Desarrollo de un modelo de predicción del estado de una cama

Juan Valeri Pomet Chamorro

20/01/2026

Índice

Desarrollo de un modelo de predicción del estado de una cama **2**
Índice **3**
Contexto **4**
Herramientas y librerías utilizadas **4**
Herramientas 4
Librerías 4
Algoritmos elegidos **5**
Pesos 6
Sesgos 6
Análisis de Datos **6**
Preprocesamiento y transformación 6
Resultados 7
Análisis de Errores 8
Arquitectura y Despliegue del modelo en Render **9**
Diagrama de arquitectura Comunicación Cliente-Servidor 9
Ciclo de Feedback 9
Pasos a seguir para conseguir una web completa **10**
Bibliografía **11**

Contexto

Este proyecto consiste en la creación de un modelo de computer vision en python que detecte la diferencia entre una cama hecha y una no hecha. El modelo será entrenado con 500 imágenes de la misma cama con sábanas beige con el mismo entorno, en este caso particular no se busca la generalización del modelo, por lo contrario, se busca que el modelo prediga de manera precisa el estado de una cama con un color de sábanas específico en un espacio definido y único.

Este modelo está diseñado específicamente para el sector de servicios de limpieza del hogar en el entorno de hoteles y servicios de alquiler de hogares. Servirá como herramienta de confirmación de la correcta realización de una cama.

Herramientas y librerías utilizadas

Herramientas

Para realizar el código del modelo se ha utilizado Visual Studio Code, se ha desplegado el modelo en una página web online utilizando la plataforma de Render.
Los distintos lenguajes de programación que han sido usados en la creación del modelo han sido python, html, css y javascript.

Librerías

flask
tensorflow
numpy
pillow
scipy
oopencv-python-headless
gdown
pathlib
cv2
os

Algoritmos elegidos

El modelo es una Red Neuronal Convolucional (CNN), ha sido entrenada utilizando aprendizaje supervisado a través de distintos algoritmos.

Se ha utilizado la convolución para detectar patrones de orden y desorden en una imagen aplicando tres capas de convolución para extraer los atributos visuales relevantes a cada imagen siendo estos en la primera capa los bordes, gradientes de intensidad, patrones repetitivos y orientaciones. En la segunda capa los pliegues, arrugas y discontinuidades. Y en la tercera el orden visual global, simetría, continuidad de superficies y la distribución del desorden.

ReLu es la función de activación más popular para redes neuronales profundas debido a su simplicidad. Muestra el valor de los datos positivos y convierte los valores negativos en cero.

La función sigmoide se utiliza para obtener una probabilidad binaria dando como resultado 0 o 1, en el modelo 0 significa que la cama está hecha y 1 que la cama no está hecha.

Se utiliza Max pooling para reducir el tamaño de las imágenes dividiendo cada una en distintas secciones y eligiendo el píxel de mayor valor de dicho área, este proceso reduce la calidad de las imágenes pero mantiene sus detalles esenciales. Al trabajar con imágenes más simples al modelo le resulta más fácil detectar patrones.

Flatten Convierte las matrices de dos dimensiones en un vector de una dimensión.

Se utiliza Dense para combinar todas las características que se han obtenido para tomar una decisión final.

La función de pérdida se utiliza para medir cómo de incorrecto ha sido la predicción comparando la predicción del modelo con la etiqueta real de la imagen.

El Optimizador Adam se ha utilizado para poder actualizar los pesos eficientemente usando gradientes.

Se ha utilizado la retropropagación para corregir el modelo. Funciona calculando el error, propagando hacia atrás y ajustando los pesos y sesgos.

Se ha utilizado ImageDataGenerator para normalizar las imágenes y crear transformaciones de ellas para mejorar la generalización del modelo.

Se utilizan distintos algoritmos de la librería de OpenCV para capturar imágenes de video, interpolar y codificar imágenes.

Se utiliza Flask para conectar las distintas partes del proyecto haciendo que el modelo pueda ser utilizado por una interfaz web y pueda predecir estados del video de la cámara en directo.

Pesos

Los pesos determinan cual es la fuerza de conexión entre las neuronas de las diferentes capas del modelo.

Sesgos

Los sesgos (Bias) permiten ajustar el comienzo de la activación de cada neurona permitiendo al modelo adaptarse a variaciones.

Análisis de Datos

Preprocesamiento y transformación

Los datos utilizados en el entrenamiento del modelo han sido 500 imágenes de una única cama realizadas con un teléfono móvil. En la mitad de ellas la cama estaba hecha y en la otra mitad deshecha.

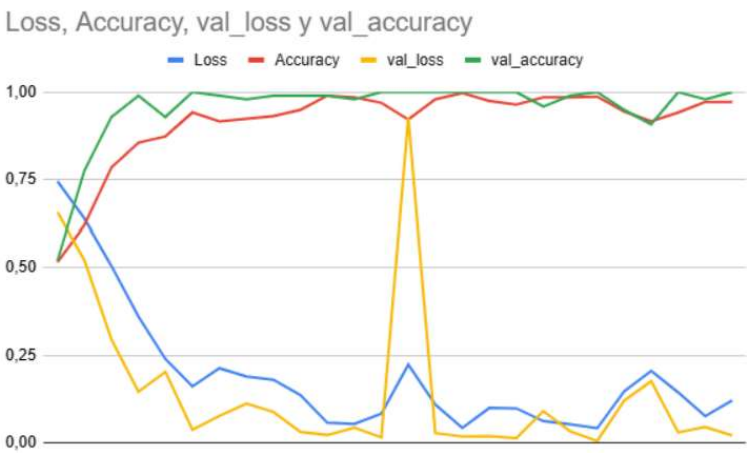
Para obtener una valoración del estado de la cama desde cualquier ángulo se han realizado fotos desde distintas posiciones y distancias hasta la cama.

Cuando se hace o se deshace una cama los patrones en las sábanas varían cada vez. Para que el modelo no diera resultados erróneos si veía un patrón distinto que con los que se había entrenado cada 10 fotos desde distintos ángulos se rehacía o se deshacía la cama para incrementar este número de patrones.

Además de entrenar el modelo con las imágenes realizadas se les han aplicado una serie de transformaciones para incrementar la precisión a la hora de predecir con condiciones distintas a las de las imágenes.
Las transformaciones aplicadas a las imágenes incluyen reescalado, rotaciones, desplazamiento de altura, desplazamiento de anchura, zoom, cambio de transparencia, cambio de brillo y flip horizontal.

Resultados

Loss	Accuracy	val_loss	val_accuracy
0,7451	0,5165	0,6567	0,5204
0,6384	0,6203	0,5215	0,7755
0,5056	0,7848	0,2953	0,9286
0,3612	0,8557	0,1462	0,9898
0,2395	0,8734	0,2028	0,9286
0,161	0,9418	0,0366	1
0,213	0,9165	0,0769	0,9898
0,1892	0,9241	0,1119	0,9796
0,18	0,9316	0,0881	0,9898
0,136	0,9494	0,0305	0,9898
0,0569	0,9899	0,022	0,9898
0,0536	0,9848	0,0433	0,9796
0,0828	0,9696	0,0151	1
0,2236	0,9215	0,927	1
0,1093	0,9797	0,0274	1
0,0422	0,9975	0,0177	1
0,0993	0,9747	0,0185	1
0,0981	0,9646	0,0132	1
0,0619	0,9848	0,0902	0,9592
0,0525	0,9848	0,0322	0,9898
0,0412	0,9873	0,005	1
0,1473	0,9443	0,1203	0,949
0,2055	0,9165	0,1758	0,9082
0,1436	0,9418	0,0291	1
0,0759	0,9722	0,0448	0,9796
0,1211	0,9722	0,0205	1



El entrenamiento se organiza por épocas, cada una muestra cuatro datos tras ser completadas, Loss indica el error del modelo por lo que cuanto más bajos sean los valores mejor son los resultados, Accuracy indica precisión o mejor dicho el porcentaje de aciertos del modelo, cuanto más alto sea este valor mejor es el modelo.
Estos dos datos tienen sus variantes de validación los cuales muestran los porcentajes de error y precisión en la validación de la predicción de los datos.

El modelo ha terminado su entrenamiento trás 27 épocas.

Trás visualizar los datos del entrenamiento se puede observar en las primeras épocas el modelo aprende rápidamente los patrones de cada grupo de imágenes. Esto es causado por fuga semántica debido a la existencia de imágenes muy similares en los datos de entrenamiento y de verificación además de ser un problema visualmente sencillo.

Trás la quinta época el modelo mantiene la precisión alta y los errores bajos, llegando a 100% de precisión en algunas iteraciones de validación, esto no es habitual en modelos grandes con un grán volumen de datos, en este caso se debe a un dataset reducido.

Aún habiendo usado un método de early stopping el modelo ha mostrado indicaciones de sobreajuste esto se puede observar claramente en la época 14 donde el valor de el error de validación escala hasta 0,927 después de haber estado descendiendo durante 6 iteraciones consecutivas. El modelo vuelve a mostrar dos picos de sobreajuste en las épocas 19 y 22.

Este sobreajuste ha sido causado debido a que el método de early stopping utilizado en el código de entrenamiento tiene “patience=5”, significando que el modelo espera a que el modelo mejore durante 5 iteraciones seguidas, si hay un dato que no indica mejora el proceso de espera vuelve a 1, en este caso el método monitorea val_loss y trás las 16 épocas obligatorias definidas no han habido 5 mejoras consecutivas hasta la época 22.
Este sobreajuste se podría haber evitado reduciendo el número de épocas mínimas obligatorias.

Análisis de Errores

El modelo, al estar entrenado con imágenes de una única cama en un entorno definido, resulta ser preciso a la hora de definir el estado de dicha cama. Cuando se realiza una prueba de definición del estado de otra cama diferente en un entorno distinto o la misma cama con un color de sábanas más oscuro el modelo empieza a fallar.
Al no estar generalizado, el modelo aun acertando con cierta exactitud falla.

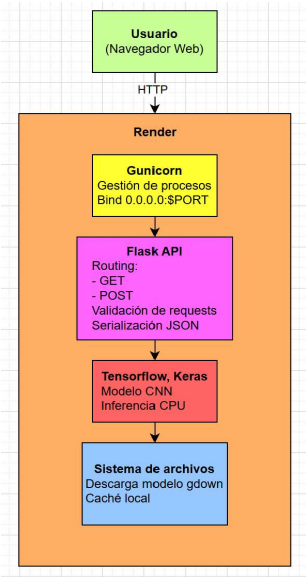
Cuando se ejecuta el modelo y la cámara está enfocando a una superficie lisa el modelo puede indicar que hay una cama hecha.

Un factor esencial que no se tuvo en cuenta a la hora de realizar las imágenes de entrenamiento fue la iluminación, se hicieron fotos únicamente con luz solar entrando por la ventana por eso si la iluminación de la habitación varía enormemente de la de las imágenes con las que se ha entrenado el modelo éste empieza a fallar.

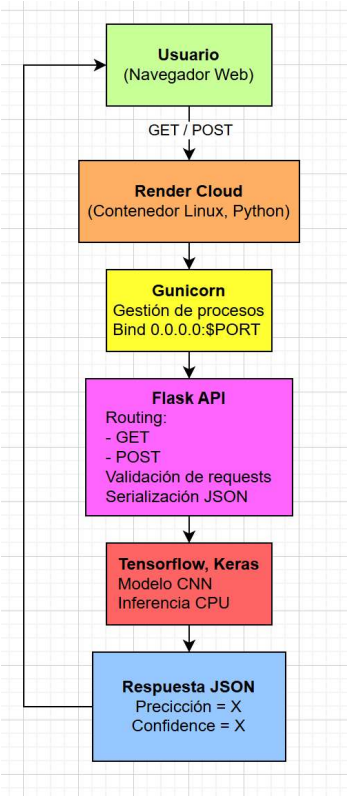
Un problema al que no se ha encontrado aparente solución es al del funcionamiento del modelo en Render.
El modelo en local funciona de manera correcta y predice con alta precisión el estado de una cama en directo ya sea la misma o una distinta en unas condiciones similares a las de las imágenes de entrenamiento. El problema llega cuando se sube el proyecto a Render, se tuvo que adaptar el código de acceso a la cámara porque el servidor no puede acceder a ella. El nuevo código permite a la web acceder a la cámara del usuario y tomar una única foto la cual es enviada al servidor vía POST y analizada por el modelo. Esta predicción nunca se completa y se muestra en pantalla un mensaje de error: “Error en la predicción”.

Arquitectura y Despliegue del modelo en Render

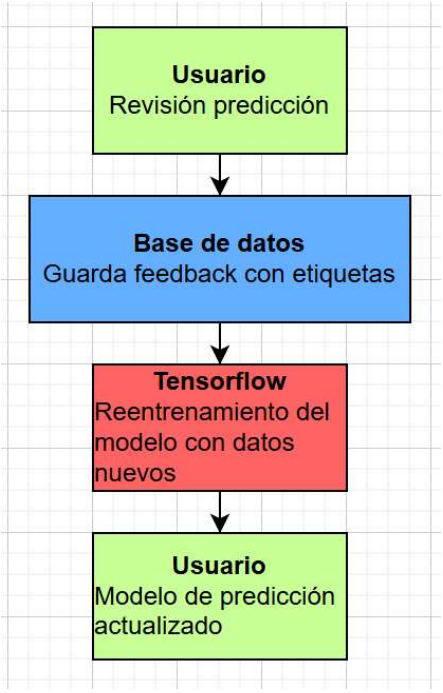
Diagrama de arquitectura



Comunicación Cliente-Servidor



Ciclo de Feedback



Pasos a seguir para conseguir una web completa

El primer paso sería hacer que el modelo sea más generalizado, esto se conseguiría realizando un grán volumen de imágenes a distintas camas con distintos colores de sábanas en distintos entornos con distinta iluminación.

Una vez se haya perfeccionado el modelo de detección del estado de una cama se podría realizar varios modelos más, los cuales serían accesibles desde la página web para la confirmación de la correcta realización de distintas tareas, incluyendo la correcta renovación de papel de baño, renovación de las toallas, recogida del area de cocina, recolocación de los objetos del salón.

Se podría realizar un checklist con las distintas tareas a realizar, estas se confirman trás revisarlas con los modelos de computer vision.

Además de confirmar la correcta realización de las tareas, el modelo podría indicar el grado de precisión del trabajo realizado, pudiendo servir para el propietario de la vivienda como una manera de valoración del trabajo realizado e incluso servir de medida para la creación de un incentivo económico para el servicio de limpieza trás haber realizado un trabajo preciso.

También se podría realizar un apartado dentro de la página web para la comunicación entre los propietarios y el servicio de limpieza.

Bibliografía

Microsoft (s.f.) *Visual Studio Code – Download* [online]. Disponible en: https://code.visualstudio.com/download (Acceso: 14 Diciembre 2025).

Render (s.f.) *Render – Cloud hosting & services* [online]. Disponible en: https://render.com/ (Acceso: 11 Enero 2026).

Google (s.f.) *Google Drive* [online]. Disponible en: https://drive.google.com/ (Acceso: 15 Enero 2026).

diagrams.net (s.f.) *draw.io* [online]. Disponible en: https://www.drawio.com/ (Acceso: 19 Enero 2026).