UNIVERSIDAD DE GRANADA

Deep Learning para Multi-clasificación

Máster de Ingeniería Informática

Sistemas Inteligentes para la Gestión en la Empresa

Ramón Gago Carrera

Freddy Javier Frere Quintero

1 ¿Cuál es el objetivo?

Trabajar con:

Redes neuronales

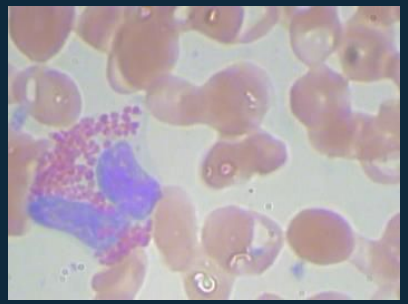
- Fundamentos teóricos
- Red empleada
- Discusión de resultados

2

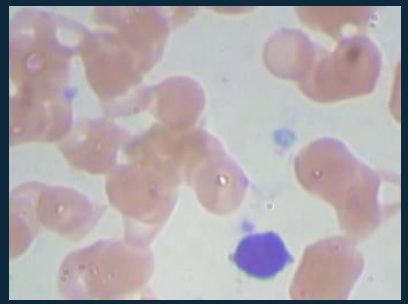
Fundamentos teóricos

La idea...

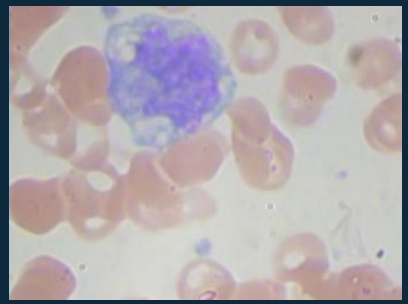
Sobre qué se utiliza:



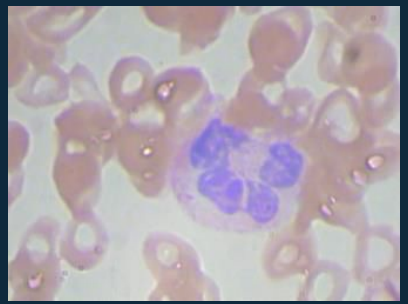
Eosinófilos



Linfocitos



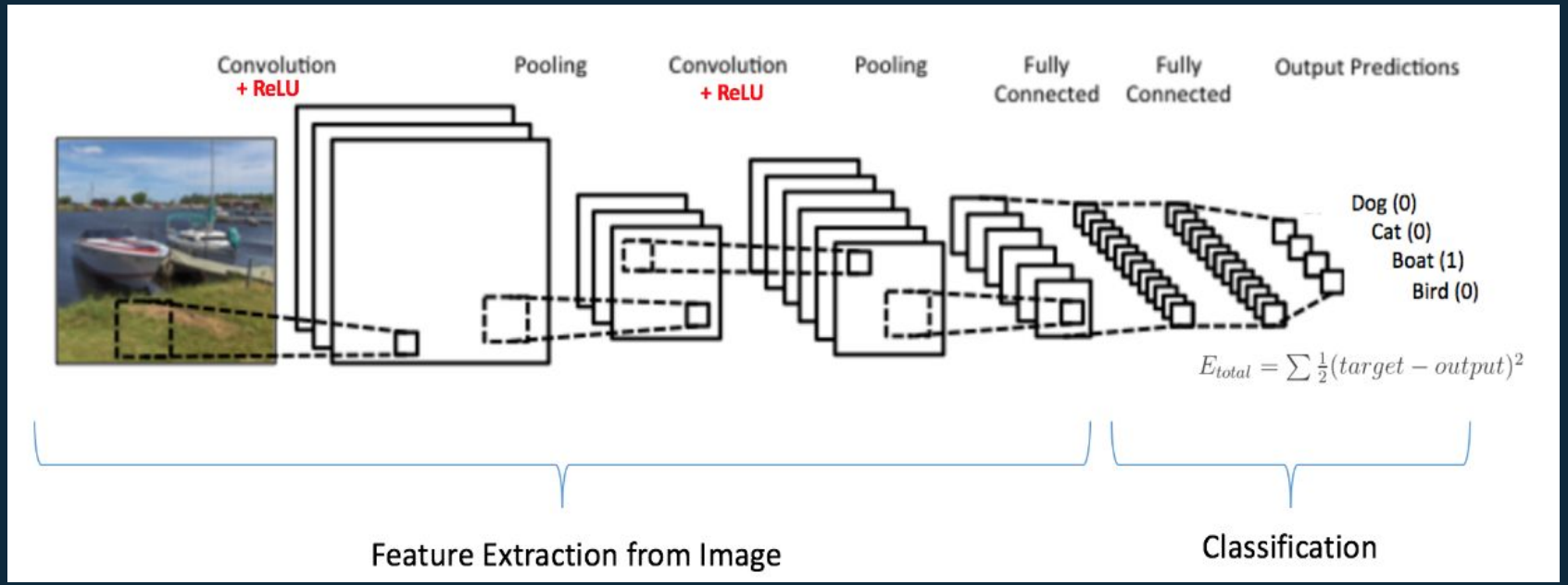
Monocitos



Neutrófilos

3 ¿Cómo se crea la red?

Un ejemplo de multi-clasificación



Parámetros de la red

Filters

Sobre las capas convolutivas 2D, como parámetro de agrupamiento

Units

Sobre las capas dense, como tamaño de salida

Dropout

Grado de omisión de neuronas ocultas para que no se adapten entre sí

Epochs

Cada iteración de la red por el lote de entradas en la que haya ajuste de variables

Steps per epoch

Número de lotes que deben producir los generadores para completar una época

Parámetros clave

Función de pérdida

- Binary_crossentropy
- Categorical_crossentropy
- Mean_squared_logarithmic_error

Algoritmos de optimización

- RMSProp
- Adadelta
- Adam
- SGD
- Adagrad

4

Construcción de la red

Modelo

3 Conv2D | 3 Activ(relu) | 3 MaxPooling2D | 1 Flatten | 2 Dense | 1 Dropout

```
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32,  kernel_size = c(3, 3), activation = "relu", input_shape = c(150,150, 3)) %>%  “Clasificacion y segmentación de imágenes” “A,A,P”
  layer_activation('relu') %>%  “Unidad lineal rectificada”
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%  “Reducir # de parametros”
  layer_conv_2d(filters = 32,  kernel_size = c(3, 3), activation = "relu") %>%
  layer_activation('relu') %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 64,  kernel_size = c(3, 3), activation = "relu") %>%
  layer_activation('relu') %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%   “Aplanar - 1 d”
  layer_dense(units = 256, activation = "relu") %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 4, activation = "softmax")  “Función Exp Normalizada - Redes Multiclase”
```

13 capas

Compilación - Función de Pérdida

MIN. LOSS		ACCURACY MÁX.
0.102	Binary_crossentropy	0.955
0.440	Categorical_crossentropy	0.873
0.014	Mean_squared_logarithmic_error	0.915

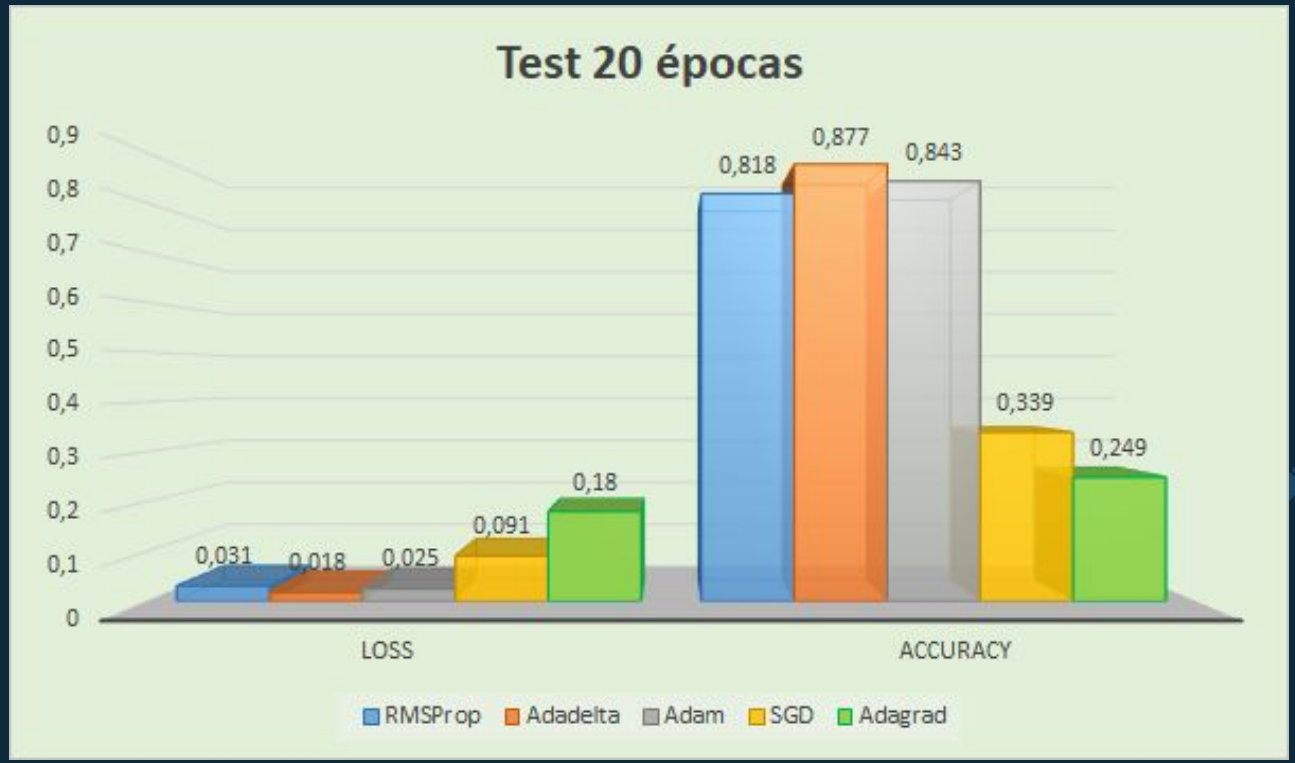
5

¿Qué resultados se han obtenido?

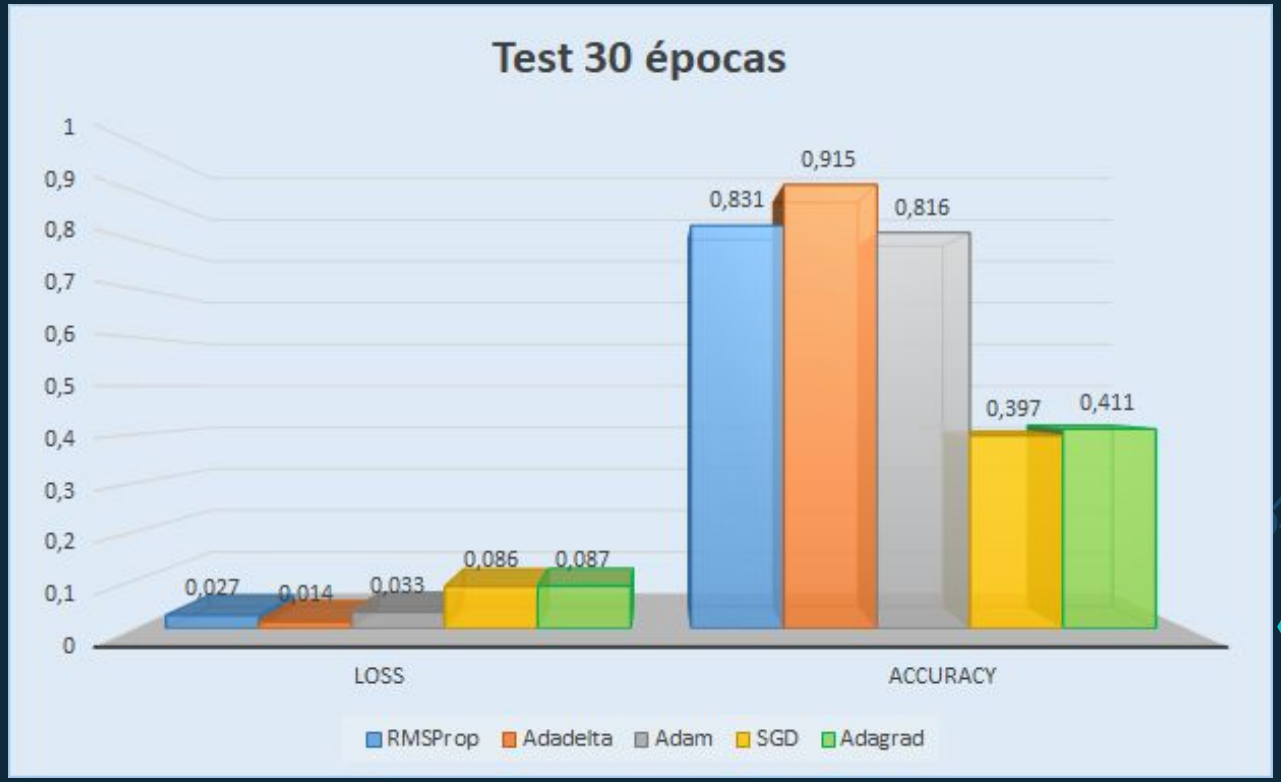
Algoritmos de optimización

Algoritmos de optimización

Algoritmos de optimización



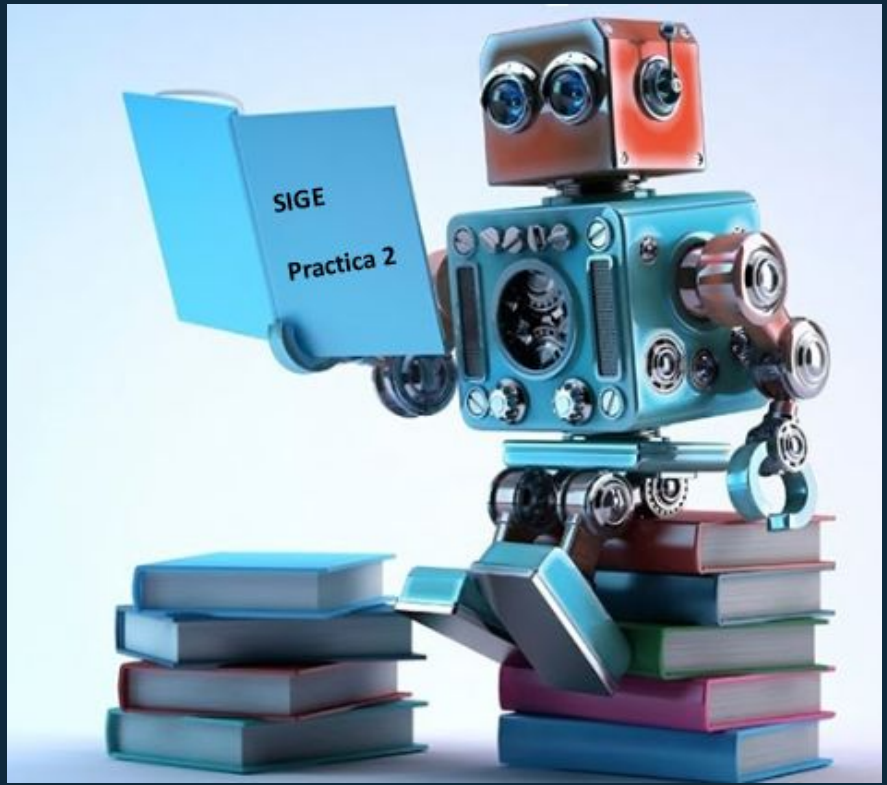
Algoritmos de optimización



6

¿Qué conclusiones se pueden extraer?

Conclusiones



¡Gracias!

¿Alguna pregunta?