TRABAJO FINAL: DETECCIÓN DE SEÑALES

Visión artificial

9 DE DICIEMBRE DE 2020

DAVID RODRÍGUEZ Y ALEJANDRO MEZA

Índice

1.Tecnologías usadas

Con el fin de desarrollar el trabajo de una manera más cómoda, se ha utilizado GitHub para gestionar las versiones del proyecto y poder almacenar archivos necesarios para los objetivos.

2.Metodología

La metodología de trabajo que se ha seguido ha sido iterativa e incremental; produciendo una versión nueva del programa en cada iteración.

3.Planificación del proyecto

Planificación de objetivo 1

Autor de la versión	Número de la versión	Fecha	Descripción
Todos	Versión 1.0	02/12/2020	Lectura de imágenes de las carpetas de los datasets de manera exitosa; previamente habiendo aplicado un resize.
Todos	Versión 1.1	03/12/2020	Obtención de descriptores, mediante diversas técnicas.
Todos	Versión 1.2	08/12/2020	Corrección de aspectos fallidos de la versión anterior y creación de diccionario.
Todos	Versión 1.3	09/12/2020	Implementación de funciones para obtener descriptores sobre cierto grupo de imágenes mediante diversas técnicas.
Todos	Versión 1.4	26/12/2020	Creación del dataset e implementación básica de regresión logística.
Todos	Versión 1.5	27/12/2020	Implementación de gráficas en el modelo de regresión logística.

Todos ---------- Todos	Versión 1.6 ---------- Versión final	28/12/2020 ---------- 29/12/2020	Corrección de aspectos fallidos de la versión anterior. ---------- Implementación de SVM. Obtención de resultados definitivos en SVM y regresión logística. Elección definitiva de mejor modelo. Implementación de técnicas de preprocesamiento de imágenes.

Planificación de objetivo 2

Autor de la versión	Número de la versión	Fecha	Descripción
Todos	--- No es versión--	30/12/2020 y 31/12/2020	Obtención de imágenes de no señales y creación de parches.
Todos	Versión 1.0	1/1/2021	Creación de diccionario de elementos que son no señales. Entrenamiento de modelo con datos de señales y no señales. Implementación básica de ventana deslizante.
Todos	Versión 1.1 (final)	2/1/2021	Corrección de ventana deslizante e implementación de clasificador SVM tras obtención de ventana

Planificación de objetivo 3

Autor de la versión	Número de la versión	Fecha	Descripción
Todos	Versión 1.0	3/1/2021	Predicción de las señales encontradas mediante la técnica de la ventana deslizante. Creación de programa capaz de crear parches de imágenes con el fin de aumentar el dataset.
Todos	Versión 1.1	4/1/2021	Aplicación de técnicas de preprocesamiento de imagen para mejorar resultados obtenidos.
Todos	Versión 1.2	5/1/2021	Refinamiento de la versión anterior.
Todos	Versión final	6/1/2021	Refinamiento de la versión anterior e inclusión de explicaciones breves en el notebook

Día 7/01/2021: finiquito de memoria y creación de vídeo.

4. Objetivo 1

Para cumplir el objetivo 1, se ha dividido este en dos tareas distintas:

- Creación de datos: a partir de las imágenes, se han obtenido sus descriptores; siendo estas las características que hemos usado.
- Utilización de modelo de predicción: A partir del conjunto de características que ha sido obtenido, se usa un modelo existente para predecir nuevos datos.

1. Creación de datos

1.1 Relacionar carpetas con imágenes

Para la creación de datos, se ha creado primero una función que sea capaz de relacionar el nombre de cada carpeta de señales con el tipo de señales que contiene. Se consideró que hacer esto es fundamental, pues aportará mayor facilidad después a la hora de clasificar imágenes y una visión más intuitiva del código.

Después, se creó una función, que, con ayuda de la anterior, es capaz de crear un diccionario cuyas claves son el tipo de señal, y sus valores son las imágenes.

1.2 Obtención de descriptores

Tal como se puede apreciar en el notebook, se han realizado una serie de funciones capaces de obtener descriptores mediante diferentes técnicas*: HOG, SHIFT, FASTBRIEF y ORB*.

El objetivo principal de un descriptor de características es poder generalizar el objeto de tal manera, que el mismo objeto produzca un descriptor de características similares cuando se vea afectado por diferentes condiciones.

Pese a la idea inicial de probar con distintos métodos, los que no son HOG devuelven descriptores de dimensiones no uniformes; luego para crear un dataset, esto resulta verdaderamente problemático.

Por tanto, la técnica elegida ha sido HOG. HOG significa simplemente “histograma de gradientes orientados”. Se centra en contar la ocurrencia de la orientación del gradiente en partes concretas de una imagen.

1.3 Creación de dataset

Tras obtener las características, se ha creado el dataset con el que se va a trabajar. Se obtienen el conjunto de entrenamiento y de prueba. El porcentaje ha sido del 60% y 40% respectivamente.

2. Utilización de modelo de predicción

Una vez con los conjuntos de dato creados, es momento de aplicar modelos de *machine learning para* conseguir clasificar los distintos tipos de señales que hay.

Si bien es cierto que se pudo realizar un único modelo, se consideró positiva la idea de probar dos modelos distintos y enfrentarlos entre sí. Decidiendo a partir de la precisión en el conjunto de prueba, cuál de los dos modelos es el ideal.

En primer lugar, se probó **regresión logística**. Debido a que si n es grande (relativo a m), siendo n el número de características y m el número de ejemplos se recomienda usar regresión logística. Como ese no es el caso exacto de nuestro problema, se pensó que no se iba a obtener un buen resultado. Efectivamente, eso ocurrió.

[124]:
```
filas = prediccionTestmatriz.shape[0]
soluciones = [] #lista de soluciones que se han predicho

for i in range(filas):
    aux = prediccionTestmatriz[i,:] #se obtiene la fila de la matriz
    posicionesUnos = (np.where(aux==1)[0]) #se obtienen las posiciones unos
    signalPredicha = (np.argmax(prediccionesPorSignal[posicionesUnos]))
    nombreSignals= deNumeroASignal(posicionesUnos[signalPredicha])
    soluciones.append(signals_types[nombreSignals])
```

[125]:
```
ytestComparar = list(ytest)


aciertos =  0
for i in range(len(ytestComparar)):
    if ytestComparar[i] == soluciones[i]:
        aciertos+=1

porcentajeAciertos = ( aciertos/(len(soluciones)) )*100
print(f"El porcentaje de aciertos es {porcentajeAciertos}")
```

```
El porcentaje de aciertos es 14.857142857142858
```

El porcentaje de acierto en el conjunto de prueba **no llegó ni al 15%;** por lo que se descartó inmediatamente el modelo.

Como alternativa, se pensó en la utilización de **SVM**, basándose en algunos artículos que lo señalaban como un modelo competente para la clasificación de imágenes.

Al crear el modelo se aplicó *GridSearchCV*. ¿En qué beneficia su uso? Un clasificador SVM tiene una serie de hiperparámetros; como lo son C o gamma. Encontrar una combinación de estos que proporcione un funcionamiento ideal, puede ser una tarea bastante compleja, así que puede considerarse crear una cuadrícula -*grid*- y probar todas las combinaciones posibles. Es ahí donde *GridSearchCV* entra en escena, y elige la mejor combinación de manera automática.

Tras probar el modelo con los datos de entrenamiento, se consiguió un resultado que superaba a la regresión logística de sobremanera: una **precisión del 83%.** Por tanto, de cara al objetivo número dos se consideró exclusivamente SVM.

VelocidadMaxima70	0.96	0.96	0.96	26
VelocidadMaxima80	0.76	0.88	0.81	25
accuracy			0.83	524
macro avg	0.73	0.70	0.70	524
weighted avg	0.82	0.83	0.82	524

2.1 Preprocesamiento de datos tras elección de modelo

De cara al preprocesamiento de imágenes, se decidió por la aplicación de un filtro de suavizado gaussiano y ecualización del histograma. La idea fue comparar ambas posibilidades, y ver con cuál se obtenía una mayor precisión en la creación de una SVM.

Empezando por la opción quizás menos prometedora a priori; ecualización del histograma, se obtuvo una precisión del 80%. Siendo esta inferior a la obtenida sin ningún tipo de preprocesamiento.

Por otro lado, el suavizado mediante filtro gaussiano -se implementó mediante una librería existente de cv2- se probó con diferentes tamaños de ventanas: 3x3, 5x5, 7x7, 9x9 y 11x11. Los resultados obtenidos fueron los siguientes:

-Ventana de 3x3: precisión del 85,71%
-Ventana de 5x5: precisión del 85,90%
-Ventana de 7x7: precisión del 86,47%
-Ventana de 9x9: precisión del 86,47%
-Ventana de 11: precisión del 86,47%

Como se puede observar, el uso de una ventana de 7x7 **mejora la precisión del clasificador en un 3%**, aumentando así las posibilidades de éxito de cara a la futura clasificación de señales mediante la técnica de ventana deslizante.

```
Classification report for -
GridSearchCV(estimator=SVC(),
             param_grid=[{'C': [1, 10, 100, 1000], 'kernel': ['linear']},
                         {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001],
                          'kernel': ['rbf']}]):
                                        precision    recall  f1-score   support

                    AdelantamientoProhibido       0.90      0.95      0.93        20
        AdelantamientoProhibidoParaCamiones       1.00      0.96      0.98        27
                      CalzadaConPrioridad       0.97      1.00      0.98        28
                                 CedaElPaso       1.00      1.00      1.00        29
                     CirculacionProhibida       1.00      0.75      0.86         8
              CurvaPeligrosaHaciaLaDerecha       0.40      0.40      0.40         5
             CurvaPeligrosaHaciaLaIzquierda       0.00      0.00      0.00         3
           CurvasPeligrosasHaciaLaIzquierda       0.75      0.75      0.75         4
                      DetencionObligatoria       1.00      1.00      1.00        10
              DirPermitidasRectoEIzquierda       1.00      1.00      1.00         3
               DirPermitidasRectoYDerecha       1.00      0.80      0.89         5
                           EntradaProhibida       0.94      1.00      0.97        15
       EntradaProhibidaAVehiculosMercancias       1.00      1.00      1.00         6
                    EntradaProhibidasCiclos       1.00      0.25      0.40         4
            EstrechamientoCalzadaPorDerecha       0.00      0.00      0.00         4
                         FinDeProhibiciones       1.00      0.67      0.80         3
              FinProhibicionAdelantamiento       1.00      0.67      0.80         3
       FinProhibicionAdelantamientoCamiones       0.75      1.00      0.86         3
                      FinVelocidadMaxima80       1.00      0.83      0.91         6
                  InterseccionConPrioridad       0.76      0.72      0.74        18
InterseccionSentidoObligatorioGiratorio       0.71      1.00      0.83         5
                                    Kids       0.25      0.29      0.27         7
                                   Obras       0.54      1.00      0.70        20
                            OtrosPeligros       0.83      0.94      0.88        16
                  PasoDeAnimalesEnLibertad       1.00      1.00      1.00        10
                           PasoDePeatones       1.00      0.33      0.50         3
                    PasoObligatorioDerecha       1.00      1.00      1.00        28
                    PasoObligatorioizquierda       1.00      1.00      1.00         4
                        PavimentoDeslizante       0.75      0.43      0.55         7
              PavimentoDeslizanteNieveHielo       1.00      0.33      0.50         6
                         PerfilIrregular       0.83      1.00      0.91         5
                                Semaforos       0.83      0.62      0.71         8
                       SentidoObligatorio       1.00      0.94      0.97        16
                  SentidoObligatorioDerecha       1.00      0.89      0.94         9
                 SentidoObligatorioIzquierda       1.00      1.00      1.00         6
                       VelocidadMaxima100       0.86      1.00      0.93        19
                       VelocidadMaxima120       0.88      0.79      0.83        19
                        VelocidadMaxima20       1.00      0.33      0.50         3
                        VelocidadMaxima30       0.84      0.70      0.76        30
                        VelocidadMaxima50       0.80      0.80      0.80        30
                        VelocidadMaxima60       0.85      0.89      0.87        19
                        VelocidadMaxima70       0.93      1.00      0.96        26
                        VelocidadMaxima80       0.68      0.84      0.75        25

                                 accuracy                           0.86       525
                                macro avg       0.84      0.76      0.78       525
                             weighted avg       0.87      0.86      0.85       525
```

Breve explicación de qué hace cv2.GaussainBlur: Aplica una función gaussiana sobre una imagen, provocando la reducción de ruido y detalle.

5. Objetivo 2

El segundo objetivo se ha dividido principalmente en tres tareas:

- Obtención de imágenes de elementos que no son señales.
- Creación de dataset con nuevos datos.

- Implementación ventana deslizante con SVM

5.1 Obtención de imágenes de elementos que no son señales

Se ha obtenido imágenes de no señales a partir del dataset proporcionado (se ha procedido a hacer parches a partir de las imágenes contenidas) y de algunas imágenes de internet.

Las imágenes escogidas han sido de una serie de elementos que se considera que pueden estar en la vía pública, y, por ende, aparecer en una imagen candidata a tener señales: *carretera, cielo, coche, árboles, persona, semáforo, ventana, paso de cebra…*

En un principio el número de no señales fue bastante bajo, pero se decidió aumentarlo. Al hacer esto, el clasificador mejoró.

5.2 Creación de dataset con nuevos datos

Con el nuevo conjunto de datos se ha procedido a crear un diccionario de no señales, para posteriormente volver a crear un dataset. Esta vez, conteniendo tanto señales como no señales.

Al igual que el objetivo 1, se ha obtenido las características mediante el método Hog. Siguiendo así, con la idea del objetivo anterior.

5.3 Implementación ventana deslizante con SVM

Se implementa un recorrido sobre la imagen utilizando una ventana deslizante. Se empieza con una ventana cuadrada con las dimensiones del lado mínimo de la imagen original. A partir de ahí, disminuirá iteración a iteración hasta ser menor o igual que 30.

En cada una de estas iteraciones se dividirá el lado de la ventana entre 1.5 y se volverá a analizar la imagen teniendo una separación entre ventanas de la mitad del lado de la ventana con la que se está trabajando. Evitando de esta manera, pasar por alto alguna señal.

Tras obtener la ventana, se aplica el clasificador sobre esta para determinar si hay una señal o no.

El clasificador previamente se entrenó con el conjunto de entrenamiento y se probó con el conjunto de prueba. Los resultados obtenidos fueron los siguientes:

```
Classification report for -
GridSearchCV(estimator=SVC(),
             param_grid=[{'C': [1, 10, 100, 1000], 'kernel': ['linear']},
                         {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001],
                          'kernel': ['rbf']}]):
              precision    recall  f1-score   support

   No Signal       0.99      1.00      1.00      5320
      Signal       0.99      0.95      0.97       525

    accuracy                           0.99      5845
   macro avg       0.99      0.97      0.98      5845
weighted avg       0.99      0.99      0.99      5845
```

6. Objetivo 3

6.1 Introducción

Como parte final del proyecto, se unen los dos objetivos anteriores para crear una función, que dada una imagen detecte si existe una señal, y que pueda determinar de cuál se trata.

6.2 Creación del dataset

Se siguen los pasos de los objetivos anteriores, aunque no se muestran ejemplos; pues son los mismos que los anteriores.

6.3 Creación de los clasificadores

Se han reusado los clasificadores de los objetivos anteriores, pudiéndose ver en estos su eficacia con un conjunto de entrenamiento determinado.

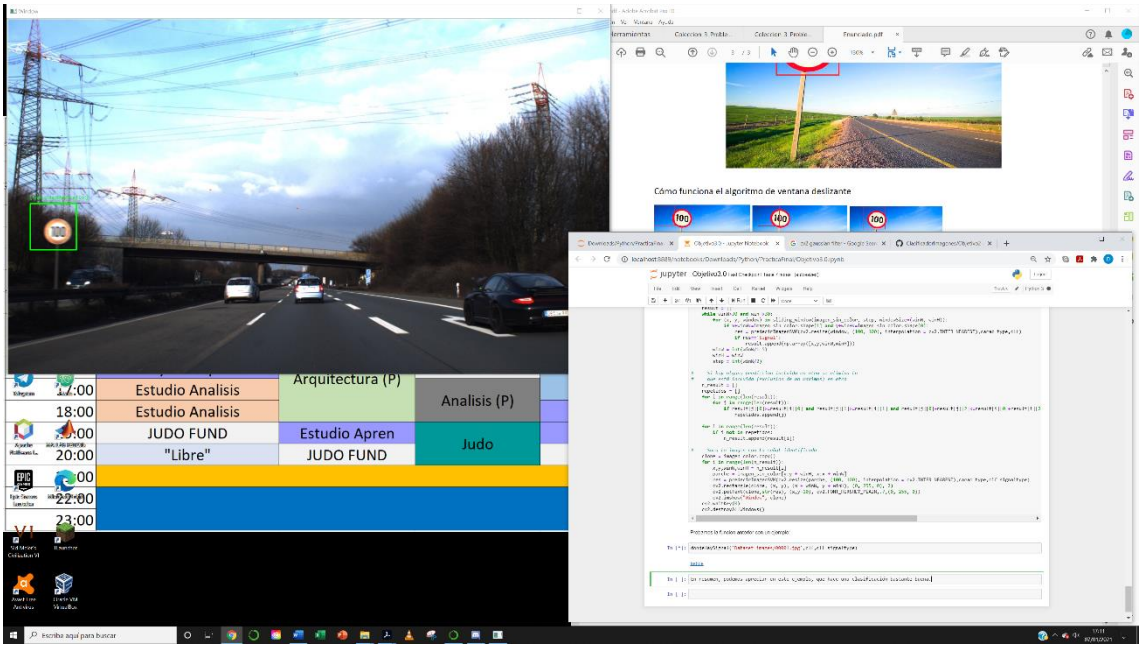
6.4 Predictor Final

Por último, se ha creado una función capaz de devolver una ventana con una imagen, enmarcando las señales encontradas y, encima de estas, la predicción del tipo de señal.

La función 'dondeHaySignal' consta de tres partes:

1. Se predicen los parches que son señal y se guardan en una lista.
2. excluimos los parches no máximos de la lista.
3. Se muestra la imagen con las señales identificadas.

Ejemplo de soluciones:




Window