IJCRT.ORG ISSN : 2320-2882
INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

DEEP LEARNING IN ARCHAEOLOGICAL DISCOVERY

Shibakali Gupta[C], Mrinmoy Ghosh[2], Spandan Mondal[3]

Department of Computer Science & Engineering, University Institute of Technology
Burdwan, West Bengal, India

Abstract

It's important to save archaeological monuments as they play a pivotal part in helping us understand mortal history and their accomplishments for times with no or little spoken sources. The first step for this purpose is an effective system for collecting and establishing information about objects of interest for archaeologists. One of the best way to automate this process is using deep neural networks. Archaeology is the scientific study of mortal history. Computers have been used in the field of archaeology for multitudinous times and it has now come an essential wide tool for archaeologists. As the use or operation of computers becomes increasingly necessary to the work of the archaeologist, which they bear a clear understanding of the impact of information technology upon their discipline. In this work, we propose a hierarchical Convolutional Neural Network (CNN) model to classify archaeological objects, *artifacts*. This paper explains the influence and the development of computers on all aspects of archaeological disquisition and interpretation, from check, excavation and m*useums* to galleries, education and communicating the history. To compare and validate our system, we run trials on the same data set using deep knowledge model. The instigation of the computers can begin with the appraisal at all stages of archaeological disquisition and data analysis. The main themes to materialize are the eventuality of computers as active agents for study rather than as just unresisting tools, and the symbiotic relationship between the development of digital technologies and archaeological proposition.

Keywords - Archaeology, CNN, Artifacts, Museums, Excavation.

I. INTRODUCTION

Archaeology studies physical remains of objects in order to understand mortal history and culture for times with no or little spoken sources. It helps us have a regard at the lives of people in the history and how effects have changed through time. While history uses written records to dissect events and explain mortal life, archaeology investigates what humans made and left before and makes sense of how, where, and when they lived. Places with physical remains of mortal conditioning in the history are called archaeological spots. Archaeological spots can be of different types like agreements, cemeteries or grounds. They can be visible or not visible above ground. Studying these accoutrements informs us of unlisted mortal history in the history, and therefore it is important to cover and conserve archaeological spots. To do so, it's necessary to first identify and validate similar spots. Deep literacy has shown great eventuality in automating processes in numerous operations and outperformed classical machine literacy styles. It has performed well in image bracket, machine restatement, speech recognition, image captioning, healthcare sphere operations, vaticination of events, and numerous further. It can work with 2D image data, 3D point pall data, hyperactive- spectral image data, and interpret sequence to sequence data. Among deep literacy models, Convolutional Neural Networks(CNN) have been proved to work well on image data, while piled bus- encoders are more suitable for learning features from the input data and garbling them to a compressed vector, from which a decoder learns to induce the original input data. Deep literacy models Generally bear a lot of training data to be suitable to generalize well. In numerous operations where the quantum of training data isn't sufficient, transfer literacy can be applied. Transfer literacy refers to training a model on a large set of training data first, and also using the pre-trained model as a point extractor for a new operation with small dataset. There are numerous models pre-trained on ImageNet data; Xception,VGG.Net, ResNet, Inception, etc, which are shown to perform well as point extractors for other image- grounded disciplines.

II. OBJECTIVES

In recent years, it has become clear that archaeologists will only be able to harvest the full potential of computer technology if they become aware of the specific pitfalls and potentials inherent in the archaeological data and research process. Archaeoinformatics (AI) science is an emerging discipline that attempts to uncover, quantitatively represent and explore specific properties and patterns of archaeological information. Fundamental research on data and methods for a self-sufficient archaeological approach to information processing produces quantitative methods and computer software specifically geared towards archaeological problem solving and understanding. AI science is capable of complementing and enhancing almost any area of scientific archaeological research. It incorporates a large part of the methods and theories developed in quantitative archaeology since the 1960s but goes beyond former attempts at quantifying archaeology by exploring ways to represent general archaeological information and problem structures as computer algorithms and data structures. This opens archaeological analysis to a wide range of computer-based information processing methods fit to solve problems of great complexity. It also promotes a formalized understanding of the discipline's research objects and creates links between archaeology and other quantitative disciplines, both in methods and software technology.

III. PROCESS FLOW

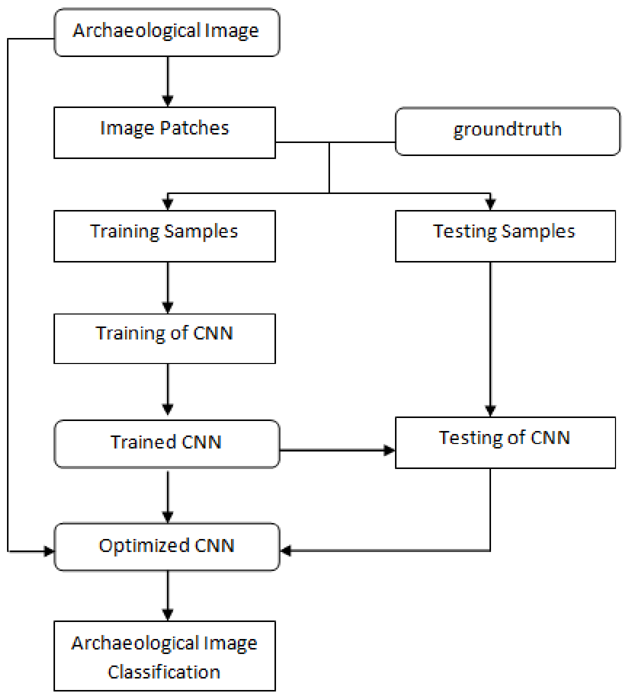


Figure 1. process flow

IV. ALGORITHM

Step 1: Choose a Dataset

Step 2: Prepare Dataset for Training and Testing.

Step 3 : Normalizing dataset

Step 4: Shuffle the Dataset

Step 5: Assigning Labels and Features

Step 6 : Define, compile and train the CNN Model

Step 7 : Accuracy and Score of model.

V. DATA SET

The training dataset contains a total of 1020 images in color, divided into three different image classes, e.g. Pot, Durga idol and Ganesh idol and the test dataset contains a total of 60 images where each class contains 20 images. All the images are in 224 X 224.

Figure 2. sample dataset of pot

Figure 3. sample dataset of ganesh idol.

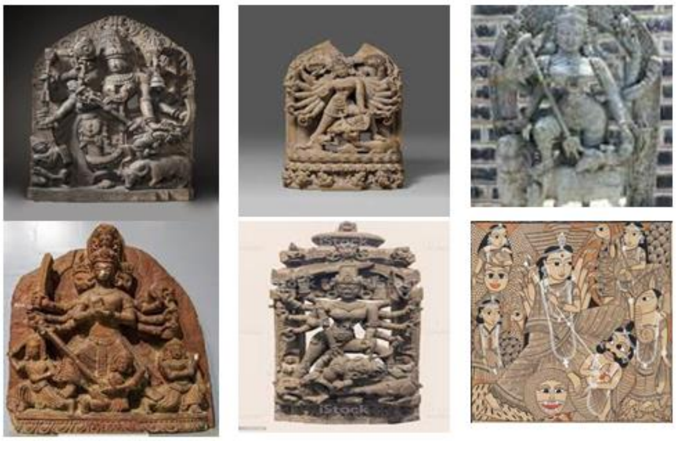


Figure 4. sample dataset of durga idol.

VI. TOOLS & TECHNOLOGIES USED

The thesis uses free GPUs from Google Colab (Colaboratory) and the deep learning framework used is PyTorch.

VII. ARCHITECTURE OF THE MODEL

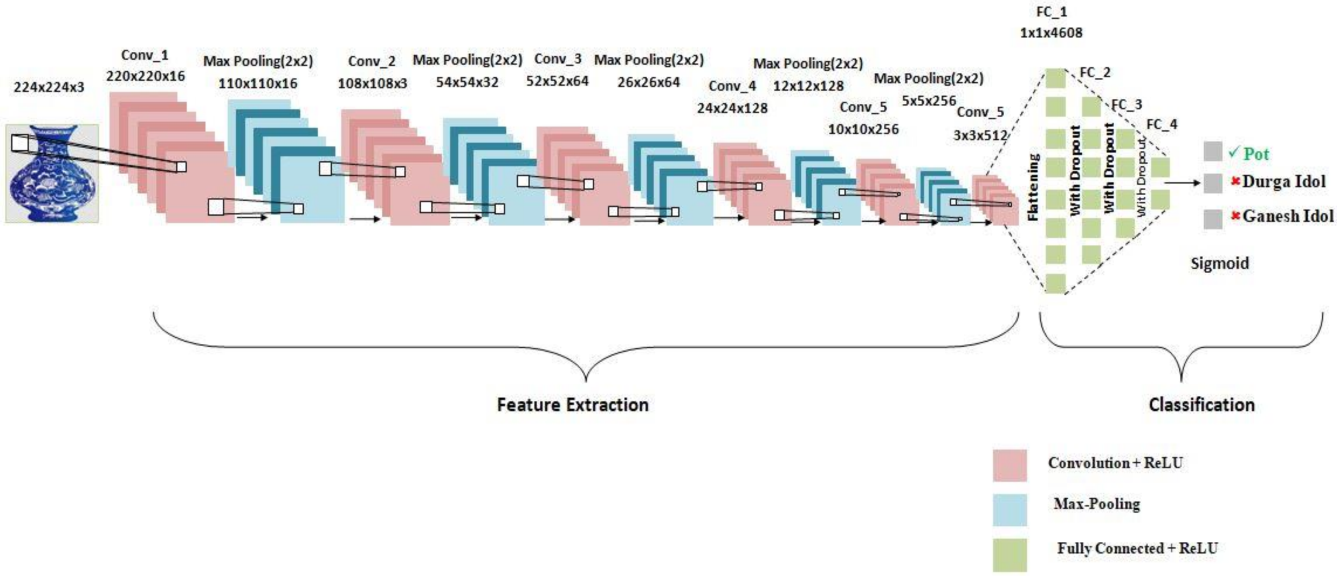


Figure 5. a cnn model for an archaeological recognition system.

Our CNN models are multi-class classifiers consisting of 6 Convolutional layers each of which is followed by a max pooling function and a ReLU operation which is defined in the following equation.

$$ReLU(x) = max(x, 0) \quad (1)$$

In the end, there is one fully connected layer with a Sigmoid function outputting the probabilities for each class label. The structure used for the CNN models is depicted in Figure 5.

First Layer:
The input for ConvNet is a 224×224 colour image which passes through the first convolutional layer with 16 feature maps or filters having size 5×5 and a stride of one. The image dimensions changes from 224×224×1 to 220×220×16.

Second Layer:
Then the ConvNet applies max pooling layer or sub-sampling layer with a filter size 2×2 and a stride of one. The resulting image dimensions will be reduced to 110×110×16.

Third Layer:
The input for 2nd Convolutional layer is a 110×110 colour image which passes through the convolutional layer with 16 feature maps or filters having size 3×3 and a stride of one. The image dimensions changes from 110×110×16 to 108×108×32.

Fourth Layer:
Then the ConvNet applies max pooling layer or sub-sampling layer with a filter size 2×2 and a stride of one. The resulting image dimensions will be reduced to 54×54×32.

Fifth Layer:
The input for 3rd Convolutional layer is a 54×54×32 colour image which passes through the convolutional layer with 32 feature maps or filters having size 3×3 and a stride of one. The image dimensions changes from 54×54×32 to 52×52×64.

Sixth Layer:
Then the ConvNet applies max pooling layer or sub-sampling layer with a filter size 2×2 and a stride of one. The resulting image dimensions will be reduced to 26×26×64.

Seventh Layer:
The input for 4th Convolutional layer is a 26×26×64 colour image which passes through the convolutional layer with 64 feature maps or filters having size 3×3 and a stride of one. The image dimensions changes from 26×26×64 to 24×24×128.

Eighth Layer:
Then the ConvNet applies max pooling layer or sub-sampling layer with a filter size 2×2 and a stride of one. The resulting image dimensions will be reduced to 12×12×128.

Ninth Layer:

The input for 5th Convolutional layer is a 12×12×128 colour image which passes through the convolutional layer with 256 feature maps or filters having size 3×3 and a stride of one. The image dimensions changes from 12×12×128 to 10×10×256.

Tenth Layer:

Then the ConvNet applies max pooling layer or sub-sampling layer with a filter size 2×2 and a stride of one. The resulting image dimensions will be reduced to 5×5×256.

Eleventh Layer:

The input for 6th Convolutional layer is a 5×5×256 colour image which passes through the convolutional layer with 256 feature maps or filters having size 3×3 and a stride of one. The image dimensions changes from 5×5×256 to 3×3×512.

Twelve Layer:

The twelve layer is a fully-connected (FC1) convolutional layer and activation function ReLu used. Each of the units in twelve layer is connected to all the 4608 nodes (3×3×512) in the eleventh layer.

Thirteen Layer:

The thirteen layer is a fully-connected (FC2) convolutional layer with dropout function and activation function ReLu used.

Fourteen Layer:

The fourteen layer is a fully-connected (FC3) Convolutional layer with dropout function and activation function ReLu used.

Fifteen Layer:

The fifteen layer is a fully-connected (FC4) Convolutional layer with dropout function and activation function ReLu used.

Output Layer:

Finally, there is a fully connected sigmoid output layer with 3 possible values corresponding to the digits from 0 to 2.

VIII. LOSS FUNCTION AND OPTIMIZER

This criterion computes the cross entropy loss between input and target. It is useful when training a classification problem with *C* classes. If provided, the optional argument weight should be a 1D *Tensor* assigning weight to each of the classes. This is particularly useful when you have an unbalanced training set.

torch.optim is a package implementing various optimization algorithms. Most commonly used methods are already supported, and the interface is general enough, so that more sophisticated ones can be also easily integrated in the future. Here, Adam optimizer algorithm used.

IX. TRAINING THE MODEL

Below, we have a function that performs one training epoch. It enumerates data from the DataLoader, and on each pass of the loop does the following:

- Gets a batch of training data from the DataLoader
- Zeros the optimizer's gradients
- Performs an inference - that is, gets predictions from the model for an input batch
- Calculates the loss for that set of predictions vs. the labels on the dataset
- Calculates the backward gradients over the learning weights
- Tells the optimizer to perform one learning step - that is, adjust the model's learning weights based on the observed gradients for this batch, according to the optimization algorithm we chose
- It reports on the loss for every 1 batches.

Finally, it reports the average per-batch loss for the last batches, for comparison with a validation run.

X. SAVING AND LOADING MODELS

When loading model weights, we needed to instantiate the model class first, because the class defines the structure of a network. We might want to save the structure of this class together with the model, in which case we can pass model to the saving function.

XI. EXPERIMENT RESULT

```
[19] dataiter = iter(test_loader)
     images, labels = dataiter.next()

     classes = ('Durga', 'Ganesh', 'Pot')
     #print(labels)
     # print images
     imshow(torchvision.utils.make_grid(images[0]))
     print('GroundTruth: ', ' '.join(f'{classes[labels[j]]:5s}' for j in range(1)))
```

```
GroundTruth:  Pot
```

```
images=images.to(device)
outputs = model(images)
_, predicted = torch.max(outputs, 1)

print('Predicted: ', ' '.join(f'{classes[predicted[j]]:5s}'
                              for j in range(1)))
```

```
Predicted:  Pot
```

Figure 6. experiment result-1.

```
[90] dataiter = iter(test_loader)
     images, labels = dataiter.next()

     classes = ('Durga', 'Ganesh', 'Pot')
     #print(labels)
     # print images
     imshow(torchvision.utils.make_grid(images[0]))
     print('GroundTruth: ', ' '.join(f'{classes[labels[j]]:5s}' for j in range(1)))
```

```
GroundTruth:  Durga
```

```
images=images.to(device)
outputs = model(images)
_, predicted = torch.max(outputs, 1)

print('Predicted: ', ' '.join(f'{classes[predicted[j]]:5s}'
                              for j in range(1)))
```

```
Predicted:  Durga
```

Figure 7. experiment result-2.

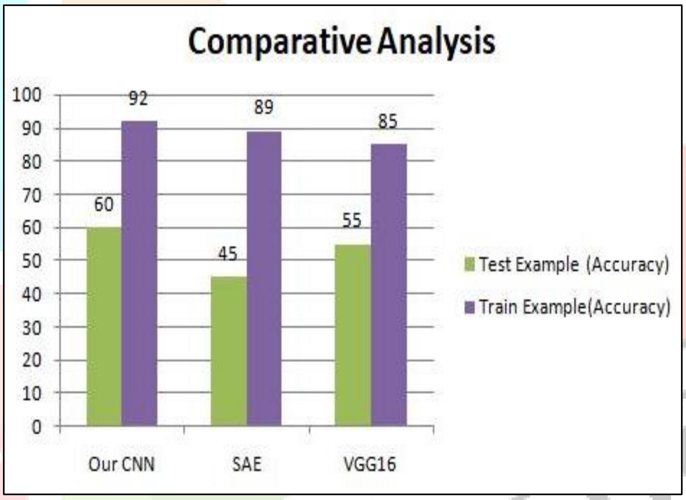
XII. MODEL EVALUATION ON THE TEST SET

```
[34] correct = 0
     total = 0
     # since we're not training, we don't need to calculate the gradients for our outputs
     with torch.no_grad():
         for i,(images, labels) in enumerate(test_loader):
             # Move tensors to the configured device
             images = images.to(device)
             labels = labels.to(device)
             # calculate outputs by running images through the network
             outputs = model(images)
             # the class with the highest energy is what we choose as prediction
             _, predicted = torch.max(outputs.data, 1)
             total += labels.size(0)
             correct += (predicted == labels).sum().item()
             #print(total)

     print(f'Accuracy of the network on the 60 test images: {100 * correct // total} %')

Accuracy of the network on the 60 test images: 60 %
```

XIII. COMPARATIVE ANALYSIS



XIV. CONCLUSION

DL models are gaining importance in the realm of archaeology application, offering higher accuracy than any previous non-intrusive approach. There is a potential research gap in which deep learning may be utilized for archaeological application in diverse structures classification. DL methods can assist archaeologists in detecting objects faster than standard conventional methods whilst saving money and time as shown in the articles that has been reviewed. To date, there is no open and public access standard data, that are utilized by archaeologists worldwide; therefore, a standard dataset highly can assist archaeologists to evaluate ML and DL models in object detection and classification.

XV. REFERENCES

[1] Juan Barcelo, "*Computational Intelligence in Archaeology*" , July-2008, DOI: 10.4018/978-1-59904-489-7

[2] Lorenzo Mantovan & Loris Nanni, "*The Computerization of Archaeology: Survey on Artificial Intelligence Techniques*", *SN Computer Science*, 14 August 2020, s42979-020-00286-w

[3] Teija Oikarinen & Helena Karasti, " *Exploring eScience for Archaeology: Digitalization of Archaeological Data in the Context of Digital Curation*", January 2012

[4] Dr.AR.Saravanakumar, S.Paranthaman, "*Recent Development Of Computer Applications In Archaeology*", *IJRAR*, June 2019 P- ISSN 2349-5138, Volume 6, Issue 2

[5] Josep Puyol-Gruart, "*Computer science, artificial intelligence and archaeology*", *Journal-BAR International Series*, Volume 757

[6] W. John, "*Reconstructing History with Computer Graphics*," *IEEE Comput. Graph. Appl.*, vol. 11, pp. 18-20, 1991.

[7] J. Y. Zheng and Z. Zhong Li, "*Virtual recovery of excavated relics*," *Computer Graphics and Applications, IEEE*, vol. 19, pp. 6-11.

[8] Soroush, Mehrnoush, Alireza Mehrtash, Emad Khazraee, and Jason A. Ur. *"Deep learning in archaeological remote sensing: Automated qanat detection in the kurdistan region of Iraq. Remote Sensing"* 12, no. 3 (2020): 500.

[9] Phelan, Keith, and Daniel Riordan. *"Detection of ringforts from aerial photography using machine learning."* In 2020 31st Irish Signals and Systems Conference (ISSC), pp. 1-6. IEEE, 2020.

[10] Verschoof-van der Vaart, Wouter B., and Karsten Lambers. *"Applying automated object detection in archaeological practice: A case study from the southern Netherlands."* Archaeological Prospection (2021).

[11] Barceló, J. A. "Towards a True Automatic Archaeology. Integrating Technique and Theory," in *Layers of Perception. Advanced Technological Means to Illuminate our Past. Proceedings of the XXXV Computer Applications and Quantitative Methods in Archaeological Conference, edited by A. Poluschny, K. Lambers, and I. Herzog, 413–417. Bonn: Dr. Rudolf Habelt GmbH, 2007.*

[12] Cowgill, G.L. 1967. "Computer Applications in Archaeology." Computers and the Humanities 2 (1): 17- 23.

[13] Raper, J. 2009. "Geographic Information Science." Annual Review of Information Science and Technology 43: 1–117.

[14] J. Y. Zheng and Z. Zhong Li, "*Virtual recovery of excavated relics*," *Computer Graphics and Applications, IEEE*, vol. 19, pp. 6-11.

[15] J C Gardin *Methods for the descriptive analysis of archaegical material* American Antiquity 32 13-30 1967.

[16] Wilcock, J.D., Shennan, S.J., 1975, Computer Analysis of Pottery Shapes. In *Computer Applications in Archaeology* 1975. Edited by S. Laflin. University of Birmingham, England, pp. 98–106.

[17] Kampel, M., Sablatnig R., 2003b, An automated pottery archival and reconstruction system. *Journal of Visualization and Computer Animation*, 14(3): 111–120.