MÁSTER EN INGENIERÍA
DE SISTEMAS Y CONTROL

Quick, Stat!: Un análisis completo de la base de datos Quick, Draw! usando redes recurrentes

Felipe Alcalde Awad

Curso 2023/2024
Defensa: septiembre 2024

Director: Raúl Fernández Fernández

Máster en Ingeniería de Sistemas y control

Quick, Stat!: Un análisis completo de la base de datos Quick, Draw! usando redes recurrentes

Proyecto específico propuesto por profesor

Nombre del estudiante: Felipe Alcalde Awad
Nombre del director: Raúl Fernández Fernández

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado:

Felipe Alcalde Awad

Firma del alumno

Resumen

El *dataset* o conjunto de datos Quick, Draw!, publicado por Google en 2017, contiene más de 50 millones de dibujos vectoriales realizados por usuarios de todo el mundo en el juego en línea del mismo nombre. A diferencia de la mayoría de los conjuntos de datos de imágenes existentes, Quick, Draw! almacena los dibujos como series temporales de posiciones de lápiz en lugar de matrices de bits compuestas por píxeles. Esto lo convierte en el conjunto de datos de dibujos con metadatos de trazos más grande disponible hoy en día. Debido a su tamaño y a la naturaleza de su fuente, existe una escasa información sobre la calidad de los dibujos contenidos. Este trabajo presenta un análisis exhaustivo de este conjunto de datos y la implementación de una red neuronal profunda convolucional recurrente para evaluar su calidad. En el análisis inicial se exploran los diferentes formatos del conjunto de datos, se examina la distribución de dibujos por categoría, la temporalidad de la recolección de datos y la distribución geográfica de los participantes. El análisis principal se enfoca en la extracción y evaluación de tres parámetros: número de trazos, número de segmentos y puntuación de clasificación. Estos parámetros proporcionan información sobre la complejidad de los dibujos en cada categoría y de la calidad de los datos. Para extraer la puntuación de clasificación, se crea y entrena una red neuronal profunda convolucional recurrente. Esta red se utiliza posteriormente para clasificar los dibujos del *dataset* y así otorgar a cada categoría un índice de confianza de las predicciones correctas de la clasificación. Con este análisis, se espera proporcionar información valiosa sobre la calidad y características de los datos contenidos en el *dataset* para apoyar el desarrollo de futuras investigaciones y aplicaciones en el ámbito del aprendizaje automático.

Palabras clave: "*Quick, Draw!*", *dataset*, red neuronal convolucional recurrente, Google, clasificación de imágenes, aprendizaje automático, aprendizaje profundo, análisis de datos, series temporales, visión artificial.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

En noviembre de 2016 Google hizo público un juego interactivo llamado *Quick, Draw!* —presentado unos meses antes en la conferencia Google I/O en Mountain View, California—. En este juego cientos de miles de personas de todo el mundo pudieron demostrar sus habilidades artísticas haciendo dibujos (trazos simples a un solo color) en 20 segundos, mientras una arquitectura neuronal [1] intentaba adivinarlos.

Mediante esta interesante iniciativa, Google pudo almacenar más de 3 TB de datos: mil millones de dibujos formados por vectores y metadatos que podrían ser utilizados más adelante para la investigación. Más tarde, estos datos se pusieron a disposición del público en un *dataset* (conjunto de datos) en varios formatos, que incluye más de 50 millones de dibujos [1]. Aunque Google no ha especificado públicamente los criterios detrás de esta selección, el *dataset* proporciona una muestra significativa de los dibujos originales.

Uno de los primeros trabajos utilizando este *dataset* fue publicado por David Ha y Douglas Eck [2], miembros del equipo de investigadores sobre inteligencia artificial de Google, llamado Google Brain (ahora fusionado con el laboratorio de investigación DeepMind). En este trabajo se presenta una red neuronal recurrente (RNN) llamada *sketch-rnn*, capaz de generar dibujos con secuencias de trazos de objetos comunes (las categorías de *Quick, Draw!)* utilizando un modelo *VAE (*autoencoder variacional*)*. Este trabajo fue pionero debido a que los modelos tradicionales se basaban principalmente en imágenes rasterizadas (mapas de bits), mientras que esta investigación exploró una representación vectorial de menor dimensión inspirada en el trazo humano [2]. Como resultado de este trabajo, el *paper* publicado ha tenido un gran impacto científico recibiendo más de 930 citas en Google Scholar hasta la fecha.

Además de este trabajo, no tardaron en aparecer múltiples proyectos creativos[1] y decenas de trabajos científicos y académicos referenciando *Quick, Draw!* [2]. Esto reafirma la gran utilidad y potencial de este conjunto de datos en un amplio espectro de campos. Sin embargo, existe una escasez de información sobre la calidad de estos dibujos. Esto se debe a dos factores: el tamaño del conjunto de datos y la naturaleza de su fuente.

Esta circunstancia ha motivado a diversos investigadores a publicar estudios que analizan los datos contenidos en el *dataset*. Un ejemplo de ello es *Quick, Stat!* [3]. Este trabajo se centró en el análisis de un porcentaje reducido de los datos, específicamente de tres categorías. Si bien *Quick, Stat!* ofrece una valiosa primera aproximación al análisis de los dibujos de *Quick, Draw!,* el análisis de un mayor volumen de datos y la inclusión de una mayor variedad de categorías ofrecería una información más profunda y completa sobre los datos.

[1] Véase https://github.com/googlecreativelab/quickdraw-dataset?tab=readme-ov-file#projects-using-the-dataset

[2] En febrero de 2024, la frase "Quick, Draw! dataset" muestra 278 resultados en Google Scholar

1.1 Objetivo

El objetivo del presente trabajo es hacer un análisis completo del *dataset* de *Quick, Draw!* para abordar la escasez de información sobre la calidad de sus registros. Se tomará como punto de partida las ideas contenidas en el trabajo de *Quick, Stat!,* y se ampliará su alcance a todas las categorías, utilizando los más de 50 millones de registros que forman el conjunto completo de datos.

Se empezará explorando los diferentes formatos en los que se encuentra disponible el *dataset*. Cada formato ofrece ventajas particulares y en cada etapa del análisis se tendrá que elegir el formato más adecuado.

A continuación, se realizará una evaluación de la distribución de dibujos por categoría, la temporalidad de la recolección de datos y los aspectos geográficos de la muestra. Esto proporcionará una visión holística de cómo se compone el *dataset* y cómo varían los registros en función de diferentes variables.

Se examinarán dos características específicas de los dibujos: la cantidad de trazos y el número total de segmentos por dibujo. Se extraerán la media y la desviación estándar de estos parámetros para cada categoría. Esto permitirá identificar la complejidad de cada categoría, así como patrones y posibles anomalías en los datos.

Para las etapas finales del análisis se creará y entrenará una red neuronal profunda convolucional recurrente. Esta red neuronal se utilizará para clasificar 7 millones y medio de registros (22,000 registros por cada una de las 345 categorías). A través de esta clasificación, se obtendrán tres parámetros adicionales por cada una de las categorías: la media y la desviación estándar del índice de confianza, y el *F1-score*. El índice de confianza mide la certeza del modelo en sus predicciones, mientras que el *F1-score* es el rendimiento del modelo combinando la métrica de precisión y *recall*.

Con esta metodología, se espera proporcionar una evaluación detallada de la calidad de los registros en el *dataset* de *Quick, Draw!* y aportar información valiosa que ayude al desarrollo de futuras investigaciones y de aplicaciones en el ámbito del aprendizaje automático.

2 EL *DATASET QUICK, DRAW!*

Los archivos de datos de *Quick, Draw!*[3], son descritos como un dataset de 50 millones de dibujos divididos en 345 categorías. La mayoría de las categorías se representan con una sola palabra, aunque algunas utilizan frases cortas, como "*The great wall of china*". En cualquiera de los casos, las categorías representan objetos cotidianos o conceptos simples.

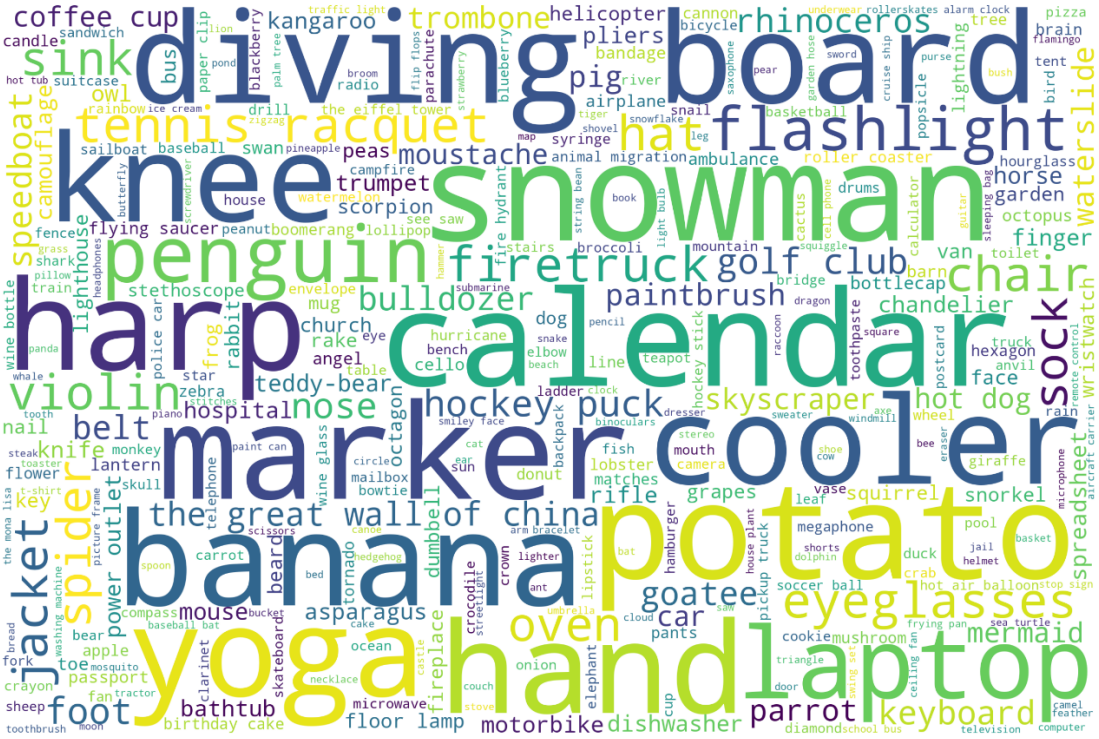


Figura 1. Nube de palabras con las 345 categorías de Quick, Draw!. El tamaño de la fuente indica la frecuencia de cada categoría en el conjunto de datos.

Google advierte que los dibujos han sido individualmente moderados pero que pueden contener aun así "contenido inapropiado". No se describen los métodos que se han utilizado para la selección y el filtrado de estos dibujos. Tampoco se da información de la cantidad de registros (dibujos) por categoría, por lo que este será uno de los primeros datos que se obtendrá para el análisis.

2.1 Formatos del *dataset*

El *dataset* de *Quick, Draw!* está disponible para descarga a través de la plataforma Google Cloud[4] en 4 formatos diferentes (ver Figura 2). Cada una de ellas está dividida en ficheros individuales para cada categoría.

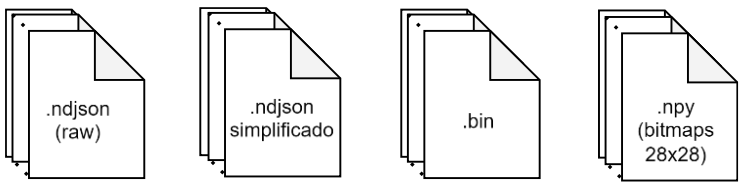


Figura 2. Formatos disponibles del *dataset* Quick, Draw!

[3] Véase https://github.com/googlecreativelab/quickdraw-dataset

[4] Véase https://console.cloud.google.com/storage/browser/quickdraw_dataset

2.1.1 Ndjson *raw*

Consiste en ficheros, uno por categoría, con los dibujos sin modificaciones en formato ndjson[5]. Cada línea equivale a un registro y contiene los datos mostrados en la Tabla 1.

Tabla 1. Formato ndjson raw del *dataset*.

Nombre del campo	Formato	Descripción
key_id	Entero	Número identificativo
word	*String*	Palabra que se ha dibujado (categoría)
recognized	Booleano	Si la palabra fue adivinada por el juego
timestamp	*Datetime*	Fecha y hora de creación del dibujo
countrycode	*String*	Código de dos caracteres del país
drawing	*JSON Array*	Los trazos del dibujo con datos de tiempo [[// primer trazo [x0, x1, x2, x3, ...], // coordenadas x [y0, y1, y2, y3, ...], // coordenadas y [t0, t1, t2, t3, ...] // tiempo [ms] transcurridos], [// segundo trazo [x0, x1, x2, x3, ...], [y0, y1, y2, y3, ...], [t0, t1, t2, t3, ...]], ... // trazos adicionales]

Se advierte de la gran diferencia en cuanto al tamaño de los lienzos de los dibujos. Estas diferencias dependen del tamaño de pantalla del dispositivo que utilizó cada participante.

Los ficheros de cada categoría pesan entre 115 MB (categoría *line*) y 1,6 GB (categoría *calendar*). El peso total de la versión Ndjson raw del *dataset* es de 195 GB.

2.1.2 Ndjson simplificado

Tiene el mismo formato que la versión ndjson *raw,* pero omitiendo los datos de temporalidad en los trazos, además, el dibujo ha sido escalado y simplificado utilizando el siguiente proceso[6] :

- Alinear el dibujo a la esquina superior izquierda, para tener valores mínimos de 0.
- Escalar uniformemente el dibujo, para tener un valor máximo de 255.
- "Remuestrear" todos los trazos con una separación de 1 píxel.
- Simplificar todos los trazos utilizando el algoritmo Ramer-Douglas-Peucker con un valor épsilon de 2,0.

Tabla 2. Formato ndjson simplificado del *dataset*.

Nombre del campo	Formato	Descripción
key_id	Entero	Número identificativo
word	*String*	Palabra que se ha dibujado (categoría)
recognized	Booleano	Si la palabra fue adivinada por el juego

[5] Véase https://github.com/ndjson/ndjson-spec

[6] Véase https://github.com/googlecreativelab/quickdraw-dataset?tab=readme-ov-file#simplified-drawing-files-ndjson

timestamp	*Datetime*	Fecha y hora de creación del dibujo
countrycode	*String*	Código de dos caracteres del país
drawing	*JSON Array*	Los trazos del dibujo [[// primer trazo [x0, x1, x2, x3, ...], // coordenadas x [y0, y1, y2, y3, ...], // coordenadas y], [// segundo trazo [x0, x1, x2, x3, ...], [y0, y1, y2, y3, ...],], ... // trazos adicionales]

Los ficheros de cada categoría en el *dataset* simplificado pesan entre 25 MB (categoría *line*) y 180 MB (categoría *calendar*). El peso total de esta versión del *dataset* es de 22,3 GB (una reducción casi el 89% respecto a la versión *raw*).

2.1.3 Binario (.bin)

Consiste en los mismos trazos de la versión simplificada, pero en un formato binario con el cual se logra reducir aún más el tamaño de los ficheros. Los ficheros binarios de cada categoría pesan entre 4 MB (categoría *line*) y 45 MB (categoría *calendar*). El peso total de esta versión del *dataset* es de 6 GB.

Aunque Google no ofrece documentación sobre la estructura de los datos en los ficheros binarios, sí proporciona ejemplos de scripts en Python y NodeJS para poder interpretarlos[7]. Analizando estos scripts, se ha podido inferir la estructura de los datos binarios, la cual se muestra en la Figura 3. Los trazos quedan guardados en el campo de longitud variable *drawing*. Para cada trazo se almacena primero el número de puntos que lo componen (n_points), seguido de las coordenadas X (X1, X2, ..., Xn_points) y las coordenadas Y (Y1, Y2, ..., Yn_points) de cada uno de los puntos.

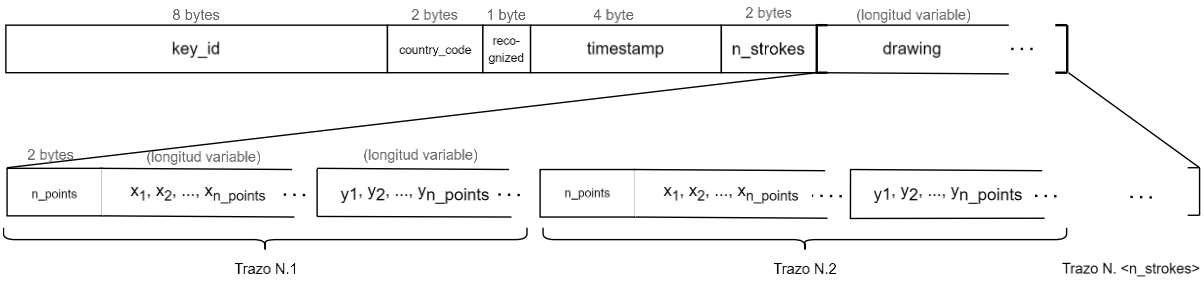


Figura 3. Estructura de datos binaria de un registro de dibujo en el *dataset*, detallando los campos informativos y la estructura del campo drawing que contiene los trazos.

[7] Véase https://github.com/googlecreativelab/quickdraw-dataset/blob/master/README.md#binary-files-bin

2.1.4 Numpy (.npy)

Se trata de una versión con representaciones rasterizadas (mapa de bits) guardado como un *array* unidimensional numpy[8] de 784 elementos, representando los pixeles del dibujo con un tamaño de 28x28.

En esta versión del conjunto de datos se muestran solo las versiones finales de los dibujos y se pierde toda la información de temporalidad de los trazos. También se advierte que a diferencia de la versión ndjson, los dibujos han sido centrados en lugar de estar alineados a la esquina superior izquierda.

Los ficheros ".npy" de cada categoría pesan entre 85 MB (categoría 209, *panda*) y 254 MB (categoría 273, *snowman*), lo cual coincide con el número de registros en esas categorías (son las categorías con menor y mayor número de registros respectivamente). El peso total de esta versión del *dataset* es de 40 GB.

La Figura 4 muestra las imágenes del primer registro de las primeras 15 categorías (consultar Anexo 2 para ver una versión con el total de las categorías). Es importante mencionar que se ha tenido que aplicar un operador inverso (negativo) [4] a las imágenes para mostrarlas como trazos negros sobre un lienzo blanco, ya que los pixeles blancos están guardados con valor 0 (que suele representar el color negro en una escala de grises). También se han ampliado un poco las imágenes ya que en su tamaño original (28x28) no se llegan a apreciar bien.

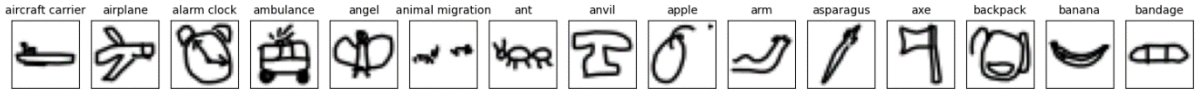


Figura 4. Primer dibujo de las primeras 15 categorías del dataset en formato numpy.

2.1.5 Comparación de un registro en los diferentes formatos

En las figuras 5, 6 y 7 se muestra el segundo registro de la categoría 2 (*airplane)* utilizando los datos de las diferentes versiones del *dataset*. Para el caso del *dataset* ndjson raw, se ha representado el dibujo utilizado un lienzo de tamaño X_{min} - X_{max} x Y_{min} - Y_{max}, con origen en (X_{min}, Y_{min}). Donde X e Y son las coordenadas de los trazos. De esta forma el dibujo está alineado a la esquina superior izquierda del lienzo, cosa que ya está hecha en el *dataset* versión ndjson simplificado.

Lo primero que se puede observar es la pérdida de detalle. En el *dataset* simplificado se pierden los trazos curvos y en el caso concreto mostrado en las figuras, las ventanas del avión acaban siendo simples líneas. La versión numpy, con dibujos rasterizados, facilita la visualización pero puede perder detalles debido a su formato pequeño. Esto dificulta la identificación humana de los dibujos, como se muestra en la Figura 7. Aunque los modelos de inteligencia artificial pueden ser menos sensibles al ojo humano a esta reducción de tamaño en los dibujos, se ha demostrado que la baja resolución afecta negativamente a los sistemas CNN de clasificación de imágenes [5].

[8] Véase https://numpy.org/devdocs/reference/generated/numpy.lib.format.html

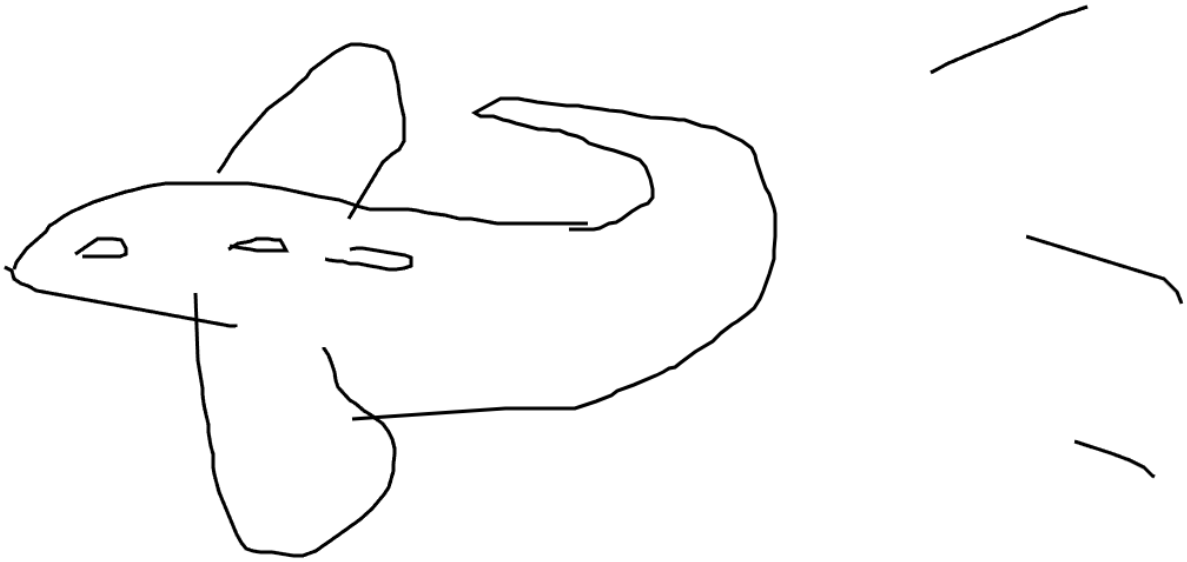


Figura 5. Muestra de un dibujo de la categoría 2 (*airplane)* en formato ndjson raw.

Figura 6. Muestra de un dibujo de la categoría 2 (*airplane)* en formato ndjson simplificado.

Figura 7. Muestra de un dibujo de la categoría 2 (*airplane)* en formato numpy.

2.2 Análisis inicial del *dataset*

Para poder realizar el análisis inicial de los datos, se descargaron todos los ficheros del *dataset* —en sus 4 formatos— a una cuenta de Google Drive (servicio de almacenamiento en la nube). Esta cuenta posteriormente se conectó a un entorno de programación en Python desde el navegador, llamado Google Colaboratory[9]. Finalmente, se crearon diferentes scripts para recorrer el total de los registros en las 345 categorías y así obtener una variedad de estadísticas sobre sus metadatos. A menos que se especifique lo contrario, se utilizaron el total de los registros del *dataset* para obtener las estadísticas presentadas.

2.2.1 Distribución de registros por categoría

Entre todas las categorías hay un total de 50.426.266 registros. No todas las categorías tienen el mismo número de registros, como se puede ver en el histograma mostrado en la Figura 8, si no que van desde 113.613 (en la categoría 209, *panda*), a 340.029 (en la categoría 273, *snowman*). Esta desigualdad entre categorías se puede medir con el ratio de desequilibrio o desbalanceo. Este parámetro es definido como la razón entre el número de muestras de la clase mayoritaria y el de la clase minoritaria [6]. Un conjunto de datos totalmente balanceado tendría un ratio de desbalanceo con valor de 1. Para esta base de datos, el ratio de desbalanceo es de 3, al tener la clase mayoritaria casi el triple de muestras que la minoritaria. Ratios mayores a 1 pueden afectar el rendimiento de los modelos de aprendizaje automático, ya que pueden mostrar una preferencia hacia las clases más frecuentes generando así resultados sesgados. Una posible solución a este problema es utilizar, para cada clase, un número de registros que no superen el de la clase minoritaria, en este caso 113.613.

La Figura 9 muestra la distribución del número de registros por categoría. Podemos ver que, aunque hay categorías con un número elevado de registros (más de 300.000), la mayoría de las categorías contienen entre 110.000 y 150.000 registros.

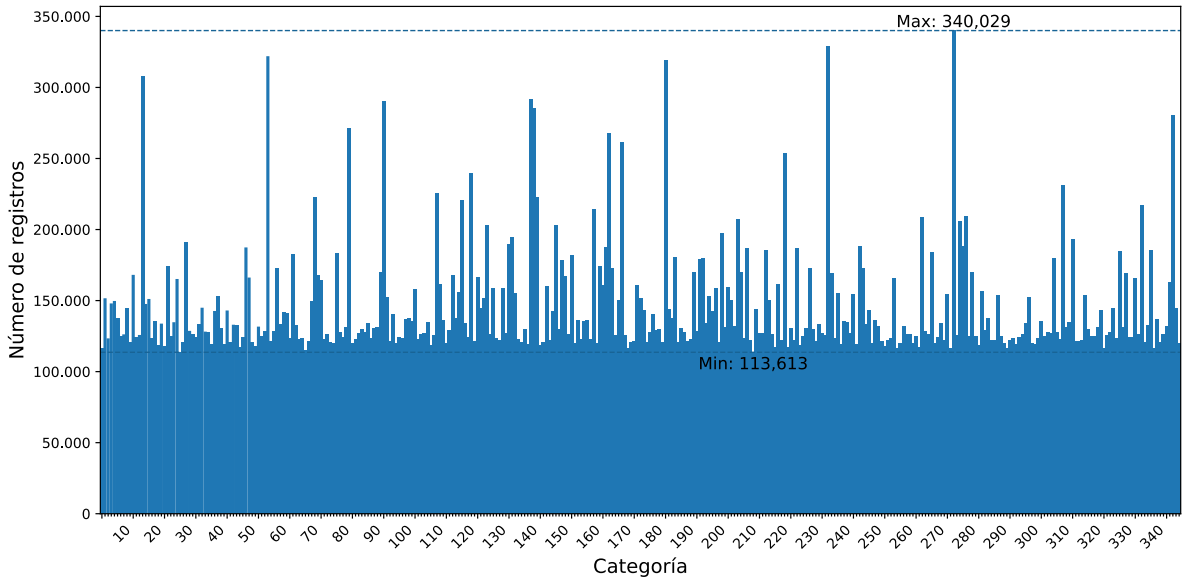


Figura 8. Histograma del número de registros por categoría.

[9] Véase https://colab.research.google.com/

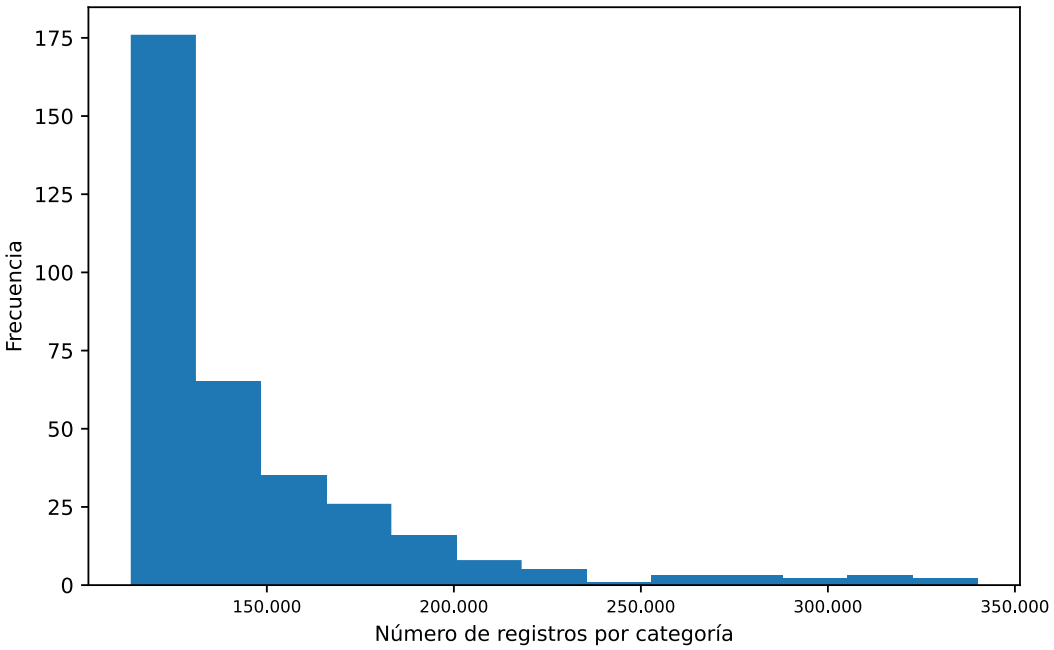


Figura 9. Histograma de frecuencias de número de categorías en función del número de registros.

La Tabla 3 muestra las 10 categorías con mayor y con menor número de registros (lista *Top 10* y *Bottom 10*). Ahora bien, en las tablas del Anexo 1 se pueden consultar las 345 categorías con su nombre original en orden alfabético[10], su traducción en castellano y el número de registros de cada categoría.

Tabla 3. Las 10 categorías con más y con menos registros.

10 categorías con más registros	
Categoría	**N. dibujos**
N. 237 (snowman)	340.029
N. 233 (potato)	329.204
N. 54 (calendar)	321.981
N. 181 (marker)	319.136
N. 14 (banana)	307.936
N. 138 (hand)	291.773
N. 91 (diving board)	290.239
N. 139 (harp)	285.403
N. 343 (yoga)	280.442
N. 80 (cooler)	271.444

10 categorías con menos registros	
Categoría	**N. dibujos**
N. 209 (panda)	113.613
N. 26 (bed)	113.862
N. 66 (ceiling fan)	115.413
N. 255 (screwdriver)	116.313
N. 337 (whale)	116.502
N. 1 (aircraft carrier)	116.504
N. 290 (stove)	116.535
N. 321 (tractor)	116.677
N. 272 (snowflake)	116.685
N. 169 (leg)	116.804

[10] Se han convertido todos los nombres de los ficheros a minúsculas antes del indexado.

2.2.2 Evaluación temporal de la recolección de datos

Analizando el campo *timestamp* en la versión ndjson del conjunto de datos, se puede observar que:

- La fecha de inicio de recolección datos es: 23 de enero de 2017
- La fecha final de recolección de datos es: 29 de marzo de 2017

Los datos se recolectaron durante 37 días efectivos, distribuidos en un periodo de 65 días con 28 días de inactividad entre medias. En la Figura 10 se muestra la cantidad de dibujos (en millones) almacenados por día. Estos datos muestran que el *dataset* público está contenido en un marco temporal muy concreto y que, aunque el juego de *Quick, Draw!* siga en línea 7 años después de su publicación[11], Google no ha ampliado ni actualizado el conjunto de datos.

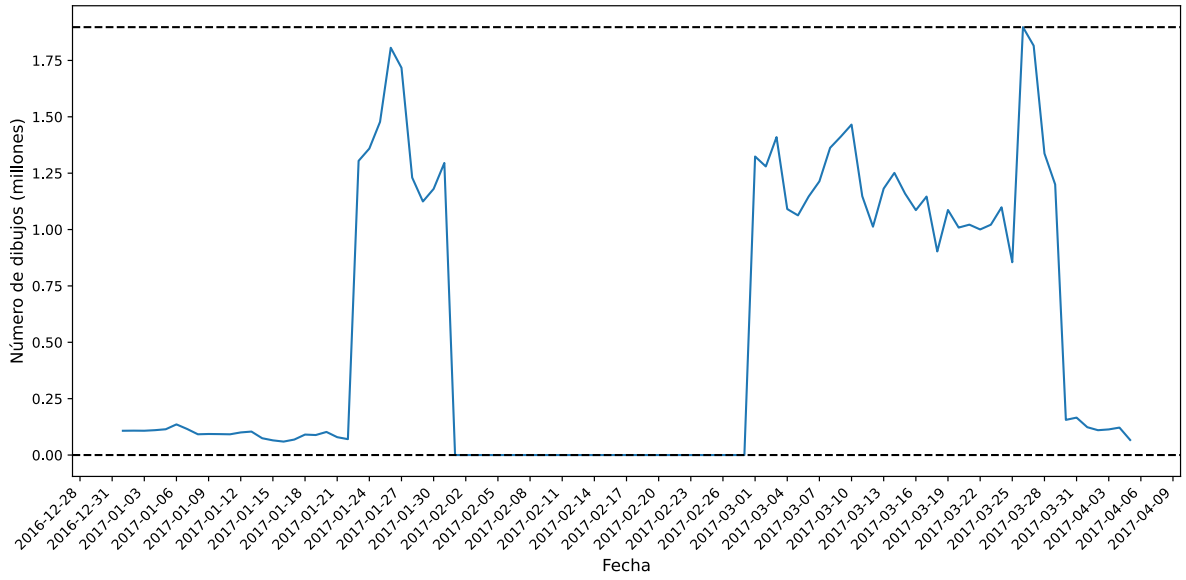


Figura 10. Número de registros integrados al *dataset* por día, en millones.

Después de haber analizado la distribución temporal de los datos, se realizó un estudio de la organización del conjunto de datos para conocer el criterio que se utilizó para guardar los registros. El objetivo era saber si los registros tenían un orden aleatorio o un orden cronológico. Mediante el análisis del *timestamp* de los registros se llegó a la conclusión de que los registros están mezclados.

[11] Véase https://quickdraw.withgoogle.com/

2.2.3 Aspectos geográficos de la muestra

El siguiente parámetro que se analizó es la distribución geográfica de los dibujos. Para ello, se utilizó el código de país presente en el *dataset*, el cual sigue el estándar ISO 3166-1 alpha-2[12] para representar los países.

De los 50,4 millones de registros, casi la mitad de ellos resultaron ser de EE. UU. El resto está dividido entre otros 216 países. Como se puede ver en la Figura 11, el conjunto de datos tiene un sesgo importante hacia los dibujos realizados en EE. UU. También podemos observar en la Tabla 4 que en el otro extremo de la lista hay 73 países que tienen menos de mil dibujos —incluso llegando a ver algunos con tan solo 1— y por lo tanto no son nada representativos en el conjunto de datos.

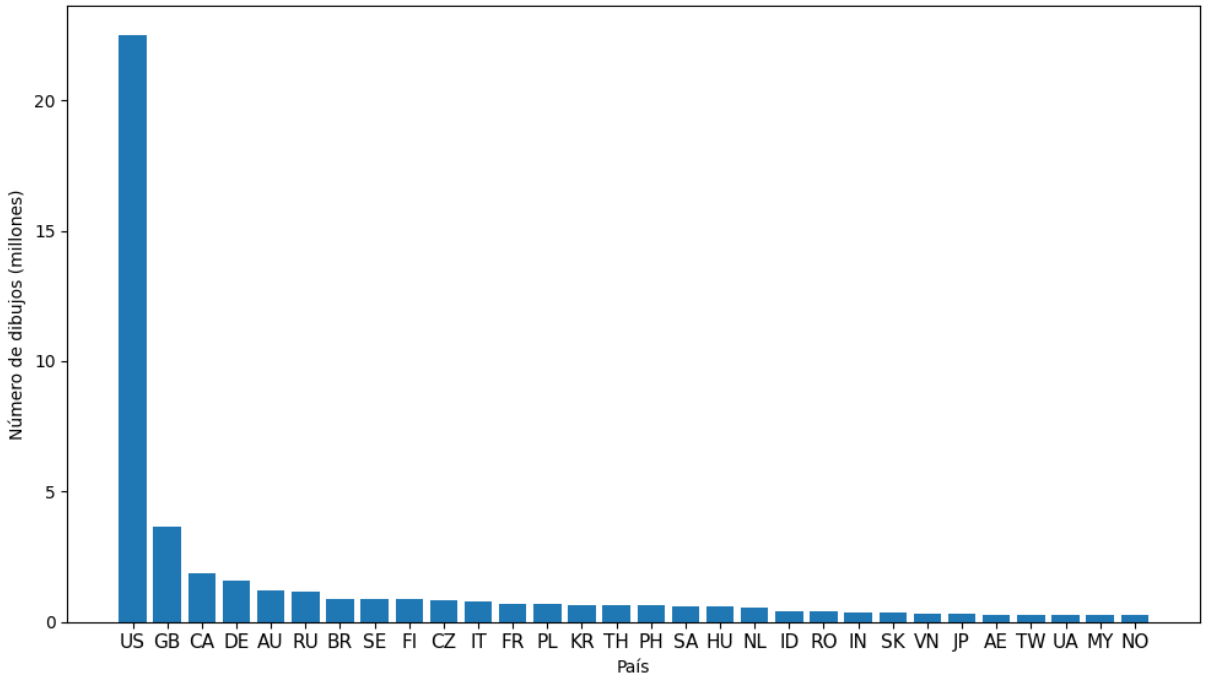


Figura 11. Cantidad de registros por país de origen (Top 30).

En caso de que la procedencia geográfica de los dibujos sea importante para el uso que se le quiera dar a este conjunto de datos, hay que tener presente que los datos no están equilibrados y que, para poder equilibrarlos, el número de registros disponibles puede ser menor de lo esperado. Si se toman como base los registros de EE. UU., se tienen más de 22 millones de registros, pero si se toman los 6 países con más dibujos, el número de registros por país tendría que limitarse a un millón para tener la misma cantidad de registros por cada país. Este número de registros (Tabla 4) va bajando considerablemente entre más países se quieran incorporar:

- 50 países.- 96.000 registros por país.
- 100 países.- 7.200 registros por país.
- 200 países.- 40 registros por país.

[12] El estándar ISO 3166-1 alpha-2 es un sistema de códigos de dos letras reconocido internacionalmente para representar países y territorios. La lista completa de códigos puede consultarse en el sitio web de la Organización Internacional de Normalización.

Tabla 4. Cantidad de registros por país. En negro, países que no tienen registros de todas las categorías.

0) US: 22.513.394	44) BE: 152.316	88) TT: 13.434	132) NG: 1.999	176) AO: 294
1) GB: 3.650.783	45) IL: 128.612	89) LU: 13.180	133) NA: 1.903	177) AS: 286
2) CA: 1.844.184	46) AR: 118.763	90) AL: 12.752	134) AW: 1.895	178) BT: 270
3) DE: 1.574.723	47) EE: 111.872	91) UY: 12.332	135) BZ: 1.644	179) TJ: 268
4) AU: 1.190.919	48) ZA: 103.743	92) AM: 12.274	136) MQ: 1.572	180) RW: 243
5) RU: 1.175.012	49) BA: 96.850	93) GU: 11.617	137) GY: 1.332	181) HT: 237
6) BR: 894.262	50) IQ: 86.837	94) DO: 10.185	138) SM: 1.264	182) AI: 224
7) SE: 868.312	51) LT: 86.333	95) AZ: 9.410	139) SO: 1.219	183) LS: 220
8) FI: 858.441	52) GR: 79.437	96) MO: 9.201	140) VI: 1.201	184) MH: 188
9) CZ: 812.211	53) KZ: 79.346	97) RE: 8.917	141) SX: 1.186	185) TM: 176
10) IT: 788.031	54) EG: 78.212	98) GT: 7.835	142) ZW: 1.104	186) WS: 154
11) FR: 694.096	55) LV: 75.886	99) HN: 7.565	143) DM: 1.075	187) ET: 152
12) PL: 684.238	56) KW: 74.879	100) PA: 7.265	144) PF: 1.042	188) DJ: 150
13) KR: 660.061	57) DZ: 74.013	101) KG: 7.210	145) FJ: 945	189) MW: 150
14) TH: 650.632	58) CL: 73.945	102) MV: 7.128	146) GF: 912	190) MR: 147
15) PH: 637.180	59) BY: 69.899	103) KE: 6.948	147) GD: 871	191) GA: 139
16) SA: 615.546	60) CO: 67.033	104) MN: 6.490	148) SN: 788	192) CV: 136
17) HU: 589.661	61) QA: 61.409	105) SV: 5.942	149) TZ: 775	193) VG: 103
18) NL: 555.091	62) KH: 59.000	106) BO: 5.919	150) MG: 772	194) PG: 88
19) ID: 428.246	63) SI: 58.580	107) MU: 5.716	151) GH: 747	195) PM: 62
20) RO: 389.510	64) JO: 50.044	108) JM: 5.303	152) CI: 742	196) LR: 53
21) IN: 373.585	65) PK: 49.283	109) CN: 5.233	153) LI: 693	197) ML: 47
22) SK: 358.253	66) ZZ: 40.740	110) UZ: 4.984	154) UG: 625	198) CD: 46
23) VN: 339.203	67) PE: 39.591	111) JE: 4.877	155) MS: 568	199) BF: 41
24) JP: 331.631	68) MA: 39.475	112) PY: 4.708	156) YE: 562	200) SZ: 36
25) AE: 291.526	69) BH: 39.451	113) LK: 4.595	157) AG: 528	201) BJ: 27
26) TW: 289.795	70) GE: 36.331	114) GG: 4.108	158) BW: 523	202) CX: 27
27) UA: 288.263	71) MK: 36.076	115) BS: 4.096	159) FM: 522	203) CF: 23
28) MY: 258.479	72) MD: 29.537	116) AX: 4.074	160) AF: 476	204) TG: 23
29) NO: 258.145	73) OM: 25.097	117) IM: 3.831	161) KN: 475	205) ST: 21
30) HR: 240.846	74) IS: 24.591	118) MP: 3.695	162) BU: 450	206) PW: 19
31) IE: 240.688	75) BD: 24.520	119) NC: 3.385	163) AN: 448	207) TO: 14
32) NZ: 240.627	76) PR: 24.164	120) BM: 3.296	164) MZ: 441	208) VU: 11
33) TR: 235.600	77) PS: 22.021	121) KY: 3.180	165) VC: 436	209) CG: 7
34) RS: 216.069	78) VE: 21.358	122) CW: 2.981	166) ZM: 420	210) TL: 6
35) BG: 213.811	79) EC: 20.131	123) SR: 2.830	167) MF: 398	211) GN: 6
36) HK: 200.548	80) LB: 19.638	124) MM: 2.725	168) AD: 391	212) GM: 2
37) AT: 198.016	81) TN: 18.941	125) LY: 2.482	169) MC: 389	213) IR: 2
38) DK: 190.343	82) NP: 17.462	126) LA: 2.454	170) TC: 385	214) BI: 2
39) MX: 187.205	83) MT: 17.263	127) BB: 2.431	171) LC: 382	215) SJ: 1
40) ES: 184.208	84) CY: 14.515	128) NI: 2.384	172) CM: 350	216) NF: 1
41) CH: 181.524	85) BN: 14.345	129) GI: 2.329	173) YT: 348	217) SS: 1
42) PT: 180.839	86) ME: 13.769	130) GP: 2.232	174) SC: 339	
43) SG: 158.370	87) CR: 13.639	131) FO: 2.130	175) GL: 309	

El análisis reveló una distribución desigual de muestras por país y categoría. Algunos países carecen de registros para ciertas categorías, ya sea por no tener un número de muestras suficientes, o por tener una distribución no uniforme entre ellas. La Tabla 4 muestra en color negro los países sin representación en todas las categorías.

2.3 Análisis del número de trazos y de los segmentos

El análisis del número de trazos y segmentos extiende el análisis presentado en *Quick, Stat!* [3]. A diferencia del trabajo de referencia, que se limitó a analizar únicamente tres categorías y una muestra de registros, en este trabajo se ha optado por abarcar la totalidad de las 345 categorías disponibles y el total de registros del *dataset*.

Con el objetivo de evitar tablas difíciles de interpretar o figuras que ocupen varias páginas, se han creado gráficos concisos que ofrecen una perspectiva global de los datos. Estos gráficos se complementan con tablas o gráficos más detallados disponibles en los anexos. La tabla del Anexo 1 recoge los resultados principales detallados por categoría y define el número asignado a cada categoría, nomenclatura que se utilizará a lo largo de toda la memoria.

2.3.1 Cantidad de trazos por registro

El número de trazos de un dibujo se define como la cantidad de veces que el "lápiz" toca el lienzo para dibujar sin levantarlo. El número de trazos puede ser un indicador del número de diferentes elementos que lo forman y, por lo tanto, de su complejidad.

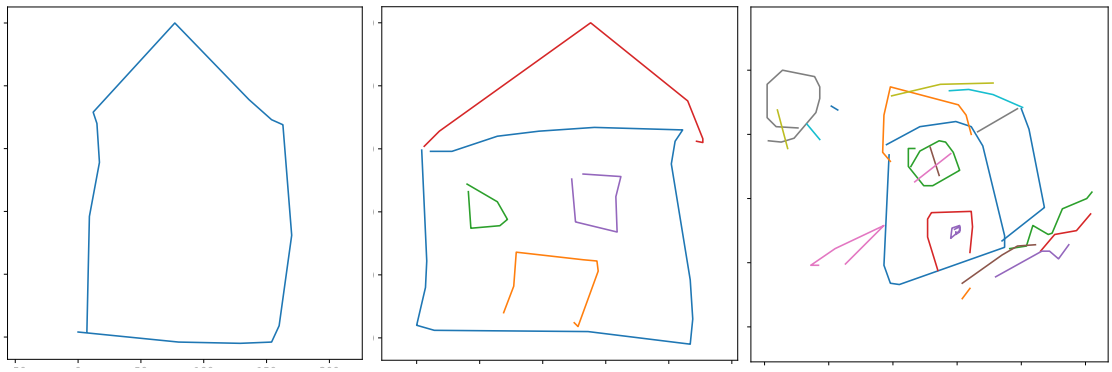


Figura 12. Muestras de tres dibujos de la categoría 155 con 1, 5 y 25 trazos respectivamente. Cada trazo se muestra en un color diferente. El incremento en el número de trazos agrega más elementos al dibujo y, por lo tanto, más complejidad.

Como se puede observar en la Figura 13, la media del número de trazos de la mayoría de las categorías está comprendida entre 4 y 8, pero todas las categorías tienen por lo menos algún dibujo realizado con una gran cantidad de trazos (número de trazos superior a 100). Algunas de ellas llegando a tener dibujos con más de 500 trazos —1847 en el caso extremo, en la categoría 217 (*peas)*—. Estos son casos claros de *outliers* en el conjunto de datos. También podemos observar una desviación estándar pequeña (entre 3 y 6) para casi todas las categorías. Esto muestra que, aunque todas las categorías tienen ejemplares con un número de trazos que es, como mínimo, diez veces mayor que la media, el número de trazos no es una variable muy dispersa.

La Figura 14 ilustra de una manera más clara la distribución de trazos de los registros de las 345 categorías con un gráfico de distribución superpuesta. Con el objetivo de establecer una comparación equitativa entre las categorías, se utilizó una muestra de 113.613 elementos de cada una. Esta cantidad corresponde al total de elementos en la categoría con menor número de registros y al 33% de la categoría con mayor número de registros. Para optimizar la visualización de los datos, el eje x se ha truncado en 25 trazos debido a la escasa representación de dibujos que superan esta cantidad.

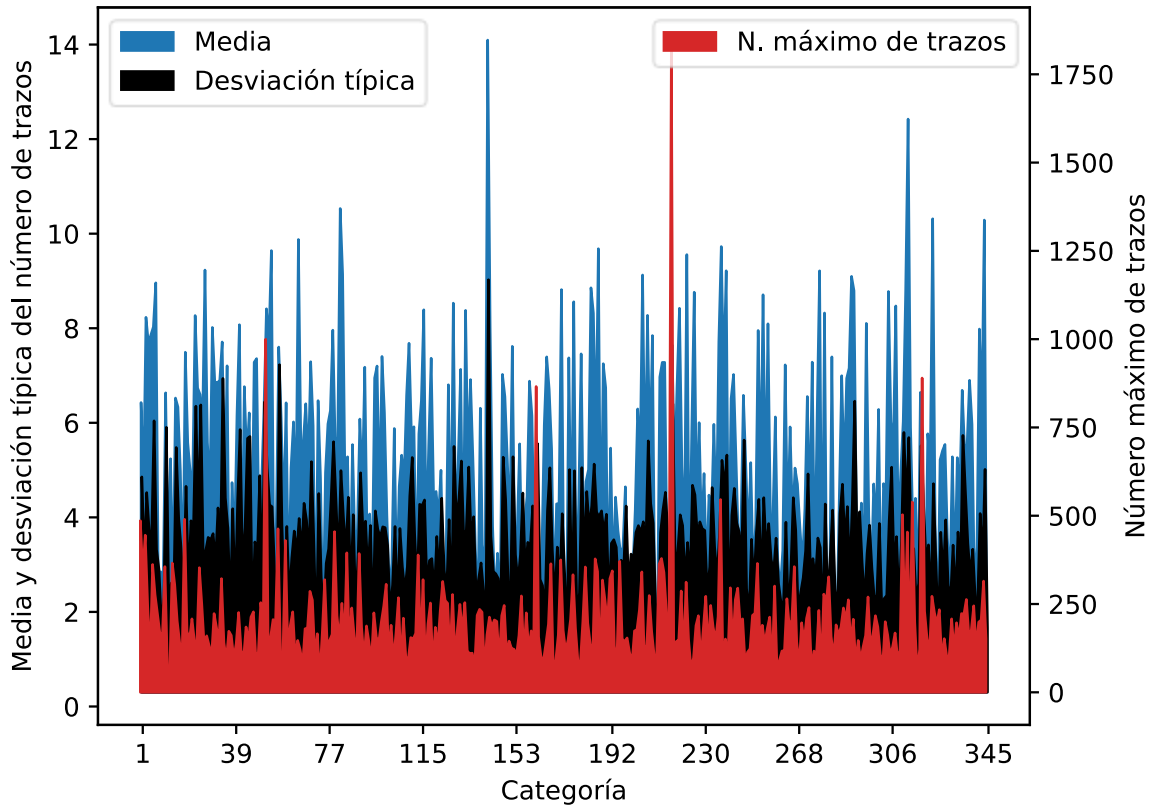


Figura 13. Media, desviación estándar y máximo número de trazos en cada categoría *Quick, Draw!*

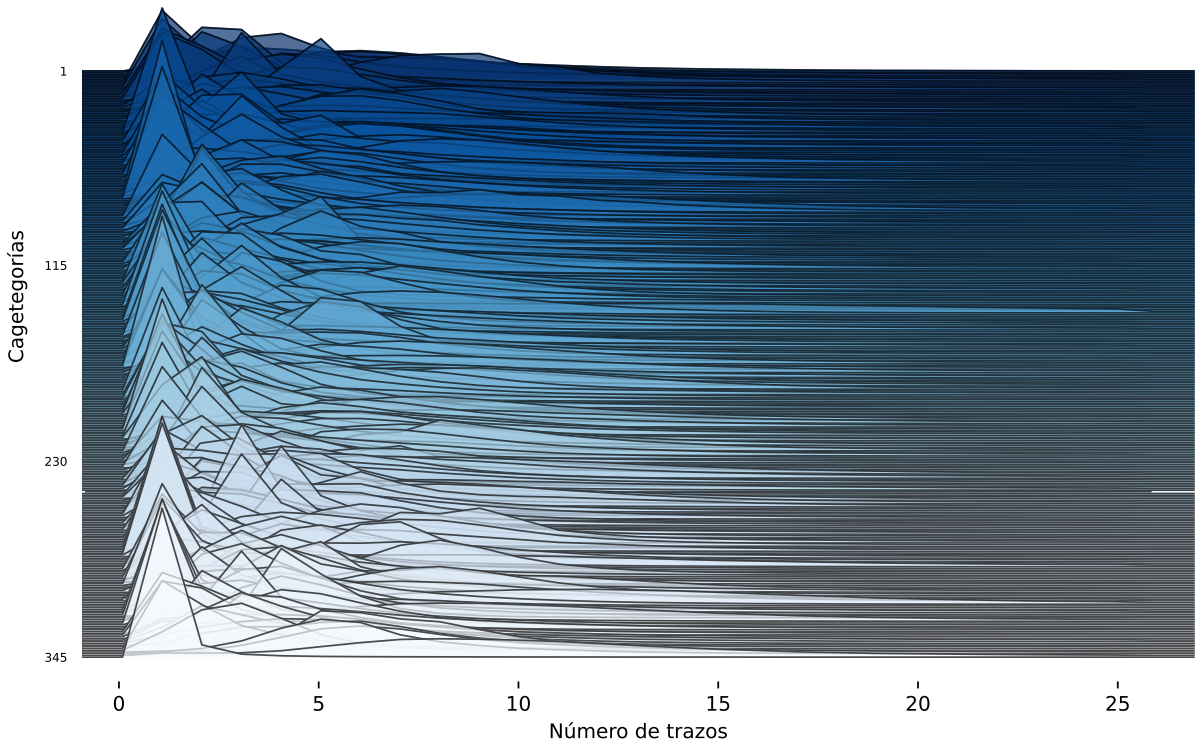


Figura 14. Distribución del número de trazos de las 345 categorías de *Quick, Draw!*. *Eje x truncado en 25 trazos.*

En el Anexo 3 se puede ver con más detalle la distribución de trazos de cada una de las categorías. Cabe destacar las categorías 142 (*hedgehog*), 313 (*tiger*), y 82 (*cow*) que presentan la media más alta

en el número de trazos, situándose entre 10 y 14. Por otro lado, las categorías 72 (*circle*), 174 (*line*), 345 (*zigzag*) tienen la media más baja en el número de trazos, rondando 1. Los resultados detallados de la media y la desviación estándar de trazos de cada categoría han sido incorporados a la tabla del Anexo 1.

2.3.2 Número total de segmentos por registro

El número total de segmentos corresponde a la cantidad de líneas rectas (vectores) que forman todos los trazos del dibujo. Este parámetro lo podemos considerar como la *longitud* del dibujo. Aunque es posible realizar un dibujo largo con un solo trazo (sin haber levantado nunca el "lápiz"), en un lienzo de tamaño limitado se necesita cambiar de dirección para seguir dibujando, y cada cambio de dirección crearía un segmento nuevo. El número total de segmentos se puede calcular con la siguiente fórmula.

$$N_{segmentos} = \sum_{k=1}^{N_{trazos}} N_{vertices\_trazo_k} - 1 \quad (1)$$

Es decir, por cada trazo, se suman todos menos el último punto que lo define.

El número de segmentos también está relacionado con la complejidad del dibujo, ya que, si este contiene líneas curvas y/o muchos cambios de dirección en los trazos, se generarían un mayor número de segmentos.

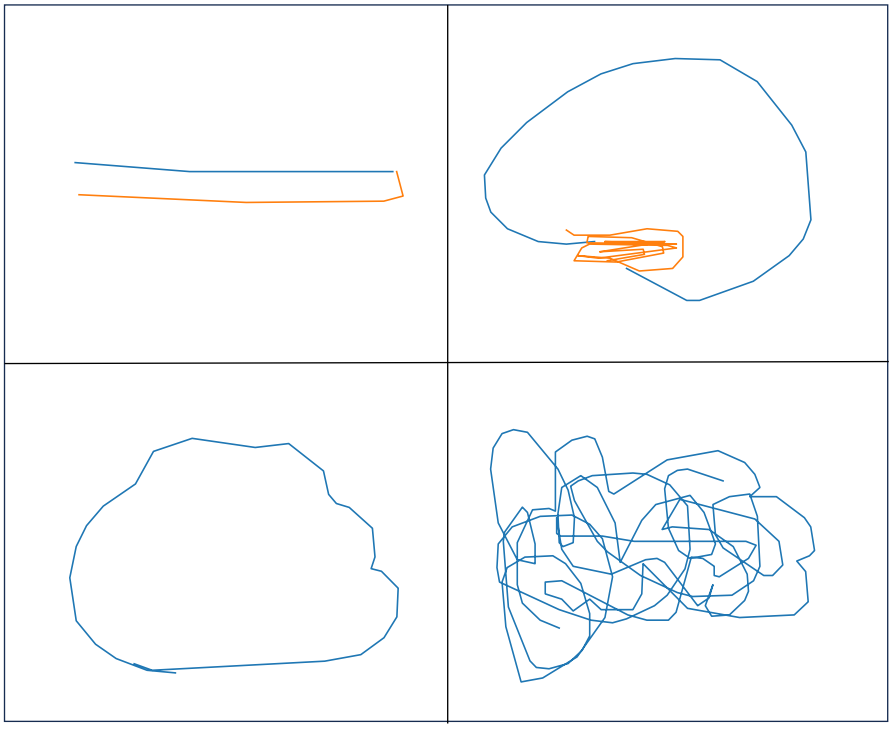


Figura 15. Ejemplos de dibujos con la misma cantidad de trazos pero diferentes números de segmentos. Arriba: ejemplos de la categoría 28 (belt); a la izquierda, 2 trazos y 6 segmentos; a la derecha, 2 trazos y 52 segmentos. Abajo: ejemplos de la categoría 41 (brain); a la izquierda, 1 trazo y 28 segmentos; a la derecha, 1 trazo y 181 segmentos.

Como se puede observar en la Figura 16, la mayoría de las categorías tienen una media del número de segmentos entre 20 y 50, pero todas las categorías tienen por lo menos un registro con más de 500 segmentos, siendo el máximo 3.560 en un registro de la categoría 80 (*cooler)*. Estos registros pueden ser dibujos con un nivel extremo de detalle (lo cual es poco probable debido al tiempo limitado para realizarlos), o *outliers* que solo meten ruido en el conjunto de datos.

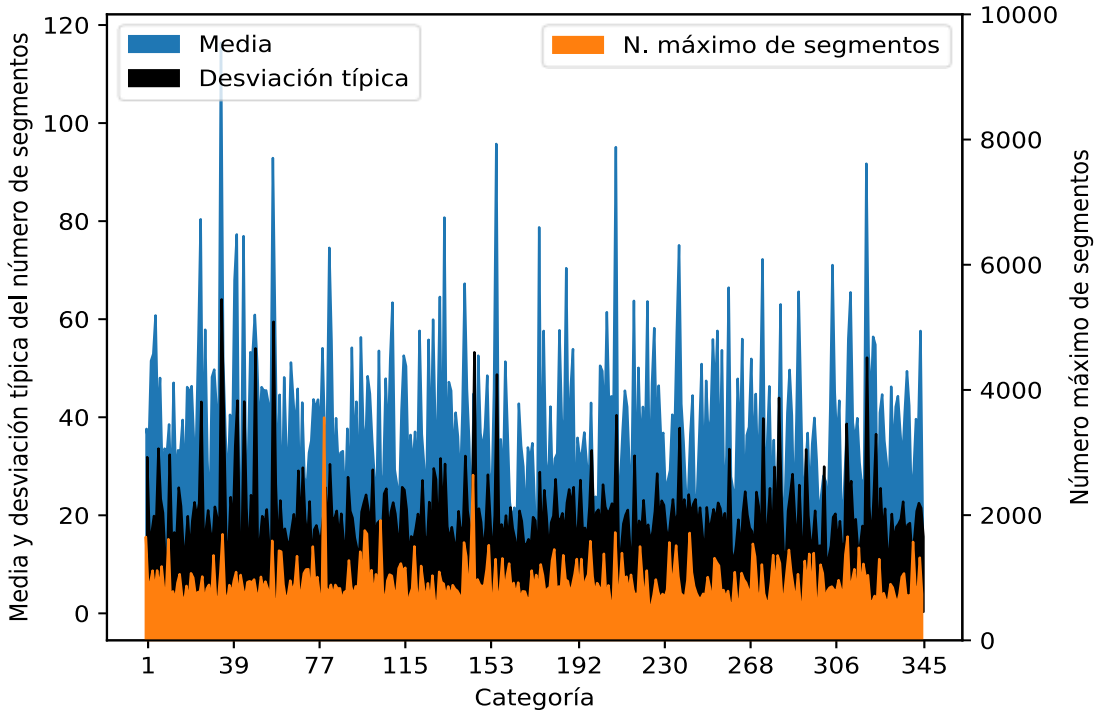


Figura 16. Media, desviación estándar y máximo número de segmentos en cada categoría de *Quick, Draw!*

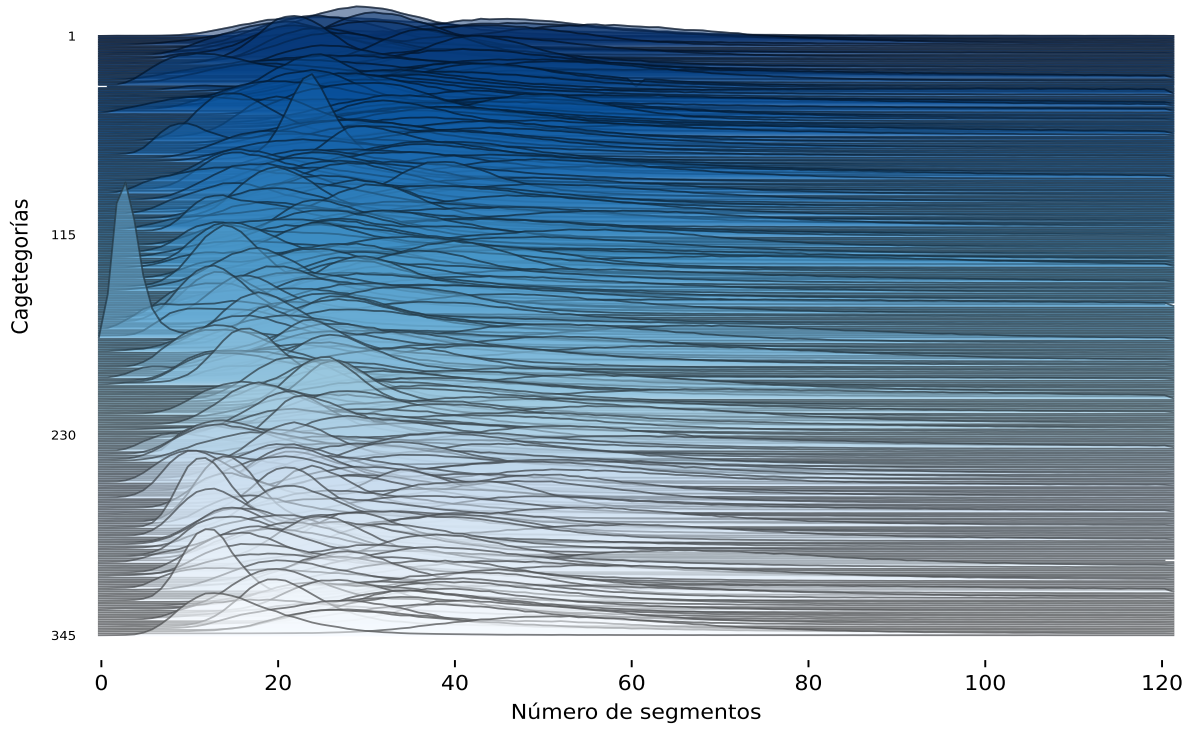


Figura 17. Distribución del número de segmentos de las 345 categorías de *Quick, Draw!. Eje x truncado en 120 segmentos.*

En el Anexo 4 se puede ver con más detalle la distribución de segmentos de cada una de las categorías. Cabe destacar las categorías 34 (*blackberry*), 156 (*hurricane*), y 209 (*panda*) que presentan la media más alta en el número de segmentos, situándose entre 95 y 116. Por otro lado, las categorías 174 (*line*), 325 (*triangle*), 283 (*stairs*) tienen la media más baja en el número de segmentos, situándose entre 10 y 12. Los resultados detallados de la media y la desviación estándar de segmentos de cada categoría han sido incorporados a la tabla del Anexo 1.

3 CONJUNTO DE DATOS PARA EL MODELO DE CLASIFICACIÓN

3.1 Justificación del formato

Poco tiempo después de que el juego de *Quick, Draw!* viera la luz, Google publicó un tutorial de TensorFlow[13] en el cual se describe el cómo crear y entrenar una red neuronal recurrente para clasificar los dibujos del conjunto de datos. [7] [8]

Una de las primeras cosas que llama la atención es que Google no utiliza ninguno de los múltiples formatos del dataset para entrenar el modelo. En su lugar, emplea un formato binario optimizado específicamente para TensorFlow (TF), llamado TFRecord[14]. TF es un *framework* creado por Google para el aprendizaje automático y cálculo numérico, con un énfasis especial en el entrenamiento e inferencia de redes neuronales profundas [9].

Dado que uno de los objetivos de este trabajo es construir una arquitectura para clasificar los registros, se ha optado por utilizar Keras[15] y TF. Keras es una librería de Python que facilita la creación y entrenamiento de modelos de aprendizaje profundo, mientras que TF actúa como *backend* para construir y entrenar la red neuronal encargada de clasificar los dibujos. Por esta razón, se ha decidido utilizar el formato TFRecord para almacenar los datos de entrada.

Además, el proceso de conversión a TFRecord incluye un pre-procesamiento de los datos que se detallará más adelante. Este pre-procesamiento es conveniente, e incluso imprescindible en algunos aspectos, para la creación del modelo de clasificación RNN.

3.2 Creación de conjunto de datos en formato TFRecord

Aunque Google ofrece la descarga de ficheros en formato TFRecord creados a partir de un subconjunto del dataset, este subconjunto consta de 10.000 registros por categoría para entrenamiento y 1.000 por categoría para validación. Cada conjunto está dividido en 10 *shards* o particiones. Estas muestras solo representan entre el 3% y el 8,5% (según la categoría) del conjunto de datos original por lo que su uso se ha descartado. Para poder entrenar el modelo con una muestra mucho más representativa, se ha optado por crear un subconjunto propio de los datos en este formato. Como punto de partida se ha utilizado el código Python proporcionado por el equipo de TF[16] para convertir el *dataset* en formato ndjson simplificado a ficheros TFRecord, adaptándolo a Keras y a TF 2.x.

3.2.1 Pre-procesamiento inicial

Para asegurar la aleatoriedad en los datos, el primer paso fue el de barajar individualmente los registros de cada fichero del *dataset* original ndjson simplificado. Estos 345 ficheros barajados, 1 por categoría, se guardaron en una carpeta con nombre *shuffled_dataset* para después ser utilizados como base en el proceso de conversión a los conjuntos de datos de entrenamiento y de validación en formato TFRecord.

3.2.2 Conversión a TFRecord y estructura del formato

Durante el proceso de conversión al formato TFRecord, se realizaron las siguientes operaciones sobre los registros de los ficheros ndjson de *shuffled_dataset*:

[13] Véase https://www.tensorflow.org/
[14] Véase https://www.tensorflow.org/tutorials/load_data/tfrecord
[15] Véase https://keras.io/
[16] Véase raw.githubusercontent.com/tensorflow/models/r1.13.0/tutorials/rnn/quickdraw/create_dataset.py

1) Eliminar toda la información que no es necesaria para el propósito del clasificador (*key_id*, *recognized*, *timestamp* y *countrycode*). Solo se deja la categoría y los trazos del dibujo (*word* y *drawing*).
2) La categoría (palabra dibujada), se cambia por un índice de categoría (1-345).
3) Los trazos del dibujo se convierten a una estructura de datos que representa una secuencia de todos los puntos que forman los trazos del dibujo, junto con un indicador booleano que indica final de cada trazo. ($[X_1, Y_1, 0]$, $[X_2, Y_2, 0]$,…, $[X_n, Y_n, 1]$,$[X_{n+1}, Y_{n+1}, 0]$…)
4) Se normalizan estos puntos para que quepan en un cuadrado unitario (valores entre 0 y 1 para las coordenadas X e Y).
5) Se convierten las coordenadas de los puntos a deltas de posición, representando los cambios en X e Y entre cada punto y el anterior. Esto permite obtener información sobre la dirección y la magnitud del movimiento del trazo, mientras que se elimina la importancia de sus posiciones absolutas.

La estructura de datos final se ilustra en la Figura 18 y consiste en un número entero, *class_index*, que es un índice correspondiente a la categoría del dibujo; una lista, *ink,* de longitud variable (tamaño [x,3]) con la secuencia de los deltas de las coordenadas normalizadas que forman los trazos del dibujo junto con su indicador de final de trazo; y una variable, *shape,* que indica la dimensión de la lista *ink*.

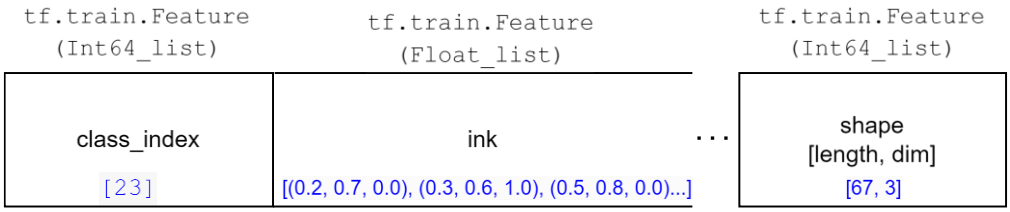


Figura 18. Estructura y ejemplo de datos en fichero TFRecord.

3.2.2 Número de muestras

El objetivo al crear estos ficheros fue por un lado el de elegir un número de muestras cercano al total disponible y al mismo tiempo el de crear un conjunto de datos totalmente balanceado. Para ello se utilizó como referencia el número de muestras de la categoría minoritaria (113.613), fijando finalmente 110.000 registros de cada categoría, quedando el dataset totalmente balanceado con un ratio de desbalanceo de 1.

Estos datos se dividirán de la siguiente manera:

- 80% subconjunto de entrenamiento (88.000 registros por categoría; 30,36 millones de registros en total).
- 20% subconjunto de validación (22.000 registros por categoría; 7,59 millones de registros en total).

3.2.3 Selección de muestras y fragmentación de los ficheros

El código original para la creación del conjunto de datos en formato TFRecord genera 10 particiones TFRecord. Esta división de los datos en múltiples archivos, en lugar de tener millones de registros en uno solo, facilita su manipulación, transferencia y permite barajar los datos sin sobrecargar la

memoria. La documentación de TF recomienda esta práctica para paralelizar la lectura y el procesamientode los datos [17].

Para mezclar los registros, el código aprovecha las particiones. Itera aleatoriamente sobre las 345 particiones del conjunto de datos original (ndjson) y asigna su primer registro a una de las 10 particiones TFRecord, también de forma aleatoria. Este proceso se repite para los registros restantes hasta alcanzar el número deseado por categoría.

El código final, utilizado para crear el conjunto de datos propio, es una adaptación del original para dividir los datos en 100 particiones (en lugar de 10). Esto se decidió hacerlo debido a la necesidad de guardar una cantidad de datos mayor a la que está pensada el código original. El resultado final es de 100 particiones TFRecord para el conjunto de entrenamiento con un peso por partición de entre 176MB y 178MB, y otras 100 para el conjunto de validación con peso de 44MB cada una. Uniendo los dos conjuntos, el peso total es de 21,1 GB.

Tomando en cuenta que el conjunto de datos TFRecord se creó con el 75% del número total de registros del dataset original, y que el peso de este era de 22,3GB, el formato TFRecord ocupa un espacio ligeramente mayor que el original. La diferencia en el tamaño puede explicarse en parte por el cambio en el formato de almacenamiento de los trazos, que ahora se guardan como deltas de posición utilizando números de coma flotante. Aun ocupando un mayor espacio, este formato ofrece ventajas significativas para el procesamiento de datos en TensorFlow. Principalmente, facilita una lectura más eficiente de los datos durante el entrenamiento del modelo al permitir el procesamiento en paralelo de los registros. Esto es particularmente útil para realizar el llenado con ceros necesario para estandarizar la longitud de los datos en los registros durante el entrenamiento.

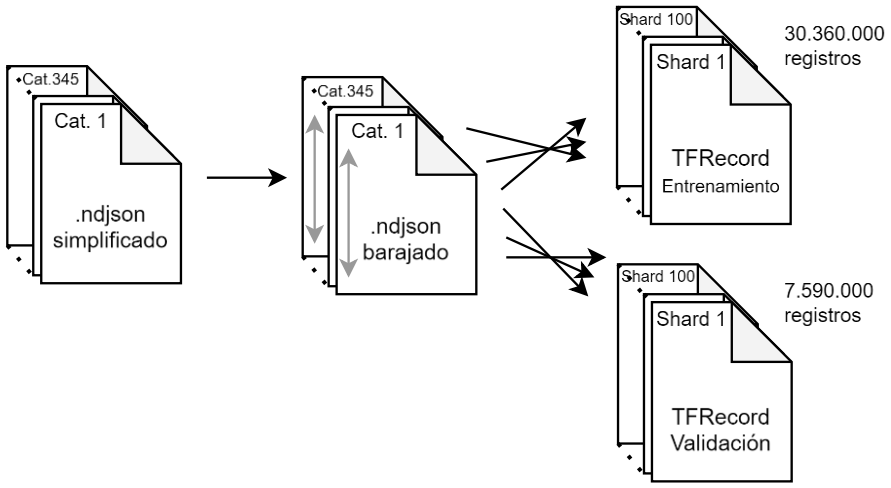


Figura 19. Proceso de creación de los conjuntos de datos en formato TFRecord.

[17] Véase segunda nota dentro de https://www.tensorflow.org/tutorials/load_data/tfrecord

4 MODELO PARA EL CLASIFICADOR

4.1 Marco teórico

4.1.1 Introducción al Aprendizaje Automático y Redes Neuronales

El aprendizaje automático (*machine learning*) es un subcampo de la inteligencia artificial que se centra en el desarrollo de algoritmos que permiten a las máquinas aprender y hacer predicciones a partir de datos.

Las redes neuronales, un enfoque del aprendizaje automático inspirado en el cerebro humano, fueron propuestas desde la década de 1940 pero han tenido fluctuaciones en popularidad y su investigación se ha visto obstaculizada por falta de algoritmos eficientes y falta de poder computacional. Avances recientes, como el uso de GPUs para entrenarlas y el desarrollo de nuevos algoritmos, han impulsado el desarrollo de redes más complejas y potentes. El uso de estas redes con múltiples capas se conoce como aprendizaje profundo o *deep learning*. [10]

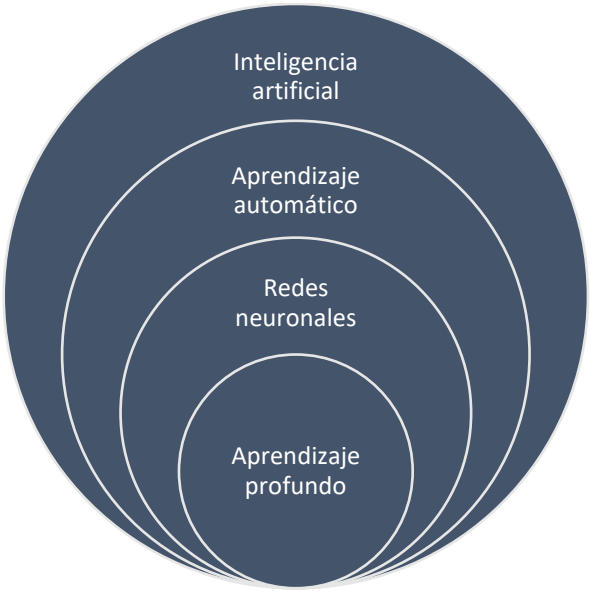


Figura 20. Diagrama de Venn de la estructura jerárquica entre inteligencia artificial, aprendizaje automático, redes neuronales y aprendizaje profundo.

En el caso del *deep learning*, se utilizan diversas capas de procesamiento para aprender representaciones de datos. A través de múltiples niveles de abstracción, se realizan una serie de transformaciones lineales y no lineales que, a partir de los datos de entrada, generan una salida próxima a la esperada. El objetivo del entrenamiento de una red neuronal es el de ajustar los parámetros (pesos y sesgos de cada "neurona") para minimizar la función de pérdida (*loss),* que representa la penalización de una mala predicción. [11]

4.1.2 Aprendizaje supervisado y problemas de clasificación

El objetivo del aprendizaje supervisado es el de predecir una salida a partir de ejemplos de entrada-salida conocidos. En problemas de clasificación, estas salidas representan categorías discretas. El algoritmo analiza los datos de entrenamiento para identificar patrones que relacionan las características de las entradas con sus respectivas clases, construyendo un modelo capaz de generalizar y clasificar nuevas entradas no vistas durante el entrenamiento. La evaluación de este modelo se realiza mediante un conjunto de datos independiente al que se utilizó en el entrenamiento, midiendo su capacidad para predecir correctamente las clases de nuevas instancias. [12]

4.1.3 Redes Neuronales Convolucionales (CNN)

CNN

Una red neuronal convolucional, CNN por sus siglas en inglés, es la que utiliza una función lineal especial llamada convolución. En la práctica, muchas librerías de aprendizaje profundo utilizan una función muy parecida a la convolución llamada función de correlación cruzada [13]. Para el caso de una imagen bidimensional I, y un kernel bidimensional K, esta función se define como:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i + m, j + n) K(m,n) \quad (2)$$

Las CNN se han popularizado enormemente en los últimos años al conseguir muy buenos resultados en el área de visión artificial. Cada capa de una red neuronal convolucional aprende diferentes niveles de abstracción como pueden ser líneas, bordes, o texturas. Las capas convolucionales pueden aprender jerarquías espaciales de patrones preservando relaciones espaciales. Una primera capa puede aprender a reconocer elementos básicos como aristas, y una segunda capa puede aprender patrones compuestos de elementos básicos aprendidos en la capa anterior. Al ir aumentando el número de capas se pueden aprender patrones más complejos y abstractos [11].

En una red neuronal convolucional, a los kernels se les suele llamar filtros. Cada filtro puede detectar una característica y es habitual utilizar múltiples filtros por capa. De cada filtro se obtendrá una capa llamada mapa de características que comparte los mismos pesos (el número de pesos es igual al tamaño del kernel) y un parámetro adicional, el sesgo. El hecho de que todas las neuronas en un mapa de características compartan los mismos parámetros, permite que la red generalice el reconocimiento de patrones a lo largo de toda la imagen y reduce significativamente el número de parámetros del modelo en comparación a las capas densamente conectadas. [14]

Capas convolucionales unidimensionales

Las capas convolucionales también pueden utilizarse con datos en una sola dimensión, como en las series temporales. En estos casos la convolución produce una especie de línea de tiempo que muestra cuándo aparecen diferentes características en los datos de entrada. Si movemos un evento a un momento posterior en la entrada, la misma representación del evento aparecerá en la salida, solo que más tarde. [13]

Etapas en una capa convolucional

Una capa convolucional suele tener tres etapas. La primera fase realiza convoluciones en paralelo con diferentes filtros, resultando en diferentes activaciones lineales. En la segunda etapa, a veces llamada etapa de detección, cada activación lineal se somete a una función de activación no lineal. Por último, la tercera etapa consiste en una operación de agrupación llamada *pooling*.

Operación de Pooling

La operación de *pooling* reemplaza la salida de la red en una ubicación determinada con un valor que resume la información de las salidas cercanas. Existen diferentes técnicas para condensar los datos en una capa de *pooling*, como la media, la norma L^2 o el valor máximo en una vecindad rectangular [13]. Esta última técnica se conoce como *max-pooling* y suele ser una de las más habituales [11].

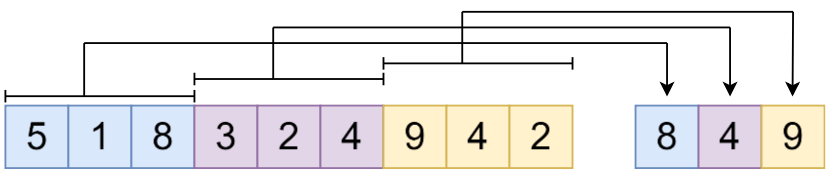


Figura 21. Ejemplo de max-pooling (1D) de tamaño 3 aplicado a un vector de tamaño 9.

Normalización por lotes (BN)

BN es un mecanismo que acelera el entrenamiento de redes neuronales profundas y, en algunos casos, mejora el rendimiento general del modelo. Se normalizan las entradas de cada capa para cada lote durante el entrenamiento, estabilizando así la distribución de las entradas. [15]

4.1.4 Redes Neuronales Recurrentes (RNN), BRNN y LSTM

RNN

Las redes neuronales recurrentes (RNN) son adecuadas para el procesamiento de secuencias temporales y series de datos debido a su capacidad para mantener una memoria interna que captura dependencias temporales. Las RNN se especializan en procesar secuencias de longitud variable y su característica distintiva es su capacidad de aplicar la misma regla de actualización (un conjunto de operaciones matemáticas que modifican el estado interno de la red) en cada paso de tiempo de una secuencia. Esto permite que la red comparta parámetros a lo largo de una estructura computacional profunda, donde cada salida depende no solo de la entrada actual, sino también de las salidas previas (salidas internas de las neuronas en pasos de tiempo anteriores). A diferencia de las redes convolucionales que operan en secuencias temporales mediante la aplicación de filtros locales, las RNN mantienen un estado interno que evoluciona a lo largo del tiempo, lo que les otorga la capacidad de modelar dependencias a largo plazo. [13]

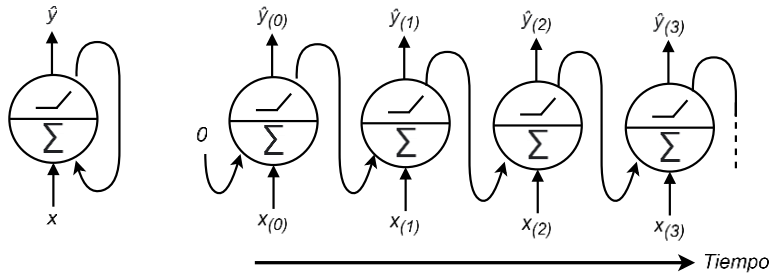


Figura 22. Una neurona recurrente (izquierda) representada en su desarrollo temporal (derecha). (Fuente: adaptado de [14]).

BRNN

Para ciertas tareas es beneficioso tener acceso tanto al contexto futuro como al pasado. Aunque las RNNs estándar procesan secuencias en orden temporal, ignorando el contexto futuro, existen soluciones como agregar ventanas de tiempo futuro a la entrada o introducir un retraso entre entradas y objetivos. Las redes neuronales recurrentes bidireccionales (BRNNs) ofrecen una mejor solución al procesar cada secuencia de entrenamiento hacia adelante y hacia atrás en dos capas ocultas recurrentes separadas, conectadas a la misma capa de salida. Esto proporciona a la capa de salida un contexto pasado y futuro completo para cada punto en la secuencia de entrada. [16]

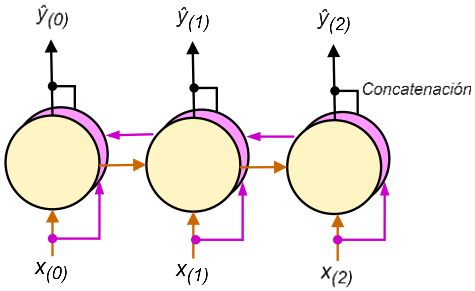


Figura 23. Representación de una red neuronal recurrente bidireccional, donde dos capas recurrentes operan sobre las mismas entradas, una leyendo los datos de izquierda a derecha y la otra de derecha a izquierda, y luego combinando sus salidas en cada paso de tiempo, típicamente mediante concatenación. (Fuente: adaptado de [14]).

LSTM

Para una arquitectura RNN estándar, el rango de contexto al que se puede acceder en la práctica es bastante limitado. El problema es que la influencia de una entrada dada decae o explota exponencialmente a medida que circula por las conexiones recurrentes [16]. Las redes *Long-Short Term Memory* (LSTM), son una variante de las RNN que solucionan este problema y que amplían su memoria mediante el uso de una célula con tres puertas para permitir el paso selectivo de la información. El mecanismo de cada una de estas puertas—la puerta de entrada, la de olvidar, y la de salida—es análogo a una función de activación sigmoide que va de 0 a 1 [17].

Utilizar LSTM como arquitectura de red en una red neuronal recurrente bidireccional da como resultado LSTM bidireccional. Este modelo tiene la capacidad de considerar dependencias a largo plazo en ambas direcciones de la entrada. [16]

4.1.5 Funciones de activación

En una red neuronal, una función de activación es la que decide si una neurona debe activarse o no, y con qué intensidad. Las funciones de activación introducen no linealidad a la red y hacen posible el tener una red profunda (si todas las capas fueran lineales estas podrían combinarse en una sola capa).

El modelo más antiguo y simple de una de estas neuronas es el perceptrón, el cual toma un vector de entrada binario y produce una salida binaria utilizando un umbral. Aprende ajustando sus parámetros para minimizar el error en la estimación de la salida para cada entrada. Sin embargo, su función de activación es discontinua, lo que lo hace inadecuado para técnicas de optimización basadas en gradientes. Las redes neuronales modernas usan otras funciones de activación no lineales, como la sigmoide o la tangente hiperbólica, y más recientemente, la función de unidad lineal rectificada (ReLU) que muchas veces funciona mejor en la práctica para las redes neuronales profundas. [18]

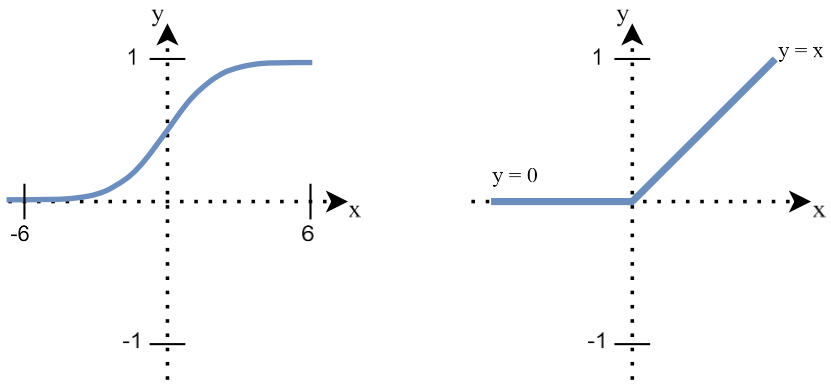


Figura 24. Función sigmoide (izquierda) y función ReLU (derecha).

Softmax

La función softmax transforma un conjunto de números reales en probabilidades. Se utiliza en modelos de aprendizaje automático, como la regresión logística multinomial, y a menudo como la capa final de una red neuronal para producir una distribución de probabilidad sobre las posibles clases de salida. [19]

4.2 Implementación del modelo

En esta sección se detalla el proceso de implementación del modelo de clasificación para el reconocimiento de los dibujos del conjunto de datos *Quick, Draw!,* así como las herramientas y referencias que han sido fundamentales para su desarrollo. Este modelo de clasificación se utilizará posteriormente para clasificar el total de los registros del *dataset* de entrenamiento y asignar una puntuación de clasificación a cada categoría.

4.2.1 Estructura del modelo

Para la creación del modelo de clasificación, se ha utilizado como referencia la estructura del modelo de clasificación descrita en el trabajo de *Quick, Stat!,* así como el modelo y código propuestos por el equipo de TensorFlow en [8], pero adaptado para TF 2.x. El modelo está basado en la arquitectura de red neuronal convolucional (CNN) y recurrente (RNN) secuencial.

El modelo se compone de las siguientes capas:

- **Capa de entrada**: Esta capa define la entrada de la red, que consiste en secuencias de 3711 elementos (el valor máximo de número de puntos encontrado en el dataset), con 3 dimensiones cada uno (delta de posición x, delta de posición y, e indicador de final de trazo).

- **Capa de enmascaramiento**: Esta capa se encarga de ignorar los valores 0.0 en las entradas. Esto es útil ya que los datos son de longitud variable y se usará un relleno con ceros.

- **Capas convolucionales 1D**:
  - Conv1D con 48 filtros, tamaño kernel=5, activación=ReLU. + Capa BN (normalización)
  - Conv1D con 64 filtros, tamaño kernel=5, activación=ReLU. + Capa BN (normalización)
  - Conv1D con 96 filtros, tamaño kernel=3, activación=ReLU. + Capa BN (normalización)

- **Capas LSTM bidireccionales:**
  - LSTM bidireccional (analiza la secuencia en ambas direcciones) con 128 unidades. El parámetro *return=True* indica que la capa devolverá una secuencia de salida en lugar de un solo vector.
  - LSTM bidireccional (similar a la anterior).

- **Capa de agrupación:**
  - GlobalAveragePooling1D: Promedia los valores a lo largo de la dimensión de la secuencia, reduciendo la salida de las capas LSTM a un solo vector.

- **Capa de salida:**
  - Capa densa (completamente conectada) que produce 345 salidas (número de categorías en Quick, Draw!). La función de activación softmax normaliza las salidas a valores entre 0 y 1, interpretándolas como probabilidades de pertenencia a cada clase.

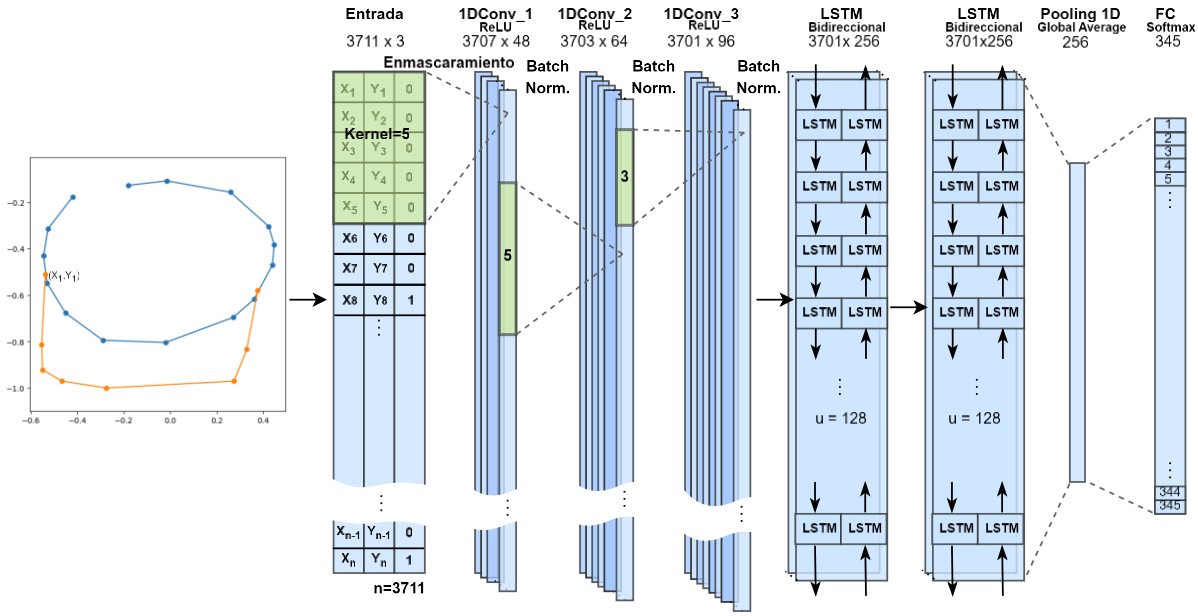


Figura 25. Representación gráfica del modelo de la red neuronal utilizada como clasificador.

4.2.2 Implementación en Keras

Para la creación del modelo se ha utilizado Keras con backend de TensorFlow 2.x. Se ha creado una función, *build_model()*, en la que crean las capas de la red neuronal. A continuación se muestra el fragmento de código Python utilizado para la creación del modelo.

```
from tensorflow.keras import layers, models

def build_model():
    model = models.Sequential()

    # Capa entrada y filtro
    model.add(layers.Input(shape=(3711, 3)))
    model.add(layers.Masking(mask_value=0.0))

    # Capas convolucionales
    model.add(layers.Conv1D(48, 5, activation='relu'))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv1D(64, 5, activation='relu'))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv1D(96, 3, activation='relu'))
    model.add(layers.BatchNormalization())

    # LSTM
    model.add(layers.Bidirectional(layers.LSTM(128,return_sequences=True)))
    model.add(layers.Bidirectional(layers.LSTM(128,return_sequences=True)))

    # Pooling y Softmax
    model.add(layers.GlobalAveragePooling1D())
    model.add(layers.Dense(345, activation='softmax'))

    return model
```

La Figura 26 muestra un resumen de la arquitectura de la red, generado por la librería *keras_sequential_ascii*[18].

```
           OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

               Input   #####   3711    3
             Masking   ????? -------------------         0     0.0%
                       #####   3711    3
              Conv1D    \|/  -------------------       768     0.1%
                relu   #####   3707   48
  BatchNormalization    μ|σ  -------------------       192     0.0%
                       #####   3707   48
              Conv1D    \|/  -------------------     15424     2.1%
                relu   #####   3703   64
  BatchNormalization    μ|σ  -------------------       256     0.0%
                       #####   3703   64
              Conv1D    \|/  -------------------     18528     2.5%
                relu   #####   3701   96
  BatchNormalization    μ|σ  -------------------       384     0.1%
                       #####   3701   96
       Bidirectional   ????? -------------------    230400    30.8%
                       #####   3701  256
       Bidirectional   ????? -------------------    394240    52.6%
                       #####   3701  256
GlobalAveragePooling1D Y^avg -------------------         0     0.0%
                       #####         256
               Dense   XXXXX -------------------     88665    11.8%
             softmax   #####         345
=================================================================
Total params: 748857 (2.86 MB)
Trainable params: 748441 (2.86 MB)
Non-trainable params: 416 (1.62 KB)
```

Figura 26. Resumen de la arquitectura de la red neuronal, mostrando las dimensiones de los datos en cada capa, el número de pesos y el porcentaje del total de pesos en cada operación.

Se puede observar que más del 82% de los pesos de la red neuronal están en las capas LSTM bidireccionales. Esta concentración de parámetros indica que la red neuronal otorga una gran importancia a la secuencia temporal de los trazos en los dibujos. Por otro lado, las capas convolucionales iniciales, aunque con menos pesos, permiten extraer pequeñas características locales de los trazos que se irán detectando a lo largo del tiempo por las capas LSTM que les suceden.

[18] https://github.com/stared/keras-sequential-ascii

5 ENTRENAMIENTO DEL MODELO

5.1 Marco teórico

5.1.1 Optimización y funciones de pérdida

El proceso de entrenamiento de una red neuronal consiste en aprender los valores de los parámetros de la red (pesos y sesgos) que minimicen la pérdida (*loss*). Este es un proceso iterativo que consta de dos fases: la propagación hacia adelante (*forwardpropagation*) y la retropropagación (*backpropagation*). En la propagación hacia adelante se expone la red a los datos de entrenamiento y estos cruzan toda la red hasta convertirse en una predicción, es decir, una etiqueta. Después se calcula que tan buena ha sido esa predicción utilizando una función de pérdida. Una vez calculada la pérdida, se propaga esta información hacia atrás partiendo de la capa de salida y pasando por cada una de las capas de la red. Cada neurona que contribuye a la salida recibe una señal proporcional a su contribución aproximada a la predicción. Un algoritmo de optimización modifica los pesos de las conexiones en cada iteración aproximando la pérdida lo más posible a cero. [11]

Prácticamente todos los algoritmos de optimización en el aprendizaje profundo están basados en un algoritmo llamado descenso de gradiente estocástico (SGD). Este algoritmo es una extensión del algoritmo de descenso de gradiente (GD) para poder utilizar un conjunto de datos de entrenamiento grande sin que sea computacionalmente prohibitivo. SGD, en lugar de calcular el gradiente de la función de pérdida en el conjunto de datos completo, hace una estimación del gradiente utilizando una pequeña fracción de los datos. Luego, sigue el gradiente estimado cuesta abajo con el objetivo de encontrar mínimos en la función de pérdida [13]. Existen múltiples variantes de SDG que intentan mejorar su rendimiento. Algunos de los más comunes son: AdaGrad, RMSProp y Adam [20].

Heurística de recorte de gradiente

En [13] se detalla que en redes neuronales profundas, especialmente en redes recurrentes que procesan secuencias temporales largas, pueden surgir regiones con gradientes extremadamente grandes (explosión de gradientes). Estos gradientes desproporcionados pueden llevar a actualizaciones de parámetros inestables y perjudicar el proceso de aprendizaje. La heurística de recorte de gradiente aborda este problema estableciendo un umbral para la magnitud del gradiente. Si la norma del gradiente supera este umbral, se escala (recorta) para que su magnitud quede dentro de un rango seguro. Esto evita actualizaciones de parámetros excesivamente grandes y mejora la estabilidad del entrenamiento.

5.2.2 Hiperparámetros

Los hiperparámetros son parámetros que controlan el proceso de aprendizaje. No se aprenden durante el entrenamiento del modelo, sino que se establecen antes de comenzar. Los hiperparámetros más importantes en el entrenamiento de redes neuronales se exponen a continuación.

Tamaño del lote (batch size)

Determina el número de muestras que se utilizan para calcular el gradiente de la función de pérdida en cada iteración. En [13], se hacen unas puntualizaciones importantes sobre los factores que definen el tamaño de lote:

- Lotes grandes mejoran la estimación del gradiente, pero con beneficios decrecientes.
- La memoria limita el tamaño del lote.
- Lotes de tamaño potencia de 2 suelen ofrecer mejor rendimiento en GPUs.
- Lotes pequeños tienen un efecto regularizador y disminuyen el error de generalización, mejorando el rendimiento del modelo en datos nuevos. Este efecto es óptimo con lotes de tamaño 1, posiblemente debido a la "aleatoriedad" que añaden al aprendizaje. Sin embargo,

esto requiere una tasa de aprendizaje más baja para mantener la estabilidad, y aumenta el tiempo de entrenamiento.

Tasa de aprendizaje (learning rate)

Controla la magnitud de las actualizaciones de los pesos en cada iteración del entrenamiento y se puede ver como la velocidad en la que se sigue el gradiente de la función de pérdida. Una tasa de aprendizaje demasiado alta puede llevar a que el modelo no converja, mientras que una tasa demasiado baja puede hacer que el entrenamiento sea muy lento y que quede atrapado en un mínimo local. Es una práctica común ir modificando la tasa de aprendizaje mientras va avanzando el entrenamiento del modelo. Una forma de hacerlo es ir decrementando la tasa de aprendizaje linealmente hasta llegar a un número de iteración predefinida. Un indicador de que la tasa de aprendizaje es demasiado alta es cuando la curva de aprendizaje muestra oscilaciones abruptas. [13]

Número de epochs

Indica cuántas veces el algoritmo de optimización recorrerá el conjunto de datos de entrenamiento para actualizar los parámetros del modelo. Un número insuficiente de *epochs* puede llevar a un modelo subajustado, mientras que un número excesivo puede provocar sobreajuste. En [13] se hace una observación interesante sobre los *datasets* que son tan grandes en tamaño que el sobreajuste deja de ser un problema. Debido esta tendencia al alza en el tamaño de los conjuntos de datos, está convirtiéndose en una práctica común en el aprendizaje automático utilizar una sola vez cada registro o incluso hacer una pasada incompleta a través del conjunto de datos de entrenamiento. En estos casos, el término *epoch* no se refiere a recorrer el conjunto de datos completo, sino a esa pasada incompleta.

5.1.3 Métricas de evaluación

Las métricas de evaluación permiten cuantificar el rendimiento de un modelo de clasificación durante el entrenamiento y la validación. En este trabajo, se utilizaron las siguientes métricas:

- Pérdida (*Loss*): La función de pérdida mide el error del modelo en sus predicciones. En este caso, se utilizó la entropía cruzada categórica dispersa (*sparse categorical crossentropy*) [21], que es adecuada para problemas de clasificación multiclase donde las etiquetas son números enteros. El objetivo del entrenamiento es minimizar esta pérdida.

- Precisión (*Accuracy*): Mide la proporción de predicciones correctas sobre el total de predicciones. Es una métrica útil cuando las clases están equilibradas, como en este caso, donde se tiene el mismo número de muestras por cada categoría durante el entrenamiento.

- Pérdida de validación (*Validation loss*): Es el valor de la función de pérdida calculado en el conjunto de datos de validación. Sirve para evaluar cómo el modelo generaliza a datos no vistos durante el entrenamiento. Un buen modelo debería tener una pérdida de validación baja.

- Precisión de validación (*Validation accuracy*): Es la precisión calculada en el conjunto de datos de validación. Indica el rendimiento del modelo en datos nuevos y ayuda a detectar el sobreajuste. Un buen modelo debería tener una precisión de validación alta.

5.2 Entorno para el entrenamiento

Debido al tamaño del modelo y del conjunto de datos, y tras realizar unas pruebas iniciales, se descartó el entrenar el modelo en el entorno Google Colaboratory. En su lugar, se optó por utilizar un entorno propio con mayor almacenamiento y sin restricciones de uso. A continuación se presentan las características de hardware y software utilizadas.

5.2.1 Hardware

Se utilizó un computador personal con una tarjeta gráfica NVidia RTX 2070 de 8GB RAM y con 2.304 núcleos CUDA. La memoria RAM de la tarjeta gráfica determinó el tamaño máximo de lote utilizable durante el entrenamiento, mientras que la cantidad de núcleos CUDA definió la capacidad computacional disponible.

5.2.2 Software

Para garantizar el entrenamiento del modelo en Windows utilizando los núcleos CUDA de la GPU, en lugar del procesador principal, se seleccionaron las siguientes versiones de software, asegurando su compatibilidad mutua:

- Python 3.10
- Keras 2.10.0
- Cudnn 8.1.0
- Tensorflow 2.10.0

5.3 Configuración y ejecución del entrenamiento

En esta sección se describe el proceso de entrenamiento del modelo clasificador, detallando los valores de los hiperparámetros elegidos y los pasos seguidos para utilizar los ficheros de entrenamiento y validación en formato TFRecord. Para el código utilizado para el entrenamiento del modelo se ha utilizado como referencia el código propuesto por el equipo de TensorFlow [19], adaptándolo a Keras y a TF 2.x.

5.3.1 Configuración de hiperparámetros

El tamaño de lote se estableció en 64, el máximo permitido por la tarjeta gráfica. Esto implica que los 30.360.000 registros del dataset de entrenamiento se procesarían en 474.375 pasos (número de registros / tamaño de lote). Sin embargo, dado que cada paso requiere aproximadamente 1 segundo, completar un *epoch* con todos los registros tomaría entre 5 y 6 días. Para facilitar la gestión del entrenamiento y el seguimiento del progreso, se decidió establecer 15.812 pasos por *epoch* (total de pasos / 30), lo que reduce la duración de cada *epoch* a unas 4 horas. Al finalizar cada *epoch*, se realiza una fase de validación con 500 pasos (equivalente a 32.000 registros del *dataset* de validación) y se guarda un archivo con los pesos del modelo. Esta estrategia permite procesar prácticamente todos los registros del conjunto de datos en 30 epochs, garantizando que no se pierda todo el progreso del entrenamiento si este se interrumpe. En caso de ser necesario, el entrenamiento podría reanudarse partiendo de cualquiera de los archivos de pesos generados.

Los hiperparámetros restantes se establecieron siguiendo las recomendaciones del equipo de TF para el modelo clasificador. En concreto, la tasa de aprendizaje se fijó en 1e-4 y el recorte de gradiente (*clipnorm*) en 9. Como algoritmo de optimización se utilizó Adam [20]. El detalle de todos los hiperparámetros y sus valores se presenta en la Tabla 5.

[19] Véase https://raw.githubusercontent.com/tensorflow/models/r1.13.0/tutorials/rnn/quickdraw/train_model.py

Tabla 5. Valores de los hiperparámetros para el entrenamiento del modelo.

Hiperparámetro	Valor
Tamaño de lote	64
Pasos por epoch	*15812*
Número de epochs	30
Pasos de validación	*500*
Tasa de aprendizaje	*1e-4*
Recorte de gradiente	*9.0*
Optimizador	Adam
Función de pérdida	Entropía cruzada categórica dispersa

5.3.2 Estrategia de procesamiento de datos TFRecord

El haber creado los conjuntos de datos de entrenamiento y validación en particiones con formato TFRecord ayudó a garantizar la aleatoridad en el orden de carga de los registros sin sobrecargar los recursos del sistema. Al inicio del proceso, el orden de los 100 ficheros correspondientes a cada conjunto de datos se baraja para finalmente cargarlos como uno solo. Este barajado inicial no consume prácticamente recursos y asegura un orden diferente en los registros en cada ciclo de entrenamiento. Después, para asegurar la aleatoriedad en los lotes formados durante cada paso del entrenamiento, se utiliza un búfer (*shuffle buffer*)[20] de tamaño de 303.600 registros (1/100 parte del total de muestras). Se toman los primeros registros del conjunto de datos hasta llenar el búfer, luego se van tomando los registros del búfer de forma aleatoria en cada paso del entrenamiento, reemplazando los registros utilizados con nuevos registros. Esto mantiene el búfer lleno hasta que se agoten todos los registros del conjunto de datos de entrenamiento. Es aquí cuando termina un ciclo de entrenamiento, habiendo transcurrido un total de 30 *epochs*.

El formato TFRecord también ofreció ventajas de optimización en el pre-procesamiento y en la carga de datos. A medida que se leen los registros TFRecord, dado que los dibujos tienen longitudes variables, se aplica un relleno de ceros (*padding*)[21] para que todas las secuencias alcancen la longitud máxima definida (3711 puntos, que es el número de puntos que tiene el registro con mayor longitud en todo el *dataset*). Esta función añade ceros después de los valores contenidos en el tensor que guarda los trazos, hasta alcanzar la longitud deseada de 3711. Este paso es crucial para asegurar que todos los registros tengan la misma dimensionalidad, requisito indispensable para el procesamiento por lotes. El formato TFRecord permitió acelerar este proceso mediante la opción *num_parallel_calls=tf.data.AUTOTUNE*[22] *de TF*, que permite procesar los registros en paralelo de forma optimizada. Adicionalmente, se utiliza la función *prefetch*[23] para cargar los datos del siguiente lote en segundo plano mientras se procesa el lote actual.

5.3.3 Ejecución del entrenamiento

El entrenamiento del modelo se realizó en ciclos de 30 *epochs*. Cada *epoch* con una duración aproximada de 4 horas, lo que resulta en un tiempo total de entrenamiento por ciclo de

[20] Véase https://www.tensorflow.org/api_docs/python/tf/data/Dataset#shuffle
[21] Véase https://www.tensorflow.org/api_docs/python/tf/pad
[22] Véase https://www.tensorflow.org/guide/data_performance#parallelizing_data_transformation
[23] Véase https://www.tensorflow.org/guide/data_performance#prefetching

aproximadamente 5 días. La ejecución de cada ciclo se inició manualmente, partiendo de un fichero de pesos seleccionado del ciclo anterior.

Tras finalizar el primer ciclo de entrenamiento, habiendo recorrido por primera vez el conjunto de datos de entrenamiento completo (30 *epochs* y 474.360 pasos en total), el modelo alcanzó una precisión de clasificación del 65,8% y una pérdida de 1,368 (Figura 27). Durante esta primera fase de entrenamiento se observó un aumento progresivo en la precisión del modelo al finalizar cada *epoch*, indicando un aprendizaje satisfactorio. Sin embrago, también se observó un comportamiento anómalo en la precisión reportada al final de cada *epoch* en la fase de validación (Figura 28). Aunque en general la precisión de validación acompañaba la tendencia de la precisión, en algunos *epochs* se observaron disminuciones bruscas en la precisión de validación, seguidas de recuperaciones en *epochs* posteriores. A pesar de este comportamiento, se descartó la posibilidad de sobreajuste, dado que el modelo solo había recorrido el conjunto de datos una vez, y se decidió continuar el entrenamiento, considerando que el modelo seguía mostrando una tendencia general de mejora.

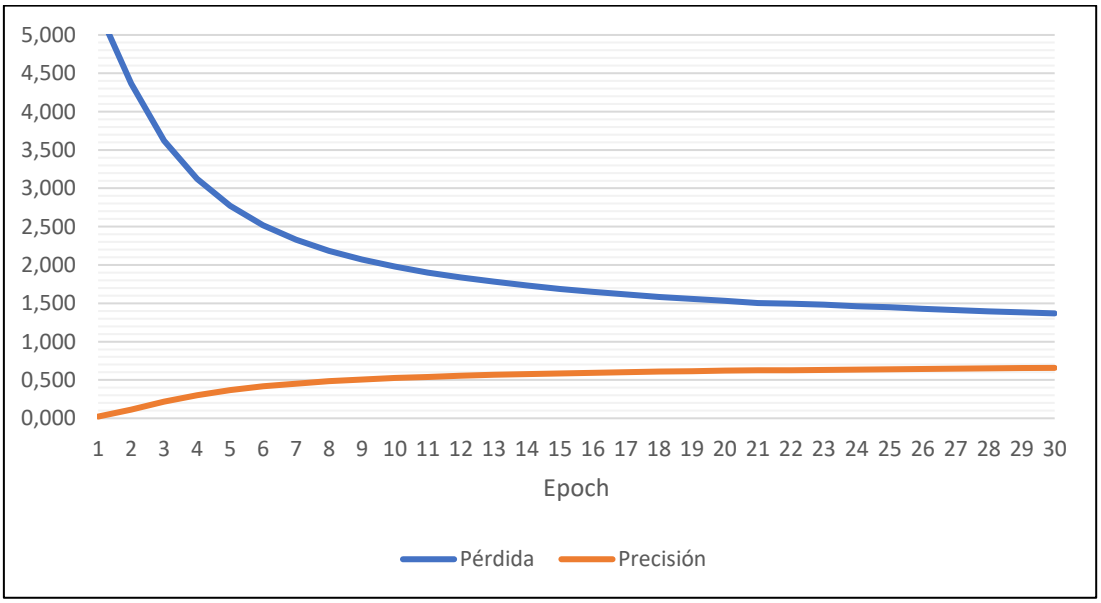


Figura 27. Evolución de la precisión y la pérdida en el primer ciclo de entrenamiento. En 30 *epochs* (474.360 pasos*), el modelo alcanzó una* precisión del 65,8% y una pérdida de 1,368.

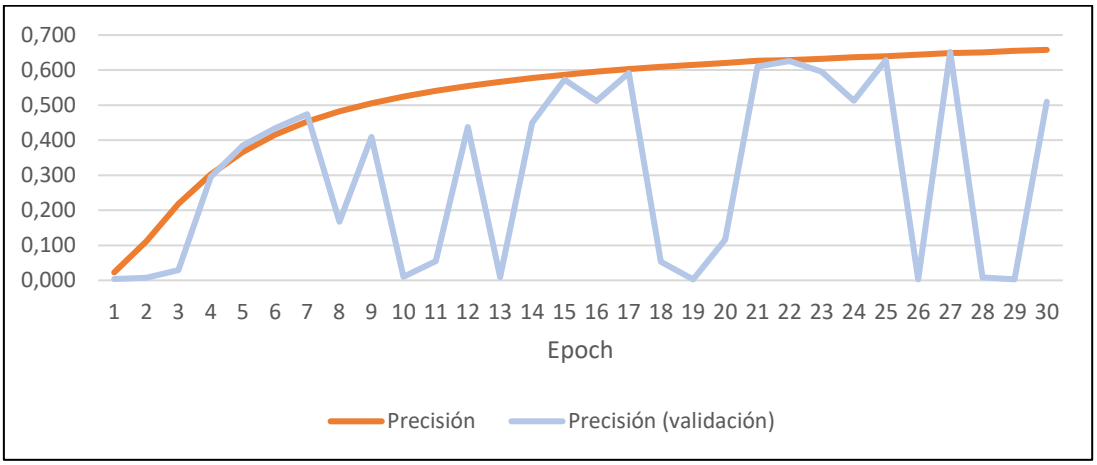


Figura 28. Evolución de la precisión y de la precisión de validación en el primer ciclo de entrenamiento. La precisión muestra un comportamiento fluctuante, aumentando en algunos *epochs* y disminuyendo bruscamente en otros, para luego recuperarse en *epochs* posteriores.

A partir del segundo ciclo de entrenamiento, se redujo la tasa de aprendizaje en un factor de 10, pasando de 1e-4 a 1e-5, para tratar de solventar las fluctuaciones observadas en la precisión de validación. Se realizaron 8 ciclos adicionales de entrenamiento con esta tasa de aprendizaje reducida, siempre partiendo de un fichero de pesos de un ciclo anterior. El fichero fue seleccionado manualmente, experimentando con diferentes criterios de selección en cada ciclo: 1) el de mayor precisión, ignorando la precisión de validación; 2) el de mayor precisión, pero con una precisión de validación alta; y 3) el de mayor precisión de validación. El criterio que aparentemente ofreció mejores resultados fue el de seleccionar la mayor precisión ignorando la precisión de validación, ya que la precisión seguía aumentando (aunque cada vez a un ritmo más lento) y se observó que la precisión de validación tendía a recuperarse en *epochs* posteriores.

Al finalizar el noveno ciclo de entrenamiento, habiendo procesado en total más de 4 millones de pasos, el modelo alcanzó el 76,8% de precisión y 0.904 de pérdida (76,4% de precisión de validación y 0.904 de pérdida de validación). Dado que en este último ciclo solo se obtuvo una mejora del 0.01% en la precisión de entrenamiento respecto al ciclo anterior, se evidenciaron signos de estancamiento en el aprendizaje. Por ello, se decidió realizar un último ciclo adicional, reduciendo la tasa de aprendizaje a 1e-6. Con esta tasa, el comportamiento oscilante en la precisión de validación desapareció, y se logró mejorar la precisión del modelo en un 0.003%.

Aunque el entrenamiento parecía haber concluido, la adquisición de una nueva tarjeta gráfica con mayor capacidad de memoria permitió explorar el uso de un tamaño de lote más grande. Se realizó un último ciclo de entrenamiento con un tamaño de lote de 160 y una tasa de aprendizaje ligeramente superior, 2e-6 (ver Tabla 6). Este ajuste resultó en una mejora adicional del 0.003% en la precisión del modelo (ver Figura 29). Tras este último ciclo, se dio por finalizado el entrenamiento. El fichero de pesos seleccionado para el modelo clasificador fue aquel que obtuvo la mejor precisión de validación, cuyas métricas finales se detallan en la Tabla 7.

Tabla 6. Hiperparámetros (tamaño de lote y tasa de aprendizaje) utilizados en cada ciclo de entrenamiento y métricas del fichero de pesos seleccionado al finalizar cada ciclo.

Ciclo	Tamaño de lote	Tasa de aprendizaje	Precisión	Precisión de validación	Pérdida	Pérdida de validación
1	64	1e-4	0.658	0.510	1.368	2.152
2	64	1e-5	0.693	0.696	1.217	1.218
3	64	1e-5	0.734	0.729	1.043	1.058
4	64	1e-5	0.743	0.003	1.007	20.526
5	64	1e-5	0.747	0.730	0.988	1.047
6	64	1e-5	0.751	0.003	0.967	12.180
7	64	1e-5	0.756	0.004	0.951	12.439
8	64	1e-5	0.758	0.746	0.940	1.002
9	64	1e-5	0.768	0.764	0.904	0.923
10	64	1e-6	0.769	0.768	0.897	0.903
11	160	2e-6	0.771	0.769	0.893	0.906

Tabla 7. Métricas de evaluación del fichero de pesos seleccionado para el modelo clasificador.

Precisión	0.7705
Pérdida	*0.8926*
Precisión de validación	0.7686
Pérdida de validación	0.906

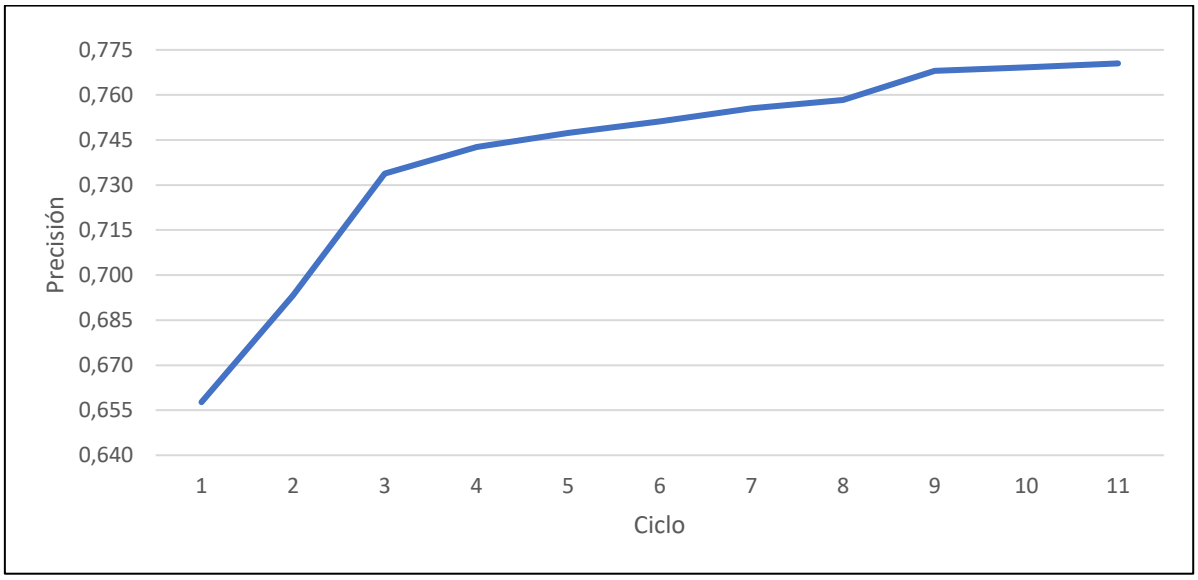


Figura 29. Evolución de la precisión del modelo a lo largo de 11 ciclos de entrenamiento. Cada ciclo equivale a 15812 pasos (1 epoch). A partir del ciclo 9, se observa un estancamiento en la mejora de la precisión.

5.4 Evaluación del dataset utilizando el modelo de clasificación

Una vez entrenado el modelo, se cargó el fichero de pesos obtenido en el entrenamiento (Tabla 7), y se procesó el *dataset* de validación completo (7,59 millones de registros) clasificando todos sus registros. Se calcularon las siguientes métricas para cada una de las 345 categorías:

- **Índice de confianza promedio**: Este índice representa la media de las probabilidades asignadas por el modelo a las predicciones correctas de una categoría específica. Un valor alto indica que el modelo tiende a estar más seguro al predecir correctamente dicha categoría.

- **Desviación estándar del índice de confianza**: Esta métrica mide la dispersión de los valores del índice de confianza para una categoría determinada. Una desviación estándar baja sugiere que el modelo es más consistente en su nivel de confianza al predecir correctamente esa categoría.

- **F1-Score**: El F1-score es una métrica que combina la precisión y el *recall*, proporcionando una medida equilibrada del rendimiento del modelo en cada clase.

- **Matriz de confusión**: Tabla que muestra la distribución de clasificaciones del modelo, indicando cuántos registros de cada categoría fueron clasificados correctamente y, en el caso de clasificaciones incorrectas, en qué categoría fueron clasificados.

Adicionalmente, se realizó un estudio con algunas categorías al azar (18 categorías) para obtener una estimación del número de *outliers* en el conjunto de datos. Aunque este estudio no entra dentro de los objetivos principales del proyecto, se consideró importante realizarlo para poder comparar resultados con el análisis similar de tres categorías incluido en el trabajo de referencia *quick, stat!*. El procedimiento de este análisis fue el de convertir a formato TFRecord los archivos del *dataset* original de las categorías seleccionadas, para finalmente evaluar todos sus registros con el modelo clasificador, agrupándolos en intervalos basados en desviaciones estándar.

5.5 Resultados

Las métricas completas obtenidas a través del modelo clasificador han sido integradas a la tabla del Anexo 1. Como resultados relevantes, se presentan las tablas 8-13 que muestran las 5 categorías con los valores más altos y más bajos en cada una de las métricas del modelo clasificador. La matriz de confusión resultó ser demasiado densa para poder incorporarla a una tabla o figura; sin embargo, su análisis ha contribuido a los resultados presentados en la siguiente sección.

Tabla 8. Las 5 categorías con mayor índice de confianza promedio. En orden descendente: 289 (stop sign), 0,97; 141 (headphones), 0,96; 283 (stairs), 0,96; 284 (star), 0,95; 165 (ladder), 0,95.

				N. Trazos		N. Segmentos		Índice Confianza		
Categoría		Tres muestras	N. dibujos	μ	σ	μ	σ	μ	σ	F-1
289	stop sign (señal de alto)		119.814	7,16	2,80	39,15	13,10	0,97	0,10	0,94
141	headphones (audífonos)		118.906	3,42	2,04	41,17	17,73	0,96	0,11	0,93
283	stairs (escaleras)		128.981	1,77	2,48	15,03	12,88	0,96	0,12	0,92
284	star (estrella)		137.619	1,60	1,98	24,11	10,35	0,95	0,12	0,93
165	ladder (escalera)		125.389	5,96	1,99	15,57	11,79	0,95	0,12	0,93

Tabla 9. Las 5 categorías con menor índice de confianza promedio. En orden ascendente: 181 (marker), 0,50; 327 (truck), 0,59; 80 (cooler), 0,60; 195 (mug), 0,60; 48 (bus), 0,61.

				N. Trazos		N. Segmentos		Índice Confianza		
Categoría		Tres muestras	N. dibujos	μ	σ	μ	σ	μ	σ	F-1
181	marker (marcad		319.136	3,53	3,14	24,50	19,87	0,50	0,17	0,38
327	truck (camión)		131.354	5,41	2,73	44,65	16,03	0,59	0,16	0,51
80	cooler (enfriado		271.444	5,47	4,34	35,95	25,69	0,60	0,21	0,47
195	mug (taza)		152.918	3,72	2,34	36,95	17,45	0,60	0,12	0,62
48	bus (autobús		166.208	7,36	3,22	52,79	19,89	0,61	0,14	0,59

Tabla 10. Las 5 categorías con mayor dispersión en el índice de confianza. En orden descendente: 57 (camouflage), 0,30; 126 (frog), 0,25; 192 (mouse), 0,24; 181 (marker), 0,17; 92 (dog), 0,25.

				N. Trazos		N. Segmentos		Índice Confianza		
Categoría		Tres muestras	N. dibujos	μ	σ	μ	σ	μ	σ	F-1
57	camouflage (camuflaje)		172.710	7,60	7,23	92,82	59,49	0,63	0,30	0,49
126	frog (rana)		159.047	6,81	3,95	55,82	22,73	0,70	0,25	0,59
92	dog (perro)		152.159	7,18	3,92	54,25	19,47	0,65	0,25	0,54
192	mouse (ratón)		178.826	5,47	3,45	35,84	18,27	0,74	0,24	0,61
227	pliers (alicates)		172.549	4,08	3,59	36,44	28,53	0,75	0,24	0,60

Tabla 11. Las 5 categorías con menor dispersión en el índice de confianza. En orden ascendente: 289 (stop sign), 0,10; 105 (envelope), 0,11; 141 (headphones), 0,11; 284 (star), 0,12; 165 (ladder), 0,12.

				N. Trazos		N. Segmentos		Índice Confianza		
Categoría		Tres muestras	N. dibujos	μ	σ	μ	σ	μ	σ	F-1
289	stop sign (señal de alto)		119.814	7,16	2,80	39,15	13,10	0,97	0,10	0,94
105	envelope (sobre)		134.863	2,56	2,06	22,87	15,06	0,91	0,11	0,92
141	headphones (audífonos)		118.906	3,42	2,04	41,17	17,73	0,96	0,11	0,93
284	star (estrella)		137.619	1,60	1,98	24,11	10,35	0,95	0,12	0,93
165	ladder (escalera)		125.389	5,96	1,99	15,57	11,79	0,95	0,12	0,93

Tabla 12. Las 5 categorías con mayor puntuación F1. En orden descendente: 289 (stop sign), 0,94; 141 (headphones), 0,93; 284 (star), 0,93; 165 (ladder), 0,93; 105 (envelope), 0,92.

				N. Trazos		N. Segmentos		Índice Confianza		
Categoría		Tres muestras	N. dibujos	μ	σ	μ	σ	μ	σ	F-1
289	stop sign (señal de alto)		119.814	7,16	2,80	39,15	13,10	0,97	0,10	0,94
141	headphones (audífonos)		118.906	3,42	2,04	41,17	17,73	0,96	0,11	0,93
284	star (estrella)		137.619	1,60	1,98	24,11	10,35	0,95	0,12	0,93
165	ladder (escalera)		125.389	5,96	1,99	15,57	11,79	0,95	0,12	0,93
105	envelope (sobre)		134.863	2,56	2,06	22,87	15,06	0,91	0,11	0,92

Tabla 13. Las 5 categorías con menor puntuación F1. En orden ascendente: 181 (marker), 0,38; 48 (bus), 0,59; 327 (truck), 0,51; 57 (camouflage), 0,49; 1 (aircraft carrier), 0,47.

				N. Trazos		N. Segmentos		Índice Confianza		
Categoría		Tres muestras	N. dibujos	μ	σ	μ	σ	μ	σ	F-1
181	marker (marcador)		319.136	3,53	3,14	24,50	19,87	0,50	0,17	0,38
24	bear (oso)		134.762	6,74	3,86	55,92	22,44	0,62	0,21	0,47
80	cooler (enfriador)		271.444	5,47	4,34	35,95	25,69	0,60	0,21	0,47
1	aircraft carrier (portaaviones)		116.504	6,43	4,86	37,64	31,88	0,64	0,23	0,47
128	garden hose (manguera de		121.843	8,53	5,50	59,91	29,58	0,64	0,23	0,48

Las categorías seleccionadas aleatoriamente para el análisis de *outliers son:* 18, 36, 75, 83, 87, 93, 158, 180, 185, 192, 195, 230, 231, 294, 312, 320, 325, 341. Los resultados obtenidos se pueden consultar en la Tabla 14, que muestra el número de registros, y en la Tabla 15, que muestra el porcentaje de registros.

Tabla 14. Distribución del número de registros en función de la probabilidad dada por el clasificador para pertenecer a su categoría. Las columnas representan los siguientes intervalos basados en desviaciones estándar (σ) respecto al promedio (μ) de la probabilidad asignada a cada categoría: σ: [μ, μ+σ), -σ: [μ-σ, μ), 2σ: [μ+σ, μ+2σ), -2σ: [μ-2σ, μ-σ), 3σ: [μ+2σ, μ+3σ), -3σ: [μ-3σ, μ-2σ), 4σ: [μ+3σ, μ+4σ), -4σ: [μ-4σ, μ-3σ), 5σ: [μ+4σ, μ+5σ), -5σ: [μ-5σ, μ-4σ), 6σ: [μ+5σ, μ+6σ), -6σ: [μ-6σ, μ-5σ).

Cat.	σ	-σ	2σ	-2σ	3σ	-3σ	4σ	-4σ	5σ	-5σ	6σ	-6σ
18	75.215	15.010	0	8.387	0	7.187	0	8.131	0	14.719	0	6.726
36	66.085	18.215	0	10.518	0	8.179	0	8.119	0	8.248	0	0
75	64.782	18.440	0	10.529	0	7.767	0	7.048	0	11.699	0	0
83	69.174	17.218	0	9.281	0	7.119	0	6.870	0	10.414	0	6.854
87	60.958	17.472	0	11.379	0	9.508	0	10.264	0	13.829	0	0
93	54.736	15.951	0	12.349	0	12.356	0	19.083	0	7.138	0	0
158	106.568	28.716	0	18.037	0	16.703	0	20.098	0	24.002	0	0
180	73.473	15.081	0	8.118	0	6.153	0	6.184	0	11.620	0	0
185	65.026	16.333	0	9.809	0	8.319	0	9.029	0	12.054	0	0
192	41.202	21.925	19.360	23.053	0	49.265	0	24.021	0	0	0	0
195	36.125	30.005	18.118	20.269	754	15.466	0	13.435	0	18.746	0	0
230	21.614	18.183	12.796	20.493	0	44.302	0	16.051	0	0	0	0
231	74.856	13.652	0	7.564	0	5.764	0	5.728	0	8.913	0	10.230
294	73.430	14.614	0	7.782	0	6.222	0	6.549	0	15.765	0	0
312	85.830	13.202	0	5.605	0	3.703	0	2.978	0	2.829	0	7.236
320	39.666	34.585	22.180	20.892	0	11.122	0	6.075	0	4.688	0	4.063
325	85.230	16.200	0	6.366	0	3.400	0	2.234	0	1.816	0	7.924
341	89.379	13.968	0	5.763	0	3.766	0	2.902	0	2.643	0	13.881

Tabla 15. Distribución porcentual del número de registros en función de la probabilidad dada por el clasificador para pertenecer a su categoría. Las columnas representan los siguientes intervalos basados en desviaciones estándar (σ) respecto al promedio (μ) de la probabilidad asignada a cada categoría: σ: [μ, μ+σ), -σ: [μ-σ, μ), 2σ: [μ+σ, μ+2σ), -2σ: [μ-2σ, μ-σ), 3σ: [μ+2σ, μ+3σ), -3σ: [μ-3σ, μ-2σ), 4σ: [μ+3σ, μ+4σ), -4σ: [μ-4σ, μ-3σ), -5σ: [μ-5σ, μ-4σ), -6σ: [μ-6σ, μ-5σ). Se han excluido las columnas correspondientes a 5σ y 6σ ya que no contienen datos.

Cat.	σ	-σ	2σ	-2σ	3σ	-3σ	4σ	-4σ	-5σ	-6σ
18	55,56%	11,09%	0,00%	6,20%	0,00%	5,31%	0,00%	6,01%	10,87%	4,97%
36	55,36%	15,26%	0,00%	8,81%	0,00%	6,85%	0,00%	6,80%	6,91%	0,00%
75	53,87%	15,33%	0,00%	8,75%	0,00%	6,46%	0,00%	5,86%	9,73%	0,00%
83	54,50%	13,57%	0,00%	7,31%	0,00%	5,61%	0,00%	5,41%	8,20%	5,40%
87	49,39%	14,16%	0,00%	9,22%	0,00%	7,70%	0,00%	8,32%	11,21%	0,00%
93	45,01%	13,12%	0,00%	10,15%	0,00%	10,16%	0,00%	15,69%	5,87%	0,00%
158	49,77%	13,41%	0,00%	8,42%	0,00%	7,80%	0,00%	9,39%	11,21%	0,00%
180	60,91%	12,50%	0,00%	6,73%	0,00%	5,10%	0,00%	5,13%	9,63%	0,00%
185	53,93%	13,55%	0,00%	8,14%	0,00%	6,90%	0,00%	7,49%	10,00%	0,00%
192	23,04%	12,26%	10,83%	12,89%	0,00%	27,55%	0,00%	13,43%	0,00%	0,00%
195	23,62%	19,62%	11,85%	13,25%	0,49%	10,11%	0,00%	8,79%	12,26%	0,00%
230	16,20%	13,63%	9,59%	15,36%	0,00%	33,20%	0,00%	12,03%	0,00%	0,00%
231	59,08%	10,77%	0,00%	5,97%	0,00%	4,55%	0,00%	4,52%	7,03%	8,07%
294	59,05%	11,75%	0,00%	6,26%	0,00%	5,00%	0,00%	5,27%	12,68%	0,00%
312	70,71%	10,88%	0,00%	4,62%	0,00%	3,05%	0,00%	2,45%	2,33%	5,96%
320	27,69%	24,14%	15,48%	14,58%	0,00%	7,76%	0,00%	4,24%	3,27%	2,84%
325	69,20%	13,15%	0,00%	5,17%	0,00%	2,76%	0,00%	1,81%	1,47%	6,43%
341	67,56%	10,56%	0,00%	4,36%	0,00%	2,85%	0,00%	2,19%	2,00%	10,49%

5.5.1 Análisis de resultados

El análisis de los resultados obtenidos tras la clasificación de 7,59 millones de registros (22.000 por cada una de las 345 categorías) utilizando el modelo de clasificación entrenado revela información valiosa sobre la calidad y características del conjunto de datos de *Quick, Draw!*.

En primer lugar, se observa una amplia variación en el índice de confianza promedio entre las categorías. Mientras algunas categorías como 289 (*stop sign*), 141 (*headphones*) y 283 (*stairs*) presentan una alta confianza (superior a 0,95), otras como 181 (*marker*), 327 (*truck*) y 80 (*cooler*) muestran una confianza notablemente inferior (alrededor de 0,50-0,61). Esta variabilidad sugiere que algunas categorías son inherentemente más difíciles de clasificar que otras, ya sea debido a la ambigüedad en los dibujos, la similitud con otras categorías o la calidad de los datos de entrenamiento. La categoría 181 (*marker*), con el índice de confianza más bajo, podría representar un desafío particular para el modelo debido a la simplicidad de los trazos en los dibujos y su similitud con otras categorías como la 218 (*pencil*), categoría que la matriz de confusión muestra que se confunde con esta categoría con mucha frecuencia. Por otro lado, la categoría 289 (*stop sign*), con el índice de confianza más alto, podría ser más fácil de reconocer debido a su forma distintiva y estandarizada.

En cuanto a la dispersión del índice de confianza, se observa una mayor variabilidad en categorías como 57 (*camouflage*), 126 (*frog*) y 92 (*dog*) en comparación con categorías como 289 (*stop sign*), 105 (*envelope*) y 141 (*headphones*). Una alta variabilidad del índice de confianza de una categoría puede

deberse a que los datos de entrenamiento contienen más ruido (dibujos sin sentido o que no corresponden a la categoría), o que sea una categoría que se confunda fácilmente con otras. La categoría 57 (*camouflage*), siendo la categoría con mayor variabilidad en el índice de confianza, al analizar los resultados de la matriz de confusión se observó que es la categoría que se confunde con un mayor número de otras categorías.

Se observa una alta correlación en el promedio del índice de confianza y la puntuación F1. Esto sugiere que las categorías en las que el modelo muestra mayor confianza en sus predicciones correctas también tienden a tener un mejor rendimiento general (la puntuación F1 contempla también el número de falsos negativos, es decir, número de muestras de esa categoría que se han clasificado erróneamente como si fueran de otra). El análisis de la puntuación F1 revela un buen rendimiento del modelo en la mayoría de las categorías, con valores superiores a 0,70. Sin embargo, algunas categorías como 181 (*marker*), 24 (*bear*) y 80 (*cooler*) presentan puntuaciones F1 inferiores a 0,50, lo que indica un rendimiento deficiente en estas categorías. Es importante destacar que no se encontró una correlación directa entre la puntuación F1 y el número de trazos o segmentos en los dibujos. Esto sugiere que la complejidad estructural de los dibujos no es el factor determinante del rendimiento del modelo en la clasificación.

En el análisis de valores atípicos (*outliers*) en los datos, consideramos los registros con una probabilidad menor a 4 desviaciones estándar de la media. De las 18 categorías analizadas, los resultados fueron:

- 9 categorías (18, 83, 87, 158, 185, 195, 231, 294, 341): Entre 10% y 15% de sus registros son valores atípicos.
- 7 categorías (36, 75, 158, 180, 312, 320, 325): Entre 5% y 10% de sus registros son valores atípicos.
- 2 categorías (192 y 230): No presentaron ningún valor atípico.

Estos datos muestran una variación significativa en la presencia de registros atípicos entre las diferentes categorías. Esta variación puede deberse a que existen categorías más fáciles de representar que otras. Las categorías difíciles de representar tienen mayor probabilidad de contener dibujos sin sentido, incompletos, o tachados por frustración del participante. Por último, los datos proporcionan una idea del porcentaje general de registros atípicos en el conjunto completo de datos: aproximadamente un 10%.

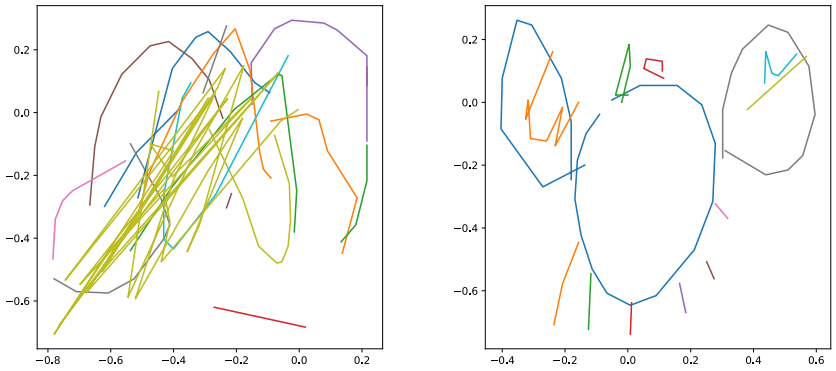


Figura 30. Dos registros de la categoría 83 (crab) pertenecientes al grupo de *outliers* por tener una probabilidad de pertenecer a su categoría menor a 4 desviaciones estándar de la media. Izquierda: ejemplo de ruido (dibujo tachado sin sentido). Derecha: muestra atípica con baja probabilidad de clasificación debido a su diferencia con ejemplos típicos de la categoría.

6 CONCLUSIONES

Quick, Draw! es el *dataset* vectorial más grande de dibujos de trazos simples a un solo color en la actualidad. Con 50.426.266 registros distribuidos en 345 categorías, el análisis de este conjunto de datos ha resultado en las siguientes observaciones y aportes:

1. Muestra limitada: Aún siendo el conjunto de datos de dibujos más grande en la actualidad, el *dataset* disponible al público es solo una pequeña fracción de los datos recopilados por Google.

2. Desbalance de categorías: Existe una disparidad significativa en la cantidad de muestras por categoría, llegando algunas a triplicar a otras en número.

3. Sesgo geográfico: Aunque los datos provienen de diversas partes del mundo, hay una sobrerrepresentación notable de Estados Unidos. Este sesgo geográfico puede limitar la generalización de los modelos a otras regiones y culturas.

4. Calidad de los datos: A pesar de la afirmación de Google sobre la moderación individual de los registros, es evidente la presencia de ruido en el *dataset*: dibujos sin sentido, tachados o incompletos. El análisis de valores atípicos (*outliers*) revela que aproximadamente el 10% de los registros en algunas categorías podrían considerarse atípicos. Sin embargo, no todos estos pueden considerarse ruido. Algunos son dibujos válidos que representan correctamente su categoría, pero difieren significativamente de otras muestras típicas, lo que explica su clasificación como atípicos.

5. Falta de actualizaciones: Contrario a lo prometido, Google no ha actualizado el conjunto de datos desde su publicación hace 6 años.

6. Formato de los datos: Para el entrenamiento de modelos neuronales con TensorFlow, se recomienda la conversión al formato TFRecord, ya que ninguno de los formatos originales es óptimo para este propósito.

7. Complejidad de los dibujos: Las categorías con mayor complejidad promedio contienen hasta 14 trazos y 116 segmentos. Sin embargo, se han identificado registros extremos con hasta 3.711 puntos (suma de segmentos y trazos), que complican innecesariamente el modelo de clasificación. Se recomienda filtrar estos casos atípicos, ya que probablemente representen ruido en los datos. Tomando en cuenta el tiempo limitado que se tuvo para hacer cada dibujo, es prácticamente imposible realizar un dibujo con tanto detalle como para tener tantos puntos.

8. Rendimiento del modelo clasificador: No se encontró correlación entre la puntuación asignada por el modelo a cada categoría y la complejidad de los dibujos (medida por el número medio de segmentos o trazos). Esto sugiere que la complejidad estructural de los dibujos no influye significativamente en la confianza o precisión del modelo al realizar las clasificaciones. Factores como la ambigüedad en la representación de objetos, la similitud entre categorías y la calidad de los datos de entrenamiento parecen tener un mayor impacto en el rendimiento del modelo.

9. El desarrollo de una arquitectura y un modelo en código abierto para la clasificación de todos los dibujos del conjunto de datos. El código y el archivo con los pesos del modelo entrenado pueden descargarse de: https://github.com/falcalde21/Quick-Stat-TFM

6.1 Trabajos futuros

En futuras investigaciones se podría abordar los desafíos identificados en este análisis, como el desarrollo de estrategias de preprocesamiento más sofisticadas y la creación de modelos más robustos al ruido y a la variabilidad entre categorías. La exploración de nuevas arquitecturas de redes neuronales y técnicas de entrenamiento podría mejorar aún más el rendimiento de los modelos en la clasificación y generación de dibujos a partir de este conjunto de datos.

7 BIBLIOGRAFÍA

[1] I. Johnson, «Drawings in the Cloud: introducing the Quick, Draw! dataset,» diciembre 2017. [En línea]. Available: https://cloud.google.com/blog/products/gcp/drawings-in-the-cloud-introducing-the-quick-draw-dataset. [Último acceso: febrero 2024].

[2] D. Ha y D. Eck, «A Neural Representation of Sketch Drawings,» *CoRR,* vol. abs/1704.03477, 2017.

[3] R. Fernandez-Fernandez, J. G. Victores, D. Estevez y C. Balaguer, «Quick, Stat!: A Statistical Analysis of the Quick, Draw! Dataset,» de *EUROSIM 2019*, Logroño, 2019.

[4] G. Pajares y J. M. de la Cruz, «Transformación de imágenes,» de *Visión por computador: imágenes digitales y aplicaciones. 2ª edición*, Madrid, RA-MA, 2007, p. 59.

[5] G. Jaiswal y S. P. Kannojia, «Effects of Varying Resolution on Performance of CNN based Image Classification An Experimental Study,» *International Journal of Computer Sciences and Engineering,* vol. 6, nº 9, pp. 451-456, 2018.

[6] A. Orriols-Puig y E. Bernado-Mansilla, «Evolutionary rule-based systems for imbalanced datasets,» *Soft Computing,* vol. 13, nº 3, pp. 213-225, 2009.

[7] «Recurrent Neural Networks for Drawing Classification,» TensorFlow, noviembre 2017. [En línea]. Available: http://web.archive.org/web/20171209094809/https://www.tensorflow.org/versions/master/tutorials/recurrent_quickdraw.

[8] MarkDaoust, «GitHub repo tensorflow/docs,» 2020. [En línea]. Available: https://github.com/tensorflow/docs/blob/master/site/en/r1/tutorials/sequences/recurrent_quickdraw.md. [Último acceso: septiembre 2023].

[9] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin , S. Ghemawat, G. Irving, M. Isard , M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu y X. Zheng, «TensorFlow: A System for Large-Scale Machine Learning,» de *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, Savannah, GA., 2016.

[10] L. Hardesty, «Explained: Neural networks. MIT News Office.,» 14 abril 2017. [En línea]. Available: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[11] J. Torres, Deep Learning, Introducción práctica con Keras (PRIMERA PARTE), Barcelona: Kindle Direct Publishing, 2018.

[12] S. J. Russell y P. Norvig, Artificial Intelligence: A Modern Approach, Pearson Education, Inc., 2010.

[13] I. Goodfellow, Y. Bengio y A. Courville, Deep Learning, MIT Press, 2016.

[14] A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media, Inc., 2017.

[15] S. Ioffe y C. Szegedy, «Batch normalization: Accelerating deep network training by reducing internal covariate shift,» de *International conference on machine learning*, Lille, France, 2015.

[16] A. Graves, Studies in Computational Intelligence,Volume 385: Supervised Sequence Labelling with Recurrent Neural Networks., Berlin, Heidelberg: Springer, 2012.

[17] J. Torres, Deep Learning, Introducción práctica con Keras (SEGUNDA PARTE), Barcelona: Kindle Direct Publishing, 2019.

[18] S. Mwanje y C. Mannweiler, Towards Cognitive Autonomous Networks, John Wiley & Sons Ltd., 2020.

[19] Wikipedia, «Softmax function,» [En línea]. Available: https://en.wikipedia.org/wiki/Softmax_function.

[20] D. P. Kingma y J. L. Ba, «Adam: A method for stochastic optimization,» arXiv preprint arXiv:1412.6980, 2014.

[21] Keras Team, «Keras documentation: Probabilistic losses,» [En línea]. Available: https://keras.io/api/losses/probabilistic_losses/#sparsecategoricalcrossentropy-class. [Último acceso: 2024].

8 LISTADO DE SIGLAS, ABREVIATURAS, ACRÓNIMOS Y ANGLICISMOS

API: Interfaz de programación de aplicaciones (*application programming interface*).

Array: Variable estructurada formada por un número de variables simples del mismo tipo.

Accuracy: Precisión.

Backend: La "parte trasera" del código. Se refiere a toda la parte oculta (no visible para el usuario), que hace funcionar a una aplicación.

Batch: Lote (conjunto) de datos.

BN: normalización por lotes (*batch normalization)*

BRNN: Red neuronal recurrente bidireccional (*bidirectional recurrent neural network*).

CNN: Red neuronal convolucional (*convolutional neural network)*.

Dataset: Conjunto organizado de datos que se utiliza para realizar análisis.

Datetime: Formato de datos que incluye la fecha y la hora.

Epoch: Traducido a veces como época, iteración o ciclo. Una iteración completa sobre un conjunto de datos o sobre una muestra de este.

Feature: Característica.

Feature map: Mapa de características o mapa de activación. Matriz que refleja la respuesta de una CNN a una entrada, indicando la fuerza de activación para cada ubicación y permitiendo la extracción de características y patrones relevantes.

Framework: Biblioteca de código, marco de trabajo, entorno de trabajo, o esquema de trabajo generalmente utilizado por programadores para realizar desarrollo de software.

GD: Descenso de gradiente (gradient descent).

GPU: Unidad de procesamiento de gráficos (graphics processing unit).

Learning rate: Tasa de aprendizaje.

MIT: Massachusetts Institute of Technology.

Padding: Relleno. Técnica que consiste en añadir valores adicionales (generalmente ceros) alrededor de los bordes de una matriz o vector de datos.

Paper: Artículo científico.

Pooling: Combinación de características extraídas de diferentes partes de una entrada para crear una representación más compacta y significativa. A veces traducida como capa de agrupación, de reducción o de submuestreo.

Raw: En su forma original, sin procesar ni modificar.

Recall: Sensibilidad o exhaustividad. Es el ratio de verdaderos positivos.

Remuestrear: Del inglés *resample.* Proceso de cambiar la resolución de una imagen o señal digital.

SGD: Descenso de gradiente estocástico (stochastic gradient descent).

Shard: Fragmento o partición horizontal de una base de datos más grande.

String: Cadena de caracteres.

RNN: Red neuronal recurrente (*recurrent neural network*).

TF: TensorFlow.

Timestamp: Marca temporal, conocida también como sello de tiempo o registro de tiempo.

VAE: Autoencoder variacional.

9 ANEXOS

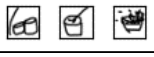
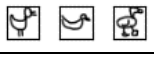
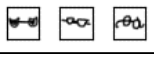
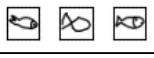
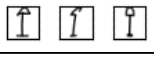
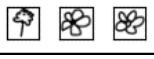
9.1 Anexo 1. Listado de categorías, número de muestras por categoría, media y desviación estándar del número de trazos, número de segmentos y puntuación de clasificación de cada categoría.

				N. Trazos		N. Segmentos		Índice Confianza		
Categoría		Tres muestras	N. dibujos	μ	σ	μ	σ	μ	σ	F-1
1	aircraft carrier (portaaviones)		116.504	6,43	4,86	37,64	31,88	0,64	0,23	0,47
2	airplane (avión)		151.623	4,64	3,10	34,08	15,71	0,81	0,18	0,80
3	alarm clock (despertador)		123.399	8,23	4,52	51,52	16,49	0,91	0,16	0,87
4	ambulance (ambulancia)		148.004	7,68	3,72	52,91	19,88	0,82	0,19	0,73
5	angel (ángel)		149.736	7,86	3,54	60,77	18,65	0,93	0,15	0,88
6	animal migration (migración de animales)		137.847	8,03	6,04	38,92	33,62	0,80	0,24	0,66
7	ant (hormiga)		124.612	8,96	3,32	48,04	23,35	0,90	0,17	0,85
8	anvil (yunque)		126.231	2,69	2,90	28,35	20,53	0,85	0,19	0,79
9	apple (manzana)		144.722	2,85	1,74	33,63	11,38	0,90	0,15	0,88
10	arm (brazo)		120.951	2,59	2,31	32,55	12,21	0,84	0,18	0,78
11	asparagus (espárrago)		168.102	6,63	5,90	38,50	32,39	0,76	0,22	0,68
12	axe (hacha)		124.122	3,13	2,07	23,52	13,52	0,82	0,20	0,75
13	backpack (mochila)		125.801	5,24	2,61	47,06	16,52	0,87	0,19	0,83
14	banana (plátano)		307.936	2,47	2,33	27,32	15,14	0,80	0,20	0,75
15	bandage (vendaje/tirita)		147.614	6,52	5,48	33,33	25,66	0,91	0,17	0,82
16	barn (granero)		151.139	6,33	4,00	33,00	21,85	0,80	0,20	0,74
17	baseball bat (bate de béisbol)		123.809	4,98	3,35	39,48	13,28	0,86	0,19	0,79
18	baseball (béisbol)		135.375	2,52	2,43	25,22	13,44	0,87	0,17	0,80
19	basket (canasta)		118.458	7,49	4,66	46,19	19,83	0,84	0,20	0,75
20	basketball (baloncesto)		133.793	5,58	2,59	44,96	12,11	0,82	0,18	0,77
21	bat (murciélago)		118.114	4,96	3,93	46,37	18,94	0,87	0,19	0,79
22	bathtub (bañera)		174.336	4,82	3,71	34,99	22,67	0,76	0,19	0,72

				N. Trazos		N. Segmentos		Índice Confianza		
Categoría			N. dibujos	μ	σ	μ	σ	μ	σ	F-1
23	beach (playa)		124,938	8.27	6.35	35.13	19.53	0.79	0.21	0.75
24	bear (oso)		134,762	6.74	3.86	55.92	22.44	0.62	0.21	0.47
25	beard (barba)		165,202	6.60	6.38	80.38	43.16	0.81	0.19	0.76
26	bed (cama)		113,862	5.94	3.14	28.18	16.04	0.84	0.20	0.77
27	bee (abeja)		120,890	9.23	3.30	57.89	16.97	0.91	0.16	0.88
28	belt (cinturón)		191,119	5.59	3.58	31.22	21.02	0.83	0.21	0.74
29	bench (banco)		128,695	4.56	3.52	19.62	16.72	0.68	0.20	0.60
30	bicycle (bicicleta)		126,527	8.02	3.66	48.21	14.37	0.80	0.15	0.80
31	binoculars (binoculares)		124,190	6.86	3.04	49.76	21.75	0.88	0.19	0.82
32	bird (pájaro)		133,572	6.85	4.20	42.33	19.52	0.64	0.19	0.54
33	birthday cake (pastel de cumpleaños)		144,982	6.89	3.29	37.79	16.59	0.65	0.14	0.63
34	blackberry (mora)		128,153	7.71	6.94	116.45	64.05	0.72	0.21	0.63
35	blueberry (arándano)		127,878	4.42	4.41	51.43	40.15	0.77	0.23	0.64
36	book (libro)		119,364	7.21	3.78	36.29	14.86	0.84	0.19	0.81
37	boomerang (búmeran)		142,682	1.67	2.06	25.92	12.87	0.86	0.18	0.79
38	bottlecap (tapa de botella)		153,207	4.74	4.18	40.48	23.72	0.81	0.21	0.68
39	bowtie (corbata de moño)		130,283	3.32	2.29	32.48	17.12	0.94	0.14	0.90
40	bracelet (pulsera)		119,416	6.15	4.39	67.98	34.10	0.79	0.20	0.72
41	brain (cerebro)		143,033	8.07	5.86	77.32	43.42	0.83	0.20	0.78
42	bread (pan)		120,570	2.89	2.93	26.10	14.21	0.80	0.23	0.67
43	bridge (puente)		133,010	6.77	3.78	26.14	16.18	0.83	0.20	0.78
44	broccoli (brócoli)		132,826	5.22	5.67	76.97	43.22	0.82	0.18	0.78
45	broom (escoba)		116,927	6.21	5.71	29.82	21.59	0.77	0.19	0.70
46	bucket (cubo)		124,064	3.03	1.84	31.45	10.86	0.76	0.18	0.71
47	bulldozer (buldócer)		187,409	7.28	3.48	53.34	24.12	0.81	0.21	0.77
48	bus (autobús)		166,208	7.36	3.22	52.79	19.89	0.61	0.14	0.59

				N. Trazos		N. Segmentos		Índice Confianza		
Categoría			N. dibujos	μ	σ	μ	σ	μ	σ	F-1
49	bush (arbusto)		120,520	3.52	4.88	60.89	54.06	0.81	0.19	0.73
50	butterfly (mariposa)		117,999	4.64	2.45	51.47	12.60	0.94	0.13	0.92
51	cactus (cactus)		131,676	6.32	6.45	36.76	15.01	0.92	0.15	0.89
52	cake (pastel)		124,905	8.41	5.05	46.11	19.84	0.64	0.14	0.60
53	calculator (calculadora)		128,375	7.72	4.26	45.39	18.06	0.88	0.16	0.86
54	calendar (calendario)		321,981	9.64	4.22	45.52	21.22	0.77	0.20	0.69
55	camel (camello)		121,399	2.91	2.90	42.93	14.03	0.92	0.15	0.89
56	camera (cámara)		128,772	3.80	2.12	36.84	12.47	0.90	0.15	0.87
57	camouflage (camuflaje)		172,710	7.60	7.23	92.82	59.49	0.63	0.30	0.49
58	campfire (fogata)		133,395	6.10	3.86	58.57	26.38	0.88	0.17	0.87
59	candle (vela)		141,545	3.91	2.16	25.19	12.22	0.87	0.18	0.83
60	cannon (cañón)		141,394	6.42	3.81	44.60	23.04	0.84	0.21	0.78
61	canoe (canoa)		123,767	2.26	2.28	19.50	15.52	0.79	0.20	0.74
62	car (coche)		182,764	5.05	3.14	48.14	17.28	0.71	0.18	0.69
63	carrot (zanahoria)		132,459	6.02	3.71	32.84	14.73	0.91	0.16	0.87
64	castle (castillo)		122,534	4.22	3.25	33.14	14.05	0.86	0.17	0.82
65	cat (gato)		123,202	9.88	3.98	51.17	17.22	0.81	0.19	0.76
66	ceiling fan (ventilador de techo)		115,413	5.47	3.23	43.92	20.23	0.87	0.17	0.83
67	cell phone (teléfono celular)		121,130	5.61	4.30	37.18	15.50	0.74	0.17	0.72
68	cello (violonchelo)		149,725	6.40	3.94	45.89	29.10	0.68	0.16	0.62
69	chair (silla)		222,706	5.07	3.17	20.98	16.51	0.87	0.17	0.84
70	chandelier (candelabro)		167,502	7.29	5.18	43.02	29.74	0.88	0.19	0.81
71	church (iglesia)		164,225	5.93	3.47	27.75	19.26	0.85	0.19	0.76
72	circle (círculo)		122,876	1.27	1.42	25.48	15.37	0.72	0.16	0.74
73	clarinet (clarinete)		126,214	6.47	4.50	32.74	22.84	0.78	0.22	0.67

	Categoría		N. dibujos	N. Trazos		N. Segmentos		Índice Confianza		F-1
				μ	σ	μ	σ	μ	σ	
74	clock (reloj)		120,536	4.82	2.74	35.19	10.07	0.92	0.13	0.89
75	cloud (nube)		120,265	1.57	1.88	43.65	17.05	0.85	0.17	0.82
76	coffee cup (taza de café)		183,432	4.95	3.03	41.70	17.96	0.67	0.19	0.60
77	compass (brújula)		127,609	5.97	3.33	43.61	15.82	0.91	0.16	0.86
78	computer (computadora)		123,885	6.26	4.40	36.61	15.20	0.86	0.18	0.75
79	cookie (galleta)		131,353	7.96	5.60	54.11	19.76	0.84	0.17	0.83
80	cooler (enfriador)		271,444	5.47	4.34	35.95	25.69	0.60	0.21	0.47
81	couch (sofá)		119,662	5.70	3.04	38.53	16.33	0.88	0.18	0.84
82	cow (vaca)		123,083	10.53	4.99	74.57	30.43	0.83	0.21	0.76
83	crab (cangrejo)		126,930	9.14	4.02	55.48	19.31	0.87	0.17	0.82
84	crayon (crayón)		129,953	4.50	3.26	25.88	19.31	0.64	0.21	0.54
85	crocodile (cocodrilo)		127,932	5.28	4.43	39.83	20.93	0.83	0.20	0.77
86	crown (corona)		134,089	2.33	2.43	30.34	15.17	0.93	0.13	0.91
87	cruise ship (crucero)		123,410	5.54	4.06	32.81	20.34	0.83	0.18	0.77
88	cup (taza)		130,721	3.45	2.12	33.10	15.01	0.62	0.19	0.49
89	diamond (diamante)		131,587	3.74	3.21	21.35	12.61	0.88	0.16	0.85
90	dishwasher (lavavajillas)		169,759	6.08	4.94	37.69	27.80	0.73	0.22	0.63
91	diving board (trampolín)		290,239	4.67	3.67	23.01	20.87	0.74	0.23	0.60
92	dog (perro)		152,159	7.18	3.92	54.25	19.47	0.65	0.25	0.54
93	dolphin (delfín)		121,613	3.30	2.79	32.42	15.39	0.82	0.20	0.73
94	donut (dona)		140,751	4.07	3.83	43.15	13.20	0.91	0.15	0.90
95	door (puerta)		120,230	2.68	1.72	21.66	12.36	0.90	0.18	0.81
96	dragon (dragón)		124,362	6.95	4.14	56.31	20.70	0.73	0.23	0.66
97	dresser (cómoda)		123,395	7.20	3.75	38.42	22.56	0.88	0.19	0.80
98	drill (taladro)		136,786	4.26	3.58	32.99	24.21	0.87	0.20	0.77

	Categoría		N. dibujos	N. Trazos		N. Segmentos		Índice Confianza		F-1
				μ	σ	μ	σ	μ	σ	
99	drums (tambores)		137,299	7.40	3.80	48.38	21.19	0.88	0.20	0.82
100	duck (pato)		135,480	6.22	3.78	44.79	17.77	0.72	0.19	0.64
101	dumbbell (mancuerna)		157,975	4.04	3.41	35.33	29.32	0.89	0.18	0.82
102	ear (oreja)		122,897	2.44	1.98	28.01	14.57	0.90	0.18	0.84
103	elbow (codo)		126,253	3.03	2.57	23.31	18.41	0.79	0.21	0.72
104	elephant (elefante)		126,969	5.88	3.81	53.58	18.89	0.82	0.20	0.78
105	envelope (sobre)		134,863	2.56	2.06	22.87	15.06	0.91	0.11	0.92
106	eraser (borrador)		118,339	4.66	3.77	28.39	24.40	0.71	0.24	0.62
107	eye (ojo)		125,888	5.32	2.95	47.84	25.45	0.94	0.14	0.92
108	eyeglasses (gafas)		225,762	4.79	2.44	33.83	16.29	0.91	0.16	0.86
109	face (rostro)		161,666	6.58	2.95	51.89	20.68	0.79	0.18	0.75
110	fan (ventilador)		136,158	7.68	4.45	63.40	22.63	0.87	0.18	0.78
111	feather (pluma)		119,910	5.72	5.27	29.33	20.29	0.79	0.20	0.73
112	fence (cerca)		129,426	5.92	3.64	26.24	15.77	0.83	0.19	0.77
113	finger (dedo)		167,957	2.57	2.52	22.27	16.25	0.81	0.19	0.73
114	fire hydrant (hidrante)		137,242	5.77	4.28	41.50	25.76	0.82	0.22	0.74
115	fireplace (chimenea)		155,570	6.46	4.25	52.60	25.18	0.90	0.17	0.84
116	firetruck (camión de bomberos)		220,695	8.39	4.37	50.47	21.15	0.81	0.21	0.70
117	fish (pez)		134,150	3.39	2.51	29.91	11.31	0.90	0.16	0.87
118	flamingo (flamenco)		124,569	5.53	3.09	36.42	15.72	0.86	0.18	0.80
119	flashlight (linterna)		239,763	7.37	3.13	29.67	14.82	0.87	0.18	0.84
120	flip flops (sandalias)		121,518	4.43	2.40	37.05	17.25	0.88	0.19	0.84
121	floor lamp (lámpara de pie)		166,355	4.55	3.60	26.26	20.29	0.86	0.19	0.78
122	flower (flor)		144,818	4.00	2.66	57.63	17.14	0.92	0.15	0.89
123	flying saucer (platillo volador)		151,966	5.00	4.41	36.56	27.15	0.83	0.20	0.74

				N. Trazos		N. Segmentos		Índice Confianza		
	Categoría		N. dibujos	μ	σ	μ	σ	μ	σ	F-1
124	foot (pie)		203,086	2.36	2.89	30.22	15.89	0.81	0.19	0.73
125	fork (tenedor)		126,077	2.88	2.42	22.24	13.72	0.90	0.17	0.84
126	frog (rana)		159,047	6.81	3.95	55.82	22.73	0.70	0.25	0.59
127	frying pan (sartén)		123,824	3.13	2.15	31.61	16.26	0.86	0.19	0.82
128	garden hose (manguera de jardín)		121,843	8.53	5.50	59.91	29.58	0.64	0.23	0.48
129	garden (jardín)		158,527	3.68	3.88	37.37	27.33	0.86	0.20	0.81
130	giraffe (jirafa)		127,182	5.19	4.40	39.08	14.25	0.90	0.16	0.88
131	goatee (barba de chivo)		190,002	7.13	5.19	64.47	31.63	0.86	0.18	0.78
132	golf club (palito de golf)		194,843	2.72	3.48	21.89	19.45	0.74	0.19	0.65
133	grapes (uvas)		155,305	8.38	3.74	80.69	30.49	0.81	0.17	0.79
134	grass (césped)		123,071	5.05	5.07	29.54	17.71	0.86	0.17	0.83
135	guitar (guitarra)		120,451	6.91	3.50	47.19	14.99	0.72	0.17	0.68
136	hamburger (hamburguesa)		129,672	5.22	4.01	45.42	17.59	0.88	0.16	0.85
137	hammer (martillo)		119,012	3.33	2.35	30.86	14.69	0.82	0.20	0.78
138	hand (mano)		291,773	1.96	2.70	41.01	12.92	0.89	0.14	0.86
139	harp (arpa)		285,403	6.31	3.04	36.06	20.85	0.91	0.16	0.87
140	hat (sombrero)		222,610	2.84	2.44	25.73	16.54	0.84	0.20	0.78
141	headphones (audífonos)		118,906	3.42	2.04	41.17	17.73	0.96	0.11	0.93
142	hedgehog (erizo)		120,527	14.10	9.03	67.30	32.12	0.90	0.17	0.85
143	helicopter (helicóptero)		159,938	8.04	3.61	45.76	20.91	0.92	0.16	0.89
144	helmet (casco)		121,899	3.81	2.85	35.04	17.57	0.82	0.20	0.76
145	hexagon (hexágono)		142,435	2.18	2.81	16.82	17.61	0.77	0.16	0.71
146	hockey puck (disco de hockey)		203,301	3.24	2.71	44.84	53.27	0.77	0.22	0.69
147	hockey stick (palo de hockey)		130,110	1.90	2.16	21.29	15.43	0.69	0.17	0.66
148	horse (caballo)		178,286	7.02	5.27	52.55	21.39	0.79	0.20	0.75

	Categoría		N. dibujos	N. Trazos		N. Segmentos		Índice Confianza		F-1
				μ	σ	μ	σ	μ	σ	
149	hospital (hospital)		167,448	6.55	4.38	36.78	18.60	0.88	0.17	0.84
150	hot air balloon (globo aerostático)		126,350	5.21	2.65	33.21	12.09	0.91	0.15	0.87
151	hot dog (perro caliente)		181,999	3.77	2.44	37.96	17.47	0.85	0.19	0.79
152	hot tub (bañera de hidromasaje)		120,279	7.62	5.28	48.49	28.31	0.79	0.23	0.65
153	hourglass (reloj de arena)		135,957	3.25	3.21	28.77	19.64	0.92	0.14	0.89
154	house plant (planta de interior)		122,996	3.72	2.63	24.40	10.59	0.92	0.15	0.88
155	house (casa)		135,420	5.55	3.20	45.78	19.96	0.84	0.16	0.82
156	hurricane (huracán)		136,245	2.80	4.52	95.74	48.74	0.66	0.19	0.56
157	ice cream (helado)		123,133	3.86	2.86	32.06	13.96	0.93	0.14	0.91
158	jacket (chaqueta)		214,124	4.45	3.47	35.51	18.16	0.84	0.18	0.78
159	jail (cárcel)		120,131	6.88	3.36	25.66	15.53	0.84	0.17	0.81
160	kangaroo (canguro)		174,470	6.15	4.24	51.37	19.97	0.85	0.20	0.80
161	key (llave)		160,965	3.24	2.49	37.41	14.32	0.89	0.19	0.84
162	keyboard (teclado)		187,766	4.35	5.56	22.63	21.63	0.68	0.22	0.61
163	knee (rodilla)		267,540	2.79	2.69	19.19	17.34	0.78	0.22	0.70
164	knife (cuchillo)		172,656	3.16	2.56	21.55	15.18	0.77	0.22	0.66
165	ladder (escalera)		125,389	5.96	1.99	15.57	11.79	0.95	0.12	0.93
166	lantern (linterna)		149,912	7.39	4.14	42.84	20.93	0.82	0.22	0.72
167	laptop (computadora portátil)		261,501	6.71	5.05	34.17	18.82	0.76	0.17	0.76
168	leaf (hoja)		125,571	5.45	3.30	29.89	14.05	0.84	0.17	0.79
169	leg (pierna)		116,804	2.17	2.26	17.83	14.00	0.66	0.19	0.59
170	light bulb (bombilla)		120,879	4.29	3.37	33.87	15.13	0.86	0.17	0.83
171	lighter (encendedor)		121,260	4.29	2.57	31.39	13.35	0.88	0.17	0.83
172	lighthouse (faro)		160,903	8.82	4.08	34.72	19.64	0.87	0.19	0.82
173	lightning (relámpago)		151,560	2.20	2.82	22.76	18.93	0.87	0.18	0.82

	Categoría		N. dibujos	N. Trazos		N. Segmentos		Índice Confianza		F-1
				μ	σ	μ	σ	μ	σ	
174	line (línea)		143,549	1.28	1.48	4.67	10.41	0.71	0.20	0.72
175	lion (león)		120,949	7.37	5.01	78.77	28.84	0.90	0.17	0.86
176	lipstick (labial)		127,623	3.14	2.18	21.72	13.13	0.78	0.20	0.72
177	lobster (langosta)		140,175	8.56	4.99	57.67	25.13	0.73	0.22	0.64
178	lollipop (paleta)		128,849	2.57	1.89	31.14	19.76	0.89	0.15	0.86
179	mailbox (buzón)		130,053	4.28	2.40	25.38	12.28	0.86	0.18	0.80
180	map (mapa)		120,629	7.46	5.05	42.19	18.50	0.87	0.18	0.83
181	marker (marcador)		319,136	3.53	3.14	24.50	19.87	0.50	0.17	0.38
182	matches (fósforos)		143,969	4.76	4.55	31.55	27.37	0.79	0.23	0.66
183	megaphone (megáfono)		137,334	5.01	3.78	32.58	19.36	0.85	0.19	0.79
184	mermaid (sirena)		180,304	8.86	4.24	57.77	18.97	0.92	0.16	0.89
185	microphone (micrófono)		120,570	8.32	5.13	39.70	20.43	0.85	0.19	0.79
186	microwave (microondas)		130,533	5.60	4.11	33.10	16.11	0.85	0.18	0.82
187	monkey (mono)		127,633	9.68	4.01	70.46	22.56	0.79	0.21	0.72
188	moon (luna)		121,661	2.84	4.14	33.49	25.30	0.84	0.20	0.72
189	mosquito (mosquito)		123,029	7.25	3.68	46.07	21.91	0.76	0.24	0.65
190	motorbike (motocicleta)		169,931	6.74	4.07	53.94	25.76	0.76	0.18	0.70
191	mountain (montaña)		128,540	2.10	2.15	16.95	13.93	0.90	0.16	0.84
192	mouse (ratón)		178,826	5.47	3.45	35.84	18.27	0.74	0.24	0.61
193	moustache (bigote)		179,924	2.64	3.53	31.75	27.21	0.87	0.20	0.79
194	mouth (boca)		134,135	4.43	3.37	32.84	16.57	0.88	0.18	0.84
195	mug (taza)		152,918	3.72	2.34	36.95	17.45	0.60	0.12	0.62
196	mushroom (seta)		142,167	3.26	3.09	30.96	15.23	0.84	0.17	0.81
197	nail (uña/clavo)		158,593	3.36	2.88	25.65	21.80	0.84	0.22	0.70
198	necklace (collar)		120,580	4.65	4.24	42.96	33.26	0.85	0.19	0.77

	Categoría		N. dibujos	N. Trazos		N. Segmentos		Índice Confianza		F-1
				μ	σ	μ	σ	μ	σ	
199	nose (nariz)		197,573	2.31	2.53	21.85	16.84	0.85	0.20	0.76
200	ocean (océano)		131,493	2.86	3.22	23.82	20.61	0.76	0.19	0.70
201	octagon (octágono)		159,474	2.82	3.13	20.10	21.16	0.74	0.16	0.70
202	octopus (pulpo)		150,152	4.16	3.68	50.53	17.16	0.94	0.14	0.91
203	onion (cebolla)		132,297	6.29	3.83	49.45	26.28	0.87	0.19	0.79
204	oven (horno)		206,910	5.25	3.69	38.03	21.74	0.68	0.20	0.54
205	owl (búho)		169,632	9.13	3.91	61.49	20.47	0.85	0.19	0.80
206	paint can (lata de pintura)		123,446	4.73	3.31	41.66	21.03	0.71	0.21	0.64
207	paintbrush (pincel)		187,002	8.27	5.62	44.34	22.41	0.81	0.21	0.72
208	palm tree (palmera)		121,959	5.23	4.29	43.81	19.82	0.91	0.15	0.87
209	panda (panda)		113,613	7.85	3.88	95.11	40.44	0.82	0.19	0.75
210	pants (pantalones)		144,264	2.05	2.18	24.64	12.04	0.83	0.16	0.77
211	paper clip (clip)		127,129	1.47	1.68	27.52	12.84	0.91	0.17	0.86
212	parachute (paracaídas)		127,319	6.97	3.68	37.37	18.28	0.93	0.14	0.89
213	parrot (loro)		185,530	7.28	4.17	45.44	20.08	0.72	0.20	0.65
214	passport (pasaporte)		150,265	7.28	4.53	42.08	21.12	0.80	0.21	0.73
215	peanut (cacahuete)		126,626	2.83	3.68	30.10	13.34	0.84	0.18	0.78
216	pear (pera)		116,904	2.13	1.77	26.79	11.94	0.88	0.17	0.83
217	peas (guisantes)		161,656	7.65	6.40	63.80	32.18	0.79	0.19	0.77
218	pencil (lápiz)		122,001	4.85	2.92	23.13	13.49	0.70	0.21	0.65
219	penguin (pingüino)		253,791	6.78	3.27	50.09	18.89	0.85	0.18	0.81
220	piano (piano)		116,870	8.42	4.05	33.17	16.51	0.80	0.20	0.74
221	pickup truck (camioneta)		130,740	5.08	3.83	42.18	23.09	0.68	0.19	0.63
222	picture frame (marco de fotos)		122,371	3.51	2.65	35.07	16.41	0.80	0.17	0.81
223	pig (cerdo)		186,770	9.56	3.47	63.60	18.10	0.89	0.18	0.84

	Categoría		N. dibujos	N. Trazos		N. Segmentos		Índice Confianza		F-1
				μ	σ	μ	σ	μ	σ	
224	pillow (almohada)		118,753	2.45	2.65	23.04	14.51	0.74	0.24	0.62
225	pineapple (piña)		125,071	5.84	4.68	46.77	16.71	0.91	0.14	0.89
226	pizza (pizza)		130,371	8.76	4.48	58.16	20.83	0.91	0.17	0.85
227	pliers (alicates)		172,549	4.08	3.59	36.44	28.53	0.75	0.24	0.60
228	police car (coche de policía)		130,024	6.00	3.91	46.50	18.24	0.83	0.19	0.74
229	pond (estanque)		121,620	3.66	3.76	35.60	23.00	0.65	0.21	0.59
230	pool (piscina)		133,439	4.93	3.73	36.79	22.08	0.68	0.22	0.52
231	popsicle (paleta helada)		126,707	2.79	2.40	24.72	17.16	0.89	0.17	0.83
232	postcard (tarjeta postal)		125,706	4.47	2.95	26.99	14.89	0.85	0.18	0.72
233	potato (papa/patata)		329,204	3.95	4.63	29.20	19.36	0.70	0.22	0.59
234	power outlet (tomacorriente)		169,462	5.97	3.37	40.56	23.26	0.87	0.19	0.80
235	purse (bolso)		123,320	3.80	2.74	35.91	17.98	0.79	0.19	0.73
236	rabbit (conejo)		155,288	7.71	3.66	57.40	20.20	0.86	0.19	0.81
237	raccoon (mapache)		119,588	9.73	5.21	75.10	37.83	0.70	0.22	0.58
238	radio (radio)		135,728	6.94	4.62	42.76	19.47	0.79	0.18	0.73
239	rain (lluvia)		134,680	9.22	5.32	35.00	23.24	0.93	0.15	0.90
240	rainbow (arcoíris)		126,845	2.59	1.95	25.03	18.22	0.92	0.13	0.91
241	rake (rastrillo)		154,639	6.51	4.05	23.86	24.25	0.80	0.18	0.75
242	remote control (mando a distancia)		119,644	7.02	4.61	36.49	19.84	0.88	0.18	0.83
243	rhinoceros (rinoceronte)		188,484	5.24	3.88	44.38	22.28	0.82	0.21	0.76
244	rifle (rifle/fusil)		172,444	3.43	3.68	21.16	23.27	0.75	0.23	0.68
245	river (río)		133,271	3.80	3.13	24.36	17.44	0.76	0.19	0.70
246	roller coaster (montaña rusa)		143,570	6.58	5.64	36.74	21.58	0.84	0.20	0.78
247	rollerskates (patines de ruedas)		119,772	5.10	2.50	50.89	16.46	0.93	0.14	0.90
248	sailboat (velero)		136,506	3.52	1.92	25.12	11.65	0.90	0.15	0.89

	Categoría		N. dibujos	N. Trazos		N. Segmentos		Índice Confianza		F-1
				μ	σ	μ	σ	μ	σ	
249	sandwich (sándwich)		131,732	5.16	3.54	47.49	21.64	0.83	0.20	0.75
250	saw (sierra)		121,256	2.80	2.56	35.76	16.81	0.91	0.14	0.88
251	saxophone (saxofón)		118,107	3.86	3.68	39.54	18.65	0.89	0.17	0.85
252	school bus (autobús escolar)		122,041	7.95	4.38	55.91	21.24	0.62	0.18	0.56
253	scissors (tijeras)		123,390	4.77	2.53	44.05	18.88	0.87	0.18	0.84
254	scorpion (escorpión)		165,689	8.71	4.42	57.66	22.57	0.87	0.18	0.81
255	screwdriver (destornillador)		116,313	4.39	3.27	26.74	21.41	0.78	0.21	0.71
256	sea turtle (tortuga marina)		119,876	8.09	3.87	53.71	18.08	0.89	0.18	0.85
257	see saw (balancín)		131,936	3.98	3.11	20.97	20.44	0.90	0.17	0.83
258	shark (tiburón)		126,050	3.99	2.85	32.10	12.87	0.81	0.18	0.76
259	sheep (oveja)		126,121	6.12	3.56	66.43	33.54	0.92	0.15	0.89
260	shoe (zapato)		120,231	3.10	2.61	27.87	12.68	0.84	0.20	0.78
261	shorts (pantalones cortos)		124,970	2.46	2.20	25.06	10.33	0.75	0.15	0.75
262	shovel (pala)		117,194	3.30	2.16	24.36	13.92	0.81	0.20	0.74
263	sink (lavabo/fregadero)		208,410	7.23	3.90	47.89	19.13	0.89	0.18	0.85
264	skateboard (patineta)		128,733	3.73	1.89	32.91	12.34	0.94	0.13	0.91
265	skull (calavera)		126,174	5.91	2.87	55.97	20.46	0.94	0.13	0.91
266	skyscraper (rascacielos)		183,709	3.89	4.42	20.01	24.85	0.78	0.21	0.69
267	sleeping bag (saco de dormir)		119,691	5.04	3.72	36.31	20.10	0.78	0.23	0.69
268	smiley face (cara sonriente)		124,386	4.70	2.01	38.02	17.46	0.85	0.16	0.82
269	snail (caracol)		133,757	3.80	2.41	51.94	16.73	0.93	0.15	0.90
270	snake (serpiente)		122,273	2.97	2.72	31.02	12.87	0.80	0.20	0.74
271	snorkel (tubo de snorkel)		154,533	4.18	3.29	44.80	24.77	0.89	0.19	0.82
272	snowflake (copo de nieve)		116,685	6.56	4.92	26.83	21.85	0.90	0.17	0.88
273	snowman (muñeco de nieve)		340,029	4.01	2.80	43.38	12.65	0.94	0.13	0.92

	Categoría		N. dibujos	N. Trazos		N. Segmentos		Índice Confianza		F-1
				μ	σ	μ	σ	μ	σ	
274	soccer ball (balón de fútbol)		125,349	7.28	3.13	72.15	39.78	0.90	0.16	0.86
275	sock (calcetín)		205,715	2.83	2.67	27.93	14.77	0.86	0.18	0.82
276	speedboat (lancha rápida)		188,580	6.19	3.56	34.03	18.31	0.81	0.21	0.75
277	spider (araña)		209,447	9.21	3.36	46.32	25.54	0.88	0.17	0.84
278	spoon (cuchara)		125,028	1.89	1.73	23.47	13.21	0.80	0.20	0.74
279	spreadsheet (hoja de cálculo)		170,200	8.32	4.28	35.36	29.89	0.78	0.19	0.73
280	square (cuadrado)		125,145	1.61	2.17	16.30	13.45	0.70	0.17	0.71
281	squiggle (garabato)		118,441	1.86	3.15	34.49	43.97	0.65	0.22	0.56
282	squirrel (ardilla)		156,883	7.39	4.15	63.04	25.19	0.87	0.19	0.81
283	stairs (escaleras)		128,981	1.77	2.48	15.03	12.88	0.96	0.12	0.92
284	star (estrella)		137,619	1.60	1.98	24.11	10.35	0.95	0.12	0.93
285	steak (filete)		122,042	4.41	4.08	40.85	20.88	0.72	0.23	0.63
286	stereo (estéreo)		122,444	7.00	4.70	49.68	24.45	0.78	0.18	0.72
287	stethoscope (estetoscopio)		153,794	4.90	3.47	40.35	28.42	0.91	0.17	0.82
288	stitches (puntos de sutura)		125,192	6.94	4.23	19.66	20.02	0.89	0.19	0.81
289	stop sign (señal de alto)		119,814	7.16	2.80	39.15	13.10	0.97	0.10	0.94
290	stove (estufa)		116,535	9.10	4.29	65.64	26.19	0.83	0.18	0.78
291	strawberry (fresa)		122,301	8.79	6.46	43.82	15.22	0.91	0.15	0.88
292	streetlight (farola)		123,280	3.87	3.32	22.11	14.40	0.86	0.19	0.78
293	string bean (judía verde)		119,083	3.27	4.12	28.93	33.44	0.67	0.23	0.53
294	submarine (submarino)		124,362	4.30	3.05	36.82	20.00	0.88	0.18	0.83
295	suitcase (maleta)		126,442	2.74	2.20	22.84	14.18	0.83	0.21	0.72
296	sun (sol)		133,781	8.10	3.24	34.58	14.99	0.93	0.13	0.92
297	swan (cisne)		152,088	4.05	3.84	39.89	20.95	0.81	0.19	0.75
298	sweater (suéter)		120,184	2.62	2.92	31.18	13.57	0.76	0.15	0.72

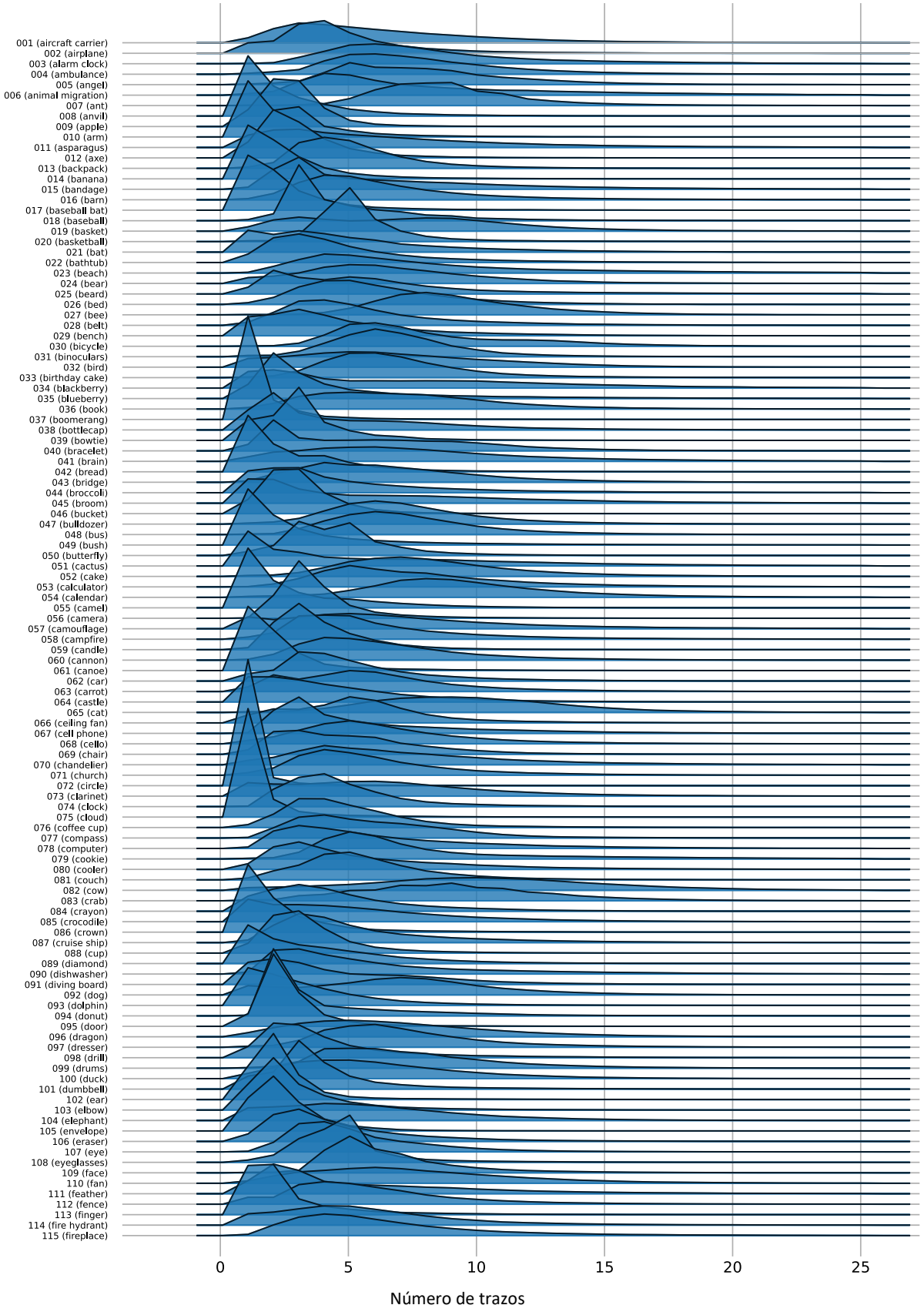
				N. Trazos		N. Segmentos		Índice Confianza		
Categoría			N. dibujos	μ	σ	μ	σ	μ	σ	F-1
299	swing set (conjunto de columpios)		119,357	4.71	2.25	19.15	13.82	0.92	0.15	0.89
300	sword (espada)		123,802	3.78	2.36	24.08	13.69	0.88	0.18	0.82
301	syringe (jeringa)		135,823	6.28	3.87	28.11	29.96	0.87	0.19	0.80
302	t-shirt (camiseta)		125,233	2.02	2.08	26.88	9.05	0.86	0.15	0.83
303	table (mesa)		128,021	4.71	2.29	20.14	11.42	0.80	0.18	0.78
304	teapot (tetera)		126,804	4.43	2.35	41.74	13.78	0.89	0.18	0.84
305	teddy-bear (oso de peluche)		179,568	8.78	3.93	71.05	19.64	0.79	0.18	0.77
306	telephone (teléfono)		127,885	6.03	5.06	45.13	20.33	0.84	0.22	0.68
307	television (televisión)		123,137	5.54	2.91	34.53	14.15	0.93	0.13	0.89
308	tennis racquet (raqueta de tenis)		231,151	8.47	3.59	43.39	18.82	0.90	0.15	0.87
309	tent (tienda de campaña)		131,527	3.69	2.20	19.33	12.79	0.88	0.17	0.83
310	The Eiffel Tower (La Torre Eiffel)		134,801	4.93	3.50	23.34	14.51	0.91	0.15	0.89
311	The Great Wall of China (La Gran Muralla China)		193,015	6.03	5.80	34.21	38.68	0.73	0.24	0.58
312	The Mona Lisa (La Mona Lisa)		121,383	8.79	4.22	56.04	23.06	0.92	0.15	0.90
313	tiger (tigre)		121,067	12.42	5.69	65.48	27.00	0.74	0.20	0.63
314	toaster (tostadora)		122,434	5.91	3.36	38.87	17.37	0.83	0.21	0.76
315	toe (dedo del pie)		153,652	3.66	3.34	34.99	19.21	0.81	0.19	0.73
316	toilet (inodoro)		129,888	4.40	2.24	39.84	12.80	0.88	0.18	0.84
317	tooth (diente)		125,064	2.15	2.61	30.78	13.98	0.88	0.18	0.81
318	toothbrush (cepillo de dientes)		124,828	6.64	5.52	26.01	17.86	0.88	0.18	0.82
319	toothpaste (pasta de dientes)		131,037	4.83	4.29	27.40	16.91	0.69	0.23	0.59
320	tornado (tornado)		143,271	2.06	3.06	91.77	52.18	0.73	0.14	0.75
321	tractor (tractor)		116,677	5.77	3.42	53.23	21.83	0.83	0.19	0.75
322	traffic light (semáforo)		125,321	5.06	2.13	42.78	14.11	0.92	0.13	0.91
323	train (tren)		127,948	10.32	4.71	56.46	24.29	0.86	0.19	0.80

	Categoría		N. dibujos	N. Trazos		N. Segmentos		Índice Confianza		F-1
				μ	σ	μ	σ	μ	σ	
324	tree (árbol)		144,721	3.86	3.05	54.77	36.58	0.86	0.18	0.80
325	triangle (triángulo)		123,170	1.52	1.64	14.28	11.20	0.92	0.13	0.91
326	trombone (trombón)		184,759	5.22	3.69	40.92	25.53	0.74	0.24	0.59
327	truck (camión)		131,354	5.41	2.73	44.65	16.03	0.59	0.16	0.51
328	trumpet (trompeta)		169,547	5.54	3.95	35.15	21.36	0.71	0.19	0.66
329	umbrella (paraguas)		124,084	3.56	2.56	30.76	12.63	0.93	0.13	0.91
330	underwear (ropa interior)		124,548	2.63	2.32	23.62	12.90	0.86	0.19	0.75
331	van (furgoneta)		165,909	5.28	3.41	46.26	20.28	0.64	0.20	0.56
332	vase (jarrón)		126,475	2.78	2.26	31.89	14.43	0.88	0.17	0.83
333	violin (violín)		217,260	5.26	3.69	42.11	17.63	0.68	0.17	0.60
334	washing machine (lavadora)		120,851	4.53	2.81	44.35	18.26	0.85	0.16	0.85
335	watermelon (sandía)		132,939	6.69	5.73	37.31	19.24	0.79	0.22	0.70
336	waterslide (tobogán acuático)		185,364	5.42	4.57	30.87	22.80	0.82	0.21	0.76
337	whale (ballena)		116,502	5.55	3.45	40.83	15.97	0.86	0.19	0.80
338	wheel (rueda)		136,659	6.90	2.76	49.41	18.04	0.84	0.18	0.84
339	windmill (molino de viento)		120,644	5.97	3.33	42.36	18.42	0.89	0.17	0.86
340	wine bottle (botella de vino)		126,373	2.48	2.30	23.02	11.85	0.89	0.17	0.85
341	wine glass (copa de vino)		132,302	3.37	2.22	29.28	15.45	0.93	0.13	0.89
342	wristwatch (reloj de pulsera)		162,645	7.98	4.08	39.64	20.85	0.89	0.18	0.84
343	yoga (yoga)		280,442	6.19	3.08	37.49	22.50	0.79	0.24	0.69
344	zebra (cebra)		144,608	10.29	5.02	57.62	21.48	0.84	0.16	0.80
345	zigzag (zigzag)		120,072	1.30	1.66	18.01	15.65	0.84	0.16	0.79
		Total:	50,426,266							
		Media:	146,163	5.28	3.54	39.36	20.16	0.82	0.18	0.77

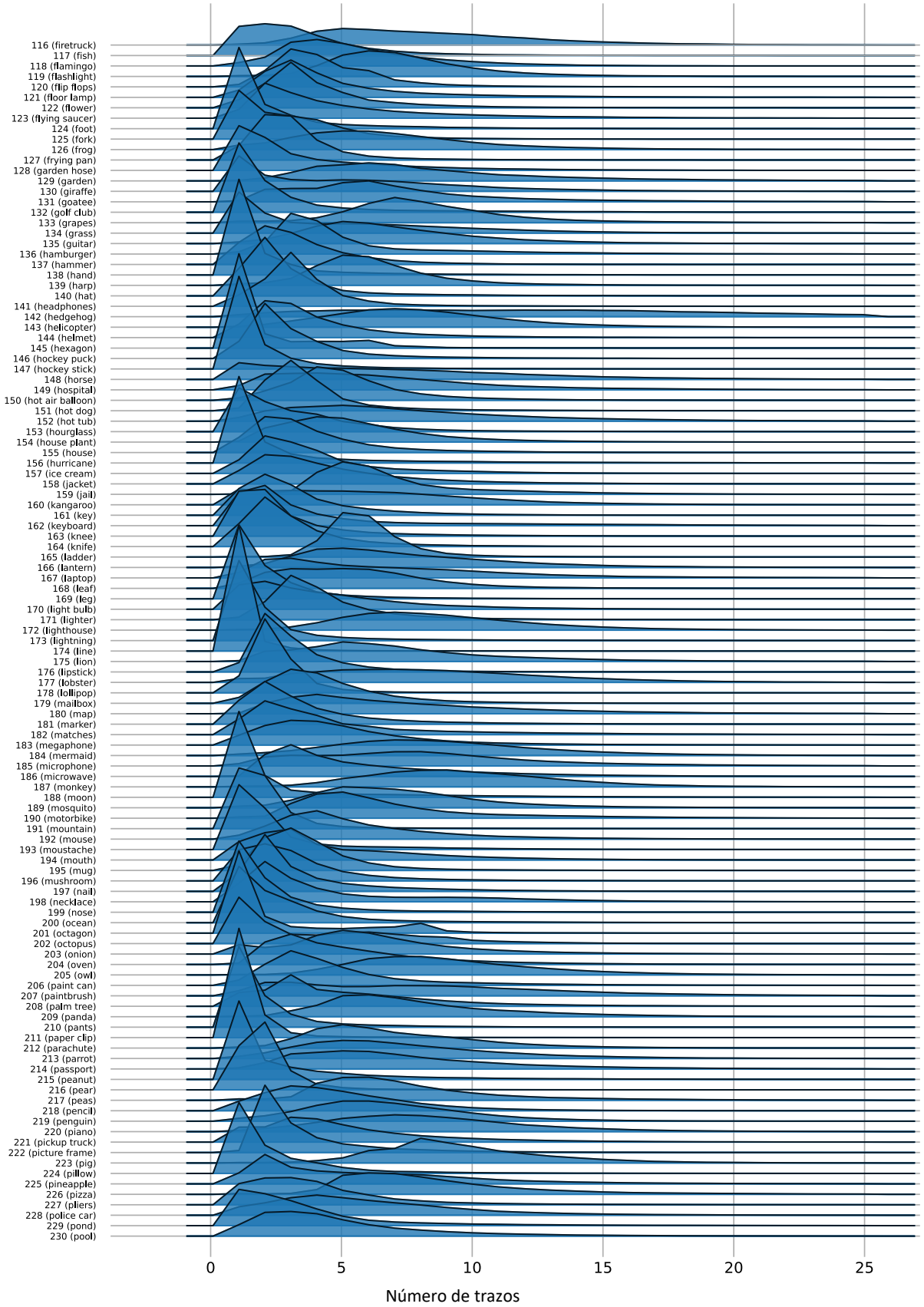
9.2 Anexo 2. Mapas de bits del primer registro de cada categoría de *Quick, Draw!*

9.3 Anexo 3. Distribución del número de trazos en cada categoría de *Quick, Draw!*

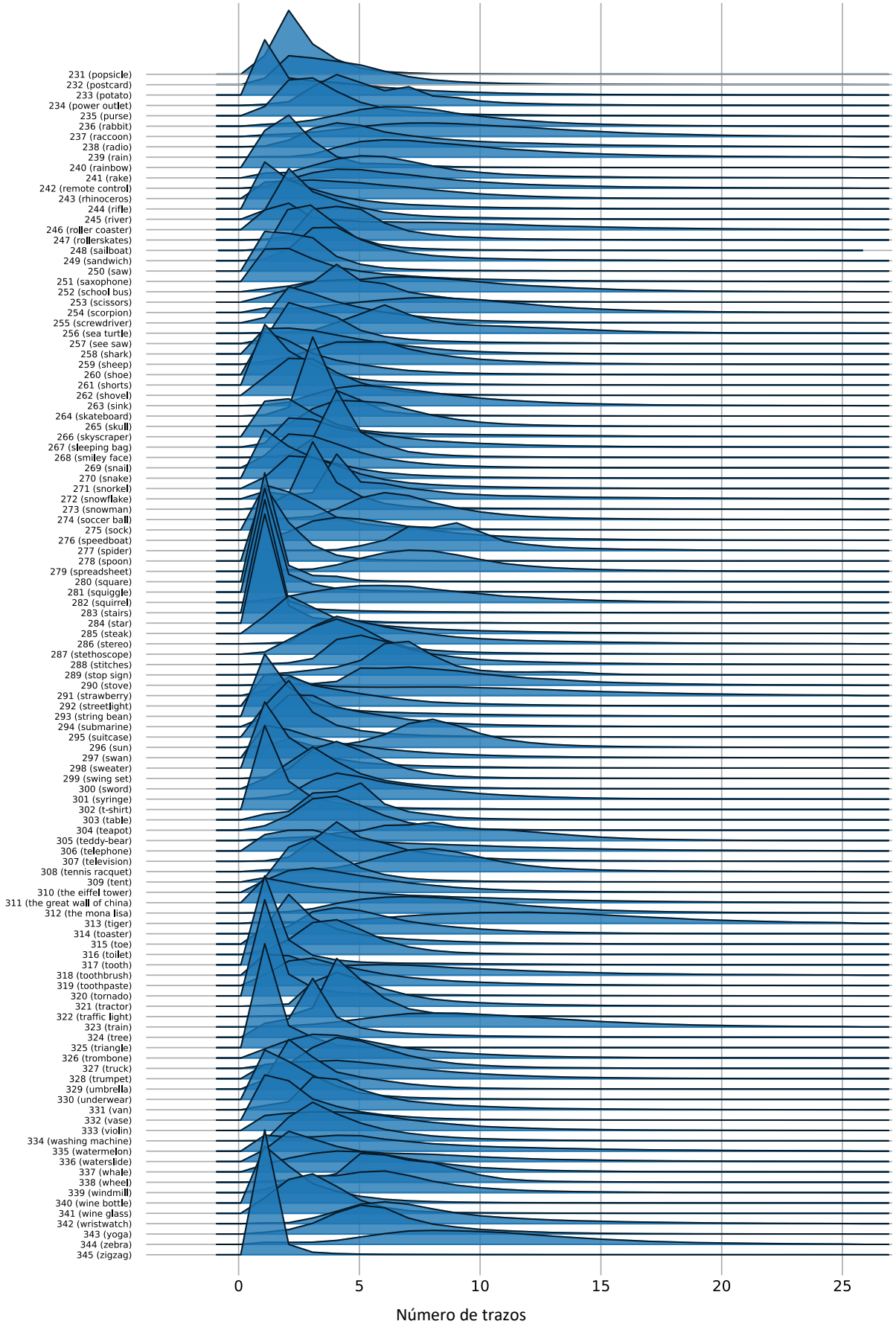
9.3.1 Categorías 1 a 115



9.3.2 Categorías 116 a 230

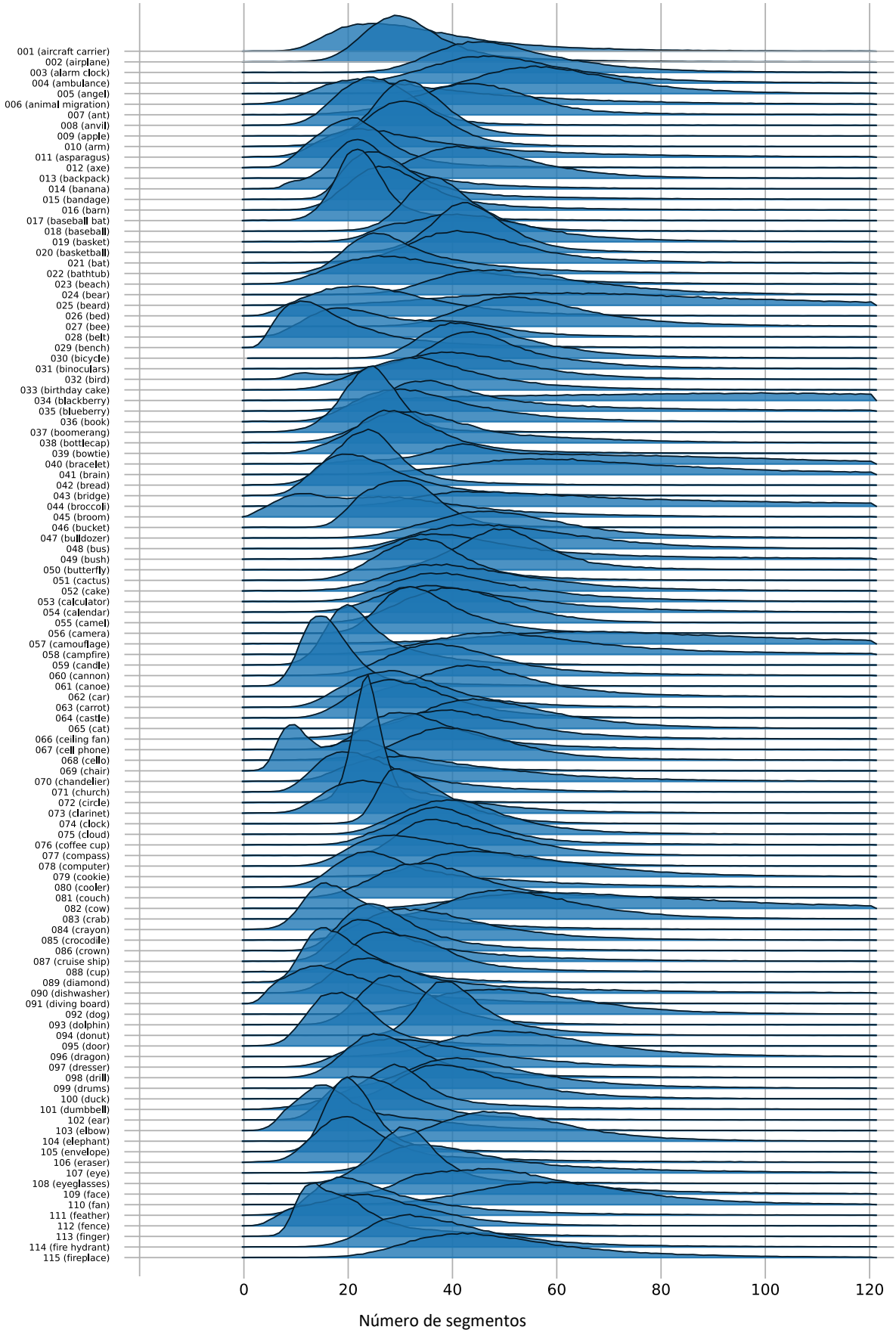


9.3.2 Categorías 231 a 345

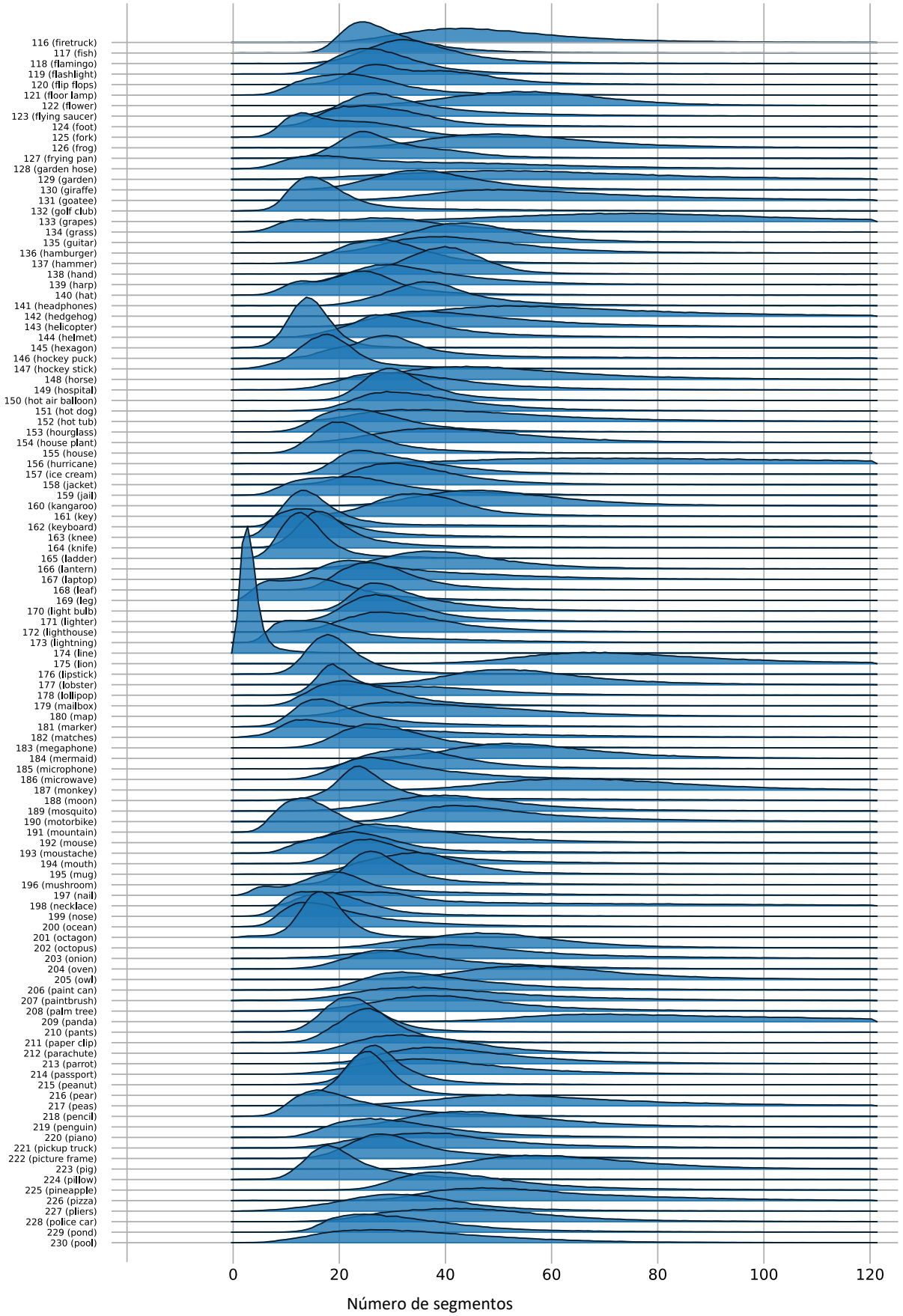


9.4 Anexo 4. Distribución del número de segmentos en cada categoría de *Quick, Draw!*

9.4.1 Categorías 1 a 115



9.4.2 Categorías 116 a 230



9.4.3 Categorías 231 a 345