Taller 2 - Traffic Sign Classification and Recognition

Jaramillo Espinosa Luis Santiago

Facultad de ingeniería, Pontificia Universidad Javeriana de Bogotá

1. COMPRENSIÓN DEL DATASET

¿QUÉ INFORMACIÓN PRESENTA EL DATASET?

El DataSet tomado de Kaggle (https://www.kaggle.com/datasets/wjybuqi/traffic-sign-classification-and-recognition/code) consta de 2 carpetas, 'test_dataset' y 'train_dataset', el primero consta de 324 imágenes clasificadas en 10 tipos diferentes de señales de transito. Mientras que el 'train_dataset' consta de 6.025 imágenes también clasificadas en 10 tipos de señales.

DESCRIBIR LAS CARACTERÍSTICAS DE LAS IMÁGENES

Existen 10 tipos de imágenes para este DataSet:

'GuideSign': Son señales grandes azules que muestran información para guiar al conductor y hacer que tome el carril indicado dependiendo del destino.

'M1' y 'M4': Son letreros azules que se caracterizan por tener una flecha indicando una dirección.

'M5': Son letreros azules que indican el carril que deben tomar los carros o buses.

'M6': Es azul e indica carril de bicicleta.

'M7': Es azul, con un triangulo blanco indicando paso peatonal.

‘P1’: Letrero circular, indicando prohibido parquear.

‘P10_50’: Letrero circular rojo indicando velocidad máxima de 50.

‘P12’: Letrero Chino.

‘W1’: Letrrero amarillo triangular

INDICAR CONJUNTOS DE ENTRENAMIENTO Y PRUEBAS

Los conjuntos son ‘test_dataset’ y ‘train_dataset’, el primero consta de 324 imágenes clasificadas en 10 tipos diferentes de señales de transito. Mientras que el ‘train_dataset’ consta de 6.025 imágenes también clasificadas en 10 tipos de señales.

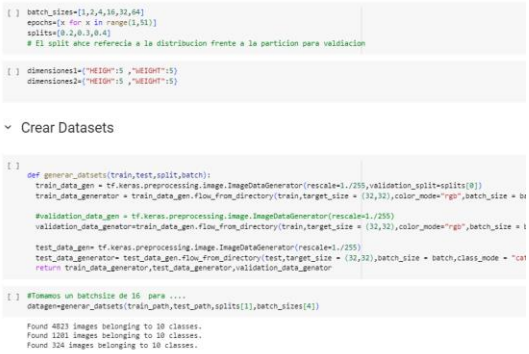
INDICAR LAS CLASES DE CLASIFICACIÓN

De acorde al DataSet se puede utilizar los 10 tipos diferentes de imágenes como clases de

clasificación: ‘GuideSign’, ‘M1’, ‘M4’, ‘W1’, ‘M5’, ‘M6’, ‘M7’, ‘P1’, ‘P10_50’, ‘P12’, ‘W1’. Son categóricas pero se pueden transformar a numéricas para facilitar el modelo.

2. CONSTRUCCION DEL DATASET

Se particiona el DataSet 80% y 20% como solicitaba por medio del paramtero Split , provisto en los genearadores de data .



3. ELABORACION DEL MODELO 1

Se crea el modelo con:

Capa convolucional con 32 detectores de características, tamaño de kernel de

5x5, función de activación lineal rectificada (ReLU) de paso 1 y con panding.

• Capa de max pooling de tamaño 5x5

• Capa de Flattening

• Capa full conectada con 100 neuronas y función de activación lineal rectificada (ReLU)

• Capa de salida del mismo número de las clases a clasificar y función de activación softmax.

```
def modelo1(train_dataset,test_dataset,validation_dataset,pixel):
  # Inicializar la CNN
  classifier = tf.keras.Sequential()
  #1) Capa convolucional
  capa1=Conv2D(filters = 32,
               padding='same',
               kernel_size = (5, 5),
               use_bias=1,
               input_shape = (pixel, pixel, 3),
               activation = "relu")
  #2) Capa de Maxppolin
  capa2=MaxPooling2D(pool_size = (5,5))
  # 3) Dropaout
  capa3=Dropout(0.25)
  #4) Capa de Flatten
  capa4=Flatten()
  #5) Full conection
  capa5=Dense(100, use_bias=1, activation = "relu")
  #6) Capa de Salida
  capa6=Dense(10, use_bias=1, activation = "softmax")

  #Armado de la red neuronal
  classifier.add(capa1)
  classifier.add(capa2)
  classifier.add(capa2)
  classifier.add(capa4)
  classifier.add(capa5)
  classifier.add(capa6)

  classifier.summary()

  return classifier
```

4. ELABORACION DEL MODELO 2

```
modelo2(train_dataset,test_dataset,validation_dataset,pixel):
  # Inicializar la CNN
  classifier = tf.keras.Sequential()
```

```
  #1) Capa convolucional
  capa1=Conv2D(filters = 48,
               padding='same',
               kernel_size = (3, 3),
               use_bias=1,
               input_shape = (pixel,
pixel, 3),
               activation = "relu")
  #2) Capa de Maxppolin
  capa2=MaxPooling2D(pool_size =
(2,2))
  # 3) Dropaout
  capa3=Dropout(0.25)
  #4) 2 Capa covolucional
  capa4=Conv2D(filters = 96,
               padding='same',
               kernel_size = (3, 3),
               use_bias=1,
               input_shape = (pixel,
pixel, 3),
               activation = "relu")
  # 5) argramos otra capa de
maxpooling
  capa5=MaxPooling2D(pool_size =
(2,2))
  # 6) Agrgamso otra capa de dropoout
  capa6=Dropout(0.25)
  # 7) agrgamos una capa de Flatten
  capa7=Flatten()
  #8) Full conection
  capa8=Dense(100, use_bias=1,
activation = "relu")
  capa9=Dense(100, use_bias=1,
activation = "relu")
  #9) Capa de Salida
  capa10=Dense(10, use_bias=1,
activation = "softmax")



  classifier.add(capa1)
  classifier.add(capa2)
  classifier.add(capa2)
  classifier.add(capa4)
  classifier.add(capa5)
  classifier.add(capa6)
  classifier.add(capa7)
  classifier.add(capa8)
  classifier.add(capa9)
  classifier.add(capa10)
```

```
  classifier.summary()
  return classifier
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 48)        1344

max_pooling2d_1 (MaxPoolin   multiple                  0
g2D)

conv2d_2 (Conv2D)            (None, 8, 8, 96)          41568

max_pooling2d_2 (MaxPoolin   (None, 4, 4, 96)          0
g2D)

dropout_2 (Dropout)          (None, 4, 4, 96)          0

flatten_1 (Flatten)          (None, 1536)              0

dense_2 (Dense)              (None, 100)               153700

dense_3 (Dense)              (None, 100)               10100

dense_4 (Dense)              (None, 10)                1010

=================================================================
Total params: 207722 (811.41 KB)
Trainable params: 207722 (811.41 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Para la construcción del clasificador la red neuronal en un segundo parámetro de configuración deberá contar con las siguientes especificaciones

• Capa convolucional con 48 detectores de características, tamaño de kernel de 3x3, funcion de activación lineal rectificada (ReLU) de paso 1 y con panding.

• Capa de max pooling de tamaño 2x2

• Capa convolucional con 96 detectores de características, tamaño de kernel de 3x3, funcion de activación lineal rectificada (ReLU) de paso 1 y con panding.

• Capa de max pooling de tamaño 2x2

• Capa de Flattening

• Capa full conectada con 100 neuronas y funcion de activación lineal rectificada (ReLU)

• Capa full conectada con 100 neuronas y funcion de activación lineal rectificada (ReLU)

• Capa de salida con las 10 calses de clasificación

5. COMPILACION DE LOS MODELOS

Para el tema de la compilación de los algoritmos se creó una función que compilaba tanto el modelo 1 como el modelo 2 empleando para ellos las mismas características descritas a partir de :

- Una función de perdida de entropía cruzada para variables categóricas
- Un optimizador de Adam
- 4 metricas , accuracy, precisión , f1_score , y recall

Esto con la finalidad de que se pudieran comparar, puesto que emplean el mismo entorno de compilación.

En cuanto al nuemro de pasaos seleccionado para el modelo no se emplearon solo 1 unidad, sino que se utilizaron 150 pasos tanto para validación como para entrenamiento teniendo en cuenta la distribución de datos de los generadores y los batch size respectivamente para tener una mejor distribución del número e veces que va a revisar los datos.

```
defCompilacion(modelo1,model
o2):
#compilaciond de los modelos
#'f1_score', 'recall
modelo1.compile(loss
="categorical_crossentropy",
optimizer="adam",metrics=['a
cc',tf.keras.metrics.Precisi
on(name="precision"),tf.kera
s.metrics.F1Score(name="f1_s
core"),tf.keras.metrics.Reca
ll(name="recall")],run_eager
ly=True)

modelo2.compile(loss
="categorical_crossentropy",
optimizer="adam",metrics=['a
cc',tf.keras.metrics.Precisi
on(name="precision"),tf.kera
s.metrics.F1Score(name="f1_s
core"),tf.keras.metrics.Reca
ll(name="recall")],run_eager
ly=True)
```

```
Compilacion(modelo_1,modelo_
2)
```

```
pasostrain1=datagen[0].n//da
tagen[0].batch_size
pasosvalidation1=datagen[2].
n//datagen[2].batch_size
```

Se empleo asi como en el modelo uno un numero de 10 épocas y un bacsize de 16 asi como una división de las imágenes en 32x32 pixeles .

Los pasos fueron alrededor de 150 por época Esto con la finalidad que el algoritmo tuviera un mejor vistazo sobre la información

6. Entrenamiento y evaluación del modelo 1

Inicialmente el modelo durante el entrenamiento presentaba una precisión del 0.70 con un acuraccy del 0.53, debido a que hasta ese momento estaba reconociendo sus primeras señales, ya sobre la época 9 y 10 tanto su precisión como accuracy tenían un valor de 0.83 y 0.85

En cuento al f1_score y al recall obtenido en la época 10 se registraron unos valores de 0.71 y 0.74 respectivamente indicándonos una alta precisión y sensibilidad del modelo.

```
scores=modelo_1.evaluate(dat
agen[1],steps=pasosvalidatio
n1)
scores=modelo_1.evaluate(dat
agen[1],steps=pasosvalidatio
n1)
37/37
[==============================
===] - 1s 17ms/step - loss:
0.5819 - acc: 0.8395 -
precision: 0.9286 - f1_score:
0.7236 - recall: 0.7623
```

Al momento de evaluar el modelo con los datos de prueba se pudo confirmar que la exactitud trazado por el entrenamiento y validación era consecuente con las prueba puesto que estas arrojaron el mismo valor

resultante de 0.83, sumado a esto tras evaluar el modelo otro de los indicares claves fue recall el cual fu disminuyendo causando un aumento en la precisión del modelo.

7. Entrenamiento y evaluación del modelo 2

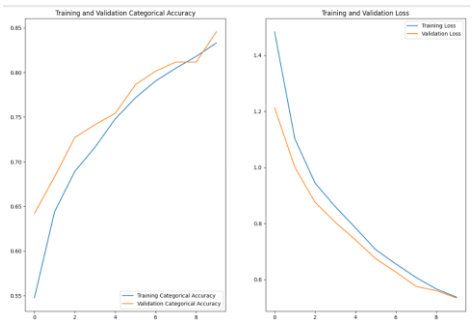
Inicialmente el modelo durante el entrenamiento presentaba una precisión del 0.79 con un acuraccy del 0.65, debido a que hasta ese momento estaba reconociendo sus primeras señales, ya sobre la época 9 y 10 tanto su precisión como accuracy tenían un valor de 0.96 y 0.95

8. AJUSTES

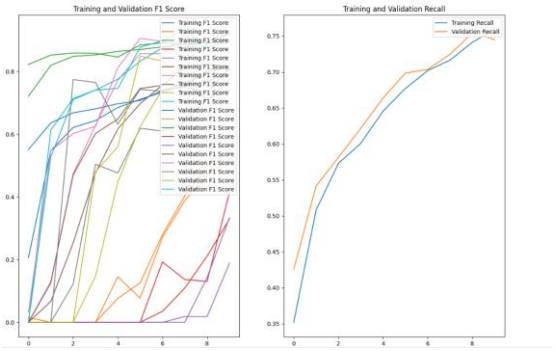
Se empleo como ajuste principal la utilización de Dropouts con valore de 0.25 tras cada Capa de Maxpooling 2D en ambos modelos los cual proporciono una mejora sustancia en el modelo 2 comparado con los resultados del modelo 1, ya que el modelo 2 paso de un test accuracy de 0.85 A uno de 0.96 con una precisión de 0.96 , para el caso del modelo 1 no afecto mucho ya que se mantuvo por el orden de una accuray y precisión del 0.85

9. ANALISIS DE RESULTADOS

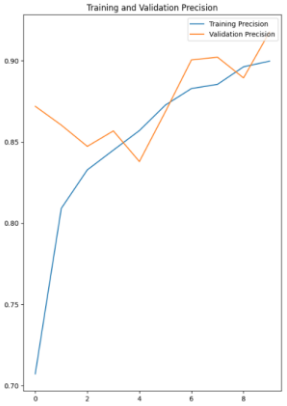
Resultado de Training Modelo1



Para el caso del modelo uno durante el cálculo del del acuracy categórico se pudo observar como a medida que transcurren las épocas obtiene cada vez un mejor rendimiento tanto en el recall como en el categorical acuraccy pasando de un 0.45 y un 0.36

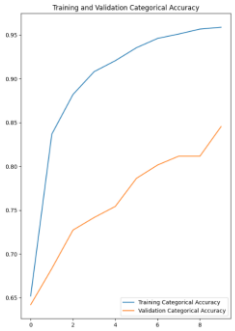


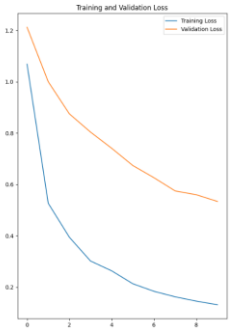
Como se comento con anterioridad los datos de entrenamiento denotaba un mayor incremento en la precisión del modelo conforme se acercaba a las 10 épocas estipulado , sin embargo un aspecto curioso es que en cuestiones de validación el comportamiento del modelo no es similar ya que como lo expresa la siguiente gráfica, durante las primeras 4 épocas la precisión disminuyo hasta su valle más próximo , mientras que tras las mismas esta empezó a incrementar llegando a su pico más alto en la ultima época, esto ha de verse impacto como producto por la gran frecuencia de señales tipo M4 que pudieron confundir un poco el modelo que fue entendiendo mejor la distribución de su información a partir de la época 5 donde se ve un gran incremento en la precisión de los modelos.

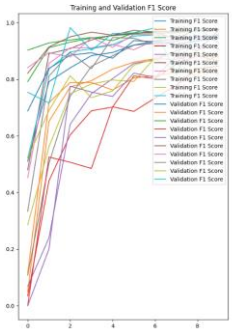


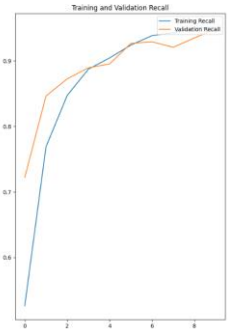
Resultado de entrenamiento del Modelo 2

Para el caso del modelo 2 se observa una clara separación sobre la validación cuestión de perdida, es decir que mientras en validación la función de perdida disminuía de forma moderad o gradual en la de entrenamiento se percibía una disminución abrupta indicando que el conjunto de validación indicando que las señales que se deseaban predecir se encontraban acrodemente clasificadas.


Training and Validation Categorical Accuracy
Training Categorical Accuracy
Validation Categorical Accuracy


Training and Validation Loss
Training Loss
Validation Loss


Training and Validation F1 Score


Training and Validation Recall
Training Recall
Validation Recall