Scene Classification Based On VGGNet Model

1 Algorithm Flow

Since the input to VGG must be a 224 x 224 RGB image, the average of the three channels needs to be calculated during preprocessing, and the average is subtracted from each pixel, which allows for fewer iterations after processing and can make convergence faster. Meanwhile, the class names of different categories are marked to facilitate the reading of subsequent files. In addition, VGG's training set should contain 20% of the test set (training in the training, not the same as the test set).

The image passes through a 3×3 convolution kernel. The reason for choosing this size is several reasons:

(1)3×3 is the smallest size capable of capturing pixel eight neighborhood information.

(2)The limited receptive field of two 3×3 stacked roll base layers is 5×5; the receptive field of three 3×3 stacked roll base layers is 7×7, so large-scale convolution can be replaced by stacking of small-sized convolution layers. Layer, and feel the same size.

(3)Multiple 3×3 volume base layers have more nonlinearity (more layer nonlinear functions) than a large size filter volume base layer, making the decision function more decisive.

(4)Multiple 3×3 convolutional layers have fewer parameters than a large-sized filter, assuming that the input and output feature maps of the volume base layer are the same size. Then three 3×3 convolution layer parameters number $3\times(3\times3\times C\times C)=27C^2$; a 7×7 convolution layer parameter is $49C^2$. So you can think of three 3×3 filters as a decomposition of 7×7filter (the middle layer has nonlinear decomposition and plays the role of implicit regularization.

In some convolutional layers, a 1×1 convolution kernel is also used, which can be seen as a linear stretch in the input channel, followed by a nonlinear decomposition. The step size of the convolution layer is set to 1 pixel, for example, the padding of the 3×3 convolution layer is 1 pixel. The pooling layer uses max pooling. After a part of the convolutional layer, the max pooling window is 2×2 and the step size is set to 2.

The convolutional layer is followed by three fully-connected layers (FCs).The first two fully connected layers have FC of 4096 neurons, and the third fully connected layer has 1000 neurons of FC for classification. The full connectivity layer configuration of all networks is the same.

After the fully connected layer is SoftMax which is used for classification. All hidden layers (in the middle of each conv layer) use ReLU as the activation function. VGGNet does not use Local Response Normalization (LRN). This standardization does not improve performance on some data sets, but leads to more memory consumption and computation time (LRN: Local Response Normalization, local response normalization, for enhancement). The generalization ability of the network).However, this operation was modified as appropriate because the operating network was not as deep as the original network and LRN was used.

2 Training Data (take the data set rsi-cb256 as an example)

2.1 Operating Environment

The operating environment is MATLAB R2016a, which is used with MatConvNet.System memory 24.0GB, i7-4790K CPU @ 4.00GHz.

2.2 Parameter Settings

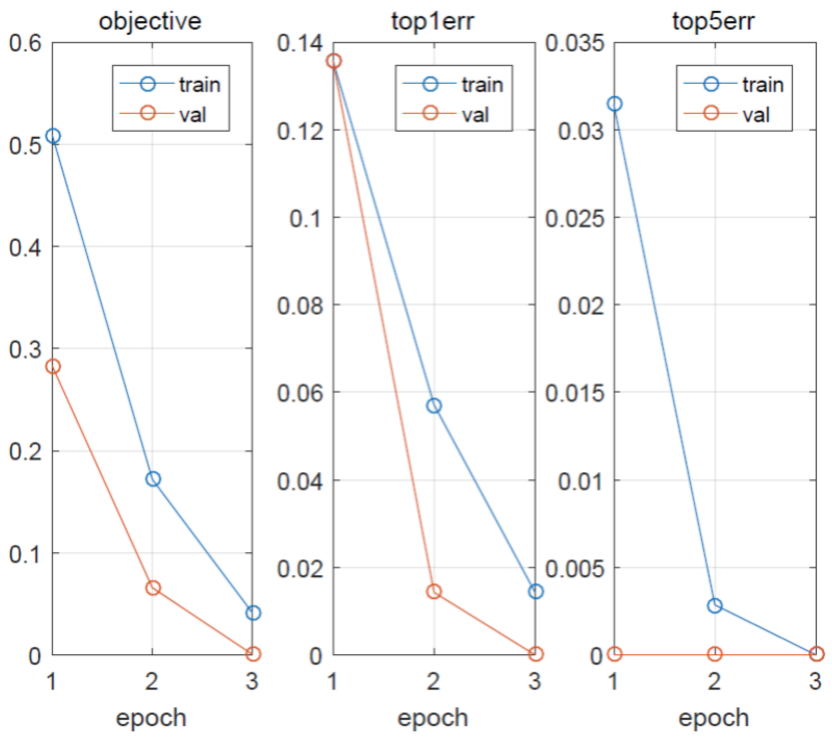
Through experiments, we can find that the accuracy of the 21-layer network model is relatively high. If we continue to deepen the network depth, it will easily cause over-fitting, which will lead to a decrease in accuracy. The VGG network parameters under the rsi-cb256 data set are shown in Table 1.

In addition, because the data requires 50% of the test data and 50% of the training data, this time, 100 training sets, 20 test sets (val) in the training set, and 100 test sets are used.

Table 1 VGG network parameters under RSI-CB256 dataset

layer	**0.00**	**1.00**	**2.00**	**3.00**	**4.00**	**5.00**	**6.00**	**7.00**	**8.00**	**9.00**	**10.00**
type	input	conv	relu	lrn	mpool	conv	relu	lrn	mpool	conv	relu
name	n/a	conv1	relu1	norm1	pool1	conv2	relu2	norm2	pool2	conv3	relu3
support	n/a	11.00	1.00	1.00	3.00	5.00	1.00	1.00	3.00	3.00	1.00
filt dim	n/a	3.00	n/a	n/a	n/a	64.00	n/a	n/a	n/a	256.00	n/a
filt dilat	n/a	1.00	n/a	n/a	n/a	1.00	n/a	n/a	n/a	1.00	n/a
num filts	n/a	64.00	n/a	n/a	n/a	256.00	n/a	n/a	n/a	256.00	n/a
stride	n/a	4.00	1.00	1.00	2.00	1.00	1.00	1.00	2.00	1.00	1.00
pad	n/a	0.00	0.00	0.00	0x1x0x1	2.00	0.00	0.00	0x1x0x1	1.00	0.00
rf size	n/a	11.00	11.00	11.00	19.00	51.00	51.00	51.00	67.00	99.00	99.00
rf offset	n/a	6.00	6.00	6.00	10.00	10.00	10.00	10.00	18.00	18.00	18.00
rf stride	n/a	4.00	4.00	4.00	8.00	8.00	8.00	8.00	16.00	16.00	16.00
data size	224.00	54.00	54.00	54.00	27.00	27.00	27.00	27.00	13.00	13.00	13.00
data depth	3.00	64.00	64.00	64.00	64.00	256.00	256.00	256.00	256.00	256.00	256.00
data num	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
data mem	6MB	7MB	7MB	7MB	2MB	7MB	7MB	7MB	2MB	2MB	2MB
param mem	n/a	91KB	0B	0B	0B	2MB	0B	0B	0B	2MB	0B
layer	**11.00**	**12.00**	**13.00**	**14.00**	**15.00**	**16.00**	**17.00**	**18.00**	**19.00**	**20.00**	**21.00**
type	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmxl
name	conv4	relu4	conv5	relu5	pool5	fc6	relu6	fc7	relu7	fc8	loss
support	3.00	1.00	3.00	1.00	3.00	6.00	1.00	1.00	1.00	1.00	1.00
filt dim	256.00	n/a	256.00	n/a	n/a	256.00	n/a	4096.00	n/a	4096.00	n/a
filt dilat	1.00	n/a	1.00	n/a	n/a	1.00	n/a	1.00	n/a	1.00	n/a
num filts	256.00	n/a	256.00	n/a	n/a	4096.00	n/a	4096.00	n/a	7.00	n/a
stride	1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.00	1.00	1.00	1.00
pad	1.00	0.00	1.00	0.00	0x1x0x1	0.00	0.00	0.00	0.00	0.00	0.00
rf size	131.00	131.00	163.00	163.00	195.00	355.00	355.00	355.00	355.00	355.00	355.00
rf offset	18.00	18.00	18.00	18.00	34.00	114.00	114.00	114.00	114.00	114.00	114.00
rf stride	16.00	16.00	16.00	16.00	32.00	32.00	32.00	32.00	32.00	32.00	32.00
data size	13.00	13.00	13.00	13.00	6.00	1.00	1.00	1.00	1.00	1.00	1.00
data depth	256.00	256.00	256.00	256.00	256.00	4096.00	4096.00	4096.00	4096.00	7.00	1.00
data num	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	1.00
data mem	2MB	2MB	2MB	2MB	360KB	160KB	160KB	160KB	160KB	280B	4B
param mem	2MB	0B	2MB	0B	0B	144MB	0B	64MB	0B	112KB	0B

Figure 1 RSI-CB256 data set network effect



3 Accuracy Assessment

1.Dataset RSI-CB256

Table 2 Confusion matrix under RSI-CB256 dataset

OA=98.29%		Predicted Class							
		airplane	bare_land	coastline	forest	marina	snow_mountain	storage_room	sum
Actual Class	airplane	88	0	0	0	0	12	0	100
	bare_land	0	100	0	0	0	0	0	100
	coastline	0	0	100	0	0	0	0	100
	forest	0	0	0	100	0	0	0	100
	marina	0	0	0	0	100	0	0	100
	snow_mountain	0	0	0	0	0	100	0	100
	storage_room	0	0	0	0	0	0	100	100
	sum	88	100	100	100	100	112	100	700

2.Dataset FNWPU_RESISC45

Table 3 Confusion matrix under the FNWPU_RESISC45 data set

OA=90.71%		Predicted Class							
		railway	meadow	harbor	forest	desert	circular_farmland	beach	sum
Actual Class	railway	92	1	0	0	3	0	4	100
	meadow	3	96	0	1	0	0	0	100
	harbor	2	1	82	1	0	0	14	100
	forest	0	15	0	85	0	0	0	100
	desert	0	0	0	1	97	0	2	100
	circular_farmland	4	0	0	0	2	92	2	100
	beach	1	0	0	1	7	0	91	100
	sum	102	113	82	89	109	92	113	700

3.Dataset AID

Table 4 Confusion matrix under the AID dataset

OA=71.86%		Predicted Class							
		port	pond	playground	meadow	desert	baseballfield	bareland	sum
Actual Class	port	78	17	1	3	0	0	1	100
	pond	13	81	1	2	1	0	2	100
	playground	17	23	40	3	0	14	3	100
	meadow	0	14	0	84	0	1	1	100
	desert	1	0	0	0	96	1	2	100
	baseballfield	9	29	2	7	0	53	0	100
	bareland	0	2	1	0	26	0	71	100
	sum	118	166	45	99	123	69	80	700

The data image of the data set AID is 600×600, and the input requirement of the VGGNet network must be 224×224. Inputting through random cropping image will cause the resolution of the original image to decrease, resulting in the feature extraction is not obvious, so the whole The accuracy is seriously degraded.

In addition, through the evaluation of the results, we found that many objects, especially the playground and baseballfield, which were misclassified into ponds. The reason is that the shape of the pond is similar to that of the playground, and the lakes in the pond are mostly green, which is very similar to the color of the baseball field.

Figure 2 A sample of the aid data set

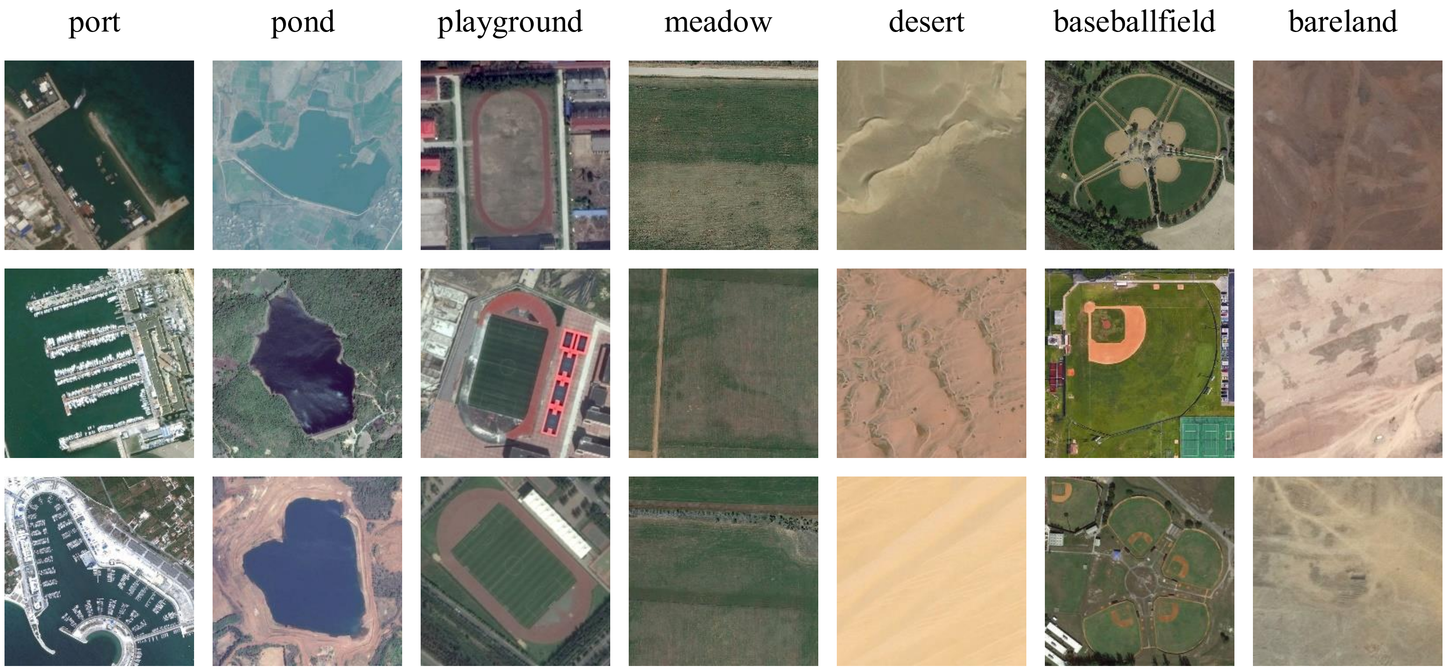
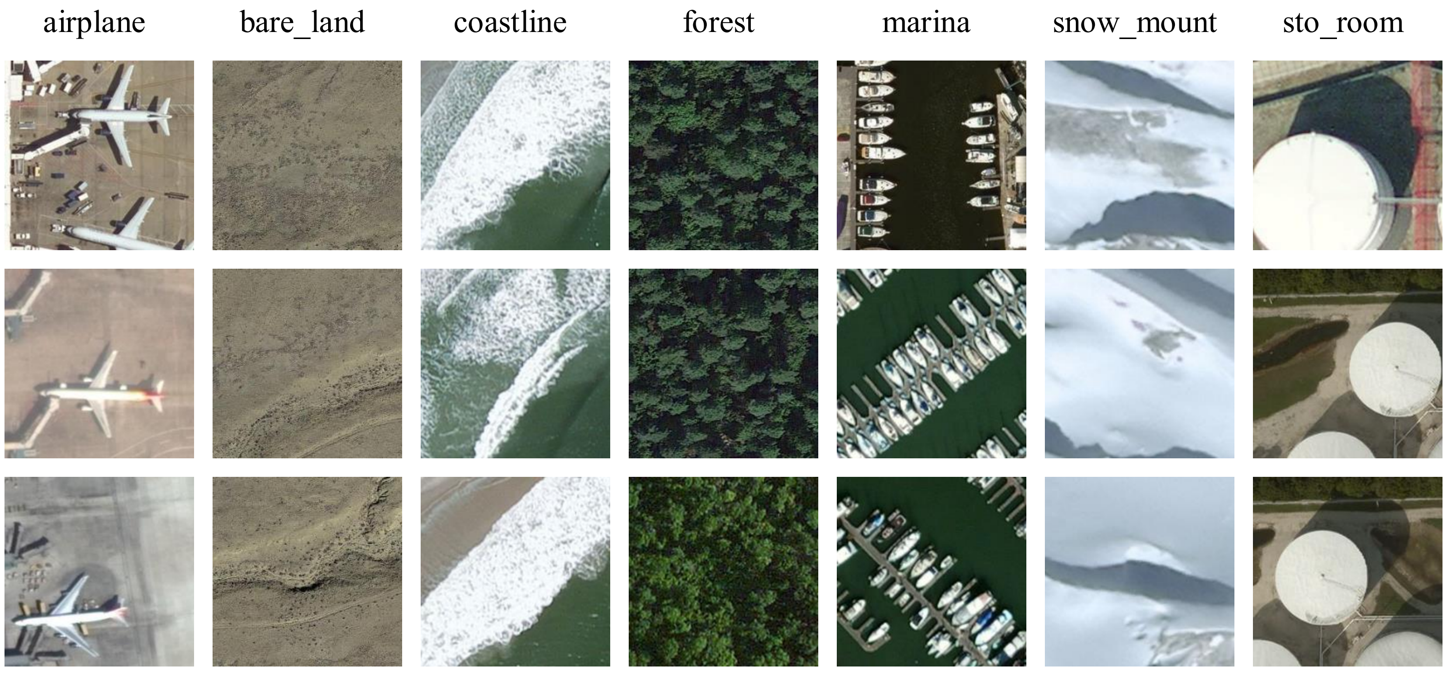


Figure 3 Partial sample of the rsi-cb256 data set



Randomly extract the images in different datasets. By comparison, we can also find that compared with the rsicb256 dataset, the sample images in the aid dataset have different color differences, and the texture features are not the same, which also leads to the aid dataset. One of the important reasons for the reduced accuracy of scene classification.

4 Program Instructions

1.First, you need to configure matconvnet correctly, and put the downloaded matconvnet-1.0-beta25 in the MATLAB root directory.

1.Use Microsoft Visual C++ 2015 Professional (C) for C++ compilation, enter the command mex -setup C++ statement from the command line, and then run the matconvnet-1.0-beta25\ matconvnet-1.0-beta25\matlab\ vl_compilenn.m file to complete the initial configuration.

3.Prepare the data training, open the generateDATA.m file, and modify the 7 categories that you want to train in the class of line2. The suffix of the modified file in ext is .tif or .jpg.Enter your own data set in line23, 25, 38, 40, 63, 65.Because the data requires 50% of the test data and 50% of the training data, this time using 100 training sets, 20 training sets (val), 100 test sets, can be completed in line18,19,20 Data modification.,

4.Open the test.m file and change the document path in line13 and line18 to the path where you need to store the corresponding information for epoch.It should be noted here that when training with different data sets, the data folder of the first data set needs to be deleted. Otherwise, the training of the first set of models is still used, which easily causes the accuracy to drop from 95% to 23%. Happening.

5.Run the test.m file. The accuracy matrix in the running result indicates OA. The two columns of the res matrix are: prediction result and actual category.

References

Simonyan K , Zisserman A . Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.

Chatfield K , Simonyan K , Vedaldi A , et al. Return of the Devil in the Details: Delving Deep into Convolutional Nets[J]. Computer Science, 2014.