# Siamese Neural Networks for One-shot Image Recognition

Shay Kricheli

Advanced Seminar in Machine Learning
Department of Computer Science, Ben Gurion University

May 2021

## Seminar Presentation Overview

- The article we'll discuss was written by Gregory Koch, Richard Zemel and Ruslan Salakhutdinov and was published in 2015 (Gregory Koch & Salakhutdinov, 2015)
- It discusses the task of performing image recognition in cases where little data is available
- This is suggested by a technique called ***one-shot learning***
- The technique uses a ***Siamese*** neural network which is a special architecture of neural networks

## Preliminaries

- Image Recognition
- Convolutional Neural Network
- Siamese Neural Network
- One-Shot Learning

## Image Recognition - Overview

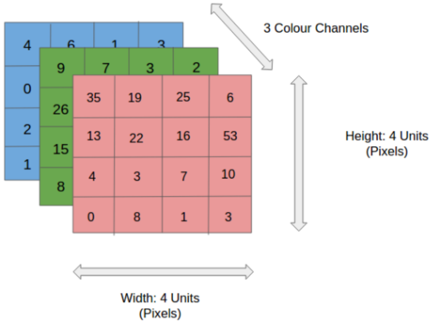
- Image recognition is a subcategory of Computer Vision and Artificial Intelligence
- It represents a set of methods for detecting and analyzing images to enable the automation of a specific task
- It is a technology that is capable of identifying places, people, objects and many other types of elements within an image, and drawing conclusions from them by analyzing them (Deepomatic, 2007)

## Convolutional Neural Network (CNN)

- A specialized type of artificial neural network that roughly mimics the human vision system, most commonly applied to analyze visual imagery
- A CNN is able to successfully capture the spatial dependencies in an image through the application of relevant filters (Sumit Saha, 2018)
- Convolutional networks use convolution in place of general matrix multiplication in at least one of their layers
- In recent years, CNNs have become pivotal to many computer vision applications (TechTalks, 2020)

## CNN Motivation - I

- Let us consider a small $4 \times 4$ pixels image



- Each pixel is represented by an integer in the range $[0, 255]$
- Let us naively try to evaluate how many parameters are needed to define it

## CNN Motivation - II

- This image is defined by $4 \cdot 4 \cdot 3 = 48$ parameters
- A high resolution image like $8K$ will have $7680 \times 4320 \times 3 = 99,532,800$ parameters
- So a CNN reduces the images into a form which is easier to process, without losing features which are critical for getting a good prediction

## CNN History

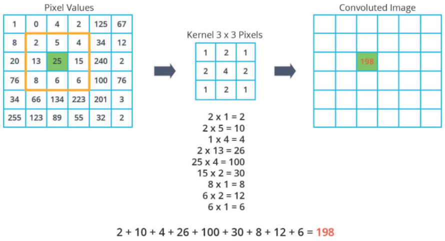
- Were first introduced in the 1980s by Yann LeCun, a French postdoctoral computer science researcher
- The early version of CNNs, called LeNet (after LeCun), could recognize handwritten digits
- ConvNets remained on the sidelines because at the time, the technique was only applicable to images with low resolutions
- In 2012, AlexNet (named after its main creator, Alex Krizhevsky) showed that perhaps the time had come to revisit deep learning: it won the 2012 ImageNet computer vision contest with an amazing 85 percent accuracy, while the runner-up scored a modest 74 percent on the test

## CNN Architecture

- Convolutional neural networks are composed of multiple layers of artificial neurons
- When you input an image into a ConvNet, each of its layers generates several activation maps
- Activation maps highlight the relevant features of the image

## Convolution

- A convolution is carried out by an appropriate weight matrix that is called a ***filter*** or ***kernel***
- A filter takes a patch of pixels as input, multiplies their color values by its weights and sums them up



|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 1 | 0 | 4 | 2 | 125 | 67 |
| 8 | 2 | 5 | 4 | 34 | 12 |
| 20 | 13 | 25 | 15 | 240 | 2 |
| 76 | 8 | 6 | 6 | 100 | 76 |
| 34 | 66 | 134 | 223 | 201 | 3 |
| 255 | 123 | 89 | 55 | 32 | 2 |

|  |  |  |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

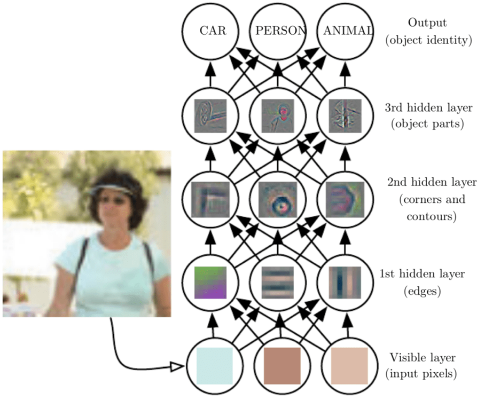
2 x 1 = 2
2 x 5 = 10
1 x 4 = 4
2 x 13 = 26
25 x 4 = 100
15 x 2 = 30
8 x 1 = 8
6 x 2 = 12
6 x 1 = 6

2 + 10 + 4 + 26 + 100 + 30 + 8 + 12 + 6 = 198

- This operation of multiplying pixel values by weights and summing them is called ***convolution***

## Convolution Layers

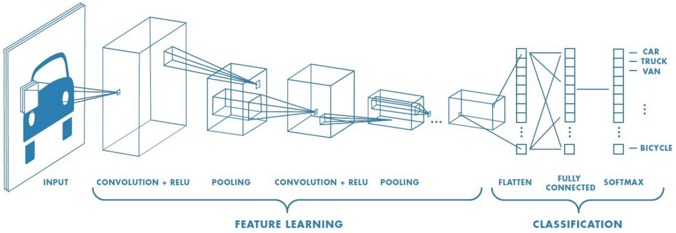
- The first (or bottom) layer usually detects basic features such as horizontal, vertical, and diagonal edges
- The next layer extracts more complex features, such as corners and edges
- As you move deeper the layers start detecting higher-level features such as objects, faces and more



CAR
PERSON
ANIMAL
Output (object identity)
3rd hidden layer (object parts)
2nd hidden layer (corners and contours)
1st hidden layer (edges)
Visible layer (input pixels)

## Complete CNN Architecture

- A CNN is usually composed of several convolution layers, but it also contains other components - like batch normalization and pooling
- In between each layer a non-linear activation function occurs, usually ReLu in the hidden layers and Softmax in the output layer
- Pooling layers (max or mean pooling) are responsible for reducing the spatial size of the convolved features
- At the end, the output is flattened and a fully connected layer outputs the chosen class

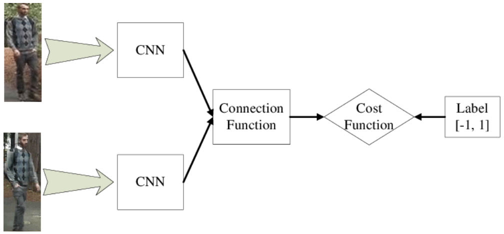


## Siamese Neural Network (SNN)

- Siamese nets were first introduced in the early 1990's by Bromley and LeCun
- Were used to solve signature verification
- The problem was modeled as an image matching problem
- The model was replicated across the top and bottom sections to form twin networks (Bromley et al., 1993)

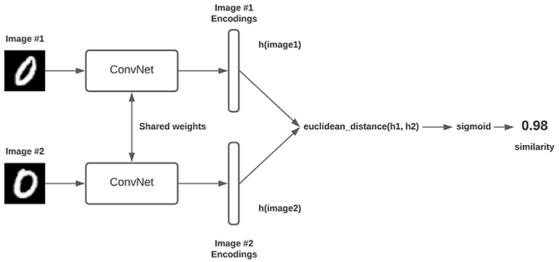
## SNN Architecture

- A SNN is a neural network that uses the same weights while working in conjunction on two different input vectors
- These vectors are used to compute two comparable output vectors
- These two output vectors are then joined by some metric to determine something of interest



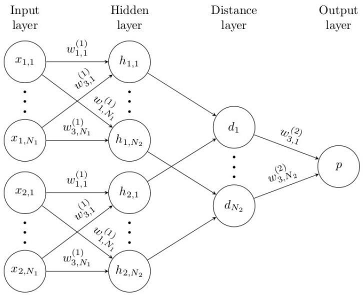
## SNN Use Cases

- Recognizing handwritten checks
- Comparing signatures
- Matching queries with indexed documents (Yi, Lei, & Li, 2014)
- Face recognition between two images
- And as was done in the article - digit matching



## Basic SNN Example

- 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
- The weight matrices at each layer are shared between the twins



## One-Shot Learning

- Most machine learning algorithms require training on hundreds or thousands of examples, and possibly more
- One-Shot Learning aims to learn information about object categories from one, or only a few, training examples
- It is estimated that an average child learns about $10 \sim 30$ thousand object categories in the world by the age of six
- Learning systems, like humans, can use prior knowledge about object categories to classify new objects

## One-Shot Learning Seminal Work

- The seminal work towards one-shot learning dates back to the early 2000's (Fei-Fei, Fergus, & Perona, 2006)
- The authors developed a variational Bayesian framework for one-shot image classification
- More recently one-shot learning for character recognition was done with a method called Hierarchical Bayesian Program Learning (Lake, Salakhutdinov, & Tenenbaum, 2013)
- The goal of HBPL is to determine a structural explanation for the observed pixels
- However, inference under HBPL is difficult since the joint parameter space is very large, leading to an intractable integration problem

## One-Shot Learning Seminal Work

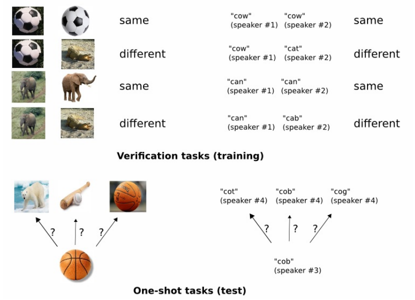
- Some researchers have considered other modalities or transfer learning approaches
- A more recent work uses a generative Hierarchical Hidden Markov model for speech primitives combined with a Bayesian inference procedure to recognize new words by unknown speakers (Lake, ying Lee, Glass, & Tenenbaum, 2014)

## The Article Approach - I

- In the article, they learn image representations via a supervised metric-based approach
- They first pre-train a Siamese neural network on some sufficiently large dataset
- Then they reuse that network's features for one-shot learning without any retraining on the MNIST dataset

# The Article Approach - II

- The main hypothesis the article lays out is that networks which do well at at verification should generalize to one-shot classification



## The Article Approach - III

- First they learn a neural network that can discriminate between the class-identity of image pairs
- This is the standard ***verification*** task for image recognition
- The verification model learns to identify input pairs according to the probability that they belong to the same class or different classes
- This model can then be used to evaluate new images, exactly one per novel class, in a pairwise manner against the test image
- The pairing with the highest score according to the verification network is then awarded the highest probability for the one-shot task

## The Article Approach - IV

- They restrict their attention to character recognition
- They although mention the basic approach can be replicated for almost any modality
- They also hypothesize that if the features learned by the verification model are sufficient to confirm or deny the identity of characters from one set of alphabets, then they ought to be sufficient for other alphabets
- This is, provided that the model has been exposed to a variety of alphabets to encourage variance amongst the learned features

## The General Model - I

- A Siamese convolutional neural network with $L$ layers
- Each layer has $N_l$ units $(1 \leq l \leq L)$
- ReLU units in the first $L-2$ layers
- Sigmoid units in the remaining layers
- Filters of varying size and a fixed stride of 1
- The RELU in each layer is optionally followed by max-pooling with a filter size and stride of 2

## The General Model - II

- $\mathbf{h_{1,l}}$ represents the hidden vector in layer $l$ for the first twin
- $\mathbf{h_{2,l}}$ represents the same for the second twin
- $W_{l-1,l}$ represents the 3-D tensor representing the feature maps for layer $l$
- $\mathbf{b_l}$ represents the bias vector for layer $l$
- So the $k$'th filter map in each layer takes the following form:

$$a_{1,m}^{(k)} = max - pool(max(0, W_{l-1,l}^{(k)} * \mathbf{h_{1,l}} + \mathbf{b_l}), 2)$$
$$a_{2,m}^{(k)} = max - pool(max(0, W_{l-1,l}^{(k)} * \mathbf{h_{2,l}} + \mathbf{b_l}), 2)$$

## The General Model - III

- The units in the final layer are flattened into a single vector
- This convolutional layer is followed by a fully-connected layer
- Then one more layer computing the induced $L_1$ distance metric between each Siamese twin
- This outputs to a single sigmoidal output unit

## The General Model - IV

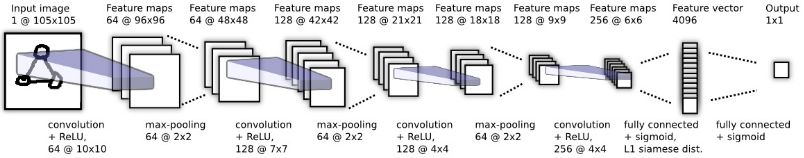
- Thus the final prediction vector is given as:

$$\mathbf{p} = \sigma\Big(\sum_{j=1}^{L} \alpha_j |h^j_{1,L-1} - h^j_{2,L-1}|\Big)$$

- $\sigma$ is the sigmoidal activation function
- $\alpha_j$ are additional parameters that are learned by the model during training, weighting the importance of the component-wise distance

## The General Model - V

- This example shows the largest version of the model considered in the article
- This network also gave the best result on the verification task



## Learning Architecture

- Let $M$ represent the minibatch size and let us assume $i$ indexes the $i$'th minibatch
- Let $\mathbf{x_k^{(i)}} = (x_{k,1}^{(i)}, x_{k,2}^{(i)}, ..., x_{k,M}^{(i)})$ be the feature vector for batch $i$ and twin $k \in \{1, 2\}$
- Let $\mathbf{y}(\mathbf{x_1^{(i)}}, \mathbf{x_2^{(i)}})$ be a length-$M$ vector which contains the labels for the minibatch
- $y_j(\mathbf{x_1^{(i)}}, \mathbf{x_2^{(i)}}) = 1$ whenever $x_{1,j}^{(i)}$ and $x_{2,j}^{(i)}$ are from the same character class and $y_j(\mathbf{x_1^{(i)}}, \mathbf{x_2^{(i)}}) = 0$ otherwise

## Loss Function

- The article uses a regularized cross-entropy objective on the binary classifier:

$$\begin{aligned}\mathcal{L}(\mathbf{x_1^{(i)}},\mathbf{x_2^{(i)}}) = \quad & \mathbf{y}(\mathbf{x_1^{(i)}},\mathbf{x_2^{(i)}}) \log \mathbf{y}(\mathbf{x_1^{(i)}},\mathbf{x_2^{(i)}}) \; + \\ & \left(1-\mathbf{y}(\mathbf{x_1^{(i)}},\mathbf{x_2^{(i)}})\right) \log \left(1-\mathbf{p}(\mathbf{x_1^{(i)}},\mathbf{x_2^{(i)}})\right) + \\ & \lambda^T|\mathbf{w}|\end{aligned}$$

- $\mathbf{p}$ is according to the final prediction vector
- $\lambda$ is the $L_2$ regularization vector for all the weights
- $\mathbf{w}$ is a representative vector of the weights for all the layers

## Optimization

- The loss objective is combined with standard backpropagation algorithm
- The gradient is additive across the twin networks due to the tied weights
- Let $\eta_j$ and $\mu_j$ be the gradient descent learning rate and momentum for the $j$'th neuron in some layer respectively
- So the update rule at epoch $T+1$ for the weight between the $j$'th neuron in some layer and the $k$'th neuron in the successive layer is:

$$\begin{aligned}\mathbf{w_{k,j}}^{(T+1)} &= \mathbf{w_{k,j}}^{(T)} + \Delta\mathbf{w_{k,j}}^{(T+1)} + 2\lambda_j|\mathbf{w}_{k,j}| \\ \Delta\mathbf{w_{k,j}}^{(T+1)} &= \mu_j\Delta\mathbf{w_{k,j}}^{(T)} - \eta_j\nabla\mathbf{w_{k,j}}^{(T)}\end{aligned}$$

- Where $\nabla\mathbf{w_{k,j}}^{(T)}$ is the partial derivative with respect to $\mathbf{w_{k,j}}^{(T)}$

## Weight initialization

- All network weights in the convolutional layers were initialized from a normal distribution with zero-mean and a standard deviation of $10^{-2}$
- Biases were also initialized from a normal distribution, but with mean $0.5$ and standard deviation $10^{-2}$
- In the fully-connected layers, the biases were initialized in the same way as the convolutional layers
- The weights in the fully-connected layers were drawn from a much wider normal distribution with zero-mean and standard deviation $2 \cdot 10^{-1}$

## Learning Schedule

- Although there are different learning rates for each layer, they were decayed uniformly across the network by 1 percent per epoch, so that roughly $\eta_j^{T+1} = 0.99\eta_j^T$ for any $j$'th layer
- They fixed the momentum to start at $0.5$ in each layer, increasing linearly each epoch until reaching the value $\mu_j$, the individual momentum term for the $j$'th layer
- Each network was trained for a maximum of 200 epochs

## Validation

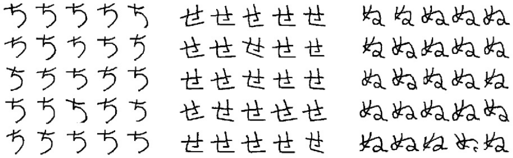
- They monitored one-shot validation error on a set of 320 one-shot learning tasks generated randomly from the alphabets
- When the validation error did not decrease for 20 epochs, they stopped and used the parameters of the model at the best epoch according to the one-shot validation error
- If the validation error continued to decrease for the entire learning schedule, they saved the final state of the model generated by this procedure

## Hyperparameter Optimization

- The article employs a Bayesian optimization framework called Whetlab to perform hyperparameter selection
- For this task they used $\eta_j \in [10^{-4}, 10^{-1}], \mu_j \in [0, 1]$ and $\lambda_j \in [0, 0.1]$
- They set the optimizer to maximize one-shot validation set accuracy
- The score assigned to a single Whetlab iteration was the highest value of this metric found during any epoch

## Affine Distortions

- In addition, they augmented the training set with small affine distortions



- For each image pair $x_1, x_2$, they generated a pair of affine transformations $T_1, T_2$ to yield $x_1' = T_1(x_1)$ and $x_2' = T_2(x_2)$
- $T_1, T_2$ are determined stochastically by a multidimensional uniform distribution
- For an arbitrary transform $T$, we have $T = (\theta, \rho_x, \rho_y, s_x, s_y, t_x, t_y)$ with $\theta \in [-10, 10]$, $\rho_x, \rho_y \in [-0.3, 0.3]$, $s_x, s_y \in [0.8, 1.2]$ and $t_x, t_y \in [-2, 2]$
- Each of these components of the transformation is included with probability $0.5$

## Model Training - I

- They first trained the model on a subset of the Omniglot data set, which contains a variety of different letters from alphabets across the world

Aurek-Besh
Futurama
Greek
Hebrew
Korean
Latin
Malay
Sanskrit

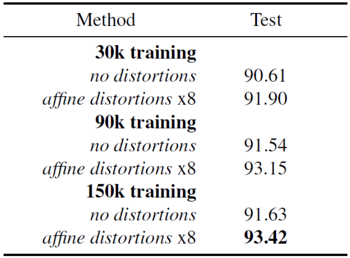
- To train the verification network, they put together three different data set sizes with 30,000, 90,000, and 150,000 training examples by sampling random *same* and *different* pairs
- They set aside sixty percent of the total data for training: 30 alphabets out of 50

## Model Training - II

- They fixed a uniform number of training examples per alphabet so that each alphabet receives equal representation during optimization
- By adding affine distortions, they also produced an additional copy of the data set corresponding to the augmented version of each of these sizes
- They added eight transforms for each training example, so the corresponding data sets have 270,000, 810,000, and 1,350,000 effective examples

## Verification Results

- The following table below lists the final verification results for each of the six possible training sets, where the listed test accuracy is reported at the best validation checkpoint and threshold



| Method | Test |
|---|---|
| **30k training** | |
| *no distortions* | 90.61 |
| *affine distortions* x8 | 91.90 |
| **90k training** | |
| *no distortions* | 91.54 |
| *affine distortions* x8 | 93.15 |
| **150k training** | |
| *no distortions* | 91.63 |
| *affine distortions* x8 | **93.42** |

- Once they have optimized a Siamese network to master the verification task, they are ready to demonstrate the discriminative potential of the learned features at one-shot learning

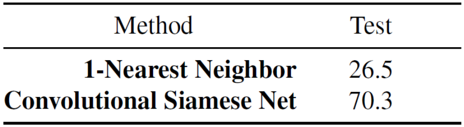
## MNIST One-Shot Learning

- Let $\mathbf{x}$ be a test image, some column vector which we wish to classify into one of $C$ categories
- We are also given some other images $\{\mathbf{x}_c\}_{c=1}^{C}$, a set of column vectors representing examples of each of those $C$ categories
- We can now query the network using $\mathbf{x}, \mathbf{x}_c$ as our input for a range of $1 \leq c \leq C$
- The predicted class corresponds to the maximum similarity:

$$C^* = \underset{c}{\operatorname{argmax}}\, \mathbf{p}^{(c)}$$

## Final Results

- The following table shows the results from the experiment



| Method | Test |
|---|---|
| **1-Nearest Neighbor** | 26.5 |
| **Convolutional Siamese Net** | 70.3 |

- We can see the major difference between the performance of the SNN and the neighbor baseline
- We can now query the network using $\mathbf{x}, \mathbf{x}_c$ as our input for a range of $1 \leq c \leq C$
- 70 % is a tremendous generalization result from the features learned on Ominglot and then doing one-shot on MNIST

## Conclusions

- The following table shows the results from the experiment
- The article presented a strategy for performing one-shot classification by first learning deep convolutional Siamese neural networks for verification
- The described network outperform all available baselines by a significant margin and come close to the best numbers achieved by the previous authors
- The authors argue that the strong performance of these networks on this task indicate not only that human-level accuracy is possible with the described metric learning approach, but that this approach should extend to one-shot learning tasks in other domains, especially for image classification

## References

Bromley, J., Bentz, J., Bottou, L., Guyon, I., Lecun, Y., Moore, C., . . . Shah, R. (1993). Signature verification using a siamese time delay neural network.

Deepomatic, V. a. p. (2007). What is image recognition?

Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Gregory Koch, R. Z., & Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition.

Lake, B. M., Salakhutdinov, R. R., & Tenenbaum, J. (2013). One-shot learning by inverting a compositional causal process. , *26*.

Lake, B. M., ying Lee, C., Glass, J. R., & Tenenbaum, J. (2014). One-shot learning of generative speech concepts.

Sumit Saha, T. D. S. (2018). A comprehensive guide to convolutional neural networks — the eli5 way.

TechTalks. (2020). What are convolutional neural networks (cnn)?

Yi, D., Lei, Z., & Li, S. (2014, 07). Deep metric learning for practical person re-identification. *Proceedings - International Conference on Pattern Recognition*.