Final Project

Xiaolong Lin

November 29, 2023

1 Dataset

In this task, I used a dataset for animal classification[1], which was created from Google Images. This dataset consists of 5,400 animal images belonging to 90 different categories. Some of the animal categories include: antelope, badger, bat, bear, bee, beetle, bison, boar, butterfly, caterpillar, chimpanzee, and more. Examples of some samples are shown in Figure 1.

Figure 1: Some samples from animals dataset

In terms of data preprocessing, firstly, because the sizes of each image in the dataset are different, for the convenience of subsequent training, I resized all the images to 224×224. Afterward, I applied data augmentation techniques to all the images to prevent overfitting during training. Data augmentation includes random flipping, random cropping, and colorjitter. Data augmentation further increases the diversity of the dataset, allowing the model to learn more generalized features. Finally, I standardized all the images, making the distribution of samples in the dataset follow a standard normal distribution.

[1] https://www.cvmart.net/dataSets

Lastly, I divided the dataset into a training set and a test set in a 4:1 ratio. Test set is used to evaluate the performance of the trained model.

2 The Architectures of Neural Networks

2.1 ResNet

As for the convolutional neural networks, I implemented the ResNet model. ResNet is a convolutional neural network structure proposed by Kaiming He, aimed at alleviating the problem of network degradation when the network becomes deeper. In traditional deep neural networks, as the number of layers increases, the network becomes increasingly difficult to train due to the issues of gradient vanishing or exploding. ResNet addresses this problem by introducing skip connections. The illustration of the residual block is shown as Figure 2.

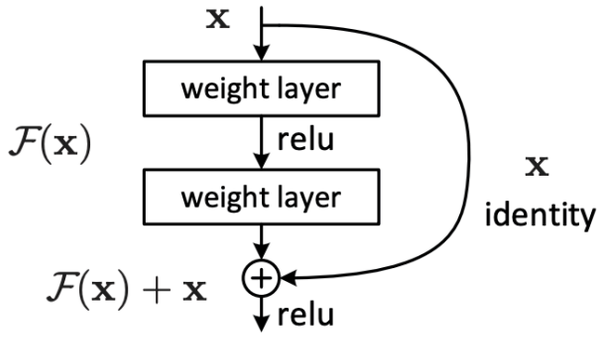

Figure 2: Residual Block

As for the version the ResNet, I implemented the ResNet34. The detail architecture is listed as following: the input image is first processed by a convolutional layer with a kernel size of 7×7, followed by a max-pooling layer for downsampling. After that, it goes through four sets of convolutional layers, each set consisting of several residual blocks. The number of residual blocks in these four sets are 3, 4, 6, and 3, with respective output channel numbers of 64, 128, 256, and 512. The size of the convolutional kernels is 3×3 for all of them. After processing through these layers of convolution, the extracted feature maps undergo global average pooling to obtain the final feature map. Subsequently, the final feature map is passed through a fully connected layer for classification.

2.2 Vision Transformer

Transformer is a neural network model proposed by Google in 2017. It does not utilise CNN modules or RNN modules. Instead, it introduces an attention mechanism module. Transformer consists of two parts: the Encoder module and the Decoder module. The Encoder module is primarily used to encode input information, while the Decoder module

decodes relevant output based on the information encoded by the Encoder module. The model structure is as shown in the Figure 3.

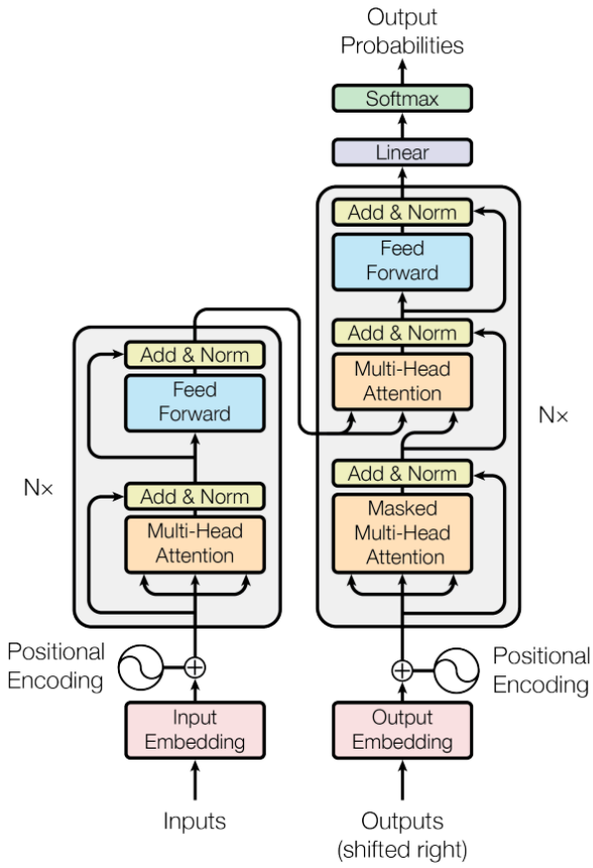

Figure 3: The architecture of Transformer

Vision Transformer (ViT) is a variant of the Transformer architecture primarily designed for processing visual tasks. The framework of ViT is as shown in the Figure 4. ViT utilizes the Encoder module of Transformer. The input images are initially divided into several 16×16 pixel patches, which are then flattened and fed into the Encoder module. The Encoder module consists of 12 layers of self-attention mechanism modules. Additionally, ViT introduces a special token called [cls] to aggregate global information from the input images. After processing through the Encoder module, the global information aggregated by [cls] is used for the final classification task.

In this assignment, I implemented the base version of the ViT model, which consists of 12 layers of self-attention mechanism modules with a hidden state dimension of 768.

3 Training Details

Both models are trained on an Nvidia Tesla V100. The batch size is set to 32, and the initial learning rate is set to 1e-2, with the stepwise learning rate decay mechanism. The loss function used for training is the cross-entropy loss. I use SGD as the optimizer for the models, with a momentum of 0.9 and a weight decay of $1e-3$. Each model is

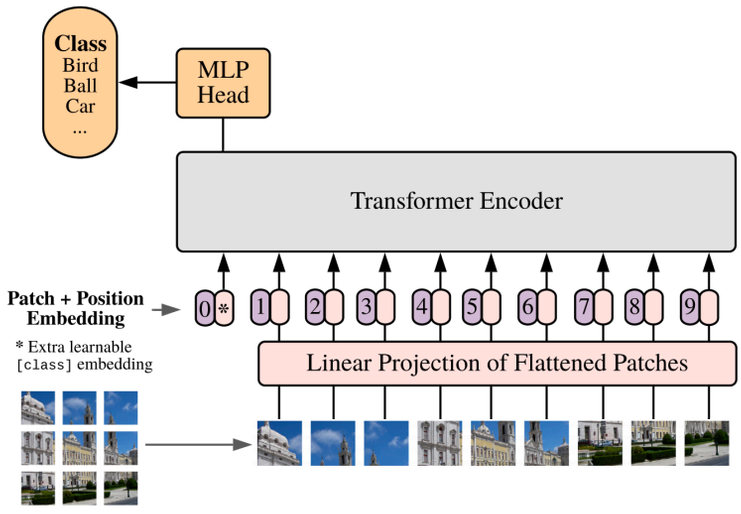

Figure 4: The architecture of ViT

trained for a total of 100 epochs, and evaluation on the test set is performed after each epoch of training.

4 Experiments Results

4.1 ResNet

The experimental results of the ResNet34 model are shown in the Figure 5. After 100 epochs of training, the model achieved an accuracy of around 56% on the test set. In addition, I investigated using a pre-trained ResNet34 model that had been pre-trained on the ImageNet dataset as the initial model. The experimental results after 100 epochs of training are also shown in the Figure 6. The accuracy on the test set was around 95%, which is significantly higher than the ResNet34 model without pretraining.

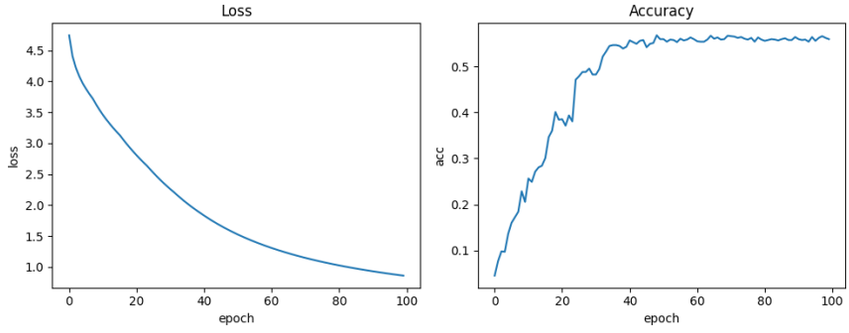

Figure 5: The result of ResNet

4.2 ViT

The experimental results of the ViT model are shown in the Figure 7. After 100 epochs of training, the model achieved an accuracy of around 32% on the test set. We can observe that when pretrained weights are not used, the performance of the ViT model

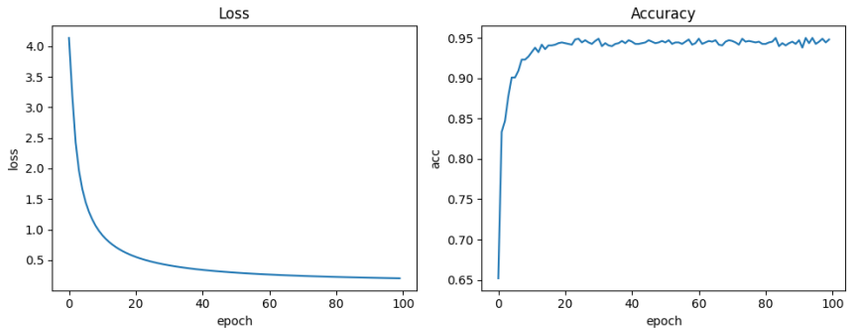

Figure 6: The result of ResNet using pretrained weights

is significantly lower than that of ResNet34. Subsequently, I used a ViT model pretrained on the ImageNet dataset as the initial model, and the experimental results after 100 epochs of training are shown in the Figure 8. We can see that using pretrained weights resulted in a significant improvement in the accuracy of the ViT model, with an accuracy of around 98% on the test set. Furthermore, after just one epoch of training, the model already achieved a very high accuracy.

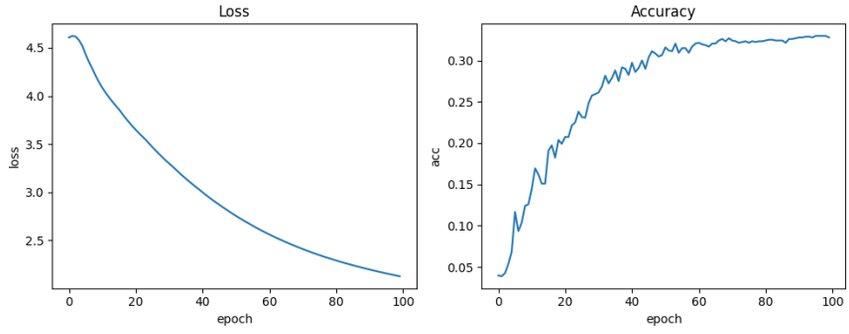

Figure 7: The result of ViT

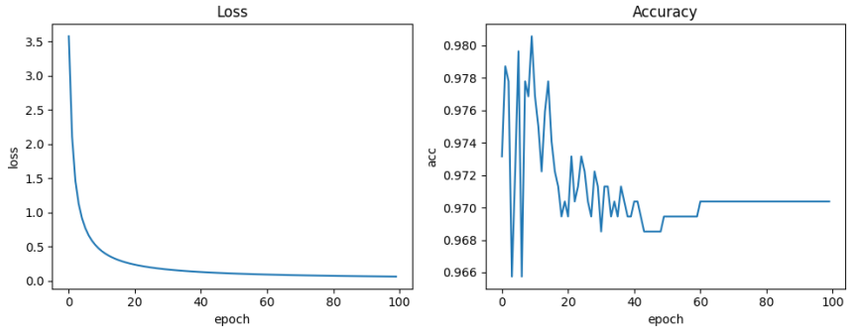

Figure 8: The result of ViT using pretrained weights

4.3 Conclusions

Based on the above experimental results, we can get the following observations:

- Using pretrained weights significantly improves the model's performance. Pretrained

weights help the model obtain a better initialization, thereby reducing the risk of getting stuck in local optima during training.

- When pretrained weights are not used, the performance of the ViT model is inferior to the ResNet34 model without using pretrained weights. However, when pre-trained weights are applied, the ViT model outperforms ResNet34. While the ViT model exhibits strong feature extraction capabilities, it comes with a larger number of parameters. Without a good initialization, it becomes challenging to achieve favorable results in subsequent training.

5 Challenges

Challenge 1 After training, the model's accuracy was not satisfactory.
Solution 1 After consulting relevant blogs and materials, I initialized the model's parameters with the weights pretrained on the ImageNet dataset for the following training. This approach led to the model achieving a better accuracy.

Challenge 2 The models overfit during training.
Solution 2 To address this, I introduce the Dropout mechanism and apply three different data augmentation techniques to the dataset. As a result, these methods alleviate the overfitting issue to some extent.

Challenge 3 The ViT's structure is complex, and I don't how to implement it in the initial stage.
Solution 3 I referenced some code on GitHub, and after understanding the code, I try to make modifications to certain parts. Eventually, I successfully implemented the ViT model.