One-shot Image Recognition via Siamese Neural Network

Tonji Luo, Peng Chen, Wansen Zhang, Yu Shi, Lingfeng Chen

Department of Electrical and Computer Engineering
University of California, San Diego

Abstract: One-shot learning is a challenging task in machine learning, in which we need to classify a test image given one or limited training data.In this paper, we implemented one-shot image recognition via Siamese Neural Network (SNN) with few data on the quick-draw program. Initially, we implemented two Convolution Neural Networks (CNN) as the body of SNN. Then we connected their output layers to the distance layer. While training, we calculated the contrastive loss function to update the weights of the whole network. While testing, given a new image, we calculated its Euclidean distance with different classes as outputs. Finally, we compared distances between classes to determine the similarity of the images. The SNN architecture shows its capability to learn features in limited training set, as well as identify new classes never trained before.

Keywords: Siamese Neural Network(SNN); One-shot learning; Quick-draw

1. Introduction

1.1. Overview

Now considering we are in the following scenario: we need to build up a face recognition system with only few images of employees in a company provided with a few pictures of each people. Traditionally, we need a large number of samples to train a machine-learning model to perform satisfactory classification result. However, we could not ask employees to submit tons of their photos. This is a typical problem of salable data required learning algorithm[1]. Large demand on training data and labels would bring a lot of issues. It would become time-consuming to obtain the raw data, as well as the pre-processing on the training data and feed them to the model.

In this paper, to address this issue, we focus on a similar one-shot learning problem associated with the Google quick-draw program. We selected five classes of doodle draws: diamond, bird, burger, house and cup as our training set. We aim to design a one-shot classifier which can tell whether the test image belong to our training class and which class it belongs to. After training, the model can tell the difference between an unknown new class and known class it is from once people draw an image treated as test image.

1.2. Proposed Solution

Traditionally, people would like to apply PCA or RDA before doing further classification. However, these approaches are assuming that the data are linearly separable. For non-linear separation processes, neural networks are quite powerful. In this paper, we are going to apply Siamese Neural Network (SNN) on this one-shot image recognition. This neural network can discriminate the image pairs. In other words, given any two images from our dataset, this network will predict whether those images are from the same class.

This approach is built upon twin CNN networks who share their weights with each other. As the core part of the SNN, they can learn which features are more important and automatically adapt the weights among the neurons. Applied ReLU layers as activation function in the learning architecture, our network can capture the non-linearities and utilize them to predict the labels more accurately without making many strong assumptions and imposing priors.

Intuitively, we modified the fully connected layer of CNN by increasing one more layer to simulate the non-linearity better. Since any continuous function defined on [0,1] can be represented in the following form[5]:

$$f(x_1, x_2, ..., x_n) = \sum_{j=1}^{2n+1} g_j(\sum_{i=1}^{n} \phi_{ij}(x_i))$$

And this form can be simulated by two-layer perceptrons, so we would like to adopt this idea in our paper.

1.3. Hypothesis

In this project, we did experiments to verify the hypotheses consisting the following five parts:

- SNN only needs few training samples to achieve satisfactory result.
- SNN could tell whether the test image is a new class by comparing the outputs.
- CNN could not tell the similarity between two images.
- CNN could not tell whether the test image is a new class.
- The fully connected layers at the end of CNN can help to simulate non-linearity functions.

2. Approach

2.1. Overview

In this section, we will mainly discuss about the architectures of CNN and SNN we used to solve this problem. But first of all, we should figure out why we want to use neural network as the fundamental part of image recognition. The advantages of neural network are:

- We do not have to do too much data-preprocessing since the network will choose features and adapt their weights automatically.
- We do not have to do some modification on data format such as one-hot encoding, normalization.
- Parameters that need to compute only include weigh ω and bias b.

While it still has some limitations:

- The parameters tend to be large as the layers and neurons of each layer increase, making the model complex.
- The large number of parameters would increase the occurrence probability of over-fitting.
- The gradient would decrease to 0 during back-propagation, which is impossible for weights to be updated.

These properties of traditional neural network imply it is not suitable for image recognition. Therefore, we need to introduce Convolution Neural Network.

2.2. Convolution Neural Network

In the traditional back propagating neural network, neurons are connected with each other. However, this is unnecessary and computationally expensive. Luckily, the convolution operation with a convolution kernel can extract the regional features such as some simple line pattern. So we can do the convolution on images and use the detected features to classify them.

2.3. Convolution + ReLU Layer

Convolution can detect region pattern efficiently. There are all kinds of kernels or filters that can differentiate features in images. Kernel can be used to detect patterns from simple vertical lines to a curve of a predefined degree. Most commonly, they can be used to detect line features by taking derivative on images, as shown in Figure.1.

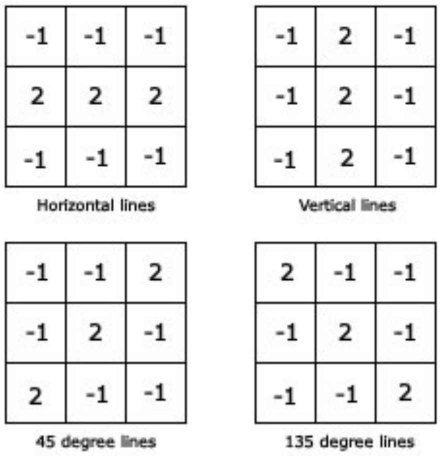


Figure 1: Line pattern detection

We need to design different layers to implement the features extraction, since different layer would be responsible for different tasks. Conventionally, the detection would be from simple to complex or from common to special. Usually, the more convolution kernels, the more features can be detected. We implemented three convolution layers to extract the features [2], followed by ReLU function.

The ReLU function can prevent the gradient from being 0 and is much easier to compute compared to sigmoid function or tanh function. Additionally, it can make the negative part output be 0, resulting the sparsity of the neural network and leverage the model to prevent overfitting [3].

2.4. Max Pooling Layer

Max Pooling is designed to reduce the dimension and drop the redundant features. After convolution, the max pooling can take the greatest value to represent the region values, as shown in Figure.2. Typically, this operation would not affect the result but to reduce the calculation. But sometimes it would drop useful information due to the wrong pool size or it is not appropriate to use max pooling for this problem. They should be applied right after every convolution layer.

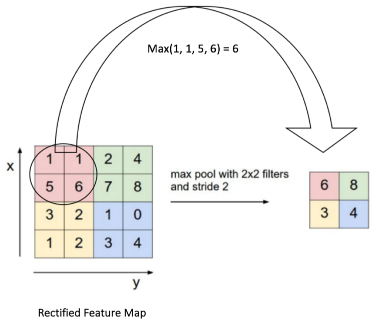


Figure 2: Max Pooling

2.5. Fully-Connected Layer

The fully connect layer is the last layer before the output layer. After three-time convolution and a max-pooling, the fully connect layer can finalize the features to a decision value that how possible the test image belongs to this class. The number of this layer can be more than one to capture the non-linearity of features. So we implement two fully connect layers in our model.Note that it is different from the conventional structure in CNN since we need to prepare to compute Euclidean distance for the SNN model.

2.6. The Architeture of CNN

As shown in the Figure.3, we implemented three convolution and ReLu layers, as well as one max pooling layers. Eventually, we used two fully-connected layers to perceive non-linearity.

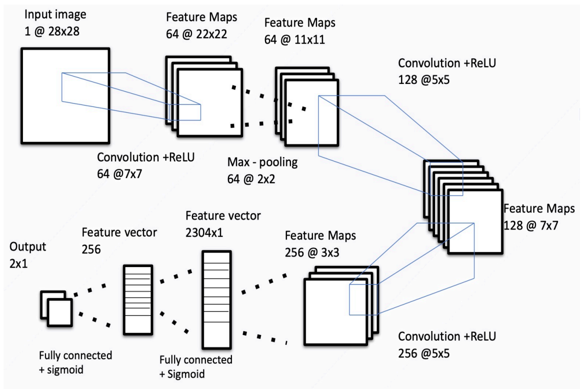


Figure 3: CNN Architecture

2.7. Siamese Neural Network

The loss function we used in the network is the contrastive loss based on Euclidean distance. The Euclidean distance is defined as:

$$D(X_1, X_2) = ||G_w(X_1) - G_w(X_2)||_2$$

Where X_1, X_2 is the input images, $G_w(X_1)$, $G_2(X_w 2)$ are the output of CNN. The loss can be explicitly expressed as:

$$L(W, Y, X_1, X_2) = (1 - Y)D_W^2 + Y * max(0, m - D_w)^2$$

Where Y is the label and m is a parameter called margin. The margin defines a radius around $G_w(X)$. Dissimilar pairs contribute to the loss function only if their distance is within this radius[4].

Actually it become quite simple after we build the CNN networks. We only need to connect two CNN together to share the weight and calculate the Euclidean distance of their outputs. The architecture of SNN is shown in Figure.4.

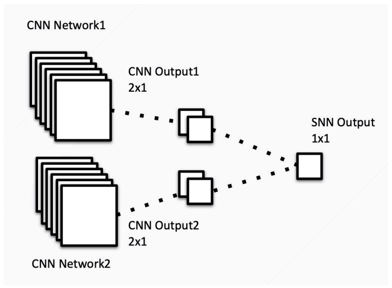


Figure 4: CNN Architecture

3. Experiments

3.1. Overview

In this project, we performed both CNN and SNN architectures to classify images. The applied CNN architecture is very similar the ones in SNN to give us an idea of the branch behaviour.

3.1.1 Dataset

The dataset we used are scripted for the Google "Quick Draw" dataset. The Quick Draw Dataset is a collection of 50 million drawings across 345 categories, contributed by players of the game Quick Draw!. The drawings were captured as timestamped vectors, tagged with metadata including what the player was asked to draw and in which country the player was located.

We choose five categories: Cup, House, Bird, Hamburger and Diamond as our training set. For each categories we selected 20 pictures and set their dimensions as 28×28. Notice that the training set we use is rather small for we want to explore the efficient of SNN when the dataset is smaller than what is usually used in Neural Network. The images from the five classes are shown in Figure.5:

Cup image

House image

Bird image

House image

Diamond image

Figure 5: Example images in the Dataset

And we also have a new class pig which should be tested in SNN testing process as in Figure.6:

Figure 6: Pig image

3.2. CNN

3.3. Training process

To test the performance of Convolution Neural Network, we use subsets with small number of images. The numbers of training images are 15, 25, 50 and 100. The five classes used in CNN training process are cup, bird, diamond, hamburger and house. One-hot encoder is chosen to represent the label. Since we only use a small number of training images, we use batch gradient descent while training.

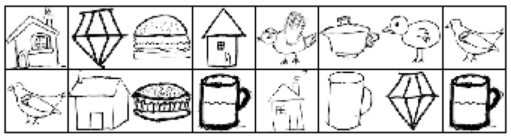
3.4. Test process

The number of images in test set is much larger than in training set. 1,000 images are randomly chosen from five class same as training set. We calculate the final test loss and accuracy after the training progress complete. In addition, a confusion matrix is calculated in the meantime.

3.5. SNN

3.5.1 Training process

To train the SNN model, we randomly choose 100 sets of images from the training set. Each set is composed by two individual images, if the two images are from the same class, we label it as 0 and label it as 1 otherwise. We can demonstrate this process by choosing 8 sets shown in Figure.7:



```
[[1.]
 [1.]
 [0.]
 [1.]
 [1.]
 [0.]
 [1.]
 [1.]]
```

Figure 7: Training set formation

The label of 0 or 1 in Siamese Neural Network is not only used to distinguish the images from the different classes but also stands for the magnitude of the Euclidean distance of two images.

To evaluate the network performance under limited training samples. We randomly selected 100 paired images in each epoch. To address the unbalanced data, we selected the same random amount of paired images (same or different) into the training set. Since sketches are not complicated images, we used the architecture with three convolution layers followed by two fully connected layers. Finally, the outputs from two CNN come to a distance layer, with a Contrastive loss function, the similarity of images can be determined. Therefore, if the Euclidean distance of two images are close to 0, it's safe to say they are from a same class. And they have higher probability to belong to different classes if the Euclidean distance between them is close to 1.

Run the test code with 10 pairs and plot the Contrastive Loss shown in Figure.8:

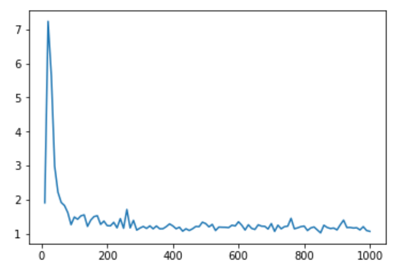


Figure 8: Contrastive loss of the test code

We can see that the converge speed of the SNN model is really fast, and 100 epoch is big enough for SNN to converge. Therefore, we will use 100 epoch for each experiments.

3.5.2 Testing process

The test process can be divided into two parts:

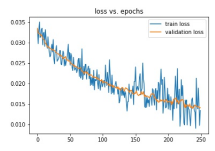
- Test the SNN model with the same 5 classes in the training set.
  In this section, we test the SNN with the same 5 classes in the training set. The test set is chosen from the same Google "Quick Draw" Dataset categories but contain 100 images in each class(different with the training images). Then we can test these 500 images using SNN model to compute the accuracy of each class and the overall accuracy.

- Test the SNN model with a new class(pig) which never shows up in the training process.
  In this part we test the model with the pig class which never shows up in the training process to see if the SNN model can distinguish whether the image belong to a new class.

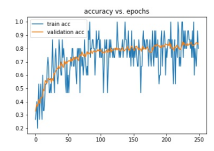
4. Results

4.1. CNN

4.1.1 Training Loss and Accuracy

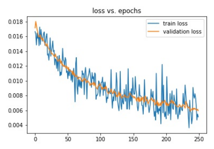


(a) Training loss

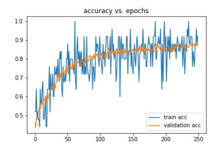


(b) Training Accuracy

Figure 9: Training Size: 15

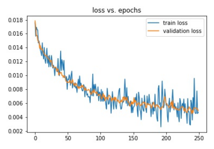


(a) Training loss

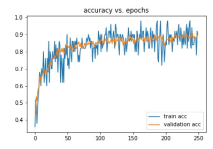


(b) Training Accuracy

Figure 10: Training Size: 25

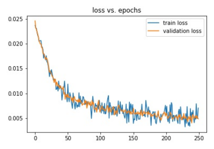


(a) Training loss

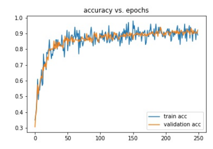


(b) Training Accuracy

Figure 11: Training Size: 50

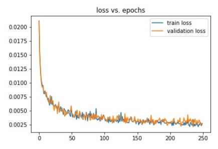


(a) Training loss

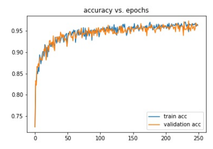


(b) Training Accuracy

Figure 12: Training Size: 100



(a) Training loss



(b) Training Accuracy

Figure 13: Training Size: 1000

4.1.2 Test Accuracy

The test accuracy for different number of training images are: (1) 15 images: 82.59% (2) 25 images: 87.11% (3) 50 images: 89.19% (4) 100 images: 90.79% (5) 1000 images: 97.89%

4.1.3 Confusion Matrix

As we can see from the confusion matrix, each category is easy to classify with high accuracy, limited confusion occurs from trained classes.

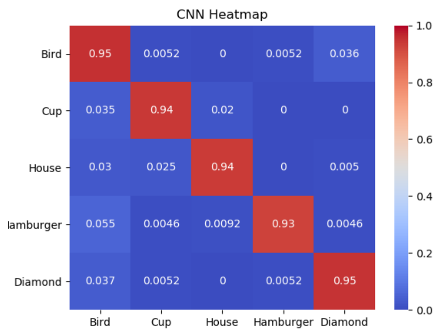


Figure 14: Confusion Matrix

4.1.4 Comment

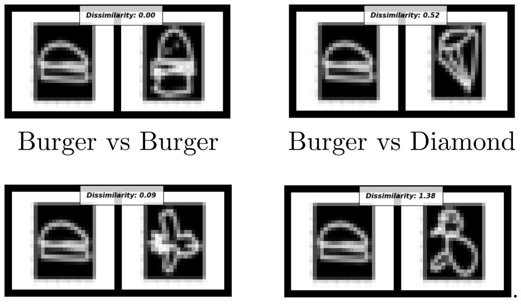
We can tell from the results that, with small number of training set, both loss and accuracy are worse than the large set. The curves look more chaos in the small training set. But the accuracy in small data set is higher than 80%. And with only 50 images, 10 for each class, the accuracy is about 90%. This may result from the small number of classes and the low complexity of images (i.e., small number of features).

From the confusion map, the performance of CNN model is excellent in classifying graffiti. The accuracy for each class is about 95 %(with about 500 training images).

4.2. SNN

4.2.1 Test results with origin 5 categories

The output of the Siamese Neural Network is the dissimilarity between two images(based on the Euclidean distance) shown as Figure.10:



Burger vs Burger

Burger vs Diamond

Burger vs Bird

Burger vs Bird

Figure 15: Test result

As it shown in the above figures, the test image for this time is a picture of a burger. This image has **0 Dissimilarity** with another burger and has different value of dissimilarity with images from other classes. We can use the dissimilarity value as the criterion to classify the test images.

The loss of SNN is as shown below: As we can see here, the convergence shows up quickly. The training set is labeled in blue and validation set is labeled in orange. The fast convergence without further decrease of loss indicates the behaviour of the SNN seems not be improved via more iterations. This also can be attributed to the varience in the training set for the same category make it more difficult to train. See figure. 16

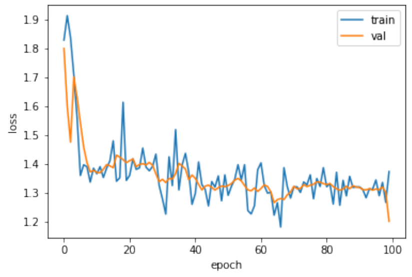


Figure 16: SNN loss

The SNN classification heatmap as shown below: See figure. 17 The capability of SNN to classify objects is not as good as CNN. Some terms are easy to be mislabeled such as house and cup. The accuracy for each class is around 70% to 80%.

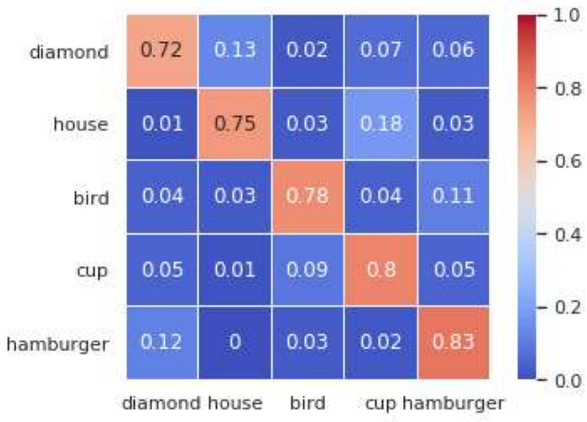


Figure 17: SNN classification heatmap

As we can see from the comparison, CNN has a better performance on the classification. SNN shows its ability in estimating the similarity between two images, especially for limited training set. It even can handle unknown classes which is not presented in the training set. Due to the great variance of training set, SNN has a larger dependency on the images have shown up, so the feature caught from SNN is not as accurate and detailed as CNN.

4.2.2 Test results with the new class

In this section, we use the pig images which have never showed up in the training process as the test set, the test results as shown in Figure.13:

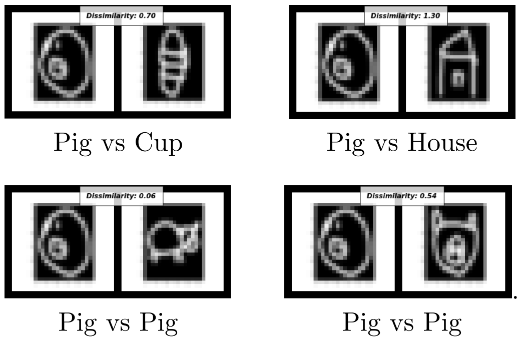


Figure 18: New class Pig vs other classes

The above figures show that the new test image pig have large dissimilarity with cup and house(other classes) and have rather small dissimilarity with other pig images.

If we use the same new class in the test process in CNN, it could be only classified into one of the existed classes which is not correct. In the mean time, SNN can identify this image into a brand new class, this is a unique advantage of SNN.

The above result clearly shows that the SNN algorithm is very effective in determine whether a new test case belong to the existed classes.

5. Conclusion

5.1. Comparison of SNN and CNN

From the above result, we can draw the following conclusion:

- The accuracy of CNN is higher than SNN under this circumstance.

  From The testing result, the accuracy of CNN with 100 images is 90.79% which is higher than the overall test accuracy of SNN(77.6%). This shows the weakness of SNN in the classification problem with respect to CNN.

- SNN shows a good performance in estimating the similarity between two images.

  SNN can determine the similarity between two images. Therefore, SNN is more flexible than CNN which could just classify a test case into one certain class.

- SNN can handle unknown classes which are not in the training set.

  This particular feature for SNN can identify if a certain object is the same as the existed classes. That's why the SNN algorithm is mainly used in facial recognition.

- SNN has highly dependency on the training set.

  For SNN can quantified the similarity between two images, it has high demand on the training set. If the images contain in the training set vary a lot, the accuracy of the SNN model will be affected.

5.2. Difficult classification issues in SNN

The major issues we face during this classification are two aspects See figure. 19: 1. the high similarity between figures in different classes, such as sketched house and cup; 2. the diversity between figures in the same class.

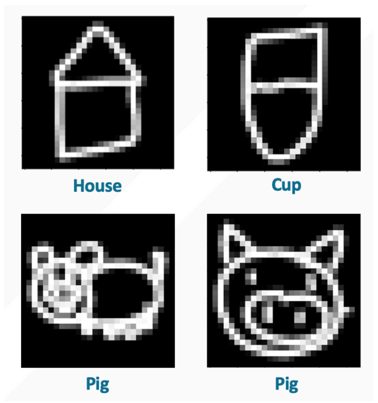


Figure 19: Difficult samples in SNN classification

This kind of training pairs can be very confusing for the SNN model. Although CNN will face the same problem, it would have bigger influence on Siamese neural networks. Therefore, one character of SNN is that this model is hard to train but good to use.

References

[1] Li Fei-Fei, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):594–611, **2006.**

[2] Koch, G., Zemel, R., Salakhutdinov, R. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop* (Vol. 2).**2015.**

[3] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, **2009.**

[4] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with applica- tion to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, **2005.**

[5] Kolmogorov, A. N. "On the Representation of Continuous Functions of Several Variables by Superposition of Continuous Functions of one Variable and Addition," *Doklady Akademii. Nauk USSR,* 114, 679-681.**1957.**