VISIÓN POR COMPUTADOR

DETECCIÓN DE ACCIDENTES AUTOMOVILÍSTICOS POR MEDIO DE VIDEOS DE CCTV

2020

Aplicación de arquitecturas de redes neuronales convolucionales para la clasificación de sucesos de choques en automóviles

Descripción

Proyecto enfocado a la detección de accidentes automovilísticos que puedan presentarse en calles o carreteras.

La detección de accidentes se obtiene a través de videos simulados de CCTV (sistema cerrado de televisión)

Los videos provienen del software de simulación BeamNg y de videos reales de CCTV

El conjunto de datos

- Para la fase de entrenamiento se cuenta con 34 videos para accidente y 34 para no accidente.
- La fase de prueba contiene 25 videos de no accidente y 25 videos de accidente.
- La reducción del conjunto de datos es necesaria para ***evitar el desbalance*** en los datos.
- 118 videos en total.

Pretratamiento

- Los videos originales (1280x720 píxeles, 30 fps)se sometieron a un re escalado de 224x224píxeles, 30 fps.
- Por cada escena de video se seleccionaron segmentos de video que se consideran accidente y en donde no.

Pretratamiento

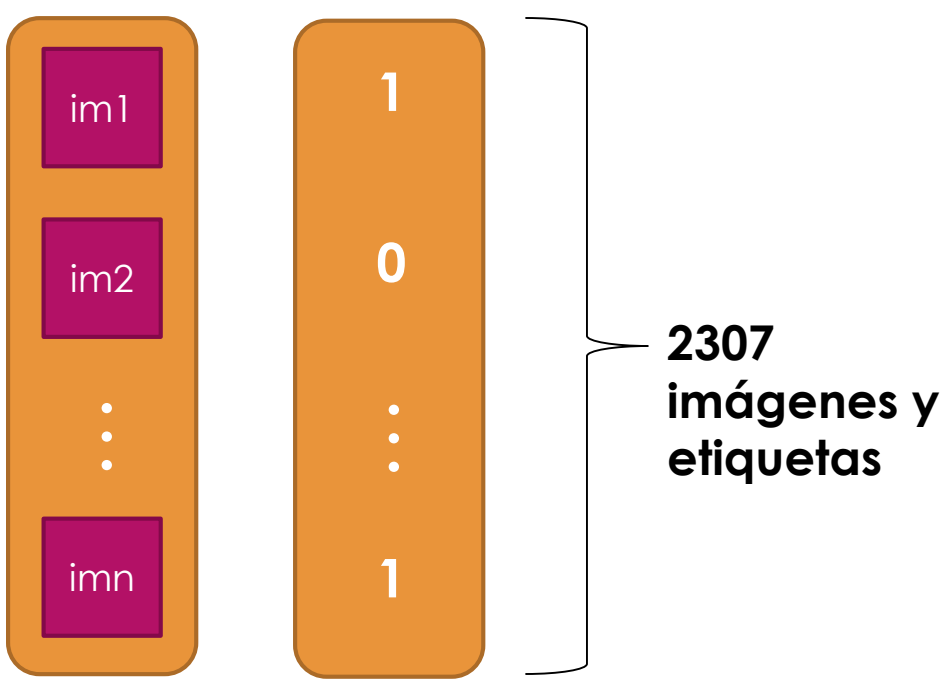
- Los videos pretratados son procesados como imágenes individuales.

Video

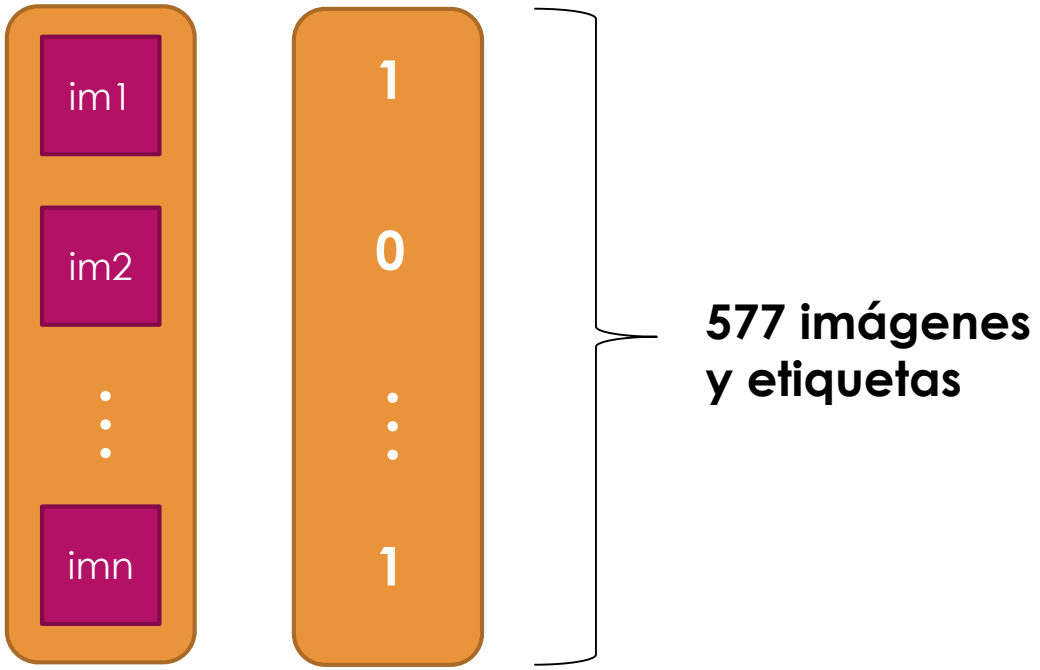
Secuencia de imágenes

Tratamiento

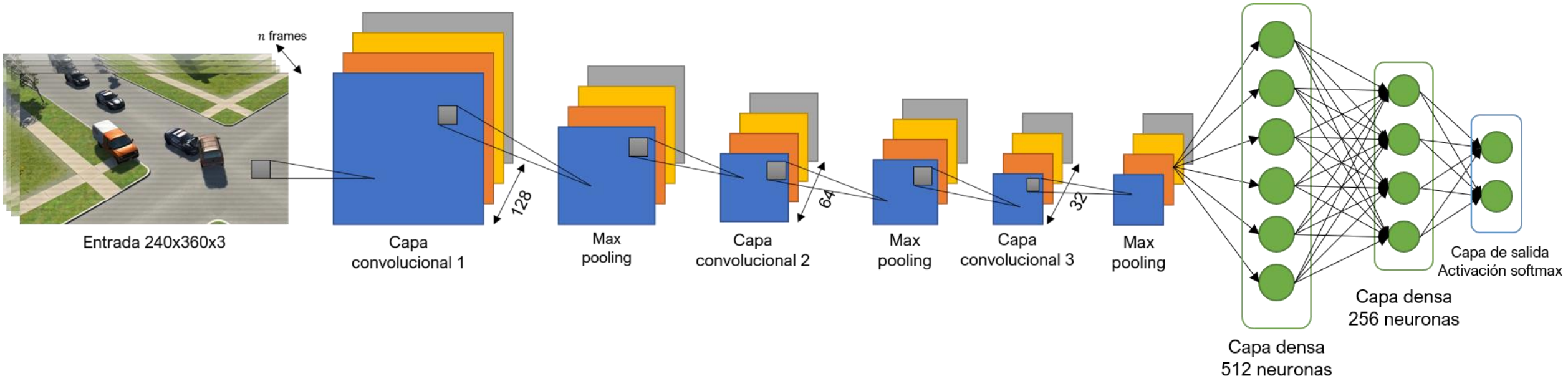
Entrenamiento



Prueba



Modelo preliminar

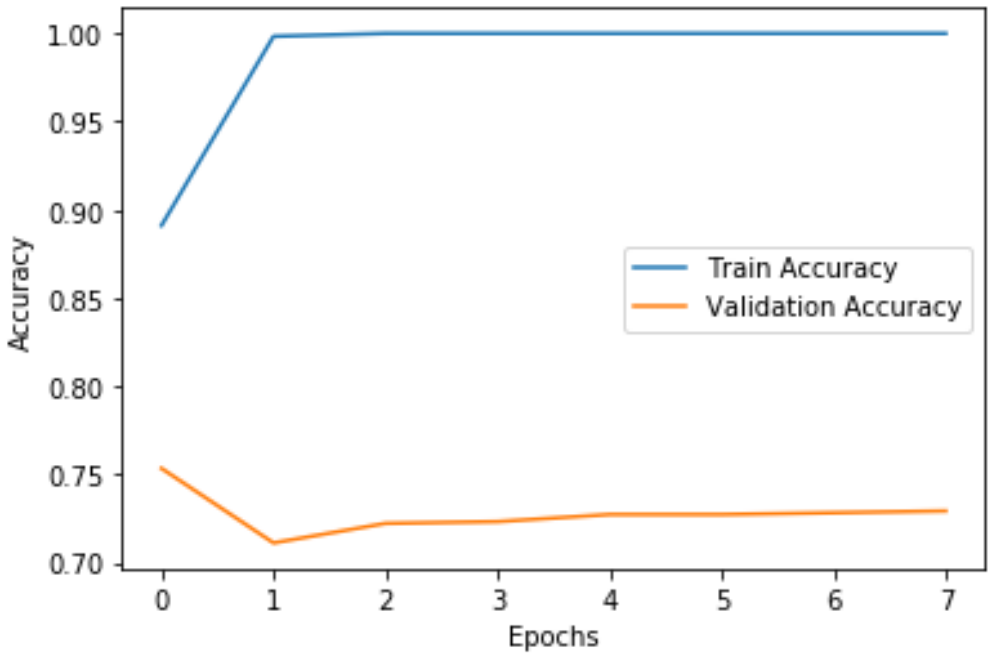


El problema del sobre entrenamiento

```
history = model_CNN.fit(x_tr, Y_train, epochs=8, batch_size=8,
                        validation_data=(x_te, Y_test))
```

```
Train on 1828 samples, validate on 994 samples
Epoch 1/8
1828/1828 [==============================] - 22s 12ms/sample - loss: 0.2432 - acc: 0.8911 - val_loss: 0.6225 - val_acc: 0.7535
Epoch 2/8
1828/1828 [==============================] - 20s 11ms/sample - loss: 0.0092 - acc: 0.9984 - val_loss: 0.9384 - val_acc: 0.7113
Epoch 3/8
1828/1828 [==============================] - 20s 11ms/sample - loss: 2.7692e-04 - acc: 1.0000 - val_loss: 1.0713 - val_acc: 0.7223
Epoch 4/8
1828/1828 [==============================] - 20s 11ms/sample - loss: 1.4528e-04 - acc: 1.0000 - val_loss: 1.1031 - val_acc: 0.7233
Epoch 5/8
1828/1828 [==============================] - 20s 11ms/sample - loss: 1.0349e-04 - acc: 1.0000 - val_loss: 1.1205 - val_acc: 0.7274
Epoch 6/8
1828/1828 [==============================] - 20s 11ms/sample - loss: 8.0380e-05 - acc: 1.0000 - val_loss: 1.1179 - val_acc: 0.7274
Epoch 7/8
1828/1828 [==============================] - 20s 11ms/sample - loss: 6.6533e-05 - acc: 1.0000 - val_loss: 1.1418 - val_acc: 0.7284
Epoch 8/8
1828/1828 [==============================] - 20s 11ms/sample - loss: 5.6200e-05 - acc: 1.0000 - val_loss: 1.1587 - val_acc: 0.7294
```

```
<matplotlib.legend.Legend at 0x7f5dbc72ec18>
```



Modelos pre entrenados con Transfer Learning

- Uso de modelos VGG16, MobileNetV2, InceptionV3, Resnet, Xception
- Congelamiento de capas
- Adición de 3 capas densas con número de neuronas 512, 256 y 2 respectivamente. Última capa incluye activación *softmax*.
- Descenso de gradiente estocástico como optimizador ($lr = 0.001, momentum = 0.9$)

Resultados

Modelo	Precisión entrenamiento	Precisión prueba
VGG16	0.9765	0.4506
MobileNet	1.0	0.4756
Inception	1.0	0.5881
Resnet	1.0	0.5995
Xception	1.0	0.5530

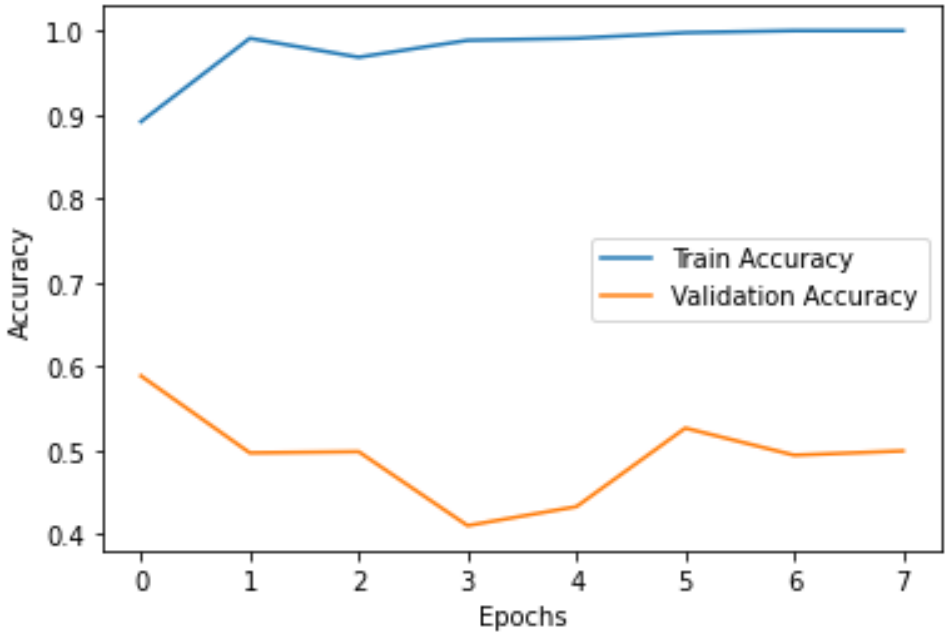
Potenciando el modelo Resnet101V2

- Descongelamiento de capas iniciales (10).
- Adición de capas densas con número de neuronas de 1024, 512, 256 y 2 respectivamente.
- Activación *LeakyReLU* ($\alpha = 0.3$)
- Última capa densa activación *softmax*.
- *Adam* como optimizador($lr = 0.025$)
- 8 *epochs*
- 16 *batch size*

Resultados

Modelo	Precisión entrenamiento	Precisión prueba
Resnet 101 V2 + Transfer Learning	0.9578	0.6231

Resultados

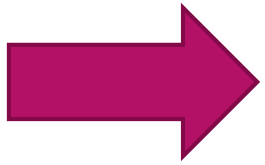



```
historyResnet = modelo_transfer_Resnet.fit(X_train, Y_train, epochs=8, batch_size=16,
                                           validation_data=(X_test, Y_test))
```

```
Epoch 1/8
131/131 [==============================] - 25s 189ms/step - loss: 0.7334 - accuracy: 0.8916 - val_loss: 2.6872 - val_accuracy: 0.5886
Epoch 2/8
131/131 [==============================] - 23s 174ms/step - loss: 0.0447 - accuracy: 0.9909 - val_loss: 4.6427 - val_accuracy: 0.4968
Epoch 3/8
131/131 [==============================] - 23s 174ms/step - loss: 0.1758 - accuracy: 0.9683 - val_loss: 16.7301 - val_accuracy: 0.4982
Epoch 4/8
131/131 [==============================] - 23s 173ms/step - loss: 0.0610 - accuracy: 0.9885 - val_loss: 14.2940 - val_accuracy: 0.4101
Epoch 5/8
131/131 [==============================] - 23s 173ms/step - loss: 0.0421 - accuracy: 0.9909 - val_loss: 9.7944 - val_accuracy: 0.4327
Epoch 6/8
131/131 [==============================] - 23s 173ms/step - loss: 0.0153 - accuracy: 0.9976 - val_loss: 9.7227 - val_accuracy: 0.5263
Epoch 7/8
131/131 [==============================] - 23s 173ms/step - loss: 2.6558e-05 - accuracy: 1.0000 - val_loss: 9.4812 - val_accuracy: 0.4940
Epoch 8/8
131/131 [==============================] - 23s 173ms/step - loss: 5.8093e-06 - accuracy: 1.0000 - val_loss: 9.5214 - val_accuracy: 0.4991
```

Predicciones realizdas

Entrada



Salida

Conclusiones

- Posiblemente el sobre entrenamiento exista dependiendo de la misma naturaleza de los datos.
- No es posible representar en su gran mayoría los diferentes sucesos de accidentes.
- Para versiones futuras es recomendable adquirir un conjunto de datos más amplios e implementar modelos convolucionales 3d.
- Reentrenar el modelo Resnet 101 V2 en su totalidad