Architectures for Siamese Neural Networks

Bharathikannan Nithyanantham
University of Siegen
bharathikannan.nithyanantham@student.uni-siegen.de

Abstract

Traditional supervised machine learning approaches require large amounts of data for each category and all the categories need to be present in the training data. This approach is not suitable for applications where we have less amount of data for each class. One of the approaches for solving this problem is to use distance-based methods like the siamese neural network. Siamese networks tackle the problem of making classification on very small samples. This is a very hard problem and some parameters in the architecture needs to be carefully chosen to achieve good results. In this paper, we explore different architectures and parameters for siamese neural networks and analyze the impact of those on the performance of the model.

1 Introduction

Siamese neural networks have two identical networks with the same weights and they work on two different input images. It produces two feature vectors and these vectors are passed through a similarity function to get the similarity. Siamese networks have different loss functions. The most common ones are contrastive loss and triplet loss. In this research, the contrastive loss is used. Contrastive loss minimizes the l_1 distance between two image features.

For learning with contrastive loss, two images $X1$ and $X2$ are given to the siamese model f to get the feature representations, $f_w(X1)$ and $f_w(X2)$ respectively. Then l_1 distance between $f_w(X1)$ and $f(X2)$ is calculated. Then a dense layer with a sigmoid function is used to get the final similarity score. If the output is close to 1, then two images are similar and different if close to 0.

1.1 Dataset

The Omniglot dataset collected by Lake *et el.* (1) is used in this research. It is one of the most widely used dataset for few-shot learning. Omniglot has 1623 classes which makes this dataset hard and more suitable for few-shot classification. It has 50 alphabets, and each alphabet has many characters. There are 20 samples for each character and each sample is written by a different person. The training set has 30 alphabets, 964 characters, and 19,280 samples. The test set has 20 alphabets, 659 classes, and 13,180 samples. Each character is considered a different class. If two characters are from the same class, then the output label is 1 and 0 if they are from a different class.

2 Related work

Bromly and LeCun introduced Siamese Neural Network in their paper Bromley *et el.* (2). They used this method for signature verification. Two identical networks with the same weights extract features for signatures and they measured the similarity between two vectors using cosine similarity.

Chopra *et el.* (3) presented contrastive loss function for training the model. Instead of using cosine similarity, they learned the similarity metric by specifying that during training. They decreased the $L1$ norm distance of the feature vectors of the same classes and increased the $L1$ norm distance of

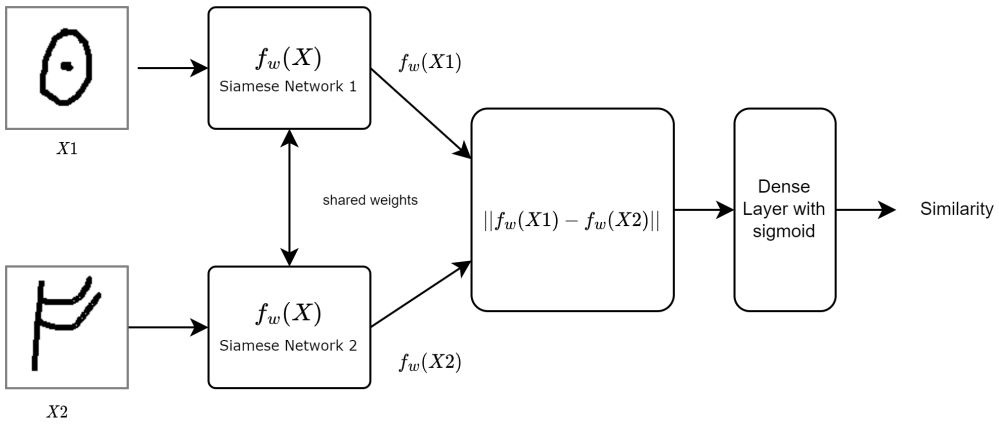

Figure 1: Siamese neural network block diagram

different classes in the feature space. Finally, a sigmoid function is used to get the similarity score. This method is best suitable when we have new categories that were never seen during training.

Koch *et el* (4) presented a convolutional neural network architecture for one shot classification. They achieved very good results and compared their performance with other state-of-the-art networks.

3 Methodology

My research is to answer the question "What is the impact of the architecture details on the performance of a Siamese Neural Network?". To answer this question, models with different parameters, hyperparameters, and architectures are trained and the performance of each model is analyzed.

3.1 Baseline model

After experimenting with many architectures, the model architecture which gave the best performance is selected as a baseline model. It is used to test other models with different architectures and parameters. The model consists of a backbone Convolutional neural network (CNN), a lambda layer, and a dense layer. The backbone consists of 3 convolutional layers, 3 Max Pooling layers, and a Dense layer at the end. The convolutional layer has output filter sizes of 32, 64, and 128 successively and each layer has $l2$ regularization. Each hidden layer has Relu activation function, batch normalization, and dropout with dropout rate 0.3. Exponential decay is used as a learning rate schedule. Adam is used for optimization. Two images are passed through the backbone network to get their feature representations. Then it is passed into the lambda layer for getting the $l1_{norm}$ between the feature vectors. Then it is passed to the fully connected dense layer with sigmoid activation function to get the final similarity score. This architecture is inspired by Koch *et el* (4). The first convolutional layer has 32 filters instead of 64 and it is the only difference in our architecture and the hyperparameters vary.

3.2 Training and evaluation

TensorFlow-GPU framework is used for all the experiments. All the models are trained on batches of batch size 128 for 5000 iterations on GPU. Each batch has equal amounts of positive and negative pairs. The evaluation is done for 1 shot 20-way for all the models. None of the test images are seen during training.

4 Experiments

4.1 Baseline model

The baseline model gave around 80% training accuracy and 75% validation accuracy. This is still a very good accuracy because 1 shot 20 way is a very difficult problem. The model increased the accuracy at every iteration. The validation loss was close to training loss and showed that the model didn't overfit. Overall the model training was good and gave a good performance. This baseline model is selected to test all other models with different architectures and hyperparameters.

4.2 Final embedding layer

The first layer in the neural network recognizes basic features such as edges and basic shapes. The next layers aggregate that information and learn shapes and higher-level features. The deeper layers learn high-level features. The final embedding layer has the encoded representation of the character in a lower dimension.

4.2.1 Fewer neurons in the final embedding layer

It will be difficult for a siamese network to encode many details with fewer feature maps. The dense layer in the backbone network is changed from 256 to 128 and all other parameters are kept the same as the baseline model. The validation accuracy dropped to 65%. The final embedding layer is again changed to 64 and the first conv2D layer is removed. Now the amount of information encoded in model 3 will be very less. Due to this the validation accuracy dropped to 50%.

4.2.2 More neurons in the final embedding layer

If the size of the final embedding layer is too large, then it tries to encode fewer details in a large dimensional space. It slows the convergence and affects the performance. The final embedding layer size of the baseline model is changed to 512 and all the parameters are kept same. It had a similar performance to baseline. An extra conv2D layer with a filter size of 256 is added and the performance was same again. The final embedding layer size is changed from 512 to 1024. The validation accuracy dropped to 65%. This shows that an increase in the size of the final embedding layer affect the performance. With more parameters, it also takes more time to converge.

4.3 Regularization

Overfitting is one of the common problems in machine learning. Our model can perform well on training data but it may not perform well on test data. And it becomes worse if we haven't seen the data during training time. For one-shot learning, the dataset is very small and none of the data is seen at the training time. So there is a high chance that the model will overfit the training data. Regularization is used to prevent overfitting. There are many regularization techniques. In this research two methods are used to prevent the siamese model to overfit. They are dropout and L2 regularization. Dropout randomly drops some hidden units in each layer. L2 regularization adds an extra penalty term to the cost function.

A model is trained without both L2 regularization and dropout. The performance was low and got only 30% validation accuracy. A model is trained only without L2 Regularization and the performance was very poor. The validation accuracy was below 10%. Dropout affects the performance at the start, so it had a very poor performance. A model trained only without dropout got around 65% validation accuracy. The validation accuracy was not close to the training accuracy and it had slight overfitting. Dropout may not be useful at the start, but it is very helpful at the end of the training process. So for a siamese network, we need both L2 regularization and dropout to prevent the model from overfitting.

4.4 Learning rate

Learning rate decay helps the model to learn fast initially so that it can avoid small local minima and in the end, it makes the model to converge to a minima by avoiding oscillation. The baseline model is trained with exponential learning rate decay.

A model is trained with a fixed learning rate of 0.0001. The model achieved high accuracy, but the model didn't generalize well. The training accuracy was not close to validation accuracy. Still, the performance is comparable to the baseline model. A model is trained with a fixed learning rate of 0.001. The accuracy fluctuated a lot and it didn't converged to a minima. The performance decreased significantly. Then a model is trained with an even higher learning rate of 0.01 and the performance was very poor. This shows that a smaller learning rate is good for the model as it reduces oscillation at the end. An exponential learning rate is used in the baseline as it reduces the learning rate at the end and at the same time accelerates the learning at the start. Adam learning rate is used for optimization and it also has an adaptive learning rate optimization, but when it is combined with an exponential learning rate, it gives much better results.

4.5 Batch normalization

Batch normalization is used to increase the speed of the training and makes it more stable during training. The distribution of the activations in one layer affects the successive layers. If we learn for one distribution and if it changes then it won't generalize well. Batch normalization makes each layer to learn independently by reducing the covariance shift. In siamese networks, there is a lot of variation in the distribution of train data, and the test data is never seen during training time. The baseline model is trained without batch normalization and the validation accuracy was only around 30%. This shows that, batch normalization is very important for siamese networks.

4.6 Weight initialization

Deep neural networks suffer from exploding and vanishing gradients. If weights are initialized with zeros then all units in a layer will learn the same thing and it becomes a linear function. If weights are initialized with large values then in backpropagation, the gradients increase exponentially at every backward layer and explode at some point. If it is small then the gradients decrease exponentially and there will be no update in the weights at the initial layers. Proper initialization of weights is important to prevent this. Xavier uniform method is used in the baseline model. Weights are initialized to zeros to the baseline model and the performance was very bad. The model didn't learned anything. This shows that a proper weight initialization is important for training the model.

4.7 Multi-layer perceptrons

Multi Layer Perceptron is a simple feed-forward network. It consists of an input layer, output layer, and many hidden layers. It is used for simpler tasks. The backbone convolutional network in the siamese network is changed to multi-layer perceptrons. The network has 3 hidden units with batch norm and dropout for each layer. The size of each layer is 2048, 1024, and 512 successively. It got only around 30% 20-way validation accuracy. It has a lot of parameters and as a result it took a lot of time to train. The performance increased very slowly. Multi-later perceptrons are not good for image data because each pixel doesn't know the neighboring pixel and they have more parameters to train. Convolutional neural networks dominate computer vision field and the result shows convolutional network outperforms multi-layer oerceptrons.

4.8 Accuracy of the model with different k way

The K-way denotes the number of classes present in the support set. The problem gets tougher and tougher with more classes. The probability of choosing one image from two images is much higher than choosing an image from 20 images. The baseline model accuracy is taken for 20-way. The baseline model is evaluated for different k values. The model performance decreased with increase in the k value. The value of k depends upon the application and this result shows how this value can affect the performance of the model. For applications like face verification (1:1), an image is compared with another image. But in face recognition (1:k) a single image is compared with many images of different classes making it a little harder problem.

Table 1: Results for all the architectures is tabulated here. Accuracy is taken for 1 shot 20-way in both training and validation set.

		Train Accuracy (%)	Val Accuracy (%)
Baseline Model		81.3	73.2
Final Embedding layer	Dense - 128	81.6	64
	Dense - 64	56	52.4
	Dense - 512	80	75.6
	Conv2D (256) + Dense - 512	73.6	74
	Conv2D (256) + Dense - 1024	73.2	65.6
Regularization	without Dropout	74	66
	without L2 Regularization	8.4	8.4
	without both methods	26	30
Learning rate	0.0001	92.4	76.8
	0.001	59.6	47.2
	0.01	22.8	20.4
Weight Initialization	Zero Initialization	60	50
Batch Normalization	without Batch Normalization	40	31.2
MLP	Dense (2048, 1024 and 512)	42	34.4

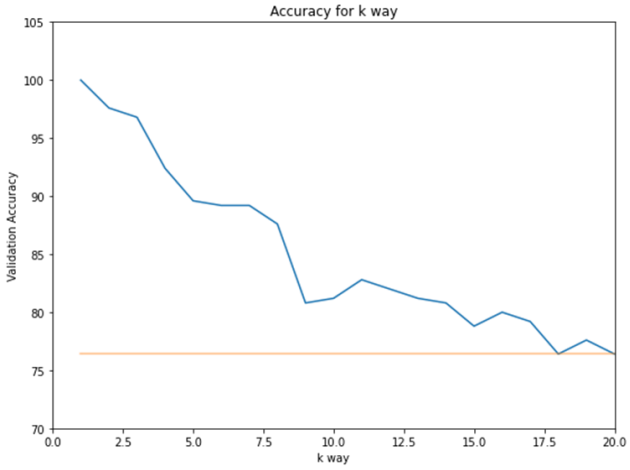

Figure 2: Accuracy for different k values for the baseline model

5 Conclusion

Different architectures are implemented and the impact of those on the performance is outlined. We have argued the reason why some parameter and architecture is very important for a siamese network than traditional supervised methods. The performance over different k values is presented. The best performing model is the baseline model and it shows that a carefully chosen architecture for the siamese network improves the performance. In future, different datasets like mini-ImageNet and AT&T which have more complex patterns can be used. Loss functions also have a great impact on the model performance. Triplet loss, hard triplet loss, or quadruplet loss can be used.

6 Acknowledgments

A special thanks to the Scientific Computing service of ZIMT from the University of Siegen for providing the computing power to train all the models. More than 20 architectures were tested for this research and each model took around 20 minutes to train in NVIDIA Tesla V100 GPU. Code is inspired from (https://www.kaggle.com/code/kartik2112/omniglot-dataset-siamese-networks). Omniglot data set is from (https://github.com/brendenlake/omniglot).

References

[1] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," vol. 350, no. 6266, pp. 1332–1338. Publisher: American Association for the Advancement of Science.

[2] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Advances in Neural Information Processing Systems*, vol. 6, Morgan-Kaufmann.

[3] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546 vol. 1. ISSN: 1063-6919.

[4] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," p. 8.