AI Generated Image Detection

Zheng Shouwen

21097982d

COMP4423 - Assignment 2

***Abstract*— Due to the diffuse AI generation of text2image and image2image. deep Fack has become a means to seriously affect computer ethics. And one of the main purposes of AI detection techniques is to prevent the abuse of AI generation. At present, this field has become a popular cutting edge machine learning field.**

Keywords—ResNet, Deep Fack, CNN, AI Detection, Image Processing

Introduction

In this project, residual networks are mainly used. At the same time, in order to obtain clear implementation logic, some complex CV Tasks are decomposed. Through verification on the validation set and evaluation on the unseen test set, the excellent performance and robustness of this model are fully demonstrated. Some visual evaluation methods such as dynamic loss tracking, ROC curve verification, and confusion matrix are also used in the project to increase the understanding of the model.

During the process, some challenges also appeared frequently, such as tensor storage exceeding the scope of the graphics card, Epoch adjustment issues, and training set division and preprocessing issues. Through improvement and learning, these challenges that emerged were successfully resolved.

This article will be divided into four parts: the first part will introduce the decomposed CV tasks, the second part will explain in detail all the steps and algorithm analysis, the third part will evaluate the model, and finally the main challenges and solutions in the project will be explained. plan.

Requirements tasks analysis

The given goal is to differentiate between real-world pictures captured on campus and AI-generated images of corresponding architectures. I will divide it into several important tasks：

Collection and expansion of data sets

Expand the data set using the given folder image as the basic standard. Store data in jsonl in the format: file name + annotation.

Preprocessing data sets

Create a list of all images and annotations. The image_inf class stores basic image data (index, file_name, label, file_path). At the same time, the RGB images are processed to obtain a tensor list and divide the data set train/validation/test. At the same time, operations such as resizing the image, normalizing pixel values, and balancing categories in the data set are performed.

Model establishment and iterative tuning

Pay attention to the optimization issues of ResNet, adjust the hyperparameters and the number of convolution kernels in different layers, and perform data enhancement. At the same time, backpropagation is performed to update the weights and biases of the model (methods such as cross-validation)

Model evaluation and deployment

Use the training set and unseen test set to evaluate the performance of the model, calculate and visualize the model's accuracy, ROC, loss change, and confusion matrix to measure its classification performance.

Bonus items

Identify AI-generated picture styles: surrealism, comic style, old black and white photos. Relabel these labels and position them as a 4-class classification problem to adjust the model again.

Divide this abstract goal into a series of general processes. When training CNN, these are the basic steps, and they also require me to tune and innovate solutions to deal with subsequent challenges and improve the performance of the model.

Algorithm steps and implementation

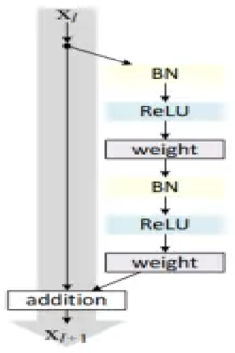
For reading data, read the data in jsonl line by line to form a list with information.

In order to ensure the robustness of the model in real scenes, I added scenes with different environments and lighting to the data set, and tried to balance and expand the various styles of AI-generated pictures.

Process this list to find the corresponding photo path, and it is more convenient to use numbers instead of annotations. List the photos and annotations separately, and then convert them into nparray form. Preprocess and adjust photos through CV2 conversion.

Divide the two lists (label and image) into training sets, validation sets and test sets, convert nparray into tensors, and perform normalization at the same time. It should be noted that TensorDataset and DataLoader are called at this time to prevent memory overflow.

The basic principle of the ResNet model used is to process the output of the previous layer through BN processing and ReLu processing, and then add the weights to obtain the input of the next layer. This will ensure that features will not be lost due to convolution.



(Figure1. ResNet Process)

The adopted ViT processes the image into a sequence of patches and uses a converter to capture global dependencies. Usually performed after ResNet convolution

In the Bonus part, I use four classification tasks (0: realistic pictures, 1: AI-anime style, 2: AI-surreal style, 3: AI-old photo style) while modifying the fully connected layer (fc), the output layer size to ensure the model is normal.

Evaluation model

After multiple builds, the accuracy of the model on the validation set is: 96.15%; the accuracy on the unseen test set is: The following are the scores obtained by different model optimization methods:

Model	Validation_Set	Test_set
Res_LowData	71.34%	66.52%
Res_BigData	92.31%	90.72%
Vit_BigData	51.92%	45.17%
Best_Opt	96.15%	93.02%
Bonus_Part	93.72%	89.67%

(Tabel 1. Model Accurancy)

I summarize the models I tried and give four different situations:

- Case 1: Res_LowData, the first time to train, the data set is the given default data set, the amount of data is small.
- Case 2: Res_BigData, the data set is an extended data set, with about 300 jobs and a large amount of data.
- Case 3: Vit_BigData, the data set is an expanded data set, and Vit is used for layer filtering to implement the attention mechanism.
- Case 4: Best_Opt, performs feature selection and hyperPare- tuning, and performs tensor normalization at the same time.
- Case 5: Bonus_Part, which is trained in the final Bonus Part.

It is worth noting that in the case of Vit, the individual expects the situation to be more accurate than the second time. But on the contrary, there is a big reduction. I found that the main reason is that the data set is not large enough. Due to the lack of inductive bias limitations of the transformer, the migration effect in downstream tasks is not obvious. There will be more specific explanations in subsequent chapters.

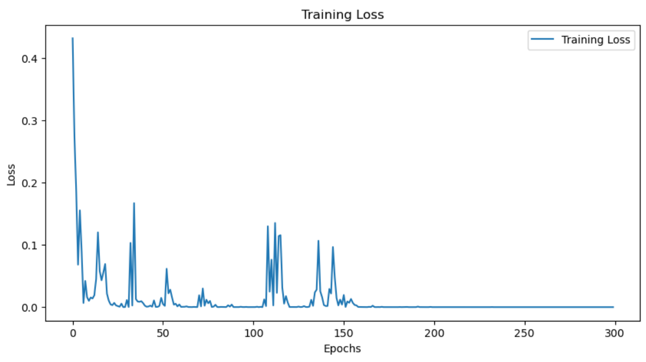
Regarding the training speed of the model, I also summarized:

Case	**Time**
Low_Data	15m 03s
Big_Data	89m 42s
Opt_Para	36m 43s
Bonus_part	42m 34s

(Tabel 2. Model Training time)

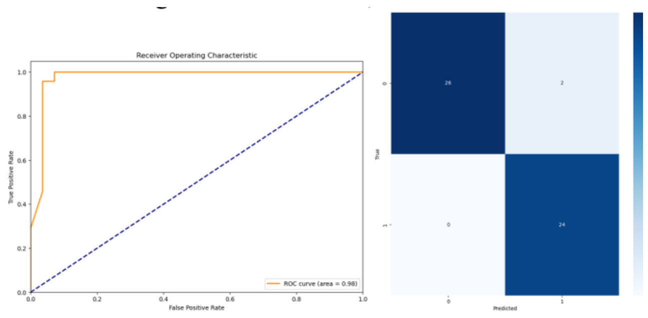
- Case 1: Low_Data, Fast when the data set is small.
- Case 2: Big_Data, When the data set is large, the time increases greatly.
- Case 3: Opt_Para, Increase the number of batches and use mixed precision and scaling loss in each epoch to reduce the time, there is a significant improvement.
- Case 4: Bonus_part, which is trained in the final Bonus Part

During training, we can see that Loss gradually decreases with each Epoch, and finally approaches 0.0002, indicating that the model is gradually optimizing, and at the same time, there are no obvious abnormalities overall, indicating that the anti-overfitting effect is excellent, and the model is robust.



(Figure2. Loss changing diagram)

At the same time, the ROC curve can also illustrate the excellent classification ability of the model, and the confusion matrix illustrates the data distribution in the validation set.



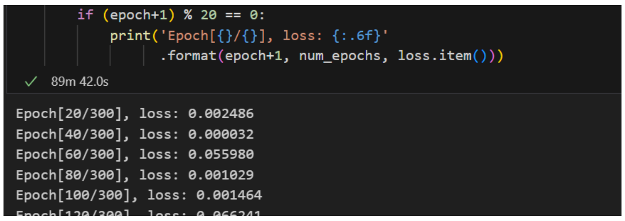
(Figure3. AUC and Confusion Mertrix)

Challenges and Solutions

During the process, the most interesting problem encountered was about tensor storage. Since all the preprocessing information is converted into tensors and stored in the CPU for the first time, an error is reported:

Through reflection and information review, DataLoader can be used to convert tensors in batches, and the problem is solved.

Secondly, the model construction time is too long, and it takes more than 1 hour to process a huge data set:



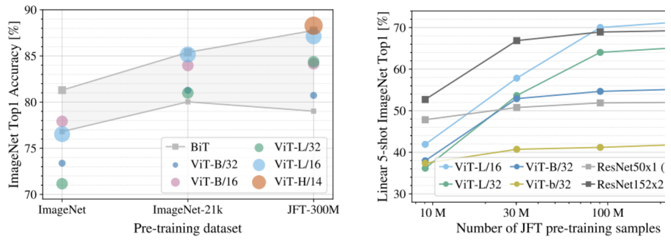
```
    if (epoch+1) % 20 == 0:
        print('Epoch[{}/{}], loss: {:.6f}'
              .format(epoch+1, num_epochs, loss.item()))
✓ 89m 42.0s
Epoch[20/300], loss: 0.002486
Epoch[40/300], loss: 0.000032
Epoch[60/300], loss: 0.055980
Epoch[80/300], loss: 0.001029
Epoch[100/300], loss: 0.001464
```

(Figure4. Training time)

I found that I can reduce the time by changing the batch parameters and adding feature scaling and mixed precision operations in epochs.

Regarding the processing of the image data set, at the beginning, the given image was damaged and needed to be repaired. At the same time, the collected photos were too large (more than 7Mb), so the data set was optimized through image compression and image repair.

Another problem is that after using ViT, the accuracy of the verification set decreases. By reading the paper, we can find that in the case of low data sets, the ResNet pre-trained model performs better, proving the effectiveness of ResNet inductive bias. However, when the data set increases, the inductive bias loses its advantage compared with Transformer, not even an inductive bias. So, I have reason to believe that this result is reasonable.



(Figure5. Vit and ResNet Compare)

Conclution and Reflection

In this work, I first divided the required goals into a series of small tasks to reduce the complexity of the tasks.

By using the ResNet algorithm for model training, we are also trying to train the ViT model. Finally, through multiple model optimizations and tests, an optimal model was finally obtained.

I evaluated this model on the validation set and unseen test set, and conducted visual analysis, and finally concluded that the performance of the model was superior.

In the Bonus part, I performed 4 classifications by adjusting parameters and annotation, and finally obtained an accuracy of 89.67%.

In this assignment, I gained a valuable opportunity to complete image prediction and CNN construction, which allowed me to understand the cutting-edge information of this subject, and at the same time cultivated my ability to read papers. I have a deep understanding of the fields of AI and CV. Development is full of confidence.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," arXiv, 2010.11929, 2021.*)*

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv, 1512.03385, 2015.*)*