CIFAR - 10

PROYECTO ELECTIVO DATA SCIENCE

DESCRIPCIÓN BREVE

CIFAR-10 es un conjunto de datos de visión por computadora utilizado para el reconocimiento de objetos.

La finalidad de este proyecto es reconocer una imagen y predecir a cuál de las 10 clases de CIFAR-10 pertenece.

INTEGRANTES

Francisca Alejandra Hernández Piña

Marcelo Antonio Lazo Chávez

CONTENIDO

ÍNDICE DE FIGURAS

1. CONTEXTO DEL PROYECTO

CIFAR-10 es un conjunto de datos de visión por computadora utilizado para el reconocimiento de objetos.

Este conjunto de datos contiene 60,000 imágenes en color de 32x32 píxeles distribuidas en 10 clases de objetos, con 6,000 imágenes por clase, estas son:

- Aviones
- Automóviles
- Pájaros
- Gatos
- Venados
- Perros
- Ranas
- Caballos
- Barcos
- Camiones

Figura 1: Conjunto de datos CIFAR-10
Fuente: https://www.cs.toronto.edu/~kriz/cifar.html

De este total de 60,000 imágenes, 50,000 son para entrenamiento y 10,000 para testeo. Para este proyecto en específico, se quitó el 10% de imágenes de entrenamiento para hacer la validación, quedando la distribución de datos de la siguiente forma:

Figura 2: Distribución de imágenes para los modelos
Fuente: Elaboración propia

La finalidad de este proyecto es reconocer una imagen y predecir a cuál de las 10 clases de CIFAR-10 pertenece.

2. ANÁLISIS DE DISTRIBUCIÓN Y LIMPIEZA DE DATOS

Los datos que se analizarán se encuentran distribuidos en 6 archivos denominados *batches*, cada uno posee un conjunto de 10,000 imágenes, siendo uno de estos utilizado como información de testeo de los modelos. A continuación, se muestra una imagen con la distribución:

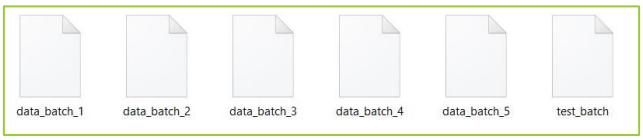

Figura 3: Distribución de datos CIFAR-10
Fuente: Elaboración propia

La dimensión de una imagen a color es de 32x32 píxeles y se compone de 3 canales (*Red*, *Green*, *Blue* = RGB) por cada píxel presente en ellas. Los valores de cada canal varían entre 0 a 255, lo que permite colorear cada píxel de la imagen. Considerando lo anterior, se tiene 32 x 32 x 3 = 3072 datos por imagen.

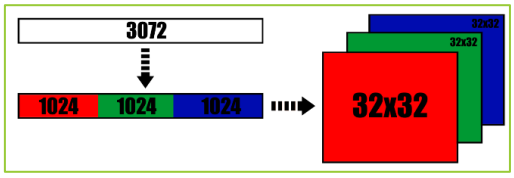

Figura 4: Composición de una imagen
Fuente: Elaboración propia

Dado a la composición de los datos que se utilizarán, no es necesario realizar una limpieza de datos exhaustiva como se ha hecho durante el curso. En este caso solo se realiza una normalización aplicando la fórmula vista en clases:

$$\bar{x}^{(j)} = \frac{x^{(j)} - min^{(j)}}{max^{(j)} - min^{(j)}}$$

Figura 5: Fórmula Normalización
Fuente: Apuntes Clases Data Science

A modo de ejemplificar, en las siguientes figuras se puede observar una imagen sin normalizar y una normalizada.

	0	1	2	3	4
0	59	62	63	43	46
1	77	125	99	76	142
2	105	69	136	105	74
3	16	20	20	0	0
4	37	101	67	35	105

Figura 7: Imagen sin normalizar
Fuente: Elaboración propia

	0	1	2	3	4
0	0.952381	0.956349	0.984127	0.952381	0.956349
1	0.972222	0.968254	0.980159	0.972222	0.972222
2	0.968254	0.988095	0.972222	0.968254	0.988095
3	0.964286	0.956349	0.984127	0.964286	0.956349
4	0.984127	0.976190	0.968254	0.988095	0.968254

Figura 6: Imagen normalizada
Fuente: Elaboración propia

La normalización se realiza para que todas las imágenes tengan una escala en común, además, permite que los modelos la trabajen como una distribución normal.

3. DESARROLLO Y TESTING DE LOS MODELOS

En este proyecto se utilizaron los siguientes modelos:

- Red neuronal convolucional
- PCA
- Random Forest
- Regresión Logística
- SVM
- KNN

Cabe mencionar qué, a excepción de la red neuronal convolucional, todos los modelos restantes utilizaron la técnica PCA para reducir la dimensionalidad del dataset e identificar estructuras internas en estos.

RED NEURONAL CONVOLUCIONAL (CNN)

Para crear una red neuronal convolucional se necesita hacer uso de la biblioteca de Machine Learning TensorFlow y de la biblioteca de redes neuronales Keras.

El formato de trabajo de la biblioteca Keras se basa en un modelo secuencial, es decir, una pila lineal de capas que se ejecutan de forma secuencial valga la redundancia. Además, se hace uso de las convoluciones, estas consisten en tomar "grupos de píxeles cercanos" a los cuales se les aplica producto escalar contra una matriz de dimensiones reducidas llamada *kernel* para generar una nueva matriz de salida, a este proceso se le llama **filtro**. Luego a estas matrices generadas se les aplica la función de activación **ReLu**, función que se encarga de transformar todos los números negativos a 0. Cabe señalar que se realizan múltiples **filtros** a cada una de las imágenes.

Finalmente, se aplica el proceso *subsampling* que reduce el tamaño de las imágenes filtradas, teniendo como objetivo dejar las características más importantes de cada filtro. Para este modelo se hace uso de *Max-Pooling* como método de *subsampling*.

Finalizada la etapa anterior, las capas convolucionales se transforman a capas de una red neuronal tradicional, en otras palabras, la matriz de *n x n x m* se transforma en un vector columna de n*n*m filas. Por último, el modelo se entrena como una red neuronal tradicional.

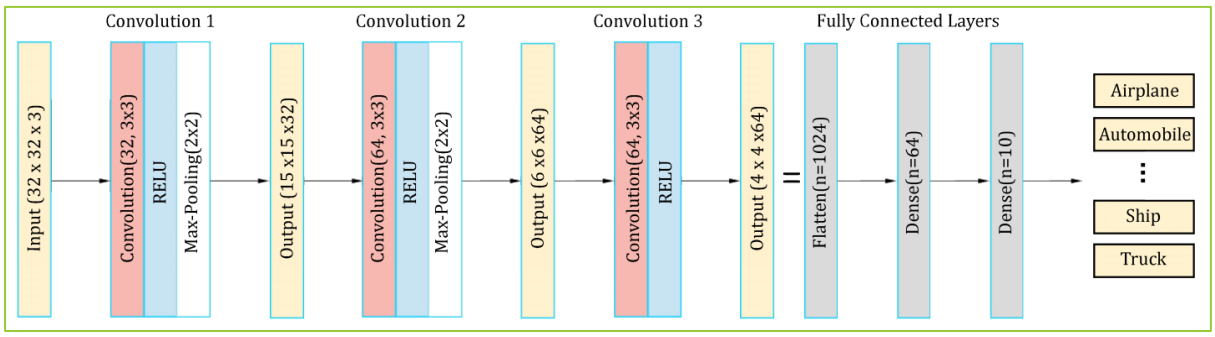

Figura 8: Convoluciones
Fuente: Elaboración propia

A continuación, se muestra el código asociado para la creación del modelo, su entrenamiento y resultados respectivos:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
history = model.fit(train_data, train_label, epochs=10,
                    validation_data=(validation_data, validation_label))
```

Figura 9: Definición y entrenamiento del modelo CNN
Fuente: Elaboración propia

```
[[720  26  66  26  24   7  18   8  67  38]
 [ 17 836   8  10   9   7  13   5  14  81]
 [ 55   7 637  51  57  78  70  20  10  15]
 [ 18  10  73 522  47 187  87  32   8  16]
 [ 23   4 106  90 563  58  87  59   8   2]
 [ 11   2  59 161  30 649  34  42   5   7]
 [  8   5  44  51  17  32 826   7   4   6]
 [ 10   3  35  59  62  94   9 704   5  19]
 [ 58  47  21  17   6  11   6   5 790  39]
 [ 17  66   9  17   7  11  11  18  19 825]]
```

Figura 10: Matriz de confusión CNN
Fuente: Elaboración propia

	precision	recall	f1-score	support
0	0.77	0.72	0.74	1000
1	0.83	0.84	0.83	1000
2	0.60	0.64	0.62	1000
3	0.52	0.52	0.52	1000
4	0.68	0.56	0.62	1000
5	0.57	0.65	0.61	1000
6	0.71	0.83	0.76	1000
7	0.78	0.70	0.74	1000
8	0.85	0.79	0.82	1000
9	0.79	0.82	0.81	1000
accuracy			0.71	10000
macro avg	0.71	0.71	0.71	10000
weighted avg	0.71	0.71	0.71	10000

Figura 11: Reporte de clasificación CNN
Fuente: Elaboración propia

APLICACIÓN DE PCA

Dado que el dataset tiene un total de 3072 columnas por imagen, se tomó la decisión de aplicar esta técnica para reducir sus dimensiones y posteriormente utilizar modelos más estándar. En palabras simples, se utilizó la técnica PCA para calcular la cantidad de componentes del modelo. Además, se buscó un k óptimo para que se tuviese una varianza aproximada de 99%, siendo este valor 658.

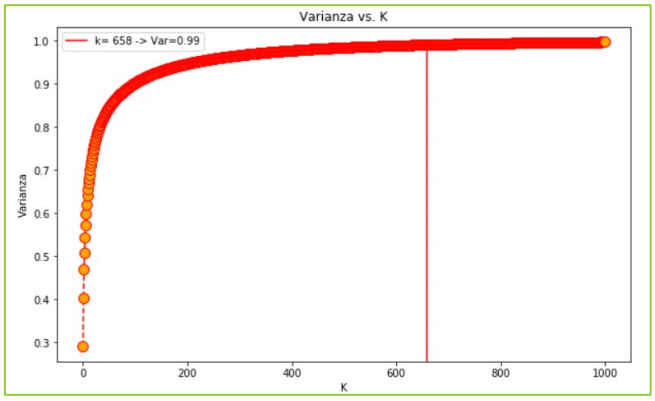

Figura 12: Gráfico Varianza vs K en PCA
Fuente: Elaboración propia

Finalmente, al momento de entrenar los modelos con los datos sin normalizar/normalizados, se observó que la variación en la precisión de los modelos no fue alta, por tanto, normalizando o no, no fue relevante para la predicción de imágenes de cada uno de los modelos.

RANDOM FOREST

Resultados del modelo Random Forest:

```
[[480  62  63  37  28  29  34  44 174  49]
 [ 43 530  22  44  19  25  38  44  65 170]
 [106  45 266  88 193  75  93  57  45  32]
 [ 68  69  81 245  78 185 107  60  46  61]
 [ 60  32 150  68 364  59 136  75  30  26]
 [ 44  60 106 177  82 306  71  78  46  30]
 [ 14  52 109  82 130  80 433  42  23  35]
 [ 56  70  69  90  94 105  54 314  43 105]
 [125  96  16  42  22  33  16  27 553  70]
 [ 58 231  21  45  14  28  36  63  87 417]]
```

	precision	recall	f1-score	support
0	0.46	0.48	0.47	1000
1	0.43	0.53	0.47	1000
2	0.29	0.27	0.28	1000
3	0.27	0.24	0.26	1000
4	0.36	0.36	0.36	1000
5	0.33	0.31	0.32	1000
6	0.43	0.43	0.43	1000
7	0.39	0.31	0.35	1000
8	0.50	0.55	0.52	1000
9	0.42	0.42	0.42	1000
accuracy			0.39	10000
macro avg	0.39	0.39	0.39	10000
weighted avg	0.39	0.39	0.39	10000

Figura 13: Matriz de confusión y reporte de clasificación Random Forest
Fuente: Elaboración propia

REGRESIÓN LOGÍSTICA

Resultados del modelo Regresión Logística:

```
[[491  52  61  32  17  28  23  59 169  68]
 [ 59 460  35  38  30  45  37  49  80 167]
 [ 97  45 282  88 116  83 136  82  46  25]
 [ 41  59  94 255  75 191 137  53  34  61]
 [ 55  19 140  68 290  94 169 112  28  25]
 [ 33  42  90 185  90 333  81  72  46  28]
 [ 20  32  78 130  98  96 460  42  17  27]
 [ 44  46  74  69  86  87  34 448  37  75]
 [146  70  24  32  13  35  12  21 539 108]
 [ 73 180  16  27  18  29  52  57  87 461]]
```

	precision	recall	f1-score	support
0	0.46	0.49	0.48	1000
1	0.46	0.46	0.46	1000
2	0.32	0.28	0.30	1000
3	0.28	0.26	0.27	1000
4	0.35	0.29	0.32	1000
5	0.33	0.33	0.33	1000
6	0.40	0.46	0.43	1000
7	0.45	0.45	0.45	1000
8	0.50	0.54	0.52	1000
9	0.44	0.46	0.45	1000
accuracy			0.40	10000
macro avg	0.40	0.40	0.40	10000
weighted avg	0.40	0.40	0.40	10000

Figura 14: Matriz de confusión y reporte de clasificación Regresión Logística
Fuente: Elaboración propia

SVM

Resultados del modelo Support Vector Machine:

```
[[524  63  55  28  20  12  15  27 150 106]
 [ 42 627   9  23  13  16  19  27  32 192]
 [ 83  44 356  88 123  75 101  53  27  50]
 [ 23  63  83 356  66 185  93  48  18  65]
 [ 34  25 155  84 379  76 101  91  25  30]
 [ 18  46  87 217  73 365  57  74  16  47]
 [ 10  63  81 105  78  54 520  31  10  48]
 [ 27  61  51  73  78  76  20 493  21 100]
 [113  91  26  26  11  28   9  11 604  81]
 [ 55 189   7  22   7  10  28  31  42 609]]
```

	precision	recall	f1-score	support
0	0.56	0.52	0.54	1000
1	0.49	0.63	0.55	1000
2	0.39	0.36	0.37	1000
3	0.35	0.36	0.35	1000
4	0.45	0.38	0.41	1000
5	0.41	0.36	0.38	1000
6	0.54	0.52	0.53	1000
7	0.56	0.49	0.52	1000
8	0.64	0.60	0.62	1000
9	0.46	0.61	0.52	1000
accuracy			0.48	10000
macro avg	0.48	0.48	0.48	10000
weighted avg	0.48	0.48	0.48	10000

Figura 15: Matriz de confusión y reporte de clasificación SVM
Fuente: Elaboración propia

KNN

Resultados del modelo k-Nearest Neighbor:

```
[[196   0  15   3  58   0   2   0 726   0]
 [123  16   7   4 112   0   4   0 734   0]
 [105   0  53   6 204   2   2   0 628   0]
 [ 96   0  14  16 172   1   1   0 700   0]
 [ 77   0  10   3 283   0   2   0 625   0]
 [111   0  22  11 174   2   1   0 679   0]
 [ 75   0  11  11 241   1  15   0 646   0]
 [122   0  24  12 237   1   3   2 599   0]
 [ 77   0   6   2  46   0   0   0 869   0]
 [ 86   0  18  10 142   1   1   0 741   1]]
```

	precision	recall	f1-score	support
0	0.18	0.20	0.19	1000
1	1.00	0.02	0.03	1000
2	0.29	0.05	0.09	1000
3	0.21	0.02	0.03	1000
4	0.17	0.28	0.21	1000
5	0.25	0.00	0.00	1000
6	0.48	0.01	0.03	1000
7	1.00	0.00	0.00	1000
8	0.13	0.87	0.22	1000
9	1.00	0.00	0.00	1000
accuracy			0.15	10000
macro avg	0.47	0.15	0.08	10000
weighted avg	0.47	0.15	0.08	10000

Figura 16: Matriz de confusión y reporte de clasificación kNN
Fuente: Elaboración propia

4. SELECCIÓN DEL MEJOR MODELO

Para este proyecto en específico, la métrica del modelo más relevante es la precisión, ya que solo importa si el modelo es preciso al momento de identificar una imagen.

Entonces, ¿qué modelo es el mejor para reconocer imágenes?

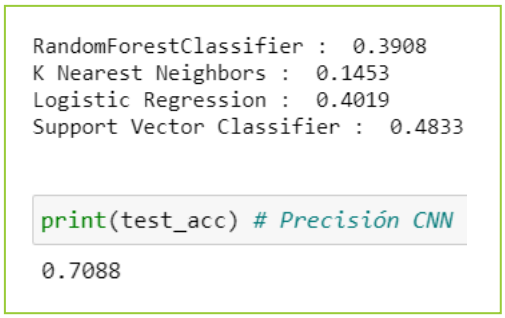

Figura 17: Comparación de precisión de modelos
Fuente: Elaboración propia

Tal como se aprecia en la figura anterior, al comparar los resultados de las métricas de cada modelo, se deduce que el mejor modelo aplicado son las redes neuronales convolucionales.

¿Por qué?

Porque tiene una precisión mucho mayor que el resto de los modelos la cual es del 70.88%.

5. APLICACIÓN Y CONCLUSIONES

Finalmente, se crea una función que extrae una imagen del dataset de prueba (indicando de 0 a 9999 por parámetro para elegir la imagen), la cual se entrega al modelo creado y predice a que clase pertenece.

En el siguiente ejemplo, se puede observar que el modelo encuentra cierto porcentaje de similitud de la imagen (caballo) con los elementos: caballo, perro, venado, gato y ave. El modelo asigna porcentajes de similitud a cada elemento, en este caso, predice que la imagen pertenece a un caballo ya que es el porcentaje más alto existente.

Figura 18: Selección de imagen de prueba
Fuente: Elaboración propia

Para mostrar las predicciones correctas/incorrectas del modelo, se opta por mostrar la predicción correspondiente (en las figuras aparece como “ganador”) y si esta respuesta es o no válida.

Figura 20: Predicción correcta del modelo
Fuente: Elaboración propia

Figura 19: Predicción incorrecta del modelo
Fuente: Elaboración propia