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 (UMR 7264), 06300 Nice, France

March 19, 2021

Abstract

In recent years, deep learning has achieved great success in many fields, such as computer vision and natural language processing. However, this kind of techniques hasn't been widely explored in archaeology and specifically, in the classification of animals from bones collected from archaeological sites. As we know, archaeological research is a very specific discipline in its approach to specimens as well as data. The methods used by archaeozoologists 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 Deep Learning methods to distinguish morphologically closely related species by specific skeleton parts: Astragalus. Transfern Learning is used with a Resnet18 arquitecture to train 2D images rendered from a 3D astragalus model, achieving accuracy of 0.6 for the classification between 4 classes of the bone.

Keywords
Astragalus, Transfer learning, Deep learning, Classification, Archeology

1 Introduction

Arch-AI-Story is an interdisciplinary research project (coordinated by Isabelle Thery-Parisot, University Côte d'Azur, CEPAM CNRS UMR 7264), whose objective is to investigate the potential application of Artificial Intelligence and Applied Mathematics to Archaeological and History researches. Along with the interest in learning and researching about applied algorithms and artificial intelligence, the first-year students of the Master's program Data Science Artificial intelligence had the opportunity to have access to archeology data from Arch-AI-Story through the case study class.

The initial goal of this project is to apply an artificial intelligence approach to classify the Astragalus bones of four species [1] (Sheep, Goat, Mouflon, Chamois[1]), through the 3-D bone data set provided by researchers from CEPAM. In anatomy, the astragalus is a tarsal bone that participates in flexion-extension movements of the foot. After 2 months of joint work, we proposed 3 approaches to extract 2D image data from 3D bone models and successfully built a deep learning model to classify the bones of the four above species. The result of this project is a good start and also affirms the potential of applying artificial intelligence to support work in the field of archeology.

2 Materials & Methods

2.1 Original 3-D data

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-

[1] In our dataset we refer chamois as a rupicapra label

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 [2].

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 Data Generation

In this work the initial data is 45 3D models of the astragalus bone, which have 4 classes: goat, mouflon, chamois and sheep. 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, these are rendered manually from the MeshLab [3] program, obtaining a total of 270 images which we call *dataset1* (Figure 1).

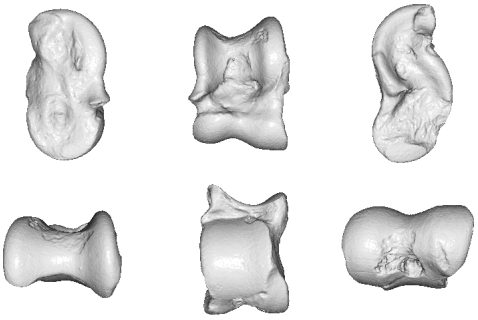

Figure 1: Sample dataset1.

Sampling manually is a hard task and time consuming. With the objective to generate more images from different angles of the 3D models we look for libraries in python to manage 3D data. Our next approach of generate data is using a script with python [4] and the open3D library [5] that allows us to manipulate the 3D model as a triangular mesh from a .ply file. This triangular mesh does not have normals for faces or vertices and a mesh with no normals does not look like a 3D object. The normal is a vector perpendicular to the tangent plane at any point on the surface of the object, and the vertex normal is the normal computed at the vertex of the triangle mesh on the surface. Vertex normals are used for smooth shading that allow to produce a continuous shading across the surface of the triangular mesh. After computing the vertex normals, we oriented all the models with the anterior view of the astragalus bone facing the observer. This orientation represents the vector orientation (0,0,1) on the 3D space, from this initial orientation we got 3 more orientations with the vectors [(1,0,1),(0,1,1),(1,1,1)]. A camera is placed such that captures the plane (100) in miller indices [6], corresponding to the Anterior view of the astragalus bone surface for each orientation, then the 3D model is rotated circularly 360 degrees around axis (0,0,1), the rotation is set in 18 steps to complete a whole lap. In each step the camera render a 2D RGB image of the 3D model and having 4 orientation for each model we end up with a total of 3240 images in RGB format wich we call *dataset2* (Figure 2).

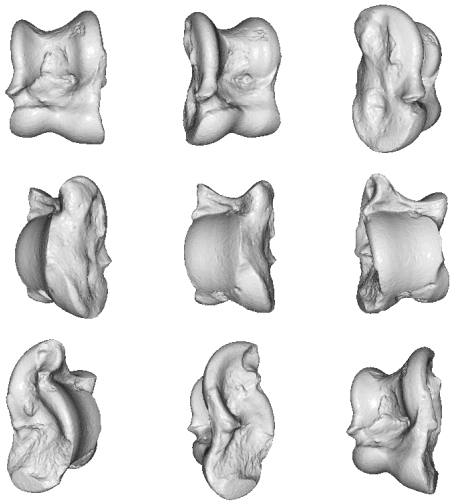

Figure 2: Sample dataset2.

However we realized 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 approach we use the vector orientation (0,0,1) of the 3D model and 3 cameras are placed that captures the planes [(100),(010),(001)] corresponding to Anterior, Lateral and Proximal views of the astragalus bone, then the model is rotated circularly 360 degrees around axis (0,0,1). The rotation is set in 18 steps, but to get a better a results a zoom adjust is done manually on each 3D model before the cameras render the 2D images, this adjust is performed in order to have similar dimensions on all the dataset. After renderization we get between 14-18 RGB images in each camera, these images are first resize in 256x256 format and then converted into grayscale format. Finally we take the 3 grayscale image views for each rotation and merged them in a single image as pseudo-RGB format, in this way we managed to unify the information of 3 views

[(100),(010),(001)] of the 3D model in a single image for multiple rotations, which represents the input in the deep neural network. This approach gives us 722 pseudo-RGB images in total for the *dataset3* (Figure3).

Figure 3: Sample dataset3.

2.3 Data analyzing

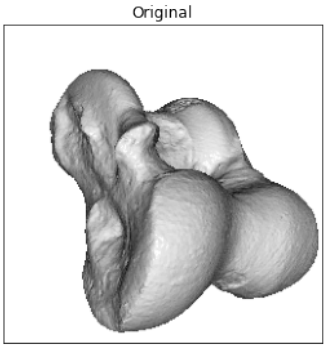

Figure 4: 2-D Image sample from *dataset2*

To understand our data, we will analyzing on 2-D image export from 3-D model(Figure 4). Firstly, we will compute histogram of this example image and plot it to see the distribution off pixels of this image.(Figure 5)

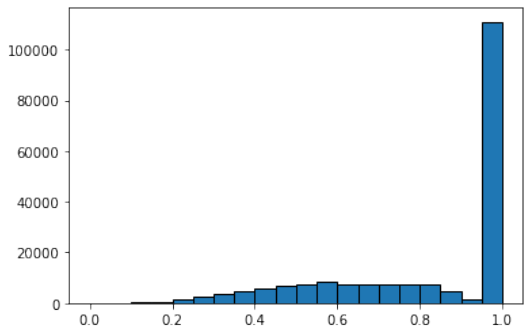

Figure 5: Histogram of pixels on image 2-D.

We get intuition about contrast, brightness, intensity distribution of that image. Left region of histogram shows the amount of darker pixels in image and right region shows the amount of brighter pixels. there is an uneven distribution in shadows and highlights. The hypothesis for this difference may come from the change in brightness, contrast at the edge, and the angle of the 3-D model when we change the view point.Let's check this hypothesis.

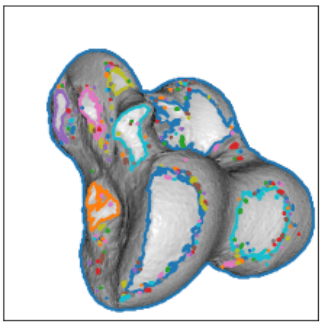

Figure 6: Find contours of example bone image

Note: Contours are defined as the line joining all the points along the boundary of an image that are having the same intensity. Contours come in handy in shape analysis, finding the size of the object of interest, and object detection.

After finding the contours of the image and drawing it on top of the original image, we can change the color shift from light to dark at the edges of the contours based on the change in bone surface.(Figure 6)

So we can use this feature to extract the training image. To strengthen this hypothesis, we will filter the image through a number of filters to separate the areas we are interested in.(Figure 7)

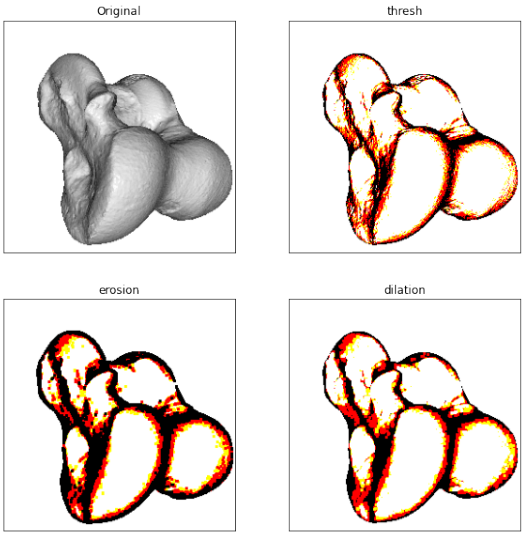

Figure 7: Filter area near contours of bone

With the above results, we can confidently use the dark and light change feature where there is a change in the bone surface to generate data from 3D animal bone models.

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) [11]. The use of data augmentation will help the training data-set increase quickly. The modification retains the outstanding features of the original data.
In principle, Data augmentation includes several standard image manipulation functions, such as the rotate function or the crop function. Users interact with and utilize these functions 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

For the image classification problem, there are many supervised learning models that can be applied. Transfer learning is one of them. This method has been used very successfully in the classification of medical images and especially in the classification of fracture human shoulder bone [12]. Realizing that there are similarities in the dataset, we chose this approach to approach this topic. In addition, given the lack of clarity and qualification in the features of 2-D images extracted from 3-D models and the amount of data, we need a model that is complex enough and has proven effective in classifying images on a number of typical datasets. That is also the reason we choose the transfer learning method to build predictive models.

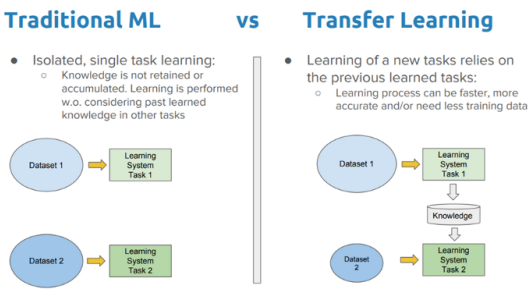

Figure 8: Traditional & Transfer Learning.

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 [13] and develop on Pytorch [7] (a library for deep learning). (Figure 8)

2.6 Model structure

Convolutional Neural Network

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNN have the ability to learn these filters/characteristics.
The architecture of a CNN is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

ResNet and Residual block

Before moving to ResNet18, we will reiterate about the special structural design of the ResNet [14] bearing. When training Deep CNN models (large number of layers, large number of parameters, etc.) we often have the problem of vanishing gradient [15] or exploding gradient [16]. 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 ($\geq$ 50 layers), the accuracy is reduced.

Residual block [10] [14] 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.(Figure 9)

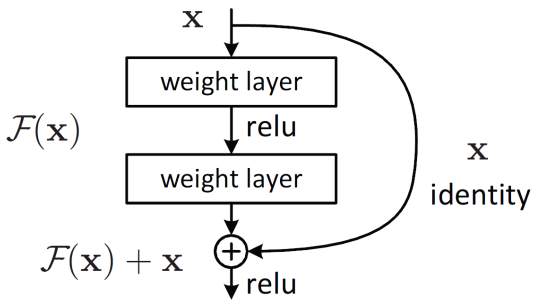

Figure 9: Basic residual block [10]

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: $F(x)+x$. skip connection is a standard module in many convolutional architectures. By using a skip connection, this module will provide an alternative path for the gradient (with backpropagation). It is experimentally validated that this additional paths are often beneficial for the model convergence. Skip connections in deep architectures, as the name suggests, skip some layer in the neural network and feeds the output of one layer as the input to the next layers (instead of only the next one). 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.

ResNet-18 Model

ResNet [14] 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 10)

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2 $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Figure 10: ResNet-18 layers structure [8]

Consider the above diagram, we can see how layers are configured in the ResNet-18 architecture. First there is a convolution layer with 7×7 kernel size and stride 2. After this there is the beginning of the skip connection. The input from here is added to the output that is achieved by 3×3 max pool layer and two convolution layers with kernel size 3×3, 64 kernels each. This was the first residual block.

Then from here, the output of this residual block is added to the output of two convolution layers with kernel size 3×3 and 128 such filters. This constituted the second residual block. Then the third residual block involves the output of the second block through skip connection and the output of two convolution layers with filter size 3×3 and 256 such filters. The fourth and final residual block involves output of third block through skip connections and output of two convolution layers with same filter size of 3×3 and 512 such filters.

Finally, average pooling is applied on the output of the final residual block and received feature map is given to the fully connected layers followed by softmax function to receive the final output. The output of each layer is shown in the diagram and input is changed in the skip connections according to that.

To complete our model, we replaced the last fully connected layer by a new fully connected layer which outputs 4 categories which tells the probability of the image being Sheep, Goat, Chamois or Mouflon.

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

To compute input size, 4 bytes/number (float on cuda). The unit of this part is megabyte (MB - is 10^6 bytes). Input size is total size of each input image we trained this model. Forward/backward pass size is the sum of all output sizes from each layer in our model. Params size is total size of all parameters of this model. Estimated Total Size is the sum of Input size, Forward/backward pass size, and Params size.

3 Results

3.1 Data collection

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:(Figure 11)

- 2000 - 2D images for train data set
- 435 - 2D images for validation data set
- 280 - 2D images for test data set

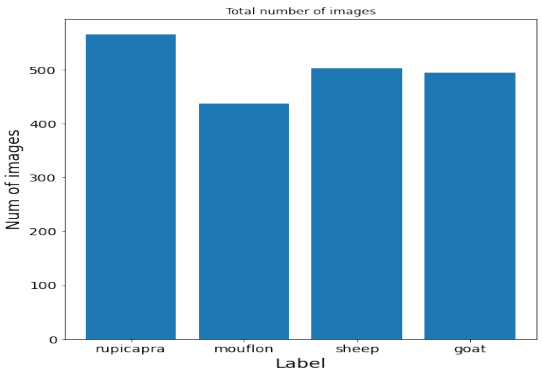

Figure 11: Amount of training data

Our input data have size 256×256 pixels with RGB color mode. To train model, we used some transforms when loading data with DataLoader of torch library and resize its to 224×224 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 which was trained by pre-trained model.

3.2 Training process

Experimental environment

We build model with Python programming language. The following is the execution environment of the model building and testing.

- Python -v3.8
- Pytorch -v1.8.0
- torchvision -v0.9.0
- OpenCV 4.5.1
- Open3D 0.11.2
- Pillow 8.1.2
- CUDA 11.2 + cuDNN -v7 (optional)
- Ubuntu -v20.04 + Window 10
- MeshLab
- Sublime Text + Colab + Jupyter

Optimization function

Adaptive Moment Estimation (Adam) [9] is a optimizer method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients vt and also keeps an exponentially decaying average of past gradient mt. - 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). We compute the decaying averages of past and past squared gradients m_t and v_t respectively as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Computing bias-corrected first and second moment estimates:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

and Adam update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

In training process, the following hyperparameters are tuned :

- 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* – 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 12,13)

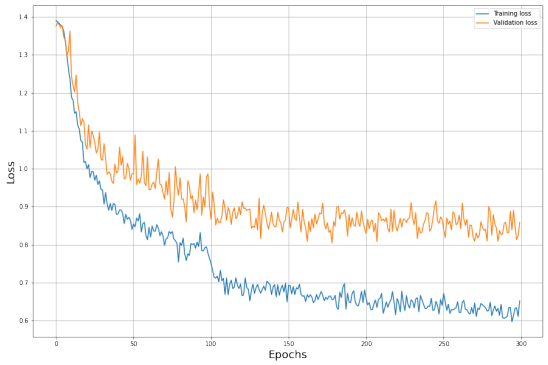

Figure 12: Loss values in training process

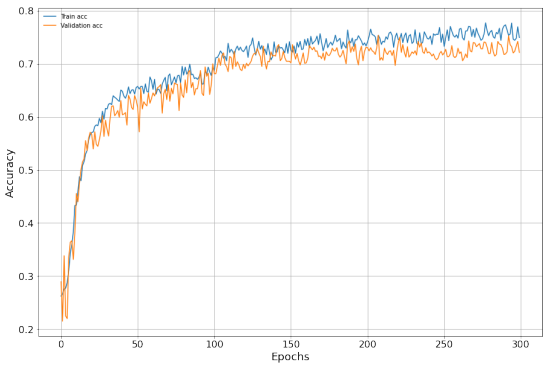

Figure 13: Accuracy values in training process

With close observation and monitoring during the model training, we made timely adjustments to the metadata for the optimizer function during 300 epochs. Specifically, at the 100^{th}, and 175^{th} epoch, we found the model has the signs of slowing down in learning. Therefore, we have reduced the learning rate parameter and increased the penalty constant L2. Raising the penalty constant will avoid overfitting the model when reducing the learning rate parameter.

3.3 Validation model

After training model, we will validate model on test data set. We will reiterate the definition of a number of factors that evaluate a classification model.

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Figure 14: Confusion matrix table

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

F1 score - F1 Score is the weighted average of Precision and Recall.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

F1 score: 0.596429
Accuracy score: 0.596429

	precision	recall	f1-score	support
goat	0.75	0.50	0.60	60
mouflon	0.76	0.39	0.51	57
rupicapra	0.56	0.78	0.65	80
sheep	0.53	0.64	0.58	83
accuracy			0.60	280
macro avg	0.65	0.57	0.59	280
weighted avg	0.63	0.60	0.59	280

Figure 15: Classification report in test dataset

Figure 16: Heat map of confusion matrix

Looking at the Heatmap of confusion matrix(Figure 16), we can observe the distribution of predictions on test data. With this plot, we will identify labels that are often confusing to give correct judgment in updating the model as well as the data.

With only 45 3-D models, we got a potential result on test data set. We got accuracy 0.600 in all test data includes 280 images. With weighted average, our result on precision is 0.63, F1 score is 0.59 and recall is 0.6. To check and visualize the performance of the multi-class classification problem, we used the AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve. It is one of the most important evaluation metrics for checking any classification model's performance.

AUC - ROC [17] curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting False as False and True as True. The ROC curve is plotted with TPR (True Positive Rate) against the FPR (False Positive Rate) where TPR is on the y-axis and FPR is on the x-axis.

Defining terms used in AUC-ROC Curve:

$$TPR = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$FPR = \frac{FalsePositive}{TrueNegative + FalsePositive}$$

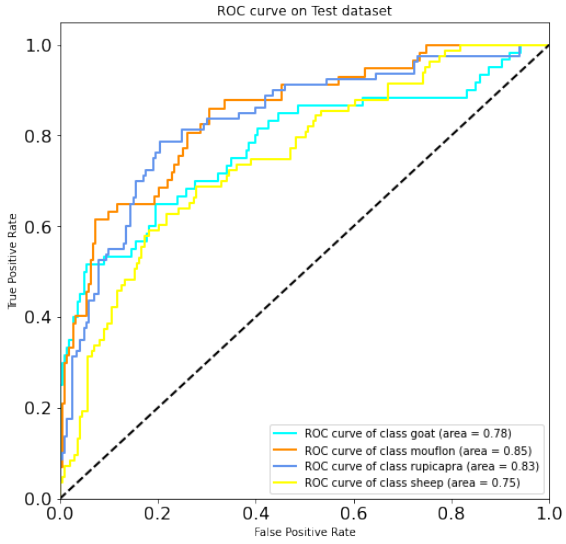

Figure 17: AUC-ROC curve on test dataset

The AUC value of class mouflon and chamois are 0.82 and 0.83, it means there is a 82% and 83% chance that the model will be able to distinguish between positive class and negative class with sheep and chamois. General view on this result, the AUC score looks like good for the starting step of this project.(Figure 17)

3.4 Summary result

Model result			
Dataset name	Train accuracy	Validation accuracy	Test accuracy
Dataset 1	0.788	0.449	0.45
Dataset 2	0.865	0.667	0.40
Dataset 3	0.860	0.724	0.5964

Table 1: Table comparing result from each datasets.

With datasets extracted from 3 approachs to extract 2D from 3D models, we can see the last approach have the best result. So we will look at the detail result on the 3^{rd} approach with RGB type image. With a test dataset without generated data from data augmentation, we got 60% accuracy and this is a good signal for this new approach.

3.5 Result on fact image

A random sample from the images from test data and the corresponding prediction.(Figure 18)

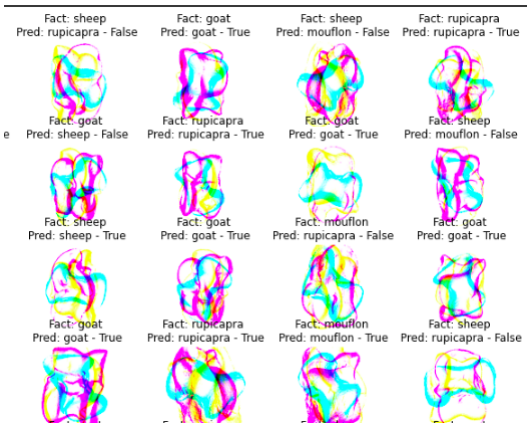

Figure 18: AUC-ROC curve on test dataset

4 Discussion

The generation of data from 3D models is a very important and careful task to have good results. As seen in this process, we managed to have a more homogeneous data in *dataset3*, in which it

was also possible to merge the information from the Anterior, Lateral and Proximal views in a single RGB image considering each layer of the RGB format a view of the astragalus bone. In addition, the orientation of the bone and laterality that it presents must be taken into account, in our work the default orientation that was chosen for all models was having the Anterior part of the bone in the frontal view, and the right laterality.

With a completely new 3D image data approach, we have created our own way to extract training data based on RGB image principles. The 3-layer cut in 3 spatial dimensions and then combined together on the same 2D image with 3 main color systems specific to RBG color mode. It can be said that this is a rather daring and risky step because this is not a common way to apply it to 2-dimensional image classification problems. The initial approach with this technique is quite successful, however, there are still some incomplete points leading to the result is not really as good as our expectation. There are a number of reasons we can point out as follows.

First, we had the 3 layers put together inadvertently change some of the pixels that were important for the classification. Therefore, there is confusion between species. Therefore for this method to be more successful, it is necessary to evaluate the image and filter the important areas more carefully.

Second, the model we build is of high complexity, so it is necessary to carefully and planned hyper parameters fine-tuning. So to improve model accuracy, building a principle to find the best parameters is essential.

Finally, balancing the data between the layers in the data set will help improve the model. Currently, our data between classes is not very well balanced. This can also be a cause for the discrepancy between the predicted results of each label. Besides, we need to have more 3D data to extract more training images. The addition of data will help us confirm the accuracy and reliability of the built model.

Conclusions

Archeology uses certain parameters to differentiate the astragalus bone of four Caprinae species: goat, mouflon, chamois and sheep. In this work we managed to take to take from 3d models samples of 2D 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 a 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 86.0% in the training data, 72.4 in the validation data, and 59.6% in the test data.

While we achieve good results, however we truly 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.

We believe the approach we have proposed in this project will also be a good way to solve this classification problem. To increase the performance of this approach, we needed time to experiment on many different datasets and figure out how to fine-tune the filters and parameters in 2-D image extraction. In addition, we will also test this dataset with some of the more complex deep learning networks.

References

[1] Barone R. 1976. Anatomie comparée des mammifères domestiques. Ostéologie. Paris, Vigot, deuxième édition revue et augmentée, 428 p.

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

[3] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, MeshLab: an Open-Source Mesh Processing Tool Sixth Eurographics Italian Chapter Conference, page 129-136, 2008

[4] G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995

[5] Qian-Yi Zhou and Jaesik Park and Vladlen Koltun, Open3D: A Modern Library for 3D Data Processing, arXiv:1801.09847, 2018

[6] Ashcroft, N. W., Mermin, N. D. (1976). Solid State Physics. Cengage Learning.

[7] Paszke, Adam and Gross, Sam and Chintala, Soumith and Chanan, Gregory and Yang, Edward and DeVito, Zachary and Lin, Zeming and Desmaison, Alban and Antiga, Luca and Lerer, Adam, Automatic differentiation in PyTorch, 2017

[8] Napoletano, Paolo Piccoli, Flavio Schettini, Raimondo. (2018). Anomaly Detection in Nanofibrous Materials by CNN-Based Self-Similarity. Sensors (Basel, Switzerland). 18. 10.3390/s18010209.

[9] Kingma, D. P., Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. International Conference on Learning Representations, 1–13.

[10] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[11] P. Andreini, S. Bonechi, M. Bianchini, A. Mecocci, and F. Scarselli, "Image generation by GAN and style transfer for agar plate image segmentation," *Comput. Methods Programs Biomed.*, vol. 184, p. 105268, 2020.

[12] F. Uysal, F. Hardalaç, O. Peker, T. Tolunay, and N. Tokgöz, "Classification of fracture and normal shoulder bone x-ray images using ensemble and transfer learning with deep learning models based on convolutional neural networks," *CoRR*, vol. abs/2102.00515, 2021.

[13] X. Yu and S. Wang, "Abnormality diagnosis in mammograms by transfer learning based on resnet18," *Fundam. Informaticae*, vol. 168, no. 2-4, pp. 219–230, 2019.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, IEEE Computer Society, 2016.

[15] Z. Dai and R. Heckel, "Channel normalization in convolutional neural network avoids vanishing gradients," *CoRR*, vol. abs/1907.09539, 2019.

[16] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *CoRR*, vol. abs/1211.5063, 2012.

[17] X. Zhang, X. Li, Y. Feng, and Z. Liu, "The use of ROC and AUC in the validation of objective image fusion evaluation metrics," *Signal Process.*, vol. 115, pp. 38–48, 2015.