Capítulo 5

Experimentos y Resultados

En este capítulo se presentan los experimentos realizados y los resultados obtenidos con distintas versiones del modelo YOLO en los datos de El Vergel y MBG-V2, con el objetivo de encontrar la mejor configuración para comparar modelos y determinar cuál es el más adecuado para la detección de criaderos de mosquitos.

5.1. Datos El Vergel

Se muestran los experimento realizados con los datos El Vergel y los resultados que se obtuvieron.

5.1.1. Experimentos

Cabe destacar que todos los experimentos fueron realizados utilizando únicamente YOLOv4, siendo este el primer modelo implementado para evaluar el desempeño con nuestros conjuntos de datos.

Evolución del desempeño del modelo con distintos subconjuntos de El Vergel

El proceso de etiquetado del conjunto de datos *El Vergel* tomó aproximadamente cuatro meses. Durante este periodo, se realizaron experimentos con el modelo YOLOv4 utilizando distintas versiones del conjunto, cada una con un número creciente de imágenes etiquetadas. El objetivo fue observar si las métricas del modelo mejoraban conforme se incrementaban los datos. Los resultados correspondientes

se presentan en esta subsección.

La evolución del número de instancias por categoría se muestra en la Tabla 5.1 y en la Figura 5.1 (el cuarto mes corresponde a todo el conjunto de datos etiquetados).

Categoría	Mes 1	Mes 2	Mes 3	Mes 4
Imágenes etiquetadas	143	2002	4121	5174
Tinaco	166	2430	4976	6244
Cubeta	101	1265	2326	2740
Maceta	99	889	1491	1665
Llanta	25	314	590	744
Charco	60	419	1065	2803

Tabla 5.1: Cantidad de instancias etiquetadas por categoría en cada mes.

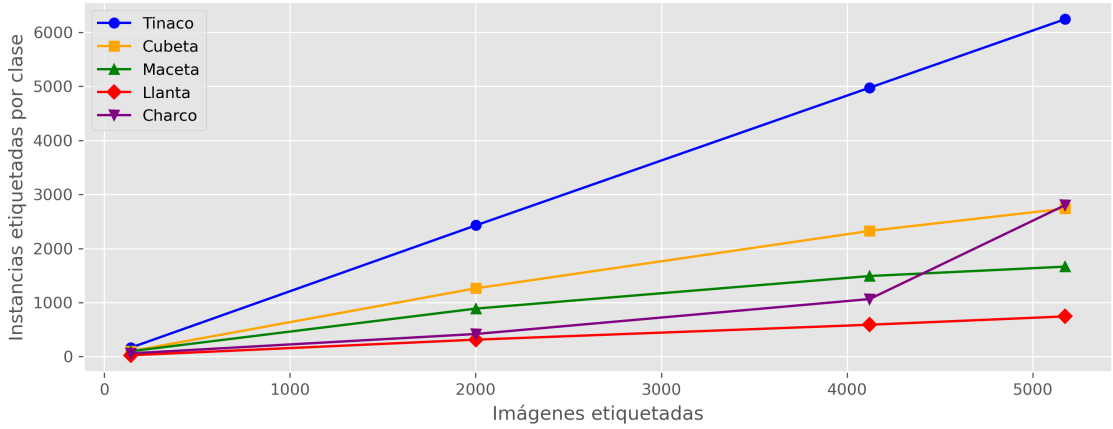


Figura 5.1: Evolución de instancias etiquetadas respecto al número de imágenes etiquetadas.

Como era de esperarse, el número de instancias etiquetadas por categoría aumenta con el incremento en el número de imágenes etiquetadas. Este crecimiento tiende a seguir un patrón aproximadamente lineal, siendo más evidente en la categoría *Tinaco*, y con comportamientos similares en *Cubeta*, *Maceta* y *Llanta*. En el caso de *Charco*, se observa un incremento especialmente significativo en el último mes del proceso de etiquetado.

Por otro lado, también se evaluó la evolución de la precisión promedio (*average precision*) por categoría conforme aumentaba el número de imágenes etiquetadas. Estos resultados se presentan en la Tabla 5.2 y en la Figura 5.2.

Categoría	Mes 1	Mes 2	Mes 3	Mes 4
Tinaco	93.86 %	98.60 %	98.41 %	98.93 %
Cubeta	58.75 %	58.48 %	60.38 %	61.24 %
Maceta	15.35 %	51.38 %	57.38 %	51.16 %
Llanta	45.83 %	74.99 %	81.10 %	87.26 %
Charco	53.33 %	68.07 %	72.79 %	81.88 %

Tabla 5.2: Precisión promedio por categoría en cada mes.

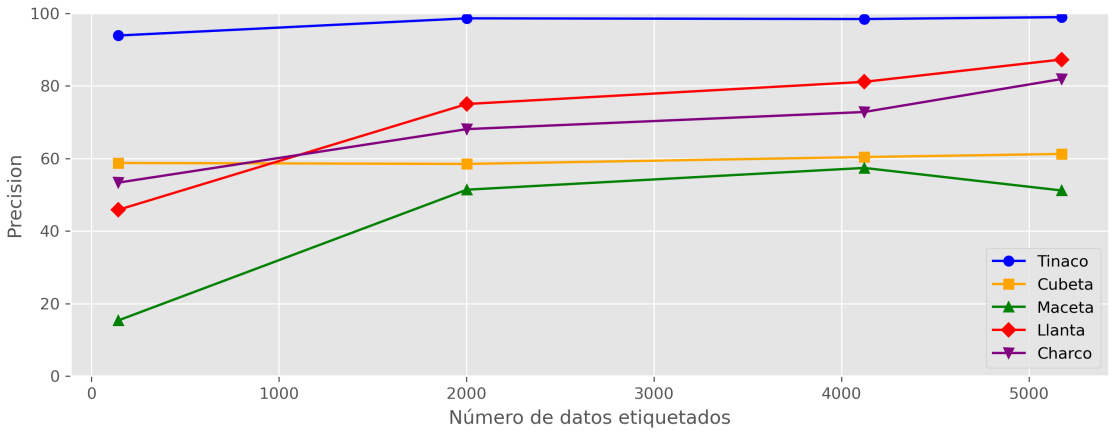


Figura 5.2: Evolución de la precisión promedio por categoría respecto al número de imágenes etiquetadas.

Como se puede observar en la Figura 5.2, el incremento en el número de imágenes etiquetadas contribuye a una mejora en la precisión del modelo YOLOv4 en todas las categorías, aunque con diferentes magnitudes.

La categoría *Tinaco* alcanzó una precisión superior al 90 % desde el primer mes, incrementando hasta casi el 99 %. En el caso de *Llanta*, se observa un incremento notable desde 45.83 % hasta 87.26 %. Para *Maceta*, la precisión pasó de 15.35 % a 57.38 % en el tercer mes, aunque luego disminuyó ligeramente. La categoría *Charco* mejoró de 53.33 % a 81.88 %.

Por el contrario, la categoría *Cubeta* presentó un crecimiento menos pronunciado, pasando de 58.75 % en el primer mes a 61.24 % en el cuarto. Esto sugiere que la mejora en precisión no fue tan significativa como en otras categorías.

En términos generales, se aprecia que a partir del segundo mes (con más de 2000 imágenes etiquetadas), la precisión mejora notablemente. Sin embargo, algunas categorías como *Cubeta* y *Maceta* no

muestran mejoras sustanciales. Esto podría deberse a que son objetos de menor tamaño y más difíciles de distinguir visualmente, lo cual complica tanto su etiquetado como el aprendizaje del modelo para detectarlos correctamente.

Finalmente se obtuvieron algunas métricas más generales para determinar el comportamiento del modelo ante el incremento de los datos, las métricas utilizadas fueron *mAP@50*, *precision* y *recall*. Los resultados pueden observarse en la Tabla 5.3 y en la Figura 5.3.

Métrica \Datos Etiquetados	**143**	**2002**	**4121**	**5174**
mAP@50 (%)	58.29	70.30	74.01	76.09
Precision (%)	75	77	75	76
Recall (%)	57	77	81	84

Tabla 5.3: Métricas globales del modelo YOLOv4 según cantidad de datos etiquetados.

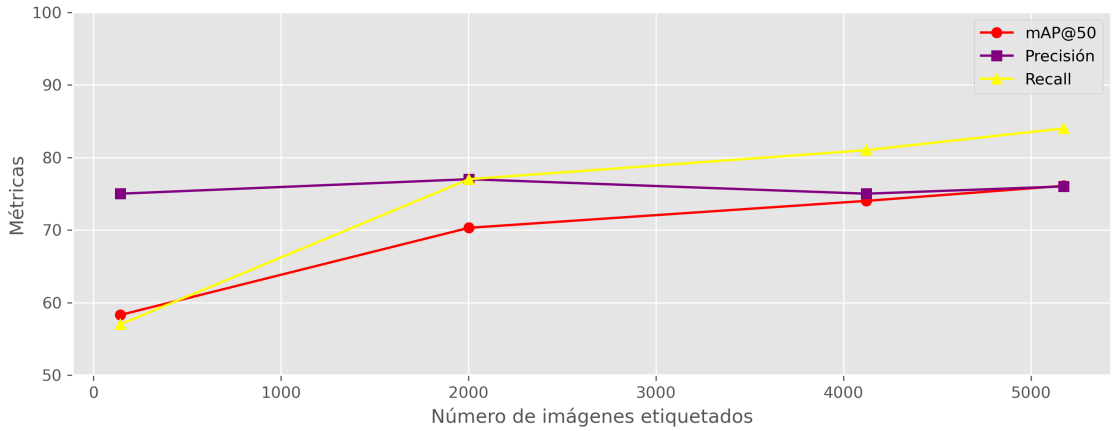


Figura 5.3: Evolución de las métricas mAP@50, precision y recall respecto al número de imágenes etiquetadas.

Los resultados muestran que el mAP@50 mejora progresivamente a medida que el número de datos etiquetados incrementa, esto se traduce a que el modelo aprende mejor a localizar y clasificar los distintos criaderos de mosquitos confome los datos incrementan. Esto es un comportamiento esperado ya que el modelo se vuelve más robusto y generalizable al ser entrenado con un mayor conjunto de datos. En cuanto a la precisión se puede observar que presenta un comportamiento oscilante incrementa en el conjunto de 2,002 imágenes pero disminuye en el de 4,121. La precisión mide la confianza con la que el modelo realiza predicciones, la oscilación puede deberse que al incrementar los datos, también incrementan las instancias

difíciles o ambiguas, lo que podrían ser las categorías de *Maceta* y *Cubeta*, esto afecta temporalmente al modelo para tener una alta precisión. El recall presenta una tendencia creciente pasando de 57 % a 84 %, el recall que nos indica el porcentaje de objetos reales que el modelo debió haber detectado. El crecimiento sostenido indica que el modelo mejora su capacidad de encontrar todos los verdaderos criaderos a medida que se entrena con más datos, lo cual es deseable en contextos como salud pública donde es preferible detectar de más que dejar pasar.

Utilizando distintas entradas del modelo

Uno de los principales hiperparámetros a configurar en los modelos YOLO es la **resolución de entrada**, que define a qué tamaño se redimensionan las imágenes antes de ser procesadas por la red. Modificar esta resolución puede impactar significativamente el rendimiento del modelo, ya que puede facilitar o dificultar la detección de objetos, dependiendo de la resolución original del conjunto de datos etiquetado.

Sin embargo, aumentar la resolución de entrada implica un mayor consumo de memoria en la GPU. Esto se debe a que una mayor cantidad de información requiere procesar mapas de activación más grandes, lo cual incrementa el número de operaciones de convolución y, por ende, el uso de recursos computacionales.

En nuestro caso, el conjunto de datos está compuesto por imágenes de alta resolución, por lo que lo ideal sería utilizar una entrada con una resolución similar. No obstante, esta configuración resulta demasiado exigente para la capacidad de memoria de nuestras GPUs. Por esta razón, se decidió evaluar dos configuraciones factibles: una resolución de entrada de 896×896 y otra de 1024×1024, ambos modelos se entrenaron con la misma configuración de hiperparámetros. Los resultados se observan en la Tabla 5.4.

Métricas	896 × 896	1024 × 1024
Tiempo entrenamiento (h)	35	20
AP Charco	81.88	81.11
AP Cubeta	61.24	68.03
AP Llanta	82.26	88.50
AP Tinaco	98.93	98.68
AP Maceta	51.16	55.60
Precision	76	78
Recall	84	85
mAP@50	76.09	78.38

Tabla 5.4: Métricas de acuerdo al modelo YOLOv4 con distintas resolución de entrada.

La Tabla 5.4 muestra que el tiempo de entrenamiento con una resolución de entrada de 1024×1024 es considerablemente menor que con 896×896, con una diferencia notable de 15 horas. En cuanto a la categoría *Charco*, el cambio de resolución no tiene un impacto significativo en el desempeño del modelo. Por otro lado, en la categoría *Cubeta*, se observa una mejora notable al usar mayor resolución, lo cual sugiere que los detalles relevantes para esta clase —caracterizada por instancias de menor escala— se preservan mejor con mayor tamaño de entrada.

En la categoría *Tinaco*, la precisión se mantiene prácticamente constante, lo cual indica que el modelo es robusto para detectar objetos de esta clase, posiblemente debido al tamaño mayor de sus instancias. En el caso de *Llanta*, se aprecia una mejora significativa al aumentar la resolución (de 82.26 a 88.50), lo que podría atribuirse a una mejor representación de los bordes o de las formas circulares. Por último, la categoría *Maceta* presenta una mejora moderada (de 51.16 a 55.60), lo que refleja cierta sensibilidad de esta clase a la resolución de entrada.

En términos generales, tanto la precisión como el *recall* mejoran con el aumento de resolución, lo cual indica que el modelo logra detectar más objetos sin incrementar sustancialmente los falsos positivos. El valor de *mAP@50* también se incrementa, reflejando un mejor rendimiento global en la detección de objetos.

Utilizando distintas particiones del conjunto de datos

Una característica fundamental en el proceso de entrenamiento de modelos es la elección adecuada de la partición del conjunto de datos. La configuración más común es una división del 80 % para entrenamiento y 20 % para validación; sin embargo, existen otras combinaciones posibles, incluyendo aquellas que incorporan un subconjunto adicional para prueba.

Con el objetivo de analizar el impacto que pueden tener distintas particiones en el desempeño del modelo, se entrenó YOLOv4 utilizando tres configuraciones diferentes: dos de ellas contemplan únicamente conjuntos de entrenamiento y validación (80 %–20 % y 70 %–30 %), y una tercera configuración que incluye un subconjunto de prueba, dividiendo los datos en 60 % para entrenamiento, 20 % para validación y 20 % para prueba. Este análisis permite evaluar cómo la proporción de datos asignada a cada conjunto puede influir en la capacidad de generalización del modelo.

Los resultados se muestran en la Tabla 5.5.

Métricas	**80 % - 20 %**	**70 % - 30 %**	**60 % - 20 % - 20 %**
AP Charco	81.11	82.41	80.95
AP Cubeta	68.03	66.62	61.12
AP Llanta	88.50	87.44	88.887
AP Tinaco	98.68	98.83	98.44
AP Maceta	55.60	55.07	50.86
Precision	78	79	72
Recall	85	84	85
mAP@50	78.38	78.07	76.05

Tabla 5.5: Métricas de acuerdo al modelo YOLOv4 con distintas particiones del conjunto de datos.

Inicialmente, debido a la cantidad limitada de datos etiquetados, se optó por una partición del conjunto de datos en un 80 % para entrenamiento y un 20 % para validación. A medida que se fueron etiquetando más imágenes y el conjunto de datos creció, se realizaron experimentos adicionales con otras particiones, como 70 % – 30 %.

Los resultados de estos experimentos muestran que las particiones 80 % – 20 % y 70 % – 30 % arrojan métricas similares. No obstante, al utilizar una división de 60 % – 20 % – 20 %, en la cual se incorpora explícitamente un conjunto de prueba, se observa una disminución en el rendimiento, posiblemente

atribuida a una menor cantidad de datos disponibles para el entrenamiento. A pesar de esta reducción, contar con una partición que incluya un conjunto de prueba es esencial para realizar una evaluación más robusta y realista del comportamiento general del modelo.

5.1.2. Resultados

Datos etiquetados

Uno de los objetivos principales fue la creación de una base de datos etiquetada utilizando imágenes del conjunto *El Vergel*, fundamental para el entrenamiento de modelos adaptados a la detección de criaderos de mosquitos con características propias de nuestra zona de estudio.

Del total de 10,098 imágenes obtenidas tras el posprocesamiento, se etiquetaron 5,174 imágenes. Las 4,924 imágenes restantes fueron descartadas por no contener ningún tipo de criadero de mosquito, ya que no aportaban información útil al modelo. Estas imágenes solían corresponder a zonas con vegetación o áreas donde no se observaban objetos de interés.

Tras un exhaustivo proceso de etiquetado, se logró anotar un total de 14,196 instancias distribuidas entre las cinco categorías definidas: *Tinaco*, *Charco*, *Maceta*, *Cubeta* y *Llanta*. La Figura 5.4 muestra la distribución de las instancias etiquetadas por categoría.

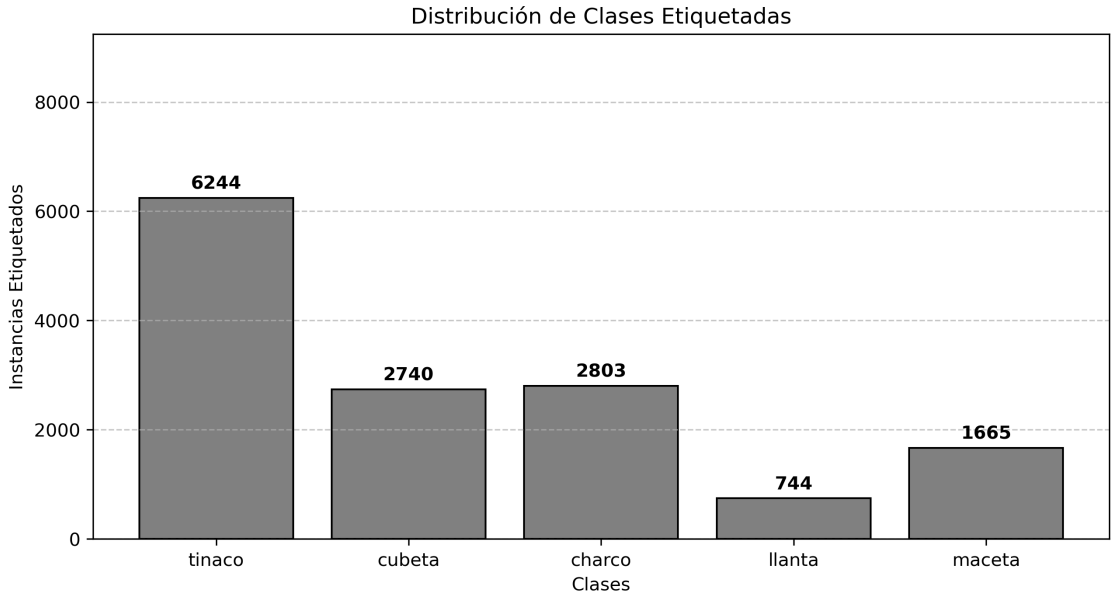


Figura 5.4: Distribución de instancias etiquetadas en el conjunto de datos El Vergel.

Como se observa en la Figura 5.4, la categoría *Tinaco* presenta el mayor número de instancias etiquetadas, con un total de 6,244. Esto se debe a que los tinacos suelen estar ubicados en los techos de las viviendas, lo que los hace fácilmente visibles en las imágenes aéreas tomadas con drones. Además, por ser elementos comunes en la mayoría de los hogares, era altamente probable encontrar al menos uno por vivienda.

Las categorías *Cubeta* y *Charco* presentaron una cantidad similar de instancias: 2,740 y 2,803 respectivamente. En el sur de México, particularmente en Tapachula, el uso de cubetas es muy común en las viviendas, ya que son utilizadas para almacenar y transportar agua, así como en diversas tareas domésticas. Por otro lado, muchas de las fotografías fueron tomadas en temporada de lluvias, lo que aumentó significativamente la presencia de charcos, un factor clave en la reproducción de mosquitos.

En cuanto a la categoría *Maceta*, se etiquetaron 1,665 instancias. Aunque no todas las viviendas contaban con macetas, aquellas que sí las tenían solían tener más de una. Finalmente, la categoría con menor cantidad de instancias fue *Llanta*, con 744 elementos etiquetados. Este tipo de objeto fue identificado tanto en las calles como en los techos de las viviendas.

Contar con una base de datos etiquetada representativa de nuestra zona de estudio es crucial para entrenar modelos capaces de detectar criaderos de mosquitos con precisión. Aunque la mayoría del trabajo se centró en este proceso de etiquetado, se enfrentaron ciertos desafíos, especialmente con las categorías más pequeñas. A pesar de que la resolución de las imágenes era alta, la altitud desde la que fueron tomadas dificultaba la visibilidad de objetos pequeños como cubetas y macetas. Desde una vista aérea, estos objetos tienden a presentar una forma circular poco distinguible, lo que dificultó tanto su identificación como su correcto etiquetado.

Por el contrario, categorías como *Tinaco*, *Llanta* y *Charco* no presentaron mayores problemas en el proceso de anotación, ya sea por su tamaño o por su forma particular, como en el caso de los tinacos y las llantas, que resultaban fácilmente identificables en las imágenes.

Comparación de los modelos

Posterior al proceso de etiquetado completo del conjunto de datos *El Vergel* y a la realización de los experimentos correspondientes, se procedió al entrenamiento de los modelos YOLO: *v4 tiny*, *v4*, *v7 tiny*, *v7 x*, *v8 tiny*, *v8 x*, *v12 tiny* y *v12 x*. Se utilizaron los pesos preentrenados de cada modelo con el

conjunto de datos COCO.

La configuración básica utilizada para el entrenamiento consistió en un número de épocas igual a 200, un tamaño de *batch* de 8 y una resolución de entrada de 1024×1024 píxeles, una partición de los datos de 60 % de entrenamiento (3,104 imágenes), 20 % de validación (1,035 imágenes) y 20 % de prueba (1,035 imágenes). Estos hiperparámetros fueron seleccionados en función de los experimentos previos, resultando más beneficiosos para la detección, y, en el caso del tamaño de *batch*, por su buen desempeño experimental.

Es importante señalar que no se realizó una búsqueda exhaustiva de hiperparámetros óptimos para cada modelo, manteniéndose por defecto aquellos no mencionados previamente. La elección de no efectuar esta búsqueda se debió al alto costo computacional que implicaría, optándose así por conservar la configuración predeterminada, la cual permite un entrenamiento estable.

Todos los modelos fueron entrenados utilizando una tarjeta gráfica NVIDIA Quadro TRX 8000, con una memoria de 48.601 GiB.

Dado que la partición del conjunto de datos se definió en 60 % para entrenamiento, 20 % para validación y 20 % para prueba, lo ideal sería mantener estos porcentajes de manera uniforme en cada una de las categorías al realizar la división. Sin embargo, este procedimiento resulta bastante complejo, ya que cada imagen puede contener múltiples categorías, lo que genera un problema que requiere el uso de metaheurísticas para ser resuelto, implicando así un mayor tiempo de programación y de cómputo.

Por tal motivo, en lugar de garantizar la proporción exacta de 60 %, 20 % y 20 % en todas las categorías, se optó por realizar la partición del conjunto de datos mediante una selección aleatoria. La distribución obtenida en los tres subconjuntos se muestra en la Figura 5.5.

Si bien la proporción establecida no se cumple con exactitud, la distribución alcanzada constituye una buena aproximación. Es probable que, aun con el uso de metaheurísticas, no se lograra una correspondencia exacta con los porcentajes deseados y que la mejora obtenida fuera marginal. Por esta razón, se consideró adecuada esta aproximación y se decidió evaluar los modelos empleando dicha partición del conjunto de datos.

Los resultados obtenidos de los datos de prueba se muestran en la Tabla 5.6.

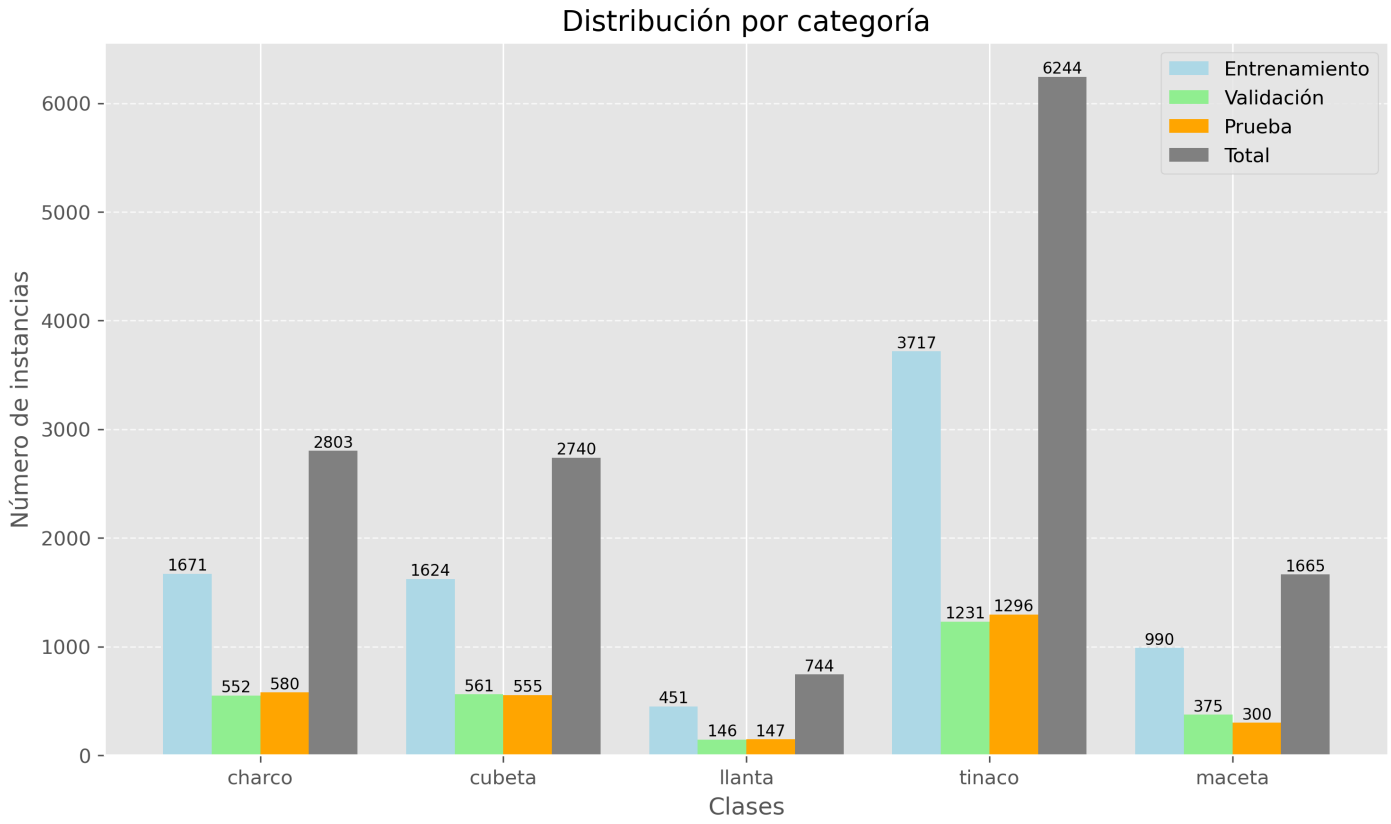


Figura 5.5: Distribución de categorías en los subconjuntos de entrenamiento, validación y prueba del conjunto de datos El Vergel.

Métrica	v4 tiny	v4	v7 tiny	v7 x	v8 n	v8 x	v12 n	v12 x
Tiempo de entrenamiento (h)	3.00	42.0	3.0	15.0	7.0	13.0	7.0	30.0
Tiempo de test (s)	18.00	37.0	35.0	58.0	30.0	61.0	30.0	89.0
Consumo de memoria (GiB)	7.00	47.00	6.0	23.0	2.0	16.0	6.0	45.0
AP charco	42.13	75.79	82.9	83.4	82.5	81.9	83.0	83.6
AP cubeta	45.55	64.36	61.5	56.7	54.1	58.6	55.3	64.2
AP llanta	85.68	79.65	85.5	86.4	80.6	84.5	79.2	82.8
AP tinaco	98.06	98.78	99.2	98.3	98.6	99.0	99.1	99.2
AP maceta	49.21	52.04	50.7	44.5	35.4	53.2	40.5	58.4
Precisión	83.00	69.00	75.2	70.4	70.3	76.0	74.2	73.7
Recall	65.00	86.00	73.5	72.9	68.4	72.4	68.5	76.4
mAP@50 test	61.21	74.13	76.0	73.9	70.2	75.5	71.4	77.7

Tabla 5.6: Resultados de entrenamiento y evaluación de diferentes modelos YOLO para los datos *El Vergel*, con los mejores valores resaltados.

En la Figura 5.6 se puden apreciar los tiempos de entrenamiento, y prueba, así como el consumo de

la memoria gráfica para el entrenamiento de los modelos. Primeramente se puede distinguir una clara diferencia en el tiempo para lo modelos con arquitecturas más grandes en comparación con los modelos con arquitecturas más reducidos, el tiempo es notable y mayor para las arquitecturas más complejas, con un total de 3, 104 imágenes de entrenamiento el modelo YOLO v4 tardó más en ser entrenado, superando las 40 horas, los modelos que tardaron menor tiempo fueron los YOLOv4 tiny y YOLOv7 tiny tardando alrededor de tres horas de entrenamiento. Es interesante notar que para el modelo YOLOv4 la diferencia en el tiempo de entrenamiento en su versión más reducida, es abismal, siendo 14 veces más rápido.

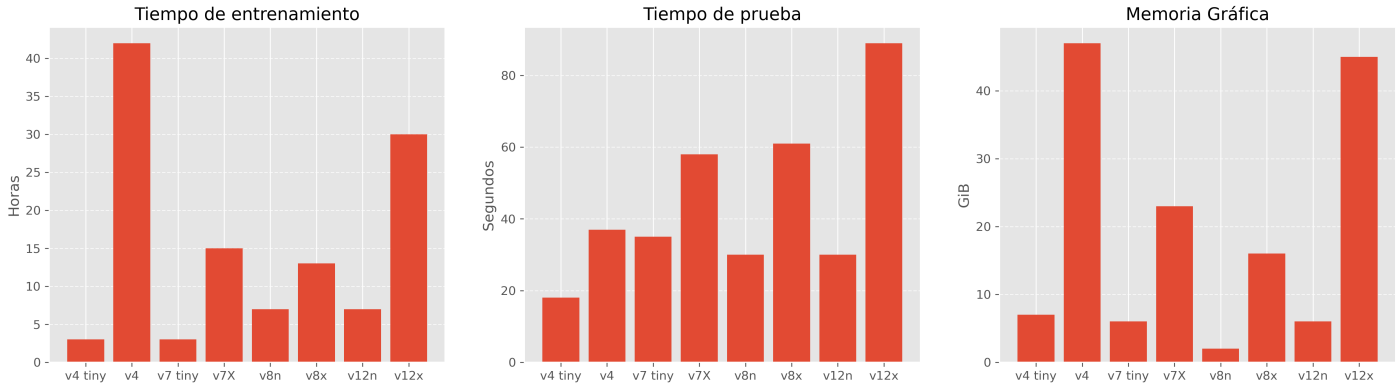


Figura 5.6: Tiempo de entrenamiento, prueba y consumo de memoria de los modelos con los datos El vergel.

Para el caso del tiempo de inferencia con un total de 1,035 imágenes nuevamente se observa una diferencia notable de los modelos con arqtuitecturas mayores a sus versiones más ligeras, siendo más rápidas las versiones ligeras, siendo la versión YOLOv4 tiny la más rápida con 18 segundos de inferencia, y la versión YOLOv4 la más rápida de los arqutiecturas más grandes. A pesar de que el tiempo de entrenamiento para el YOLOv4 es mayor en comparación de todos los modelos, su rendimiento en la inferencia es sobresaliente. El modelo más tardado fue YOLOv12x con 89 segundos.

En cunato el consumo de memoria en la tarjeta gráfica, la v4 resultó ser la más costosa con 47 GiB en promedio, ocupando casi en totalidad la memoria disponible, posteriormente le sigue v12x con 45 GiB. Y la que consumió menos fue v8n con 2 GiB, a pesar de que fue la que ocupó menos memoria, no fue la más rápida en el entrenamiento.

En cuanto a la precisión promedio o *average precision* para cada una de las categorías, se puede observar la Figura 5.7.

En 5.7 se puede observar claramente las mejores entre ambas versiones de los modelos, para v4, se puede apreciar que no existe una mejora significativa de la tiny a la v4 en las categorías *llanta*, *tinaco* y *maceta*, pero sí existe una mejor significativa en las categorías *cubeta* y *charco*, principalmente en esta última donde la versión tiny no es muy buena detectando este tipo de objetos que no tienen una forma definida, pero la versión v4 mejora considerablemente.

Para la v7 no existe mejora en las categorías de *llanta*, *tinaco* y *charco*, inclusive baja el rendimiento ligeramente en las categorías *cubeta* y *maceta*, indicando que posiblemente la versión v7 no es robusta a la detección de objetos de tamaños pequeños.

Para la v8 las categorías que no presentan mejoras considerables son *llanta*, *tinaco* y *charco* pero exite una mejora en *cubeta* y una mejora considerable en *maceta*.

Para la v12, las categorías que no presentan mejoras considerables son *llanta*, *tinaco* y *charco* pero sí presentan mejoras principalmente en *maceta* y *cubeta*, siendo este el modelo más robusto para las categorías más pequeñas.

Como se puede observar en la mayoría de los modelos, no presentan mejoras significativas en las categorías *llanta*, *tinaco* y *charco*, categorías con formas bien definidas como el caso de *llanta* y *tinaco* y de buen tamaño como *tinaco* y *charco*. Y en donde los modelos sacan mayor ventaja a sus versiones más ligeras son en las categorías más pequeñas como *cubeta* y *maceta*. Además, *tinaco* resulta ser una categoría que no presenta ningún problame para todos los modelos, y tienen buen rendimiento en estos, *llanta* y *charco* presentan en general un rendimiento bueno. Donde en general los modelos tiene un menor desempeño con en las categorías de *cubeta* y *maceta*, las categorías más pequeñas.

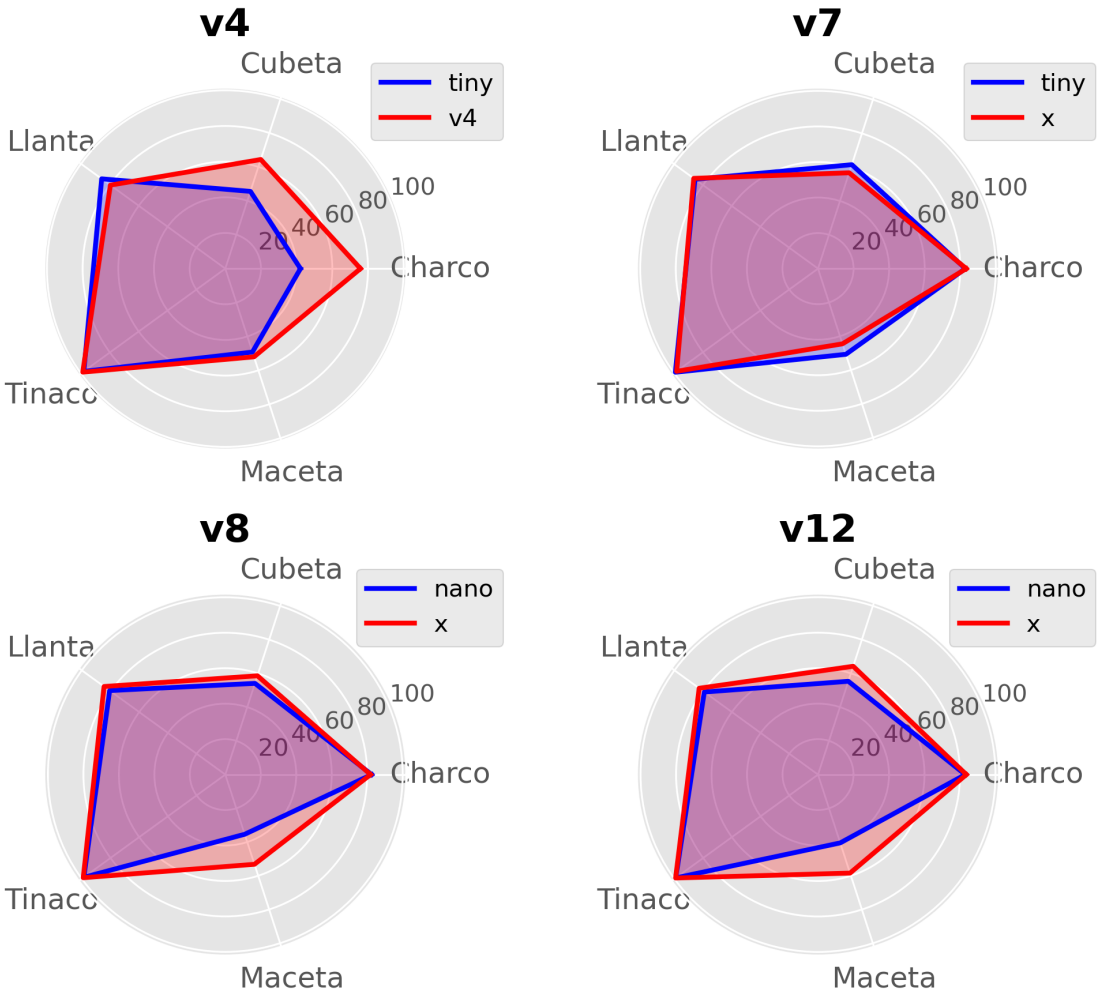


Figura 5.7: Precisión promedio por categorías en los modelos en los datos El Vergel. Las gráficas en color rojo corresponden a las arquitecturas más complejas, mientras que las de color azul representan las arquitecturas más simples.

En la Figura 5.8 se presentan los valores de *precision* y *recall* para cada uno de los modelos evaluados. El modelo YOLOv4-tiny muestra una precisión del 83 % y un recall del 65 %, lo que evidencia una diferencia considerable entre ambas métricas. Esto indica que, si bien el modelo es altamente preciso al momento de clasificar correctamente las instancias detectadas (es decir, comete pocos falsos positivos), también omite una proporción significativa de objetos que debería identificar, incurriendo en un número elevado de falsos negativos.

Por el contrario, el modelo YOLOv4 exhibe una precisión del 69 % y un recall del 86 %, invirtiéndose así los papeles respecto al modelo anterior. En este caso, el modelo detecta una mayor cantidad de

instancias relevantes (menor tasa de falsos negativos), pero con un mayor número de falsas detecciones (mayor tasa de falsos positivos). Para la problemática particular que nos ocupa, donde es más importante *no omitir objetos relevantes*, un *recall* alto resulta más beneficioso que una *precisión* elevada. Cabe señalar que ambos modelos presentan un desequilibrio significativo entre precisión y recall.

En cuanto a los modelos YOLOv7-tiny, YOLOv7x, YOLOv8n, YOLOv8x y YOLOv12x, se observa un mejor equilibrio entre *precisión* y *recall*, con diferencias poco significativas entre ambas métricas, lo cual refleja un rendimiento más balanceado.

El modelo YOLOv12n, por su parte, presenta un ligero desequilibrio similar al de YOLOv4-tiny, mostrando una mayor *precisión* que *recall* y, por ende, una tendencia a generar más falsos negativos.

En resumen, los modelos que presentan el mejor compromiso entre ambas métricas, manteniendo además valores elevados, son YOLOv7-tiny, YOLOv8x y YOLOv12x.

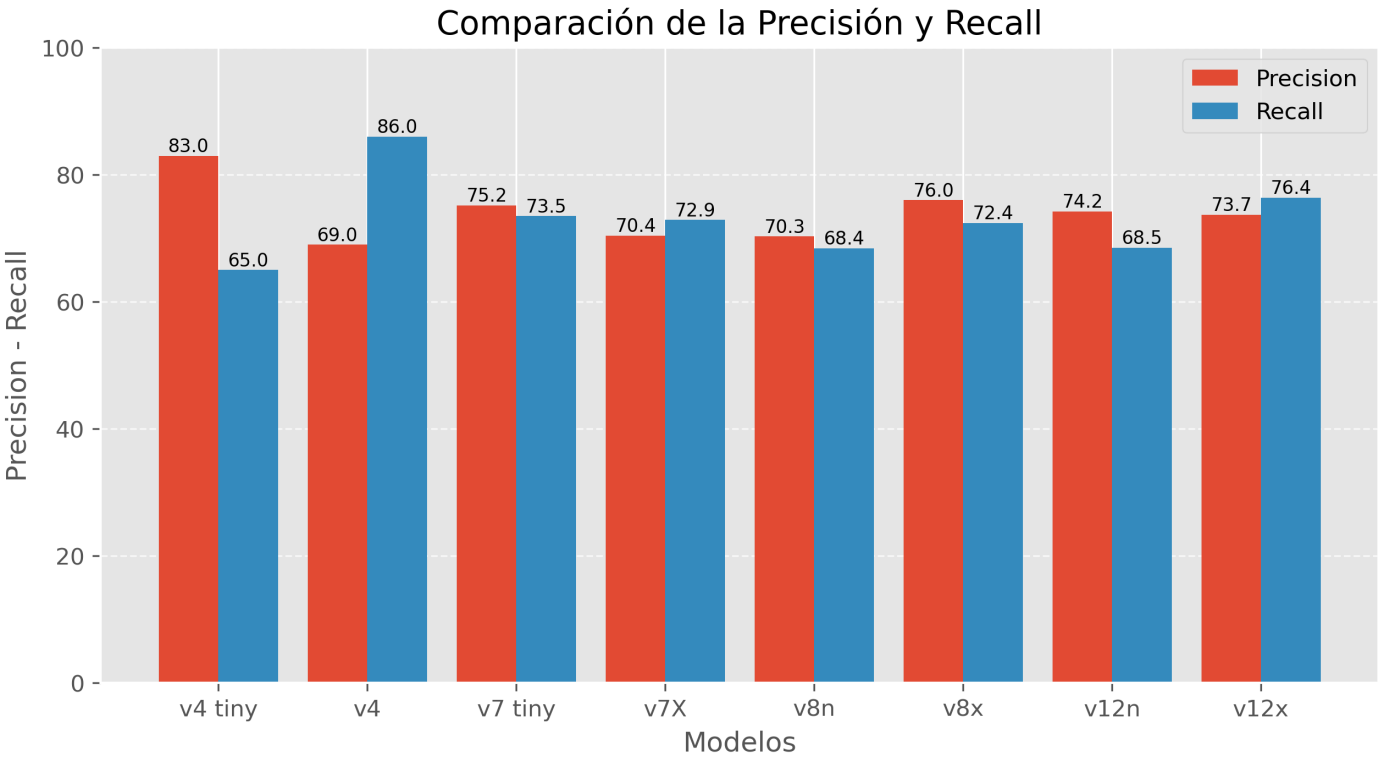


Figura 5.8: Precisión y Recall de los modelos.

En la Figura 5.9 se puede observar la métrica *mAP@50* de todos los modelos de las que podemos destacar que el mejor modelo en cuanto términos de métricas es el modelo YOLO v12x y el de menor

rendimiento es el YOLOv4 tiny.

En la tabla 5.7 se puede observar el orden de los modelos del mejor rendimiento al menor.

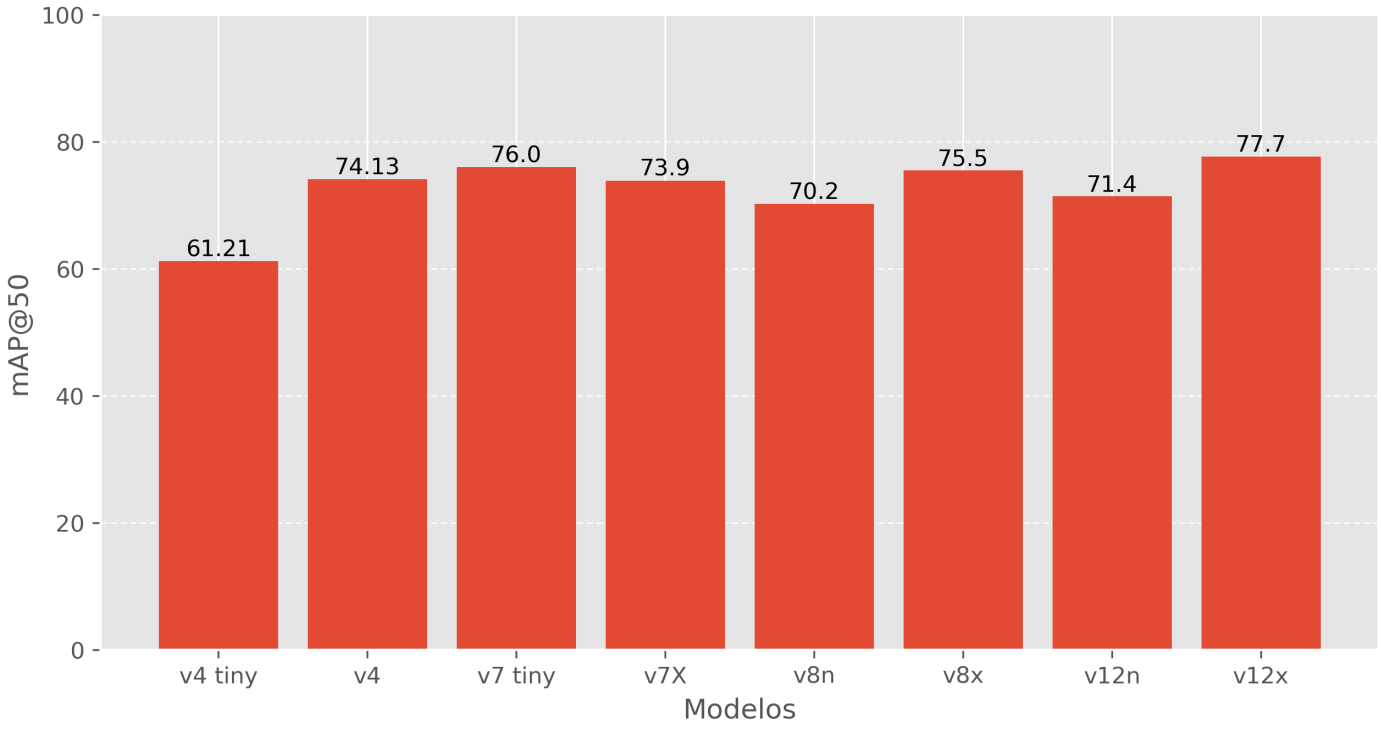


Figura 5.9: mAP@50 de los modelos.

Número	Modelos
1	v12x
2	v7 tiny
3	v8x
4	v4
5	v7x
6	v12n
7	v8n
8	v4 tiny

Tabla 5.7: Orden del mejor modelo al de menor rendimiento.

A pesar de que el modelo YOLOv12x presenta los mejores resultados en términos de métricas, su tiempo de inferencia es el más alto, con un total de 89 segundos. Dado que el tiempo de procesamiento es un factor relevante para la problemática abordada, este aspecto debe ser cuidadosamente conside-

rado. En este contexto, el modelo más adecuado, considerando tanto el rendimiento como la eficiencia temporal, sería YOLOv7-tiny, con un tiempo de inferencia de 35 segundos. Como segunda opción viable se encuentra el modelo YOLOv4, con un tiempo de 37 segundos y el mejor valor de *recall* entre todos los modelos evaluados. Una tercera alternativa es el modelo YOLOv8x, con un tiempo de inferencia de 61 segundos.

Para analizar el comportamiento del modelo en el subconjunto de prueba, se estudia la matriz de confusión correspondiente al modelo YOLOv12x (véase la Figura 5.10), el cual obtuvo los valores más altos en las métricas de evaluación. A partir de dicha matriz, se puede observar el desempeño por categoría. Para la clase *charco*, el modelo realiza un 81 % de predicciones correctas, mientras que en un 23 % de los casos confunde estas instancias con objetos de fondo. En el caso de *cubeta*, el 68 % de las predicciones son correctas, pero el 42 % se confunden con el fondo, lo cual representa una limitación importante para esta categoría. Para *llanta*, el modelo alcanza un porcentaje confiable del 83 %, mientras que para *tinaco* logra una precisión casi perfecta del 98 %. En contraste, la categoría *maceta* presenta un porcentaje de aciertos del 63 %, y un 23 % de confusiones con el fondo, lo cual también es un error significativo.

Con base en este análisis, se concluye que el modelo YOLOv12x demuestra un desempeño sólido en las categorías *tinaco* y *llanta*, un rendimiento aceptable en *charco*, y presenta dificultades importantes en la detección de las clases *cubeta* y *maceta*.

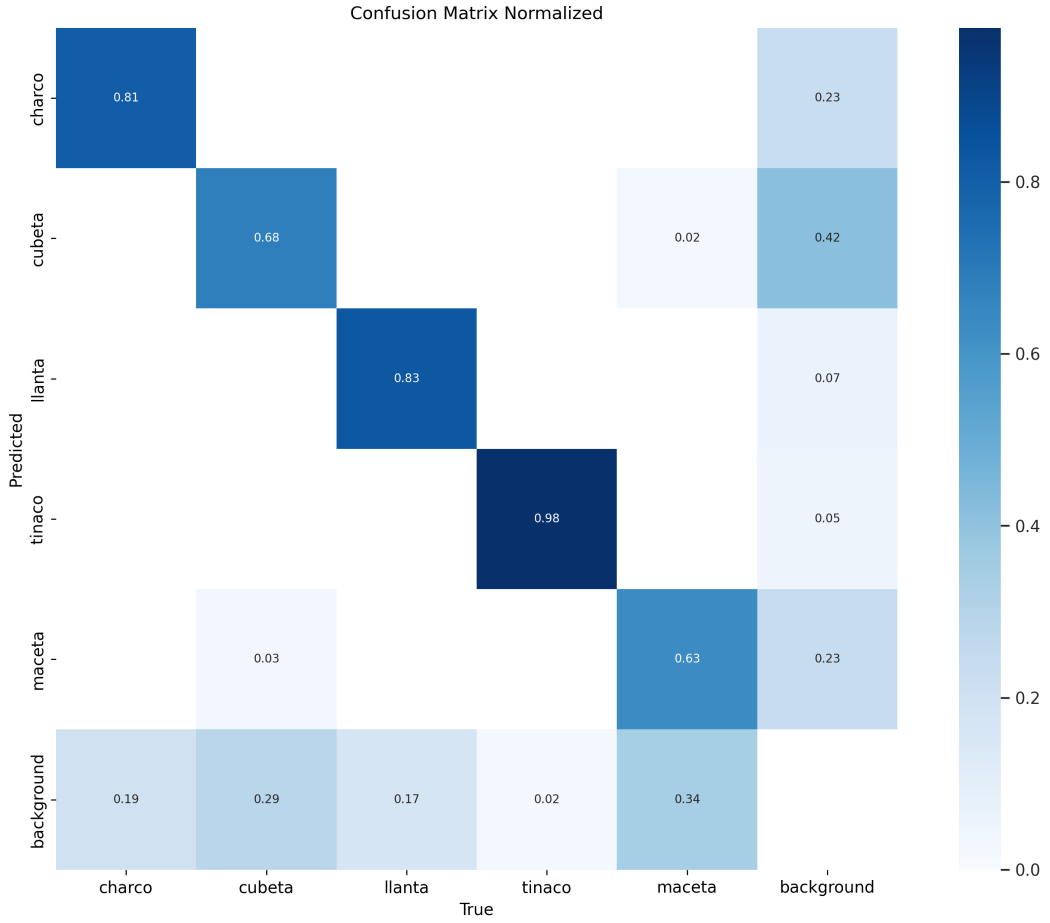


Figura 5.10: Matriz de confusión normalizado de las categorías del modelo YOLOv12x.

Los resultados de la matriz de confusión concuerdan con la gráfica 5.11, donde podemos observar que *tinaco* tiene una alta precisión de *precision* y *recall*, casi perfecto, para el caso de *charco* y *llanta* cuentan con una curva decente entre ambas métricas (0.836 y 0.828). Y para el caso de *cuebta* y maceta *maceta* las curvas son bajas (0.642 y 0.584), lo que indica que el modelo comete errores en su detección, o deja pasar objetos de estas clases.

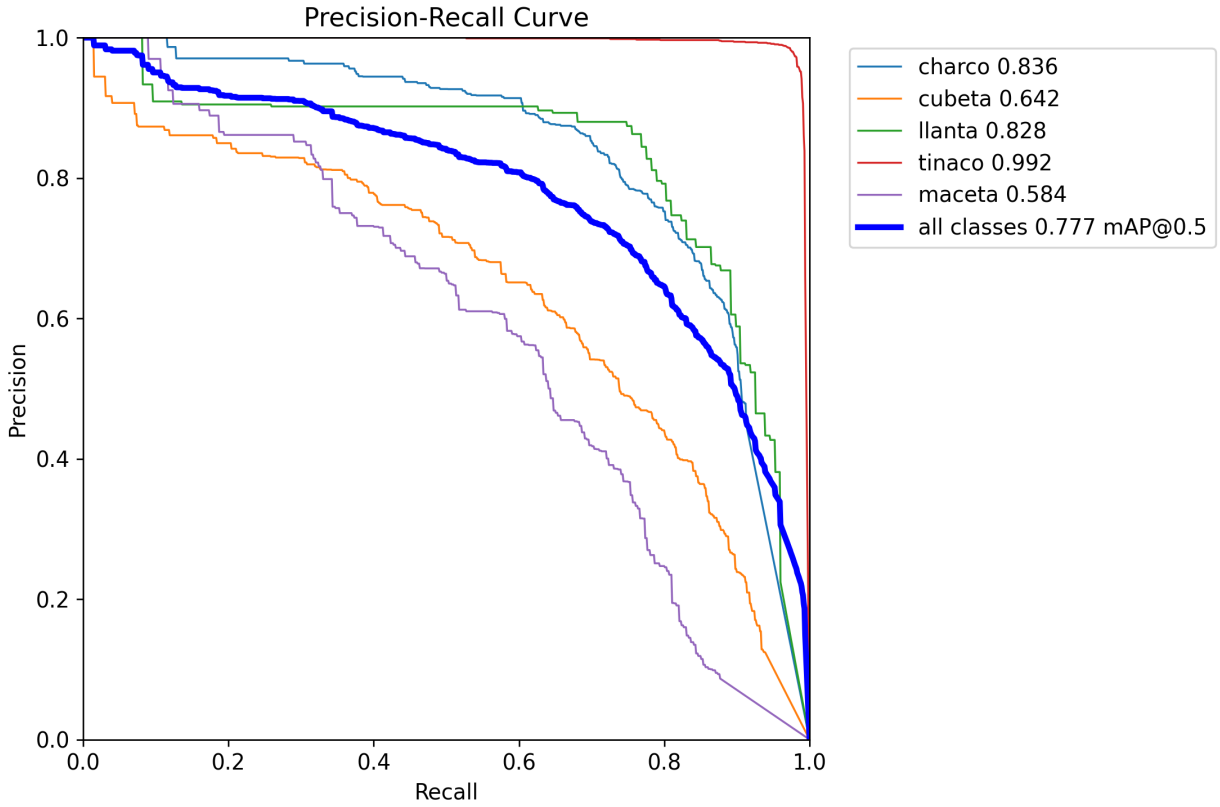


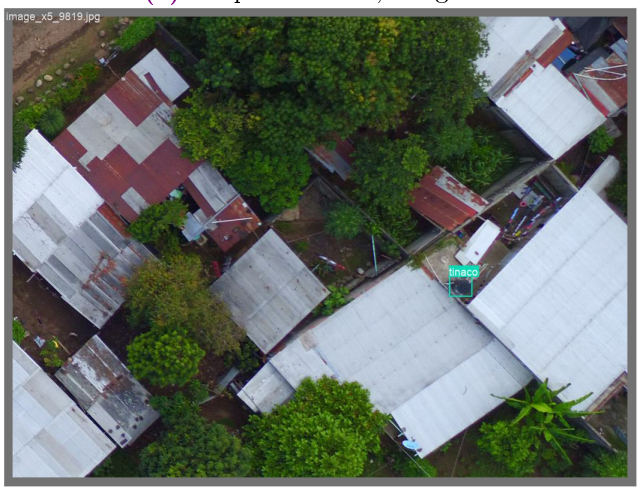
Figura 5.11: Curva Precision-Recall de YOLOv12x. Cada línea representa una categoría de objetos detectados (charco, cubeta, llanta, tinaco, maceta). La línea azul gruesa indica el promedio de todas las clases.

En la Figura 5.12 se observan algunas de las predicciones del modelo YOLOv12x, para algunas imágenes del conjunto de prueba.

Estos resultados muestran el comportamiento de los distintos modelos YOLOs probados ante las diferentes categorías consideradas en el conjunto El Vergel, de esta manera podemos conocer los puntos fuertes y débiles de cada modelo y cual resulta ser el mejor para resolver el problema de detección de criadetos de mosquitos.

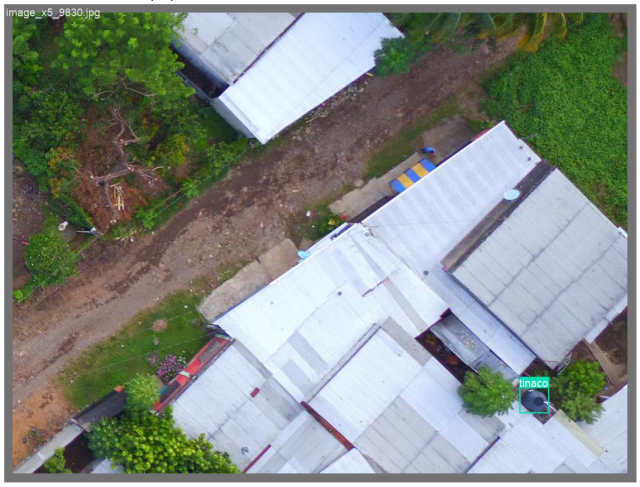
(a) Etiquetas reales, imágen 1

(b) Etiquetas predichas, imágen 1



(c) Etiquetas reales, imágen 2

(d) Etiquetas predichas, imágen 2



(e) Etiquetas reales, imágen 3

(f) Etiquetas predichas, imágen 3

Figura 5.12: Ejemplos de predicciones de YOLOv12x. Del lado izquierdo se encuentran las etiquetas reales y del lado derecho las predicciones.