# GANs in action

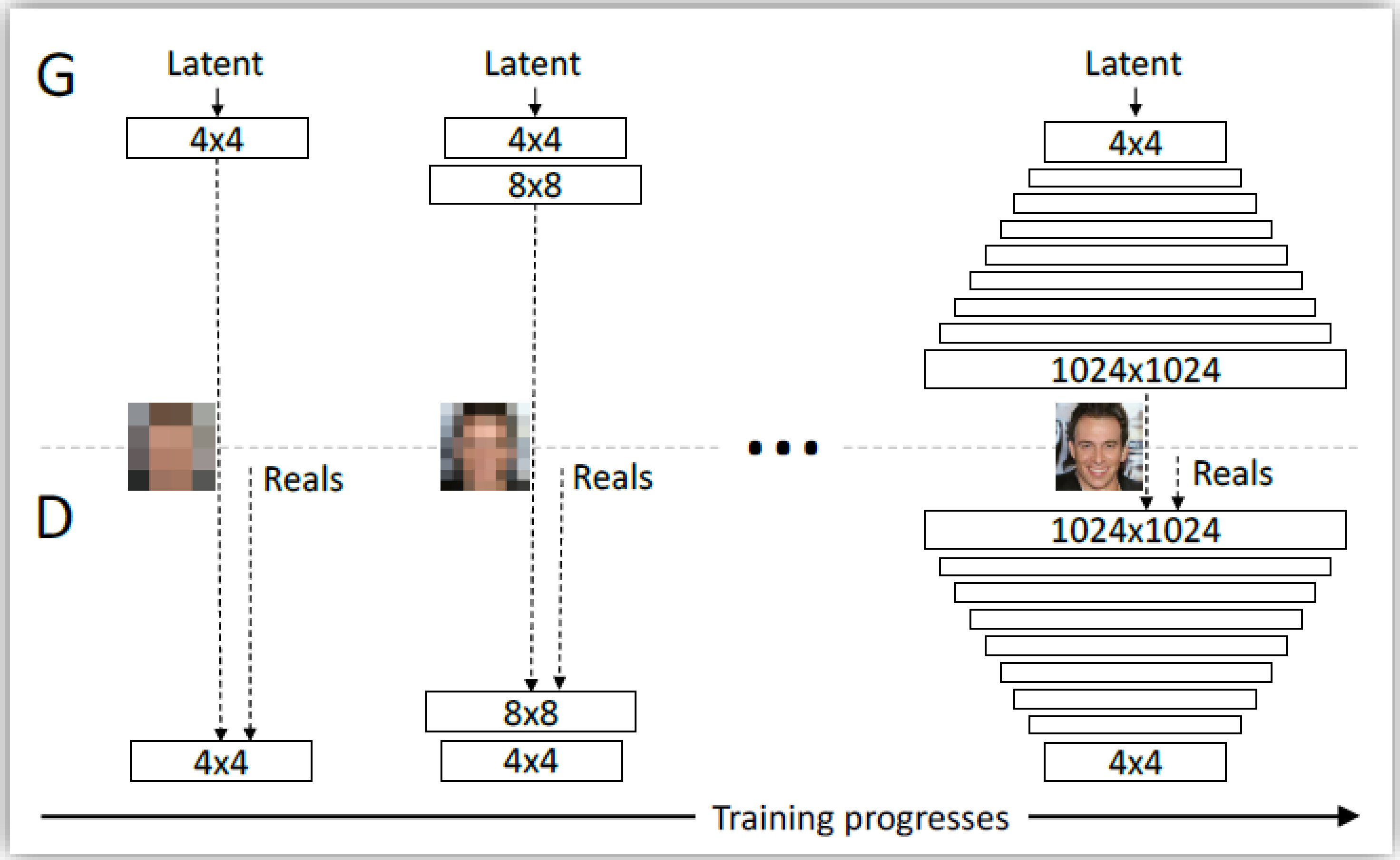
ProGAN

한병찬

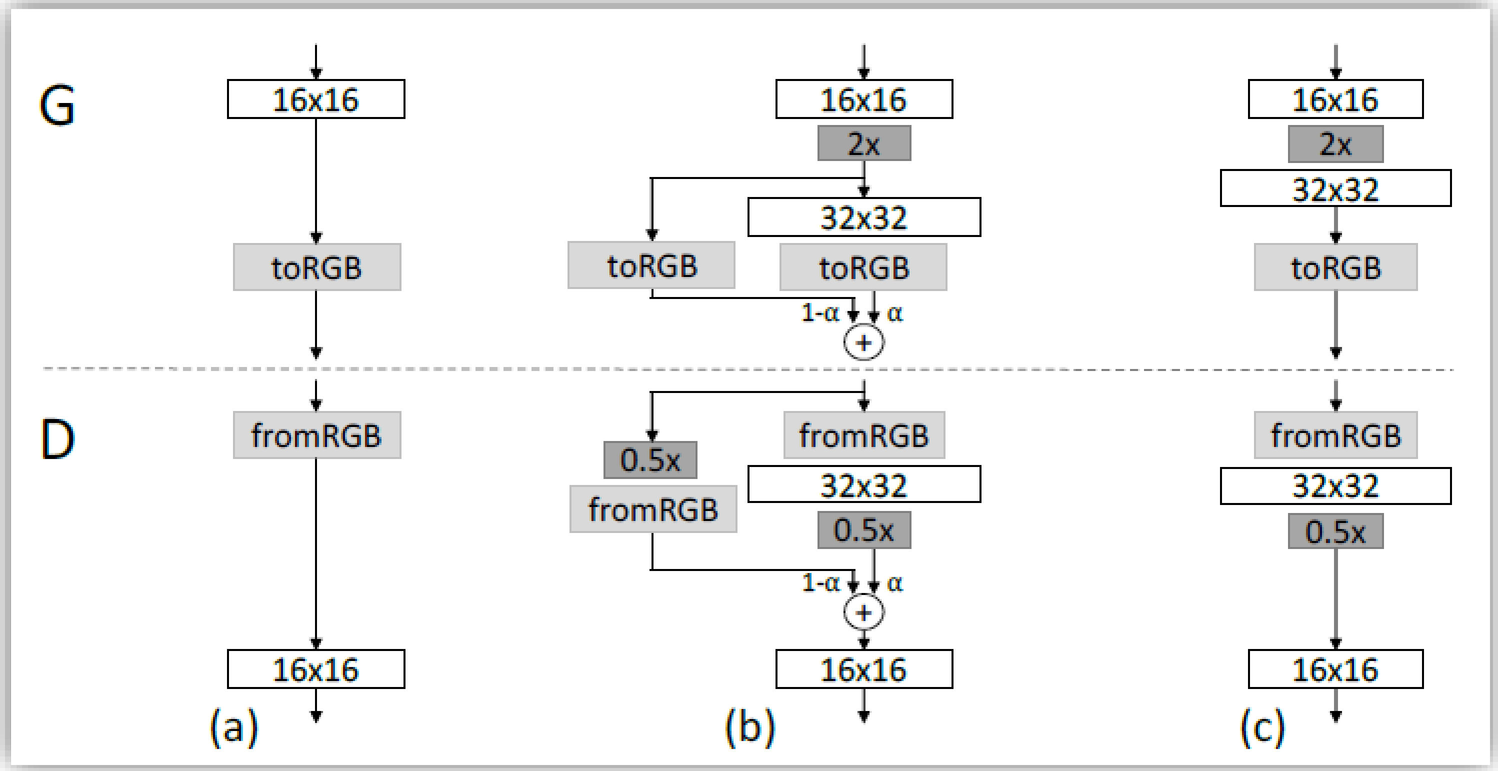
# Contents

1. Progressive structure
2. Minibatch std
3. Pixel Normalization
4. Equalized Learning Rate

# Progressive structure



# Progressive structure



# Progressive structure

| Generator | Act. | Output shape | Params |
|---|---|---|---|
| Latent vector | – | 512 × 1 × 1 | – |
| Conv 4 × 4 | LReLU | 512 × 4 × 4 | 4.2M |
| Conv 3 × 3 | LReLU | 512 × 4 × 4 | 2.4M |
| Upsample | – | 512 × 8 × 8 | – |
| Conv 3 × 3 | LReLU | 512 × 8 × 8 | 2.4M |
| Conv 3 × 3 | LReLU | 512 × 8 × 8 | 2.4M |
| Upsample | – | 512 × 16 × 16 | – |
| Conv 3 × 3 | LReLU | 512 × 16 × 16 | 2.4M |
| Conv 3 × 3 | LReLU | 512 × 16 × 16 | 2.4M |
| Upsample | – | 512 × 32 × 32 | – |
| Conv 3 × 3 | LReLU | 512 × 32 × 32 | 2.4M |
| Conv 3 × 3 | LReLU | 512 × 32 × 32 | 2.4M |
| Upsample | – | 512 × 64 × 64 | – |
| Conv 3 × 3 | LReLU | 256 × 64 × 64 | 1.2M |
| Conv 3 × 3 | LReLU | 256 × 64 × 64 | 590k |
| Upsample | – | 256 × 128 × 128 | – |
| Conv 3 × 3 | LReLU | 128 × 128 × 128 | 295k |
| Conv 3 × 3 | LReLU | 128 × 128 × 128 | 148k |
| Upsample | – | 128 × 256 × 256 | – |
| Conv 3 × 3 | LReLU | 64 × 256 × 256 | 74k |
| Conv 3 × 3 | LReLU | 64 × 256 × 256 | 37k |
| Upsample | – | 64 × 512 × 512 | – |
| Conv 3 × 3 | LReLU | 32 × 512 × 512 | 18k |
| Conv 3 × 3 | LReLU | 32 × 512 × 512 | 9.2k |
| Upsample | – | 32 × 1024 × 1024 | – |
| Conv 3 × 3 | LReLU | 16 × 1024 × 1024 | 4.6k |
| Conv 3 × 3 | LReLU | 16 × 1024 × 1024 | 2.3k |
| Conv 1 × 1 | linear | 3 × 1024 × 1024 | 51 |
| Total trainable parameters | | | **23.1M** |

| Discriminator | Act. | Output shape | Params |
|---|---|---|---|
| Input image | – | 3 × 1024 × 1024 | – |
| Conv 1 × 1 | LReLU | 16 × 1024 × 1024 | 64 |
| Conv 3 × 3 | LReLU | 16 × 1024 × 1024 | 2.3k |
| Conv 3 × 3 | LReLU | 32 × 1024 × 1024 | 4.6k |
| Downsample | – | 32 × 512 × 512 | – |
| Conv 3 × 3 | LReLU | 32 × 512 × 512 | 9.2k |
| Conv 3 × 3 | LReLU | 64 × 512 × 512 | 18k |
| Downsample | – | 64 × 256 × 256 | – |
| Conv 3 × 3 | LReLU | 64 × 256 × 256 | 37k |
| Conv 3 × 3 | LReLU | 128 × 256 × 256 | 74k |
| Downsample | – | 128 × 128 × 128 | – |
| Conv 3 × 3 | LReLU | 128 × 128 × 128 | 148k |
| Conv 3 × 3 | LReLU | 256 × 128 × 128 | 295k |
| Downsample | – | 256 × 64 × 64 | – |
| Conv 3 × 3 | LReLU | 256 × 64 × 64 | 590k |
| Conv 3 × 3 | LReLU | 512 × 64 × 64 | 1.2M |
| Downsample | – | 512 × 32 × 32 | – |
| Conv 3 × 3 | LReLU | 512 × 32 × 32 | 2.4M |
| Conv 3 × 3 | LReLU | 512 × 32 × 32 | 2.4M |
| Downsample | – | 512 × 16 × 16 | – |
| Conv 3 × 3 | LReLU | 512 × 16 × 16 | 2.4M |
| Conv 3 × 3 | LReLU | 512 × 16 × 16 | 2.4M |
| Downsample | – | 512 × 8 × 8 | – |
| Conv 3 × 3 | LReLU | 512 × 8 × 8 | 2.4M |
| Conv 3 × 3 | LReLU | 512 × 8 × 8 | 2.4M |
| Downsample | – | 512 × 4 × 4 | – |
| Minibatch stddev | – | 513 × 4 × 4 | – |
| Conv 3 × 3 | LReLU | 512 × 4 × 4 | 2.4M |
| Conv 4 × 4 | LReLU | 512 × 1 × 1 | 4.2M |
| Fully-connected | linear | 1 × 1 × 1 | 513 |
| Total trainable parameters | | | **23.1M** |

# Pixel Normalization

$$b_{x,y} = a_{x,y} / \sqrt{\frac{1}{N}\sum_{j=0}^{N-1}(a_{x,y}^{j})^2 + \epsilon}, \text{ where } \epsilon = 10^{-8}$$

```
class PixelNorm(nn.Module):
    def __init__(self):
        super(PixelNorm, self).__init__()
        self.epsilon = 1e-8

    def forward(self, x):
        return x / torch.sqrt(torch.mean(x ** 2, dim=1, keepdim=True) + self.epsilon)
```

# Equalized Learning Rate

$$W \sim N(0, Var(W))$$

$$Var(W) = \sqrt{\frac{2}{n_{in}}} \longrightarrow \sqrt{\frac{2}{k * k * n_{in}}}$$

```
self.scale = (2 / (in_channels * (kernel_size ** 2))) ** 0.5
```

# ProGAN