ISABELLA CAROLINA FERREIRA MENDONÇA

REDES NEURAIS CONVOLUCIONAIS GERANDO ARTE: UMA APLICAÇÃO EM OBRAS DE JÊ AMÉRICO

Trabalho de Conclusão de Curso submetido à disciplina "MAC0499 - Trabalho de Formatura Supervisionado", sob a orientação do Prof. Alfredo Goldman

São Paulo, janeiro de 2019

Agradecimentos

Para Marisa, Anésio, Gabriela, Maria e Janice por todo suporte durante esses anos.

Para Álvaro, por todo auxílio e companheirismo incondicional.

Para todos os amigos pela companhia nas horas boas e ruins.

Para Alfredo, por todo o acompanhamento na execução desse trabalho.

Para Jê, pela parceria, paciência e boas conversas.

Muito obrigada a todos, não estaria aqui senão por vocês.

may came home with a smooth round stone
as small as a world and as large as alone

for whatever we lose (like a you or a me)
it's always ourselves we find in the sea
(E. E. Cummings)

Resumo

Este trabalho pretende explorar o uso de redes neurais convolucionais com obras de Jê Américo para gerar novas obras, estas com características e estéticas próprias, auxiliando o autor a explorar mais uma forma de criação.

As redes neurais convolucionais tem se mostrado muito efetivas nas áreas de classificação e reconhecimento de imagens por serem capaz de lidar com enormes entradas de dados, além de terem sido recentemente utilizadas para geração de imagens originais com redes adversárias generativas e redes adversárias generativas convolucionais profundas.

DCGANs (sigla do inglês para *Deep convolutional generative adversarial networks*, cujo objetivo é escalar o uso de redes adversárias generativas, possuem algumas mudanças arquiteturais em relação às CNNs.

Com execução de três diferentes experimentos utilizando DCGANs, conseguimos entender melhor o funcionamento de tal tecnologia quando aplicado à obras de arte e sua capacidade de gerar imagens originais a partir das mesmas.

Palavras-chave: redes neurais convolucionais; arte; Jê Américo.

Lista de ilustrações

Figura 1 – Sem título 15
Figura 2 – Obra da série Móbiles 16
Figura 3 – Rede neural feed foward 19
Figura 4 – Rede neural recorrente 20
Figura 5 – Rede neural convolucional 21
Figura 6 – Rede neural convolucional 22
Figura 7 – Rede adversária generativa 24
Figura 8 – “calçada: vermelho” 31
Figura 9 – Resultado obtido após treinamento alfa 32
Figura 10 – Fragmentos das obras de Jê Américo utilizados no primeiro experimento 33
Figura 11 – Tempo de execução do primeiro treinamento 34
Figura 12 – Histograma do resultado da função de custo do gerador na primeira e última iteração do treinamento durante o experimento 1 34
Figura 13 – Histograma do resultado da função de custo do discriminador na primeira e última iteração do treinamento durante o experimento 1 35
Figura 14 – Treinamento da rede no primeiro experimento após 25 iterações 36
Figura 15 – Resultado obtido após primeiro treinamento 36
Figura 16 – Obras de Jê Américo utilizados no segundo experimento 37
Figura 17 – Treinamento da rede no segundo experimento após 1 iteração 38
Figura 18 – Tempo de execução do primeiro treinamento 38
Figura 19 – Histograma de erro do gerador na primeira e última iteração do treinamento durante o experimento 3 39
Figura 20 – Histograma de erro do discriminador na primeira e última iteração do treinamento durante o experimento 3 39
Figura 21 – Treinamento da rede no segundo experimento após 7 iterações 40
Figura 22 – Resultado obtido após segundo treinamento 40

Sumário

1 INTRODUÇÃO . . . 13
1.1 Objetivos . . . 13
1.2 Organização do trabalho . . . 13

2 A OBRA DE JÊ AMÉRICO . . . 15

3 TECNOLOGIAS . . . 19
3.1 Conceitos . . . 20
3.1.1 Redes neurais convolucionais . . . 20
3.1.1.1 Arquitetura . . . 21
3.1.1.1.1 Convolução . . . 22
3.1.1.1.2 Não linearidade . . . 22
3.1.1.1.3 Agrupamento (*Pooling*) . . . 22
3.1.1.1.4 Camadas totalmente conectadas (*Fully connected layer*) . . . 23
3.1.2 Redes adversárias generativas . . . 23
3.1.3 Redes adversárias generativas convolucionais profundas . . . 25
3.2 Aplicação em estudos relacionados . . . 26

4 RESULTADOS E DISCUSSÃO . . . 31
4.1 Experimento alfa (A1) . . . 31
4.1.1 Tratamento das obras . . . 32
4.1.2 Resultados obtidos . . . 32
4.2 Primeiro experimento (E1) . . . 33
4.2.1 Tratamento das obras . . . 33
4.2.2 Realização do experimento . . . 34
4.2.3 Resultados obtidos . . . 36
4.3 Segundo experimento (E2) . . . 37
4.3.1 Tratamento das obras . . . 37
4.3.2 Realização do experimento . . . 38
4.3.3 Resultados obtidos . . . 40

5 CONCLUSÃO . . . 43

A INFORMAÇÕES COMPLEMENTARES . . . 45
A.1 Experimento alfa (A1) . . . 45
A.1.1 Resultados obtidos . . . 45
A.2 Primeiro experimento (E1) . . . 49

A.2.1 Histogramas da função de custo do gerador 49
A.2.2 Histogramas da função de custo do discriminador 54
A.2.3 Resultados obtidos 58
A.3 Segundo experimento (E2) 63
A.3.1 Histogramas da função de custo do gerador 63
A.3.2 Histogramas da função de custo do discriminador 67
A.3.3 Resultados obtidos 71

REFERÊNCIAS 77

1 Introdução

Criatividade é manifestada em diversas atividades humanas, incluindo descobertas científicas, inovações tecnológicas e criação artística.

Por ser uma habilidade intrínseca à humanidade, desde o início da computação houve a dúvida se uma máquina seria capaz de gerar artefatos originais. Durante o desenvolvimento do primeiro computador, Ada Lovelace declarou que este "não possuía quaisquer pretensões de originar algo" e que poderia "fazer qualquer coisa que soubéssemos como ensiná-lo a realizar".

Através do estudo de neurociências, psicologia da criatividade e simulações, conseguimos começar a compreender o misterioso processo criativo [1]. Alguns escritores, como Boden [2], propõem que a criatividade é resultado de uma combinação de representações. Simonton [3] propõe que a criatividade seguiria um processo Darwiniano, onde novas ideias são geradas por variações e seleção.

Inspirado por propostas como essas, pode-se compreender melhor o processo criativo humano para que, assim, pudéssemos desenvolver algoritmos computacionais capazes de entendê-lo e replicá-lo. Em termos de criação artística, diversas pesquisas [4–6] foram desenvolvidas com o uso de redes neurais, especificamente, redes neurais convolucionais.

Redes neurais convolucionais consistem em camadas de pequenas unidades computacionais que processam informação visual hierarquicamente, onde cada camada processa uma certa característica da imagem de entrada sendo capaz de gerar uma nova imagem que combina as características extraídas anteriormente. [6]

1.1 Objetivos

A proposta deste trabalho é explorar o uso de redes neurais convolucionais com obras de Jê Américo para gerar novas obras, estas com características e estéticas próprias, auxiliando o autor a explorar mais uma forma de criação. Com participação do artista durante o processo, esperamos avaliar a coerência e criatividade do material produzido, sendo de extrema importância para entendermos as necessidades a serem desenvolvidas e exploradas.

1.2 Organização do trabalho

O presente trabalho divide-se em cinco capítulos:

- o primeiro capítulo (1) oferece uma descrição do problema e o objetivo do trabalho;
- o segundo capítulo (2) propõe uma reflexão sobre a produção e obras de Jê Américo;
- o terceiro capítulo (3) discute as tecnologias possíveis para realização do trabalho e sua utilização em trabalhos relacionados;
- o quarto capítulo (4) demonstra os resultados obtidos a partir da implementação proposta;
- o quinto capítulo (5) apresenta os aprendizados do presente trabalho com posterior análise dos experimentos apresentados neste trabalho.

2 A obra de Jê Américo

Formado em Arquitetura e Urbanismo pela Universidade de São Paulo em 1994, o artista plástico Jê Américo iniciou sua carreira de desenhista técnico em escritórios de arquitetura na segunda metade da década de 70, nos quais elaborava seus desenhos com canetas nanquins, esquadros e régua paralela.

Após a sugestão de um amigo artista plástico em 2008, Jê elaborou sua primeira série de obras que retratavam mapas a partir de seus contornos e já enunciavam o uso das linhas, marca importante nos seus trabalhos posteriores.

Figura 1 – Obra sem título

Uma das principais características de suas obras subsequentes, a ausência de linhas de contorno tornam as formas inefáveis e permitem diferentes camadas de leitura, cada vez mais profundas, de sua composição.

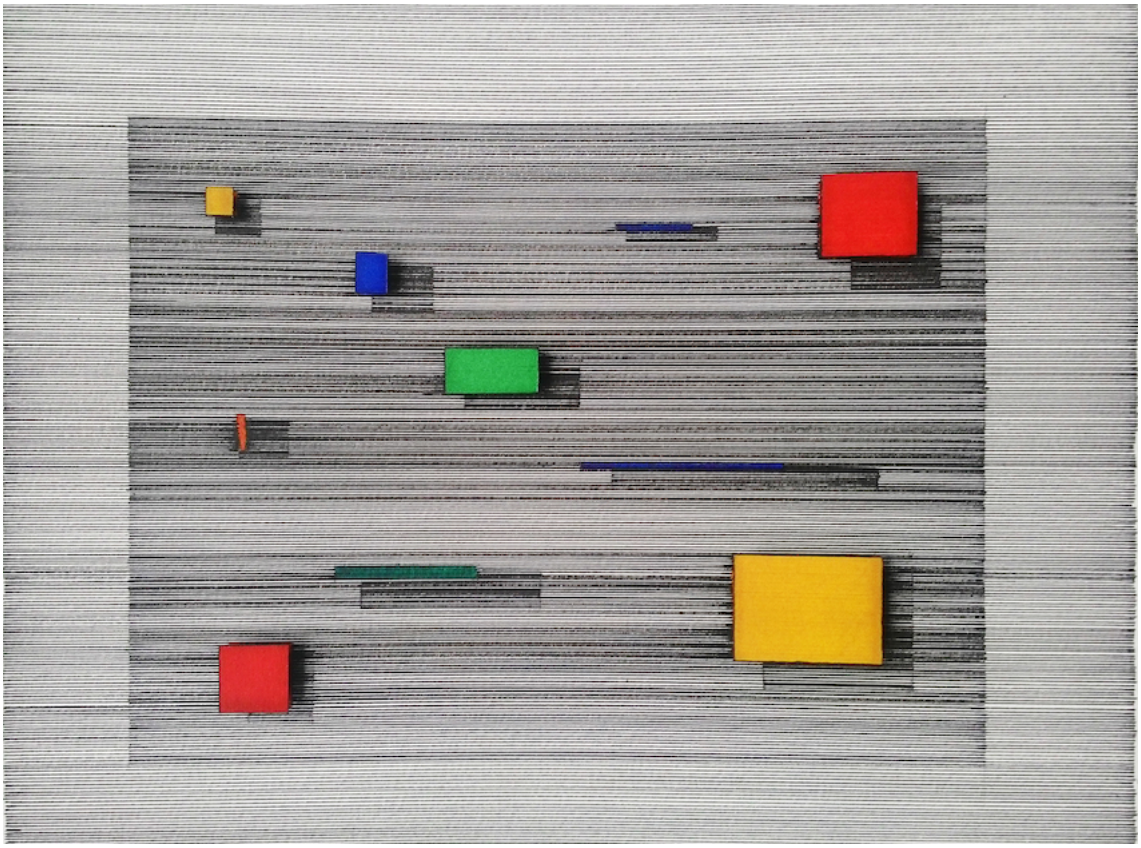


Figura 2 – Obra da série Móbiles

Paulo Gallina, crítico de arte, define os desenhos de Américo como construção de um "pensamento de pintura", que definem espaços por sua relação de complementaridade ou negação cromática.

> *"Uma das coisas que percebi em minhas reflexões sobre a produção artística, é que o artista confia plenamente em suas intuições. Não há certeza, mas você acredita que por ali existe um caminho que vai dar em algum lugar. Minha produção inicial se deu exatamente dessa forma. Não pesquisei ou fiz algum estudo prévio, simplesmente comecei a produzir. Depois de um tempo, você olha o que fez e começa a identificar afinidades estéticas com obras que você viu. No meu caso, encontrei minha filiação no movimento concreto, mas isso foi uma coisa que eu descobri a posteriori. Eu não pensei e fiz, fiz e depois eu vi que eu estava dentro deste contexto, da arte concreta."*
>
> — Jê Américo

Segundo o artista, na composição e elaboração de seu trabalho não se dá com uma análise e concepção a *priori* e sim a *posteriori*, onde pode-se entendê-lo após sua execução. Para Américo [7], sua produção artística constrói-se majoritariamente ao confiar

plenamente em suas intuições, onde não há pesquisa ou estudo prévio, mas que cria afinidade estética com o movimento concreto.

Jê comenta que é leitor voraz de textos sobre divulgação científica, especialmente sobre teoria da relatividade e mecânica quântica há aproximadamente 25 anos. Esse interesse o levou a explorar outras áreas do conhecimento, como a matemática, a linguagem, biologia, entre outros, entretanto, se viu particularmente interessado pelo campo da ciência da computação, principalmente por inteligência artificial.

Ao ser questionado pela sua motivação em participar do presente trabalho, o artista comenta que leu o relato de um experimento similar com composições musicais e questionou-se sobre o resultado que poderia obter a partir de sua produção artística.

> *"O historiador israelense Yuval Harari em seu livro "Homo Deus" descreve um experimento onde um professor de música desenvolve um algoritmo para "aprender" o padrão compositivo de peças para coral do alemão Johan Sebastian Bach. Posteriormente, as composições elaboradas pelo programa foram submetidas a audiência, que, desconhecendo a forma como haviam sido elaboradas, se emocionaram profundamente com as músicas. Ao ler este relato, me perguntei qual seria a opinião do compositor barroco se tivesse oportunidade de ouvir as obras compostas pelo programa, mas infelizmente não teremos oportunidade de obtermos uma resposta. Sou artista plástico e me senti motivado a participar de um experimento similar. As notas musicais seriam substituídas por pixels de imagens das obras que compus ao longo dos últimos dez anos."*
>
> — Jê Américo

Em relação à sua expectativa para com os resultados do trabalho, Américo comenta que levantava perguntas filosóficas sobre os limites da própria natureza humana, questionando-se se, ao ver as imagens compostas por meio do algoritmo, pensaria que poderiam ser compostas por ele e se seria possível que o código capturasse algo sobre sua essência criativa.

Por utilizar diversas formas geométricas básicas, a obra de Jê Américo soou interessante para uso no projeto, pois possibilitaria entender se, ao utilizar composições sem formas rebuscadas e muito detalhes que poderiam ser encontradas em obras de movimentos que prezam pelo realismo, o computador seria capaz de replicá-las.

3 Tecnologias

Redes neurais artificiais são definidas como um sistema computacional cujo modelo inspira-se na estrutura neural humana conectados em diferentes camadas.

Essencialmente, as redes neurais realizam diversas operações matemáticas nos dados de entrada utilizando alguns parâmetros, os ajusta utilizando o viés aprendido e, então, os normaliza.

Uma das principais funcionalidades das redes neurais é o aprendizado iterativo no qual, a partir de uma amostra inicial, a rede é capaz de definir seus parâmetros até que consiga gerar um resultado satisfatório.

Além disso, as ANNs (sigla do inglês para *Artificial Neural Networks*) são capazes de lidar com altos volumes de dados, muitas vezes com ruídos que poderiam interferir em seu desempenho.

As redes neurais *feed foward* são um dos tipos de redes neurais artificias mais simples pois os dados são processados em apenas uma direção. Apesar de as informações serem apenas propagados unidirecionalmente, cada camada é, geralmente, uma unidade de processamento. Uma das aplicações frequentes deste tipo de rede ocorre na área de visão computacional e reconhecimento de fala.

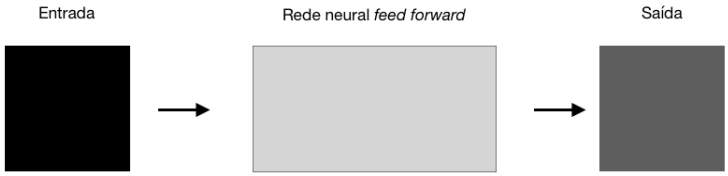


Figura 3 – Estrutura de uma rede neural *feed foward*.

Diferentemente das redes neurais *feed foward*, as redes neurais recorrentes utilizam laços de recorrência e têm como propósito analisar os dados para prever o resultado de suas camadas.

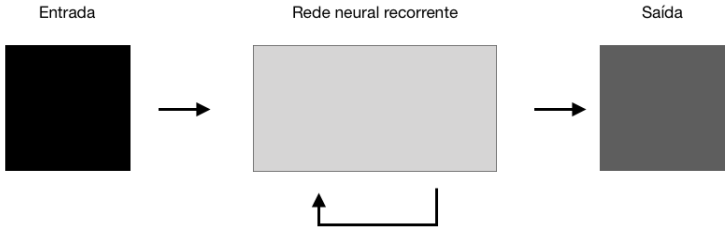


Figura 4 – Estrutura de uma rede neural recorrente.

Neste modelo, cada camada é "alimentada"com a saída das camadas anteriores e, em cada etapa, os dados são reprocessados e otimizados para realizar uma melhor predição do resultado a ser gerado na camada posterior. Caso a predição não seja boa, alguns ajustes são realizados para que, gradualmente, a rede se torne melhor.

Outro tipo de redes neurais bastante utilizadas na área de processamento de imagens durante os últimos anos, as redes neurais convolucionais apresentam estrutura similar às redes neurais *feed foward*. Com uso de convoluções, que preservam a relação espacial entre os pixels, as redes neurais convolucionais possuem alta capacidade de classificação com alto número de parâmetros que devem ser aprendidos por meio de treinamento. [8]

3.1 Conceitos

3.1.1 Redes neurais convolucionais

CNNs (sigla do inglês para *Convolutional Neural Networks*) consistem de pequenas camadas de processamento de imagem visual hierarquicamente.

Cada uma dessas camadas funciona como um filtro que, ao processar uma imagem de um arquivo de entrada, extrai uma certa característica da imagem original. Assim, é possível gerar uma imagem de saída que consiste em um mapa de diferentes atributos filtrados dessas características. [6]

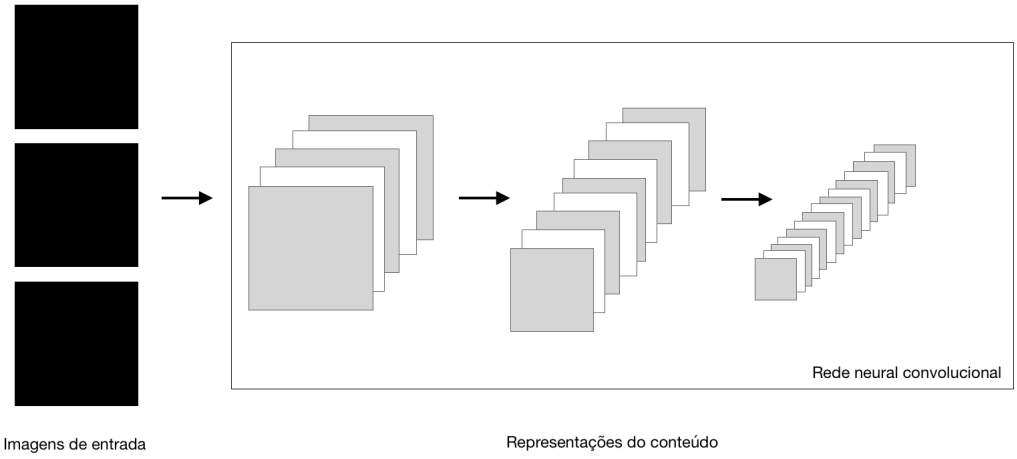


Figura 5 – Imagens de entrada são representadas como um conjunto de filtros de mapas de atributos a cada estágio de processamento da rede neural convolucional.

As redes neurais convolucionais tem se mostrado muito efetivas nas áreas de classificação e reconhecimento de imagens por serem capaz de lidar com enormes entradas de dados.

3.1.1.1 Arquitetura

As redes neurais convolucionais realizam quatro operações principais em cada uma de suas camadas:

1. Convolução

2. Não linearidade

3. *Pooling*

4. Camadas totalmente conectadas

Em sua arquitetura, após receber uma imagem de entrada, a rede executa diversas operações de convolução (que podem ou não serem acompanhadas por uma camada não linear) e *pooling*, seguidas por um número de camadas totalmente conectadas.

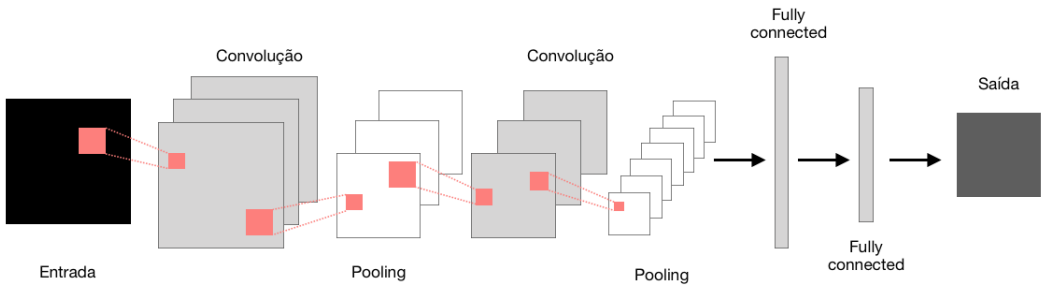


Figura 6 – Arquitetura genérica de uma rede neural convolucional.

3.1.1.1.1 Convolução

Uma das principais camadas de uma rede neural convolucional é a camada de convolução, cujo objetivo é extrair atributos da imagem de entrada preservando a relação espacial entre os *pixels*.

Os filtros de convolução aplicados nesta camada são operações matemáticas cuja definição é determinada pela rede durante o processo de treinamento. Ao agrupá-los em diferentes etapas do processo, a rede neural se torna capaz de reconhecer informações mais abstratas e profundas de uma imagem por meio de um processo de aprendizado hierárquico.

3.1.1.1.2 Não linearidade

A camada de não linearidade, geralmente utilizada após a camada de convolução, tem como objetivo introduzir um componente não linear em uma rede neural convolucional.

Uma das operações mais utilizadas nessa camada, a função de ativação ReLU (sigla do inglês para *Rectified Linear Unit*) substitui os pixels de valor negativo por zero e pode ser escrita por $f(x) = \max(0,x)$.

Apesar de ser uma das funções mais utilizadas na camada de não linearidade de uma rede neural convolucional e obter resultados satisfatórios para a maior parte dos casos de uso, a função de ativação ReLU pode afetar o resultado final por não mapear os valores negativos apropriadamente.

Outras funções de ativação como Tanh (Tangente Hiperbólica) e Sigmoide também podem ser utilizadas nesta camada.

3.1.1.1.3 Agrupamento (*Pooling*)

Com objetivo de reduzir o número de parâmetros, a camada de agrupamento (*pooling*) reduz a dimensionalidade de um mapa de atributos, reduzindo também o custo

computacional e controlando a perda de generalização (*overfitting*).

Maxpooling é o tipo de *pooling* usado frequentemente nesta camada e extrai o maior valor de uma janela de *pooling*.

3.1.1.1.4 Camadas totalmente conectadas (*Fully connected layer*)

Utilizando perceptron multi camadas (ou MLP, sigla para *Multi Layer Perceptron*), as camadas totalmente conectadas são compostas por neurônios ligados entre si.

Ao receber como entrada o mapa de atributos gerados pelas camadas anteriores com representação de alto nível das características da imagem de entrada, as camadas totalmente conectadas são capazes de classificar tais dados em diversas classes ou também aprender combinações não lineares destes atributos.

3.1.2 Redes adversárias generativas

Conceito introduzido por Ian Goodfellow et al. [9], as redes adversárias generativas (também conhecidas como GANs, sigla para *Generative Adversarial Networks*) usam redes neurais convolucionais para gerar dados originais a partir de dos dados de entrada inseridos.

O modelo consiste do uso de duas redes neurais: um modelo gerador e um discriminador, que aprende a determinar se uma amostra é proveniente da amostra real ou da amostra gerada. Podemos considerá-los, metaforicamente, um fraudador e um policial, respectivamente.

Nessa arquitetura, o papel do policial é analisar uma imagem e determinar se é uma imagem original ou "falsa". Já o fraudador é responsável por criar conteúdo realista o suficiente para que o policial seja "enganado". Assim, as duas redes passam a usar a saída da outra para aperfeiçoarem seus modelos e aprimorarem os resultados até que as falsificações sejam convincentes.

O processo inicia-se a partir de um ruído z, obtido a partir de uma distribuição normal ou uniforme, que é utilizado em um gerador G para criar uma imagem x, onde $x = G(z)$.

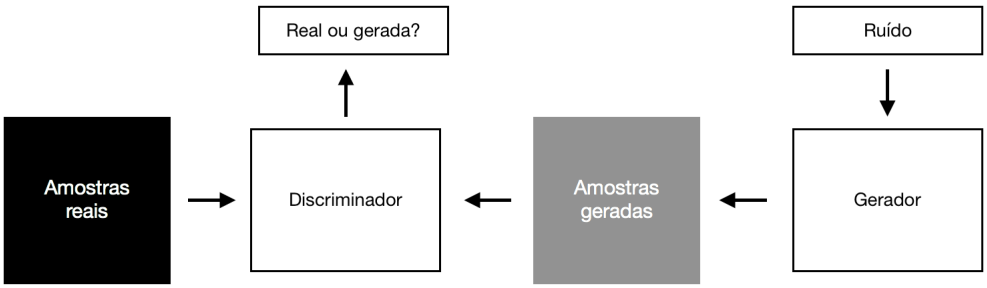


Figura 7 – Estrutura geral de uma rede adversária generativa.

A partir disso, o discriminador analisa as imagens de entrada e as imagens geradas separadamente e determina a probabilidade da imagem ser real. Seu objetivo é de maximizar a chance de reconhecer imagens reais como reais e imagens geradas como falsas. Para calcularmos a função de custo utilizamos entropia cruzada, obtendo:

$$\max_D V(D) = \underbrace{\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)]}_{\text{reconhece imagens reais melhor}} + \underbrace{\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]}_{\text{reconhece imagens geradas melhor}} \tag{3.1}$$

Para o gerador, o objetivo é gerar imagens com o maior valor possível para $D(x)$:

$$\min_G V(G) = \underbrace{\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))]}_{\text{otimiza G para enganar o discriminador}} \tag{3.2}$$

Assim, pode-se dizer que GANs são um jogo *minimax* onde G quer minimizar V enquanto D quer maximizá-lo.

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.3}$$

O pseudo código proposto em [9] demonstra como ocorre o treinamento da rede neural adversária generativa, utilizando $k = 1$:

Algoritmo 1 Treinamento de rede neural adversária generativa utilizando *mini-batch* gradiente descendente estocástico

1: **for** número de iterações de treinamento **do**
2: **for** k passos **do**
3: Amostra do *mini-batch* de m amostras de ruído $z^{(1)}, \ldots, z^{(m)}$ de $p_g(z)$
4: Amostra do *mini-batch* de m exemplos $x^{(1)}, \ldots, x^{(m)}$ de $p_{data}(x)$
5: Atualiza o discriminador ascendendo seu gradiente estocástico:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(x^{(i)}\right)\right)\right) \right]$$

6: Amostra do *mini-batch* de m amostras de ruído $z^{(1)}, \ldots, z^{(m)}$ de $p_g(z)$
7: Atualiza o gerador descendendo seu gradiente estocástico:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(x^{(i)}\right)\right)\right)$$

3.1.3 Redes adversárias generativas convolucionais profundas

DCGANs (sigla do inglês para *Deep convolutional generative adversarial networks*) foram introduzidas por Chintala et al. [10]. Sua arquitetura é uma versão modificada da arquitetura utilizada nas redes adversárias generativas propostas por Ian Goodfellow et al. [9], já que estas se apresentam bastante instáveis durante seu treinamento, além de produzirem imagens com ruídos e muitas vezes incompreensíveis.

Com objetivo de escalar o uso das redes adversárias generativas, as redes adversárias generativas convolucionais profundas possuem três principais mudanças arquiteturais em relação às CNNs.

A primeira modificação proposta é a substituição das funções de *pooling* por *strided convolutions*, que permitem que a rede neural aprenda sua própria sua taxa de redução de escala espacial. Tal substituição é realizada em ambos discriminador e gerador.

Strided convolution é um tipo de convolução que desloca o filtro (*kernel*) com um passo (*stride*) fixo, usualmente maior que um. Com seu uso, o número de unidades nas camadas convolucionais é reduzido, diminuindo o tempo de treinamento e memória utilizada. [11]

A eliminação de camadas completamente conectados é a segunda alteração arquitetural proposta, substituindo-as por convoluções, como *global average pooling*, que aumentam a estabilidade do modelo.

Utilizando transformações lineares dos mapas de atributos vetorizados, *global average pooling* são mais vantajosas em relação às *fully connected layers* por serem nativas à estrutura convolucional, além de não utilizarem parâmetros de otimização. [12]

Como terceira modificação proposta, o uso de *batch normalization* visa estabilizar o aprendizado da rede ao normalizar os dados de entrada de cada unidade. Seu uso ajuda a lidar com problemas no treinamento causados por erros de inicialização mas pode resultar em instabilidade no modelo. Para evitar tal problema seu uso não é aplicado na camada de saída do gerador nem na camada de entrada do discriminador.

Nas camadas de não linearidade do gerador, a função de ativação ReLU é utilizada com exceção da camada de saída, que utiliza a função Tanh. No discriminador, todas as camadas de não linearidade utilizam LeakyReLU.

Tal arquitetura demonstra performance competitiva com outros algoritmos de aprendizado não supervisionado.

3.2 Aplicação em estudos relacionados

Desenvolvido por Chintala et al. [10], um algoritmo [13] foi desenvolvido em Lua, utilizando a biblioteca Torch para implementação da DCGAN para geração de imagens originais.

Ao iniciar, alguns parâmetros iniciais são definidos:

```
opt = {
   dataset = 'folder',      -- indica a origem das imagens
   batchSize = 64,          -- indica o número de imagens em um batch
   loadSize = 96,
   fineSize = 64,
   nz = 100,                -- #  da dimensão para Z
   ngf = 64,                -- #  de filtros do gerados na primeira camada de convolução
   ndf = 64,                -- #  de filtros do discriminador na primeira camada de convolução
   nThreads = 4,            -- #  de threads a serem usada
   niter = 25,              -- #  de iterações para iniciar o aprendizado
   lr = 0.0002,             -- taxa de aprendizado inicial para adam
   beta1 = 0.5,             -- momentum de adam
   ntrain = math.huge,      -- #  de exemplos por epoch, math.huge utiliza todo dataset
   display = 1,             -- exibir imagens durante o treinamento. 0 = false
   display_id = 10,         -- id da janela de exibição
   gpu = 0,                 -- gpu = 0 utiliza apenas CPU.
   name = 'experiment1',
   noise = 'normal',        -- função utilizada para geração de z: uniforme / normal
}
```

A rede neural convolucional definida para o discriminador é composta por diversas camadas convolucionais, seguidas por camadas de não linearidade (utilizando a função de ativação LeakyReLU) e *batch normalization*. O discriminador recebe uma imagem de 64x64 e calcula a probabilidade da imagem ser real utilizando a função Sigmoide.

```
local netD = nn.Sequential()

-- input is (nc) x 64 x 64
netD:add(SpatialConvolution(nc, ndf, 4, 4, 2, 2, 1, 1))
netD:add(nn.LeakyReLU(0.2, true))
-- state size: (ndf) x 32 x 32
netD:add(SpatialConvolution(ndf, ndf * 2, 4, 4, 2, 2, 1, 1))
netD:add(SpatialBatchNormalization(ndf * 2)):add(nn.LeakyReLU(0.2, true))
-- state size: (ndf*2) x 16 x 16
netD:add(SpatialConvolution(ndf * 2, ndf * 4, 4, 4, 2, 2, 1, 1))
netD:add(SpatialBatchNormalization(ndf * 4)):add(nn.LeakyReLU(0.2, true))
-- state size: (ndf*4) x 8 x 8
netD:add(SpatialConvolution(ndf * 4, ndf * 8, 4, 4, 2, 2, 1, 1))
netD:add(SpatialBatchNormalization(ndf * 8)):add(nn.LeakyReLU(0.2, true))
-- state size: (ndf*8) x 4 x 4
netD:add(SpatialConvolution(ndf * 8, 1, 4, 4))
netD:add(nn.Sigmoid())
-- state size: 1 x 1 x 1
netD:add(nn.View(1):setNumInputDims(3))
-- state size: 1

netD:apply(weights_init)
```

Já o gerador é definido por uma rede neural convolucional que utiliza consecutivas camadas de convolução, camadas de não linearidade com função de ativação ReLU e *batch normalization* e uma camada e não linearidade com a função de ativação Tanh.

```
local netG = nn.Sequential()
-- input is Z, going into a convolution
netG:add(SpatialFullConvolution(nz, ngf * 8, 4, 4))
netG:add(SpatialBatchNormalization(ngf * 8)):add(nn.ReLU(true))
-- state size: (ngf*8) x 4 x 4
netG:add(SpatialFullConvolution(ngf * 8, ngf * 4, 4, 4, 2, 2, 1, 1))
netG:add(SpatialBatchNormalization(ngf * 4)):add(nn.ReLU(true))
-- state size: (ngf*4) x 8 x 8
netG:add(SpatialFullConvolution(ngf * 4, ngf * 2, 4, 4, 2, 2, 1, 1))
netG:add(SpatialBatchNormalization(ngf * 2)):add(nn.ReLU(true))
-- state size: (ngf*2) x 16 x 16
netG:add(SpatialFullConvolution(ngf * 2, ngf, 4, 4, 2, 2, 1, 1))
netG:add(SpatialBatchNormalization(ngf)):add(nn.ReLU(true))
-- state size: (ngf) x 32 x 32
netG:add(SpatialFullConvolution(ngf, nc, 4, 4, 2, 2, 1, 1))
netG:add(nn.Tanh())
-- state size: (nc) x 64 x 64

netG:apply(weights_init)
```

Pode-se observar que, assim como demonstrado em 3.1.3, a estrutura de ambos gerador e discriminador são similares, exceto pelo uso de diferentes funções de ativação na camada de não linearidade.

Além disso, o uso de um otimizador Adam (algoritmo para otimização de gradientes estocásticos) para acelerar o treinamento da rede pode ser observado a partir da definição dos parâmetros em 3.2.

Como demonstrado em 3.1.2, o gerador tende à minimizar V (3.2) enquanto o discriminador tende à maximizá-lo (3.1). Assim, a cada etapa do treinamento, ambas funções são calculadas para que possamos obter $\min_G \max_D V(D, G)$ (3.3).

Para realizar o cálculo da função de custo do gerador, temos:

```
local fDx = function(x)
   gradParametersD:zero()

   -- train with real
   data_tm:reset(); data_tm:resume()
   local real = data:getBatch()
   data_tm:stop()
   input:copy(real)
   label:fill(real_label)

   local output = netD:forward(input)
   local errD_real = criterion:forward(output, label)
   local df_do = criterion:backward(output, label)
   netD:backward(input, df_do)

   -- train with fake
   if opt.noise == 'uniform' then -- regenerate random noise
       noise:uniform(-1, 1)
   elseif opt.noise == 'normal' then
       noise:normal(0, 1)
   end
   local fake = netG:forward(noise)
   input:copy(fake)
   label:fill(fake_label)

   local output = netD:forward(input)
   local errD_fake = criterion:forward(output, label)
   local df_do = criterion:backward(output, label)
   netD:backward(input, df_do)

   errD = errD_real + errD_fake

   return errD, gradParametersD
end
```

Para o cálculo da função de custo do gerador, temos:

```
local fGx = function(x)
   gradParametersG:zero()
```

```
   label:fill(real_label)

   local output = netD.output -- netD:forward(input) was already executed in fDx, so save c
   errG = criterion:forward(output, label)
   local df_do = criterion:backward(output, label)
   local df_dg = netD:updateGradInput(input, df_do)

   netG:backward(noise, df_dg)
   return errG, gradParametersG
end
```

Assim, durante o treinamento, pode-se calcular 3.3:

```
-- (1) Update D network: maximize log(D(x)) + log(1 - D(G(z)))
optim.adam(fDx, parametersD, optimStateD)

-- (2) Update G network: maximize log(D(G(z)))
optim.adam(fGx, parametersG, optimStateG)
```

Para geração de imagens após o treinamento da rede, alguns parâmetros também são definidos:

```
opt = {
    batchSize = 32,         -- número de amostras a serem produzidas
    noisetype = 'normal',   -- tipo de distribuição do ruído: uniforme / normal.
    net = '',               -- caminho para a rede geradora
    imsize = 1,             -- tamanho da imagem gerada. 1 = 64px. 2 = 80px, 3 = 96px, ...
    noisemode = 'random',   -- tipo de ruído
    name = 'generation1',   -- nome do arquivo a ser salvo
    gpu = 1,                -- uso de GPU, onde 0 = CPU, 1 = GPU
    display = 1,            -- mostrar imagem: 0 = falso, 1 = verdadeiro
    nz = 100,
}
```

Assim, a partir da definição, o ruído z é gerado

```
noiseL = torch.FloatTensor(opt.nz):uniform(-1, 1)
noiseR = torch.FloatTensor(opt.nz):uniform(-1, 1)
if opt.noisemode == 'line' then
   -- do a linear interpolation in Z space between point A and point B
   -- each sample in the mini-batch is a point on the line
    line  = torch.linspace(0, 1, opt.batchSize)
    for i = 1, opt.batchSize do
        noise:select(1, i):copy(noiseL * line[i] + noiseR * (1 - line[i]))
    end
elseif opt.noisemode == 'linefull1d' then
   -- do a linear interpolation in Z space between point A and point B
   -- however, generate the samples convolutionally, so a giant image is produced
    assert(opt.batchSize == 1, 'for linefull1d mode, give batchSize(1) and imsize > 1')
```

```
    noise = noise:narrow(3, 1, 1):clone()
    line  = torch.linspace(0, 1, opt.imsize)
    for i = 1, opt.imsize do
        noise:narrow(4, i, 1):copy(noiseL * line[i] + noiseR * (1 - line[i]))
    end
elseif opt.noisemode == 'linefull' then
   -- just like linefull1d above, but try to do it in 2D
    assert(opt.batchSize == 1, 'for linefull mode, give batchSize(1) and imsize > 1')
    line  = torch.linspace(0, 1, opt.imsize)
    for i = 1, opt.imsize do
        noise:narrow(3, i, 1):narrow(4, i, 1):copy(noiseL * line[i] + noiseR * (1 - line[i]))
    end
end
```

e a imagem pode ser construída.

```
local images = net:forward(noise)
```

O código completo está disponível em [13].

4 Resultados e discussão

Em todos os treinamentos, foi utilizado um computador MacBook Pro Mid 2015, com processador Intel Core i7 e 16 GB de RAM e sistema operacional MacOS High Sierra 10.13.6. Como entrada de todos os treinamentos, as imagens utilizadas possuem tamanho de 64 por 64 pixels - estas ajustadas por tratamento do próprio algoritmo proposto por Chintala et. al por limitações da estrutura da rede. [13]

Em todos os experimentos foram utilizadas 64 camadas de convolução tanto no gerador quanto no discriminador. Outros parâmetros, como o tamanho do *batch*, *momentum* e taxa de aprendizado foram mantidos os valores propostos em 3.2.

4.1 Experimento alfa (A1)

Com intenção de validar a implementação escolhida, imagens das obras de *Jê Américo* em pequena escala foram utilizadas no primeiro experimento a fim de avaliar se o uso da rede proposta adequava-se à necessidade do projeto, além de permitir a iteração mais ágil com o artista acerca dos resultados obtidos.

Figura 8 – “calçada: vermelho” de *Jê Américo*

4.1.1 Tratamento das obras

Inicialmente, cerca de 70 obras digitalizadas foram disponibilizadas pelo artista em diversos tamanhos e resoluções, entre elas "calçada: vermelho"(imagem 8).

Dado a restrição da quantidade de obras disponíveis, tornou-se necessário aumentar o número de imagens disponíveis para o treinamento do algoritmo. Entre as opções avaliadas estava a repetição das imagens disponíveis para uso ou fragmentá-las em pequenas partes.

Como a estrutura das redes neurais convolucionais extrai um filtro de características de cada imagem e depois os combina para a geração, criou-se a hipótese de que, ao fragmentar as obras, seria possível isolar os elementos usados pelo artista em partes menores para que a rede fosse capaz aprendê-los e combiná-los em uma composição maior. Além disso, também era necessário padronizar o tamanho das imagens para o treinamento.

Apesar de o algoritmo utilizar imagens de 64x64 pixels, os fragmentos foram gerados com tamanho de 256x256 pixels para preservar maior quantidade de características a cada fragmento. Dessa forma, obteve-se aproximadamente 18 mil imagens a serem utilizadas posteriormente na execução do algoritmo.

4.1.2 Resultados obtidos

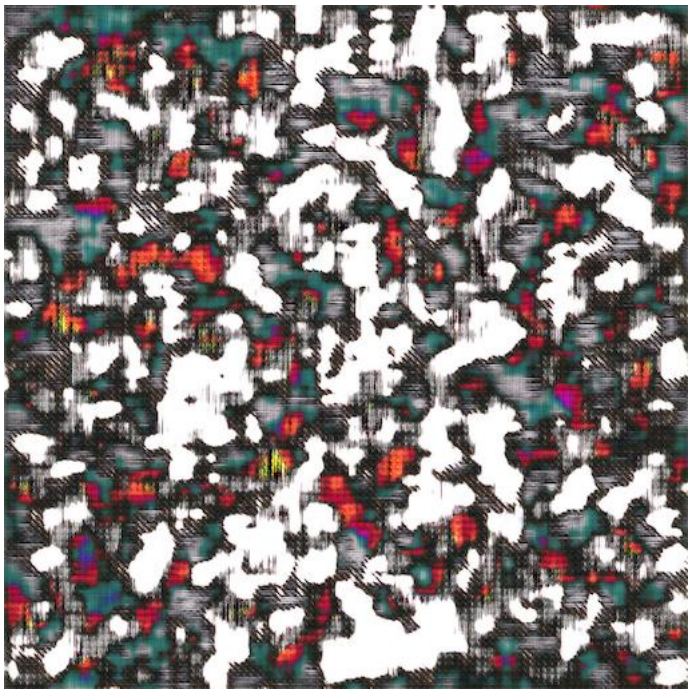


Figura 9 – Resultado obtido após treinamento alfa

A presença de diversas "manchas brancas", como pode-se observar em um dos resultados apresentados (imagem 9), característica não presente nas obras de Américo,

foi um dos principais problemas encontrados nas imagens geradas por este primeiro experimento.

Uma das hipóteses para a causa desse efeito é a falta de tratamento das bordas das imagens apresentadas, como pode-se observar na figura 8 e, posteriormente, na figura **??**.

Apesar de apresentar esboços de linhas, uma das características mais marcantes das produções do autor, os elementos das imagens geradas ainda se distanciam bastante de outros nuances das obras elaboradas por Jê.

Com a existência de outra possibilidade de tratamento das obras (conforme apresentado em 4.1.1) e a percepção de que alguns elementos da produção artística de Jê era apresentado no resultado obtido, decidiu-se continuar o uso do algoritmo proposto.

4.2 Primeiro experimento (E1)

Com melhor compreensão do funcionamento algoritmo e sua iteração com as criações de Jê, um experimento mais elaborado foi realizado. Para isso, novas obras em alta qualidade foram disponibilizadas pelo autor, totalizando 123 imagens.

4.2.1 Tratamento das obras

Na tentativa de evitar os problemas apresentados anteriormente em 4.1 neste experimento, o tratamento incluiu a retirada das bordas de todas as imagens.

Com objetivo de manter mais características das obras em cada fragmento, as ilustrações foram divididas em partes maiores em relação ao experimento inicial, com 1024 pixels por 1024 pixels. Para torná-las menores e facilitar o carregamento delas no algoritmo, os fragmentos foram posteriormente redimensionadas ao tamanho de 256 pixels por 256 pixels, gerando 3682 imagens.

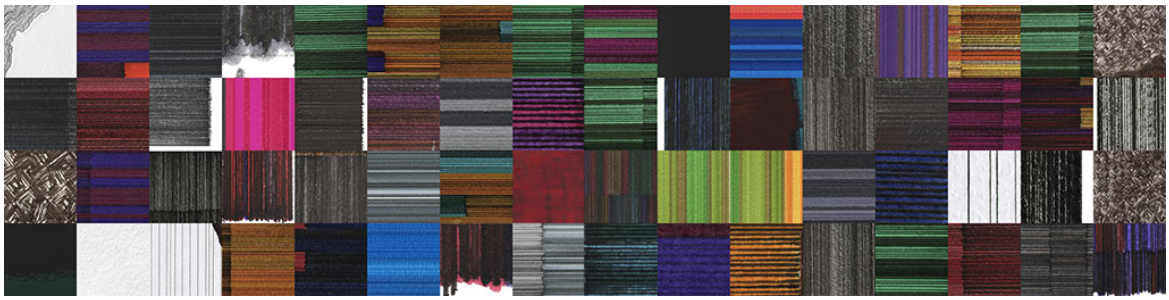


Figura 10 – Composição de fragmentos das obras de *Jê Américo* utilizados no primeiro experimento

4.2.2 Realização do experimento

Após o tratamento das obras, o número de amostras geradas para o treinamento ainda era baixo. Assim, foi necessário que a amostra fosse replicada quatro vezes para obtenção de um resultado satisfatório, resultando em 14728 imagens utilizadas.

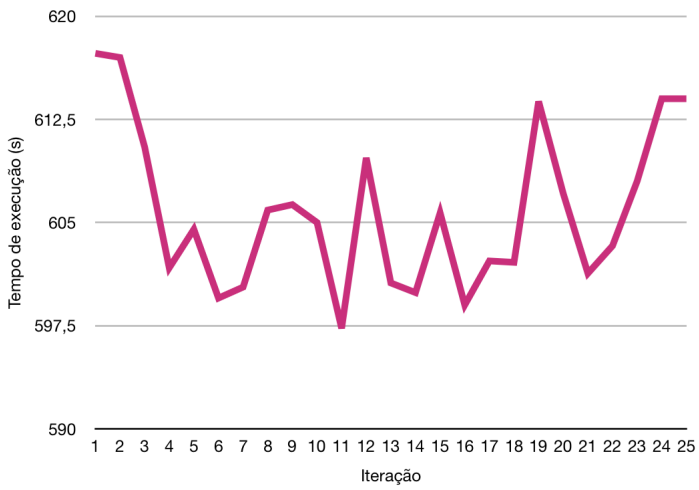


Figura 11 – Tempo de execução do primeiro treinamento

Durante o treinamento da rede, cada iteração levou em média 605,9s, levando um tempo total de execução de aproximadamente 4 horas e 15 minutos, como é possível observar na figura 11.

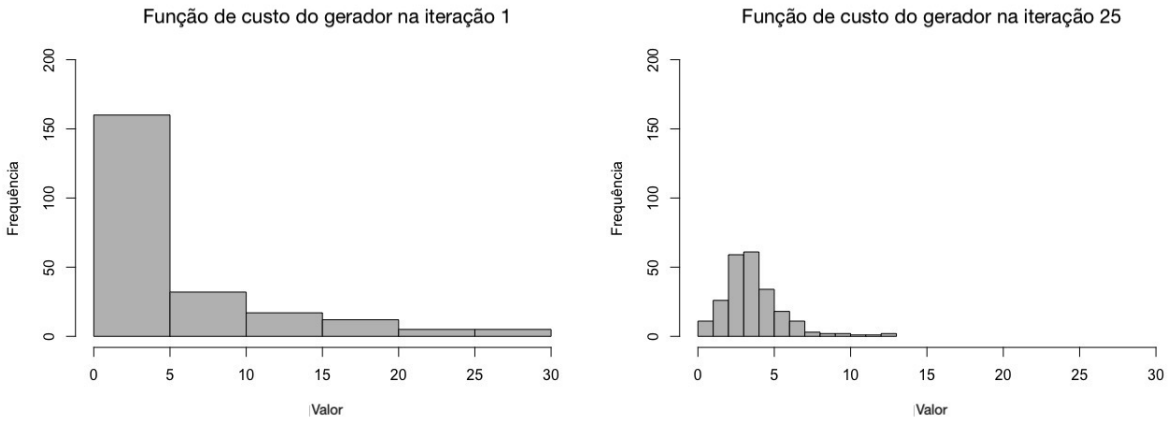


Figura 12 – Histograma do resultado da função de custo do gerador na primeira e última iteração do treinamento durante o experimento 1

Todos os gráficos tiveram seus eixos padronizados ao maior valor atingido no treinamento para facilitar a comparação entre eles.

Como pode-se observar na figura 12, a função de custo do gerador, cujo objetivo é minimizar a função apresentada em 3.2, teve seus valores reduzidos entre a primeira e

última iteração do treinamento. Além disso, é possível observar que os valores oscilam entre as iterações, dado que a rede tende a realizar a operação de $\min_G \max_D V(D, G)$ apresentada em 3.3 durante as iterações.

Os histogramas detalhados por iteração do gerador do experimento E1 podem ser encontrados em A.2.1.

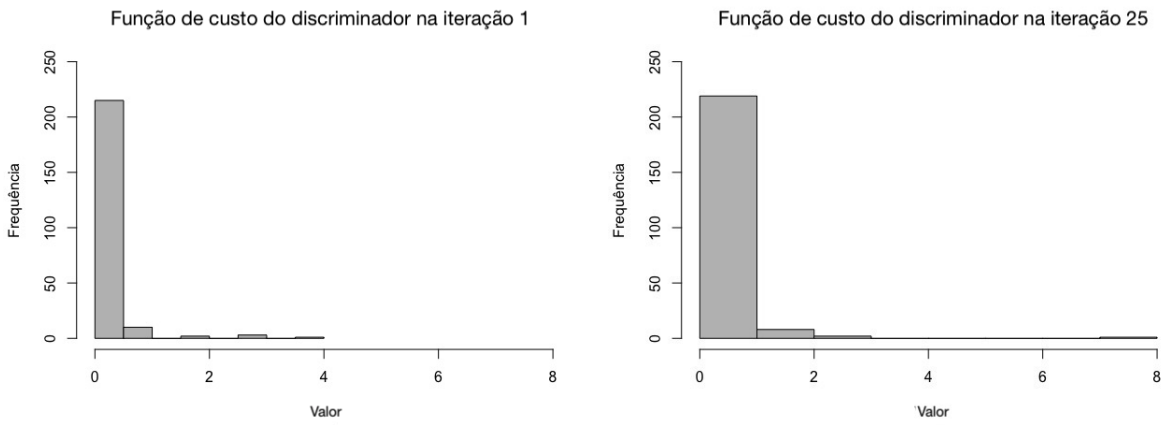


Figura 13 – Histograma do resultado da função de custo do discriminador na primeira e última iteração do treinamento durante o experimento 1

Já valor da função de custo do discriminador, cujo objetivo é maximizar o valor da função apresentada em 3.1,teve seus valores aumentados entre a primeira e última iteração do treinamento. Assim como a função de custo do gerador, também é possível observar que os valores oscilam entre as iterações.

Os histogramas detalhados por iteração do discriminador do experimento E1 podem ser encontrados em A.2.2.

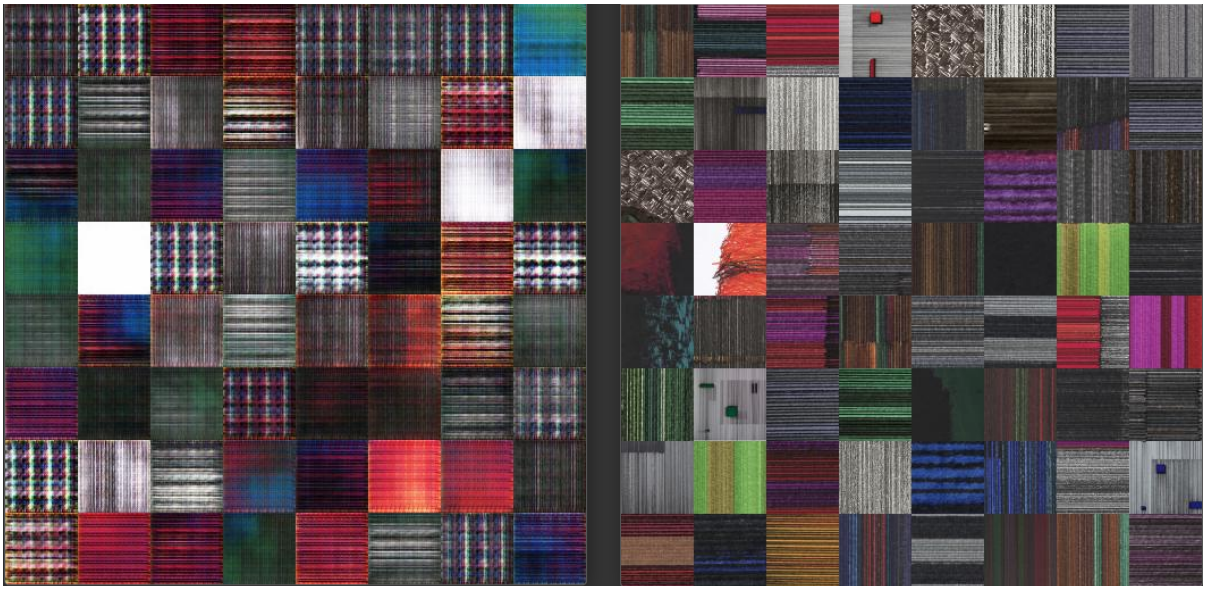


Figura 14 – Treinamento da rede no primeiro experimento após 25 iterações: imagens à esquerda são produzidas pelo gerador e à direita são imagens de entrada

4.2.3 Resultados obtidos

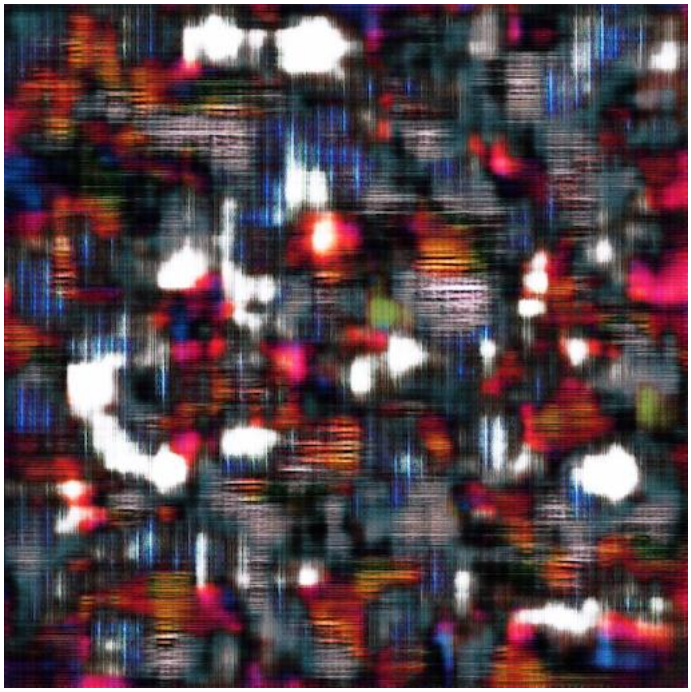


Figura 15 – Resultado obtido após primeiro treinamento

Assim como no experimento A1, também é possível observar na Figura 15 a presença de algumas "manchas brancas"em menor escala nas imagens geradas por este experimento.

Como a hipótese anterior havia sido eliminada no tratamento das obras, a possível causa das "manchas"é a presença de obras da série de mapas, como a imagem 1 - que

possui vasta área branca, esta não compondo parte da área de interesse da obra para o treinamento.

Além disso, assim como no experimento anterior, ainda não é possível observar outros elementos da obra do artista além das linhas por ter-se utilizado, novamente, fragmentos das imagens e não sua representação completa.

Mais resultados podem ser encontrados em A.2.3.

4.3 Segundo experimento (E2)

4.3.1 Tratamento das obras

A fim de gerar imagens com maior verossimilhança às obras de Jê Américo, neste experimento optou-se por não fragmentar as obras e utilizá-las como originais.

Para isso, todas as imagens disponibilizadas pelo autor foram redimensionadas para o tamanho de 256 pixels de largura.

Além disso, na tentativa de eliminar as "manchas brancas"observadas no resultado anterior (figura 15), as quatro obras da série de mapas (da qual a figura 1 faz parte) foram excluídas do *input* deste treinamento, assim gerando um total de 119 obras disponíveis para o treinamento.

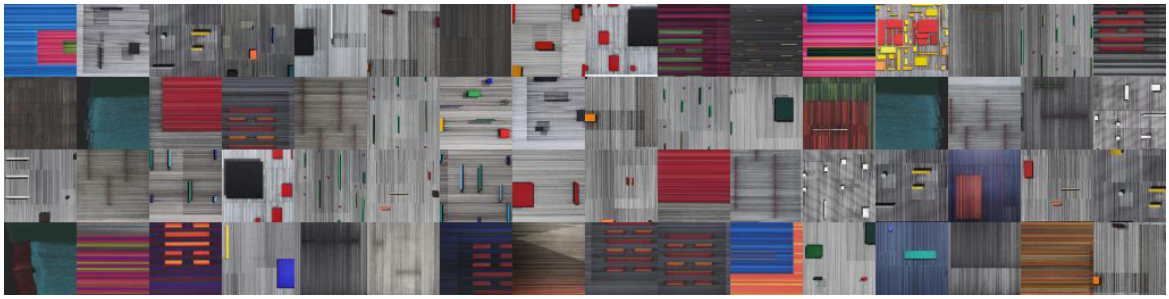


Figura 16 – Obras de *Jê Américo* utilizados no segundo experimento

4.3.2 Realização do experimento

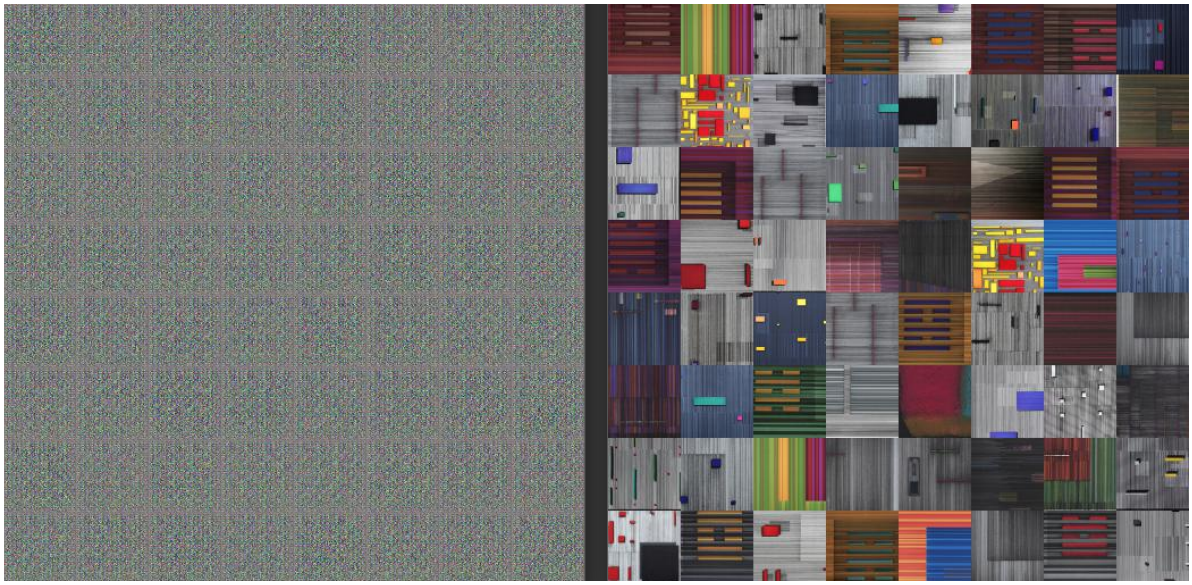


Figura 17 – Treinamento da rede no segundo experimento após 1 iteração: imagens à esquerda são produzidas pelo gerador e à direita são imagens de entrada

Tal qual o experimento E1, após o tratamento das obras, o número de amostras disponíveis para a execução do treinamento ainda era bastante baixo. Da forma, foi necessário replicar as imagens 128 vezes para que o número de amostras se aproximasse do utilizado anteriormente, resultando em um total de 15232 imagens.

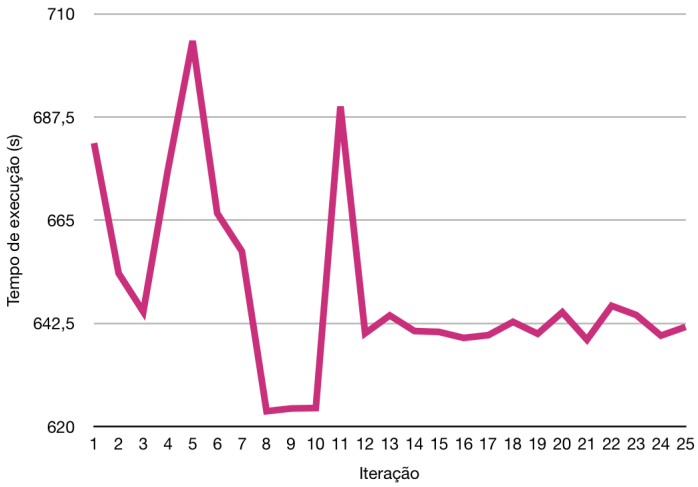


Figura 18 – Tempo de execução do primeiro treinamento

Neste treinamento, cada iteração do treinamento da rede levou em média 649,2 segundos, totalizando um tempo de execução de aproximadamente 4 horas e 30 minutos, como é possível observar na figura 18.

O tempo total é bastante semelhante ao resultado do experimento anterior mas a média elevou-se um pouco. Uma das hipóteses desse aumento foi o uso de imagens padronizadas apenas pela largura, já que a rede é mais adequada para o uso de imagens quadradas, requerendo, assim, maior processamento.

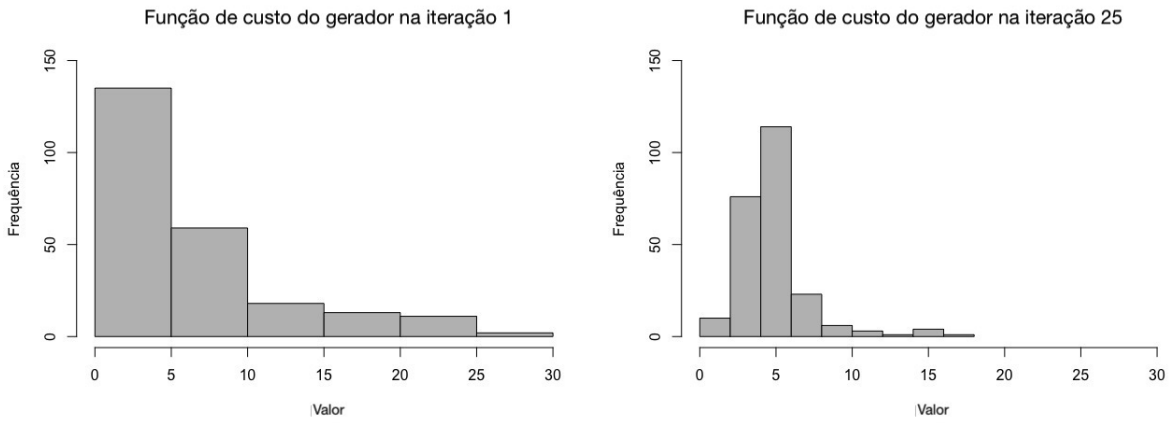


Figura 19 – Histograma de erro do gerador na primeira e última iteração do treinamento durante o experimento 3

Tal qual o primeiro experimento, pode-se observar que o valor da função de custo do gerador tende minimizar-se conforme as iterações do treinamento ocorrem.

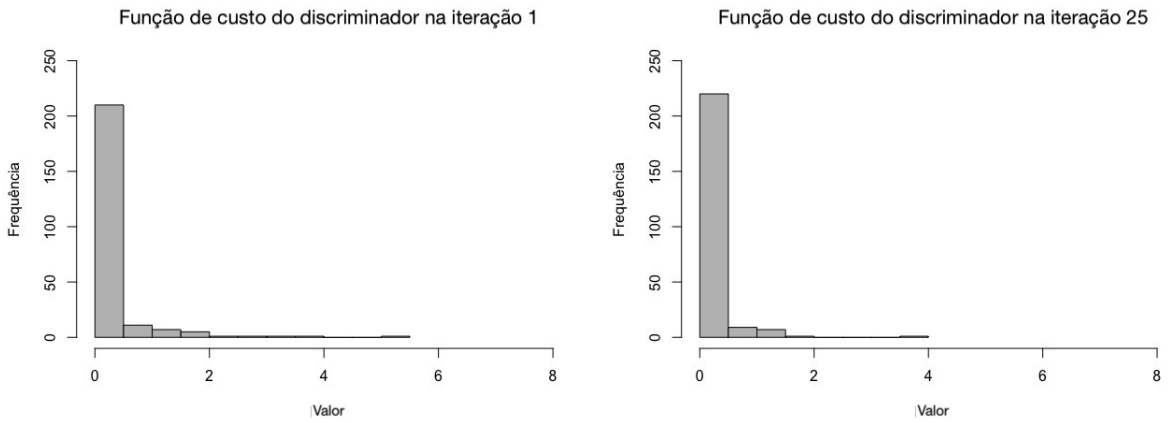


Figura 20 – Histograma de erro do discriminador na primeira e última iteração do treinamento durante o experimento 3

Já o valor da função de custo do discriminador tende maximizar-se após as iterações do treinamento, como demonstra-se em 20.

Os histogramas detalhados por iteração do gerador e do discriminador podem ser encontrados em A.3.2.

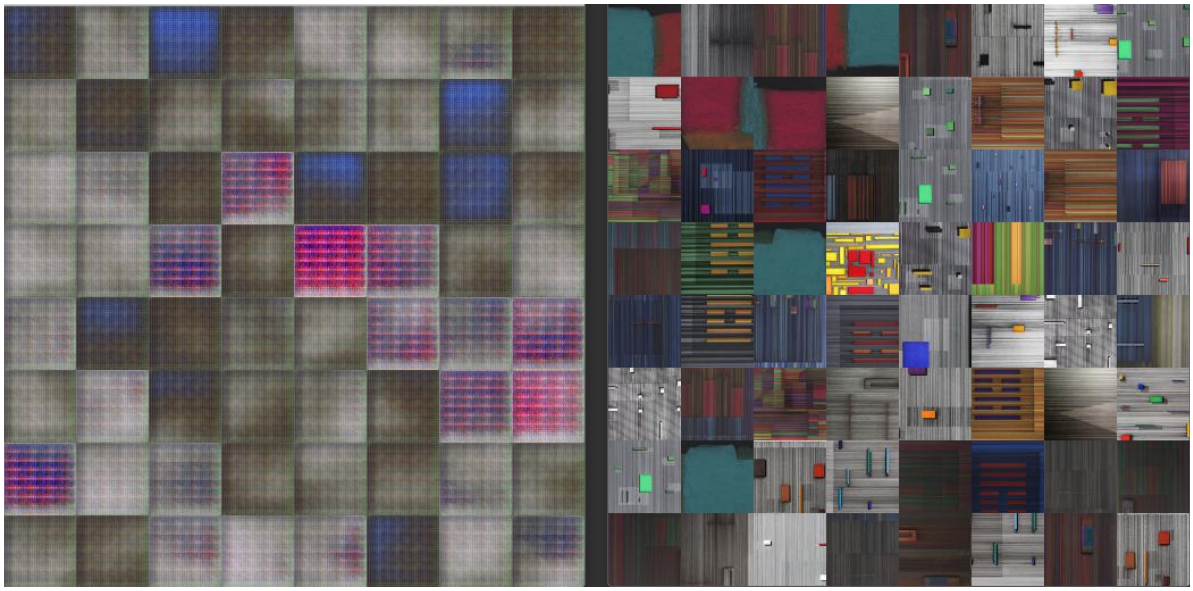


Figura 21 – Treinamento da rede no segundo experimento após 7 iterações: imagens à esquerda são produzidas pelo gerador e à direita são imagens de entrada

4.3.3 Resultados obtidos

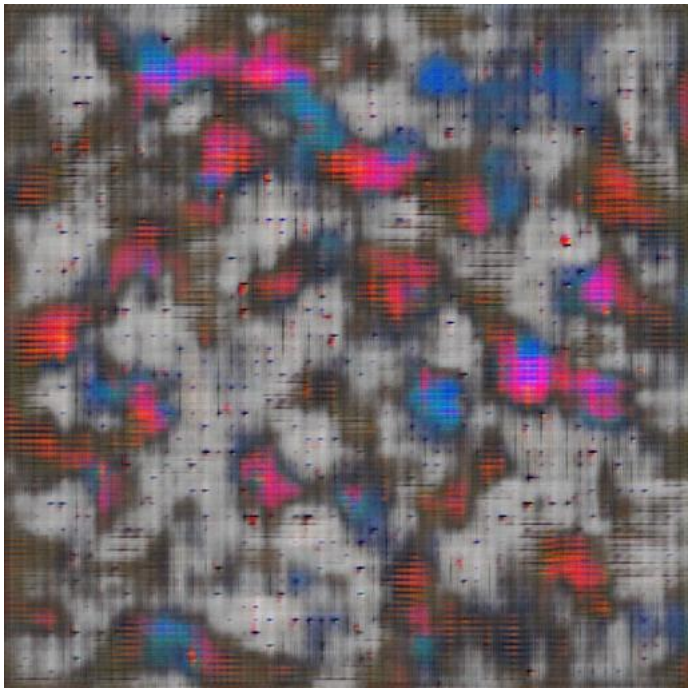


Figura 22 – Resultado obtido após segundo treinamento

Como pode-se perceber na figura 22, os resultado obtido após o segundo treinamento não apresentam as "manchas brancas"presente nos resultados dos experimentos anteriores, apesar de apresentar "manchas"de colorações variadas, dada a variação de cores usadas nas diversas obras do artista.

Por ter-se utilizado as obras completas do autor, as imagens geradas apresentam mais elementos característicos de sua produção, como, por exemplo, a presença de formas geométricas e sombras.

Mais resultados deste experimento podem ser encontrados em A.3.3.

5 Conclusão

O estudo teórico de redes neurais convolucionais e sua aplicação em diversas arquiteturas permitiu compreender seus possíveis usos, capacidades e limitações.

Em relação ao uso de redes adversárias generativas convolucionais profundas e do algoritmo proposto por Chintala et al. [10], foi possível entender melhor a aplicação das redes neurais convolucionais para geração de imagens originais.

O algoritmo estudado e utilizado apresentou, com ressalvas, grande potencial para obtenção de futuras obras com maior semelhança às obras originais de Jê Américo.

Os resultados obtidos no experimento alfa estavam completamente fora dos padrões que o artista desenvolveu, o que, para ele, "resultou em manchas coloridas, tramas pretas e manchas brancas"onde foi possível reconhecer interessantes novos padrões nas "manchas coloridas".

Em relação aos resultados obtidos no primeiro experimento, Jê comenta que o resultado estético soou mais interessante:

> *"Gostei das manchas e das tramas, porém a composição geral não se assemelha as composições que faço, onde há o predomínio de formas geométricas, e no caso, as imagens obtidas apresentam outros princípios de composição."*

Sobre os resultados obtidos no segundo experimento, Jê comenta que, apesar de serem utilizadas obras completas no lugar de fragmentos, para ele "o padrão compositivo não se alterou significativamente, mas sim o padrão de cores".

Uma possibilidade não explorada no presente trabalho por limitações de *hardware* foi o aumento das dimensões das imagens de entrada, que poderia atingir até 128 pixels por 128 pixels, podendo fornecer resultados melhores ao treinamento da rede. Além disso, o pequeno número de imagens originais restringiu a variedade de resultados obtidos nos experimentos apresentados, também se mostrando como uma possibilidade a ser explorada em trabalhos futuros.

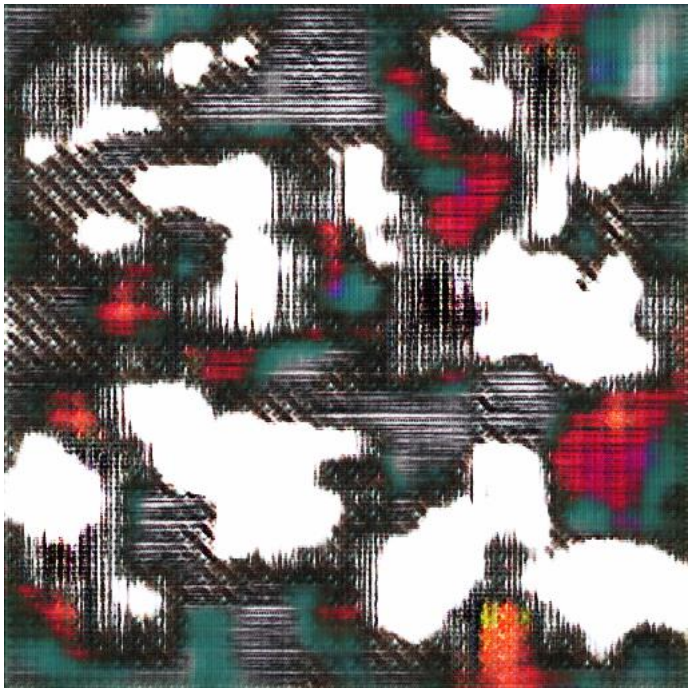
Apesar de não se aproximarem completamente às obras do artista, o resultado obtido foi bastante satisfatório por conseguir trazer importantes características da produção de Jê Américo.

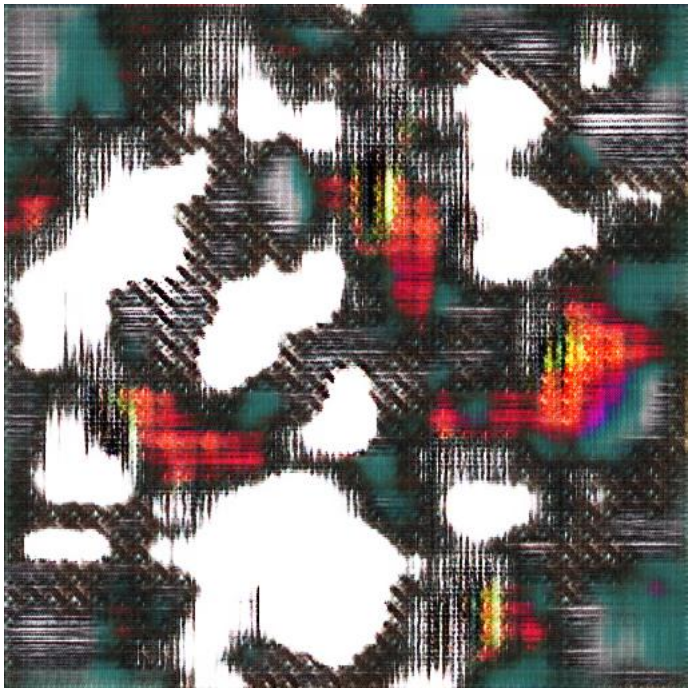
Além disso, observar como o tratamento das obras do artista resultava em resultados diversos foi bastante interessante e, para Américo, o fez pensar sobre como se dá seu próprio processo criativo.

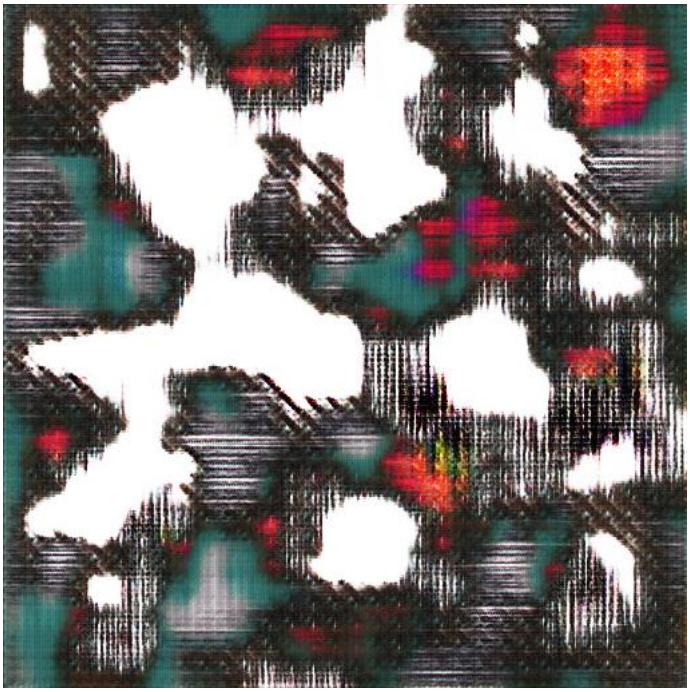
A Informações complementares

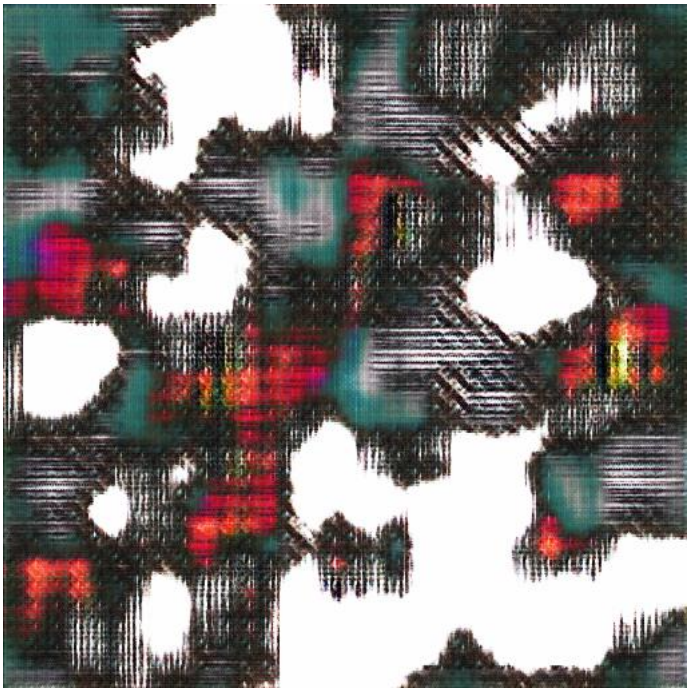
A.1 Experimento alfa (A1)

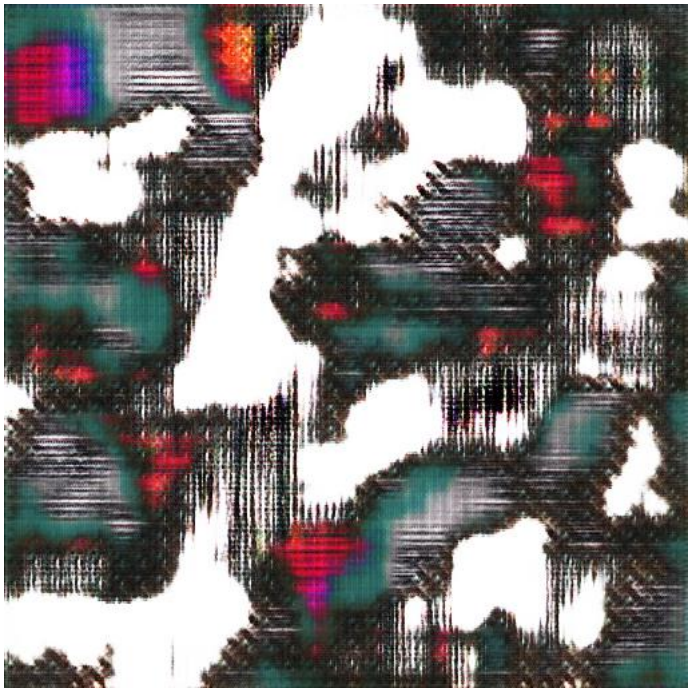
A.1.1 Resultados obtidos

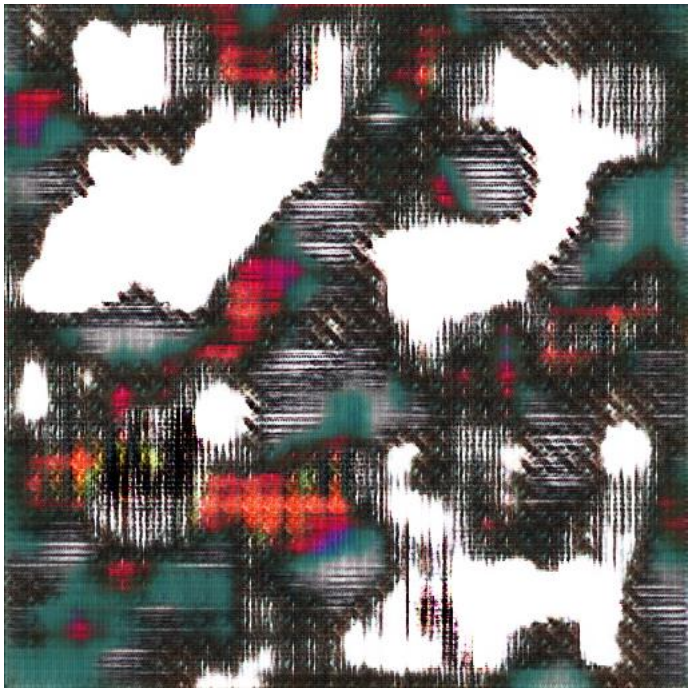


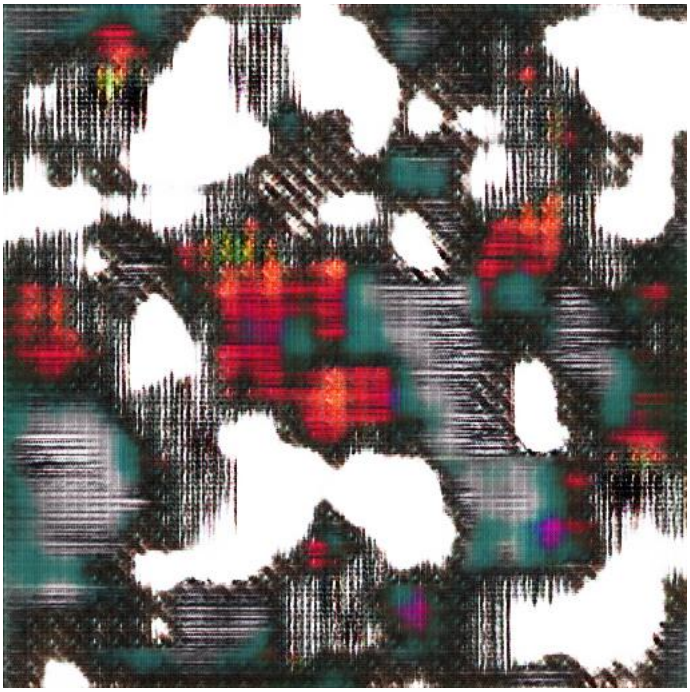


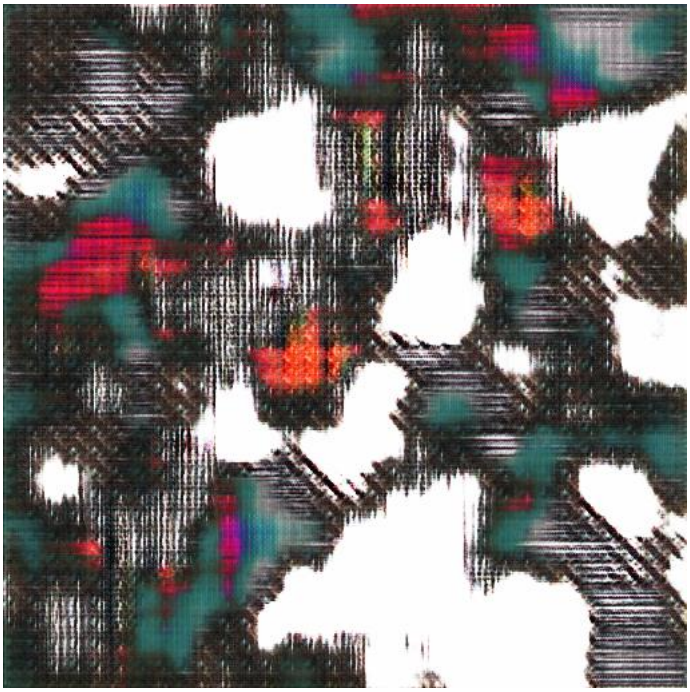












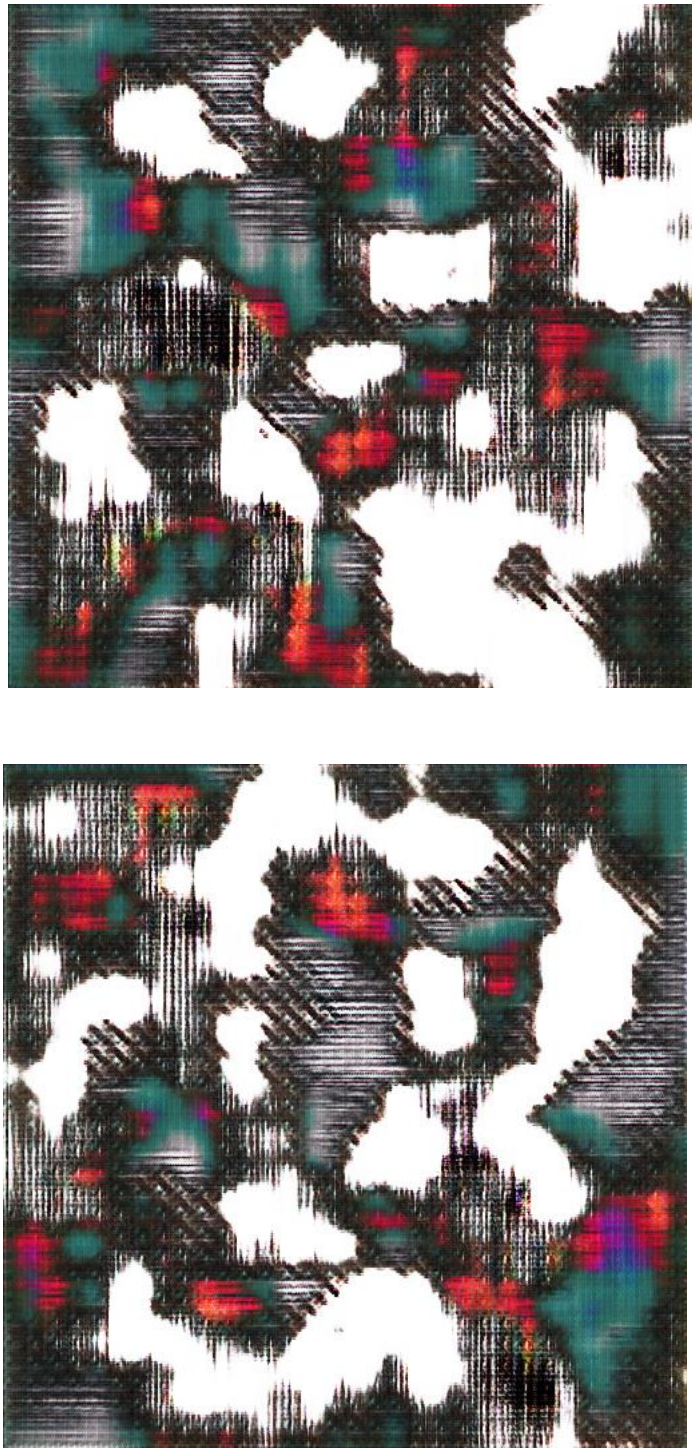
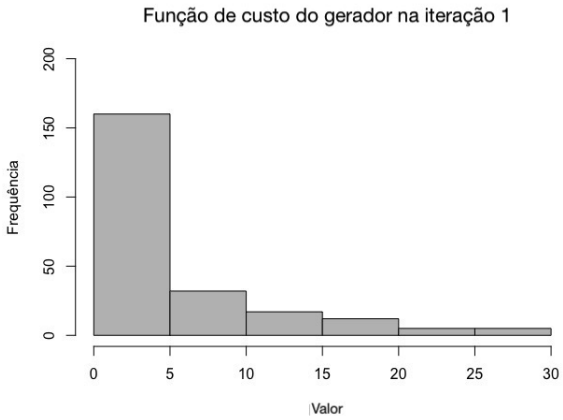
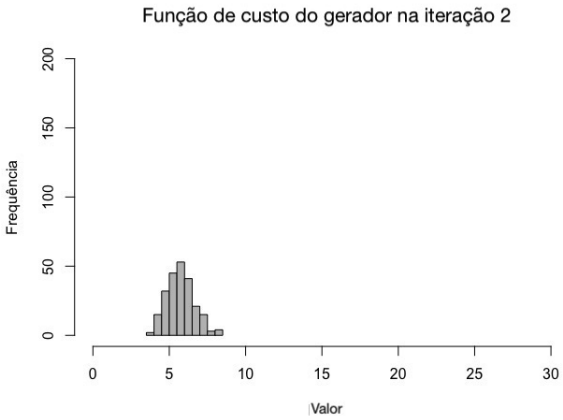


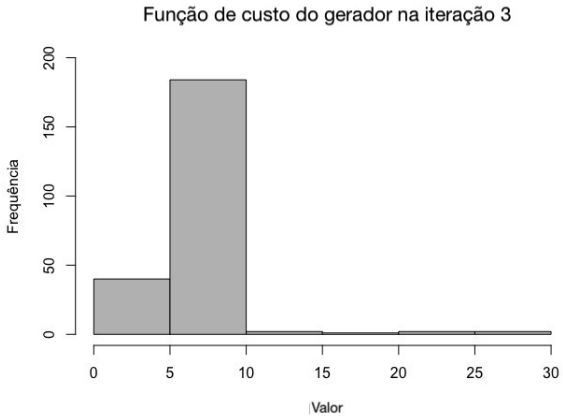
Figura A.1 – Resultados obtidos após treinamento do experimento alfa

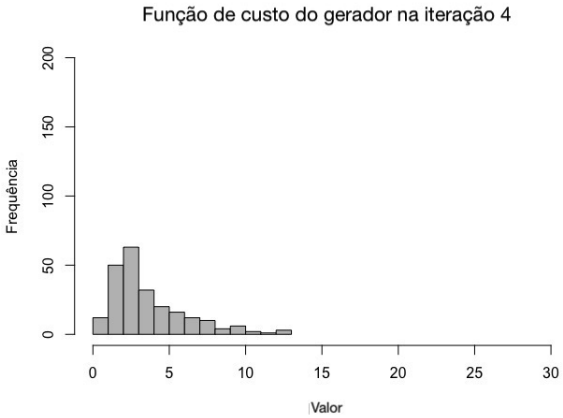
A.2 Primeiro experimento (E1)

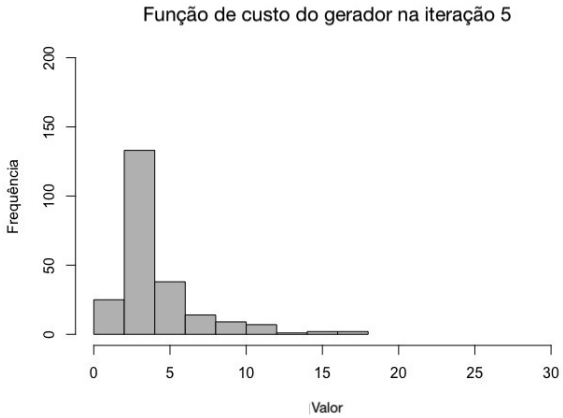
A.2.1 Histogramas da função de custo do gerador

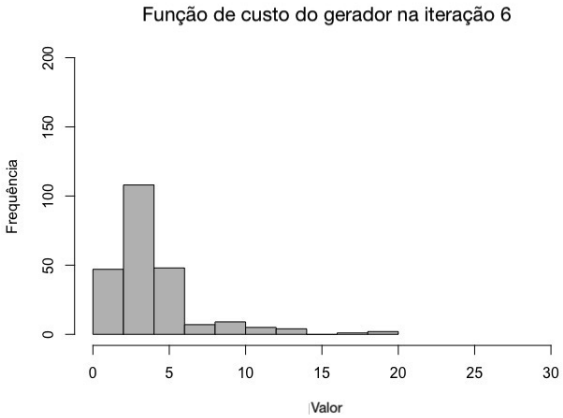

Função de custo do gerador na iteração 1
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor

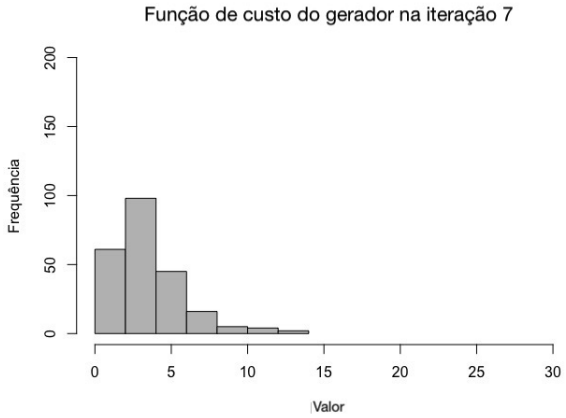

Função de custo do gerador na iteração 2
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor

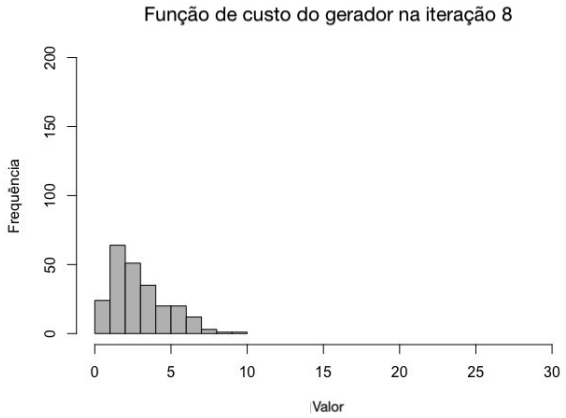

Função de custo do gerador na iteração 3
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor

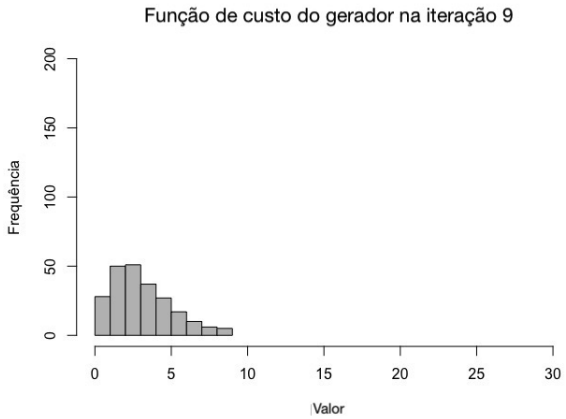

Função de custo do gerador na iteração 4
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor

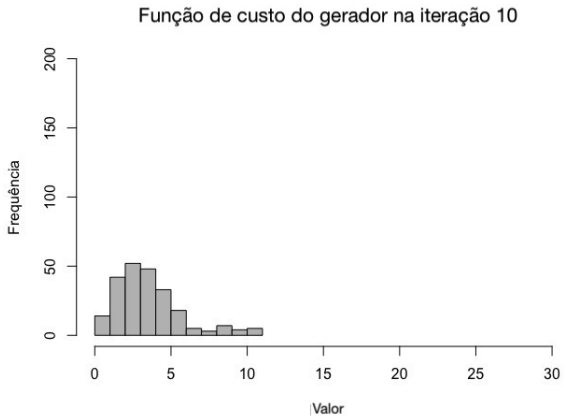

Função de custo do gerador na iteração 5
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor

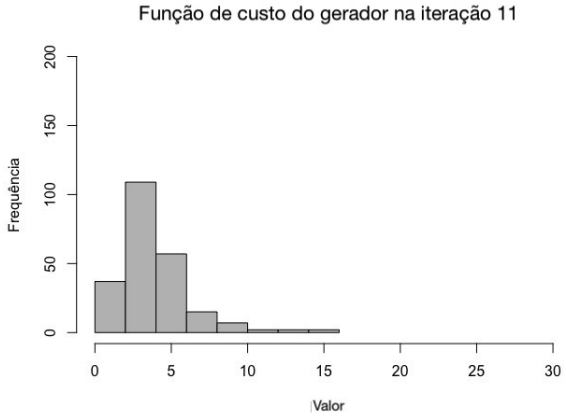

Função de custo do gerador na iteração 6
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor

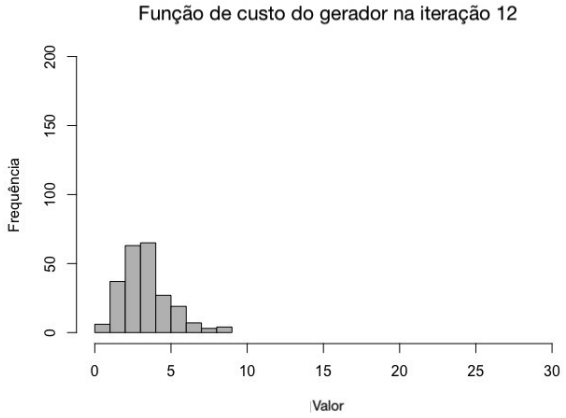

Função de custo do gerador na iteração 7
Frequência
Valor

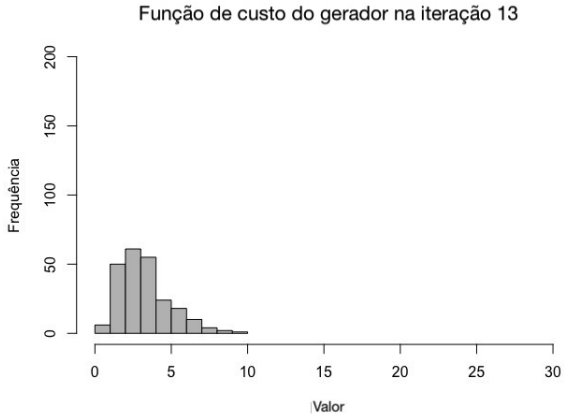

Função de custo do gerador na iteração 8
Frequência
Valor

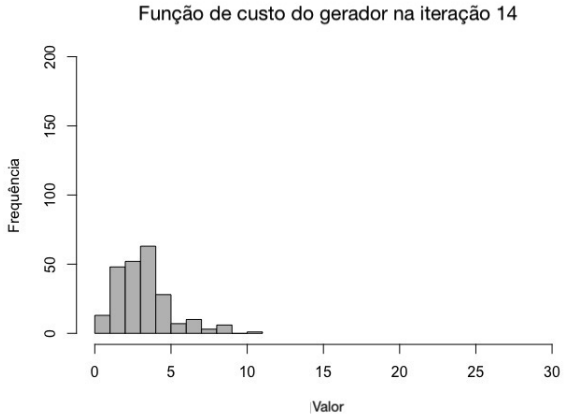

Função de custo do gerador na iteração 9
Frequência
Valor

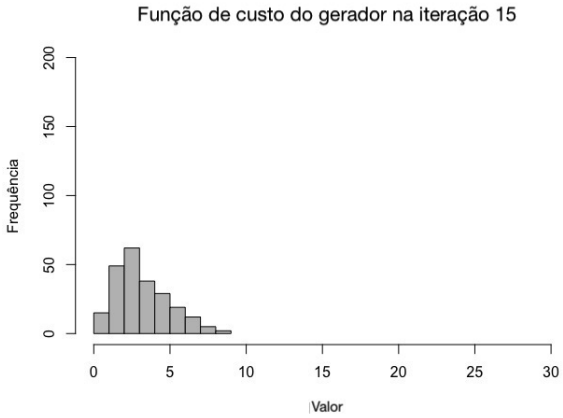

Função de custo do gerador na iteração 10
Frequência
Valor

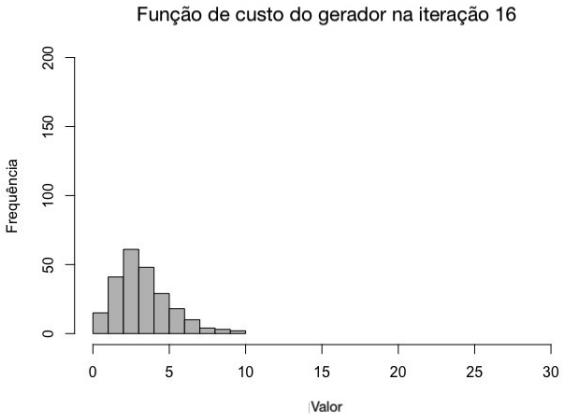

Função de custo do gerador na iteração 11
Frequência
Valor

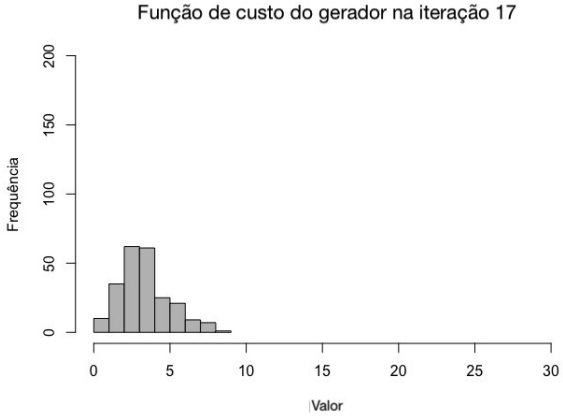

Função de custo do gerador na iteração 12
Frequência
Valor

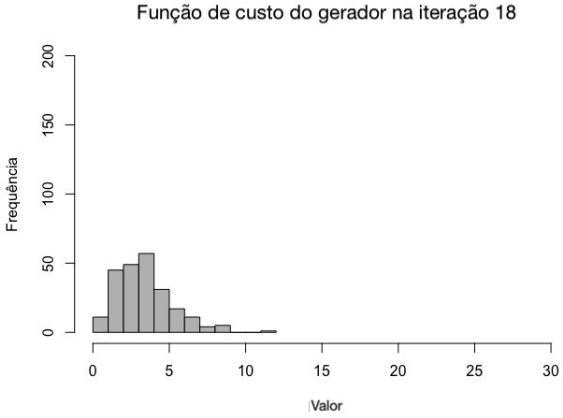

Função de custo do gerador na iteração 13
Frequência
Valor

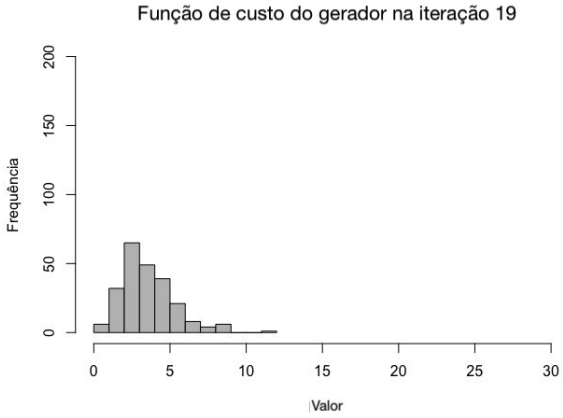

Função de custo do gerador na iteração 14
Frequência
Valor

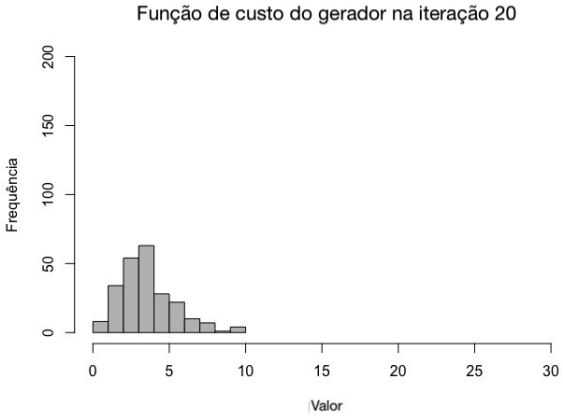

Função de custo do gerador na iteração 15
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor

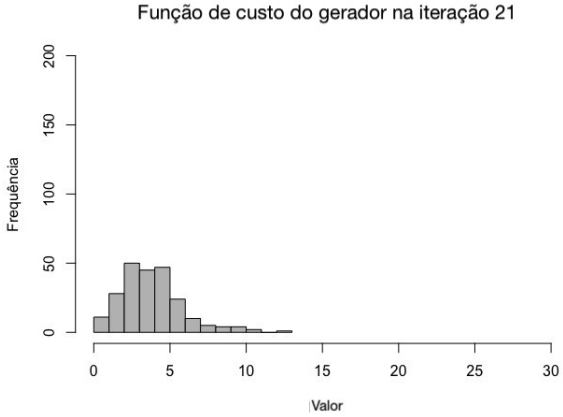

Função de custo do gerador na iteração 16
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor

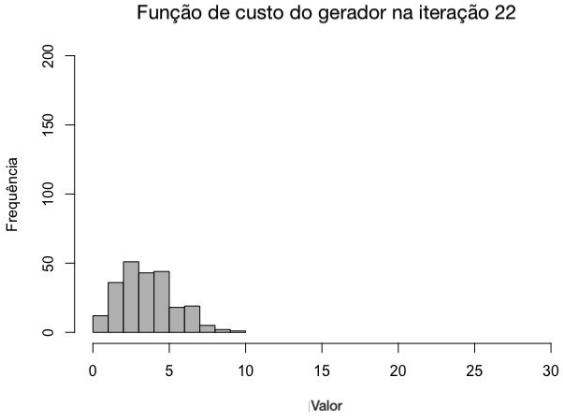

Função de custo do gerador na iteração 17
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor


Função de custo do gerador na iteração 18
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor


Função de custo do gerador na iteração 19
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor


Função de custo do gerador na iteração 20
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor


Função de custo do gerador na iteração 21
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor


Função de custo do gerador na iteração 22
Frequência
0
50
100
150
200
0
5
10
15
20
25
30
Valor

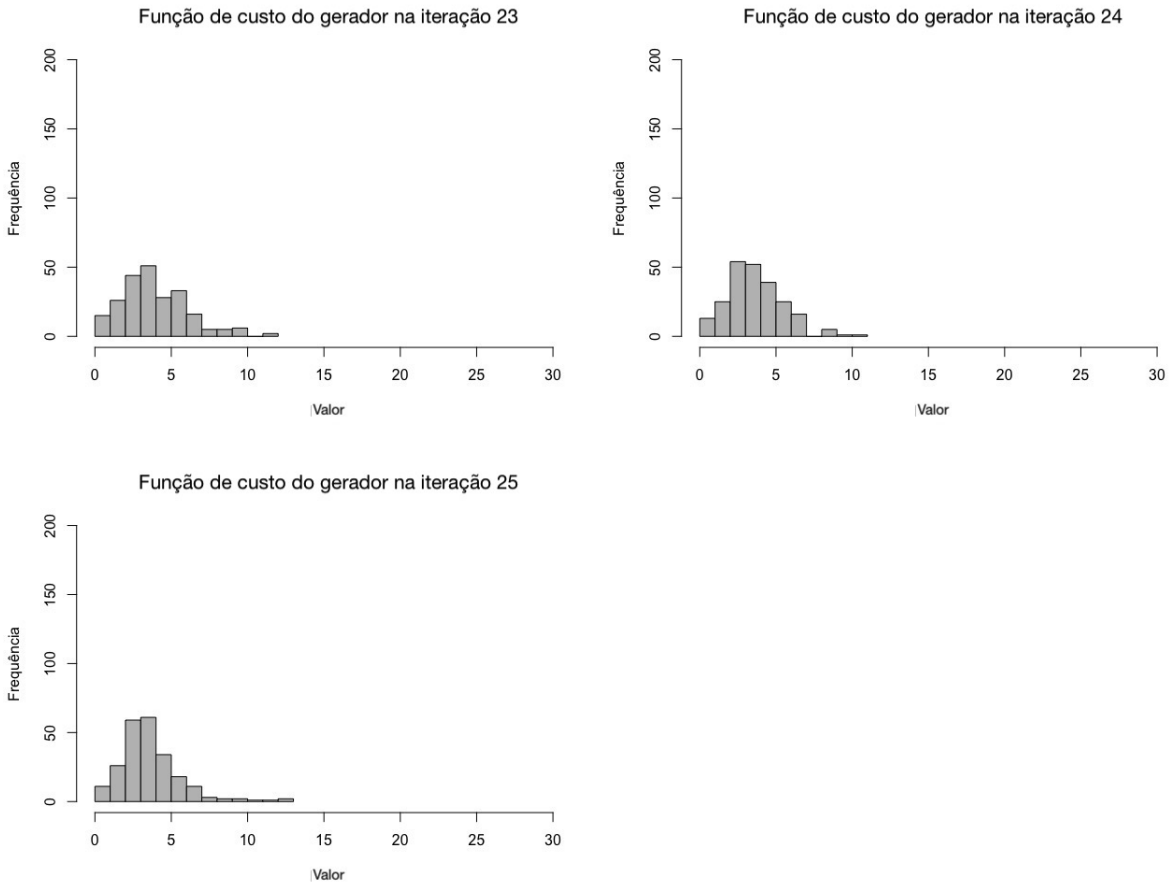
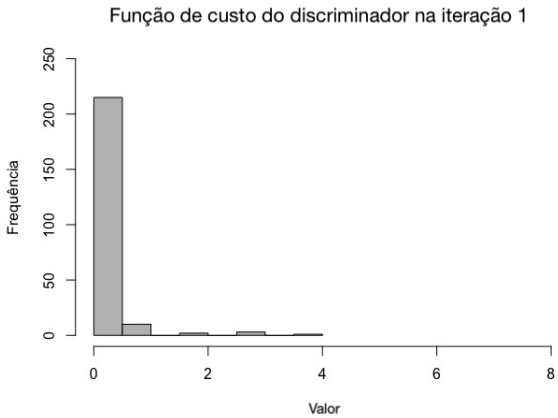
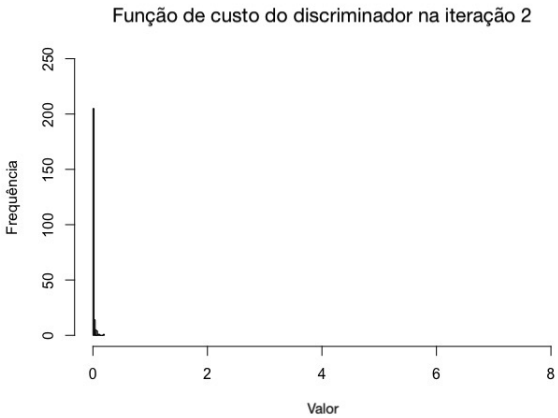
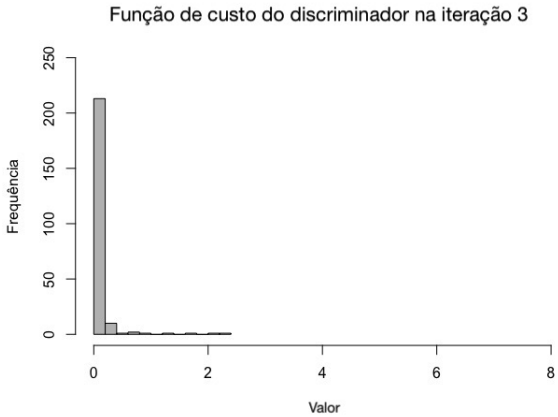


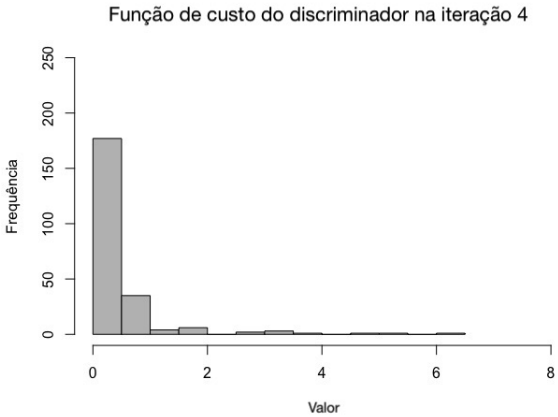
Figura A.2 – Histogramas da função de custo do gerador a cada etapa de treinamento do primeiro experimento

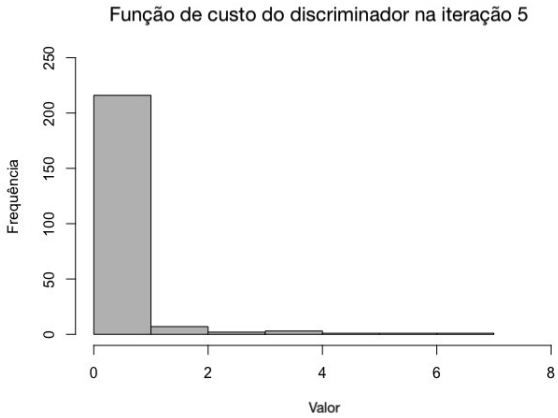
A.2.2 Histogramas da função de custo do discriminador

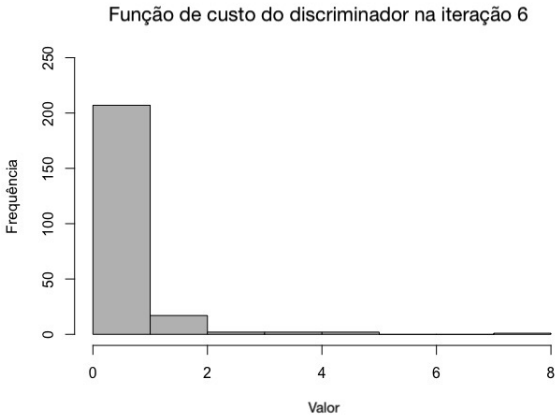


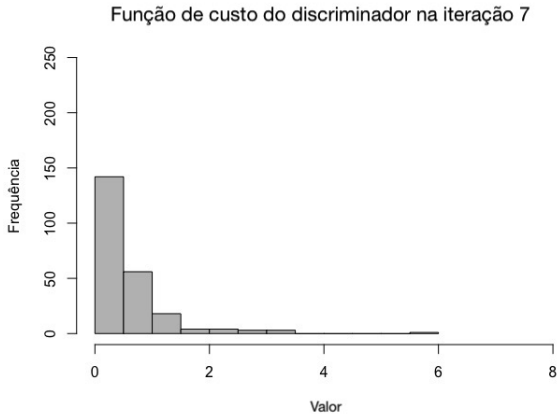


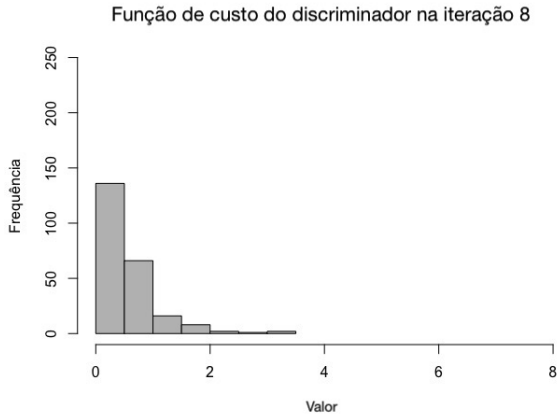


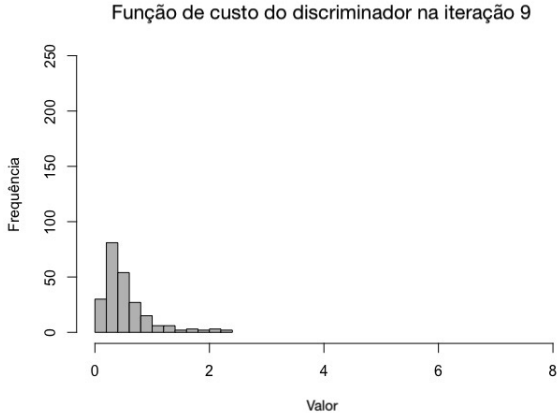


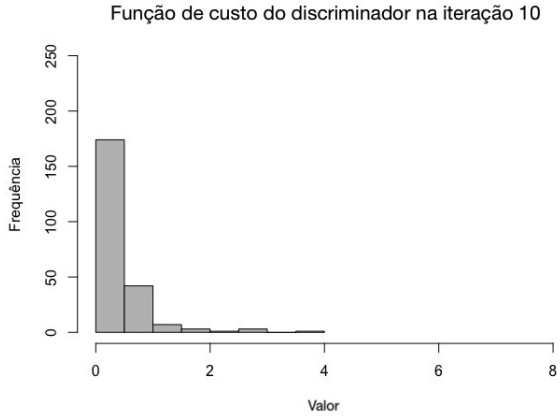


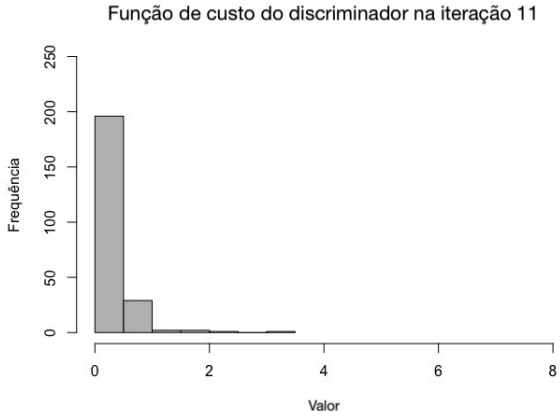


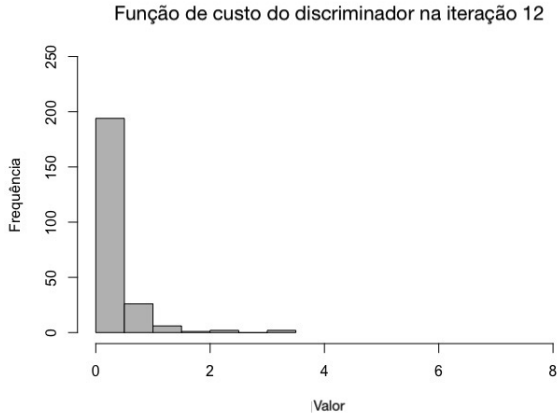

Função de custo do discriminador na iteração 7
Frequência
Valor

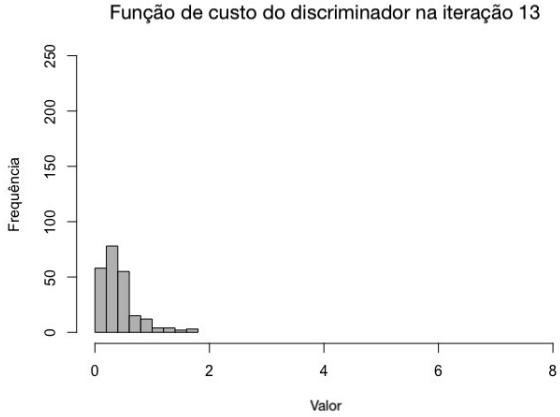

Função de custo do discriminador na iteração 8
Frequência
Valor

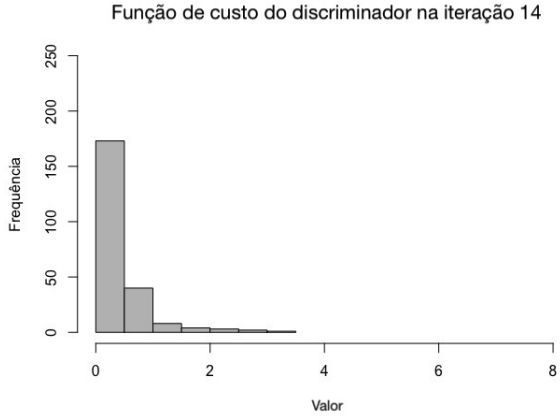

Função de custo do discriminador na iteração 9
Frequência
Valor

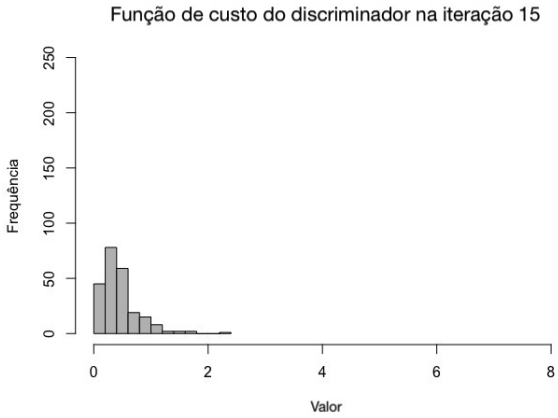

Função de custo do discriminador na iteração 10
Frequência
Valor

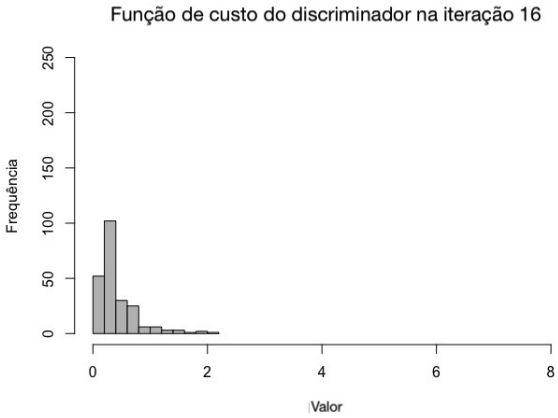

Função de custo do discriminador na iteração 11
Frequência
Valor

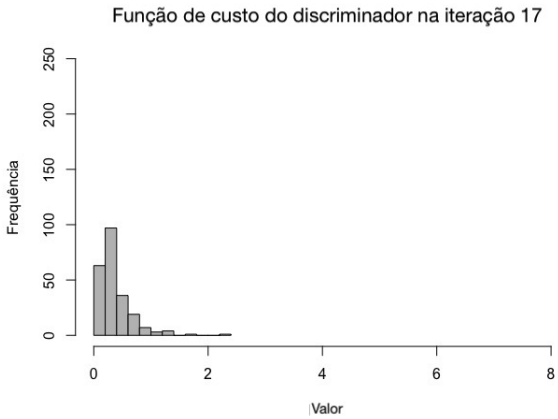

Função de custo do discriminador na iteração 12
Frequência
Valor

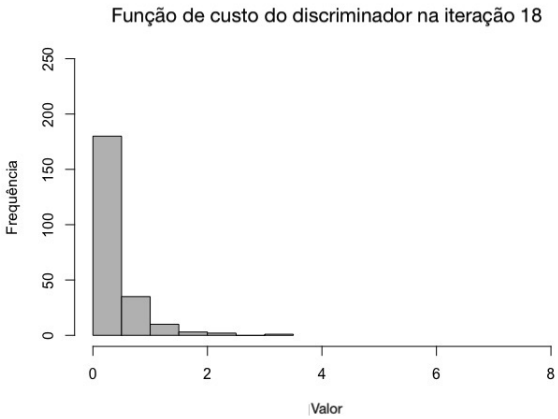

Função de custo do discriminador na iteração 13
Frequência
Valor

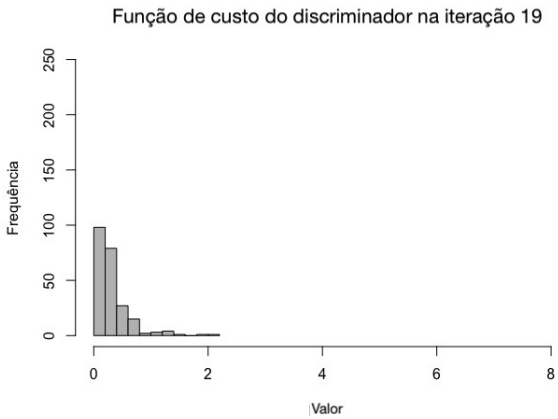

Função de custo do discriminador na iteração 14
Frequência
Valor

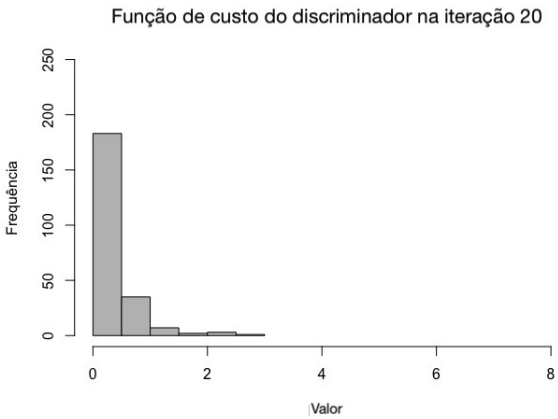

Função de custo do discriminador na iteração 15
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

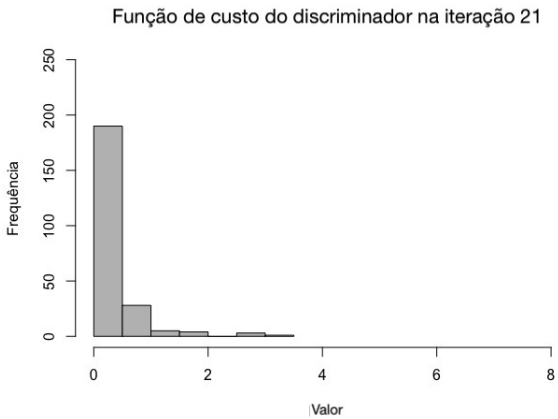

Função de custo do discriminador na iteração 16
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

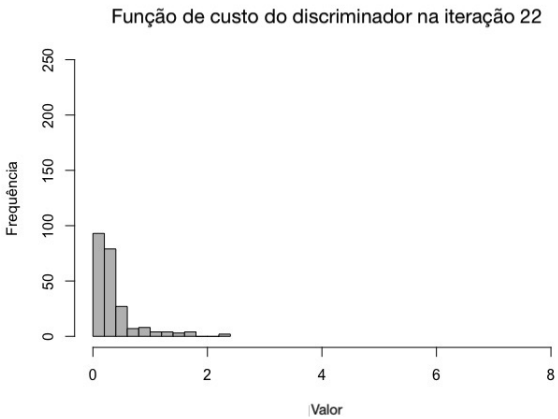

Função de custo do discriminador na iteração 17
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor


Função de custo do discriminador na iteração 18
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor


Função de custo do discriminador na iteração 19
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor


Função de custo do discriminador na iteração 20
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor


Função de custo do discriminador na iteração 21
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor


Função de custo do discriminador na iteração 22
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

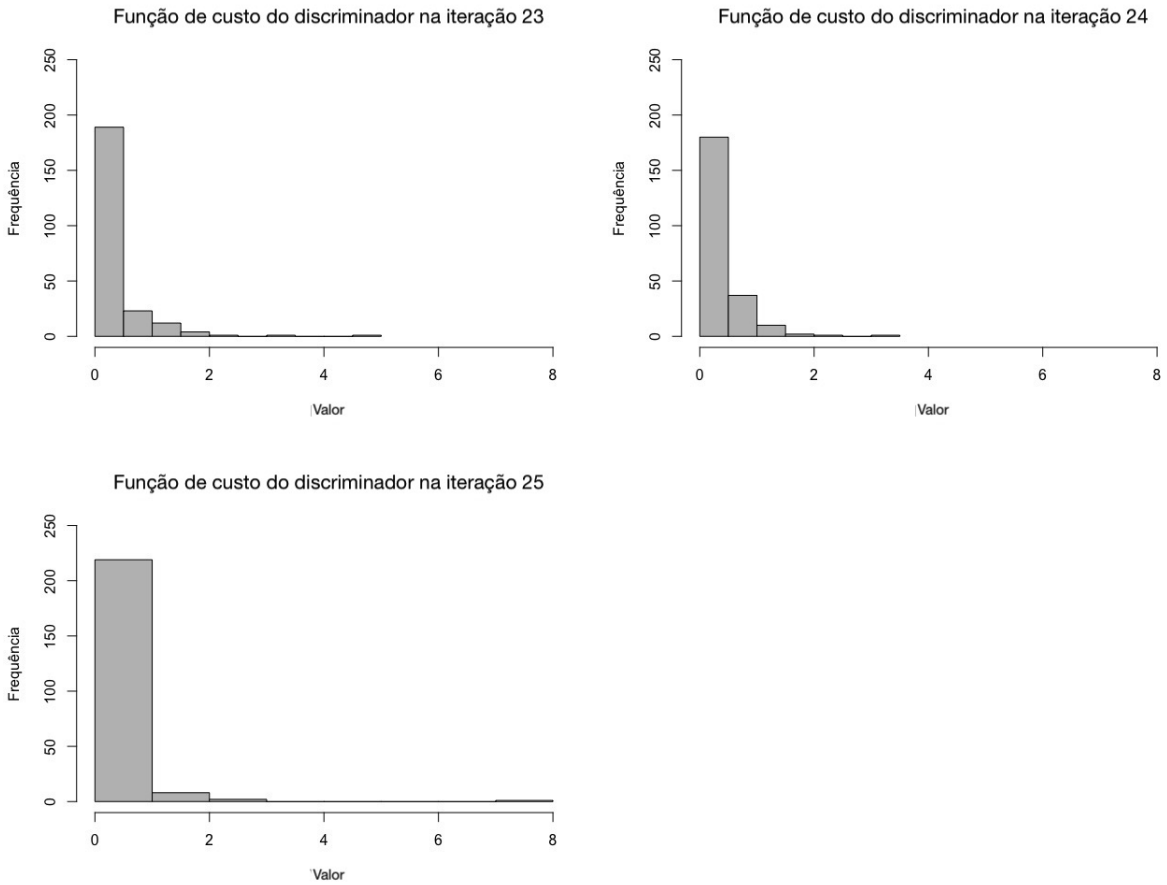
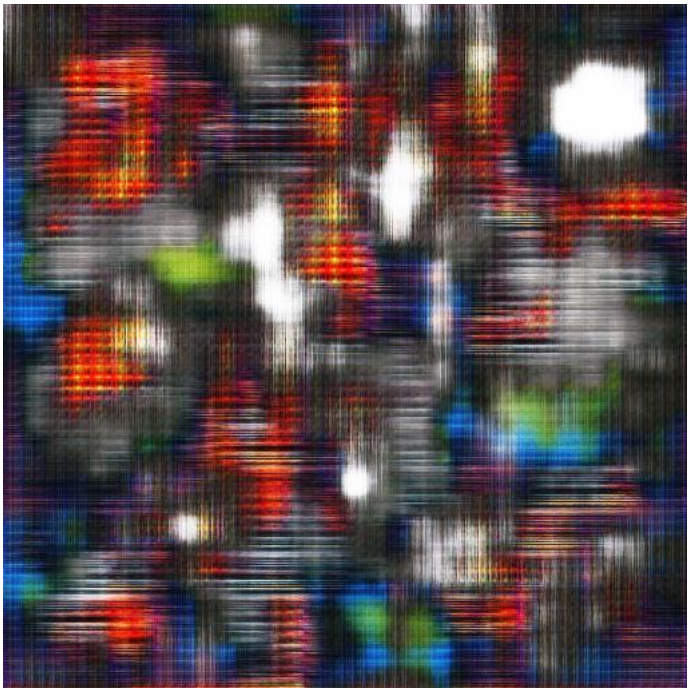
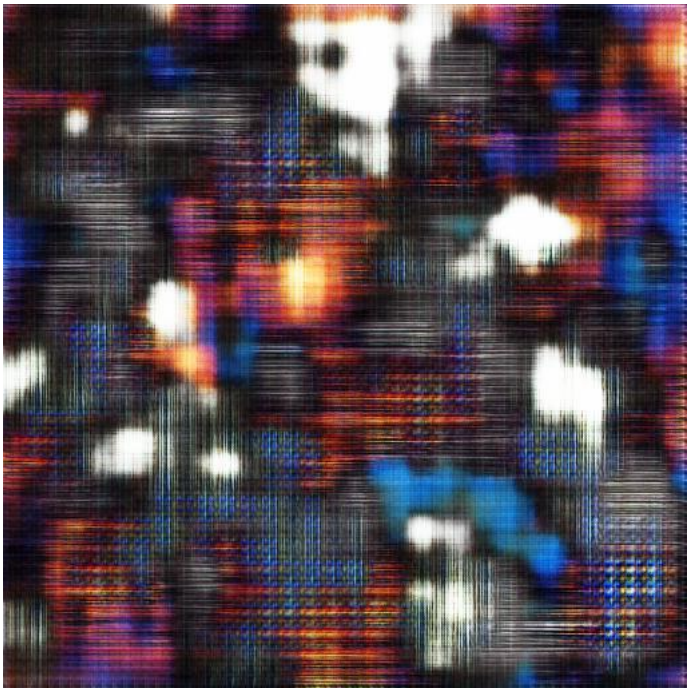
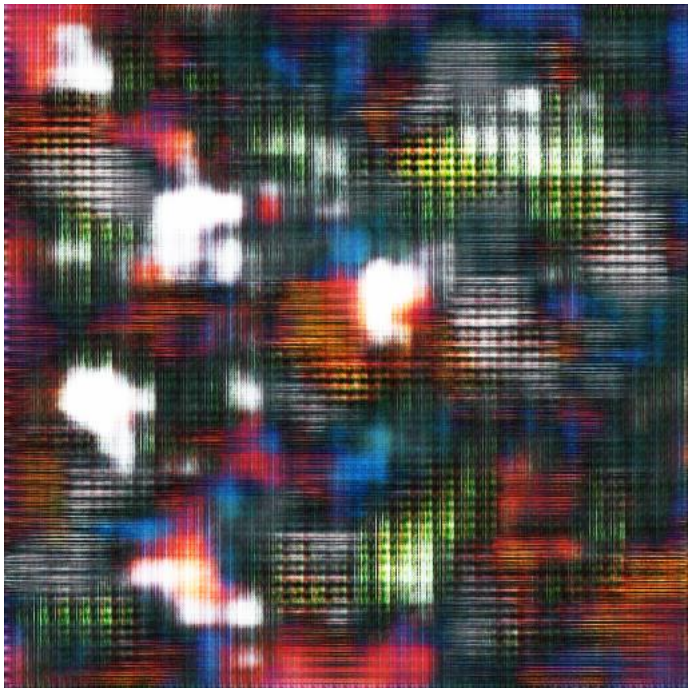


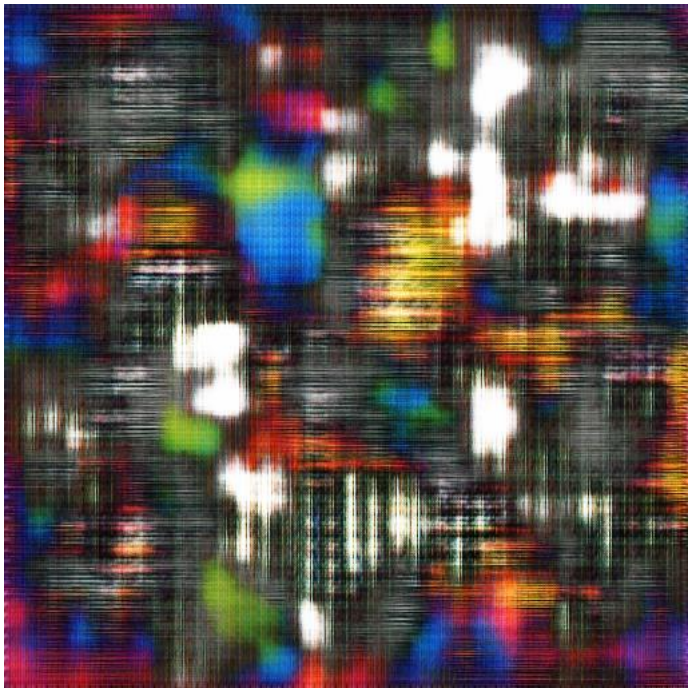
Figura A.3 – Histogramas da função de custo do discriminador a cada etapa de treinamento do primeiro experimento

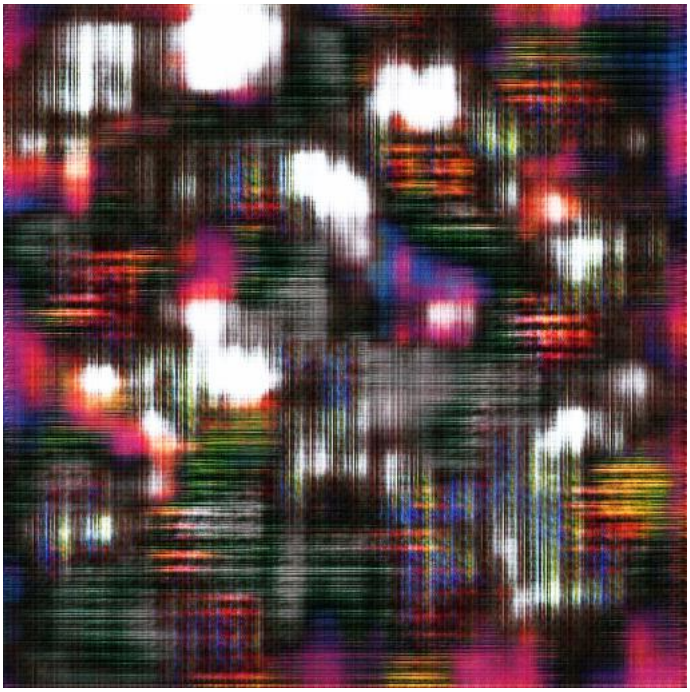
A.2.3 Resultados obtidos

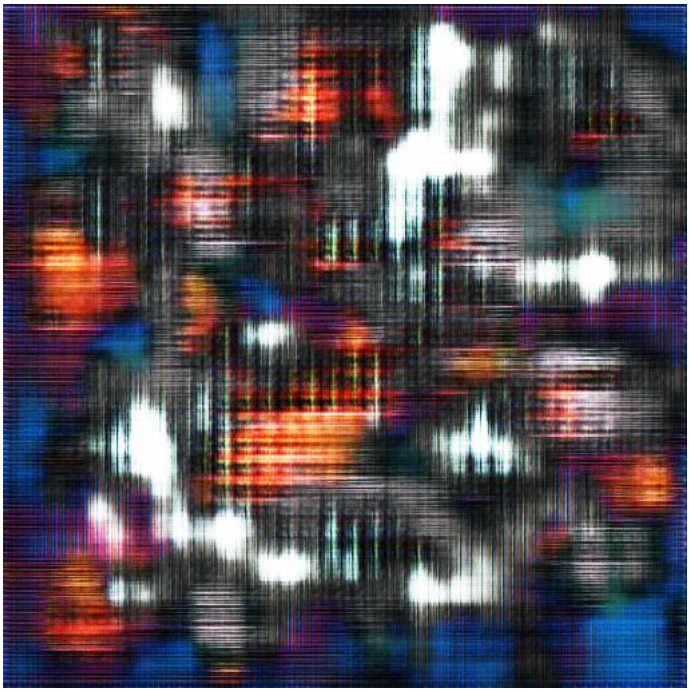


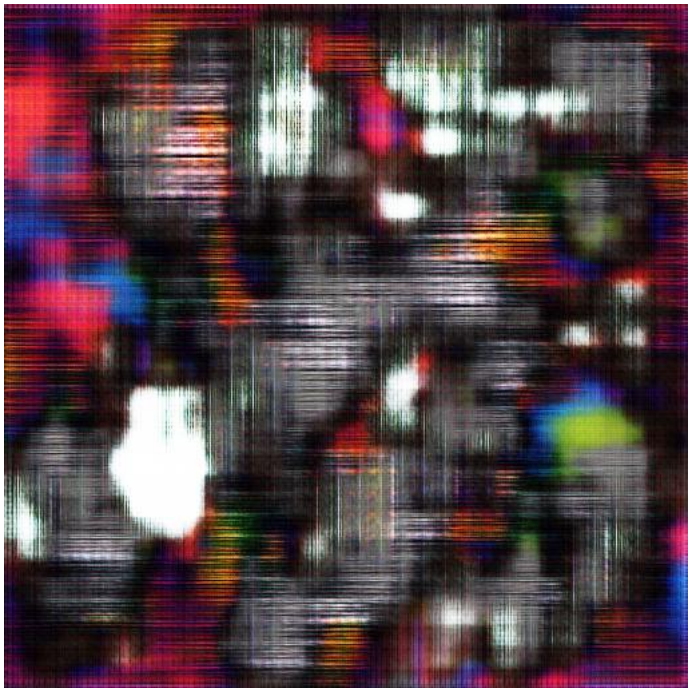


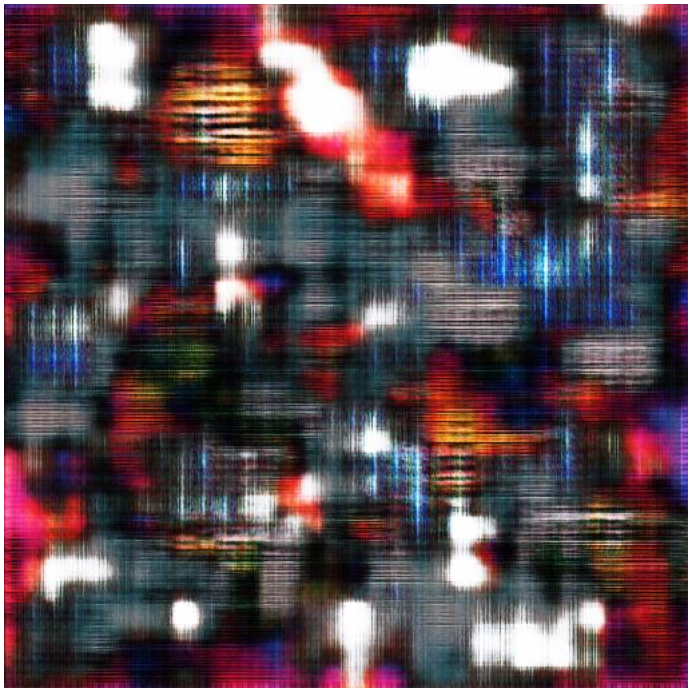


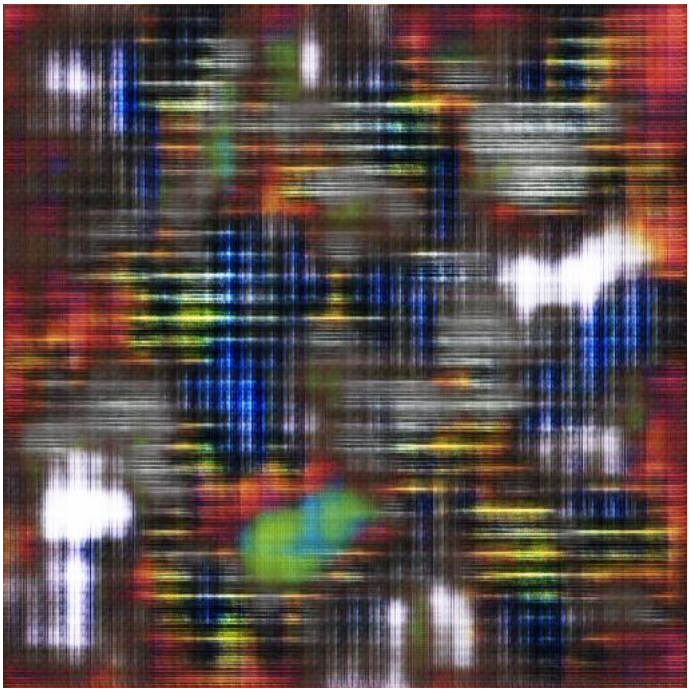












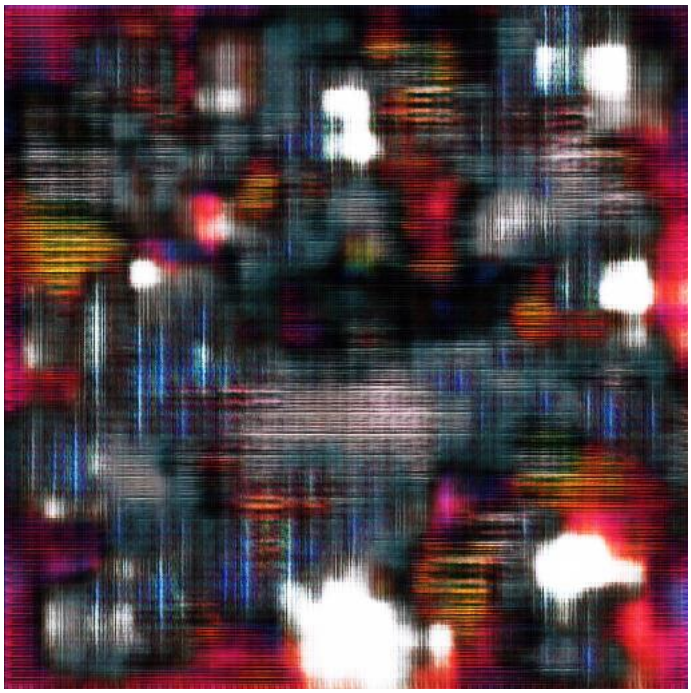
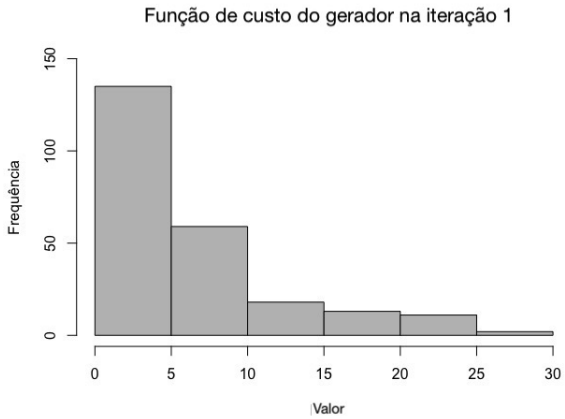
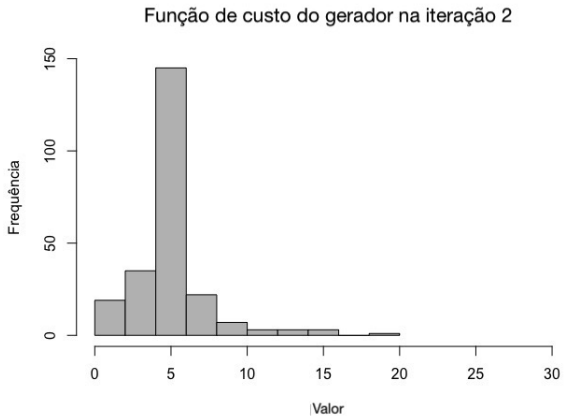


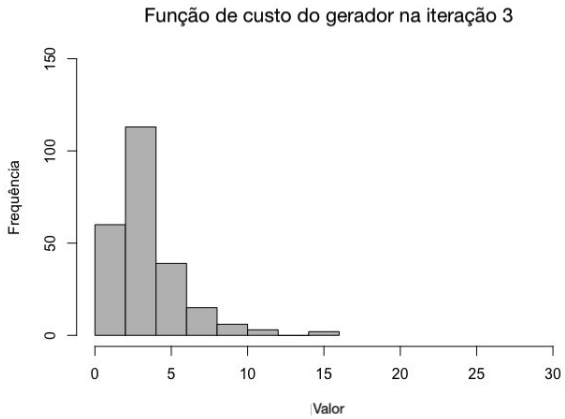
Figura A.4 – Resultados obtidos após primeiro treinamento

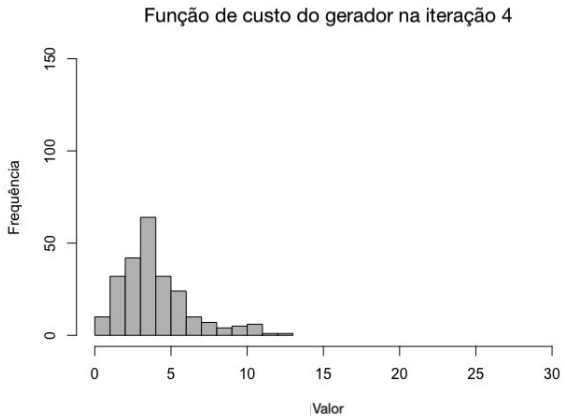
A.3 Segundo experimento (E2)

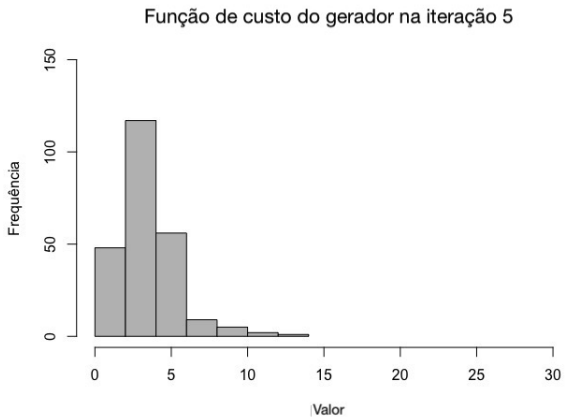
A.3.1 Histogramas da função de custo do gerador

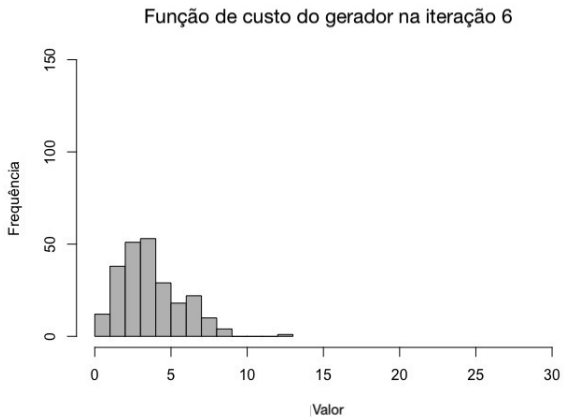


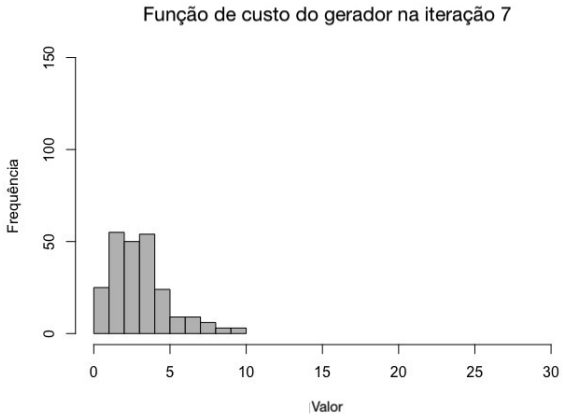


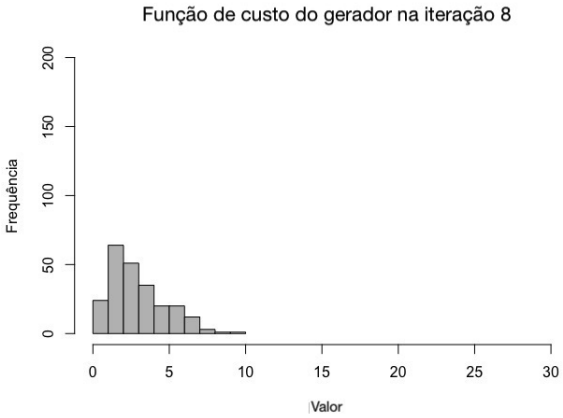


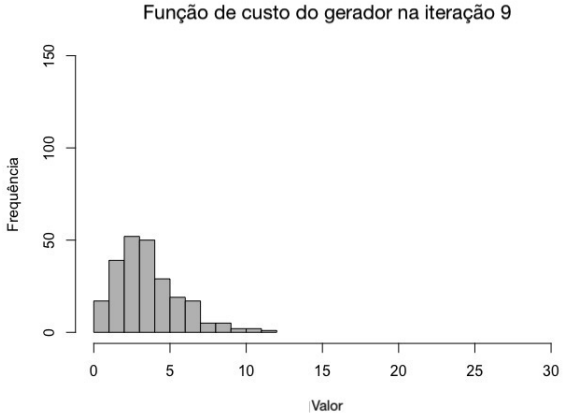


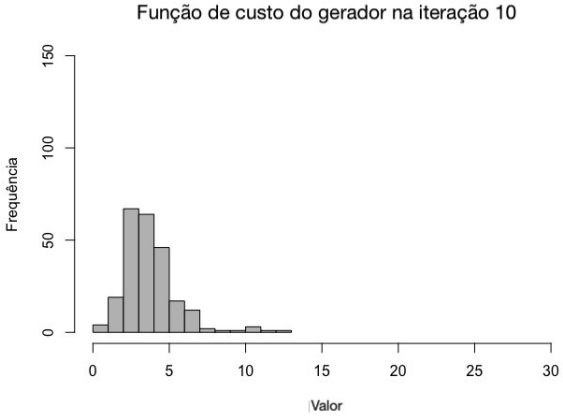


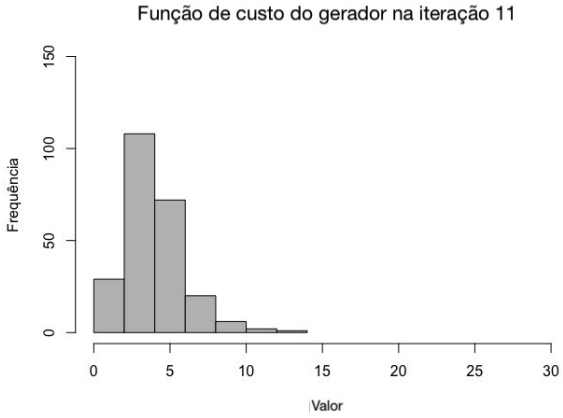


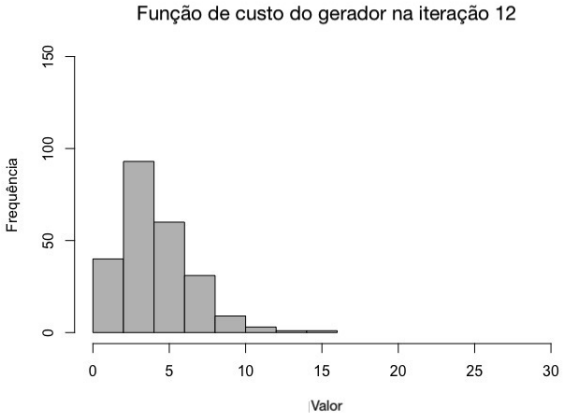

Função de custo do gerador na iteração 7
Frequência
Valor

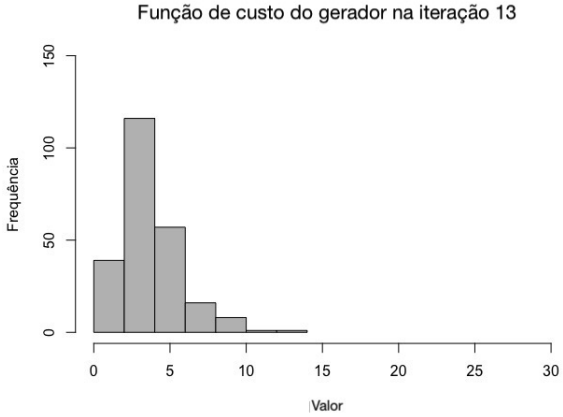

Função de custo do gerador na iteração 8
Frequência
Valor

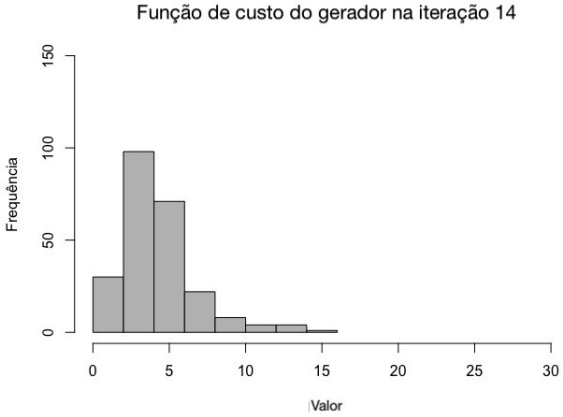

Função de custo do gerador na iteração 9
Frequência
Valor

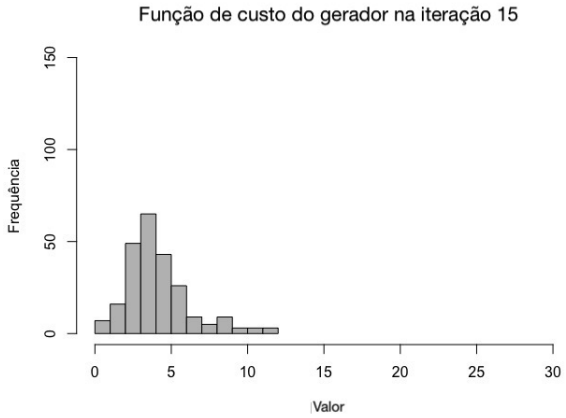

Função de custo do gerador na iteração 10
Frequência
Valor

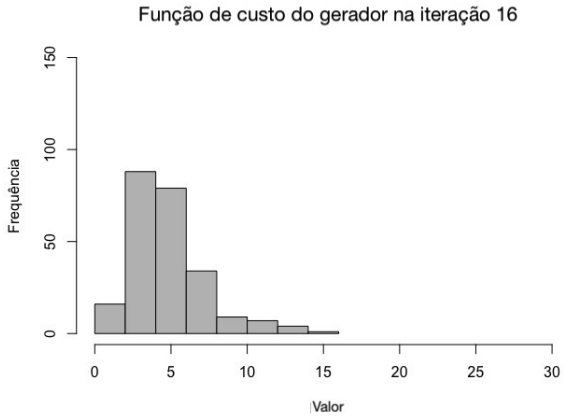

Função de custo do gerador na iteração 11
Frequência
Valor

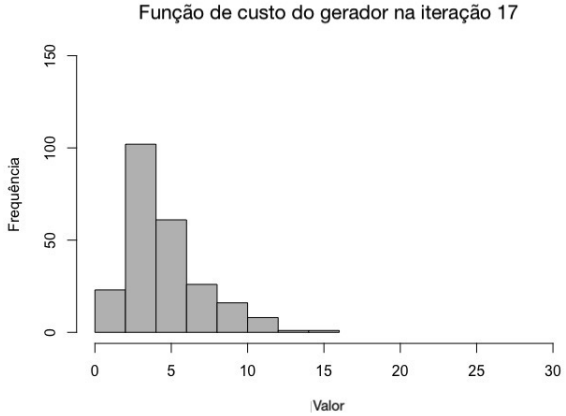

Função de custo do gerador na iteração 12
Frequência
Valor

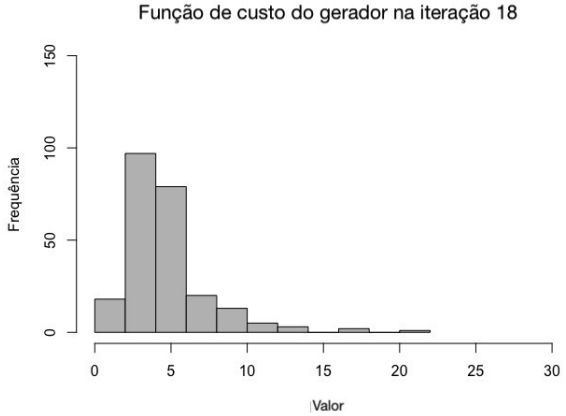

Função de custo do gerador na iteração 13
Frequência
Valor

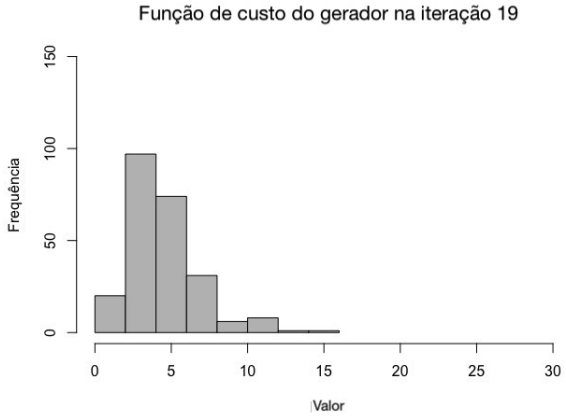

Função de custo do gerador na iteração 14
Frequência
Valor

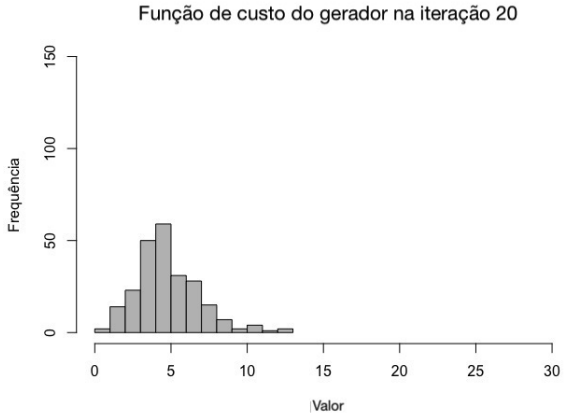

Função de custo do gerador na iteração 15
Frequência
Valor

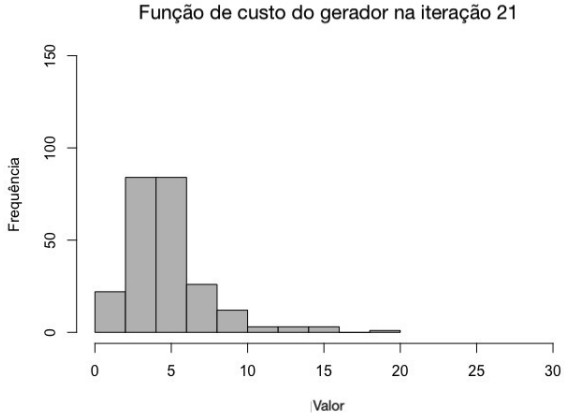

Função de custo do gerador na iteração 16
Frequência
Valor

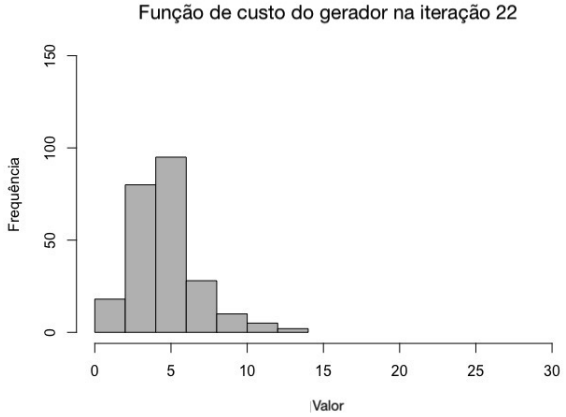

Função de custo do gerador na iteração 17
Frequência
Valor


Função de custo do gerador na iteração 18
Frequência
Valor


Função de custo do gerador na iteração 19
Frequência
Valor


Função de custo do gerador na iteração 20
Frequência
Valor


Função de custo do gerador na iteração 21
Frequência
Valor


Função de custo do gerador na iteração 22
Frequência
Valor

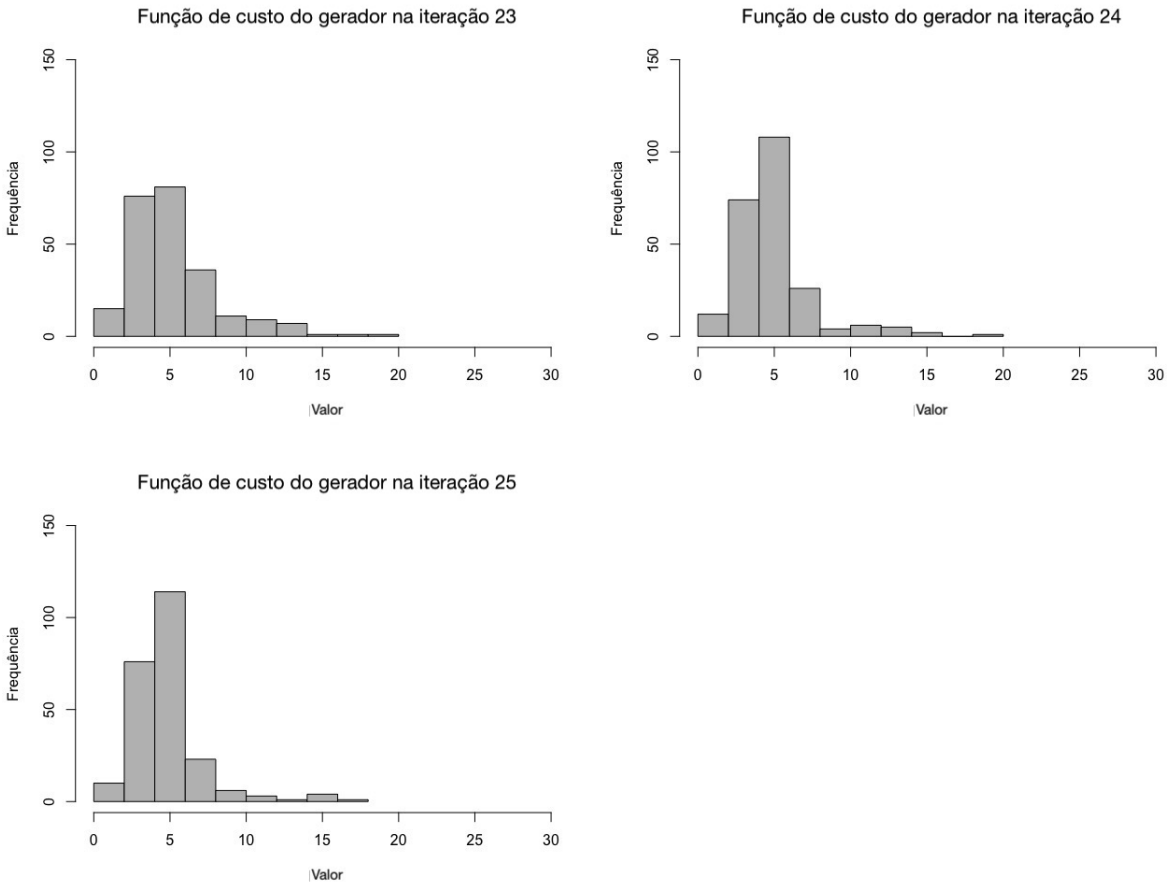
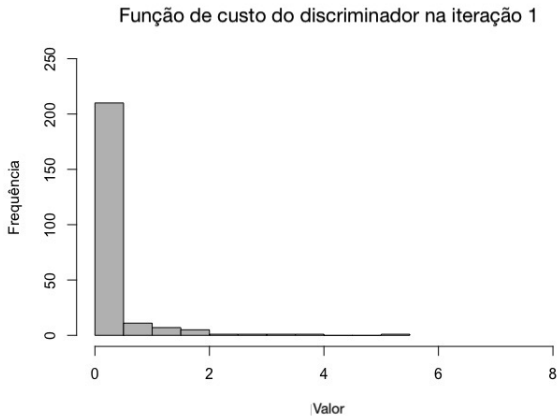
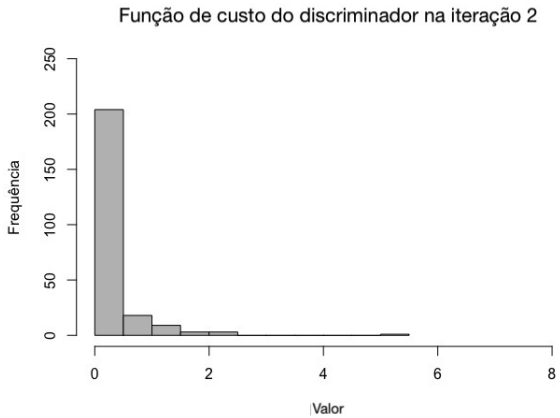
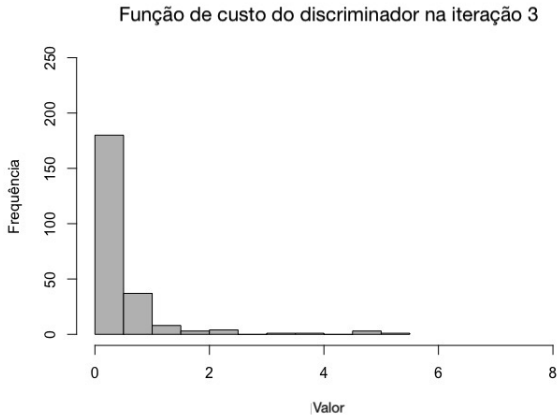


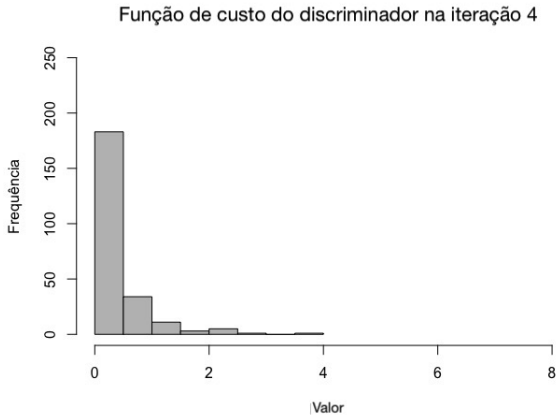
Figura A.5 – Histogramas da função de custo do gerador a cada etapa de treinamento do segundo experimento

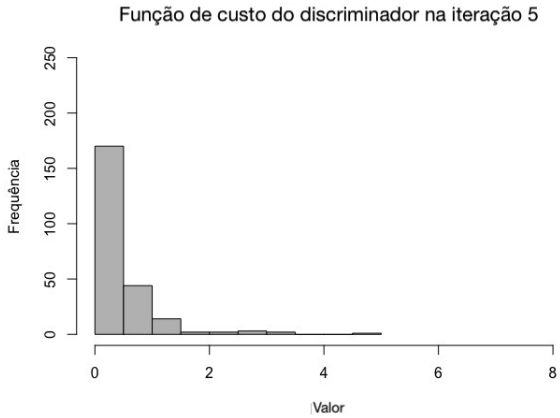
A.3.2 Histogramas da função de custo do discriminador

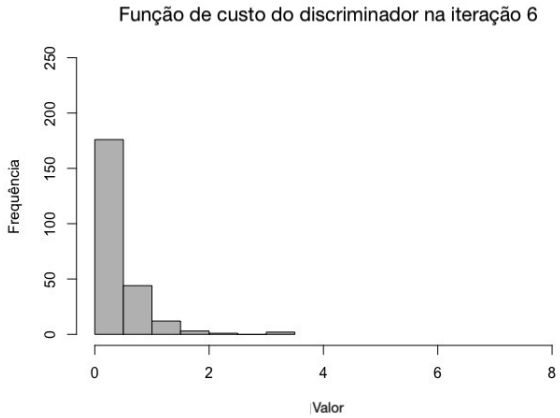


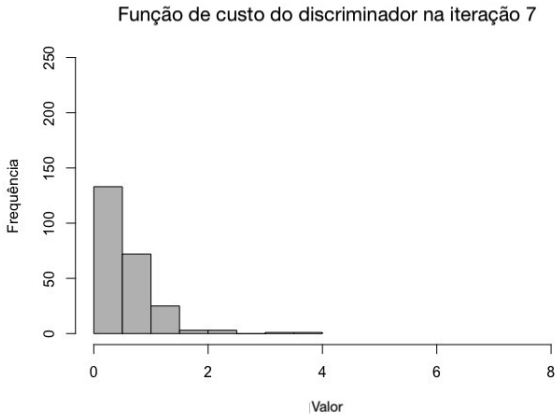


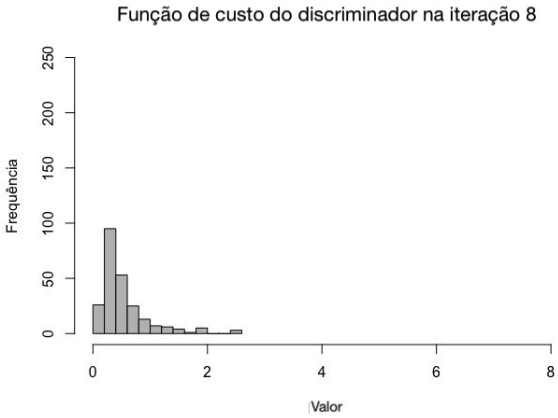


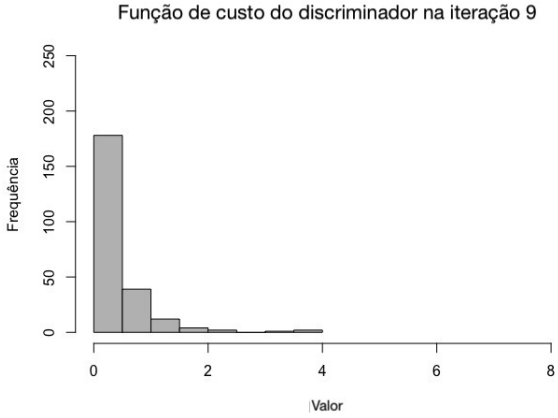


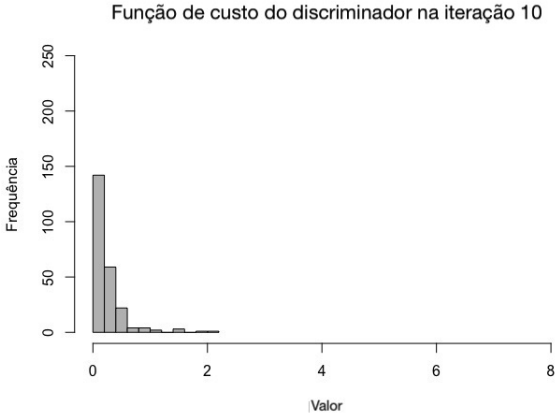


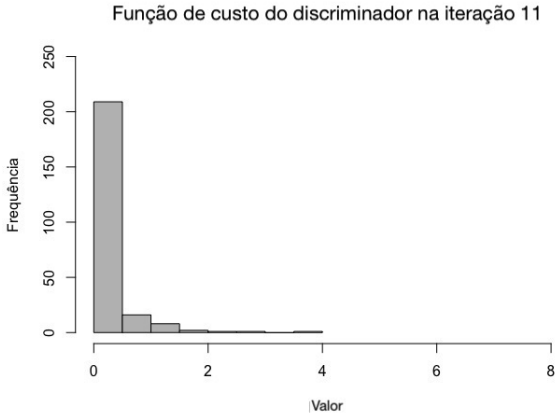


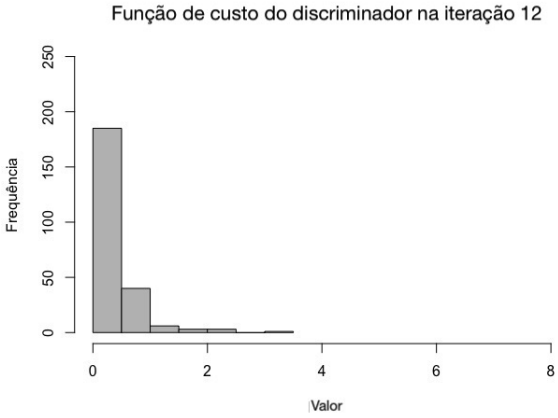

Função de custo do discriminador na iteração 7
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

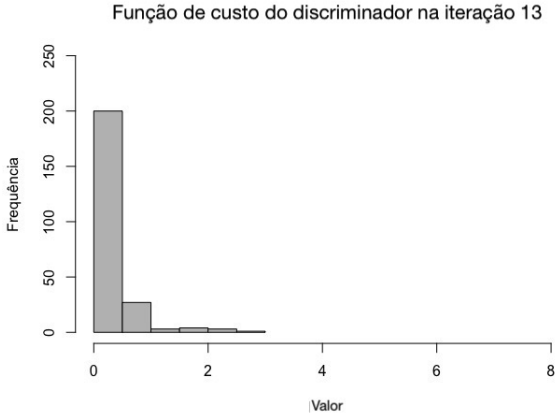

Função de custo do discriminador na iteração 8
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

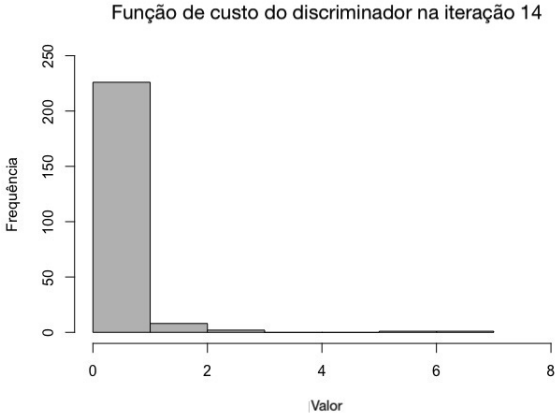

Função de custo do discriminador na iteração 9
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

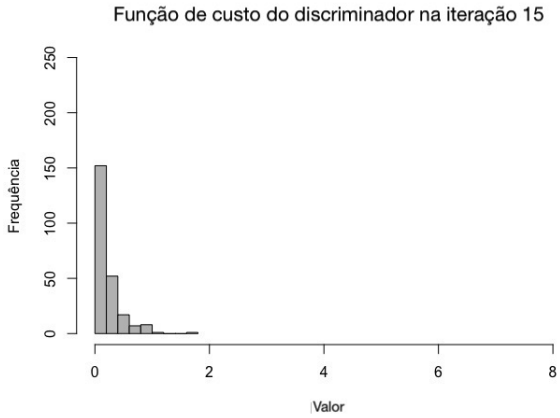

Função de custo do discriminador na iteração 10
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

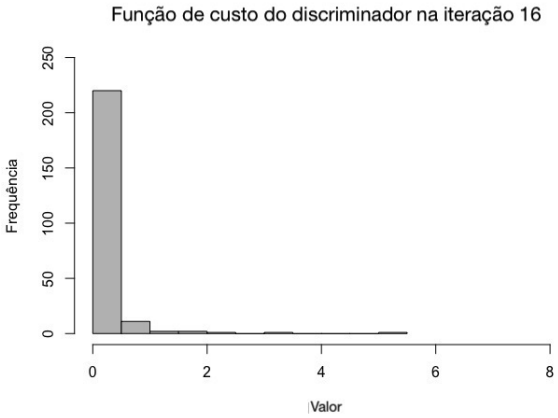

Função de custo do discriminador na iteração 11
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

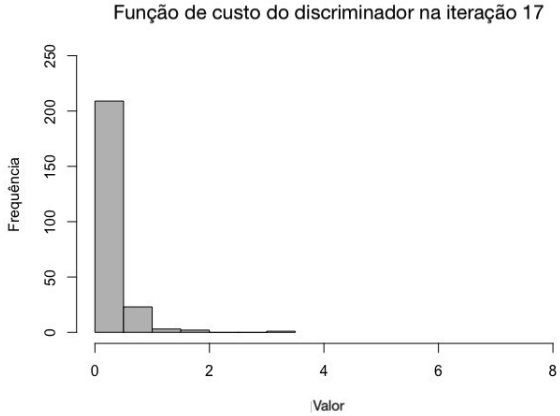

Função de custo do discriminador na iteração 12
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

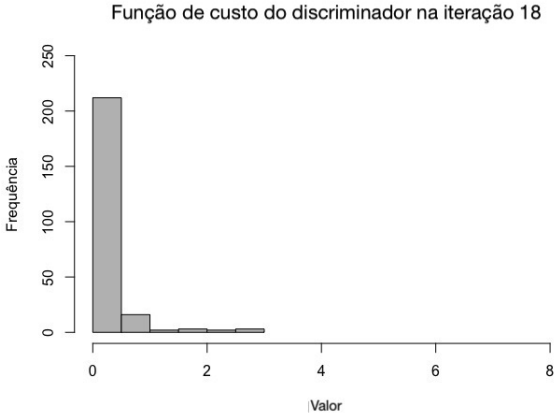

Função de custo do discriminador na iteração 13
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

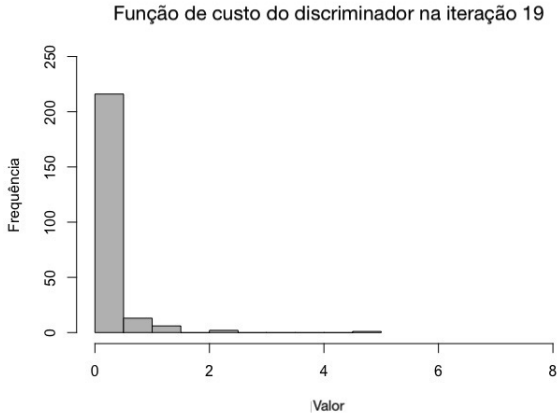

Função de custo do discriminador na iteração 14
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

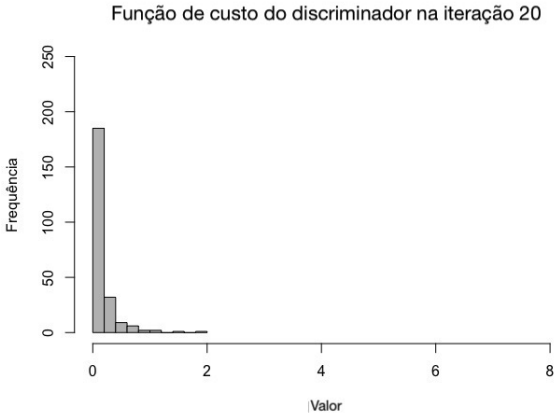

Função de custo do discriminador na iteração 15
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

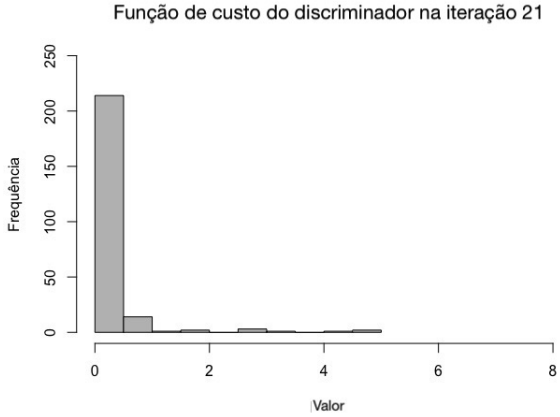

Função de custo do discriminador na iteração 16
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

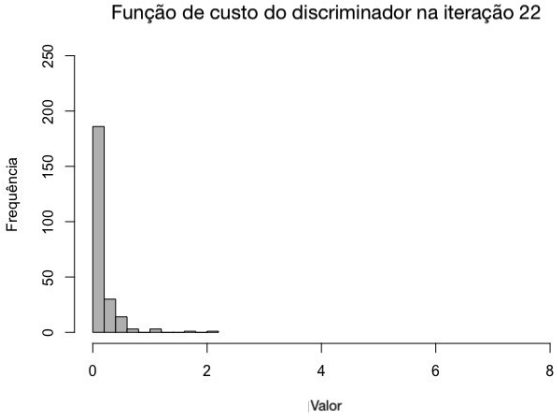

Função de custo do discriminador na iteração 17
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor


Função de custo do discriminador na iteração 18
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor


Função de custo do discriminador na iteração 19
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor


Função de custo do discriminador na iteração 20
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor


Função de custo do discriminador na iteração 21
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor


Função de custo do discriminador na iteração 22
Frequência
0
50
100
150
200
250
0
2
4
6
8
Valor

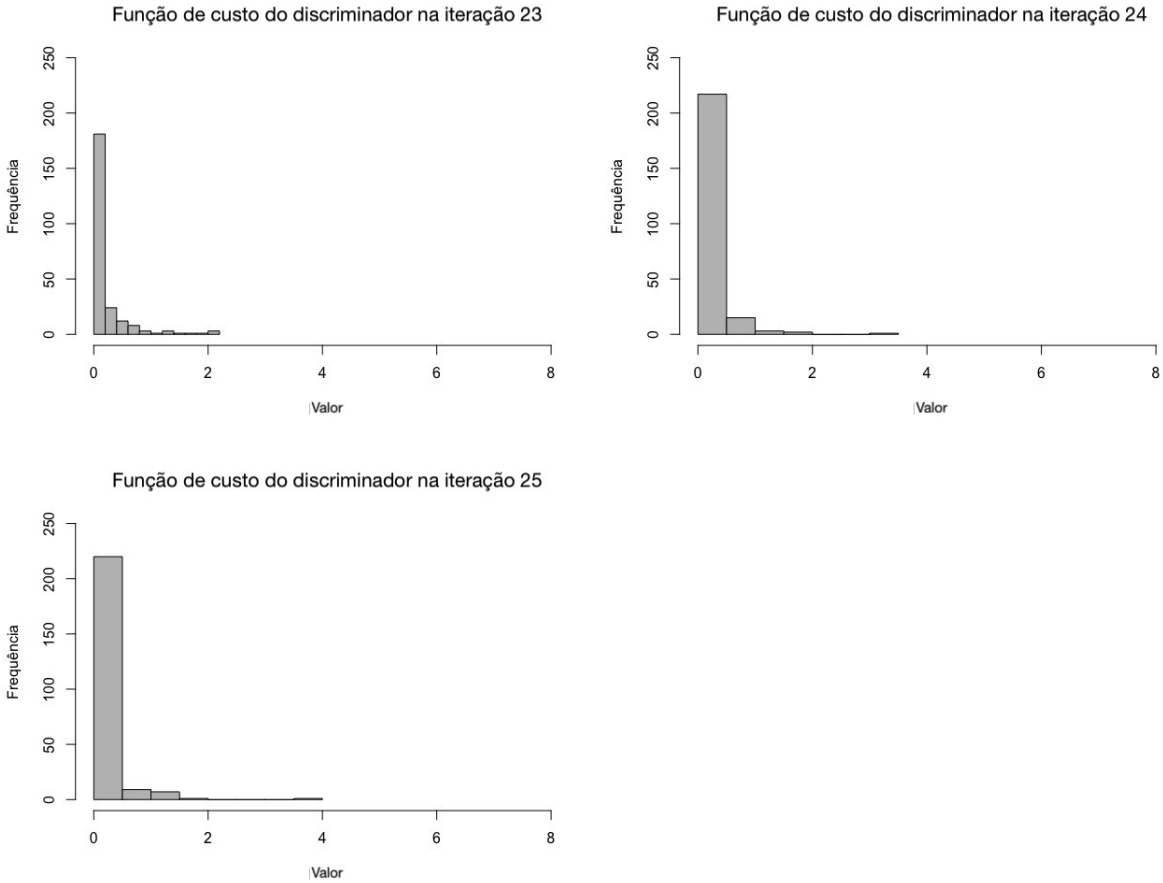
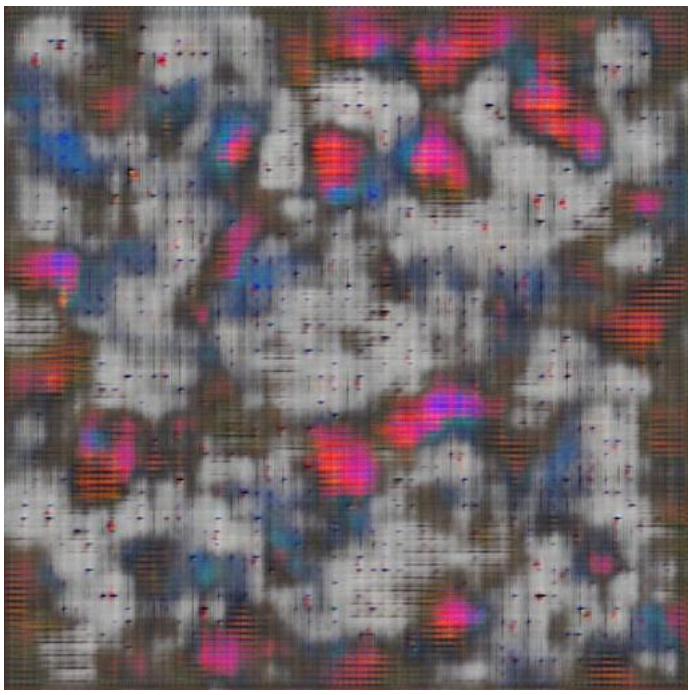
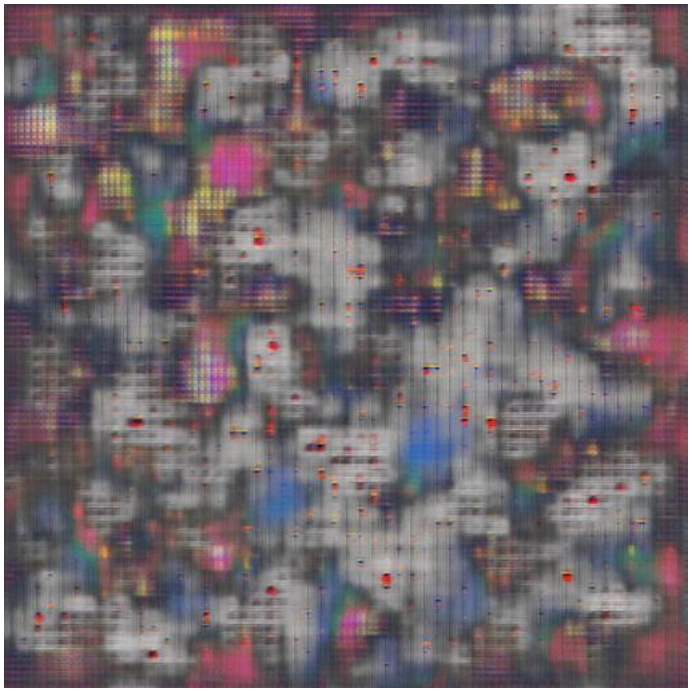
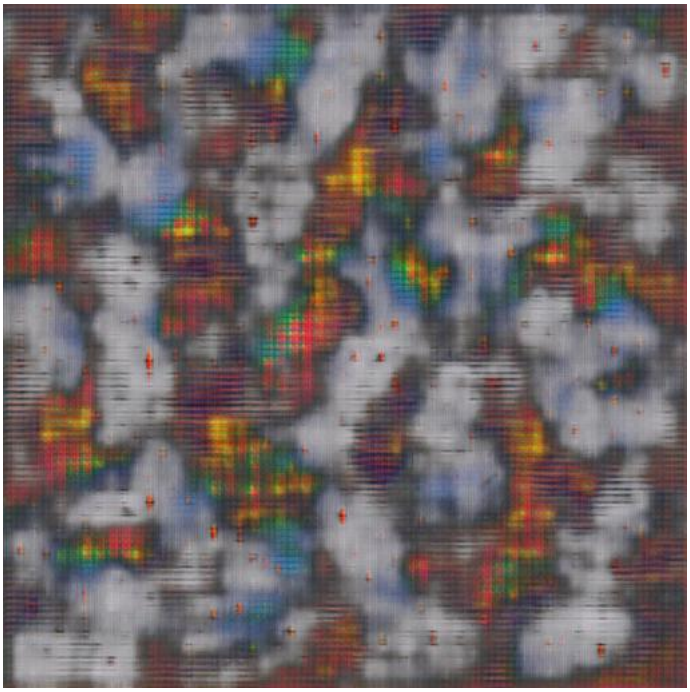


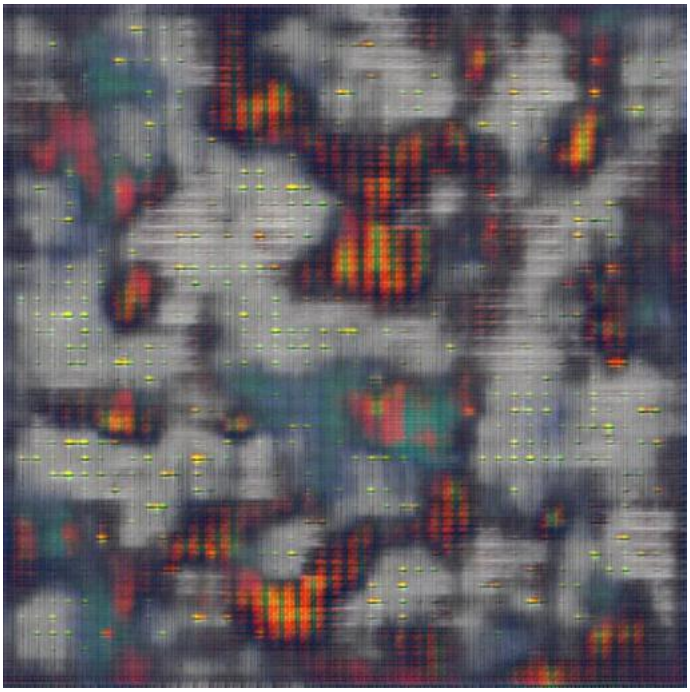
Figura A.6 – Histogramas da função de custo do discriminador a cada etapa de treinamento do segundo experimento

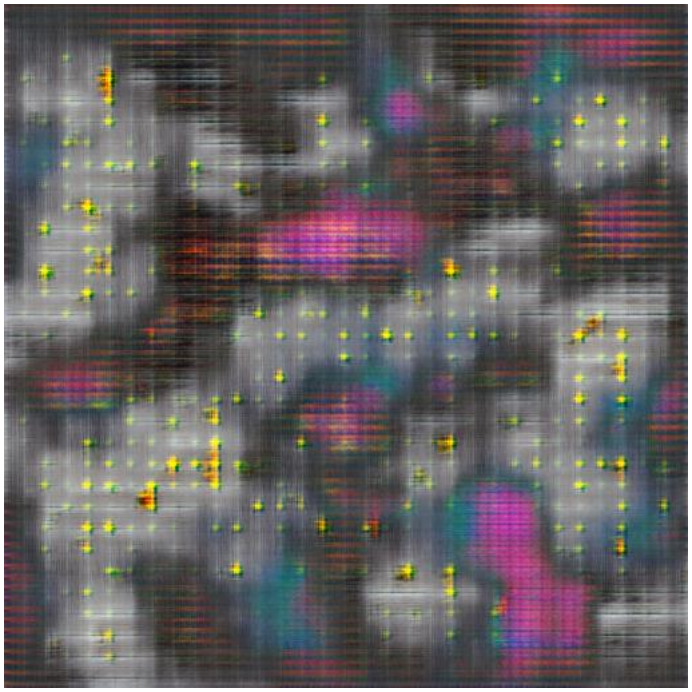
A.3.3 Resultados obtidos

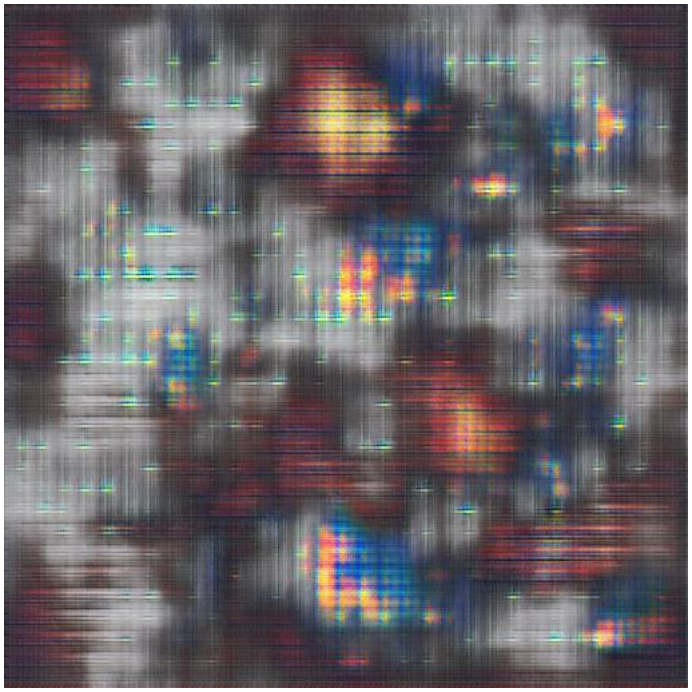


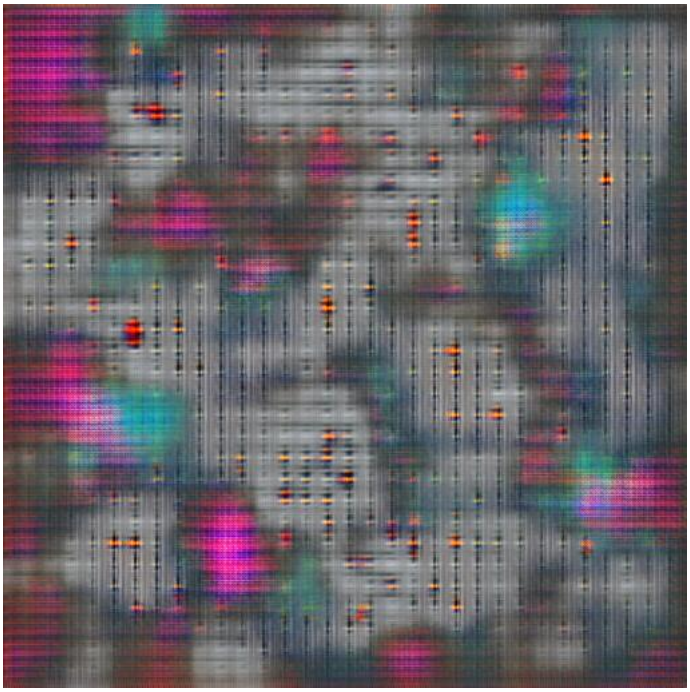


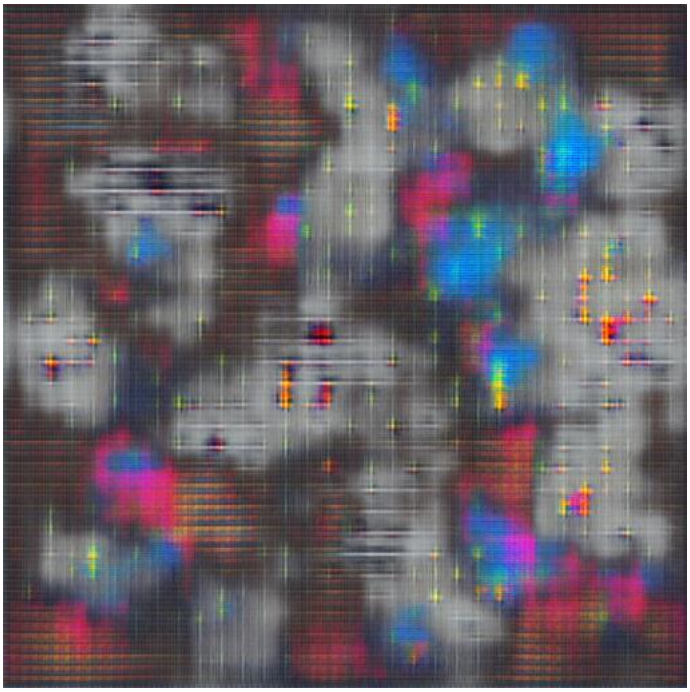


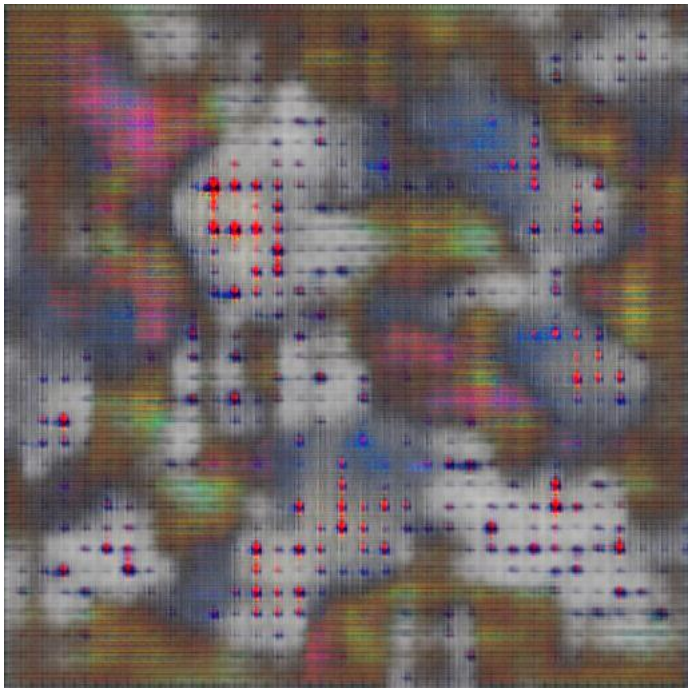












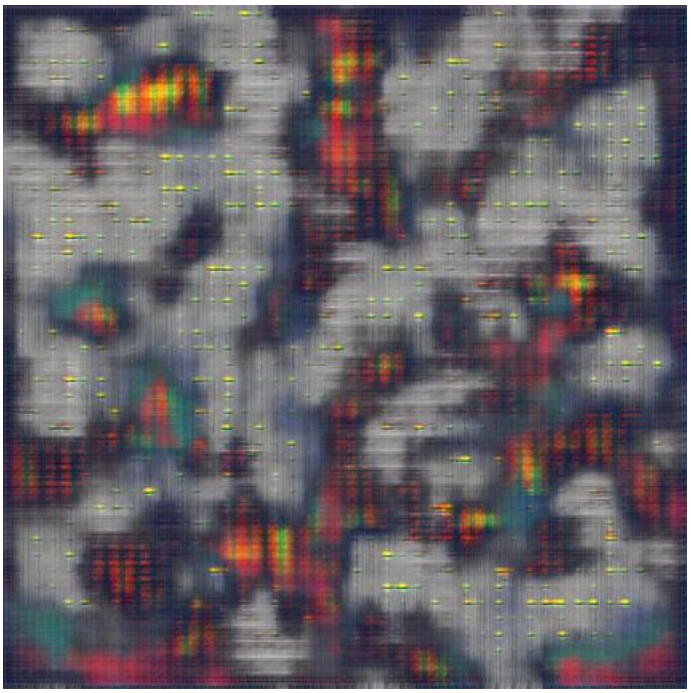


Figura A.7 – Resultados obtidos após segundo treinamento

Referências

[1] S. DiPaola and L. Gabora, "Incorporating characteristics of human creativity into an evolutionary art algorithm." http://ivizlab.sfu.ca/papers/genprog2009.pdf. Acessado: 2018-04-16. Citado na página 13.

[2] M. A. Boden, *The creative mind: Myths and mechanisms*. Routledge, 2nd ed., 2004. Citado na página 13.

[3] D. K. Simonton, *Origins of Genius: Darwinian Perspectives on Creativity*. Oxford University Press, 1st ed., 1999. Citado na página 13.

[4] A. E. et al, "Can: Creative adversarial networks generating "art" by learning about styles and deviating from style norms." https://arxiv.org/pdf/1706.07068.pdf, 2017. Acessado: 2018-04-16. Citado na página 13.

[5] P. Thagard and T. C. Stewart, "The aha! experience: Creativity through emergent binding in neural networks." https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1551-6709.2010.01142.x, 2010. Acessado: 2018-04-16. Citado na página 13.

[6] A. S. E. Leon A. Gatys and M. Bethge, "A neural algorithm of artistic style." https://arxiv.org/pdf/1508.06576.pdf, 2015. Acessado: 2018-04-16. Citado 2 vezes nas páginas 13 e 20.

[7] F. LDN, "Jê américo, o arquiteto da arte." https://www.focusldn.com/brazilian-interviews/je-americo, 2018. Acessado: 2018-11-17. Citado na página 16.

[8] I. L. Maxime Oquab, Leon Bottou and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks." http://openaccess.thecvf.com/content_cvpr_2014/papers/Oquab_Learning_and_Transferring_2014_CVPR_paper.pdf, 2014. Acessado: 2018-05-22. Citado na página 20.

[9] M. M. B. X. D. W.-F. S. O. A. C. Ian J. Goodfellow, Jean Pouget-Abadie and Y. Bengio, "Generative adversarial nets." http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf, 2014. Acessado: 2018-05-22. Citado 3 vezes nas páginas 23, 24 e 25.

[10] L. M. Alec Radford and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks." https://arxiv.org/pdf/1511.06434.pdf, 2016. Acessado: 2018-10-10. Citado 3 vezes nas páginas 25, 26 e 43.

[11] T. Brosch and R. Tam, "Efficient training of convolutional deep belief networks in the frequency domain for application to high-resolution 2d and 3d images." https://www.mitpressjournals.org/doi/abs/10.1162/NECO_a_00682, 2015. Acessado: 2019-01-25. Citado na página 25.

[12] Q. C. Min Lin and S. Yan, "Network in network." https://arxiv.org/pdf/1312.4400.pdf, 2014. Acessado: 2019-01-25. Citado na página 25.

[13] S. Chintala, "Dcgan.torch: Train your own image generator." https://github.com/soumith/dcgan.torch, 2015. Citado 3 vezes nas páginas 26, 30 e 31.