1st Summer School Physical Engineering UNI 2022-Lima,Perú

Reporte de Clasificación de Imágenes

Alvaro Alejandro Siesquén Abad[1]

[1]Facultad de Ciencias Físicas, Universidad Nacional Mayor de San Marcos
[1]alvaro.siesquen@unmsm.edu.pe

March 1, 2022

Resumen

En este trabajo se realizó un entrenamiento de una red neuronal para clasificación de imágenes de perros y gatos. Se modificaron algunos parámetros para ver el comportamiento de la red neurona y determinar cual es la que da mayor precisión.

Según los resultados se pudo determinar una combinación óptima, debido a que tenía una mejor precisión de entrenamiento y se ejecutaba en un tiempo menor

Palabras Clave
deep learning, clasificación, max pooling, cross entropy loss

1 Metodología

Se realizó el Proyecto 1: Clasificación de imágenes y usamos el lenguaje de programación Python en el entorno Google Colaboratory. Las librerías importadas son torch, torch.nn, torchvision y matplotlib y se clasificaron imágenes de perros y gatos, por lo cual se usó la dataset chetankv/dogs-cats-images de Kaggle.

Se descargó y descomprimió la dataset, se hizo un procesamiento en las imágenes que uniformiza las dimensiones de las imágenes a 224 px × 224 px y las transforma en un tensor para poder trabajar con las redes neuronales, además, con el objetivo de mejorar la precisión del entrenamiento se rotan las imágenes cierta cantidad de grados y se voltean las imágenes, de manera aleatoria. Se realizó el entrenamiento por lotes de 128 imágenes agrupándolas de manera aleatoria.

Para el modelo se usaron capas de convolución, Batch Norm Layer para prevenir el sobre entrenamiento, la función de activación ReLU y un filtro de Max pooling. Al ser una red neuronal para clasificar imágenes se usó la función de pérdida Cross Entropy Loss y el optimizador fue SGD, el entrenamiento constó de 10 épocas.

Después de obtener los resultados principales se variaron algunos parámetros uno por uno para ver si se lograba aumentar la precisión.

2 Resultados

Se ejecutó una celda con el siguiente código

```
plt.imshow(train_ds[10][0].numpy().transpose(1,2,0))
plt.show()
```

Esto con la finalidad de asegurar que las imágenes se habían guardado de forma correcta con todas las características que se le dio, esto se ve en la figura 1, se ejecutó por segunda vez la celda y se obtiene la misma imagen pero con un aspecto diferente a la anterior, lo que confirma que los procesos aleatorios funcionan bien.

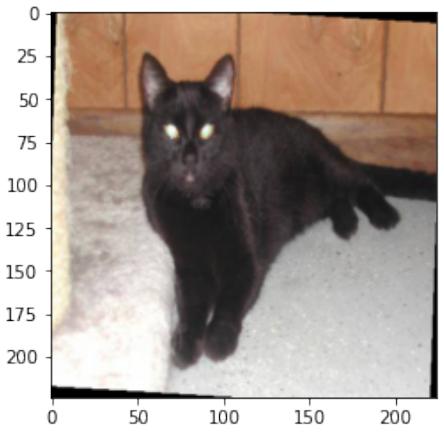

Figura 1: Imagen del dataset después de asignarle los atributos.

Después de ejecutarse las celdas correspondientes obtuvimos los resultados principales y los comparamos con resultados obtenidos variando ciertos detalles:

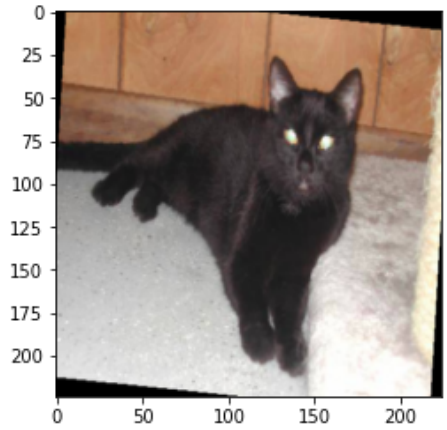

Figura 2: Segunda ejecución de la celda.

2.1 Resultado principal

La ejecución de la celda con la función de entrenamiento demoró en ejecutar 10 minutos y 43 segundos. El output de la última época fue:

```
epoch: 9, train loss: 0.2538215669374617,
train acc: 89.075%,
test loss: 0.4286079127341509,
test acc: 81.0%
```

Se usaron 10 épocas y SGD como optimizador, como se mencionó en la sección de Metodología, con estas características la precisión de entrenamiento para la última época fue de de 89.075 %.

Evaluando el modelo para perros y gatos en un bucle for el programa acertó todas las predicciones.

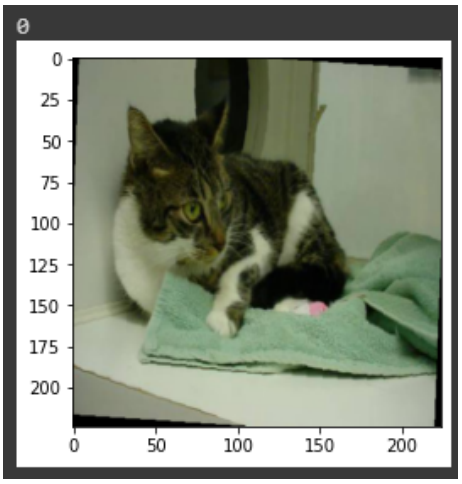

Figura 3: Evaluación 1

2.2 Modificaciones realizadas

Optimizador Adam

Se cambió el optimizador de SGD a Adam para ver el comportramiento de la precisión en este caso.

La ejecución de la celda con la función de entrenamiento demoró en ejecutar 12 minutos y 44 segundos. Los resultados de la ejecución de la última época son:

```
epoch: 9, train loss: 0.5992992771050286,
train acc: 68.6%,
test loss: 0.5972188673913479,
test acc: 68.8%
```

La red neuronal falló varias predicciones, esto se entiende debido a la baja precisión de entrenamiento.

5 épocas

Se redujo el entrenamiento sólo a 5 épocas, manteniendo los valores originales para los demás parámetros.

La ejecución de la celda con la función de entrenamiento demoró en ejecutar 6 minutos y 19 segundos. Los resultados de la ejecución de la última época son:

```
epoch: 4, train loss: 0.43417909410264754,
train acc: 80.125%,
test loss: 0.4176624231040478,
test acc: 80.75%
```

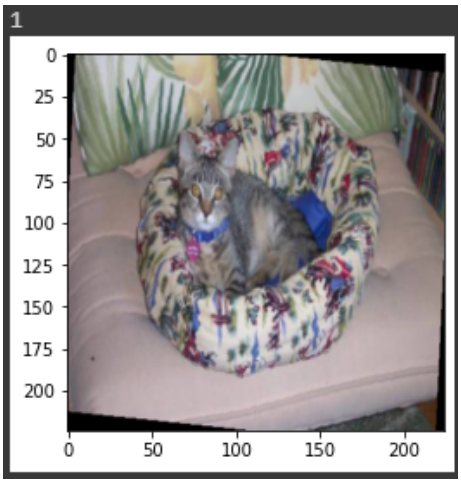

Figura 4: Predicción errada para 5 épocas

La red neuronal pudo reconocer casi todas las imágenes que se colocaron a pesar de tener la mitad de épocas de entrenamiento.

Sin shuffling

Se colocó False como parámetro lo que producirá que los batches se agrupen en orden.

La ejecución de la celda con la función de entrenamiento demoró en ejecutar 12 min y 30 segundos, Los resultados de la ejecución de la última época se muestran a continuación:

```
epoch: 9, train loss: 0.42932475216331933,
train acc: 82.875%,
test loss: 1.710862185806036,
test acc: 50.0%
```

La red falló varias veces al intentar reconocer gatos pues los clasificaba como perros.

3 Discusión de resultados

Los resultados mostraron mayor precisión del código principal frente a cualquiera de las modificaciones realizadas, considerando además que para todos los entrenamientos de 10 épocas, el principal fue el que tuvo menor tiempo de ejecución.

Conclusiones

El código del resultado principal es el más óptimo debido a su menor tiempo de ejecución, mayor precisión de entrenamiento y mejores predicciones.

La mayor caída porcentual se dio cuando se cambió el optimizador por lo que este tendría un rol notable frente a otrs en la precisión de entrenamiento, esta afirmación adquiere mayor validez al ver que cuando se redujo las épocas de entrenamiento a 5, la precisión no disminuyó.

Agradecimientos

Agradezco a mi instructor Elvin Muñoz Vega por enseñarnos el curso con tanto esmero y por resolver las dudas que tuvimos en el camino.

Referencias

[1] Dataset: chetankv/dogs-cats-images. Kaggle. https://www.kaggle.com/chetankv/dogs-cats-images

[2] Pytorch Documentation https://pytorch.org/docs/stable/index.html

[3] Torchvision. https://pytorch.org/vision/stable/index.html