CLASIFICACIÓN DE IMÁGENES DE RETINOPATÍA DIABÉTICA DE ACUERDO A SU GRAVEDAD

JESSICA FERNANDA PEDRAZA CADENA

2151853

Universidad Industrial de Santander

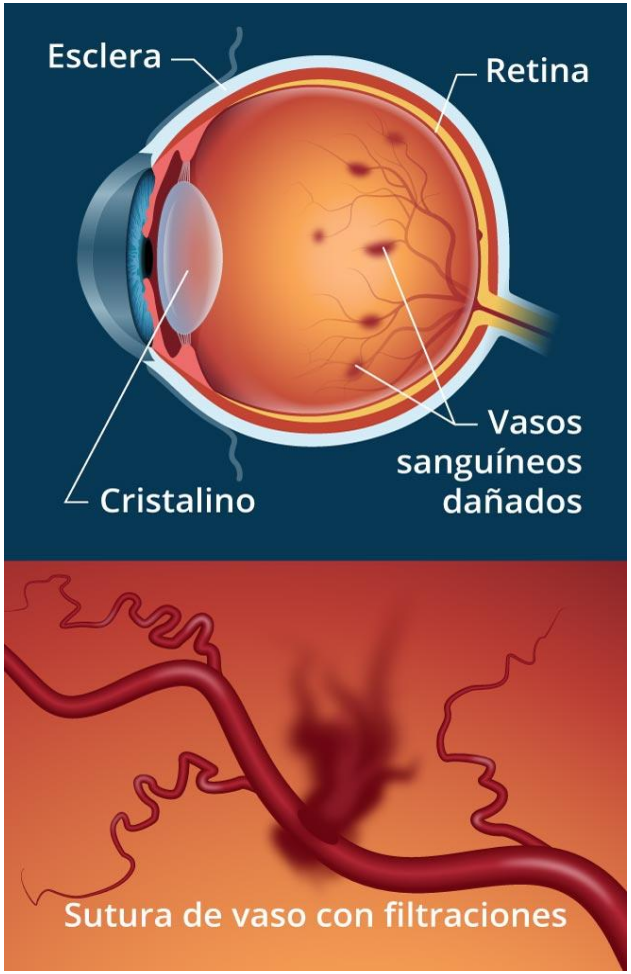
DESCRIPCIÓN

A partir de un dataset que contiene imágenes de escaneo de retina con filtro gaussiano de tamaño 224 x 224 se pretende entrenar una red neuronal para que aprenda a clasificar por sí sola si una persona tiene retinopatía diabética y la gravedad o la etapa en la que se encuentra. Además de esto se compara el accuracy de dicha red con otros métodos de clasificación.

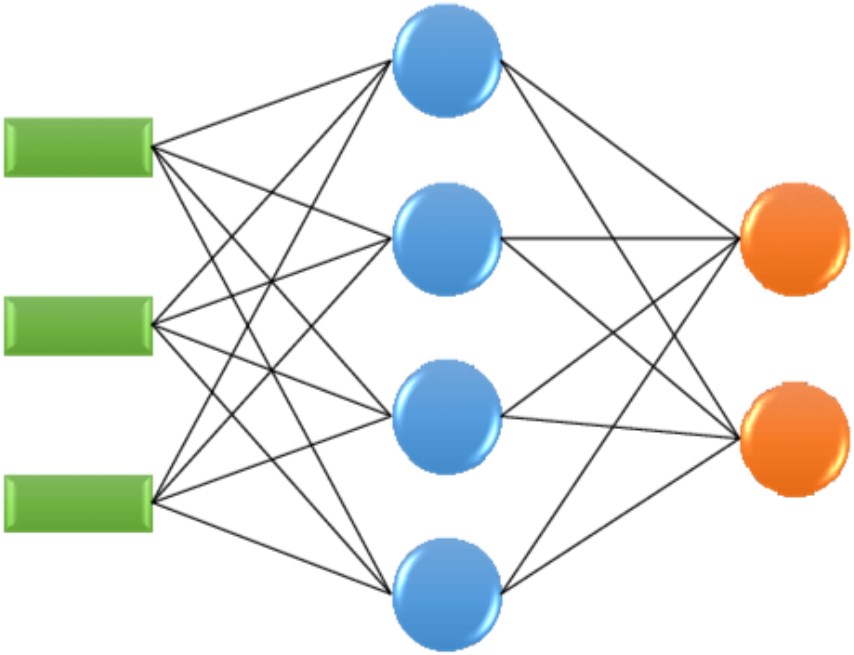
Universidad Industrial de Santander

¿QUÉ ES LA RETINOPATÍA DIABÉTICA?

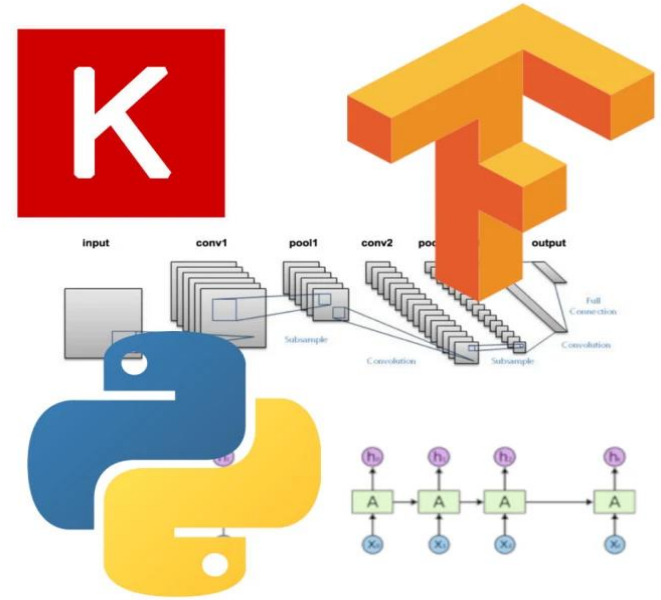
Daño a la retina con peligro para la visión ocasionado por la diabetes, es la causa principal de ceguera entre pacientes en edad productiva.

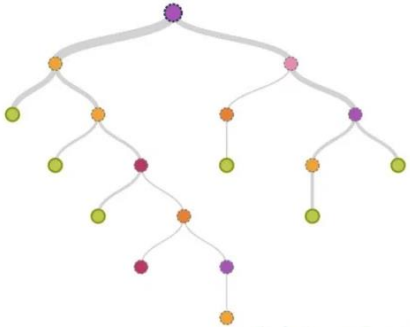


TEMA PRINCIPAL



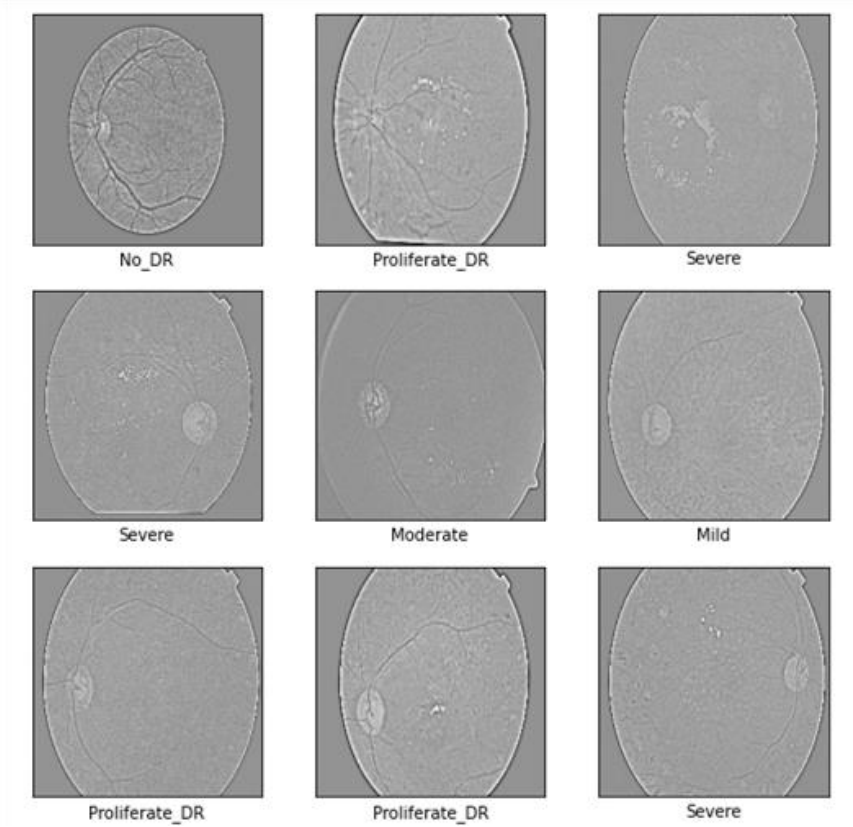
Capa de entrada Capa oculta Capa de salida





Universidad Industrial de Santander

FUNCIONAMIENTO Y SIMULACIÓN



- 0 No_DR (No tiene Retinopatía diabetica)
- 1 Mild (Retinopatía diabetica leve)
- 2 Moderate (Retinopatía diabetica moderada)
- 3 Severe (Retinopatía diabetica severa)
- 4 Proliferate_DR (Retinopatía diabetica proliferada)

(1307, 50176) → TRAIN

(261, 50176) → TEST

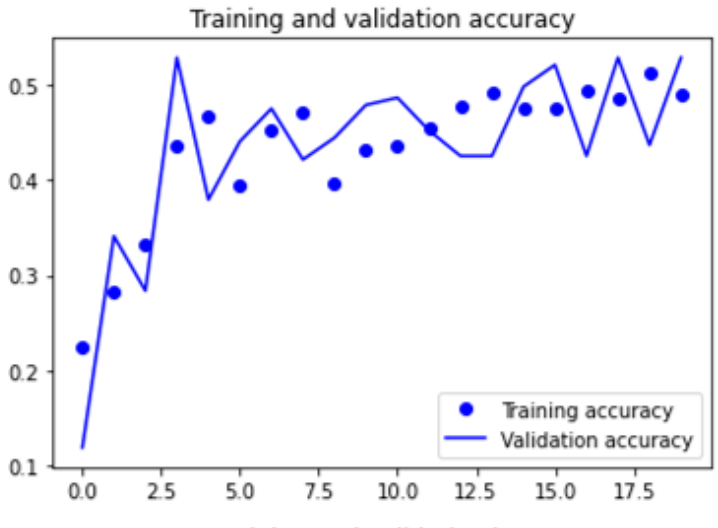
IR AL NOTEBOOK

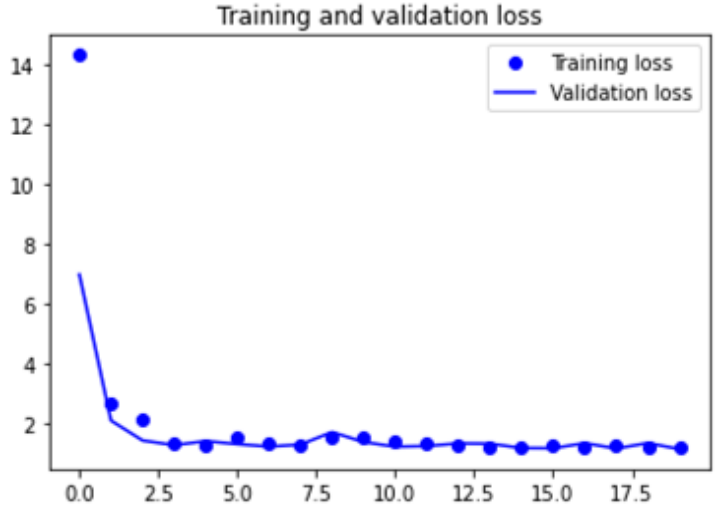
Universidad Industrial de Santander

RESULTADOS

```
[ ]  test_loss,test_acc = model.evaluate(X_test, y_test,verbose=1)
     print('Test accuracy:', test_acc)
```

```
9/9 [==============================] - 0s 16ms/step - loss: 1.1976 - accuracy: 0.5287
Test accuracy: 0.5287356376647949
```





Universidad Industrial de Santander

Clasificación con **DecisionTreeClassifier**

```
[ ] from sklearn.utils import shuffle
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.model_selection import cross_val_score
    from sklearn.metrics import accuracy_score

    est = DecisionTreeClassifier(max_depth=20) #maxima profundidad del arbol
    X_s, y_s = shuffle(X_train,y_train, random_state=0)

    n=int(len(X_s)*0.70) #cantidad de datos para train 70%
    est.fit(X_s[:n], y_s[:n])

    print('%.3f'%accuracy_score(est.predict(X_s[n:]),y_s[n:]))
```

```
0.435
```

Clasificación con **GaussianNB**

```
[ ] from sklearn.naive_bayes import GaussianNB
    from sklearn.model_selection import cross_val_score
    est = GaussianNB()

    est.fit(X_train,y_train)
    predictions = est.predict(X_train)
    print("%.3f"%accuracy_score(predictions, y_train))
```

```
0.599
```

Clasificación con **RandomForestClassifier**

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import cross_val_score
    from sklearn.model_selection import KFold
    from sklearn.metrics import *

    est = RandomForestClassifier()

    s = cross_val_score(est, X_train, y_train, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score))
    print("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
```

```
accuracy 0.567 (+/- 0.05791)
```

Universidad Industrial de Santander

CONCLUSIONES

- Al ejecutar varias veces la red neuronal realizada, la mayor precisión obtenida fue del 52%, es decir que al pasarle una imagen completamente nueva no podremos confiar en el resultado aún, evidentemente no es un buen puntaje, para versiones próximas queda el compromiso de realizar los ajustes pertinentes para así mejorar el modelo y los resultados del mismo.
- Al realizar el entrenamiento con otros modelos que también permiten clasificar, pudimos observar que: con DecisionTreeClasifier se obtuvo 43%, con GaussianNB se obtuvo 59% y con RandomForestClassifier se obtuvo 56% de precisión en los resultados, dando a entender que de los 4 (contando nuestra red) el mejor en este caso en cuanto a confiabilidad es el GuaussianNB.

Universidad Industrial de Santander