Universidad Nacional Autónoma de México
Posgrado en Ciencia e Ingeniería de la Computación

Mezcla visual entre dibujos usando autocodificadores variacionales

TESIS

Que para optar por el grado de:
Maestro en Ciencia e Ingeniería de la Computación

PRESENTA:
Derek Cheung

DIRECTOR DE TESIS:
Dr. Gibran Fuentes Pineda
IIMAS-UNAM

Ciudad Universitaria, CD. MX. Diciembre 2018

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Resumen

La capacidad de mezclar conceptos visuales es una habilidad humana que se ha manifestado a lo largo de la historia de las artes visuales. Sin embargo, las técnicas existentes que realizan esta compleja tarea requieren anotaciones manuales que pueden ser muy laboriosas y están limitadas en el tipo de mezclas que pueden producir. Parte del problema radica en la falta de una técnica cuantitativa automatizada para evaluar a qué nivel una obra visual representa una mezcla. Alentados por avances recientes en redes neuronales profundas para tareas generativas, este trabajo propone nuevas técnicas para la evaluación y generación de mezclas visuales usando redes neuronales recurrentes. Estas técnicas generativas incorporan la técnica evaluativa como guía para remodelar la salida de un autocodificador variacional, un tipo de red neuronal artificial generativa. Resultados experimentales con dibujos, un subdominio simplificado del arte visual, indican que estas técnicas generan dibujos que pueden ser considerados mezclas visuales con mayor frecuencia, en comparación a técnicas que no integran información de un clasificador externo. Lo anterior se valida con una encuesta realizada a las personas sobre la percepción que tienen de los dibujos generados por estas técnicas, en relación a si corresponden a mezclas visuales o no.

Agradecimientos

A Aideé por inspirar el tema de esta tesis, por motivarme y alentarme y por compartir en mis aspiraciones.

A mi tutor Gibrán por su apoyo, orientación y sugerencias indispensables.

A los nuevos amigos que me he encontrado durante mis estudios y mis viajes, quienes me abrieron los ojos.

A la UNAM, el Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas y el Posgrado en Ciencia e Ingeniería de la Computación por darme la oportunidad de cursar este programa de maestría.

A mi país natal, Canadá, por proporcionar la educación necesaria para poder continuar mis estudios en México.

Agradezco la beca recibida por parte del Consejo Nacional de Ciencia y Tecnología.

Índice general

Índice de figuras

Índice de cuadros

Glosario

Abreviaturas por sus siglas en inglés.

CC BY 3.0	Licencia Atribución 3.0 No portada `https://creativecommons.org/licenses/by/3.0/`
CC BY-SA 4.0	Licencia Atribución-CompartirIgual 4.0 Internacional `https://creativecommons.org/licenses/by-sa/4.0/`

GAN	Red generativa antagónica (*Generative Adversarial Network*)
GD	Gradiente descendiente (*Gradient descent*)
LR	Taza de aprendizaje (*Learning rate*)
LSTM	Memoria larga de corto plazo (*Long short term memory*)
MDM	Modelo de mezclada de densidad (*Mixture density model*)
MLP	Perceptrón multicapa (*Multilayer perceptron*)
ReLU	Unidad lineal rectificada (*Rectified linear unit*)
RNN	Red neuronal recurrente (*Recurrent neural network*)
SGD	Gradiente descendiente estocástico (*Stochastic gradient descent*)
VAE	Auto-codificador variacional (*Variational autoencoder*)

Lista de símbolos

x	Un valor escalar
$\boldsymbol{x}$	Un vector
x_i	Elemento al índice i del vector $\boldsymbol{x}$
$x^{(i)}$	El i-ésimo elemento en un conjunto
$\boldsymbol{X}$	Una matriz
$\boldsymbol{X_i}$	Renglón i de la matriz $\boldsymbol{X}$
$f(\boldsymbol{x};\boldsymbol{\theta})$	Una función de $\boldsymbol{x}$ parametrizada por $\boldsymbol{\theta}$
$f(\boldsymbol{x})$	Una versión más corta de $f(\boldsymbol{x};\boldsymbol{\theta})$
μ	La media aritmética
σ	La desviación estándar
$\mathcal{N}(0,1)$	Una distribución gaussiana con $\mu = 0$ y $\sigma = 1$
$U(\text{mín}, \text{máx})$	Una distribución uniforme entre los valores mín y máx.
$\frac{dy}{dx}$	La derivada de y con respecto a x
$\frac{\varphi y}{\varphi x}$	La derivada parcial de y con respecto a x

Capítulo 1

Introducción

Figura 1.1: Artista desconocido. *La Virgen del Cerro*. Siglo XVIII. Casa Nacional de la Moneda, Potosí, Bolivia. Fotografía del usuario *PMRMaeyaert* de Wikipedia, recortada e iluminada para mejorar su claridad (CC BY-SA 4.0).

1.1. Motivación

El estudio de técnicas de Inteligencia Artificial para mezclar dos o más conceptos ha fascinado tanto a investigadores como a espectadores en los últimos años con proyectos de alto perfil. En 2016, la empresa de tecnología IBM introdujo una aplicación que sugiere recetas con nuevas mezclas de ingredientes. En 2018, Google LLC publicó un proyecto que crea sonidos de instrumentos imaginarios mezclados de otros existentes; ambos proyectos recibieron gran aclamación (Kleeman, 2016; Schneider, 2018).

Esta tesis parte de la mezcla visual, que ocupa un lugar especial en la historia de Latinoamérica. Esta región fue el lugar de nacimiento del estilo artístico denominado «mestizo», durante las primeras etapas del periodo colonial. Este estilo mestizo se caracteriza por la integración visual entre conceptos europeos e indígenas, como lo ejemplifica la pintura *La Virgen del Cerro* en la figura 1.1. Como lo describe el historiador de arte Bailey, esta pintura combina una representación del Cerro Rico en Potosí, Bolivia, con la cara de la Virgen María, en una figura híbrida que une el concepto de la Virgen como la protectora de los mineros con la noción andina de un cerro siendo la encarnación de un dios. Según Bailey, este tipo de fusión sincrista reunió las dos culturas en una manera que les permitió mantener sus propias tradiciones mientras sobrevivan en una sociedad colonial (Bailey, 2005). El hibridismo del arte de Latinoamérica es un tema bien discutido, que cada vez se interpreta de una manera más positiva (Lucie-Smith, 2004; Bailey, 2005). El ejemplo de la Virgen del Cerro se encuentra en buena compañía en nuestras culturas compartidas, que están llenas de una zoología silvestre de dragones, sirenas, basiliscos, ninfas,

Figura 1.2: Reproducción de la serpiente emplumada *Quetzalcóatl* como aparece en el Códice Borgia. Imagen del usuario *Eddo* de Wikipedia (CC BY 3.0).

grifos, vampiros, centauros y más. En Mesoamérica existen entre otros el dios de la lluvia Tlaloc y la serpiente emplumada Quetzalcóatl, ambos representados como seres antropomórficos con partes de otras criaturas. De aquí en adelante, el término «mezcla visual» se refiere específicamente a imágenes de este tipo que cuentan con partes visualmente reconocibles de dos o más animales u objetos simultáneamente[1]. Por conveniencia, los términos «categoría de objeto» y «concepto visual» se refieren a uno de estos animales u objetos base.

La motivación de esta tesis también parte de la importancia científica de estudiar este tipo de mezcla visual. El neurólogo Ramachandran ha descrito la capacidad de crear nuevas combinaciones mentales, como ángeles y centauros, como una característica exclusivamente humana, que nos separa de los monos y otros animales

[1]Varios pintores artificiales también ocupan este tipo de mezcla visual, los cuales típicamente usan procesos para descomponer una imagen en varias partes reconocibles para combinarlas posteriormente de diferentes formas (Pereira y Cardoso, 2002; Xiao y Linkola, 2015; Mukaiyama, 2013; Cunha et al., 2017).

(Ramachandran, 2012). La capacidad para tomar dos entradas distintas y crear estructuras nuevas emergentes ha sido señalada como responsable de la invención de herramientas, temas de arte y de ciencia (Mithen, 1996; Fauconnier y Turner, 2008). Una investigación en particular describe «mezcla conceptual» como una operación general de cognición, a la par de analogía y recursión, que involucra la proyección selectiva de estructuras sobre un espacio mental de mezclas, en donde una nueva estructura puede ser elaborada (Fauconnier y Turner, 1998). Estas proyecciones y estructuras han sido traducidos a modelos computacionales que describen dominios visuales pequeños, por ejemplo mezclas entre dibujos de un barco y de una casa (Pereira y Cardoso, 2002, 2006; Ribeiro et al., 2003; Cunha et al., 2017).

Recientemente, las redes neuronales artificiales han demostrado la capacidad de generar mezclas visuales con un enfoque distinto, que involucra un modelo generativo llamado «autocodificador variacional» que produce todo tipo de imagen y un filtro que intenta seleccionar únicamente aquellas imágenes que pueden ser consideradas mezclas visuales (Ha y Eck, 2017; Chen et al., 2017). Estos filtros, que efectivamente remodelan la salida del modelo generativo, no parecen funcionar frecuentemente y tampoco han sido bien estudiados en el contexto de mezclar dos o más categorías de objetos[2]. Este trabajo de investigación se enfoca en el estudio de estas técnicas usando autocodificadores variacionales.

Dibujos

Para delimitar el alcance de esta investigación, se trabaja con pintores artificiales que dibujan bocetos, un sub-dominio simplificado del arte visual. Estos dibujos son

[2]La remodelación se estudió originalmente de forma secundaria como una característica de los modelos generativos como los autocodificadores variacionales o las redes generativas antagónicas.

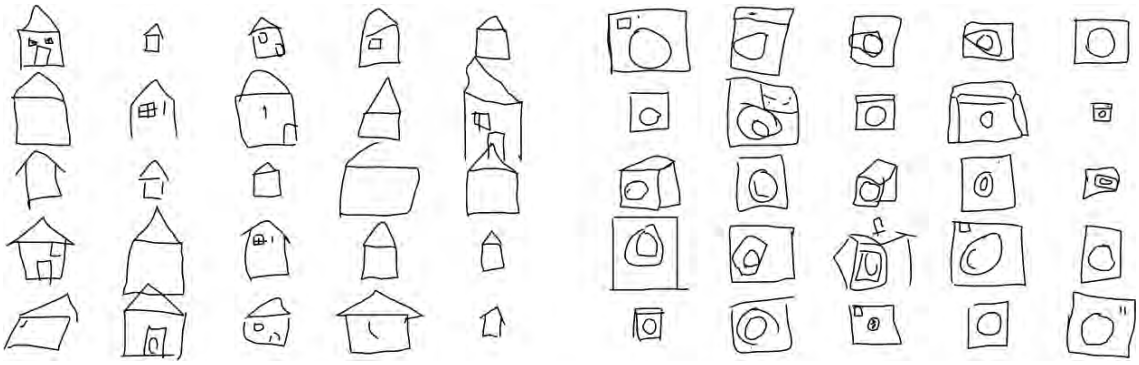


Figura 1.3: Dibujos de casas (lado izquierdo) y lavadoras (lado derecho) hechos por humanos, tomados al azar de la base de datos *Quick, Draw!* (Ha y Eck, 2017).

caracterizados por formas simplificadas que representan algún concepto visual, como los dibujos de casas y lavadoras en la figura 1.3. En términos prácticos, el dominio de los dibujos es simplificado en comparación con el de las pinturas. Los dibujos son representados en blanco y negro, y carecen de sombreados, gradientes y colores. Por lo tanto, pueden ser definidos simplemente como secuencias de puntos, que al conectarse forman las líneas y curvas del dibujo como en un juego de *conecta los puntos*. En esta tesis, el término «dibujo» se refiere a una de estas secuencias de puntos, mientras que el término «imagen» se refiere a una matriz de píxeles. Los dibujos reducen la dimensionalidad de una categoría de objeto de miles de píxeles[3] a menos de cien puntos en una secuencia. Por lo tanto, trabajar con dibujos es menos costoso computacionalmente, más escalable y también evita los problemas de borrosidad que son comúnmente encontrados en sistemas que trabajan con píxeles. En general, el dibujo de algún objeto encapsula su forma global de una manera más minimalista usando menos datos que una pintura o fotografía, lo que hace que sea más fácil trabajar con ellos.

[3]Por ejemplo, hay 16,384 píxeles en una imagen tamaño 128x128.

1.2. Objetivos

El objetivo de esta tesis es desarrollar técnicas para remodelar la salida de un modelo generativo de dibujos tipo autocodificador variacional, que incrementen la probabilidad de que un dibujo generado sea considerado una mezcla visual entre dos categorías[4].

Objetivos específicos Esta tesis tiene los siguientes objetivos específicos.

- Analizar las ventajas y desventajas de modelos generativos basados en redes neuronales profundas en comparación con otros pintores artificiales existentes hasta el momento.
- Examinar a detalle las técnicas en existencia que utilizan autocodificadores variacionales para realizar mezcla visual.
- Desarrollar un nuevo modelo evaluativo basado en redes neuronales profundas que evalúe automáticamente si un dibujo representa una mezcla visual en función de su similitud a varias categorías de objetos.
- Proponer e implementar nuevas técnicas para remodelar la salida de un autocodificador variacional con el fin de generar dibujos que sean considerados mezclas visuales con mayor frecuencia.
- Evaluar las capacidades, tanto de las técnicas existentes como de las nuevas propuestas, para realizar mezcla visual.

[4]Se decidió enfocarse en mezclas entre solamente dos categorías para simplificar la tarea.

1.3. Planteamiento del problema

Los objetivos de esta tesis presentan una variedad de dificultades computacionales, las cuales se detallan a continuación.

1.3.1. Pintores artificiales que dibujan

Aunque existen varios pintores que realizan mezcla visual con dibujos, estos pintores tienen varias limitaciones. Primero, todos trabajan con dibujos descompuestos de antemano en varias partes inmutables que sólo pueden ser reorganizadas en un espacio bidimensional. Por lo tanto, estos pintores están limitados a mezclas visuales compuestas por combinaciones de estas partes. Además, la tarea de descomponer dibujos en sus partes es laboriosa dado que cada parte de cada dibujo tiene que ser etiquetada individualmente. Actualmente los pintores de esto tipo en existencia se limitan a experimentos con un dibujo por categoría dividido en hasta veinte partes y entre dos y diez categorías (Pereira y Cardoso, 2002; Mukaiyama, 2013; Cunha et al., 2017). En contraste, en este trabajo el objetivo es mezclar dos categorías con un pintor que puede generar miles de dibujos que no están limitados a un conjunto fijo de partes y filtrar los que pueden ser considerados mezclas visuales.

Para llevar a cabo la selección de un pintor adecuado, se realizó una revisión de los pintores artificiales en existencia, la cual se encuentra en el capítulo 2. En resumen, la mayoría están demasiado enfocados a dominios pequeños (como retratos) para ser usados para la mezcla visual entre categorías de objetos en general. No obstante, existe un tipo de pintor basado en redes neuronales profundas que puede aprender a generar diferentes tipos de dibujos a través de ejemplos, llamado «autocodificador

variacional» (Kingma y Welling, 2014). Los autocodificadores variacionales toman miles de dibujos de entrenamiento de varias categorías[5] y los convierten en puntos en un espacio de alta dimensionalidad llamado «espacio latente»; estos espacios comúnmente contienen alrededor de 10^{128} puntos distintos. El mismo modelo puede ser usado para convertir un punto del espacio latente de nuevo en un dibujo. La mayoría de estos puntos corresponden a dibujos visualmente similares a los dibujos de entrenamiento, pero algunos también pueden ser interpretados como mezclas visuales. Sin embargo, estos puntos no son comunes en el espacio latente y tampoco parecen estar organizados en regiones que son fáciles de definir o encontrar (Graves, 2013; Ha y Eck, 2017; Chen et al., 2017).

Una posible solución para encontrar estos dibujos es implementar técnicas de búsqueda eficientes a través de aprendizaje de máquinas; este enfoque ha sido aplicado para realizar mezclas entre imágenes de la misma categoría (Higgins et al., 2017; Engel et al., 2017; Berthelot et al., 2018), pero no ha sido estudiado para realizar mezclas entre dos categorías distintas. Sin embargo, estas técnicas requieren evaluar dibujos en cada paso de la búsqueda de manera automatizada, lo que deriva en el problema que se formula en la siguiente sección.

1.3.2. Evaluación

Se requiere un modelo evaluativo automatizado que puede evaluar la similitud de un dibujo con múltiples categorías de objetos simultáneamente. Sin embargo, actualmente no existe un modelo que realice esta tarea. Varios pintores artificiales

[5]Aunque también es laborioso construir bases de datos para entrenar un autocodificador variacional, actualmente existen varias bases de datos compatibles (Ha y Eck, 2017).

aplican el enfoque de descomponer un dibujo o imagen en partes que pueden atribuirse a cada categoría (Pereira y Cardoso, 2002; Mukaiyama, 2013; Cunha et al., 2017). Desafortunadamente, estos trabajos realizan estas descomposiciones a mano, no de manera automatizada. Existen varios modelos de redes neuronales profundas que realizan una descomposición similar, como la identificación de rasgos faciales (Le et al., 2012) o la separación de objetos de su contexto (Lin et al., 2014). Sin embargo, todos trabajan con fotografías en formato de matrices de píxeles y no son fáciles de aplicar a dibujos, que además se encuentran en formato de secuencias de lineas.

Por otro lado, también existen varios modelos que clasifican secuencias, aunque ninguno trabaja con dibujos (Lipton et al., 2016; Ng et al., 2015; Dai y Le, 2015). Para poder entrenar estos modelos para reconocer dibujos, se requiere una base de datos con miles de ejemplos. Una base de datos de dibujos suficientemente grande que cuente con información sobre la descomposición no existe, pero sí hay varias bases de datos de dibujos grandes que pueden ser adaptadas a la tarea de clasificar similitudes sin realizar una descomposición (Eitz et al., 2012; Sangkloy et al., 2016; Ha y Eck, 2017).

Por estas dificultades, el enfoque que se siguió fue adaptar una base de datos de dibujos existente y un modelo clasificador de secuencias existente para realizar la tarea de evaluar la similitud de un dibujo con múltiples categorías; el capítulo 5 entra en detalle a este tema. Con este clasificador, fue posible implementar técnicas de búsqueda eficientes para filtrar la salida de un modelo generativo; el capítulo 6 entra en detalla a este tema.

1.4. Hipótesis

Este trabajo parte de la suposición de que un autocodificador variacional entrenado para generar dibujos en dos categorías por separado, también puede generar dibujos que representen una mezcla entre estas categorías, pero que es difícil desarrollar técnicas para encontrar y generar frecuentemente estos dibujos. Esta suposición se basa en experimentos realizados por otros investigadores; el capítulo 4 contiene más detalle. Por lo tanto, la hipótesis principal de este trabajo es que es posible desarrollar una técnica que integre información de una red evaluativa para filtra la salida de una red generativa y que además esta técnica resultará en dibujos que cumplen más con la definición de mezcla visual presentada anteriormente que otras técnicas que no cuentan con este tipo de información. Esta hipótesis se basa en antecedentes de pintores artificiales que mejoran su rendimiento integrando pasos de retroalimentación; el capítulo 2 presenta una revisión al respecto. También se basa en el hecho de que existen redes neuronales que realizan tareas de clasificación similares en otros dominios; el capítulo 5 contiene más información.

1.5. Contribuciones

Las principales contribuciones de esta tesis son:

- **Revisión de pintores artificiales.** Una revisión de lo que se ha hecho en el campo de los pintores artificiales y su relevancia en la mezcla visual.
- **Clasificación de similitud automática.** Una red neuronal entrenada para

evaluar automáticamente la similitud de un dibujo con un conjunto de categorías de objetos usando múltiples etiquetas.

- **Técnicas para remodelar la salida de un autocodificador variacional.** Nuevas técnicas para remodelar la salida de un autocodificador variacional usando retroalimentación del clasificador de similitud mencionado.

- **Análisis de técnicas para buscar mezclas visuales.** Experimentos que cuantifican las capacidades, tanto de las técnicas en existencia como de las nuevas propuestas, para buscar en un espacio grande dibujos que optimizan la similitud con dos categorías de objetos. Los experimentos incluyen una encuesta realizada a personas sobre la mezcla visual para cuantificar qué tanto los humanos consideran que estos dibujos son mezclas visuales.

1.6. Organización

Esta tesis se encuentra organizada de la siguiente manera:

- En el capítulo 2 se hace una revisión de la historia de los pintores artificiales.

- En el capítulo 3 se exponen los fundamentos de las redes neuronales, indispensables para los capítulos que siguen.

- El capítulo 4 profundiza en las técnicas más prominentes que utilizan redes generativas profundas para realizar mezcla visual.

- El capítulo 5 presenta un modelo clasificador que automáticamente evalúa similitudes entre dibujos y categorías de dibujos.

- El capítulo 6 describe nuevas técnicas para realizar mezcla visual con redes generativas profundas.
- El capítulo 7 presenta los experimentos realizados y los resultados obtenidos.
- El capítulo 8 resume las conclusiones y propone posibles direcciones de trabajo futuro.

Capítulo 2

Pintores artificiales: una revisión

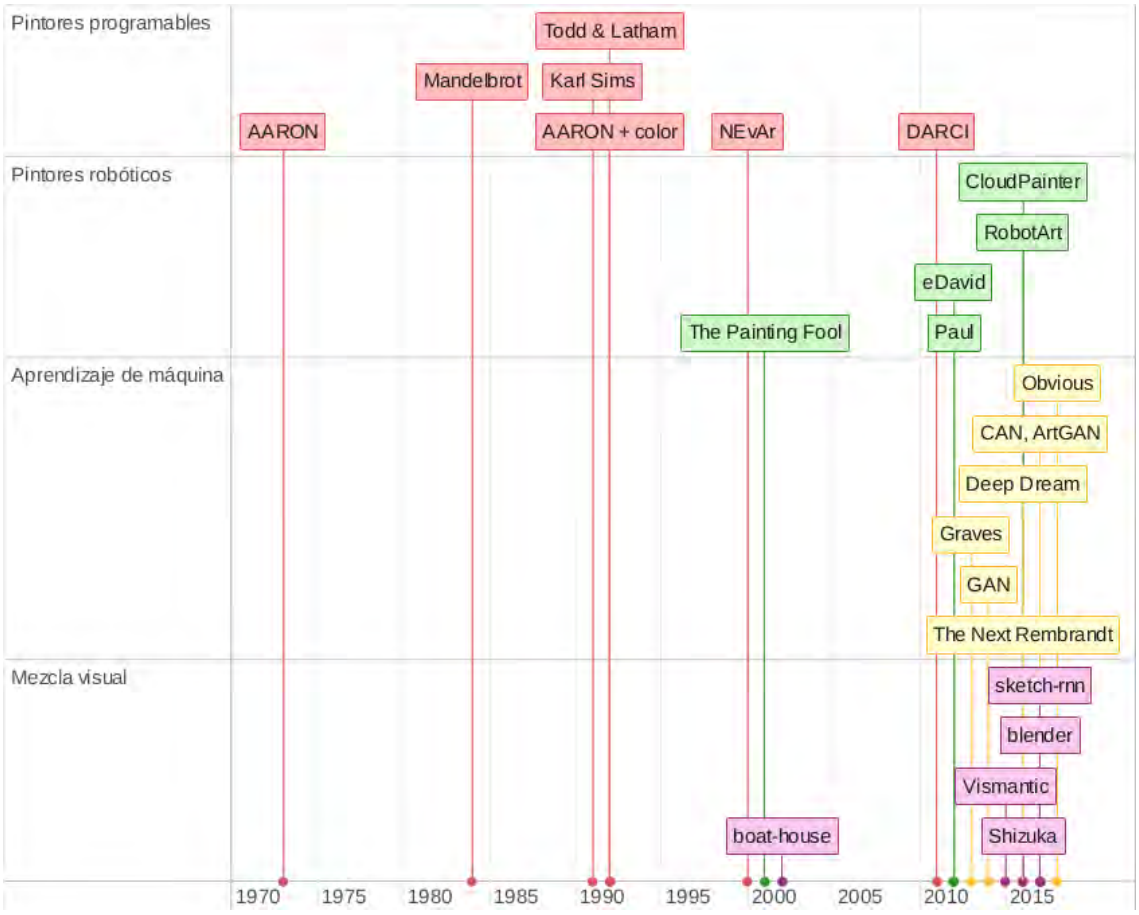


Figura 2.1: Cronograma de pintores artificiales, 1970 al presente.

Para facilitar la investigación sobre la mezcla visual en pintores artificiales, es necesario elegir un pintor que pueda servir como una plataforma para la experimentación. Esta tesis se enfoca en técnicas del aprendizaje de máquinas, pero existen un rango diverso de pintores que utilizan otros enfoques. Es importante resumir estos antecedentes para entender en cuáles aspectos los otros enfoques son insuficientes y en cuáles áreas el aprendizaje de máquinas puede ofrecer una ventaja. Los pintores artificiales más prominentes pueden ser agrupados en cuatro categorías generales según el enfoque utilizado; este capítulo proporciona una revisión breve de cada uno de ellos.

2.1. Pintores programables

Durante el cenit del arte expresivo en los años 1950 emergió una nueva raza de pintor, los *Méta-Matics* de Jean Tinguely. Estas máquinas fueron compuestas de brazos de madera que siguieron caminos predeterminados por engranajes y palancas, creando mayormente arte abstracto. La llegada de la computadora personal permitió que el robot de madera fuera reemplazado por una máquina trazadora llamada «plóter», como hizo el artista y académico Harold Cohen con su pintor artificial AARON (Cohen, 1988). Cohen programó su máquina con varias secuencias de movimientos para generar una variedad de dibujos sencillos de humanos y plantas; su trabajo representa un salto de lo abstracto a lo representativo. La contribución de Benoit Mandelbrot en los años 1980 fue el descubrimiento de funciones matemáticas llamadas «fractales», que trazadas producían curvas o formas geométricas que se contenían a sí mismas. Algunos fractales producen formas que imitan las encontradas en la naturaleza como colinas y copos de nieve (Mandelbrot y Pignoni,

1983).

La introducción de los algoritmos evolutivos incrementó la complejidad de los pintores programables. En vez de líneas y curvas, los nuevos programas trabajan con matrices de píxeles y secuencias compuestas de transformaciones como escala, rotación y deformación. Estos pintores generan grandes cantidades de secuencias de transformaciones, que una vez aplicadas a una imagen base producen nuevas imágenes. Las secuencias que crean imágenes favorables, según un evaluador humano, son combinadas para generar nuevas secuencias e imágenes. En cada iteración, las partes de las secuencias que resultan en efectos visuales más deseables sobreviven mientras que las otras desaparecen. Después de muchas iteraciones, las secuencias que quedan tienen una concentración más alta de partes favorables y por lo tanto corresponden a imágenes favorables, como la selección natural. Aunque Sims (1991) fue el primero en aplicar el algoritmo evolutivo a la generación de imágenes abstractas, otros investigadores extendieron la idea general para producir edificios (Soddu, 2002), caricaturas (Romero y Machado, 2007), criaturas fantásticas (Lambert et al., 2013) y otros tipos de imágenes. Inicialmente, cada imagen producida fue evaluada por un ser humano utilizando una interfaz específicamente diseñada (Unemi, 1998) o por voto democrático como en el experimento *Electric Sheep* (Draves, 1999). En cambio, DARCI (Norton et al., 2011) reemplazó al evaluador humano por una red neuronal artificial que intenta identificar imágenes con propiedades deseables, aunque todavía no lo puede hacer con confianza[1].

La mayoría de estos trabajos son demasiado específicos para ser útiles en esta

[1]En un estudio realizado por los mismos autores, el sistema correctamente asoció una palabra con su correspondiente imagen el 60 % de las veces. En una encuesta incluida en el mismo artículo, el pintor recibió una puntuación promedio de 2.7 de 5 (Norton et al., 2013)

tesis. Los primeros pintores presentados se componen de funciones que dibujan un grupo pequeño de formas u objetos y sería difícil modificarlas para generar otras. Sigue siendo laborioso trabajar con los pintores que se basan en la computación evolutiva, debido a que se requiere un evaluador humano en cada paso del proceso generativo.

2.2. Pintores robóticos

Con la creciente disponibilidad comercial de los brazos robóticos industriales aparecieron pintores que se enfocan en el modelado del acto físico de pintar. Estos robots trabajan a partir de alguna imagen de referencia, pintando una réplica trazo por trazo usando movimientos programados para replicar ciertos tipos de pinceladas y efectos, como el sombreado. Pintores robóticos como *eDavid* y *Paul* ejemplifican este proceso por su inclusión de pasos de retroalimentación, en los cuales el pintor toma una fotografía de su trabajo en progreso y la compara con la imagen de referencia para minimizar las diferencias (Deussen et al., 2012; Tresset y Leymarie, 2012). Otros pintores como *The Painting Fool* realizan los mismos pasos de construir una imagen trazo por trazo digitalmente, sin utilizar un brazo robótico (Colton, 2012). Hoy en día existe una buena cantidad de pintores robóticos experimentales o realizados por equipos estudiantiles que funcionan usando variaciones de este concepto básico. En la competencia inaugural de *RobotArt*, fundada por el empresario Andrew Conru en 2016, 17 equipos de distintas universidades registraron sus propios pintores robóticos (Sayej, 2016).

Aunque estos pintores son interesantes, no es muy claro cómo pueden ser aplicados a la tarea de mezclar. La mayoría de las investigaciones están más interesadas

en la técnica de físicamente realizar pincelados de varios estilos que en el contenido de las imágenes. También producen pinturas que son variaciones en una imagen de referencia, un sesgo difícil de superar en cuestión a la autonomía. En general, el conocimiento que manejan es más relacionado a estilos y movimientos que formas.

2.3. Pintores basados en técnicas de aprendizaje de máquinas

El resurgimiento de las redes neuronales artificiales a partir de los años 2000 condujo al desarrollo de las llamadas redes neuronales generativas. Cada pintor de este grupo, aprende su propia secuencia de operaciones para convertir matrices de píxeles aleatorios a imágenes que parecen visualmente similares a un conjunto de imágenes de entrenamiento, que a su vez representan algún concepto o estilo visual. Por ejemplo, se han entrenado modelos generativos para producir imágenes de números manuscritos (Gregor et al., 2015), fotografías de objetos generales (Springenberg, 2016; van den Oord et al., 2016) e imágenes estilo anime (Jin et al., 2017). Recientemente, las redes neuronales también han sido aplicadas a la generación de imágenes que parecen pinturas. En 2016, el proyecto de publicidad *The Next Rembrandt* dio a conocer una imagen de 148 millones de píxeles en el estilo del maestro holandés Rembrandt van Rijn. En 2018 el colectivo francés *Obvious* vendió una imagen impresa modelada a partir de un conjunto de obras de artistas y pintores de los siglos XIV al XX en una subasta de Christie's en Nueva York por 432,000 USD (aprox. 8,400,000 MXN) (Brown, 2016; Tan et al., 2017; Elgammal et al., 2017; Nugent, 2018).

Las redes neuronales también pueden ser usadas de otras maneras. Una aplicación es enseñarles a extraer y a transferir estilos artísticos entre imágenes. Estos modelos separan características como textura, pincelada y otros elementos que definen el estilo artístico del contenido de una imagen. Después, estas características pueden ser transferidas a otras imágenes también usando redes neuronales, creando una variación de la imagen en un nuevo estilo. Uno de los ejemplos presentados fue una fotografía de un paisaje urbano representado en el estilo de la pintura *La noche estrellada* de Vincent van Gogh (Gatys et al., 2016). Mientras tanto, otros investigadores aplicaron la técnica para insertar un objeto nuevo a una imagen, cambiando el estilo del primero para corresponder con el del último (Zhao et al., 2015). Otra forma de usar las redes neuronales consiste en voltear una red neuronal entrenada para identificar objetos para visualizar las formas con mayor probabilidad de ser identificadas. La técnica, llamada *Deep Dream*, involucra un proceso en el cual el pintor empieza con una matriz de píxeles aleatorios y realiza modificaciones iterativamente para maximizar la probabilidad de que la imagen representa algún objeto. Como en la otra aplicación, este proceso puede insertar nuevos objetos en cualquier imagen (Mordvintsev et al., 2015; McCaig et al., 2016).

El estado del arte para modelos generativos trabaja mayormente con fotografías representadas por matrices pequeñas de píxeles, lo que hace que una gran parte de los esfuerzos de investigación se desvía a resolver problemas sobre el tamaño y realismo de las imágenes creadas. Dado que esta tesis trabaja con dibujos, un dominio que no padece de estos problemas, la mayoría de los antecedentes no son directamente útiles. Las técnicas de transferencia de estilo y *Deep Dream* pueden hacer que un objeto aparece de la nada en una imagen pero no tratan al problema

de determinar dónde insertarlo y cómo integrarlo con las formas de los otros objetos preexistentes, dos problemas importantes para la tarea de la mezcla visual.

2.4. La mezcla visual

Pintores que se especializan en la mezcla entre varias imágenes son relativamente nuevos. *Vismantic* (Xiao y Linkola, 2015) mezcla imágenes fotográficas haciendo uso de tres técnicas clásicas: la yuxtaposición, el reemplazamiento y la fusión. La primera es análoga a un *collage*, donde varios elementos están simplemente posicionados unos junto a otros sin traslaparse. En el reemplazamiento, uno de los objetos reemplaza por completo a otro en su ambiente original.

Por ejemplo, un foco reemplaza el bulbo de una flor sin alterar sus hojas para aludir al tema de energía verde o sustentable. Con la llamada fusión, la textura de algún objeto es transferida a otro objeto sin impactar su entorno. El pintor *Vismantic* es supervisado en todas estas tareas por un operador humano, quien aprueba sus acciones. El pintor *Shizuka* (Mukaiyama, 2016) trabaja con imágenes de siluetas de animales divididas en varias partes. Estas partes se combinan posteriormente en nuevas configuraciones, creando criaturas fantásticas como pájaro-hombres. Los pintores *boat-house* (Pereira y Cardoso, 2002) y *blender* (Cunha et al., 2017) trabajan con dibujos divididos en partes. El primero realiza mezclas entre un dibujo de un barco y otro de una casa, mientras que el segundo trabaja con dibujos de las categorías cerdo, cactus y ángel. Un programador divide varios dibujos fuente en partes y también define lineamientos sobre la composición, por ejemplo que la cabeza de un ángel debe conectarse a la parte superior de su cuerpo. En *blender*, el pintor artificial puede usar estas reglas opcionales para auto-evaluar sus creaciones y así

elegir combinaciones que maximizan el número de reglas cumplidas. Estos últimos dos trabajos parten de la «teoría de la mezcla conceptual», la cual se describe como un proceso que involucra la proyección selectiva de estructuras sobre un espacio mental de mezclas, en donde una nueva estructura puede ser elaborada (Fauconnier y Turner, 1998).

En abril de 2017, investigadores de Google Brain introdujeron *sketch-rnn*, el primer pintor basado en redes generativas profundas para producir dibujos de varias categorías de objetos como camiones y gatos (Ha y Eck, 2017). Algunos de sus experimentos lograron realizar mezclas convirtiendo cada dibujo a un formato especial en donde operaciones matemáticas, como interpolación, pueden ser aplicadas para realizar mezcla visual. La interpolación también ha sido utilizado como una forma de generar mezclas entre frases (Bowman et al., 2016) y fotografías de caras humanas (Berthelot et al., 2018), entre otros. El modelo *sketch-rnn* forma la base de esta tesis y se presenta a detalle en el capítulo 3, mientras que las técnicas utilizadas para mezclar se presentan en el capítulo 4. Una comparación cualitativa entre los dibujos hechos por *sketch-rnn* y por los otros pintores artificiales que realizan la mezcla visual, se presenta en la sección 7.3.1.

Capítulo 3

Fundamentos

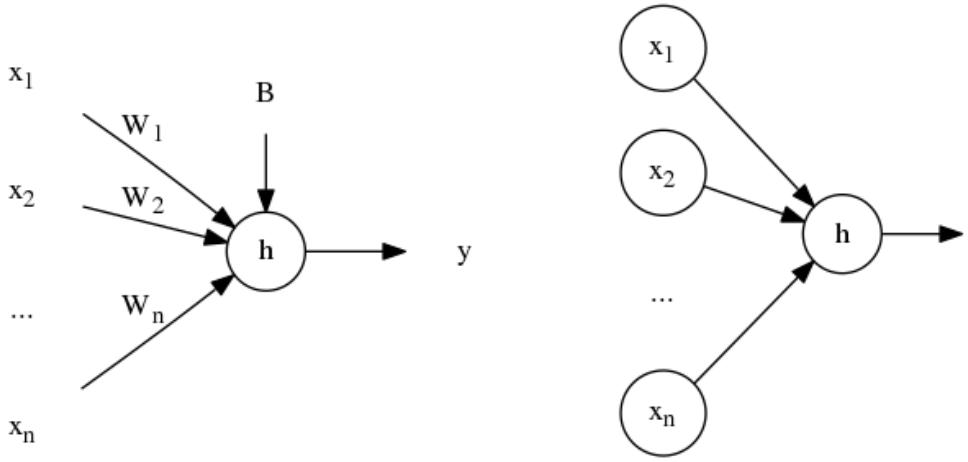


Figura 3.1: Una neurona artificial a la izquierda y en notación abreviada a la derecha.

3.1. La neurona artificial

Las redes neuronales artificiales se componen por un conjunto de funciones programáticas llamadas neuronas artificiales. La neurona artificial acepta como entrada un vector compuesto por un conjunto de valores $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$, el cual se multiplica por un vector de pesos $\boldsymbol{w}$ cuyo resultado lo suma con un sesgo b para producir un valor escalar y conocido como la salida. Lo anterior, ilustrado en la figura 3.1, se formaliza matemáticamente con la siguiente fórmula:

$$y = \sum_{i=0}^{n} w_i x_i + b \tag{3.1}$$

donde n es el número de valores en el vector de entrada. En una neurona, los pesos $\boldsymbol{w}$ y el sesgo b de la neurona pueden ser ajustados para transformar un valor dado a una salida deseada. Esta flexibilidad permite que la neurona funcione como un amplificador, filtro, multiplicador y mucho más.

3.1.1. Funciones de activación

La definición de la neurona en la ecuación 3.1 permite que la salida de una neurona artificial sea cualquier número real, es decir $y \in \mathbb{R}$. En la práctica, puede ser útil limitarla a un rango predeterminado. Para hacer esto, se puede agregar una función de activación φ.

$$y = \varphi\left(\sum_{i=0}^{n} w_i x_i + b\right) \tag{3.2}$$

El cuadro 3.1 presenta una lista de funciones de activación de uso común. La función logística o sigmoidea y la tangente hiperbólica *tanh* son en realidad ambas

Nombre	Función	Rango
Identidad	$f(x) = x$	$(-\text{ínf}, \text{ínf})$
Logística o sigmoidea	$f(x) = \dfrac{1}{1+\exp^{-x}}$	$(0, 1)$
tanh	$f(x) = \tanh(x)$	$(-1, 1)$
ReLU	$f(x) = \begin{cases} 0 & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases}$	$[0, \text{ínf})$
ReLU agujerada	$f(x) = \begin{cases} 0.01x & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases}$	$(-\text{ínf}, \text{ínf})$

Cuadro 3.1: Funciones de activación comunes y sus rangos.

funciones sigmoideas y son especialmente útiles cuando se desea limitar tanto el valor mínimo como el máximo de la salida. Por ejemplo, en el dominio de imágenes el valor de un píxel puede estar limitado a un rango de 0 a 1 que representa totalmente oscuro y totalmente encendido respectivamente. En el dominio del control de un automóvil, la aceleración puede estar limitada a -1 para frenar al máximo y 1 para acelerar al máximo. Algunas funciones definen límites superiores pero no inferiores. Por ejemplo, la unidad lineal rectificada (ReLU, por sus siglas en inglés) cancela por completo cualquier valor negativo pero deja intacto cualquier valor positivo, mientras que la ReLU agujereada o *leaky* hace lo mismo para los valores positivos pero disminuye los valores negativos en vez de cancelarlos (Nair y Hinton, 2010; Maas et al., 2013).

3.2. Redes multicapa

Para representar funciones complejas como la función de lógica binaria XOR, uno puede conectar la entrada de una neurona a la salida de otra. Varias neuronas conectadas de esta manera forman una red conocida como una red neuronal

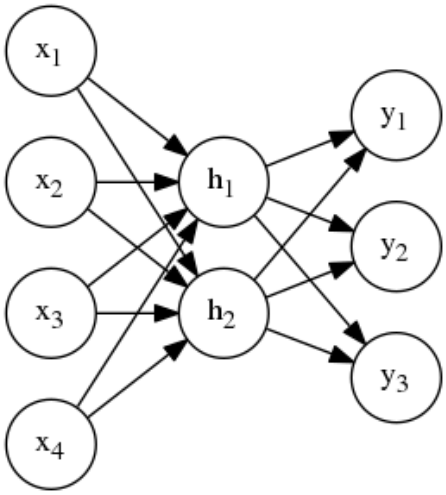


Figura 3.2: Una perceptrón multicapa con tres neuronas artificiales en la capa de entrada (izquierda), dos en la única capa oculta (centro) y tres en la capa de salida (derecha).

artificial. Un caso especial surge cuando la red está construida sin ciclos ni retroalimentación. Una red de este tipo es conocida como una perceptrón multicapa o *multilayer perceptron* (MLP, por sus siglas en inglés) y forma la base de las redes neuronales artificiales modernas. Una MLP se compone por tres o más capas de neuronas artificiales: una capa de entrada, una o más capas ocultas y una capa de salida, como se muestra en la figura 3.2. Cada nodo en la figura representa una neurona artificial que tiene una o más entradas y una salida escalar; las múltiples flechas saliendo de un nodo representan la misma salida replicada e indican la dirección de flujo de este valor. En una MLP, las neuronas están conectadas exclusivamente a todas las neuronas en la próxima capa. Las salidas de las neuronas en la capa i forman un vector $\boldsymbol{y}^{(i)} = [y_1, y_2, ..., y_m]$ de la siguiente manera:

$$\boldsymbol{y}^{(i)} = \varphi^{(i)} \left(\boldsymbol{W}^{(i)} \boldsymbol{y}^{(i-1)} + \boldsymbol{b}^{(i)} \right) \tag{3.3}$$

donde $\boldsymbol{W}$ es una matriz de pesos y $\boldsymbol{b}$ un vector de sesgos.

3.3. Optimizadores

Una red neuronal aproxima una función o fenómeno matemático tras modelar cómo un conjunto de variables dependientes cambia cuando ciertas variables independientes cambian; los conjuntos se conocen como la salida y la entrada respectivamente. En la tarea de aproximar, el objetivo es minimizar la diferencia entre la salida conocida de la función y la salida predicha por la red. En este contexto, una función que mide esta diferencia se conoce como una función de pérdida.

3.3.1. Funciones de pérdida

La tarea más relevante para este trabajo es la clasificación binaria multi-etiqueta, en la cual una entrada se asocia de antemano con un conjunto de etiquetas y la red neuronal intenta predecir estas etiquetas dada solamente la misma entrada. En el contexto de esta tesis, la entrada es un dibujo y una etiqueta señala similitud a alguna categoría. Se dice que un dibujo lleva una etiqueta si éste es similar a la categoría de la etiqueta. Un dibujo puede ser similar a múltiples categorías a la vez por lo que la tarea se llama clasificación multi-etiqueta; si esto no fuera el caso la tarea sería conocida como clasificación multi-clase. El conjunto de etiquetas asignado a una entrada toma la forma de un vector $\boldsymbol{y}$. La predicción de la red toma la forma de un vector $\hat{\boldsymbol{y}}$ que está compuesto por neuronas de salida con activación sigmoidea, una para cada etiqueta.

En este contexto, una función que mide la capacidad de la red para predecir un

$\hat{\boldsymbol{y}}$ similar a $\boldsymbol{y}$ se conoce como la entropía cruzada binaria multi-etiqueta ℓ_{XE}:

$$\ell_{XE} = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right) \tag{3.4}$$

donde n es el número de etiquetas o categorías, también conocido como la dimensionalidad del vector $\boldsymbol{y}$.

3.3.2. Gradiente descendente

Dada una función de pérdida y una red compuesta de neuronas artificiales, el siguiente paso es ajustar los pesos y sesgos de las neuronas, también conocidos como los parámetros, en una manera deliberada que minimice la función. Uno de los algoritmos que puede realizar esta tarea es conocido como gradiente descendente y tiene los siguientes pasos:

1. Pasar todas las entradas para las cuales existe una salida conocida por la red para calcular la salida predicha.

2. Usar la función de pérdida para calcular la diferencia entre la salida predicha y la salida conocida.

3. Evaluar el gradiente de la función de pérdida con respecto a cada uno de los parámetros.

4. Seguir la pendiente más inclinada del gradiente para realizar una pequeña modificación a cada parámetro, de tal forma que se reduzca la pérdida.

5. Repetir desde paso 1.

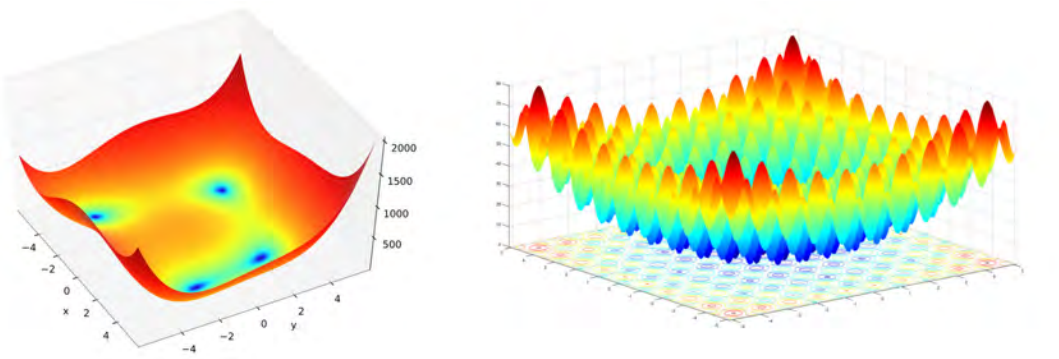


Figura 3.3: Izquierda a derecha: las funciones *Himmelblau* y *Rastrigin* de Himmelblau (1972) y Mühlenbein et al. (1991) respectivamente. Imágenes de los usuarios *Morn the Gorn* y *Diegotorquemada* de Wikipedia en el dominio público.

El tamaño de la modificación en el cuarto paso toma la siguiente forma:

$$\Delta\boldsymbol{\theta} = -\alpha\frac{df}{d\boldsymbol{\theta}} \tag{3.5}$$

donde $\boldsymbol{\theta}$ es una matriz que contiene todos los parámetros de la red y α es una tasa de escala y f es la función de pérdida.

El algoritmo puede ser modificado con la adición de una propiedad llamada «momento», que trata dos problemas que pueden ser ilustrados con la ayuda de la figura 3.3. Primero, la gráfica de la derecha contiene muchos valles o puntos mínimos. Solamente uno de ellos es el más profundo y por lo tanto hay que explorar múltiples opciones, pero este algoritmo no tiene ninguna manera de salir de un valle una vez que ya está adentro. Segundo, la gráfica de la izquierda contiene regiones planas donde el gradiente es muy pequeño en cualquier dirección. El algoritmo tampoco puede progresar aquí, dado que el gradiente no prefiere fuertemente a ninguna dirección en particular. Con la adición de momento, el tamaño de la próxima modificación se convierte en una función de la modificación anterior y el gradiente actual,

lo que permite que suban las colinas contra el gradiente o que se crucen llanos en su ausencia. Con el momento, el tamaño de la modificación en el cuarto paso toma la siguiente forma:

$$\begin{aligned} \boldsymbol{v} &= \beta \boldsymbol{v} - \gamma \frac{df}{d\boldsymbol{\theta}} \\ \Delta \boldsymbol{\theta} &= \boldsymbol{v} \end{aligned} \tag{3.6}$$

donde $\boldsymbol{v}$ es el momento y β y γ son tasas de escala configurables. Implementaciones modernas del GD tienden a utilizar tasas β y γ dinámicas, que empiezan altas y se reducen automáticamente conforme pasa el tiempo. Otras pueden tener un momento adaptable, como el optimizador *Adam* (Kingma y Ba, 2015).

Gradiente descendiente estocástico

Para aplicar el algoritmo del GD, es necesario pasar cada una de las potencialmente millones de entradas por la red para calcular el gradiente verdadero antes de realizar una sola actualización de los parámetros. Una modificación llamada el gradiente descendente estocástico (SGD, por sus siglas en inglés) evita el problema con una aproximación del gradiente. En este enfoque, el gradiente es aproximado usando un conjunto pequeño de entradas seleccionadas al azar y se actualizan los parámetros usando este gradiente. La selección aleatoria de entradas, la aproximación del gradiente y la actualización de los parámetros se repiten iterativamente como en el algoritmo del GD (Bottou, 2010).

3.3.3. El algoritmo de retropropagación

En teoría, el gradiente de la función de pérdida con respecto a cada uno de los parámetros puede ser calculado simplemente con la aplicación de la regla de la

cadena:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{3.7}$$

donde z es una variable que depende de la variable y, y se desea derivar z con respecto a x. La regla puede ser aplicada repetidamente para calcular el gradiente de una función compleja esencialmente rompiéndola en capas derivables, e ir calculando iterativamente el gradiente de cada capa con respecto a las otras. Esta regla permite que el gradiente de la función de pérdida sea propagado hacia atrás desde la función de pérdida a cada parámetro de la red de la siguiente manera:

1. Tomar una entrada y su salida esperada de la base de datos, p.ej. un dibujo y la categoría de objeto que representa.

2. Calcular la salida de cada neurona de la red al alimentar la primera capa con la entrada, comenzando con esta capa y procediendo hasta la última.

3. Calcular el valor de la función de pérdida, comparando las salidas de la última capa con la salida esperada.

4. Calcular el gradiente de la función de pérdida con respecto a cada parámetro de la red, comenzando desde la última capa y procediendo hasta la primera.

5. Modificar los parámetros de acuerdo con la ecuación 3.5 usando estos gradientes.

3.4. Redes neuronales recurrentes

En ciertas tareas, es útil tomar en consideración no solamente los datos de entrada sino la secuencia u orden en que están organizados. Por ejemplo, en un dibujo

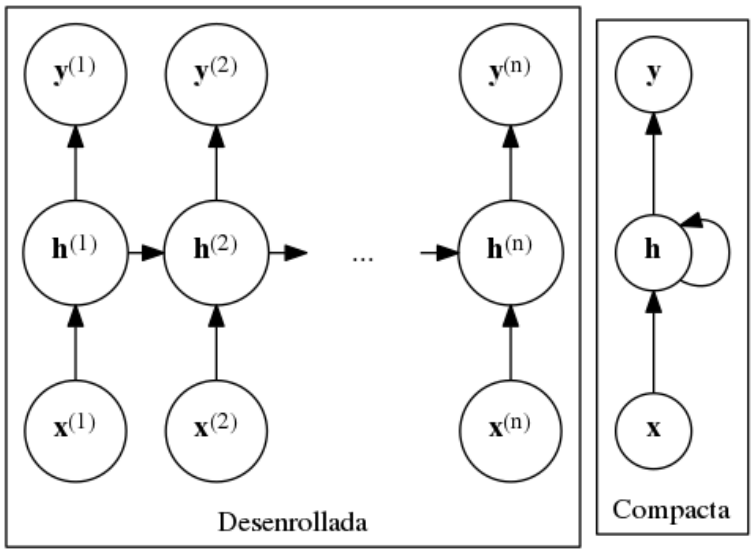


Figura 3.4: Dos formas de representar la arquitectura general de una red neuronal recurrente.

en formato de secuencia de puntos la probabilidad de que el siguiente punto caiga dentro de cierta región depende fuertemente de lo que ya se había trazado hasta el momento. Las redes neuronales recurrentes (RNN, por sus siglas en inglés) procesan secuencias punto por punto, usando capas de neuronas artificiales para actualizar un estado interno $\boldsymbol{h}$, llamado su memoria, iterativamente. La figura 3.4 muestra dos maneras de representar esta arquitectura, en la cual n es el número de puntos en la secuencia y $\boldsymbol{x}^{(i)}$ es el i-ésimo punto de la secuencia. La forma «desenrollada», a la izquierda, hace explícito el flujo de información pero puede ocultar el hecho de que los pesos y sesgos de la RNN son los mismos para cada punto. La forma a la derecha representa la misma red en forma compacta. El acto de guardar el estado interior de la red después de ver un punto y volver a usar esta información para el siguiente punto, es lo que hace que esta arquitectura sea recurrente.

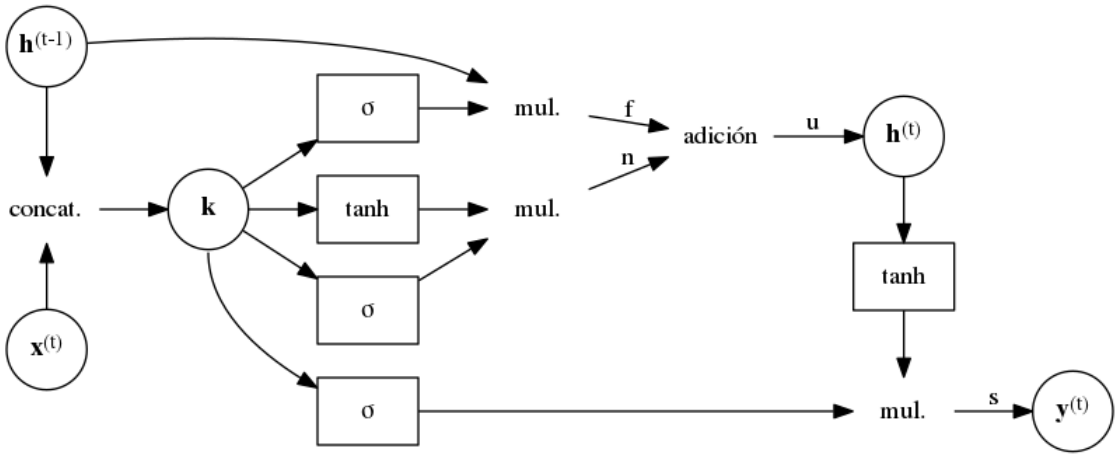


Figura 3.5: La estructura de una celda de memoria a corto y largo plazo. Los círculos representan un vector, cada rectángulo indica una capa de neuronas artificiales con función de activación señalada por su etiqueta y los nodos sin contornos son operaciones matemáticas.

Memoria a corto y largo plazo

Una forma de implementar el concepto de memoria es a través de una entidad especial llamada «celda de memoria a corto y largo plazo» (LSTM, por sus siglas en inglés) (Hochreiter y Schmidhuber, 1997). La figura 3.5 ilustra la arquitectura de una celda LSTM. Estas celdas generan dos vectores: uno es interpretado como su memoria y el otro es interpretado como su salida.

El primer paso es decidir qué información de la memoria es relevante para el punto que está siendo procesado actualmente. Su decisión está basada tanto en el estado de la memoria después del último punto $\boldsymbol{h}^{(t-1)}$ como en el punto actual $\boldsymbol{x}^{(t)}$. Una capa de redes neuronales con activación sigmoidea convierte una concatenación entre estos dos vectores, $\boldsymbol{k}$, a un nuevo vector de pesos. Estos pesos se multiplican por $\boldsymbol{h}^{(t-1)}$ para producir un dato $\boldsymbol{p}_1$. Lo anterior se indica con la flecha marcada con f en la figura 3.5.

El segundo paso es proponer una actualización al estado $\boldsymbol{h}^{(t-1)}$. De nuevo, una capa sigmoidea procesa $\boldsymbol{k}$ para producir un vector de pesos, mientras una capa nueva con activación *tanh* también procesa $\boldsymbol{k}$ para producir un nuevo vector. Estos dos vectores se multiplican para producir un dato $\boldsymbol{p}_2$. Lo anterior se indica con la flecha marcada con n.

El tercer paso genera el estado actualizado $\boldsymbol{h}^{(t)}$ sumando el estado anterior ponderado $\boldsymbol{p}_1$ y la actualización propuesta $\boldsymbol{p}_2$. El paso se indica con la flecha marcada con u.

Finalmente, se procesa $\boldsymbol{h}^{(t)}$ y se multiplica el resultado por pesos derivados nuevamente del dato $\boldsymbol{k}$ para generar una respuesta $\boldsymbol{y}^{(t)}$. Lo anterior se indica con la flecha marcada con s. Todas estas operaciones se formalizan de la siguiente manera:

$$\begin{aligned} \boldsymbol{k} &= \text{Concat}(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}) \\ \boldsymbol{p}_1 &= \sigma(f(\boldsymbol{k}))\boldsymbol{h}^{(t-1)} \\ \boldsymbol{p}_2 &= \sigma(f(\boldsymbol{k}))\tanh(f(\boldsymbol{k})) \\ \boldsymbol{h}^{(t)} &= \boldsymbol{p}_1 + \boldsymbol{p}_2 \\ \boldsymbol{y}^{(t)} &= \sigma(f(\boldsymbol{k}))\tanh(f(\boldsymbol{h}^{(t)})) \end{aligned} \tag{3.8}$$

donde Concat es una operación de concatenación, σ y tanh son funciones de activación y $f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, donde $\boldsymbol{W}$ es una matriz de pesos y $\boldsymbol{b}$ un vector de sesgos. Dado que las celdas LSTM están definidas por funciones derivables, se puede definir los gradientes de la RNN usando el algoritmo de retropropagación y se puede ajustar todos los parámetros de la celda siguiendo estos gradientes.

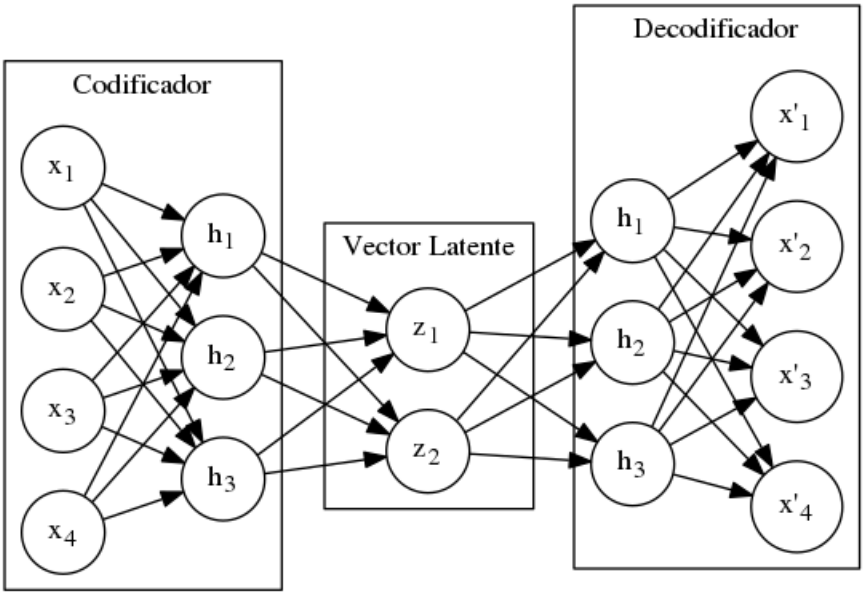


Figura 3.6: La arquitectura general de un autocodificador, con una capa oculta tanto para el codificador como el decodificador.

3.5. Auto-codificadores

Los autocodificadores variacionales (VAE, por sus siglas en inglés) son modelos generativos introducidos por Kingma y Welling (2014) que se derivan de un grupo de redes neuronales llamadas «autocodificadores». Los modelos en este último grupo son entrenados para comprimir información en un formato compacto de su propio diseño del cual se puede recuperar la información original. La estructura de un autocodificador estándar cuenta con dos partes. Primero, el codificador toma un dato de entrada y busca convertirlo en una representación interna $\boldsymbol{z}$, también conocida como un vector latente. Conceptualmente, este paso proyecta el dato a un nuevo espacio de diferente dimensionalidad, llamado el espacio latente. Segundo, el decodificador toma el vector latente e intenta reconstruir la entrada original usando solamente

este vector. La meta del autocodificador es minimizar la pérdida de información entre la entrada y la salida reconstruida. Al mismo tiempo, también busca realizar esta tarea usando un vector latente que tenga dimensionalidad menor que la de la entrada original. Los dos factores tienen que estar en equilibrio; se permite tener errores en la reconstrucción si es redituable en términos de compresión. Por ejemplo, si un dibujo tiene una característica poco común puede ser mejor simplemente ignorarla durante la codificación que dedicar una dimensión en el vector latente a representarla.

Tanto el codificador como el decodificador pueden ser implementados por una red de cualquier tipo; esto incluye las redes neuronales recurrentes introducidas en la sección anterior. Además, no es necesario que el codificador tenga la misma arquitectura que el decodificador. La función de pérdida para un autocodificador es simplemente la diferencia entre el dato original y el dato reconstruido y es conocida como la pérdida de reconstrucción ℓ_r.

Los VAE tienen una modificación importante que les permite ser modelos generativos. En vez de interpretar el resultado del codificador $\boldsymbol{h}$ como un vector latente que se puede pasar directamente al decodificador, estos modelos utilizan $\boldsymbol{h}$ para parametrizar una distribución probabilística que puede ser muestreada aleatoriamente para generar varios vectores latentes. Por ejemplo, se puede entrenar el codificador para producir un vector de medias $\boldsymbol{\mu}$ y un vector de desviaciones estándar $\boldsymbol{\sigma}$ que juntos definen una distribución gaussiana. Se puede producir un vector latente $\boldsymbol{z}$

simplemente generando muestras de esta distribución:

$$\begin{aligned} \boldsymbol{\mu} &= \boldsymbol{W}_{\mu}\boldsymbol{h} + \boldsymbol{b}_{\mu} \\ \boldsymbol{\sigma} &= \boldsymbol{W}_{\sigma}\boldsymbol{h} + \boldsymbol{b}_{\sigma} \\ \boldsymbol{z} &= \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \mathcal{N} \end{aligned} \tag{3.9}$$

donde $\boldsymbol{W}$ es una matriz de pesos, $\boldsymbol{b}$ un vector de sesgos y $\mathcal{N}$ una distribución gaussiana con media 0 y desviación estándar 1. Aunque esta distribución no es derivable, el gradiente de $\boldsymbol{z}$ con respecto a $\boldsymbol{\mu}$ y $\boldsymbol{\sigma}$ se puede calcular directamente sin necesitar el gradiente de la gaussiana. Este llamado truco de re-parametrización es un paso importante para poder utilizar los algoritmos de retropropagación y SGD con los VAE a pesar de la aleatoriedad del vector latente.

En muchas tareas, la distribución de los datos en la base de datos es gaussiana. Para promover que los vectores latentes tengan una distribución similar y por lo tanto una variedad similar, los VAE agregan una función de pérdida llamada la divergencia *Kullback-Leibler* (Kullback y Leibler, 1951). Esta función calcula la divergencia entre la distribución de los vectores latentes y una distribución gaussiana con media 0 y desviación estándar 1, escrita en forma corta como $\mathcal{N}(0, 1)$, de la siguiente manera:

$$\begin{aligned} \hat{\boldsymbol{\sigma}} &= \exp\left(\frac{\boldsymbol{\sigma}}{2}\right) \\ \ell_{KL} &= -\frac{1}{2n}\sum_{i=0}^{n}\left(1 + \hat{\sigma}_i - \mu_i^2 - \exp(\hat{\sigma}_i)\right) \end{aligned} \tag{3.10}$$

donde μ es la media, σ la desviación estándar y n la dimensionalidad de $\boldsymbol{z}$. La función de pérdida para los VAE es la suma de la pérdida de reconstrucción y la

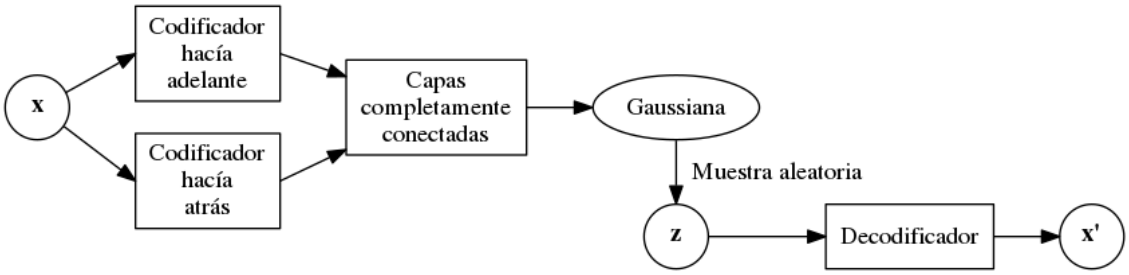


Figura 3.7: La arquitectura general de *sketch-rnn*.

pérdida KL.

El detalle que hace los VAE generativos es la capacidad, una vez que el modelo ha sido entrenado, de saltar el codificador e ir directamente al paso decodificador. Como el espacio latente tiene una distribución $\mathcal{N}(0, 1)$, entonces un nuevo vector latente puede ser generado simplemente muestreando esta misma distribución en lugar de derivarlo de una entrada conocida. Este nuevo vector se decodifica en una nueva muestra que es similar a los datos de entrenamiento.

3.5.1. *sketch-rnn*

Esta tesis utiliza el modelo *sketch-rnn* propuesto por Ha y Eck (2017) como base de experimentación, lo que hace que su comprensión sea imperativa. Este modelo, que representa el estado del arte para la generación de dibujos basados en redes neuronales, es un VAE cuyo codificador es implementado por una red recurrente bidireccional y el decodificador por una red recurrente[1]. La figura 3.7 muestra la arquitectura general de este modelo. El codificador es bidireccional, lo que significa que la red procesa el dibujo tanto en sentido normal como contrario simultáneamente.

[1]Esta arquitectura fue originalmente propuesta por Graves (2013) para generar escritura y fue modificada posteriormente por Ha y Eck para generar dibujos.

Los dos resultados se concatenan y se pasan por capas de redes neuronales completamente conectadas para generar una media y una desviación que parametrizan una distribución gaussiana, como en el enfoque de un VAE.

El decodificador maneja otra red recurrente que realiza la tarea de reconstruir la secuencia de puntos original de un vector latente. Esta red emite predicciones sobre el siguiente punto en la secuencia utilizando tanto su memoria como el vector latente. Cada predicción toma la forma de un modelo de mezcla de densidades (MDM, por sus siglas en inglés), que se compone por múltiples distribuciones gaussianas. Los MDM se definen por la siguiente fórmula (Graves, 2013):

$$p(\Delta x, \Delta y) = \sum_{j=1}^{m} \Psi_j \mathcal{N}(\Delta x, \Delta y | \mu_{x,y}, \mu_{y,j}, \sigma_{x,y}, \sigma_{y,j}, \rho_{xy,j}) \tag{3.11}$$

donde μ es la media aritmética, σ la desviación estándar, ρ el parámetro de correlación para cada una de las m distribuciones normales bivariadas y Ψ es un vector de tamaño m que tiene distribución categórica ($\sum_{j=1}^{m} \Psi_j = 1$) y representa los pesos de la mezcla.

La función de pérdida de la reconstrucción conceptualmente evalúa la capacidad del MDM para explicar el próximo punto en la secuencia y se define matemáticamente de la siguiente manera (Ha y Eck, 2017):

$$\ell_r = -\frac{1}{n_{\text{máx}}} \sum_{i=1}^{n_s} \log \left(p^{(i)}(\Delta x, \Delta y) \right) \tag{3.12}$$

donde $p^{(i)}(\Delta x, \Delta y)$ es el MDM por el punto i, $n_{\text{máx}}$ es el tamaño de la secuencia más grande de la base de datos y n_s es el tamaño de la secuencia en cuestión. El modelo *sketch-rnn* también cuenta con una función de pérdida que se trata de dos

probabilidades: la que el punto se conecte con una línea visible en vez de invisible al punto anterior y la que el punto sea el último de la secuencia. La función que define la pérdida de la reconstrucción de estos dos conceptos es nuevamente una pérdida de la entropía cruzada y se detalla en el artículo que introduce *sketch-rnn* (Ha y Eck, 2017). La función de pérdida final es la suma de estas dos funciones más la pérdida KL presentada en la sección anterior.

El proceso de decodificar un vector latente a un dibujo difiere del proceso de calcular la pérdida de reconstrucción. Mientras en la pérdida el próximo punto de la secuencia se conoce de antemano, en la generación el próximo punto depende del punto anterior, que en sí depende de una muestra aleatoria tomada del MDM generado en el paso anterior. Por lo tanto, la conversión tiene que realizarse de manera iterativa, un proceso significativamente más lento.

Cuando *sketch-rnn* se entrena con múltiples categorías de objetos, el modelo no recibe información que especifica a cuál categoría pertenece cada dibujo de entrenamiento. No hay un control que pueda usarse para especificar la categoría del dibujo generado durante la reconstrucción.

3.6. TensorFlow

Afortunadamente, no es necesario implementar a mano las complejidades de los fundamentos presentados en este capítulo ni lidiar con problemas técnicos como la representación digital precisa de números masivos. Existen varias bibliotecas con implementaciones eficientes revisadas por la comunidad para todos los algoritmos, funciones y estructuras necesarias. Una de estas bibliotecas es *TensorFlow*, originalmente desarrollada por *Google Brain* y liberada al público general en 2015. El

proyecto, escrito en C++ con una interfaz programática en Python, es tanto popular[2] como activo[3]. Una implementación de *sketch-rnn* en *TensorFlow* se encuentra libremente disponible[4].

Aunque los aspectos físicos de la computadora, como la memoria y la CPU, no tienen ningún impacto en los resultados matemáticos de las ecuaciones, la necesidad de realizar una cantidad masiva de cálculos matriciales y vectoriales invita a usar un equipo especial llamado unidad de procesamiento gráfico (GPU, por sus siglas en inglés). Dichos componentes son diseñados para realizar estos tipos de operaciones de una manera más rápida y eficiente que las CPU. *TensorFlow* también proporciona una capa de abstracción en la cual el programador no necesita pensar en las particularidades de optimizar sus interacciones con el equipo subyacente. Es decir, el investigador puede disfrutar el ahorro de tiempo de los GPU sin tener que realizar grandes modificaciones a su código fuente.

[2] `https://www.tensorflow.org/about/uses`
[3] Una versión nueva estrenó en agosto de 2018.
[4] `https://github.com/tensorflow/magenta/tree/master/magenta/models/sketch_rnn`

Capítulo 4

Técnicas de mezcla visual usando redes neuronales

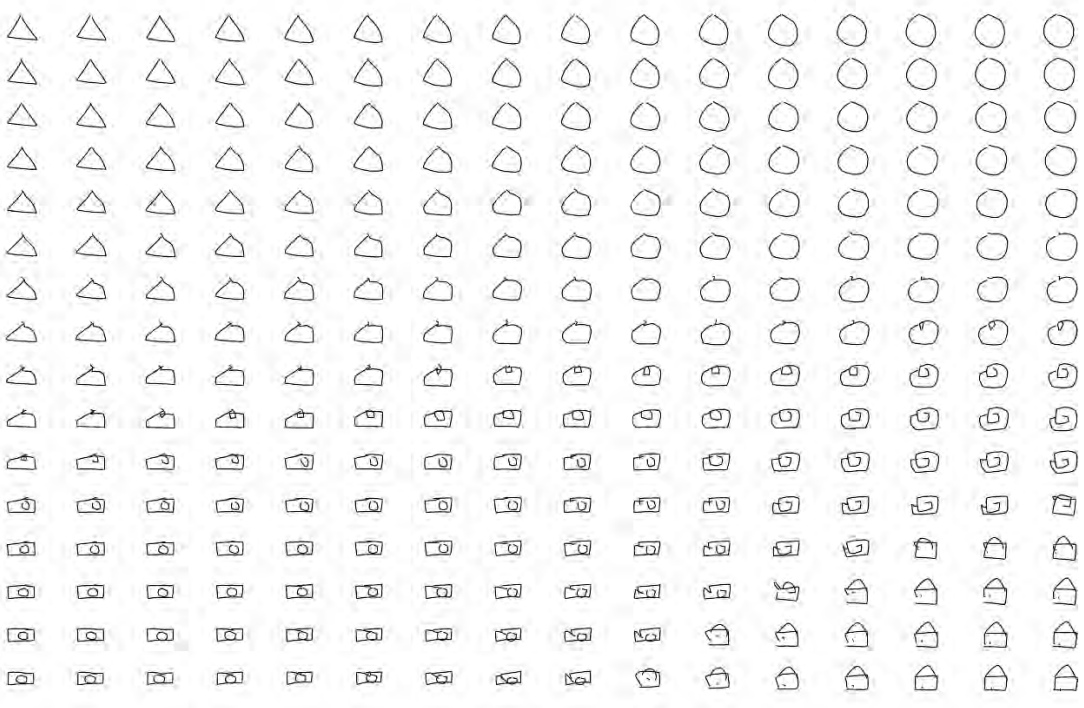


Figura 4.1: Múltiples series de interpolaciones entre dibujos de triángulos, círculos, casas y cámaras. Los dibujos de las esquinas fueron hechos por humanos y tomados de la base de datos *Quick, Draw!* (Ha y Eck, 2017) mientras que los demás fueron generados por un modelo *sketch-rnn* mediante la técnica de interpolación.

La autonomía persiste como un gran reto para los sistemas que buscan realizar la mezcla visual. En específico, trabajos anteriores han requerido intervención humana para descomponer imágenes fuente en partes manejables para que sus modelos puedan enfocarse en la tarea de generar combinaciones nuevas usando dichas partes (Pereira y Cardoso, 2002; Lambert et al., 2013; Xiao y Linkola, 2015; Cunha et al., 2017). Recientemente, investigadores que trabajan con modelos generativos basados en aprendizaje de máquinas han descubierto que sus modelos también son capaces de realizar una especie de mezcla visual sin necesitar este paso. En algunos de estos trabajos la mezcla surge como un efecto secundario, mientras que en otros el modelo original es modificado para aumentar su capacidad de mezclar. En este capítulo, se presentan las técnicas más prominentes para realizar la mezcla visual con redes neuronales artificiales, así como una discusión de su relevancia e impacto en esta tesis.

Entre los modelos de aprendizaje de máquinas generativos, destacan dos arquitecturas en particular por su popularidad: las redes generativas antagónicas (GAN, por su siglas en inglés) y los autocodificadores variacionales (VAE, por sus siglas en inglés). La clave para mezclar usando estos modelos es su vector latente, ubicado entre el codificador y decodificador para los VAE y antes de la generadora para las GAN. Este vector latente representa un concepto de alto nivel, como un dibujo o una imagen, usando un conjunto de números que pueden ser manipulados con funciones matemáticas para modificar el concepto original.

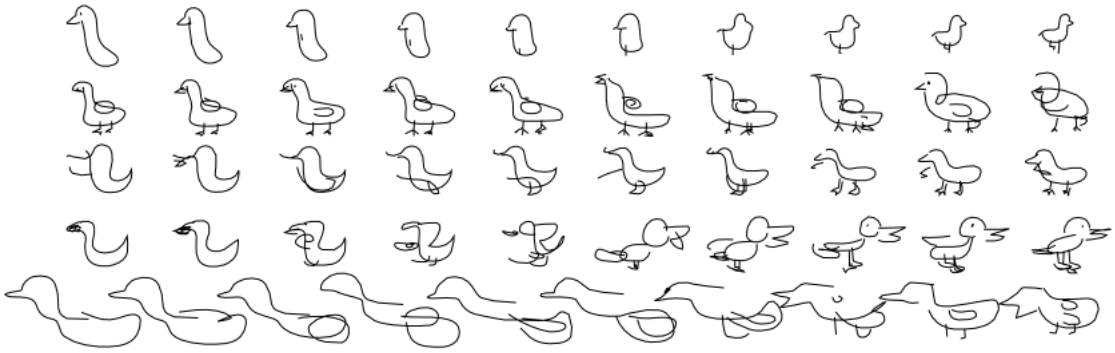


Figura 4.2: Cinco series de interpolaciones, una por fila, entre dibujos de patos. Las columnas a los extremos izquierdo y derecho contienen los dibujos originales.

4.1. Interpolación

Experimentos con interpolaciones se encuentran con frecuencia en artículos sobre autocodificadores variacionales (Bowman et al., 2016; Ha y Eck, 2017; Berthelot et al., 2018; Chen et al., 2017). La técnica combina dos vectores latentes usando una función de interpolación lineal, como la de la ecuación 4.1, o esférica, como la de la ecuación 4.2:

$$\boldsymbol{z}_t = \boldsymbol{z}_a t + \boldsymbol{z}_b (1-t), 0 \leq t \leq 1 \tag{4.1}$$

$$\begin{aligned} \cos\Omega &= \boldsymbol{z}_a \cdot \boldsymbol{z}_b \\ \boldsymbol{z}_t &= \frac{\sin\left((1-t)\Omega\right)}{\sin\Omega}\boldsymbol{z}_a + \frac{\sin(t\Omega)}{\sin\Omega}\boldsymbol{z}_b \end{aligned} \tag{4.2}$$

En ambos casos, la variable t controla la ponderación de la interpolación, en la cual $t = 1$ asigna todo el peso al primer vector y $t = 0$ todo el peso al segundo. Cuando $0 < t < 1$, el vector latente interpolado existe numéricamente entre los dos vectores, lo que en algunos casos se traduce en un dibujo que se encuentra visualmente localizado entre los dos conceptos correspondientes. La figura 4.2 muestra un ejemplo de interpolación entre dibujos de patos. En cada fila, los dos dibujos a

los extremos fueron tomados de la base de datos y convertidos en vectores latentes. La fila muestra una secuencia de interpolaciones entre estos vectores en la cual t se reduce constantemente desde $t = 1$ a $t = 0$ de izquierda a derecha. Las imágenes en el centro contienen características de ambos dibujos originales y al mismo tiempo todavía parecen dibujos de patos. Se refiere a este tipo de mezcla por el término «intracategoría», dado que se realiza entre las propiedades de una misma categoría de categoría de objeto. En contraste, la figura 4.1 muestra un ejemplo de interpolación entre dibujos de triángulos, círculos, cámaras y casas; a estas últimas mezclas se les denomina como «intercategoría» dado que se realiza entre dos o más categorías distintas y el producto no pertenece completamente a ninguna de ellas.

4.2. Aritmética

Una extensión de la interpolación es la suma y resta aritmética de dos o más vectores latentes que, como la interpolación, ha sido realizada tanto con las GAN como con los VAE (Radford et al., 2015; Ha y Eck, 2017). Los desarrolladores de *sketch-rnn* presentaron un ejemplo en el cual el vector latente de la cabeza de un gato se sumó con el vector latente de un cerdo completo menos el vector latente de la cabeza de un cerdo. Conceptualmente, el resultado de esta operación es un gato con cuerpo de cerdo. Sin embargo, esta técnica aún no funciona con tanta precisión. Consideramos que al dibujo obtenido en sus experimentos le falta la cola distintiva del dibujo del cerdo y al mismo tiempo agregó unas piernas que no están presentes en los dibujos originales. Por lo tanto, consideramos que el resultado final estuvo en desacuerdo con la especificación descrita en la operación aritmética. Las categorías con que experimentaron también se beneficiaron del hecho de que la base de datos de

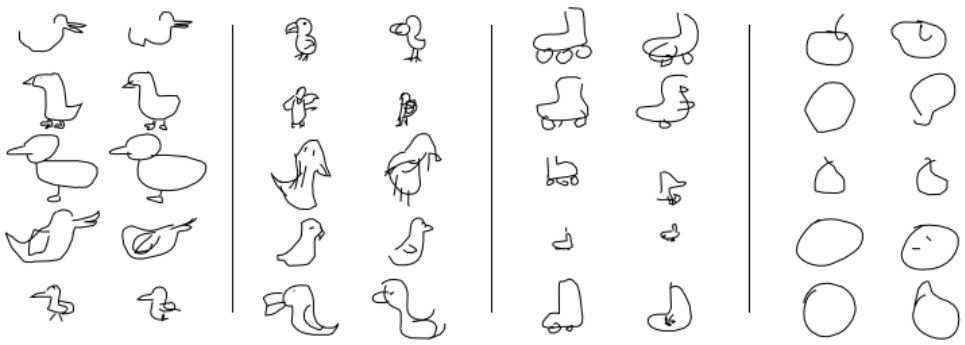


Figura 4.3: Ejemplos de dibujos reconstruidos. En cada panel, la columna a la izquierda contiene el dibujo original. De izquierda a derecha, la categoría del dibujo original en cada panel es: pato, cotorra, patín y circulo. El modelo utilizado para reconstruir los dibujos fue entrenado solo con dibujos de patos.

entrenamiento contiene dibujos de gatos sin cuerpo y de gatos completos etiquetados como la misma categoría; con otras categorías este no es necesariamente el caso.

4.3. Reconstrucción

Los modelos VAE en específico tienen un efecto secundario que también puede ser utilizado para mezclar dibujos: la reconstrucción (Ha y Eck, 2017). La figura 4.3 muestra varios ejemplos de esta técnica. Para producir el efecto, a un modelo entrenado para representar solamente una categoría de dibujos se le da la tarea de representar un dibujo de una categoría distinta. En cada fila de cada panel, el dibujo de la derecha corresponde a la reconstrucción del dibujo de la izquierda. Algunas veces el dibujo conserva aquellas características del dibujo desconocido que son suficientemente similares a las de la categoría conocida, pero reemplaza o simplemente elimina aquellas que no. En el panel de la izquierda, un modelo entrenado únicamente con patos reconstruye dibujos de patos que parecen visualmente similares

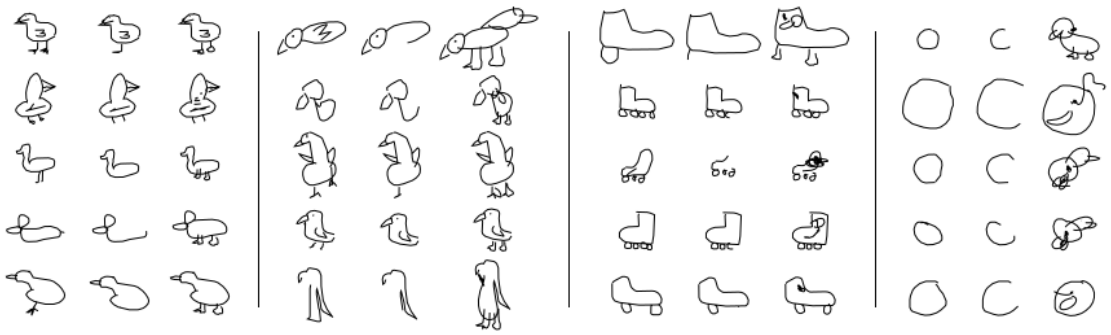


Figura 4.4: Ejemplos de dibujos autocompletado. En cada panel de izquierda a derecha: el dibujo original, recortado y autocompletado. De izquierda a derecha, la categoría del dibujo original en cada panel es: pato, cotorra, patín y circulo. El modelo utilizado para reconstruir los dibujos fue entrenado solo con dibujos de patos.

al dibujo original. En los otros paneles, el mismo modelo reconstruye dibujos de cotorras, patines y círculos respectivamente, creando en algunos casos dibujos que parecen tener propiedades de las dos categorías. Por ejemplo, la reconstrucción del patín en la segunda fila del tercer panel parece tener un ala y un pico.

4.4. Autocompletado

Los autores de *sketch-rnn* también presentaron una técnica que automáticamente completa dibujos parciales con la ayuda de una modificación al modelo original (Ha y Eck, 2017). Recordemos que *sketch-rnn* utiliza el vector latente para condicionar la salida del decodificador, lo que resulta en un dibujo distinto para cada vector latente. Mientras tanto, la técnica de autocompletado utiliza una modificación llamada modo incondicional, en la cual se eliminan tanto el codificador como el vector latente. Por lo tanto, el decodificador de este modelo produce un dibujo tomando en cuenta solamente los trazos anteriores para decidir la posición del próximo trazo. El modelo

se entrena con dibujos de una sola categoría.

Para generar, se le proporciona un dibujo parcial al decodificador que lo procesa para actualizar solamente su estado interno: conceptualmente, el decodificador es adelantado como si estuviera generando el dibujo dado. Después, el decodificador se ejecuta sin mayor intervención, agregando trazos según lo dictado por su estado interno actual. La figura 4.4 contiene varios ejemplos. Cada fila de cada panel presenta de izquierda a derecha: un dibujo tomado de la base de datos, el mismo dibujo recortado y el dibujo recortado autocompletado. Nuevamente, el panel de la izquierda utiliza un modelo entrenado únicamente con patos para autocompletar dibujos de la misma categoría, mientras que en los otros paneles el mismo modelo completa dibujos de cotorras, patines y círculos respectivamente. Es importante tomar en cuenta que el dibujo final incluye todos los trazos del dibujo parcial, lo cual representa un sesgo introducido por el usuario y también niega al modelo la oportunidad de hacer sus propias modificaciones al último.

4.5. Etiquetado externo

Una alternativa a la descomposición a mano de un dibujo o imagen es el entrenamiento de un modelo de aprendizaje de máquinas para realizar la tarea automáticamente. En este enfoque, se asigna una etiqueta que contiene información descriptiva a cada imagen en una base de datos y se entrena un modelo para automáticamente predecir la imagen dada solamente la etiqueta. En un experimento realizado por Mansimov et al. (2016), imágenes de varios objetos incluyendo autobuses escolares fueron asociadas con etiquetas que señalaron tanto el objeto como su color. El modelo generativo entonces generó autobuses de varios colores, incluyendo por ejemplo

verde, según la descripción de la etiqueta proporcionada a pesar del hecho de que no existió un autobús verde en la base de datos. La técnica también ha sido implementada con las GAN para generar imágenes tanto fotográficas como de estilo ilustrativo *anime* (Reed et al., 2016; Jin et al., 2017).

Recientemente, un concepto llamado *latent constraints* fue introducido en el cual diversos modelos GAN utilizan etiquetas externas para encontrar un sub-espacio dentro del espacio latente de un modelo VAE que contenga muestras que exhiban ciertas propiedades (Engel et al., 2017). Este trabajo contenía un experimento en el cual se aplicó la técnica a fotografías de caras humanas. Empezando con una fotografía codificada en un vector latente, aplicaron el algoritmo de gradiente descendente para mover el vector a un punto en el espacio que correspondiera más con cierta característica. En un ejemplo, una fotografía de una persona con pelo negro fue modificada de tal manera que su pelo cambió a castaño, mientras que el resto de la imagen permaneció igual.

4.6. Limitaciones y potencialidades

De las técnicas mencionadas en este capítulo, la interpolación destaca por su inclusión frecuente en artículos sobre los VAE y las GAN (Radford et al., 2015; Bowman et al., 2016; Chen et al., 2017; Berthelot et al., 2018). En particular, Ha y Eck (2017) presentaron varias pruebas usando la interpolación para mezcla visual intercategoría, el tipo de mezcla visual que esta tesis maneja. La figura 4.1 muestra una réplica de una de sus experimentos aunque con otras categorías, en la cual una serie multi-direccional de interpolaciones entre cuatro categorías de dibujos produce muestras que hasta cierto punto representan mezcla visual entre ellas. Un aspecto

negativo de este experimento es que los dibujos en la parte de en medio no pueden ser fácilmente identificados como ninguna de las cuatro categorías, a diferencia de la figura 4.2 en la cual todas las muestras se ven como dibujos de patos. Esto sugiere que para una mezcla intercategoría con *sketch-rnn*, una interpolación matemática no corresponde tan bien con una interpolación conceptual, sea porque el espacio latente contiene huecos o por otras razones[1]. Parece que la técnica todavía no ha sido perfeccionada, pero aún así su popularidad proporciona una buena línea base para realizar una comparación contra las nuevas técnicas propuestas en el capítulo 6.

La reconstrucción y el autocompletado son técnicas que parecen prometedoras, especialmente en el dominio de los dibujos. No obstante, estas técnicas no han sido tan bien estudiadas en cuestión de sus capacidades para mezclar y este trabajo busca contribuir a un mejor entendimiento en este aspecto.

Las técnicas que dependen de la existencia de etiquetas externas no pueden ser analizadas en esta tesis dada la falta de una base de datos de dibujos etiquetada de esta manera. Adicionalmente, estas técnicas solamente pueden mezclar propiedades que se encuentran dentro de estas etiquetas, lo que lleva a un compromiso entre el nivel de detalle en las descripciones y el nivel de influencia humana impuesta sobre el modelo; esta tesis busca evitar estos problemas. El modelo *latent constraints* no es aplicable dado que su discriminador requiere que las mezclas generadas sean indistinguibles de las encontradas en la base de datos, lo cual no es compatible con la mezcla

[1]Una posibilidad es el llamado efecto *tent-pole*, en el que los vectores latentes en medio de dos vectores conocidos son menos probables de pertenecer a la distribución original, especialmente cuando la interpolación es lineal con un espacio latente de dimensionalidad alta con distribución gaussiana o uniforme (White, 2016).

intercategoría. Por ejemplo, la mezcla entre un cerdo y un gato puede ser fácilmente distinguida de un dibujo de un cerdo o un dibujo de un gato; el discriminador entonces gana fácilmente y fracasa el entrenamiento antagónico. Recientemente, ha surgido un proceso que intenta suavizar el espacio latente recompensando el modelo para tener un espacio latente que tiene interpolaciones similares a interpolaciones predefinidas. Este proceso, llamado interpolación restringida con un autocodificador de manera antagónica, utiliza un discriminador entrenado para diferenciar una muestra interpolada de una muestra tomada de una base de datos (Berthelot et al., 2018). El modelo hace la misma suposición de que las muestras generadas por la interpolación deben ser indistinguibles de muestras no interpoladas, que también lo hace incompatible con la mezcla intercategoría.

Capítulo 5

Clasificador de factores de similitud

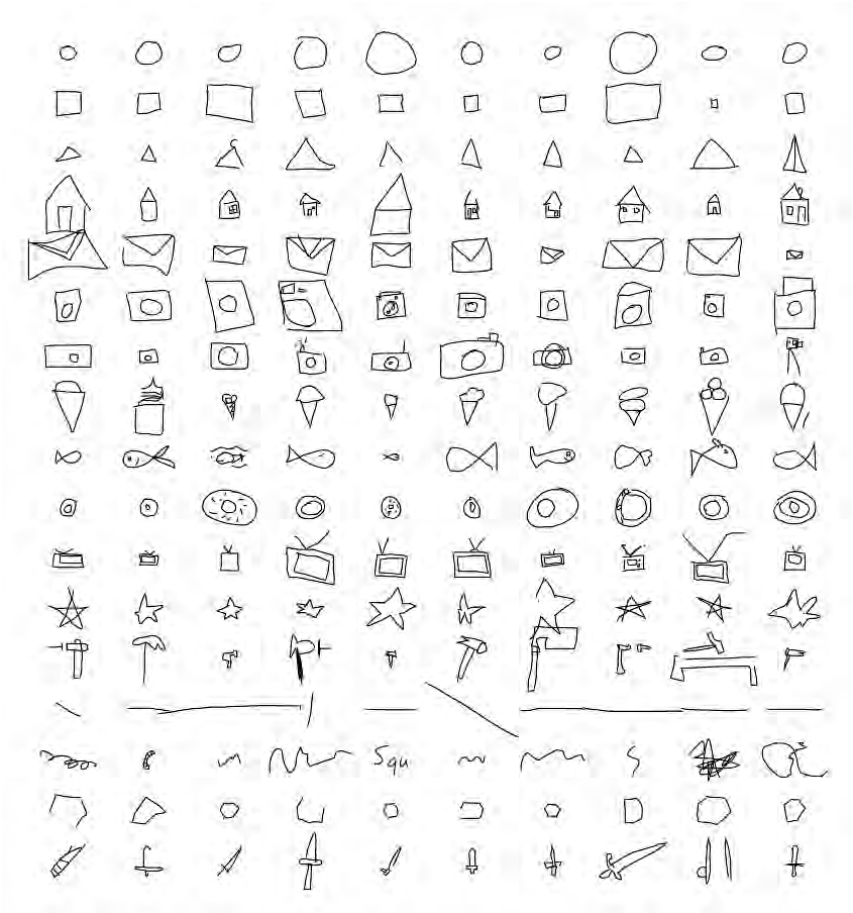


Figura 5.1: Varias categorías de dibujos hechos por humanos de la base de datos *Quick, Draw!* (Ha y Eck, 2017). En orden: círculo, cuadrado, triángulo, casa, sobre de correo, lavadora, cámara, cono de nieve, pescado, donas, televisión, estrella, martillo, línea, garabato, hexágono y espada.

En este trabajo de investigación se desarrolló un clasificador basado en redes neuronales para predecir factores de similitud entre un dibujo y un conjunto de categorías. El modelo representa una manera de identificar automáticamente si un dibujo es similar a dos o más categorías simultáneamente, lo cual es un indicativo de una mezcla según la definición presentada en la sección 1.1. Este clasificador es usado en las técnicas de mezcla visual presentadas más adelante, para proporcionar la información necesaria para permitirles aprender con menos intervención humana. El clasificador presentado en esta sección se puede entender de forma general como una RNN bi-direccional compuesta por LSTM que implementa tres conceptos: *linear gain*, un esquema de etiqueta de alto nivel y un esquema de acrecentamiento que inserta puntos aleatorios. La tarea que el modelo realiza es clasificación binaria multi-etiqueta.

5.1. Base de datos

El clasificador, así como las técnicas presentadas más adelante, trabaja con dibujos en formato de una secuencia de trazos. Estos dibujos vienen de la base de datos *Quick, Draw!*, liberada al público[1] en abril de 2017 por Google Inc. La base de datos cuenta con 50 millones de dibujos en 345 categorías realizados por 15 millones de participantes humanos. Para construir la base de datos, empleados de Google crearon una versión digital del juego de mesa *Pictionary*. En el juego, a cada participante se le presenta una palabra como «nube» o «espada» y 20 segundos para realizar un dibujo representativo en un lienzo digital implementado como una página web usando un pincel digital controlable por el ratón. Al mismo tiempo, un modelo

[1] `https://quickdraw.withgoogle.com/data`

de aprendizaje de máquinas intenta adivinar la palabra analizando el dibujo. Como en *Pictionary*, el participante gana si la computadora adivina correctamente la palabra en cuestión. La meta cooperativa del juego combinada con el tiempo limitado anima al usuario a producir dibujos sencillos y directos para cada categoría[2].

La manera digital en que recolectaron los dibujos les permitió grabar la secuencia de movimientos seguida por el participante. En la base de datos, esta información se da como una secuencia ordenada de puntos para cada dibujo, sin otra información de tiempo. Cada uno de estos puntos se da como un vector con cinco dimensiones[3]; en esta tesis, el término «trazo» se refiere a uno de estos vectores. Dos trazos pueden ser conectados visualmente por una línea negra; el término «curva» se refiere a una secuencia de trazos conectados visualmente. Una curva siempre empieza y termina con trazos que no se conectan visualmente al trazo anterior y posterior respectivamente.

La figura 5.1 presenta dibujos tomados de la base de datos en varias categorías. Aunque son sencillos, estos dibujos no son triviales. Por ejemplo, la categoría de dibujo CUADRADO, para la cual un dibujo podría ser realizado con un mínimo de cuatro líneas, consiste de un promedio de 17 trazos en la base de datos. Otras categorías más complejas, como LAVADORA, pueden tener entre 20 y 80 trazos. La razón de esto es que una línea en uno de estos dibujos está formada por muchos trazos intermediarios que al parecer fueron tomados en intervalos de tiempo regulares. Este hecho permite la representación de pequeños detalles a pesar de la sencillez de la forma general del dibujo.

[2]En comparación a otras bases de datos de dibujos como *TU-Berlin* (Eitz et al., 2012) y *Sketchy* (Sangkloy et al., 2016).

[3]Una descripción técnica sobre el formato de los datos se encuentra en el apéndice A.

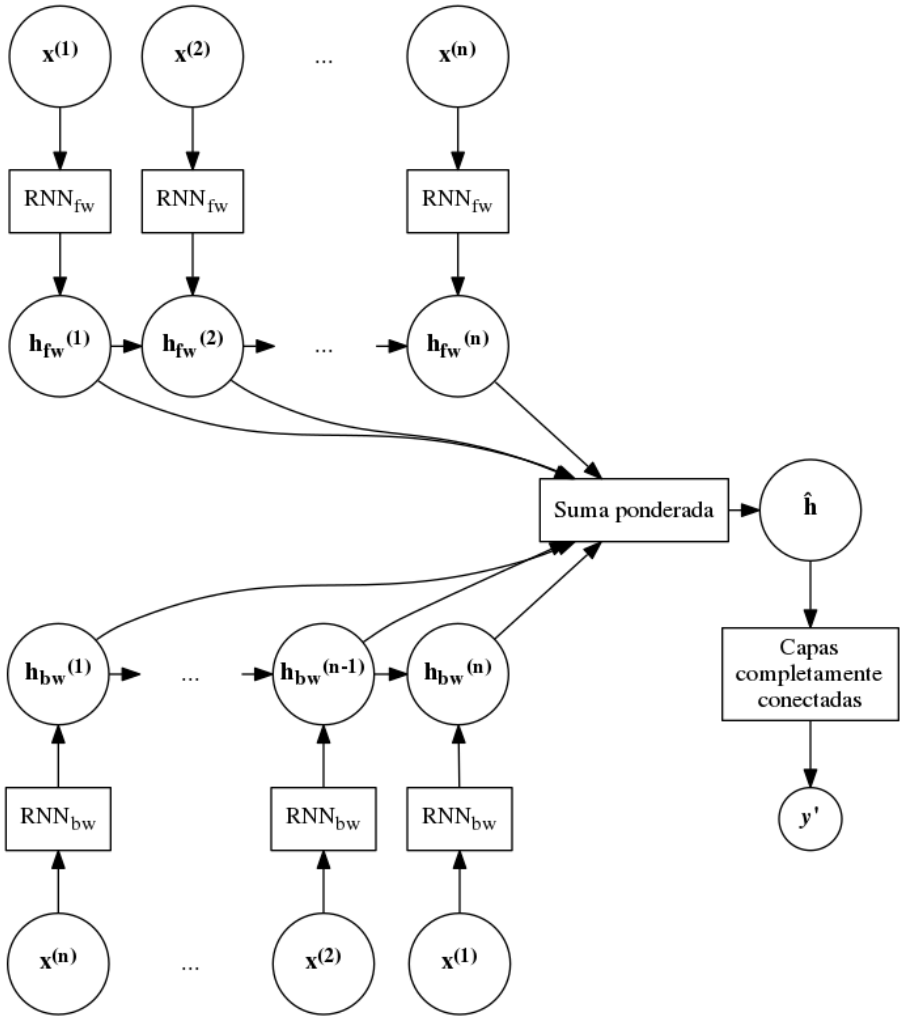


Figura 5.2: La arquitectura del clasificador. *fw* indica la RNN hacia adelante y *bw* la RNN hacia atrás.

5.2. Arquitectura

La arquitectura del clasificador consiste de dos RNN tipo LSTM, como se ilustra en la figura 5.2. La primera genera una respuesta $\boldsymbol{h}_{fw}^{(i)}$ para cada trazo i del estímulo $\boldsymbol{x}$, que es una secuencia de trazos. La segunda RNN es idéntica con respecto a su estructura, pero recibe la secuencia $\boldsymbol{x}$ en orden inverso y genera una respuesta

distinta $\boldsymbol{h}_{bw}^{(t-i)}$ para cada trazo, donde t es el número de trazos en $\boldsymbol{x}$. La respuesta $\boldsymbol{h}_{fw}^{(i)}$ se concatena con la respuesta $\boldsymbol{h}_{bw}^{(t-i)}$ para formar una respuesta combinada $\boldsymbol{h}^{(i)}$ de la siguiente manera:

$$\boldsymbol{h}^{(i)} = \text{Concat}(\boldsymbol{h}_{fw}^{(i)}, \boldsymbol{h}_{bw}^{(t-i)}) \tag{5.1}$$

donde $\boldsymbol{h}$ tiene dimensionalidad igual al número de categorías.

Para convertir estas respuestas en un vector que represente factores de similitud, la solución sencilla es usar $\boldsymbol{h}^{(t)}$, el estado después de pasar todos los trazos del dibujo por la red. En contraste, este modelo ocupa el promedio de todos los estados intermedios $\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, ..., \boldsymbol{h}^{(t)}$. Este enfoque se llama, de forma variada, *target replication* si es un promedio sencillo (Lipton et al., 2016) o *linear gain* si el promedio es ponderado dependiendo del índice del trazo[4] (Ng et al., 2015; Dai y Le, 2015). Este clasificador ocupa la última forma de la siguiente manera:

$$\begin{aligned} \text{ASUM}(t) &= \sum_{j=1}^{t} j \\ \omega^{(i)} &= \frac{i}{\text{ASUM}(t)} \\ \hat{\boldsymbol{h}} &= \alpha \sum_{i=1}^{t-1} \omega^{(i)} \boldsymbol{h}^{(i)} + (1-\alpha)\boldsymbol{h}^{(t)} \end{aligned} \tag{5.2}$$

donde t es el número de trazos en la secuencia y α es un parámetro que controla el peso de los estados intermediarios $\boldsymbol{h}^{(1)}, ..., \boldsymbol{h}^{(t-1)}$ con respecto al estado final $\boldsymbol{h}^{(t)}$. Finalmente, una capa de neuronas completamente conectada con función de activación sigmoidea convierte $\hat{\boldsymbol{h}}$ en un vector de salida $\hat{\boldsymbol{y}}$. La diferencia entre $\hat{\boldsymbol{y}}$ y el vector

[4]Estos trabajos han reportado mejoras con respecto al exactitud usando el promedio de los estados en los dominios de música, vídeos y diagnóstico médico, aunque ninguno ha aplicado la técnica a dibujos.

de similitud esperado por $\boldsymbol{x}$ se calcula usando la función de pérdida de la entropía cruzada binaria multi-etiqueta presentada en la ecuación 3.4. Las funciones que definen las RNN y las capas de neuronas son derivables y por lo tanto el modelo puede ser entrenado usando gradiente descendente estocástico con retropropagación.

Esquema de etiquetas de alto nivel. El vector de similitud esperado por cada categoría $\boldsymbol{y}$, se compone de factores de similitud que pueden tener uno de tres valores[5]. Si la categoría α es totalmente similar a la categoría β, el factor de similitud esperado por un dibujo de α a β será 1. Si no existe ninguna similitud, el factor esperado será 0. Sin embargo, si el dibujo es de alguna manera similar pero a la vez disimilar, el factor esperado será 0.5. Los varios factores no son mutuamente excluyentes, siguiendo el enfoque de la clasificación multi-etiqueta. Es decir, un dibujo puede ser parcialmente o totalmente similar a múltiples categorías a la vez.

Acrecentamiento de datos. El clasificador también ocupó un nuevo tipo de acrecentamiento de datos: la inserción de una curva compuesta de trazos aleatorios. La secuencia de trazos fue generado tomando muestras repetidamente de una distribución $\mathcal{N}(0, 1)$. Esta curva fue insertada en un lugar aleatorio entre otras curvas y ayudó al clasificador a ser más robusto a secuencias de puntos no relevantes.

Hasta donde llega nuestro conocimiento, el modelo presentado en esta sección es el primero en aplicar el aprendizaje de máquinas para asignar a dibujos, factores de

[5]El concepto de entrenar una red neuronal usando lógica difusa fue originalmente presentado por Pal y Mitra (1992). El uso de un valor escalar en la ecuación de la entropía cruzada binaria tiene precedente: el suavizado de etiquetas (Szegedy et al., 2016)

	Categoría				
Clasificador	A	B	C	D	E
1	Casa	Lavadora	Televisión	Dona	Montaña.
2	Cotorra	Camión	Pie	Bicicleta	Hacha.
3	Buzón	Bolsa	Almohada	Arco Iris	Marcador.

Cuadro 5.1: Categorías usadas para entrenar los clasificadores.

similitud de múltiples categorías simultáneamente[6].

5.3. Validación

Se entrenaron tres clasificadores idénticos con un conjunto de cinco categorías distintas cada uno. Las categorías elegidas se encuentran en el cuadro 5.1. Se asignó a cada categoría un vector de similitud que describe su relación a las otras categorías en su conjunto. Se eligieron los tres conjuntos de categorías y sus vectores de similitud para corresponder con los tres experimentos presentados más adelante en la sección 7.2.1. Se configuró cada clasificador con hiperparámetros inspirados en el modelo *sketch-rnn*[7]. Se entrenó cada clasificador con un conjunto de 20,000 dibujos para cada categoría o 100,000 en total hasta que la pérdida del clasificador se estabilizó. Todos los dibujos usados vinieron de la base de datos *Quick, Draw!* descrita anteriormente.

Se evaluó el clasificador con respecto a su capacidad para identificar correctamente la categoría de un dibujo nunca visto anteriormente. Para convertir el vector de similitud generado por el clasificador a una predicción de una categoría, se calculó

[6]Para imágenes en formato de imagen de píxeles existen algunos trabajos que realizan la clasificación multi-etiqueta como el de Boutell et al. (2004), mientras Koch et al. (2015) presentó una arquitectura que mide la similitud entre imágenes.

[7]El apéndice B resume los valores exactos utilizados, así como los datos de entrenamiento.

	Categoría				
Clasificador	A	B	C	D	E
1	Granero	Cámara	Marco	Galleta	Carpa
2	Cotorra	Camión	Pie	Bicicleta	Hacha
3	Martillo	Cubeta	Microondas	Audífonos	Lápiz

Cuadro 5.2: Categorías alternativas.

Clasificador	Categorías	Precisión	Exhaustividad	F1
1		.987	.987	.987
2	Entrenamiento	.966	.966	.966
3		.966	.965	.965
1		.878	.869	.873
2	Alternativas	.762	.751	.756
3		.779	.739	.758

Cuadro 5.3: Datos de validación para los clasificadores.

la diferencia simple entre el vector de similitud predicho y los vectores asignados de antemano a cada categoría y se tomó la categoría con diferencia menor como la predicción del modelo. Se calculó la precisión, exhaustividad y puntuación F1[8] de estas predicciones usando 200 dibujos no vistos anteriormente por categoría o 1,000 en total. Esta evaluación proporciona una medida de la capacidad del clasificador de reconocer dibujos nuevos de categorías conocidas. Para fortalecer la validación, también se evaluó el clasificador usando dibujos de categorías desconocidas. Para hacer esto, se introdujo un conjunto de categorías nuevas en el cual cada categoría es similar en nuestra opinión subjetiva a una de las originales; el cuadro 5.2 muestra estas selecciones.

El cuadro 5.3 reporta los resultados con los datos de evaluación de los tres clasificadores. Los valores para las métricas de precisión, exhaustividad y F1 son altos:

[8]Estas métricas se encuentran en uso común para evaluar la tarea de clasificación (Powers, 2011; Engel et al., 2017).

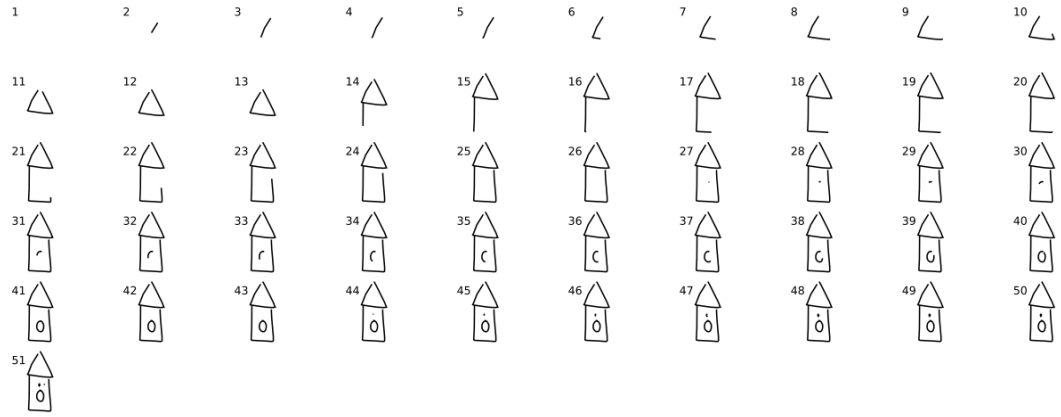


Figura 5.3: Un dibujo siendo construido trazo por trazo.

arriba de 0.96 en todas las métricas. Dado que en la gran mayoría el clasificador identificó correctamente la categoría del dibujo, estos valores se consideran adecuados para la tarea de clasificar dibujos. Los resultados de los datos alternativos fueron peores, aunque es posible explicar este hecho por las diferencias entre las categorías originales y alternativas. No obstante, ningún clasificador obtuvo calificación menor que 0.75 en cualquier métrica. Estas evaluaciones proporcionan evidencia sobre la capacidad del modelo clasificador para reconocer dibujos similares a las categorías deseadas.

Un análisis subjetivo del comportamiento del modelo en el tiempo presenta más evidencia de su habilidad para reconocer mezclas. Este análisis se puede realizar mediante una visualización del comportamiento del clasificador a cada paso de tiempo. La figura 5.3 contiene un ejemplo de un dibujo siendo construido trazo por trazo; consideramos que el dibujo representa una mezcla visual entre las categorías CASA y LAVADORA. La figura 5.4 ilustra la respuesta del clasificador después de analizar cada trazo del mismo dibujo. Es evidente que al aparecer un triángulo en el dibujo

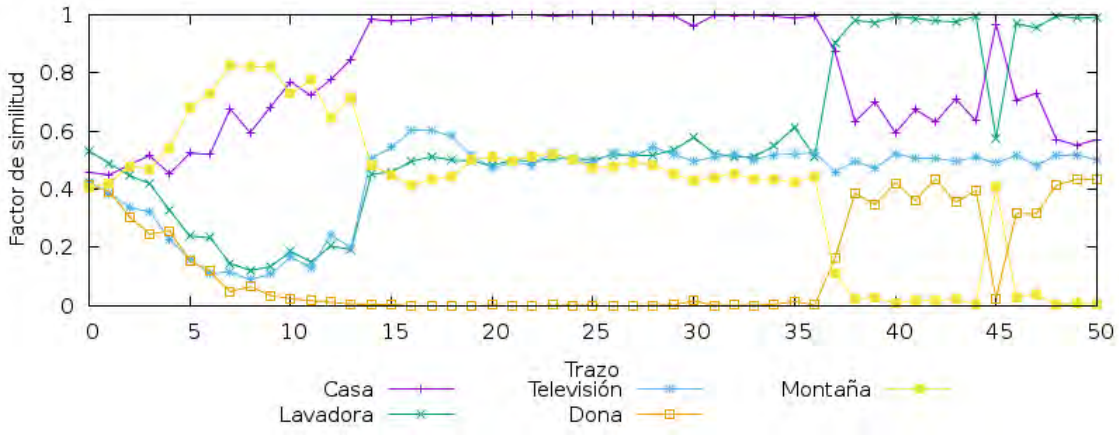


Figura 5.4: Análisis del dibujo en la figura 5.3 usando el clasificador propuesto.

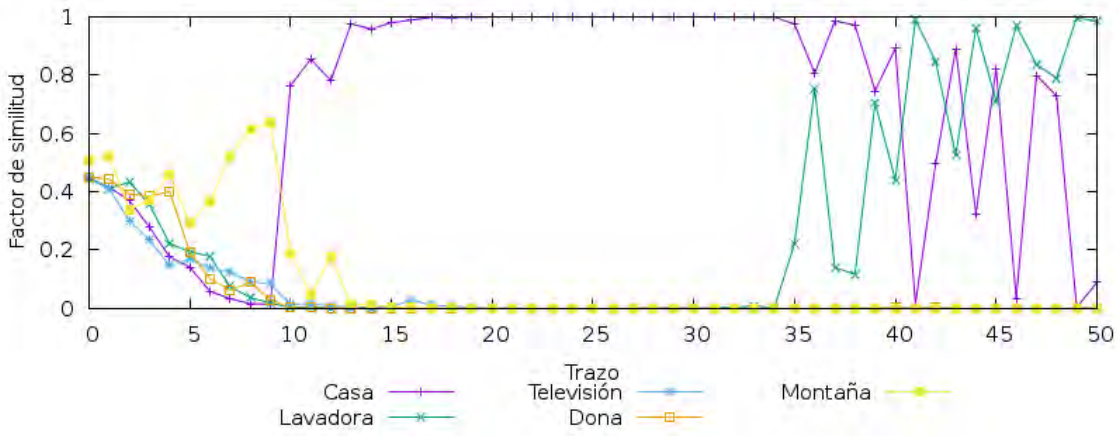


Figura 5.5: Análisis del dibujo en la figura 5.3 usando un modelo básico.

el clasificador responde con factores altos para las categorías CASA y MONTAÑA, que contienen formas triangulares. Al agregar un círculo, el clasificador eleva los factores de las categorías CASA y a LAVADORA sobre los de las otras categorías. Tenga en cuenta que el vector de similitud para CASA define factores de 0.5 para las categorías TELEVISIÓN, MONTAÑA y LAVADORA y un factor de 0 para DONA; la sección contiene más información 7.2.1.

En contraste, la figura 5.5 muestra cómo un clasificador sin *linear gain* y sin

etiquetas de lógica difusa no reconoce cuando el dibujo demuestra características de mezclas. En particular, al modelo le falta reconocer el hecho de que el dibujo parcial es similar a TELEVISIÓN y MONTAÑA y que el dibujo completo es también de alguna manera similar a DONA.

En general, las métricas de precisión y exhaustividad señalan la capacidad del modelo para reconocer similitudes entre categorías de dibujos conocidas y no conocidas. En nuestra opinión, las gráficas adicionales agregan evidencia visual de que el clasificador es capaz para reconocer cuando un dibujo es similar a múltiples categorías a la vez.

Capítulo 6

Mezcla visual por búsqueda

En el capítulo 4 se presentó evidencia para apoyar la hipótesis de que el espacio latente del modelo *sketch-rnn* entrenado contiene dibujos con propiedades de la mezcla visual a realizar. También se mencionó que en la práctica, estas técnicas no generan mezclas frecuentemente cuando la tarea es intercategoría. Dado un modelo clasificador, como el que se acaba de presentar, debe ser posible buscar más eficientemente en este espacio los dibujos que tengan una combinación específica de factores de similitud deseable; en este capítulo, se refiere a una de estas combinaciones por el término «vector de similitudes deseadas» o simplemente $\boldsymbol{v}_{des}$. Si $\boldsymbol{v}_{des}$ está construido para definir factores de similitud altos en exactamente dos categorías y factores de similitud menores para las otras, los dibujos encontrados deberían parecerse visualmente a dos categorías a la vez, lo cual puede ser interpretado como una mezcla intercategoría[1]. Las secciones a continuación presentan tres nuevas técnicas para buscar estos dibujos usando *sketch-rnn* como el modelo generativo y el clasificador como guía. La técnica de filtración pone a prueba la suposición de que existen dibujos que pueden considerarse mezclas en el espacio latente del modelo generativo. La búsqueda simple propone una manera más eficiente para buscar estos dibujos. Finalmente, la búsqueda ancha propone una modificación a la búsqueda simple para buscar una variedad más grande de dibujos.

[1]La composición exacta del vector $\boldsymbol{v}_{des}$ depende del experimento a realizar; la sección 7.2.3 presenta experimentos.

6.1. Filtración

La primera técnica implementa un tipo de filtración muy sencillo en el cual se ordena un conjunto aleatorio de dibujos, quedándose con los que obtuvieron vectores de similitud más similares a $\boldsymbol{v}_{des}$. Para el primer paso de la filtración, el modelo *sketch-rnn* se entrena con dos categorías de dibujos a la vez y de forma aleatoria se obtienen varias muestras de su espacio latente[2]. Dado que el modelo es explícitamente incentivado por la pérdida Kullback-Leibler (KL) para desarrollar un espacio latente organizado de tal forma que siga una distribución $\mathcal{N}(0, 1)$, este proceso es tan simple como la toma aleatoria de vectores de esta misma distribución. Estos vectores latentes son convertidos en dibujos usando el modelo *sketch-rnn* entrenado.

Para el segundo paso, un modelo clasificador se entrena para reconocer factores de similitud para varias categorías, una para cada dimensión de $\boldsymbol{v}_{des}$. El clasificador entrenado evalúa los dibujos aleatorios para asignar vectores de similitud predichos $\boldsymbol{v}$ para cada uno. Finalmente, los dibujos con $\boldsymbol{v}$ más similar a $\boldsymbol{v}_{des}$ son seleccionados mientras que los otros son descartados. La comparación puede ser realizada con cualquier función; en este trabajo se ocupa una versión ponderada de la entropía cruzada binaria multi-etiqueta presentada más adelante en la sección 7.2.3. Hasta donde tenemos conocimiento, esta es la primera técnica que usa un clasificador autónomo de este tipo para filtrar dibujos que representan mezcla visual.

[2]Observe que para entrenar este modelo, dibujos de ambas categorías son proporcionados sin diferenciar entre los dos. No hay un control que pueda usarse para especificar la categoría del dibujo generado durante la reconstrucción. La teoría es que al entrenar el modelo para reconstruir dos categorías de dibujos, su espacio latente contendría muestras que representan la mezcla visual entre estas dos categorías.

6.2. Búsqueda sencilla por SGD

Una observación sobre el espacio latente de un modelo *sketch-rnn* es que su tamaño es sumamente grande[3], por lo que la filtración puede ser ineficaz. La tarea de buscar dibujos en este espacio multi-dimensional se puede formular como un problema de optimización multi-objetivo, en el cual cada factor de similitud es un objetivo distinto. En este caso, el algoritmo de gradiente descendente puede ser aplicado para encontrar el vector latente que obtenga $\boldsymbol{v}$ más similar a $\boldsymbol{v}_{des}$ cuando es evaluado por el clasificador del capítulo anterior. Más que encontrar un solo vector, el algoritmo puede ser aplicado para buscar una región entera de vectores latentes con esta propiedad. Si la región es suficientemente grande, el modelo puede generar nuevos dibujos simplemente tomando muestras de esta región.

La figura 6.1 muestra una técnica de búsqueda que implementa este enfoque. Antes de empezar el proceso de entrenamiento, un vector de medias $\boldsymbol{\mu}$ y un vector de desviaciones estándar $\boldsymbol{\sigma}$ son elegidos al azar como un punto de partida. Estos dos vectores parametrizan una distribución gaussiana, de la cual se toma un vector aleatorio. El vector se convierte en un dibujo por un modelo *sketch-rnn*, entrenado anteriormente en las categorías a mezclar. Este dibujo es evaluado por el clasificador descrito anteriormente, generando un vector de similitud predicho $\boldsymbol{v}$. Este vector es comparado con $\boldsymbol{v}_{des}$ usando la entropía cruzada binaria multi-etiqueta ℓ_{XE} definida en la ecuación 3.4 como la técnica de filtración.

Además de la pérdida de la entropía, se agrega al paso de entrenamiento una

[3]Para un espacio latente con de 128 dimensiones (Ha y Eck, 2017), hay aproximadamente 10^{128} dibujos distintos, aunque hay razón para creer que los VAE solamente utilizan una porción pequeña de sus dimensiones (Engel et al., 2017). En la práctica el número de dibujos visualmente distintos puede ser unos órdenes de magnitud menor.

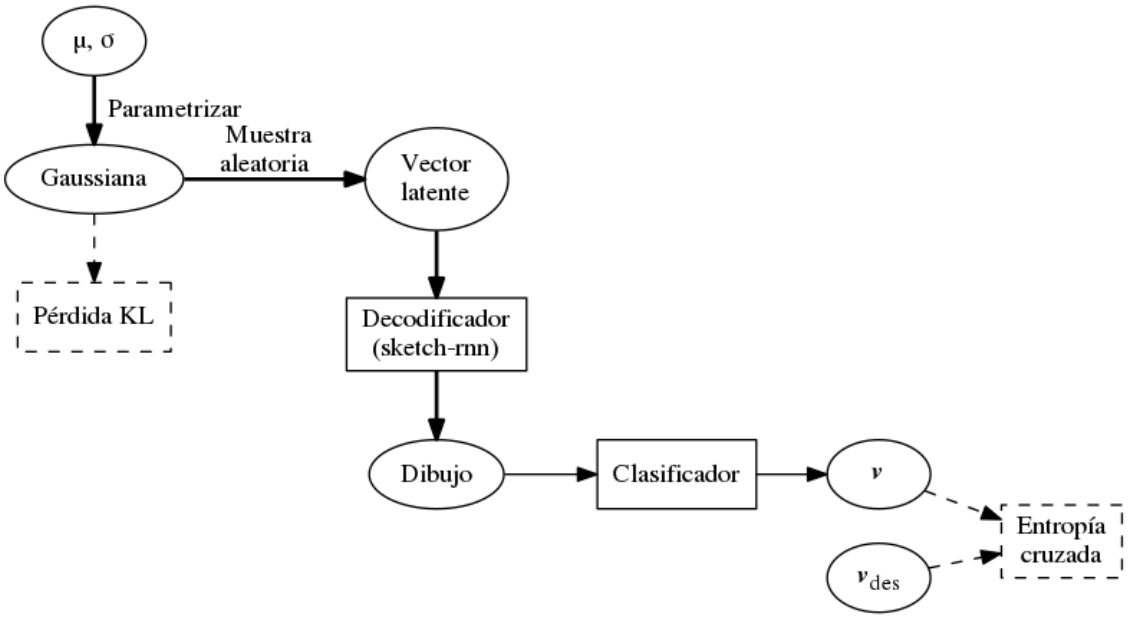


Figura 6.1: Arquitectura de la técnica de búsqueda sencilla. Los rectángulos sólidos representan redes neuronales artificiales, mientras que los rectángulos punteados representan funciones de pérdida. El camino resaltado representa los pasos necesarios para la generación una vez que el modelo ha sido entrenado.

función adicional ℓ_{KL}, la pérdida de la divergencia KL descrita en la ecuación 3.10. Esta función promueve que la técnica de búsqueda produzca dibujos más verosímiles[4] que se alejan de lugares en el espacio latente en los cuales el clasificador puede ser engañado por dibujos visualmente no interesantes[5]. La función de pérdida final para esta técnica se define por la siguiente ecuación:

$$\text{Pérdida} = \ell_{XE} + \omega_{KL}\ell_{KL} \tag{6.1}$$

donde ω_{KL} es un peso asignado a la pérdida KL. El gradiente de la función de pérdida

[4]Ha y Eck (2017) detallan el efecto de la pérdida KL a la llamada coherencia de los dibujos generados por *sketch-rnn* en su artículo.

[5]La facilidad con que una red neuronal clasificadora es engañada fue explorado con más profundidad por Nguyen et al. (2015).

es calculado[6] con respecto al parámetro $\boldsymbol{\mu}$ y también con respecto al parámetro $\boldsymbol{\sigma}$, y estos dos valores son actualizados siguiendo el gradiente más inclinado como en el enfoque de SGD. El proceso se repite por un número predeterminado de iteraciones. Para generar dibujos, se puede simplemente tomar vectores latentes de la distribución gaussiana definida por $\boldsymbol{\mu}$ y $\boldsymbol{\sigma}$; estos vectores latentes se convierten en dibujos usando el mismo modelo *sketch-rnn*.

Hasta donde llega nuestro conocimiento, esta técnica de búsqueda es la primera en usar SGD y un clasificador de este tipo para buscar en el espacio latente de un VAE por vectores latentes que corresponden a objetos representativos de una mezcla intercategoría.

Amortizador

Un problema técnico surge con la manera en que el decodificador convierte un vector latente en una secuencia, que no es derivable y por lo tanto no es compatible con el algoritmo de retropropagación. En específico, este paso requiere la toma iterativa de una secuencia de muestreos usando el decodificador, para la cual no existe un gradiente. Aún si esto no fuera el caso, el proceso iterativo de convertir un vector latente en una secuencia de trazos es muy tardado. Una solución es la aplicación de una técnica llamada «amortización» en la cual una red neuronal es entrenada para simular el funcionamiento del proceso no derivable. Como notó el investigador Engel et al. (2017), hay muchos ejemplos de la aplicación de esta técnica en el aprendizaje de máquina, incluyendo el generador de una GAN y el codificador de un VAE.

La figura 6.2 presenta la arquitectura del modelo amortizador. La rama inferior ilustra el proceso original, en el cual un vector latente aleatorio es convertido en un

[6]El gradiente de la pérdida KL se define en Kingma y Welling (2014).

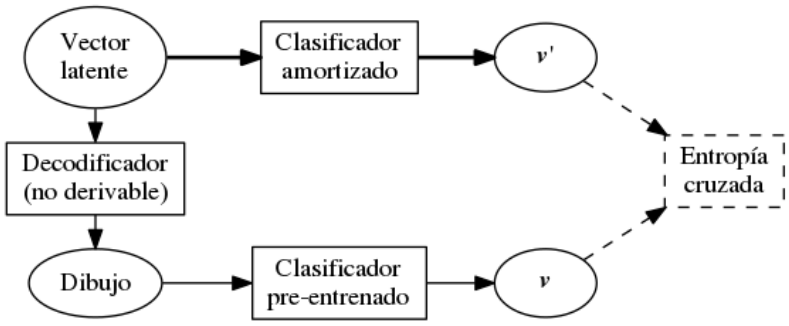


Figura 6.2: Arquitectura del amortizador. Los rectángulos representan una red neuronal artificial. Las redes de la rama inferior vienen ya entrenadas, así que durante este proceso solamente se entrena la red de la rama superior.

dibujo por el decodificador *sketch-rnn* y posteriormente evaluado por el clasificador para generar un vector de similitud. La rama superior con líneas resaltadas ilustra el amortizador, en el cual un clasificador nuevo, formado por capas de redes neuronales artificiales completamente contactadas con activación sigmoidea, aprende a generar el mismo vector de similitud usando solamente la información del vector latente. La media de la entropía cruzada binaria multi-etiqueta de la ecuación 3.4 entre los dos vectores es calculada y el gradiente del resultado es usado para entrenar el clasificador nuevo como en el enfoque de SGD. Una vez entrenado, los componentes de la rama superior del modelo funcionan de forma independiente de los de la rama inferior para producir un vector de similitud directamente del vector latente, así amortizando el clasificador.

6.3. Búsqueda ancha por SGD

La técnica de búsqueda ancha ilustrada en la figura 6.3 es una extensión de la técnica de búsqueda sencilla. Mientras la técnica anterior parametriza una sola

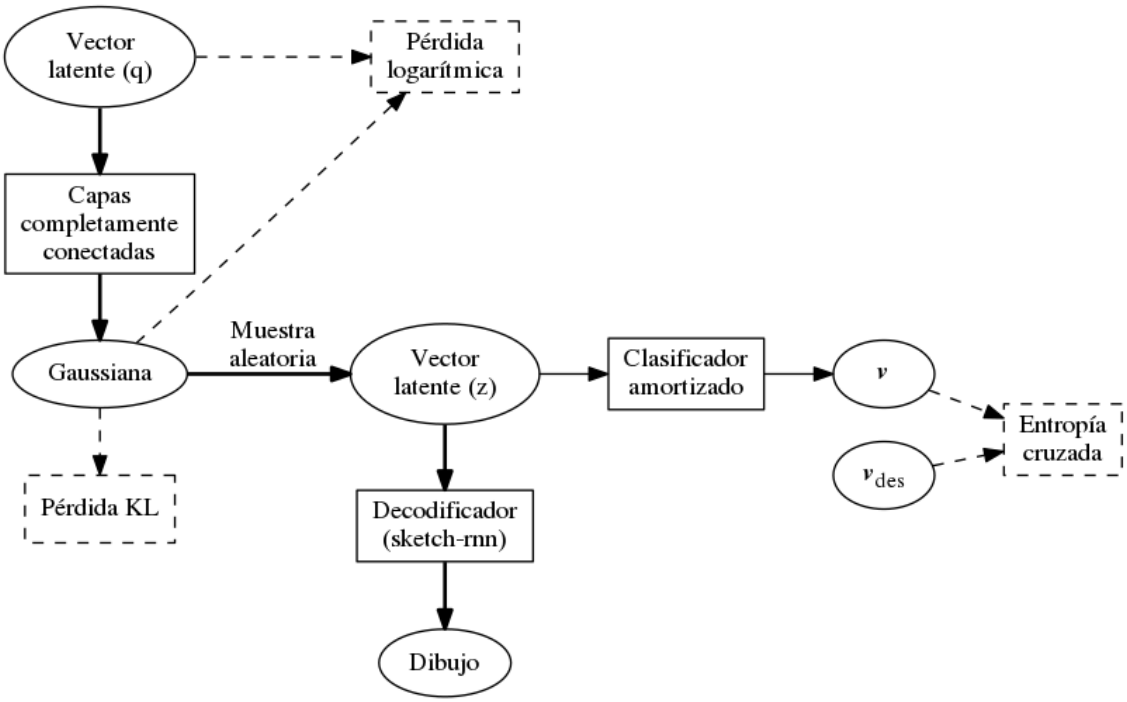


Figura 6.3: Arquitectura de la técnica de búsqueda ancha. Los rectángulos sólidos representan redes neuronales artificiales, mientras que los rectángulos punteados representan funciones de pérdida. El camino resaltado representa los pasos necesarios para la generación una vez que el modelo ha sido entrenado.

distribución gaussiana, esta técnica introduce capas de redes neuronales para generar una distribución gaussiana personalizada a un estímulo. Este estímulo viene en forma de un vector latente $\boldsymbol{q}$ tomado de una distribución $\mathcal{N}(0, 1)$ que tiene la misma dimensionalidad que el espacio latente del modelo *sketch-rnn* Por lo tanto, la técnica puede ser descrita conceptualmente como la búsqueda de una manera para transformar cualquier $\boldsymbol{q}$ para que sea más probable que corresponda a una mezcla visual.

El vector $\boldsymbol{q}$ pasa por múltiples capas completamente conectadas con función de activación sigmoidea que lo convierten en dos vectores, uno que contiene medias y otro desviaciones estándar. Una distribución gaussiana es construida usando estos

dos vectores como sus parámetros y una segunda muestra es tomada de ella para generar un vector latente $\boldsymbol{z}$ de la misma dimensionalidad que $\boldsymbol{q}$. El vector nuevo $\boldsymbol{z}$ es pasado al clasificador, amortizado para que el modelo sea derivable, que le asigna un vector de similitud $\boldsymbol{v}$. Este vector es comparado con $\boldsymbol{v}_{des}$ usando la entropía cruzada binaria multi-etiqueta ℓ_{XE} de la ecuación 3.4 para generar un término de pérdida.

El modelo también utiliza dos funciones de pérdida adicionales. La primera es nuevamente la pérdida KL de la ecuación 3.10 que promueve que la media y desviación estándar generadas por las capas de neuronas sean cercanos a 0 y 1, respectivamente. La segunda función es la verosimilitud logarítmica o *log-likelihood* de $\boldsymbol{q}$ dada la distribución gaussiana generada por las capas completamente conectadas:

$$\ell_P = -\frac{1}{n}\sum_{i=0}^{n}\frac{(q_i - \mu_i)^2}{2\sigma_i^2} - \frac{\log(2\pi)}{2} - \log(\sigma_i) \tag{6.2}$$

donde n es la dimensionalidad de $\boldsymbol{q}$ y π es la constante matemática pi. Esta función promueve que la media y la desviación estándar sean personalizadas a cada vector aleatorio dado $\boldsymbol{q}$, lo cual causa que la variedad de los vectores $\boldsymbol{z}$ se acerque a la de los vectores $\boldsymbol{q}$. La función de pérdida usada para entrenar el modelo es una combinación de estas tres funciones:

$$\text{Pérdida} = \ell_{XE} + \omega_{KL}\ell_{KL} + \omega_P\ell_P \tag{6.3}$$

donde ω_{KL} y ω_P son pesos asignados a la divergencia KL y verosimilitud logarítmica, respectivamente. El modelo se entrena con el algoritmo de retropropagación y SGD usando el enfoque estándar. Una vez entrenado, cualquier vector $\boldsymbol{q}$ puede ser pasado por la red para generar un nuevo dibujo.

Los pasos de esta técnica fueron inspirados en múltiples fuentes. La traducción de vector a vector usando neuronas artificiales fue inspirado por las GAN y en específico el modelo *latent constraints* (Engel et al., 2017). En contraste con su trabajo, la técnica de búsqueda ancha reemplaza el discriminador de las GAN por el clasificador y no implementa un entrenamiento antagónico. Su trabajo también asume que las muestras generadas deben ser similares a las de la base de datos, un hecho que es incompatible con la tarea de realizar la mezcla intercategoría. El uso de una pérdida de verosimilitud logarítmica fue inspirada en el trabajo original de Graves (2013) sobre la generación probabilística de secuencias. Hasta donde tenemos conocimiento, esta es la primera técnica que utiliza neuronas artificiales y un clasificador de este tipo para buscar en el espacio latente de un VAE por vectores latentes que corresponden a objetos demostrativos de una mezcla intercategoría.

Capítulo 7

Experimentos

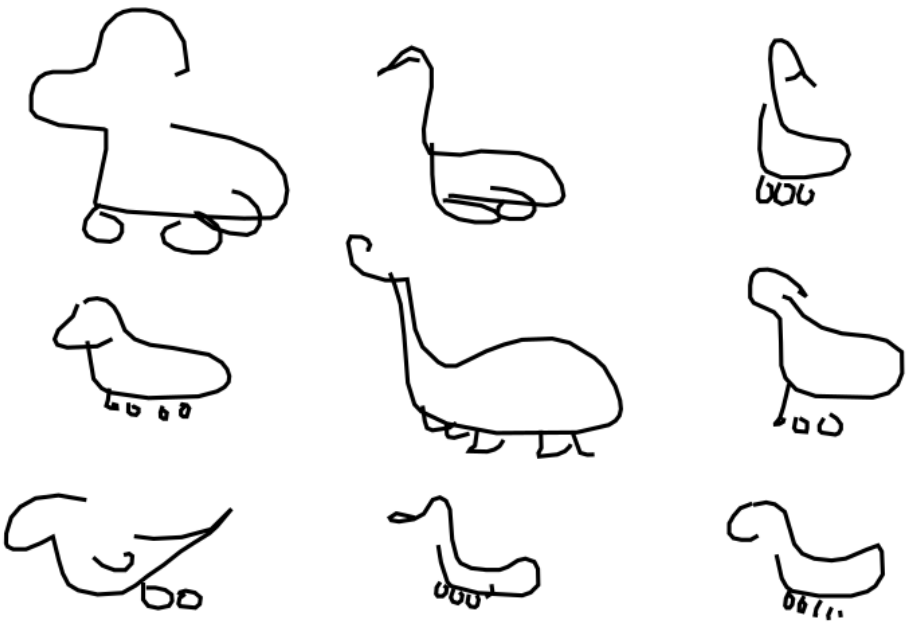


Figura 7.1: Dibujos seleccionados de patos con ruedas de patín, encontrados por la técnica de búsqueda ancha y realizados por un modelo *sketch-rnn* entrenado solamente para generar patos o patines por separado.

7.1. Metodología

Como se mencionó en la sección 1.4, este trabajo parte principalmente de la hipótesis de que es posible desarrollar una técnica que integre información de un modelo evaluativo para remodelar la salida de un modelo generativo basado en redes neuronales profundas. En este trabajo, el modelo generativo es *sketch-rnn* y el modelo evaluativo es el clasificador. Para poner a prueba esta hipótesis, se entrenaron las nuevas técnicas propuestas de filtración, búsqueda simple y búsqueda ancha usando retroalimentación del mismo clasificador. Para evaluar los dibujos generados, se ocupó un clasificador con la misma arquitectura pero entrenado con dibujos distintos a los usados para las técnicas generativas. Los vectores de similitud asignados a los dibujos por el clasificador fueron analizados con respecto al nivel con que describen mezcla visual según la definición presentada en la sección 1.1. Para establecer una base de comparación, las técnicas de interpolación, reconstrucción y autocompletado también fueron evaluadas de la misma manera. En este capítulo, el término «experimento» se refiere a los pasos de entrenar los modelos que se acaban de mencionar, generar dibujos de muestra y evaluar estos dibujos. Finalmente, se realizó una encuesta a personas sobre la percepción que tienen de los dibujos generados por estas técnicas en relación a si corresponden a mezclas visuales o no.

	Categorías							
Rol	Exp. 1	Exp. 2	Exp. 3	Vector de similitud				
Principal	Casa	Pato	Buzón	1	.5	.5	0	.5
Principal	Lavadora	Patín	Bolsa	.5	1	.5	.5	0
Secundaria	Televisión	Calcetín	Almohada	.5	.5	1	0	0
Secundaria	Dona	Lentes	Arcoiris	0	.5	0	1	0
Secundaria	Montaña	Taladro	Marcador	.5	0	0	0	1

Cuadro 7.1: Categorías y vectores de similitud para los tres experimentos.

7.2. Configuración

7.2.1. Mezclas experimentales

Se realizaron tres experimentos con tres mezclas distintas. Cada experimento se trata de una mezcla entre solamente dos categorías principales para simplificar la tarea. Estas categorías se eligieron de tal forma que tuvieron alguna similitud entre ellas; se esperaba que esto ayudara al modelo *sketch-rnn* a generar un espacio latente más suave, dado que la similitud proporciona una zona de transición natural. Además, se eligieron tres categorías secundarias para evitar que el clasificador sea engañado con dibujos que obtienen factores de similitud[1] altos pero visualmente solo parecen garabatos. Se eligió la primera categoría secundaria para ser similar a las dos categorías principales, mientras que las otras dos para ser similares a exactamente una de las categorías principales pero al mismo tiempo no ser similar a ninguna otra categoría secundaria. A continuación se presentan ejemplos visuales y descripciones de las categorías elegidas. Las descripciones en esta sección representan nuestra percepción subjetiva, pero creemos que son en general intuitivas.

[1] Se definió el término «factor de similitud» en el capítulo 5.

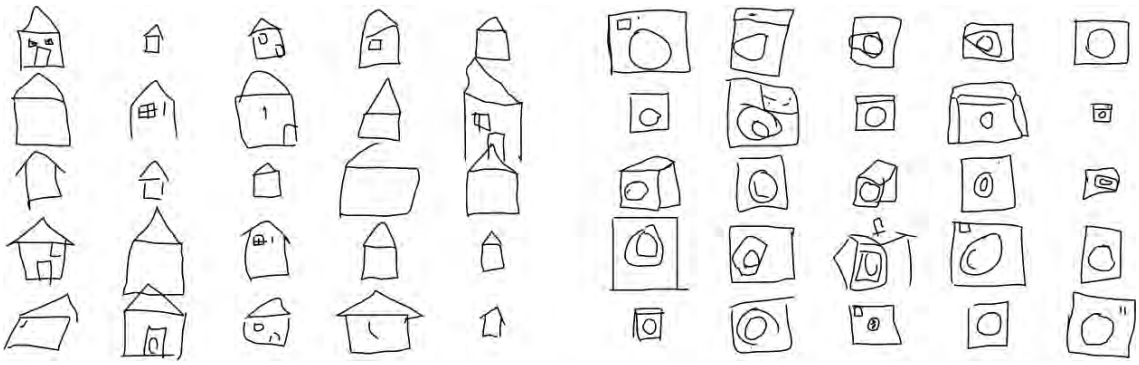


Figura 7.2: Dibujos de casas (a la izquierda) y lavadoras (a la derecha) hechos por humanos, tomados al azar de la base de datos *Quick, Draw!* (Ha y Eck, 2017).

Experimento 1. Este experimento se trata de la mezcla entre las categorías CASA y LAVADORA. En la base de datos, las casas se caracterizan por una forma cuadrada coronada por una forma triangular. A veces incluyen dentro del cuadrado una puerta que toca el suelo y una ventana rectangular que no. En cambio, las lavadoras se caracterizan por una forma cuadrada con un círculo grande en su centro. A veces contienen un botón arriba del círculo. La figura 7.2 presenta dibujos de ambas categorías tomados de la base de datos. Se eligió TELEVISIÓN como la primera categoría secundaria, debido a su forma cuadrada que se comparte con las casas y las lavadoras, DONA por su forma circular similar a la puerta de la lavadora y MONTAÑA por su forma triangular similar al techo de la casa. Los vectores de similitud definidos en el cuadro 7.1 reflejan estas creencias.

Experimento 2. Este experimento se trata de la mezcla entre las categorías PATO y PATÍN. En la base de datos, los dibujos de patos se caracterizan por un cuerpo curvado en forma de codo y una cabeza con un pico. Frecuentemente cuentan con un par de piernas debajo del cuerpo. Los dibujos de los patines se caracterizan por un cuerpo que también es curvado. La característica más llamativa de los patines es

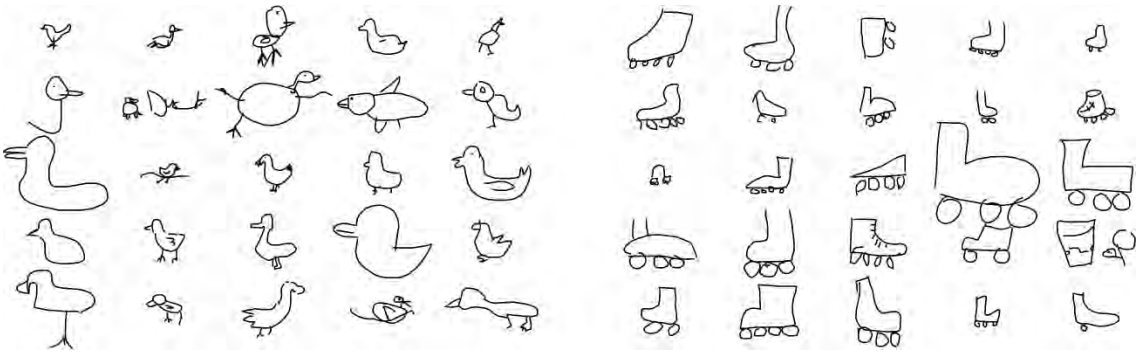


Figura 7.3: Dibujos de patos a la izquierda y patines a la derecha hechos por humanos, tomados al azar de la base de datos *Quick, Draw!* (Ha y Eck, 2017).

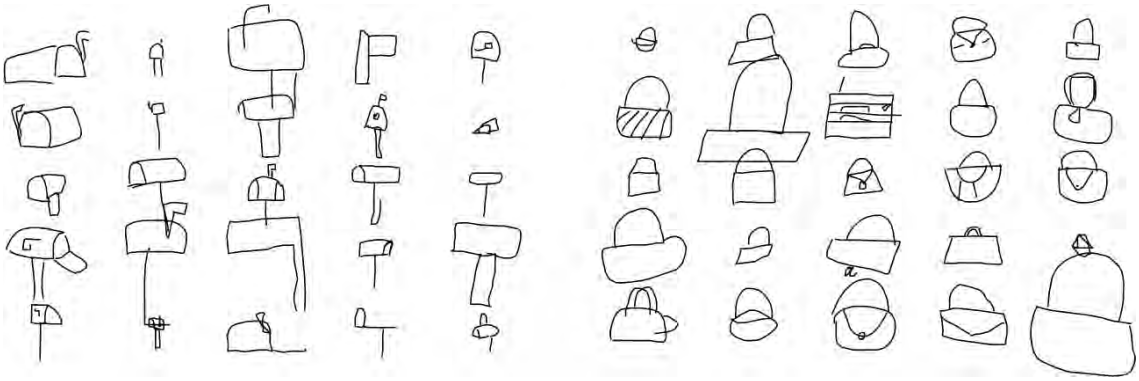


Figura 7.4: Dibujos de buzones a la izquierda y bolsas a la derecha hechos por humanos, tomados al azar de la base de datos *Quick, Draw!* (Ha y Eck, 2017).

la inclusión de dos o hasta cuatro ruedas debajo del cuerpo. La figura 7.3 muestra algunos ejemplos hechos por humanos. Las categorías adicionales elegidas en este experimento son CALCETÍN debido a su forma de bota, LENTES por tener dos círculos parecidos a ruedas y TALADRO por tener un pico como el pato.

Experimento 3. Este experimento se trata de la mezcla entre las categorías BUZÓN y BOLSA. En la base de datos, los buzones se caracterizan por una forma rectangular colocada arriba de un palo o poste. De vez en cuando los dibujos

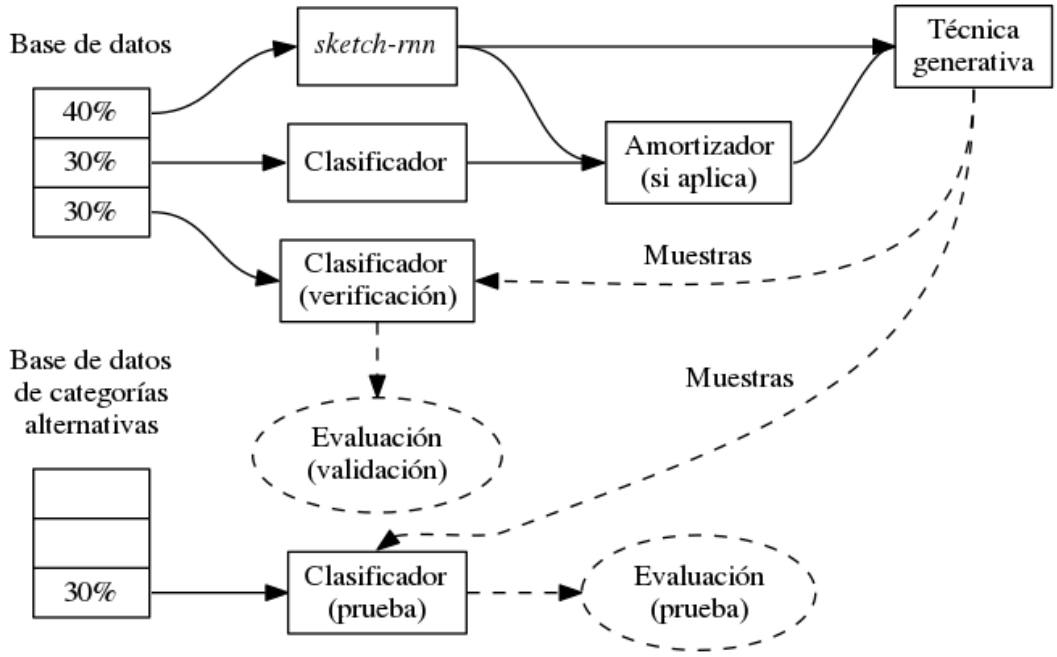


Figura 7.5: Diagrama de cómo se repartieron los dibujos de entrenamiento entre los varios modelos. La línea punteada representa el flujo de dibujos después de terminar todo tipo de entrenamiento.

incorporan una bandera rígida, la cual regularmente sobresale por arriba. Los dibujos de bolsas también se caracterizan por una forma rectangular, solo que ahora tienen un arco o semicírculo en la parte superior. Se eligió ALMOHADA como la primera categoría secundaria debido a su forma básicamente rectangular pero un poco redondeada a la vez, como el buzón y la bolsa. También se eligieron ARCOIRIS debido a su similitud a las correas de la bolsa y MARCADOR debido a su apariencia de un palo derecho.

7.2.2. Entrenamiento

La base de datos *Quick, Draw!*, presentada en la sección 5.1, cuenta con 70,000 dibujos por cada categoría. Se dividieron estos dibujos en tres conjuntos distintos

que se usaron para entrenar el modelo *sketch-rnn*, un clasificador usado para remodelar y un clasificador de verificación, respectivamente. La figura 7.5 ilustra el esquema de partición de la base de datos. El clasificador usado para remodelar sirve como guía para las técnicas introducidas en el capítulo 6. El clasificador de verificación sirve como una manera de evaluar los dibujos generados. Además, se entrenó un clasificador de prueba con un conjunto de categorías similares a las usadas para entrenar los otros dos clasificadores; las categorías alternativas se presentaron anteriormente en el cuadro 5.2 del capítulo 5. El clasificador de prueba sirve como una manera alternativa de evaluar los dibujos generados usando categorías que tienen una distribución distinta a las usadas para entrenar las técnicas de remodelación. Los tres clasificadores comparten la misma arquitectura.

Para mezclar entre dos categorías, se entrenó el modelo *sketch-rnn* con una categoría para las técnicas de reconstrucción y autocompletado[2] y dos para interpolación, filtración, búsqueda simple y búsqueda ancha para un total de 30,000 y 60,000 dibujos respectivamente. Se entrenó cada clasificador con 5 categorías por un total de 100,000 dibujos cada uno. Se entrenó el modelo amortizador usando vectores latentes tomados de una distribución uniforme entre -3 y 3 y convertidos en dibujos por el modelo *sketch-rnn*.

Se ocupó la implementación de *sketch-rnn* realizada en TensorFlow y liberada por los mismos autores del artículo original[3]. Se implementaron todas las técnicas de remodelación, así como el modelo amortizador, en TensorFlow. Se configuraron todos los modelos con hiperparámetros inspirados en la configuración de *sketch-rnn*;

[2]Para estas dos técnicas, es técnicamente posible entrenar el modelo base con más categorías, aunque no hay antecedentes que lo han probado.

el apéndice B presenta los valores exactos utilizados.

7.2.3. Evaluación

En cada experimento, se generaron 10,000 dibujos por cada una de las técnicas de interpolación, reconstrucción, autocompletado, filtración, búsqueda simple y búsqueda ancha. Los dibujos de la técnica de filtración fueron filtrados de un conjunto de 100,000 dibujos, como se describió en la sección 6.1. Para establecer una línea base, se tomaron un conjunto de 10,000 dibujos de la base de datos que no fueron vistos anteriormente por el clasificador de verificación ni de prueba. Además, se tomaron 10,000 vectores latentes aleatoriamente de una distribución $\mathcal{N}(0, 1)$, los cuales se decodificaron en dibujos para evaluar el espacio latente. Para las técnicas de reconstrucción y autocompletado, las evaluaciones presentadas más adelante son de una combinación de dibujos generados. Se generaron la mitad de estos usando dibujos de una de las categorías de la mezcla y un modelo entrenado con la categoría restante, y la otra mitad de forma inversa.

Ponderación de la entropía cruzada binaria multi-etiqueta.

Se convirtieron los vectores de similitud otorgados por el clasificador en una puntuación escalar a través de un promedio ponderado por categoría:

$$\hat{\ell}_{XE} = -\frac{1}{n}\sum_{i=1}^{n} \omega_i \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)\right) \tag{7.1}$$

donde $\hat{\boldsymbol{y}}$ es el vector de similitud predicho, $\boldsymbol{y}$ el vector de similitud esperado, ω_i es un peso asignado a la categoría i, $\sum_{i=1}^{n} \omega_i = 1$ y n es el número de categorías.

[3] `https://github.com/tensorflow/magenta/tree/master/magenta/models/sketch_rnn`

Categoría	Factor Deseado	Peso	Rol	Ejemplo
1	1	.4156	Principal	Casa
2	1	.4156		Lavadora
3	0.5	.0562	Secundario	Televisión
4	0.25	.0562		Dona
5	0.25	.0562		Montaña

Cuadro 7.2: Los pesos asignados a cada categoría para el promedio ponderado.

Se definió el vector de similitud esperado $\boldsymbol{y}$, que funciona como guía para las técnicas de filtración, búsqueda simple y búsqueda ancha, de la siguiente manera. El valor para los dos factores que corresponden a las categorías principales es simplemente 1, dada la definición de una mezcla presenta en la sección 1.1. No obstante, la composición de los otros factores de similitud es difícil de definir. La televisión, por ejemplo, es visualmente parecida a una casa, pero es difícil asignar un valor de similitud específico, como 0.4 o 0.6. Por lo tanto se decidió, usando nuestra opinión subjetiva, usar el promedio entre los dos vectores deseados para las categorías principales, definidos en el cuadro 7.1.

$$\boldsymbol{y} = [1, 1, 0.5, 0.25, 0.25] \tag{7.2}$$

Para reflejar la incertidumbre inherente en esta última decisión se eligieron los pesos $\boldsymbol{\omega}$ de la entropía cruzada binaria multi-etiqueta ponderada para poner más importancia a las categorías principales; el cuadro 7.2 establece los valores elegidos. Este esquema también refleja la suposición, nuevamente basada en nuestra opinión subjetiva, que no es tan importante tener un factor exacto por una categoría secundaria como lo es tener un factor muy cercano a 1 para las categorías de la mezcla[4].

Se configuraron las técnicas de filtración, búsqueda simple y búsqueda ancha para trabajar sobre esta versión de la entropía cruzada binaria multi-etiqueta. El vector de similitud esperado fue el mismo para todos los experimentos.

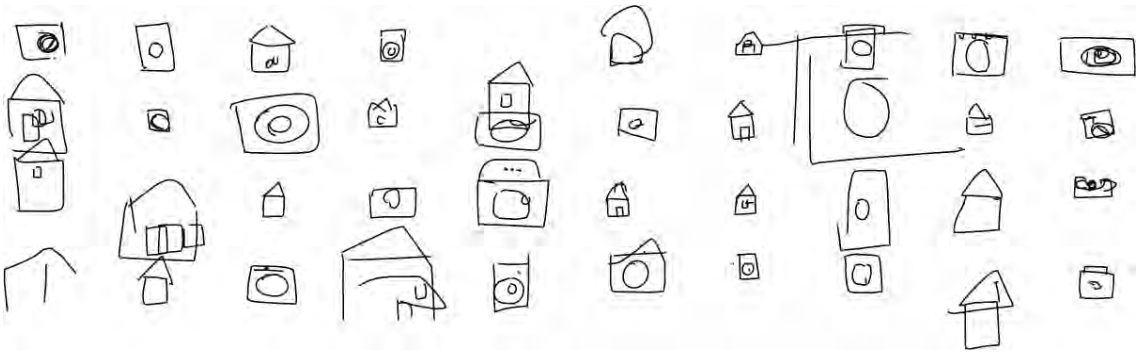
7.3. Resultados

A continuación se presentan los resultados de los tres experimentos realizados. Esta sección comienza con algunas muestras de dibujos generados por interpolación, reconstrucción, autocompletado, filtración, búsqueda simple y búsqueda ancha, acompañadas de descripciones intuitivas. Enseguida se presenta un análisis cuantitativo de los vectores de similitud asignados a los dibujos generados por el clasificador.

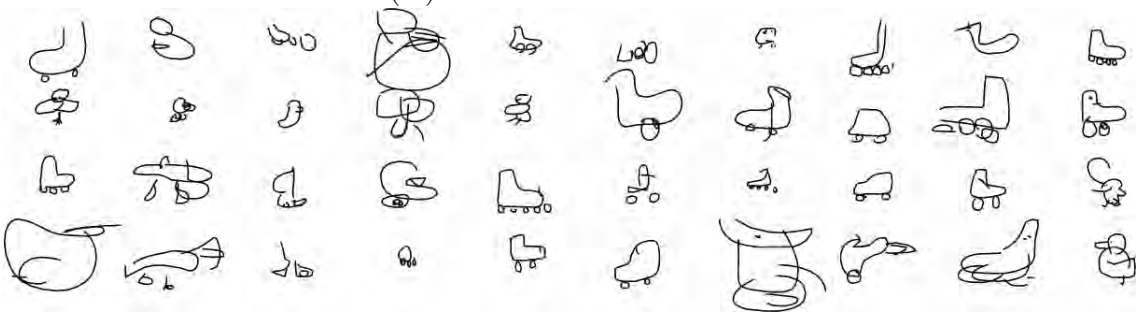
7.3.1. Análisis visual

Interpolación. La figura 7.6 muestra algunos ejemplos de dibujos generados mediante la técnica de interpolación. La mayoría se parecen más a una categoría principal de la mezcla que a la otra. Por ejemplo, consideramos que el dibujo del conjunto (a) en primer renglón y la segunda columna (1,2) y su vecino a la derecha (1,3) se parecen a las categorías LAVADORA y CASA, respectivamente. Sin embargo, algunas muestras del mismo conjunto destacan por lo que parece una forma exterior de una casa y una forma interior de una lavadora; consideramos que, por ejemplo, el dibujo en el último renglón y sexta columna (4,6) parece tener paredes cuadradas con un techo triangular y a la vez una puerta circular adentro. Lo mismo ocurre

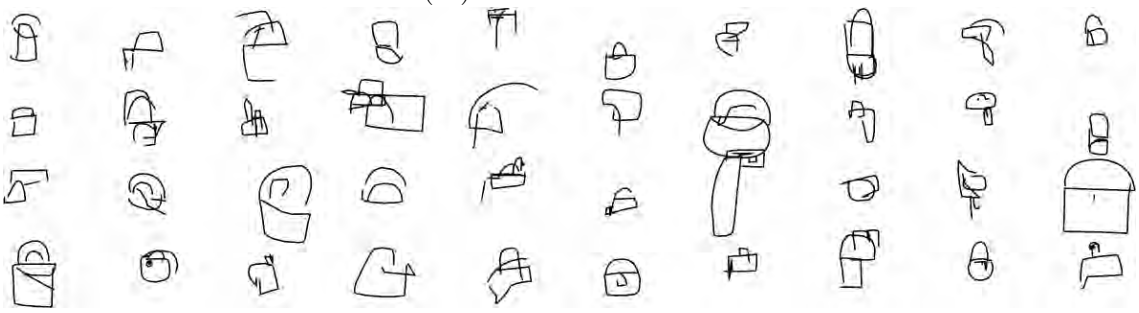
[4]Sin estos pesos, las técnicas de remodelación parecieron poner más importancia en hacer que el dibujo generado no fuera similar a las tres categorías secundarias que similar a las dos categorías principales.



(a.) Casa con lavadora.



(b.) Pato con patín.



(c.) Buzón con bolsa.

Figura 7.6: **Interpolación** Dibujos generados por la técnica de interpolación, seleccionados al azar.

con los dibujos del conjunto (b), que corresponde a la mezcla entre un pato y un patín, pero al parecer no muestran características de ambas categorías. Sin embargo, existen algunos dibujos como el del último renglón y la columna ocho (4,8), que parece tener el cuerpo y pico de un pato y ruedas de un patín. Entre los buzones y las bolsas del conjunto (c), dibujos como el de la posición (2,5) parecen agregar las

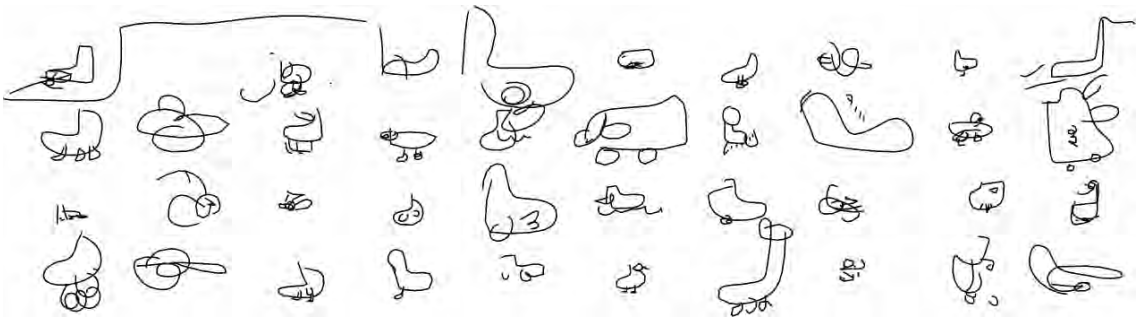


Figura 7.7: **Reconstrucción**. Dibujos de mezclas entre PATO y PATÍN generados por la técnica de reconstrucción, seleccionados al azar.

correas de una bolsa a un buzón con su poste. No consideramos que la mayoría de los otros dibujos tengan características de mezcla entre dos categorías.

Reconstrucción. La figura 7.7 presenta dibujos generados por la reconstrucción. La mayoría de ellos tampoco parecen mostrar características de mezcla visual. El dibujo en el tercer renglón y quinta columna y (3,5) puede ser interpretado como un patín con alas de la categoría PATO, pero consideramos que los otros dibujos son generalmente difíciles de interpretar como mezclas. En general, la técnica tampoco parece una manera eficiente para generar dibujos que parecen mezclas. Los resultados visuales de las otras dos mezclas experimentales fueron similares y son incluidos en el apéndice C.

Autocompletado. La técnica de autocompletado parece generar dibujos que pueden ser considerados mezclas más frecuentemente que las técnicas anteriores. En la figura 7.8, el dibujo en la primera renglón y última columna y (1,10) parece agregar una puerta circular de la categoría LAVADORA al techo de la categoría CASA. Sin embargo, aún existen varios dibujos que nosotros consideramos que no muestran características de una mezcla, como el dibujo en el tercer renglón y primera columna

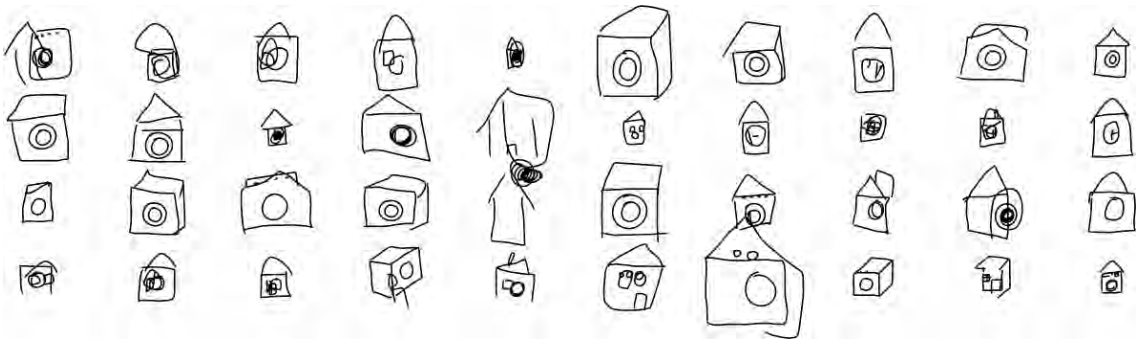


Figura 7.8: **Autocompletado**. Dibujos de mezclas entre CASA y LAVADORA generados por la técnica de autocompletado, seleccionados al azar.

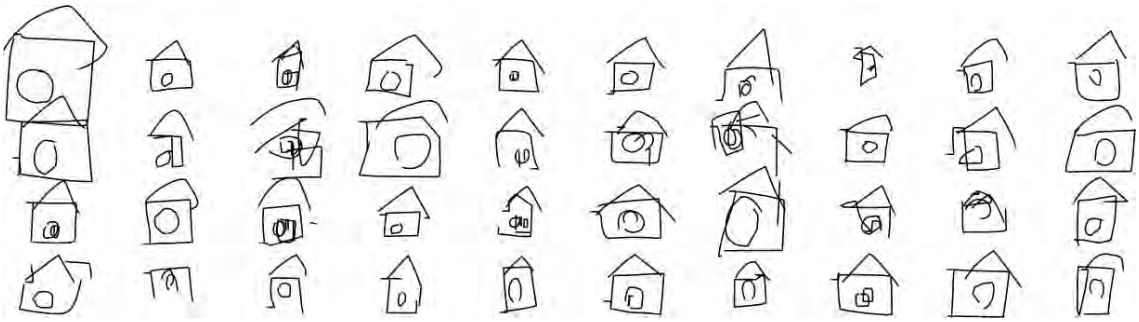


Figura 7.9: **Filtración.** Dibujos de mezclas entre CASA y LAVADORA ordenados por entropía cruzada binaria multi-etiqueta ponderada y presentados en orden de izquierda a derecha y de arriba a abajo.

(3,1). Un error común con esta técnica es ejemplificado por el techo extendido del dibujo en el último renglón y séptima columna y (4,7), en el cual el decodificador sigue dibujando sin terminar en un punto más común. Los resultados visuales de las otras dos mezclas experimentales fueron similares y son incluidos en el apéndice C.

Filtración. La filtración produjo varias muestras que pueden ser consideradas más como una mezcla visual, de acuerdo con las definiciones presentadas previamente. Por ejemplo, varios de los dibujos en el primer renglón parecen tener la puerta de la categoría LAVADORA y el techo de la categoría CASA. No obstante, la técnica no

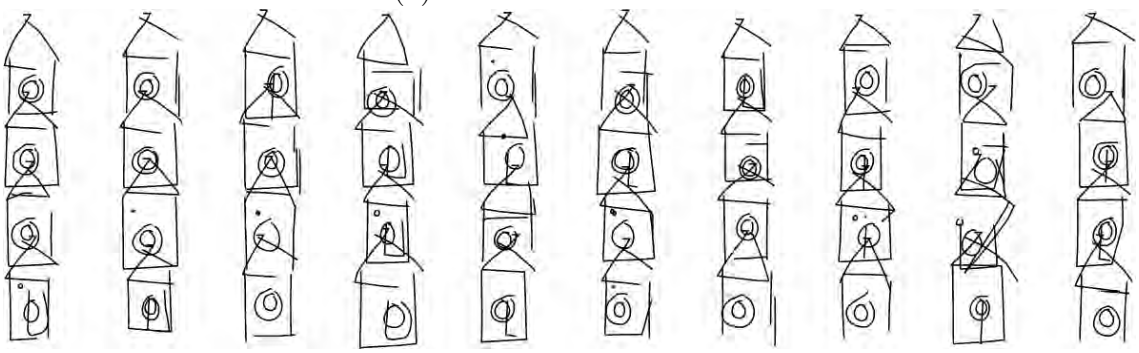
funcionó tan bien como se esperaba. Aunque la figura 7.9 viene ordenada, hay dibujos como el del segundo renglón y tercera columna y (2,3) que a nuestra consideración no parece más a una mezcla que otros dibujos ordenados más abajo[5], como el dibujo en el último renglón y penúltima columna (4,9). Los resultados visuales de las otras dos mezclas experimentales fueron similares y son incluidos en el apéndice C.

Estos experimentos permiten comprobar que un modelo *sketch-rnn* entrenado con solamente dos categorías de dibujos tiene la capacidad de generar dibujos que pueden ser considerados mezclas sin necesidad de entrenarlo con ejemplos de mezclas.

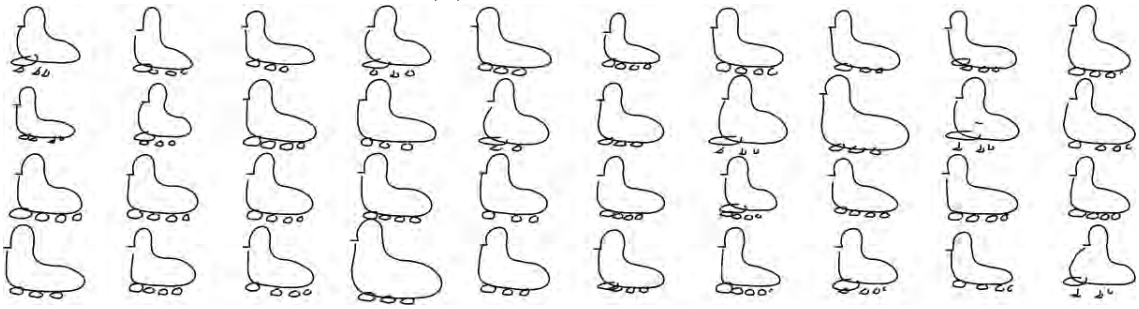
Búsqueda simple con SGD. La técnica de búsqueda simple con SGD logró localizar una zona dentro del espacio latente que contuviera una colección densa de dibujos que nosotros consideramos mezclas. Casi todos los dibujos en el primer conjunto de la figura 7.10 parecen exhibir el techo y paredes exteriores de la categoría CASA con la puerta circular de la categoría LAVADORA, y algunos también agregan puntos que se parecen a los botones de esta última. De forma similar, consideramos que los dibujos en el conjunto (b) tienen características tanto de la categoría PATO como de la categoría PATÍN. Un pico muy pequeño a la izquierda de cada dibujo parece ser de la categoría PATO mientras que las ruedas parecen ser de la categoría PATÍN. El cuerpo de cada dibujo puede verse como una forma ambigua, generalmente similar a la bota de un patín pero a la vez redondeada, como el cuerpo de un pato. Lo mismo ocurre con los dibujos del conjunto (c), en el cual todos los dibujos parecen tener una forma redondeada con correas que está colocada sobre un poste,

[5]Una posibilidad es que el clasificador tiene hoyos en los cuales puede ser fácilmente engañado; tal propiedad ha sido mostrada por clasificadores en el dominio de imágenes (Nguyen et al., 2015).

(a) Casa con lavadora.



(b) Pato con patín.



(c) Buzón con bolsa.

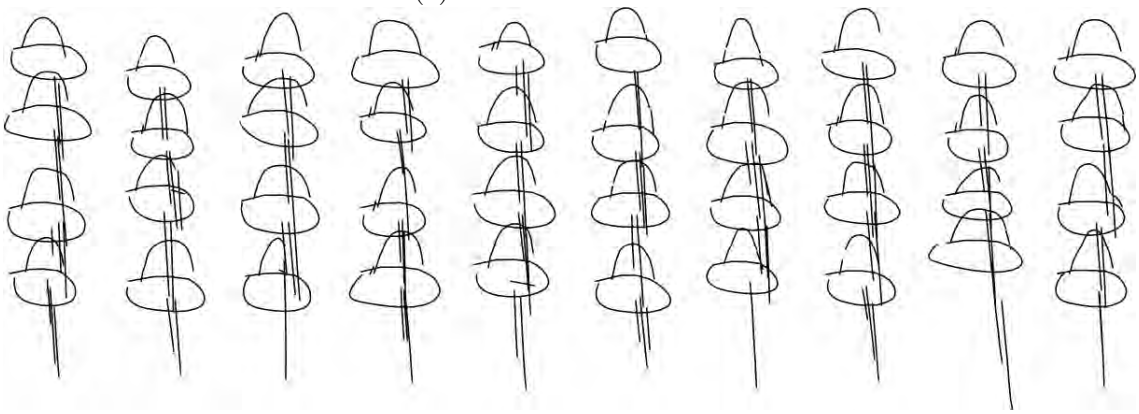
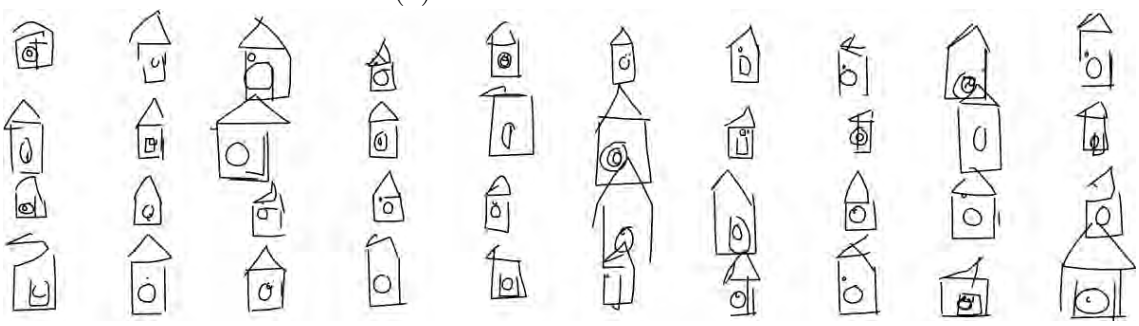


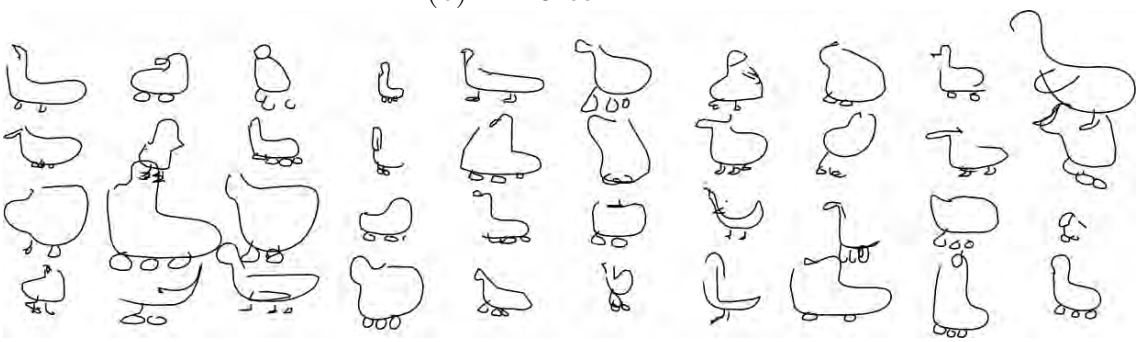
Figura 7.10: **Búsqueda simple**. Dibujos generados por la técnica de búsqueda simple, seleccionados al azar.

características que consideramos representativas de bolsas y buzones respectivamente. Por lo general, consideramos que hay pocos dibujos que no contienen ninguna característica de una mezcla. Sin embargo, parece que la diversidad visual entre los

(a) Casa con lavadora.



(b) Pato con patín.



(c) Buzón con bolsa.

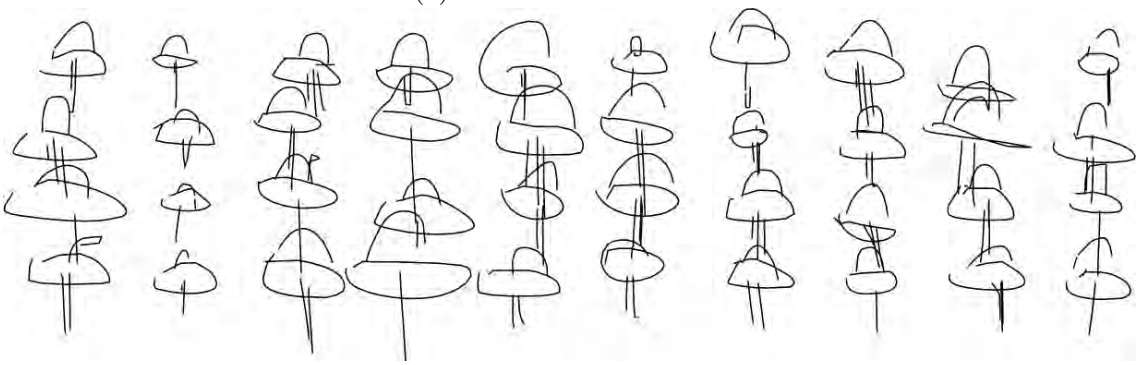


Figura 7.11: **Búsqueda ancha**. Dibujos generados por la técnica de búsqueda ancha, seleccionados al azar.

dibujos es más baja en comparación con los conjuntos de dibujos generados por las otras técnicas.

Búsqueda ancha con SGD. Parece que una alta proporción de los dibujos generados por la técnica de búsqueda ancha también presentan características de mezclas.

Algunos de los dibujos en la figura 7.11 parecen tener formas muy distintas a los dibujos encontrados por la búsqueda simple, lo que sugiere que la búsqueda simple pierde varios dibujos que pueden ser considerados mezclas. Por ejemplo, consideramos que varias casas del conjunto (a) tienen una puerta circular con solamente un anillo mientras que todas las casas en la figura 7.10 tienen puertas con doble anillo. En general, parece que la diversidad visual entre los dibujos en esta figura es mayor de la búsqueda simple; esto es especialmente evidente comparando las mezclas entre PATO y PATÍN de los conjuntos (b) de las figuras 7.10 y 7.11. Sin embargo, consideramos que algunos dibujos generados por la búsqueda ancha no muestran características tan obvias de mezcla visual.

Comparación cualitativa con otros pintores artificiales existentes

Al igual que *sketch-rnn*, Shizuka (Mukaiyama, 2013), *blender* (Cunha et al., 2017) y *boat-house* (Pereira y Cardoso, 2002) se basan en dibujos hechos por humanos. En contraste, estos sistemas generalmente se limitan a operaciones de reorganización, en las cuales cada dibujo es descompuesto por un humano en partes inmutables que son desplazadas en el espacio bidimensional. Mientras estas partes no pueden ser modificadas en estos sistemas, en *sketch-rnn* cualquier parte del dibujo es mutable. Un triángulo, por ejemplo, no se compone simplemente por tres líneas, sino por una curva suavizada definida por varios puntos intermedios; esto le permite al modelo generar una variedad de formas que parecen triángulos pero que son visualmente distintas unas de otras. Asimismo, algunas mezclas visuales entre las categorías PATO y PATÍN parecen contener una forma que empieza como la bota de un patín pero termina en una cabeza redondeada o pico puntiagudo de un pato, al parecer sin

un punto obvio de separación. Algunos dibujos generados por *sketch-rnn* también difieren por la repetición de algún rasgo visual. Por ejemplo, las mezclas visuales entre PATO y PATÍN pueden tener cero o hasta cuatro ruedas o pies. De manera similar, algunas mezclas visuales entre CASA y LAVADORA cuentan con botones de lavadora mientras que otras no. La capacidad de cambiar cualquier parte del dibujo también puede ser una desventaja, ya que algunos dibujos generados por *sketch-rnn* no tienen una forma reconocible. Este problema no debería surgir en sistemas que trabajan con partes predeterminadas que son inmutables.

Otra desventaja visual de los dibujos generados por *sketch-rnn* es que la mayoría parecen ser variaciones del mismo tema visual. Por ejemplo, parece que las mezclas visuales entre CASA y LAVADORA siempre se construyen colocando un círculo dentro de un cuadrado con un triángulo arriba. Aunque la forma del círculo, cuadrado y triángulo puede variar, la mayoría de los dibujos parecen seguir este mismo patrón. En contraste, los otros sistemas generan múltiples permutaciones posicionales de un conjunto limitado de partes inmutables.

7.3.2. Análisis cuantitativo

Partiendo de la definición de mezcla visual presentada en la sección 1.1, proponemos seis métricas para evaluar las diferencias entre los vectores de similitud otorgados por el clasificador.

Promedio de la entropía cruzada ponderada

La definición de la entropía cruzada binaria multi-etiqueta ponderada[6] presentada en la sección 7.2.3 parte de la entropía cruzada binaria multi-etiqueta simple

	Clasificador	
Técnica	Validación	Prueba
Base de datos	0.50±0.13	0.50±0.13
Espacio latente	0.6±0.3	0.6±0.4
Interpolación	0.5±0.2	0.6±0.3
Reconstrucción	0.5±0.3	0.6±0.3
Autocompletado	0.5±0.4	0.7±0.6
Filtración	**0.37±0.10**	**0.5±0.2**
Búsqueda simple	**0.36±0.04**	**0.41±0.1**
Búsqueda ancha	**0.37±0.08**	**0.42±0.11**

Cuadro 7.3: Promedio de las entropías ponderadas medias para cada mezcla y su desviación estándar.

introducida en la sección 3.3.1, la cual es una métrica bien estudiada para cuantificar la diferencia entre dos vectores. Por brevedad, en esta sección se ocupa el término «entropía ponderada» para referirse a esta métrica. En este caso, la entropía cruzada se calcula entre el vector de similitud predicho y el vector de similitud deseado. El promedio de la entropía ponderada representa el nivel con que los dibujos generados minimizan la diferencia entre estos dos vectores.

El cuadro 7.3 reporta el promedio de las entropías ponderadas para cada mezcla junto con su desviación estándar[7]. El cuadro muestra que las técnicas de filtración, búsqueda simple y búsqueda ancha obtuvieron valores de entropía ponderada más bajos que las técnicas de interpolación, reconstrucción y autocompletado. Por lo tanto, se puede decir que estas técnicas generaron dibujos que obtuvieron vectores más similares al vector deseado. Lo mismo pasó con el clasificador de prueba, aunque

[6]Recordemos que la ponderada representa incertidumbre sobre la composición exacta deseada de las categorías secundarias.

[7]En este capítulo, los valores fueron reportados siguiendo las reglas de las cifras significativas: todas las desviaciones fueron redondeadas a una cifra significativa, con excepción del caso en donde esta cifra es igual a uno. La media fue entonces reportada hasta la misma exactitud que tiene esta desviación redondeada.

	Clasificador	
Técnica	Validación	Prueba
Base de datos	0.136±0.019	0.13±0.02
Espacio latente	0.29±0.06	0.36±0.09
Interpolación	0.24±0.03	0.3±0.10
Reconstrucción	0.289±0.014	0.29±0.07
Autocompletado	0.236±0.012	0.36±0.17
Filtración	**0.096±0.004**	**0.22±0.07**
Búsqueda simple	**0.04±0.008**	**0.1±0.06**
Búsqueda ancha	**0.077±0.008**	**0.113±0.017**

Cuadro 7.4: La desviación estándar media de la entropía ponderada

con éste todas las técnicas producen muestras más lejanas al vector deseado.

Desviación estándar de la entropía cruzada ponderada

La desviación estándar de la misma entropía ponderada es principalmente una medida de la frecuencia con que una técnica genera dibujos que obtengan una entropía ponderada cercana a la media. Como muestra el cuadro 7.4, la desviación estándar de la entropía ponderada fue alta en las técnicas de interpolación, reconstrucción y autocompletado. En contraste, esta desviación fue un orden de magnitud menor con las técnicas que incorporan retroalimentación del clasificador. Con el clasificador de prueba, las desviaciones estándares de las nuevas técnicas propuestas subieron, pero quedaron más pequeñas que las de las otras técnicas.

Mapas de calor y frente de Pareto.

Una manera común en la estadística de analizar con más detalle la distribución de un conjunto de datos es a través de un mapa de calor. El mapa organiza los vectores de similitud por ocurrencia: si hay muchos dibujos que obtienen el mismo vector[8], se

muestra una marca brillante en el mapa en el lugar que corresponde a los valores del vector. La intensidad del brillo se escala logarítmicamente con el número de dibujos. En los mapas que se muestran a continuación, el eje horizontal representa el factor de similitud de la primera de las dos categorías principales de la mezcla, mientras que el eje vertical representa el factor de similitud para la segunda[9]. Dado que la esquina superior derecha contiene los compartimentos que representan factores altos para ambas categorías simultáneamente, es deseable que la distribución de los dibujos esté centrada en esa región. Mientras tanto, la apariencia de dibujos en las otras regiones significa que el proceso no pudo evitar la producción de dibujos que obtuvieron factores bajos.

El frente de Pareto es una métrica frecuentemente usada en optimización multi-objetivo (Van Veldhuizen y Lamont, 1998; Kolar et al., 2009; Everingham et al., 2002). El frente aplicado a dos factores de similitud indica el límite superior al que puede llegar uno de los factores sin que el otro factor disminuya. En los mapas de calor a continuación, el frente de Pareto está indicado por una línea blanca. El área debajo de la línea contiene aquellos vectores de similitud para los cuales el proceso generativo ha logrado generar un dibujo que es estrictamente igual o mejor en todos los factores de similitud. Si la línea se encuentra muy pegada al borde superior y al borde derecho entonces el proceso ha demostrado que es capaz de generar dibujos que obtuvieron dos factores de similitud que se acercan simultáneamente al valor superior.

[8]Se agruparon factores de similitud que se difieren por menos de 0.04 en el mismo punto.
[9]Los factores para las otras categorías no se muestran en estos mapas.

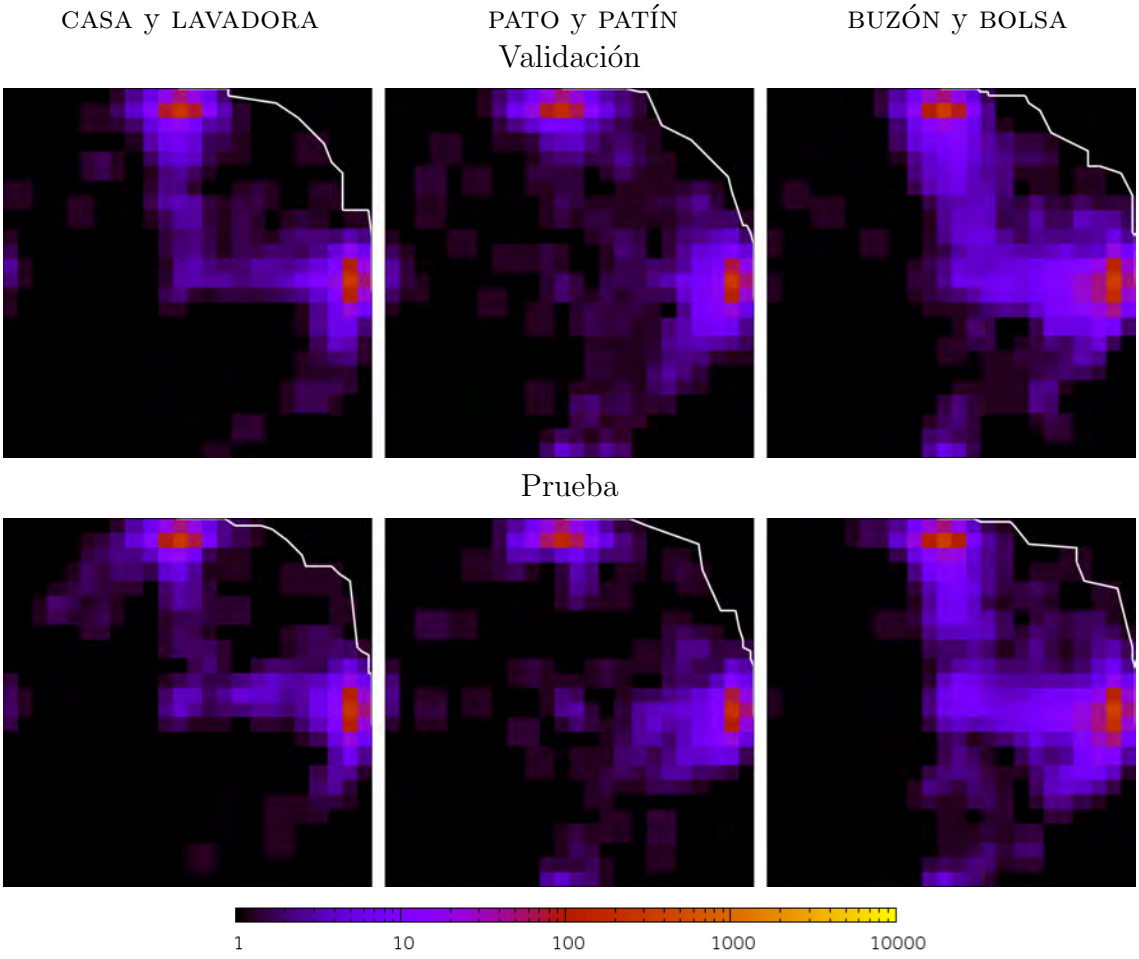


Figura 7.12: Mapas de calor para la base de datos. Cada mapa representa 10,000 dibujos.

Base de datos. En la figura 7.12, el mapa de la izquierda corresponde al experimento sobre la mezcla entre CASA y LAVADORA, el central entre PATO y PATÍN y el de la derecha entre BUZÓN y BOLSA. El eje horizontal representa el factor de similitud para la primera categoría de la mezcla y el eje vertical el factor para la segunda. Los dos ejes tienen una escala de 0 a 1 de izquierda a derecha y de abajo hacia arriba. En los mapas de calor, las zonas con concentración de dibujos más alta están ubicadas en las regiones en las cuales uno de los dos factores de similitud es 1

y el otro es 0.5. Hay una correspondencia directa entre este resultado y los vectores de similitud asignados a estas dos categorías. Una peculiaridad es que los frentes de Pareto en estos mapas señala la existencia de dibujos en la base de datos que obtuvieron dos factores de similitud altos simultáneamente. Este hecho puede ser explicado en parte por ruido en la base de datos, en la cual existe una cantidad nada insignificante de dibujos que parecen mal categorizados[10].

Espacio latente. La figura 7.13 muestra mapas de calor para dibujos tomados aleatoriamente del espacio latente, que también se concentran en las mismas dos zonas que los dibujos de la base de datos. Este efecto es esperado, dado que se entrenó el modelo para generar dibujos que son similares a los de la base de datos. Una diferencia importante es que los mapas muestra una distribución de dibujos más diversa que la de la base de datos. Esto incluye dibujos que no son similares a ninguna de las dos categorías, y también dibujos que son muy similares a ambas.

Interpolación. La figura 7.14 muestra los mapas de calor para la técnica de interpolación. A pesar del hecho de que los vectores latentes interpolados están situados en medio de las dos categorías, los mapas no difieren mucho de los del espacio latente. La diferencia más evidente es una reducción en la cantidad de dibujos que no se parecen a ninguna categoría. El frente de Pareto se encuentra más cerca de las orillas de los mapas y especialmente de la esquina superior derecha. Sin embargo, las zonas de alta concentración no están cerca de esta línea, lo que significa que estos dibujos no se generan frecuentemente. En general, los mapas no demuestran evidencia para decir que la interpolación frecuentemente genera dibujos con dos vectores

[10]Morris (2017) explora la confiabilidad de la base de datos en más detalle en una publicación de *blog*.

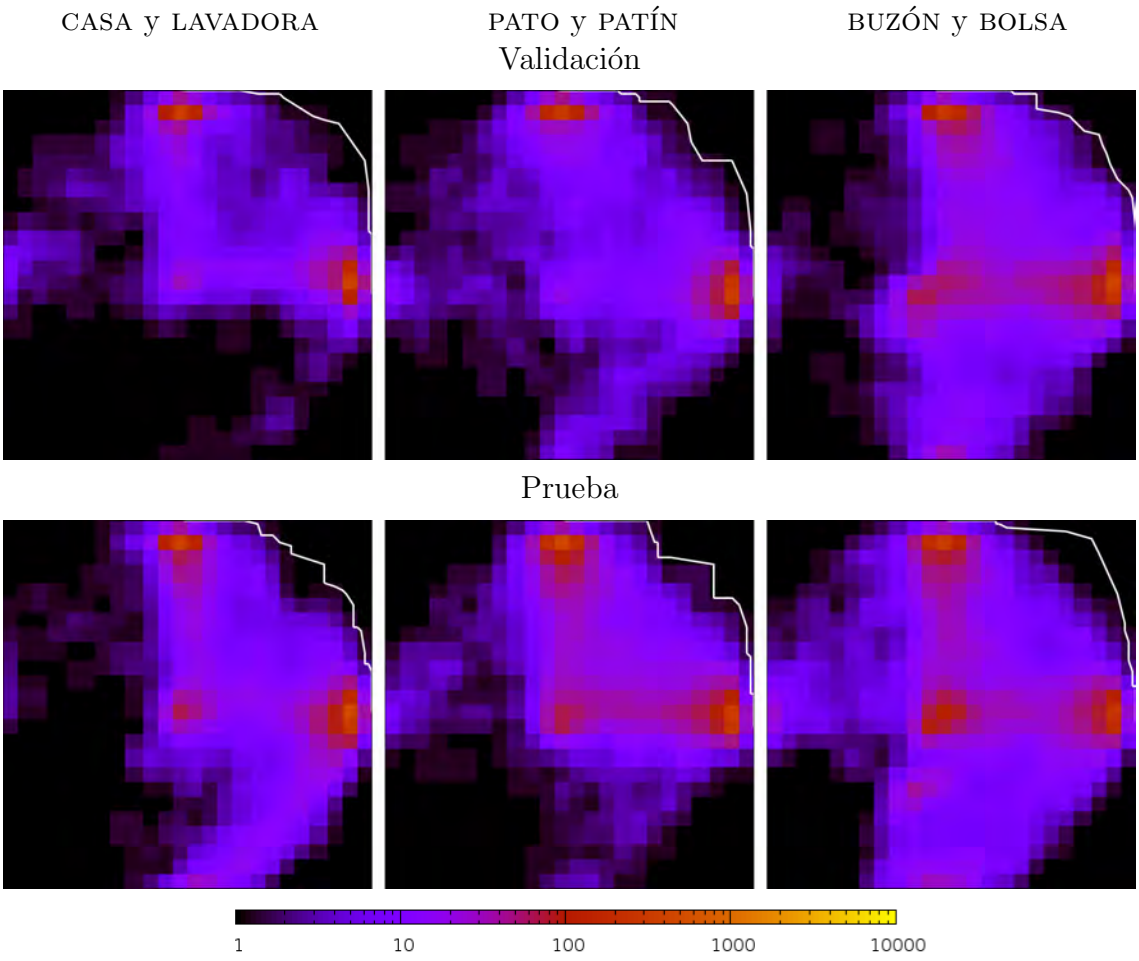


Figura 7.13: Mapas de calor para el espacio latente. Cada mapa representa 10,000 dibujos.

de similitud altos simultáneamente.

Reconstrucción. Los mapas de calor para la técnica de reconstrucción son parecidos a los de la interpolación. Las zonas rojas muestran que los dibujos están algo más agrupados en el cuadrante superior derecho del mapa, lo cual es más deseable. Sin embargo, la presencia de partes azules en los otros cuadrantes del mapa significa que los dibujos con vectores de similitud no deseables siguen siendo comunes.

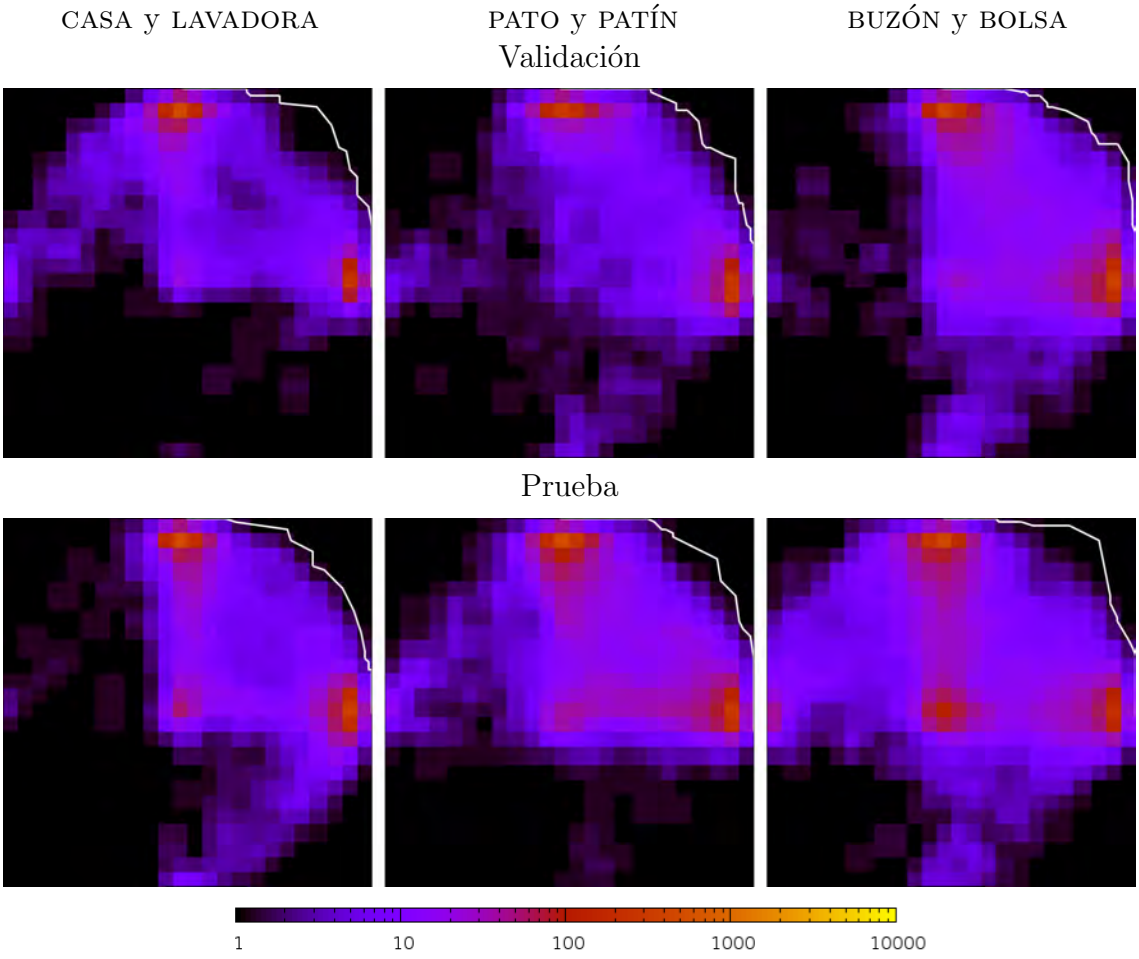


Figura 7.14: Mapas de calor para la técnica de interpolación. Cada mapa representa 10,000 dibujos.

En general, los mapas tampoco demuestran evidencia que la técnica frecuentemente genera dibujos que son similares a las dos categorías de la mezcla simultáneamente.

Autocompletado. Los mapas de calor para la técnica de autocompletado en la figura 7.16 muestran una mejora más evidente. Las zonas de concentración rojas y naranjas forman una línea a lo largo de la orilla del mapa siguiendo parte del frente

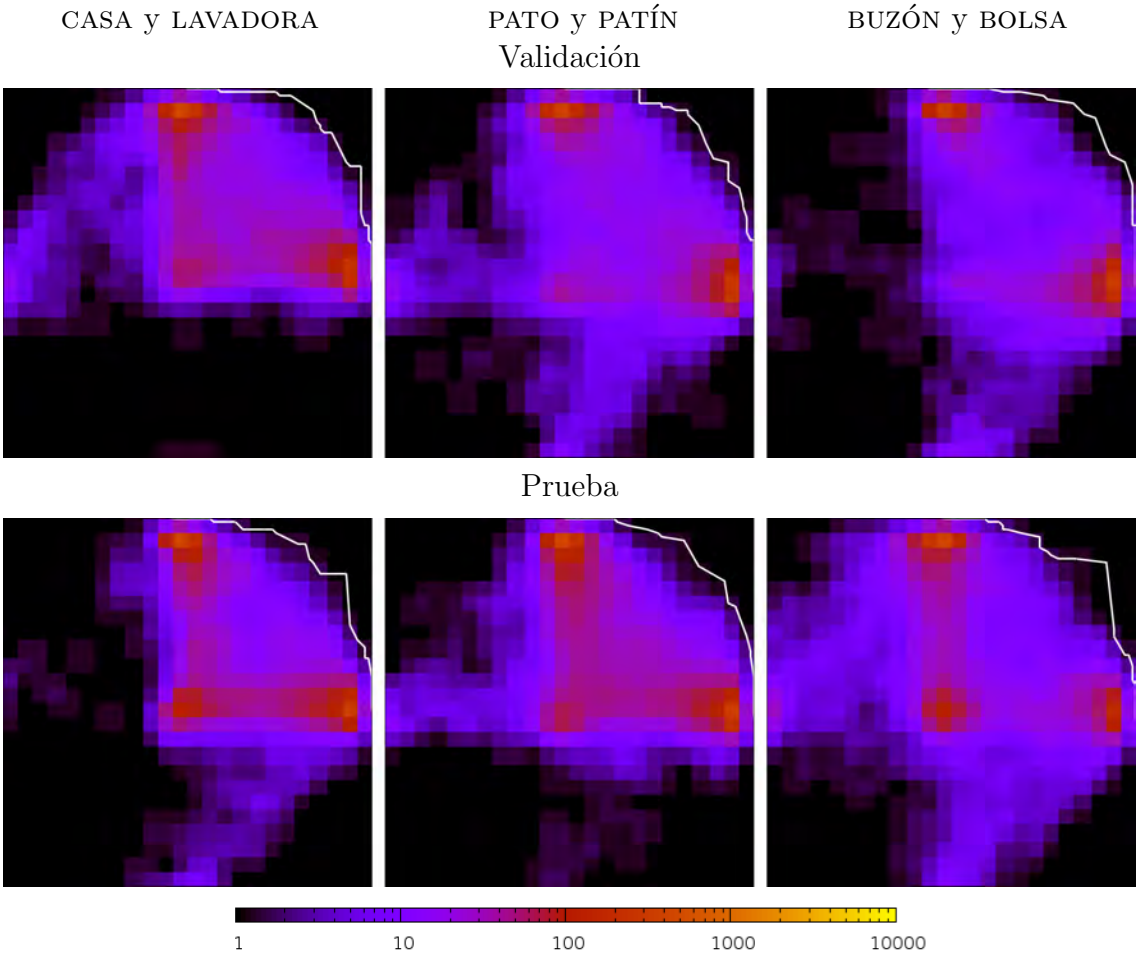


Figura 7.15: Mapas de calor para la técnica de reconstrucción. Cada mapa representa 10,000 dibujos.

de Pareto; este efecto es más visible en el mapa de la derecha que corresponde a la mezcla entre BUZÓN y BOLSA. Sin embargo, todavía hay muchas zonas de alta concentración que se encuentran muy lejos de la esquina superior derecha.

Filtración. Los mapas de calor de la filtración en la figura 7.17 muestran una distribución distinta. La concentración de color en el cuadrante superior derecho es claramente visible en estos mapas. Los otros cuadrantes del mapa se encuentran

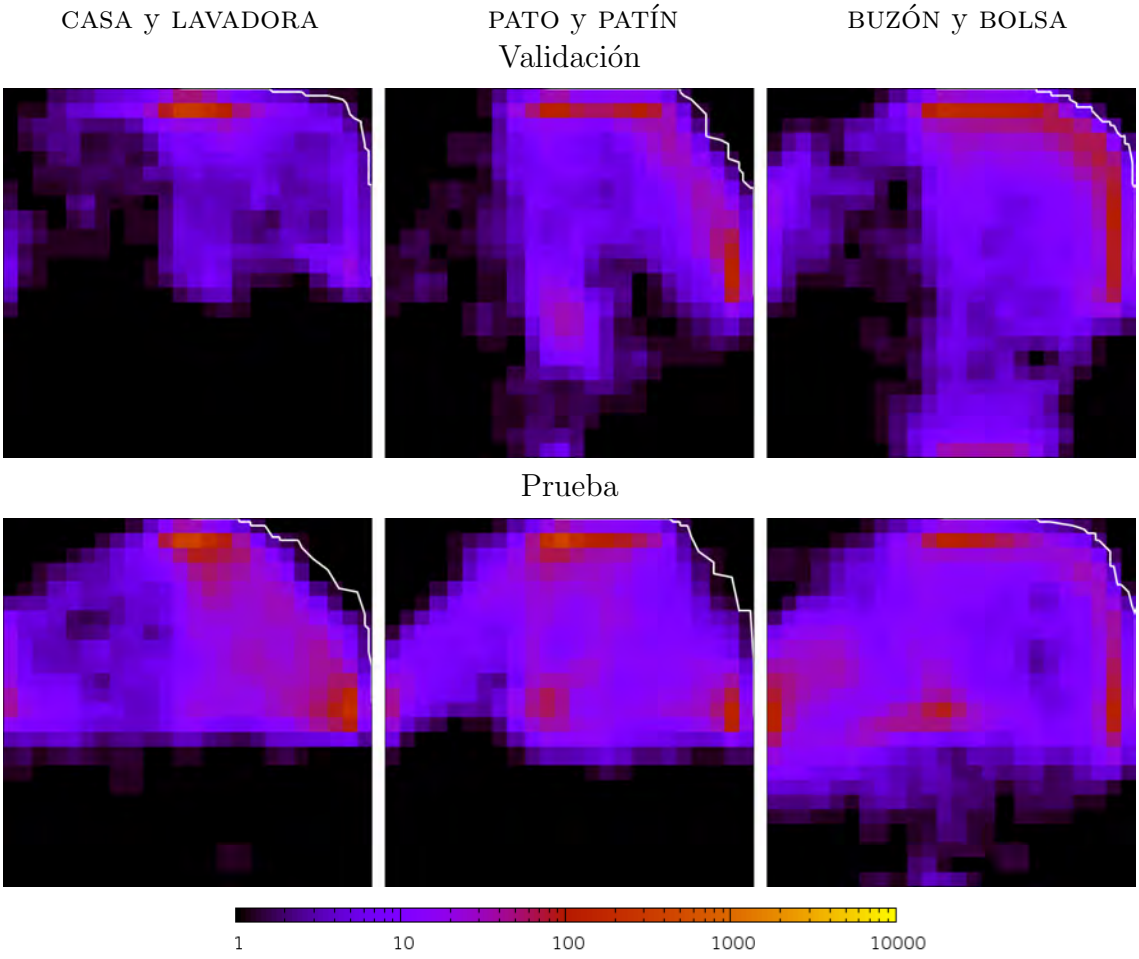


Figura 7.16: Mapas de calor para la técnica de autocompletado. Cada mapa representa 10,000 dibujos.

mayormente oscuros, a diferencia de los otros mapas presentados anteriormente El frente de Pareto se encuentra cerca de la esquina superior derecha; en el clasificador de verificación la banda roja se encuentra pegada al frente aunque este efecto no es tan fuerte con el clasificador de prueba.

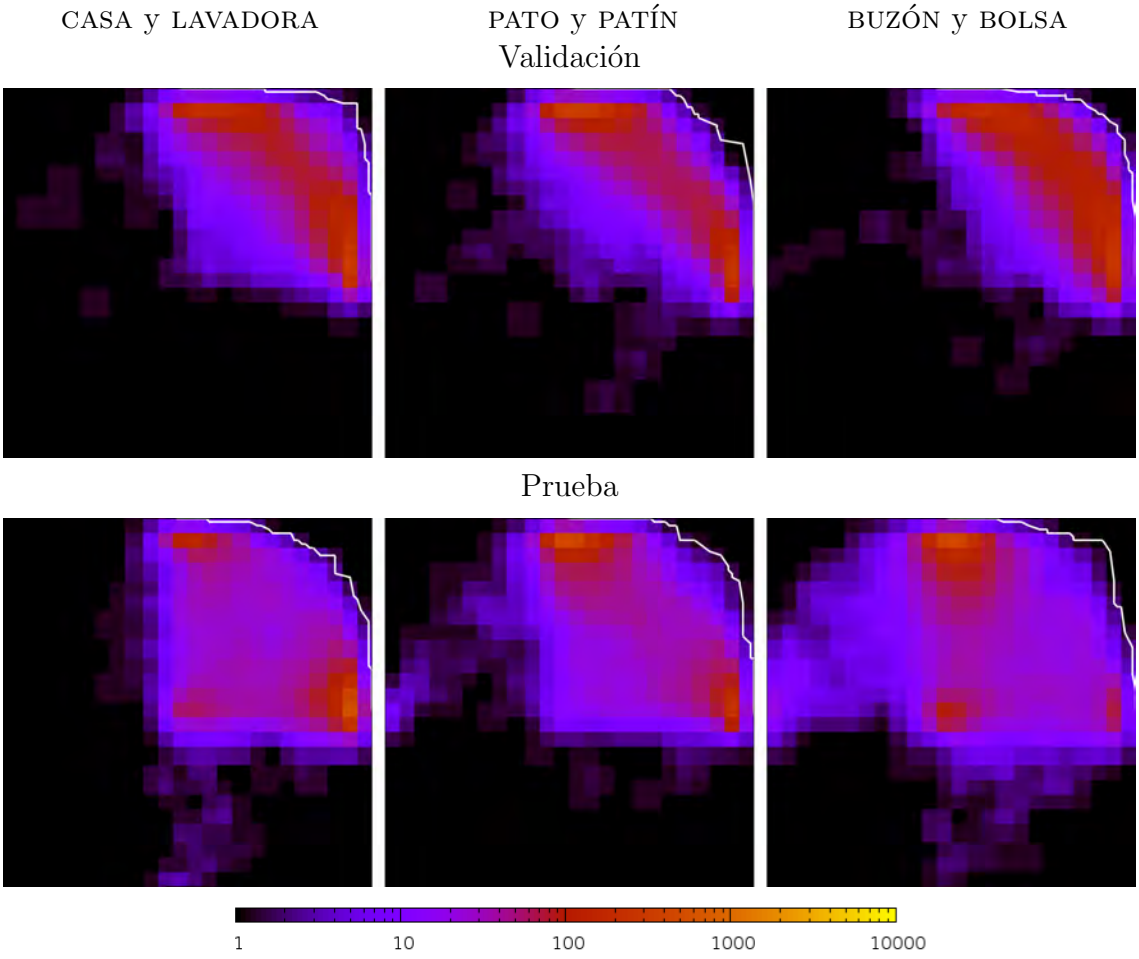


Figura 7.17: Mapas de calor para la técnica de filtración. Cada mapa representa 10,000 dibujos.

Búsqueda simple. Los mapas de calor de la figura 7.17 muestran distribuciones distintas a las de los mapas presentados anteriormente. Las zonas con mayor concentración de dibujos forman una banda roja y naranja que sigue de cerca la curva del frente de Pareto. El color oscuro de los otros cuadrantes en el mapa significa que el proceso no genera dibujos en estas regiones con frecuencia. El frente de Pareto se encuentra más alejado de las orillas y de la esquina superior derecha del mapa que el de la técnica de filtración, lo cual puede indicar que la técnica personaliza sus

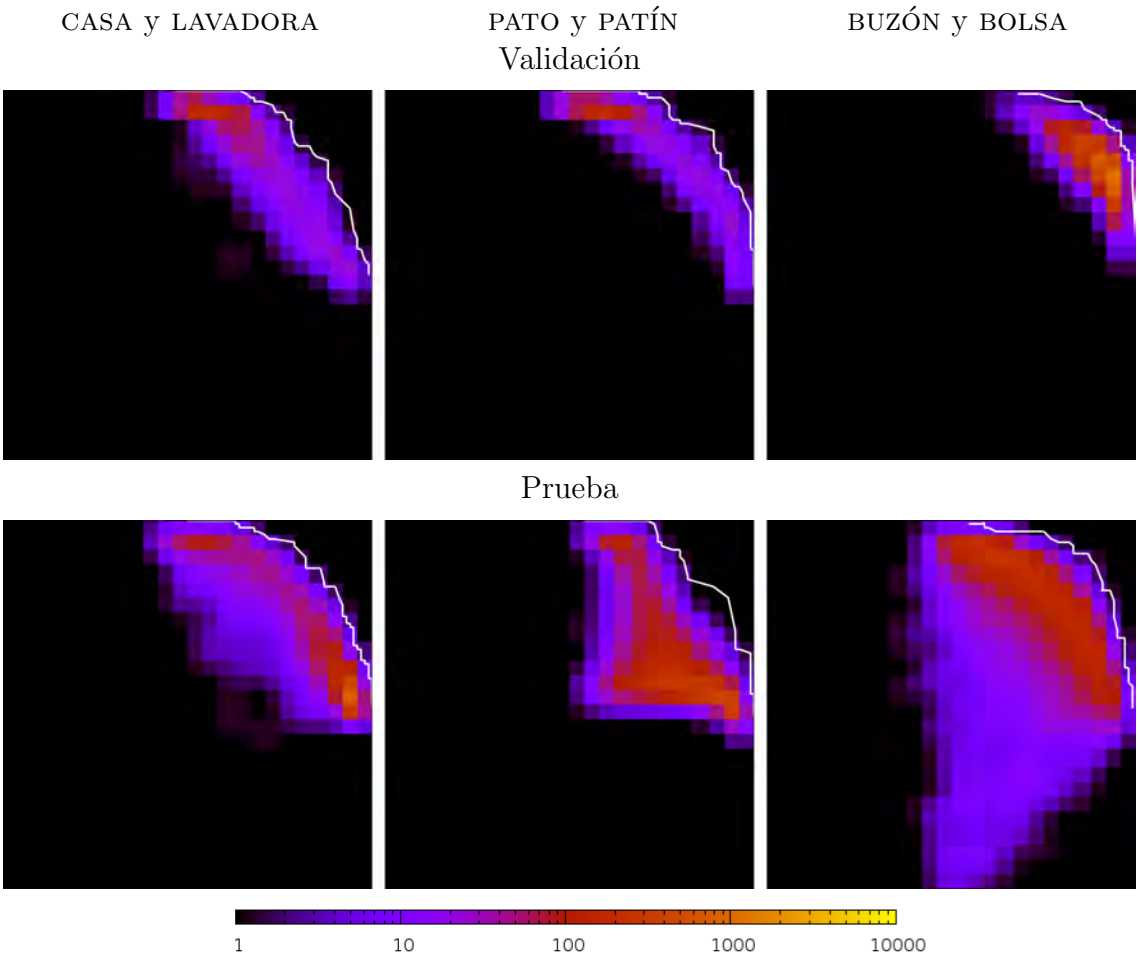


Figura 7.18: Mapas de calor para la técnica de búsqueda simple. Cada mapa representa 10,000 dibujos.

dibujos al clasificador usado durante el proceso generativo, y que los clasificadores de verificación y prueba no consideran que estos dibujos son tan similares a las dos categorías a mezclar.

Búsqueda ancha. Finalmente, los mapas de color de la búsqueda ancha en la figura 7.19 también muestran la formación de una banda roja bien definida que se acerque a las orillas del mapa, lo que significa una concentración alta de dibujos

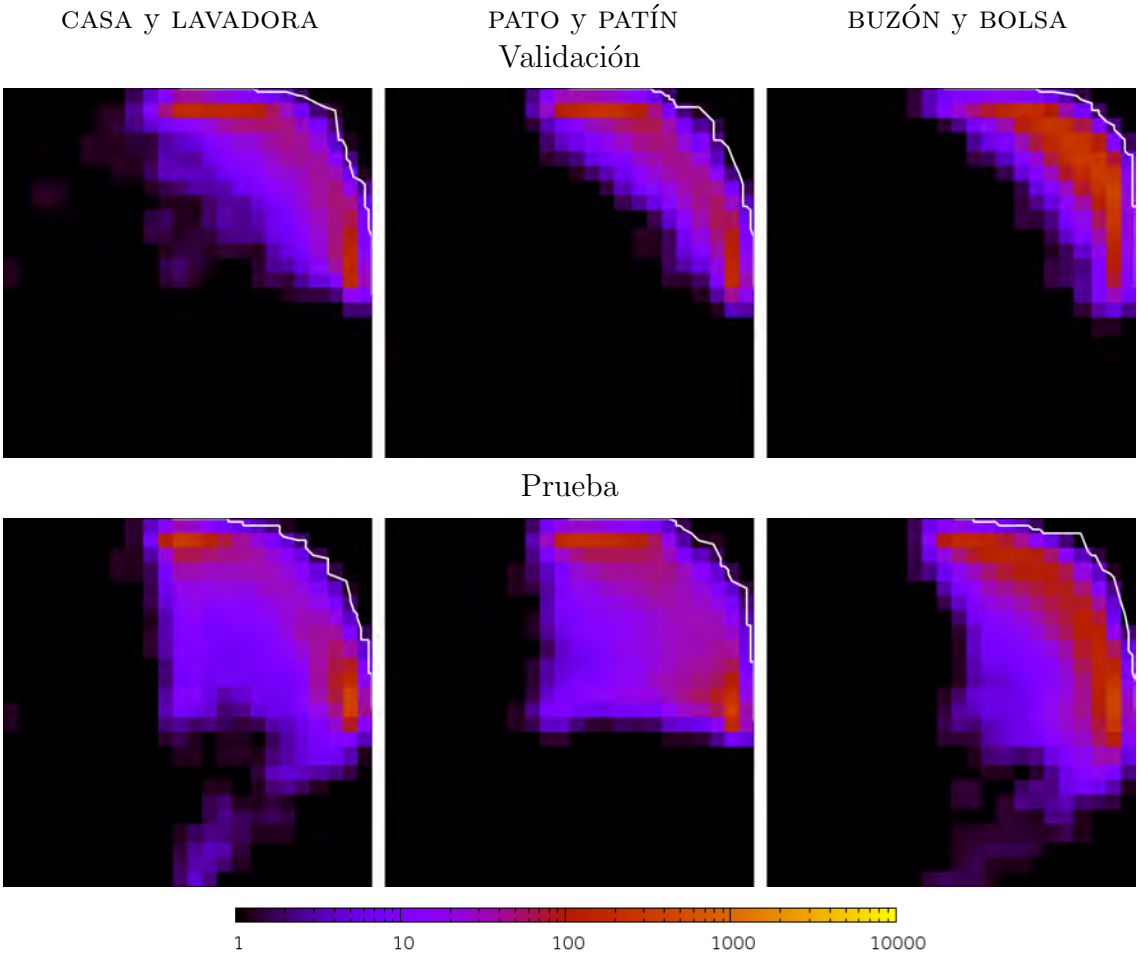


Figura 7.19: Mapas de calor para la técnica de búsqueda ancha. Cada mapa representa 10,000 dibujos.

que son similares, según el clasificador, a las dos categorías simultáneamente. En contraste con la búsqueda simple, las orillas azules de la banda se extienden más hacia el centro de cada mapa, lo que significa una proporción mayor de dibujos que no obtuvieron factores de similitud cercanos a 1 para ninguna de las dos categorías.

En general, los mapas de calor ayudan a diseccionar la entropía cruzada binaria ponderada así como explicar su desviación estándar alta. Estos mapas sugieren que

las técnicas de filtración, búsqueda simple y búsqueda ancha generan más frecuentemente dibujos que son similares, según los clasificadores de verificación y prueba, a las dos categorías principales, esto en comparación con las técnicas de interpolación, reconstrucción y autocompletado.

2Hit@k.

Una métrica que se usa frecuentemente en la tarea de clasificación multi-etiqueta es la llamada *Hit@k* (Karpathy et al., 2014). Esta métrica cuantifica la proporción de dibujos por los cuales una dada categoría, p.ej. CASA, está dentro de los k categorías a que el dibujo es más similar. La métrica *2Hit@k* simplemente extiende esta definición a dos factores: los que corresponden a las categorías a mezclar. Para calcular esta métrica, el vector de similitud es ordenado con respecto a sus factores de similitud del más alto al más bajo. Se dice que este dibujo es *2Hit@k* si los k factores más altos incluyen ambas categorías a mezclar. El caso especial *2Hit@2* mide la proporción de dibujos que son más similares a las dos categorías a mezclar que a cualquier otra categoría, mientras *2Hit@3* mide la proporción de dibujos que tienen las dos categorías a mezclar en los primeros tres lugares.

Las tasas *2Hit@2* en el cuadro 7.5 indican que entre un cuarto y un tercio de los dibujos de la base de datos y del espacio latente se parecen más a las categorías a mezclar que a las otras categorías. La técnica de autocompletado obtuvo una tasa arriba del 50 % pero las técnicas de interpolación y reconstrucción no. La filtración obtuvo una tasa del 79 %, lo cual sugiere que la proporción de dibujos *2Hit@2* en el espacio latente es alrededor de 1 en cada 125 dibujos. La búsqueda simple obtuvo una tasa *2Hit@2* cerca de 100 %, mientras que la búsqueda ancha obtuvo una tasa

	Clasificador			
	Validación		Prueba	
Técnica	2Hit@2	2Hit@3	2Hit@2	2Hit@3
Base de datos	23.3±0.3	69.8±1.2	25.7±1.3	73.4±1.0
Espacio latente	32±4	79±5	29.7±0.8	81.7±0.3
Interpolación	39±7	82±7	36.6±1.1	86.0±1.0
Reconstrucción	37±5	85±4	34±2	89.0±1.5
Autocompletado	60±14	85.6±10.7	46±9	88±4
Filtración	**79±6**	**95±2**	**51±3**	**93.1±1.0**
Búsqueda simple	**97.7±1.7**	**99.85±0.11**	**69±27**†	**97.9±1.8**
Búsqueda ancha	**81±10**	**94±4**	**68±6**	**93±4**

Cuadro 7.5: Promedio de las tasas *2Hit@k* en los tres experimentos y su desviación estándar. †Las tasas fueron 86, 31 y 91 en los tres experimentos.

más cercana a la de la filtración. La métrica *2Hit@3* presenta un comportamiento muy parecido, solo más sesgado hacía el 100 %.

Con el clasificador de prueba, la métrica *2Hit@3* no cambió mucho. Sin embargo, con la métrica *2Hit@2* las técnicas de filtración, búsqueda simple y búsqueda ancha no presentaron tasas tan altas en comparación con el clasificador de validación, aunque con ambos clasificadores las tasas para estas técnicas son mayores que las de las técnicas que no incorporan información del clasificador. Esto parece sugerir que los dibujos encontrados por técnicas que incorporan retroalimentación de un clasificador, están fuertemente personalizados a rasgos que el clasificador de prueba no necesariamente reconoce.

Desviación estándar de los vectores latentes.

Para aquellos procesos que producen vectores latentes, una medida sencilla de la diversidad de los dibujos producidos es la desviación estándar de sus vectores latentes $\sigma_{\boldsymbol{v}}$. En específico, la métrica reportada en esta sección es el promedio entre todas las

Técnica	σ_z
Base de datos	0.984±0.004
Interpolación	0.987±0.001
Reconstrucción	1.014±0.006
Autocompletado	0.996±0.001
Filtración	**0.984±0.002**
Búsqueda simple	**0.82±0.02**
Búsqueda ancha	**0.945±0.009**

Cuadro 7.6: Media de la desviación estándar de los vectores latentes.

desviaciones estándar calculadas de manera independiente para cada dimensión del vector latente. Recordemos que el modelo *sketch-rnn* se entrena[11] para codificar los dibujos de la base de datos en vectores latentes con una distribución lo más cercana posible a $\mathcal{N}(0, 1)$. Si el valor $\sigma_{\boldsymbol{v}}$ de una técnica generativa se acerque al valor $\sigma_{\boldsymbol{v}}$ de la base de datos, podemos esperar que la diversidad visual también sea similar.

El valor $\sigma_{\boldsymbol{v}}$ de la técnica de búsqueda simple fue el más bajo, un hecho que corresponde a la limitada diversidad visual que observamos en sus dibujos en comparación a las otras técnicas[12]. El valor $\sigma_{\boldsymbol{v}}$ de la técnica de búsqueda ancha fue mayor que el de la búsqueda simple pero a la vez menor que el de la base de datos. Los valores $\sigma_{\boldsymbol{v}}$ de las técnicas de interpolación y filtración fueron muy cercanos al de la base de datos. Sin embargo, los valores $\sigma_{\boldsymbol{v}}$ de las técnicas de reconstrucción y autocompletado fueron mayores que el de la base de datos, lo cual sugiere que la diversidad visual de los dibujos generados por estas técnicas es mayor que la de los dibujos de la base de datos.

[11]El hecho de que las desviaciones estándar no estuvieran más cercanas a 1 se debe en parte a los hiperparámetros del modelo *sketch-rnn*, que ponen menos importancia en esa pérdida. El apéndice B contiene más detalles.

[12]Los VAE solamente utilizan una fracción de las dimensiones de su espacio latente, lo cual puede explicar el hecho de que esta desviación estándar no sea más cercana a 0 (Engel et al., 2017).

7.3.3. Encuesta

Se realizó una encuesta para cuantificar qué tanto consideran los humanos una mezcla visual dibujos tomados aleatoriamente del espacio latente y los que crean las técnicas de interpolación, reconstrucción, autocompletado, filtración, búsqueda simple y búsqueda ancha. La encuesta se llevó a cabo a través de una página web. Cada participante evaluó un máximo de 60 dibujos de mezclas visuales seleccionados al azar de un conjunto de 420 dibujos; este conjunto está compuesto por 20 dibujos[13] de cada una de las 7 técnicas por cada uno de los 3 experimentos. Cada dibujo evaluado por el participante fue presentado junto con dos dibujos adicionales, cada uno representando una de las dos categorías de la mezcla visual correspondiente. Cada dibujo evaluado por el participante vino acompañado con las siguientes instrucciones:

> *Por favor, analiza los tres dibujos y elige la respuesta que mejor coincida con su opinión.*
> [los dibujos adicionales, etiquetados con las letras «A» y «B»]
> [el dibujo a evaluar, etiquetado con la letra «C»]
> *Declaración: «El dibujo C representa una mezcla visual entre las dos categorías de objetos mostradas en los dibujos A y B.»*
> [5 opciones]

Las 5 opciones siguieron una escala tipo Likert: «totalmente en desacuerdo», «en desacuerdo», «ni en acuerdo ni en desacuerdo», «de acuerdo» y «totalmente de acuerdo».

Un total de 142 personas, tanto hombres como mujeres de entre 15 y 53 años de edad, respondieron la encuesta durante un periodo de 3 días. El 72 % de los

[13] Seleccionados al azar. Para la filtración, se usaron los 20 mejores dibujos, según el clasificador de similitudes, de un conjunto de 100,000 generado al azar.

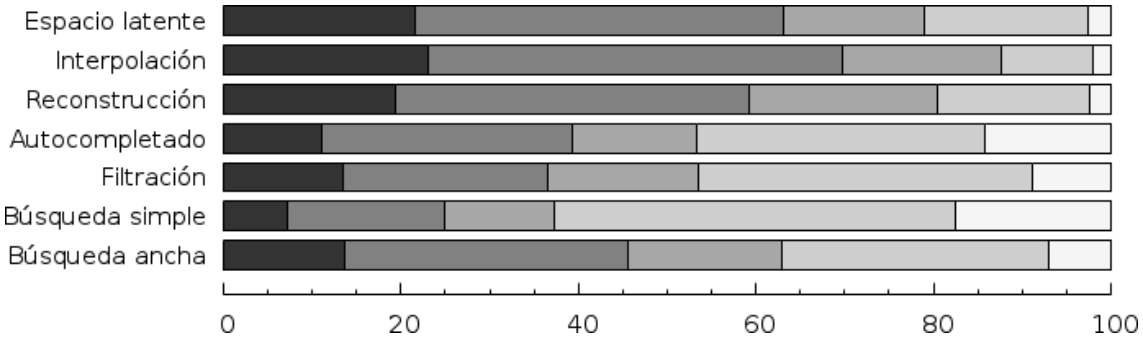


Figura 7.20: Porcentaje de respuestas recibidas por técnica de «totalmente en desacuerdo» (izquierda, oscuro) a «totalmente de acuerdo» (derecha, claro) para la mezcla visual entre CASA y LAVADORA.

participantes terminaron de evaluar los 60 dibujos; se recolectaron alrededor de 6,000 evaluaciones en total. Los participantes tardaron 8 segundos en promedio para evaluar cada dibujo.

Experimento 1

La figura 7.20 muestra los porcentajes de cada respuesta recibida para dibujos de mezclas visuales entre CASA y LAVADORA. La proporción de todas las respuestas positivas («totalmente de acuerdo» o «de acuerdo») fue mayor que la proporción de todas las respuestas negativas («totalmente en desacuerdo» o «en desacuerdo») para las técnicas de autocompletado, filtración y búsqueda simple (7 %, 10 % y 38 % mayor). Lo opuesto ocurrió para el resto de las técnicas.

Experimento 2

La figura 7.21 muestra los porcentajes de cada respuesta recibida para dibujos de mezclas visuales entre PATO y PATÍN. La proporción de todas las respuestas positivas fue mayor que la proporción de todas las respuestas negativas para las técnicas de

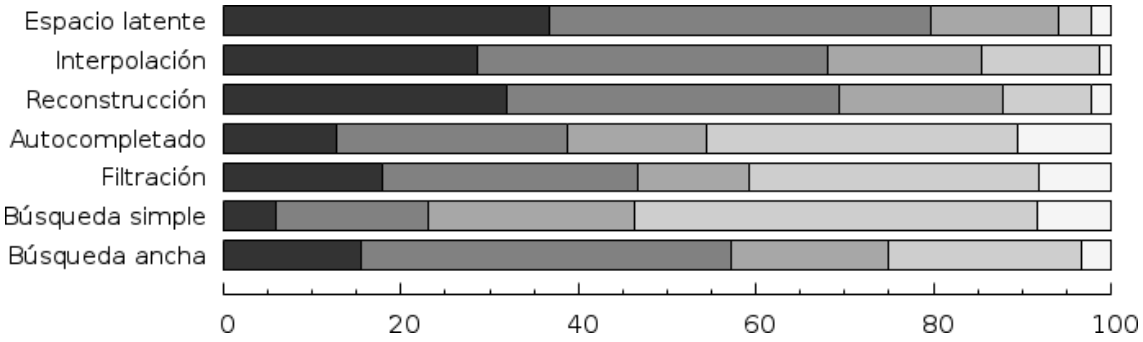


Figura 7.21: Porcentaje de respuestas recibidas por técnica para la mezcla visual entre PATO y PATÍN.

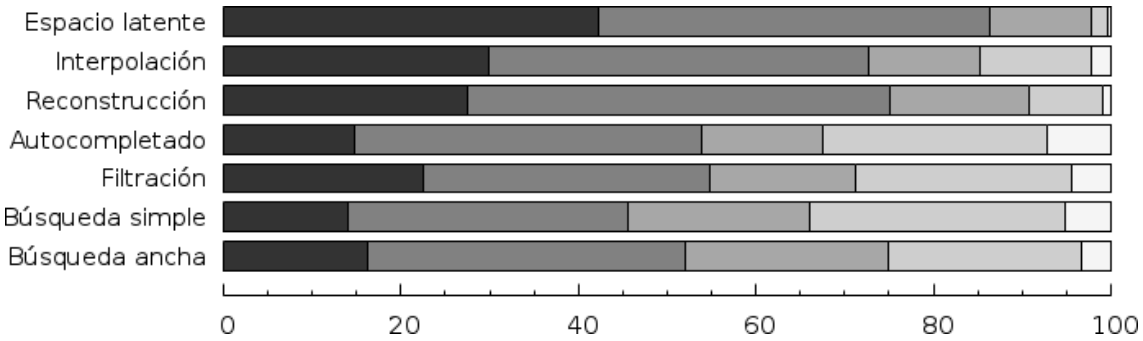


Figura 7.22: Porcentaje de respuestas recibidas por técnica para la mezcla visual entre BUZÓN y BOLSA.

autocompletado y búsqueda ancha (7 % y 32 % mayor). Lo opuesto ocurrió para todas las otras técnicas.

Experimento 3

La figura 7.22 muestra los porcentajes de cada respuesta recibida para dibujos de mezclas visuales entre BUZÓN y BOLSA. La proporción de todas las respuestas positivas fue menor que la proporción de todas las respuestas negativas en todas las técnicas.

Prueba t de varianzas desiguales Welch

La prueba t de varianzas desiguales de Welch (Welch, 1947) es una prueba paramétrica para comparar la distribución de dos conjuntos de valores y es adecuada para respuestas tipo Likert (Sullivan y Artino Jr, 2013). La prueba empieza con la hipótesis nula de que los dibujos generados por una técnica son considerados mezclas visuales igual o menos frecuentemente que dibujos generados por una segunda técnica. El resultado es una probabilidad p de obtener los resultados que se obtuvieron dada la hipótesis nula; una probabilidad baja es evidencia para rechazar esta hipótesis.

La prueba presenta evidencia para rechazar la hipótesis nula ($p < 0.001$) cuando se comparan las técnicas de autocompletado, filtración, búsqueda simple y búsqueda ancha con las técnicas de interpolación y reconstrucción o un muestreo aleatorio del espacio latente. Además presenta evidencia para rechazar la hipótesis nula ($p < 0.01$) cuando se compara la técnica de búsqueda simple con las técnicas de autocompletado, filtración y búsqueda ancha[14]. La prueba no presenta evidencia para rechazar la hipótesis nula cuando se comparan las técnicas de autocompletado, filtración y búsqueda ancha entre sí.

En general, los dibujos generados por la técnica de autocompletado y por las nuevas técnicas propuestas (filtración, búsqueda simple y búsqueda) fueron considerados mezclas visuales más frecuentemente que los generados por interpolación, reconstrucción y los generados al azar del espacio latente. En 2 de los 3 experimentos

[14]Con la excepción de la comparación entre la técnica de búsqueda simple y la de autocompletado en el tercer experimento ($p = 0.5$).

los dibujos generados por autocompletado y búsqueda simple fueron considerados mezclas visuales más frecuentemente que no. Estas conclusiones generalmente coinciden con el análisis realizado con el clasificador de similitudes en la sección anterior, con la excepción de que los participantes de la encuesta consideran más una mezcla visual los dibujos hechos por la técnica de autocompletado en comparación a este clasificador de similitudes.

Capítulo 8

Conclusiones

8.1. Resumen

En esta tesis se abordó el problema de mezcla visual usando modelos de aprendizaje de máquina, en específico autocodificadores variacionales. Se eligieron los dibujos de línea como un sub-dominio más manejable del arte visual. Una revisión del estado del arte permitió clasificar los pintores artificiales en cuatro categorías: programables, robóticos, basados en técnicas de aprendizaje de máquinas y enfocados en mezcla visual. Entre los retos que más afectan a los pintores artificiales que dibujan, resaltan su dependencia en anotaciones manuales, que pueden ser muy laboriosas, y sus limitaciones con respecto a los tipos de mezclas que pueden producir. También destaca la falta de una técnica cuantitativa automatizada para evaluar a qué nivel un dibujo representa mezcla visual.

Para tratar el problema de evaluación, se propuso un nuevo modelo clasificador basado en redes neuronales profundas. Se entrenó el modelo para medir la similitud entre un dibujo dado y múltiples categorías de objetos. Experimentos con dibujos de categorías de objetos conocidas y también desconocidas indicaron una modesta capacidad de evaluar similitudes por parte del modelo evaluativo. Este rendimiento resultó ser suficiente para poder aplicarlo como guía para la generación, aunque no se realizó experimentos para relacionar sus evaluaciones con la percepción humana.

El nuevo clasificador permitió el desarrollo de nuevas técnicas que remodelan la salida de un modelo generativo tipo autocodificador variacional usando información del clasificador. La nueva técnica de filtración integró retroalimentación del clasificador para seleccionar dibujos con características de mezcla visual. La nueva técnica de búsqueda simple aplicó el algoritmo del gradiente descendente para aprender un sub-espacio del espacio latente del autocodificador variacional que contenga mayormente mezclas, usando el clasificador como guía. La nueva técnica de búsqueda ancha agregó un segundo espacio latente a esta técnica para aumentar la variedad de los dibujos generados, también usando el clasificador como guía. En cuestión a las anotaciones manuales, todas las técnicas de remodelación analizadas en esta tesis eliminaron la dependencia en una base de datos de dibujos individualmente descompuestos en partes. En su lugar, las técnicas ocuparon una base de datos etiquetada a nivel categoría, en la cual solo se definieron similitudes entre categorías pero no se especificó cómo fueron similares. De esta manera, las nuevas técnicas no se limitaron a un conjunto fijo de partes como los pintores no basados en redes neuronales profundas.

Se realizaron tres experimentos con distintas mezclas intercategoría para comparar las técnicas de remodelación propuestas con las técnicas existentes. Se generó un conjunto grande de dibujos por cada técnica y se evaluó cada uno de estos dibujos con respecto a la definición de mezcla visual introducida en la sección 1.1. Se realizó la evaluación usando una versión del clasificador mencionado que fue entrenado con dibujos distintos a los usados para entrenar el clasificador usado para la remodelación. Los resultados de esta evaluación indican que las nuevas técnicas propuestas generan dibujos que el clasificador de evaluación considera una mezcla

visual con mayor frecuencia, esto en comparación a las técnicas de interpolación, reconstrucción, autocompletado y muestreo aleatorio del espacio latente.

Se realizó una encuesta para cuantificar qué tanto consideran los humanos una mezcla los dibujos generados por las varias técnicas mencionadas. Las participantes de la encuesta consideraron que dibujos generados por las técnicas de autocompletado, filtración, búsqueda simple y búsqueda ancha, son una mezcla visual más frecuentemente que dibujos generados por las técnicas de interpolación y reconstrucción o tomados aleatoriamente del espacio latente. En dos de los tres experimentos, la mayoría de los participantes de la encuesta consideraron que los dibujos generados por autocompletado y búsqueda simple son una mezcla visual.

8.2. Trabajo a futuro

A continuación se señalan varias líneas de investigación posibles.

- Realizar experimentos con más categorías. En particular, determinar la configuración de etiquetas mínima necesaria para realizar mezclas visuales. Explorar mezclas entre categorías que no comparten ninguna similitud.

- Adaptar el enfoque presentado en esta tesis a otros dominios. Música parece una opción natural dado que también utiliza secuencias.

- Adaptar el enfoque presentado en esta tesis a una red generativa antagónica, que también tiene un espacio latente.

- Integrar el clasificador a las técnicas de interpolación, reconstrucción y autocompletado, usando redes neuronales profundas para ajustar sus parámetros

con fines de mejorar su frecuencia. La técnica de autocompletado parece particularmente prometedora.

- Realizar experimentos para evaluar si la búsqueda ancha puede transformar cualquier dibujo en una mezcla que conserve una similitud al dibujo original. En teoría el modelo debería poder realizar esta tarea en un solo paso, en contraste con el enfoque iterativo de *latent constraints* (Engel et al., 2017).

- Modificar el modelo ACAI (Berthelot et al., 2018) reemplazando la red generativa antagónica por el nuevo modelo evaluativo para evaluar el efecto de integrar esta información durante el entrenamiento en lugar de hacerlo posteriormente.

- Realizar más estudios con humanos para validar las opiniones subjetivas presentadas en esta tesis, por ejemplo para estudiar hasta qué nivel el concepto de similitud modelado por el clasificador corresponde a la percepción humana de similitud.

Bibliografía

Gauvin A Bailey. *Art of Colonial Latin America*. Phaidon Press, 2005.

D. Berthelot, C. Raffel, A. Roy, y I. Goodfellow. Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer. *ArXiv e-prints*, July 2018.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. En *Proceedings of COMPSTAT'2010*, páginas 177–186. Springer, 2010.

Matthew R Boutell, Jiebo Luo, Xipeng Shen, y Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, y Samy Bengio. Generating sentences from a continuous space. En *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, páginas 10–21, 2016.

Mark Brown. 'New Rembrandt' to be unveiled in Amsterdam, 2016. URL `http://www.theguardian.com/artanddesign/2016/apr/05/new-rembrandt-to-be-unveiled-in-amsterdam`.

William Andrew Carman. *Exploring Place through Sketch Drawing*. PhD thesis, The University of Georgia, 2008.

Linda Carson. *Drawing Accuracy, Quality and Expertise*. PhD thesis, University of Waterloo, 2012.

Yajing Chen, Shikui Tu, Yuqi Yi, y Lei Xu. Sketch-pix2seq: a model to generate sketches of multiple categories. *arXiv preprint arXiv:1709.04121*, 2017.

Harold Cohen. How to draw three people in a botanical garden. En *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI)*, páginas 846–855, 1988.

Simon Colton. *The Painting Fool: Stories from Building an Automated Painter*, pages 3–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-31727-9.

Joao M Cunha, Joao Gonçalves, Pedro Martins, Penousal Machado, y Amılcar Cardoso. A Pig, an Angel and a Cactus Walk Into a Blender: A Descriptive Approach to Visual Blending. En *Proceedings of the 8th International Conference on Computational Creativity (ICCC)*, 2017.

Andrew M Dai y Quoc V Le. Semi-supervised sequence learning. En *Advances in neural information processing systems*, páginas 3079–3087, 2015.

Oliver Deussen, Thomas Lindemeier, Sören Pirk, y Mark Tautzenberger. Feedback-guided stroke placement for a painting machine. En *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, CAe '12, páginas 25–33, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association. ISBN 978-1-4503-1584-5.

Scott Draves. Electric sheep (1999). `https://electricsheep.org`, 1999.

Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, y Marc Alexa. Sketch-based shape retrieval. *ACM Trans. Graph.*, 31(4):31–1, 2012.

Ahmed M. Elgammal, Bingchen Liu, Mohamed Elhoseiny, y Marian Mazzone. CAN: creative adversarial networks, generating "art" by learning about styles and deviating from style norms. *CoRR*, abs/1706.07068, 2017. URL `http://arxiv.org/abs/1706.07068`.

Jesse Engel, Matthew Hoffman, y Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models. *CoRR*, abs/1711.05772, 2017. URL `http://arxiv.org/abs/1711.05772`.

Mark Everingham, Henk Muller, y Barry Thomas. Evaluating image segmentation algorithms using the pareto front. En *European Conference on Computer Vision*, páginas 34–48. Springer, 2002.

Gilles Fauconnier y Mark Turner. Conceptual integration networks. *Cognitive science*, 22(2):133–187, 1998.

Gilles Fauconnier y Mark Turner. *The way we think: Conceptual blending and the mind's hidden complexities*. Basic Books, 2008.

LA Gatys, AS Ecker, y M Bethge. A neural algorithm of artistic style. *Nature Communications*, 2015.

Leon A. Gatys, Alexander S. Ecker, y Matthias Bethge. Image style transfer using convolutional neural networks. En *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Ian Goodfellow, Yoshua Bengio, y Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, y Daan Wierstra. Draw: A recurrent neural network for image generation. En *International Conference on Machine Learning*, páginas 1462–1471, 2015.

David Ha y Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, y Alexander Lerchner. Beta-vae: Learning basic visual concepts with a constrained variational framework. En *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

David Mautner Himmelblau. *Applied nmiscar programming*. McGraw-Hill Companies, 1972.

Sepp Hochreiter y Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kurt Hornik, Maxwell Stinchcombe, y Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, y Zhihao Fang. Towards the automatic anime characters creation with generative adversarial networks. *CoRR*, abs/1708.05509, 2017. URL `http://arxiv.org/abs/1708.05509`.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, y Li Fei-Fei. Large-scale video classification with convolutional neural networks. En *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, páginas 1725–1732, 2014.

Junghyun Kim. *Home Drawing Events: A Case Study of a Boy and His Dragons*. PhD thesis, The Pennsylvania State University, 2015.

Diederik P Kingma y Jimmy Ba. Adam: A method for stochastic optimization. En *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Diederik P Kingma y Max Welling. Auto-encoding variational bayes. En *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Alexandra Kleeman. Cooking with Chef Watson, I.B.M.'s Artificial-Intelligence AppA, 11 2016. URL `https://www.newyorker.com/magazine/2016/11/28/cooking-with-chef-watson-ibms-artificial-intelligence-app`.

Gregory Koch, Richard Zemel, y Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. En *ICML Deep Learning Workshop*, volumen 2, 2015.

Johann W Kolar, J Biela, y J Minibock. Exploring the pareto front of multi-objective single-phase pfc rectifier design optimization-99.2 % efficiency vs. 7kw/din 3 power density. En *Power Electronics and Motion Control Conference, 2009. IPEMC'09. IEEE 6th International*, páginas 1–21. IEEE, 2009.

Solomon Kullback y Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Nicholas Lambert, William Latham, y Frederic Fol Leymarie. The emergence and growth of evolutionary art: 1980–1993. En *ACM SIGGRAPH 2013 Art Gallery*, páginas 367–375. ACM, 2013.

Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, y Thomas S Huang. Interactive facial feature localization. En *European conference on computer vision*, páginas 679–692. Springer, 2012.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, y C Lawrence Zitnick. Microsoft coco: Common objects in context. En *European conference on computer vision*, páginas 740–755. Springer, 2014.

Z. C. Lipton, D. C. Kale, C. Elkan, y R. Wetzel. Learning to Diagnose with LSTM Recurrent Neural Networks. En *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Edward Lucie-Smith. *Latin American art of the 20th century*. Thames & Hudson, 2004.

Andrew L Maas, Awni Y Hannun, y Andrew Y Ng. Rectifier nmiscarities improve neural network acoustic models. En *Proc. icml*, volumen 30, página 3, 2013.

Benoit B Mandelbrot y Roberto Pignoni. *The fractal geometry of nature*. WH freeman New York, 1983.

Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, y Ruslan Salakhutdinov. Generating images from captions with attention. En *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

G. McCaig, S. DiPaola, y L. Gabora. Deep Convolutional Networks as Models of Generalization and Blending Within Visual Creativity. En *Proceedings of the 8th International Conference on Computational Creativity (ICCC)*, 2016.

Steven Mithen. The prehistory of the mind: a search for the origins of art, religion and science. *London: Thames and Hudson*, 1996.

Alexander Mordvintsev, Christopher Olah, y Mike Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June*, 20:14, 2015.

Colin Morris. Finding bad flamingo drawings with recurrent neural networks. `http://colinmorris.github.io/blog/bad_flamingos`, 2017.

Heinz Mühlenbein, M Schomisch, y Joachim Born. The parallel genetic algorithm as function optimizer. *Parallel computing*, 17(6-7):619–632, 1991.

Kazushi Mukaiyama. Shizuka, the artificial painter version 7 and human imagination with computing. *The 27th Annual Conference of the Japanese Society for Artificial Intelligence*, 2013.

Kazushi Mukaiyama. Shizuka, the Artificial Painter, 2016. URL `http://www.creativeai.net/posts/nY4sgvEb84eNXGPwp/shizuka-the-artificial-painter`.

Vinod Nair y Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. En *Proceedings of the 27th international conference on machine learning (ICML-10)*, páginas 807–814, 2010.

Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, y George Toderici. Beyond short snippets: Deep networks for video classification. En *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Anh Nguyen, Jason Yosinski, y Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 427–436, 2015.

David Norton, Derrall Heath, y Dan Ventura. Autonomously creating quality images. En *Proceedings of the 2nd International Conference on Computational Creativity*, volumen 301, páginas 10–15, 2011.

David Norton, Derrall Heath, y Dan Ventura. Finding creativity in an artificial artist. *The Journal of Creative Behavior*, 47(2):106–124, 2013.

Ciara Nugent. The Painter Behind These Artworks Is an AI Program. Do They Still Count as Art?, 8 2018. URL `http://time.com/5357221/obvious-artificial-intelligence-art/`.

Augustus Odena, Christopher Olah, y Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. En *Proceedings of the 34th International Conference on Machine Learning*, 2017.

SK Pal y S Mitra. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks*, 3(5):683–697, 1992.

Francisco C Pereira y Amílcar Cardoso. The boat-house visual blending experience. En *Proceedings of the Symposium for Creativity in Arts and Science of AISB 2002*, 2002.

Francisco C Pereira y Amílcar Cardoso. Experiments with free concept generation in divago. *Knowledge-Based Systems*, 19(7):459–470, 2006.

Rafael Pérez et al. *Creatividad Computacional*. Grupo Editorial Patria, 2015.

David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation, 2011.

Alec Radford, Luke Metz, y Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Vilayanur S Ramachandran. *The tell-tale brain: A neuroscientist's quest for what makes us human*. WW Norton & Company, 2012.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, y Honglak Lee. Generative adversarial text to image synthesis. En *Proceedings of The 33rd International Conference on Machine Learning. Vol. 3. 2016.*, 2016.

Paulo Ribeiro, Francisco C Pereira, Bruno F Marques, Bruno Leitão, Amílcar Cardoso, II Polo, y P de Marrocos. A model for creativity in creature generation. En *GAME-ON*, página 175, 2003.

Juan J Romero y Penousal Machado. *The art of artificial evolution: a handbook on evolutionary art and music*. Springer Science & Business Media, 2007.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, y Xi Chen. Improved techniques for training gans. En *Advances in Neural Information Processing Systems*, páginas 2234–2242, 2016.

Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, y James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):119, 2016.

Nadja Sayej. Vincent van bot: the robots turning their hand to art, 2016. URL `https://www.theguardian.com/artanddesign/2016/apr/19/robot-art-competition-e-david-cloudpainter-bitpaintr`.

David Schneider. Build Your Own Google Neural Synthesizer, 06 2018. URL `https://spectrum.ieee.org/geek-life/hands-on/build-your-own-google-neural-synthesizer`.

Karl Sims. *Artificial evolution for computer graphics*, volumen 25. ACM, 1991.

Celestino Soddu. New naturality: a generative approach to art and design. *Leonardo*, 35(3):291–294, 2002.

J. T. Springenberg. Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. En *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Gail M Sullivan y Anthony R Artino Jr. Analyzing and interpreting data from likert-type scales. *Journal of graduate medical education*, 5(4):541–542, 2013.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, y Zbigniew Wojna. Rethinking the inception architecture for computer vision. En *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 2818–2826, 2016.

Wei Ren Tan, Chee Seng Chan, Hernán E Aguirre, y Kiyoshi Tanaka. Artgan: Artwork synthesis with conditional categorical gans. En *Image Processing (ICIP), 2017 IEEE International Conference on*, páginas 3760–3764. IEEE, 2017.

Patrick A Tresset y F Leymarie. Sketches by paul the robot. En *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, páginas 17–24. Eurographics Association, 2012.

Tatsuo Unemi. A design of multi-field user interface for simulated breeding. En *Proceedings of the third Asian Fuzzy Systems Symposium*, pages 489–494, 1998.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. En *Advances in Neural Information Processing Systems*, páginas 4790–4798, 2016.

David A Van Veldhuizen y Gary B Lamont. Evolutionary computation and convergence to a pareto front. En *Late breaking papers at the genetic programming 1998 conference*, páginas 221–228, 1998.

Bernard L Welch. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

Tom White. Sampling generative networks: Notes on a few effective techniques. arxiv preprint. *arXiv preprint arXiv:1609.04468*, 2016.

Ping Xiao y Simo Linkola. Vismantic: Meaning-making with images. En *Proceedings of the Sixth International Conference on Computational Creativity*, páginas 158–165. International Association for Computational Creativity, 2015.

Yandan Zhao, Xiaogang Jin, Yingqing Xu, Hanli Zhao, Meng Ai, y Kun Zhou. Parallel style-aware image cloning for artworks. *IEEE transactions on visualization and computer graphics*, 21(2):229–240, 2015.

Apéndice A

Base de datos

Categoría	Promedio	Categoría	Promedio	Categoría	Promedio
Casa	34±11	Pato	63±19	Buzón	34±11
Lavadora	54±16	Patín	60±17	Bolsa	44±14
Televisión	49±15	Calcetín	32±10	Almohada	29±10
Dona	50±12	Lentes	46±12	Arcoiris	35±17
Montaña	25±10	Martillo	42±16	Marcador	28±14

Cantidad de trazos promedio por dibujos de varias categorías.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ x_1 & y_1 & p_{a_1} & p_{b_1} & p_{c_1} \\ x_2 & y_2 & p_{a_2} & p_{b_2} & p_{c_2} \\ & & \dots & & \\ x_n & y_n & p_{a_n} & p_{b_n} & 1 \end{bmatrix} \tag{A.1}$$

La matriz de la ecuación A.1 muestra la forma general de una secuencia de n trazos, cada uno representado por un vector $[x, y, p_1, p_2, p_3]$. Las primeras dos dimensiones del vector representan el desplazamiento del lápiz digital, mientras que las otras tres dimensiones forman un vector *one-hot* que indica si el lápiz se encuentra colocado o levantado del papel virtual. La última dimensión p_c indica el fin del dibujo con un valor de 1 y de lo contrario lleva un valor de 0. El desplazamiento es relativo al trazo anterior y representa un movimiento en un espacio bidimensional con origen $(0, 0)$.

Apéndice B

Configuración y entrenamiento

Todos los parámetros presentados a continuación fueron tomados directamente de la implementación original de *sketch-rnn* o fueron inspirados en ella[1]. En los cuadros a continuación, el símbolo $U(\text{mín}, \text{máx})$ se refiere a una distribución uniforme entre los valores mínimo y máximo señalado.

Tanto el clasificador como el modelo *sketch-rnn* ocuparon el mismo esquema de acrecentamiento de datos encontrado en el código original de *sketch-rnn*, el cual cuenta con dos partes. El primero escala o estrecha el dibujo, modificando su tamaño, mientras el segundo elimina aleatoriamente puntos de la secuencia. El código original del modelo evita que se elimine el último punto en una curva[2]. El clasificador aplicó

Entidad	Parámetro	Valor
Codificador	Nodos LSTM del RNN *fw* y *bw*	256
	Probabilidad de deserción recurrente	0.9
	Dimensionalidad del espacio latente	128
Decodificador	Nodos LSTM	512
	Probabilidad de deserción recurrente	0.9
Optimizador	Peso de pérdida KL	0.01
	Peso de pérdida KL máximo	0.5
	Tasa de aumento de pérdida KL	0.99995
	Pasos de entrenamiento	100,000 y 200,000†
	Pasos para que se establece la pérdida	60,000 y 120,000†

Parámetros de configuración para el modelo *sketch-rnn*. †Para modelos entrenados con una y dos categorías respectivamente.

[1] `https://github.com/tensorflow/magenta/tree/master/magenta/models/sketch_rnn`

Entidad	Parámetro	Valor
RNN *fw* y *bw*	Nodos LSTM	256
	Probabilidad de deserción recurrente	0.9
	Peso asignado a estados intermedios	0.33
	Pasos de entrenamiento	100,000
	Pasos para que se establece la pérdida	40,000
Aumento de datos	Factor de escala aleatorio	0.15
	Probabilidad de cortar trazos	0.1
	Probabilidad de insertar trazos	0.33
	Factor de tamaño máximo de inserción	0.33

Parámetros de configuración para el clasificador.

Entidad	Parámetro	Valor
Interpolación	Punto de interpolación t	U(0.3, 0.7)
Autocompletado	Recortado del dibujo original	$U(0.1, 0.3)^{\dagger}$
	Aumento de longitud máxima	1.5 veces
Todas	Temperatura τ	$0.01^{\ddagger}$

Parámetros de configuración para las técnicas de interpolación, reconstrucción y autocompletado. †El recorte permite más oportunidades de reemplazar partes del dibujo con sus propios trazos. ‡Este nivel de temperatura hace que el dibujo represente más frecuentemente al vector latente proporcionado, sin permitir mucha exploración.

este esquema antes de pasar el dibujo a su propio esquema de acrecentamiento definido en la sección 5.2. En el paso de decodificar un vector latente a un dibujo con *sketch-rnn*, el proceso de tomar muestras del MDM puede ser controlado por un factor llamado la temperatura τ. La temperatura escala los parámetros tanto de la distribución usada para determinar si el trazo será el último de la curva o del dibujo como los de las distribuciones gaussianas usadas para determinar la posición del trazo. Cuando la temperatura es 0 el muestreo vuelve determinista y siempre devolverá en el punto más probable (Ha y Eck, 2017). El proceso de tomar muestras

[2]Recordar que el término «curva» se refiere a una secuencia de trazos conectados por líneas visibles.

Parámetro	Valor
Umbral	Mejor 1 %
Muestras consideradas	100,000
Distribución fuente	$\mathcal{N}(0, 1)$
Temperatura τ	$0.1^{\dagger}$

Parámetros de configuración para la técnica de filtración. †Este nivel de temperatura permite más variedad en los dibujos generados.

Parámetro	Valor
Capas	5
Neuronas por capa	512
Pasos de entrenamiento	30,000
Pasos para que se establece la pérdida	10,000
Distribución del muestreo	$U(-3, 3)^{\dagger}$
Temperatura τ	codicioso

Parámetros de configuración para el modelo amortizador. †Para proporcionar mayor cobertura en contraste con una distribución gaussiana.

del MDM puede funcionar en un modo especial llamado «codicioso». Es este modo, el muestreo del MDM normalmente usado para determinar los datos del siguiente trazo es simplemente reemplazado por la toma del valor más probable. El efecto neto es que cada vector latente solamente corresponde a un solo dibujo. Este modo fue activado para agilizar el proceso de conversión de un vector latente a una secuencia de trazos.

Todos los modelos mencionados en esta sección ocuparon el optimizador *Adam* (Kingma y Ba, 2015) con un tamaño de lote de 100. La tasa de aprendizaje fue variable, siguiendo la fórmula:

$$\text{Tasa de aprendizaje} = 0.00001 + \left((0.001 - 0.00001) \times 0.9999^{t}\right) \tag{B.1}$$

donde t es el número de pasos de entrenamiento realizados hasta el momento. En

Entidad	Parámetro	Valor
Búsqueda simple	Peso de pérdida KL	0.2
	Límite mínima de pérdida KL	0.01
	Pasos de entrenamiento	50,000
	Pasos para que se establece la pérdida	10,000
	Temperatura τ	0.1
Búsqueda ancha	Capas	2
	Neuronas por capa	64
	Peso de pérdida KL	0.2
	Límite mínima de pérdida KL	0.01
	Peso de pérdida probabilística	0.2
	Pasos de entrenamiento	100,000
	Pasos para que se establece la pérdida	30,000
	Temperatura τ	0.1

Parámetros de configuración para la técnica de búsqueda simple y búsqueda ancha.

el modelo *sketch-rnn* el peso de la pérdida KL también fue variable, siguiendo una fórmula similar:

$$\text{Peso KL} = f - \big((f - i) \times 0.9995^{t}\big) \tag{B.2}$$

donde i es la tasa inicial, f la tasa deseada o final y t el número de pasos de entrenamiento realizado hasta el momento. Estas dos fórmulas se encuentran en el código original del modelo *sketch-rnn*.

Apéndice C

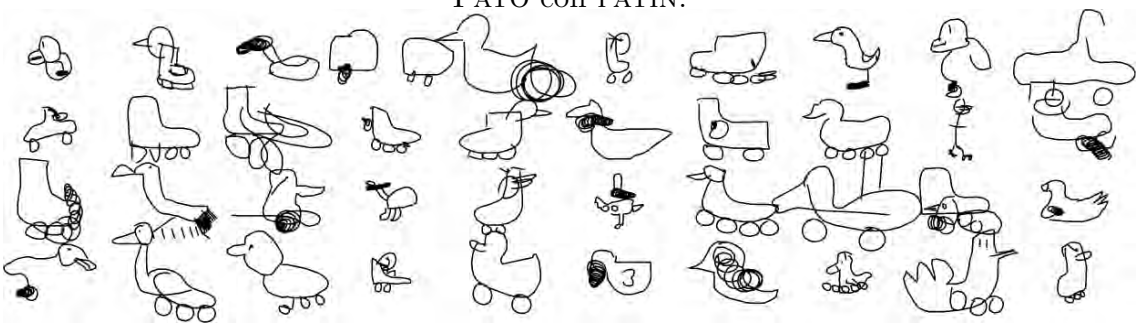
Muestras visuales adicionales

Casa con lavadora.

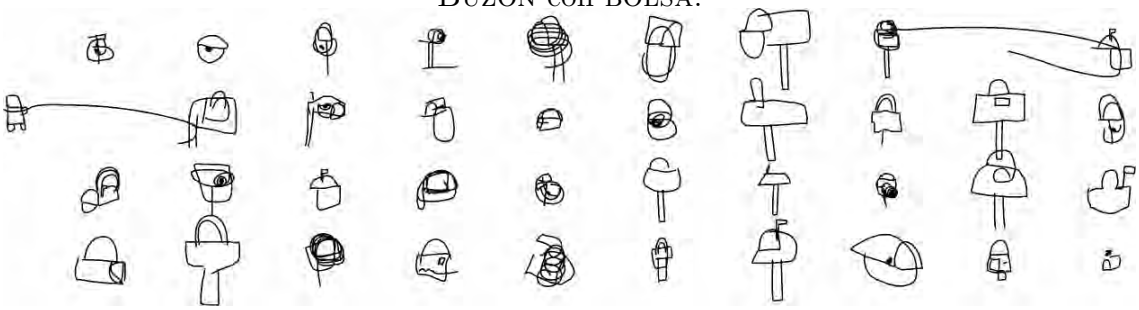
Buzón con bolsa.

Reconstrucción. Dibujos generados por la técnica de reconstrucción, seleccionados al azar.

Pato con patín.

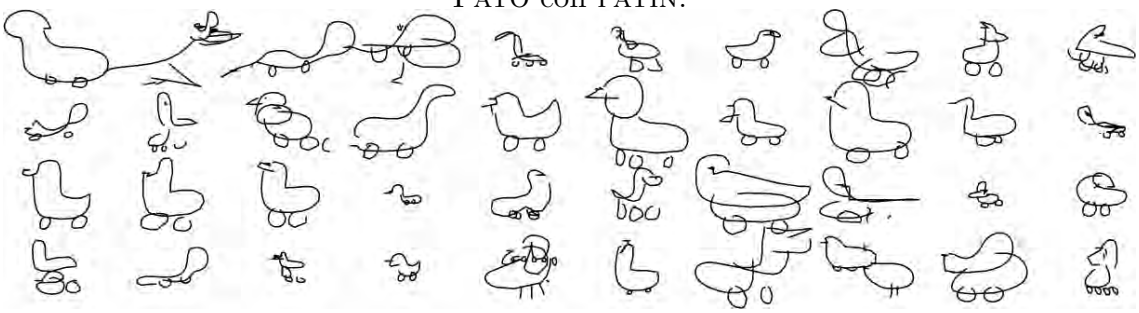


Buzón con bolsa.

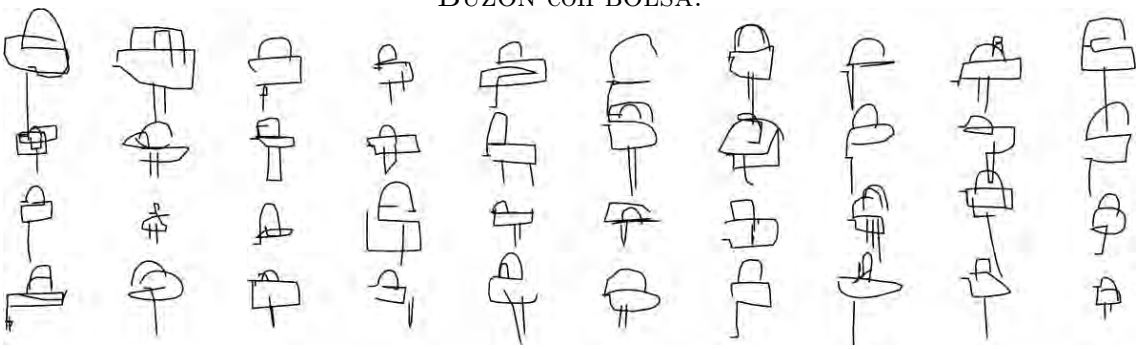


Autocompletado. Dibujos generados por la técnica de autocompletado, seleccionados al azar.

PATO con PATÍN.



BUZÓN con BOLSA.



Filtración. Dibujos de mezclas ordenados por entropía cruzada binaria multi-etiqueta ponderada y presentados en orden de izquierda a derecha y de arriba a abajo.

Dibujos reconstruidos de mezclas entre CASA y LAVADORA, autocompletados entre PATO y PATÍN y filtrados entre BUZÓN y BOLSA se encuentran en la sección 7.3.1.

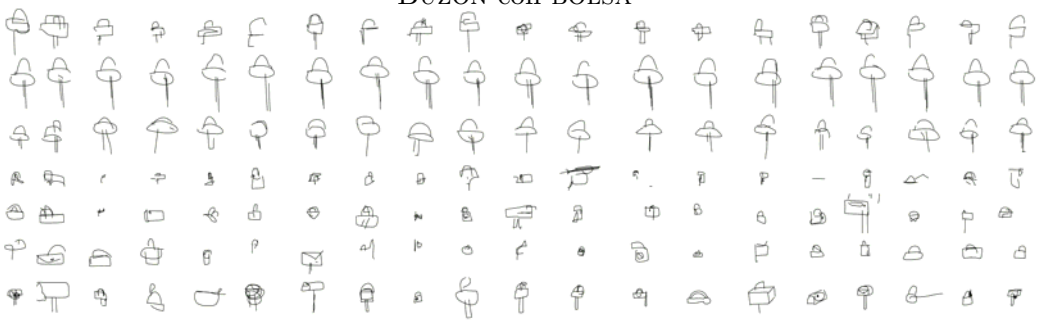
Apéndice D

Encuesta

Casa con lavadora

Pato con patín

Buzón con bolsa



Los 420 dibujos usados en la encuesta. Cada fila contiene 20 dibujos generados por una de las 7 técnicas, que son de arriba abajo: filtración, búsqueda simple, búsqueda ancha, muestra aleatoria del espacio latente, interpolación, reconstrucción, y autocompletado.