In [13]:

```
from tensorflow import keras
model = keras.models.load_model('./model_final')
```

In [ ]:

In [14]:

```
import PIL
import tensorflow as tf
from PIL import Image
from tensorflow.keras.utils import array_to_img
from tensorflow.keras.utils import img_to_array
import numpy as np
import matplotlib.pyplot as plt
import PIL
```

In [ ]:

In [ ]:

In [15]:

```
def get_lowres_image(img, upscale_factor):
 """Return low-resolution image to use as model input."""
 return img.resize(
 (img.size[0] // upscale_factor, img.size[1] // upscale_factor),
 PIL.Image.BICUBIC,
 )
```

In [16]:

```
def upscale_image(model, img):
 """Predict the result based on input image and restore the image as RGB."""
 ycbcr = img.convert("YCbCr")
 y, cb, cr = ycbcr.split()
 y = img_to_array(y)
 y = y.astype("float32") / 255.0
 input = np.expand_dims(y, axis=0)
 out = model.predict(input)
 out_img_y = out[0]
 out_img_y *= 255.0
 # Restore the image in RGB color space.
 out_img_y = out_img_y.clip(0, 255)
 out_img_y = out_img_y.reshape((np.shape(out_img_y)[0], np.shape(out_img_y)[1]))
 out_img_y = PIL.Image.fromarray(np.uint8(out_img_y), mode="L")
 out_img_cb = cb.resize(out_img_y.size, PIL.Image.BICUBIC)
 out_img_cr = cr.resize(out_img_y.size, PIL.Image.BICUBIC)
 out_img = PIL.Image.merge("YCbCr", (out_img_y, out_img_cb,out_img_cr)).convert(
 "RGB")
 return out_img
```

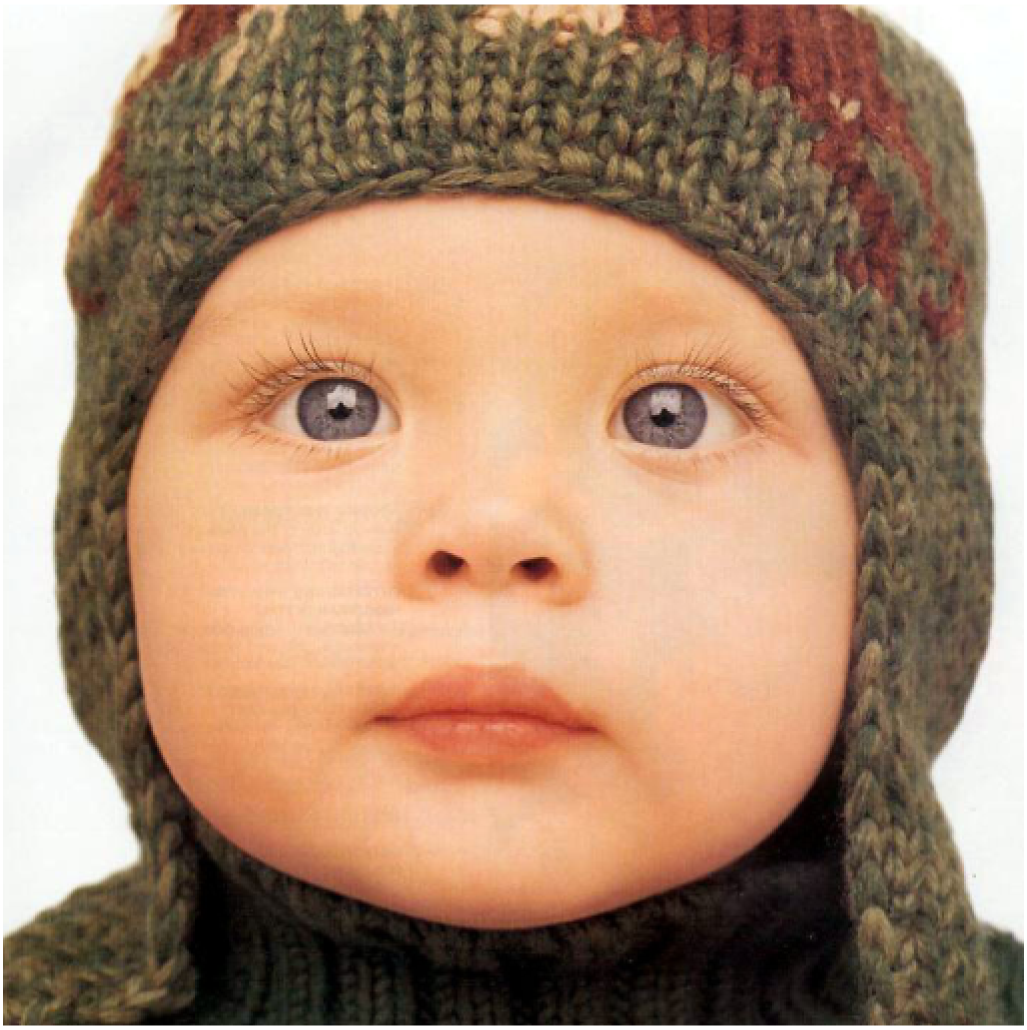
In [17]:

```
upscale_factor=3

total_bicubic_psnr = 0.0
total_test_psnr = 0.0
```

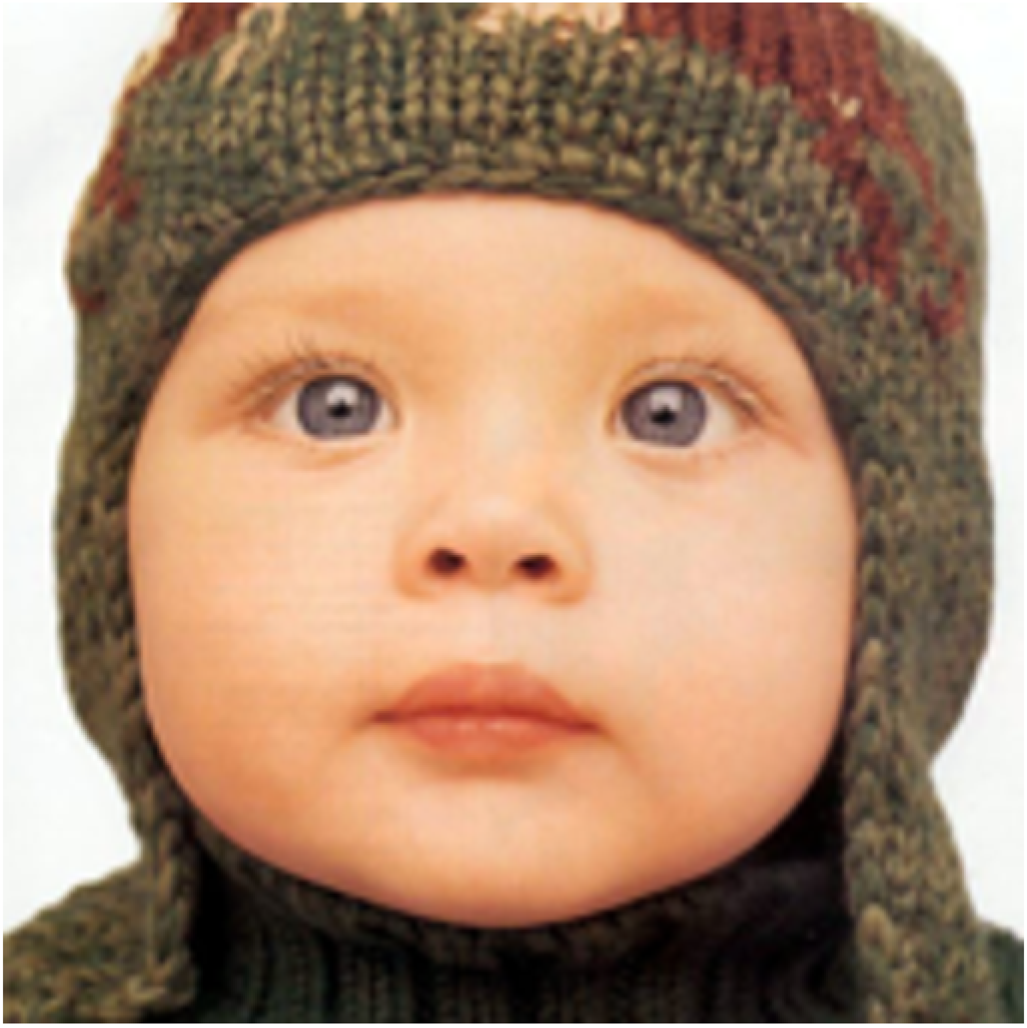
In [18]:

```
from tensorflow.keras.utils import load_img
from PIL import Image
import IPython.display as display
for n in range(1,6):
    s = str(n)
    path="test/"+s+".png"
    img = Image.open(path)
#     display.display(img)
    lowres_input = get_lowres_image(img, upscale_factor)
    w = lowres_input.size[0] * upscale_factor
    h = lowres_input.size[1] * upscale_factor
    highres_img = img.resize((w, h))
    prediction = upscale_image(model, lowres_input)
    lowres_img = lowres_input.resize((w, h))
    lowres_img_arr = img_to_array(lowres_img)
    highres_img_arr = img_to_array(highres_img)
    predict_img_arr = img_to_array(prediction)
    bicubic_psnr = tf.image.psnr(lowres_img_arr, highres_img_arr, max_val=255)
    test_psnr = tf.image.psnr(predict_img_arr, highres_img_arr, max_val=255)
    total_bicubic_psnr += bicubic_psnr
    total_test_psnr += test_psnr
    print(
    "PSNR of low resolution image and high resolution image is %.4f" % bicubic_psnr
    )
    print("PSNR of prediction and high resolution is %.4f" % test_psnr)
#     plot_results(lowres_img, 0, "lowres")
#     plot_results(highres_img, 0, "highres")
#     plot_results(prediction, 0, "prediction")
    print("highres_img")
    display.display(highres_img)

    print("lowres_img")
    display.display(lowres_img)

    print("prediction")
    display.display(prediction)

print("Avg. PSNR of lowres images is %.4f" % (total_bicubic_psnr / 5))
print("Avg. PSNR of reconstructions is %.4f" % (total_test_psnr / 5))
```

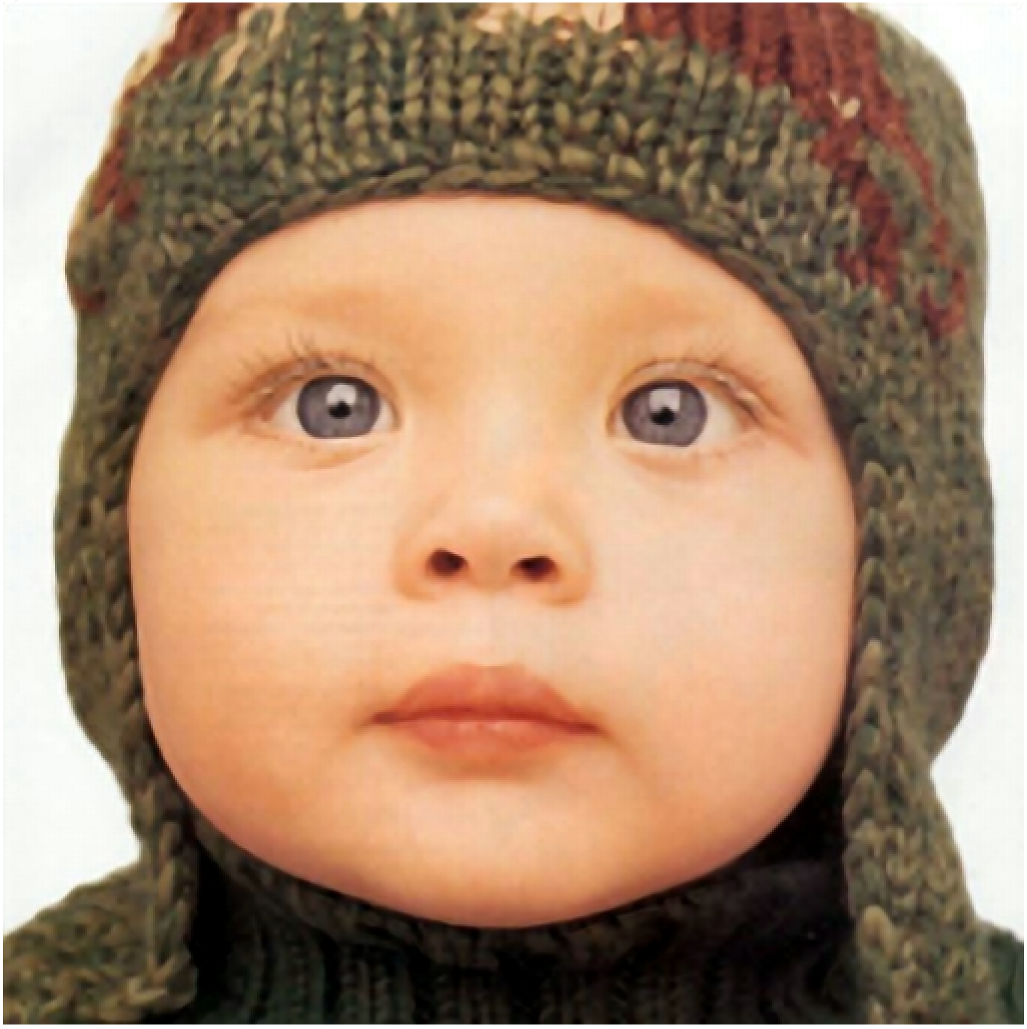
```
1/1 [==============================] - 0s 54ms/step
PSNR of low resolution image and high resolution image is 32.5055
PSNR of prediction and high resolution is 32.9955
highres_img
```



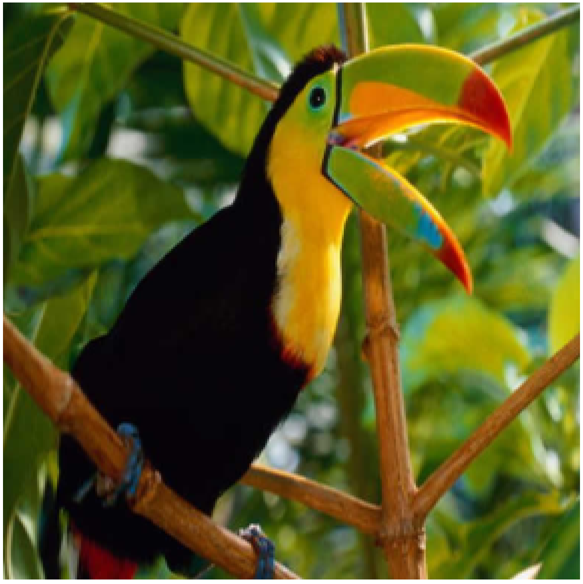
lowres_img



```
prediction
```



```
1/1 [==============================] - 0s 48ms/step
PSNR of low resolution image and high resolution image is 30.6093
PSNR of prediction and high resolution is 32.0511
highres_img
```



```
lowres_img
```

prediction

```
1/1 [==============================] - 0s 24ms/step
PSNR of low resolution image and high resolution image is 22.7739
PSNR of prediction and high resolution is 26.2099
highres_img
```

lowres_img

```
prediction
```

```
1/1 [==============================] - 0s 25ms/step
PSNR of low resolution image and high resolution image is 29.9746
PSNR of prediction and high resolution is 30.2754
highres_img
```

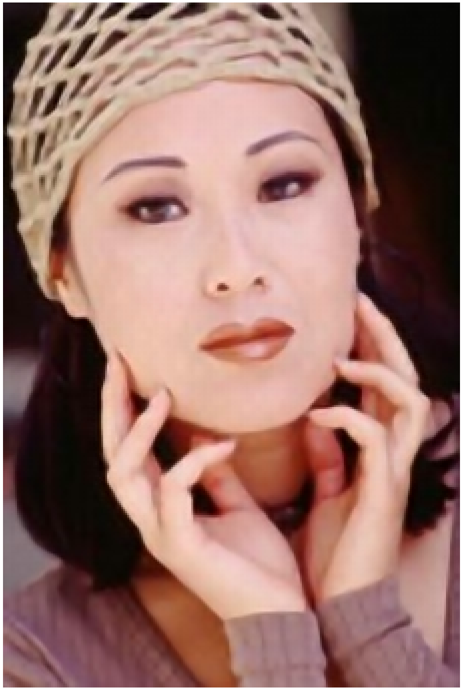
```
lowres_img
```

```
prediction
```

```
1/1 [==============================] - 0s 28ms/step
PSNR of low resolution image and high resolution image is 27.2204
PSNR of prediction and high resolution is 29.4972
highres_img
```

```
lowres_img
```

```
prediction
```



```
Avg. PSNR of lowres images is 28.6167
Avg. PSNR of reconstructions is 30.2058
```

In [50]:

```
# lowres_input = get_lowres_image(img, upscale_factor)
# w = lowres_input.size[0] * upscale_factor
# h = lowres_input.size[1] * upscale_factor
# highres_img = img.resize((w, h))
# prediction = upscale_image(model, lowres_input)
# lowres_img = lowres_input.resize((w, h))
# lowres_img_arr = img_to_array(lowres_img)
# highres_img_arr = img_to_array(highres_img)
# predict_img_arr = img_to_array(prediction)
# bicubic_psnr = tf.image.psnr(lowres_img_arr, highres_img_arr, max_val=255)
# test_psnr = tf.image.psnr(predict_img_arr, highres_img_arr, max_val=255)
# total_bicubic_psnr += bicubic_psnr
# total_test_psnr += test_psnr
# print(
# "PSNR of low resolution image and high resolution image is %.4f" % bicubic_psnr
# )
# print("PSNR of prediction and high resolution is %.4f" % test_psnr)
# plot_results(lowres_img, 0, "lowres")
# plot_results(highres_img, 0, "highres")
# plot_results(prediction, 0, "prediction")
# print("highres_img")
# display.display(highres_img)

# print("lowres_img")
# display.display(lowres_img)

# print("prediction")
# display.display(prediction)

# # highres_img.save("highres_img.jpg")
# # lowres_img.save("lowres_img.jpg")
# # prediction.save("prediction.jpg")
```

In [ ]:

```
```