Exploiting Convolutional Neural Networks for Galaxy Classification

Chao Gao *

School of Computer Engineering and Science Shanghai University, Shanghai, 200444, China

* Corresponding Author Email: 319468363@shu.edu.cn

Abstract. Galaxy classification is an important work in the field of astronomy. The efficiency of manually solving this problem is too low, and deep learning provides many methods for image classification. Using deep learning methods to automatically classify galaxies can greatly improve the efficiency of this task. Therefore, this paper uses mature Convolutional Neural Networks (CNN) models and self-designed simple CNN models, and compares their performance on the Galaxy10 DECals Dataset. In the experiment, the Inception network performed best, with an accuracy rate of 0.63 on the test set, but the training time was as long as about 2 hours. The simple CNN model ranked second, with an accuracy rate of 0.54, and the training time was only 20 minutes. The accuracy of other models is about 0.5, and the training time also takes 2 hours. The result performance of the model is worse than expected, which may be due to the less data used. The result of Inception is much ahead, and its solution to the problem provides a solution for the task of dealing with small data sets.

Keywords: Convolutional neural network; galaxy classification; deep learning.

1. Introduction

Galaxy classification is a significant task in astronomy [1]. The classification of tens of thousands of galaxy images is a long and monotonous process. In this case, a method that can automatically identify and classify galaxy images appears particularly important.

From the perspective of computer vision, this problem is essentially a sub-task of image classification. Now there are many mature deep learning models in this field. Here five mainstream image classification models are leveraged: Visual Geometry Group Network (VGGNet) [2], Deep Residual Network (ResNet) [3], Inception [4], Dense Convolutional Network (DenseNet) [5] and Mobile Network (MobileNet) [6].

VGGNet was created by DeepMind with the Oxford University Virtual Geometry Group. It has three fully connected layers, a Softmax output layer, and five convolutional layers (varying in the number of sub-layers). Max pooling is used to divide the layers. The ReLU function is used by the activation unit of every hidden layer. VGGNet won the second place in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition [7].

ResNet was proposed by Microsoft Research Institute. A neural network of 152 layers that was successfully trained and won the 2015 ILSVRC competition was created using ResNet Unit. The VGG19 network serves as the foundation for the modification of the ResNet network, and the short-circuit method is used to add the residual unit.

An essential turning point in the development of Convolutional Neural Networks (CNN) classifiers is the Inception Network. Prior to the creation of Inception, the majority of well-known CNNs simply stacked convolutional layers on top of one another to create a deeper network in an attempt to improve performance. With the help of several convolution or pooling operations carried out in parallel on the input image, Inception creates a network with an outstanding local topology that stitches together all of the output results to create an extremely detailed feature map.

When the number of layers of CNN becomes deeper, the path from the output to the input will become longer, which will lead to a problem: the gradient is likely to disappear when it propagates back to the input through such a long path. Is there a way to make the network deep gradient and not disappear? DenseNet proposes a very simple method. DenseNet solves this problem by directly

establishing a dense connection between all the previous layers and the following layers to reuse the features.

MobileNet is a classic network of lightweight networks. Deep separable convolution is its primary innovation, and the network as a whole is made up of stacks of deep detachable modules. On this basis, it improves the computational efficiency of the convolution layer, thereby reducing the time-consuming operation of the model and ensuring that the accuracy is not greatly affected.

The task of classifying galaxies can be tackled using these five models. In reality, some people have successfully solved these kinds of issues with DenseNet. They compared the performance of several pre-trained CNN models using the same dataset. As a result, DenseNet121 performs better than the other models and, in less than 30 minutes, reaches an approximate test-set accuracy of 0.89.

However, it is believed that such results are not universal and general. This work designs a simple CNN model and compare it with the above five CNN models without pre-training weights on the same data set. Inception performs best, followed by the designed simple CNN model. Accurately speaking, except that Inception is more advanced in accuracy, the performance of other models is not much different. It is believed that there is a certain upper limit on the performance of the CNN model in a specific application scenario, which is determined by its own structure.

2. Method

2.1. Dataset

The data set used in the experiment is Galaxy10 DECals Dataset, which contains more than 17,000 labeled RGB images of ten galaxies. Due to the limited memory of the experimental device, this work only extracted 3340 samples (334 samples per class) as data sets [8].

The structure of the CNN model is shown in Fig 1 and contains 7 convolutional layers, 7 pooling layers and 2 fully connected layers. All hidden layers use ReLU as the activation function. The model is compiled by choosing Adam as the optimizer and categorical crossentropy as the loss function. This work also added Batch Normalization and Dropout to the model to reduce overfitting and improve performance [9, 10].

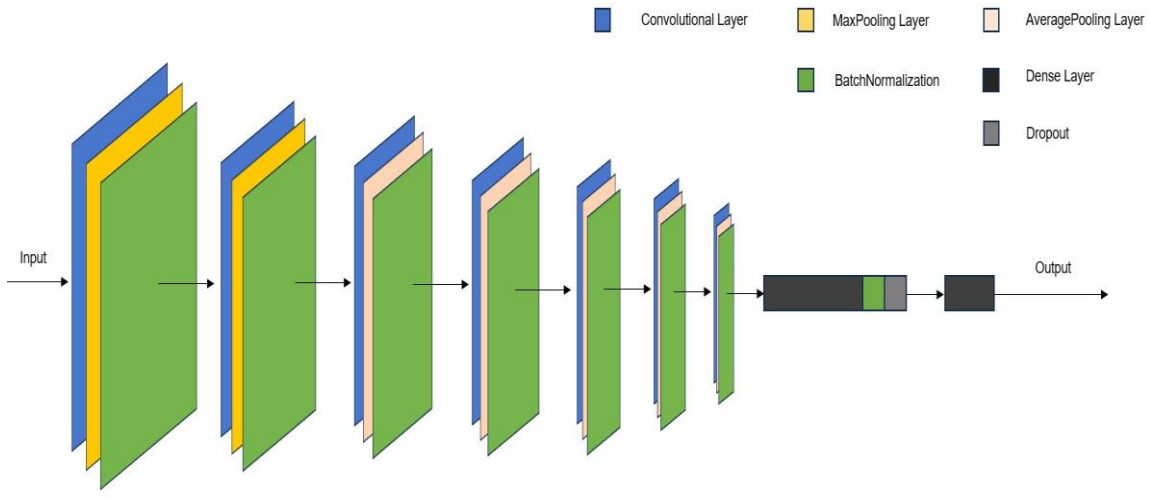

Fig 1. Architecture of CNN model (Figure Credits: Original).

2.2. Convolutional Layers

Mathematically, the definition of convolution is as follows: Let f(x) and g(x) be two integrable functions on R, and let

$$h(x) = \int_{-\infty}^{\infty} f(\tau)g(x-\tau)d\tau \quad (1)$$

Be the integral. It can be demonstrated that the aforementioned integrals exist for nearly all real values x. This integral thus defines a new function h(x), designated the convolution of function f and g, with different values of x, represented by $h(x) = (f \cdot g)(x)$.

The information of the picture is stored by a two-dimensional matrix, and the convolution operation is performed on a picture, which is a convolution of the two-dimensional matrix representing the picture. The values stored in the matrix are discrete, so this work should consider discrete numerical convolution. Discrete convolution is a special operation between two discrete sequences according to certain rules to multiply and add their relevant sequence values respectively.

Convolution operations in computer vision functions similarly to sliding window operations. A tiny window is slid over the input image to form a new feature map, or convolution feature. At each place, the dot product of the pixel value in the window and the convolution nuclear, or filter, is computed. Different convolution kernels correlate to adjacent feature maps, and various convolution cores can extract various properties, including forms, edges, colors, and so forth.

Due to gradient descent and back propagation, deep learning does not require manual design of convolution kernels. In the convolutional neural network, the convolution kernel is some random values at the time of initialization. After training, the kernel matrix can be learned. In other words, the convolution kernel is some parameters that can be trained by machine learning.

This is exactly the work to be done by the convolutional layer. In the CNN model, it is expected to increase the number of convolution layers and filters per volume as much as possible to extract as many features of the image as possible.

2.3. Pooling Layers

One of the most often utilized parts of the convolutional neural network nowadays is the pooling layer. Its function is to extract higher-level visual information by reducing the dimension of the data. Moreover, applying the pooling layer can minimize computation, lower the number of model parameters, and avoid overfitting. The model employs both average and maximum pooling.

After pooling the region, the maximum value in the picture region is chosen as the value in the forward step. The gradient propagates back through the forward process's greatest value in the backward process, leaving the gradient at other points at zero. This is the maximum pooling process; the mean pooling process involves substituting the mean value for the maximum value in the preceding phase.

Maximum pooling has the benefit of being able to recognize the image's edge and texture structure. Mean pooling has the benefit of lowering the estimated mean's deviation and enhancing the model's resilience. Consequently, the mean optimization is used in the back to guarantee model stability, while the maximum pooling layer is used in the front to learn the edge and texture structure of the original image.

2.4. Fully Connected Layers

The principle of the fully connected layer is very simple. It is connected to each neuron in the previous layer and connected to each neuron in the latter layer, creating a dense connection structure. This means that each neuron is connected to the entire adjacent layer of neurons. These connections are controlled by weights and bias parameters.

Different aspects of the input data are extracted by the convolutional layer and the pooling layer. The fully connected layer integrates these features and combines them into a higher-level representation. The first fully connected layer contains many learnable parameters, which are adjusted by back propagation and gradient descent during training, so that the network can adapt to training data and learn complex data relationships. The second fully connected layer is the output layer. The integrated features are passed to the activation function Softmax to generate the scores of each category, and then these scores are converted into category probabilities to determine which category the input data belongs to.

2.5. Mainstream CNN Models

A traditional convolutional neural network architecture is called VGG. The main concept is to use several small-sized convolution kernels and pooling layers to build a very deep network topology. VGG's primary contribution is demonstrating that network depth can enhance performance and serve as an inspiration for later convolutional neural network architecture.

The main goal of ResNet is to provide residual connections, or jump connections, to address the issues of gradient disappearance and explosion in deep networks. The network is easier to train and optimize because to the residual connection, which enables information to skip some layers directly. The main advancement of ResNet is the addition of residual blocks, which enable the network to effortlessly traverse hundreds or even thousands of layers of depth.

Using several convolution kernels of varying sizes and convolution layer types (e.g., 1x1, 3x3, 5x5 convolution, etc.) to collect features at various scales and abstract levels is the fundamental idea behind Inception. Enhancing the model's expressiveness and performance without sacrificing its computing economy is the design objective of Inception.

MobileNet is intended for situations when resources are limited, like embedded and mobile devices. Its main concept is to minimize computation and parameter count by employing depthwise separable convolution. By breaking down the ordinary convolution into deep convolution and point-by-point convolution, deep separable convolution dramatically lowers computing costs without sacrificing performance.

The key thought of Inception is to use multiple convolution kernels of different sizes and different types of convolution layers (such as 1x1,3x3,5x5 convolution, etc.) to capture features of different scales and abstract levels. The design goal of Inception is to increase the expression capacity and performance of the model while maintaining the computational efficiency of the model.

3. Result

3.1. Evauluation Indexes

This paper used a total of six models to experiment on the same data set, and compared the experimental results in three aspects: precision score, recall score, and f1 score. These three parameters are important indicators for evaluating classification tasks.

The percentage of real positive predictions among all anticipated positive instances is known as precision. Stated differently, it assesses how well the model predicts the favorable outcomes. The precision score is calculated as:

$$Precision = \frac{TruePositives}{TruePositives+FalsePositives} \tag{2}$$

Where the quantity of successfully anticipated positive cases is called TruePositives, and the quantity of falsely predicted positive instances is called FalsePositives.

Recall score evaluates how well a model can recognize positive examples within a given dataset. The reacall score is calculated as:

$$Recall = \frac{TruePositives}{TruePositives+FalseNegatives} \tag{3}$$

Where FalseNegatives are the number of cases that are mistakenly labeled as negative when they should have been classed as positive, and TruePositives are the number of correctly predicted positive instances.

The F1 score is a trustworthy indicator of a model's accuracy since it takes recall and precision into account. The harmonic mean of recall and precision is used to calculate it.

$$F1 = \frac{2\times Precision\times Recall}{Precision+Recall} \tag{4}$$

3.2. Result Comparison

It can be seen from Tabe 1 that the precision score, recall score and f1 score of the Inception network are the highest among all models. The performance of the simple CNN model is second only to Inception. The remaining models perform poorly, but are not far from the simple CNN model. At the same time, mobilenet performed worst among these models, and none of its three indicators exceeds 0.5.

Table 1. Comparison table of precision score, recall score, and f1 score of six models.

Model	Precision Score	Recall Score	F1 Score
VGGNet	0.5030	0.4860	0.4900
ResNet	0.5359	0.5533	0.5293
Inception	0.6287	0.6254	0.6003
DenseNet	0.5060	0.5309	0.4857
MobileNet	0.4880	0.4868	0.4874
The Proposed CNN model	0.5419	0.5472	0.5306
Model	Precision Score	Recall Score	F1 Score

Out of the five established models, only MobileNet is lightweight; nonetheless, it is also challenging to attribute this issue to model sizing. By adding more convolution layer filters and fully connected layer neurons to the basic CNN model, this work enhances the parameter scale. Additionally, a comparative experiment with the original CNN model is conducted. Simply said, there is one difference between the light and heavy models: the former has 32 neurons in each layer, while the latter has 256 neurons in each layer. Table 2 displays the precision, recall, and f1 scores for the two. It is evident that while other factors stay the same, increasing the model's size has no positive effect on its performance.

Table 2. Performance comparison of CNN models with different parameter scales.

Model	Precision Score	Recall Score	F1 Score
Lighte CNN Model	0.5419	0.5472	0.5306
Heavy CNN Model	0.4671	0.4535	0.4346

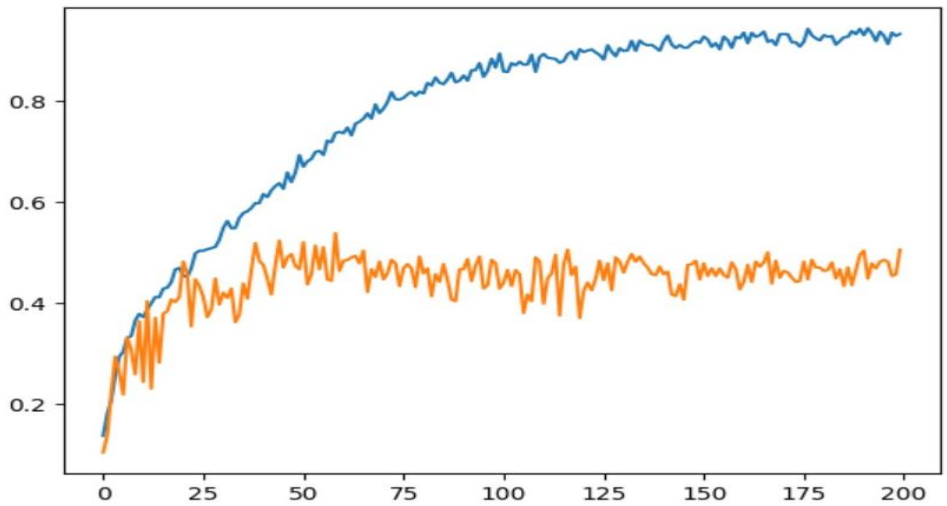

Fig 2. The accuracy curve of light CNN model, where the blue line represents the curve of the training set, and the yellow line represents the one of validation set (Figure Credits: Original).

However, the larger the parameter size of the model, the faster the convergence speed of the model. This can be seen through the comparison of the accuracy curves in Fig 2 and Fig 3.

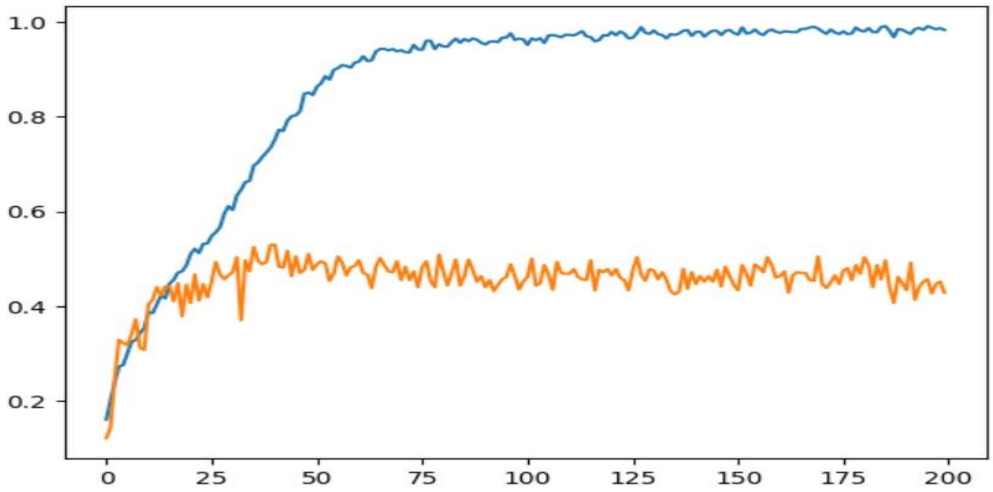

Fig 3. The accuracy curve of heavy CNN model, where he blue line represents the curve of the training set, and the yellow line represents the one of validation set (Figure Credits: Original).

4. Discussion

In the experiment, the biggest factor limiting the performance of the model is data. Due to the limitation of equipment, the amount of data for each training model cannot exceed 4000. This work selected 3340 images from more than 17,000 images, with only 334 samples for each type of galaxy. This level of data is still too little for these mature heavyweight models. If all the data can be used for training, they will perform better. This is an important reason why this work get different conclusions from predecessors.

However, even in the case of few data sets, the Inception model can still stand out. Its precision score, recall score and f1 score all reach more than 0.6, which is what other models fail to do. This may be because each convolution operation in the galaxy image obtains too little information, and Inception stitches the result matrix of different convolution layers to form a deeper matrix, thus extracting more useful information. The ability of other networks to extract information from a small amount of data is obviously better than that of the Inception network, which also provides an idea of optimizing the model when this work uses the CNN model to process data such as the galaxy image, which itself contains less information. That is to find ways to enhance the convolution operation of the extracted information.

In fact, the author still used the InceptionResNet model combined with Inception and ResNet to train, and its accuracy reached 0.68. However, because the two they have different focuses on optimizing the CNN model, the effect of their combination should be better, so it is not put into the comparative experiment.

In addition, this research has also made a lot of efforts in the adjustment and testing of simple CNN models. This work also imitates the structure of those mature models, increase the convolution layer and reduce the size of the convolution kernel as much as possible, try different activation functions, and finally barely get ahead of other networks except Inception in terms of accuracy. However, the convergence rate of the simple CNN model is significantly lower than that of the mature models. From the experimental results mentioned above, it can be inferred that it may be because the parameter size is not large enough.

The training time of the simple CNN model is still very fast, about 20 minutes. However, those heavyweight models take too much time and take about 2 hours to complete. The problem of overfitting has always existed, but it is not worth the loss to end the training as soon as possible just for the problem of over-fitting of the results. This work has tested it with a simple CNN model. It ends too early, the model convergence level is too low, and the accuracy rate is lower.

5. Conclusion

In the galaxy image classification task, a simple CNN model can achieve the same effect as most heavyweight models without pre-trained parameters. When processing data with less information in the sample, the Inception network has achieved excellent results by stitching different convolutional layer result matrices to enhance the model's ability to extract information. The upper bound of the CNN model is not to increase the complexity of the network, but to optimize some operations of the model for specific problems.

There are two ideas for the optimization of the model. One is to refer to the design idea of Inception, and try to add components to enhance the extracted features in the structure of the model. Like the Inception splicing feature matrix, a suitable splicing scheme can be explored through many experiments. The second is to perform special processing on the processed data. If a reasonable transformation mapping method can be found, like Fourier transform, the digital matrix of the image can be transformed to obtain more feature extraction materials for training, there may be better results. This research will also explore these two paths and strive to achieve good results.

References

[1] Baqui, P. O., V. Marra, L. Casarini, R. Angulo, L. A. Diaz-Garcia, C. Hernández-Monteagudo, P. A. A. Lopes et al. The miniJPAS survey: star-galaxy classification using machine learning. Astronomy & Astrophysics, 2021, 645: A87.

[2] Simonyan, Karen, and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014: arXiv preprint arXiv: 1409. 1556.

[3] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016: 770-778.

[4] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, 1-9.

[5] Huang, GAO, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, 4700-4708.

[6] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2017, arXiv preprint arXiv: 1704.04861.

[7] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, 2019, 248-255.

[8] Galaxy10 DECals Dataset, URL: https://github.com/henrysky/Galaxy10, Last Accessed: 2023/11/16.

[9] Gu, Jiuxiang, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu et al. Recent advances in convolutional neural networks. Pattern recognition, 2018, 77: 354-377.

[10] Yamashita, Rikiya, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. Insights into imaging, 2018, 9: 611-629.