

# Clasificación de Imágenes de Mosquitos Mediante Deep Learning y Bag of Visual Words con Preprocesado

Antonio Barco Pérez

antoniobarco@correo.ugr.es

Carlos Muñoz Sánchez

mscarlos@correo.ugr.es

Luisa María Meseguer Pérez

luisameseguer@correo.ugr.es

## Abstract

Presentamos un modelo de clasificación de imágenes de mosquitos entre cuatro clases de alta importancia epidemiológica. Utilizamos el dataset de la plataforma de ciencia ciudadana Mosquito Alert [2] y proponemos dos soluciones para el problema: una basada en Bag of Words con descriptores SIFT y otra basada en Deep Learning con distintas redes convolucionales del estado del arte. Además, experimentamos añadiendo un preprocesado a las imágenes que consiste en utilizar Mask R-CNN para recortar las regiones de interés de las mismas, con el objetivo de mejorar la calidad del dataset. El modelo final de BoW con SIFT obtiene un accuracy del 53.74% y cuando aplicamos el preprocesado 54.67%. En cuanto al enfoque con Deep Learning, el mejor modelo obtiene un 82.90% y cuando aplicamos el preprocesado 80.70%.

## 1. Introducción

Los mosquitos son los animales que más muertes causan al año. Aunque los datos varían según las fuentes, se estima que estos insectos son responsables de alrededor de un millón de fallecimientos al año en todo el mundo [9].

Mosquito Alert [2] es una plataforma de ciencia ciudadana que tiene como objetivo detectar y controlar la expansión de ciertas especies invasoras de mosquitos en España y Europa. Su base de datos [3] recoge imágenes tomadas por ciudadanos y científicos colaboradores de las especies *aedes aegypti* (mosquito de la fiebre amarilla), *aedes albopictus* (mosquito tigre), *aedes japonicus* (mosquito del Japón), *aedes koreicus* (mosquito de Corea) y *culex pipens* (mosquito común). Dichas especies están consideradas como importantes vectores en la transmisión de enfermedades como el Dengue (arbovirus), Chikungunya (arbovirus), Zika (arbovirus), Fiebre amarilla (arbovirus), Fiebre del Nilo Occidental (arbovirus) o Dirofilariasis (nemátodo parásito) [1, 4].

Por este motivo, es de vital importancia que la comunidad científica dedique esfuerzos en el control de dichas especies y así evitar epidemias tal y como se hizo en

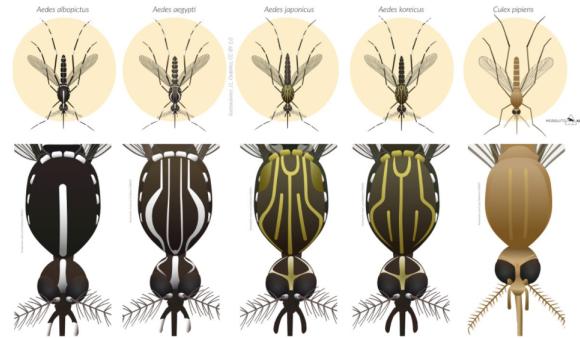


Figure 1. Tipos de Mosquitos

Fuerteventura en 2017 [5].

En este trabajo, nuestro objetivo es obtener un modelo de clasificación que sea capaz de etiquetar correctamente imágenes como las que se encuentran en el dataset de Mosquito Alert. Los tres principales retos que presenta esta tarea tienen que ver con la naturaleza de dicha base de datos.

En primer lugar, tenemos 2205 fotografías en total. Al ser los mosquitos en general parecidos entre sí y haber mucha variabilidad entre los ejemplos, corremos el riesgo de no tener suficientes datos como para entrenar eficazmente nuestros modelos, especialmente las redes profundas con una gran cantidad de parámetros. Por otro lado, las clases están fuertemente desbalanceadas. Mientras que las especies *aedes albopictus* y *culex pipens* cuentan con más de 1000 ejemplos, *aedes aegypti*, *aedes japonicus* y *aedes koreicus* no llegan a los 100. Por último, la gran mayoría de fotografías no son imágenes profesionales tomadas a los mosquitos bajo un microscopio o enfocadas directamente al tórax, que es la región de mayor importancia para la clasificación, sino que contienen una diversa cantidad de elementos extra que hacen que sólo una pequeña región de cada ejemplo contenga realmente al mosquito. Además, a veces el mosquito directamente no aparece y en su lugar, se ven, por ejemplo, solo las picaduras sobre la piel, añadiendo así mucha variabilidad.



Figure 2. Imágenes Difíciles

Para abordar estos desafíos, hemos implementado estrategias como el aumento artificial de imágenes, conocido también como "data augmentation", y hemos integrado un procesamiento previo a través de un modelo de detección de objetos, específicamente Mask R-CNN [7], que se enfoca en recortar las áreas donde aparecen los insectos. Utilizando estas dos versiones del dataset, hemos llevado a cabo una serie de experimentos.

Nuestro primer enfoque consistió en el desarrollo y entrenamiento de un modelo de Bag of Words. Dado que los mosquitos suelen clasificarse según los patrones de su tórax, la idea de interpretar estos patrones como elementos distintivos para formar un "vocabulario" de palabras visuales se mostró como una opción prometedora.

Además, exploramos la eficacia de varios modelos de redes neuronales convolucionales para la clasificación de imágenes. Entre estos se incluyen algunos ya establecidos y reconocidos en el campo, como LeNet, AlexNet, VGGNet y ResNet, así como otros modelos que hemos diseñado y entrenado desde cero.

## 2. Trasfondo

Para entender el proceso seguido en el desarrollo de este trabajo es necesario entender los conceptos enumerados a continuación:

**1. Data Augmentation:** La técnica de data-augmentation [16] no solo permite expandir los conjuntos de datos existentes, sino que también ayuda a mejorar la robustez y la generalización de los modelos de ML. Al introducir variaciones en los datos de entrenamiento, los modelos pueden aprender a no depender de características irrelevantes, lo que los hace más capaces de manejar datos del mundo real que pueden variar de maneras inesperadas. Por ejemplo, en el procesamiento de imágenes, técnicas como el giro, el recorte, y la alteración de la iluminación pueden generar nuevas imágenes a partir de una sola imagen original, lo que ayuda a los modelos a reconocer objetos bajo diferentes condiciones.

Además, el aumento de datos es especialmente valioso en campos donde los datos son escasos o costosos de obtener, como en la medicina o la entomología<sup>1</sup>. En

estos casos, generar imágenes médicas sintéticas, por ejemplo, puede permitir a los investigadores entrenar modelos de ML sin necesidad de recopilar grandes cantidades de datos sensibles.

2. **Detección:** La detección de objetos [8] en visión por computadora es un proceso que implica identificar y localizar la presencia de objetos específicos en imágenes o videos. Se utiliza un modelo de aprendizaje automático, como redes neuronales convolucionales, para realizar predicciones sobre la ubicación y la clase de los objetos en la imagen. El resultado final muestra cajas delimitadoras alrededor de los objetos detectados junto con sus etiquetas correspondientes, siendo una tecnología fundamental con aplicaciones en vigilancia, conducción autónoma, reconocimiento facial y diversas áreas.
3. **K-Means:** Es un algoritmo sin parámetros que agrupa las características sin etiqueta en clústeres. Esto se consigue buscando un centroide que minimice la distancia al resto de puntos. Esta distancia se calcula con la ecuación euclídea, considerada una métrica estándar para problemas geométricos:

$$\text{Dist}(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (1)$$

4. **SIFT:** Es un algoritmo de detección y descripción de características invariante a traslaciones, rotaciones, escalado y variación de iluminación. Su funcionamiento se basa en la construcción de un espacio de escala, determinado por la aplicación de operaciones de escala en la Diferencia de Gaussianas. Posteriormente, identifica puntos de interés al asignarles orientaciones del gradiente como descriptores, lo que permite capturar información relevante independientemente de la orientación y el tamaño de la característica en la imagen.

5. **Support Vector Vlassifier (SVC):** Se trata de un algoritmo de aprendizaje supervisado utilizado para la clasificación de datos. Su objetivo es encontrar el hiperplano óptimo que maximiza la separación entre diferentes clases en un espacio multidimensional. En términos sencillos, un hiperplano es una superficie de decisión que divide el conjunto de datos en regiones correspondientes a diferentes categorías. Lo distintivo del SVC radica en su capacidad para manejar datos no linealmente separables mediante el uso de funciones de kernel, que transforman el espacio de características original a uno de mayor dimensionalidad.

El proceso de entrenamiento del SVC implica la identificación de vectores de soporte, que son aquellos puntos más cercanos al hiperplano separador y los que se

<sup>1</sup>La entomología es la rama de la zoología que se dedica al estudio de

los insectos.

tienen en cuenta para maximizar el margen entre estos y el hiperplano.

6. **BoVW:** Es un algoritmo de procesamiento de imágenes para representarlas en forma de vector de características visuales. Cada imagen cuenta con puntos clave o características locales identificadas como regiones con información local relevante, detectada mediante cualquier detector/descriptor, en nuestro caso SIFT. Estas características se dividen en varios grupos utilizando el algoritmo de agrupación K-means, donde cada grupo (cluster) tendrá características con descriptores similares. Dados estos clusters podemos utilizar la técnica de cuantización de vectores (VQ) para así representar cada cluster en forma vectorial, es decir, en una palabra visual. Esta palabra visual identifica un patrón local específico, y el conjunto de todos nuestros clústeres formará el vocabulario de palabras visuales que identificará todos los tipos de patrones que se encuentren en cada imagen que queramos clasificar. Así, podremos obtener la información vectorial de una imagen como la cantidad de respuesta a todos los patrones de nuestro vocabulario de palabras, y así, usar esa representación para un clasificador que trata vectores de características, en nuestro caso Support Vector Classifier(SVM). Ver figura 3.

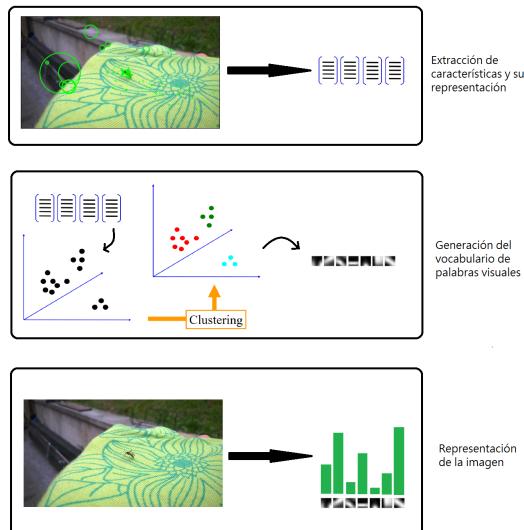


Figure 3. Esquema BoVW

7. **Regiones de Interés:** Las Regiones de Interés son fundamentales para la detección precisa de mosquitos. Una ROI es una porción específica de una imagen que contiene información relevante, en este caso, un

mosquito. El propósito de identificar las ROIs es permitir que los clasificadores se concentren exclusivamente en estas áreas, lo que podría mejorar significativamente la precisión de la clasificación. Usando el modelo Mask R-CNN se pueden detectar con precisión las ROIs, asegurando que se ajusten a la ubicación y tamaño exactos de los mosquitos en las imágenes. Esto es crucial para lograr un análisis detallado y efectivo de los datos de los mosquitos.

8. **CNN:** Las redes neuronales convolucionales (CNN) son una categoría especializada de redes neuronales que se utiliza principalmente en tareas de procesamiento de imágenes y visión por computadora. Son una variante del concepto general de redes neuronales, que son un subconjunto del aprendizaje automático y el núcleo de los algoritmos de aprendizaje profundo.

Una CNN se compone de una entrada y una serie de capas, que incluyen capas convolucionales, capas de agrupación y capas totalmente conectadas. Cada capa en la CNN tiene una función específica:

- **Capa Convolucional:** Es el bloque principal de una CNN. Emplea un conjunto de filtros para escanear una imagen y detectar características como bordes y texturas. Los filtros son matrices más pequeñas que se aplican a porciones de la imagen de entrada para generar un mapa de características.
- **Capa de Agrupación:** Se utiliza para reducir la dimensionalidad de cada mapa de características y conservar las características más importantes. Esto ayuda a disminuir el costo computacional y reduce la posibilidad de sobreajuste.
- **Capa Totalmente Conectada:** Se usa principalmente para la clasificación y toma las características de las capas anteriores para finalmente realizar la clasificación requerida.

Estas redes se entrena mediante un proceso de retroalimentación y ajuste de los pesos, similar a cualquier otra red neuronal. Utilizan un algoritmo de optimización, como el descenso de gradiente estocástico, para minimizar una función de pérdida y ajustar los pesos durante el entrenamiento.

Las CNNs han demostrado ser extremadamente efectivas en tareas como la clasificación de imágenes, el reconocimiento de objetos y también en otros dominios como la interpretación de lenguaje natural. Este rendimiento superior se debe a su capacidad para aprender automáticamente y generalizar características de los datos [11, 12].

### 3. Trabajos Relacionados

El método BoW, originalmente utilizado en procesamiento del lenguaje natural para representar textos a través de la frecuencia de palabras [13], ha sido el precursor del modelo Bag of Visual Words (BoVW) en visión por computador. Este último enfoque, que describe imágenes mediante la agrupación de características visuales recurrentes, ha jugado un papel fundamental en el desarrollo de técnicas avanzadas de detección de objetos. Investigaciones pioneras en BoVW [10], [14] han impulsado la creación de modelos más sofisticados como Mask R-CNN y Faster R-CNN, que han revolucionado la clasificación y el reconocimiento de imágenes. Estos modelos han incorporado conceptos de BoVW para optimizar la precisión y eficiencia en la detección de objetos, reflejando un avance significativo en el campo de la visión por computadora.

La tarea de detección de objetos ha experimentado un gran avance en los últimos 10-15 años gracias a trabajos como [7] Mask R-CNN y [6] Faster R-CNN. El modelo que utilizamos en este trabajo trabaja sobre Faster R-CNN, cuyo esquema de funcionamiento es el siguiente:

1. **Obtención del mapa de características:** La imagen pasa a través de alguna red pre entrenada como ResNet, VGG16 o similares para obtener un mapa de características de alto nivel.
2. **Region Proposal Network (RPN):** La RPN es una red neural que opera sobre el mapa de características para generar propuestas de regiones candidatas que podrían contener objetos. En cada posición del mapa de características, la RPN predice un conjunto de anclas (cajas delimitadoras predefinidas con diferentes escalas y aspectos) y asigna a cada ancla dos valores: la probabilidad de que la región propuesta contenga un objeto y las correcciones para ajustar la posición y el tamaño de la caja delimitadora respecto a la posición de la ancla.
3. **ROI pooling:** las cajas delimitadoras se procesan para obtener la representación final de las regiones de interés detectadas.
4. **Clasificación final:** Las RoIs procesadas se introducen en dos ramales separados, uno para la clasificación y otro para la regresión. La rama de clasificación evalúa qué objeto (o si hay alguno) está presente en la RoI. La rama de regresión ajusta las coordenadas de la caja delimitadora para que coincida mejor con los contornos exactos del objeto.

Otros enfoques que han buscado mejorar principalmente la eficiencia de Faster R-CNN se pueden encontrar en modelos como [15] You Only Look Once: Unified, Real-Time Object Detection que unifican las tareas de la RPN y el

clasificador para obtener una RPN que es específica para cada clase, lo que permite ahorrar tiempo de ejecución (a cambio de una ligera pérdida en la precisión).

### 4. Métodos

Para paliar el desbalanceo de clases, utilizamos pesos inversamente proporcionales a la frecuencia de clases. De esta manera, se dará más importancia a la correcta clasificación de las clases minoritarias. De no ser así, el modelo observaría que la mejor estrategia para disminuir el error sería predecir sólo las clases mayoritarias.

Utilizar BoVW para clasificar las distintas clases de mosquitos es una buena idea puesto que la clasificación se basa en detectar patrones característicos, principalmente en el tórax, por lo que no es tan importante su localización respecto al resto de objetos en la imagen. Si conseguimos que nuestro modelo detecte aquellos patrones característicos en el tórax de cada especie de mosquito, será capaz de identificar si en una imagen se encuentra un mosquito albopictus, aegypti, culex o japonicus/koreicus. Además, SIFT es un modelo robusto frente a transformaciones de escala, rotación e iluminación, por lo que ayuda al problema de obtener fotos desde cualquier perspectiva realizadas en sitios con iluminaciones diferentes.

Otra perspectiva para abordar el problema es Deep Learning. Estos modelos convolucionales en auge son una buena opción en la extracción automática de características a partir de imágenes, lo que es esencial para distinguir las variaciones sutiles en la apariencia de diferentes especies de mosquitos. Además, la enorme cantidad de parámetros entrenables en las redes profundas permite capturar patrones intrincados y adaptarse a la diversidad morfológica de los mosquitos. Dentro de esta opción, hemos usado tanto redes convolucionales propias, como conocidas y preentrenadas. Estas ultimas ayudan a mitigar el problema de la falta de datos suplidos con las miles de imágenes con las que han sido pre-entrenadas. Las opciones escogidas han sido:

- **LeNet:** Esta arquitectura fue diseñada para el reconocimiento de dígitos obteniendo un buen desempeño en su función. Podemos partir de esta red más básica y luego probar otras más potentes, pues reconocer el patrón del torax del mosquito puede llegar a ser una tarea similar a reconocer dígitos.
- **AlexNet:** Esta es una elección adecuada gracias a su equilibrio entre profundidad y complejidad computacional. Además, ha demostrado un buen rendimiento en tareas de clasificación de imágenes, lo cual es crucial para la identificación precisa de diferentes especies de mosquitos.
- **VGGNet:** Como siguiente opción, queremos observar el desempeño de un modelo con muchos más

parámetros que AlexNet.

- **ResNet:** El uso de este es casi obligatorio, pues representa un hito en el ámbito de la visión por computadora. Esta arquitectura de red neuronal convolucional (CNN) se distingue por su capacidad para integrar cientos, incluso miles, de capas convolucionales, superando las limitaciones de escalabilidad de las CNNs tradicionales que restringían su eficacia y profundidad.

Respecto a la detección, hemos utilizado el modelo Mask R-CNN preentrenado en el dataset COCO como detector para recortar las regiones de las imágenes en las que aparezcan los mosquitos. Este modelo es a día de hoy una de las principales referencias en cuanto a detección y segmentación de objetos se refiere. Sin embargo, el dataset de entrenamiento que utiliza no contempla la clase “mosquito”. Este inconveniente podría echar por tierra todas nuestras intenciones de utilizar Mask R-CNN como detector, ya que tampoco disponemos de un dataset adecuado para reentrenarlo para detectar mosquitos. Sin embargo, dado que no hemos encontrado un modelo que sí contemple esta clase, hemos dado con la manera de hacer que este modelo pueda recortar las regiones con mosquitos para la mayoría de nuestros ejemplos.

Durante la fase de experimentación, notamos que el módulo RPN de Mask R-CNN propone con frecuencia las zonas en las que aparecen los mosquitos, ya que detecta que en esas regiones hay “algo”. Aunque el clasificador no sea capaz de decir que esas zonas contienen a estos insectos, sí que define una bounding box a su alrededor (que es lo que necesitamos para recortar las imágenes). Si fuésemos capaces de saber de qué forma etiqueta las regiones con mosquitos, podríamos aprovechar esta información para recuperar las zonas con mosquitos.

En el siguiente apartado, mostraremos cómo hemos conseguido obtener dicha información y cómo podemos entonces utilizar las coordenadas de regiones de interés para recortar las fotos.

## 5. Experimentos

### Modelo de recorte:

El módulo RPN propone regiones interesantes independientemente de su contenido. Por este motivo, es habitual que se propongan al clasificador las zonas con mosquitos. Existe un umbral de confianza que determina qué regiones se van a dar como buenas como output final. En concreto, si el clasificador puede etiquetar estas zonas con una confianza superior al umbral, se da como correcta y se da como salida. Nótese que este modelo ofrece como salida una máscara de segmentación que nosotros no usamos. Por tanto, podríamos usar Faster R-CNN en lugar de Mask R-CNN (ya que este último es una mejora del primero), pero

no lo hacemos porque Mask R-CNN tiene una api, documentación y tutoriales más actualizados y sencillos de implementar.

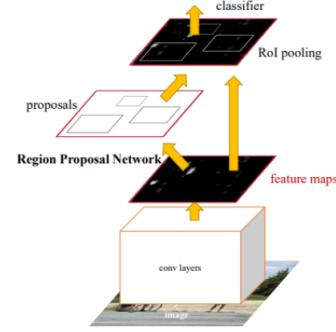


Figure 4. Arquitectura Faster R-CNN

Debido a que el modelo no está preparado para detectar mosquitos, lo habitual es que las regiones con mosquitos se etiqueten con baja confianza. Jugando con el umbral de confianza y ajustándolo cuidadosamente, podemos conseguir que estas zonas sean parte de la salida del proceso de detección.

Para ello, tomamos una serie de imágenes como muestra y ejecutamos predicciones sobre ellas con diferentes valores para el umbral de confianza. El objetivo es dar con el valor óptimo. Para que el valor sea óptimo, tiene que cumplir que:

1. Sea suficientemente bajo como para que las regiones con mosquitos (baja confianza de etiquetado) pasen el corte.
2. Sea suficientemente alto como para que la cantidad de cajas sea la menor posible y sea más sencillo intuir en cuál se halla el mosquito.

Debido a que no hay una métrica que permita evaluar si el número de cajas es óptimo, tuvimos que evaluar a mano las imágenes y las cajas propuestas por el modelo y elegir en base a nuestra percepción personal qué umbral funcionaba mejor. Concluimos que el mejor valor que podíamos usar era 0.4 (40% de confianza en el etiquetado).



Figure 5. Importancia del Umbral

Una vez tenemos el mejor escenario posible para detectar las regiones con mosquitos, tenemos que enfrentarnos a la tarea de discernir qué caja de las propuestas para cada imagen contiene al mosquito. Para ello, realizamos una predicción sobre un conjunto de imágenes de muestra del dataset y anotamos con qué clase se etiquetaba a

los mosquitos más frecuentemente. Obtuimos dos conclusiones de este experimento:

1. La etiqueta más habitual para las regiones con mosquitos es la clase “pájaro”.
2. Cuando en la imagen hay una mano sosteniendo al mosquito (algo bastante común), el modelo suele detectar la mano con precisión. Por tanto, las regiones etiquetadas como “persona” suelen representar manos sujetando el mosquito, y como el modelo sí contempla la categoría “persona”, estas cajas suelen tener un porcentaje de confianza mayor.

Considerando estas dos características, implementamos la siguiente heurística para obtener las cajas con mosquitos:

1. Si alguna caja se detecta como pájaro, quedate con esa.
2. Si no hay ninguna, quedate con la que tenga mayor confianza de etiquetado (que habitualmente será una mano con el mosquito).

Con esta heurística, pudimos ejecutar el modelo sobre las imágenes de nuestro dataset para obtener así un nuevo dataset de imágenes recortadas. Esta gráfica muestra el nº de veces que se encuentra una caja con una etiqueta concreta:

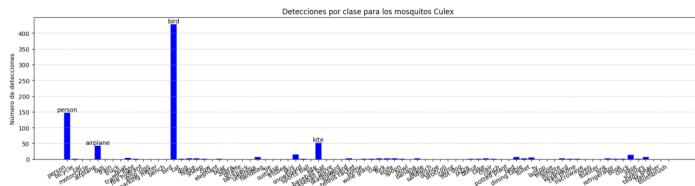


Table 1. La clase pájaro es la que más se detecta, seguida por la clase persona

Cada vez que el modelo no detecta ninguna caja en la imagen, no la contamos para el dataset, por lo que el dataset preprocesado tiene un menor número de ejemplos. Aproximadamente, el 25% de las imágenes se descartan por este motivo.

Es complicado determinar si el modelo detecta bien a los mosquitos o no, ya que la única manera que tenemos de saberlo es mirando una por una las fotografías y evaluándolas personalmente. Hemos hecho eso para una muestra de los ejemplos y calculamos que alrededor del 75% de los recortes son de buena calidad. El resto, o no contienen mosquitos o no suponen un recorte significativo sobre la imagen original.

#### Dentro de BoVW:

Hiper parámetros de SVC en BoW: Tras experimentar decremento y aumento de regularización y probar con los kernels ‘rbf’ y ‘poly’, la mejor combinación que encontramos fue dejar los Hiper parámetros por defecto de la librería sklearn.svm y especificar balanceo de pesos. Se obtuvo un accuracy final del 54.67%.

SVC detalles	accuracy
por defecto	53,74
menor regularización	50,89
kernel poly degree=3	53,74
kernel poly degree=5	51,6
kernel poly degree=2	53,02
con segmentación	54,67

Table 2. Accuracy en variaciones BoVW

En la curva de aprendizaje obtenida, al comienzo es capaz de clasificar en training hasta un 80% de accuracy pero eso significa sobreajustarse a los datos, por lo que en validación obtiene un accuracy muy bajo. Conforme obtenemos más datos, el accuracy de training va disminuyendo, pues clasificar más ejemplos es más difícil, pero el accuracy de validation va aumentando, ya que está aprendiendo a clasificar.

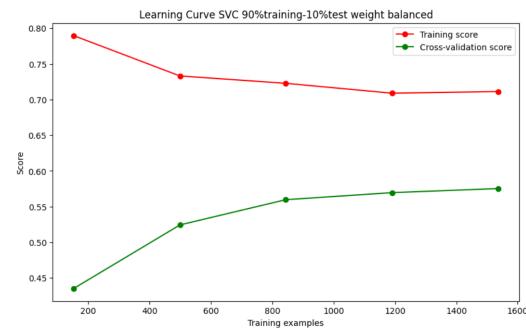


Figure 6. Curvas de aprendizaje BoVW

No existe una gran mejoría al aplicar el recorte de imágenes. Esto se debe a 2 principales motivos:

1. La robustez de SIFT frente a escalas hace que, al realizar un zoom al mosquito, no se detecten más keypoints de los que ya había antes (aunque si debería ayudar a eliminar keypoints no relacionados).

2. También aunque se haga una detección, seguimos encontrando keypoints que no pertenecen al mosquito como vemos en la figura 6.

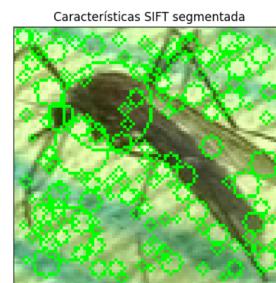


Figure 7. Características SIFT

Tampoco existe una mejoría al aplicar data augmentation. Debido a la robustez de SIFT frente a escalas y rotaciones, lo único que conseguimos es duplicar datos, lo cual no aporta ninguna robustez y capacidad de generalización al modelo.

Dentro de Deep Learning, hemos probado las siguientes arquitecturas, sin optimizar en profundidad sus modelos e hiperparámetros:

- **ResNet18 sin preprocessado:** Obtiene el mejor resultado, un 82.90% de accuracy.

- **ResNet18 con preprocessado:** El preprocessado ha empeorado un poco el resultado, con un 80.70%. Nuestra hipótesis es la siguiente: al realizar una detección previa, aquellas imágenes que no hayan encontrado un objeto, debido a que no superan el umbral de confianza, se descarta del dataset. Esto provoca que se entrene al modelo con menos imágenes y la calidad del entrenamiento de este modelo empeore.

- **CNN1:** Es una arquitectura propia con 51M de parámetros que se ha ajustado al problema con un 69.8% de accuracy. Supera a otras arquitecturas conocidas y potentes como AlexNet y VGGNet como veremos a continuación. A pesar de que AlexNet tenga un número de parámetros similar a esta, ha demostrado un rendimiento menor. AlexNet y VGGNet contienen pesos preentrenados que deben modificar en pro de ajustarse a nuestro problema, por lo tanto, si su dataset ImageNet es muy distinto al nuestro, MosquitoAlert, necesitará mayor tiempo de ejecución que nuestra propia red sin preentrenar para conseguir resultados similares.

- **LeNet:** Una red simple que ha podido trasladar el problema de reconocimiento de dígitos a la clasificación de mosquitos mejor que otros modelos más complejos, con un 60.2% de accuracy.

- **CNN2:** Es una arquitectura propia con 406K parámetros, que consideramos no contiene la complejidad suficiente para adaptarse a este problema, pues obtiene un 45.4% de accuracy.

Estas han sido las conclusiones principales de los experimentos. En la tabla 3 observamos el rendimiento de otros modelos como AlexNet, VGGNet y BoVW

Modelo	¿Pre entrenado?	Nº de parámetros	Accuracy final	Ránking
CNN 1	No	51M	69,80%	3
CNN 2	No	406K	45,40%	7
LeNet	Sí	5,6M	60,20%	4
AlexNet	Sí	57M	43,40%	8
VGGNet	Sí	134M	43,40%	8
ResNet18 (sin preprocessado)	Sí	11,7M	82,90%	1
ResNet18 (con preprocessado)	Sí	11,7M	80,70%	2
BoVW (sin preprocessado)	No	-	53,74%	6
BoVW (con preprocessado)	No	-	54,67%	5

Table 3. Conclusión

## 5.1. Dataset

Como se ha explicado en la introducción, el dataset de mosquitos con el que trabajamos tiene una serie de problemas. Son imágenes no profesionales y con desbalanceo de clases. Como medidas contra ello, hemos implementado el recorte. Pero no todo es tan bonito, pues al segmentar encontramos distintas situaciones comprometidas que pueden empeorar el rendimiento de nuestros modelos, como se ve en la figura 9:



Figure 8. Recorte correcto



Figure 9. Recorte incorrecto

## 6. Conclusiones

Tras los experimentos realizados, podemos llevar a cabo una comparación entre las propuestas de Deep Learning y BoVW atendiendo al accuracy final de dichos modelos. Como ya hemos visto, el mejor modelo de DL (ResNet18 fine-tuneado) se proclama como nuestra mejor propuesta para resolver este problema.

Nuestra principal hipótesis acerca de por qué BoVW no es mejor que DL es que las imágenes de mosquitos son tomadas desde perspectivas muy distintas, por lo que los patrones que BoVW detecta puede cambiar mucho en función de esta. Con más imágenes, es posible que BoVW pueda alcanzar a DL, aunque los modelos de DL también se benefician de contar con conjuntos de entrenamiento más grandes.

En cuanto al preprocessado de imágenes, no hemos conseguido aumentar el rendimiento de la red con el nuevo dataset. Sin embargo, podemos extraer algunas hipótesis y conclusiones interesantes sobre este experimento: El modelo sin recortar ofrece mejores resultados en un factor x1.03, pero el modelo recortado es más rápido en un factor de x2.6. Es decir, hemos conseguido reducir el tiempo de

entrenamiento con una pérdida en el rendimiento bastante leve. Este dataset recortado tiene imágenes más pequeñas (a priori, menos información) y, como se ha comentado en el apartado de recorte de las fotografías, el nº de ejemplos se ha reducido por algunos factores asociados con el procedimiento de detección. Con esta información en la mano, podemos afirmar que, si el rendimiento no ha bajado demasiado, el nuevo dataset ha conseguido paliar casi por completo la reducción de los ejemplos con una mejora en la calidad de los mismos. Por tanto, consideramos que es una buena idea preprocesar las imágenes en contextos en los que los ejemplos contienen mucho ruido y concluimos que merecería la pena dedicar esfuerzos en elaborar un detector que sí esté especializado en detectar mosquitos para resolver este problema con más precisión que nuestras propuestas.

Atendiendo a los resultados en accuracy de los modelos, concluimos que la mejor opción para resolver este problema es mediante Deep Learning con ResNet sin preprocesado de imágenes. Sin embargo, vemos importante tener presente la ventana de oportunidad para la mejora que ofrece el preprocesado si se consigue dar con un detector de mosquitos especializado en esta tarea.

## References

- [1] Enfermedades que transmiten. <https://www.mosquitoalert.com/enfermedades-que-transmiten/>, 2024. 1
- [2] Mosquito alert. <https://www.mosquitoalert.com>, 2024. 1
- [3] Mosquito alert image dataset. <https://www.mosquitoalert.com/en/mosquito-images-dataset/>, 2024. 1
- [4] Mosquitos transmisores de enfermedades. <https://www.mosquitoalert.com/sobre-mosquitos/mosquito-transmisores-de-enfermedades/>, 2024. 1
- [5] Se confirma la erradicación del mosquito aedes aegypti de fuerteventura. <https://www.ricet.es/noticias/se-confirma-la-erradicacion-del-mosquito-aedes-aegypti-de-fuerteventura>, 2024. 1
- [6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Faster r-cnn. <https://arxiv.org/abs/1506.01497>, 2017. 4
- [7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. <https://arxiv.org/abs/1703.06870>, 2017. 2, 4
- [8] IBM. Image segmentation. <https://www.ibm.com/topics/image-segmentation>, 2024. 2
- [9] ISGlobal. Mosquito: el animal más letal del mundo. <https://www.isglobal.org/-/mosquito-el-animal-mas-lethal-del-mundo>, 2023. 1
- [10] Abdul Amir Abdullah Karim and Rafal Ali Sameer. Image classification using bag of visual words (bovw). *Al-Nahrain Journal of Science*, 21(4):76–82, 2018. 4
- [11] Naeemullah Khan. *Artificial Intelligence and Machine Learning with Python*. Trinity Term, Department of Engineering Science, University of Oxford, 2021. 3
- [12] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, Cambridge CB3 0FB, U.K., 2006. 3
- [13] Chirag Mehta. Us presidential speeches tag cloud. <https://chir.ag/projects/preztags/>, 2007. 4
- [14] Wisam Abdulazeez Qader, Musa M. Ameen, and Bilal I. Ahmed. An overview of bag of words: Importance, implementation, applications and challenges. In *Proceedings of the Fifth International Engineering Conference on Developments in Civil Computer Engineering Applications*, Erbil, Iraq, June 2019. 4
- [15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. 4
- [16] Amazon Web Services. What is data augmentation? <https://aws.amazon.com/es/what-is/data-augmentation/>, 2024. 2