

Universidad Nacional de Rosario

FACULTAD DE CIENCIAS EXACTAS, INGENIERÍA Y AGRIMENSURA



Licenciatura en ciencias de la computación

BASES DE DATOS AVANZADAS

*Bases de datos espaciales, temporales y
espacio-temporales*

Cavagna, Lucas Gastón

Febrero 2023

Índice general

1	Presentación del objetivo	2
2	Recetario automatizado	3
2.1	Construcción - Obtención de imágenes	3
2.2	Construcción - Data Augmentation	4
2.3	Construcción - Preparación del dataset y red neuronal . . .	7
3	Interacción con el usuario	9
3.1	Recopilador de imágenes	9
3.2	Predicción y cantidades	10
3.3	Recetario	11

1 Presentación del objetivo

El objetivo principal de este trabajo práctico es aplicar los conceptos estudiados en la materia Procesado de Imágenes mediante el desarrollo de un proyecto práctico e interesante. En nuestro caso, proponemos crear un sistema que identifique piezas, estilo lego, a partir de imágenes de manera automática y, en función de la cantidad disponible que coloque el usuario, sugiera recetas predefinidas, por ejemplo un avión. De esta manera, combinamos técnicas de procesamiento de imágenes con un sistema de recomendación, aplicando los conocimientos adquiridos de forma práctica y útil; y obtenemos nuevos durante el transcurso del trabajo práctico.

2 Recetario automatizado

La idea de este informe es ir punto por punto viendo todo lo utilizado para la concepción de nuestro trabajo practico; y de esta manera que podamos explicar con detalle todo lo utilizado en él.

Vamos a dividir de trabajo en dos partes: la *construcción* del mismo; y la *presentación*. Donde en la primera parte esta todo lo necesitado para construir el sistema y en la segunda todo aquello necesario para su presentación para el usuario

2.1 Construcción - Obtención de imágenes

Para poder llevar a cabo nuestro recetario, el primer paso fundamental fue obtener imágenes representativas de las piezas que conformarían las recetas. La calidad y diversidad de estas imágenes son un factor determinante en el desempeño de nuestra red neuronal, ya que un conjunto de datos bien construido permite mejorar la precisión del modelo y su capacidad de generalización ante variaciones en el entorno real.

Con este objetivo en mente, procedimos a la toma de fotografías asegurándonos de capturar la mayor cantidad de variabilidad posible. Para ello, consideramos una serie de factores clave que afectan la apariencia de las piezas y que, por ende, influyen en el desempeño del modelo de reconocimiento. Entre estos factores se encuentran:

- **Tipo de pieza:** Nos aseguramos de fotografiar cada una de las piezas del conjunto en múltiples situaciones y posiciones.
- **Orientación de la cámara:** Variamos los ángulos de captura para garantizar que la red neuronal no dependa de una perspectiva fija.
- **Cantidad y dirección de la luz ambiental:** Se tomaron imágenes bajo diferentes condiciones de iluminación para hacer el modelo más robusto a cambios en el entorno.
- **Texturas del fondo o superficie:** Probamos distintos fondos para evitar que el modelo dependa de un patrón específico y fomentar su capacidad de generalización.

- **Objetos solapados a las piezas:** En algunas imágenes, añadimos elementos que parcialmente cubrían las piezas para analizar el comportamiento del modelo ante oclusiones parciales.

En particular, obtuvimos imágenes de **nueve tipos de piezas distintas**, cada una con una forma y estructura característica. Para facilitar su identificación y procesamiento, asignamos nombres específicos a cada una de ellas:

- *compuerta*
- *cuadrado*
- *cuadradoHueco*
- *parrilla*
- *puente*
- *rectángulo*
- *triánguloEquilátero*
- *ventana*
- *triánguloIsósceles*

Para cada una de estas piezas, se adquirió una gran cantidad de imágenes bajo las condiciones mencionadas anteriormente.

2.2 Construcción - Data Augmentation

Ahora bien, disponemos de una gran cantidad de imágenes, como se mencionó en la sección anterior. Sin embargo, nos enfrentamos a un problema crucial: aunque para un observador humano esta cantidad pueda parecer suficiente, para una red neuronal sigue siendo relativamente pequeña. Las redes neuronales requieren grandes volúmenes de datos para generalizar correctamente y evitar problemas como el sobreajuste. Para abordar esta limitación, aplicaremos el concepto de ***data augmentation***, una técnica ampliamente utilizada en el aprendizaje automático que consiste en aplicar diferentes transformaciones a las imágenes adquiridas con el fin de aumentar la cantidad de datos disponibles.

Mediante el *data augmentation*, buscamos generar variaciones en las imágenes sin alterar su significado esencial, lo que permite que la red neuronal aprenda patrones más robustos y generalizables. Este enfoque es especialmente útil cuando se trabaja con conjuntos de datos relativamente pequeños, ya que ayuda a mejorar la capacidad de la red para reconocer las características esenciales de las imágenes sin depender excesivamente de ejemplos específicos.

Para la construcción de nuestro recetario, hemos decidido aplicar una serie de transformaciones cuidadosamente seleccionadas para maximizar la diversidad de los datos. Las transformaciones elegidas son las siguientes:

- **Rotaciones:** Aplicamos rotaciones en incrementos de 90 grados. Esto permite que la red neuronal aprenda a reconocer los elementos de las imágenes sin depender de su orientación específica.
- **Reflejo o espejado:** Se aplican transformaciones de espejado tanto en el eje vertical como en el horizontal.
- **Adición de ruido:** Para mejorar la robustez del modelo ante variaciones en la calidad de la imagen, aplicamos tres tipos de ruido distintos:
 - **Ruido sal y pimienta:** Introduce píxeles blancos y negros aleatorios en la imagen, emulando interferencias o pérdidas de información.
 - **Ruido gaussiano:** Agrega una distribución normal de ruido a la imagen, lo que simula condiciones de captura menos favorables.
 - **Ruido uniforme:** Genera un ruido distribuido uniformemente sobre la imagen, lo que permite evaluar la respuesta del modelo ante perturbaciones homogéneas.

Todas estas transformaciones se implementan dentro de la función *data_augmentation*, la cual opera en conjunto con diversas funciones accesorias que permiten la manipulación y el preprocesamiento de las imágenes. Estas funciones trabajan en conjunto para aplicar las modificaciones de manera eficiente y ordenada, garantizando que todas las imágenes sean correctamente transformadas y almacenadas.

Una vez implementadas estas transformaciones, es esencial validar su correcto funcionamiento. Para ello, utilizamos la función *cargar_imagenes_directorio*, que se encarga de cargar todas las imágenes de un directorio específico, y la función *graficar_imagenes_dic*, que permite visualizar

de forma estructurada un conjunto de imágenes transformadas. Estas herramientas nos permiten verificar los efectos del *data augmentation* y asegurarnos de que las transformaciones aplicadas cumplen con nuestros objetivos sin distorsionar la información esencial de las imágenes.

En la Figura 1, podemos observar una muestra representativa de los resultados obtenidos tras aplicar la función de aumentación sobre una imagen de prueba. Esta representación gráfica nos proporciona una visión clara de cómo varían las imágenes y nos permite evaluar si la diversidad generada es adecuada para el entrenamiento del modelo.

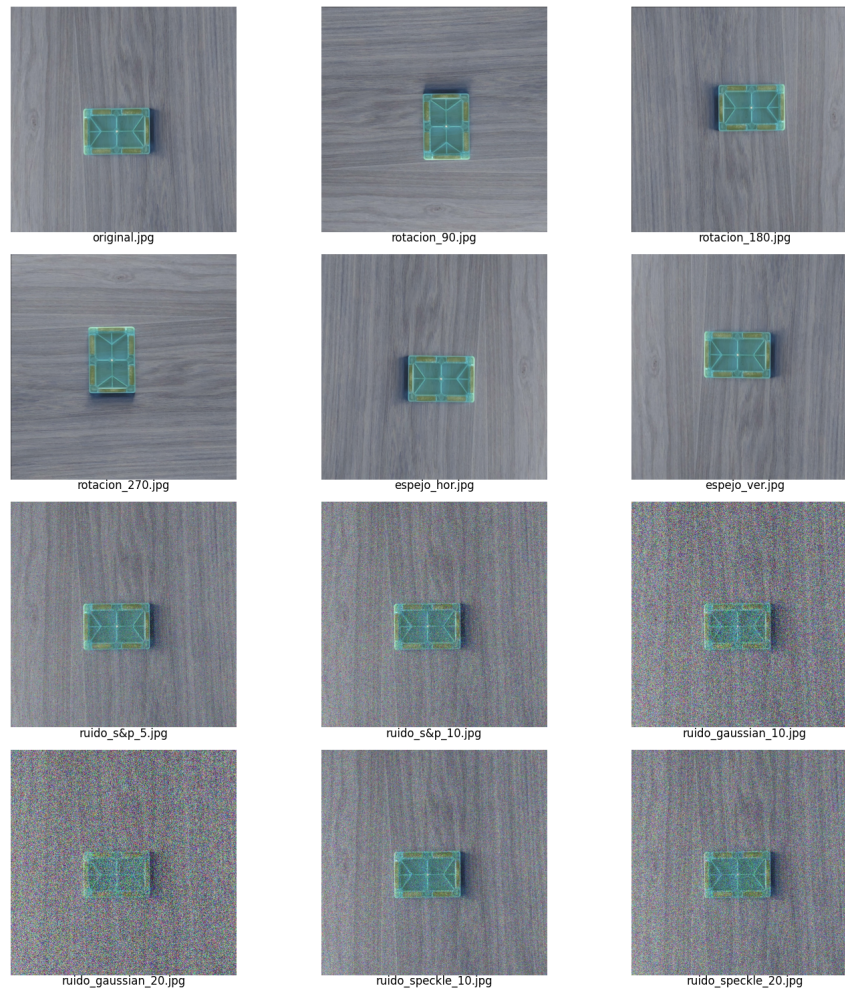


Figura 1: Prueba de la función de *data augmentation*.

Finalmente, para garantizar que todas las categorías de imágenes sean enriquecidas con las nuevas transformaciones, utilizamos las funciones *process_folder* y *data_augmentation_all*. Estas funciones se encargan de aplicar las técnicas de aumentación sobre el conjunto completo de imágenes, asegurando así que el modelo tenga acceso a un conjunto de datos balanceado y diversificado

en términos de perspectivas, iluminaciones y distorsiones.

Con este proceso completado, hemos abordado satisfactoriamente el desafío de aumentar la cantidad de datos disponibles para el entrenamiento de nuestra red neuronal. Ahora, con un *dataset* más amplio y representativo, podemos proceder con los siguientes pasos.

2.3 Construcción - Preparación del dataset y red neuronal

El siguiente paso del cual tenemos que preocuparnos es la preparación del *dataset* para el entrenamiento de la red neuronal. Este es un paso crucial, ya que la calidad y organización de los datos influirán directamente en el desempeño del modelo.

En primera instancia, necesitamos cargar todas las imágenes que hemos conseguido gracias al *data augmentation*. Para ello, utilizamos la función *cargar_data_set*, la cual se encarga de levantar todas las imágenes y asignarles su categoría correspondiente en base a la carpeta en la que se encuentran almacenadas. Este proceso nos permite asegurarnos de que cada imagen está correctamente etiquetada y lista para su posterior procesamiento en la red neuronal.

Una vez cargadas las imágenes, nos enfrentamos a un pequeño inconveniente: los datos se cargan en orden, lo que significa que las imágenes están agrupadas por categoría. Para evitar que el modelo se sesgue por la estructura de los datos, es fundamental randomizar el conjunto de datos, asegurando así una distribución más uniforme de las clases a lo largo del *dataset*. Para lograr esto, utilizamos un **algoritmo de permutación aleatoria** en el segundo bloque de esta sección, el cual reorganiza los datos de manera eficiente.

Sin embargo, aquí nos encontramos con otra limitación técnica importante. Aunque ya tenemos los datos listos para su uso en la red neuronal, necesitamos dividir el *dataset* en 10 partes y guardarlas nuevamente en Google Drive. Esto se debe a que las máquinas proporcionadas por Google Colab tienen capacidades limitadas. Si intentáramos cargar todos los datos de una sola vez y entrenar la red neuronal, nos enfrentaríamos a errores de falta de memoria principal, impidiendo así el correcto entrenamiento del modelo. La estrategia de dividir el *dataset* en partes nos permite entrenar la red neuronal en fragmentos más manejables, evitando problemas de memoria y asegurando un proceso de entrenamiento más eficiente.

Una vez que los datos han sido preparados y almacenados, podemos proceder al diseño de la **estructura de la red neuronal**. En este punto, realizamos múltiples pruebas y ajustes, ya que

encontrar una arquitectura óptima con los recursos limitados que nos proporciona Google Colab resultó ser un desafío significativo. Tras numerosas iteraciones, logramos definir una arquitectura eficiente para la red neuronal. Esta arquitectura está basada en un enfoque de múltiples capas con pocas neuronas en cada una, lo que nos permitió **maximizar** las conexiones y **minimizar** el uso de recursos computacionales. Y en particular usamos como neuronas de salida neuronas con la función de activación *softmax*, la cual nos devuelve un porcentaje de pertenencia de la imagen a una categoría.

Con la arquitectura definida, pasamos al entrenamiento del modelo utilizando la función *entrenar_modelo_eficiente*. Este proceso se realiza de manera iterativa, pasando por cada uno de los subconjuntos del *dataset* para garantizar un aprendizaje progresivo y estable de la red neuronal. En particular, aplicamos esta función dos veces, con el objetivo de reforzar el aprendizaje del modelo y mejorar su capacidad de generalización.

Finalmente, una vez que el entrenamiento ha concluido, procedemos a cargar nuevamente el modelo entrenado y realizamos una prueba para evaluar su capacidad de predicción. No se reservaron imágenes específicamente para el testeo, ya que decidimos aprovechar al máximo todas las imágenes disponibles para el entrenamiento. Sin embargo, se utilizó una pequeña muestra de los datos para realizar pruebas preliminares y verificar la efectividad del modelo.

Este proceso, aunque laborioso, nos permitió entrenar una red neuronal eficiente dentro de las limitaciones de la plataforma y obtener un modelo funcional con buenos resultados.

3 Interacción con el usuario

Finalmente, con el objetivo de poner a prueba el modelo y hacerlo más accesible para usuarios sin conocimientos profundos en el tema, diseñamos una interfaz sencilla y funcional. Esta interfaz se divide en tres secciones, cada una con una función específica. Para acceder a ella solo se necesita ejecutar la totalidad del código de la interfaz y deslizarse hasta abajo de todo donde se despliega la interfaz. En el caso de que se quiera una visión más limpia también se genera un link que de acceso a una página web temporal. Se puede hacer que esta interfaz sea permanente si se utiliza una plataforma adicional pero consideramos que para la envergadura del proyecto no era pertinente. Además, que ya esto se escapa al contenido de la materia.

Como primer paso, definimos variables globales que almacenan enlaces a recursos en GitHub (*NR_URL*) y Google Sheets (*SHEET_URL*), así como estados globales donde se guardan las recetas (*diccionario_datos* y *diccionario_enlaces*) y los elementos de entrada (*state_images* y *state_parts*). Además, establecemos funciones encargadas de definir los valores dentro de estas variables globales, como *diccionario_y_enlaces*, y otras responsables de interactuar con ellas, como *filtrar_diccionario_por_numeros_laxa*.

3.1 Recopilador de imágenes


La primera sección de la interfaz permite al usuario cargar imágenes de las piezas individuales de cada categoría y especificar la cantidad de cada una en los campos correspondientes.

Una vez que los datos han sido correctamente ingresados, el usuario puede avanzar a la siguiente sección.


Ordenar Imágenes por Número

Carga varias imágenes y asigna un número a cada una. Escribe los números separados por comas en el mismo orden que las imágenes cargadas.

Cargar Imágenes


Drop File Here
- or -
Click to Upload

Imágenes



Números asignados (separados por comas)

Ejemplo: 2, 1, 3

Estado de la carga

Clear

Submit

Use via API  · Built with Gradio  · Settings 

3.2 Predicción y cantidades

La segunda sección de la interfaz está dedicada a la predicción de categorías, ofreciendo dos modos de operación: manual y automático.

En el modo manual, el usuario debe asociar cada imagen cargada con la categoría correspondiente, siguiendo el mismo orden en el que fueron ingresadas. Una vez asignadas las categorías, se presiona el botón **Predicción MANUAL** para validar la selección.

En el modo automático, la interfaz realiza la predicción con solo presionar el botón correspondiente, mostrando los resultados en la parte derecha de la pantalla.

Aquí puedes realizar predicciones automáticas o manuales sobre las categorías disponibles. Usa los botones y controles para interactuar con el sistema.

Predicción automática

Predicción manual

Número entero

1

Categorías

compuerta

Procesar MANUAL

Procesar Automático

Mostrar predicciones

Partes disponibles

Resultado

3.3 Recetario

La última sección de la interfaz permite generar recomendaciones de recetas basadas en las piezas disponibles.

Para ello, se ofrecen dos tipos de búsqueda: estricta y laxa. La búsqueda estricta considera únicamente las piezas exactas con las que se cuenta, mientras que la búsqueda laxa permite realizar sustituciones entre piezas de categorías equivalentes, como *compuerta*, *cuadradoHueco*, *parrilla* y *ventana*.

Recopilador de imágenes Predicción y cantidades **Recetario**

Aquí puedes buscar formas para hacer con tus piezas. La búsqueda precisa usa las piezas exactas que tiene y la búsqueda laxa realiza sustituciones por piezas similares.

Tipo de búsqueda

estricta

Buscar en el recetario

Imágenes encontradas

Resultados de la búsqueda