Trabajo práctico 1: Optimización y Redes Neuronales

Juan José Cordero Gómez
Escuela de computación
Instituto Tecnológico de Costa Rica

Luis Diego Hidalgo Blanco
Escuela de computación
Instituto Tecnológico de Costa Rica

Ricardo Sánchez Alpizar
Escuela de computación
Instituto Tecnológico de Costa Rica

3. (40 PUNTOS) REDES CONVOLUCIONALES PARA DETECCIÓN DE GLAUCOMA EN IMÁGENES DE FONDO DE OJO

1. (20 puntos) Implemente el filtro de "Unsharp masking" para la mejora de las imágenes, según lo especificado en el material del curso.

a) Compruebe y comente su uso para las imágenes de fondo de ojo, mostrando los resultados. Use al menos dos valores distintos de la ganancia λ.

: El filtro de unsharp masking, realza los bordes de las imágenes y ayuda en los modelos de clasificación de imágenes. Para aplicar este filtro, se emplea la siguiente formula:

$$G = U + (U * N) \times \lambda$$

Donde:

$U =$ Imagen original.

$\lambda =$ Coeficiente de ganancia.

$N =$ Núcleo Gaussiano de desenfoque (blur).

$(U * N)$ implica la convolución entre la imagen original y el filtro Gaussiano de desenfoque (blur).

El detalle del código empleado se encuentra en Jupyter notebook llamado: TP1 - UnsharpMask-AlexNet.ipynb.

Una vez implementado el código esto es un ejemplo de la salida obtenida al emplear diferentes coeficientes de ganancia:

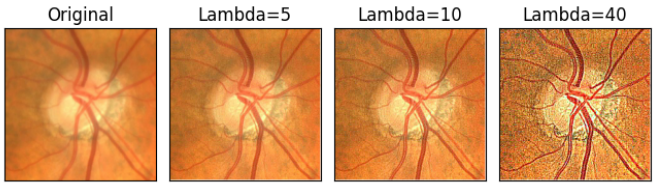

Fig. 1. Ejemplo de uso de filtro de Unsharp masking.

Como se puede observar diferentes valores del coeficiente de ganancia λ resultan en diferentes saturaciones, es necesario realizar el ajuste según el objetivo del modelo y acorde a las imágenes del data set.

Con $\lambda = 10$ parece ser un balance adecuado donde es evidente el realse de los bordes con respecto a la imagen original.

El $\lambda = 5$ parece tener un efecto poco perceptible que podría no aportar mucho al procesamiento del modelo.

Por otro lado el $\lambda = 40$ luce bastante saturado lo cual podría ser perjudicial para el modelo en términos del "ruido" que se observa en la figura 1.

2. (20 puntos) Implemente manualmente (especificando las capas) en pytorch la arquitectura de AlexNet. Entrene la red usando el conjunto de datos de imágenes de fondo de ojo. Calibre los hiper-parámetros necesarios para obtener los mejores resultados posibles y repórtelos. Ejecute el entrenamiento 10 veces por 15 épocas por corrida, y reporte la tasa de aciertos, falsos positivos y falsos negativos promedio y su desviación estándar para esas 10 corridas.

b) Compare los resultados respecto a lo obtenido con el perceptrón multi-capa y coméntelos.

: Se procede a realizar varias pruebas con el modelo con el fin de encontrar un learning rate (LR) que más lo beneficia y produce resultados más precisos, para lo cual se prueban dos veces los siguientes LR:

- 0.0
- 0.01
- 0.03
- 0.001
- 0.003

Los resultados de ambas pruebas son promediados para determinar el LR más efectivo para el modelo.

El detalle de cada entrenamiento y prueba del modelo se encuentra en el Jupyter notebook llamado: TP1 - UnsharpMask-AlexNet.ipynb.

A continuación se muestra el resumen de los resultados, todas las pruebas se realizaron con 210 imágenes, tabla: I:

De la tabla anterior se puede identificar el LR de 0.01 como uno de los que mejores resultados produce, por lo que se procede a hacer 10 repeticiones con dicho LR y a calcular

TABLE I
RESUMEN DE RESULTADOS CON DIFERENTES LR.

Iteración	LR	# Aciertos	# Fallos	Falsos positivos	Falsos negativos	Precisión (%)
1	0.1	178	32	29	3	84.7619
2	0.1	169	41	11	30	80.4762
3	0.01	204	6	5	1	97.1429
4	0.01	196	14	13	1	93.3333
5	0.03	192	18	11	7	91.4286
6	0.03	187	23	13	10	89.0476
7	0.001	197	13	10	3	93.8095
8	0.001	200	10	6	4	95.2381
9	0.003	199	11	8	3	94.7619
10	0.003	201	9	5	4	95.7143
Desviación estándar: 10.6212.						

la desviación estándar del mismo:

TABLE II
RESUMEN DE RESULTADOS CON UN LR CONSTANTE DURANTE 10 PRUEBAS.

Iteración	LR	# Aciertos	# Fallos	Falsos positivos	Falsos negativos	Precisión (%)
1	0.01	201	9	9	0	95.7143
2	0.01	200	10	8	2	95.2381
3	0.01	200	10	8	2	95.2381
4	0.01	199	11	6	5	94.7619
5	0.01	199	11	10	1	94.7619
6	0.01	1898	21	20	1	90
7	0.01	194	16	3	13	92.381
8	0.01	202	8	5	3	96.1905
9	0.01	197	13	3	10	93.8095
10	0.01	202	8	8	0	96.1905
Desviación estándar: 3.8483.						

Como es de esperar los resultados empleando el mismo LR son bastante consistentes, aún cuando han sido entrenados tomando diferentes muestras de imágenes y por cada corrida se realiza un reinicio de kernel.

TABLE III
COMPARACIÓN DE MLP Y ALEXNET.

Iteración	LR	MLP	AlexNet	MLP	AlexNet	MLP	AlexNet	MLP	AlexNet	MLP	AlexNet
		# Aciertos		# Fallos		Falsos positivos		Falsos negativos		Precisión (%)	
1	0.01		201		9		9		0		95.7143
2	0.01		200		10		8		2		95.2381
3	0.01		200		10		8		2		95.2381
4	0.01		199		11		6		5		94.7619
5	0.01		199		11		10		1		94.7619
6	0.01		1898		21		20		1		90
7	0.01		194		16		3		13		92.381
8	0.01		202		8		5		3		96.1905
9	0.01		197		13		3		10		93.8095
10	0.01		202		8		8		0		96.1905