Contents

1 Abstract

Digitizing and uniformizing the structure of handwritten fossil catalogues exhibits a great potential for increasing the accuracy of paleontological data analysis by increasing sample sizes. Approximately 90, 000 of such samples reside in the archives of the National Museum of Kenya, and an

ongoing effort is to store this data in a digital format for better accessibility. A previous project utilized a commercial optical character recognition service for automated reading of these catalogues. This generalist handwriting detection model lacked the ability to detect special characters used to denote tooth samples, and could not utilize prior knowledge of the vocabulary that is more likely to be present in the data, leading to loss of information and detection mistakes.

This thesis aims to build a specialist character recognition model to increase the accuracy of the bone or tooth type specifying column of the digitized data by fine-tuning a state-of-the-art optical character recognition model with few-shot transfer learning. This is performed by first finding most accurate recognition models, variants of convolutional neural networks or vision transformers, and most successful transfer learning methods for adapting a model to a new character set. Then, the character recognition accuracy of combinations of these methods are benchmarked using handlabeled image segments from the fossil catalogues. The final aim of this work is to use the best-performing model to obtain an accurate reading of the catalogues of the National Museum of Kenya, and publish the final model to be used by the paleontological community for further digitization efforts.

2 Introduction

The field of paleoecology conducts data analysis on fossil specimens. Such analysis is quite literally started from the ground: after a fossil specimen has been found, it is carefully measured and identified: which bone and species the fragment is from, and how old it is. On site, such information is logged on field slips, small thin sheets of paper with a pre-printed form. The analysis has then been traditionally conducted by collecting such entries, sometimes collected in handwritten tabular catalogues, and running statistical tests on the sample set. With this analysis, facts from distant past, such as climate, habitats and vegetation can be deduced [1]. Syntheses of such results consequently allow us to answer larger questions, such as how ecosystems reacted to climate changes, how mass extinction events came about, and what the living world could be like [11]. Understandably, answering such questions has become ever more pressing.

To find answers to large-scale problems, more sophisticated computational data analysis methods have come about, relying on large datasets. Due to the infeasibility of collecting stacks of fields slips across sometimes multiple continents, specimens residing in archives of institutions have been converted to digital, public databases. One such institution is the National Museum of Kenya that holds a large fraction of data collected from one of the most valuable fossil sites globally, the lake Turkana. The digitization effort was started by using commercial optical character recognition software, combined with heuristical and machine learning approaches, resulting in satisfactory accuracy on conventional handwritten text. However, a large hurdle in the existing approach were the special characters used to denote which teeth each specimen contains. The aim of this work is to digitize these markings accurately.

Specifically, this work uses as input data both scan images of the fossil slips and catalogues, and outputs from the Azure AI Vision software [6]. The existing outputs consist of sentence and word-level readings, along with bounding boxes defining the location of each word or sentence. The main research question is the following:

How, given the input data, can the accuracy of the readings of the tooth markings be improved?

The direct impact of this work is an improved precision of the tooth element entries in the digitized fossil catalogues of the National Museum of Kenya, but the results are applicable to a

wider domain of problems. Intuitively, the results are directly applicable to other fossil archives using similar notation: only a fine-tuning of the models to the new archival data is necessary. For other handwritten archives, the results presented can be used to improve recognition accuracy, especially in cases where the data contains characters other than latin letters or arabic numerals. Additionally, this work presents a potential solution for when the target character set can be expressed with multivariate output data. This could, for instance, be handwriting with occasional underlinings, where the bivariate output could be the letter and a boolean variable for whether the character was underlined.

The rest of this thesis is organized as follows. First, the necessary background theory is presented. For deep neural networks, the following concepts are introduced: the basic network structure, how training is conducted, basic building blocks of character-recognizing network architectures, performance-improving heuristics, and transfer learning. For paleoecology, the background covers foundational ecological laws followed by a brief introduction to methods used in paleoenvironmental reconstruction, especially focusing on inferences from tooth data. As the last background section, the composition of mammal teeth is presented. Second, related work is presented, both on handwritten archive digitization and transfer learning with character-recognizer models. Next, the experimental setup is introduced, covering dataset creation, labeling and data preprocessing, followed by base model and transfer learning method selection. After this, results of the experiments are presented and discussed. Finally, the work is concluded.

3 Fundamentals on paleoecology

3.1 Basics on ecology

3.2 Paleoenvironmental reconstruction

3.3 Composition of mammal teeth

Since geological events tend to erode organic remains the faster the remain decomposes, the hardest materials in the corpse represent largest fractions of fossil datasets. These hard materials include shells, bones and especially teeth, and the last is prominent in fossil data analysis also due to the fact that they encode a diverse set of information on the livelihood of the organism [1]. The identification of the fossil remain is done at the finest resolution possible, preferring taxon information over just identifying the genus, for instance. Finest-resolution information derived from dental fossils are the taxon the tooth is from, and which tooth or teeth are found in the specimen. This section presents the naming and indexing system for mammal teeth commonly used in paleontological datasets, as described by Hillson [3], and some common shorthand versions present in the dataset digitized in this work.

Specimens including more complete fragments of the jaw are described with terminology related to the jaw bones. All mammals share the same bone structure around the mouth: the lower jaw consists of two bones called *mandibles*, joining in the middle, whereas the upper jaw consists of bones called *maxilla* and *premaxilla*, that also form large parts of the face. A common trait across many mammals is also that the permanent teeth erupt in the youth of the animal, replacing the 'milk' or *decidous* teeth. Shorthands commonly used for these terms are 'mand' for mandibles, and capital letter 'D' for the decidous teeth.

The tooth rows of mammals are classified to four classes; *incisor*, *canine*, *premolar* and *molar*

and indexed with a numbering system. Moving from the middle of the tooth row towards the side, there are up to three incisors, used for catching food and denoted with the letter 'i'. Behind them is the canine tooth, used for cutting, and in case of predators, killing. This tooth is denoted with the letter 'c'. Behind the canine are up to four premolars, noted with 'p'. These teeth vary most between taxa in form and function with functions including cutting, holding and chewing food. The teeth at the back of the row are called molars, 'm', and are primarily used for chewing. Molars, like the other tooth types, vary in number between taxa, and are at most three. The numbers are always increasing when moving back in the tooth row, but in the case of missing teeth in a taxon, the numbers do not necessarily start from one: instead, the number is chosen to have teeth with same numbers as alike each other as possible. Thus, a taxon with only two premolars might only have the teeth P3 and P4.

Location of the tooth present in the fossil is described with directional terms specifying the side, jaw and the location on the jaw. The most intuitive are left and right describing the side, where one needs to note that each denotes the side from the viewpoint of the animal, not the observer. Mammal teeth are always symmetrical, thus every tooth always has the equivalent other-jaw counterpart. The distance of a tooth from the throat is described with the terms *distal*, 'far from to the mouth' and *mesial*, 'close to the mouth'. For skeletal bones, the term *proximal*, 'close to the center of the body' is often used instead of 'mesial'. Short-form versions for these terms include capital 'L' or 'Lt' for left, capital 'R' or 'Rt' for right, 'dist.' for distal and 'prox' for proximal. The jaw, upper or lower, has three dominant notation styles: one is to sub- or superscript tooth index numbers, other is to over- or underline tooth markings, and the last style, prominent in digital fossil data, is to set the tooth type letter to upper- or lowercase. In each of these systems, a superscipt, underline, or capital letter denotes upper jaw, and conversely subscript, overline or lowercase letter denotes the lower jaw. An illustration of the mammal tooth system is presented in Figure 1. Terminology with corresponding shorthands are summarized in Table 1 and jaw notation styles in Table 2.
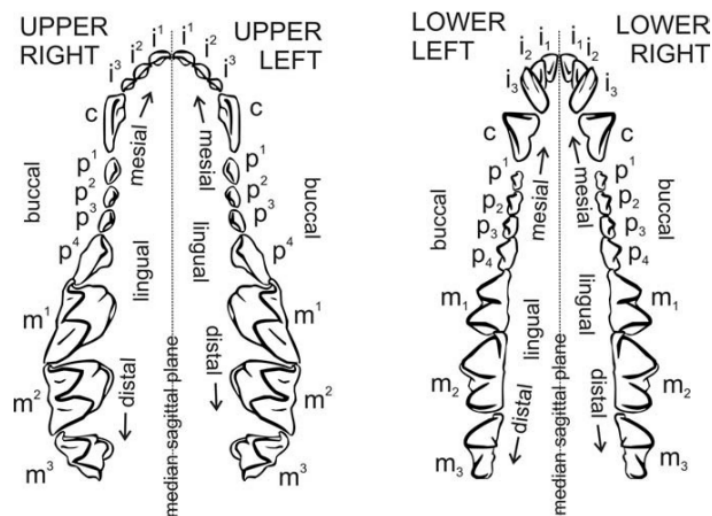

Figure 1: Mammal teeth composition, from Hillson [3].

Term	Meaning	Shorthands
Mandible	Lower jaw bone	mand.
Maxilla, Premaxilla	Upper jaw bones	
Deciduous	'Milk teeth'	D, d
Incisor	Tooth type (front, middle)	I, i
Canine	Tooth type (between incisor and premolar)	C, c
Premolar	Tooth type (between canine and molar)	P, p
Molar	Tooth type (back of tooth row)	M, m
Distal	Far from body center / mouth	dist.
Mesial	Close to the mouth	
Proximal	Close to body center	prox.

Table 1: Terminology related to mammal teeth with corresponding shorthands

Jaw	Line Notation	Sub/Superscript Notation	Digital Notation
Upper	$M^{\underline{1}}$	m^1	M1
Lower	$M_{\overline{1}}$	m_1	m1

Table 2: Dental marking styles, Example: first molar. Line notation displayed in common style combining sub- and superscripts.

4 Deep Neural Networks for Optical Character Recognition

This chapter presents relevant background on deep neural networks (DNN), also known in literature as artificial neural networks (ANN) or, for historical reasons, multilayer perceptrons (MLP). The aspects presented are constrained to those relevant to the problem at hand, optical character recognition (OCR), that is also used as a running example.

4.1 Deep Neural Networks

Neural networks are multivariate functions that share a specific form. The function parameters, usually floating-point numbers, are called weights and are organized in groups called layers. The first layer is called the input layer, after which there are multiple hidden layers, followed by the output layer. Weights of adjacent layers are combined by activation functions, that are constrained to nonlinear functions with scalar inputs and outputs [7]. Simplest of the activation functions is the rectified linear unit ReLU, shown in Equation 1. A neural network is usually visualized with a graph structure, as seen in Figure TODO, where a node represents a weight, and an edge denotes that the value on the first layer is used to compute the value on the latter.

$$\text{ReLU}(x) = \max(0, x) \tag{1}$$

The computation of an output based on an input in the network is called the feed-forward, as the computation runs forward layer by layer through the network. The process starts from the input layer, which is simply the input organized as a vector. Each intermediate value on the first hidden layer, noted h_d below, is computed by taking a linear combination of the layer weight vector

θ and the input vector $\mathbf{x}$ of size N, adding the bias term θ_0, and passing the result through the activation function a:

$$h_d = a \left[\theta_0 + \sum_{i=1}^{N} \theta_{di} x_i \right] \tag{2}$$

Different types of layers, such as convolutional or transformer layers denote that this single-layer computation process is performed differently from the standard form. When many layer types are present, layers using the computation in Equation 2 are called fully connected or dense layers.

The computation proceeds from the first hidden layer in a similar manner: the next layer values, also called activations, are computed using the weights of the layer and the previous layer activations with Equation 2. The activations of the output layer is the output of the network. More complex networks are generally constructed by increasing the network size to up to hundreds of layers with hundreds of millions of parameters, and by using different types of layers.

The universal function approximator theorem states that functions belonging to the neural network family are capable of approximating any mapping from any type or shape of input to any output with arbitrary precision [7]. Naturally, due to high computational costs of finding the optimal weights and the large search space of possible networks, this theoretical optimum is rarely reached.

Examples of input-output mappings relevant in this work are the following, ordered from simplest to most complex: tooth type classification, constrained multilabel classification, and sequence-to-sequence learning. The problem of tooth type classification takes in an input image of a dental marking, such as input in Figure TODO a, and decides which tooth the image denotes. As mammals have up to eleven teeth on each side of two jaws, the classes would be these 44 teeth, such as 'rm1', 'lP4' or 'li2', using the computational notation presented in Section TODO ref mammal teeth. The network would output a 44-element discrete probability distribution obtained by using the softmax function refeq:softmaxTODO, where each value notes the probability that the image contains the tooth this value is chosen to represent. The final output would then be the largest probability found in this vector.

A better approach that could encode the fact that all teeth of same type share a similar input feature , a letter in the image, could be formulating the problem as a multilabel problem [10]: the output would be three of the aforementioned probability vectors, one with four elements representing 'M', 'P', 'C' or 'I', one with four elements for tooth indices, and two two-element vectors for left-right and upper-lower jaw. As this formulation lacks the notion that some tooth-type pairs never exist, such as the 4th canine, this is a case of constrained multiclass classification, where some label pairs are marked as impossible combinations.

Generalizing the mapping problem further, one could also input a variable-length image of the entire dental marking comprising of multiple words, and outputting the text on this image. Due to the variable output length, a special technique called sequence-to-sequence learning is employed [8]. This encodes the fixed-length output layer to variable-length output text. Even though all of the problems presented in this section recognize characters, generally the term 'optical character recognition' is used for this type of mapping. Models for solving these problems accurately are very large, for instance the Microsoft TrOCR has approximately 500,000 parameters [5].

Finding the best network weights for any of these types of input/output mapping problems, training a network, is conducted with the same process. The general recipe for training a neural network is the subject of the next section.

4.2 Training neural networks

After one has defined a neural network structure, initialized the weights of the network to some initial values and obtained a sufficiently large set of input-output pairs from the problem at hand, one can start training the network. Training is an iterative process where inputs are used to predict the output, which is then compared to the ground truth. A more detailed account of these iterations is presented next.

At the start of training, a small fraction, commonly 10-20%, is reserved as test data. Cross-validation is generally not used with neural networks due to the computational cost of the training process (ref?). The remainder, the training set, is divided into batches, sizes of which are commonly exponents of 2.

In an iteration of neural network training, a batch, stacked in a tensor of dimension one larger than the dataset, is passed through the network: outputs are computed based on the training inputs with the feed forward process. After this, a loss function is used to evaluate the quality of the output: the loss function maps the network output and the correct output recorded in the training batch and outputs a scalar value, small value denoting a good match. Loss functions used in optical character recognition are presented in Section 4.2.1. After this follows a step called backpropagation: the gradient of the loss function with respect to the neural network weights is computed. The algorithm used in this computation is called automatic differentiation, and is able to compute gradients with equal computational complexity as the feed forward by utilizing a variant of the chain rule and by proceeding backward in the network [7].

Once the gradient is computed, the next step is to choose how to adjust the network weights based on the gradient information. The simplest approach is, given a predefined step size known as the learning rate, to adjust the weights by the magnitude of the learning rate in the direction of fastest decreasing gradient, a heuristic called gradient descent. Different options for this approach are generally called optimizers and finding a suitable one is a fairly complex problem. Other known optimizers are stochastic gradient descent (SGD), that inserts randomness to the weight-adjusting steps, and Adam (Adaptive Moment Estimation), that uses moments or gradients obtained in previous steps to add smoothness to the movement trajectory [7]. After a weight-adjusting step is completed with the optimizer, the training iteration is completed.

The training process consists of repeatedly performing the aforementioned training iterations. Once the all batches of the whole dataset are used for training, a training step known as epoch is completed. An usual a training process completes dozens or hundreds of epochs. The training is terminated once a predefined stopping condition, such as a number of epochs or a sufficiently small gradient magnitude, is met. The goal of the training process is to find the global minimum of the loss function with respect to the network weights, as this setting would correspond to the optimal approximation of the input-output mapping. Like with optimizers, determining optimal stopping conditions is a difficult problem area within neural network optimization.

After the training is completed, the test dataset laid to the side at the start of training is used to evaluate the predictive power of the network on unseen data, also known as generalization performance. Metrics for measuring the accuracy of a readily-trained model are presented in Section 4.2.2. At this point the model should be considered 'frozen', as adjusting it would optimize the model to the small test set, not the general problem. This pitfall is known as 'data leakage' [4].

As is evident from the generality of the previous description, there are numerous specific aspects to consider when designing highly accurate neural networks. The rest of this chapter presents a snapshot of this problem area, focusing on those relevant to our problem of recognizing handwritten characters, and the rest of the work will experiment with parts of these aspects. A summarizing

pseudocode of the neural network training process with gradient descent is presented in Algorithm 1.

Algorithm 1 Neural Network Training

1: **Input:** training_data, epochs, learning_rate
2: Initialize weights W randomly
3: Initialize biases b randomly
4: **for** epoch $= 1$ to epochs **do**
5:     **for** each (input, target) in training_data **do**
6:         $output \leftarrow$ ForwardPropagation($input, W, b$)
7:         $loss \leftarrow$ CalculateLoss($output, target$)
8:         $(gradients\_W, gradients\_b) \leftarrow$ BackwardPropagation($input, output, target, W, b$)
9:         $W \leftarrow W - learning\_rate \times gradients\_W$
10:         $b \leftarrow b - learning\_rate \times gradients\_b$
11:     **end for**
12:     Print("Epoch:", epoch, "Loss:", loss)
13: **end for**
14: **Output:** W, b ▷ Trained weights and biases

4.2.1   Loss functions

Loss functions are needed within the neural network training process to evaluate the model output quality in each training iteration. These functions map two equally shaped inputs, the predicted and true labels, to a scalar value describing match quality, a low value representing a good match [7]. Thus, for instance, the simple function $f : x, y \rightarrow |x - y|$ would qualify as a loss function. The most commonly used loss functions in optical character recognition are cross-entropy for character classification and the CTC loss for sequence learning.

Loss functions are constructed usin maximum likelihood estimation. When one frames the neural network as outputting a conditional distribution $P(y|X)$, y being the network output and X the input, each correct label in the training set should have a high probability in this distribution. The likelihood is obtained by taking the product of all ground truth label occurence probabilities, and the training goal becomes maximizing this value. Loss functions are derived from the maximum likelihood formalization, so that the network parametrization associated with zero loss is equivalent to the maximum likelihood parametrization. These derivations are out of scope of this work, but can be found in Prince's book section 5.7 [7] for cross-entropy and the original paper [2] for the CTC loss.

The cross-entropy loss function maps pairs of class probability vectors to a loss value. The model output vector describes the probabilities of the input belonging to each of the possible classes and the ground truth vector has the correct class set to one, and all other probabilities to zero. The loss function L is constructed using the Kullback-Leibler divergence between the empirical data distribution $q(y)$, a point mass distribution of the correct labels, and the model output distribution $Pr(y|\theta)$:

$$L(\theta) = \int_{-\infty}^{\infty} q(y) \log[q(y)]dy - \int_{-\infty}^{\infty} q(y) \log[Pr(y|\theta)]dy, \tag{3}$$

8

where the first term is omitted since it has no dependence on the parameters θ. As the distributions are discrete, the loss reduces to

$$L(\theta) = -\sum_{i=1}^{N} \log[Pr(y_i|f(x_i, \theta))], \tag{4}$$

where N denotes dataset size, y_i the correct label, and $f(x_i, \theta)$ is the neural network output.

TODO: sequence ocr models often measure output quality with the CTC (Connectionist Temporal Classification) loss.

4.2.2 Evaluating model performance

Evaluation metrics, numeric values decribing the generalization performance of the trained neural network on unseen data, are used to compare which techniques for implementing neural networks tend to perform better than others. Using standard metrics and benchmark problems, such as the MNIST problem of handwritten digit classification, has allowed comparisons across large bodies of research.

The most common evaluation metrics used when classifying handwritten symbols are accuracy and the F1 score. Accuracy score simply notes the percentage of test images that were classified to the correct class, expressed as a number between zero and one. The main deficiency of using accuracy is brought to light when considering a problem where the probabilities of a class being correct are highly uneven. This is the case in many medical cases: for instance in a cancer screening, one in 500 patients, 0.2 percent, could have cancer. A dummy classifier stating everyone to be healthy would achieve a stellar accuracy of 0.998, but is clearly useless. Any useful model should outperform the dummy, thus comparisons between classifiers would be needed to made between values from 0.998 to 1.00, a rather inconvenient range.

The metrics of precision, recall and F1 are used in these cases. Precision measures the fraction of true positives identified by the model, recall the fraction of true positives out of positives detected by the model. The always-healthy dummy cancer classifier would score zero for both metrics, better describing the uselessness of the approach. The F1 score is the harmonic mean of these scores:

$$F1 = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

For the case of classifying dental markings, the accuracy score is used as the evaluation metric, since the distribution of occurences of different tooth types are not highly uneven. As the same case applies to most other handwritten symbol classification problems, previous work is also compared using the accuracy score.

TODO: ref for this section

9

4.3 Architectures

4.3.1 Convolutional layers

4.3.2 Transformers

4.3.3 Autoencoders

4.4 Techniques and heuristics for improving performance

4.5 Transfer learning

4.5.1 Foundation models

5 Related work

Search strategy: few seed papers and snowball search. Related conferences: Frontiers in Handwriting Recognition

Notes on choices: there is math OCR and music OCR, but they use large datasets and no transfer learning, they dont suffer from the limited target data problem. Also, the problem domain is too different from my problem to be informative.

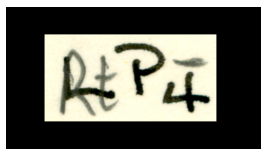## 5.1 Approaches to digitization of handwritten archival data

obvious solution: sit down and type

5.2 Approaches to character recognition with small target domain datasets

6 Experimental setup

6.1 Data description

6.1.1 Notes on creating the dataset

6.1.2 Unicode characters used for data labeling

To label the text found in cropped-out tooth fragment handwriting images, a few nonobvious conventions had to be set in place to construct a labeling system that can be assumed to be easier to learn for a machine learning model. The main guiding rule in these decisions was to encode each feature in the text in one consistent manner. What is meant by features and manners of denoting is explained next.

The unicode system [9] constructs all known characters as signs called graphenes. Each graphene can consist of any number of code points, with each code point having an unique identifier, denoted with "U+code point id". Examples of graphenes with one code point are latin letters, such as 'K',

special characters, such as '@', '%' and '+', or letters from different writing systems, such as 'ω', 'א' or '𝔄'. Examples of multi-code point graphenes are latin letters with accents, such as 'ê', or emoji characters with non-default skin tone, such as 👍. Code points added to the main code point, such as the circumflex accent '◌̂' are called modifiers.

The guiding principle in labeling the data was to encode each concept in the text as one unicode code point. A concept could be, for instance, the number two, or a character being positioned in subscript. The aim of this decision is to allow the model to find common image traits between characters of a similar type: a subscript character has dark pixels in lower positions, and shapes of all number two's have similar curvatures, for instance. As a second principle, it was chosen that each single character in the image, such as "letter C" or "a subscript four with a horizontal top line", would always be labeled as one graphene. These rules makes the encoding choices nonobvious: for example, a subscript number two would intuitively be labeled as the unicode code point '$_2$', but this was not done, since this graphene does not contain the code point for number two, and as a one code point graphene has no code point to extract to be used among the other subscript numbers. Another intuitive choice, '$_2$', would violate the one graphene per character rule.

The special characters in the dental fossil handwriting consist of sub- and superscript numbers, and characters with a horizontal line on top or bottom. Additionally, these two modifiers sometimes co-occur. Both denote which jaw the fragment is from: subscript and horizontal line on top of the character denote lower jaw, whereas superscript or line at the bottom of character signal upper jaw. In a few rare occurences, fractions are present to denote which proportion of the tooth is remaining in the sample. Note that ambiguous notations of for instance subscript number with a horizontal line at the bottom are allowed with this writing system. The labeling notation chosen preserves the option to label these ambiguities.

The following code points were chosen to denote the tooth marking system in the data labels. The base code point modified with unicode modifiers was always chosen to be the latin letter or number present in the character. In the case of fractions, the number in the denominator was chosen as the base code point. The horizontal line on top of a character was denoted with the combining macron modifier (U+0304, eg. Ā), the line at the bottom respectively with the combining macron below (U+0331, eg. A̱). As the unicode system lacks sub- or superscript modifiers, other accent modifiers were used instead. A subscript character was denoted with the combining caron (U+030C, eg. Ǎ), and respectively the superscript with the combining circumflex accent (U+0302, eg. Â). For fraction nominators, a modifier was chosen for each digit present in the dataset (TODO: add here after chosen). These choices were made to improve human readability of the dataset, as the modifier choices are not relevant from the model perspective. A sample of the handlabeled dataset can be found in Figure 3.
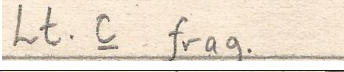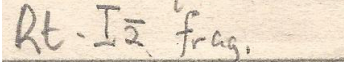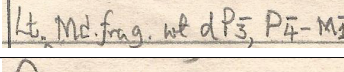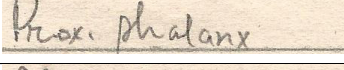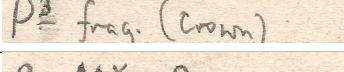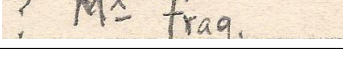
Input Image	Label
Lt. C̲ frag.	Lt. C̲ frag.
Rt. I2̄ frag.	Rt. I2̄ frag.
Lt. Md. frag. wt dP3̌, P4̄-M1̄	Lt. Md. frag. wt dP3̌, P4̌-M1̌
Prox. phalanx	Prox. phalanx
P3̲ frag. (Crown)	P3̲ frag. (Crown)
? Mx̲ frag.	? Mx̂ frag.

Table 3: Samples of input images and their corresponding labels.

6.2 Data preprocessing

6.3 Methods: base models and transfer learning tehniques

6.3.1 Problem formulation

6.3.2 Base model selection

6.3.3 Transfer learning method selection

7 Results and discussion

8 Conclusions

References

[1] J. T. Faith and R. L. Lyman. *Paleozoology and Paleoenvironments: Fundamentals, Assumptions, Techniques.* Cambridge University Press, 2019.

[2] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd international conference on Machine learning.* 2006, pp. 369–376.

[3] S. Hillson. "Tooth Form in Mammals". In: *Teeth.* Cambridge Manuals in Archaeology. Cambridge University Press, 2005, pp. 7–145.

[4] C. Huyen. *Designing machine learning systems.* " O'Reilly Media, Inc.", 2022.

[5] M. Li, T. Lv, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei. *TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models.* 2021. arXiv: `2109.10282 [cs.CL]`.

[6] Microsoft. *Azure AI Vision.* Software available at `https://portal.vision.cognitive.azure.com/`. Version 2024-02-01. Accessed: 2024-02-21.

[7]   S. J. Prince. *Understanding Deep Learning*. The MIT Press, 2023. URL: http://udlbook.com.

[8]   I. Sutskever. "Sequence to Sequence Learning with Neural Networks". In: *arXiv preprint arXiv:1409.3215* (2014).

[9]   The Unicode Consortium. *The Unicode Standard*. https://home.unicode.org/. [Accessed: 2024-09-04]. 2024.

[10] M.-L. Zhang and Z.-H. Zhou. "A Review on Multi-Label Learning Algorithms". In: *IEEE Transactions on Knowledge and Data Engineering* 26.8 (2014), pp. 1819–1837. DOI: 10.1109/TKDE.2013.39.

[11] I. Žliobaitė, M. Fortelius, R. L. Bernor, L. W. van den Hoek Ostende, C. M. Janis, K. Lintulaakso, L. K. Säilä, L. Werdelin, I. Casanovas-Vilar, D. A. Croft, L. J. Flynn, S. S. B. Hopkins, A. Kaakinen, L. Kordos, D. S. Kostopoulos, L. Pandolfi, J. Rowan, A. Tesakov, I. Vislobokova, Z. Zhang, M. Aiglstorfer, D. M. Alba, M. Arnal, P.-O. Antoine, M. Belmaker, M. Bilgin, J.-R. Boisserie, M. R. Borths, S. B. Cooke, J. A. van Dam, E. Delson, J. T. Eronen, D. Fox, A. R. Friscia, M. Furió, I. X. Giaourtsakis, L. Holbrook, J. Hunter, S. López-Torres, J. Ludtke, R. Minwer-Barakat, J. van der Made, B. Mennecart, D. Pushkina, L. Rook, J. Saarinen, J. X. Samuels, W. Sanders, M. T. Silcox, and J. Vepsäläinen. "The NOW Database of Fossil Mammals". In: *Evolution of Cenozoic Land Mammal Faunas and Ecosystems: 25 Years of the NOW Database of Fossil Mammals*. Ed. by I. Casanovas-Vilar, L. W. van den Hoek Ostende, C. M. Janis, and J. Saarinen. Cham: Springer International Publishing, 2023, pp. 33–42. ISBN: 978-3-031-17491-9. DOI: 10.1007/978-3-031-17491-9_3. URL: https://doi.org/10.1007/978-3-031-17491-9_3.