Semillero SIIC

Comenzamos 12:05

PhD MSc Jorge Espinosa
jeespinosa@elpoli.edu.co

Octubre 10 - 2023

POLITÉCNICO COLOMBIANO
JAIME ISAZA CADAVID

Semillero en Inteligencia Computacional

Agenda

Avances

- Prototipo de software basado en técnicas de machine learning aplicadas a la visión por computador para la categorización de representaciones rupestres en petroglifos digitalizados de la provincia de San Luis (Argentina).
  - Ormolgud Gonzalez - Juan Carlos Salazar Muñoz

- Desarrollo de un prototipo de aplicación móvil nativa para sistemas Android, mediante la aplicación técnicas de visión por computadora e inteligencia artificial basadas en la detección de la pose humana en tiempo real, con el propósito de cuantificar la velocidad en la fase concéntrica de la ejecución de la sentadilla.
  - Valentina Palacio – Mateo Présiga

POLITÉCNICO COLOMBIANO
JAIME ISAZA CADAVID

Semillero de Investigación de Inteligencia Computacional

PROTOTIPO DE SOFTWARE BASADO EN TÉCNICAS DE MACHINE LEARNING APLICADAS A LA VISIÓN POR COMPUTADOR PARA LA CATEGORIZACIÓN DE REPRESENTACIONES RUPESTRES EN PETROGLIFOS DIGITALIZADOS DE LA PROVINCIA DE SAN LUIS (ARGENTINA)

GONZALEZ CARDONA ORMOLGUD ormolgud_gonzalez82182@elpoli.edu.co
SALAZAR MUÑOZ JUAN CARLOS juan_salazar82182@elpoli.edu.co

GOBERNACIÓN DE ANTIOQUIA

Contexto:

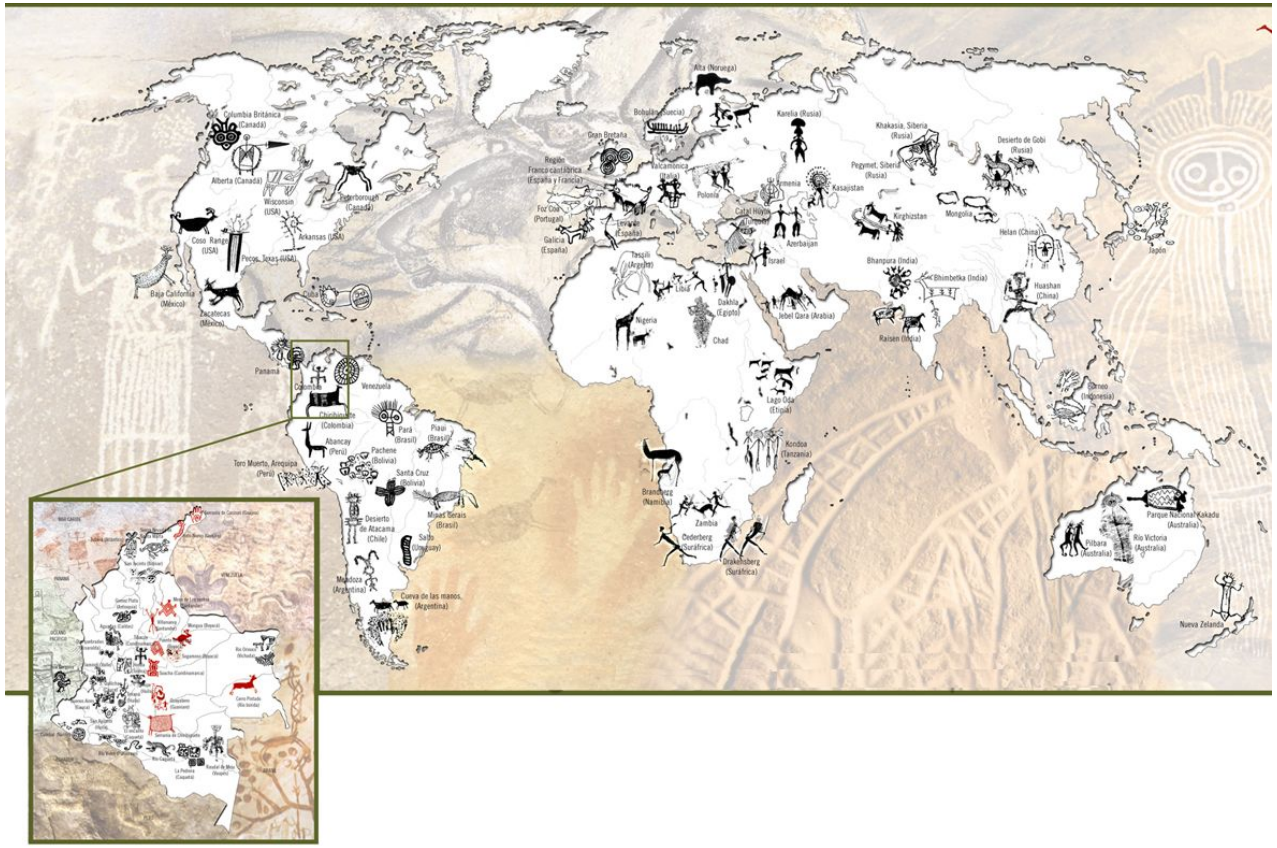
La inteligencia artificial y la arqueología no son comúnmente asociadas pero el potencial que pueden lograr es asombroso, podemos ver esto en los diferentes estudios llevados a cabo durante los últimos años.

Estudiar el arte rupestre encierra ciertas complejidades, entre las cuales se puede contar la alta dificultad técnica de datación aún con las sofisticadas técnicas modernas.

El machine learning es una técnica que aunque no es nueva, ha sido usada de forma continua y variada en el estudio del patrimonio cultural, incluyendo aspectos de interés para la antropología. Los primeros algoritmos de machine learning ya comenzaban a aparecer en la década de los setentas, no obstante en la actualidad se ha evidenciado una gran explosión en su uso, producto de una capacidad para enfrentar problemas cada vez más complejos entre los que podemos encontrar el análisis de imágenes, el aprendizaje profundo, el reconocimiento de patrones y el reconocimiento de objetos.

En el mundo

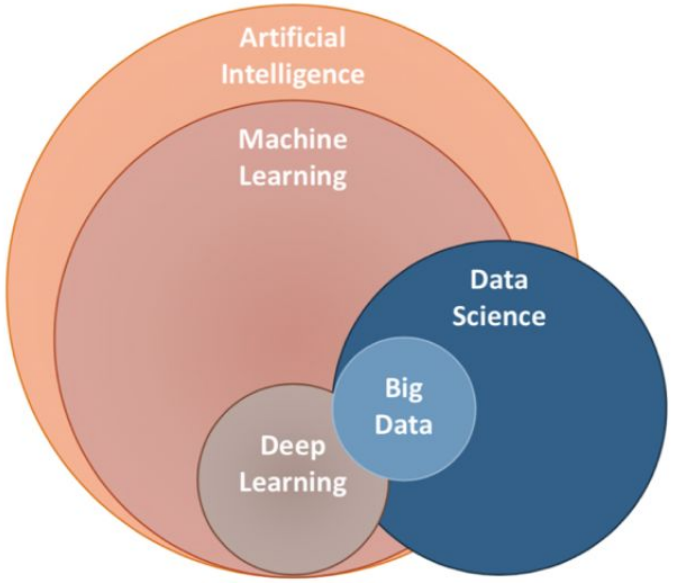
Representaciones rupestres



Complejidad y estudio

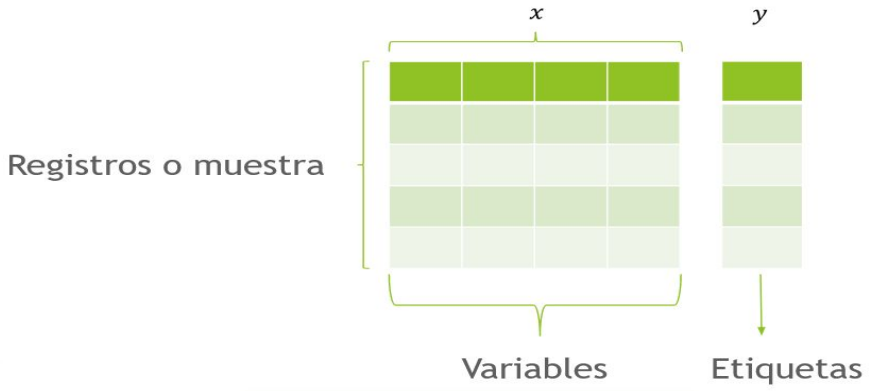
- Preservación
- Datación
- Interpretación.

Machine Learning

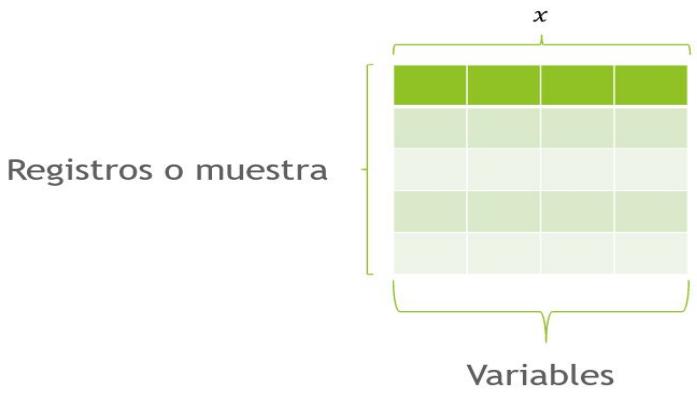


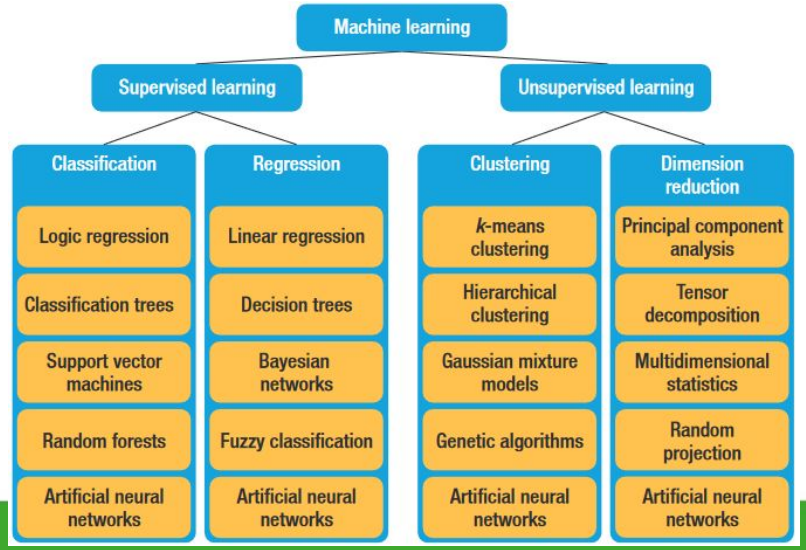
Programmatic Spain. https://www.programaticaly.com/education

Aprendizaje supervisado



Aprendizaje no supervisado





OBJETIVO GENERAL:

Desarrollar un prototipo software utilizando técnicas de machine learning aplicadas a la visión artificial para el análisis de imágenes de motivos rupestres que permitan caracterizarlos, a partir de una colección digital de imágenes rupestres de la provincia de San Luis (Argentina).

OBJETIVOS ESPECÍFICOS:

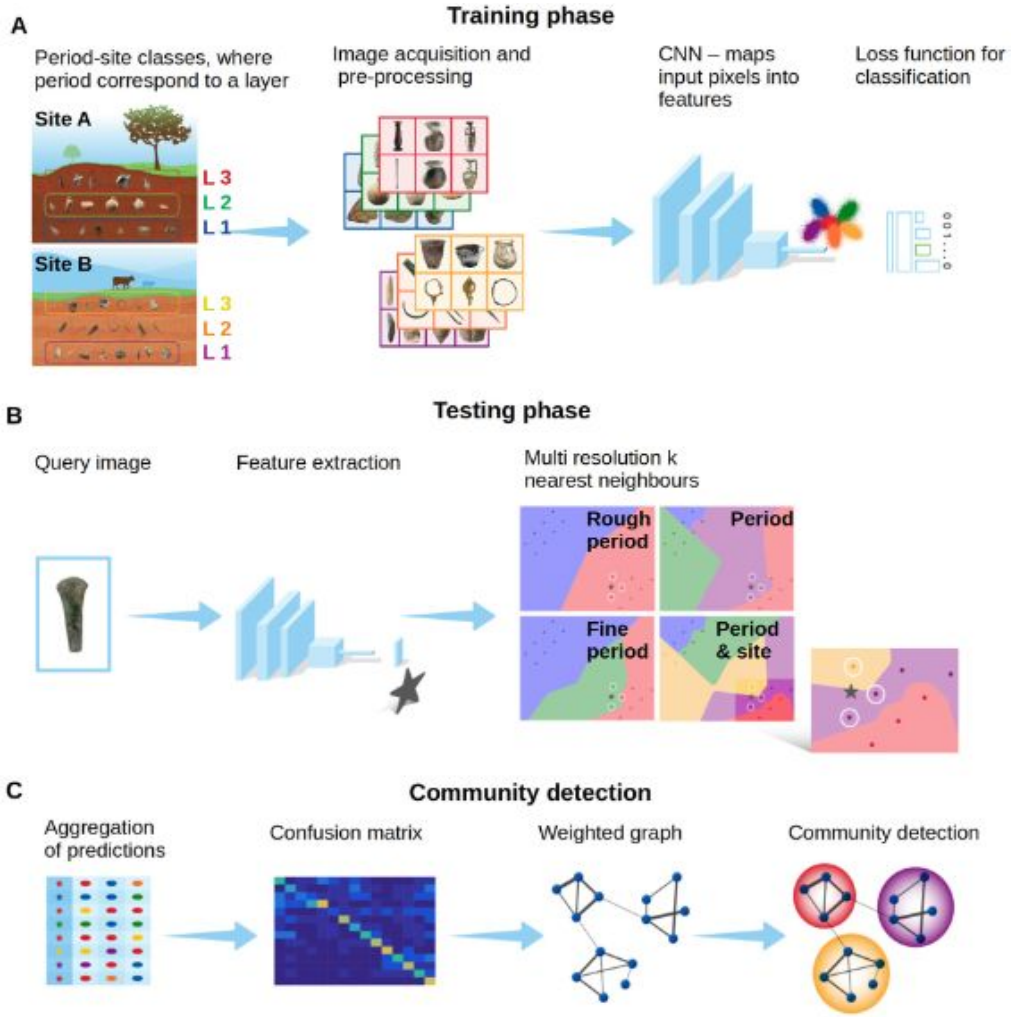
Realizar una revisión del estado del arte en bases de datos científicas para identificar las principales técnicas y algoritmos de visión artificial y machine learning que se utilizan para el reconocimiento, segmentación, interpretación y clasificación de petroglifos, así como de las métricas para cuantificar comparativamente la tarea de clasificación.

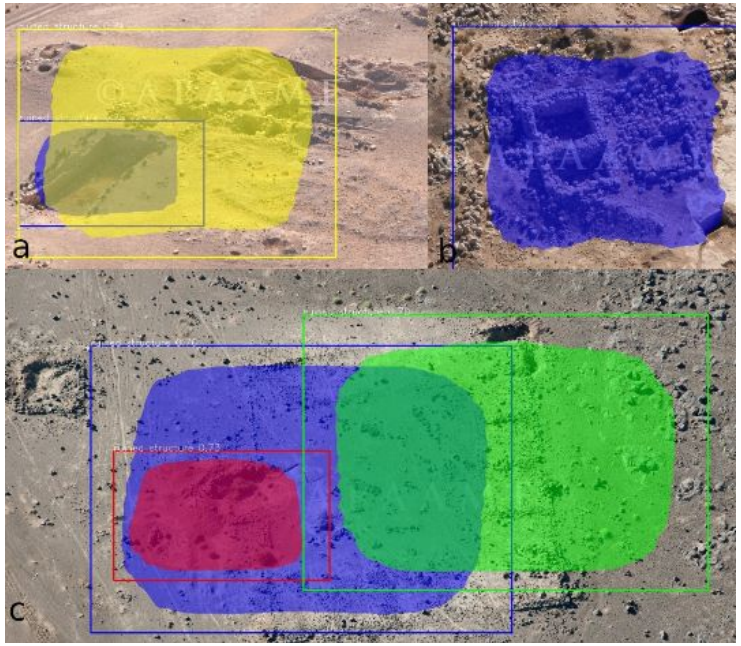
Hacer un análisis comparativo entre varios agrupamientos sobre la colección de imágenes del objeto de estudio, realizados por los algoritmos de aprendizaje de máquina escogidos y evaluando su desempeño de acuerdo los costos computacionales y las métricas escogidas, comparándolas con la clasificación realizada por un experto en Antropología.

Identificar patrones entre unidades morfológicas dentro de los resultados obtenidos en el agrupamiento realizado por el algoritmo escogido, para definir un diccionario de motivos rupestres.

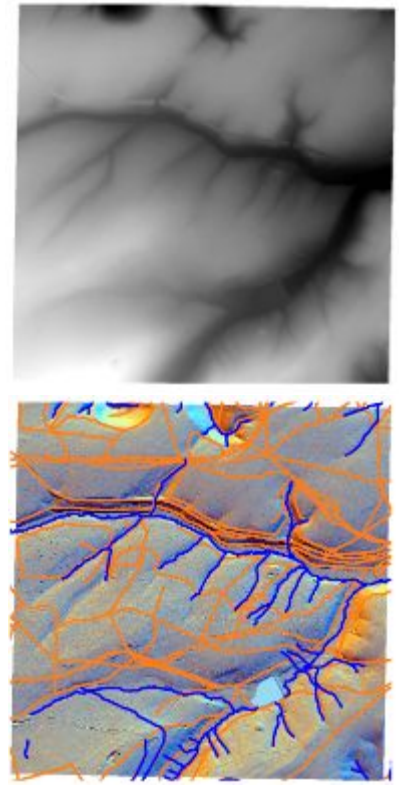
Elaborar un prototipo de software de acuerdo al mejor resultado obtenido entre los modelos definidos y estructurados en la presente investigación, que tome en cuenta los procesos de carga de imágenes, su procesamiento y captura de resultados.

Machine Learning en Arqueología



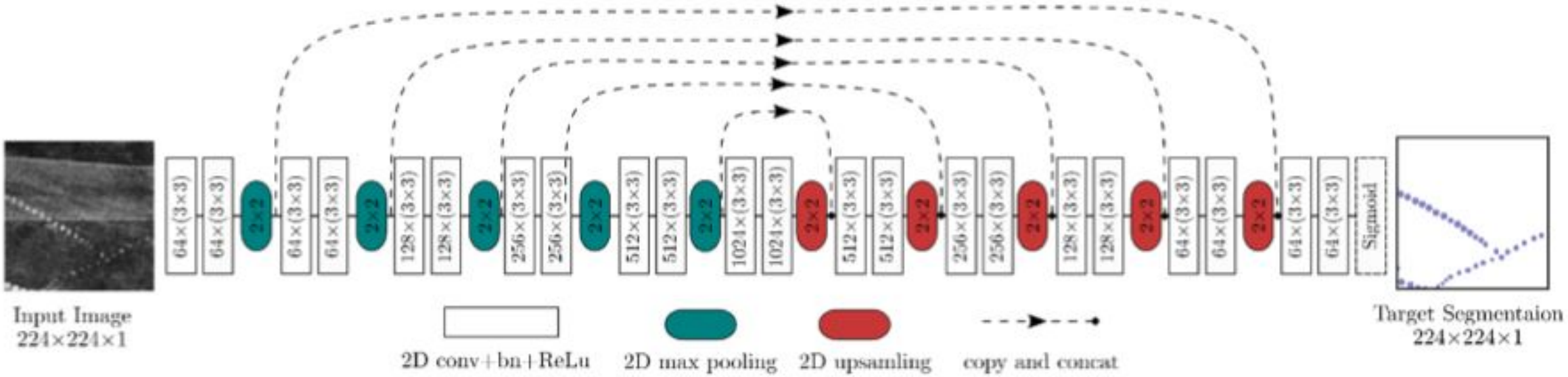


True label	Predicted label: burial	architectural	find	people_working
burial	92.0%	7.0%	1.0%	-
architectural	3.6%	96.4%	-	-
find	2.4%	-	97.6%	-
people_working	1.3%	1.3%	-	97.4%

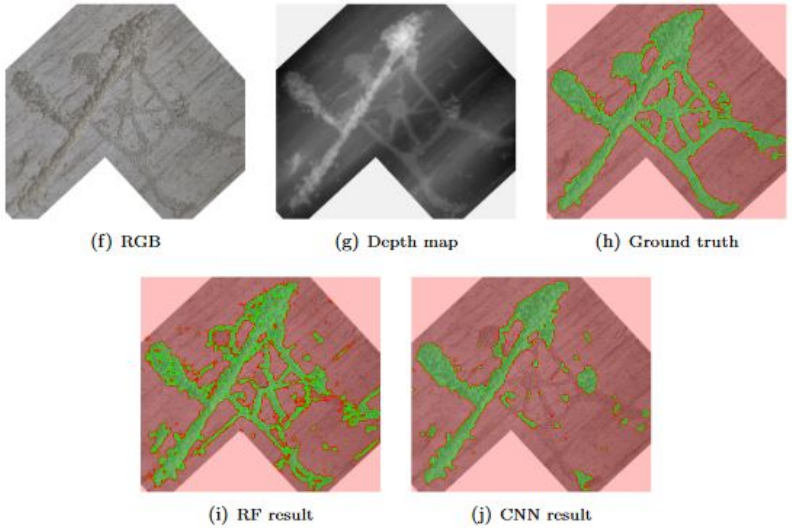


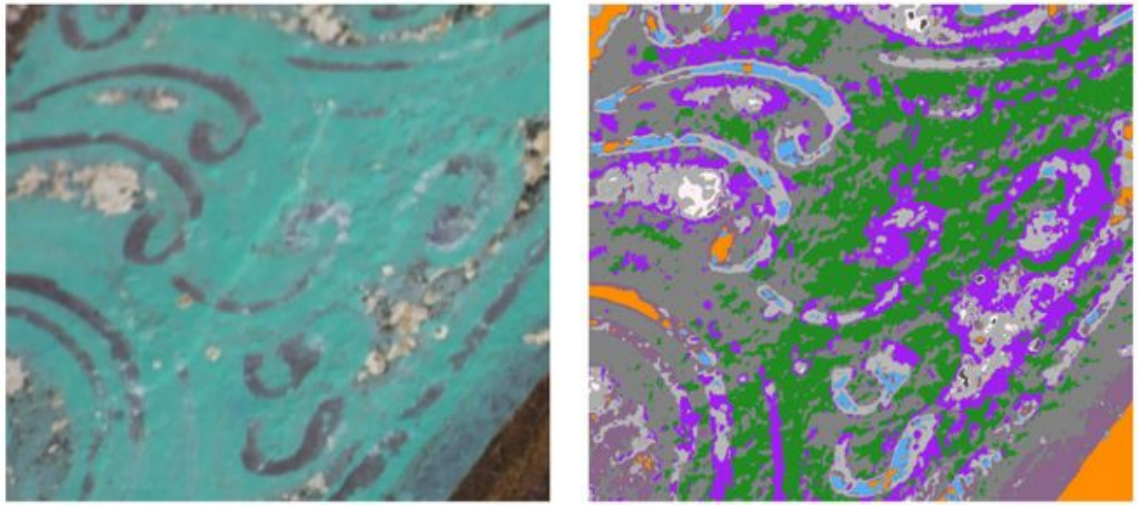
POLITÉCNICO COLOMBIANO
Jaime Isaza Cadavid

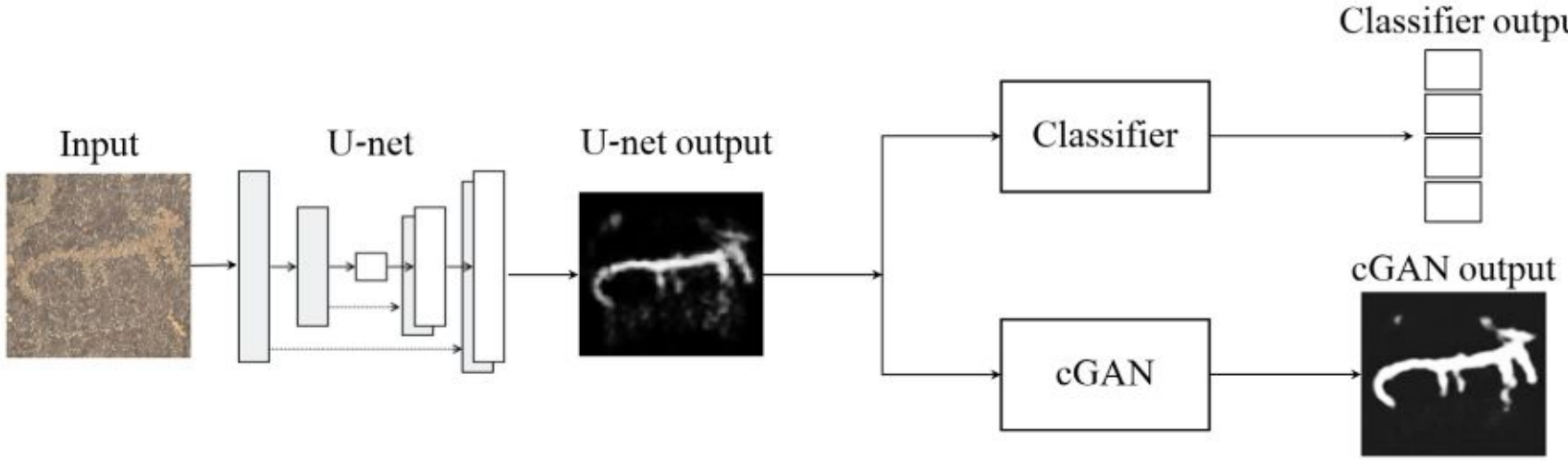
Machine Learning en Arqueología



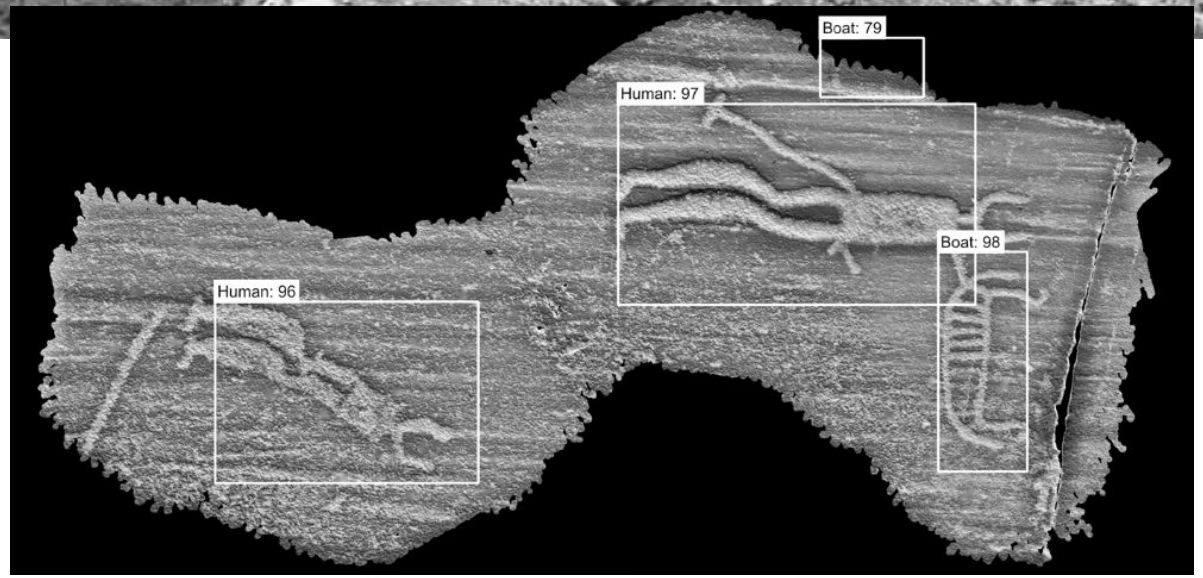
Machine Learning en estudios rupestre







Machine Learning en rupestre



Pregunta de investigación

¿Es posible desarrollar un prototipo adecuado mediante el uso de técnicas de aprendizaje profundo aplicadas a la visión artificial para realizar un agrupamiento formal de motivos rupestres de la Provincia de San Luis (Argentina)?

Estado del arte

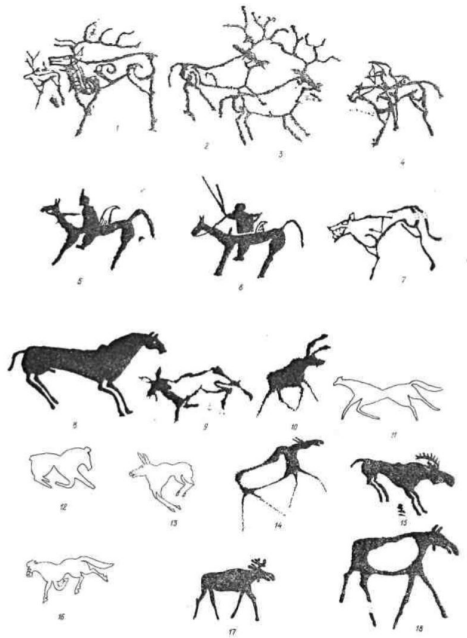


Figura 10: Imágenes utilizadas en el experimento de la máquina: 1-16 - Yenisei (12 - *Kyzlasov, 19601; 13, 16 - *Gryaznoye, 1971+); 17-18 - Angara *Okladnikov, 1966 (I)+

3.2. Lista de señales.

1. Cuerpo sólido - ED≈FH≈BC
2. Casco magro - ED>FH>BC
3. Cuerpo con "cintura" - ED>FH<BC
4. Casco pesado - ED<FH>BC
5. Carcasa lineal - (ED + BC)< $\frac{EB}{5}$
6. Piernas (I)-EI y BM se cruzan en la parte superior o son paralelas
7. Piernas (II) - EI y BM se cruzan en la parte inferior
8. Piernas (III) - EC y BL se cruzan en la parte inferior
9. La cabeza es alargada - AN>> $\frac{AE}{2}$
10. Cabeza acortada - AN< $\frac{AE}{2}$
11. El ojo se conjuga con el contorno de la cabeza
12. El ojo se separa del contorno de la cabeza
13. La oreja tiene forma triangular (fig. 2)
14. Un rizo en el omóplato
15. Remolino en el cuerpo
16. Un rizo en la cadera
17. Sultán con cresta en la cabeza (fig. 6:2)
18. Los pies de la parte trasera hacia el espectador se desplazan mecánicamente al mismo plano que los pies de la parte delantera hacia el espectador (figuras 6:2; 10:11)
19. Cornamenta de reno
20. Cornamenta de alce
21. La cola es corta
22. Cola media (no por debajo de la falda)
23. La cola es larga (por debajo de la falda)
24. El animal mira hacia la izquierda (del observador)
25. El animal mira hacia la derecha (del observador)

La lista contiene principalmente atributos cualitativos, algunos de ellos formalizados de forma imprecisa. Por supuesto, un sistema de descripción de este tipo,

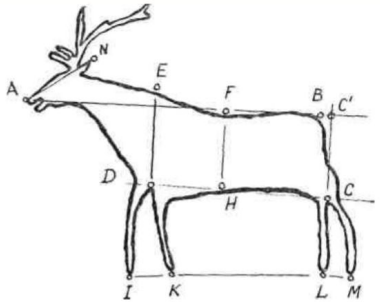


Figura 9. Puntos de partida para una descripción formalizada

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1. УТ.III.60.9	0	1	0	0	0	1	0	0	1	0	1	0	1	1	1	1	0	0	1	0	1	0	0	1	0
2. УТ.III.27.3	0	1	0	0	0	1	0	1	1	0	1	0	0	0	1	1	0	0	1	0	0	1	0	0	1
3. УТ.III.27.4	0	1	0	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0	1
4. УТ.III.60.12	0	1	0	0	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0	0	0	1	0	1	0
5. УТ.III.77.1	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
6. УТ.III.13.2	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
7. УТ.III.60.18	0	1	0	0	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0	0	0	0	1	1	0
8. О.III.5.8	0	1	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
9. Т.II.22.1	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0
10. УТ.II.6.31	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
11. Сулек	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0
12. СЧ	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	0
13. Т.III.ск.	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0
14. ЧЛ.24.57	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
15. О.III.5.6.	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
16. Т. III.ск.	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0
17. Ангара	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
18. Ангара	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

$$f_{ij} = \frac{s^2}{kl}$$

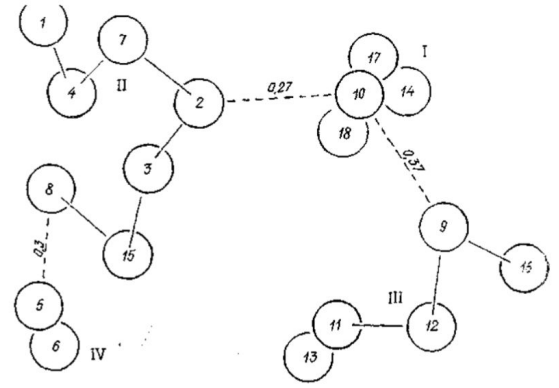
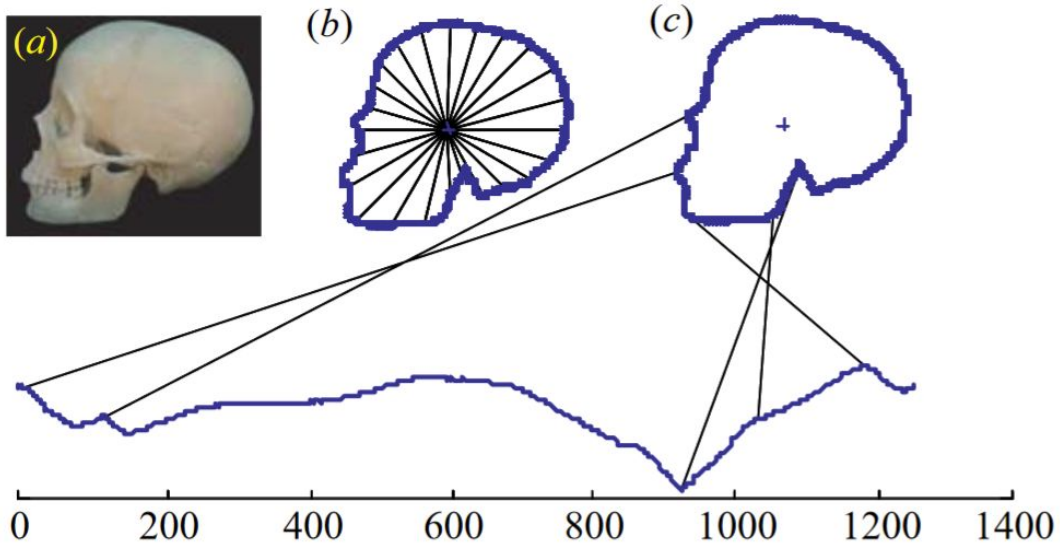
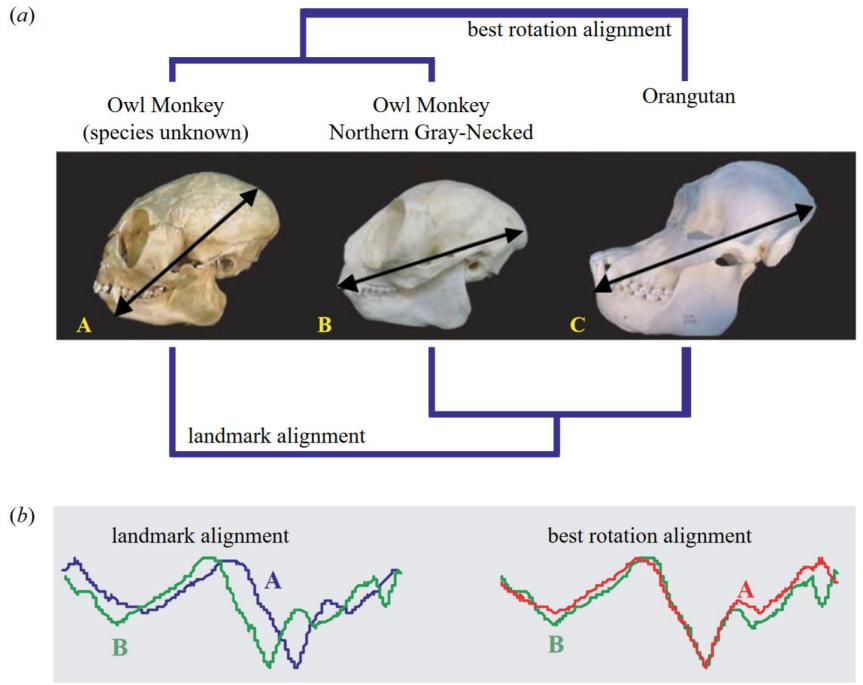


Tabla X. Gráfico de conexiones entre grupos de dibujos de distintas épocas y culturas:
I - Neolítico; II - Estilo escita-siberiano; III - Cultura tashtyk; IV - Época turca.

Y. A. Sher, *ПЕТРОГЛИФЫ СРЕДНЕЙ И ЦЕНТРАЛЬНОЙ АЗИИ*. Moscú: Nauka, 1980.





L. Wei, E. Keogh, X. Xi, y S.-H. Lee, «Supporting anthropological research with efficient rotation invariant shape similarity measurement», *J. R. Soc. Interface*, vol. 4, n.º 13, pp. 207-222, oct. 2006, doi: 10.1098/rsif.2006.0168

#	Título	Número de figuras	Tipo de descriptor	Descriptor	Clasificador	Métrica	Segmentación	Clasificación	Año
1	Petroglyphs in Central Asia, Nauka (in Russian)	18	Forma	Matriz de características arbitrarias basadas medidas	k-NN	Cualitativa	○	✓	1980
2	Pattern recognition applied to Rock Art	Sin precisar	Sin descriptor	Sin descriptor	Red de Kohonen	Cualitativa	○	✓	2001
3	Shape analysis of petroglyphs in central Asia	52	Esqueletización	Arreglo codificado	Distancia comparativa uno a uno	Cualitativa	○	✓	2006
4	Supporting anthropological research w ith efficient rotation invariant shape similarity measurement	Sin precisar	Forma	Serie de tiempo	Distancia euclidiana	Cualitativa	○	✓	2006
5	Augmenting the generalized hough transform to enable the mining of petroglyphs	18	Región	Modificación de la transformada generalizada de Hough	Dnn measure	Accuracy	○	✓	2009
6	Tow ards Indexing and Data Mining All the World's Rock Art	18	Región	Modificación de la transformada generalizada de Hough	Dnn measure	Accuracy	○	✓	2010
7	Kohonen Netw orks Applied to Rincón del Toro Rock Art Site Analysis	68	Sin descriptor	Sin descriptor	Red de Kohonen	Accuracy	○	✓	2010
8	An efficient and effective similarity measure to enable data mining of petroglyphs	143	Región	Modificación de la transformada generalizada de Hough	Dnn measure	Cualitativa	○	✓	2011
9	Differences of Petroglyph Styles among Archaeological Sites w ith Spatial and Time-w ise Distances	52	Esqueletización	Arreglo codificado	Distancia comparativa uno a uno	Cualitativa	○	✓	2011
10	Detection and Classification of Petroglyphs in Gigapixel Images -- Preliminary Results	1 imagen con múltiples figuras	Color y textura	RGB, GLCM	SVM	Accuracy	✓	○	2011
11	Petroglyph Recognition Using Self-Organizing Maps and Fuzzy Visual Language Parsing	500	Región	Transformada de Radón	SOM & Fuzzy	Accuracy	○	✓	2012
12	Petroglyph Classification using the Image Distortion Model	51	Derivados	Horizontal and vertical gradients	IDM, KNN	Accuracy	○	✓	2012
13	Automated petroglyph image segmentation w ith interactive classifier fusion	1 imagen con múltiples figuras	Color y textura	RGB, SIFT, GLCM, LBP	SVM	Precision, recall y F1	✓	○	2012
14	Combining Unsupervised Clustering w ith a Non-linear Deformation Model for Efficient Petroglyph Recognition	1530 (generadas de 51)	Forma	Shape contexts	SOM, IDM	Accuracy	○	✓	2013
15	Segmentation and Recognition of Petroglyphs Using Generic Fourier Descriptors	1215	Forma	Generic Fourier Descriptors	Distancia euclidiana	Accuracy	✓	✓	2014

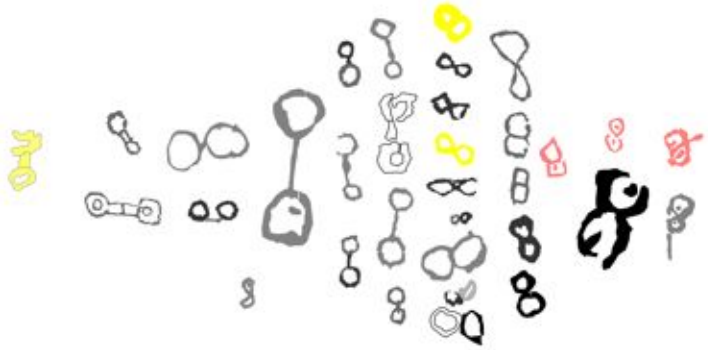
16	Graph-Based Shape Similarity of Petroglyphs	100 en diez clases	Grafo, Forma, Región	Graph edit distance (GED) & Graph embedding (GE), Shape contexts (SC), Inner Distance Shape Context (IDSC), Generalised Hough Transform (GHT)	k-NN	Accuracy	○	✓	2015
17	Automated Classification of Petroglyphs	1388	Grafo, Forma, Región	Shape contexts (SC), Inner Distance Shape Context (IDSC), Generalised Hough Transform (GHT), Graph embedding (GE)	k-NN	Accuracy	○	✓	2015
18	Interactive segmentation of rock-art in high-resolution 3D reconstructions	26 superficies 3D	Topográfica	Mapa de profundidad	Random Forest	Dice Similarity Coefficient (DSC), Hit Rate (HR), False Acceptance Rate (FAR)	✓	○	2015
19	A Mobile Application for Supporting Archaeologists in the Classification and Recognition of Petroglyphs	Sin precisar en 17 clases	Derivados	Horizontal and vertical gradients	k-NN	Accuracy	○	✓	2016
20	Interactive 3D Segmentation of Rock-Art by Enhanced Depth Maps and Gradient Preserving Regularization	26 superficies 3D	Topográfica	Mapa de profundidad	Random Forest	Dice Similarity Coefficient (DSC), Hit Rate (HR), False Acceptance Rate (FAR)	✓	○	2016
21	Computational Analysis of Petroglyphs	100 en diez clases	Grafo, Forma, Región	Graph edit distance (GED) & Graph embedding (GE), Shape contexts (SC), Inner Distance Shape Context (IDSC), Generalised Hough Transform (GHT)	k-NN	Accuracy	✓	✓	2016
22	PetroSurf3D - A high-resolution 3D Dataset of Rock Art for Surface Segmentation	26 superficies 3D	Topográfica	Mapa de profundidad	Random Forests (RF), Redes Neuronales Convolucionales (CNNs)	Índice de Jaccard, Intersección sobre unión (IU), Promedio sobre clases (mIU), precisión de píxel (PA), dice similarity coefficient (DSC) y False acceptance rate (FAR)	✓	○	2017
23	The 3D-Pitoti Dataset: A Dataset for high-resolution 3D Surface Segmentation	26 superficies 3D	Topográfica	Mapa de profundidad	Random Forests (RF), Redes Neuronales Convolucionales (CNNs)	Índice de Jaccard, Intersección sobre unión (IU), Promedio sobre clases (mIU), precisión de píxel (PA), dice similarity coefficient (DSC) y False acceptance rate (FAR)	✓	○	2017
24	Information Visualization: from Petroglyphs to CoDe Graphs	Sin estimar en 17 clases	Derivados	Horizontal and vertical gradients	k-NN	Precision, recall	○	✓	2018
25	Reconstructing rock art chronology w ith transfer learning: A case study from Arnhem Land, Australia	98 motivos en 6 clases	Codificación de imágen con CNN	Vector de activación de penúltima capa	One-shot learning	Accuracy	○	✓	2021
26	BEGL: boundary enhancement w ith Gaussian Loss for rock-art image segmentation	26 superficies 3D	Sin descriptor	Sin descriptor	ResNet, UNet y BEGL	Pixel accuracy, average precision, recall, F1-score, mean intersection over union (MIoU) and dice similarity coefficient (DSC) metrics	✓	○	2023

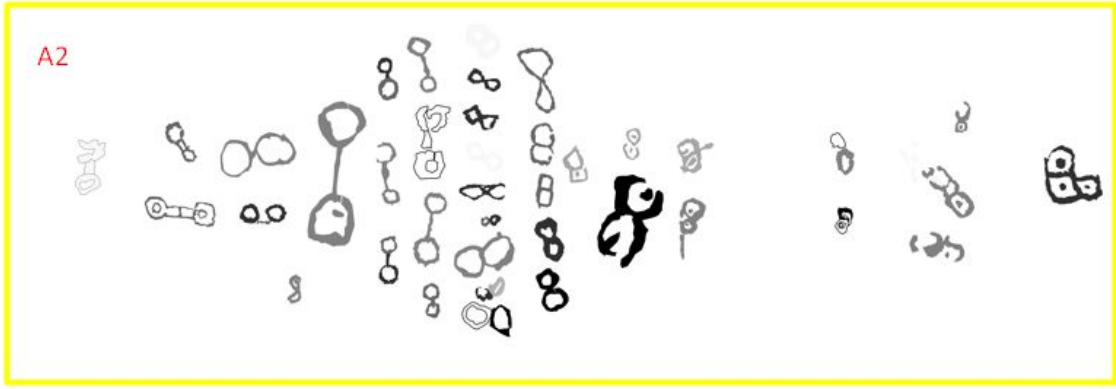
Datos

Preprocesamiento de data

Imagen png con todos los motivos incluidos
Imagen en escala de grises
Se segmenta la data para (figuras de forma individual)
    imagenes en carpeta
Imagen en una sola capa (eliminar Profundidad (RGB)

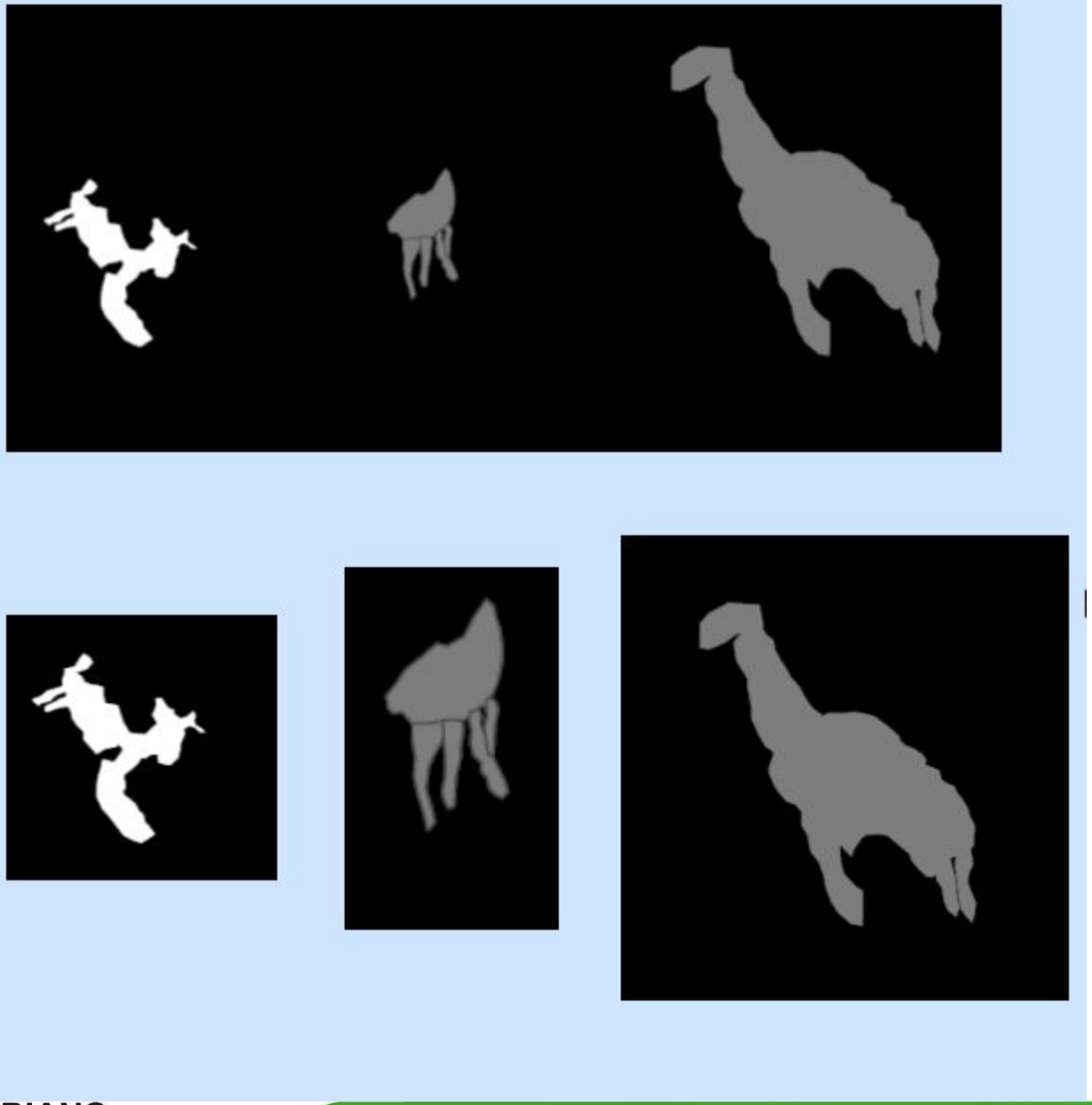
cortar imagen desde su último pixel (1)
volver imagen cuadrada
expandir fondo de la imagen para permitir rotación sin pérdida
keras
data augmenting




A2

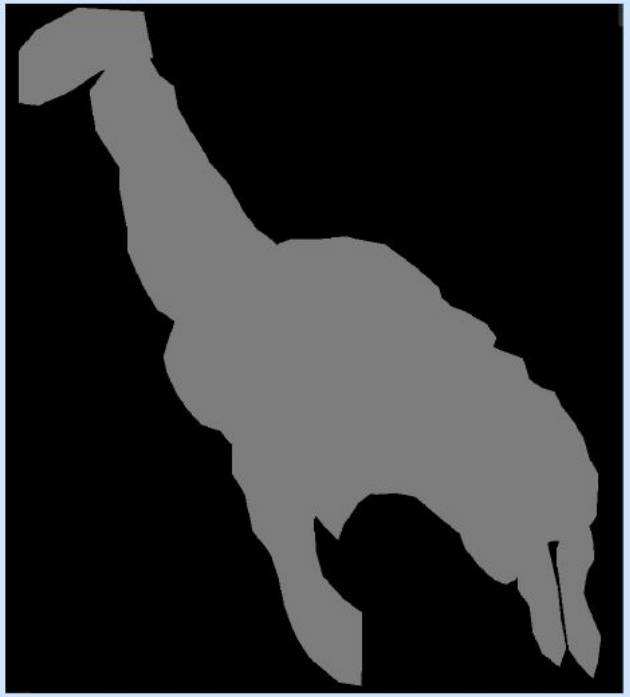

A3

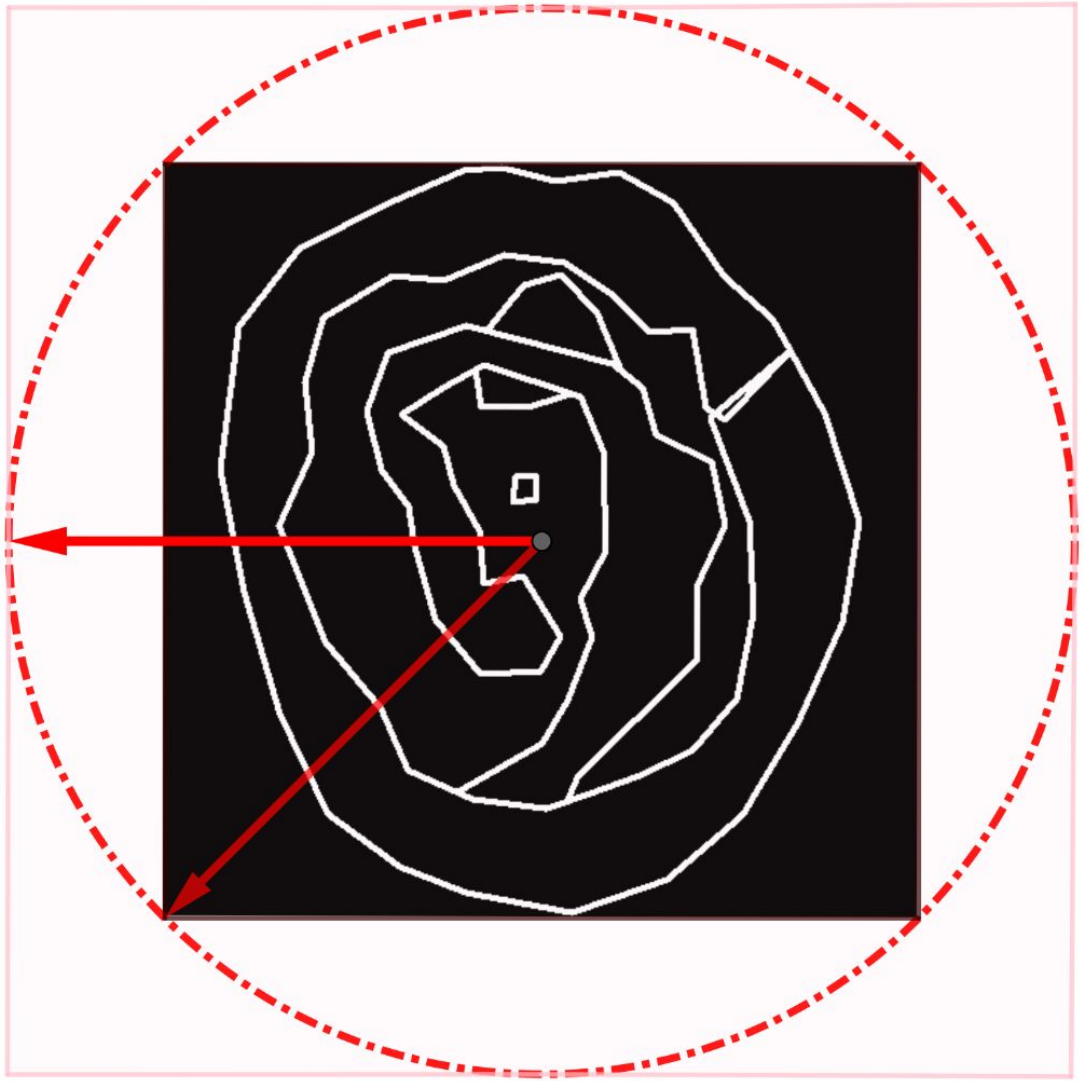

A4



```python
def preProcesingImages(sourcePath, targetPath):
  number = 0
  folders = os.listdir(sourcePath)
  folders.sort()
  for folder in folders:
    group_path = sourcePath + "/" + folder
    files_names = os.listdir(group_path)
    files_names.sort()
    for file_name in files_names:
      img_path = group_path + "/" + file_name
      imagecopy = cv2.imread(img_path, cv2.IMREAD_UNCHANGED)
      img = imagecopy[:,:,0] #Se aplana

      #Se recortan bordes
      sizeOriginal = img.shape[:2]
      heightOriginal = sizeOriginal[0]
      widthOriginal = sizeOriginal[1]
      top = -1
      down = heightOriginal
      row = 0
      left = -1
      right = widthOriginal
      col = 0

      while top == -1:
        if np.count_nonzero(img[row], axis = 0) != 0:
          top = row
        row += 1
      row = heightOriginal - 1
      while down == heightOriginal:
        if np.count_nonzero(img[row], axis = 0) != 0:
          down = row
        row -= 1

      while left == -1:
        if np.count_nonzero(img[:, col], axis = 0) != 0:
          left = col
        col += 1
      col = widthOriginal - 1
      while right == widthOriginal:
        if np.count_nonzero(img[:, col], axis = 0) != 0:
          right = col
        col -= 1

      copy = img[top:down + 1, left:right + 1]
```

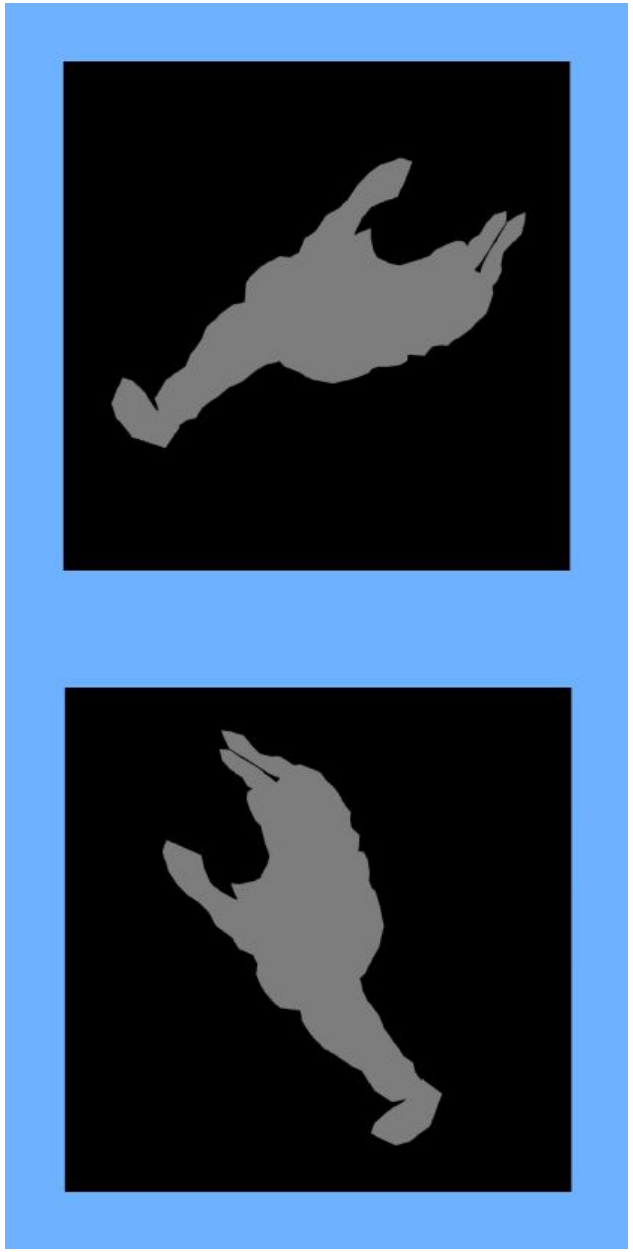
```python
      #Squared
      sizeOriginal = copy.shape[:2]
      heightOriginal = sizeOriginal[0]
      widthOriginal = sizeOriginal[1]
      resized = []
      if heightOriginal > widthOriginal: #Rellenar a los lados
        widthHalf = int((heightOriginal - widthOriginal) / 2)
        fill = np.zeros((heightOriginal, widthHalf), int)
        resizedImage = np.hstack((fill, copy))
        if heightOriginal % 2 == 0:
          resizedImage = np.hstack((resizedImage, fill))
        else:
          fill = np.zeros((heightOriginal, widthHalf + 1), int)
          resizedImage = np.hstack((resizedImage, fill))
      elif heightOriginal < widthOriginal: #Rellenar arriba y abajo
        heightHalf = int((widthOriginal - heightOriginal) / 2)
        fill = np.zeros((heightHalf, widthOriginal), int)
        resizedImage = np.vstack((fill, copy))
        if widthOriginal % 2 == 0:
          resizedImage = np.vstack((resizedImage, fill))
        else:
          fill = np.zeros((heightHalf + 1, widthOriginal), int)
          resizedImage = np.vstack((resizedImage, fill))
      else:
        resizedImage = copy

      file_name = file_name[:-4]
      if len(str(number)) == 1:
        zeros = "00"
      elif len(str(number)) == 2:
        zeros = "0"
      else:
        zeros = ""

      targetFolder = targetPath + "/" + zeros + str(number)
      if not os.path.exists(targetFolder):
          os.makedirs(targetFolder)

      cv2.imwrite(targetFolder + "/" + folder + file_name + "_" + zeros
                  + str(number) + ".png", resizedImage)
    number = number + 1
```





```python
def fillToRotate(sourcePath, targetPath):
  folders = os.listdir(sourcePath)
  folders.sort()
  for folder in folders:
    group_path = sourcePath + "/" + folder
    files_names = os.listdir(group_path)
    files_names.sort()
    for file_name in files_names:
      img_path = group_path + "/" + file_name
      imagecopy = cv2.imread(img_path, cv2.IMREAD_UNCHANGED)

      sizeOriginal = imagecopy.shape[:2]
      heightOriginal = sizeOriginal[0]
      widthOriginal = sizeOriginal[1]

      if heightOriginal > widthOriginal: #Rellenar a la izquierda
        fill = np.zeros((heightOriginal, 1), int)
        resizedImage = np.hstack((fill, imagecopy))
        side = heightOriginal
      elif heightOriginal < widthOriginal: #Rellenar arriba
        fill = np.zeros((1, widthOriginal), int)
        resizedImage = np.vstack((fill, imagecopy))
        side = widthOriginal
      else:
        resizedImage = imagecopy
        side = widthOriginal

      cathetus = int(side / 2)
      radius = int(sqrt(2 * cathetus ** 2 ))

      widhtLateral = radius - cathetus

      fill = np.zeros((side, widhtLateral), int)
      resizedImage = np.hstack((fill, resizedImage))
```

```python
widhtFinal = 2 * widhtLateral + side
if widhtFinal % 2 == 0:
  resizedImage = np.hstack((resizedImage, fill))
  fill = np.zeros((widhtLateral, 2 * widhtLateral + side), int)
  resizedImage = np.vstack((fill, resizedImage))
  resizedImage = np.vstack((resizedImage, fill))

else:
  fill = np.zeros((side, widhtLateral + 1), int)
  resizedImage = np.hstack((resizedImage, fill))
  fill = np.zeros((widhtLateral, 2 * widhtLateral + side + 1), int)
  resizedImage = np.vstack((fill, resizedImage))
  fill = np.zeros((widhtLateral + 1, 2 * widhtLateral + side + 1), int)
  resizedImage = np.vstack((resizedImage, fill))

targetFolder = targetPath + "/" + folder
if not os.path.exists(targetFolder):
    os.makedirs(targetFolder)

file_name = file_name[:-4]
cv2.imwrite(targetFolder + "/" + file_name + ".png", resizedImage)
```




Keras
Simple. Flexible. Powerful.


TensorFlow
Keras

Config Data augmentation

```
[ ]  data_augmentation = keras.Sequential(
       [
         layers.RandomFlip(),
         layers.RandomRotation(0.99, fill_mode = 'constant'),
       ]
     )
```

- tf.keras.layers.RandomCrop
- tf.keras.layers.RandomFlip
- tf.keras.layers.RandomTranslation
- tf.keras.layers.RandomRotation
- tf.keras.layers.RandomZoom
- tf.keras.layers.RandomContrast

```python
def dataAugmenting(sourcePath, targetPath, augmentationRate):
  maxFigures = 0
  folders = os.listdir(sourcePath)
  folders.sort()

  for folder in folders:
    group_path = sourcePath + "/" + folder
    files_names = os.listdir(group_path)
    files = len(files_names)
    if maxFigures < files:
      maxFigures = files

  maxFigures = maxFigures * augmentationRate

  for folder in folders:
    group_path = sourcePath + "/" + folder
    files_names = os.listdir(group_path)
    files_names.sort()
    files = len(files_names)
    files = int(maxFigures / files)
    for file_name in files_names:
      img_path = group_path + "/" + file_name
      imagecopy = cv2.imread(img_path)
      file_name = file_name[:-4]
      targetFolder = targetPath + "/" + folder
      if not os.path.exists(targetFolder):
        os.makedirs(targetFolder)
      original = cv2.imread(img_path, cv2.IMREAD_UNCHANGED)
      cv2.imwrite(targetFolder + "/" + file_name + "_00000" + ".png", original)

      for i in range(files):
        imageAugmented = data_augmentation(imagecopy)
        copyimageAugmented = imageAugmented[:,:,0] #Se aplana
        match len(str(i)):
          case 1:
            zeros = "000"
          case 2:
            zeros = "00"
          case 3:
            zeros = "0"
          case _:
            zeros = ""
        cv2.imwrite(targetFolder + "/" + file_name + "_A" + zeros + str(i) + ".png", copyimageAugmented.numpy() )
```


A1Z15F03_014_A0007...


A1Z15F03_014_A0008...


A1Z15F03_014_A0009...


A1Z15F03_014_A0010...


A1Z15F03_014_A0011....

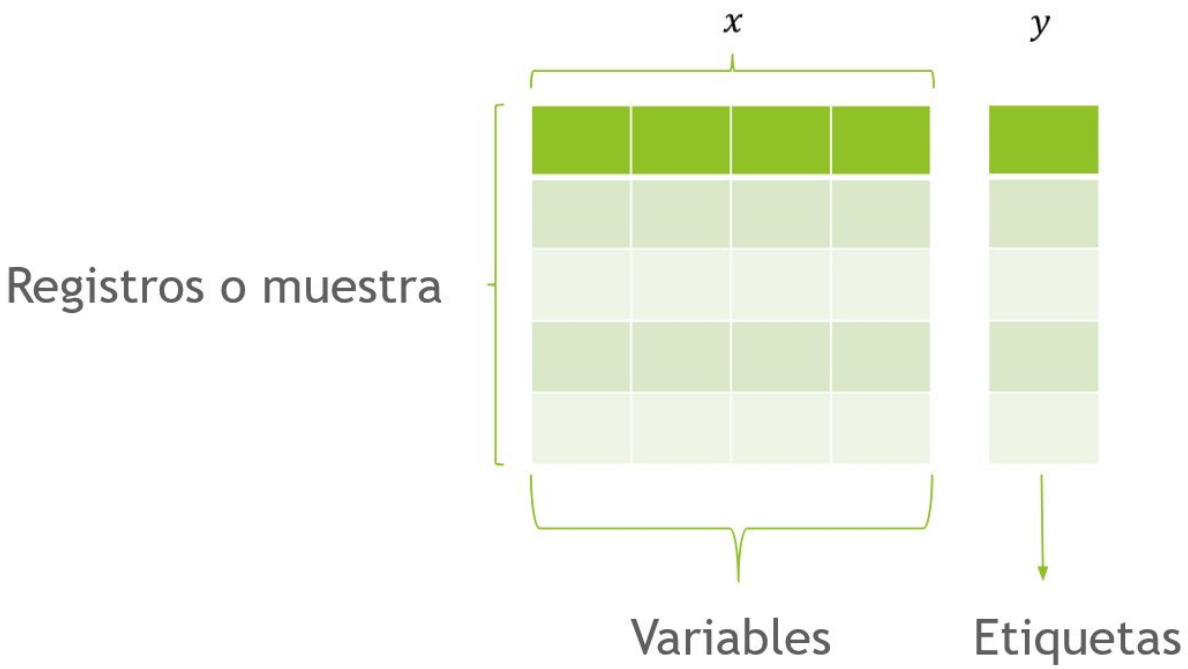

A1Z15F03_014_O000...

POLITÉCNICO COLOMBIANO
JAIME ISAZA CADAVID

TRANSFER LEARNING

GOBERNACIÓN DE ANTIOQUIA

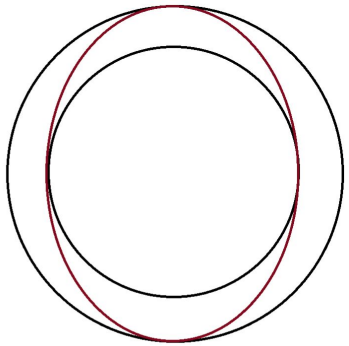
Contexto



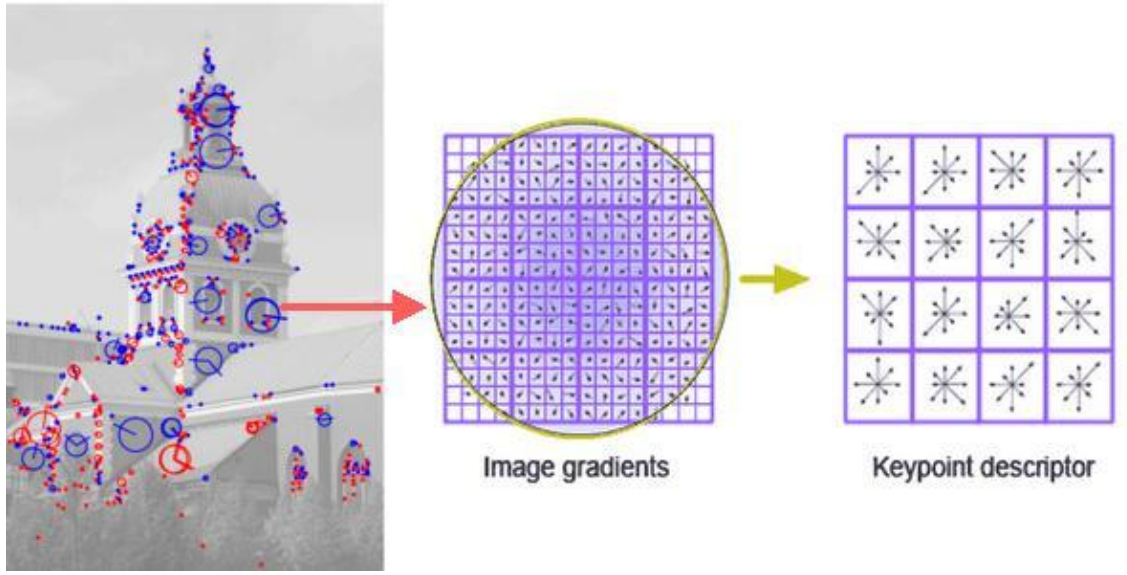
Característica (Feature)

Descriptores de forma:

- Basados en volumen
  - Momentos geométricos
  - Momentos de Zernike
- Basados en contorno
  - Transformada de Hough
  - Descriptores de Fourier

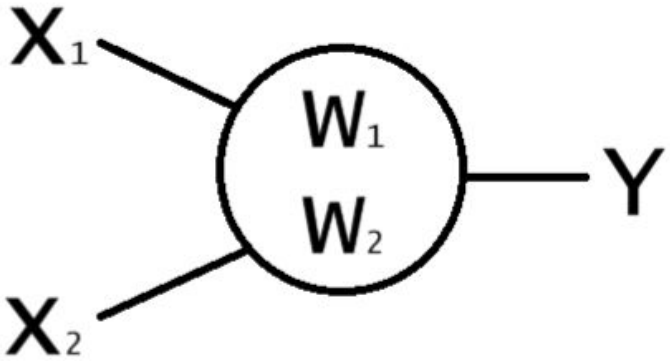


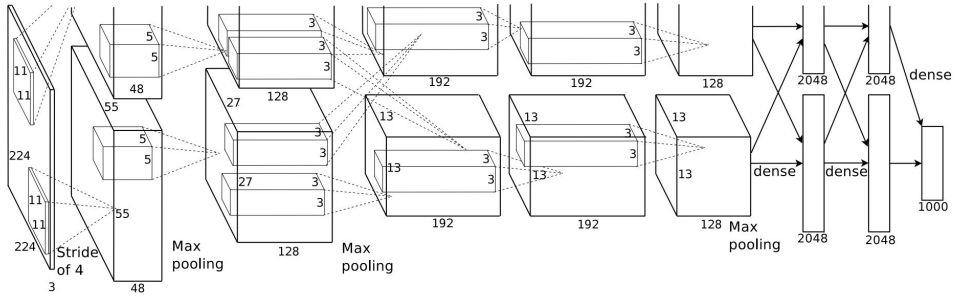
$$h_0 = \eta_{20} + \eta_{02}$$
$$h_1 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$
$$h_2 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$
$$h_3 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$
$$h_4 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$
$$h_5 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})]$$
$$h_6 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$



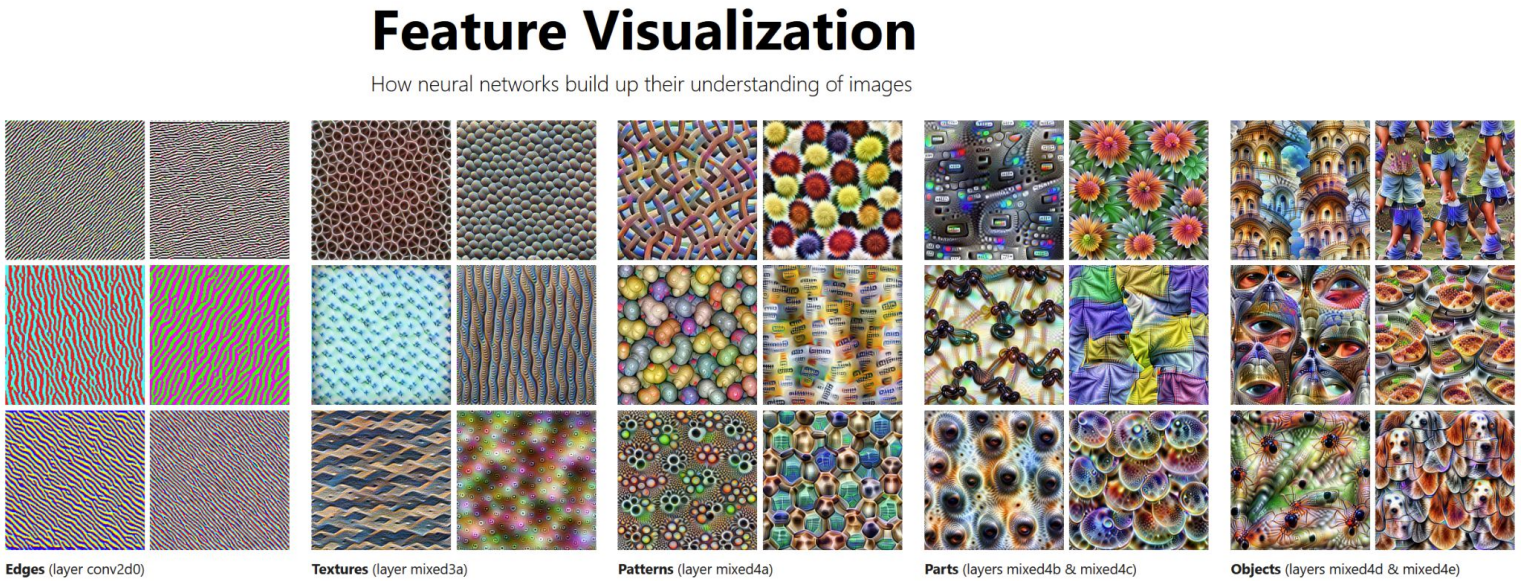
Adel, Ebtsam & Elmogy, Mohammed & El-Bakry, Hazem. (2014). Image Stitching based on Feature Extraction Techniques: A Survey. International Journal of Computer Applications. 99. 1-8. 10.5120/17374-7818.

CNN



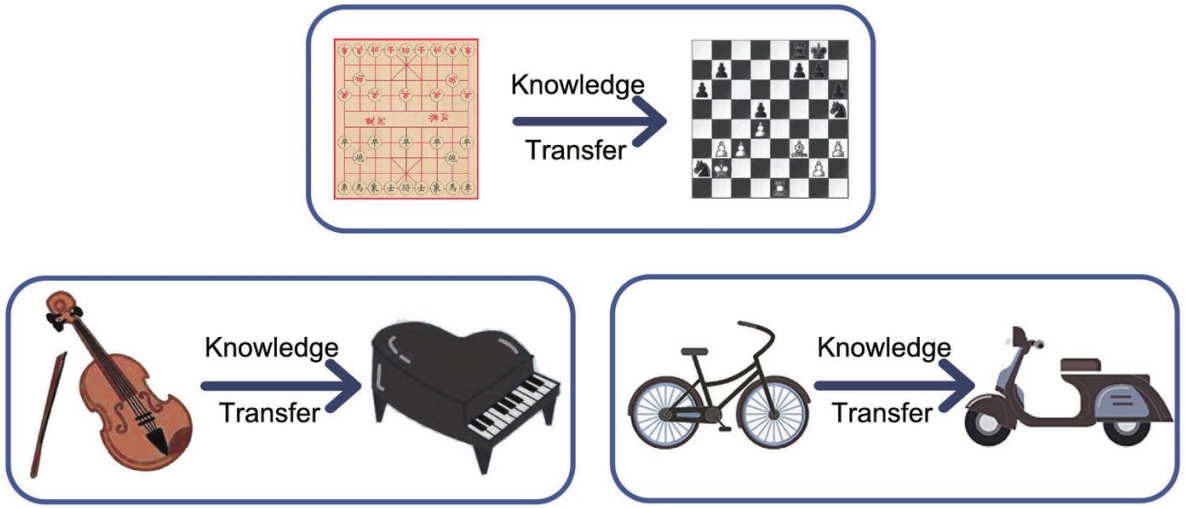


Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.

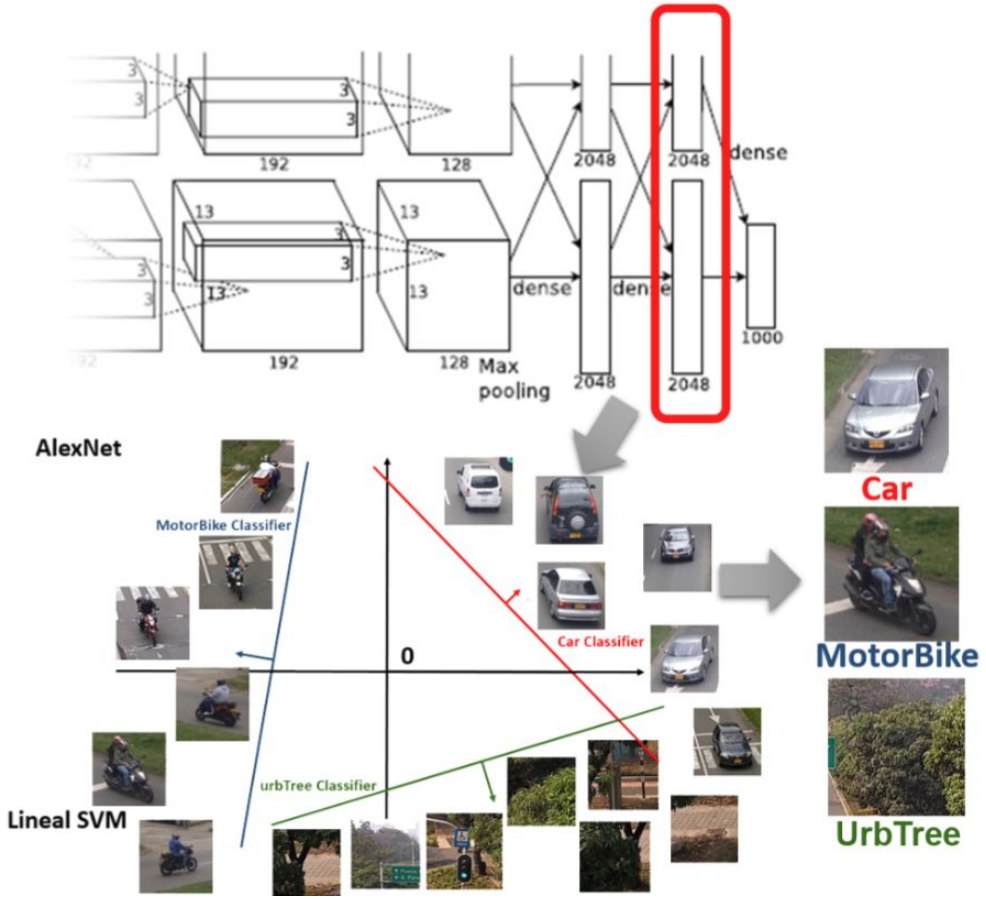


Source: (https://microscope.openai.com/about)

Concepto



Intuitive examples about transfer learning (Source: https://arxiv.org/pdf/1911.02685.pdf)



Espinosa, J. E., Velastin, S. A., & Branch, J. W. (2018). Motorcycle Classification in Urban Scenarios using Convolutional Neural Networks for Feature Extraction (arXiv:1808.09273). arXiv. https://doi.org/10.48550/arXiv.1808.09273

Ejemplo

```
mobilenet_v2 = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"

feature_extractor_model = mobilenet_v2
```

```
feature_extractor_layer = hub.KerasLayer(
    feature_extractor_model,
    input_shape = (224, 224, 3),
    trainable = False)
```

```
feature_batch = feature_extractor_layer(image_batch)
print(feature_batch.shape)
```

```
(64, 1280)
```

```
num_classes = len(class_names)

model = tf.keras.Sequential([
  feature_extractor_layer,
  tf.keras.layers.Dense(num_classes)
])

model.summary()
```

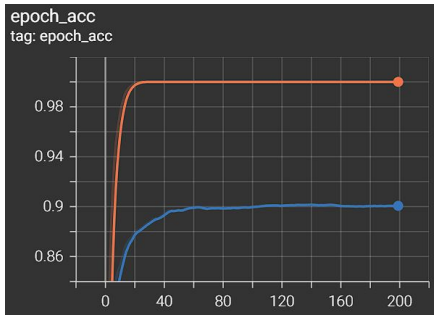
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 1280)              2257984

 dense (Dense)               (None, 106)               135786

=================================================================
Total params: 2,393,770
Trainable params: 135,786
Non-trainable params: 2,257,984
_________________________________________________________________
```

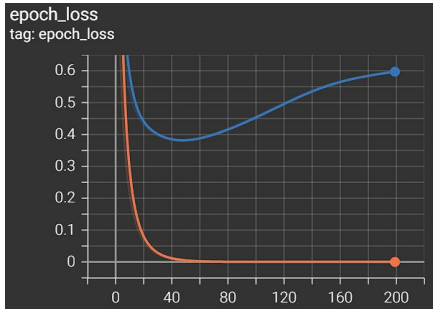
```
predictions = model(image_batch)
```

```
model.compile(
  optimizer = tf.keras.optimizers.Adam(),
  loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits = True),
  metrics = ['acc'])

log_dir = "Logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(
    log_dir=log_dir,
    histogram_freq=1) # Enable histogram computation for every epoch.
```

```
num_epochs = 200
```

```
history = model.fit(train_ds,
                    validation_data = val_ds,
                    epochs = num_epochs,
                    callbacks = tensorboard_callback)
```






```
Epoch 47/200
205/205 [==============================] - 61s 298ms/step - loss: 0.0060 - acc: 1.0000 - val_loss: 0.3814 - val_acc: 0.8964
```

POLITÉCNICO COLOMBIANO
Jaime Isaza Cadavid

EL POLI
ACREDITADO EN ALTA CALIDAD
CALIDAD ACADÉMICA Y HUMANA

¡GRACIAS!

YouTube

Politécnico Colombiano Jaime Isaza Cadavid

@PolitecnicoJIC

www.politecnicojic.edu.co