



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MEJORA DE LA GENERALIZACIÓN DE CLASIFICADORES CONVOLUCIONALES
YA ENTRENADOS, USANDO FEEDBACK VISUAL DE USUARIO

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN Y AL TITULO DE
INGENIERO CIVIL EN COMPUTACIÓN

ANDRÉS SEBASTIÁN FERRADA LAGOS

PROFESOR GUÍA:
JOSÉ MANUEL SAAVEDRA R.

MIEMBROS DE LA COMISIÓN:
PROF. INTEGRANTE: ALEXANDRE BERGEL
PROF. INTEGRANTE: BENJAMIN BUSTOS
PROF. EXTERNO : PABLO ROMAN - USACH

SANTIAGO DE CHILE
SEPTIEMBRE 2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN
POR: ANDRÉS SEBASTIÁN FERRADA LAGOS
FECHA: SEPTIEMBRE 2019
PROF. GUÍA: JOSÉ MANUEL SAAVEDRA R.

MEJORA DE LA GENERALIZACIÓN DE CLASIFICADORES CONVOLUCIONALES
YA ENTRENADOS, USANDO FEEDBACK VISUAL DE USUARIO

Dentro de los métodos de *Machine Learning*, las redes neuronales convolucionales han logrado sorprendentes resultados en los últimos años. Sin embargo, es difícil identificar cómo funcionan y cómo corregir errores puntuales en modelos ya entrenados.

El presente trabajo busca corregir modelos ya entrenados sin agregar más datos o cambiar la arquitectura subyacente. Esto es posible a través de establecer una comunicación entre el modelo y el usuario experto que permita mejorar el desempeño del modelo. Por un lado se generan visualizaciones de lo que el modelo considera relevante en la imagen de entrada, y por el otro, el usuario puede indicar si estas áreas son o no relevantes. Con este contraste se espera agregar información, aumentando la generalización del modelo, lo que se ve traducido en mejoras en la clasificación.

Para entregar tal información al modelo se estudian dos métodos. El primer método corresponde a editar las áreas seleccionadas con modelos generativos de imágenes (*image inpainting*), de forma de llenar las áreas seleccionadas con patrones distintos. El segundo método corresponde a plantear funciones de pérdida, las cuales castigan al modelo cuando este genere altas activaciones en las áreas consideradas como irrelevantes.

Como resultado de la evaluación del trabajo, se observa que los métodos de reemplazo resultan no ser los indicados, ya que tienen problemas para ajustarse a las áreas irrelevantes dado la arquitectura requerida por el algoritmo de visualización CAM (*Class Activation Mapping*). Por tal motivo, se plantea una segunda propuesta basada en adaptar una función de pérdida. Aquí es necesario considerar las diversas variables de forma de balancear el objetivo de clasificación con el objetivo de eliminar activaciones irrelevantes. Tal proceso concluye con la creación de PASA (Pérdida por Activación Selectiva Ajustada).

Se estudia el comportamiento del modelo en diversos conjuntos de datos. Los resultados indican que el método PASA logra cambiar las activaciones de forma satisfactoria, a la vez que corrige clasificaciones en las imágenes seleccionadas. Lamentablemente, la propuesta no logra producir cambios significativos en métricas de clasificación en el conjunto de prueba.

Al analizar las suposiciones iniciales se determina que el modelo estudiado si bien presenta características irrelevantes, estas no son del tipo que genera confusión en el conjunto de prueba. Lo que existe es una gran redundancia de características. También se logró determinar que es posible encontrar las características irrelevantes de forma visual, pero no a través de la propuesta CAM, sino a través de observar directamente las activaciones.

Por otro lado, si se conocen las características irrelevantes a priori, el método propuesto puede generar efectos positivos como se demuestra en el *dataset X-RAY*, sobre el que se logra una mejora significativa en las métricas de clasificación.

Agradecimientos

Primero agradecer a mis padres que han estado siempre apoyándome todo estos años y sin los cuales no estaría donde estoy.

También gracias a todas las personas que he conocido por mi paso por la universidad que han hecho de este viaje la experiencia que es. Especialmente *Queue* y toda la gente que conocí en La Radio Integral.

Por último gracias a José Saavedra mi profesor guía que me apoyo en este proceso siempre con sus oportunos consejos con los cuales se dio forma a este trabajo.

Tabla de Contenido

Introducción	1
1. Marco teórico	4
1.1. Definiciones	4
1.1.1. Convolución	4
1.1.2. Redes neuronales	6
1.1.3. Redes convolucionales	7
1.1.4. Algunas capas de las redes convolucionales	7
1.1.5. Clasificación	8
1.1.6. Arquitectura VGG	10
1.1.7. Características y filtros	10
1.1.8. Modelo de visualización CAM o <i>Class Activation Mapping</i>	12
1.1.9. <i>Data augmentation</i> o Aumento de datos	13
1.1.10. Modelos generativos	14
1.1.11. Métricas de clasificación	15
1.2. Problema	16
1.3. Resumen	17
2. Propuesta	18
2.1. Descripción de la propuesta	18
2.1.1. ¿Por qué son difíciles de entender las redes convolucionales?	18
2.1.2. Ayudando al modelo a aproximar la superficie objetivo	19
2.1.3. Extraer información del modelo: Entender el modelo a partir de visualizaciones	20
2.1.4. ¿Cómo entregar información al modelo?	22
2.1.5. Consideraciones cruciales sobre efectividad de la propuesta	23
2.1.6. Resultados esperados	23
2.2. Hipótesis	24
2.3. Preguntas de investigación	24
2.4. Objetivos	25
2.4.1. Objetivo General	25
2.4.2. Objetivos Específicos	25
2.5. Trabajo relacionado	25
2.6. Resumen	27
3. Diseño experimental	28

3.1.	Experimento en términos generales	28
3.2.	Preliminares	30
3.2.1.	<i>Dataset</i> a analizar	30
3.2.2.	Entrenamiento inicial	30
3.2.3.	Implementación y herramientas utilizadas	31
3.2.4.	Límite superior del desempeño del clasificador	33
3.2.5.	Selección de áreas irrelevantes	33
3.3.	Métodos similares para comparación	34
3.3.1.	Resultados	34
3.3.2.	Análisis modelos base y métodos similares	35
3.4.	Resumen	36
4.	Reajuste basado en reemplazos	38
4.1.	Resumen de propuesta con reemplazos	38
4.2.	Reemplazo generativo	39
4.3.	Reemplazo recorte aleatorio	40
4.4.	<i>Dropout</i> selectivo	40
4.5.	Resultados reemplazos	41
4.6.	Revisión de los supuestos	43
4.7.	Resumen	46
5.	Reajuste basado en funciones de pérdida	47
5.1.	Pérdida por activación selectiva (PAS)	47
5.1.1.	Selección de λ	48
5.1.2.	Aumento de las máscaras	49
5.1.3.	Detalle resultados PAS	50
5.2.	Pérdida por activación selectiva ajustada (PASA)	52
5.2.1.	Resultados pérdida PASA	55
5.3.	Resumen	58
6.	Resultados para diversos <i>datasets</i>	59
6.1.	Sketches	59
6.1.1.	Selección de máscaras	60
6.1.2.	Resultados	60
6.1.3.	Ánálisis	61
6.2.	Experimento Símbolos	62
6.2.1.	Resultados	62
6.2.2.	Ánálisis	63
6.3.	Experimento rayos X	64
6.3.1.	Resultados	65
6.3.2.	Ánálisis	65
6.4.	Resumen	67
7.	Discusión de resultados	68
7.1.	Exploración de los filtros de forma global	68
7.1.1.	¿Existen características irrelevantes?	69
7.1.2.	¿La inspección visual global encuentra las características irrelevantes?	70

7.1.3. ¿La visualización de CAM apunta a características no relevantes?	73
7.1.4. Efectos después de reajustar el clasificador	74
7.1.5. Efectos al reajustar enfocado a una característica irrelevante	79
7.2. Resumen	82
Conclusión	82
8. Anexos	90
8.1. Derivación fórmula gradiente	90
8.1.1. Gradiente entropía cruzada respecto a <i>softmax</i>	90
8.1.2. Gradiente de logits [3]	91
8.1.3. Gradiente de pesos logits	92
8.1.4. Gradiente de <i>global average pooling</i>	92
8.1.5. Gradiente de activación filtro	92
8.1.6. Gradiente de pesos <i>kernel</i> última capa	93
8.1.7. Formulas resultantes	93
8.2. Explicación general del código	93
8.2.1. Dependencias	93
8.2.2. ¿Cómo ejecutar código?	94
8.2.3. ¿Cómo conseguir los datos?	94
8.2.4. Estructura del código	94
8.2.5. Proceso para reproducir resultados	96

Índice de Tablas

1.1.	Matriz de confusión caso binario.	15
3.1.	Ejemplo <i>dataset</i> subconjunto de <i>IMAGENET</i>	30
3.2.	Resumen modelo después de entrenamiento inicial	31
3.3.	<i>Accuracy</i> modelo cota	33
3.4.	Ejemplo selección áreas irrelevantes.	33
3.5.	<i>Accuracy</i> métodos similares	34
3.6.	Reporte clasificación métodos similares.	35
3.7.	Visualización modelos base	35
4.1.	Ejemplo reemplazos generativos	39
4.2.	Ejemplo reemplazos con recortes	40
4.3.	<i>Accuracy</i> reemplazos	41
4.4.	Reporte clasificación reemplazos	41
4.5.	Visualización reemplazos	42
5.1.	Ejemplo de máscaras	49
5.2.	<i>Accuracy</i> PAS	50
5.3.	Reporte clasificación PAS	50
5.4.	Visualizaciones PAS	51
5.5.	Ruido en la pérdida y oscilaciones en la clasificación	52
5.6.	<i>Accuracy</i> PASA	55
5.7.	Reporte clasificación PASA	55
5.8.	Visualizaciones PASA	56
5.9.	Imágenes seleccionadas del modelo I con su activación final y área seleccionada	57
5.10.	Ruido en la pérdida y oscilaciones en la clasificación	57
6.1.	Ejemplo imágenes <i>dataset sketches</i>	59
6.2.	<i>Accuracy Sketches</i>	60
6.3.	Reporte clasificación Sketches	60
6.4.	Visualización Sketches	61
6.5.	Ejemplo datos Símbolos	62
6.6.	<i>Accuracy</i> símbolos	62
6.7.	Reporte clasificación Símbolos	63
6.8.	Visualizaciones Símbolos	63
6.10.	<i>Accuracy</i> Xray	65
6.9.	Ejemplo <i>dataset</i> Xray	65

6.11. Reporte clasificación Xray	65
6.12. Visualizaciones xray	66
7.1. Dos filtros que muestran patrones que el experto selecciona como no relevantes	72
7.2. Imágenes seleccionadas y filtros relevantes	74
7.3. Diferencias de promedios globales por característica antes y después de aplicar el método	76
7.4. Dos filtros que muestran patrones que el experto selecciona como no relevantes	81
7.5. Dos filtros que muestran patrones que el experto selecciona como no relevantes	81

Índice de Ilustraciones

1.1. Diagrama convolución	5
1.2. Diagrama red neuronal	6
1.3. Relación entre entropía cruzada y log-verosimilitud.	9
1.4. Imagen adversaria de clasificador entrenado en IMAGENET	11
1.5. Diagrama del funcionamiento visualización CAM	12
2.1. Representación de clasificador y datos	19
2.2. Representación de clasificador y datos	20
2.3. Ejemplo de los dos tipos de visualizaciones	21
2.4. Diagrama propuesta de reemplazos	22
3.1. Diagrama arquitectura imágenes animales	31
3.2. Ejemplo herramienta de selección de áreas irrelevantes	32
4.1. Diagrama propuesta en 4 etapas	39
4.2. Diagrama arquitectura	44
5.1. Selección de λ	48
5.2. Selección de cantidad máscaras	49
5.3. Histograma de activaciones	54
5.4. Superficie de valores	54
7.1. Modelo al agregar variables	70
7.2. Ejemplo visualización por filtro	71
7.3. Ejemplo visualización por filtro	72
7.4. Reporte de clasificación experto vs irrelevantes	73
7.5. <i>Accuracy</i> en conjunto de validación dado cantidad de características	75
7.6. Resumen cambios característica 34.	76
7.7. Resumen cambios característica 19	77
7.8. Resumen cambios característica 30	78
7.9. Resumen cambios característica 7	78
7.10. Características en diversos criterios	79
7.11. Ejemplo de filtro irrelevante	80
8.1. Diagrama arquitectura	90

Introducción

En los últimos años, los modelos de *machine learning* basados en redes neuronales convolucionales se han posicionado en el estado del arte de diversas áreas de investigación. Algunos ejemplos son: la dramática disminución del error de clasificación en problemas con miles de clases [34], la posibilidad de no solo clasificar sino de detectar objetos en imágenes/vídeos en tiempo real [31], la clasificación de cuerpos de texto [10] y el modelo de aprendizaje reforzado que derrotó al campeón mundial de Go en el 2016 [38]. Estos avances evidencian que el estudio y aplicación de estos modelos tanto en la academia como en la industria se ha convertido en un tema de gran importancia.

A pesar de los resultados demostrados, una crítica que se repite sobre estas técnicas es que son difíciles de interpretar o constituyen cajas negras. En consecuencia, se ha comenzado a investigar la *interpretabilidad* de estos y otros modelos de aprendizaje con datos. Interpretabilidad es un término amplio que abarca diversos objetivos [28], por ejemplo, se puede buscar justificar la decisión de un modelo (característica muchas veces requerida por la legislación), se puede hablar de identificación de errores, de la necesidad de tener confianza en el modelo o también para tratar de extraer la información que aprende el modelo de forma de expandir el conocimiento del problema.

Dentro de esta línea de investigación han aparecido diversos intentos por generar visualizaciones que expliquen la predicción de una red convolucional [30], estas visualizaciones buscan acercar los conceptos que aprende la red convolucional a conceptos conocidos como lo son las imágenes.

Dado tal contexto, el siguiente trabajo propone un método para mejorar la predicción de clasificadores ya entrenados utilizando visualizaciones y un usuario experto. En términos generales la mejora del clasificador es generada por un reajuste de parámetros. La particularidad de este reajuste es que se alimenta con datos que el usuario experto determina, es decir detecta posibles errores al observar visualizaciones. Esta forma de comunicación usuario-modelo presenta dos ventajas. Primero desde la visualización se extrae información del modelo lo cual puede ser útil para aprender del fenómeno que modela (comunicación modelo - usuario) , por otra parte, el usuario puede enseñar al modelo a no cometer errores con lo que ya conoce del dominio. Es decir, el ajuste de parámetros corresponde a la comunicación del experto con el modelo.

Una de las razones que motiva este método es que tanto el experto como el clasificador pueden indicar áreas relevantes. Por una parte, el modelo puede generar una visualización o mapa de calor sobre áreas importantes de la entrada, por el otro el experto a su vez puede

justificar su elección con ciertas áreas. Se razona que si, tanto el experto como el clasificador identifican áreas relevantes para una cierta predicción podría ser útil compararlas.

El problema de fondo a combatir es el sobreajuste de aprender ciertos patrones útiles en entrenamiento, pero que en un caso general sean perniciosos. Por ejemplo, todas las fotos de pesas salen con brazos ¿Son los brazos partes de las pesas? No realmente, si aparece ahora un brazo ¿debería clasificarse como pesa? No realmente. Sin embargo, los modelos de datos no tienen esta información ya que se basan en las correlaciones observadas. Para abordar este problema el presente trabajo se centra solamente en áreas no relevantes, ya que se asumen dos supuestos. El primer supuesto es que los datos presentan solamente áreas relevantes y un fondo irrelevante, diversos *dataset* coinciden en este supuesto [34] [24] [21]. A modo de ejemplo *IMAGENET*, donde las áreas relevantes son partes de animales y el resto corresponde al ambiente donde vive el animal. El segundo supuesto consiste en que es más fácil identificar un área irrelevante que una relevante. A modo de ejemplo, si se observan imágenes de radiografías es posible no tener total certeza del área relevante para predecir una enfermedad, pero es razonable que el código de serie de la radiografías sea irrelevante (un caso real de un algoritmo de predicción, donde se aprecia que el clasificador se ve influido en base al número de serie de imágenes radiológicas se puede observar en el trabajo de Zech et al. [54]). Se espera que al filtrarse las áreas consideradas importantes o al eliminarse las áreas irrelevantes el clasificador aprenda características robustas que mejoren su desempeño.

Este trabajo se centra en los clasificadores por redes convolucionales. Se utilizan estos modelos debido a que su diseño está pensado para trabajar con imágenes [25] y debido a que existen diversas visualizaciones que son necesarias para el proceso de reajuste [30]. Dentro de este trabajo se utilizará reiteradamente el término *característica*, especialmente cuando se refiere a patrones asociados a las redes convolucionales, por lo tanto, es necesario dar una definición. Según el conocimiento del autor no existe una definición formal para este término, el cual se utiliza vagamente para referirse al conjunto de patrones que activa cierta sección del modelo. Por ejemplo, se menciona que una unidad de un modelo contiene la característica de caras si al presentar caras en la entrada esa cierta unidad se activa consistentemente. Dado lo anterior se entiende por característica irrelevante si una unidad se activa consistentemente frente a patrones que el experto denomina irrelevantes. En términos de este trabajo se asocian las características a los filtros, los cuales corresponde a las unidades de las redes convolucionales (mayores detalles en sección 1.1.7).

Una vez definidos los preliminares, la implementación del método requiere de diversos componentes. Se requiere de un clasificador ya entrenado para el problema seleccionado, se requiere un algoritmo para generar visualizaciones, se requiere de un usuario (o experto) que tenga conocimientos del dominio de las entradas, se requiere un mecanismo para seleccionar áreas en las imágenes y por último se requiere un método para reajustar los parámetros dado las selecciones del experto. El clasificador es una variación de las redes neuronales convolucionales, la arquitectura de este clasificador se diseñó con diversas capas utilizadas en el estado del arte (mayores detalles en sección 1.1.6). La visualización utilizada corresponde a generar mapas de calor, es decir, para cada imagen de entrada se genera una imagen del mismo tamaño con valores de intensidad que representan la importancia de cada sección. Esta visualización es necesaria para encontrar áreas que el modelo entrenado considera relevantes (mayores detalles en sección 1.1.8). Para la selección de áreas irrelevantes, el experto utiliza

una interfaz, en esta mediante un cursor indica que píxeles considera como irrelevantes. La selección se guarda como una matriz binaria del tamaño de la imagen, estas matrices binarias en el documento se conocen como máscaras binarias o máscaras. Por último para ajustar los pesos del clasificador dado esta nueva información suministrada (imagen, máscara binaria) se plantean dos alternativas. Una primera alternativa se basa en reemplazar las áreas irrelevantes generando un nuevo *dataset* (se explica este método en detalle en el Capítulo 4). Una segunda alternativa se basa en mantener la imagen sin cambio, pero introducir elementos que penalicen activaciones en la función que optimiza el reajuste (se explica este método en detalle en el Capítulo 5).

Con estos antecedentes, la hipótesis consiste en que la selección dada por el experto, la visualización utilizada y el reajuste conforman un método efectivo para eliminar características irrelevantes, lo cual a su vez producirá una mejora en la generalización del clasificador asociado.

Para analizar la validez de la hipótesis se evalúan métricas de clasificación tales como *accuracy* (exactitud), *precision* (precisión) y *recall* (recuperación o cobertura). Si el método resulta exitoso se espera que el clasificador aprenda características relevantes que aumenten estas métricas en los datos de prueba. A su vez también se espera cambios en las áreas que el clasificador considera relevante, es decir, se espera que después de aplicar el método las visualizaciones irrelevantes desaparezcan, por lo tanto, también se grafican visualizaciones antes y después del método propuesto. También es necesario comparar este desempeño con métodos similares, se implementan experimentos alternativos para comparar si el aumento en métricas es significativo (mayores detalles en sección 3.3). Estas pruebas se realizan en 4 conjuntos de datos distintos para conseguir evaluar robustamente la utilidad del método.

De conseguirse mejoras considerables en métricas de clasificación la utilidad de este método es alta, ya que en el caso de algoritmos de *machine learning* es imposible corregir un errores de clasificación en específico, usualmente la forma de solucionar tal problema es obtener más datos y reentrenar sin ningún enfoque en el problema puntual. Con métodos como el propuesto sería posible observar las áreas de la imagen que generan el problema y guiar al modelo hasta una mejora.

Del trabajo realizado se extraen las siguientes contribuciones.

1. Se logra identificar la existencia de características redundantes o innecesarias en el contexto estudiado, se observa que con un 33 % del total de las características se puede mantener la misma *accuracy* que el modelo completo.
2. Del estudio de las activaciones de los filtros se logra también corroborar la hipótesis de que la inspección visual es útil para detectar tales características. Sin embargo, también se detectó que las visualizaciones no son del todo adecuadas para el objetivo planteado, debido a que es difícil encontrar el nexo entre activaciones puntuales y el comportamiento en general de los filtros.
3. Por último se genera un método capaz de modificar visualizaciones de forma arbitraria manteniendo o mejorando las clasificaciones de las imágenes seleccionadas, pero sin aumento de *accuracy* en el conjunto de prueba.

Capítulo 1

Marco teórico

La función de este capítulo es construir una base teórica para comprender el trabajo realizado. Primero en la sección definiciones se plantean los temas previos necesarios para explicar la propuesta. En la Sección Problema se explica en detalle el problema a resolver.

1.1. Definiciones

1.1.1. Convolución

El punto de partida es explicar qué tipo de clasificador se utilizará. Como en este trabajo se utiliza las redes neuronales convolucionales, es necesario primero explicar la convolución. La convolución es una operación entre dos señales que representa cierta respuesta entre ambas [41]. El caso relevante para este trabajo es la convolución en dominios discretos, la cual se define como:

$$(g * f)[n] = \sum_{m=a}^b g[m]f[n-m] \quad (1.1)$$

Esta operación tiene gran importancia en el análisis de imágenes, ya que permite analizar la respuesta de un cierto patrón en una imagen. Al observar la ecuación 1.1 se puede observar que la operación es similar a aplicar cierta ponderación de los pesos g sobre una entrada f , la gran diferencia con operaciones como el producto punto (suma ponderada) es que esta operación se aplica en diversas posiciones n . En cada posición o ventana n se centra los pesos de g respecto a f , posterior a esto se calcula la suma ponderada correspondiente.

En la ecuación 1.1 se muestra el caso unidimensional de la convolución, sin embargo, es posible definir para cualquier cantidad de dimensiones. En el contexto pertinente las convoluciones se definen sobre 2 dimensiones, por lo tanto, f corresponde a matrices de intensidad y g representa una ventana de dos dimensiones. A su vez dependiendo de las dimensiones de entrada varía la dimensión de salida, si se calcula la convolución en 2 dimensiones para todas

las posiciones posibles de n se tendrá una nueva imagen que representa la convolución de f con g .

El concepto importante sobre las convoluciones es que son operaciones que se aplican en una localidad, por lo tanto, tienen en consideración la información espacial contenida en la imagen. La convolución reutiliza los pesos de g en las diversas ventanas, al desplazar esta ventana se está considerando el mismo patrón en distintas localidades.

En términos de la convolución sobre imágenes, estas se definen por los siguientes parámetros:

- **Kernel (pesos del filtro):** Es la lista de pesos que multiplican la imagen en la localidad correspondiente, es decir, g en la ecuación 1.1. El patrón que generará alta activación en la convolución depende directamente de la magnitud y distribución de los pesos. En este trabajo como se mencionará en las secciones siguientes estos pesos se obtienen de forma automática mediante una optimización.
- **Padding (relleno):** En ciertos casos es necesario agregar filas/columnas de relleno en los bordes de la imagen. Estas son necesarias para suavizar los efectos de aplicar el *kernel* en los bordes y para controlar el tamaño de la imagen resultante. Esta última característica es de gran importancia en las redes convolucionales. Por ejemplo, suele utilizarse la terminología *padding SAME* para referirse al caso donde se agrega tantas filas/columnas que se genera una imagen del mismo tamaño que la original.
- **Stride (salto):** Como se mencionaba en la ecuación 1.1 la convolución va desplazándose según n . Usualmente este salto es de 1 celda, pero es posible ajustar a cualquier tipo de salto. Su función es tratar de controlar el tamaño de la salida de la convolución, pero esta vez disminuyendo las dimensiones.

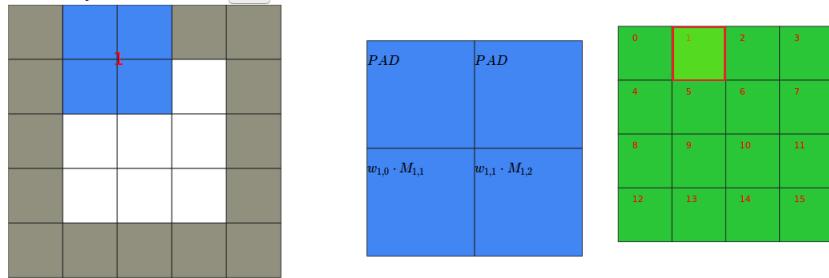


Figura 1.1: Diagrama convolución 2D de una imagen 3x3 con *kernel* 2x2, *padding* 1 *stride* 1. Se observa el cálculo de la segunda ventana (índice 1). Izquierda: Se observa la entrada con la imagen en blanco *padding* en gris. Centro: Ampliación de ventana observada por el *kernel* en la segunda posición o índice 1. Derecha: Se observa salida de la convolución, en rojo salida que representa la segunda posición o índice 1.

Dados estos parámetros el funcionamiento de la convolución se observa en la 1.1. Notar como la entrada original es 3x3 y la salida tiene mayores dimensiones gracias al *padding* utilizado.

En la literatura hay extenso uso de la convolución para analizar características en imágenes. Por ejemplo, el filtro *SOBEL* [42] genera alta activación en presencia de bordes o cambios

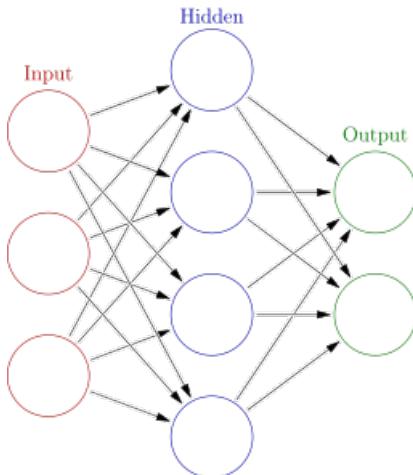


Figura 1.2: Diagrama de red neuronal. Autor Glosser.ca https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

bruscos de intensidad. Las redes convolucionales son en cierta forma un modelo que busca automáticamente los pesos adecuados para la operación convolución.

1.1.2. Redes neuronales

Los modelos utilizados para aprender en este trabajo se basan en redes neuronales convolucionales, por esto se requiere definir primero las redes neuronales que preceden a estos modelos.

Las redes neuronales son modelos paramétricos que aproximan funciones [32]. Una red neuronal consta de capas que funcionan de la siguiente forma:

1. Se toma el vector de entrada $(A, 1)$, el cual se multiplica por una matriz que representa los pesos de la capa correspondiente.
2. El vector resultante tendrá dimensiones $(D, 1)$ donde D depende de las dimensiones de la matriz de la capa correspondiente. Se suele hablar en tal caso que la capa posee D unidades.
3. A cada término del vector resultante se le aplica una función de activación no lineal. Este paso es necesario, ya que de usarse una activación lineal la red deja de ser un aproximador universal [11]. Usualmente se utilizan funciones como la función *sigmoidal*, *tangente hiperbólica*, *ReLU* o *softmax*.
4. El resultado se utiliza como entrada de la siguiente capa.

Usualmente se encadenan varias capas hasta una última capa, la cual debe tener la dimensión de la función que se quiere aproximar.

Es un resultado conocido que una red neuronal con 1 capa oculta y salida (como en la figura 1.2) de suficientes unidades puede aproximar cualquier función a una precisión arbitraria [11].

A pesar de la garantía del teorema de aproximación universal en la práctica los modelos de redes neuronales no son prácticos en entradas tales como imágenes. Esto ocurre debido a que las relaciones importantes en imágenes son localizadas en partes de la entrada. Por ejemplo, si se requiere detectar el patrón de un ojo, este patrón abarca solamente una parte de la imagen. Las redes neuronales buscan patrones en la totalidad de la entrada, por lo tanto, para detectar tal patrón deben aprender el mismo patrón en todas las posiciones posibles. Una forma más eficiente de representar tal patrón podría ser utilizar patrones más pequeños que se desplazaran por la entrada, de esta forma se representa con un solo patrón que es invariante a la traslación.

1.1.3. Redes convolucionales

El problema anterior se soluciona con las redes neuronales convolucionales, las cuales están preparadas para enfocarse en localidades espaciales de la entrada al realizar convoluciones en vez de productos puntos.

Las capas de las redes neuronales generan vectores como resultados, las capas convolucionales usan y generan imágenes (en realidad tensores, pero es intuitivo pensar estos tensores como imágenes de varios canales). Cada capa tiene una cantidad de canales o filtros que son el símil de las unidades en las capas neuronales.

Al igual que en las redes neuronales las redes convolucionales se encadenan y se les aplican no linealidades.

La característica de las redes convolucionales que es fundamental para este trabajo es que sus diversos filtros van realizando activaciones localizadas, es decir, cada activación tiene asociada una cierta parte de la entrada. Esto hace posible seguir la activación en sentido inverso de forma de encontrar las secciones de la entrada que influyen en la activación de un filtro.

1.1.4. Algunas capas de las redes convolucionales

Además de las capas de convolución las redes neuronales convolucionales utilizan un sin número de otras capas. Las más importantes para el contexto son las siguientes:

- ***Max pooling:*** Esta capa realiza un operación similar a la convolución, al evaluar por ventanas la entrada (comparar ecuación 1.2 con ecuación 1.1). La diferencia es que en este caso se utiliza el máximo de la ventana en vez del producto punto. Usualmente se utiliza con ventanas sin intersección de forma de reducir las dimensiones espaciales de un filtro, esta reducción de las dimensiones del filtro permite una menor utilización de recursos. Además de este efecto meramente práctico, este proceso de agregación de información se ha argumentado que facilita la detección de patrones complejos [7].

$$\text{maxpool}(f)[n] = \max \{f[x] \mid x \in \text{Ventana}(n)\} \quad (1.2)$$

- **Global average pooling (GAP):** Capa que calcula un promedio por filtro, lo cual permite convertir las activaciones convolucionales en vectores. Si se observa la fórmula 1.3 al tener i, f (punto índice i , filtro índice f) fijos la doble sumatoria corresponde a iterar toda la matriz que representa el filtro f y luego al dividir por la cantidad de celdas (es decir calcular el promedio). Esta capa es necesaria para poder conectar redes convolucionales con redes neuronales tradicionales, esta unión se utiliza generalmente en la capa final para predecir un escalar o un vector.

$$GAP(i, f) = \frac{1}{W \cdot H} \sum_x^H \sum_y^W Act(i, x, y, f) \quad (1.3)$$

- **Dropout:** Técnica donde se apagan ciertas unidades en la red de forma aleatoria. La capa *Dropout* se implementa multiplicando elemento a elemento una máscara binaria con la entrada. De esta forma todas las unidades que se multipliquen por 1 quedan igual y las que se multiplican por 0 quedan filtradas. En la fórmula 1.4 se representa como multiplicar la capa anterior *Act* por M . Una forma de generar M es generar una matriz del mismo tamaño de *Act* donde cada entrada es una variable aleatoria que toma valores 0 o 1 con probabilidad $p_{dropout}$ (es decir una distribución de Bernoulli). El uso de esta aleatoriedad es reducir la interdependencia de las unidades, esto tiene un efecto positivo en el entrenamiento tal como se aprecia en el trabajo de Srivastava et al. [43].

$$\begin{aligned} M(i, j) &\sim Be(p_{dropout}) \\ Dropout(i, x, y, f) &= Act \odot M \end{aligned} \quad (1.4)$$

- **ReLU:** No linealidad utilizada como función de activación. Como se aprecia en la fórmula 1.5 esta función corresponde a filtrar la sección negativa de una función y en el resto de los casos no cambiar los valores. Su utilidad radica en que permite un entrenamiento más rápido que otras funciones de activación [5].

$$ReLU(x) = \max(0, x) \quad (1.5)$$

- **Softmax:** Capa que escala un vector tal que cada elemento sea un valor entre 0-1 y que la sumatoria de todos los términos entregue 1. Se utiliza usualmente en las salidas de los clasificadores multiclas para representar la salida de una probabilidad conjunta. En la ecuación 1.6 el término exponencial garantiza siempre valores positivos, mientras que la división por el total de los valores por fila es lo que asegura que el valor sume 1 por fila.

$$h_{k,l} = \frac{e^{u_{k,l}}}{\sum_t^C e^{u_{k,t}}} \quad (1.6)$$

1.1.5. Clasificación

Con la explicación de los diversos tipos de redes neuronales, queda por explicar cómo se ajustan estos modelos para resolver el problema de clasificación. La definición de clasificación es asignar una cierta categoría a cada punto, generalmente a cada punto le corresponde una

única categoría. El problema de clasificación es generar un modelo que asigne la categoría correcta a cada punto. Este trabajo se centra en mejoras a clasificadores, debido a que si bien la mayoría de los conceptos que se mencionan podrían trasladarse al problema de regresión (predicción de escalar) la literatura de visualización está en su mayoría enfocada a clasificación.

Para clasificar con redes neuronales es necesario definir una forma de predecir un punto y una función de pérdida. La forma de clasificar el punto es mediante predecir la probabilidad conjunta de cada clase definida. Para esto se coloca en la última capa tantas unidades como clases se requiera predecir. Estas unidades se les aplica la capa *softmax* 1.6, esta capa se asegura que los valores estén entre 0-1 y que todos sumen 1 tal como una probabilidad conjunta. Se toma el máximo como la predicción del clasificador.

$$L = CE = -\frac{1}{N} \sum_i^N \sum_j^C P_{ij} \cdot \log(h_{ij}) = -\frac{1}{N} \sum_i^N \sum_j^C Y_{ij} \cdot \log(h_{ij}) = -\frac{1}{N} \sum_i^N \log(h_{i*}) \quad (1.7)$$

Para entrenar una red convolucional se requiere una función que indique que tan lejos se está del objetivo [13]. Para el caso del problema de clasificación se utiliza la llamada entropía cruzada (ecuación 1.7). Esta ecuación es simplemente tomar el promedio del logaritmo negativo para cada punto i en el conjunto de datos. Su utilización permite tener una estimación de la cercanía entre dos distribuciones, que son la distribución de los datos observados y la distribución generada por las probabilidades del modelo.

También es importante mencionar la forma de codificar las categorías. En el contexto de clasificación binaria se utiliza las categorías 0 o 1, lo que significa que estos son los únicos valores posibles para la matriz de categorías Y (matriz con todas las categorías de los datos de entrenamiento). Esto genera que la matriz Y sea una matriz de 1 columnas y N filas (siendo N la cantidad total de puntos). Para un problema de múltiples clases el proceso es análogo, pero Y es una matriz de N filas por C columnas (C corresponde a la cantidad de clases). A modo de ejemplo, si el punto i – *esimo* es clase c_i la fila i tendrá un 1 solamente en la columna c_i y tendrá 0s en todas las demás columnas. Se puede interpretar tal notación como una representación binaria **claseX** o **noclaseX**. Este tipo de notación se conoce como representación *one hot*.

$$\prod_i^N \hat{P}_{i*} \rightarrow \log \left(\prod_i^N \hat{P}_{i*} \right) = \sum_i^N \log(\hat{P}_{i*}) = \sum_i^N \sum_j^C \log(\hat{P}_{ij}) \sim -1 \cdot CE$$

Figura 1.3: Relación entre entropía cruzada y log-verosimilitud al tener vectores *one hot*. P_{i*} indica la probabilidad del punto i dado su clase correspondiente. A cada punto se le asocia una única clase, lo que permite a la sumatoria sobre C desaparecer. De la misma forma se puede obviar el término Y_{ij} ya que éste es 1 solamente cuando coincide con P_{i*}

Hay dos formas de interpretar el uso de la entropía cruzada. Desde un punto de vista teórico para el caso de clasificación la minimización de la entropía cruzada consigue resulta-

dos equivalentes a la maximización de la log-verosimilitud (demonstración en figura 1.3). La verosimilitud mide dado una cierta distribución cual es la probabilidad de que los datos observados fueran generados por tal distribución. Dado todo lo anterior el optimizar la entropía cruzada entrega los parámetros de mayor verosimilitud según los datos observados. Por otro lado, desde la intuición la entropía cruzada penaliza según qué tan lejos h_{ij} (predicción de probabilidad del punto) este de 1, ya que en el rango de $0 \leq h_{ij} \leq 1$ el logaritmo negativo es monótono decreciente con mínimo para $h_{ij} = 1$.

Después de definir la función de pérdida la forma de obtener los valores de todos los parámetros es mediante una optimización de los pesos. Si bien es posible utilizar otros métodos de optimización las redes neuronales usualmente utilizan *backpropagation* [33]. Este algoritmo permite estimar el gradiente de la función de pérdida en un subconjunto del total de datos, lo que ahorra tiempo de procesamiento y permite utilizar conjuntos de datos enormes.

1.1.6. Arquitectura VGG

Si bien los parámetros de una red convolucional pueden determinarse mediante optimización el que tipo de capas y en qué orden es siempre debatible. Por una parte, es posible validar el diseño mediante datos experimentales, por otra parte, el diseño es también guiado por una serie de heurísticas basadas en arquitecturas exitosas de la literatura. Este trabajo no es la excepción, las arquitecturas que se presentarán en las secciones de desarrollo están basadas en las arquitecturas VGG.

Las arquitecturas VGG aparecen con el trabajo de Simonyan and Zisserman [39] donde se buscaba generar mejoras en el problema de clasificación para el conjunto de datos IMAGE-NET. En aquel trabajo se propone crear redes con bloques de capas convolucionales. Estos bloques tienen la particularidad de ser siempre filtros de *kernel* 3x3 y también duplicar la cantidad de filtros después de la capa *max pooling*. Al estos utilizar solamente filtros de 3x3 se logra ahorrar parámetros. Por ejemplo, al tener 3 capas convolucionales con *kernels* de 3x3 se tiene el mismo campo receptivo que una capa de 7x7, pero con una menor cantidad de parámetros ($3 \cdot (3^2 \cdot 3^2 \cdot C)$) vs ($7^2 \cdot 7^2 \cdot C$). Al tener menos parámetros es posible entrenar con los mismos recursos modelos más profundos. Las capas de *max pooling* tienen el efecto de reducir el tamaño de las activaciones, con activaciones de menor tamaño es posible aumentar la cantidad de filtros con la misma memoria que hubiese utilizado las activaciones sin el proceso de *pooling*. Los autores en Simonyan and Zisserman [39] argumentan que una mayor cantidad de filtros en capas más profundas permiten capturar patrones más complejos lo que mejora el desempeño.

1.1.7. Características y filtros

Una vez que se ha definido el tipo de clasificadores a utilizar, es necesario definir un término de amplio uso en la literatura **las características**. En este trabajo se define característica o *feature* como un término que agrupa los tipos de imágenes que activan una cierta unidad, es decir, asociar a las unidades cierto tipo de patrones. Como se mencionó anteriormente

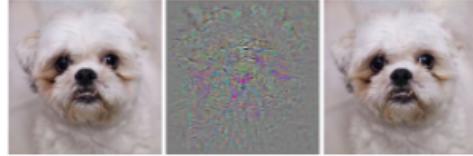


Figura 1.4: A la izquierda imagen clasificada como perro, al centro ruido adversario, a la derecha imagen con ruido clasificada como camello. Imagen con Licencia CC extraída de Szegedy et al. [44]

no hay definición formal del término característica, aún peor no hay una forma objetiva de conseguir o extraer las características. Esto ocurre debido a que especialmente en las redes neuronales no es posible tener una representación intuitiva de una unidad, usualmente se visualiza una serie de representaciones de la unidad o se observa muchas de sus activaciones [52]. La descripción de una característica termina siendo relativa al observador (el experto en términos de este trabajo) y la forma en que visualiza la característica. Como la definición de característica depende de un tercero, existe un problema, ¿Podría ser que el modelo de clasificación generara características o patrones irreconocibles para el experto? La respuesta a esta pregunta no es fácil, debido a que los resultados experimentales indican que existe evidencia tanto a favor como en contra. Por una parte, al analizar y agrupar activaciones en clasificadores de *IMAGENET* se evidencia que en su mayoría presentan patrones que son fácilmente reconocibles. Sus capas iniciales se activan para características simples como colores, bordes en diversas orientaciones o círculos, mientras que en capas más profundas se activan para conceptos de alto nivel como gatos, muebles, caras [55]. Por otro lado, se ha observado en estos mismos clasificadores que es posible introducir un ruido o pequeñas perturbaciones y cambiar completamente la clasificación de una imagen. Por ejemplo, es posible convertir la predicción de un perro en un camello sin cambios apreciables al observador (observar figura 1.4) [44] [37]. Con esto se tiene que las *características* pueden relacionarse con patrones que un experto reconozca, pero hay que recordar que son solo aproximaciones.

Basado en que existe interpretación para las características que componen al clasificador, en este trabajo se propone refinarlas utilizando el conocimiento del experto.

Para simplificar el problema, se asocian las características solamente a los filtros de la última capa. Esto tiene relación con la visualización elegida, como se explicará en la sección siguiente. También en general es en las últimas capas donde ocurre la clasificación, por lo tanto, son las más importantes para determinar las características relevantes a la predicción.

Este trabajo requiere identificar características irrelevantes para su ajuste. Por lo tanto, se requiere definir qué es una característica relevante o irrelevante. Se define característica relevante como características que el experto considera que captura información no redundante para clasificar la imagen. Como se ha mencionado para determinar una característica es necesario conocer todos los patrones que activan una cierta unidad, lo que implica que tratar de representar fidedignamente una característica es un trabajo difícil. En vez de buscar características se utiliza las visualizaciones como un *proxy* fácilmente identifiable. De estas visualizaciones se seleccionan áreas irrelevantes como representantes de las características irrelevantes.

Estas áreas irrelevantes apuntan a patrones en la imagen que el experto considera irrelevante, se espera utilizar estas áreas irrelevantes como ejemplos de lo que no debe activarse. De esta forma se entregan indicios al clasificador de cómo mejorar sus predicciones.

En resumen, las características son las unidades que mantienen los patrones de importancia para la predicción, si bien es posible visualizarlas su interpretación sigue siendo complicada. Debido a esto solamente se seleccionan áreas irrelevantes para dar pistas al modelo de como modificar sus características.

1.1.8. Modelo de visualización CAM o *Class Activation Mapping*

Para este trabajo es necesario tener una representación de lo que el modelo observa para realizar una predicción. Específicamente se requiere generar visualizaciones que indiquen áreas de importancia para diversas imágenes, este tipo de visualización se conoce como mapas de valor.

La visualización que se utiliza en este trabajo es el CAM (*class activation mapping*) [56], la utilidad del algoritmo CAM es generar mapas de calor que entregan áreas de mayor importancia para cierta clase. A existir C clases existen C imágenes del mismo tamaño de la entrada que indican áreas relevantes. Lamentablemente este tipo de visualización funciona solamente para un tipo específico de arquitectura, la cual tiene capas convolucionales seguidas por GAP (*global average pooling*) y un clasificador de una capa.

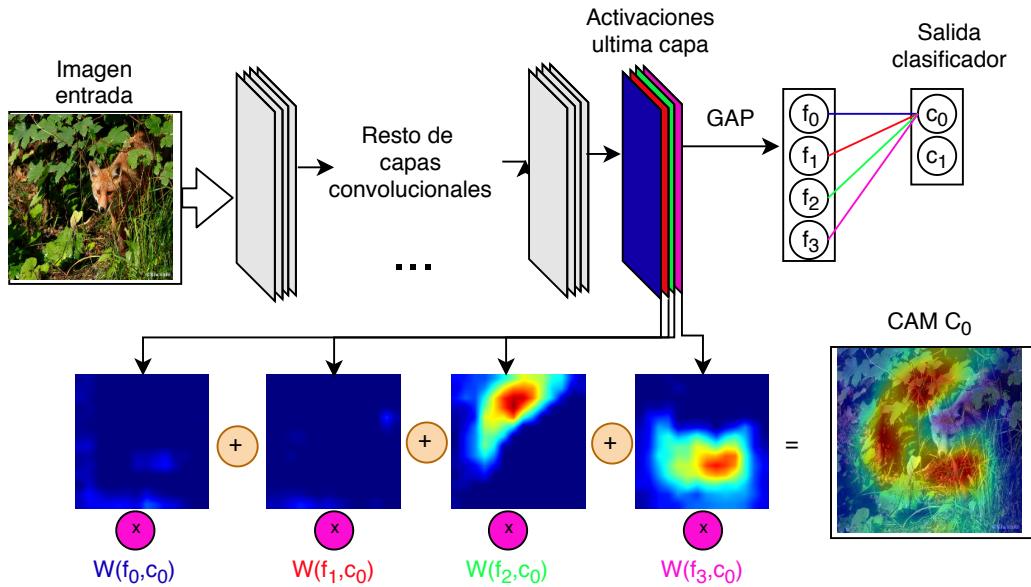


Figura 1.5: Diagrama del funcionamiento visualización CAM

El proceso para generar esta visualización es simple y se aprecia en la figura 1.5. A modo de ejemplo se tiene el problema de clasificación donde existen clases C_0 y C_1 , se tiene además un arquitectura convolucional que termina en 4 características (f_0, f_1, f_2, f_3). Para obtener la visualización primero es necesario calcular las activaciones de la red dada la imagen de

entrada, se elige la clase para generar el CAM, se seleccionan todos los filtros de la capa convolucional final, se ponderan cada filtro por su peso en el clasificador de una capa y se suman generando una nueva imagen. Después de generar esta imagen queda realizar un ajuste de las dimensiones, esto es necesario debido a que las capas de *max pooling* reducen la dimensión progresivamente, lo que implica que las capas finales tienen dimensiones muy inferiores a la imagen original. En el trabajo de Zhou et al. [56] se realiza un redimensionamiento bilineal para ajustar esta visualización a la imagen original, lo cual también se aplica en el presente trabajo.

Como se observa este proceso intenta reproducir la clasificación, pero sin reducir los filtros a escalares, de esta forma se consigue una representación visual de la importancia de cada filtro en el proceso.

Una característica de este tipo de arquitectura es su simpleza de interpretación, es posible interpretar cada filtro de la última capa como un extractor de características, luego para obtener la clasificación se calcula promedio para cada filtro lo que representa el nivel de presencia de la característica en la imagen. Además, la capa de salida es una combinación lineal, lo que permite analizar la contribución de cada filtro por separado en el peso asociado. Se puede pensar en estos pesos como una métrica de la importancia global de cada filtro respecto a una cierta clase.

Otro detalle importante de mencionar es que al analizar la visualización CAM implícitamente se centra el análisis en las características de las últimas capas. Esto simplifica bastante el análisis, ya que no es necesario considerar efectos de filtros combinados (es decir como un filtro en un capa intermedia afecta la salida) o de un clasificador más complejo que el de una capa (es decir como afectaría que el clasificador final fuera una red neuronal multicapa).

1.1.9. *Data augmentation* o Aumento de datos

Una de las propuestas sobre como reajustar los parámetros corresponde a modificar los datos en sus áreas irrelevantes. Este método tiene cierta similitud con una técnica llamada *Data augmentation*.

En general a mayor cantidad de datos se pueden entrenar modelos más complejos, pero a su vez conseguir grandes cantidades de datos categorizados puede ser costoso. En respuesta a esta problemática se han desarrollado métodos como *Data augmentation*, los cuales buscan generar más datos a partir de ciertas transformaciones en los datos ya presentes. Estas transformaciones deben ser tales que no cambien la categoría del punto. Se utilizaron con gran éxito en problemas como *IMAGENET* [22] donde mediante traslaciones, rotaciones, cortes y cambios de colores se simulan cambios que pueden sufrir los datos en el mundo real. Este proceso según Krizhevsky et al. [22] posibilitó usar arquitecturas más complejas.

Una desventaja de este método es que las transformaciones no son aplicables a cualquier *dataset*, ya que dependiendo del tipo de transformaciones pueden introducir cambios en la categoría. Por ejemplo, si se utilizan imágenes de espectrogramas (imágenes de las frecuencias de una señal en el tiempo, donde cada celda significa un par frecuencia/tiempo distinto)

una traslación indicaría un dato completamente distinto con posiblemente una categoría completamente distinta. La propuesta trata de generar datos guiados por un experto de forma de adaptarse al caso correspondiente.

Data augmentation es similar a la propuesta en el sentido que busca generar imágenes que ayuden al modelo a aprender una cierta característica que se conoce sobre el *dataset*. En el caso tradicional *data augmentation* comunica al modelo sobre la invarianza a traslación, rotación o color (información que maneja el experto sobre dominio del problema), lo cual mejora su clasificación en los datos de prueba.

1.1.10. Modelos generativos

Como se ha mencionado se requiere una forma de modificar las imágenes en áreas irrelevantes. Para lograr tal modificación o edición de datos se utiliza los modelos generativos de imágenes.

Los métodos generativos son aquellos que se utilizan para tratar de parametrizar una cierta distribución, generando datos similares a los que se utiliza para entrenar el modelo. Entre los modelos que se aplican al dominio de las imágenes están las GANs o *Generative Adversarial Networks* [16]. Estos modelos requieren entrenar dos redes convolucionales una como generador y otra como discriminador. El generador trata de dibujar imágenes que el discriminador confunda por imágenes que efectivamente estaban en los datos de entrenamiento. Por su parte el discriminador entrena de forma de diferenciar entre imágenes generadas y reales. Esta competencia entre ambas redes termina en un modelo capaz de crear imágenes bastante realistas. El objetivo de este proceso es que las redes entrenadas aprendan diversos patrones del *dataset* lo cual es útil para una serie de problemas.

La propuesta utiliza los modelos generativos para realizar cambios a las imágenes que poseen áreas irrelevantes. Esto es algo distinto que simplemente generar imágenes, pero afortunadamente existe investigación en este tema que se conoce como *image inpainting*. Se utiliza el trabajo de Yu et al. [53] donde se desarrolla una arquitectura convolucional generativa que aprende a reemplazar trozos de imágenes. Las ventajas de esta arquitectura es que puede realizar reemplazos dado una máscara arbitraria, además de ya estar entrenada para datos similares a *IMAGENET*.

Su funcionamiento se basa en reconstruir de forma iterativa la imagen:

- Se calculan activaciones dado la imagen con la máscara correspondiente.
- Se rellena las secciones faltantes con patrones suavizados que la red considera plausibles.
- Una segunda red refina estos patrones suavizados. Este refinamiento considera tanto patrones locales como globales.

El utilizar redes convolucionales en la refinación permiten llenar patrones complejos que requieren de conocimiento de los datos, por ejemplo, es capaz de reconstruir caras completamente.

Entre los factores importantes para este proceso está que el reemplazo sea con un patrón que no afecte la clasificación o que genere un nuevo patrón que distraiga al modelo de aprender áreas relevantes.

1.1.11. Métricas de clasificación

Respecto a las métricas con las cuales analizar los clasificadores, todas las métricas de clasificación utilizadas parten de la matriz de confusión. La matriz de confusión aparece de cruzar las predicciones del modelo con las verdaderas categorías y anotar las frecuencias de cada par (predicción / categoría real). En el caso de clasificación binaria (observar tabla 1.1) en la esquina superior izquierda están los puntos positivos que fueron clasificados correctamente, en la esquina superior derecha están los positivos que fueron clasificados como negativos, en la esquina inferior izquierda están los negativos que eran positivos y en la celda restante los correctamente clasificados como negativos. Luego es posible extraer las siguientes métricas:

	Predicción positiva	Predicción negativa
Clase positiva	tp (verdaderos positivos)	fn (falsos negativos)
Clase negativa	fp (falsos positivos)	tn (verdaderos negativos)

Tabla 1.1: Matriz de confusión caso binario.

- *Accuracy* (exactitud): La *accuracy* corresponde a promediar la cantidad de puntos correctamente clasificados tanto negativos como positivos por la cantidad total de puntos, es decir, sumar la diagonal de la matriz de confusión y dividir sobre el total. Se puede extender esta definición a un problema de múltiples clases sumando todos los puntos correctamente categorizados divididos por el total de puntos. Esta es una de las métricas más utilizadas debido a su simplicidad al entregar un porcentaje fácil de entender sobre la efectividad del modelo, a pesar de esto sufre problemas al estar las clases desbalanceadas. Si se tiene 90 % de elementos de una clase un clasificador que entrega siempre esa clase obtiene 90 % de accuracy sin aprender absolutamente nada.

$$Accuracy = \frac{tp + tn}{tp + fp + tn + fn} \quad (1.8)$$

- *Precision* (precisión) : Corresponde a la cantidad de puntos correctamente clasificados como positivos sobre el total de predicciones positivas. Esta métrica indica que tan creíble o preciso es el clasificador, es decir, la probabilidad de que si se observa una predicción positiva efectivamente el punto pertenezca a tal categoría. Esta métrica se puede extender a múltiples clases si se calcula varias veces para cada categoría como un problema binario de **ClaseX vs NoClaseX**. La utilidad de esta métrica es alta cuando se requiere analizar el balance o costo de los falsos positivos. Si el costo de un falso positivo es alto es necesario que el modelo tenga una alta precisión, si trato de predecir cuando una persona tiene una enfermedad peligrosa espero que el modelo sea preciso para no preocupar al paciente sin razón.

$$Precision = \frac{tp}{tp + fp} \quad (1.9)$$

- *Recall* (Recuperación o cobertura): Corresponde a dividir los puntos correctamente clasificados como positivos sobre el total de puntos realmente positivos. Esta métrica indica la probabilidad de que un punto que realmente sea positivo sea detectado como tal por el modelo. Se utiliza el mismo proceso explicado en precisión para extender al caso de múltiples clases. Es de gran utilidad cuando se analiza el costo del falso negativo. Si el costo de pasar por alto un punto positivo es alto se requiere una alta recuperación. Si se utiliza un sistema para detectar explosivos es importante que se recupere todos los valores que existen.

$$Recall = \frac{tp}{tp + fn} \quad (1.10)$$

- F1: Corresponde a la media armónica entre recuperación y precisión. La media armónica tiene la característica de ser sensible a valores bajos, por lo tanto penaliza de mejor forma si uno de los dos componentes baja demasiado. Es útil ya que expresa en un solo escalar el desempeño, pero tomando en consideración los porcentajes dados por precisión y recuperación.

$$F1 = 2 \cdot \frac{precision \cdot recuperacion}{precision + recuperacion} \quad (1.11)$$

Se utiliza múltiples métricas para entender desde diversas escalas el modelo. Por un parte *accuracy* es fácil de evaluar e interpretar al expresar la eficacia del modelo en un solo escalar. Lamentablemente esta métrica puede esconder diversos problemas cuando las clases no se encuentran balanceadas. Por esto se calculan métricas para cada clase por separado, estas métricas son precisión, recuperación, F1. Con estas se espera tener un detalle de que clases son las más difíciles para el modelo y además observar como el modelo va cambiando en detalle. Para el caso del trabajo la métrica más importante es F1, debido a que no hay preferencia entre falsos positivos o falsos negativos. Se utiliza las demás métricas para tener un detalle de los cambios en el modelo.

1.2. Problema

Como en muchos otros trabajos el objetivo consiste en mejorar la predicción en el problema de clasificación, es decir aumentar las predicciones correctas en los datos no utilizados para entrenamiento. Usualmente esta mejora se basa en proponer alguna mejora al algoritmo de entrenamiento. Ya sea una nueva capa para la arquitectura, una nueva forma de aplicar *backpropagation* o editando la función de pérdida. Una dirección menos explorada consiste en formas de mejorar o reajustar el modelo ya entrenado. Esta dirección es interesante debido a que puede ahorrar costos y aun más importante el modelo ya maneja cierta información del problema.

Como se mencionó las redes neuronales son extractores de características automáticas, trabajos recientes han estudiado las formas de estas características en diversas visualizaciones. Esto permite a un experto contrastar las características contenidas en el modelo con las características que conoce como relevantes o irrelevantes. Dado esto se plantea si es posible mejorar la clasificación mediante un refinamiento de características dado por las visualizaciones y el experto.

El método propuesto de ser factible tiene una serie de beneficios. Primero los modelos de aprendizaje en general buscan correlaciones para realizar sus predicciones, estas correlaciones aprendidas en entrenamiento pueden ser perniciosas al aplicarse en casos más generales. De hecho, uno de los problemas recurrentes de estos modelos es el sobreajuste que corresponde a que el modelo aprenda patrones muy específicos para el conjunto de entrenamiento y que por lo tanto falle al predecir en el conjunto de prueba. Este método también puede ayudar al modelo a ignorar los patrones irrelevantes creados por el proceso de extracción de datos. Por ejemplo, en diversos casos suelen repetirse ciertas partes de la imagen ya sea en marcas de agua, objetos estáticos o incluso números de serie del aparato que toma las imágenes. La presencia de estos patrones induce al modelo a aprender características irrelevantes y la eliminación de estos patrones es un proceso arduo y costoso. Por último, este método puede ayudar a filtrar características poco relevantes. En el caso de un conjunto de datos de clasificación de fotos de animales, si los fondos de animales se repiten el modelo puede terminar dando una importancia demasiado elevada al entorno lo que si bien funciona en ciertos contextos no es ideal para el caso general.

En todos estos casos el experto mediante la información que entrega filtra patrones irrelevantes, lo que permite al modelo concentrarse en los patrones robustos que generalizan mejor.

1.3. Resumen

En este capítulo con el objetivo de comprender el contexto del problema, se presentaron incrementalmente los temas de convolución, clasificación, redes neuronales, visualizaciones. De estos temas se desprende que las redes neuronales convolucionales son modelos complejos, pero aun así es posible extraer las áreas que el modelo considera relevante. Dado esto se define el problema a resolver, el cual corresponde a mejorar la generalización de un clasificador contrastando áreas de relevancia entre un experto y el modelo (visualizaciones del modelo).

Capítulo 2

Propuesta

En este capítulo se expone la propuesta que trata de resolver el problema. En la sección Propuesta, se habla del razonamiento detrás del método a desarrollar junto con su descripción detallada y posibles desafíos en su implementación. Las secciones hipótesis, preguntas de investigación tienen la función de guiar el trabajo de investigación a preguntas concretas y demarcar los límites del trabajo. Por último la sección trabajo relacionado menciona las soluciones alternativas que presenta la literatura asociada.

2.1. Descripción de la propuesta

2.1.1. ¿Por qué son difíciles de entender las redes convolucionales?

La gran dificultad de los modelos de redes convolucionales nace de que el proceso de decisión depende de millones de parámetros, los cuales son calculados a partir de los datos observados al entrenar el modelo.

La definición de caja negra implica que no se conoce el funcionamiento interno de un proceso. Por la parte de los parámetros es erróneo mencionar que las redes convolucionales sean cajas negras, en el sentido de que se conoce como se llega a una predicción. Dado los parámetros del modelo es fácil ir paso por paso revisando cómo se activan y se obtiene la predicción final. Se habla de las redes neuronales como cajas negras debido a que se desconoce que representan las transformaciones de capas intermedias. Por ejemplo, si nos detenemos en la capa final tenemos que las activaciones forman puntos en un espacio multidimensional de D dimensiones, donde D son la cantidad de unidades en la última capa. Se puede encontrar mediante los pesos del clasificador qué características aportan de forma positiva, negativa y en qué magnitud, pero ¿Qué significa que la variable 3 se active 0.4? Estas unidades en el espacio D dimensional son una serie de combinaciones no lineales sobre las unidades de la entrada, lo cual es algo no trivial de interpretar.

Por otro lado, todos los modelos dependen de los datos. Los problemas de regresión o

clasificación se pueden interpretar como la búsqueda de cierta superficie en el espacio de la entrada [13]. Para el caso de regresión se trata de conseguir la superficie que asigne a cada punto en el espacio de entrada el valor de salida correcto, para el caso de clasificación se trata de asignar a cada punto una confianza de pertenecer a cada clase. Lamentablemente en el mundo real no es posible trabajar con los infinitos puntos que pueblan esta superficie, solo se tiene una muestra dada por los datos de entrenamiento. El problema aparece debido a las infinitas funciones que coinciden con los valores de entrenamiento, muchas de estas soluciones cargan con los sesgos de los datos observados. Por ejemplo, se observa en la figura 2.1 como el clasificador (línea verde punteada) puede ajustar todos los datos observados (puntos rojos y azules), pero va a fallar en clasificar en el caso general. Esto también ocurre en los datos del mundo real, cuando las condiciones de entrenamiento distan mucho de las condiciones reales, algunos casos simples podrían ser tomar todas las fotos de entrenamiento con la misma iluminación, se utilizaron datos sintéticos, se mantiene el mismo fondo en todas las imágenes, etc.

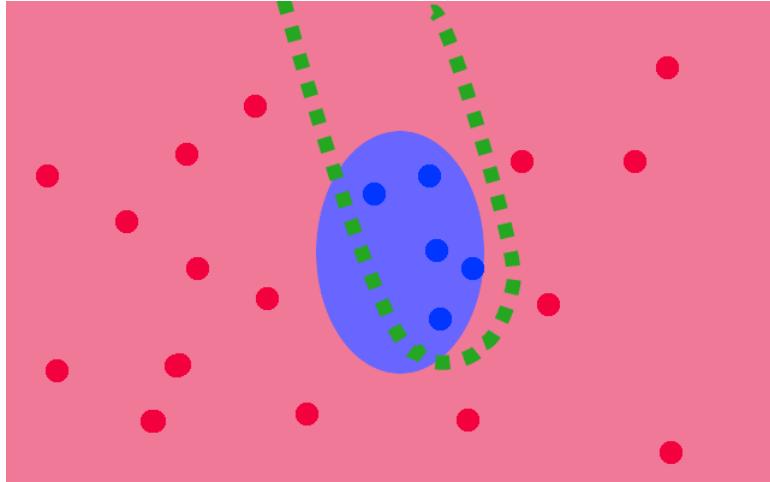


Figura 2.1: Representación de clasificador y datos. Se representa un clasificador binario entre puntos rojos y azules. El clasificador en verde delimita la frontera entre los puntos.

2.1.2. Ayudando al modelo a aproximar la superficie objetivo

Dado estos antecedentes la propuesta trata de comunicar o dar pistas sobre la forma de la superficie objetivo para mejorar la clasificación del modelo. En general cuando se requiere mejorar un clasificador la primera recomendación consiste en buscar más datos [18]. Aun así no todos los datos son igualmente valiosos, por ejemplo si se observa la figura 2.2 de entregar más puntos lejos de la circunferencia no habrá cambios. La propuesta plantea que si se tiene una idea sobre la superficie objetivo (conocimiento de expertos) y un método para entregar tal información al modelo (generar reajuste) se podrían sugerir cambios a la frontera del clasificador mejorando su generalización.

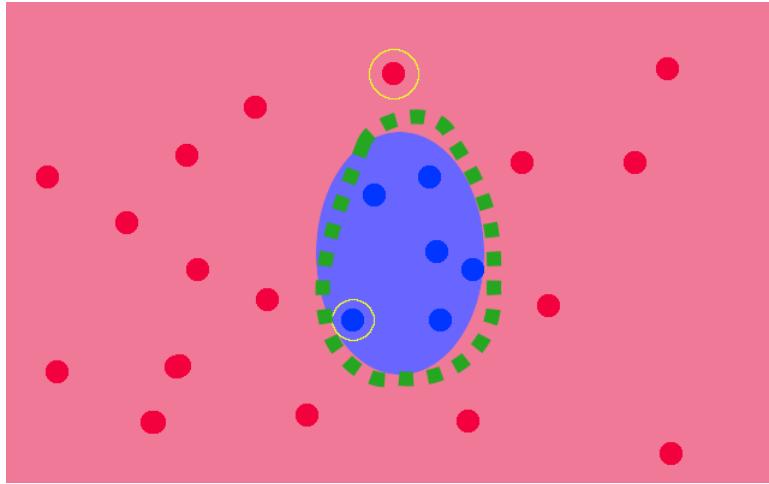


Figura 2.2: Al agregar más puntos en lugares de conflicto el clasificador mejora.

Concretamente la “comunicación” consiste en

1. Ver la representación de la superficie objetivo o algún *proxy* (comunicar modelo – experto).
2. Agregar puntos o editar la función de pérdida de forma de indicar al modelo que está equivocado (comunicar experto - modelo).

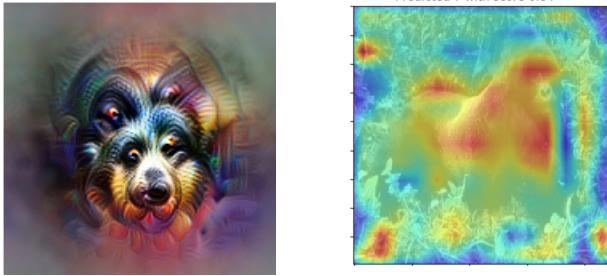
Dado que el caso general sigue siendo muy difícil se trata de llevar el problema al siguiente contexto:

Todo el trabajo se realiza sobre problemas de clasificación de imágenes, donde el experto posea conocimiento del dominio que se ve expresado en distinguir áreas irrelevantes para una clasificación dada. Esto tiene el objetivo de que el experto pueda identificar claramente una respuesta positiva o negativa y que áreas de tales imágenes contribuyen. Si observa que un área es irrelevante, pero el modelo la indica como de alta relevancia el experto puede seleccionarla para que el reajuste cambie tal activación.

2.1.3. Extraer información del modelo: Entender el modelo a partir de visualizaciones

Dentro de las visualizaciones existen dos tipos: las que intentan generar imágenes representativas y las que dado una imagen de entrada resaltan ciertas áreas de importancia. Un ejemplo de cada una se observa en la figura 2.3.

El primer grupo es útil para intentar representar de forma global un filtro o más estructuras dentro de la red convolucional. Por ejemplo en Dosovitskiy and Brox [12] se generan imágenes que maximizan la activación dada una cierta unidad, de esta forma se asocia una sola representación a cada filtro y se puede hablar de una lista de características con las que clasifica el modelo. Si bien estos métodos pueden ayudar a explicar una característica, no es posible utilizar tal imagen en el contexto del problema planteado. Esto ocurre debido a que



(a) Ejemplo de visualización global por filtro

(b) Ejemplo de visualización que indica áreas importantes para una imagen en específico

Figura 2.3: Izquierda: La imagen se genera por un proceso de optimización de la activación de cierto filtro. Al parecer el filtro detecta hocicos de perros. Derecha: El rojo indica alta activación. La visualización se coloca sobre la imagen original para observar a que áreas se refiere.

no indica áreas importantes para alguna imagen en específico, sino que trata de representar el modelo en su totalidad.

Dentro del segundo tipo existen métodos como el trabajo de Zeiler and Fergus [55] que arma una red paralela de forma de propagar hacia la entrada partes importantes, Simonyan et al. [40] que dado una unidad calcula gradientes resaltando la entrada o Zhou et al. [56] que utilizan activaciones de filtros de forma ponderada. Los primeros dos métodos tienen la ventaja de que generan áreas muy finas que indican los píxeles importantes, pero por otro lado CAM es fácil de interpretar e implementar. En un final se decide utilizar el CAM debido a su simplicidad.

La decisión sobre cual tipo de representación utilizar no es trivial ya que ambos tipos tienen ventajas, desventajas. De tomar la visualización global es fácil observar si un filtro es irrelevante, pero se haría necesario replantear la forma en que se entrega el *feedback*. Y aun peor es que en cada iteración cambian los filtros por lo tanto la visualización calculada se vuelve obsoleta. Por el lado de las visualizaciones centradas en áreas estas se enfocan en activaciones, por lo tanto, no hay certeza si la selección que el experto llama irrelevante corresponde a un filtro en específico (CAM es suma ponderada) o si este patrón de activación se repite de forma consistente en el *dataset*. Aun así, tienen la ventaja de que permiten reutilizar la selección, ya que las áreas irrelevantes se asocian a un trozo dentro de la imagen, por lo tanto, siguen siendo relevantes al cambiar el filtro.

Se decide visualizaciones por área, ya que es demasiado trabajoso volver a calcular visualizaciones para cada unidad en una capa convolución (fácilmente la cantidad de filtros llega a 50 o 100 filtros) y volver a seleccionar irrelevantes para cada iteración del reajuste.

2.1.4. ¿Cómo entregar información al modelo?

Una vez que se detecta que el modelo observa áreas superfluas o erróneas se debe tener alguna forma comunicar al modelo que deje de observar tales áreas. Además, se busca que tal método no requiera cambios drásticos como insertar o modificar la arquitectura con el objetivo de facilitar su utilización.

Dado esto se evalúan dos opciones en este trabajo.

La primera alternativa se basa en la idea de *Data augmentation*. Se parte de un modelo ya entrenado para el cual se consiguen visualizaciones, el experto observa estas visualizaciones seleccionando áreas que considera irrelevantes, el proceso genera una serie de pares (imagen, área irrelevante) que determinan la información a incorporar al modelo. La idea intuitiva es tratar de corregir esas imágenes ejemplares, mediante crear nuevas imágenes que sirvan de contra ejemplos a las altas activaciones observadas en esos pares. Lo cual se realiza tomando cada una de estas imágenes y dibujando encima del área seleccionada un patrón que genere una activación distinta. El objetivo de este proceso es que el modelo ya no relacione el patrón antiguo con la clase observada lo cual debería incentivar al modelo a dar importancia a otros patrones más relevantes. Se puede observar un diagrama del proceso en la figura 2.4. Ahora conseguir estos reemplazos de forma manual sería difícil y un trabajo intensivo. Afortunadamente como se mencionó se ha investigado bastante sobre modelos generativos de imágenes, los cuales son propicios para automatizar el reemplazo.

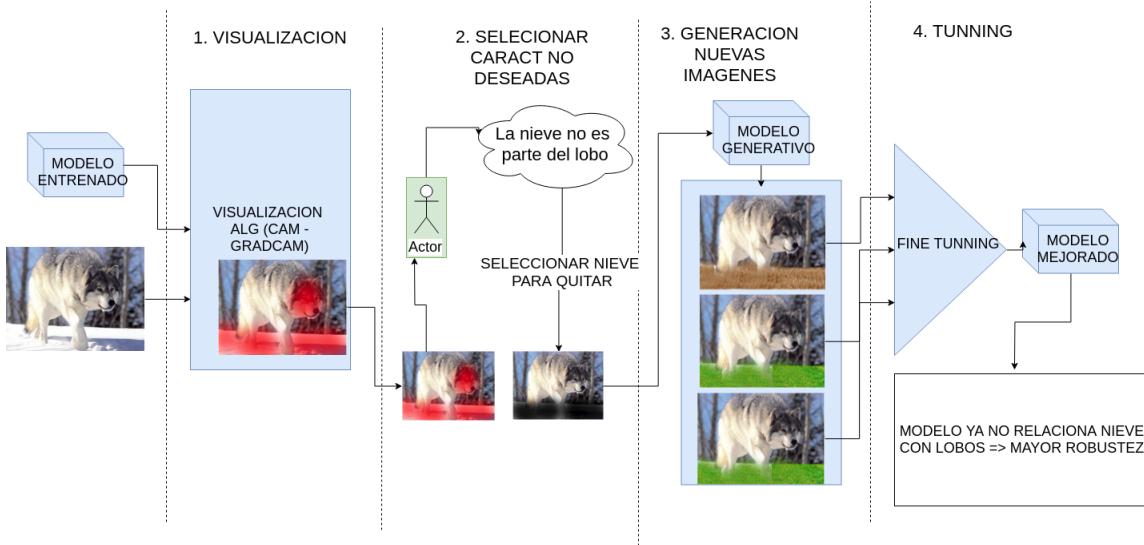


Figura 2.4: Diagrama propuesta por reemplazos.

La segunda forma que se vislumbra para comunicar al modelo y al experto es incorporar las preferencias del experto en la función de pérdida. Esto es un enfoque similar al proceso de regularización, donde se trata de ajustar los pesos a una cierta estructura. El proceso podría reutilizar los pares (imagen, área seleccionada) y penalizar dependiendo de la activación en tales áreas. Si bien el agregar un término que penaliza áreas irrelevantes agrega directamente la información deseada, también produce una nueva dinámica a la hora de optimizar el

modelo. Si no se tiene cuidado esta nueva pérdida puede terminar modificando el modelo de forma de que no logre clasificar correctamente.

2.1.5. Consideraciones cruciales sobre efectividad de la propuesta

Si bien el método es plausible en términos generales, es importante mencionar algunos de los detalles fundamentales para que el método sea exitoso.

En primera instancia es necesario que la visualización sea una muestra fidedigna de lo que el modelo considera importante, si la visualización no está directamente relacionada con la predicción el *feedback* será inútil. Por otro lado, la visualización debe ser extrapolable, se asume que los errores que indican las visualizaciones volverán a ocurrir en diversos otros puntos y por lo tanto afectan la generalización del modelo. Dado los antecedentes presentados en la sección 1.1.7 las visualizaciones parecen cumplir con tales requerimientos, por lo tanto se espera que sean una herramienta útil para detectar características irrelevantes.

Existe también una limitante sobre los tipos de datos que son viables para utilizar con este tipo de método. Es necesario que los datos tengan patrones relevantes y áreas con patrones irrelevantes. Si se analizan *datasets* donde la imagen completa es relevante el método no tiene aplicación. Un ejemplo de tal *dataset* podría ser fotos de animales desde muy cerca donde no existe fondo. Relacionado a este punto es que el experto debe tener certeza de que áreas al menos son erróneas, de lo contrario estaría dando un *feedback* perjudicial al modelo. Debido a esto en este trabajo se eligen *datasets* con claras áreas relevantes e irrelevantes que el experto fácilmente reconozca.

Respecto a la generación de puntos según características irrelevantes, es necesario que el generador de imágenes reemplace con cierta variabilidad el área a reemplazar, si el patrón de reemplazo es muy parecido al trozo original es posible que el modelo no cambie.

Al editar las secciones irrelevantes también se está dejando la mayoría de la imagen original sin cambios. Esto podría ser un problema, ya que existe la posibilidad de que el modelo sobreajuste a los trozos que se mantienen constantes lo cual disminuiría nuevamente la generalización. Si este efecto es considerable se debe de alguna forma suavizar el efecto del reajuste.

Si se agrega un nuevo término a la función de pérdida, como se plantea en la segunda forma de reajuste se introduce un nuevo problema. Es vital el balance entre el término que representa la pérdida del clasificador y el nuevo término. Si la función no se ajusta correctamente la optimización podría terminar empeorando la clasificación del modelo con tal de minimizar el coste del nuevo término.

2.1.6. Resultados esperados

Se plantea que con el reajuste los patrones que producían error desaparecerán de las imágenes seleccionadas y a su vez deberían surgir nuevos patrones que si sean relevantes.

Luego estos nuevos patrones deberían generalizar mejor en los datos de prueba al haber pasado los filtros del experto.

También puede ser que no se encuentren filtros con características nocivas, pero se encuentren filtros de baja relevancia o redundantes, en tal caso es posible filtrar la atención del aprendizaje. Por ejemplo, si se detectan filtros redundantes y se logran cambiar dan la oportunidad al modelo de utilizar tales filtros en aprender una característica distinta que si sea relevante.

2.2. Hipótesis

Dado un clasificador de redes convolucionales ya entrenado y un experto con conocimientos del dominio, es posible seleccionar áreas irrelevantes sobre visualizaciones las cuales después de un algoritmo de reajuste generan una mejora en el desempeño del clasificador con respecto a los datos de prueba.

Esta hipótesis está sujeta a ciertos alcances.

Primero se limita los clasificadores a modelos con amplia literatura en visualización. En este caso se centra el trabajo en las redes convolucionales, las cuales son ideales al estar diseñadas para trabajar con imágenes. Se elige el algoritmo CAM como único algoritmo de visualización debido a su simplicidad, no se estudia o compara con otros. Se asume que los datos poseen tanto áreas relevantes como irrelevantes, se limitan a *dataset* de estas características. Los métodos de reajuste de parámetros se limitan solamente a métodos basados en reemplazo y métodos basados en funciones de pérdida. Sobre la forma de medir la mejora del clasificador se utiliza como en la mayoría de la literatura un conjunto de prueba para medir la generalización del modelo.

2.3. Preguntas de investigación

- ¿Cómo analizar con una métrica objetiva la cantidad de características irrelevantes?
- ¿Es la visualización CAM efectiva para detectar características irrelevantes?
- ¿Cuál es el porcentaje de características irrelevantes?
- ¿Qué método de ajuste de parámetros es más eficaz, en términos de métricas de clasificación?
- ¿Cómo varía la visualización con el método de ajuste?
- ¿Afecta por igual este ajuste a las diversas clases?
- ¿Qué ocurre al aumentar el número de selecciones?

2.4. Objetivos

2.4.1. Objetivo General

Corresponde a generar un algoritmo que mediante un proceso de 3 etapas. Visualización, selección y reajuste produzca mejoras en un clasificador ya entrenado. Lo cual se mide en un aumento de los puntos predichos correctamente en los datos de prueba.

2.4.2. Objetivos Específicos

- Implementar el algoritmo de visualización CAM.
- Crear una interfaz para la selección de áreas irrelevantes.
- Diseñar e implementar el algoritmo de reajuste con reemplazo de áreas irrelevantes.
- Diseñar e implementar el algoritmo de reajuste con funciones de pérdida.
- Realizar el experimento en diversos dataset para una comparación robusta.
- Estudiar los efectos de aumentar la cantidad de selecciones.
- Lograr con el método propuesto eliminar las activaciones en áreas irrelevantes seleccionadas.
- Lograr mejoras de al menos 10 % en *accuracy* del conjunto de prueba para el método propuesto.

2.5. Trabajo relacionado

Respecto a las visualizaciones algunos trabajos relevantes son: Zeiler and Fergus [55] mediante la *DeconvNet* fue uno de los primeros en proponer una forma de obtener mapas de calor para las imágenes de entrada, Yosinski et al. [52] logró crear una herramienta de visualización además de observar imágenes optimizadas para activar cierta unidad, Zhou et al. [56] mediante un ligero cambio de arquitectura permite tener en la capa final filtros que indican áreas de relevancia para una clase, Selvaraju et al. [36] propone un método similar, pero para cualquier arquitectura. Estos métodos entregan indicios sobre la predicción de una red convolucional, pero es importante mencionar que esta sigue siendo un área en constante expansión donde no se ha encontrado una bala de plata que logre entregar la solución definitiva al problema.

Sobre utilizar visualizaciones para mejorar la clasificación, usualmente se utilizan para argumentar ciertos cambios en la arquitectura. En Zeiler and Fergus [55] mediante visualizaciones de activaciones se detectan unidades muy similares, lo cual sugiere reducir la cantidad de filtros. En Tan and Le [46] utilizan visualizaciones CAM para ejemplificar como cambian las características frente a capas más profundas, filtros más anchos o mayor resolución. También es habitual observar visualizaciones de activaciones mediante *T-sne* o *PCA* para tratar de argumentar sobre cierto filtro o para analizar el *dataset* [8]. A conocimiento del autor hasta

el momento no se ha incorporado las visualizaciones dentro del algoritmo de entrenamiento o reajuste.

Respecto a la propuesta la idea base tiene similitudes con el área de investigación llamada *Active learning*. *Active learning* es otra perspectiva del problema de clasificación donde ningún dato ha sido categorizado. Es la responsabilidad del modelo que dato consultar al experto de forma de optimizar la predicción del modelo y el tiempo del experto. Un trabajo relevante es Kyun et al. [23], donde se entrena un modelo no supervisado que se ve mejorado de forma incremental por datos que consulta a un experto. La diferencia con la propuesta es que en tal caso se parte de crear un modelo con la menor cantidad de datos posible, en este caso se parte desde un modelo ya entrenado el cual se trata de mejorar lo más posible.

Dado que la mayoría de los métodos de *machine learning* tratan de lograr con aprendizaje completamente automatizado existe que poca literatura que introduce alguna clase de interacción con algún supervisor o experto. Dentro de los trabajos que si incorporan ayuda de un experto esta el área de investigación *Human in the loop* [50]. En esta se trata de incorporar un experto que ayude a seleccionar los puntos relevantes, que verifique las salidas del modelo y que trate de ajustar los parámetros acordes al modelo. En cierta forma estos procesos ya se realizan al intentar proponer cualquier tipo de algoritmo de *machine learning*. Lo que se propone en Xin et al. [51] y Xin et al. [50] son formas de optimizar tal proceso. Aun así, estas propuestas son bastante distintas a la actual debido a que se centran en el proceso a un nivel meta, es decir, hablan sobre optimizar los procesos no los algoritmos, no intentan incorporar la información que proporciona el experto a la optimización del modelo como en el trabajo actual.

También hay ciertos puntos en común con *Reinforcement learning* (aprendizaje de refuerzo), en este paradigma se optimizan las acciones que un agente realiza de forma de maximizar su recompensa en el tiempo. Existe diversa literatura sobre cómo alinear un agente a ciertas preferencias de un experto [9] [26] [17]. También existen ejemplos de combinar aprendizaje de refuerzo con aprendizaje supervisado [49], pero nunca se ha utilizado algo similar a contrastar las áreas generadas por una visualización.

El reajuste de parámetros para reutilizar un modelo ya entrenado es una forma de *Transfer learning* [48] [45]. *Transfer learning* consiste en transferir un modelo de un problema de clasificación a otro. En la mayoría de los casos esto implica utilizar datos del nuevo dominio con la arquitectura ya entrenada, lo que permite entrenar modelos complejos con una cantidad de datos muy reducida. Este trabajo solamente toma la idea general de reutilizar modelos ya entrenados, las técnicas de *Transfer learning* no son aplicables al contexto actual.

Dentro de la motivación de incorporar nueva información a los datos los métodos más relevantes son *data augmentation*. *Data augmentation* tradicionalmente corresponde a realizar ciertas transformaciones sobre los datos sin alterar su categoría, permitiendo que el modelo tenga mayor información sobre la cual generalizar. Usualmente la técnica más usada es *data warping* que corresponde a trasladar, rotar o cambiar colores para obtener imágenes similares de la misma clase [22]. Existen métodos bastante parecidos a la propuesta en Perez and Wang [29], Antoniou et al. [6], Lemley et al. [27] donde se trata de aprender una forma de hacer *Data augmentation* con una red convolucional auxiliar. Estos métodos usan GANs para generar nuevas imágenes a partir del *dataset* entregado, pero hasta el momento no hay trabajo sobre

cómo controlar las características específicas que se utilizan.

2.6. Resumen

Las redes neuronales utilizan complejas transformaciones no lineales, las cuales se ajustan según los datos observados. Este proceso muchas veces falla en aprender características del todo robustas. Se argumenta que es posible extraer áreas de importancia, tanto de un clasificador como de un experto. El utilizar estas áreas como nuevos datos junto con un reajuste podría mejorar la generalización del clasificador. Dado la bibliografía analizada no hay muchos trabajos que exploren esta dirección.

Capítulo 3

Diseño experimental

En este capítulo se detalla el experimento con el cual se analizará el desempeño del método propuesto. Primero se explica en términos generales el experimento completo y el razonamiento detrás de su diseño. La segunda sección explica los detalles técnicos del experimento tales como el *dataset* a utilizar, la forma de entrenar el clasificador inicial, metodología de selección de áreas, etc. Para finalizar en la sección métodos similares se detallan algoritmos alternativos para comparar con los resultados de la propuesta.

3.1. Experimento en términos generales

La propuesta consiste en 3 etapas. Una etapa de visualización dada por el algoritmo CAM, una etapa de selección dada por el usuario experto y una etapa de reajuste. Para acelerar el desarrollo de los experimentos se realiza las primeras 2 etapas de forma separada al reajuste, de esta forma se realiza selección de áreas irrelevantes una sola vez. Esto también es necesario para comparar los diversos métodos de reajuste, ya que si varían las imágenes seleccionadas su reajuste no es del todo comparable.

Para la visualización se realiza el cálculo del CAM para cada imagen en los datos de entrenamiento. Esto es necesario ya que siempre se debe separar los datos de entrenamiento de los datos de prueba para no sesgar la evaluación. El algoritmo CAM genera por cada imagen C visualizaciones (C son la cantidad de clases), para facilitar el proceso de selección solamente se visualiza la clase verdadera de cada imagen. Esta decisión se basa en que la clase objetivo es la que debe ajustarse para una predicción correcta, por lo tanto, es la que requiere correcciones en su visualización.

El criterio de selección de áreas relevantes se explica en mayor detalle en la sección Selección de áreas irrelevantes.

Los métodos de reajuste se explican en mayor detalle en los siguientes capítulos, pero su funcionamiento en general depende de volver a optimizar la función de pérdida. Debido a esto es necesario definir un criterio de detención. Se utilizó una cantidad fija de 100 iteraciones para

todos los experimentos, donde una iteración corresponde a aplicar *backpropagation* con un subconjunto de 60 puntos. La cantidad de iteraciones fue determinada al observar la función de pérdida y determinar en qué iteración ya no seguía disminuyendo de forma considerable.

La etapa de reajuste tiene por objetivo cambiar las visualizaciones, pero a la vez seguir clasificando correctamente. Para balancear ambos objetivos se decide colocar en los datos un porcentaje de imágenes aleatorias y otro porcentaje con las imágenes de áreas irrelevantes. Si se reajusta solamente con imágenes de áreas irrelevantes es probable que el modelo sobreajuste. En cambio, al utilizar ambos el gradiente promedio debería balancear el cambio. La proporción utilizada fue 90 % imágenes aleatorias, 10 % imágenes generadas es decir 50 aleatorias y 5 generadas por cada 55 imágenes. Esta proporción se determinó separando 200 puntos de los datos de entrenamiento, en estos puntos se experimenta con diversas proporciones y se elige el con mayores métricas en clasificación.

La metodología para evaluar el éxito del método propuesto se basa en dos criterios.

El primer criterio realiza una comparación de diversas métricas entre el clasificador ya entrenado (de ahora en adelante denominado clasificador base) con el clasificador resultante del método propuesto. Esto es relevante debido a que la propuesta plantea mejorar un clasificador ya entrenado, lo que produce que no sean importantes las métricas en términos absolutos sino en términos de diferencias respecto al clasificador base.

Las métricas a utilizar son una serie de métricas usualmente utilizadas para clasificadores, todas estas se evalúan en un *dataset* no utilizado para el entrenamiento. También se utilizan visualizaciones antes y después del método para evaluar en términos cualitativos el cambio de las visualizaciones. Las métricas utilizadas son:

Resumen clasificación: Primero se calculan las matrices de confusión para el clasificador entrenado y luego con estas se calcula precisión, recuperación y F1. Estas métricas permiten conocer clase a clase como el clasificador balancea falsos positivos, falsos negativos y que clases son las más difíciles.

Accuracy: Se mide *accuracy* en conjuntos de entrenamiento, validación y prueba al inicio y fin de los experimentos. El aumento de la generalización debería observarse como un incremento de *accuracy* en el conjunto de validación o de prueba.

Visualizaciones en imágenes seleccionadas: Uno de los objetivos es cambiar las activaciones de las imágenes ejemplares eliminando altas activaciones en las áreas irrelevantes. Por esto se recopilan imágenes en distintas etapas del experimento para evaluar cualitativamente la efectividad del método propuesto. Se colorean estas visualizaciones según el máximo valor por cada CAM, esto es necesario ya que la escala de los valores varía en el tiempo. Debido a esto colores en distintas imágenes no son comparables.

Clasificación en imágenes seleccionadas: A su vez es importante revisar cómo cambian las clasificaciones de las imágenes ejemplares junto con las visualizaciones. Se espera que de mejorar la visualización junto con la clasificación se esté mejorando el modelo en general.

El segundo criterio corresponde a la comparación entre el método propuesto y una lista de

métodos similares. Se busca determinar si los resultados de la propuesta son significativamente mayores, al compararlos con los de algoritmos similares. Mayores detalles de estos métodos en sección 3.3.

Si hay diferencias considerables en las métricas según ambos criterios es razonable mencionar que el método propuesto logra el objetivo de mejorar un clasificador ya entrenado con la ayuda del usuario experto.

3.2. Preliminares

3.2.1. Dataset a analizar

Dado que se requiere tener conocimiento experto del dominio se decide usar un *dataset* donde sea fácil diferenciar áreas relevantes o irrelevantes. Un problema que cumple tales condiciones es la clasificación de animales, toda persona tiene un conocimiento intuitivo de que partes conforman un animal, que se considera fondo y en qué se diferencian diversas especies. Se procede a tomar un subconjunto de 4 clases del *dataset IMAGENET* [34]. Se eligen lobos blancos, coyotes, zorros y zorros blancos. Se eligen animales similares de forma de dificultar al modelo encontrar patrones simples, se espera que esto incentive al modelo a aprender el entorno del animal para clasificar. Se espera que el modelo intente detectar características como la nieve, lo cual puede ser pernicioso. Por ejemplo, si en entrenamiento todas las fotos de lobos blancos, zorros blancos aparecen en la nieve, esto no indica que no pueden aparecer en otros entornos o incluso zoológicos. En la tabla 3.1 se observan algunas imágenes del *dataset* junto con la cantidad de puntos en entrenamiento.

Clases	Lobo blanco	Coyote	Zorro	Zorro blanco
Imágenes				
En entrenamiento	977	1664	1851	1030

Tabla 3.1: Ejemplo imágenes *dataset* subconjunto *IMAGENET*

3.2.2. Entrenamiento inicial

El modelo inicial corresponde a una arquitectura de 5 bloques convolucionales inspirados en la arquitectura VGG16 [39]. Al igual que la arquitectura VGG cada bloque tiene dos convoluciones con *padding SAME* (*padding* que después de convolucionar genera una respuesta del mismo tamaño de la entrada), al igual que en VGG mientras más profundo en la red se baja la resolución, pero se aumentan los filtros. Dado que en un inicio el modelo no alcanzaba

la exactitud esperada se agregaron también bloques de *batch normalization* (capa que ajusta la escala del promedio, desviación estándar de activaciones) [20] y bloques residuales (suman la salida de una capa con su entrada) [19]. Hoy en día es casi estándar utilizar estas capas por lo tanto hace más realista el modelo a analizar, la única limitante es dada por el algoritmo de visualización que obliga a usar solo una capa lineal de salida después de las convolucionales.

El método para conseguir esta arquitectura en específico consiste en agregar o quitar bloques hasta alcanzar una accuracy que se consideró suficiente para el conjunto de entrenamiento. En la figura 3.1 se observa la arquitectura final con los bloques de convolución y las diversas capas ya mencionadas.

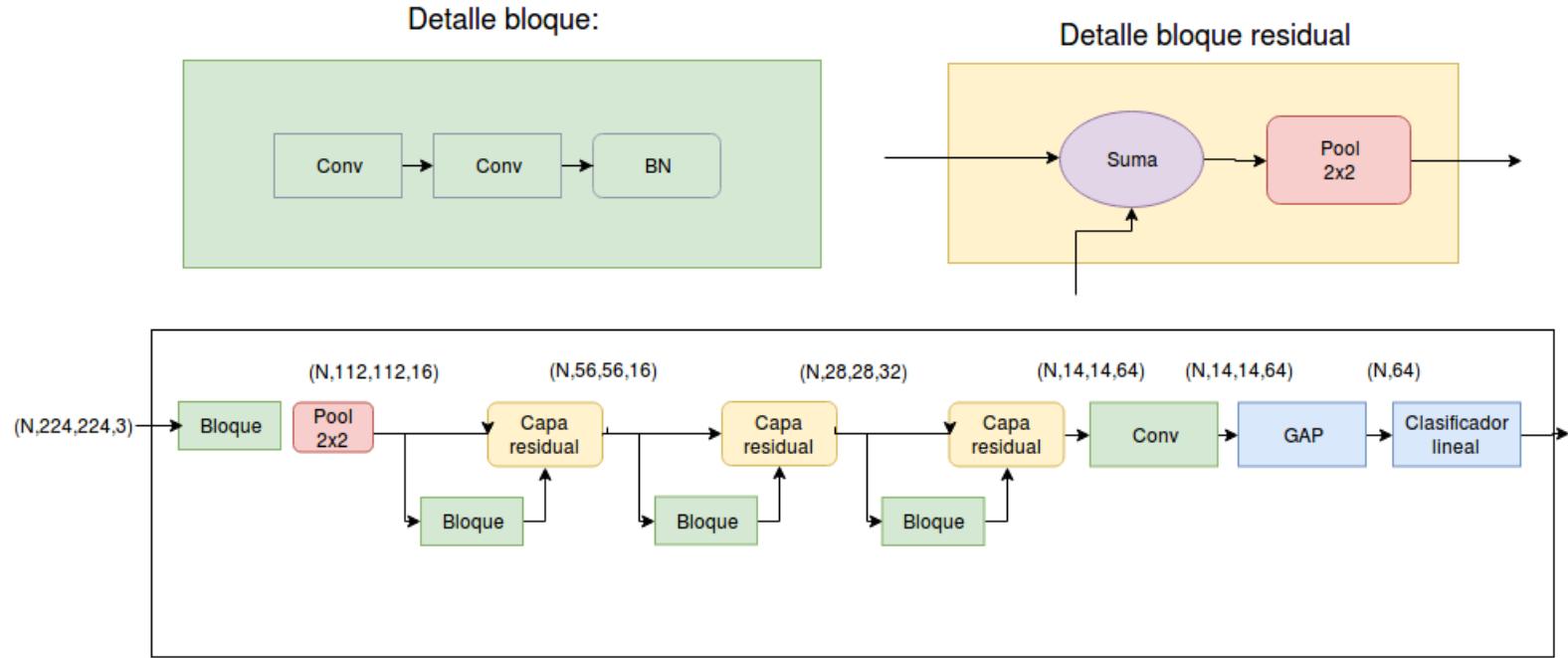


Figura 3.1: Diagrama arquitectura imágenes animales

Acc train	Acc val	Acc test
0.82	0.76	0.75

Tabla 3.2: Resumen modelo después de entrenamiento inicial

Como se aprecia en la tabla 3.2 existe cierto sobreajuste en el modelo entrenado lo que puede ser indicio de que hay características extraídas del conjunto de entrenamiento que son perniciosas al usarse en los conjuntos de validación, prueba. Si se logra identificar y modificar tales características debería existir una mejora en *accuracy*.

3.2.3. Implementación y herramientas utilizadas

Una explicación más extensa del código y el cómo utilizarlo se encuentra en los anexos 8.2.

Para la implementación del método se usa Python 3.6 con diversas librerías científicas como numpy, scipy, pandas, matplotlib. Para implementar los modelos de redes convolucionales se utiliza Tensorflow que es una librería que genera grafos con operaciones matemáticas, la cual entrega todas las herramientas para desarrollar las arquitecturas, funciones de pérdida y visualizaciones que se mencionan.

Todos estos modelos se implementan desde cero dado la literatura mencionada anteriormente, la única excepción es el modelo generativo donde se utilizó el código y el modelo suministrado por Yu et al. [53], principalmente debido a que ya estaba entrenado con el *dataset* de *IMAGENET* que era muy similar al *dataset* seleccionado.

De la implementación, es relevante mencionar la interfaz gráfica creada para el sistema de *feedback*. Se puede observar una demostración de su funcionamiento en la figura 3.2. Debido a que se necesita constantemente visualizar los CAM, seleccionar áreas irrelevantes se diseña una interfaz específicamente con este fin. En esta se selecciona un par (*arquitectura - dataset*) ya entrenado y se carga todas las imágenes del *dataset* de entrenamiento. Como es necesario dar seguimiento a una imagen se implementa un identificador para cada imagen lo que permite iterar al *dataset* en tuplas (índice, imagen, categoría). Después de seleccionar cierta imagen se pueden navegar sus diversas visualizaciones además de seleccionar mediante el ratón qué áreas se determina irrelevante. Se guarda esta información como un diccionario de índices y máscaras binarias, el cual posteriormente el método propuesto lee a la hora de reajustar.

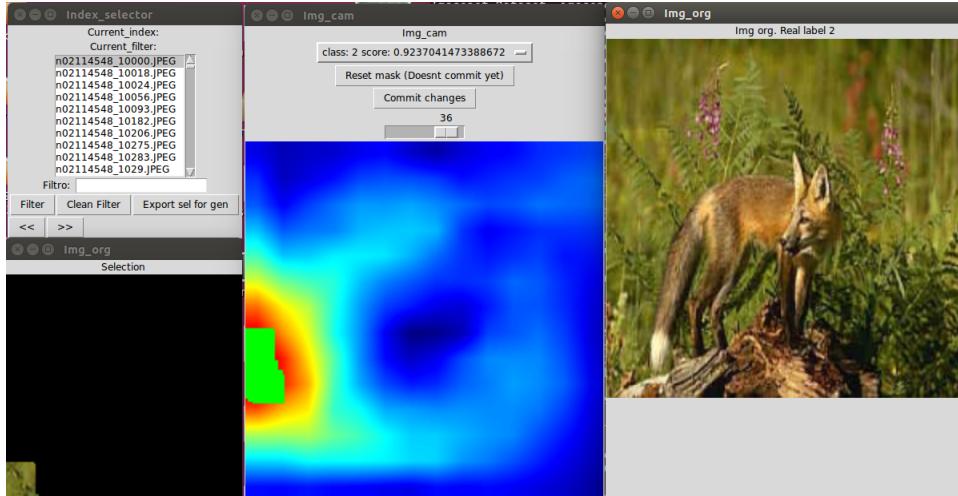


Figura 3.2: En la esquina superior izquierda lista de índices del *dataset*, en la esquina inferior izquierda sección actualmente seleccionada de la imagen. En visualización de CAM y selección en verde. A la derecha imagen original con su categoría correspondiente.

Se libera el código por si es de interés en algún futuro para replicar los experimentos o reutilizar las implementaciones mencionadas. Está disponible en github https://github.com/aferral/generative_supervised_data_augmentation.

3.2.4. Límite superior del desempeño del clasificador

Como se menciona en las primeras secciones si el modelo tiene suficiente capacidad ocurrirá que al aumentar los datos y reentrenar generará un modelo más complejo que mejore su generalización. El método propuesto busca modificar el modelo de forma similar, por lo tanto, es bueno estimar cómo se comportaría el clasificador al recibir nuevos datos. Para simular este proceso se entrena sumando los datos de validación y una porción de los datos de prueba. Esto simplemente para revisar si existen parámetros que usando el mismo modelo entreguen una mejor predicción, algo similar a buscar una cota superior de la *accuracy*. Con este estimado se puede comparar con la mejora que produce el método propuesto.

Método	Train	Validation	Test
Base	0.82	0.76	0.75
K Todos los datos	0.88275	0.87	0.88

Tabla 3.3: *Accuracy* modelo cota

En los resultados de la tabla 3.3 se aprecia que el modelo aún posee un margen de mejora, ya que se puede subir más de 10 puntos porcentuales en *accuracy*.

3.2.5. Selección de áreas irrelevantes

Para seleccionar las imágenes candidatas se exploró el *dataset* de entrenamiento seleccionando todas las imágenes mal clasificadas de las clases lobos blancos, zorros blancos. Por una parte, se eligen estas clases ya que poseen la mayor confusión por lo que tienen mayor posibilidad de mejora y por otro lado las imágenes mal clasificadas deberían indicar problemas con el modelo actual. Importante recordar que estas imágenes, secciones se toman al inicio del experimento, no hay un recálculo del *feedback*.

En cada imagen observada se trata de decidir sobre la visualización del CAM de la clase objetivo ignorando las demás. Esto debido a que se requiere mejorar la visualización que explica la clase objetivo y además se espera que la magnitud de la activación en las clases no objetivo baje automáticamente por la pérdida de clasificación.

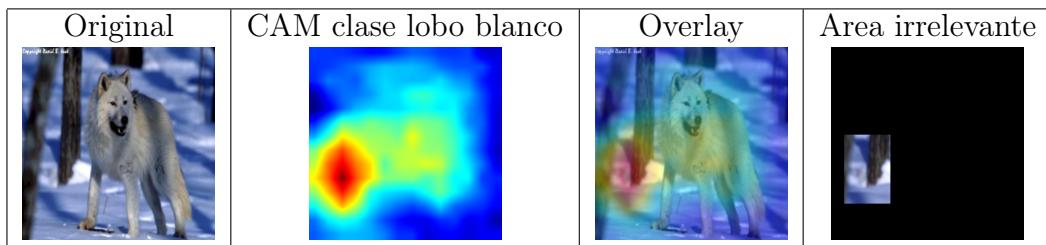


Tabla 3.4: Ejemplo selección áreas irrelevantes.

En la figura 3.4 se observa un ejemplo de una selección. Se parte de observar el CAM de

la clase lobo blanco el cual indica una gran activación para el área de los troncos la cual se selecciona para posterior utilización.

En esta primera etapa solo se utilizaron 5 imágenes para seleccionar áreas irrelevantes, una vez que se logre mejorar tanto visualización como clasificación en este subconjunto sin perjudicar el modelo en general se buscara aumentar el numero de imágenes seleccionadas.

3.3. Métodos similares para comparación

Como se mencionó anteriormente es necesario determinar métodos similares para comparar la propuesta. Se requiere que tales métodos no modifiquen la arquitectura propuesta y de ser posible que no requiera entrenar el modelo desde cero. Dado estas características se plantean 3 experimentos que corresponden a lo que se denomina métodos similares.

- **Entrenamiento continuado:** El primer experimento consiste en simplemente continuar el entrenamiento por 100 iteraciones y observar cómo cambian las métricas analizadas. Este experimento es solamente una prueba de sanidad sobre el modelo, se espera que no existan grandes cambios de *accuracy* ya que después de entrenar por 10 épocas se observó que ya no existían grandes cambios en la función de pérdida.
- **Entrenar con énfasis en imágenes mal clasificadas:** La propuesta trata de enfocar el entrenamiento en ciertas imágenes, para que el modelo corrija características irrelevantes. Pero quizás la mejora aparece simplemente de repetir muchas veces las imágenes mal clasificadas. Se hace necesario comparar con un método donde se continúa entrenando, enfocándose en tales imágenes. Para la implementación de tal método en cada subconjunto de imágenes se extrae el 30 % de forma aleatoria y un 70 % de imágenes mal clasificadas calculadas al inicio del experimento (no se recalcula mal clasificadas por iteración). Este porcentaje se determinó luego de realizar el experimento con unas 5 combinaciones distintas y se selecciona el que mejor resultados obtuvo.
- **Reentrenar con método de *Data augmentation*:** Dada la similitud con la propuesta se hace interesante comparar su desempeño. Se experimenta como mejora el modelo si se entrena desde cero con *Data augmentation* clásico, es decir rotaciones, traslaciones, cambios de color.

3.3.1. Resultados

Método	Train	Validation	Test
Base	0.82025	0.76	0.75
A Continuado	0.8735	0.76	0.72
B Mal clasificados 30 70	0.85475	0.74	0.72
C Data augmentation	0.89125	0.77	0.78

Tabla 3.5: *Accuracy* métodos similares

Método	clases	Train			Validation			Test		
Base	Lobo blanco	P: 0.76	R: 0.74	F1: 0.75	P: 0.67	R: 0.66	F1: 0.66	P: 0.63	R: 0.59	F1: 0.61
	Coyote	P: 0.80	R: 0.87	F1: 0.83	P: 0.73	R: 0.78	F1: 0.75	P: 0.76	R: 0.74	F1: 0.75
	zorro	P: 0.89	R: 0.89	F1: 0.89	P: 0.84	R: 0.84	F1: 0.84	P: 0.81	R: 0.91	F1: 0.86
A Continuado	Zorro blanco	P: 0.79	R: 0.69	F1: 0.74	P: 0.71	R: 0.65	F1: 0.68	P: 0.70	R: 0.63	F1: 0.66
	Lobo blanco	P: 0.89	R: 0.72	F1: 0.79	P: 0.71	R: 0.57	F1: 0.63	P: 0.65	R: 0.53	F1: 0.58
	Coyote	P: 0.87	R: 0.92	F1: 0.90	P: 0.75	R: 0.78	F1: 0.76	P: 0.73	R: 0.70	F1: 0.72
B Mal clasificados	zorro	P: 0.95	R: 0.90	F1: 0.93	P: 0.87	R: 0.82	F1: 0.84	P: 0.84	R: 0.84	F1: 0.84
	Zorro blanco	P: 0.75	R: 0.89	F1: 0.82	P: 0.64	R: 0.79	F1: 0.70	P: 0.58	R: 0.73	F1: 0.64
	Lobo blanco	P: 0.74	R: 0.86	F1: 0.80	P: 0.61	R: 0.69	F1: 0.65	P: 0.59	R: 0.65	F1: 0.62
C Data augmentation	Coyote	P: 0.85	R: 0.90	F1: 0.87	P: 0.73	R: 0.77	F1: 0.75	P: 0.71	R: 0.76	F1: 0.73
	zorro	P: 0.91	R: 0.93	F1: 0.92	P: 0.81	R: 0.85	F1: 0.83	P: 0.83	R: 0.90	F1: 0.86
	Zorro blanco	P: 0.90	R: 0.64	F1: 0.75	P: 0.76	R: 0.51	F1: 0.61	P: 0.67	R: 0.40	F1: 0.50

Tabla 3.6: Reporte clasificación métodos similares.

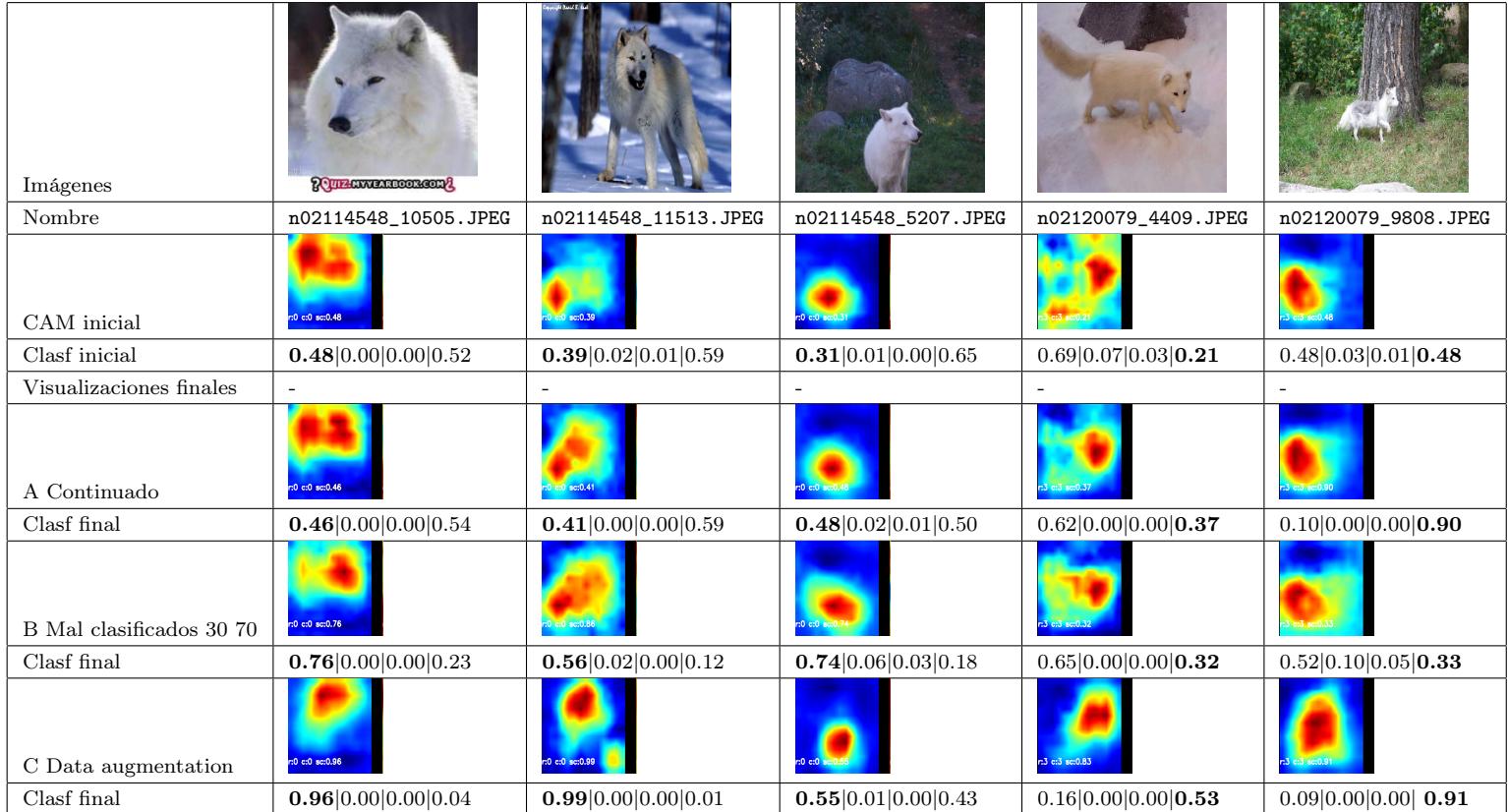


Tabla 3.7: Visualización modelos base

3.3.2. Análisis modelos base y métodos similares

Al observar los reportes de clasificación en la tabla 3.6 se aprecia que las clases de mayor confusión son los lobos blancos con los zorros blancos donde hay diferencias cercanas al

10% en métrica F1. Esta diferencia es entendible al ser su pelaje tan similar, tener hábitats similares y al solo diferenciarse en áreas pequeñas (hocico, tamaño).

Respecto a Coyotes y zorros el clasificador parece bastante competente al lograr una métrica F1 sobre 0,7 (tabla 3.6). Esta gran precisión sobre las clases coyote y zorros se explica en cierto desbalance de clases (ver tabla 3.1 con conteo por clase), al coyotes y zorros tener más puntos la importancia de estas clases en la función de pérdida es mayor.

Se observa que para el caso del entrenamiento continuado (tabla 3.5) hay una mejora apreciable de *accuracy* en entrenamiento, pero no se ve traducida a una mejora en los conjuntos de validación o prueba. Como existe una mejora en entrenamiento, con peor o igual desempeño en validación o prueba indica que el modelo está sobre ajustando a los datos de entrenamiento. Revisando los reportes de clasificación (tabla 3.6) se observa que el modelo mantiene la tendencia observada en el modelo base, los zorros blancos, lobos blancos siguen siendo el problema.

Para el modelo que entreno enfocándose en mal clasificados (tabla 3.5) vuelve a ocurrir cierto sobreajuste que se traduce en inferior desempeño en validación, prueba. Al revisar el reporte de clasificación se aprecia que el real problema fue los zorros blancos los cuales empeoran considerablemente aun cuando no hay indicio de esto en las métricas para entrenamiento.

En *data augmentation* se observa una clara mejora en todas las clases (tabla 3.6). Con una mejora del 3% en el conjunto de prueba. Las clases mejoran para todas las métricas respecto al modelo base. Si bien las clases más difíciles siguen siendo lobos blancos con zorros blancos ahora sus métricas no están tan bajas relativas a coyotes y zorros.

Las visualizaciones graficadas (tabla 3.7) muestran el CAM de la clase objetivo. Un detalle interesante sobre las visualizaciones es que diversas veces las áreas de mayor activación apuntan a lugares que parecen extraños. Por ejemplo, en la primera imagen la clase lobo se enfoca en un manchón cercano a las orejas, en la segunda imagen se enfoca en los troncos, en la tercera en la esquina izquierda, lo mismo ocurre en las imágenes restantes.

La evolución de las diversas visualizaciones no es considerable si se comparar con la visualización del modelo base lo cual es interesante al existir cambios apreciables en las métricas globales de *accuracy* y las probabilidades de predicción por imagen. Esto puede indicar que los cambios se están produciendo relativos a magnitud y no en las capas convolucionales, mismas activaciones convolucionales con pesos del clasificador de mayor magnitud tendrán un mayor valor de predicción. *Data augmentation* muestran clasificaciones bastante altas y las visualizaciones parecen bastante certeras en la mayoría de los casos.

3.4. Resumen

Se explica el experimento, el cual se basa en recopilar métricas de clasificación, con lo cual se espera detectar mejoras en la predicción. Se recopilan los cambios en métricas respecto al clasificador entrenado (clasificador base) y se recopilan resultados para métodos similares.

Se detallan aspectos técnicos importantes para los próximos capítulos, como son *dataset*, interfaces utilizadas y la metodología de selección de áreas irrelevantes. Los resultados indican que los métodos similares (excepto entrenar desde cero con *data augmentation*) no producen mejora, por lo que si el método produce mejoras tendrá que compararse con *data augmentation* que resulta ser el mejor método similar. También se aprecia de las visualizaciones que el modelo se enfoca en diversas áreas no deseadas.

Capítulo 4

Reajuste basado en reemplazos

Como se mencionó la propuesta es un proceso de 3 etapas. La etapa importante es el reajuste, en este capítulo se explica la forma de reajustar parámetros mediante reemplazar en las imágenes seleccionadas las áreas irrelevantes. Este reemplazo genera nuevas imágenes, las cuales pueden ser utilizadas para aplicar la optimización de los pesos nuevamente. El capítulo explica una serie de algoritmos propuestos, se observan los resultados del experimento y por último se analizan los resultados.

4.1. Resumen de propuesta con reemplazos

El reajuste basado en reemplazos se puede definir en 4 etapas (ver figura 4.1):

Visualización: Se parte por visualizaciones de las áreas de la entrada que contribuyen mayormente a una clasificación. En este caso la visualización utilizada es el CAM que permite generar una imagen para cada clase indicando áreas de mayor activación para esa clase en particular. Para acotar la cantidad de imágenes a observar se visualiza las imágenes mal clasificadas en el conjunto de entrenamiento, se espera que estas imágenes sean más relevantes para el reajuste al presentar una clasificación incorrecta.

Selección: Un experto en el dominio visualiza si el modelo de clasificación observa características no robustas para la predicción de la categoría. De encontrar tales áreas se utiliza una herramienta que permite seleccionar y generar una máscara binaria que se almacena para los siguientes pasos.

Generación: Una vez identificadas estas áreas se edita las imágenes seleccionadas. Con el modelo generador mencionado anteriormente se generan diversas imágenes sin la característica a eliminar. Todas las imágenes generadas se agregan al *dataset* original.

Ajuste: La última etapa es usar este nuevo *dataset* para un reajuste que debería mostrar contraejemplos a la característica no robusta. Se espera que estos cambios no solo eliminen las características irrelevantes, sino que mejoren métricas cuantitativas como el error de

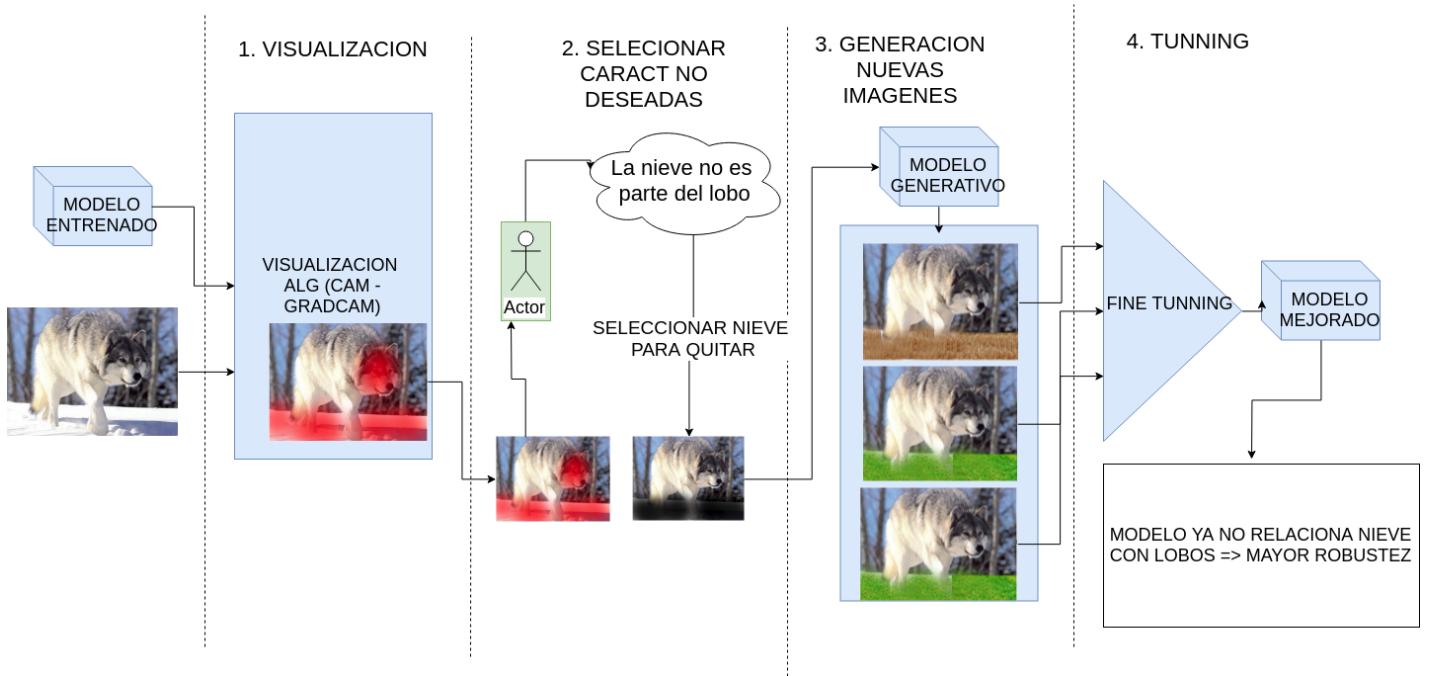


Figura 4.1: Diagrama propuesta en 4 etapas.

precisión, recuperación, F1 u otras similares sobre el problema de clasificación.

4.2. Reemplazo generativo

El reemplazo generativo corresponde a editar el área seleccionada en la imagen utilizando un modelo de *image inpainting*, el cual reemplaza partes específicas de una imagen. Se utilizó el modelo detallado en Yu et al. [53] el cual solamente requiere una imagen y una máscara para realizar el reemplazo.

El objetivo de tal reemplazo es cambiar el patrón de activación en esa área, lo cual al reajustar introducirá nuevos patrones en el modelo.

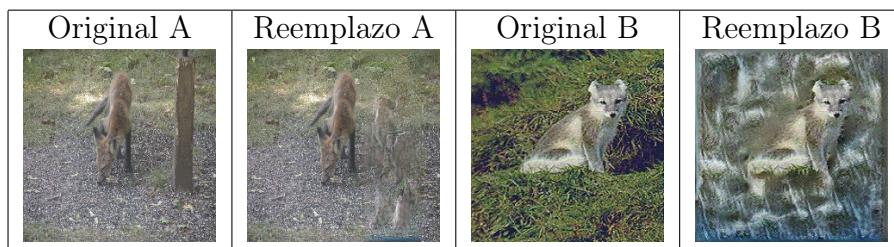


Tabla 4.1: Ejemplo reemplazos generativos

Como se observa en la tabla 4.1 ejemplos de los reemplazos. Se aprecia que este método logra reemplazar de forma satisfactoria patrones pequeños como quitar el pie del ciervo o el agua en la foto del zorro.

de la izquierda. Y a su vez también logra reemplazar patrones extensos como quitar todo el pasto de la imagen de la derecha. A pesar de estos resultados el algoritmo presenta una desventaja al no ser posible controlar el resultado, el reemplazo es determinista y no es posible cambiarlo. Esto es un problema si la imagen de reemplazo es prácticamente idéntica a la original lo cual ocurre si el patrón seleccionado es un fondo sin mucha variación. Debido a esto se inspeccionan visualmente las imágenes de reemplazo antes de proceder, de tener patrones muy similares se intenta editar la máscara o cambiar el ejemplo.

4.3. Reemplazo recorte aleatorio

Para comparar el reemplazo generativo es interesante ver que tan necesario es tener un reemplazo dado el contexto o basta con variar radicalmente la activación. Debido a esto se propone como reemplazo un área aleatoria de cualquiera de las otras imágenes. Sin embargo, puede ser problemático al ser posible que incluya recortes de pedazos de animales de otras clases. Por esto también se revisa antes de utilizarse en el reajuste. En la tabla 4.2 se observa un ejemplo al reemplazar un árbol con este método.

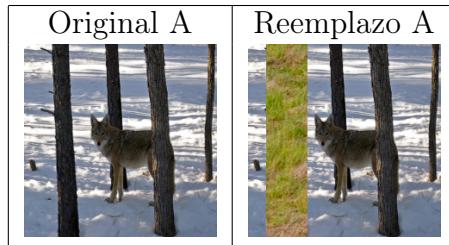


Tabla 4.2: Ejemplo reemplazos con recortes

4.4. *Dropout* selectivo

Otra idea que surge en la etapa de selección y reemplazo fue de editar la ya mencionada capa *dropout* para permitir seleccionar qué áreas apagar. El razonamiento se basa en que los reemplazos inducen a cambios en las activaciones lo cual es una versión menos extrema que apagar ciertas unidades. Por lo tanto, es interesante ver los efectos de *dropout* de forma comparada. Pero este *dropout* no apaga sus unidades de forma uniforme ni aleatoria, sino que se centra en las últimas capas convolucionales de mayor activación en el área seleccionada.

Algoritmo:

1. **Entrada:** Dado imagen y la área seleccionada como irrelevante (máscara binaria asociada a la imagen seleccionada).
2. **Preparar visualización:** Conseguir activaciones convolucionales finales ponderadas por pesos de la clase objetivo (mismo proceso del CAM).

3. **Calcular activación por filtro:** Utilizar la máscara 2D en cada filtro y obtener promedio de activación. Lo cual indica que tan activada está el filtro en el área seleccionada como irrelevante.
4. **Ordenar:** Dado esos promedios ordenar de mayor a menor.
5. **Filtrar:** Se eligen los k promedios más altos y se los lleva a 0.
6. **Aplicar corrección:** Aplicar factor de corrección a las activaciones no apagadas.
7. **Salida:** Tal como el algoritmo *dropout* modifica las activaciones de la red. En este caso se obtiene activaciones filtradas según magnitud en área irrelevante. Esta activación se utiliza para la siguiente capa.

Para los resultados de esta sección se utiliza $k = 20$ lo cual en porcentaje es apagar aproximadamente el 30% de las unidades. Al igual que en *dropout* se aplica un factor de corrección sobre las demás activaciones para mantener el promedio similar en entrenamiento y prueba similares.

4.5. Resultados reemplazos

Método	Train	Validation	Test
Base	0.82025	0.76	0.75
D Generativo	0.86925	0.76	0.75
E Recortes	0.84275	0.75	0.75
F Dropout selectivo	0.87775	0.76	0.75

Tabla 4.3: Accuracy reemplazos

Método	clases	Train	Validation			Test		
Base	Lobo blanco	P: 0.76 R: 0.74 F1: 0.75	P: 0.67	R: 0.66	F1: 0.66	P: 0.63	R: 0.59	F1: 0.61
	Coyote	P: 0.80 R: 0.87 F1: 0.83	P: 0.73	R: 0.78	F1: 0.75	P: 0.76	R: 0.74	F1: 0.75
	zorro	P: 0.89 R: 0.89 F1: 0.89	P: 0.84	R: 0.84	F1: 0.84	P: 0.81	R: 0.91	F1: 0.86
	Zorro blanco	P: 0.79 R: 0.69 F1: 0.74	P: 0.71	R: 0.65	F1: 0.68	P: 0.70	R: 0.63	F1: 0.66
D Generativo	Lobo blanco	P: 0.82 R: 0.80 F1: 0.81	P: 0.67	R: 0.66	F1: 0.66	P: 0.66	R: 0.69	F1: 0.68
	Coyote	P: 0.88 R: 0.89 F1: 0.88	P: 0.76	R: 0.74	F1: 0.75	P: 0.76	R: 0.68	F1: 0.72
	zorro	P: 0.93 R: 0.91 F1: 0.92	P: 0.84	R: 0.83	F1: 0.84	P: 0.85	R: 0.90	F1: 0.87
	Zorro blanco	P: 0.78 R: 0.82 F1: 0.80	P: 0.69	R: 0.74	F1: 0.71	P: 0.66	R: 0.68	F1: 0.67
E Recortes	Lobo blanco	P: 0.74 R: 0.82 F1: 0.78	P: 0.62	R: 0.69	F1: 0.65	P: 0.67	R: 0.72	F1: 0.70
	Coyote	P: 0.80 R: 0.94 F1: 0.87	P: 0.69	R: 0.86	F1: 0.77	P: 0.69	R: 0.83	F1: 0.75
	zorro	P: 0.95 R: 0.88 F1: 0.91	P: 0.89	R: 0.77	F1: 0.83	P: 0.86	R: 0.82	F1: 0.84
	Zorro blanco	P: 0.85 R: 0.63 F1: 0.72	P: 0.80	R: 0.56	F1: 0.66	P: 0.78	R: 0.53	F1: 0.63
F Dropout	Lobo blanco	P: 0.82 R: 0.84 F1: 0.83	P: 0.67	R: 0.71	F1: 0.69	P: 0.61	R: 0.63	F1: 0.62
	Coyote	P: 0.85 R: 0.93 F1: 0.89	P: 0.73	R: 0.79	F1: 0.76	P: 0.73	R: 0.77	F1: 0.75
	zorro	P: 0.95 R: 0.90 F1: 0.92	P: 0.87	R: 0.81	F1: 0.84	P: 0.86	R: 0.88	F1: 0.87
	Zorro blanco	P: 0.86 R: 0.79 F1: 0.82	P: 0.73	R: 0.69	F1: 0.71	P: 0.69	R: 0.59	F1: 0.64

Tabla 4.4: Reporte clasificación reemplazos

Imágenes					
Nombre	n02114548_10505.jpeg	n02114548_11513.jpeg	n02114548_5207.jpeg	n02120079_4409.jpeg	n02120079_9808.jpeg
Mascaras					
CAM inicial					
Clasf inicial	0.48 0.00 0.00 0.52	0.39 0.02 0.01 0.59	0.31 0.01 0.00 0.65	0.69 0.07 0.03 0.21	0.48 0.03 0.01 0.48
Visualizaciones finales	-	-	-	-	-
D Generativo					
Clasf final	0.99 0.00 0.00 0.01	1.00 0.00 0.00 0.00	1.00 0.00 0.00 0.00	0.00 0.00 0.00 1.00	0.00 0.00 0.00 1.00
E Recortes					
Clasf final	0.98 0.00 0.00 0.02	0.97 0.01 0.00 0.03	0.99 0.00 0.00 0.00	0.00 0.00 0.00 1.00	0.01 0.00 0.00 0.98
F Dropout selectivo					
Clasf final	0.80 0.00 0.00 0.20	0.95 0.00 0.00 0.05	0.91 0.00 0.00 0.08	0.07 0.00 0.00 0.93	0.01 0.00 0.00 0.99

Tabla 4.5: Visualización reemplazos

Al observar la tabla 4.3 se aprecia que los métodos mejoran solamente en entrenamiento manteniéndose constantes en validación o test respecto al modelo base. En cierta forma es un resultado positivo, ya que no sobreajusta como ocurre con los métodos similares A,B (entrenamiento continuado y entrenamiento enfocado a mal clasificados).

Al analizar los reportes de clasificación (tabla 4.4) se puede observar lo siguiente:

En el reemplazo genérico se aprecia mejora consistente excepto en coyotes del conjunto de prueba.

En recortes hay diversos problemas con las clases zorros, zorros blancos las cuales se ven compensadas en parte por las mejoras en clase coyotes, lobos blancos.

En *dropout selectivo* se aprecian resultados similares al reemplazo generativo, pero ahora cae en la clase de zorros blancos.

De estos valores se puede observar que el sobreajuste en las clases zorros blancos, lobos

blancos (ya que solo se usaron reemplazos de estas clases) no parece ocurrir de hecho ocurre lo contrario. El detalle muestra que existen mejoras, pero muchas veces a costos de empeorar en otras clases (observar tabla 4.4 en conjunto de prueba). Dado que no se alcanzan mejoras superiores a 1% en métricas de *accuracy* estas mejoras no son significativas.

En cuanto a visualizaciones (tabla 4.5) ocurre algo inesperado se observa que se refuerza la visualización inicial ignorando el área seleccionada donde debería bajar la activación, lo cual es exactamente lo opuesto al efecto deseado. Y no es un problema de que los pesos no cambien, ya que se aprecia que las clasificaciones cambian radicalmente desde una predicción errónea a una alta confianza en la clase verdadera. Por otro lado, este tipo de activaciones recuerda al entrenamiento continuado donde se aprecian un refuerzo a la visualización inicial. Todo esto implica que la solución planteada tiene un problema en los supuestos, ya que se esperaba que se corrigieran las activaciones señaladas.

Los resultados no fueron los esperados, se plantearon diversas variaciones del experimento para cambiar los resultados. Se intentó bajando la cantidad de imágenes normales, aumentar el *learning rate*, utilizar más imágenes generadas, cambiar la selección de imágenes, dar peso extra a las imágenes seleccionadas sin embargo los resultados indican que no hay mejora considerable y peor aún las visualizaciones no se alejan de las áreas irrelevantes.

4.6. Revisión de los supuestos

El método hasta ahora es el siguiente. Se observan visualizaciones de predicciones, se identifican áreas no deseadas, luego se generan imágenes con diversos reemplazos. Este nuevo *dataset* se utiliza para reajustar el modelo al continuar la aplicación de la función de pérdida. La hipótesis consiste en que al insertar variación en esas áreas y aplicar repetidas veces *backpropagation* se está obligando al modelo a reconocer nuevos patrones lo que producirá un cambio en activaciones antiguas, borrando la incidencia de patrones que el experto considera dañinos.

Para ahondar en la validez de la hipótesis es importante entender cuál es la función de gradiente que se utiliza y cómo se relaciona con el modelo propuesto, ya que de entender el gradiente se puede entender cómo se mueven los parámetros dada la pérdida observada en los datos.

Para esto se calculó el gradiente de la pérdida respecto a los pesos de la última capa y el gradiente de la pérdida respecto a la última capa convolucional. Se analizan solo estos dos debido a que estos se utilizan en la visualización y por ende en la clasificación final. Por otro lado, por la regla de la cadena los gradientes se van construyendo de atrás hacia adelante lo que significa que los gradientes de las últimas capas son fundamentales para analizar el efecto posterior en la red.

A continuación, un diagrama simplificado de la red convolucional y los gradientes relevantes. Todos estos gradientes se calculan en detalles en los anexos 8.1.2, 8.1.1, 8.1.3, 8.1.5.

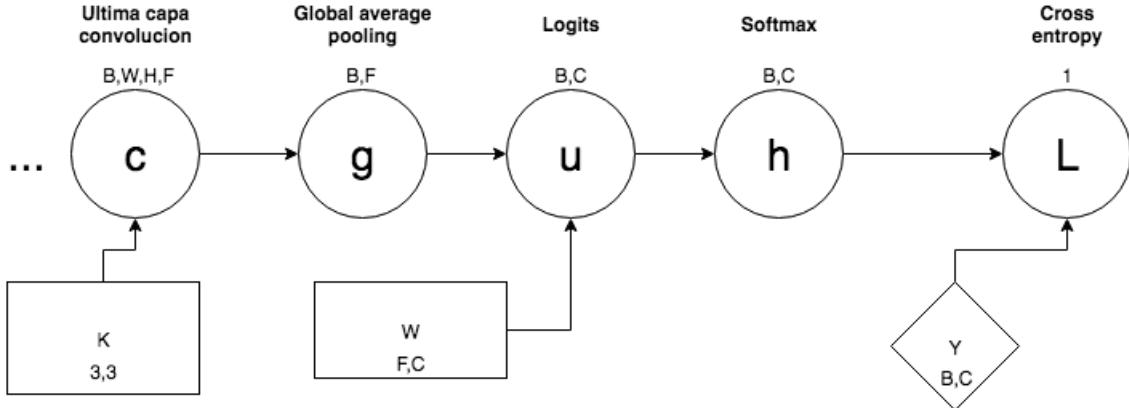


Figura 4.2: Diagrama arquitectura

$$\frac{\partial L}{\partial c_{a,b,c,d}} = -\frac{1}{N} \frac{1}{w \cdot h} \sum_s^s (Y_{a,s} - h_{a,s}) \cdot W_{d,s} \quad (4.1)$$

$$\frac{\partial L}{\partial W_{m,n}} = -\frac{1}{N} \sum_r^r (Y_{r,n} - h_{r,n}) \cdot g_{r,m} \quad (4.2)$$

Para un análisis más cercano a la realidad en las ecuaciones se asume que la red se alimenta por un conjunto de imágenes, que las categorías se entregan como vectores *one hot* y que el $W_{m,n}$ posee las unidades en sus columnas. Para consultar cómo se llegan a estas fórmulas consultar el apéndice 8.1.

Si se observa la fórmula de los pesos del clasificador (fórmula 4.2) $\frac{\partial L}{\partial W_{m,n}}$ se puede ver un término $(Y_{r,n} - h_{r,n})$ este término puede parecer extraño (esto sólo es válido si se usan categorías *one hot* ver apéndice para detalles) pero es realmente bastante simple. Es una clase de interruptor que cambia según si se está analizando un peso asociado a la clase correcta o no. Si se está en una unidad correspondiente a la clase objetivo de la imagen r tenemos que $Y_{r,n} = 1$ y el término se vuelve $(1 - h_{r,n})$, este término es mayor mientras más lejos esté la predicción de 1, lo que penaliza a la predicción si no está con alta confianza indicando la clase real. Si se analiza un peso que no corresponde a la clase de la imagen r se tendrá que $Y_{r,n} = 0$ lo que produce $(-h_{r,n})$.

Al analizar el gradiente sobre W primero es necesario borrar el signo negativo, ya que al minimizar se sigue el sentido contrario al gradiente. El término se puede ver como una suma ponderada, sobre los diversos puntos donde se evalúa el gradiente. El término $g_{r,m}$ (activación escalar siempre positiva obtenida del promedio del filtro r, m) indica que tan activa estaba la característica asociada al peso $W_{r,m}$. En términos simples al ser los términos $g_{r,m}$ siempre positivos (promedios de salidas *ReLU*) se tienen dos casos: Si es la clase correcta el peso aumenta según distancia a 1, si es la clase incorrecta el peso disminuye tan lejos esté del 0, se realiza este proceso por todos los elementos del conjunto y se saca promedio.

El primer problema es que g es siempre positivo al ser parte de la salida con *ReLU*, esto provoca que el gradiente este dominado por el signo de $(Y_{r,n} - h_{r,n})$ lo que va generar que los pesos de la última capa siempre suban para la clase correcta y bajen para las demás. Esto provoca que los pesos de la última capa refuerzen toda activación observada en la clase real, lo cual también refuerza patrones espurios observados en esta clase. La hipótesis inicial planteaba que la imagen con reemplazo empujará al modelo a aprender otros patrones, pero es un proceso lento, ya que los pesos solamente pueden subir o quedar igual (ya que son pesos asociados a la clase objetivo de la imagen). Aun así, es posible cambiar la visualización, ya que es posible cambiar la proporción de la activación. Este cambio ocurre si se logra que el g asociado a la característica irrelevante esté siempre en cero, los demás pesos al tener activaciones mayores aumentaron en mayor forma lo que después de repetidos pasos hará ínfima la contribución de la característica irrelevante en la capa *softmax*. Pero para que esto ocurra debería existir un proceso que se preocupe de reemplazar el área con un patrón que genere poca activación en el g a borrar lo cual actualmente no se realiza. Y aun así existe el segundo problema.

El segundo problema es que el gradiente se satura rápidamente. Después de unas pocas iteraciones de repetir la misma imagen el término $h_{r,n}$ se acerca demasiado a 1 para la clase correspondiente y a 0 para las demás, lo que detiene al gradiente en el término $(Y_{r,n} - h_{r,n})$ de seguir propagándose. En experimentos se observó que después de 30 o 40 iteraciones el término se satura. Se intentó realizar un proceso algo más agresivo agregando solamente las imágenes a cambiar sin imágenes aleatorias de relleno, pero resulta en rápido sobreajuste al ajustarse mucho a las imágenes observadas que se ve reflejado en una caída del *accuracy* a 0,6.

Respecto a la ecuación del gradiente en la capa convolucional la fórmula es similar, pero con una sumatoria sobre las clases s . Esta indica que crece a medida que su contribución sobre todas las clases coincide con la categoría es decir que el peso asociado sea negativo si no es de esa categoría y peso positivo si es de esa categoría. El problema sigue siendo el término que se satura fácilmente y además se observa que no hay dependencia de b, c es decir todo valor dentro de un filtro tiene la misma culpa, esto es debido a que la arquitectura usa *global average pooling* lo cual es un promedio por filtro. Sería útil que pudiera culpar ciertas áreas más que otras.

Si bien el *dropout selectivo* no genera reemplazos sufre un problema parecido al solamente poder enmascarar el gradiente lo cual mantiene los mismos problemas de arriba.

De todo este análisis se aprecia que ciertas intuiciones sobre el modelo eran erradas. Para el caso de una persona si se le entregan dos imágenes que se caracterizan igual, exactamente iguales excepto por un hidrante verde la persona deduce que el hidrante verde no es relevante al estar ausente en la segunda imagen. En cambio, una red convolucional agregaría importancia igual al patrón hidrante presente, y al patrón hidrante ausente (el cual no es completamente opuesto al hidrante) y sacaría promedio lo cual no elimina el hidrante de su filtro. Al menos en la formulación actual el problema no tiene un mecanismo que indique un **evita observar tal patrón**. Esta arquitectura en la última capa aprende de subir todos los pesos asociados a clases positivas (según activación), bajar todos los pesos asociados a clases negativas (según activación) y luego promediar sobre muchas imágenes.

Con todo lo anterior el problema se resume en que la pérdida no tiene un mecanismo que garantiza eliminar el área irrelevante. Por un lado, si bien es posible aprender nuevos patrones es un proceso que requiere de bastantes iteraciones manejando las activaciones, no se puede dirigir a las áreas específicas y después de saturar la pérdida se detiene completamente.

4.7. Resumen

Se plantean diversos métodos basados en el reajuste por reemplazos. Este método genera nuevas imágenes con las cuales se procede reajustar los parámetros con la misma función de pérdida de clasificación. Los resultados muestran que no hay un cambio considerable en métricas de clasificación. Tampoco existe cambios en la visualización. Mediante analizar el gradiente de la función de pérdida se detecta que el modelo por reemplazos no es el indicado para cambiar visualizaciones. Esto se explica en la rápida saturación del gradiente al presentar imágenes muy similares y que no existe método específico para penalizar activaciones en las áreas seleccionadas.

Capítulo 5

Reajuste basado en funciones de pérdida

Dado lo anterior era necesario cambiar el enfoque utilizado. El nuevo enfoque debe garantizar balancear la clasificación correcta de las diversas imágenes y por otro lado ajustar las visualizaciones dado las máscaras seleccionadas. Las visualizaciones dependen de los pesos de modelo, por lo tanto, lo que en definitiva se busca es penalizar si estos inducen a altas activaciones en las áreas seleccionadas. Penalizar los pesos del modelo para cumplir ciertas preferencias no es una idea nueva ya que se usa extensamente en diversos métodos de regularización como son L1 (mantener pesos esparcidos), L2 (mantener suavidad) la diferencia es que en este caso se va a enfocar en activaciones. Respecto a penalizar activaciones existen ejemplos de este enfoque en la literatura, se aprecia una idea similar en Salimans et al. [35] para asegurarse que dos modelos adversarios mantuvieran activaciones similares, o en Gatys et al. [14] donde compara activaciones de una imagen generadas con cierto estilo y la imagen original para realizar *style transfer*. En general se usan como una función secundaria a la función de pérdida teniendo cuidado de balancear los efectos de ambos términos.

5.1. Pérdida por activación selectiva (PAS)

En un inicio se pensó en penalizar la activación del CAM, ya que permite penalizar exactamente la visualización observada, pero tiene el problema de que termina bajando los pesos de la capa de clasificación en vez de los filtros, también complica el diseño ya que hay que decidir sobre cuál CAM se está enfocando. Por eso se decide aplicar la penalización sobre los 64 filtros de la última capa convolucional que generan la visualización CAM.

La función que se plantea es la siguiente:

$$L = CE + \frac{\lambda}{N} \sum_a \sum_d \sum_b \sum_c |Mask_{(a,b,c)} \cdot C_{(a,b,c,d)}| \quad (5.1)$$

Esto soluciona los problemas planteados anteriormente ya que por una parte hay un término directamente asociado a la activación irrelevante y por otro ya no es dependiente de la

saturación del gradiente.

Al analizar la ecuación 5.1, se tiene que el primer término de la izquierda CE corresponde a la pérdida por entropía cruzada $-\frac{1}{N} \sum_i^N \sum_j^C Y_{ij} \cdot \log(h_{ij})$ que se mantiene para seguir clasificando las imágenes. El segundo término es una triple sumatoria (se ignora la última sumatoria que simplemente calcula un promedio) que realiza lo siguiente: toma la máscara 2D asociada a la imagen y la multiplica punto a punto con las activaciones convolucionales, esta máscara binaria entonces filtra tales regiones dado el área seleccionada. El valor absoluto es necesario si se utilizan capas convolucionales sin $ReLU$ en la salida, ya que de existir valores negativos esta función no tiene mínimo y los pesos bajan indefinidamente. Esta sumatoria se divide por N para tener un promedio por conjunto y además se multiplica por una escalar λ que balancea su contribución respecto a la entropía cruzada.

5.1.1. Selección de λ

El correcto ajuste de λ es fundamental para esta función de pérdida. Si se utiliza una λ muy baja no habrá influencia en la eliminación de activaciones, por otro lado, si la λ es muy alta se tiende a apagar todas las activaciones lo que perjudica la clasificación. Para determinar el valor se parte del término $\frac{1}{14 \cdot 14 \cdot 64}$ (64 activaciones de 14×14) que sería el factor a aplicar si se tomara un promedio de todas las unidades. Luego se considera el tamaño de la máscara como el 20 % de la totalidad de la imagen lo que deja el término en $\frac{5}{14 \cdot 14 \cdot 64}$. Por último, se crea una serie de candidatos para explorar valores cercanos. En los 9 candidatos está en valor base, el valor base duplicado hasta 4 veces y el valor base dividido en 5 hasta 4 veces. Esta lista de candidatos se evalúa en el conjunto de validación con los siguientes resultados.

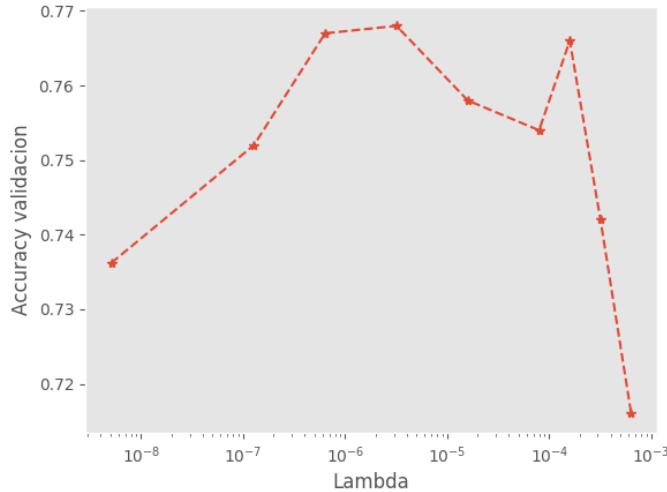


Figura 5.1: Selección de λ dado accuracy de validación

Los resultados en la figura 5.1 muestran un máximo cercano a $\frac{1}{5 \cdot 14 \cdot 14 \cdot 64}$ el cual se utiliza como el λ en los experimentos que siguen.

5.1.2. Aumento de las máscaras

También se analiza la posibilidad de que la cantidad de imágenes seleccionadas sea muy baja como para producir cambios significativos. Debido a esto se selecciona una cantidad aún mayor de imágenes y ahora con secciones mucho más grandes. Se incrementa a 300 imágenes con sus máscaras. El criterio de selección cambia respecto a las imágenes anteriormente usadas, en vez de visualizar el CAM se selecciona todo lo que corresponde a fondo en la imagen. De esta forma se trata de incentivar al modelo a aprender patrones de los animales y no del ambiente. Debido a que iterar con 300 imágenes a la vez está por sobre las capacidades técnicas disponibles se utiliza el mismo conjunto de imágenes de tamaño 55 en el cual 50 son imágenes aleatorias, 5 son imágenes que se sacan al azar de la lista de imágenes con máscaras.

Debido a la cantidad enorme de máscaras se trató de realizar trazos grandes lo cual muchas veces deja lugares sin llenar, afortunadamente las máscaras deben pasar por un proceso de re-escalamiento para ajustarse a las dimensiones de las activaciones (las activaciones son de 14x14 las imágenes de 224x224). Este proceso termina por llenar las áreas no tan marcadas. Ejemplos de estas imágenes se observan en la tabla 5.1.

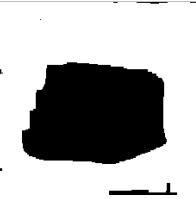
Imagen	Mascara	Imagen con mascara	Mascara luego de resize
			

Tabla 5.1: Ejemplo de máscaras

Dado la cantidad de máscaras es relevante observar el efecto de entrenar con cantidades incrementales de máscaras. Se realizan 9 incrementos y se evalúa *accuracy* en validación.

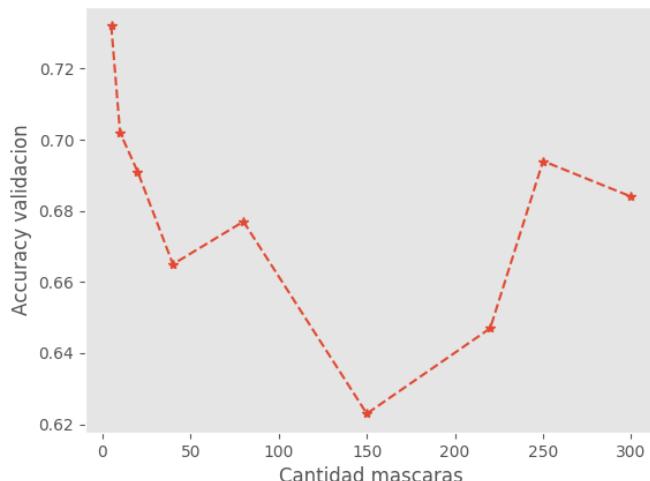


Figura 5.2: Selección de cantidad máscaras

Los resultados aparecen en la figura 5.2. En primer lugar, todos los puntos muestran una baja en *accuracy*, la cual aumenta hasta llegar a 150 donde al sube algo, pero siempre muy por debajo de la *accuracy* inicial (0.76). Al parecer el modelo no logra adaptarse a un cambio tan brusco, ya que se le indica que debe bajar activaciones prácticamente en toda la imagen. Probablemente los filtros no logran ser tan específicos para activarse solo en áreas seleccionadas lo que termina por empeorar el modelo.

5.1.3. Detalle resultados PAS

Método	Train	Validation	Test
Base	0.82025	0.76	0.75
F PAS	0.8755	0.76	0.72
H pérdida Cam 250	0.7565	0.69	0.64

Tabla 5.2: *Accuracy* PAS

Método	clases	Train	Validation	Test
Base	Lobo blanco	P: 0.76 R: 0.74 F1: 0.75	P: 0.67 R: 0.66 F1: 0.66	P: 0.63 R: 0.59 F1: 0.61
	Coyote	P: 0.80 R: 0.87 F1: 0.83	P: 0.73 R: 0.78 F1: 0.75	P: 0.76 R: 0.74 F1: 0.75
	zorro	P: 0.89 R: 0.89 F1: 0.89	P: 0.84 R: 0.84 F1: 0.84	P: 0.81 R: 0.91 F1: 0.86
	Zorro blanco	P: 0.79 R: 0.69 F1: 0.74	P: 0.71 R: 0.65 F1: 0.68	P: 0.70 R: 0.63 F1: 0.66
F PAS	Lobo blanco	P: 0.86 R: 0.81 F1: 0.83	P: 0.69 R: 0.67 F1: 0.68	P: 0.60 R: 0.58 F1: 0.59
	Coyote	P: 0.84 R: 0.92 F1: 0.88	P: 0.73 R: 0.77 F1: 0.75	P: 0.70 R: 0.73 F1: 0.72
	zorro	P: 0.93 R: 0.92 F1: 0.92	P: 0.82 R: 0.83 F1: 0.82	P: 0.82 R: 0.89 F1: 0.85
	Zorro blanco	P: 0.85 R: 0.79 F1: 0.82	P: 0.76 R: 0.69 F1: 0.72	P: 0.65 R: 0.56 F1: 0.60
H PAS 250	Lobo blanco	P: 0.52 R: 0.89 F1: 0.66	P: 0.43 R: 0.76 F1: 0.55	P: 0.43 R: 0.72 F1: 0.54
	Coyote	P: 0.90 R: 0.67 F1: 0.77	P: 0.77 R: 0.55 F1: 0.64	P: 0.79 R: 0.52 F1: 0.63
	zorro	P: 0.85 R: 0.89 F1: 0.87	P: 0.78 R: 0.80 F1: 0.79	P: 0.78 R: 0.86 F1: 0.81
	Zorro blanco	P: 0.81 R: 0.53 F1: 0.64	P: 0.66 R: 0.45 F1: 0.53	P: 0.54 R: 0.37 F1: 0.44

Tabla 5.3: Reporte clasificación PAS

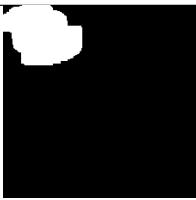
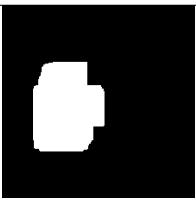
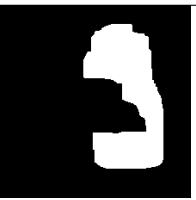
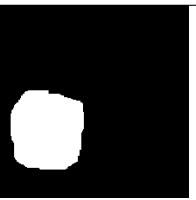
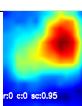
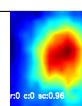
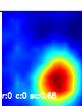
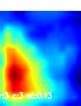
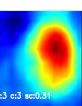
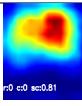
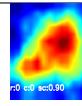
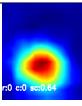
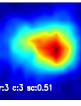
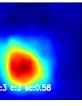
Imagenes					
Nombre	n02114548_10505.JPG	n02114548_11513.JPG	n02114548_5207.JPG	n02120079_4409.JPG	n02120079_9808.JPG
Máscaras					
CAM inicial	 r0 c0 sc=0.46	 r0 c0 sc=0.38	 r0 c0 sc=0.31	 r3 c3 sc=0.21	 r3 c3 sc=0.48
Clasf inicial	0.48 0.00 0.00 0.52	0.39 0.02 0.01 0.59	0.31 0.01 0.00 0.65	0.69 0.07 0.03 0.21	0.48 0.03 0.01 0.48
Visualizaciones finales	-	-	-	-	-
F PAS	 r0 c0 sc=0.95	 r0 c0 sc=0.96	 r0 c0 sc=0.68	 r3 c3 sc=0.13	 r3 c3 sc=0.31
Clasf final	0.95 0.00 0.00 0.05	0.96 0.01 0.00 0.02	0.68 0.03 0.02 0.27	0.57 0.22 0.08 0.13	0.13 0.24 0.32 0.31
H PAS 250	 r0 c0 sc=0.81	 r0 c0 sc=0.90	 r0 c0 sc=0.64	 r3 c3 sc=0.51	 r3 c3 sc=0.58
Clasf final	0.81 0.00 0.00 0.18	0.90 0.02 0.01 0.07	0.64 0.10 0.08 0.18	0.44 0.03 0.03 0.51	0.33 0.05 0.04 0.55

Tabla 5.4: Visualizaciones PAS

Respecto a las métricas de *accuracy* (tabla 5.2) se observa que para el caso de las 5 máscaras ocurre algo similar a lo observado en otros métodos es decir otra vez mejora en *train*, se estanca en *val* y disminuye en *test*. En cambio, en el entrenamiento con las 250 máscaras se nota una marcada disminución en todos los conjuntos.

Observando el reporte de clasificación (tabla 5.3) se observa que en validación hay una mejora en todas las clases excepto zorro, en prueba ligera disminución en todas. Para el caso de las 250 máscaras no hay mejora alguna.

Respecto a visualizaciones (tabla 5.4) finalmente existen cambios marcados. Para el caso del método F las visualizaciones tienden a colocarse prácticamente en el rincón opuesto a la selección, lo cual es bueno por una parte ya que indica que la función está logrando su objetivo, pero por otro lado el rincón opuesto tampoco tiene al animal. Respecto a utilizar las 250 máscaras el resultado es mixto hay algunas que logran el objetivo otras se mantienen (caso de los zorros blancos).

En cuanto a clasificaciones de las imágenes seleccionadas (tabla 5.4), ambos métodos son insatisfactorios al no alcanzar probabilidades altas o directamente fallar en clasificar. Otra vez los zorros blancos son los peor clasificados.

Al controlar el experimento se apreció que las probabilidades realizan un notorio zig-zag.

Para tratar de entender mejor este fenómeno se grafica la pérdida en el tiempo junto con la clasificación en el tiempo.

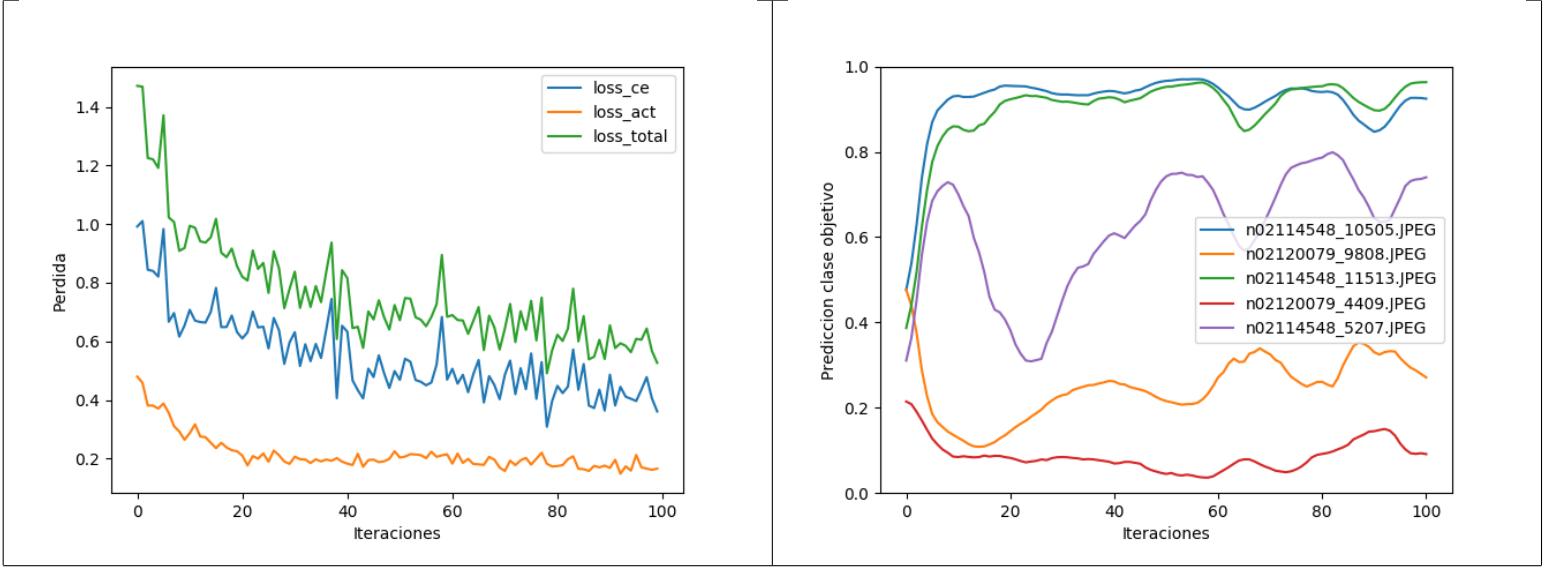


Tabla 5.5: Ruido en la pérdida y oscilaciones en la clasificación

Se puede ver en la tabla 5.5 que la pérdida es algo ruidosa lo cual no es extraño, ya que se gráfica por iteración, también se aprecia una tendencia a la baja en el tiempo lo cual es un buen signo. El componente que penaliza la activación (en amarillo) baja hasta la iteración 30 donde se mantiene constante. Dado las visualizaciones parece aceptable ese nivel de activación. Al observar las predicciones se observa un problema, en las primeras iteraciones se aprecia que los lobos blancos suben mientras los zorros blancos bajan. Esto no es sorprendente, ya que se espera que la clasificación bajase inicialmente al perturbar las activaciones, pero también se espera que pasado un tiempo la tendencia se revirtiera. Se observa que las predicciones tienen comportamientos como ciclos de subida y bajada. Lo más probable sea el efecto del gradiente del clasificador mezclado con el gradiente de las activaciones que empujan los parámetros a un ciclo.

Los resultados no han logrado mejora en las métricas globales y tampoco se ha logrado corregir las imágenes ejemplares (solo se ha logrado corregir activaciones). Se espera que de lograr solucionar el problema con la predicción, se logren cambios tangibles en métricas como *accuracy* fuera del entrenamiento.

5.2. Pérdida por activación selectiva ajustada (PASA)

El parámetro λ es fijo para todas las imágenes y para todas las iteraciones. El que oscila la predicción y que ciertas imágenes no se clasifiquen correctamente mientras en otras se mantienen bajas es un indicio de que λ no debería ser para todos igual o que quizás debería disminuir. Dado esto se plantea que la mejor forma de variar el término sería controlarlo mediante la probabilidad de la clase objetivo. El razonamiento es que el término de activación

tiene que de dar una clase de impulso inicial para que el filtro se centre en otro objetivo, pero no debería estorbar a medida que el clasificador aumenta su predicción.

Se plantean la siguiente función de pérdida.

$$\begin{aligned}
 L &= \frac{1}{N} \sum_a (CE_{(a)} + Act_{(a)} \cdot Ajuste_{(a)}) \\
 L &= \frac{1}{N} \sum_a \left(CE_{(a)} + Act_{(a)} \cdot B^{\frac{ht_{(a)}}{th}} \right) \\
 L &= \frac{1}{N} \sum_a \left(CE_{(a)} + \left(\frac{1}{F} \frac{1}{m+1} \sum_d \sum_b \sum_c |Mask_{(a,b,c)} \cdot C_{(a,b,c,d)}| \right) \cdot B^{\frac{ht_{(a)}}{th}} \right)
 \end{aligned} \quad (5.2)$$

Esta nueva función (ecuación 5.2) elimina la necesidad de λ al utilizar diversos promedios. Primero promedia la sumatoria por filtro dado la cantidad de elementos en la máscara binaria $\frac{1}{m+1}$ (se suma 1 para evitar división por cero en casos de cero elementos). Luego se promedia por todos los filtros en el término $\frac{1}{F}$. Después de ambos promedios queda un escalar por cada imagen del conjunto, este escalar se denomina el término $Act_{(a)}$. Para lograr que el término $Act_{(a)}$ cambie dado la probabilidad se utiliza el término $Ajuste_{(a)}$ que decae exponencialmente. El factor B se eleva a $\frac{ht_{(a)}}{th}$ donde $ht_{(a)}$ representa la probabilidad de la clase real, y th es un factor de tolerancia. La función del segundo término es decaer exponencialmente al aumentar la probabilidad de la clase correcta, la velocidad de decaimiento depende del th .

Una característica de esta fórmula es que posee el mismo mínimo que la función anterior es decir clasificación correcta, activación seleccionada en cero, pero a su vez el efecto de la activación va disminuyendo. También se plantea que el factor de activación no debe desaparecer del todo, lo cual se logra al estar la probabilidad ht_a acotada por 1, por lo que el segundo término se mantiene en el mínimo de $A \cdot B^{\frac{1}{th}}$. Se puede apreciar tal efecto en la figura 5.4.

Para determinar los valores de th , B se consigue un histograma del término A sin utilizar máscara alguna. Con este estimado se calcula un promedio, mínimo y máximo. En este histograma (figura 5.3) se observa un promedio de 0.6, mínimo de 0.1 (activación posee *ReLU*, por lo tanto el mínimo teórico es 0) y un máximo de 1.9. Dado estos valores de activación y tomando un factor B de 0.4 el coste de activación estaría fluctuando en un rango equivalente a una entropía cruzada de un probabilidad entre 0.96 a 0.46. Después de una exploración de parámetros similar a la de λ se tiene que B de 0.4 y th de 0.7 entregan la mayor *accuracy* en validación.

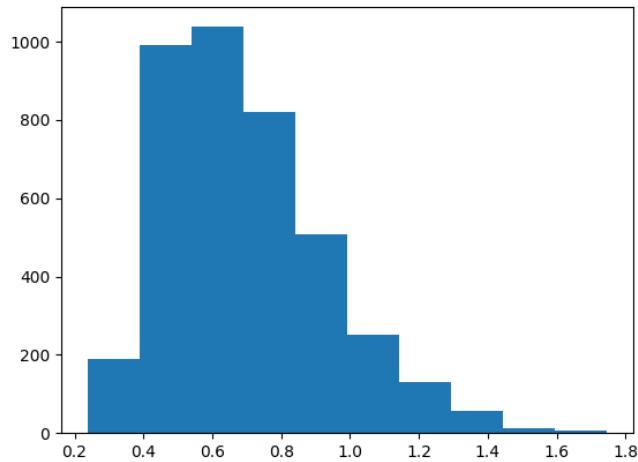


Figura 5.3: Histograma término $Act_{(a)}$ sin mascaras

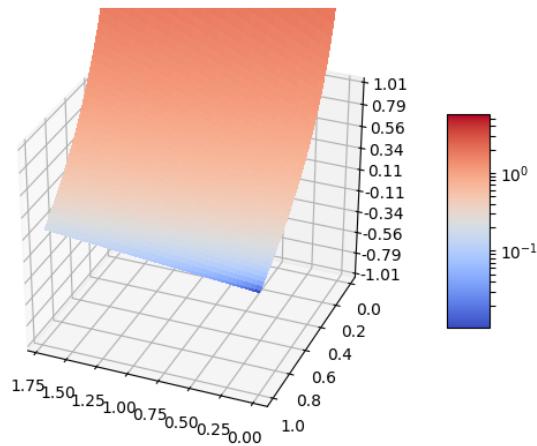


Figura 5.4: Superficie de valores A , ht_a

Dado estos cambios ya no es obligatorio ajustar parámetros, ya que estas variables son solo tolerancias en contraste a λ que de no ajustarse correctamente el problema degenera. Si B , th se mueven en rangos de 0-1 la función de pérdida tienden a converger a la misma solución de la entropía cruzada.

5.2.1. Resultados pérdida PASA

Método	Train	Validation	Test
Base	0.82025	0.76	0.75
I Pérdida PASA	0.874	0.78	0.75
J Pérdida PASA 250	0.799	0.67	0.63

Tabla 5.6: *Accuracy* PASA

Método	clases	Train			Validation			Test		
Base	Lobo blanco	P: 0.76	R: 0.74	F1: 0.75	P: 0.67	R: 0.66	F1: 0.66	P: 0.63	R: 0.59	F1: 0.61
	Coyote	P: 0.80	R: 0.87	F1: 0.83	P: 0.73	R: 0.78	F1: 0.75	P: 0.76	R: 0.74	F1: 0.75
	zorro	P: 0.89	R: 0.89	F1: 0.89	P: 0.84	R: 0.84	F1: 0.84	P: 0.81	R: 0.91	F1: 0.86
	Zorro blanco	P: 0.79	R: 0.69	F1: 0.74	P: 0.71	R: 0.65	F1: 0.68	P: 0.70	R: 0.63	F1: 0.66
I PASA	Lobo blanco	P: 0.89	R: 0.76	F1: 0.82	P: 0.75	R: 0.60	F1: 0.67	P: 0.70	R: 0.57	F1: 0.63
	Coyote	P: 0.84	R: 0.91	F1: 0.88	P: 0.74	R: 0.82	F1: 0.77	P: 0.74	R: 0.75	F1: 0.74
	zorro	P: 0.92	R: 0.93	F1: 0.92	P: 0.84	R: 0.85	F1: 0.84	P: 0.81	R: 0.92	F1: 0.86
	Zorro blanco	P: 0.83	R: 0.82	F1: 0.82	P: 0.75	R: 0.74	F1: 0.74	P: 0.68	R: 0.62	F1: 0.65
J PASA 250	Lobo blanco	P: 0.59	R: 0.85	F1: 0.70	P: 0.47	R: 0.69	F1: 0.56	P: 0.44	R: 0.64	F1: 0.52
	Coyote	P: 0.85	R: 0.85	F1: 0.85	P: 0.72	R: 0.70	F1: 0.71	P: 0.71	R: 0.67	F1: 0.69
	zorro	P: 0.93	R: 0.85	F1: 0.89	P: 0.84	R: 0.75	F1: 0.79	P: 0.86	R: 0.79	F1: 0.82
	Zorro blanco	P: 0.79	R: 0.58	F1: 0.67	P: 0.59	R: 0.47	F1: 0.52	P: 0.36	R: 0.27	F1: 0.31

Tabla 5.7: Reporte clasificación PASA

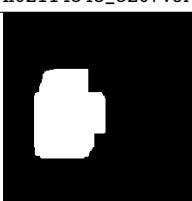
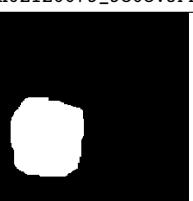
					
Imagenes	n02114548_10505.jpeg	n02114548_11513.jpeg	n02114548_5207.jpeg	n02120079_4409.jpeg	n02120079_9808.jpeg
Mascaras					
CAM inicial	 r0 c0 sc=0.48	 r0 c0 sc=0.38	 r0 c0 sc=0.31	 r3 c3 sc=0.21	 r1 c1 sc=0.48
Clasf inicial	0.48 0.00 0.00 0.52	0.39 0.02 0.01 0.59	0.31 0.01 0.00 0.65	0.69 0.07 0.03 0.21	0.48 0.03 0.01 0.48
Visualizaciones finales	-	-	-	-	-
I PASA	 r0 c0 sc=0.96	 r0 c0 sc=0.96	 r0 c0 sc=0.95	 r3 c3 sc=0.98	 r3 c3 sc=0.95
Clasf final	0.96 0.00 0.00 0.04	0.98 0.01 0.00 0.00	0.95 0.02 0.00 0.03	0.01 0.00 0.00 0.99	0.03 0.01 0.00 0.95
J PASA 250	 r0 c0 sc=0.97	 r0 c0 sc=0.98	 r0 c0 sc=0.93	 r3 c3 sc=0.90	 r3 c3 sc=0.93
Clasf final	0.97 0.00 0.00 0.02	0.98 0.00 0.00 0.02	0.53 0.04 0.03 0.11	0.08 0.01 0.01 0.90	0.06 0.01 0.00 0.93

Tabla 5.8: Visualizaciones PASA

En *accuracy* (tabla 5.6) se observan los mejores valores hasta el momento con una mejora del 2 %, aunque el conjunto de prueba sigue sin mejora. Al utilizar el conjunto de las 250 máscaras los resultados siguen siendo inferiores incluso al modelo base.

En cuanto a visualizaciones (tabla 5.8) ambos métodos se alejan de la mayoría de las áreas seleccionadas lo cual es el efecto deseado.

Las clasificaciones de las imágenes seleccionadas se logran corregir y alcanzan ahora alta probabilidad.

Se grafican las imágenes con sus máscaras y activaciones para poder apreciar mejor la corrección de las áreas (observar figura 5.9). Se observa que si bien se logra que las áreas importantes tengan altas activaciones hay intersección entre áreas no deseadas y altas activaciones especialmente para zorros blancos. Se intentó eliminar este efecto subiendo el factor B lo cual efectivamente fuerza al modelo a apagarse en esas áreas, pero termina empeorando la clasificación global. En resumen, se logra un intercambio de cierta activación en el área seleccionada a cambio de una mejor predicción.

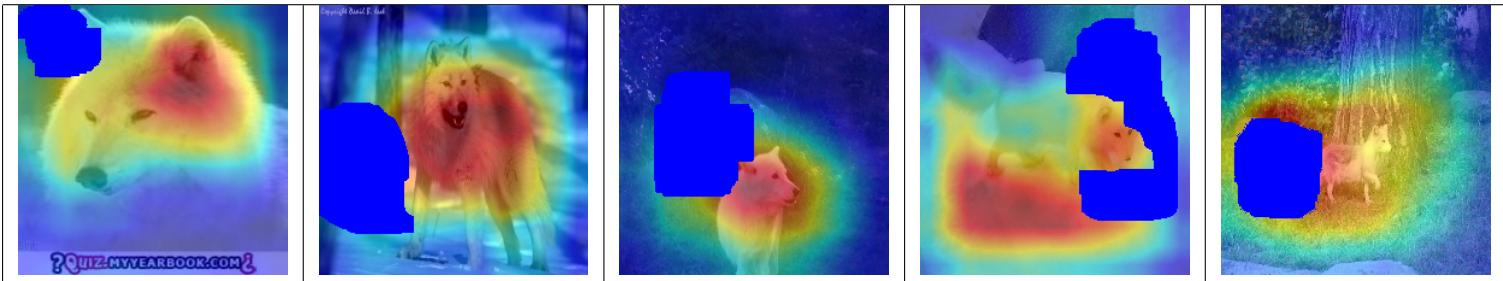


Tabla 5.9: Imágenes seleccionadas del modelo I con su activación final y área seleccionada

Se vuelve a graficar en la figura 5.10 la pérdida junto con la predicción en el tiempo para contrastar con los resultados de la función anterior. En la pérdida se aprecia una suave caída del término de activación, junto con una muy ruidosa entropía cruzada que parece estancarse. En las predicciones se logra el efecto deseado al subir rápidamente las clasificaciones para todos los datos. Si bien aún poseen oscilaciones estas son menores y no impiden la convergencia a la probabilidad 1.

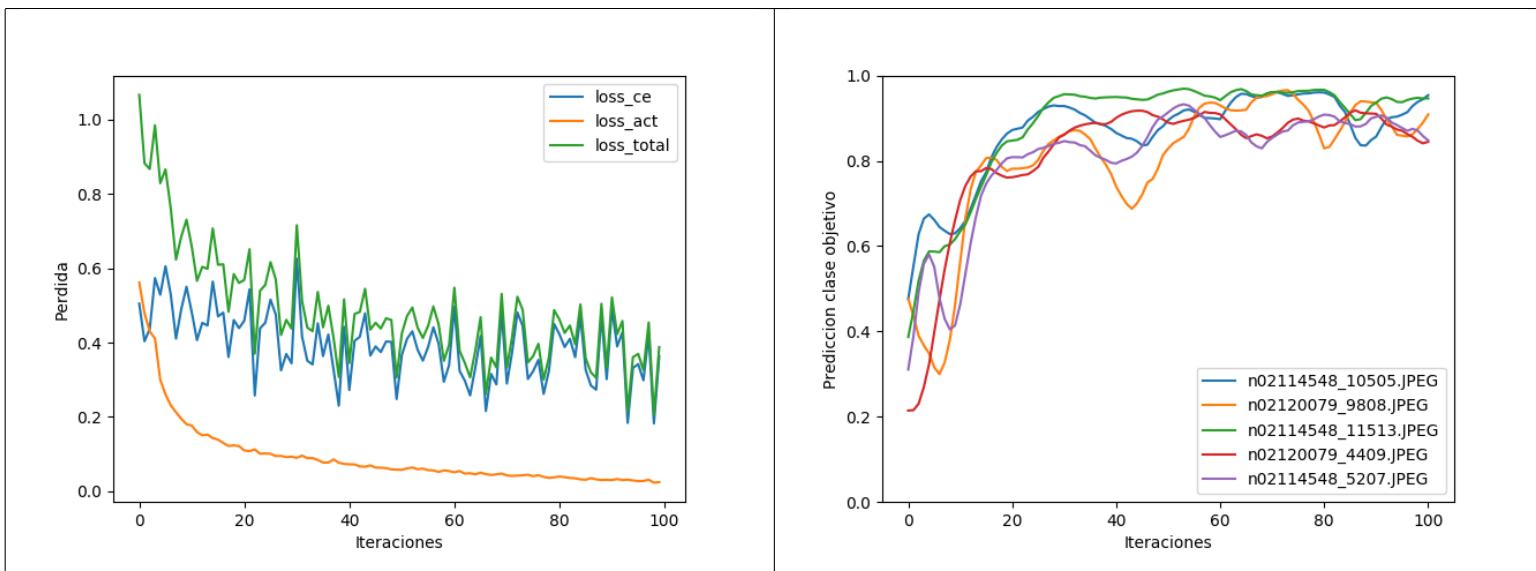


Tabla 5.10: Ruido en la pérdida y oscilaciones en la clasificación

Los resultados indican que se logra el efecto de cambiar activaciones en el modelo y a la vez mantener clasificación correcta, sin embargo esto no produce efectos positivos significativos al modelo en general. Solo se ha logrado corregir imágenes y mantener el modelo sin cambios a nivel global.

Respecto al segundo tipo de máscara que utiliza trozos extensos de la imagen ha demostrado nuevamente que solo empeora el modelo.

5.3. Resumen

Se presentan dos funciones de pérdida con las cuales reajustar las imágenes seleccionadas, pérdida por activación selectiva (PAS) y pérdida por activación selectiva ajustada (PASA). Se detalla los desafíos de plantear una función tales como la búsqueda de parámetros, el balance con la función de clasificación y las oscilaciones. Los resultados para el caso de PASA finalmente se logra consistentemente ajustar la visualización a las áreas seleccionadas, pero no se detectan mejoras considerables en métricas de clasificación.

Capítulo 6

Resultados para diversos *datasets*

Dado los resultados obtenidos, es importante contrastar si estos se mantienen al aplicar el mismo método a conjuntos de datos distintos. En este caso se analizan tres nuevos conjuntos de datos. Un *dataset* de dibujos (o *sketches*), un *dataset* de símbolos en latex y por último imágenes de rayos X en aeropuertos.

6.1. Sketches

El dataset *quickdraw* [1] son 50 millones de dibujos repartidos en 345 categorías distintas. La mayor parte de los datos fueron recopilados mediante un juego donde usuarios dibujaban una categoría que salía en pantalla y un modelo de clasificación intentaba adivinar su clase. Debido a esto hay bastante variabilidad en la calidad de los datos e incluso es posible encontrar dibujos mal categorizados. La motivación de usar tal *dataset* es verificar el cómo reacciona el método propuesto ante datos con pocos elementos característicos como es el caso de los trazos donde las secciones importantes son unas pocas líneas y el resto es el fondo blanco.

Las clases seleccionadas fueron: buses, aviones, autos, patines. En la figura 6.1 se observan algunos ejemplos.

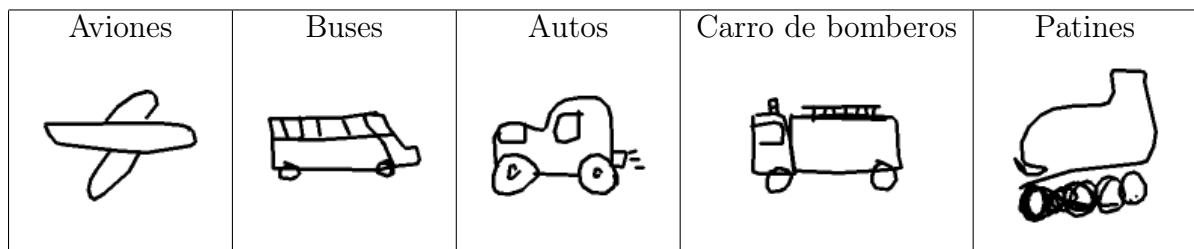


Tabla 6.1: Ejemplo imágenes *dataset sketches*

La arquitectura utilizada es la misma del experimento con animales. El entrenamiento inicial alcanza un 0.95 en *accuracy* en conjunto de entrenamiento 0.91 en validación, 0.91

en prueba. Lo cual es bastante más alto de lo que se esperaba ya que son clases bastante parecidas con dibujos muy variables.

6.1.1. Selección de máscaras

La selección de áreas irrelevantes fue difícil, ya que las visualizaciones muchas veces indican áreas grandes y al parecer sin ningún patrón presente. Esto se debe a que los filtros tratan de relacionar las imágenes con cantidad de espacio en blanco. Por ejemplo, dibujos de aviones tienden a ocupar más espacio que los dibujos de buses. En general en este *dataset* las visualizaciones CAM ayudan bastante poco ya que apuntan a área muy grandes, no hay certeza sobre la sección es específico que genera la mayor activación.

6.1.2. Resultados

Para este *dataset* por brevedad solamente se utiliza el entrenamiento continuado como modelo similar, debido a que los resultados del *dataset* de animales muestra que sus resultados son prácticamente idénticos. Para comparar se toma el modelo con pérdida PASA que es el mejor hasta el momento.

Método	Train	Validation	Test
Base	0.915	0.91	0.91
K Continuado	0.92	0.92	0.92
L PASA	0.92	0.92	0.92

Tabla 6.2: *Accuracy Sketches*

Método	clases	Train			Validation			Test		
Base	Aviones	P: 0.99	R: 0.96	F1: 0.97	P: 0.99	R: 0.95	F1: 0.97	P: 0.99	R: 0.96	F1: 0.97
	Buses	P: 0.90	R: 0.81	F1: 0.85	P: 0.90	R: 0.81	F1: 0.85	P: 0.90	R: 0.81	F1: 0.85
	Autos	P: 0.93	R: 0.93	F1: 0.93	P: 0.92	R: 0.93	F1: 0.93	P: 0.93	R: 0.93	F1: 0.93
	Carro bomberos	P: 0.85	R: 0.92	F1: 0.89	P: 0.85	R: 0.92	F1: 0.89	P: 0.85	R: 0.92	F1: 0.88
	Patines	P: 0.94	R: 0.97	F1: 0.95	P: 0.93	R: 0.97	F1: 0.95	P: 0.94	R: 0.96	F1: 0.95
K Continuado	Aviones	P: 0.98	R: 0.97	F1: 0.98	P: 0.98	R: 0.97	F1: 0.98	P: 0.99	R: 0.97	F1: 0.98
	Buses	P: 0.87	R: 0.87	F1: 0.87	P: 0.87	R: 0.87	F1: 0.87	P: 0.86	R: 0.88	F1: 0.87
	Autos	P: 0.94	R: 0.93	F1: 0.93	P: 0.93	R: 0.93	F1: 0.93	P: 0.94	R: 0.93	F1: 0.93
	Carro bomberos	P: 0.89	R: 0.90	F1: 0.90	P: 0.89	R: 0.90	F1: 0.89	P: 0.89	R: 0.90	F1: 0.89
	Patines	P: 0.97	R: 0.96	F1: 0.96	P: 0.96	R: 0.95	F1: 0.96	P: 0.97	R: 0.95	F1: 0.96
L PASA	Aviones	P: 0.97	R: 0.98	F1: 0.98	P: 0.97	R: 0.98	F1: 0.97	P: 0.98	R: 0.98	F1: 0.98
	Buses	P: 0.86	R: 0.87	F1: 0.87	P: 0.86	R: 0.87	F1: 0.87	P: 0.88	R: 0.89	F1: 0.89
	Autos	P: 0.94	R: 0.93	F1: 0.93	P: 0.94	R: 0.93	F1: 0.93	P: 0.91	R: 0.92	F1: 0.92
	Carro bomberos	P: 0.89	R: 0.90	F1: 0.90	P: 0.89	R: 0.89	F1: 0.89	P: 0.88	R: 0.91	F1: 0.89
	Patines	P: 0.96	R: 0.96	F1: 0.96	P: 0.96	R: 0.95	F1: 0.96	P: 0.93	R: 0.94	F1: 0.94

Tabla 6.3: Reporte clasificación Sketches

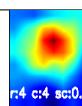
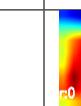
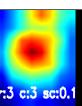
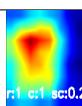
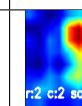
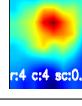
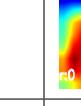
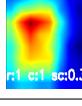
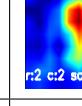
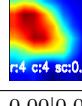
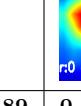
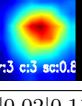
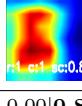
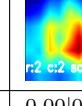
Imágenes					
Nombre	4644317375234048	4905290711433216	4932170596483072	6395809920712704	6613196754386944
Mascaras					
CAM inicial	 r:4 c:4 sc:0.1	 r:0 c:0 sc:0.1	 r:3 c:3 sc:0.1	 r:1 c:1 sc:0.2	 r:2 c:2 sc:0.4
Clasf inicial	0.00 0.00 0.00 0.67 0.33	0.32 0.00 0.00 0.00 0.68	0.00 0.82 0.00 0.17 0.00	0.00 0.21 0.00 0.79 0.00	0.00 0.00 0.43 0.12 0.45
Visualizaciones finales	-	-	-	-	-
K Continuado	 r:4 c:4 sc:0.2	 r:0 c:0 sc:0.8	 r:3 c:3 sc:0.1	 r:1 c:1 sc:0.1	 r:2 c:2 sc:0.7
Clasf final	0.05 0.06 0.00 0.60 0.29	0.91 0.08 0.01 0.00 0.00	0.00 0.81 0.00 0.19 0.00	0.00 0.35 0.00 0.65 0.00	0.00 0.00 0.75 0.15 0.10
L PASA	 r:4 c:4 sc:0.8	 r:0 c:0 sc:0.8	 r:3 c:3 sc:0.8	 r:1 c:1 sc:0.8	 r:2 c:2 sc:0.8
Clasf final	0.00 0.03 0.01 0.07 0.89	0.83 0.08 0.07 0.02 0.00	0.02 0.11 0.07 0.80 0.00	0.00 0.86 0.00 0.14 0.00	0.00 0.00 0.95 0.05 0.00

Tabla 6.4: Visualización Sketches

6.1.3. Análisis

Al observar las tablas de *accuracy* con los reportes de clasificación (tablas 6.2 6.3) se tiene que el entrenamiento continuado logra cierta mejora ínfima respecto al modelo base. El modelo con PASA es prácticamente idéntico, solamente cambia a cuál clase clasifica mejor. Dado esto la mejora es ínfima tal como ocurre en el subconjunto de *IMAGENET*.

En cuanto a clasificación las imágenes seleccionadas (tabla 6.4 filas 5 y 8) no logran mejorar bajo el entrenamiento continuado, mientras que el modelo de PASA logra siempre aumentar la probabilidad de la imagen sin afectar de forma considerable el modelo en validación o prueba.

Respecto a las visualizaciones (tabla 6.4 filas 4, 7 y 9) el área resaltada es evitada consistentemente en el caso de PASA. En el caso del entrenamiento continuado solamente mantiene la tendencia observada.

En general se observan los mismos efectos que el *dataset* de imágenes de animales, es decir no hay mejora en métricas globales, pero logra modificar activación y clasificación en imágenes seleccionadas.

6.2. Experimento Símbolos

Para complementar el estudio del modelo se decide simplificar el problema mediante un nuevo *dataset* especialmente construido para presentar características irrelevantes.

El problema de clasificación se enfoca en clasificar símbolos de latex ε ξ ζ los cuales presentan formas similares. Estos símbolos se generan en una imagen de 72x72 en escala de grises. Los símbolos siempre están en la misma posición, pero cambian de tamaño, orientación y proporción. Además en la esquina inferior izquierda se presentan los símbolos *, / o vacío de forma aleatoria. Se intenta engañar al modelo a aprender que las características de *, son importantes cuando no lo son. Este ejemplo fue inspirado en un problema real donde se requería clasificar símbolos impresos en los cuales aparecían muchas veces líneas punteadas, líneas horizontales u otros símbolos sin importancia. Ejemplos de los datos en la tabla 6.5.

Debido a lo simple del modelo se utiliza una arquitectura algo más pequeña solo 3 bloques convolucionales sin *batch normalization* ni bloques residuales. Se quitó todos estos debido a que de lo contrario alcanzaba prácticamente el 100 % de *accuracy* en entrenamiento.

Se utilizó PASA que en *datasets* anteriores logró cambiar tanto clasificaciones como visualizaciones. Debido a que las imágenes presentan alta probabilidad con la predicción correcta es necesario utilizar valores altos para B , th , se utiliza B de 0.8 , th de 2.

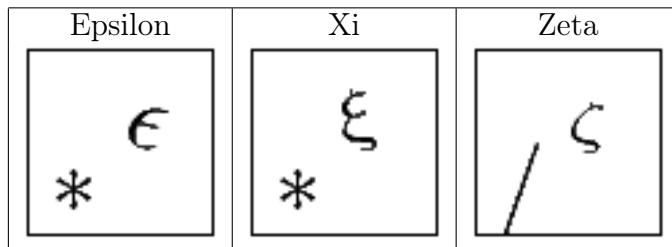


Tabla 6.5: Ejemplo datos Símbolos

6.2.1. Resultados

Método	Train	Validation	Test
Base	0.732	0.73	0.74
M Continuado	0.762	0.76	0.74
N PASA	0.68	0.68	0.69

Tabla 6.6: *Accuracy* símbolos

Método	clases	Train			Validation			Test		
Base	Xi	P: 0.61	R: 0.36	F1: 0.45	P: 0.61	R: 0.36	F1: 0.45	P: 0.70	R: 0.45	F1: 0.55
	Epsilon	P: 0.98	R: 1.00	F1: 0.99	P: 0.98	R: 1.00	F1: 0.99	P: 0.93	R: 0.99	F1: 0.96
	Zeta	P: 0.65	R: 0.83	F1: 0.73	P: 0.65	R: 0.83	F1: 0.73	P: 0.61	R: 0.79	F1: 0.69
M Continuado	Xi	P: 0.65	R: 0.53	F1: 0.58	P: 0.65	R: 0.53	F1: 0.58	P: 0.69	R: 0.55	F1: 0.61
	Epsilon	P: 0.95	R: 1.00	F1: 0.98	P: 0.95	R: 1.00	F1: 0.98	P: 0.89	R: 0.99	F1: 0.94
	Zeta	P: 0.71	R: 0.78	F1: 0.74	P: 0.71	R: 0.78	F1: 0.74	P: 0.63	R: 0.69	F1: 0.66
N PASA	Xi	P: 0.59	R: 0.36	F1: 0.45	P: 0.59	R: 0.36	F1: 0.45	P: 0.62	R: 0.46	F1: 0.53
	Epsilon	P: 0.84	R: 1.00	F1: 0.91	P: 0.84	R: 1.00	F1: 0.91	P: 0.83	R: 0.99	F1: 0.90
	Zeta	P: 0.57	R: 0.68	F1: 0.62	P: 0.57	R: 0.68	F1: 0.62	P: 0.57	R: 0.61	F1: 0.59

Tabla 6.7: Reporte clasificación Símbolos

Imágenes			
Nombre	epsilon_46.png	xi_22.png	zeta_49.png
Mascaras			
CAM inicial			
Clasf inicial	0.01 0.98 0.01	0.60 0.00 0.40	0.55 0.00 0.45
Visualizaciones finales	-	-	-
M Continuado			
Clasf final	0.00 0.99 0.00	0.66 0.00 0.34	0.60 0.00 0.40
N PASA			
Clasf final	0.13 0.69 0.17	0.50 0.01 0.50	0.50 0.01 0.49

Tabla 6.8: Visualizaciones Símbolos

6.2.2. Análisis

Al observar la tabla 6.6 el entrenamiento continuado muestra ligera mejora en validación, pero no se ve traducido a prueba, lo cual es esperado dado los resultados en *datasets* anteriores.

De observar las visualizaciones CAM de las imágenes iniciales (tabla 6.8 filas 4) se aprecia que efectivamente las características irrelevantes presentan activaciones, notar el símbolo / en la clase ζ y el enfoque de * para la clase ε . Estas activaciones se mantienen en el entrenamiento continuado (tabla 6.8 filas 7). En cambio el método PASA logra claramente eliminar la activación (tabla 6.8 fila 9). Otro detalle notable, es que el CAM está prácticamente realizando la función de un detector de bordes, esto ocurre debido a que los filtros son tan simples que su activación se centra en áreas muy finas (en contraste a las extensas activaciones en los demás *datasets*).

La métrica *accuracy* baja en las imágenes seleccionadas lo cual da cuenta que el modelo no es lo suficiente complejo como para generar un filtro que no se active en las áreas seleccionadas y que a su vez genere alta activación en áreas importantes. De hecho, se puede notar que no logra borrar del todo su activación en patrones como el asterisco probablemente por su similitud con las ondulaciones de los símbolos.

Se intentan diversas variaciones de valores de B , th , pero ocurre que o se disminuye tanto los parámetros que ya no hay cambios en activaciones o al ser los parámetros muy grandes la disminución de las activaciones evita la mejora en las clasificaciones. Esto refuerza la idea que el modelo no sea lo suficientemente complejo para lograr filtros que ignoren la característica espuria y se activen en áreas relevantes.

En resumen, en este *dataset* el modelo empeora el clasificador al no lograr disminuir las activaciones y clasificar a la vez.

6.3. Experimento rayos X

Este último *dataset* trata sobre detección de elementos peligrosos en aeropuertos. Se tienen 4 clases Pistola, Navaja, Shuriken y Otros. La clase Otros representa diversos objetos pequeños que también se mezclan con las otras 3 clases principales. La cantidad de puntos en los diversos conjuntos de entrenamiento, validación, prueba son 700, 300, 900 respectivamente. Ejemplos de las imágenes de este *dataset* en tabla 6.9.

Se selecciona este *dataset* debido a que tiene una característica en particular. Todos los datos de entrenamiento son la misma imagen rotada con diversos objetos que la acompañan, mientras que en las imágenes de prueba aparecen solamente el objeto sin elementos mezclados. Esto podría parecer un caso más fácil, pero las imágenes de la clase Otros muchas veces contenían grandes cantidades de espacio en blanco. Esto produce confusión entre las imágenes de prueba y las imágenes de la clase Otros. Este efecto ocurre de forma considerable para la clase navaja que deja de por sí bastantes áreas en blanco debido a su tamaño.

La estrategia para conseguir las máscaras consiste en seleccionar unas cuantas imágenes ejemplares y seleccionar varias imágenes de la clase Otros con grandes áreas en blanco. Se utilizan 14 imágenes con áreas seleccionadas.

Dado que las imágenes en entrenamiento son muy similares a las imágenes en los datos de prueba, la probabilidad de predicción en entrenamiento es prácticamente 1 para todas las

Método	Train	Validation	Test
Base	1.0	1.00	0.73
L Continuado	1.0	1.00	0.70
O PASA	0.998	0.99	0.82

Tabla 6.10: *Accuracy* Xray

imágenes. Debido a esto es necesario utilizar valores de B , th altos se utiliza 5 y 0.8.

6.3.1. Resultados

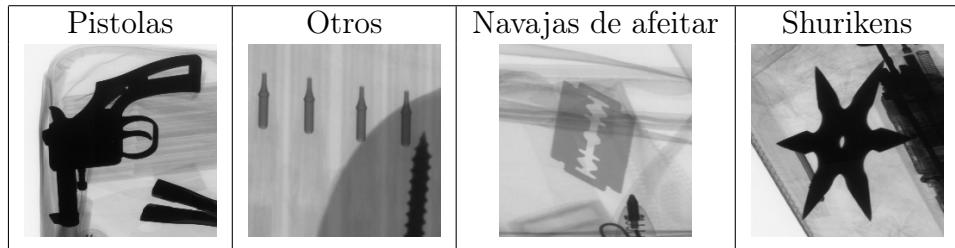


Tabla 6.9: Ejemplo dataset Xray

Método	clases	Train			Validation			Test		
Base	Pistola	P: 1.00	R: 1.00	F1: 1.00	P: 1.00	R: 1.00	F1: 1.00	P: 0.84	R: 0.53	F1: 0.65
	Otros	P: 1.00	R: 1.00	F1: 1.00	P: 1.00	R: 1.00	F1: 1.00	P: 0.78	R: 0.93	F1: 0.85
	Navaja	P: 1.00	R: 1.00	F1: 1.00	P: 1.00	R: 1.00	F1: 1.00	P: 0.36	R: 0.12	F1: 0.18
	Shuriken	P: 1.00	R: 1.00	F1: 1.00	P: 1.00	R: 1.00	F1: 1.00	P: 0.53	R: 0.77	F1: 0.63
L Continuado	Pistola	P: 1.00	R: 1.00	F1: 1.00	P: 1.00	R: 1.00	F1: 1.00	P: 0.68	R: 0.53	F1: 0.60
	Otros	P: 1.00	R: 1.00	F1: 1.00	P: 1.00	R: 1.00	F1: 1.00	P: 0.79	R: 0.87	F1: 0.83
	Navaja	P: 1.00	R: 1.00	F1: 1.00	P: 1.00	R: 1.00	F1: 1.00	P: 0.35	R: 0.11	F1: 0.17
	Shuriken	P: 1.00	R: 1.00	F1: 1.00	P: 1.00	R: 1.00	F1: 1.00	P: 0.48	R: 0.79	F1: 0.60
O PASA	Pistola	P: 1.00	R: 1.00	F1: 1.00	P: 1.00	R: 1.00	F1: 1.00	P: 0.58	R: 0.88	F1: 0.69
	Otros	P: 1.00	R: 1.00	F1: 1.00	P: 0.99	R: 0.99	F1: 0.99	P: 0.96	R: 0.83	F1: 0.89
	Navaja	P: 1.00	R: 0.99	F1: 0.99	P: 0.98	R: 0.98	F1: 0.98	P: 0.93	R: 0.96	F1: 0.95
	Shuriken	P: 1.00	R: 1.00	F1: 1.00	P: 1.00	R: 1.00	F1: 1.00	P: 0.82	R: 0.47	F1: 0.60

Tabla 6.11: Reporte clasificación Xray

6.3.2. Análisis

Respecto a los resultados de *accuracy* (tabla 6.10) el método propuesto produce una baja ínfima en entrenamiento, validación, pero una mejora considerable en el conjunto de prueba.

El principal problema del clasificador base es la clase Navaja, la cual se confunde con la clase Otros en el conjunto de prueba. Esto se aprecia en el resumen de clasificación (tabla

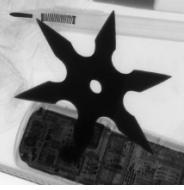
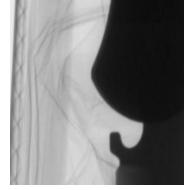
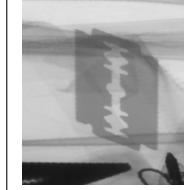
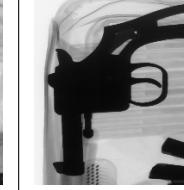
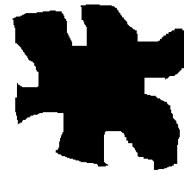
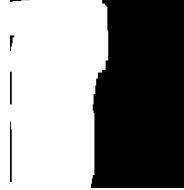
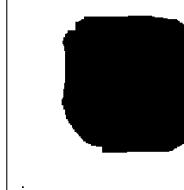
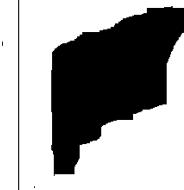
Imágenes				
Nombre	train_16	train_479	train_513	train_632
Mascaras				
CAM inicial	 r3 c3 sc1.00	 r1 c1 sc1.00	 r2 c2 sc1.00	 r0 c0 sc1.00
Clasf inicial	0.00 0.00 0.00 1.00	0.00 1.00 0.00 0.00	0.00 0.00 1.00 0.00	1.00 0.00 0.00 0.00
Visualizaciones finales	-	-	-	-
L Continuado	 r3 c3 sc1.00	 r1 c1 sc1.00	 r2 c2 sc1.00	 r0 c0 sc1.00
Clasf final	0.00 0.00 0.00 1.00	0.00 1.00 0.00 0.00	0.00 0.00 1.00 0.00	1.00 0.00 0.00 0.00
O PASA	 r3 c3 sc0.99	 r1 c1 sc1.00	 r2 c2 sc1.00	 r0 c0 sc0.99
Clasf final	0.00 0.01 0.00 0.99	0.00 1.00 0.00 0.00	0.00 0.00 1.00 0.00	0.99 0.00 0.00 0.00

Tabla 6.12: Visualizaciones xray

6.11), donde el método PASA logra incremento considerable para la métrica F1 (de 0.18 a 0.95). Dada las demás clases también hay ciertas mejoras, aunque también hay cierta disminución en métricas para la clase Shuriken.

Respecto a las activaciones (tabla 6.12), la mayoría desde un inicio estaba correcta solamente se marcan para evitar cambios en el proceso de reajuste. La segunda imagen es uno de los tantos objetos de la clase Otros que muestran grandes trozos de espacio en blanco, al seleccionar el espacio en blanco se puede apreciar que la función logra apagar su activación. Respecto a probabilidad de predicción todas parten altas y se mantiene altas, aunque estrictamente el método produce una disminución de la probabilidad inicial.

Finalmente existe una mejora en métricas globales utilizando el conocimiento experto. Aun así, este es un caso muy simplificado, ya que en un caso general puede que no se tenga este conocimiento del patrón a eliminar.

6.4. Resumen

Los resultados de 3 *dataset* distintos entregan conclusiones similares a los resultados obtenidos en el *dataset IMAGENET*. La visualización logra modificarse, pero sin mayores cambios en las métricas de clasificación. Una excepción es el *dataset X-RAY* donde se conocían las características que producían problemas en el modelo, dado esto se logró modificar el clasificador generando una mejora de 10% en métricas de clasificación. Por lo tanto, el modelo tiene cierta aplicación en casos acotados para el resto de los casos parece existir problemas para mejorar el clasificador.

Capítulo 7

Discusión de resultados

Dado los resultados observados es claro que no se alcanzaron las expectativas y que el método propuesto presenta dificultades para mejorar la generalización del clasificador ya entrenado. Para dilucidar que variables son las responsables de tales resultados, se realiza una serie de experimentos sobre los resultados obtenidos en el *dataset* de animales.

7.1. Exploración de los filtros de forma global

Para analizar los resultados y buscar donde los supuestos sobre el modelo fallan se analiza las características que detecta el modelo. Como se mencionó en la sección marco teórico, entender las características que residen en los filtros y su relación con el clasificador es un área de constante investigación [30]. No hay una sola metodología que funcione en el caso general. Afortunadamente en este caso la arquitectura es un clasificador lineal de los promedios de las características convolucionales de la última capa, por lo que es posible simplificar el análisis a buscar el significado de cada filtro de la última capa de forma separada y considerar su peso asociado como la importancia global. Se centra el análisis en el *dataset* subconjunto de *IMAGENET* con su clasificador entrenado. También se considera que el modelo está fijo excepto por la última capa, es decir, ya no es una red convolucional, sino que un vector de entrada 64 (64 filtros en la última capa que se promedian por filtro dando un vector de 64 dimensiones) y una capa de clasificación con 4 categorías.

Antes de comenzar el análisis es necesario determinar objetivamente las características irrelevantes. Hasta este punto se había dejado el concepto a interpretación del experto, sin embargo, ahora que es necesario un análisis de los componentes del modelo es necesario responder ¿Cuándo un filtro es irrelevante? De lo contrario es imposible contrastar la decisión del experto. En este caso el análisis se basa en la literatura de selección de variables. En sí es difícil llegar a una definición robusta de la utilidad de una característica en singular, ya que depende del modelo y su relación con otras características. Por ejemplo, muchos métodos de selección de características analizan la contribución de una variable aislada del resto, lo cual no es siempre lo indicado, ya que no se factoriza la posible relación que esta pueda tener con otras variables. Quizás la variable X no tenía tanta importancia por si sola, pero

al combinarse con la variable Y generaba una buena característica. Por el otro lado, una forma más robusta es analizar subconjuntos de variables seleccionando los que tengan mayor *accuracy* para un modelo fijo, lamentablemente para esto habría que entrenar 64! modelos lo cual hace infactible tal análisis.

Dado lo anterior se trata de mezclar ambas estrategias al mantener varias métricas. Se espera que al agregar este conjunto, se logre determinar si es una característica irrelevante o no. Se utilizó información mutua [47] [2], prominencia de característica según arboles de decisión *extratrees* [4], [15], un clasificador de una variable. También se utiliza un clasificador lineal entrenado con regularización L1 para comparar, regularización L1 incentiva al clasificador a usar solamente algunas de las dimensiones en el clasificador.

7.1.1. ¿Existen características irrelevantes?

Dado las 3 métricas descritas se ponderan las 3 métricas de forma igualitaria y se genera un ordenamiento. Esto permite ordenar las características de la más importante a la menos importante.

Para probar que efectivamente este orden refleje la importancia de una característica, se generan clasificadores agregando las variables de forma incremental desde la más relevante a la menos relevante para apreciar cómo afecta el desempeño del modelo. Se itera desde agregar 1 variable a agregar las 64 variables, reajustando en cada subconjunto con las variables restantes. Este reajuste corresponde a tomar la totalidad de vectores en entrenamiento, tomar las X dimensiones seleccionadas de estos puntos y reajustar la salida lineal para el problema de clasificación de las 4 categorías. La evaluación es siempre con el conjunto de validación y se calcula *accuracy* promedio para las 4 clases. Se gráfica también el valor de *accuracy* del modelo inicial para tener una referencia de cuando el modelo reducido resulta ser equivalente al modelo de las 64 características. Se produce el gráfico de la figura 7.1 .

Se observa que realmente existen filtros irrelevantes y son una cantidad considerable. Estos son irrelevantes, ya que es posible con una cantidad mucho menor al total de filtros generar un modelo de similar *accuracy* o de forma equivalente es posible quitar las características de menor relevancia y el clasificador no sufre cambios (una vez reajustado claro está). De los 64 filtros se pueden quitar cómodamente entre 30 a 40 filtros sin mayor pérdida de *accuracy* o incluso ganancia.

El punto de corte indica que con solo el 40 % de los filtros originales o 25 características es posible mantener la *accuracy* del 0,76.

El clasificador entrenado con pérdida L1 también concuerda con estos resultados, ya que mantiene una *accuracy* de 0,76 y utiliza solamente 42 filtros para su predicción.

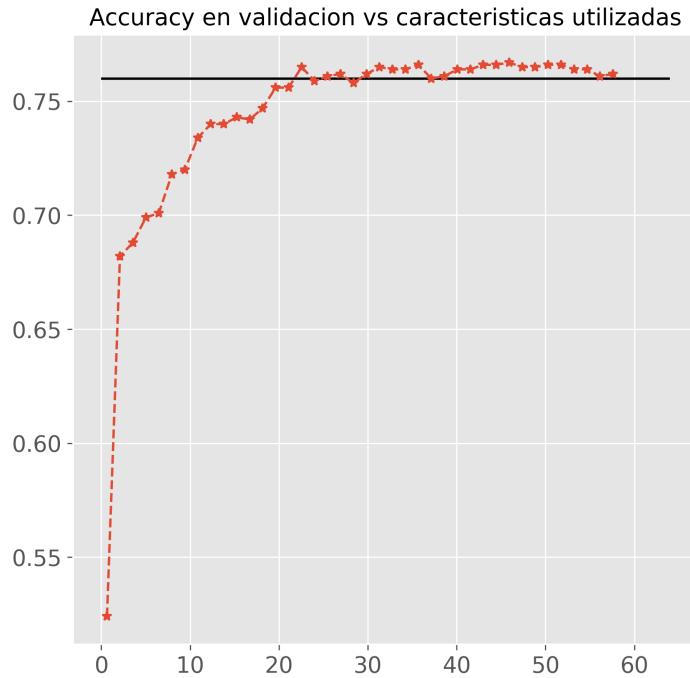


Figura 7.1: *Accuracy* al agregar variables según ranking de selección de variables

7.1.2. ¿La inspección visual global encuentra las características irrelevantes?

Una vez se conocen que existen características innecesarias es necesario verificar si en el *dataset*/clasificador utilizado es posible indicar visualmente cuando una característica es irrelevante, lo cual es parte de la hipótesis inicial. Para probar la veracidad de esta se compara la lista de características irrelevantes a partir del ordenamiento de la sección anterior con una lista generada por un usuario experto.

Para transformar el ranking de características en una lista de categorías relevantes/irrelevante se utiliza el corte de 25 filtros mencionado en la sección anterior. Por lo tanto, hay 25 características relevantes y 39 características irrelevantes.

Para decidir el experto observa una serie de activaciones similar al trabajo de Yosinski et al. [52], tal visualización se genera de la siguiente forma. Primero se elige alguno de los 64 filtros, dado un filtro se procesa todas las imágenes de entrenamiento anotando y organizando las activaciones de ese filtro, se generan 9 rangos equiespaciados entre la mínima y máxima activación, se eligen 9 puntos al azar en cada intervalo y se dibujan. Por último, cada imagen dibujada es filtrada para solamente mostrar las partes de mayor activación en el filtro correspondiente, para esto se aplica OTSU (algoritmo de binarización automática) sobre el filtro para obtener una máscara binaria que se aplica sobre la imagen. Esta visualización busca representar qué tipo de imágenes se activan dado un filtro. Un ejemplo de tal visualización



Figura 7.2: Ejemplo de visualización global por filtro. Característica 6 al parecer se enfoca en hocicos de lobos, zorros blancos

se observa en la figura 7.2.

El criterio utilizado por el experto para determinar si una característica era irrelevante es revisar si los retazos filtrados contenían partes considerables del animal a clasificar.

Al realizar este análisis el experto determinó diversos filtros como irrelevantes. Algunos con alta confianza como lo son los filtros 26 o 22 (observar tabla 7.1). En estos se observan diversos trozos de ambiente que ni siquiera son un patrón consistente.

Pero a su vez hay otros en los cuales no es clara la distinción. Por ejemplo, el filtro 29 (observar figura 7.3) donde por un lado selecciona al animal, pero por otro parte la selección es muy grande y ruidosa.

Filtro 22	Filtro 26		
VAL 1.64 - 0	VAL NO VALUES - 1	VAL 1.23 - 2	
VAL 1.04 - 3	VAL 0.85 - 4	VAL 0.65 - 5	
VAL 0.48 - 6	VAL 0.28 - 7	VAL 0.12 - 8	

Tabla 7.1: Dos filtros que muestran patrones que el experto selecciona como no relevantes

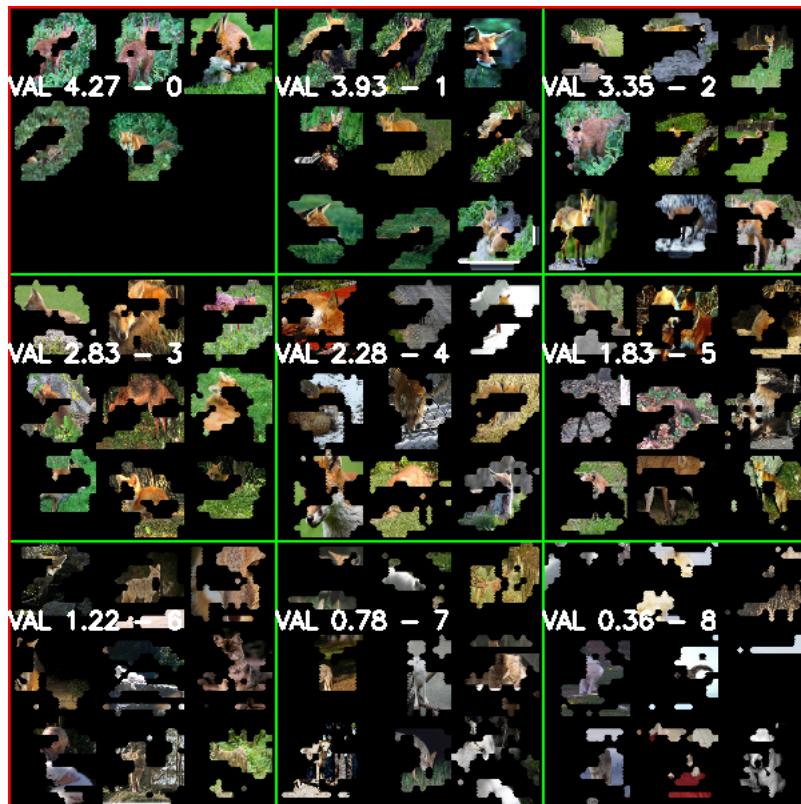


Figura 7.3: Filtro 29 el cual es difícil de clasificar como relevante o no relevante

Una vez con la categorización de relevantes/irrelevantes dada por el experto, se contrasta con los resultados de selección de características. Se compara asumiendo las categorías del

ordenamiento de características como las categorías correctas y las categorías del experto como la predicción.

	precision	recall	f1-score	support
Irrelevante	0.82	0.59	0.69	39
Relevante	0.56	0.80	0.66	25
avg / total	0.72	0.67	0.67	64

Figura 7.4: Resultados de clasificación según categorización dada por experto respecto a relevante según ranking

Los resultados se encuentran en la figura 7.4. El experto seleccionó 36 características como relevantes y 28 como no relevantes.

Estos resultados indican que el análisis visual tiene cierto mérito, ya que en primer lugar existe alta precisión en la selección de características no relevantes (es decir que la mayoría de las veces que se predice no relevante efectivamente es irrelevante), la recuperación de 0.6 se explica en que la cantidad de características no relevantes es aún mayor de lo que visualmente se aprecia (la cantidad de elementos seleccionados fue 28 mientras que los métodos de selección indican 39 por lo tanto el porcentaje de recuperación es bajo sobre el total). Debido a esto mismo la cantidad de características relevantes es poco precisa, estos resultados se alinean con la intuición inicial de que es fácil determinar características no relevantes, pero difícil identificar las relevantes.

7.1.3. ¿La visualización de CAM apunta a características no relevantes?

Hasta ahora las hipótesis iniciales se mantienen, pero el punto fundamental es si la visualización utilizada apunta hacia las características no relevantes.

Para probar esto se observan las imágenes seleccionadas en los primeros experimentos, para analizar si efectivamente las áreas irrelevantes observadas apuntan a características no relevantes.

Cuando se realizó la selección de imágenes se comenzó por las imágenes mal clasificadas que presentaran alta activación en áreas no relevantes. Es decir, se asumió que las visualizaciones locales que indicaban alta activación en áreas sin importancia indicarían altas activaciones en varios filtros irrelevantes.

Se procede a graficar las 10 activaciones más altas de las imágenes seleccionadas filtrando en las áreas seleccionadas. Se busca encontrar si los filtros que se activan mayormente en esas áreas pertenecen principalmente a los filtros relevantes o irrelevantess.

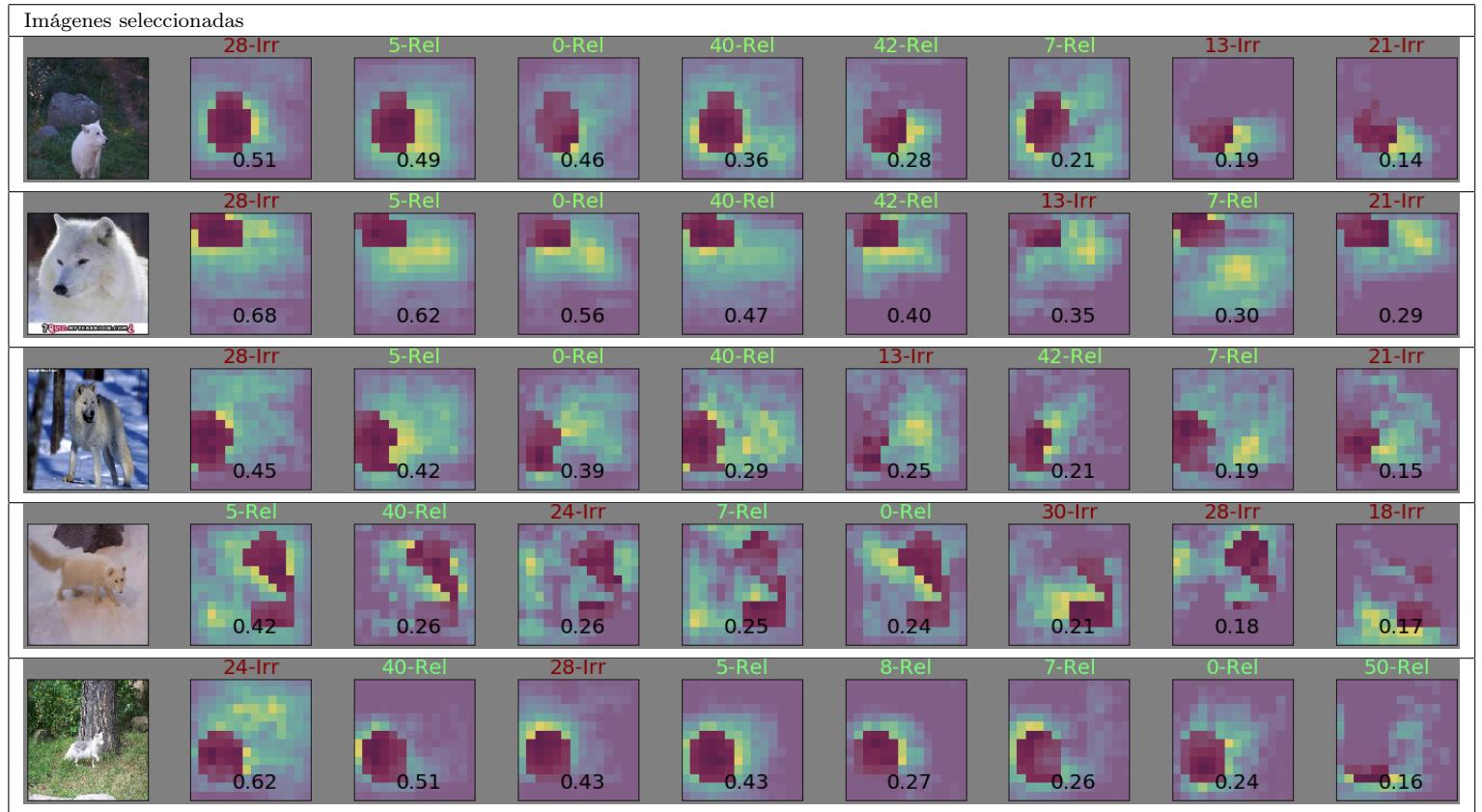


Tabla 7.2: Imágenes con sus mayores activaciones. En morado selección irrelevante, ordenadas según valor en área seleccionada.

Se encuentran proporciones casi iguales de relevantes como irrelevantes. Lo que indica que, si bien es factible encontrar de forma visual características irrelevantes en un análisis global, la metodología utilizando CAM no necesariamente apunta a características no relevantes.

Esto se explica en diversos factores, para empezar cuando se selecciona un área se hace tal selección sobre un agregado de filtros realmente no existe un filtro culpable sino ponderaciones de estos. Por otra parte, una sola activación no es representativa de todos los tipos de áreas que selecciona un filtro. Al parecer los filtros presentan diversas formas que no pueden apreciarse al solo observar una activación.

Si bien estos resultados indican que la visualización utilizada no se enfoca necesariamente en características irrelevantes esto no impide que quizás puede generar efectos positivos en las características irrelevantes que si se enfoca.

7.1.4. Efectos después de reajustar el clasificador

Queda por analizar cómo cambian tanto los filtros relevantes como irrelevantes en el modelo propuesto. Para los experimentos que siguen se usa PASA.

Primero se contabilizan la cantidad de características irrelevantes después del proceso.

Se observa el gráfico de características no relevantes según un nuevo ranking de características (figura 7.5). Estas son las nuevas características irrelevantes que resultan después de aplicar PASA.

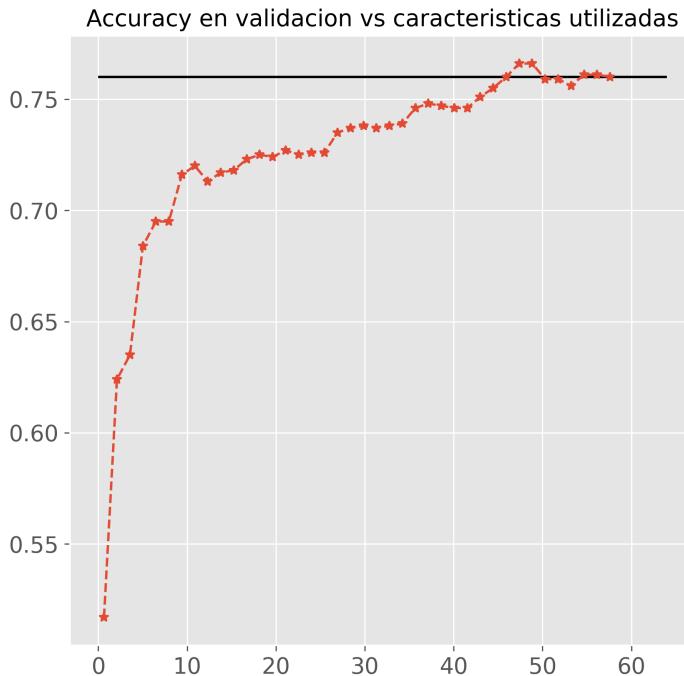


Figura 7.5: *Accuracy* en conjunto de validación dado cantidad de características

En este caso el corte ocurre en 43 características que son el 67% de las características totales. Al parecer las características principales son de menor calidad que en el modelo anterior ya que se requiere una mucho mayor cantidad para alcanzar un porcentaje de *accuracy* cercano al 70 %.

También se observa la tendencia general de las activaciones mediante los promedios por características. Se calculan los promedios antes y después para conseguir la diferencia. Graficando estas diferencias para las 64 características se consigue la tabla 7.3.

Se observa una tendencia general a bajar la activación, con ciertas excepciones.

Se analiza en mayor detalles las 2 características de mayor crecimiento y las 2 características de mayor decrecimiento.

Las de mayor crecimiento son las características 34, 19. Estas características enfocan principalmente a coyotes y zorros. En la figura 7.6 esquina inferior izquierda aparecen comparaciones de imágenes globales antes y después, en estas se aprecia que no hay un cambio considerable parece más un reordenamiento. El histograma indica que ahora la clase coyote tiene mayores activaciones al desplazarse su histograma. En la parte superior se observa

-0.194	-0.116	-0.215	-0.101	-0.185	-0.070	0.043	-0.226
-0.067	-0.079	0.156	0.094	0.930	-0.190	0.543	0.132
-0.022	-0.039	0.154	1.007	-0.023	-0.081	-0.020	0.122
0.645	0.123	-0.015	0.079	-0.115	0.347	-0.667	-0.066
-0.061	-0.092	1.096	0.977	0.099	-0.046	0.551	0.312
0.064	-0.085	0.128	0.362	-0.077	-0.001	0.413	-0.001
-0.015	0.384	0.038	0.072	-0.108	0.410	-0.041	0.566
0.063	0.121	-0.077	-0.105	-0.008	-0.120	-0.067	0.008

Tabla 7.3: Diferencias de promedios globales por característica antes y después de aplicar el método

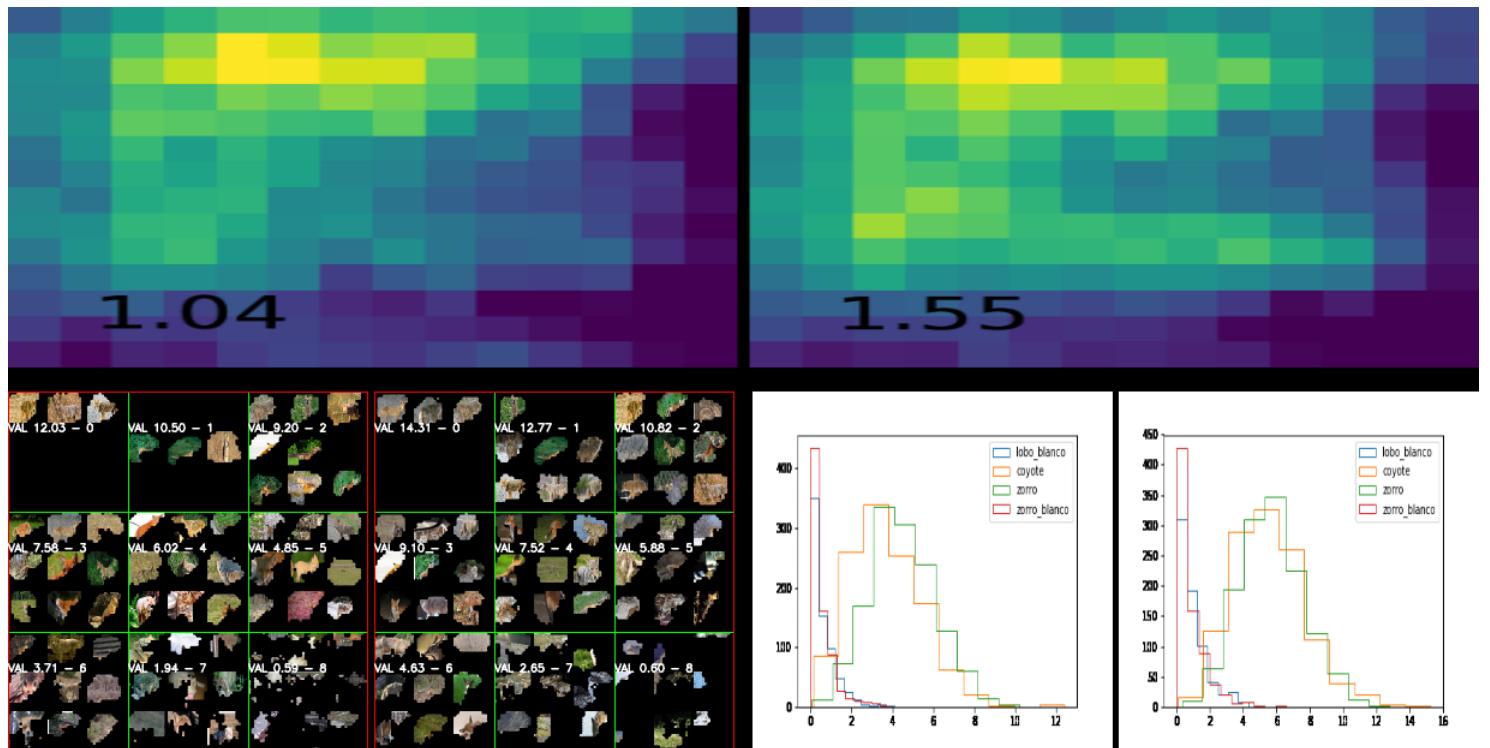


Figura 7.6: Resumen cambios característica 34. Primera fila muestra comparación activaciones imágenes seleccionadas, segunda fila cambios en activaciones globales e histogramas.

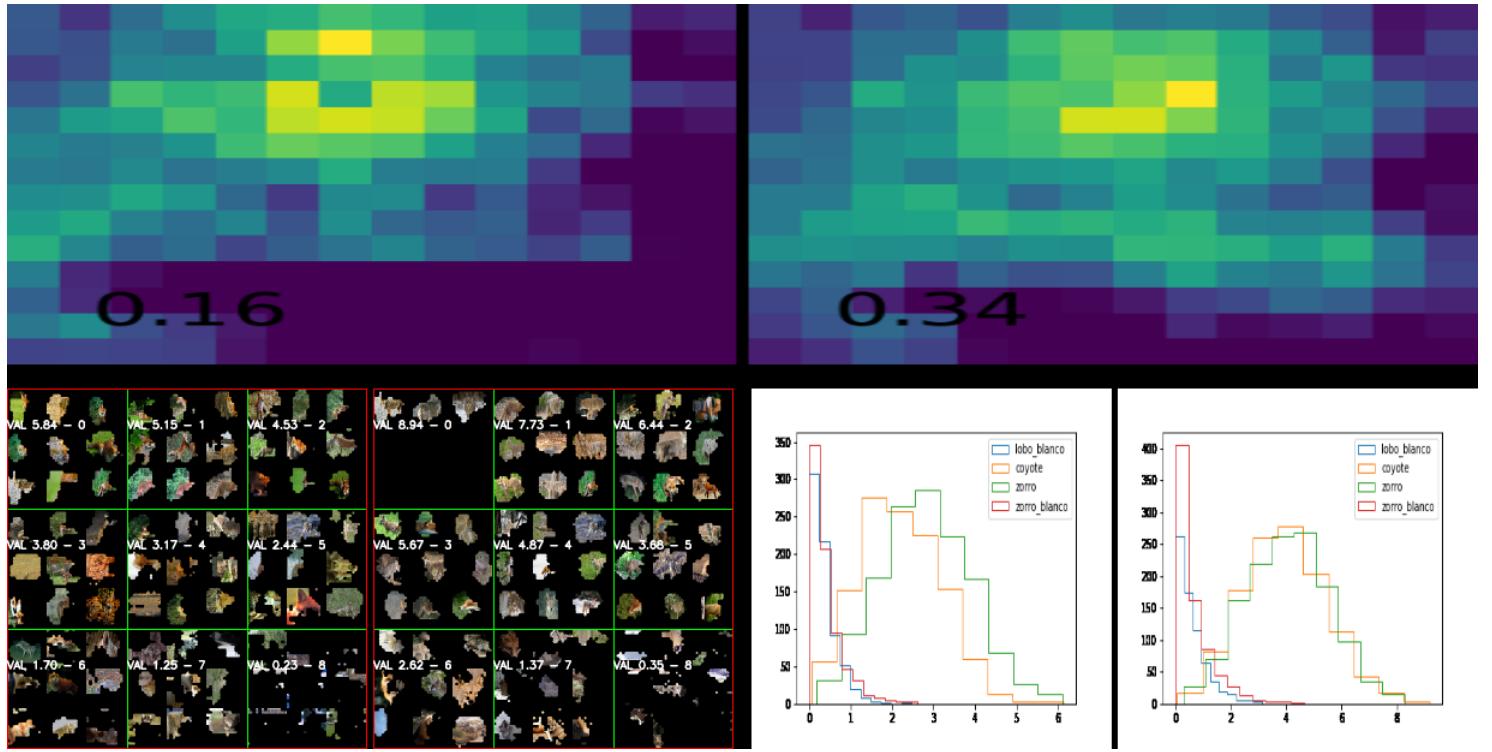


Figura 7.7: Resumen cambios característica 19

una activación de ejemplo antes y después para demostrar el incremento. En la figura 7.7 es prácticamente lo mismo, pero el cambio de zorros a coyotes es más notorio basta mirar las mayores activaciones.

Las de mayor decrecimiento son las características 7, 30. La característica 30 (observar 7.8) parece enfocada en coyotes, antes del cambio parece más enfocada a coyotes en general y después del cambio se enfoca en coyotes en la nieve. Se gráfica también la activación de uno de los zorros blancos del experimento PASA la cual tenía una alta activación en esa característica antes del cambio (en morado área seleccionada, en rojo promedio áreas seleccionadas) notar como la activación prácticamente desaparece. Esta tendencia al parecer es global notar como los histogramas que no son coyotes se desplazan a la izquierda bajando su activación. La característica 7.9 es difícil de describir parece zorros blancos en el pasto, no se aprecian grandes cambios pero el histograma muestra que los lobos blancos ahora son más prevalentes en los intervalos medios. Esta característica en las imágenes seleccionadas se relacionaba con una alta activación a la nieve, pero como se observa en las visualizaciones no es tan simple a nivel global caracterizar el filtro.

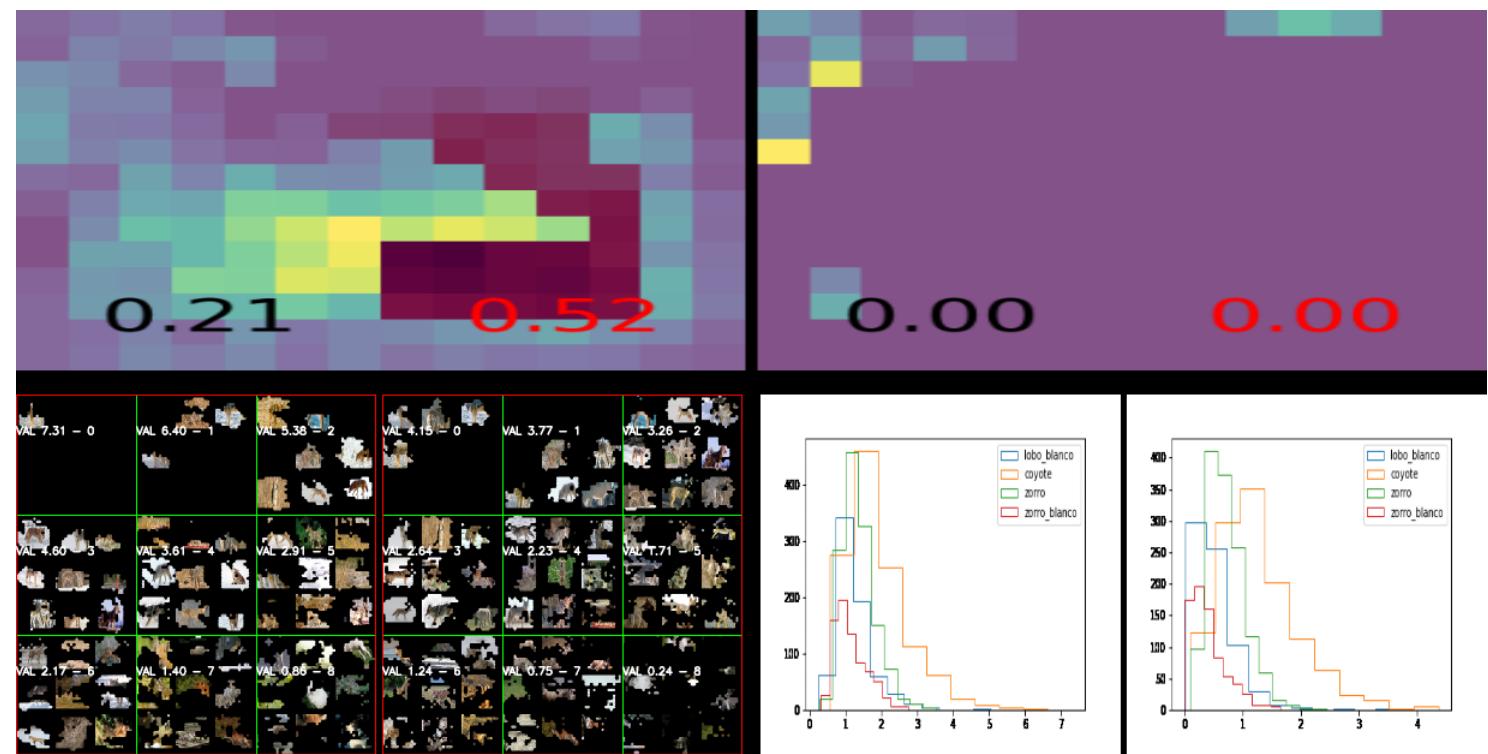


Figura 7.8: Resumen cambios característica 30

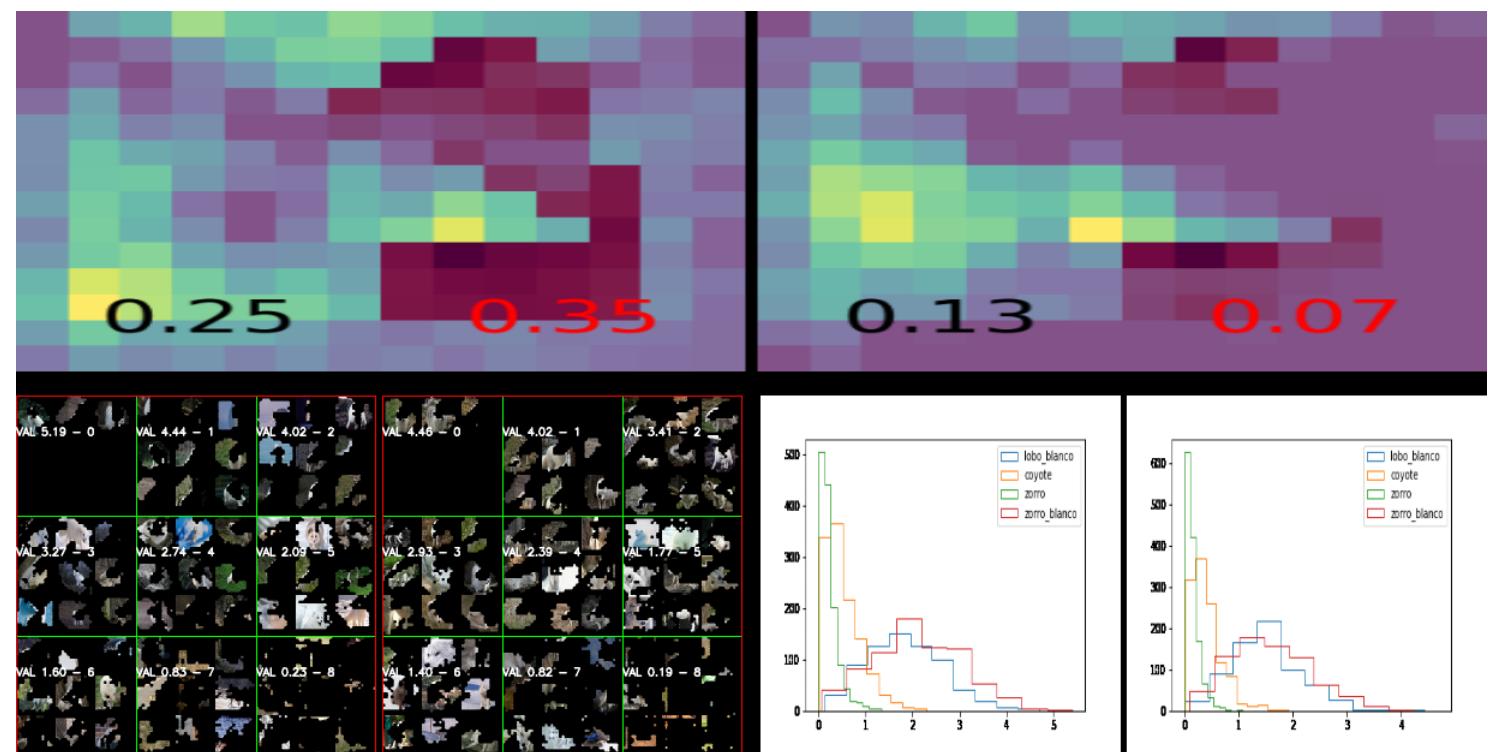


Figura 7.9: Resumen cambios característica 7

Todas estas imágenes ejemplifican la dificultad de relacionar los cambios de activación con los cambios globales. Ya que si bien aparecen claros cambios en una visualización específica

el cambio a nivel global es difícilmente perceptible.

Si bien las características con mayores incrementos son características relacionadas con coyotes/zorros no se mantiene esta tendencia en el resto. Tampoco hay algún patrón claro al analizar relación entre las características de mayor o menor activación del experimento con las que caen o suben (figura 7.10). Lo que está claro es que suben tanto irrelevante como relevantes, bajan tanto irrelevante como relevantes.

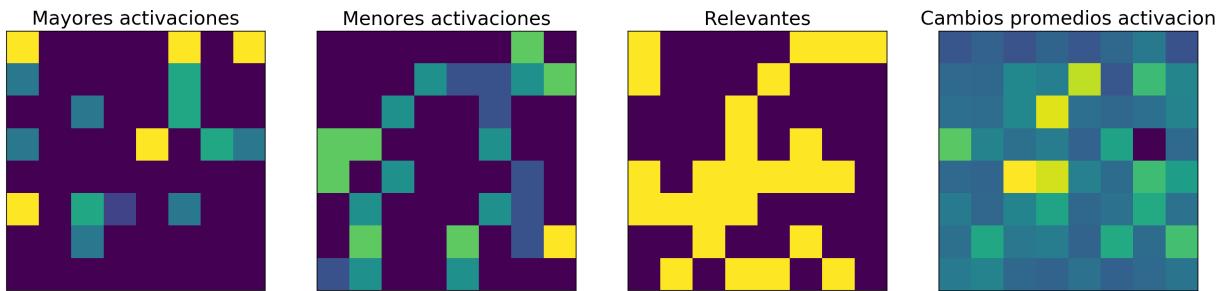


Figura 7.10: Características en diversos criterios

7.1.5. Efectos al reajustar enfocado a una característica irrelevante

Dado que no se logra conseguir selecciones de filtros irrelevantes al utilizar el CAM queda la interrogante de que ocurre si se selecciona específicamente activaciones de filtros irrelevantes. Se plantea la posibilidad de conseguir las selecciones dado las visualizaciones globales de las secciones anteriores, de esta forma se consigue una lista de máscaras nueva centrada específicamente en características irrelevantes. Luego de analizar la lista de características irrelevantes se decide utilizar la característica 39 (figure 7.11).

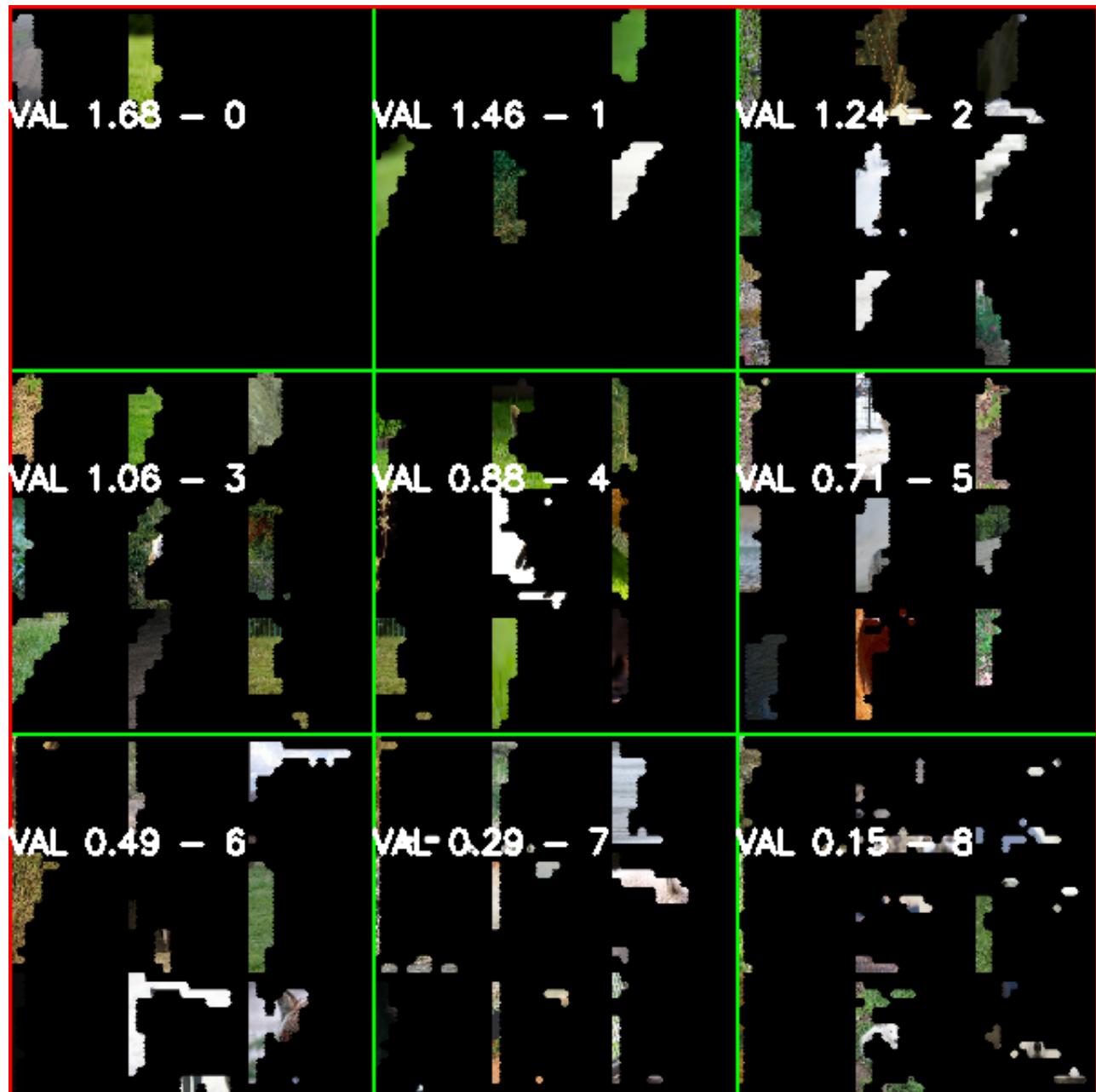


Figura 7.11: Filtro 39 irrelevante que apunta hacia pasto verde

Se selecciona la característica 39 ya que se enfoca claramente en pedazos del fondo y ni siquiera es consistente en qué tipo de terreno. Por lo tanto, es una clara característica irrelevante. De esta característica se usan solamente las activaciones arriba del percentil 70 y se crean máscaras con su binarización. Luego dado este nuevo conjunto de máscaras se reajusta con PASA.

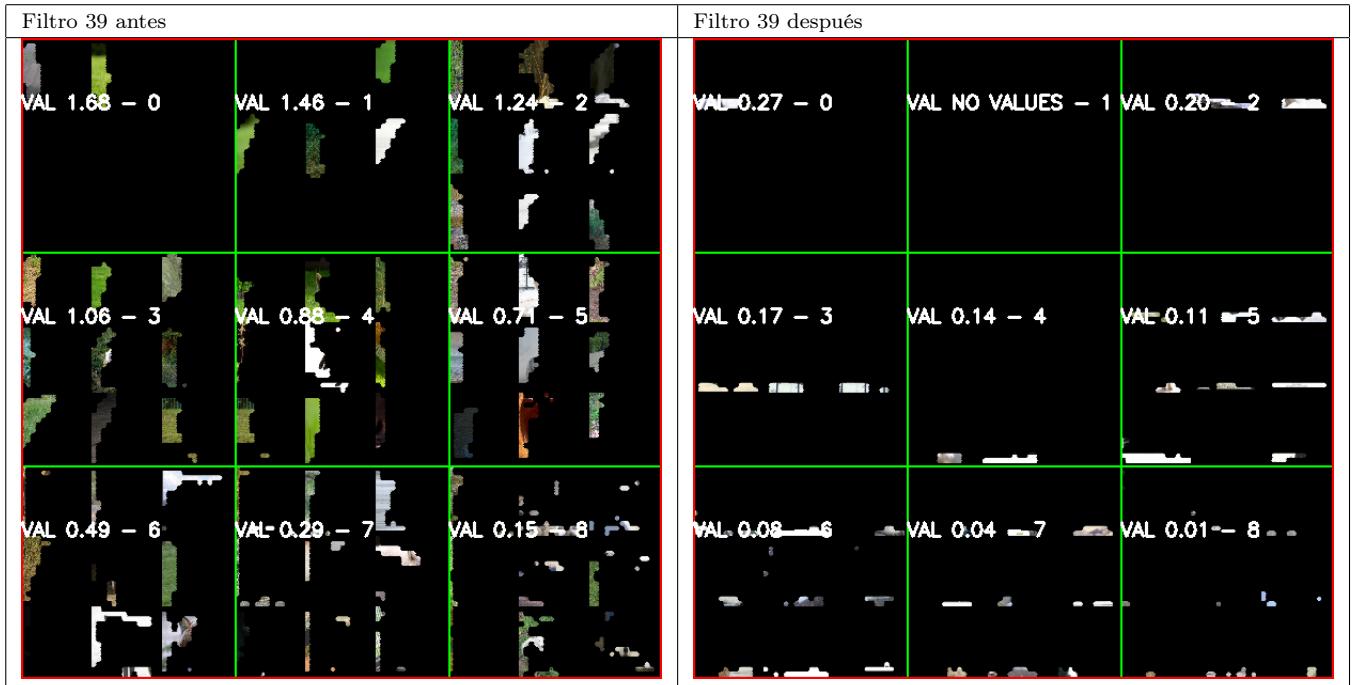


Tabla 7.4: Dos filtros que muestran patrones que el experto selecciona como no relevantes

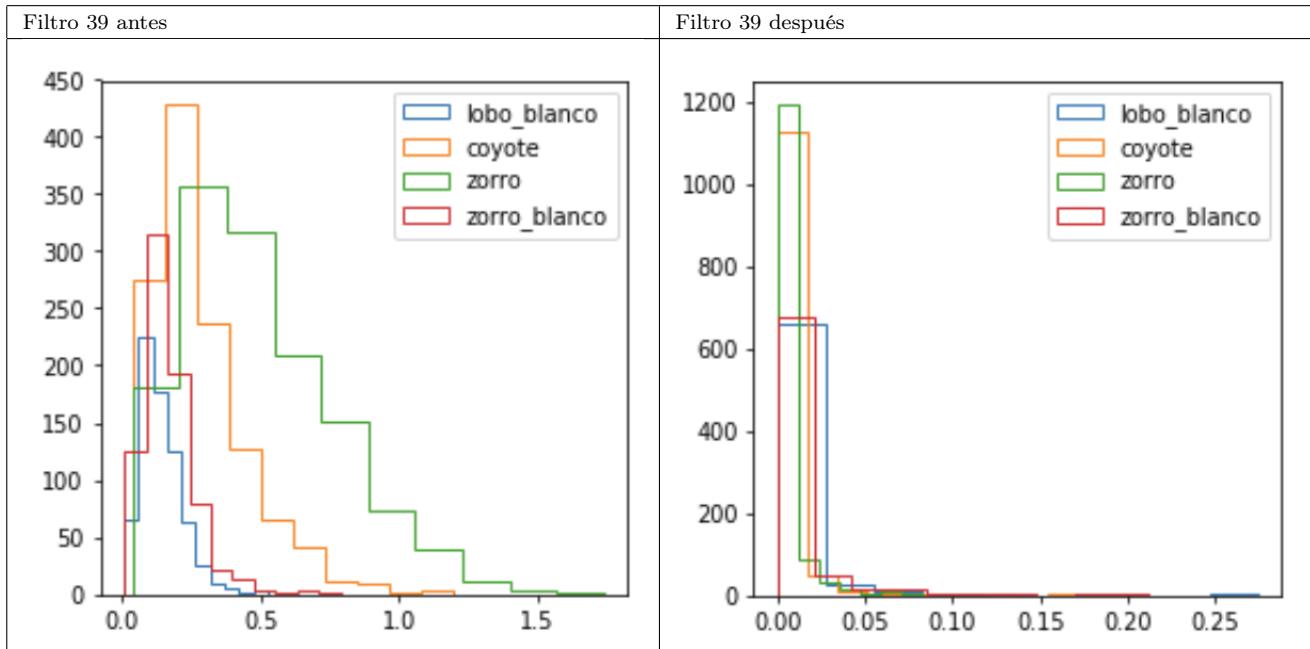


Tabla 7.5: Dos filtros que muestran patrones que el experto selecciona como no relevantes

Los resultados se aprecian en la tabla 7.5. Finalmente se logran cambios apreciables de forma global en una característica, pero estos cambios no son los deseados. Se observa cómo se apaga completamente el filtro y no emergen nuevos patrones, sino que simplemente filtro deja de activarse. Esto indica que el método sigue teniendo dificultades para inducir al clasificador a aprender un nuevo patrón.

7.2. Resumen

Dado que los resultados en capítulos anteriores no fueron los esperados, se hace necesario esclarecer el cómo se relacionan las visualizaciones con las características irrelevantes. Con tal objetivo, primero se determina si existen características irrelevantes en el clasificador estudiado. Los resultados indican que solamente con 25 de 64 características se puede mantener el mismo nivel de *accuracy* que el clasificador completo, por lo tanto existen características redundantes. Las visualizaciones de activaciones determinan que un experto mediante inspección visual logra detectar las características redundantes, lo cual se alinea con las suposiciones iniciales. Pero a pesar de esto la visualización utilizada no es un buen indicador de la relevancia o irrelevancia, dado los resultados de la sección 7.1.3. Por último experimentos sobre una característica irrelevante aislada, indican que el método propuesto puede eliminar una característica, pero falla en reajustar una nueva característica en su lugar.

Conclusión

Considerando los objetivos planteados en este trabajo la implementación de los métodos se logra de forma satisfactoria. Es decir, se desarrolla un método capaz de modificar activaciones en imágenes de forma arbitraria sin perturbar la clasificación en las imágenes seleccionadas. Sin embargo, este método no alcanza las mejoras previstas en métricas de clasificación en el conjunto de prueba. Principalmente debido a que la relación entre las visualizaciones y las características irrelevantes no cumple las suposiciones iniciales.

El método tiene utilidad cuando se conocen las regiones que inducen error en el conjunto de prueba, ya que permite eliminar su activación sin requerir de un nuevo modelo o datos. Esto se evidencia en el caso de los rayos X donde hay mejoras de 10 puntos porcentuales en *accuracy*, pero lamentablemente en la mayoría de los casos no se logró encontrar tales áreas mediante la visualización utilizada. En un inicio se pensó que el modelo contenía diversas características irrelevantes o erróneas que lograban un buen ajuste en el conjunto de entrenamiento, pero a costa de producir confusión en el conjunto de prueba. Esta intuición se basaba en las extrañas visualizaciones CAM que en variadas imágenes parecían indicar áreas no relevantes al problema. El análisis de las activaciones de los filtros de la última capa y posterior selección de características encuentra que existen características que de eliminarse mejoran el modelo, pero su incidencia es mucho menor de lo esperado. Lo que sí es considerable, es la gran redundancia en las características. Se determinó que un 60 % de las características eran redundantes. A pesar de esto, los últimos experimentos muestran que el método propuesto falla en poder reiniciar tales filtros de forma que aprendan patrones distintos.

No fue posible encontrar el vínculo entre los patrones seleccionados como innecesarios en el CAM y las características innecesarias. Dadas las visualizaciones seleccionadas como anómalas, estas tenían gran contribución de filtros relevantes como irrelevantes, por lo tanto no son una forma robusta de encontrar los filtros irrelevantes. También se observan otros problemas como que los filtros son mucho más flexibles de lo que se esperaba, lo que se evidenció en altos cambios en visualizaciones CAM que prácticamente no producían cambios en activa asociados. Por otro lado, el CAM por construcción no permite llegar a los píxeles importantes, sino a que ciertas regiones lo que dificulta su interpretación.

Un camino para mejorar el modelo consistiría en encontrar la forma de asociar las visualizaciones con el error en el conjunto de prueba, de forma de asociar una métrica objetiva con la metodología de selección visualizaciones. Si se lograra tal objetivo se podría guiar la selección de candidatos o incluso automatizar la selección, además de facilitar enormemente realizar repetidas iteraciones de reajuste. Por lo tanto, una posible dirección de trabajo futuro

sería aplicar métodos similares a *Active learning*, donde el modelo sugiere que imágenes son las mejores para consultar al experto.

Los resultados obtenidos de los análisis de activaciones globales podrían ser la base para un nuevo método, lo cual solucionaría algunos de los problemas presentados, pero requiere reformular completamente el ciclo de *feedback* ya que con solo una iteración de *backpropagation* las visualizaciones quedarían obsoletas lo que obligaría a una nueva consulta al experto.

Bibliografía

- [1] Quick, Draw! Dataset dataset de sketch utilizado. <https://github.com/googlecreativelab/quickdraw-dataset>. Accessed: 2018-08-25.
- [2] Informacion mutua Sklearn implementacion de informacion mutua utilizada (sklearn). https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mutual_info_score.html#sklearn.metrics.mutual_info_score. Accessed: 2018-08-25.
- [3] Calculo gradiente softmax calculo gradiente softmax. <https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/>. Accessed: 2018-08-25.
- [4] Extra trees Sklearn implementacion de extra-trees utilizada (sklearn). <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html>. Accessed: 2018-08-25.
- [5] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018. URL <http://arxiv.org/abs/1803.08375>.
- [6] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.
- [7] Y-Lan Boureau, J Ponce, and Yann Lecun. A theoretical analysis of feature pooling in visual recognition. pages 111–118, 11 2010.
- [8] David M. Chan, Roshan Rao, Forrest Huang, and John F. Canny. t-sne-cuda: Gpu-accelerated t-sne and its applications to modern data. *CoRR*, abs/1807.11824, 2018. URL <http://arxiv.org/abs/1807.11824>.
- [9] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.
- [10] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun. Very deep convolutional networks for natural language processing. *CoRR*, abs/1606.01781, 2016. URL <http://arxiv.org/abs/1606.01781>.
- [11] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. doi: 10.1007/

- BF02551274. URL <https://doi.org/10.1007/BF02551274>.
- [12] Alexey Dosovitskiy and Thomas Brox. Inverting convolutional networks with convolutional networks. *CoRR*, abs/1506.02753, 2015. URL <http://arxiv.org/abs/1506.02753>.
 - [13] Peter Eston. *Understanding Machine Learning:From Theory to Algorithms*.
 - [14] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015. URL <http://arxiv.org/abs/1508.06576>.
 - [15] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, Apr 2006. ISSN 1573-0565. doi: 10.1007/s10994-006-6226-1. URL <https://doi.org/10.1007/s10994-006-6226-1>.
 - [16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
 - [17] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems*, pages 2625–2633, 2013.
 - [18] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24:8 – 12, 05 2009. doi: 10.1109/MIS.2009.36.
 - [19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL <http://arxiv.org/abs/1512.03385>.
 - [20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL <http://arxiv.org/abs/1502.03167>.
 - [21] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.
 - [22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
 - [23] Shin Dong Kyun, Minhaz Uddin Ahmed, and Phill-Kyu Rhee. Incremental deep learning for robust object detection in unknown cluttered environments. *CoRR*, abs/1810.10323, 2018. URL <http://arxiv.org/abs/1810.10323>.
 - [24] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL <http://yann.lecun.com/exdb/mnist/>.

- [25] Yann Lecun, Leon Bottou, Y Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998. doi: 10.1109/5.726791.
- [26] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *CoRR*, abs/1811.07871, 2018. URL <http://arxiv.org/abs/1811.07871>.
- [27] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation - learning an optimal data augmentation strategy. *CoRR*, abs/1703.08383, 2017. URL <http://arxiv.org/abs/1703.08383>.
- [28] Fabian Offert. "i know it when i see it". visualization and intuitive interpretability. *arXiv preprint arXiv:1711.08042*, 2017.
- [29] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017. URL <http://arxiv.org/abs/1712.04621>.
- [30] Zhuwei Qin, Fuxun Yu, Chenchen Liu, and Xiang Chen. How convolutional neural network see the world - A survey of convolutional neural network visualization methods. *CoRR*, abs/1804.11191, 2018. URL <http://arxiv.org/abs/1804.11191>.
- [31] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL <http://arxiv.org/abs/1804.02767>.
- [32] Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996. ISBN 0521460867.
- [33] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6. URL <http://dl.acm.org/citation.cfm?id=65669.104451>.
- [34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL <http://arxiv.org/abs/1409.0575>.
- [35] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. URL <http://arxiv.org/abs/1606.03498>.
- [36] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016. URL <http://arxiv.org/abs/1610.02391>.
- [37] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are

- adversarial examples inevitable? *CoRR*, abs/1809.02104, 2018. URL <http://arxiv.org/abs/1809.02104>.
- [38] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.
 - [39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL <http://arxiv.org/abs/1409.1556>.
 - [40] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
 - [41] Steven W. Smith. *The Scientist Engineer’s Guide to Digital Signal Processing*. California Technical Pub, 1997. ISBN 0966017633.
 - [42] Irwin Sobel. An isotropic 3x3 image gradient operator. *Presentation at Stanford A.I. Project 1968*, 02 2014.
 - [43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 06 2014.
 - [44] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL <http://arxiv.org/abs/1312.6199>.
 - [45] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. *CoRR*, abs/1808.01974, 2018. URL <http://arxiv.org/abs/1808.01974>.
 - [46] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. URL <http://arxiv.org/abs/1905.11946>.
 - [47] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010. URL <http://portal.acm.org/citation.cfm?id=1953024>.
 - [48] Karl Weiss, Taghi Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3, 12 2016. doi: 10.1186/s40537-016-0043-6.
 - [49] Marco A Wiering, Hado van Hasselt, Auke-Dirk Pietersma, and Lambert Schomaker.

Reinforcement learning algorithms for solving classification problems. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 91–96. IEEE, 2011.

- [50] Doris Xin, Litian Ma, Jialin Liu, Stephen Macke, Shuchen Song, and Aditya G. Parameswaran. Accelerating human-in-the-loop machine learning: Challenges and opportunities. *CoRR*, abs/1804.05892, 2018. URL <http://arxiv.org/abs/1804.05892>.
- [51] Doris Xin, Litian Ma, Jialin Liu, Stephen Macke, Shuchen Song, and Aditya G. Parameswaran. Helix: Accelerating human-in-the-loop machine learning. *CoRR*, abs/1808.01095, 2018. URL <http://arxiv.org/abs/1808.01095>.
- [52] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *CoRR*, abs/1506.06579, 2015. URL <http://arxiv.org/abs/1506.06579>.
- [53] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. *arXiv preprint arXiv:1801.07892*, 2018.
- [54] John R. Zech, Marcus A. Badgeley, Manway Liu, Anthony B. Costa, Joseph J. Titano, and Eric K. Oermann. Confounding variables can degrade generalization performance of radiological deep learning models. *CoRR*, abs/1807.00431, 2018. URL <http://arxiv.org/abs/1807.00431>.
- [55] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. URL <http://arxiv.org/abs/1311.2901>.
- [56] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015. URL <http://arxiv.org/abs/1512.04150>.

Capítulo 8

Anexos

8.1. Derivación fórmula gradiente

Dada la arquitectura estudiada se tiene el siguiente diagrama. Se ignora la capa *ReLU* debido a que el gradiente es prácticamente el mismo (el cambio es que no hay gradiente si la activación es negativa).

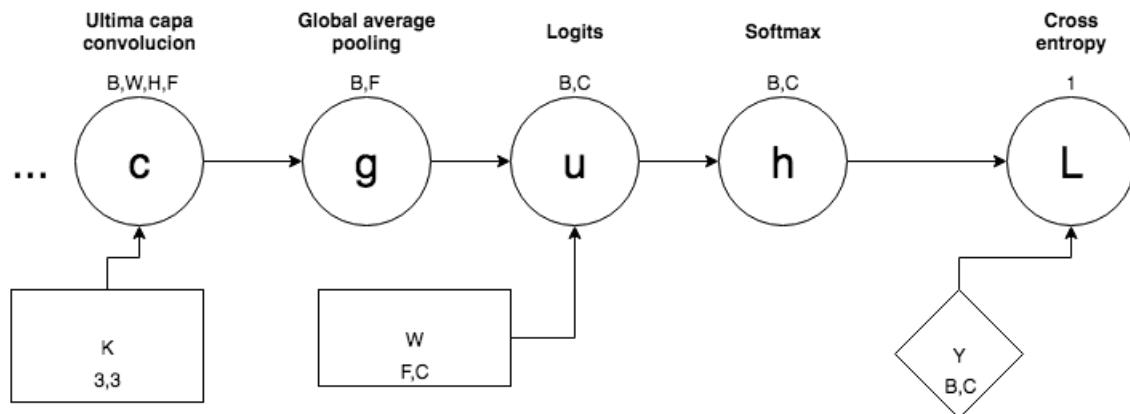


Figura 8.1: Diagrama arquitectura

8.1.1. Gradiente entropía cruzada respecto a *softmax*

La pérdida corresponde a la entropía cruzada dada por

$$L = CE = -\frac{1}{N} \sum_i^N \sum_j^C Y_{ij} \cdot \log(h_{ij}) \quad (8.1)$$

El gradiente respecto a la capa *softmax* es

$$h_{k,l} = \frac{e^{u_{k,l}}}{\sum_t^C e^{u_{k,t}}} \quad (8.2)$$

$$\frac{\partial L}{\partial h_{a,b}} = -\frac{1}{N} \frac{1}{h_{a,b}} Y_{a,b}$$

8.1.2. Gradiente de logits [3]

$$\frac{\partial L}{\partial u_{k,l}} = \sum_a^a \sum_b^b \frac{\partial L}{\partial h_{a,b}} \frac{\partial h_{a,b}}{\partial u_{k,l}}$$

$$\frac{\partial h_{a,b}}{\partial u_{k,l}} = \begin{cases} h_{k,l} \cdot (1 - h_{k,l}), & \text{for } a = k \wedge b = l \\ -h_{k,b} \cdot h_{k,l}, & \text{for } a = k \wedge b \neq l \\ 0, & \text{for } a \neq k \end{cases} = \delta_{k,a} \cdot h_{k,l} \cdot (\delta_{l,b} - h_{a,b})$$

$$\sum_a^a \sum_b^b \frac{\partial L}{\partial h_{a,b}} \frac{\partial h_{a,b}}{\partial u_{k,l}} = -\frac{1}{N} \sum_a^a \sum_b^b \frac{1}{h_{a,b}} Y_{a,b} \cdot \delta_{k,a} \cdot h_{k,l} \cdot (\delta_{l,b} - h_{a,b})$$

$$-\frac{1}{N} \sum_b^b \frac{1}{h_{k,b}} Y_{k,b} \cdot h_{k,l} \cdot (\delta_{l,b} - h_{k,b})$$

$$-\frac{1}{N} \frac{1}{h_{k,b^*}} Y_{k,b^*} \cdot h_{k,l} \cdot (\delta_{l,b^*} - h_{k,b^*}) \quad ^1$$

$$\frac{\partial L}{\partial u_{k,l}} = \begin{cases} -\frac{1}{N} \cdot (1 - h_{k,b^*}) = -\frac{1}{N} \cdot (1 - h_{k,l}) & \text{for } (Y_{k,b^*} = Y_{k,l}) \Leftrightarrow (Y_{k,l} = 1) \\ -\frac{1}{N} \cdot h_{k,l} \cdot (0 - 1) & \text{for } (Y_{k,b^*} \neq Y_{k,l}) \Leftrightarrow (Y_{k,l} = 0) \end{cases} = -\frac{1}{N} \cdot (Y_{k,l} - h_{k,l}) \quad (8.3)$$

¹Este reemplazo es solo valido cuando $Y_{i,j}$ es una matriz de vectores *one hot*. Es decir, solamente un valor es 1 por fila y todos los demás son 0.

8.1.3. Gradiente de pesos logits

$$\begin{aligned}
\frac{\partial L}{\partial W_{m,n}} &= \sum_{-}^r \sum_{-}^s \frac{\partial L}{\partial u_{r,s}} \cdot \frac{\partial u_{r,s}}{\partial W_{m,n}} \\
u_{r,s} &= \sum_{-}^t g_{rt} \cdot W_{ts} \\
\frac{\partial u_{rs}}{\partial W_{mn}} &= \delta_{sn} \cdot g_{rm} \\
\frac{\partial L}{\partial W_{m,n}} &= \sum_{-}^r \sum_{-}^s \frac{\partial L}{\partial u_{r,s}} \cdot \delta_{sn} \cdot g_{rm} = \sum_{-}^r \frac{\partial L}{\partial u_{r,n}} \cdot g_{rm}
\end{aligned} \tag{8.4}$$

8.1.4. Gradiente de *global average pooling*

$$g_{a,b} = \frac{1}{wh} \cdot \sum_{t_1}^w \sum_{t_2}^h c_{a,t_1,t_2,b} \tag{8.5}$$

$$\begin{aligned}
\frac{\partial L}{\partial g_{x,y}} &= \sum_{-}^r \sum_{-}^s \frac{\partial L}{\partial u_{r,s}} \cdot \frac{\partial u_{r,s}}{\partial g_{x,y}} \\
\frac{\partial u_{r,s}}{\partial g_{x,y}} &= \delta_{r,x} \cdot W_{y,s} \\
\frac{\partial L}{\partial g_{x,y}} &= \sum_{-}^s \frac{\partial L}{\partial u_{x,s}} \cdot W_{y,s}
\end{aligned} \tag{8.6}$$

8.1.5. Gradiente de activación filtro

$$\begin{aligned}
\frac{\partial L}{\partial c_{a,b,c,d}} &= \sum_{-}^r \sum_{-}^s \frac{\partial L}{\partial g_{r,s}} \cdot \frac{\partial g_{r,s}}{\partial c_{a,b,c,d}} \\
\frac{\partial g_{r,s}}{\partial C_{a,b,c,d}} &= \delta_{r,a} \cdot \delta_{d,s} \cdot \frac{1}{w \cdot h} \\
\frac{\partial L}{\partial c_{a,b,c,d}} &= \frac{\partial L}{\partial g_{a,d}} \cdot \frac{1}{w \cdot h}
\end{aligned} \tag{8.7}$$

8.1.6. Gradiente de pesos *kernel* última capa

Para el caso de los pesos del *kernel* hay que aplicar una convolución entre el gradiente de la última capa respecto a la pérdida y las activaciones rotadas de la capa anterior, se decide no calcular este último ya que basta con revisar el gradiente hasta la última capa convolucional para entender la tendencia.

8.1.7. Formulas resultantes

Reemplazando las fórmulas el gradiente de la pérdida respecto a los pesos de la última capa y el gradiente de las activaciones son respectivamente.

$$\begin{aligned} \frac{\partial L}{\partial c_{a,b,c,d}} &= \sum_{-}^s \left(-\frac{1}{N} \cdot (Y_{a,s} - h_{a,s}) \cdot W_{d,s} \cdot \frac{1}{w \cdot h} \right) \\ &\quad - \frac{1}{N} \frac{1}{w \cdot h} \sum_{-}^s (Y_{a,s} - h_{a,s}) \cdot W_{d,s} \end{aligned} \tag{8.8}$$

$$\begin{aligned} \frac{\partial L}{\partial W_{m,n}} &= \sum_{-}^r \left(-\frac{1}{N} \cdot (Y_{r,n} - h_{r,n}) \cdot g_{r,m} \right) \\ &\quad - \frac{1}{N} \sum_{-}^r (Y_{r,n} - h_{r,n}) \cdot g_{r,m} \end{aligned} \tag{8.9}$$

8.2. Explicación general del código

El código se encuentra disponible en https://github.com/aferral/mejora_clasificador_feedback_CAM con el fin de ayudar a reproducir los resultados o posibles extensiones futuras.

8.2.1. Dependencias

Este código necesita python 3.6 y las dependencias especificadas en el archivo `./requirements.txt`. Para instalar todas las dependencias se recomienda utilizar un nuevo ambiente de desarrollo, lo cual puede realizarse mediante:

```
1 virtualenv -p python3.6 venv # crear un nuevo ambiente
2 source venv/bin/activate # activar el ambiente
3 pip install -r requirements.txt # Instala depedencias
4 (Ejecutar experimentos)
5 deactivate # Desabilita el ambiente
```

8.2.2. ¿Cómo ejecutar código?

Todos los *scripts* de este proyecto asumen que la estructura base parte en la raíz del directorio, por lo tanto, la forma de llamar un *script* es mediante módulos es decir:

```
1 python -m datasets.imagenet_data # ESTO SI  
2 python datasets/imagenet_data.py # ESTO NO
```

8.2.3. ¿Cómo conseguir los datos?

Forma fácil: Es posible descargar todos los datos en este zip https://drive.google.com/file/d/1Sbm0o_JTHuh17J4CiXdzUi83g4-kpF_V/view?usp=sharing. Descomprimir los contenidos en la carpeta `./temp`. Los contenidos descomprimidos utilizan 37 G de espacio.

IMAGENET subset dataset: Ejecutar el *script* en `tf_records_parser` lo cual descargara el subconjunto de imágenes y los procesara en formato *tfrecord* que leen los clasificadores. Con esto el *dataset* ya esta habilitado.

Sketches dataset: Es necesario seguir el proceso detallado en https://github.com/aferral/Recuperacion_sketchs/blob/master/preproceso.py con esto se genera los *tfrecords*. Luego colocar `./temp/tf_records_quickdraw`.

Symbols dataset: Se requiere seguir el proceso detallado en `./datasets/simple_figures.py`

Otros datasets: Se encuentran *datasets* extras, los cuales no se usaron en todos los experimentos. Mayores detalles en `cifar10_data.py`, `cwr_dataset.py`.

8.2.4. Estructura del código

Classification_models

Agrupa las diversas arquitecturas a utilizar. Es necesario múltiples modelos debido a que cambian las dimensiones por cada dataset y además cuando es necesario agregar capas. Los clasificadores más importantes son `imagenet_subset_conv_loss.py` que define la función PASA y `classification_model.py` que define la base para todos los clasificadores.

Tf_recordparser

Para acelerar el proceso de los datos todos los *dataset* se convierten a un tipo de archivo llamado *tfrecord*, el cual esta optimizado para ejecutarse en *tensorflow* (librería de proceso de datos). En esta carpeta se encuentran los *scripts* que transforman los diversos *datasets* a

tal formato. Todos están listos para ejecutarse mientras se suministren las carpetas con los datos (revisar primeras líneas de cada archivo para la ubicación de las carpetas).

Datasets

Se crea una clase especial para agrupar todas las funcionalidades necesarias para los *datasets* (`./datasets/dataset.py`). Los *datasets* específicos extienden esta clase definiendo la ubicación de los *tfrecords* y otras funciones de ser necesarias.

ImageNet_Utils

Repositorio extra necesario para descargar los datos de *IMAGENET* de forma automática.

Select_tool

Agrupa todos los objetos necesarios para la interfaz de visualización y selección. Para desplegar la interfaz utilizar:

```
1 python -m select_tool.main
```

Vis_exp

Genera diversas visualizaciones dado los datos procesados. Puede generar una visualización interactiva de una reducción de dimensionalidad de las imágenes, puede transformar todas las visualizaciones CAM en imágenes, puede producir visualización de cada filtro en específico, etc. En esta carpeta es de gran importancia `./vis_exp/visualization_exp.py` que contiene las visualizaciones generadas en el Capítulo discusiones (visualización de activaciones dado todo el *dataset*).

Config_files

En esta carpeta se guardan en formato *JSON* los archivos que representan los entrenamientos de los clasificadores. Se utilizan para agrupar parámetros en configuraciones. A medida que se entrenen modelos los resultados aparecerán en esta carpeta.

Plot_utils

Toma los resultados de diversos experimentos y los agrega en tablas. Fue utilizado para generar las tablas en *LATeX*de forma automática.

Image_generator

Funciones utilizadas para el método de reajuste por reemplazos. Para descargar el modelo generativo es necesario ejecutar `deploy_generative_inpainting.sh`, el cual descarga el modelo generativo entregado por https://github.com/JiahuiYu/generative_inpainting.

```
1 sh deploy_generative_model.sh
```

8.2.5. Proceso para reproducir resultados

1. **Descarga de datos:** Revisar sección anterior “¿Cómo conseguir los datos?”
2. **Entrenar modelos iniciales:** Para entrenar el clasificador dado un *dataset* se debe escribir un archivo de configuración, ejemplos de estos se encuentran en `./config_files/train_files`. Una vez seleccionado el archivo ejecutar:

```
1 python -m exp_scripts.do_interactive_exp <path_a_config_file>
```

3. **Localizar configuración de resultado:** El entrenamiento genera un modelo, el cual se guarda a disco con un archivo de configuración que lista todos los parámetros utilizados. Estos archivos de resultados se encuentran en `./config_files/train_results`.
4. **Seleccionar áreas irrelevantes:** Se selecciona las áreas irrelevantes mediante la interfaz. Para abrir el modelo entrenado utilizar el siguiente comando.

```
1 python -m select_tool.main <path_a_result_config>
```

5. **Exportar selección:** Para exportar las selecciones de la interfaz es necesario presionar “export sel for gen”. Esto generará dos archivos uno con todos los parámetros de la selección en `./config_files/select_files` y su máscara en `./config_files/mask_files`.
6. **Configurar reajuste:** Con estos parámetros solo queda configurar la forma del reajuste. Esto se realiza en `./exp_scripts/exp_backprops.py` donde es necesario editar las últimas líneas con todos los parámetros detallados anteriormente. En estas mismas líneas se elige el tipo de ajuste a realizar. Una vez configurado realizar:

```
1 python -m exp_scripts.exp_backprops
```

7. **Visualizar resultados:** Los resultados del experimento del paso anterior se encuentran en `./out_backprops/<nombre_exp>`. Dentro de esta carpeta se encuentran visualizaciones de cada CAM por iteración, la *accuracy* en cada iteración y las matrices de confusión. Además, se grafican las imágenes con errores antes y después del método.