Last updated on Thursday, 29 August 2019

2019 IFN680 - Assignment Two
(Siamese network)

Assessment information

- Code and report submission due on **Sunday 27th October, 23.59pm**
- Use **Blackboard** to submit your work
- Recommended group size: three people per submission. Smaller group sizes allowed (1 or 2 people OK. Completion of the same tasks required).

Overview

- You will implement a deep neural network classifier to predict whether two images belong to the same class.
- The approach you are asked to follow is quite generic and can be applied to problems where we seek to determine whether two inputs belong to the same equivalence class.
- You will write code, perform experiments and report on your results.

Introduction

Despite impressive results in object classification, verification and recognition, most deep neural network based recognition systems become brittle when the view point of the camera changes dramatically. Robustness to geometric transformations is highly desirable for applications like wild life monitoring where there is no control on the pose of the objects of interest. The images of different objects viewed from various observation points define *equivalence classes* where by definition two images are said to be equivalent if they are views from the same object.

These equivalence classes can be learned via *embeddings* that map the input images to high dimensional vectors. During training, equivalent images are mapped to vectors that get pulled closer together, whereas if the images are not equivalent their associated vectors get pulled apart.

Background

Common machine learning tasks like classification and recognition involve learning an appearance model. These tasks can be interpreted and even reduced to the problem of learning manifolds from a training set. Useful appearance models create an invariant representation of the objects of interest under a range of conditions. A good representation should combine *invariance* and *discriminability*. For example, in facial recognition where the task is to compare two images and determine whether they show the same person, the output of the system should be invariant to the pose of the heads. More generally, the category of an object contained in an image should be invariant to viewpoint changes.

This assignment borrows ideas from a system developed for a manta ray recognition system. The motivation for our research work is the lack of fully automated identification systems for manta rays. The techniques developed for such systems can also potentially be applied to other marine species that bear a unique pattern on their surface. The task of recognizing manta rays is challenging because of the heterogeneity of photographic conditions and equipment used in acquiring manta ray photo ID images like those in the figures below.

Two images of the same Manta ray

Many of those pictures are submitted by recreational divers. For those pictures, the camera parameters are generally not known. The state of the art for manta ray recognition is a system that requires user input to manually align and normalize the 2D orientation of the ray within the image. Moreover, the user has to select a rectangular region of interest containing the spot pattern. The images have also to be of high quality. In practice, marine biologists still prefer to use their own decision tree that they run manually.

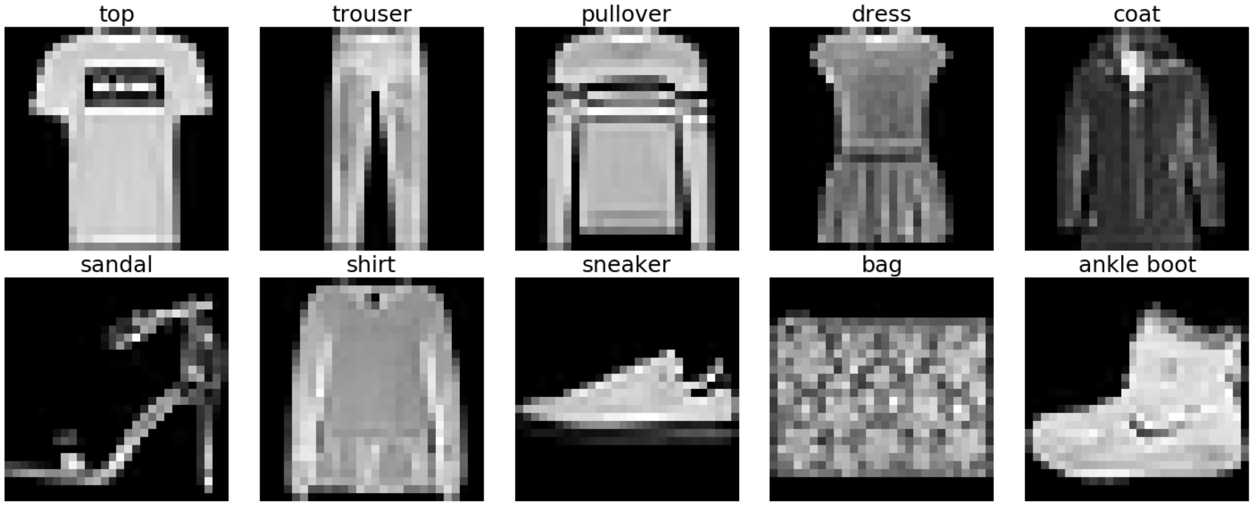
In order to develop robust algorithms for recognizing manta spot patterns, a research student (Olga, one of the tutors of this unit) and Dr. Frederic Maire have considered the problem of recognizing artificially generated patterns subjected to projective transformations that simulate changes in the camera view point. Artificial data allowed us to experiment with a large amount of patterns and compare different network architectures to select the most suitable for learning geometric equivalence. Our experiments have demonstrated that Siamese[1] convolutional neural networks are able to discriminate between patterns subjected to large homographic transformations. Promising results have been also obtained with real images of manta rays.

Training such a complex neural network requires access to good computing facilities. This is why in this assignment, you will work with a simpler dataset. Namely the Fashion-MNIST dataset. The dataset has small (28x28) grayscale images of clothing in ten categories: top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot.

You will build a classifier to predict whether two images correspond to the same clothing type or not. Note this is not the same problem as building a classifier! The idea is to build a model that does not memorize each class but learns a representation of each image?. Such a model is able to distinguish between classes and determine if two images belong to the same class or not.

---

[1]Siamese network are defined in the next section.



Images from Fashion-MNIST dataset

Learning equivalence classes

A ***Siamese network*** consists of two identical subnetworks that share the same weights followed by a distance calculation layer. The input of a Siamese network is a pair of images P_i and P_j. If the two images are deemed from the same equivalence classe, the pair is called a ***positive pair***, whereas for a pair of images from different equivalence classes, the pair is called *a* ***negative pair***.

The input images P_i and P_j are fed to the twin subnetworks to produce two vector representations $f(P_i)$ and $f(P_j)$ that are used to calculate a proxy distance. The training of a Siamese network is done with a collection of positive and negative pairs. Learning is performed by optimizing a ***contrastive loss*** function (see details later). The aim of the training algorithm is to minimize the distance between a pair of images from the same equivalence class while maximizing the distance between a pair of images from different equivalence classes.

With a conventional neural network classifier, we need to know in advance the number of classes. In practice, this is not always possible. Consider a face recognition system. If its use is limited to a fixed group of people of let say 10,000 people, you can train a convolutional neural network with 10,000 outputs where each output corresponds to a person. But this approach is not suitable when the group of people is more fluid. We want to be able to use the same network without having to retrain it when new people arrive. What is needed is a neural network that can take two input images and predict whether these are images of the same person. This neural network learns to compare pair of faces in general, and does not learn about specific people. This is a significant difference. You do not have to retrain the neural network if your population changes.

Siamese network architecture

A Siamese network consists of **two identical subnetworks** that **share the same weights** followed by a distance calculation layer. The input of a Siamese network is a pair of images (P_i, P_j) and a label y_{ij} . If the two images are deemed from the same equivalence classe, the pair is called a positive pair, and the target value is $y_{ij} = 0$. Whereas for a pair of images from different equivalence classes, the pair is called a negative pair, and the target value is $y_{ij} = 1$. The target value y_{ij} can be interpreted as the desired distance between the embedding vectors. The input images (P_i, P_j) are fed to the twin subnetworks to produce two vector representations $f(P_i)$, $f(P_j)$ that are used to calculate a proxy distance. The training of a Siamese network is done on a collection of positive and negative pairs.

Learning is performed by optimizing a contrastive loss function:

$$\mathcal{L}_c(P_i, P_j) = \begin{cases} \frac{1}{2} D(P_i, P_j)^2, & \text{if } y_{ij} = 0 \\ \frac{1}{2} \max(0, m - D(P_i, P_j))^2, & \text{if } y_{ij} = 1, \end{cases}$$

where $D(P_i, P_j)$ is the Euclidean distance between $f(P_i)$ and $f(P_j)$. The margin $m > 0$ determines how far the embeddings of a negative pair should be pushed apart.

Required tasks and experiments

- Load the Fashion-MNIST dataset (use keras.datasets.fashion_mnist.load_data).
- Split the dataset such that
  - the images with labels in ["top", "trouser", "pullover", "coat", "sandal", "ankle boot"] are used for training and testing
  - the images with labels in ["dress", "sneaker", "bag", "shirt"] are only used for testing. None of these images should be used during training.
- Implement and test the contrastive loss function described earlier in this document.
- Build a Siamese network.
- Train your Siamese network on your training set. Plot the training and validation error vs time.
- Evaluate the generalization capability of your network by
  - testing it with pairs from the set of images with labels ["top", "trouser", "pullover", "coat", "sandal", "ankle boot"]
  - testing it with pairs from the set of images with labels ["top", "trouser", "pullover", "coat", "sandal", "ankle boot"] union ["dress", "sneaker", "bag", "shirt"]
  - testing it with pairs from the set of images with labels in ["dress", "sneaker", "bag", "shirt"]
- Present your results in tables and figures.

Implementation hints

- You need the functional model of Keras to implement the Siamese network.
  Refer to https://keras.io/getting-started/functional-api-guide/ for examples.
- For the shared network of the Siamese network, you can adopt the CNN network architecture used in the solution Week 07 prac. Number of layers and filters can be increased to improve performance.
- There are plenty of examples of loss functions at
  https://github.com/keras-team/keras/blob/master/keras/losses.py
- Keep 80% of the images with labels ["top", "trouser", "pullover", "coat", "sandal", "ankle boot"] for training (and 20% for testing).
- Keep 100% of the images with labels in ["dress", "sneaker", "bag", "shirt"] for testing.

Submission

You should submit via Blackboard

- A **report in pdf format** strictly **limited to 8 pages in total** in which you present your experimental results using tables and figures. Only one person per group has to submit the assignment. Make sure you list the members of your group in the report and in the code. The feedback will be given to the person submitting the assignment and is expected to be shared with the other team members.
- Your Python file **my_submission.py** containing all your code and instructions on how to repeat your experiments.

Marking criteria

- **Report**: 10 marks
  - Structure (sections, page numbers), grammar, no typos.
  - Clarity of explanations.
  - Figures and tables (use for explanations and to report performance).

Levels of Achievement

10 Marks	7 Marks	5 Marks	3 Marks	1 Mark
Report written at the highest professional standard with respect to spelling, grammar, formatting, structure, and language terminology.	Report is very-well written and understandable throughout, with only a few insignificant presentation errors. Methodology, experiments are clearly presented.	The report is generally well-written and understandable but with a few small presentation errors that make one of two points unclear. Clear figures and tables.	Large parts of the report are poorly-written, making many parts difficult to understand. Use of sections with proper section titles. No figures or tables.	The entire report is poorly-written and/or incomplete and/or impossible to understand. The report is in pdf format.

To get "i Marks", the report needs to satisfy <u>all the positive items and none of the negative items</u> of the columns "j Marks" for all j<i. For example, if your report is not in pdf format, you will not be awarded more than 1 mark.

- **Code quality**: 10 marks
  - Readability, meaningful variable names.
  - Proper use of Python constructs like numpy arrays, dictionaries and list comprehension.
  - Header comments in classes and functions.
  - Function parameter documentation.
  - In-line comments.

Levels of Achievement

10 Marks	7 Marks	5 Marks	3 Marks	1 Mark
Code is generic. Minimal changes would be needed to run same experiments on a different dataset.	Proper use of numpy array operations. Avoid unnecessary loops. Useful in-line comments. Code structured so that it is straightforward to repeat the experiments	No magic numbers (that is, all numerical constants have been assigned to variables). Appropriate use of auxiliary functions. Each function parameter documented (including type and shape of parameters)	Header comments with instructions on how to run the code to repeat the experiments.	Code looks like a random spaghetti plate

To get "i Marks", the report needs to satisfy <u>*all the positive items and none of the negative items*</u> *of the columns "j Marks" for all j<i.*

- **Tasks and experiments** 20 marks
  - Creation and verification of the dataset : 4 marks
  - Implement and test the contrastive loss function : 4 marks.
  - Build a Siamese network : 4 marks
  - Successful training of the Siamese network : 4 marks
  - Evaluate the generalization capability of the Siamese network : 4 marks

Final Remarks

- Do not underestimate the workload. Start early. You are strongly encouraged to ask questions during the practical sessions.
- Email questions to yue.xu@qut.edu.au