Métodos de optimización de la gradiente de descenso en una red neuronal convolucional

Víctor Jesús Sotelo Chico[1]

[1]Universidad Nacional de Ingeniería

Seminario de Tesis I

Contenido

Introducción

En la actualidad es indispensable emplear mucho tiempo en el entrenamiento de redes neuronales profundas, por lo que surge la necesidad de encontrar métodos que aceleren este proceso.

Objetivos

- Entender las ventajas y desventajas de distintos métodos de optimización de la gradiente de descenso.
- Obtener la capacidad de discriminar entre distintos métodos de optimización.
- Lograr un mejor entendimiento de las redes neuronales profundas.

Aprendizaje Automático

Consiste en lograr que las computadoras sean capaces de realizar una predicción o tomar decisiones sin haber estado programada explícitamente para realizar esta tarea.
El Aprendizaje Automático puede ser divido de la siguiente forma :

- Aprendizaje Supervisado
- Aprendizaje No Supervisado
- Aprendizaje por Refuerzo

Redes Neuronales Artificiales

Estas redes toman como inspiración la arquitectura del cerebro para la construcción de sistemas inteligente. Actualmente son la base para el desarrollo de la inteligencia artificial.

Comparación neuronas biológicas y artificiales

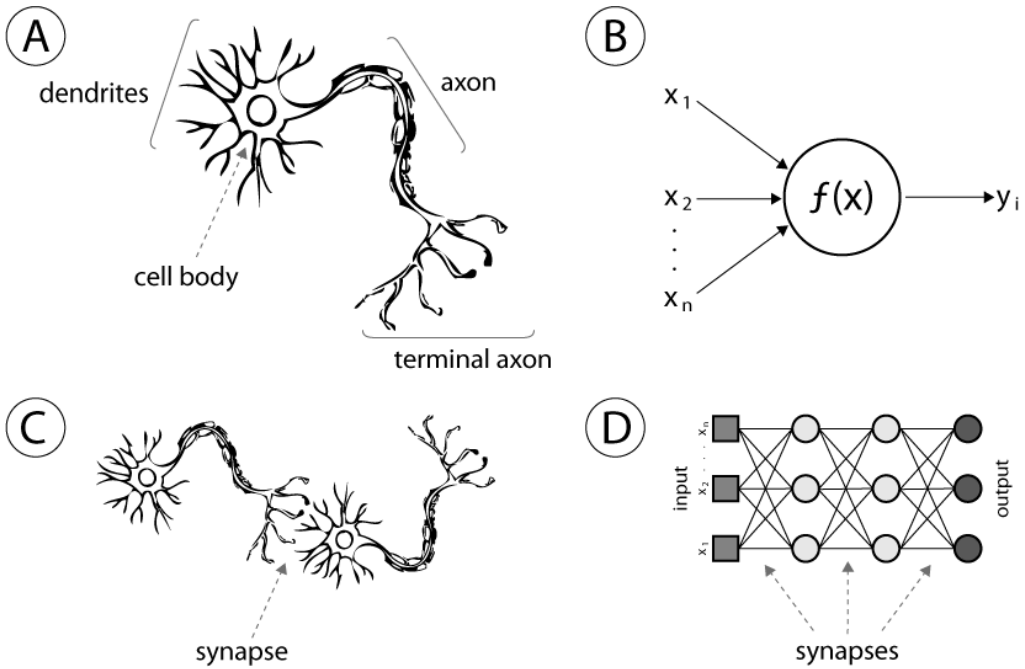

FIGURE – Redes neuronales biológicas y artificiales

Redes neuronales Prealimentadas

Es un tipo de red neuronal más simple que existe. Esta red puede clasificarse en :

- Perceptron simple
- Perceptron Multicapas
- Redes neuronales convolucionales

Esquema Redes neuronales Prealimentadas

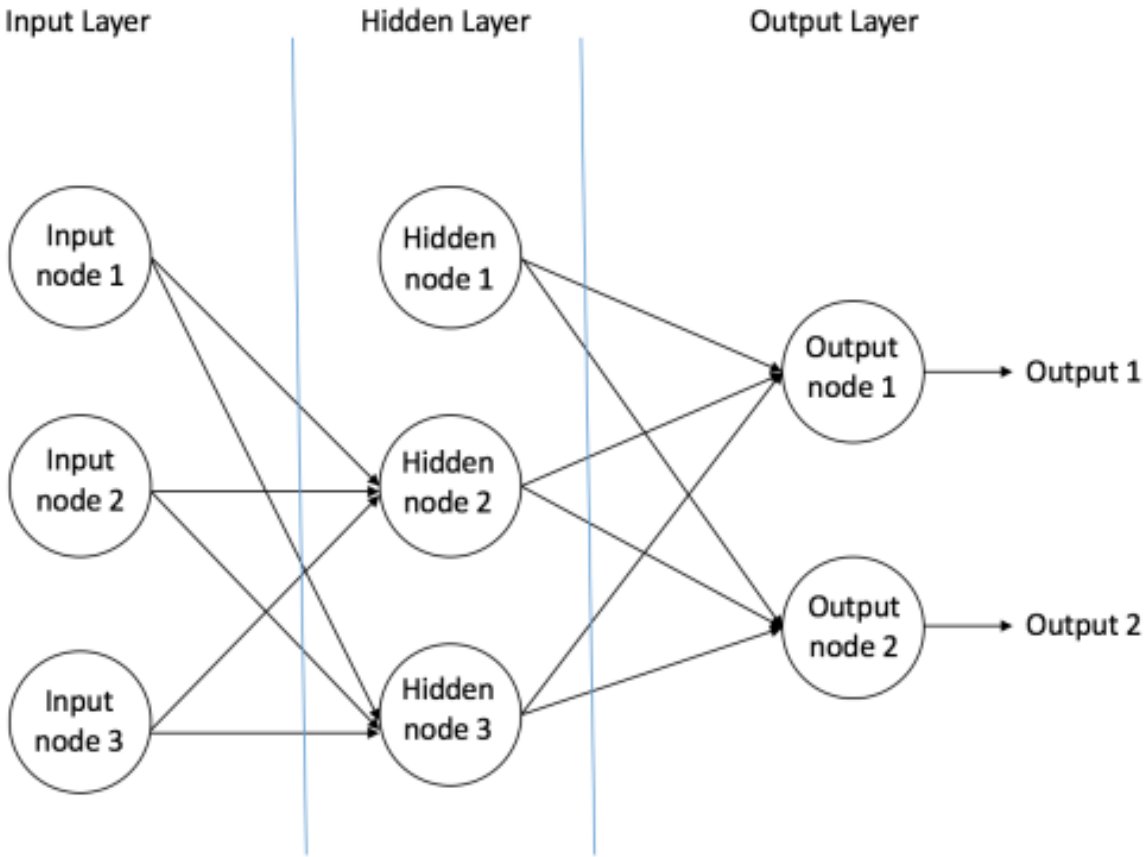

FIGURE – Esquema de Redes Neuronales Prealimentadas

Back Propagation

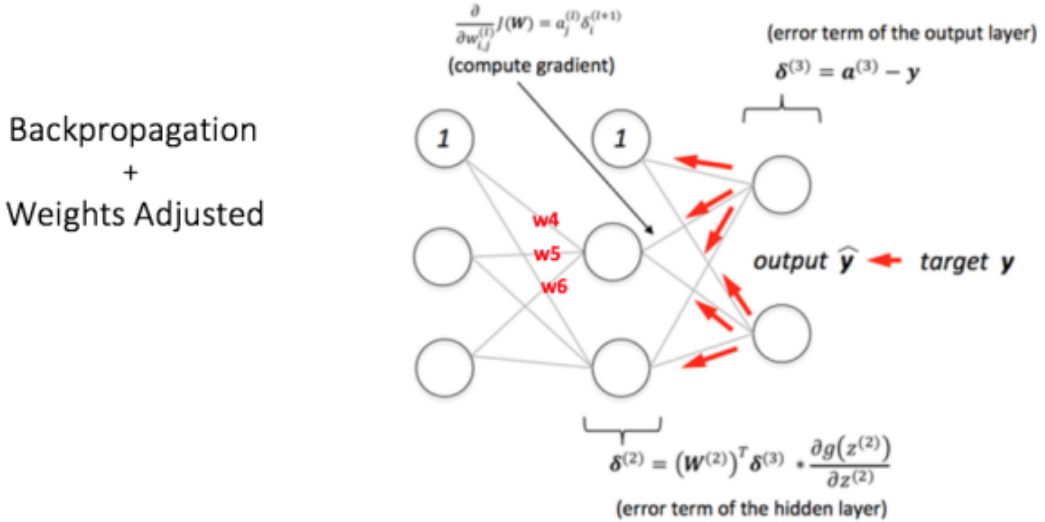

FIGURE – Propagación hacia atrás

Redes Neuronales Convolucionales(CNN)

Las CNN son un tipo de redes neuronales especiales para procesar datos como imágenes. La primera CNN fue creada por Yann LeCun.

Capas de una red neuronal convolucional

- Input Layer
- Convolutional Layer
- Pooling Layer
- Fully Conected Layer
- Output Layer

Input Layer

Se encarga de recibir la información de entrada.

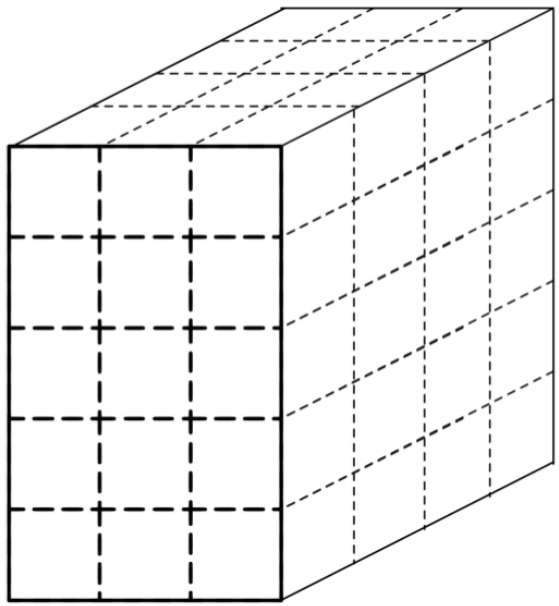

FIGURE – Estructura de la imagen de entrada

Convolutional Layer

La capa calculará el producto punto entre la región de las neuronas de la capa de entrada y los pesos a los que están colocados localmente en la capa de salida. Posee los siguientes hiperparámetros :

- Filter size.
- Output depth.
- Stride.
- Zero padding.

Convolutional Layer

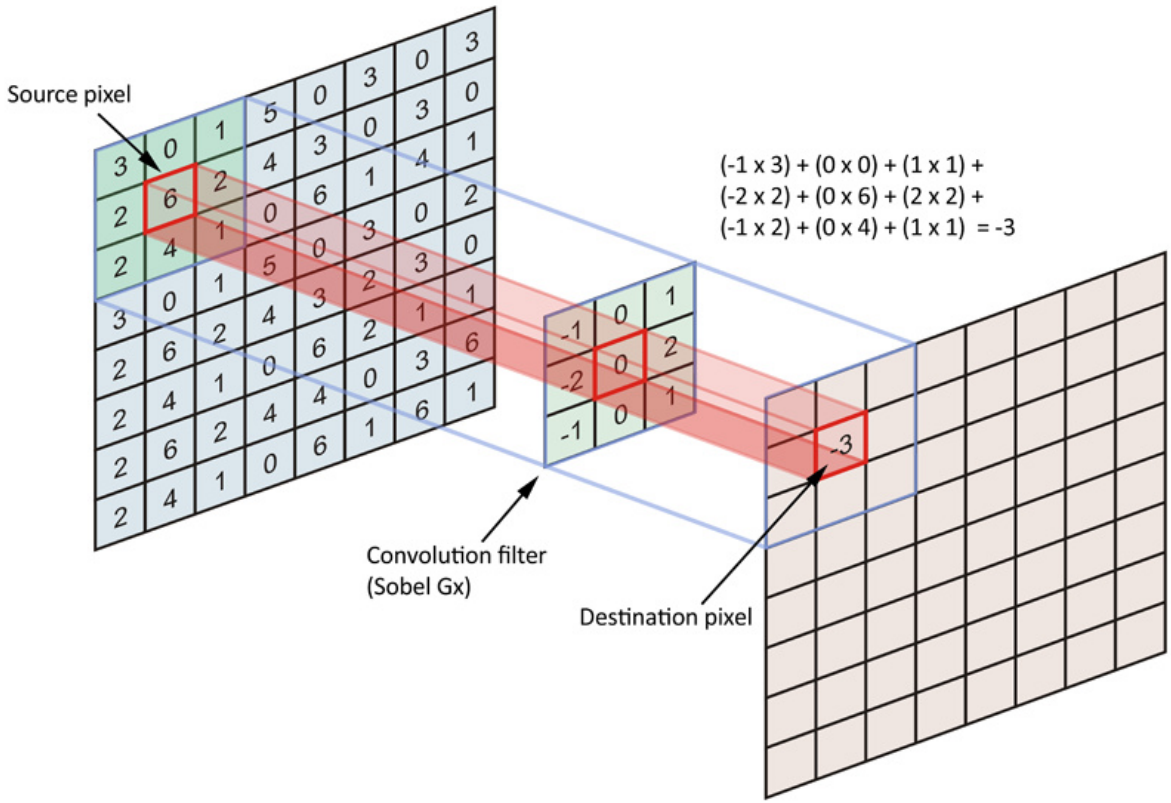

FIGURE – Operacion de convolución

Pooling Layer

Se encarga de reducir el tamaño espacial(ancho,alto) de los datos de representación.

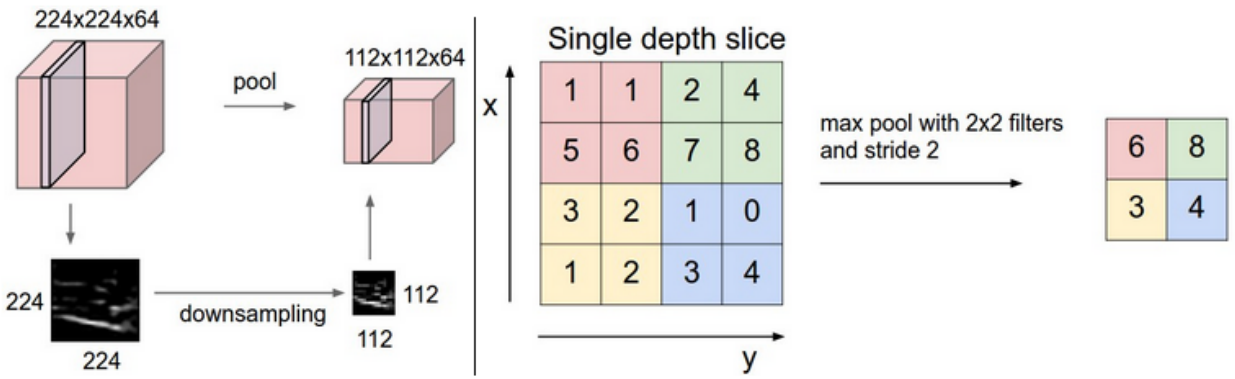

FIGURE – Operación Max pool

Fully Conected Layer

Esta capa será la encargada de reconocer a que clase pertenece una imagen de prueba de acuerdo a su puntaje o probabilidad.

Gradiente de Descenso

La gradiente de descenso es una forma de minimizar la función de costo $J(\theta)$ parametrizada por los parámetros $\theta \in \Re^d$.

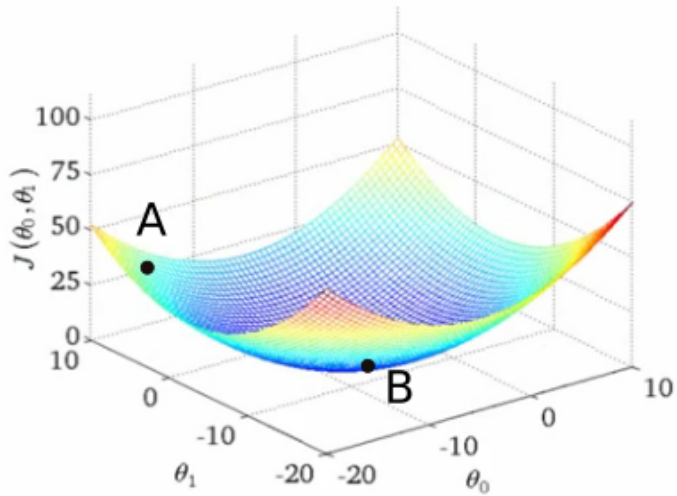

FIGURE – Gradiente de descenso

Variantes de la Gradiente de Descenso

Existen 3 variantes de la gradiente de descenso :

- Batch gradient descent
- Stochastic gradient descent
- Mini-batch gradient descent

Métodos para optimizar la gradiente de descenso

- Momentum
- Nesterov Momentum
- Adagrad
- RMSprop
- Adam

Momentum

El momentum es un método que ayuda a la SGD a acelerar en la dirección correcta, mientras evitas las oscilaciones

$$\begin{aligned} \nu_t &= \gamma \nu_{t-1} + \eta \nabla_\theta J(\theta) \\ \theta &= \theta - \nu_t \end{aligned} \tag{1}$$

Nesterov

Esta técnica es una variante de momentum donde se usa el término $\gamma\nu_{t-1}$ para mover los parámetros de θ.

$$\begin{aligned} \nu_t &= \gamma\nu_{t-1} + \eta\nabla_\theta J(\theta - \gamma\nu_{t-1}) \\ \theta &= \theta - \nu_t \end{aligned} \tag{2}$$

Adagrad

Es un algoritmo optimización basado en la gradiente de descenso, del tipo adaptativo.

$$\begin{aligned} g_{t,i} &= \nabla_\theta J(\theta_{t,i}) \\ \theta_{t+1,i} &= \theta_{t,i} - \eta \cdot g_{t,i} \end{aligned} \tag{3}$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \tag{4}$$

- $G_{t,ii}$: suma de los cuadrados de las gradientes pasadas con respecto a θ_i
- ϵ es un término pequeño para evitar la división por 0.

RMSprop

Método del tipo adaptativo que se desarrollo con el objetivo resolver el problema de disminuir radicalmente la tasa de aprendizaje en Adagrad.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{5}$$

Adam

Adaptative moment estimation o Adam, calcula una tasa de aprendizaje adaptativo para cada parámetro.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{6}$$

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{7}$$

$$\theta_{t+1} = \theta_{t+1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{8}$$

Resultados CIFAR-10

Método	Precisión(%)	Tiempo de ejecución(s)
Momemtum	53.04	303.07
Nesterov	54.09	305.59
Adagrad	44.42	310.20
RMSprop	68.91	322.20
Adam	68.02	316.51

FIGURE – Precisión 50000 epochs

Resultados CIFAR-10 - Precisión

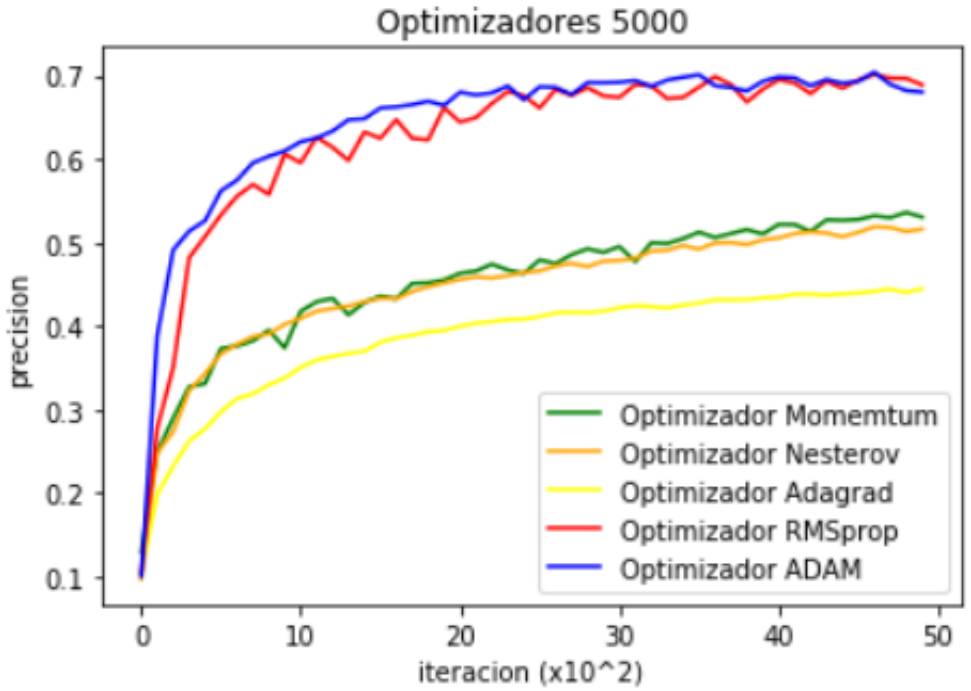

FIGURE – Comparación de precisión para 5000 epochs

Errores Función de Costo CIFAR-10

Método	Error en función de costo
Momemtum	1.26
Nesterov	1.38
Adagrad	1.73
RMSprop	0.19
Adam	0.30

FIGURE – Comparación de errores función de costo 5000 epochs

Errores Función de Costo CIFAR-10

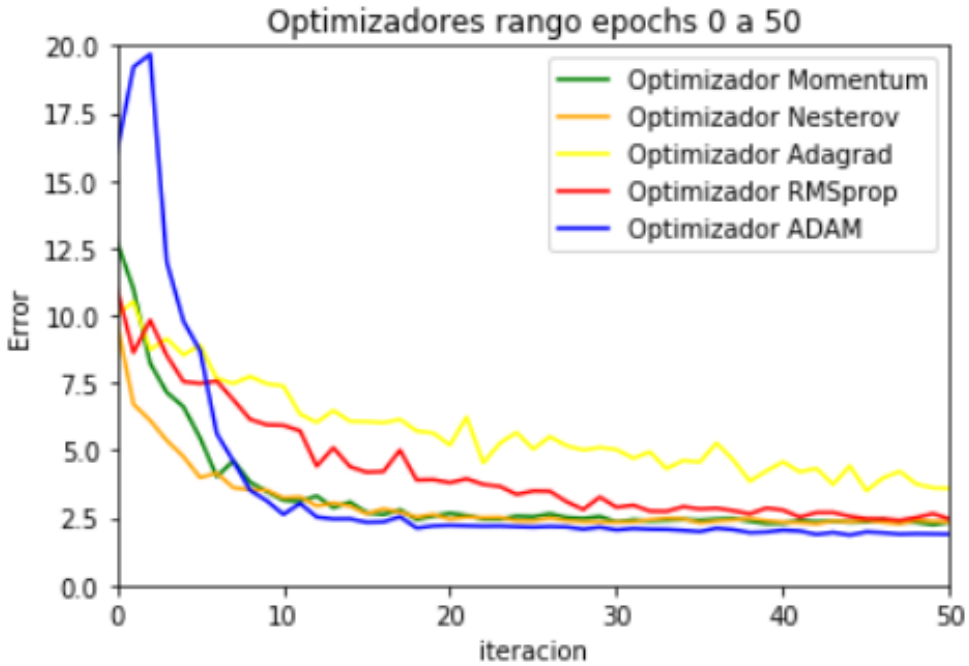

FIGURE – Comportamiento de errores en la función de costo 5000 epochs

Resultados CIFAR-10 - Precisión

Método	Precisión(%)	Tiempo de ejecución(s)
Momemtum	60.80	585.41
Nesterov	62.37	605.19
Adagrad	47.62	598.54
RMSprop	70.31	621.17
Adam	70.42	622.18

FIGURE – Precisión 100000 epochs

Resultados CIFAR-10 - Precisión

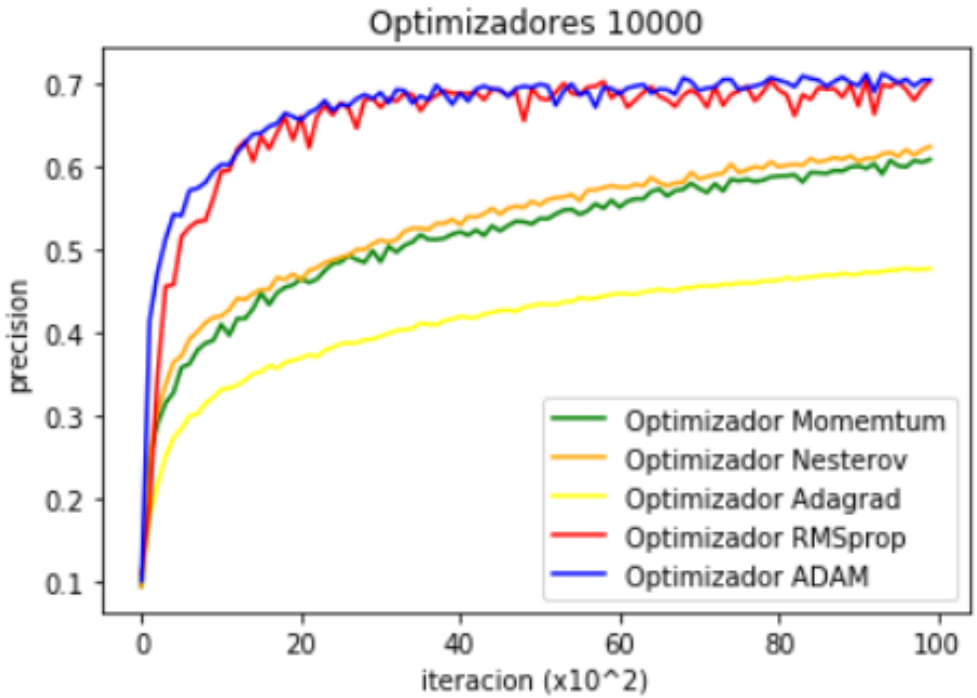

FIGURE – Comparación de precisión para 10000 epochs

Errores Función de Costo CIFAR-10

Método	Error en función de costo
Momemtum	1.10
Nesterov	1.08
Adagrad	1.64
RMSprop	0.19
Adam	0.14

FIGURE – Comparación de errores función de costo 10000 epochs

Errores Función de Costo CIFAR-10

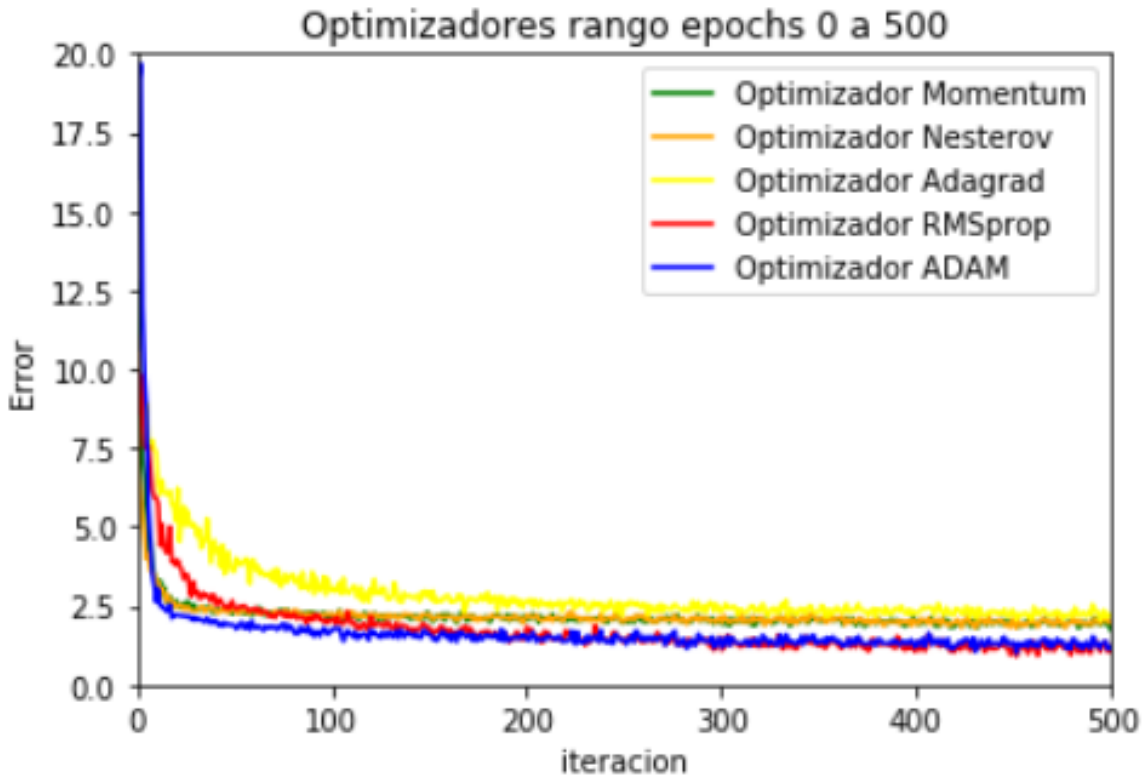

FIGURE – Comparación de las errores rango 0-500

Resultados CIFAR-100 - Precisión

Método	Precisión(%)	Tiempo de ejecución(s)
Momemtum	17.33	597.75
Nesterov	17.33	595.53
Adagrad	7.73	628.75
RMSprop	36.66	622.50
Adam	35,13	624.64

FIGURE – Precisión 100000 epochs

Resultados CIFAR-100 - Precisión

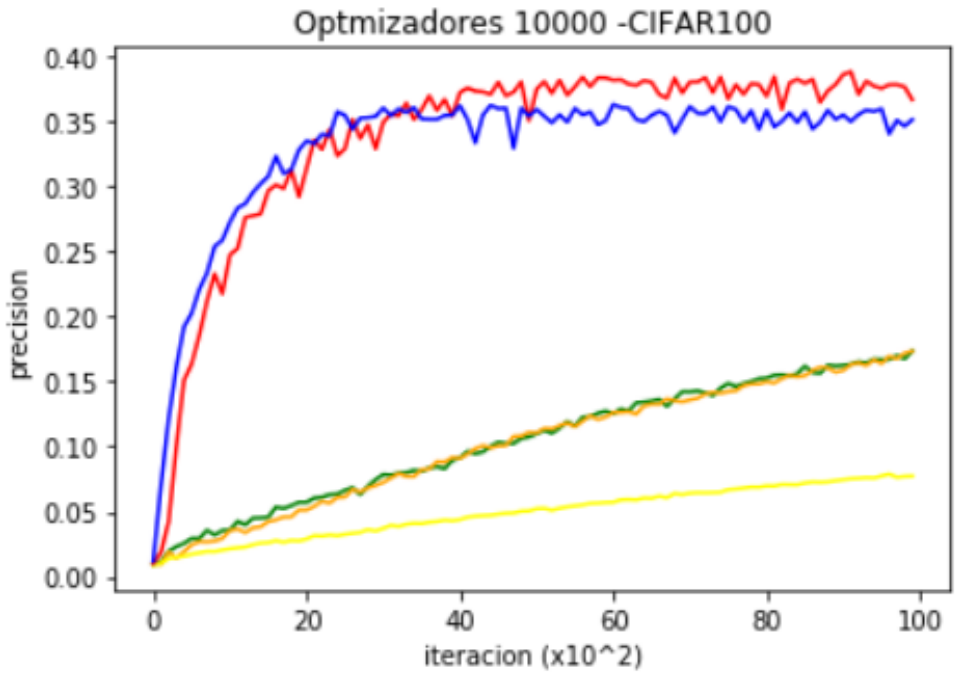

FIGURE – Comparación de precisión para 10000 epochs

Errores Función de Costo CIFAR-100

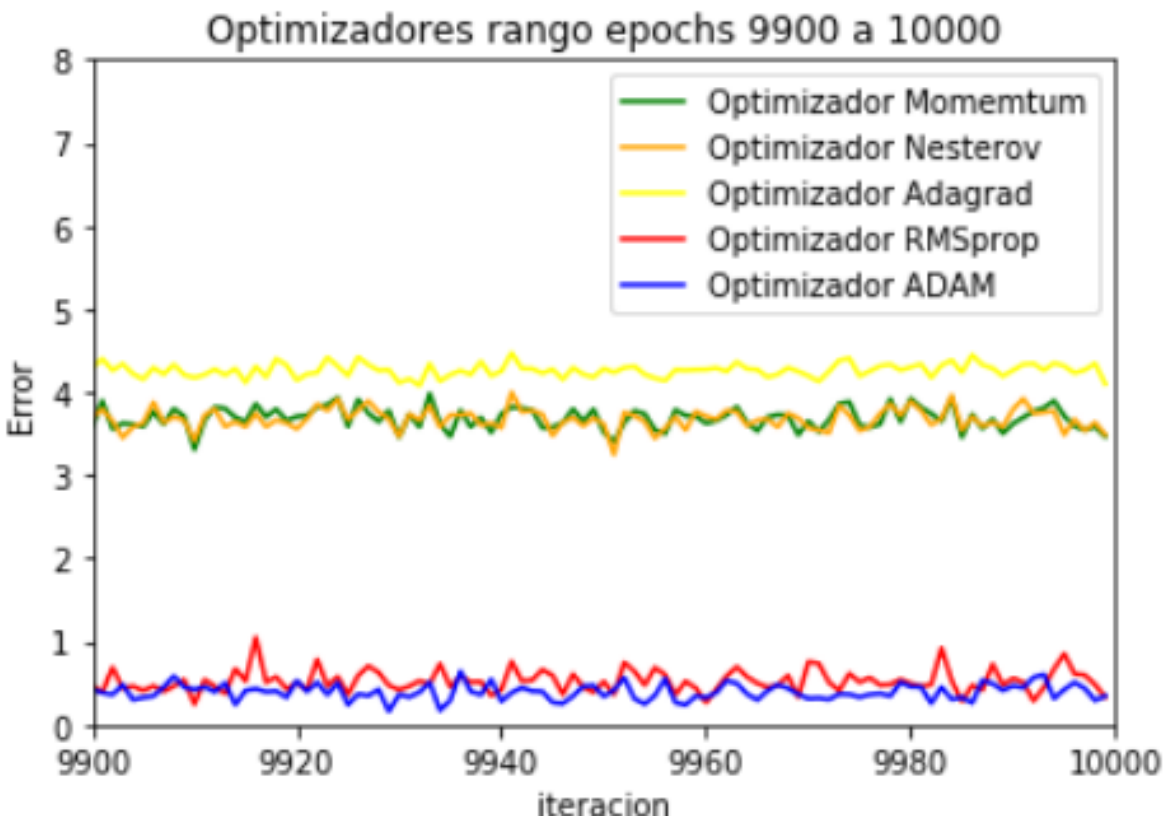

FIGURE – Comparación los errores rango 9900-10000

Conclusiones

- Los métodos de optimización Adam y RMSprop obtuvieron los mejores resultados de precisión en ambas pruebas.
- A pesar de que el método de optimización Adam fue propuesto a partir del RMSprop. Adam fue superado en algunas de pruebas realizadas.
- Adam es el método que tiene un decaimiento más acelerado al calcular el error en la función de costo cross-entropy.

Conclusiones

- Entre los métodos adaptativos Adam, RMSprop y Adagrad . Solo este último obtuvo los peores resultados, esto se debió a su dificultad de trabajar con la suma de las gradientes al cuadrado lo cual poco a poco redujo su tasa de aprendizaje.
- El RMSprop como una mejora del Adagrad, obtuvó mejores resultados que este último. Esto debido a que RMSprop trabaja con el promedio de la raíz de la gradiente anterior y tasas de decaimiento para controlar el problema de la disminución de la tasa de aprendizaje del método Adagrad.

Trabajos Futuro

- Correcto diseño de una red neuronal convolucional.
- Obtener resultados con distintos hardwares.
- Realizar una implementación más interactiva.