Wildfire Classification

Author: Anthony Ibarra

Abstract:

This document shows the capabilities of a CNN current potential to classify images of 228X228; and classify them as fire or no fire. In turn with the model from Neo Benedict (Benedict) which was adapted as a basis for this project.

Introduction:

During the total process of the project there were many hurdles to overcome while programming it on python using pytorch as the main NN software. At the start an extremely simple model was used for testing and caused the computer to freeze entirely, thus the source of an already well enough model for image classification.

In testing it was discovered that some images, from what I can recall less than 5 are different in dimension from the rest which are of 228X228.

Method description:

The model used was as stated by Benedict it is a CNN for image classification (Benedict). For the model itself a combination of 2d Convolution, 2d Max Pooling as layers, Linear, and Relu for the activation functions. Regarding the loss function CrossEntropy was used and Stochastic Gradient Descent for the optimizer. The model was adjusted to fit the classification of 228X228 images instead of the 5x5 images for Benedict's case.

Network is as follows:

```
NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=44944, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
```

```
  (fc3): Linear(in_features=84, out_features=2, bias=True)
)
```

Experiment results:

During the experiment only the accuracy and Avg loss were taken during testing. Overall results were quite good but not great. The model ran for 5 epochs with 64 as the batch size. Note the accuracy would have been higher if batch size was at 32 and epochs were also increased ex. 10 epoch; but my computer was not the fastest so compromise had to be made.

Results are as followed:

```
Epoch 1
-------------------------------
loss: 0.695756  [    64/ 8940]
loss: 0.583279  [ 6464/ 8940]
Test Error:
 Accuracy: 73.8%, Avg loss: 0.504840

Epoch 2
-------------------------------
loss: 0.591716  [    64/ 8940]
loss: 0.517298  [ 6464/ 8940]
Test Error:
 Accuracy: 77.7%, Avg loss: 0.456839

Epoch 3
-------------------------------
loss: 0.563735  [    64/ 8940]
loss: 0.466288  [ 6464/ 8940]
Test Error:
 Accuracy: 81.8%, Avg loss: 0.407514

Epoch 4
-------------------------------
loss: 0.465664  [    64/ 8940]
loss: 0.382732  [ 6464/ 8940]
Test Error:
 Accuracy: 84.6%, Avg loss: 0.334842
```

Epoch 5
----
loss: 0.260112 [64/ 8940]
loss: 0.338108 [6464/ 8940]
Test Error:
 Accuracy: 86.9%, Avg loss: 0.318536

References:

Benedict, Neo. “Building an Image Classification Model From Scratch Using PyTorch | by Benedict Neo | bitgrit Data Science Publication.” *Medium*, 22 April 2021, https://medium.com/bitgrit-data-science-publication/building-an-image-classification-model-with-pytorch-from-scratch-f10452073212. Accessed 28 April 2023.