ISSN 1870-4069

Proceso de desarrollo de aplicaciones para el dominio social media usando técnicas de Deep Learning

Aarón Colmenares Morales[1], Laura Nely Sánchez Morales[1], Giner Alor Hernández[1], José Luis Sánchez Cervantes[2], Ulises Juárez Martínez[1]

[1] Tecnológico Nacional de México/ I.T. Orizaba, Veracruz, México
[2] CONACYT-Tecnológico Nacional de México/ I.T. Orizaba, México
ingacm3000@gmail.com, lauransanchezmorales@gmail.com, galor@ito-depi.edu.mx,jlsanchez@conacyt.mx, ujuarez@ito-depi.edu.mx

Resumen. Los avances del Deep Learning, son cada vez más notorios en el desarrollo de software. Sin embargo, a pesar de tener un gran avance aún no se explotan las técnicas de Deep Learning para los generadores automáticos de software, con atención especial en el desarrollo de aplicaciones *Social Media*. Por lo tanto, se propone un proceso de generación automática de aplicaciones basado en técnicas de Deep Learning, a partir de una imagen digital que representa una interfaz de usuario. Para lo cual se deben identificar UIDPs (*User Interface Design Patterns*) válidos para interfaces de aplicaciones de *Social Media* a través de técnicas de aprendizaje profundo. Las ventajas de este trabajo es el enriquecer las aplicaciones, reducir costos y mejorar el desempeño. Por último, se presenta un caso de estudio que muestra la implementación de las técnicas de Deep Learning en la identificación de UIDPs.

Palabras clave: generación automática de software, procesamiento de imágenes, proceso de desarrollo, aprendizaje profundo, social media, UIDPs.

Application Development Process for the Social Media Domain using Deep Learning Techniques

Abstract. Advances on Deep Learning techniques are becoming more notorious in software development. However, Deep learning techniques are not yet used for automatic software development into Social Media domain. This paper propones an automatic development process of social media applications based on Deep Learning techniques. The proposed process generates applications from a digital image that represents a User Interface Design Pattern (UIDP). The advantages of this work are to enrich applications, to reduce costs and to improve performance. Finally, a case study that shows the implementation of Deep Learning techniques for identification of UIDPs is presented.

Keywords: automatic software generation, image processing, development process, deep learning, social media, UIDPs.

1. Introducción

Día a día se generan nuevos y variados métodos en el desarrollo de software más rápidos, certeros y confiables. Desde el punto de vista de la ingeniería del software y de forma específica dentro de la etapa de diseño de software, los UIDPs son una solución recurrente que resuelve problemas comunes de diseño, como son: las resoluciones de diferentes dispositivos, el diseño responsivo o problemas de navegación; todas estas soluciones se plantean a través de un lenguaje común para los diseñadores. Por su parte, el uso de aplicaciones de dominio *Social media*, cuenta con una gran demanda para los usuarios, debido a que se han expandido los nichos de mercado cada vez más al entorno social.

Por otra parte, el Deep Learning es un subconjunto de técnicas de inteligencia artificial que permite modelos computacionales compuestos de múltiples capas para aprender un conjunto de datos etiquetados, con el contenido de varias capas permiten el reconocimiento del lenguaje, la detección de algún objeto e inclusive la traducción. Dentro del Deep Learning encontramos las CNN (*Convolutional Neural Networks*), las cuales son diseños para procesar datos que llegan en forma de múltiples arreglos, generando una rápida detección de bordes, líneas, entre otras características simples.

De acuerdo a la literatura revisada en este documento encontramos que a pesar de que existen generadores de software similares y se encuentra orientados a otros dominios, existen también procesos poco orientados para las aplicaciones *Social Media* que manipulen técnicas de Deep Learning y ayuden en el reconocimiento de UIDPs en imágenes. Desde esta perspectiva, la identificación de cada UIDP es importante para clasificar y determinar el tipo de dispositivo adecuado para su funcionamiento. Tal situación, motiva a plantear un proceso de generación automática de aplicaciones basado en técnicas de Deep Learning.

El primer pilar de tres en esta investigación, es la generación de software aplicando el uso de una imagen o un boceto generado a mano alzada. El segundo pilar, es el desarrollo de software basado en el aprendizaje profundo, donde una de las características principales que se aprovechó en el desarrollo de software es el aprendizaje a través del entrenamiento, el cual mejora el tiempo de desarrollo. El tercer pilar son los UIDPs, ya que permiten identificar y restaurar ciertos errores de una aplicación, entre los cuales permiten mantener una resolución estándar y rectificar errores generados de diseño por el desarrollador, cumpliendo con una aplicación fácil de usar para el usuario.

Como contribuciones para la ingeniería de software, se tienen cuatro aspectos importantes: (1) generar beneficios para que el proceso de desarrollo de software sea mucho más intuitivo y que sea tan bueno como lo hace un ser humano, (2) diseñar un modelo de *machine learning* y un conjunto de reglas que permitan la generación automática de código, (3) tener un mejor desempeño para aumentar la precisión en la clasificación de imágenes que mejore los ciclos de desarrollo del software y (4) enriquecer las aplicaciones que se sumen a las actividades diarias y aumente la motivación para el desarrollo de software.

Este documento está estructurado de la siguiente manera, en la sección 2 se presenta el estado del arte referente a los diversos trabajos relacionados para la

generación automática de software, divido en dos partes: herramientas para la generación de aplicaciones o desarrollo de software y técnicas Deep Learning en el desarrollo de software. En la sección 3 se hace una descripción de los UIDPs identificados, analizados y aplicados para la generación automática de aplicaciones del dominio *Social Media*. En la sección 4 se detalla el prototipo para la clasificación de interfaces de usuario. Finalmente, en la sección 5 se presentan las conclusiones y trabajo a futuro.

2. Estado del arte

A continuación, se presenta la revisión del estado del arte sobre los trabajos relevantes para la generación de aplicaciones o desarrollo de software y técnicas de Deep Learning en el desarrollo de software.

2.1. Herramientas para la generación de aplicaciones o desarrollo de software

Para las herramientas de generación de aplicaciones se reportan trabajos como Rosales-Morales et al. [1] donde proponen una metodología para el desarrollo de software, y a través de una evaluación cualitativa y cuantitativa, se obtienen resultados que favorecen a Visual Paradigm y Dreamweaver como herramientas para el desarrollo de software. En Cortes-Camarillo et al. [2] analizaron diversos patrones de diseño, para diferentes dispositivos y plataformas para el desarrollo aplicaciones educativas. Cortes-Camarillo et al. [3] presentaron Edugene, una herramienta enfocada en crear contenido educativo parar tres tipos de aplicaciones como MOOC (*Massive Online Open Courses*), blog o wiki. Desde una perspectiva diferente, en Cortes-Camarillo et al. [4] presentaron Atila, un generador de aplicaciones educativas basado en UIDPs. La característica principal de Atila es generar aplicaciones rápidas; además, establece un lenguaje nativo para mejorar el desempeño de las aplicaciones, la compatibilidad y generar código a partir de un archivo XML. En Sánchez-Morales et al. [5] se propuso un proceso que consta de tres fases: análisis de imagen, configuración y generación de código fuente; el resultado dio una contribución para la ingeniera de software. Da-Cost, Neto y de-Oliveira [6] presentaron un estereotipo para MDD (*Model-Driven Development*). El estereotipo UI (*User Interface*) basado para portales Web, captura los procesos y las especificaciones de la UI. Este modelo reduce el desarrollo *time-to-market*, esfuerzo y costo, además contribuye a la calidad, productividad y conservando el funcionamiento en las aplicaciones Web. Por otra parte, Sánchez-Morales et al. [7] propusieron el desarrollo de un componente de software que utiliza técnicas de procesamiento de imágenes y reconocimiento de patrones para la generación automática de aplicaciones multidispositivo. Como resultados se obtiene el reconocimiento de diferentes elementos que forman parte de una imagen representativa de una interfaz de usuario.

2.2. Técnicas Deep Learning en el desarrollo de software

Dentro del desarrollo del software el uso de técnicas de Deep Learning son importantes y tienen diversas aplicaciones. Desde esta perspectiva, en Halbe y Joshi [8] presentaron un enfoque que recibe una imagen escaneada previamente del diseño de la GUI (*Graphical User Interface*), el sistema segmenta los controles HTML. El enfoque logró reconocer los siguientes controles: botón de radio, casilla de verificación, cuadro de texto y botón de comando. Por otra parte, en Baveye et al. [9] platearon MemoNet, los resultados sugieren que el modelo depende de las emociones inducidas por las imágenes y la información emocional incrementa el desempeño del modelo para imágenes neutrales y positivas.

En Bianco et al. [10] destacaron que, para el reconocimiento correcto de una imagen de un logotipo, la imagen debe de estar compuesta y ejecutada por una CNN. Los resultados confirmaron la factibilidad de los métodos propuestos superando los del estado del arte. Liu et al. [11] proponen que los métodos de imagen fusionada basados en Deep Learning se aplicaron para fotografía digital, multi modularidad de imagen e imágenes de teledetección. Los resultados sostienen que existen dificultades en los trabajos convencionales y son resumidas en métodos de fusión y evaluaciones. Pang et al. [12] crearon un gran conjunto de datos ad-hoc con ejemplos positivos y negativos desde la base de datos.

Los resultados muestran, que el método Deep Learning de seguimiento de objetivos visuales es más competitivo con respecto a los algoritmos del estado del arte. Krishevsky, Sustskever y Hinton [13] emplearon un nuevo método de regularización llamado Dropout. Los resultados demostraron que las pruebas de la red visual tienen un alto índice de clasificación y asignación al nombrar correctamente las imágenes. Huang et al. [14] planteó que el MRBDL (*Multi-concept Retrieval using Bimodal* Deep Learning) capturó correlaciones semánticas entre una imagen visual con sus etiquetas libres contextuales.

Estas pruebas muestran la mejora en los datos semánticos, visuales y de texto. Nedzved et al. [15] presentó un esquema en la combinación de características de bibliotecas y de un intérprete con un conjunto de funciones para el procesamiento de imagen. Como resultado, se dividió en dos partes: (1) a desarrolladores de software profesional, son cambios sin necesidad de compilar; y (2) a usuarios, la cual cambia la interfaz gráfica parar mejorar su entorno.

Los dominios abordados por los trabajos revisados se enfocan principalmente en: UIDPs, ingeniería de software, A.I. (*Artificial Intelligence*). Cada trabajo está ligado a uno, dos y hasta tres aspectos relacionados con: la generación de código fuente, el soporte de UIDPs, el soporte multidispositivo, y el uso de alguna técnica de Deep Learning. Sin embargo, se identificó que los trabajos no cubren todas las características analizadas al mismo tiempo.

Todos los trabajos soportan el desarrollo multidispositivo, pero no necesariamente permiten la generación de código fuente, o la implementación de UIDPs en el desarrollo de sus interfaces, y además el uso de técnicas de Deep Learning como parte de su proceso de desarrollo. Por tal motivo, se observa la oportunidad de abordar estas técnicas para plantear un proceso de desarrollo de software basado en técnicas de

Deep Learning, que parta del uso de imágenes generadas a mano alzada que representen interfaces de usuario.

3. UIDPs para social media

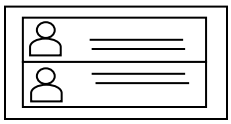
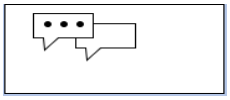
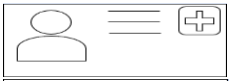
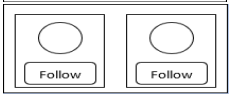
En esta sección se identifican y analizan los UIDPs adecuados y válidos para el desarrollo de aplicaciones *Social Media*. El análisis consistió un conjunto de aplicaciones de *Social Media* existentes, con el objetivo de identificar los UIDPs utilizados en cada aplicación.

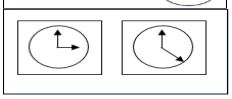
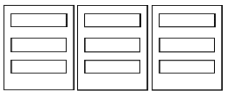
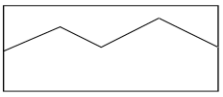
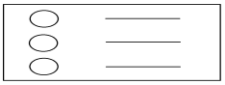
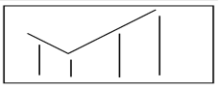
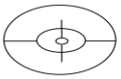
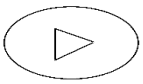
Para las aplicaciones se identificaron dos tipos de UIDPs y se dividen en UIDPs por reputación e interacción. Los UIDPs por reputación permiten construir prestigio a los usuarios mediante el contenido, información compartida, la relación con más personas y seguir sus actividades. Los UDIPs por interacción por el contrario permiten compartir información, conectarse con más personas y seguir sus actividades. En la Tabla 1 se describen los UIDPs identificados por reputación. Y en la Tabla 2 se describen los UIDPs identificados por interacción.

Tabla 1. UIDPs por reputación.

Reputación	Descripción	Imagen
Leaderboards	Se usan para mostrar usuarios altamente competitivos de las actividades que desarrollan.	
Achievements	Se aplica para desplegar las metas o el logro alcanzado en alguna actividad y compartirlo con tu círculo de amigos.	

Tabla 2. UIDPs por interacción.

Interacciones Sociales	Descripción	Imagen
Frient list	Ofrece una manera de explorar a los usuarios de tu círculo cercano, da oportunidad de encontrar contactos frecuentes o la forma de compartir la información con los demás.	
Chat	Permite a los usuarios contactar con otros usuarios de su mismo círculo o externos y provee una forma de comunicación privada de persona a persona.	
Friend	Crea un círculo de personas que desean interactuar y compartir contenido.	
Follow	Permite hacer un seguimiento de la información publicada por un usuario, temas o personas específicas de interés por parte del usuario.	

Interacciones Sociales	Descripción	Imagen
Reaction	Califica el contenido y da información de preferencias y gustos.	
Invite friends	Interactúa con amigos e invita a más amigos.	
Activity Stream	Actualiza las actividades de los contactos y que respondan a las acciones.	
Menú horizontal	Muestra en forma de lista las opciones de la aplicación en forma horizontal.	
Menú Vertical	Muestra en forma de lista las opciones de la aplicación en forma vertical.	
Galería	Da la forma de ver imágenes de modo secuencial, de principio a fin o cuando se desea ver una imagen lo más posiblemente reservada de resolución.	
Panel	Permite ver las características o propiedades del aspecto.	
Ranking	Valora o califica por parte de un usuario un aspecto que contenga la aplicación.	
Picker	Permite seleccionar los ajustes de un calendario, paleta de color, entre otros.	
Music Player	Reproduce música y sonidos seleccionados.	

De los patrones de diseño identificados, se seleccionaron Chat, Galería, Menú horizontal, Menú vertical, Music player, Panel, Picker y Ranking, que son soportados hasta el momento por el prototipo generado.

4. Prototipo

En esta sección se describe un prototipo como prueba de concepto de implementación de las técnicas de Deep Learning para la identificación de UIDPs en imágenes generadas a mano alzada de tipo *Social Media*.

Es necesario mencionar que esta investigación está basada en el proceso de generación de software descrito por Cortes-Camarillo et al. [3], Cortes-Camarillo et al. [4], Sánchez-Morales et al. [5] y Sánchez-Morales et al. [7], el cual consta de tres etapas principales: análisis de la imagen, configuración y generación de código. De

manera particular, en esta investigación se realizaron cambios en la etapa de análisis de la imagen. En este caso, se usa el formato PNG para las imágenes soportadas con una resolución de 1050 por 650 pixeles. Es importante mencionar que el algoritmo no está limitado a esta única resolución de imágenes, es posible adaptar el algoritmo para diferentes resoluciones de forma automática. Sin embargo, para fines de pruebas se aplicó una única resolución. Por otra parte, en este trabajo se aplican algoritmos de Deep Learning a través de una red neuronal convolucional usando los modelos de AlexNet, LeNet o de propia creación; el proceso que realiza la red neuronal convolucional consiste en tomar una parte de una imagen y extraer un subconjunto de pixeles para la siguiente capa, así consecutivamente hasta la clasificación de cada UIDP encontrado en la imagen. Para la etapa de configuración, el tipo de aplicaciones soportadas son *Social Media* de tipo Microbloggin, Streaming, y Redes sociales; y se tiene soporte para aplicaciones móviles para las plataformas Android™ y iOS™ y para Web con HTML5

Finalmente, en la generación de código para la plataforma Web el código generado es HTML5 combinado con el *framework* de Boostrap, este *framework* es utilizado ampliamente para navegar de forma responsiva en cualquier dispositivo tanto de escritorio como móvil; para la plataforma Android se tienen proyectos en Java; por último, para iOS™ se tiene Objetive-C. A continuación, en la Fig. 1 se observa un diagrama de flujo con cada una de las etapas que retroalimenta el proceso de generación de software. De manera específica, en este prototipo la aportación principal se centra en la identificación de UIDPs a través de la CNN, donde los algoritmos utilizados dadas sus características permiten obtener mejores resultados de identificación de UIDPs en imágenes generadas a mano alzada con respecto a un trabajo previo reportado en Sánchez-Morales et al. [7].

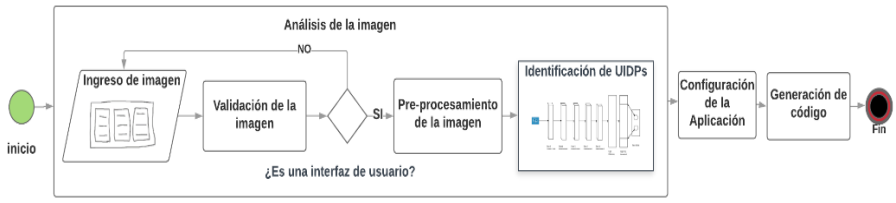


Fig. 1. Diagrama de Flujo del proceso de generación de software a partir de técnicas Deep Learning.

El prototipo desarrollado se programó utilizando Python 3.6.0. y en el entorno de desarrollo Spyder con la versión 3.3.2. Python es un intérprete, orientado a objetos y un lenguaje de programación de alto nivel con semántica dinámica. Comúnmente usa un script o conexión de diálogo para conectar componentes ya existentes además es de código abierto [16].

Para desarrollar el prototipo se realizaron un conjunto de pruebas de entrenamiento para obtener el modelo final. En cada prueba se utilizaron un conjunto de imágenes clasificadas en 4 tipos: imágenes ideales, no ideales, reales en blanco y negro y reales a color. Las imágenes ideales son imágenes generadas mediante cualquier herramienta de dibujo y no presentan deformaciones (ver Fig. 2-a).

Las imágenes no ideales son aquellas generadas a mano alzada, desde cualquier aplicación o herramienta de dibujo o bien a través de una tableta digitalizadora (ver Fig. 2-b). Las imágenes reales en blanco y negro son imágenes generadas a mano alzada en hojas de papel y digitalizadas a través de una cámara o un scanner (ver Fig. 2-c). Finalmente, las imágenes reales a color son imágenes generadas a mano alzada, en hojas de papel de diferentes colores utilizando bolígrafos también de diferentes colores (ver Fig. 2-d).

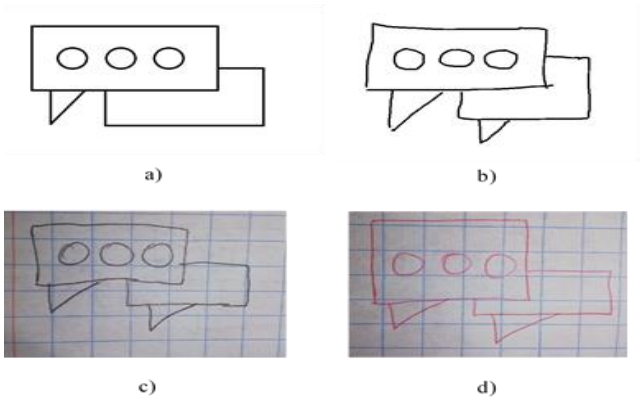


Fig. 2. Tipos de imágenes utilizadas para el entrenamiento de la red neuronal convolucional: a) imagen ideal, b) imagen no ideal, c) imagen real en blanco y negro, d) imagen real a color.

Los aspectos de cada prueba se encuentran organizados en Tablas de la siguiente forma. En la primera fila, se tiene el número de la prueba con el respectivo nombre de la API y el tipo de modelo aplicado, seguido por el total de imágenes usadas para el entrenamiento. En la segunda fila, se observan los encabezados para el nombre de los UIDPs que se emplearon en las pruebas, la precisión y el tiempo de respuesta del modelo. Al final de cada tabla, se tiene los batch y épocas realizados por cada prueba de entrenamiento. Para la prueba 1 ver la Tabla 3, para la prueba 2 ver la Tabla 4, para la prueba 3 ver la Tabla 5 y para la prueba 4 ver la Tabla 6.

Tabla 3. Resultados de la prueba 1.

Prueba 1. DeepLearning-Lenet		250 imágenes
UIDPs	Precisión	Tiempo (seg.)
Acordeon	.45	21.32
Barra de etiqueta	.45	21.32
Login	.45	21.32
Master Detalle	.45	21.32
Tarjeta	.45	21.32
256 batch y 3 épocas		

Tabla 4. Resultados de la prueba 2.

Prueba 2. Tensorflow-Libre		600 imágenes
UIDPs	Precisión	Tiempo (seg.)
Galería	1.0000000	30.00
Reproductor de música	3.1575022e-13	45.0
1699 batch y 100 épocas		

Tabla 5. Resultados de la prueba 3.

Prueba 3. TensorFlow-Libre		800 imágenes
UIDPs	Precisión	Tiempo (seg.)
Galería	0.4339219	23.407269
Menú horizontal	0.422323233	23.407269
Menú vertical	0.123212333232	23.407269
Panel	0.0221412	23.407269
1401 batch y 10 épocas		

Para la última prueba se realizó un pre-procesamiento, que consiste en la conversión de una imagen natural a una imagen binaria aplicando el procesamiento de imágenes. Para ejemplificar este proceso se usó una imagen generada a mano alzada que representa un Menú horizontal (ver Fig. 3-a). El resultado de convertir la imagen en escala de grises se observa en la Fig. 3-b. Posteriormente, el segundo paso es el uso del algoritmo de núcleo gaussiano que permite eliminar el ruido de los tonos grises [17] y ocupar solamente los bordes obtenidos (ver Fig. 3-c). Cabe mencionar que se pueden presentar restricciones ante el cambio de brillo de una imagen ocasionando que los algoritmos usados en el pre-procesamiento omitan algunos bordes del patrón. Lo anterior propicia que no se reconozcan los UIDPs correctamente dentro de la imagen, para solucionar este problema es necesario combinar otros algoritmos que mejoren el contraste y brillo de la imagen sin perder los bordes.

Fig. 3. Pasos del pre-procesamiento: a) imagen original de tipo real en blanco y negro, b) resultado del método de escala de grises y c) resultado del método de núcleo gaussiano.

Una vez realizado el pre-procesamiento de la imagen, se realiza el entrenamiento de la red neuronal convolucional y los resultados obtenidos se observan en la Tabla 6.

Tabla 6. Resultados de la prueba 4.

Prueba 4. TensorFlow-AlexNet		1400 imágenes
UIDPs	Precisión	Tiempo (seg.)
Gallería	.99983644	8.24
Menú horizontal	.56006614	8.32
Menú vertical	.99995962	8.40
Music player	.9624471	8.18
Panel	.9980167	7.98
Picker	.8734556	7.91
Ranking	.98580235	7.85
699 batch y 10 épocas		

Como se observa, los resultados obtenidos en la prueba 4 fueron mucho más exactos, obteniendo un 99% de eficiencia en la clasificación de los UIDPs y con un tiempo de respuesta de 8 segundos en promedio. Por lo tanto, este es el modelo final seleccionado para la clasificación de imágenes generadas a mano alzada que representen interfaces de usuario de aplicaciones *Social Media*.

5. Caso de estudio

En esta sección se aborda un caso de estudio para ejemplificar el funcionamiento del prototipo desarrollado.

En este caso de estudio, se seleccionaron 6 patrones de diseño adecuados para generar un prototipo de aplicación de Social Media. El prototipo es una aplicación para la reproducción de música y utiliza los UIDPs descritos a continuación:

- Galería: para desplegar al usuario las imágenes de álbumes disponibles para reproducir.
- Menú horizontal: despliega las opciones disponibles de la aplicación, como inicio, álbum y configuración.
- Music player: permite reproducir las diferentes pistas de música.
- Panel: permite ver las características del álbum o pista a reproducir.
- Picker: permite elegir el horario del temporalizador para finalizar una canción.
- Ranking: Para calificar una canción o álbum.

Las imágenes utilizadas para representar la aplicación de este caso de estudio tienen dimensiones de 1050 por 650 pixeles en formato PNG. Los UIDPs generados se pueden observar en la Fig. 4.

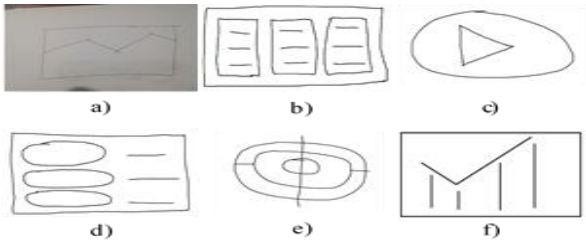
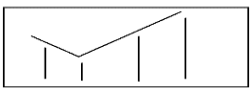
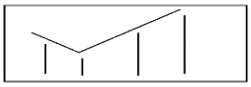
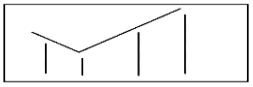
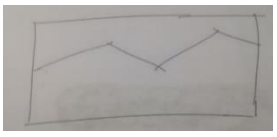
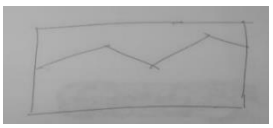
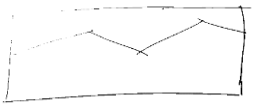
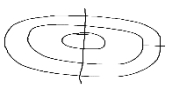
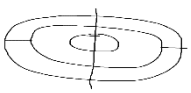
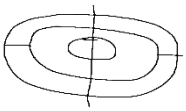


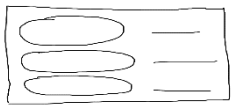
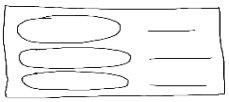
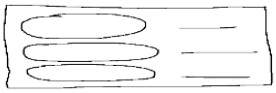
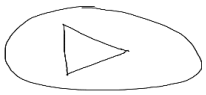
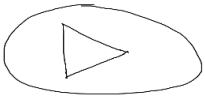
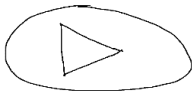
Fig. 4. UIDPs utilizados para la generación de una aplicación social media.: a) Galería, b) Menú horizontal, c) Music player, d) Panel, e) Picker y f) Ranking.

Para utilizar el prototipo de la red neuronal convolucional previamente entrenado, el primer paso es indicar la ruta de la imagen que se va utilizar, en este caso la primera imagen ingresada es una imagen real en blanco y negro que representa una Galería (ver Fig. 4-a), la segunda es una imagen no ideal que representa un Menú horizontal (ver Fig. 4-b), la tercera es una imagen no ideal que representa al Music player (ver Fig. 4-c), la cuarta es una imagen no ideal que representa un Panel (ver Fig. 4-d), la quinta es una imagen no ideal que representa un Picker (ver Fig. 4-e) y la sexta es una imagen ideal que representa un Ranking (ver Fig. 4-f).

El segundo paso, consiste en aplicar un pre-procesamiento a través de un conjunto de algoritmos de procesamiento de imágenes. En la Tabla 7 se observan los resultados obtenidos del pre-procesamiento aplicado a cada patrón.

Tabla 7. Imágenes de prueba: Ranking (ideal), Menú horizontal (no ideal) y Galería (real blanco y negro) con su respectivo preprocesamiento.

Patrón original	Escalas grises	Núcleo Gaussiano
		
Ranking		
		
Menú horizontal		
		
Galería		
		
Picker		

Patrón original	Escalas grises	Núcleo Gaussiano
		
Panel		
		
Music Player		

Una vez aplicado el pre-procesamiento, el paso 3 consiste utilizar la red neuronal convolucional entrenada y los resultados obtenidos para cada patrón se observan en la Tabla 8.

Tal como se puede observar, los resultados de eficiencia de clasificación mostrados en la Tabla 8 obtienen porcentajes superiores al 70% en los patrones Rankin, Galería, Panel, Picker y Music player. Mientras que el Menú horizontal tiene resultados inferiores al 70%, sin embargo, su clasificación es correcta.

Tabla 8. Resultados obtenidos para la identificación de patrones: a = Ranking, b = Menú horizontal, c = Galería, d = Panel, e = Picker y f = Music player.

	a	b	C	d	e	f
a	**.98580235**	.0023476905	.0066808336	.0008515252	.0008665596	.0018575224
b	.008125944	**.560063**	.002517612	.0037316438	.00053650513	.0008372874
c	.0007908905	.0008986282	**.9975389**	2.825912e-05	.00013480928	.0006040252
d	.0005736141	.08623368	.0011079037	**.7965732**	.0006951616	.0010907252
e	.000964381	.007865683	.009596186	.010660858	**.8734549**	.09668942
f	.0001126410	.1.89209e-05	.0007694098	4.704886e-05	3.334518e-05	**.99901175**

Desde el entorno de Spyder se muestran las pantallas de salida con la clasificación de cada patrón (ver Fig. 5).

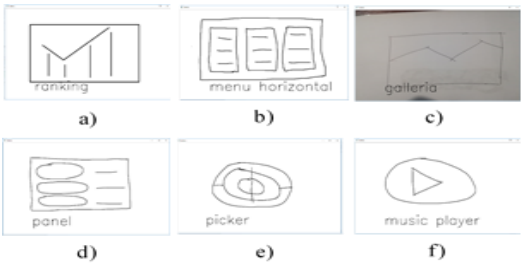


Fig. 5. Resultados de la clasificación de imágenes: a) Ranking, b) Menú horizontal, c) Galería, d) Panel, e) Picker y f) Music player.

Los resultados obtenidos determinan el correcto funcionamiento de la red neuronal convolucional y garantizan una adecuada clasificación, con lo que se tiene un margen de referencia que permitirá ampliar la investigación y abrir la oportunidad para crear un nuevo estándar o enfoque para el desarrollo de software. Lo anterior considerando que el tiempo promedio de respuesta es menor a 60 segundos. Con tal resultado se ve claramente el impacto que genera una red neuronal convolucional para la clasificación de imágenes.

6. Conclusiones y trabajo a futuro

Los resultados obtenidos gracias al apoyo de las técnicas de Deep Learning aportan un gran avance para estimular futuros desarrollos para diferentes ámbitos como *e-Commerce y* aplicaciones médicas. Este aporte en la ingeniería de software, da un panorama amplio para los desarrolladores que buscan procedimientos exactos, confiables y con costo menor de lo promediado. Cabe aclarar que la integración del Deep Learning, conlleva a una nueva línea de investigación para desarrollar e implementar aplicaciones en diferentes dominios, creando nuevas técnicas, métodos, paradigmas y metodologías que ampliarán una expectativa diferente a las técnicas convencionales que aún se trabajan.

El caso de estudio presentado demostró un claro ejemplo de que las diferentes imágenes de prueba (imágenes ideales, no ideales, reales en blanco y negro, y reales a color), pueden ser clasificadas sin ningún problema. Los resultados obtenidos no solamente obtuvieron un impacto importante para la clasificación, sino también el tiempo de espera para reconocer un patrón puede concluirse que es adecuado.

Este trabajo es una extensión de investigaciones previas que se presentan en Cortes-Camarillo et al. [3], Cortes-Camarillo et al. [4], Sánchez-Morales et al. [5] y Sánchez-Morales et al. [7] cuyas pruebas y resultados son un margen de referencia para mejorar los algoritmos de identificación de UIDPs en imágenes generadas a mano alzada.

Para el trabajo a futuro se pretende ampliar el abanico de patrones de diseño para este dominio y considerar nuevos dominios como *e-Commerce* y aplicaciones médicas. También se pretende generar nuevos modelos considerando otras APIs de Deep Learning para poner a prueba la eficiencia de la clasificación con nuevos UIDPs y diseños de aplicaciones más robustas que consideren la composición de patrones de diseño de interfaces de usuario.

Agradecimientos. Los autores agradecen al Instituto Tecnológico de México por su apoyo en este trabajo. Este trabajo de investigación fue patrocinado por el Consejo Nacional de Ciencia y Tecnología (CONACYT), así como por la Secretaría de Educación Pública (SEP) a través del PRODEP.

Referencias

1. Rosales-Morales, V.Y., Alor-Hernández, G., García-Alcaráz, J.L., Zatarain-Cabada, R., Barrón-Estrada, M.L.: An analysis of tools for automatic software development and automatic code generation. Facultad de Ingeniería, No. 77, pp. 75–87 (2015)

2. Cortes-Camarillo, C.A., Alor-Hernández, G., Olivares-Zepahua, B.A., Rodríguez-Mazahua, L., Peláez-Camarena, S.G.: Análisis comparativo de patrones de diseño de interfaz de usuario para el desarrollo. Research in Computing Science, vol. 126, pp. 31–41 (2016)
3. Cortes-Camarillo, C.A., Alor-Hernández, G., Sánchez-Morales, L.N., Rosales-Morales, V.Y., Rodríguez-Mazahua, L., Sánchez-Cervantes, J.L.: EduGene: A UIDP-Based Educational App Generator for Multiple Devices and Platforms. International Journal of Human-Computer Interaction, pp. 1–23 (2018)
4. Cortes-Camarillo, C.A., Rosales-Morales, V.Y., Sánchez-Morales, L.N., Alor-Hernández, G., Rodríguez-Mazahua, L.: Atila: A UIDPs-based educational application generator for mobile devices. In: International Conferece on Electronics, Commnications and Computers (CONIELECOMP), pp. 1–7 (2017)
5. Sánchez-Morales, L.N., Alor-Hernández, G., Miranda-Luna, R., Rosales-Morales, V.Y., Cortes-Camarillo, C.A.: Generation of User Interfaces for Mobile Applications Using Neuronal Networks. New Perspectives on Applied Industrial Tools and Techniques, pp. 211–231 (2017)
6. da-Cost, S.L., Neto, V.V.G., de-Oliveira, J.L.: A User Interface Stereotype to build Web Portals. In: 9th Latin American Web Congress, pp. 10–18 (2014)
7. Sánchez-Morales, L.N., Rosales-Morales, V.Y., Alor-Hernández, G., Posada-Gómez, R., Muñoz-Contreras, H., Juárez-Martínez, U.: Módulo de generación de aplicaciones multidispositivo a partir del procesamiento de imágenes. Research in Computing Science on Computer science and computer engineering, Issue 9, pp. 81–94 (2015)
8. Halbe, A., Joshi, A.R.: A Novel Approach to HTML Page Creation Using Neural Network. Procedia Computer Science, vol 45, pp. 197–204 (2015)
9. Baveye, Y., Cohendet, R., Da-Silva, M.P., Le-Callet, P.: Deep Learning for Image Memorability Prediction: The Emotional Bias. In: Proceedings of the 2016 ACM on Multimedia Conference, pp. 491–495 (2015)
10. Bianco, S., Buzzelli, M., Mazzini, D., Schettini, R.: Deep learning for logo recognition. Neurocomputing, vol. 245, pp. 23–30 (2017)
11. Liu, Y., Chen, X., Wang, Z., Wang, Z. J., Ward, R. K., Wang, X.: Deep learning for pixel-level image fusion: Recent Advances and future prospects. Information Fusion, vol. 42, pp. 158–173 (2018)
12. Pang, S., del-Coz, J.J., Yu, Z., Luaces, O., Díez, J.: Deep learning to frame objects for visual target tracking. Engineering Applications of Artificial Intelligence, vol. 65, pp. 406–420 (2017)
13. Krishevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems, vol. 1, pp. 1–9 (2012)
14. Huang, C., Xu, H., Xie, L., Zhu, J., Xu, C.: Large-scale Semantic Web Image Retrieval Using Bimodal Deep Learning Techniques. Information Sciences, pp. 1–39 (2017)
15. Nedzved, A., Gurevich, I., Trusova, Y., Ablameyko, S.: Software Development Technology with Automatic Configuration to Classes of Image Processing Problems. Software and Hardware for pattern recognition and image analysis, vol. 23, no. 2, pp. 269–277 (2013)
16. What is Python? Executive Summary. https://www.python.org/doc/essays/blurb/. Último acceso: 2018/09/26
17. Gonzalez, R., Woods, R.: Digital image processing. 4th ed. Pearson, New Jersey (2017)