Revista Electrónica Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional

ISSN 2007-1957

RECONOCIMIENTO DE OBJETOS PARA APOYO A PERSONAS INVIDENTES BASADO EN DEEP LEARNING

Israel Rivera Zárate
Instituto Politécnico Nacional-CIDETEC
irivera@ipn.mx

Miguel Hernández Bolaños
Instituto Politécnico Nacional-CIDETEC
mbolanos@ipn.mx

Jesús Pimentel Cruz
Instituto Politécnico Nacional-CIDETEC
jpimente@ipn.mx

Abstract

The present work shows the performance analysis of an object recognition system based on deep learning techniques that will be used in a mobile device. This proposal is considered based on the high performance offered by the new multi-core and graphic processing (GPU) architectures that are incorporated into today's mobile devices. It is intended to include this phase of object recognition in a wider system that supports blind people.

Keywords: Artificial Intelligence, Deep Learning, Convolutional Neuronal Network, Classifier.

Resumen

El presente trabajo muestra el análisis del desempeño de un sistema de reconocimiento de objetos basado en técnicas de aprendizaje profundo (deep learning) que será empleado en un dispositivo móvil. Esta propuesta se considera con base en el alto desempeño que ofrecen las nuevas arquitecturas multinúcleo y de procesamiento gráfico (GPU's) que vienen incorporadas en los dispositivos móviles de hoy día. Se pretende incluir esta fase de reconocimiento de objetos en un sistema más amplio que sirva de apoyo a personas invidentes.

Palabras clave: Inteligencia Artificial, Aprendizaje Profundo, Red Neuronal Convolucional, Clasificador.

La Organización Mundial de la Salud (OMS) señala que cerca de mil millones de personas en el mundo sufren alguna discapacidad visual, mental, motriz, auditiva o de lenguaje. De acuerdo con información del Instituto Nacional de Estadística, Geografía e Informática (INEGI), en México, cifras oficiales indican que un millón 795 mil personas tiene alguna discapacidad donde el 26 por ciento es de tipo visual (INEGI 2014).

ISSN 2007-1957

Por lo tanto, este trabajo consiste en un sistema de apoyo para personas invidentes en el que se interactúa con objetos reales en los que se utilicen elementos ultrasónicos para la detección pero no se requiere el empleo de etiquetas para la identificación. En su lugar se propuso un sistema basado en un dispositivo móvil que permita el reconocimiento de objetos mediante la captura y procesamiento de imágenes sin importar su perspectiva empleando técnicas de aprendizaje profundo (*deep learning*).

Para desarrollar este trabajo se utilizó *Torch7*, una herramienta para el aprendizaje profundo desarrollada por la Universidad de Toronto. Esta herramienta tiene un gran rendimiento cuando se ejecuta conjuntamente en CPU y GPU. El clasificador propuesto tiene cuatro clases donde tres clases son sillas de diferente tipo y la última clase corresponde a la clase de nada o algo que simplemente no corresponde a una silla.

Desarrollo

El desarrollo se divide en cinco secciones que son las siguientes: datos de carga, modelo, función de pérdida, tren y pruebas [1]. Los datos de carga tienen que leer todas las imágenes cuando éstas tienen dimensiones de tamaño de 32 píxeles en anchura y altura. Esta sección intercambia nuestro conjunto de datos con cuatro clases. El archivo de modelo contiene las capas numéricas del modelo original y contiene tres capas. La primera capa es la convolución espacial con función de transferencia Tanh y Maxpooling. La segunda capa también es la convolución espacial con la misma función de transferencia Tanh y su respectivo Maxpool. La última capa realiza una clasificación lineal para determinar el mejor resultado [2].

Para este desarrollo la GPU empleada es Nvidia GTX860M; tiene 640 núcleos CUDA y 2 GB en memoria RAM. CUDA es la tecnología para ejecutar tareas paralelas en Nvidia GPU[3]. Torch7 está optimizado para el uso de CUDA. Este software sólo está disponible en Ubuntu 14.04 y versiones posteriores. No hay soportes para operar otros sistemas, pero es posible ejecutar Torch7 en MacOS. Torch7 utiliza lenguaje Lua que es una extensión de C++. Lua se libera para tareas paralelas como el proceso *Multicore* utilizando *OpenMP* [4].

En la figura 1 se muestran algunas imágenes para cada clase que representa nuestro desarrollo (experimento).

Figura 1. Conjunto de datos para la formación de 32 píxeles en los que aparecen dos muestras para cada clase respectiva.

La figura 1 muestra las cuatro clases empleadas: la primera pareja muestra silla de oficina color azul, la siguiente pareja es silla común color negro, después silla de oficina color negro y el último puede ser cualquier cosa. Cada clase está representada por 180 imágenes. El entrenamiento es supervisado y cada imagen tiene una etiqueta que indica su clase.

Lo más importante en este desarrollo es medir los diferentes tiempos en que se efectúa el entrenamiento. Posteriormente se cambian las dimensiones del tamaño de imagen. Las dimensiones normales son 32 píxeles, a continuación, se cambian las dimensiones a 16 píxeles, después a 8 píxeles, incluyendo también 64, 128 y 256 píxeles, recordando que el cambio de tamaño es en el ancho y la altura. El objetivo es comprobar el modelo que termina primero y también medir el número de épocas necesarias para obtener una buena precisión. Las épocas son el número de veces que requiere el entrenamiento para obtener el valor de precisión buscado [5].

ISSN 2007-1957

Pruebas y resultados

La figura 2 muestra el comportamiento del modelo cuando las imágenes de entrada son de 32 píxeles por lado con los canales rojo, verde y azul. La imagen entra en el modelo y se le aplican tres capas. La primera capa consiste en la convolución espacial con 16 características con *Kernel* de tamaño tres. Posteriormente se aplica la función transferencia *Tanh* y por último el *Maxpool* con *Kernel* de tamaño dos. El resultado es la entrada de la siguiente capa que también tiene convolución espacial pero con 256 características con *Kernel* de tamaño tres. Finalmente se continúa con otra etapa de *Tanh* y *Maxpool*.

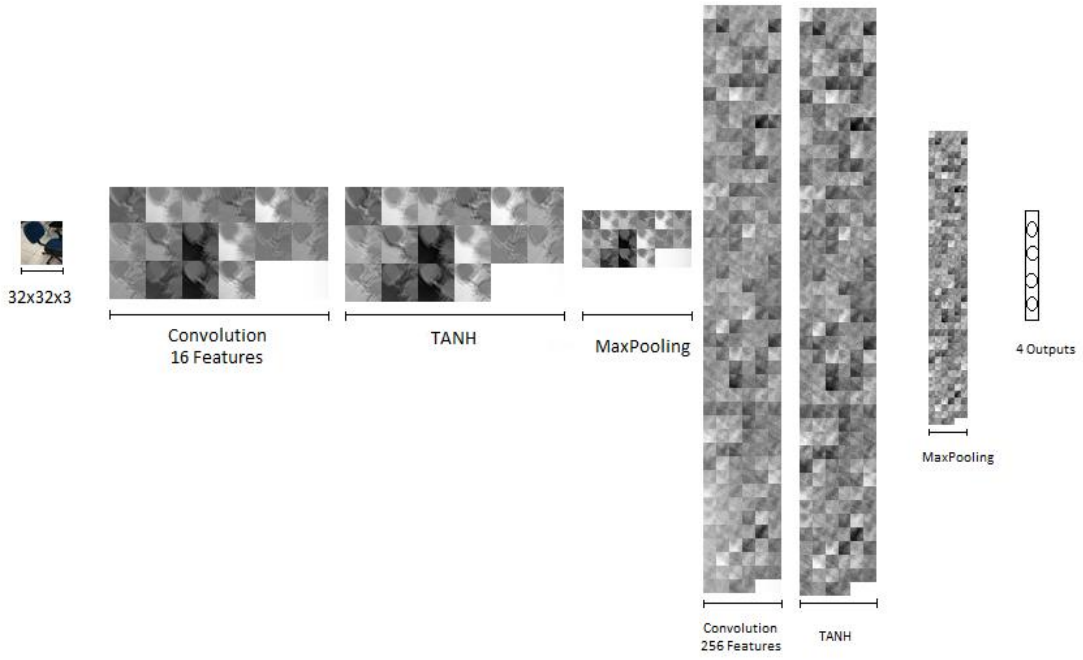

Figura 2. Descripción gráfica modelo 32, se observan modificaciones en cada una de las capas.

La tabla 1 especifica las diferentes características de los modelos considerados, donde el modelo con 16 píxeles la capa es menor que el modelo 32 píxeles.

Los gráficos de la figura 3 muestran 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.

La tabla 2 indica que el modelo 32 píxeles es la mejor opción con menos número en épocas y tiempo, otro punto importante es la

ISSN 2007-1957

Tabla 1. Detalles para cada modelo, donde se muestran sus características respectivas

Layer	Feature	Model 8	Model 16	Model 32	Model 64	Model 128	Model 256
1	Network	Conv.	Conv.	Conv.	Conv.	Conv.	Conv.
	Output	16 feat.	16 feat.	16 feat.	32 feat.	64 feat.	128 feat.
	Kernel / classifier	5x5	5x5	5x5	5x5	5x5	5x5
	Function	Tanh	Tanh	Tanh	Tanh	Tanh	Tanh
	Out layer	Max pool 2x2	Max Pool 2x2	Ma x pool 2x2	Max pool 2x2	Max pool 2x2	Max pool 2x2
2	Network	reshape	Conv.	Conv.	Conv.	Conv.	Conv.
	Output	16x2x2	256	256	512	1024	2048
	Kernel / classifier	Linear 128	5x5	5x5	5x5	5x5	5x5
	Function	Tanh	Tanh	Tanh	Tanh	Tanh	Tanh
	Out layer	Linear 4	Max Pool 2x2	Max Pool 2x2	Max Pool 2x2	Max Pool 2x2	Max Pool 2x2
3	Network		Reshape	Reshape	Conv.	Conv.	Conv.
	Output		256x5x5	256x5x5	256	512	1024
	Kernel / classifier		Linear 128	Linear 128	5x5	5x5	5x5
	Function		Tanh	Tanh	Tanh	Tanh	Tanh
	Out layer		Linear 4	Linear 4	Max Pool 2x2	Max Pool 2x2	Max Pool 2x2
4	Network				Reshape	Conv.	Conv.
	Output				256x4x4	256	512
	Kernel / classifier				Linear 128	5x5	5x5
	Function				Tanh	Tanh	Tanh
	Out layer				Linear 4	Max Pool 2x2	Max Pool 2x2
5	Network					Reshape	Conv.
	Output					256x5x5	256
	Kernel / classifier					Linear 128	5x5
	Function					Tanh	Tanh
	Out layer					Linear 4	Max Pool 2x2
6	Network						Reshape
	Output						256x5x5
	Kernel / classifier						Linear 128
	Function						Tanh
	Out layer						Linear 4

ISSN 2007-1957

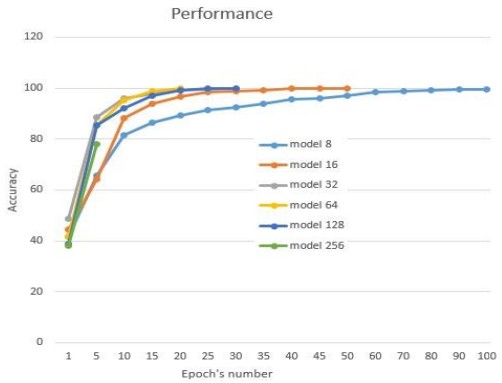

Fig. 3. Precisión alcanzada en 5 épocas

baja potencia en GPU y el tiempo de respuesta rápido. El modelo 16 píxeles es excelente, este modelo necesita más épocas pero menos tiempo y demanda en GPU, este punto es relevante para sistemas embebidos que no pueden utilizar mucha energía como dispositivo móvil, pero el móvil requiere GPU Nvidia, no hay muchos dispositivos con estas GPU en el mercado, o sistemas en tiempo real es una buena idea utilizar este modelo. El modelo 8 requiere muchas épocas y tiempo. De lo contrario, no se recomienda utilizar la imagen muy pequeña porque no tiene mucha información y puede tener un error considerable. Cuando cambia las dimensiones superiores a 64 píxeles los datos crecen y la información también, y se ve en dicha tabla cuando muestra que en 17 épocas para obtener 100% de precisión porque hay más datos, pero el tiempo no es bueno en comparación con el modelo 32. En aplicaciones en las que el tiempo de respuesta y la precisión no son importantes. La última observación es que los modelos 128 y 256 no son recomendables para usar este conjunto de datos con sus modelos porque los datos y las operaciones crecen. Estas capas profundas requieren más poder y mucho tiempo; el número de épocas no es menor que el modelo de 64 píxeles. El tiempo requerido es más largo.

Tabla 2. Detalles sobre el comportamiento y el rendimiento de cada modelo

Feature	Model 8	Model 16	Model 32	Model 64	Model 128	Model 256
Epoch	117	41	30	17	23	3
GPU RAM	440 MB	434 MB	445 MB	488 MB	651 MB	730 MB
GPU Uses (%)	43	34	83	91	99	99
Time (Sec)	56	30	40	165	1800	2700+

Conclusiones

En este artículo se describe 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 un error significativo. Los modelos 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 épocas. Como trabajo a futuro es probar otra capa profunda como la técnica de RELU y DROPOUT que son especiales para el modelo con grandes imágenes.

ISSN 2007-1957

Referencias

[1] Krizhevsky, Alex. (2009). Convolutional Deep Belief Networks on CIFAR-10.

[2] Krizhevsky, Alex. (2009). Convolutional Neural Networks for Object Classification in CUDA.

[3] Nvidia GTX860M. Specification, www.geforce.com/hardware/notebook-gpus/geforce-gtx-860m.

[4] Torch7, www.torch.ch, consultado el 15 de junio de 2017.

[5] Krizhevsky, Alex. (2011). Sutskever Ilya: ImageNet Classification with Deep Convolutional Neural Networks.