Deep Learning: Finetune Resnet 18

Objective: Finetune Resnet 18 on Tiny ImageNet dataset with Cross-Entropy as the final classification loss function

Link of Google Colab Code File:

https://colab.research.google.com/drive/1Ns854EnAUXNu_KZBWWrh_ld-olRTe81h?usp=sharing

Steps Involved:

Due to time constraints, I downloaded the tiny imagenet dataset directly from the following link in the Colab and used that for analysis, instead of loading the file in the drive and using.

```
!wget http://cs231n.stanford.edu/tiny-imagenet-200.zip
```

1. Dataset Used: Tiny Imagenet dataset.

Tiny ImageNet is the subset of ImageNet dataset.The dataset contains

- 100,000 images
- 200 classes (500 for each class).
- Images are of size 64×64 colored and coloured.
- Each class has 500 training images, 50 validation images, and 50 test images.

2. Data Preprocessing:

2.1. Due to computation restraints have used the first 50 classes for our analysis.

- Train Dataset size: 25000
- Test Dataset size: 10000
- train_labels = trainData.classes train_images = trainData.imgs
- test_labels = testData.classes test_images = testData.imgs

2.2. Transformed the data with following operations

- `Resize((255,255)),`
- `CenterCrop(224),`
- `Convert to Tensor`
- `Normalize([0.5,0.5,0.5],[0.5,0.5,0.5])`

2.3. Created a Train and Test Data Loaders each of batch size 50.

2.4. Visualizing a batch of data

2.5. Moved data and labels to GPU if available.

3. Observation:

3.1. Pretrained Resnet18 model.

3.1.1. Accuracy on Pretrained Resnet18 model : 74.83%

3.2. Fine Tuned Resnet18 model

Freezed rest of the pretrained layers of the model and changed the final layer as

```
            Linear-68                  [-1, 128]          65,664
              ReLU-69                  [-1, 128]               0
            Linear-70                   [-1, 50]           6,450
================================================================
Total params: 11,248,626
Trainable params: 72,114
Non-trainable params: 11,176,512
----------------------------------------------------------------
```

3.2.1. Trained the fine tuned resnet18 model using

3.2.1.1. CrossEntropy as finally layer loss function

No. of Epochs	Loss Function	Optimizer	Training Accuracy
10	CrossEntropyLoss	Adam	76.204%

```
Epoch:0 Loss :1.8650309127569198 f1 Score :<function f1_score at 0x7fb716d94ef0> Accuracy :52.928%
Epoch:1 Loss :1.1486958615779876 f1 Score :<function f1_score at 0x7fb716d94ef0> Accuracy :67.364%
Epoch:2 Loss :1.0280085101127625 f1 Score :<function f1_score at 0x7fb716d94ef0> Accuracy :70.748%
Epoch:3 Loss :0.9683730068206787 f1 Score :<function f1_score at 0x7fb716d94ef0> Accuracy :72.21600000000001%
Epoch:4 Loss :0.9199486204385757 f1 Score :<function f1_score at 0x7fb716d94ef0> Accuracy :73.456%
Epoch:5 Loss :0.8928772776126862 f1 Score :<function f1_score at 0x7fb716d94ef0> Accuracy :74.024%
Epoch:6 Loss :0.8644607219696044 f1 Score :<function f1_score at 0x7fb716d94ef0> Accuracy :74.612%
Epoch:7 Loss :0.8428471851348877 f1 Score :<function f1_score at 0x7fb716d94ef0> Accuracy :75.28%
Epoch:8 Loss :0.8255913119912147 f1 Score :<function f1_score at 0x7fb716d94ef0> Accuracy :75.872%
Epoch:9 Loss :0.80792843657732 f1 Score :<function f1_score at 0x7fb716d94ef0> Accuracy :76.20400000000001%
Completed
```

3.2.1.2. Evaluation Metrics

i. Accuracy: 76.204%

ii. Confusion Matrix:

```
print(confusion_matrix)
```

```
tensor([[4.2220e+03, 1.3700e+02, 1.0000e+00,  ..., 3.2000e+01, 1.4000e+01,
         2.3000e+01],
        [1.1900e+02, 3.5510e+03, 9.6000e+01,  ..., 1.1000e+01, 6.0000e+00,
         4.2000e+01],
        [9.0000e+00, 9.5000e+01, 4.1780e+03,  ..., 5.9000e+01, 1.9000e+01,
         2.2000e+01],
        ...,
        [1.0000e+01, 3.0000e+00, 7.6000e+01,  ..., 3.4950e+03, 3.7500e+02,
         1.9000e+01],
        [3.0000e+00, 3.0000e+00, 2.2000e+01,  ..., 3.0500e+02, 4.1040e+03,
         7.0000e+00],
        [5.6000e+01, 6.0000e+01, 3.9000e+01,  ..., 5.3000e+01, 1.0000e+01,
         3.0990e+03]])
```

iii. Recall Score (Class wise)

```
Recall Score:  [0.8444 0.7102 0.8356 0.596  0.5676 0.7042 0.7088 0.7724 0.811  0.7472
 0.8964 0.5932 0.7698 0.8116 0.7314 0.6336 0.6608 0.5734 0.7224 0.8704
 0.7094 0.554  0.7318 0.5992 0.7444 0.7278 0.6804 0.807  0.603  0.708
 0.7016 0.48   0.6926 0.8014 0.5518 0.795  0.599  0.6938 0.6916 0.674
 0.9076 0.5994 0.8212 0.6294 0.7506 0.8612 0.82   0.699  0.8208 0.6198]
```

iv. Precision Score (Class wise)

```
Precision Score:  [0.8171086  0.6602826  0.77356046 0.62935585 0.60641026 0.67568606
 0.71610427 0.7630903  0.7802578  0.7446681  0.8615917  0.628523
 0.721868   0.7706039  0.7084463  0.6262107  0.66653216 0.60870486
 0.7360913  0.8664145  0.63794965 0.57588357 0.7293203  0.62546974
 0.76710635 0.7101874  0.6656232  0.8094283  0.6542969  0.7008513
 0.71885246 0.5328597  0.70789045 0.78430223 0.5993917  0.7815572
 0.62670016 0.6758231  0.7012776  0.6813587  0.8713518  0.6301514
 0.81339145 0.658506   0.75573903 0.82680494 0.8        0.695384
 0.7918194  0.64602876]
```

v. f1 Score = 2*(precision * recall)/(precision +recall)

```
# F1 score
F1 = 2 * (precision * recall) / (precision + recall)
print("F1 Score for the classes: ", F1)
```

```
F1 Score for the classes:  [0.8305301  0.6843322  0.80338436 0.6122239  0.5863636  0.68964845
 0.7124334  0.76771694 0.79533195 0.7459319  0.8786512  0.61035085
 0.7450639  0.79057086 0.7197403  0.62988365 0.6636537  0.59052527
 0.72918135 0.8684027  0.6717803  0.56472987 0.73055804 0.61205316
 0.7555827  0.71888584 0.6729305  0.80821234 0.627602   0.7044075
 0.71012145 0.5050505  0.70016176 0.79275894 0.57461214 0.78822136
 0.6125371  0.6846936  0.69640523 0.6776594  0.8891066  0.61439115
 0.8172771  0.6436241  0.7531607  0.843652   0.80987656 0.6971873
 0.8060493  0.6326426 ]
```