UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Curso: Física Computacional

Informe de Trabajo de Investigación Formativa

Docente:

DANNY GIANCARLO APAZA VELIZ

Estudiantes:

- Huarancca Leon, Rodrigo Alonso
- Panocca Merma, Jhon Franklin
- Paredes Mescco, Joaquin Gonzalo
- Tinta Aguilar, Miguel Ángel
- Zapana Romero, Pedro Luis Christian

Arequipa – Perú

2023

1. Título de investigación: Clasificación de Imágenes mediante Autómatas Celulares
2. Descripción de la realidad problemática: En la era digital actual, la clasificación precisa de imágenes es esencial en diversas aplicaciones, desde reconocimiento facial hasta diagnósticos médicos. Sin embargo, los métodos convencionales pueden enfrentar desafíos en la gestión eficiente de la complejidad y la adaptación a patrones dinámicos. Este proyecto aborda la necesidad de desarrollar un enfoque innovador utilizando autómatas celulares para mejorar la precisión y la eficiencia en la clasificación de imágenes.
    - *2.1. Problema general:* La falta de eficacia en los métodos tradicionales de clasificación de imágenes frente a la creciente complejidad y dinámica de los conjuntos de datos visuales.
    - *2.2. Problemas específicos:*
        - *2.2.1. Identificación de patrones de imágenes: Los métodos convencionales pueden tener dificultades para identificar patrones complejos y sutiles en las imágenes, especialmente aquellos que varían en términos de iluminación, posición y escala.*
        - *2.2.2. Optimización de la eficiencia computacional en la clasificación: Los métodos tradicionales pueden ser computacionalmente intensivos, lo que resulta en tiempos de procesamiento prolongados y recursos significativos.*

3. Objetivo general: *Evaluar la eficacia de la clasificación de imágenes mediante el uso de autómatas celulares, explorando su aplicabilidad y rendimiento en comparación con enfoques convencionales.*
    - *3.1. Objetivos específicos:*
        - *Objetivo 1: Revisar la literatura científica para comprender los enfoques actuales en la clasificación de imágenes, Identificar los desafíos y limitaciones asociados con los métodos existentes.*
        - *Objetivo 2: Implementar un prototipo de clasificador de imágenes mediante el uso de un lenguaje de programación de alto nivel y técnicas de procesamiento de imágenes*

- *Objetivo 3: Evaluar el rendimiento de procedimiento aplicados mediante el análisis de los resultados*

4. Justificación de la investigación:

   El uso de autómatas celulares ofrece un enfoque novedoso que puede superar limitaciones presentes en los métodos convencionales, así como fortalece los nuestro conocimiento sobre esta técnica haciendo uso de matemáticas y física. Además nos permite aplicar esta tecnología en conjunto con otras, en este caso técnicas de graficación y procesamiento de imágenes.

5. Hipótesis de investigación.
   - *5.1. Hipótesis alterna: El modelo de clasificación de imágenes basado en autómatas celulares supera significativamente en precisión y eficiencia a los métodos convencionales.*
   - *5.2. Hipótesis nula: No hay diferencia significativa en la precisión y eficiencia entre el modelo de clasificación de imágenes basado en autómatas celulares y los métodos convencionales.*

6. Variables.
   - *6.1. Variable independiente: Estudio de un modelo autómata celular*
   - *6.2. Variable dependiente: Precisión y eficacia en clasificación de imágenes*
   - *6.3. Operacionalización de las variables.*

     Indicadores Medibles:

     Tiempos de Ejecución de Programas de Procesamiento de imágenes y adquisición de parámetros: Incluye el programa que calcula los parámetros de las figuras geométricas necesarias para las reglas del autómata, la ejecución de este programa se medirá en segundos y dependerá de la complejidad de la imagen.

     Tiempo de Ejecución de los modelos de autómatas: Tiempo que demora el autómata en ejecutarse, este valor variará dependiendo a cuantas generaciones tendrá el programa de automatas.

     Escala de Medición:

     Todos los indicadores de medición serán en segundos.

7. Antecedentes de investigación:
   - 7.1. Artículo 1:

CLASIFICACIÓN DE IMÁGENES DE TEXTURA MEJORADA MEDIANTE EL USO DE UN AUTÓMATA CELULAR INSPIRADO EN LA CORROSIÓN [1]

En este artículo, se aborda el problema de clasificar imágenes de texturas sintéticas y naturales. Para enfrentar este desafío, se propone un método innovador que combina conceptos de modelado de corrosión y autómatas celulares para generar un descriptor de textura. Se identifican los procesos fundamentales de corrosión metálica (por picaduras) y se aplican a imágenes de texturas incorporando los mecanismos básicos de corrosión en la función de transición del autómata celular. Se analiza la morfología superficial de la imagen antes y durante la aplicación de la función de transición del autómata celular. En cada iteración, se obtiene la masa acumulativa del producto corroído para construir cada uno de los atributos del descriptor de textura. En un paso final, este descriptor de textura se utiliza para la clasificación de imágenes mediante la aplicación del Análisis Discriminante Lineal.

El método se probó en las conocidas bases de datos Brodatz y Vistex. Además, para verificar la robustez del método, se evaluó su invarianza al ruido y a la rotación. Para ello, se obtuvieron variantes diferentes de las dos bases de datos originales mediante la adición de ruido y la rotación de las imágenes. Los resultados mostraron que el descriptor de textura propuesto es efectivo para la clasificación de texturas, como se evidencia en las altas tasas de éxito en todos los casos. Esto indica el potencial de emplear métodos inspirados en fenómenos naturales en otros campos. En resumen, el enfoque propuesto demuestra ser eficaz y robusto para la clasificación de texturas, mostrando promisorios resultados en diversas condiciones.

7.2. Artículo 2:

AUTÓMATAS CELULARES PARA LA SEGMENTACIÓN Y CLASIFICACIÓN DE IMÁGENES MULTIESPECTRALES [2]

El artículo empieza dando una concepción general de los autómatas celulares a lo largo del tiempo, y luego presenta la evolución de este en

diferentes situaciones (supervisada y no supervisada). Se presenta el autómata celular básico de diferentes vecindades y luego de esto la segmentación de una imagen. De todo este artículo se resalta la parte del pseudocódigo para un autómata celular determinístico de segmentación de imágenes que es presentado por Vezhnevets en el año 2005 y a partir de este.

Finalmente utiliza este algoritmo bajo cierta sensibilidad y especificidad para una región geográfica del espacio llegando a la conclusión de que los autómatas celulares por segmentación resulta muy útil para tratar de manera detallada la clasificación de imágenes.

8. Bases teóricas: (Cellular Automata: A Discrete Universe" por Andrew Ilachinski.)

8.1. Que es un Autómata Celular:

Los autómatas celulares, como propuestos por Andrew Ilachinski en su obra seminal "Cellular Automata: A Discrete Universe" [3], representan una clase intrigante de modelos matemáticos y computacionales que ofrecen una perspectiva única sobre la dinámica de sistemas complejos. Este trabajo se centra en la definición y comprensión de los autómatas celulares tal como se presentan en la obra de Ilachinski, explorando sus elementos fundamentales y su impacto en diversos campos científicos.

Definición de Autómata Celular según Ilachinski

Ilachinski define los autómatas celulares como sistemas discretos compuestos por una red o cuadrícula de celdas, cada una de las cuales puede estar en uno de varios estados. Estas celdas interactúan entre sí de acuerdo con reglas locales simples y deterministas. La evolución temporal del sistema se lleva a cabo en pasos discretos, donde el estado futuro de una celda está determinado por el estado actual de esa celda y el estado de sus vecinos, de acuerdo con las reglas predefinidas [3].

Elementos Fundamentales de los Autómatas Celulares

En el texto de Ilachinski, se destacan elementos clave que definen los autómatas celulares:

Estructura de celdas: Una cuadrícula espacial donde cada celda tiene estados discretos.
Reglas de transición: Normas que definen cómo cambian los estados de las celdas en función de los estados de las celdas vecinas.
Discretización temporal: La evolución del sistema se produce en pasos discretos, lo que permite observar patrones emergentes a lo largo del tiempo [3].

8.2. Para qué Sirve

Modelado de Fenómenos Naturales:

- Física: Los autómatas celulares pueden ser utilizados para simular sistemas físicos, como la difusión de gases o fluidos, y para estudiar la formación de patrones en fenómenos naturales.
- Biología: Son útiles para modelar la dinámica de poblaciones, el crecimiento celular, la difusión de enfermedades, entre otros aspectos biológicos.

Computación y Simulación:

- Simulaciones Computacionales: Proporcionan un marco para simular y estudiar la dinámica de sistemas complejos, lo que ayuda a comprender mejor su comportamiento sin la necesidad de modelos analíticos complejos.
- Procesamiento Paralelo: La naturaleza distribuida de los autómatas celulares los hace adecuados para el procesamiento paralelo y la simulación en sistemas informáticos.

Comprensión de Sistemas Complejos:

- Emergencia de Patrones: Permiten observar cómo pequeñas reglas locales pueden dar lugar a patrones complejos y a menudo sorprendentes, lo que ayuda a comprender la emergencia de la complejidad a partir de interacciones simples.

- Teoría de la complejidad: Contribuyen a la teoría de la complejidad al proporcionar modelos que pueden ayudar a explicar la dinámica de sistemas complejos y sus propiedades emergentes.

Investigación Interdisciplinaria:

- Intersección de Disciplinas: Los autómatas celulares han sido utilizados en una variedad de campos, desde la física y la biología hasta la sociología y la economía, ofreciendo un marco común para entender los sistemas complejos en diferentes áreas.

Herramienta Educativa:

- Enseñanza y Aprendizaje: Los autómatas celulares pueden servir como herramientas educativas para ilustrar conceptos de dinámica de sistemas y para fomentar la comprensión de cómo los sistemas complejos pueden evolucionar a partir de reglas simples.

8.3. Cómo funciona:

El funcionamiento básico de un autómata celular, según [2] lo presentado en el libro, se puede explicar de la siguiente manera:

1) Estructura de Celdas:
   Cuadrícula espacial: Los autómatas celulares están compuestos por una cuadrícula o retícula de celdas, cada una con un estado particular.
2) Evolución Temporal:
   Pasos discretos: La evolución del autómata se produce en pasos discretos, donde cada paso representa un nuevo estado en el tiempo del sistema.
3) Reglas de Transición:
   Reglas locales: Las celdas interactúan entre sí siguiendo reglas locales simples y deterministas.
   Estado futuro: El estado futuro de una celda está determinado por su estado actual y el estado de sus vecinos inmediatos, de acuerdo con estas reglas de transición.
4) Dinámica Emergente:

Complejidad a partir de la simplicidad: A pesar de las reglas locales simples, los autómatas celulares pueden generar patrones y comportamientos complejos, algunos de los cuales pueden ser impredecibles o sorprendentes.

5) Variabilidad en Reglas:

Diversidad de comportamientos: Cambiar las reglas de transición puede dar lugar a comportamientos y patrones radicalmente diferentes en el autómata celular.

8.4. Ejemplos de Aplicaciones

Modelado de Propagación de Enfermedades

Los autómatas celulares pueden simular la propagación de enfermedades en poblaciones. Cada celda puede representar a un individuo y las reglas de transición podrían reflejar la probabilidad de infección basada en la cercanía a otros individuos infectados. Esto permite analizar y prever la propagación de enfermedades, la efectividad de medidas preventivas y estrategias de contención.

Simulación de Patrones en Ecología y Biología

En la ecología, los autómatas celulares se utilizan para modelar patrones de crecimiento de poblaciones, migraciones animales, distribución de recursos, y dinámicas de ecosistemas. Estos modelos permiten comprender cómo pequeñas interacciones locales entre especies pueden dar lugar a patrones complejos en niveles superiores de organización.

Dinámica de Fluidos y Modelado de Movimiento de Partículas

Los autómatas celulares pueden representar sistemas de fluidos y partículas en movimiento. Cada celda puede corresponder a una sección de fluido o una partícula, y las reglas de transición pueden simular la interacción entre estas entidades, lo que permite modelar fenómenos como la convección, la difusión de sustancias o la formación de vórtices.

Análisis de Tráfico y Movilidad Urbana

En el campo de la ingeniería de transporte, los autómatas celulares pueden simular el flujo de vehículos en carreteras y calles urbanas. Las celdas pueden representar secciones de carreteras y las reglas de transición reflejan el movimiento y la interacción entre vehículos, lo que facilita la comprensión de congestiones, patrones de tráfico y la eficacia de diseños viales.

Modelado de Fenómenos Físicos en Materiales

Para simular materiales sólidos, líquidos o incluso propiedades magnéticas, los autómatas celulares pueden ser útiles. Cada celda puede representar un átomo o una región de un material, y las reglas de transición describen la interacción entre estas regiones, lo que permite estudiar propiedades macroscópicas y fenómenos de materiales.

9. Marco conceptual.

En este marco conceptual, se presentará una descripción general de los autómatas celulares, su definición, elementos fundamentales y aplicaciones en diversos campos. Además, se establecerá la relación entre los autómatas celulares y la clasificación de imágenes, el problema a abordar en esta investigación.

9.1. Autómatas celulares:

Los autómatas celulares son modelos matemáticos y computacionales compuestos por una cuadrícula de celdas, cada una con estados discretos. La evolución temporal se realiza en pasos discretos, donde las reglas locales determinan el cambio de estado de una celda según el estado de sus vecinas [3].

9.2. Elementos Fundamentales de los Autómatas Celulares:

Según Ilachinski [1], los autómatas celulares son sistemas discretos compuestos por una cuadrícula de celdas que evolucionan en pasos discretos, con reglas locales deterministas que rigen la transición de estados.

9.3. Aplicabilidad en Clasificación de Imágenes:

La aplicación de autómatas celulares en la clasificación de imágenes representa una estrategia innovadora que busca modelar patrones

complejos y dinámicos presentes en conjuntos de datos visuales [2]. Este enfoque no solo aborda la tarea de clasificación, sino que también permite la captura eficiente de información rica en imágenes.

9.4. Modelado de Fenómenos Naturales:

Ilachinski [1] proporciona una base teórica sólida para entender cómo los autómatas celulares pueden modelar fenómenos naturales. Desde la física hasta la biología, estos modelos ofrecen una representación matemática de la dinámica de sistemas complejos, estableciendo así conexiones profundas con la clasificación de imágenes.

9.5. Clasificación de imágenes usando openCV de python

Este proceso hace uso de diversas técnicas y algoritmos como por ejemplo la conversión a escala de grises, detección de bordes mediante el algoritmo de Canny y la aproximación para detectar líneas de contorno de imagenes y clasificarlas como figuras geométricas, todo con el objetivo de obtener parámetros necesarios para que los autómatas celulares puedan replicar la imagen para clasificarla.

9.6. Hipótesis y suposiciones fundamentales.

La hipótesis alterna, propone que el modelo de clasificación basado en autómatas celulares supera significativamente en precisión y eficiencia a los métodos convencionales [2].

Esta hipótesis se considera incorrecta, primero porque al estudiar la naturaleza del autómata celular, se llega a la conclusión de que este es de naturaleza constructiva y no analítica.

La hipótesis nula, por otro lado, plantea la ausencia de diferencia significativa, una afirmación que será evaluada y confirmada o refutada en el estudio específico.

Esta hipótesis se concreta como correcta, debido a que los resultados obtenidos así como los parámetros necesarios para que un autómata celular clasifique una imagen, requieren de más pasos adicionales,

además tampoco es eficiente porque el autómata requiere de múltiples generaciones para llegar un resultado.

10. Modelo de Investigación: Investigación Explicativa o causal

    Este estudio adopta un enfoque de investigación explicativa o causal, buscando no solo describir la relación entre el uso de autómatas celulares y la clasificación de imágenes, sino también comprender las causas y efectos subyacentes. Se explorarán variables específicas para identificar los mecanismos que explican la mejora en la precisión y eficiencia de la clasificación de imágenes al emplear autómatas celulares, permitiendo así establecer relaciones causales claras.

11. Método de investigación.

    11.1. Enfoque de investigación : Enfoque Cualitativo

Dada la naturaleza de la investigación, inicialmente sería describir de manera puntual los autómatas celulares pero luego se realiza y de manera más profunda un enfoque cualitativo debido a que se realizará la observación de un fenómeno en el entorno natural. Se trata de datos que son difíciles de cuantificar.

11.2. Alcance de investigación: Alcance Explicativo: Limitado a el análisis y comparación de técnicas.

11.3. Diseño de investigación: Experimental y Descriptivo

Para esta investigación se hizo una relación entre causa y efecto, es decir que observaremos las variaciones que puede tener la ejecución del programa en donde la variable principal serán las imágenes y observar cómo se dan los cambios para poder describir de mejor manera el fenómeno.

12. Población y tamaño de muestra.

La muestra utilizada para este proyecto consta de una imagen en formato JPG de dimensiones : ancho 802 píxeles y alto 544 pixeles a una resolución de 72pp[8]

13. Técnicas e instrumentos.

- Python: Se trata de un lenguaje orientado a objetos, fácil de interpretar y con una sintaxis que permite leerlo de manera semejante a como se lee el inglés. Es un lenguaje interpretado, esto significa que el código de programación se convierte en bytecode y luego se ejecuta por el intérprete, que, en este caso, es la máquina virtual de Python. Para este proyecto será usado como fuente principal para la elaboración del código fuente para los autómatas celulares.[5]
- OpenCV: OpenCV (Open Source Computer Vision Library) es una biblioteca de funciones de programación principalmente para la visión de la computadora en tiempo real. Dentro de la investigación, se requiere clasificar imágenes por lo que la librería de openCV será de mucha utilidad por las funciones y características que nos puede ofrecer al momento de tratar imágenes.[6]
- numpy: es una biblioteca para el lenguaje de programación Python que da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.Para poder tratar las imágenes en openCV se realiza mediante matrices, por lo que es necesario tener una biblioteca especializada en matrices como los es numpy.[7]
- Google Colaboratory: Es un servicio de computación en la nube basado en cuadernos de Jupyter ofrecidos por Google. Permite a los usuarios acceder a recursos gratuitos de CPUs y GPUs ideales para el procesamiento de datos usando python (12.7 GB de RAM y 107.7 para la versión gratuita). Además este servicio se ofrece con un plan PRO que elimina las limitaciones de almacenamiento temporal y de recursos que ofrece la versión gratuita. [8]

14. Procesamiento y análisis de datos.

14.1. Procesamiento de Imágenes usando openCV: El objetivo principal de esta actividad es extraer características claves en muestras de imágenes que sean útiles para la creación de autómatas, que serán capaces de replicar la imagen original.

Los valores principales que se buscan fueron características de círculos, como el radio y la posición de su centro con respecto a el origen, también

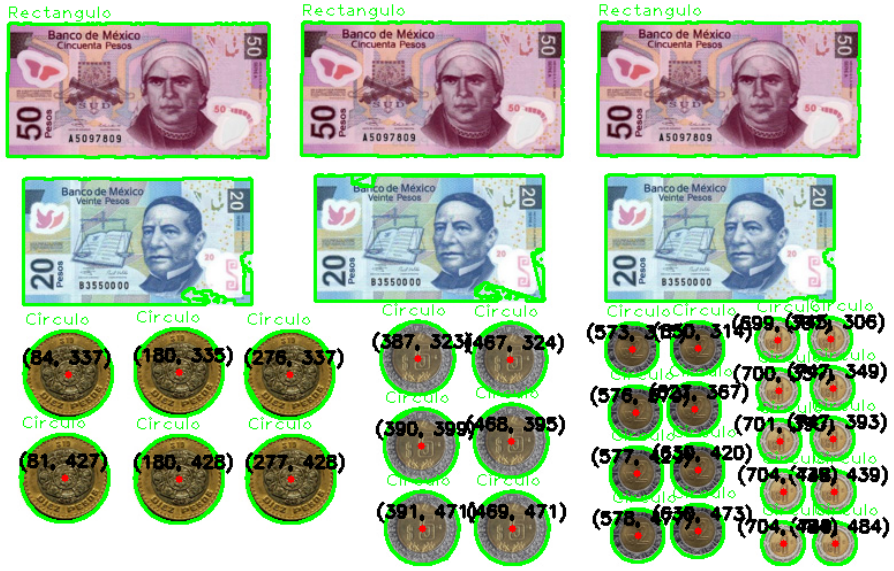
se requieren información para la construcción rectángulos, por lo cual se necesita de valores de ancho como de alto, así como un centroide para poder ubicarse en el plano.

Este proceso se realiza de la siguiente manera: se analiza la imagen usando openCV con los procesos recomendados en su documentación[5]: La conversión a escala de grises, aplicación del algoritmo de Canny para detectar los bordes y contornos de imagen, dibujar los contornos de la imagen.
Mediante este proceso se obtendrán listas con información necesaria para que los autómatas puedan desarrollarse en formas circulares y rectangulares.

```python
#Importar bibliotecas necesarias
import numpy as np
from google.colab.patches import cv2_imshow  #Para mostrar imágenes en Google Colab
import cv2  #OpenCV para procesamiento de imágenes

#Cargar la imagen desde un archivo
image = cv2.imread('/content/uploads_recortable-billetes-y-monedas.jpg')

#Obtener dimensiones de la imagen
height, width, channels = image.shape

#Imprimir las dimensiones de la imagen
#print(f"Dimensiones de la imagen: Altura = {height}, Ancho = {width}, Canales = {channels}")

#Convertir la imagen a escala de grises
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#Aplicar detección de bordes con el algoritmo de Canny
canny = cv2.Canny(gray, 10, 150)
canny = cv2.dilate(canny, None, iterations=1)  #Dilatar los bordes
canny = cv2.erode(canny, None, iterations=1)   #Erosionar para eliminar pequeños detalles

#Encontrar contornos en la imagen
```

```
cnts, _ = cv2.findContours(canny, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

#Dibujar los contornos en la imagen original
cv2.drawContours(image, cnts, -1, (0, 255, 0), 2)

#Dimensiones de la imagen
height, width = image.shape[:2]

#Agregar un eje de coordenadas a la esquina inferior izquierda
#cv2.line(image, (0, height), (width, height), (255, 0, 0), 1)  # Eje X
#cv2.line(image, (0, 0), (0, height), (255, 0, 0), 1)  # Eje Y

#Inicializar listas para almacenar información sobre círculos y rectángulos
circles = []
rectangles = []

#Iterar a través de los contornos encontrados
for c in cnts:
    #Aproximar el contorno segun vertices
    epsilon = 0.01 * cv2.arcLength(c, True) ## se calula el perimetro del
contorno
    approx = cv2.approxPolyDP(c, epsilon, True) #se aproxima curvas de cada
contorno, el tercer parametro confirma que es un poligono cerrado.
    x, y, w, h = cv2.boundingRect(approx) #se usa x,y para la esquina
superior de los rectangulos
    ##se usa w y h como altura y ancho del rectangulo


    if len(approx) == 3: ##Si el poligono tiene 3 vertices
      ##Parametros usados para identificar el triangulo en la imagen:
        #Texto que se agrega, coordenadas de texto, fuente,etc.
        cv2.putText(image, 'Triangulo', (x, y - 5), 1, 1, (0, 255, 0), 1)

    if len(approx) == 4:
        aspect_ratio = float(w) / h
        print('aspect_ratio= ', aspect_ratio)
        if aspect_ratio == 1:
          ##Parametros usados para identificar el cuadrado en la imagen:
        #Texto que se agrega, coordenadas de texto, fuente,etc.
            cv2.putText(image, 'Cuadrado', (x, y - 5), 1, 1, (0, 255, 0), 1)
            ##Se agrega las coordenadas del cuadrado a la lista rectangulos
            rectangles.append({'x': x, 'y': y, 'width': w, 'height': h})

        else:
          ##Parametros usados para identificar el rectangulo en la imagen:
        #Texto que se agrega, coordenadas de texto, fuente,etc.
```

```
            cv2.putText(image, 'Rectangulo', (x, y - 5), 1, 1, (0, 255, 0),
1)
            ##Se agrega las coordenadas del cuadrado a la lista rectangulos
            rectangles.append({'x': x, 'y': y, 'width': w, 'height': h})

    if len(approx) > 10:  #Si el poligono tiene mas de 10 vertices se
considera circulo
       ##Datos del circulo
        area = cv2.contourArea(c)
        perimeter = cv2.arcLength(c, True)
        ##Ajuste del circulo para poder incluirlo a la lista
        circularity = 4 * np.pi * (area / (perimeter * perimeter))
        if circularity > 0.5:
            cv2.putText(image, 'Circulo', (x, y - 5), 1, 1, (0, 255, 0), 1)
            #Agregar información del círculo a la lista
            center_x = int(x + w / 2)
            center_y = int(y + h / 2)
            radius = int(max(w, h) / 2)
            circles.append({'center_x': center_x, 'center_y': center_y,
'radius': radius})

        #Dibujar el contorno y el centro del objeto para referencias
        cv2.drawContours(image, [approx], 0, (0, 255, 0), 2)
        cv2.putText(image, f'({center_x}, {center_y})', (center_x - 40,
center_y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                    (0, 0, 0), 2)
        cv2.circle(image, (center_x, center_y), 3, (0, 0, 255), -1)

# Mostrar la imagen con los contornos, el eje de coordenadas y las
coordenadas de los centros
cv2_imshow(image)

# Imprimir la información sobre círculos y rectángulos
print("Circles:", circles)
print("Rectangles:", rectangles)
```



Como resultado obtendremos una imagen analizada, pero lo más importante dos listas que serán usadas como parte de las reglas de los Autómatas celulares.
Tiempo de Ejecución:

```
✓ 0 s  completado a las 18:56
```

14.2. Creación de los Autómatas celulares que replicarán la imagen:

14.2.1. Modelo 1

```python
import numpy as np
import matplotlib.pyplot as plt

# Tamaño del grid
```

```
filas = 800
columnas = 800

# Crear una matriz vacía para representar el grid
grid = np.zeros((filas, columnas))

# Función para mostrar el grid con círculos
def mostrar_grid(grid, circulos):
  # Mostrar el grid en blanco y negro
  plt.imshow(grid, cmap='binary')

  # Dibujar cada círculo en rojo como contorno
  for circulo in circulos:
    plt.gca().add_patch(plt.Circle((circulo['center_x'],
circulo['center_y']), circulo['radius'], color='red', fill=False))

  # Eliminar los ejes del gráfico
  plt.xticks([])
  plt.yticks([])

  # Mostrar el gráfico
  plt.show()

# Función para crear círculos en el grid
def crear_circulo(grid, centro, radio):
  # Crear una máscara circular y asignar valores de 1 a los píxeles dentro
del círculo
  y, x = np.ogrid[-centro[0]:grid.shape[0]-centro[0],
-centro[1]:grid.shape[1]-centro[1]]
  mascara = x*x + y*y <= radio*radio
  grid[mascara] = 1

# Función para crear rectángulos en el grid
def llenar_rectangulo(grid, rectangulo):
  # Asignar valores de 1 a los píxeles dentro del rectángulo
  x, y = rectangulo['x'], rectangulo['y']
  ancho, alto = rectangulo['width'], rectangulo['height']
  grid[y:y+alto, x:x+ancho] = 1

# Función para actualizar el grid en cada paso de tiempo
def actualizar_grid(grid):
  # Crear un nuevo grid para almacenar los valores actualizados
  nuevo_grid = np.zeros_like(grid)
  for i in range(1, filas - 1):
    for j in range(1, columnas - 1):
      # Sumar los valores de los vecinos
      suma_vecinos = (
        grid[i - 1, j - 1] + grid[i - 1, j] + grid[i - 1, j + 1] +
        grid[i, j - 1] + grid[i, j + 1] +
```

```
        grid[i + 1, j - 1] + grid[i + 1, j] + grid[i + 1, j + 1]
      )
      # Aplicar reglas del autómata celular
      if suma_vecinos >= 3:
        nuevo_grid[i, j] = 1
      else:
        nuevo_grid[i, j] = 0

  return nuevo_grid

# Lista de círculos y rectángulos proporcionados
circulos = [
  {'center_x': 749, 'center_y': 484, 'radius': 19},
  {'center_x': 704, 'center_y': 484, 'radius': 19},
  {'center_x': 578, 'center_y': 477, 'radius': 23},
  {'center_x': 630, 'center_y': 473, 'radius': 23},
  {'center_x': 469, 'center_y': 471, 'radius': 32},
  {'center_x': 391, 'center_y': 471, 'radius': 32},
  {'center_x': 748, 'center_y': 439, 'radius': 19},
  {'center_x': 704, 'center_y': 439, 'radius': 19},
  {'center_x': 577, 'center_y': 423, 'radius': 23},
  {'center_x': 630, 'center_y': 420, 'radius': 23},
  {'center_x': 277, 'center_y': 428, 'radius': 38},
  {'center_x': 180, 'center_y': 428, 'radius': 38},
  {'center_x': 81, 'center_y': 427, 'radius': 38},
  {'center_x': 701, 'center_y': 395, 'radius': 19},
  {'center_x': 747, 'center_y': 393, 'radius': 19},
  {'center_x': 390, 'center_y': 399, 'radius': 32},
  {'center_x': 468, 'center_y': 395, 'radius': 32},
  {'center_x': 576, 'center_y': 370, 'radius': 23},
  {'center_x': 627, 'center_y': 367, 'radius': 23},
  {'center_x': 700, 'center_y': 351, 'radius': 19},
  {'center_x': 747, 'center_y': 349, 'radius': 19},
  {'center_x': 276, 'center_y': 337, 'radius': 38},
  {'center_x': 84, 'center_y': 337, 'radius': 38},
  {'center_x': 180, 'center_y': 335, 'radius': 38},
  {'center_x': 573, 'center_y': 315, 'radius': 23},
  {'center_x': 630, 'center_y': 314, 'radius': 23},
  {'center_x': 467, 'center_y': 324, 'radius': 32},
  {'center_x': 387, 'center_y': 323, 'radius': 32},
  {'center_x': 699, 'center_y': 307, 'radius': 19},
  {'center_x': 745, 'center_y': 306, 'radius': 19}


]

rectangulos = [
```

```
  {'x': 31, 'y': 32, 'width': 228, 'height': 116},
  {'x': 543, 'y': 30, 'width': 227, 'height': 117},
  {'x': 286, 'y': 30, 'width': 227, 'height': 117}

]

# Crear círculos y rectángulos en el grid
for circulo in circulos:
  crear_circulo(grid, (circulo['center_y'], circulo['center_x']),
circulo['radius'])

for rectangulo in rectangulos:
  llenar_rectangulo(grid, rectangulo)

# Ejecutar la simulación con solo 3 generación
generaciones = 3
for _ in range(generaciones):
  grid = actualizar_grid(grid)

# Mostrar el grid con círculos
mostrar_grid(grid, circulos)
```

Definición del tamaño del grid:
Se establece el tamaño del grid en 800 filas y 800 columnas.

Creación de un grid vacío:
Se crea una matriz llena de ceros para representar el grid.

Función para mostrar el grid con círculos:
mostrar_grid toma el grid y una lista de círculos como entrada. Grafica el grid en blanco y negro con círculos dibujados como contornos rojos.

Función para crear círculos:
crear_circulo llena el grid con valores de 1 para los píxeles dentro de un círculo dado por su centro y radio.

Función para llenar rectángulos:
llenar_rectangulo asigna valores de 1 a los píxeles dentro de un rectángulo, definido por su posición inicial, ancho y alto.

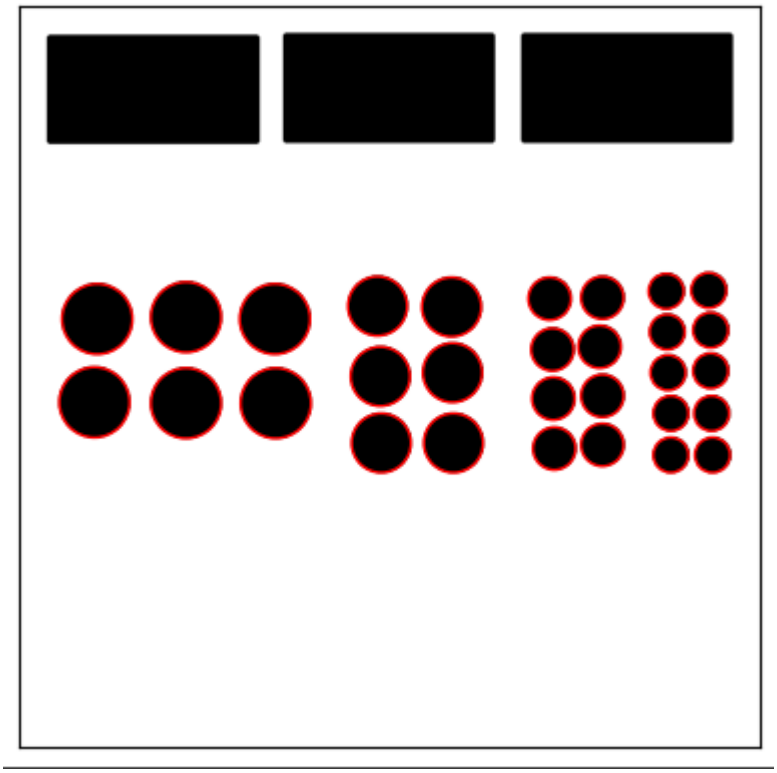
Función para actualizar el grid:
actualizar_grid aplica reglas de un autómata celular para cada celda del grid y genera un nuevo grid con los valores actualizados.

Definición de círculos y rectángulos:
Se definen las posiciones y tamaños de los círculos y rectángulos que se agregarán al grid.

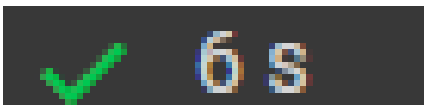
Creación de círculos y rectángulos en el grid:
Se utilizan las funciones crear_circulo y llenar_rectangulo para dibujar los círculos y rectángulos en el grid.

Ejecución de la simulación:
Se actualiza el grid una sola vez utilizando la función actualizar_grid. En este caso, la simulación se ejecuta solo una vez (generaciones = 1).

Mostrar el grid con círculos:
Se utiliza la función mostrar_grid para visualizar el grid resultante con los círculos dibujados.



Resultado de consola: El programa se ejecutó con 3 generaciones de autómatas y se demoró 6 segundos



Modelo 2

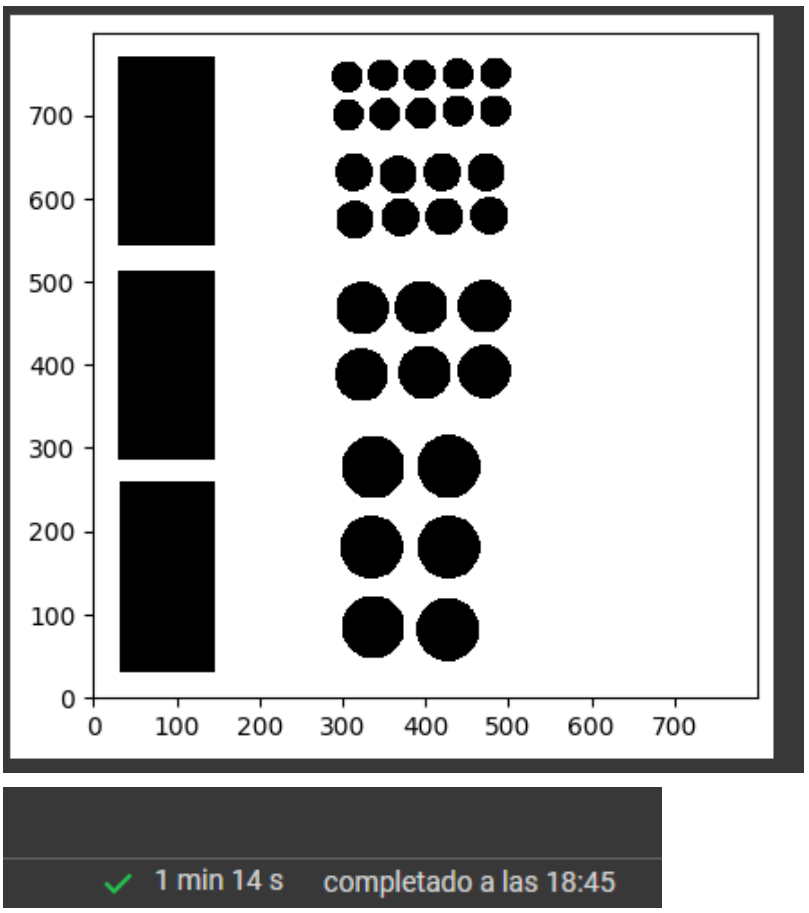
Este modelo usará los valores obtenidos del programa de analisis de imagenes y calculara el estado del autómata, teniendo en cuenta si estos estan dentro de los limites del circulo (distancia y radio) o del rectangulo(area del rectangulo)

```python
##Librerias necesarias
```

```
import numpy as np
import matplotlib.pyplot as plt


def initialize_grid(size):
  #creando la matriz con 0
  return np.zeros((size, size), dtype=int)

  ##Visualizacion de la matriz
def display_grid(grid):
  plt.imshow(grid, cmap='binary', interpolation='nearest') #Esquma de colores
binario blanco y negro por defecto
  plt.gca().invert_yaxis() #invertir el eje y puesto que Matplotlib por
defecto considera el area de trabajo a el 4to cuadrante
  plt.show()
#determina que la celula en i j debe estar viva en la proxima generacion
def reglas_circulos(grid, i, j, circles):

  #La distancia se calcula  de la forma euclidiana para la ubiacion del
centro: raiz cuadrada (x1-x2)**2 + (y1-y2)**2
  distance_to_centers = np.sqrt((i - np.array([circle['center_x'] for circle in
circles]))**2 +
                (j - np.array([circle['center_y'] for circle in circles]))**2)

  #La funcion devolvera true si las celulas deben estar cerca del centro,
usando el radio.
  return np.any(distance_to_centers < np.array([circle['radius'] for circle in
circles]))

 ##De la misma manera que los circulos, pero en lugar de centro se escoge la
esquina superior izquierda(x,y) y ancho y largo
def reglas_rectangulos(grid, i, j, rectangles):

  for rectangle in rectangles:
   #si la celula esta dentro del rectangulo , sigue viva y de vuelve true
    if rectangle['x'] <= i < rectangle['x'] + rectangle['width'] and \
      rectangle['y'] <= j < rectangle['y'] + rectangle['height']:
       return True
  return False

 ##Generacion de automatas.
def generate_ca(size, generations, circles, rectangles):
  grid = initialize_grid(size) ##Crea una cuadricula del tamaño requerido
  all_generations = [grid.copy()] ##Lista que almacenara todas las
generaciones del automata

  for _ in range(generations): ##Itera por las generaciones y la cuadricula
    new_grid = np.zeros_like(grid)
    for i in range(size):
```

```
      for j in range(size):
       ##Si se cumple una de las reglas para circulo o rectangulo la celda se
asigna a 1, es decir viva
        if reglas_circulos(grid, i, j, circles) or reglas_rectangulos(grid, i, j,
rectangles):
          new_grid[i, j] = 1
    grid = new_grid.copy()
    all_generations.append(grid.copy())

  return all_generations


size = 800
generations = 3
##Variables obtenidas del programa anterior
circles = [{'center_x': 749, 'center_y': 484, 'radius': 19}, {'center_x': 704,
'center_y': 484, 'radius': 19}, {'center_x': 578, 'center_y': 477, 'radius': 23},
{'center_x': 630, 'center_y': 473, 'radius': 23}, {'center_x': 469, 'center_y':
471, 'radius': 32}, {'center_x': 391, 'center_y': 471, 'radius': 32},
{'center_x': 748, 'center_y': 439, 'radius': 19}, {'center_x': 704, 'center_y':
439, 'radius': 19}, {'center_x': 577, 'center_y': 423, 'radius': 23},
{'center_x': 630, 'center_y': 420, 'radius': 23}, {'center_x': 277, 'center_y':
428, 'radius': 38}, {'center_x': 180, 'center_y': 428, 'radius': 38},
{'center_x': 81, 'center_y': 427, 'radius': 38}, {'center_x': 701, 'center_y':
395, 'radius': 19}, {'center_x': 747, 'center_y': 393, 'radius': 19},
{'center_x': 390, 'center_y': 399, 'radius': 32}, {'center_x': 468, 'center_y':
395, 'radius': 32}, {'center_x': 576, 'center_y': 370, 'radius': 23},
{'center_x': 627, 'center_y': 367, 'radius': 23}, {'center_x': 700, 'center_y':
351, 'radius': 19}, {'center_x': 747, 'center_y': 349, 'radius': 19},
{'center_x': 276, 'center_y': 337, 'radius': 38}, {'center_x': 84, 'center_y':
337, 'radius': 38}, {'center_x': 180, 'center_y': 335, 'radius': 38},
{'center_x': 573, 'center_y': 315, 'radius': 23}, {'center_x': 630, 'center_y':
314, 'radius': 23}, {'center_x': 467, 'center_y': 324, 'radius': 32},
{'center_x': 387, 'center_y': 323, 'radius': 32}, {'center_x': 699, 'center_y':
307, 'radius': 19}, {'center_x': 745, 'center_y': 306, 'radius': 19}]

rectangles = [
  {'x': 31, 'y': 32, 'width': 228, 'height': 116}, {'x': 543, 'y': 30, 'width': 227,
'height': 117}, {'x': 286, 'y': 30, 'width': 227, 'height': 117}]

#Ejecucion del Programa
ca_generations = generate_ca(size, generations, circles, rectangles)
display_grid(ca_generations[-1]) ##se pone -1 para acceder al ultimo elemento
de la lista para visualizarse
```

Resultado de consola: El programa se ejecuto con 3 generaciones de automatas y se demoro 1min 14 segundos.



14.3. Conclusiones:

- Observando los tiempos de ejecución de los programas de procesamiento de imagen y los modelos de autómatas:

	Procesamiento con openCV	Autómata con 3 generación, modelo 01	Autómata con 3 generaciones, modelo 02
Segundos	0	6	74

Podemos observar que realizar el procesamiento es casi instantáneo debido al uso de las librerías, en cambio los modelos preparados demoran en proporción de cuantas generaciones de autómatas existen.

- Los modelos hacen uso de los datos obtenidos por el Procesamiento con openCV por lo que si se quiere clasificar imágenes con autómatas

se necesitan datos anteriores, por lo que su uso agregaría tiempo de ejecución por lo que nuestros métodos se consideran ineficientes.

- Con estos resultados se concreta la naturaleza de los autómatas celulares de ser procesos constructivos.

15. Recursos humanos.

Debido a que el trabajo es en primera parte netamente una revisión bibliográfica y en la segunda parte implementación del algoritmo, los materiales a usar serán:

Recursos Humanos:

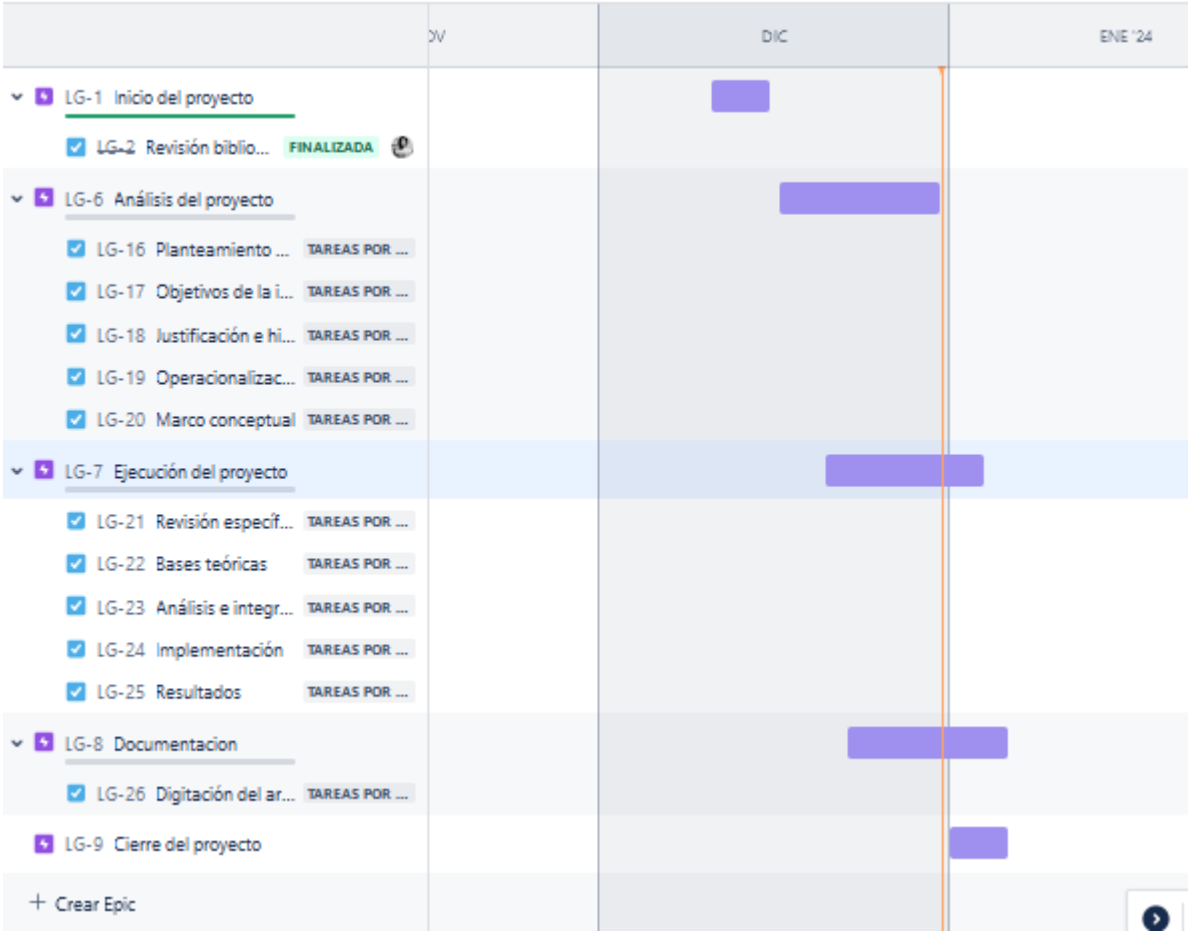
- DANNY GIANCARLO APAZA VELIZ (Docente - asesor)

- Tinta Aguilar, Miguel Ángel (líder de grupo)
- Panocca Merma, Jhon Franklin
- Paredes Mescco, Joaquin Gonzalo
- Huarancca Leon, Rodrigo Alonso
- Zapana Romero, Pedro Luis Christian

Recursos financieros.

Tipo	Categoría	Recurso	Descripción	Monto
Recursos disponibles	Infraestructura	Equipo	Laptop	Evaluación
Recurso necesarios	Gastos de trabajo de campo	Servicio de Internet	Internet móvil	S/ 10.00
		Papel	Hojas para la impresión	S/ 2.00
	Materiales	Buscadores	Elsevier, ScienceDirect	–

16. Cronograma de actividades.



17. Referencias

[1] N. R. da Silva, P. Van der Weeën, B. De Baets, and O. M. Bruno, “Improved texture image classification through the use of a corrosion-inspired cellular automaton,” Neurocomputing, vol. 149, pp. 1560–1572, Feb. 2015, doi: 10.1016/j.neucom.2014.08.036.A.

[2] Rueda & W. Torres, *Autómatas celulares para la segmentación y clasificación de imágenes multiespectrales*, V Jornadas Nacionales de Geomática 2013. Caracas - Venezuela

[3] A. Ilachinski, Cellular automata: A discrete universe. Singapore: World Scientific, 2001.

[4] “Welcome to Python.org”. Python.org. Accedido el 30 de diciembre de 2023. [En línea]. Disponible: https://www.python.org

[5] “OpenCV: Image Processing, Table of contents https://docs.opencv.org/4.x/d7/da8/tutorial_table_of_content_imgproc.html(accessed Dec. 31, 2023).

[6] “NumPy Documentation.” https://numpy.org/doc/ (accessed Dec. 30, 2023).

[7] “Google Colaboratory”. Google Colab. Accedido el 30 de diciembre de 2023. [En línea]. Disponible: https://colab.research.google.com/?hl=es

ANEXOS:

[8] “Red Magisterial | Material recortable de billetes y monedas”. Red Magisterial | Materiales Educativos Digitales. Accedido el 30 de diciembre de 2023. [En línea]. Disponible: https://www.redmagisterial.com/med/17533-material-recortable-de-billetes-y-monedas/

MATRIZ DE CONSISTENCIA

Problema	**Objetivos**	**Hipótesis**	**Variables**	**Metodología Técnicas e Instrumentos**
Problema principal: La falta de eficacia en los métodos tradicionales de clasificación de imágenes frente a la creciente complejidad y dinámica de los conjuntos de datos visuales. **Problemas específicos:** **P1.-** Identificación de patrones de imágenes: Los métodos convencionales pueden tener dificultades para identificar patrones complejos y sutiles en las imágenes, especialmente aquellos que varían en términos de iluminación, posición y escala. **P2.-** Optimización de la eficiencia computacional en la clasificación: Los métodos tradicionales pueden ser computacionalmente intensivos, lo que resulta en tiempos de procesamiento prolongados y recursos significativos.	**Objetivo General:** Evaluar la eficacia de la clasificación de imágenes mediante el uso de autómatas celulares, explorando su aplicabilidad y rendimiento en comparación con enfoques convencionales. **Objetivo Específico:** **O1.-** Revisar la literatura científica para comprender los enfoques actuales en la clasificación de imágenes, Identificar los desafíos y limitaciones asociados con los métodos existentes. **O2.-** Implementar un prototipo de clasificador de imágenes mediante el uso de un lenguaje de programación de alto nivel y técnicas de procesamiento de imágenes **O3.-** Evaluar el rendimiento de procedimiento aplicados mediante el análisis de los resultados	**Hipótesis alterna.-** El modelo de clasificación de imágenes basado en autómatas celulares supera significativamente en precisión y eficiencia a los métodos convencionales. **Hipótesis nula.-** No hay diferencia significativa en la precisión y eficiencia entre el modelo de clasificación de imágenes basado en autómatas celulares y los métodos convencionales.	**Variable independiente:** Estudio de un modelo autómata celular **Variable dependiente:** Precisión y eficacia en clasificación de imágenes	**Método y Diseño de Investigación:** **Método.-** Enfoque cualitativo, Dada la naturaleza de la investigación, inicialmente sería describir de manera puntual los autómatas celulares pero luego se realiza y de manera más profunda un enfoque cualitativo debido a que se realizará la observación de un fenómeno en el entorno natural. Se trata de datos que son difíciles de cuantificar. **Diseño.-** Experimental y descriptivo. Para esta investigación se hizo una relación entre causa y efecto, es decir que observaremos las variaciones que puede tener la ejecución del programa en donde la variable principal serán las imágenes y observar cómo se dan los cambios para

				poder describir de mejor manera el fenómeno. **Técnicas e instrumentos:** - **Python** - **OpenCV** - **Numpy** - **Google Collaboratory**

Enlace a repositorio del proyecto:
https://github.com/mtinta/TIF_FisicaComputacional
Enlace a los cuadernos de Google collab:
Programa de Procesamiento de Imágenes:
https://colab.research.google.com/drive/1EPMf7AKVE2lIq69SP3rdqFLwo0RMWGu4#scrollTo=7eaGtoiHeflu
Modelo Automata celular 01:
https://colab.research.google.com/drive/1ZHFvHGlaoWI4wFUMTWHtS6EEyZBYBNCT?usp=sharing#scrollTo=51Wo3Jb6mWeI
Modelo Autómata celular 02 :
https://colab.research.google.com/drive/1p5y9KqHPQro_w_HEEu8s5mJS1ACpKP6P#scrollTo=8cNACmoNyRH3
Video de Ejecución:
https://drive.google.com/file/d/1ybrHdNi-bAsyL6aU0wOjKFNE2FPaeJB4/view?usp=sharing