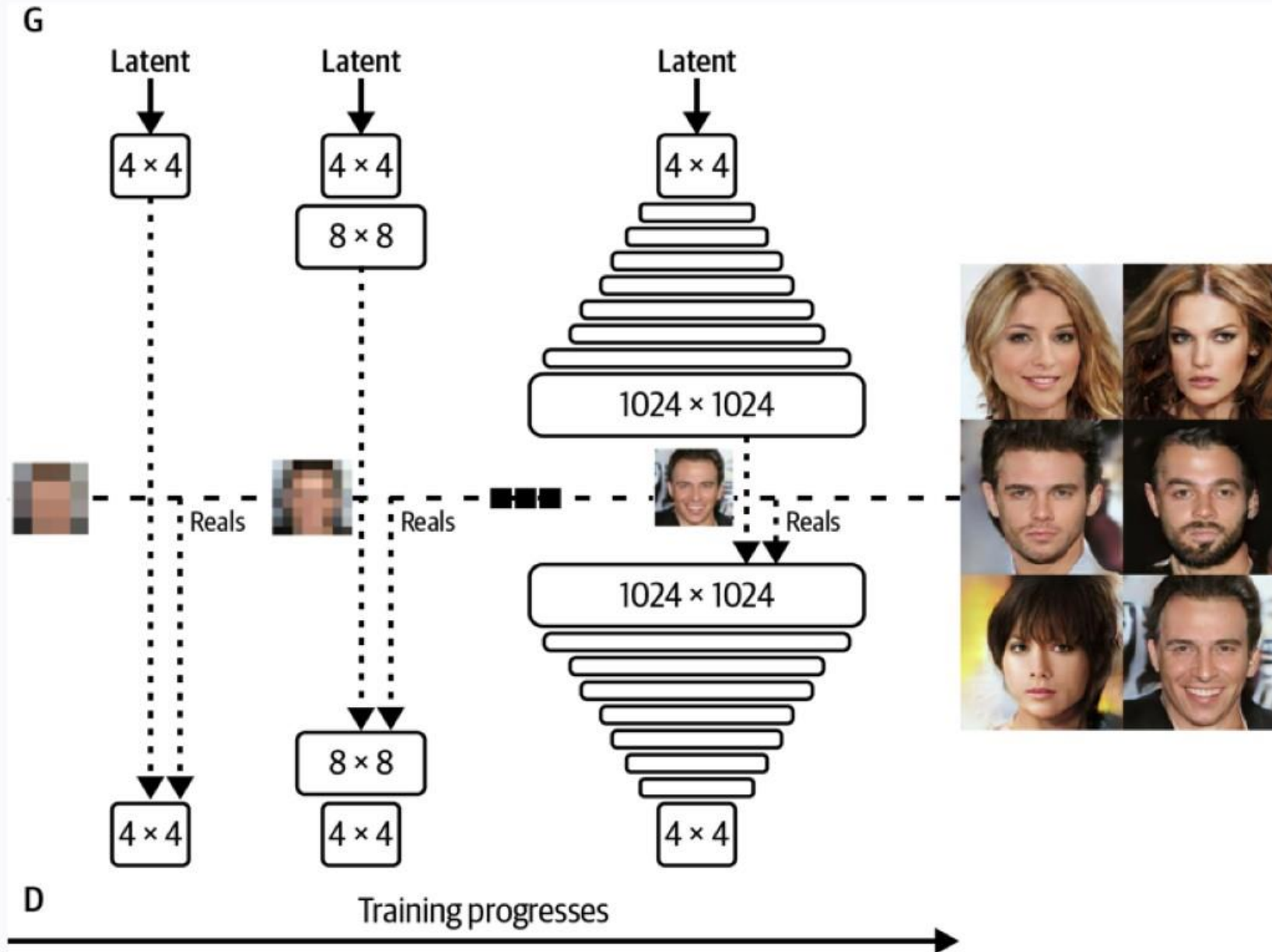


Figure 10-5. The ProGAN training mechanism, and some example generated faces.

The Training Process

1

Low Resolution

The training starts with 4x4 pixel images, where the generator and discriminator are trained to work with this low-resolution input.

2

Transition Phase

New layers are gradually added to the generator and discriminator, with the α parameter controlling the blending of the old and new layers.

3

High Resolution

Once the transition is complete, the network enters the stabilization phase, where it can fine-tune the output at the higher resolution without the old layers.

Achieving High-Quality Results



Improved Quality

The progressive training approach allows the generator to gradually learn the high-level structures and details, leading to higher-quality and more realistic output images.



Improved Stability

The gradual transition between resolutions helps to maintain the stability of the training process, avoiding sudden changes that could disrupt the learning.



Improved Efficiency

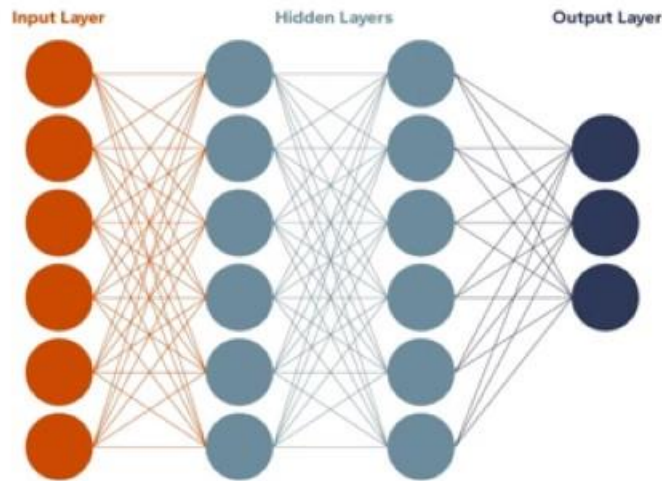
The lower-resolution training stages are computationally less expensive, allowing for faster experimentation and iteration during the overall training process.



Conclusion

The ProGAN paper introduces a novel approach to training GANs called "progressive training". By starting with low-resolution images and gradually increasing the resolution, the generator can learn the high-level structures more effectively, leading to higher-quality and more stable results. The gradual transition between resolutions also helps to maintain the stability of the training process, while the lower-resolution training stages are computationally less expensive, improving the overall efficiency of the training.

Minibatch Standard Deviation



The minibatch standard deviation layer is an extra layer in the discriminator that appends the standard deviation of the feature values, averaged across all pixels and across the minibatch as an additional (constant) feature.

Why Minibatch Standard Deviation?

Variety in Output

This layer helps to ensure the generator creates more variety in its output.

Discriminating Fake Batches

If variety is low across the minibatch, then the standard deviation will be small, and the discriminator can use this feature to distinguish the fake batches from the real batches!

Incentivizing the Generator

Therefore, the generator is incentivized to ensure it generates a similar amount of variety as is present in the real training data.

1

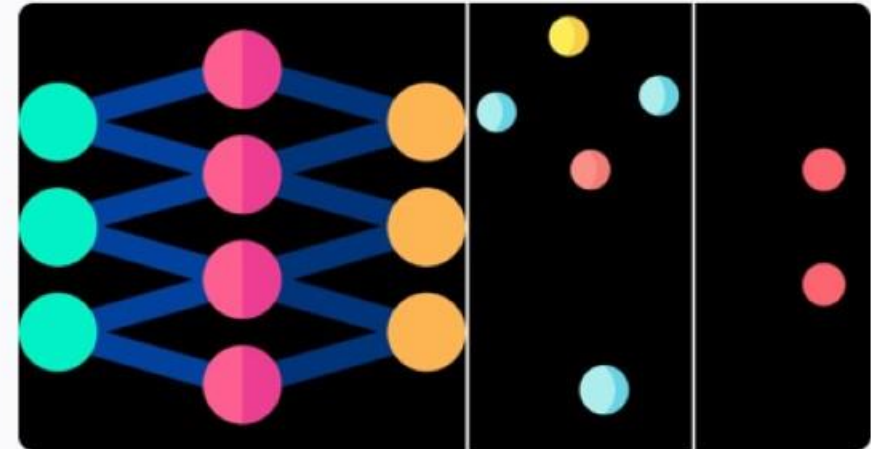
Real Data Variety

The generator learns to produce outputs with a similar level of variation as the real training data.

2

Improved Realism

This leads to more realistic and diverse generated outputs.



How it Works



1

Feature Values

The minibatch standard deviation layer calculates the standard deviation of the feature values.

2

Averaging

This standard deviation is averaged across all pixels and across the minibatch.

3

Additional Feature

The average standard deviation is then appended as an additional (constant) feature to the discriminator's input.

Impact on Discriminator

The discriminator can then use this additional feature to distinguish between real and fake batches.

Real Batches

Real batches will typically have a higher standard deviation, indicating greater variety.

Fake Batches

Fake batches, if they lack variety, will have a lower standard deviation.

Generator's Response

The generator, in order to fool the discriminator, learns to generate outputs with a similar level of variety as the real training data.

1

Low Variety

If the generator produces outputs with low variety, the discriminator will easily identify them as fake.

2

Increased Variety

The generator learns to increase the variety in its outputs to match the real data.

3

Improved Realism

This leads to more realistic and diverse generated outputs.



Benefits of Minibatch Standard Deviation

Increased Realism

Generates more realistic and diverse outputs.

Improved Performance

Helps the discriminator better distinguish between real and fake batches.

Enhanced Training

Incentivizes the generator to learn from the real data.

Applications

The minibatch standard deviation layer is commonly used in generative adversarial networks (GANs) for tasks such as image generation, text generation, and audio synthesis.



Image Generation

Generates realistic and diverse images.



Text Generation

Creates coherent and creative text.



Audio Synthesis

Generates high-quality audio samples.

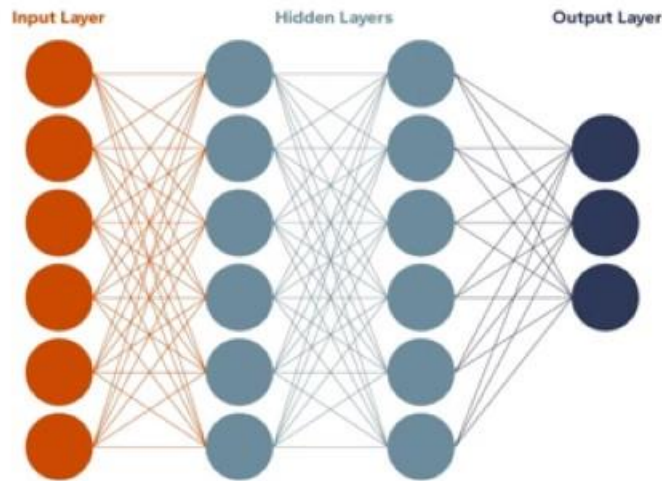


Conclusion

The minibatch standard deviation layer is a valuable technique for improving the performance and realism of generative adversarial networks. By encouraging the generator to produce outputs with a similar level of variety as the real training data, it helps to create more realistic and diverse generated outputs.



Equalized Learning Rates and Pixelwise Normalization



ProGAN, a groundbreaking generative adversarial network (GAN) architecture, employs several innovative techniques to achieve remarkable image generation quality. One of these techniques is the use of equalized learning rates, which address the issue of imbalanced training speeds between different layers. Another key feature is pixelwise normalization, which helps to stabilize the training process and prevent signal explosion.

Equalized Learning Rates: Addressing Training Imbalance

He Initialization

Weights in neural networks are typically initialized using methods like He initialization, where the standard deviation is inversely proportional to the square root of the number of inputs. This helps stabilize training.

Adam and RMSProp

Modern optimizers like Adam and RMSProp normalize gradient updates, making them independent of weight scale. However, this can lead to slower adjustments for layers with fewer inputs.

Equalized Learning Rates

ProGAN addresses this imbalance by using equalized learning rates. Weights are initialized with a simple Gaussian distribution, and normalization is applied dynamically during layer calls. This ensures that the optimizer treats all weights as having similar dynamic ranges, resulting in more balanced training.

Pixelwise Normalization: Stabilizing Feature Vectors

1

Feature Vector Normalization

Pixelwise normalization in ProGAN's generator normalizes the feature vector at each pixel to a unit length. This helps prevent the signal from becoming excessively large as it propagates through the network.

2

Preventing Signal Explosion

By normalizing the feature vectors, pixelwise normalization helps to stabilize the training process and prevent the signal from spiraling out of control. This is crucial for achieving high-quality image generation.

3

No Trainable Weights

The pixelwise normalization layer does not have any trainable weights. This means that it does not introduce additional parameters that need to be learned during training, keeping the model efficient and focused on learning the underlying data distribution.



He Initialization: A Closer Look

1

He Initialization

He initialization is a common technique for initializing weights in neural networks. It aims to prevent vanishing or exploding gradients during training by setting the initial weights to values that are neither too small nor too large.

2

Standard Deviation Scaling

In He initialization, the standard deviation of the weight distribution is scaled inversely proportional to the square root of the number of inputs to the layer. This ensures that layers with more inputs have smaller initial weights, which helps to stabilize training.

3

Improved Stability

He initialization has been shown to improve the stability of the training process, particularly for deep neural networks. It helps to prevent the network from getting stuck in local minima and promotes faster convergence.

Adam and RMSProp: Popular Optimizers

Adam

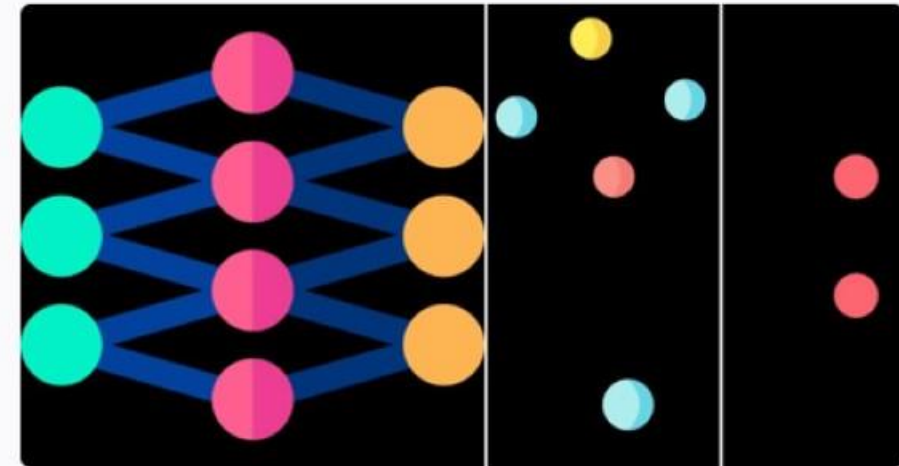
Adam (Adaptive Moment Estimation) is a popular optimization algorithm that combines the benefits of both momentum and RMSProp. It adapts the learning rate for each parameter based on its historical gradients.

RMSProp

RMSProp (Root Mean Square Propagation) is another widely used optimization algorithm that addresses the problem of vanishing or exploding gradients. It uses a moving average of squared gradients to adjust the learning rate for each parameter.

Gradient Normalization

Both Adam and RMSProp normalize the gradient update for each weight, making the update size independent of the weight's magnitude. This can lead to slower adjustments for weights with larger dynamic ranges.



Equalized Learning Rates: A Dynamic Approach



He Initialization	Equalized Learning Rates
Static scaling of weights at initialization	Dynamic scaling of weights during layer calls
Weights with larger dynamic ranges take longer to adjust	Optimizer treats all weights as having similar dynamic ranges
Can lead to training imbalance	Promotes balanced training speeds between layers

Pixelwise Normalization: A Closer Look



Feature Vector Normalization

Pixelwise normalization normalizes the feature vector at each pixel to a unit length. This helps to prevent the signal from becoming excessively large as it propagates through the network.



Stabilizing Training

By normalizing the feature vectors, pixelwise normalization helps to stabilize the training process and prevent the signal from spiraling out of control. This is crucial for achieving high-quality image generation.



No Trainable Weights

The pixelwise normalization layer does not have any trainable weights. This means that it does not introduce additional parameters that need to be learned during training, keeping the model efficient and focused on learning the underlying data distribution.



Benefits of Equalized Learning Rates

1

Balanced Training

Equalized learning rates ensure that all layers in the network train at a similar pace, preventing any single layer from dominating the training process.

2

Improved Stability

By preventing training imbalance, equalized learning rates contribute to a more stable training process, reducing the risk of the model getting stuck in local minima.

3

Faster Convergence

Balanced training speeds can lead to faster convergence, as the model can learn the underlying data distribution more efficiently.

4

Enhanced Performance

Overall, equalized learning rates can lead to improved performance in terms of both image quality and training efficiency.

Benefits of Pixelwise Normalization

1

Signal Control

Pixelwise normalization helps to prevent the signal from becoming excessively large as it propagates through the network, ensuring that the model remains stable and does not suffer from signal explosion.

2

Improved Gradient Flow

By normalizing the feature vectors, pixelwise normalization can improve the flow of gradients through the network, making it easier for the model to learn the underlying data distribution.

3

Enhanced Image Quality

Overall, pixelwise normalization contributes to improved image quality by helping to stabilize the training process and prevent signal explosion.



Thank You