Customized code

def resize\_image\_to\_square(image\_np,image\_size=(256,256),preserve\_aspect\_ratio=True):

    image\_np\_extra = image\_np.astype(np.float32)[np.newaxis, ...]

    if  image\_np\_extra.max()>1.0:

         image\_np\_extra = image\_np\_extra / 255.

    if len( image\_np\_extra.shape) == 3:

     image\_np\_extra = tf.stack([ image\_np\_extra,  image\_np\_extra,  image\_np\_extra], axis=1)

     image\_np\_extra= crop\_center(image\_np\_extra)

     image\_np\_extra= tf.image.resize(image\_np\_extra, image\_size, preserve\_aspect\_ratio=True)

    return image\_np\_extra

@functools.lru\_cache(maxsize=None)

def load\_image(image\_url, image\_size=(256, 256), preserve\_aspect\_ratio=True):

  """Loads and preprocesses images."""

  # Cache image file locally.

  image\_path = tf.keras.utils.get\_file(os.path.basename(image\_url)[-128:], image\_url)

  # Load and convert to float32 numpy array, add batch dimension, and normalize to range [0, 1].

  img = tf.io.decode\_image(

      tf.io.read\_file(image\_path),

      channels=3, dtype=tf.float32)[tf.newaxis, ...]

  img = crop\_center(img)

  img = tf.image.resize(img, image\_size, preserve\_aspect\_ratio=True)

  return img

def show\_n(images, titles=('',)):

  n = len(images)

  image\_sizes = [image.shape[1] for image in images]

  w = (image\_sizes[0] \* 6) // 320

  plt.figure(figsize=(w  \* n, w))

  gs = gridspec.GridSpec(1, n, width\_ratios=image\_sizes)

  for i in range(n):

    plt.subplot(gs[i])

    plt.imshow(images[i][0], aspect='equal')

    plt.axis('off')

    plt.title(titles[i] if len(titles) > i else '')

  plt.show()

import cv2

cap = cv2.VideoCapture(r"/video/a.mp4")

frame\_size=256

fourc = cv2.VideoWriter\_fourcc(\*'MP4V')

out = cv2.VideoWriter(r"/video/video\_a\_stylized.mp4",fourc, 60.0, (frame\_size,frame\_size))

style\_name="fire"

print(f"using style\_name={style\_name}")

counter = 0

while True:

    print(f"{counter}")

    flag,frame=cap.read()

    if flag:

        image\_rgb\_np = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

        print(image\_rgb\_np)

        resized\_image\_np = resize\_image\_to\_square(image\_rgb\_np, image\_size=(frame\_size,frame\_size))

        outputs = hub\_module(tf.constant(resized\_image\_np), tf.constant(style\_images[style\_name]))[0]

        stylized\_image= outputs[0]

        image\_pil = tf.keras.preprocessing.image.array\_to\_img(stylized\_image[0])

        image\_bgr\_np = cv2.cvtColor(np.array(np\_image) , cv2.COLOR\_RGB2BGR)

        out.write(image\_bgr\_np )

    else:

         break

    counter = counter+1

out.release()

import cv2

frame\_size=256

style\_name="fire"

cap = cv2.VideoCapture(0)

while True:

    flag,frame=cap.read()

    if flag:

        image\_rgb\_np = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

        print(image\_rgb\_np)

        resized\_image\_np = resize\_image\_to\_square(image\_rgb\_np, image\_size=(frame\_size,frame\_size))

        outputs = hub\_module(tf.constant(resized\_image\_np), ctf.constant(style\_images[style\_name]))[0]

        stylized\_image = outputs[0]

        image\_pil = tf.keras.preprocessing.image.array\_to\_img(stylized\_image[0])

        image\_bgr\_np = cv2.cvtColor(np.array(image\_pil) , cv2.COLOR\_RGB2BGR)

        cv2.imshow("style transfer", image\_bgr\_np)

    if cv2.waitKey(25) & 0xFF == ord('q'):

         break

cap.release()

cv2.destroyAllWindows()