Universidad Zaragoza
1542

Trabajo Fin de Grado

Un espacio invariante a objetos dinámicos mediante el uso de CycleGANs

A dynamic-object-invariant space via CycleGANs

Autor

Laura Delfau Luesma

Directora

Berta Bescós Torcal

Codirector

José Neira Parra

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2019

Resumen

En este trabajo se presenta un modelo de aprendizaje profundo para convertir imágenes que muestran contenido dinámico, como vehículos o peatones, en imágenes estáticas realistas. Para ello, se utiliza el modelo CycleGAN, el cual realiza una traducción de una imagen perteneciente a un conjunto de datos de entrada a otra imagen que pertenece al conjunto de datos que se desea obtener. El problema de traducción de imagen a imagen es una clase de problema de visión y gráficos donde el objetivo es aprender el mapeo entre una imagen de entrada y una de salida. Además, el modelo CycleGAN permite aprender simultáneamente un mapeo inverso, es decir, la traducción de imágenes estáticas a imágenes dinámicas, aunque este no será el objetivo primordial del trabajo.

La traducción de imágenes dinámicas a estáticas conlleva la detección de los objetos dinámicos contenidos en la imagen y la reconstrucción del posible fondo estático de tales zonas de la imagen, obteniendo como resultado una imagen estática realista. Para ello, sobre el modelo CycleGAN se introducen diferentes implementaciones en el entrenamiento con el propósito de mejorar los resultados obtenidos. Una de estas mejoras conlleva la incorporación de máscaras con la información dinámica de la imagen en el entrenamiento, así como un reescalado de las funciones de pérdidas de la red en función del número de píxeles dinámicos. La introducción de técnicas utilizadas en esteganografía y de técnicas de detección de esquinas en imágenes suponen también una mejora de nuestras reconstrucciones.

Las imágenes generadas con nuestro modelo pueden ser utilizadas en aplicaciones de realidad aumentada o realidad virtual, en sistemas de mapeo y localización simultáneos (SLAM), como conjunto de datos para simulaciones de coches autónomos, para representaciones virtuales, *etc.*

A la hora de validar nuestro trabajo se utiliza una red de segmentación semántica para obtener la información semántica del conjunto de imágenes traducidas. Con dicha información se realiza una evaluación para determinar la calidad de nuestras reconstrucciones. El porcentaje de píxeles dinámicos y estáticos de la salida de esta red permite evaluar el realismo de las imágenes así como la calidad de la traducción de imagen dinámica a imagen estática.

Índice

Índice de figuras

Índice de tablas

1 Introducción

1.1 Motivación

Los objetos dinámicos degradan la precisión de problemas y tareas de localización y navegación robótica basados en visión. El enfoque habitual utilizado por la comunidad científica para lidiar con estos objetos dinámicos consiste en detectarlos en las imágenes capturadas por la cámara y entonces clasificarlos como información no válida para tales aplicaciones. Trabajos recientes proponen sin embargo modificar las imágenes para que el contenido dinámico sea convertido de manera realista en estático [1, 2]. Estos se basan en la siguiente hipótesis: la combinación de experiencia y contexto permite alucinar o "pintar" la escena estática detrás de los objetos dinámicos de manera que esta tenga una apariencia consistente tanto geométrica como semánticamente.

Convertir imágenes que presentan contenido dinámico en imágenes estáticas realistas revela varios desafíos:

1. Detectar el contenido dinámico de las imágenes, como por ejemplo vehículos, animales y personas. También entran dentro de esta categoría las sombras y los reflejos que estos generan.

2. Pintar la región de la imagen que estos objetos ocluyen con una representación plausible. La imagen resultante conseguirá ser realista si las regiones pintadas son consistentes de manera tanto geométrica como semántica con el contenido estático de la imagen.

El primer desafío puede ser abordado con enfoques geométricos de visión por computador multivista si se dispone de una secuencia de imágenes. Este proceso consiste normalmente en estudiar la consistencia del flujo óptico a lo largo de la secuencia [3–5]. En el caso en el que sólo una imagen está disponible, son los algoritmos de aprendizaje profundo los que destacan en la resolución de esta tarea mediante el uso de redes neuronales convolucionales (CNNs) y el conocimiento previo de qué clases de objetos son dinámicos y cuáles no [6] (Figura 1).

En cuanto al segundo desafío, enfoques recientes de alucinación de imágenes que no utilizan aprendizaje profundo suelen utilizar estadísticas de la parte de la imagen restante para para rellenar los agujeros [7]. Mientras este enfoque produce generalmente resultados suaves, está limitado por las estadísticas de la imagen y no tiene el concepto de coherencia semántica. Sin embargo, las redes neuronales pueden aprender contenidos

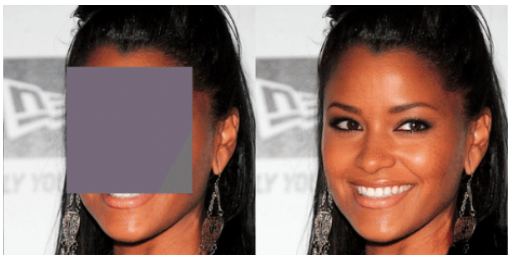
(a) Los coches aparcados pueden ser clasificados como dinámicos gracias al uso de CNNs entrenadas con el conocimiento previo de qué clases son dinámicas, a pesar de que no se están moviendo.

(b) El sofá sería clasificado como estático con una CNN, pero si se utiliza geometría multivista se podría clasificar como dinámico. Las personas moviéndose podrían ser detectadas con ambos métodos.

Figura 1: Ejemplos de diferentes tipos de objetos dinámicos según cómo detectarlos.

semánticos y representaciones escondidas muy significativas que han sido utilizadas recientemente para hacer "inpainting" (ver Figura 2).

(a) Las zonas de la imagen dañada pueden ser correctamente recuperadas utilizando las estadísticas de la propia imagen, al ser estas pequeñas. No es necesario tener un entendimiento semántico de la imagen para reconstruirla.



(b) Esta reconstrucción puede hacerse únicamente con técnicas de aprendizaje, puesto que la imagen reconstruida tiene coherencia semántica además de geométrica.

Figura 2: Ejemplos de métodos de inpainting con y sin aprendizaje profundo.

Ambos desafíos pueden ser combinados y vistos como una única tarea: trasladar una imagen dinámica en su correspondiente representación estática. En esta dirección Isola *et al.* [8] proponen una solución para realizar traslación de imagen a imagen de manera generalizada, a la cual se le llama Pix2Pix.

El principal problema de este sistema, así como de otros que trabajan con aprendizaje profundo con imágenes, es que para lograr una buena precisión es necesario obtener grandes conjuntos de datos emparejados. Esto puede resultar muy costoso dependiendo de la aplicación, y no siempre es posible. En nuestro caso, en el que queremos trasladar una imagen con contenido dinámico en una imagen estática, sería altamente costoso obtener pares de imágenes tomadas con las mismas características: enfoque, ilumina-

ción, perspectiva, *etc.* con objetos dinámicos y sin ellos. El trabajo Empty Cities [1, 2] utiliza el simulador de conducción autónoma CARLA [9] para conseguir los datos de entrenamiento. Consiguen resultados impresionantes eliminando los objetos dinámicos de imágenes renderizadas con el mismo simulador pero, sin embargo, estos resultados se ven altamente comprometidos cuando se trata de eliminar los objetos de imágenes de escenas del mundo real. En este trabajo de fin de grado se quiere abordar este obstáculo utilizando un modelo que permita trabajar con un conjunto de datos no emparejados para que así sea posible trabajar con imágenes reales en vez de sintéticas.

Las imágenes generadas pueden utilizarse en aplicaciones de realidad aumentada o realidad virtual: uno podría moverse virtualmente por una ciudad en la que no hubiera tráfico de personas o de coches. También pueden ser de interés para la cinematografía, o para empresas proveedoras de representaciones virtuales de las ciudades (como el "street-view" de Google) como una medida de privacidad para reemplazar el proceso de emborronamiento de matrículas o caras. Otras aplicaciones de estas imágenes que tienen alta demanda hoy en día son el aumento de datos para simulaciones de coches autónomos. Por último, estas imágenes podrían ser de gran utilidad para sistemas de localización basados en imágenes como puede ser el *SLAM*. *SLAM* es un sistema de mapeo y localización simultáneos (del inglés *Simultaneous Localization and Mapping*), donde es común asumir que la escena en la que se trabaja es estática. La precisión de estos sistemas se ve altamente afectada su precisión cuando dichas escenas presentan objetos dinámicos [10].

1.2 Objetivo

El principal objetivo del trabajo es convertir imágenes reales RGB que muestran contenido dinámico, como vehículos o peatones, en imágenes RGB estáticas realistas. Para ello, se va a utilizar el modelo de traslación de imágenes CycleGAN [11], el cual utiliza un conjunto de entrenamiento de imágenes no emparejadas. Esto es al contrario que su predecesor Pix2Pix [8], el cual necesita un conjunto de entrenamiento de imágenes perfectamente emparejadas. Al igual que el trabajo Empty Cities [2] añade modificaciones a Pix2Pix para adaptarlo al problema concreto de trasladar imágenes con contenido dinámico en estático, en este trabajo se quiere adaptar el complejo modelo de CycleGAN a este mismo problema con el propósito de mejorar los resultados.

Por otro lado, el modelo CycleGAN [11], el cual será presentado en detalle en la sección 3.2, añade un mapeo inverso, permitiendo aprender simultáneamente la traducción de imágenes estáticas a dinámicas en el mismo entrenamiento. La obtención de

imágenes dinámicas realistas no es el objetivo principal de este trabajo pero también se procederá más adelante a analizar estos resultados.

1.3 Alcance

El trabajo planteado continúa la línea de investigación de **Empty Cities** [2] buscando solventar la limitación principal que presenta: Empty Cities necesita de la utilización de un conjunto de imágenes sintéticas para poder así conseguir datos emparejados. Mientras que los resultados obtenidos para imágenes sintéticas son prometedores, los resultados para imágenes reales pierden realismo. Además, Empty Cities trabaja con imágenes en escala de grises, viéndose limitado el rango de sus aplicaciones. Este trabajo se ha desarrollado sin embargo con imágenes a color para que esto no suceda.

Para poder alcanzar los objetivos propuestos en primer lugar se ha realizado un estudio en detalle del estado del arte, y más en concreto del trabajo **Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks** [11], también conocido como CycleGAN, así como del código que implementa dicho modelo. También será necesario configurar el entorno de trabajo con los requisitos necesarios: *Linux*, *Python 3*, *CPU o NVIDIA GPU*, *CUDA CuDNN* y *Pytorch* entre otros.

A continuación, se ha procedido a la creación de un conjunto de datos no emparejados de imágenes reales urbanas tomadas desde un coche. Una vez se ha tenido este dataset se ha procedido a investigar y desarrollar diferentes implementaciones que puedan ofrecer una mejora en los resultados del entrenamiento del modelo CycleGAN [11]. Finalmente, se han evaluado las implementaciones realizadas, y para ello se han investigado modelos de *segmentación semántica*, eligiendo **ERFnet** [6] como candidato final, y se implementan dos evaluaciones basadas en la información semántica del píxel.

1.4 Descripción del documento

La estructura del documento es la siguiente. En el Capítulo 2 están descritos los fundamentos teóricos de redes generativas adversarias, redes neuronales convolucionales y aprendizaje profundo sobre los que se apoya el trabajo. A continuación, en el Capítulo 3, se explica la línea de investigación que se sigue, el modelo de traducción de imagen a imagen utilizado (CycleGAN) y las implementaciones desarrolladas para conseguir el objetivo del trabajo.

El Capítulo 4 describe cómo se ha obtenido el dataset utilizado para entrenar nuestro modelo. La evaluación cuantitativa junto con ejemplos cualitativos de los resultados obtenidos se encuentran en el Capítulo 5. El Capítulo 6 y el Capitulo 7 contienen las conclusiones y trabajo futuro a realizar respectivamente. Finalmente, el diagrama de Gantt del proyecto se muestra en el Anexo B.

2 Fundamentos

El trabajo presentado utiliza el modelo de aprendizaje profundo CycleGAN [11], el cual es un modelo que realiza traducción de imagen a imagen con conjuntos de datos de entrenamiento no emparejados. CycleGAN se basa en las redes generativas adversarias conocidas como GANs por sus siglas en inglés (*Generative Adversarial Networks*). Las GANs han tenido un enorme éxito en los últimos años. Su arquitectura está compuesta por dos redes neuronales, que suelen ser convolucionales cuando se trabaja con imágenes. A continuación se van a explicar en detalle los fundamentos básicos necesarios para llegar a entender el funcionamiento de una CycleGAN [11].

2.1 Convolución

Una convolución en dos dimensiones consiste en aplicar un filtro a una imagen utilizando una matriz que contiene los coeficientes del filtro, a la cual se le suele llamar máscara o *kernel*. El valor del píxel de salida se calcula mediante la suma de los píxeles vecinos ponderada con los coeficientes del kernel. En la Figura 3 se presenta un ejemplo de convolución en dos dimensiones. La zona azul es el mapa de características de entrada y la zona verde es el mapa de salida. Un núcleo (área sombreada) se desliza a través del mapa de entrada. En cada lugar se calcula el producto entre cada elemento del núcleo y el elemento de entrada al que se superpone y se suman los resultados para obtener así la salida en el lugar actual [12].

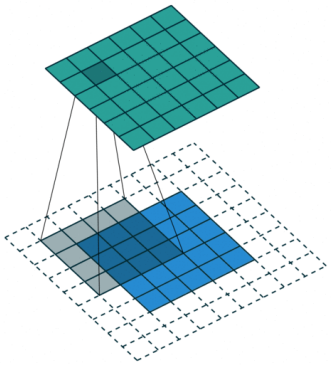


Figura 3: Cálculo de los valores de salida de una convolución [12]

Además del kernel de la convolución, también es necesario definir los siguientes parámetros: zancada o *stride* y relleno o *padding*. La zancada es la distancia entre dos posiciones consecutivas del núcleo, y el relleno es el número de píxeles concatenados al principio y al final de un eje en el mapa de características de entrada. En la Figura 3

el relleno está representado por las celdas sin color que bordean el mapa de entrada. El relleno permite obtener un mapa de características de salida del mismo tamaño que el mapa de entrada, y la zancada permite reducir el tamaño del mapa de salida.

2.2 Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (también llamadas CNNs del inglés *Convolutional Neural Networks*) son un tipo de red neuronal artificial diseñada para emular el comportamiento de la corteza visual. Las neuronas corticales individuales responden a los estímulos solo en una región restringida del campo visual conocida como campo receptivo. Por otro lado, los campos receptivos de diferentes neuronas se superponen parcialmente de tal manera que cubren todo el campo visual. Esta es la base detrás de las Redes Neuronales Convolucionales, donde el objetivo es aprender características específicas de los datos de entrada.

Las CNN consisten en múltiples capas de filtros convolucionales de diferentes dimensiones. Después de cada capa, normalmente suele haber una función para realizar un mapeo causal no-lineal. Cada parte de una CNN está entrenada para realizar una tarea, por lo que el entrenamiento de cada una de ellas se realiza individualmente. Este tipo de redes son las más utilizadas cuando se trabaja con imágenes. De esta manera, las CNN son capaces de transformar la entrada original, capa por capa, usando técnicas convolucionales y de reducción de muestras.

Tienen aplicaciones en el reconocimiento de imágenes y de vídeo [13], sistemas de recomendación, clasificación de imágenes [14] [15], reconstrucción de imágenes [16], segmentación semántica [17], entre otras.

2.3 Redes Generativas Adversas

Las **Redes Generativas Adversas** (del inglés *Generative Adversarial networks*, GANs) son un modelo generativo compuesto por dos redes neuronales, una generadora denominada **Generador** y otra discriminadora denominada **Discriminador** (Figura 4).

La red generadora aprende a asignar elementos de un espacio latente a una distribución de datos determinada, mientras la red discriminadora diferencia entre elementos

de la distribución de datos original y los candidatos producidos por el generador. El objetivo del aprendizaje del Generador es aumentar el índice de error del Discriminador, es decir, *engañar* a la red discriminadora produciendo nuevos elementos sintéticos que parecen pertenecer a la distribución de datos originales. Por otro lado, el objetivo del Discriminador es aumentar el índice de error del Generador, es decir, aprender a *diferenciar* entre imágenes reales y aquellas producidas por el Generador.

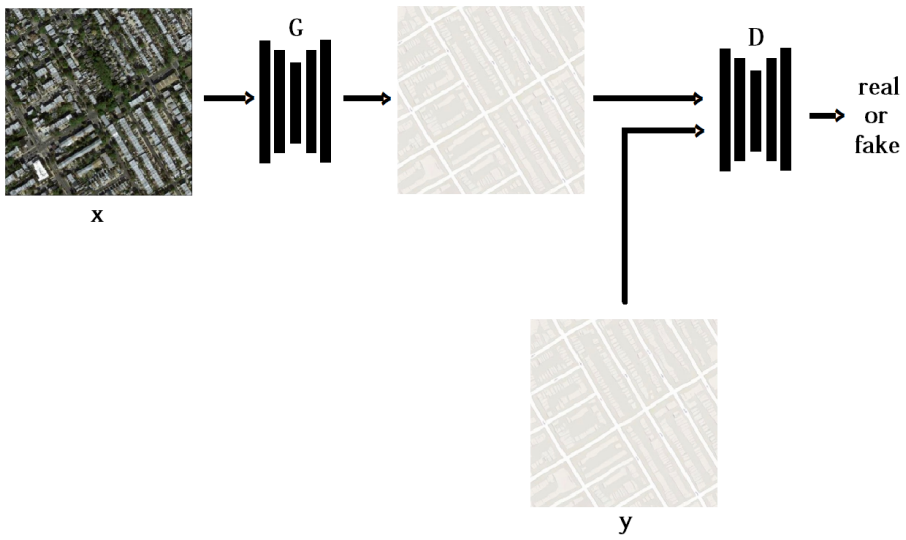


Figura 4: Representación del proceso que siguen las imágenes a través del generador G y discriminador D. En este ejemplo se aprende a traducir imágenes de una distribución que contiene muestras de vista de satélite a una distribución de imágenes de vista de mapa. El generador traduce la imagen de una vista satélite x en un mapa $G(x)$ y, el discriminar evalúa si es real comparándolo con una foto real de un mapa y [18].

Las GANs han logrado resultados asombrosos en la generación de imágenes y en el aprendizaje de la representación. La clave del éxito de las GANs es la idea de una pérdida adversa que obliga a que las imágenes generadas sean, en principio, indistinguibles de las imágenes en el dominio de destino [19].

Para aprender la distribución del generador p_G sobre los datos x se representa un mapeo al espacio de datos como $G(\theta_G)$, donde G es una función diferenciable representada por un perceptrón multicapa con parámetros θ_G. También se define un segundo perceptrón multicapa $D(x; \theta_G)$ que da salida a una matriz la cual indica la probabilidad de que cada parche de la imagen sea falso o real. $D(x)$ representa la probabilidad de que x provenga de los datos y no de p_G [19].

2.4 Traducción de imagen a imagen

La traducción de imagen a imagen es una clase de problema de visión y gráficos donde el objetivo es aprender el mapeo entre una imagen de entrada y una imagen de salida. Muchas tareas en el procesamiento de imágenes pueden definirse como un problema de traducción de imagen a imagen. Por ejemplo, la transferencia de estilo de una imagen, la transfiguración de objetos, la superresolución, *etc* (Figura 5).

(a) Transferencia de estilo (b) Transfiguración de objeto

Figura 5: Ejemplos de traducciones de imagen a imagen

La mayoría de trabajos que hacen traducción de imagen a imagen emplean en el entrenamiento datos emparejados píxel a píxel. Es decir cada píxel de cada imagen de la distribución de entrada tiene una etiqueta en la distribución de los datos de salida. Uno de los trabajos más relevantes en este área es Pix2Pix [20], el cual utiliza una red generativa adversaria condicional para obtener un modelo de aprendizaje profundo capaz de generalizar a cualquier tipo de imagen de entrada y salida.

Para algunas aplicaciones no siempre es posible obtener datos emparejados, o el coste que esto conlleva es demasiado alto. En los últimos años la comunidad de visión por computador ha abordado este problema creando modelos de traducción de imágenes que no requieran datos de entrenamiento completamente emparejados. *CoGAN* [21] utiliza una estrategia de reparto de pesos para aprender una representación común entre los dominios [11], Ming-Yu Liu *et al.* [22], entre otros, basan su trabajo en redes generativas adversas (GANs) y autocodificadores variacionales (VAEs) . A diferencia de los enfoques anteriores, **Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks** [11] no se basa en ninguna función de similitud predefinida y específica entre la entrada y la salida, ni supone que la entrada y la salida deban estar en el mismo espacio. De esta forma, proporcionan una solución de propósito general para muchas tareas de visión y gráficos. Por esta razón, hemos elegido el modelo que ellos ofrecen para el desarrollo de este trabajo.

Se quiere destacar con el ejemplo de la Figura 6 la diferencia, para el objetivo

concreto de este trabajo de fin de grado, entre unos datos de entrenamiento emparejados y unos no emparejados. Todos los píxeles de la imagen de la izquierda de la Figura 6a tienen su etiqueta en la imagen derecha. Sin embargo, en el caso de la Figura 6b no hay una equivalencia píxel a píxel entre las dos imágenes. En el caso de la Figura 6a es muy costoso obtener un número significativo de pares de imágenes con las mismas características de enfoque, iluminación, perspectiva, *etc.* y el trabajo Empty Cities recurre entonces al uso de un simulador de coches autónomos. Por otro lado, el uso de un modelo como el de CycleGAN permite el uso de un dataset de pares de imágenes no emparejadas, como las del ejemplo de la Figura 6b, y así trabajar directamente con imágenes reales de manera menos costosa.

(a) Par de imágenes emparejadas

(b) Par de imágenes no emparejadas

Figura 6: La obtención de imágenes emparejadas consiste en conseguir pares de imágenes con las mismas características (enfoque, iluminación, perspectiva, *etc.*). Esto puede resultar muy costoso dependiendo de la aplicación, y no siempre es posible. Por ello, se suele utilizar pares de imágenes sintéticas (a) [1]. En cambio, la utilización de imágenes no emparejadas (b) hace posible trabajar con imágenes reales.

3 Metodología

3.1 Empty Cities

El trabajo realizado continúa la línea de investigación de **Empty Cities** [2], donde se presenta un marco de aprendizaje profundo para convertir imágenes que muestran contenido dinámico, como vehículos o peatones, en imágenes estáticas realistas. **Empty Cites** desarrolla un modelo de aprendizaje supervisado para el cual necesita como datos de entrenamiento pares de imágenes dinámicas y estáticas con una correspondencia pixel a pixel. Debido a la arquitectura del modelo utilizado y a la dificultad para obtener suficientes datos de entrenamiento, se ven obligados a utilizar un simulador para generar datos emparejados con y sin objetos dinámicos.

Debido a que los datos sintéticos que utilizan en el entrenamiento pertenecen a una distribución diferente a imágenes de escenarios reales, la calidad de sus resultados se ve comprometida cuando se trabaja con tales imágenes. En la Figura 7 se pueden ver algunos ejemplos de los resultados que Empty Cities obtiene en imágenes sintéticas (Figura 7b). La reconstrucción del fondo estático detrás de los objetos tiene coherencia tanto semántica como geométrica. Sin embargo, puede verse como los resultados cuando se utilizan imágenes reales (Figura 7d) han perdido esta coherencia. Además, este modelo trabaja con imágenes en escala de grises, limitando su rango de aplicación.

(a) sintética dinámica (b) fake estática (c) real dinámica (d) fake estática

Figura 7: Comparación del resultado obtenido con el Modelo Empty Cities [1] con imágenes sintéticas del simulador de conducción autónoma CARLA [9] (7b) e imágenes reales (7d).

3.2 CycleGAN

Una CycleGAN es una **Red Generativa Adversarias** [11] (del inglés *Generative Adversarial Network*, GAN) que utiliza dos generadores y dos discriminadores (Figura 8). El objetivo de la CycleGAN es traducir imágenes de un dominio de origen **X** a un dominio de destino **Y** sin el uso de datos emparejados, motivación obtenida por la dificultad y el coste de obtener datos de entramiento emparejados. Para ello, la red aprende un mapeo G:**X**→**Y** de tal manera que la distribución de imágenes de G(**X**) sea indistinguible de la distribución **Y** utilizando **pérdidas adversas** (Eqn. 1) para ajustar la distribución de las imágenes generadas a la distribución de los datos objetivo. Además, añade un mapeo inverso *F:**Y***→***X*** e introduce una **pérdida de consistencia del ciclo** (Eqn. 2) para hacer cumplir *F(G(**X**))*≈***X*** (y viceversa) y así evitar que los mapeos aprendidos G y F se contradigan entre sí [11].

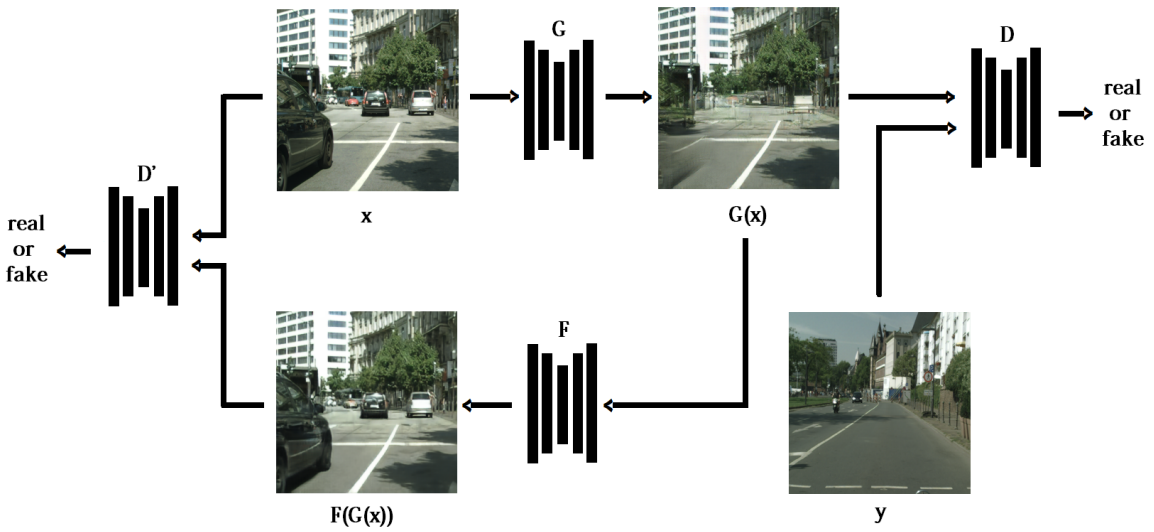


Figura 8: Representación del proceso que siguen las imágenes a través de los generadores y discriminadores. En este ejemplo se aprende a traducir imágenes de una distribución que contiene objetos dinámicos, vehículo o peatones, a una distribución sin dichos objetos y viceversa. El Generador **G** traduce la imagen dinámica real x en una estática *G(x)*, a continuación el Generador **F** traduce la imagen generada *G(x)* en *F(G(x))* obteniendo nuevamente una imagen dinámica. A su vez, el Discriminador **D** evalúa si es real o no *G(x)* comparándola con una imagen real estática y, y el Discriminador **D'** realiza una evaluación similar con *F(G(x))* y x.

Pérdida Adversa (Eqn. 1): donde G es el generador que intenta generar imágenes *G(x)* similares a las imágenes del domino **Y**, mientras que D_Y trata de distinguir entre las imágenes reales y las generadas *G(x)*. Se introduce una pérdida adversaria similar para el mapeo *F:**Y***→***X*** y su discriminador D_X. Denotamos la distribución de datos

como $y \sim p_{data(y)}$ y $x \sim p_{data(x)}$ [11].

$$\begin{aligned}\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data(y)}}[\log(D_Y(y)] \\ + \mathbb{E}_{x \sim p_{data(x)}}[\log(1 - D_Y(G(x))]\end{aligned} \tag{1}$$

Pérdida de Consistencia del Ciclo (Eqn. 2): donde G es el generador que intenta generar imágenes *G(x)* similares a las imágenes del dominio **Y** y **F** es el generador que genera imágenes recuperadas *F(G(x))*. Se introduce una pérdida de consistencia del ciclo similar para *G(F(**Y**))*$\approx$***Y***. Denotamos la distribución de datos como $y \sim p_{data(y)}$ y $x \sim p_{data(x)}$ [11].

$$\begin{aligned}\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data(x)}}[\| F(G(x)) - x \|_1] \\ + \quad \mathbb{E}_{y \sim p_{data(y)}}[\| G(F(y)) - y \|_1]\end{aligned} \tag{2}$$

Además, se ha añadido una última pérdida para preservar la composición del color entre la entrada y la salida: **Pérdida de Identidad** (Eqn. 3). Se regulariza el generador para que esté cerca de un mapeo de identidad cuando se proporcionan muestras reales del dominio objetivo como entrada al generador [11].

$$\begin{aligned}\mathcal{L}_{identity}(G, F) = \mathbb{E}_{y \sim p_{data(y)}}[\| G(y) - y \|_1] \\ + \quad \mathbb{E}_{x \sim p_{data(x)}}[\| G(x) - y \|_1]\end{aligned} \tag{3}$$

Se supone que hay una relación subyacente entre los modelos (por ejemplo: dos representaciones diferentes de la misma escena subyacente) y esa es la relación que se busca encontrar.

El modelo está formado por **dos generados** y **dos discriminadores**.

- **Generador**: Traduce imágenes de un dominio de origen **X** a un dominio de destino **Y**. Su objetivo es aumentar el índice de error del Discriminador, es decir, *engañar* a la red discriminadora produciendo imágenes sintéticas que parezcan pertenecer al conjunto de datos original.

  La red está formada por tres capas convolucionales, las dos últimas de *stride*-2, un bloque de nueve ResNets, dos deconvoluciones de *stride*-2 y una última capa convolucional. Además, se utiliza la normalización de instancias [11] (Figura 9).

- **Discriminador**: Distingue entre elementos de la distribución de datos original y los candidatos producidos por el Generador. Su objetivo es incrementar el índice de error del Generador, es decir, aprender a *diferenciar* entre imágenes reales e imágenes producidas por el Generador.

  Para las redes del discriminador utilizan 70×70 PatchGANS. Esta arquitectura puede trabajar con imágenes de tamaño arbitrario de manera totalmente convolutiva [11]. Está formada por cinco capas convolucionales, siendo las tres primeras de *stride*-2, en las cuales también se utiliza normalización de instancia (Figura 10).

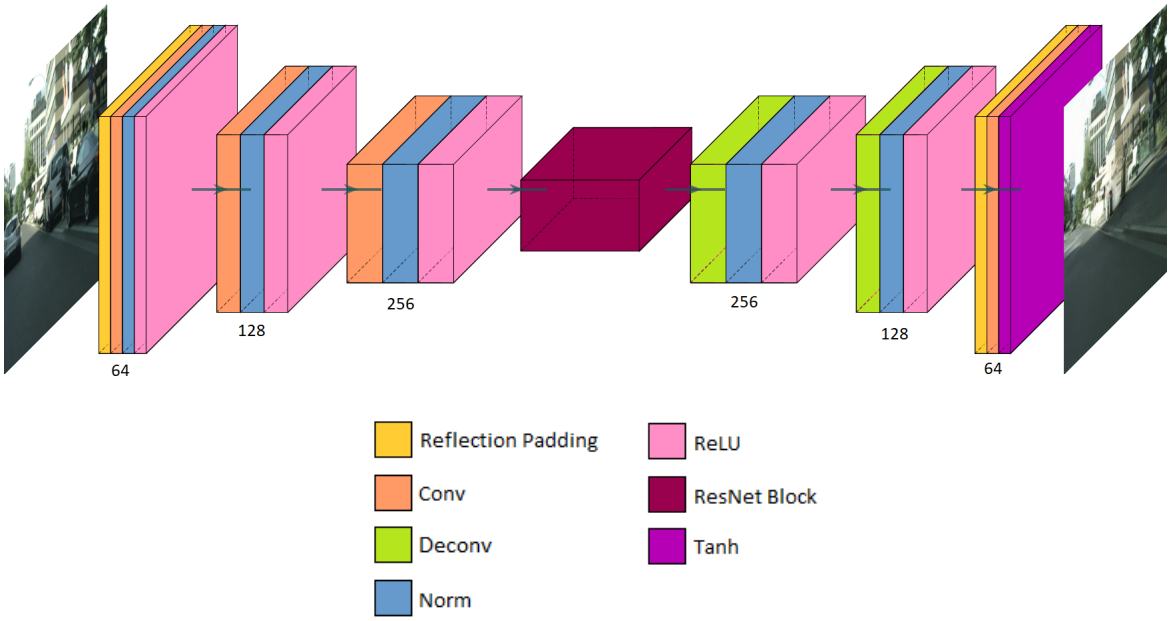


Figura 9: Arquitectura del Generador: tres capas convolucionales, un bloque de nueve ResNets, dos capas deconvolucionales y una última capa convolucional

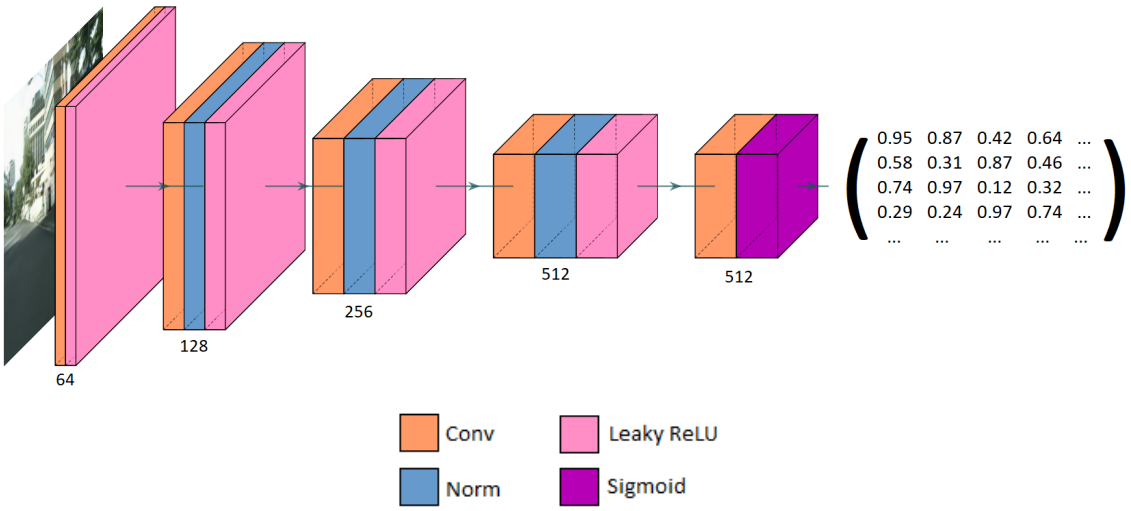


Figura 10: Arquitectura del Discriminador: cinco capas convolucionales.

3.3 Máscaras

La primera idea que se ha desarrollado, a la hora de mejorar la traducción de imágenes, ha sido la implementación de una nueva entrada a la red que ofreciera información de aquellas zonas de la imagen que tuvieran mayor relevancia en el entrenamiento. Como el principal objetivo del trabajo es la traducción de imágenes dinámicas a estáticas, consideramos que las zonas donde más debe centrarse nuestra red son aquellos píxeles de la imagen clasificados como dinámicos. Por tanto, se ha añadido un nuevo valor de entrada en la red con esta información.

Para ello, se ha llevado a cabo la generación de una máscara binaria (Figura 11) que representa el conjunto de píxeles estáticos y dinámicos que contiene la imagen [1]. Con ello, se asume que se tiene un conjunto de imágenes en el dataset que contiene la información semántica de la imagen correspondiente, es decir, la asociación de una etiqueta o categoría a cada píxel presente en dicha imagen. En caso contrario, se podría generar tal información mediante una red de segmentación semántica. El etiquetado de estas imágenes representa elementos de paisajes urbanos, por ello, se utilizará dicha información para generar la máscara correspondiente en función a estas etiquetas, estableciendo valor '255' en la máscara en aquellos píxeles de la imagen cuyas etiquetas corresponden a vehículos o peatones y valor '0' en caso contrario. Obteniendo así una clasificación de píxeles estáticos y dinámicos.

Esta máscara se utilizará como un nuevo canal de entrada (Figura 12), añadido a los canales RGB que ya teníamos, tanto para el generador como para el discriminador de nuestra CycleGAN [1].

(a) Imagen original (b) Imagen semántica (c) Máscara

Figura 11: Ejemplo de máscara. (a) muestra la imagen original utilizada en el entrenamiento, (b) es la imagen que contiene la información semántica de cada uno de los píxeles y (c) es la máscara binaria obtenida a partir de (b) que se genera al inicio del entrenamiento. La máscara (c) tiene diferente tamaño ya que se le ha modificado la proporción para el proceso de entrenamiento.

En el caso de la traducción de imágenes estáticas a dinámicas, se va a añadir también una máscara binaria con la misma finalidad. En esta traducción, la máscara de la propia imagen no nos sirve de utilidad, ya que se necesita una máscara que contenga objetos dinámicos, de esta manera se ayudará a la red a fijarse en aquellas zonas dónde generar objetos. Por tanto, en un principio, se va a añadir como entrada la misma máscara que la utilizada para la traducción de imágenes dinámicas.

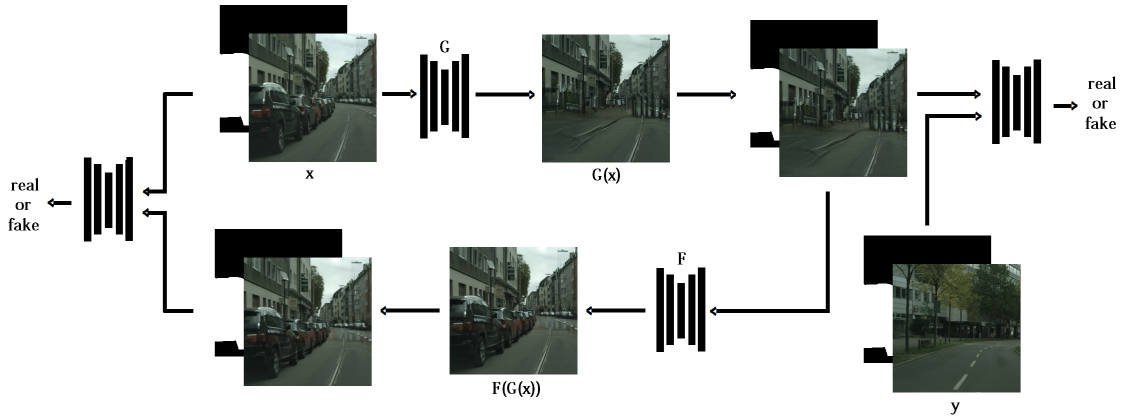


Figura 12: Representación del proceso que siguen las imágenes a través de los generadores y discriminadores. Se incluye la máscara como nueva entrada en los generadores.

3.3.1 Aplicación de pesos

Una implementación ligada a la generación de máscaras es la posibilidad de establecer un peso de reescalado manual a la pérdida de cada elemento del lote para cada una de las diferentes funciones de pérdida utilizadas en el modelo.

La finalidad de utilizar estos pesos es compensar la proporción de píxeles dinámicos a la hora de realizar el entrenamiento del modelo, ya que es menor en comparación con el porcentaje de píxeles estáticos. En nuestro caso, queremos que el modelo se centre mayormente en aquellos píxeles que son dinámicos, ya que serían las zonas de la imagen donde queremos que tenga una mayor importancia el error obtenido.

A la hora de generar el kernel de los pesos únicamente comprobamos el porcentaje de píxeles dinámicos y estáticos que hay, estableciendo el valor resultante del mayor porcentaje en aquellos píxeles que son dinámicos y, el valor resultante del menor porcentaje en los estáticos (Eqn. 4). De esta manera, intentamos que la red se fije más en los píxeles dinámicos sin perder tampoco de vista los píxeles estáticos.

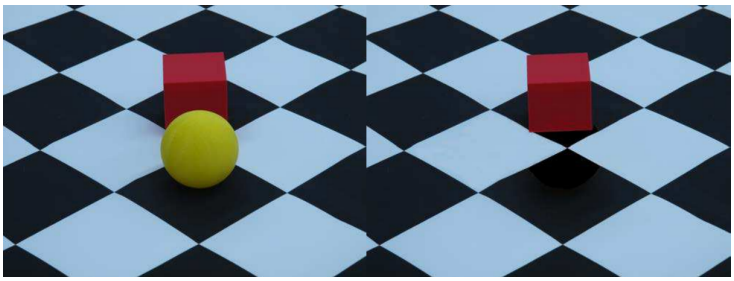
$$W = \begin{cases} Si \quad P_D > P_E \begin{cases} W[p_d] = P_D \\ W[p_e] = P_E \end{cases} \\ Si \quad P_E > P_D \begin{cases} W[p_d] = P_E \\ W[p_e] = P_D \end{cases} \end{cases} \tag{4}$$

Siendo p_d el conjunto de píxeles dinámicos y p_e el conjunto de píxeles estáticos de la imagen, los cuales son indicados por la máscara. Por último, P_D es el porcentaje de píxeles dinámicos y P_E el porcentaje de píxeles estáticos de la imagen.

3.4 Detección de manipulación de imágenes.

Con la evolución de las técnicas de edición, la distinción entre imágenes auténticas e imágenes manipuladas se ha vuelto cada vez más difícil (Figura 13). Hay trabajos que han intentado detectar estas imágenes como el trabajo de Zhou *et al.* [23]. Por ello, se ha seguido su marco de aprendizaje para estudiar aquellas características cuya implementación podrían ser de utilidad en el **Discriminador**.

(a) La parte izquierda de la imagen correspondería a la foto original y la derecha a la foto manipulada



(b) La imagen de la izquierda presenta la misma escena que en la derecha pero habiéndole añadido un objeto más.

Figura 13: Ejemplo de dos imágenes manipulada

Mientras que los modelos actuales de aprendizaje profundo representan de manera adecuada las características jerárquicas del contenido de una imagen RGB, ningún trabajo previo había investigado anteriormente el aprendizaje de las distribuciones de ruido en la detección hasta el trabajo de Zhou *et al.* [23].

El ruido es la variación aleatoria del brillo o el color en las imágenes digitales producido por el dispositivo de entrada. La visualización de esta alteración genera el conocido "*grano*", que son píxeles que no se corresponden con la luminancia y tonalidad real de la imagen (Figura 14).

(a) Imagen sin ruido (b) Imagen con ruido

Figura 14: Comparación entre una imagen con ruido y sin ruido

El objetivo de nuestro trabajo es eliminar aquellas regiones de la imagen que contienen elementos dinámicos. La intuición detrás de esto es que al quitar elementos de una imagen y rellenar sus huecos, es improbable que las características de ruido entre la imagen original y la modificada coincidan.

Recientemente, los métodos basados en características de ruido local, como el *Modelo Rico en Estegoanálisis* (conocido principalmente como SRM, del inglés *Steganalysis Rich Model*) utilizado en la detección de manipulación [24], han mostrado un rendimiento prometedor. Estos métodos extraen características de ruido local de los píxeles adyacentes, capturando la inconsistencia entre las regiones alteradas y las auténticas.

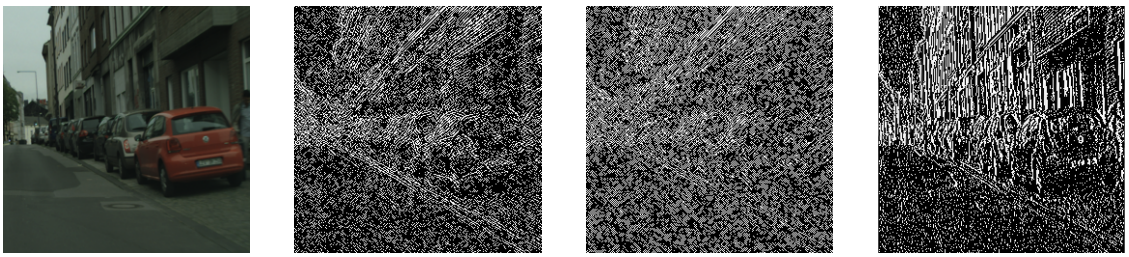
El ruido se modela por el residuo entre el valor de un píxel y la estimación del valor de ese píxel producido al interpolar sólo los valores de los píxeles vecinos.

Siguiendo el trabajo original de Fridrich *et al.* [24], en el que se utilizan núcleos

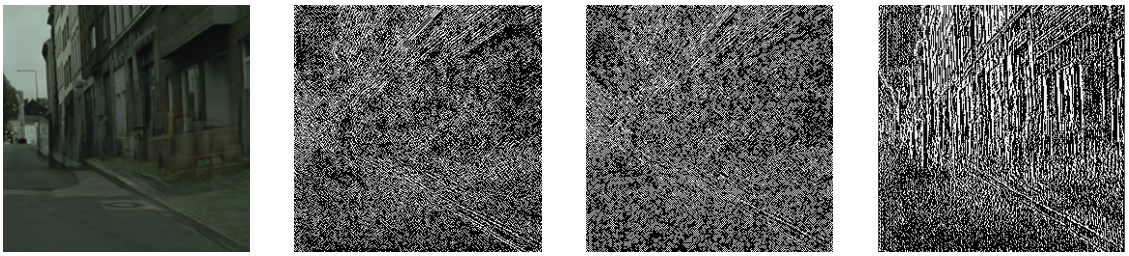
de filtro SRM para extraer las características de ruido local de las imágenes (Figura 16), Zhou *et al.* [23] publica que usando únicamente 3 de esos núcleos se obtiene un rendimiento decente y, aplicando todos los kernels no se obtiene una ganancia de rendimiento significativa (Figura 15).

$$\frac{1}{4}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 2 & -4 & 2 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \frac{1}{12}\begin{bmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & 1 \end{bmatrix} \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figura 15: Los tres núcleos SRM utilizados para extraer las características de ruido. El primer núcleo ha sido utilizado anteriormente en esteganografía [24] [25], el segundo núcleo es resultado de la optimización de los coeficientes de un núcleo circularmente simétrico 5x5 utilizando el algoritmo de Nelder-Mead [24] y, por último, el tercer núcleo se calcula como un filtro lineal de paso alto de píxeles vecinos con los coeficientes correspondientes [24].



(a) real.png (b) SRM0.png (c) SRM1.png (d) SRM2.png



(e) fake.png (f) SRM0.png (g) SRM1.png (h) SRM2.png

Figura 16: Ejemplo de las imágenes utilizadas como nuevos canales de entrada en el discriminador, resultado de aplicar cada filtro SRM sobre la imagen RGB. El resultado obtenido por cada filtro es la visualización de las características de ruido de cada imagen. La intuición detrás del estudio realizado plantea la improbabilidad de que las características de ruido de una imagen real y una modificada coincidan.

En el modelo CycleGAN es tarea del discriminador clasificar cada parche de cada imagen generada como real o falso (modificado). Por tanto, se utilizan las características de ruido local calculadas como un nuevo canal de entrada en el discriminador

(Figura 17), incrementando la posibilidad de aprender, más fácilmente, a distinguir las imágenes reales de las falsas.

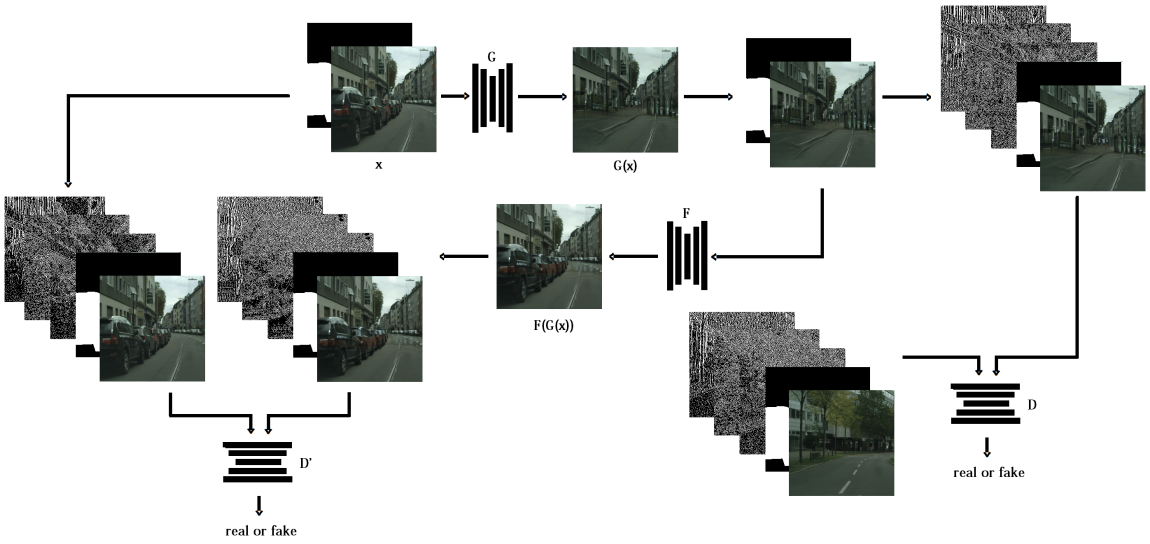


Figura 17: Representación del proceso que siguen las imágenes a través de los generadores y discriminadores. Se incluyen las imágenes obtenidas de aplicar el filtro SRM como nueva entrada en los discriminadores.

3.5 Detector FAST

A la vista de los resultados obtenidos, se aprecia que las zonas reconstruidas detrás de los objetos dinámicos a veces carecen de altos gradientes. Por ello, se quiere forzar a que las imágenes reconstruidas tenga una misma respuesta a filtros de detección de esquinas que las imágenes originales, obteniendo de esta manera resultados más nítidos.

Entre todos los algoritmos de detección de esquinas se ha elegido FAST (del inglés *Features from Accelerated Segment Test*) [26], por su buena respuesta y por su sencillez a la hora de aproximarlo de manera convolucional. Además, es muy adecuado para aplicaciones de procesamiento en tiempo real debido a su rendimiento a alta velocidad.

La detección de esquinas se va a implementar calculando la probabilidad de esquina para la imagen original como para la imagen recuperada y así, comprobar la desviación de la predicción obtenida mediante una **pérdida BCE** (Eqn. 7). Con esto, se espera reducir las zonas borrosas de las imágenes obteniendo una mejor definición.

El detector de esquinas **FAST** utiliza un círculo de 16 píxeles (un *círculo de Bresenham de radio 3*) para clasificar si un punto $\boldsymbol{p}$ es realmente una esquina. Si un

conjunto de $\boldsymbol{N}$ píxeles contiguos (denotado cada píxel individualmente por $I_{p\to x}$) son más brillantes que la intensidad del píxel $\boldsymbol{p}$ (denotado por I_p) más un valor umbral $\boldsymbol{t}$ o todos más oscuros que la intensidad del píxel $\boldsymbol{p}$ menos el valor umbral $\boldsymbol{t}$, entonces $\boldsymbol{p}$ se clasifica como esquina.

Cada píxel $\boldsymbol{x}$ del *círculo de Bresenham* tiene un estado establecido (Eqn. 5).

$$S_{p\to x} = \begin{cases} Darker & si \quad I_{p\to x} \leq I_p - t \\ Similar & si \quad I_p - t < I_{p\to x} < I_p + t \\ Brighter & si \quad I_p + t \leq I_{p\to x} \end{cases} \tag{5}$$

Para ello, se ha aproximado la detección de esquinas **FAST**: implementando una capa convolucional, y por tanto, diferenciable, de 16 kernels con pesos fijos (Figura 18) [2].

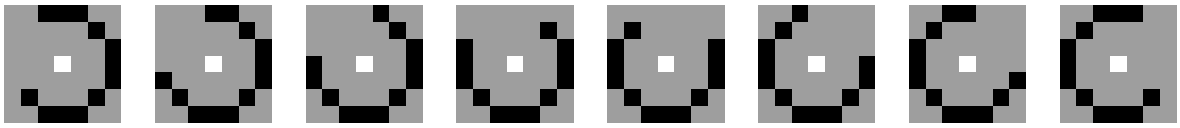


Figura 18: Ejemplo ilustrativo de algunos de los kernels utilizados en la detección de esquinas. El píxel central blanco tiene valor 1, los píxeles grises 0 y los píxeles negros tienen valor -1/12

A la hora de realizar la convolución es necesario tener la intensidad de cada píxel. Para ello, se va a traducir cada imagen a escala de grises obteniendo así dicho valor.

Una vez que se tienen los valores de la intensidad se realiza la convolución sobre la imagen obteniendo las diferentes respuestas a los filtros detectores de esquinas [2].

A continuación, elevamos cada valor al cuadrado, ya que nos interesan tanto los valores positivos como los negativos y, para cada píxel, nos quedamos únicamente con el valor del núcleo que ha obtenido mayor puntuación [2].

Finalmente, añadimos el valor umbral $\boldsymbol{t}$, del que hablábamos anteriormente, y aplicamos una convolución *sigmoide* para obtener un resultado comprendido entre [0, 1]. De esta forma obtenemos la probabilidad de que un píxel sea una esquina [2].

Las características de las imágenes reales (*ground truth features*) se generan de la misma forma. Pero, sobre estos resultados obtenidos, se añade una última transformación convirtiendo los valores a un resultado binario (Eqn. 6).

$$p = \begin{cases} 0 & si \quad p < 0{,}5 \\ 1 & si \quad p \geq 0{,}5 \end{cases} \tag{6}$$

Calculadas las probabilidades de esquina (Figura 19), tanto para la imagen original como para la generada por el segundo generador (la imagen recuperada), se mide a continuación la desviación de la predicción obtenida.

Para ello, se utiliza la función de pérdida **BCE** (Eqn. 7) que mide la *entropía cruzada binaria* entre el objetivo y la salida, es decir, la calidad del modelo.

$$\mathcal{L}(x, y) = mean(L)$$
$$L = \{l_1, ..., l_N\}^T, \quad l_n = -w_n[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)] \tag{7}$$

donde N es el tamaño del lote, x es el conjunto a evaluar, y el conjunto objetivo y w un peso dado a la pérdida de cada elemento del lote. Esta función es utilizada para medir el error de una reconstrucción.

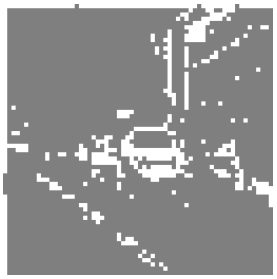
3.5.1 Aplicación pesos

A la hora de calcular la función de pérdida **BCE** se establece un peso de reescalado manual, similar a lo que se ha realizado con los pesos de la máscara (Eqn. 4). La finalidad de utilizar estos pesos es compensar la proporción de píxeles que son esquina, a la hora de realizar el entrenamiento del modelo, ya que es menor en comparación con el porcentaje de píxeles que no son esquina. En este caso, queremos que el error producido en aquellos píxeles donde realmente había una esquina sea mas significativo.

(a) Imagen dinámica real

(b) Detección de esquinas de (a)

(c) Imagen dinámica recuperada



(d) Detección de esquinas de (c)

(e) Imagen estática real

(f) Detección de esquinas de (e)

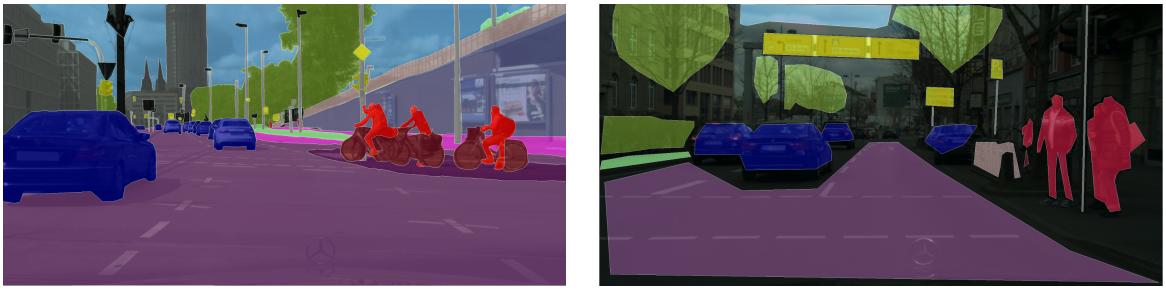
(g) Imagen estática recuperada

(h) Detección de esquinas de (g)

Figura 19: Las imágenes mostradas a la derecha son un ejemplo visual de los resultados obtenidos por la detección de esquinas, siendo los píxeles blancos las esquinas detectadas. Como se observa, el conjunto de esquinas detectadas en las imágenes reales muestra zonas más definidas y claras, en cambio los píxeles en las imágenes recuperadas son más dispersos. Además, de esta forma también es más fácil detectar errores como artefactos producidos por la red convolucional (*e.g.* en la imagen (c) en la zona superior derecha).

4 Conjunto de datos utilizado

A la hora de realizar el entrenamiento y la evaluación de nuestro modelo se ha utilizado el dataset **CityScapes** [27]. Este conjunto de datos a gran escala contiene un conjunto diverso de secuencias de vídeo estéreo grabadas en escenas urbanas de 50 ciudades diferentes, con anotaciones a nivel de píxeles de alta calidad de 5000 fotogramas, además de un conjunto mayor de 20000 fotogramas con anotaciones débiles (Figura 20).



(a) Anotación de alta calidad (b) Anotación de baja calidad

Figura 20: La diferencia entre estos conjuntos es que en (b) se marcan polígonos que cubren objetos individuales, mientras que en (a) se ofrecen anotaciones de alta calidad. Los colores superpuestos codifican las clases semánticas. [27]

Los datos utilizados corresponden al conjunto de 5000 fotogramas de alta calidad. Este conjunto está formado por 2975 imágenes para el entrenamiento, 500 para la validación del modelo y 1525 imágenes como conjunto de pruebas.

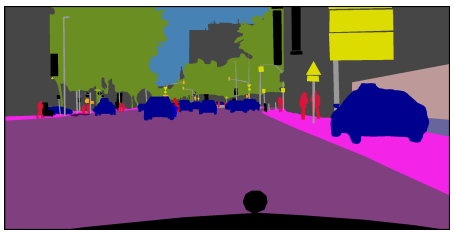
El dataset contiene cuatro tipos de imágenes por cada fotograma (Figura 21).

- **leftImg8bit.png**: Imagen RGB.
- **color.png**: Imagen donde cada píxel está codificado por una clase color.
- **labelIds.png**: Imagen donde cada píxel está codificado por un entero que indica la clase a la que pertenece. En total hay 34 clases diferentes (persona, cielo, coche, señal, *etc.*).
- **instanceIds.png**: Imagen donde cada píxel está codificado por un ID de instancia. Este identificador indica si el píxel pertenece a un vehículo (sin tener en cuenta matrículas) o a una persona.

Para cada imagen del conjunto, los objetos del primer plano etiquetados nunca tienen huecos, es decir, si hay algún fondo visible 'a través' de algún objeto del primer

plano, se considera parte del primer plano. Ejemplo: hojas de árboles frente a una casa o el cielo (todo árbol), ventanas transparentes de un coche (todo coche).

(a) leftImg8bit.png



(b) color.png

(c) instanceIds.png

(d) labelIds.png

Figura 21: Ejemplo de una imagen del conjunto de datos de CityScapes.

Sobre este conjunto de datos se ha realizado una separación entre imágenes dinámicas y estáticas. Para ello, se ha comprobado el porcentaje de píxeles dinámicos que contiene cada imagen quedándonos con aquellas imágenes que no tienen más del 1.5 % de píxeles dinámicos como imágenes estáticas y, aquellas que tienen un mínimo del 20 % como dinámicas. Obtenemos así 1185 imágenes dinámicas y 1049 estáticas para el entrenamiento y, 199 dinámicas y 148 estáticas para la validación de los modelos.

Además, antes de realizar este proceso, se ha modificado el tamaño original de cada imagen (2048x1024) para obtener unas dimensiones de 1000 píxeles de ancho por 800 píxeles de alto. De esta manera intentamos centrar más la imagen en la carretera y eliminar la parte inferior del capó del coche que aparece en la mayoría de las imágenes.

Por último, en el modelo solo se tienen en cuenta las imágenes RGB y aquellas que contienen el identificador de clase (*labelIds.png*). Estas últimas serán utilizadas para generar posteriormente la máscara binaria (Figura 22).

(a) color.png

(b) labelIds.png

Figura 22: Ejemplo del conjunto de datos utilizado en el entrenamiento del modelo.

5 Evaluación

A la hora de realizar la evaluación de cada una de las implementaciones desarrolladas se procede inicialmente al entrenamiento del modelo, añadiéndole de manera incremental cada una de las implementaciones explicadas anteriormente. Al modelo CyleGAN [11] sin ninguna de las implementaciones se le denominará *Modelo Original*. El modelo con máscaras será *Modelo Mask* (Sec. 3.3), *Modelo Weight* si además tiene pesos (Sec. 3.3.1). A este último modelo añadiéndole la detección de manipulación de imágenes se le denominará *Modelo Noise* (Sec. 3.4) y, por último, *Modelo FAST* si también se añade el detector FAST (Sec. 3.5). Finalmente, como se explicará más adelante, se ha realizado un nuevo modelo denominado *Modelo FASTwithoutNoise*, el cual reúne todas las implementaciones desarrolladas a excepción de la detección de manipulación.

Una vez finalizado el entrenamiento, para cada uno de los modelos, se realiza la evaluación tanto cualitativa como cuantitativa. Para ello, se ha obtenido el resultado de la traducción del conjunto de validación (148 imágenes dinámicas y 148 estáticas) con cada modelo entrenado. A continuación, se muestran los resultados obtenidos.

5.1 Evaluación Cualitativa

5.1.1 Modelo Original

Los resultados mostrados en la Figura 23 corresponden a la evaluación del *Modelo Original* de CycleGAN [11] entrenado y evaluado en nuestro dataset de entrenamiento y validación respectivamente. Se puede observar en la Figura 23b que el modelo intenta mimetizar el coche blanco con el edificio del fondo y con la carretera, pero el coche sigue siendo fácilmente distinguible. Por otro lado, se aprecia que la estructura estática de la imagen se mantiene estable, a falta de una ligera pérdida de definición en las zonas de alto gradiente de la imagen, como puede observarse en las sombras de la carretera en el primer plano de la imagen. Por otro lado, si se ven las Figuras 23d y 23e, se puede observar que el modelo es capaz de introducir artefactos en la imagen estática similares a una fila de coches. A pesar de que estos artefactos no tienen la textura correcta y se confunden unos coches con otros, la forma de los artefactos es similar a la que querríamos conseguir. Siendo estos resultados el punto de partida de este trabajo de fin de grado, se van a añadir diferentes mejoras para obtener imágenes más dinámicas / estáticas y sobre todo más realistas.

(a) real dinámica (b) falsa estática (c) recuperada dinámica

(d) real estática (e) falsa dinámica (f) recuperada estática

Figura 23: Resultados del Modelo Original

5.1.2 Modelo Mask

El *Modelo Mask*, el cual introduce la máscara binaria de los objetos dinámicos en el entrenamiento, presenta una mejora importante en la traducción de imágenes (Figura 24). Prácticamente elimina cualquier aparición de objetos dinámicos rellenando las zonas implicadas, obteniendo así un resultado muy realista. Más concretamente, en la Figuras 24a y 24b se observa que las instancias dinámicas son casi completamente eliminadas y, en el peor de los casos, son perfectamente mimetizadas con el fondo estático, como es el caso de los dos focos traseros de la furgoneta negra. En cuanto a la pérdida de definición que apreciábamos en el modelo original, no es tan notable en el caso actual. Además, en la traducción de imágenes estáticas a dinámicas, se generan elementos distinguibles como ventanas, ruedas, retrovisores, *etc.* Aunque los tipos de objetos son fácilmente distinguibles por su forma y algunos elementos, su textura aún deja mucho que desear.

5.1.3 Modelo Weight

Las imágenes obtenidas como resultado en el *Modelo Weight* son muy similares a las del *Modelo Mask* (Figura 25). Los pesos que se introducen con esta implementación en la función de pérdida llevan a la red a compensar el bajo número de píxeles dinámicos en el dataset frente a los estáticos. Lo importante de esta implementación es observar que en los resultados las partes estáticas de las imágenes se mantienen definidas.

(a) real dinámica (b) falsa estática (c) recuperada dinámica (d) máscara para la real dinámica

(e) real estática (f) falsa dinámica (g) recuperada estática (h) máscara para la real estática

Figura 24: Resultados del Modelo con máscaras.

Observando todo el conjunto de resultados se puede apreciar que las zonas originalmente estáticas de las imágenes aparecen igualmente definidas, y las zonas reconstruidas mantienen un grado de realismo ligeramente superior al de la implementación anterior.

(a) real dinámica (b) falsa estática (c) recuperada dinámica (d) máscara para la real dinámica

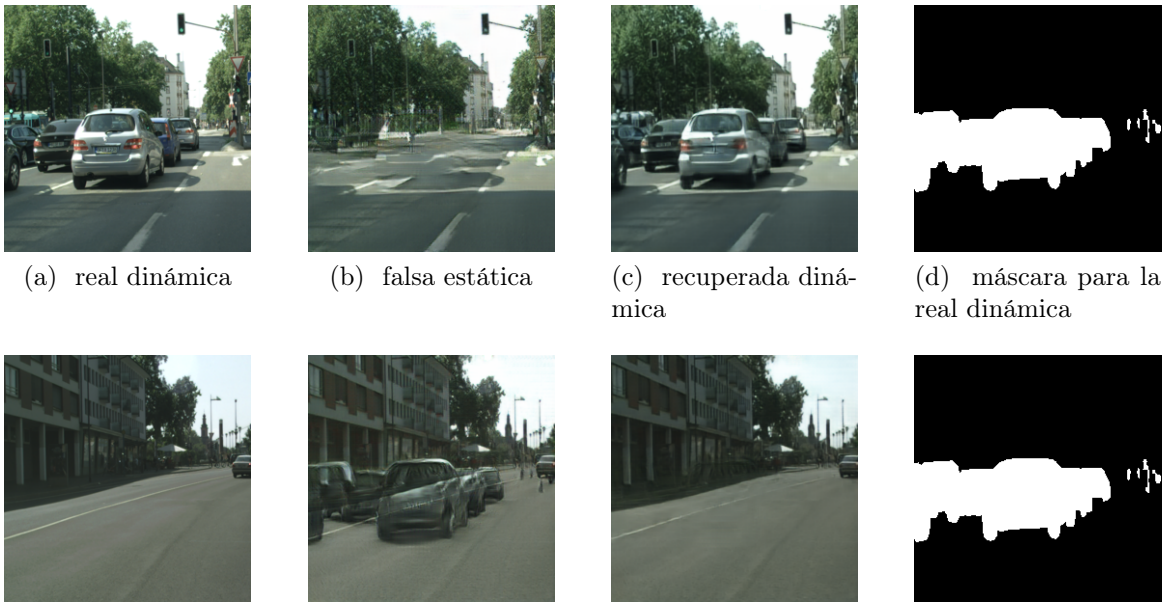
(e) real estática (f) falsa dinámica (g) recuperada estática (h) máscara para la real estática

Figura 25: Resultados del Modelo con máscaras y pesos.

5.1.4 Modelo Noise

Los resultados presentados en la Figura 26 corresponden a la evaluación del *Modelo Noise*. En ellos, se puede observar que la traducción de imágenes estáticas a dinámicas empeora considerablemente en comparación con el modelo anterior. En general, como se ve en la Figura 26b, las instancias que se quieren eliminar se muestran ligeramente transparentes apareciendo como fantasmas, lo cual no ocurría con las implementaciones anteriores.

Por otro lado, la traducción a imágenes dinámicas también se ve afectada negativamente por esta implementación. Observando la Figura 26f se puede contemplar que no se elimina completamente la estructura estática de la imagen, aún podrían apreciarse la carretera y los edificios detrás de los vehículos. En este ejemplo, también se puede contemplar cómo la máscara presenta pequeñas zonas dinámicas difíciles de asociar con algún objeto dinámico, provocando irregularidades en la imagen generada.



(a) real dinámica (b) falsa estática (c) recuperada dinámica (d) máscara para la real dinámica

(e) real estática (f) falsa dinámica (g) recuperada estática (h) máscara para la real estática

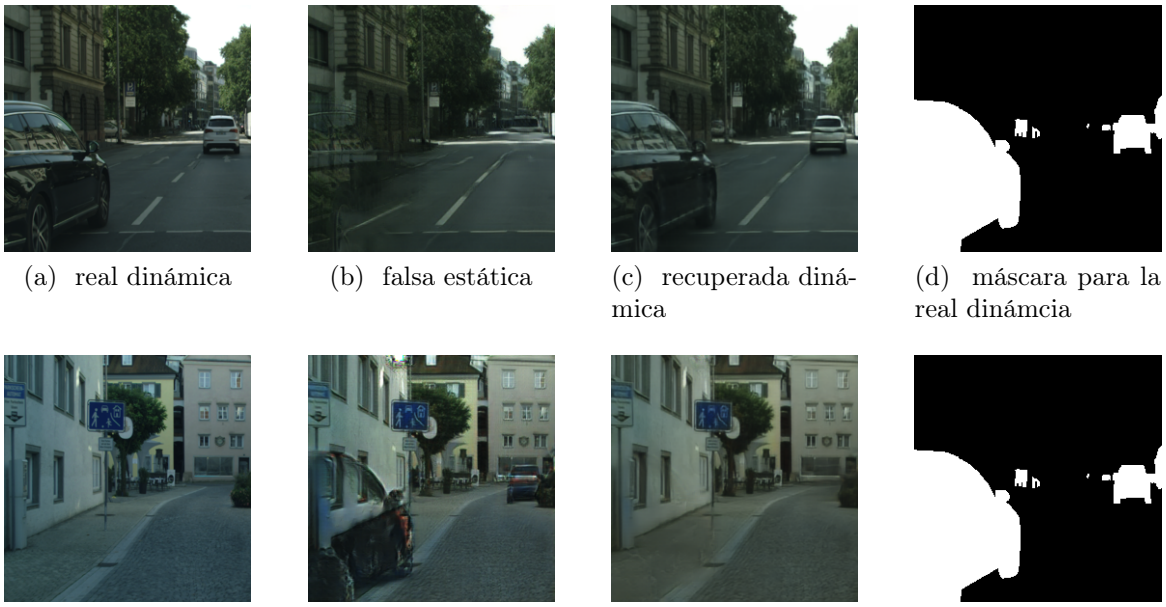
Figura 26: Resultados del Modelo con máscaras, pesos y detección de ruido.

5.1.5 Modelo FAST

Los resultados del *Modelo FAST* muestran una mejora visual respecto al *Modelo Noise* presentado anteriormente, aunque se sigue observando que la calidad de las imágenes generadas es inferior al resto de modelos entrenados (Figura 27). En la Figura 27b

todavía puede apreciarse la superficie del vehículo negro junto a los reflejos sobre este. Por otro lado, el vehículo blanco se mimetiza correctamente con el fondo de la imagen, a excepción de una pérdida de definición en la zona generada.

En el caso de la traducción de imágenes estáticas a dinámicas se observa la misma evolución. Las instancias dinámicas aumentan su consistencia pareciendo más realista, aún así se sigue observando el fondo que habría detrás de ellas (Figura 27f).



(a) real dinámica (b) falsa estática (c) recuperada dinámica (d) máscara para la real dinámcia

(e) real estática (f) falsa dinámica (g) recuperada estática (h) máscara para la real estática

Figura 27: Resultados del Modelo con máscaras, pesos, detección de ruido y detector FAST.

5.1.6 Modelo FAST without Noise

A la vista de que los resultados visuales empeoraban con la implementación de la detección de manipulación (Sec.3.4), se ha decidido realizar una evaluación sin esta característica. Para ello, se ha entrenado un nuevo modelo, al cual denominaremos *Modelo FASTwithoutNoise*, y se ha traducido el conjunto de imágenes de validación.

Como se esperaba, los resultados visuales obtenidos por el nuevo modelo mejoran considerablemente. En la Figura 28b se observa que la fila de vehículos se ha mimetizado completamente con el fondo de la imagen. Aún así, mirando en detalle el conjunto de la imagen aún pueden apreciarse que queda trabajo por hacer, esto puede apreciarse en la falta de definición de la acera generada o en el fondo estático de la imagen, el

cual no debería presentar ninguna modificación.

Finalmente, en la traducción de imágenes estáticas a dinámicas recuperamos la consistencia de las instancias, pudiéndose diferenciar vehículos individuales y diferentes partes de estos, como ruedas, ventanillas y faros.

(a) real dinámica (b) falsa estática (c) recuperada dinámica

(d) real estática (e) falsa dinámica (f) recuperada estática

Figura 28: Resultados del Modelo FASTwithoutNoise

5.1.7 Modelo Empty Cities

A la hora de realizar la evaluación también se tienen en cuenta los resultados obtenidos con el *Modelo Empty Cities* [1], ya que este trabajo continúa su línea de investigación. **Emtpy Cities** solo realiza traducción de imágenes dinámicas a estáticas. Por lo tanto, solo se podrá considerar la evaluación de sus resultados para este caso.

Además, este modelo ha sido entrenado con un conjunto de imágenes sintéticas (Dataset *CARLA* [28]), en escala de grises. Por ello, no se obtendrán resultados tan buenos para las imágenes reales RGB utilizadas en nuestro modelo, como para este conjunto de imágenes. En las Figuras 29 y 30, se muestran algunos de los resultados obtenidos para el Dataset *CARLA* y para el dataset *City Scapes* con este modelo.

(a) sintética dinámica (b) falsa estática (a) sintética dinámica (b) falsa estática

Figura 29: Resultados del Modelo Empty Cities con el dataset CARLA.

(a) real dinámica (b) falsa estática (a) real dinámica (b) falsa estática

Figura 30: Resultados del Modelo Empty Cities con el dataset City Scapes.

Como se observa, se obtienen buenos resultados para un conjunto de imágenes sintéticas a traducir pero, en el caso de querer traducir imágenes reales los resultados obtenidos se muestran borrosos. La red no consigue generar zonas estáticas definidas para una imagen real, además, las imágenes obtenidas se muestran en escala de grises.

Finalmente, en las Figuras 31, 32, 33 y 34 se van a mostrar los resultados obtenidos de la traducción de una imagen dinámica a estática y viceversa, para cada modelo,

con la misma imagen de entrada. Se aprecia que el cambio más significativo en los resultados tiene lugar cuando se pasa de utilizar el modelo original a utilizar el de las máscaras y pesos. En el caso de traducción de una imagen dinámica a estática, los objetos dinámicos desaparecen visualmente casi por completo o se mimetizan con el entorno estático. Por otro lado, en el caso de traducción de una imagen estática a dinámica, los resultados más relevantes son los obtenidos por el modelo que utiliza máscaras y el que añade pesos, siendo estos muy similares. En ambos modelos, los objetos dinámicos generados son distinguibles individualmente además de poder apreciarse elementos como retrovisores, matrículas, *etc.* Aunque la mejora es menos incremental o menos notoria, el modelo que utiliza FAST sin utilizar las características de ruido también genera imágenes realistas en ambos tipos de traducción. Esto se aprecia sobre todo en la última fila de la Figura 32b, en la cual es difícil reconocer que se ha eliminado a los dos peatones. En el Anexo A se muestran más resultados obtenidos por cada modelo.

Traducción de una imagen **dinámica** a **estática** con cada modelo

Figura 31: Resultado obtenido de la traducción de una imagen dinámica a una estáticas en los diferentes modelos entrenados.

Figura 32: Resultado obtenido de la traducción de una imagen dinámica a una estáticas en los diferentes modelos entrenados.

Traducción de una imagen **estática** a **dinámica** con cada modelo

Figura 33: Resultado obtenido de la traducción de una imagen estática a una dinámica en los diferentes modelos entrenados.

Figura 34: Resultado obtenido de la traducción de una imagen dinámica a una estáticas en los diferentes modelos entrenados.

5.2 Evaluación Semántica

Debido a que es muy difícil realizar una evaluación numérica de los modelos generados, al no disponer del *ground-truth* del conjunto de resultados de nuestras imágenes, se ha optado por la utilización de una red de segmentación semántica para comprobar si efectivamente los píxeles dinámicos han sido convertidos en píxeles estáticos, y viceversa.

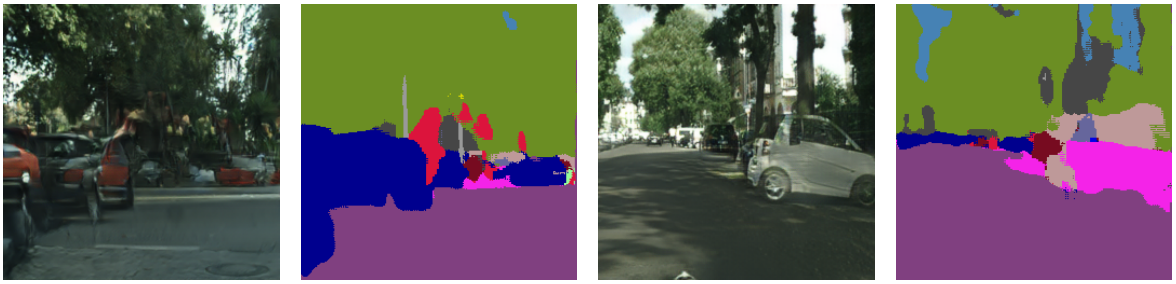
Los enfoques tradicionales basados en visión tenían inicialmente como objetivo desarrollar técnicas especificas para detectar elementos de tráfico como el pavimento de las carreteras, los automóviles, las señales o los peatones [29] [6]. Sin embargo, los avances en el aprendizaje profundo han permitido unificar estos problemas de clasificación en una tarea: la **segmentación semántica**. El objetivo de la **segmentación semántica** es asociar una etiqueta o categoría a cada píxel presente en una imagen. Se utiliza para reconocer conjuntos de píxeles que conforman distintas categorías.

ERFNet (del inglés *Efficient Residual Factorized ConvNet for Real-time Semantic Segmentation*) [6] es una arquitectura que logra una precisa y rápida segmentación semántica por píxeles, lo que la hace adecuada para innumerables aplicaciones, como la compresión de escenas en vehículos, que requieren tanto robustez como operatividad en tiempo real.

Los resultados obtenidos de la traducción del conjunto de validación han sido procesados por la red ERFNet para evaluar la cantidad de píxeles dinámicos que contenía cada imagen. Algunos de los resultados obtenidos son los siguientes.

5.2.1 Modelo Original

Los resultados mostrados en la Figura 35 corresponden a la evaluación semántica de las imágenes falsas obtenidas con el *Modelo Original*. La Figura 35b es la segmentación semántica obtenida con la red ERFNet de la imagen falsa de la Figura 35a. Se observan píxeles que corresponden a instancias dinámicas como coches y personas, aunque en ninguno de estos casos se presenta un contorno definido que permita distinguir los objetos individualmente de manera visual. Por otro lado, en la Figura 35d, que corresponde a la segmentación semántica de la Figura 35c, se puede apreciar que el coche del primer plano sí que es interpretado como “acera”, pero se sigue detectando la presencia de elementos dinámicos en el fondo de la imagen.



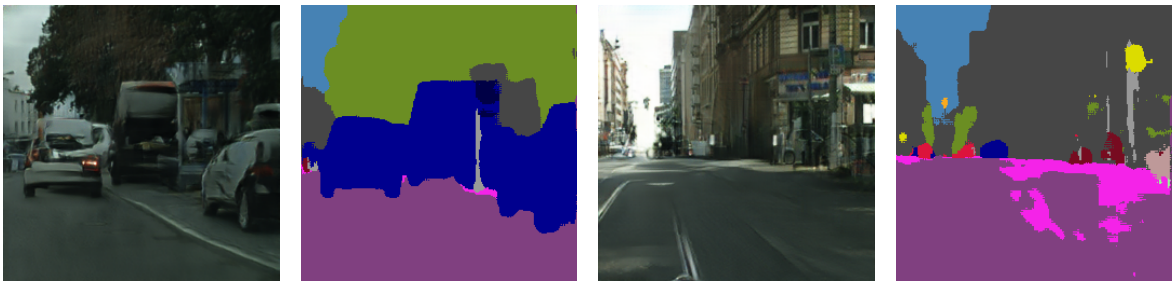
(a) falsa dinámica (b) segmentación falsa dinámica (c) falsa estática (d) segmentación falsa estática

Figura 35: Resultados ERFnet del modelo original obtenidos para las imágenes mostradas en la Figura 23. Los colores se encuentran definidos en la Tabla 1

.

5.2.2 Modelo Mask

Aunque se siguen detectando pequeños porcentajes de píxeles dinámicos en las imágenes estáticas falsas presentadas por el *Modelo Mask* (Figura 36d), los resultados de las dinámicas falsas mejoran considerablemente mostrando siluetas definidas de los objetos dinámicos, pudiendo apreciarse en la Figura 36b. Se aprecia que la segmentación de los vehículos de la Figura 36b es muy parecida a la del ground truth.



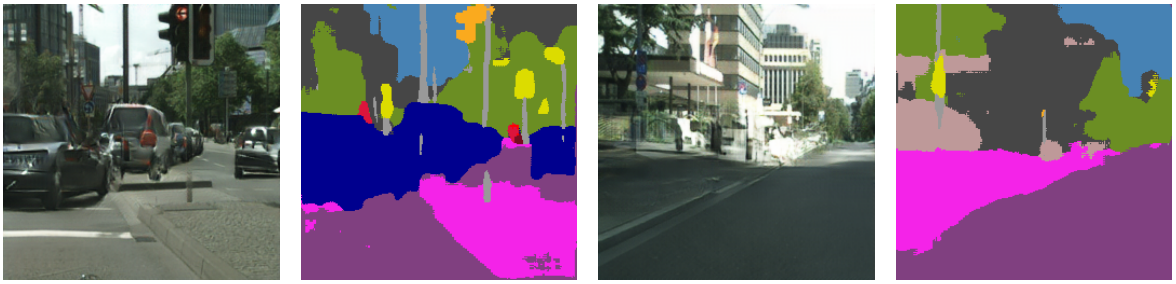
(a) falsa dinámica (b) segmentación fake dinámica (c) falsa estática (d) segmentación falsa estática

Figura 36: Resultados ERFnet del modelo con máscaras obtenidos para las imágenes mostradas en la Figura 24. Los colores se encuentran definidos en la Tabla 1

.

5.2.3 Modelo Weight

La segmentación semántica obtenida en los resultados del *Modelo Weight* es muy similar a los resultados semánticos presentados por parte del *Modelo Mask* (Figura 37). Como se ha explicado anteriormente, en este modelo lo importante es mantener definidas las zonas estáticas de las imágenes, esto puede apreciarse en las Figuras 37b y 37d. La segmentación de estas zonas es de alta precisión.



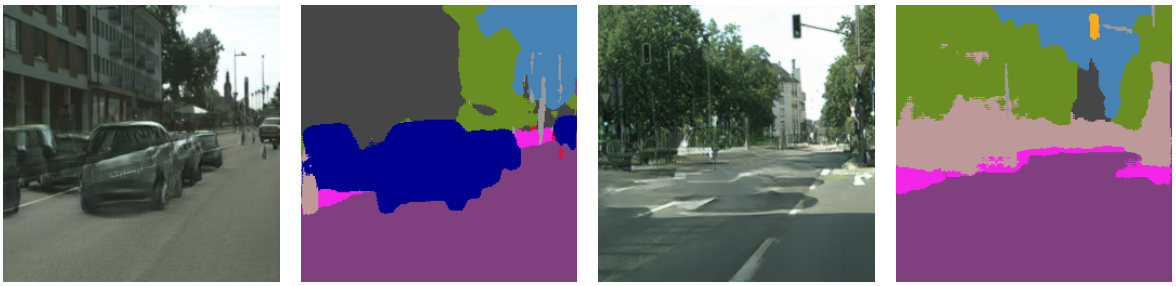
(a) falsa dinámica (b) segmentación falsa dinámica (c) falsa estática (d) segmentación falsa estática

Figura 37: Resultados ERFnet del modelo con máscaras y pesos obtenidos para las imágenes mostradas en la Figura 25. Los colores se encuentran definidos en la Tabla 1

.

5.2.4 Modelo Noise

A pesar de que visualmente las imágenes presentan una degradación en la calidad obtenida, como veíamos en la Sección 5.1.4, la red ERFNet continúa detectando correctamente las instancias dinámicas generadas, así como las estáticas reconstruidas. Esto puede observarse en la Figura 38b, donde a pesar de que los vehículos muestran transparencias su información semántica sigue evaluándose como dinámica. Por el contrario, observando la evaluación semántica del conjunto de resultados estáticos falsos se contempla un aumento en el porcentaje de píxeles detectados como dinámicos. Además, a pesar de que no es el caso de la Figura 38b, gran parte de los objetos dinámicos detectados pierden consistencia, obteniendo siluetas indistinguibles.



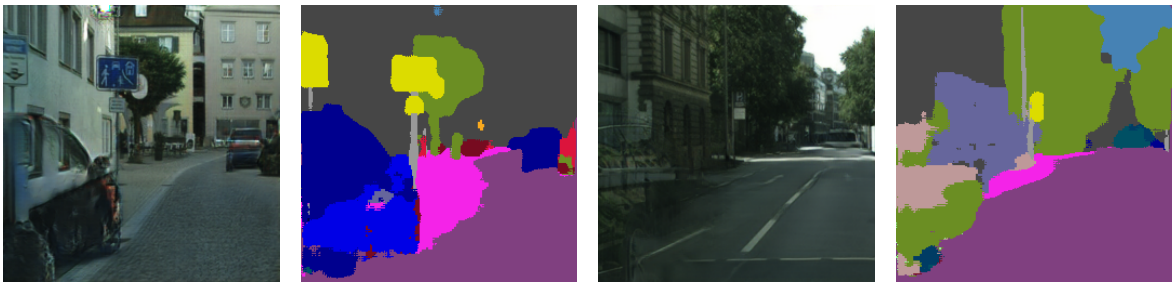
(a) falsa dinámica (b) segmentación falsa dinámica (c) falsa estática (d) segmentación falsa estática

Figura 38: Resultados ERFnet del modelo con máscaras, pesos y detección de ruido obtenidos para las imágenes mostradas en la Figura 26. Los colores se encuentran definidos en la Tabla 1

.

5.2.5 Modelo FAST

Los resultados semánticos obtenidos para el *Modelo FAST* (Figura 39), en el caso de la traducción de imágenes dinámicas a estáticas, son similares al *Modelo Noise* (Figura 39d). Por el contrario, se observa que el conjunto de píxeles dinámicos detectados como semánticos en las imágenes dinámicas falsas aumenta en comparación con el anterior. Aún así, el problema comentado en el *Modelo Noise*, la perdida de definición en las siluetas, se sigue manteniendo.

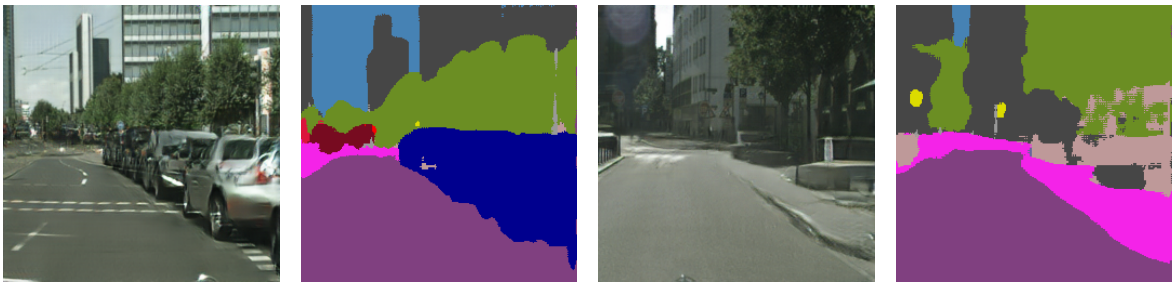


(a) falsa dinámica (b) segmentación falsa dinámica (c) falsa estática (d) segmentación falsa estática

Figura 39: Resultados ERFnet del modelo con máscaras, pesos, detección de ruido y detector FAST obtenidos para las imágenes mostradas en la Figura 27. Los colores se encuentran definidos en la Tabla 1

.

5.2.6 Model FAST without Noise



(a) falsa dinámica (b) segmentación falsa dinámica (c) falsa estática (d) segmentación falsa estática

Figura 40: Resultados ERFnet del modelo con máscaras, pesos y detector FAST obtenidos para las imágenes mostradas en la Figura 28. Los colores se encuentran definidos en la Tabla 1

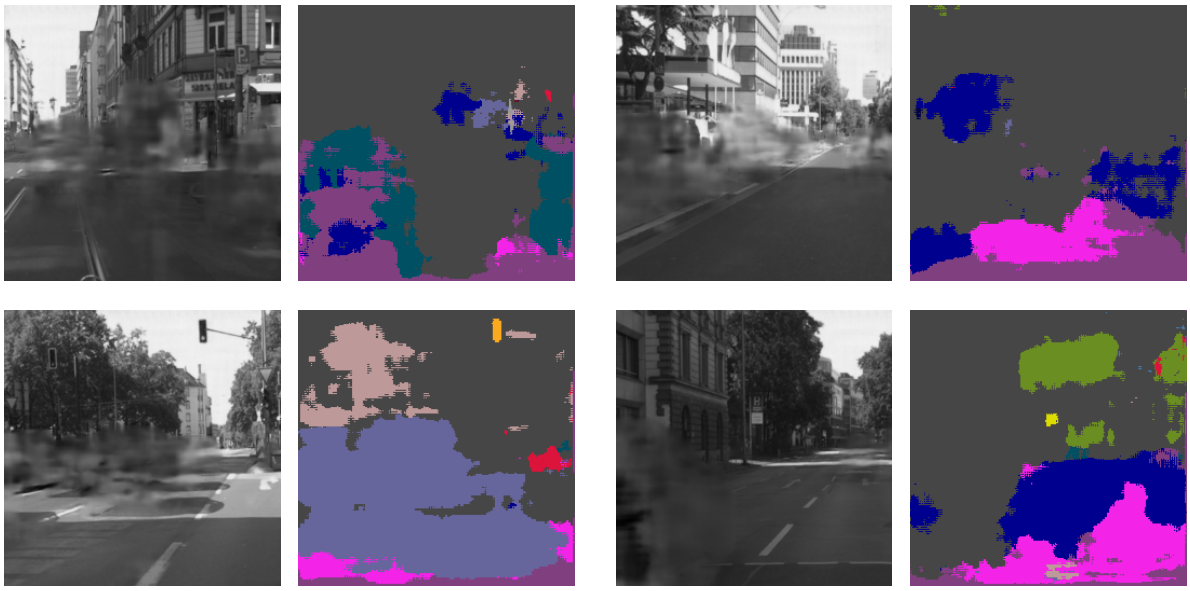
.

Finalmente, se evalúa la información semántica del *Modelo FASTwithoutNoise*. Como se puede observar en la Figura 40b se obtiene una imagen totalmente definida, en la cual se presentan figuras lisas sin irregularidades. En la generación de imágenes

estáticas no se obtienen los mejores resultados hasta el momento pero, aún así, son resultados destacables (Figura 40d).

5.2.7 Modelo Empty Cities

Los resultados semánticos obtenidos con el *Modelo Empty Cities* no muestran ninguna figura definida, haciendo imposible la visualización de instancias en las escenas generadas (Figura 41). Es importante destacar que el color de las imágenes de entrada aporta a las redes de segmentación semántica una alta cantidad de información. El hecho de que Empty Cities sólo trabaje con imágenes en escala de grises hace que no sea completamente "justo" el compararnos directamente con ellos con esta métrica.



(e) falsa estática (f) segmentación falsa estática (g) falsa estática (h) segmentación falsa estática

Figura 41: Resultados ERFnet del Modelo Empty Cities. Los colores se encuentran definidos en la Tabla 1

5.3 Evaluación Cuantitativa

Una vez obtenida la información semántica de cada imagen, se han realizado dos evaluaciones. La primera evaluación se basa en la clasificación de las imágenes en estáticas y dinámicas y la segunda evaluación, se basa en el porcentaje de píxeles dinámicos que contiene cada imagen.

5.3.1 Clasificación de las imágenes.

A continuación, se va a explicar como se ha realizado la clasificación de las imágenes en estáticas o dinámicas a partir de la información semántica obtenida.

Esta evaluación ha seguido los valores de clasificación utilizados inicialmente para clasificar las imágenes utilizadas en el dataset de entrenamiento, es decir, las imágenes que no contienen más del 1.5 % de píxeles dinámicos son clasificadas como imágenes estáticas y, aquellas que tienen un mínimo del 20 % de píxeles dinámicos como dinámicas.

A la hora de detectar los píxeles dinámicos de una imagen se ha observado el valor RGB devuelto por ERFnet para cada píxel. ERFnet devuelve un valor RGB específico para cada clase (persona, conductor, coche, etc.) (Tab. 1). Por tanto, en función a esa información se ha realizado la clasificación de cada imagen.

,

	RGB
DESCONOCIDO	**(0, 0, 0)**
PERSONA	**(220, 20, 60)**
CONDUCTOR	**(255, 0, 0)**
COCHE	**(0, 0, 142)**
CAMIÓN	**(0, 0, 70)**
BUS	**(0, 60, 100)**
TREN	**(0, 80, 100)**
MOTO	**(0, 0, 230)**
BICI	**(119, 11, 32)**
CARRETERA	**(128, 64, 128)**
ACERA	**(244, 35, 232)**
EDIFICIO	**(70, 70,70)**
MURO	**(102, 102, 156)**
VALLA	**(190, 153, 153)**
SEMÁFORO	**(250, 170, 30)**
SEÑAL	**(220, 220, 0)**
VEGETACIÓN	**(107, 142, 35)**
TERRENO	**(152, 251, 152)**
CIELO	**(70, 130, 180)**

Tabla 1: Codificación de ERFnet para clasificar los píxeles dinámicos.

Lo que se espera a la hora de evaluar los resultados es que, de manera general, en todos los modelos implementados las imágenes que anteriormente eran dinámicas ahora sean clasificadas como estáticas, y lo mismo para el caso contrario. Como se puede observar (Tab. 2 y Tab. 3), en función a los resultados obtenidos por ERFnet, la implementación cuyos resultados han sido más significativos, en los dos tipos de traducción, es la utilización de máscaras.

Traducción de imágenes **dinámicas** a **estáticas**

	Dinámicas	Estáticas	No definidas
Modelo Empty Cities	22	63	63
Modelo Original	5	77	66
Modelo Mask	0	132	16
Modelo Weight	0	133	15
Modelo Noise	0	135	13
Modelo FAST	0	133	15
Modelo FASTwithoutNoise	**0**	**135**	**13**

Tabla 2: Clasificación de los resultados obtenidos tras aplicar ERFnet sobre las imágenes dinámicas que han sido traducidas a estáticas.

Por otro lado, en la traducción de imágenes dinámicas a estáticas podemos ver que, aunque pequeña, se produce una mejora de los resultados con el uso de pesos y la detección de ruido. Por el contrario, el detector FAST en este caso nos empeora levemente los resultados obtenidos hasta el momento.

Además, se observa que con el *Modelo FASTwithoutNoise* aumentan los resultados clasificados correctamente respecto al *Modelo FAST* pero, a su vez, se observa que ofrecen el mismo número de resultados clasificados correctamente que el *Modelo Noise*. Esto nos lleva a pensar que la detección de manipulación de imágenes junto con la detección FAST no está funcionando correctamente en este tipo de traducción.

Traducción de imágenes **estáticas** a **dinámicas**

	Dinámicas	Estáticas	No definidas
Modelo Original	96	3	49
Modelo Mask	**145**	**1**	**2**
Modelo Weight	135	2	11
Modelo Noise	95	2	51
Modelo FAST	113	0	35
Modelo FASTwithoutNoise	128	1	19

Tabla 3: Clasificación de los resultados obtenidos tras aplicar ERFnet sobre las imágenes estáticas que han sido traducidas a dinámicas. En este caso no se incluye el Modelo Empty Cities ya que no realiza traducción de imágenes estáticas a dinámicas.

En la Tabla 3 se observa que tanto la utilización de pesos como la del detector de ruido produce una reducción en los resultados esperados en la traducción de imágenes estáticas a dinámicas. En cambio, el detector FAST en este caso nos mejora los

resultados que habíamos empeorado con las características anteriores. Finalmente, en este tipo de traducción, se aprecia un aumento considerable en las imágenes dinámicas detectadas correctamente con el *Modelo FASTwithoutNoise*.

5.3.2 Evaluación del porcentaje de píxeles dinámicos.

La segunda evaluación, realizada a partir de las imágenes semánticas obtenidas de la red ERFnet, ha consistido en evaluar el porcentaje de píxeles dinámicos que contiene cada imagen. Con esta evaluación se quiere observar el aumento o disminución, en función del tipo de traducción realizada, en el porcentaje de píxeles dinámicos que contiene cada imagen resultado de la traducción del conjunto de validación, para cada modelo entrenado. La detección de píxeles dinámicos se ha llevado a cabo igual que en la evaluación anterior, observando el valor RGB de cada píxel (Tab. 1).

El conjunto de valores, que representan los porcentajes de píxeles dinámicos para cada una de las imágenes, se muestran mediante un **diagrama de caja** (Figura 42 y Figura. 43). De esta manera, el diagrama indica a simple vista la mediana, los cuartiles y los percentiles de los porcentajes dinámicos calculados.

Inicialmente se observa que el *Modelo Original* ofrece una mejora significativa, en los resultados obtenidos, respecto al *Modelo Emtpy Cities* en ambas traducciones (Figura 42 y Figura 43). Esto indica que el modelo de entrenamiento elegido *CycleGAN* es completamente válido para nuestro objetivo.

En la **traducción de imágenes dinámicas a estáticas** (Figura 42), se observa que el *Modelo Mask* ofrece una mejora considerable en el porcentaje de píxeles dinámicos detectados, siendo mejorado levemente sus resultados en el *Modelo Weight*. Por otro lado, los resultados obtenidos en el *Modelo Noise* muestran un aumento en el máximo porcentaje de píxeles dinámicos obtenido, pero a su vez el cuartil Q3 (valor por debajo del cual quedan las tres cuartas partes) se reduce considerablemente, siendo el mejor resultado obtenido entre los modelos. Por otro lado, se observa que el *Modelo FAST* ofrece un leve aumento en el porcentaje de píxeles dinámicos sobre el *Modelo Noise*.

Finalmente, los resultados obtenidos por el *Modelo FASTwithoutNoise* son similares al *Modelo FAST* y al *Modelo Noise*, continuando este último como el modelo con mejores resultados obtenidos en relación a los cuartiles. Además, se observa una reducción considerable en el máximo porcentaje de píxeles dinámicos obtenidos con el *Modelo FASTwithoutNoise*.

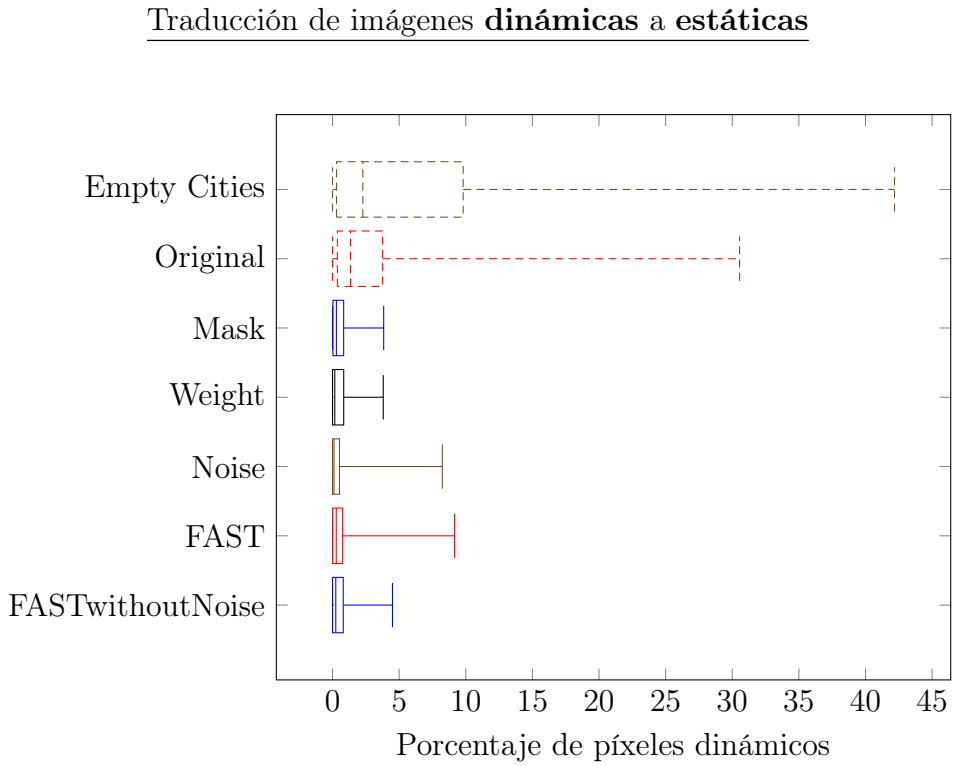


Figura 42: Diagrama de caja - Porcentaje de píxeles dinámicos: Imágenes dinámicas a estáticas. Cuanto menor sea el porcentaje de píxeles dinámicos mejor es la solución obtenida.

Por último, en la **traducción de imágenes estáticas a dinámicas** (Figura 43) se observa como el *Modelo Mask* ofrece mejores resultados que el resto de modelos, siendo el *Modelo Noise* el peor, por debajo incluso del *Modelo Original*. Además, el *Modelo FAST* presenta un aumento en el porcentaje de píxeles dinámicos detectados respecto al *Modelo Noise*, mejorando significativamente el cuartil Q1 (valor por debajo del cual queda un cuarto), y respecto a la traducción de imágenes dinámicas. Esto último puede deberse a que el *detector FAST* se centra en contornos, viéndose aumentados en el caso de generar objetos dinámicos.

Además, en el caso del *Modelo FASTwithoutNoise* se aprecia un aumento relevante en el porcentaje de píxeles dinámicos respecto al *Modelo Noise* y al *Modelo FAST*, obteniendo unos resultados similares al *Modelo Weight* pero viéndose aumentado el porcentaje máximo de píxeles dinámicos.

En general, se observa cierta incompatibilidad entre la *detección de manipulación* y el resto de implementaciones diseñadas.

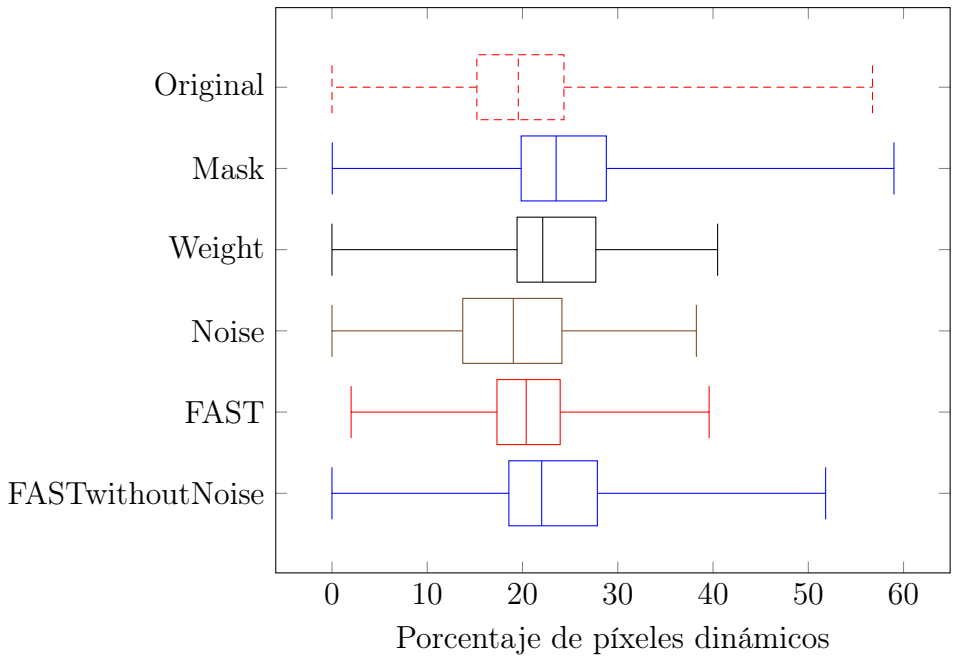


Figura 43: Diagrama de caja - Porcentaje de píxeles dinámicos: Imágenes estáticas a dinámicas. En este caso, no se podrá obtener un porcentaje de píxeles dinámicos igual al porcentaje total de la imagen. Debido a la aplicación de máscaras se le indica a la red la zona donde debería generar los objetos dinámicos, por tanto, el máximo porcentaje de píxeles dinámicos a obtener sería el porcentaje de píxeles dinámicos que contiene la máscara.

6 Conclusiones

Con este trabajo se ha presentado un marco de aprendizaje que coge como entrada una imagen RGB con contenido dinámico, como vehículos o peatones, y la traduce en una imagen RGB estática realista.

Para ello, se utiliza el modelo *CycleGAN* [11], que utiliza un conjunto de entrenamiento no emparejado. Además, el modelo añade un mapeo inverso, permitiendo aprender también la traducción de imágenes estáticas a dinámicas.

El principal objetivo del trabajo es la traducción de imágenes dinámicas a estáticas. Por tanto, con la finalidad de mejorar los resultados obtenidos, se introducen diferentes características en el entrenamiento del modelo. Aunque no es el objetivo principal también se tiene en cuenta la mejora producida en la traducción de imágenes estáticas.

Los resultados preliminares son prometedores. Inicialmente, ya se observa una mejora importante del *Modelo CycleGAN* [11] frente al *Modelo Empty Cities* [2], en el que se ve un aumento en el número de imágenes traducidas correctamente. Posteriormente, los resultados evaluados correctamente se incrementan nuevamente añadiendo el conjunto de implementaciones diseñadas, produciéndose una mejora visual en la imagen generada y en la información semántica del píxel.

Por otro lado, aunque la *detección de manipulación* funciona adecuadamente en el trabajo original [23], se percibe cierta incompatibilidad en nuestro modelo entre la implementación desarrollada y el resto de características, ocasionando un aumento en el número de resultados erróneos cuando se encuentra activa. Apreciándose, en mayor medida, en el caso de la traducción de imágenes estáticas a dinámicas (Tabla 3 y Figura 43). Dicha incompatibilidad puede deberse a un ajuste erróneo en los hiperparámetros, dando mayor relevancia al error producido por la *detección de manipulación* que por el resto de la red. También es posible que la *detección de manipulación* no funcione correctamente a la hora de generar imágenes dinámicas ya que este es un caso que no se ha contemplado en el trabajo original [23] .

Por tanto, como trabajo futuro se contempla la creación de máscaras dinámicas, teniendo en cuenta el entorno, el aumento de la información semántica de cada objeto dinámico contenido en la imagen y la mejora de los hiperparámetros utilizados.

7 Trabajo futuro

Como trabajo futuro se contempla la creación de máscaras de manera inteligente, mejorando así la traducción de imágenes estáticas a dinámicas. Estás máscaras se diseñarán teniendo en cuenta el marco de las imágenes estáticas. La finalidad de esta idea consiste en obtener imágenes coherentes con el entorno, permitiendo que la red aprenda las zonas idóneas donde establecer los nuevos objetos dinámicos. Además, se pretende añadir más información semántica permitiendo diferenciar individualmente cada objeto dinámico contenido en una imagen, con ello se espera obtener objetos dinámicos más definidos y realistas.

También se desea realizar un ajuste del valor de los hiperparámetros utilizados en la *detección de manipulación*. El objetivo es mejorar los resultados obtenidos con la implementación de esta característica ya que, como se ha indicado anteriormente, se observa cierta incompatibilidad con el resto de implementaciones diseñadas.

Finalmente, se quiere realizar una evaluación del conjunto de datos generados sobre SLAM, ya que en un sistema de mapeo y localización simultáneos es común asumir que la escena en la que se trabaja es estática.

Referencias

[1] B. Bescos, J. Neira, R. Siegwart, and C. Cadena, "Empty Cities: Image Inpainting for a Dynamic-Object-Invariant Space," pp. 5460–5466, IEEE, 2019.

[2] B. Bescos, C. Cadena, and J. Neira, "Dynamic-to-static image translation for visual slam," in *Workshop at RSS 2019 Scene and Situation Understanding for Autonomous Driving*.

[3] Y. Wang and S. Huang, "Motion segmentation based robust RGB-D SLAM," in *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pp. 3122–3127, IEEE, 2014.

[4] P. F. Alcantarilla, J. J. Yebes, J. Almazán, and L. M. Bergasa, "On combining visual slam and dense scene flow to increase the robustness of localization and mapping in dynamic environments," in *2012 IEEE International Conference on Robotics and Automation*, pp. 1290–1297, IEEE, 2012.

[5] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments," in *Robotics and Automation (ICRA), 2018 IEEE International Conference on*, Institute of Electrical and Electronics Engineers, 2018.

[6] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, pp. 263–272, 2018.

[7] A. Telea, "An image inpainting technique based on the fast marching method," *Journal of graphics tools*, vol. 9, no. 1, pp. 23–34, 2004.

[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[10] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[11] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[12] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2016.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2014.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[15] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[16] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," *arXiv preprint arXiv:1704.03489*, 2017.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014.

[18] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[19] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.

[21] M. Liu and O. Tuzel, "Coupled generative adversarial networks," *CoRR*, vol. abs/1606.07536, 2016.

[22] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," *CoRR*, vol. abs/1703.00848, 2017.

[23] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, "Learning rich features for image manipulation detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[24] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012.

[25] A. Ker and R. ohme, "Revisiting weighted stego-image steganalysis," *Electronic Imaging 2008, International Society for Optics and Photonics*, vol. 6819, 03 2008.

[26] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *In European Conference on Computer Vision*, pp. 430–443, 2006.

[27] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[29] E. Romera, L. M. Bergasa, and R. Arroyo, "Can we unify monocular detectors for autonomous driving by using the pixel-wise semantic segmentation of cnns?," *CoRR*, vol. abs/1607.00971, 2016.

Anexos

A Resultados - Traducción de imágenes

En las figuras que se presentan a continuación (Figura 44 - Figura 49) se muestran los resultados obtenidos para tres traducciones de imágenes dinámicas a estáticas y tres traducciones de imágenes estáticas a dinámicas, con cada uno de los modelos entrenados.

Como se ha hablado anteriormente, el *Modelo withoutNoise* ofrece mejores resultados para la traducción de imágenes dinámicas. En cambio, los mejores resultados obtenidos para la traducción de imágenes estáticas se consiguen con el *Modelo Mask*.

En la Figura 44 se observa que el *Modelo Mask* y el *Modelo Weight* presentan buenos resultados, viéndose empeorados a continuación por los siguientes modelos hasta llegar al *Modelo withoutNoise*, que consigue eliminar prácticamente casi todas las zonas dinámicas.

Si se observa con detenimiento la Figura 45 se pueden ver leves mejoras en la definición de la imagen falsa (fondo de la calle, acera, edificios). En este caso, el *Modelo Weight* y el *Modelo withoutNoise* presentan resultados similares, aunque este último muestra una imagen más definida y clara, esto puede apreciarse principalmente en el fondo de la calle.

En la Figura 46 se puede apreciar que el mejor resultado obtenido es el presentado por el *Modelo withoutNoise*, consiguiendo eliminar el mayor porcentaje de píxeles dinámicos de la imagen, mostrando una carretera casi lisa.

En la traducción de imágenes estáticas a dinámicas los resultados obtenidos no son tan buenos como en el caso anterior. En la Figura 47 el mejor resultado es el presentado por el *Modelo Weight* ya que presenta un conjunto de vehículos más definido que el resto de modelos, pudiendo diferenciarse con mayor facilidad.

Los resultados obtenidos en la Figura 48 son muy similares entre sí, a excepción del *Modelo Original*. Inicialmente, es posible diferenciar individualmente los objetos representados en las imágenes. Observando detenidamente podría decirse que el mejor resultado es el ofrecido por el *Modelo Mask*, ya que presenta una mayor definición en

los objetos, apreciándose incluso una matrícula en el primer coche, visible en la imagen, de la derecha.

Finalmente, en la última figura (Figura 49) los mejores resultados obtenidos son los presentados por el *Modelo Mask* y el *Modelo withoutNoise*, ya que el resto de imágenes muestran objetos dinámicos transparentes. Aún así, en el *Modelo withoutNoise* se puede observa que el primer coche sigue permitiendo ver levemente los arbustos que se encuentran detrás.

Figura 44: Resultado obtenido de la traducción de una imagen dinámica a una estáticas en los diferentes modelos entrenados.

Figura 45: Resultado obtenido de la traducción de una imagen dinámica a una estáticas en los diferentes modelos entrenados.

Figura 46: Resultado obtenido de la traducción de una imagen dinámica a una estáticas en los diferentes modelos entrenados.

Figura 47: Resultado obtenido de la traducción de una imagen estática a una dinámica en los diferentes modelos entrenados.

Figura 48: Resultado obtenido de la traducción de una imagen estática a una dinámica en los diferentes modelos entrenados.

Figura 49: Resultado obtenido de la traducción de una imagen estática a una dinámica en los diferentes modelos entrenados.

B Diagrama de Gantt

En la Figura 50 se muestra el **Diagrama de Gantt** que se ha seguido durante la elaboración de este trabajo de fin de grado. Se compone de seis secciones dividas en función al conjunto de tareas que integran.

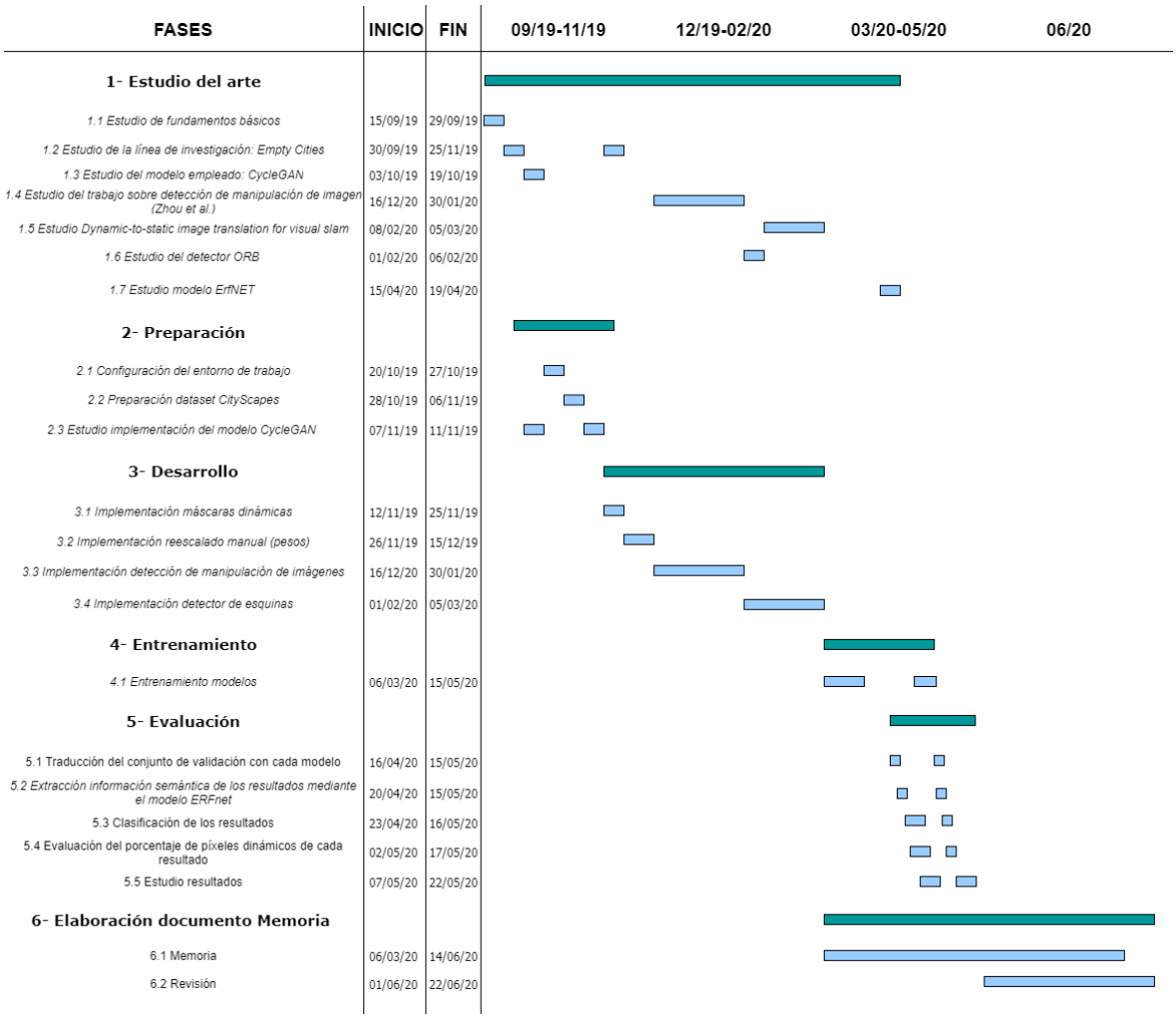


Figura 50: Diagrama de Gantt