In [1]:
```
import tensorflow as tf
```

In [2]:
```
import os
import math
import numpy as np
```

In [3]:
```
from tensorflow import keras
from tensorflow.keras import layers
```

In [4]:
```
from tensorflow.keras.utils import load_img
```

In [5]:
```
from tensorflow.keras.utils import array_to_img
```

In [6]:
```
from tensorflow.keras.utils import img_to_array
```

In [7]:
```
from tensorflow.keras.utils import image_dataset_from_directory
```

In [8]:
```
from IPython.display import display
```

In [9]:
```
root_dir = os.path.join("./data")
```

In [10]:
```
crop_size = 300
upscale_factor = 3
input_size = crop_size // upscale_factor
batch_size = 4

train_ds = image_dataset_from_directory(
    root_dir,
    batch_size=batch_size,
    image_size=(crop_size, crop_size),
    validation_split=0.2,
    subset="training",
    seed=1337,
    label_mode=None,
)

valid_ds = image_dataset_from_directory(
    root_dir,
    batch_size=batch_size,
    image_size=(crop_size, crop_size),
    validation_split=0.2,
    subset="validation",
    seed=1337,
    label_mode=None,
)
```

```
Found 17186 files belonging to 1 classes.
Using 13749 files for training.
Found 17186 files belonging to 1 classes.
Using 3437 files for validation.
```

In [11]:
```
def scaling(input_image):
    input_image = input_image / 255.0
    return input_image


# Scale from (0, 255) to (0, 1)
train_ds = train_ds.map(scaling)
valid_ds = valid_ds.map(scaling)
```

In [12]:
```
# import pillow
# import PIL.Image
import sys
from PIL import Image
sys.modules['Image'] = Image
from PIL import Image
print(Image.__file__)
import Image
print(Image.__file__)
```

```
C:\Users\anura\anaconda3\lib\site-packages\PIL\Image.py
C:\Users\anura\anaconda3\lib\site-packages\PIL\Image.py
```

In [13]:
```
for batch in train_ds.take(1):
    for img in batch:
        display(array_to_img(img))
```

In [14]:

```
dataset = os.path.join(root_dir, "images")
test_path = os.path.join(dataset, "test")

test_img_paths = sorted(
    [
        os.path.join(test_path, fname)
```

```
        for fname in os.listdir(test_path)
        if fname.endswith(".jpg")
    ]
)
```

In [15]:

```
# Use TF Ops to process.
def process_input(input, input_size, upscale_factor):
    input = tf.image.rgb_to_yuv(input)
    last_dimension_axis = len(input.shape) - 1
    y, u, v = tf.split(input, 3, axis=last_dimension_axis)
    return tf.image.resize(y, [input_size, input_size], method="area")


def process_target(input):
    input = tf.image.rgb_to_yuv(input)
    last_dimension_axis = len(input.shape) - 1
    y, u, v = tf.split(input, 3, axis=last_dimension_axis)
    return y


train_ds = train_ds.map(
    lambda x: (process_input(x, input_size, upscale_factor), process_target(x))
)

valid_ds = valid_ds.map(
    lambda x: (process_input(x, input_size, upscale_factor), process_target(x))
)
```

In [16]:

```
for batch in train_ds.take(1):
    for img in batch[0]:
        display(array_to_img(img))
    for img in batch[1]:
        display(array_to_img(img))
```

In [17]:

```
def get_model(upscale_factor=3, channels=1):
    conv_args = {
        "activation": "relu",
        "kernel_initializer": "Orthogonal",
        "padding": "same",
    }
    inputs = keras.Input(shape=(None, None, channels))
    x = layers.Conv2D(64, 5, **conv_args)(inputs)
    x = layers.Conv2D(64, 3, **conv_args)(x)
    x = layers.Conv2D(32, 3, **conv_args)(x)
    x = layers.Conv2D(channels * (upscale_factor ** 2), 3, **conv_args)(x)
    outputs = tf.nn.depth_to_space(x, upscale_factor)

    return keras.Model(inputs, outputs)
```

In [18]:

```
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1.inset_locator import zoomed_inset_axes
from mpl_toolkits.axes_grid1.inset_locator import mark_inset
import PIL

def plot_results(img, prefix, title):
    """Plot the result with zoom-in area."""
    img_array = img_to_array(img)
    img_array = img_array.astype("float32") / 255.0

    # Create a new figure with a default 111 subplot.
    fig, ax = plt.subplots()
    im = ax.imshow(img_array[::-1], origin="lower")

    plt.title(title)
    # zoom-factor: 2.0, location: upper-left
    axins = zoomed_inset_axes(ax, 2, loc=2)
    axins.imshow(img_array[::-1], origin="lower")

    # Specify the limits.
    x1, x2, y1, y2 = 200, 300, 100, 200
    # Apply the x-limits.
    axins.set_xlim(x1, x2)
    # Apply the y-limits.
    axins.set_ylim(y1, y2)

    plt.yticks(visible=False)
```

```
    plt.xticks(visible=False)

    # Make the line.
    mark_inset(ax, axins, loc1=1, loc2=3, fc="none", ec="blue")
    plt.savefig(str(prefix) + "-" + title + ".png")
    plt.show()

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,
    )


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 [19]:

```
class ESPCNCallback(keras.callbacks.Callback):
    def __init__(self):
        super(ESPCNCallback, self).__init__()
        self.test_img = get_lowres_image(load_img(test_img_paths[0]), upscale_factor

    # Store PSNR value in each epoch.
    def on_epoch_begin(self, epoch, logs=None):
        self.psnr = []

    def on_epoch_end(self, epoch, logs=None):
        print("Mean PSNR for epoch: %.2f" % (np.mean(self.psnr)))
        prediction = upscale_image(self.model, self.test_img)
        plot_results(prediction, "epoch-" + str(epoch), "prediction")

    def on_test_batch_end(self, batch, logs=None):
        self.psnr.append(10 * math.log10(1 / logs["loss"]))
```

In [20]:

```
early_stopping_callback = keras.callbacks.EarlyStopping(monitor="loss", patience=10)

checkpoint_filepath = "./tmp/checkpoint"

model_checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
```

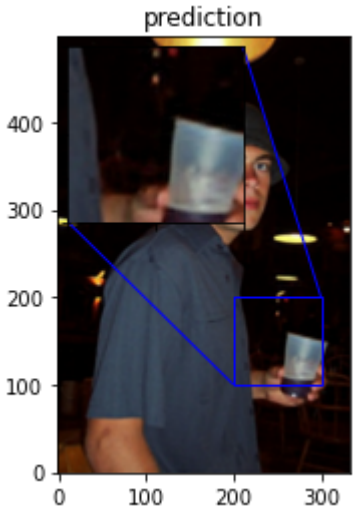
```
    monitor="loss",
    mode="min",
    save_best_only=True,
)

model = get_model(upscale_factor=upscale_factor, channels=1)
model.summary()

callbacks = [ESPCNCallback(), early_stopping_callback, model_checkpoint_callback]
loss_fn = keras.losses.MeanSquaredError()
optimizer = keras.optimizers.Adam(learning_rate=0.001)
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None, None, 1)]   0

 conv2d (Conv2D)             (None, None, None, 64)    1664

 conv2d_1 (Conv2D)           (None, None, None, 64)    36928

 conv2d_2 (Conv2D)           (None, None, None, 32)    18464

 conv2d_3 (Conv2D)           (None, None, None, 9)     2601

 tf.nn.depth_to_space (TFOpL  (None, None, None, 1)    0
 ambda)

=================================================================
Total params: 59,657
Trainable params: 59,657
Non-trainable params: 0
_________________________________________________________________
```

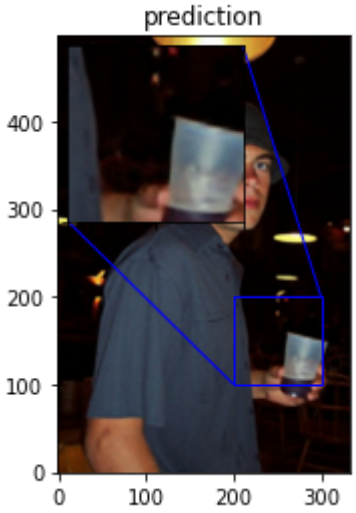
In [21]:

```
# epochs = 10
epochs=50

model.compile(
    optimizer=optimizer, loss=loss_fn,
)

model.fit(
    train_ds, epochs=epochs, callbacks=callbacks, validation_data=valid_ds, verbose=
)

# The model weights (that are considered the best) are loaded into the model.
model.load_weights(checkpoint_filepath)


model.save('./model_final')
```

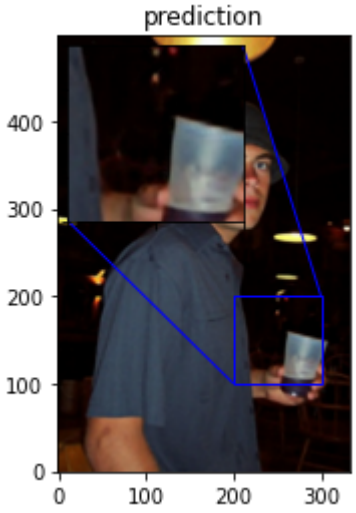
```
Epoch 1/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0037Mean PSNR for epo
ch: 25.35
```



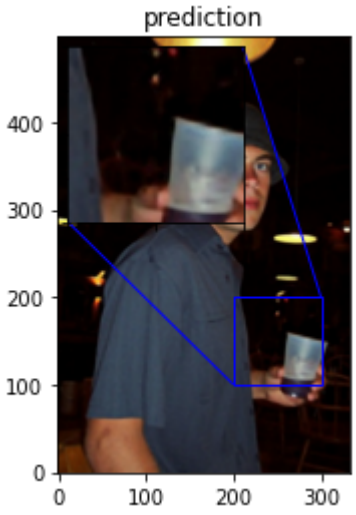

```
3438/3438 [==============================] - 695s 202ms/step - loss: 0.0037 - val_lo
ss: 0.0030
Epoch 2/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0029Mean PSNR for epo
ch: 25.48
```



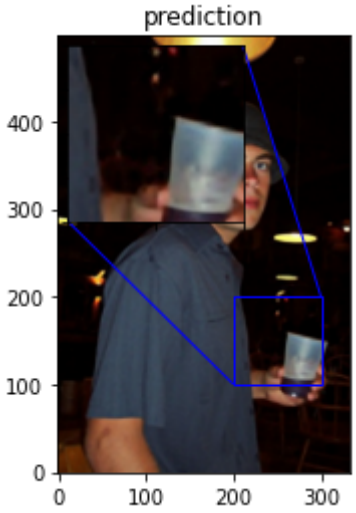

```
3438/3438 [==============================] - 703s 205ms/step - loss: 0.0029 - val_lo
ss: 0.0029
Epoch 3/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0029Mean PSNR for epo
ch: 25.54
```



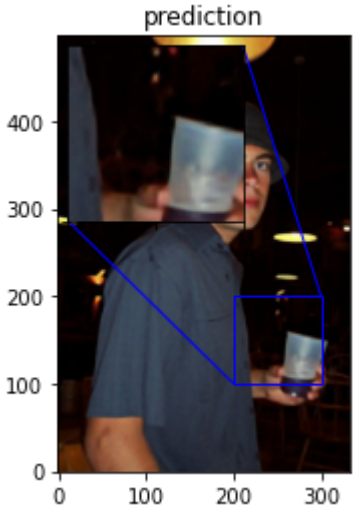

```
3438/3438 [==============================] - 705s 205ms/step - loss: 0.0029 - val_lo
ss: 0.0029
Epoch 4/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0028Mean PSNR for epo
ch: 25.55
```



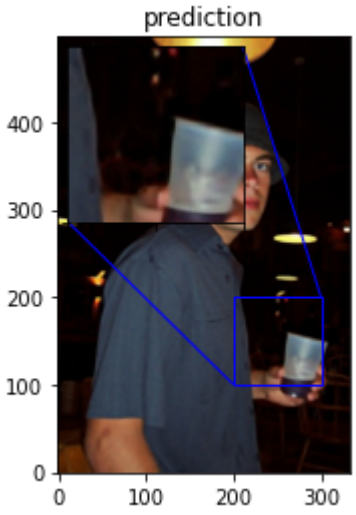

```
3438/3438 [==============================] - 706s 205ms/step - loss: 0.0028 - val_lo
ss: 0.0029
Epoch 5/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0028Mean PSNR for epo
ch: 25.58
```



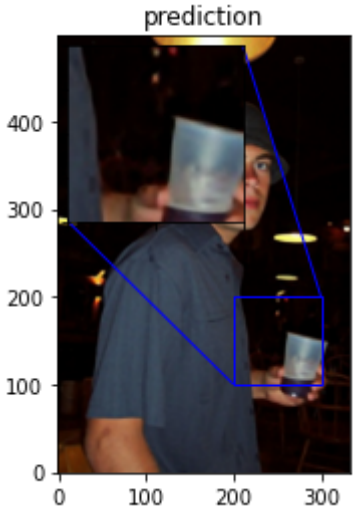

```
3438/3438 [==============================] - 704s 205ms/step - loss: 0.0028 - val_lo
ss: 0.0028
Epoch 6/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0028Mean PSNR for epo
ch: 25.61
```



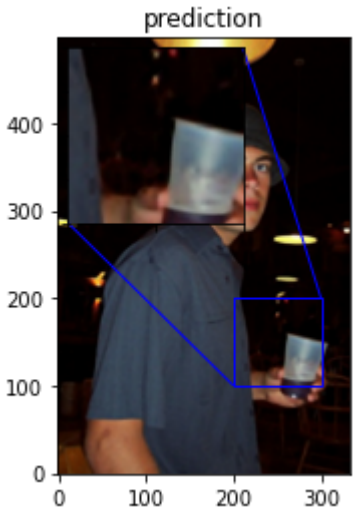

```
3438/3438 [==============================] - 704s 205ms/step - loss: 0.0028 - val_lo
ss: 0.0028
Epoch 7/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0028Mean PSNR for epo
ch: 25.64
```



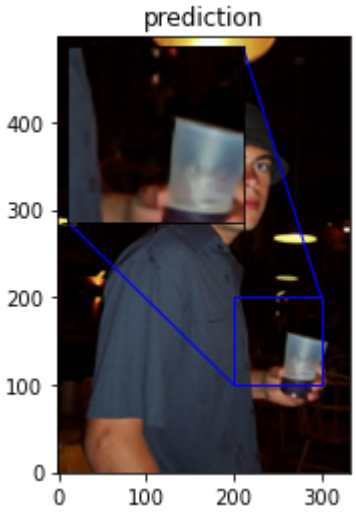

```
3438/3438 [==============================] - 704s 205ms/step - loss: 0.0028 - val_lo
ss: 0.0028
Epoch 8/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0028Mean PSNR for epo
ch: 25.65
```



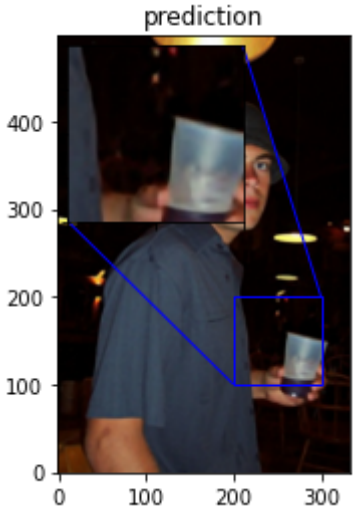

```
3438/3438 [==============================] - 704s 205ms/step - loss: 0.0028 - val_lo
ss: 0.0028
Epoch 9/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0028Mean PSNR for epo
ch: 25.62
```



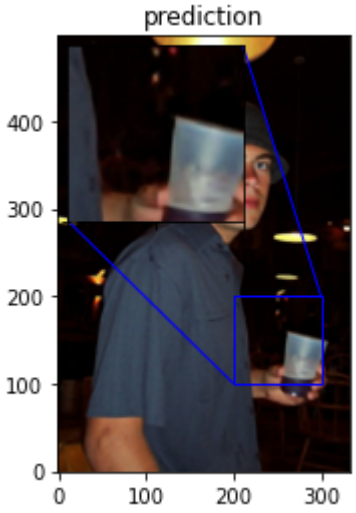

```
3438/3438 [==============================] - 704s 205ms/step - loss: 0.0028 - val_lo
ss: 0.0028
Epoch 10/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.68
```



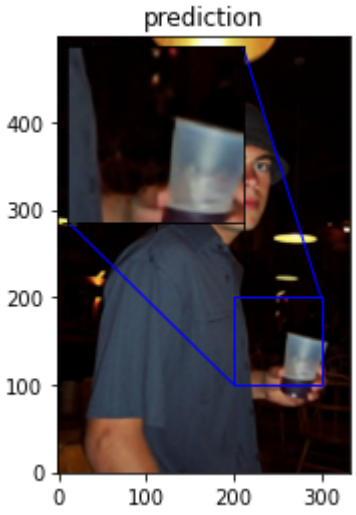

```
3438/3438 [==============================] - 705s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0028
Epoch 11/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.70
```



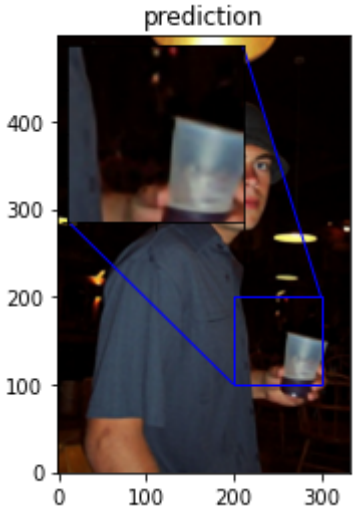

```
3438/3438 [==============================] - 714s 208ms/step - loss: 0.0027 - val_lo
ss: 0.0028
Epoch 12/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.69
```



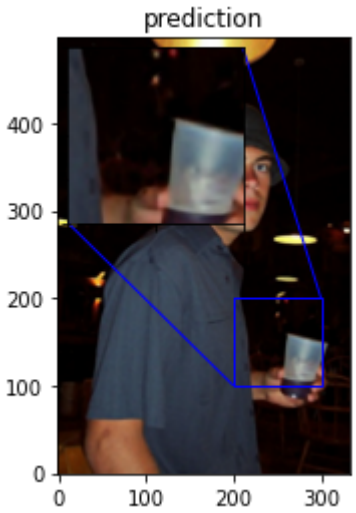

```
3438/3438 [==============================] - 704s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0028
Epoch 13/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.72
```



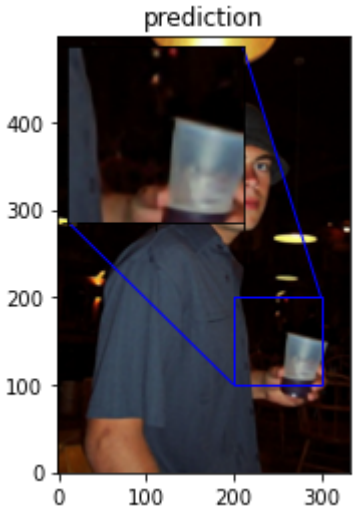

```
3438/3438 [==============================] - 707s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0028
Epoch 14/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.70
```



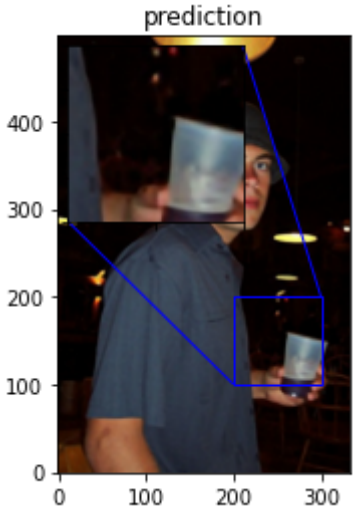

```
3438/3438 [==============================] - 708s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0028
Epoch 15/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.72
```



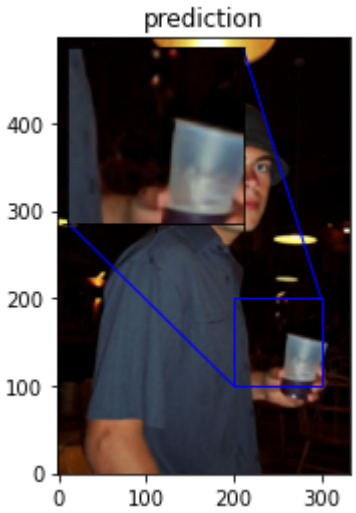

```
3438/3438 [==============================] - 710s 207ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 16/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.73
```



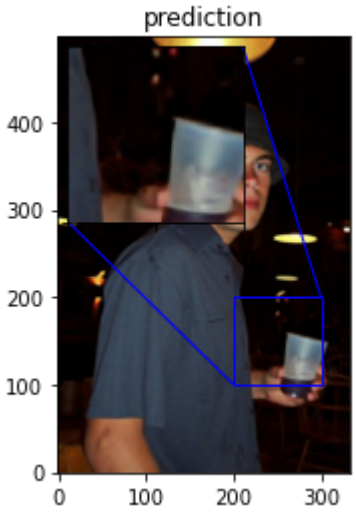

```
3438/3438 [==============================] - 708s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 17/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.73
```



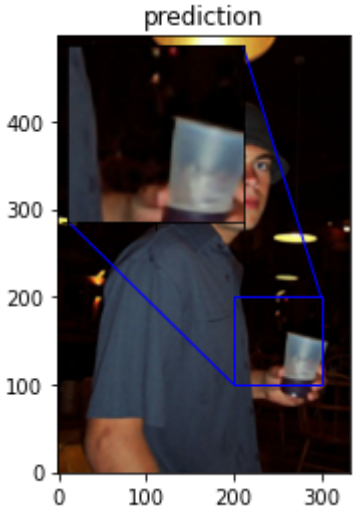

```
3438/3438 [==============================] - 705s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 18/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.74
```



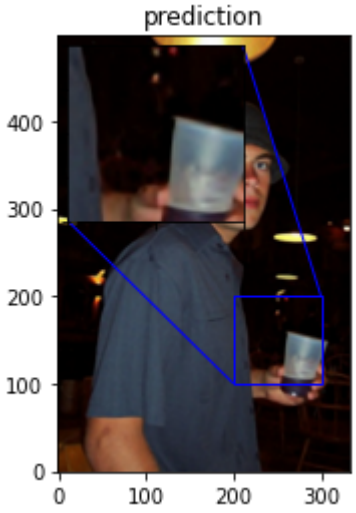

```
3438/3438 [==============================] - 711s 207ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 19/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.73
```



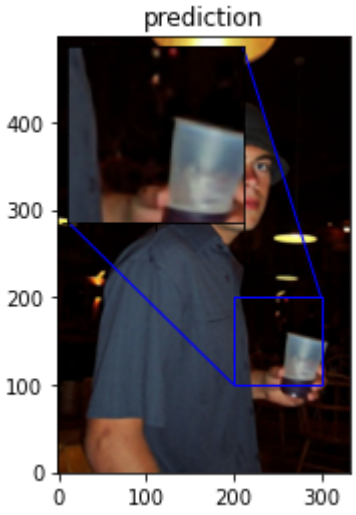

```
3438/3438 [==============================] - 742s 216ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 20/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.74
```



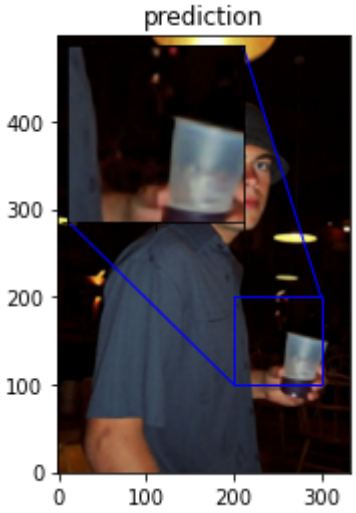

```
3438/3438 [==============================] - 708s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 21/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.72
```



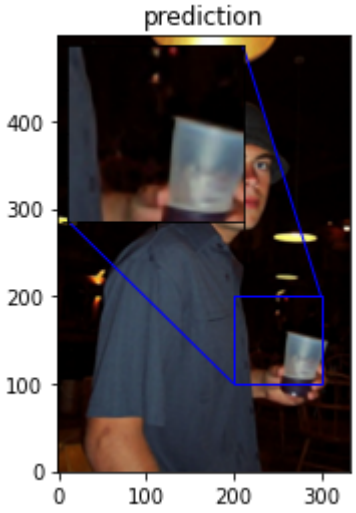

```
3438/3438 [==============================] - 718s 209ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 22/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.76
```



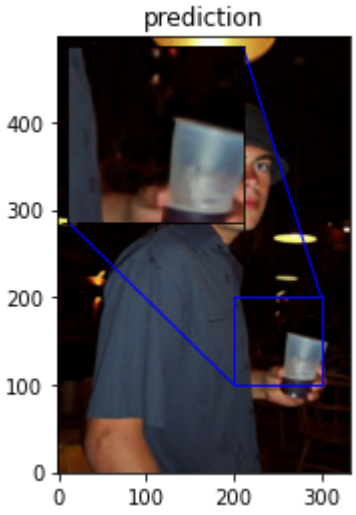

```
3438/3438 [==============================] - 709s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 23/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.73
```



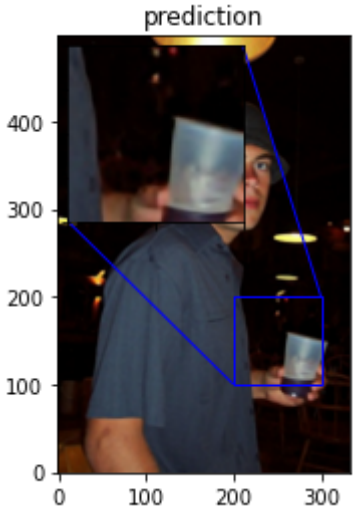

```
3438/3438 [==============================] - 728s 212ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 24/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.76
```



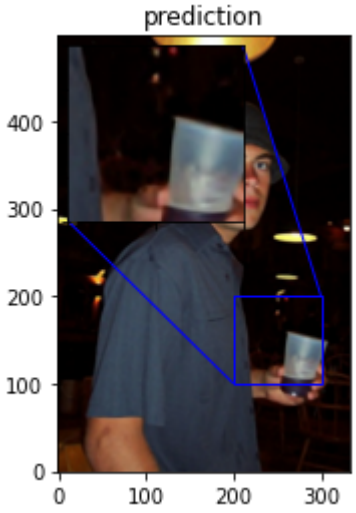

```
3438/3438 [==============================] - 707s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 25/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.74
```



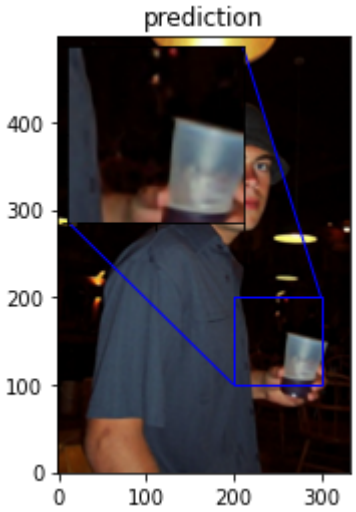

```
3438/3438 [==============================] - 708s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 26/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.76
```



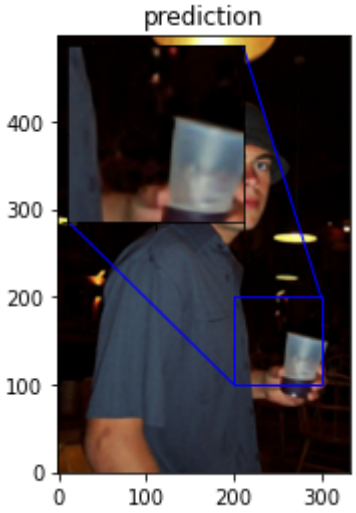

```
3438/3438 [==============================] - 706s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 27/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.77
```



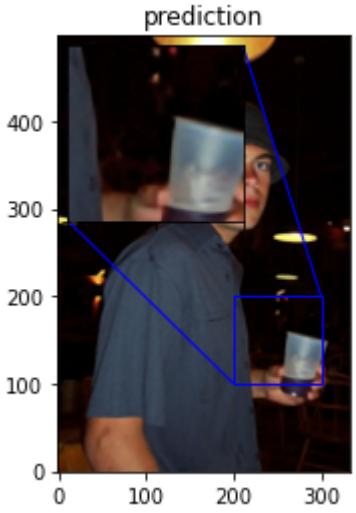

```
3438/3438 [==============================] - 705s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 28/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.77
```



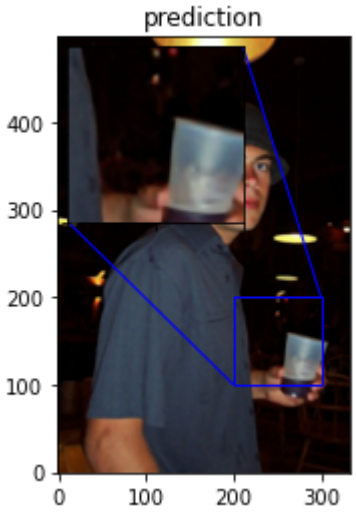

```
3438/3438 [==============================] - 705s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 29/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.76
```



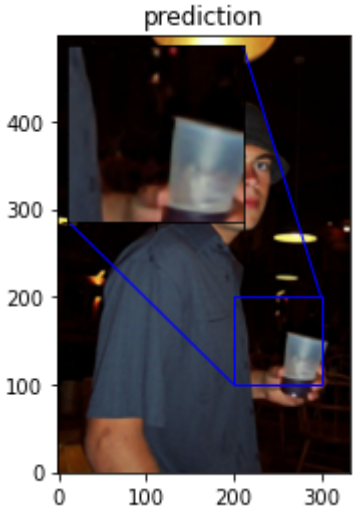

```
3438/3438 [==============================] - 712s 207ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 30/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.73
```



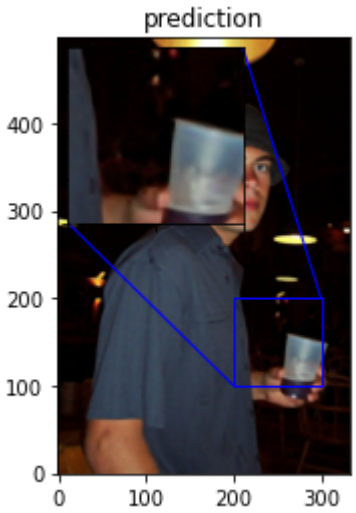

```
3438/3438 [==============================] - 713s 207ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 31/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.75
```



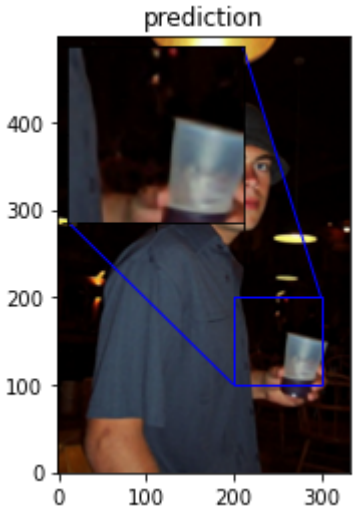

```
3438/3438 [==============================] - 706s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 32/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.76
```



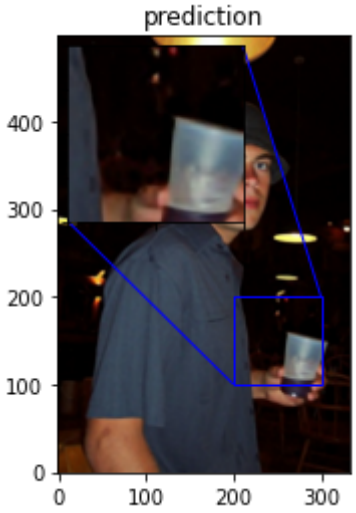

```
3438/3438 [==============================] - 707s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 33/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.78
```



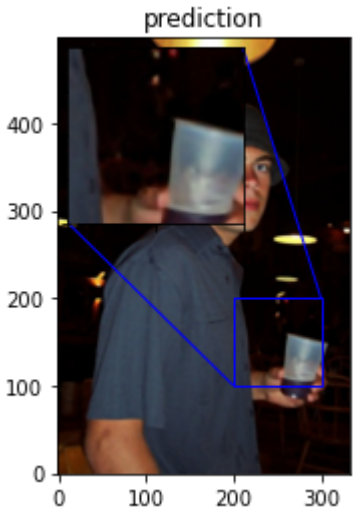

```
3438/3438 [==============================] - 720s 209ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 34/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.75
```



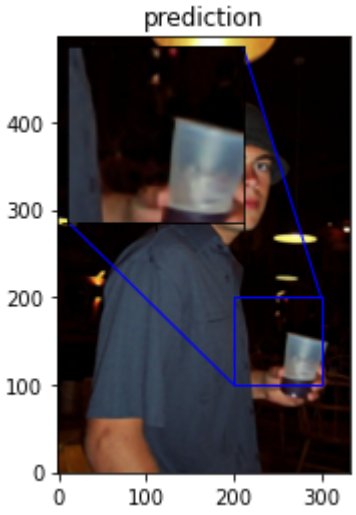

```
3438/3438 [==============================] - 705s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 35/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.76
```



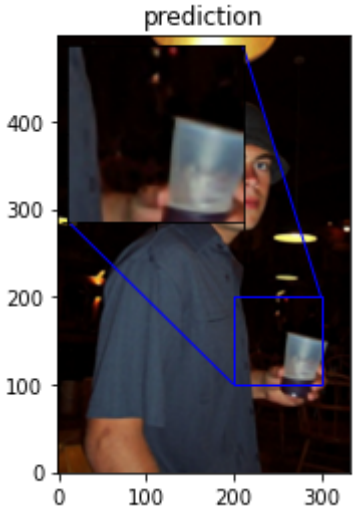

```
3438/3438 [==============================] - 707s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 36/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.78
```



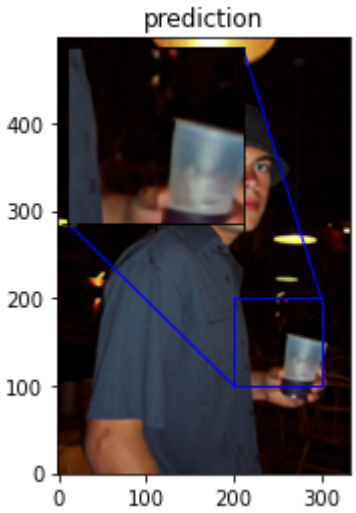

```
3438/3438 [==============================] - 706s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 37/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.77
```



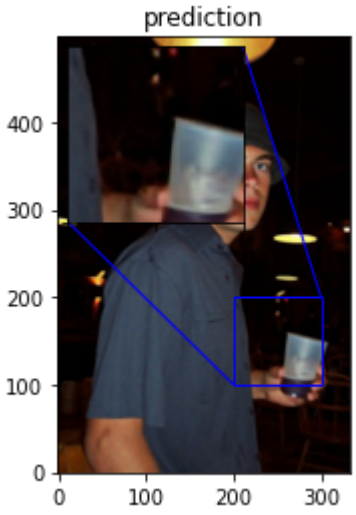

```
3438/3438 [==============================] - 706s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 38/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.77
```



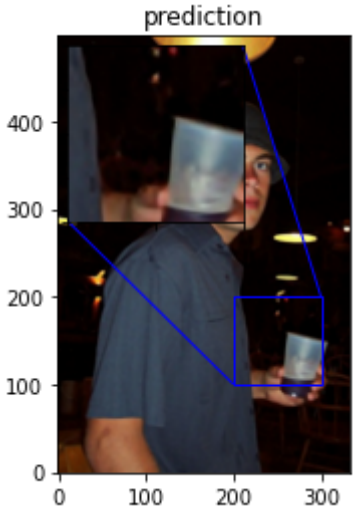

```
3438/3438 [==============================] - 706s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 39/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.75
```



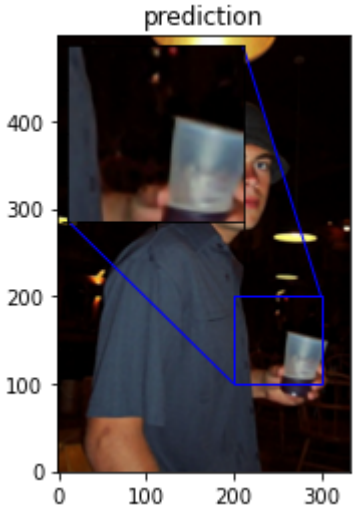

```
3438/3438 [==============================] - 714s 208ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 40/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.77
```



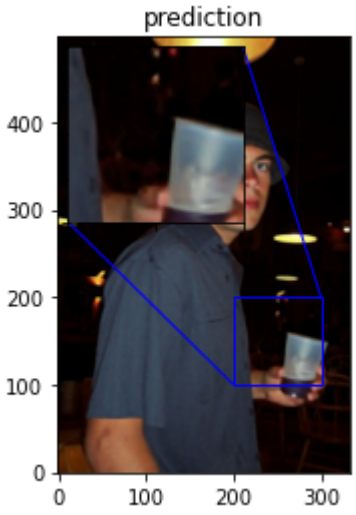

```
3438/3438 [==============================] - 707s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 41/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.77
```



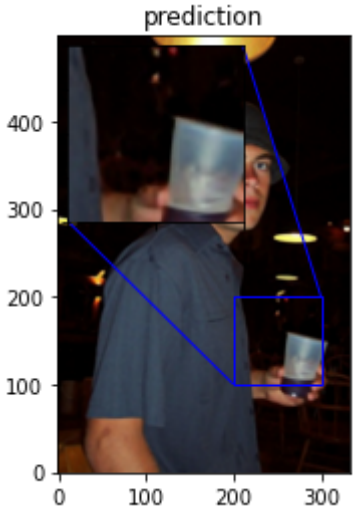

```
3438/3438 [==============================] - 720s 210ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 42/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.77
```



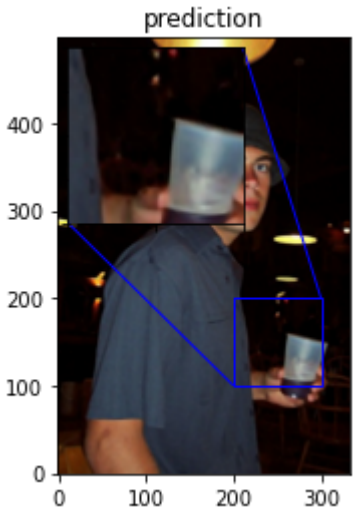

```
3438/3438 [==============================] - 713s 207ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 43/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.79
```



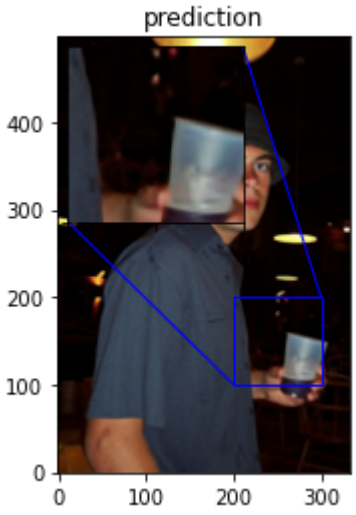

```
3438/3438 [==============================] - 706s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 44/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.78
```



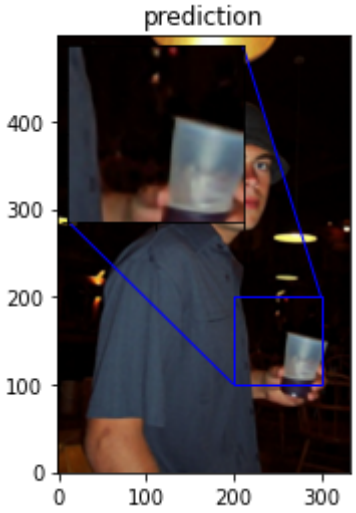

```
3438/3438 [==============================] - 707s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 45/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.76
```



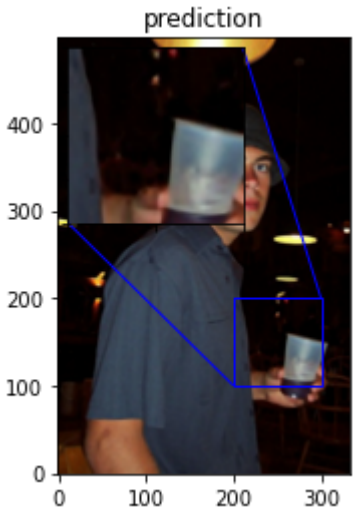

```
3438/3438 [==============================] - 709s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 46/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0026Mean PSNR for epo
ch: 25.78
```



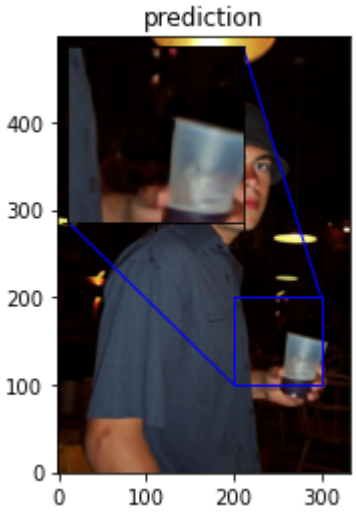

```
3438/3438 [==============================] - 706s 205ms/step - loss: 0.0026 - val_lo
ss: 0.0027
Epoch 47/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.77
```



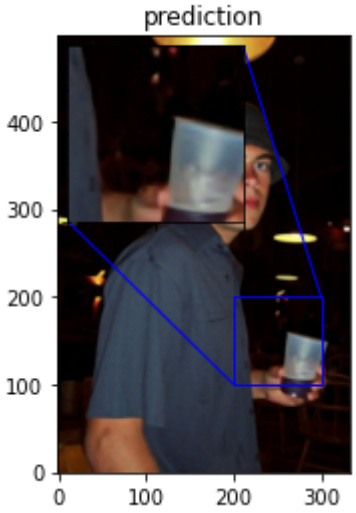

```
3438/3438 [==============================] - 706s 205ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 48/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.77
```




```
3438/3438 [==============================] - 707s 206ms/step - loss: 0.0027 - val_lo
ss: 0.0027
Epoch 49/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0026Mean PSNR for epo
ch: 25.79
```




```
3438/3438 [==============================] - 707s 206ms/step - loss: 0.0026 - val_lo
ss: 0.0027
Epoch 50/50
3438/3438 [==============================] - ETA: 0s - loss: 0.0027Mean PSNR for epo
ch: 25.78
```




```
3438/3438 [==============================] - 713s 207ms/step - loss: 0.0027 - val_lo
ss: 0.0027
INFO:tensorflow:Assets written to: ./model_final\assets
```

In [ ]:

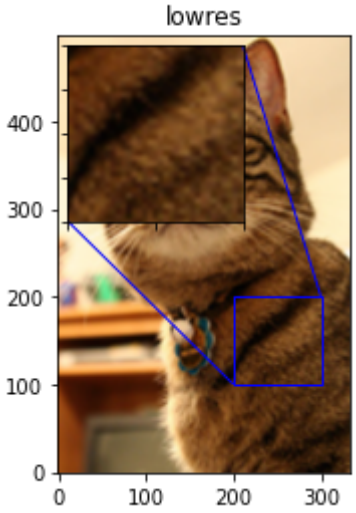
```
print("hello")
```

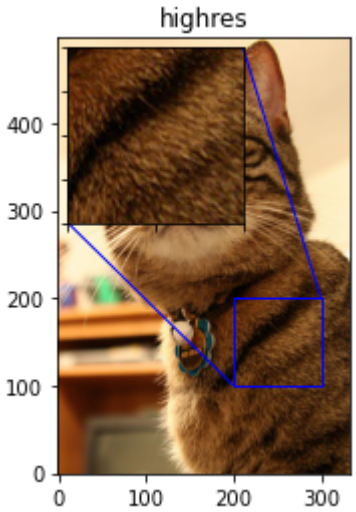
In [80]:

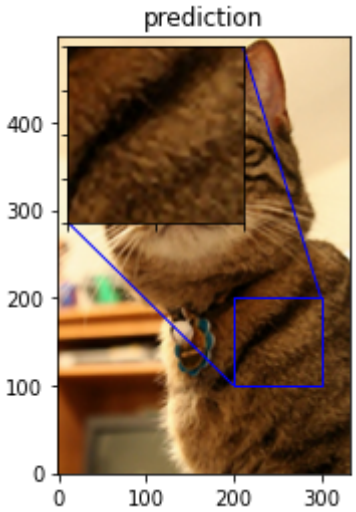
```
total_bicubic_psnr = 0.0
total_test_psnr = 0.0

for index, test_img_path in enumerate(test_img_paths[50:60]):
    img = load_img(test_img_path)
    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_p
    )
    print("PSNR of predict and high resolution is %.4f" % test_psnr)
    plot_results(lowres_img, index, "lowres")
    plot_results(highres_img, index, "highres")
    plot_results(prediction, index, "prediction")

print("Avg. PSNR of lowres images is %.4f" % (total_bicubic_psnr / 10))
print("Avg. PSNR of reconstructions is %.4f" % (total_test_psnr / 10))
```

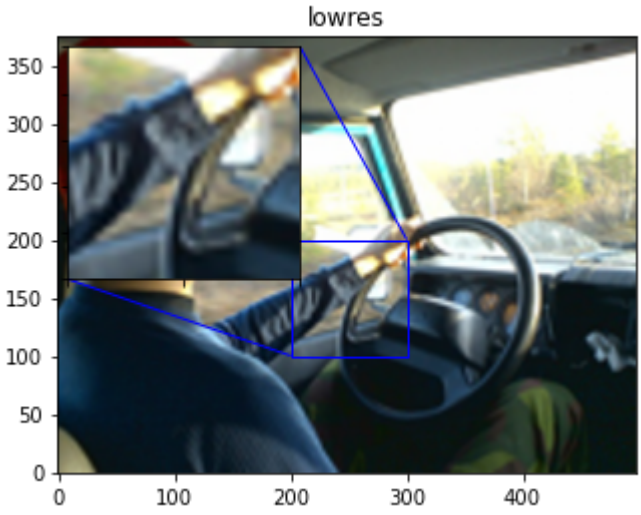
```
PSNR of low resolution image and high resolution image is 26.7161
PSNR of predict and high resolution is 27.1135
```

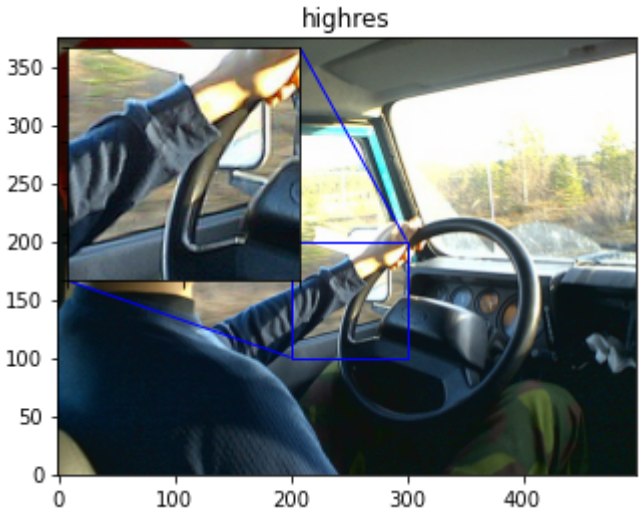


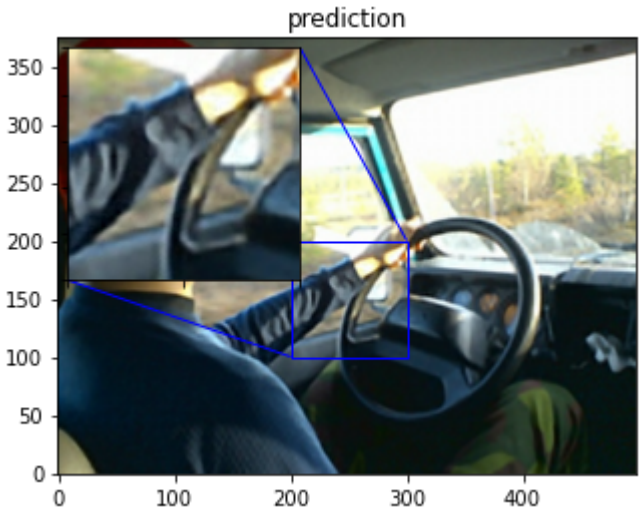




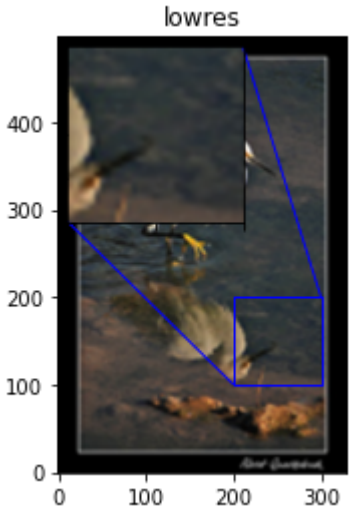
```
PSNR of low resolution image and high resolution image is 26.4780
PSNR of predict and high resolution is 27.6336
```

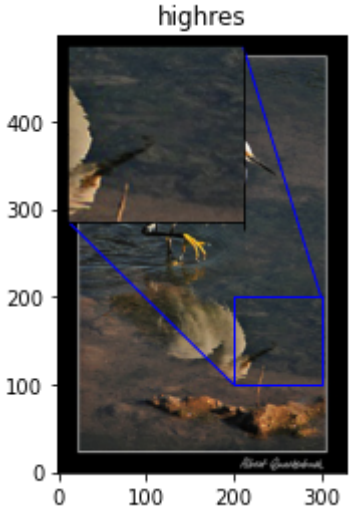


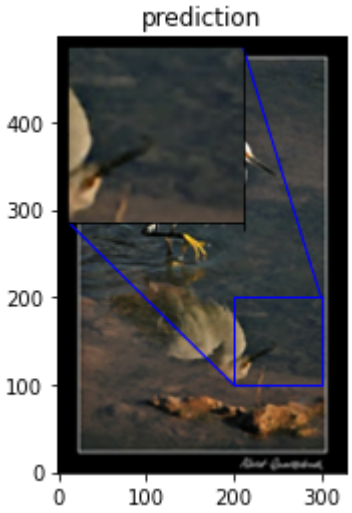





```
PSNR of low resolution image and high resolution image is 22.4373
PSNR of predict and high resolution is 23.4782
```

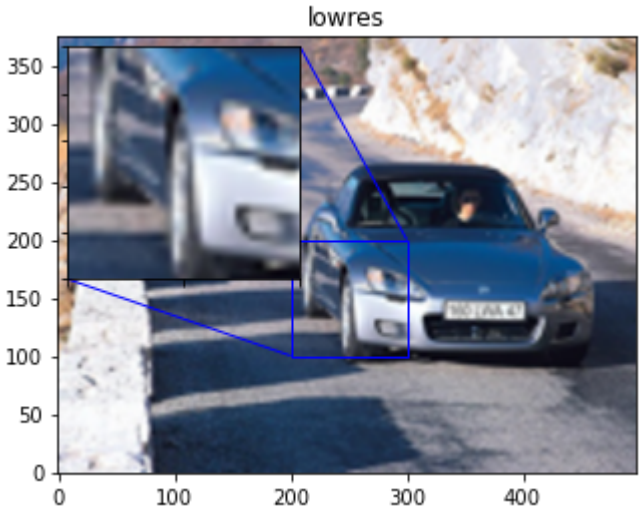


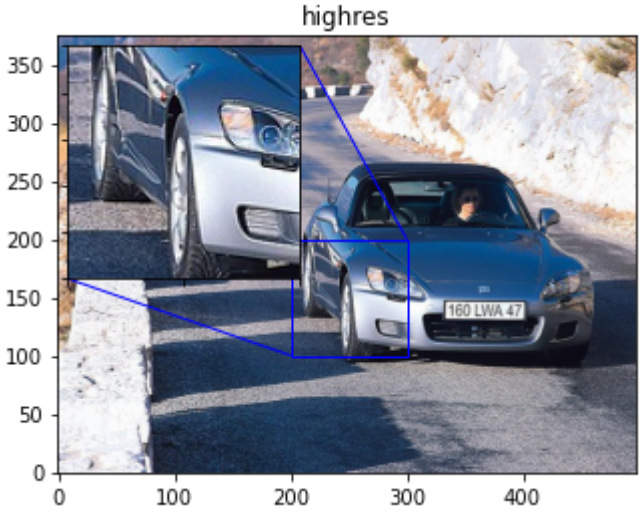


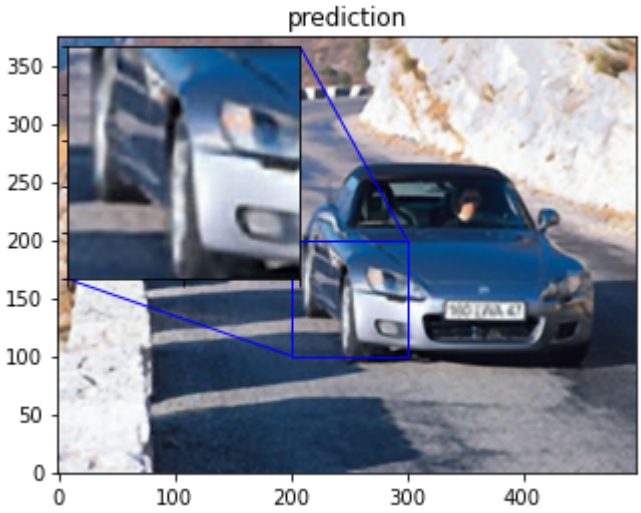



```
PSNR of low resolution image and high resolution image is 22.4946
PSNR of predict and high resolution is 22.8259
```

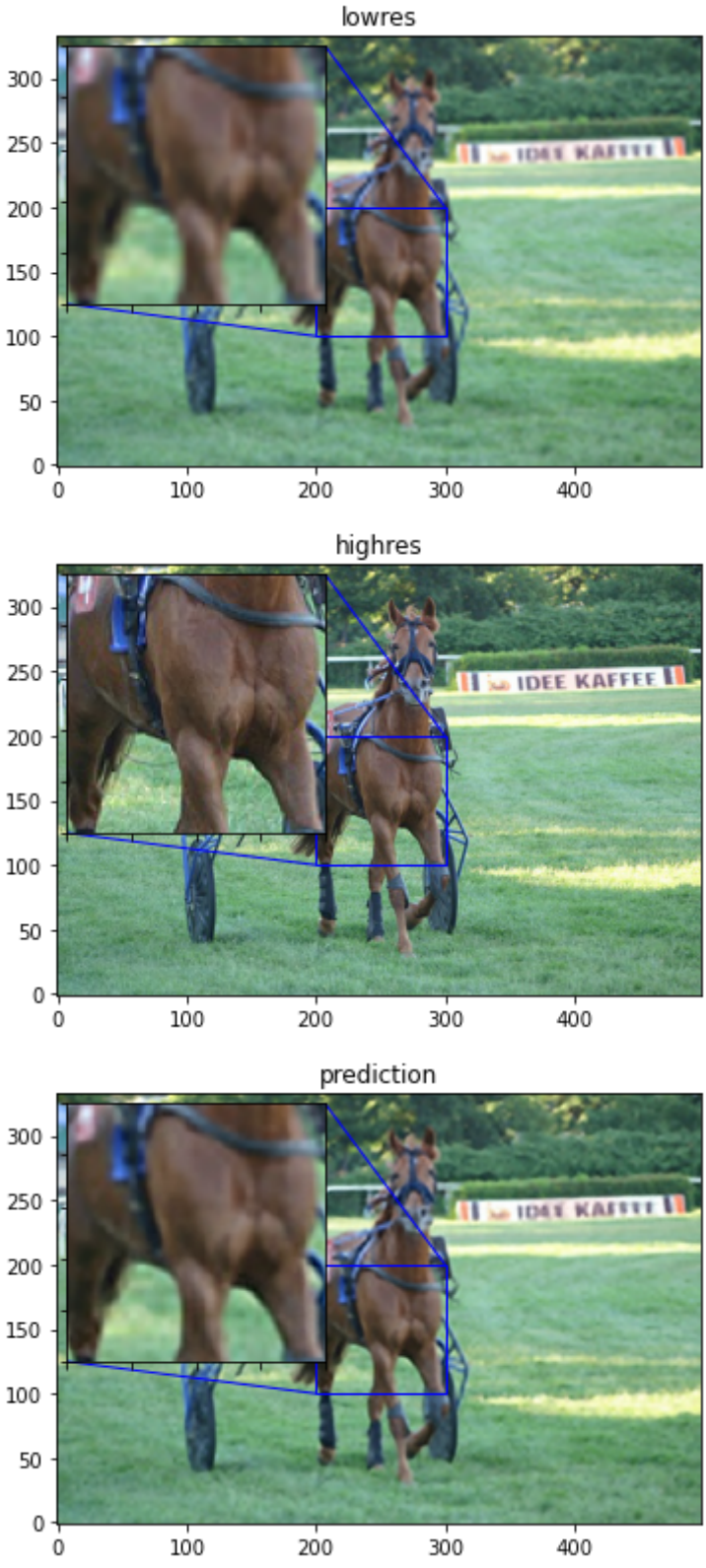

```
PSNR of low resolution image and high resolution image is 21.8302
PSNR of predict and high resolution is 22.4925
```



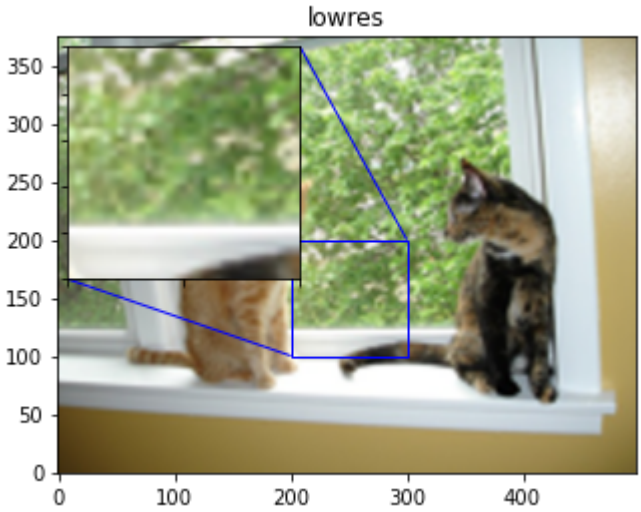


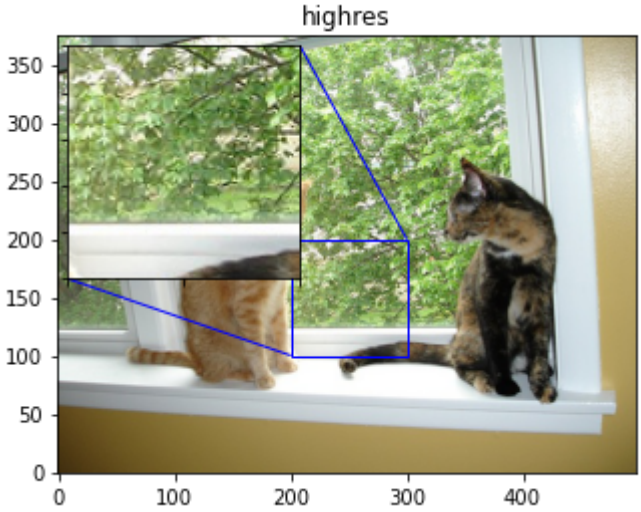


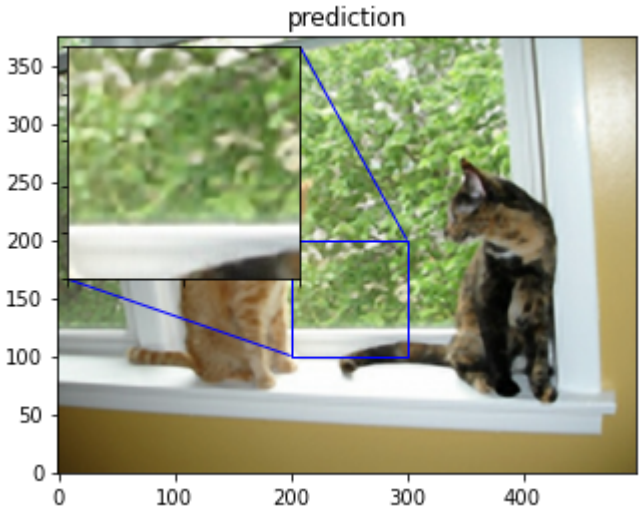

```
WARNING:tensorflow:5 out of the last 15 calls to <function Model.make_predict_functi
on.<locals>.predict_function at 0x000002090351D0D0> triggered tf.function retracing.
Tracing is expensive and the excessive number of tracings could be due to (1) creati
ng @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3)
passing Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option
that relaxes argument shapes that can avoid unnecessary retracing. For (3), please r
efer to https://www.tensorflow.org/guide/function#controlling_retracing and https://
www.tensorflow.org/api_docs/python/tf/function for  more details.
PSNR of low resolution image and high resolution image is 24.2919
PSNR of predict and high resolution is 25.0663
```



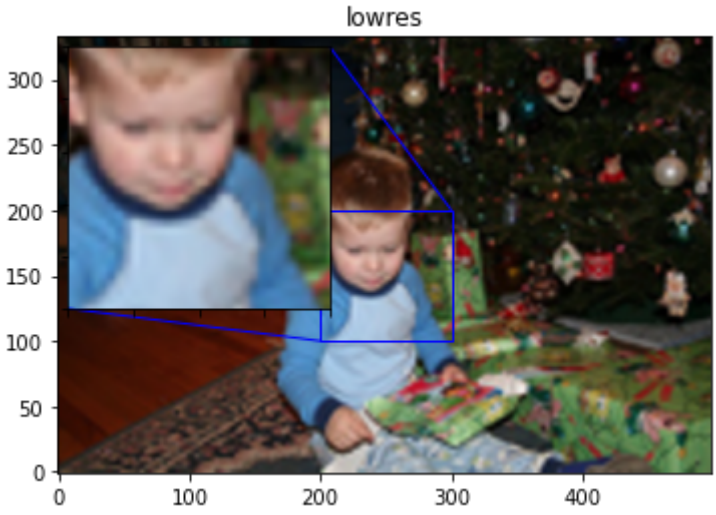

```
PSNR of low resolution image and high resolution image is 23.2296
PSNR of predict and high resolution is 23.6755
```

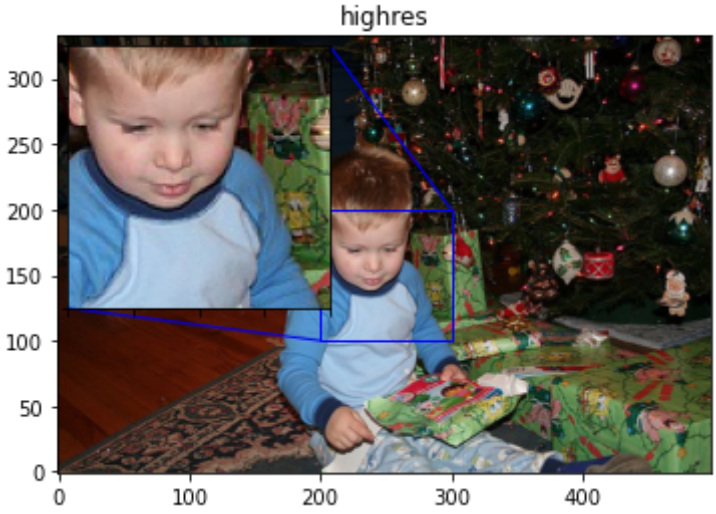


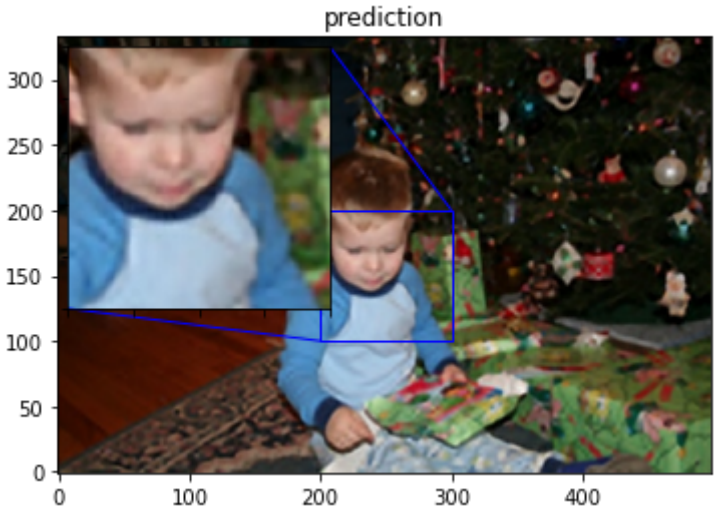




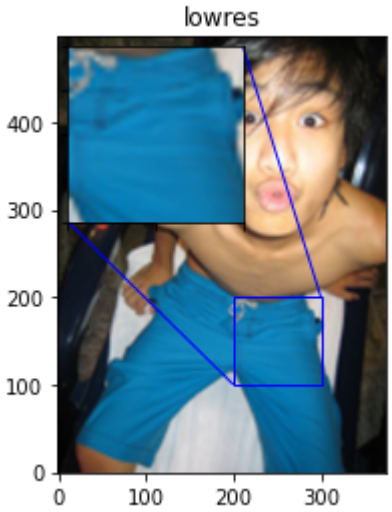

```
PSNR of low resolution image and high resolution image is 24.3504
PSNR of predict and high resolution is 24.8538
```

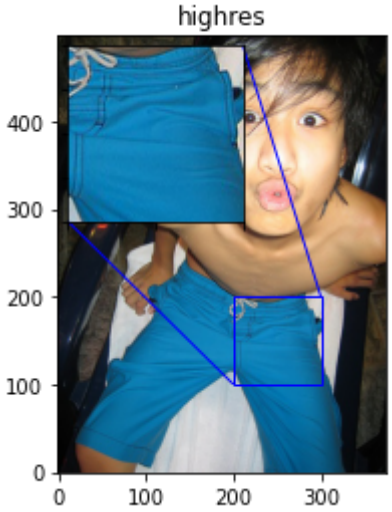


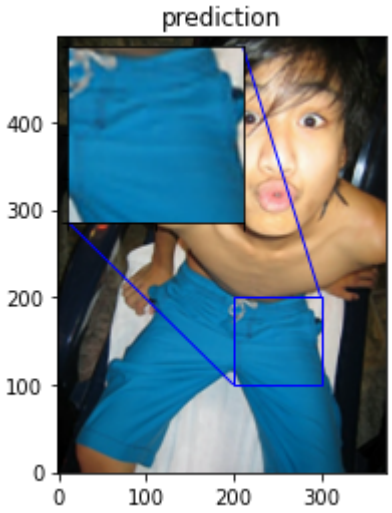




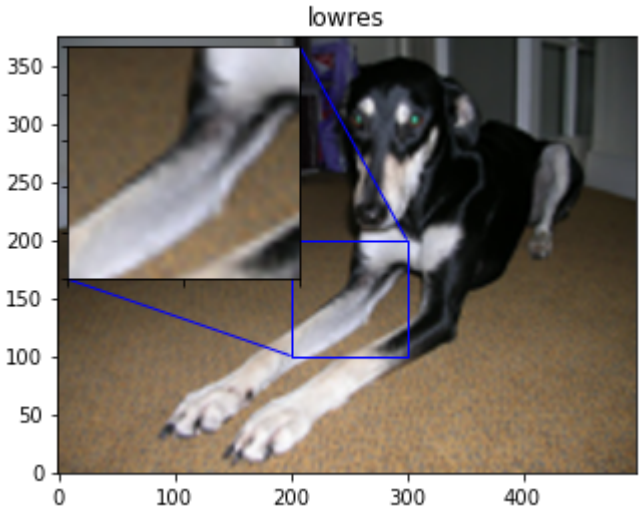

```
PSNR of low resolution image and high resolution image is 28.1352
PSNR of predict and high resolution is 28.9987
```

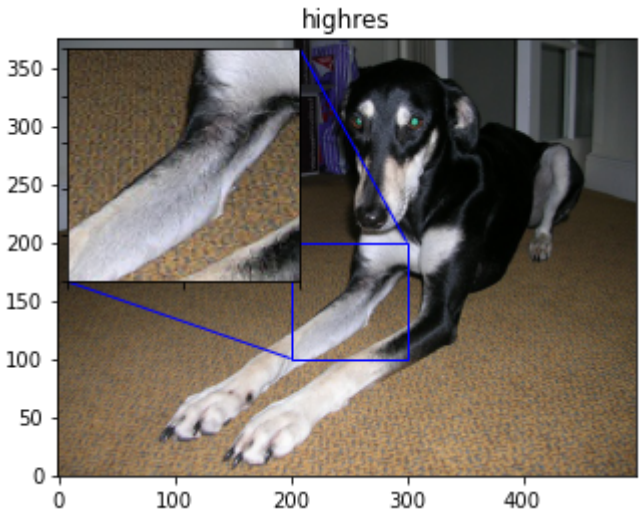


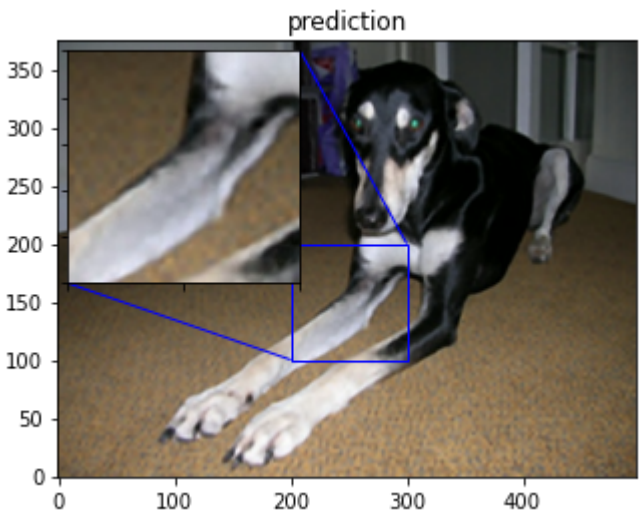





```
PSNR of low resolution image and high resolution image is 26.5851
PSNR of predict and high resolution is 26.9208
```








```
Avg. PSNR of lowres images is 24.6548
Avg. PSNR of reconstructions is 25.3059
```

In [81]:
```
model.save('./model_epoch1')
```

```
INFO:tensorflow:Assets written to: ./model_epoch1\assets
```

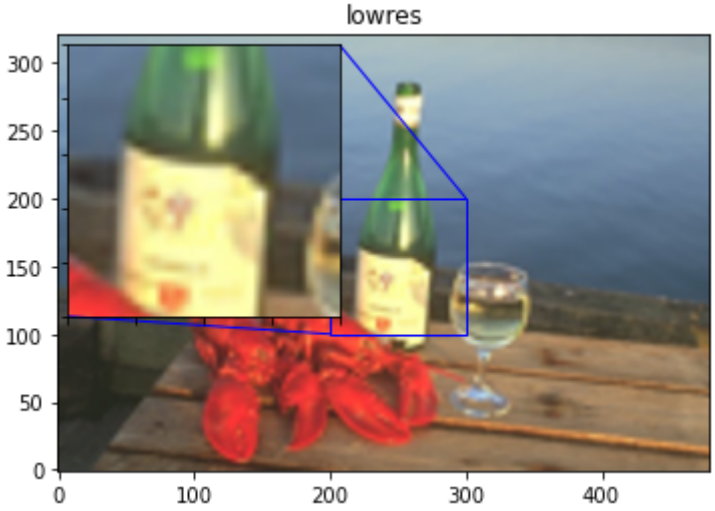
In [44]:
```

```

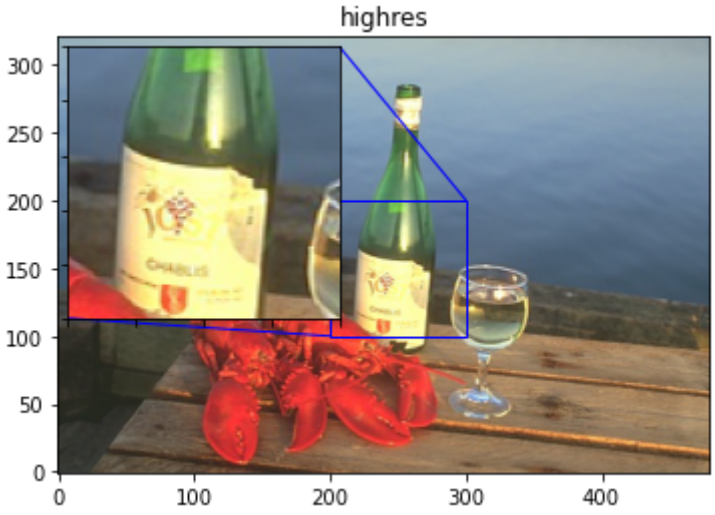
```
'pwd' is not recognized as an internal or external command,
operable program or batch file.
```

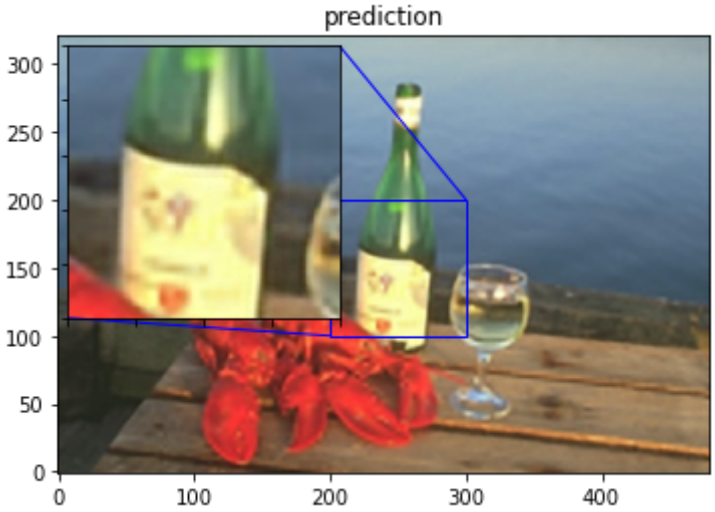
In [46]:
```
# from tensorflow import keras
# modell = keras.models.load_model('./model_')
```

In [47]:
```
# total_bicubic_psnr = 0.0
# total_test_psnr = 0.0

# for index, test_img_path in enumerate(test_img_paths[50:60]):
#     img = load_img(test_img_path)
#     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(modell, 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
#     )
#     print("PSNR of predict and high resolution is %.4f" % test_psnr)
#     plot_results(lowres_img, index, "lowres")
#     plot_results(highres_img, index, "highres")
#     plot_results(prediction, index, "prediction")

# print("Avg. PSNR of lowres images is %.4f" % (total_bicubic_psnr / 10))
# print("Avg. PSNR of reconstructions is %.4f" % (total_test_psnr / 10))
```

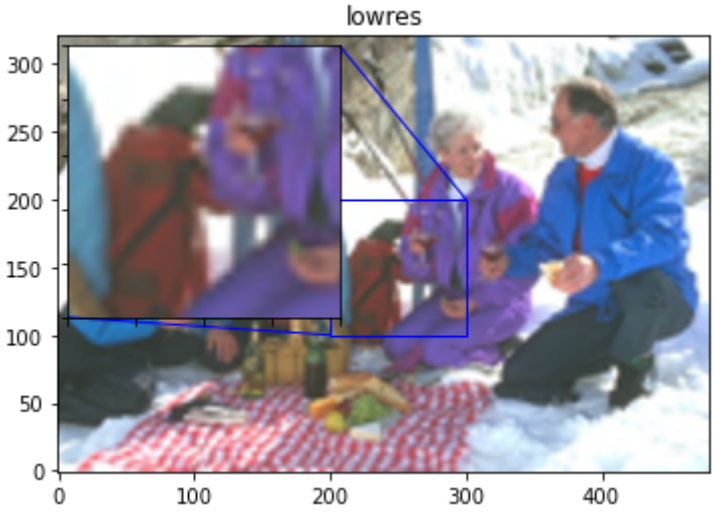
```
PSNR of low resolution image and high resolution image is 29.8502
PSNR of predict and high resolution is 30.2815
```

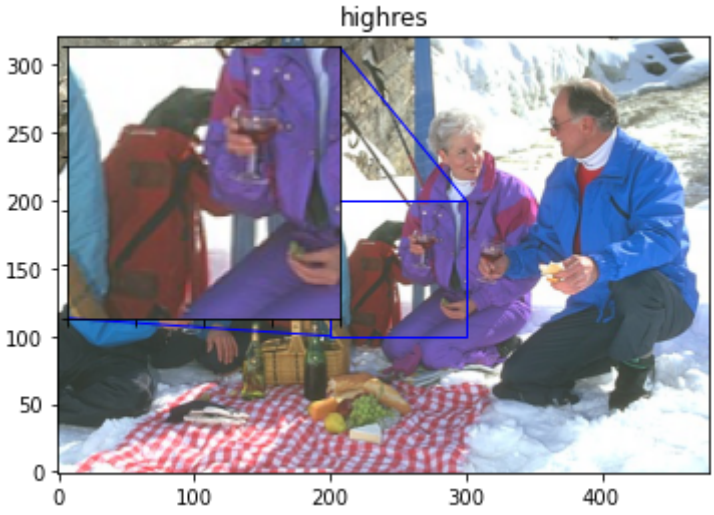


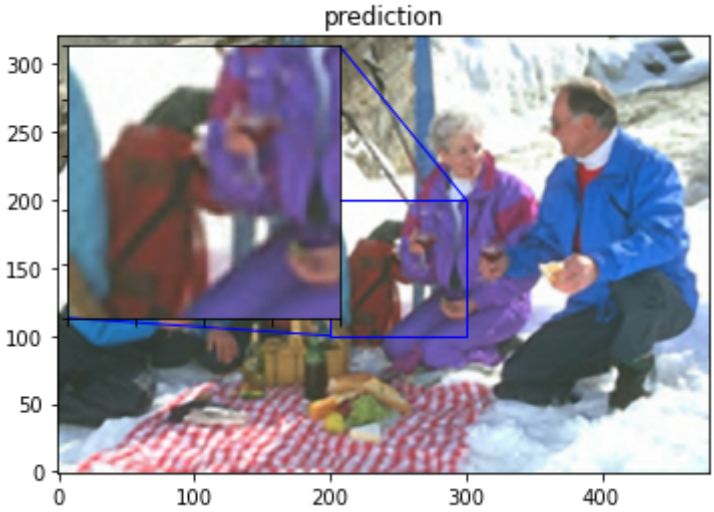




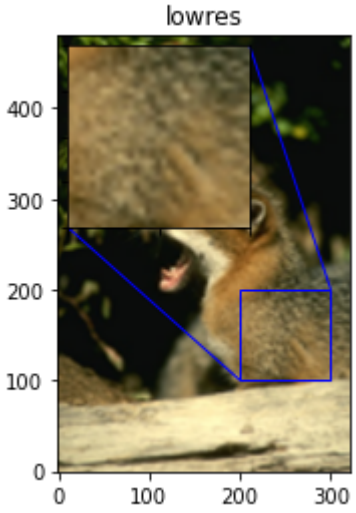

```
PSNR of low resolution image and high resolution image is 24.9783
PSNR of predict and high resolution is 25.8959
```

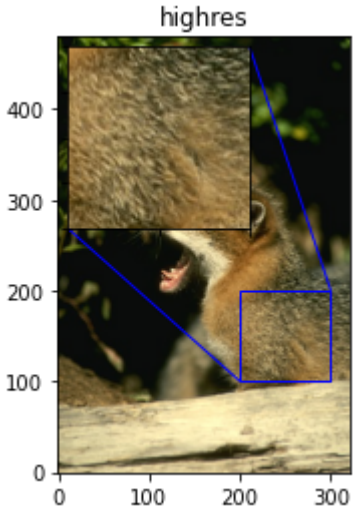


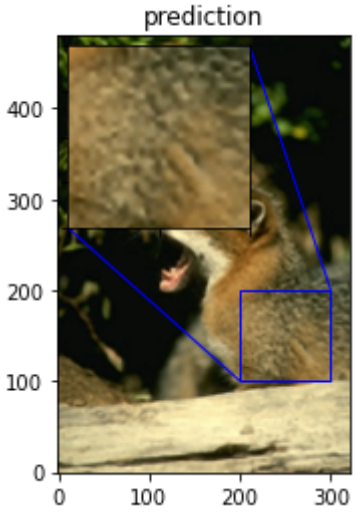




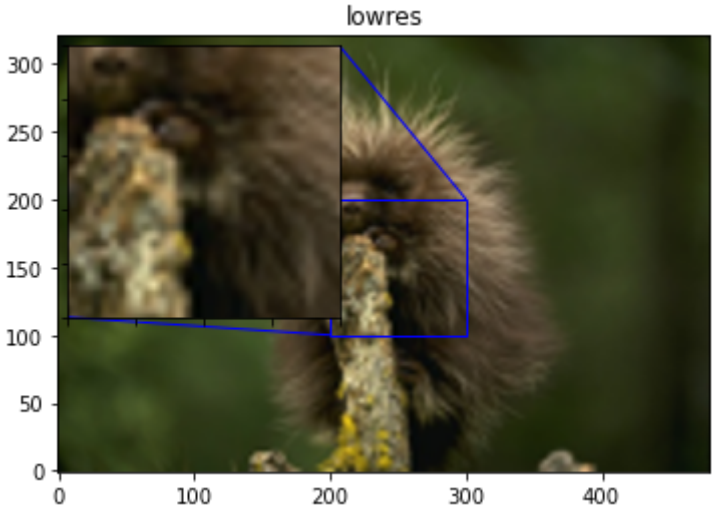

```
PSNR of low resolution image and high resolution image is 27.7724
PSNR of predict and high resolution is 28.3722
```

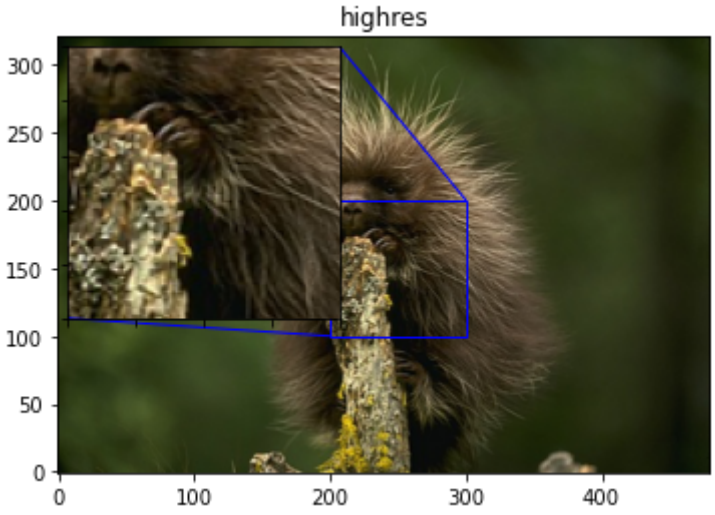


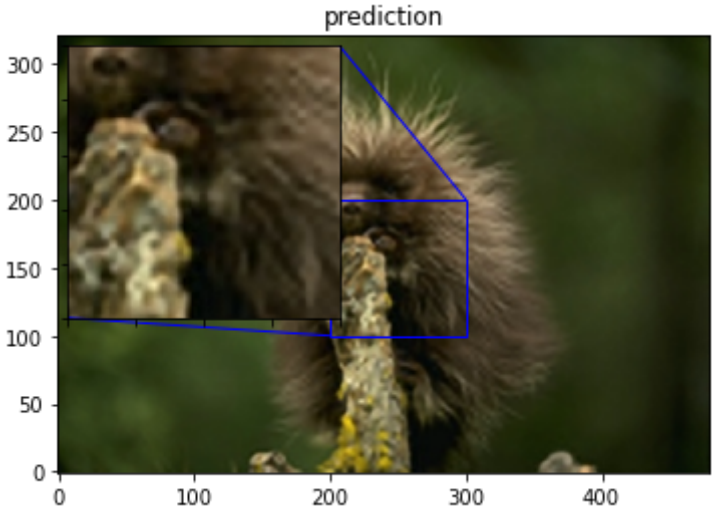




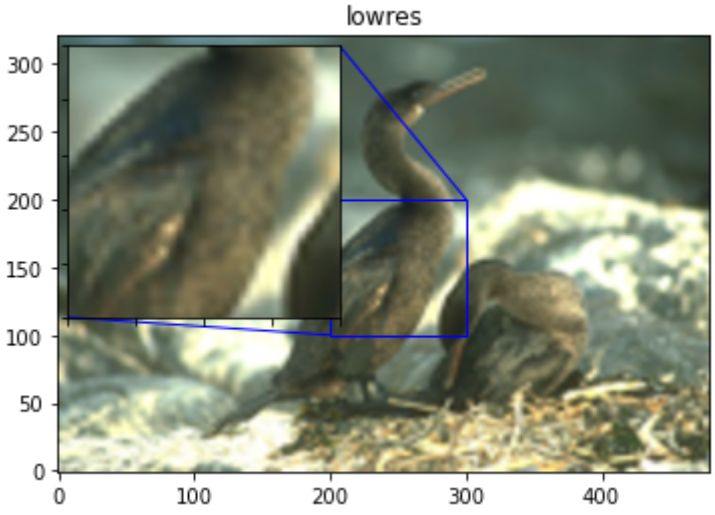

```
PSNR of low resolution image and high resolution image is 28.0314
PSNR of predict and high resolution is 28.2546
```

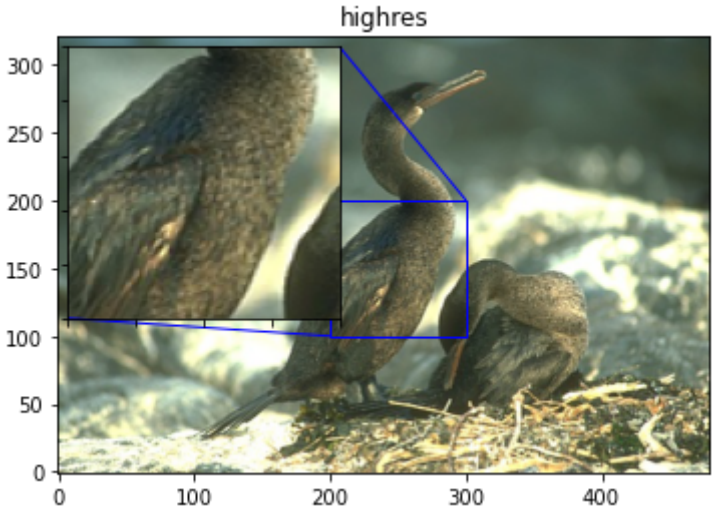


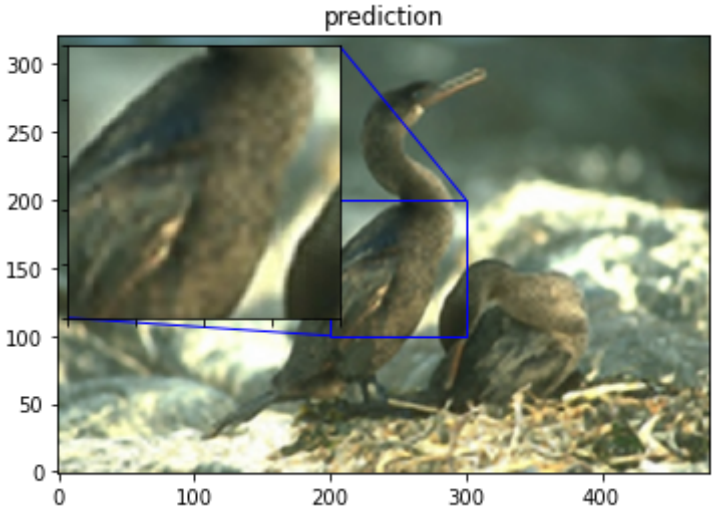




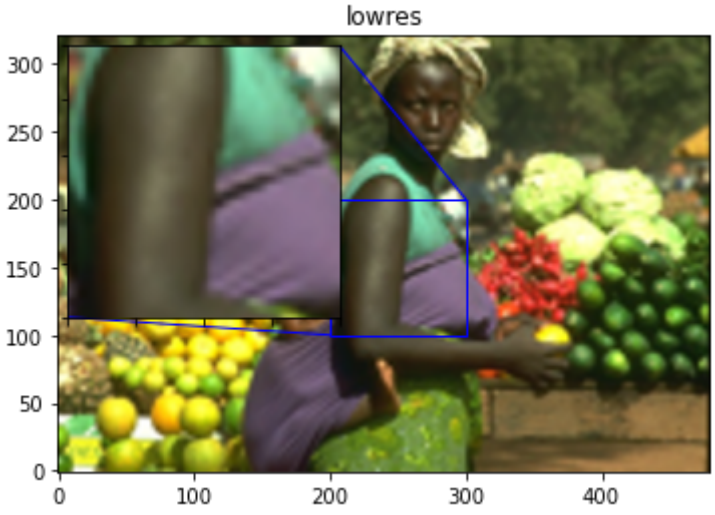

```
PSNR of low resolution image and high resolution image is 25.7630
PSNR of predict and high resolution is 26.3259
```

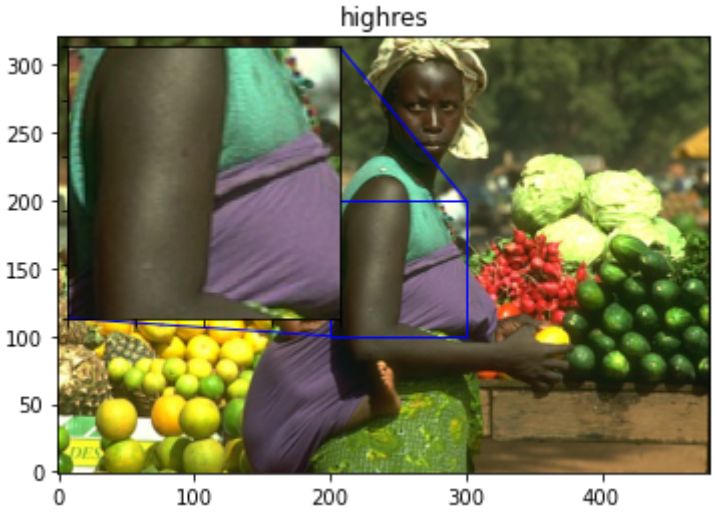


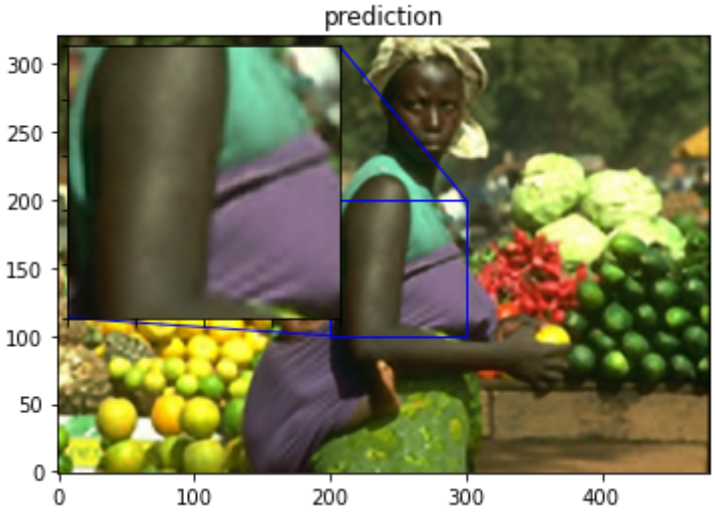




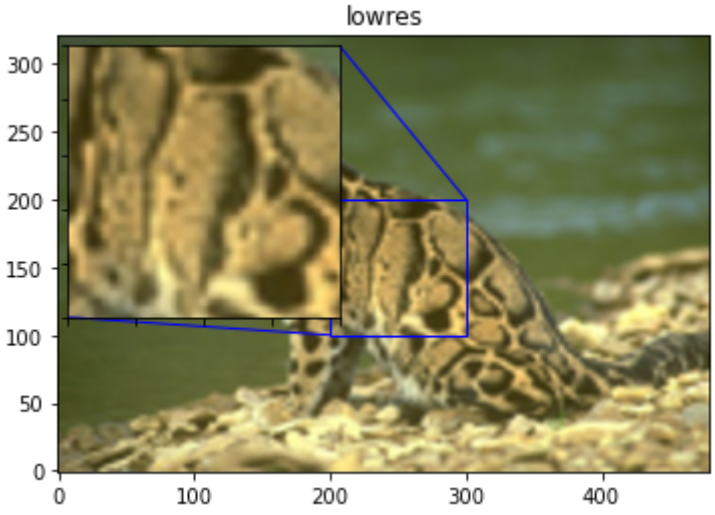

```
PSNR of low resolution image and high resolution image is 25.7874
PSNR of predict and high resolution is 26.5285
```

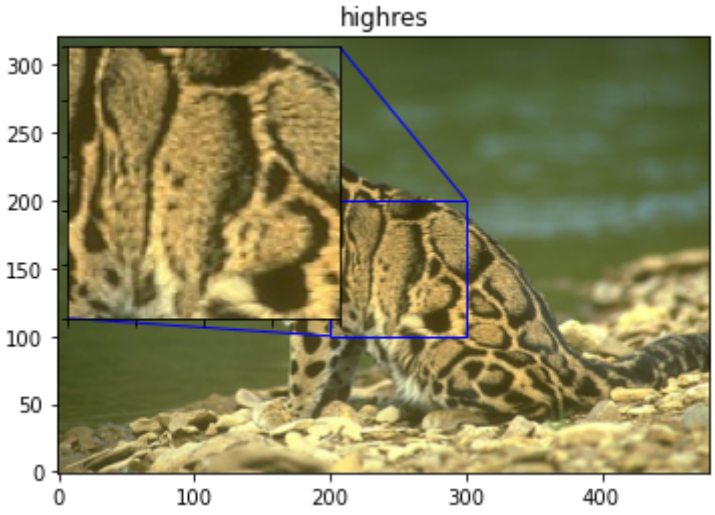


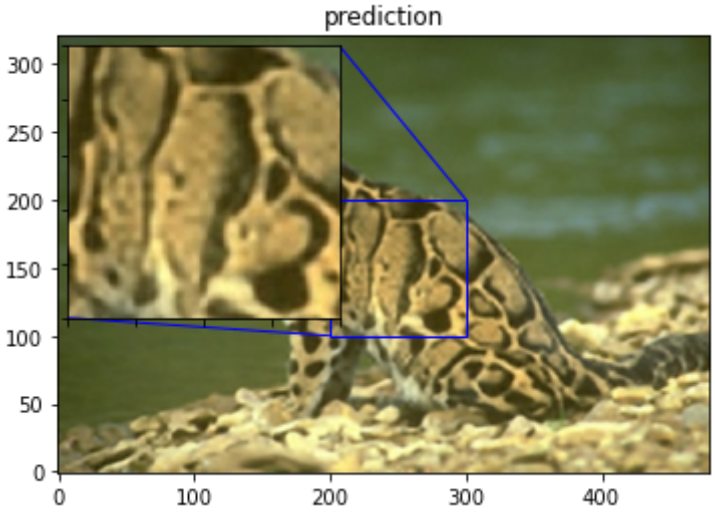




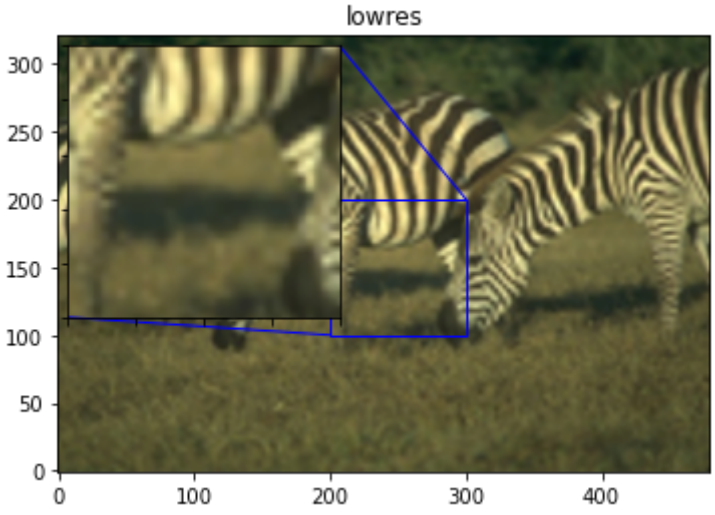

```
PSNR of low resolution image and high resolution image is 26.2512
PSNR of predict and high resolution is 27.1253
```

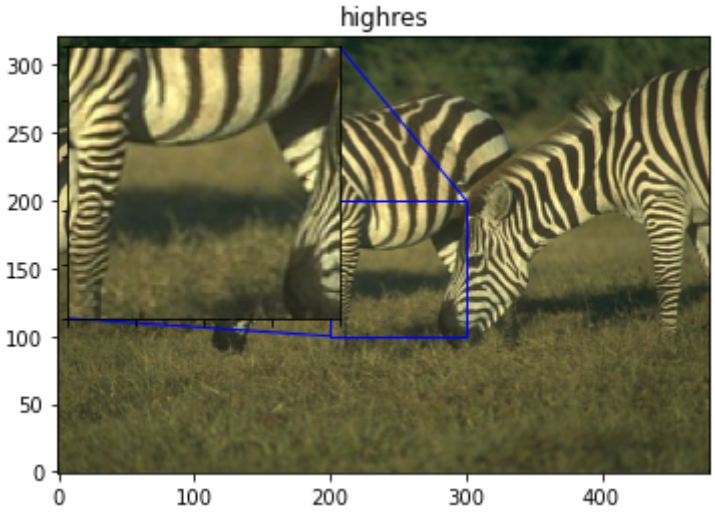


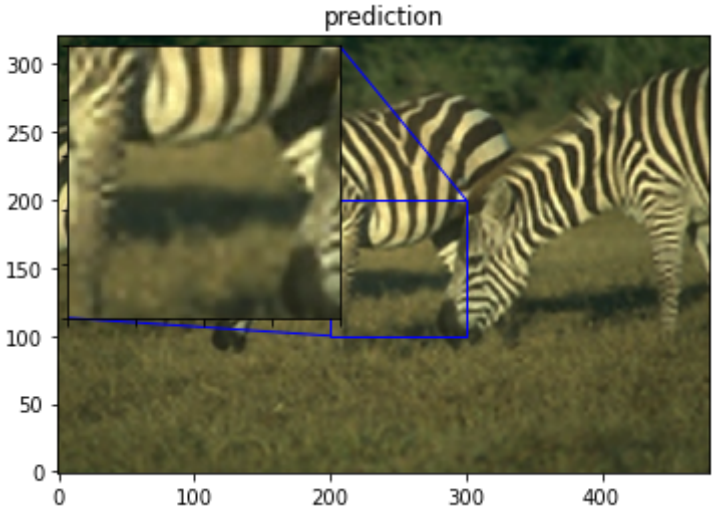




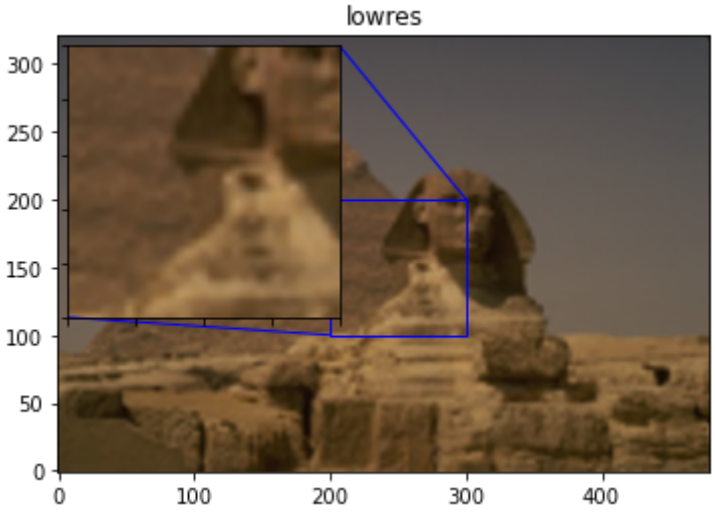

```
PSNR of low resolution image and high resolution image is 23.3820
PSNR of predict and high resolution is 24.6839
```

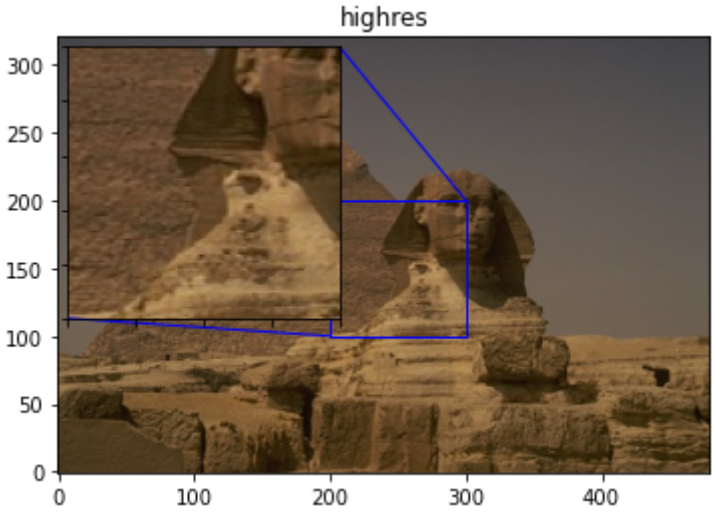


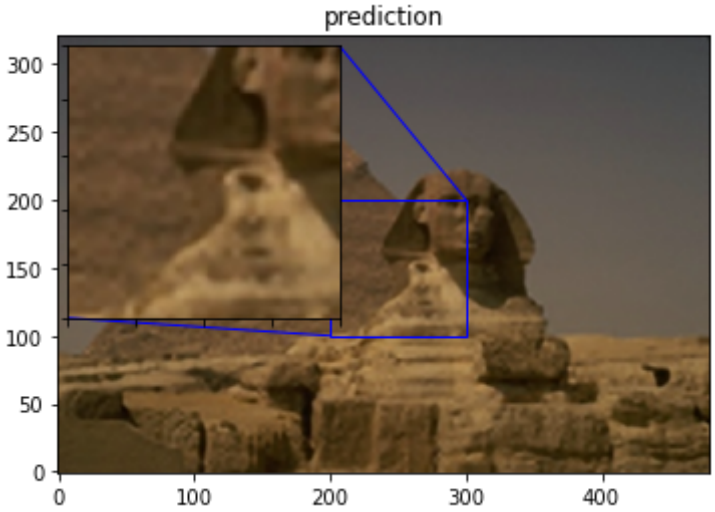




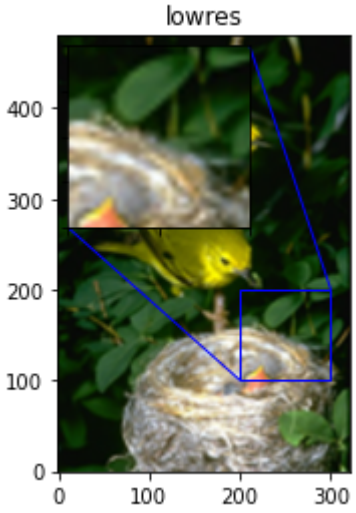

```
PSNR of low resolution image and high resolution image is 29.8914
PSNR of predict and high resolution is 30.0418
```

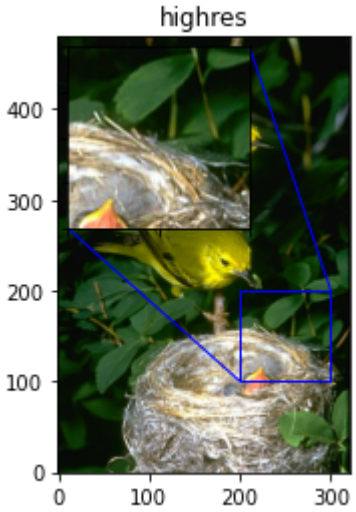


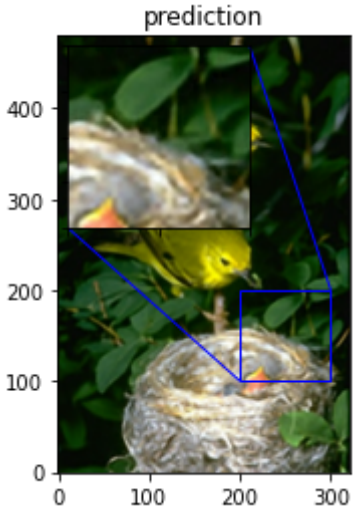





```
PSNR of low resolution image and high resolution image is 25.1712
PSNR of predict and high resolution is 25.6863
```








```
Avg. PSNR of lowres images is 26.6879
Avg. PSNR of reconstructions is 27.3196
```

In [ ]: