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

W2AVV++: SISTEMA QUE COMBINA EL ANÁLISIS DE TEXTO CON DESCRIPTORES VISUALES Y AUDITIVOS PROFUNDOS PARA LA RECUPERACIÓN DE VIDEOS SIN ETIQUETAR

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

RODRIGO ANDRÉS HERNÁNDEZ PUGA

PROFESOR GUÍA:
JUAN MANUEL BARRIOS NUÑEZ

MIEMBROS DE LA COMISIÓN:
NANCY HITSCHFELD KAHLER
BÁRBARA POBLETE LABRA
JULIO GODOY DEL CAMPO

SANTIAGO DE CHILE
2020

Resumen

La cantidad de contenido multimedia que se genera en el mundo es cada día mayor. Herramientas como Google y YouTube facilitan la búsqueda de este tipo de datos, utilizando la metadata asociada a un video como por ejemplo su categoría y su título. Un caso de uso particular es cuando un usuario busca acciones u objetos sobre videos que no tienen metadata asociada. Por ejemplo, buscar en un disco duro con videos y recuperar los que tengan escenas de mujeres cantando. La mayoría de los sistemas que intentan resolver este problema se enfocan en el análisis del contenido visual de los videos ignorando el canal auditivo.

El objetivo general de este trabajo es implementar un sistema que permita la búsqueda por texto de videos sin etiquetar, utilizando un modelo de redes neuronales que combine descriptores visuales y auditivos. Se trabaja extendiendo el modelo W2VV++, que utiliza solo descriptores visuales, para agregar el componente auditivo.

La hipótesis de este trabajo es que el uso del audio es relevante en un sistema que resuelve el problema de la recuperación de videos sin etiquetar. Para esto se implementa un modelo base que no utiliza las pistas de audio de los videos y uno nuevo que sí las utiliza, para luego compararlos en base a ciertas métricas relevantes.

Se entrena un modelo de redes neuronales utilizando el dataset MSR-VTT, que contiene un conjunto de 200.000 pares video-descripción. Cada descripción de texto es vectorizada usando las técnicas Bag-of-Words, word2vec y una red GRU. Se obtienen descriptores visuales de cada video extrayendo sus cuadros y utilizando modelos pre entrenados de redes CNN. También se obtienen descriptores auditivos de las pistas de audio utilizando modelos pre entrenados. Se proponen dos esquemas de fusión: Early Fusion para unir los descriptores visuales y auditivos de un video en un único descriptor audiovisual; e Intermediate Fusion en donde se unen los descriptores visuales en un único descriptor visual y los descriptores auditivos en un único descriptor auditivo, y luego se fusionan ambos para generar un único descriptor audiovisual. Finalmente, se entrena una red neuronal que proyecta los descriptores de texto al espacio de descriptores audiovisuales.

Usando el conjunto de datos de prueba, ambos esquemas audiovisuales obtienen levemente mejores resultados que el modelo visual. Al realizar un experimento con 20 consultas donde se buscan conceptos audiovisuales, el esquema Early presenta una mejora de un 20 % de Precision@1 con respecto a W2VV++ y el esquema Intermediate presenta una mejora de un 12 % de Precision@10 con respecto a W2VV++. Se incluyen además los resultados de la participación con un modelo audiovisual propuesto en la conferencia TRECVID, en donde se obtuvieron valores de Mean infAP de 0,041 y 0,040 en dos entregas distintas. Cabe destacar que el modelo propuesto es el único entre todos los participantes en utilizar el audio.

Para las tres mujeres más importantes de mi vida,
Luisa, Paola y Carla.

Agradecimientos

Es imposible agradecer en una sola página a todas las personas que he conocido en estos años en la facultad y a quienes me han acompañado en este camino, pero lo intentaré.

Gracias a mi familia por todo el esfuerzo de siempre hacer de nuestra casa un hogar. A mi abuela Luisa por ser la gran matriarca que siempre fue, hasta el último de sus días. A mis papás por dar todo de sí mismos cada día para que a nosotros sus hijos no nos falte nada. Por preocuparse cada mañana en la que veían que yo estaba medio inconsciente encima del escritorio terminando algún trabajo o estudiando. Gracias por ayudarme a ser quien soy, no habría llegado ni a la mitad de este camino si no fuera por ustedes.

Gracias a todas las personas que conocí en mi primer año en la facultad, en particular los de mi sección. A Pedruca, JP, Darío, Pinga, Perroni y Alex, por ser mis grandes pilares en plan común. Gracias por las muchas, muchas, muchísimas risas durante cada día que pasé con ustedes. Por el apoyo y la ayuda para entender cosas que parecían imposibles para mi. Por ser incondicionales y nunca abandonarme.

Gracias a los grandes cabros de "Pareto's", por hacer de ese rincón lleno de computadores en el DCC un lugar en donde jamás me sentí solo. Un lugar donde podía hablar de lo que quisiera y reír cuanto quisiera. A Pareto, Badilla, Diego, Juan, Nacho y Javier. Gracias a Gino's Pizza por alimentarnos en ese rincón una cantidad innumerable de veces. Ese lugar siempre será el primer recuerdo alegre que aparecerá en mi cabeza al pensar en el DCC, porque ahí conocí a algunas de las mejores personas que he conocido en la vida.

Gracias a los que más allá de la U, se convirtieron en amigos de la vida. A Alonso Reyes, Cristóbal Miranda, Rodrigo Quezada y Martín Panza, por ser los mejores amigos que uno podría pedir. Gracias por darme apoyo en cada momento en que lo necesité, por darme su amistad, por soportar lo insoportable que puedo llegar a ser y por siempre hacerme reír. Gracias a ustedes he podido llegar tan lejos y estar donde estoy.

Gracias a mi comisión por darse el trabajo de revisar mi tesis y en especial a Juan Barrios, por ser un tremendo profesor guía y un excelente académico. Gracias por las conversaciones, las enseñanzas, las risas y los consejos. Gracias a Orand por dejarme usar sus servidores, sin los cuales no podría haber entrenado ni a un pokémon.

Finalmente, quiero agradecer a Carla Arias, por ser la gran compañera de vida que es. Gracias por nunca dejar que me rinda, por estar siempre a mi lado y por ayudarme a ser mejor persona cada día.

Tabla de Contenido

1. Introducción **1**
1.1. Motivación . . . 1
1.2. Hipótesis . . . 3
1.3. Objetivos . . . 3
1.3.1. Objetivo general . . . 3
1.3.2. Objetivos específicos . . . 4
1.4. Metodología . . . 4
1.5. Estructura de la tesis . . . 5

2. Marco Teórico **6**
2.1. Redes Neuronales . . . 6
2.1.1. Perceptrón . . . 6
2.1.2. Multi Layer Perceptron . . . 7
2.1.3. Arquitectura de una red neuronal . . . 7
2.1.4. Proceso de entrenamiento de una red neuronal . . . 8
2.2. Redes Convolucionales . . . 12
2.2.1. Tipos de Capas en una Red Convolucional . . . 13
2.2.2. Deep Features . . . 17
2.2.3. ResNet . . . 18
2.2.4. ResNeXt . . . 19
2.3. Análisis de Audio . . . 20
2.3.1. SoundNet . . . 21
2.3.2. AENet . . . 23
2.4. Análisis de Texto . . . 25
2.4.1. Bag-of-Words . . . 25
2.4.2. Word2Vec . . . 26
2.4.3. Redes Neuronales Recurrentes . . . 29
2.4.4. Gated Recurrent Unit . . . 29
2.5. Métricas Importantes . . . 30

3. Revisión Bibliográfica **33**
3.1. Word2VisualVec (W2VV) . . . 33
3.2. Word2VisualVec++ (W2VV++) . . . 35

4. Descripción e Implementación del Sistema **37**
4.1. Descripción General del Sistema . . . 37
4.2. Dataset Utilizado . . . 37

4.3. Representación Visual . . . 38
4.4. Representación Auditiva . . . 38
4.5. Representación de Texto . . . 39
4.6. Combinación de descriptores unimodales . . . 39

5. Resultados **43**
5.1. Experimentos sobre conjunto de test . . . 44
5.2. Experimentos sobre consultas audiovisuales . . . 47

6. Participación en TRECVID **50**
6.1. Ad-hoc Video Search . . . 50
6.2. Hybrid Intermediate W2AVV++ . . . 52
6.3. Resultados . . . 54

7. Conclusiones **60**
7.1. Trabajo a Futuro . . . 62

Bibliografía **63**

Anexo A. Resultados de experimentos con consultas audiovisuales **67**

Anexo B. Ad-hoc Video Search (AVS) 2019 **78**
B.1. Equipos participantes . . . 78
B.2. Resultados de las cuatro entregas realizadas . . . 79
B.3. Mean infAP de todos los participantes . . . 80
B.4. Consultas de AVS 2019 . . . 82
B.5. Resultados de cada entrega con respecto a todos los participantes . . . 84

Índice de Tablas

5.1. Tamaño de cada conjunto de datos utilizado para los experimentos 44
5.2. Resultados del Experimento 1: Función de Pérdida 44
5.3. Resultados del Experimento 2: Learning Rate 45
5.4. Resultados del Experimento 3: Función de Optimización 45
5.5. Resultados del Experimento 4: Batch Size 46
5.6. Resultados del Experimento 5: Triplet Ranking Loss Alpha Margin 46
5.7. Resultados de experimento con 20 consultas con conceptos audiovisuales . . . 48
6.1. inferred Average Precision obtenido por consulta en cada entrega. 56
B.1. Información de los equipos participantes en la competencia AVS 2019. 78
B.2. Listado de los resultados y posiciones obtenidas por las 4 entregas realizadas en cada consulta de AVS 2019. 79
B.3. Listado de todas las entregas de cada equipo y su Mean infAP correspondiente. 81

Índice de Ilustraciones

1.1. Búsqueda en YouTube: “a dog barking” . . . 1
1.2. Ejemplo de video de MSR-VTT . . . 3
2.1. Gráficos de funciones Sigmoid, Tanh y ReLU. . . . 7
2.2. Ejemplo de un MLP con dos capas ocultas . . . 8
2.3. Efectos del learning rate sobre la pérdida . . . 10
2.4. Ilustración del uso de Dropout . . . 12
2.5. Ilustración de una ConvNet . . . 13
2.6. Ilustración de un volumen de entrada en una capa convolucional . . . 14
2.7. Ejemplo de filtros aprendidos en la primera capa convolucional . . . 14
2.8. Ejemplo de convoluciones de una dimensión . . . 14
2.9. Ejemplo del uso de zero-padding . . . 15
2.10. Ejemplo de operaciones de una capa convolucional . . . 16
2.11. Ejemplo de capa de pooling . . . 17
2.12. Bloque de capas de una red residual . . . 18
2.13. Comparativa entre bloque de ResNet y bloque de ResNeXt . . . 19
2.14. Ejemplo de un espectrograma . . . 21
2.15. Ilustración de la arquitectura de SoundNet . . . 22
2.16. Ilustración del modelo AENet. . . . 23
2.17. Ilustración de las arquitecturas que componen a Word2Vec. . . . 28
2.18. Ilustración de una GRU. . . . 30
2.19. Ground Truth de la consulta Q en ejemplo de Average Precision . . . 31
2.20. Ejemplo de Average Precision . . . 32
3.1. Ilustración de la arquitectura de W2VV . . . 35
3.2. Ilustración de la arquitectura de W2VV++ . . . 36
4.1. Distribución de categorías de videos de dataset MSR-VTT . . . 38
4.2. Visualizaciones de conceptos semánticos que SoundNet aprende . . . 39
4.3. Ilustración del esquema Early Fusion . . . 40
4.4. Representación del modelo W2AVV++ utilizando Early Fusion . . . 41
4.5. Representación del modelo W2AVV++ utilizando Intermediate Fusion . . . 42
5.1. Precision@K obtenido por cada sistema en promedio . . . 49
6.1. Representación del modelo Hybrid Intermediate W2AVV++ utilizado en AVS 53
6.2. Mean infAP obtenido por cada sistema entregado en AVS . . . 55
6.3. Captura de video recuperado por la consulta 612 . . . 57
6.4. Capturas de videos recuperados por la consulta 618. . . . 57
6.5. Videos recuperados en la consulta 631 por `AudioV02`. . . . 58
6.6. Videos recuperados en la consulta 631 por `Visual02`. . . . 59
A.1. Videos recuperados en la consulta 1 del experimento audiovisual. . . . 67

A.2. Videos recuperados en la consulta 2 del experimento audiovisual. 68
A.3. Videos recuperados en la consulta 3 del experimento audiovisual. 68
A.4. Videos recuperados en la consulta 4 del experimento audiovisual. 69
A.5. Videos recuperados en la consulta 5 del experimento audiovisual. 69
A.6. Videos recuperados en la consulta 6 del experimento audiovisual. 70
A.7. Videos recuperados en la consulta 7 del experimento audiovisual. 70
A.8. Videos recuperados en la consulta 8 del experimento audiovisual. 71
A.9. Videos recuperados en la consulta 9 del experimento audiovisual. 71
A.10. Videos recuperados en la consulta 10 del experimento audiovisual. 72
A.11. Videos recuperados en la consulta 11 del experimento audiovisual. 72
A.12. Videos recuperados en la consulta 12 del experimento audiovisual. 73
A.13. Videos recuperados en la consulta 13 del experimento audiovisual. 73
A.14. Videos recuperados en la consulta 14 del experimento audiovisual. 74
A.15. Videos recuperados en la consulta 15 del experimento audiovisual. 74
A.16. Videos recuperados en la consulta 16 del experimento audiovisual. 75
A.17. Videos recuperados en la consulta 17 del experimento audiovisual. 75
A.18. Videos recuperados en la consulta 18 del experimento audiovisual. 76
A.19. Videos recuperados en la consulta 19 del experimento audiovisual. 76
A.20. Videos recuperados en la consulta 20 del experimento audiovisual. 77
B.1. Resultados obtenidos con AudioV05 84
B.2. Resultados obtenidos con Visual05 84
B.3. Resultados obtenidos con AudioV02 85
B.4. Resultados obtenidos con Visual02 85

Capítulo 1

Introducción

1.1. Motivación

La cantidad de contenido multimedia que se genera en el mundo es cada día mayor. Con la creación de nuevas redes sociales y la facilidad de acceso a Internet de alta velocidad, existe una tendencia a generar contenido audiovisual y compartirlo en Internet. Tomando en cuenta el gran volumen de datos multimedia disponible, se vuelve necesario contar con buscadores web especializados que permitan encontrar cierto contenido en específico de manera eficaz.

Herramientas como Google y YouTube logran este objetivo con muy buenos resultados, aunque con una condición: los datos deben estar etiquetados con la información suficiente para ser recuperados por el buscador eficazmente. Por ejemplo, YouTube le permite a los usuarios subir videos acompañados de diferentes atributos que conocemos como "metadata", que sirven para identificar de mejor manera el contenido del video. Por ejemplo, el título del video, la descripción, distintos hashtags asociados, etc. Luego, cuando un usuario busca un video, el buscador utiliza la metadata de los videos para hacer un match con la consulta realizada y así poder hacer un ranking de los videos más relevantes. En la Figura 1.1 se muestra un ejemplo, en donde se buscan videos relacionados a "a dog barking". El buscador hace un match entre las palabras de la consulta y la metadata de los videos, para luego recuperar de forma ordenada aquellos en donde sí existe un match. Se observa que las palabras "dog barking" aparecen en los títulos y en las descripciones de los videos recuperados.

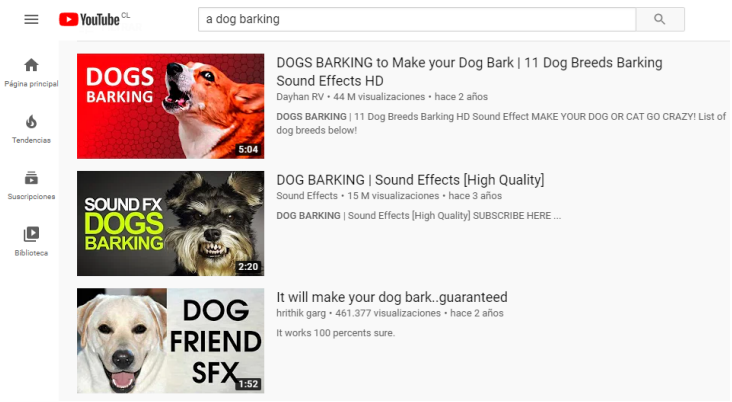


Figura 1.1: Búsqueda en YouTube: "a dog barking"

Hay dos casos de uso interesantes a considerar: buscar sobre un conjunto de videos sin metadata asociada y buscar acciones no etiquetadas. Por ejemplo, buscar en un disco duro con videos y recuperar aquellos que tengan escenas de mujeres cantando. Este problema es conocido como "Recuperación de videos sin etiquetar" y es el foco principal de este trabajo de investigación.

En el área de recuperación de información multimedia se han realizado avances notables en los últimos 15 años. La principal conferencia científica dedicada a la investigación en torno a este tema es TRECVID [2]. Esta conferencia genera grandes datasets de prueba públicos y comunes para todos los participantes, además de métricas de evaluación uniformes y en donde pueden participar equipos de todo el mundo. Cada año la conferencia propone diversas competencias o "*tasks*", en donde los equipos deben inscribirse para participar con su investigación. En particular, la tarea "Ad-hoc Video Search" (AVS) intenta resolver el problema de la recuperación de videos sin etiquetar.

La mayoría de los sistemas propuestos en la literatura que intentan resolver AVS, se enfocan en construir un modelo de redes neuronales y entrenarlo para que aprenda a analizar una consulta escrita en lenguaje natural, convertir la consulta en un vector de características conocido como "descriptor" o "feature", y luego utilizar ese descriptor de texto para encontrar los descriptores visuales correspondientes a los videos que sean similares. Por lo tanto, se realiza una proyección desde el espacio de los descriptores textuales al espacio de los descriptores visuales, para que estos puedan ser comparados.

Existe un modelo llamado Word2VisualVec++ (W2VV++) [25] que corresponde al estado del arte, ganador de la competencia AVS en el año 2018, y es el que se utiliza en este trabajo como inspiración para proponer una extensión. Este sistema utiliza modelos de redes neuronales convolucionales ResNet y ResNeXt pre entrenados para poder obtener descriptores visuales de videos. Luego, aplica técnicas de vectorización de texto para generar descriptores textuales a partir de descripciones en texto de los videos e intenta emparejar estos descriptores con los visuales mediante una función de distancia. De esta forma, el modelo aprende características similares entre un descriptor textual y uno visual, y luego es capaz de predecir descriptores visuales a partir de una consulta en texto.

Para lograr entrenar un modelo como el descrito anteriormente, se necesitan datasets que permitan realizar una proyección entre el espacio textual y el visual. En general, los datasets que se utilizan para esto consisten de videos de diferentes temas y de descripciones acerca de estos videos hechas por personas que los vieron y plasmaron en texto lo que les llamó la atención. Por ejemplo, el dataset MSR-VTT [44] contiene alrededor de 10.000 videos provenientes de YouTube y 20 descripciones en texto por video, proporcionando un total de aproximadamente 200.000 pares de video-descripción. Se puede ver un ejemplo en la Figura 1.2.

Los resultados obtenidos hasta el momento provienen de modelos entrenados en su mayoría solo con el componente visual de los videos, por lo que surgen las siguientes preguntas: ¿Qué efectos tendría en el aprendizaje de un modelo el incluir el componente auditivo? ¿Sería mejor, en términos de recuperación de videos, incluir el componente auditivo dentro del entrenamiento del modelo? ¿En cuáles casos sería mejor incluirlo y en cuáles no?

Figura 1.2: Ejemplo de video de MSR-VTT y algunas descripciones:

1. *a video game of a small animal*
2. *a scene from the ice age video game is shown*
3. *a cartoon animals runs through an ice cave in a video game*
4. *a weasel runs around in an icy cave*

En este trabajo, se propone un sistema que contempla el componente visual y el auditivo de los videos, llamado Word2AudioVisualVec++ (W2AVV++), y se evalúa la relevancia del audio en el contexto del problema de la recuperación de videos sin etiquetar.

1.2. Hipótesis

La hipótesis que motiva este trabajo es que el uso del audio es relevante en un sistema buscador de videos sin etiquetar. Un sistema entrenado con el componente visual y el auditivo de los videos podrá recuperar más videos relevantes en comparación con uno que no utiliza el audio.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo principal de este trabajo es implementar un sistema de búsqueda por texto de videos sin etiquetar, combinando los descriptores visuales y auditivos de los videos en un modelo de redes neuronales.

1.3.2. Objetivos específicos

La construcción del modelo propuesto en este trabajo involucra los siguientes objetivos específicos:

1. Implementar y entrenar el modelo W2VV++ utilizando el dataset MSR-VTT.
2. Implementar y entrenar el modelo propuesto W2AVV++ con un esquema Early Fusion utilizando el dataset MSR-VTT.
3. Implementar y entrenar el modelo propuesto W2AVV++ con un esquema Intermediate Fusion utilizando el dataset MSR-VTT.
4. Evaluar y comparar el desempeño de ambos esquemas mediante el uso de las métricas Recall@K, Precision@K y Mean Average Precision.
5. Participar con el modelo W2AVV++ en la conferencia TRECVID en la competencia Ad-Hoc Video Search.

1.4. Metodología

La metodología de esta investigación se compone de los siguientes pasos:

1. Se descarga el dataset MSR-VTT y los modelos pre entrenados de las redes ResNet, ResNeXt, SoundNet y AENet.
2. Se calculan los descriptores visuales ResNet y ResNeXt de cada video usando los modelos pre entrenados.
3. Se calculan los descriptores auditivos SoundNet y AENet de la pista de audio de cada video usando los modelos pre entrenados.
4. Se utiliza como base la implementación del modelo W2VV y se extiende implementando la función de pérdida Triplet Ranking Loss, como es propuesto por Faghri et al. [12] para obtener el modelo W2VV++.
5. Se implementa el modelo W2AVV++ con un esquema Early Fusion, generando un descriptor audiovisual por video mediante la fusión de los 4 descriptores mencionados y el entrenamiento de una red neuronal general.
6. Se implementa el modelo W2AVV++ con un esquema Intermediate Fusion, generando un descriptor visual por video mediante la fusión de los 2 descriptores visuales por un lado, un descriptor auditivo por video mediante la fusión de los 2 descriptores auditivos por otro lado; y finalmente un descriptor audiovisual por video al fusionar ambos descriptores mencionados para después entrenar una red neuronal general.
7. Utilizando parte del dataset MSR-VTT como test, se compara el desempeño de los modelos implementados mediante el uso de métricas relacionadas a la recuperación de información.

8. Se participa en la tarea AVS de la conferencia TRECVID con una versión alternativa del modelo propuesto llamada Hybrid Intermediate W2AVV++, donde primero se fusionan los descriptores visuales generando un descriptor visual por video y luego se fusiona esto con los descriptores auditivos. Los resultados de esta participación se publican en Hernández et al. [17].

1.5. Estructura de la tesis

Este trabajo está organizado de la siguiente forma:

1. En el capítulo 2 se describe el marco teórico en el cual está basada esta investigación, incluyendo los diferentes tipos de redes neuronales convolucionales utilizadas para extraer los descriptores, las técnicas utilizadas para vectorizar las descripciones de cada video y las métricas utilizadas para evaluar el desempeño de los modelos implementados.

2. En el capítulo 3 se habla sobre el estado del arte en referencia a modelos que intentan resolver el problema de recuperación de videos sin etiquetar.

3. En el capítulo 4 se describe en detalle la implementación del modelo propuesto en este trabajo.

4. En el capítulo 5 se exponen los resultados obtenidos mediante el uso de tablas e imágenes.

5. En el capítulo 6 se habla sobre la participación en la conferencia TRECVID con el modelo propuesto. Se discute sobre la competencia en la cual se participa, el dataset utilizado y los resultados obtenidos.

6. Finalmente, en el capítulo 7 se concluye sobre la investigación realizada y los resultados obtenidos. Además, se habla sobre la contribución que este trabajo representa y sobre las posibles proyecciones de este a futuro.

Capítulo 2

Marco Teórico

En este capítulo se describen diferentes conceptos importantes utilizados dentro de esta investigación. Se presenta una base teórica sobre las redes neuronales y las redes neuronales convolucionales, detallando los modelos ResNet, ResNeXt, SoundNet y AENet. Además se explica el uso de técnicas para el análisis y vectorización de texto y el uso de métricas relevantes para evaluar los resultados del modelo propuesto.

2.1. Redes Neuronales

El contenido de esta sección y de la Sección 2.2 está basado en los libros "Neural Networks and Deep Learning" [31], "Deep Learning" [15] y en el material del curso "CS231n Convolutional Neural Networks for Visual Recognition" de la Universidad de Stanford [21].

Una red neuronal es un modelo computacional que consiste de un conjunto de unidades conocidas como "neuronas", las que se conectan de diferentes maneras entre ellas. Cada neurona tiene una función de activación que determina su salida de acuerdo a su entrada. La red recibe datos contenidos en tensores, los procesa en capas compuestas por neuronas y los propaga a través de las conexiones entre estas. Cada neurona realiza una operación sobre una parte de los datos y el resultado es entregado a la siguiente neurona. Cuando se llega a la última capa de la red se obtiene como resultado la aproximación de una función diferenciable, el que luego de varias iteraciones se puede ir refinando para resolver un problema.

2.1.1. Perceptrón

El perceptrón es un modelo matemático desarrollado al final de la década de 1950 por el científico Frank Rosenblatt [32] y está inspirado en el funcionamiento de una neurona biológica. En este modelo los datos de entrada son inputs (x_0, x_1, x_2), los que son ponderados con pesos (w_0, w_1, w_2), valores que expresan la importancia de un input con respecto al output de una neurona. El output de una neurona está determinado por:

$$output = \begin{cases} 0, & \text{si } f(\sum_i w_i x_i) \leq \text{ umbral} \\ 1, & \text{si } f(\sum_i w_i x_i) > \text{ umbral} \end{cases} \tag{2.1}$$

La Ecuación 2.1 se puede reescribir como:

$$output = \begin{cases} 0, & \text{si } f(w \cdot x + b) \leq 0 \\ 1, & \text{si } f(w \cdot x + b) > 0 \end{cases}$$

El *bias* (b) puede ser visto como una medida de qué tan fácil es que el perceptrón se encienda o no (retorne como output 1 o 0). Si un perceptrón tiene un *bias* muy grande, entonces es más fácil que entregue como output 1. Existen diferentes funciones de activación (f), entre ellas:

1. **Sigmoid**: $\boldsymbol{\sigma(x) = 1/(1 + \epsilon^{-x})}$. Se puede observar graficada en la Figura 2.1.
2. **Tanh**: $\boldsymbol{\tanh(x) = 2\sigma(2x) - 1}$. Esta función es similar a la sigmoidea, la diferencia es que la salida de esta función se centra en el 0, como se puede ver en la Figura 2.1.
3. **ReLU (Rectified Linear Unit)**: $\boldsymbol{f(x) = \max(0, x)}$. Se puede observar en la Figura 2.1.

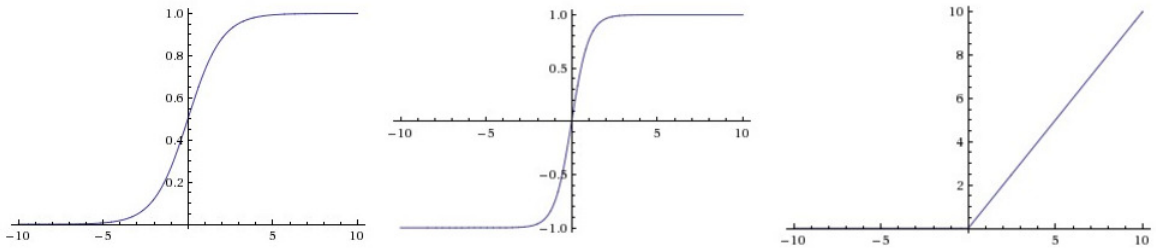


Figura 2.1: **Izquierda**: Función Sigmoid. **Centro**: Función Tanh.
Derecha: Función ReLU

2.1.2. Multi Layer Perceptron

Los Multi Layer Perceptron (MLP desde ahora en adelante) consisten de red compuesta de múltiples perceptrones, los que se organizan en varias capas. El objetivo de un MLP es aproximar cierta función $f : \mathbb{R}^n \to \mathbb{R}^m$, por ejemplo, un clasificador que mapea un input $x = (x_1, \ldots, x_n)$ a un output $y = (y_1, \ldots, y_m)$. El MLP aprende los valores de los parámetros que resultan en la mejor aproximación de la función f.

El término *feedforward* proviene del sentido en el que los datos fluyen a través de la red, siendo consumidos por la capa de entrada como x, operados por las capas intermedias a través de diferentes cálculos que definen a f y luego siendo entregado un output y. Existen redes que incluyen conexiones de *feedback*, que son conocidas como Redes Neuronales Recurrentes y serán explicadas en la Sección 2.4.3.

2.1.3. Arquitectura de una red neuronal

Como se puede ver en la Figura 2.2, una red neuronal está formada por capas de neuronas, las que se pueden dividir en 3 tipos:

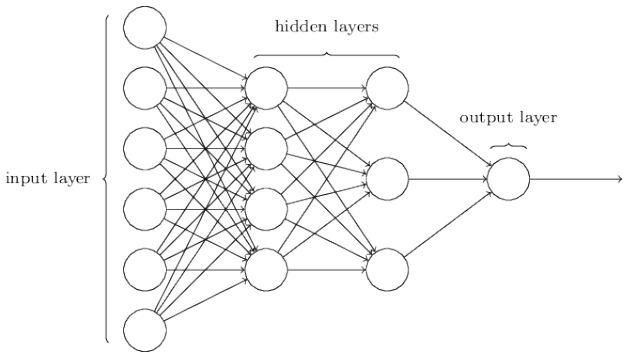


Figura 2.2: Ejemplo de un MLP con dos capas ocultas

La capa de entrada (Input Layer) funciona como un receptor de los datos de entrada y no se cuenta como parte del número de capas de una red porque no realiza operaciones. La red de la Figura 2.2 tiene 3 capas: las dos capas ocultas y la capa de salida. Las capas intermedias son conocidas como capas ocultas (Hidden Layers). Las neuronas de estas capas contienen los datos que han sido computados hasta ese punto. Una característica de estas capas es que son *fully connected*, lo que significa que todos los outputs de la capa anterior están conectados como inputs de la capa actual. La capa de salida (Output Layer) corresponde a la última capa de la red y es la encargada de entregar el valor de salida.

Está demostrado que una red neuronal con al menos una capa oculta es un "aproximador universal" [7]. Dada alguna función continua $f(x)$ y una constante $\epsilon > 0$, existe una red neuronal $g(x)$ con una capa oculta tal que $\forall x, |f(x) - g(x)| < \epsilon$. La capacidad de una red para ser un aproximador universal depende de que la capa oculta tenga suficientes neuronas [19]. El problema que esto conlleva es que la red podría ser incapaz de realmente aprender y generalizar de manera correcta. Frente a esto, usar modelos con más capas ocultas puede reducir la cantidad de neuronas requeridas para modelar la función deseada. Un caso en donde la profundidad de la red entrega mejor desempeño son las Redes Convolucionales, de las que se hablará en la Sección 2.2.

2.1.4. Proceso de entrenamiento de una red neuronal

La idea de entrenar una red es que aprenda observando datos reales y luego reciba datos que no ha visto antes y aproxime un resultado. Por ejemplo, una red neuronal se puede entrenar para clasificar fotos en categorías: fotos de perros y fotos de gatos.

Antes de comenzar a entrenar una red, hay que escoger una función de pérdida o función de costo. Esta función indica qué tan cerca del resultado esperado se encuentra la aproximación realizada por la red, es decir, sirve para medir la calidad del output. Si el resultado de la función de pérdida (conocido como "pérdida") es muy alto, entonces la red no está aproximando de manera correcta la función objetivo. Un ejemplo de una función de pérdida es la función MSE (Mean Squared Error):

$$C = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.2}$$

En la Ecuación 2.2, n es la cantidad de datos de entrenamiento, i es el i-ésimo dato de

entrenamiento, y_i es el *ground truth* del i-ésimo dato de entrenamiento e $\hat{y}_i$ es la predicción realizada por la red para el i-ésimo dato de entrenamiento. El *ground truth* de un dato corresponde al resultado esperado, por ejemplo, en un problema de clasificación corresponde a la etiqueta real de un dato.

Luego de escoger una función de pérdida, el objetivo del entrenamiento es ajustar los pesos y *biases* de la red de tal manera que la pérdida sea lo más pequeña posible. Para lograr esto se utiliza un algoritmo conocido como *backpropagation*, que consiste en utilizar la expresión de la derivada parcial de la función de pérdida C con respecto a un peso w (o un *bias* b). Esta expresión representa qué tan rápido cambia la pérdida cuando los pesos y los *biases* son modificados. El objetivo de *backpropagation* es calcular $\frac{\partial C}{\partial w}$ y $\frac{\partial C}{\partial b}$ con respecto a cualquier peso w o *bias* b en la red, para luego ajustarlos de modo que cuando se vuelva a realizar un paso hacia adelante en la red, la pérdida sea menor.

Para que este método funcione, la función de pérdida se escribe como un promedio $C = \frac{1}{n}\sum_x C_x$ sobre las funciones de pérdida C_x para cada ejemplar de entrenamiento x. Se asume esto dado que el algoritmo de *backpropagation* calcula las derivadas parciales $\frac{\partial C_x}{\partial w}$ y $\frac{\partial C_x}{\partial b}$ para cada dato de entrenamiento y se obtienen las derivadas parciales $\frac{\partial C}{\partial w}$ y $\frac{\partial C}{\partial b}$ promediando todos los datos de entrenamiento.

El algoritmo define las siguientes ecuaciones importantes:

$$z^{(L)} = w^{(L)} \cdot a^{(L-1)} + b$$
$$a^{(L)} = \sigma(z^{(L)})$$
$$C = (a^{(L)} - y)^2$$

Donde L corresponde a la última capa, w es el peso de una neurona, a es la activación de una neurona, C es la función de pérdida y b es el *bias*. Luego, se definen las siguientes derivadas parciales:

$$\frac{\partial C}{\partial w^{(L)}} = \frac{\partial C}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial w^{(L)}}$$
$$\frac{\partial C}{\partial b^{(L)}} = \frac{\partial C}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial b^{(L)}}$$
$$\frac{\partial C}{\partial a^{(L-1)}} = \frac{\partial C}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}}\frac{\partial z^{(L)}}{\partial a^{(L-1)}}$$

Cada derivada parcial de los pesos y los *biases* se guarda en el gradiente ∇, que en este caso quedaría definido como:

$$\nabla C(w_1, b_1, \ldots, w_n, b_n) = \begin{bmatrix} \frac{\partial C}{\partial w_1} \\ \frac{\partial C}{\partial b_1} \\ \vdots \\ \frac{\partial C}{\partial w_n} \\ \frac{\partial C}{\partial b_n} \end{bmatrix}$$

Un *batch* corresponde a un grupo de datos que se pasa por la red, el que se divide en *mini-batches*, que son grupos pequeños de datos. El gradiente es calculado dentro de un *mini-batch*, donde por cada dato se promedia el output de cada peso y *bias* y luego ese promedio se convierte en el output del gradiente. Después de pasar por cada *mini-batch*, se actualizan estos parámetros con las siguientes ecuaciones:

$$w^{(l)} = w^{(l)} - \eta \frac{\partial C}{\partial w^{(l)}}$$
$$b^{(l)} = b^{(l)} - \eta \frac{\partial C}{\partial b^{(l)}}$$

Donde l corresponde a una capa en particular y η corresponde a un hiperparámetro conocido como *learning rate*, que sirve para controlar qué tan rápido se ajustan los pesos y *biases* de la red. Es importante elegir el valor de este hiperparámetro con cuidado, debido a que si el valor es muy bajo entonces se avanza muy poco a través del gradiente. En cambio, si el valor es muy grande, entonces puede que nunca se logre converger.

Una *epoch* consiste en un paso completo de todos los datos de entrada por la red. En la Figura 2.3 se puede observar los efectos de distintos valores del *learning rate* sobre la pérdida en una red a través de las *epochs*. Con valores bajos de *learning rate*, las actualizaciones son muy lentas por lo que se necesita mayor tiempo de entrenamiento. Con valores muy altos, la pérdida disminuye más rápido pero tiende a quedarse "atrapada" en peores valores.

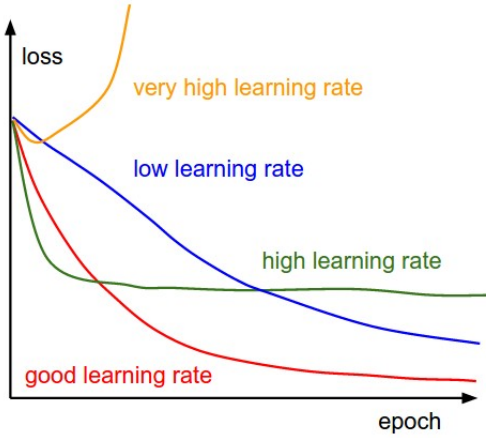


Figura 2.3: Ilustración que muestra los efectos de diferentes valores de *learning rate* sobre la pérdida a lo largo de las epochs.

Para las capas anteriores, se vuelven a repetir los pasos del *backpropagation* y se usa lo calculado para la última capa para ir encadenando más derivadas parciales y así avanzar hacia atrás. Actualizar los pesos y *biases* de la capa L depende solo de la función de pérdida, y esta depende de los pesos y *biases* conectados a esa capa (que provienen de la capa oculta $L-1$). De la misma forma, actualizar la capa $L-1$ depende de los cálculos hechos en la capa L y de los pesos y *biases* de la capa $L-1$, como se puede ver a continuación:

$$\frac{\partial C}{\partial w^{(L-1)}} = \frac{\partial C}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}}$$
$$= \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}}$$

$$\frac{\partial C}{\partial b^{(L-1)}} = \frac{\partial C}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}}$$
$$= \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}}$$

Existen distintos métodos de optimización para la función de pérdida y así minimizar su resultado. Un ejemplo es Stochastic Gradient Descent (o SGD), un método iterativo que consiste en tomar un *mini-batch* de datos aleatorios y actualizar los pesos y *biases* en base al promedio del gradiente del *mini-batch*.

En el entrenamiento, los pesos y *biases* son iniciados de manera aleatoria con valores pequeños, generalmente entre [0,1]. Los pasos para entrenar de manera resumida son los siguientes:

1. Definir una función de pérdida, como MSE, que recibe un vector de pesos o de *bias*.
2. Partir en un punto aleatorio con los pesos y *biases*, obteniendo un valor de output con forward propagation.
3. Calcular el gradiente usando *backpropagation*.
4. Ajustar los pesos en la dirección opuesta del gradiente.

Se repiten estos pasos hasta converger a un mínimo global de la función de pérdida. Existen dos problemas presentes durante el entrenamiento: *underfitting* y *overfitting*. El primero ocurre cuando un modelo no logra capturar la tendencia intrínseca de los datos de entrada, es decir, no logra aprender lo suficiente de los datos. Este problema en general es resultado de un modelo excesivamente simple o de muy pocas iteraciones (cantidad de veces que un *batch* es pasado a través de la red) de los datos. El segundo problema, *overfitting*, ocurre cuando un modelo aprende demasiado de los datos que está mirando hasta el punto de capturar el ruido de estos. Es decir, el problema ocurre cuando el modelo se ajusta demasiado bien a los datos que analiza. Esto es resultado de un modelo excesivamente complicado o de demasiadas iteraciones de los datos de entrenamiento.

Para controlar el grado de *overfitting* que un modelo presenta, existen técnicas de regularización como las que se presentan a continuación:

1. **Regularización L1 y L2**: A cada peso w se le añade el término $\lambda|w|$ para L1 y $\frac{1}{2}\lambda w^2$ para L2, donde λ es la fuerza de regularización.

2. **Dropout** [39]: Consiste en dejar activa una neurona según cierta probabilidad p -un nuevo hiper parámetro- o apagarla (dejarla en cero). Esta técnica es aplicada durante el entrenamiento y puede ser configurada por capa. Al usarla, se evita que ciertas neuronas se conecten fuertemente y se aprendan los datos. Se puede ver una ilustración del Dropout en la Figura 2.4.

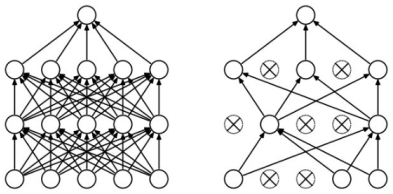


Figura 2.4: **Izquierda**: figura de una red neuronal estándar.
Derecha: La misma red neuronal con la aplicación de Dropout.

Optimización del Learning Rate

Existen técnicas que regulan el *learning rate* de manera dinámica durante el proceso de entrenamiento:

1. **Adagrad**: Este método adapta el *learning rate* según los parámetros, utilizando valores bajos con parámetros asociados con *features* que aparecen frecuentemente, y valores altos con *features* poco frecuentes. Es útil para usar con datos dispersos.

2. **RMSprop**: Este método ajusta el método Adagrad para frenar la reducción agresiva del *learning rate*. Utiliza el promedio de los gradientes cuadrados y lo pasa a través de los pesos.

3. **Adam (Adaptive Moment Estimation)**: Guarda el promedio de los gradientes cuadrados y el promedio de los gradientes anteriores. Este valor se comporta como si fuera un "freno" del decaimiento del *learning rate*.

2.2. Redes Convolucionales

Las Redes Neuronales Convolucionales (CNN o ConvNets) son otro tipo de redes neuronales *feedforward* que usa convoluciones sobre los datos. Estas redes asumen que los datos de entrada son imágenes, lo que permite darle ciertas características a la arquitectura de la red para reducir la cantidad de parámetros e implementar una función de propagación más eficiente. Una característica importante es el ordenamiento que tienen las neuronas en las capas de una ConvNet. Por ejemplo, si la red recibe como input una imagen de dimensiones 200x200x3 (200 pixeles de ancho, 200 pixeles de alto y 3 canales de color), una neurona *fully connected* tendría 200*200*3 = 120.000 pesos para aprender, y esto es solo una neurona. Dependiendo de cuántas neuronas tenga cada capa, la cantidad de parámetros que la red debe aprender podría ser gigantesca. Las CNN usan a su favor la estructura de la imagen y ordenan las neuronas en 3 dimensiones: ancho, alto y profundidad, en donde "profundidad" aquí se refiere a los colores de la imagen. La imagen de ejemplo anterior tendría un volumen

de activación con dimensiones 200x200x3. En una ConvNet las neuronas en una capa solo se conectan con una pequeña región de la capa anterior. La salida de una ConvNet reduce el volumen de entrada 3D a un volumen de salida 3D, por ejemplo, de 1x1x10 dimensiones. En la Figura 2.5 se puede ver un ejemplo:

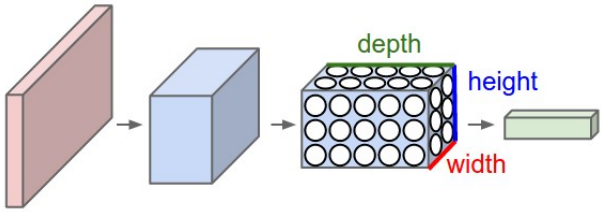


Figura 2.5: Una CNN ordena sus neuronas en 3 dimensiones (ancho, alto y profundidad). El bloque rojo representa a la imagen de entrada, en donde el ancho y alto de la capa corresponden a las dimensiones de la imagen y la profundidad a los canales de colores de la imagen (Red, Green, Blue).

2.2.1. Tipos de Capas en una Red Convolucional

Una ConvNet se compone de una secuencia de capas, en donde cada capa transforma un volumen de activaciones en otro a través de alguna función diferenciable. En general se usan tres tipos principales de capas: capa convolucional, capa de *pooling* y capa *fully connected*. En esta sección se describen los distintos tipos de capas y sus hiperparámetros.

Capa convolucional

Los parámetros (o pesos) de estas capas consisten de una serie de filtros que la red aprende, conocidos como *kernels*. Cada filtro es espacialmente reducido a lo largo de su ancho y alto, pero se extiende por completo a través de la profundidad de un input. Por ejemplo, un filtro en la capa inicial de una red podría tener dimensiones 5x5x3. Al ir entrenando la red, cada filtro va identificando diferentes características o *features*. Los filtros que pertenecen a capas menos profundas (más cercanas al input) aprenden características de bajo nivel como bordes de objetos, mientras que los filtros que pertenecen a capas más profundas (más cercanas al output) aprenden características de alto nivel como formas, caras o números.

En el paso hacia adelante de la ConvNet cada filtro realiza una convolución a lo alto y ancho del volumen de entrada. Mientras se va deslizando el filtro sobre el volumen de entrada se van produciendo mapas de activación o *feature maps*, los que representan la activación de cada filtro cuando identifican alguna característica visual relevante como un borde o un símbolo. Cuando los filtros han recorrido todos los pixeles de un volumen de entrada, todos los *feature maps* generados son apilados a lo largo de la dimensión de profundidad de esa capa y se produce un volumen de salida.

Cada neurona se conecta a una región local del volumen de entrada. Esta conectividad es conocida como "campo receptivo" de una neurona. Este campo considera solo una parte local del ancho y alto del volumen de entrada. Se puede ver un ejemplo de esto en la Figura 2.6:

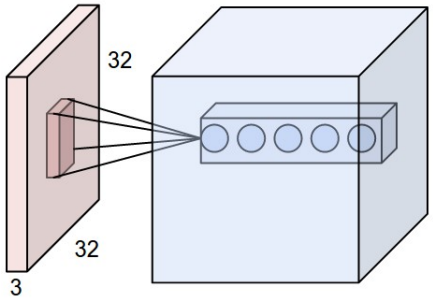


Figura 2.6: Ejemplo de un volumen de entrada de dimensiones 32x32x3 (figura izquierda) y del volumen de una capa convolucional formada por neuronas. Cada neurona solo está conectada a una región local del volumen de entrada de manera espacial.

Existen 3 hiperparámetros fundamentales que controlan el tamaño del volumen de salida:

1. **Profundidad:** Cantidad de filtros que se usan, donde cada uno busca aprender algo distinto de la entrada. Un set de varias neuronas que operan sobre la misma región de la entrada se conoce como *depth column*.

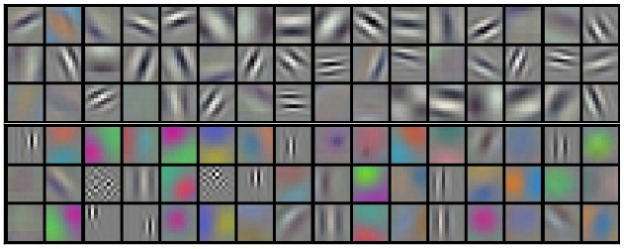


Figura 2.7: Ejemplo de los filtros aprendidos en la primera capa convolucional del modelo propuesto por Krizhevsky et al. [23]

2. **Stride:** Representa la distancia con la que se mueve el filtro aplicado sobre la entrada. Es decir, cuando el *stride* es 1 entonces los filtros se mueven un pixel por convolución. Si el *stride* es 2, entonces el filtro se desliza 2 pixeles cada vez. Mientras mayor sea el valor del *stride*, menor será el tamaño de salida espacialmente.

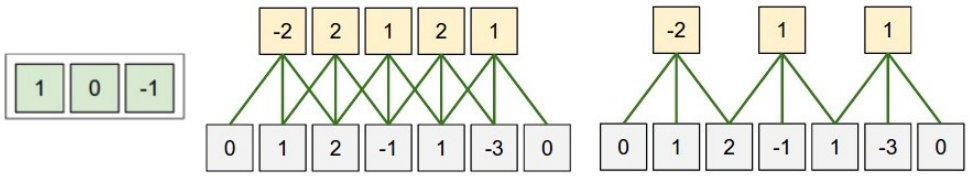


Figura 2.8: Ilustración de convoluciones de una dimensión en el eje X. A la izquierda (bloques verdes) se encuentra el filtro [1,0,-1]. A la derecha se observa el resultado (bloques amarillos) de aplicar el filtro en la entrada (bloques grises) usando un *stride* igual a 1 y luego usando un *stride* igual a 2.

3. **Zero-padding:** Indica la cantidad de ceros que se agregan en los bordes del volumen de entrada. Esto ayuda a controlar el tamaño espacial de los volúmenes de salida, ya que permite preservar el tamaño que tiene el volumen de entrada, obteniendo outputs con el mismo ancho y alto que el input.

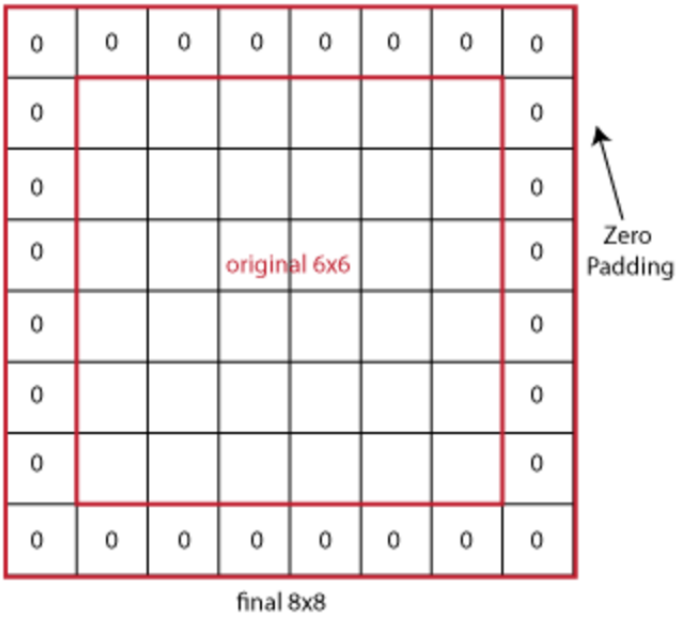


Figura 2.9: Ejemplo de uso de *zero-padding* igual a 1 sobre un volumen de entrada. Se observa que se agregan pixeles con valor 0 alrededor de los bordes de la entrada.

Se puede ver un ejemplo de una capa convolucional y sus operaciones en la Figura 2.10. Al igual que en las redes neuronales regulares, en las ConvNets también se realiza *backpropagation* durante el proceso de aprendizaje para actualizar los pesos y *biases* de la red. En este caso, la operación en el paso hacia atrás también es una convolución pero con los filtros rotados espacialmente. Esta rotación es producto de la derivación de la pérdida en la red.

Usualmente, después de una capa convolucional se agrega una capa con una función de activación. Esta función se aplica sobre la salida de la capa convolucional, y las dimensiones del volumen de salida de esa capa en general se mantienen. Entre las funciones de activación utilizadas en una ConvNet se encuentran las mismas funciones clásicas que se utilizan en una red neuronal regular, como la función sigmoid, tanh o RELU.

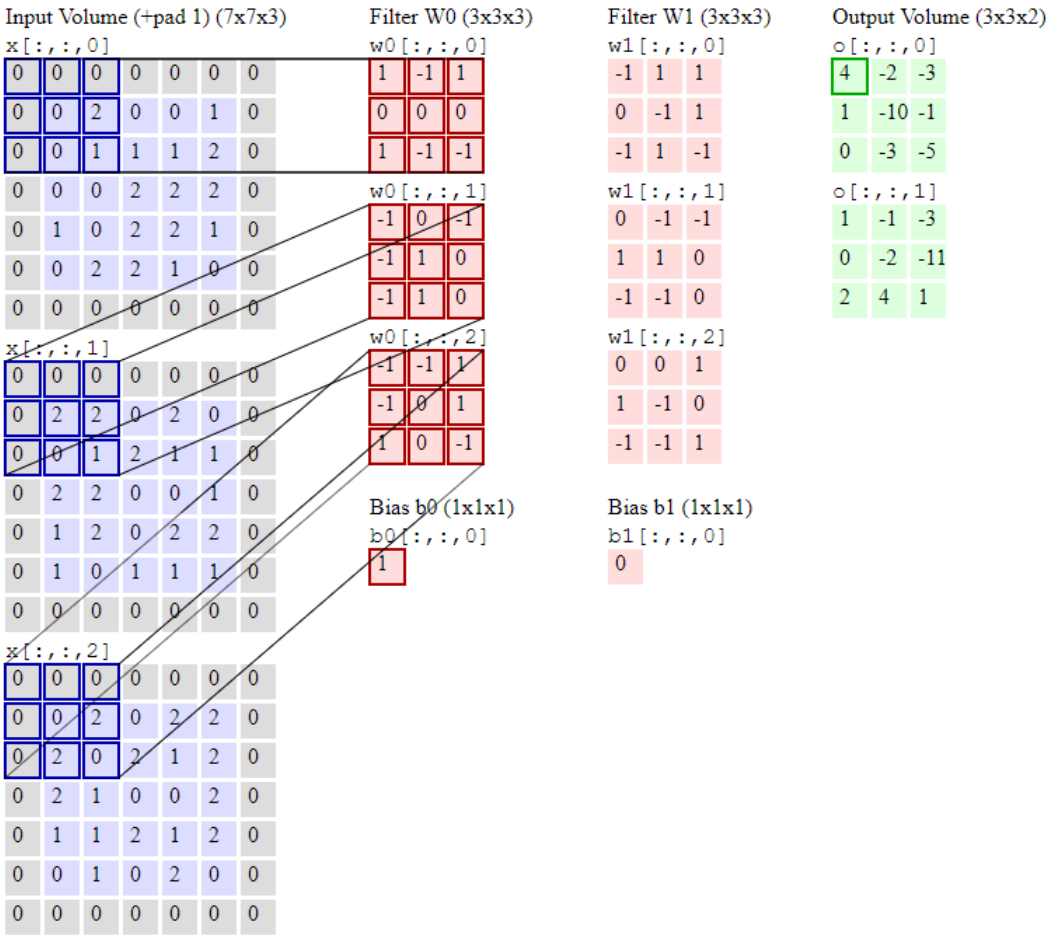


Figura 2.10: Ejemplo de una capa convolucional y sus operaciones. En azul se tiene un volumen de entrada de dimensiones 7x7x3 (los datos son de 5x5x3 pero se agrega *zero-padding* de valor uno). En rojo se tienen los filtros de dimensiones 3x3x3, los que son convolucionados para generar el volumen de salida, ilustrados de color verde. Cada filtro es operado según la tercera dimensión (profundidad) y luego de sumar el *bias* correspondiente, se obtiene el valor de un punto del volumen de salida.

Capa de pooling

El objetivo de este tipo de capas es disminuir progresivamente el tamaño espacial de los volúmenes que se van generando para así reducir la cantidad de parámetros y operaciones que la red realiza. Típicamente, las capas de *pooling* se insertan entre capas convolucionales sucesivas. Estas capas operan de manera independiente en cada nivel de la dimensión de profundidad del input y lo redimensionan espacialmente. La forma más común de realizar esto es usando filtros MAX de tamaño 2x2 con un *stride* de valor 2, descartando de esta forma un 75 % de las activaciones (ver Figura 2.11). Luego cada operación MAX toma 4 números y selecciona al máximo. De esta forma, la dimensión de profundidad se mantiene intacta.

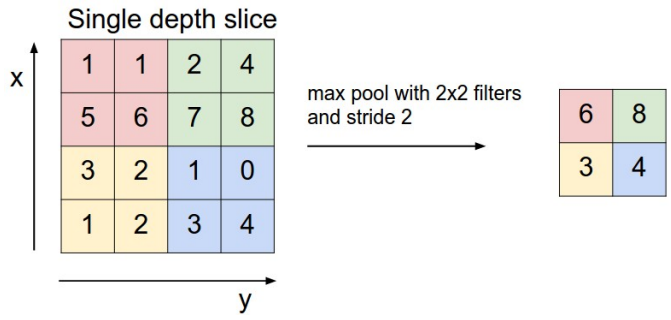


Figura 2.11: Ejemplo de una capa de *pooling*, en donde se realiza un *downsample* del volumen de salida. Usando filtros de max *pooling* de 2x2, se toma el máximo por cada zona y se genera un nuevo volumen de salida.

En el caso de una capa de max *pooling*, el algoritmo de *backpropagation* consiste en enfocarse en el input que tenía el valor máximo en el paso hacia adelante. Por lo tanto, durante el paso hacia adelante se debe guardar el índice de la activación con el valor máximo para luego calcular el gradiente en el paso hacia atrás.

Capa fully connected

Estas capas contienen neuronas que están conectadas a todas las activaciones de la capa anterior, al igual que en las redes neuronales normales. Estas neuronas son activadas dependiendo de una multiplicación matricial más la suma del *bias*. La entrada de estas capas es la salida de la última capa de pooling o de convolución, a la que se le aplica la operación *flatten*, que corresponde a "desenrollar" una matriz y convertirla en un tensor de 1 dimensión.

2.2.2. Deep Features

Las ConvNet se pueden usar como extractores de características (tambien llamadas descriptores o *features*). Esto se realiza entrenando una red para que aprenda diferentes elementos relevantes de las imágenes de entrada, como por ejemplo bordes o figuras. Luego, al pasar una imagen de prueba a través de la red, se puede extraer el output de alguna de sus capas ocultas para obtener los *features* que se aprendieron en esa capa. Los *Deep Features* se pueden obtener en capas más profundas de una ConvNet, donde se aprenden características de alto nivel. La forma en la que se realiza esta extracción consiste en utilizar ConvNets que han sido entrenadas previamente con datasets muy grandes como ImageNet [8] o COCO [27] en problemas de clasificación. Luego, se pasan los datos de prueba a través de la red y se extraen los *features* obtenidos en alguna capa anterior a las capas *fully connected*. Existen múltiples ConvNets pre-entrenadas para ser utilizadas como extractores de características, tales como AlexNet [24] y VGG [35].

En este trabajo se utilizan cuatro modelos distintos en donde dos se utilizan para extraer descriptores visuales y los otros dos se usan para extraer descriptores auditivos. Estos modelos son ResNet (Sección 2.2.3), ResNeXt (Sección 2.2.4), SoundNet (Sección 2.3.1) y AENet (Sección 2.3.2).

2.2.3. ResNet

Mientras una ConvNet tenga más capas y por lo tanto sea más profunda, es más difícil entrenarla debido a la cantidad de parámetros que debe aprender. A medida que se agregan más capas a un modelo, aparece un problema conocido como *vanishing gradient*, en donde el gradiente va desapareciendo durante el *backpropagation* debido a que su valor es muy pequeño, provocando que los pesos no se actualicen y se estanquen. Para facilitar el entrenamiento de una red con mucha profundidad y evitar el problema anterior, se desarrolló un tipo de red conocido como Residual Neural Network (o ResNet) [16].

La principal característica de una red residual, es la de incorporar *skip connections* o *shortcuts* (ver Figura 2.12), los que se utilizan para saltar por sobre una o más capas. Los *shortcuts* toman el input de un bloque de capas y le aplican una función identidad, para luego agregar el output de este salto a la salida del bloque de capas apiladas. El uso de *shortcuts* de funciones identidad no agrega parámetros adicionales ni complejidad computacional, lo que es una ventaja para la red completa. Esta puede ser entrenada por completo utilizando Stochastic Gradient Descent y *backpropagation*, al igual que una ConvNet convencional.

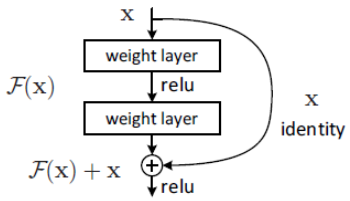


Figura 2.12: Ilustración del uso de un *shortcut* en un bloque de capas de una red residual.

La definición formal del aprendizaje residual propuesto con las ResNet es la siguiente: sea $H(x)$ una función a ser aproximada por unas capas apiladas en bloque (no necesariamente toda la red), en donde x representa a los inputs de la primera capa de este bloque. Se plantea como hipótesis que múltiples capas no lineales pueden aproximarse de manera asintótica a la función residual definida como $F(x) := H(x) - x$, en donde se asume que el input y el output son de las mismas dimensiones. En lugar de que el bloque se aproxime a $H(x)$, se transforma esta definición para que un bloque de capas se aproxime a una función residual $F(x) + x$.

Se utiliza este aprendizaje residual cada cierto número de capas apiladas, en general cada 2 o 3 capas. Un bloque como el de la Figura 2.12 se define formalmente como:

$$y = F(x, \{W_i\}) + x \tag{2.3}$$

En la Ecuación 2.3 x e y son los vectores de entrada y salida de las capas de un bloque respectivamente. La función $F(x, \{W_i\})$ representa el mapeo residual que se espera aprender. Las conexiones *shortcut* de esta fórmula no agregan parámetros extra ni complejidad computacional. Esto permite comparar una red residual directamente con una ConvNet tradicional, ya que ambas poseen la misma cantidad de parámetros, profundidad, ancho y costo computacional. En la Ecuación 2.3 las dimensiones de x y de F deben ser iguales, pero si esto no ocurre se puede realizar una proyección lineal W_s en el *shortcut* para poder equiparar las dimensiones, como se puede apreciar en la Ecuación 2.4. Es importante notar que este tipo

de bloques de capas apiladas y el uso de *shortcuts* aplica tanto para capas *fully connected* como para capas convolucionales.

$$y = F(x, \{W_i\}) + W_s x \tag{2.4}$$

Es importante destacar que este modelo obtuvo el primer lugar en ILSVRC 2015 (ImageNet Large Scale Visual Recognition Competition)[1]. Esta arquitectura ha tenido una influencia mayor en el diseño de redes neuronales profundas luego de su publicación debido a los buenos resultados obtenidos. En particular, este tipo de red se utiliza en este trabajo como un extractor de características, lo que será explicado en detalle en el Capítulo 4.

2.2.4. ResNeXt

El modelo ResNeXt fue propuesto por Xie et al. [43] en el año 2016. Esta red consiste en repetir de forma paralela un bloque residual que agrega una serie de transformaciones con la misma topología (ver Figura 2.13). Este módulo en la red realiza las transformaciones, cada una en un *embedding* de baja dimensionalidad, y agrega sus outputs sumándolos. El diseño resulta en una arquitectura homogénea de múltiples capas que tiene una cantidad de parámetros similar a la de una ResNet, pero que obtiene mejores resultados. El principal aporte de este modelo es el de agregar una nueva dimensión en una red convolucional (en adición al ancho, alto y profundidad) llamado "cardinalidad", que corresponde al tamaño del set de transformaciones mencionado. Los autores demuestran que incrementar la cardinalidad en una red es más efectivo que agregar más capas en profundidad o aumentar el ancho de las capas. Este modelo obtuvo el segundo lugar en ILSVRC 2016 (ImageNet Large Scale Visual Recognition Competition)[2].

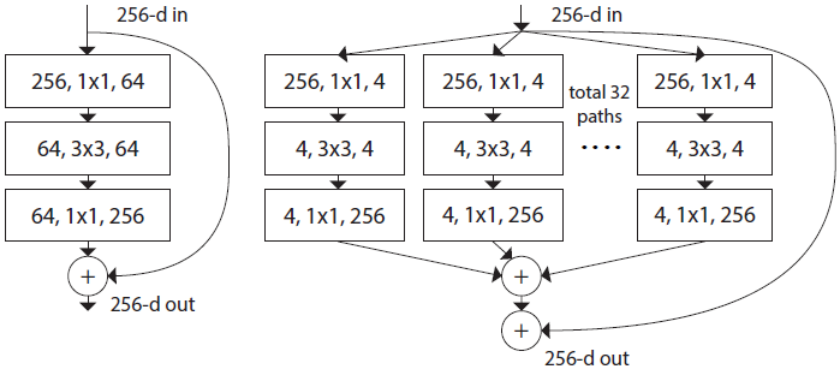


Figura 2.13: **Izquierda**: Un bloque de ResNet. **Derecha**: Un bloque de ResNeXt con cardinalidad = 32.

La red ResNeXt consiste de una pila de bloques residuales. Los bloques comparten la misma topología y se basan en dos reglas que se inspiran en las redes VGG y ResNet: (a) si los bloques producen volúmenes de salida del mismo tamaño, entonces comparten los mismos hiper-parámetros; (b) cada vez que un volúmen de salida se reduce por un factor de 2, el ancho de los bloques es multiplicado por un factor de 2. Esta segunda regla asegura que la

[1] http://image-net.org/challenges/LSVRC/2015/
[2] http://image-net.org/challenges/LSVRC/2016/

complejidad computacional del modelo en términos de FLOPs[3] sea similar para todos los bloques.

Los bloques realizan una combinación de 3 operaciones: *separar*, *transformar* y *agregar*. *Separar*: el vector de entrada x es separado en *embeddings* de baja dimensionalidad x_i. *Transformar*: los *embeddings* son transformados o escalados en $w_i x_i$. *Agregar*: las transformaciones en todos los *embeddings* son sumadas: $\sum_{i=1}^{D}$.

Se definen las transformaciones sumadas como:

$$F(x) = \sum_{i=1}^{C} T_i(x) \tag{2.5}$$

donde $T_i(x)$ puede ser una función arbitraria que aplica una transformación sobre x. En la Ecuación 2.5, C representa el tamaño del set de transformaciones que serán sumadas, el que se conoce como cardinalidad. Es importante considerar que todas las transformaciones T_i tienen la misma forma. Luego, la Ecuación 2.5 sirve para construir la función residual de la Ecuación 2.6, donde y es el output:

$$y = x + \sum_{i=1}^{C} T_i(x) \tag{2.6}$$

Este modelo mejora los resultados obtenidos por ResNet y al igual que ese tipo de red, es utilizado en este trabajo como un extractor de características visuales, lo que sera detallado en el Capítulo 4.

2.3. Análisis de Audio

El sonido es un tipo de energía compuesto por vibraciones y se traslada a través de ondas. Para poder observar las características y comportamientos de este tipo de ondas, se analiza su espectro de frecuencias. Esto es la distribución de amplitudes para cada frecuencia de un fenómeno ondulatorio como el sonido. Para obtenerlo, se utiliza la Transformada de Fourier (FT) sobre una ventana del audio. La FT es una transformación matemática que se usa para transformar señales entre el dominio temporal y el dominio de frecuencias. Un audio se separa en muchas ventanas pequeñas y a cada una se le aplica la FT para obtener el espectro de frecuencias de cada ventana.

Un espectrograma es una representación visual del espectro, en donde se muestra la amplitud de una señal sobre el tiempo y en varias frecuencias presentes a la vez en una onda. Usualmente se utilizan gráficos en 2D donde el eje X corresponde al tiempo y el eje Y corresponde a las frecuencias, las que pueden ser vistas como el tono de un sonido, con las frecuencias más bajas (sonidos graves) en la parte baja del espetrograma y las frecuencias más altas (sonidos agudos) en la parte alta. La amplitud de una frecuencia en particular en un punto del tiempo dado es representada en un espectrograma con el color del gráfico, en donde en general se usan colores oscuros para representar amplitudes bajas y colores brillantes para las amplitudes altas (ver Figura 2.14). Esta visualización permite obtener información de un

[3] Floating-Point Operations, en cantidad de multiplicaciones y adiciones.

audio, y ha sido ampliamente utilizada en el campo del procesamiento de señales auditivas.

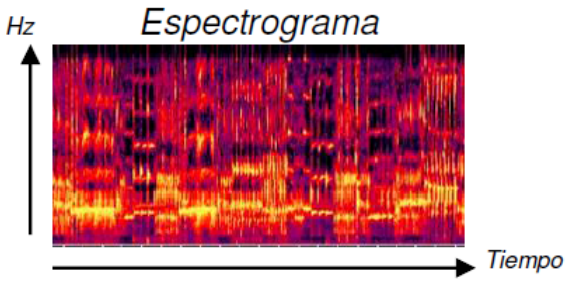


Figura 2.14: Ejemplo de un espectrograma

Existen diferentes descriptores de bajo nivel utilizados para analizar el audio, como por ejemplo Amplitude Envelope, Root-Mean-Square energy, Zero-Crossing Rate, MFCC, entre otros (ver Knees and Schedl [22]). El problema de estos descriptores es que son de bajo nivel, por lo que no hay un análisis más profundo sobre las características del audio. Buscando obtener descriptores auditivos de alto nivel, en los últimos años se han propuesto modelos de redes neuronales convolucionales aplicadas en el sonido. A continuación, se presentan dos modelos de ConvNets que han sido pre-entrenados y que pueden ser utilizados como extractores de características auditivas.

2.3.1. SoundNet

SoundNet transfiere lo aprendido en el espacio visual hacia el espacio auditivo. Este modelo fue propuesto por Aytar et al. [3] en el año 2016. Usa la sincronización natural entre la parte visual y la parte auditiva de un video para así aprender una representación acústica de videos sin etiquetar. El principal aporte que entrega es el desarrollo de una representación semántica de larga escala para el sonido natural.

SoundNet es una red neuronal convolucional profunda cuyo entrenamiento está inspirado en *transfer learning*. *Transfer learning* es una técnica utilizada en el campo de *machine learning*, en donde se utiliza lo aprendido por un modelo que ya está entrenado para resolver un problema y se aplica para resolver otro problema similar. En este caso, se utiliza lo aprendido por modelos de redes neuronales pre-entrenados para detectar objetos y escenas (VGG entrenado con ImageNet [8] y VGG entrenado con Places [46]), y se transfieren estos conocimientos al espacio auditivo para que SoundNet aprenda.

SoundNet se entrena con un dataset compuesto por 2 millones de videos. Estos videos son extraídos desde Flickr y se caracterizan por ser cortos, naturales y por no ser editados de manera profesional. Los videos fueron recopilados con el uso de tags populares y la duración de cada uno varía entre algunos segundos hasta varios minutos. Se convierten las pistas de audio de los videos a formato MP3, se reduce la tasa de muestra a 22 kHZ y se convierte el audio a un solo canal. Para analizar cada pista de audio, estas se dividen en pequeñas ventanas que son entregadas a la capa de entrada de la red. Cada ventana es de aproximadamente 5 segundos.

La arquitectura de SoundNet se compone de una serie de convoluciones de una dimensión

seguido de capas RELU. Se usan capas convolucionales que permiten que la red sea invariante a traslaciones del input. Solo se usan capas convolucionales y capas de *pooling*, incluyendo las funciones no lineales. La capa de salida de la red produce como output una matriz de dimensiones [tiempo, dim], en donde *tiempo* es una dimensión que depende de la duración del audio de entrada y *dim* corresponde al tamaño del descriptor. Se puede ver a grandes rasgos la arquitectura de SoundNet en la Figura 2.15.

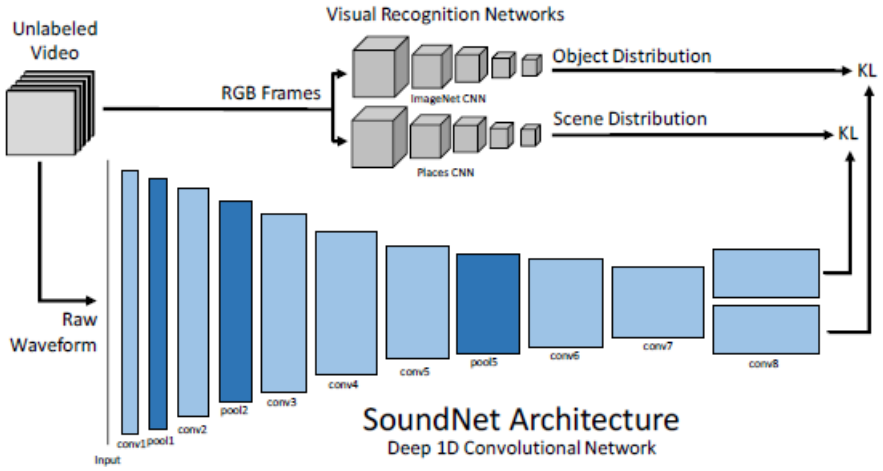


Figura 2.15: Ilustración de la arquitectura de 8 capas convolucionales de SoundNet utilizada para el reconocimiento de sonido natural.

Para entrenar SoundNet se sigue un modelo "profesor-estudiante", donde se tiene como modelo profesor dos modelos que operan sobre el espacio visual para entrenar un modelo estudiante que se encuentra en el espacio auditivo.

Formalmente, sea $x_i \in \mathbb{R}^D$ una onda e $y_i \in \mathbb{R}^{3\times T\times W\times H}$ su video correspondiente con $1 \leqslant i \leqslant N$, donde W, H, T son las dimensiones de ancho, alto y número de ventanas muestreadas de un video respectivamente. Durante el entrenamiento de SoundNet, el objetivo es usar las probabilidades condicionales de una red visual profesora $g_k(y_i)$ para poder entrenar la red estudiante $f_k(x_i)$ para que aprenda a reconocer conceptos según el sonido, donde k enumera los conceptos que se están transfiriendo a la red estudiante. Se usa como función de pérdida $min_\theta \sum_{k=1}^{K} \sum_{i=1}^{N} D_{KL}(g_k(y_i)||f_k(x_i;\theta))$ en donde D_{KL} corresponde a la divergencia Kullback-Leibler (también conocida como entropía relativa) definida como $D_{KL}(P||Q) = \sum_j P_j \log \frac{P_j}{Q_j}$. Se elige un parámetro $K = 2$ debido a que se transfiere lo aprendido por dos redes: la que reconoce escenas y la que reconoce objetos.

Para reconocer las categorías correspondientes a sonidos, los autores usan una capa anterior a la última para extraer los *features* que la red ha aprendido de la entrada, para luego pasarlos por un SVM para entrenarlo como clasificador usando una cantidad pequeña de datos etiquetados con los conceptos que se busca categorizar.

SoundNet se utiliza en el desarrollo de W2AVV++ como un extractor de características auditivas, lo que es detallado en el Capítulo 4.

2.3.2. AENet

Este modelo fue propuesto por Takahashi et al. [41] en el año 2018. Consiste de una red neuronal profunda dedicada al reconocimiento de eventos auditivos. Los eventos auditivos pueden durar varios segundos. Se utiliza una arquitectura de red neuronal convolucional que opera sobre un input temporal grande y además se usa una técnica de incremento de datos para crear el dataset con el que se entrena este modelo.

La principal característica de AENet es que modela un evento auditivo por completo, tomando señales que duran varios segundos como un solo input. Lo anterior significa que el evento auditivo puede ocurrir en cualquier momento en la pista de audio de un video y extenderse solo por una parte, como se ve reflejado en la Figura 2.16.

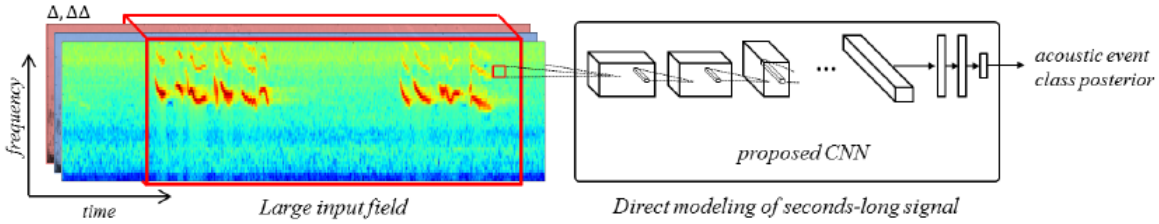


Figura 2.16: Ilustración del modelo AENet. La red toma varios segundos de audio y entrega la probabilidad de que el evento pertenezca a una clase.

AENet es una red neuronal convolucional que aplica convoluciones a lo largo de la duración del audio y en sus frecuencias para manejar cambios en el tono. La red completa está inspirada en VGG. Todas las capas ocultas del modelo AENet a excepción de la última capa *fully connected* aplican RELU al input. Durante el entrenamiento, la red minimiza la función de pérdida *cross entropy* y utiliza regularización l_1 con *backpropagation*, lo que se puede ver en la Ecuación 2.7:

$$\arg\min_{W} \sum_{i,j} L(x_j^i, y_j^i, W) + \rho||W||_1 \tag{2.7}$$

en donde x_j es el vector input j, y_j es su clase correspondiente, W es el set de parámetros de la red y ρ corresponde a una constante que tiene un valor propuesto por los autores de 10^{-6}.

Dado que las redes neuronales convolucionales necesitan de una gran cantidad de datos de entrenamiento para poder aprender de mejor manera, AENet utiliza una técnica de incremento de datos. La mayoría de los sonidos que están mezclados dentro de un evento pertenecen a una misma clase, a menos de que se pueda diferenciar la cantidad de fuentes de sonido. Por ejemplo, si hay dos sonidos diferentes de vidrio quebrándose o de diez explosiones diferentes, aquellos sonidos pertenecen a la misma clase respectivamente. Tomando en cuenta esta propiedad, se propone un incremento en los sonidos al mezclar de manera aleatoria dos sonidos de la misma clase. Además de mezclar los sonidos, estos se perturban al modificar levemente las frecuencias de cada fuente de sonido, aumentando o atenuando la frecuencia de una banda en particular. Un dato incrementado s_{aug} es generado desde las señales de dos

fuentes de la misma clase s_1 y s_2:

$$s_{aug} = \alpha\Phi(s_1(t), \psi_1) + (1 - \alpha)\Phi(s_2(t - \beta T), \psi_2)$$

en donde α, $\beta \in [0, 1)$ son valores aleatorios, T es el *delay* máximo y $\Phi(\cdot, \psi)$ es una función ecualizadora parametrizada por ψ. AENet utiliza una función ecualizadora de segundo orden parametrizada por $\psi = (f_0, g, Q)$ en donde $f_0 \in [100, 6000]$ es la frecuencia central, $g \in [-8, 8]$ es la ganancia y $Q \in [1, 9]$ es un factor que ajusta el ancho de banda del ecualizador. Se pueden obtener muestras aleatorias al seleccionar al azar los parámetros α, β y ψ en cada incremento de datos. Este método es nombrado Equalized Mixture Data Augmentation (EMDA).

Para entrenar AENet, se necesita contar con un dataset con una gran cantidad de clases de eventos auditivos y que además las clases no sean definidas en un nivel semántico demasiado alto (esto significa que no deben ser tan específicas como "reparando un motor"). Considerando lo anterior, los autores construyen un nuevo dataset con clases diseñadas según los siguientes criterios:

1. Las clases cubren la mayor cantidad de eventos auditivos que puedan ocurrir en videos domésticos posible.

2. Los eventos auditivos deben ser atómicos, es decir, no compuestos ni que se sobrepongan. Por ejemplo, una clase "fiesta de cumpleaños" no corresponde debido a que consiste de habla humana y aplausos.

3. La división de clases tampoco debiese ser demasiado específica. Por ejemplo, la clase "campana de iglesia" no debiera ser dividida por tipos de tono del sonido.

El dataset creado y utilizado para entrenar AENet se compone de pistas de audio que provienen de FreeSound [13]. Este es un repositorio de muestras de audio subidas por usuarios común y corrientes, categorizadas en 28 eventos. Un problema presente en este repositorio es que dado que los sonidos están etiquetados de manera libre por los usuarios, existen sonidos irrelevantes en ciertas clases. Por ejemplo, un sonido etiquetado como "gato" podría no ser de un gato real sino que de un sonido producido por un sintetizador. También ocurre que los sonidos muchas veces son ruidosos, lo que representa un desafío. Por otro lado, una ventaja de esto es que los sonidos son realistas y permite a la red aprender sonidos similares a los que se presentan en videos domésticos en general. El dataset construido para AENet extiende las clases de FreeSound a 41 clases, incluyendo clases más diversas provenientes de RWCP Sound Scene Database [30]. Para poder reducir el ruido en los datos, se normalizan todos los sonidos y se eliminan las partes silenciosas. Los sonidos que tengan una duración mayor a 12 segundos son divididos en dos sonidos distintos. Finalmente, todas las pistas son convertidas a una tasa de muestreo de 16 kHZ, de 16 bits por muestra y a un solo canal. Se utilizan pequeñas ventanas de tiempo que se van desplazando como entrada de la red, en donde cada ventana es de 2 segundos de sonido. Una vez entrenada, AENet sirve para extraer las activaciones aprendidas por la red como descriptores auditivos, los que se pueden utilizar en análisis de video.

En este trabajo AENet se utiliza como un extractor de características auditivas, lo que se describe con mayor detalle en el Capítulo 4.

2.4. Análisis de Texto

2.4.1. Bag-of-Words

Bag-of-words (BoW) es una representación del texto en donde se describe la aparición de las palabras dentro de un documento. En esta representación no se considera la gramática del texto ni el orden de las palabras, solo la aparición de estas. Permite transformar un texto, ya sea una frase o un documento extenso, en un vector o descriptor.

En general, para utilizar este modelo primero es necesario obtener o generar un vocabulario de palabras conocidas/relevantes. Luego, se construye una "bolsa" de las palabras relevantes dentro del texto que se busca vectorizar. Esto permite comparar documentos, diciendo que son similares si sus bolsas de palabras relevantes son similares. Para construir un ejemplo[4] concreto de BoW, se presenta el siguiente texto[5]:

It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,

En este extracto, cada línea se considera como un documento por separado y el extracto completo es el corpus. Analizando el texto, se puede obtener un vocabulario para el modelo. Las palabras únicas que aparecen en el corpus son las siguientes:

- "it"
- "was"
- "the"
- "best"
- "of"
- "times"
- "worst"
- "age"
- "wisdom"
- "foolishness"

Se puede ver que el vocabulario está formado por 10 palabras (en donde se ignoran las mayúsculas y la puntuación), con un corpus compuesto por 24 palabras en total. Ahora cada documento del corpus debe ser transformado a un vector que contendrá su información.

Tomando en cuenta que el vocabulario tiene 10 palabras, se transforma cada documento en un vector de largo 10, en donde cada posición del vector corresponde al puntaje de una palabra del vocabulario dentro de ese documento. Un ejemplo de puntaje, es utilizar la aparición de la palabra en el documento: marcar un 1 si la palabra aparece y marcar un 0 si la palabra no aparece (sin importar cuántas veces aparece). Si se utiliza el orden del vocabulario presentado antes, es decir [it, was, the, best, of, times, worst, age, wisdom, foolishness], los vectores representantes de cada documento serían los siguientes:

1. "it was the best of times" = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0].
2. "it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0].
3. "it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0].

[4] Obtenido de *https://machinelearningmastery.com/gentle-introduction-bag-words-model/*
[5] Extracto del libro "A Tale of Two Cities", de Charles Dickens.

4. “it was the age of foolishness” = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1].

Si se tiene un corpus demasiado grande, el largo del vector que representa a cada documento va a ser bastante grande, lo que va a provocar que aparezcan vectores con muchos 0. Para reducir el vocabulario existen múltiples técnicas:

1. Ignorar mayúsculas y puntuación (como en el ejemplo anterior).
2. Ignorar palabras frecuentes pero que no entregan mayor información. Estas son conocidas como *stop words*, por ejemplo, “a”, “of”, “to”, etc.
3. Ignorar o corregir palabras con faltas de ortografía.
4. Reducir palabras a su raíz usando algoritmos de *stemming* o *lemmatization*, por ejemplo reducir “running” a “run”.

En el modelo de ejemplo presentado antes, se usaron como pesos la presencia de cada palabra en el documento. Existen otros métodos como contar la cantidad de veces que una palabra aparece en el documento, lo que permite darle mayor importancia a palabras frecuentes. Un problema de esto, es que aparecen palabras con alta frecuencia pero que no contienen información relevante para el modelo como otras palabras que pueden no ser tan frecuentes pero sí más específicas y relevantes dentro de un documento en particular.

Una forma de enfrentar el problema anterior, es reescalar la frecuencia de las palabras según qué tan frecuentes son en todos los documentos, de tal manera que los puntajes de las palabras como “the” sean penalizados. Esta técnica se llama Term Frequency - Inverse Document Frequency o TF-IDF. Term Frequency (TF) es la frecuencia de una palabra en el documento actual. Inverse Document Frequency (IDF) es el puntaje de qué tan escasa es la palabra a lo largo de todos los documentos. Con este método, se asigna un puntaje o peso a cada palabra, en donde no todas son igual de importantes. El puntaje resalta aquellas palabras que son específicas dentro de un documento pero que contienen información relevante.

El modelo Bag-of-Words tiene como ventaja que es sencillo de generar y entrega información importante acerca del contenido de un documento. Por otro lado, el vocabulario requiere de un diseño cuidadoso, para que cada vector tenga información importante de un documento. La gran dimensionalidad que los vectores pueden tener es una desventaja al entrenar una red neuronal con ellos. Finalmente, este modelo descarta información que puede ser relevante, como el orden de las palabras, el contexto y la gramática.

En W2AVV++ se utiliza BoW como un método de vectorización de texto, en conjunto a otros dos métodos. En el Capítulo 4 se presentan más detalles acerca de esto.

2.4.2. Word2Vec

El modelo Word2Vec fue propuesto el año 2013 por Mikolov et al. [29] y consiste en dos redes neuronales que producen representaciones vectoriales de palabras continuas. Los modelos son entrenados a partir del contexto lingüistico de las palabras, usando corpus de texto muy grandes y producen un espacio vectorial, donde cada palabra en el corpus es

representada con un vector en ese espacio. Los vectores que se encuentren cerca indican que sus palabras correspondientes comparten contextos similares.

El objetivo principal de Word2vec es desarrollar técnicas que puedan ser utilizadas para aprender vectores de palabras de alta calidad a partir de datasets con billones de palabras. Los autores proponen técnicas para medir la calidad de las representaciones vectoriales que word2vec produce, por ejemplo realizando operaciones algebraicas sobre los vectores. Ellos demuestran que *vector("Rey") - vector("Hombre") + vector("Mujer")* resulta en un vector que está muy cerca de la representación vectorial de la palabra "Reina".

Word2vec trata de maximizar el *accuracy* de estas operaciones vectoriales mientras se minimiza la complejidad computacional, la que es definida como el número de parámetros que la red debe aprender durante el entrenamiento y es proporcional a:

$$O = E \times T \times Q$$

en donde E es el número de *epochs* de entrenamiento, T es la cantidad de palabras en el set de entrenamiento y Q es un parámetro que se define para cada modelo en particular. Todos los modelos son entrenados usando Stochastic Gradient Descent y *backpropagation*.

Word2vec extiende un modelo llamado Feedforward Neural Network Language Model (NNLM) [4]. Este modelo consiste de las siguientes capas: input, proyección, oculta y output. En la capa de input, una cantidad N de palabras es codificada y luego proyectada hacia la capa de proyección, usando una matriz de proyección de dimensionalidad $N \times D$. La arquitectura NNLM es compleja computacionalmente entre la capa de proyección y la capa oculta, debido a que los valores en la capa de proyección son densos. La capa oculta es utilizada para calcular la distribución de probabilidad sobre todas las palabras del vocabulario. Luego, la complejidad correspondiente a cada ejemplar de entrenamiento se define como:

$$Q = N \times D + N \times D \times H + H \times V$$

en donde V es el tamaño del vocabulario y H es el tamaño de la capa oculta.

Mikolov propone dos arquitecturas para el aprendizaje de representaciones distribuidas de palabras que intentan minimizar la complejidad computacional. Ambas son entrenadas en dos pasos: primero, se aprenden vectores de palabras continuas usando un modelo simple y después se usa una NNLM N-grama para ser entrenada sobre las representaciones obtenidas.

Continuous Bag-of-Words

Es similar a la red NNLM, pero se remueve la capa oculta y la capa de proyección es compartida para todas las palabras (no solo en la matriz de proyección). Se construye un clasificador *log-linear* utilizando cuatro palabras siguientes y cuatro palabras anteriores a la palabra actual, en donde el entrenamiento consiste en clasificar correctamente esta palabra. Esto convierte la complejidad del entrenamiento en:

$$Q = N \times D + D \times \log_2(V)$$

Esta arquitectura es llamada CBOW porque utiliza representaciones continuas distribui-

das del contexto de las palabras, donde el orden de estas no influencia a la proyección. Se puede observar esta arquitectura en la Figura 2.17.

Continuous Skip-gram

Cada palabra actual funciona como un input a un clasificador *log-linear* con una capa de proyección continua, y luego se predicen palabras dentro de un rango antes y después de la palabra actual. Al aumentar este rango, se mejora la calidad de los vectores que se obtienen pero también aumenta la complejidad computacional. Se asume que las palabras más lejanas a la palabra actual están menos relacionadas a esta, por lo que se les da menos peso al obtener muestras durante el entrenamiento. La complejidad del entrenamiento en esta arquitectura es proporcional a:

$$Q = C \times (D + D \times \log_2(V))$$

en donde C es la distancia máxima entre las palabras. Esto significa que si por ejemplo, se elige $C = 5$, para cada palabra de entrenamiento se elige de manera aleatoria un número R en el rango [1,C] y luego se usan R palabras hacia atrás y hacia adelante con respecto a la palabra actual como si fueran etiquetas correctas.

Sentence Embedding

Word2vec genera descriptores que representan a cada palabra de una frase en un espacio vectorial común. Para obtener un descriptor que represente a la frase completa, una estrategia simple es mezclar los descriptores de cada palabra aplicando *mean pooling* sobre estos.

Este modelo también es utilizado como un método de vectorización de texto en W2AVV++, lo que será detallado en el Capítulo 4.

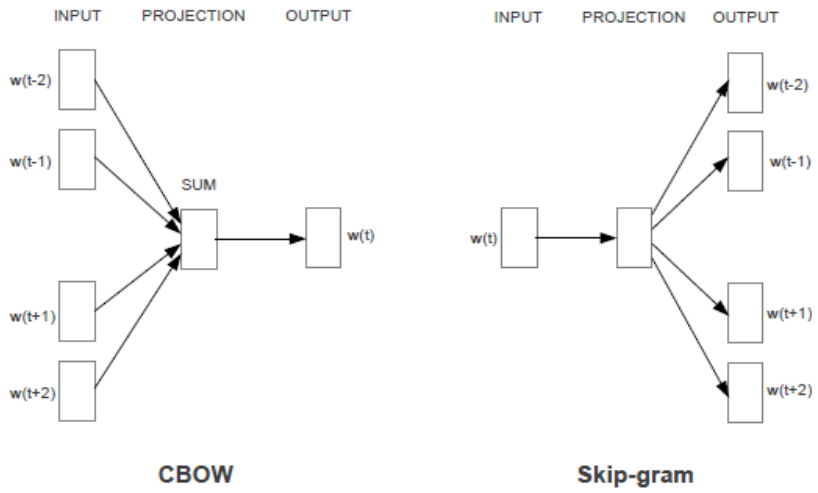


Figura 2.17: Ilustración de las arquitecturas que componen a Word2Vec. **Izquierda**: Modelo CBOW, clasifica la palabra actual con respecto a su contexto. **Derecha**: Modelo Skip-gram, predice las palabras que rodean a la palabra actual con respecto a esta misma.

2.4.3. Redes Neuronales Recurrentes

Las redes neuronales recurrentes o RNN por sus siglas en inglés (Recurrent Neural Networks), están basadas en el trabajo de Rumelhart et al. [34], aunque se hicieron más populares con el paso del tiempo y con el desarollo de las redes Long Short-Term Memory (LSTM) por Hochreiter and Schmidhuber [18].

Las RNN son una clase particular de redes neuronales cuya principal característica es la de tener una memoria interna. Esta memoria interna o estado interno, le permite a la red procesar secuencias de inputs y mantener guardada información que puede ser relevante para los inputs futuros. Este comportamiento hace que sean ideales para aplicarlas en campos como reconocimiento del habla, análisis de series temporales, entre otros.

Una RNN consiste de un estado oculto h y un output opcional y que opera en una secuencia de largo variable $x = (x_1, \ldots, x_T)$. En cada paso temporal t, el estado oculto $h_{<t>}$ de la RNN es actualizado según la función:

$$\mathbf{h}_{<t>} = f(\mathbf{h}_{<t-1>}, x_t)$$

en donde f es una función no lineal de activación. La idea detrás de una RNN es que es similar a una red neuronal *feed forward* normal, pero con la diferencia de que cada capa corresponde a un período temporal distinto y además se guarda un estado de la red.

En casos en donde existan secuencias demasiado largas o pasos temporales muy grandes, la RNN va a necesitar guardar información durante muchos pasos. Esto causa que sea una red difícil de entrenar y que además sufra de *vanishing gradient*. Para enfrentar este problema, fueron creadas las Gated Recurrent Units (GRU).

2.4.4. Gated Recurrent Unit

Las Gated Recurrent Units o GRU, fueron creadas por Cho et al. [6]. Estas unidades ocultas fueron propuestas como una mejora de las RNN y están inspiradas en el funcionamiento de las LSTM. Una GRU funciona como una célula que contiene múltiples operaciones, que son llamadas compuertas o *gates*. En particular, están compuestas por una *update gate* y una *reset gate*. La activación de una unidad oculta j depende de estas dos compuertas, donde la *reset gate* r_j se calcula como:

$$r_j = \sigma([\mathbf{W}_r x]_j + [\mathbf{U}_r \mathbf{h}_{<t-1>}]_j)$$

en donde σ es la función sigmoid y $[.]_j$ denota el elemento número j de un vector. Las variables x y $\mathbf{h}_{t-1}$ son el input y el estado oculto anterior respectivamente. $\mathbf{W}_r$ y $\mathbf{U}_r$ son las matrices de pesos que son aprendidas por la red. Luego, la *update gate* es calculada por:

$$z_j = \sigma([\mathbf{W}_z x]_j + [\mathbf{U}_z \mathbf{h}_{<t-1>}]_j)$$

La activación de la unidad h_j es calculada por:

$$h_j^{<t>} = z_j h_j^{<t-1>} + (1 - z_j)\tilde{h}_j^{<t>}$$

$$\tilde{h}_j^{<t>} = \phi([\mathbf{W}x]_j + [\mathbf{U}(\mathbf{r} \odot \mathbf{h}_{<t-1>})]_j)$$

Cuando la *reset gate* es cercana a 0, el estado oculto de la red es forzado a ignorar el estado oculto anterior y se reinicia tomando solo en cuenta el input actual. Esto permite al estado oculto olvidar cualquier información que es determinada como irrelevante en el futuro, por lo que se produce una representación más compacta.

Por otro lado, la *update gate* controla qué tanta información del estado oculto anterior es traspasada al estado oculto actual. Esto le permite a la RNN recordar información a largo plazo. Dado que cada unidad oculta tiene sus propias *reset gates* y *update gates*, cada unidad oculta es capaz de aprender a capturar dependencias sobre escalas temporales distintas. Aquellas unidades que aprendan a capturar dependencias de corto plazo van a tender a tener *reset gates* que se activan de manera frecuente, por el contrario, aquellas unidades que aprendan a capturar dependencias de largo plazo van a tener *update gates* que se activan de manera frecuente. Se puede ver una ilustración de una GRU en la Figura 2.18.

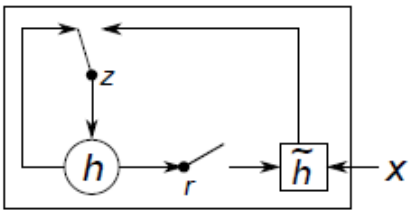


Figura 2.18: Ilustración de una GRU. La *update gate* z decide si el estado oculto va a ser actualizado con un nuevo estado oculto $\tilde{h}$. La *reset gate* r decide si es que el estado oculto anterior es ignorado o no.

Las características de una GRU permiten trabajar con secuencias de texto y aprender sobre el contexto de las palabras. Este modelo es utilizado como otro método de vectorización de texto en W2AVV++, lo que será detallado en el Capítulo 4.

2.5. Métricas Importantes

El sistema buscador de videos W2AVV++ entrega como resultado una lista ordenada de videos según su similitud con la consulta realizada, con el mejor resultado en la primera posición. Para analizar los resultados obtenidos, se utilizan ciertas métricas relevantes en el campo de la recuperación de información. Estas métricas tienen relación con la calidad de los resultados obtenidos, es decir, si los videos recuperados son relevantes para la consulta. Se define el Ground Truth de una consulta como los videos que son relevantes para esta.

Recall@K

Recall es la proporción de videos que son relevantes con respecto a la consulta realizada:

$$\text{recall} = \frac{|\{\text{videos relevantes}\}| \bigcap |\{\text{videos recuperados}\}|}{|\{\text{videos relevantes}\}|}$$

Recall@K (R@K) es la proporción de videos relevantes recuperados dentro de las primeras K posiciones de la lista. Por ejemplo, si con una consulta obtengo un R@10 igual a 0,2 significa que un 20 % del total de videos relevantes aparecen en los top-10 resultados de la consulta. Para evaluar un sistema se promedian los R@K de todas las consultas realizadas y se obtiene uno general.

Precision@K

Precision es la porción de videos recuperados por una consulta que son relevantes:

$$\text{precision} = \frac{|\{\text{videos relevantes}\}| \bigcap |\{\text{videos recuperados}\}|}{|\{\text{videos recuperados}\}|}$$

Precision@K (P@K) es la porción de videos recuperados que son relevantes en las primeras K posiciones. Es decir, si con una consulta obtengo un P@10 de 50 %, la mitad de los videos que recuperé en los primeros 10 resultados son relevantes. Para evaluar un sistema se promedian los P@K de todas las consultas realizadas y se obtiene uno general.

Mean Average Precision

Average Precision es una métrica que combina el recall y el precision de los resultados. Para una consulta, el average precision es el promedio del precision obtenido para cada elemento relevante recuperado. A continuación, se presenta un ejemplo:

Sea la consulta Q: "videos en donde aparezca un perro", y un conjunto G de videos en una base de datos. Para esta consulta se obtiene una lista de videos ordenados del más al menos relevante, es decir, un ranking. A modo de ejemplo, sea G un conjunto de 10 videos en donde los 3 videos de la Figura 2.19 son el Ground Truth para la consulta Q.

Figura 2.19: Ground Truth de la consulta Q

Se obtiene un ranking de 4 videos ordenados, como se puede ver en la Figura 2.20. Se calcula el precision en las posiciones donde hay un video relevante y si no se encuentra un video que sea *ground truth* se obtiene un valor de 0, para luego calcular el promedio de esto obteniendo el Average Precision (AP) de la consulta.

Mean Average Precision (mAP) es una métrica utilizada cuando existen múltiples consultas. Para cada consulta Q, se calcula su Average Precision (AP) correspondiente y luego el mAP se obtiene calculando el promedio entre todas las consultas realizadas:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i$$

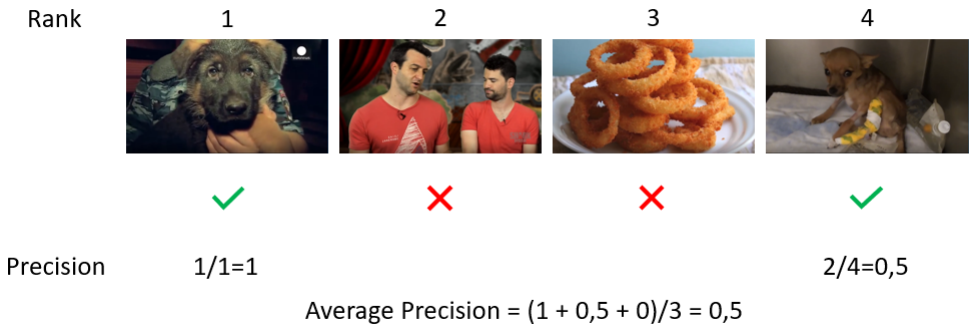


Figura 2.20: Videos recuperados por la consulta Q y el Average Precision obtenido.

Estas métricas son importantes ya que entregan información útil acerca del desempeño de las consultas realizadas. Mientras mayor sea el R@K, el P@K y el mAP obtenido por un sistema, significa que obtiene mejores resultados. Los resultados obtenidos por W2AVV++ son presentados en el Capítulo 5.

Capítulo 3

Revisión Bibliográfica

En este capítulo se detallan dos modelos importantes desarrollados para enfrentar el problema de la recuperación de videos sin etiquetar. El primero es W2VV [10], que fue el ganador de la competencia Matching and Ranking de la competencia Video-to-Text Description en la conferencia TRECVID 2016 [38].

El segundo modelo es W2VV++ [25], el que presenta una mejora al sistema W2VV original, fue el sistema ganador de la competencia Ad-Hoc Video Search en la conferencia TRECVID 2018 y además corresponde al estado del arte actual con respecto a la recuperación de videos sin etiquetar. Este sistema sirve como base para la propuesta del modelo W2AVV++ de este trabajo de investigación.

3.1. Word2VisualVec (W2VV)

W2VV fue creado con la finalidad de predecir descriptores visuales a partir del texto, logrando además importantes resultados en el problema de Video-to-Text en donde se generan descripciones textuales del contenido de los videos. Este sistema consiste en un modelo de redes neuronales profundas entrenado para proyectar una descripción textual de un video hacia el espacio de descriptores visuales. En este espacio, la similitud entre un video x y una descripción q puede ser calculada como una distancia entre descriptores de un mismo espacio.

Formalmente, sea $\phi(x) \in \mathbb{R}^d$ un descriptor visual de d dimensiones, extraído usando ConvNets pre-entrenadas, y sea $r(q) \in \mathbb{R}^d$ un descriptor textual generado a partir de la descripción q del video x. Se calcula la similitud entre ambos descriptores usando la similitud coseno entre $\phi(x)$ y $r(q)$. La red es entrenada con el objetivo de minimizar el error cuadrático medio (Mean Squared Error o MSE) entre los descriptores.

Para generar los descriptores de texto se usa una mezcla de 3 técnicas: BoW, word2vec y una RNN con GRU. Con respecto al uso de BoW, en este caso cada dimensión de un vector BoW corresponde a que aparezca una palabra específica en la frase input:

$$s_{bow}(q) = (c(w_1, q), c(w_2, q), \ldots, c(w_m, q))$$

donde $c(w, q)$ corresponde a la aparición de la palabra w en q y m es el tamaño del vocabulario.

El modelo word2vec utilizado en W2VV tiene 500 dimensiones y es entrenado con un corpus de texto formado por tags en inglés de 30 millones de imágenes de Flickr, utilizando la arquitectura *skip-gram*. Esto resulta en un vocabulario de 1,7 millones de palabras en total. Se obtiene un descriptor word2vec de cada descripción al realizar *mean pooling* sobre cada una de sus palabras:

$$s_{word2vec}(q) := \frac{1}{|q|} \sum_{w \in q} v(w)$$

donde $v(w)$ corresponde al *embedding* de una palabra individual y $|q|$ es el largo de la frase.

Finalmente, se utiliza una GRU para crear un tercer descriptor a partir del texto. Para inicializar los pesos de la red se reutiliza el modelo word2vec entrenado con los tags de Flickr. El último estado oculto de la GRU se toma como la vectorización de la frase, la que tiene 1.024 dimensiones.

Luego, la vectorización múltiple se obtiene al concatenar las tres representaciones mencionadas:

$$s_q = [s_{bow}(q), s_{word2vec}(q), h_{|q|}]$$

El vector $s(q)$ pasa a través de una red MLP hasta llegar a la capa de salida $r(q)$, la que se encuentra en el espacio de descriptores visuales. Para entrenar la red, se utiliza Mean Squared Error (MSE) como función de pérdida:

$$l_{mse}(x, q; \theta) = (r(q) - \phi(x))^2$$

W2VV se entrena con el objetivo de minimizar la pérdida MSE global dado un set de entrenamiento $D = (x, q)$ formado por un conjunto de pares video-frase:

$$\arg\min_{\theta} \sum_{(x,q) \in D} l_{mse}(x, q; \theta)$$

El sistema utiliza cuatro ConvNets pre-entrenadas distintas para generar los descriptores visuales: CaffeNet [20], GoogLeNet [40], GoogLeNet-shuffle [28] y ResNet-152 [16]; además de una ConvNet 3-D [42]. Se realiza *mean pooling* sobre los cuadros de un mismo video y así se obtiene un descriptor por video.

Los experimentos realizados con estos modelos pre entrenados en W2VV consideran el uso de solo una CNN por cada entrenamiento, comparando luego el desempeño del modelo al usarlos por separado. Entre todos los nombrados, al usar ResNet se obtuvieron los mejores resultados. En la Figura 3.1 se puede ver una ilustración del funcionamiento de W2VV:

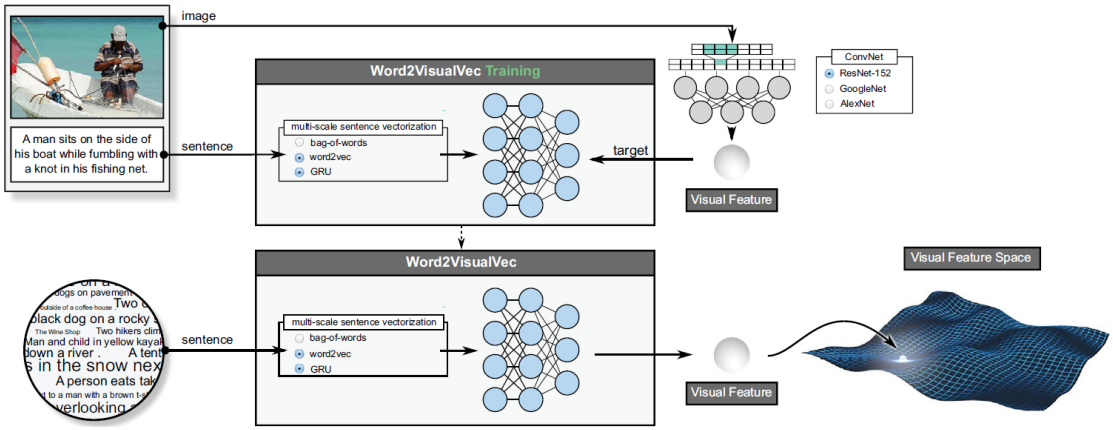


Figura 3.1: Ilustración de la arquitectura de W2VV. El modelo vectoriza una frase como input al usar Bag-of-Words, word2vec y una GRU. El vector pasa a través de una MLP para predecir un descriptor visual obtenido por una ConvNet como GoogleNet o ResNet.

3.2. Word2VisualVec++ (W2VV++)

W2VV++ es una versión mejorada de W2VV, propuesta por Li et al. [25] y fue el modelo ganador de la competencia Ad-hoc Video Search en TRECVID 2018 [1]. El objetivo de este modelo es el mismo que el modelo original, es decir, predecir descriptores visuales a partir del texto.

Este modelo cambia la función de pérdida utilizada en W2VV por una Triplet Ranking Loss [12]. Esta función utiliza la distancia en el espacio vectorial entre una frase y su video correspondiente, y la distancia entre la misma frase y el video más lejano en ese espacio dentro de un conjunto dado. Con esto logra que la red aprenda cuáles son los videos negativos para una frase ya que se encuentran a mayor distancia. Dado un par video-descripción (v, s), la función de pérdida es definida como:

$$l(v, s; \theta) = \max_{v}(0, \alpha + S_\theta(v^-, s) - S_\theta(v, s))$$

en donde v^- es el *hardest negative video sample* de la descripción s. Durante el entrenamiento, el *hardest negative video sample* se define como el video más distinto a la frase s dentro de un mini-batch. La función de pérdida tiene como objetivo maximizar la distancia con los videos que sean distintos a s y minimizar la distancia con aquellos que sean similares. Se utiliza como función de similitud para este cálculo la similitud coseno $S_\theta(v, s)$ entre $\phi(v)$, que representa a un descriptor visual, y $r(s)$, que representa a un descriptor de texto.

Para la representación del texto, W2VV++ reutiliza la vectorización múltiple propuesta por W2VV. Es decir, dada una descripción de texto, esta es vectorizada en paralelo por tres estrategias de vectorización: Bag-of-Words, word2vec y GRU. Cada una de estas vectorizaciones es concatenada y se forma un solo vector. Este vector es entregado a una MLP que proyecta hacia el espacio de descriptores visuales.

Para la representación de los videos, se sigue una estrategia similar a la usada en W2VV, donde se utilizan modelos pre-entrenados de ConvNets como extractores de características sobre los cuadros de un video y luego se promedian para obtener un descriptor por video. Se obtienen cuadros de un video cada 0,5 segundos, para después extraer un descriptor visual de cada cuadro utilizando los modelos ResNet-152 [9] y ResNeXt-101 [28]. Cada uno de estos descriptores es de 2.048 dimensiones, los que luego son promediados usando *mean pooling* sobre todos los descriptores de los cuadros de un video, generando 2 descriptores de 2.048 dimensiones por video (cada uno correspondiente a cada ConvNet mencionada).

Para la competencia en TRECVID, los autores probaron distintos modelos variando el descriptor visual utilizado. Por ejemplo, un modelo que utiliza solo ResNeXt-101 y otro que utiliza ambos descriptores, concatenando ambos por cada video y luego utilizando una capa *fully connected* para realizar *feature re-learning* [11].

La técnica *feature re-learning* consiste en utilizar una nueva capa *fully connected* para procesar los descriptores obtenidos por cada video y aprender un nuevo descriptor. Para realizar esto, se proyectan los descriptores originales ya obtenidos por ResNet y ResNeXt hacia un nuevo espacio vectorial. Cada nuevo descriptor es representado por:

$$\phi(v) = Wv + b,$$

donde v es un descriptor de entrada de [N,1], W es una matriz afín de [M,N] inicializada con el método Xavier [14] y b es el *bias* de [M,1]. El modelo que obtuvo mejores resultados fue el que utiliza ResNet y ResNeXt en conjunto con *feature re-learning*, el que se puede ver en la Figura 3.2.

El modelo W2VV++ es entrenado utilizando dos datasets: MSR-VTT [44] y TGIF [26].

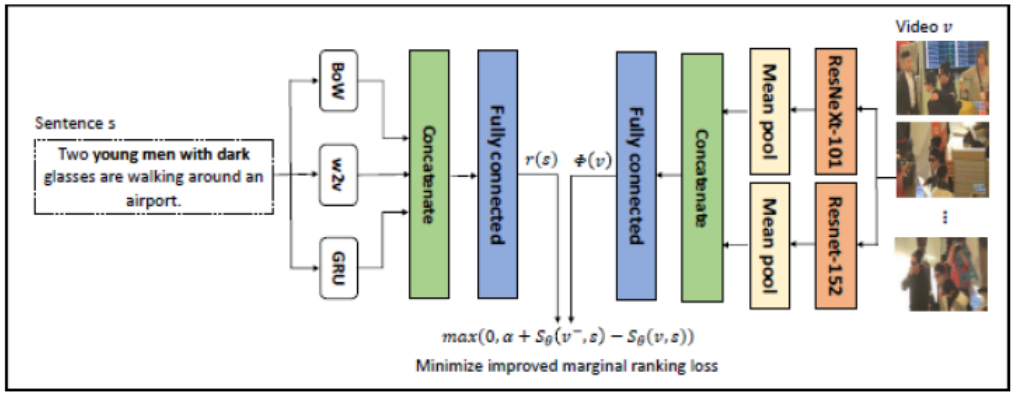


Figura 3.2: Ilustración de la arquitectura de W2VV++. Se obtiene un descriptor ResNet y otro ResNeXt por video, luego se concatenan y pasan por una capa *fully connected* para obtener un descriptor visual final. La descripción es vectorizada y luego proyectada hacia el espacio visual usando un MLP.

Capítulo 4

Descripción e Implementación del Sistema

4.1. Descripción General del Sistema

En esta investigación se propone una extensión del modelo W2VV++ que llamaremos Word2AudioVisualVec++ o W2AVV++. En este trabajo se desea comprobar si agregar un componente auditivo al modelo mejora la eficacia al recuperar videos.

El modelo W2AVV++ utiliza el mismo mecanismo de vectorización multiple utilizado en W2VV++, es decir, se usa una combinación de las técnicas BoW, word2vec y GRU. También se reutilizan los modelos de ConvNets pre-entrenados ResNet-152 y ResNeXt-101 para la representación visual de los videos, y se usa la misma función de pérdida Triplet Ranking Loss. La novedad del modelo propuesto radica en el uso de descriptores auditivos extraídos desde las pistas de audio mediante el uso de ConvNets pre-entrenadas y dos estrategias de combinación de descriptores unimodales: Early Fusion e Intermediate Fusion.

4.2. Dataset Utilizado

Para el entrenamiento del modelo propuesto se utiliza el dataset MSR-VTT [44] (que también fue utilizado por el modelo W2VV++). El dataset consiste de 10.000 videos provenientes de YouTube con un total 41,2 horas de video. Cada video tiene 20 descripciones, realizadas por 1.317 trabajadores de Amazon Mechanical Turk (AMT). Cada descripción fue realizada por un grupo de trabajadores luego de mirar el video. Esto produce un total de 200.000 pares video-descripción, con un total de 1.856.523 palabras y un vocabulario de tamaño 29.316. Los videos son separados en 20 categorías distintas, cuya distribución se puede observar en la Figura 4.1.

Uno de los problemas que se presentaron con este dataset, es que algunos de los videos ya no se encuentran disponibles en YouTube debido a que han sido deshabilitados o porque infringían derechos de autor. El dataset utilizado finalmente quedó reducido a un total de 8.006 videos, con un total de 160.120 pares video-descripción. Se utiliza una división de un 90 % del dataset para entrenar el modelo y un 10 % para realizar la validación en cada *epoch*. Para realizar testing sobre el modelo entrenado, se utiliza parte del dataset de testing

propuesto por Xu et al. [44] en el "Microsoft Multimedia Challenge"[1]. Finalmente, la cantidad de videos fue: 7.205 videos para entrenamiento, 801 videos para validación y 1.000 videos para testing.

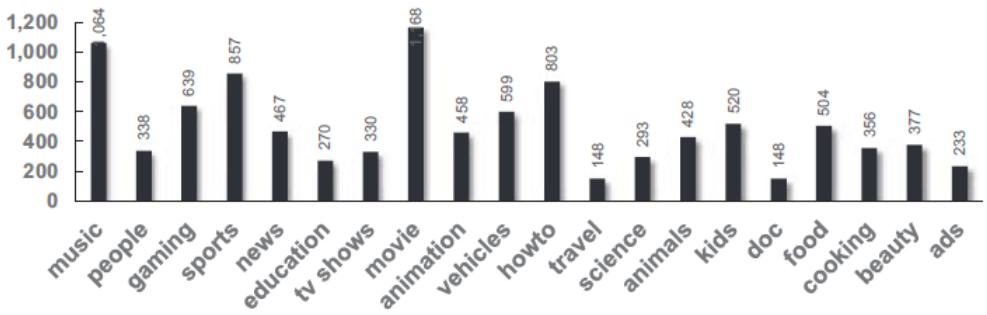


Figura 4.1: Distribución de categorías de videos del dataset MSR-VTT propuesto por Xu et al. [44].

El modelo W2VV++ utiliza además el dataset TGIF [26], pero para el modelo propuesto W2AVV++ no es considerado puesto que consiste de archivos en formato ".gif" sin pista de audio.

4.3. Representación Visual

Se utiliza la misma estrategia que en W2VV++, usando dos modelos ConvNets pre-entrenados como extractores de características: ResNet-152 y ResNeXt-101. Por cada video, se extrae un cuadro cada 0,5 segundos y para cada uno se extrae un descriptor ResNet y otro ResNeXt, ambos de 2.048 dimensiones. Luego, se toman todos los descriptores ResNet de un video y se realiza *mean pooling* sobre ellos, generando un descriptor ResNet por cada video. Esta misma estrategia se realiza para obtener un descriptor ResNeXt por cada video.

Para extraer los descriptores usando las ConvNets mencionadas, se usa la capa "flatten0" que se encuentra antes de la última capa *fully connected* y luego de la última capa de *pooling* ya que en este punto del modelo se pueden obtener descriptores semánticos de los cuadros antes de ser mapeados a las etiquetas de esa red en particular.

Con respecto a los modelos utilizados, estos fueron pre-entrenados con el dataset ImageNet pero no de la misma forma. ResNet-152 fue entrenado con una versión de ImageNet con 10 millones de imágenes correspondientes a un total de 10 mil clases. Por otro lado, ResNeXt-101 fue entrenado con "ImageNet Shuffle" [28], que consiste en una reorganización de ese dataset con cerca de 13 mil clases.

4.4. Representación Auditiva

Se utilizan dos modelos pre-entrenados: SoundNet y AENet. En el caso de SoundNet se usa una versión que tiene 8 capas convolucionales y se selecciona la capa 24 como extractor

[1] http://ms-multimedia-challenge.com/2017/

de descriptores. Esta es una capa de pooling que genera una matriz de [T, D], en donde T es relativo al largo de la pista de audio y D corresponde a la dimensionalidad del descriptor, en este caso 1.024. Para obtener un vector con largo fijo, se realiza *max pooling* por las columnas de la matriz y se obtiene un descriptor de [1,D]. En la Figura 4.2 se pueden observar visualizaciones[2] de videos que tienen sonidos similares.

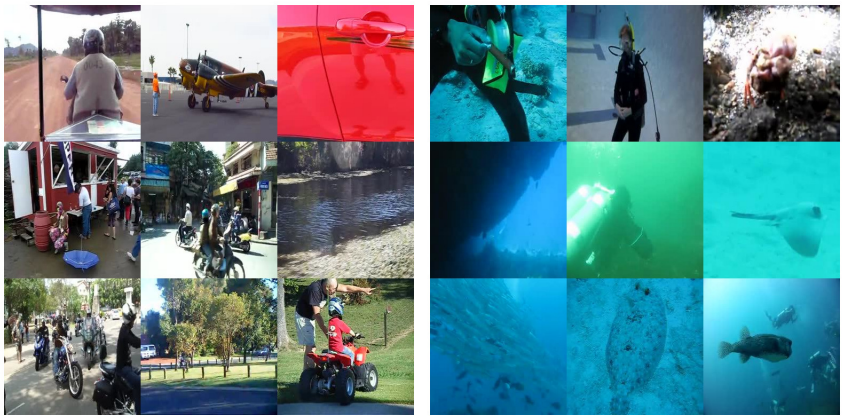


Figura 4.2: Visualizaciones de conceptos semánticos que SoundNet aprende a través del sonido. **Izquierda**: Conceptos relacionados a sonidos de motor. **Derecha**: Conceptos relacionados a sonidos "bajo el agua".

Con respecto a AENet, se utiliza el mismo modelo pre-entrenado descrito en Takahashi et al. [41] y se usa la última capa *fully connected* como extractor. Esta capa genera una matriz de [T, D] similar a la generada por SoundNet, en donde nuevamente se aplica *max pooling* a lo largo de la variable temporal y se obtiene el descriptor AENet final de 1.024 dimensiones.

4.5. Representación de Texto

Para la representación del texto, W2AVV++ utiliza el mismo mecanismo de vectorización múltiple propuesto por W2VV++. Se utiliza BoW con un vocabulario de tamaño 8.645, obtenido con las mismas descripciones de los videos, eliminando las *stop words*. Se utiliza el mismo modelo word2vec usado en W2VV++ de 500 dimensiones y la misma GRU de 1.024 dimensiones. Las tres técnicas generan descriptores que son concatenados, entregando un descriptor de 9.669 dimensiones, el que luego es proyectado hacia el espacio de descriptores de video a través de un MLP que entrega un vector de 2.048 dimensiones.

4.6. Combinación de descriptores unimodales

Un esquema Early Fusion fusiona las modalidades en el espacio de los descriptores [37]. Consiste en primero extraer los descriptores unimodales de manera individual antes de realizar cualquier proceso de aprendizaje. Se unen los descriptores con alguna estrategia, usualmente concatenación, y finalmente se realiza un proceso de entrenamiento y aprendizaje con el

[2] http://soundnet.csail.mit.edu/

descriptor multimodal. Una de las ventajas de este tipo de esquemas es la necesidad de realizar solo una fase de entrenamiento para obtener el descriptor multimodal. Por otro lado, una de sus desventajas es la dificultad que tiene para combinar descriptores de diferentes modalidades en una representación común. En la Figura 4.3 se puede observar una ilustración de este esquema.

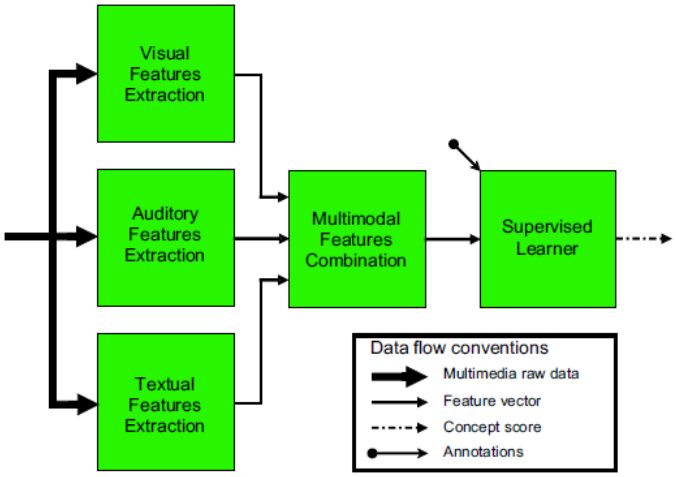


Figura 4.3: Ilustración del esquema Early Fusion. Se realiza la combinación de descriptores unimodales antes de que se aprendan conceptos por el sistema.

Siguiendo la estrategia Early Fusion, por cada video se generan 4 descriptores, los que son concatenados generando un descriptor audiovisual de 6.144 dimensiones. Este es luego procesado por una capa *Fully Connected* para realizar *feature re-learning*. La salida de esta capa genera un descriptor de video de 2.048 dimensiones, el que es representado en la Figura 4.4 por $\phi(v)$. Para la representación del texto, se utiliza la vectorización múltiple generada por la concatenación de un descriptor generado por BoW, word2vec y GRU. Este descriptor es proyectado hacia el espacio del descriptor de video al igual que en el modelo W2VV++.

Por otro lado, se propone el uso de un esquema Intermediate Fusion en el que se extraen los descriptores unimodales pero antes de unirlos, primero se realiza una operación sobre los descriptores de cada modalidad por separado aplicando la técnica *feature re-learning*. Para realizar esto, luego de obtener los descriptores unimodales, se usa una rama visual en donde se concatenan los descriptores visuales y luego se aplica *feature re-learning* sobre esta concatenación, para generar un nuevo descriptor visual. Los pesos aprendidos en este proceso son guardados, para luego ser utilizados al momento de procesar los videos de prueba. Lo mismo se realiza con una rama auditiva. Después de tener los dos nuevos descriptores unimodales por video, estos se concatenan y se vuelve a aplicar *feature re-learning* sobre ellos para finalmente obtener un descriptor audiovisual. La ventaja de un esquema Intermediate Fusion es que es capaz de aprender algunas características presentes en cada modalidad por separado, sin la necesidad de entrenar un clasificador por modalidad.

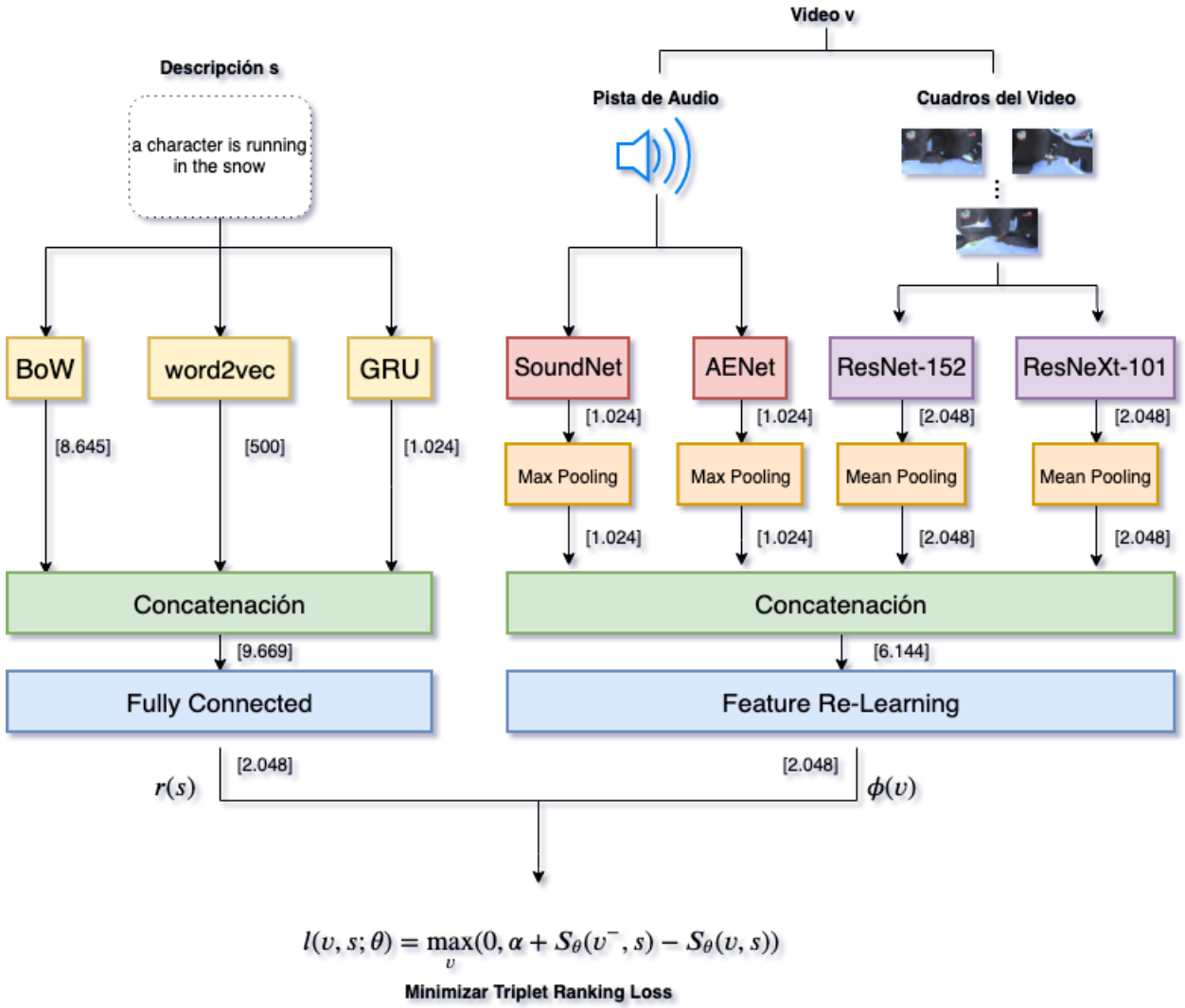


Figura 4.4: Representación del modelo W2AVV++ utilizando un esquema Early Fusion. Se extraen descriptores para cada modalidad y se combinan en un solo descriptor audiovisual.

Siguiendo la estrategia Intermediate Fusion, se extraen los mismos 4 descriptores que en el esquema Early Fusion. En este caso, los descriptores auditivos son concatenados en una rama auditiva, generando un descriptor de 2.048 dimensiones, el que luego es procesado por una capa *fully connected* y se retorna un nuevo descriptor auditivo de 2.048 dimensiones. Por otro lado, en la rama visual los descriptores visuales son concatenados generando un descriptor de 4.096 dimensiones, el que luego es procesado por una capa *fully connected* y se retorna un nuevo descriptor auditivo de 2.048 dimensiones. Las salidas de ambas ramas son concatenadas, generando un descriptor de 4.096 dimensiones y usando una nueva capa *fully connected* se realiza otro proceso de entrenamiento que entrega un descriptor de video de 2.048 dimensiones, representado por $\phi(v)$ en la Figura 4.5. Para la representación de texto se sigue la misma estrategia que en el esquema Early Fusion.

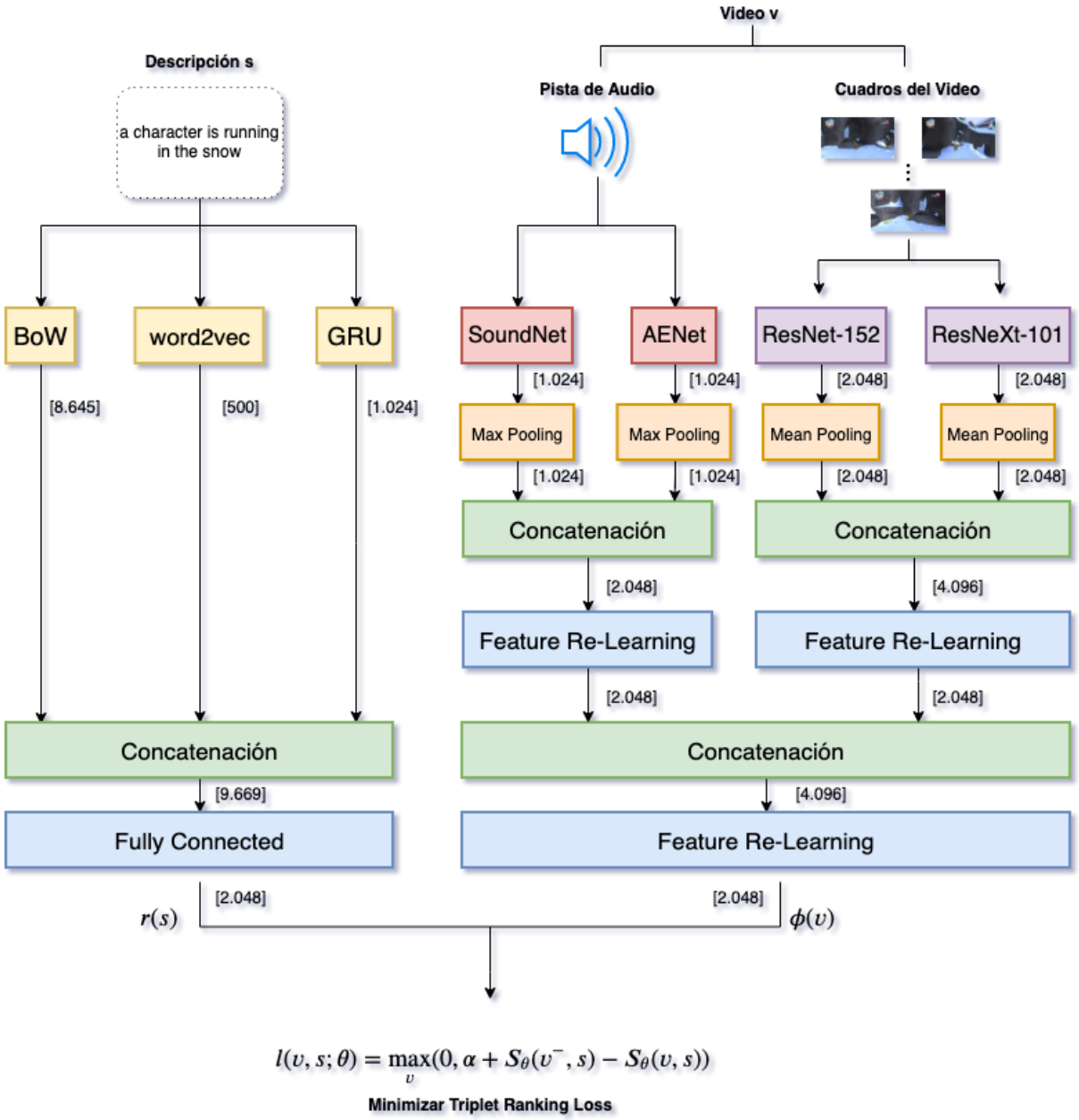


Figura 4.5: Representación del modelo W2AVV++ utilizando un esquema Intermediate Fusion. Se extraen descriptores para cada modalidad, pero se combinan los descriptores por modalidad separados. Se realiza *feature re-learning* sobre la rama auditiva y la rama visual, donde luego la salida de ambos es combinada para volver a aplicar *feature re-learning* y así generar un descriptor audiovisual.

Capítulo 5

Resultados

En este capítulo se presentan experimentos realizados con W2AVV++ y sus resultados. Los experimentos consideran comparar tres modelos en paralelo:

- **W2VV++**: Se utiliza como *baseline*, entrenado con MSR-VTT usando solo descriptores visuales.
- **Early Fusion W2AVV++**, entrenado con MSR-VTT.
- **Intermediate Fusion W2AVV++**, entrenado con MSR-VTT.

Se entrenan estos tres modelos por separado utilizando diferentes configuraciones con los siguientes hiperparámetros:

1. **Función de Pérdida**: MSE vs Triplet Ranking Loss
2. **Learning Rate**: 0,0001 vs 0,001
3. **Función de Optimización**: RMSprop vs Adam
4. **Batch Size**: 100 vs 128
5. **Triplet Ranking Loss Alpha Margin** (α): 0,2 vs 0,5

Los tres modelos se entrenan con un máximo de 100 *epochs* utilizando *early stopping*, lo que significa que se va guardando el *validation accuracy* obtenido al final de cada *epoch* y si este valor no mejora durante las siguientes 10 *epochs*, el entrenamiento se detiene. El valor del *learning rate* inicial se mantiene en cada *epoch* mientras el *validation accuracy* aumenta, si esto no ocurre durante dos *epochs* entonces se disminuye el *learning rate* a la mitad.

En la Tabla 5.1 se puede observar la distribución de cada conjunto utilizado para entrenar, validar y testear los tres modelos. El entrenamiento de los modelos fue realizado en un servidor con S.O. Ubuntu 16, con un procesador Intel Core i7-4770K de 4 núcleos físicos @ 3.50 GHz, 8 hilos, 32 GB de memoria RAM y una GPU Nvidia GeForce GTX 980 Ti de 6 GB.

Para evaluar los modelos ya entrenados, se deben obtener los descriptores de cada video del conjunto de test. Es decir, por cada video se obtiene un descriptor visual, un descriptor audiovisual con Early Fusion W2AVV++ y un descriptor audiovisual con Intermediate Fusion

W2AVV++. Estos descriptores funcionan como el Ground Truth de cada descripción, donde cada modelo debe predecir el video Y correspondiente a cada descripción X que se le pasa a cada uno. Por cada descripción X, cada modelo genera un descriptor $r(X)$ al que se le calcula la similitud coseno con los descriptores de todos los videos $\phi(Y)$, y luego se ordenan estas similitudes en orden descendente para obtener un ranking.

	Conjunto de Entrenamiento	Conjunto de Validación	Conjunto de Test
# Videos	7.205	801	1.000
# Pares video-descripción	144.100	16.020	20.000

Tabla 5.1: Tamaño de cada conjunto de datos utilizado para los experimentos

Como a cada descripción le corresponde solo un video como Ground Truth, la métrica R@K calcula el porcentaje de descripciones para las cuales el resultado correcto es encontrado dentro de los primeros K videos recuperados. Es importante notar que para una descripción podrían existir más videos relevantes, pero para mantener la simplicidad de los experimentos no se considera ese caso.

5.1. Experimentos sobre conjunto de test

Experimento 1: Función de Pérdida

Se entrenan los tres modelos usando como función de pérdida MSE y Triplet Ranking Loss, dejando los demás hiperparámetros fijos: *batch size* 100, *alpha margin* de 0,5 en Triplet Ranking Loss, *learning rate* de 0,0001 y usando RMSprop. Los resultados obtenidos por cada modelo con el conjunto de test se pueden ver en la Tabla 5.2, donde se muestran los valores en las métricas R@K y mAP.

	MSE				Triplet Ranking Loss			
	R@1	R@5	R@10	mAP	R@1	R@5	R@10	mAP
W2VV++	1,7 %	7,0 %	10,4 %	0,049	**10,0 %**	**25,6 %**	**36,2 %**	**0,181**
Early W2AVV++	1,2 %	4,1 %	7,0 %	0,034	**9,9 %**	**27,5 %**	**37,3 %**	**0,186**
Intermediate W2AVV++	1,3 %	4,3 %	7,2 %	0,036	**10,1 %**	**28,0 %**	**38,4 %**	**0,193**

Tabla 5.2: Resultados del Experimento 1: Función de Pérdida

Se observa claramente que Triplet Ranking Loss (TRL) obtiene mejores resultados que MSE. Es importante notar que en general, ambos esquemas de W2AVV++ obtienen mejores resultados que el modelo W2VV++ que es solo visual. Cabe destacar que el esquema Intermediate Fusion obtiene levemente mejores resultados que el Early Fusion con esta configuración de hiperparámetros.

Experimento 2: Learning Rate

Se entrenan los tres modelos con dos valores de *learning rate*, dejando los demás hiperparámetros fijos: *batch size* 100, Triplet Ranking Loss con *alpha margin* de 0,5 y usando RMSprop. Los resultados se pueden ver en la Tabla 5.3.

	Learning Rate = 0,0001				Learning Rate = 0,001			
	R@1	R@5	R@10	mAP	R@1	R@5	R@10	mAP
W2VV++	10,0 %	25,6 %	36,2 %	0,181	**10,5 %**	**27,7 %**	**38,9 %**	**0,194**
Early W2AVV++	9,9 %	27,5 %	37,3 %	0,186	**11,1 %**	**28,3 %**	**38,3 %**	**0,197**
Intermediate W2AVV++	10,1 %	28,0 %	38,4 %	0,193	**10,7 %**	**28,8 %**	**38,7 %**	**0,198**

Tabla 5.3: Resultados del Experimento 2: Learning Rate

Se puede ver que al usar un *learning rate* de 0,001 se obtienen mejores resultados a través de todas las métricas involucradas. Como el valor 0,001 es más alto que 0,0001, en las primeras *epochs* la red va disminuyendo su pérdida en saltos más grandes, debido a que va aprendiendo cosas nuevas más rápidamente. Sin embargo, es importante notar que cuando no se mejora el *validation accuracy* obtenido por dos *epochs* consecutivas, el *learning rate* se disminuye a la mitad. Nuevamente, los dos modelos W2AVV++ obtienen levemente mejores resultados que el modelo W2VV++. En este caso, para R@1 el esquema Early Fusion obtiene un mejor resultado, pero para R@5 y R@10 el esquema Intermediate Fusion lo supera por un márgen pequeño.

Experimento 3: Función de Optimización

Se entrenan los tres modelos usando como función de optimización RMSprop y Adam, dejando los demás hiperparámetros fijos: *batch size* 100, *learning rate* de 0,001 y Triplet Ranking Loss con *alpha margin* de 0,5. Los resultados se pueden ver en la Tabla 5.4.

	RMSprop				Adam			
	R@1	R@5	R@10	mAP	R@1	R@5	R@10	mAP
W2VV++	**10,5 %**	**27,7 %**	**38,9 %**	**0,194**	8,9 %	25,4 %	35,8 %	0,173
Early W2AVV++	**11,1 %**	**28,3 %**	**38,3 %**	**0,197**	10,1 %	25,7 %	35,3 %	0,176
Intermediate W2AVV++	**10,7 %**	**28,8 %**	**38,7 %**	**0,198**	9,8 %	25,6 %	35,5 %	0,175

Tabla 5.4: Resultados del Experimento 3: Función de Optimización

Se concluye a partir de los resultados de este experimento que es más conveniente usar RMSprop por sobre Adam al entrenar estos modelos. Estos resultados son los mismos que destacan en el experimento anterior.

Experimento 4: Batch Size

Se entrenan los tres modelos usando un *batch size* de 100 y uno de 128, dejando los demás hiperparámetros fijos: Triplet Ranking Loss con *alpha margin* de 0,5, *learning rate* de 0,001 y usando RMSprop. Los resultados se pueden ver en la Tabla 5.5.

	Batch Size = 100				Batch Size = 128			
	R@1	R@5	R@10	mAP	R@1	R@5	R@10	mAP
W2VV++	10, %5	27,7 %	38,9 %	0,194	**12,0 %**	**33,5 %**	**45,2 %**	**0,229**
Early W2AVV++	11,1 %	28,3 %	38,3 %	0,197	**12,4 %**	**34,3 %**	**45,5 %**	**0,232**
Intermediate W2AVV++	10,7 %	28,8 %	38,7 %	0,198	**10,8 %**	**30,8 %**	**42,6 %**	**0,201**

Tabla 5.5: Resultados del Experimento 4: Batch Size

En este experimento, se observa una mejora en los resultados al utilizar un *batch size* de 128. Es interesante notar que con esta configuración, el modelo W2VV++ obtiene mejores resultados que el modelo Intermediate W2AVV++, pero aún así la propuesta Early W2AVV++ es superior.

Experimento 5: Triplet Ranking Loss Alpha Margin

Se entrenan los tres modelos usando como función de pérdida Triplet Ranking Loss y se varía el valor de su *alpha margin*, dejando los demás hiperparámetros fijos: *batch size* 128, *learning rate* de 0,001 y usando RMSprop. Los resultados se pueden ver en la Tabla 5.6.

	$\alpha = 0{,}2$				$\alpha = 0{,}5$			
	R@1	R@5	R@10	mAP	R@1	R@5	R@10	mAP
W2VV++	12,0 %	33,1 %	44,8 %	0,224	12,0 %	**33,5 %**	**45,2 %**	**0,229**
Early W2AVV++	11,9 %	32,8 %	43,8 %	0,221	**12,4 %**	**34,3 %**	**45,5 %**	**0,232**
Intermediate W2AVV++	**12,6 %**	**34,2 %**	**45,3 %**	**0,232**	10,8 %	30,8 %	42,6 %	0,201

Tabla 5.6: Resultados del Experimento 5: Triplet Ranking Loss Alpha Margin

Se observa que el modelo Intermediate W2AVV++ se beneficia del uso de un $\alpha = 0{,}2$ a diferencia de los otros dos modelos. De hecho, al usar este valor, el modelo obtiene resultados casi idénticos a los obtenidos al modelo Early W2AVV++, incluso obteniendo un mejor R@1. Estos son los mejores resultados obtenidos si se comparan todos los experimentos. Cabe destacar también que ambos modelos audiovisuales obtienen levemente mejores resultados que el modelo visual. En este caso, no es posible determinar si un valor en particular para el hiperparámetro α es mejor de forma transversal, puesto que depende del tipo de esquema que sea utilizado.

A partir de los experimentos anteriores, se concluye que el modelo W2AVV++ tanto en su versión Early Fusion como en su versión Intermediate Fusion, obtiene mejores resultados que el modelo W2VV++, aunque por un márgen pequeño.

Gran parte de las descripciones que forman el conjunto de prueba contienen palabras que se enfocan solamente en elementos visuales que llamaron la atención de la persona que vió el video. Esto podría estar relacionado con la calidad de los resultados entregados por los modelos, especialmente los audiovisuales. Un modelo audiovisual no podrá generalizar de manera correcta videos que tienen sonidos particulares si la descripción correspondiente solo habla de los elementos visuales del video, como “una persona con una polera roja”.

Tomando en cuenta esto, se propone un enfoque en un escenario más específico, en donde se desea encontrar videos con consultas que tengan palabras que tengan relación directa con conceptos audiovisuales. Por ejemplo, una acción que tiene sonidos particulares sería "cantar" o "manejar una motocicleta". Si los modelos audiovisuales son capaces de recuperar videos con mejores resultados que el modelo visual en este escenario específico, se puede concluir que es una mejor propuesta en este caso.

5.2. Experimentos sobre consultas audiovisuales

Se proponen 20 consultas en texto que contienen conceptos audiovisuales para evaluar la calidad de las respuestas de cada modelo. Se utilizan ambos modelos W2AVV++ con los hiperparámetros configurados como en el experimento 5, con la diferencia de que se usa el esquema Early con un $\alpha = 0,5$ y el esquema Intermediate con un $\alpha = 0,2$. Además, con el fin de comparar la capacidad de los modelos, también se usa el modelo W2VV++ del experimento 5 para evaluar los videos que es capaz de recuperar en contraste con los modelos audiovisuales.

En la Tabla 5.7 se pueden ver las 20 consultas y los resultados obtenidos por cada modelo mencionado. Se utiliza como métrica de evaluación Precision@K (P@K) con K = 1, 5 y 10. Esta métrica entrega la cantidad de videos relevantes que son recuperados en los primeros top-K videos. Por ejemplo, P@10 = 50 % significa que la mitad de los primeros 10 videos recuperados son relevantes para la consulta realizada. También se calcula el promedio para cada P@K obtenido por cada sistema, para comparar los resultados en promedio para consultas de este tipo.

Según los resultados presentados, se observa que los modelos audiovisuales obtienen mejores resultados de forma más notoria. Para P@1, el modelo Early Fusion W2AVV++ obtiene un 85 % en promedio, mientras que el modelo visual W2VV++ obtiene un 65 %, lo que significa que el modelo audiovisual recupera un video relevante en la primera posición un 20 % más de veces. Para P@5, ambos modelos audiovisuales obtienen un 71 % en promedio, mientras que el modelo W2VV++ obtiene un 67 %, lo que significa que los modelos audiovisuales obtienen en promedio un 4 % más de videos relevantes en las primeras 5 posiciones. Para P@10, el modelo Intermediate Fusion W2AVV++ obtiene un 67 % en promedio, mientras que el modelo W2VV++ obtiene un 55 %, lo que significa que el modelo audiovisual obtiene en promedio un 12 % más de videos relevantes en las primeras 10 posiciones. Esto se puede ver graficado en la Figura 5.1.

Es importante considerar que este experimento considera solo 20 consultas, lo que no es suficiente para generalizar y concluir que siempre los modelos audiovisuales obtendrán resultados mucho mejores que el modelo visual. Para esto sería necesario generar un conjunto muy grande de consultas audiovisuales y evaluar los resultados de cada modelo.

En el Anexo A se presentan capturas de los primeros 5 videos recuperados por cada sistema para cada consulta realizada en este experimento. En algunas capturas no se observa lo que se busca con la consulta pero sí se escucha, por lo que en estos casos se considera como un video relevante. Cada video relevante es encerrado con un marco verde y cada video irrelevante con un marco rojo.

		Early Fusion W2AVV++			Intermediate Fusion W2AVV++			W2VV++		
#	**Consulta**	**P@1**	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**
1	a dog barking	0 %	**40 %**	30 %	0 %	20 %	**40 %**	**100 %**	**40 %**	20 %
2	a girl singing	**100 %**	**100 %**	**100 %**	**100 %**	60 %	50 %	**100 %**	**100 %**	70 %
3	a man talking about something	**100 %**	**100 %**	90 %	**100 %**	**100 %**	**100 %**	**100 %**	80 %	70 %
4	a man or a woman dancing around	0 %	**40 %**	**50 %**	0 %	0 %	10 %	0 %	0 %	0 %
5	playing with a baby	**100 %**	**100 %**	70 %	**100 %**	**100 %**	**90 %**	**100 %**	80 %	80 %
6	riding a motorcycle	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	**100 %**	90 %
7	a choir of people singing	**100 %**	40 %	**30 %**	**100 %**	**60 %**	**30 %**	**100 %**	40 %	**30 %**
8	a volleyball match	**100 %**	**100 %**	**80 %**	**100 %**	**100 %**	**80 %**	**100 %**	**100 %**	**80 %**
9	a waterfall flowing	**100 %**	**60 %**	**60 %**	**100 %**	**60 %**	**60 %**	**100 %**	**60 %**	**60 %**
10	driving a car	**100 %**	60 %	80 %	**100 %**	**80 %**	**90 %**	0 %	60 %	70 %
11	having or giving an interview	**100 %**	**80 %**	70 %	**100 %**	**80 %**	**80 %**	0 %	60 %	60 %
12	two men discussing	**100 %**	20 %	20 %	0 %	**80 %**	**70 %**	**100 %**	**80 %**	60 %
13	a narrator talks	**100 %**	80 %	70 %	**100 %**	**100 %**	**80 %**	**100 %**	60 %	60 %
14	a talk show host talking	**100 %**	**100 %**	**90 %**	**100 %**	80 %	80 %	**100 %**	80 %	50 %
15	people laughing	**100 %**	**80 %**	**70 %**	**100 %**	**80 %**	**70 %**	0 %	60 %	40 %
16	a man yelling	**100 %**	**40 %**	**30 %**	0 %	0 %	0 %	0 %	20 %	10 %
17	a baby crying or talking	0 %	40 %	60 %	0 %	**60 %**	**70 %**	0 %	**60 %**	50 %
18	a parrot talking	**100 %**	**100 %**	70 %	**100 %**	**100 %**	**80 %**	**100 %**	**100 %**	**80 %**
19	playing ping pong	**100 %**	**100 %**	**90 %**	**100 %**	**100 %**	**90 %**	**100 %**	**100 %**	**90 %**
20	a crowd cheering	**100 %**	40 %	40 %	**100 %**	**60 %**	**60 %**	0 %	**60 %**	30 %
	Promedio obtenido	**85 %**	**71 %**	65 %	75 %	**71 %**	**67 %**	65 %	67 %	55 %

Tabla 5.7: Resultados de experimento con 20 consultas con conceptos audiovisuales. Se calcula el Precision@K (P@K) obtenido por cada sistema en cada consulta con K=1, 5 y 10. Finalmente se calcula el promedio de Precision@1, Precision@5 y Precision@10 obtenido por cada sistema.

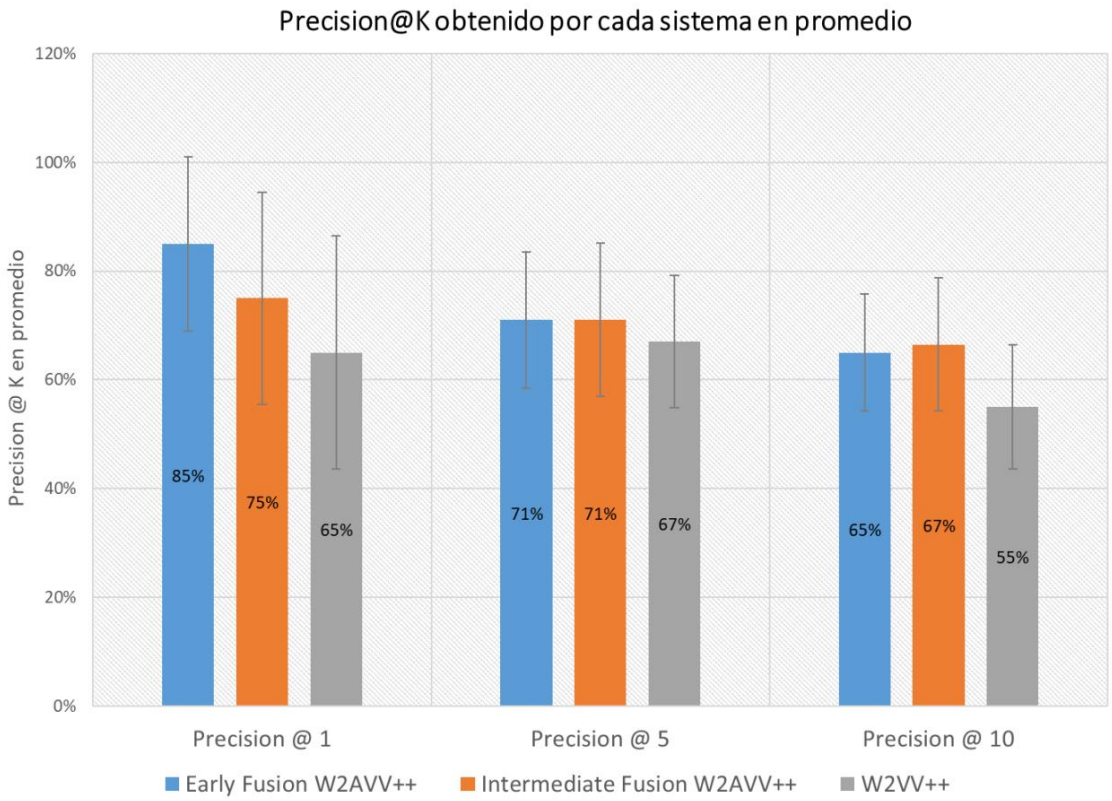


Figura 5.1: Precision@K obtenido por cada sistema en promedio.

Capítulo 6

Participación en TRECVID

6.1. Ad-hoc Video Search

En este capítulo se detalla la participación en la conferencia TRECVID 2019 [2], específicamente en la competencia Ad-hoc Video Search, utilizando como propuesta una versión modificada del modelo W2AVV++ propuesto en este trabajo.

TRECVID [36] es una conferencia científica dedicada a la investigación sobre análisis de contenido multimedia, especialmente videos. Su objetivo es alentar la investigación en el campo de la recuperación de información. Anualmente los organizadores generan o liberan datasets grandes para evaluar con métricas uniformes a todos los equipos inscritos en distintas tareas, y permiten la participación de equipos de todo el mundo para compartir conocimiento. Esta conferencia no tiene fines económicos ni otorga premios.

TRECVID propone diversas competencias o *tasks* en donde los equipos participantes deben inscribirse. "Ad-hoc Video Search" (AVS) es la tarea donde un sistema automático es capaz de analizar una consulta de texto y recuperar videos que contengan acciones, objetos, personas, localidades, entre otras cosas indicadas en tal consulta. Dada una colección de videos de prueba sin etiquetar y una lista de 30 consultas ad-hoc, un sistema participante debe recuperar por cada consulta una lista de máximo 1.000 videos de la colección, rankeados según tengan la mayor probabilidad de contener la consulta. Esta competencia se ha realizado desde el año 2016.

La presencia de cada consulta dentro de un video se asume como binaria, es decir, está o no dentro de un video. En el año 2019, se utiliza como dataset de prueba la colección V3C1 [5] (la que es extraída a partir de V3C [33], que es un dataset más grande), que está compuesta por 7.475 videos de Vimeo [1]. Este dataset está segmentado para la competencia en 1.082.657 segmentos de video más cortos generados a partir de los 7.475 originales.

Para evaluar la calidad de los resultados entregados por cada sistema participante, un comité de jueces los evalúa siguiendo algunas reglas. Si una consulta está presente para algún cuadro o secuencia de cuadros dentro de un video, entonces se considera presente para todo ese segmento. En una consulta, el término "contains x" se refiere a que un segmento de video contiene x hasta cierto punto tal que x sea reconocible por un humano. Por lo tanto,

[1] https://vimeo.com/

visibilidad o audición parcial bastan para identificar un objeto o acción. Si un video contiene objetos físicos representando lo que dice la consulta, como fotos, pinturas, modelos o versiones de juguete del objetivo de la consulta (por ejemplo, Barack Obama v/s una foto de Barack Obama), no significa que la consulta sea verdadera para ese video. Si un video contiene otro video dentro de él, en ese caso la consulta puede ser verdadera.

Los equipos participantes deben declarar el tipo de datos de entrenamiento usados:

1. **A**: el sistema usa solo datos de entrenamiento de Internet Archive (IACC), publicados por TRECVID.
2. **D**: el sistema usa cualquier dataset de entrenamiento que contenga descripciones de los videos.
3. **E - no annotation category**: el sistema solo usa datos de entrenamiento recolectados automáticamente usando las consultas oficiales, sin ningún otro tipo de descripción manual realizada en cada ejemplar recolectado. Por ejemplo, se puede utilizar Google para recopilar datos usando cada consulta oficial, pero no se pueden realizar anotaciones sobre los datos.
4. **F - no annotation category**: el sistema solo usa datos de entrenamiento recolectados automáticamente usando una consulta construida manualmente a partir de la consulta oficial.

Para la competencia se permite el envío de 4 conjuntos de resultados o "*runs*" por equipo. Existen 3 categorías para cada *run*:

1. **Fully automatic runs**: El sistema toma una consulta como entrada y produce un resultado sin intervención humana de ningún tipo.
2. **Manually assisted runs**: Un humano puede formular una consulta inicial basada en la consulta oficial, pero sin conocimientos previos sobre los resultados que esa consulta entrega sobre el dataset de prueba. Luego el sistema toma la consulta reformulada como entrada y produce un resultado sin intervención humana posterior.
3. **Relevance feedback**: El sistema toma una consulta oficial como input y produce resultados iniciales, luego un humano puede elegir los mejores 5 resultados y entregarle esta información como feedback al sistema para producir un set final de resultados. Este proceso es permitido solo una vez por entrega.

Por cada consulta, se crean dos *pool* de videos a partir de los resultados de un sistema. El *top pool* se conforma del 100 % de los videos rankeados desde la posición 1 hasta la 250 de una entrega. El *bottom pool* se conforma de un 11 % de videos rankeados desde la posición 251 hasta la 1.000 seleccionados al azar y que no se encuentren en el *top pool*. Estas muestras de videos son analizadas por un grupo de jueces humanos para determinar si son resultados correctos.

La métrica usada en AVS para evaluar los resultados enviados por un sistema es "inferred Average Precision" (infAP) [45]. Esta métrica fue creada para poder evaluar la información

recuperada por un sistema cuando se tienen colecciones muy grandes de datos, en donde es difícil juzgar e identificar todos los documentos relevantes. infAP estima el Average Precision sobre la colección completa de datos a partir de una muestra aleatoria. La estimación es el resultado de un experimento aleatorio, en que dada una lista de resultados rankeados a partir de una consulta, se siguen los siguientes pasos:

1. Se selecciona un documento relevante al azar de la colección, y se denomina el *rank* de este documento en la lista como i.
2. Se elige un *rank* al azar del set $\{1, \ldots, i\}$.
3. Se retorna la relevancia del documento en el *rank* i de forma binaria.

Al seguir estos pasos, se puede estimar el Average Precision sobre una colección grande de datos al realizar este experimento aleatorio sobre la muestra analizada por los jueces.

Los plazos establecidos para participar en esta competencia son establecidos a principio de cada año. En el caso de TRECVID 2019, el 24 de Julio fue la fecha límite para entregar la participación de un equipo en la competencia AVS. El 30 de Agosto fueron publicados los resultados y el 29 de Octubre fue la fecha límite para entregar un paper que debía resumir la participación de un equipo. El modelo que fue propuesto para participar en esa competencia se presenta en la siguiente sección, además de ser detallado en Hernández et al. [17].

6.2. Hybrid Intermediate W2AVV++

Debido a que no se alcanzaron a desarrollar los modelos Early e Intermediate W2AVV++ presentados en las secciones anteriores dentro de los plazos establecidos para esta competencia, fue propuesta una mezcla entre ambos esquemas que fue denominada como "Hybrid Intermediate W2AVV++".

Se siguen las mismas ideas que en los esquemas Early e Intermediate con respecto a la representación de las modalidades multimedia de un video, es decir, se utilizan modelos preentrenados de ConvNets para cada modalidad. Para la representación visual se utilizan los modelos ResNet-152 y ResNeXt-101, mientras que para la representación auditiva se utilizan los modelos SoundNet y AENet. Por lo tanto, para cada video se generan estos 4 descriptores.

Los descriptores auditivos son concatenados inmediatamente en una rama auditiva, mientras que los descriptores visuales son concatenados y se les aplica *feature re-learning* en una rama visual. La rama auditiva genera un descriptor auditivo de 2.048 dimensiones y la rama visual genera un descriptor visual de 2.048 dimensiones también. Luego, las salidas de ambas ramas son concatenadas generando un descriptor de 4.096 dimensiones y usando una nueva capa *fully connected* se realiza un segundo proceso de *feature re-learning* que entrega un descriptor de video de 2.048 dimensiones, representado por $\phi(v)$ en la Figura 6.1. Para la representación de texto se sigue la misma estrategia que en los modelos Early e Intermediate W2AVV++. Además, se utiliza la misma función de pérdida Triplet Ranking Loss.

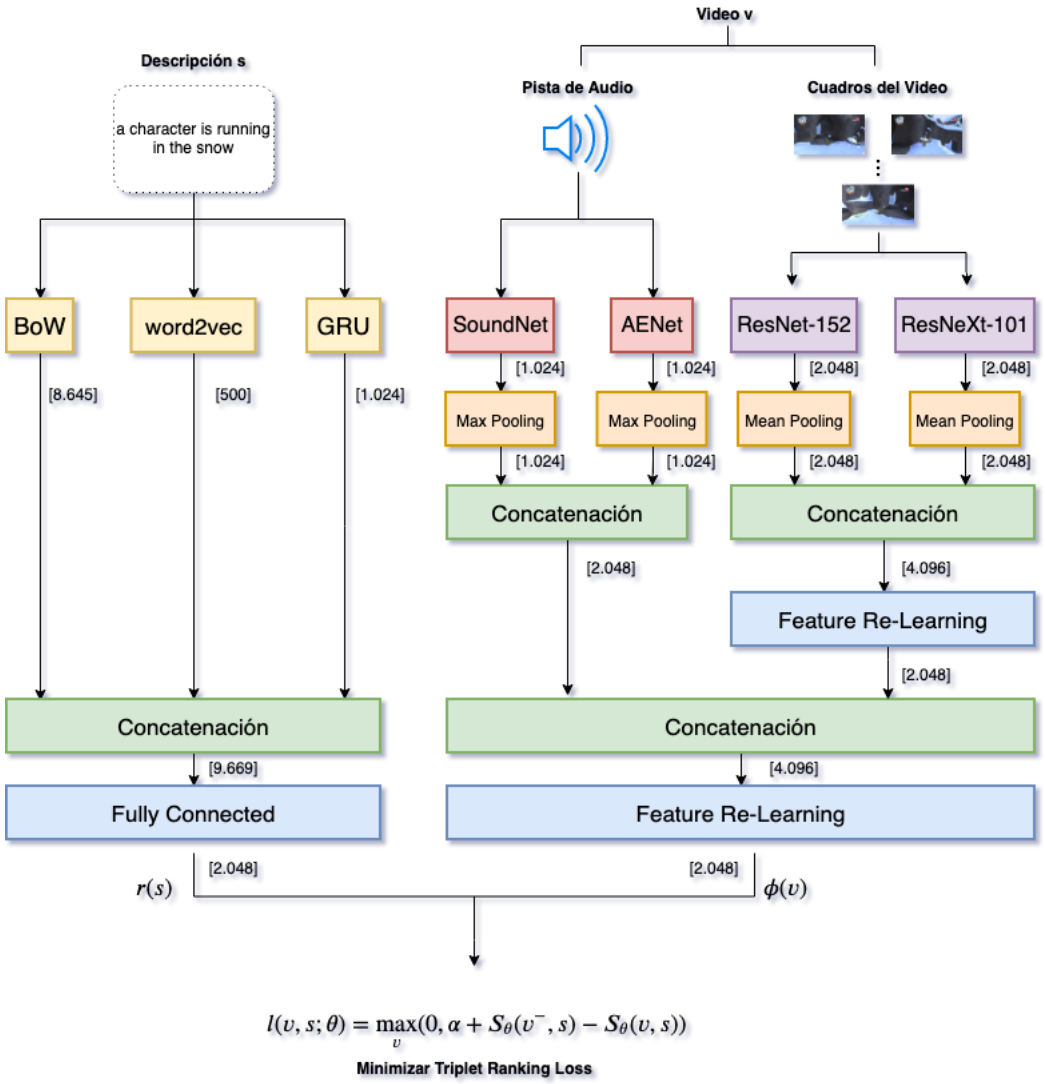


Figura 6.1: Representación del modelo Hybrid Intermediate W2AVV++ utilizado en AVS. Se extraen descriptores para cada modalidad, pero se combinan los descriptores por modalidad separados. Se realiza *feature re-learning* visual de forma preliminar, cuyo resultado es combinado con los descriptores auditivos para luego generar descriptores audiovisuales.

Para entrenar este modelo se utiliza el dataset MSR-VTT con la misma configuración que en los modelos Early e Intermediate W2AVV++. Sin embargo, para testear el modelo se debía utilizar el dataset V3C1 correspondiente a la competencia AVS. Con respecto a este dataset, se presentó un problema particular que afectó directamente en el sistema propuesto. Solo 586.730 de los videos de V3C1 tienen una pista de audio válida (aproximadamente un 54 % del dataset), mientras que el resto son frames sin una pista de audio o esta es demasiado corta por lo que ni SoundNet ni AENet son capaces de extraer descriptores auditivos. Para poder enfrentar este problema, se siguieron los siguientes pasos:

1. Se entrenaron dos modelos: un modelo solo visual W2VV++ y un modelo audiovisual Hybrid Intermediate W2AVV++, ambos entrenados con MSR-VTT.

2. Si un video de V3C1 permite la extracción de un descriptor auditivo, entonces se calcula el descriptor audiovisual del video y además se calcula su descriptor solo visual.

3. Si un video de V3C1 no permite la extracción de un descriptor auditivo, entonces se obtiene solo un descriptor visual del video y su descriptor audiovisual es un vector de ceros. Se guarda cada descriptor visual en una matriz y cada descriptor audiovisual en otra.

4. Por cada consulta de prueba, el modelo W2VV++ entrenado predice un descriptor visual y el modelo W2AVV++ entrenado predice un descriptor audiovisual. Se implementa la similitud ente modalidades entre una consulta dada y cualquiera de los videos del dataset V3C1 como la similitud coseno entre el descriptor textual y el descriptor del video. Por cada consulta, se calcula la similitud coseno entre el descriptor visual generado a partir de la consulta y el descriptor visual calculado a partir de cada video. Se hace lo mismo entre el descriptor audiovisual generado a partir de la consulta y el descriptor audiovisual calculado a partir de cada video. Esto genera dos matrices de similitud, una correspondiente a los descriptores visuales y otra a los descriptores audiovisuales.

5. Finalmente, se selecciona el máximo entre cada fila (que corresponde a un video) en las dos matrices, y luego se entrega una lista de 1.000 videos rankeados según la similitud coseno en orden descendiente.

6.3. Resultados

Se enviaron 4 *runs* a evaluación en AVS. Run 1 y Run 3, que llamaremos `AudioV05` y `AudioV02` respectivamente, consisten en los resultados obtenidos por la estrategia descrita en la sección anterior. Run 2 y Run 4, que llamaremos `Visual05` y `Visual02` respectivamente, consisten en los resultados obtenidos por un sistema basado solo en W2VV++ (modelo visual) y solo se usan como base de comparación. Los detalles de cada entrega son los siguientes:

1. **`AudioV05` (audiovisual)**: Utiliza los descriptores visuales (ResNet-152 y ResNeXt-101) y los descriptores audiovisuales (visuales + SoundNet + AENet) predichos a partir de cada consulta, entrenando los modelos W2VV++ y W2AVV++ como se explicó en la sección anterior. Para la función de pérdida, se utiliza un *alpha margin* con un valor de $\alpha = 0{,}5$.

2. **`Visual05` (visual)**: Usa solo los descriptores visuales predichos a partir de cada query, entrenando un modelo W2VV++. Esta entrega funciona como una base de comparación para `AudioV05`. Para la función de pérdida, se utiliza un *alpha margin* con un valor de $\alpha = 0{,}5$.

3. **`AudioV02` (audiovisual)**: Lo mismo que en `AudioV05`, pero se utiliza un *alpha margin* con un valor de $\alpha = 0{,}2$.

4. **`Visual02` (visual)**: Lo mismo que en `Visual05`, pero se utiliza un *alpha margin* con un valor de $\alpha = 0{,}2$. Esta entrega funciona como una base de comparación para `AudioV02`.

Los resultados generales de cada entrega se pueden ver en la Figura 6.2, según la métrica Mean infAP. Si bien `Visual05` logra resultados generales superiores a los obtenidos por `AudioV05`, `AudioV02` logra resultados superiores a `Visual02`. Esto indica que ninguno de los dos modelos es siempre superior al otro al utilizar los mismos hiperparámetros en ambos, y como se podrá ver a continuación, la calidad de los videos recuperados por cada sistema varía dependiendo de cada consulta.

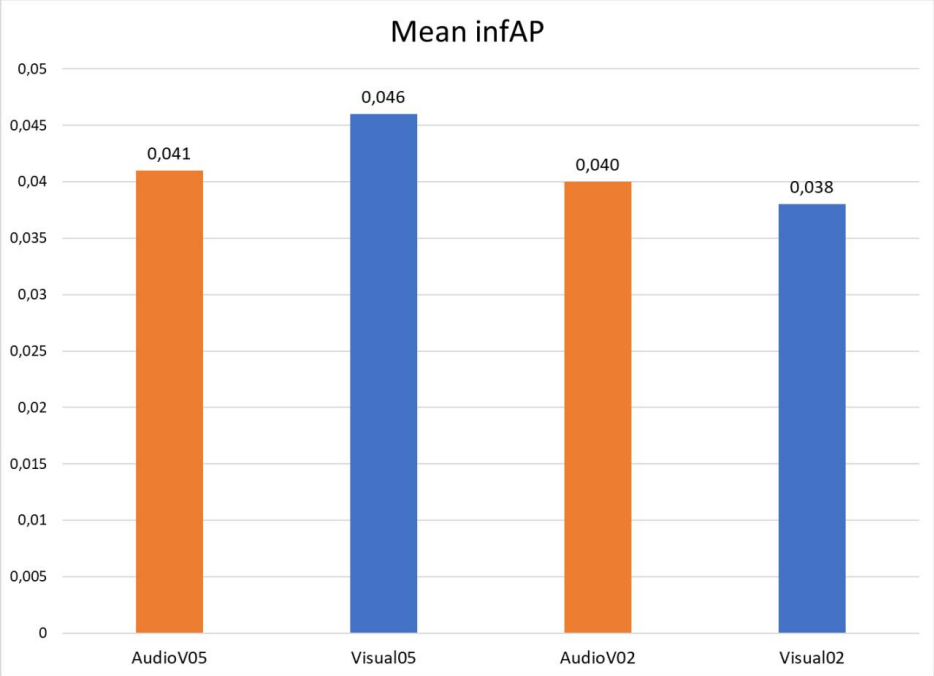


Figura 6.2: Mean infAP obtenido por cada sistema entregado en AVS

En la Tabla 6.1 se muestra el infAP obtenido por cada sistema para once consultas cuyos resultados son interesantes. El resultado obtenido para todas las consultas se puede ver en el Anexo B.2, los equipos que participaron en AVS se pueden ver en el Anexo B.1 y el Mean infAP obtenido en todas las entregas de cada uno de ellos se puede ver en el Anexo B.3.

Se puede ver que para las consultas 612, 618, 620 y 636 los modelos audiovisuales obtienen mejores resultados que sus contrapartes visuales. Para la consulta 621, a pesar de que la entrega `Visual05` obtiene resultados levemente mejores que la entrega `AudioV05`, la entrega `AudioV02` logra mejores resultados que la entrega `Visual02`. Para la consulta 624, la entrega `AudioV05` obtiene un infAP bajo de 0,008 pero la entrega `Visual05` obtiene 0,000, lo que significa que el modelo visual no entrega resultados válidos. Para las consultas 631 y 633, la entrega `AudioV02` obtiene resultados notoriamente mejores que la entrega `Visual02`. Por otro lado, en las consultas 623, 637 y 640, los modelos visuales obtienen resultados mucho mejores que los modelos audiovisuales.

Los resultados presentados en la tabla anterior sugieren que usar la propuesta audiovisual puede mejorar la calidad de la recuperación de videos en ciertos escenarios. A continuación, se toman como referencia las consultas 612, 618 y 631 para mostrar algunos resultados interesantes.

Query ID	Query	AudioV05	Visual05	AudioV02	Visual02
612	a truck being driven in the daytime	**0,038**	0,030	**0,036**	0,029
618	coral reef underwater	**0,093**	0,071	**0,135**	0,114
620	a person with a painted face or mask	**0,268**	0,254	**0,222**	0,208
621	person in front of a graffiti painted on a wall	0,031	**0,032**	**0,053**	0,045
623	a person wearing shorts outdoors	0,078	**0,104**	0,048	0,048
624	a person in front of a curtain indoors	**0,008**	0,000	0,005	0,005
631	a man and a woman dancing together indoors	0,109	0,109	**0,138**	0,026
633	a group of people walking on the beach	0,105	0,105	**0,117**	0,110
636	a man and a baby both visible	**0,025**	0,013	**0,014**	0,009
637	a shirtless man standing up or walking outdoors	0,059	**0,165**	0,056	**0,067**
640	a red hat or cap	0,017	**0,054**	0,019	**0,033**

Tabla 6.1: inferred Average Precision obtenido por consulta en cada entrega.

Consulta 612: "a truck being driven in the daytime"

Esta consulta busca encontrar videos en donde aparezca un camión siendo manejado durante el día. Según los resultados vistos, ambas entregas audiovisuales consiguen un infAP más alto que sus contrapartes visuales. Si se toma la lista de 1.000 videos rankeados de las entregas `AudioV05` y `Visual05`, se observa que por ejemplo el video "shot00859" (ver Figura 6.3 como referencia), que corresponde a un camión con el motor encendido a plena luz de día, aparece 29 veces (separado en 29 segmentos cortos) en la lista. Lo interesante con respecto a esto, es que en el caso del modelo audiovisual de la entrega `AudioV05`, dentro de los primeros 100 videos recuperados 10 corresponden a segmentos de este video. Es decir, se obtiene un recall@100 de un 35 %. Por otro lado, para el modelo visual del `Visual05`, solo se obtiene un recall@100 de un 21 %. Por lo tanto, el modelo audiovisual logra rankear de mejor manera videos relevantes en este caso. En el video solo se escucha el motor del camión encendido y el movimiento del camión que pasa al fondo.

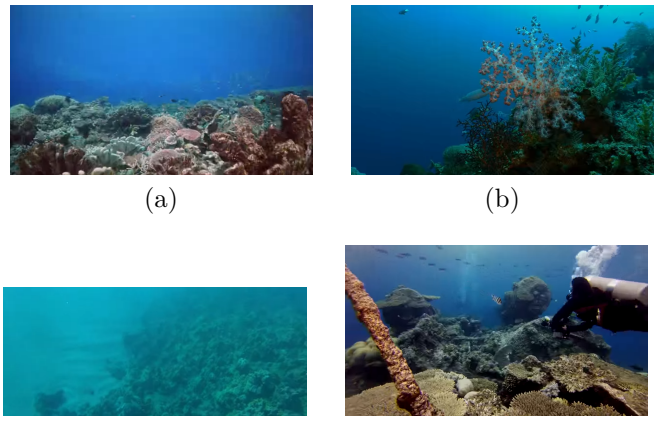
Figura 6.3: Captura de video recuperado por la consulta 612.

Consulta 618: "coral reef underwater"

Esta consulta busca encontrar videos en donde aparezcan arrecifes de coral bajo el agua. Los resultados obtenidos por ambos modelos audiovisuales son mejores que los modelos visuales, aunque en este caso sucede algo peculiar. Al ver algunos de los videos que están rankeados en las primeras posiciones en los resultados de cada entrega audiovisual (ver Figura 6.4 como referencia), efectivamente estos corresponden a la consulta ingresada ya que se ven arrecifes de coral bajo el agua. Sin embargo, la pista de audio de la mayoría de los videos no es un audio natural capturado directamente sino que corresponde a canciones o música sobrepuesta, la que no tiene relación alguna con el contenido del video.

Una de las causas de porqué los modelos audiovisuales obtienen mejores resultados que los visuales podría estar relacionada a los videos con los que los modelos fueron entrenados y sus descripciones. Si se busca entre las 200.000 descripciones del dataset MSR-VTT, existen 7 videos con la palabra "coral", 8 con la palabra "reef" y aproximadamente 40 con la palabra "underwater". Lo interesante es que varios de estos videos también tienen una pista de audio correspondiente a una canción o música sobrepuesta, por lo que es posible que el modelo audiovisual haya aprendido a identificar ese tipo de audio como una característica particular de los videos en donde aparecen arrecifes de coral bajo el agua.

Lamentablemente, esta conclusión no es realmente algo positivo, puesto que ese tipo de audio podría estar presente también en cualquier otro video, lo que causaría que el sistema recupere videos que no corresponden a la consulta.



(a) (b)

(c) (d)

Figura 6.4: Capturas de videos recuperados por la consulta 618.

Consulta 631: “a man and a woman dancing together indoors”

Esta consulta busca encontrar videos en donde aparezca un hombre y una mujer bailando juntos en espacios interiores. Si bien los modelos de `AudioV05` y `Visual05` obtienen el mismo valor de infAP, el sistema del `AudioV02` obtiene resultados mucho mejores que `Visual02`. Si se comparan los videos recuperados en las primeras posiciones por estos últimos dos sistemas, se puede ver que el modelo audiovisual efectivamente identifica videos correspondientes a la consulta y el modelo visual se equivoca en algunos (ver Figura 6.5 y Figura 6.6).

Una característica interesante de los segmentos recuperados por cada sistema es la duración de cada uno de ellos. Los videos recuperados en las primeras 20 posiciones por el sistema audiovisual tienen en su mayoría una duración mínima de 2 segundos y algunos se extienden por más de 5 segundos. Se puede escuchar con claridad una pista de audio en cada uno de estos videos, y todas consisten en música o una canción que acompaña al baile de las personas en el video. Por otro lado, los videos recuperados por el sistema visual tienen en su mayoría una duración máxima de 1 segundo, en donde no se logra escuchar con claridad la pista de audio. Esto sugiere que el modelo audiovisual obtiene mejores resultados porque tiene la capacidad de identificar características relevantes en el audio de un video con cierta duración mínima, como la música que suena en un baile, y utilizar esta información para identificar mejor los videos que corresponden a la consulta.

(a) (b) (c)

(d) (e) (f)

(g) (h)

Figura 6.5: Videos recuperados en la consulta 631 por `AudioV02`.

(a) (b) (c) (d) (e) (f) (g) (h)

Figura 6.6: Videos recuperados en la consulta 631 por `Visual02`.

Resultados generales

Los resultados obtenidos en consultas como la 623, 637 y 640, indican que si en la consulta no hay menciones ni palabras que hagan alusión a eventos auditivos, como "hablar" o "cantar", u objetos que emitan sonidos como un "motor" o un "camión", es poco probable que el uso del audio mejore los resultados obtenidos por un modelo que identifica elementos visuales que sí están presentes en la consulta. Por el contrario, es probable que al usar descriptores auditivos en conjunto a los visuales, los primeros contaminen lo que el modelo aprende y por lo tanto, recupera videos no relacionados. Por ejemplo, en la consulta 640 "a red hat or cap", en donde ninguna palabra indica una acción u objeto que involucre el sonido, los resultados obtenidos por los sistemas audiovisuales son peores que los sistemas visuales.

Si bien los resultados obtenidos por la propuesta audiovisual no son tan buenos en comparación a los obtenidos por los demás equipos, hay que considerar que las consultas de pueba de AVS tienen un sesgo importante hacia la búsqueda visual en videos. Por ejemplo, las consultas "person in front of a graffiti painted on a wall", "a bald man" y "a person wearing a backpack" solo contienen elementos visuales, por lo que utilizar el audio no presenta ningún beneficio al momento de realizar una búsqueda.

Es importante notar que entre los 10 equipos que participaron en AVS (ver Anexo B.1) con un total de 47 entregas realizadas, W2AVV++ es el único sistema que utiliza el audio de los videos. Esto le entrega un valor agregado al modelo propuesto ya que es único en su tipo en el contexto de AVS. Además, se destaca que el equipo IMFD_IMPRESEE fue el único equipo latinoamericano que participó en la competencia.

En el Anexo B.4 se pueden ver todas las consultas utilizadas en la competencia y en el Anexo B.5 se pueden ver los gráficos en los que aparecen los resultados obtenidos en comparación a los resultados generales de los demás equipos.

Capítulo 7

Conclusiones

En este trabajo de investigación se implementó un sistema que permite la recuperación de videos sin etiquetar mediante búsqueda por texto, utilizando un modelo de redes neuronales entrenado con descriptores visuales y auditivos obtenidos a partir de los videos. Primero se presentó la base teórica que forma parte de las soluciones existentes del problema de la recuperación de videos sin etiquetar, el estado del arte establecido por el sistema W2VV++ y también se detallaron los modelos auditivos utilizados en la nueva propuesta audiovisual. Para comparar los resultados del sistema propuesto, se implementó y entrenó el modelo W2VV++ utilizando el dataset MSR-VTT. Este modelo se usa como *baseline* para el modelo audiovisual.

La hipótesis que motiva este trabajo es que el uso del audio es relevante en un sistema buscador de videos sin etiquetar. Para comprobar esto fueron propuestos ciertos objetivos y una metodología a seguir, a partir de la cual se obtienen las siguientes conclusiones:

- Se implementó el modelo audiovisual W2AVV+ usando dos esquemas de combinación de descriptores distintos, llamados Early Fusion W2AVV++ e Intermediate Fusion W2AVV++. Ambos fueron entrenados con el dataset MSR-VTT al igual que el modelo base W2VV++. Usando el conjunto de prueba de este dataset se comprueba que los modelos audiovisuales obtienen mejores resultados por un pequeño margen. El modelo W2VV++ obtiene un mAP igual a 0,229 y ambos modelos audiovisuales obtienen un mAP igual a 0,232.

- Se realizaron 20 consultas con conceptos audiovisuales para evaluar la calidad de los videos recuperados por cada sistema. Estas consultas buscan evitar el sesgo hacia conceptos solo visuales presentes en la mayoría de las descripciones presentes en el conjunto de prueba. Las consultas mezclan elementos visuales como “a dog” con conceptos auditivos como “barking”. Se comprueba que en este caso ambos modelos audiovisuales obtienen mejores resultados que el modelo visual por un margen mayor. Early W2AVV++ obtiene un P@1 de 85 % en promedio mientras que Intermediate W2AVV++ obtiene un P@1 de 71 % y W2VV++ obtiene un P@1 de 65 %. Por lo tanto, el modelo Early W2AVV++ tiene una mayor capacidad de recuperar un video relevante en la primera posición que los otros modelos, lo que es sumamente importante para los usuarios que usan un buscador de videos ya que es el primer resultado que se mira. Por otro lado, Intermediate W2AVV++ obtiene un P@10 de 67 % mientras que Early W2AVV++ obtiene un P@10 de 65 % y W2VV++ obtiene un P@10 de 55 %. Esto quiere decir que

al considerar un rango más amplio de resultados, el modelo Intermediate W2AVV++ recupera una mayor cantidad de resultados relevantes en las primeras posiciones.

- Se concluye que si solo se usa el primer resultado del buscador al realizar una consulta audiovisual es mejor utilizar el esquema Early pero si se desea obtener una mayor cantidad de videos relacionados entonces es mejor usar el esquema Intermediate. Es posible mezclar lo mejor de ambos modelos al comparar la similitud de la consulta realizada con cada video recuperado por estos y seleccionando la que sea mayor, aprovechando las capacidades de cada uno.

- Hay una pequeña diferencia en el costo de entrenar ambos modelos ya que Intermediate W2AVV++ utiliza la técnica *feature re-learning*, lo que representa un paso adicional donde se deben pasar todos los descriptores de una modalidad por una capa *fully connected* extra que el modelo Early W2AVV++ no tiene. Sin embargo, este paso solo toma algunos minutos durante el entrenamiento con el dataset MSR-VTT, por lo que no significa una gran desventaja en eficiencia.

Utilizando una variante del modelo audiovisual propuesto, se participó en la conferencia internacional TRECVID 2019, específicamente en la competencia Ad-hoc Video Search (AVS). Esta participación es descrita en Hernández et al. [17]. Se destaca la importancia de participar en una conferencia de nivel mundial, en donde se proponen sistemas que establecen el estado del arte en el área y una organización externa formada por expertos evalúa todo el trabajo realizado a través de una competencia estructurada. Al observar las propuestas de los investigadores que participan, se pueden identificar las tendencias que existen en el campo y los elementos que permiten obtener mejores resultados. Esta retroalimentación permite que futuros investigadores del área sepan qué temas se deben desarrollar más. Sobre los resultados obtenidos por el equipo IMFD_IMPRESEE en TRECVID 2019 se puede concluir que:

- La mayoría de las consultas de prueba en AVS 2019 contienen conceptos exclusivamente visuales (por ejemplo “a bald man”), por lo que no se logran aprovechar al máximo las capacidades del modelo W2AVV++.

- Existen consultas donde se obtuvo mejores resultados con los modelos audiovisuales que con los modelos visuales. Por ejemplo, en la consulta “a truck being driven in the daytime” el modelo `AudioV05` recupera una mayor cantidad de videos relevantes en las primeras 100 posiciones que el modelo `Visual05`. En la consulta “a man and a woman dancing together indoors”, el modelo `AudioV02` recupera más videos de mayor duración que `Visual02`, por lo que el modelo audiovisual logra identificar características relevantes del audio como la música que suena durante un baile.

- Utilizar solamente el dataset MSR-VTT durante el entrenamiento de W2AVV++ representa una desventaja en comparación a un modelo como W2VV++ que además usa el dataset TGIF, el que contiene 100.000 GIFs y 120.000 descripciones de texto. Al ser modelos de redes neuronales profundas, se ven beneficiados al tener más datos durante el entrenamiento, por lo que W2AVV++ se encuentra en desventaja a causa de la falta de mejores datasets audiovisuales.

- Lamentablemente, solo un 54 % de los videos del dataset V3C1 usado en las pruebas de AVS 2019 tiene una pista de audio válida, lo que afecta en los resultados obtenidos

por el modelo audiovisual ya que no se pueden aprovechar al máximo sus capacidades si no existe un audio a analizar en un video.

A partir de cada uno de estos puntos, se concluye que la hipótetis de este trabajo era correcta, especialmente en el escenario particular en donde se tienen consultas que contienen conceptos audiovisuales como “un bebé llorando” o “una mujer cantando”.

Por último, es necesario resaltar la poca importancia que se le da al uso del audio por parte de la comunidad del área de recuperación de información multimedia. Esto se evidencia en que gran parte de las consultas usadas en la competencia AVS de TRECVID 2019 solo buscan elementos visuales (como “a red hat or cap”), los datasets usados para entrenamiento no contienen audio (como TGIF), solo un 54 % del dataset de prueba V3C1 contiene videos con una pista de audio válida y que el equipo IMFD_IMPRESEE fue el único entre los 10 equipos participantes en AVS 2019 que consideró el audio como fuente de información. Con este trabajo esperamos fomentar en la comunidad el uso del audio para investigar en el problema de búsqueda de videos sin etiquetar.

7.1. Trabajo a Futuro

El sistema audiovisual propuesto puede continuar desarrollándose de la siguiente forma:

- Generando un vocabulario de palabras/conceptos audiovisuales de forma previa y ponderarlos de diferente forma durante la vectorización de las descripciones. Por ejemplo, se podría generar un vocabulario audiovisual en donde se guardan las palabras que tienen relación a estos conceptos como “cantar”, “bailar”, “llorar”, “reír”, entre otros. Luego, al realizar la vectorización de texto, si una descripción contiene palabras que se encuentran en el vocabulario audiovisual, se le puede otorgar cierta ponderación mayor y generar un vector visual y uno audiovisual, para luego combinar ambos.

- Dado que existen consultas en donde se buscan conceptos netamente visuales y otras con conceptos audiovisuales, se podría enfrentar este problema utilizando un modelo combinado entre una red visual y otra audiovisual según concepto. Es decir, detectar en un principio si la descripción que se está procesando contiene en su mayoría conceptos exclusivamente visuales y usar un modelo como W2VV++ para procesarla, o si contiene conceptos audiovisuales y usar un modelo como W2AVV++. Luego, al procesar consultas nuevas, se utilizan ambos modelos y se recuperan los videos comparando las respuestas de los dos.

- Creando nuevos datasets audiovisuales que se construyan a partir de descripciones audiovisuales de los videos. Esto quiere decir que se debe establecer como regla para las personas que describen cada video, que deben tomar en cuenta lo que ven y lo que escuchan y así generar descripciones más completas del contenido del video. Esto permitiría entrenar los modelos audiovisuales con datasets más adecuados para aprovechar las características de cada uno.

Bibliografía

[1] George Awad, Asad Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzal Godil, David Joy, Andrew Delgado, Alan F. Smeaton, Yvette Graham, Wessel Kraaij, Georges Quénot, Joao Magalhaes, David Semedo, and Saverio Blasi. Trecvid 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search. In *Proceedings of TRECVID 2018*. NIST, USA, 2018.

[2] George Awad, Asad Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzal Godil, Andrew Delgado, Alan F. Smeaton, Yvette Graham, Wessel Kraaij, and Georges Quénot. Trecvid 2019: An evaluation campaign to benchmark video activity detection, video captioning and matching, and video search & retrieval. In *Proceedings of TRECVID 2019*. NIST, USA, 2019.

[3] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. In *Advances in Neural Information Processing Systems*, 2016.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

[5] Fabian Berns, Luca Rossetto, Klaus Schoeffmann, Christian Beecks, and George Awad. V3C1 dataset: An evaluation of content characteristics. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval, ICMR 2019, Ottawa, ON, Canada, June 10-13, 2019.*, pages 334–338, 2019.

[6] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, 2014.

[7] George Cybenko. Approximation by superpositions of a sigmoidal function. *MCSS*, 2 (4):303–314, 1989.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255, 2009.

[9] Jianfeng Dong, Shaoli Huang, Duanqing Xu, and Dacheng Tao. Dl-61-86 at trecvid 2017: Video-to-text description. *TRECVID Workshop*, 2017.

[10] Jianfeng Dong, Xirong Li, and Cees G. M. Snoek. Predicting visual features from text

for image and video caption retrieval. *IEEE Trans. Multimedia*, 20(12):3377–3388, 2018.

[11] Jianfeng Dong, Xirong Li, Chaoxi Xu, Gang Yang, and Xun Wang. Feature re-learning with data augmentation for content-based video recommendation. In *2018 ACM Multimedia Conference on Multimedia Conference, MM 2018, Seoul, Republic of Korea, October 22-26, 2018*, pages 2058–2062, 2018.

[12] Fartash Faghri, David J. Fleet, Jamie Kiros, and Sanja Fidler. VSE++: improving visual-semantic embeddings with hard negatives. In *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, page 12, 2018.

[13] Frederic Font, Gerard Roma, and Xavier Serra. Freesound technical demo. In *ACM Multimedia Conference, MM '13, Barcelona, Spain, October 21-25, 2013*, pages 411–412, 2013.

[14] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 249–256, 2010.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[17] Rodrigo Hernández, Jesus Perez-Martin, Nicolás Bravo, Juan Manuel Barrios, and Benjamin Bustos. IMFD IMPRESEE at TRECVID 2019: Ad-hoc video search and video to text. In *2019 TREC Video Retrieval Evaluation, TRECVID 2019, Gaithersburg, MD, USA, November 12-13, 2019*. National Institute of Standards and Technology (NIST), 2019. URL https://www-nlpir.nist.gov/projects/tvpubs/tv19.papers/imfd_impresee.pdf.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[19] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[20] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014*, pages 675–678, 2014.

[21] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition, 2019. URL http://cs231n.github.io/.

[22] Peter Knees and Markus Schedl. *Music Similarity and Retrieval - An Introduction to Audio- and Web-based Strategies*, volume 36 of *The Information Retrieval Series*. Springer, 2016.

[23] 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.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012.

[25] Xirong Li, Jianfeng Dong, Chaoxi Xu, Jing Cao, Xun Wang, and Gang Yang. Renmin university of china and zhejiang gongshang university at trecvid 2018: Deep cross-modal embeddings for video-text retrieval. In *TRECVID Workshop. https://github.com/li-xirong/avs*, 2018.

[26] Yuncheng Li, Yale Song, Liangliang Cao, Joel R. Tetreault, Larry Goldberg, Alejandro Jaimes, and Jiebo Luo. TGIF: A new dataset and benchmark on animated GIF description. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4641–4650, 2016.

[27] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, pages 740–755, 2014.

[28] Pascal Mettes, Dennis C. Koelma, and Cees G. M. Snoek. The imagenet shuffle: Reorganized pre-training for video event detection. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval, ICMR 2016, New York, New York, USA, June 6-9, 2016*, pages 175–182, 2016.

[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[30] Satoshi Nakamura, Kazuo Hiyane, Futoshi Asano, Takanobu Nishiura, and Takeshi Yamada. Acoustical sound database in real environments for sound scene understanding and hands-free speech recognition. In *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000, 31 May - June 2, 2000, Athens, Greece*, 2000.

[31] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com/.

[32] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 0033-295X.

[33] Luca Rossetto, Heiko Schuldt, George Awad, and Asad A Butt. V3c–a research video collection. In *International Conference on Multimedia Modeling*, pages 349–360. Springer, 2019.

[34] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[35] 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.

[36] Alan F. Smeaton, Paul Over, and Wessel Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.

[37] Cees Snoek, Marcel Worring, and Arnold W. M. Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th ACM International Conference on Multimedia, Singapore, November 6-11, 2005*, pages 399–402, 2005.

[38] Cees G. M. Snoek, Jianfeng Dong, Xirong Li, Xiaoxu Wang, Qijie Wei, Weiyu Lan, Efstratios Gavves, Noureldien Hussein, Dennis C. Koelma, and Arnold W. M. Smeulders. University of amsterdam and renmin university at trecvid 2016: Searching video, detecting events and describing video. *TRECVID Workshop*, 2016.

[39] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[40] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9, 2015.

[41] Naoya Takahashi, Michael Gygli, and Luc Van Gool. Aenet: Learning deep audio features for video analysis. *IEEE Trans. Multimedia*, 20(3):513–524, 2018.

[42] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 4489–4497, 2015.

[43] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[44] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. MSR-VTT: A large video description dataset for bridging video and language. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 5288–5296, 2016.

[45] Emine Yilmaz and Javed A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*, pages 102–111, 2006.

[46] Bolei Zhou, Àgata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 487–495, 2014.

Anexo A

Resultados de experimentos con consultas audiovisuales

Se presentan los resultados obtenidos en cada consulta audiovisual por los tres modelos entrenados. La primera fila corresponde a los resultados del modelo Early W2AVV++, la segunda fila corresponde a los resultados del modelo Intermediate W2AVV++ y la tercera fila corresponde a los resultados del modelo W2VV++.

Figura A.1: Videos recuperados en la consulta 1 del experimento audiovisual.

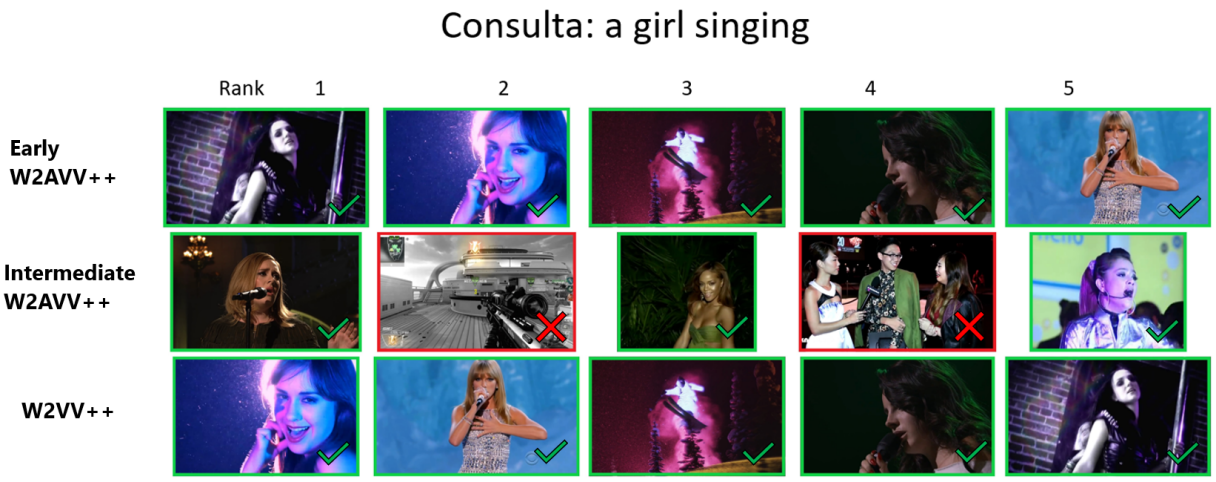


Figura A.2: Videos recuperados en la consulta 2 del experimento audiovisual

Figura A.3: Videos recuperados en la consulta 3 del experimento audiovisual.

Figura A.4: Videos recuperados en la consulta 4 del experimento audiovisual.

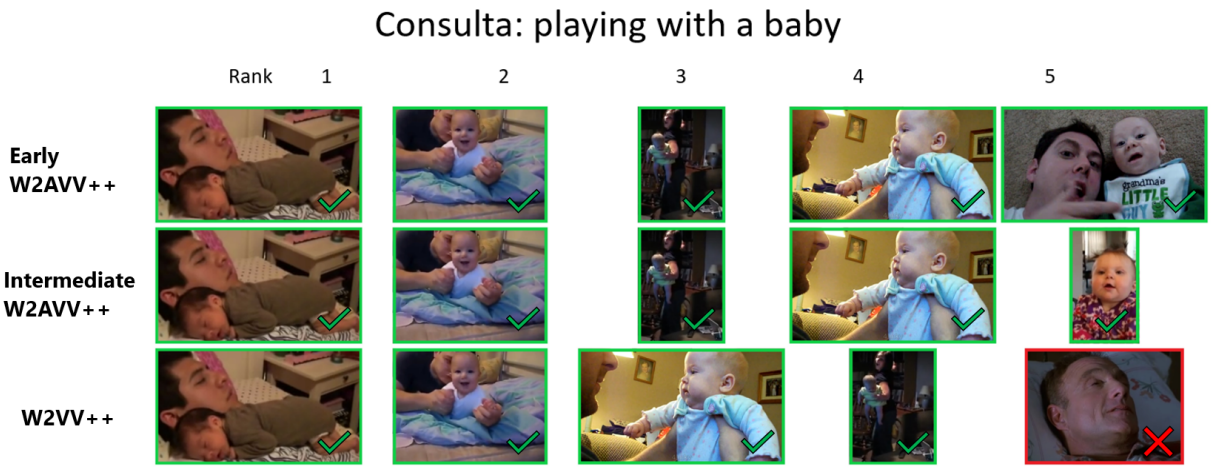


Figura A.5: Videos recuperados en la consulta 5 del experimento audiovisual.

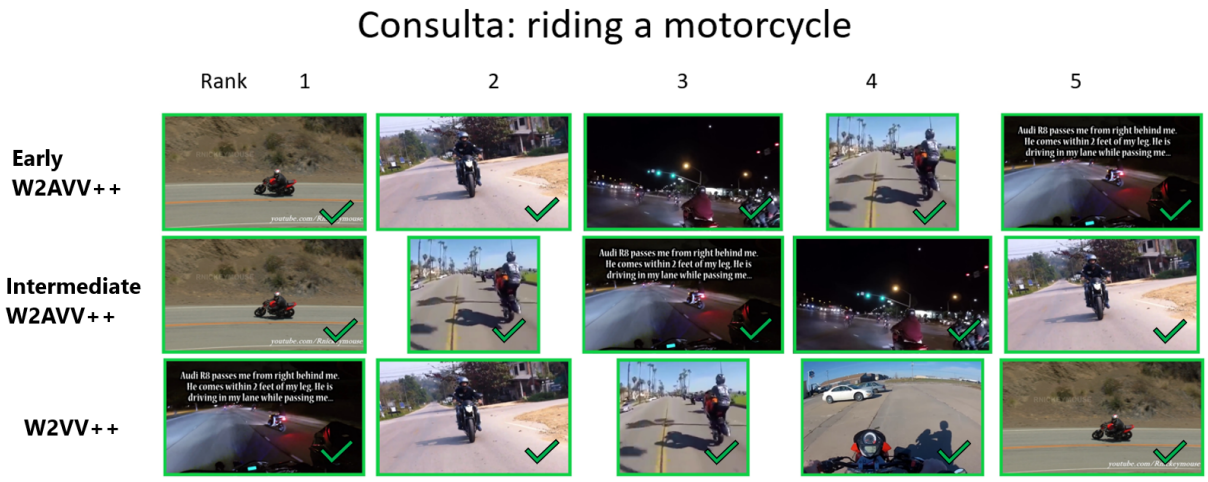


Figura A.6: Videos recuperados en la consulta 6 del experimento audiovisual.

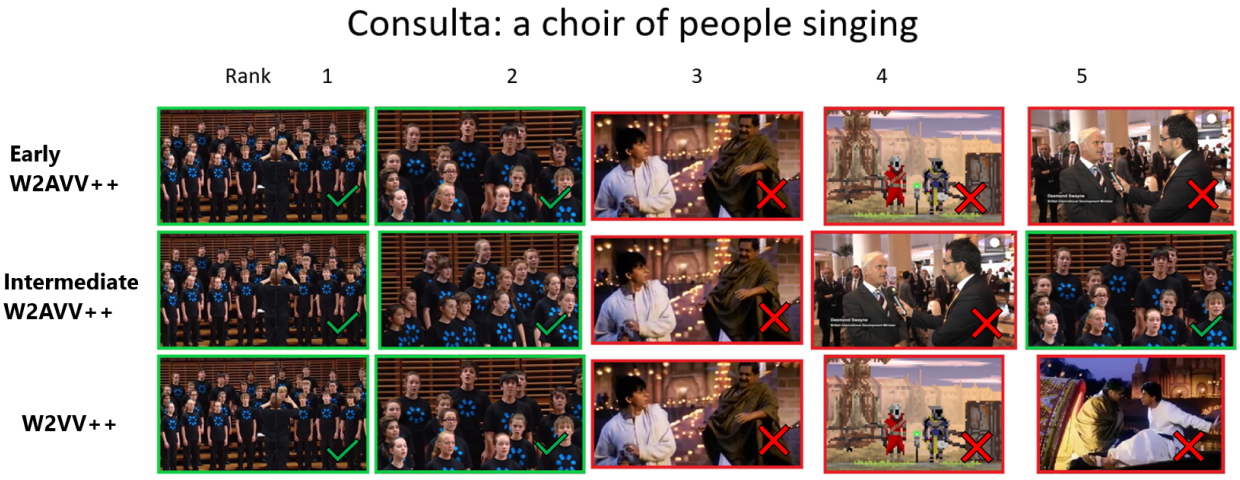


Figura A.7: Videos recuperados en la consulta 7 del experimento audiovisual.

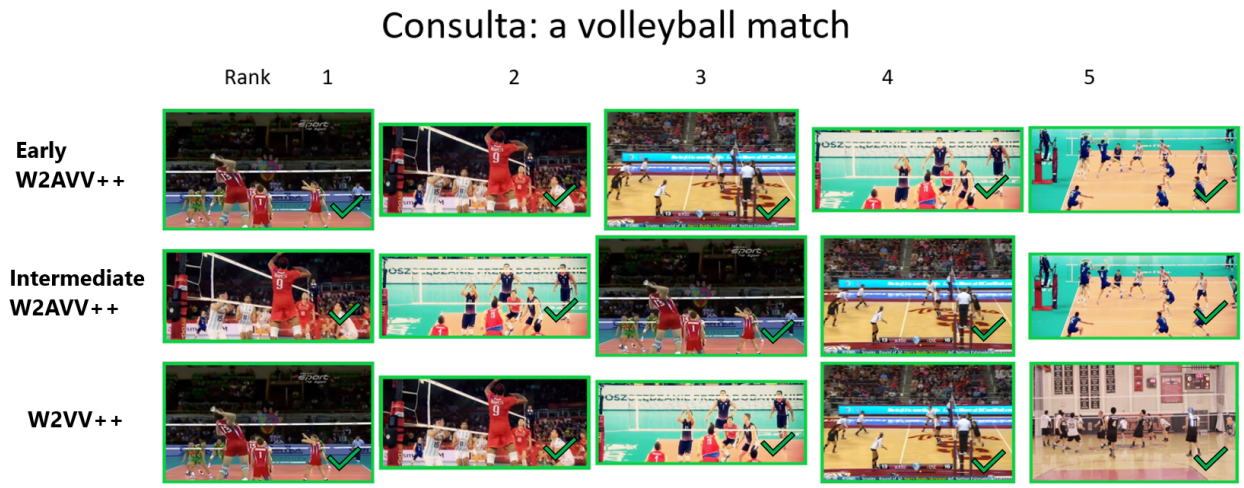


Figura A.8: Videos recuperados en la consulta 8 del experimento audiovisual.

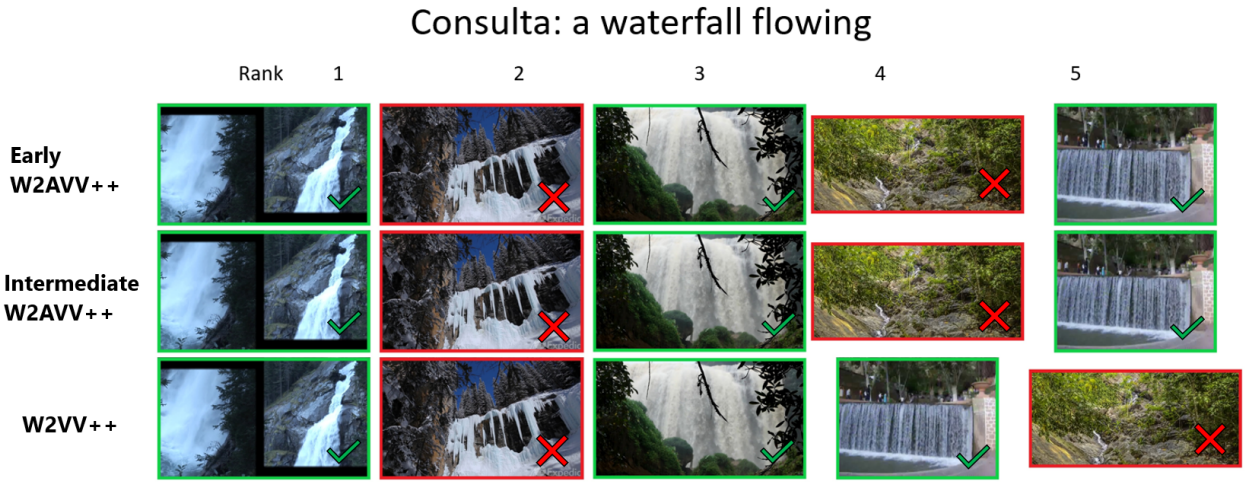


Figura A.9: Videos recuperados en la consulta 9 del experimento audiovisual.

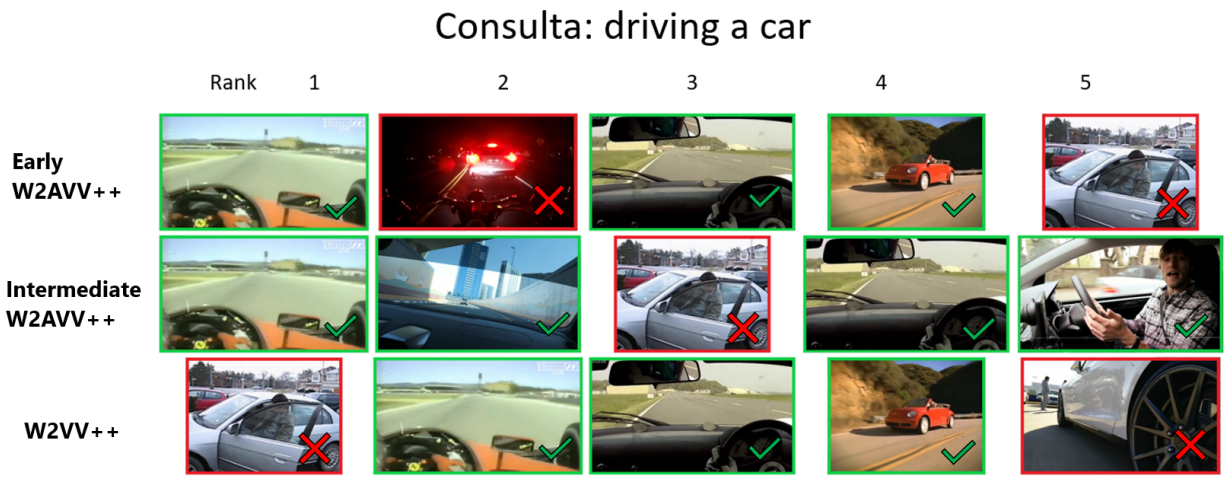


Figura A.10: Videos recuperados en la consulta 10 del experimento audiovisual.

Figura A.11: Videos recuperados en la consulta 11 del experimento audiovisual.

Figura A.12: Videos recuperados en la consulta 12 del experimento audiovisual.

Figura A.13: Videos recuperados en la consulta 13 del experimento audiovisual.

Figura A.14: Videos recuperados en la consulta 14 del experimento audiovisual.

Figura A.15: Videos recuperados en la consulta 15 del experimento audiovisual.

Figura A.16: Videos recuperados en la consulta 16 del experimento audiovisual.

Figura A.17: Videos recuperados en la consulta 17 del experimento audiovisual.

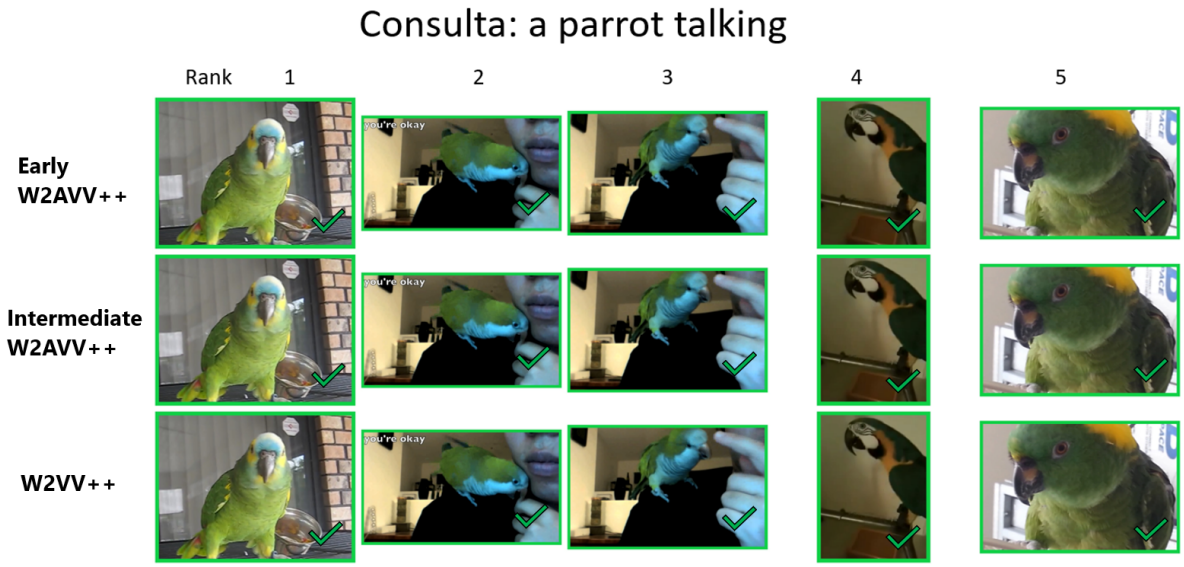


Figura A.18: Videos recuperados en la consulta 18 del experimento audiovisual.

Figura A.19: Videos recuperados en la consulta 19 del experimento audiovisual.

Figura A.20: Videos recuperados en la consulta 20 del experimento audiovisual.

Anexo B

Ad-hoc Video Search (AVS) 2019

B.1. Equipos participantes

ID Equipo	Organización	Región
ATL	Alibaba group, Zhejiang University	Asia
RUCMM	Renmin University of China Zhejiang Gongshang University	Asia
WasedaMeiseiSoftbank	Waseda University Meisei University SoftBank Corporation	Asia
Inf	Monash University (Australia) Renmin University (China) Shandong University (China)	Norteamérica, Asia y Australia
VIREO	City University of Hong Kong	Asia
kindai_kobe	Department of Informatics, Kindai University Graduate School of System Informatics, Kobe University	Asia
FIU_UM	Florida International University	Norteamérica
IMFD_IMPRESEE	**Millennium Institute Foundational Research on Data (IMFD) Chile Impresee Inc ORAND S.A. Chile**	**Latinoamérica**
SIRET	Charles University	Europa
EURECOM	EURECOM	Europa

Tabla B.1: Nombres e identificadores de los equipos participantes en la competencia AVS 2019. Nuestro equipo corresponde a IMFD_IMPRESEE.

B.2. Resultados de las cuatro entregas realizadas

Query ID	AudioV05 (Run 1)		Visual05 (Run 2)		AudioV02 (Run 3)		Visual02 (Run 4)	
	infAP	**Posición**	**infAP**	**Posición**	**infAP**	**Posición**	**infAP**	**Posición**
611	0,075	20/47	**0,080**	**17/47**	0,003	43/47	0,018	28/47
612	**0,038**	**34/47**	0,030	37/47	0,036	35/47	0,029	38/47
613	0,004	37/47	0,003	40/47	**0,005**	**36/47**	0,004	38/47
614	0,012	35/47	**0,013**	**34/47**	0,009	38/47	0,009	39/47
615	**0,004**	**31/47**	0,003	32/47	0,003	33/47	0,001	38/47
616	0,001	39/47	0,001	38/47	**0,002**	**31/47**	0,002	32/47
617	0,001	39/47	**0,003**	**32/47**	0,001	40/47	0,002	34/47
618	0,093	43/47	0,071	44/47	**0,135**	**34/47**	0,114	37/47
619	0,008	41/47	**0,010**	**40/47**	0,000	46/47	0,000	47/47
620	**0,268**	**8/47**	0,254	10/47	0,222	11/47	0,208	12/47
621	0,031	39/47	0,032	38/47	**0,053**	**36/47**	0,045	37/47
622	**0,019**	**38/47**	0,018	40/47	0,016	43/47	0,016	44/47
623	0,078	34/47	**0,104**	**31/47**	0,048	36/47	0,048	37/47
624	**0,008**	**42/47**	0,000	47/47	0,005	43/47	0,005	44/47
625	0,083	28/47	0,086	27/47	0,066	32/47	**0,093**	**26/47**
626	0,074	36/47	0,073	37/47	0,092	34/47	**0,129**	**32/47**
627	0,015	33/47	**0,015**	**32/47**	0,010	38/47	0,010	39/47
628	0,005	28/47	0,005	27/47	**0,015**	**22/47**	0,015	23/47
629	0,001	23/47	0,001	22/47	**0,004**	**9/47**	0,004	10/47
630	0,031	17/47	**0,031**	**16/47**	0,019	30/47	0,026	26/47
631	0,109	23/47	0,109	22/47	**0,138**	**16/47**	0,026	36/47
632	0,002	40/47	**0,002**	**39/47**	0,001	41/47	0,000	42/47
633	0,105	28/47	0,105	27/47	**0,117**	**25/47**	0,110	26/47
634	0,005	35/47	0,004	36/47	**0,017**	**22/47**	0,017	23/47
635	0,060	23/47	0,070	19/47	0,071	18/47	**0,074**	**17/47**
636	**0,025**	**21/47**	0,013	29/47	0,014	28/47	0,009	34/47
637	0,059	25/47	**0,165**	**17/47**	0,056	26/47	0,067	23/47
638	0,012	41/47	0,012	40/47	0,017	36/47	**0,018**	**34/47**
639	0,000	41/47	**0,000**	**40/47**	0,000	42/47	0,000	43/47
640	0,017	27/47	**0,054**	**14/47**	0,019	23/47	0,033	18/47

Tabla B.2: Listado de los resultados y posiciones obtenidas por las 4 entregas realizadas en cada consulta de AVS 2019.

B.3. Mean infAP de todos los participantes

ID Equipo_Run	Mean infAP
F_M_C_D_ATL.19_2	0,163
F_M_C_D_ATL.19_1	0,160
F_M_C_D_RUCMM.19_1	0,160
F_M_C_D_ATL.19_4	0,157
M_M_C_D_WasedaMeiseiSoftbank.19_2	0,153
F_M_C_D_RUCMM.19_2	0,152
M_M_C_D_WasedaMeiseiSoftbank.19_3	0,136
M_M_C_D_WasedaMeiseiSoftbank.19_1	0,133
F_M_C_D_RUCMM.19_4	0,127
F_M_C_D_RUCMM.19_3	0,124
F_M_C_D_WasedaMeiseiSoftbank.19_1	0,123
F_M_C_D_Inf.19_3	0,118
F_M_C_D_Inf.19_2	0,118
M_M_C_D_VIREO.19_2	0,118
F_M_C_D_Inf.19_4	0,117
F_M_C_D_Inf.19_1	0,117
M_M_C_D_WasedaMeiseiSoftbank.19_4	0,114
F_M_C_D_ATL.19_3	0,098
F_M_C_D_kindai_kobe.19_2	0,087
F_M_C_D_kindai_kobe.19_1	0,087
F_M_C_E_FIU_UM.19_4	0,082
F_M_N_D_kindai_kobe.19_5	0,081
F_M_C_D_kindai_kobe.19_3	0,080
F_M_C_E_FIU_UM.19_1	0,080
F_M_C_E_FIU_UM.19_3	0,079
F_M_C_E_FIU_UM.19_6	0,078
F_M_N_D_VIREO.19_5	0,075
F_M_C_D_VIREO.19_2	0,067
M_M_C_D_VIREO.19_1	0,066

ID Equipo_Run	Mean infAP
F_M_C_E_FIU_UM.19_2	0,065
F_M_N_E_FIU_UM.19_7	0,063
F_M_C_E_FIU_UM.19_5	0,061
F_M_C_D_VIREO.19_4	0,061
F_M_C_D_VIREO.19_3	0,060
F_M_C_D_kindai_kobe.19_4	0,059
F_M_C_D_IMFD_IMPRESEE.19_2 (Visual05)	**0,046**
F_M_C_D_IMFD_IMPRESEE.19_1 (AudioV05)	**0,041**
F_M_C_D_IMFD_IMPRESEE.19_3 (AudioV02)	**0,040**
F_M_C_D_IMFD_IMPRESEE.19_4 (Visual02)	**0,038**
M_M_C_A_SIRET.19_2	0,035
M_M_C_A_SIRET.19_3	0,035
M_M_C_A_SIRET.19_1	0,034
F_M_C_D_VIREO.19_1	0,033
M_M_C_A_SIRET.19_4	0,033
F_M_C_A_EURECOM.19_3	0,020
F_M_C_A_EURECOM.19_1	0,014
F_M_C_A_EURECOM.19_2	0,014

Tabla B.3: Listado de todas las entregas de cada equipo y su Mean infAP correspondiente.

B.4. Consultas de AVS 2019

1. **611**: a drone flying
2. **612**: a truck being driven in the daytime
3. **613**: a door being opened by someone
4. **614**: a woman riding or holding a bike outdoors
5. **615**: a person smoking a cigarette outdoors
6. **616**: a woman wearing a red dress outside in the daytime
7. **617**: one or more picnic tables outdoors
8. **618**: coral reef underwater
9. **619**: one or more art pieces on a wall
10. **620**: a person with a painted face or mask
11. **621**: person in front of a graffiti painted on a wall
12. **622**: a person in a tent
13. **623**: a person wearing shorts outdoors
14. **624**: a person in front of a curtain indoors
15. **625**: a person wearing a backpack
16. **626**: a race car driver racing a car
17. **627**: a person holding a tool and cutting something
18. **628**: a man and a woman holding hands
19. **629**: a black man singing
20. **630**: a man and a woman hugging each other
21. **631**: a man and a woman dancing together indoors
22. **632**: a person running in the woods
23. **633**: a group of people walking on the beach
24. **634**: a woman and a little boy both visible during daytime
25. **635**: a bald man
26. **636**: a man and a baby both visible
27. **637**: a shirtless man standing up or walking outdoors

28. **638**: one or more birds in a tree
29. **639**: inside views of a small airplane flying
30. **640**: a red hat or cap

B.5. Resultados de cada entrega con respecto a todos los participantes

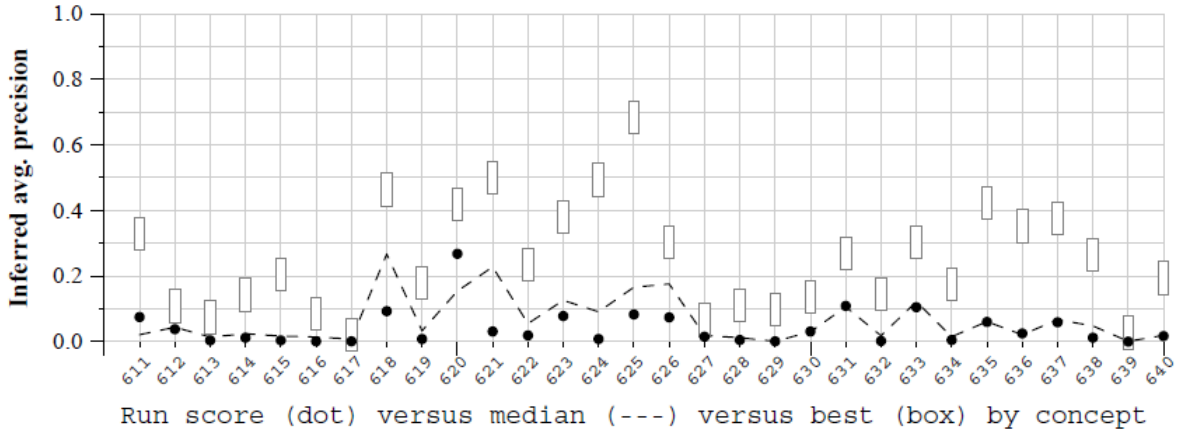


Figura B.1: Resultados obtenidos con `AudioV05`. La línea punteada indica la mediana de todas las entregas juntas, los rectángulos indican los mejores resultados y el punto negro indica el Mean infAP obtenido por `AudioV05`.

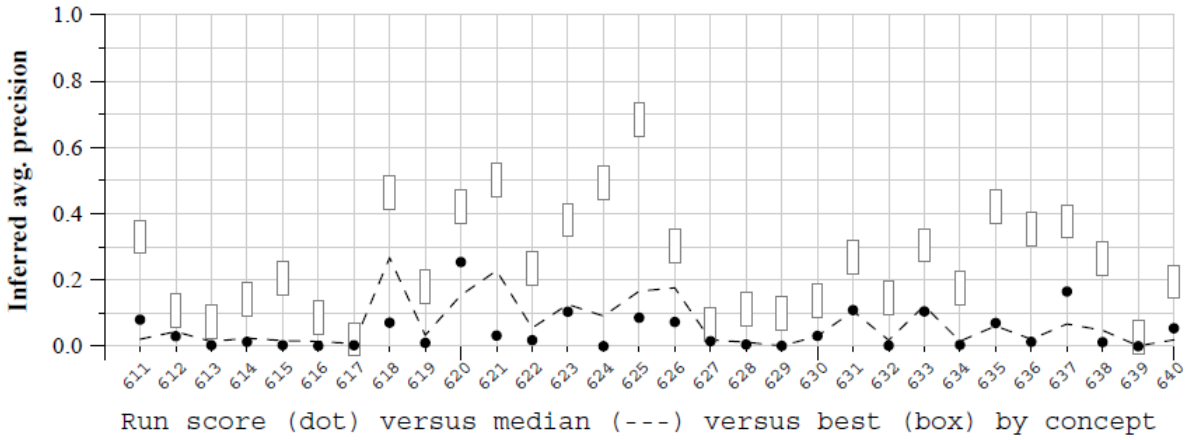


Figura B.2: Resultados obtenidos con `Visual05`. La línea punteada indica la mediana de todas las entregas juntas, los rectángulos indican los mejores resultados y el punto negro indica el Mean infAP obtenido por `Visual05`.

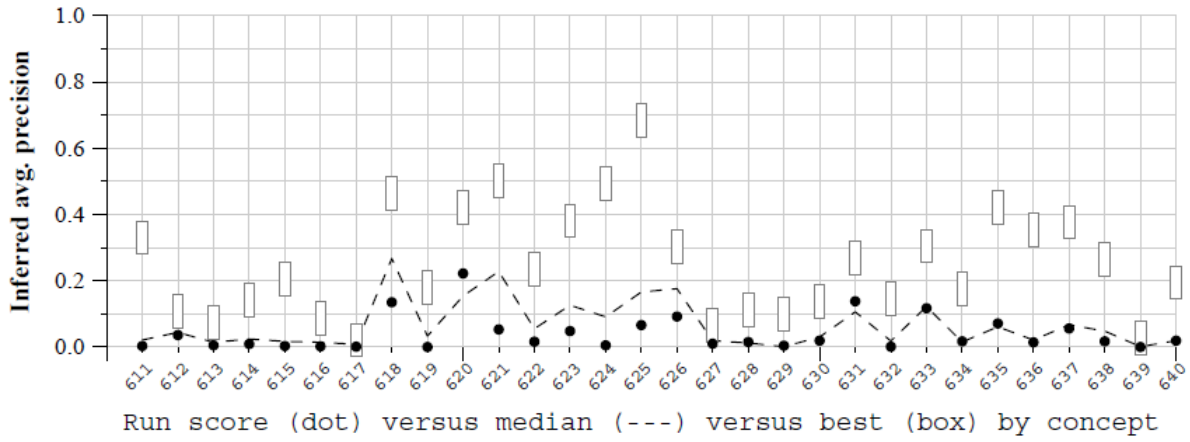


Figura B.3: Resultados obtenidos con `AudioV02`. La línea punteada indica la mediana de todas las entregas juntas, los rectángulos indican los mejores resultados y el punto negro indica el Mean infAP obtenido por `AudioV02`.

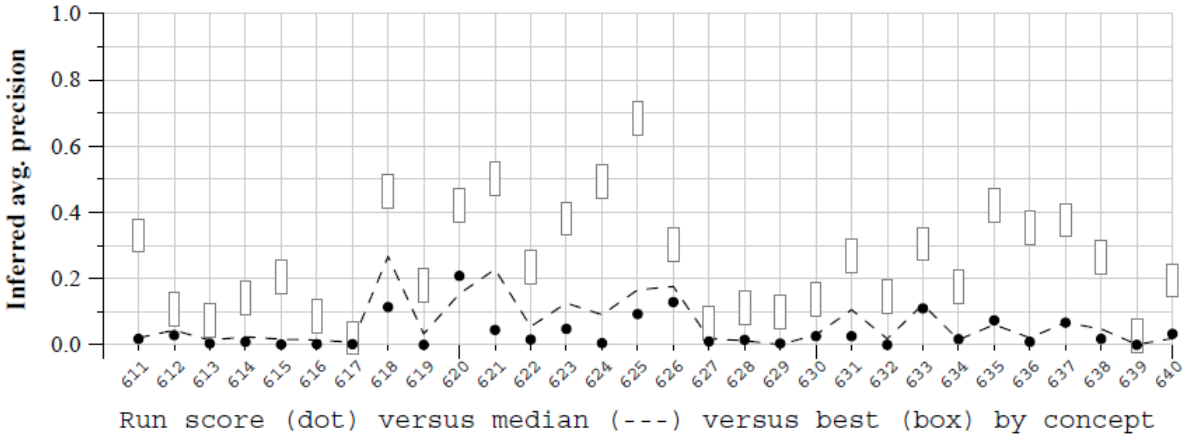


Figura B.4: Resultados obtenidos con `Visual02`. La línea punteada indica la mediana de todas las entregas juntas, los rectángulos indican los mejores resultados y el punto negro indica el Mean infAP obtenido por `Visual02`.