ورقة بحثية مجلة مفتوحة الوصول

التعلّم العميق في مجال الدقّة الفائقة

د. سامر سليمان*، ريم جبيلي**
*(قسم الميكاترونيكس، جامعة المنارة
البريد الإلكتروني: samer.sulaiman@manara.edu.sy)
**(قسم الحاسبات والتحكم الآلي، جامعة تشرين
البريد الإلكتروني: reemjbily123@gmail.com)

الملخص

تستهدف تقنيات الدقّة الفائقة الحصول على صورة عالية الدقّة من صورة منخفضة الدقّة، وقد زادت أهميّة هذا المجال بسبب ظهور الحاجة إلى وجود صور عالية الدقّة في العديد من التطبيقات المهمّة مثل الصور الطبيّة والأمنيّة وغيرها. تطوّرت طُرق الحصول على الصور فائقة الدقّة بعد ظهور تقنيات التعلّم العميق deep learning والتي أظهرت نتائج جيّدة في هذه المهمّة، ونظراً لأهميّة مجال الصور فائقة الدقّة والتعلّم العميق سنقوم في هذه المقالة بشرح أحد نماذج التعلّم العميق المستخدمة في الحصول على صورة عالية الدقّة من صورة منخفضة الدقّة وطريقة بنائه وتدريبه بالاعتماد على أحد مكاتب التعلّم العميق الشهيرة وباستخدام أحد منصات google المستخدمة في التدريب وهي google Colaboratory.

كلمات مفتاحية – الدقّة الفائقة super resolution، التعلّم العميق deep learning، الشبكات العصبيّة neural networks.

I. مقدمة

الصور عالية الدقّة High Resolution Images هي الصور التي تمتلك عدد بيكسلات أكبر في الإنش الواحد أي تملك كميّة معلومات أكبر (يُقصد بكميّة المعلومات الأكبر تفاصيل أكثر في الصورة مثل الخطوط الرفيعة والحواف البارزة... إلخ) [1].

وتُعرف عملية ال super resolution بأنّها عمليّة استرجاع صورة عالية الدقّة من صورة منخفضة الدقّة [5].

تعتبر هذه الصور ذات أهميّة كبيرة وذلك بسبب الحاجة إليها في العديد من الاستخدامات مثل التطبيقات الطبية والمراقبة الذكيّة smart surveillance والاستشعار عن بُعدremote sensing ومهام الرؤية الحاسوبية computer vision tasks [2].

لا يمكن بالضرورة الحصول على صور عالية الدقّة بالرغم من تطوّر أجهزة التصوير الرقميّة وذلك بسبب التأثيرات الخارجيّة التي يمكن أن تسبب تشوّه الصور أو انخفاض التفاصيل الموجودة في الصورة عند التقاطها، فمثلاً عند التقاط صور لوجه شخص ما موجود في مسافة بعيدة عن الكاميرا فإنّ تفاصيل الوجه تكون غير واضحة أو لا يمكن الحصول على الميّزات landmarks الخاصة بوجه هذا الشخص وبالتالي ستؤثر على إمكانيّة التعرّف عليه. كما أنّه لا يمكن استخدام الصور منخفضة الدقّة الملتقطة بأجهزة تصوير غير متطوّرة [2]، وبالتالي ستكون عمليّة الحصول على صور ذات دقّة عاليّة من صور ذات دقّة منخفضة عمليّة مهمّة حتماً. تُعرف الصور الناتجة عن عمليّة تحويل الصور منخفضة الدقّة إلى عالية الدقة بالصور ذات الدقّة الفائقة.

II. طرق الحصول على الصور فائقة الدقّة

IMAGE SUPER RESOLUTION TECHNIQUES

يمكن للحصول على الصور فائقة الدقّة بتطبيق تقنيات معالجة الصورة مثل Frequency Domain Approach و Spatial Domain Super-Resolution Methods وغيرها [3]،

ولكن سنناقش هنا الطرق المعتمدة على التعلّم العميق في الحصول على الصور المرغوبة.

A. الحصول على الصور عالية الدقّة عن طريق نماذج التعلّم العميق

أظهرت نماذج التعلّم العميق نتائج إيجابيّة في الحصول على صور فائقة الدقّة من صور منخفضة الدقّة حيث تطوّرت النماذج من التسلسليّة مثل SRCNN وحتى النماذج الحاليّة المعتمدة على generative adversarial networks وقد أعطت نتائج حتّى في النماذج ذات الطبقات الالتفافيّة القليلة، فنموذج SRCNN يمتلك 3 شبكات عصبيّة فقط ويُبدي نتائج جيدة في معايير القياس المتعارف عليها ولذلك زاد الاهتمام في تطوير هذه النماذج [4] [5].

B. تطوّر نماذج التعلّم العميق المستخدمة في الحصول على الصور عالية الدقّة

في البداية تمّ استخدام نماذج تسلسليّة sequential models مُتدربّة على أزواج صور عالية الدقّة ومنخفضة الدقّة كنماذج supervised، أي أنّ مجموعة البيانات مكونّة من صور منخفضة الدقّة والصور عاليّة الدقّة مقابلة لها وبالتالي عند تدريب النموذج سيتم استخدام الصورة منخفضة الدقّة كدخل ثم يتم تحديث البارامترات اعتماداً على حساب الخطأ بين الصور الناتجة عن النموذج (خرج النموذج) والصور عالية الدقّة الموجودة.

سيتم في هذه المقالة إعطاء شرح مفصل عن نموذج Super Resolution Convolution Neural Networks SRCNN المستخدمة في عملية استخلاص الصور عالية الدقة من صور منخفضة الدقة [4].

1) هيكليّة النموذج:

يعتبر من أبسط نماذج التعلّم العميق المستخدمة في الحصول على الصور فائقة الدقّة، والذي يتكون من ثلاث طبقات التفافيّة تمتلك البارامترات التالية:

جدول 1 مكونات نموذج SRCNN

الطبقة	عدد المرشحات	حجم المرشح	تابع التنشيط
الأولى	128	9 x 9	relu
الثانية	64	3 x 3	relu
الثالثة	1	5 x 5	linear

2) التطبيق العملي لاستخدام النموذج:

يمكن بناء النموذج اعتماداً على أحد أطُر العمل المشهورة مثل Tensorflow, Keras or Pytorch، هنا سيتم استخدام Keras الخاصة بالTensorflow وذلك لسهولة استخدامها وفهمها بالنسبة لأي مبتدئ.

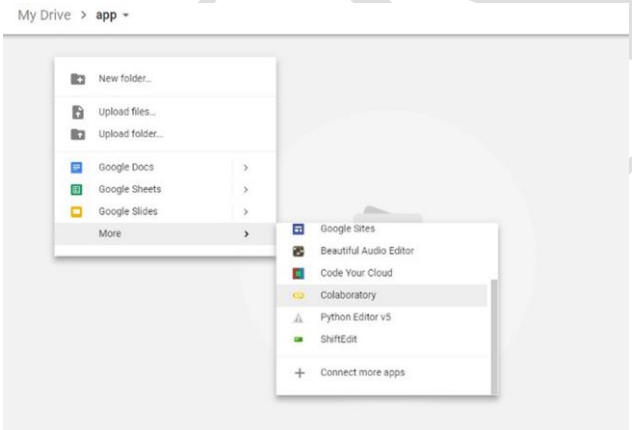
يمكن استخدام بيئة عمل google colab التي توفرها google لتطبيق النموذج عملياً وهي مشابهة لـ jupyter notebooks واتي توفّر GPU مجانيّة ومساحة تخزين Hard Disk تصل إلى 107.72 GB وRAM تصل إلى 12.69 GB حالياً.

- **تهيئة بيئة العمل:**

يتضمن ذلك إنشاء notebook وحفظها مباشرةً في google drive الخاص بالمطوّر، ولتنفيذ ذلك يتم ذلك عن طريق الضغط على الزر اليميني ثمّ اختيار More من القائمة المنسدلة ثمّ اختيار خيار Colaboratory أي:

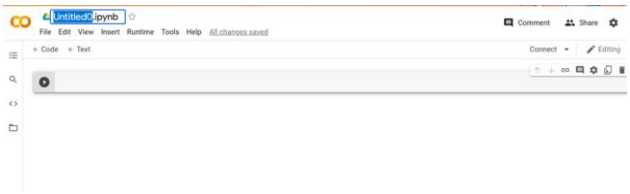
Right Click→More→Colaboratory

يظهر الشكل التالي توضيحاً للخطوة السابقة:



الشكل 1 . إنشاء notebook وحفظها مباشرةً في google drive

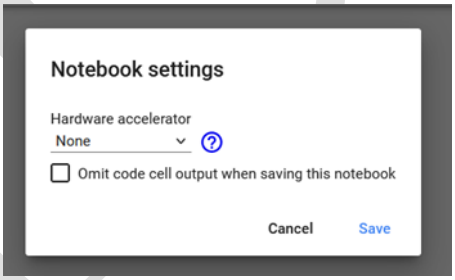
ستظهر الـ notebook باسم Untitled0 كما في الشكل التالي:



الشكل 2 . نافذة الـ notebook الجديدة

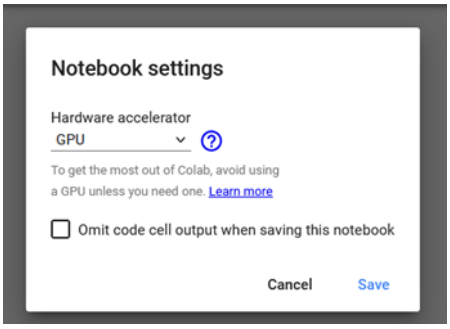
يمكن تغيير الاسم طبعاً بكتابة الاسم الجديد مكان الاسم الحالي، وسنقوم بتغيير الاسم إلى Tutorial.

هذه الـ notebook تستخدم CPU للتدريب بشكل افتراضي ولتغييرها إلى GPU أو TPU يجب النقر على خيار Runtime ثمّ اختيار change runtime type فيظهر الشكل التالي:



الشكل 3 . نافذة تغيير نوع العتاد الصلب المستخدم في التدريب

نضغط على السهم بجانب None ثم نقوم باختيار GPU كما يظهر:



الشكل 4 . اختيار الـGPU للتدريب

ثمّ نضغط زر save وبهذا تكون بيئة العمل جاهزة.

- **تضمين المكتبات:**

يمكن كتابة تعليمات لغة python في خليّة ثم تنفيذها بالنقر على زر التشغيل في بداية كل خليّة أو ضغط shift ثمّ enter الذي يقوم بالتنفيذ والانتقال إلى خليّة جديدة وداخل الخليّة سنقوم باستدعاء المكتبات الضرورية لبناء النموذج، كما يظهر في الكود البرمجي التالي:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential #for creating the model
from tensorflow.keras.layers import Input, Conv2D #for adding layers to the model
from tensorflow.keras.optimizers import Adam #the optimizer of the model
from tensorflow.keras.callbacks import ModelCheckpoint #checkpoints are used to save the weights and bia
import cv2 #for image processing
import numpy as np #for matrices manipulation
import matplotlib.pyplot as plt #for displaying images
import math #for math operations
import os #for file manipulation
import h5py #for dealing with h5 file
```

الشكل 5 . استدعاء المكتبات اللازمة لبناء النموذج وتدريبه

شرح التعليمات:

السطر الأول: هو استدعاء لمكتبة tensorflow والتصريح عنها باسم tf ليتم استخدام tf بدلاً من tensorflow في التعليمات اللاحقة.

السطر الثاني: استخدام النموذج Sequential من مكتبة tensorflow.Keras وصنف models (تجدر الإشارة إلى أنّه في النموذج التسلسلي، يتم إضافة الطبقات تباعاً ويمكن إضافتها بشكل تفرّعي عن طريق نموذج يدعى Model).

السطر الثالث: استدعاء أصناف بناء الطبقات من مكتبة tensorflow.Keras وصنف layers وهذه الأصناف هي Input لتشكيل طبقة الدخل وهي أول طبقة في النموذج وConv2D لتشكيل الطبقات الالتفافيّة للنموذج.

السطر الرابع: استدعاء لتابع التحسين Adam وتوابع التحسين هي خوارزميات تستخدم لتغيير صفات النموذج مثل الأوزان ومعامل التعلّم من أجل تقليل الخسارة ويوجد أنواع عديدة منها أشهرها gradient descent و SGD وAdam وغيرها.

السطر الخامس: استخدام الـ ModelCheckpoint وذلك لحفظ الأوزان والانحيازات.

السطر السادس: استدعاء مكتبة open cv وهي مكتبة معروفة تُستخدم لإجراء عمليات على الصور وتتضمن الكثير من التوابع المهمّة في معالجة الصورة.

السطر السابع: استدعاء مكتبة numpy باسم np وهي مكتبة مشهورة التعامل مع المصفوفات.

السطر الثامن: استدعاء matplotlib.pyplot لعرض الصور.

السطر التاسع: استدعاء مكتبة math لإجراء العمليات الحسابيّة.

السطر العاشر: استدعاء مكتبة os للتعامل مع الملفات والمجلدات من إنشاء وحذف وغيرها.

السطر الحادي عشر: استدعاء مكتبة التعامل مع ملفات الـ h5.

- **تشكيل مجموعة بيانات الدخل للنموذج:**

دخل النموذج هو عبارة عن صور منخفضة الدقّة بأبعاد محدّدة والخرج هو صورة تسمى صورة فائقة الدقّة ذات أبعاد أكبر من الصورة منخفضة الدقّة بمقياس معيّن، فمثلاً إذا كانت صورة الخرج المرغوبة ذات أبعاد 200 x200 فإنّ صورة الدخل منخفضة الدقّة هي 100 x100 أي أنّ النموذج يحوّل الصورة ذات الأبعاد 100 x100 والتي تعتبر منخفضة الدقّة إلى صورة فائقة الدقّة ذات أبعاد 200 x 200 أي أنّ مقياس التكبير هو 2.

وبالتالي، لتشكيل صور الدخل منخفضة الدقّة يجب تصغير أبعاد كلّ صورة عالية الدقّة إلى النصف أو إلى قيمة مقياس محددة ولأن نموذج SRCNN يأخذ دخل مساوٍ لحجم الخرج يتم إعادة تكبير الصورة إلى الأبعاد الأساسيّة، بعبارة أخرى:

First step:

LR width=HR width/scale.

LR height=HR height/scale.

Second step:

LR width_final=HR width.

LR height_final=HR height.

يوضّح الشكل التالي االتعليمات اللازمة للحصول على الصورة منخفضة الدقّة:

```
#two resize operation to produce training data and labels
lr_img2 = cv2.resize(hr_img, (shape[1] // 2, shape[0] // 2))
lr_img2 = cv2.resize(lr_img2, (shape[1], shape[0]))
```

الشكل 6 . الكود البرمجي للحصول على الصور منخفضة الدقّة

- **بناء النموذج:**

يتكون النموذج من ثلاث طبقات وبمعاملات محدّدة كما ذكرنا سابقاً وبالتالي لبناء النموذج سنكتب:

```
#building the model
def SRCNN():
    SRCNN = Sequential()
    SRCNN.add(Conv2D(128,(9,9),activation='relu', padding='valid',use_bias=True,
                     input_shape=(32, 32, 1)))
    SRCNN.add(Conv2D(64,(3,3),activation='relu', padding='same', use_bias=True))
    SRCNN.add(Conv2D(1,(5,5),activation='linear', padding='valid', use_bias=True))
    adam = Adam(learning_rate=0.0001)
    SRCNN.compile(optimizer=adam, loss='mean_squared_error', metrics=['mean_squared_error'])
    return SRCNN
```

الشكل 7 . الكود البرمجي لبناء النموذج

نعرّف النموذج SRCNN كنموذج تسلسلي في السطر الأول، ليتم بعدها إضافة الطبقة الأولى كما في السطر التالي مع البارامترات التالية:

128 مرشّح وبأبعاد 9X9 لكلّ مرشّح وتابع تنشيط relu (تابع التنشيط هو تابع رياضي يقوم بتغيير قيمة الأوزان في طبقة ما قبل تمريرها إلى الطبقات التالية وتابع relu يسمح للقيم الموجبة بالانتقال للطبقة التالية كما هي أمّا القيم السالبة فيقوم بجعلها صفر). بدون وجود padding ويشار لذلك في العبارة:

Padding='valid'

والـ padding هو تقنية لجعل حجم الصورة مناسب للعمليات المراد تطبيقها باعتبار أنّ طبقة الـ convolution تغير أبعاد صورة الدخل.

عند استخدام padding من نوع valid يتم الإبقاء على حجم مصفوفة الدخل كما هو فإذا كانت مصفوفة الدخل ذات الأبعاد (h X w) ستصبح (h_new X w_new) وهي الأبعاد الناتجة عن تمرير الدخل على طبقة convolution وفق علاقة رياضيّة هي:

$$dimension = \frac{dimension - filter_{size} + 2 * padding}{stride} + 1$$

حيث:

dimension: أحد أبعاد الصورة إما الطول أو العرض.

filter_size: هو حجم المرشح المستخدم وفي هذه الطبقة هو 3

padding: يتم الحفاظ على حجم الدخل كما هو حتّى بعد المرور على طبقة الـ convolution في حال كان من نوع same وذلك عن طريق زيادة عدد من الأسطر والأعمدة الصفريّة على مصفوفة الدخل أما في حال كان valid فهو يسمح بتغيير أبعاد الدخل.

stride: يحدد عدد الأسطر والأعمدة التي سيتم تجاوزها عند إجراء العملية الالتفافيّة من قبل المرشح وفي مثالنا هو دائماً 1 (القيمة الافتراضيّة).

عند الحاجة إلى استخدام الانحياز يتم ذلك عن طريق إعطاء البارامتر use_bias القيمة True

يتم تحديد أبعاد الدخل باستخدام البارامتر input_shape وهنا نجعل الأبعاد هي (32,32,1) أي أنّ الدخل يجب أن يملك ارتفاع 32 وعرض 32 ولكن بقناة واحدة أي صورة بلون رمادي. كذلك الأمر لباقي الطبقات مع الاختلاف من ناحية عدد المرشّحات وأبعادها ونوع الـ padding مع ملاحظة أن تابع التنشيط في الطبقة الأخيرة هو linear أي يسمح للقيمة بالعبور كما هي، ثم تعريف تابع التحسين adam، كما يلي:

```
adam = Adam(learning_rate=0.0001)
```

يعبر معدّل التعلّم learning rate عن القيمة التي تتحكّم بمقدار التغيّر في الأوزان والانحيازات في الشبكة أثناء عمليّة التعلّم بحيث إذا كان معدّل التعلّم كبيراً فإنّ دقّة النموذج ستقلّ ولكن التدريب سيحدث بسرعة كبيرة، بينما إذا كان معدل التعلّم صغيراً ستزداد دقّة النموذج ولكن التدريب سيحدث ببطء وبالتالي فإنّ اختيار قيمة مناسبة لمعدّل التعلّم أمر مهم.

- **ترجمة النموذج compiling the model:**

أي جعل هذا النموذج ذو وجود مادي مناسب في الذاكرة، ويظهر في الكود البرمجي التالي:

```
SRCNN.compile(optimizer=adam,
              loss='mean_squared_error',
              metrics=['mean_squared_error'])
```

تحتاج عمليّة الترجمة إلى تحديد تابع التحسين وهنا تم استخدام التابع Adam، وتحديد تابع الخسارة (وهو التابع الذي يقوم في حساب الفرق بين خرج النموذج الفعلي والخرج الحقيقي المطلوب الوصول إليه) وهو هنا mean_squared_error ويقوم هذا النوع من توابع الخسارة في حساب الفرق بين كل بيكسل من صورة الخرج مع ما يقابلها من الخرج الحقيقي المرغوب وتربيع هذا الفرق ثمّ حساب المجموع الكلّي لمربع الفروق. أما البارامتر metrics هو اختياري وهو لتحديد أي قيمة نريد عرض تغير قيمها أثنا عمليّة التدريب وهنا نريد التركيز على قيم الخطأ.

- **تدريب النموذج:**

هو المرحلة النهائية حيث يتم تدريب النموذج على صور الدخل الرماديّة عن طريق التابع fit الذي يظهر في الكود البرمجي التالي:

```
srcnn_model.fit(data, label, batch_size=128,
                validation_data=(val_data, val_label),
                callbacks=callbacks_list,
                shuffle=True, epochs=100, verbose=1)
```

سيتم شرح البارامترات المستخدمة ضمن الكود البرمجي السابق كما يلي:

data: هي بيانات الدخل (الصور منخفضة الدقّة).

label: بيانات الخرج الحقيقيّة الموافقة للدخل (الصور عالية الدقّة).

batch_size: عدد عناصر الدخل التي سيتم التدريب عليها في كلّ تكرار.

validation_data: عناصر التحقّق وهي زوج صور منخفضة الدقّة وعالية الدقّة لاختبار النموذج على عناصر لم يتدرّب عليها وذلك لزيادة دقّة النموذج والتحقّق من وجود overfitting (قيمة غير مرغوبة).

callbacks: لتنفيذ تعليمات محدّدة بعد الانتهاء من كلّ عملية تكرار.

shuffle: لخلط بيانات التدريب قبل كل تكرار وذلك لزيادة دقّة النموذج.

epochs: لتحديد عدد مرات التدريب.

verbose: ليحدد شكل التدريب.

الكود البرمجي الذي يقوم بتحميل بيانات التدريب والتحقّق وتعريف checkpoint للحفاظ على قيم الوزان والانحيازات أثناء عملية التدريب:

```
#training the model
def train():
    srcnn_model = SRCNN()
    data, label = read_training_data("./crop_train.h5")
    val_data, val_label = read_training_data("./test.h5")

    checkpoint = ModelCheckpoint("SRCNN.h5", monitor='val_loss', verbose=1, save_best_only=True,
                                 save_weights_only=False, mode='min')
    callbacks_list = [checkpoint]
    #start the training
    srcnn_model.fit(data, label, batch_size=128, validation_data=(val_data, val_label),
                    callbacks=callbacks_list, shuffle=True, epochs=100, verbose=1)


if __name__ == "__main__":
    train()
```

- **اختبار النموذج على صور دخل غير مدرّب عليها:**

في البداية سيتم تحميل بنية النموذج كما يلي:

srcnn_model=SRCNN()

الخطوة الثانية هي تحميل الأوزان الناتجة عن التدريب عن طريق السطر:

scrnn_model.load_weights(model_pth)

حيث model_pth هو مسار أوزان النموذج الناتجة عن عمليّة التدريب. تشكيل صورة الاختبار لتوافق دخل النموذج. يبيّن الكود التالي عمليّة تشكيل صورة الاختبار الموافقة.

```
Y_img = cv2.resize(img[:, :, 0], (shape[1] // scale, shape[0] // scale), cv2.INTER_CUBIC)
Y_img = cv2.resize(Y_img, (shape[1], shape[0]), cv2.INTER_CUBIC)
img[:, :, 0] = Y_img
img = cv2.cvtColor(img, cv2.COLOR_YCrCb2BGR)
#save the input (low resolution) image
cv2.imwrite(INPUT_NAME, img)

Y = np.zeros((1, img.shape[0], img.shape[1], 1), dtype=float)
Y[0, :, :, 0] = Y_img.astype(float) / 255.
```

من أجل الحصول على خرج النموذج يمكن استخدام السطر البرمجي التالي:

```
pre = srcnn_model.predict(Y, batch_size=1) * 255.
```

يوضح الكود البرمجي التالي عملية تخزين خرج النموذج وهو الصورة ذات الدقّة الفائقة:

```
pre = srcnn_model.predict(Y, batch_size=1) * 255.
pre[pre[:] > 255] = 255
pre[pre[:] < 0] = 0
pre = pre.astype(np.uint8)
img = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
img[6: -6, 6: -6, 0] = pre[0, :, :, 0]
img = cv2.cvtColor(img, cv2.COLOR_YCrCb2BGR)
#save the output (super resolution) image of the model
cv2.imwrite(OUTPUT_NAME, img)
```

من أجل حساب تابع الـ PSNR والذي يعبّر عن مقدار الاختلاف في بيكسلات صورتين (خرج النموذج وهو الصورة فائقة الدقّة والصورة الفعلية أي ذات الدقّة العالية) نقوم بكتابة الأسطر البرمجية التالية:

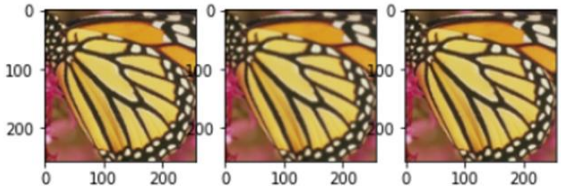
```
# psnr calculation:
im1 = cv2.imread(IMG_NAME, cv2.IMREAD_COLOR)
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2YCrCb)[6: -6, 6: -6, 0]
im2 = cv2.imread(INPUT_NAME, cv2.IMREAD_COLOR)
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2YCrCb)[6: -6, 6: -6, 0]
im3 = cv2.imread(OUTPUT_NAME, cv2.IMREAD_COLOR)
im3 = cv2.cvtColor(im3, cv2.COLOR_BGR2YCrCb)[6: -6, 6: -6, 0]

print("bicubic:")
print(cv2.PSNR(im1, im2))
print("SRCNN:")
print(cv2.PSNR(im1, im3))
```

III. النتائج

تم الاختبار على صورة الفراشة الشهيرة من مجموعة البيانات Set5. يبيّن الشكل (8) الصيغ الثلاث للصورة المختبرة:

- الصورة الأولى من اليسار هي الصورة عالية الدقّة.
- الصورة في الوسط هي الصورة منخفضة الدقّة.
- الصورة أقصى اليمين هي الصورة ذات الدقّة الفائقة.



الشكل 8 . الصيغ المختلفة للصورة

وقيم الـ psnr هي:

bicubic: 24.69

SRCNN: 30.35

حيث bicubic هي الصورة التي تمّ تكبيرها وتصحيحها باستخدام الـ interpolation المسمّى bicubic وهو أحد تقنيات التقريب المستخدمة في معالجة الصورة.

IV. الخاتمة:

قمنا في هذه المقالة بتوضيح أهميّة وجود صور ذات دقّة فائقة وأحد نماذج التعلّم العميق المستخدمة في الحصول على صور فائقة الدقّة والتطبيق العملي لهذا النموذج مع اختباره على صورة من مجموعة بيانات مشهورة، وتشكّل هذه المقالة انطلاقة لمن يريد الدخول في مجال التعلّم العميق والدقّة الفائقة التي تعتبر أحد مجالات معالجة الصورة.

يتوفّر الكود البرمجي كاملاً في الـrepository التالي:

https://github.com/ReemJbily/SRCNN-google-colaboratory-tutorial

من أجل التوسّع يمكن العودة إلى الـrepository الأصلي على الرابط التالي:

https://github.com/MarkPrecursor/SRCNN-keras

المراجع:

[1] Gaidhani, P. (n.d.). *Super Resolution.*

[2] Chen, H., He, X., Qing, L., Wu, Y., Ren, C., & Zhu, C. (2021, Mar 3). Real-World Single Image Super-Resolution. p. 18.

[3] Karimi, E., Kangarloo, K., & javadi, S. (2014, Feb). A Survey on Super-Resolution Methods for Image Reconstruction.

[4] Dong, C., Loy, C., He, K., & Tang, X. (2015, Jul 31). Image Super-Resolution Using Deep. p. 14.

[5] Wang, Z., Chen, J., Hoi, S., Fellow, & IEEE. (2020, Feb 8). Deep Learning for Image Super-resolution:. p. 24.

منشورات المؤلف (د. سامر سليمان):

[1]. Sulaiman, S., Lehnert, R., Dai, Q. (2009). Improved QoS-Aware Awitching Mechanism for PIM-SM Protocol. 3th IEEE IMSAA.

[2]. Türk, S., Sulaiman, S., Haidine, A., Michaelis, Th., Lehnert, R. (2009). Approaches for the migration of optical backbone networks towards Carrier Ethernet: GLOBECOM 2009/EFSOI09.

[3]. Sulaiman, S., Haidine, A., Lehnert, R. (2009). Performance Evaluation of Center Search Algorithms Used for Dynamic Rendevzous Point Relocation.

[4]. Sulaiman, S., Haidine, A., Lehnert, R., Türk, S. (2009). Comparative Study of Multicast Protection Algorithms Using Shared Links in 100GET Transport Network.

[5]. Dai, Q., Lehnert, R. Sulaiman, S. (2008). An Adaptive Packet Dropping Algorithm for Improved VoIP Quality at ADSL.

[6]. Sulaiman, S. (2008). Optimization of Multicast Distribution Trees. Workshop der Fachgruppe 5.2.1, TU Dortmund.

[7]. Sulaiman, S. (2006). Investigation of End-2-End delay in IP-Multicast Networks: *Workshop der TU Dresden, der Universität Twente*.

[8]. Sulaiman, S. (2003). IP Multicast Routing Protocols: MMB/Dagstuhl Seminar Performance of MobileSystem, SchlossDagstuhl, 9.-12.

[9]. Baumann, M., Marandin, D., Sulaiman, S. (2002). Combined Modelling of TCP and MultiRED in DiffServ Networks: Proc. 2nd Polish-German Teletraffic Symposium PGTS 2002; Gdansk; 23.-24.9.

[10]. Baumann, M., Marandin, D., Sulaiman, S. (2005). Combined modelling of TCP and multiRED in DiffServ networks: European Transactions on Telecommunications, Vol. 16, No. 3, pp. 217-224.

[11]. Alkheir, J., Sulaiman, S., Mualla, R. (2020). Performance Evaluation on the Effect of Different Text Representation Models on the Image Captioning Systems. *Tishreen University Journal*, Vol. 42, No. 4, print ISSN: 2097-3081.

[12]. Alkubeily, M., Sulaiman, S., Esber, GM. (2021). Designing a Virtual Platform for Modeling Nodes in Wireless Sensor Networks at the Central Processing

Unit Level. *Journal of Engineering Sciences and Information Technology*, Vol. 5, No. 5.

[13]. Alkubeily, M., Sulaiman, S., Esber, GM. (2021). Performance Evaluation of the Kernel Based Wireless Sensor Network Simulator Using an Authentication Algorithm. *Tishreen University Journal*, Vol. 43, No. 4.

[14]. Alkheir, J., Sulaiman, S., Mualla, R. (2021). Using Image Pre-classification to improve the accuracy of the image captioning systems. *Journal for the Engineering sciences*, Vol.37 No.2

[15]. Alkheir, J., Sulaiman, S., Mualla, R. (2020). Performance Evaluation on the Effect of Different Text Representation Models on the Image Captioning Systems. *Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series*, Vol. 42 No.4.

[16]. Alkubeily, M., Sulaiman, S., Esber, GM. (2020). WINDOWS FORM APPLICATION FOR VIRTUAL MINIMIZED PLATFORM KERNEL FOR WIRELESS SENSOR NETWORK SIMULATOR. *Far East Journal of Electronics and Communications*, Vol. 42 No.4.