Autoencoders para colorear imágenes

Presentado por:

David Felipe Rojas 2170136

Nicolás Galván 2170104

Hazel Pinzón 2163022

Universidad Industrial de Santander

Formulación del problema:

Alrededor de hace 60 años aún no se conocía lo que era una fotografía a color, por lo que los momentos, recuerdos, sentimientos, vivencias y lugares quedaban plasmados e inmortalizados en fotografías con escasa resolución y en escalas de grises o a blanco y negro.

Si bien dichas imágenes nos evocan cierta nostalgia y recuerdos de la época, en algunas ocasiones deseamos que esas fotografías se encuentren a color para así poder tener una percepción más cercana a la realidad, tal como se encuentra plasmada en la fotografía original.

Formulación del problema:

Por lo tanto, para resolver dicho problema de colorización de imágenes se ha ido evolucionando de métodos de colorizacion manual que resultaban costosos e ineficientes a métodos por Inteligencia Artificial con herramientas de Machine Learning, tales como las redes neuronales de Autoencoders.

Objetivos

- Implementar distintos modelos de autoencoders, cambiando entre ellos parámetros como la cantidad de capas o si hay o no concatenación entre ellas.

- Identificar métricas mundialmente aceptadas para la evaluación del pintado de imágenes realizado por los modelos implementados.

- Comparar los distintos modelos mediante las métricas halladas anteriormente, de tal manera que se pueda determinar cuál da mejores resultados.

- Realizar un análisis de los resultados obtenidos para así formular las conclusiones y definir un posible trabajo futuro.

Recolección De Datos:

El dataset utilizado fue encontrado en Kaggle, el cual contiene 7129 imágenes de calles, edificios, montañas, glaciares, árboles, etc. y su correspondiente imagen en escala de grises en dos carpetas diferentes.

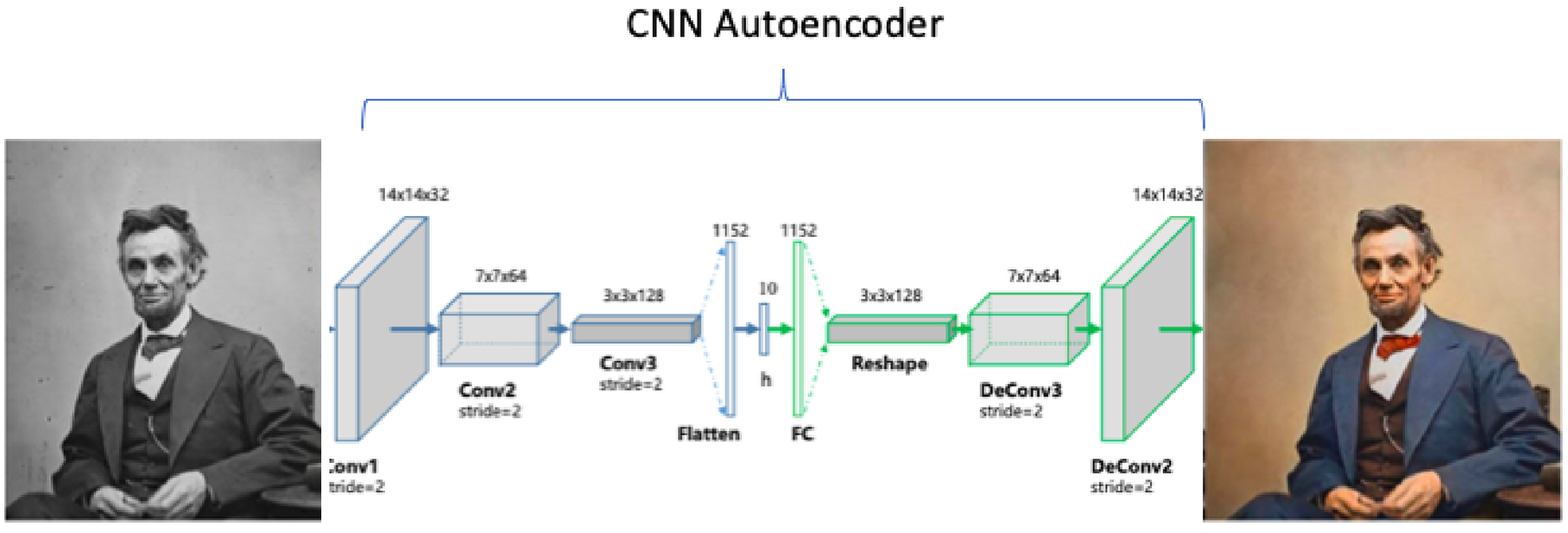
Tramiento De Datos:

Del total de imágenes sólo usaremos 3.000, las cuales hemos dividido en dos partes: las imágenes de entrenamiento constan de 2.500, mientras que las imágenes de prueba contienen 500. Después de dividir la matriz de imágenes, les hemos aplicado un reshape para que las imágenes sean de 160x160 píxeles.

```
train_gray_image = gray_img[:2500]
train_color_image = color_img[:2500]

test_gray_image = gray_img[2500:]
test_color_image = color_img[2500:]
# reshaping
train_g = np.reshape(train_gray_image,(len(train_gray_image),SIZE,SIZE,3))
train_c = np.reshape(train_color_image, (len(train_color_image),SIZE,SIZE,3))
print('Train color image shape:',train_c.shape)



test_gray_image = np.reshape(test_gray_image,(len(test_gray_image),SIZE,SIZE,3))
test_color_image = np.reshape(test_color_image, (len(test_color_image),SIZE,SIZE,3))
print('Test color image shape',test_color_image.shape)
```

```
Train color image shape: (2500, 160, 160, 3)
Test color image shape (500, 160, 160, 3)
```

Conceptualización del modelo



Conceptualización del modelo sin concatenar:

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 160, 160, 3)]     0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 80, 80, 128)       3584
_________________________________________________________________
leaky_re_lu_16 (LeakyReLU)   (None, 80, 80, 128)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 40, 40, 256)       295168
_________________________________________________________________
batch_normalization_5 (Batch (None, 40, 40, 256)       1024
_________________________________________________________________
leaky_re_lu_17 (LeakyReLU)   (None, 40, 40, 256)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 20, 20, 512)       1180160
_________________________________________________________________
batch_normalization_6 (Batch (None, 20, 20, 512)       2048
_________________________________________________________________
leaky_re_lu_18 (LeakyReLU)   (None, 20, 20, 512)       0
_________________________________________________________________
conv2d_transpose_8 (Conv2DTr (None, 40, 40, 256)       1179904
_________________________________________________________________
leaky_re_lu_19 (LeakyReLU)   (None, 40, 40, 256)       0
_________________________________________________________________
conv2d_transpose_9 (Conv2DTr (None, 80, 80, 128)       295040
_________________________________________________________________
leaky_re_lu_20 (LeakyReLU)   (None, 80, 80, 128)       0
_________________________________________________________________
conv2d_transpose_10 (Conv2DT (None, 160, 160, 3)       3459
_________________________________________________________________
leaky_re_lu_21 (LeakyReLU)   (None, 160, 160, 3)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 160, 160, 3)       39
=================================================================
Total params: 2,960,426
Trainable params: 2,958,890
Non-trainable params: 1,536
_________________________________________________________________
```

```
=================================================================
input_1 (InputLayer)         [(None, 160, 160, 3)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 80, 80, 128)       3584
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 80, 80, 128)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 40, 40, 128)       147584
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 40, 40, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 20, 256)       295168
_________________________________________________________________
batch_normalization (BatchNo (None, 20, 20, 256)       1024
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 20, 20, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 512)       1180160
_________________________________________________________________
batch_normalization_1 (Batch (None, 10, 10, 512)       2048
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 10, 10, 512)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 5, 5, 512)         2359808
_________________________________________________________________
batch_normalization_2 (Batch (None, 5, 5, 512)         2048
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 5, 5, 512)         0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 10, 10, 512)       2359808
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 10, 10, 512)       0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 20, 20, 256)       1179904
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 20, 20, 256)       0
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 40, 40, 128)       295040
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 40, 40, 128)       0
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 80, 80, 128)       147584
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 80, 80, 128)       0
_________________________________________________________________
conv2d_transpose_4 (Conv2DTr (None, 160, 160, 3)       3459
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 160, 160, 3)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 160, 160, 3)       39
=================================================================
Total params: 7,977,258
Trainable params: 7,974,698
Non-trainable params: 2,560
```

Conceptualización del modelo concatenado:

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 160, 160, 3)	0	
sequential_10 (Sequential)	(None, 80, 80, 128)	3584	input_2[0][0]
sequential_11 (Sequential)	(None, 40, 40, 128)	147584	sequential_10[0][0]
sequential_12 (Sequential)	(None, 20, 20, 256)	296192	sequential_11[0][0]
sequential_13 (Sequential)	(None, 10, 10, 512)	1182208	sequential_12[0][0]
sequential_14 (Sequential)	(None, 5, 5, 512)	2361856	sequential_13[0][0]
sequential_15 (Sequential)	(None, 10, 10, 512)	2359808	sequential_14[0][0]
concatenate_5 (Concatenate)	(None, 10, 10, 1024)	0	sequential_15[0][0] sequential_13[0][0]
sequential_16 (Sequential)	(None, 20, 20, 256)	2359552	concatenate_5[0][0]
concatenate_6 (Concatenate)	(None, 20, 20, 512)	0	sequential_16[0][0] sequential_12[0][0]
sequential_17 (Sequential)	(None, 40, 40, 128)	589952	concatenate_6[0][0]
concatenate_7 (Concatenate)	(None, 40, 40, 256)	0	sequential_17[0][0] sequential_11[0][0]
sequential_18 (Sequential)	(None, 80, 80, 128)	295040	concatenate_7[0][0]
concatenate_8 (Concatenate)	(None, 80, 80, 256)	0	sequential_18[0][0] sequential_10[0][0]
sequential_19 (Sequential)	(None, 160, 160, 3)	6915	concatenate_8[0][0]
concatenate_9 (Concatenate)	(None, 160, 160, 6)	0	sequential_19[0][0] input_2[0][0]
conv2d_11 (Conv2D)	(None, 160, 160, 3)	75	concatenate_9[0][0]

Total params: 9,602,766
Trainable params: 9,600,206
Non-trainable params: 2,560

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input_11 (InputLayer)	[(None, 160, 160, 3)	0	
sequential_90 (Sequential)	(None, 80, 80, 128)	3584	input_11[0][0]
sequential_91 (Sequential)	(None, 40, 40, 256)	296192	sequential_90[0][0]
sequential_92 (Sequential)	(None, 20, 20, 512)	1182208	sequential_91[0][0]
sequential_93 (Sequential)	(None, 40, 40, 256)	1179904	sequential_92[0][0]
concatenate_15 (Concatenate)	(None, 40, 40, 512)	0	sequential_93[0][0] sequential_91[0][0]
sequential_94 (Sequential)	(None, 80, 80, 128)	589952	concatenate_15[0][0]
concatenate_16 (Concatenate)	(None, 80, 80, 256)	0	sequential_94[0][0] sequential_90[0][0]
sequential_95 (Sequential)	(None, 160, 160, 3)	6915	concatenate_16[0][0]
concatenate_17 (Concatenate)	(None, 160, 160, 6)	0	sequential_95[0][0] input_11[0][0]
conv2d_76 (Conv2D)	(None, 160, 160, 3)	75	concatenate_17[0][0]

Total params: 3,258,830
Trainable params: 3,257,294
Non-trainable params: 1,536

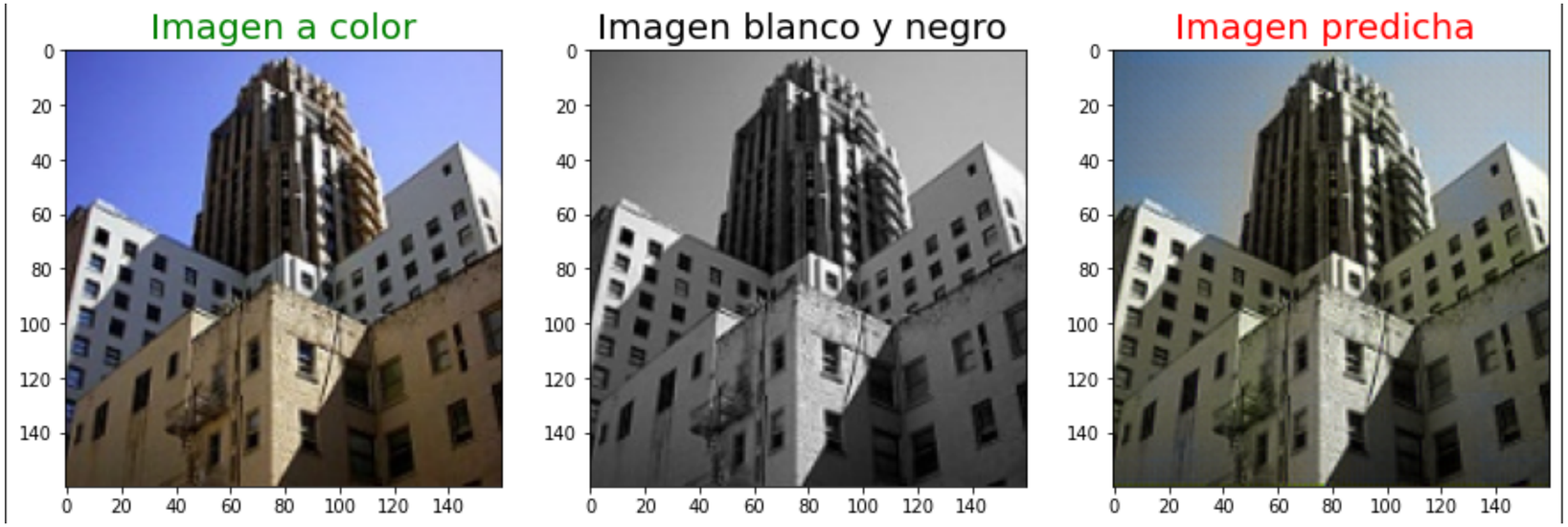
Parámetros de entrenamiento de los modelos

```
[ ]  checkpointer = tf.keras.callbacks.ModelCheckpoint(filepath="modelo-AE-concatM.h5",
                                    verbose=0,
                                    save_best_only=True)
     tensorboard = tf.keras.callbacks.TensorBoard(log_dir='./logs',
                                histogram_freq=0,
                                write_graph=True,
                                write_images=True)
```

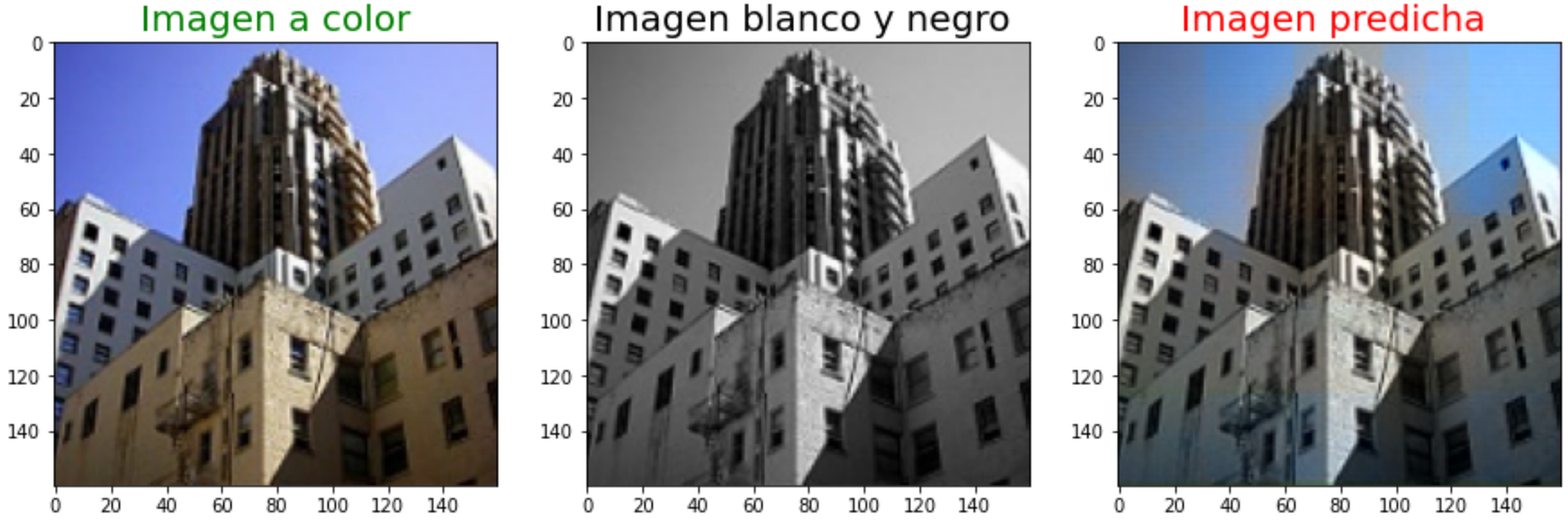
```
     autoencoder.compile(optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001), loss = 'mean_absolute_error',
                   metrics = ['accuracy'])

     history = autoencoder.fit(train_g, train_c, epochs = 50,validation_split=0.05,batch_size = 50,callbacks=[checkpointer, tensorboard],verbose = 1)
```

```
Epoch 1/50
48/48 [==============================] - 61s 587ms/step - loss: 0.1958 - accuracy: 0.3644 - val_loss: 0.1168 - val_accuracy: 0.3833
Epoch 2/50
48/48 [==============================] - 26s 534ms/step - loss: 0.0886 - accuracy: 0.3927 - val_loss: 0.1017 - val_accuracy: 0.3353
Epoch 3/50
48/48 [==============================] - 26s 536ms/step - loss: 0.0727 - accuracy: 0.4365 - val_loss: 0.0845 - val_accuracy: 0.3386
```

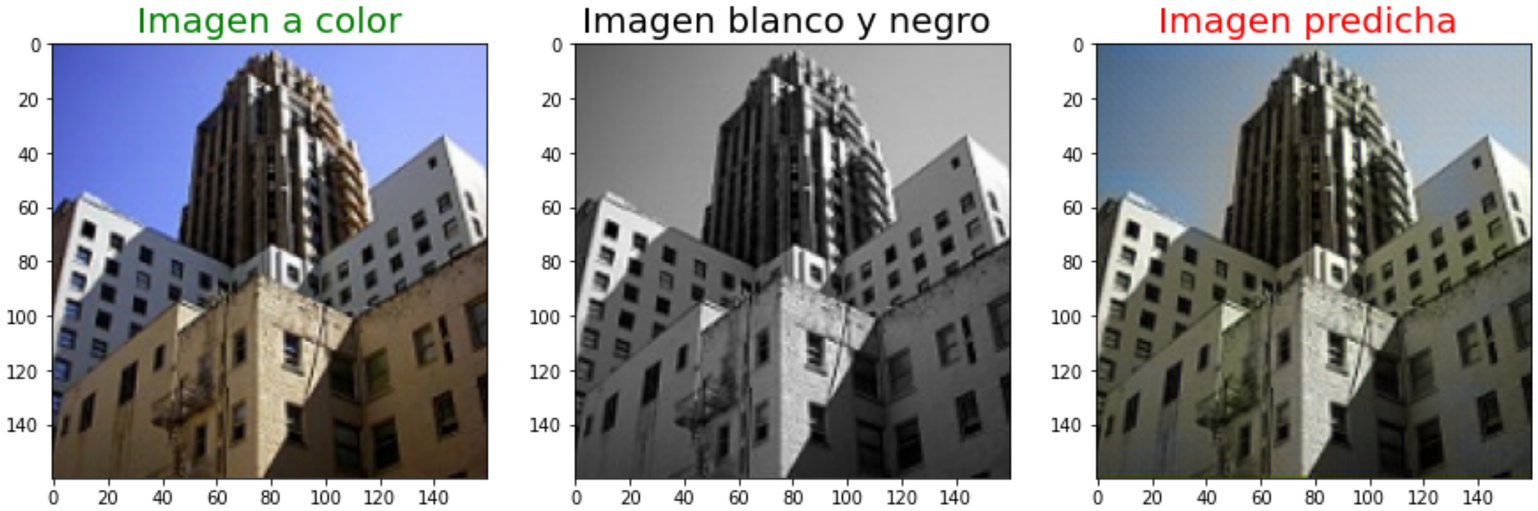
Autoencoder sin concatenar capas



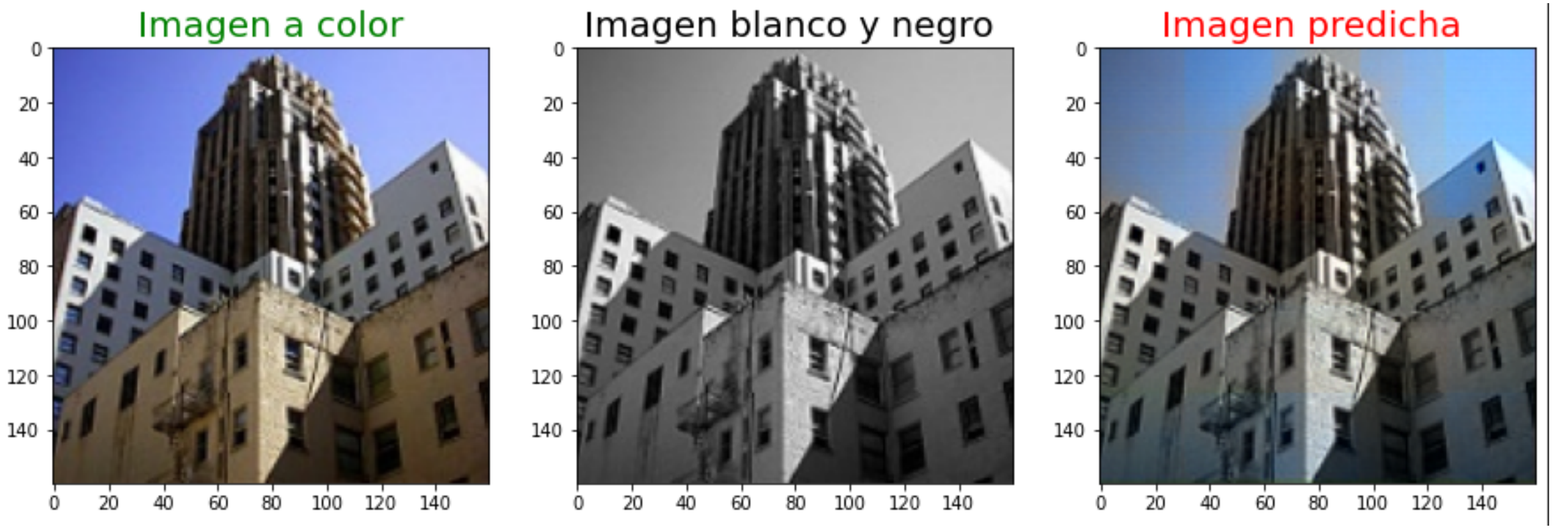
Autoencoder sin concatenar y con más capas



Autoencoder concatenando capas



Autoencoder concatenado y con más capas



Comparación de métricas

Metrica	AE sin concatenar	AE sin concatenar más capas	AE concatenado	AE concatenado más capas
MSE	0.007172	0.006462	0.006256	0.006086
RMSE	0.000019	0.000019	0.000017	0.000017
PSNR	95.039504	94.758902	96.088031	96.158390
FSIM	0.961650	0.929927	0.970350	0.968667
UQI	0.944186	0.946654	0.953323	0.951419
SSIM	0.999996	0.999998	0.999997	0.999997
VIF	0.980677	1.056835	0.976695	0.987062

Conclusiones:

1.Como se puede observar en los resultados obtenidos existen ciertas diferencias en las predicciones en los autoencoders dependiendo de la composición de los mismos en cuanto a parámetros y capas.

2.Haciendo un promedio entre todas las métricas podemos concluir de que el mejor autoencoder es el de “concatenado con más capas”.

3.Una de las diferencias más notorias son las predicciones de los autoencoders con capas concatenadas y sin capas concatenadas, esto se debe a que al no pasar la información de las imágenes entre las distintas capas, se ocasiona una pérdida de información y de calidad en la imagen. Por lo tanto, se recomienda realizar la concatenación y paso de información entre las distintas capas para mejores resultados.

4.Los autoencoders son una buena herramienta para esta tarea de coloración y al final dieron resultados excelentes en las diversas métricas.

Bibliografía:

- Dataset original. https://www.kaggle.com/theblackmamba31/landscape-image-colorization

- Bank, D., Koenigstein, N., and Giryes, R., “Autoencoders”, arXiv e-prints, 2020.

- Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful image colorization. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9907 LNCS, 649–666.

- Samajdar T., Quraishi M.I. (2015) Analysis and Evaluation of Image Quality Metrics. In: Mandal J., Satapathy S., Kumar Sanyal M., Sarkar P., Mukhopadhyay A. (eds) Information Systems Design and Intelligent Applications. Advances in Intelligent Systems and Computing, vol 340. Springer, New Delhi.