



Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos



Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Domain Adaptation for Face Landmark Detection

Autor: Alberto Ruiz Alejandro
Tutor: Luis Baumela

Madrid, Julio 2024

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

*Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial
Título: Domain Adaptation for Face Landmark Detection
Julio 2024*

*Autor: Alberto Ruiz Alejandro
Tutor: Luis Baumela
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid*

Abstract

Domain adaptation addresses the challenge of transferring knowledge from a labeled source domain to an unlabeled target domain, which is crucial for tasks where labeled data is scarce or expensive to obtain. This thesis explores the application of the Domain-Adversarial Neural Network (DANN) method, as proposed by Ganin et al. in 2015, to achieve effective domain adaptation in different contexts. Initially, we apply DANN to the problem of domain adaptation between the SVHN and MNIST datasets, two benchmark datasets in the field of digit recognition. Through extensive experimentation, we demonstrate that the DANN method significantly improves the classification performance on the target domain by learning domain-invariant features. Subsequently, we extend the application of DANN to the more complex task of facial landmark detection, which poses unique challenges due to variations in facial expressions, lighting conditions, and occlusions across different domains. Although the results did not meet expectations, the experiments and their outcomes are thoroughly discussed, providing a foundation for future work in this area.

Keywords: domain adaptation, landmark detection, source, target, DANN, synthetic data.

Resumen

La adaptación de dominio aborda el desafío de transferir conocimiento desde un dominio fuente etiquetado a un dominio objetivo no etiquetado, lo cual es crucial para tareas donde los datos etiquetados son escasos o costosos de obtener. Esta tesis explora la aplicación del método de Red Neuronal Adversaria de Dominio (DANN, por sus siglas en inglés), propuesto por Ganin et al. en 2015, para lograr una adaptación de dominio efectiva en diferentes contextos. Inicialmente, aplicamos DANN al problema de la adaptación de dominio entre los conjuntos de datos SVHN y MNIST, dos conjuntos de datos de referencia en el campo del reconocimiento de dígitos. A través de experimentos extensivos, demostramos que el método DANN mejora significativamente el rendimiento de clasificación en el dominio objetivo al aprender características invariantes al dominio. Posteriormente, extendemos la aplicación de DANN a la tarea más compleja de la detección de landmarks, que plantea muchas dificultades debido a las variaciones en expresiones faciales, condiciones de iluminación y occlusiones en diferentes dominios. Aunque los resultados no cumplieron con las expectativas, los experimentos y sus resultados se discuten a fondo, proporcionando una base para trabajos futuros en esta área.

Palabras clave: adaptación de dominio, detección de landmarks, fuente, destino, DANN, datos sintéticos.

Índice

1. Introducción	6
1.1. Objetivos	7
1.2. Organización del documento	7
2. Estado del arte	8
2.1. Adaptación de dominio	8
2.1.1. Contexto y diferentes enfoques	8
2.1.2. DA clásico: métodos poco profundos	11
2.1.3. DA clásico: métodos profundos	14
2.1.3.1. Enfoques posibles	14
2.1.3.2. Deep DA	16
2.1.4. Más allá de la clasificación de imágenes	18
2.2. Detección de Landmarks	19
2.2.1. Primeras propuestas	19
2.2.2. Propuestas Modernas: redes neuronales	20
2.2.2.1. Algoritmos	20
3. Domain Adversarial Neural Network	22
3.1. Introducción	22
3.2. Formulación del problema	22
3.3. Divergencia entre los dominios y distancia proxy	22
3.4. Arquitectura	23
4. Implementación	26
4.1. Conjuntos de dígitos: SVHN y MNIST	26
4.1.1. Arquitectura	26
4.2. Detección de Landmarks	27
4.2.1. Datasets	27
4.2.1.1. Face Synthetics	27
4.2.1.2. 300W-LP	29
4.2.2. Arquitectura del modelo	30
4.2.3. Flujo de ejecución	31
4.2.4. Métricas	31
5. Experimentos	33
5.1. Conjuntos de dígitos	33
5.1.1. Solo destino: MNIST	33
5.1.2. Solo fuente: SVHN	33
5.1.3. DANN	34
5.2. Detección de landmarks	37
5.2.1. Solo destino: 300W-LP	37
5.2.2. Solo fuente: Face Synthetics	38
5.2.3. DANN	39
6. Conclusiones y líneas de trabajo futuras	44

1. Introducción

En la última década se han logrado grandes avances en el campo del deep learning debido a diversos factores como las mejoras en el hardware, desarrollándose GPUs cada vez más potentes, o la disponibilidad de grandes cantidades de datos, con el auge del big data. Sin embargo, el deep learning sigue teniendo importantes limitaciones. Una de las más importantes es que requieren grandes cantidades de datos etiquetados. Debido a esto últimamente han surgido datos sintéticos y simuladores como una solución prometedora.

Comparado con los datos reales, los sintéticos son más fáciles de generar, están preanotados y son menos costosos [1]. También pueden evitar cuestiones éticas y prácticas. Además, introducen oportunidades únicas en el sentido de que permiten obtener datos de entrenamiento que pueden ser difíciles o imposibles de colección en el mundo real. Otra ventaja de ellos es que permiten aprender mejores representaciones del mundo. Éstas se parecen más al aprendizaje presente en sistemas biológicos y en seres humanos.

El aprendizaje que se logra usando datos sintéticos se caracteriza por ser **multimodal**, **continuo** y **embebido**. El aprendizaje multimodal se refiere al proceso de aprendizaje de conocimiento a partir de información sincronizada en el tiempo de múltiples sensores, como la entrada auditiva, visual y háptica. El continuo permite aprender tareas en una secuencia, de manera que no se olviden las tareas anteriores de la secuencia durante el proceso. Por último, el embebido es aquel en el que se aprende a partir de información multimodal, obtenida mediante la interacción con el entorno.

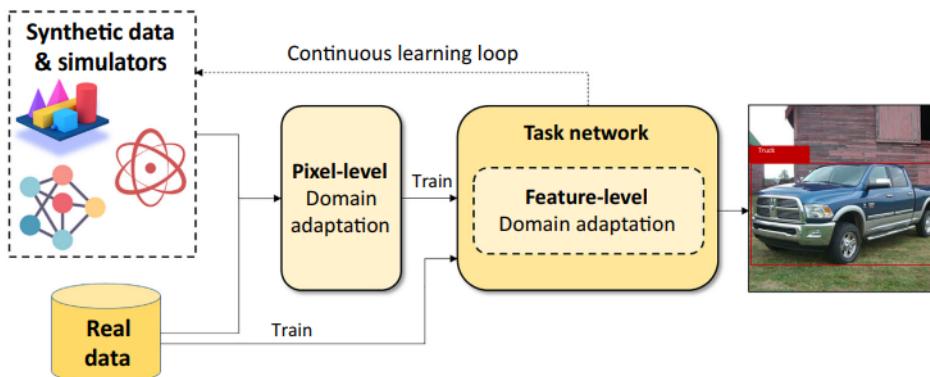


Figura 1: Integración de los pipelines del simulador y el aprendizaje [1]

Una línea de investigación que se ve prometedora es la de integrar los *pipelines* de síntesis y aprendizaje (ver figura 1). Esto se ha visto anteriormente sobretodo en el aprendizaje por refuerzo, pero puede ser extendido al aprendizaje supervisado y permitir los aprendizajes comentados en el párrafo anterior.

Sin embargo, el uso de datos generados sintéticamente produce lo que se conoce como *domain shift*, un problema que ha ganado interés en los últimos tiempos. Éste ocurre cuando la distribución de datos de test es diferente de la de los datos de entrenamiento. En el caso que nos ocupa, el domain shift consiste en que la apariencia de las imágenes generadas sintéticamente no es exactamente igual a las reales. Para solucionarlo se han desarrollado diferentes técnicas. La aleatorización de dominio, los modelos híbridos o la adaptación de dominio son algunas de ellas. Este trabajo se centra en esta última,

implementando la técnica Domain Adversarial Neural Networks (DANN), propuesta en 2015 por Ganin et al. [2]. Esto se hará primero para un dominio sencillo, utilizando los famosos datasets de dígitos MNIST y SVHN, y luego se desarrollará un detector de landmarks y aplicará la técnica en los datasets Face Synthetics y 300W-LP.

1.1. Objetivos

Este trabajo se centra en la adaptación de dominio y en su uso en conjunto con el dataset de imágenes sintéticas Face Synthetics en la tarea de detección de landmarks. Los objetivos principales que persigue son:

- Revisar las técnicas de adaptación de dominio que ha habido en el campo de la visión por computador hasta la fecha.
- Revisar las técnicas de detección de landmarks que ha habido hasta la fecha, principalmente aquellas que utilizan aprendizaje profundo.
- Estudiar con detalle la técnica DANN, y replicar los resultados del paper original en los datasets de dígitos MNIST y SVHN.
- Desarrollar un detector de landmarks en el conjunto Face Synthetics, y aplicar la técnica DANN en este escenario.

Mediante la consecución de dichos objetivos se pretende lograr un conocimiento profundo de la adaptación de dominio y del estado del arte de éste, y aplicarlo en un problema real para ver si es factible y da buenos resultados.

1.2. Organización del documento

El documento está organizado como sigue: en el capítulo 2 se revisa el estado del arte de la adaptación de dominio en el contexto de la visión por computador y también el de la tarea de detección de landmarks. En el caso de la adaptación de dominio se comentarán tanto técnicas poco profundas como aquellas que usan deep learning. Aunque la mayoría de trabajos tratan el problema de clasificación, se añaden también propuestas en otro tipo de tareas como la detección o la segmentación.

En el capítulo 3 se entra en más detalle en la técnica DANN [2], explicándose a fondo la idea y como funciona. En el capítulo 4 se detalla la implementación que se ha hecho de la adaptación de dominio: las arquitecturas, los datasets y las métricas utilizadas. En el capítulo 5 se exponen y discuten los distintos experimentos realizados, analizando también lo que no ha funcionado (en el caso de la detección de landmarks) para futuros intentos que se hagan de esta técnica en éste u otro problema. Finalmente en el último capítulo se exponen las distintas conclusiones y líneas de futuro que podrían seguirse partiendo de este trabajo.

2. Estado del arte

En esta sección se repasan las distintas técnicas que se han propuesto de adaptación de dominio y de detección de landmarks. Hacer adaptación de dominio sobre esta tarea es algo de lo que no se ha encontrado ningún trabajo, y de ahí también parte de la motivación de éste. En primer lugar se revisarán los métodos de adaptación de dominio, tanto los más antiguos basados en técnicas no profundas y características creadas manualmente, como los más recientes basados todos ellos en aprendizaje profundo. Para finalizar la sección se revisarán también las técnicas más destacadas de la detección de landmarks, poniendo el foco de nuevo en el aprendizaje profundo.

2.1. Adaptación de dominio

2.1.1. Contexto y diferentes enfoques

En este primer apartado del estado del arte, se revisa el modelo clásico de domain adaptation, junto con sus diversas variantes y enfoques desarrollados hasta la fecha. Además, se explora la relación entre el domain adaptation y otros enfoques de aprendizaje de representaciones, como el transfer learning y el self-supervised learning. El objetivo es proporcionar una visión general de la posición del domain adaptation en relación con otros marcos de aprendizaje, y también mostrar otras técnicas relevantes que, aunque no sean tratadas en este trabajo, conviene tener presentes al hablar sobre DA.

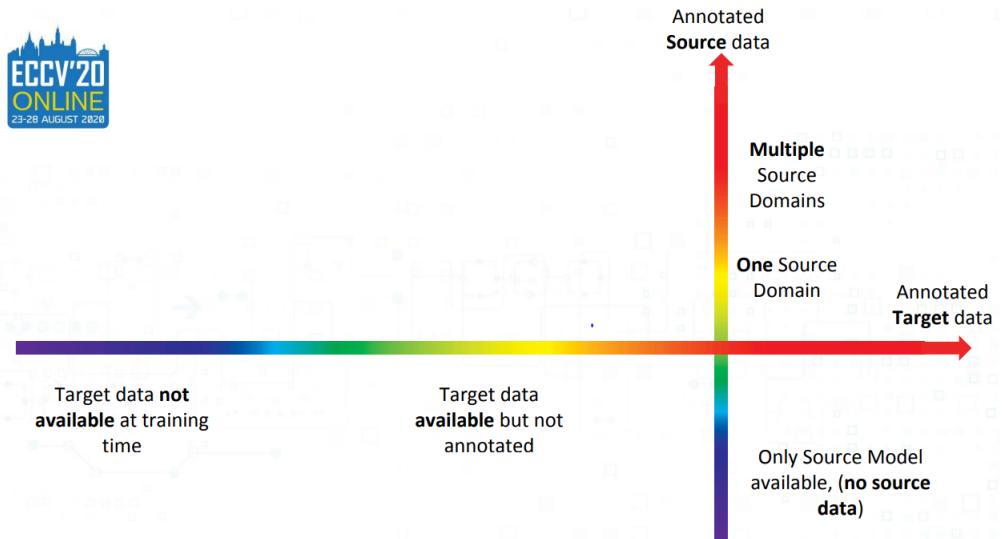


Figura 2: Esquema de dos ejes correspondientes a los datos de dominios fuente y destino [3].

Todo lo que se expone a lo largo de este apartado se ve esquematizado en las figuras 2 y 3. En ellos nos basaremos para explicar uno a uno los distintos enfoques que existen, y hacernos una idea de lo que son cada uno y como se relacionan.

Primero echemos un vistazo a la figura 2. En ella se ven dos ejes que cruzan. El eje horizontal representa los datos del dominio destino o *target domain*, mientras que el vertical los datos del dominio fuente o *source domain*. Sobre dicha gráfica se irán posicionando los distintos modelos según las características que tengan. En el eje horizontal y de izquierda a derecha se tienen modelos cuyos datos del dominio destino

no están disponibles durante el entrenamiento, están disponibles pero no anotados y finalmente están disponibles y anotados. Respecto al eje vertical y de abajo a arriba, empezamos por modelos en los que se tiene acceso únicamente al modelo fuente y, seguidamente, aquellos que tienen acceso al modelo y a los datos anotados, tanto para un único dominio fuente como varios.

Ahora explicamos cada uno de los modelos de la figura 3. Empezamos por el DA clásico, que es el más estudiado y el que se implementa en este trabajo. Este corresponde al **Transductive Unsupervised Domain Adaptation**, y está cercano a la intersección de ambos ejes. En él los datos destino o *target data* están disponibles durante el entrenamiento pero no anotados. Se puede tener uno o más dominios fuente. El término *transductive* significa que los datos destino utilizados para la parte de test son los mismos que se usan para entrenar.

Cercano al anterior se encuentra el **Semi-supervised Domain Adaptation**, que difiere en que parte de los datos destino están anotados. Además, no suele ser transductive, por lo que los datos de entrenamiento y test (del dominio destino) no es necesario que coincidan. Cercano al semi-supervised DA se tiene el bien conocido **Transfer Learning**, en el cual los datos destino están anotados, pero del dominio fuente sólo podemos acceder al modelo, no a los datos.

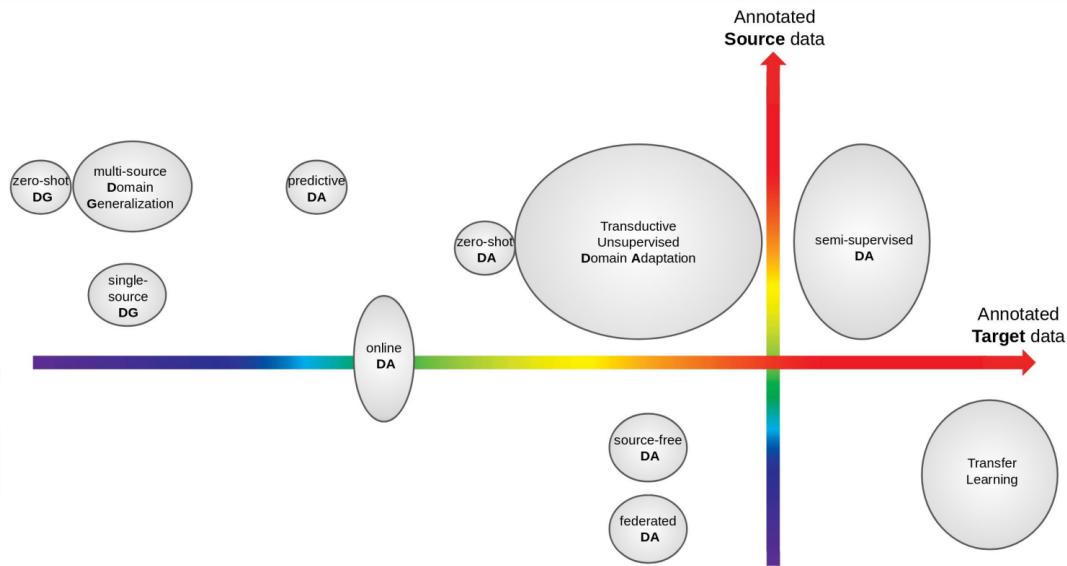


Figura 3: Esquema de la figura 2, pero completado con los distintos enfoques del DA, así como DG y Transfer Learning [3]

Moviéndonos ahora hacia la izquierda se encuentran técnicas de una dificultad mayor. En **Multi-source Domain Generalization** no se tiene acceso a los datos destino durante el entrenamiento, lo que significa que realmente no sabemos los dominios destino a los cuales nos queremos adaptar. El objetivo, por tanto, es aprender un modelo que funcione bien para cualquier dominio destino que se pueda ver durante las pruebas. Para lograrlo, se suele apoyar en el entrenamiento en múltiples dominios fuente, aunque también ha sido estudiado en el caso de uno sólo. En [4], [5] y [6] se explora estas dos técnicas de domain generalization (DG).

A continuación se exponen casos particulares de DA y DG: **zero-shot DA** y **zero-shot DG**. Aunque comparten su nombre, tienen objetivos distintos. En zero-shot

DA la clasificación es sobre el mismo conjunto de clases que el dominio fuente, pero puede extraer información adicional de las distintas categorías. En zero-shot DG la clasificación se hace utilizando un conjunto de clases nunca vistas antes. [7] y [8] son trabajos relativos al zero-shot DA y zero-shot DG respectivamente.

Como un caso intermedio entre DA y DG se tiene el **Predictive DA**, en el que hay múltiples dominios fuente, pero estando anotado sólo uno. Además, los datos destino no están anotados. Algo particular de este modelo es que el dominio destino tiene metadatos, por lo que durante el entrenamiento se puede realizar algo similar al DA, aunque no se tengan las etiquetas como se mencionaba anteriormente. [9] y [10] son ejemplos de esta técnica.

Otro modelo intermedio entre ambos es el **Online DA**. En él los datos destino sin anotar no se pueden usar al principio del entrenamiento, pero se van volviendo disponibles progresivamente a lo largo de éste, entrando como un stream de datos no etiquetados. La posición sobre el eje horizontal se debe a que en algunos casos se necesita acceso a los datos del dominio fuente, mientras que en otros basta con el acceso únicamente al modelo fuente. En [11] se estudia esta técnica.

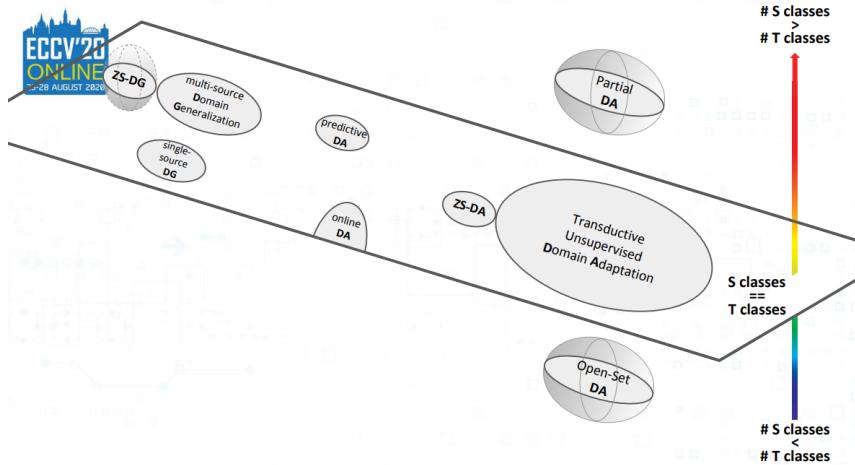


Figura 4: Esquema de los enfoques considerando una tercera dimensión [3]

Ahora pasamos a ver los modelos situados en el cuadrante de abajo a la izquierda. Dos de ellos son **Source-free DA** y **Federated DA**. En el source-free DA solo tenemos acceso al modelo fuente, y los datos destino se encuentran sin anotar, como en el DA clásico. [12] propone esta técnica como un proceso de dos fases: en la primera el modelo fuente se descompone en dos: el extractor de características y el clasificador del dominio fuente. En la segunda se hereda dicho clasificador y se actualiza el extractor. Otro ejemplo de source-free DA se puede ver en [13].

Por otro lado, en el federated DA se tienen múltiples modelos fuente, y cada uno en un repositorio privado. Cada propietario del repositorio entrena un modelo, pero no pueden sincronizarse entre ellos ni ser gestionados de manera externa. Cada modelo constituye un nodo en una red, pero cada uno puede tener un ratio de convergencia distinto. Se incrementan entonces los pesos de los nodos que sean beneficiosos para el dominio destino. El modelo destino se actualiza considerando el gradiente de cada modelo fuente y midiendo como de bien se hace clustering sobre las features destino producidas. [14] sigue el enfoque anterior y es un ejemplo de federated DA.

Una última clasificación muy interesante y que no ha sido comentada hasta ahora, tiene que ver con el número de clases presentes en los dominios fuente y destino. Hasta ahora, todos los modelos analizados tenían algo en común: los dos dominios tenían el mismo número de clases. En particular, excepto para el caso de zero-shot DG, se trataban de las mismas clases. Esto no tiene porque ser así, y variando dicho número se encuentran dos nuevos modelos: **Partial DA** y **Open-set DA**. En partial DA hay un conjunto de clases adicionales en el dominio fuente, mientras que en el open-set ocurre lo contrario. Un esquema considerando lo anterior se ve en la figura 4. En [15] se hace una revisión del DA clásico junto con estas dos técnicas. [16], [17], [18] son ejemplos de trabajos que estudian el partial DA. [19] y [20] son ejemplos de open-set DA.

Cerramos este apartado comentando el caso del self-supervised learning. En él la elección de la tarea es crucial, ya que las etiquetas deben ser capaces de autogenerarse sin la necesidad de supervisión humana. Aquí lo interesante es que una vez entrenado un modelo mediante self-supervised learning, se puede aplicar tanto al transfer learning como a distintas técnicas de DA. De hecho, gracias a él hasta se ha introducido un nuevo modelo adaptativo llamado **one-shot unsupervised** [21].

2.1.2. DA clásico: métodos poco profundos

Ahora pasamos a estudiar en detalle las diferentes propuestas que se han encontrado para el caso del DA clásico que definimos brevemente en el apartado anterior. Empezamos en este apartado por resumir el problema que tratamos, y a continuación repasamos las técnicas poco profundas o *shallow methods* existentes en DA. Éstas fueron las primeras que se propusieron debido a su sencillez. Aunque consiguen resultados considerablemente peores que otras que usan deep learning, fundaron las bases y usaron conceptos que se traspasarían más adelante al usar deep learning.

En el DA partimos de un conjunto de características $\mathbf{X}_s = \{\mathbf{x}_s^i\}_{i=1}^n$, $\mathbf{X}_t = \{\mathbf{x}_t^j\}_{j=1}^m$, donde n y m denotan el número de muestras de los dominios fuente y destino respectivamente. Los subíndices se refieren a los dominios, s para el fuente y t para el destino. En el caso del dominio fuente tenemos además un conjunto de etiquetas $\{\mathbf{y}_s^i\}_{i=1}^n$. Nos centraremos en lo que se conoce como *covariate shift*, que ha sido el más estudiado en la literatura.

En el problema de covariate shift se tiene $p_s(x_s) \neq p_t(x_t)$ y se asume que $p_s(y|x_s) = p_t(y|x_t)$, donde p_s y p_t denotan las distribuciones de probabilidad de los dominios fuente y destino respectivamente. El objetivo de DA es entonces encontrar una transformación T tal que

$$p_s(T(x_s)) = p_t(T(x_t))$$

Uno de los problemas principales en DA es entonces definir una métrica para medir la distancia entre las distribuciones fuente y destino y reducir así el domain shift. Para ello, Saenko et al. en su trabajo [22] presentan una primera distancia entre dos muestras de ambos dominios. Formulan el problema de tal forma que el objetivo sea aprender una matriz W minimizando la distancia, sujeto a unas restricciones que imponen.

Lo anterior era una buena primera aproximación, pero presentaba diversos problemas debido a las diversas asunciones que realizaba. Éstas se traducían en que se aprendía un mapping simétrico entre ambos dominios, lo cual no siempre es lo deseado. Por ello

Kulis et al. en [23] tratan el caso asimétrico, y lo que se aprende ahora es también una matriz W , pero asimétrica. Además, sustituyen el aprendizaje de una distancia por una noción de similaridad y las restricciones por regularizadores.

Los dos trabajos anteriores tienen algo más en común, y es que realmente tratan el caso del semi-supervised DA, cuando a nosotros realmente lo que nos interesa el DA clásico, también conocido como unsupervised DA, donde las etiquetas de los datos destino no están disponibles. A partir de ahora nos centramos en él y vemos los trabajos que ha habido.

Los primeros trabajos siguieron un enfoque basado en **Subspace Representations**, que consiste en que todos los datos de entrada \mathbf{X} de un dominio dado se ven como una entidad, forman un subespacio. Estos subespacios se encuentran en los grassmannianos, y utilizan conceptos de la geometría Riemanniana, como las curvas geodésicas. Esta idea fue introducida por Gopalan et al. en [24]. En dicho trabajo generan subespacios intermedios entre el subespacio fuente y destino. Dichos subespacios se encuentran a lo largo de la curva geodésica que conecta ambos subespacios.

La idea de dicho artículo fue considerar todos estos subespacios, y proyectar las muestras fuente y destino en todos estos subespacios (fuente, destino e intermedios). Se obtiene así un vector de representación grande para cada muestra en cada dominio y a continuación, se les aplica reducción de dimensionalidad mediante PLS (Partial Least Squares). Tras todo este proceso ya se puede utilizar la representación resultante para labores de clasificación.

Gong et al. [25] adoptaron una estrategia similar, pero en lugar de muestrear los subespacios intermedios, lo cual podría introducir dependencia en los resultados basados en la elección de los subespacios, propusieron una aproximación alternativa. Su enfoque consiste en integrar sobre todos los posibles subespacios intermedios, lo que resulta en vectores de dimensión infinita compuestos por la proyección de todos estos subespacios. Sin embargo, al calcular el producto interno entre dos de estos vectores, se obtiene la siguiente integral, para la cual se encuentra una solución analítica:

$$\langle z_i^\infty, z_j^\infty \rangle = \int_0^1 (\Phi(t)^T \mathbf{x}_i)^T (\Phi(t)^T \mathbf{x}_j) dt = \mathbf{x}_i^T \mathbf{G} \mathbf{x}_j$$

Esta última representación resultante ya puede usarse en clasificadores basados en núcleos. Fernando et al. en [26] argumenta que realmente los subespacios intermedios no son necesarios, que los importantes al fin y al cabo son tanto el fuente como el destino. Propone entonces la técnica del **Subspace Alignment**. Consiste en alinear los subespacios fuente \mathbf{S}_s y destino \mathbf{S}_t para que sean lo más similares posible. Esto es visto que es sencillo, llegando a una solución cerrada detallada a continuación. El problema de optimización tiene como objetivo aprender una matriz \mathbf{M} tal que se minimice la norma Frobenius siguiente:

$$\mathbf{M}^* = \operatorname{argmin}_{\mathbf{M}} \|\mathbf{S}_s \mathbf{M} - \mathbf{S}_t\|_F^2 = \mathbf{S}_s^T \mathbf{S}_t$$

Todos los enfoques de DA basados en subespacios tratan de conseguir que los subespacios fuente y destino sean similares. Esto es bastante efectivo, pero realmente no ataca el problema del DA de manera directa. Recordemos que el problema original era la discrepancia entre las distribuciones de los datos. Por ello, a continuación se

ven algunos trabajos que sí tratan el problema directamente intentando alinear las distribuciones. Una de las medidas más utilizadas a lo largo de los años en DA es el **Maximum Mean Discrepancy** (MMD). Fue introducido por Gretton et al. en 2012 en el trabajo [27].

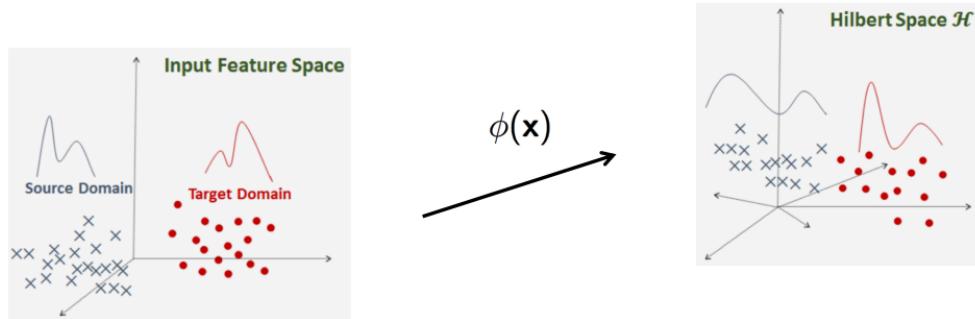


Figura 5: Esquema del funcionamiento de MMD [3]. El espacio de características de entrada se mapea a un espacio de Hilbert de dimensión alta, y es ahí donde se realiza la diferencia que se toma como medida de distancia de ambas distribuciones.

MMD compara las medias de dos conjuntos de muestras en un espacio de Hilbert de dimensión alta. Un esquema se puede ver en la figura 5, y la fórmula del MMD es la siguiente:

$$D_{MMD} = \left\| \frac{1}{n} \sum_{i=1}^n \phi(x_s^i) - \frac{1}{m} \sum_{j=1}^m \phi(x_t^j) \right\|$$

Una forma de usar MMD para DA es lo que se presenta en [28] y que se conoce como **Sample Reweighting**. En él se asigna un peso a las muestras fuente. Otra opción se propone en [32], y se trata del **Sample Selection**, que selecciona parte de las muestras fuente. Para ello, la fórmula del MMD tiene en este caso pesos binarios, indicando si la muestra se considera o no, y una restricción para asegurar la misma proporción de muestras seleccionadas en cada clase.

Cuando la diferencia entre las distribuciones es demasiado grande, las dos técnicas anteriores no son suficientes, y por ello se han propuestos otras soluciones dentro del marco del **Transformation Learning**. La idea es aprender un *mapping* a un espacio latente donde las distribuciones son similares. El primer trabajo de esta técnica es [30], introduciendo en él el **Transfer Component Analysis**. Se aprende un mapping no lineal que minimiza el MMD, lo que equivale a aprender una matriz kernel. El problema así planteado no era suficiente para dar con una solución, y hacían falta más restricciones. Entonces se reformuló el problema, dando lugar a un nuevo planteamiento cuyo esquema se puede ver en la figura 6.

Sin embargo, en la propuesta anterior el MMD no se computa en el espacio de Hilbert, como inicialmente estaba definido. Por ello, en [31] el objetivo es aprender una representación en el espacio latente tal que el MMD se minimice. En esencia, se invierte el orden que veíamos antes entre el espacio latente y el del Hilbert.

Los métodos que se han visto hasta ahora hacían uso del MMD para medir la discrepancia entre las distribuciones. Ahora se verán propuestas que usan otros estadísticos

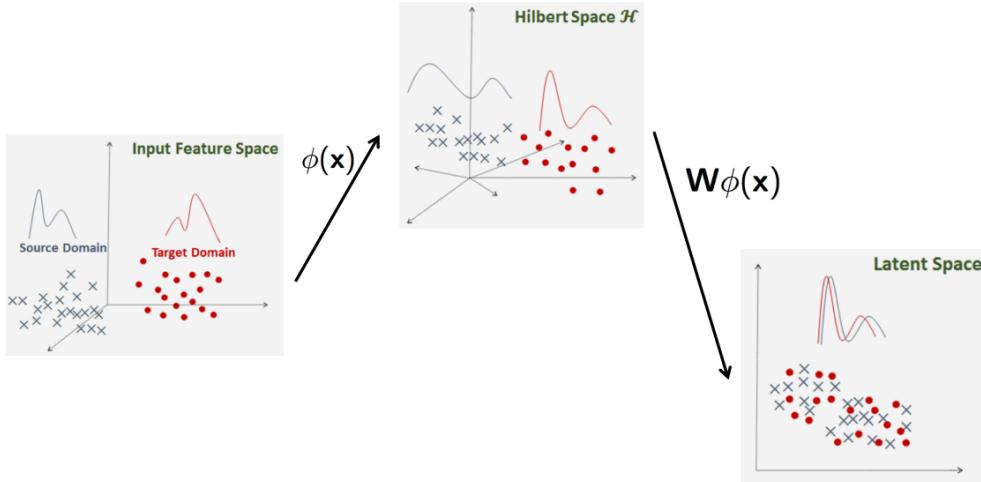


Figura 6: Esquema de la reformulación del problema de Transfer Component Analysis [3]. La muestra de datos de entrada se mapea a un espacio de Hilbert de dimensión alta, y a continuación se aplica un mapping lineal acabando en el espacio latente destino. Los parámetros de la matriz \mathbf{W} se aprenden mediante un problema de minimización.

distintos para este fin. [33] presenta CORAL, que emplea matrices de covarianza como métrica central. Inicialmente, descorrelaciona las características del dominio fuente para eliminar las dependencias lineales entre ellas. Luego, las recorrelaciona en base a la estructura de correlación del dominio destino. Matemáticamente, esto se traduce en el siguiente problema de minimización:

$$\min_A \|A^T C_S A - C_T\|_F^2$$

En esencia, se trata de encontrar una matriz A tal que al multiplicarla por la matriz de covarianza del dominio fuente C_S (dándose así una correlación débil), esté cerca de la matriz de covarianza del dominio destino C_T . Un detalle importante que diferencia CORAL de MMD es que las matrices de covarianza se computan en el espacio original, a diferencia de las medias en MMD que se hacía en el espacio de Hilbert.

Otra alternativa a CORAL o MMD son las f-divergencias. Las distribuciones en este caso se pueden estimar mediante estimación de densidad de kernels (KDE). Un ejemplo de f-divergencia es la conocida Kullback-Leibler divergence. Otro es la distancia de Hellinger. En [34] se propone el uso de la distancia de Hellinger para los problemas de sample selection y transformation learning vistos anteriormente.

Los métodos poco profundos son muy interesantes pues introdujeron los primeros conceptos y técnicas para tratar el DA. Sin embargo, todas ellas veían su rendimiento acotado, y era necesario algo más para llegar a un potencial mayor. Por ello se empezaron a proponer técnicas de deep learning para tratar el DA, y éstas se resumen en el apartado siguiente.

2.1.3. DA clásico: métodos profundos

2.1.3.1. Enfoques posibles

A lo largo de los años se han visto muchos artículos distintos tratando el DA mediante

deep learning. A grandes rasgos, todos ellos se pueden enmarcar dentro de cuatro grandes grupos. Describimos cada uno de ellos brevemente a continuación:

Una primera familia de modelos son aquellos que aplican **métodos poco profundos** de los vistos anteriormente, pero sobre **características extraídas de un modelo profundo**, conocidas como Deep Convolutional Activation Features (DeCAF). Se ha visto mediante distintos experimentos que ya solo con esta modificación se consiguen resultados mucho mejores que cuando se aplican métodos poco profundos a características creadas a mano. Esto es porque el extractor de características es capaz de aprender información general e incluso reducir el domain gap en algunos casos. Algunos trabajos que estudian esta técnica son [62] y [63].

Otro conjunto de artículos usan **modelos profundos con fine-tuning**. La idea es coger un modelo profundo preentrenado en algún conjunto de datos grande y diverso, y luego aplicar fine-tuning en el dominio fuente. Finalmente se aplica dicho modelo en el dominio destino. En los distintos experimentos se ha visto que en general cuanta más profundidad tenía el fine-tuning (a nivel de capas), mejor funcionaba. La profundidad del fine-tuning depende en gran parte de la dificultad del DA, requiriendo mayor profundidad aquellos problemas con un domain gap mayor entre los dominios. [64], [65] intentan aplicar lo anterior, y destacan la importancia de un buen fine-tuning del modelo en el origen.

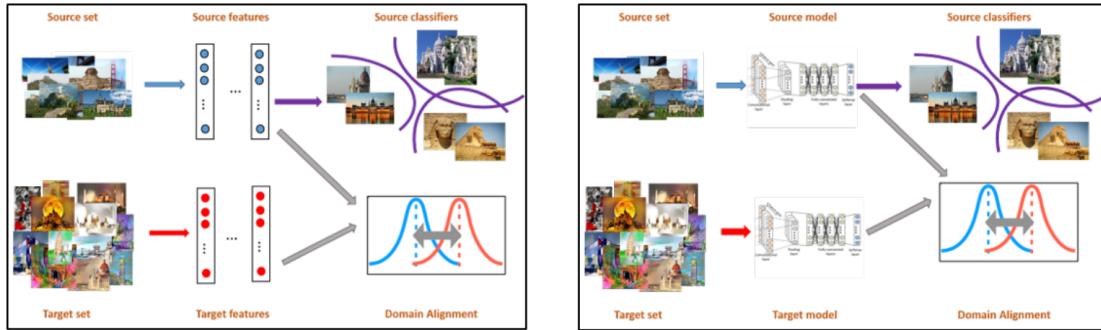


Figura 7: Esquema de la transición de modelos poco profundos a modelos profundos [3]. Nótese que la única diferencia es la parte central, donde en los modelos poco profundos se cuenta con las características, mientras que los profundos usan modelos para extraer dichas características.

Si combinamos las dos técnicas anteriores, tenemos una nueva aproximación al problema. Lo que se hace en este caso se hace un **fine-tuning del modelo profundo en el dominio fuente**, pero en vez de aplicarlo directamente en el dominio destino, se utiliza **como extractor de características**, y finalmente se aplica alguno de los **métodos no profundos**. [66] presenta DeepCORAL, que consigue mucho mejores resultados que su predecesor CORAL.

Finalmente, tenemos el **DA profundo** o *Deep DA*. Se trata de arquitecturas profundas creadas con el objetivo específico de hacer DA. Normalmente suelen inicializarse con un fine-tuning de un modelo profundo en el dominio fuente. Para pasar de métodos poco profundos a métodos profundos, esencialmente se trata de sustituir las características fuente y destino por modelos que aprendan dichas características (ver figura 7).

De todos los modelos que ha habido en las cuatro categorías comentadas, el Deep DA

es la que mejor ha funcionado en líneas generales, aunque seguido muy de cerca por el anterior, el fine-tuning del modelo profundo en el dominio fuente. A continuación nos centramos en el Deep DA y vamos viendo los distintos enfoques que se han empleado para resolver el problema.

2.1.3.2. Deep DA

Los **modelos discriminativos** están inspirados por métodos de DA clásico y tienen una arquitectura siamesa con dos flujos, uno para el conjunto de origen y otro para el conjunto de destino. Ambas pueden compartir o no los pesos, y en general se inicializan con un backbone entrenado en el conjunto fuente, usando una pérdida de clasificación de entropía cruzada. La red Siamesa luego se entrena con la misma pérdida de entropía cruzada aplicada solo al flujo fuente junto con una pérdida de alineación de dominio definida con características tanto de la fuente como del destino. Un esquema del funcionamiento de estos modelos según lo explicado se puede ver en 8.

La alineación del dominio se puede lograr minimizando la discrepancia en la distribución de características, o mediante el uso de una pérdida adversarial para aumentar la confusión de dominio. Para minimizar la discrepancia entre las distribuciones, la mayoría de los trabajos usan la pérdida MMD con función kernel [35], [36]. También se han propuesto otras alternativas como la discrepancia de momento central, [37], la pérdida CORAL, [38], o la distancia de Wasserstein [39], [40]. Por otro lado, la confusión de dominio se puede lograr mediante pérdidas adversarias como la pérdida GAN [41] y pérdida de confusión de dominio [42], o usando un clasificador de dominio y una capa de inversión del gradiente [43].

Los métodos anteriores en general comparten la misma arquitectura con los mismos pesos en ambos dominios. Esto esencialmente se traduce en intentar aprender características invariantes de dominio. Otra corriente de trabajos se podrían categorizar como **adaptación de parámetros de las redes**, e intentan adaptar las arquitecturas de ambos dominios mediante la modificación de los parámetros destino. Un par de ejemplos de ello son [44] y [45], donde se aprenden metaparámetros que adaptan los pesos y sesgos de cada capa de la red fuente a la red destino.

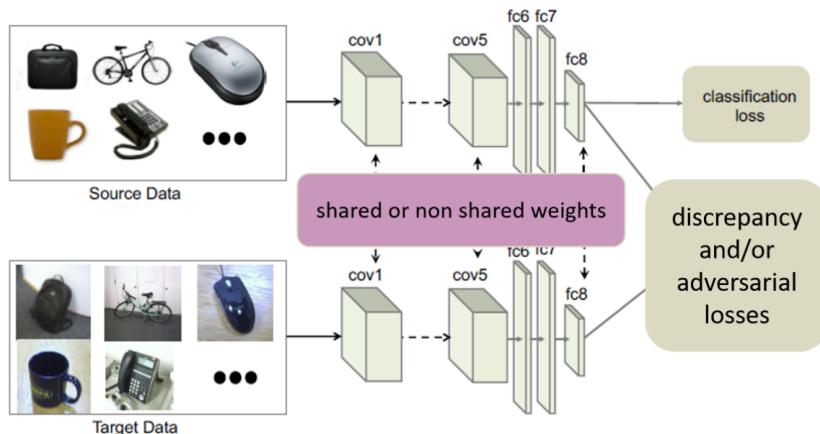


Figura 8: Esquema de los modelos discriminativos, los primeros que surgieron que usan Deep DA. [3].

Otra técnica que ha dado buenos resultados es la de **normalización batch específica de dominio**. A diferencia de los métodos anteriores, que se focalizaban en optimizar los pesos de la red, en este caso se centran en adaptar la normalización batch en cada caso para reducir el domain gap. Esto viene motivado por trabajos como [46], donde se muestra que esta técnica es equivalente a proyectar las distribuciones de características de ambos dominios en un una de referencia a través de la estandarización de las características. [47] y [48] son dos ejemplos de DA siguiendo esta tendencia. Algo más reciente es [67], donde se propone abordar las diferencias de dominio utilizando un método basado en momentum para adaptar la normalización batch e incluir una pérdida explícita para mantener su consistencia (este caso usa sin embargo la técnica source free DA).

El DA basado en **encoder-decoder** es otra familia de modelos que ha sido ampliamente estudiada. Los primeros que surgieron fueron auto-encoders profundos propuestos para PLN, y esencialmente se trataban de feedforward denoising auto-encoders apilados. En ellos una red neuronal de múltiples capas reconstruye los datos de entrada a partir de corrupciones aleatorias parciales con retropropagación. Continuando con ello, [49] demostró que dicho modelo puede ser entrenado eficientemente al marginalizar el ruido que conduce a una solución de forma cerrada para las transformaciones entre capas. También ha habido trabajos que extienden lo anterior al caso supervisado, como [50].

Por otro lado, propuestas más recientes entrena los encoder-decoder end-to-end. [51] combina una CNN estándar para la predicción de etiquetas del dominio origen con una red de deconvolución para la reconstrucción de datos destino mediante la alternancia entre entrenamiento no supervisado y supervisado. En [52] se integran tanto encoders específicos del dominio como encoders compartidos, y el modelo incorpora una pérdida de reconstrucción para un decoder compartido que depende tanto de representaciones específicas del dominio como compartidas.

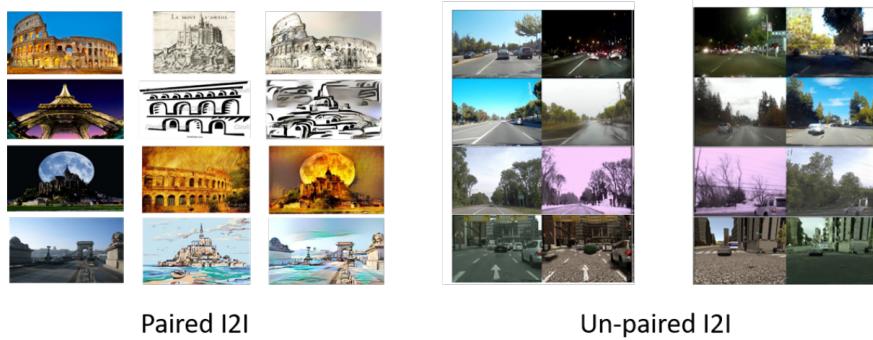


Figura 9: Imágenes que ilustran el problema tratado en las técnicas de transferencia de estilo de dominio. Las Un-paired images dan problemas a la hora de adaptar el dominio, y algunas técnicas sugieren transferir el estilo del dominio fuente al destino, dando lugar así a las paired I2I images. [3].

Otra serie de propuestas muy extendidas son las que se conocen como **transferencia de estilo de dominio**. Mientras que la mayoría de los trabajos anteriores buscaban alinear las distribuciones de los espacios de características, éstos se centran en la similitud de las imágenes entre sí. Esto surgió porque se vio que el domain shift estaba fuertemente relacionado con la apariencia de las imágenes, como por ejemplo

día o noche, o también el estilo artístico de éstas, como por ejemplo pinturas, dibujos animados, etc. Por un lado algunos artículos propusieron métodos de transferencia de estilo image-to-image I2I (ver figura 9), generando imágenes del dominio fuente pero con el estilo del destino [53], [54]. Acto seguido, se entrena el modelo de DA con estas imágenes en vez de las imágenes fuente originales.

Un conjunto de métodos diferentes son aquellos que aunque tienen el mismo destino, no son I2I. Éstos, en cambio, asumen que por debajo existe un shift de apariencia, de una manera más general. Para ello, los trabajos [60], [61] utilizan GANs. [55] extiende el trabajo anterior mediante el uso de autoencoders variacionales. En [56] se propone aprender un mapeo entre los dominios mediante una pérdida adversaria GAN a la vez que impone una pérdida consistente en ciclo, es decir, la imagen de estilo destino mapeada de regreso al estilo fuente debería coincidir con la imagen fuente original.

En [57] se combinó la consistencia de ciclo entre imágenes de entrada e imágenes estilizadas con consistencia semántica específica de la tarea, y extendió el método a la segmentación semántica. En el caso de la segmentación semántica, transferir el estilo de imágenes destino es la base de muchos métodos de DA. Algunos ejemplos son [58] y [59].

Por último, comentamos un par de técnicas que, aunque no son en sí mismas arquitecturas de deep DA, pueden aplicarse a todas las vistas hasta ahora e incrementar su rendimiento. Una de ellas es el **pseudo-etiquetado de datos destino**. El problema tiene parte de aprendizaje auto-supervisado. Para ello, se asume que al menos un subconjunto de los datos destino puede ser autoetiquetado de manera correcta. Finalmente, al juntar estas etiquetas con las del dominio fuente se consigue que el modelo de DA actúe de manera semi-supervisada.

A lo largo de los años se han propuesto diferentes estrategias para el autoetiquetado. Predicciones softmax [74] o clustering [75] son solo algunas de ellas. Otro trabajo algo más reciente, pero en la misma línea, es [68]

Para reducir el impacto de pseudo-etiquetas ruidosas durante el alineamiento de los dominios, se han explorado diversas técnicas de aprendizaje basadas en **curriculum**. La opción más usada y simple es incluir las etiquetas en orden de mayor a menor confianza. Esta confianza se determina mediante un score que se calcula mediante clasificadores de imágenes. Ejemplos que usan esta técnica son [76],[77].

2.1.4. Más allá de la clasificación de imágenes

La mayoría de los trabajos que se han visto tratan el DA aplicado a la clasificación de imágenes. Esto es lo más común, principalmente debido a su sencillez. Sin embargo, el DA no se restringe a la clasificación y por ello a continuación finalizamos el estado del arte mencionando algunas de las propuestas que ha habido tanto en la detección de objetos como la segmentación semántica.

En la **detección de objetos** se han aplicado diversas técnicas de las vistas anteriormente, localizando los objetos mediante cajas. [69] es solo un ejemplo de detección de objetos mediante DA adversario, mientras que [70] usa el pseudoetiquetado y autoaprendizaje comentados al final del apartado anterior.

Por último, la **segmentación** de imágenes también ha sido ampliamente investigada usando DA. Al igual que en el caso de detección, se han empleado técnicas adversarias

[71] y de pseudoetiquetado [72]. En este caso los modelos generativos han sido otra opción de gran popularidad, la mayoría basándose en GANs [73].

2.2. Detección de Landmarks

La detección de landmarks desempeña un papel fundamental en diversas aplicaciones como el reconocimiento facial o la detección de emociones. Consiste en identificar y localizar puntos clave en el rostro humano, como las esquinas de los ojos, la punta de la nariz o los bordes de los labios. Estos puntos de referencia proporcionan un mapa estructural robusto del rostro, permitiendo análisis o manipulaciones posteriores. Se suele realizar una vez la cara ha sido detectada en una imagen o un frame de un vídeo.

En los últimos años, el campo ha visto avances significativos gracias a la disponibilidad de grandes conjuntos de datos anotados y técnicas de aprendizaje profundo. En esta sección se revisan los primeros métodos que surgieron, así como aquellos más modernos basados en aprendizaje profundo.

2.2.1. Primeras propuestas

Los primeros enfoques para la detección de landmarks se basaban principalmente en ajustar una malla deformable de la cara. Los algoritmos más destacados en esta categoría incluyen el **Modelo de Forma Activa** (ASM), el **Modelo de Apariencia Activa** (AAM) y el **Modelo Local Restringido** (CLM) [85], [86]. Estos algoritmos calculaban las ubicaciones de los landmarks basándose en la malla obtenida, utilizando a menudo métodos estadísticos como base. Si bien estos enfoques demostraron una buena precisión de predicción en entornos controlados con iluminación adecuada y rostros frontales, su rendimiento era malo para la mayoría de las imágenes en entornos no controlados.

El siguiente conjunto de métodos que surgieron involucraban técnicas como Random Forest y Gradient Boosting, destacando el algoritmo **ERT** [87]. Estos métodos ofrecían una mayor precisión pero aún tenían dificultades con caras ocluidas, ángulos extremos o mala iluminación. En el mundo real a menudo se requiere la detección de landmarks en entornos no controlados, algo en lo que estos algoritmos todavía eran mejorables.

El algoritmo ERT mencionado pertenece a la librería Dlib [88], que es open-source. Se trata de un método basado en cascada que utiliza Gradient Boosting. Este algoritmo comienza con una plantilla de una cara y la refina iterativamente. Sin embargo, requiere la detección inicial de la cara, comúnmente utilizando el detector de caras Viola-Jones. La principal ventaja de ERT es su alta velocidad, procesando cada cara en alrededor de 1 milisegundo, y sigue siendo popular debido a su eficiencia y al hecho de ser open-source.

Recientemente, los algoritmos basados en redes neuronales han ganado popularidad y son los que más se utilizan actualmente. Han mostrado una gran mejora en los entornos no controlados, y funcionan mucho mejor en poses grandes (el concepto de pose se ve en 4.2.1 al hablar de los datasets utilizados). Lo que resta de sección se centrará en las principales técnicas de aprendizaje profundo que han sido utilizadas hasta la fecha.

2.2.2. Propuestas Modernas: redes neuronales

Los trabajos basados en redes neuronales han seguido principalmente dos enfoques: regresión de mapas de calor y regresión directa de las coordenadas. En la figura 10 se ilustran ambos métodos. De manera resumida, consisten en lo siguiente:

- **Regresión de mapas de calor:** predice un mapa de calor para cada landmark: se construye un mapa de calor 2D para cada uno y los valores en los mapas de calor se interpretan como probabilidades de que estén en dichas ubicaciones de la imagen. Normalmente, argmax o una variación de éste se utiliza para determinar las coordenadas exactas de los landmarks en los mapas de calor.
- **Regresión directa de las coordenadas:** predicen directamente las coordenadas de los landmarks. Suele ser más rápido que la regresión de mapas de calor, pero también menos preciso.

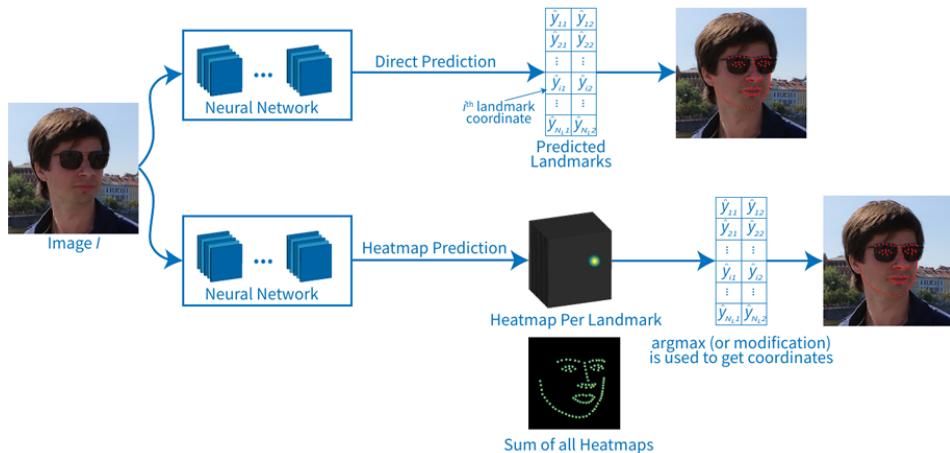


Figura 10: Esquema ilustrativo del funcionamiento de los algoritmos de detección de landmarks según el enfoque de regresión directa (fila superior) y mapas de calor (fila inferior).

La mayoría de los algoritmos se basan en una red predefinida o *backbone*. En este trabajo también nos referiremos a él como *encoder* o extracto de características. En muchos casos, los backbone para la detección de landmarks y de la estimación la pose del cuerpo son los mismos. Los métodos de regresión directa suelen utilizar backbones ampliamente conocidos de ImageNet, como ResNet, MobileNetV2, MobileNetV3 y ShuffleNet-V2. Los métodos basados en mapas de calor comúnmente utilizan la arquitectura de red Hourglass, pero también algunas menos conocidas como HRNet y CU-Net.

2.2.2.1. Algoritmos

Los algoritmos de detección de landmarks utilizan los backbone mencionados, pero con modificaciones en el entrenamiento, en la inferencia o incluso en el propio backbone. A continuación se resumen algunos de estos métodos, indicando para cada uno el backbone utilizado, el tipo de algoritmo (mapa de calor o regresión) y la idea más novedosa que propone.

Dense Face Alignment (DeFA) [89] es un enfoque *shape-model based*. Utiliza una red neuronal convolucional personalizada como backbone. Es el único algoritmo descrito en esta sección donde se usa una malla facial 3D deformable. El algoritmo es interesante porque permite construir una malla facial 3D densa utilizando solo una imagen 2D.

Style Aggregated Network (SAN) [90] y **AVS** [91] son dos métodos que usan mapas de calor. Intentan resolver un problema que había relacionado con el estilo de la imagen, donde dependiendo de la luminosidad y los colores se predecían landmarks de manera inconsistente. Se diferencian en el procedimiento que usan, pero ambos entran una GAN para cambiar el estilo de la imagen.

Look at Boundary (LAB) [92] combina los dos enfoques: mapas de calor y regresión directa. Utiliza 4 módulos Hourglass apilados y su contribución más importante fue que construía un mapa de calor de la frontera de la cara como una representación intermedia entre la imagen original y la predicción de los landmarks.

Wing Loss [93] fue la primera técnica que puso el foco en las pérdidas que se estaban utilizando. Hizo una comparativa e introdujo una nueva que combinaba la pérdida $L1$ para desviaciones de landmarks grandes y $\ln(\cdot)$ para las pequeñas y medianas. Esta pérdida fue utilizada en métodos posteriores como **AWing** [94].

Otras propuestas que merece la pena destacar son **LUVLi** [95], basado en mapas de calor y siendo la única que utiliza CU-Net como backbone, **Deep Adaptive Graph** (DAG) [96], regresión directa que utiliza redes convolucionales con grafos y destaca frente a otros algoritmos cuando las caras tienen un mayor grado de oclusión, o **PIP-Net** [97], que combina mapas de calor y regresión directa e intenta reducir el coste computacional del enfoque basado en mapas de calor, a menudo muy alto. Para ello propone usar una red con tres cabezas, que comparten el mismo backbone y pueden procesar los datos en paralelo.

Para terminar la sección, destacar que en este trabajo se utilizará una regresión directa de landmarks 3D de dos coordenadas, con una ResNet50 como backbone o extracto de características. Los datasets usados como fuente y destino son respectivamente Face Synthetics y 300W-LP, que se explican con más detalle en la sección 4.

3. Domain Adversarial Neural Network

Esta sección tiene como objetivo explicar con más profundidad la técnica de adaptación de dominio que se ha utilizado en el trabajo. La técnica que se verá fue propuesta por Ganin et al. en 2015 [2]. En primera instancia, se repasarán los conceptos claves de dicha técnica, para a continuación comentar como se ha llevado a cabo en la práctica y los experimentos que se han hecho en la siguiente sección.

3.1. Introducción

La propuesta de Ganin se conoce como Domain Adversarial Neural Network (DANN). Aborda la adaptación de dominio no supervisada, es decir, tenemos acceso a los datos y etiquetas del dominio fuente pero únicamente a los datos del dominio destino. Como ya se comentó en un apartado anterior, este es el caso más popular y, por tanto, el que ha sido más estudiado. Este trabajo centra el estudio en problemas de clasificación, aunque puede ser fácilmente extendido a otro tipo de problemas.

DANN integra el aprendizaje de las características profundas y la adaptación del dominio en un mismo proceso de entrenamiento. De esta forma, el objetivo es aprender características que sean discriminativas e invariantes al dominio del que proceden los datos. Para ello, la arquitectura DANN consta de dos clasificadores: el principal, que asigna la clase a la que pertenece la muestra, y el de dominio, que se encarga de predecir el dominio del que procede la muestra.

Ambos trabajan de manera adversaria de manera similar a lo que sería una GAN. El resultado final que se pretende es que el discriminador de dominio no sea capaz de distinguir entre dominios, a la vez que las instancias se clasifiquen con el mejor rendimiento posible. Esto indicaría que el modelo está aprendiendo las características invariantes que se mencionaban anteriormente. A continuación se ve de manera resumida la teoría existente tras la técnica.

3.2. Formulación del problema

Sea X el espacio de entrada e $Y = \{0, 1, \dots, L - 1\}$ el conjunto de posibles etiquetas. Sean D_S y D_T los dominios fuente y destino respectivamente, tratándose de distribuciones sobre $X \times Y$. Sean S y T dos muestras cogidas respectivamente de D_S y D_T^X , siendo D_T^X la distribución marginal de D_T sobre X

$$S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \sim (D_S)^n, \quad T = \{\mathbf{x}_i\}_{i=n+1}^N \sim (D_T^X)^{n'}$$

donde $N = n + n'$ es el número total de muestras. El objetivo del algoritmo de adaptación de dominio no supervisado es obtener un clasificador $\eta : X \rightarrow Y$ con un riesgo bajo en el dominio destino (aún sin tener las etiquetas de dicho dominio)

$$R_{D_T}(\eta) = \Pr_{(\mathbf{x}, y) \sim D_T}(\eta(\mathbf{x}) \neq y)$$

3.3. Divergencia entre los dominios y distancia proxy

Este trabajo sigue una tendencia popular en la adaptación de dominio. El error en el destino se acota por el error fuente más una noción de distancia entre ambos dominios. La idea es que cuando la distancia entre ambos dominios es baja, el error fuente es un

buen indicador del error en el destino. En este caso se utiliza la \mathcal{H} -divergencia. Sea \mathcal{H} una clase de hipótesis, un conjunto de clasificadores binarios $\eta : X \rightarrow \{0, 1\}$. Entonces la \mathcal{H} -divergencia se define como sigue:

Definición 1. Dadas dos distribuciones de dominio D_S^X, D_T^X sobre X , y una clase de hipótesis \mathcal{H} . La \mathcal{H} -divergencia entre D_S^X y D_T^X es:

$$d_{\mathcal{H}}(D_S^X, D_T^X) = \left| \Pr_{\mathbf{x} \sim D_S^X} [\eta(\mathbf{x}) = 1] - \Pr_{\mathbf{x} \sim D_T^X} [\eta(\mathbf{x}) = 1] \right|$$

Al ser difícil de calcular la distancia anterior, incluso en su versión empírica (ver [2]), lo que se hace es entrenar de manera supervisada un algoritmo que discrimine los dominios. En este caso las etiquetas son 0 y 1 correspondiendo a cada uno de los dominios. Finalmente se llega a una distancia “proxy” que aproxima la \mathcal{H} -divergencia.

Por último, en [2] se presenta un teorema que avala todo lo anterior. Se concluye que se debe encontrar un clasificador que tenga un riesgo bajo en ambas distribuciones, precisando para ello un balance entre el riesgo fuente y la \mathcal{H} -divergencia, puesto que si la \mathcal{H} -divergencia es pequeña, un riesgo fuente bajo implica necesariamente un riesgo destino bajo. Esto implica, en última instancia, la necesidad de encontrar un clasificador que aprenda una representación de las muestras que no permita distinguir el dominio del que proceden.

3.4. Arquitectura

Una vez justificada la teoría tras el problema, queda ver como se lleva esto a la práctica. Se crea entonces un modelo de redes neuronales que sea capaz de conseguir la adaptación de dominio. Un análisis detallado se puede ver en [2].

La arquitectura consta de tres componentes: el extractor de características, el predictor de etiquetas y clasificador de dominio. Sean $G_f(\cdot; \theta_f), G_y(\cdot; \theta_y), G_d(\cdot; \theta_d)$ las redes neuronales del extractor de características, predictor de etiquetas y clasificador de dominio respectivamente, con parámetros θ_f, θ_y y θ_d . Las pérdidas del predictor de etiquetas y de dominio se denotan respectivamente por:

$$\begin{aligned} \mathcal{L}_y^i(\theta_f, \theta_y) &= \mathcal{L}_y(G_y(G_f(\mathbf{x}_i; \theta_f); \theta_y), y_i) \\ \mathcal{L}_d^i(\theta_f, \theta_d) &= \mathcal{L}_d(G_d(G_f(\mathbf{x}_i; \theta_f); \theta_d), d_i) \end{aligned}$$

Entrenar DANN consiste entonces en optimizar la siguiente función:

$$E(\theta_f, \theta_y, \theta_d) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}_y^i(\theta_f, \theta_y) - \lambda \left(\frac{1}{n} \sum_{i=1}^n \mathcal{L}_d^i(\theta_f, \theta_d) + \frac{1}{n'} \sum_{i=n+1}^{N'} \mathcal{L}_d^i(\theta_f, \theta_d) \right)$$

Se buscan los parámetros $\hat{\theta}_f, \hat{\theta}_y, \hat{\theta}_d$ tales que

$$\begin{aligned} (\hat{\theta}_f, \hat{\theta}_y) &= \operatorname{argmin}_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d) \\ \hat{\theta}_d &= \operatorname{argmax}_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d) \end{aligned}$$

Como se puede observar, la mayor diferencia respecto a un algoritmo de aprendizaje estándar es que aquí no todos los parámetros óptimos minimizan la pérdida, sino que los del clasificador de dominio se maximizan para que así no distinga entre los dominios. El parámetro λ permite establecer un equilibrio entre la pérdida de clasificación y la de discriminación, y se explicará un poco más adelante.

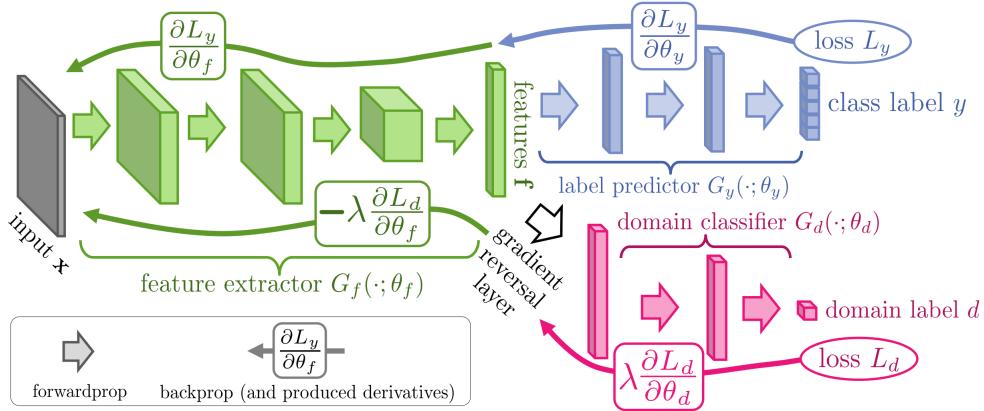


Figura 11: Arquitectura de la técnica DANN [2]. En ella se identifican de manera clara los tres componentes: extractor de características o *encoder*, predictor y clasificador de dominio o discriminador.

La arquitectura del modelo se puede ver en la figura 11. En ella se ven claramente los tres componentes mencionados, cada uno en un color. El extractor junto con el predictor de etiquetas formarían una arquitectura típica de un problema de clasificación. La diferencia se encuentra en el componente de clasificador de dominio. El extractor se conecta a dicho clasificador mediante una capa de *inversión del gradiente*, que es una de las contribuciones más importantes de este trabajo.

Durante la propagación hacia delante la capa de inversión del gradiente actúa como la transformación identidad, mientras que en la retropropagación se multiplican los gradientes por -1 y por una constante λ . Este parámetro es el que balancea el predictor y el discriminador. Cuando su valor es alto, el discriminador influye más durante el entrenamiento, mientras que cuando su valor es bajo sucede justo lo contrario. el modelo se centra más en la tarea de predecir correctamente las etiquetas. Determinar este valor es crucial para que el entrenamiento resulte satisfactorio. En la parte de experimentos se detallan los valores que se han escogido.

Volviendo a la capa de inversión del gradiente, formalmente es una pseudo-función $\mathcal{R}(\mathbf{x})$ definida por los dos siguientes ecuaciones (aunque incompatibles)

$$\begin{aligned}\mathcal{R}(\mathbf{x}) &= \mathbf{x} \\ \frac{d\mathcal{R}}{d\mathbf{x}} &= -\mathbf{I}\end{aligned}$$

donde \mathbf{I} es la matriz identidad. Este cambio de signo implica que los gradientes del predictor de etiquetas y el clasificador de dominios se resten en vez de sumarse durante la retropropagación.

Salvo esto último comentado, este proceso de entrenamiento es igual al descenso de gradiente estocástico, y por ello es sencillo de implementar. De esta forma al entrenar

se obtienen características invariantes de dominio y discriminativas al mismo tiempo, y entonces el predictor de etiquetas puede utilizarse para predecir en el dominio destino.

4. Implementación

Tras el estudio de la técnica descrita anteriormente, ésta se utilizó para, al igual que en el trabajo original, realizar la adaptación de dominio en problemas sencillos. Esto ayudaría posteriormente en la tarea de detección de landmarks. Para ello, se implementó usando pytorch la técnica DANN con imágenes de dos datasets populares: MNIST y SVHN. En este caso SVHN es el dominio fuente y MNIST el destino. Posteriormente, en el caso de la detección de landmarks el dominio fuente es Face Synthetics y el destino 300W-LP.

En esta sección se detallan las arquitecturas que se han utilizado para ambos problemas. Ambas son muy parecidas, al constar de los mismos componentes y éstos tener los mismos objetivos, con la diferencia de que la detección de landmarks es un problema de regresión en vez de clasificación. Además, para el caso de la detección de landmarks se añade además una descripción de cada uno de los datasets, el flujo de ejecución de un entrenamiento y las métricas usadas en el problema. Todo esto se omite para el caso de los dígitos por la simplicidad de dicho problema.

4.1. Conjuntos de dígitos: SVHN y MNIST

4.1.1. Arquitectura

La arquitectura utilizada está formada por tres componentes, según lo visto en 3.1.3. El extractor de características comprende un conjunto de capas convolucionales y de pooling, es decir, una arquitectura convolucional común. Tanto el clasificador de etiquetas como el de dominio están formados por capas feed forward y una última capa de acuerdo a la finalidad del componente. En el caso del clasificador de etiquetas tiene 10 neuronas de salida correspondientes a las etiquetas, mientras que en el de dominio hay una única neurona que representa la probabilidad de que una imagen pertenezca al dominio fuente, utilizando para ello una sigmoide.

El número de capas así como la configuración de los hiperparámetros se basó en las distintas implementaciones que había hasta la fecha de este método. También se probaron configuraciones propias o la propuesta en [2], aunque sin mucho éxito. Finalmente, los mejores resultados se obtuvieron utilizando la arquitectura de la figura 12 .

El extractor de características está formado por 3 bloques de 2 capas convolucionales y 1 de pooling, y finalmente una capa lineal que redimensiona la salida para que se pueda conectar correctamente tanto al predictor como al discriminador. Éstos últimos son simplemente redes feed forward de 1 capa en el caso del clasificador y 2 para el discriminador.

Las pérdidas utilizadas para el predictor de etiquetas y el de dominio fueron respectivamente CrossEntropy y BinaryCrossEntropy (BCE). A diferencia del trabajo original donde se usó descenso del gradiente estocástico junto con un learning rate scheduler personalizado, aquí el optimizador elegido fue Adam con learning rate de 10^{-4} y weight decay. Se utilizó un tamaño de batch de 64 para todos los experimentos.

El parámetro λ comentado en el apartado anterior coincide con el del trabajo original, pero sin embargo se aplica únicamente al discriminador, no al extractor de características. Éste varía de 0 a 1 gradualmente según la siguiente fórmula:

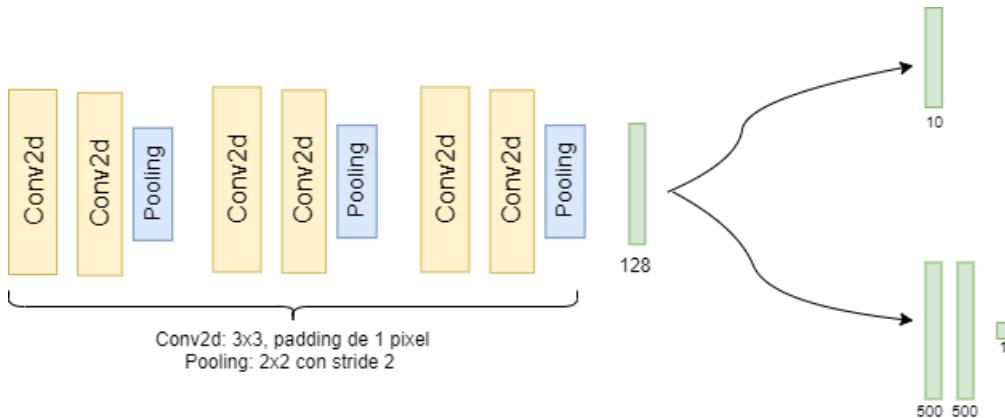


Figura 12: Arquitectura usada para la técnica DANN con los conjuntos de dígitos. El predictor consta únicamente de la capa softmax que asigna las probabilidades, mientras que el discriminador tiene 2 capas de 500 neuronas. La capa del inverso del gradiente se aplica durante la retropropagación del discriminador.

$$\lambda_p = \frac{2}{1 + \exp(-10p)} - 1, \quad p = \frac{i + \text{epoch} \cdot \text{min_len}}{\text{nepochs} \cdot \text{min_len}}$$

donde p se calcula según el número de iteraciones y la época en la que se encuentre el entrenamiento: i es el número del batch actual en el que se encuentra el entrenamiento, epoch y nepochson respectivamente la época actual y el número total de épocas del entrenamiento, y finalmente min_len es el número de iteraciones por época. Este valor es el mínimo de las iteraciones necesarias en ambos dominios, y de ahí el nombre de min_len.

4.2. Detección de Landmarks

En este caso, todo el código se encuentra integrado dentro de los repositorios github del grupo de investigación de Visión por Computador [79]. Se dispone de un repositorio centralizado que gestiona todas las distintas tareas, llamado Images_Framework, y dentro de éste se encuentra el repositorio llamado Dann16_Recognition, el que he utilizado yo. Images_Framework gestiona de manera centralizada todo lo necesario para correr los experimentos, como son los datasets, las anotaciones o la evaluación de los modelos.

4.2.1. Datasets

4.2.1.1. Face Synthetics

En [80] Microsoft presentó un nuevo conjunto de caras sintéticas con unas características nunca antes vistas hasta la fecha, el conjunto *Face Synthetics*. Otras propuestas creaban caras sintéticas con simplificaciones, o usaban adaptación de dominio [78] o entrenamiento adversario [2] para reducir el *domain gap*. Esto se debía a las distintas dificultades que había en sintetizar imágenes con un alto nivel de realismo y que además estuviesen anotadas.

Microsoft en este artículo se centró únicamente en mejorar la calidad de las imágenes sintéticas, lo que en última instancia consiguió resultados a la altura del estado del

arte en tareas como la detección de landmarks o el *face parsing*. Para ello utilizaron procedimientos de generación para crear y renderizar aleatoriamente nuevas caras 3D sin intervención manual. Primero se tomaron muestras de un modelo facial generativo 3D que captaba la diversidad de la población humana. Luego, se complementó cada rostro al azar con diferentes cabellos, ropa y accesorios.

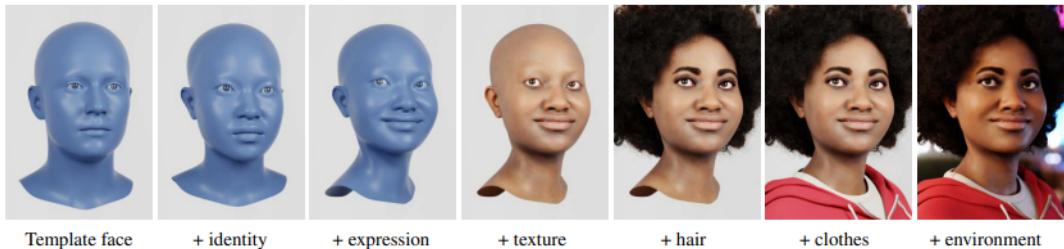


Figura 13: Proceso de síntesis de una cara, añadiendo distintos componentes al modelo 3D generado

Para el modelo 3D se utilizó un rig facial basado en *blendshapes*, siendo éstos diferentes formas o configuraciones predefinidas del modelo facial que pueden mezclarse o combinarse entre sí para crear una variedad de expresiones faciales. Una vez hecho esto, se le aplicó una expresión, textura, pelo y ropa. Por último, se renderizaba mediante Cycles, un renderizador de trazado de rayos fotorrealista. Este proceso se puede ver en la figura 13

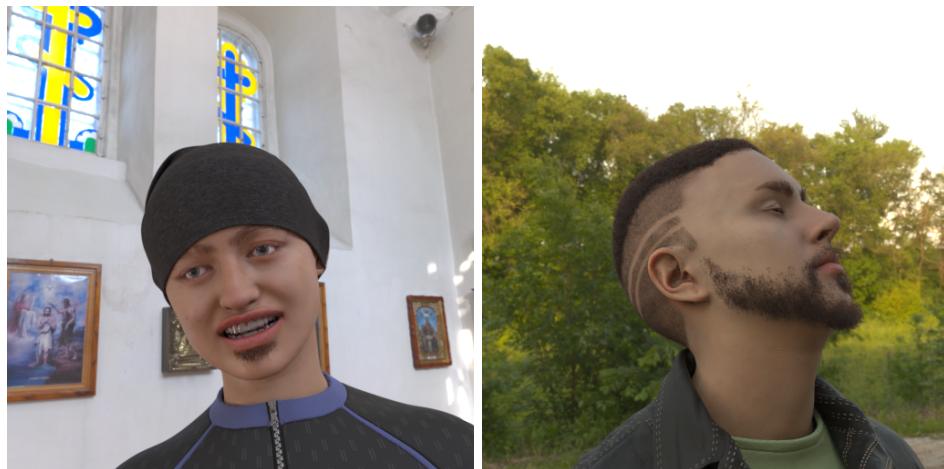


Figura 14: Imágenes del conjunto de Microsoft Face Synthetics. Al igual que 300W-LP, contiene imágenes de hasta 90º de rotación.

Una característica notable de este trabajo fue la posibilidad de generar conjuntos de datos con distinto número de anotaciones con bastante facilidad. Mientras sería imposible para un ser humano anotar todas las etiquetas manualmente, esto se hace de manera casi instantánea con el método presentado en [80]. Esto lo hace también muy útil para otro tipo de problemas aparte de face parsing y detección de landmarks, y justifica su uso en este trabajo.

El conjunto que del que se dispone está formado por 100,000 imágenes con anotaciones de 68 landmarks en el formato iBUG [81], con dos anotaciones extra para las pupilas. Se prescinde de estas dos anotaciones extra y se trabaja con las 68 anotaciones restantes.

En la figura 14 se ven dos imágenes de este dataset.

4.2.1.2. 300W-LP

El dataset *300W-LP* fue creado por Xiangyu Zhu et al. en su trabajo [83]. En él, se trató el problema de la detección de landmarks, pero mejorando lo que había hasta entonces. Antes de este trabajo, el problema de la detección de landmarks se había visto limitado al uso de imágenes con poses pequeñas o medianas. Éstas se refieren a imágenes en las que la persona está en su mayor parte de frente a la cámara, con un máximo de 45º de rotación.

Xiangyu Zhu et al. introducen entonces un framework llamado *3D Dense Face Alignment* (3DDFA) para crear imágenes en poses grandes (hasta 90º de rotación) y dan lugar al dataset 300W-LP. Lo que hacen en dicho framework es utilizar 3D Morphable Model [84] para estimar las caras 3D a partir de imágenes 2D, proyectar dichas caras de nuevo a 2D alineando los landmarks de manera precisa, y finalmente refinar las predicciones de los landmarks mediante una regresión en cascada.

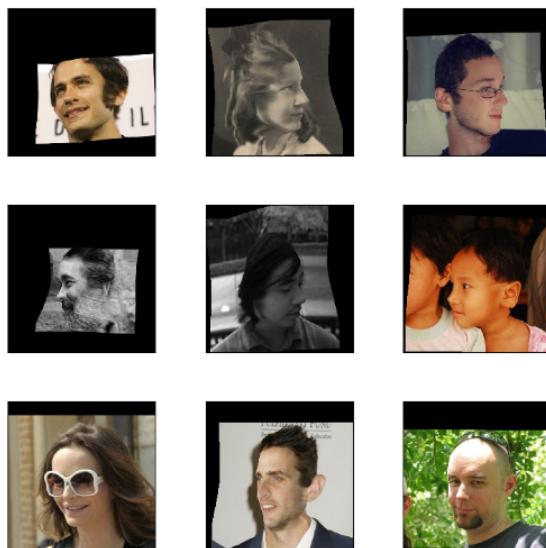


Figura 15: Imágenes del conjunto 300W-LP, creadas aplicando el 3DMM al conjunto 300W.

Con este método no solo se consiguieron resultados que mejoraron el estado del arte de entonces, basado enteramente en métodos 2D, sino que se crearon datasets más realistas gracias a las poses grandes, lo que propició que tanto la detección de landmarks y otras tareas como el reconocimiento de caras o la estimación de la pose mejorasen con el uso de estos datasets.

El dataset 300W-LP consta de alrededor de 60000 imágenes con 68 anotaciones en formato iBUG, que coinciden por tanto con Face Synthetics cuando no se consideran las anotaciones extra. En la figura 15 se pueden observar algunas imágenes de este dataset. Por último, en la figura 16 se ven una imagen de Face Synthetics y otra de 300W-LP, ambas anotadas.

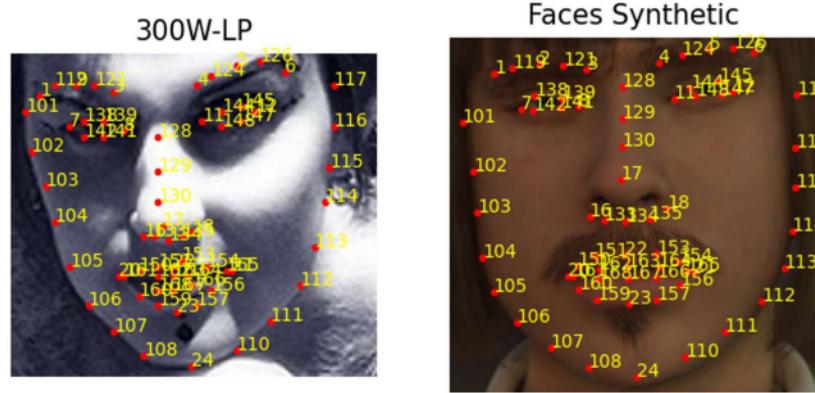


Figura 16: Imágenes anotadas de los conjuntos de datos utilizados.

4.2.2. Arquitectura del modelo

De los tipos de modelos que se vieron en 2.5, el utilizado en este trabajo es un regresor directo de las coordenadas de los landmarks. Siguiendo el esquema de la figura 11, el modelo utilizado para la detección de landmarks consta de tres componentes. El extractor de características es aquel que contiene todas las capas convolucionales para extraer las características de las imágenes que permitan aprender representaciones de ambos dominios. En este caso se ha elegido utilizar para ello una red ResNet-50 preentrenada con los pesos de ImageNet.

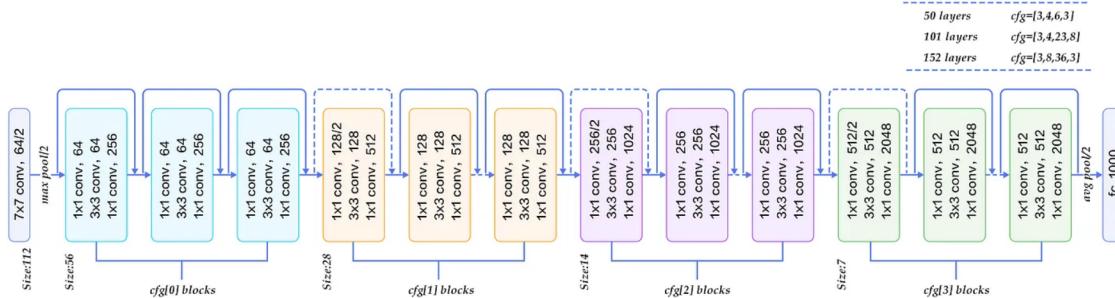


Figura 17: Arquitectura de una ResNet-50 [82]. Comparte arquitectura con las versiones ResNet-101 y ResNet-152. La única diferencia es el número de bloques convolucionales, que se puede ver en la tabla situada en la parte superior derecha.

De las que hay disponibles en pytorch es una versión de complejidad media, teniendo como particularidad que introduce cuellos de botella en sus bloques residuales. De esta forma, cada bloque residual consta de una convolución puntual 1×1 que reduce la dimensión de la entrada, una convolución 3×3 , y finalmente otra convolución 1×1 que aumenta la dimensionalidad. En la figura 17 se puede observar la arquitectura de esta red.

Por otro lado, el predictor (o, en este caso, el regresor) es una única capa fully connected que recibe las características extraídas de la ResNet y predice las coordenadas (x, y) de las 68 anotaciones de los landmarks. Finalmente, el discriminador son 2 capas fully connected de 1024 neuronas con una activación sigmoide al final que predice la probabilidad de que la imagen sea del dominio fuente o destino. Como se puede ver, es exactamente la misma arquitectura que en el caso de los dígitos en el capítulo anterior.

(ver figura 12), solo que ahora el extractor de características es una red preentrenada, ya que este problema es mucho más complejo. Por último, mencionar que el parámetro λ del discriminador es el mismo y funciona exactamente igual que en el caso de los dígitos.

4.2.3. Flujo de ejecución

En la figura 18 se puede ver un esquema simplificado del flujo de ejecución de un entrenamiento y evaluación cuando se hace adaptación de dominio. El resto de experimentos realizados (no aplicar adaptación de dominio, o evaluar en el dominio fuente además del destino) se deducen fácilmente de dicho diagrama.

En la parte de arriba de la figura se ven los datasets correspondientes a ambos dominios, con sus respectivas anotaciones en ficheros de texto. Las anotaciones de entrenamiento y de validación, éste último solo en el caso del dominio fuente, se usan para entrenar el modelo, el cual genera tres archivos pytorch correspondientes a cada uno de los tres componentes entrenados. De ellos se descarta el discriminador, y con el extractor de características y el predictor se evalúa el modelo entrenado, utilizando para ello las anotaciones de test del dominio de destino. Esto produce un fichero de texto con las anotaciones reales y las predicciones, y finalmente se calcula el NME del experimento usando dicho fichero.

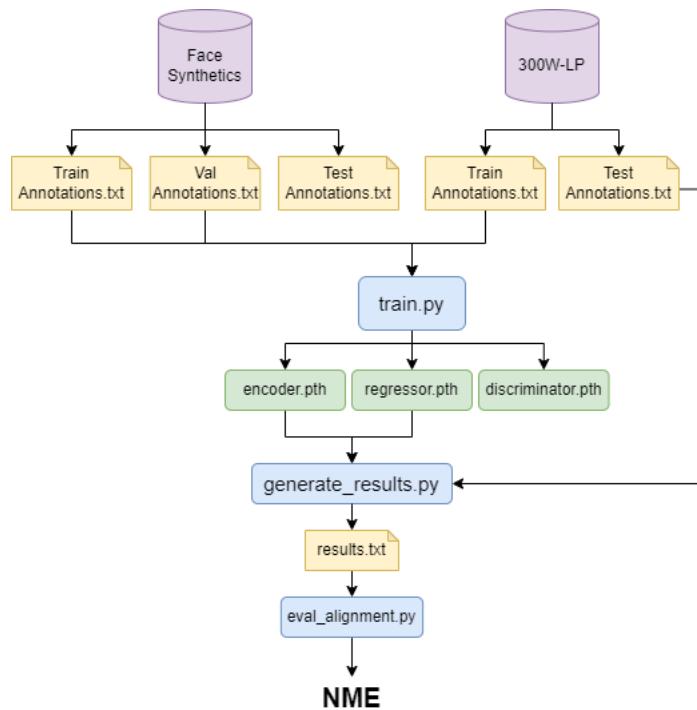


Figura 18: Flujo de ejecución simplificado del experimento DANN

4.2.4. Métricas

Para evaluar los modelos se han utilizado las tres métricas más comunes en la detección de landmarks: Normalized Mean Error, Area Under the Curve y Failure Rate. Se explican a continuación.

NME

El *Normalized Mean Error* o NME es una métrica común en problemas de regresión, que cuantifica el error promedio entre las predicciones y los valores reales, normalizado por una distancia de referencia específica para hacer que la métrica sea invariante a las distintas escalas. Esta distancia de referencia es lo que se denomina factor de normalización. La fórmula del NME es la siguiente:

$$\text{NME} = \frac{1}{N} \sum_{i=1}^N \frac{\|y_i - \hat{y}_i\|^2}{d}$$

Donde N es el número de anotaciones (68 en este caso), y_i , \hat{y}_i son respectivamente los valores reales y predichos, d es el factor de normalización, y $\|\cdot\|$ es la distancia euclídea. En el caso de la detección de landmarks, el factor de normalización suele ser la distancia interocular, definida como la distancia entre las esquinas de ambos ojos. De esta forma la métrica es consistente para diferentes tamaños de imágenes y de los rostros en ellas. Para concluir, y como es de esperar, cuanto menor sea el valor del NME mejor será el modelo entrenado. En este contexto valores aceptables del NME son aquellos que están por debajo de 10.

AUC

Area Under the Curve o AUC es una métrica muy conocida en muchos problemas de aprendizaje automático. En el caso de la detección de landmarks, funciona como sigue. Primero, para cada imagen en el conjunto de test se calcula el NME. Luego, estos valores se utilizan para trazar una curva de distribución acumulativa del error. Esta curva muestra la fracción de imágenes cuyo NME es menor o igual a un cierto umbral, con los valores de NME trazados en el eje X.

Para cuantificar el rendimiento general, se calcula el área bajo esta curva de distribución acumulativa del error (AUC). Normalmente, los valores de NME se toman dentro del rango [0, 10 %] para centrarse en los márgenes de error relevantes, donde el 10 % se refiere a la distancia de normalización. El valor de AUC calculado se escala al rango [0, 1], donde un valor más cercano a 1 indica un mejor rendimiento. Esto se debe a que un AUC más alto significa que una mayor porción del conjunto de test tiene valores bajos de NME.

FR

La métrica *Failure Rate* o FR mide la proporción de imágenes de test para las cuales el NME excede un umbral específico, generalmente establecido en el 10 %. Este umbral indica el error máximo permitido más allá del cual la predicción se considera errónea. Por tanto, un FR más bajo indica un mejor rendimiento del modelo. La expresión matemática del FR es la siguiente,

$$\text{FR} = \frac{1}{K} \sum_{k=1}^K [\text{NME}_k \geq 10\%] \times 100$$

donde K es el número de imágenes de test y NME_k es el NME para la imagen k .

5. Experimentos

Durante la realización del trabajo, se han llevado a cabo múltiples experimentos. A grandes rasgos, se pueden categorizar en tres tipos:

- **Sólo destino:** se entrena un clasificador común usando únicamente el dominio destino. En consecuencia, la precisión obtenida es la mejor de los tres experimentos e indica una cota superior para hacerse una idea de cual sería el escenario ideal.
- **Sólo fuente:** se entrena un clasificador común con el dominio fuente, y se prueba en ambos dominios. El resultado en el dominio destino es el baseline del problema, el cual se pretende mejorar aplicando la técnica DANN de adaptación de dominio.
- **DANN:** aquí entrenamos al mismo tiempo el predictor y el discriminador, implementando de esta forma DANN según lo explicado en la sección 3. Se entrena con ambos dominios, aunque las etiquetas únicamente son visibles en el dominio fuente.

En el resto de la sección se detallan los distintos experimentos y se discuten sus resultados, primero para el problema más sencillo de los conjuntos de dígitos y finalmente para la detección de landmarks.

5.1. Conjuntos de dígitos

Los datasets que se han utilizado en cada experimento para entrenar, validar y testear se pueden ver en la tabla 1. A modo de resumen, indica tanto los dominios fuente y destino utilizados, así como los resultados que se verán más adelante

Método\Dataset	Entrenamiento	Validación	Test
Solo destino	MNIST	MNIST	MNIST
Solo fuente	SVHN	SVHN	SVHN y MNIST
DANN	SVHN y MNIST	SVHN	SVHN y MNIST

Tabla 1: Datasets usados para cada uno de los métodos probados. El dominio fuente es SVHN y el destino MNIST

5.1.1. Solo destino: MNIST

Este primer experimento es trivial y como se mencionó sirve para saber cual es la precisión máxima que se podría obtener en el dominio destino, lo cual sucede cuando se entrena con las etiquetas en dicho dominio. Debido a la sencillez del problema bastaron 8 épocas para llegar a un 0.99 de precisión. En la figura 19 se puede ver las gráficas de las pérdidas y precisión de los conjuntos de entrenamiento y validación.

5.1.2. Solo fuente: SVHN

En este experimento se pretende ver como de bien funciona un modelo entrenado en el dominio fuente cuando lo probamos en el dominio destino, sin aplicar adaptación de dominio. Como se mencionaba anteriormente permite establecer un baseline para

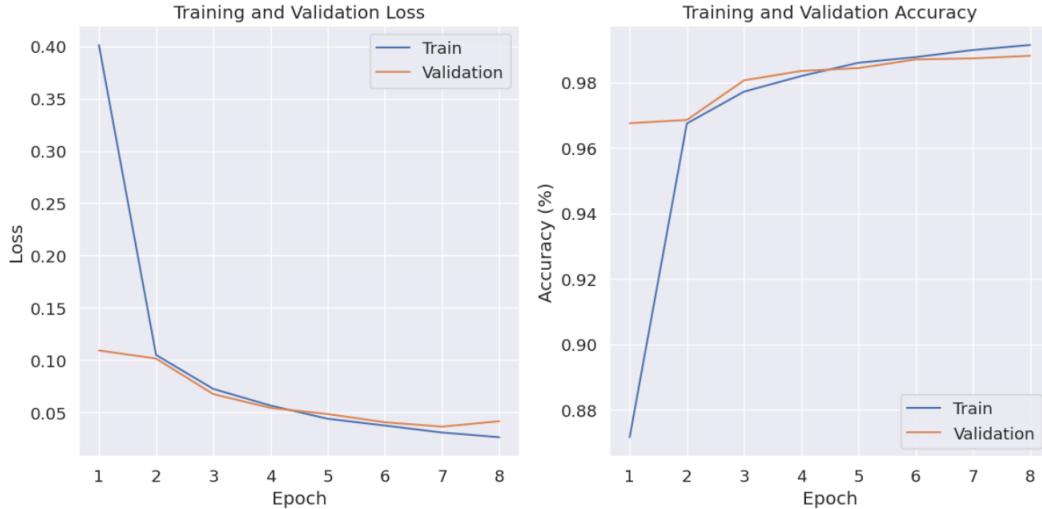


Figura 19: Gráficas de pérdida y precisión para entrenamiento y validación del experimento sólo destino en MNIST.

corroborar que efectivamente el DANN funciona y lo mejora. En este caso con 13 épocas se consiguió un 0.91 de precisión en SVHN y un 0.67 en MNIST. Es este último valor el que mejorará al aplicar DANN. Las gráficas de pérdida y precisión y las matrices de confusión se pueden ver respectivamente en las figuras 20 y 21.

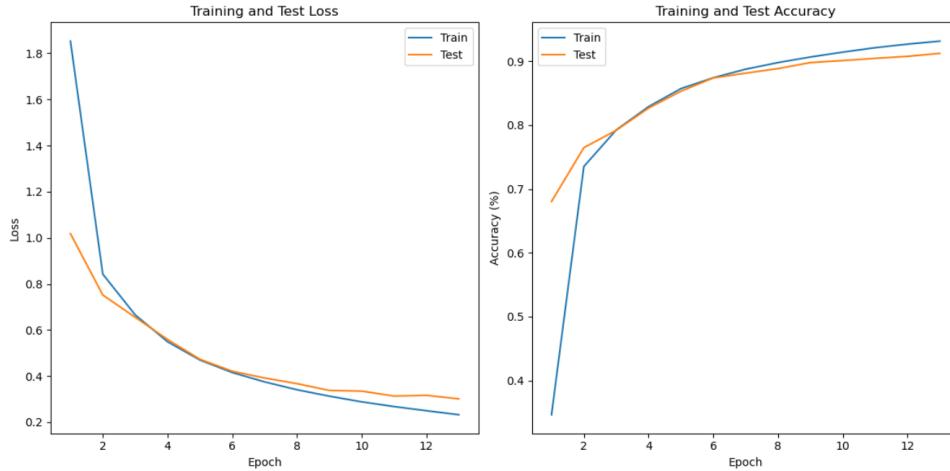


Figura 20: Gráficas de pérdida y precisión para entrenamiento y validación del experimento sólo fuente en SVHN.

5.1.3. DANN

Finalmente se aplica la técnica de adaptación de dominio, consiguiendo unos resultados bastante satisfactorios. Se entrena la red con un número de épocas elevado, en concreto 250, pero sin embargo en la época 11 ya se consigue el mejor resultado posible y por ello se para el entrenamiento. Se hace de esta forma debido al parámetro λ presente en el clasificador de dominio.

Al establecer en 250 el número de épocas, λ comienza con valores bajos cercanos al 0,

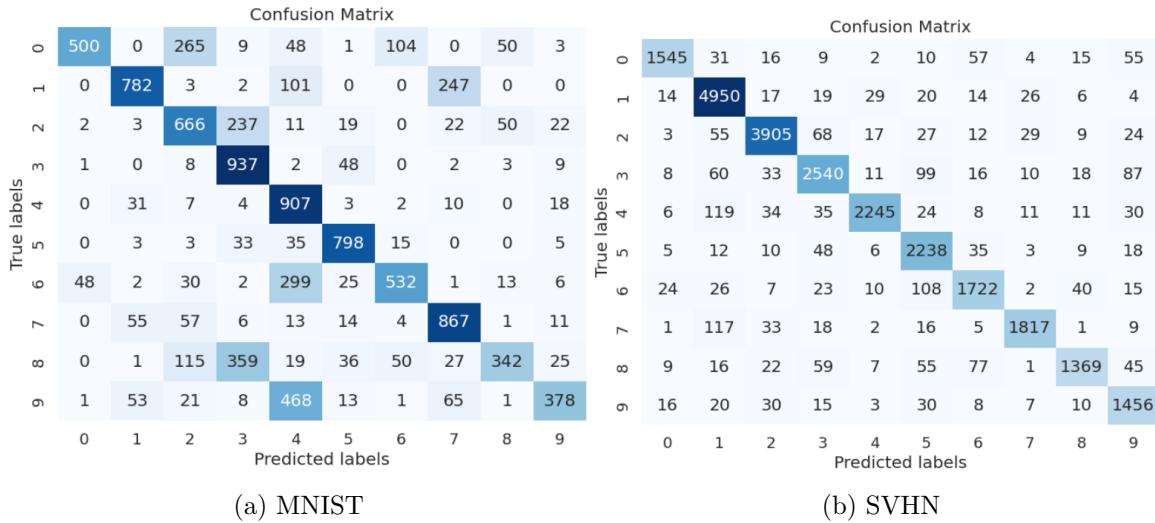


Figura 21: Matrices de confusión del experimento sólo fuente

y esto hace que tanto predictor como discriminador aprendan a un ritmo similar, sin pisarse entre ellos. Si aumentáramos λ más rápido, los dos componentes interactuarían mucho antes de haber convergido, dificultando así el entrenamiento adversario.

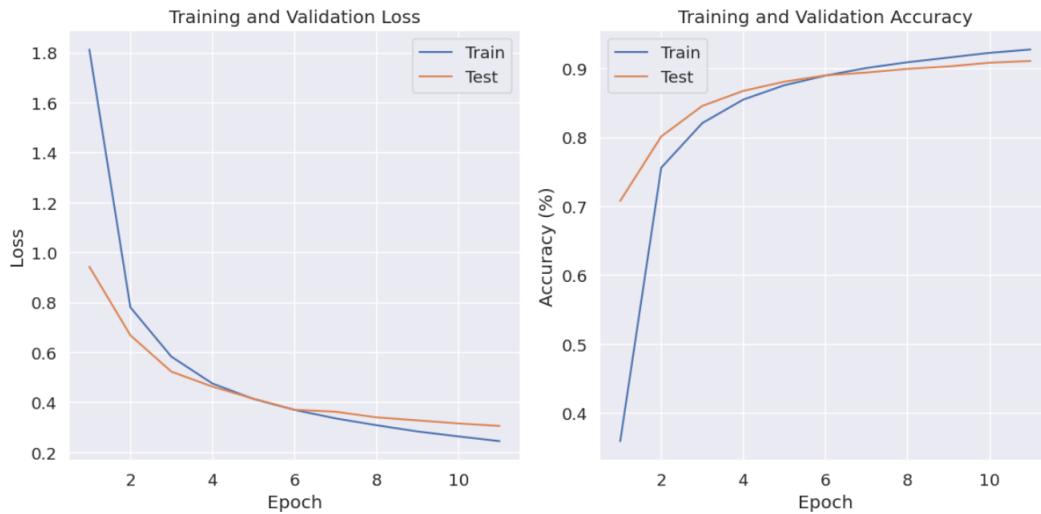


Figura 22: Gráficas de pérdida y precisión para entrenamiento y validación (en SVHN) del experimento DANN.

En la figura 22 se muestran las gráficas de pérdida y precisión en este problema. Éstas son únicamente del conjunto SVHN, porque recordemos que las etiquetas de MNIST no están disponibles en un entrenamiento DANN. Otras gráficas muy interesantes que se pueden analizar en este contexto son las de la figura 23. En ella se observan las precisiones del clasificador de dominio durante el entrenamiento, así como la pérdida de éste. Se puede ver como en las primeras épocas predice que la mayoría de imágenes son del dominio destino, pero que finalmente ambas precisiones se acercan bastante al 0.5. Éste es el valor ideal, pues indica que realmente se está dando el entrenamiento adversario al no poder distinguir el clasificador entre un dominio y otro.

Respecto a la gráfica de la pérdida, vemos que se estabiliza en un valor alrededor del

1.38. Esto de nuevo es un gran indicador de que todo está funcionando correctamente. La razón detrás del 1.38 radica en la fórmula de la pérdida utilizada. Como se mencionaba en 3.2.1, al trasladar DANN a su implementación en Pytorch la pérdida utilizada es una BCE. La fórmula de dicha pérdida promediada para N muestras es la siguiente

$$\text{BCE}_{\text{avg}} = \frac{1}{N} \sum_{i=1}^N [-y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)]$$

donde y_i, \hat{y}_i son respectivamente los valores reales del dominio y las predicciones hechas. Al utilizarse en conjunto con una activación sigmoide, las predicciones incorrectas no son 0 o 1, sino que se acercan más bien a un valor en torno al 0,5. Aproximando a este valor las predicciones incorrectas se tiene lo siguiente

$$\begin{aligned}\text{BCE}(1, 0,5) &= -(1 \cdot \log(0,5) + 0 \cdot \log(0,5)) = -\log(0,5) = \log(2) \\ \text{BCE}(0, 0,5) &= -(0 \cdot \log(0,5) + 1 \cdot \log(0,5)) = -\log(0,5) = \log(2).\end{aligned}$$

Finalmente, si sustituimos estos valores en la expresión de BCE_{avg} llegamos al siguiente valor

$$\text{BCE}_{\text{avg}} = \frac{1}{N} \left(\frac{N}{2} \cdot 0 + \frac{N}{2} \cdot \log(2) \right) = \frac{1}{N} \cdot \frac{N}{2} \cdot \log(2) = \frac{\log(2)}{2} \approx 0,693.$$

Como la pérdida mostrada en la gráfica es la suma de las pérdidas de ambos dominios, se tiene que el valor ideal es 1.38 aproximadamente. Por último, en la figura 24 se encuentran las matrices de confusión de ambos dominios, y se nota una gran mejoría en el caso de MNIST respecto al experimento de sólo fuente.

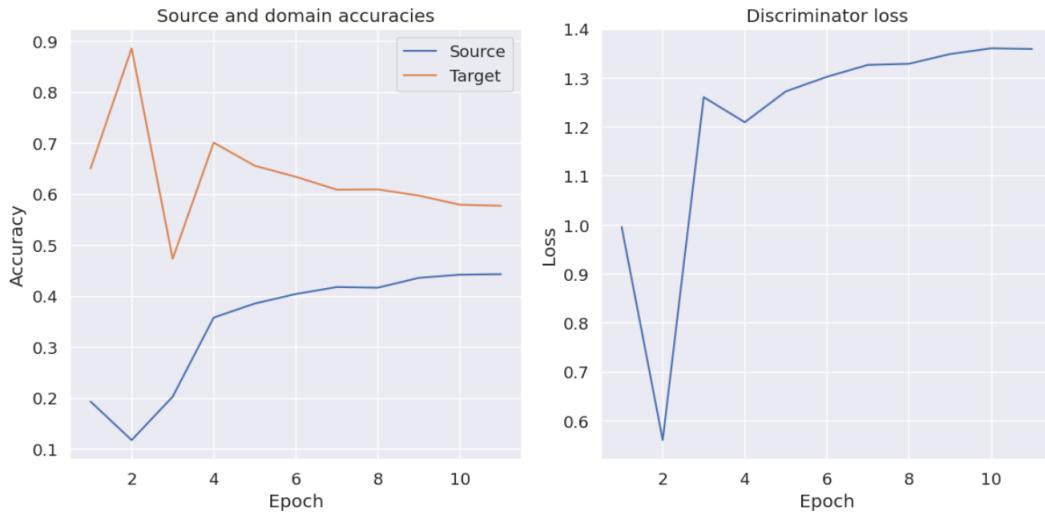


Figura 23: Gráficas de precisión y pérdida para el clasificador de dominio.

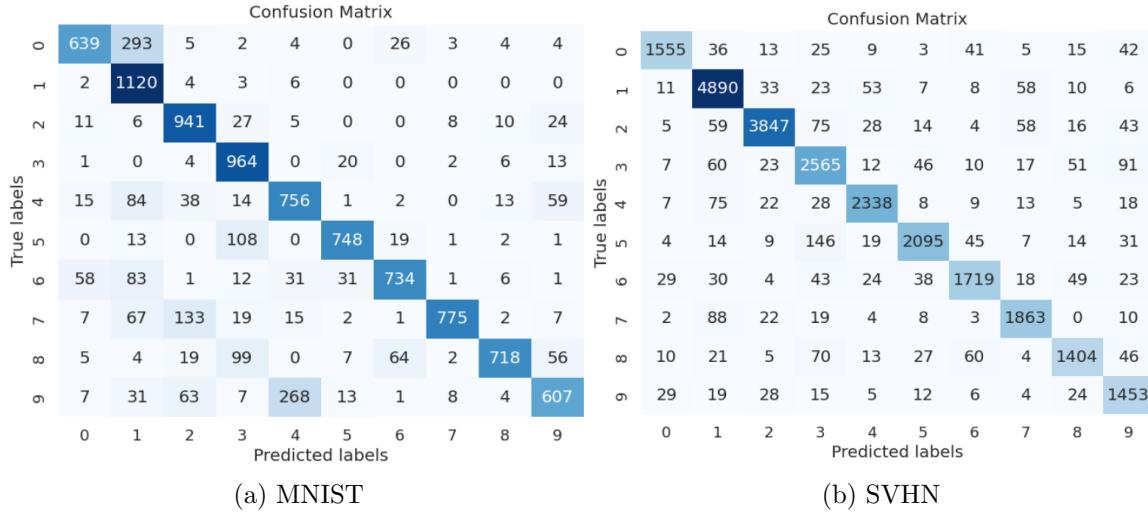


Figura 24: Matrices de confusión del experimento DANN

Método\Accuracy	MNIST	SVHN
Solo destino	0.99	-
Solo fuente	0.67	0.91
DANN	0.80	0.91

Tabla 2: Precisión obtenida para cada uno de los métodos probados. El dominio fuente es SVHN y el destino MNIST

5.2. Detección de landmarks

Ahora se ven los experimentos para el problema más interesante del trabajo, la detección de landmarks. En el caso del DANN no se han llegado a tener resultados satisfactorios. Aún así se muestra todo lo intentado y se argumenta que es lo que podría estar pasando, para que en un futuro se puedan obtener mejores resultados. Al igual que en el caso de los dígitos, en la tabla 3 se ven los datasets utilizados en los distintos experimentos

Método\Dataset	Entrenamiento	Validación	Test
Solo destino	300W-LP	300W-LP	300W-LP
Solo fuente	Face Synthetics	Face Synthetics	Face Synthetics y 300W-LP
DANN	Face Synthetics y 300W-LP	Face Synthetics	Face Synthetics y 300W-LP

Tabla 3: Datasets usados para cada uno de los métodos probados. El dominio fuente es Face Synthetics y el destino 300W-LP

5.2.1. Solo destino: 300W-LP

El primer experimento que se hizo fue entrenar y evaluar en 300W-LP. Este experimento era el más sencillo debido a que ya se le había dedicado tiempo anteriormente y se disponía de un modelo entrenado que ya funcionaba bien. Los resultados de este experimento ayudan a saber cual es el mejor resultado que se podría tener.

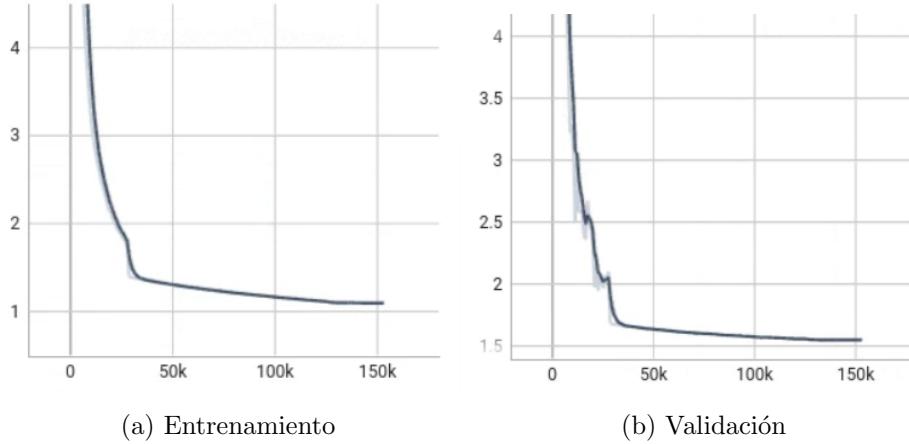


Figura 25: Gráficas de pérdida en el entrenamiento sin DANN en 300W-LP. El eje x representa el número de iteraciones, equivalente a 176 épocas. La curva en azul oscuro está suavizada al usar tensorboard, mientras que la real es la que está por debajo en un tono mucho más suave.

En la figura 25 se ven las gráficas correspondientes al experimento que mejores resultados dio. Se entrenaron alrededor de 170 épocas. Esto se debe a que este dataset es más complejo que el de Face Synthetics, entre otras razones porque estaba formado por imágenes reales, no sintéticas.

5.2.2. Solo fuente: Face Synthetics

Lo siguiente fue entrenar en el dominio fuente, y ver que resultados se conseguían tanto en él como en el dominio destino. En la figura 26 se muestran las gráficas de las pérdidas del mejor entrenamiento conseguido. Nótese que en este caso se añade también la pérdida de validación en el dominio destino 300W-LP. Esto se hace simplemente por monitorización, para saber en qué valores se mueve este modelo, que es el que interesa que sea bajo al hacer la adaptación de dominio.

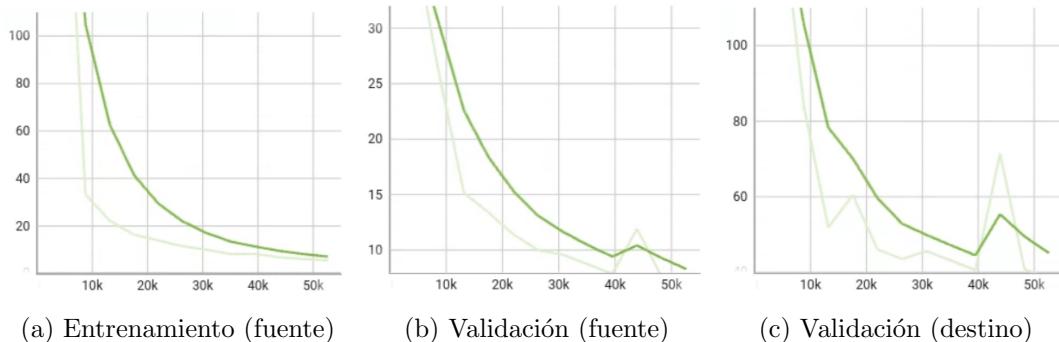


Figura 26: Gráficas de pérdida en el entrenamiento sin DANN en Face Synthetics. Entre paréntesis se indica el dominio. El eje x representa el número de iteraciones, equivalente a 15 épocas. La curva en verde oscuro está suavizada al usar tensorboard, mientras que la real es la que está por debajo en un tono mucho más discreto.

Del entrenamiento se concluye que se necesitaron muchas menos épocas para entrenarlo. Con 15 épocas ya se conseguían resultados comparables a los de 300W-LP. Además,

se observa un domain gap significativo entre ambos dominios. La pérdida de validación en este caso no logró bajar de 40, dando por tanto como resultado una diferencia de más de 35 comparado con el entrenamiento directamente en 300W-LP que se veía anteriormente. Los resultados de las métricas en estos dos primeros experimentos se ven en la tabla 4.

Método \ Métrica	NME	AUC	FR
Solo destino (test 300W-LP)	6.19	45.58	15.42
Solo fuente (test Face Synthetics)	5.81	41.9	13.08
Solo fuente (test 300W-LP)	27.76	2.29	82.04

Tabla 4: Métricas para los entrenamientos previos a la adaptación de dominio. Al entrenar sobre el dominio fuente se sacaron las métricas para ambos dominios. Se ve claramente un rendimiento peor al evaluar en un dominio donde no fue entrenado.

5.2.3. DANN

Ahora se mostrarán los distintos experimentos que se han realizado aplicando la técnica DANN de la misma forma que se hizo en el caso de los conjuntos de dígitos. Aquí se documentan los experimentos realizados para saber qué se ha probado y discutir los resultados. Los experimentos se dividen en dos categorías: aquellos que entran todos los componentes desde 0, como en el trabajo original, y los que parten de un extractor de características y predictor preentrenados (el del entrenamiento solo fuente) e intentan poner el foco en entrenar bien el discriminador. Este segundo conjunto de experimentos se probó cuando lo primero no funcionaba.

A modo de resumen, se partió de la misma configuración del entrenamiento en los conjuntos SVHN y MNIST, y a partir de ahí se fueron probando diferentes ajustes. Lo que difirió entre los distintos experimentos realizados fue lo siguiente:

- **Optimizador:** aunque la mayoría de experimentos se realizaron con Adam, una de las cosas que se cuestionaron fue la influencia del momento. Por ello se probó a usar también RMSProp.
- **Learning rate:** valores de 10^{-3} hasta 10^{-5} .
- **λ :** el parámetro del discriminador. Un valor más alto da más peso al discriminador comparado con el predictor. Se probó a que variase de 0 a 1, como en el paper original, y también mantenerlo constante en valores como 1, 3 o 5.
- **Épocas:** se entrenaron hasta un total de 60 épocas, aunque nunca hacían falta tantas, el entrenamiento empeoraba antes de completar dicho número o se estancaba.
- **Normalización batch:** una de las razones por las que podía estar fallando el DANN era la presencia de estas capas en el backbone ResNet utilizado. La normalización batch produce una alineación implícita entre dominios, puesto que los pesos de la capa se ajustan según se tome un batch del dominio fuente o destino. Esto va en contra del DANN, donde los pesos de la red subyacente son los mismos para ambos dominios.

De todos los entrenamientos, se mostrarán únicamente los más relevantes. Por ejemplo, RMSProp no dio resultados que merezca la pena compartir, así como también algún

experimento que se hizo utilizando la pérdida Wasserstein, que ayuda a entrenar redes basadas en algoritmos adversarios, como las GAN.

Todo desde cero

Esta propuesta a priori luce más prometedora que la de partir de un modelo preentrenado, al menos ya que se sigue la técnica como originalmente estaba propuesta. En todos los experimentos del DANN se ilustran las mismas gráficas: la pérdida de entrenamiento y validación en el dominio fuente y el destino, la precisión del discriminador en cada uno de los dominios, y la pérdida del discriminador.

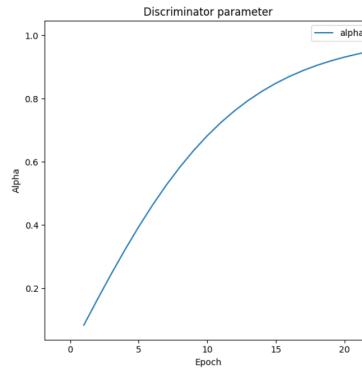


Figura 27: Valores del parámetro λ para el primer experimento realizado

El primer experimento se puede ver en la figura 28. En él el learning rate era de 10^{-3} , optimizador Adam, λ varía de 0 a 1 según la figura 27 y se pusieron 56 épocas, aunque con 22 paró por el *early stopping*. Este primer experimento es una excepción y solo muestra la pérdida de validación en el dominio fuente, el resto presenta ambas. Dicho esto, observando a las gráficas se ve como el discriminador parece que funciona y acaba llegando a una precisión que se acerca en ambos dominios a alrededor del 50 %. Sin embargo, la pérdida de validación sube desde la época 12, obteniendo por ello una alineación incorrecta de ambos dominios.

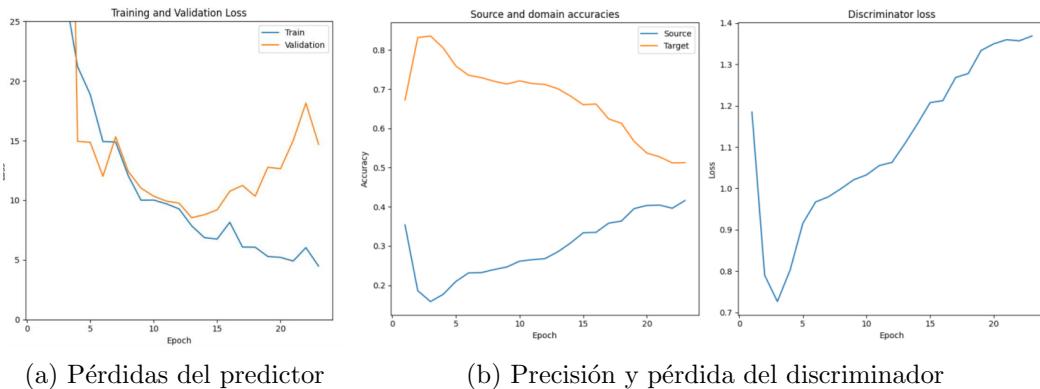


Figura 28: Gráficas de pérdida en el entrenamiento del primer experimento con DANN realizado.

El siguiente experimento simplemente fue variar el learning rate a uno más bajo, para ver como se comportaba la red en esta situación. Se probaron tanto 10^{-4} como 10^{-5} y los resultados fueron similares, ligeramente diferentes a cuando el learning rate era

de 10^{-3} . En la figura 29 se ve el caso de 10^{-4} , donde se observa un comportamiento similar en el caso de la pérdida, a partir de la época 12 o 13 sube en el dominio fuente, un poco antes en el destino. Aunque el discriminador parece funcionar, no se consigue un buen rendimiento del predictor. Esto se ve claramente en la pérdida de validación en el destino, que llega a valores de alrededor de 50, mientras que la del dominio fuente se queda en torno a 10 o 15 durante todo el entrenamiento.

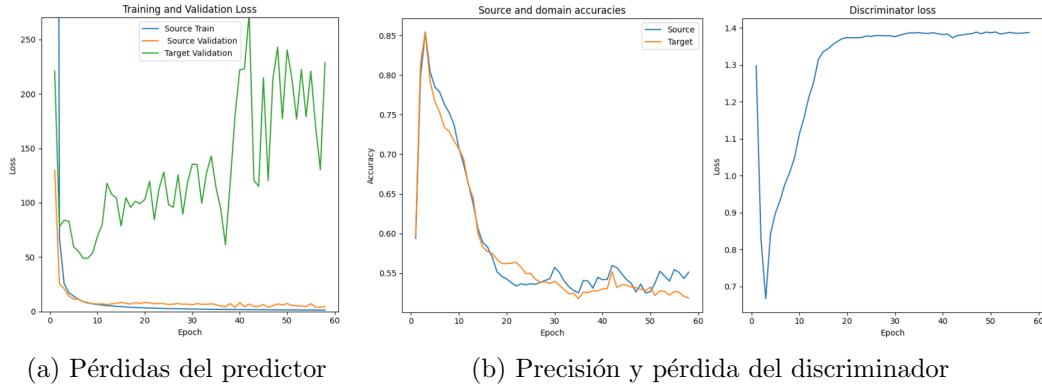


Figura 29: Gráficas de pérdida en el entrenamiento del segundo experimento con DANN realizado.

Acto seguido se volvió a dejar el learning rate a 10^{-3} , visto que no era el causante del problema. Lo que se intentó en siguientes experimentos fue hacer que el discriminador tuviese más peso desde el principio, de forma que aunque costase al principio la tarea de regresión de los landmarks, se consiguiesen juntar los dominios y conseguir un mejor resultado. Se probaron distintos valores de λ entre 1 y 5, constantes durante todo el entrenamiento, y el comportamiento obtenido fue el mismo. En la figura 30 se puede ver el caso cuando λ es 1.

Las gráficas muestran que aunque el discriminador funcione, de nuevo el domain gap es muy grande entre ambos dominios. De hecho, en este caso la pérdida está en torno a valores de 20-30 por encima de los anteriores experimentos. Esto se debe a que al empezar con un valor alto de λ desde el comienzo, no se le da oportunidad tanto al extractor como el predictor de aprender como es debido y al algoritmo le cuesta conseguir la misma pérdida que se tenía antes.

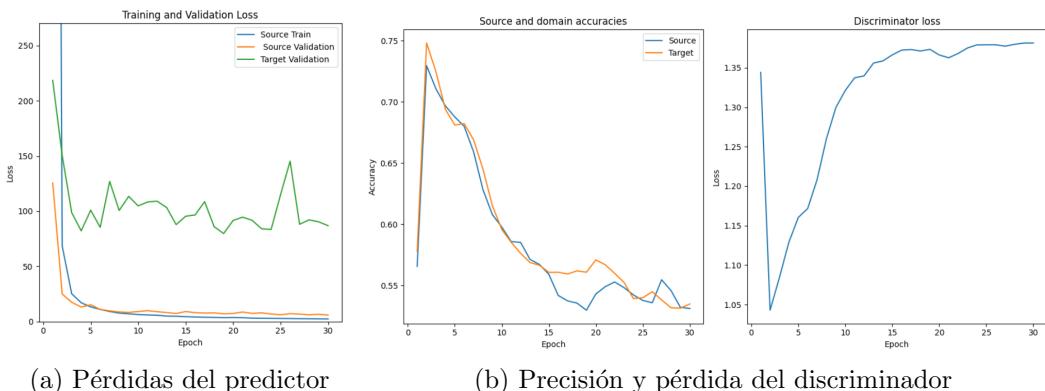


Figura 30: Gráficas de pérdida en el entrenamiento del tercer experimento con DANN realizado.

Por último, se intentó desactivar las capas de normalización batch presentes en el modelo. Algunos trabajos señalaban que la adaptación de dominio sufría debido a esto, ya que le costaba juntar los dominios [98]. Por ello, en el dominio destino se optó por desactivar dichas capas a la hora de entrenar. El resultado se encuentra en la figura 31.

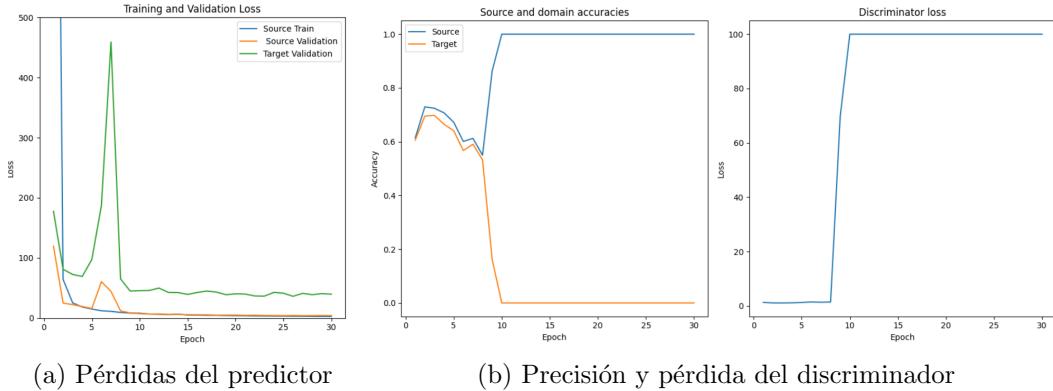


Figura 31: Gráficas de pérdida en el entrenamiento del cuarto experimento con DANN realizado.

Este intento tampoco funcionó, de hecho es peor que los anteriores. A partir de una época determinada el discriminador no puede más y es totalmente inútil. La pérdida de validación en el dominio destino es la más baja hasta el momento en los experimentos de DANN, pero realmente son los que se conseguían cuando se entrenaba sin DA. Realmente, aunque es cierto que a veces puede ser beneficioso desactivar la normalización batch, no es algo que se haya demostrado que funcione en todos los casos, pues otros trabajos sugieren que puede ayudar a estabilizar el entrenamiento [99].

Usando un modelo preentrenado

El último conjunto de experimentos que se describen son aquellos que usan el modelo que fue preentrenado en el dominio fuente. Como no funcionaba el DANN siguiendo el enfoque anterior, se pensó que teniendo un buen regresor en el dominio fuente se podría entrenar el DANN y centrarse en la tarea de discriminación, intentando reducir el domain gap de esta forma. Los experimentos son algo similar a los anteriores, es decir, se prueba a modificar los mismos parámetros, para así poder comparar los dos enfoques de manera más objetiva.

Un primer experimento que se intentó fue análogo al primero de todos los del DANN, con todos los componentes con learning rate de 10^{-3} . Ocurrió lo esperado, la pérdida del extractor de características volvió a subir bastante con respecto a lo que se tenía preentrenado y luego no conseguía bajar al mismo nivel. Se omiten en este caso las gráficas. Otro experimento posterior fue el de bajar el learning rate a 10^{-4} de los componentes preentrenados (extractor y predictor) y dejar como estaba el discriminador. Así el discriminador aprendería a un ritmo mayor con la esperanza de que se consiguiese algo de adaptación de dominio. Tampoco funcionó, como se puede ver en la figura 32, donde todas las pérdidas se mantienen igual e incluso en las últimas épocas la de validación en destino ya comienza a subir.

También se probó en este caso a desactivar la normalización batch en este caso, ocu-

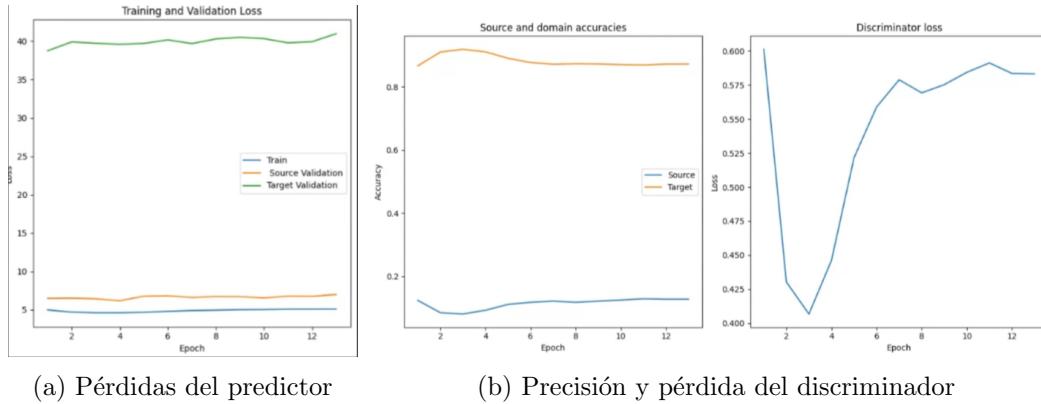


Figura 32: Gráficas de pérdida en el entrenamiento del quinto experimento con DANN realizado.

rriendo exactamente lo mismo que pasaba antes, ilustrado en la figura 31. De hecho, aquí el discriminador colapsaba mucho antes, desde la primera o segunda época. Por último, se probaron a mantener el learning rate según lo comentado anteriormente, más bajo en las partes preentrenadas, y establecer un λ constante grande. Por mostrar los resultados de alguno de estos experimentos, en la figura 33 se ve el caso en que λ es 5.

Como se puede ver, de nuevo sigue sin reducirse el domain gap, pero sí se nota que el valor del parámetro λ del discriminador es bastante alto. En vez de mantenerse constante la pérdida de validación, llega un punto que no puede más y comienza a subir. Respecto al discriminador, baja su accuracy como debe pero lentamente y muy poco, partiendo de 0.85 y llegando a 0.67 y 0.73 para los dominios fuente y destino respectivamente. En conclusión, nada de lo intentado tampoco aquí ha surgido efecto y por tiempo se ha tenido que dejar de probar otras cosas.

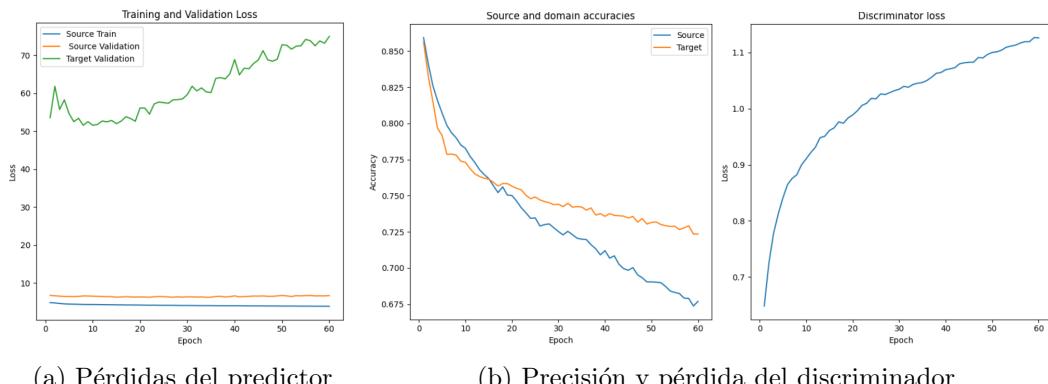


Figura 33: Gráficas de pérdida en el entrenamiento del sexto experimento con DANN realizado.

6. Conclusiones y líneas de trabajo futuras

En la última sección de este trabajo se resume lo que se ha logrado con la realización del mismo y lo que no, discutiéndose los resultados y finalmente proponiendo cuales serían los siguientes pasos que se deberían seguir a partir de aquí.

Una parte importante del TFM ha sido el estado del arte, tanto de la adaptación de dominio como de la detección de landmarks. Esto se ha conseguido de manera satisfactoria, revisando las diferentes técnicas que ha habido en ambos casos. En la adaptación de dominio, aunque casi todos los métodos recientes utilizan aprendizaje profundo, conviene tener una amplia panorámica de como ha ido evolucionando desde el comienzo, para conocer los problemas que han ocurrido y cómo se han solucionado. Este conocimiento podría ayudar en un futuro para solucionar los problemas que se encontrasen.

Por otro lado, se ha estudiado con profundidad la técnica DANN propuesta por Ganin et al. en 2015. Ésta entrena la red de manera adversaria, similar a como funciona una GAN. Además, al igual que en el trabajo original, se ha conseguido de manera satisfactoria implementar esta técnica en un problema sencillo. Escogiendo el conjunto de datos SVHN como dominio fuente y MNIST como dominio destino se ha conseguido mejorar mediante DANN la precisión en un 15 %.

Por último, la tarea más compleja e interesante era la de usar esta misma técnica en el contexto de la detección de landmarks. Teniendo ya funcionando el algoritmo en el problema anterior, se pensó que sería factible hacerlo funcionar en este escenario. Sin embargo, esta parte inadecuada ha sido la más compleja, por ser un tema de investigación abierto en la literatura. Se han intentado y documentado todos los experimentos con el objetivo de que en un futuro se pueda partir de donde se ha dejado y a ser posible mejorarlos.

Possiblemente haya varias razones por las cuales lo último ha fallado, aunque la realidad es que no se saben exactamente cuales son. Para empezar, la diferencia en complejidad entre los conjuntos de datos usados en ambos casos es bastante considerable. Además, el problema tratado en el trabajo original y que se ha implementado también aquí es de clasificación, mientras que este era de regresión. Realmente, esto no se ha demostrado que sea necesariamente un problema, pero conviene tenerlo en cuenta también.

También se han identificado otras posibles causas del mal funcionamiento del DANN al predecir landmarks, basándose en distintos artículos sobre adaptación del dominio. Una de ellas es la normalización batch, como se ya se comentó en la sección de experimentos. En este caso, hay opiniones en ambos bandos, hay quienes sugieren que es perjudicial, mientras que otros dicen justo lo contrario. Con algún experimento a priori en nuestro caso la normalización batch es beneficiosa, pero haría falta un estudio más detallado de esto para afirmarlo con certeza. Lo mismo pasa con la pérdida Wasserstein, la cual ha sido beneficiosa en muchos casos en el problema de adaptación de dominio y al entrenar GANs, pero de nuevo aquí no ha dado sus frutos.

Una última justificación es el hecho de que en ninguna de las funciones que se optimizan en el algoritmo DANN se tiene en cuenta el rendimiento del modelo en el dominio destino, lo cual implica que el error del modelo en este dominio puede crecer sin que se tomen medidas correctoras en el entrenamiento.

Para finalizar el trabajo y como conclusión final, se trata de un problema muy complejo

que además no ha sido investigado todavía. En particular, solo existe un autor que haya aplicado técnicas de adaptación de dominio a tareas relacionadas con caras [98], y entre ellas no está la detección de landmarks. Antes de atacar este problema directamente, aplicarlo a otros dominios ya estudiados y más complejos que los conjuntos de dígitos ayudaría a ir aumentando el nivel de complejidad de la tarea y ganar conocimiento apoyándose en los trabajos que ya estén hechos. Una vez se tenga una base sólida sobre la adaptación dominio y se hallan implementado diferentes técnicas en diferentes problemas se estaría más preparado para el que se ha intentado resolver en este trabajo.

Referencias

- [1] Celso M. de Melo, Antonio Torralba, Leonidas Guibas, James DiCarlo, Rama Chellappa, Jessica Hodgins, *Next-generation deep learning based on simulators and synthetic data*, Trends in Cognitive Sciences, 2021.
- [2] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Victor Lempitsky, *Domain-Adversarial Training of Neural Networks*, Journal of Machine Learning Research (JMLR), 2015.
- [3] Gabriela Csurka, Timothy M. Hospedales, Timothy M. Hospedales, Tatiana Tommasi, *Domain Adaptation for Visual Applications*, Tutorial at European Conference on Computer Vision (ECCV), 2020.
- [4] Da Li, Yongxin Yang, Yi-Zhe Song, Timothy M. Hospedales, *Deeper, Broader, Artier Domain Generalization*, Computer Vision and Pattern Recognition, 2017.
- [5] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, Silvio Savarese, *Generalizing to Unseen Domains via Adversarial Data Augmentation*, Advances in Neural Information Processing Systems (NeurIPS), 2018.
- [6] Fengchun Qiao, Long Zhao, Xi Peng, *Learning to Learn Single Domain Generalization*, Computer Vision and Pattern Recognition, 2020.
- [7] Kuan-Chuan Peng, Ziyan Wu, Jan Ernst, *Zero-Shot Deep Domain Adaptation*, European Conference on Computer Vision (ECCV), 2018.
- [8] Massimiliano Mancini, Zeynep Akata, Elisa Ricci, Barbara Caputo, *Towards Recognizing Unseen Categories in Unseen Domains*, European Conference on Computer Vision (ECCV), 2020.
- [9] Yongxin Yang, Timothy M. Hospedales Queen Mary, University of London, *Multivariate Regression on the Grassmannian for Predicting Novel Domains*, Computer Vision and Pattern Recognition (CVPR), 2016.
- [10] Massimiliano Mancini, Samuel Rota Bulò, Barbara Caputo, Elisa Ricci, *Ada-Graph: Unifying Predictive and Continuous Domain Adaptation through Graphs*, Computer Vision and Pattern Recognition (CVPR), 2019.
- [11] Massimiliano Mancini, Hakan Karaoguz, Elisa Ricci, Patric Jensfelt, Barbara Caputo, *[K]itting in the Wild through Online Domain Adaptation*, International Conference on Robotics and Automation (ICRA), 2018.
- [12] Jian Liang, Dapeng Hu, Jiashi Feng , *Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation*, International Conference on Machine Learning (ICML), 2020.
- [13] Jogendra Nath Kundu, Naveen Venkat, Rahul M V, R. Venkatesh Babu, *Universal Source-Free Domain Adaptation*, Computer Vision and Pattern Recognition (CVPR), 2020.
- [14] Xingchao Peng, Zijun Huang, Yizhe Zhu, Kate Saenko , *Federated Adversarial Domain Adaptation*, International Conference on Learning Representations (ICLR), 2020.

- [15] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, Michael I. Jordan, *Universal Domain Adaptation*, Computer Vision and Pattern Recognition (CVPR), 2019.
- [16] Zhangjie Cao, Lijia Ma, Mingsheng Long, Jianmin Wang, *Partial Adversarial Domain Adaptation*, European Conference on Computer Vision (ECCV), 2018.
- [17] Zhangjie Cao, Kaichao You, Mingsheng Long, Jianmin Wang, Qiang Yang, *Learning to Transfer Examples for Partial Domain Adaptation*, Computer Vision and Pattern Recognition (CVPR), 2019.
- [18] Jian Liang, Yunbo Wang, Dapeng Hu, Ran He, Jiashi Feng, *A Balanced and Uncertainty-aware Approach for Partial Domain Adaptation*, European Conference on Computer Vision (ECCV), 2020.
- [19] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, Tatsuya Harada, *Open Set Domain Adaptation by Backpropagation*, European Conference on Computer Vision (ECCV), 2018.
- [20] Hong Liu, Zhangjie Cao, Mingsheng Long, Jianmin Wang, Qiang Yang, *Separate to Adapt: Open Set Domain Adaptation via Progressive Separation*, Computer Vision and Pattern Recognition (CVPR), 2019.
- [21] Antonio D’Innocente, Francesco Cappio Borlino, Silvia Bucci, Barbara Caputo, Tatiana Tommasi, *One-Shot Unsupervised Cross-Domain Detection*, European Conference on Computer Vision (ECCV), 2020.
- [22] Kate Saenko, Brian Kulis, Mario Fritz, Trevor Darrell, *Adapting Visual Category Models to New Domains*, European Conference on Computer Vision, 2010.
- [23] Brian Kulis, Kate Saenko, Trevor Darrell, *What you saw is not what you get: Domain adaptation using asymmetric kernel transforms*, Computer Vision and Pattern Recognition (CVPR), 2011.
- [24] Raghuraman Gopalan, Ruonan Li, Rama Chellappa, *Domain adaptation for object recognition: An unsupervised approach*, Computer Vision and Pattern Recognition (CVPR), 2011.
- [25] Boqing Gong, Yuan Shi, Fei Sha, Kristen Grauman, *Geodesic Flow Kernel for Unsupervised Domain Adaptation*, Computer Vision and Pattern Recognition (CVPR), 2012.
- [26] Basura Fernando, Amaury Habrard, Marc Sebban, Tinne Tuytelaars, *Unsupervised Visual Domain Adaptation Using Subspace Alignment*, International Conference on Computer Vision (ICCV), 2013.
- [27] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schoelkopf, Alex Smola, *A Kernel Two-Sample Test*, Journal of Machine Learning Research (JMLR), 2012.
- [28] Vincenzo M. Salari, Haibo He, K. Barker *Adaptive SVMs for Domain Adaptation in Image Classification*, International Conference on Pattern Recognition (ICPR), 2010.

- [29] Mingming Gong, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, Bernhard Schölkopf, *Domain Adaptation with Conditional Transferable Components*, International Conference on Machine Learning (ICML), 2013.
- [30] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, Qiang Yang, *Domain adaptation via transfer component analysis*, Trans Neural New (TNN), 2011.
- [31] Mahsa Baktashmotlagh, Mehrtash T. Harandi, Brian C. Lovell, Mathieu Salzmann, *Unsupervised Domain Adaptation by Domain Invariant Projection*, International Conference on Computer Vision (ICCV), 2013.
- [32] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, Qiang Yang, *Domain adaptation via transfer component analysis*, International Conference on Computer Vision (ICCV), 2013.
- [33] Baochen Sun, Jiashi Feng, Kate Saenko *Return of Frustratingly Easy Domain Adaptation*, Association for the Advancement of Artificial Intelligence (AAAI), 2016.
- [34] M. Baktashmotlagh, M. Harandi, B. Lovell, M. Salzmann, *Domain adaptation on the statistical manifold*, Computer Vision and Pattern Recognition (CVPR), 2014.
- [35] H. Shen, S.-I. Yu, Y. Yang, D. Meng, A. Hauptmann *Unsupervised video adaptation for parsing human motion*, European Conference on Computer Vision (ECCV), 2014.
- [36] M. Yamada, L. Sigal, M. Raptis, *No bias left behind: Covariate shift adaptation for discriminative 3d pose estimation*, European Conference on Computer Vision (ECCV), 2012.
- [37] B. Chidlovskii, S. Clinchant, G. Csurka, *Domain adaptation in the absence of source domain data*, Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD), 2016.
- [38] M. Yang, L. Zhang, X. Feng, D. Zhang, *Fisher discrimination dictionary learning for sparse representation*, International Conference on Computer Vision (ICCV), 2011.
- [39] G. Csurka, B. Chidlovskii, S. Clinchant, *Adapted domain specific class means*, ICCV workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV), 2015.
- [40] G. Csurka, D. Larlus, A. Gordo, J. Almazan, *What is the right way to represent document images?*, CoRR, vol. arXiv:1603.01076, 2016.
- [41] M. Long, J. Wang, G. Ding, J. Sun, P. S. Yu, *Transfer joint matching for unsupervised domain adaptation*, Conference on Computer Vision and Pattern Recognition (CVPR), 2014.
- [42] H. Shimodaira, *Improving predictive inference under covariate shift by weighting the log-likelihood function*, Journal of Statistical Planning and Inference, vol. 90, no. 2, pp. 227–244, 2000.

- [43] M. Sugiyama, S. Nakajima, H. Kashima, P. v. Buenau, M. Kawanabe, *Direct importance estimation with model selection and its application to covariate shift adaptation*, Annual Conference on Neural Information Processing Systems(NIPS), 2008.
- [44] Artem Rozantsev, Mathieu Salzmann, Pascal Fua, *Beyond Sharing Weights for Deep Domain Adaptation*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2018.
- [45] Artem Rozantsev, Mathieu Salzmann, Pascal Fua, *Residual Parameter Transfer for Deep Domain Adaptation*, In Computer Vision and Pattern Recognition (CVPR), 2018.
- [46] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, Bohyung Han, *Domain-Specific Batch Normalization for Unsupervised Domain Adaptation*. In Computer Vision and Pattern Recognition (CVPR), 2019.
- [47] Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, Qi Tian, *Towards Discriminability and Diversity: Batch Nuclear Norm Maximization Under Label Insufficient Situations*. In Computer Vision and Pattern Recognition (CVPR), 2020.
- [48] Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, Elisa Ricci, *Unifying Predictive and Continuous Domain Adaptation through Graphs*. In Computer Vision and Pattern Recognition (CVPR), 2019.
- [49] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, Fei Sha, *Marginalized Denoising Autoencoders for Domain Adaptation*. In International Conference on Machine Learning (ICML), 2012.
- [50] Gabriela Csurka, Boris Chidlovskii, Stephane Clinchant, Sophia Michel, *Unsupervised Domain Adaptation with Regularized Domain Instance Denoising*. In European Conference on Computer Vision Workshop (ECCV), 2016.
- [51] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, *Deep Reconstruction-classification Networks for Unsupervised Domain Adaptation*. In European Conference on Computer Vision (ECCV), 2016.
- [52] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dumitru Erhan, Dilip Krishnan, *Domain Separation Networks*. In Advances in Neural Information Processing Systems (NeurIPS), 2016.
- [53] Sun Huang, Serge Belongie, *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization*. In International Conference on Computer Vision (ICCV), 2017.
- [54] Yijun Li, Ming-Yu Liu, Xuetong Li, Ming-Hsuan Yang, Jan Kautz, *A Closed-form Solution to Photorealistic Image Stylization*. In European Conference on Computer Vision (ECCV), 2018.
- [55] Ming-Yu Liu, Thomas Breuel, Jan Kautz, *Unsupervised Image-to-Image Translation Networks*. In Advances in Neural Information Processing Systems (NeurIPS), 2017.

- [56] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A, Efros. *Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks*. In International Conference on Computer Vision (ICCV), 2017.
- [57] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, Trevor Darrel, *CyCADA: Cycle-Consistent Adversarial Domain Adaptation*. In International Conference on Machine Learning (ICML), 2018.
- [58] Wei-Lun Chang, Hui-Po Wang, Wen-Hsiao Peng, Wei-Chen Chiu, *All about Structure: Adapting Structural Information across Domains for Boosting Semantic Segmentation*. In Computer Vision and Pattern Recognition (CVPR), 2019.
- [59] Yanchao Wang, Dong Lao, Ganesh Sundaramoorthi, Stefano Soatto, *Phase Consistent Ecological Domain Adaptation*. In Computer Vision and Pattern Recognition (CVPR), 2020.
- [60] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, Dilip Krishnan, *Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks*. In Computer Vision and Pattern Recognition (CVPR), 2017.
- [61] Yaniv Taigman, Adam Polyak, Lior Wolf, *Unsupervised Cross-domain Image Generation*. In International Conference on Learning Representations (ICLR), 2017.
- [62] Tatiana Tommasi, Novi Patricia, Barbara Caputo, Tinne Tuytelaars, *A Deeper Look at Dataset Bias*. In Gabriela Csurka (editor), *Domain Adaptation in Computer Vision Applications*, Advances in Computer Vision and Pattern Recognition, pages 95–114. Springer, 2017.
- [63] Baochen Sun, Jiashi Feng, Kate Saenko, *Return of Frustratingly Easy Domain Adaptation*. In Association for the Advancement of Artificial Intelligence (AAAI), 2016.
- [64] Sumit Chopra, Suhrid Balakrishnan, Raghuraman Gopalan, *DLID: Deep Learning for Domain Adaptation by Interpolating Between Domains*. In International Conference on Machine Learning workshop (ICML), 2013.
- [65] Brian Chu, Vashisht Madhavan, Oscar Beijbom, Judy Hoffman, Trevor Darrell, *Best Practices for Fine-tuning Visual Classifiers to New Domains*. In European Conference on Computer Vision workshop (TASK-CV), 2016.
- [66] Baochen Sun, Kate Saenko. *Deep CORAL: Correlation Alignment for Deep Domain Adaptation*, In European Conference on Computer Vision workshop (TASK-CV), 2016.
- [67] X. Liu, F. Xing, G. El Fakhri, J. Woo, *Memory Consistent Unsupervised Off-the-Shelf Model Adaptation for Source-Relaxed Medical Image Segmentation*, in Medical Image Analysis, 2022.
- [68] C. Wei, K. Shen, Y. Chen, T. Ma, *Theoretical Analysis of Self-training with Deep Networks on Unlabeled Data*, arXiv preprint arXiv:2010.03622, 2021.
- [69] F. Yu, D. Wang, Y. Chen, N. Karianakis, T. Shen, P. Yu, D. Lymberopoulos, S. Lu, W. Shi, X. Chen, *SC-UDA: Style and Content Gaps Aware Unsupervised Domain Adaptation for Object Detection*, in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2022.

- [70] Y.-J. Li, X. Dai, C.-Y. Ma, Y.-C. Liu, K. Chen, B. Wu, Z. He, K. Kitani, P. Vadja, *Cross-Domain Object Detection via Adaptive Self-Training*, arXiv preprint arXiv:2111.13216, 2021.
- [71] K. Mei, C. Zhu, J. Zou, S. Zhang, *Instance Adaptive Self-Training for Unsupervised Domain Adaptation*, in European Conference on Computer Vision (ECCV), 2020.
- [72] X. Liu, B. Hu, L. Jin, X. Han, F. Xing, J. Ouyang, J. Lu, G. E. Fakhri, J. Woo, *Domain Generalization Under Conditional and Label Shifts via Variational Bayesian Inference*, in International Joint Conference on Artificial Intelligence (IJCAI), 2021.
- [73] F. Xing, X. Liu, J. Kuo, G. Fakhri, J. Woo, *Brain MR Atlas Construction Using Symmetric Deep Neural Inpainting*, in IEEE Journal of Biomedical and Health Informatics, 2022.
- [74] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, *Asymmetric Tritraining for Unsupervised Domain Adaptation*, in ICML, 2017 .
- [75] Vivek Sharma, Naila Murray, Diane Larlus, M. Saquib Sarfraz, Rainer Stiefelhagen, Gabriela Csurka, *Unsupervised Meta-Domain Adaptation for Fashion Retrieval*, in WACV, 2020.
- [76] Subhankar Roy, Aliaksandr Siarohin, Enver Sangineto, Samuel Rota Bulo, Nicu Sebe, Elisa Ricci, *Unsupervised Domain Adaptation using Feature-Whitening and Consensus Loss*, in CVPR, 2019.
- [77] Weichen Zhang, Wanli Ouyang, Wen Li, Dong Xu, *Collaborative and Adversarial Network for Unsupervised Domain Adaptation*, Computer Vision and Pattern Recognition (CVPR), 2018.
- [78] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, R. Webb, *Learning from simulated and unsupervised images through adversarial training*, Computer Vision and Pattern Recognition (CVPR), 2017.
- [79] *Repositorio github del grupo de Visión por Computador*, Universidad Politécnica de Madrid, [En línea] Available: <https://github.com/pcr-upm>, [Último acceso: 07/17/2024].
- [80] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Matthew Johnson, Virginia Estellers, Thomas J. Cashman, Jamie Shotton, *Fake It Till You Make It: Face analysis in the wild using synthetic data alone*, Computer Vision and Pattern Recognition (CVPR), 2021.
- [81] *I. B. U. G. (iBUG)*, *Facial point annotations*, Imperial College London, [En línea] Available: <https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/>, [Último acceso: 07/04/2024].
- [82] *ResNet50*, [En línea] Available: <https://blog.devgenius.io/resnet50-6b42934db431>, [Último acceso: 07/03/2024].

- [83] Xiangyu Zhu, Zhen Lei, Xiaoming Liu, Hailin Shi, Stan Z. Li, *Face Alignment Across Large Poses: A 3D Solution*, Computer Vision and Pattern Recognition (CVPR), 2016.
- [84] V. Blanz, T. Vetter, *Face recognition based on fitting a 3D morphable model*, Pattern Analysis and Machine Intelligence, IEEE Transactions, 2003.
- [85] N.Wang, X.Gao, D.Tao, H.Yang, X.Li, *Facial feature point detection: A comprehensive survey*, Neurocomputing, 2018.
- [86] Y. Wu, Q.Ji, *Facial landmark detection: A literature survey*, Int. J. Comput. Vision, 2019.
- [87] V. Kazemi, J. Sullivan, *One millisecond face alignment with an ensemble of regression trees*, Computer Vision and Pattern Recognition (CVPR), 2014.
- [88] D. E. King, *Dlib-ml: A machine learning toolkit*, Journal Machine Learning Research, 2009.
- [89] Y. Liu, A. Jourabloo, W. Ren, X. Liu, *Dense face alignment*, International Conference on Computer Vision Workshops (ICCVW), 2017.
- [90] X. Dong, Y. Yan, W. Ouyang, Y. Yang, *Style aggregated network for facial landmark detection*, Computer Vision and Pattern Recognition (CVPR), 2018.
- [91] S. Qian, K. Sun, W. Wu, C. Qian, J. Jia, *Aggregation via separation: Boosting facial landmark detector with semi-supervised style translation*, International Conference on Computer Vision (ICCV), 2019.
- [92] W. Wu, C. Qian, S. Yang, Q. Wang, Y. Cai, Q. Zhou, *Look at boundary: A boundary-aware face alignment algorithm*, Computer Vision and Pattern Recognition (CVPR), 2018.
- [93] Z. Feng, J. Kittler, M. Awais, P. Huber, X. Wu, *Wing loss for robust facial landmark localisation with convolutional neural networks*, Computer Vision and Pattern Recognition (CVPR), 2018.
- [94] X. Wang, L. Bo, L. Fuxin, *Adaptive wing loss for robust face alignment via heatmap regression*, International Conference on Computer Vision (ICCV), 2019.
- [95] A. Kumar, T. K. Marks, W. Mou, Y. Wang, M. Jones, A. Cherian, T. Koike-Akino, X. Liu, C. Feng, *Luvli face alignment: Estimating landmarks location, uncertainty, and visibility likelihood*, Computer Vision and Pattern Recognition (CVPR), 2020.
- [96] W. Li, Y. Lu, K. Zheng, H. Liao, C. Lin, J. Luo, C.-T. Cheng, J. Xiao, L. Lu, C.-F. Kuo, S. Miao, *Structured landmark detection via topology adapting deep graph learning*, in Computer Vision ECCV 2020, Cham: Springer International Publishing, 2020.
- [97] H. Jin, S. Liao, L. Shao, *Pixel-in-pixel net: Towards efficient facial landmark detection in the wild*, International Journal of Computer Vision, 2021.
- [98] F. Kuhnke, J. Ostermann, *Domain Adaptation for Head Pose Estimation Using Relative Pose Consistency*, IEEE Transactions on Biometrics, Behavior, and Identity Science, 2023.

- [99] F. Kuhnke, J. Ostermann, *Deep Head Pose Estimation Using Synthetic Images and Partial Adversarial Domain Adaption for Continuous Label Spaces*, International Conference on Computer Vision (ICCV), 2019.