CNN - Clasificación de Rostros Reales vs Falsos

Gabriel Bernal Pinto y David Garcia Diaz

January 2024

Introduction

En este proyecto hemos realizado una red neuronal convolucional capaz de clasificar imágenes de rostros reales y falsos. Posteriormente decidimos cambiar la arquitectura a MobileNet y usar otro modelo diferente para así poder mejorar el redimiento de la clasificación.

Diseño del Modelo

Preprocesamiento de Datos

- Tamaño de las imágenes: 256x256 píxeles.
- Transformaciones aplicadas: Resize, Random Horizontal Flip, Random Rotation, ToTensor, Normalización.
- División del conjunto de datos en entrenamiento (70%) y prueba (30%).

Arquitectura del Modelo (CNNClassifier)

- Capa de convolución 1: 3 canales de entrada, 32 canales de salida, kernel de 3x3, stride de 1, padding de 1.
- Batch Normalization después de la primera capa de convolución.
- Capa de pooling (MaxPool2d) con kernel de 2x2 y stride de 2.
- Capa de convolución 2: 32 canales de entrada, 64 canales de salida, kernel de 3x3, stride de 1, padding de 1.
- Batch Normalization después de la segunda capa de convolución.
- Capa completamente conectada (fc1) con 64x64x64 nodos de entrada y 128 nodos de salida.
- Capa completamente conectada (fc2) con 128 nodos de entrada y 2 nodos de salida (clasificación binaria).

Función de Entrenamiento

- Función de pérdida: CrossEntropyLoss.
- Optimizador: Adam con tasa de aprendizaje de 0.001 y decaimiento de peso de 1e-4.
- Programación de tasa de aprendizaje con StepLR, reduciendo la tasa a la mitad cada 5 épocas.

Resultados

Entrenamiento

- Número de épocas: 15.
- Pérdida de entrenamiento por época (ejemplo):
  - Epoch [1/15], Train Loss: 8.2011
  - Epoch [2/15], Train Loss: 2.1896
  - Epoch [3/15], Train Loss: 2.7748
  - Epoch [4/15], Train Loss: 1.2889
  - Epoch [5/15], Train Loss: 1.2253
  - Epoch [6/15], Train Loss: 0.8901
  - Epoch [7/15], Train Loss: 0.7034
  - Epoch [8/15], Train Loss: 0.7668
  - Epoch [9/15], Train Loss: 0.7057
  - Epoch [10/15], Train Loss: 0.6921
  - Epoch [11/15], Train Loss: 0.5825
  - Epoch [12/15], Train Loss: 0.5407
  - Epoch [13/15], Train Loss: 0.5755
  - Epoch [14/15], Train Loss: 0.6376
  - Epoch [15/15], Train Loss: 0.5299

Evaluación

- Precisión (Accuracy) en el conjunto de prueba: 62.50%.

Conclusiones

- La arquitectura de la red neuronal CNN parece ser relativamente simple, con dos capas de convolución seguidas de capas completamente conectadas.
- La pérdida de entrenamiento disminuye a medida que avanza el entrenamiento, lo que sugiere que el modelo está aprendiendo de los datos de entrenamiento.

- La precisión en el conjunto de prueba es del 62.50% lo cual indica un rendimiento normal o moderado.

  Después de obtener estas conclusiones deducimos que la mejor opción para mejorar la red era mediante el uso de una arquitectura preentrenada como es MobileNet (transfer learning)

Modelo MobileNet

Para este nuevo modelo se aplicaron transformaciones de datos utilizando la clase `transforms.Compose` de PyTorch. Las transformaciones incluyen el redimensionamiento de las imágenes a 256x256 píxeles, volteo horizontal aleatorio, rotación aleatoria de hasta 10 grados, conversión a tensor y normalización. Se utilizó la clase `ImageFolder` de PyTorch para cargar el conjunto de datos y luego se dividió en conjuntos de entrenamiento y prueba en una proporción del 70-30 mediante `random_split`. Se crearon dataloaders para ambos conjuntos con un tamaño de lote de 32.

Construcción del Modelo (MobileNetV2)

Se diseñó el modelo utilizando la arquitectura MobileNetV2 preentrenada, cargada mediante `torch.hub.load`. Se modifica la capa clasificadora para adaptarla a la tarea con dos capas lineales y una función de activación ReLU, seguida de una capa de dropout con una tasa del 30%, un modelo bastante simple.

Configuración de Entrenamiento

Se seleccionó la función de pérdida de entropía cruzada (`nn.CrossEntropyLoss`) y el optimizador Adam con una tasa de aprendizaje de 0.001 y una penalización por peso de 1e-4. Se utilizó un marcador de tasa de aprendizaje (`StepLR`) con un paso de 5 epochs y un factor de reducción de 0.1.

Entrenamiento

El modelo fue entrenado durante 15 epochs:

- Epoch [1/15], Train Loss: 0.6924
- Epoch [2/15], Train Loss: 0.6001
- Epoch [3/15], Train Loss: 0.5356
- Epoch [4/15], Train Loss: 0.5068
- Epoch [5/15], Train Loss: 0.4524
- Epoch [6/15], Train Loss: 0.3611
- Epoch [7/15], Train Loss: 0.2871
- Epoch [8/15], Train Loss: 0.2550
- Epoch [9/15], Train Loss: 0.2134

- Epoch [10/15], Train Loss: 0.1691
- Epoch [11/15], Train Loss: 0.1750
- Epoch [12/15], Train Loss: 0.1448
- Epoch [13/15], Train Loss: 0.1270
- Epoch [14/15], Train Loss: 0.1309
- Epoch [15/15], Train Loss: 0.1603

Evaluación

El modelo se evaluó en el conjunto de prueba después del entrenamiento. La precisión del modelo en las imágenes de prueba se calculó y se informó como el porcentaje de predicciones correctas sobre el total de predicciones. En este caso, se obtuvo una precisión del 87.50%.

Conclusiones

- La red neuronal MobileNetV2 preentrenada demostró ser más eficaz para la clasificación que el modelo previo.
- La pérdida de entrenamiento fue decreciendo durante los epochs de entrenamiento, lo que nos dice que el modelo sigue aprendiendo.
- La precisión del 87.50% en el conjunto de prueba es un indicador positivo, pero se deberían realizar evaluaciones adicionales con otros conjuntos de datos para validar si el modelo es extrapolable para otras situaciones.

Recomendaciones

- Experimentar con hiperparámetros como la tasa de aprendizaje y la tasa de dropout para optimizar el rendimiento del modelo.
- Usar técnicas de validación cruzada para obtener una evaluación más robusta del modelo.
- Explorar técnicas de aumento de datos adicionales para mejorar la capacidad del modelo para generalizar a nuevas instancias.
- Guardar el modelo entrenado para su uso futuro y posiblemente implementarlo en un entorno de producción si demuestra un rendimiento satisfactorio en diversas evaluaciones.