.|.

Campus Puebla

Materia

Fundamentación de Robótica TE3001B

Tema

Clasificación de imágenes con redes neuronales convolucionales

Integrantes

José Jezarel Sánchez Mijares A01735226

Antonio Silva Martínez

Fecha

Mayo 25 2023

Instrucciones

Resuelve los siguientes ejercicios en Python y responde a los planteamientos indicados en cada uno. Incluye en tu reporte las salidas de tus programas, y una conclusión breve para esta actividad sobre la efectividad de los métodos probados.

Ejercicio 1 (50 puntos)

Para el conjunto de datos Fashion MNISTLinks to an external site. (problema de clasificación de 10 clases), ajuste una red neuronal convolucional y evalúe su rendimiento con validación cruzada. Reporta los problemas a los que te enfrentaste para obtener tu modelo.

Para este ejercicio utilizamos un como base un código brindado por el profesor, primero como no estamos seguros de las dimensiones de las imágenes y cómo podemos obtener diferentes resultados las redimensionamos a un solo tamaño luego normalizamos y creamos las diez clases basados en la base de datos de Fashion MNISTLinks.

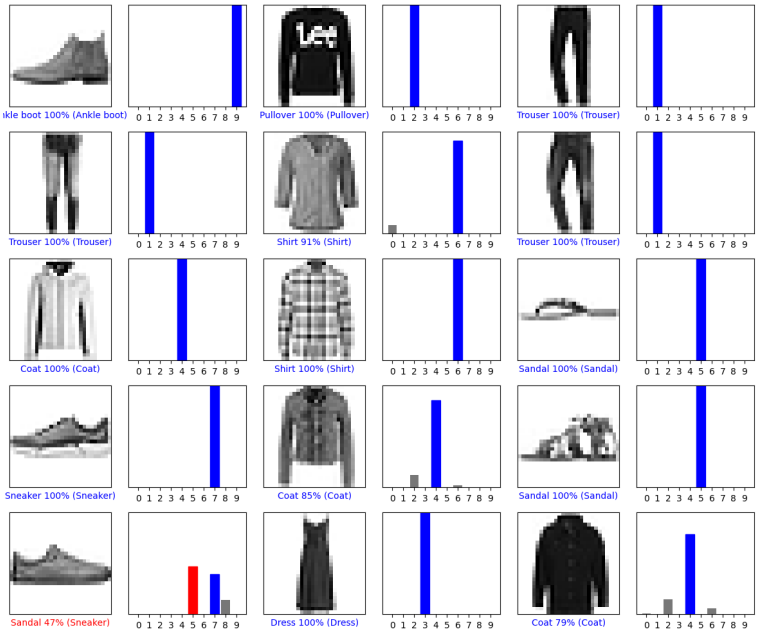
Luego ajustamos la red usando la validación cruzada, para esto definimos kf = KFold(n_splits=5, shuffle=True) para crear un objeto KFold que dividirá los datos en 5 pliegues y se mezclarán aleatoriamente antes de dividirlos. En el bucle “for fold, (train_index, test_index) in enumerate(kf.split(train_images))” iteramos sobre los pliegues generados por KFold.split(). En cada iteración, se obtienen los índices de entrenamiento (train_index) y los índices de prueba (test_index) para el pliegue en el que se encuentra.

Posterior a eso creamos un nuevo modelo en cada pliegue y se ajusta utilizando los datos de entrenamiento y validación específica en cada pliegue, posterior a esto solo imprimimos los resultados de la precisión del modelo y algunas gráficas ilustrativas

#Mostramos imagenes

-

```
Epoch 1/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.4997 - accuracy: 0.8252
Epoch 2/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3686 - accuracy: 0.8654
Epoch 3/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3340 - accuracy: 0.8793
Epoch 4/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3111 - accuracy: 0.8865
Epoch 5/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2907 - accuracy: 0.8919
Epoch 6/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2774 - accuracy: 0.8976
Epoch 7/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2679 - accuracy: 0.9005
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2551 - accuracy: 0.9048
Epoch 9/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2445 - accuracy: 0.9086
Epoch 10/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2361 - accuracy: 0.9118
313/313 - 1s - loss: 0.3465 - accuracy: 0.8797 - 583ms/epoch - 2ms/step
313/313 [==============================] - 1s 2ms/step
```



Ejercicio 2 (50 puntos)

Para el conjunto de datos de German Traffic Sign Recognition BenchmarkLinks to an external site. (GTSRB) (problema de clasificación de más de 40 clases), ajuste una red neuronal convolucional y evalúe su rendimiento con validación cruzada. Reporta los problemas a los que te enfrentaste para obtener tu modelo.

Para este código aplicamos la misma metodología que el anterior, donde decidimos el número de grupos para la validación cruzada, delimitamos los tamaños de la imagenes analizar, ya que, si bien en la primera parte no era necesario del todo, en la segunda si era importante hacerlo ya que había imágenes que no tenían el mismo tamaño lo que afectaría la precisión del modelo

```
Class stop: 780
Class no_way_general: 630
Class no_way_trucks: 420
Class no_way_one_way: 1110
Class attention_general: 1200
Class attention_left_turn: 210
Class attention_right_turn: 360
Class attention_curvy: 330
Class attention_bumpers: 390
Class attention_slippery: 510
Class attention_bottleneck: 270
Class attention_construction: 1500
Class attention_traffic_light: 600
Class attention_pedestrian: 240
Class attention_children: 540
Class attention_bikes: 270
Class attention_snowflake: 450
Class attention_deer: 780
Class lifted_general: 240
Class turn_right: 689
Class turn_left: 420
Class turn_straight: 1200
Class turn_straight_right: 390
Class turn_straight_left: 210
Class turn_right_down: 2070
Class turn_left_down: 300
Class turn_circle: 360
Class lifted_no_overtaking_general: 240
Class lifted_no_overtaking_trucks: 240
[ 0  0  0 ... 42 42 42]
Epoch 1/10
```

```
Class lifted_no_overtaking_trucks: 240
[ 0  0  0 ... 42 42 42]
Epoch 1/10
1226/1226 [==============================] - 298s 242ms/step - loss: 0.8310 - accuracy: 0.7818
Epoch 2/10
1226/1226 [==============================] - 305s 249ms/step - loss: 0.1391 - accuracy: 0.9636
Epoch 3/10
1226/1226 [==============================] - 296s 241ms/step - loss: 0.0741 - accuracy: 0.9799
Epoch 4/10
1226/1226 [==============================] - 287s 233ms/step - loss: 0.0519 - accuracy: 0.9862
Epoch 5/10
1226/1226 [==============================] - 280s 228ms/step - loss: 0.0373 - accuracy: 0.9893
Epoch 6/10
1226/1226 [==============================] - 283s 231ms/step - loss: 0.0300 - accuracy: 0.9909
Epoch 7/10
1226/1226 [==============================] - 286s 234ms/step - loss: 0.0240 - accuracy: 0.9926
Epoch 8/10
1226/1226 [==============================] - 280s 228ms/step - loss: 0.0193 - accuracy: 0.9944
Epoch 9/10
1226/1226 [==============================] - 273s 223ms/step - loss: 0.0215 - accuracy: 0.9938
Epoch 10/10
1226/1226 [==============================] - 272s 222ms/step - loss: 0.0179 - accuracy: 0.9948
<keras.callbacks.History at 0x7f9106e915d0>
```

Conclusiones:

La validación cruzada es una técnica que nos ayuda a evaluar el rendimiento y la generalización de nuestro modelo de aprendizaje automático de una manera más robusta. A continuación, algunos de los puntos claves que aprendimos a partir de trabajar con la validación cruzada fue que nos ayudó a tener una estimación más precisa del rendimiento, detectar el overfitting, en la selección de modelos y ajuste de hiperparametros además de utilizar eficientemente nuestros datos

En general, la validación cruzada es una técnica valiosa para evaluar modelos de aprendizaje automático, ayudándonos a comprender mejor su rendimiento y generalización. Nos brinda resultados más confiables, detecta problemas de sobreajuste y nos permite tomar decisiones más informadas sobre la selección de modelos y la configuración de hiper parámetros.

Liga a Git:
https://github.com/J3z4r3l/Clasificacion_imagenes_redes_neuronales_convolucionales_Ant_Jz.git