Image Classification with One-shot Learning

Nguyen Nicolas and Md Sakib Rahman
School of Electronics, Information & Electrical Engineering
Shanghai Jiao Tong University
(Dated: December 18, 2020)

[Deep Learning algorithms are quite familiar and efficient to classify images when a large amount of data is provided. But when it comes to learn from a small amount of data, these kinds of algorithms aren't enough efficient. Training model with small labelled data may create over-fitting issues. Few-shot learning *(FSL)* algorithms are proposed to tackle this problem. During this project, we focused on a particular type of *FSL* algorithm, called *One-shot Learning* algorithms. Particularly, we will rely on the Siamese neural network as a CNN.]

I. INTRODUCTION

As a human being, when new objects are presented to us, we quickly pickup patterns, shapes and other main characteristics of these objects. Then when we are presented with same kind of objects later, we could recognize them without any difficulty. It means that for human, we only need a few examples of each objects to classify them into categories or classes.

A. image classification with One-shot Learning

Deep learning have made major advances in many fields such as image classification [1], speech recognition [2] and language learning [3] but the main issue of this kind of algorithms is that they require a large amount of data. Moreover, these kind of algorithms are trained to accomplish a specific task. In the case of standard image classification, for every input image of different classes, the algorithm generates different numbers representing the probability of the image belonging to each class. Gathering and labelling thousands of images is very difficult and time consuming. On the other hand, if we want to introduce a new class or example we have to retrain the model again. Recently, there have been many research on algorithms that could bypass this issue by requiring a few examples of each classes. These kind of algorithms are applicable in many fields that couldn't collect a large amount of data:

- Face recognition : Collecting data of people is an ethical issue of private life. Thus, One Shot Learning is adapted to this kind of issue.

- Signature verification : It is also impossible to provide a large amount of data for Signature verification. FSL algorithms could be useful for banks in banknotes verification or any kind of document verification.

B. The Dataset

In this project we used the *Omniglot* [4] dataset (*figure 1*) which contains 1623 hand drown characters from 50 alphabets. But for every characters (which is seen as one class) there are just 20 examples.

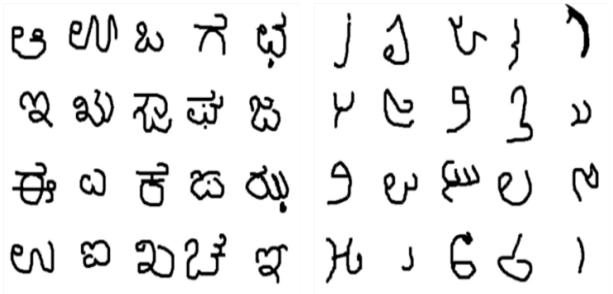

Figure 1. Examples of Ominglot characters

II. RELATED WORKS

It seems that research on One-shot Learning is less developed that other kinds of Few-shot Learning methods (*e.g.* meta learning algorithms). The first papers about one-shot learning algorithms have been published by *Li Fei-Fei et al. Lake et al.* addressed the issue of One-shot Learning from the point of view of cognitive sciences, by implementing an algorithm for character recognition called Hierarchical Bayesian Program Learning (*HBPL*) [5]. We especially exploited *Koch et al.* with their subject concerning one-shot image recognition utilizing a Siamese Neural networks [6]. *Vinyals et al.* developed a model of one-shot learning algorithm using Matching Networks as a CNN. This algorithm relies on a principle of transfer learning [7]. *Sounak et al.* have used one shot learning to build an offline signature verification system [8] which is totally writer independent .
Lastly, the *Ominglot* dataset was provided *Lake et al.* and it makes an ideal dataset for testing small-scale one-shot classification, as we will discuss below.

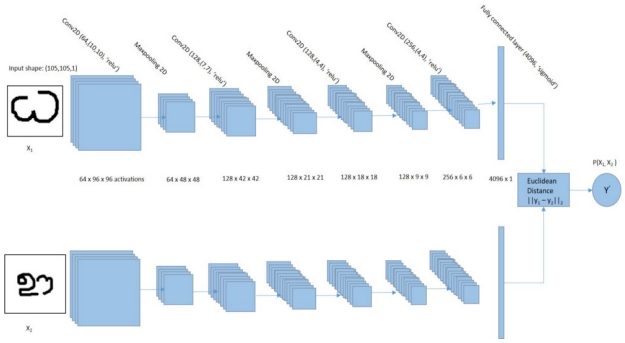

Figure 2. Siamese Network

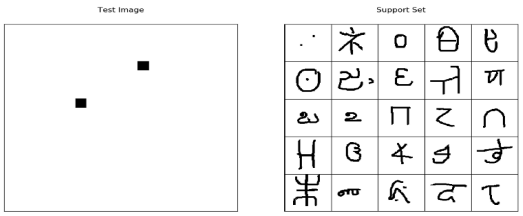

Figure 3. Test image and Support Set

III. METHOD

Let consider a small labelled training set S, which has N examples, and each vectors has the same dimension with a distinct label y.

$$S = \{(x_1, y_1), ..., (x_N, y_N)\} \tag{1}$$

The problem is to classify $\hat{x}$, *i.e.* predict which $y \in S$ is the same as $\hat{y}$

In our project, we restrict the issue of image classification to **character classification**. But the method can be applied to any sort of images.

A. Model Architecture

If we use a CNN for a classical deep learning algorithm, it will obviously overfit with our data given the amount of data we exploit for one shot learning.

The term *Siamese* means twins. The two CNN shown above are not different networks but these are not copies of the same Network: They share the same parameters.

The principle of this method is to give to the neural net 2 images and train it to guess whether these 2 images belongs to the same class or not. This means that the Siamese Networks outputs the probability that the 2 images share the same class(*figure 2*).

The two input images are passed trough the CNN to generate a same length feature vector. Then, due to a **metric function** (which we will explicit later) and a **sigmoid function** we can compare these two vectors to see if the 2 images belong to the same class or not.

Let consider x_1 and x_2 the 2 images in our dataset, and let consider "x_1 and x_2 belongs to the same class" as the notation $x_1 \circ x_2$.

The output of this Network, and a sigmoid function gives it between 0 and 1, as a probability. By convention, we refer to 1 when the images belong to the same class, and 0 in the other case. So, our model learns from a similarity function.

$$d(x1, x2) = degree\ of\ difference\ between\ images \tag{2}$$

We are going to train our model batch by batch and each minibach have i indexes the i^{th} minibatch. When $d(x_1^{(i)}, x_2^{(i)}) = 1$, we can assume that x_1 and x_2 form a same class and $d(x_1^{(i)}, x_2^{(i)}) = 0$ then we can assume images are not in same class. So the loss function will be following form:

$$\begin{aligned} L(x_1, x_2, t) = & \ t.log(p(x_1 \circ x_2)) \\ & + (1-t).log(1 - p(x_1 \circ x_2)) \\ & + \lambda.||w||_2 \end{aligned} \tag{3}$$

The SiameseNet classifies the test image as whatever image in the support set it thinks is most similar to the test image. We can query the model network using x, x_c as our input range from $C = 1$ to C^2. So, we will calculate the maximum similarity.

$$C(\hat{x}, S) = argmax_c P(\hat{x} \circ x_c) \quad x_c \in S \tag{4}$$

B. Nearest Neighbor Model

A simple way of doing classification is by the K-Nearest-Neighbour Method, but there is only one example per class so it is $1 - NN$. By defining as C our classifier :

$$C(\hat{x}) = argmin_{c \in S} ||\hat{x} - x_c|| \tag{5}$$

In an N-way one shot learning, we compare a test image with N different images and select that image which has highest similarity with the test image as the prediction.

C. Random Model

The random model aims to verify that our model is at least a better model that a model which makes random predictions.

D. Mapping the problem to binary classification task

This problem can be mapped into a supervised learning task where out dataset contains pairs of (x_i, y_i) where x_i

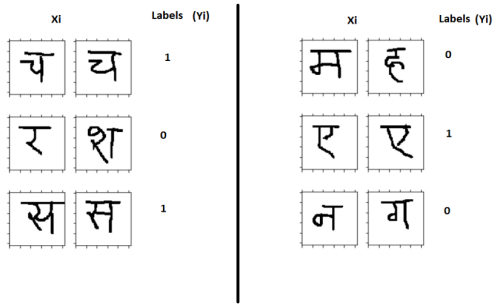

Figure 4. Comparison of images

is the input and y_i is the output. We will generate pairs randomly from all alphabets in the training data.
Thus we need to create pairs of images along with the target variable to be fed as input to the Siamese Network (*figure 4*).

E. Another approach : Triplet Loss

1. Loss function

It is possible to use another fundamental approach by exploiting an other metric function, as described in [9]. In fact, given that we want to compare 2 images and associate a small distance embedding for same-class images and a large distance embedding for images that don't have belong to the same class, we can define them with the following **triplet**:

- A starting picture : **the anchor**
- A picture from the same class as the anchor : **the positive**
- A picture from a different class to the anchor : **The negative**

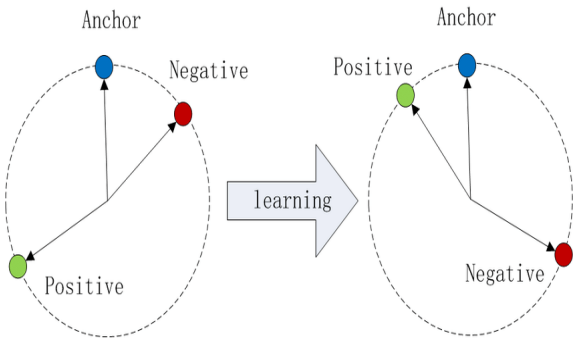

Figure 5. Triplet Loss

Let's denote these 3 images respectively by A, P, N:

$$dist(A,P) < dist(A,N) \ i.e. \ dist(A,P) - dist(A,N) < 0$$

But in order to get a better result we have to implement a margin. In fact, without a certain margin, the network would learn to simple solutions. This can be seen as a hyper parameter. So let's introduce a **margin** such that:

$$dist(A,P) - dist(A,N) + margin < 0$$

Then the new loss function can be defined as:

$$\mathcal{L} = max(dist(A,P) - dist(A,N) + margin, 0)$$

2. Triplet Batch

In order to construct a triplets for the training set every sample of our batch will contain 3 images of the dataset : A, P and N.
It is possible to take our triplet randomly. But we need to implement our Siamese Net in Order to clearly differentiate our classes. Based on how we defined our loss function, we can define 3 categories of triplets :

- **Easy Triplets** : Triplets for which $\mathcal{L} = 0$: $d(A,P) + margin < d(A,N)$. So, the loss is 0.
- **Hard Triplets** : triplets where the negative is closer to the anchor than the positive: $d(A,N) < d(A,P)$
- **Semi-hard Triplets** : triplets where the negative is not closer to the anchor than the positive, but which sill have positive loss: $d(A,P) < d(A,N) < d(A,P) + margin$

According to these given information, We have trained our Neural Network model with the library **Keras** and with a pre-existing code (see the references).

IV. EXPERIMENTS AND RESULTS

In this section we describe the result of our experiments. We used the library **Keras** on Python to implement our algorithm. The Keras library in Python makes it much easier to build a CNN.

A. Validation of the model

For every pair of input images, our model generates a similarity score between 0 and 1. But this score is relative, meaning that this score doesn't give a precise

Figure 6. Example of a 4-way one shot learning

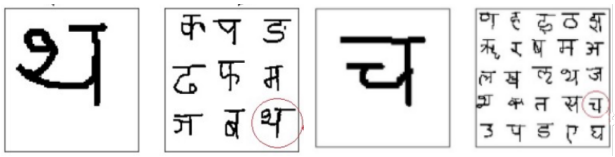

Figure 7. Repeating the procedure for different N

estimation of whether our model is precise or not.
A more precise way to assert the precision of our model is the **N-way one shot Learning**

Here, the same character is compared to other 4 characters which are different. Consider that the score-comparison results are s_1, s_2, s_3 and s_4 (*figure 5*). If our model is well-trained, we must have s_1 as the number which is the nearest from 1 because s_1 refers to the most similar pair of character (here it is the only similar pair above the 3 others).
Thus, if s_1 is considered as the maximum score, we consider the pair 1 as a 'correct prediction' and the others as 'incorrect prediction'. We repeat this k times and the percentage of correct predictions is :

$$\text{percentage of correct prediction} = \frac{100 * n_{correct}}{k}$$

Now we can do the same for different N : Smaller values of N will lead to more correct predictions and large values of N(*figure 6*) will lead to less correct predictions when repeated multiples times.

B. 1-Nearest Neighbor model

Here, the inputs image are represented as matrices : a flatten function transforms the matrices as vectors. Let 2 image be represented (after b passed through the CNN) as 2 vectors A and B such as:

$$A = \begin{bmatrix} a_1 \\ a_2 \\ ... \\ a_p \end{bmatrix} \quad B = \begin{bmatrix} b1 \\ b2 \\ ... \\ bp \end{bmatrix}$$

And employing the L_2 distance gives :

$$distance(A, B) = ||A - B|| = \sqrt{\sum_{i=0}^{p}(a_i - b_i)} \quad (6)$$

Thus, in N-way one Shot learning, we compare the L_2 distances of the test images with all the images of the support set. Then we identify the character for which we got the minimum L_2 distance.Similarly to N way one shot learning, we repeat this for multiple trials and compute the average prediction score over all the trials.

C. Evaluation and metrics

During training test, we evaluate how far the distance embedding from each class are from each another. But we must have to evaluate the whole dataset but there is just to check that the network is converging correctly for all the classes.

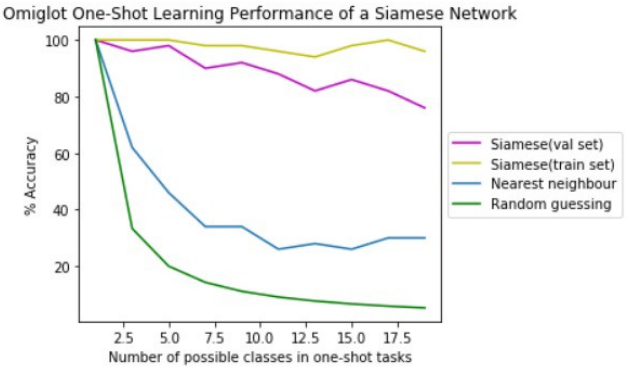

Figure 8. Comparison of the different methods

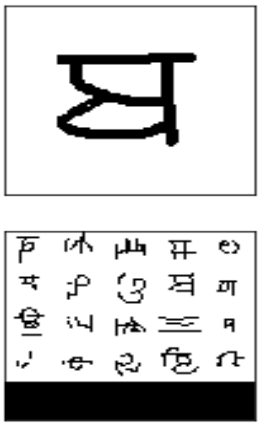

Figure 9. Generated output

Image Verification, being the task you're given two images and you have to tell if they are of the same per-

son/class or category. Since in context of the problem we have in hand of image verification, classifying similar images of characters as same is the main problem we need to solve, therefore choosing **accuracy** is the right metric we should use to validate the performance of our model on new unseen data (*Figure 8*). Here model is train with 20,0000 iterations, one iteration being one full pass over the data set.

D. Observations

We tested the N-way testing for different values of N : 1,3,5,...,19. For each N-way that we tested, 50 trials were performed and then we computed the average accuracy over these 50 trials.
The figure presents the comparison between the 4 models.

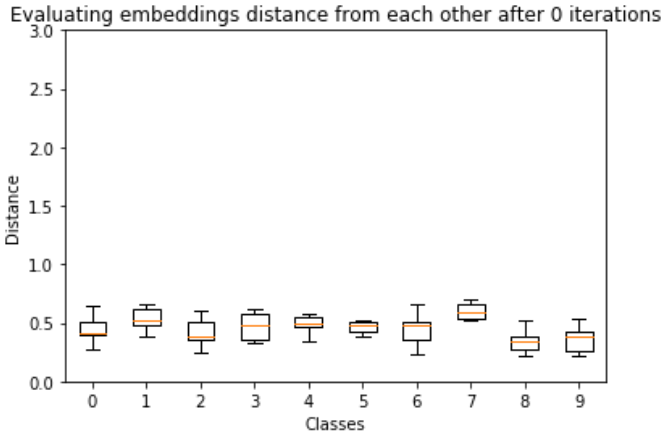

Figure 10. Evaluation Metrics using triplet loss

We observe that the Siamese Model performs much than the Nearest neighbor model. However, there is some gap between the results on the training set and the validation set which indicate that the model is over fitting. This may be occur due to learning decay for each layer(as it is being mentioned in the original paper). On the other hand the number of iterations during training time can make this overfit issue. Choosing accuracy is the right metric we should use to validate the performance of our model on new unseen data.

In the figure (*Figure 10*), the distances between classes take values between 0.5 and 1.5 approximately, indicating that their respective embedding are better produced by the network.

V. CONCLUSION

In this paper, we have presented a learning method called one-shot learning using Siamese network. The method is outlined new results comparing the performance of the difference methods and comparing existing state-to-art on Omniglot dataset. We try to demonstrate how well our model perform on that dataset with different methods. Although our model still there have some outfitting issues due to learning decay for each layer in the model. We feel this will help other to improve the existing methods to keep improving in the future work.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.
[2] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: an overview," in *2013 IEEE international conference on acoustics, speech and signal processing*, 2013.
[3] A. Hassan and A. Mahmood, "Convolutional recurrent deep learning model for sentence classification," *IEEE Access*, 2018.
[4] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the annual meeting of the cognitive science society*, 2011.
[5] L. Fe-Fei *et al.*, "A bayesian approach to unsupervised one-shot learning of object categories," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 1134–1141, IEEE, 2003.
[6] G. R. Koch, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, 2015.
[7] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," 2017.
[8] S. Dey, A. Dutta, J. I. Toledo, S. K. Ghosh, J. Llados, and U. Pal, "Signet: Convolutional siamese network for writer independent offline signature verification," 2017.
[9] X. Dong and J. Shen, "Triplet loss in siamese network for object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.