TNI_GAN

April 27, 2022

Vérifions que nous disposons d'un GPU nvidia (CUDA)

[1]:
```
!nvidia-smi
```

```
Tue Apr 26 22:38:05 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 496.76       Driver Version: 496.76       CUDA Version: 11.5     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce … WDDM  | 00000000:01:00.0  On |                  N/A |
|  0%   43C    P8    11W / 200W |    900MiB /  8192MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A       868    C+G   …ser\Application\brave.exe    N/A      |
|    0   N/A  N/A      1640    C+G   Insufficient Permissions       N/A      |
|    0   N/A  N/A      3808    C+G   …b3d8bbwe\ScreenSketch.exe    N/A      |
|    0   N/A  N/A      4472    C+G   …y\ShellExperienceHost.exe    N/A      |
|    0   N/A  N/A      5072    C+G   C:\Windows\explorer.exe         N/A      |
|    0   N/A  N/A      5348    C+G   …\app-1.0.9004\Discord.exe    N/A      |
|    0   N/A  N/A      9788    C+G   …artMenuExperienceHost.exe    N/A      |
|    0   N/A  N/A      9896    C+G   …n1h2txyewy\SearchHost.exe    N/A      |
|    0   N/A  N/A     10424    C+G   …perience\NVIDIA Share.exe    N/A      |
|    0   N/A  N/A     11788    C+G   …wekyb3d8bbwe\Video.UI.exe    N/A      |
|    0   N/A  N/A     12260    C+G   …ekyb3d8bbwe\YourPhone.exe    N/A      |
|    0   N/A  N/A     14144    C+G   …urrent\LogiOptionsMgr.exe    N/A      |
|    0   N/A  N/A     14624    C+G   Insufficient Permissions       N/A      |
|    0   N/A  N/A     14712    C+G   …werToys.PowerLauncher.exe    N/A      |
|    0   N/A  N/A     14832    C+G   …2txyewy\TextInputHost.exe    N/A      |
|    0   N/A  N/A     15392    C+G   …e\Current\LogiOverlay.exe    N/A      |
```

```
|    0   N/A  N/A     15440    C+G   …EarTrumpet\EarTrumpet.exe    N/A      |
|    0   N/A  N/A     16656    C+G   …8wekyb3d8bbwe\Cortana.exe    N/A      |
|    0   N/A  N/A     17012    C+G   …in7x64\steamwebhelper.exe    N/A      |
|    0   N/A  N/A     17028    C+G   …er_engine\wallpaper32.exe    N/A      |
|    0   N/A  N/A     19116    C+G   …t\Teams\current\Teams.exe    N/A      |
|    0   N/A  N/A     19380    C+G   …lPanel\SystemSettings.exe    N/A      |
|    0   N/A  N/A     20972    C+G   …fyw5nnt\app\Messenger.exe    N/A      |
|    0   N/A  N/A     22688    C+G   …t\Teams\current\Teams.exe    N/A      |
|    0   N/A  N/A     24536    C+G   …ekyb3d8bbwe\HxOutlook.exe    N/A      |
|    0   N/A  N/A     26600    C+G   …\app-1.0.9004\Discord.exe    N/A      |
+-----------------------------------------------------------------------------+
```

Imports

[2]:
```
import numpy as np
from tensorflow.keras.models import Sequential, Model, load_model
from tensorflow.keras.layers import Conv2DTranspose, Conv2D,␣
 ↪BatchNormalization, LeakyReLU, Activation, Flatten, Dense, Reshape, Dropout,␣
 ↪Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

Fichiers

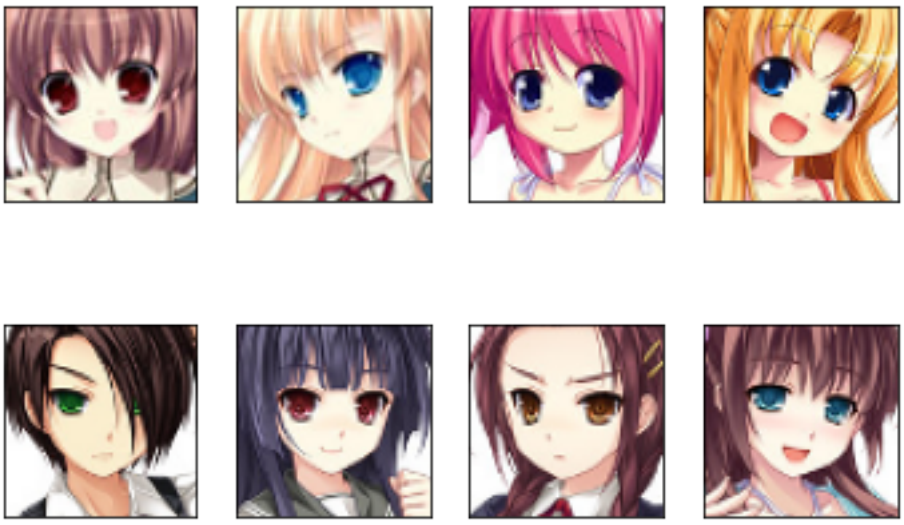
[ ]:
```
# téléchargement direct du dataset (recommandé mais un peu long)  -  UNIX ONLY
!wget --no-check-certificate -O faces.zip "https://onedrive.live.com/download?
 ↪cid=649AF7D49C493C3C&resid=649AF7D49C493C3C%215719&authkey=ALaohQ5arSsPBPI"
```

Extraction des fichiers du dataset dans notre environnement local

[ ]:
```
# UNZIP du fichier téléchargé - UNIX ONLY
!unzip  -o faces.zip -d faces &> /dev/null && echo "Fichiers extraits dans le␣
 ↪dossier faces"
```

```
Fichiers extraits dans le dossier /content/faces
```

[3]:
```
import matplotlib as mat
import matplotlib.pyplot as plt
for i in range(8):
    plt.subplot(2,4,i+1)
    img = plt.imread("faces/data/550"+str(i)+".png")
    plt.imshow(img)
    plt.xticks([])
    plt.yticks([])
```



1 Définissons le GAN

1.0.1 1. Discriminateur

[4]:
```
def define_Discriminator(in_shape=(64,64,3)):
  model = Sequential()
  model.add(Conv2D(64, (3,3), padding='same', input_shape=in_shape))
  model.add(LeakyReLU(alpha=0.2))
  model.add(Conv2D(64, (3,3), strides=(2, 2), padding='same'))
  model.add(LeakyReLU(alpha=0.2))
  model.add(Conv2D(64, (3,3), strides=(2, 2), padding='same'))
  model.add(LeakyReLU(alpha=0.2))
  model.add(Flatten())
  model.add(Dropout(0.4))
  model.add(Dense(1, activation='sigmoid'))
  optimizer = Adam(lr=OPT_LR, beta_1=0.5)
  model.compile(loss='binary_crossentropy', optimizer=optimizer,␣
 ↪metrics=['accuracy'])
  return model
```

1.0.2 2. Générateur

[5]:
```
def define_Generator(latent_dim):
  model_G = Sequential()
  #16x16 image
  n_nodes =  16 * 16 * 256
  model_G.add(Dense(n_nodes, input_dim=latent_dim))
  model_G.add(LeakyReLU(alpha=0.2))
```

```
  model_G.add(Reshape((16, 16, 256)))
  #Upsample to 32x32
  model_G.add(Conv2DTranspose(128, (4,4), strides=(2,2), padding='same'))
  model_G.add(LeakyReLU(alpha=0.2))
  #Upsample to 64x64
  model_G.add(Conv2DTranspose(128, (4,4), strides=(2,2), padding='same'))
  model_G.add(LeakyReLU(alpha=0.2))
  model_G.add(Conv2D(3, (7,7), activation='tanh', padding='same'))
  return model_G
```

1.0.3 3. GAN

[6]:
```
def define_GAN(model_G, model_D):
        #Discriminator's weights not trainable
        model_D.trainable = False
        model = Sequential()
 #Adding both models in an only one
        model.add(model_G)
        model.add(model_D)
        optimizer = Adam(lr=OPT_LR, beta_1=0.5)
        model.compile(loss='binary_crossentropy', optimizer=optimizer)
        return model
```

Fonctions utilitaires

[7]:
```
def load_real_images():
  datagen = ImageDataGenerator(rescale=1./255)
  X = datagen.flow_from_directory('faces', target_size= (64,64),␣
 ↪batch_size=8192, class_mode='binary')
  #Load images in a list and then return it as an array
  img_list = []
  batch_index = 0
  while batch_index <= X.batch_index:
    data = X.next()
    img_list.append(data[0])
    batch_index += 1
  img_array = np.asarray(img_list)
  return img_array
```

[8]:
```
# Génere n points aléatoires sur le vecteur de dimension latent_dim
def generate_latent_points(latent_dim, n_samples):
    X = np.random.randn(latent_dim * n_samples)
    X = X.reshape(n_samples, latent_dim)
    return X

# Génère n images aléatoires
def generate_fake_images(model_G, latent_dim, n_samples):
```

```
    X_input = generate_latent_points(latent_dim, n_samples)
    X = model_G.predict(X_input)
    y = np.zeros((n_samples, 1))
    return X, y

def get_real_images(dataset, n_samples):
    i = np.random.randint(0, dataset.shape[0], n_samples)
    X = dataset[i]
    y = np.ones((n_samples,1))
    return X, y
```

Fonctions d'affichages

[9]:
```
# Génère et affiche 5 images avec le générateur en paramètre
def plot_images(epoch_num, model, nb_imgs=5):
    X = generate_fake_images(model, LATENT_DIM, nb_imgs)
    print(X[0].shape)
    # plot the result
    plt.figure(figsize=(19, 19))
    i=1
    for img in X[0]:
      plt.subplot((epoch_num%nb_imgs)+1, nb_imgs, i)
      plt.imshow(np.clip((img*255), 0, 255).astype('uint8'))
      plt.title(f"époque {epoch_num}")
      i=i+1
    plt.show()

# Enregistre le modèle sur le drive et évalue la performance des modèles
def summarize_performance(epoch, model_G, model_D, dataset, latent_dim,␣
 ↪nb_of_samples=100):
  X_real, y_real = get_real_images(dataset, nb_of_samples)
  _, acc_real = model_D.evaluate(X_real, y_real, verbose=0)
  x_fake, y_fake = generate_fake_images(model_G, latent_dim, nb_of_samples)
  _, acc_fake = model_D.evaluate(x_fake, y_fake, verbose=0)
  print('Accuracy real: %.0f%%, fake: %.0f%%' % (acc_real*100, acc_fake*100))
```

Fonctions de training

[10]:
```
def train_GAN(model_G, model_D, model_GAN, dataset, latent_dim, nb_epochs=50,␣
 ↪nb_batch=128):
  batch_per_epoch = int(dataset.shape[0] / nb_batch)
  half_batch = int(nb_batch / 2)
        #Range for epochs enumeration
  for i in range(nb_epochs):
  #Enumerate batches over the training set
    for j in range(batch_per_epoch):
      X_real, y_real = get_real_images(dataset, half_batch)
```

```
      X_fake, y_fake = generate_fake_images(model_G, latent_dim, half_batch)
      #Concatenate fake and real images / labels after generating them
      X, y = np.vstack((X_real, X_fake)), np.vstack((y_real, y_fake))
      #Then training discriminator and GAN
      d_loss, _ = model_D.train_on_batch(X, y)
      X_gan = generate_latent_points(latent_dim, nb_batch)
      y_gan = np.ones((nb_batch, 1))
      g_loss = model_GAN.train_on_batch(X_gan, y_gan)
      if(j%50==0):
        print('%d, %d/%d, d=%.3f, g=%.3f' % (i+1, j+1, batch_per_epoch, d_loss,␣
↪g_loss))
    summarize_performance(i, model_G, model_D, dataset, latent_dim)
    #afficher les images
    plot_images(i+1, model_G)
```

2 Corps d'exécution

Définissons maintenant les constantes du programme

[11]:
```
NUM_EPOCHS = 150
BATCH_SIZE = 64
OPT_LR = 2e-4
LATENT_DIM = 100
```

[12]:
```
# Discriminateur
discriminateur = define_Discriminator()

# Générateur
generateur = define_Generator(LATENT_DIM)

# GAN
model_GAN = define_GAN(generateur, discriminateur)

# Chargeons nos images
dataset=load_real_images()
```

```
C:\Users\Thiba\anaconda3\envs\classenv\lib\site-
packages\tensorflow\python\keras\optimizer_v2\optimizer_v2.py:374: UserWarning:
The `lr` argument is deprecated, use `learning_rate` instead.
  warnings.warn(
```

```
Found 21244 images belonging to 1 classes.
```

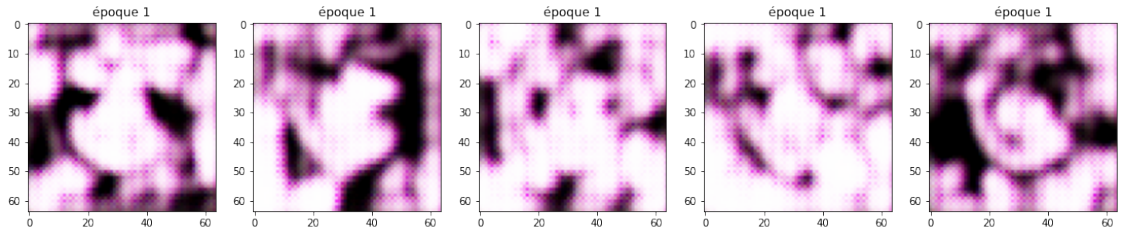
```
C:\Users\Thiba\AppData\Local\Temp\ipykernel_10700\2187398373.py:11:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
```

```
'dtype=object' when creating the ndarray.
  img_array = np.asarray(img_list)
```

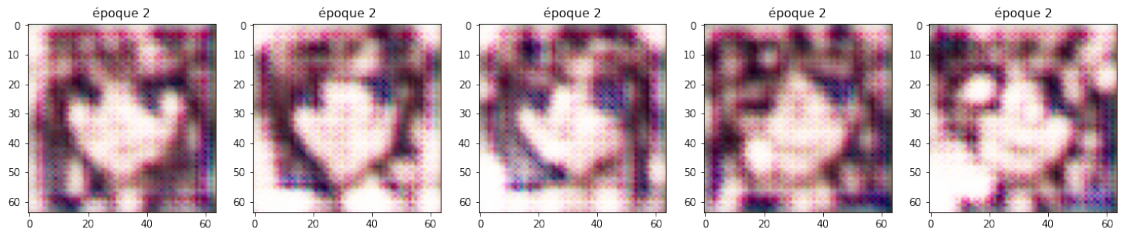
Entrainement:

[13]:
```
train_GAN(generateur, discriminateur, model_GAN, dataset[0], LATENT_DIM,␣
  ↪nb_epochs=NUM_EPOCHS, nb_batch=BATCH_SIZE)
```

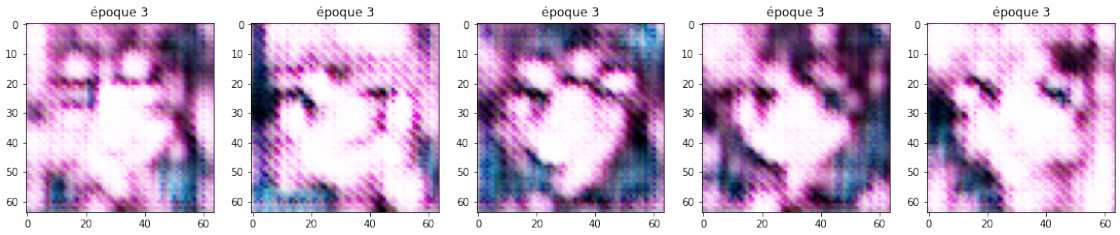
```
1, 1/128, d=0.650, g=0.693
1, 51/128, d=0.649, g=0.948
1, 101/128, d=0.614, g=0.783
Accuracy real: 26%, fake: 60%
(5, 64, 64, 3)
```



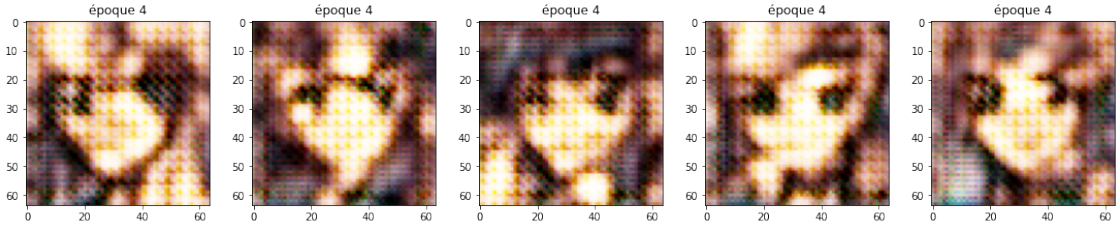

```
2, 1/128, d=0.691, g=0.894
2, 51/128, d=0.625, g=0.935
2, 101/128, d=0.794, g=0.623
Accuracy real: 62%, fake: 2%
(5, 64, 64, 3)
```



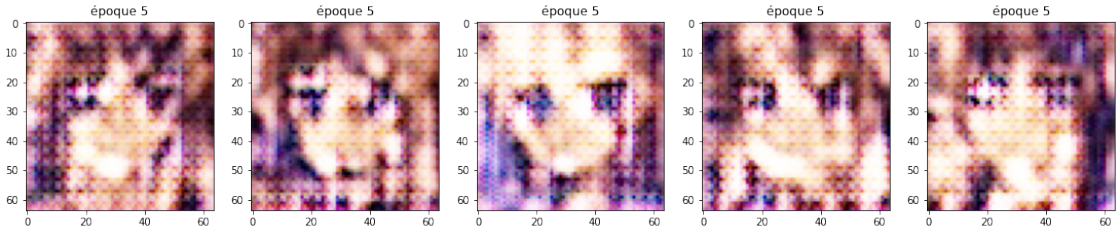

```
3, 1/128, d=0.757, g=0.634
3, 51/128, d=0.626, g=0.862
3, 101/128, d=0.687, g=0.729
Accuracy real: 19%, fake: 100%
(5, 64, 64, 3)
```



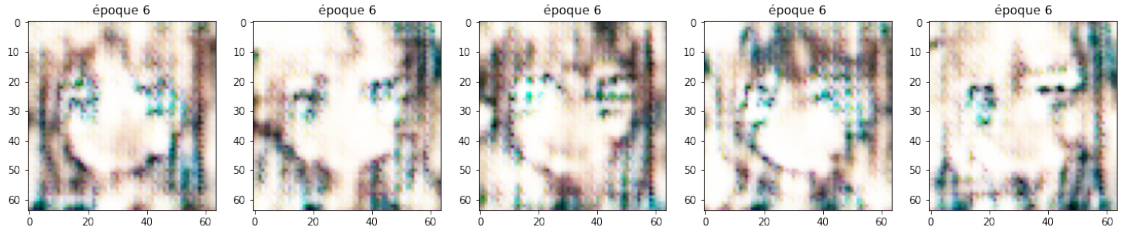

```
4, 1/128, d=0.567, g=0.894
4, 51/128, d=0.649, g=0.758
4, 101/128, d=0.634, g=0.979
Accuracy real: 88%, fake: 6%
(5, 64, 64, 3)
```



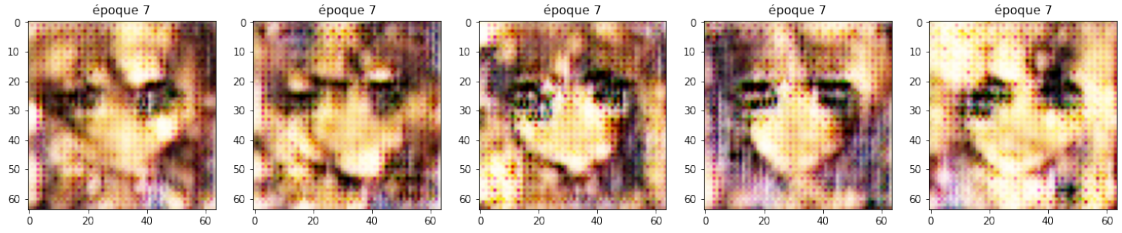

```
5, 1/128, d=0.746, g=0.681
5, 51/128, d=0.683, g=0.732
5, 101/128, d=0.652, g=0.801
Accuracy real: 92%, fake: 77%
(5, 64, 64, 3)
```



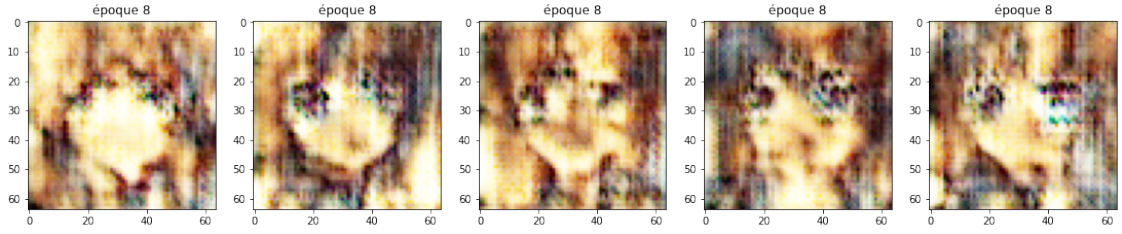

```
6, 1/128, d=0.662, g=0.769
6, 51/128, d=0.681, g=0.853
6, 101/128, d=0.704, g=0.697
Accuracy real: 4%, fake: 100%
(5, 64, 64, 3)
```



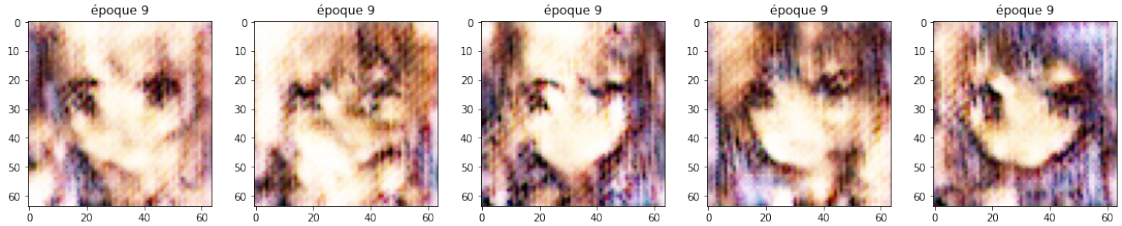

```
7, 1/128, d=0.579, g=0.906
7, 51/128, d=0.682, g=0.691
7, 101/128, d=0.675, g=0.909
Accuracy real: 40%, fake: 4%
(5, 64, 64, 3)
```



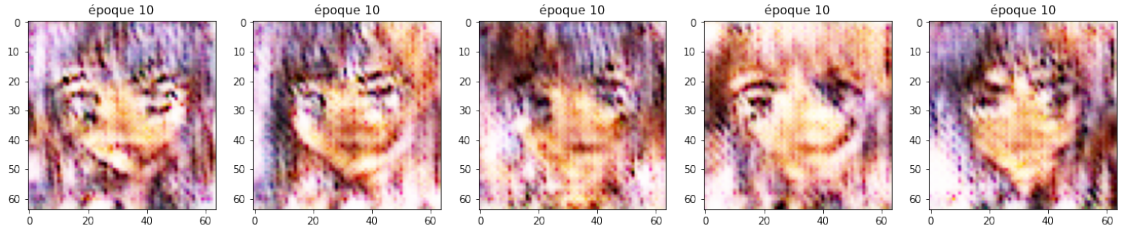

```
8, 1/128, d=0.814, g=0.538
8, 51/128, d=0.684, g=0.816
8, 101/128, d=0.670, g=0.741
Accuracy real: 77%, fake: 31%
(5, 64, 64, 3)
```



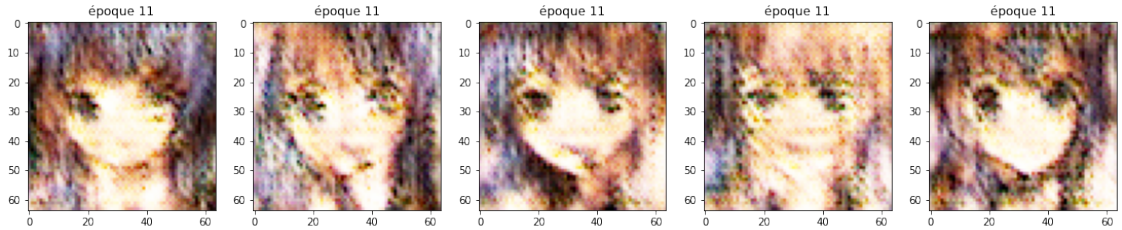

```
9, 1/128, d=0.698, g=0.697
9, 51/128, d=0.669, g=0.742
9, 101/128, d=0.589, g=0.818
Accuracy real: 92%, fake: 58%
(5, 64, 64, 3)
```



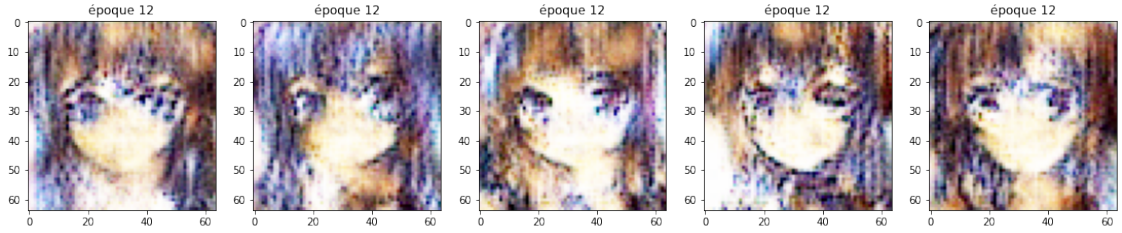

```
10, 1/128, d=0.670, g=0.749
10, 51/128, d=0.688, g=0.696
10, 101/128, d=0.667, g=0.704
Accuracy real: 63%, fake: 18%
(5, 64, 64, 3)
```



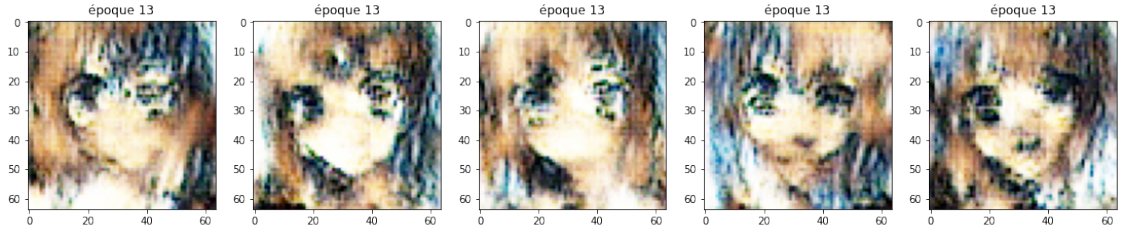

```
11, 1/128, d=0.709, g=0.666
11, 51/128, d=0.705, g=0.811
11, 101/128, d=0.701, g=0.846
Accuracy real: 90%, fake: 67%
(5, 64, 64, 3)
```



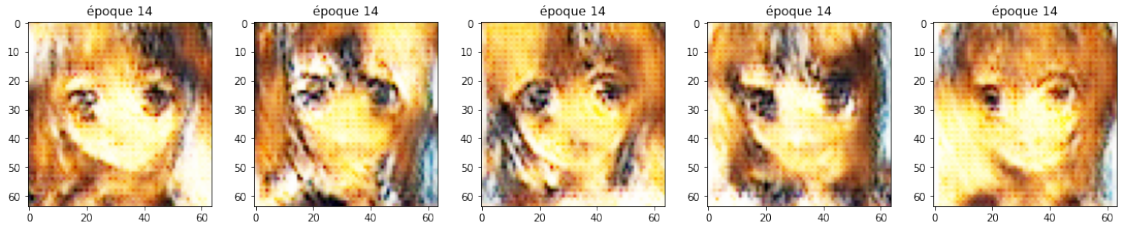

```
12, 1/128, d=0.654, g=0.785
12, 51/128, d=0.688, g=0.796
12, 101/128, d=0.668, g=0.777
Accuracy real: 53%, fake: 88%
(5, 64, 64, 3)
```



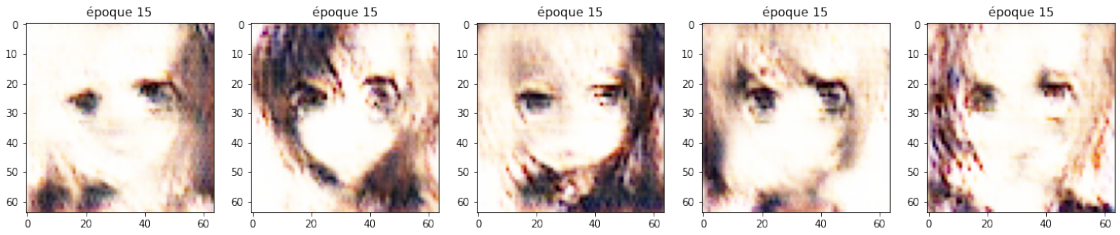

```
13, 1/128, d=0.664, g=0.750
13, 51/128, d=0.635, g=0.722
13, 101/128, d=0.708, g=0.886
Accuracy real: 92%, fake: 43%
(5, 64, 64, 3)
```



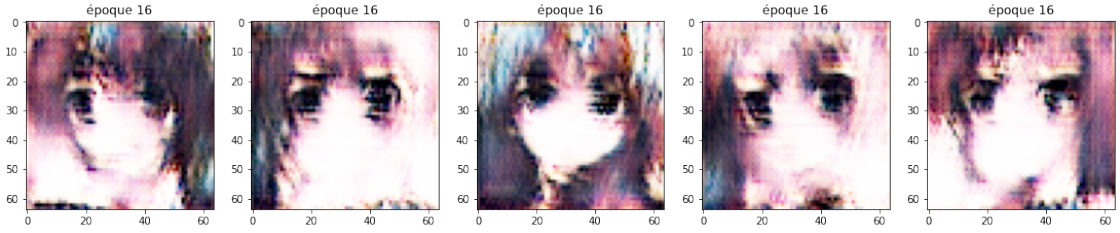

```
14, 1/128, d=0.684, g=0.793
14, 51/128, d=0.666, g=0.918
14, 101/128, d=0.685, g=0.659
Accuracy real: 66%, fake: 61%
(5, 64, 64, 3)
```



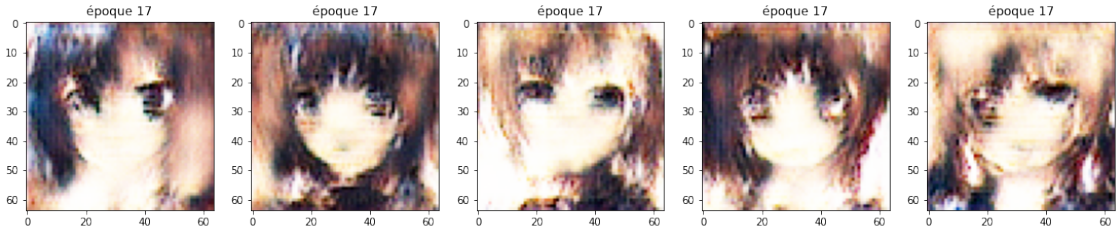

```
15, 1/128, d=0.683, g=0.676
15, 51/128, d=0.654, g=0.678
15, 101/128, d=0.737, g=0.929
Accuracy real: 62%, fake: 89%
(5, 64, 64, 3)
```



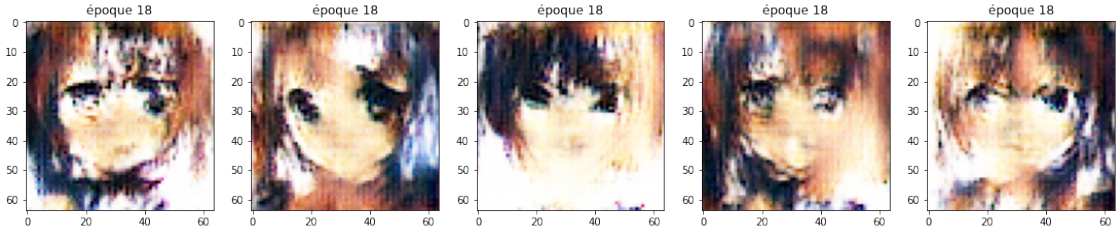

```
16, 1/128, d=0.656, g=0.839
16, 51/128, d=0.736, g=0.742
16, 101/128, d=0.690, g=1.123
Accuracy real: 51%, fake: 85%
(5, 64, 64, 3)
```



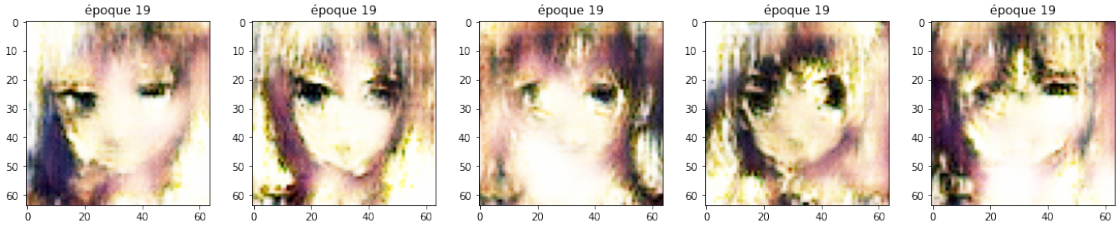

```
17, 1/128, d=0.641, g=0.754
17, 51/128, d=0.702, g=0.748
17, 101/128, d=0.756, g=0.734
Accuracy real: 68%, fake: 6%
(5, 64, 64, 3)
```



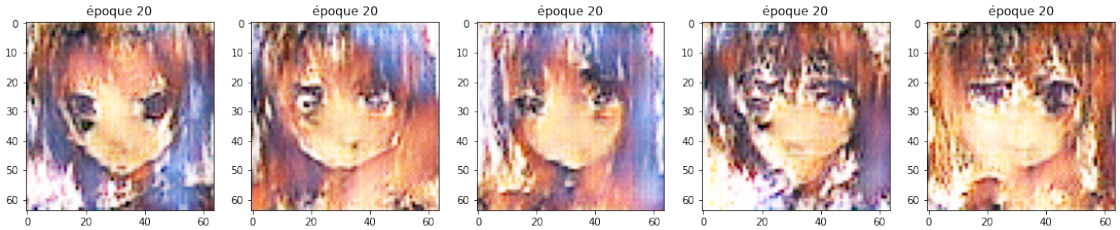

```
18, 1/128, d=0.741, g=0.744
18, 51/128, d=0.685, g=0.857
18, 101/128, d=0.670, g=0.840
Accuracy real: 63%, fake: 25%
(5, 64, 64, 3)
```



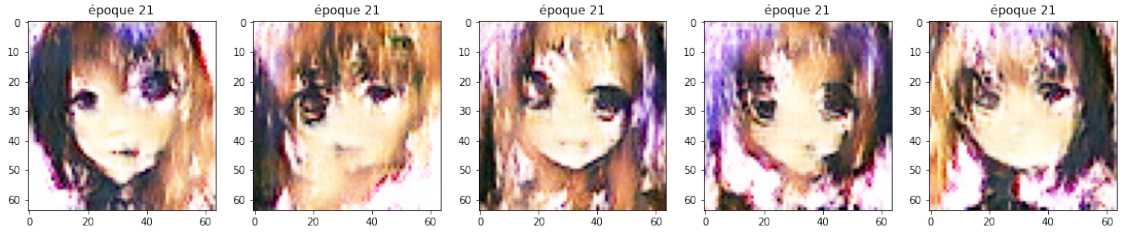

```
19, 1/128, d=0.730, g=0.745
19, 51/128, d=0.640, g=0.817
19, 101/128, d=0.663, g=0.735
Accuracy real: 38%, fake: 56%
(5, 64, 64, 3)
```



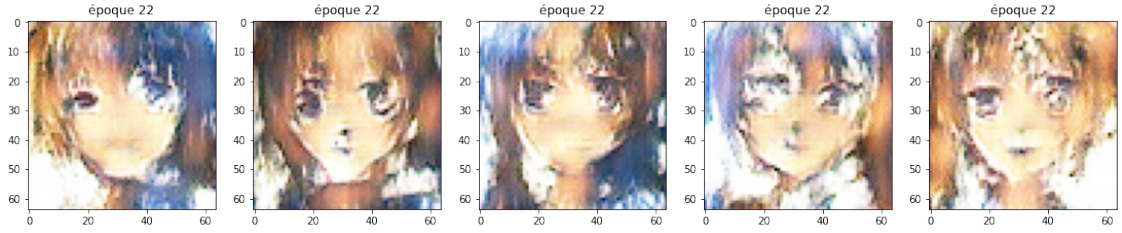

```
20, 1/128, d=0.689, g=0.695
20, 51/128, d=0.619, g=0.874
20, 101/128, d=0.659, g=0.908
Accuracy real: 20%, fake: 99%
(5, 64, 64, 3)
```



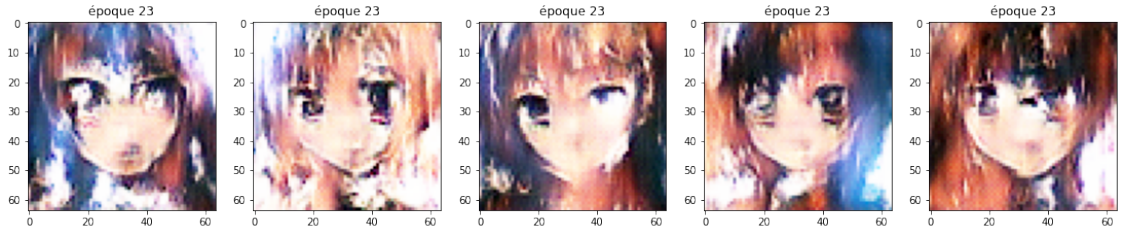

```
21, 1/128, d=0.630, g=0.711
21, 51/128, d=0.770, g=0.674
21, 101/128, d=0.666, g=0.989
Accuracy real: 73%, fake: 44%
(5, 64, 64, 3)
```



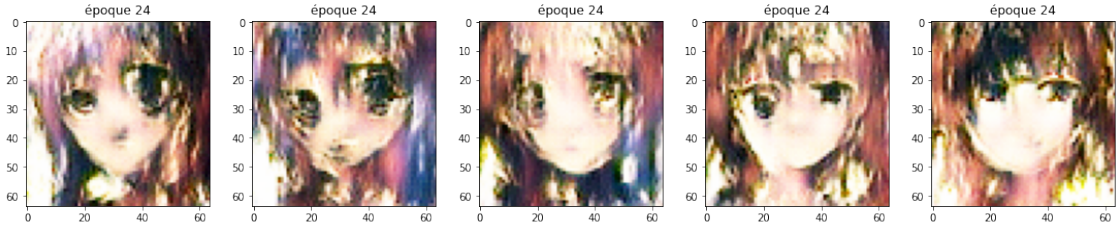

```
22, 1/128, d=0.692, g=0.637
22, 51/128, d=0.651, g=0.731
22, 101/128, d=0.684, g=0.654
Accuracy real: 66%, fake: 96%
(5, 64, 64, 3)
```



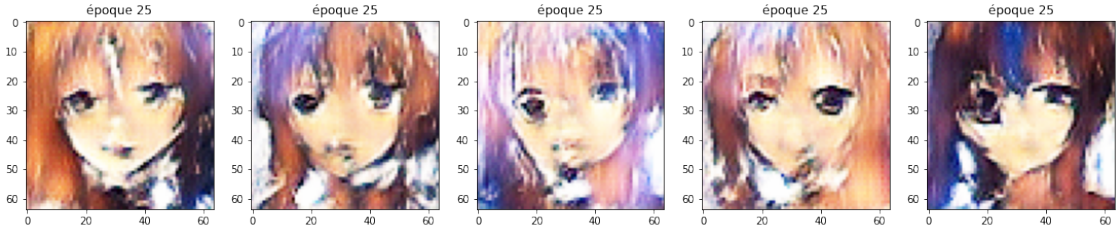

```
23, 1/128, d=0.638, g=0.708
23, 51/128, d=0.686, g=0.681
23, 101/128, d=0.652, g=0.861
Accuracy real: 89%, fake: 34%
(5, 64, 64, 3)
```



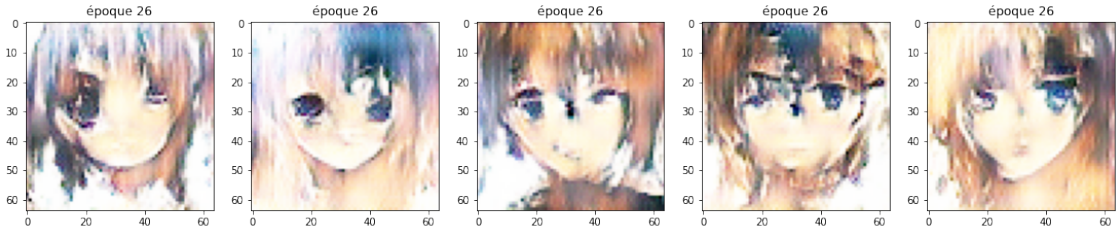

```
24, 1/128, d=0.695, g=0.839
24, 51/128, d=0.647, g=0.953
24, 101/128, d=0.682, g=0.756
Accuracy real: 71%, fake: 76%
(5, 64, 64, 3)
```



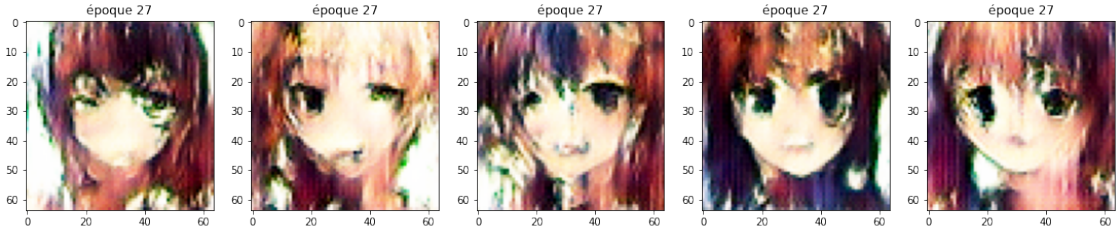

```
25, 1/128, d=0.653, g=0.791
25, 51/128, d=0.706, g=0.424
25, 101/128, d=0.667, g=0.723
Accuracy real: 88%, fake: 46%
(5, 64, 64, 3)
```



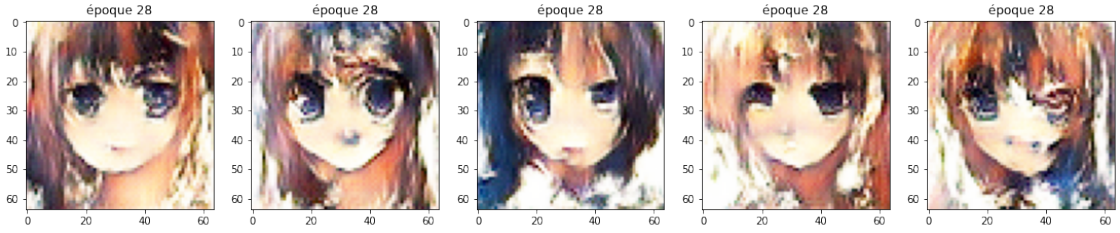

```
26, 1/128, d=0.675, g=0.914
26, 51/128, d=0.624, g=0.793
26, 101/128, d=0.670, g=0.699
Accuracy real: 96%, fake: 38%
(5, 64, 64, 3)
```



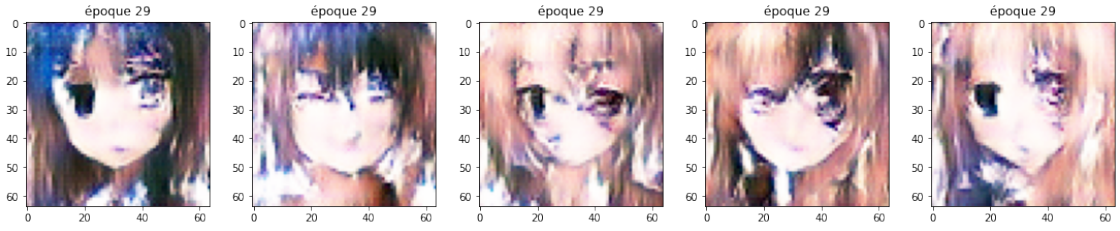

```
27, 1/128, d=0.687, g=0.876
27, 51/128, d=0.604, g=0.899
27, 101/128, d=0.625, g=0.794
Accuracy real: 29%, fake: 97%
(5, 64, 64, 3)
```



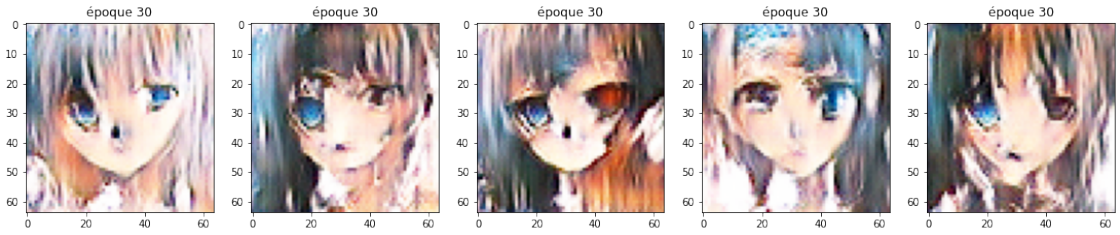

```
28, 1/128, d=0.606, g=0.909
28, 51/128, d=0.675, g=0.731
28, 101/128, d=0.669, g=0.735
Accuracy real: 17%, fake: 99%
(5, 64, 64, 3)
```



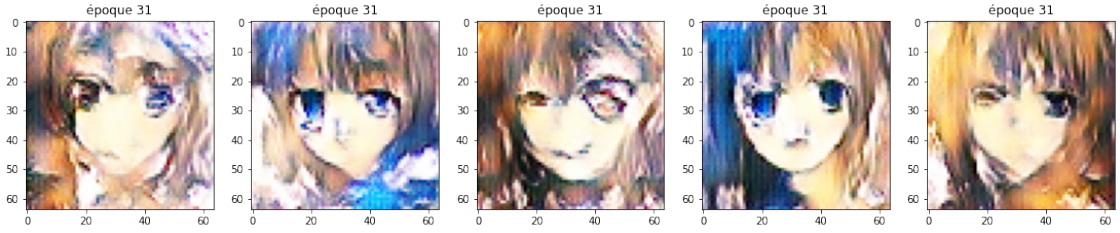

```
29, 1/128, d=0.565, g=0.674
29, 51/128, d=0.664, g=0.786
29, 101/128, d=0.593, g=0.767
Accuracy real: 47%, fake: 88%
(5, 64, 64, 3)
```



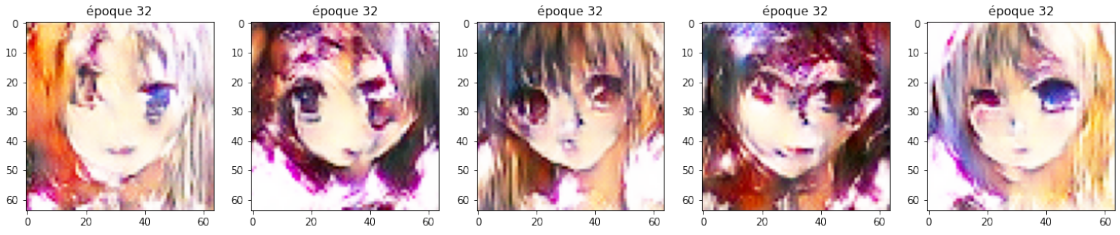

```
30, 1/128, d=0.644, g=0.775
30, 51/128, d=0.636, g=0.720
30, 101/128, d=0.693, g=0.541
Accuracy real: 95%, fake: 8%
(5, 64, 64, 3)
```



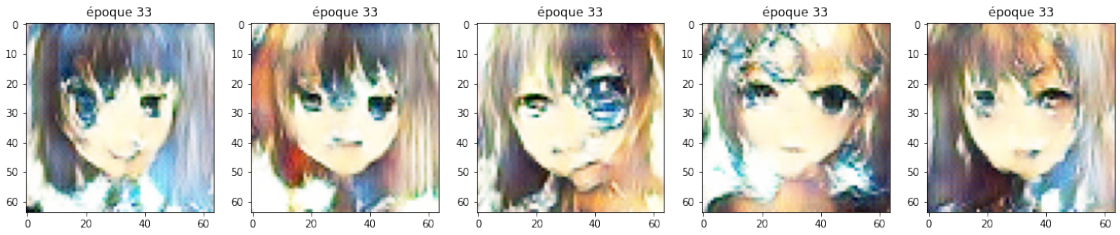

```
31, 1/128, d=0.773, g=0.978
31, 51/128, d=0.686, g=0.813
31, 101/128, d=0.681, g=0.793
Accuracy real: 15%, fake: 99%
(5, 64, 64, 3)
```



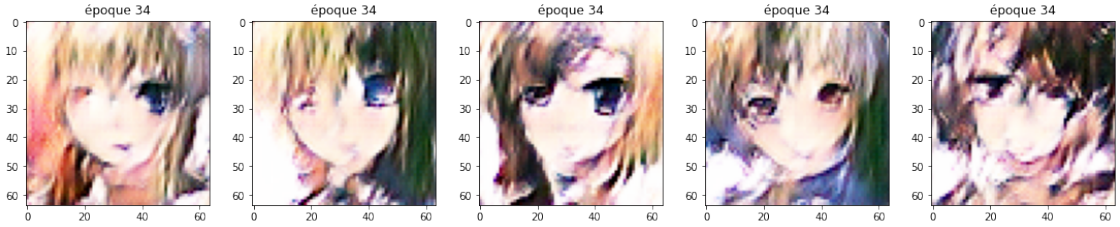

```
32, 1/128, d=0.559, g=0.640
32, 51/128, d=0.681, g=0.725
32, 101/128, d=0.660, g=0.928
Accuracy real: 37%, fake: 98%
(5, 64, 64, 3)
```



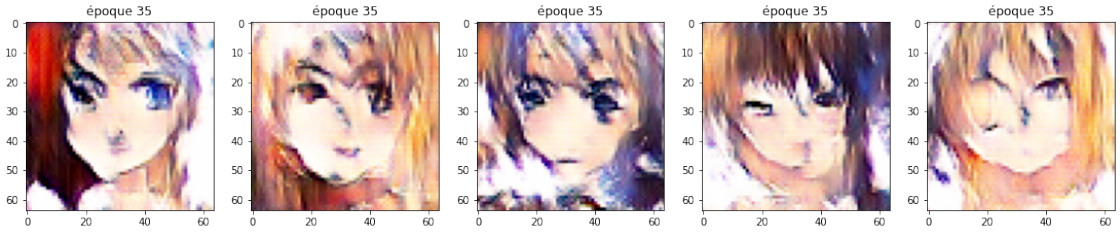

```
33, 1/128, d=0.614, g=0.799
33, 51/128, d=0.653, g=1.067
33, 101/128, d=0.676, g=0.740
Accuracy real: 100%, fake: 0%
(5, 64, 64, 3)
```



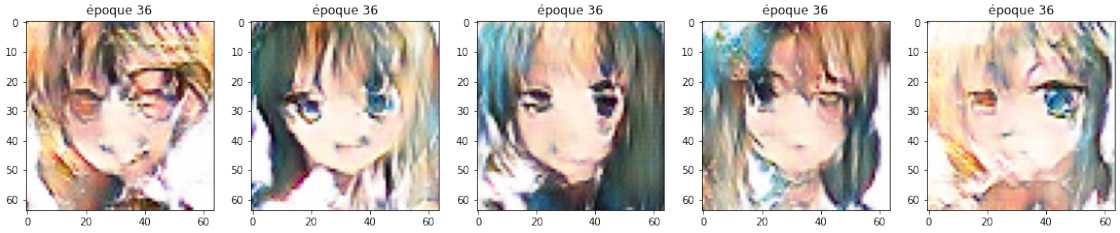

```
34, 1/128, d=0.994, g=1.412
34, 51/128, d=0.664, g=1.342
34, 101/128, d=0.642, g=0.864
Accuracy real: 96%, fake: 0%
(5, 64, 64, 3)
```



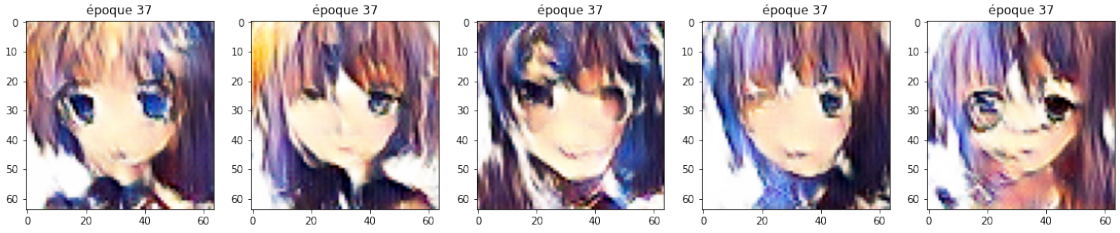

```
35, 1/128, d=0.847, g=1.149
35, 51/128, d=0.666, g=0.679
35, 101/128, d=0.747, g=1.769
Accuracy real: 41%, fake: 74%
(5, 64, 64, 3)
```



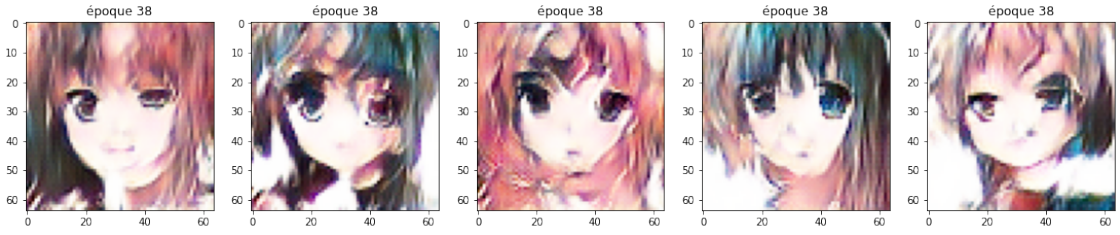

```
36, 1/128, d=0.661, g=0.842
36, 51/128, d=0.636, g=0.916
36, 101/128, d=0.675, g=1.070
Accuracy real: 40%, fake: 94%
(5, 64, 64, 3)
```



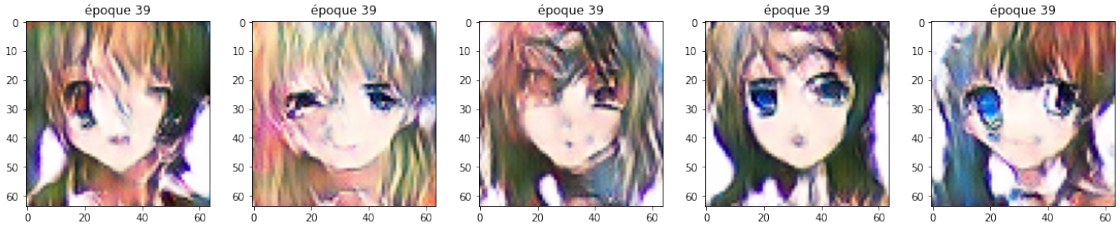

```
37, 1/128, d=0.614, g=0.641
37, 51/128, d=0.726, g=0.705
37, 101/128, d=0.645, g=0.830
Accuracy real: 41%, fake: 68%
(5, 64, 64, 3)
```



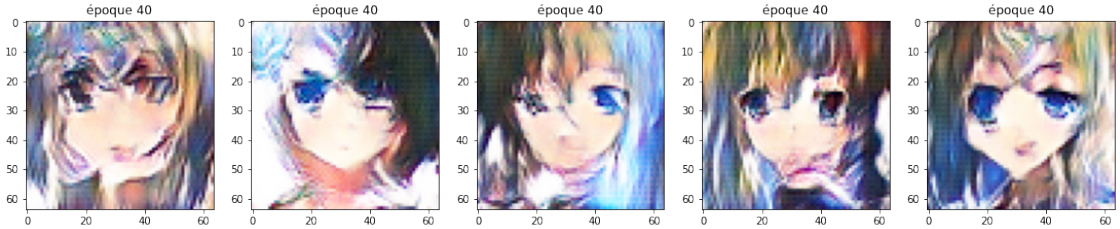

```
38, 1/128, d=0.661, g=0.959
38, 51/128, d=0.728, g=0.868
38, 101/128, d=0.643, g=0.571
Accuracy real: 87%, fake: 12%
(5, 64, 64, 3)
```



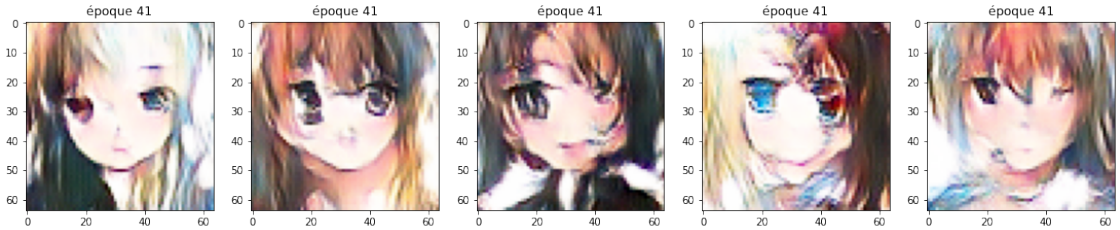

```
39, 1/128, d=0.798, g=0.966
39, 51/128, d=0.660, g=0.986
39, 101/128, d=0.678, g=0.764
Accuracy real: 52%, fake: 92%
(5, 64, 64, 3)
```



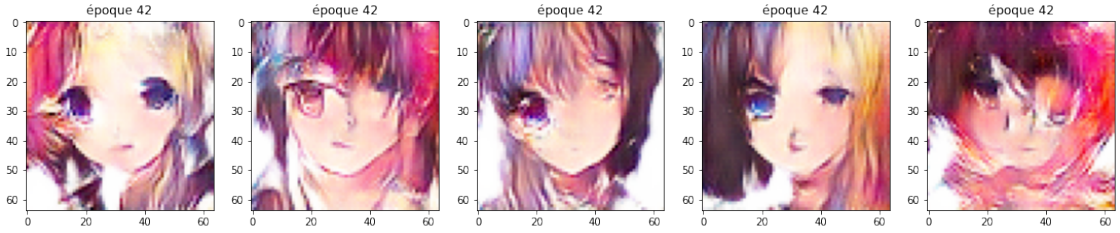

```
40, 1/128, d=0.599, g=0.787
40, 51/128, d=0.762, g=0.434
40, 101/128, d=0.624, g=0.994
Accuracy real: 43%, fake: 91%
(5, 64, 64, 3)
```



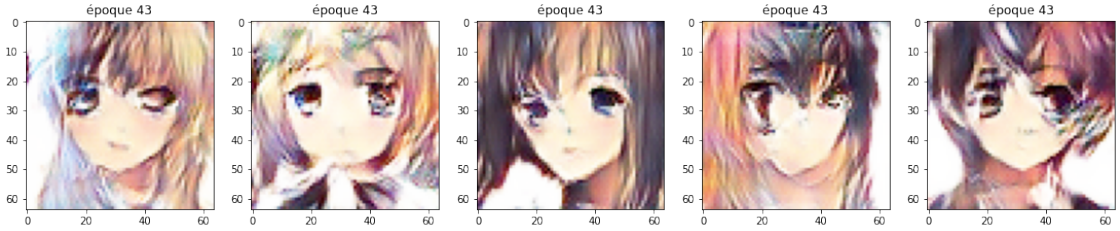

```
41, 1/128, d=0.633, g=0.777
41, 51/128, d=0.617, g=0.768
41, 101/128, d=0.641, g=0.708
Accuracy real: 16%, fake: 100%
(5, 64, 64, 3)
```



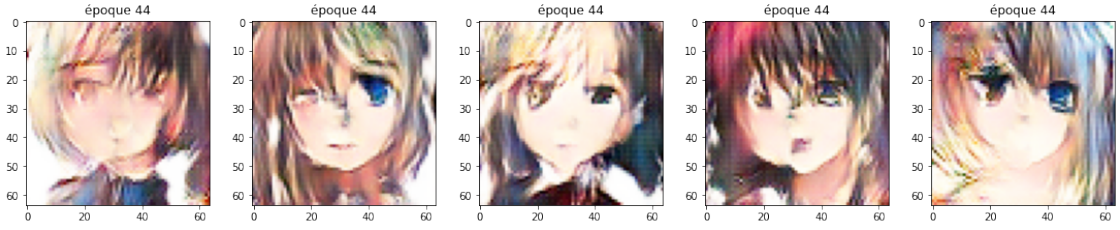

```
42, 1/128, d=0.561, g=0.652
42, 51/128, d=0.672, g=0.771
42, 101/128, d=0.664, g=0.760
Accuracy real: 0%, fake: 100%
(5, 64, 64, 3)
```



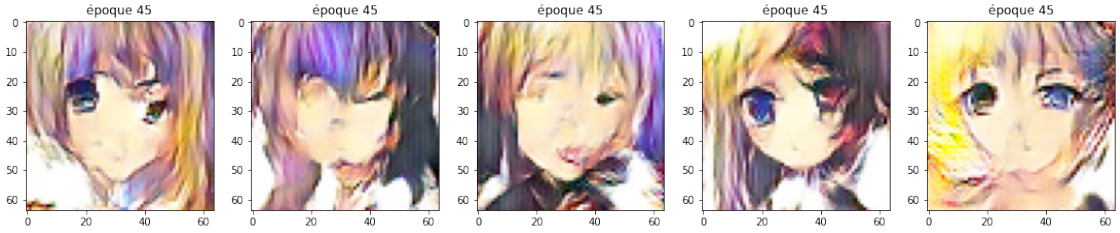

```
43, 1/128, d=0.527, g=0.631
43, 51/128, d=0.719, g=0.729
43, 101/128, d=0.664, g=0.758
Accuracy real: 28%, fake: 98%
(5, 64, 64, 3)
```



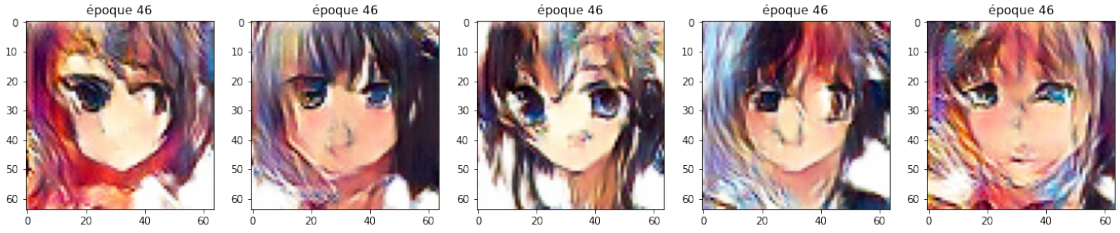

```
44, 1/128, d=0.604, g=0.733
44, 51/128, d=0.723, g=0.417
44, 101/128, d=0.707, g=1.307
Accuracy real: 58%, fake: 82%
(5, 64, 64, 3)
```



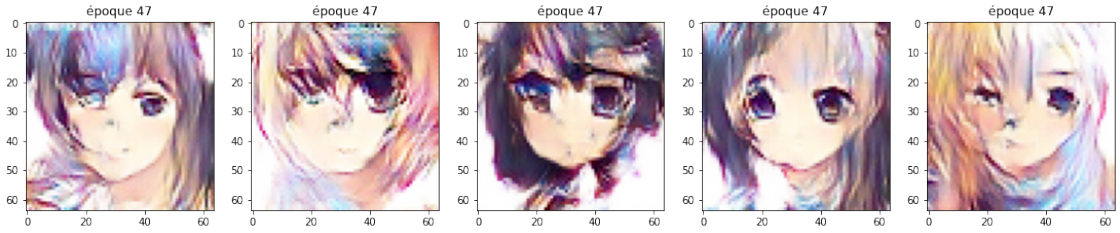

```
45, 1/128, d=0.631, g=0.691
45, 51/128, d=0.642, g=0.781
45, 101/128, d=0.649, g=0.719
Accuracy real: 81%, fake: 54%
(5, 64, 64, 3)
```



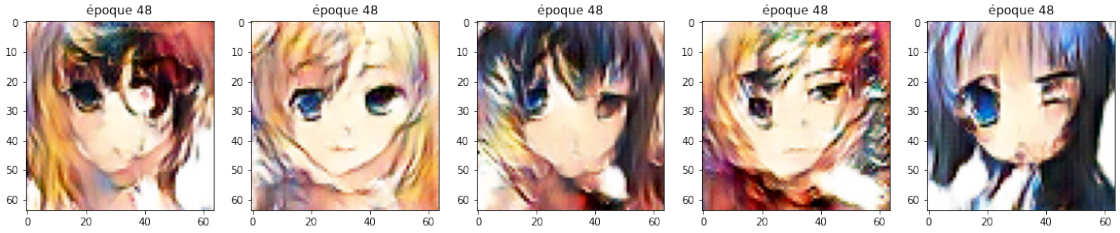

```
46, 1/128, d=0.684, g=0.839
46, 51/128, d=0.655, g=0.639
46, 101/128, d=0.662, g=0.690
Accuracy real: 93%, fake: 30%
(5, 64, 64, 3)
```



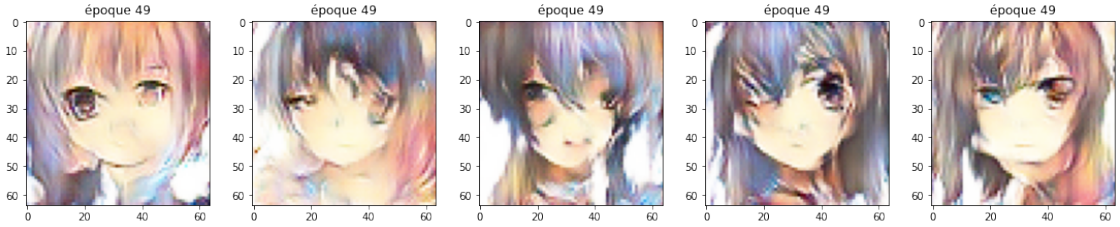

```
47, 1/128, d=0.711, g=0.802
47, 51/128, d=0.688, g=0.691
47, 101/128, d=0.630, g=0.774
Accuracy real: 21%, fake: 100%
(5, 64, 64, 3)
```



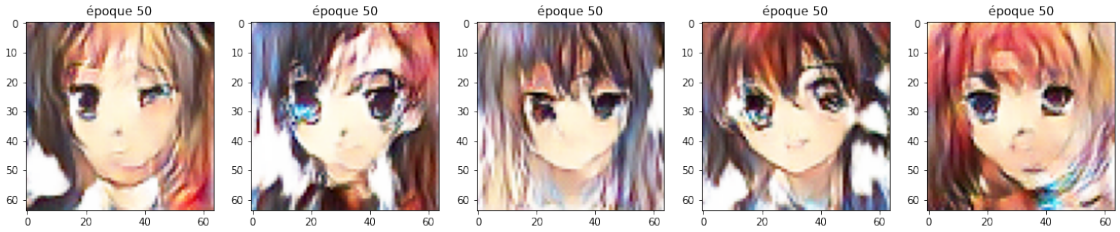

```
48, 1/128, d=0.550, g=0.684
48, 51/128, d=0.642, g=0.759
48, 101/128, d=0.705, g=0.689
Accuracy real: 9%, fake: 99%
(5, 64, 64, 3)
```



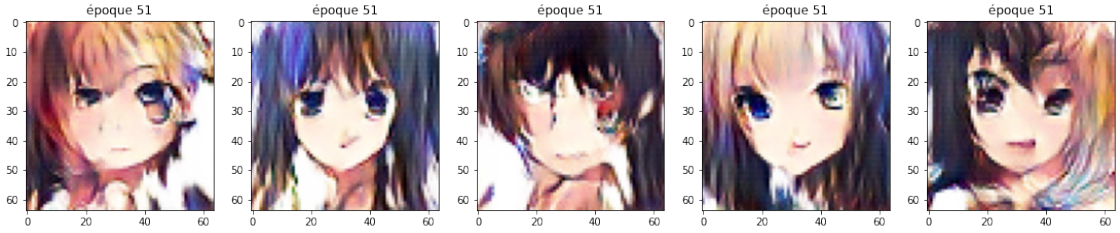

```
49, 1/128, d=0.590, g=0.746
49, 51/128, d=0.673, g=0.730
49, 101/128, d=0.678, g=0.665
Accuracy real: 27%, fake: 100%
(5, 64, 64, 3)
```



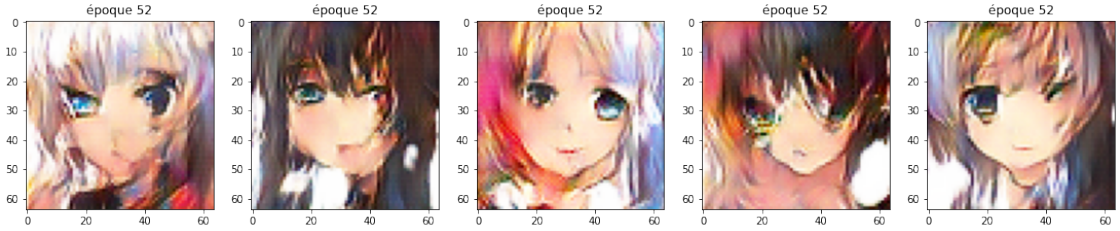

```
50, 1/128, d=0.586, g=0.820
50, 51/128, d=0.765, g=0.650
50, 101/128, d=0.686, g=0.791
Accuracy real: 50%, fake: 82%
(5, 64, 64, 3)
```



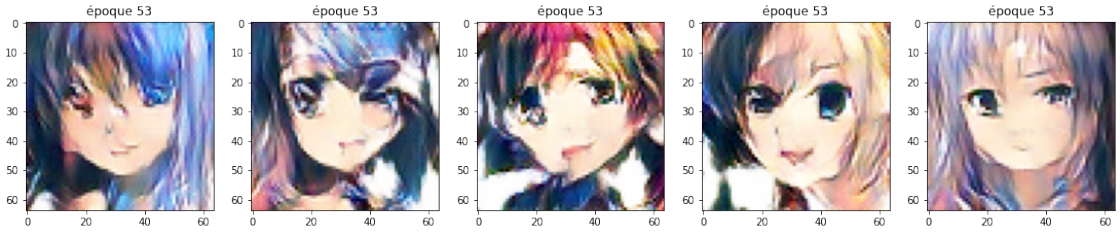

```
51, 1/128, d=0.658, g=0.791
51, 51/128, d=0.669, g=0.724
51, 101/128, d=0.662, g=0.747
Accuracy real: 81%, fake: 20%
(5, 64, 64, 3)
```



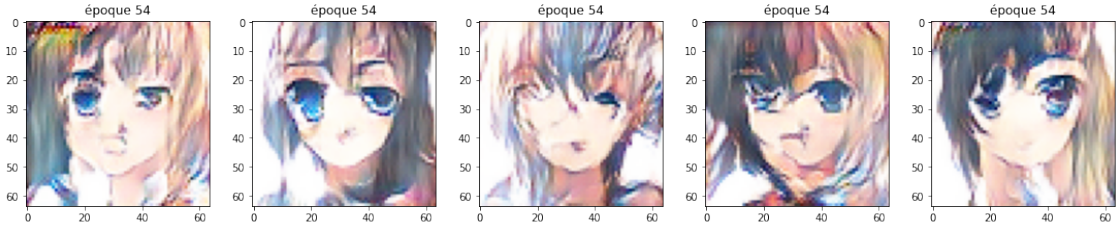

```
52, 1/128, d=0.724, g=0.833
52, 51/128, d=0.650, g=0.863
52, 101/128, d=0.682, g=0.781
Accuracy real: 59%, fake: 81%
(5, 64, 64, 3)
```



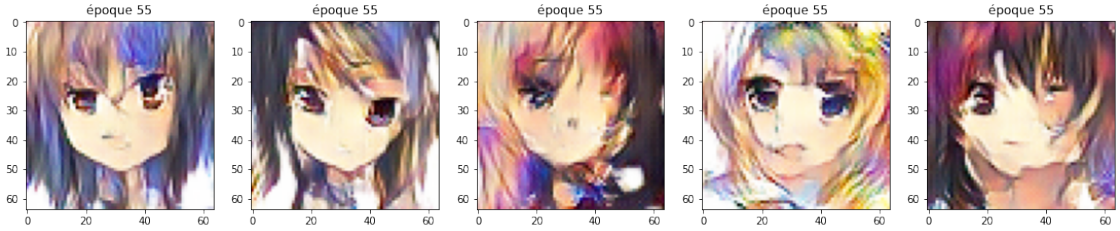

```
53, 1/128, d=0.651, g=0.660
53, 51/128, d=0.645, g=0.759
53, 101/128, d=0.655, g=0.875
Accuracy real: 82%, fake: 12%
(5, 64, 64, 3)
```



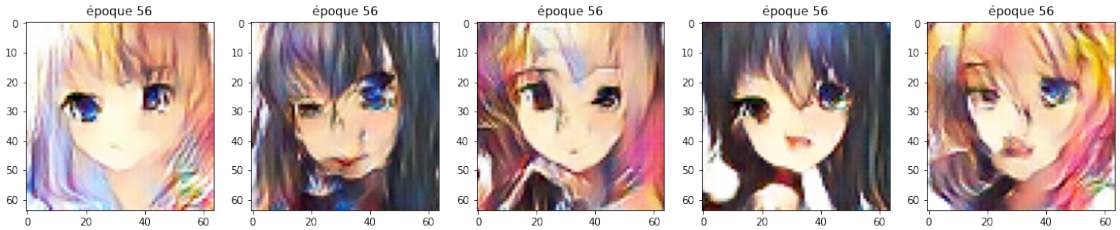

```
54, 1/128, d=0.720, g=0.854
54, 51/128, d=0.672, g=0.713
54, 101/128, d=0.704, g=0.635
Accuracy real: 94%, fake: 12%
(5, 64, 64, 3)
```



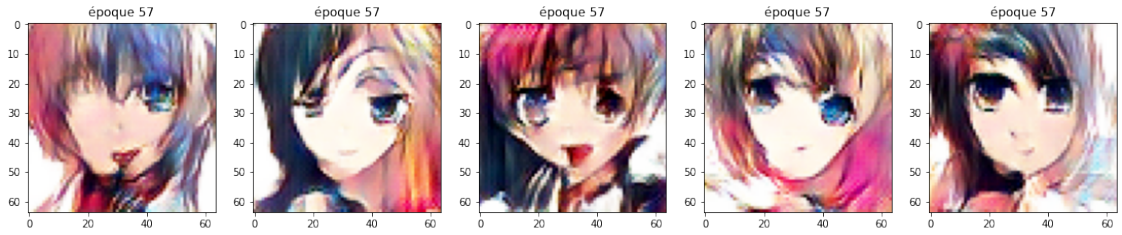

```
55, 1/128, d=0.717, g=0.754
55, 51/128, d=0.673, g=0.693
55, 101/128, d=0.697, g=0.662
Accuracy real: 58%, fake: 75%
(5, 64, 64, 3)
```



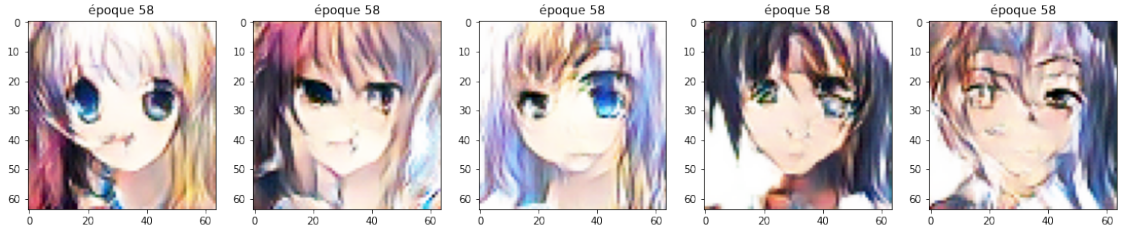

```
56, 1/128, d=0.658, g=0.700
56, 51/128, d=0.656, g=0.732
56, 101/128, d=0.658, g=0.847
Accuracy real: 72%, fake: 71%
(5, 64, 64, 3)
```



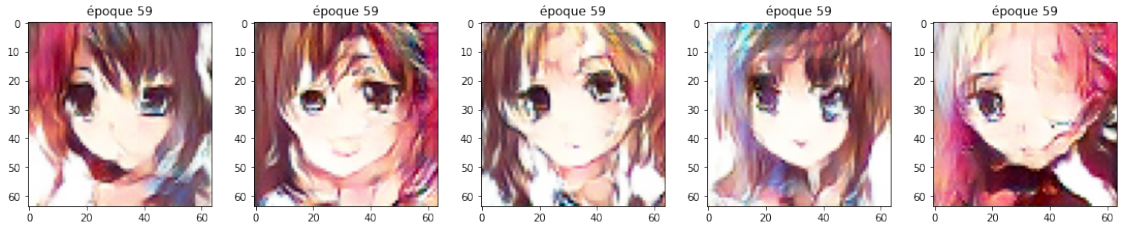

```
57, 1/128, d=0.656, g=0.811
57, 51/128, d=0.678, g=0.920
57, 101/128, d=0.714, g=0.847
Accuracy real: 57%, fake: 61%
(5, 64, 64, 3)
```



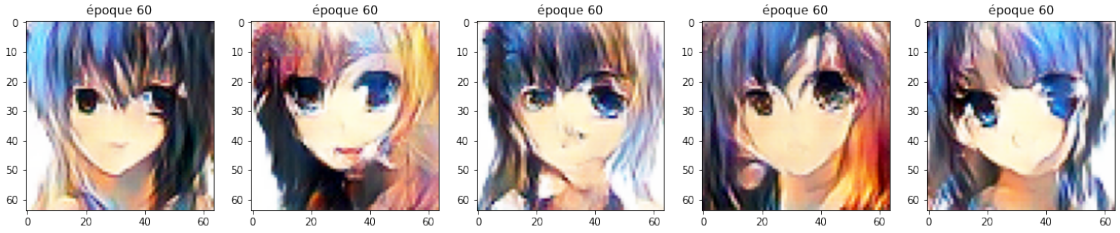

```
58, 1/128, d=0.673, g=0.763
58, 51/128, d=0.659, g=0.641
58, 101/128, d=0.655, g=0.896
Accuracy real: 6%, fake: 100%
(5, 64, 64, 3)
```



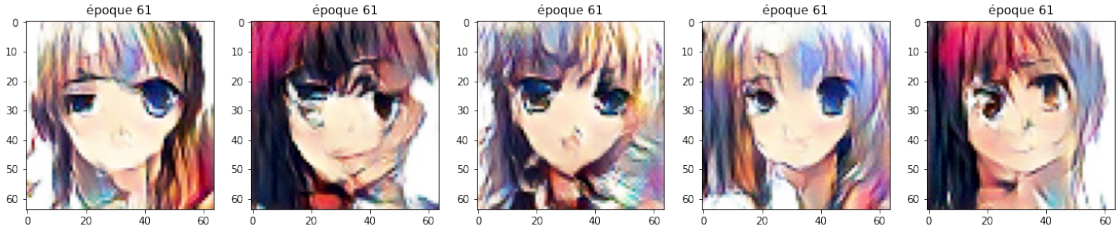

```
59, 1/128, d=0.571, g=0.654
59, 51/128, d=0.682, g=1.087
59, 101/128, d=0.663, g=0.718
Accuracy real: 61%, fake: 53%
(5, 64, 64, 3)
```



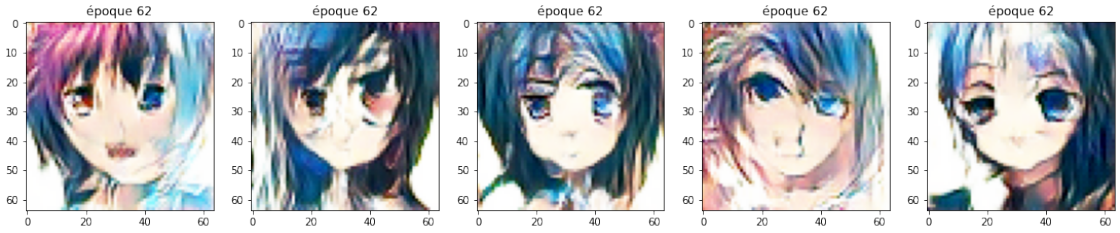

```
60, 1/128, d=0.689, g=0.715
60, 51/128, d=0.686, g=0.845
60, 101/128, d=0.708, g=0.786
Accuracy real: 46%, fake: 48%
(5, 64, 64, 3)
```



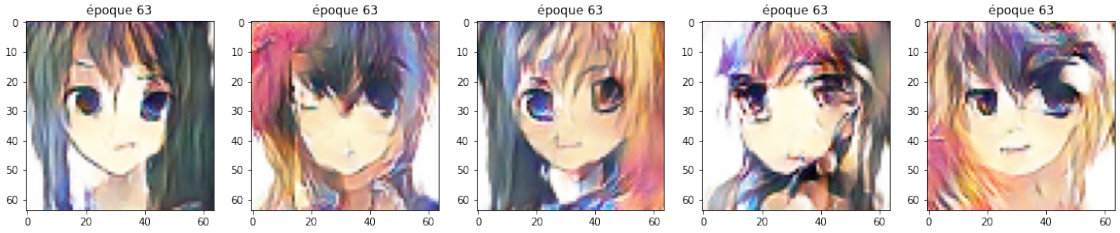

```
61, 1/128, d=0.700, g=0.686
61, 51/128, d=0.666, g=0.764
61, 101/128, d=0.684, g=0.793
Accuracy real: 71%, fake: 50%
(5, 64, 64, 3)
```



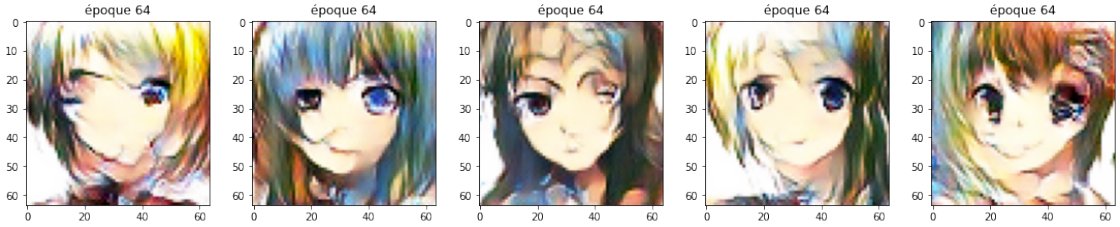

```
62, 1/128, d=0.679, g=0.784
62, 51/128, d=0.695, g=0.710
62, 101/128, d=0.680, g=0.792
Accuracy real: 20%, fake: 91%
(5, 64, 64, 3)
```



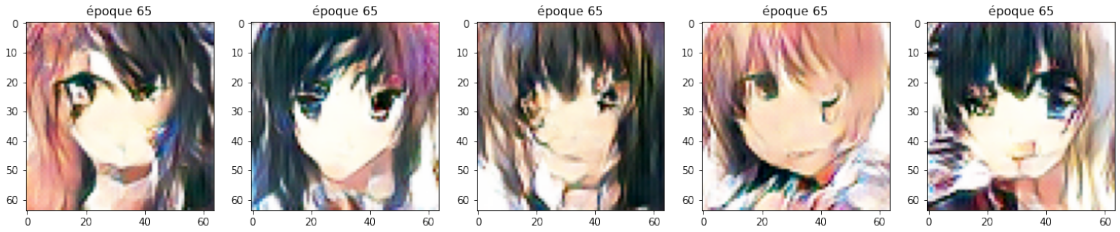

```
63, 1/128, d=0.630, g=0.812
63, 51/128, d=0.686, g=0.586
63, 101/128, d=0.671, g=0.779
Accuracy real: 49%, fake: 66%
(5, 64, 64, 3)
```



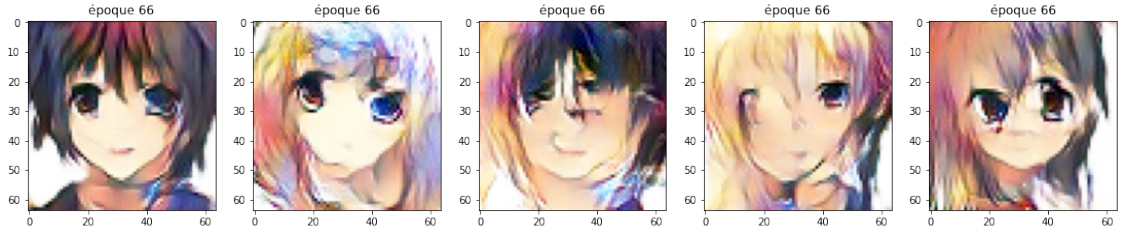

```
64, 1/128, d=0.681, g=0.717
64, 51/128, d=0.715, g=0.804
64, 101/128, d=0.665, g=0.696
Accuracy real: 40%, fake: 73%
(5, 64, 64, 3)
```



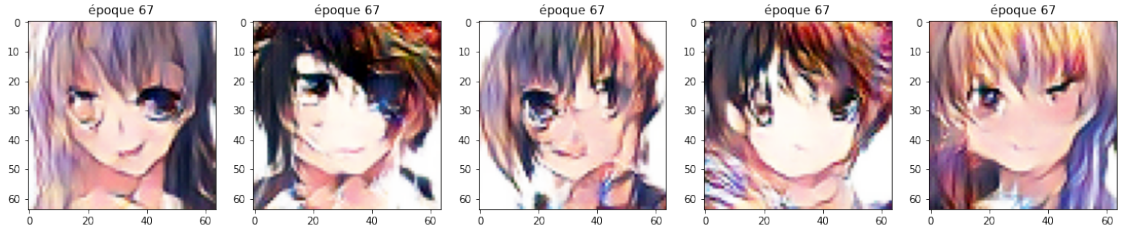

```
65, 1/128, d=0.670, g=0.718
65, 51/128, d=0.660, g=0.753
65, 101/128, d=0.660, g=0.758
Accuracy real: 44%, fake: 58%
(5, 64, 64, 3)
```



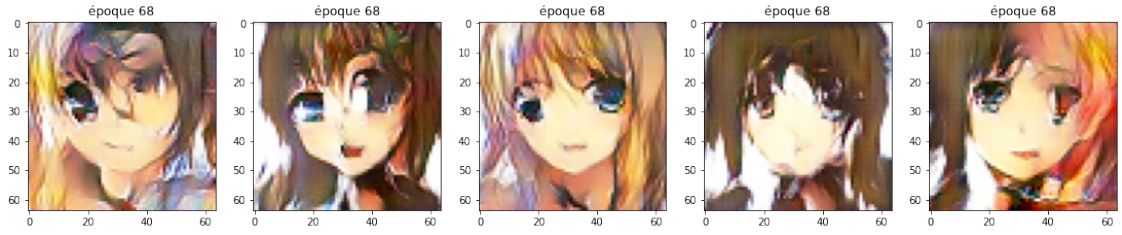

```
66, 1/128, d=0.686, g=0.717
66, 51/128, d=0.677, g=0.681
66, 101/128, d=0.687, g=0.879
Accuracy real: 91%, fake: 10%
(5, 64, 64, 3)
```



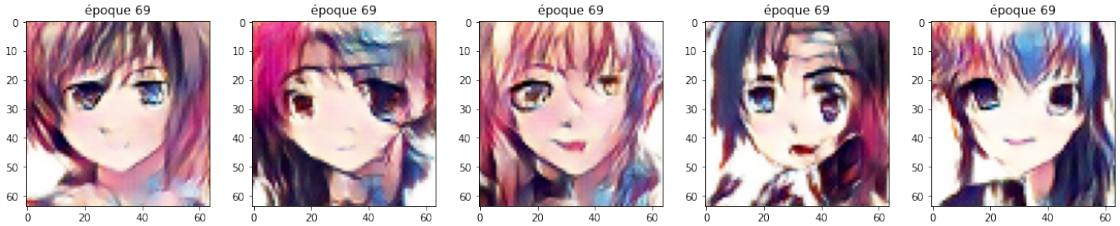

```
67, 1/128, d=0.756, g=0.915
67, 51/128, d=0.642, g=0.793
67, 101/128, d=0.683, g=0.658
Accuracy real: 92%, fake: 4%
(5, 64, 64, 3)
```



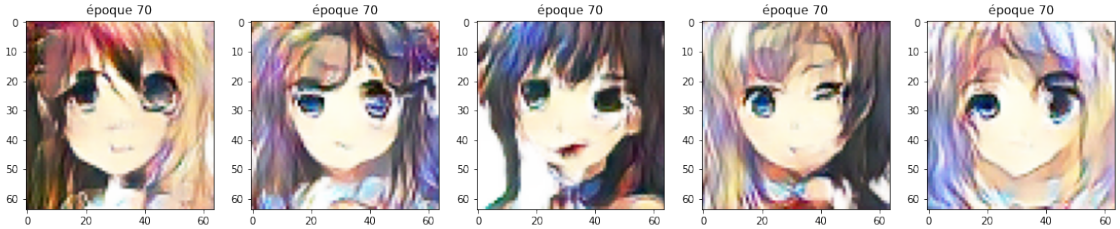

```
68, 1/128, d=0.757, g=0.908
68, 51/128, d=0.651, g=0.763
68, 101/128, d=0.669, g=0.746
Accuracy real: 25%, fake: 89%
(5, 64, 64, 3)
```



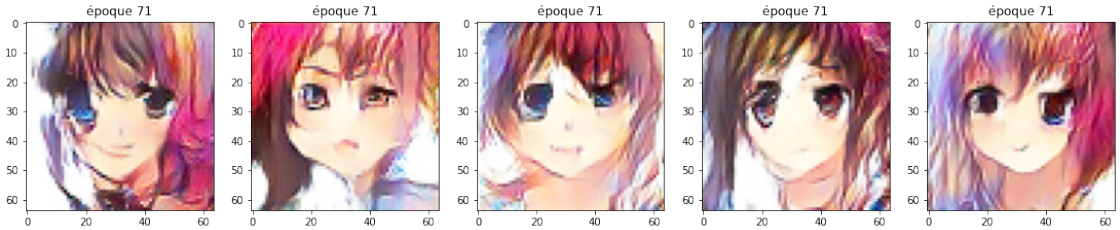

```
69, 1/128, d=0.640, g=0.640
69, 51/128, d=0.675, g=0.790
69, 101/128, d=0.718, g=0.734
Accuracy real: 71%, fake: 16%
(5, 64, 64, 3)
```



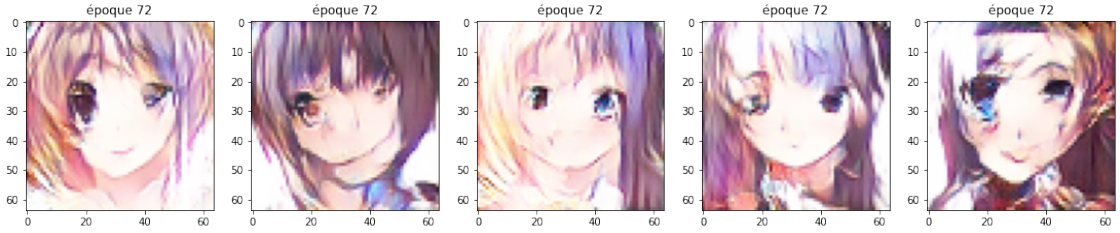

```
70, 1/128, d=0.749, g=0.782
70, 51/128, d=0.675, g=0.752
70, 101/128, d=0.675, g=0.738
Accuracy real: 59%, fake: 46%
(5, 64, 64, 3)
```



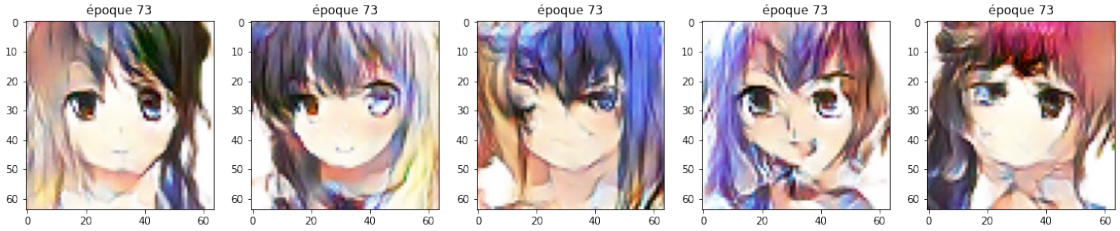

```
71, 1/128, d=0.681, g=0.845
71, 51/128, d=0.660, g=0.693
71, 101/128, d=0.697, g=0.809
Accuracy real: 46%, fake: 69%
(5, 64, 64, 3)
```



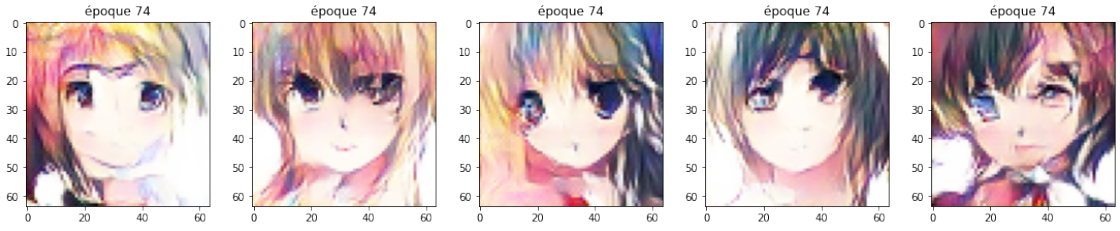

```
72, 1/128, d=0.685, g=0.800
72, 51/128, d=0.681, g=0.650
72, 101/128, d=0.710, g=0.683
Accuracy real: 78%, fake: 69%
(5, 64, 64, 3)
```



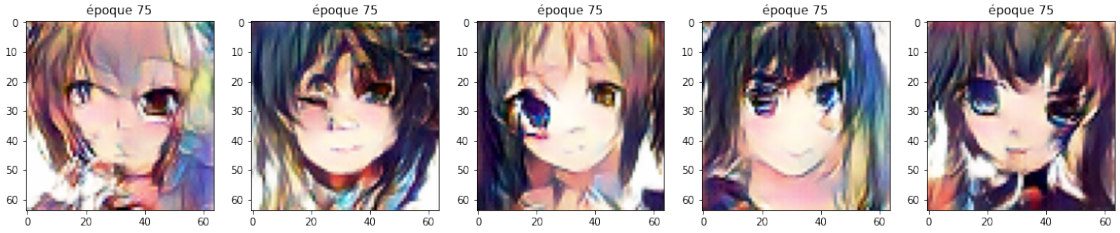

```
73, 1/128, d=0.668, g=0.887
73, 51/128, d=0.692, g=0.804
73, 101/128, d=0.689, g=0.663
Accuracy real: 47%, fake: 90%
(5, 64, 64, 3)
```



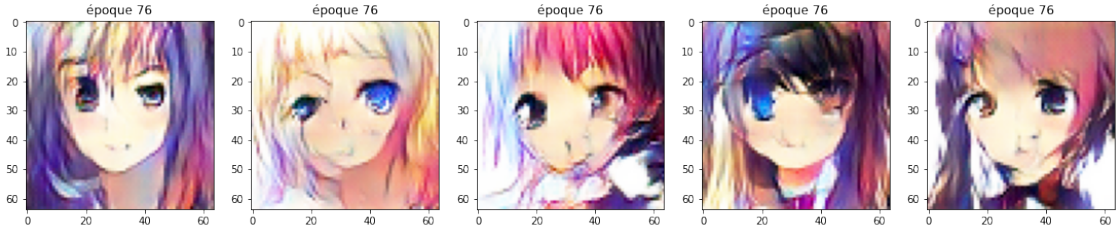

```
74, 1/128, d=0.650, g=0.835
74, 51/128, d=0.672, g=0.692
74, 101/128, d=0.692, g=0.728
Accuracy real: 59%, fake: 62%
(5, 64, 64, 3)
```



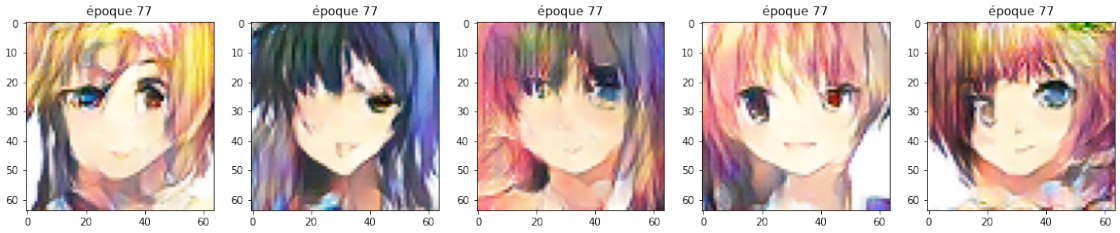

```
75, 1/128, d=0.682, g=0.607
75, 51/128, d=0.728, g=0.822
75, 101/128, d=0.682, g=0.813
Accuracy real: 60%, fake: 50%
(5, 64, 64, 3)
```



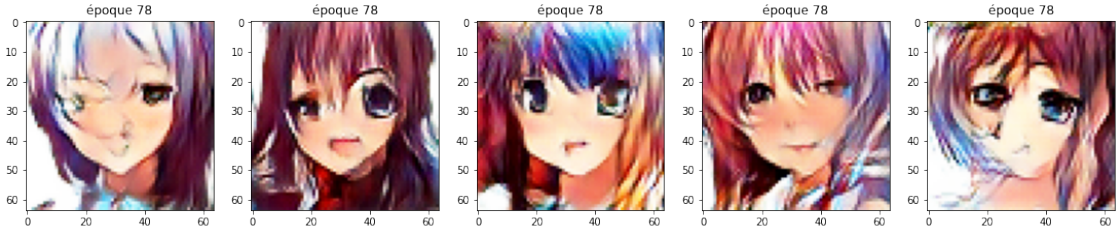

```
76, 1/128, d=0.696, g=0.763
76, 51/128, d=0.692, g=0.755
76, 101/128, d=0.670, g=0.720
Accuracy real: 36%, fake: 94%
(5, 64, 64, 3)
```



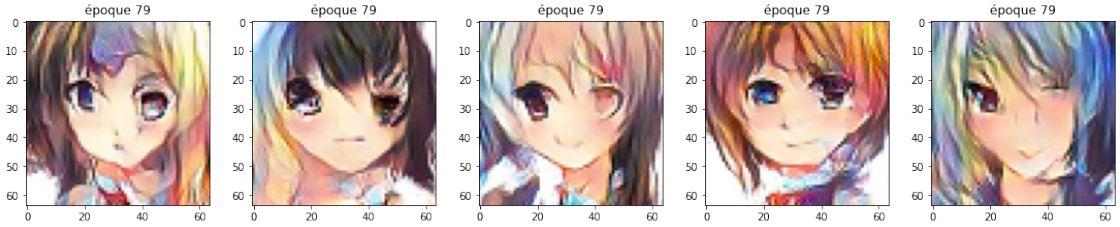

```
77, 1/128, d=0.638, g=0.744
77, 51/128, d=0.654, g=0.670
77, 101/128, d=0.715, g=0.700
Accuracy real: 83%, fake: 37%
(5, 64, 64, 3)
```



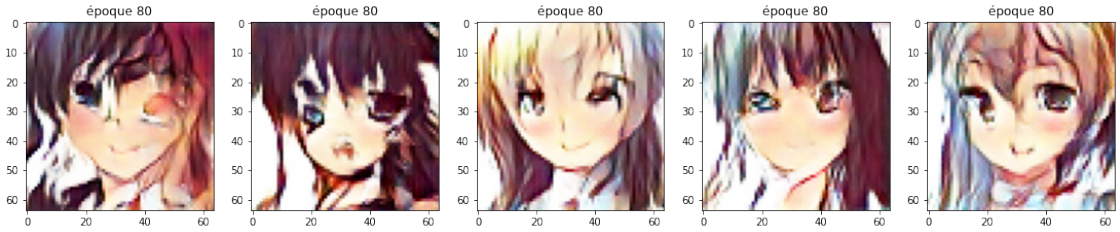

```
78, 1/128, d=0.694, g=0.663
78, 51/128, d=0.664, g=0.682
78, 101/128, d=0.686, g=0.799
Accuracy real: 47%, fake: 39%
(5, 64, 64, 3)
```



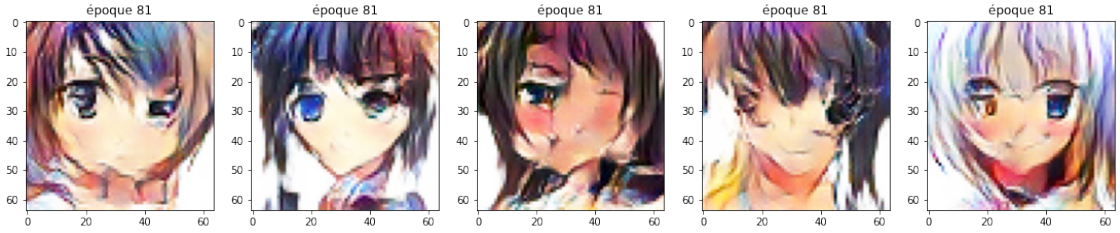

```
79, 1/128, d=0.708, g=0.711
79, 51/128, d=0.694, g=0.621
79, 101/128, d=0.678, g=0.714
Accuracy real: 78%, fake: 12%
(5, 64, 64, 3)
```



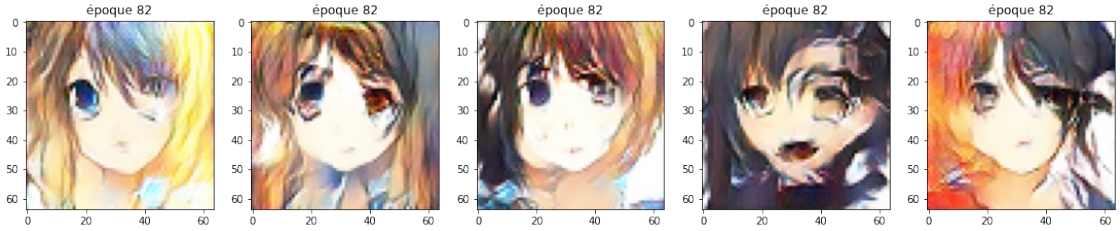

```
80, 1/128, d=0.750, g=0.738
80, 51/128, d=0.693, g=0.577
80, 101/128, d=0.671, g=0.685
Accuracy real: 16%, fake: 91%
(5, 64, 64, 3)
```



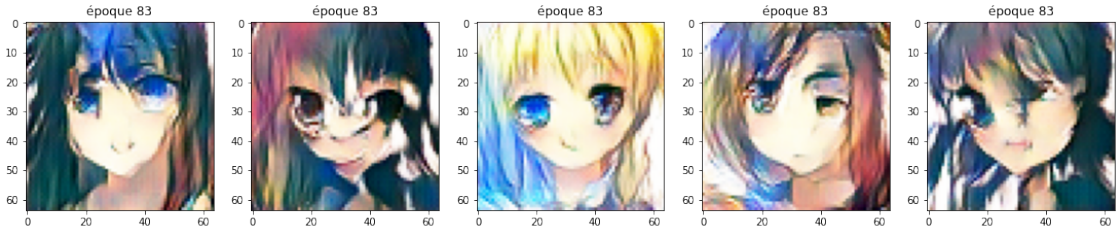

```
81, 1/128, d=0.643, g=0.846
81, 51/128, d=0.698, g=0.730
81, 101/128, d=0.668, g=0.691
Accuracy real: 45%, fake: 76%
(5, 64, 64, 3)
```



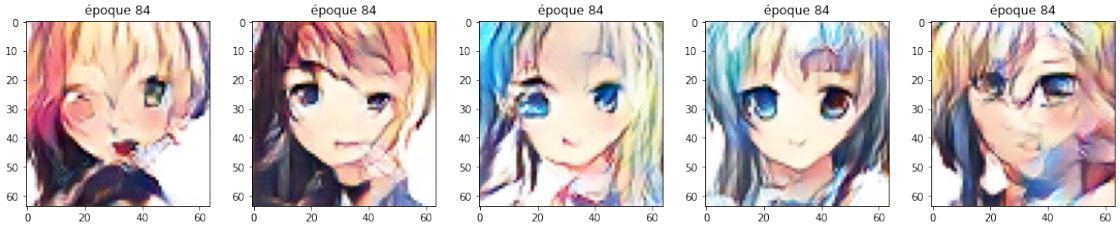

```
82, 1/128, d=0.663, g=0.717
82, 51/128, d=0.653, g=0.680
82, 101/128, d=0.694, g=0.772
Accuracy real: 55%, fake: 78%
(5, 64, 64, 3)
```



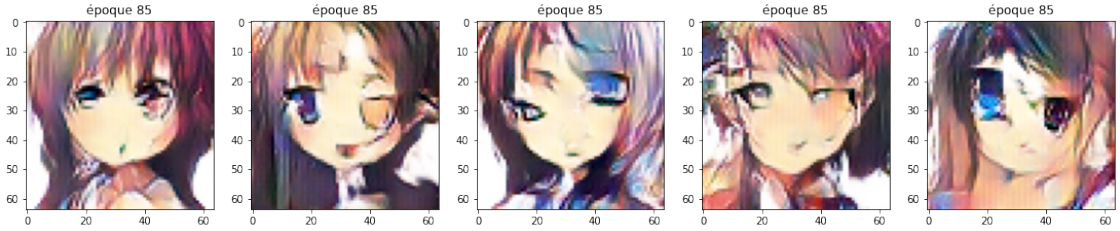

```
83, 1/128, d=0.656, g=0.468
83, 51/128, d=0.674, g=0.651
83, 101/128, d=0.689, g=0.697
Accuracy real: 64%, fake: 28%
(5, 64, 64, 3)
```



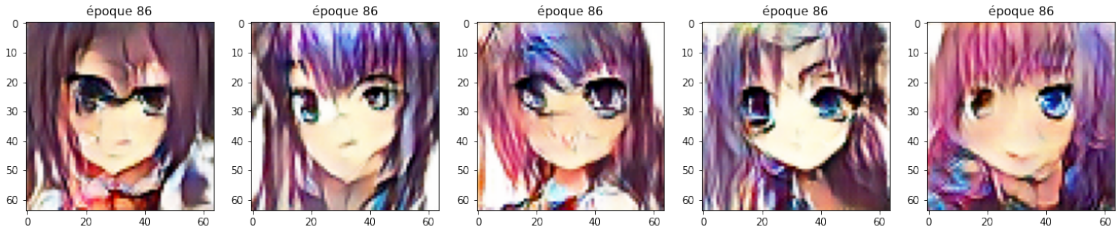

```
84, 1/128, d=0.710, g=0.699
84, 51/128, d=0.681, g=0.602
84, 101/128, d=0.655, g=0.709
Accuracy real: 50%, fake: 73%
(5, 64, 64, 3)
```



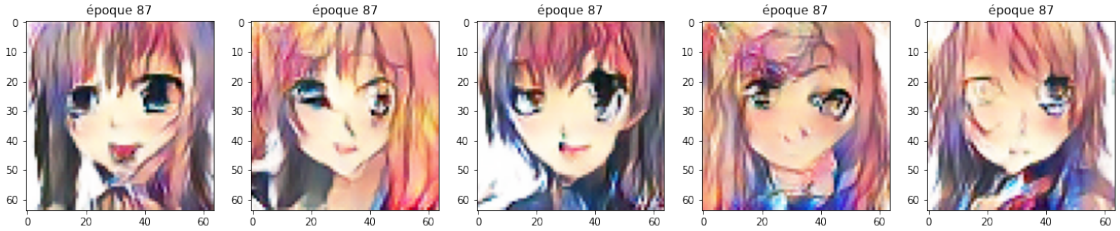

```
85, 1/128, d=0.657, g=0.812
85, 51/128, d=0.710, g=0.938
85, 101/128, d=0.680, g=0.795
Accuracy real: 70%, fake: 74%
(5, 64, 64, 3)
```



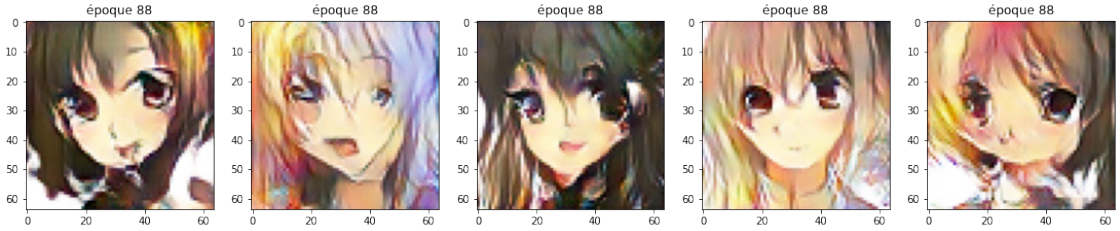

```
86, 1/128, d=0.660, g=0.582
86, 51/128, d=0.670, g=0.728
86, 101/128, d=0.641, g=0.710
Accuracy real: 85%, fake: 18%
(5, 64, 64, 3)
```



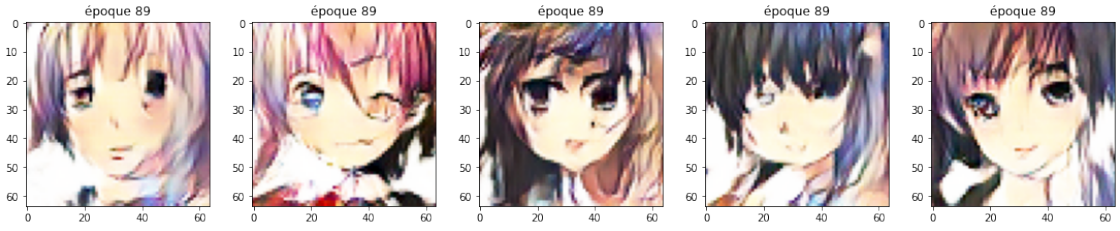

```
87, 1/128, d=0.769, g=1.108
87, 51/128, d=0.711, g=1.034
87, 101/128, d=0.684, g=0.717
Accuracy real: 64%, fake: 38%
(5, 64, 64, 3)
```



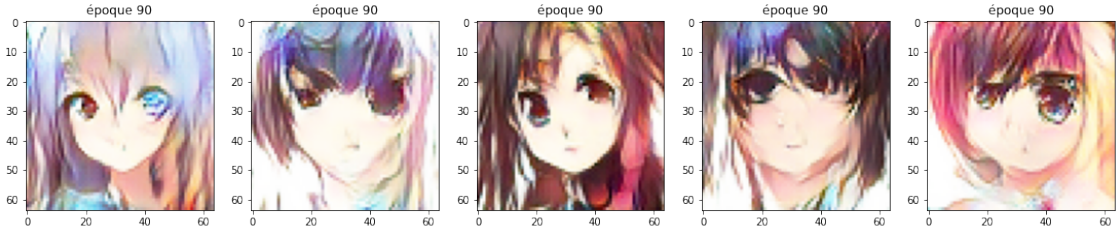

```
88, 1/128, d=0.701, g=0.705
88, 51/128, d=0.689, g=0.617
88, 101/128, d=0.677, g=0.710
Accuracy real: 92%, fake: 2%
(5, 64, 64, 3)
```



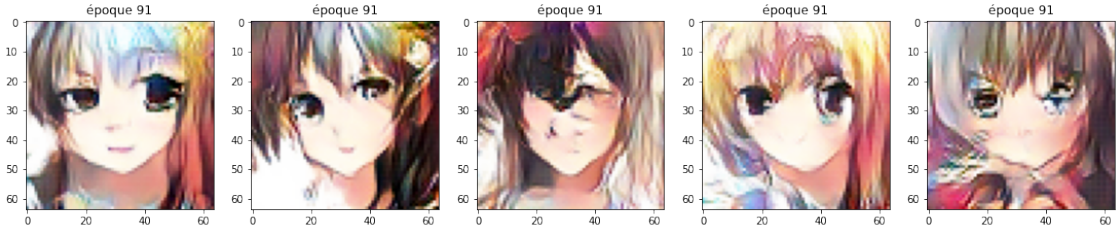

```
89, 1/128, d=0.756, g=0.765
89, 51/128, d=0.686, g=0.687
89, 101/128, d=0.690, g=0.799
Accuracy real: 80%, fake: 24%
(5, 64, 64, 3)
```



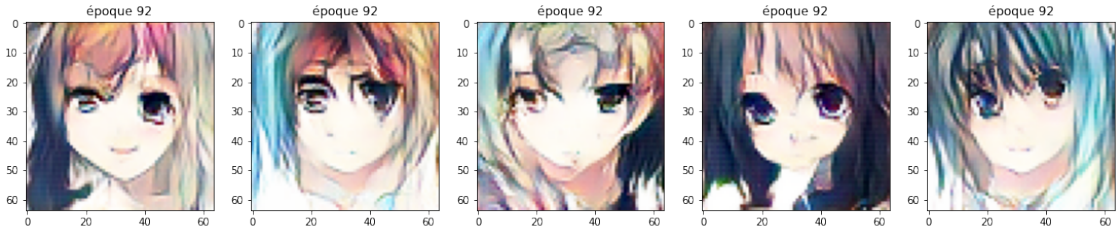

```
90, 1/128, d=0.717, g=0.734
90, 51/128, d=0.678, g=0.771
90, 101/128, d=0.677, g=0.761
Accuracy real: 88%, fake: 17%
(5, 64, 64, 3)
```



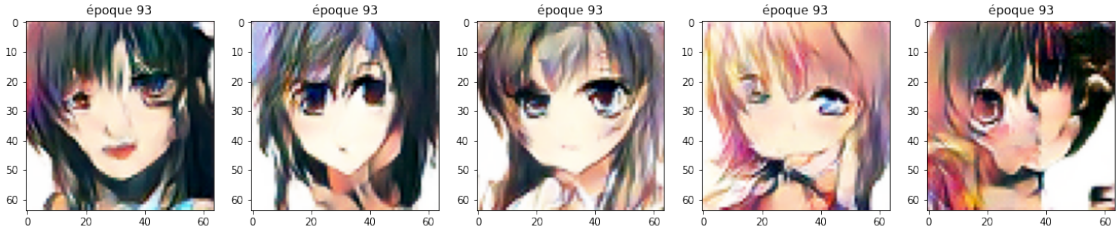

```
91, 1/128, d=0.707, g=0.604
91, 51/128, d=0.698, g=0.703
91, 101/128, d=0.682, g=0.678
Accuracy real: 41%, fake: 86%
(5, 64, 64, 3)
```



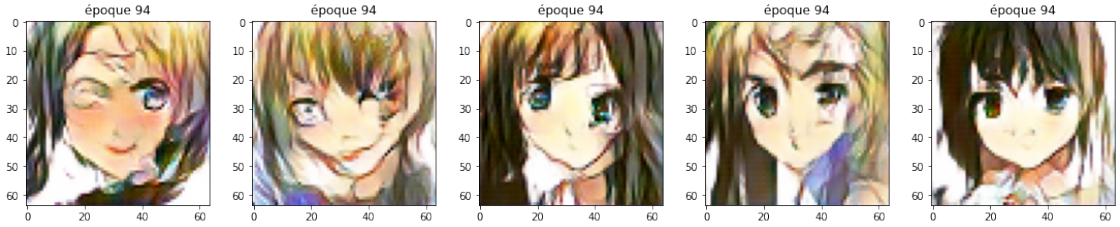

```
92, 1/128, d=0.655, g=0.689
92, 51/128, d=0.711, g=0.696
92, 101/128, d=0.737, g=0.826
Accuracy real: 86%, fake: 7%
(5, 64, 64, 3)
```



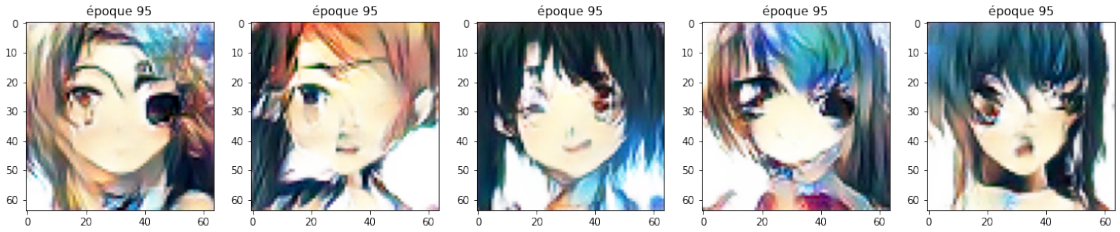

```
93, 1/128, d=0.800, g=1.066
93, 51/128, d=0.682, g=0.783
93, 101/128, d=0.695, g=0.722
Accuracy real: 55%, fake: 58%
(5, 64, 64, 3)
```



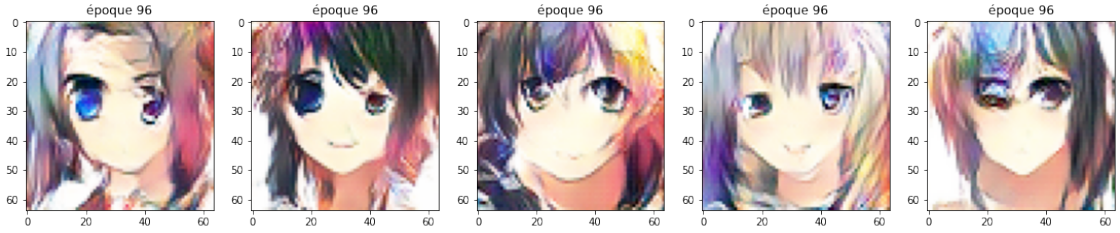

```
94, 1/128, d=0.683, g=0.844
94, 51/128, d=0.662, g=0.789
94, 101/128, d=0.697, g=0.806
Accuracy real: 28%, fake: 72%
(5, 64, 64, 3)
```



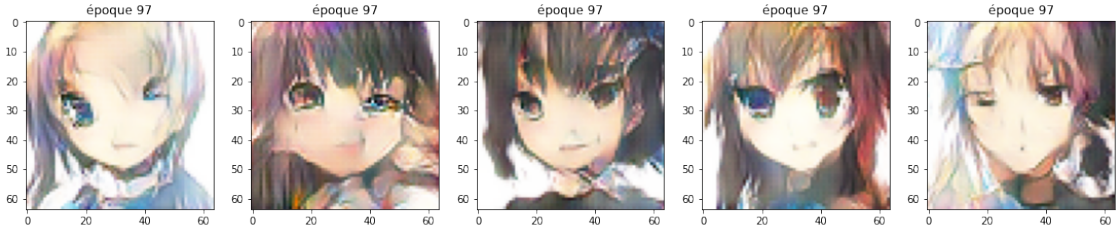

```
95, 1/128, d=0.683, g=0.808
95, 51/128, d=0.670, g=0.693
95, 101/128, d=0.720, g=0.916
Accuracy real: 13%, fake: 75%
(5, 64, 64, 3)
```



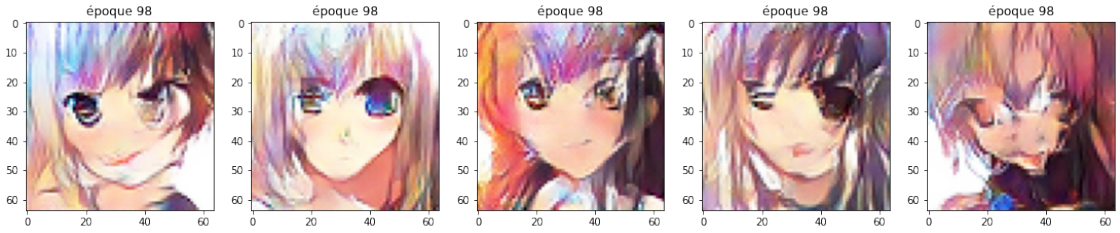

```
96, 1/128, d=0.665, g=0.725
96, 51/128, d=0.678, g=0.729
96, 101/128, d=0.679, g=0.645
Accuracy real: 71%, fake: 28%
(5, 64, 64, 3)
```



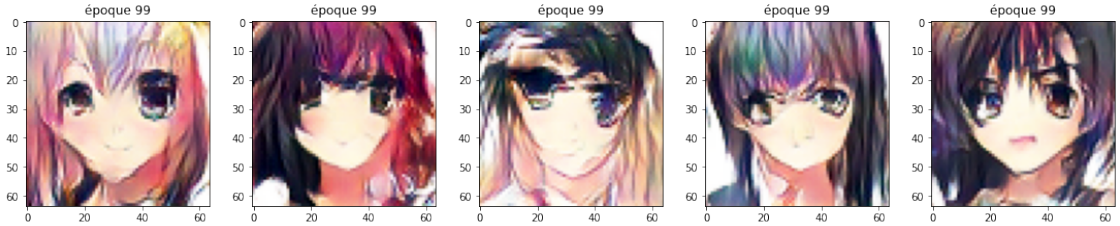

```
97, 1/128, d=0.708, g=0.793
97, 51/128, d=0.678, g=0.885
97, 101/128, d=0.697, g=0.696
Accuracy real: 0%, fake: 100%
(5, 64, 64, 3)
```



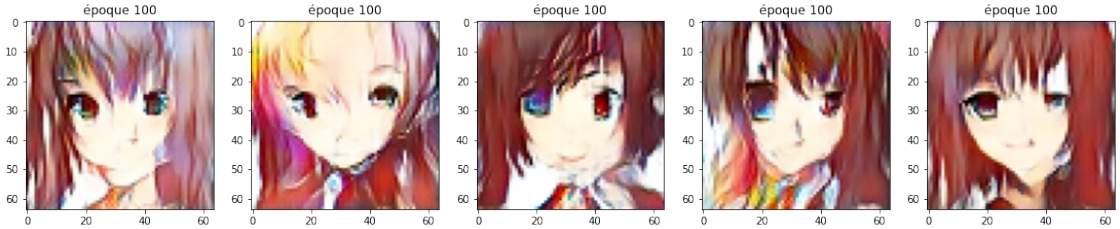

```
98, 1/128, d=0.567, g=0.887
98, 51/128, d=0.680, g=0.712
98, 101/128, d=0.671, g=0.720
Accuracy real: 67%, fake: 36%
(5, 64, 64, 3)
```



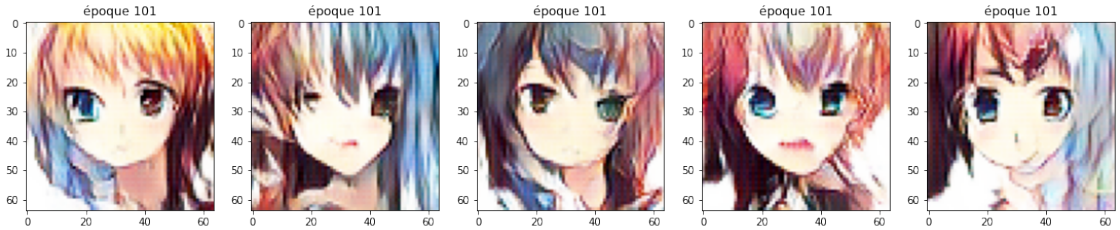

```
99, 1/128, d=0.719, g=0.745
99, 51/128, d=0.736, g=0.899
99, 101/128, d=0.678, g=0.715
Accuracy real: 66%, fake: 39%
(5, 64, 64, 3)
```



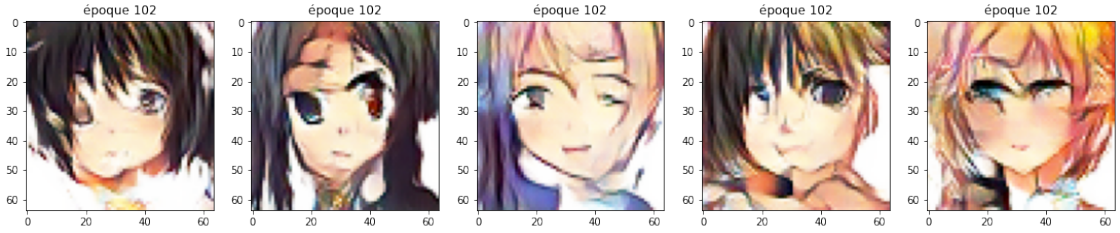

```
100, 1/128, d=0.699, g=0.641
100, 51/128, d=0.681, g=0.676
100, 101/128, d=0.712, g=0.695
Accuracy real: 80%, fake: 44%
(5, 64, 64, 3)
```



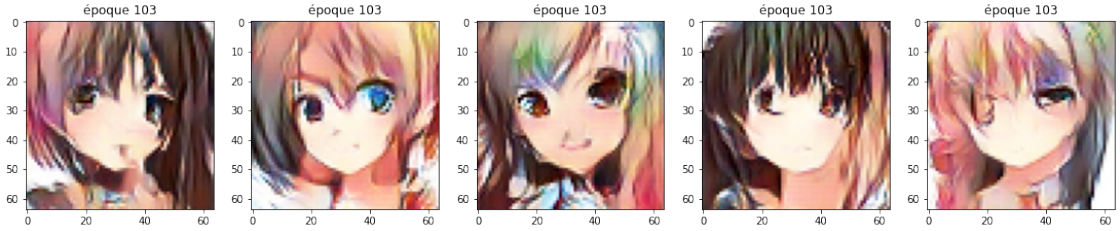

```
101, 1/128, d=0.681, g=0.817
101, 51/128, d=0.700, g=0.767
101, 101/128, d=0.651, g=0.791
Accuracy real: 54%, fake: 93%
(5, 64, 64, 3)
```



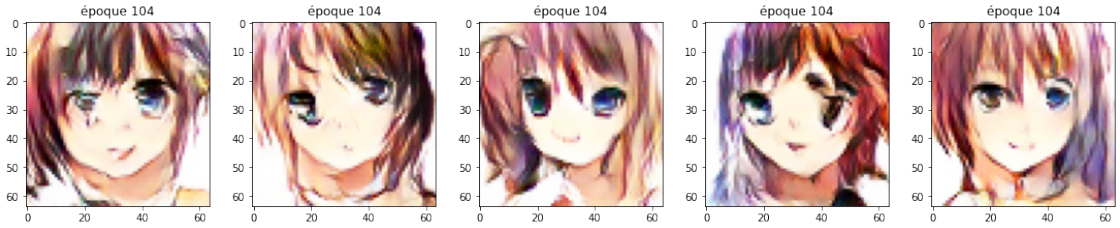

```
102, 1/128, d=0.627, g=0.715
102, 51/128, d=0.683, g=0.793
102, 101/128, d=0.683, g=0.780
Accuracy real: 34%, fake: 79%
(5, 64, 64, 3)
```



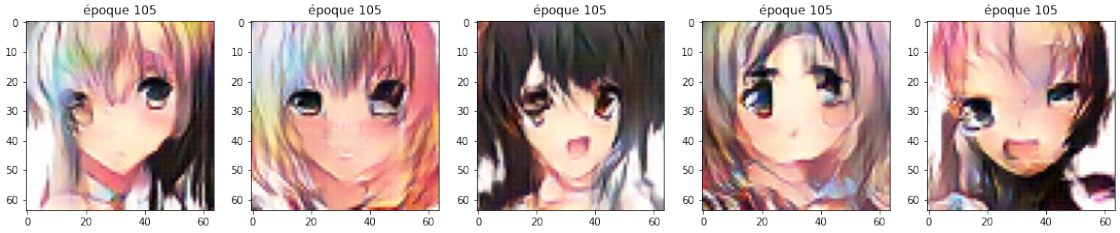

```
103, 1/128, d=0.669, g=0.739
103, 51/128, d=0.653, g=0.726
103, 101/128, d=0.663, g=0.865
Accuracy real: 33%, fake: 90%
(5, 64, 64, 3)
```



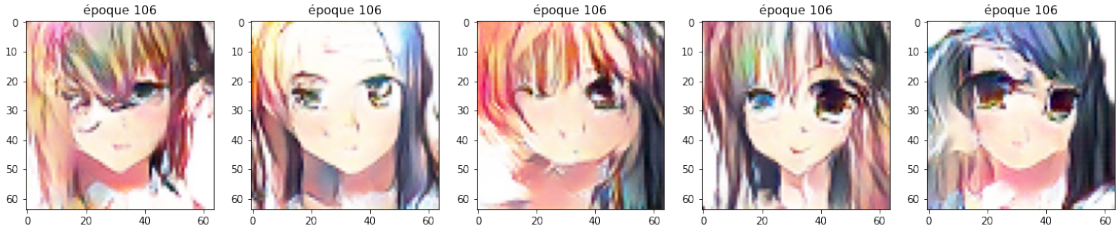

```
104, 1/128, d=0.637, g=0.776
104, 51/128, d=0.675, g=0.680
104, 101/128, d=0.681, g=0.704
Accuracy real: 63%, fake: 44%
(5, 64, 64, 3)
```



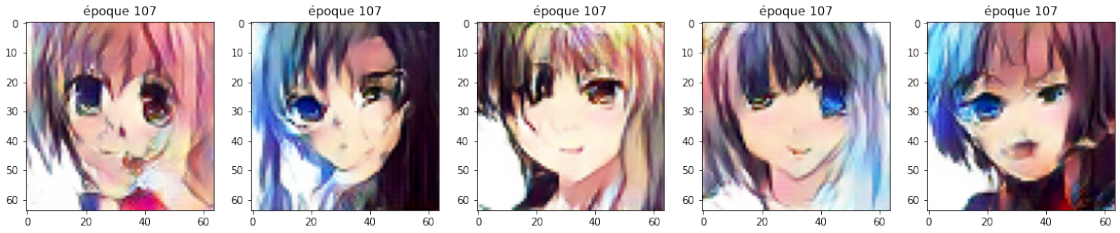

```
105, 1/128, d=0.709, g=0.748
105, 51/128, d=0.714, g=0.614
105, 101/128, d=0.717, g=0.836
Accuracy real: 28%, fake: 97%
(5, 64, 64, 3)
```



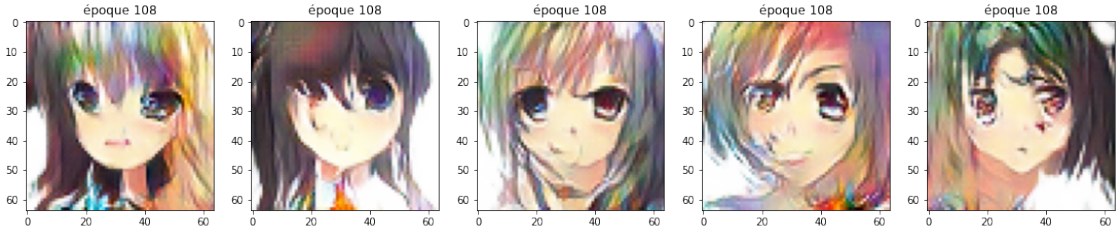

```
106, 1/128, d=0.648, g=0.707
106, 51/128, d=0.679, g=0.760
106, 101/128, d=0.697, g=0.623
Accuracy real: 24%, fake: 96%
(5, 64, 64, 3)
```



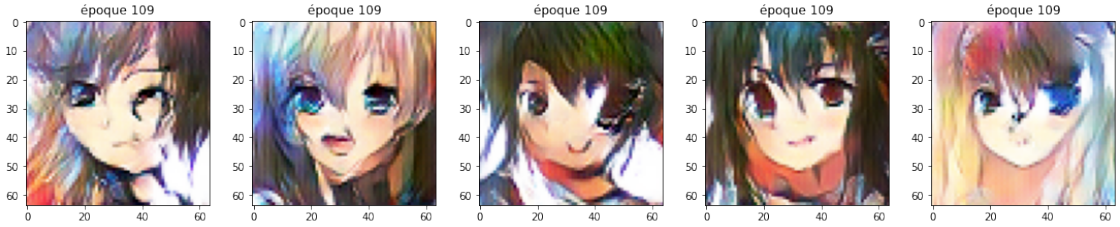

```
107, 1/128, d=0.634, g=0.847
107, 51/128, d=0.695, g=0.847
107, 101/128, d=0.657, g=0.731
Accuracy real: 87%, fake: 28%
(5, 64, 64, 3)
```



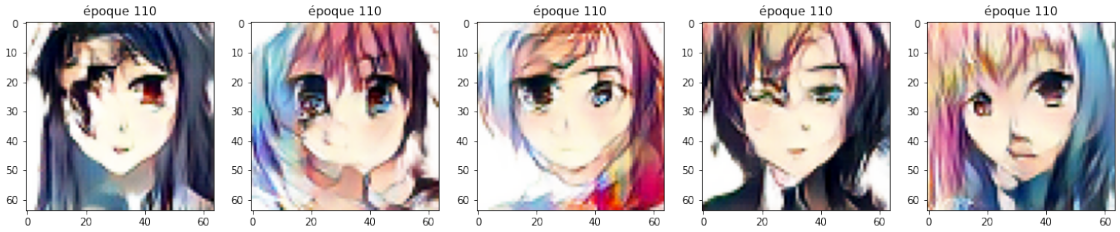

```
108, 1/128, d=0.711, g=0.732
108, 51/128, d=0.662, g=0.761
108, 101/128, d=0.680, g=0.662
Accuracy real: 39%, fake: 60%
(5, 64, 64, 3)
```



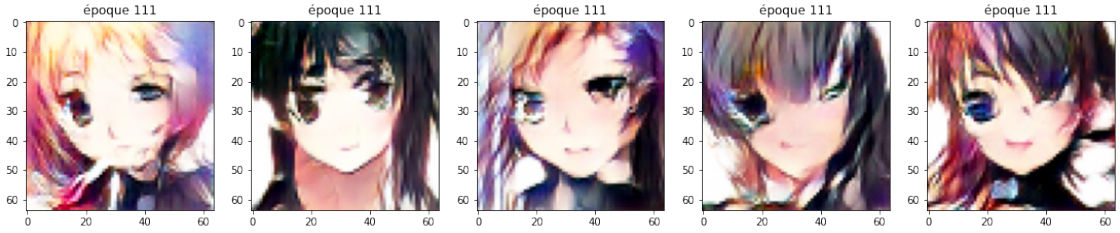

```
109, 1/128, d=0.681, g=0.762
109, 51/128, d=0.701, g=0.668
109, 101/128, d=0.693, g=0.679
Accuracy real: 52%, fake: 70%
(5, 64, 64, 3)
```



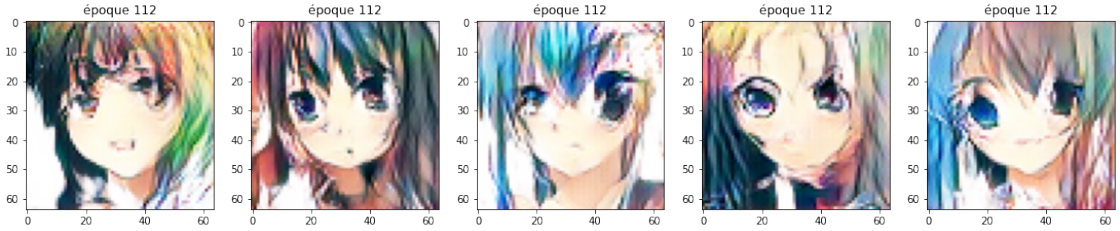

```
110, 1/128, d=0.657, g=0.671
110, 51/128, d=0.676, g=0.695
110, 101/128, d=0.696, g=0.817
Accuracy real: 74%, fake: 34%
(5, 64, 64, 3)
```



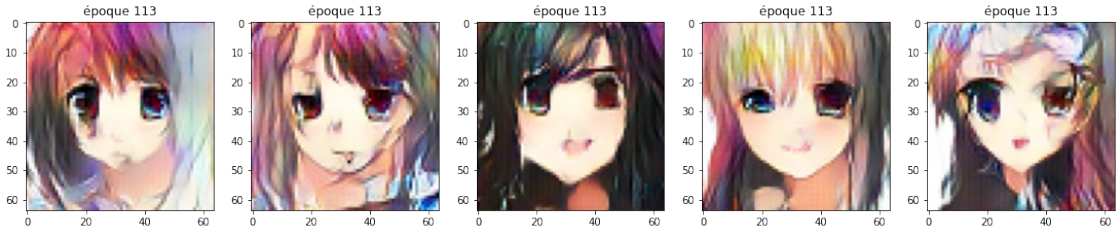

```
111, 1/128, d=0.710, g=0.704
111, 51/128, d=0.685, g=0.664
111, 101/128, d=0.700, g=0.802
Accuracy real: 44%, fake: 40%
(5, 64, 64, 3)
```



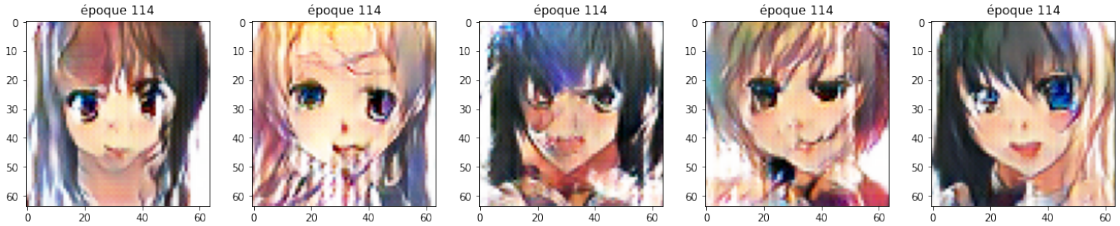

```
112, 1/128, d=0.722, g=0.803
112, 51/128, d=0.663, g=0.697
112, 101/128, d=0.683, g=0.783
Accuracy real: 18%, fake: 72%
(5, 64, 64, 3)
```



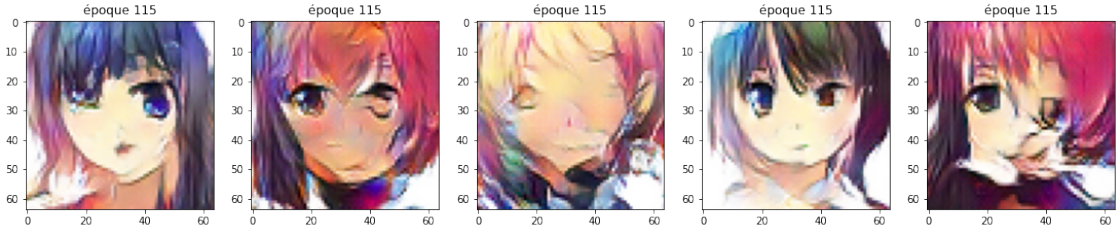

```
113, 1/128, d=0.670, g=0.840
113, 51/128, d=0.700, g=0.669
113, 101/128, d=0.712, g=0.823
Accuracy real: 21%, fake: 99%
(5, 64, 64, 3)
```



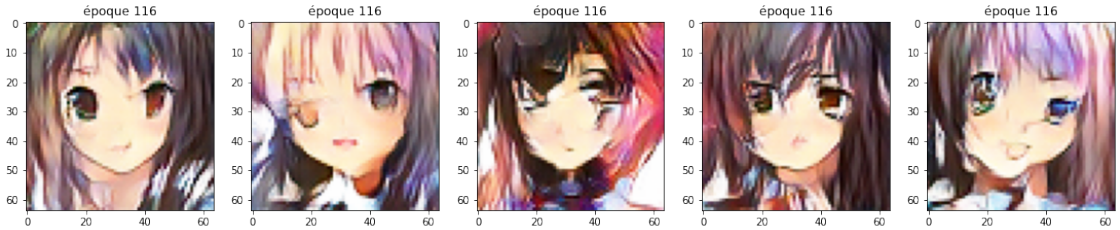

```
114, 1/128, d=0.606, g=0.650
114, 51/128, d=0.669, g=0.719
114, 101/128, d=0.742, g=0.855
Accuracy real: 89%, fake: 16%
(5, 64, 64, 3)
```



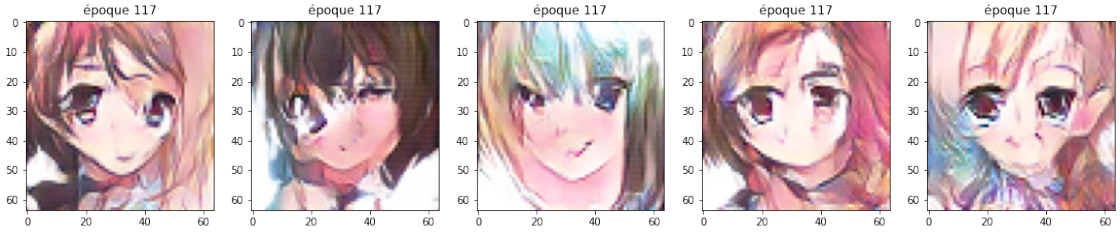

```
115, 1/128, d=0.723, g=0.744
115, 51/128, d=0.737, g=0.956
115, 101/128, d=0.700, g=0.904
Accuracy real: 86%, fake: 18%
(5, 64, 64, 3)
```



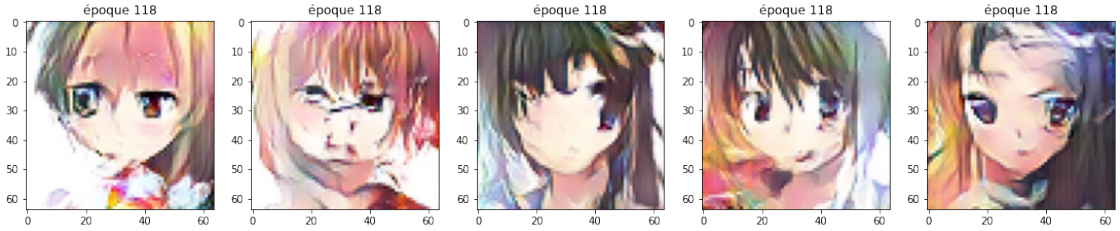

```
116, 1/128, d=0.725, g=0.704
116, 51/128, d=0.676, g=0.855
116, 101/128, d=0.692, g=0.754
Accuracy real: 78%, fake: 22%
(5, 64, 64, 3)
```



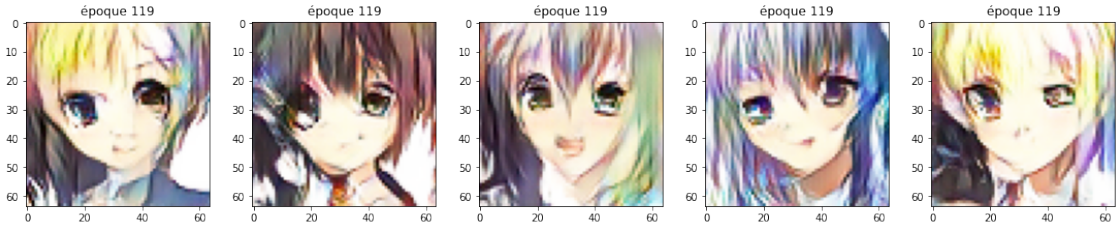

```
117, 1/128, d=0.726, g=0.794
117, 51/128, d=0.692, g=0.820
117, 101/128, d=0.708, g=0.798
Accuracy real: 50%, fake: 70%
(5, 64, 64, 3)
```



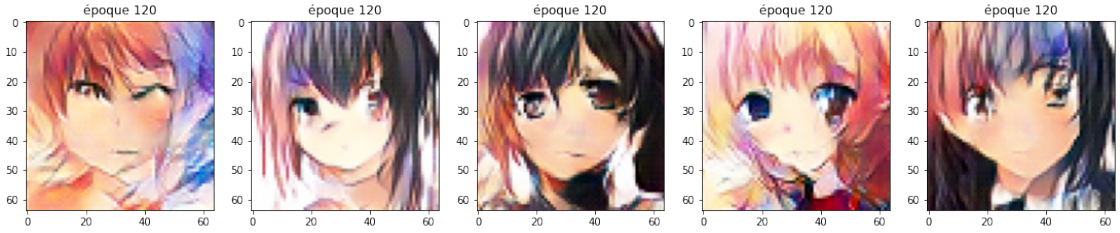

```
118, 1/128, d=0.675, g=0.690
118, 51/128, d=0.661, g=0.674
118, 101/128, d=0.682, g=0.815
Accuracy real: 86%, fake: 14%
(5, 64, 64, 3)
```



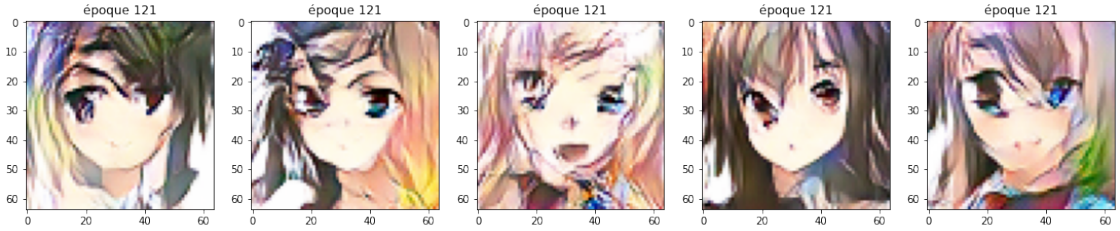

```
119, 1/128, d=0.733, g=0.887
119, 51/128, d=0.657, g=0.827
119, 101/128, d=0.684, g=0.685
Accuracy real: 42%, fake: 76%
(5, 64, 64, 3)
```



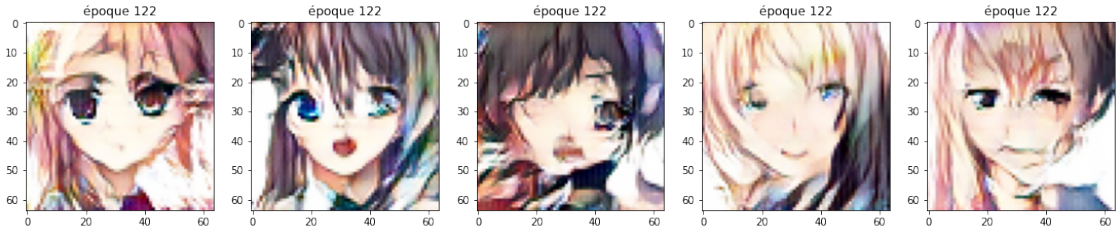

```
120, 1/128, d=0.662, g=0.768
120, 51/128, d=0.688, g=0.789
120, 101/128, d=0.679, g=0.713
Accuracy real: 77%, fake: 42%
(5, 64, 64, 3)
```



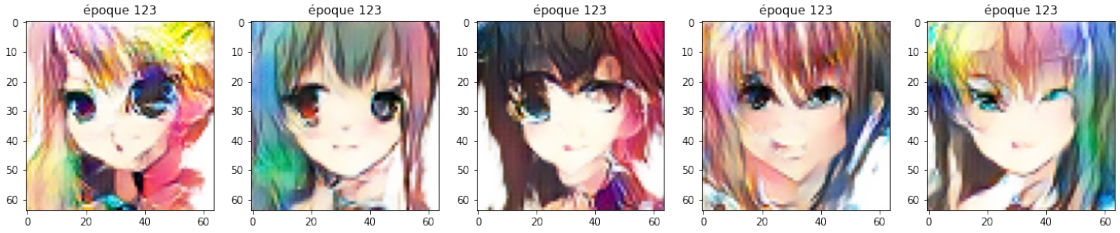

```
121, 1/128, d=0.712, g=0.683
121, 51/128, d=0.678, g=0.788
121, 101/128, d=0.678, g=0.741
Accuracy real: 45%, fake: 75%
(5, 64, 64, 3)
```



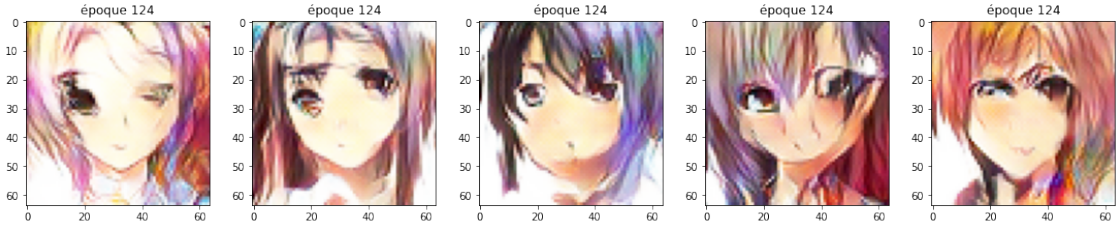

```
122, 1/128, d=0.661, g=0.673
122, 51/128, d=0.719, g=0.745
122, 101/128, d=0.668, g=0.713
Accuracy real: 15%, fake: 95%
(5, 64, 64, 3)
```



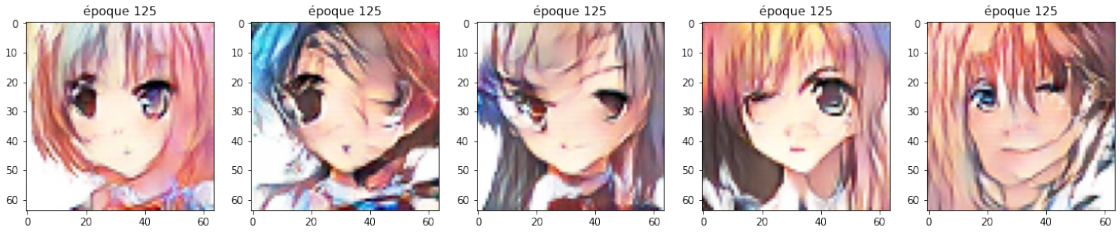

```
123, 1/128, d=0.646, g=0.785
123, 51/128, d=0.699, g=0.670
123, 101/128, d=0.679, g=0.772
Accuracy real: 65%, fake: 39%
(5, 64, 64, 3)
```



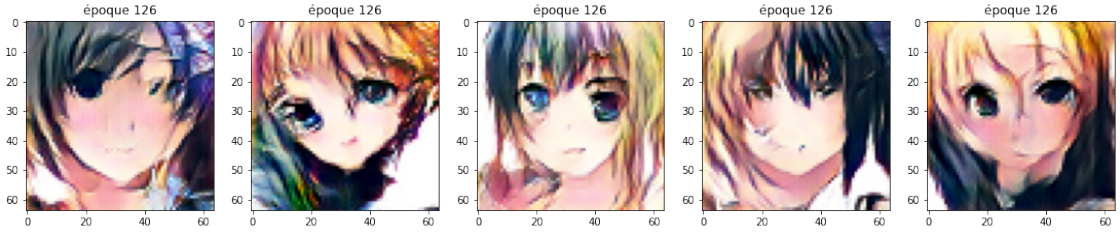

```
124, 1/128, d=0.701, g=0.731
124, 51/128, d=0.681, g=0.701
124, 101/128, d=0.665, g=0.617
Accuracy real: 58%, fake: 50%
(5, 64, 64, 3)
```




```
125, 1/128, d=0.695, g=0.735
125, 51/128, d=0.706, g=0.793
125, 101/128, d=0.697, g=0.735
Accuracy real: 23%, fake: 94%
(5, 64, 64, 3)
```

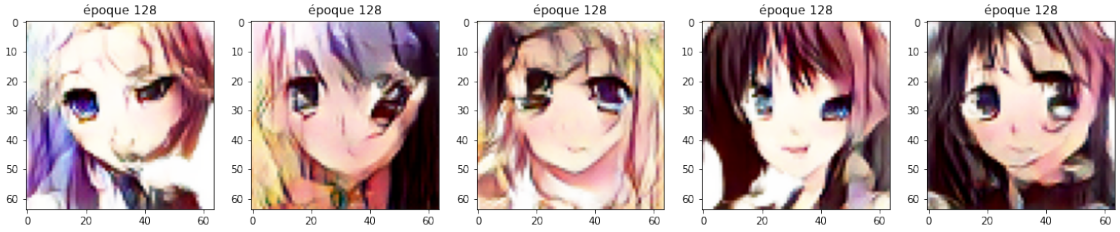



```
126, 1/128, d=0.638, g=0.658
126, 51/128, d=0.670, g=0.897
126, 101/128, d=0.684, g=0.666
Accuracy real: 38%, fake: 74%
(5, 64, 64, 3)
```

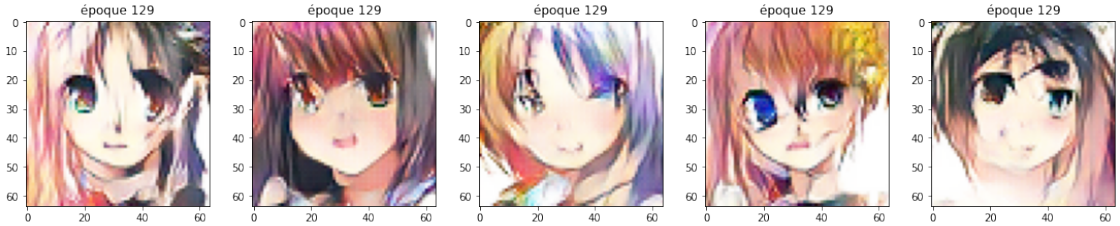



```
127, 1/128, d=0.673, g=0.846
127, 51/128, d=0.714, g=0.910
127, 101/128, d=0.664, g=0.742
Accuracy real: 58%, fake: 56%
(5, 64, 64, 3)
```

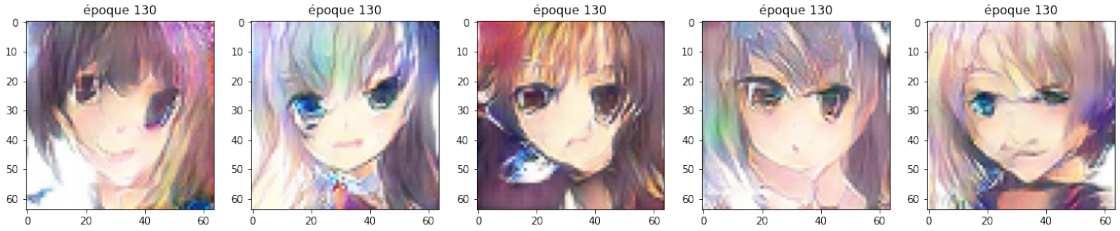

```
128, 1/128, d=0.682, g=0.635
128, 51/128, d=0.674, g=0.745
128, 101/128, d=0.665, g=0.684
Accuracy real: 59%, fake: 61%
(5, 64, 64, 3)
```



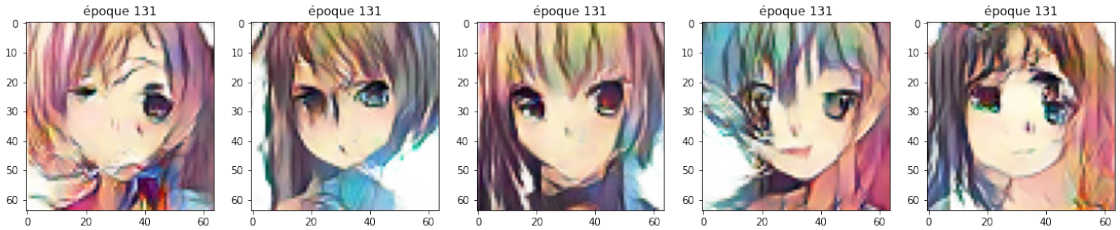

```
129, 1/128, d=0.673, g=0.755
129, 51/128, d=0.730, g=0.758
129, 101/128, d=0.675, g=0.720
Accuracy real: 83%, fake: 12%
(5, 64, 64, 3)
```



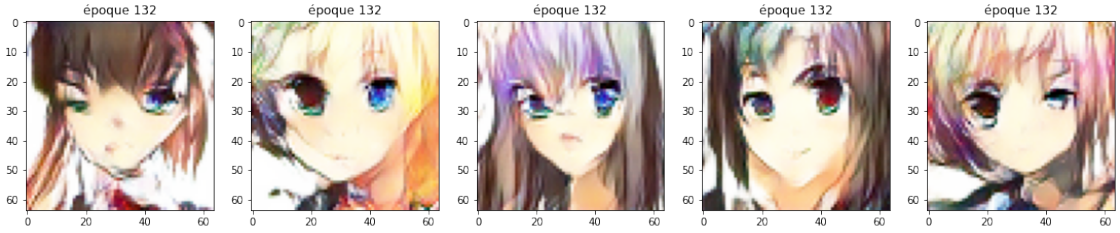

```
130, 1/128, d=0.741, g=0.687
130, 51/128, d=0.664, g=0.739
130, 101/128, d=0.686, g=0.764
Accuracy real: 77%, fake: 2%
(5, 64, 64, 3)
```



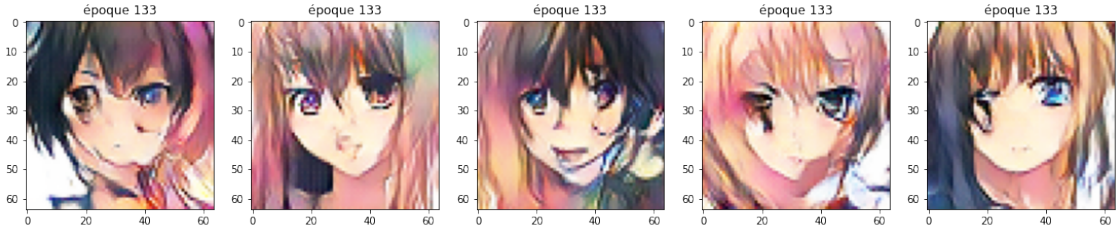

```
131, 1/128, d=0.801, g=0.985
131, 51/128, d=0.702, g=0.779
131, 101/128, d=0.664, g=0.656
Accuracy real: 100%, fake: 4%
(5, 64, 64, 3)
```



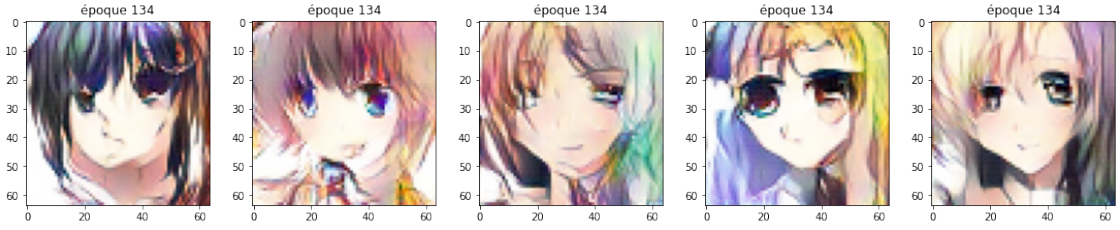

```
132, 1/128, d=0.794, g=0.880
132, 51/128, d=0.700, g=0.718
132, 101/128, d=0.712, g=0.818
Accuracy real: 51%, fake: 57%
(5, 64, 64, 3)
```



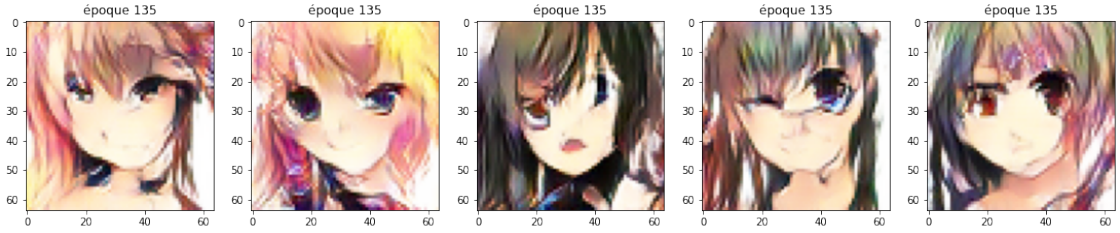

```
133, 1/128, d=0.687, g=0.687
133, 51/128, d=0.686, g=0.771
133, 101/128, d=0.746, g=0.743
Accuracy real: 62%, fake: 85%
(5, 64, 64, 3)
```



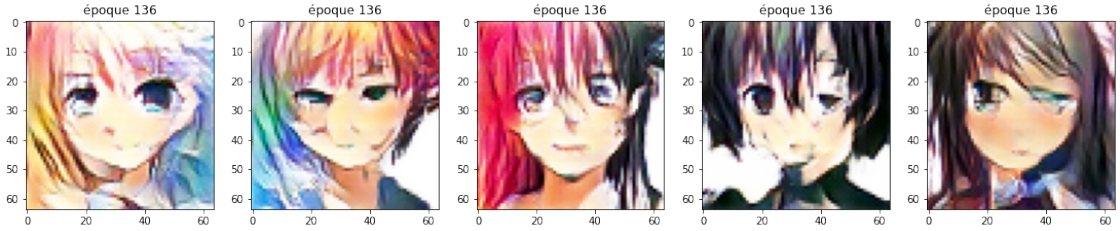

```
134, 1/128, d=0.643, g=0.827
134, 51/128, d=0.672, g=0.838
134, 101/128, d=0.687, g=0.836
Accuracy real: 22%, fake: 97%
(5, 64, 64, 3)
```



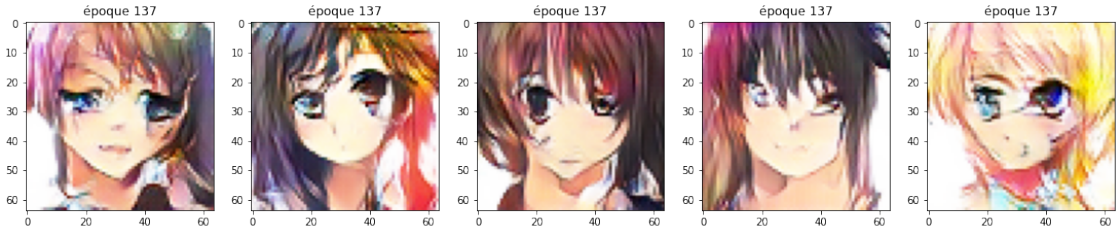

```
135, 1/128, d=0.647, g=0.759
135, 51/128, d=0.663, g=0.720
135, 101/128, d=0.679, g=0.718
Accuracy real: 68%, fake: 33%
(5, 64, 64, 3)
```



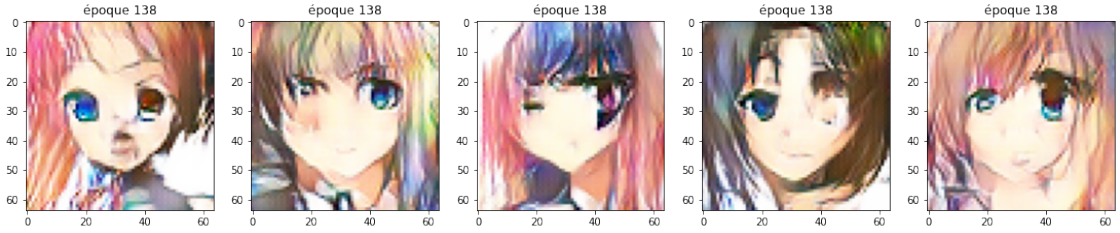

```
136, 1/128, d=0.716, g=0.732
136, 51/128, d=0.733, g=0.851
136, 101/128, d=0.694, g=0.790
Accuracy real: 19%, fake: 90%
(5, 64, 64, 3)
```



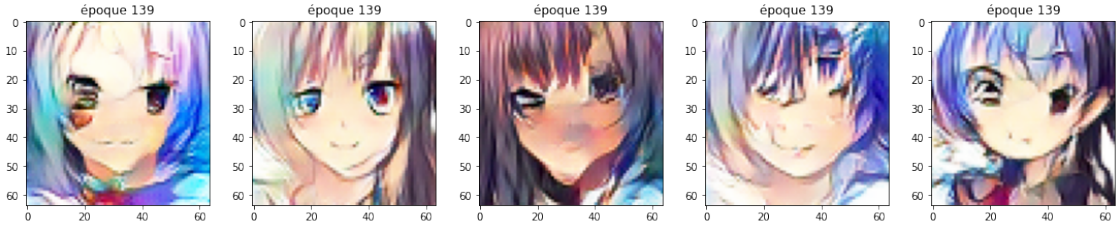

```
137, 1/128, d=0.612, g=0.849
137, 51/128, d=0.666, g=0.680
137, 101/128, d=0.657, g=0.846
Accuracy real: 65%, fake: 71%
(5, 64, 64, 3)
```



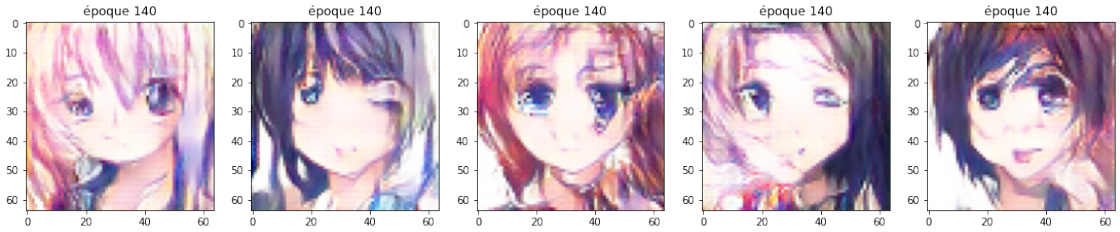

```
138, 1/128, d=0.668, g=0.703
138, 51/128, d=0.673, g=0.612
138, 101/128, d=0.700, g=0.956
Accuracy real: 66%, fake: 47%
(5, 64, 64, 3)
```



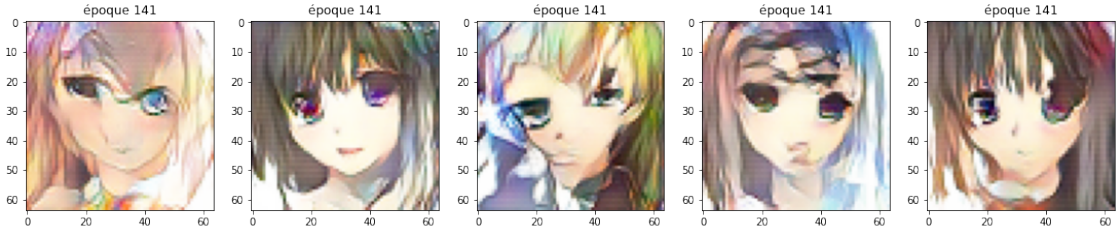

```
139, 1/128, d=0.685, g=0.851
139, 51/128, d=0.675, g=0.802
139, 101/128, d=0.682, g=0.795
Accuracy real: 67%, fake: 53%
(5, 64, 64, 3)
```



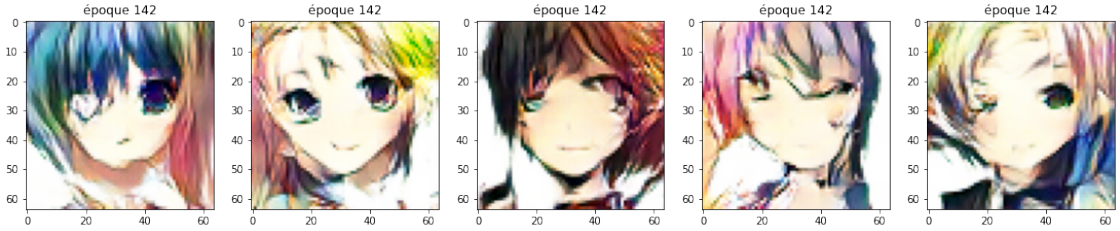

```
140, 1/128, d=0.680, g=0.750
140, 51/128, d=0.689, g=0.879
140, 101/128, d=0.659, g=0.835
Accuracy real: 81%, fake: 8%
(5, 64, 64, 3)
```



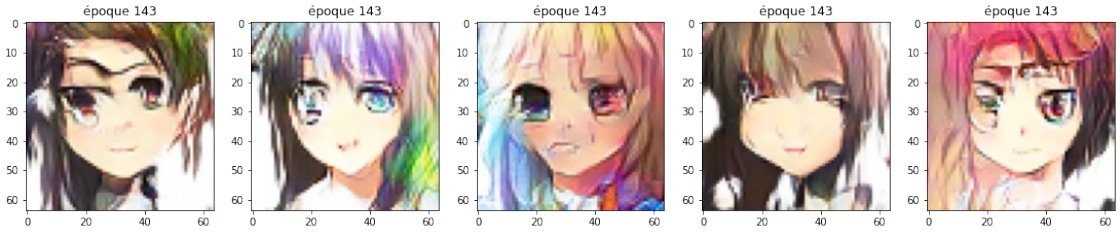

```
141, 1/128, d=0.824, g=0.954
141, 51/128, d=0.680, g=0.810
141, 101/128, d=0.668, g=0.730
Accuracy real: 0%, fake: 100%
(5, 64, 64, 3)
```



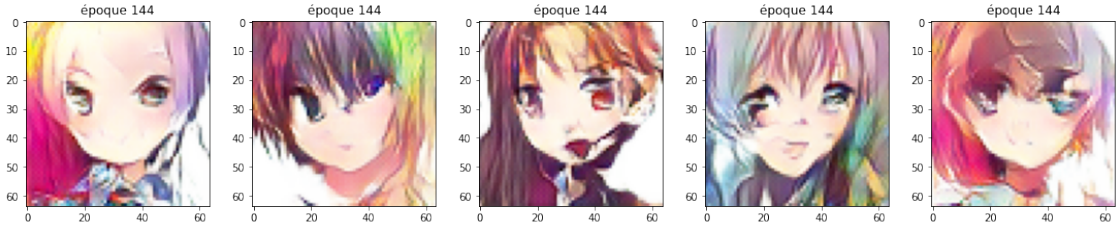

```
142, 1/128, d=0.552, g=0.860
142, 51/128, d=0.677, g=0.720
142, 101/128, d=0.672, g=0.752
Accuracy real: 22%, fake: 90%
(5, 64, 64, 3)
```



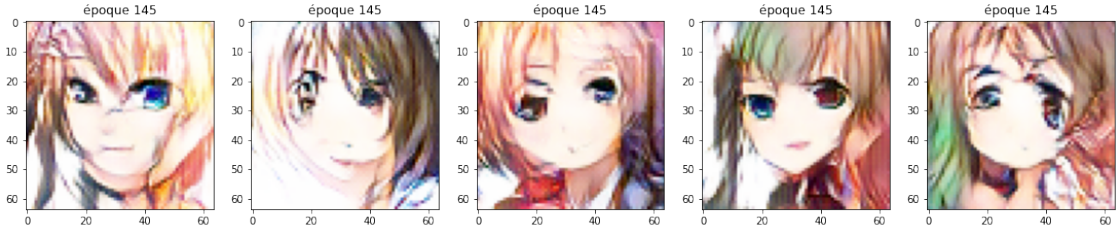

```
143, 1/128, d=0.624, g=0.672
143, 51/128, d=0.663, g=0.759
143, 101/128, d=0.638, g=0.874
Accuracy real: 39%, fake: 95%
(5, 64, 64, 3)
```



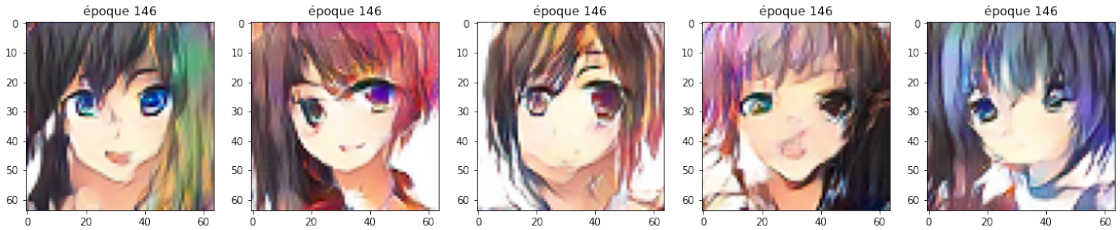

```
144, 1/128, d=0.640, g=0.715
144, 51/128, d=0.695, g=0.776
144, 101/128, d=0.679, g=0.717
Accuracy real: 66%, fake: 59%
(5, 64, 64, 3)
```



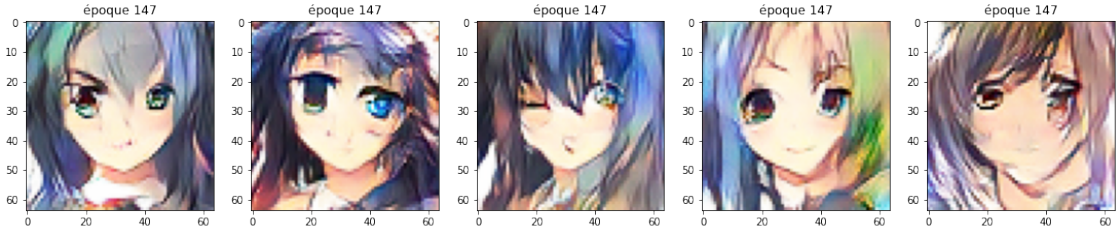

```
145, 1/128, d=0.684, g=0.909
145, 51/128, d=0.677, g=0.735
145, 101/128, d=0.690, g=0.899
Accuracy real: 38%, fake: 62%
(5, 64, 64, 3)
```



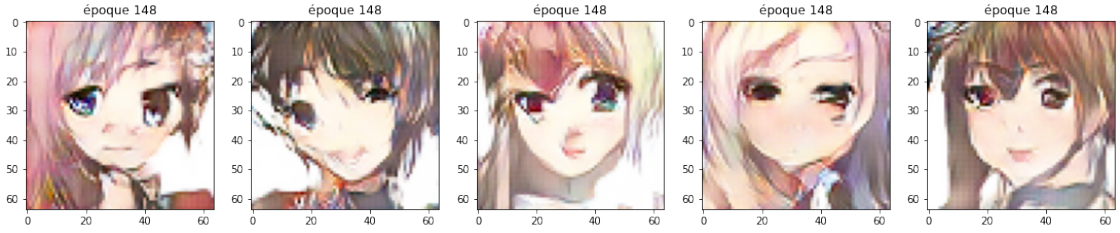

```
146, 1/128, d=0.681, g=0.675
146, 51/128, d=0.672, g=0.727
146, 101/128, d=0.679, g=0.697
Accuracy real: 19%, fake: 100%
(5, 64, 64, 3)
```



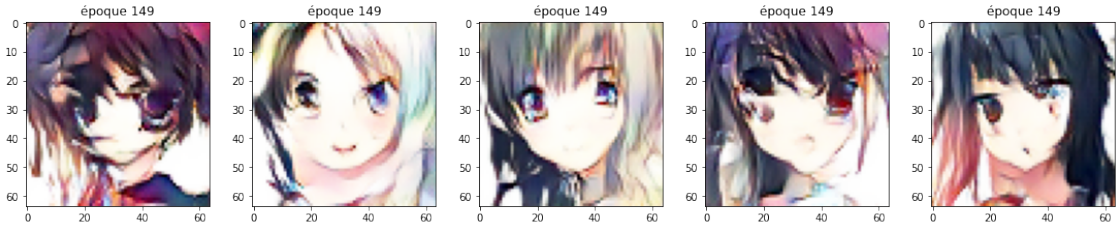

```
147, 1/128, d=0.615, g=0.722
147, 51/128, d=0.690, g=0.791
147, 101/128, d=0.648, g=0.687
Accuracy real: 74%, fake: 34%
(5, 64, 64, 3)
```



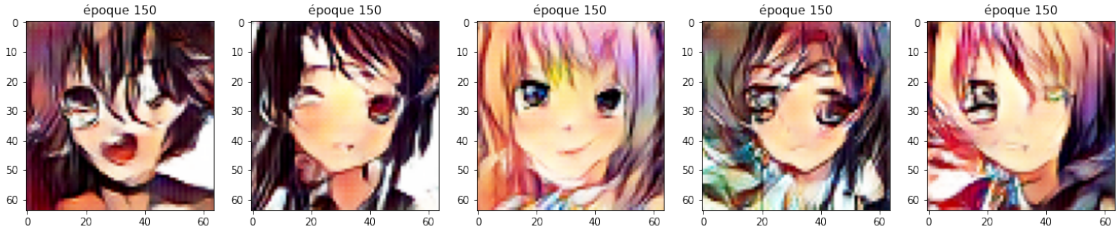

```
148, 1/128, d=0.711, g=0.894
148, 51/128, d=0.685, g=0.746
148, 101/128, d=0.752, g=0.911
Accuracy real: 87%, fake: 35%
(5, 64, 64, 3)
```




```
149, 1/128, d=0.687, g=0.760
149, 51/128, d=0.672, g=0.734
149, 101/128, d=0.694, g=0.977
Accuracy real: 45%, fake: 79%
(5, 64, 64, 3)
```

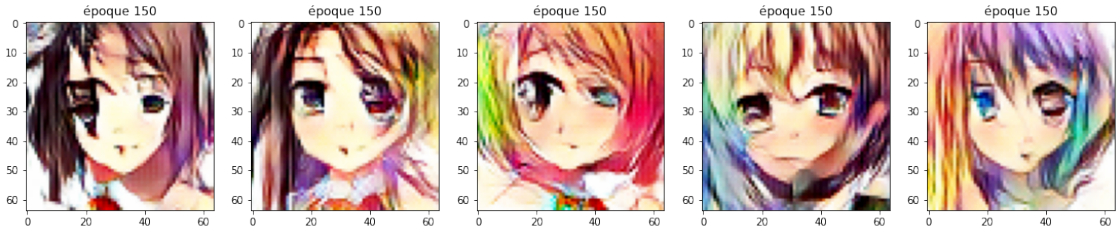



```
150, 1/128, d=0.645, g=0.793
150, 51/128, d=0.679, g=0.648
150, 101/128, d=0.699, g=0.791
Accuracy real: 7%, fake: 95%
(5, 64, 64, 3)
```

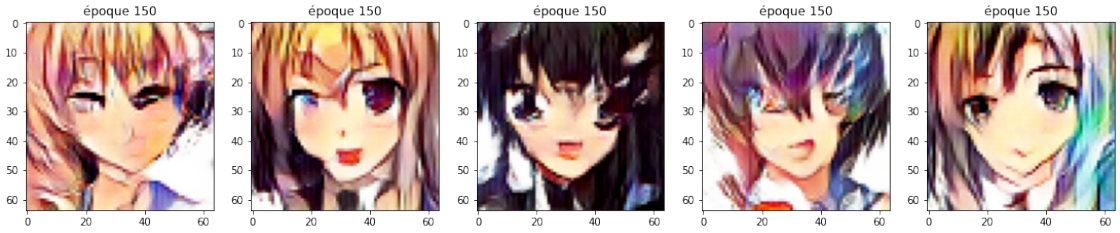


[16]:
```
plot_images(NUM_EPOCHS, generateur)
plot_images(NUM_EPOCHS, generateur)
plot_images(NUM_EPOCHS, generateur)
plot_images(NUM_EPOCHS, generateur)
```

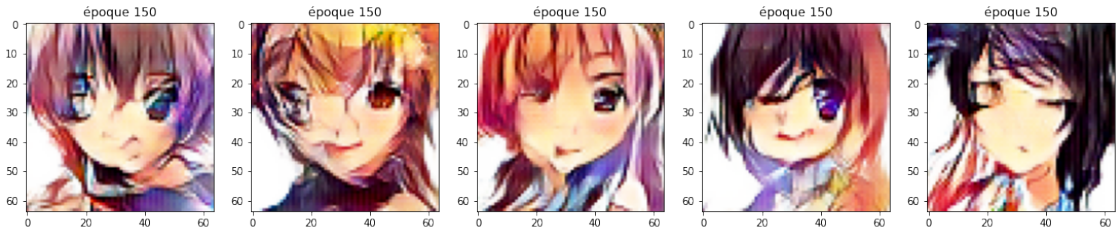
```
(5, 64, 64, 3)
```



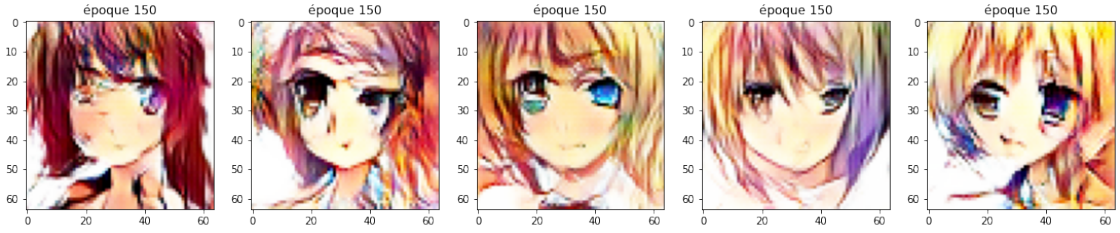

```
(5, 64, 64, 3)
```




```
(5, 64, 64, 3)
```



```
(5, 64, 64, 3)
```



[18]:
```
generateur.save('model_anime_' +str(NUM_EPOCHS)+ '.h5')
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.
```

[ ]:
```

```