**Clasificador de imágenes basado en el algoritmo K-means (Vecinos más cercanos)**

Configuración del modelo, en esta etapa se configura el número de capas ocultas que tendrá el modelo, así como también la función de activación que en este caso es relu y softmax, cabe señalar que ambas funciones de activación trabajan con capas densas y son las más recomendadas para trabajar con imágenes.

model = keras.Sequential([

    keras.layers.Flatten(input\_shape=(28, 28)),

    keras.layers.Dense(128, activation='relu'),

    keras.layers.Dense(10, activation='softmax')

])

Compilación del modelo, se utiliza el algoritmo Adam la cual combina atributos de K-means y AdaGrap, a su vez se utiliza el argumento sparse\_categorical\_crossentropy, el cual sirve para calcular la función de pérdida del modelo, y por último se utiliza un atributo llamado accuracy para evaluar que tan exacto es el modelo.

model.compile(optimizer='adam',

              loss='sparse\_categorical\_crossentropy',

              metrics=['accuracy'])

Posteriormente se entrena el modelo, para este caso se determinó el número de épocas en 10, es decir 10 iteraciones que le servirán al modelo para entrenarse, este número mientras mayor sea mayor es el tiempo de ejecución y mejores resultados se obtendrán.

model.fit(train\_images, train\_labels, epochs=10)

La exactitud del modelo se determina con el siguiente código.

test\_loss, test\_acc = model.evaluate(test\_images,  test\_labels, verbose=2)

print('\nTest accuracy:', test\_acc)

Y el resultado es 0.87 es decir el modelo posee un 87% de exactitud con un total de 10 épocas o iteraciones:

313/313 - 1s - loss: 0.3415 - accuracy: 0.8793 - 518ms/epoch - 2ms/step

Test accuracy: 0.8792999982833862

La codificación complete en **gitlab**, y esta es la base para probar otros algoritmos que puedan clasificar imágenes con mayor precisión y en el menor tiempo posible, actualmente me encuentro desarrollando una nueva red neuronal utilizando arboles de decisión y red neuronales convolucionales. Cabe señalar que el código que se encuentra en gitlab se puede ejecutar de manera online en google collab, plataforma en la cual estoy trabajando actualmente el algoritmo mencionado.

La red neuronal se puede alojar en un servidor preferentemente que trabaje con JS, se utiliza una API de tensorflow para poder hacer el envio de información, el dia lunes 28 de marzo hare un informe más detallado de como instalar la API y como configurar el servidor y archivos que se necesiten para que la red neuronal se implemente de manera exitosa.