Universidad Zaragoza

Trabajo Fin de Grado

Predicción de saliencia en videos 360º mediante aprendizaje profundo

Saliency prediction in 360º videos with deep learning

Autor

Mateo Vallejo Domínguez

Directores

Edurne Bernal Berdún

Diego Gutiérrez Pérez

Titulación

Grado en Ingeniería Informática

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2022

AGRADECIMIENTOS

Gracias a mis directores Edurne y Diego por guiarme durante el desarrollo del proyecto, además de al resto de miembros del Graphics and Imaging Lab por su asistencia en diversas labores a lo largo del mismo. Gracias a mi familia y a mis amigos por su constante apoyo, y gracias a mis compañeros de promoción por su ayuda y compañía durante estos últimos 4 años.

Predicción de saliencia en videos 360º mediante aprendizaje profundo

RESUMEN

El desarrollo de tecnologías de realidad virtual está introduciendo múltiples avances en un gran número de industrias, como en el entretenimiento, la formación profesional y la medicina. Sin embargo, cómo diseñar y mostrar experiencias de forma atractiva, inmersiva, y cómoda para el usuario sigue siendo uno de los principales retos asociados a la realidad virtual, por lo que existe una necesidad de estudiar cómo los usuarios perciben y exploran estos entornos virtuales.

Para modelar el comportamiento visual de los usuarios, tradicionalmente se ha recurrido al estudio y análisis de las regiones que tienden a llamar la atención de los usuarios, denominadas regiones salientes. El campo de investigación de predicción de saliencia se encarga de estudiar y predecir la atención del sistema visual humano modelando las probabilidades de recibir fijaciones oculares según los estímulos visuales recibidos. A la hora de tratar de predecir la saliencia de contenido de realidad virtual, los modelos de predicción de saliencia para pantallas tradiciones no se adaptan correctamente a visores de realidad virtual, ya que los usuarios solo ven una región limitada del contenido total y pueden decidir mirar en direcciones concretas. De manera similar, los modelos de predicción de saliencia para imágenes estáticas no se adaptan correctamente a vídeos, ya que información contenida en fotogramas previos podría afectar a la saliencia de fotogramas posteriores, como ocurre en el seguimiento de objetos en movimiento. Es por ello que para la tarea de predicción de saliencia en vídeos inmersivos 360º es necesario desarrollar modelos específicos adaptados a las condiciones de visualización de éstos.

A lo largo del desarrollo del proyecto se ha implementado un modelo de red neuronal basado en técnicas actuales de aprendizaje profundo para abordar la tarea de predicción de saliencia en vídeos 360º, prestando especial atención a la dimensión temporal de los vídeos, que parece tener un papel fundamental en la atención visual humana. Comparando el modelo desarrollado con modelos actuales del estado del arte, se han obtenido resultados mejores en todas las métricas empleadas, a la vez que mostrando un comportamiento similar al que se puede ver en observadores reales, lo cual refleja la habilidad del modelo propuesto para imitar el comportamiento visual humano.

Índice

Capítulo 1

Introducción

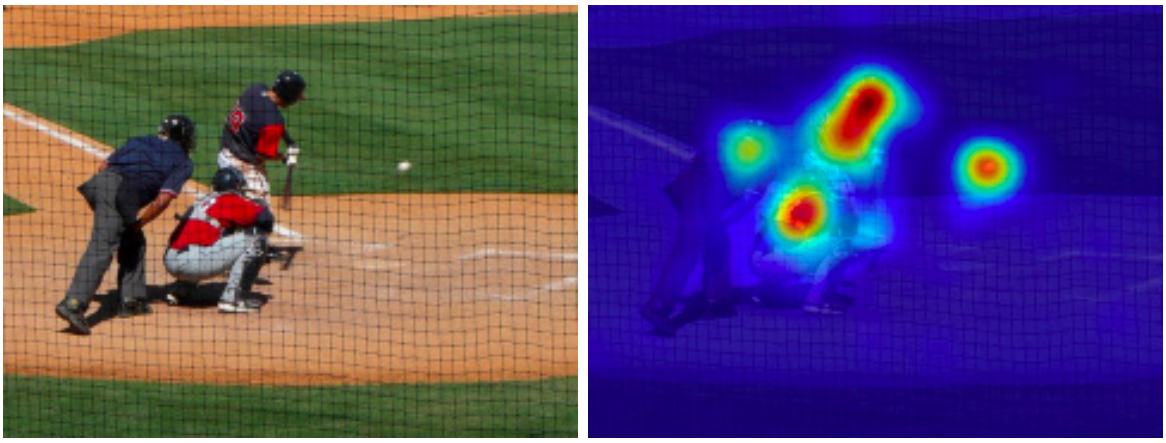
1.1. Contexto del proyecto

La realidad virtual describe el conjunto de tecnologías que permiten crear entornos virtuales en los que los usuarios pueden explorar y sentirse inmersos en escenarios simulados. Esta tecnología permite nuevos métodos de interacción y abre las puertas a nuevos formatos en la industria del entretenimiento, además de sus usos en otros sectores, donde la capacidad de hacer simulaciones realistas en entornos controlados facilita muchas tareas en campos como la educación, la formación profesional y la medicina. El sector de la realidad virtual se encuentra actualmente en los comienzos de su popularización, donde se encuentra en incremento la inversión dedicada a investigar y desarrollar estas tecnologías con el objetivo de formar la base del futuro del sector. Lejos de estar completamente establecida, la realidad virtual todavía presenta numerosos retos, como los relacionados con el desarrollo de hardware específico para la visualización de contenido VR, para el cual se necesitan pantallas de alta resolución y alta frecuencia de refresco. Además, el desarrollo de contenido para realidad virtual también introduce una serie de retos no presentes en otros contenidos tradicionales, ya que los usuarios tienen control de la cámara en todo momento por lo que los diseñadores de contenido deben desarrollar técnicas para guiar la atención de los usuarios. Es por ello que es necesario investigar la manera en la que los usuarios interactúan con esta tecnología, estudiando cómo los humanos perciben y exploran estos entornos virtuales.

Como primer paso en la comprensión de los múltiples factores que influyen en la percepción humana, se ha recurrido tradicionalmente al modelado de la atención visual, también llamado saliencia. Las regiones de un estímulo visual capaces de atraer la atención humana, denominadas áreas salientes, pueden identificarse mediante la captura y estudio de datos de la mirada provenientes de varios usuarios reales. A partir de los datos de la mirada, los cuales indican los puntos exactos donde los

observadores han dirigido su mirada, denominados fijaciones, se pueden establecer como regiones salientes aquellas que concentren un gran número de fijaciones oculares por parte de los usuarios. No obstante, la captura de datos de la mirada provenientes de distintos observadores no es una labor trivial, requiere tiempo, hardware muy específico y necesidad de múltiples pruebas de usuario, lo cual no es factible en la mayoría de los casos. Es por ello que el desarrollo de modelos capaces de predecir la saliencia establecen una atractiva alternativa, ya que son capaces de ofrecer una aproximación de la saliencia de un estímulo visual sin necesidad de capturar datos de la mirada.

Sin embargo, la predicción de saliencia se trata de una labor compleja debido a que la cognición humana juega un gran papel en la atención visual, por lo que no es suficiente analizar los factores de bajo nivel de las imágenes como los colores, el contraste y las formas, sino que también entran en juego factores mucho más complejos como el conocimiento previo de los observadores o el contexto y las condiciones de visualización. Esto hace que, a pesar de existir patrones comunes en el comportamiento de los usuarios, también existe una gran variabilidad entre observadores. Es por ello que numerosos trabajos en el estado del arte han optado por emplear redes neuronales (explicadas en detalle en el Anexo A.1) entrenadas para generar mapas de saliencia correspondientes a la probabilidad de fijación visual en los distintos puntos de las imágenes de entrada. En la Figura 1.1 se puede observar un ejemplo de un mapa de saliencia junto a su imagen original, donde se muestra mediante un mapa de calor las regiones más salientes de la imagen.



(a) Imagen original (b) Mapa de saliencia capturado

Figura 1.1: Imagen original junto a su mapa de saliencia capturado, donde la saliencia se representa como una distribución de probabilidad mediante un mapa de calor, siendo las áreas rojas las más salientes y por tanto presentan una mayor probabilidad de recibir una fijación ocular por parte del usuario [1].

En los últimos años han desarrollado un gran número de modelos de predicción

de saliencia en imágenes tradicionales, pero para contenido 360º se emplean visores de realidad virtual, donde factores como la orientación del visor y las restricciones físicas de las usuarios influyen en la fijación visual de los contenidos de las escenas. Es por ello que los modelos de predicción de saliencia para imágenes visualizadas en monitores tradicionales no se adaptan correctamente a contenido 360º. Además, la dimensión temporal de los vídeos también codifica información vital, ya que eventos pasados pueden influir en la atención de los usuarios en eventos futuros, como ocurre en el seguimiento de objetos en movimiento o por eventos relacionados con la narrativa de de los vídeos. Debido a esto los modelos de predicción de saliencia de imágenes estáticas tampoco se adaptan correctamente a vídeos con múltiples fotogramas. Sin embargo, como sucede en el caso de las imágenes, los modelos de saliencia diseñados para vídeos visualizados en monitores tradicionales que sí consideran esta dimensión temporal, tampoco se adaptan a las peculiaridades presentes en los vídeos 360º. Por ello existe una necesidad de desarrollar modelos específicos para la predicción de saliencia en vídeos de 360º que se adapten a la naturaleza de su formato. El desarrollo de estos modelos podría ser empleado en diversas aplicaciones como la generación de contenido de realidad virtual, compresión de vídeo adaptativo a regiones salientes, o el desarrollo de nuevas técnicas para la creación de vídeos 360º, como la reorientación de contenido para minimizar los movimientos de usuarios [2].

Hasta el momento, los modelos de predicción de saliencia en 360º mayoritariamente han modelado las dependencias espaciales mediante redes neuronales convolucionales [1, 3], mientras que la dimensión temporal no ha sido aún tan explorada, obviando por tanto matices tan relevantes en la atención humana como puede ser el movimiento de los objetos. Es por ello que para el desarrollo de estos modelos, se deben investigar nuevas arquitecturas y diseños que permitan abordar estos problemas, adaptándose a las características del contenido representando tanto las dependencias espaciales como temporales.

En el campo de procesamiento de lenguaje natural se encuentra en auge el desarrollo de modelos basados en la arquitectura del Transformer [4], definiendo el estado del arte en tareas como la traducción, la compresión lectora o la generación de texto [5]. Esto es debido a que la arquitectura del Transformer es capaz de modelar las dependencias temporales existentes entre las distintas palabras mediante su mecanismo de atención global. Aparte de su uso en procesamiento de lenguaje natural, recientemente se encuentran en incremento sus aplicaciones en otros campos, como en el análisis de secuencias en proteínas [6] o la clasificación de imágenes [7] y de vídeo [8], donde se emplea el mecanismo de la atención global para modelar

las dependencias espacio-temporales de los vídeos. Viendo el éxito de las recientes aplicaciones de Transformers en otros campos, en este proyecto se ha considerado que se podría emplear esta arquitectura en el campo de predicción de saliencia, aplicando el concepto de atención global para encontrar las relaciones entre los distintos fotogramas para inferir información sobre las dependencias temporales de estos. Tras diseñar, implementar, entrenar y evaluar un modelo de predicción de saliencia basado en la arquitectura Transformer, se ha visto que estas relaciones temporales logran modelar comportamientos de atención más complejos como pueden ser el seguimiento de los objetos en movimiento o la trama del vídeo. Además, se ha comparado este modelo con respecto a un trabajo del estado del arte, obteniendo mejores resultados para una variedad de métricas de evaluación de mapas de saliencia, y mostrando resultados cercanos al comportamiento real de los usuarios.

1.2. Objetivos del proyecto

El objetivo de este trabajo es por tanto realizar una implementación de un modelo de predicción de saliencia capaz de modelar las dependencias tanto espaciales como temporales de vídeos 360º. En concreto se busca estudiar cómo se puede adaptar el modelo del Transformer para abordar esta tarea, aprovechando las capacidades de cómputo de atención global inherentes a su diseño para representar las dependencias temporales. Para ello se han definido los siguientes objetivos:

- Estudio sobre predicción de saliencia y la arquitectura de los Transformer (Capítulo 2).

- Diseño e implementación de un modelo de saliencia (Capítulo 3).

- Evaluación del modelo desarrollado (Capítulo 4).

Este proyecto se ha llevado a cabo en el grupo de investigación *Graphics and Imaging Lab*, en la Universidad de Zaragoza. El trabajo del grupo se centra en los gráficos por computador, realizándose investigación en áreas de renderizado físicamente correcto, procesamiento de imágenes, fotografía computacional, realidad virtual y percepción aplicada, entre otros. El trabajo del grupo también incluye técnicas de seguimientos de mirada, aprendizaje profundo y conceptos como la saliencia.

1.3. Planificación y herramientas

El trabajo se ha desarrollado mediante el lenguaje de programación *Python*, haciendo uso de *PyTorch*, un framework de aprendizaje automático, realizando un control de versiones mediante *Github*. Para el procesamiento de los datos de vídeo se ha empleado la herramienta *FFmpeg*. El entrenamiento final del modelo se ha llevado a cabo en una GPU *Nvidia Quadro P6000* con 24 GB de memoria facilitada por el *Graphics and Imaging Lab* de la Universidad de Zaragoza.

El desarrollo del trabajo ha sido dividido en varias tareas: Primero se ha realizado un estudio previo sobre el estado del arte en aprendizaje automático, implementando un primer modelo de prueba de clasificación de vídeo con el objetivo de familiarizarse con el framework *PyTorch*. Una vez familiarizado con las herramientas se se ha comenzado el desarrollo del modelo de predicción de saliencia en vídeos 360º. Para ello primero se ha realizado el diseño del modelo, adaptando el modelo implementado previamente, y después se han realizado múltiples iteraciones sobre el modelo modificando sus parámetros y mejorando algunos aspectos de diseño. Por último, se han comparado los resultados obtenidos del modelo desarrollado con otros modelos de predicción de saliencia en 360º. Adicionalmente, de manera paralela al desarrollo y evaluación del modelo se ha realizado la redacción del documento presente. En la Tabla 1.1 se recogen las horas dedicadas a las distintas tareas mencionadas, y en la Figura 1.2 se puede observar el diagrama de Gantt del trabajo.

Tareas		Horas Dedicadas	
Trabajo previo	Estudio de conceptos	26	54
	Familiarización con las herramientas	6	
	Implementación de modelo de prueba	22	
Desarrollo del modelo	Diseño	22	158
	Implementación	136	
Evaluación final	Evaluación de distintas arquitecturas	10	56
	Evaluación y comparativa del modelo final	46	
Redacción del documento		83	
Totales		351	

Tabla 1.1: Horas totales dedicadas al desarrollo del trabajo.

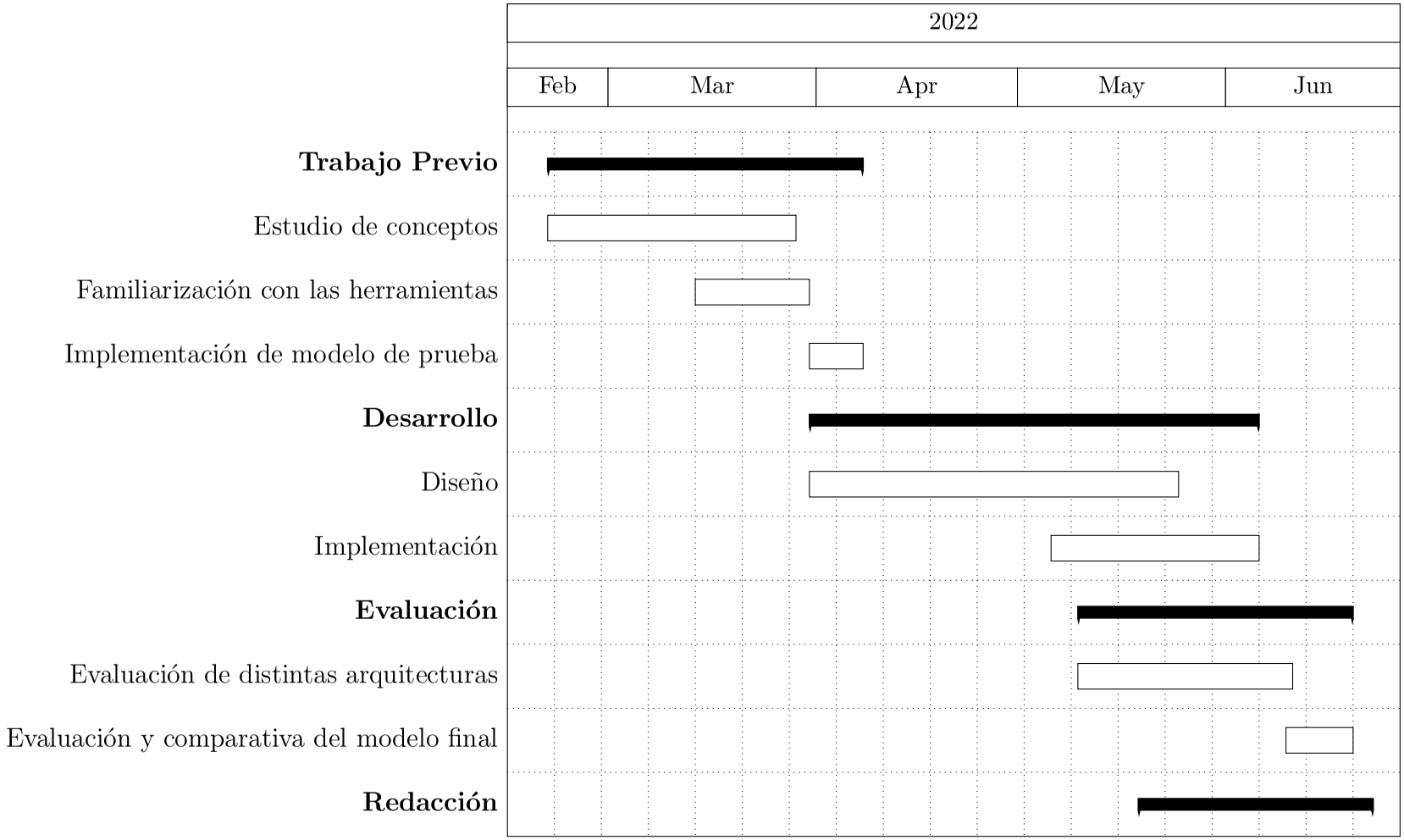


Figura 1.2: Diagrama de Gantt del trabajo.

Capítulo 2

Trabajo relacionado

2.1. Aprendizaje automático

El campo de aprendizaje automático se encarga de desarrollar sistemas capaces de resolver problemas sin haber sido explícitamente programados para su resolución. Esto se realiza mediante técnicas que emplean datos de entrada para ajustar de manera automática los sistemas desarrollados, con el objetivo de producir los datos de salida correctos.

Uno de los sistemas de aprendizaje automático más empleado son las redes neuronales artificiales, cuya estructura está inspirada en redes neuronales biológicas. Estas redes neuronales emplean diferentes capas de neuronas conectadas entre sí que aprenden en conjunto a generar las salidas deseadas según los estímulos de entrada. Existen distintos tipos de capas que pueden encontrarse conectadas de diferentes formas. Se dice que una red es *feed-forward* cuando las conexiones de las neuronas no forman un ciclo, esto es, las neuronas solo se encuentran conectadas a otras neuronas de capas posteriores, haciendo que la información fluya en una única dirección. Un ejemplo especial de *feed-forward*, son las capas de neuronas densamente conectadas, donde cada neurona de una capa se encuentra conectada a todas las neuronas de la siguiente capa. Se puede consultar una explicación detallada de las redes neuronales en el Anexo A.1.

En contraposición a las *feed-forward*, se llaman redes neuronales recurrentes a aquellas que sí contienen un ciclo donde una neurona se conecta a otra de una capa previa. Estas redes son capaces de almacenar información y tener en cuenta el estado de la red en el paso anterior a la hora de generar las salidas. Estas redes neuronales recurrentes se pueden emplear para modelar dependencias temporales. Un ejemplo de red neuronal recurrente que comúnmente se utiliza para representar estas dependencias

temporales es *Long-Short-Term-Memory* (LSTM) detallada en el Anexo A.1.2.

Una aplicación especial de las redes neuronales es en el aprendizaje profundo o *deep learning*, donde se emplean múltiples capas para extraer información a distintas escalas, donde las primeras capas extraen las características básicas de los datos de entrada y las capas subsecuentes son capaces de emplear esa información para realizar un procesamiento más complejo.

Un ejemplo de arquitectura de aprendizaje profundo comúnmente empleada en tareas de visión por computador son las redes neuronales convolucionales (ver Anexo A.1.1), que empleando filtros entrenables son capaces de detectar patrones y obtener características sobre regiones de los datos de entrada. Esto además permite reducir el número de parámetros necesarios para operar con los datos de entrada.

Además, una vez decidida la arquitectura de una red neuronal se debe entrenar a partir de los datos de entrenamiento, donde se deben considerar múltiples parámetros que afectan al rendimiento final del modelo, como pueden ser el tamaño de los datos de entrada, la función de coste empleada que determine cuándo una salida es correcta y cuando no, el número de iteraciones de entrenamiento (épocas) que realizar, etc.

2.2. Predicción de Saliencia

Se define como saliencia la cualidad que tiene un estímulo visual en concentrar la atención del sistema visual humano. La predicción de saliencia por tanto consiste en estimar las probabilidades de fijación visual por parte de los observadores para las distintas regiones de imágenes, lo cual permite estudiar la percepción humana.

Los primeros trabajos en abordar la predicción de saliencia recurrieron a métodos basados en heurísticas que obtenían mapas de saliencia a partir de factores de bajo nivel de las imágenes, como los colores, la intensidad y el contraste [9, 10]. Estos modelos, aunque sí eran capaces de destacar regiones llamativas de las imágenes de entrada, no eran capaces de imitar el comportamiento visual humano, ya que no todas estas regiones son salientes ni presentan la misma saliencia. Una de las principales limitaciones de estos modelos es su falta de conciencia semántica, lo que los aleja del comportamiento humano en el que las imágenes son fuertemente analizadas en términos de instancias (árbol, animal, persona, etc.) además de características de bajo nivel (borde, brillo, contraste, etc.).

El desarrollo de nuevas técnicas en el campo de aprendizaje profundo supuso un gran avance para la predicción de saliencia, donde el desarrollo de redes neuronales

convolucionales capaces de reconocer elementos complejos de más alto nivel, como objetos concretos o caras, permitieron desarrollar modelos que resolvían exitosamente los casos donde los métodos anteriores fallaban [1, 11].

Estos modelos estaban centrados en obtener predicciones de mapas de saliencia de imágenes estáticas, pero también se empezó a investigar su aplicación en vídeos, empleando redes neuronales recurrentes para modelar las dependencias temporales de los fotogramas [12, 13].

El desarrollo de la tecnología de realidad virtual, despertó un interés por estudiar cómo los usuarios perciben y exploran los entornos virtuales [2], por lo que comenzaron a desarrollarse modelos de predicción de saliencia para imágenes 360º. Algunos de estos adaptaban modelos previos diseñados para contenido tradicional 2D [14, 15] y otros desarrollaban modelos específicamente adaptados a estas nuevas condiciones de visualización [16, 3].

Por último, también se empezó a investigar la predicción de saliencia para vídeos 360º, donde se desarrollaron modelos que combinaban los conocimientos previos de los modelos de predicción de saliencia en imágenes 360º con los modelos de predicción de saliencia en vídeos, que por tanto utilizaban redes neuronales recurrentes para predecir la saliencia en vídeos 360º [17, 18].

Sin embargo, entre las problemáticas asociadas al uso estas arquitecturas, la más restrictiva a la hora de predicción de saliencia en vídeos es su limitada memoria, solo siendo capaces de establecer relaciones temporales entre unos pocos fotogramas de cada vídeo. Es por ello que en el presente trabajo se propone el uso de la arquitectura conocida como Transformer, ya que son capaces de inferir las dependencias espacio-temporales existentes entre secuencias de gran duración, no contando por tanto con esta restricción de memoria.

2.3. Transformers

Los llamados Transformers surgen originalmente en el campo de procesamiento de lenguaje natural, que busca desarrollar un entendimiento del lenguaje humano mediante modelos de aprendizaje automático. Debido a que el significado de una frase no viene dada por una palabra individual sino cómo esa palabra se relaciona con el resto de la frase, el campo de procesamiento de lenguaje natural se ha centrado en investigar maneras en las que modelar estas dependencias temporales. Para ello se han desarrollado modelos basados en redes neuronales recurrentes capaces

de almacenar información previa. Los modelos de redes neuronales recurrentes que mejores resultados obtenían hasta el momento empleaban mecanismos de atención como parte de su diseño, pero con la introducción del Transformer se demostró que empleando solo la atención global se podía prescindir de otros componentes recurrentes [4]. El Transformer es un modelo de aprendizaje profundo que emplea exclusivamente el mecanismo de atención global para resolver tareas de procesamiento de lenguaje natural, obteniendo resultados cercanos o mejores que aquellos otros modelos que empleaban redes neuronales recurrentes.

En la Figura 2.1 se puede observar la arquitectura del Transformer, donde se puede apreciar que sigue una estructura de encoder-decoder. El encoder se encarga de encontrar las relaciones de los distintos datos de entrada entre sí mediante mecanismos de atención global, mientras que el decoder emplea esta información para generar las salidas correspondientes. Una explicación en profundidad de los distintos componentes del Transformer se encuentra en el Anexo A.2.

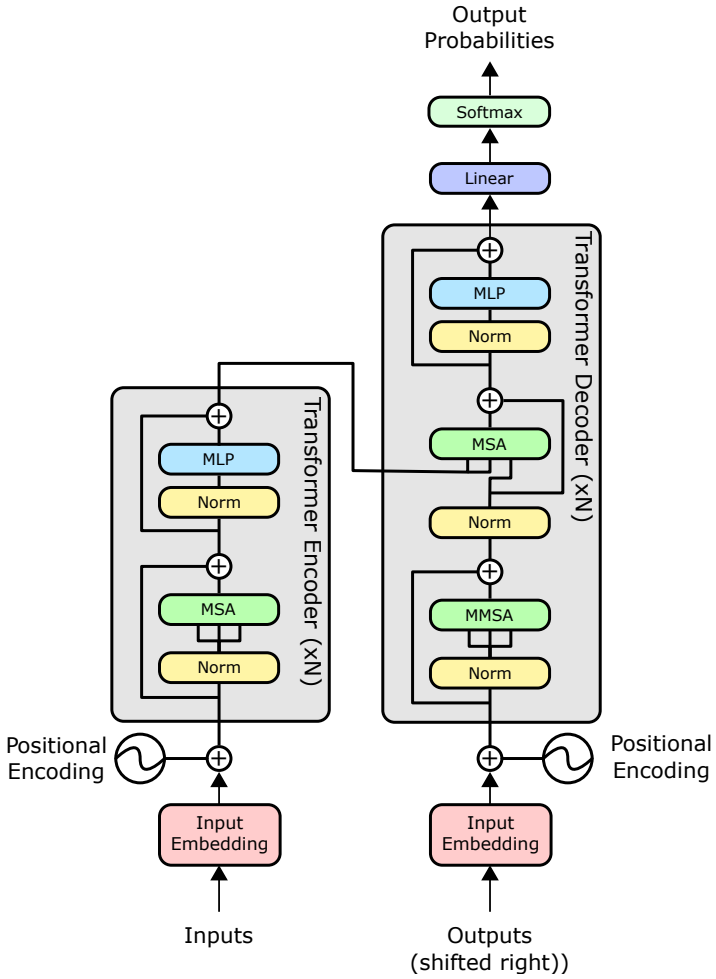


Figura 2.1: Arquitectura del Transformer formado por dos componentes principales: El encoder en la parte izquierda y el decoder en la parte derecha. Ver Anexo A.2 para más información.

Este modelo ha definido el estado del arte en procesamiento de lenguaje natural, donde hoy en día se continúan desarrollando modelos basados en la arquitectura de Transformer con gran éxito en diversas tareas de procesamiento de lenguaje natural [5, 19, 20]. Además, actualmente se encuentra en incremento el número de modelos basados en la arquitectura del Transformer para diferentes tareas, como la visión por computador [21, 7].

2.3.1. Video Vision Transformer (ViViT)

El Video Vision Transformer (ViViT) [8] se trata de un modelo de clasificación de vídeo cuya arquitectura está basada en el Vision Transformer (ViT) [7], un modelo de clasificación de imagen mediante Transformers, cuya arquitectura se puede observar en la Figura 2.2.

El modelo de ViT divide las imágenes de entrada en una secuencia de parches más pequeños $(x_1, x_2, ..., x_n)$ a los que aplica una proyección lineal para convertirlos en sus vectores de características correspondientes. Estos vectores de características, también llamados *tokens*, son procesados por una serie de capas de encoders del Transformer. Una vez estos tokens han sido procesados por los encoders, una capa de neuronas densamente conectada (MLP) genera como salida la probabilidad de que el vídeo pertenezca a cada clase.

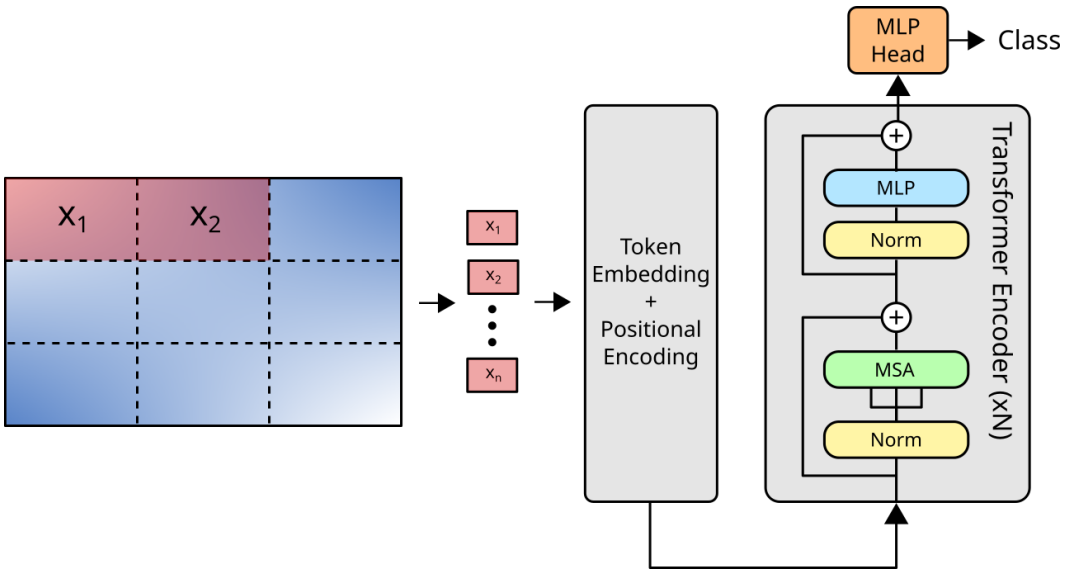


Figura 2.2: Diagrama del Vision Transformer (ViT). En la izquierda se puede ver el mecanismo de división de las imágenes de entrada en parches más pequeños que son proyectados y procesados por el modelo cuya arquitectura se puede observar en la parte derecha.

ViViT adapta esta arquitectura para clasificación de vídeo. Por tanto, la diferencia

principal entre este modelo y el anterior es en procesamiento de los los datos de entrada. ViT tiene como entrada una sola imagen donde se necesita extraer información espacial de las distintas regiones, mientras que ViViT tiene como entrada una secuencia de imágenes de las que se necesita extraer información tanto espacial como temporal. Los autores proponen dos métodos para tokenizar los videos de entrada, *uniform frame sampling* donde se extraen parches de cada fotograma de manera independiente que después son concatenados secuencialmente, y *tubelet embedding*, donde se divide el video en parches espacio-temporales que, a diferencia con el método anterior, se extienden en la dimensión temporal.

En la Figura 2.3 se encuentra un diagrama de ViViT. Se puede observar que la parte derecha, correspondiente con la arquitectura del modelo, es idéntica a la arquitectura de ViT, mientras que en la parte izquierda se representa el mecanismo de *tubelet embedding* mediante el cual se obtienen los distintos parches espacio-temporales que son proyectados para obtener los tokens de los datos de entrada.

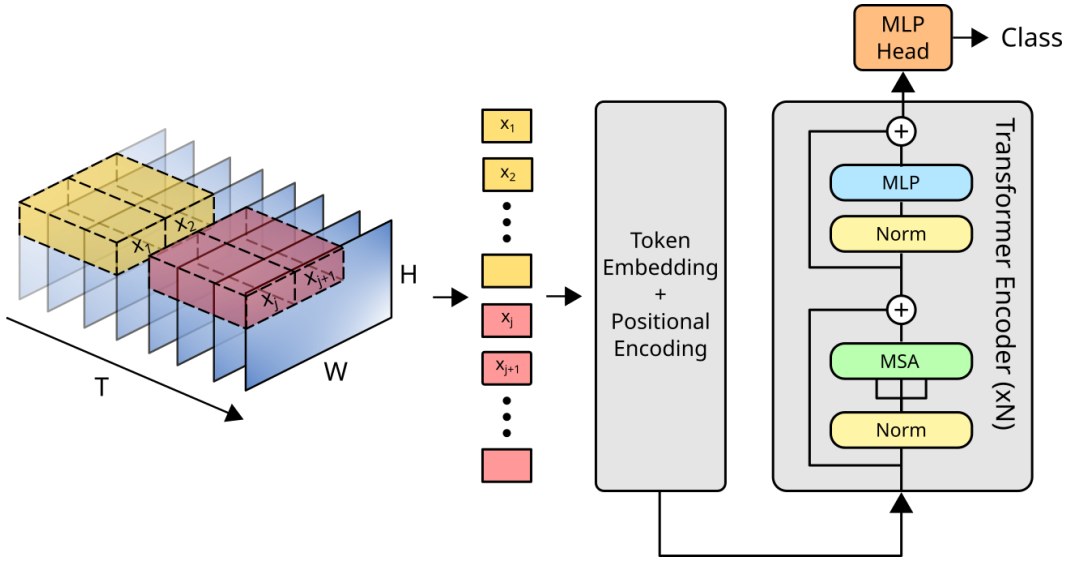


Figura 2.3: Arquitectura del Video Vision Transformer (ViViT). En la parte izquierda el mecanismo de *Tubelet Embedding* donde se extraen parches espacio-temporales que son proyectados a sus vectores de características correspondientes.

ViViT ha conseguido superar las puntuaciones obtenidas por modelos previos para múltiples datasets, demostrando que se puede utilizar el modelo del transformer para modelar las dependencias espacio-temporales en vídeos. Por ello, en el trabajo presente se propone emplear esta arquitectura como base sobre la que desarrollar un modelo de predicción de saliencia.

Capítulo 3

Modelo de predicción de saliencia

Se ha implementado un modelo de predicción de saliencia para vídeos 360º basado en el Transformer. El modelo implementado hace uso del mecanismo de atención global para codificar las dependencias espacio-temporales existentes entre las distintas regiones del vídeo a lo largo del tiempo. Estas características espacio-temporales aportan información vital al trata de identificar relaciones complejas existentes en los datos, como puede ser la relación entre el movimiento de los objetos y su saliencia asociada.

3.1. Diseño

El diseño del modelo se ha basado en la arquitectura del Video Vision Transformer (ViViT). Para ello se ha mantenido la estructura básica del modelo, principalmente el tratamiento de los datos de entrada y la capa de encoders, pero se ha modificado la salida del modelo, ya que para la tarea de clasificación tan solo es necesario generar una única salida, mientras que para la tarea de predicción de saliencia se necesita generar un fotograma de salida por cada fotograma de entrada.

En la Figura 3.1 se puede observar un diagrama del modelo, formado por tres secciones principales. Primero, se convierten los datos de entrada en los tokens, después se calcula la atención global de estos tokens mediante una serie de encoders, y por último se reconstruyen los mapas de saliencia a partir de los tokens procesados por los encoders.

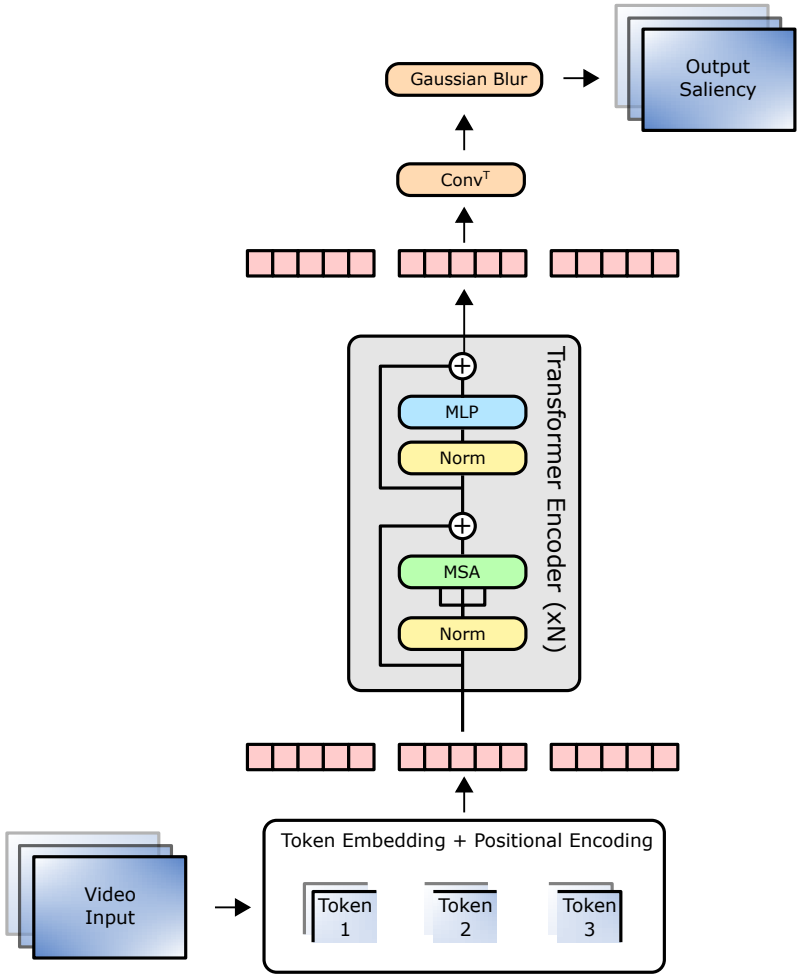


Figura 3.1: Arquitectura del modelo desarrollado

3.1.1. Token Embedding y Positional Encoding

Dado el conjunto de fotogramas de entrada, primero se dividen en parches espacio-temporales y se convierten a tokens mediante *tubelet embedding* (ver Sección 2.3.1), donde se aplica una convolución tridimensional para convertir cada parche en un vector unidimensional de características. Para más información sobre la operación de convolución ver Anexo A.1.1. Una vez obtenidos los tokens se aplica *positional encoding*, una operación que añade a cada token información sobre su posición en la secuencia de tokens. Estas modificaciones permitirán a los tokens cercanos a compartir más características y por tanto priorizarse entre sí, recibiendo menor influencia de tokens más lejanos tanto espacialmente como temporalmente. Tanto la convolución del token embedding como la información posicional del *positional encoding* se tratan de parámetros que aprenden junto a la red ajustando sus pesos. Una vez obtenidos los tokens, se agrupan en una matriz y se envían al primer encoder.

3.1.2. Transformer Encoder

El encoder del Transformer está formado por 3 componentes principales: Multi-Head-Self-Attention (MSA), Multi-Layer-Preceptron (MLP) y las conexiones residulales. MSA es el componente que se encarga de calcular la atención global de los tokens mediante el mecanismo de atención de múltiples cabezas detallado en el Anexo A.2. Este componente devuelve una matriz en la que se encuentran codificadas las relaciones entre cada token. Para hacer el cálculo de la atención global se necesita proporcionar antes las tres matrices de *Query* Q, *Key* K y *Value* V. Estas matrices se calculan a partir de los tokens de entrada X, con dimensión de token d_x, mediante sus matrices de peso W^Q, W^K y W^V aprendidas durante el entrenamiento. En la figura 3.2 se muestra un esquema de la obtención de las matrices Q, K, V para el cálculo de la atención.

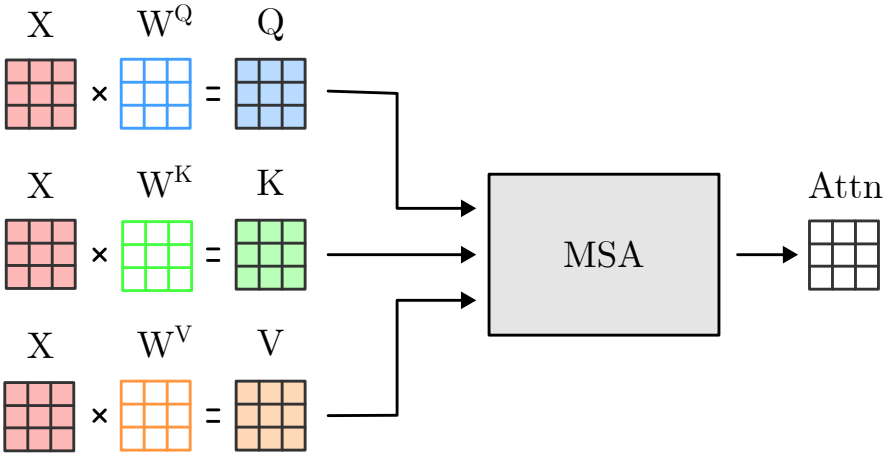


Figura 3.2: Cálculo de la matrices *Query* Q, *Key* K y *Value* V a partir de la matriz de tokens X y las matrices de pesos W^Q, W^K y W^V. El componente de *Multi-Head-Self-Attention* MSA emplea estas matrices para calcular la atención *Attn*.

De esta forma durante el entrenamiento se establecen los pesos que mejor representan y extraen la atención entre los distintos fotogramas. Por tanto la operación realizada por la MSA es

$$\text{Atención}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_x}}\right)V, \tag{3.1}$$

donde se aplica la función de activación softmax (Equación 3.2) para normalizar los datos y asegurarse que todos sean positivos y menores que 1.

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{d_x} e^{x_j}} \tag{3.2}$$

La capa MLP se trata de una capa densamente conectada, esto es una capa de neuronas todas conectadas entre sí, que realiza un procesamiento adicional sobre la atención

calculada en el paso anterior. Después de cada capa MSA y MLP se añaden conexiones residuales, que consisten en sumar los tokens de la capa anterior y normalizarlos. Esto permite propagar la información original del token, que contiene información de posición agregada durante el positional encoding. Si no se añadiesen estas conexiones residuales, después de calcular la atención por primera vez se perdería la información posicional. Los nuevos tokens generados por la salida del encoder se envían a la entrada del siguiente encoder, o en el caso de tratarse de la última capa, a la salida de los encoders.

3.1.3. Obtención de los mapas de saliencia

Una vez se ha calculado la atención global mediante los encoders se necesita convertir los tokens a los mapas de saliencia de salida. Para ello se aplica una convolución traspuesta, que realiza el proceso inverso del *token embedding*, convirtiendo los tokens en parches que después son juntados y reorganizados para formar los fotogramas de salida. Esta convolución traspuesta también se trata de un componente que aprende junto a la red.

Debido a que la reconstrucción de cada token se realiza de manera independiente, se ha observado que se pierde información local en los bordes de los parches, causando discontinuidades visibles en los mapas de saliencia resultantes. En la Figura 3.3 se puede observar un ejemplo de un mapa de saliencia obtenido mediante un modelo temprano del trabajo, donde se pueden distinguir claramente los bordes de los parches debido a las discontinuidades que generan en el mapa de saliencia. Es por ello que se ha añadido un filtro gaussiano al final de la reconstrucción de los mapas de saliencia para aliviar las discontinuidades de los bordes.

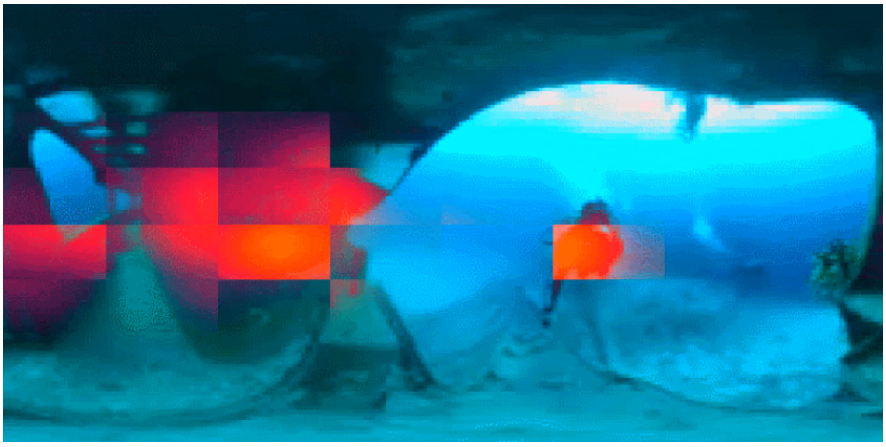


Figura 3.3: Ejemplo de artefactos en un mapa de saliencia generado, donde las regiones salientes (marcadas en color rojo) contienen discontinuidades visibles en los bordes de los parches causados por la pérdida de información local entre parches.

3.2. Detalles de entrenamiento e implementación

El modelo ha sido desarrollado en el lenguaje de programación *Python* haciendo uso del framework *PyTorch*, y ha sido implementado para su ejecución en GPU.

3.2.1. Dataset

Para el entrenamiento del modelo se ha utilizado el dataset de VR-Eyetracking [22], formado por 208 vídeos 360º junto a sus respectivos datos de fijaciones oculares capturadas de un mínimo de 31 participantes por vídeo. También se ha empleado el dataset Sports-360 [23], formado por 104 vídeos 360º junto a sus respectivos datos de fijaciones oculares. Durante el desarrollo del proyecto se ha implementado un programa para transformar el dataset, generando un dataset procesado que contiene los vídeos reescalados y sus mapas de saliencia correspondientes. El modelo final utiliza vídeos en resolución 512×256 píxeles con 16 fotogramas de longitud, por lo que se han adaptado los vídeos del dataset, reescalándolos y dividiéndolos en fragmentos más pequeños formados por secuencias de fotogramas muestreados a una frecuencia de 3 fotogramas por segundo. Con esto, a partir de cada vídeo se pueden extraer múltiples secuencias, con lo que se ha conseguido un total de 1561 donde cada una comprende unos 5,33 segundos en tiempo real. Se ha seleccionado esta frecuencia de muestreo debido a que el tamaño de las secuencias se encuentra limitado por el tamaño de la memoria del hardware y de aumentar la frecuencia de muestreo sin aumentar la longitud de la secuencia, los fragmentos comprenderían regiones muy pequeñas en el tiempo que no serían representativas de la acción del vídeo.

Para obtener los mapas de saliencia *ground truth* asociados a cada fotograma de las secuencias se han agrupado las coordenadas de fijaciones de los usuarios para sus respectivos fotogramas. A partir de estas fijaciones se han generado los mapas de saliencia agregando por cada punto de fijación un distribución normal centrada en dichas coordenadas. Debido a que se tratan de vídeos 360º representados mediante una proyección equirectangular se ha aplicado una distorsión a las distribuciones normales para mantener su representación en coordenadas equirectangulares. Esto tiene el efecto de estirar horizontalmente los contenidos del mapa de saliencia a medida a la que se acercan a los extremos inferiores y superiores del fotograma correspondientes con los polos de la esfera. Con esto se forman los mapas de saliencia *ground truth*, que representan la atención visual de los fotogramas que se buscan generar mediante el modelo. En la Figura 3.4 se puede observar un diagrama de su proceso de obtención.

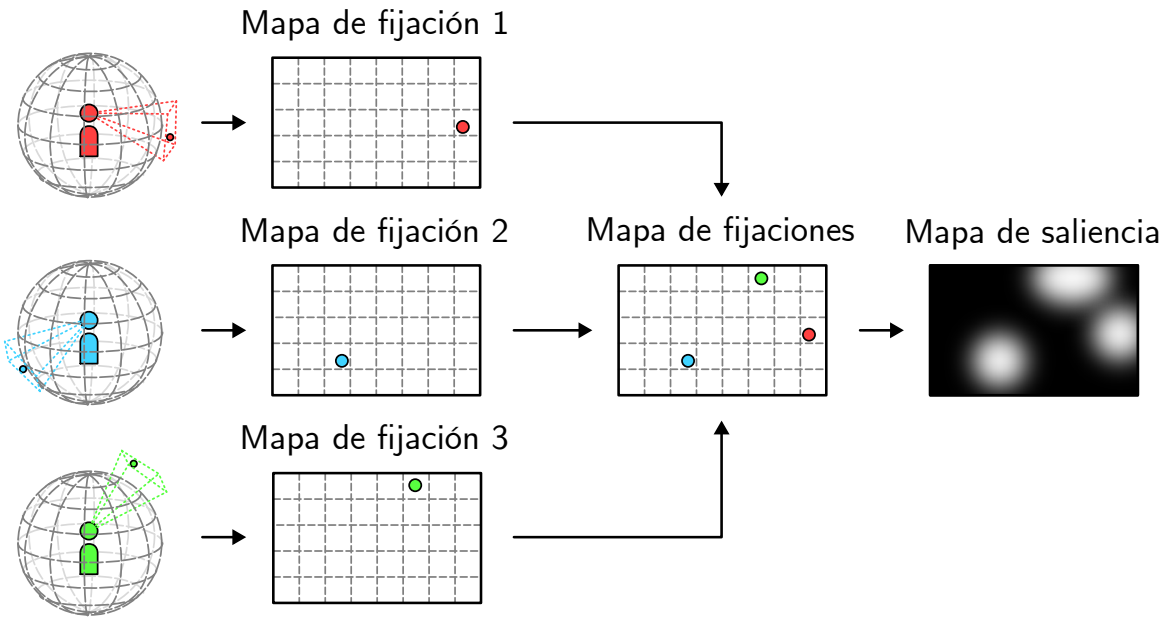


Figura 3.4: Proceso de obtención de los mapas de saliencia *ground truth*. Primero se capturan los puntos de fijación de los usuarios, después se obtienen las coordenadas de los puntos de fijación y se agrupan. Por último se construye el mapa de saliencia agregando una distribución normal centrada en cada punto de fijación, aplicando una distorsión correspondiente a la proyección equirectangular.

En la Figura 3.5 se encuentra un ejemplo de un fragmento de un segmento del dataset procesado, donde se puede observar un conjunto de fotogramas reescalados junto a sus mapas de saliencia generados. Notese cómo la saliencia evoluciona en el tiempo.

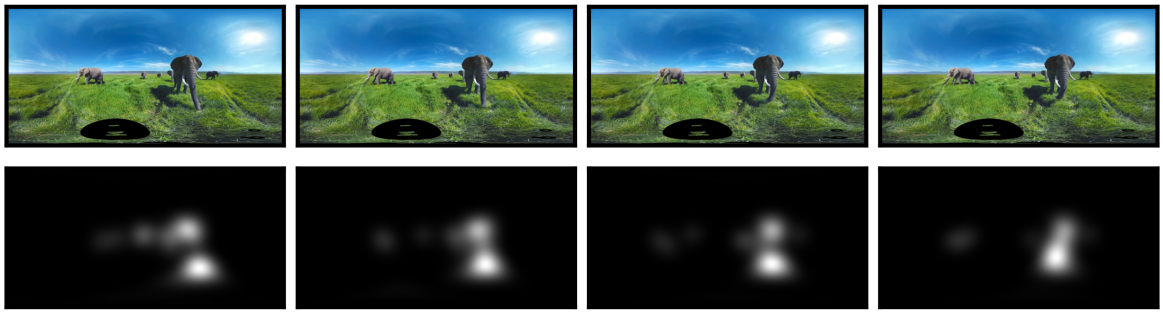


Figura 3.5: Ejemplo de fragmento de vídeo del dataset empleado. La fila superior contiene fotogramas del vídeo original y la fila inferior contiene sus mapas de saliencia correspondientes en escala de grises, donde blanco indica probabilidad igual a uno y negro probabilidad igual a cero.

Debido a que los usuarios comienzan la visualización de los vídeos mirando al frente, se ha decidido descartar los primeros 30 fotogramas, ya que durante ese periodo de tiempo las fijaciones oculares se concentran en el centro del vídeo. Esto genera un mapa

de saliencia sesgado que no representa correctamente la saliencia de los fotogramas, pudiendo afectar negativamente al entrenamiento del modelo.

Además, se ha observado que la acción de los vídeos en muchos casos se produce en el centro del vídeo, lo cual lleva a que los modelos entrenados con estos vídeos tienden a priorizar los píxeles centrales. Por lo que en los casos en los que son suministrados vídeos donde la acción no se encuentra en el centro no saben adaptarse y producen mapas de saliencia incorrectos. Por ello se ha decidido rotar verticalmente de manera aleatoria cada vídeo cada vez que se le pasa a la red. En la Figura 3.6 se encuentra un ejemplo de un mismo fotograma y su mapa de saliencia capturado rotado mediante 4 configuraciones diferentes.

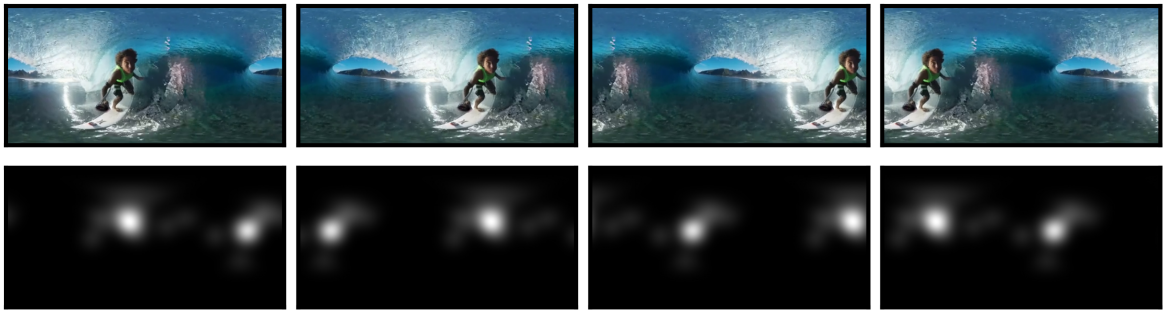


Figura 3.6: Fotograma y mapa de saliencia correspondiente rotado verticalmente en 4 configuraciones diferentes.

Con esto, durante el entrenamiento del modelo, este aprende de un mismo vídeo múltiples veces pero con distinta rotación, lo cual le ayudará a aprender a generar las salidas según los contenidos de los fotogramas y no según las posiciones de los píxeles. Esto además aumenta considerablemente la variabilidad de los vídeos, produciendo un aumento artificial del dataset.

3.2.2. Entrenamiento

Con el conjunto de datos de entrada y los mapas de saliencia *ground truth* se entrena el modelo mediante aprendizaje automático supervisado. Para ello los datos de entrada son separados en 3 categorías: datos de entrenamiento, datos de validación y datos de test. Se han seleccionado 5 vídeos del dataset VR-Eyetracking como datos de test que se utilizarán para evaluar el funcionamiento del modelo final una vez ha sido entrenado. De los 203 vídeos restantes, se han seleccionado 90 % como datos de entrenamiento y 10 % como datos de validación. Durante el entrenamiento el modelo lee los datos de entrada, genera los mapas de saliencia y los compara con los mapas de saliencia *ground truth* mediante una función de coste. Después un optimizador emplea el valor de la

función de coste para entrenar el modelo ajustando los parámetros de la red de forma en la que se minimice dicha función. Por tanto el optimizador elige los parámetros de la red que proporcionan un mapa de saliencia a la salida de la red más cercano al mapa de saliencia *ground truth*. Un entrenamiento completo está formado por varias épocas. En cada época se realiza el proceso de cálculo de función de coste y posterior optimización con cada uno de los datos de entrenamiento. Al final de cada época se emplean los datos de validación para realizar una evaluación del estado actual del modelo. Cabe destacar que el optimizador no se utiliza durante la validación, por lo que el modelo nunca aprende de los datos de validación.

Función de coste

La función de coste empleada se trata de Mean-Square-Error (MSE). Esta función de coste se calcula como la media del error de todos los n píxeles, donde el error en un píxel i se define como la diferencia entre su valor en el mapa de saliencia predicho por el modelo $\hat{y}_i$ y su valor en el mapa de saliencia *ground truth* y_i, elevado al cuadrado (Equación 3.3).

$$\text{MSE} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{3.3}$$

Parámetros internos del modelo final

Para la selección de los parámetros finales del modelo, tras experimentar con múltiples valores se ha decidido mantener los valores propuestos en la implementación original del Transformer [4]. Por ello el modelo final implementado consiste de $N = 6$ capas de encoders con dimensión interna de $d_{\text{modelo}} = 512$. El número de cabezas empleadas para la atención es de $h = 8$ y la dimensión de la capa interna del MLP es de $d_{\text{MLP}} = 2048$. Además, se ha añadido al mecanismo de atención un valor dropout igual a 0,1, lo cual desactiva aleatoriamente algunas neuronas para evitar el sobre-ajuste durante el entrenamiento del modelo. Se ha decidido emplear un tamaño de parche igual a $16 \times 16 \times 2$. Esto significa que los parches estarán formados por 16 pixeles de ancho y alto que codifican la información espacial y 2 capas de profundidad que codifican la información temporal. Debido a que la atención global se calcula sobre los tokens, se busca seleccionar un tamaño de parche que codifique la unidad mínima de información de una región en un rango de tiempo. Por ello se ha considerado apropiado el tamaño de parche de $16 \times 16 \times 2$, que además coincide con el tamaño de parche empleado por ViViT [8]. Con esto un parche representa la evolución del movimiento de un fotograma con el siguiente en una región pequeña del vídeo. A partir de la información de cómo

evolucionan estas pequeñas regiones del vídeo se pueden calcular las relaciones de estas regiones mediante la atención global.

Selección del modelo final

El mejor modelo no siempre corresponde con el que más épocas de entrenamiento realiza, ya que puede ocurrir un caso de *overfitting* donde el modelo aprende demasiado los datos de entrada y pierde la capacidad de abstraer y generar salidas correctas para datos distintos a aquellos que ha utilizado durante el entrenamiento. Por ello se ha buscado el número de épocas que producen el mejor modelo. Para ello se han utilizado los 20 videos del dataset Sports-360 separados para la validación, correspondientes a un total de 88 secuencias. A partir de estos vídeos se han calculado las métricas detalladas en la Sección 4.1 con 20 modelos diferentes obtenidos durante el entrenamiento desde 50 épocas a 1000 épocas en incrementos de 50. Tras realizar estos cálculos, se han comparado las métricas obtenidas para todos los modelos y se ha seleccionado el modelo que mejores resultados obtiene en conjunto. Con esto se ha decidido seleccionar el modelo entrenado con 200 épocas ya que obtiene los mejores resultados en las métricas de KLD y SIM, además de el segundo mejor resultado en CC y el tercer mejor en NSS.

Hiperparámetros de entrenamiento

Tras varias pruebas con otros optimizadores, se ha decidido emplear el optimizado ADAM [24] debido a que ofrecía buenos resultados además de tratarse del optimizador empleado en la implementación original del Transformer [4]. El optimizador ha sido configurado con un *learning rate* de 10^{-4}, lo cual regula lo grandes que da los pasos al realizar el ajuste de los parámetros entrenables.

Para realizar el entrenamiento, no es necesario realizar todos los pasos con cada vídeo individual, sino que se pueden agrupar los vídeos por tandas para acelerar el entrenamiento. En el entrenamiento del modelo final, el número de vídeos por tanda elegido, llamado *batch size*, ha sido de 2.

El entrenamiento se ha llevado a cabo en en un tarjeta gráfica *Nvidia Quadro P6000*, donde se ha entrenado durante 200 épocas a lo largo de un total de 1 día y 56 minutos. El modelo final tiene un total de 20.497.921 parámetros.

Capítulo 4

Evaluación

Una vez entrenado el modelo final se ha evaluado su rendimiento tanto cualitativamente (Sección 4.2) como cuantitativamente (Sección 4.3). La evaluación se ha realizado mediante los vídeos de test (ver Sección 3.2.2) además de nuevos vídeos provenientes del dataset Sports-360 [23], que contiene 104 vídeos de diversas actividades deportivas como baloncesto, bicicletas, baile, etc. Los cuales han sido procesados de la misma forma que el dataset VR-Eyetrackiong como se explica en la Sección 3.2.1.

4.1. Métricas

Se busca evaluar el rendimiento del modelo de manera cuantitativa para medir los resultados obtenidos y compararlo con otros modelos existentes. Para ello se emplean métricas que calculan las diferencias entre los mapas de saliencia generados por el modelo y los mapas de saliencia *ground truth*. Debido a que no existe una sola métrica capaz de evaluar la calidad de las predicciones, se han empleado una serie de métricas que permiten obtener una idea general de su funcionamiento. Las métricas elegidas se tratan de las siguientes: Similarity (SIM), Pearson Correlation Coefficient (CC), Kullback-Leibler Divergence (KLD), Normalized Scanpath Saliency (NSS) y Reciever Operator Curve (ROC). Se han seleccionado estas métricas debido a que tradicionalmente son las más empleada en la literatura para la evaluación de modelos de predicción de la saliencia [25, 26].

Similarity (SIM): Dados dos mapas de saliencia P y Q calcula la similaridad de las dos distribuciones de probabilidad como la suma de los valores mínimos de cada píxel después de haber normalizado los mapas de entrada

$$SIM(P,Q) = \sum_i \text{mín}(P_i, Q_i), \quad \text{donde} \sum_i P_i = \sum_i Q_i = 1. \tag{4.1}$$

Esta métrica es muy sensible a valores ausentes y penaliza las distribuciones que no representan toda la densidad del *ground truth*, por lo que los falsos positivos tienden a ser penalizados menos que los falsos negativos. Debido a que los mapas de entrada son normalizados, el rango de valores que toma esta métrica es entre 1 y 0, donde 1 indica que ambas distribuciones de probabilidad son idénticas. Por ello se busca maximizar esta métrica.

Pearson Correlation Coefficient (CC): Dados dos mapas de saliencia P y Q interpretados como variables aleatorias de distribuciones de probabilidad, el coeficiente de correlación de Pearson calcula la relación lineal entre ellas

$$CC(P, Q) = \frac{\sigma(P, Q)}{\sigma(P) \times \sigma(Q)} \tag{4.2}$$

donde $\sigma(P, Q)$ es la covarianza de P y Q. Esta métrica trata de manera simétrica los falsos positivos y falsos negativos, lo cual implica que no es capaz de distinguir cuando ocurre cada error. El rango de valores de esta métrica se encuentra entre -1 y 1, donde 1 indica una correlación perfecta, y -1 una correlación completamente opuesta a la perfecta, por lo que se busca maximizar esta métrica.

Kullback-Leibler Divergence (KLD): Existen múltiples implementaciones de esta métrica, en este caso se ha empleado la variante KLD-Judd. Dado el mapa de saliencia predicho P y el mapa de saliencia *ground truth* Q, calcula la pérdida de información cuando P se utiliza para aproximar Q como

$$KLD(P, Q) = \sum_i Q_i \log\left(\epsilon + \frac{Q_i}{\epsilon + P_i}\right), \tag{4.3}$$

donde ϵ es una constante de regularización. Esta métrica no es simétrica y penaliza más los falsos positivos. Devuelve un valor de 0 cuando las distribuciones son idénticas y aumenta a medida a la que aumentan las diferencias entre las distribuciones, sin ningún límite superior. Por lo tanto se busca minimizar esta métrica.

Normalized Scanpath Saliency (NSS): Dados un mapas de saliencia P y el mapa binario de fijaciones B (valor del píxel igual a 1 si existe una fijación en esa posición y 0 en caso contrario), calcula la saliencia media normalizada en los puntos de fijación:

$$\begin{aligned} NSS(P, B) &= \frac{1}{N} \sum_i \overline{P}_i \times B_i \\ \text{donde } N &= \sum_i B_i \text{ y } \overline{P} = \frac{P - \mu(P)}{\sigma(P)}. \end{aligned} \tag{4.4}$$

Esta métrica es muy sensible a falsos positivos. Un valor positivo indica que los mapas se asemejan y un valor negativo indica que no, por lo que se busca maximizar esta métrica.

Reciever Operator Curve (ROC): Esta métrica calcula la habilidad que tiene un mapa de saliencia de representar las fijaciones reales efectuadas por un observador. Cada valor de la curva se computa seleccionando el $t\,\%$ de los píxeles más salientes del mapa de saliencia y se observa cuantas fijaciones del conjunto se encuentran dentro de las regiones seleccionadas. Todos aquellos puntos de fijación que se encuentren dentro de las regiones se consideran *true positive*, mientras que aquellos que se encuentran fuera se consideran como *false negative*. El valor de la curva para un porcentaje t se calcula como el *recall* de la selección de las fijaciones según el porcentaje t.

$$ROC(t) = \frac{\text{true\_positive(t)}}{\text{true\_positive(t)} + \text{false\_negative(t)}}. \tag{4.5}$$

Mapas de saliencia que representen un comportamiento similar a las fijaciones con el que se comparan mostrarán unos valores ROC altos para valores bajos de t, convergiendo rápidamente a $ROC = 1$. Por tanto, cuanto más se aleje el mapa de saliencia de la tendencia seguida por las fijaciones, más lenta será la convergencia de la curva, acercándose cada vez más a un comportamiento aleatorio (representado por una recta de pendiente 1).

4.2. Resultados obtenidos

El modelo final se ha evaluado cualitativamente con los vídeos seleccionados para el test. En las Figuras 4.1, 4.2, 4.3 se pueden observar ejemplos de las predicciones ofrecidas por el modelo propuesto para tres vídeos distintos. Notese que no se muestran fotogramas consecutivos, sino que se ha muestreado algunos fotogramas de la secuencia para poder apreciar las evolución temporal del vídeo, ya que fotogramas consecutivos tienden a mostrar mínimas diferencias. No obstante, los resultados para la totalidad de la secuencia pueden ser encontrados en el Anexo B. También se puede encontrar una lista de reproducción de los vídeos generados en el enlace `https://youtube.com/playlist?list=PLqbqFkEk0bj7Og7ovzk01YwhXv98xd8-R`.

En la Figura 4.1 se encuentran los resultados obtenidos con el primer vídeo de test. En este vídeo un buceador se aproxima a un barco sumergido y entra en su interior. Se puede ver en el *ground truth* como la mayoría de observadores reales establecen dos regiones salientes principales, el buceador y el barco. En los mapas de saliencia

generados se puede observar como el modelo ha sido capaz de identificar las regiones salientes más importantes incluyendo el buceador y el barco sobre los cuales realiza un seguimiento a largo de vídeo.

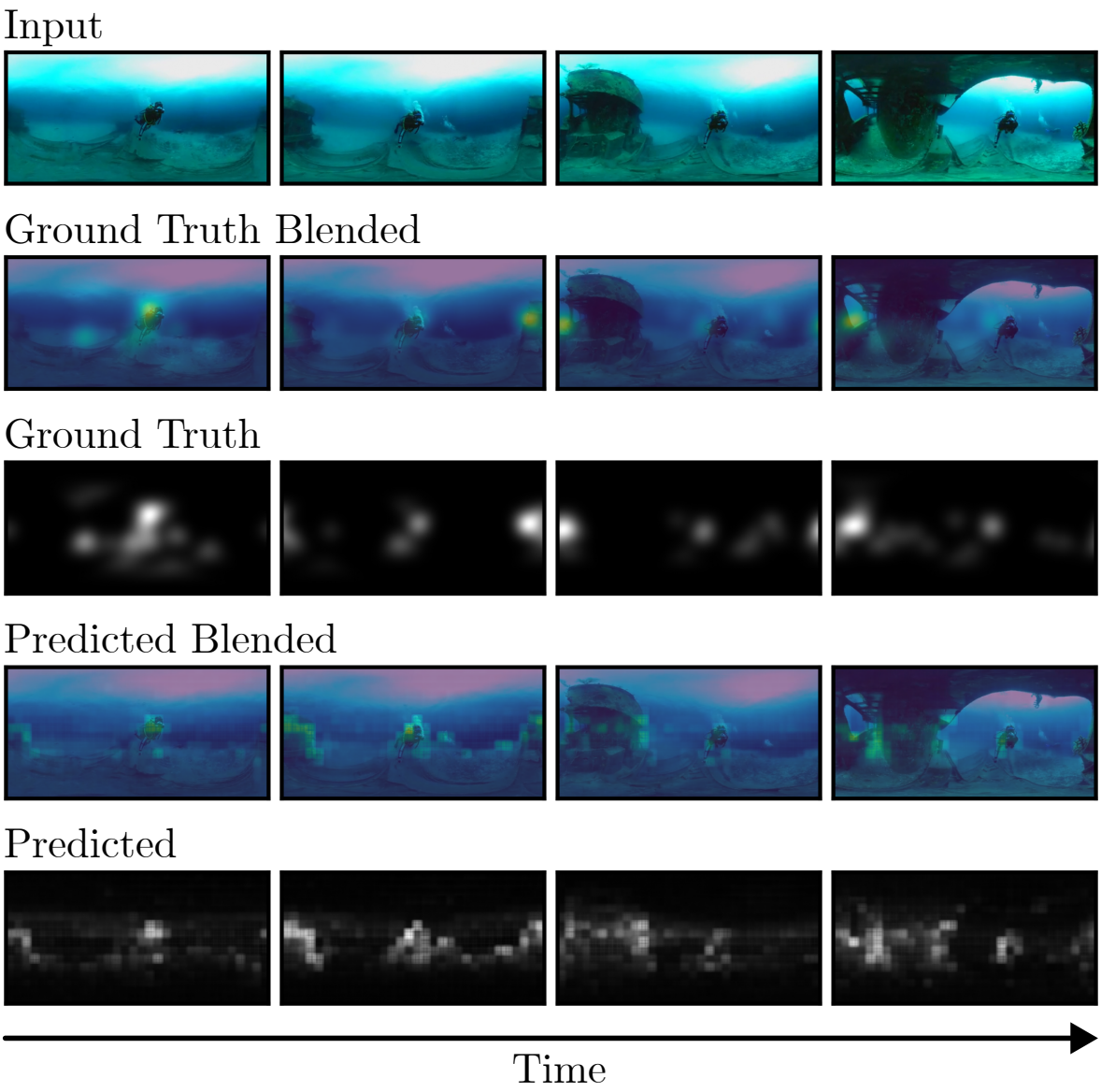


Figura 4.1: Resultados obtenidos con el vídeo 172 del dataset VR-Eyetracking. En la primera fila se muestran los fotogramas del vídeo original, en las filas 2 y 3 se muestran los mapas de saliencia *ground truth* y en las filas 4 y 5 se muestran los mapas de saliencia generados por el modelo. En las filas 3 y 5 se representan las regiones salientes en escala de grises, indicando con valores más cercanos a blanco cuanta mayor es la probabilidad, y en las filas 2 y 4 los colores más cálidos representan mayor saliencia.

En la figura 4.2 se encuentran los resultados obtenidos con el segundo vídeo de test, donde tres rinocerontes vagan por una pradera. En los mapas de saliencia *ground truth* se puede observar como se concentra la atención sobre el rinoceronte central. En los mapas de saliencia generados se puede observar que el modelo ha identificado dos

regiones salientes principales en los rinocerontes central y derecho. A medida que avanza el vídeo el rinoceronte derecho se aleja de la cámara y el modelo deja de detectarlo como región saliente e incrementa la atención en el rinoceronte izquierdo, lo cual se puede apreciar que también ocurre en el mapa de saliencia *ground truth*.

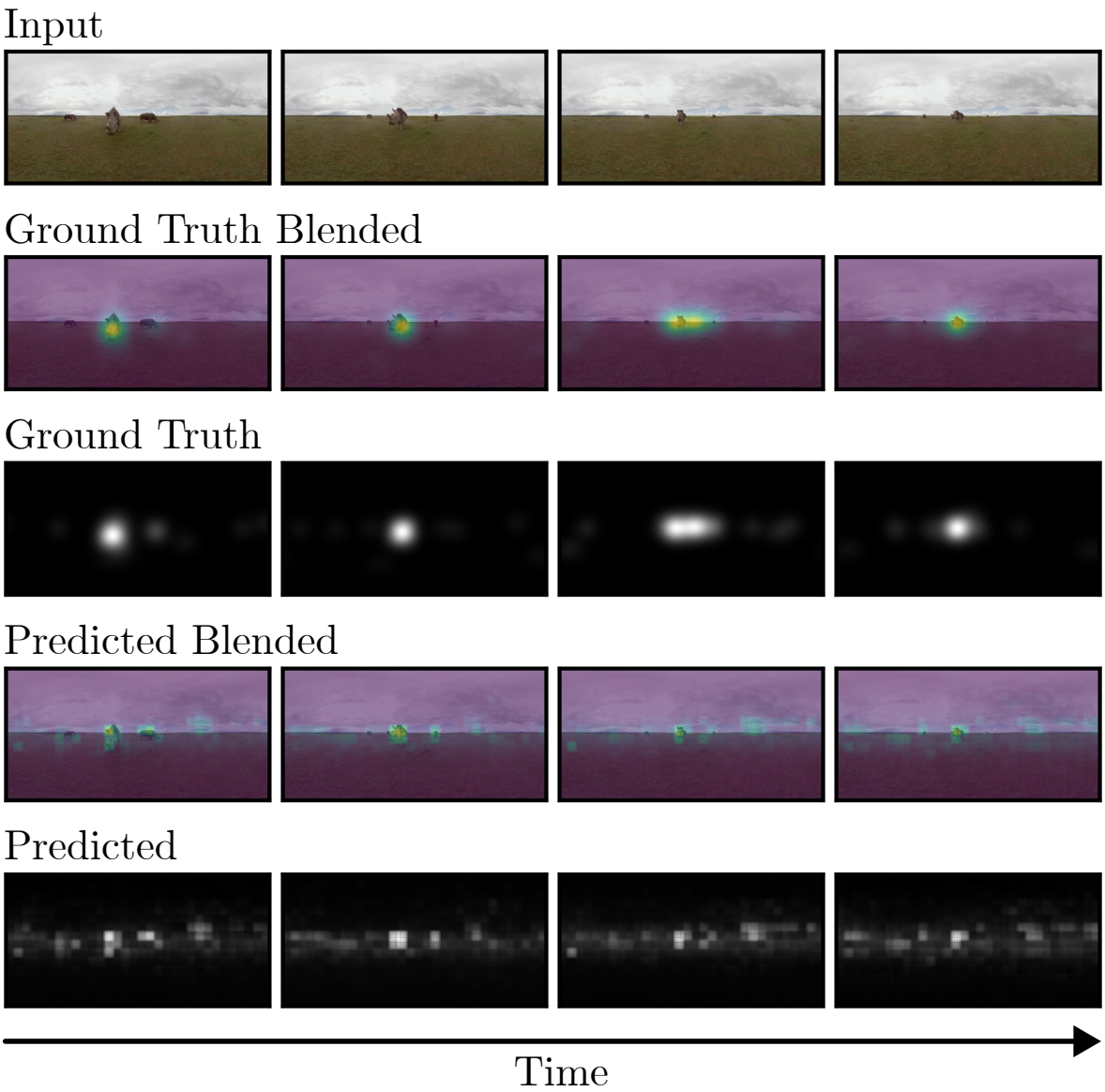


Figura 4.2: Resultados obtenidos con el vídeo 213 del dataset VR-Eyetracking. En la primera fila se muestran los fotogramas del vídeo original, en las filas 2 y 3 se muestran los mapas de saliencia *ground truth* y en las filas 4 y 5 se muestran los mapas de saliencia generados por el modelo. En las filas 3 y 5 se representan las regiones salientes en escala de grises, indicando con valores más cercanos a blanco cuanta mayor es la probabilidad, y en las filas 2 y 4 los colores más cálidos representan mayor saliencia.

En la figura 4.3 se encuentran los resultados obtenidos con un vídeo del dataset Sports-360. En este vídeo una persona se tumba en el suelo mientras otra salta por encima con una bicicleta. En el mapa de saliencia *ground truth* se puede ver

claramente que la región más saliente se concentra en la bicicleta en movimiento. El mapa de saliencia generado por el modelo comienza disperso a lo largo de la imagen, pero a medida a la que se acerca la bicicleta, el modelo produce una región saliente siguiendo el movimiento de la bicicleta, asemejándose al comportamiento que tienen los observadores reales en el *ground truth*.

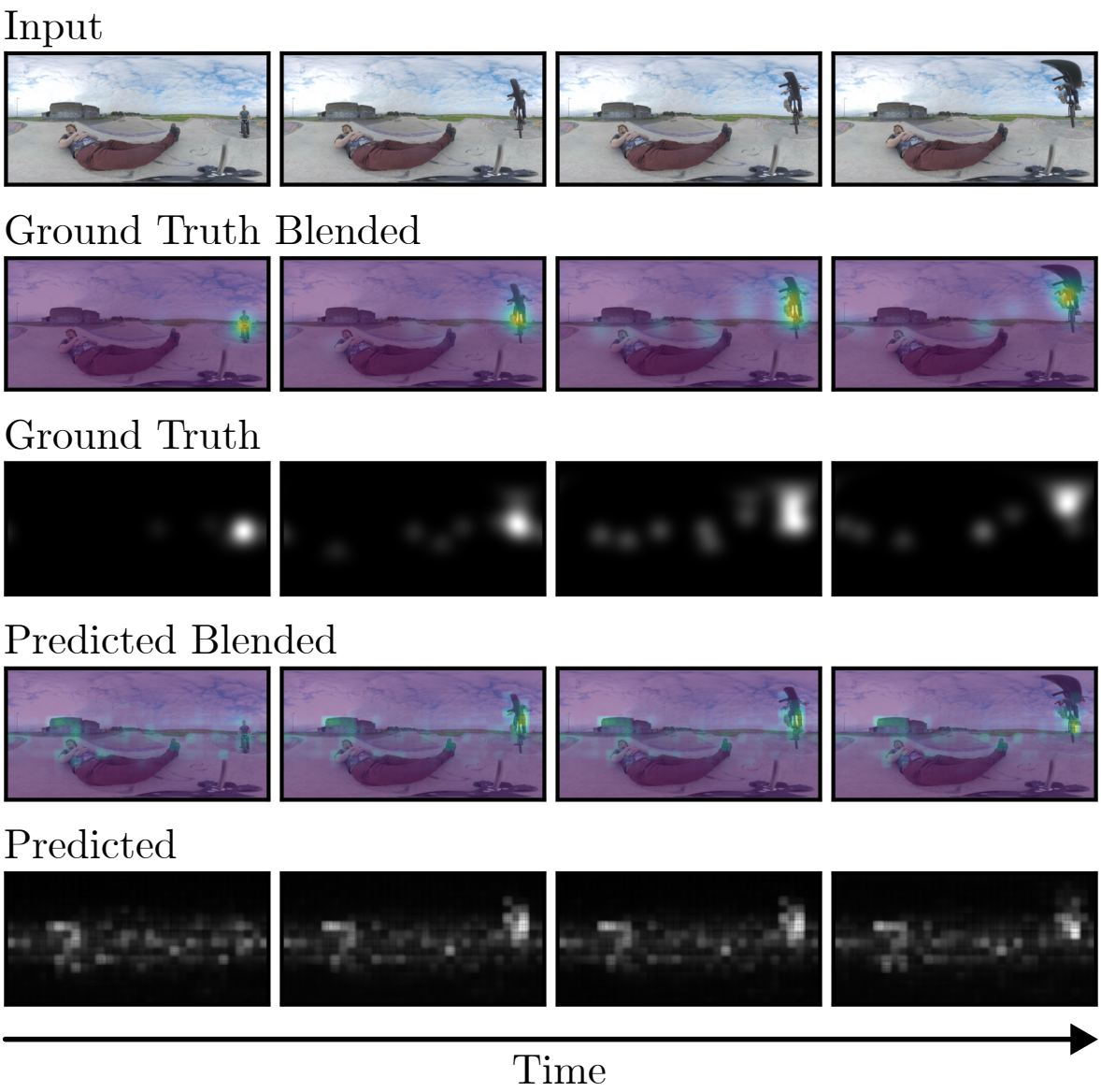


Figura 4.3: Resultados obtenidos con el vídeo 242 del dataset Sports-360. En la primera fila se muestran los fotogramas del vídeo original, en las filas 2 y 3 se muestran los mapas de saliencia *ground truth* y en las filas 4 y 5 se muestran los mapas de saliencia generados por el modelo. En las filas 3 y 5 se representan las regiones salientes en escala de grises, indicando con valores más cercanos a blanco cuanta mayor es la probabilidad, y en las filas 2 y 4 los colores más cálidos representan mayor saliencia.

4.3. Evaluación respecto al estado del arte

Para la evaluación cuantitativa del modelo se han empleado las métricas descritas en la Sección 4.1, utilizando los vídeos de test del dataset Sports-360 como se indica en la Sección 3.2.1. Para realizar la evaluación se ha comparado su funcionamiento con otros modelos existentes. Para ello se ha empleado un modelo popular dentro del estado del arte, CP-360 [27]. Este modelo emplea redes neuronales convolucionales (ver Anexo A.1.1) y redes neuronales recurrentes LSTM (ver Anexo A.1.2) para obtener la saliencia a partir de los vídeos de entrada. En la figura 4.4 se encuentra una comparación de los mapas de saliencia generados por el modelo implementado y por el modelo CP-360.

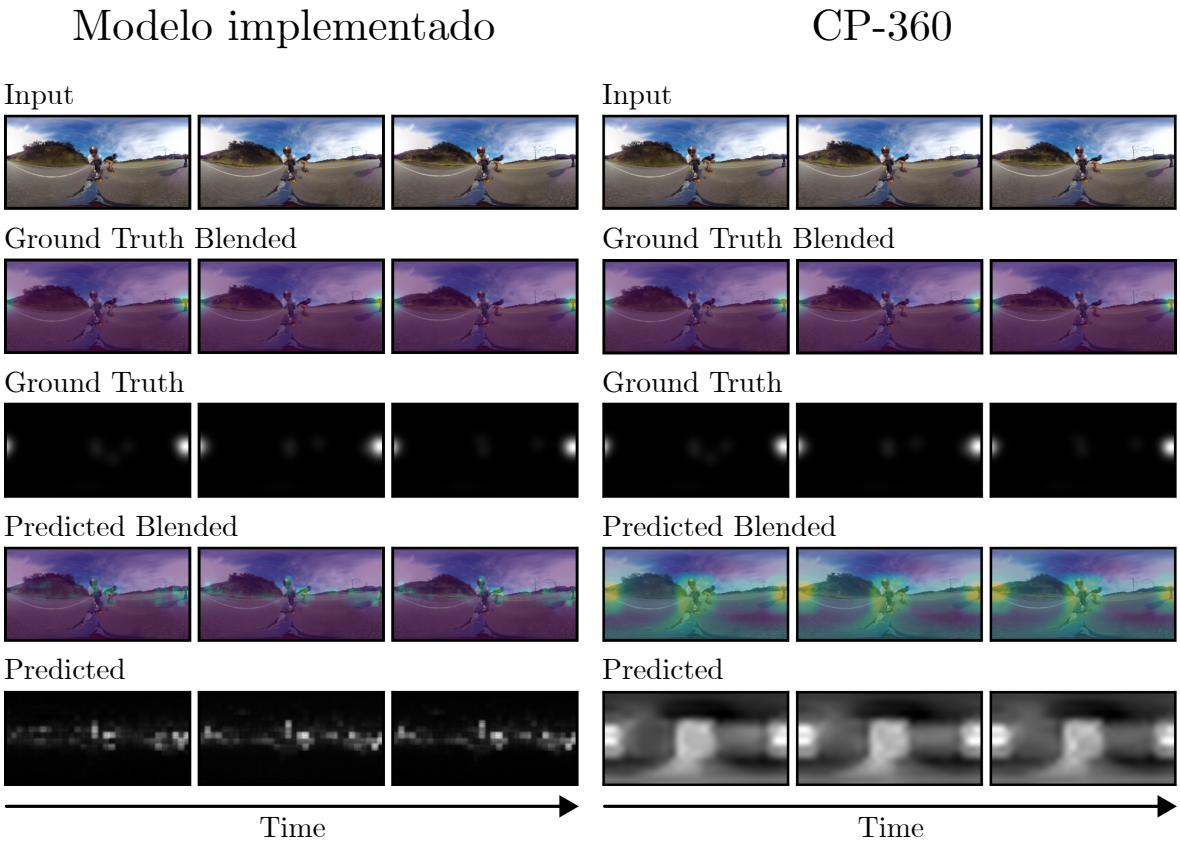


Figura 4.4: Comparación de mapas de saliencia generados. En la izquierda se encuentran los generados por el modelo implementado y en la derecha los generados por el modelo CP-360. En la primera fila se muestran los fotogramas del vídeo original, en las filas 2 y 3 se muestran los mapas de saliencia *ground truth* y en las filas 4 y 5 se muestran los mapas de saliencia generados por el modelo. En las filas 3 y 5 se representan las regiones salientes en escala de grises, indicando con valores más cercanos a blanco cuanta mayor es la probabilidad, y en las filas 2 y 4 los colores más cálidos representan mayor saliencia.

Se puede observar que el modelo CP-360 ha identificado correctamente como región más saliente el patinador de detrás y como segunda región más saliente los dos

patinadores de enfrente, aunque los resultados también incluyen un gran numero de falsos positivos.

En la figura 4.5 se muestran las curvas ROC obtenidas con todos los vídeos de test tanto para el modelo implementado como para el modelo CP-360. Se puede observar que la curva correspondiente al modelo implementado se encuentra relativamente cercana a la curva del *ground truth*, encontrándose muy alejada de lo que correspondería con un modelo aleatorio. Se puede observar que de media, seleccionando tan solo el 20 % de las regiones más salientes ya se incluyen el 90 % de las fijaciones oculares, convergiendo a $ROC = 1$ en torno al 50 % de las regiones más salientes. Lo que parece indicar que el modelo propuesto es capaz de representar el comportamiento visual humano. La curva del modelo CP-360 se encuentra mucho más alejada de la curva *ground truth* que el modelo implementado. Esto indica que el modelo implementado produce un comportamiento más cercano al de los usuarios reales que el modelo CP-360.

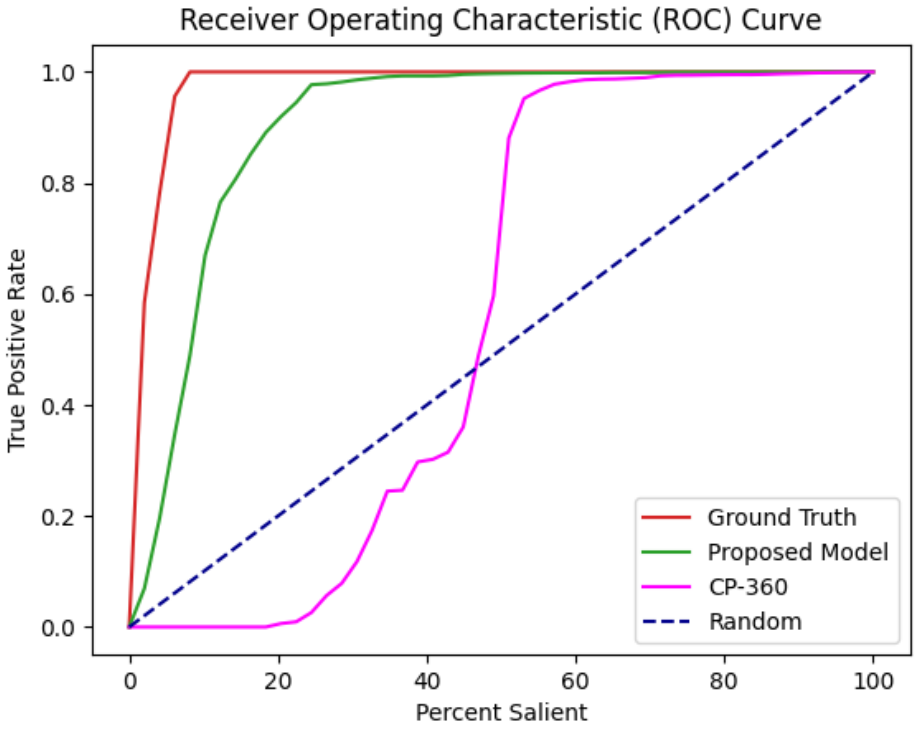


Figura 4.5: Curvas ROC del *ground truth* (rojo), modelo implementado (verde), CP-360 (rosa) y aleatorio (azul) para los vídeos de test del dataset Sports-360.

En la Tabla 4.3 se encuentra la comparación de las métricas obtenidas con el modelo implementado en este trabajo y con el modelo CP-360, empleando los mismos datos de test. Se puede observar que el modelo implementado obtiene mejores resultados en todas las métricas calculadas, lo cual podría indicar la superioridad del enfoque

propuesto, en las que las arquitecturas LSTM son reemplazadas por los Transformers. La razón de su superioridad se podría atribuir al mecanismo de atención global, que es capaz de encontrar relaciones entre dos fotogramas independientemente de lo lejos que se encuentren en la secuencia, mientras que los modelos LSTM operan con información del estado anterior, lo cual acaba causando que a lo largo del tiempo se pierda información de estados previos gradualmente.

	SIM ↑	CC ↑	KLD ↓	NSS ↑
Modelo implementado	**0.3375**	**0.3048**	**6.3080**	**1.387**
CP-360 [27]	0.2761	0.2338	8.3600	0.9515

Capítulo 5

Conclusiones

A lo largo del trabajo se ha realizado el proceso completo de investigación, diseño, desarrollo y evaluación de un modelo de aprendizaje profundo para predicción de saliencia en vídeos 360º.

Primero se han estudiado los modelos actuales del estado del arte, identificando sus limitaciones a la hora de modelar las relaciones temporales y proponiendo soluciones novedosas con la arquitectura del Transformer.

Después se ha aprendido a utilizar frameworks modernos de desarrollo de sistemas de aprendizaje automático, habiendo realizado una implementación propia del modelo, su ciclo de entrenamiento y su evaluación en el lenguaje de programación Python mediante el framework PyTorch, diseñándolo para su ejecución en GPU. Se ha desarrollado la arquitectura del modelo, observando los resultados obtenidos y mejorando distintos aspectos a lo largo de la implementación. Se ha realizado un proceso de tratamiento de datos donde se han adaptado varios datasets, generando sus mapas de saliencia asociados. Se ha implementado un bucle de entrenamiento del modelo configurado para ser ejecutado en una GPU.

Además, se han utilizado varias métricas tradicionalmente empleadas en el ámbito de la saliencia para tomar decisiones informadas a la hora de seleccionar los mejores valores para los parámetros del modelo.

El resultado final obtenido es un modelo que produce un comportamiento similar a los humanos, como se esperaba alcanzar en los comienzos del trabajo. Habiendo comparado el modelo implementado contra un trabajo del estado del arte y demostrando que en conjunto obtiene mejores resultados en todas las métricas empleadas. Los resultados obtenidos parecen indicar que los mecanismos de atención global presentes en el modelo del Transformer realizan mejor labor a la hora de modelar las dependencias temporales de los vídeos que las redes neuronales recurrentes.

5.1. Limitaciones y trabajo futuro

Aunque se ha conseguido una buena aproximación al comportamiento real de los usuarios, siendo capaces de identificar las zonas salientes en la mayoría de los vídeos, todavía queda margen de mejora, ya que se trata de un problema con un nivel de complejidad muy elevado.

Se ha observado que, a pesar de que el cálculo de la atención se realiza de manera correcta en la mayoría de casos, una vez calculada la atención el modelo tiene algunas dificultades empleando esa información para generar los mapas de saliencia de salida. Por lo tanto, versiones futuras de este modelo deberían centrarse en cómo mejorar la generación de datos de salida. En el modelo implementado, se generan los datos de salida mediante una única convolución traspuesta. No obstante, la decodificación de la información contenida en los tokens para la obtención de los mapas de saliencia es una tarea compleja y es posible que una sola convolución traspuesta no sea suficiente. En el modelo original del Transformer se emplea una arquitectura específica para generar los datos de salida (decoder). Por tanto, se propone como trabajo futuro el desarrollo de un modelo de predicción de saliencia empleando una arquitectura encoder-decoder. En la Figura 5.1 se encuentra un diagrama de un posible diseño para este modelo.

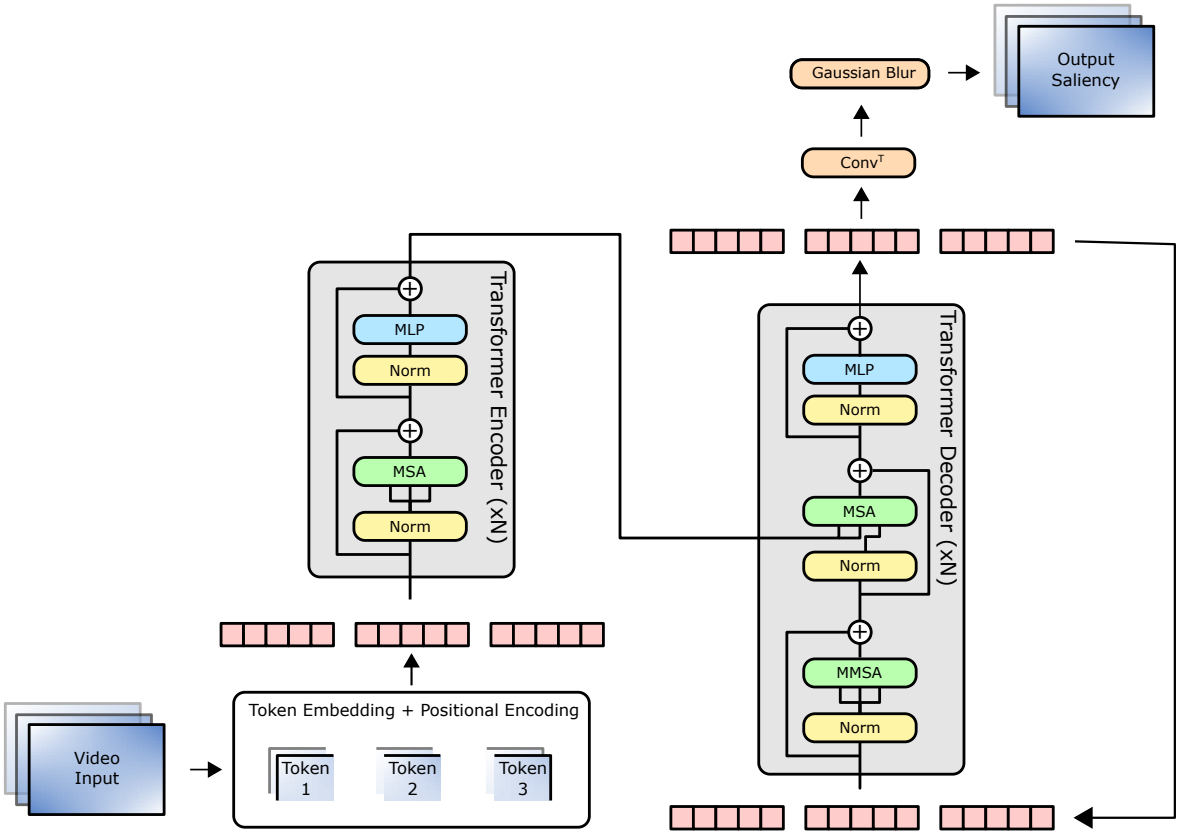


Figura 5.1: Arquitectura mejorada propuesta formado por dos regiones principales, el encoder y el decoder. El encoder se encargaría de calcular la atención global para los tokens de entrada y el decoder emplearía esa información para generar los tokens de saliencia de salida, que después serían reconstruidos a fotogramas mediante una convolución traspuesta.

Otra mejora que se podría aplicar al modelo es la incorporación de información adicional sobre los datos de entrada, como podrían ser mapas de flujo óptico o mapas de profundidad. Esta información adicional podría ayudar al modelo tener un mayor conocimiento de los contenidos de las escenas, permitiéndole tomar decisiones más informadas.

Por último, se ha observado en algunas ocasiones que la información visual no es suficiente para modelar completamente la atención de los usuarios. Esto es debido a que los vídeos se tratan de contenido audiovisual, que por tanto incluyen sonido. Es por ello que estudiar saliencia a partir de la información visual exclusivamente está obviando completamente el contexto adicional ofrecido por el sonido del vídeo. Un ejemplo de esto es la tendencia a mirar a la cara de la persona que esté hablando, el cual se ha observado que el modelo implementado tiene dificultades de identificar. Por tanto se estima que futuros modelos que incorporen información sobre el sonido del vídeo obtendrán resultados mejores en estos casos.

Capítulo 6

Bibliografía

[1] Junting Pan, Cristian Canton Ferrer, Kevin McGuinness, Noel E. O'Connor, Jordi Torres, Elisa Sayrol, and Xavier Giro-i Nieto. Salgan: Visual saliency prediction with generative adversarial networks, 2017.

[2] Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein. Saliency in VR: How Do People Explore Virtual Environments? *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1633–1642, April 2018. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[3] Daniel Martin, Ana Serrano, and Belen Masia. Panoramic convolutions for 360° single-image saliency prediction. In *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, 2020.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017. arXiv: 1706.03762.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[6] Ananthan Nambiar, Maeve Heflin, Simon Liu, Sergei Maslov, Mark Hopkins, and Anna Ritz. Transforming the language of life: Transformer neural networks for protein prediction tasks. In *Proceedings of the 11th ACM International Conference*

on Bioinformatics, Computational Biology and Health Informatics, BCB '20, New York, NY, USA, 2020. Association for Computing Machinery.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv:2010.11929 [cs]*, June 2021. arXiv: 2010.11929.

[8] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A Video Vision Transformer. *arXiv:2103.15691 [cs]*, November 2021. arXiv: 2103.15691.

[9] Erkut Erdem and Aykut Erdem. Visual saliency estimation by nonlinearly integrating features using region covariances. *Journal of vision*, 13, 03 2013.

[10] R. Achantay, S. Hemamiz, F. Estraday, and S. Süsstrunky. Frequency-tuned salient region detection. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pages 1597–1604, 06 2009.

[11] Srinivas S. S. Kruthiventi, Kumar Ayush, and R. Venkatesh Babu. Deepfix: A fully convolutional neural network for predicting human eye fixations. *IEEE Transactions on Image Processing*, 26(9):4446–4456, 2017.

[12] Cagdas Bak, Aysun Kocak, Erkut Erdem, and Aykut Erdem. Spatio-temporal saliency networks for dynamic saliency prediction, 2016.

[13] Lai Jiang, Mai Xu, Tie Liu, Minglang Qiao, and Zulin Wang. Deepvs: A deep learning based video saliency prediction approach. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 625–642, Cham, 2018. Springer International Publishing.

[14] Marc Assens, Kevin McGuinness, Xavier Giro-i Nieto, and Noel E. O'Connor. Saltinet: Scan-path prediction on 360 degree images using saliency volumes, 2017.

[15] Fang-Yi Chao, Lu Zhang, Wassim Hamidouche, and Olivier Déforges. Salgan360: Visual saliency prediction on 360 degree images with generative adversarial networks. *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 01–04, 2018.

[16] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. SphereNet: Learning Spherical Representations for Detection and Classification in Omnidirectional Images. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pages 525–541, Cham, 2018. Springer International Publishing.

[17] Yasser Dahou, Marouane Tliba, Kevin McGuinness, and Noel O'Connor. ATSal: An Attention Based Architecture for Saliency Prediction in 360 Videos. Technical Report arXiv:2011.10600, arXiv, November 2020. arXiv:2011.10600 [cs] type: article.

[18] Yanyu Xu, Ziheng Zhang, and Shenghua Gao. Spherical dnns and their applications in 360° images and videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[21] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding?, 2021.

[22] Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao. Gaze prediction in dynamic 360° immersive videos. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5333–5342, 2018.

[23] Hou-Ning Hu, Yen-Chen Lin, Ming-Yu Liu, Hsien-Tzu Cheng, Yung-Ju Chang, and Min Sun. Deep 360 pilot: Learning a deep agent for piloting through 360° sports video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[25] Jesús Gutiérrez, Erwan David, Yashas Rai, and Patrick Le Callet. Toolbox and dataset for the development of saliency and scanpath models for

omnidirectional/360° still images. *Signal Processing: Image Communication*, 69:35–42, 2018. Salient360: Visual attention modeling for 360° Images.

[26] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. What Do Different Evaluation Metrics Tell Us About Saliency Models? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(3):740–757, March 2019. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[27] Hsien-Tzu Cheng, Chun-Hung Chao, Jin-Dong Dong, Hao-Kai Wen, Tyng-Luh Liu, and Min Sun. Cube padding for weakly-supervised saliency prediction in 360° videos, 2018.

Anexos

Anexos A

Marco Teórico

A.1. Redes Neuronales

Una red neuronal es un circuito de neuronas que dados unos datos de entrada es capaz de generar nuevos datos de salida combinando múltiples capas de neuronas. Una neurona está compuesta por un conjunto de entradas x_i con sus respectivos pesos w_i y un sesgo o *bias* b junto a su función de activación f. Una neurona produce una salida y que se conecta a otras neuronas como parte del conjunto de entradas de esas neuronas. En la Figura A.1 se encuentra un diagrama de los componentes de una neurona.

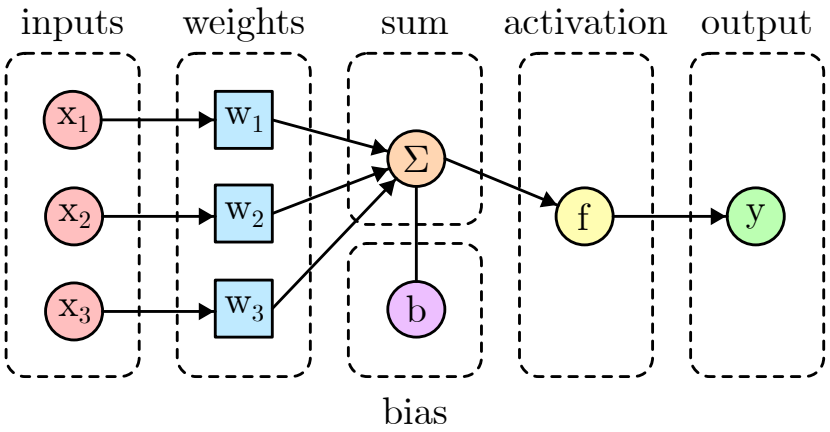


Figura A.1: Diagrama de neurona artificial.

La salida de una neurona se obtiene como la suma de las entradas ponderadas por sus pesos respectivos más el *bias*, aplicando la función de activación (Ecuación A.1).

$$f\left(b + \sum_{i}^{N} w_i x_i\right). \tag{A.1}$$

Una red neuronal está formada por una capa de entrada, una capa de salida y 0 o más capas ocultas intermedias, como se muestra en la Figura A.2

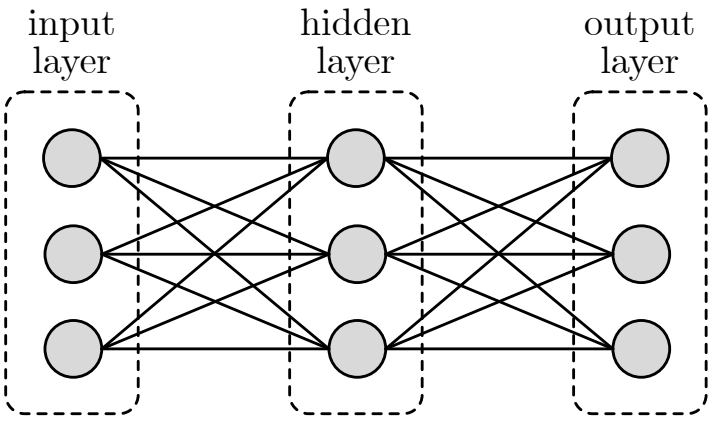


Figura A.2: Ejemplo de red neuronal *feed-forward* densamente conectada con una capa oculta.

Durante el entrenamiento de la red se ajustan automáticamente los parámetros de los pesos w_i y el *bias* b, con lo cual las neuronas aprenden a generar las salidas relevantes según sus datos de entrada.

A.1.1. Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales se tratan de redes neuronales que realizan la operación de convolución para extraer información local de un conjunto de datos de entrada. La operación de convolución consiste en aplicar un filtro sobre los datos de entrada mediante *kernel* entrenable. En la Figura A.3 se encuentra un diagrama de una operación de convolución sobre los datos de entrada.

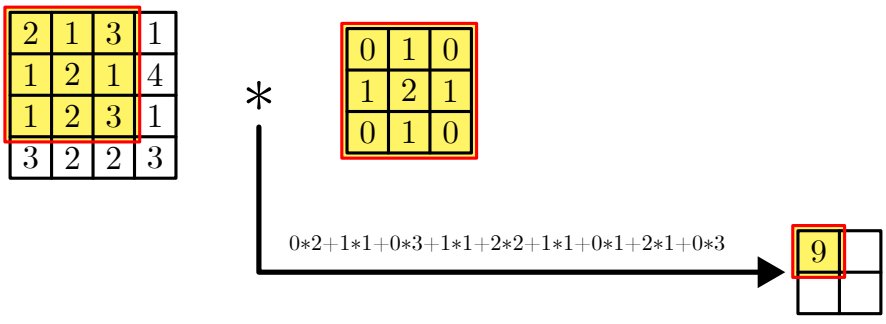


Figura A.3: Operación de convolución con un kernel de tamaño 3×3 sobre un conjunto de datos de tamaño 4×4, lo cual produce una salida de tamaño 2×2.

Estas redes son muy populares en el campo de visión por computador ya que son especialmente buenas detectando patrones locales. Esto permite realizar arquitecturas formadas por varias capas convolucionales donde las primeras capas extraen información sobre las características de la imagen de entrada como pueden ser

los contornos, y capas posteriores detectan patrones de mayor alto nivel como pueden ser ojos, caras, formas geométricas, etc.

A.1.2. Redes Neuronales Recurrentes (RNN)

Una red neuronal recurrente se trata de una red neuronal en la que una o varias de las neuronas de una capa se encuentran conectadas a neuronas en capas anteriores formando un ciclo. Estas redes neuronales por tanto son capaces de almacenar información y recordar estados previos de la red.

Un ejemplo de red neuronal recurrente es Long-Short-Term-Memory (LSTM). Estas redes neuronales recurrentes están formadas por 5 componentes principales: *cell state*, *hidden state*, *forget gate*, *input gate* y *output gate*. El *cell state* define el estado de la red y se trata del componente recurrente que recuerda el estado anterior de la red. El *hidden state* corresponde con la salida. El resto de componentes se encargan de regular el flujo de información actualizando el estado del *cell state*. El *forget gate* se encarga de decidir cual información del estado descartar y cual mantener, el *input gate* se encarga de decidir cuales valores del estado actualizar con los datos actuales, y el *output gate* se utiliza para filtrar la información del *cell state* que se emplea para actualizar el *hidden state* y generar la salida para el paso actual.

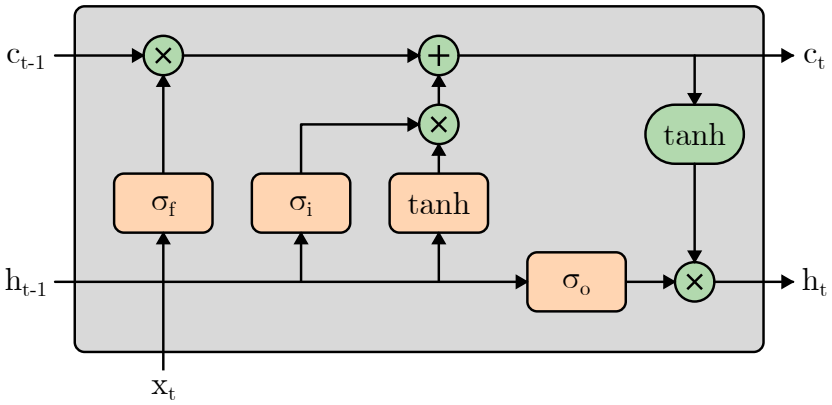


Figura A.4: Diagrama del modelo LSTM. Las capas σ_f, σ_i y σ_o corresponden con el *forget gate*, *input gate* y *output gate* respectivamente. La línea superior c_t corresponde con el *cell state*, que junto a la salida h_t se pasa como entradas a la red en la siguiente iteración.

Estas redes se emplean comúnmente para codificar información temporal como puede ser en el procesamiento de lenguaje natural o procesamiento de vídeos.

A.2. Transformer

El modelo del Transformer trabaja con representaciones vectoriales de las características de los datos de entrada, donde cada uno de estos vectores de características se denomina *token*. En la figura A.5 se puede observar un diagrama de la arquitectura del modelo donde se puede ver que sigue una estructura de encoder-decoder. El encoder se encarga de calcular la atención global de los datos de entrada, encontrado las relaciones existentes entre todos los tokens, y el decoder emplea esta información para generar los datos de salida. El modelo contiene múltiples encoders en serie uno después del otro, además de múltiples decoders en serie de la misma manera.

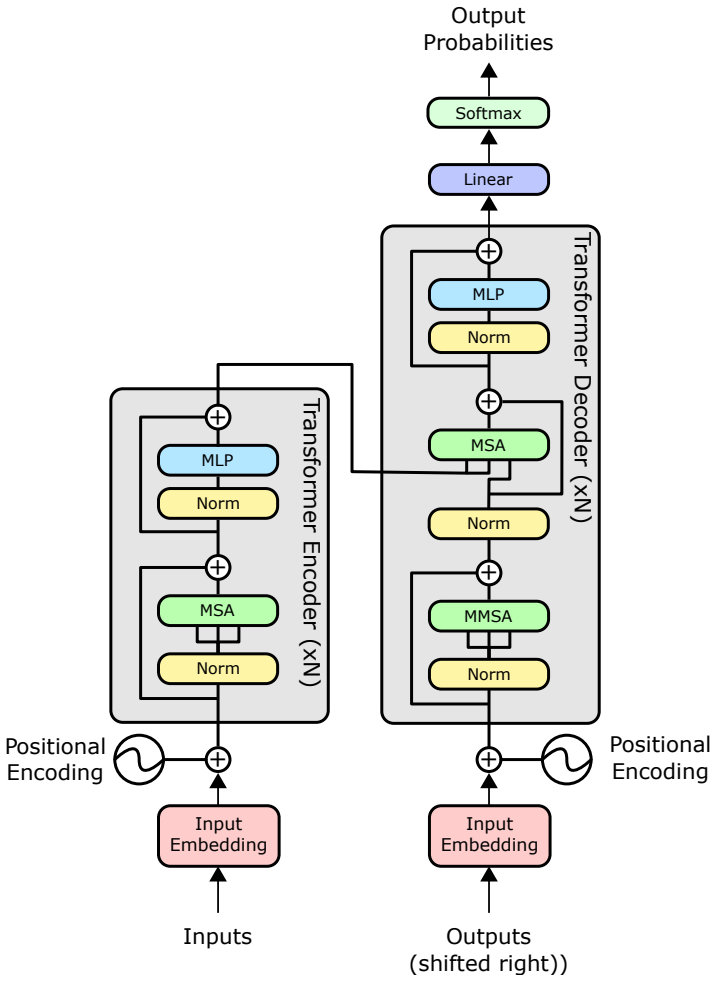


Figura A.5: Arquitectura del modelo de Transformer.

La principal diferencia de los Transformers sobre los modelos anteriores es que el cálculo de atención se realiza de manera global y de una sola vez, en vez de manera local y secuencialmente como ocurre en aquellos modelos basados en redes neuronales recurrentes y convolucionales.

A.2.1. Embedding y Encoding

El proceso de conversión de los datos de entrada a una secuencia de tokens se denomina el proceso de *embedding*.

Debido a que la atención se calcula de manera global, se pierde la información local de los tokens, y tokens lejanos entre sí se tratan de la misma manera que los cercanos, cuando en realidad el orden y la cercanía de los tokens puede proporcionar información sobre cómo se deberían priorizar entre sí. Para solucionar este problema se introduce el concepto de *positional encoding*, con el que se modifican los tokens para incluir información sobre la posición y orden de los tokens para aumentar el grado de similaridad de aquellos tokens que se encuentren en posiciones cercanas.

A.2.2. Encoder

El encoder está formado por dos componentes principales, Multi-Head-Self-Attention (MSA) que corresponde con el mecanismo de atención detallado en la Sección A.2.3, y una capa Multi-Layer-Perceptron (MLP) que corresponde con una capa de neuronas densamente conectada (feed-foward fully connected) explicada en el anexo A.1.

Al final de cada componente MSA y MLP se añade una conexión residual en la que se suma el valor del token al final del paso anterior. Esto se hace para propagar la información contenida en el token inicial, lo cual incluye la información posicional del token añadida en el paso de *positional encoding*. De esta manera los encoders que se encuentren en capas más lejanas del inicio siguen teniendo acceso a la información posicional de cada token.

A.2.3. Atención

El mecanismo de atención busca definir cómo se relacionan los distintos datos de entrada entre sí. A partir de los vectores de características de los tokens ser pueden establecer relaciones de atención entre ellos. Para modelar estas relaciones, se definen 3 vectores de características diferentes, *Query vector*, *Key vector* y *Value vector*. Cada token tendrá sus vectores de query, key y value correspondientes. El vector de key representa las características que describen el token correspondiente, el vector query representa las características con las que busca emparejarse dicho token, y el vector de value representa las características de los tokens que serán utilizadas para la labor específica de la red.

Si se quiere ver lo relacionado que está un token con otro, se compara la similitud del vector query del primero con el vector key del segundo para ver si las características que representan son similares. Cabe destacar que esta dependencia no es simétrica, ya que si se quiere calcular la relación del segundo token con el primero entonces se tendrían que usar la query del segundo con la key del primero, lo cual puede dar resultados completamente diferentes, indicando que en ese caso la dependencia es en una sola dirección.

Para un conjunto de vectores query q_1 y value v_1 provenientes de un primer token, y un vector de key k_2 provenientes de un segundo token, con dimensiones de token d_k, se calcula la atención de el primero con el segundo como

$$\text{Atención}(q_1, k_2, v_1) = \text{softmax}\left(\frac{q_a \cdot k_b}{\sqrt{d_k}}\right) v_1. \tag{A.2}$$

Esta operación se debería realizar para todas las combinaciones de parejas de tokens para los vectores value de todos los tokens, lo cual requeriría mucho tiempo si se calculase de manera secuencial. Por suerte se puede calcular la atención de todos los tokens mediante una misma operación si se agrupan todos los vectores en matrices. De esta manera, la matriz de query Q contiene los vectores de query de todos los tokens de entrada, y similarmente con las matrices de key K y value V. Con esto se puede obtener la atención de todos los tokens mediante una única operación

$$\text{Atención}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V, \tag{A.3}$$

lo cual se puede calcular de manera muy rápida en hardware moderno.

Las matrices Q, K y V se generan a partir de los tokens de entrada mediante unas matrices de peso W^Q, W^K y W^V respectivamente. Estas matrices de peso se tratan de parámetros entrenables por lo que el modelo aprende cómo seleccionar las características de los tokens para obtener el mejor resultado.

Para optimizar el rendimiento del mecanismo de atención, en vez de utilizar una sola matriz Q, K y V, se utiliza el mecanismo de múltiples cabeceas, donde cada cabeza tiene sus propias matrices de pesos W^Q, W^K y W^V con las que calculan la atención sobre los tokens de entrada de manera independiente . Esto permite a cada cabeza a concentrarse en distintas características de los tokens y así poder realizar un procesamiento más complejo de los datos de entrada. Una vez se obtienen las salidas de todas la cabezas, se concatenan los resultados y se multiplican por una matriz entrenable W^O, lo cual permite obtener una salida del mismo tamaño que la matriz de tokens empleada por el modelo.

A.2.4. Decoder

Una vez calculada la atención global de los datos de entrada se generan los datos de salida. Para ello se emplea el decoder, formado por tres componentes principales, Masked-Multi-Head-Self-Attention (MMSA), Multi-Head-Self-Attention (MSA) y Multi-Layer-Perceptron (MLP).

El decoder genera un nuevo token de salida en cada paso, para ello emplea de entrada los tokens de salida generados en pasos anteriores. Estos tokens generados son procesados por el componente MMSA, el cual es idéntico al MSA con la excepción de que se enmascaran todos los tokens cuyas posiciones sean posteriores al token actual para que el calculo de la atención solo se produzca sobre los tokens en posiciones anteriores.

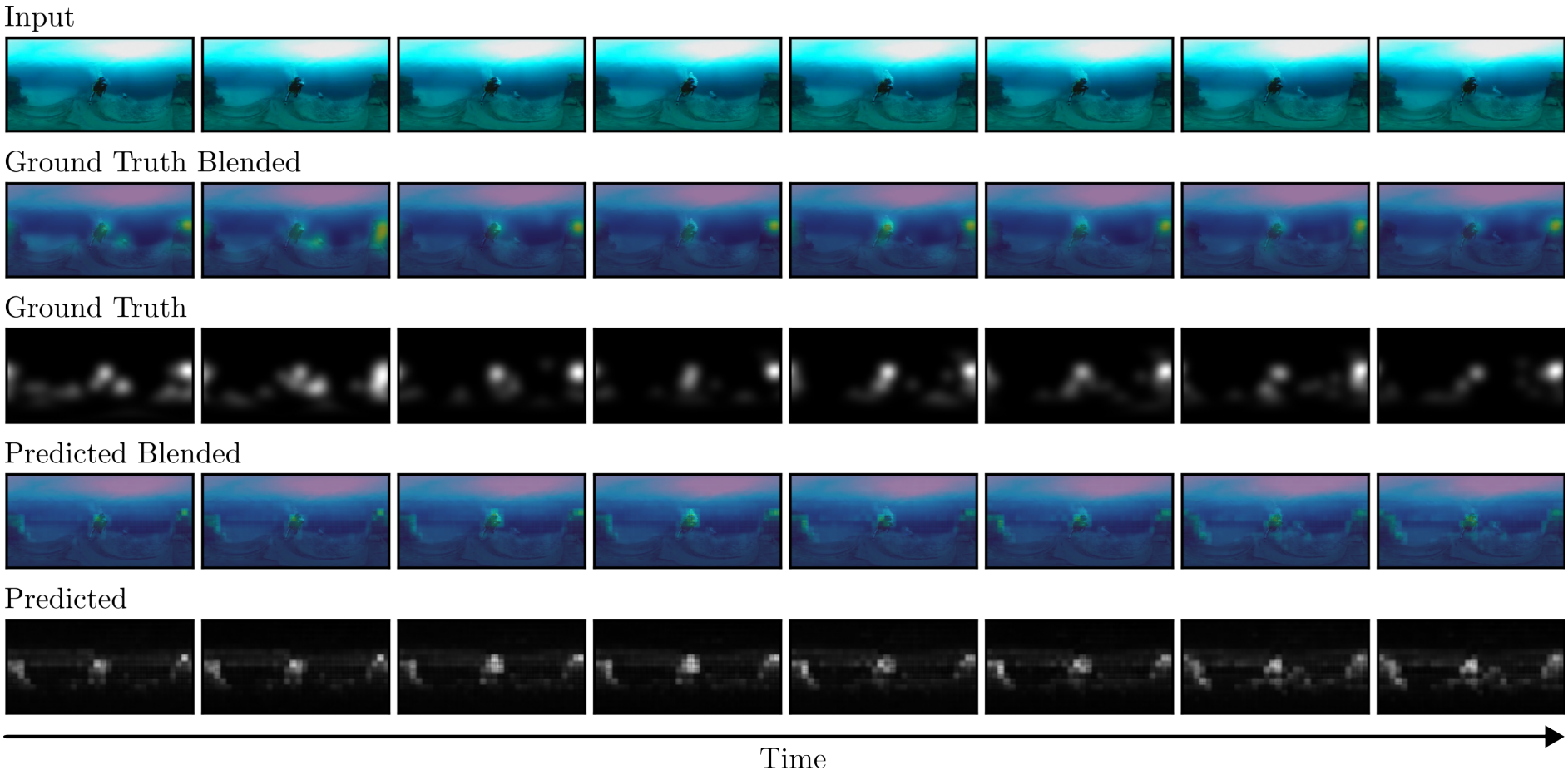
El MSA se trata de exactamente el mismo que el encoder explicado en el apartado anterior. La única diferencia se encuentra en que las matrices K y V se generan a partir de la atención global calculada por los encoders (esto es la salida del último encoder) mientras que la matriz Q se genera a partir de la salida del MMSA.

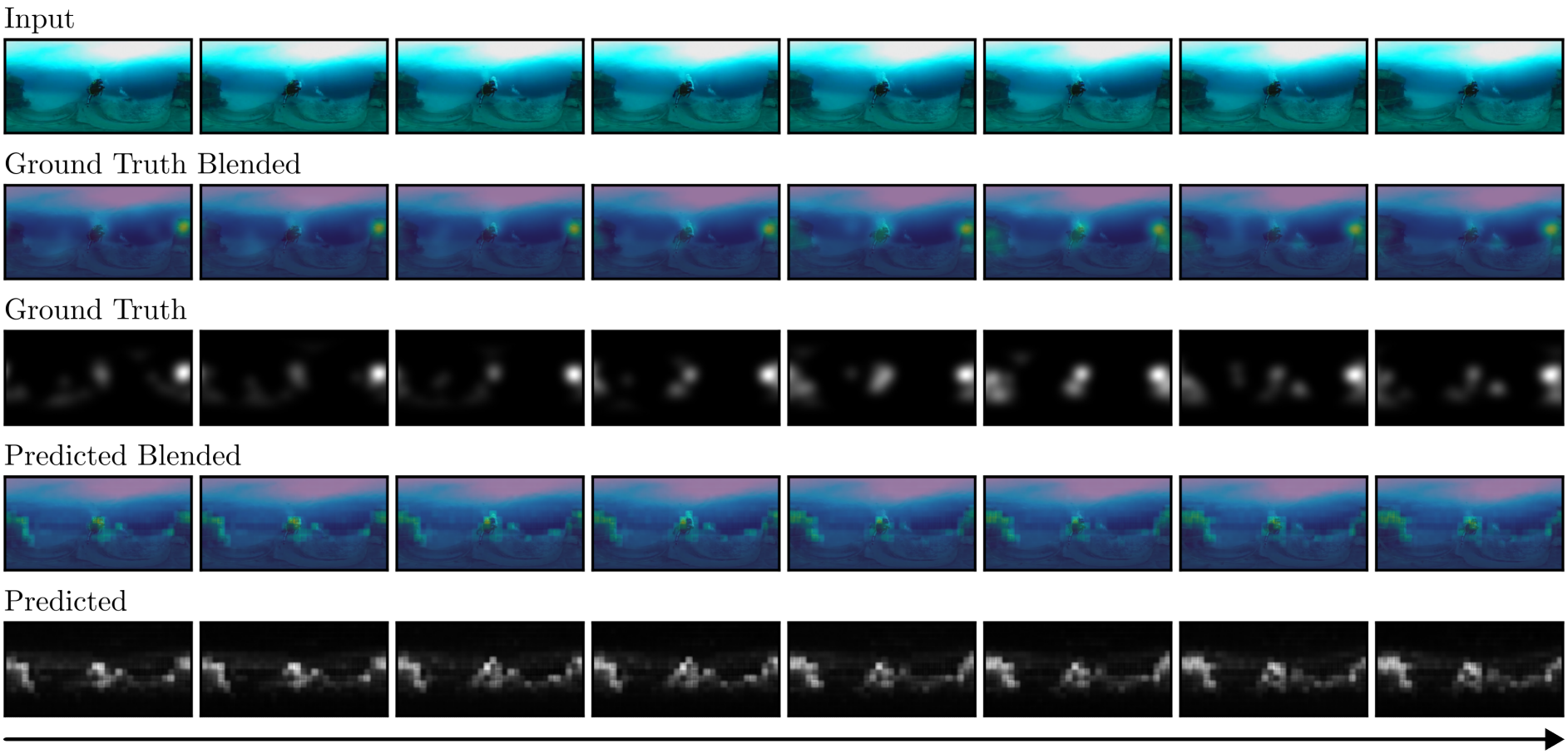
Por último se encuentra una capa MLP al final de la misma manera que en el encoder, para realizar procesamiento adicional sobre la atención calculada, con conexiones residuales de la misma manera que en el encoder.

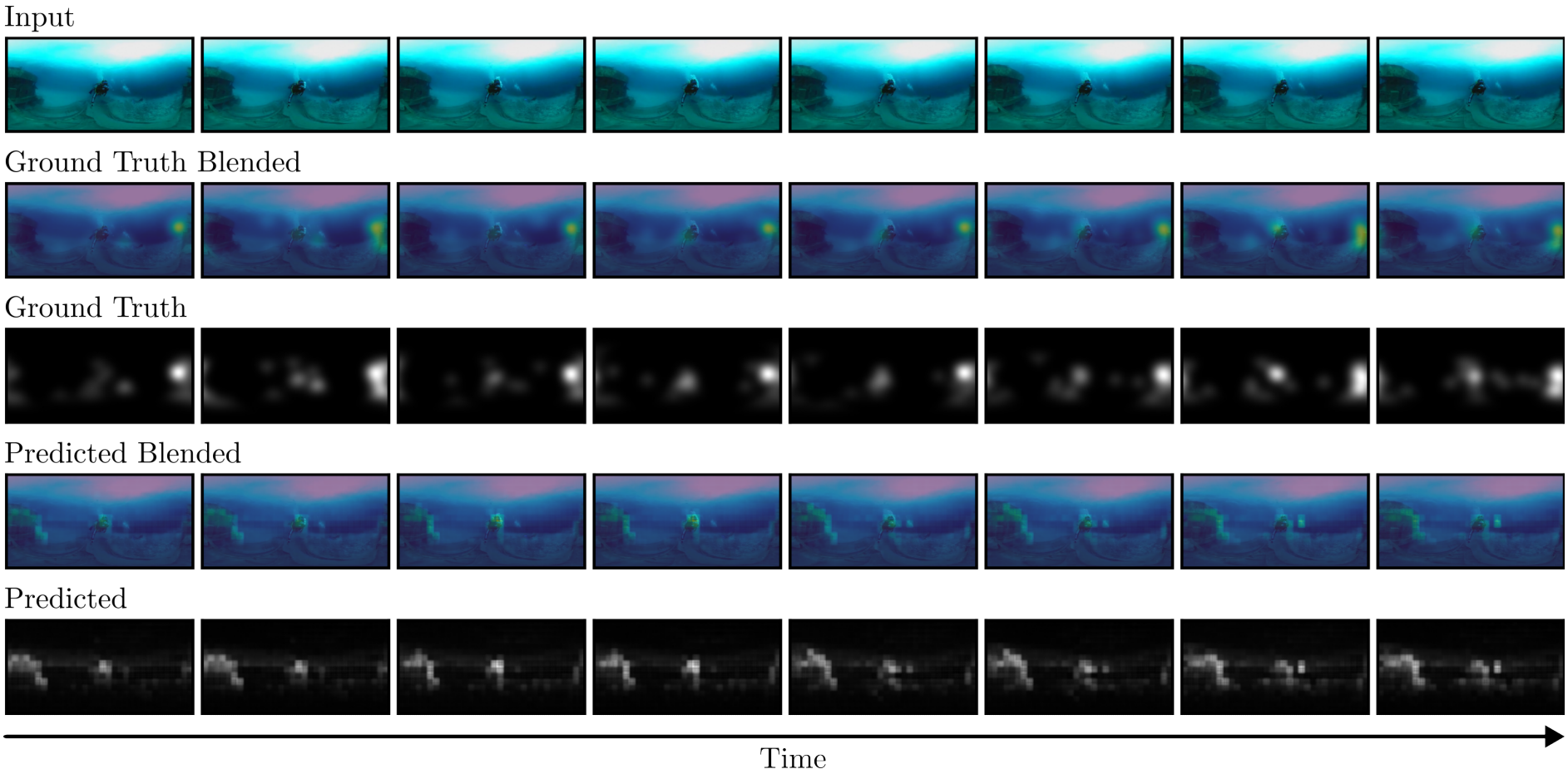
Anexos B

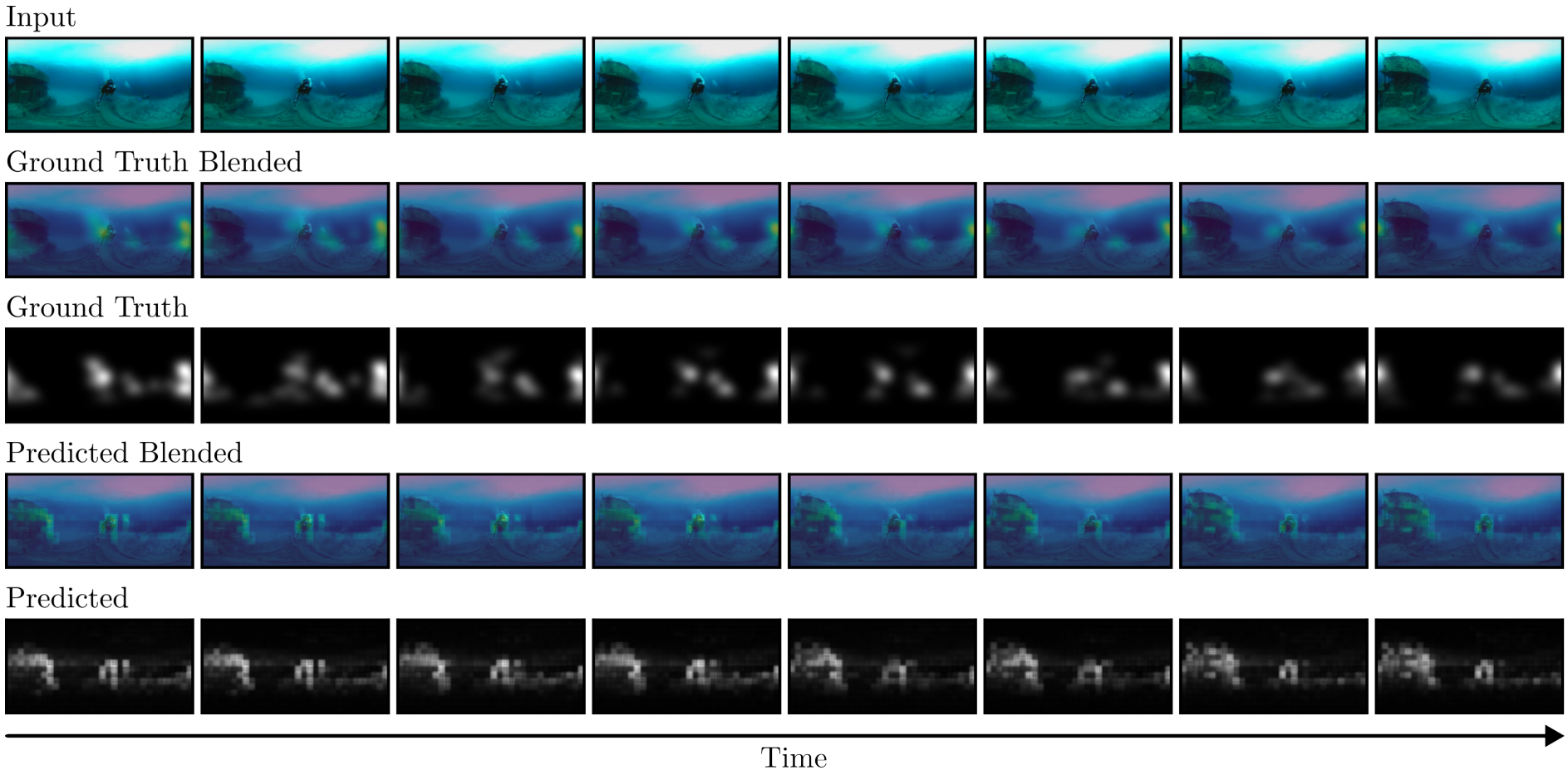
Resultados Completos

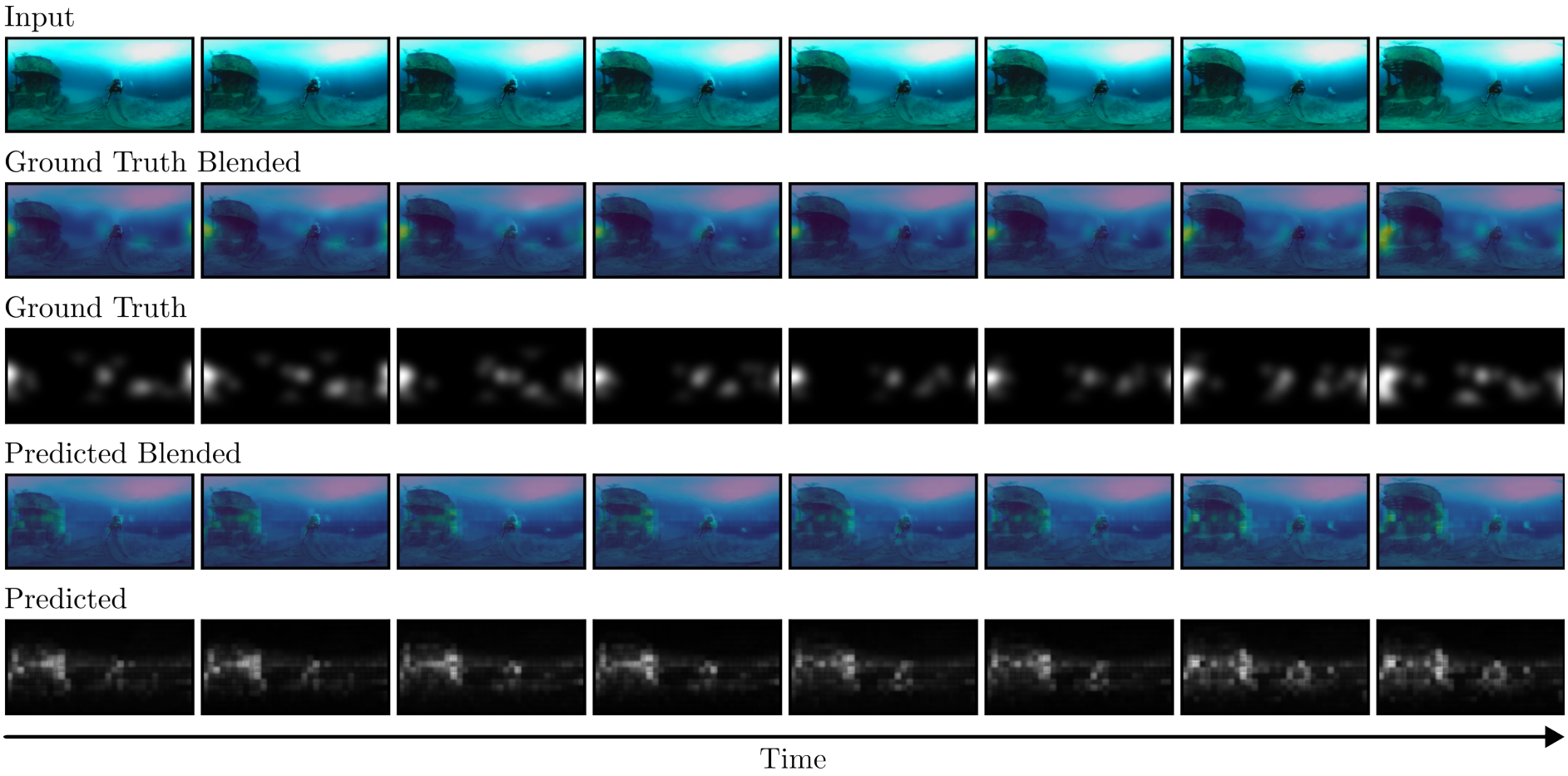
B.1. Vídeo 172 de VR-Eyetracking

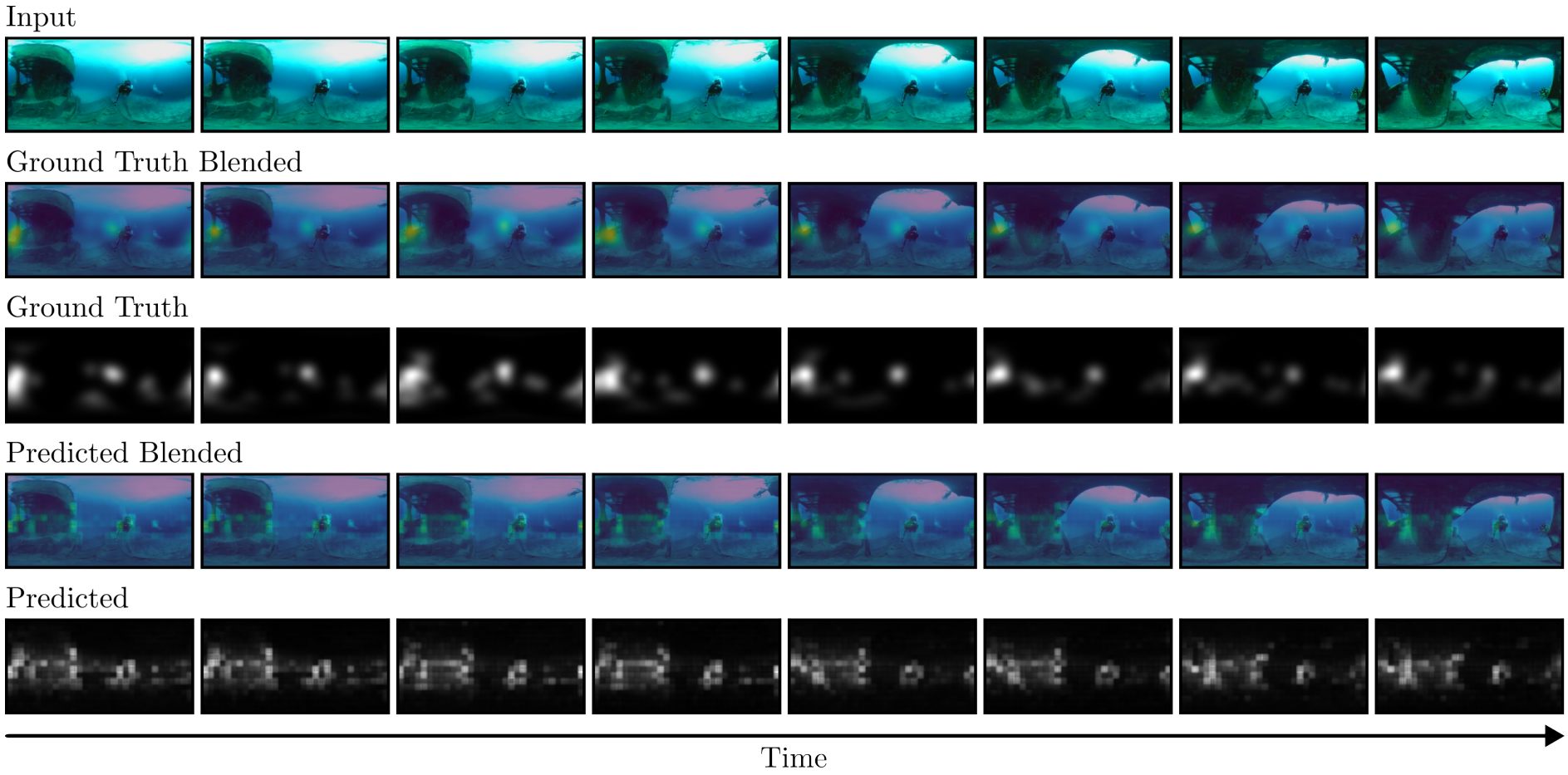

Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time


Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted

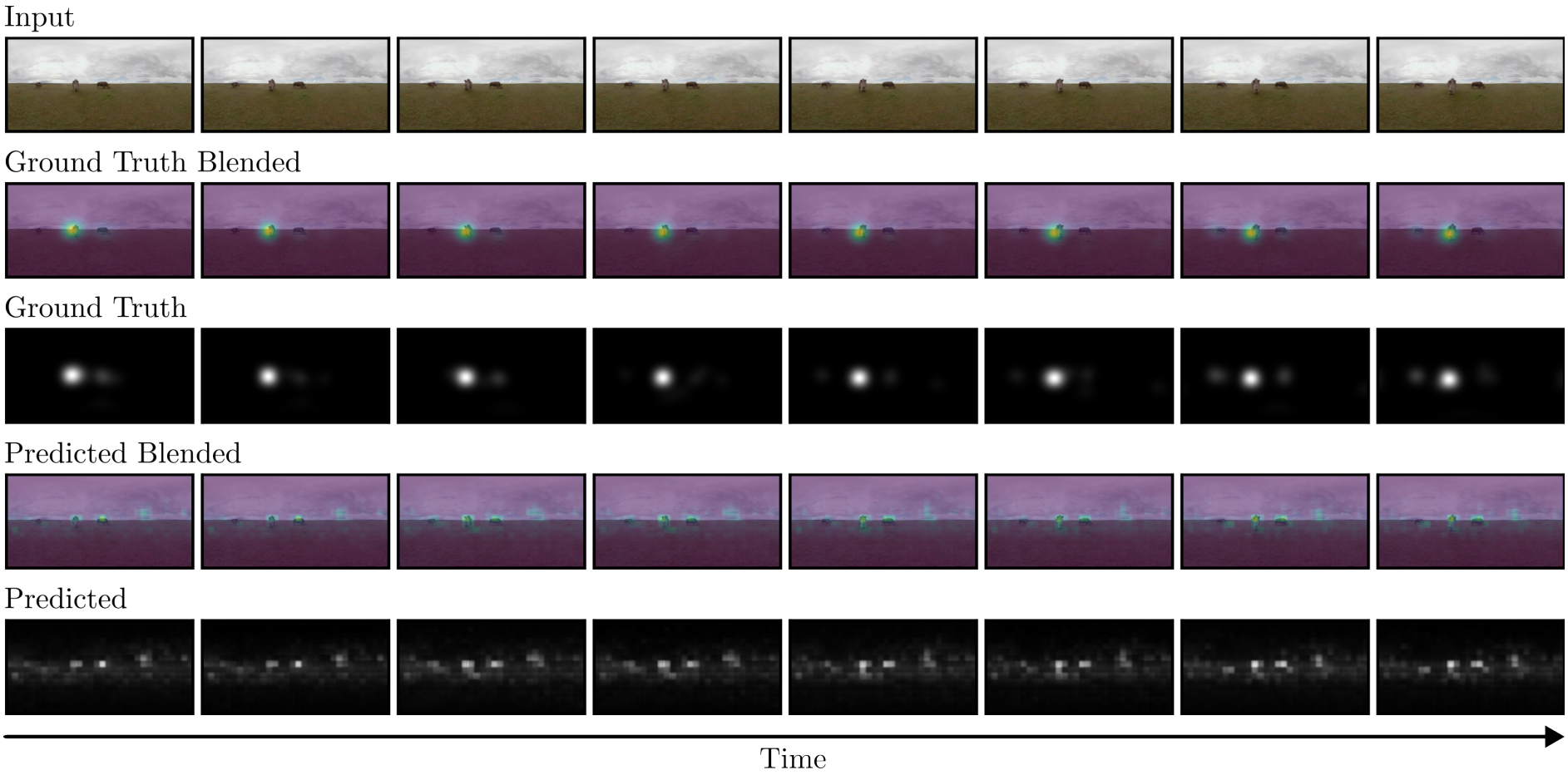

Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time

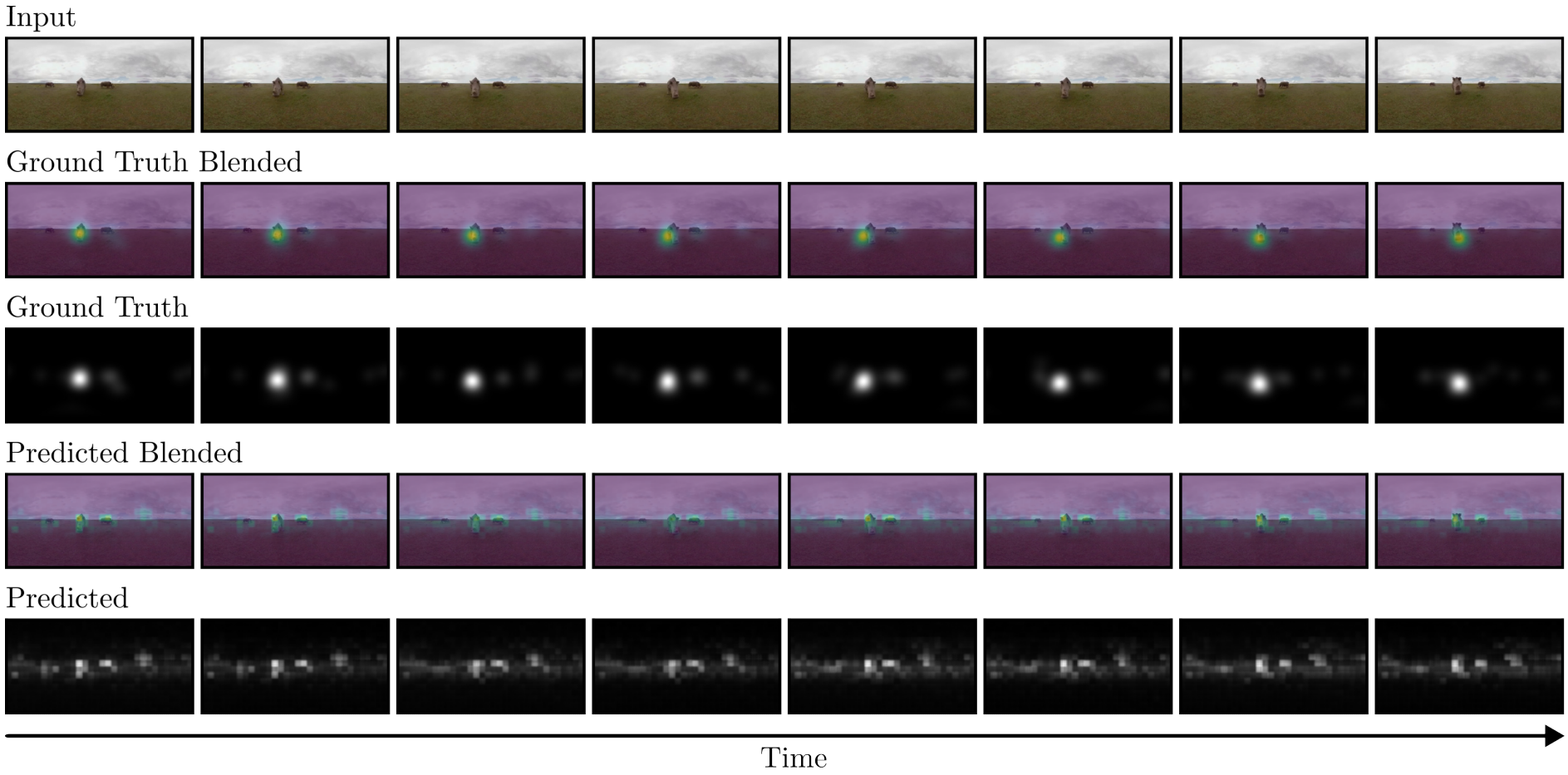

Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time

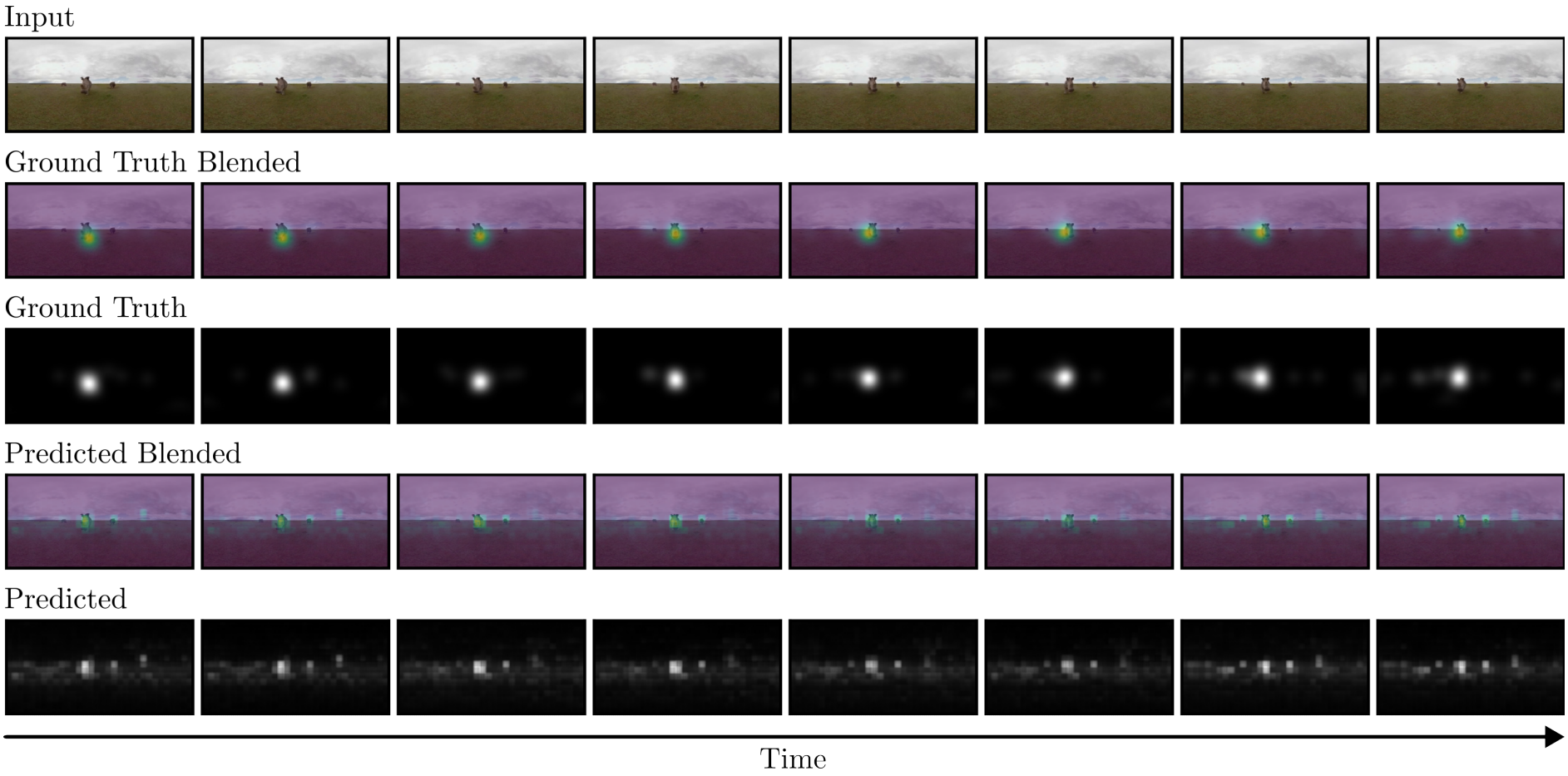

Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time

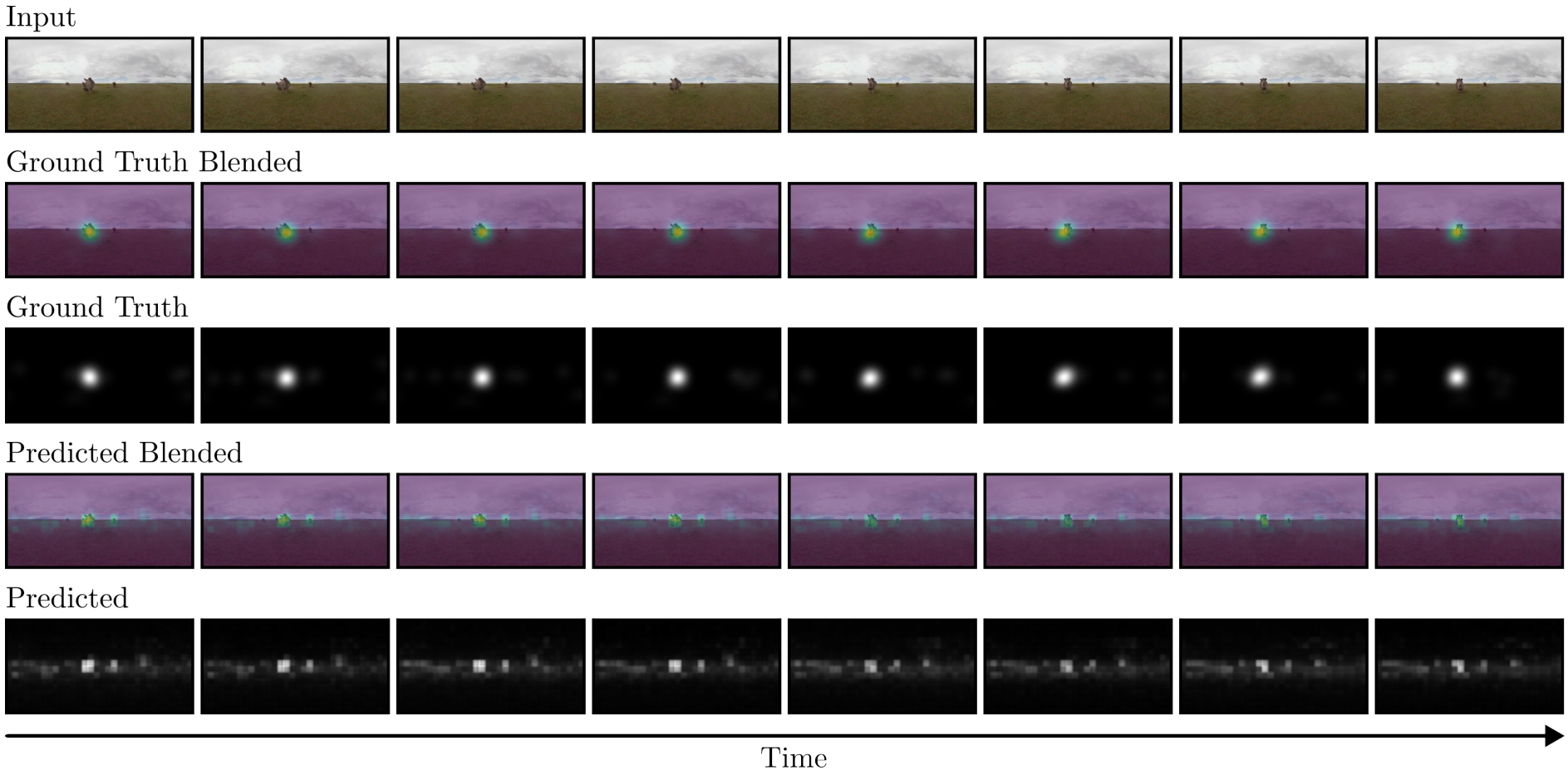

Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time

B.2. Vídeo 107 de VR-Eyetracking

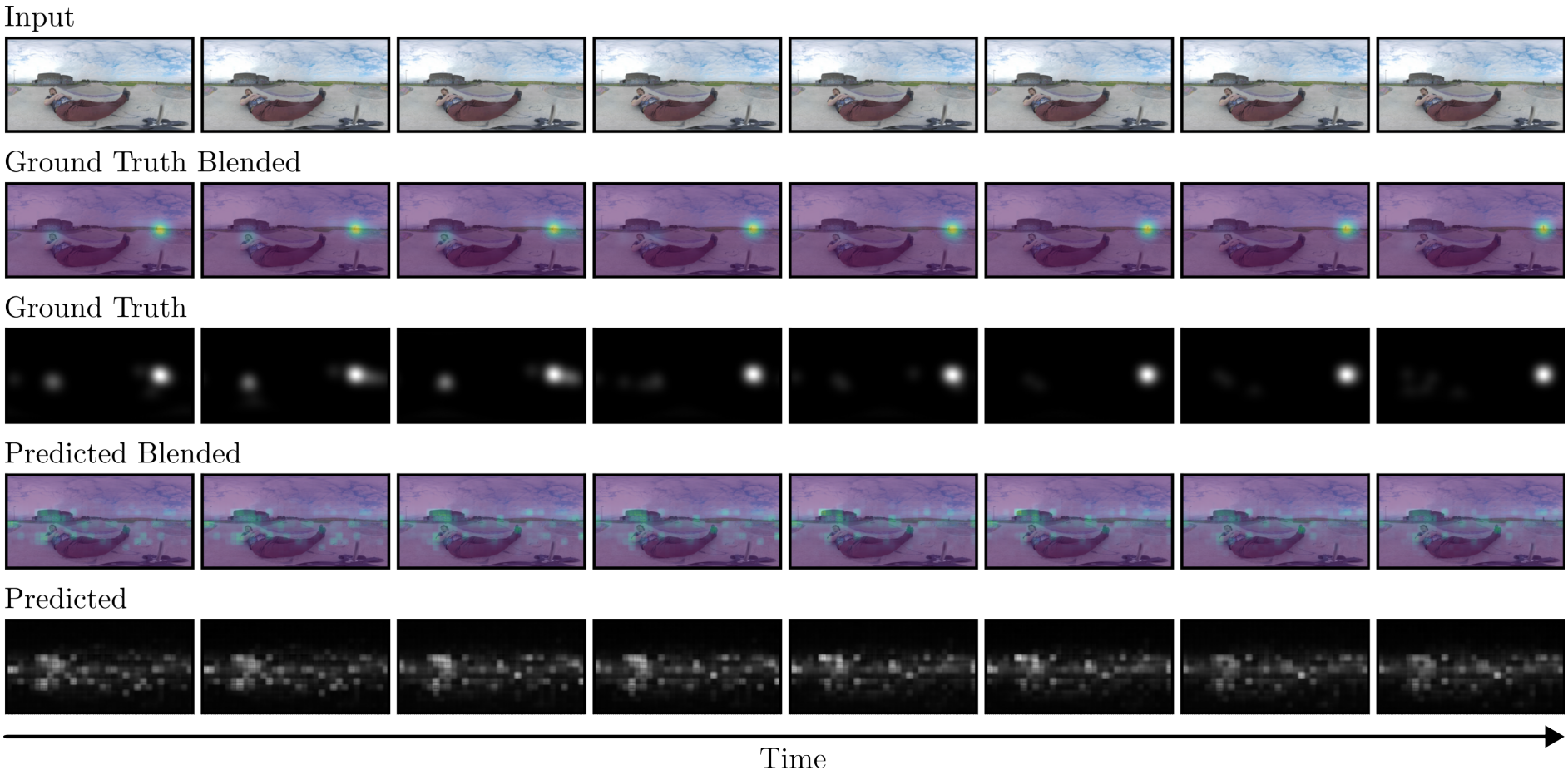

Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time

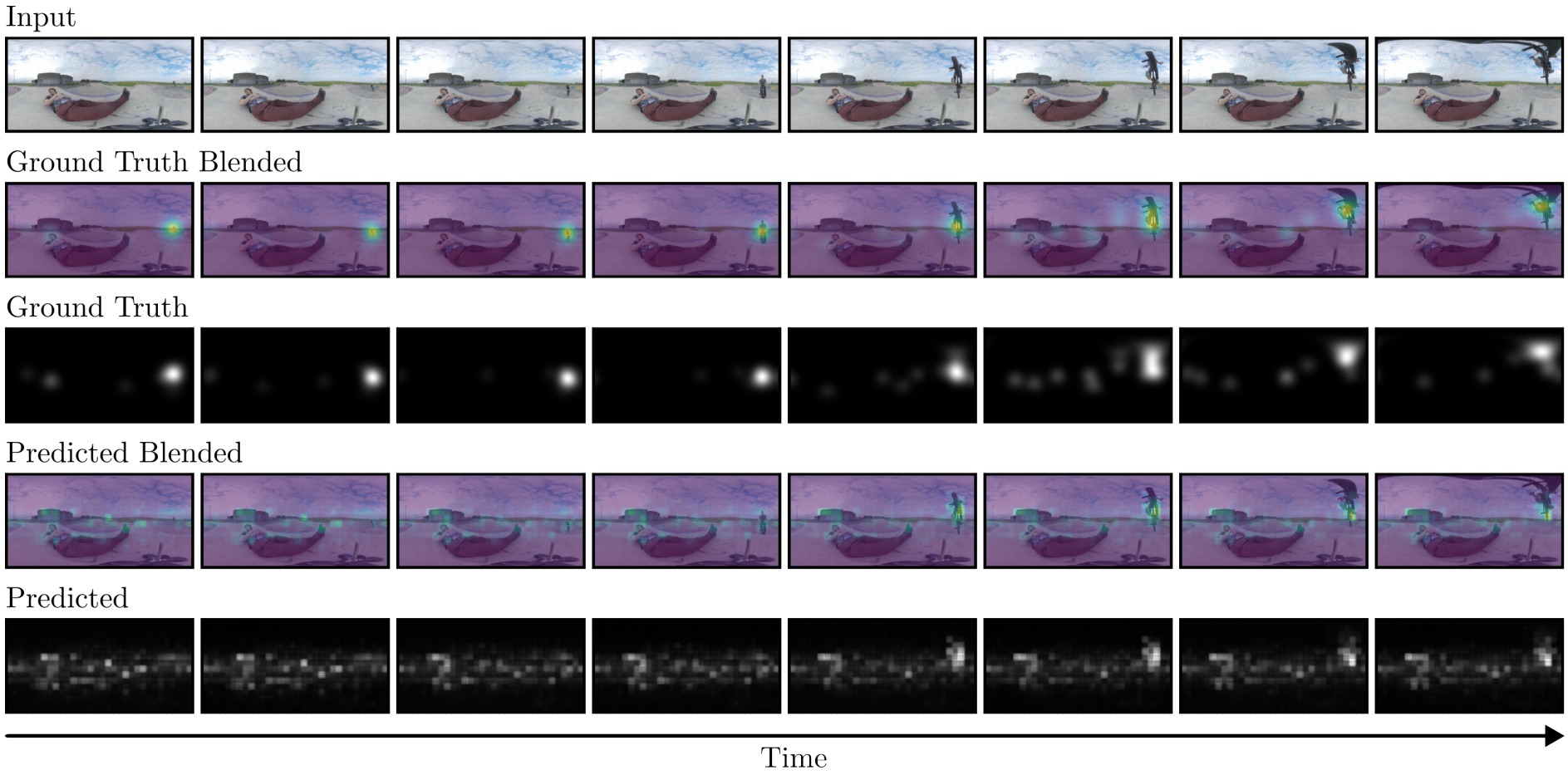

Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time

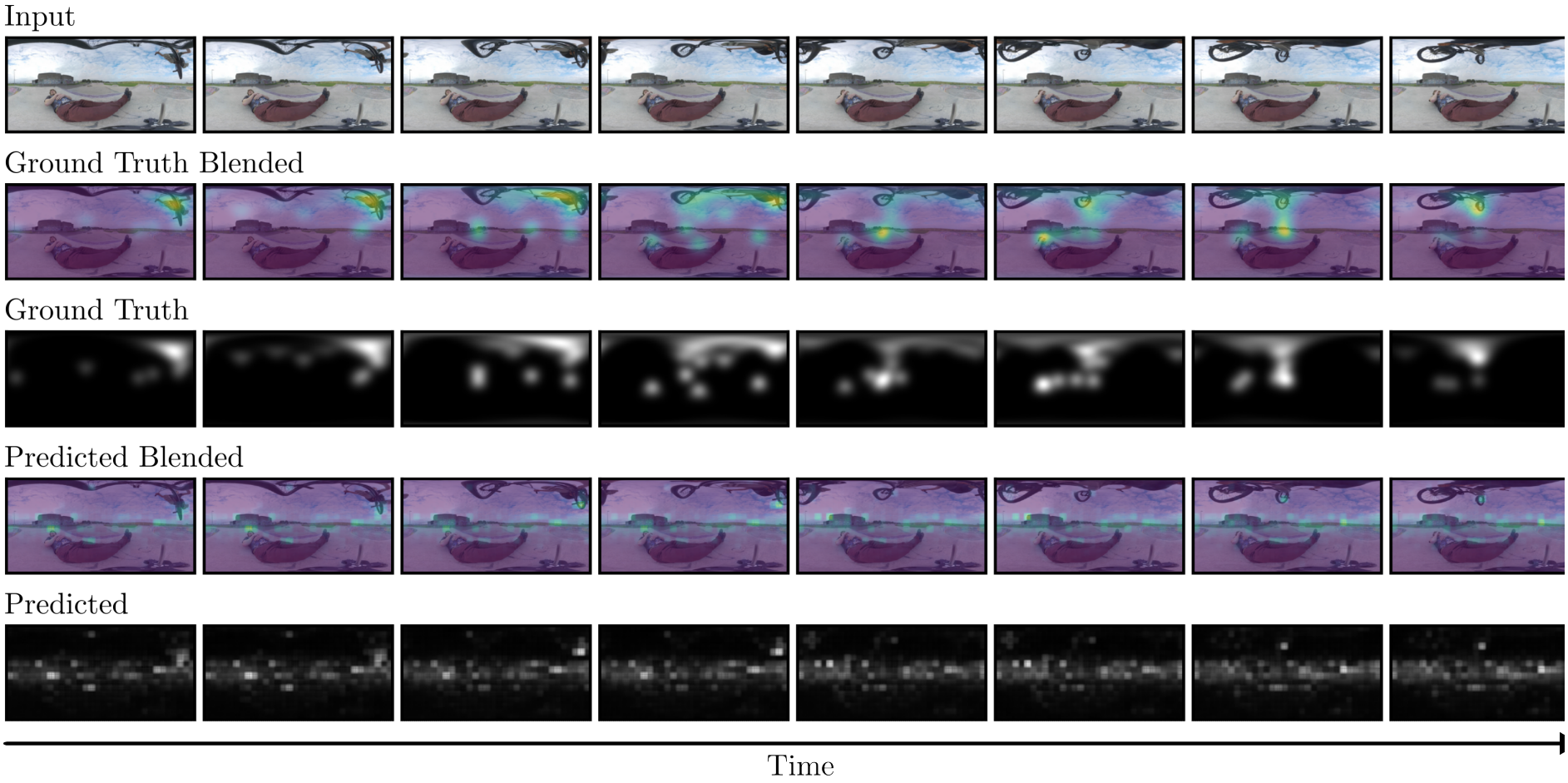

Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time

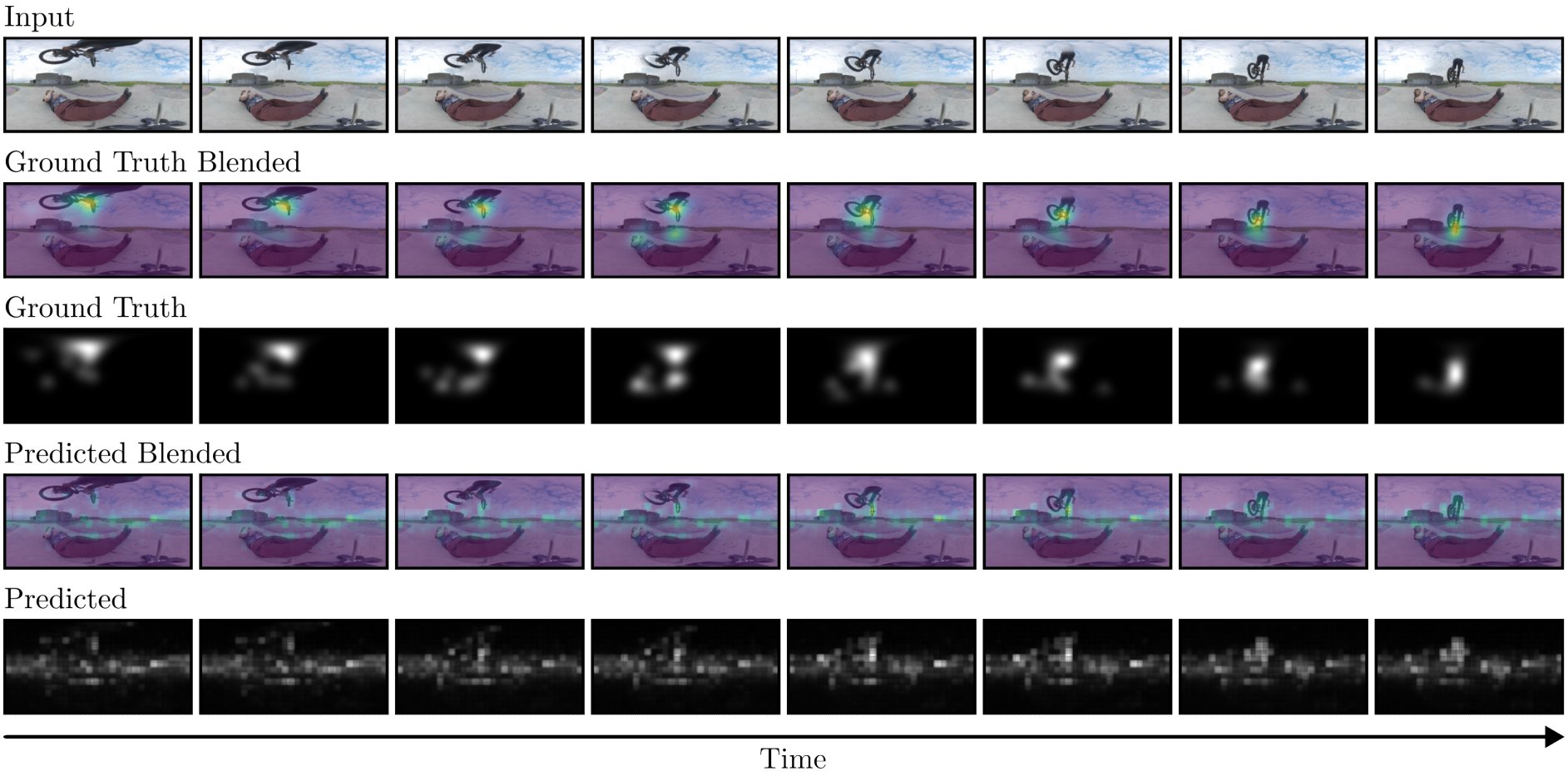

Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time

B.3. Vídeo 242 de Sports-360


Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time


Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time


Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time


Input
Ground Truth Blended
Ground Truth
Predicted Blended
Predicted
Time