# SISTEMA DE APOYO A INVIDENTES BASADO EN DEEP LEARNNING EMPLEANDO DISPOSITIVO MÓVIL

## Sistema de Apoyo a Invidentes Basado en Deep Learnning Empleando Dispositivo Móvil

**Javier Maldonado Romo**

*Instituto Politécnico Nacional-CIDETEC*

*javier.mr.21@gmail.com*

**Israel Rivera Zárate**

*Instituto Politécnico Nacional-CIDETEC*

*irivera@ipn.mx*

**Miguel Hernández Bolaños**

*Instituto Politécnico Nacional-CIDETEC*

*mbolanos@ipn.mx*

**Resumen**

El presente artículo brinda una descripción de las fases de entrenamiento, desarrollo y optimización de un sistema basado en Deep learning (aprendizaje profundo) que sirva como apoyo a personas invidentes. El sistema propuesto explota las capacidades del aprendizaje automático y su implementación en arquitecturas multiprocesador integradas en los dispositivos móviles a fin de facilitar ya sea la interacción o bien la evasión de objetos comunes en el entorno del usuario. El sistema, realiza procesamiento de video en un dispositivo móvil y aplica técnicas de reconocimiento de objetos basado en Deep learning para generar un mensaje acústico ante la identificación de mobiliario de oficina, particularmente las sillas de distintos tipos y colores ubicadas en distintas posiciones relativas a la persona invidente.

## Introducción

El Deep learning es parte del aprendizaje máquina donde mediante un proceso de aprendizaje supervisado se entrena a redes neuronales multicapa de forma que posteriormente y sin una supervisión explícita la red sea capaz de realizar su tarea sobre un conjunto de datos que jamás ha visto [1]. Esto último se entiende como: “la capacidad que tienen las computadoras de actuar o aprender sin ser explícitamente programadas para ello” [2]. Se debe reconocer que el Deep learning ha mostrado un gran avance en tareas de reconocimiento de voz, visión por computadora, traducción automática así como el reconocimiento de objetos [3].

El trabajo propuesto pretende emplear las técnicas de reconocimiento de objetos del Deep learning para conformar un sistema de apoyo a invidentes en un dispositivo móvil capaz de reconocer hasta 3 tipos diferentes de sillas comúnmente encontradas en oficinas sin que afecte la orientación en que se encuentren. De modo que un usuario invidente recibe un mensaje acústico que le informa de la detección o identificación de objetos silla en su proximidad. La propuesta que se desarrolló en el presente proyecto establece el uso de un dispositivo móvil con una arquitectura Quad core operando a 1.4 GHz. Con 2 MB de memoria RAM con sistema Android 6.0 Marshmallow y cámara fotográfica de 5 Mega pixeles. Este sistema se encargará de la adquisición de imágenes del mundo real para su posterior procesamiento de reconocimiento de objetos, en el caso particular las sillas y su identificación mediante señal audible. Ver figura 1.


**Figura 1. Sistema Propuesto para el reconocimiento de objetos.**

**Desarrollo**

Se propone definir 10 posibles clases conformadas por 3 diferentes tipos de sillas, 3 diferentes tipos de mesas así como una puerta, el monitor de una computadora y hasta una ventana. Una vez establecidas las clases se determinará el ángulo en el que el usuario está posicionado respecto al objeto a partir del entrenamiento que se haga dentro del escenario para cada una de las clases. A partir de la perspectiva del usuario se proyectará un modelo tridimensional en el dispositivo móvil y se generará un mensaje acústico.

La metodología adoptada en el presente proyecto consiste de 4 pasos esenciales:

Metodología

[Paso 1:] Estimación de la pose.

[Paso 2:] Etapa de entrenamiento.

[Paso 3:] Etapa de prueba.

[Paso 4:] Generación acústica en el entorno de realidad aumentada.

**Estimación de la pose**

La parte central del trabajo consiste en la predicción de la pose del objeto a identificar, para ello se obtuvieron muestras de los objetos a predecir considerando su orientación. Se establecieron 8 puntos separados 45º entre cada punto cardinal que corresponde al norte, noreste, este, sureste, sur, suroeste, oeste y noroeste. El clasificador principal contiene 4 clases de las cuales 3 de las clases son sillas y la restante es la clase que hace referencia a la nada. Para realizar la captura de las muestras, se realizó una aplicación de Android que hace uso de los sensores del mismo dispositivo como el acelerómetro y la brújula digital para poder vincular la imagen de la captura con la lectura del dispositivo móvil que es el observador, con esto nos permite crear nuestro conjunto de datos que ahora se tendrán modelos de 8 salidas con el mismo número de entradas. Ver figura 2.

A)

B)

**Figura 2. Muestras para el conjunto de la clase A) NORTE y la clase B) SUR, los objetos están a 0 y 180 grados respectivamente respecto al usuario.**

Una vez que se obtienen las muestras, se debe garantizar el número de muestras satisfagan la clasificación. Para ello, se hará uso del modelo CIFAR-10 que permitirá validar si el número de muestras son suficientes para poder ser clasificadas. La tabla 1 muestra las características del modelo. La figura 3 muestra el Dataset CIFAR que es un conjunto de datos de acceso abierto para crear modelos cuando no se dispone de datos propios para empezar a trabajar. Existen 50,000 imágenes para entrenar y 10,000 para evaluar.

**Tabla 1. Características del modelo CIFAR-10.**

| Capa | Función | Parámetros |
|---|---|---|
| 1 | Convolución espacial | 16 características Kernel 5x5 |
| | Tanh | |
| | MaxPooling | Kernel 2x2 |
| 2 | Convolución espacial | 256 características Kernel 5x5 |
| | Tanh | |
| | MaxPooling | Kernel 2x2 |
| 3 | Linear | Salida 128 |
| | Tanh | |
| | Linear | 8 clases |

Para garantizar una cantidad de muestras cumplan con la obtención de un buen desempeño en clasificación, se utilizan técnicas para ayudar a la clasificación permitiendo reducir el número de épocas, esta técnica se llama aumentación de los datos (Augmented Data) [4]. La técnica de aumentación de los datos permite extraer, así como permitir al sistema sobresalir cantidades de datos que ayuden a disminuir el procesamiento para clasificar los datos.


**Figura 3. El Dataset CIFAR.**

Para aumentar los datos se deben de seguir las siguientes recomendaciones [5], cada una de las muestras se le agrega un poco de ruido puede ser ruido aleatorio o con alguna distribución para que la clasificación de las muestras se realice de una mejor forma bajo estas condiciones, el agregar un ruido a las muestras permitiendo desempeñar una mejor clasificación considerando condiciones imprevistas como los cambios en iluminación ambiental principalmente.

Una vez que se haya agregado algún tipo de ruido a las muestras, se aplican las siguientes características a las mismas para poder incrementar todavía más el conjunto de datos de cada clase. Para las muestras se aplica lo que se conoce como Flip y Crop, la primera lo que aplica es un giro a los datos de la muestra puede ser horizontalmente o verticalmente, generando una muestra totalmente nueva a la anterior, pero con los mismos datos en diferente posición, generando otra muestra más que permite identificar mejor a la clase, la segunda permite seleccionar una zona de la muestra para realizar un acercamiento o zoom de la misma para que el nuevo conjunto extraiga datos específicos de la muestra permitiendo enfatizar las características que conformen a cada clase. Ver figuras 4 y 5 respectivamente.

**Figura 4. Muestras con la técnica de flip y crop.**

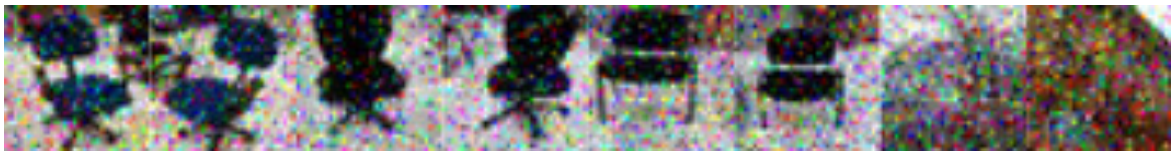

**Figura 5. Muestras con ruido aleatorio del 50%.**

El modelo se definió como una solución general. Ahora, se deben hacer combinaciones hasta determinar el mejor modelo particular para nuestro caso, por lo que se puede considerar el uso de la programación evolutiva donde es una buena práctica observar el comportamiento usando una variación aleatoria. Las diferentes características en cada modelo se obtendrán de la variable aleatoria, el experimento hará una solicitud de servicio web para obtener los números aleatorios para las características de cada modelo, como el tamaño del núcleo, agrupación y función de transferencia principalmente. Entre las funciones de transferencia se han considerado: Tanh, ReLU, Linear, SoftMAx y LogSoftAx [6].

**Etapa de entrenamiento**

El entrenamiento no supervisado se utilizará para poder inicializar los pesos que conforman a las redes neuronales, así como el bias. Se sabe que cuando se realiza un entrenamiento

supervisado se inicializan los valores de los respectivos pesos y bias de forma aleatoria, por lo tanto existen técnicas como el aprendizaje no supervisado permitiendo agrupar los diferentes conjuntos que componen a cada clase, este comportamiento proporciona que los pesos se ajusten, una vez que se estén ajustando los pesos y los bias que componen a las redes neuronales lo que generará que el entrenamiento supervisado comience con la inicialización de las redes más cercas a la función objetivo, debido al realizar el entrenamiento supervisado requerirá un menor tiempo para converger a la solución. Ver tabla 3.

**Tabla 3. Características para el entrenamiento No supervisado.**

| Capa | Característica | Parámetro |
|---|---|---|
| **1** | **Número de kernel** | **32** |
| | **Número de iteraciones** | **20** |
| | **Tamaño del lote** | **1** |
| **2** | **Número de kernel** | **32** |
| | **Número de iteraciones** | **20** |
| | **Tamaño de lote** | **1** |

El aprendizaje profundo no cuenta con reglas definidas para un problema en específico, esto no quiere decir que las soluciones sean no deterministas, sin embargo, no hay soluciones deterministas para este tipo de problemas, pero se tienen una gran cantidad de soluciones, de las cuales todas las posibles soluciones podrán ser buenas o muy buenas, pero no se podría conseguir la mejor solución. Se pueden obtener una gran cantidad de soluciones para problemas de aprendizaje profundo, hasta el momento contamos con una solución general que es el modelo CIFAR-10, pero se ha estado minimizando los tiempos aplicando estrategias para ese fin. Sin embargo la literatura no menciona una metodología para la creación de

modelos, es por ello que se implementó una programación genética [7]. Ver algoritmo siguiente.

```
Data: 5 individuos para generar la población
Result: 500 modelos evolutivos
Iniciailización;
for 500 iteraciones para cada uno de los modelos do
    Leer semilla actual;
    N ← numero aleatorio% 100;
    if N ≤ 25 then
        Generación de mutación ;
        M ← numero aleatorio% 100;
        if M ≤ 50 then
            Mutación para una hoja, seleccionar la función de transferencia,
             pooling o el tamaño del kernel.
        end
        else
            Mutación para la rama, seleccionar un tamaño aleatorio del
             tamaño actual de profundidad del árbol, seleccionar la función
             de transferencia, pooling o el tamaño del kernel.
        end
    else if N > 25 and N < 50 then
        Generación de reproducción;
    else
        Generación de cruza ;
        Seleccionar 2 individuos;
        Individuo 1: ;
        C ← numero aleatorio% 100;
        if C ≤ 50 then
            Selección para una hoja
        end
        else
            Selección de una rama del tamaño de profundidad del árbol
        end
        Individuo 2: ;
        C ← numero aleatorio% 100;
        if C ≤ 50 then
            Selección para una hoja
        end
        else
            Selección de una rama del tamaño de profundidad del árbol
        end
        Intercambio entre los dos individuos;
end
```

Para generar los individuos aleatorios, se definirán sus respectivas características a partir de números aleatorios, para ello se realizó un programa que genera hasta 1000 individuos con características totalmente aleatorios a partir de las semillas generadoras que se obtienen del medio ambiente a través de un servicio web. Ver tabla 4.

**Tabla 4. Construcción de características aleatorias.**

| Capa | Característica | Probabilidad |
|---|---|---|
| **Función de Transferencia** | **Tanh** | **20%** |
| | **ReLU** | |
| | **Lineal** | |
| | **SoftMax** | |
| | **LogSoftMax** | |
| **Max Pooling** | **Aparecer** | **50%** |
| | **No aparecer** | |
| **Tamaño Kernel** | **1 -32** | **3.125%** |

A cada uno de los 1000 individuos se ejecutará con el conjunto de datos que se ha utilizado con las adecuaciones e implementaciones antes mencionadas. Por cada modelo se obtiene los tiempos de ejecución, así como la efectividad de la clasificación por lo tanto, por este medio se obtendrá la población inicial de 5 individuos que podrán mejorar a la solución general que se experimentó con el modelo cifar10. Los mejores individuos se muestran sus características en las siguientes tablas.

**Etapa de prueba**

Se llevaron a cabo los pasos de la metodología descrita, la cual detalla los diferentes puntos importantes de la metodología implementada, describiendo los resultados obtenidos. Todos los experimentos se realizaron con un control de paro a 10 épocas, este parámetro es una bandera para determinar el rendimiento y la exactitud para todos los modelos, la época número 10 es un parámetro que se tomó sin alguna consideración fue puramente empírica la decisión, debido a que puede haber modelos que no se completen del todo debido a sus

respectivas características, por lo tanto para evitar ciclos infinitos se consideraran solamente 10 épocas.

La primera evaluación es la ejecución del conjunto de datos con la técnica de aumentación de datos, no es necesario ejecutar las muestras sin la técnica mencionada, debido a que es una recomendación implementar, por lo tanto se obtuvo una exactitud del 76.06% en un tiempo de 128 segundos, este resultado se debe de mejorar debido que el modelo original cifar10.

El siguiente paso es inicializar las variables que componen a las redes neuronales como el peso y el bias de cada neurona, el cual es implementado a través de un entrenamiento no supervisado, una vez que se haya realizado el entrenamiento se inicializan los pesos del modelo original anterior, se obtiene una exactitud en 10 épocas aumento la exactitud a 81.07% en un tiempo de 112 segundos, como se puede observar inicializar los pesos con un entrenamiento previo permite reducir el tiempo computacional obteniendo una mejor respuesta en el mismo número de épocas ejecutadas, además el tiempo también se redujo.

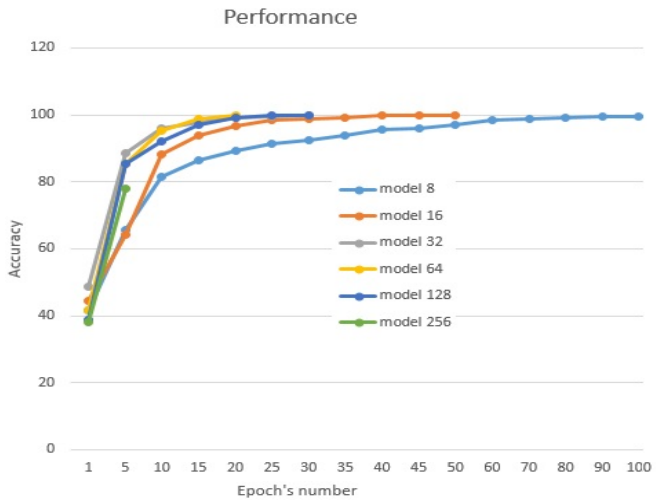


**Figura 6. Desempeño alcanzado por época de acuerdo al modelo**.

La figura 6 muestra el comportamiento de cada modelo hasta obtener la precisión del 100%. El modelo de 256 píxeles tiene muchos datos, es lento en comparación con otros modelos; ha requerido 46 minutos para calcular en tres épocas, por lo que el resultado queda lejos de un buen desempeño. Por lo tanto, el modelo de 256 píxeles se descarta.

**Generación acústica**

Una vez que se obtenga el entrenamiento al 100% de exactitud el programa se cancela la ejecución para realizar la implementación. Bajo la siguiente arquitectura cliente servidor con un dispositivo móvil. Debido a que el móvil no es capaz de ejecutar las tareas, por esa razón es necesario la implementación en la nube para poder obtener el nivel de procesamiento necesario para la clasificación.

El dispositivo móvil es capaz de ejecutar el aprendizaje profundo, pero debido a las limitantes del mismo se tomó la decisión de realizar el procesamiento de la clasificación en la nube, utilizando una computadora de preferencia con tecnología CUDA para poder realizar con mejor rendimiento el tiempo del procesamiento, para ello se requiere un enlace de comunicación entre el dispositivo móvil y el equipo de cómputo [8].

Se implementó un socket TCP, pero debido a las características del socket que evita la perdida de los paquetes ofrece un tiempo para la experiencia de usuario un promedio de 3 fotogramas por segundo, es buena la respuesta ya que la imagen original es recibida por el servidor. Con un socket UDP el tiempo de respuesta incrementa a 7 fotogramas por segundo prácticamente es el doble en comparación con el socket TCP, pero también la imagen llega con pérdida de datos, ya que es una imagen que está compuesta por una secuencia de colores, la imagen recibida es rellenada con datos basura generando imagen desconocida, para evitar eso se tiene la clase de la nada para cuando sea detectada esa imagen sea clasificada como algo que no existe en el escenario real. Ver figura 7 siguiente.

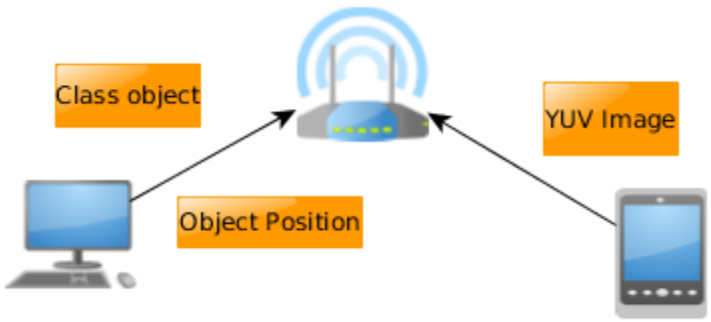


**Figura 7. Comunicación entre el dispositivo móvil y el equipo de cómputo.**

El socket UDP es la mejor opción, sin embargo, mantiene un índice bajo para el requerido que es de 24 fotogramas por segundo, para ello se realiza un cambio de codificación de imagen. La

imagen original tiene un formato RGB, por lo que una imagen de 640x480 en este formato alcanza 900KB, es una imagen muy cerca de 1MB, pero al implementar un cambio de formato a YUV se reduce a la mitad que es 450KB, para realizar el cambio de formato se implementan las siguientes ecuaciones, a partir de los valores del formato RGB se puede hacer una compresión de los datos para reducir el ancho de banda.

$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$

$$U = -0.1678 * R - 03313 * G + 0.5 * B + 128$$

$$V = (R - Y) * 0.713 + 128$$

La imagen se comprime a la mitad de datos para poder ser enviada por el cliente que es el dispositivo móvil para ser recibida por el equipo de cómputo que realiza la conversión de YUV a RGB logrando obtener la imagen original, este procedimiento alcanza los 9 frames por segundos, ejecutado en todo el sistema, para poder incrementar más el tiempo de respuesta es reduce la imagen a 320x240 reduciendo así la imagen y su respectivo ancho de banda a 225KB, para ser llevada a cabo se aplica el procedimiento a los pixeles pares. Con esta implementación se alcanza hasta 12 frames por segundo en promedio, es posible reducir la imagen todavía más su tamaño, pero se considera que la imagen a esta resolución mantiene información importante para futuras implementaciones. Para realizar esta integración en el equipo de cómputo, el servidor contiene los siguientes bloques que constituyen la arquitectura de la integración de diferentes herramientas de trabajo.

Teniendo la arquitectura del dispositivo móvil de acuerdo a la siguiente arquitectura, la cual se utiliza un servicio para hacer la adquisición de la imagen y el respectivo envió de la misma a través de sockets UDP para que el servidor la clasifique e indique el objeto detectado, posteriormente se realiza la reproducción de audio con la biblioteca TTS de Android. Esta API realiza la interpretación de texto a audio una vez que se indique el objeto clasificado.

**Resultados y Conclusiones**

En el presente trabajo se describió el comportamiento y las características de los diferentes modelos que se aplican a un clasificador de cuatro clases. Se observó que el modelo con menos dimensiones como 8 y 16 píxeles, estos modelos tienen poca información en comparación con el modelo de 32 píxeles. El modelo de 8 píxeles no es recomendable porque tiene poca información por lo que podría producir mucho error. El modelo de 16 y 32 píxeles son excelentes; ambos modelos son ideales para aplicaciones en tiempo real porque su tiempo de respuesta es rápido y preciso. Por su parte el modelo de 64 píxeles para el procesamiento de información es el mejor porque tiene mucha información, pero requiere al menos 5 minutos para el entrenamiento. Si hay una aplicación donde el tiempo no es importante este modelo es perfecto. Los modelos 128 y 256 no son recomendables utilizarlos, ambos modelos tienen mucha información que disminuye el rendimiento y no reduce el número de la época. Como

trabajo futuro es probar otra capa profunda como la técnica de RELU y DROPOUT que son especiales para el modelo con grandes imágenes. Ver figura 8.

**Figura 8. Estimación de la pose de un objeto tridimensional y su posición identificada.**

Cabe añadir finalmente que mediante la colaboración de la estudiante del CICS Santo Tomás, alumna Lesley Abigail Rivera Garduño del tercer semestre de la carrera en Optometría se logró la colaboración de personas con capacidades visuales disminuidas que sirvieron como evaluadoras del prototipo y que participaron voluntariamente. Ver figura 9.

**Figura 9. Prueba del prototipo en aproximación, evasión y uso de objetos.**

De acuerdo con las opiniones vertidas en un cuestionario se puede resumir que consideraron el prototipo muy apropiado en la detección a distancia de los objetos (sillas) del orden de 1 m. en promedio y que al estar en la proximidad les resulta más fácil su evasión o su aproximación en caso de requerir su uso. Asimismo, la respuesta audible del sistema que se ejecuta en tiempo real permitió al usuario su ubicación espacial en la habitación de prueba sin ningún problema y se apega a las recomendaciones emitidas por organismos como el IMSS en su manual de Normas para la accesibilidad de las personas con discapacidad. Finalmente se reconoce que el sistema propuesto resulta un apoyo aceptable al usuario.

**Bibliografía.**

[1] do Monte Lima, J.P.S., Simes, F.P.M., Uchiyama, H. et al. Depth assisted rectification for real-time object detection and pose estimation. Machine Vision and Applications (2016) 27: 193. doi:10.1007/s00138-015-0740-8

[2] E. Marchand, H. Uchiyama and F. Spindler, Pose Estimation for Augmented Reality: A Hands-On Survey, in IEEE Transactions on Visualization and Computer Graphics, vol. 22, no. 12, pp. 2633-2651, Dec. 1 2016. doi: 10.1109/TVCG.2015.2513408

[3] J. Shotton et al, Efficient Human Pose Estimation from Single Depth Images, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 12, pp. 2821-2840, Dec. 2013. doi: 10.1109/TPAMI.2012.241

[4] J. R. Rambach, A. Tewari, A. Pagani and D. Stricker, Learning to Fuse: A Deep Learning Approach to Visual-Inertial Camera Pose Estimation, 2016 IEEE International Symposium onMixed and Augmented Reality (ISMAR), Merida, 2016, pp. 71-76. doi: 10.1109/ISMAR.2016.19

[5] Keze Wang, Shengfu Zhai, Hui Cheng, Xiaodan Liang, and Liang Lin, Human Pose Estimation from Depth Images via Inference Embedded Multi-task Learning, Proc. of ACM International Conference on Multimedia (ACM MM) (Oral), 2016

[6] Javier Maldonado Romo, Mauricio Olgun-Carbajal, Israel Rivera Zarate, Raul Galvan, Analyzed Performance for a Chairs Classifier through Deep Learning, Research in Computing Science, an open access research journal on Computer science and computer engineering, Advances in Computer Science,pp. 149156

[7] Shotton, Jamie and Fitzgibbon, Andrew and and Blake, Andrew and Kipman, Alex and Finocchio, Mark and Moore, Richard and Sharp, Toby, Real-Time Human Pose Recognition in Parts from a Single Depth Image, CVPR, 2011

[8] M. Meilland, T. Drummond and A. I. Comport, A Unified Rolling Shutter and Motion Blur Model for 3D Visual Registration, 2013 IEEE International Conference on Computer Vision, Sydney, VIC, 2013, pp. 2016-2023. doi: 10.1109/ICCV.2013.252