CASE STUDY
MASTER 1 DATA SCIENCE & ARTIFICIAL INTELLIGENCE

Deep learning on animal astragalus bone classification from archaeological data

Minh Nhat Do, Franz Franco Gallo*

Manon Vuillien, Vannalisa Coli, Lionel Gourichon†

Université Côte d'Azur, CNRS, CEPAM, 06300 Nice, France

March 12, 2021

Abstract

With the development of science today, especially Artificial Intelligence, people today can apply it to many different fields. Archeology is also one of the fields with the potential to apply artificial intelligence. As we all know, archaeological research is a very specific discipline in its approach to specimens as well as data. The methods used by archaeologists to identify animals from bones recovered from archaeological sites are based on a combination of several anatomical and bone measurement criteria. However, many species with very similar morphological characteristics limit the identification of animals at the species level, especially when bones are not well preserved, broken, or fragmented. Therefore, this potential study proposes the development of a new methodological approach based on Machine Learning methods to distinguish morphologically closely related species by specific skeleton parts: astragalus.

Keywords
Astragalus, transfer learning, Deep learning, bone classification, Archeology

1 Introduction

write something to introduce CNRS, CEPAM... some background about the astragalus bone

*Master student, Data Science & Artificial Intelligence , Université Côte d'Azur

†Post-doctorante, Université Côte d'Azur , CEPAM (UMR 7264), Nice

2 Materials & Methods

2.1 Construct 3D data

To build 3D model, our mentor selected bones come from the osteological reference collections of the Muséum national d'Histoire naturelle (MnHn) in Paris, from Daniel Helmer's reference collection preserved in Jalès in Ardèche, a branch of the ArchéOrient laboratory (UMR 5133, CNRS), and the collections of the Cultures et Environnements, Préhistoire, Antiquité et Moyen-Âge laboratory (CEPAM, UMR 7264, CNRS) in Nice, complemented by specimens of goats collected by Ms.Vuillien as part of her doctoral thesis in prehistory[1].

The bones were modelled in 3D using a surface scanner made accessible to the CEPAM laboratory (technical platform "2D and 3D surveys, modelling and archiving of data") by following an acquisition protocol developed in the framework of Vuillien's doctoral thesis.

2.2 Extract image data

In this work the initial data is 45 3D models of the astragalus bone, which need to predict between 4 classes: goat, mouflon, rupicapra and sheep. However, the size of this data is not enough to train a model using deep learning methods, that is why we need to generate more data from these 3D models. Our first natural approach to generate this data was made by taking the 6 main views of the 3D model: front, left, right, top, bottom and back of each model, that are rendered manually from the MeshLab program, obtaining a total of 270 images which we call dataset1.

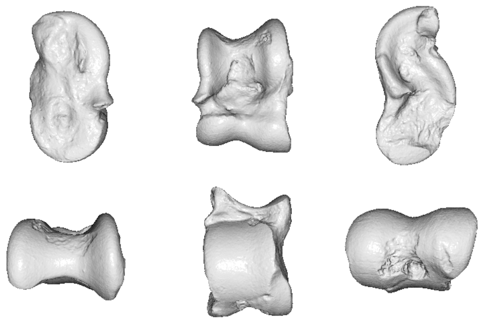


Figure 1: Example dataset1.

However this is very tedious task to generate more images from different angles. That is why in our next approach we implement a script using python and the open3D library that allows us to manipulate the 3D model as a triangular mesh and calculate the normals vertex. From this script we were able to rotate the model on the +x axis in 5 units of measure in the Open3D library, which are approximately equivalent to 20 sexagesimal degrees for each rotation. We select 4 orientations of the model and for each orientation on average we obtain 18 images which makes a total of 72 images per model, and for the 45 3D models in total we generate a data of 3240 images in RGB format which we call dataset2.

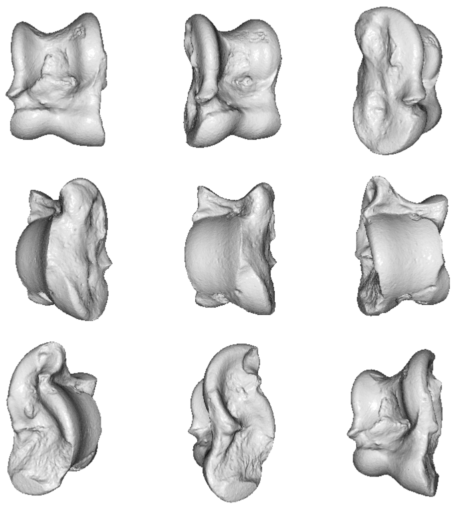


Figure 2: Example dataset2.

However we notice that the data generated by the rotation is not homogeneous in each class, so we need to somehow represent more information from the 3D model in a single image input. In the following dataset we only use one pattern of rotation of the model with which we take the views in the sagital, coronal and axial planes for each point of rotation, these images are then converted into grayscale format and merged them in a single image as pseudo-RGB format, in this way we managed to unify the information of 3 views of the 3D model in a single image for the input in the deep neural network. The rotation is given in the + x axis with an angle of 20 sexagesimal degrees with which we obtain between 14-18 images per model, which finally gives us 722 RGB images in the dataset3.

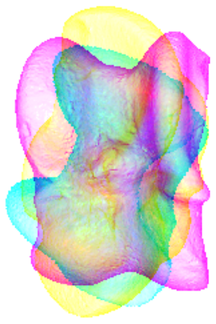


Figure 3: Example dataset3.

2.3 Data processing

Minh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh nhat FranzzMinh

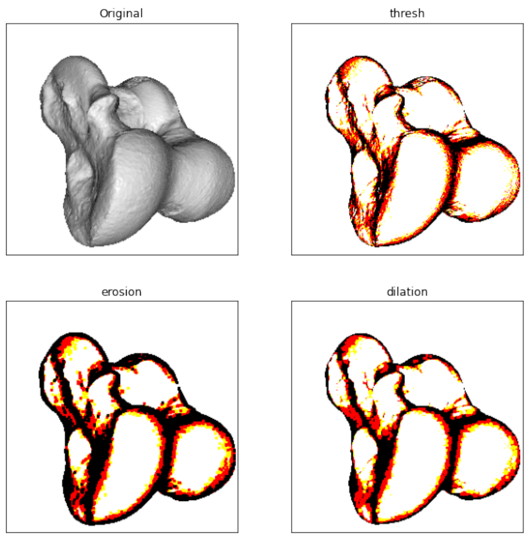


Figure 4: Basic residual block.

2.4 Data augmentation

Data Augmentation is an image generating technique that focuses on providing operations commonly used in the creation of image data for machine learning problems. It is the simplest technique by simply processing existing data by linear or nonlinear operations (such as creating data over a GAN network). The use of data augmentation will help the training data-set increase quickly. In biomedical, the implementation of imaging with original images usually does not affect too much in training the model to predict and classify images. The modification retains the outstanding features of the original data. In principle, Augmentor includes several layers for standard image manipulation functions, such as the Rotate layer or the Crop layer. Users interact with and utilize these classes using a large number of nifty functions, covering most of the functionality you might require when increasing image datasets for machine learning problems.

2.5 Transfer learning

With the lack of clarity and qualify in the features of the 2d images extracted from the 3d model and the amount of data, we needed a model that was sufficiently complex and had proven effective in image classification across some typical data sets. That is also the reason we choose Transfer learning method to build predictive models.

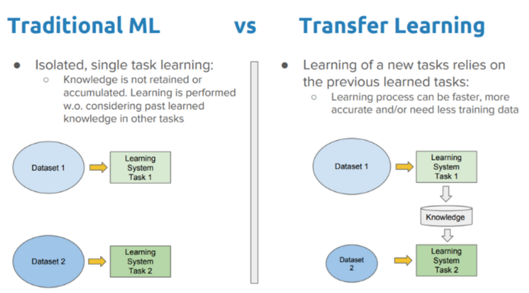

Traditional ML
vs
Transfer Learning
Isolated, single task learning:
Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
Learning of a new tasks relies on the previous learned tasks:
Learning process can be faster, more accurate and/or need less training data
Dataset 1
Learning System Task 1
Dataset 2
Learning System Task 2
Dataset 1
Learning System Task 1
Knowledge
Dataset 2
Learning System Task 2

Figure 5: Traditional & Transfer Learning.

Traditional learning is isolated and occurs purely based on specific tasks, datasets and training separate isolated models on them. No knowledge is retained which can be transferred from one model to another. In transfer learning, you can leverage knowledge (features, weights etc) from previously trained models for training newer models and even tackle problems like having less data for the newer task!. With this case, we used pre-trained model ResNet18 and develop on Pytorch.

2.6 Model structure

Before moving to ResNet18, we will reiterate about the special structural design of the ResNet bearing. When training Deep CNN models (large number of layers, large number of params, etc.) we often have the problem of vanishing gradient or exploding gradient. The fact shows that as the number of layers in the CNN model increases, so does the accuracy of the model. However, when the number of layers is too large ($>=50$ layers), the accuracy is reduced.

Residual block

Residual block was born to solve the above problem, with Residual block, it is completely possible to train CNN models with large size and complexity. without worrying about exploding / vanishing gradients.

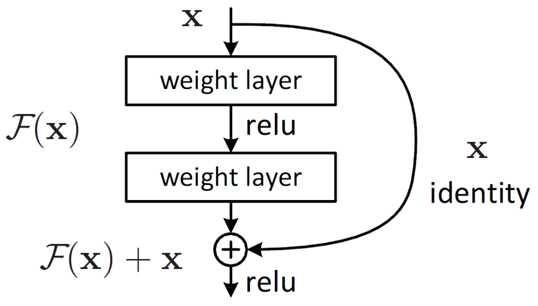

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 6: Basic residual block.

This identity mapping does not have any parameters and is just there to add the output from the previous layer to the layer ahead. The identity mapping is multiplied by a linear projection W to expand the channels of shortcut to match the residual. This allows for the input x and F(x) to be combined as input to the next layer.

$$y = F(x, W_i) + W_s x$$

The key of the Residual block is that after every 2 layers, we add input with output: F (x) + x. The Skip Connections between layers add the outputs from previous layers to the outputs of stacked layers. This results in the ability to train much deeper networks than what was previously possible.

ResNet18 Model

Resnet is a CNN network consisting of many small residual blocks made up. Deep convolutional neural networks detect objects by learning the features. Theoretically, adding multiple layers to CNN allows it to learn more features and achieve higher accuracy; However, it is not an ideal case in practice. It has been acknowledged that training accuracy tends to reach saturation and then degrades rapidly when more layers are added.

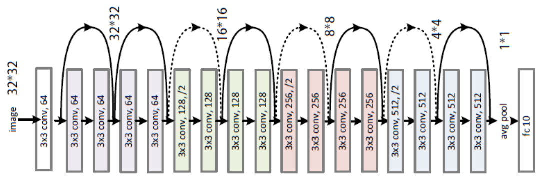


Figure 7: ResNet18 structure.

Model parameters

Total params: 11,341,252
Trainable params: 11,341,252
Non-trainable params: 0

Input size (MB): 0.57
Forward/backward pass size (MB): 62.79
Params size (MB): 43.26
Estimated Total Size (MB): 106.63

3 Results

3.1 Data

After successfully building the model and testing it on previously collected and processed data sets, we have received some positive results in the application of deep learning techniques to classify animal bones. . Below are the results achieved by the project data: From 45 3D bone models of 4 caprinae subfamily animals: sheep, goat, mouflon and chamois., we have extracted and combined:

- 465 - 2D images for train data set
- 176 - 2D images for validation data set
- 97 - 2D images for test data set

Our input data have size 256x256 pixels with RGB color mode. To train model, we used some transforms when loading data with DataLoader of torch library and resize its to 224x224 pixels to fit with default size of pre-trained model.For normalization: $Z = \frac{x-\mu}{\sigma}$ and here we use average value and standard deviation of Imagenet dataset.

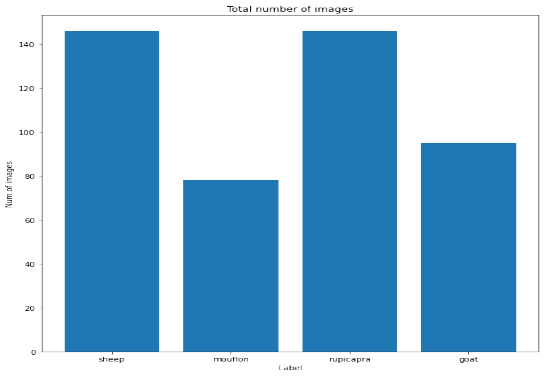


Figure 8: Amount of training data

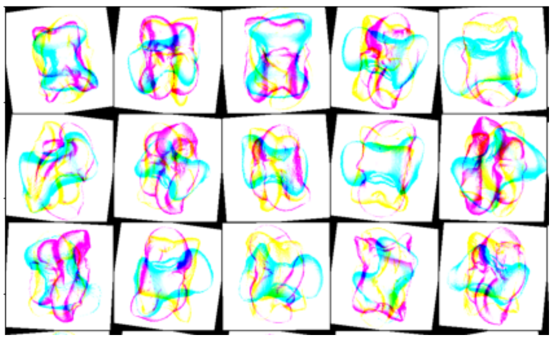


Figure 9: Example input data

That pictures are some example image in Dataloader from training data.

3.2 Training process

Optimization function

Similar to the optimal function Adadelta and RMSprop, Adam function maintains the mean of the past slope of the slope vt and also the mean of the past slope mt, similar to momentum. - While momentum is like a ball that plunges downhill, Adam is like a very heavy ball and has friction, so it easily surpasses the local minimum and reaches the optimal point (flat minimum).)

In training process, I finetuned hyperparameters on :

- *lr* – learning rate (values from 0.001 to 0.0001)
- *betas* – coefficients used for computing running averages of gradient and its square
- *eps* – term added to the denominator to improve numerical stability
- $weight_decay$ (float, optional) – weight decay (L2 penalty)

Loss function

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.

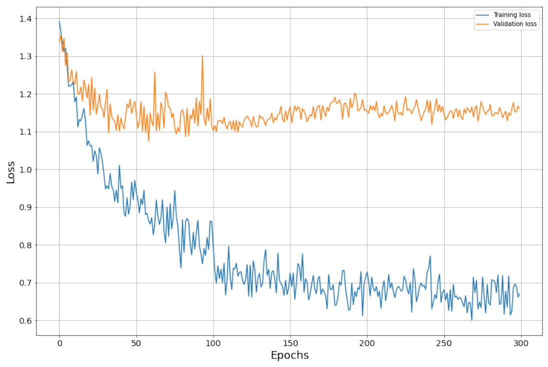


Figure 10: Loss values in training process

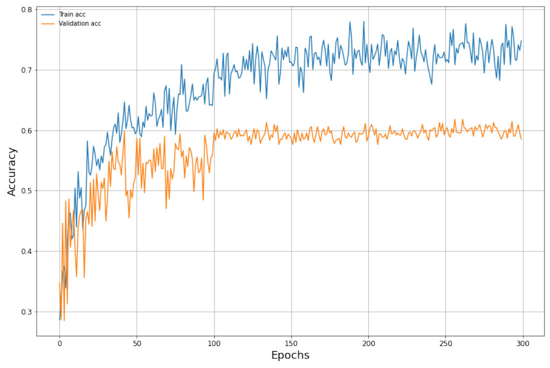


Figure 11: Accuracy values in training process

3.3 Validation model

After training model, we will validate model on test data set.

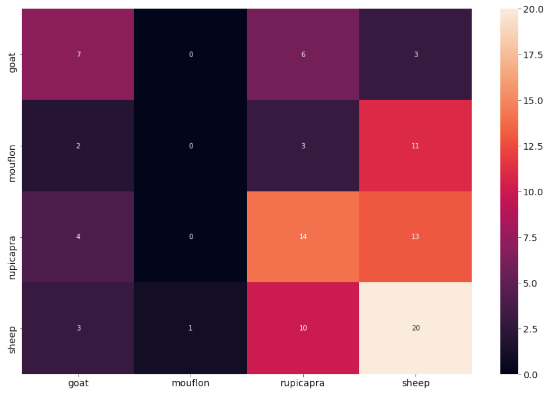
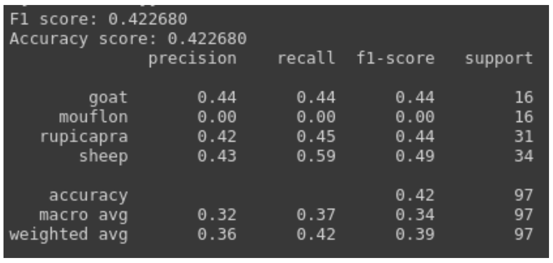


Figure 12: Accuracy values in training process

We can see in this plot, with mouflon, neither label confuse with mouflon label. However, mouflon label confused with other labels. this is an important thing to improve the result in the future.



```
F1 score: 0.422680
Accuracy score: 0.422680
              precision    recall  f1-score   support

        goat       0.44      0.44      0.44        16
     mouflon       0.00      0.00      0.00        16
   rupicapra       0.42      0.45      0.44        31
       sheep       0.43      0.59      0.49        34

    accuracy                           0.42        97
   macro avg       0.32      0.37      0.34        97
weighted avg       0.36      0.42      0.39        97
```

Figure 13: Classification report in test data

With only 45 3-D models, we got a potential result on test data set. We got accuracy 0.42 in all test data includes 97 images. In this report, sheep is the label with the best predictive outcome parameters.

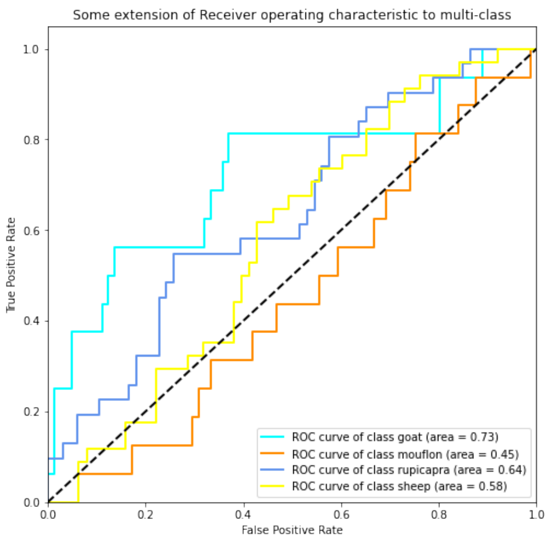


Figure 14: AUC-ROC curve on test data

To have a clear in result of validation, we will look at AUC-ROC curve plot. The AUC value of class goat and rupicabra are 0.73 and 0.64, it means there is a 70% and 64% chance that the model will be able to distinguish between positive class and negative class with sheep and rupicabra. The results on goat and mouflon are not good. If we remember in section data, we know that goat and mouflon are the labels with the least number of images in the training data set. So I think to improve the classification result, we need to balance training data set.

3.4 Test on data

We used random method to get images from test data and make prediction. Below is the result we got.

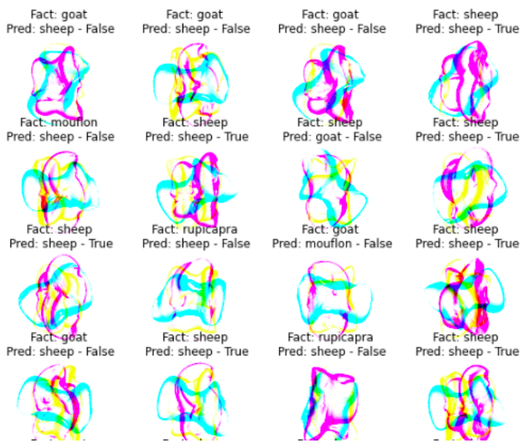


Figure 15: Result in test data

4 Discussion

We will discuss on why we have this result, what are reasons make it bad?

Conclusions

Archeology determines certain parameters to differentiate the astragalus bone in its four categories goat, mouflon, rupicapra and sheep. In this work we managed to take samples of 3D models with images in pseudo-RGB format that express enough information to train a model with deep learning. and obtain good accuracy considering only the 45 3D models we have available for the astragalus bone.

Through sampling with 3 fixed cameras implemented in python, we obtain images that represent the 3D model in a good way, providing information about the bone shape, which is used by joining these views in a single 3-layer image that simulates an pseudo-RGB image but containing the structural information of the bone for each angle of rotation. With this data we were able to obtain accuracy of xx.xx in the training data and xx.xx in the test data.

While it is true we achieve good results, however we believe that these can be improved if we train with 3D models, using libraries such as pytorch3D, Open3DML among others that allow us to manipulate 3D data by implementing tensors and integrate them with deep learning to predict the classes. This can be possible for a future work if more 3D models are available.

Acknowledgements

Anyone to thank/credit for helping your team along the way? This is the place to do it!

References

[1] Vuillien, M. 2020, Systèmes d'élevage et pastoralisme en Provence et dans les Alpes méridionales : nouvelles approches en archéozoologie, Thèse de doctorat en Préhistoire, Université Côte d'Azur, p. 687

[2] Here's another. Write these out as you would references anywhere else. The key I've assigned to this reference is 'other_ref' - therefore, a cite command in the body of your paper with 'other_ref' entered as the sole argument will automatically insert the appropriate reference number.