ECE 7650 Deep Learning with CNNs Group Project

Snap-Shot DVD Finder

Project Report

Group Members

Enze Cui – 7742045

Juwairiah Zia – 7895845

Instructor

Prof. Sherif Sherif, Ph.D., P.Eng

Table of Contents

Abstract

This report describes a neural network algorithm for a snap-shot app to detect DVD covers and classify them. The working of an algorithm is based on Siamese Neural Network (SNN) architecture. Siamese Neural Network is based on two identical standard convolution neural networks, with their output linked to a loss function. It makes use of image pairs rather than taking one input image in the form of large data volume and training a neural network on it as it occurs in the classification problem. The idea behind using Siamese Neural Network is to extract feature vector as an output from the neural network for the input DVD covers image pair and labeled as similar or dissimilar. We evaluated the similarity and dissimilarity for detecting the DVD covers using binary cross-entropy loss function and contrastive loss function. Give a single training data instance, a DVD cover image of a movie, Siamese Neural Network is the best choice in this case.

Introduction

Researchers and scientists have a history of mathematically comparing two lists of elements with the same data types and have designed and developed different coefficients and similarity rates for different contexts. In 1993, this concept was introduced as an artificial neural network algorithm for fingerprint recognition and estimated the output as the probability that the two fingerprint images belong to the same finger [1]. Then the same algorithm was implemented in 1994 for machine learning by introducing Siamese neural networks. The computational system compared two samples of a handwritten signature and detect if they have made by the same person. In addition, it learned to distinguish the correct and false signature, which proved to be important for designing and developing a signature forgery detector [2]. An android mobile application was developed using the same Siamese neural network to detect forgeries. The camera of the android cell phone takes an image of a signature, and then the Siamese neural network algorithm in the app analyzes the input image by comparing it with the other signatures used during training [3]. Since then, the Siamese neural network has been employed in various fields to detect and classify problems and has implemented several neural network architectures [4].

In reference to the android app for signature image analysis, our experiment was to develop an algorithm for a photo app where you take a picture of a DVD, and the app tells you all the information about it. This problem is related to object detection, object localization, and object recognition. We used the Stanford Mobile Visual Search Database as our training data, and testing data comprised of images, each containing one DVD in a non-frontal viewpoint [5]. Using it for our project, the goal is to generate a neural network well-trained on just one sample per class, and that can be used to predict movies by feed-in only one snapshot of the DVD cover. The output should be to find out the movie that the DVD cover belongs to.

The data set of our project is all given by Stanford mobile Visual Search Database, under the DVD_COVER folder, which contains five folders of different category images. There is one reference folder that is used as training data. It contains 100 DVD covers for 100 movies, which means each movie only has one image. Furthermore, the main folder contains four photo folders; each folder is picture taken by the device of the folder name. These are used as testing data.

Standard classification has always been the first approach for object detection, where input is fed into the neural network and the output class probabilities. Furthermore, this requires a large volume of data. A nice example is using classification to predict cats and dogs by training a model on cat and dogs pictures dataset [6]. But this model doesn't work on a dataset of few samples, and many times data is minimum. Siamese Neural Network is popular to mitigate the less training data problem. Siamese Neural Network learns to differentiate between two inputs instead of classifying them. It determines if the two inputs are different or similar, and the input can be anything. The advantages are less training data, less memory required, less computational cost, and time-consuming. Moreover, we don't need to worry about the number of classes.

A Siamese Neural Network consists of two identical neural networks with the exact same weights (Fig. 1). The parameters are shared between the two neural networks, meaning weights are updated in one, then the weights in the other neural network are updated. By using these two identical networks, we can compare and differentiate the two DVD cover images. Our architecture requires an input image pair. Therefore, we take a pair of DVD cover images, and each DVD cover image in the image pair is fed to one of these networks. The output of the left side of the neural network and the output of the right side of the neural network is compared through a loss function through distance metric, the binary and contrastive loss function in our model.

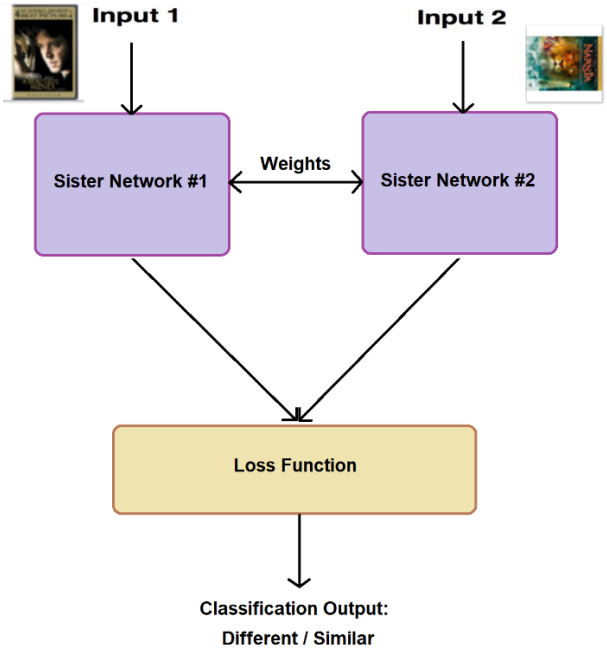

Fig. 1: Representation of the structure of Siamese Neural Network Model. Each input image is processed in the neural network, the value of the loss function is a measure of similarity between the input image pair, the final output is different (dissimilar) / similar

In the context of programming, Siamese neural network is implemented using one standard convolution neural network architecture. Because the weights are constrained to be identical for both networks, one neural network model is used, and two images are feed into it in succession. In our code, we took an input image of a DVD cover and found out the encodings of that image. Then, we take the same network without performing any updates on weights or biases and input an image of a different DVD cover and again predict its encodings. We then compare these two encodings to check whether there is a similarity between the two DVD cover images. These two encodings act as a latent feature representation of the images. Images with the same DVD cover have similar features and using this, we compared, estimated, and presented if the two images have the same DVD cover or not.

Here are the steps for training and testing of Siamese Neural Network

- Load training data
- Construct Siamese Neural Network
- Train the network by passing the first image of the image pair through the network
- Train the network by passing the second image of the image pair through the network
- Calculate the loss using the output from 1 and 2
- Back propagate the loss to calculate gradients
- Update the weights using optimizer – Adam

- Load the testing image from testing data
- Detect and localize target from the image
- Feed target to trained Siamese Neural Network
- Predict the target class with a similarity score according to the loss function

Methodology

The first hurdle we faced while preprocessing the testing data was that there were some bad photos in it, as depicted in Fig. 2. The assumption of the testing data for our project is that the image needs to be taken having a top-down viewing; the image must show the whole image of the DVD poster, not taken by side viewing or missing edges. Therefore, some bad images are excluded from testing data.

Fig. 2: Bad photos of DVD cover in the testing samples

Image Preprocessing consists of two parts – one is for the training dataset, that is reference, and the other is for testing data. The work we did for training data is mainly data expansion. Whereas for testing data, we have managed object detection and localization using some mathematical techniques and used rotation and resizing. The reason to have data expansion on training data is that we need more training data because the reference folder contains only one image for each class, presented in Fig. 3. Another issue is that photos may be taken from different angles, so we also introduced rotation when expanding the training dataset. We rotated the image in eight different angles, resized the rotated image to the same size, and generate copies of images for each movie class, as presented in Fig. 4.

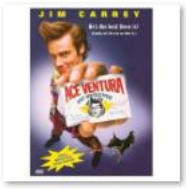

Fig. 3: Sample of training image in the reference folder

Fig 4. Rotation applied to the training image in the reference folder

Another problem we faced was finding the DVD poster in the photo; the image contains many objects, and it was taken from different angles. During the research phase, we have read a list of pre-trained object detection algorithms, and among them, we tried implementing YOLOv3, and it worked poorly with our dataset [8]. There is no DVD cover label in most of database, so object detection algorithms prefer to detect something other than the DVD cover, as depicted in Fig. 5, by detecting cast in the DVD cover

highlighted yellow bounding box. Besides this, the training data only has 100 images of the DVD poster, which is nearly impossible to retrain the detection model.

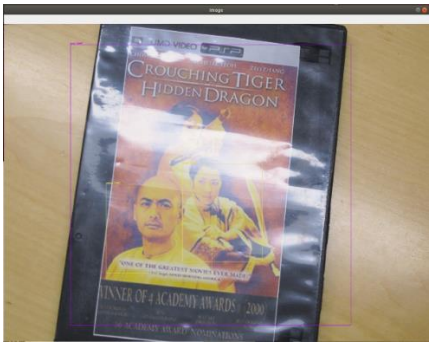

Fig. 5: The result of YOLOv3 on the dataset

Another method for DVD cover detection with a bounding box is to build a simple shape detection method from scratch as it can be easily modified with features of specific targets. For our project, we have a DVD cover, all DVD covers are rectangular in top-down viewing, and the target occupies the biggest area of the image. The way to build this shape detection method is shown in Fig. 6. First, it begins implementing a canny edge-detection function on a grayscale image, converted from an RGB image [7]. The canny edge-detection will calculate the image intensity gradient and extract edges in the image where it finds a large gradient. Then, it collects all closed contour shapes in binary edge image, resorts all contour by area from large to small, and finally finds the contour has a rectangular-like shape (Fig. 6).

Fig. 6: Canny edge-detection function for a DVD cover detection, left side is the grey-scaled image, next to it is edges extracted on it; the right side is contour as a rectangle shape

In the end, we found the image, but it is not in the shape we want it to be. Therefore, our next step was to use rotation and resize the image as the photo may be taken from a different angle, and also, the size of the object needs to be reconstructed to match the input size for our model. Using the 4-point transform method, it transfers the rectangle-like shape into a standard rectangle in a normal, top-down view image and resizes the image to correct size and orientation (Fig. 7) [8].

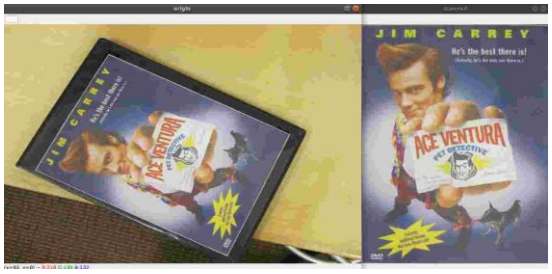

Fig. 7: 4-point transform method to resize and orientation of the image; left is the original testing image, and the right side is its top-down view

Binary Siamese Network

The implementation of binary cross-entropy loss was straightforward [9] and was performed by Enze Cui. As its output is binary, and so was the Siamese network. However, it has a problem that it does not respond level of confidence during the prediction; therefore, the model which implementing this loss function cannot rank the possible class for photos. We managed to evaluate the model by passing many testing images and calculating the accuracy at the end of each epoch. In the end, the model with an accuracy of 78.12% (Fig. 8). The loss function calculated the Euclidean difference of features extracted from two input DVD cover images and then used the sigmoid function to give the result as 0 or 1.

Fig. 8: Binary Testing Plot

Contrastive Siamese Network

We firstly tried binary cross-entropy loss, and we yield some good results. Nevertheless, we wanted to know how well the network evaluates and distinguishes between the two images; this was performed. This wasn't done efficiently with binary cross-entropy because the output of binary entropy loss was either one of two, 0, or 1. We then implemented the contrastive loss function, and it was fit to solve this problem. The network was trained for 50 epochs, using Adam and a learning rate of 0.0005 (Fig. 9). The Euclidean distance directly corresponds to the dissimilarity between the image pair output of the Siamese network. Through this measure of similarity of input DVD covers, the algorithm gives out the output by minimizing the distance between similar DVDs and maximizing the difference between dissimilar ones [6].

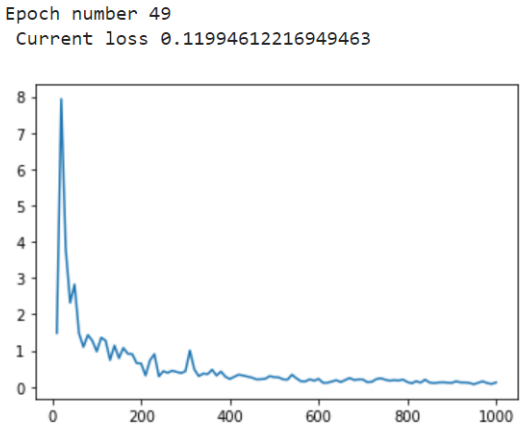

Fig. 9: Loss value over time of Contrastive Siamese Neural Network

Division of Labor

Tasks	**Subtasks**	**Individual in charge**
Proposal	Research and writing	Group
Preprocessing the Data	Analyze training and testing data	Group
	Research approach and algorithms	
	Implement and test	
Creating Siamese Network and Training It with Binary Loss Function	Research approach and algorithm	Enze Cui
	Coding	
	Implement and test	
Creating Siamese Network and Training It with Contrastive Loss Function	Research approach and algorithm	Juwairiah Zia
	Coding	
	Implement and test	
Report Writing	Writing and editing	Group

Results and Discussion

In our class project on deep learning with a convolution neural network, we implemented a Siamese Neural Network to detect and differentiate between pairs of DVD cover for a snap-shot app. We selected the Siamese network as it is useful where there is less training data, one training per class of a particular movie in our case. We trained the Siamese neural network from scratch with binary cross-entropy function and then with contrastive loss function, which we would recommend.

With binary entropy loss function, the output was either 1 that is the same class or 0, not the same class. In contrast, the testing result of the Siamese neural network with contrastive loss shows dissimilarity between the image pair, which corresponds to Euclidean distance (Fig. 10). The higher the value of distance, the higher the dissimilarity.

Fig. 10: On the left, the score is higher: the DVD covers are dissimilar; on the right side, the score is low: the DVD covers are similar

Another loss function, known as the triplet loss function, can also be implemented in this application. As mentioned before, this problem can also be solved using other neural network architectures such as ResNet, VGG13, VGG50, and GoogLeNet. This can be done with transfer learning by extracting feature vectors and then train any other machine learning model on top of these feature vectors to recognize the

DVD covers. Another method is fine-turning the neural network by removing its fully connected layer and replacing it with a new fully-connected layer, and making it learn to predict movies in DVD classes [10].

References

[1] Baldi P, Chauvin Y (1993) Neural networks for fingerprint recognition. Neural Comput 5 (3):402–418

[2] Bromley J, Guyon I, LeCun Y, et al (1994) Signature verification using a “siamese” time delay neural network. Adv Neural Inf Process Syst 6:737–744

[3] Grafilon P, Aguilar IB, Lavarias ED, et al (2017) A signature comparing android mobile application utilizing feature extracting algorithms. Int J Sci Technol Res 6(8):45–50

[4] Chicco, D. (2021). Siamese neural networks: An overview. Artificial Neural Networks, 73-94.

[5] Mina Makar, Sam Tsai, Vijay Chandrasekhar, David Chen and Bernd Girod, "Interframe Coding of Canonical Patches for Low Bit-Rate Mobile Augmented Reality," Special Issue of the International Journal of Semantic Computing, vol. 7, no. 1, pp. 5-24, March 2013. http://dx.doi.org/10.1142/S1793351X13400011. Data available at http://purl.stanford.edu/ph459zk5920.

[6] Gupta, H. (2017, July 8). One Shot Learning with Siamese Networks in PyTorch. Hacker Noon. https://hackernoon.com/one-shot-learning-with-siamese-networks-in-pytorch-8ddaab10340e

[7] Canny, John. (1986). A Computational Approach to Edge Detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on. PAMI-8. 679 - 698. 10.1109/TPAMI.1986.4767851

[8] Rosebrock, A. (2021, April 17). 4 Point OpenCV getPerspective Transform Example. PyImageSearch. https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/

[9] Rosebrock, A. (2021b, April 17). *Siamese networks with Keras, TensorFlow, and Deep Learning*. PyImageSearch. https://www.pyimagesearch.com/2020/11/30/siamese-networks-with-keras-tensorflow-and-deep-learning/

[10] Rosebrock, A. (2021c, April 17). Transfer Learning with Keras and Deep Learning. PyImageSearch. https://www.pyimagesearch.com/2019/05/20/transfer-learning-with-keras-and-deep-learning/