An Image Classification Approach based on Deep Learning and Transfer Learning

Kaixu Han¹, Jinxin He¹, Yongzhi Wang²*, Yue Xiong¹, Chi Zhang¹

¹College of Earth Sciences, Jilin University, 130061 China
²College of Geo-exploration Science and Technology, Jilin University, 130061 China

*Corresponding author: iamwangyongzhi@126.com

Abstract: Target recognition of high-resolution images is an important direction of today's classification technology, and some classification models have emerged, but there are still many technical problems to be solved. The main source of this paper is the Google Photo Collection, which includes five types of city daisy, rose, tulip, dandelion and sunflower. Using CNN (Convolutional Neural Networks) deep learning model and Google's Inception transfer learning model to train and classify the sample images, the final accuracy of the overall test set can reach 88.3%; However, the accuracy of training without transfer learning is only 60.2%. Thus, it is more efficient to combine deep learning with transfer learning to image classifications.

1. Introduction

Image classification based on all kinds of machine learning models is one of the most hottest topics during the recent years. However, selecting features manually cost too much time! Hence, can we achieve automatic selections of features? The answer is yes! Deep Learning is used to achieve this. The essence of deep learning is to improve the accuracy of image recognition or classification by constructing multiple hidden layer models and massive training data (which can be unlabeled data) to select useful features. In general, the more features are selected during recognition, the more detailed the information is, and the better accuracy of recognition. Nevertheless, as the selected features increase, the computational complexity increases and it will be sparse on each feature. Therefore, a lot of features are not always perfect, and it is necessary to know how many features determined to select (Parkhi, Vedaldi & Zisserman, 2015).

Transfer learning is to migrate the learned model parameters to the new model to help its training. Considering that most of the data or tasks are related, through the transfer learning, the learned model parameters can be shared to the new model in some way to speed up and optimize its learning efficiency. Assuming that the two domains are similar, they share a similar relationship and apply the...
logical network relationship in the source domain to the target domain for migration, such as the migration of biological viruses to computer virus propagation. Yang and Newsam(2011), Cheriyadat(2013), Pan and Yang (2010) explained the development history and challenges of transfer learning in detail; Lin et al (2015) and Schmidhuber (2014), Liu Chen et al (2018) used the large samples and transfer learning model to pre-train the CNN (Convolutional Neural Networks); then replace the full connection layer with the over-limit learning machine, and finally used the small sample to train the CNN model, which shortened the training time and improved the classification accuracy; Under the CNN model, Li Guandong et al (2019) used transfer learning to improve the accuracy of scene classification recognition; therefore, transfer learning can solve the problem of too few sample data (Lin & Chen, 2018).

Thus, this paper combines the Google's Inception v3 transfer learning model with CNN deep learning model to train and classify some sampling images; then comparing the final results with only using CNN. The remain part of this paper is divided into four parts: the second part introduces the basic principles of the CNN deep learning model and the Inception v3 transfer learning model. The third part combines the above mentioned two models to train and classify some sample images, and comparing the final results with only using CNN; and the final part summarizes a few advantages of this novel combing approach, and proposes some suggestions for the future related works.

2. Methods

2.1 CNN

CNN is known as a type of feed-forward artificial network that widely applied in computer vision field especially in image and video classification. CNN has been proven very effective and thus is an important tool in Deep Learning field (Chang, et al, 2016); (Zhou, Jin & Dong, 2017).

A CNN (Lv, Luo, Huang & Jiang, 2014) comprises of an input layer, output layer and multiple hidden layers. The hidden layers are made up of convolutional layers, non-linear (ReLu) layers, pooling layers and fully connected layers. A classic CNN architecture is shown as below:

![Simple Block Diagram of convolutional neural network](image)

**Figure 1.** Simple Block Diagram of convolutional neural network

Basically, CNN takes an image, pass it through a series of convolutional, nonlinear, pooling and fully connected layers, and in the end an output is produced. Based on observation from the CNN, process each layer carries out their particular role of transforming the input into a useful representation.

2.1.1 Convolutional layer

The first layer of CNN is referred as the Convolutional Layer, also known as Conv. Layer. This layer practices the foundation of the CNN. It enters the convolution operation and transfers the outcome to the following layer. Convolutions convey individual neuronal responses into visual stimuli (Bouvier, 2006).
2.1.2 Nonlinear layer (ReLu)

ReLu is the most frequently arrayed activation function for the outputs of the CNN neurons. However, the ReLu function is not differentiable at the starting point, which makes it tough to practice with back propagation training. Hence, Softplus function, a smoother version is being placed in to perform ReLu role.

2.1.3 Pooling layer

Pooling layers can be local and global layers. Pooling chains the output from different layers into one single layer(Ciresan, et al, 2011). Similar to Max pooling which assembles maximum values at earlier layers(Ciresan, Meier & Schmidhuber, 2012). Likewise, pooling is able to handle larger problems by eliminate a lot of data and minimalize the computation. Large size of dataset calculations are extremely pricy in sense of computation.

2.1.4 Fully connected layer

After numerous layers of convolution and pooling, the high-level classification in the CNN proceeds through the fully connected layer. At this final layer, it looks at the output of the previous layers and determines which features most correlate to a particular class.
In short, each convolutional layer practices multiple conversion filters to convert the lower-level features of the prior layer to advanced features. Then the pooling layer seizes fixed invariants after the image output is scaled and translated by Conv. layer. At the final stage, the fully connected layer calculates and assembles high-level features for further classification (Szegedy, Vanhoucke, Ioffe, Shlens & Wojna, 2016).

2.2 Inception

The Inception v3 model was developed by Google and is a combination of label smoothing, auxiliary classifiers, volume integration solutions, and so on. The volume integral solution is a layer of 5x5 convolution that can be replaced by two layers of 3x3 convolution, so that the parameters become less; and a layer of 3x3 convolution can be further replaced by 1x3 convolution and 3x1 convolution, which also makes the parameters less. In this way, by the volume integration solution, the amount of calculation can be greatly reduced, thereby slimming the network. Fig.4 is an example of the Inception module after the specific volume integration solution.

![Inception module](image)

Figure 4. Inception modules with expanded the filter bank outputs (Szegedy, et al, 2016)

The Auxiliary Classifier does not help to converge faster, but it can be used as a regularization, with a supervised signal in the middle, which can be slightly better.

Label-smoothing regularization means that each category corresponds to a number, and the sample given corresponds to only one unique ground truth, then the corresponding target is 1, the others are 0, and then the cross entropy is calculated. Now the distribution is only 1 at the target point and 0 at other places. It is too hard. Intuitively, the theoretical target distribution is too sharp, so there is a concept of soft labeling, giving all 100% probability to Other tags are evenly distributed a little bit. Don't let the network be too confident to predict that 100% is a certain class. The soft labels are as follows:

\[ q'(k) = (1 - \varepsilon) \delta_{k,y} + \frac{\varepsilon}{K} \]  

(1)
\( \varepsilon \) is a smaller number, so the label corresponding to ground truth has a majority probability, while other categories have a small probability of being lost. Over-fitting can be avoided to prevent the network from being overconfident in predicting a particular category. Thus, the new cross entropy loss becomes the weighted sum of the cross entropy losses for the corresponding distribution of each sample:

\[
H(q', p) = - \sum_{k=1}^{K} \log p(k)q'(k) = (1 - \varepsilon)H(q, p) + \varepsilon H(\mu, p) \tag{2}
\]

The resulting Inception v3 is a network that is added one by one from top to bottom according to the following Table 1.

| Network                        | Top-1 Error | Top-2 Error | Top-3 Error |
|-------------------------------|-------------|-------------|-------------|
| GoogLeNet[20]                 | 29%         | 9.2%        | 1.5         |
| BN-GoogLeNet                  | 27%         | -           | 1.5         |
| BN-Inception[7]               | 25%         | 7.8%        | 2           |
| Inception-v2                  | 23%         | -           | 3.8         |
| Inception-v2 RMSProp          | 23%         | 6.3%        | 3.8         |
| Inception-v2 Label Smoothing  | 23%         | 6.1%        | 3.8         |
| Inception-v2 Factorized 7*7   | 22%         | 5.8%        | 4.8         |
| Inception-v2 BN-auxiliary     | 21%         | 5.6%        | 4.8         |

3. Experiments

3.1 Sample images

First of all, training a convolutional neural network requires a large number of image samples, which should have the following characteristics:

1. High resolution. The higher the resolution of the remote sensing image, the more the image can reflect the characteristics of the target, that is to say, the more obvious the characteristics of the image transmitted to the neural network.

2. Cover as many target types as possible. The type of training directly determines the type that can be identified.

3. The number is huge. The number of training determines the accuracy of the classification, and generally requires more than \(10^6\) images.

Since there is no requirement for remote sensing image library, a large number of flower images cannot be obtained, so the number of pictures is difficult to meet the requirements, and the training speed and accuracy can only be improved through transfer learning. The main source of the image is Google Images. Choose high resolution and representative images from Google Flower image types. Picture types include daisy, rose, tulip, dandelion and sunflower, and the five categories are the most
common in flower images. After downloading the image, use the slicing tool to cut it into 299*299 pixels to improve the efficiency of image preprocessing. Specific examples are as follows:

![Sample Images](image)

**Figure 5.** Sample images

Finally, 1369 pictures of flowers were obtained, including 265 pictures of daisies, 375 pictures of dandelions, 351 pictures of roses, 231 pictures of sunflowers, and 242 sheets of tulips.

### 3.2 Training CNN based on Inception

We used python3.6 to write code on the platform tensorflow to implement the test process. The image is preprocessed to adapt the sample to the convolutional neural network, and the diversity of the training data set is improved by changing the image features, thereby improving the adaptability of the training model.

```python
training_images=[]
training_labels=[]
testing_images=[]
testing_labels=[]
validation_images=[]
validation_labels=[]
```

First read the image sample from the corresponding folder, and divide the picture into corresponding daisy, rose, tulip, dandelion and sunflower according to the name of the picture, and store it in the corresponding list. The corresponding tags are also stored in the list. Scramble the elements in all the lists in order.

```python
image=tf.image.resize_images(image,(299,299))
```
Read all the information in the image and decode it, channels = 3 for the color picture, including r, g, b three channels to crop or enlarge the picture to the specified 299*299, and standardize the data. The standard definition is minus its mean divided by its variance.

\[
\text{image\_raw\_data} = \text{gfile.FastGFile(file\_name,'rb').read()}
\]
\[
\text{image} = \text{tf.image.decode\_jpeg(image\_raw\_data)}
\]
\[
\text{image} = \text{tf.image.per\_image\_standardization(image)}
\]

After generating the batch image, determine the maximum number of images in the queue is 30, use \text{np.random.shuffle} to disrupt the order and convert the image.

\[
\text{np.random.shuffle(training\_images)}
\]
\[
\text{np.random.set\_state(state)}
\]
\[
\text{np.random.shuffle(training\_labels)}
\]

This way, the picture can be directly input to the neural network. In addition, more training times are needed during training. For example, if I rotate each picture once, the number of trainings will be doubled. That is to say, the diversity of the training set is increased, and the number of trainings is also increased.

Define the parameters used in the training.

\[
\text{LEARNING\_RATE} = 0.0001
\]
\[
\text{STEPS} = 300
\]
\[
\text{BATCH} = 32
\]
\[
\text{N\_CLASSES} = 5
\]

Five output neurons have five kinds of outlets, and the size of the redefinition picture is 299*299. If the picture is too large, the training is slow. The size of each batch of data is 32. CAPACITY is set to 128, the number of training steps is set to 10000, and the learning rate is 0.0001.

Enumerate all the parameters in the inception v3 model and then determine whether you need to remove them from the load list.

\[
\text{def get\_tuned\_variables():}
\]
\[
\text{exclusions} = [\text{scope.strip()} \text{for scope in CHECKPOINT\_EXCLUDE\_SCOPES.split(',')}]
\]
\[
\text{variables\_to\_restore} = []
\]
\[
\text{for var in slim.get\_model\_variables():}
\]
\[
\text{excluded} = \text{False}
\]
\[
\text{for exclusion in exclusions:}
\]
\[
\text{if var.op.name.startswith(exclusion):}
\]
\[
\text{excluded} = \text{True}
\]
\[
\text{break}
\]
\[
\text{if not excluded:}
\]
\[
\text{variables\_to\_restore.append(var)}
\]
\[
\text{return variables\_to\_restore}
\]

Enumerate all the parameter prefixes that need to be trained, and use these prefixes to find all the parameters that need to be trained.

\[
\text{def get\_trainable\_variables():}
\]
\[
\text{scopes} = [\text{scope.strip()} \text{for scope in TRAINABLE\_SCOPES.split(',')}]
\]
\[
\text{variables\_to\_train} = []
\]
\[
\text{for scope in scopes:}
\]
variables = tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES, scope)  
variables_to_train.extend(variables)  
return variables_to_train

Define the loss function and the training process.

tf.losses.softmax_cross_entropy(  
    tf.one_hot(labels, N_CLASSES), logits, weights=1.0)  

total_loss = tf.losses.get_total_loss()  

train_step=  
tf.train.RMSPropOptimizer(LEARNING_RATE).minimize(total_loss)

Get accuracy.

with tf.name_scope('evaluation'):  
correct_prediction = tf.equal(tf.argmax (logits, 1), labels)  
evaluation_step = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

Save the model every 30 steps and save the model in checkpoint_path.

Summary of training results:

Step 0, train loss= 2.6, train accuracy=22.00%
Step 30, train loss= 1.8, train accuracy=34.6%
Step 60, train loss= 1.4, train accuracy=63.7%
Step 90, train loss= 0.6, train accuracy=85.2%
Step 120, train loss= 0.3, train accuracy=85.2%
Step 150, train loss= 0.3, train accuracy=85.2%
Step 180, train loss= 0.3, train accuracy=87.9%
Step 210, train loss= 0.2, train accuracy=87.4%
Step 240, train loss= 0.2, train accuracy=86.8%
Step 270, train loss=0.3, train accuracy=88.5%
Step 300, train loss=0.2, train accuracy=88.5%

3.3 Training without transfer learning

According to previous experience, we can randomly generate every parameter and train them from
very beginning but without Transfer Learning of inception_v3. The neural network was built with four
convolutional layers, pooling layers and three connection layers. The size of inputting data is
100*100. The convolutional layer all uses the complement of 0, so the length and width of the
convolutional layer did not change, only the depth is enhanced. The length and width of the pooled
layer are reduced, but the depth not. The changed in the size of model: 100×100×3→100×100×
32→50×50×32→50×50×64→25×25×64→25×25×128→12×12×128→6×6 ×
128. After training of about 3600 pictures, and the final accuracy is 60.2%.

3.4 Discussions

To sum up, the loss rate is high and the accuracy rate is not stable at the beginning of training.
When the training step reaches more than 150 steps, the loss rate approaches 0.2 and the accuracy rate
is stable above 80%. This experimental test set is a random sample of 130 flower images from the
original image set, which is not repeated with the training images, with an average of 20-30 per type.
The pictures are grouped one by one into 299*299 pixels, and the model is calculated to obtain the maximum probability index. The overall test set accuracy rate of the test results was 88.3%. If we train from very beginning without Transfer Learning, the test probability would be much lower (only 60.2%) because of the over-fitting. On one side, if there are enough data—such as at least $10^5$ pictures, and training from very beginning is better than transfer learning. On the other side, if it is a little data training, the transfer learning would be a suitable solution.

4. Conclusions

This paper collects high-resolution flower images from Google Images, and pre-processes the images as required. Based on the inception-v3 model of Tensorflow platform, the transfer learning method is used to train the flower recognition classification model. The classification recognition rate of flower recognition in this model can reach 88.3%, and the accuracy of training without transfer learning is 60.2%. In addition, this approach requires no complicated pre-processing compared to conventional methods. It can identify data with noise or deformation, has strong adaptive learning ability, and has quick classification speed.

However, training a high recognition rate neural network requires collecting a large number of samples as a training set, but the collection of image samples is limited, resulting in a training data set that is too small, several orders of magnitude smaller than the normal number of pictures. It is inevitable to train neural network overfitting from scratch. Thus, using transfer learning to access CNN and fine-tuning it with sample images, so that the neural network of small sample training can also achieve the better classification results.

5. Acknowledgements

This work was supported by China’s 973 Research Plan (Grant No. 2015CB453005), and we also thank the anonymous reviewers for their helpful suggestions.

References

[1] Angelova A, Zhu S. Efficient object detection and segmentation for fine-grained recognition[J]. IEEE Conference on Computer Vision and Pattern Recognition, 2013: 9(4): 811–818.

[2] Chang Liang, Deng Xiaoming, Zhou Mingquan, Wu Zhongke, Yuan Ye, Yang Shuo, Wang Hongan. Convolutional Neural Networks in Image Understanding[J]. Acta Automatica Sinica, 2016, 42(09):1300-1312.

[3] CHERIYADAT A M. Unsupervised feature learning for aerial scene classification[J]. IEEE T Geosci Remote, 2013, 52: 439-451.

[4] D. C. Cireşan,U. Meier,J. Masci,et al. Flexible, high performance convolutional neural networks for image classification [C].IJCAI International Joint Conference on Artificial Intelligence, 2011, 1237–1242.

[5] D. Cireşan,U. Meier,and J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification [J].International Conference of Pattern Recognition, 2012, (February):3642–3649.

[6] J. Bouvrie. Notes on convolutional neural networks [J].In Practice, 2006, 47–60.

[7] Lin Yu, Chen Xiaoyong. Research on Road Traffic Sign Recognition Based on Inception v3 Model[J].Jiangxi Science, 2018, 36(05): 849-852.
[8] Lv Guohao, Luo Siwei, Huang Yaping, Jiang Xinlan. Regularization Method Based on Convolutional Neural Network[J]. Computer Research and Development, 2014, 51(09): 1891-1900.

[9] Liu Chen, Qu Changwen, Zhou Qiang, Li Zhi, Li Jianwei. Target classification of SAR images based on convolutional neural network transfer learning [J]. Modern Radar, 2018, 40: 38-42.

[10] Lv Guohao, Luo Siwei, Huang Yaping, Jiang Xinlan. Regularization Method Based on Convolutional Neural Network[J]. Computer Research and Development, 2014, 51(09): 1891-1900.

[11] Li Guandong, Zhang Chunju, Wang Mingxi, Zhang Xueying. Classification of high-resolution image scenes based on convolutional neural network migration[J]. Science of Surveying and Mapping, 2019(6): 1-13.

[12] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep Face Recognition. Proceedings of the British Machine Vision Conference 2015, 2015, 41.1-41.12.

[13] PAN S J, YANG Q. A survey on transfer learning[J]. IEEE T Knowl Data En, 2010, 22: 1345-1359.

[14] SCHMIDHUBER J. Deep learning in neural networks: an overview[J]. Neural Netw, 2014, 61: 85-117.

[15] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision [J]. Computer Science, 2015: 2818-2826.

[16] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-January, pp. 2818-2826.

[17] YANG Y, NEWSAM S. Spatial pyramid co-occurrence for image classification[C]//IEEE International Conference on Computer Vision. IEEE, 2011: 1465-1472.

[18] Yang Jingdong, Zhang Peng. A Method of Tongue Image Classification Based on Transfer learning for Fully Connected Neural Networks[J]. Journal of Second Military Medical University, 2018, 39(08): 897-902.

[19] ZHOU Feiyan, JIN Linpeng, DONG Jun. A Review of Convolutional Neural Networks[J]. Chinese Journal of Computers, 2017, 40(06): 1229-1251.

[20] Côté-Allard U, Fall C L, Drouin A, et al. Deep learning for electromyographic hand gesture signal classification using transfer learning[J]. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2019, 27(4): 760-771.