Application and Optimization of Image Fuzzy Control Algorithm based on Gaussian Blur in TensorFlow Training

Yongjun Zhang, Feiyang Ma*

State Key Laboratory of Information and Optical Communications, Beijing University of Post and Telecommunications, Beijing, China

* mafeiyang@bupt.edu.cn

Abstract. When training a convolutional neural network that can interpret picture ambiguity, a large number of pictures with different ambiguities are needed as a training set. This paper introduces a method to adjust the image ambiguity by adjusting the parameters in the Gaussian fuzzy algorithm. Finally, the convolutional neural network training based on TensorFlow migration learning is completed, and the ambiguity judgment of the image is realized, and the optimization effect of the fuzzy control algorithm is verified.

Keywords: TensorFlow, Gaussian fuzzy algorithm, the convolutional neural network, the fuzzy control algorithm.

1. Introduction

Gaussian blur is a widely used technique in image processing and is commonly used to reduce image noise to reduce the level of detail. It is actually a low-pass filter. For the image, it is passed in the low-frequency part, and it is filtered for the high-frequency part, and then the details such as the edge of the image are blurred. By adjusting the parameters in the Gaussian blur to change the ambiguity, a set of images with different ambiguities can be created to provide data for the training of the convolutional neural network.

2. Selection of Fuzzy Algorithm

In order to train the convolutional neural network, a model that can determine the ambiguity of the image is obtained, and a large number of images with different degrees of blurring are required as the training set. Therefore, we need to choose a fuzzy algorithm, which can control the ambiguity of the picture flexibly and conveniently, and highly reduce the blurring of photos caused by camera shake or focus failure in real life.

2.1 Mean Blur

The mean blur algorithm uses a method in which each pixel is set to the average of the surrounding pixels. The "intermediate point" takes the average of the "around points", which is a kind of "smoothing" in numerical value. On the graph, it is equivalent to producing a "fuzzy" effect, and the "intermediate point" loses detail. From the algorithm point of view, the mean blur speed is fast, but the edge details of the picture are not soft.

2.2 Gaussian Blur

If you use a simple average, it is obviously not very reasonable, because the images are continuous, the closer the point is, the closer the relationship is, and the farther away the point is, the more distant the relationship is. Therefore, the weighted average is more reasonable, the closer the distance is, the greater the weight, and the farther the distance is, the smaller the weight is. From the algorithm point of view, Gaussian blur is closer to the blur effect produced in real life, but the speed is slower.

In summary, the choice of Gaussian fuzzy algorithm can restore the fuzzy height generated in real life, which is a better training sample.

3. Gaussian Blur of Image

From a mathematical point of view, the Gaussian blurring process of an image is that the image is convolved with a normal distribution. Since the normal distribution is also called Gaussian distribution, this fuzzy algorithm is called Gaussian blur. Since the Fourier transform of the Gaussian function is another Gaussian function, Gaussian blur is the effect of the low-pass filter for the image.

3.1 Gaussian Function

Gaussian blur is an image blur filter that computes the transformation of each pixel in an image using a normal distribution. The N-dimensional space normal distribution equation is:

$$G(r,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}^N} e^{-r^2/(2\sigma^2)} \quad (1)$$

Defined in 2D space as:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)} \quad (2)$$

Gaussian blur is defined as the convolution of the image with the Gaussian distribution:

$$L(x,y,\sigma) = G(x,y,\sigma) \times I(x,y) \quad (3)$$

In two-dimensional space, the contour of the surface generated by this formula is a concentric circle that is normally distributed from the center, and its distribution curve is shown in the figure.

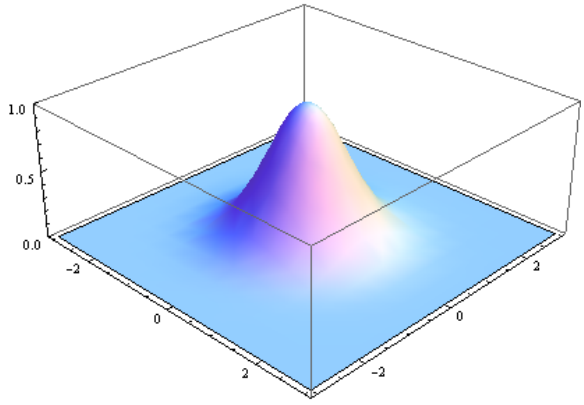

Figure 1. Two-dimensional normal distribution curve.

A convolution matrix composed of pixels whose distribution is not zero is transformed with the original image. The value of each pixel is a weighted average of surrounding neighboring pixel values. The value of the original pixel has the largest Gaussian distribution value, so there is the largest weight. The adjacent pixels are getting farther and farther away from the original pixel, and their weights are getting smaller and smaller.

3.2 Weight Matrix

Filtering is a basic method of processing an image, and a filter is a convolution kernel. The calculation matrix is the key to Gaussian blur.

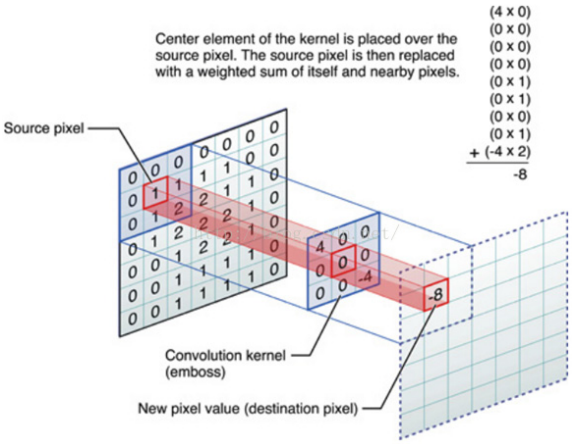

Figure 2. convolution.

The matrix in the figure is a two-dimensional original pixel matrix, a two-dimensional image filtering matrix, and a final filtered new pixel map. For each pixel of the original image, the scores of its domain pixels and the corresponding elements of the filter matrix are calculated, and then added up as the value of the current center pixel position, thus completing the filtering process.

According to the Gaussian function, the weight of each coordinate point of the matrix around the center point is calculated. The value of σ and the radius of the matrix need to be determined, assuming σ is 1.5 and the matrix radius is 1.

Assuming the coordinates of the center point are (0,0), the coordinates of the 8 points closest to it are as follows:

(-1,1)	(0,1)	(1,1)
(-1,0)	(0 ,0)	(1,0)
(-1, -1)	(0, -1)	(1, -1)

Farther points and so on.

In order to calculate the weight matrix, it is necessary to set the value of σ. Assuming σ = 1.5, the weight matrix with a blur radius of 1 is as follows:

0.04535	0.05664	0.04535
0.05664	0.07073	0.05664
0.04535	0.05664	0.04535

After the weight matrix is obtained, the sum of the pixel values of each point multiplied by the weight value is the new pixel value of the center point.

4. Optimization of Ambiguity Control Algorithm

According to the formula, the change of ambiguity can be controlled by adjusting two parameters: (1) size is the radius of the matrix, the larger the size, the higher the ambiguity, the slower the processing speed. (2) The larger the standard deviation σ, the blur the higher the degree, the less obvious the effect, and the ambiguity of the change without changing the radius is large.

4.1 Setting of Σ Value

As shown in the figure, the normal distribution is a bell-shaped curve on the graph. The closer the value is to the center, the smaller the value is from the center. When calculating the average value, we only need to use the "center point" as the origin, and other points according to their position on the normal curve, we can assign a weight to get a weighted average. However, in the experiment, it

was found that when the size increased to close to 3σ, the increase of the image ambiguity became less obvious and could not meet the requirements of the training data.

Referring to the normal distribution curve, you can know the points other than the 3σ distance, and the weight is negligible. The reverse push shows that when the blur radius is r, taking σ as r/3 is a suitable value.

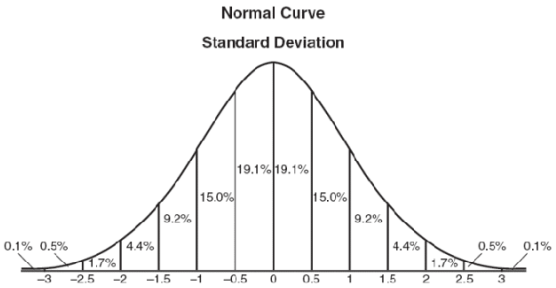

Figure 3. Normal distribution curve.

4.2 Edge Problem

An input image must deal with edge problems when convolving with any core. The essence of the edge problem is that when the edge of the image tries to convolve with the kernel, the kernel will exceed the image boundary. As shown in the figure below, the red square indicates the Gaussian kernel, and the red dotted line indicates the range of the excess image, which cannot be convoluted.

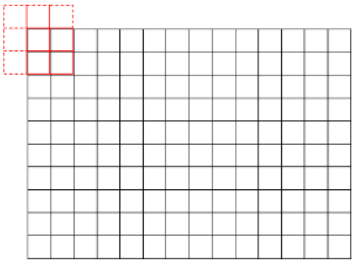

Figure 4. Edge problem in convolution.

There are generally three ways to deal with this situation. In this article, choose the third one:

1. Discard the edges. Edge parts that cannot participate in the convolution will be discarded. The disadvantage of this approach is that the output image is "one turn" smaller than the input image, especially after multiple convolutions; the image will become smaller and smaller;

2. The edge does not participate in convolution. The edge of the pixel retains the original value and remains in the output image. The disadvantage of this approach is that the edge portion is not blurred;

3. Set the part outside the image to 0 and then participate in the convolution.

4.3 Comparison Before and After Optimization

Four different degrees of blurring images produced by the algorithm before optimization can be found that when the blur radius reaches 55 or more, the change of the blur degree is no longer obvious:

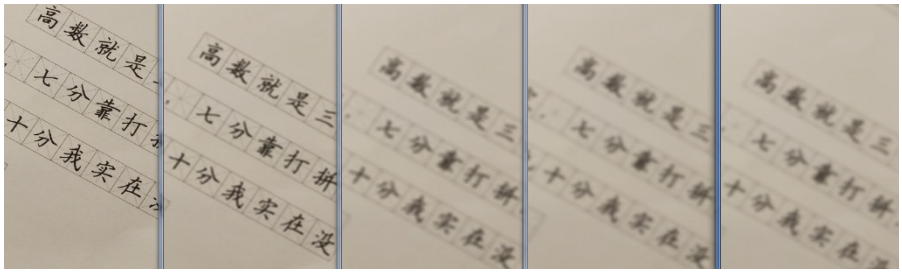

Figure 5. Blurring image before optimization.

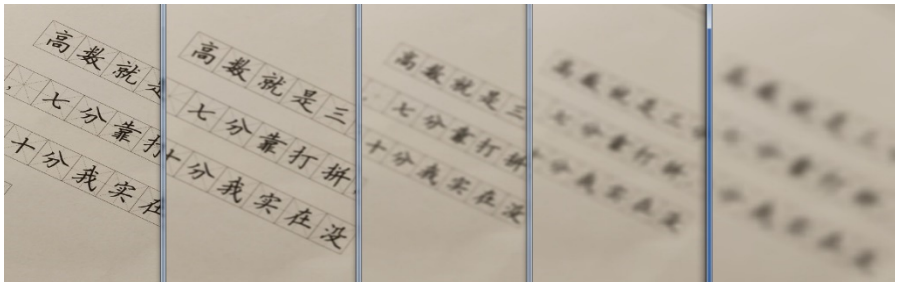

Figure 6. Blurring image after optimization.

5. Batch Image Blurring Method

Because of the huge amount of training data required, this paper uses the java language to write a batch image processing method. Read clear images from the root directory, process the images with different ambiguities, and store them in different folders.

After execution, four picture folders of different ambiguity are generated. The number in the folder name indicates the blur radius, and the clear original picture is placed in the same directory:

6. TensorFlow Training and Results Verification

The parameters of the convolutional neural network are millions, and it takes a lot of time to do the marking work when retraining a large amount of data. Therefore, this paper uses the migration learning method to train the model, which can easily apply the trained model to the new scene.

6.1 Training Process

In this paper, the Inception model is used to train and build its own image classifier under the Ubuntu system. The training process is as follows:

1) Download TensorFlow source code.

```
fancy@fancy-To-be-filled-by-O-E-M:~/AI$ pip install --ignore-installed --upgrade
 \
>  https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.4.0-cp35-cp3
5m-linux_x86_64.whl
```

2) Compile retain module.

```
fancy@fancy-To-be-filled-by-O-E-M:~/AI/tensorflow$ bazel build tensorflow/exampl
es/image_retraining:retrain
```

3) Train the model and set the step, image_dir, output_garph, and output_labels parameters.

```
fancy@fancy-To-be-filled-by-O-E-M:~/AI/tensorflow$ /home/fancy/anaconda3/bin/pyt
hon3.5 tensorflow/examples/image_retraining/retrain.py \
>      --image_dir ./data \
>      --output_graph ./model/retrained_graph.pb\
>      --how_many_training_steps 3000 \
>      --output_labels ./model/retrained_labels.txt \!
```

4) The accuracy of the training results reached 95.3%:

```
INFO:tensorflow:Final test accuracy = 95.3% (N=43)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
```

6.2 Result Verification

Use the Different ambiguity photos taken by the camera and then validate it with the optimized model:

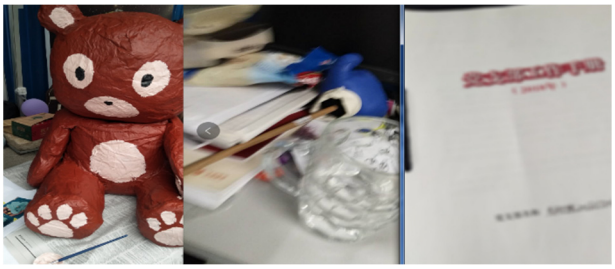

Figure 7. Different ambiguity photos taken by the camera.

(1) Clear picture, the probability that the model is judged to be clear is 0.95.

```
fancy@fancy-To-be-filled-by-O-E-M:~/AI$ tensorflow/bazel-bin/tensorflow/examples/label_image/label_image
 --output_layer=final_result --labels=./tensorflow/model/retrained_labels.txt --image=./tensorflow/model
/clear.jpg --graph=./tensorflow/model/optimized_graph.pb --input_layer=Mul
2018-12-10 08:55:07.790888: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instr
uctions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX
2018-12-10 08:55:09.199537: I tensorflow/examples/label_image/main.cc:250] clear (4): 0.95783
2018-12-10 08:55:09.227283: I tensorflow/examples/label_image/main.cc:250] blur 15 (1): 0.0403893
2018-12-10 08:55:09.227312: I tensorflow/examples/label_image/main.cc:250] blur 35 (3): 0.00175139
2018-12-10 08:55:09.227339: I tensorflow/examples/label_image/main.cc:250] blur 75 (2): 2.82074e-05
2018-12-10 08:55:09.227353: I tensorflow/examples/label_image/main.cc:250] blur 55 (0): 7.12179e-07
```

(2) The blurred picture, the model judges the probability of blur35 to 0.96.

```
fancy@fancy-To-be-filled-by-O-E-M:~/AI$ tensorflow/bazel-bin/tensorflow/examples/label_image/label_image
 --output_layer=final_result --labels=./tensorflow/model/retrained_labels.txt --image=./tensorflow/model
/blur35.jpg --graph=./tensorflow/model/optimized_graph.pb --input_layer=Mul
2018-12-10 08:57:46.261001: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instr
uctions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX
2018-12-10 08:57:47.539979: I tensorflow/examples/label_image/main.cc:250] blur 35 (3): 0.963098
2018-12-10 08:57:47.540016: I tensorflow/examples/label_image/main.cc:250] blur 75 (2): 0.0197798
2018-12-10 08:57:47.540025: I tensorflow/examples/label_image/main.cc:250] blur 15 (1): 0.0127847
2018-12-10 08:57:47.540032: I tensorflow/examples/label_image/main.cc:250] blur 55 (0): 0.00228672
2018-12-10 08:57:47.540039: I tensorflow/examples/label_image/main.cc:250] clear (4): 0.00205129
```

(3) The blurred picture2, the model judges the probability of blur77 is 0.93.

```
fancy@fancy-To-be-filled-by-O-E-M:~/AI$ tensorflow/bazel-bin/tensorflow/examples/label_image/label_image
 --output_layer=final_result --labels=./tensorflow/model/retrained_labels.txt --image=./tensorflow/model
/blur75.jpg --graph=./tensorflow/model/optimized_graph.pb --input_layer=Mul
2018-12-09 10:46:58.494891: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instr
uctions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX
2018-12-09 10:46:59.640157: I tensorflow/examples/label_image/main.cc:250] blur 75 (2): 0.93732
2018-12-09 10:46:59.640201: I tensorflow/examples/label_image/main.cc:250] blur 55 (3): 0.0603918
2018-12-09 10:46:59.640216: I tensorflow/examples/label_image/main.cc:250] clear (4): 0.00190633
2018-12-09 10:46:59.640227: I tensorflow/examples/label_image/main.cc:250] blur 15 (0): 0.000245186
2018-12-09 10:46:59.640238: I tensorflow/examples/label_image/main.cc:250] blur 35 (1): 0.000136798
```

The results show that this model can judge the ambiguity of the image. The model obtained by the fuzzy algorithm training can be applied to the project to judge whether the definition is up to standard.

7. Conclusion

This paper proposes a fuzzy algorithm for making TensorFlow training data, optimizes parameter control and trimming problems, and uses java language programming to batch process images in folders and automatically sort them into different directories, completing different the production of ambiguity image sets. The training results can be more accurate to distinguish the images with different ambiguities, which verifies the correctness and usability of this method.

References

[1]. Yann LeCun, Leon Bottou, Yoshua Bengio, et al. Gradient based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[2]. Information on https: //www.youtube.com/watch?v=FrKWiRv254g.

[3]. Vasilache Simina, Ward Kevin, Cockrell Charles et al. Unified wavelet and gaussian filtering for segmentation of CT images; application in segmentation of bone in pelvic CT images[J]. BMC Medical Informatics and Decision Making, 2009, 9(Suppl+1). W. Strunk Jr., E.B. White, The Elements of Style, third ed., Macmillan, New York, 1979.

[4]. Octavi Fors, Jorge Núñez, Xavier Otazu et al. Improving the Ability of Image Sensors to Detect Faint Stars and Moving Objects Using Image Deconvolution Techniques[J]. Sensors, 2010, 10(3).