UNIVERSITY OF BORDEAUX

INTERNSHIP REPORT

Deep Learning
Heritage Image Classification

Student:
Manh Tu Vu

Supervisor:
Marie Beurton-Aimar
Van Linh Le

January 9, 2018

Acknowledgements

First of all, I would like to express my deepest gratitude to my two supervisors, Mrs. Marie BEURTON-AIMAR and Mr. Van Linh LE for their agreements, guide, and support during the planning and developing of my internship.

I would like to thank Mr Fabien BALDACCI for his generous help and comment during my work. I would like to thank the staffs, students in LaBRI, who helped, supported with the technique and providing me a professional working environment.

I would also like to thank all the professors in the University of Bordeaux and the PUF-HCM, who imparted a lot of knowledge about learning and researching. Finally, I would like to thank my family and colleagues for their support and encouragement through my study.

Abstract

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

Image classification is a field that has many applications in life. It could be useful to categorize images, provide only images which the user are interesting, protect the children from unwanted contents such as violent or sexual.

Although, there are lots of classifiers that exists to solve this problem but, nowadays, Neural Network and Deep Learning provide one of the best solution to many problems in Image processing, and especially in Image classification.

The goal of my internship at LaBRI is research & implement the Deep learning in Image classification to classify images from Heritage repository[1]. Try to do in different approach, compare them and finally, choose the one which give the best result.

[1]https://heobs.org

Contents

List of Figures

Introduction

Heritage Observatory project has for aim to identify cultural and historical heritages, constitute a specific database, provide a data archive free for all. This platform allows us to receive the breaking news about cultural or historical heritage sites located in regions of the world that we are interesting. In order to do that, the platform will collect a huge of images about the cultural and historical heritage. The problem appears when we want to categorize those images into a specific classes when the image have no label and we can't looking to each of those images and categorize it by hand. We need to find a way to let the computer do it for us. The aim of my internship is to implement a deep-learning algorithm to classify those images.

Image classification is the task of assigning an input image one label from a fixed set of categories. There are lots of classifiers that exists, but nowadays, neural networks and deep learning currently provide the best solutions to many problems in image and speech recognition, and in natural language processing.

Deep learning, is a part of machine learning and it gives techniques for learning in neural networks. Neural networks are inspired by the human system of neurons that is able to learn from observations. It is feed by labeled data and learns automatically from that. The most adapted neural network for image recognition is the convolutional neural network (CNN). It's adapted to the recognition of 4 classes: being, heritage, scenery and other. That's why it has been implanted.

Deep learning has two major categories of image classification techniques include unsupervised and supervised classification. With unsupervised classification, all images are unlabeled and we using deep learning to learn to inherent structure from the image input data while with supervised classification, all images are labeled and we using deep learning to learn to predict the output from the image input data.
In this project, we'll implement both of those categories of image classification techniques and then compare the result to see which one are better to solve our problem.

Chapter 1

Context

1.1 Pôle Universitaire Français

The Pôle Universitaire Français (PUF) was created by the intergovernmental agreement of VietNam and France in October 2004. With ambition is building a linking program between the universities in VietNam and the advanced programs of universities in France. There are two PUF's center in VietNam: Pôle Universitaire Français de l'Universite Nationalé du Vietnam - Ha Noi located in Ha Noi capital (PUF-Ha Noi) and Pôle Universitaire Français de l'Universite Nationalé du Vietnam - Ho Chi Minh Ville located in Ho Chi Minh city (PUF-HCM).

1.1.1 PUF-HCM

PUF-HCM[1] is a department of VietNam National Univeristy at Ho Chi Minh city. From the first year of operations, PUF-HCM launched the quality training programs from France in VietNam. With target, bring the programs which designed and evaluated by the international standards for Vietnamese student. PUF-HCM always strive in our training work. So far, PUF-HCM have five linking programs with the universities in France, and the programs are organized into the subjects: Commerce, Economic, Management and Informatics. In detail:

- Bachelor and Master of Economics : linking program with University of Toulouse 1 Captiole
- Bachelor and Master of Informatics: linking program with University of Bordeaux and University of Paris 6.

The courses in PUF-HCM are provided in French, English and Vietnamese by both Vietnamese and French professors. The highlight of the programs are inspection and diploma was done by the French universities.

[1] `http://pufhcm.edu.vn`

1.2 Laboratoire Bordelais de Recherche en Informatique

The Laboratoire Bordelais de Recherche en Informatique (LaBRI)[2] is a research unit associated with the CNRS (URM 5800), the University of Bordeaux and the Bordeaux INP. Since 2002, it has been the partner of Inria. It has significantly increased in staff numbers over recent years. In March 2015, it had a total of 320 members including 113 teaching/research staff (University of Bordeaux and Bordeaux INP), 37 research staff (CNRS and Inria), 22 administrative and technical (University of Bordeaux, Bordeaux INP, CNRS and Inria) and more than 140 doctoral students and post-docs. The LaBRI's missions are: research (pure and applied), technology application and transfer and training. Today the members of the laboratory are grouped in six teams, each one combining basic research, applied research and technology transfer:

- Combinatorics and Algorithmic
- Image and Sound
- Formal Methods
- Models and Algorithms for Bio-informatics and Data Visualisation
- Programming, Networks and Systems
- Supports and Algorithms for High Performance Numerical Applications

Within these team, research activities are conducted in partnership with Inria. Besides that, LaBRI also collaborate with many other laboratories and companies on French, European and the international.

[2] `http://www.labri.fr`

1.3 Heritage Observation project

Heritage Observatory is the project has for aim to identify cultural and historical heritages, constitute a specific database, provide a data archive free for all. This platform allows us to receive the breaking news about cultural or historical heritage sites located in regions of the world that we are interesting. In order to do that, the platform will collect a huge of images about the cultural, historical heritage and then classify them into different categories.

The Heritage repository contain a huge set of images about vietnamese human, statue, ancient artifacts, building, landscape, etc. But all of them are completed unlabeled. So, the question is: "How we can classify those images to the right classes and automatic classify the new image, which the machine has never seen to the right class".

The first attempt is to categorize those images into four different classes including:

Heritage
: A place of cultural, historical, or natural significance for a group or society.

Beings
: Any form of life, such as a plant or a living creature, whether human or other animal.

Scenery
: Any form of landscapes which show little or no human activity and are created in the pursuit of a pure, unsullied depiction of nature, also known as scenery.

Other
: Any other type of image that doesn't represent a photograph, such as painting, illustration, any object.

(a)

(b)

(c)

(d)

Figure 1.1: Images from Heritage repository

1.4 The internship project

The internship is intended to be a duration to apply academic knowledge to professional environment. It shows the ability synthesis, evaluation and self-research of student. Besides, the student may study the experience from the real working environment. My internship is done under the guidance of Mrs Marie BEURTON-AIMAR in a period of six months at LaBRI laboratory.

The objective of this internship is implementing a method to automatize the classification of images.

The goal is to use deep convolutional neural networks (CNNs or ConvNets) to tackle the image classification task.

Chapter 2

Analysis

2.1 State of the art

Labeling an image according to a set of semantic categories is the goal of image classification. This is a very challenging problem because there is usually a large amount of intra-class variability, arising from different lighting conditions, misalignment, non-rigid deformations, occlusion and corruptions. Numerous efforts have been made to counter the intra-class variability by manually designing low-level features for classification tasks. Representative examples are Gabor features and local binary patterns (LBP) for texture and face classification[1], SIFT and HOG features for object recognition[2]. Although the low-level features can be hand crafted with great success for certain data and tasks, designing effective features for new data and tasks usually requires new domain knowledge because most hand-crafted features cannot simply be adapted to new conditions [3], [4].

Learning features from data of interest is considered as a plausible method of remedying the limitations of hand-crafted features. This rapidly giving way to deep neural networks (DNNs).

Artificial neural networks take inspiration from models of the biological brain, and try to reproduce some of its functions by using simple but massively interconnected processing units, the neurons. A typical neural network architecture comprises several layers of neurons feeding one another, by which the "deep" attribute. Deep learning has provided impressive results in object recognition and image classification [5] [6] [7] [8], showing always a great potential.

This work adds another piece of evidence in this sense. We use deep convolutional neural networks (CNNs or ConvNets) to tackle the image classification task. Four recently proposed promising architectures, AlexNet [5], VGG [9], GoogLeNet [6] and ResNet [8], are considered and tested. To cope with the scarcity of training image dataset, we explore various training modalities: not only the usual training from scratch but also the fine-tuning of pre-trained networks. Beside that, we try to modify the CNN parameters to obtain the better result.

In this work, we use Caffe[10] as our main library framework since it is one of the most popular libraries for deep learning (convolutional neural networks in particular). It is developed by the Berkeley Vision and Learning Center (BVLC) and community contributors. Caffe is easily customizable through configuration files, easily extendible with new layer types, and provides a very fast ConvNet implementation (leveraging GPUs, if present). It provides C++, Python and MATLAB APIs.

2.1.1 General CNN architecture

A convolutional network is a neural network that use convolutions. It is a multiplayer network (e. g. it uses several layers). In reality, those kind of network, are dividing in two part. The first one use convolutional layers - layers that use convolution patches to compute weights used in neurons. The second part, used to connect the first part to the output, is made of fully connected layers: every output of a layer is connected to every neurons of the next one without any distinctions.

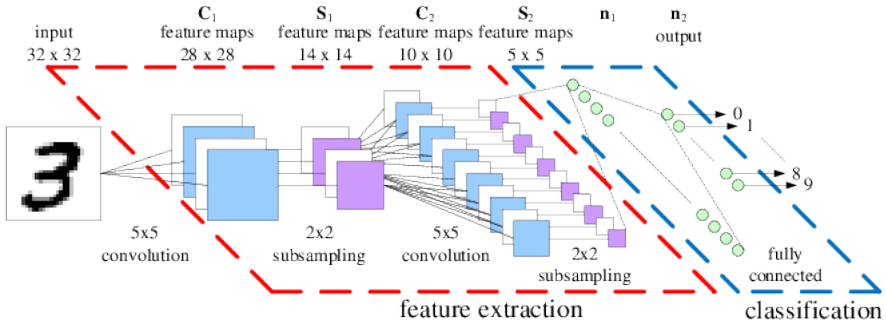

Figure 2.1: An Example CNN architecture for a handwritten digit recognition task.

The network architecture of an example CNN is depicted in Fig 2.1. The processing starts with feature extraction layers and is finished by fully connected classification layers. Using different layers delivers robust recognition accuracy and is invariant to small geometric transformations of the input images. The robust recognition accuracy makes that CNN are successfully used for classification tasks on real world data.

2.2 Scikit-learn algorithm cheat-sheet

Because it has many algorithms has able to classify images such as: Decision tree[11], SVM[12], K-nearest neighbors[13], etc. We need to find a right one to solve our problem. The Scikit-learn[14] had made an algorithm cheat-sheet to help us to choose the right one:

This algorithm cheat-sheet(see Fig 2.2) suggests to use K-mean algorithm to classify our images because of unlabeled images and, the size of our dataset is less than 10K samples. However, one can note that it is mandatory to create a labeled dataset from our unlabeled images by hand and then, we can use SGD Classifier to do it.
Finally, two ways are proposed to reach our goal: one is use Supervised Classification with SGD is our main algorithm. The other is use Unsupervised Classification with

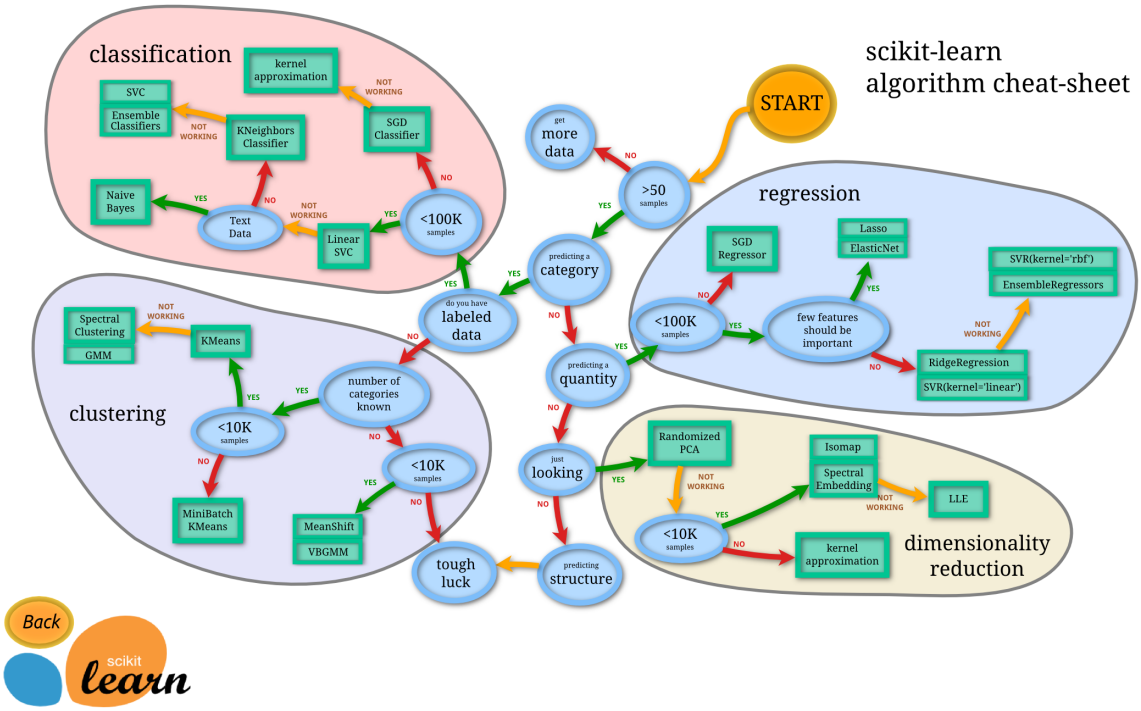

Figure 2.2: Scikit-learn algorithm cheat-sheet

K-mean is main algorithm. Both of them can be applied by using Convolution Neural Network(CNN).

2.3 Dataset

2.3.1 Heritage Repository

The Heritage repository contain a huge set of images about vietnamese human, statue, ancient artifacts, building, landscape, etc. We collected all images and build a labeled images dataset based on those images for both supervised classification and unsupervised calssification tasks.

2.3.2 ImageNet

ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers using Amazon's Mechanical Turk crowd-sourcing tool. Starting in 2010, as part of the Pascal Visual Object Challenge, an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) has been held. ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images.

Because of the similarities between our images (from Heritage Repository) and images from ImageNet. So, we choose to finetune our model from the pre-trained model of this dataset.

2.4 Image classification

2.4.1 Supervised classification

Supervised deep learning requires us to provide a training dataset, which includes a list of images labeled. However, our images are not, and we can't classify the entire dataset by hand. So, we propose the following method, which includes three steps:

- Train a Convolution Neural Network (CNN) model with a part of images based on the original images, which we mention in subsection 2.3.1
- Use the CNN model trained above to classify the entire original images
- Calculate the performance of the classification by random images in each class to review

In this work, we will implement several different CNN Model Architectures and make a comparison between them to choose the best one, which perfect match with our requirements and dataset. We also try to improve the result by make some slover parameter adjustments.

AlexNet

VGGNet

GoogLeNet

ResNet

2.4.2 Unsupervised classification

Because of when classifying images by hand, it has some special case when one image may refer to more than one class. Thus, we need the machine to help us to make decisions by comparing two images are the same class or not. We also want to separate images of one class into multi unknown sub-classes. So, by using Unsupervised Deep Learning, we want to let the machine to classify a set of images unlabeled into some unknown classes.

In this work, we use the Unsupervised Deep Embedding for Clustering Analysis (DEC)[15], which propose a new method that simultaneously learns feature representations and cluster assignments using deep neural networks to classify the unlabeled images.

2.5 The ImageNet Visual Recognition Challenge

The ImageNet Visual Recognition Challenge[1] is a competition where research teams evaluate their algorithms on the given data set, and compete to achieve higher accuracy on several visual recognition tasks such as Classification, Classification with localization or Fine-grained classification.
Because they have the same kind of goal as our project, so, we will try to implement the CNN model of the winner of this challenge and also try to improve it to get the better result.

[1] `http://www.image-net.org/challenges/LSVRC/2012`

Chapter 3

Conception

Chapter 4

Realization

4.1 The Dataset

This chapter describes how we obtain and optimize the dataset.

4.1.1 Fetch all images

The entire image dataset described on the text file named "photos.txt" line by line. Each line includes the image id and image description as below:

```
5a36f382-dbdf-11e6-95fd-d746d863c3eb | Những người ăn xin  | vie
5a36f382-dbdf-11e6-95fd-d746d863c3eb | Mendiants  | fra
17be8122-dbe0-11e6-860c-5fea02802d0a | Chợ Cũ (3) | vie
17be8122-dbe0-11e6-860c-5fea02802d0a | Vieux marché (3) | fra
400286c8-dbe1-11e6-bb4d-ff975c68de04 | Ngân hàng Đông Dương  | vie
400286c8-dbe1-11e6-bb4d-ff975c68de04 | La Banque de l'Indochine  | fra
```

In order to get the image dataset, we have to fetch each image one by one by join the image id with heobs cdn url https://cdn.heobs.org/photo/. For example, with the first line in the record above, we have the following URL:

```
https://cdn.heobs.org/photo/5a36f382-dbdf-11e6-95fd-d746d863c3eb
```

We wrote a python script to automatic read this text file & download images one by one. Totally, we have 142459 images in our dataset.

4.1.2 Remove broken images

After downloaded & look around all images, we found that it has a lot of broken images, which can't be displayable. So, we write a python script to filter all of those broken images automatically.
Finally, we have 89850 images left in our dataset.

4.1.3 Labeling image dataset

It's mandatory to label our images for both Supervised and Unsupervised Classification because with Unsupervised Classification, we need a labeled dataset to validate if the image is classified correct or not. With Supervised Classification, we need a labeled dataset to train & validate the Neural Network Model.

Before labeling our images by hand, we need to build a good images dataset, which has the same image resolution. We loop through all our images to filter all images, which has the most common image resolution and reject all others. After this step, we have 55143 images left.

Finally, we labeling our images by our self and completed our dataset, which describe as the table below:

	Being	Heritage	Scenery
N° images	1471	1832	942

4.2 Unsuppervised Deep Learning

The DEC has two phases:

Phase 1: Parameter initialization with a deep autoencoder

Phase 2: Parameter optimization (i.e., clustering)

We already tried to run this implement code with the latest Caffe version (Jul 2017), but it doesn't work because this Caffe model required some deprecated model parameters. However, because of the code delivery with Caffe and Docker, so, we continue to try with Docker and their Caffe version.

The Dockerfile in the original source code has exception but fixed by the following patch:

```
-  liblmdb-dev libboost1.54-all-dev libatlas-base-de
+  liblmdb-dev libboost1.54-all-dev libatlas-base-dev
```

4.2.1 Trying to run DEC with MNIST dataset

We'll first try to test with `MNIST`[1] dataset - the simplest example provided with the source code.
In parameter optimization phase, they use KMeans Cluster to try to predict the closest cluster each sample in the test image belongs to. The most important part of this phase is:

[1] `http://yann.lecun.com/exdb/mnist/`

```
if (Y_pred != Y_pred_last).sum() < 0.001*N:
    print acc_list
    return acc, nmi
```

This is the terminate, the application achive the goal when the current predict (Y_pred) is not different with the last predict. The value 0.001*N is threshold.

Generate the test summary result

Currently, the code just show the accuracy of the whole process. We need to write we own function to calculate the accuracy of each classes. Because with mnist dataset, we already have the correct label of each image. So, to be able to summary the result, we add the following function to compare between predict & actual result:

```
def show_result(predicts, actuals):
    summary = {}
    for idx, value in enumerate(actuals):
        actual = str(value)
        predict = str(predicts[idx])
        if actual in summary:
            if predict in summary[actual]:
                summary[actual][predict] += 1
            else:
                summary[actual][predict] = 1
        else:
            summary[actual] = {}
    print summary
```

And call this function when the application predict done

```
if (Y_pred != Y_pred_last).sum() < 0.001*N:
     classify_dataset(Y_pred, img, db)
     show_result(Y_pred, Y)
     print acc_list
     return acc, nmi
```

Classify image dataset into X classes

This feature put the images from dataset, which has the same class into one folder. This helps us to easy to verify the classification result

```
def classify_dataset(predicts, imgs, db):
    classes_dir = "classes_" + db
    if not os.path.isdir(classes_dir):
        os.makedirs(classes_dir)
    for idex, pred in enumerate(predicts):
        tmp_dir = os.path.join(classes_dir, str(pred))
        if not os.path.isdir(tmp_dir):
```

```
            os.makedirs(tmp_dir)
        dispSingleImg(imgs, idex, os.path.join(tmp_dir, str(idex) + ".jpg"))
```

We call this function when the application predict done

```
if (Y_pred != Y_pred_last).sum() < 0.001*N:
     classify_dataset(Y_pred, img, db)
     print acc_list
     return acc, nmi
```

Convert binary images into displayable images

Because of the images used in the application is stored in binary & read directly into memory. So, to be able to get the classified result, we've to convert image stored in memory (RAM) to the displayable image file.

```
def dispSingleImg(X, n, fname=None):
    h = X.shape[1]
    w = X.shape[2]
    c = X.shape[3]
    buff = np.zeros((h, w, c), dtype=np.uint8)

    buff[0:h, 0:w, :] = X[n]

    if fname is None:
        cv2.imshow('a', buff)
        cv2.waitKey(0)
    else:
        cv2.imwrite(fname, buff)
```

X: array of image dataset

n: index of image

fname: destination to save

Acquire and analytics the test result

After run this network model with the **init.caffemodel**, we acquire the result describe as the following table:

	Predict 0	1	2	3	4	5	6	7	8	9
Actual 0	17	35	53	15	1	141	6629	1	4	6
1	7658	14	16	8	153	3	1	9	2	3
2	278	15	20	229	6094	42	73	29	112	87
3	108	128	13	178	171	6	12	23	67	6434
4	65	8	3060	8	7	58	3	3612	4	0
5	60	4968	41	133	6	54	7	37	13	899
6	233	268	2	37	14	6124	187	9	0	1
7	251	16	336	17	103	1	22	317	6226	3
8	167	142	62	5922	32	36	73	72	23	295
9	80	12	3624	60	5	16	47	2910	97	106

Table 4.1: Mnist init.caffemodel test result

Because Unsupervised Classification don't know what classes actually are. It just grouping the images, which it think there's the same into one class. So, we've to figure what these classes are.

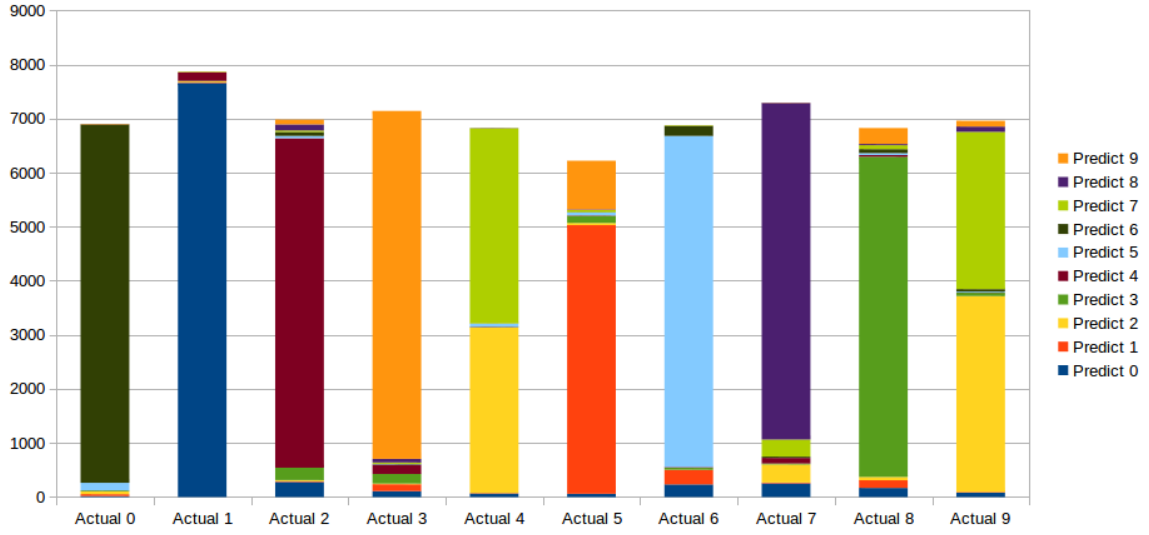

Figure 4.1: Mnist init.caffemodel test result - correct

As the result above, we can easy to figure out that almost it can predict the images belong to which classes in high performance. Although it has a little confuse when predicting 2 and 7 but we can understand because of this two digit is similar.

Now we know how to receive the result of this implement source code. However, because of Mnist dataset is too simple. It just contains images, which has 28x28 pixel and only black & white colors while our image Dataset has both bigger and true color. So, we need to try with the more complex Dataset.

4.2.2 Trying to run DEC with STL dataset

In the test datasets they provide us has STL dataset, which closest with our Dataset (96 x 96 pixel, true color). We're going to try with this Dataset before try with we own dataset.

The application failure when execute **make_stl_data.py** python file to create STL LevelDB because of the "features.pyx" file is not delivery with the source code. The author of DEC mention to this file located at pyvision project[2]. However, when testing this file, we got another error message:

```
Type Error: "int" object is not iterable
```

As the author say "It's been too long and I almost forgot about it, sorry. But you can try to refer to stackoverflow. I believe you can find your answer"[3]. So, after some debug & research, we found that the problem caused by the parameter of **function hog** in **feature.pyx** is an image object. But we provided an numpy object. So, we fixed it by replaced the following line in **features.pyx**

```
line 48.  with, height = im.size
```

by the following line:

```
line 48.  with, height = im.shape[:2]
```

Finally, it works. The accuracy result is 35.7% (35.9% in the paper).

4.2.3 Preparing we own image dataset

Because the program require our dataset must be on LevelDB database and in specific format. However, our dataset just contain raw images. So, we have to write we own function to convert our dataset to ther dataset format.

Below is the code we using to generate & save data into LevelDB:

```
import sys
import os
import Image
import numpy as np
import cv2
import cv
from joblib import Parallel, delayed
import features
import random
import dec
import pdb

def mode():
```

[2]https://github.com/cvondrick/pyvision/blob/master/vision/features.pyx
[3]https://github.com/piiswrong/dec/issues/1

```
    if "MODE" in os.environ:
        return os.environ['MODE']
    return 'validate'


def load_data(images_dir):
    ims = [read(os.path.join(images_dir, filename)) for filename in
        os.listdir(images_dir)]
    X = np.array(ims, dtype='uint8')
    n_jobs = 10
    cmap_size = (6, 10)
    N = X.shape[0]

    H = np.asarray(Parallel(n_jobs=n_jobs)(delayed(features.hog)(X[i]) for
        i in xrange(N)))

    H = H.reshape((H.shape[0], H.size / N))

    X_small = np.asarray(Parallel(n_jobs=n_jobs)(delayed(cv2.resize)(X[i],
        cmap_size) for i in xrange(N)))
    crcb =
        np.asarray(Parallel(n_jobs=n_jobs)(delayed(cv2.cvtColor)(X_small[i],
        cv.CV_RGB2YCrCb) for i in xrange(N)))
    crcb = crcb[:, :, :, 1:]
    crcb = crcb.reshape((crcb.shape[0], crcb.size / N))

    feature = np.concatenate(((H - 0.2) * 10.0, (crcb - 128.0) / 10.0),
        axis=1)
    print feature.shape

    return feature, X[:, :, :, [2, 1, 0]]


def load_label(images_dir, classes, determine):
    return np.array([get_label(classes, filename, determine) for filename
        in os.listdir(images_dir)], dtype='uint8')


def get_label(classes, filename, determine):
    if mode() == 'validate':
        return classes.index(filename.split(determine)[0])
    return 0


def load_named_label(images_dir):
    return np.array([filename for filename in os.listdir(images_dir)],
        dtype='str')

if __name__ == '__main__':
```

```
classes = ["heritage", "being", "scenery"]
images_dir = sys.argv[1]
read = lambda imname: np.asarray(Image.open(imname).convert("RGB"))
if os.path.isdir(images_dir):
    X_train, img_train = load_data(images_dir)
    Y = load_label(images_dir, classes, "_")
    labeled = load_named_label(images_dir)
    p = np.random.permutation(X_train.shape[0])
    X_total = X_train[p]
    Y_total = Y[p]
    labeled_total = labeled[p]
    np.save("custom_named_label", labeled_total)
    img_total = img_train[p]
    dec.write_db(X_total, Y_total, 'custom_total')
    dec.write_db(img_total, Y_total, 'custom_img')
    N = X_total.shape[0] * 4 / 5
    dec.write_db(X_total[:N], Y_total[:N], 'custom_train')
    dec.write_db(X_total[N:], Y_total[N:], 'custom_test')
else:
    raise Exception("Please specific image url")
```

The dataset must contain all images in the same root folder and each images in the dataset must have the following format :

`<image class>_<index>.jpg`

Run the following command in the source root to create our custom dataset:

`python make_data.py <image dataset path>`

4.2.4 Train with our dataset

After train with our dataset, we get accurency 68.74%

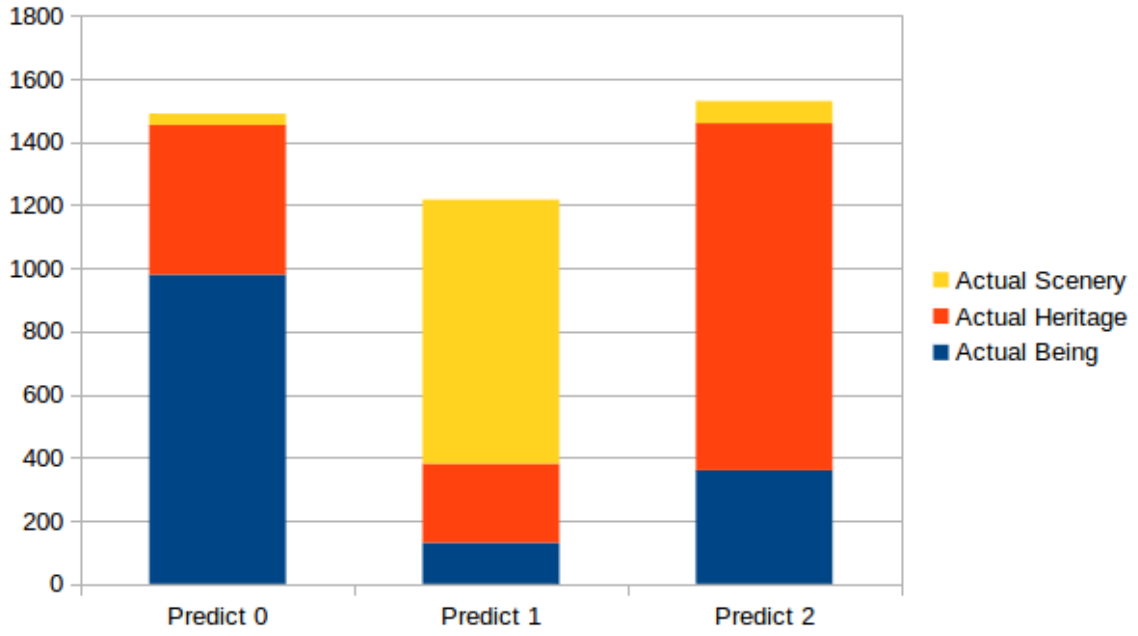

Figure 4.2: Unsupervised train result

4.2.5 Classify the whole images

.....

4.3 Suppervised Deep Learning

- *Convolutional layers* The convolutional layer is the core building block of a CNN. This layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

  Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

- *Pooling layers* Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to

reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The pooling operation provides another form of translation invariance.

- *ReLU layers* The ReLU layer applies the function f(x) = max(0, x) to all of the values in the input volume. In basic terms, this layer just changes all the negative activations to 0.This layer increases the nonlinear properties of the model and the overall network without affecting the receptive fields of the convolution layer.

- *Dropout layers* During training only, sets a random portion of x to 0, adjusting the rest of the vector magnitude accordingly. Because a fully connected layer occupies most of the parameters, it is prone to overfitting. One method to reduce overfitting is dropout. At each training stage, individual nodes are either "dropped out" of the net with probability $1 - p$ or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. Only the reduced network is trained on the data in that stage. The removed nodes are then reinserted into the network with their original weights.

- *Fully-connected layers* This layer basically takes an input volume and outputs an N dimensional vector where N is the number of classes that the program has to choose from. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

Bibliography

[1] X. Tan and B. Triggs, *Fusing Gabor and LBP Feature Sets for Kernel-Based Face Recognition*, pp. 235–249. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[2] S. Nigam, M. Khare, R. K. Srivastava, and A. Khare, "An effective local feature descriptor for object detection in real scenes," in *2013 IEEE Conference on Information Communication Technologies*, pp. 244–248, April 2013.

[3] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, July 2006.

[4] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, May 2017.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[11] J. QUINLAN, "Induction of decision trees," aug 1985.

[12] S. R. Gunn, "Support vector machines for classification and regression," oct 1998.

[13] O. Anava and K. Y. Levy, "k*-nearest neighbors: From global to local," jan 2017.

[14] scikit learn.org, "Scikit-learn algorithm cheat-sheet."

[15] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," *CoRR*, vol. abs/1511.06335, 2015.