eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO DE OBJETOS EN LA INDUSTRIA 4.0

Alumno/Alumna: **Imaz, Marin, Jon**

Director/Directora (1): Espinosa, Acereda, Jon Koldobika

Curso: 2017-2018

Fecha: Bilbao, 17 de julio de 2018

Resumen

La inteligencia artificial está causando una revolución tecnológica en todo el mundo. Este nuevo campo tecnológico aporta una gran cantidad de beneficios a nuestra sociedad tanto económica como socialmente.

Este proyecto se centra en el estudio y análisis de las diferentes alternativas disponibles para el desarrollo de un software de visión artificial con el fin de su futura implementación en la industria 4.0. El reconocimiento de objetos y la anotación de los mismos en la siguiente revolución industrial serán vitales para el correcto funcionamiento de las máquinas en la producción. A lo largo del desarrollo del proyecto, se detallará el proceso seguido para la final anotación de objetos pertenecientes a las áreas más afectadas por esta futura revolución.

Laburpena

Adimen artifizialak, teknologia aldaketa handia eragiten ari da mundu osoan zehar. Arlo teknologiko berri honek, onura izugarriak dakartza ekonomia eta gizarterako.

Proiektu hau, etorkizunean 4.0 industrian inplementatzeko ikusmen-artifizialeko software baten disenurako erabilgarri dauden teknologi eta alternatiba desberdinen ikasketa eta analisian zentratuko da. Objetu desberdinen antzemana eta hauen oharrak hartzeak datorren industria-iraultzan izugarrizko garrantzia izango du, produkzioan makinen funtzionamendu egokia ziurtatzeko. Proiektuan zehar, iraultza-industrialean kalte gehien jasoko duten lan-arloetan ohar hartzeak lortzeko jarraitutako prozesua zehaztuko da.

Abstract

Artificial intelligence is causing a technological revolution around the world. This new technological field brings a lot of benefits to our society both economically and socially.

This project is focused on the study and analysis of the different alternatives available for the development of an artificial vision software with the aim of its future implementation in Industry 4.0. The recognition of objects and the annotation of them in the next industrial revolution will be vital for the correct functioning of the machines in production. Throughout the development of the project, the process followed for the final annotation of objects belonging to the areas most affected by this future revolution will be detailed.

Índice

Contenido

Lista de ilustraciones

Lista de tablas

Introducción

Dado el interés y el avance de las tecnologías basadas en la anotación e identificación de datos u objetos en los últimos años, en este proyecto se tratará de dar solución a un problema de reconocimiento de objetos en tiempo real mediante diferentes algoritmos y aprendizaje máquina.

Para lograr el objetivo, se deberá seguir un proceso que este documento recogerá y detallará en diferentes apartados. Entre ellos, la explicación y comprensión de un estudio realizado sobre la teoría del aprendizaje máquina con el que se procederá a crear el diseño de un software que intente dar pie a la solución del problema.

La solución propuesta al problema sería realmente difícil de entender sin habernos ubicado en el contexto en el que se sitúan las tecnologías basadas en la inteligencia artificial. Por lo que se le dará especial importancia a esta parte del proyecto en la que se ubicarán las tecnologías basadas en la predicción y el reconocimiento en la sociedad actual. Para ello, se hará también un estudio y análisis del estado del arte, en el cual se realizará especial hincapié en las últimas teorías basadas en “machine learning” así como los últimos métodos utilizados tales como “deep learning” y las redes neuronales.

Un análisis a nivel social, económico y técnico nos transmitirá la verdadera importancia del tema a tratar y nos ayudará a entender la trascendencia que en las próximas décadas alcanzará la materia.

Además, se tratará de analizar las diferentes y últimas tecnologías disponibles para el diseño del software. Tras estos análisis, se detallarán los criterios tomados para la elección de la tecnología que se usará para el desarrollo del software. Después de esta breve explicación, se describirá la metodología seguida para la creación del software, así como la planificación llevada a cabo durante la creación y el proceso del proyecto.

Al final del documento, se ubicarán los anexos en los que se representarán los diferentes resultados obtenidos y parte del código elaborado en el transcurso del proyecto.

Contexto

El impacto que ha tenido la inteligencia artificial en todos los ámbitos de nuestra vida cotidiana es cada vez más reconocido mundialmente. En cuanto a progreso y a las mejoras en calidad de vida que potencialmente puede causar, la inteligencia artificial se está convirtiendo en la tecnología más importante en la historia de la humanidad. Hoy en día su auge es de una magnitud impresionante y ya es posible encontrarla en todo tipo de tecnologías: automóviles autónomos, videojuegos, software de reconocimiento de objetos, diagnósticos médicos…

Una vez habiendo situado la inteligencia artificial, vamos a centrarnos concretamente en el reconocimiento de objetos mediante el aprendizaje de máquina o "machine learning". Un área cada vez más importante en el mundo tecnológico puesto que ofrece muchas aplicaciones en un estilo de vida en el que la tecnología se ha convertido en algo imprescindible.

Dentro del ámbito de la inteligencia artificial, la visión artificial es uno de los campos de mayor crecimiento en los últimos años. Gracias a diferentes técnicas de reconocimiento, hoy en día es usada en múltiples actividades del mundo laboral, como en la medicina para la comparación de radiografías y resonancias magnéticas, o en la biología debido a la distinción de aplicaciones microscópicas y macroscópicas.

La visión artificial es una rama de la inteligencia artificial que tiene como objetivo modelar matemáticamente los procesos de percepción visual en los seres vivos y generar programas que permitan simular estas capacidades visuales por una computadora. Las aplicaciones más representativas o habituales podrían ser la detección facial, con la que hoy en día es posible desbloquear el teléfono móvil o incluso verificar nuestra identidad en aeropuertos e instituciones públicas, o la identificación de matrículas por las cámaras de tráfico de la Dirección General de Tráfico.

Además de las aplicaciones más usuales, la visión artificial es ya una realidad en la industria. Actividades como el guiado de las piezas, la identificación, la medición o la inspección de estas mismas son realizadas por aplicaciones de visión artificial.

Otro de los ejemplos futuros que podríamos poner es el de la llamada 4ª revolución industrial que se ha empezado a desarrollar a principios de este siglo XXI: el desarrollo de sistemas robóticos con inteligencia artificial.

Según [BENE13], el 47 por ciento de los empleos actuales están en riesgo de ser víctimas de la automatización en los próximos 10 o 20 años. En un futuro no muy lejano, los robots sustituirán a los humanos en abundantes empleos y profesiones y en todos los casos, será necesario que estos robots hagan uso de una visión artificial para poder interactuar con el entorno mediante el reconocimiento de los diferentes objetos presentes en su campo visual.

El estudio [BENE13], especificó los empleos más susceptibles a la robotización.

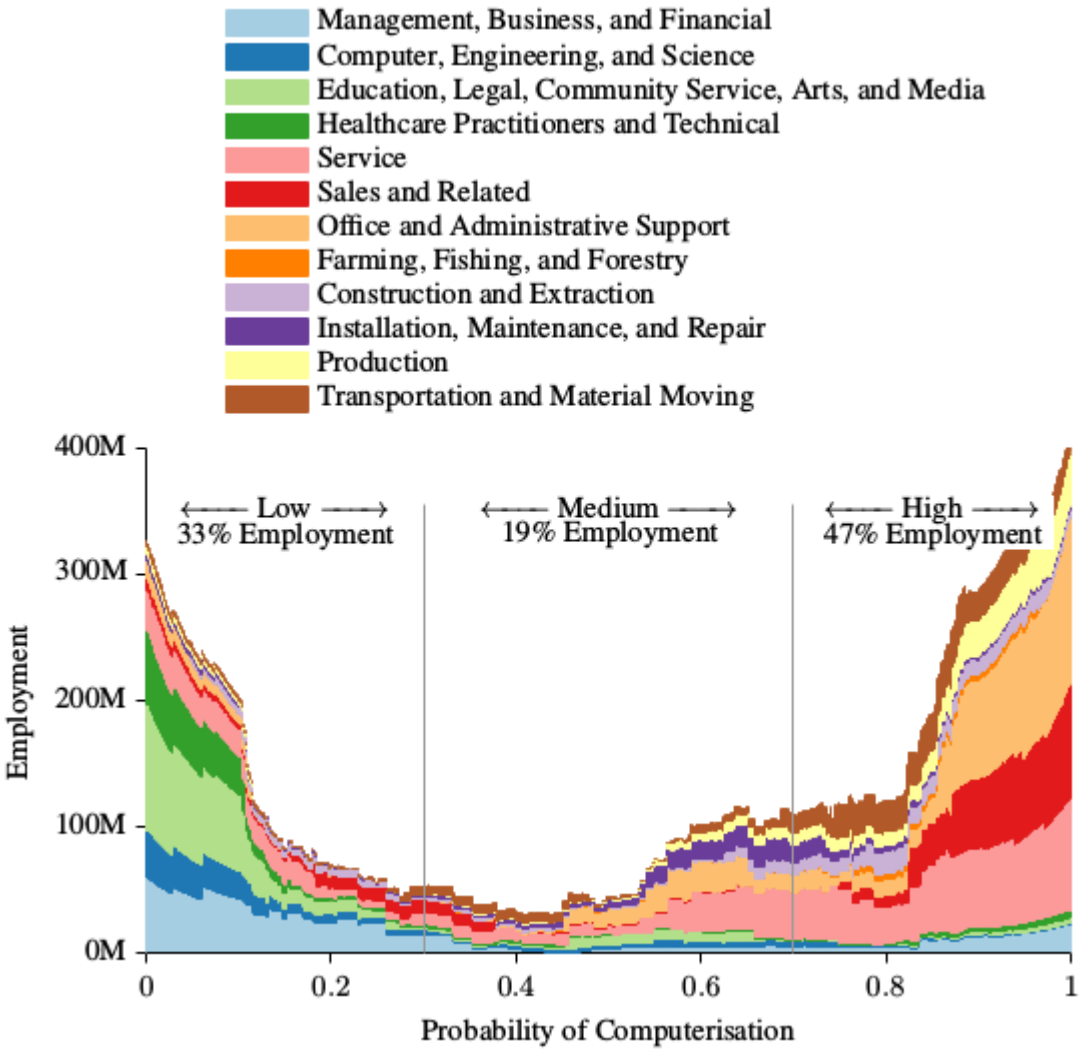


Ilustración 1: La distribución de BLS 2010 de empleo ocupacional sobre la probabilidad de informatización, junto con la distribución del trabajo en base al peligro que supone cada sector para la salud en el total del empleo de los Estados Unidos, [BENE13]

En la ilustración 1, podemos observar como los empleos más peligrosos son los candidatos más probables a ser sustituidos por robots o inteligencia artificial, mientras que los empleos menos nocivos para la salud quedarán prácticamente intactos en los próximos años. Cabe observar que el 47 por ciento de los empleos tienen un alto riesgo de ser sustituidos por máquinas por lo que la importancia del avance de la inteligencia artificial es inmensa.

Además, las recientes inversiones de las multinacionales y de las empresas más poderosas del sector tecnológico y concretamente del sector de la inteligencia artificial, junto con la creación en 2016 de la "Asociación para la Inteligencia Artificial para beneficiar a las personas y la sociedad" por parte de Google, Facebook, Microsoft, IBM y Amazon, y a la que más tarde se uniría Apple, son grandes indicios de la importancia que tendrá esta área en las siguientes décadas.

Alcance y objetivos

En este apartado se describirán tanto los objetivos propuestos del proyecto como el alcance del mismo. Se detallarán con la mayor precisión posible los puntos clave y áreas de conocimiento que se van a estudiar y que formarán parte de la solución.

El primer objetivo de este proyecto es la investigación y el estudio del área de conocimiento del "machine learning". El análisis de las diferentes tecnologías y métodos creados y estudiados por diferentes estudios e investigaciones y la comparación de los resultados de cada una de ellas.

En el campo del "machine learning" o el aprendizaje de máquina, se estudiarán los diferentes tipos de aprendizaje, las distintas categorías y los múltiples problemas de clasificación correspondientes a cada categoría.

Una vez finalizado este estudio, el proyecto se enfocará en el análisis de las diferentes alternativas con las que resolver el problema. En esta área se realizará un estudio de las distintas categorías y de las diferentes aplicaciones de cada una de ellas centrándose sobre todo en la visión por computador o "computer vision".

Tras el análisis de los diferentes métodos, se procederá a la creación de un software mediante el método elegido para obtener unos resultados aceptables. Después de la creación del mismo, se hará una comparación con los métodos anteriormente analizados. Para ello, se utilizarán y analizarán diferentes herramientas con las que poder realizar un aprendizaje para la consiguiente clasificación de los objetos.

Otro de los objetivos de este proyecto es la detección de objetos en diferentes escenarios como la casa o la ciudad. La detección de objetos será fundamental en la visión artificial que implementarán los robots que sustituyan a humanos en los empleos. Tal y como hemos visto en el anterior apartado, los empleos más susceptibles al cambio en la revolución industrial tienen una gran relación con el uso de transportes. Transportistas, trabajadores del sector primario, pesca y agricultura básicamente, y la construcción sufrirán los cambios más grandes, por lo que nos centraremos en que el reconocimiento de este tipo de objetos (coches, trenes, motocicletas o barcos) sea el principal objetivo.

Después de la pertinente detección, la finalidad del proyecto es la clasificación y la anotación de los mismos. La anotación será de gran ayuda para la interacción de los robots con el medio que los rodea y ayudará a implantar estas nuevas tecnologías en la sociedad.

Por lo tanto, el objetivo principal del proyecto es identificar y clasificar objetos en tiempo real, así como la anotación de los mismos con el fin de lograr implementar el software en futuros robots y mejorar el movimiento, la exactitud y la precisión de estos.

Beneficios del proyecto

En todo proyecto de ingeniería es de vital importancia el análisis de los beneficios que presentará el trabajo a desarrollar. Por eso mismo, se dará pie a una breve explicación de los beneficios que aportará el proyecto a nivel técnico, económico y social.

Nivel técnico

El gran objetivo del proyecto es la detección y anotación de los objetos en el medio mediante métodos innovadores de detección de objetos. Al tratarse de una materia reciente y en la cual los avances hasta el momento no han hecho más que abrir las fronteras de la investigación en este ámbito, las posibilidades de lograr nuevos y mejorados métodos son realmente altas.

El análisis de los diferentes métodos existentes proporcionará una base de conocimiento con la que empezar a crear nuevos métodos, descartando errores cometidos anteriormente e incluyendo los avances conseguidos en las recientes investigaciones. Además, el uso de los procedimientos con mejores resultados, nos será de gran ayuda para la solución final.

La implementación de esta nueva tecnología en la industria 4.0, también llamada industria inteligente, asegura el éxito de técnicas innovadoras de visión artificial y “machine learning”.

Nivel económico

La introducción del software en máquinas y la sustitución de estas mismas por humanos en los diferentes empleos, creará un gran impacto económico. La cuarta revolución industrial es y ha sido tema fundamental del Foro Económico Mundial, en el cual se ha apostado por que los países asiáticos se volverán más fuertes económicamente debido a sus ya habituales inversiones en Inteligencia Artificial.

Con la llegada de la inteligencia artificial a la industria, los países ya industrializados se verán afectados en mayor nivel puesto que gran parte de su empleo reside en industrias. Por eso, expertos en la materia predicen que los países en vías de desarrollo serán los grandes beneficiados de esta revolución. Gracias a esto, surgirán nuevas oportunidades en países en crecimiento que podrían dar un reparto más equitativo de bienes.

Un ejemplo muy claro es el ofrecido por el "World Bank Development Report" en 2016 y explicado en [BENE16].

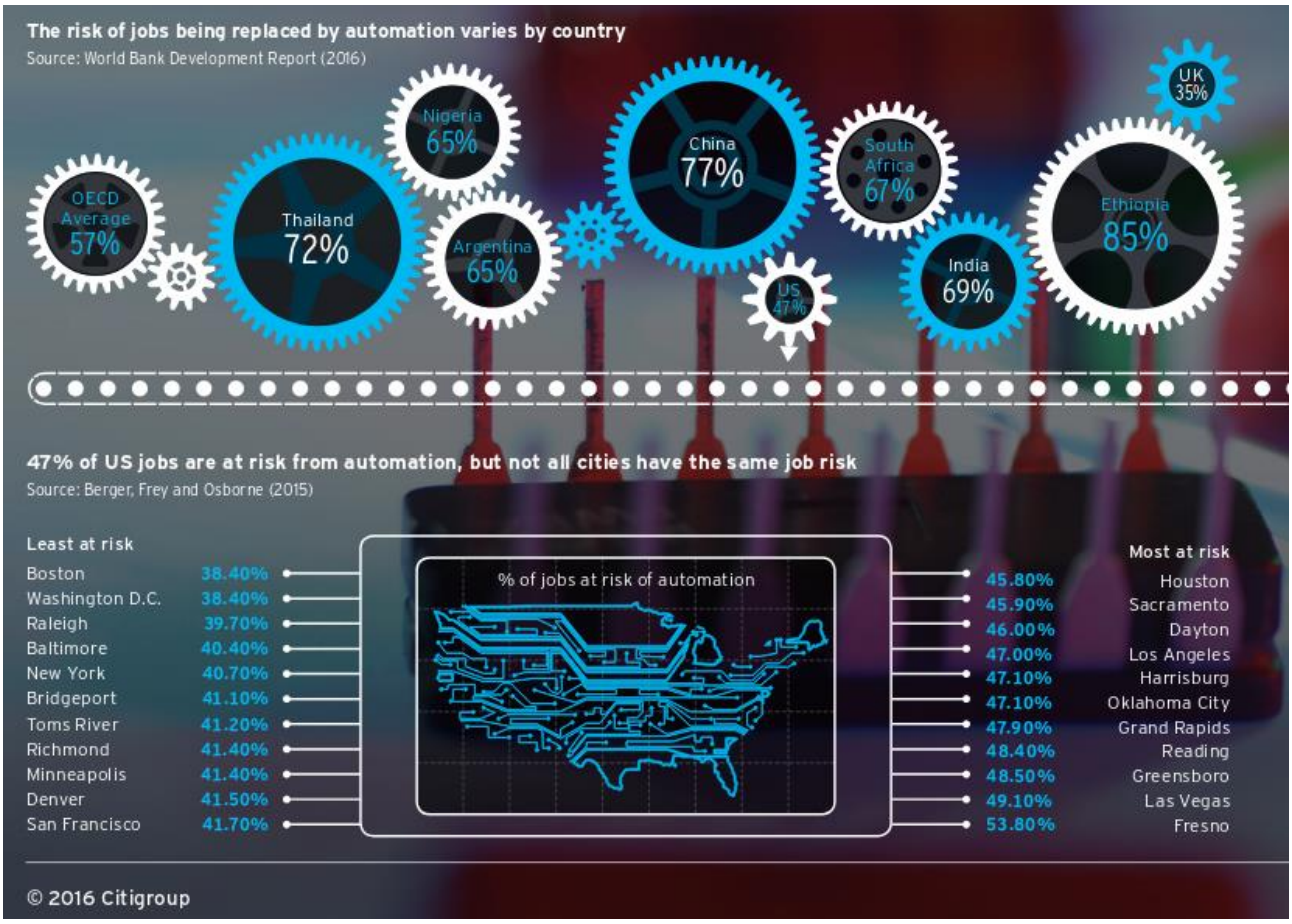


Ilustración 2: Análisis del riesgo de la sustitución de empleos por máquinas por el World Bank Development Report en 2016 [BENE16]

Tal y como se puede observar en la ilustración 2, los países en los cuales las principales fuentes económicas son el sector primario, como en gran parte de África y países industrializados masivamente como China o India, serán los grandes afectados por la nueva generación industrial.

Podríamos pensar que este cambio solo mejoraría el nivel económico de grandes empresarios y empresas tecnológicas y empobrecería aún más a las clases media y baja, pero gracias al cambio social que la sociedad se verá obligada a tomar, gran parte de estas clases trabajadoras se verían gratamente afectadas por la transformación.

Nivel social

Si bien el impacto económico de la revolución industrial será inmenso, socialmente la consecuencia de este gran cambio será aún más grande si cabe. La desaparición de muchos empleos, según el estudio anteriormente mencionado de Carl Benedikt Frey y Michael A. Osborne de la universidad de Oxford, 702 profesiones actuales están declaradas altamente automatizables. La creación de nuevas profesiones adaptadas a la situación que se vivirá en un futuro próximo harán que la sociedad tenga que dar un giro completo hacia los estudios y los empleos basados en las nuevas tecnologías.

Según [W.1], nuevas profesiones verán la luz, tales como:

- Analistas y programadores de Internet de las cosas
- Arquitectos de realidad virtual / aumentada
- Diseñadores de órganos
- Terapeutas de empatía artificial
- Ingenieros de nanorobots médicos
- Creativos emocionales / gestión de relaciones virtuales
- ...

Las habilidades creativas, innovadoras y colaborativas tomarán un gran papel en el mundo empresarial, por lo que la educación también sufrirá un cambio acorde al vivido por el mundo laboral y la enseñanza pasará a ser mucho más práctica y variada.

Junto con el cambio en el mundo laboral, la relación interpersonal sufrirá una modificación tremenda, ya que el sector de los servicios se verá muy afectado por la robotización y la sociedad deberá aprender a relacionarse con máquinas en vez de con personas.

Estado del arte

En cuanto al concepto del estado del arte, se expondrán los puntos más relevantes de la visión artificial, así como la transición que ha sufrido el área hasta el punto en que nos ubicamos actualmente.

Tal y como se ha descrito anteriormente, el objetivo del proyecto se basa en la anotación de objetos. Mediante una imagen de test o "query", el fin de nuestro software será el de devolver etiquetas textuales descriptivas. Para ello, habrá multitud de posibilidades de etiquetado. Desde colores de forma semántica u objetos genéricos hasta conceptos más específicos como el tipo de objeto.

Este etiquetado dependerá del objetivo de la aplicación, por lo que en este caso la anotación nos devolverá nombres de los objetos que haya en el campo visual.

Otro de los problemas más comunes es la descripción del problema mediante algoritmos de visión artificial. Puesto que el estudio que se llevará acabo estará enfocado en las más recientes técnicas de reconocimiento, se deberá realizar un análisis previo a la elección del método a usar.

Existen 3 tipos de modelos para la anotación: modelos generativos, modelos discriminativos y, por último, modelos basados en vecinos.

En este proyecto, nos centraremos en los modelos discriminativos dado que son los más trabajados en el estado del arte.

Modelos discriminativos

Los modelos discriminativos, se basan en modelar de forma individual las etiquetas que pueden ser predichas usando clasificadores entrenados para ello.

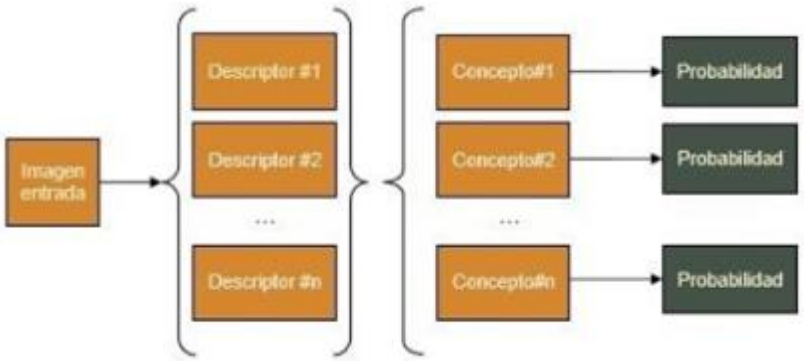


Ilustración 3: Modelo discriminativo básico de anotación automática de imágenes, [SERO16]

El primer paso del proceso será la extracción de las características visuales de la imagen en base a algoritmos y descriptores visuales.

En segundo lugar, se deberá aplicar la información lograda a los clasificadores de etiquetas para determinar la presencia o ausencia de las mismas.

Una vez tengamos la información, la regresión logística es un algoritmo utilizado para la predicción del resultado de una variable categórica en función de variables independientes. Este método es utilizado para problemas de clasificación binaria y clasificación multiclase con fronteras lineales.

Otro de los algoritmos más utilizados para la clasificación, es el árbol de decisiones. Este algoritmo, hace una correspondencia entre las características de una instancia o un elemento de un conjunto de datos, y conclusiones sobre el valor de dicha instancia. La variable de destino, puede tomar un conjunto finito de valores. En estos árboles de decisión, las ramas representan el conjunto de características que conducen a unas etiquetas de clase que son representadas por las hojas de este árbol.

Para realizar la clasificación de objetos, se podría hacer uso del algoritmo Support Vector Machine. Los datos son representados mediante puntos en el espacio y gracias a este algoritmo una línea separa las clases representadas. Los datos serán clasificados en el espacio de una categoría u otra dependiendo del lado de la línea al que pertenecen.

Tras estos primeros dos pasos, se podrían dar otros para una mejor unión de la información:

- métodos de agrupación de los descriptores visuales
- agrupación de las salidas de los clasificadores para que posean influencia entre sí
- añadir información de contexto para ayudar a los clasificadores

El objetivo final será lograr modelar las etiquetas de forma efectiva, ya sean objetos, conceptos genéricos o elementos abstractos. Existen muchas aproximaciones para el modelado de las clases, pero en general, se agrupan en modelos basados en componentes globales y basados en componentes locales.

La diferencia más significativa entre estos dos tipos de modelos es que mientras que las primeras se analiza toda la imagen, en los modelos basados en componentes locales se realiza una detección de puntos o regiones características y se modela la imagen en base a esas regiones que en todas las instancias del concepto deben ser parecidas.

Modelos basados en componentes globales

En primer lugar, analizaremos la línea de investigación de detección de objetos basada en componentes globales. Una de los primeros estudios que abrieron camino a futuras investigaciones, fue la propuesta por [PAPA00]. En esta propuesta, se propone utilizar un modelo descriptivo para definir la clase de los objetos. Así se podrán describir forma, color o textura de un objeto. Para conseguir estos resultados, las imágenes de test son procesadas mediante los wavelets de Haar, los cuales son una cierta secuencia de funciones, y las salidas de estos filtros son usadas junto con una "Support Vector Machine" (SVM), un conjunto de algoritmos de aprendizaje supervisado, para modelar objetos y generar la clase.

Investigaciones posteriores como [VIOL01] [VIOL04], las cuales mejoraban la rapidez en la detección de objetos, también se basaban en los wavelets de Haar. Este nuevo algoritmo propuesto por Viola y Jones, se basa en una nueva representación de las imágenes, definiéndolas como imágenes integrales, permite un procesamiento más rápido de los wavelets de Haar. El algoritmo de boosting utilizado, permite una selección de las características más destacadas de un objeto en base a la respuesta de los wavelets. De esta manera, cada clase solo tiene la información más discriminativa y aumenta el rendimiento del sistema. Con este estudio, lograron que después de 10 años del mismo, aún se utilice el algoritmo propuesto, puesto que es uno de los más rápidos en la detección de objetos.

En el estudio [LIEN02] se propuso una mejora del sistema establecido anteriormente. El cual sugiere la utilización de un conjunto diferente de los wavelets. Junto con los propuestos en [VIOL01], añade los wavelets que tengan una rotación de 45º, logrando una ligera mejora.

En las investigaciones [SERR05] [SERR07] se propone una utilización basada en las características inspiradas en el córtex visual humano. Hacen uso de una serie de cuatro etapas en las cuales se simula el proceso de identificación de objetos del ser humano. Con unos resultados increíbles: 98,2% en detección de caras, 98% en detección de motos… Aun así, el tiempo de computación del mismo supone un gran problema puesto que es inmenso. Este problema es creado por la concentración del proyecto en la detección de objetos monolíticos con una estructura común como las caras o los coches. Por esta causa, este desarrollo obtuvo muchos problemas con objetos cambiantes como las personas, por ejemplo.

La solución propuesta por diferentes expertos se basa en centrarse en analizar la información local de los objetos agrupándola de forma global en base a histogramas. El estudio [DALA05] se fundamenta en histogramas locales de las orientaciones de los gradientes de la imagen. La idea principal del proyecto es que la forma local y la apariencia de un objeto puede ser caracterizada por la distribución de los gradientes locales o por las direcciones de los bordes, incluso sin un conocimiento preciso de la posición de estos mismos.

Para ello, se divide la imagen de test en pequeñas regiones llamadas celdas, y se computan el histograma de las direcciones del gradiente además de tener en cuenta las variaciones de la iluminación. Generando así descriptores HOG ("Histogram Oriented Gradients" o Histograma de Gradientes Orientados) que alimentarán la SVM que generalizará y detectará las clases de objetos. Este tipo de clasificación obtuvo un resultado magnífico, obteniendo un 100% de aciertos en personas.

Un estudio posterior, [ZHU06], mejora el anterior sistema aumentando la velocidad de procesamiento gracias al uso de algoritmos de boosting. La velocidad de procesamiento pasó a ser de 500ms a 26ms por lo que fue un gran avance para las investigaciones posteriores.

Otro método utilizado en [SHOT06][SHOT08] es el uso de textons, microestructuras fundamentales en imágenes naturales. Estos descriptores permiten dar información de la forma del objeto y de su textura o color generando así un diccionario de textons. Para la creación de estos diccionarios, se aplican 17 filtros Gaussianos, derivadas de Gaussianas y Laplacianos de Gaussianas en las imágenes de test y se realiza un agrupamiento de K-means, partición de un conjunto de n observaciones en k grupos, en el que cada observación pertenece al grupo cuyo valor medio es más cercano. Cada pixel de la imagen se asigna al cluster o agrupación más cercana obteniendo un mapa llamado "texton map". Está técnica permite detectar y segmentar los objetos con una media de acierto de 88,6% y una disminución del tiempo de 30 segundos a 1 segundo.

Modelos basados en componentes locales

La mayoría de las investigaciones realizadas en cuanto a modelos discriminativos se centran en modelos basados en componentes locales.

Una clase de objetos está formada por una serie de características que aportan información local sobre el objeto y además están relacionadas entre sí físicamente.

Ilustración 4: Características locales de una imagen a anotar con el concepto "coche" y el concepto "cara"

Las características de la ilustración 4 son definidas en base a cierta información de la imagen que rodea el centro. Esta definición, dependerá del método utilizado. La relación entre las características también se puede modelar por lo que una clase se definirá por estas dos propiedades. En el caso de la detección de caras y mediante estas dos propiedades, al aparecer otra cara en la imagen, el método extraerá las mismas características que se hallarán en una posición similar. Así se podrá deducir que se ha detectado una cara.

A partir de esta breve explicación existen varios métodos diferenciables. El estudio [WEBE00] propone un modelo basado en una "constelación por partes". El cual se basa en el modelado de objetos mediante el uso de constelaciones flexibles de las características. Este sistema es capaz de identificar automáticamente las partes distintivas de un objeto entre las imágenes de test mediante un operador que se encarga de extraer ciertos patrones. A partir de estos patrones, se aplicarán técnicas de clustering, procedimiento de agrupación de una serie de vectores con criterios como la similitud o la distancia, que detectarán las diferentes partes de un objeto de forma automática. Este sistema, se encargará de la extracción del modelo estadístico de las partes obtenidas mediante el algoritmo de "expectación-maximización".

Otro de los estudios basados en la anterior explicación, pero centrado en la búsqueda de un sistema invariante a escala y a ligeras transformaciones es el propuesto por [FERG03]. Este proyecto sugiere la modelización de objetos como "constelaciones flexibles" de partes. Así, la modelización pasaría a ser probabilística que aportaría una mayor información de la clase y un mayor grado de acierto.
El problema de este tipo de proposiciones es el gran volumen de imágenes que requieren. Por eso mismo, [FEIF07] plantea un enfoque diferente del mismo concepto. El uso de un aprendizaje incremental permite al sistema aprender un modelo general con la necesidad de muy pocas imágenes. Una sola imagen proporcionaría un 70% de aciertos y tan solo 15 imágenes un 80%. Además, cuantas menos imágenes de test se necesiten, menos tiempo de entrenamiento será necesario.

Otro método perteneciente a este tipo de modelos es la detección de objetos mediante puntos característicos locales. En este caso, no será necesario el modelado de relaciones geométricas entre ellos. Al igual que en el anterior caso, también se aplica un operador que detecta puntos característicos de la imagen para describirlos mediante un descriptor. El descriptor se centrará en describir las regiones cercanas al objetivo de identificación haciendo una comparación de los puntos extraídos.

Existen otros métodos que definen el concepto de "vocabulario". A partir de este concepto, una clase de objetos son modelados en base a un vocabulario que describe partes de la clase. Por lo tanto, las etiquetas finales serán definidas por los conjuntos de palabras visuales localizadas en el vocabulario. Al igual que en los anteriores casos, para generar este vocabulario se utilizará

un operador que extraerá información de las imágenes de test. Por lo tanto, el vocabulario será creado en la fase de entrenamiento. Una de las propiedades de estos vocabularios es que tendrán mucha información que modele unas instancias, pero tendrán muy poca información en común con otro tipo de clases de objetos.

Otra de las propuestas en relación a los vocabularios, es la sugerida en [DORK05], en la que se propone una selección de las partes que mejor discriminen la clase para reducir la cantidad de información de los vocabularios. Cada clase se modelará con un reducido grupo de partes, pero estas serán las que mejor representen a la clase por lo que se lograrán tasas de acierto muy altas. Los resultados de este estudio reflejan un 99,08% de acierto en la detección de caras.

Una mejoría en la eficiencia de la representación de clases es la reutilización de las características. El entrenamiento de varias clases en el mismo modelo, invariante a la rotación y a la escala es posible mediante la utilización de un codebook que define todas las características de la clase. Estos codebooks, comparten las características en función de modelos probabilísticos aprendidos durante el entrenamiento. Este método obtiene un acierto del 94,7% en la clase coche.

Una mejoría del anterior método llegó junto con el estudio [OMME10], en el cual el objeto se descompone en diferentes parches de forma no supervisada. Los parches se agruparán posteriormente para generar las partes que componen el objeto. Esta reagrupación logra una mayor generalización ya que la comparación se hará en base a los grupos de parches en vez de a nivel de parches. Los parches permitirán la detección de otros objetos por lo que serán de gran ayuda cuando se modelen diferentes clases con el mismo algoritmo.

Otras investigaciones, [GU09] por ejemplo, se basan en las regiones que componen el objeto y no en las partes. Este método, propone la utilización de partes reales de objeto anteriormente segmentadas mediante el algoritmo. Las características de este tipo de modelos son preferibles, puesto que obtienen información natural del objeto.

Un gran avance en la materia llegó junto con el uso de las partes reales que componen lo objetos. La detección de cuerpos humanos mediante la previa detección de partes de los mismos como brazos, piernas o cabeza basándose en las características de Haar. Estas detecciones ofrecen robustez puesto que, si una parte del cuerpo no está visible, es posible una identificación mediante el resto de las partes.

En la actualidad, varios estudios apuntan hacia otros métodos nunca antes estudiados. [ENGE10] sugiere la utilización de un modelo que representa la forma de las partes en base a los modos de vibración de los elementos finitos. El objetivo de este sistema trata de encontrar la forma deseada por medio de una evaluación. De esta manera, consigue ser más inmune a la oclusión y a las variaciones estructurales. Para ello, se requiere un entrenamiento de las

posibles variaciones en la forma, pero como solo se pueden tener variaciones en 2 dimensiones, no es muy sólido frente a cambios de orientación.

Por último, cabe destacar la propiedad "Self-similarity". Una propiedad que permite ver el parentesco de las imágenes. Se trata de uno de los últimos avances en el tema y su objetivo es la comparación de las regiones locales de las imágenes contra la imagen global, también llamada Global Self-similarity. En las técnicas más modernas, se combina junto con HOG y otras técnicas como bag-of-words, método utilizado en el procesado del lenguaje con el fin de representar documentos ignorando el orden de las palabras, y GIST, Descriptor de imagen que representa el conjunto de una imagen en baja dimensionalidad, con el fin de generar un detector mejor.

Utilización del contexto

Aunque la detección de partes sea una de las líneas de investigación más destacadas, hay problemas que no se pueden solucionar con este método.

El contexto puede ser el factor adicional que permita diferenciar los objetos. La utilización del mismo mediante la identificación de la localización de las imágenes y el uso de estas para detectar los objetos presentes en ellas. Se han creado representaciones globales de las imágenes a testear que permiten obtener información acerca de su localización. Esta representación es llamada "esencia" o "GIST". Los resultados obtenidos son bastante completos con un 75% en detección de personas o un 90% en detección de edificios.

Mediante la combinación de este y de otros métodos mencionados anteriormente, se han logrado tasas de identificación bastante más altas. El estudio [LEE10] hace uso de relaciones entre objetos para detectar otros objetos desconocidos presentes en las imágenes. Esto permite aprender categorías de forma no supervisada, pero teniendo en cuenta como se crean las categorías iniciales.

Deep Learning

Se ha podido comprobar como los modelos discriminativos son capaces de detectar objetos independientes de forma aceptable pero el problema de estos modelos es que no existe una técnica válida para todo tipo de objetos.

Todo esto ha empezado a cambiar con el concepto del "deep learning" o aprendizaje profundo. Rina Dechter introdujo este concepto por primera en vez en el machine learning en el año 1986 y fue en el año 2000 cuando por primera vez el matemático Igor Aizenberg lo implementó en redes neuronales artificiales.

Esta nueva metodología se basa en el uso de arquitecturas compuestas por muchas capas y son capaces de representar funciones altamente no lineales para que un algoritmo automático pueda aprender conceptos con un nivel de abstracción muy alto. Un ejemplo de este método son las ya mencionadas redes neuronales, como la mostrada en la ilustración 5.

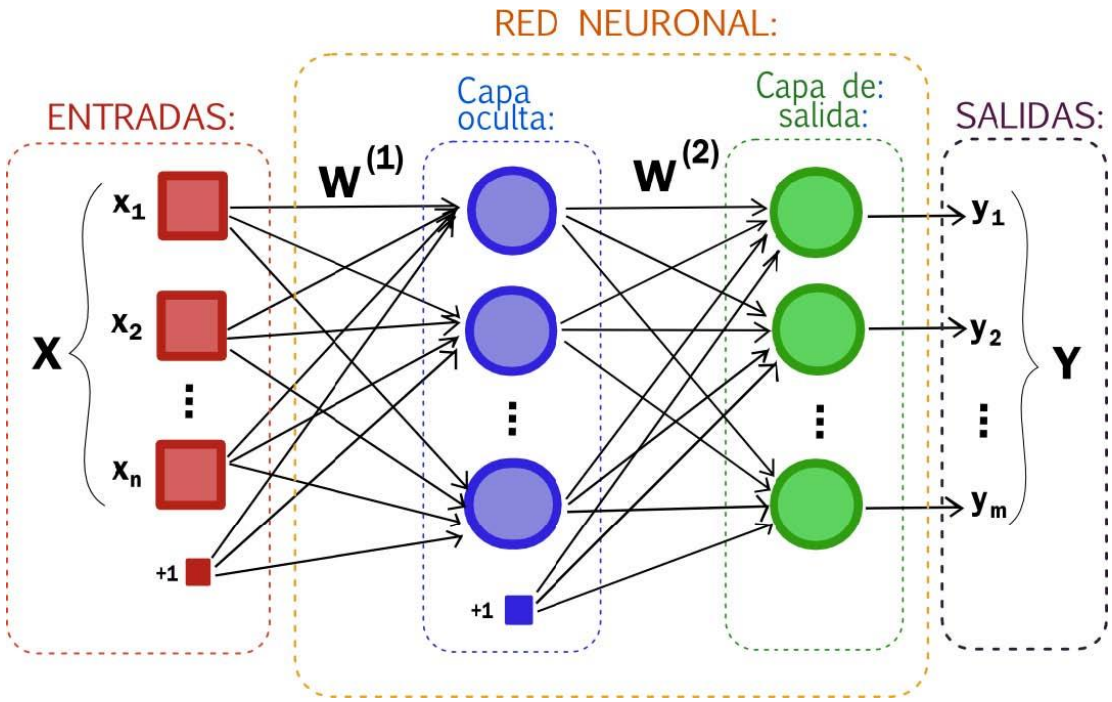


Ilustración 5: Ejemplo de una red neuronal

Una de las formas más habituales de entrenamiento de este tipo de redes en la llamada Backpropagation. Este método se basa en calcular el gradiente de una función de error sobre los diferentes pesos de la red y utilizar el error cometido y el gradiente para actualizar esos propios pesos. El problema de este algoritmo surgió cuando el aumento del número de capas produjo un acrecentamiento considerable en la tasa de error. A partir de entonces, se dejó de investigar en redes de muchas capas.

Además, con el paso de los años, se ha comprobado que el uso del Backpropagation en arquitecturas de aprendizaje profundo presenta varios problemas graves:

1. El gradiente se "diluye" en las últimas capas y la corrección del algoritmo es inapreciable por lo que no existe un entrenamiento efectivo en esas últimas capas
2. En arquitecturas de muchas capas el algoritmo se queda estancado en mínimos locales muy rápidamente
3. La importancia de datos correctamente anotados es inmensa

Entonces surgieron diferentes propuestas para solucionar estos problemas en las diferentes fases. Las propuestas más efectivas fueron el entrenamiento no supervisado o pre-entrenamiento en la primera fase y el refinamiento supervisado en la segunda fase.

Gracias a estas aproximaciones se han logrado grandes avances consiguiendo muy buenos resultados en arquitecturas profundas. De esta manera se produjo el nacimiento del área de conocimiento llamado Deep Learning. Esta área busca aprender diferentes niveles de abstracción de los datos para que, de manera global aporten una información útil de los mismos. La base de muchas investigaciones posteriores y de muchas arquitecturas propuestas, son los autoencoders, arquitecturas cuya forma básica posee dos capas.

Para empezar, hay que representar los datos de entrada en la capa intermedia de una forma compacta. Después, habría que regenerar los datos de entrada a partir de la representación anterior de la capa final. El objetivo de este tipo de aprendizajes es que el error producido entre la capa de entrada y la capa de salida sea el menor posible. A partir de este tipo de conceptos se han creado diferentes arquitecturas como la "Stacked Autoencoders".

La arquitectura Stacked Encoder es una red neuronal basada en múltiples capas de dispersos autoencoders en los cuales las salidas de cada capa están unidas a las entradas de la siguiente capa.

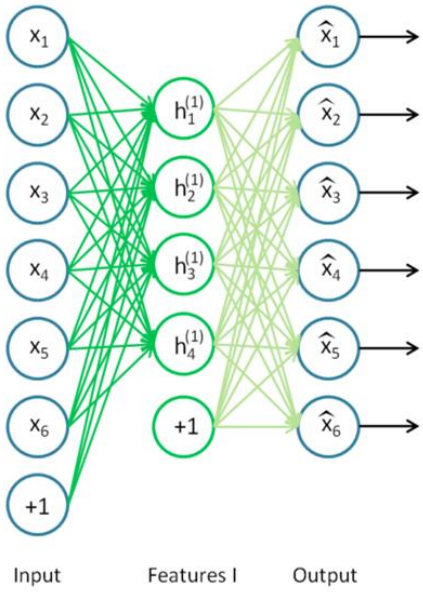


Ilustración 6: Ejemplo de una Stacked Autoencoder [W.2]

Otras técnicas como las llamadas “Restricted Boltzmann Machines” son redes neuronales regenerativas capaces de aprender distribuciones de probabilidad. Una extensión de este tipo de redes son las “Deep Belief Networks”, que permiten aprender representaciones jerárquicas de los datos de entrada de forma probabilística.

Otra de las redes más utilizadas en los últimos tiempos son las redes neuronales convolucionales o “Convolutional Neural Network”. Se trata de redes neuronales clásicas en las que los pesos de las neuronas de una capa se comparten. Por lo que todas las combinaciones de una capa se hacen con los mismos pesos.

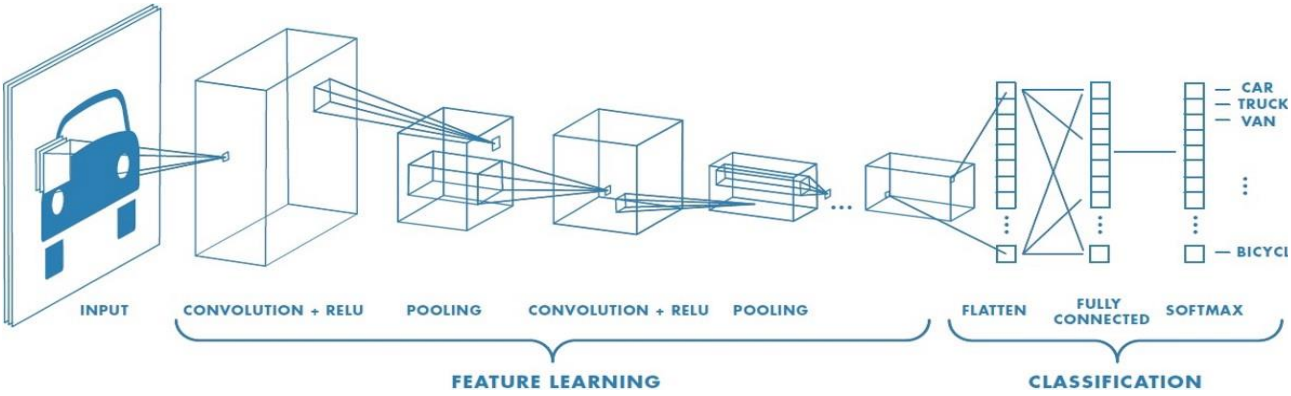


Ilustración 7: Ejemplo de una red neuronal convolucional, [W.3]

Redes neuronales convolucionales

Las redes neuronales convolucionales [W.3] son básicamente una red neuronal estándar que ha sido expandida en el espacio compartiendo ciertos pesos. Estas redes neuronales han sido diseñadas para el reconocimiento de imágenes mediante el uso de convoluciones internas. A continuación se dará una breve explicación a los métodos más usados en este tipo de redes.

Back-Propagation

Este método previamente mencionado en la explicación de los métodos dicriminativos, es un método simple para computar las derivadas parciales o gradientes de una función que tiene la forma de una función compuesta. En general, hay dos maneras de computar el gradiente de una función:

1. Diferenciación analítica: Una vez que se sabe la forma de la función, tan solo hay que calcular las derivadas parciales.
2. Diferenciación aproximada mediante el uso de la finita diferencia: En comparación con el anterior método, este es muy costoso puesto que el número de evaluaciones a realizar es *O(N),* donde *N* es el número de parámetros.

Gradiente descendente estocástico

Este método es propenso a estancarse en el mínimo local, dependiendo de la naturaleza de la función. En el caso de tener una función convexa, el algoritmo está destinado a lograr siempre la opción óptima. Dependiendo de los valores iniciales de la función, se seguirá un camino diferente hacia la solución en el cual la rapidez en la que converge dependerá del "learning rate" del algoritmo usado. Este criterio determinará si el algoritmo queda estancado en un mínimo local o es capaz de esquivarlo.

DROPOUT

Las redes neuronales profundas con un largo número de parámetros son sistemas de aprendizaje de máquina con una gran potencia. Aun así, el uso de este gran número de parámetros puede causar problemas serios por la gran cantidad de tiempo que necesitan para ser procesadas y posteriormente usadas. Para solucionar este tipo de problemas, Dropout es una técnica muy utilizada.

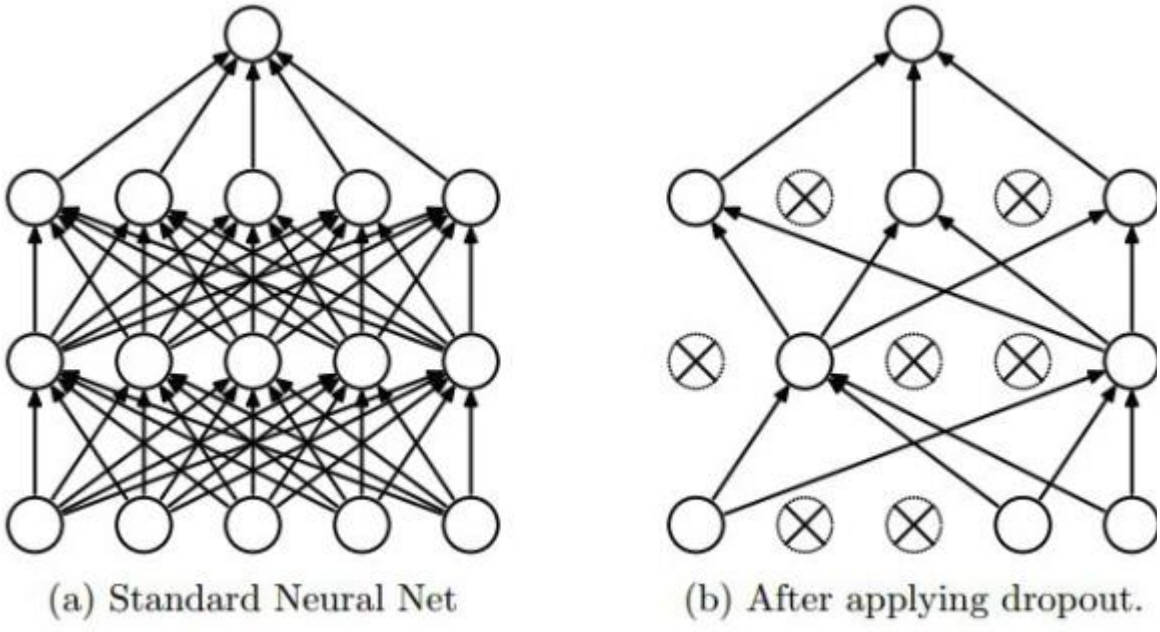


Ilustración 8: Ejemplo de una red neuronal antes y después de aplicar la técnica Dropout

La principal idea es la de borrar aleatoriamente unidades de la red junto con sus conexiones durante el entrenamiento. De esta manera, se previene la sobreadaptación de las unidades. Durante el entrenamiento, esta técnica crea muestras de las diferentes redes simplificadas. En el momento de probar la red es fácil aproximar el efecto causado por la simplificación de la red simplemente usando una red única con unos pesos menores que las creadas. Ha sido demostrado que este método mejora redes de aprendizaje supervisado, reconocimiento de voz, clasificación de documentos y biología computacional.

Max Pooling

Este método es un proceso de discretización basado en muestreos. El objetivo es la reducción de las muestras de la representación inicial, ya sean imágenes o matrices de salida de capas ocultas mediante la reducción de su dimensión y permitiendo las conjeturas sobre las características que contenían las subregiones procesadas.

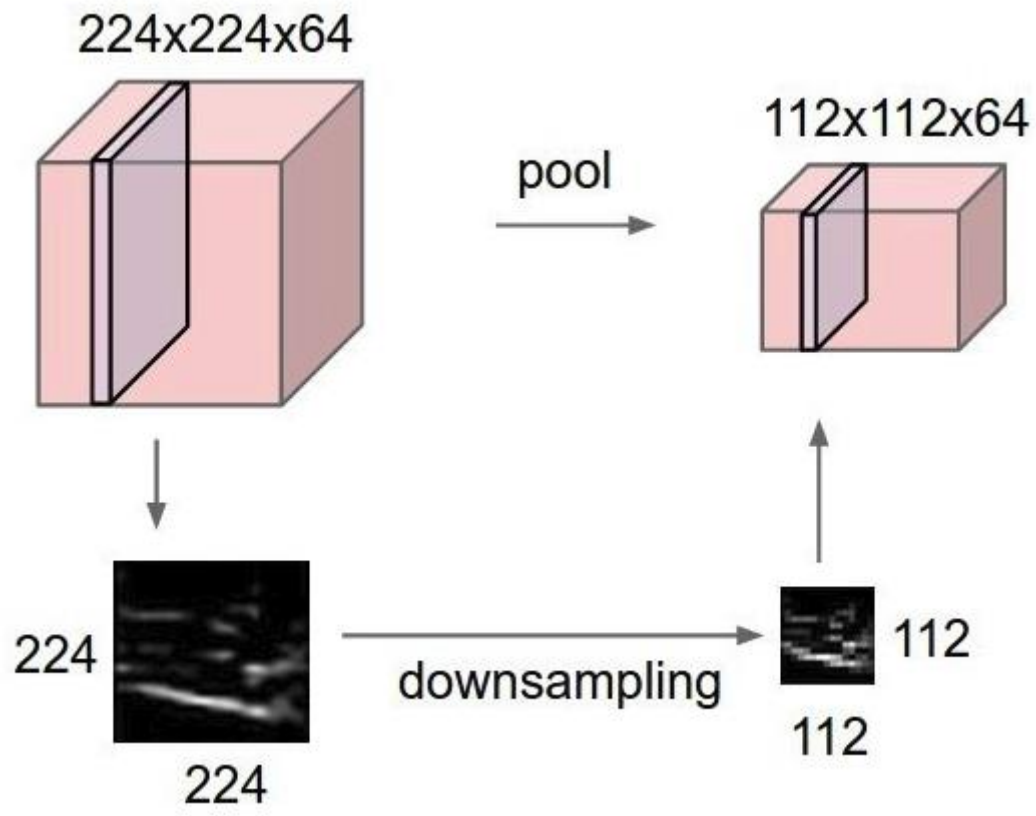


Ilustración 9: Ejemplo del uso de Max Pooling

Esta técnica es usada en gran parte para ayudar a reducir el gran coste producido por el gran número de parámetros usados en el proceso mediante la aportación de una forma abstracta de la representación. Para aplicar Max Pooling es necesario hacer uso de un filtro “Max” en subregiones no superpuestas de la representación inicial.

Batch normalization

Las redes neuronales requieren una cuidadosa puesta a punto en cuanto al aprendizaje de parámetros y la inicialización de pesos. Esta técnica hace más fácil estas mismas.

En lo que al problema de los pesos concierne, da igual cuál sea la inicialización de los pesos puesto que siempre estarán lejos de ser los finalmente aprendidos. Además, las redes neuronales profundas sufren grandes cambios en las capas posteriores cuando en una previa capa anterior ha habido una perturbación por muy pequeña que sea.

Durante la "back-propagation", este fenómeno causa distracciones en los gradientes, significando que estos mismos tienen que compensar los valores atípicos antes de aprender pesos para producir sus propias salidas.

El "batch normalization" normaliza los gradientes de estas distracciones y fluye hacia un fin común dentro de un rango de mini batch.

La tasa de aprendizaje supone otro gran problema puesto que cuando la tasa es pequeña, muy pocos gradientes corrigen los pesos puesto que los gradientes para las activaciones de las partes aisladas no deberían afectar a las ya aprendidas. Mediante el "batch normalization" estas activaciones de las partes aisladas son reducidas y se puede hacer uso de una tasa de aprendizaje más elevada. De esta manera el proceso de aprendizaje será mucho más rápido.

En los últimos años, se ha demostrado que la detección de objetos mediante arquitecturas no supervisadas supera ampliamente a todo lo anteriormente desarrollado y estudiado. Redes creadas por diferentes investigaciones tales como ImageNet 2010, ImageNet 2012 o AlexNet por ejemplo. Google realizó experimentos con la red ImageNet 2012 y determinó que obtenía resultados que mejoraban en un 60% a los experimentos que había hecho anteriormente. En 2014 Google generó la red GoogleNet con aún más capas internas y más sencillas de entrenar, logrando una tasa de errores del 6,7%. De esta manera se aproximó a la tasa de error humana, quedándose a un 1% de esta.

Aun así, cada vez son más los investigadores que trabajan con redes neuronales similares a las iniciales, y con algoritmos de entrenamiento supervisados mediante funciones de activación nuevas y reguladores específicos, lo que permite el entrenamiento supervisado de forma eficiente usando el Backpropagation. Rectified Linear Units (ReLUs) y el mecanismo dropout, mostrando que la fase de pre-entrenamiento ya no es necesaria.

Procedimientos, selección de alternativas e implementación

En este apartado se describirá el procedimiento y el análisis de las diferentes alternativas posibles para el desarrollo del proyecto. En la fase práctica del proyecto, se tratará de conseguir una anotación con unos resultados aceptables y, tras el análisis de las diferentes redes neuronales ya creadas, se hará una comparación de estas mismas con la red conseguida por el alumno.

Planteamiento del problema

Antes de continuar, se precisará detalladamente el problema a resolver en el proyecto mediante el diseño de un software.

Tal y como hemos mencionado anteriormente, el objetivo principal del proyecto es el desarrollo de un software capaz de identificar y anotar eficazmente diferentes objetos. Este software está pensado para la futura implementación en los diferentes robots y máquinas en la cuarta revolución industrial.

Antes de la creación del código y la más próxima implementación del mismo, se ha desarrollado un análisis de las diferentes alternativas con las que se podría crear el software. Posteriormente, se decidirá el método a utilizar y se empezará con la creación del software. Además, será necesario reunir cientos de imágenes de diferentes clases en un dataset para el consiguiente entrenamiento.

Después del entrenamiento, en el que se procesarán las imágenes del dataset mediante algoritmos y se creará un modelo que, junto con el código y los archivos de test, imágenes o vídeos, formará la parte práctica del proyecto.

Procedimientos a seguir

Para llevar a cabo el proyecto, habrá que seguir unas fases. Generalmente, estas fases son: creación del dataset, elección de los algoritmos, entrenamiento y prueba de los resultados.

Creación y preparación del dataset

En esta fase del proyecto se deberán reunir cientos de imágenes de diferentes clases para posteriormente ser procesadas. El objetivo principal del proyecto será detectar estas clases después del procesamiento por lo que deberemos elegir cuidadosamente estas mismas. Aunque podemos encontrar muchos ejemplos de datasets en internet, se ha preferido preparar uno propio mediante un código con el que fácilmente descargar imágenes desde la API de Búsqueda de imágenes Bing de Microsoft. Para ello, basta con crear una cuenta en la API y recibir una de las llaves temporales con las que acceder a la aplicación.

Es muy importante contrastar que las imágenes del dataset son representativas de la clase y que claramente podemos identificar a qué clase pertenecen. Una vez retiradas las imágenes que se consideran no válidas, se deberá continuar con la preparación del dataset.

Cuando hayamos supervisado todas las imágenes, deberemos clasificarlas por clases cambiándoles el nombre y poniéndoles uno que haga referencia a esta misma.

Entrenamiento

En el momento en el que el dataset que se vaya a utilizar esté listo, se dará comienzo a preparar la fase de entrenamiento. En esta fase se procesarán las imágenes del dataset y lograremos un modelo con el que posteriormente anotar y clasificar diferentes objetos. Para el procesamiento de imágenes deberemos hacer uso de uno o varios algoritmos de clasificación por lo que se procederá a hacer un análisis de los diferentes algoritmos y a una posterior elección del más adecuado para nuestro caso.

Análisis de alternativas

Antes de continuar con el proyecto y el desarrollo del software, se ha hecho un análisis de las diferentes alternativas existentes para la creación del mismo. Para llevar a cabo dicho análisis se hicieron unas simulaciones con algunas de las clases que posteriormente serían usadas en la identificación y anotación. En este caso, y como la utilización de este software se daría principalmente en la industria, y más concretamente en el ámbito del transporte, se trata de entrenar clases como “coche”, “tren” …

Tal y como se ha explicado en el estado del arte, existen muchas posibilidades para la anotación de objetos mediante métodos discriminativos. A continuación, se analizarán algunos de esos métodos.

Antes de comenzar con el análisis se ha preparado una pequeña simulación de nuestro dataset en un archivo con ciertos atributos de las clases a analizar. Este es un pequeño ejemplo de las primeras líneas del archivo:

	class_label	class_name	class_subtype	...	width	height	color_score
0	1	boat	sail	...	8.4	7.3	0.55
1	1	boat	sail	...	8.0	6.8	0.59
2	1	boat	sail	...	7.4	7.2	0.60
3	2	car	red	...	6.2	4.7	0.80
4	2	car	red	...	6.0	4.6	0.79

Ilustración 10: Ejemplo del archivo "casa.txt" en el cual se detallan los atributos de las clases "boat" y "car"

En total, contaremos con 4 clases diferentes en este ejemplo: "boat", "car", "train" y por último "lorry". Cada una de ellas contiene 5 atributos "subtype", "mass", "width", "height" y "color_score" que serán de gran ayuda en la clasificación.

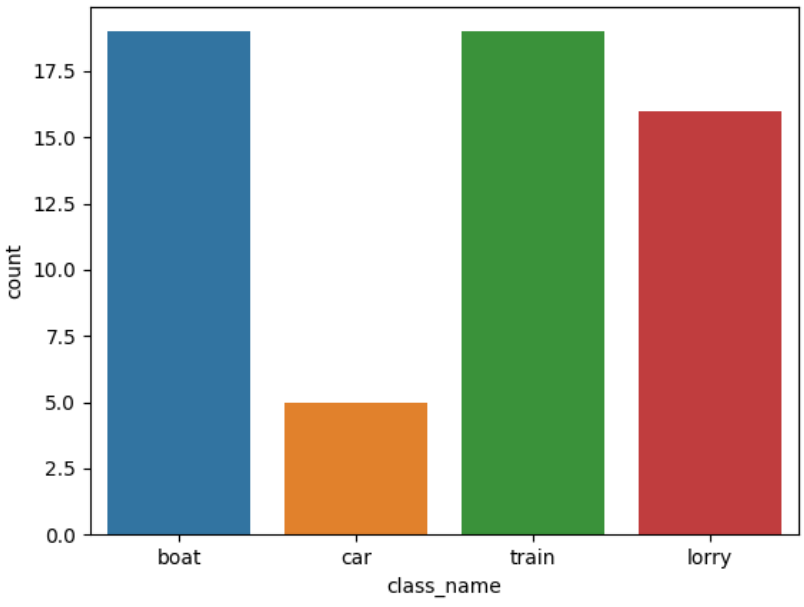


Ilustración 11: Número de ejemplos de cada clase que se van a utilizar

En la ilustración 11 podemos observar el número de ejemplos de cada clase que vamos a usar.

Contaremos con un número de ejemplos similar de todas las clases excepto de la clase "car" en la que tan solo tendremos 5 ejemplos.

Además, comprobaremos estadísticamente el dataset logrando los siguientes resultados:

	class_label	mass	width	height	color_score
count	59.000000	59.000000	59.000000	59.000000	59.000000
mean	2.542373	163.118644	7.105085	7.693220	0.762881
std	1.208048	55.018832	0.816938	1.361017	0.076857
min	1.000000	76.000000	5.800000	4.000000	0.550000
25%	1.000000	140.000000	6.600000	7.200000	0.720000
50%	3.000000	158.000000	7.200000	7.600000	0.750000
75%	4.000000	177.000000	7.500000	8.200000	0.810000
max	4.000000	362.000000	9.600000	10.500000	0.930000

Ilustración 12: Datos estadísticos del dataset preparado

Como podemos comprobar en la ilustración 12, los valores numéricos no siguen la misma escala, por lo que deberemos aplicar escalada geométrica al dataset que formamos para el entrenamiento.

Para una mejor visualización de la distribución de las variables de entrada, crearemos un diagrama de caja por cada atributo.

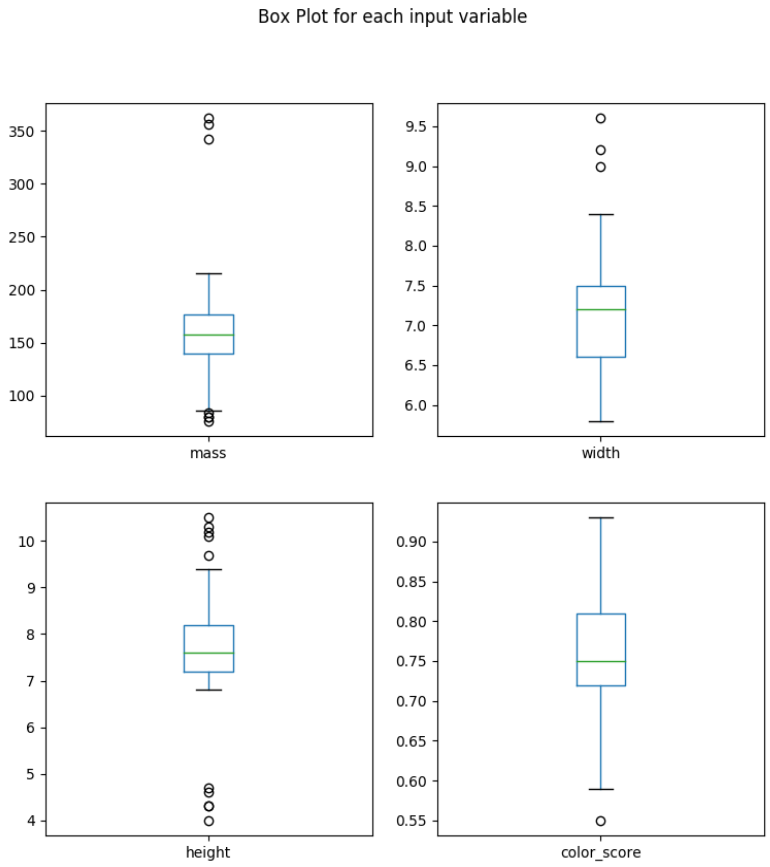


Ilustración 13: Diagrama de caja por cada atributo

Tal y como podemos comprobar en la ilustración 13, el atributo "color_score" tiene una distribución semejante a la gaussiana. Este dato puede ser muy valioso en el posterior entrenamiento y a la hora de determinar el algoritmo que se va a usar.

Para continuar con el análisis, crearemos un histograma de cada atributo para así seguir recopilando datos que pueden ser de gran importancia.

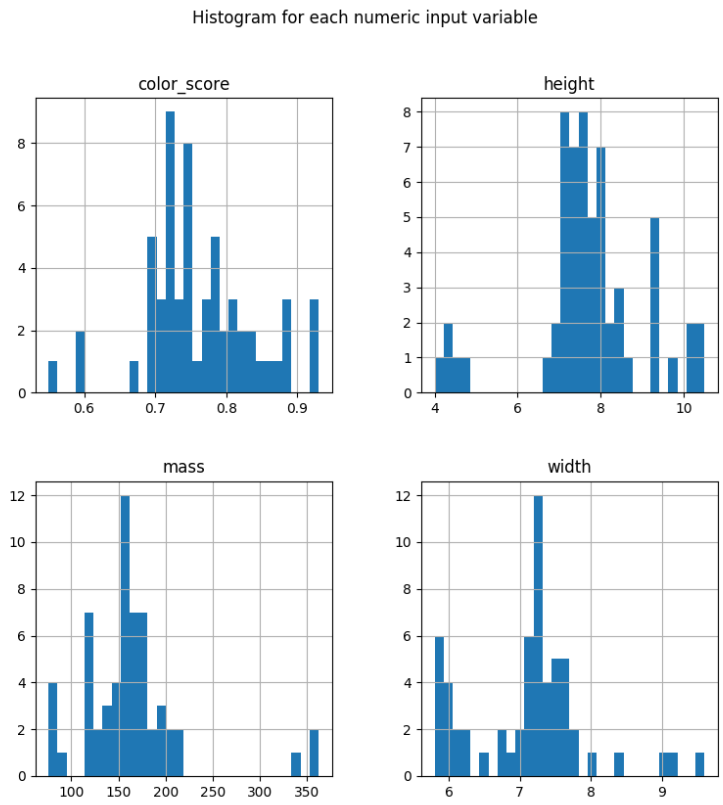


Ilustración 14: Histogramas de los atributos del dataset

En la ilustración 14 podemos observar que los atributos “mass” y “width” están correlacionados, lo que sugiere una gran relación predictiva entre ellos.

Antes de empezar con el uso y posterior comparación de los métodos, comenzaremos creando los sets de entrenamiento y escalándolos.

Para ello utilizaremos el siguiente código:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Una vez tengamos todo preparado, comenzaremos a analizar los diferentes métodos propuestos. Para ello, haremos usos de los diferentes algoritmos que nos proporciona el paquete "Scikit-learn" de python.

Regresión logística

La regresión logística [W.4] es un método usado en la estadística para predecir el resultado de una variable que puede adoptar un número limitado de categorías en función de las variables independientes o predictoras. Esta variante corresponde al caso en que se valora la contribución de diferentes factores en la ocurrencia de un evento simple.

Para comprobar la eficacia de este método en nuestro dataset se hará uso del siguiente código:

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
print('Accuracy of Logistic regression classifier on training set: {:.2f}'
     .format(logreg.score(X_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'
     .format(logreg.score(X_test, y_test)))
```

En el código tan solo aplicamos el algoritmo a nuestros datos de entrenamiento "x_train" y "y_train". A continuación, imprimimos los resultados obtenidos:

```
Accuracy of Logistic regression classifier on training set: 0.70
Accuracy of Logistic regression classifier on test set: 0.40
```

Por lo que podemos observar, los resultados no son muy satisfactorios.

Árbol de decisiones

El método del árbol de decisión [W.5] es un modelo de predicción utilizado en múltiples ámbitos que van desde la economía hasta la inteligencia artificial.

El uso de árboles de decisiones en el campo del machine learning es muy frecuente y muchas veces resulta un método muy efectivo. Este método, tal y como su nombre indica, hace uso de un árbol para la toma de decisiones. Normalmente se comienza en un simple nodo desde el que salen 2 ramas con una decisión en cada una. Al tomar una decisión se llegará a otro nodo del que volverán a salir un número de ramas y en el que se volverá a tomar una decisión. El algoritmo tratará de predecir la mejor decisión hasta llegar a su objetivo.

Un ejemplo muy simple es el siguiente:

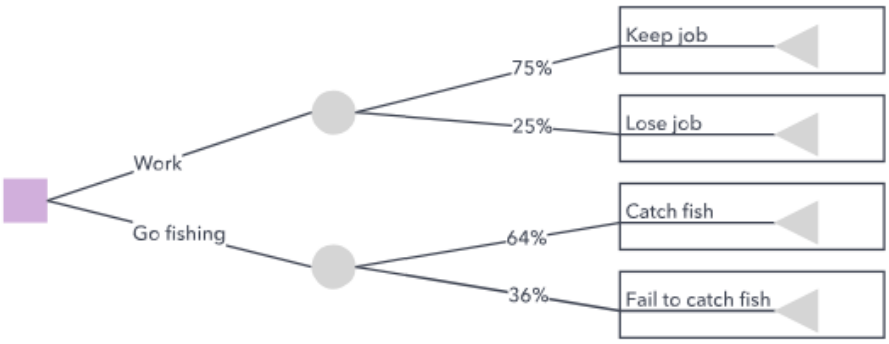


Ilustración 15: Ejemplo de un árbol de decisiones

En el ejemplo de la ilustración 15, el individuo deberá tomar una primera decisión y llegará a un segundo nodo dependiendo de esta misma. En este caso, tras tomar la decisión tendrá un porcentaje diferente de cada opción posible dependiendo del objetivo de su decisión.

Una vez entendido el proceso, aplicaremos el algoritmo:

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

Obteniendo los siguientes resultados:

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.87
```

En este caso, los resultados obtenidos son notablemente mejores que los anteriores.

K vecinos más próximos

El algoritmo "K-Nearest Neighbors" o KNN [W.6], es un método de clasificación supervisada, no paramétrico que estima la función de densidad de probabilidad de que un elemento *x* pertenezca a la clase *Y* a partir de una información proporcionada por un conjunto de prototipos. El entrenamiento se basa en los ejemplos cercanos en el espacio de los elementos.

Aplicamos el algoritmo:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
```

Con la obtención de los siguientes resultados:

```
Accuracy of K-NN classifier on training set: 0.95
Accuracy of K-NN classifier on test set: 1.00
```

Tal y como podemos comprobar, los resultados obtenidos son magníficos.

Análisis discriminante lineal

En un conjunto de objetos clasificado por grupos o clases, el análisis discriminante [W.7] lineal es equivalente a un análisis de regresión en el cual la variable dependiente es categórica y tiene como etiquetas las variables de cada uno de los grupos. Las variables independientes son continuas y determinarán a qué clase pertenecen los objetos a detectar. El objetivo es encontrar relaciones lineales entre las variables continuas que mejor discriminen en las clases dadas a los objetos. Se tratará de definir un patrón de decisión que asigne un objeto nuevo a una de las clases previamente fijadas.

Aplicamos el algoritmo:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(X_test, y_test)))
```

Con la obtención de los siguientes resultados:

```
Accuracy of LDA classifier on training set: 0.86
Accuracy of LDA classifier on test set: 0.67
```

En este caso, los resultados son aceptables, aunque con el anterior método son bastante mejores.

Clasificador bayesiano ingenuo

El clasificador bayesiano ingenuo es un clasificador probabilístico fundamentado en el teorema de Bayes y algunas hipótesis simplificadoras adicionales, con el supuesto de que existe independencia entre los predictores.

Aplicando el algoritmo:

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'
     .format(gnb.score(X_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'
     .format(gnb.score(X_test, y_test)))
```

Obtendremos los siguientes resultados:

```
Accuracy of GNB classifier on training set: 0.86
Accuracy of GNB classifier on test set: 0.67
```

En este caso, obtendremos unos resultados similares a los del análisis discriminante lineal.

Support Vector Machine

Support Vector Machine [W.8] o Máquina de Vector Soporte son un conjunto de algoritmos de aprendizaje supervisado. Este sistema se basa en el uso de un espacio de hipótesis de funciones lineales en un espacio de mayor dimensión inducido por un Kernel. En este, las hipótesis son entrenadas por un algoritmo tomado de la teoría de optimización el cual utiliza elementos de la teoría de generalización.

Aplicamos el algoritmo mediante el siguiente código:

```
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(X_train, y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
     .format(svm.score(X_test, y_test)))
```

En este caso, los resultados obtenidos son los siguientes:

```
Accuracy of SVM classifier on training set: 0.61
Accuracy of SVM classifier on test set: 0.33
```

Los resultados resultan muy pobres para la simulación de nuestro dataset.

Tal y como hemos podido observar, el algoritmo KNN ha aportado los mejores resultados y por eso mismo nos vamos a centrar en él. Aun así, crearemos una matriz de confusión [W.9] para asegurarnos de que no hemos cometido ningún error. Aunque el set de testeo es muy pequeño y será difícil representar errores mediante esta matriz, es conveniente asegurarse siempre.

```
[[4 0 0 0]
 [0 1 0 0]
 [0 0 8 0]
 [0 0 0 2]]
             precision    recall  f1-score   support

          1       1.00      1.00      1.00         4
          2       1.00      1.00      1.00         1
          3       1.00      1.00      1.00         8
          4       1.00      1.00      1.00         2

avg / total       1.00      1.00      1.00        15
```

La precisión es la proporción del total de predicciones positivas que han sido correctas y se calcula de la siguiente manera:

$$P = \frac{d}{b+d}$$

El recall, la sensibilidad o la exhaustividad, indican la proporción de casos positivos que han sido correctamente identificados. Se calculan de la siguiente manera:

$$TP = \frac{d}{c+d}$$

Finalmente, el f1-score o valor-F es la medida de precisión que tiene un test. La fórmula para el cálculo es la siguiente:

$$F_1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Exhaustividad}}{\text{Precisión} + \text{Exhaustividad}}.$$

El test, por lo tanto, ha sido lo más preciso posible, y nuestros resultados son correctos.

Como KNN ha sido el mejor de todos los algoritmos, dibujaremos un gráfico con el límite de decisión o “decision boundary” de este clasificador.

Para ello, deberemos comprobar que valor de K es el más preciso.

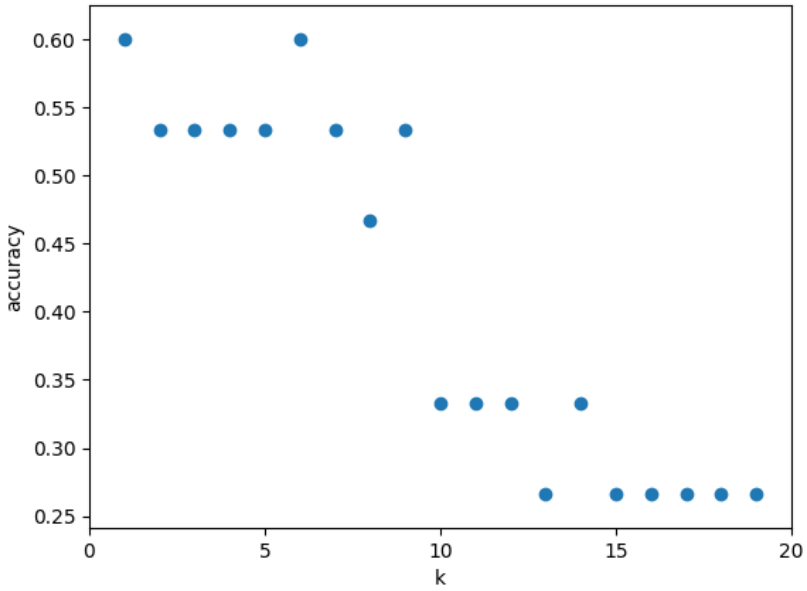


Ilustración 16: Gráfico de la precisión en base a los diferentes valores de K

Como podemos observar en la ilustración 16, el valor de K = 6 es el más preciso por lo que procederemos a dibujar gráficamente su límite de decisión.

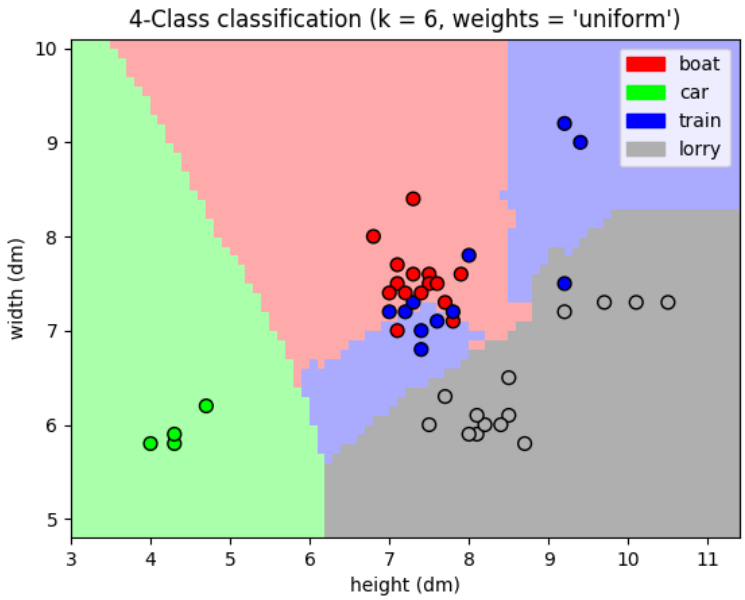


Ilustración 17: Limite de decisión de K = 6

En la ilustración 17 podemos ver como prácticamente la totalidad de los pesos están ubicados en el espacio de su propia clase. El resultado, por lo tanto, es muy bueno para este dataset.

Tras analizar los resultados obtenidos, y una vez comprobado que el algoritmo más efectivo en este caso es KNN, en el caso de centrarnos en este tipo de métodos se debería hacer uso de este algoritmo.

Aunque los resultados del test son muy altos, se ha decidido basar la parte práctica del entrenamiento del proyecto en las redes neuronales, puesto que se trata de un campo innovador y que ha demostrado tener unos resultados superiores a los conseguidos anteriormente.

Solución elegida para el entrenamiento

Una vez elegida la forma en la que practicaremos el entrenamiento, en este caso mediante redes neuronales, el siguiente objetivo será conseguir el modelo que se usará para el reconocimiento de objetos y la anotación.

Existen múltiples alternativas y herramientas para la clasificación de imágenes, pero, en este caso, se hará uso de una herramienta llamada CAFFE y desarrollada por UC Berkeley. CAFFE es un marco de aprendizaje profundo, de código abierto, escrito en C++ y con una interfaz de Python. Mediante un proceso con las diferentes aplicaciones que nos ofrece CAFFE, conseguiremos un modelo con extensión ".caffemodel". Este modelo post-entrenamiento será el utilizado para la identificación y anotación de los objetos de los archivos de test.

Cabe mencionar que, además de CAFFE, se usarán las librerias de OPENCV. Estas librerías son ampliamente utilizadas en el mundo de la inteligencia artificial y proporcionan software muy potente para la anotación y clasificación de objetos.

Preparación de archivos y diseño de la red neuronal

A continuación se procederá a la manipulación de código proporcionado por CAFFE para la creación de ciertos archivos para la creación de la red neuronal y el posterior entrenamiento.

Se deben crear dos archivos llamados "train.txt" y "test.txt". En el primero, se incluirán los nombres de las imágenes con los que se procederá el entrenamiento y en el segundo los nombres de las imágenes con las que probaremos más adelante nuestro modelo. Para ello, copiaremos el 70% de los nombres de las imágenes en el archivo "train.txt" y el restante 30% de las mismas en el archivo "test.txt".

Una vez tengamos los archivos creados, deberemos crear un archivo que contenga la imagen normalizada. El pre-procesamiento de los datos es muy importante para que el entrenamiento sea eficaz. Por eso mismo, la normalización de nuestros datos será de gran ayuda para el procesamiento de estos. El objetivo de este proceso será conseguir un nuevo set de datos en el cual el valor medio sea 0. Para ello, seguiremos el siguiente proceso matemático:

$$x_\mu = \frac{x^{(1)} + x^{(2)} + \cdots + x^{(m)}}{m} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

En esta fórmula, x_μ es el valor medio mientras que $x^{(i)}$ son los valores de cada dato de entrenamiento.

Por lo tanto, para conseguir el objetivo, nuestro nuevo set de datos deberá ser el siguiente:

$$x_{new}^{(i)} = x^{(i)} - x_{\mu}.$$

Y, por tanto, el valor medio de nuestro nuevo set de datos será cero:

$$\sum_{i=1}^{m} x_{new}^{(i)} = \sum_{i=1}^{m} x^{(i)} - m x_{\mu} = \sum_{i=1}^{m} x^{(i)} - m \frac{1}{m} \sum_{i=1}^{m} x^{(i)} = 0$$

A partir de esta breve explicación teórica, lograremos el valor medio del nuevo set de datos. Para eso, necesitaremos convertir nuestras imágenes a un formato LMDB. Este archivo es recomendable cuando estemos haciendo uso de un número elevado de imágenes en el dataset. Estos archivos son creados a partir de la compresión de las imágenes y proporcionan una gran ayuda en el procesamiento. Hoy en día, el rendimiento no solo se basa en la precisión de los resultados, sino que tiene en cuenta el tiempo de procesado y la precisión del mismo.

Una vez tengamos los archivos LMDB, continuaremos con la creación de la imagen normalizada de nuestro dataset. Para ello, CAFFE nos ofrece diferentes archivos que, junto con el lenguaje de programación Python, automatizarán la creación del mismo.

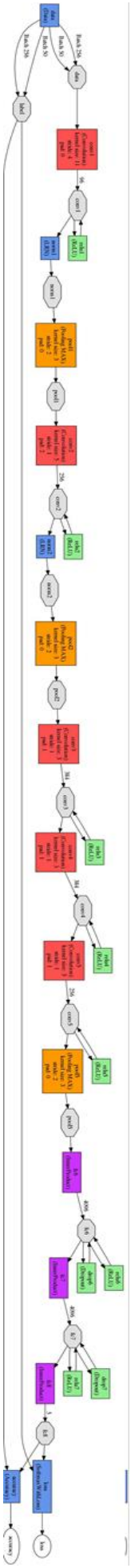
Una vez lograda la imagen normalizada de nuestro dataset, es hora de comenzar con el diseño de la red neuronal. Antes de nada, se han analizado algunas de las redes con mejores resultados hasta el momento.

AlexNet

Esta red neuronal fue creada en 2012 por el grupo SuperVision, en el cual participaron Alex Krizhevsky, Geoffrey Hinton y Iliya Stskever. Este mismo año, consiguió un error mínimo del 15,3%, convirtiéndose así en la red más precisa del momento por un 10,8% de diferencia. Los resultados de esta red fueron muy precisos para la época, pero el tiempo de procesamiento fue muy largo, ya que tardaron 6 días y tuvieron que utilizar dos GPUs muy potentes para procesar todos los datos.

Esta red está compuesta por 8 capas: 5 capas de convolución y 3 capas totalmente conectadas.

En la siguiente página podemos observar la red gráficamente.

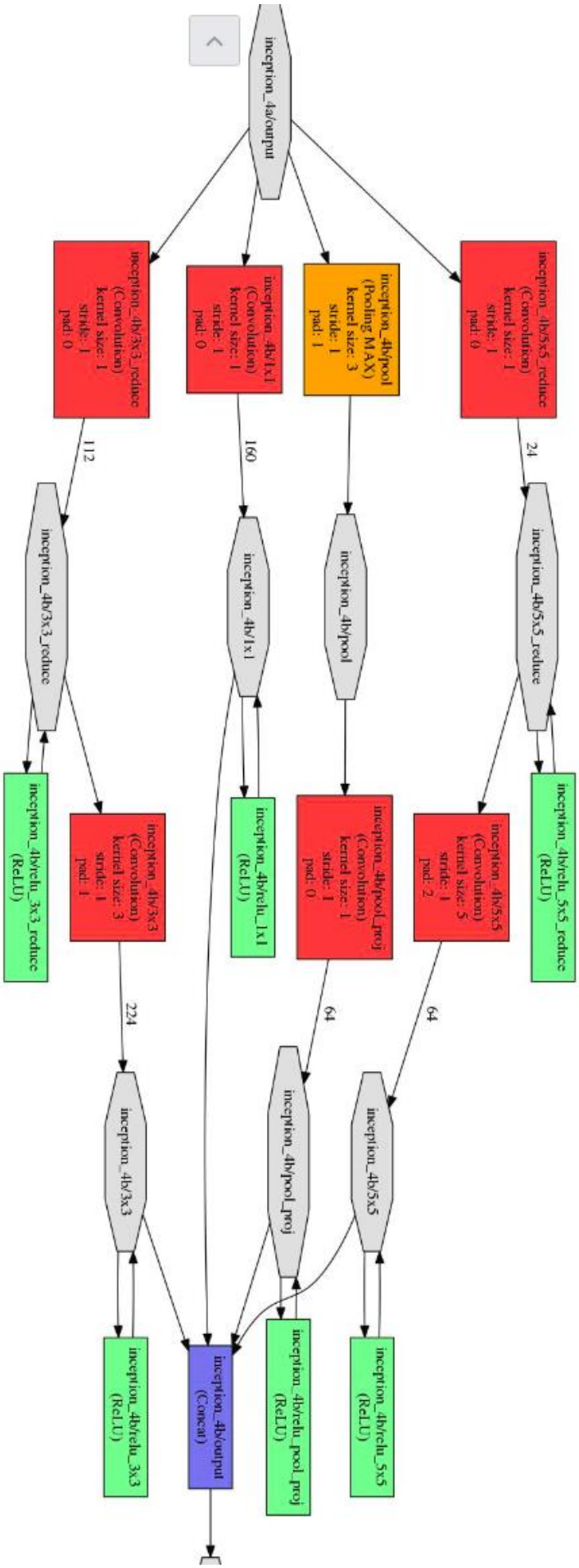


BVLC GOOGLENET

En el año 2014 Google ganó la competición ILSVRC 2014 con esta red. Alcanzo un mínimo de error de tan solo el 6,67% consiguiendo un porcentaje de error muy cercano al del ojo humano. Esta red, inspirada en anteriores, es mucho más compleja que AlexNet y hace uso de "batch-normalization", distorsiones de imagen y RMSprop. Este último módulo está basado en muchas y pequeñas convoluciones para así reducir drásticamente el número de parámetros. Consiste en una red con 22 capas, pero reduce el número de parámetros de 60 millones, utilizados en AlexNet, a tan solo 4 millones.

La red es tan grande que sería imposible representarla en el proyecto, por lo que se representará tan solo una de las capas.

En la siguiente imagen se podrán observar las diferentes convoluciones de una misma capa.


inception_4a/output
inception_4b/5x5_reduce (Convolution) kernel size: 1 stride: 1 pad: 0
inception_4b/pool (Pooling MAX) kernel size: 3 stride: 1 pad: 1
inception_4b/1x1 (Convolution) kernel size: 1 stride: 1 pad: 0
inception_4b/3x3_reduce (Convolution) kernel size: 1 stride: 1 pad: 0
24
160
112
inception_4b/5x5_reduce
inception_4b/pool
inception_4b/1x1
inception_4b/3x3_reduce
inception_4b/relu_5x5_reduce (ReLU)
inception_4b/pool_proj (Convolution) kernel size: 1 stride: 1 pad: 0
inception_4b/1x1 (ReLU)
inception_4b/5x5 (Convolution) kernel size: 5 stride: 1 pad: 2
inception_4b/3x3 (Convolution) kernel size: 3 stride: 1 pad: 1
inception_4b/relu_3x3_reduce (ReLU)
64
64
224
inception_4b/pool_proj
inception_4b/5x5
inception_4b/3x3
inception_4b/relu_5x5 (ReLU)
inception_4b/relu_pool_proj (ReLU)
inception_4b/output (Concat)
inception_4b/relu_3x3 (ReLU)

Tras comprobar la dificultad y la capacidad de procesamiento necesaria, el diseño de la red neuronal creada en el proyecto será mucho más simple y servirá para la clasificación de objetos. En cuanto a la anotación de objetos, usaremos modelos ya creados como el MobileNetSSD.

Para la creación de la red neuronal, deberemos crear dos archivos. El archivo "solver.prototxt" que contendrá los parámetros de procesamiento y el archivo "train_val.prototxt". El contenido de este archivo serán las diferentes definiciones de las capas de la red neuronal. Una vez procesados los archivos mediante la herramienta CAFFE obtendremos el archivo con extensión ".caffemodel", el cual será nuestro modelo de prueba y ya podremos testearlo mediante el código creado para ello.

Resultados

A continuación, se mostrarán los resultados más significativos de la fase práctica del proyecto.

Para esta fase, se ha decidido hacer uso de la red AlexNet para el entrenamiento de nuestro dataset ya que la red principalmente diseñada no ha dado los resultados esperados y en cambio, el entrenamiento con la red AlexNet mostró unos resultados aceptables.

Una vez obtenidos los archivos requeridos y anteriormente explicados, continuamos con el entrenamiento. El entrenamiento de nuestros datos con la red AlexNet tuvo una duración aproximada de dos días en el CPU y ejecutó 1000 iteraciones que dieron como resultado un modelo llamado "caffe_alexnet_train_iter_1000.caffemodel". En la siguiente imagen podemos observar un fragmento del procesamiento del CPU en el entrenamiento.

```
I0613 05:07:14.493777 17641 solver.cpp:351] Iteration 200, Testing net (#0)
I0613 05:07:45.456166 17643 data_layer.cpp:73] Restarting data prefetching from start.
I0613 05:10:21.811478 17643 data_layer.cpp:73] Restarting data prefetching from start.
I0613 05:12:58.158501 17643 data_layer.cpp:73] Restarting data prefetching from start.
I0613 05:15:34.591881 17643 data_layer.cpp:73] Restarting data prefetching from start.
I0613 05:18:03.685086 17643 data_layer.cpp:73] Restarting data prefetching from start.
I0613 05:19:40.445906 17641 solver.cpp:418]     Test net output #0: accuracy = 0.5458
I0613 05:19:40.446075 17641 solver.cpp:418]     Test net output #1: loss = 1.80022 (* 1 = 1.80022 loss)
I0613 05:21:13.967604 17641 solver.cpp:239] Iteration 200 (0.00760555 iter/s, 2629.66s/20 iters), loss = 1.26782
I0613 05:21:13.967684 17641 solver.cpp:258]     Train net output #0: loss = 1.26782 (* 1 = 1.26782 loss)
I0613 05:21:13.967694 17641 sgd_solver.cpp:112] Iteration 200, lr = 1e-07
I0613 05:25:57.807248 17642 data_layer.cpp:73] Restarting data prefetching from start.
I0613 05:40:09.672986 17642 data_layer.cpp:73] Restarting data prefetching from start.
I0613 05:51:13.497699 17641 solver.cpp:468] Snapshotting to binary proto file Escritorio/caffemodel/models/kaggle/
caffe_alexnet_train_iter_220.caffemodel
I0613 05:51:14.837226 17641 sgd_solver.cpp:280] Snapshotting solver state to binary proto file Escritorio/caffemodel/models/kaggle/
caffe_alexnet_train_iter_220.solverstate
I0613 05:52:50.799309 17641 solver.cpp:239] Iteration 220 (0.0105439 iter/s, 1896.83s/20 iters), loss = 1.26995
I0613 05:52:50.799484 17641 solver.cpp:258]     Train net output #0: loss = 1.26995 (* 1 = 1.26995 loss)
I0613 05:52:50.799495 17641 sgd_solver.cpp:112] Iteration 220, lr = 1e-07
I0613 05:55:59.287492 17642 data_layer.cpp:73] Restarting data prefetching from start.
I0613 06:10:04.597625 17642 data_layer.cpp:73] Restarting data prefetching from start.
I0613 06:22:36.124083 17641 solver.cpp:468] Snapshotting to binary proto file Escritorio/caffemodel/models/kaggle/
caffe alexnet train iter 240.caffemodel
```

Ilustración 18: Resultados obtenidos durante el entrenamiento

Como podemos observar en la ilustración 18, en la iteración número 200 el modelo obtuvo una precisión del 54,58%.

Una vez terminado el entrenamiento y habiendo obtenido el modelo definitivo, probamos mediante código escrito en python el modelo y para una de las clases entrenadas denominada "silla" obtuvimos el siguiente resultado.

Ilustración 19: Anotación de la clase silla

Con un 54,82% de probabilidad de que la ilustración 19 sea una silla, el software diseñado obtiene unos resultados aceptables. La predicción del software ha sido la siguiente.

```
[INFO] Tiempo en clasificar 0.045557 segundos
[INFO] 1. Etiqueta: chair, probabilidad: 0.54819
[INFO] 2. Etiqueta: table, probabilidad: 0.20216
[INFO] 3. Etiqueta: dog, probabilidad: 0.14536
[INFO] 4. Etiqueta: computer, probabilidad: 0.069255
[INFO] 5. Etiqueta: face, probabilidad: 0.035032
```

Aun así, el objetivo final del proyecto es la aplicación de la anotación de objetos a la industria 4.0. Por eso mismo se ha optado por hacer uso de la red MobileNetSSD.

Esta red es capaz de anotar de forma simultánea varios objetos en una imagen en movimiento, como un vídeo o una reproducción en directo de una cámara como las implementadas en la siguiente revolución industrial.

Los resultados obtenidos en las pruebas con clases enfocadas a identificación de objetos participantes en la revolución como transportes o personas fueron excepcionales. El modelo fue puesto a prueba en un vídeo y con resultados similares a los mostrados en la ilustración 20.

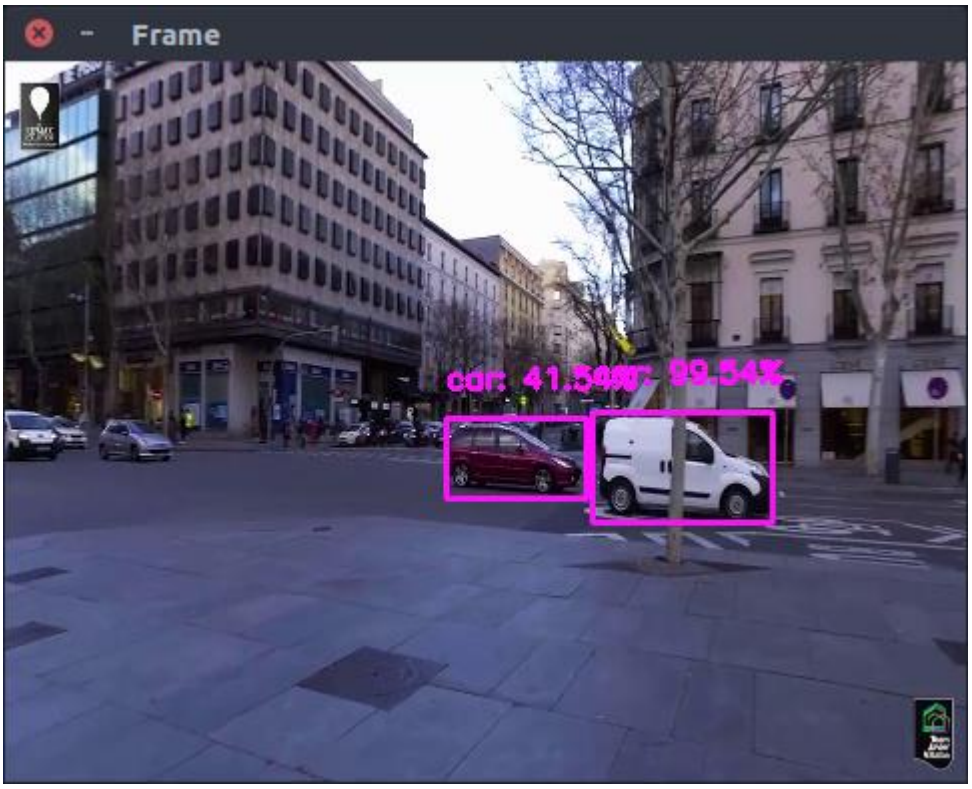


Ilustración 20: Anotación de coches en un vídeo

Y, como anteriormente hemos mencionado, este modelo es capaz de anotar diferentes clases en la misma imagen.

Ilustración 21: Anotaciones múltiples en la misma imagen

En la ilustración 21, podemos observar como aun obteniendo unos resultados magníficos, el modelo comete errores como la anotación de una silla en este caso.

Además, nuestro software anota en la terminal cada clase detectada mostrando las anotaciones de la ilustración 22.

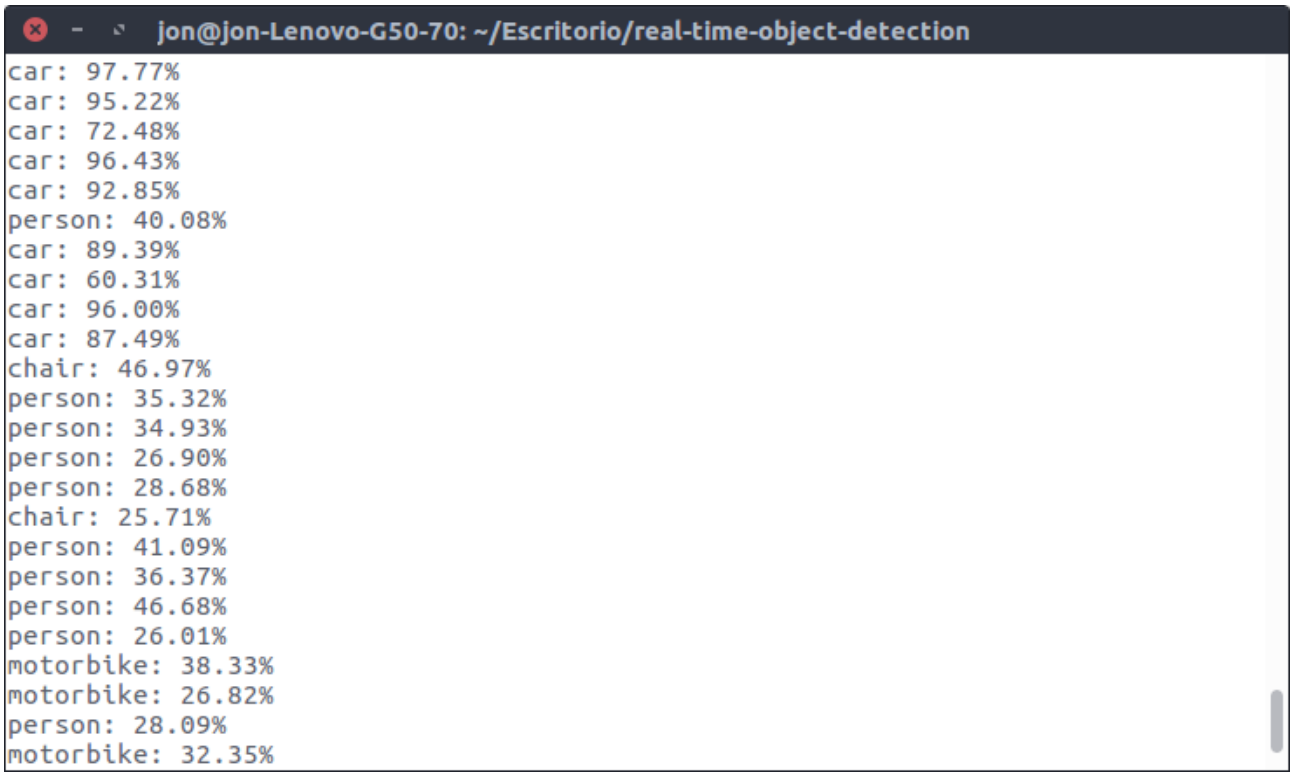
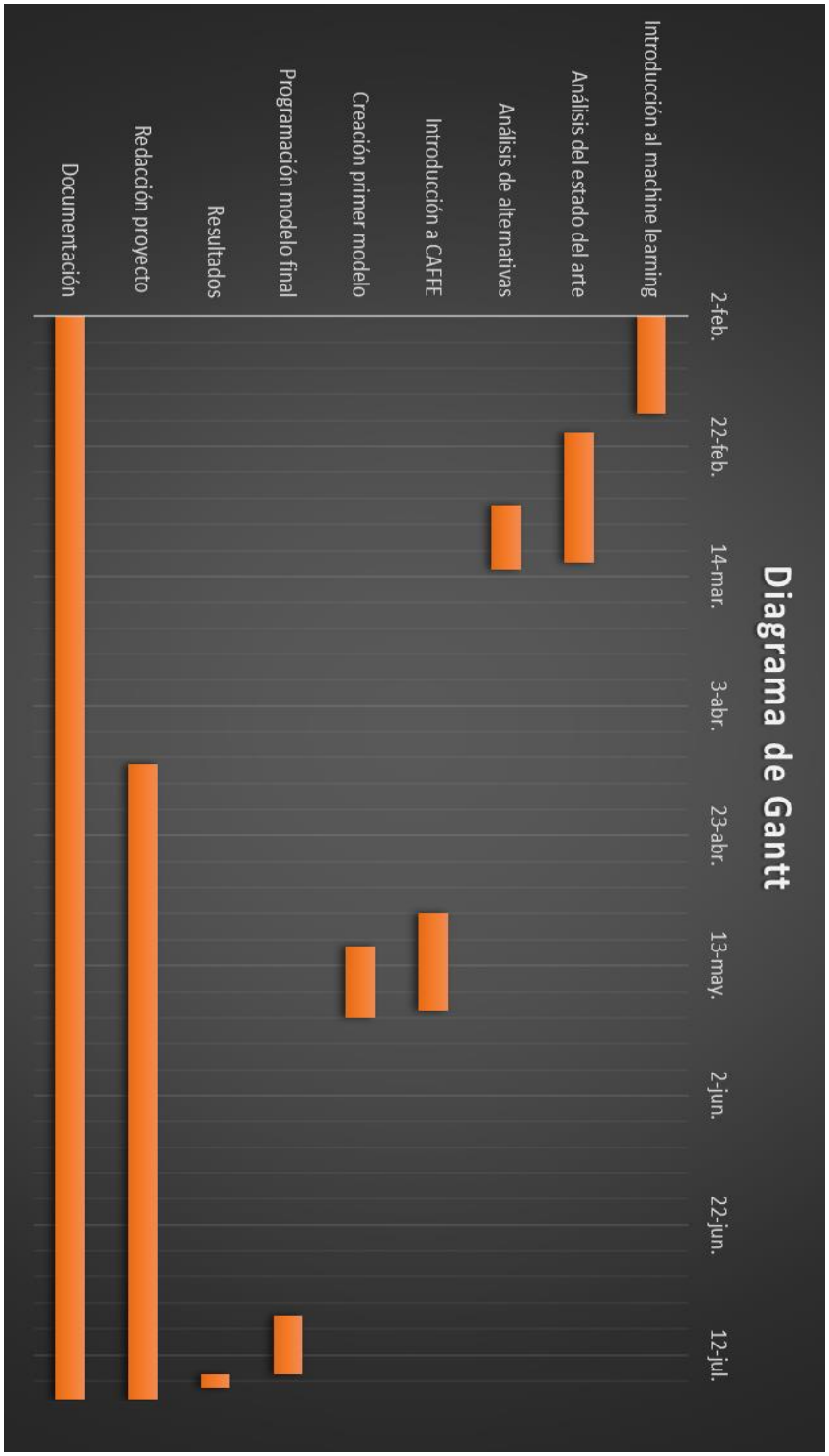


Ilustración 22: Anotaciones de los objetos reconocidos en tiempo real

Diagrama de Gantt



Aspectos económicos

En este apartado, se procederá a incluir el análisis de los costes y del presupuesto del proyecto. Se comenzará por la fase realizada hasta el momento por un ingeniero de telecomunicaciones. Se tendrán en cuenta los recursos humanos, la amortización del hardware usado en el proyecto y un porcentaje por los imprevistos que ha podido causar el proyecto.

Presupuesto de desarrollo

En esta sección se tendrán en cuenta los medios necesarios para el desarrollo de la solución hasta el momento. El mayor gasto será el sueldo del ingeniero y el equipo hardware utilizado.

La duración del proyecto ha sido de unos 6 meses, comenzando en febrero y acabando en julio. El cálculo del gasto en recursos humanos se puede contemplar en la siguiente tabla.

Recursos humanos			
Categoría	**Coste/hora**	**Tiempo**	**Coste Total**
Ingeniero de telecomunicación	50	300	15.000,00 €
Total			15.000,00 €

Tabla 1: Recursos humanos

Gastos por la seguridad social del trabajador

Concepto	**Porcentaje del sueldo**	**Total**
Seguridad social	31%	4.650,00 €

Tabla 2: Seguridad social

En el proyecto de ingeniería se ha hecho uso de varios equipos por lo que las amortizaciones se mostrarán en la siguiente tabla.

Activo	**Coste**	**Vida útil** (meses)	**Tiempo de uso en el proyecto** (meses)	**Amortización**
Portátil - Lenovo G50-80, i7-5500U, 16GB RAM, R5 M330 2GB	850,00€	48	7	123,96 €
PC - Asus I7-7700, 8GB RAM, 1HDD 1TB, GTX 1050 2GB y Monitor - Asus ROG Strix XG27VQ, Pantalla curva, 27"	1.566,00€	48	7	228,38 €
Total	2.416,00 €			352,34 €

Tabla 3: Amortización del hardware utilizado

Por lo tanto, el coste total del proyecto sería el siguiente:

Concepto	**Coste Total**
Recursos humanos	15.000,00 €
Amortizaciones	352,34 €
Seguridad Social	4.650,00 €
Subtotal	20.002,34 €
Imprevistos (7%)	1.400,17 €
Total	**21.402,51 €**

Tabla 4: Coste total del proyecto

El coste total de la realización del proyecto de ingeniería es de 21.402,51 €.

Conclusiones

Tras la finalización del proyecto, he llegado a varias conclusiones positivas sobre el desarrollo del mismo. Para empezar, el proyecto me ha permitido sumergirme en el mundo de la investigación, presentándome un problema que nunca antes había tratado y la problemática de encontrarle una solución mediante estudio e investigación. Además, el desarrollo del software y la programación del mismo en un nuevo lenguaje me han permitido adquirir ciertas aptitudes que antes no poseía.

Cabe destacar, que el estudio de la inteligencia artificial y, más concretamente, de la visión artificial, me ha hecho indagar en un interesante y prometedor ámbito que tiene un gran valor en la sociedad actual. Esta tendencia al alza tiene un gran mercado y cada vez existe un mayor interés en las empresas. Tal y como se ha tratado en algunos apartados del proyecto, el estudio de este tipo de materias de aprendizaje automático tendrá una gran relevancia en un futuro muy próximo y, por lo tanto, es muy beneficioso comprender, tanto teórica como prácticamente, la evolución que han tenido hasta el momento.

En el desarrollo práctico del software de este proyecto, he tenido que debatir las diferentes técnicas aplicables a la solución del problema y elegir una en base a los resultados obtenidos y a mi propio criterio. Una vez elegido el método para el desarrollo del software, la creación de mi propia red neuronal y la comprobación de los resultados de esta misma han ayudado a comprobar las capacidades de procesamiento requeridas para en entrenamiento de las redes neuronales posteriormente usadas en el proyecto.

Este proyecto, enfocado a la problemática de la industria 4.0, me ha hecho evaluar los pros y los contras de esta nueva revolución tanto técnica como éticamente. Aún queda un largo camino por recorrer hasta la total implementación de la misma en la sociedad, pero el enfoque de este proyecto es un gran ejemplo de las técnicas que se deberán utilizar para llevarla a cabo.

Referencias y bibliografía

Recursos de texto

[BENE13] Carl Benedikt Frey and Michael A. Osborne, September 2013, The future of employment: how susceptible are jobs to computerisation?

[BENE16] Carl Benedikt Frey, Michael A. Osborne, Craig Holmes, January 2016, Technology at work v2.0.

[DALA05] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Vol. 1, pp. 886-893). IEEE.

[DORK05] Dorkó, G. and Schmid, C. 2005. Object class recognition using discriminative local features. Technical Report RR-5497, INRIA - Rhône-Alpes.

[ENGE10] Karin Engel, Klaus D. Toennies, Hierarchical vibrations for part-based recognition of complex objects, Pattern Recognition, Volume 43, Issue 8, August 2010, Pages 2681-2691, ISSN 0031-3203

[FEIF07] Fei-Fei, L., Fergus, R., and Perona, P. 2007. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. Comput. Vis. Image Underst. 106, 1 (Apr. 2007), 59-70.

[FERG03] Fergus, R., Perona, P., and Zisserman, P. 2003. Object class recognition by unsupervised scale-invariant learning. In Proc. CVPR.

[GU10] Recognition using Regions Chunhui Gu, Joseph J. Lim, Pablo Arbelaez, Jitendra Malik. CVPR 2009, Miami, Florida.

[LEE10] Lee, Y. J; Grauman, K. “Object-graphs for context-aware category discovery”. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, p1-8.

[LIEN02] An extended set of Haar-like features for rapid object detection. Proceedings. 2002 International Conference on In Image Processing. 2002. Proceedings. 2002 International Conference on, Vol. 1 (2002), pp. I-900-I-903 vol.1.

[OMME10] Ommer, B. and Buhmann, J.M. “Learning the Compositional Nature of Visual Object Categories for Recognition”. Trans o Pattern Analysis and Machine Learning, vol 32, pp. 501-516, March. 2010.

[PAPA00] Papageorgiou, C., Poggio, T., 2000. A trainable system for object detection. Int. J. Comput. Vision 38 (1), 15-33.

[SERO16] Sergio Rodriguez Vaamonde, 2016, Detección y categorización de objetos invariante y multivista en imágenes digitales mediante visión artificial bioinspirada

[SERR05] Serre, T., L. Wolf and T. Poggio. Object Recognition with Features Inspired by Visual Cortex. In: Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Press, San Diego, June 2005.

[SERR07] Serre, Thomas; Wolf, Lior; Bileschi, Stanley; Riesenhuber, Maximilian; Poggio, Tomaso."Robust object recognition with cortex-like mechanisms". IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, v29, p411-426.

[SHOT06] Shotton, J., Winn, J., Rother, C., & Criminisi, A. (2006). TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In A. Leonardis, H. Bischof, & A. Pinz (Eds.), LNCS: Vol. 3951. Proceedings of European conference on computer vision (pp. 1-15). May 2006. New York: Springer.

[SHOT08] Jamie Shotton, Andrew Blake, and Roberto Cipolla, Multi-Scale Categorical Object Recognition Using Contour Fragments, in Trans. on PAMI, July 2008

[VIOL01] Viola, Paul and Jones, Michael. Robust Real-time Object Detection. International Journal of Computer Vision (2001)

[VIOL04] Viola, P., & Jones, M. J. (2004). Robust real-time face detection. International journal of computer vision, 57(2), 137-154.

[WEBE00] Weber, M., Welling, M., and Perona, P. 2000. Unsupervised Learning of Models for Recognition. In Proceedings of the 6th European Conference on Computer Vision-Part I -(June 26- July 01, 2000). D. Vernon, Ed. Lecture Notes In Computer Science, vol. 1842. Springer-Verlag, London, 18-32.

[ZHU06] Zhu, Q., Yeh, M., Cheng, K., and Avidan, S. 2006. Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. In Proceedings of the 2006 IEEE

Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (June 17 - 22, 2006). CVPR. IEEE Computer Society, Washington, DC, 1491-1498.

Recursos web

[W.1] http://www.expansion.com/emprendedore-empleo/empleo/2016/04/22/571a193c22601d24078b4614 .html

[W.2] http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

[W.3] https://towardsdatascience.com/the-10-deep-learning-methods-ai-practitioners-need-to-apply-885259f402c1

[W.4] http://www.estadistica.net/ECONOMETRIA/CUALITATIVAS/LOGISTICA/

regresion-logistica.pdf

[W.5] https://www.lucidchart.com/pages/decision-tree

[W.6] http://www.statsoft.com/Textbook/k-Nearest-Neighbors

[W.7] http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema1dm.pdf

[W.8] http://www.ctrl.cinvestav.mx/~yuw/pdf/MaTesJAR.pdf

[W.9] http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/

confusion_matrix.html

Anexos

Anexo 1: Código

Anexo 1.1: Código de anotación a partir del modelo creado

```
# USO
# python deep_learning_with_opencv.py --image IMAGEN A CLASIFICAR --prototxt ARCHIVO prototxt PARA CLASIFICACION --model "MODELO OBTENIDO DEL ENTRENAMIENTO --labels ARCHIVO CON LAS ETIQUETAS PARA LA ANOTACION

# importar paquetes necesarios
import numpy as np
import argparse
import time
import cv2

# construir la recepcion de argumentos
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
        help="path to input image")
ap.add_argument("-p", "--prototxt", required=True,
        help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
        help="path to Caffe pre-trained model")
ap.add_argument("-l", "--labels", required=True,
        help="path to ImageNet labels (i.e., syn-sets)")
args = vars(ap.parse_args())

# cargar la imagen del disco
image = cv2.imread(args["image"])

# cargar las etiquetas del disco
rows = open(args["labels"]).read().strip().split("\n")
classes = [r[r.find(" ") + 1:].split(",")[0] for r in rows]
```

```
28.
29. # la red neuronal utiliza requiere ciertas dimensiones para la clasifica-
    cion, en nuestro caso 227*227
30. # normalizamos las entradas con los parametros (104, 117, 123) ;
31. # Despues de esto nuestro blob tendra la siguiente forma:
32. # (1, 3, 227, 227)
33. blob = cv2.dnn.blobFromImage(image, 1, (227, 227), (104, 117, 123))
34.
35. # cargamos nuestro model del disco
36. print("[INFO] loading model...")
37. net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
38.
39. # fijamos el blob creado como entrada a la red y continuamos con la clasi-
    ficacion
40. net.setInput(blob)
41. start = time.time()
42. preds = net.forward()
43. end = time.time()
44. print("[INFO] Tiempo en clasificar {:.5} segundos".format(end - start))
45.
46. # ordenamos las clases en funcion de la probabilidad obtenida en la pre-
    diccion
47. idxs = np.argsort(preds[0])[::-1][:5]
48.
49. # bucle de las 5 mejores predicciones y anotacion
50. for (i, idx) in enumerate(idxs):
51.
52.         if i == 0:
53.                 text = "Label: {}, {:.2f}%".format(classes[idx],
54.                         preds[0][idx] * 100)
55.                 cv2.putText(image, text, (5, 25),  cv2.FONT_HERSHEY_SIM-
    PLEX,
56.                         0.7, (0, 0, 255), 2)
57.
58.
59.         print("[INFO] {}. Etiqueta: {}, probabilidad: {:.5}".format(i + 1,
60.                 classes[idx], preds[0][idx]))
61.
```

```
# Mostrar la imagen final
cv2.imshow("Image", image)
cv2.waitKey(0)
```

Anexo 1.2: Código de anotación múltiple a partir del modelo NetMobileSSD

```
1.  # USO
2.  # python real_time_object_detection.py --prototxt MobileNetSSD_deploy.pro-
    totxt.txt --model MobileNetSSD_deploy.caffemodel
3.
4.  # importación de paquetes necesarios
5.  from imutils.video import VideoStream
6.  from imutils.video import FPS
7.  import numpy as np
8.  import argparse
9.  import imutils
10. import time
11. import cv2
12.
13. # constructores de los argumentos
14. ap = argparse.ArgumentParser()
15. ap.add_argument("--video", help="path to video file. If empty, camera's
    stream will be used")
16. ap.add_argument("-p", "--prototxt",default="MobileNetSSD_deploy.pro-
    totxt.txt",
17.         help="path to Caffe 'deploy' prototxt file")
18. ap.add_argument("-m", "--model",default="MobileNetSSD_deploy.caffemodel",
19.         help="path to Caffe pre-trained model")
20. ap.add_argument("-c", "--confidence", type=float, default=0.2,
21.         help="minimum probability to filter weak detections")
22. args = vars(ap.parse_args())
23. args1 = ap.parse_args()
24.
25. # inicializacion de las listas de etiquetas para las que MobileNetSSD fue
    creado
26. CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
27.         "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
28.         "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
29.         "sofa", "train", "tvmonitor"]
30. COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
31.
```

```
32. # carga del modelo del disco
33. print("[INFO] cargando el modelo...")
34. net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
35.
36. # inicializacion del video en directo o video y del contador FPS (Frame
    per Second)
37. print("[INFO] empezando el video...")
38. if args1.video:
39.     vs = cv2.VideoCapture(args1.video)
40. else:
41.     vs = VideoStream(src=0).start()
42.
43. time.sleep(2.0)
44. fps = FPS().start()
45.
46. # bucle sobre los frames del video capturado
47. while True:
48.         # Adaptacion del video para la reproduccion del tamaño deseado
49.         if args1.video:
50.             ret, frame = vs.read()
51.             frame = cv2.resize(frame, (480,360))
52.         else:
53.             frame = vs.read()
54.             frame = imutils.resize(frame, width=400)
55.
56.
57.
58.         # redimensionamiento del frame y conversion al blob
59.     # en este caso el modelo fue creado con dimensiones 300x300
60.         (h, w) = frame.shape[:2]
61.         blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
62.                 0.007843, (300, 300), 127.5)
63.
64.         # pasamos el blob por la red y obtenemos las predicciones y detec-
    ciones
65.         net.setInput(blob)
66.         detections = net.forward()
67.
```

```
68.         # bucle sobre las detecciones
69.         for i in np.arange(0, detections.shape[2]):
70.                 # extraccion de las confidencias como la probabilidad
71.                 confidence = detections[0, 0, i, 2]
72.
73.                 # comprobación de la calidad de las detecciones
74.                 if confidence > args["confidence"]:
75.
76.                         idx = int(detections[0, 0, i, 1])
77.                         box = detections[0, 0, i, 3:7] * np.ar-
    ray([w, h, w, h])
78.                         (startX, startY, endX, endY) = box.astype("int")
79.
80.                         # dibujar la prediccion en la imagen
81.                         label = "{}: {:.2f}%".format(CLASSES[idx],
82.                                 confidence * 100)
83.                         cv2.rectan-
    gle(frame, (startX, startY), (endX, endY),
84.                                 COLORS[idx], 2)
85.                         y = startY - 15 if startY - 15 > 15 else startY
    + 15
86.                         cv2.putText(frame, label, (startX, y),
87.                                 cv2.FONT_HERSHEY_SIMPLEX, 0.5, COL-
    ORS[idx], 2)
88.         print label
89.
90.
91.         # muestra de la imagen final
92.         cv2.imshow("Frame", frame)
93.         key = cv2.waitKey(1) & 0xFF
94.
95.         # si pulsamos q, salir del programa
96.         if key == ord("q"):
97.                 break
98.
99.         # refrescar el contador FPS
100.             fps.update()
101.
```

```
102.    # parar el temporizador y mostrar resultados de la ejecucion
103.    fps.stop()
104.    print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
105.    print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
106.
107.    # limpiamos las pestañas abiertas
108.    cv2.destroyAllWindows()
109.    if args1.video:
110.        print "video"
111.    else:
112.        vs.stop()
```