VNIVERSITAT DE VALÈNCIA

Grado en Ingeniería
Ciencia de Datos

VNIVERSITAT DE VALÈNCIA

Trabajo Fin de Grado

Detección de anomalías en imágenes, estudio del problema y aplicación de la librería Anomalib

Autor: Pablo Ferrer González

Tutores: Emilio Soria Olivas

Joan Vila Frances

enero 2022

VNIVERSITAT DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria ETSE-UV

Trabajo Fin de Grado

Detección de anomalías en imágenes, estudio del problema y aplicación de la librería Anomalib

Autor: Pablo Ferrer González

Tutores: Emilio Soria Olivas

Joan Vila Frances

TRIBUNAL

PRESIDENTE/A:

VOCAL 1:

VOCAL 2:

FECHA DE DEFENSA:

CALIFICACIÓN:

Declaración de autoría:

Yo, Pablo Ferrer González, declaro la autoría del Trabajo Fin de Grado titulado "Detección de anomalías en imágenes, estudio del problema y aplicación de la librería Anomalib" y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual. El material no original que figura en este trabajo ha sido atribuido a sus legítimos autores.

Valencia, 13 de enero de 2023

Fdo: Pablo Ferrer González

Resumen:

Abordamos el problema de detección de anomalías en imágenes industriales. Se ha establecido un marco teórico básico y se ha explicado la importancia de la arquitectura de aprendizaje profundo llamada red neuronal convolucional, así como arquitecturas más específicas como los modelos generativos. También se ha estudiado la librería Anomalib, recientemente creada y diseñada específicamente para este problema, y se han presentado sus modelos y sus características comunes, así como sus requisitos computacionales. Discutimos la importancia de establecer una métrica informativa para evaluar el rendimiento del modelo así como también proponemos varias opciones interesantes, también proponemos diferentes formas de mejorar un modelo base. Además, se ha demostrado que la librería Anomalib es útil en un entorno real y se han identificado sus limitaciones y carencias, en este sentido se han realizado implementaciones y pruebas que proporcionan una guía de uso de la librería especialmente útil para implementaciones relativamente básicas o fundamentales a la hora de abordar un problema de este tipo.

Abstract:

We address the problem of anomaly detection in industrial images. A basic theoretical framework has been established and the importance of the deep learning architecture called convolutional neural network has been explained, as well as more specific architectures such as generative models. We have also studied the recently created Anomalib library, specifically designed for this problem, and presented its models and common characteristics, as well as its computational requirements. We discuss the importance of establishing an informative metric to evaluate the performance of the model, as well as proposing several interesting options, we also propose different ways to improve a base model. In addition, it has been shown that the Anomalib library is useful in a real environment and its limitations and shortcomings have been identified. In this sense, implementations and tests have been carried out that provide a particularly useful use guide for the library, especially for relatively basic or fundamental implementations when approaching a problem of this type.

Resum:

Abordem el problema de la detecció de anomalies en imatges industrials. S'ha establert un marc teòric bàsic i s'ha explicat la importància de lárquitectura dáprenentatge profund anomenada xarxa neuronal convolucional, així com arquitectures més específiques com els models generatius. També hem estudiat la biblioteca Anomalib, creada recentment i dissenyada específicament per a aquest problema, i hem presentat els seus models i les seves característiques comunes, així com els seus requisits computacionals. Discutim la importància que suposa establir una mètrica informativa per a evaluar el rendiment del model, així com proposar diverses opcions interessants, també proposem diferents maneres de millorar un model base. A més, s'ha demostrat que la biblioteca Anomalib és útil en un entorn real i s'han identificat les seves limitacions i mancances. En aquest sentit, s'han realitzat implementacions i proves que proporcionen una guia d'ús particularment útil per a la biblioteca, especialment per a implementacions relativament bàsiques o fonamentals a l'hora de abordar un problema de aquest tipus.

Agradecimientos:

Guardo una gratitud especial a mi madre Paula y a mi abuela Palmira, por su apoyo e ilusión.

Índice general

Capítulo 1

Introducción

1.1. Prefacio

1.1.1. Motivación

La presencia de datos anómalos en todo conjunto de datos es un hecho relevante que frecuentemente requiere de un análisis exhaustivo. En la mayoría de los casos se estudia primero si estos datos se deben o no a un problema de registro, para posteriormente evaluar los efectos de eliminarlos del estudio. Una segunda situación bien distinta se da cuando estamos interesados en identificar correctamente tantas anomalías como nos sea posible, problema denominado detección de anomalías.

La principal diferencia con un problema típico de clasificación binaria reside en el desbalanceo de las muestras, lo cual imposibilita la aplicación directa de un modelo sin que esto último sea tenido en cuenta. También se da que el tipo de modelos para la clasificación desbalanceada son muy diferentes en función de si los datos son estructurados (estructura de tabla con registros para cada variable) o no estructurados (texto, audio, imágenes, etc.).

A pesar de la gran variedad de aplicaciones (sección 1.3) que tiene la detección de anomalías, en este documento nos centraremos en aquellas de la sección 1.3.6 . Es pues que la aparición y desarrollo de las redes neuronales convolucionales ha posibilitado lograr modelos que obtienen resultados mucho mejores en problemas relacionados con imágenes. A raíz de esto la detección de anomalías en imágenes está recibiendo una dedicación cada vez mayor por parte de la comunidad: buscando un consenso en lo que respecta al marco teórico, creando conjuntos de datos para aplicar y evaluar modelos, así como el desarrollo de algoritmos mejores y más robustos.

1.1.2. Objetivos

Un primer objetivo de este trabajo trata de presentar un marco teórico que sirva como punto de partida a todo aquel que quiera tratar el problema de la detección de imágenes anómalas. Por ello, a lo largo de diferentes secciones presentamos las bases comunes a todo problema de aprendizaje automático prestando especial atención en la detección de anomalías, para después profundizar en lo que respecta al caso de las imágenes. Por ejemplo, hablamos de las formas de aprendizaje y profundizamos en el aprendizaje semi-

supervisado; explicamos las redes convolucionales, pero antes introducimos el aprendizaje profundo; presentamos soluciones generales y después nos centramos en modelos específicamente diseñados para la detección de anomalías en imágenes, etc.

Por otro lado, el problema se haya ahora en la suficiente madurez como para que se disponga de conjuntos de datos de calidad que permitan el desarrollo de modelos específicamente diseñados. Así pues, en este trabajo estudiamos el conjunto de datos MVTec-AD y los modelos de la librería Anomalib, además de hacer un estudio sobre las métricas relevantes para el problema desde las más simples a aquellas más sofisticadas.

Queremos estudiar también las características de los modelos más accesibles gracias a la librería Anomalib y la forma de implementarlos. Así como presentar también las formas más comunes de mejorar un modelo de aprendizaje automático y cómo aplicarlas en los modelos de la librería, la cual a pesar de disponer de potentes funcionalidades carece de una extensa documentación dada su reciente creación.

También queremos medir la latencia de los modelos Anomalib a la hora de realizar inferencia y con ello comprobar si es viable la aplicación a un caso real. Por ello, en este trabajo se aplica uno de sus modelos a la simulación de una situación real. Dicho caso simula una cadena de montaje así como todos los sensores que serían necesarios, y el proceso se enmarca desde la fase de obtención de los datos hasta la obtención y monitorización de sus resultados.

1.1.3. Organización de la memoria

En primer lugar, presentamos las distintas formas de aprendizaje sobre conjuntos de datos así como las principales aplicaciones en las que la detección de anomalías es fundamental.

Posteriormente enumeramos los principales problemas de los conjuntos de datos desbalanceados, especialmente relevantes en el caso de la detección de anomalías en imágenes. Tras ello desarrollamos un estudio que va desde los primeros algoritmos para identificar casos anómalos hasta los modelos que constituyen el estado del arte en los problemas de identificar imágenes anómalas, pasando por una explicación cómo las redes neuronales lo han hecho posible.

En tercer lugar, discutimos las principales métricas usadas en la detección de anomalías en imágenes. Después presentamos un conjunto de datos creado específicamente para evaluar modelos en este problema, así como la propia evaluación de los modelos más avanzados. Tras ello, proponemos y evaluamos distintas vías para mejorar el resultado de los modelos.

Por último, desarrollamos la aplicación de un modelo en un caso real, emulando una cadena de producción. Para lo cual conectamos a lo que podríamos considerar un pequeño ordenador los distintos sensores necesarios para obtener y procesar los datos y resultados, proceso que llevamos a cabo completamente a tiempo real.

1.2. Principales formas de aprendizaje

Existen diferentes formas de clasificar el tipo de aprendizaje que realizan los modelos, una de las más comunes es según la cantidad de datos para los cuales conocemos el valor

de su variable que queremos predecir.

1.2.1. Supervisado

Los modelos se entrenan con conjuntos de datos cuyas variables dependientes tienen valores conocidos, es decir, las muestras están etiquetadas y son las variables independientes las que tienen por objetivo predecir en base a los valores de las etiquetas según el modelo que se decida usar [1].

1.2.2. Semi-supervisado

Los modelos de aprendizaje automático no sólo requieren de una gran cantidad de datos para ser entrenados, sino que también es necesario que la variable dependiente completamente etiquetada para lograr mejores resultados.

En el paradigma del aprendizaje semi-supervisado una significativa parte de las muestras no están etiquetadas. Esto se debe a que la disponibilidad de datos etiquetados es menor que los que no lo están. Así pues, en el caso de que una porción del conjunto de datos esté etiquetada, los algoritmos semi-supervisado aprovechan esta situación para etiquetar tantas muestras como sea posible. Las principales formas de conseguirlo son:

- Label propagation [2]: etiqueta el resto de los casos basado en sus similaridades locales y dada la estructura global de los datos. En resumen, no creamos una función para el etiquetado, sino que lo hacemos en base a la vecindad, conocido como aprendizaje transductivo.
- Active learning: esta estrategia busca realizar un etiquetado más eficiente a través de conocer que muestras son las más informativas y por tanto conviene etiquetar, las cuales serán aquellas cuyas predicciones del modelo sean de menor confianza. En todo caso, esté método se caracteriza por ser un humano el que realiza el etiquetado de los casos seleccionados por el modelo [3].
- Weak supervision: es una forma de generar etiquetas utilizando información de una o más fuentes. Estas suelen ser expertos en la materia los cuales establecen unas reglas heurísticas. Esto genera una o más distribuciones condicionales ruidosas sobre los datos no etiquetados. Y el objetivo principal es aprender un modelo generativo que determine la relevancia de cada una de estas fuentes con distorsión [4],
- Transfer learning [5]: bajo esta estrategia utilizamos un modelo ya entrenado en una tarea similar, y tras entrenar el modelo con las muestras etiquetadas, realizamos un etiquetado de los casos no etiquetados.

1.2.3. No supervisado

En este caso la variable dependiente no existe o es desconocida, la estrategia de este aprendizaje tiene por objetivo encontrar perfiles característicos en el conjunto de datos utilizando únicamente las variables independientes. Estos modelos buscan qué características en común tienen los datos que pertenecen a un mismo grupo, encontrando estructuras subyacentes en nuestros datos y etiquetando las muestras en base a su situación respecto

al resto. Si bien es cierto que para ello existen una gran variedad de aproximaciones y es especialmente difícil la evaluación de cada una [6].

1.3. Principales aplicaciones

1.3.1. Ciberseguridad

Uno de las primeras aplicaciones fue la detección de intrusiones, propuesta por Dorothy Denning en 1986 [7], la cual consiste en identificar actividad maliciosa produciéndose en un sistema de computación.

A diferencia de los enfoques que buscan proteger los sistemas fijándose en la firma del ataque, los modelos de detección de anomalías resultan más eficientes y reconocen la intrusión o malware en una etapa más temprana del ataque [8]. En los últimos años se puede apreciar un auge de soluciones basadas en modelos de aprendizaje profundo (Deep Learning), como por ejemplo el uso de Autoencoders [9].

1.3.2. Fraude

El fraude consiste en engañar a un sistema con el objetivo de adquirir recursos valiosos independientemente de la industria. El principal reto para los sistemas de detección es que se requiere identificar y actuar frente al fraude en tiempo real [10].

Según el informe por parte de la empresa PwC sobre el crimen y fraude global de 2022, un cuarenta y seis por ciento de las empresas encuestadas han experimentado fraude de algún tipo en los últimos veinticuatro meses [11]. El caso más común trata de detectar si una transacción económica se adecúa al perfil del usuario, caso para el cual se están desarrollando diversos sistemas de detección. [12]

1.3.3. Medicina

El fraude en la atención médica afecta significativamente la capacidad de los programas de seguros para proporcionar una atención efectiva y asequible. Técnicas de aprendizaje automático en general y detección de anomalías en particular puede ayudar a detectar eventos fraudulentos, reduciendo de esta manera los costos e ineficiencias [13]. De hecho se calcula que en el programa estadounidense Medicare, entre un tres y diez por ciento de las reclamaciones médicas son fraudulentas lo cual supone unas pérdidas en torno a los cuarenta mil millones de euros [14].

Tradicionalmente, dada la escasez de casos respecto al total, la detección de este tipo de fraude se ha realizado mediante indicadores lo cual ha requerido una intervención manual y constante de expertos. Sin embargo, durante este último lustro se están dando exitosas aproximaciones al problema mediante técnicas de aprendizaje profundo como las redes adversariales generativas (Generative Adversarial Networks o GANs) [15] o las redes neuronales convolucionales (Convolutional Neural Networks o CNNs) [16].

En segundo lugar, la digitalización en el sector de la salud ha dado lugar ha proporcionado una cantidad de imágenes médicas, dado que las patologías constituyen una desviación del funcionamiento normal, la detección de estas anomalías es de una rele-

vancia crucial y las las Redes Convolucionales han dado lugar exitosamente a numerosas aplicaciones [17].

1.3.4. Redes sociales

En este caso las anomalías son consideradas patrones de comportamiento irregulares, a menudo ilegales, de individuos dentro de una red social; ya sea spammers, depredadores sexuales, estafadores, usuarios falsos o difusores de información falsa [18]. Sin embargo, todos los casos anteriormente mencionados tienen un comportamiento distinto y que evoluciona con el tiempo, además, el medio en el que se producen es a través de texto lo cual requiere de diseñar modelos específicos para este tipo de datos.

La detección de estos casos es crucial por el efecto que estos pueden tener en la sociedad, además, año tras año se está viendo una presencia exponencial de este tipo de anomalías [19]. Afortunadamente, se está dando un auge de modelos que consiguen la detección de estas anomalías como por ejemplo el uso de redes híbridas que combinan Redes Convolucionales y Redes Secuenciales [20].

1.3.5. Internet de las cosas

La principal característica del Internet de las Cosas supone un riesgo en si mismo, pues éste se da cuando tenemos una red colectiva de gran cantidad dispositivos conectados entre ellos mismos y la nube. El elevado número de elementos en esta red supone que por muy baja que sea la probabilidad de fallo individual o manipulación intencionada del dispositivo, en conjunto, será inevitable que se acabe produciendo.

El reporte de McKinsey [21] sitúa el impacto económico del Internet de las Cosas se sitúa entre los dos y seis trillones de dólares estadounidenses para 2025. Afortunadamente, en este campo también se está produciendo un auge de los modelos de aprendizaje profundo para la detección de sus anomalías, tales como patrones anormales de comportamiento del dispositivo, en la red a la que se conecta o en los datos de registran [22].

1.3.6. Industria

Para este ámbito nos encontramos con dos casos principales y bien diferenciados. Por un lado, la cadena de producción suele estar en continuo funcionamiento con todas sus etapas interconectadas, por lo que es de gran valor un sistema que detecte preventivamente cuando una maquina va a romperse o suponer una alteración nociva para la cadena de producción. Por otro, es también adecuado un sistema que identifique cuando se ha producido un elemento defectuoso en cualquier etapa de la cadena para que este pueda ser desechado o estudiado [23].

1.3.7. Series temporales

La dimensión temporal de estos datos requiere de un enfoque adaptado a las características del problema. En función de si el problema consiste de una o más variables supone de que se trate de una serie temporal univariante o multivariante, respectivamente.

La aparición de las Redes Neuronales Recurrentes (Recurrent Neural Networks o RRNs) y sus variantes Gated Recurrent Units (GRUs) y Long-Short Term Memory Networks (LSTMs) [24], ha permitido avances en la aplicación del aprendizaje profundo a este tipo de problemas. Sin embargo, siguen estando presentes unos desafíos ya estructurales: las anomalías no siguen un patrón, el ruido dificulta la detección, una mayor longitud de la serie temporal supone una mayor complejidad y que los modelos deben identificar las anomalías en tiempo real.

1.3.8. Video-vigilancia

En las aplicaciones de videovigilancia, los datos sin etiquetar están disponibles en grandes cantidades, imposibilitando la aplicación de modelos supervisados. Por lo tanto, se han modelado como problemas de detección de anomalías los cuales las técnicas de aprendizaje profundo están proporcionando resultados esperanzadores [25].

Supone un desafío particular el hecho de que la videovigilancia se de veinticuatro horas al día los siete días a la semana [26] así como una falta de consenso en lo que supone una anomalía en este contexto, o lo que es lo mismo, el amplio espectro de anomalías según el contexto de la videovigilancia [23].

Capítulo 2

Estado del arte

2.1. Introducción

La inspección visual automática en la cadena de producción es importante tanto para la gestión de la calidad como para reducir los gastos [27]. Este proceso tradicionalmente realizado de forma manual hoy en día se ha automatizado en una amplia variedad de sectores gracias a la popularización de diferentes sensores [28]. La aparición de técnicas que trabajan el problema en imágenes eficientemente ha permitido alcanzar resultados que posibilitan la implantación de sistemas de inteligencia artificial en casos reales, lo cual ha captado la atención y dedicación de la comunidad en los recientes años.

2.2. Retos

Sin embargo, las particularidades del problema de la detección de anomalías en imágenes presentan una serie de retos que requieren de modelos radicalmente distintos de los que aplicaríamos en un problema de clasificación común. Los cuatro retos principales que requieren de especial atención son:

1. El desbalanceo entre las clases normal y defectuoso. Pues la efectividad de los modelos y en especial los de aprendizaje profundo está limitada por la cantidad y distribución de los datos para entrenar el modelo [29].

2. La dificultad de anotar los datos. Ya que es mucho más fácil conseguir muestras de casos no defectuosos que lo contrario, por ejemplo, que se dé una anomalía y que ésta se capte bien por los sensores utilizados tiene baja probabilidad y al principio requiere de un etiquetado manual.

3. La complejidad del problema. A efectos prácticos es habitual que sea imposible especificar los distintos tipos de anomalías esperados incluso para un único problema, lo cual además suele requerir de conocimiento experto sobre el caso de estudio [30].

4. La delicadeza del problema. Dado el desbalanceo, supondrá una perturbación mayor para el modelo tanto el hecho de registrar incorrectamente las anomalías como el registrar múltiples veces algo independientemente de si es una anomalía o no. Los modelos serán también especialmente sensibles a la presencia del ruido, por lo que

deberemos eliminarlo tanto como nos sea posible realizando una elaboración cuidadosa de los datos que utilizaremos.

Históricamente, la detección de anomalías generalmente se ha llevado a cabo de manera no supervisada, abordándose como clasificación de una clase usando solo muestras normales. A partir de esta modelización, cualquier caso que se desvíe significativamente de la conocida distribución de casos nominales (casos no anómalos) se considera una anomalía.

2.3. Evolución

2.3.1. Los comienzos

El sector industrial ha estado limitado históricamente por su acotada capacidad para procesar datos de imágenes en tiempo real, debido a su complejidad y al elevado coste computacional a la hora de entrenar modelos adecuados. Como resultado, los sensores recopilaban señales simples, como las medidas del objeto o su color dominante. Esto significa que, en el caso de la detección de anomalías en imágenes, los modelos deben abordar el problema de forma indirecta, sin realizar un análisis de imágenes en sí, sino a través de otras medidas indirectas que proporcionen suficiente información

Los primeros modelos a la hora de detectar anomalías datan de principios de los 2000. Es el caso de las Support Vector Data Description (SVDD) [31] [32] que tratan de encontrar la hiperesfera más pequeña que pueda contener todas las muestras normales o el de One-Class Support Vector Machine (OCSVM) [33] [34] que separa las muestras normales de las anómalas buscando un hiperplano de máxima distancia del origen. Sin embargo, con el paso del tiempo, una mayor disponibilidad de datos y la mejora tecnológica, fueron surgiendo una amplia variedad de modelos que han ido mejorando a sus correspondientes predecesores.

Hoy en día pyOD [35] es una de las librerías más famosas para la detección de anomalías, en la que se implementa una amplia variedad de modelos desarrollados durante las últimas décadas. Una característica importante de esta librería es que los modelos que contiene permiten afrontar tanto la detección de anomalías en datos estructurados como no estructurados [36], pero es significativo que ninguno de sus modelos utiliza técnicas de aprendizaje profundo.

A continuación, explicaremos brevemente los algoritmos de pyOD que funcionan mejor a la hora de detectar anomalías en imágenes industriales según su familia de tipo de aprendizaje.

- No supervisados:

  Clustering-Based Local Outlier Factor (CBLOF) [37] asigna clusters (grupos) a las muestras y asigna la probabilidad de anomalía en base a la distancia entre los grupos asignados.

  Local Outlier Factor (LOF) [38] calcula una función de densidad de la muestra y asigna la probabilidad en base a la desviación con el resto.

- Semi-supervisados:

Extreme Gradient Boosting Outlier Detection (XGBOD) [39] extrae características de las muestras de manera no supervisada y las concatena con las características originales de la muestra, posteriormente utiliza un clasificador XGBoost para encontrar las anomalías.

- Supervisados:

  Extreme Gradient Boosting (XGB) [40] combina un ensamble de árboles [41] de clasificación con una potenciación del gradiente.

  CatBoost (CatB) [42] el cual es prácticamente igual que XGB pero propone una potenciación del gradiente distinta.

Si atendemos a la fecha en la que se publicaron estos modelos podemos extraer ciertas conclusiones. CBLOF (2003) y LOF (2000) utilizan técnicas estadísticas ya conocidas por la comunidad; por otro lado, XGBoost (2016), XGBOD (2018) y CatB (2018) mejoran los resultados mediante implementaciones más sofisticadas de árboles de decisión los cuales fueron originalmente propuestos en 1986 [41]. Así pues, vemos como la librería pyOD presenta una importante carencia de modelos basados en el Aprendizaje Profundo en general y Redes Convolucionales en particular, las cuales son actualmente conocidas por trabajar muy bien con datos de tipo imagen.

2.3.2. La era del Deep Learning

Es fácil enmarcar que la revolución del aprendizaje profundo (Deep Learning) comenzó cuando la red AlexNet [43] ganó la competición ImageNet [44] en 2012. Aunque es importante remarcar los tres factores que contribuyeron a este logro: los datos masivos (Big Data), el uso de GPUs y las redes convolucionales profundas.

En 2009 se popularizó la aplicación de GPUs para realizar cálculos algebraicos ha supuesto una importante mejora a la hora de aplicar algoritmos que puedan aprovechar su arquitectura, en algunos casos suponiendo que se ejecuten hasta setenta veces más rápido [45].

Por lo que respecta a los datos masivos, en 2009 se publicó un conjunto de datos compuesto por un poco más de tres millones de imágenes de cinco mil conceptos distintos aproximadamente [46]. Dado que las redes neuronales necesitan de gran cantidad de datos para lograr buenos resultados, este conjunto de datos posibilitó a la comunidad experimentar con estas arquitecturas.

En 2010 se dio la primera competición ImageNet (ImageNet Large Scale Visual Recognition Challenge) [44] cuyos datos eran de ciento cincuenta mil imágenes y mil conceptos. Para hacernos una idea, el ganador de ese año erró un 28 % del top 5 (significa que, entre los cinco conceptos más probables según el modelo, ninguno es correcto). Fue en la competición de 2012 con la arquitectura de AlexNet [43] cuando se presentó el primer modelo que hacía uso de técnicas que hoy conocemos como aprendizaje profundo. Dicha arquitectura consiguió porcentaje de error del 15.3 % mientras que el segundo puesto de aquel año tenía una tasa del 26.2 %.

Cabe destacar que el aprendizaje profundo tuvo un desarrollo paulatino construido bajo avances significativos como: Deep Beliefs Nets [47], dropout [48], implementación eficiente del descenso por gradiente (backpropagation) y activaciones ReLU [49].

Las tres arquitecturas de aprendizaje profundo más importantes a día de hoy son:

- Modelos neuronales multi-capa: compuestos por capas cuyos nodos están interconectados. Las operaciones son las de una regresión lineal junto a una función de activación no lineal. Aprenden a través del descenso por gradiente y su aplicación se da en datos estructurados donde los problemas son de clasificación o regresión.

- Modelos convolucionales: compuestos por capas, pero destaca el uso de kernels (o filtros) para encontrar relaciones de vecindad en los datos y reducir la carga computacional. La optimización de los parámetros de los kernels se realiza por descenso de gradiente, y su aplicación obtiene buenos resultados cuando es conveniente tener en cuenta relaciones espaciales en los datos como, por ejemplo, en imágenes.

- Modelos secuenciales: Los nodos de la arquitectura multi-capa no solo tiene en cuenta las nuevas entradas sino también las salidas de épocas anteriores. Surgen de la necesidad de tener en cuenta la relación temporal de los datos. Sus aplicaciones son variadas: audio, texto, vídeo, series temporales, etc. El modelo principal es Recurrent Neural Network (RNN) a partir del cual dos variantes han cobrado mayor relevancia: Long-Short Term Memory networks (LSTM) y Gated Recurrent Units (GRU).

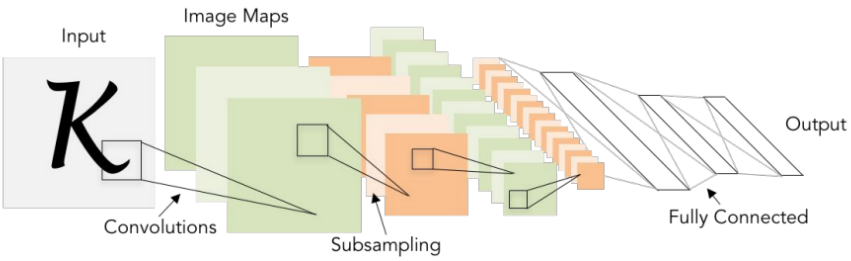

Figura 2.1: Resumen de la arquitectura de una red neuronal convolucional

El hecho de que las redes convolucionales profundas trabajan muy bien con imágenes se debe a que son capaces de extraer información local y global de una manera muy eficiente. Ello ha dado pie a una ola de modelos que buscan encontrar mejores soluciones a los problemas relacionados con imágenes, siendo la detección de anomalías uno de ellos. Antes de abordar la detección de anomalías cabe destacar varias arquitecturas que, al igual que las redes convolucionales, son comúnmente utilizadas a la hora de desarrollar algoritmos para la detección de imágenes anómalas.

En lo que respecta a nuestro problema, dos modelos generativos han sido de especial importancia: el autoencoder y la red adversarial generativa (GAN). Los modelos generativos funcionan aprendiendo a modelar la distribución de los datos de entrada, una vez que han aprendido a modelar esta distribución, pueden generar nuevos datos que sigan la misma distribución, pero que no necesariamente formen parte del conjunto de datos de entrada. Brevemente, las características de estas arquitecturas son:

- Autoencoder [50]: una red neuronal no supervisada que consiste en dos partes. Un encoder que aprende a comprimir eficientemente los datos en un espacio de características al que se le llama espacio latente, y un decoder que aprende a como recomponer una representación eficiente de los datos. Con este proceso pretende reconocer la información relevante de los datos y desechar la que no los son, como

por ejemplo el ruido. Además, son también capaces de detectar desviaciones significativas en los datos cuando las nuevas muestras no son parecidas al conjunto con el que se le ha entrenado.

- GAN [51]: es un modelo generativo que utiliza dos redes neuronales entrenadas en conjunto, una llamada generadora y otra llamada discriminadora. La generadora trata de crear datos realistas a partir de un conjunto de datos de entrada y la discriminadora trata de distinguir entre los datos creados por la generadora y los datos reales. La generadora se entrena de forma no supervisada, mientras que la discriminadora se entrena de forma supervisada. El objetivo final de la GAN es que la generadora genere datos que parezcan reales y engañen a la discriminadora.

Todos estos avances en el aprendizaje profundo han posibilitado mejores formas de extraer información relevante de los datos para solucionar una amplia variedad de problemas. Su excepcional buen funcionamiento en imágenes ha propiciado un auge en la búsqueda de nuevos modelos para solucionar problemas tales como la detección de anomalías.

En este sentido, se han construido librerías en las que se implementan gran cantidad de estos nuevos modelos. Entre ellas destaca la librería de Anomalib [52] que en contraposición a la librería pyOD, todos los modelos hacen uso de técnicas aprendizaje profundo en algún punto de su arquitectura. Además, Anomalib no sólo facilita la implementación de los algoritmos sino que proporciona herramientas para construir modelos personalizados así como configurar los existentes.

2.4. Algunos de los modelos que constituyen el estado del arte

Todos los modelos que constituyen el estado del arte tienen una prominente característica común y es el uso del aprendizaje transferido, más conocido como Transfer Learning [5]. Se caracteriza por el uso de una red pre-entrenada con objeto que esta actúe como extractora de características. Estas redes ya han sido entrenadas sobre grandes conjuntos de imágenes y en sus capas intermedias albergan información, de nivel más bajo ó más alto según la capa. Esto hace que no sea necesario dedicar grandes recursos computacionales ni disponer de grandes conjuntos de datos para el entrenamiento de redes convolucionales complejas.

Dadas las particularidades del problema, aún no se ha conseguido un consenso en cuanto a qué arquitectura es mejor para resolverlo. Si bien es cierto que la capacidad para capturar información global de las redes convolucionales ha posibilitado modelos que no solo detectan la presencia de la anomalía (clasificación) sino que son también capaces de localizarla en la imagen (segmentación). Por todo ello, a continuación describiremos brevemente los principales modelos que, a día de hoy, han proporcionado mejores resultados.

2.4.1. PatchCore

Parte de la idea de que una imagen puede ser clasificada como anómala tan pronto como una parte de ésta sea identificada como anómala. A estas partes las denomina parches, y una vez dividida en parches entiende la imagen como un mosaico de éstos.

Los parches se introducen en una red convolucional pre-entrenada y con objeto de extraer características de sus capas intermedias. Esto se debe a que las capas bajas aportan información semántica demasiado ambigua, mientras que las altas suelen estar sesgadas aportando información demasiado especifica y dependiente al conjunto de datos con el que ha sido pre-entrenada.

Las características extraídas se almacenan en un banco de memoria que tiene en cuenta la vecindad de las características de cada parche. Durante la inferencia, este banco se submuestrea de forma determinista mediante la técnica *greedy coreset* buscando el subconjunto que mejor representa al conjunto, permitiendo una aproximación de la solución y reduciendo posteriores costes tales como los asociados con la búsqueda del vecino más cercano [53].

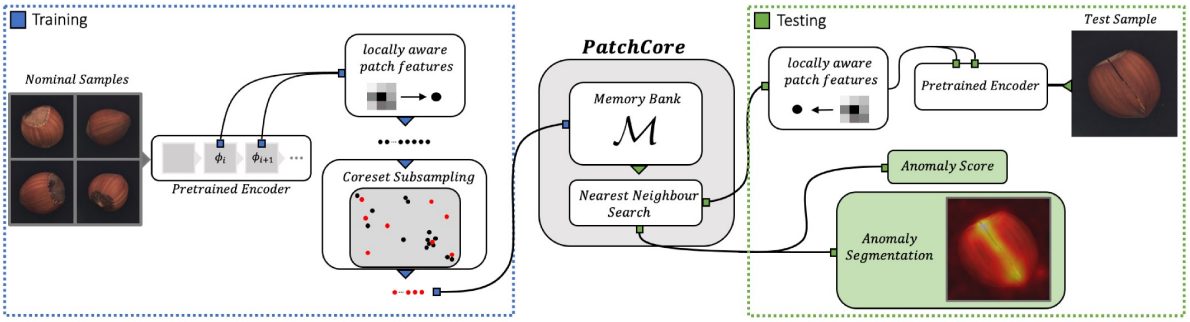

Figura 2.2: Resumen de la arquitectura PatchCore.

2.4.2. FastFlow

Se extraen características de las imágenes de entrada a partir de una una red neuronal convolucional profunda pre-entrenada. Luego se utilizan flujos normalizadores para aprender transformaciones entre distribuciones de datos con el fin de mapear las características de imágenes normales a una distribución normal estándar. Las probabilidades obtenidas en este proceso se utilizan como las puntuaciones de anomalía en la fase de inferencia.

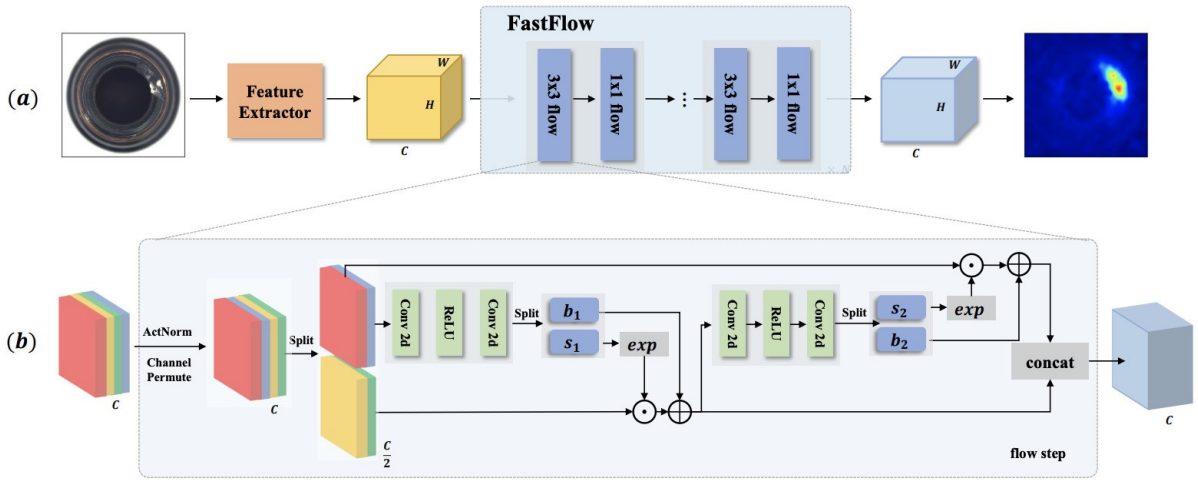

Figura 2.3: Resumen de la arquitectura FastFlow.

La parte clave de este modelo reside en la técnica de los flujos normalizadores. Esta se utiliza para aprender transformaciones entre distribuciones de datos; consisten en una serie de transformaciones no lineales que se aplican a las características de entrada con el fin de mapearlas a una distribución deseada. Una vez que se han aplicado estas transformaciones, se pueden calcular las probabilidades de cada punto en la distribución de salida. [54]

2.4.3. PaDiM

Algoritmo en el que durante la etapa de extracción de características parte de una red convolucinal pre-entenada, posteriormente trocea la imagen en parches con tal de extraer embeddings usando capas de todos los niveles de la red preentrenada.

Estos vectores de las diferentes capas se concatenan con el fin de lograr vectores que contengan información semántica de diferentes niveles de abstracción, así como distintas resoluciones, y así codificar información tanto detallada como contextual. Dado el uso de tantas capas es posible que se tenga información redundante por lo que reduce la dimensionalidad mediante una selección aleatoria de las capas.

Posteriormente se genera una distribución gaussiana multivariante para cada conjunto de características de cada parche. Representadas como una matriz de parámetros gaussianos; esto permite que el coste computacional del algoritmo escale de forma lineal con el tamaño del conjunto de datos.

Durante la inferencia se usa la distancia de Mahalanobis en cada parche para formar el mapa de anomalías, siendo las puntuaciones más altas las que indiquen mayor probabilidad de una anomalía [55].

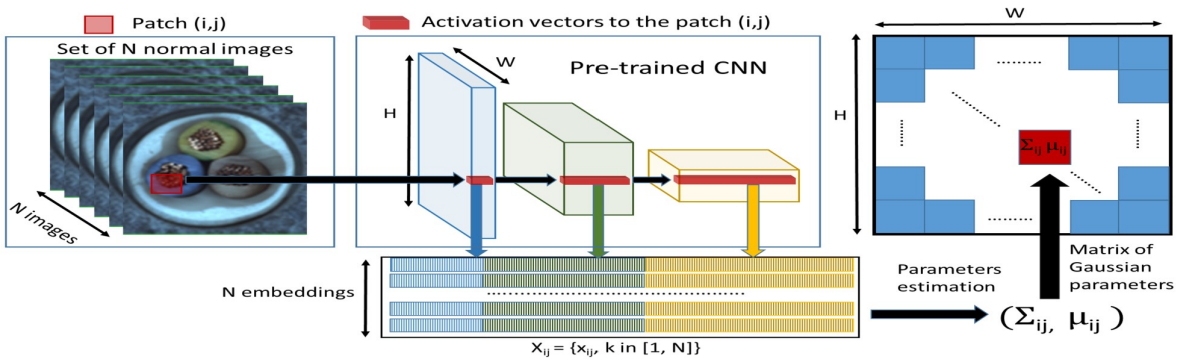

Figura 2.4: Resumen de la arquitectura PaDiM.

2.4.4. CFLOW-AD

Usa una red convolucional pre-entrenada en ImageNet que actúa como encoder para la extracción de características. Además, en las capas del encoder hace uso de un pooling multiescala para capturar información tanto local como global.

Posteriormente emplea un decoder multicapa en el que procesa las características codificadas con objetivo de estimar las probabilidades. Estas probabilidades se sobremuestrean para poder generar el mapa de anomalías [30].

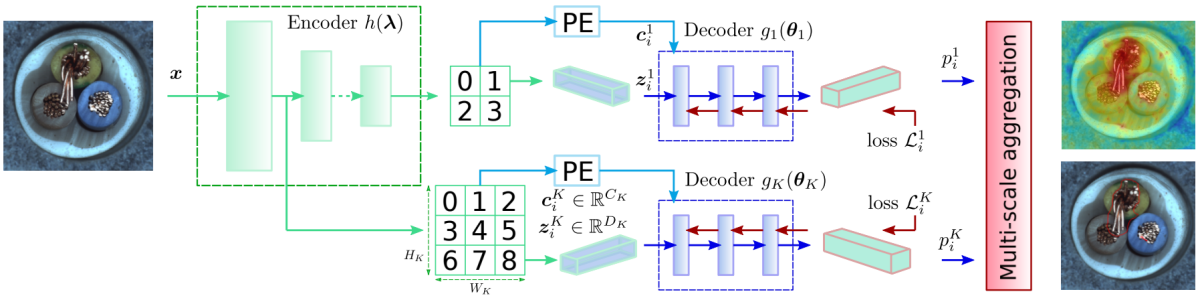

Figura 2.5: Resumen de la arquitectura CFLOW-AD.

2.4.5. STFPM

Hace uso de una red profesor pre-entrenada y una red estudiante con misma arquitectura. La red estudiante aprende la función de distribución subyacente a las imágenes buscando hacer coincidir cada vez más sus resultados según los que va viendo en la red profesor.

Para lograr una mayor robustez se hace que la red estudiante busque hacer coincidir diferentes capas de su arquitectura y no sólo la última. Con ello se consigue que la estudiante aprenda información semántica multinivel, así como ser capaz de detectar anomalías de diferentes tamaños. Durante la inferencia se comparan ambas redes donde una mayor diferencia indica una mayor probabilidad de anomalía [56].

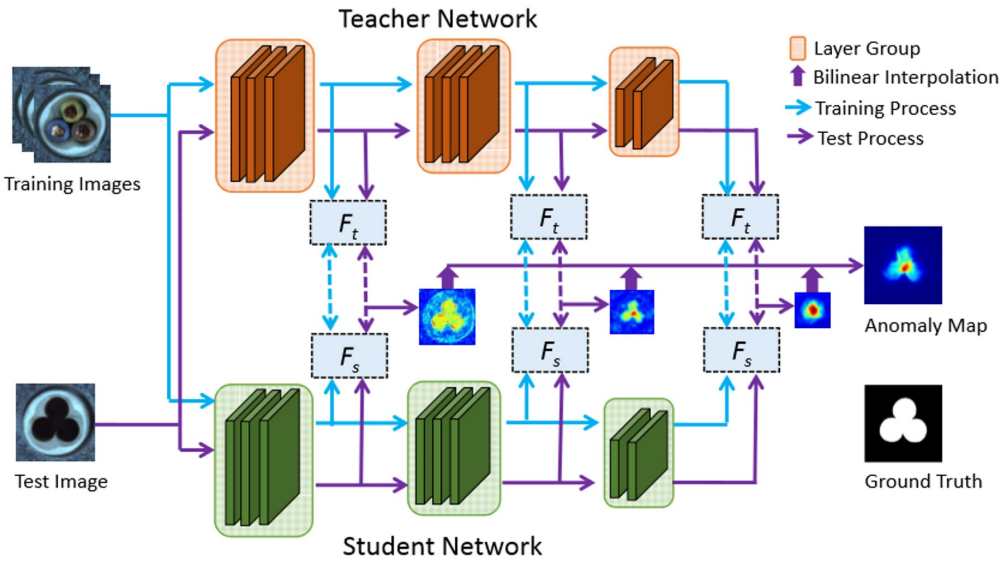

Figura 2.6: Resumen de la arquitectura STFPM.

2.4.6. DFM

Tiene una primera etapa de extracción de características en la que se parte de una red convolucional pre-entrenada. La salida de su penúltima capa se usa como vector semántico de características con una longitud fija. La segunda etapa se ocupa de la clasificación de anomalías. Para ello se hace uso de un análisis de componentes principales seguido por una estimación de densidad gaussiana condicionada a la clase [57].

2.4.7. DFKDE

Tiene una primera etapa de extracción de características en la que se parte de una red convolucional pre-entrenada. La salida de su penúltima capa se usa como vector semántico de características con una longitud fija. En la segunda etapa, dedicada a la detección de anomalías, se reducen las características mediante PCA a sus dieciséis componentes principales. Posteriormente, se emplea un kernel de densidad gaussiana para estimar la probabilidad de densidad de nuevas muestras en base a la colección de características obtenidas durante la etapa de entrenamiento. [58]

2.4.8. Conclusiones

En general, todos estos algoritmos utilizan una red neuronal pre-entrenada para extraer características de una imagen y luego procesan estas características de alguna manera para clasificar una imagen como anómala o no.

Generalmente dividen la imagen en parches y procesan cada uno de ellos por separado, valiéndose de técnicas estadísticas o de aprendizaje automático para la representación de las características extraídas de los parches. A lo largo de este proceso, suelen utilizarse también técnicas de reducción de la dimensionalidad y submuestreo. Finalmente todos establecen algún método para evaluar la probabilidad de anomalía tanto a nivel de cada píxel individual como de la imagen en sí misma.

Capítulo 3

Métricas de evaluación y el conjunto de datos MVTec-AD

3.1. Posibles métricas

Una parte fundamental en la fase de entrenamiento del modelo es la optimización de sus parámetros, para lo cual necesitamos de una métrica para evaluar los resultados durante el proceso. Además, establecer una métrica permite la comparación de los algoritmos que hayan sido desarrollados para resolver un mismo problema. Sin embargo, no existe una métrica perfecta, o lo que es lo mismo, todas tienen sus ventajas y desventajas [59].

3.1.1. Métricas de píxel

En el caso de las métricas a nivel de pixeles todos son de igual importancia. Consideremos:

- Un conjunto de imágenes $T := \{I_1, \ldots, I_n\}$.
- La métrica de anomalía para un píxel p es $A_i(p) \in \mathbb{R}$.
- Para cada píxel existe una *ground truth* (valor real) $G_i(p) \in \{0, 1\}$.
- Para poder comparar las métricas establecemos un umbral $t \in \mathbb{R}$ de manera que un píxel es anómalo si y solo si $A_i(p) > t$.

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 3.1: Matriz de confusion

Siguiendo la propuesta de la matriz de confusión (figura 3.1) los pixeles predichos pueden ser clasificados de la siguiente manera:

- Entendemos como verdaderos positivos (True Positive o TP) aquellos clasificados como anómalos y que verdaderamente lo son.

$$\mathrm{TP} = \sum_{i=1}^{n} \left| \{p \mid G_i(p) = 1\} \cap \{p \mid A_i(p) > t\} \right| \tag{3.1}$$

- Los pixeles que son falsos positivos (False Positive o FP) son lo predichos como anómalos pero que en realidad no lo son.

$$\mathrm{FP} = \sum_{i=1}^{n} \left| \{p \mid G_i(p) = 0\} \cap \{p \mid A_i(p) > t\} \right| \tag{3.2}$$

- Los verdaderos negativos (True Negative o TN) son aquellos que han sido clasificados como pixeles normales y que de hecho son normales.

$$\mathrm{TN} = \sum_{i=1}^{n} \left| \{p \mid G_i(p) = 0\} \cap \{p \mid A_i(p) \leq t\} \right| \tag{3.3}$$

- Y los falsos negativos (False Negative o FN) como aquellos que han sido predichos como normales, pero en realidad son anómalos.

$$\mathrm{FN} = \sum_{i=1}^{n} \left| \{p \mid G_i(p) = 1\} \cap \{p \mid A_i(p) \leq t\} \right| \tag{3.4}$$

Las anteriores métricas ofrecen medidas absolutas en el sentido que nos dicen la cantidad de pixeles de una imagen que son de cada tipo. En base a ellas podemos construir métricas en forma de tasas relativas:

- La tasa de verdaderos positivos expresa la proporción de verdaderos positivos sobre el total de casos anómalos. $\mathrm{TPR} = \dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$.
- La tasa de verdaderos negativos nos dice la proporción de verdaderos negativos sobre el total de casos normales. $\mathrm{FPR} = \dfrac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$.
- La precisión nos muestra la proporción de verdaderos positivos sobre el total de casos clasificados como anómalos. $\mathrm{PRC} = \dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$.

Una cuarta métrica relativa muy usada en la evaluación de algoritmos de segmentación es la Intersección sobre la Unión.

- Considerando el conjunto de todas las anomalías predichas $P = \bigcup_{i=1}^{n} \{p \mid A_i(p) > t\}$y el conjunto de ground truth $G = \bigcup_{i=1}^{n} \{p \mid G_i(p) = 1\}$.
- Su formulación matemática se puede expresar como $\mathrm{IoU} = \frac{|P \cap G|}{|P \cup G|} = \frac{\mathrm{TP}}{\mathrm{TP+FP+FN}}$.

Por último, tenemos la métrica F_1 que representa la media armónica entre el recall (TPR) y la precisión (PRC). Su fórmula puede ser expresada como: $\frac{2}{\frac{1}{\mathrm{TPR}} + \frac{1}{\mathrm{PRC}}}$.

3.1.2. Métricas de región

Tratar los pixeles como si fueran independientes introduce un sesgo hacia las grandes regiones anómalas, haciendo más difícil que el modelo se entrene para identificar regiones anómalas más pequeñas. Dado que el tamaño de los defectos puede variar, es razonable establecer métricas que se basen en la cantidad de regiones anómalas conectadas detectadas.

Las métricas de región PRO (Per Region Overlap) promedia los resultados sobre cada región conectada. Es decir, actúa de manera similar a la TPR, pero realizándolo para cada región anómala conectada de la imagen, y añadiendo después el mencionado promedio según la cantidad de regiones. Esta es especialmente útil para los casos en los que el tamaño de la anomalía no es relevante.

- Considerando que $C_{i,k}$ indica el conjunto de pixeles anómalos para una región conectada k en la imagen i.
- Así como P_i el número de pixeles predichos como anómalos según un umbral t
- El overlap por región se puede computar como:

$$\text{PRO} = \frac{1}{N} \sum_i \sum_k \frac{|P_i \cap C_{i,k}|}{|C_{i,k}|}, \tag{3.5}$$

3.1.3. Métricas independientes del umbral

Todas las métricas anteriores requieren que especifiquemos un umbral lo cual presenta un problema adicional a la hora de encontrar su valor óptimo para cada situación.

Por ello puede ser interesante evaluar las métricas en diferentes niveles de umbral, así como evaluar dos métricas al mismo tiempo. La técnica más conocida para evaluar dos métricas simultáneamente se conoce como Área Bajo la Curva (AUC). Si bien es cierto que la curva más usada es la ROC, que consiste en evaluar la tasa de verdaderos positivos frente a la de verdaderos negativos para diferentes umbrales. Enumeramos las que para nuestro problema constituyen opciones interesantes:

- Curva ROC: FPR frente a TPR
- Curva PR: TPR frente a PRC, evalúa el rendimiento en términos de precisión y recall.
- Curva PRO: FPR frente a PRO, parecida a la curva PR pero promedia el rendimiento sobre cada región conectada en lugar de sobre pixeles individuales.
- Curva IoU: FPR frente a IoU, lo cual mide la similitud entre dos imágenes en términos de la cantidad de pixeles que se superponen.

3.1.4. La necesidad del umbral

Si bien las métricas independientes del umbral comparan dos medidas al mismo tiempo y hacen que no sea necesario la elección de un umbral, presentan el problema de una

interpretabilidad más difícil, así como tratar las dos métricas elegidas cómo igual de importantes.

Al margen de esto, en última instancia deberemos decidir un umbral para poder realizar predicciones en la fase de inferencia. Para decidir su valor, la aproximación más habitual es hacerlo en base únicamente a las imágenes normales, pues no tenemos garantías de que las imágenes anómalas recojan lo suficientemente bien todos los tipos de anomalías que se pueden dar.

Recordemos que en el problema de detección de anomalías en imágenes los conjuntos de datos durante la fase de entrenamiento apenas tienen casos anómalos. Y por lo que respecta al umbral, nuestro objetivo es encontrar un umbral que separe los casos normales de los anómalos, para lo cual existen diferentes técnicas:

- Umbral máximo-píxel: selecciona como umbral la puntuación de anomalía máxima que se haya dado durante la fase de entrenamiento. En este caso si se diera la presencia de una anomalía con alta puntuación, en la fase de inferencia tendríamos un umbral muy restrictivo.
- Umbral p-Cuantil: dada la distribución de todas las puntuaciones de los pixeles durante la fase de inferencia, establece la proporción de pixeles que se clasificarán como anómalos.
- Umbral k-sigma: calcula la media y desviación típica de todas las puntuaciones de los pixeles durante la fase de inferencia y establece el umbral como $t = \mu + k\sigma$.
- Umbral máxima-área: las tres técnicas anteriores entienden los pixeles como independientes, en esta técnica se propone establecer un valor de área mínima que han de superar los pixeles vecinos clasificados cómo anómalos.

3.1.5. Conclusiones

Utilizar métricas que evalúen el rendimiento del modelo en términos de pixeles individuales puede sesgar el resultado hacia las grandes regiones anómalas, lo que dificulta la identificación de regiones anómalas más pequeñas.

Por ello es útil evaluar el rendimiento de un modelo en diferentes niveles de umbral y considerar más de una métrica al mismo tiempo. La técnica AUC, que permite evaluar dos métricas simultáneamente, puede ser útil para tener una visión más completa del rendimiento del modelo.

Sin embargo, es necesario especificar un umbral al evaluar el rendimiento de un modelo, este umbral puede ser difícil de optimizar y puede variar según la situación.

3.2. Datos utilizados

3.2.1. Introducción

El avance en modelos de inteligencia artificial siempre ha ido de la mano de conjuntos de datos de alta calidad. Este caso es más notable en los avances de visión por computador como pudimos ver en los modelos a raíz de los datos MNIST [60], y años después con

el conjunto de datos de ImageNet [44]. Al hecho de darle importancia a la calidad del conjunto de datos se le conoce como Data Centric Approach.

Cabe diferenciar el termino novelty detection de anomaly detection, pues el primero entiende como imágenes anómalas aquellas que presentan una diferencia sustancial, mientras que el segundo se refiere a una ligera diferencia la cual se puede incluso segmentar dentro de la propia imagen.

Para el desarrollo de modelos aplicados a novelty detection suele ser suficiente adaptar el conjunto de ImageNet de forma que uno, o varios, conceptos constituyan la anomalía. Sin embargo, este tipo de conjuntos de datos no permite proporcionar métricas robustas para comparar modelos enfocados a la detección de anomalías. El conjunto de datos MVTec-AD surge de la necesidad de disponer de un conjunto de datos público que permitiera desarrollar y evaluar modelos desarrollados para este tipo de problema.

La dedicación por parte de la comunidad investigadora a este tipo de problemas ha ido creciendo exponencialmente en los años recientes, tal y como podemos ver en la figura 3.2:

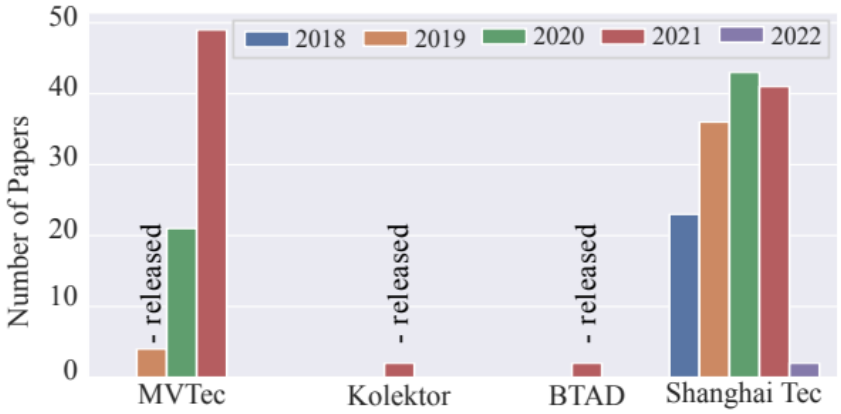

Figura 3.2: Número de publicaciones a lo largo del tiempo para diferentes conjuntos de datos de detección de anomalías

3.2.2. Características

El conjunto de datos MVTec-AD fue creado por el instituto MVTec Software GmbH, se trata de un conjunto de datos que intenta simular escenarios reales en el sector de la producción, y es uno de los conjuntos de datos más completos y ampliamente utilizados para la evaluación de algoritmos de detección de anomalías en imágenes industriales.

La popularidad de este conjunto de datos se debe a la alta calidad de sus imágenes las cuales tienen poco ruido y distorsión; también ofrece una gran variedad de objetos de diferentes formas, tamaños, colores y texturas; y está disponible de forma gratuita para su uso en investigación y desarrollo.

Las imágenes industriales del conjunto de datos son de diferentes objetos tales como cables, lentes, rodamientos y herramientas; con anomalías distintas como por ejemplo: grietas, manchas, deformaciones y falta de componentes.

El conjunto de datos está dividido en un total quince categorías, cinco se corresponden con imágenes de texturas y las diez restantes con imágenes de objetos. Teniendo en cuenta todas las categorías, hay un poco más de setenta tipos distintos de anomalías.

	Category	# Train	# Test (good)	# Test (defective)	# Defect groups	# Defect regions	Image side length
Textures	Carpet	280	28	89	5	97	1024
	Grid	264	21	57	5	170	1024
	Leather	245	32	92	5	99	1024
	Tile	230	33	84	5	86	840
	Wood	247	19	60	5	168	1024
Objects	Bottle	209	20	63	3	68	900
	Cable	224	58	92	8	151	1024
	Capsule	219	23	109	5	114	1000
	Hazelnut	391	40	70	4	136	1024
	Metal Nut	220	22	93	4	132	700
	Pill	267	26	141	7	245	800
	Screw	320	41	119	5	135	1024
	Toothbrush	60	12	30	1	66	1024
	Transistor	213	60	40	4	44	1024
	Zipper	240	32	119	7	177	1024
	Total	3629	467	1258	73	1888	-

Figura 3.3: Características del conjunto de datos MVTec-AD.

Las imágenes están disponibles en formato gris y a color (RGB) y tienen diferentes tamaños y resoluciones. Sin embargo, la forma de trabajar habitual es en su formato de 900x900 pixeles utilizando sus tres canales RGB. Está compuesto por 5354 imágenes de alta resolución donde las imágenes anómalas constituyen un poco más del 20 % del total de imágenes. Si bien es cierto que tan solo un 2.7 % de los pixeles son anómalos.

Con objetivo de emular el problema de forma tan real como sea posible, y en consonancia con los cuatro retos que comentamos en la sección 2.2, el conjunto original se presenta de tal manera que ninguna de las imágenes anómalas este presente en el conjunto de datos utilizado para el entrenamiento del modelo.

Además, el conjunto de datos proporciona una imagen binaria en la que se localiza la anomalía a nivel de píxel. Todo lo comentado hasta ahora nos permite desarrollar una fase de evaluación del modelo muy rica en información, tal y como podemos ver en la figura 3.4.

3.2.3. Métricas a utilizar

Además de todo lo expuesto en la sección 3.1 debemos añadir la consideración de que estamos frente a un conjunto de datos desbalanceado. Así pues, cuando se trata de evaluar el rendimiento de algoritmos de clasificación binaria en conjuntos de datos desbalanceados, la Curva PR es a menudo una mejor elección que la curva ROC.

La razón por la que la curva PR es normalmente más adecuada para conjuntos de datos desbalanceados es que mide la precisión (PRC o Positive Predictive Value, es decir, el número de verdaderos positivos dividido por el número total de positivos predichos) en lugar de la tasa de verdaderos positivos (TPR).

Tampoco debemos olvidar la Curva PRO (Per Region Overlap) la cual promedia el rendimiento sobre cada región conectada en lugar de sobre pixeles individuales y puede ser útil para evaluar el rendimiento en situaciones en las que el tamaño de la anomalía no es relevante.

Para este problema en concreto, sería particularmente interesante una curva que evaluara el PRO frente al PPV, ya que estaríamos combinando el aspecto de la curva PR que tienen en cuenta que es un problema desbalanceado con el aspecto de la curva PRO que

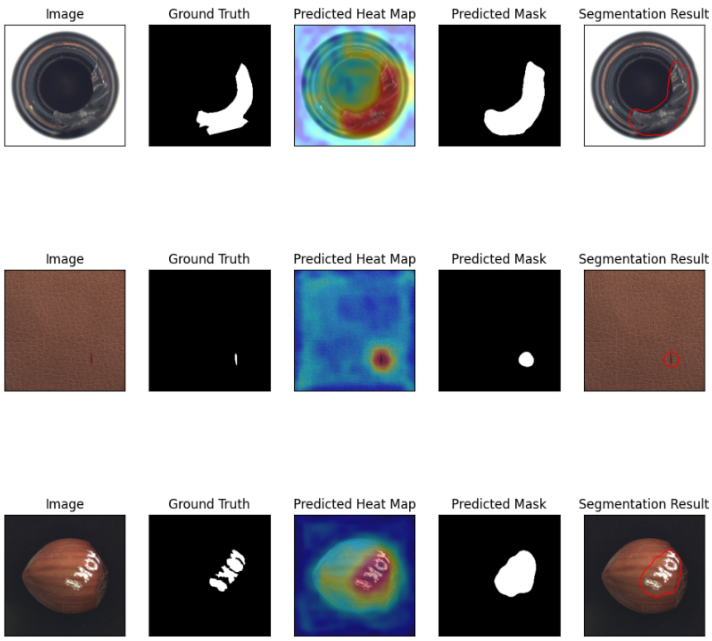

Figura 3.4: Ejemplos sobre distintas categorías en el conjunto MVTec-AD.

tiene en cuenta que el tamaño de la anomalía no es importante. Sin embargo, dado que esta curva actualmente no está implementada en la librería Anomalib, en las siguientes secciones trabajaremos con la curva PR y la curva PRO.

Capítulo 4

Evaluación de los modelos y técnicas para mejorar los resultados

En esta sección estudiaremos los resultados que obtienen las librerías pyOD y Anomalib sobre el conjunto de datos MVTec-AD, la puntuación obtenida de cada modelo es producto de realizar un promediado de su AUC-ROC obtenida en cada una de las quince categorías de forma aislada, es decir, un modelo distinto para cada categoría. Sobre la implementación que se ha llevado a cabo, se ha trabajado con los modelos disponibles en la versión 0.3.7 de Anomalib y la semilla de aleatorización ha sido el número 42, las evaluaciones se han hecho sobre los modelos con sus configuraciones por defecto en la sección 4.1 y a lo largo de la sección 4.2 se han evaluado los modelos cambiando las aspectos correspondientes a cada una de las técnicas usadas para mejorar el rendimiento de un modelo de aprendizaje automático. Por lo que respecta al conjunto de datos y su división en los conjuntos de entrenamiento y test se ha seguido la estructura propuesta en la figura 3.3.

4.1. Resultados

Si atendemos a los resultados de las tabla 4.1 podemos ver que el modelo CatB tiene el mejor rendimiento, seguido de cerca por CBLOF. Los modelos LOF, XGBOD y XGB tienen un rendimiento ligeramente inferior.

Modelo	AUC-ROC
CBLOF	0.7598
LOF	0.7419
XGBOD	0.7417
XGB	0.7432
CatB	0.8081

Cuadro 4.1: Métricas AUC-ROC de los principales modelos en pyOD

Por otro lado, en la tabla 4.2 es el modelo PatchCore el que obtiene los mejores resultados con un AUC-ROC de 0.98, seguido por CFlow y PaDiM, teniendo el resto de los modelos métricas un poco inferiores a excepción del algoritmo DFKDE con un AUC-ROC de 0.7740.

Dejando de lado que CatB supera al modelo DFKDE, son las implementaciones de

Modelo	AUC-ROC
CFlow	0.9620
DFM	0.9430
DFKDE	0.7740
FastFlow	0.8968
PatchCore	0.9800
PaDiM	0.9500
STFPM	0.8760

Cuadro 4.2: Métricas AUC-ROC de los principales modelos en Anomalib, pre-entrenados con una ResNet-50.

Anomalib las que obtienen resultados notablemente mejores. Sin embargo, a la hora de estudiar modelos de aprendizaje profundo siempre hay que tener en cuenta su requisitos computacionales y en esta ocasión no se da la excepción.

Como podemos ver en el gráfico 4.1 el modelo PatchCore consigue los mejores resultados utilizando sustancialmente menos parámetros que los dos siguientes, cabe destacar que CFLOW-AD frente a PaDiM mejora el resultado ínfimamente teniendo en cuenta que utiliza casi diez veces más parámetros, lo cual es muy relevante si los requisitos computacionales son un factor a tener en cuenta. Aunque no exista una correspondencia entre coste computacional y cantidad de parámetros, dada la magnitud de la diferencia entre modelos podemos entender el número de parámetros como una aproximación de su exigencia computacional.

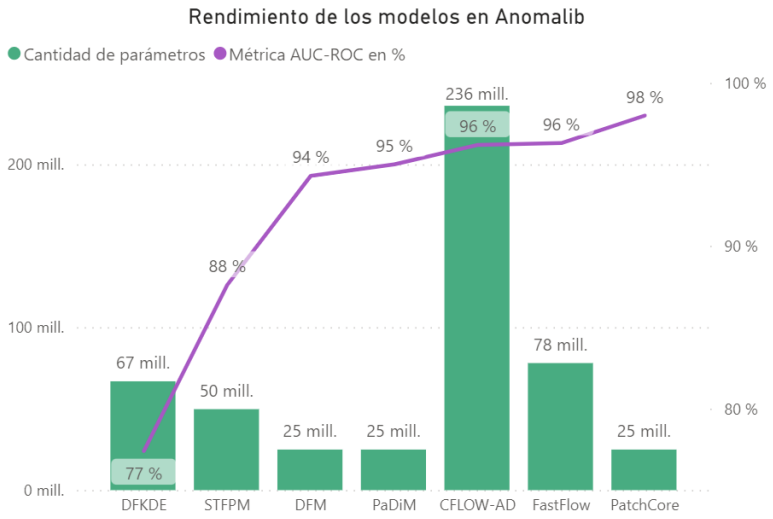

Figura 4.1: Resultado AUC-ROC frente a la cantidad de parámetros para los principales modelos de Anomalib

De hecho, en nuestro caso, a la hora de trabajar con los modelos no se ha conseguido la capacidad computacional necesaria para implementar aquellos que sobrepasan los veinticinco millones de parámetros. Lo cual ha sido un factor limitante a la hora de reproducir resultados de diversos modelos y sus respectivas configuraciones.

4.2. Posibles mejoras

4.2.1. Data Augmentation

Un problema habitual durante el aprendizaje es que el modelo se sobreajuste a los datos de los que se dispone, lo cual se da de una forma más pronunciada en el caso de que tengamos pocos. Una manera de reducir este efecto es mediante la generación sintética de nuevos datos (data augmentation), lo cual en el caso de las imágenes se puede realizar mediante su rotación, traslación, recorte, zoom, inversión, etc. Es importante tener en cuenta que solo se debe aplicar data augmentation en el conjunto de entrenamiento, ya que el objetivo es mejorar el rendimiento del modelo en datos desconocidos. La razón por la que no se debe aplicar al conjunto de test ya que este conjunto se utiliza para evaluar el rendimiento del modelo en datos reales. Por ello si se aplicara a este conjunto los resultados no sería representativos del rendimiento del modelo en datos reales.

Debido a nuestras limitaciones computacionales y a que los modelos de Anomalib ya obtenían muy buenos resultados en las categorías del conjunto de datos MVTec-AD. Para la evaluación de esta técnica trabajaremos con un conjunto de datos de naranjas que se ha creado de manera manual y específicamente para este proyecto. Dicho conjunto de datos puede ser consultado en el repositorio de GitHub a través del enlace que aquí se proporciona [1], además, una somera descripción de los mismos puede ser encontrada en la seccción A.2 del apéndice. En este conjunto de datos, los datos sintéticos se han generado volteando las imágenes horizontal y verticalmente, y cambiando su iluminación.

	ResNet-18	ResNet-50
No Augmentation	0.61	0.59
Sí Augmentation	0.57	0.75

Cuadro 4.3: Métricas AUC-ROC tras el uso modelo PaDiM en función de la red pre-entrenada que utiliza, y de si se hace el entrenamiento con datos sintéticos además de los originales

	ResNet-18	ResNet-50
No Augmentation	0.63	0.59
Sí Augmentation	0.61	0.72

Cuadro 4.4: Métricas AUC-PR tras el uso modelo PaDiM en función de la red pre-entrenada que utiliza, y de si se hace el entrenamiento con datos sintéticos además de los originales

Dados los resultados de la tabla 4.3 y 4.4 podemos extraer varias conclusiones. Si bien es cierto que el uso de imágenes adicionales no ha supuesto una diferencia en el caso de la ResNet-18 pues se llegan a obtener resultados ligeramente peores, no sucede lo mismo en el caso de la ResNet-50 en el cual conseguimos una mejora sustancial, en concreto el incremento es del 31.58 % en la AUC-ROC y del 18.03 % para la AUC-PR.

[1] El repositorío GithHub con todo el código y datos utilizados en este TFG es: https://github.com/PabloFerrerGo/pfg_anomaly_detection

4.2.2. Transfer Learning

Consiste en utilizar un modelo ya entrenado en un problema similar como base y continuar el entrenamiento con los datos específicos del problema. Esto puede ser útil cuando se dispone de una cantidad limitada de datos de entrenamiento, ya que permite aprovechar el conocimiento adquirido por el modelo previamente. Básicamente los pesos del modelo serán congelados en mayor o menor medida dependiendo de la cantidad de datos que dispongamos, y aprovechados para extraer información relevante que, posteriormente, procesaremos.

Modelo	ResNet18	ResNet50	cambio en %
DFM	0.963	0.943	2.23
DFKDE	0.762	0.774	1.55
PatchCore	0.973	0.980	0.71
PaDiM	0.891	0.950	6.21
STFPM	0.893	0.876	-1.94

Cuadro 4.5: Métricas AUC-ROC de los modelos pre-entrenados en redes ResNet-18 y ResNet50, y su respectivo cambio porcentual.

Según los resultados de la tabla 4.5 vemos que el uso de una ResNet-50 mejora en promedio un 1.75 % la métrica AUC-ROC. Además, existe documentación sobre como PatchCore utilizando una ResNet-101 [61] ha logrado mejorar su AUC-ROC hasta 0.984. Dado que ResNet-101 proporciona una precisión del 81.9 % en el problema de clasificación del conjunto de datos ImageNet. Cabe esperar que el uso de redes que han conseguido mejores resultados en dicho conjunto de datos, tales como CoCa [62], proporcionarían mejores resultados.

Sin embargo, las redes que mejor funcionan suelen hacerlo a costa de utilizar muchos parámetros (CoCa utiliza 2.100 millones y logra una precisión del 91 %) por lo que sería conveniente el uso de una red equilibrada en cuanto resultados que proporciona y sus requisitos computacionales, como es el caso de EfficientNet-B6 la cual consigue una precisión del 90.2 % utilizando 480 millones de parámetros [63].

4.2.3. Ajuste de hiperparámetros

Dada la sofisticación de los modelos, es inevitable que estos dependan también de gran cantidad de hiperparámetros los cuales pueden ser configurados en base a distintos valores y técnicas. Su configuración supone un aspecto crucial, y si bien es cierto que suelen tener valores por defecto, estos casi nunca son los óptimos para nuestro problema por lo que deben ser optimizados al menos de forma aproximada si queremos mejorar el rendimiento del modelo.

Dado que, por ejemplo, el modelo PaDiM no tiene hiperárametros que modificar, trabajaremos con el modelo DFM sobre la categoría de imágenes *capsule* y preentrenado con una ResNet-18.

En base a los resultados de las tablas 4.6 y 4.7 vemos que el tamaño del pooling no es determinante al menos según los valores utilizados, sin embargo, no cabe duda de que establecer la puntuación de anomalía realizando una reconstrucción por análisis de componentes principales es una opción mucho mejor que la vía del modelado Gaussiano.

	Pooling 2x2	Pooling 4x4
Modelado Gaussiano	0.4033	0.3646
Reconstrucción PCA	0.9441	0.9473

Cuadro 4.6: Métricas AUC-ROC tras el uso modelo DFM en función del tamaño del pooling y de la forma para establecer la puntuación de anomalía

	Pooling 2x2	Pooling 4x4
Modelado Gaussiano	0.8361	0.8570
Reconstrucción PCA	0.9857	0.9858

Cuadro 4.7: Métricas AUC-PR tras el uso modelo DFM en función del tamaño del pooling y de la forma para establecer la puntuación de anomalía

Cabe tener en cuenta que el ajuste de los hiperparámetros es un proceso iterativo en el que se ajustan los hiperparámetros normalmente después de haber entrenado un modelo base utilizado como referencia. Una forma de búsqueda habitual se llama Búsqueda de Cuadrícula (Grid Search) y consiste en evaluar diferentes combinaciones de valores de hiperparámetros, cuanto más exhaustivamente mejor, y elegir la combinación que da el mejor rendimiento. Si bien en esta sección se ha implementado un ejemplo sencillo de ajuste de hiperparámetros, la librería Anomalib ofrece soluciones de Grid Search mediante el uso de la plataorma WandB, y en el repositorio se explica como realizar este ajuste más sofisticado.

Capítulo 5

Simulación de un caso real

5.1. Descripción del caso práctico

Para que esta aplicación supusiera una aportación útil era necesario simular un escenario tan real como fuera posible, para ello se ha elegido implementar un proceso de detección de anomalías en imágenes usando la placa PYNQ-Z2 así como los sensores y actuadores necesarios (cuyas consideraciones técnicas se pueden encontrar en A.1). Esto supone pues el enmarque de un proceso completo, desde la recogida de los datos hasta la muestra de resultados, haciendo uso de la comunicación de tipo MQTT entre el dispositivo PYNQ-Z2 y nuestro ordenador, que de ahora en adelante llamaremos servidor. [1]

Sobre la comunicación elegida, denominada comunicación MQTT y muy usada en aplicaciones del campo Internet de las Cosas, esta comunicación se basa en el concepto de publicar y suscribirse a temas (o canales). Permite que unos dispositivos puedan publicar mensajes a un canal específico y otros se suscriban para recibir los mensajes. Esto permite una comunicación unidireccional, en la que un dispositivo puede enviar mensajes a otro sin necesidad de que este último envíe una respuesta. Uno de los principales beneficios de MQTT es su eficiencia en el uso de ancho de banda y recursos de red, lo cual también facilita una mayor escalabilidad. El resto de consideraciones técnicas que se han considerado relevantes se encuentran explicadas en la sección A.1 del apéndice.

El proceso, tal y como se puede comprobar en la figura 5.1 puede ser descrito de la siguiente manera. Por un lado tenemos el dispositivo que en este caso es la PYNQ-Z2:

1. Su proceso comienza con un sensor de movimiento está actualizándose en bucle y sólo se pasará a la siguiente fase si un objeto pasa por delante de su rango.

2. Cuando un objeto pasa por delante del sensor una cámara realizará una foto de dicha foto y la enviará por MQTT al servidor haciendo uso del tema *photo*.

3. Tras ello se suscribe al tema MQTT *results* a la espera de que se le comunique si el objeto es ó no anómalo.

4. Una vez recibida esta información generara el sonido (buzzer) y/o luz (led stick) correspondiente

[1] Tanto el código como los datos utilizados se pueden encontrar en https://github.com/PabloFerrerGo/pfg_anomaly_detection

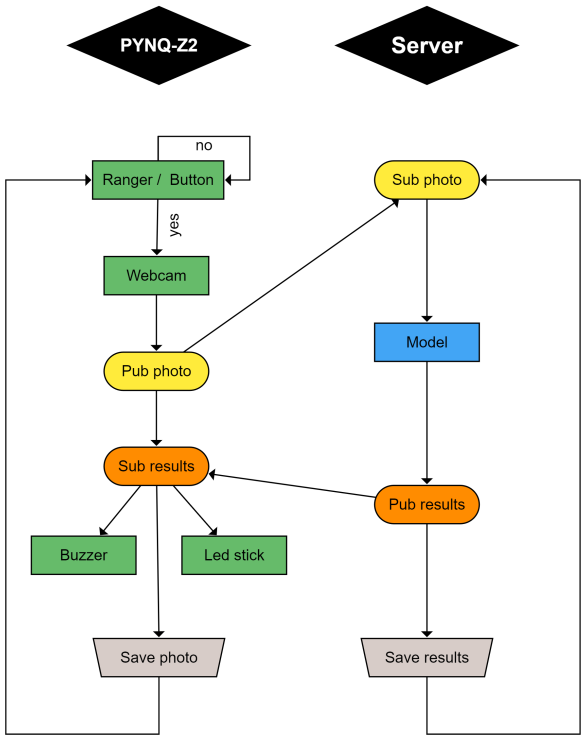

Figura 5.1: Proceso del caso simulado

5. Por último, se guardará la foto para que en el futuro se pueda monitorizar, realizar una evaluación con los nuevos datos y/o reentrenar el modelo. Cabe destacar que esta fotografía se guarda tanto en el dispositivo como en el servidor, esto se debe a que la memoria no ha supuesto un limitante y es una forma de asegurarse que guardamos la mayor cantidad de nuevos ejemplos, pues podría ocurrir un error en la comunicación y que dicha foto no llegara al servidor. Sin embargo, si la memoria sí fuera un limitante la mejor opción sería no guardar la foto en el dispositivo.

6. Tras esto se vuelve al primer paso de esta lista, y dicho bucle solo será interrumpido si se pulsa un botón.

Por otro lado, tenemos la parte del servidor:

1. Su primera acción es suscribirse al tema MQTT *photo* a la espera de que el dispositivo le envíe una.

2. Una vez recibida se la envía al modelo de aprendizaje automático para que realice una predicción. En este caso hemos utilizado el modelo PaDiM, y hemos comparado su rendimiento tanto cuando usa como red preentrenada la ResNet-18 como la ResNet-50. Los resultados de estas aproximaciones se muestran en las tablas 4.3 y 4.4

3. Le envía el resultado de si se trata de una anomalía al dispositivo mediante el tema MQTT *results*.

4. Posteriormente, aparte de la foto, se guardan resultados del modelo más complejos (como los que podemos ver en la figura .3.4) que no eran necesarios para el dispositivo

y no podían ser enviados por MQTT porque eran demasiado pesados. Por ejemplo, un mapa de calor con la probabilidad de anomalía para cada píxel, lo cual puede ser especialmente útil en la fase de monitoreo y evaluaciones posteriores.

Es importante tener en cuenta que, aunque el proceso se describe de manera secuencial, en realidad se está realizando de manera concurrente. Es decir, mientras el dispositivo está esperando la respuesta del servidor, éste puede estar procesando otra foto enviada por otro dispositivo. Además, es importante destacar que el sistema está diseñado para ser escalable, por lo que se pueden añadir más dispositivos y servidores para mejorar su rendimiento y capacidad de procesamiento.

5.2. Resultados

En las figuras 5.2 y 5.3 se hayan las curvas con las cuales se han calculado sus correspondientes métricas AUC-ROC, tal y como se muestra en la tabla 4.3.

Concretamente la figura 5.2 es el resultado del uso de un PaDiM preentrenado con una ResNet-50 y entrenado sobre el conjunto de datos propio A.2 sin aplicar *data augmentation* (DA). Por otro lado, la figura 5.3 es el resultado de aplicar el mismo modelo que en la anterior pero en este caso aplicando *data augmentation* al conjunto de datos.

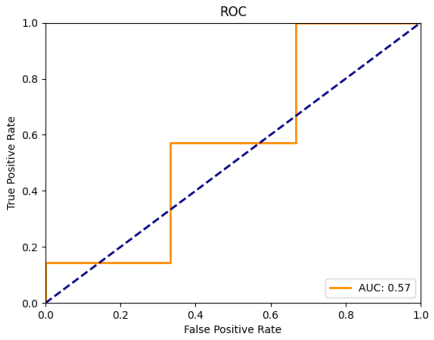

Figura 5.2: PaDiM sin DA

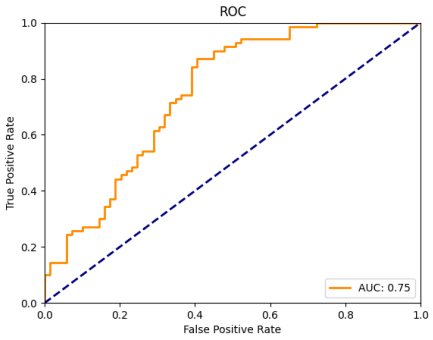

Figura 5.3: PaDiM con DA

Además, la librería Anomalib nos permite visualizar las predicciones realizadas por el modelo de formas que aportan mucha información. Como es el caso de la figura 5.4 en la que se muestran visualizaciones de las predicciones realizadas sobre una naranja normal, ó el de la figura 5.5 en el que podemos apreciar las de una naranja anómala.

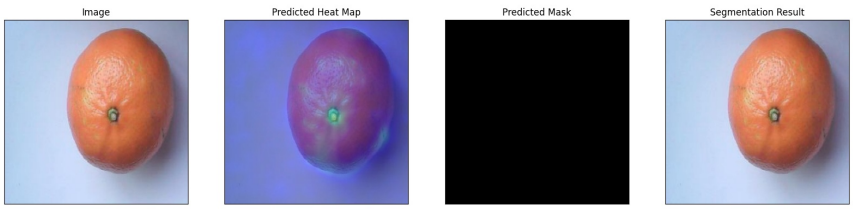

Figura 5.4: Ejemplo de predicción sobre naranja normal del modelo PaDiM con ResNet-50 y Data Augmentation

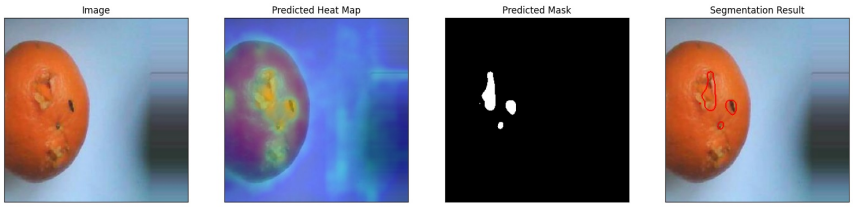

Figura 5.5: Ejemplo de predicción sobre naranja anómala del modelo PaDiM con ResNet-50 y Data Augmentation

5.3. Latencia media y posibles mejoras

El objetivo de este caso práctico no solo era comprobar que se puede implementar un modelo de la librería Anomalib en un problema real sino también identificar si la parte del modelo suponía un cuello de botella.

Hemos podido comprobar que la respuesta a la segunda cuestión era negativa, por lo general se ha conseguido una latencia de cinco segundos desde que un objeto se detecta hasta que se emite una respuesta física (luz o sonido) dependiendo de si es una anomalía. Sin embargo, la parte que corresponde al modelo ofrece una latencia media de un segundo. Tras un análisis del proceso podemos concluir que optimizar el código de la comunicación MQTT y el respectivo a de la cámara son la mejor forma de reducir la latencia del sistema.

Es importante tener en cuenta que la latencia del sistema puede ser afectada por otros factores, como la carga de trabajo del servidor, la velocidad de la conexión de red, el tamaño y complejidad del modelo de aprendizaje automático, entre otros. También es posible que la optimización del código no sea suficiente para reducir la latencia del sistema a un nivel deseado. En este caso, puede ser necesario considerar opciones como utilizar servidores más potentes o mejorar la conexión de la red.

Capítulo 6

Consideraciones finales

6.1. Conclusiones

Hemos conseguido establecer un marco teórico que sirva de apoyo a la hora de comenzar a leer literatura sobre la detección de anomalías en imágenes anómalas. Se ha justificado la importancia de la arquitectura de aprendizaje profundo llamada red neuronal convolucional al mismo tiempo que hemos explicado arquitecturas más específicas al problema que nos ocupa, tales como las redes adversariales generativas y los *autoencoders*.

Se ha prestado especial atención a la librería Anomalib, que ha sido creada recientemente la cual implementa modelos diseñados específicamente para resolver nuestro problema. Explicando las partes fundamentales de los modelos actualmente presentes en la librería, así como sus características comunes, prestando también atención a sus requisitos computacionales.

Hemos visto la importancia de establecer una métrica informativa para poder medir y comparar los resultados de un modelo. Se ha justificado la importancia de métricas no tan comunes, tales como el *Positive Predictive Value* dado el desbalanceo en los datos, o como el *Per Region Overlap* dado que trabajamos con imágenes y el tamaño de la anomalía no suele importar. Aun así, en última instancia la elección de la métrica más adecuada para la evaluación del rendimiento dependerá de las características y requisitos del problema específico.

A la vez que hemos mostrado los resultados de los modelos base, hemos propuesto diferentes vías de mejora: *data augmentation, transfer learning* y ajuste de hiperparámetros. Tras la aplicación de ellas hemos podido comprobar que todas presentan mejoras lo suficientemente significativas como para ser tenidas en cuenta. Aunque como ya se ha dicho, el nivel de mejora que aportarán depende del problema y su aplicación se trata más bien de un proceso empírico e iterativo.

Hemos comprobado que la librería Anomalib presenta las funcionalidades suficientes como para poder aplicar sus modelos en un entorno real. En concreto, se ha simulado un entorno tan realista como nos ha sido posible, y se ha identificado que la fase del modelo detectando si la nueva imagen es o no anómala no supone un importante cuello de botella.

Cabe tener en cuenta que Anomalib es una librería de código en desarrollo que ha sido creada hace un año y está siendo desarrollada por varias personas de manera colaborativa. Aunque tiene algunas limitaciones, como una falta de documentación completa o de consenso a la hora de definir el funcionamiento de los modelos. Sin embargo, la librería

se haya en constante mejora y probable que se añadan nuevas funcionalidades así como se solucionen problemas en el futuro. También, en el repositorio de este Trabajo de Fin de Grado se ha implementado un punto de partida útil para quienes quieran comenzar a trabajar con la librería, en dicho código se implementan las etapas y técnicas más comunes pero también fundamentales en todo problema de aprendizaje automático.

En resumen, en este trabajo se expone el marco teórico básico que sirve de punto de partida para el estudio del problema de detectar imágenes anómalas, se describen las características de un conjunto de datos de la suficiente calidad (MVTec-AD), se analizan las métricas relevantes y se explica los modelos de más fácil acceso todos ellos disponibles en la librería Anomalib, incluyendo su respectiva implementación la cual puede ser encontrada en el repositorio del proyecto. También se explican las principales forma de mejorar los modelos y la forma de implementarlas haciendo uso de las funcionalidades de la librería, en último lugar se comprueba que es viable la aplicación de estos modelos en un escenario real.

6.2. Trabajo futuro

Conforme a lo que se ha comentado antes sobre las carencias y limitaciones de Anomalib, surgen distintas vías de trabajo que escapan al objetivo de este TFG. Algunas de ellas son:

- Si bien la librería cuenta con métricas como el AUC-PRO y el AUC-PR, sería interesante la implementación de una nueva AUC que relacionara el *Per Region Overlap* con el *Positive Predictive Value*.
- A la hora de establecer el umbral para determinar lo que se considera anomalía, la librería suele hacer uso de un método que denomina *adapatative*, sería útil la implementación de las cuatro formas distintas para establecer el umbral que se explican en el apartado 3.1.4.
- Aprovechando que en el código del repositorio se enseña como cambiar la red preentrenada que usan los modelos, se puede pasar fácilmente a la fase de elegir aquella óptima para el problema específico. Sin embargo, los requisitos computacionales han sido un limitante importante para este proyecto y recomendamos que se tengan en cuenta a la hora de planificar futuros desarrollos.
- En el sentido del anterior punto, proponemos la exploración de entrenamiento por GPU en lugar de CPU. Para este proyecto se ha explorado la vía de usar la capacidades de calculo en la nube de *Google Colab Pro*, sin embargo, la GPU que proporcionan no es compatible con la versión 0.3.7 de Anomalib.
- Dado que se trata de un problema con muchas aplicaciones en el mundo real, sería también interesante la implementación de una métrica coste, la cual evalúa el modelo en función del coste asociado a cada tipo de error.
- A la hora de realizar el ajuste de hiperparámetros del modelo, Anomalib presenta una solución que utiliza la plataforma de WandB, permitiendo establecer una *Grid Search* y evaluar cada uno de los resultados de una manera potente. Esta vía de desarrollo es muy recomendable si se quiere implementar un flujo MLOps.

- Cabe destacar que toda la implementación de Anomalib esta hecha principalmente en Torch, crear una versión en TensorFlow permitiría el uso de nuevas funcionalidades y un mayor alcance de la librería.

Apéndice A

Apéndices

A.1. Aspectos técnicos del caso práctico

- Para la luz utilizamos el RGB LED Stick, consiste en un actuador de iluminación de tres colores basado en el LED RGB (rojo, verde y azul) de alta luminosidad y cuenta con una serie de pines de entrada y salida que permiten controlar el color y el brillo del LED.

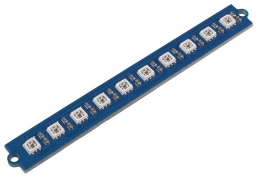

Figura A.1: Grove RGB LED Stick (10 - WS2813 Mini)

- Como sensor de movimiento utilizamos el Ultrasonic Ranger, sensor que se utiliza para medir la distancia a un objeto. Funciona enviando una señal de ultrasonidos a un objeto y midiendo el tiempo que tarda en regresar. A partir de este tiempo, el sensor puede calcular la distancia al objeto utilizando la velocidad del sonido.

Figura A.2: Grove Ultrasonic Ranger V2.0

- Para el sonido utilizamos el Grove Buzzer, se trata de módulo de sonido electromecánico que cuenta con un altavoz y una serie de pines de entrada y salida que permiten controlar el tono y el volumen del sonido producido.

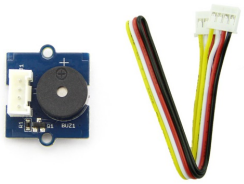

Figura A.3: Grove Buzzer V1.1 .

- La PYNQ-Z2 utilizada es una tarjeta de desarrollo basada en la arquitectura Zynq de Xilinx que permite el desarrollo de proyectos de sistemas embebidos en Python y tiene un procesador ARM Cortex-A9 y una FPGA para procesamiento y programación lógica personalizada. También tiene conectividad Ethernet y USB y varios puertos I/O para conectar dispositivos externos. Es versátil y puede ser utilizada en una amplia gama de proyectos de sistemas embebidos, como control de motores, procesamiento de imágenes y señales, y aplicaciones de IoT.

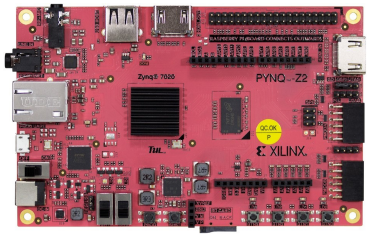

Figura A.4: PYNQ-Z2.

- Para realizar las fotografías hemos usado una cámara web VGA con resolución 640x480 píxeles que se conecta a un puerto USB 2.0. El uso de una cámara de prestaciones relativamente bajas se debe a la intención de simular un escenario real y poder generalizar los resultados.

Figura A.5: WebCam.

A.2. El conjunto de datos de creación propia

El conjunto de datos que hemos creado se trata de imágenes de naranjas tomadas con la cámara A.8 usada en el caso práctico. Las imágenes son de dimensiones 288x352

píxeles y tres canales (RGB), aunque estas después se han redimensionado a 288x288 píxeles dadas las limitaciones actuales de la librería Anomalib.

El conjunto de datos original tiene 33 imágenes de naranjas normales y siete imágenes de naranjas anómalas. Por otro lado, tras la producción sintética de nuevas imágenes se cuenta con un total de 352 imágenes normales y 72 anómalas.

Figura A.6: Ejemplos de mandarinas normales.

Figura A.7: Ejemplos de naranjas anómalas.

Figura A.8: Ejemplos de naranjas normales y anómalas tras aplicar *data augmentation*

Bibliografía

[1] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. pages 1310–1315, 2016.

[2] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. April 2019.

[3] Gregory Druck, Burr Settles, and Andrew McCallum. Active learning by labeling features. pages 81–90, August 2009.

[4] Alex Ratner. Weak supervision: The new programming paradigm for machine learning, Jul 2017.

[5] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. November 2019.

[6] Unsupervised Learning. In *Unsupervised Learning: Foundations of Neural Computation*. The MIT Press, 05 1999.

[7] D.E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, 1987.

[8] J. Jabez and B. Muthukumar. Intrusion detection system (ids): Anomaly detection using outlier detection approach. *Procedia Computer Science*, 48:338–346, 2015. International Conference on Computer, Communication and Convergence (ICCC 2015).

[9] Yang Yu, Jun Long, and Zhiping Cai. Network intrusion detection through stacking dilated convolutional autoencoders. *Security and Communication Networks*, 2017:1–10, 2017.

[10] Waleed Hilal, S. Andrew Gadsden, and John Yawney. Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, 193:116429, 2022.

[11] PricewaterhouseCoopers. Pwc's global economic crime and fraud survey 2022.

[12] Samaneh Sorournejad, Zahra Zojaji, Reza Ebrahimi Atani, and Amir Hassan Monadjemi. A survey of credit card fraud detection techniques: Data and technique oriented perspective. November 2016.

[13] Justin M. Johnson and Taghi M. Khoshgoftaar. Medicare fraud detection using neural networks. *Journal of Big Data*, 6(1), July 2019.

[14] Lewis Morris. Combating fraud in health care: An essential component of any cost containment strategy. *Health Affairs*, 28(5):1351–1356, September 2009.

[15] Samuel G Finlayson, Hyung Won Chung, Isaac S Kohane, and Andrew L Beam. Adversarial attacks against medical deep learning systems. April 2018.

[16] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, January 2017.

[17] Maximilian E Tschuchnig and Michael Gadermayr. Anomaly detection in medical imaging – a mini review. August 2021.

[18] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009.

[19] Dietram A. Scheufele and Nicole M. Krause. Science audiences, misinformation, and fake news. *Proceedings of the National Academy of Sciences*, 116(16):7662–7669, 2019.

[20] Xiao Sun, Chen Zhang, Shuai Ding, and Changqin Quan. RETRACTED ARTICLE: Detecting anomalous emotion through big data from social networks based on a deep learning method. *Multimedia Tools and Applications*, 79(13-14):9687–9687, January 2018.

[21] James Manyika, Michael Chui, Jacques Bughin, Richard Dobbs, Peter Bisson, and Alex Marrs. Disruptive technologies: Advances that will transform life, business, and the global economy, Feb 2020.

[22] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys Tutorials*, 20(4):2923–2960, 2018.

[23] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. January 2019.

[24] Peter T. Yamak, Li Yujian, and Pius K. Gadosey. A comparison between arima, lstm, and gru for time series forecasting. In *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, ACAI 2019, page 49–55, New York, NY, USA, 2019. Association for Computing Machinery.

[25] B Ravi Kiran, Dilip Mathew Thomas, and Ranjith Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. January 2018.

[26] B. Boghossian and J. Black. The challenges of robust 24/7 video surveillance systems. In *The IEE International Symposium on Imaging for Crime Detection and Prevention, 2005. ICDP 2005.*, pages 33–38, 2005.

[27] Henry Y.T. Ngan, Grantham K.H. Pang, and Nelson H.C. Yung. Automated fabric defect detection—a review. *Image and Vision Computing*, 29(7):442–458, 2011.

[28] Yiping Gao, Xinyu Li, Xi Vincent Wang, Lihui Wang, and Liang Gao. A review on recent advances in vision-based defect recognition towards industrial intelligence. *Journal of Manufacturing Systems*, 62:753–766, 2022.

[29] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D Cubuk, and Ian J Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. April 2018.

[30] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. CFLOW-AD: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. July 2021.

[31] David M.J. Tax and Robert P.W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, January 2004.

[32] Jihun Yi and Sungroh Yoon. Patch SVDD: Patch-level SVDD for anomaly detection and segmentation. June 2020.

[33] Kun Yang, Samory Kpotufe, and Nick Feamster. An efficient one-class SVM for anomaly detection in the internet of things. April 2021.

[34] Bernhard Schölkopf, Robert C. Williamson, Alexander J. Smola, John Shawe-Taylor, and John C. Platt. Support vector method for novelty detection. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 582–588. The MIT Press, 1999.

[35] Yue Zhao, Zain Nasrullah, and Zheng Li. PyOD: A python toolbox for scalable outlier detection. January 2019.

[36] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang, and Yue Zhao. ADBench: Anomaly detection benchmark. June 2022.

[37] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003.

[38] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, page 93–104, New York, NY, USA, 2000. Association for Computing Machinery.

[39] Yue Zhao and Maciej K. Hryniewicki. XGBOD: Improving supervised outlier detection with unsupervised representation learning. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2018.

[40] Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016.

[41] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986.

[42] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: unbiased boosting with categorical features. June 2017.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2014.

[45] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. ACM Press, 2009.

[46] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[47] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006.

[48] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 06 2014.

[49] Xavier Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 01 2010.

[50] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. MLSDA'14, page 4–11, New York, NY, USA, 2014. Association for Computing Machinery.

[51] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[52] Samet Akcay, Dick Ameln, Ashwin Vaidya, Barath Lakshmanan, Nilesh Ahuja, and Utku Genc. Anomalib: A deep learning library for anomaly detection. February 2022.

[53] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. June 2021.

[54] Jiawei Yu, Ye Zheng, Xiang Wang, Wei Li, Yushuang Wu, Rui Zhao, and Liwei Wu. FastFlow: Unsupervised anomaly detection and localization via 2D normalizing flows. November 2021.

[55] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. PaDiM: A patch distribution modeling framework for anomaly detection and localization. November 2020.

[56] Guodong Wang, Shumin Han, Errui Ding, and Di Huang. Student-teacher feature pyramid matching for anomaly detection. March 2021.

[57] Nilesh A Ahuja, Ibrahima Ndiour, Trushant Kalyanpur, and Omesh Tickoo. Probabilistic modeling of deep features for out-of-distribution and adversarial detection. September 2019.

[58] Samet Akcay. Deep feature kernel density estimation, 2022.

[59] Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. The MVTec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *International Journal of Computer Vision*, 129(4):1038–1059, January 2021.

[60] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[61] Papers with code - towards total recall in industrial anomaly detection.

[62] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. CoCa: Contrastive captioners are Image-Text foundation models. May 2022.

[63] Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, and Quoc V Le. Meta pseudo labels. March 2020.