Siamese Neural Networks for One-shot Image Recognition

By: Soham Bhowmick (2019A8PS0350P)
Mehul Gulati (2019A7PS0046P)
Devanshh Agarwal (2018B4A30889P)

Author Affiliation

The authors are as follows:

- Gregory Koch - Department of Computer Science, University of Toronto. Toronto, Ontario, Canada
- Richard Zemel - Department of Computer Science, University of Toronto. Toronto, Ontario, Canada
- Ruslan Salakhutdinov - Department of Computer Science, University of Toronto. Toronto, Ontario, Canada

Paper Description : Aim

- The paper aims to solve the one-shot classification problem by exploring siamese neural networks which employ a unique structure to naturally rank similarity between inputs.
- Once the network is trained, the model is able to identify various different classes after being provided just a single training example.
- The model was able to outperform other deep-learning models with near state-of-the-art performance on one-shot classification tasks.

Paper Overview : Methodology

- Our standard model is a siamese convolutional neural network with L layers each consisting of N_l units,where the hidden vector in layer l for the first twin is represented by $h_{1,l}$ and similarly as $h_{2,l}$ for the second twin.

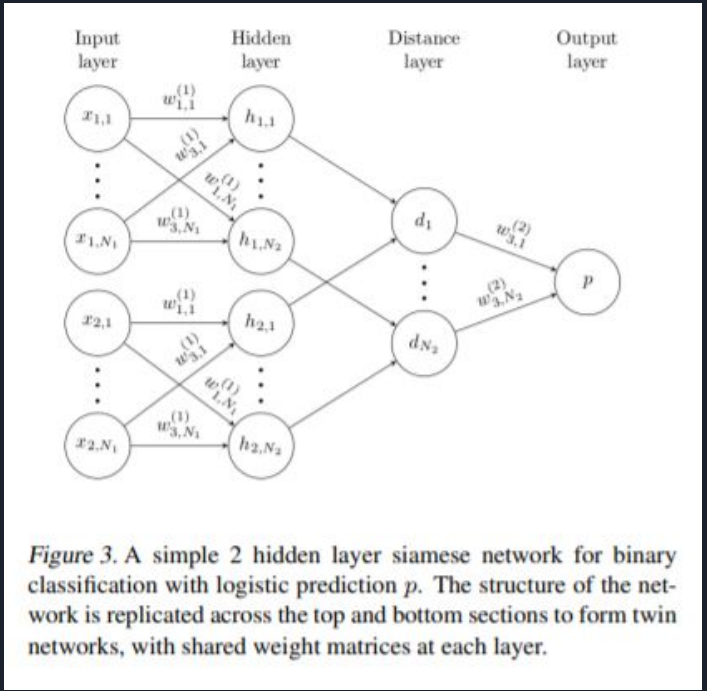


Figure 3. A simple 2 hidden layer siamese network for binary classification with logistic prediction p. The structure of the network is replicated across the top and bottom sections to form twin networks, with shared weight matrices at each layer.

Paper Overview : Methodology

- The model is composed of a sequence of convolutional layers, and each layer uses a single channel with varying filter sizes and a fixed stride of one.
- The units in the final convolutional layer are flattened into a single vector. This convolutional layer is followed by a fully-connected layer, and then one more layer computing the induced distance metric between each siamese twin, which is given to a single sigmoidal output unit.

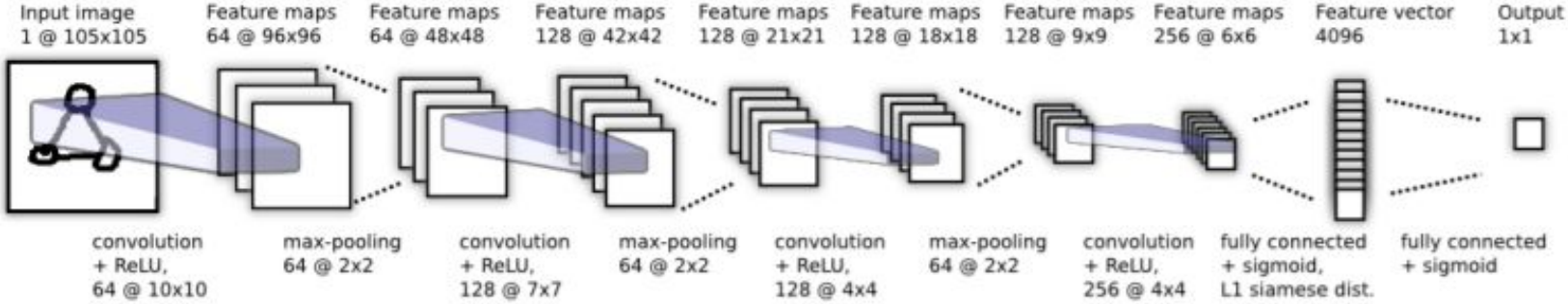


Figure 4. Best convolutional architecture selected for verification task. Siamese twin is not depicted, but joins immediately after the 4096 unit fully-connected layer where the L1 component-wise distance between vectors is computed.

```
# Convolutional Neural Network
model = Sequential()
model.add(Conv2D(64, (10,10), activation='relu', input_shape=input_shape,
                 kernel_initializer=initialize_weights, kernel_regularizer=l2(2e-4)))
model.add(MaxPooling2D())
model.add(Conv2D(128, (7,7), activation='relu',
                  kernel_initializer=initialize_weights,
                  bias_initializer=initialize_bias, kernel_regularizer=l2(2e-4)))
model.add(MaxPooling2D())
model.add(Conv2D(128, (4,4), activation='relu', kernel_initializer=initialize_weights,
                  bias_initializer=initialize_bias, kernel_regularizer=l2(2e-4)))
model.add(MaxPooling2D())
model.add(Conv2D(256, (4,4), activation='relu', kernel_initializer=initialize_weights,
                  bias_initializer=initialize_bias, kernel_regularizer=l2(2e-4)))
model.add(Flatten())
model.add(Dense(4096, activation='sigmoid',
                kernel_regularizer=l2(1e-3),
                kernel_initializer=initialize_weights,bias_initializer=initialize_bias))
```

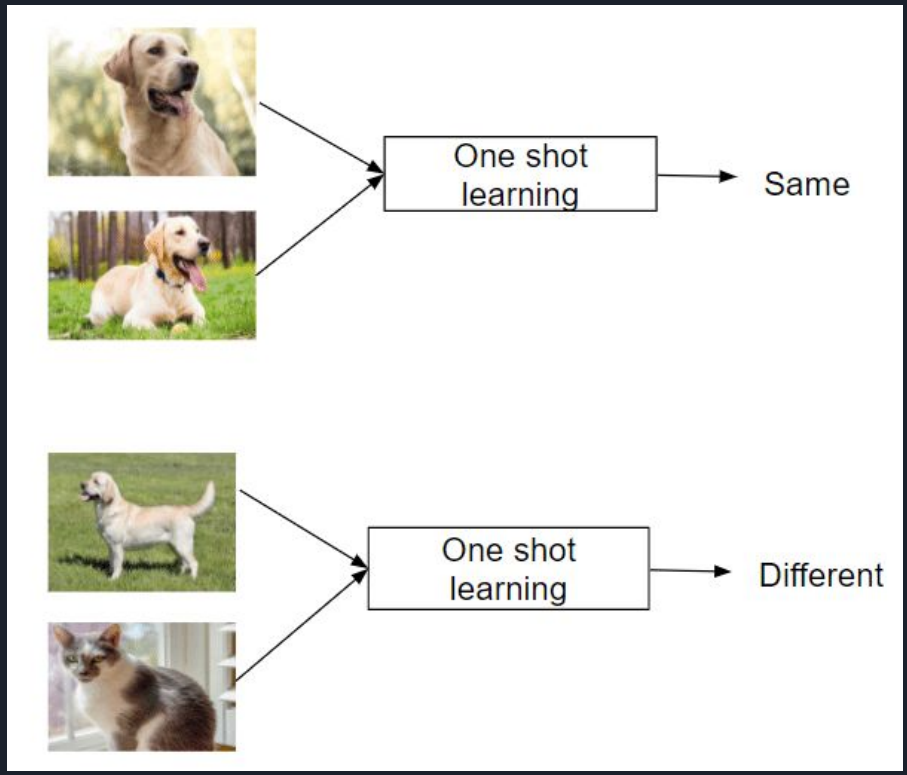
Paper Overview : Outcome

- The proposed architecture managed to accomplish the one-shot classification task of being able to identify classes using just one training example.
- The model outperforms all available baselines by a significant margin and also achieves and accuracy very close to human-level (95.5%).

Background Concepts

Siamese Neural Network: A Siamese neural network consists of two identical subnetworks, a.k.a. twin networks, joined at their outputs. Not only the twin networks have identical architecture, but they also share weights.

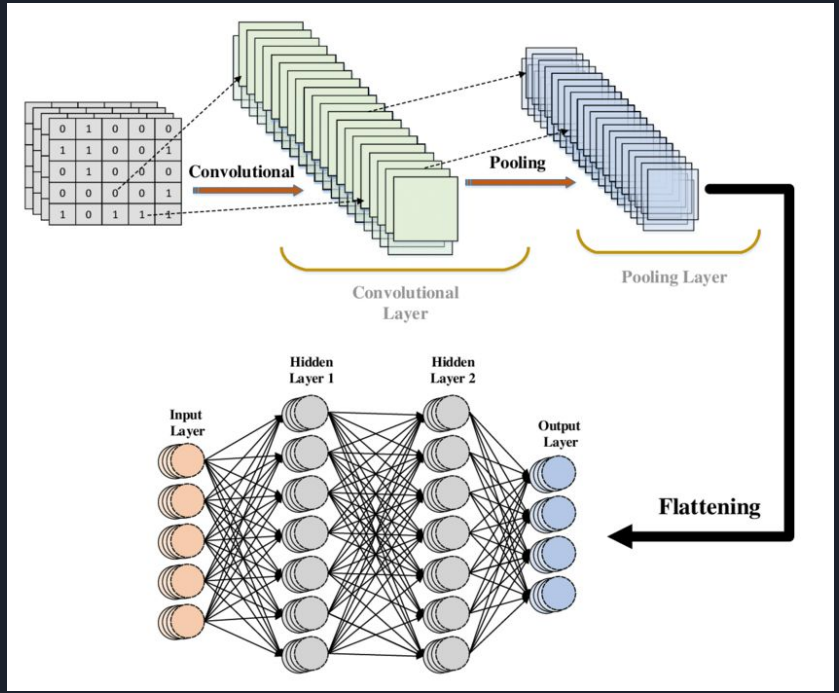
One-shot Learning: It is a classification task which involves predicting the label for an example after providing just one training example for a specific class.



Background Concepts

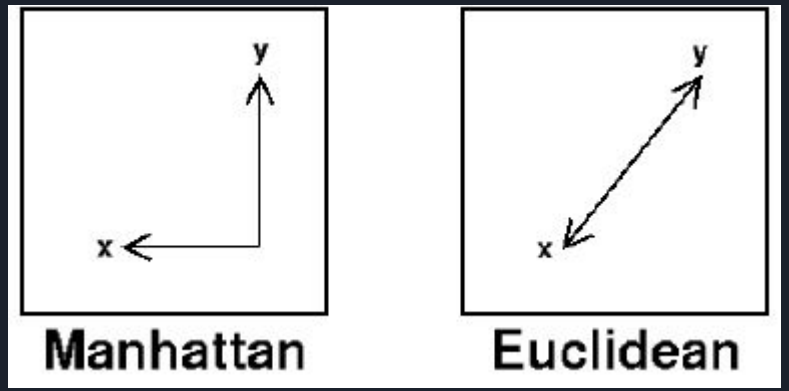
Convolutional Neural Network (CNN): A convolutional neural network, or CNN, is a deep learning neural network designed for processing structured arrays of data such as images.

Max-pooling: Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter.



Background Concepts

L1 Distance (Manhattan Distance): Also known as Manhattan Distance or Taxicab norm. L1 Norm is the sum of the magnitudes of the vectors in a space. It is the sum of absolute difference of the components of the vectors.



Dataset Details : Omniglot

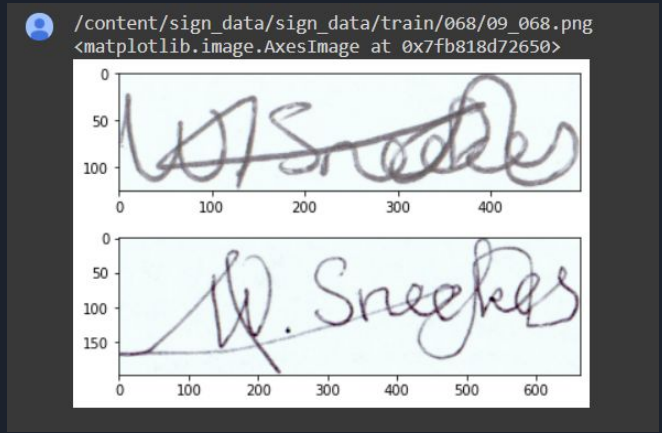
- The Omniglot data set was collected by Brenden Lake and his collaborators at MIT via Amazon's Mechanical Turk to produce a standard benchmark for learning from few examples in the handwritten character recognition domain .
- It contains examples from 50 alphabets ranging from well-established international languages like Greek and Sanskrit to lesser known local dialects.

Dataset Details : Omniglot

- The number of letters in each alphabet varies considerably from about 15 to upwards of 40 characters.
- The background set is used for developing a model by learning hyperparameters and feature mappings while the evaluation set is used only to measure the one-shot classification performance.

Dataset Details : Signature Verification

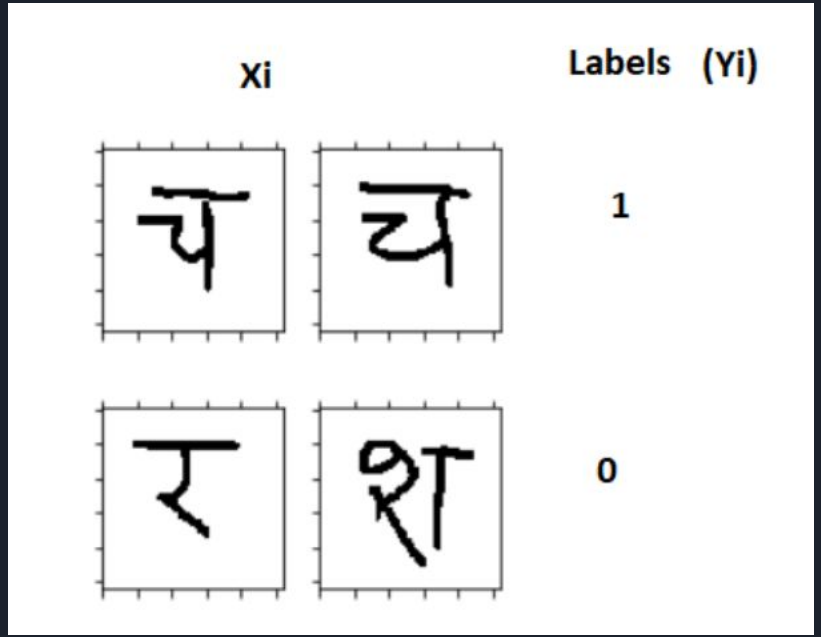
We tried the same architecture on the Signature Verification Dataset from Kaggle used to check Dutch signatures for forgery, and managed to reach an accuracy of 87.85% on running it for 2 epochs. However, the model took a lot of time to run with each epoch taking up over 30 minutes.



```
Epoch 1/3
508/508 [==============================] - ETA: 0s - loss: 1.0621 - acc: 0.6836
Epoch 00001: val_loss improved from inf to 0.50675, saving model to best_model.h5
508/508 [==============================] - 3452s 7s/step - loss: 1.0621 - acc: 0.6836 - val_loss: 0.5067 - val_acc: 0.8292
Epoch 2/3
508/508 [==============================] - ETA: 0s - loss: 0.4461 - acc: 0.8595
Epoch 00002: val_loss improved from 0.50675 to 0.40695, saving model to best_model.h5
508/508 [==============================] - 3498s 7s/step - loss: 0.4461 - acc: 0.8595 - val_loss: 0.4069 - val_acc: 0.8785
```

Implementation Details : Data Pre-processing

1. Firstly, we loaded the images by first looping over all the languages and fetching all the images for each alphabet in every language one by one. This was done for both test and validation sets.
2. Then we randomly selected languages for each batch and in every language, formed pairs of images with target of 1 if they belong to the same alphabet and 0 if they are of different alphabets.



Implementation Details : Architecture and Training

- The two input images (x1 and x2) are passed through a ConvNet to generate a fixed length feature vector for each (h(x1) and h(x2)).
- The L1 distance between the feature vectors is then used to compute a similarity score which is then optimised by training.
- We used Adam Optimizer with a learning rate of 0.00009 exponentially decaying at a rate of 0.99 and binary cross-entropy loss.
- The model was trained for 1000 epochs with a batch size of 256.

```
lr_schedule = schedules.ExponentialDecay(
    initial_learning_rate=0.00009,
    decay_steps=10000,
    decay_rate=0.99)
optimizer = Adam(learning_rate = lr_schedule)
model.compile(loss="binary_crossentropy",optimizer=optimizer)
```

```
[ ]  # Hyper parameters
     evaluate_every = 200
     batch_size = 256
     n_iter = 1000
     N_way = 20
     n_val = 250
     best = -1
```

Implementation Details: Validation and Testing

- N-way One-shot Learning has been used to validate and test the data.
- For each alphabet, a test image and a support set is generated randomly from the validation set. The test image is then compared to each of the images in the support set and the pair with the maximum similarity is used to determine the accuracy.
- If the image with the maximum similarity is the correct one, we consider it as correct and increase the number of correctly classified examples.

```python
def test_oneshot(model, N, k, s = "val", verbose = 0):
    n_correct = 0
    if verbose:
        print("Evaluating model on {} random {} way one-shot learning tasks ... \n".format(k,N))
    for i in range(k):
        inputs, targets = make_oneshot_task(N,s)
        loss = model.train_on_batch(inputs, targets)
        probs = model.predict(inputs)
        if np.argmax(probs) == np.argmax(targets):
            n_correct+=1
    percent_correct = (100.0 * n_correct / k)
    if verbose:
        print("Val Loss: {}".format(loss))
        print("Got an average of {}% {} way one-shot learning accuracy \n".format(percent_correct,N))
    return percent_correct
```

This character is compared to nine different characters

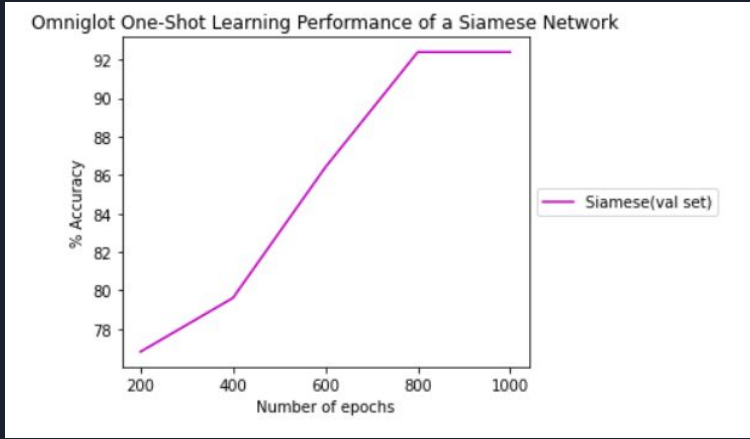
These are the 9 characters which are compared with the character on the left

These 9 comparisons will result in 9 different similarity scores.

If the similarity score for this comparison is maximum, then the prediction is correct otherwise incorrect

This set of 9 characters is also called as the "Support Set"

Results and Discussion



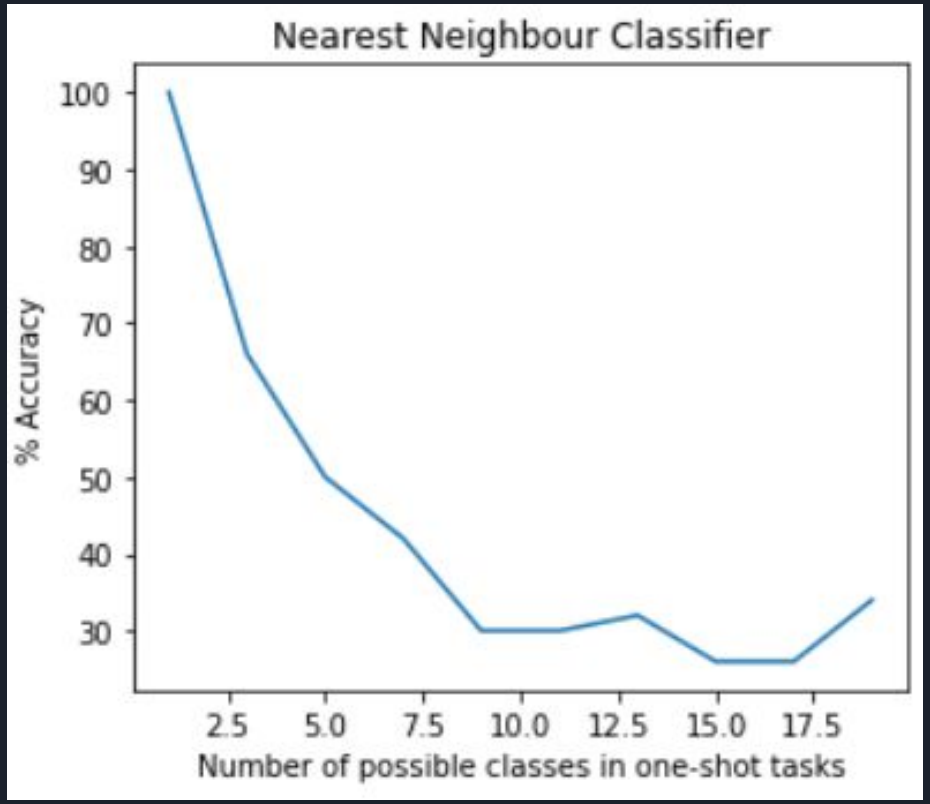

```
Time for 1000 iterations: 16.939632189273834 mins
Train Loss: 0.2726103961467743
Evaluating model on 250 random 20 way one-shot learning tasks ...

Val Loss: 0.21861477196216583
Got an average of 92.4% 20 way one-shot learning accuracy

Current best: 92.4, previous best: 92.4
```

Results and Discussion

Initially, the accuracy was not that high (~80%). However, after multiple changes we were able to achieve an accuracy of 92.4%. These were:

- Changed the learning rate from 0.00006 to 0.00009 and added a decay rate of 0.99.
- Changed the activation function from ReLU to Scaled Exponential Linear Unit (SELU).

We also plotted the accuracy for a Nearest Neighbour classifier using L2 distance on the same dataset and compared it with our model.


Nearest Neighbour Classifier
% Accuracy
100
90
80
70
60
50
40
30
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Number of possible classes in one-shot tasks

Challenges Faced

- We initially found the dataset a bit hard to manipulate as it was quite large. However, with the help of a few articles online we were able to write the functions to extract the dataset and split the data into pairs.
- The complex architecture of Siamese Networks was slightly hard to comprehend and we required other additional resources to completely understand the concept.
- None of us had prior Deep Learning experience and had to learn Tensorflow/Keras from scratch.

Future Scope

- Instead of using binary cross-entropy, alternate loss functions like triplet and contrastive loss can be used which may give better results.
- The one-shot classification model can be extended to 3D RGB images and used for applications like face recognition and verification.
- Various data augmentation techniques like changing contrast, adding noise and random choosing can be tried.
- Many of the hyper-parameters can be further tuned or a software like Whetlab can be used for further optimization.

Experience/Learning Outcome

- We learnt to use Tensorflow/Keras and applied Deep Learning techniques like CNN.
- Learnt to use Python libraries like Pickle, Matplotlib, etc.
- Understood hyperparameter optimization and tuned various hyperparameters to increase the accuracy.
- Read about various new activation functions like SeLU and GeLU which can be used as substitutes for ReLU.
- We also learnt how to read research papers end-to-end and replicate the results by ourselves.

THANK YOU