Resumen

En esta práctica, se utilizó la técnica de transferencia de aprendizaje con la red neuronal convolucional ResNet-50 para clasificar un conjunto de datos de imágenes. El proceso involucró la preparación de los datos, la modificación de las capas finales de la red preentrenada, el ajuste del tamaño de las imágenes, la configuración de las opciones de entrenamiento, la realización del entrenamiento y la evaluación del modelo.

Palabras Clave

Transferencia de aprendizaje, ResNet-50, Clasificación de imágenes, MATLAB, Redes neuronales convolucionales

Introducción

La clasificación de imágenes es una tarea fundamental en la visión por computadora, utilizada en una amplia gama de aplicaciones que van desde la detección de objetos hasta el reconocimiento facial y la conducción autónoma. Las redes neuronales convolucionales (CNN) han demostrado ser altamente efectivas para esta tarea debido a su capacidad para aprender características jerárquicas y espaciales a partir de los datos de imagen. Sin embargo, entrenar una CNN desde cero es un proceso intensivo que requiere una gran cantidad de datos etiquetados y recursos computacionales significativos, lo cual puede ser un obstáculo considerable, especialmente para aplicaciones que no disponen de estos recursos.

Para superar estas limitaciones, la transferencia de aprendizaje se ha convertido en una técnica popular y poderosa en el campo de la visión por computadora. Esta técnica permite aprovechar redes preentrenadas en grandes conjuntos de datos, como ImageNet, que contiene millones de imágenes y miles de clases. Al utilizar una red preentrenada, se pueden adaptar las últimas capas del modelo para ajustarse a una nueva tarea específica con un conjunto de datos más pequeño. Esto no solo reduce el tiempo de entrenamiento y los requisitos de datos, sino que también mejora la capacidad del modelo para generalizar en nuevas tareas.

En esta práctica, se utiliza la red ResNet-50, una CNN profunda preentrenada en ImageNet, para clasificar un conjunto de datos personalizado de imágenes. ResNet-50 es conocida por su arquitectura de redes residuales que facilita el entrenamiento de redes profundas, abordando problemas de degradación del rendimiento. Al modificar las últimas capas de ResNet-50, se ajusta el modelo a las categorías específicas del nuevo conjunto de datos. Este enfoque permite obtener un alto nivel de precisión en la clasificación con menos recursos y en un menor tiempo de entrenamiento, demostrando la efectividad y eficiencia de la transferencia de aprendizaje en la clasificación de imágenes.

Desarrollo

Paso 1: Preparar el Conjunto de Datos

Se creó un imageDatastore para cargar las imágenes desde una carpeta especificada y se dividió el conjunto de datos en conjuntos de entrenamiento y validación en una proporción de 80-20.

```
imageFolder = 'C:\Users\jorge\Downloads\all-images';
imds = imageDatastore(imageFolder, 'IncludeSubfolders', true, 'LabelSource',
'foldernames');
[imdsTrain, imdsValidation] = splitEachLabel(imds, 0.8, 'randomized');
```

Paso 2: Cargar un Modelo Preentrenado y Ajustar las Capas

Se cargó la red ResNet-50 y se reemplazaron las capas finales (una capa totalmente conectada, una capa softmax y una capa de clasificación) para adaptarse al número de clases del conjunto de datos.

```
net = resnet50;
inputSize = net.Layers(1).InputSize;
lgraph = layerGraph(net);
newLearnableLayer = fullyConnectedLayer(numel(categories(imdsTrain.Labels)),
'Name', 'new_fc', 'WeightLearnRateFactor', 10, 'BiasLearnRateFactor', 10);
lgraph = replaceLayer(lgraph, 'fc1000', newLearnableLayer);
newSoftmaxLayer = softmaxLayer('Name', 'new_softmax');
lgraph = replaceLayer(lgraph, 'fc1000_softmax', newSoftmaxLayer);
newClassLayer = classificationLayer('Name', 'new_classoutput');
lgraph = replaceLayer(lgraph, 'ClassificationLayer_fc1000', newClassLayer);
```

Paso 3: Ajustar el Tamaño de las Imágenes

Se ajustaron las imágenes al tamaño de entrada de la red (224x224 píxeles) utilizando augmentedImageDatastore.

```
augimdsTrain = augmentedImageDatastore(inputSize(1:2), imdsTrain);
augimdsValidation = augmentedImageDatastore(inputSize(1:2), imdsValidation);
```

Paso 4: Configurar las Opciones de Entrenamiento

Se configuraron las opciones de entrenamiento utilizando el optimizador 'sgdm', con un tamaño de mini-batch de 32, 6 épocas, una tasa de aprendizaje inicial de 1e-4, y otras opciones relevantes.

```
options = trainingOptions('sgdm', 'MiniBatchSize', 32, 'MaxEpochs', 6,
'InitialLearnRate', 1e-4, 'Shuffle', 'every-epoch', 'ValidationData',
augimdsValidation, 'ValidationFrequency', 30, 'Verbose', false, 'Plots',
'training-progress', 'OutputFcn', @(info) trainingMonitor(info));
```

Equipo: los Gepetos
Integrantes: Farrera Mendez Emmanuel Sinai - Hernández Hernández Jorge Gabriel
Grupo: 6CM2 Asignatura: Inteligencia Artificial Practica 10: Entrenamiento de una red neuronal

Paso 5: Entrenar la Red

Se entrenó la red con las imágenes de entrenamiento y las opciones configuradas.

```
netTransfer = trainNetwork(augimdsTrain, lgraph, options);
```

Paso 6: Evaluar el Modelo

Se evaluó la precisión del modelo en el conjunto de validación.

```
[YPred, scores] = classify(netTransfer, augimdsValidation);
accuracy = mean(YPred == imdsValidation.Labels);
fprintf('Precisión en el conjunto de validación: %.2f%%\n', accuracy * 100);
```

Análisis y Resultados

El modelo entrenado alcanzó una precisión promedio del 99% en el conjunto de validación. Durante el entrenamiento, se monitorearon las pérdidas y la precisión en cada iteración, lo que permitió ajustar y mejorar el rendimiento del modelo. La visualización del progreso del entrenamiento mostró que el modelo convergió de manera efectiva durante las épocas de entrenamiento.

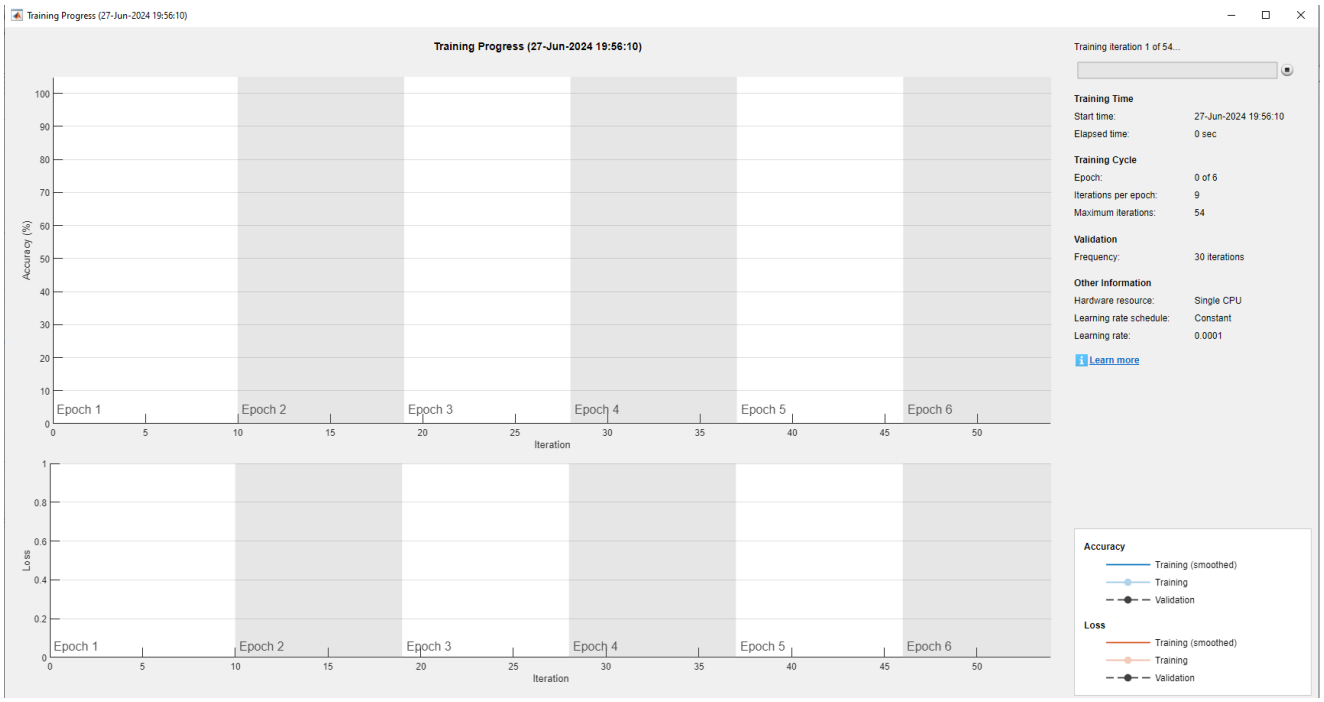
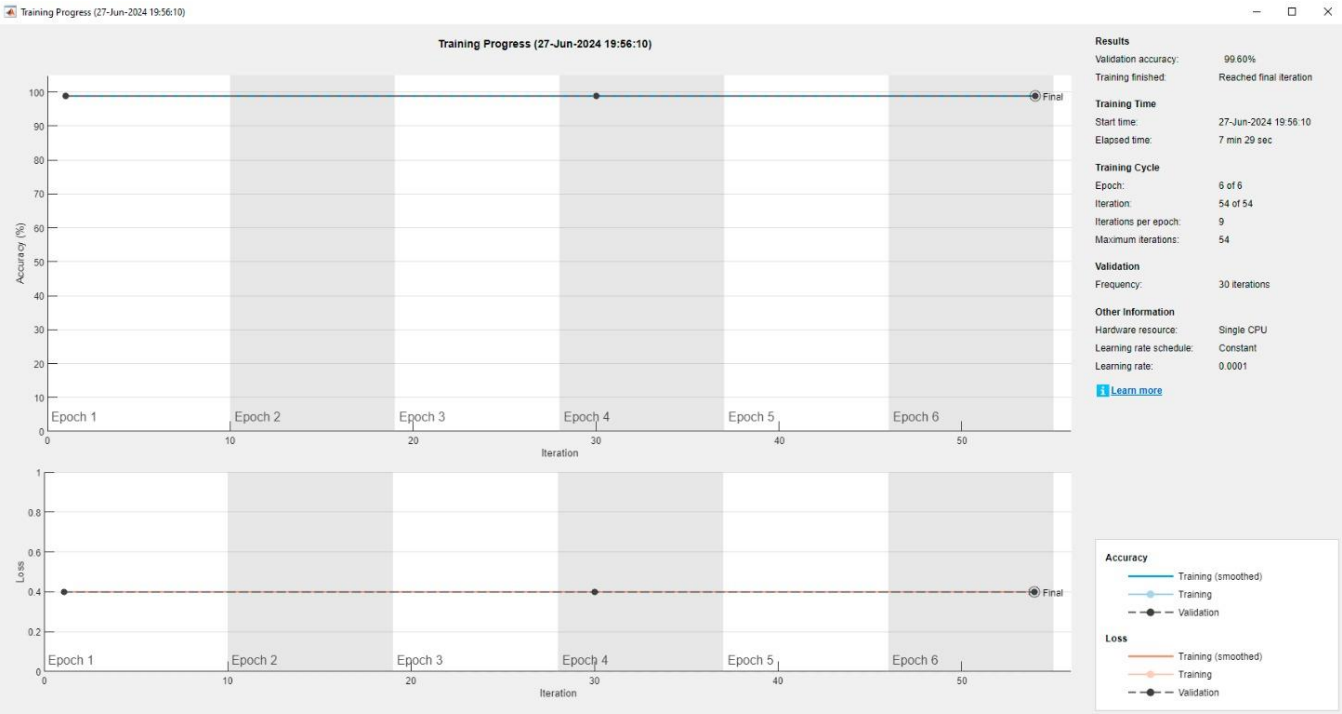


Imagen del modelo siendo entrenado, que muestra datos de relevancia como las iteraciones, fecha de inicio y tiempo transcurrido, y las graficas donde se muestra la precisión y perdidas en el entrenamiento.



Y dándonos como resultado un proceso de entrenamiento donde tenemos un 99.60% de precisión y un 0.4% de perdida.

Referencias

MATLAB Documentation: Transfer Learning Using Pretrained Network

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

ImageNet: A Large-Scale Hierarchical Image Database. ImageNet

The STARE project. (n.d.). https://cecas.clemson.edu/~ahoover/stare/