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.

Keywords: Optical character recognition, Transfer learning, Paleontological databases

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 [4]. 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 [37]. 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, holding a large fraction of data collected from one of the most valuable fossil sites globally, the lake Turkana. The sheer amount of physical samples in the museum storage makes keeping them safe a significant challenge, a risk to global heritage now recognized even by the White House and the United Kingdom government, as reported by the Wall Street Journal [19]. A project digitizing the handwritten catalogues hosted by the NMK 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 continue the project by digitizing 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 [16]. 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 [4]. 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 [7], 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.

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

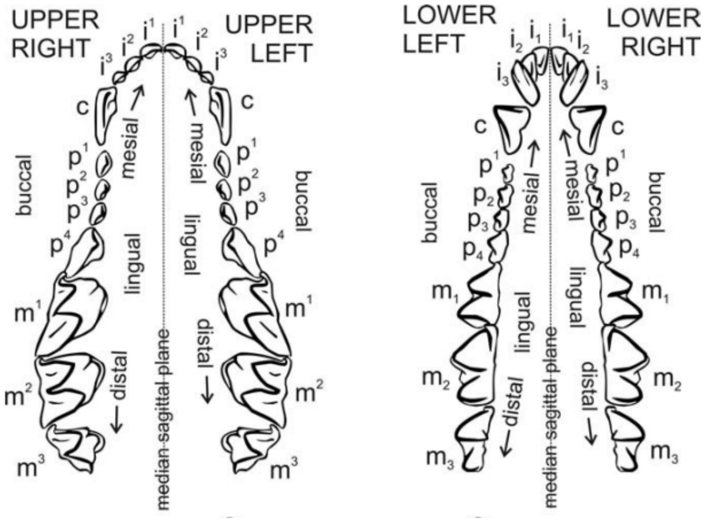


Figure 1: Mammal teeth composition, from Hillson [7].

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 [20]. 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 2, 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.

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

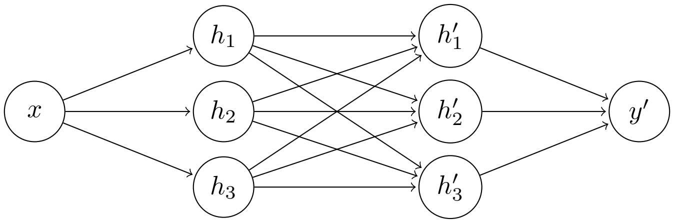


Figure 2: Visualization of a neural network with scalar input and output, and two fully connected layers, redrawn after [20].

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 [20]. 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 3.3. The network would output a 44-element discrete probability distribution obtained by using the softmax function (Equation 3), that takes as input the last layer output values and returns an output layer vector of the same length but with values between zero and one and summing to one, thus a valid probability distribution. Here, like in other classification tasks, 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.

$$\mathrm{softmax}_k[\mathbf{z}] = \frac{\exp[z_k]}{\sum_{k'=1}^{K}\exp[z_{k'}]}, \tag{3}$$

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 [35]: 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 [28]. 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 [12].

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 [20].

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 [20]. 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' [8].

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

```
Input: training_data, epochs, learning_rate
Initialize weights $W$ randomly
Initialize biases $b$ randomly
for epoch = 1 to epochs do
    for each (input, target) in training_data do
        $output \leftarrow$ ForwardPropagation($input, W, b$)
        $loss \leftarrow$ CalculateLoss($output, target$)
        $(gradients\_W, gradients\_b) \leftarrow$ BackwardPropagation($input, output, target, W, b$)
        $W \leftarrow W - learning\_rate \times gradients\_W$
        $b \leftarrow b - learning\_rate \times gradients\_b$
    end for
    Print("Epoch:", epoch, "Loss:", loss)
end for
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 [20]. Thus, for instance, the simple function $f : x, y \to |x - y|$ 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 [20] for cross-entropy and the original paper [5] 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 $Pr(y|\theta)$:

$$L(\theta) = \int_{-\infty}^{\infty} q(y) \log[q(y)]dy - \int_{-\infty}^{\infty} q(y) \log[Pr(y|\theta)]dy, \tag{4}$$

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

$$L(\theta) = -\sum_{i=1}^{N} \log[Pr(y_i|f(x_i, \theta))], \tag{5}$$

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:

$$F1 = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

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

4.3 Architectures

Neural network architectures are alternative ways of constructing the network layer computations for cases where the standard fully connected layer computation presented in Equation 2 is not ideal. Alternative layer types are used to make the model pay more attention to desired aspects [10], such as image structures ignoring the absolute position of these structures in image processing, or character sequences only before a specified position in the sequence in language models.

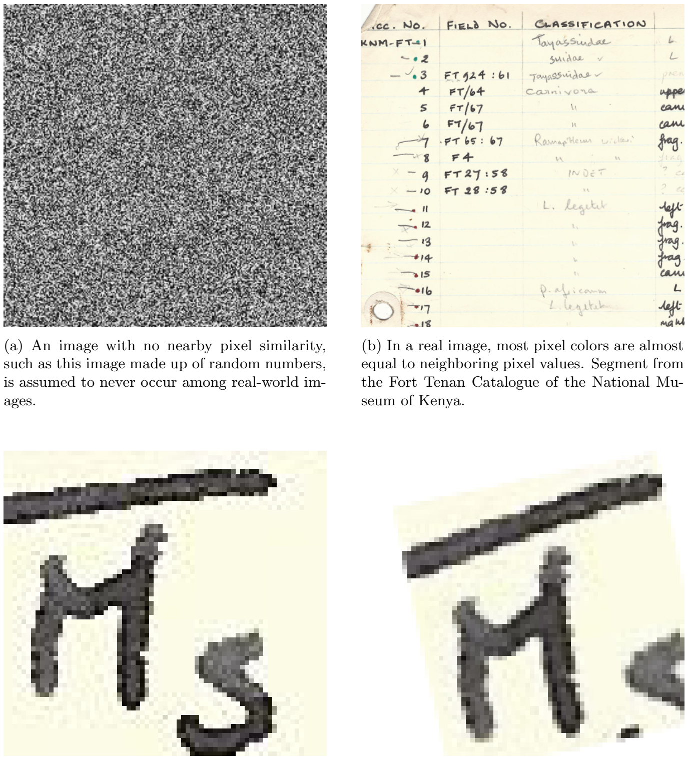
This section presents layer types used in handwritten character classification solutions along with neural networks that first introduced them. The layer types relevant in this work include convolutional layers, autoencoders, and the multi-head self-attention operation used in transformer architectures, and are presented next.

4.3.1 Convolutional layers

The primary motivation for the introduction of convolutional layers was to find a way to encode prior information on images in general to the network architecture: force the network, by adjusting its computational algorithm, to pay attention to particular aspects while ignoring others. Constraining the problem with prior assumptions allows reducing the network parameter count per layer, which frees up computational resources to training further and with more data [10].

The two main pieces of prior knowledge encoded to the convolutional layer computation are invariance to transformations and local relatedness of pixels [20]. Transformation invariance refers to the fact that morphing an image usually keeps the semantic meaning: for instance moving a letter A in an image, rotating it, coloring it red, or squishing it still preserves the fact of it being a letter A. Local relatedness encodes the fact that pixels that are next to each other are much more likely to have same intensity values than any other values, and that the pixels cannot be shuffled without losing meaning. A visualization of these assumptions can be found in Figure 3.

The layer output computation on a convolutional layer is similar to the fully connected layer computation presented in Equation 2, but takes as input only a small region around each pixel instead of the whole input, and uses same weight parameters on all input positions. To achieve



(a) An image with no nearby pixel similarity, such as this image made up of random numbers, is assumed to never occur among real-world images.

(b) In a real image, most pixel colors are almost equal to neighboring pixel values. Segment from the Fort Tenan Catalogue of the National Museum of Kenya.

(c) Transform invariance: the original tooth notation sample...

(d) ... still keeps its meaning of lower third molar after a transform

Figure 3: Visualization of the similarity and transform invariance assumptions encoded in the convolutional layer computation.

this, the weights of the layer only consist of a kernel, a small matrix of weights with uneven-numbered size in both the width and height dimensions [20]. During the computation, the kernel acts as a sliding window, moving through every possible position on the input image. For each kernel position, an output value is produced by computing the dot product between the kernel and the kernel-sized input region around the processed position, and as usual, a bias term is added and the result is passed through the nonlinear activation function. An example of this basic convolution operation on a grayscale image with kernel size 3x3 is given in Equation 7. For a colored image with three color channels, the kernel becomes three-dimensional. An illustration of such case is found in Figure 4.

$$h_{ij} = a\left[\beta + \sum_{m=1}^{3}\sum_{n=1}^{3}\omega_{mn}x_{i+m-2,j+n-2}\right], \tag{7}$$

In implementing a convolutional layer, a few design decisions need to be made: how to handle border pixels, choosing a channel count, and the shape of the convolution kernel [20]. For the bordering pixels, where some kernel pixels fall outside of the input matrix, the options are to skip

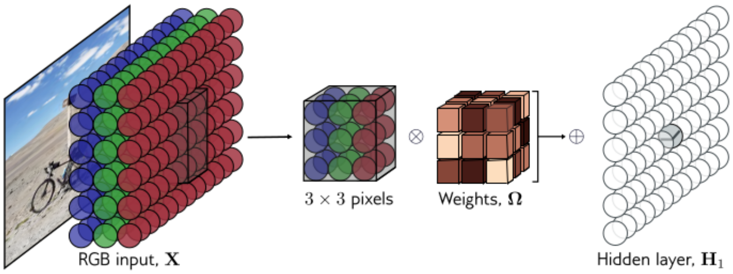


Figure 4: An illustration of a convolution computation in the case of a color image, from [20].

these computations, pad the input with zeros, or to pad the input by mirroring bordering pixels, the main difference of these being that not padding the input creates a result smaller than the input matrix. For the channel count, it has been found that sometimes running multiple convolutions side by side creates better results, and the optimal choice of channel count is application-dependent. The resulting channels are much like the familiar red, green and blue in an image, but do not have any human-decipherable meaning. Lastly, considering the kernel, three aspects need to be chosen: size, stride and dilation. Size refers to, intuitively, the height and width of the kernel. The stride, commonly 1, controls by how many pixels the kernel is slid forward after each activation computation shown in Equation 7. With dilation, the kernel itself is sparse: the weights are interleaved with a desired count of zeros. This allows for larger kernels without increasing parameter counts [20].

Usually, a convolutional layer is followed by a pooling layer. The pooling layer adds more nonlinearity to the network, and allows halving the layer size. The most common operation is the max pooling operation, where each output pixel is the maximum of a 2x2 are in the input. Other, less popular operations include the mean and average pooling operations, where the output naturally is the mean and average of the small area, respectively. These operations allow for the common structure of convolutional networks, where the layer width and height progressively decrease as the computation proceeds [20].

Many advances in convolutional networks have been motivated by the ImageNet image classification competition [24], that ranks models by their capacity of classifying images retrieved from the web to 1,000 distinct classes. The results are presented as top-1 and top-5 error rates, the percentage of samples where the correct class is not among the one or five most likely classes according to the network. Convolutional architectures that have performed well in the competition have been found valuable in handwritten character classification, thus the breakthrough models presented next are experimented with in later parts of this work. These most influential convolutional models in the history of the ImageNet challenge, that only rely on the basic convolution operation, are AlexNet [10], the VGG model family [27], and the Inception architecture used in GoogLeNet [29].

The AlexNet convolutional neural network is perhaps the most influential deep learning model of all time, popularizing deep learning as a superior feature extractor compared to human-performed feature engineering (ref?). The breakthrough was achieved mainly by speeding up the training process by utilizing graphical processing units (GPU) in training, a novel idea at the time, and simplifying the feed forward computation by replacing the previously popular hyperbolic tangent activation with the ReLU function (Equation 1) [10]. This allowed training a larger, deeper network with convolutional kernels up to the size 11x11, leading to nearly halving the best top-5 error rate with the score of 17,0%. Other, more minor but still highly influential innovations were the normalization of convolution results, overlap in the pooling areas, and using dropout regularization, intruduced in Section 4.4.

The ImageNet competition of 2014 saw the next breakthrough innovation with the winning architecture of the GoogLeNet, introducing the idea that smaller convolutional kernels with highly optimized training computation allows deeper, and therefore more capable networks [29]. The layer architecture, named Inception after the 'We need to go deeper' internet meme (see Figure 5), only used kernels of size 3x3 and 5x5, and employed dimension reduction with 1x1 convolutional kernels. They also introduced sparsity by omitting connections on dense layers inspired by knowledge on the structure of biological visual systems. While these ideas were motivated by performance-related reasons, the main aim of the study being to fix multiple-add operations available for training and to reach as high accuracy as possible, the resulting model ended up reaching a new best top-5% error rate of 6,75% in ImageNet classification.

A close runner-up in the 2014 competition, the VGG model family again proved that deep networks with smaller kernels tend to outperform shallow, larger-kernel approaches [27]. The approach of the experiments was to fix all kernels to 3x3 size, and to benchmark classification accuracy between different model depths. The deepest model with 19 layers was found to be most accurate, and additionally to be highly competitive as a feature extractor: an appendix of experiments reached new-best results on a variety of further classification tasks. While the best top-5 accuracy of 7,3% ended up with a second place in the ImageNet 2014 competition, the structural simplicity and diverse learning capacity of the VGG models had a major influence on later deep learning research.

The architectures of the AlexNet, GoogLeNet and VGG models are summarized in Figure 6.

4.3.2 Transformers and the multi-head self-attention

4.3.3 Autoencoders

4.4 Techniques and heuristics for improving performance

4.5 Transfer learning

Much of transfer learning is about what happens before one starts the neural network training process presented in Section 4.2: starting training with a neural network setup, architecture, parameterization and training process configuration already successful at a related task, is known to result in a better model [18]. As an analogue to human learning, teaching a human transcriber to read fossil specimen markings is much easier if the human being has already learned to read.

Transfer learning has the significant benefits in building accurate neural networks of aiding in optimization, generalization and regularization, and saving computational resources and labeling effort ([3], [18]). The experimental section presented by Erhan et al. [3] found a theoretical explanation for these benefits: they experimented with ordering of the training data, and found that early examples have a significant influence on the further learning trajectory, trapping the optimizer in a parameter region that is later very hard, if not impossible, to escape. For this reason, withholding training examples from the actual task until later helps the model not overfitting to the training set, as the initial examples do not trap the optimizer to an undesirable parameter region. Intuitively, as the model in this way also sees more versatile examples related to the problem, it is known that the end result generalizes better to unseen test data [18]. The practical implementation of transfer learning is better understood when one views a neural network as a hierachical feature

YO COBB I HEARD YOU LIKE DREAMS
SO I PUT A DREAM IN YO DREAM IN YO DREAM
SO YOU CAN DREAM WHILE YOU DREAM WHILE YOU DREAM
WE NEED TO GO DEEPER
INCEPTION

Figure 5: The meme that inspired the Inception architecture in GoogLeNet [29], from [9].

extractor: the first layers learn very low-level features of the input, and downstream layers combine this information to learn higher-level concepts. In the letter-recognition case, a low-level feature could be noticing edges, and a higher-level feature that the edges curve and intersect in specific ways. The idea is that the lower level features are common between tasks, and therefore it is beneficial to reuse them by only adjusting the lower layer parameters during training. This approach of not training first layers is also called 'freezing' layers. By freezing parameters the search space for optimal configuration of the remaining parameters is greatly reduced, saving both computational and training data quantity requirements significantly. As data labeling is one of the most laborious aspects of creating deep networks [8], this is a significant benefit. However, even in cases where labeled training data is abundant, pre-training on another dataset has benefits, likely due to the generalization and early example influence aspects discussed above [3].

While the principle of transfer learning was known with many variants and names before, the first survey on the topic [18] collected all these ideas under the term transfer learning and established further terminology related to the paradigm. The first problem solved by the neural network is called the source; the mapping $X \to y$ is termed the *source task*, and the input and output pair distribution $P(y|X)$ is termed the *source domain.* Similarly, the next task solved is called the target, with mapping being the *target task* and the data distribution the *target domain.* A less rigorously defined term for referring to the similarity of the source and the target is in many works

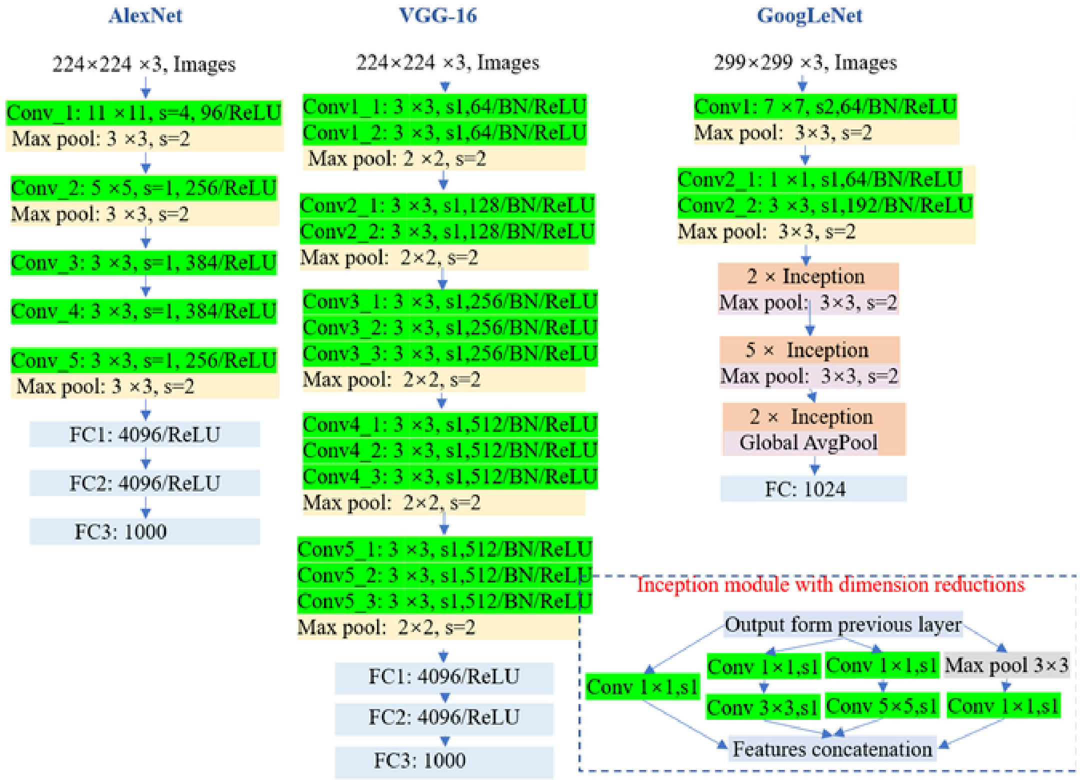

AlexNet
224×224 ×3, Images
Conv_1: 11 ×11, s=4, 96/ReLU
Max pool: 3 ×3, s=2
Conv_2: 5 ×5, s=1, 256/ReLU
Max pool: 3 ×3, s=2
Conv_3: 3 ×3, s=1, 384/ReLU
Conv_4: 3 ×3, s=1, 384/ReLU
Conv_5: 3 ×3, s=1, 256/ReLU
Max pool: 3 ×3, s=2
FC1: 4096/ReLU
FC2: 4096/ReLU
FC3: 1000
VGG-16
224×224 ×3, Images
Conv1_1: 3 ×3, s1,64/BN/ReLU
Conv1_2: 3 ×3, s1,64/BN/ReLU
Max pool: 2 ×2, s=2
Conv2_1: 3 ×3, s1,128/BN/ReLU
Conv2_2: 3 ×3, s1,128/BN/ReLU
Max pool: 2×2, s=2
Conv3_1: 3 ×3, s1,256/BN/ReLU
Conv3_2: 3 ×3, s1,256/BN/ReLU
Conv3_3: 3 ×3, s1,256/BN/ReLU
Max pool: 2×2, s=2
Conv4_1: 3 ×3, s1,512/BN/ReLU
Conv4_2: 3 ×3, s1,512/BN/ReLU
Conv4_3: 3 ×3, s1,512/BN/ReLU
Max pool: 2×2, s=2
Conv5_1: 3 ×3, s1,512/BN/ReLU
Conv5_2: 3 ×3, s1,512/BN/ReLU
Conv5_3: 3 ×3, s1,512/BN/ReLU
Max pool: 2×2, s=2
FC1: 4096/ReLU
FC2: 4096/ReLU
FC3: 1000
GoogLeNet
299×299 ×3, Images
Conv1: 7 ×7, s2,64/BN/ReLU
Max pool: 3×3, s=2
Conv2_1: 1 ×1, s1,64/BN/ReLU
Conv2_2: 3 ×3, s1,192/BN/ReLU
Max pool: 3×3, s=2
2 × Inception
Max pool: 3×3, s=2
5 × Inception
Max pool: 3×3, s=2
2 × Inception
Global AvgPool
FC: 1024
Inception module with dimension reductions
Output form previous layer
Conv 1×1,s1
Conv 1×1,s1
Conv 1×1,s1
Max pool 3×3
Conv 3×3,s1
Conv 5×5,s1
Conv 1×1,s1
Features concatenation

Figure 6: Architectures of AlexNet [10], GoogLeNet [29], and VGG-16 [27], the VGG model variant with 16 layers, from [36].

called *transfer distance*, and will also be used in this work. The subtypes of transfer learning are distinguished by which aspects of the source and the target differ from each other, and by how much. The case where the source and target task are different is called *inductive transfer*, irrespective of if the domains differ. If tasks are the same, but the source and target domain differ, *transductive transfer* is conducted. Within inductive transfer, re-using parameters obtained by training the source task is called *feature representation transfer*, a common approach. Re-using training hyperparameters is called *parameter transfer*. Other terms, not relevant for this work include *instance transfer*, where source domain data is re-used in target training, and *relational knowledge transfer*, where source and target domain are related in a systematic way.

Using the terminology of transfer learning, the problem of reading handwritten tooth notation can be formalized in the following way. The source task is image classification, either of handwritten characters or of images of various objects; the ImageNet task [24]. Which one of these works better will depend on for which one the transfer distance is shorter. This is not entirely obvious in favor of handwriting datasets as they are idealistic benchmark datasets, so they might differ from real world character images more than real-world images of other objects. The target task is detecting the tooth markings from tooth character images, and as the source and target tasks and domains differ, this is a problem of inductive transfer. The experiments are set up in a way that initially, the hyperparameters and parameters used in source task are re-used, thus both feature representation and parameter transfer is conducted. After that, a hyperparameter optimizer is run, thus the next experiments become only feature representation transfer. Should one in the future want to use the model developed in this work for catalogues from different institutions, it would be a good idea to further train the model on catalogues from the new archives. This would be transductive transfer learning, and as the only difference would be a different target domain, the transfer distance is much smaller than in these experiments.

In the literature review and experiments presented above, it is assumed that training from scratch, starting training from a random parameter initialization, in no case works better than transfer learning, a consensus largely shared among the machine learning community [11]. For this reason, research works implementing new character set recognition by training from scratch are omitted from the literature search. Only very few articles were dismissed for this reason.

4.5.1 Foundation models

Foundation models can be viewed as an extreme version of transfer learning: how far one can move in terms of transfer distance, and how versatilely one can use a single pre-trained model, while still reaching adequate target task results. Additionally, as the source task performance is irrelevant from the view of the target task [18], one can construct source tasks that have no practical value in themselves but are favorable in that obtaining labeled data is easy. This is the motivation behind unsupervised and self-supervised pretraining, tasks where the input-output pairs for the source task can be constructed automatically, and having the neural network learn a mapping behind them results in good features. These tasks include the identity mapping $X \rightarrow X$ of autoencoders, masking parts of the image and having the model learn to impaint the missing part, or learning which image pairs are transformed, for instance rotated, versions of one another. Very large models trained with lots of computational resources and then used versatilely are called *foundation models*, as they are thought of encoding foundational, generic knowledge of the data domain. A famous example of a foundational model is the GPT model family by OpenAI, trained by giving the model starts of sentences to continue. Anyone can test transferring the foundational knowledge on text by prompting the open chat variant of the model a new task, such as solving elementary math equations.

In handwritten character recognition, unsupervised pretraining is in some cases used with autoencoders: the model is trained to reconstruct the input image, and these encoder parameters are then used in the target task of classifying the images to characters [**shoponBanglaHandwrittenDigit2016**]. However, this is an approach less established than the supervised pretraining variant.

This chapter presented background on neural network training relevant for the task of classifying handwritten digits. The coarse principles to remember for the following chapters are that a neural network essentially is a function approximator. The different structures are used to aid in this approximation by encoding prior knowledge; the convolution operation encodes these for images, the self-attention for text sequences. Autoencoders are used to conduct unsupervised learning, where output equals the input. Neural network training refers to finding the best parameters to approximate the given mapping, and is conducted iteratively by creating an output, comparing it with the correct output, and adjusting the parameters in the direction of better output producing parameters. Transfer learning means that this process is started with a parameters that is assumed to be closer to the best parametrization than an average random initialization. The following chapters will review previous experiments implementing neural network training with transfer learning to solve the mapping task of outputting correct character given the image of the character, and experiment with the approaches used the previous successful attempts to correctly classify handwritten tooth fossil markings.

5 Related work

This chapter presents related work from two viewpoints. First, in Section 5.1, work on digitizing handwritten fossil catalogues is reviewed. As the work in this area is highly limited, Section 5.2 presents work using similar techniques to ones used here: publications aiming to recognize individual handwritten characters with deep neural networks and transfer learning with limited target domain data.

The literature search was conducted with a snowball search, proceeding with relevant forward and backward citations of a few key articles. Additionally, approximately last five to ten years of conference proceedings most relevant for this problem area were scanned. For related work on fossil digitization, any found work touching on the subject area will be commented on due to their limited amount. For work conducting a similar analysis of handwritten character recognition, more strict conditions were placed. For results, to be considered, the work had to be explicit about the best accuracy score achieved, mentioning the size of the training dataset, number of characters within the classification problem, and the percentage of correct inferences with the test set. The source of the base model had to be given along with its source task, training data and network architecture. The exact method of conducting the model fine-tuning had to be explicit enough to be reproducable, work simply stating they used transfer learning or leaving gaps in the training process were omitted. Additionally, publication venue reputation and acceptable text and presentation quality were taken into account, and some work had to be omitted because of undecipherable method descriptions or a lack of necessary detail.

5.1 Approaches to digitization of handwritten fossil catalogues

While digitizing handwritten fossil catalogues holds, should all state of the art optical character recognition and data management methods be in full use for solving the problem, perhaps a dramatic potential for improving the quality of paleoecological research, surprisingly little has as of now been done in this area. Likely due to the simple fact that the problem has attracted little attention, most current work assumes the most rudimentary method: a human being sitting down, and transcribing.

Due to large amounts of transcription work already done, digital repositories of fossil data do exist. A notable review by Uhen et al. [32] reviews such databases and encourages further digitization of unpublished data, but does not take a stand on how one should complete such work. Another more extensive review, a whole chapter on digitization methods in paleontology has been written [15], but the chapter exclusively considers digitizing physical bone and plant pieces with various 3D imaging methods, and briefly touches on preferring digital journals over paper-printed distribution of research.

For actual, automated handwriting recognition, a few mentions do exist in previous work. The data management themed article by Groom et al. [6] does identify the primary problem of this thesis: the article mentions the fact that most fossil data exists as handwritten records with sufficient information to conduct analysis without the physical sample, and a big advancement in data availability would be to efficiently and in a consistent manner convert this data to a structured database. Optical character recognition is presented, but more as a futuristic option not feasible with current methods, and any data quality suggestions implicitly assume that transcription is completed by a human being. This review is still relevant for automated digitization as many important considerations in data structuring, standardization and quality management are presented - revealing the new challenge that should large-scale automated digitization succeed, the next big line of work would be how to structure and manage the digital paleontological data.

The only found work solving the exact same challenge as this work is the article by Shanmugavel et al. [25]. While the problem is exactly the same, the methods only contain some of the most basic computer vision processes, edge and contour detection, and a complete lack of results reveal that not much was achieved in this project. However, it serves as a proof that such attempts have taken place in previous years.

Concluding the review on research articles about digitizing handwritten fossil data, it seems that no successful projects have as of yet been completed outside of the Data Science student project in spring 2024 for the National Museum of Kenya, for which this work is a continuation. For this reason, the experiments in this thesis will start out with the simplest problem reductions and only present more advanced techniques as ideas for future work. The rationale behind this decision is that since not much has yet been done, it is best to start out from simple and established implementations to verify simple hypotheses before proceeding to more recently developed mehods.

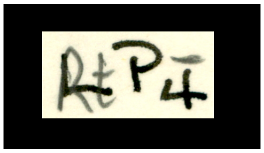
5.2 Approaches to handwritten character recognition with small target domain datasets

Due to the limited amount of work on optical character recognition on fossil catalogues, other problem domains were chosen to draw inspiration from for constructing the model training setting. Interestingly, the nature of the fossil catalogue problem has many aspects in common with reading small, regional Indian languages, many of which are based on the Sanskrit language, and with certain historical scripts: small training datasets, modifiers, cursive writing and varying character sizes. While on first thought, languages where a character is a compound of multiple smaller units, such as the radicals in Chinese or Korean would be an analogous problem, these are not generally solved with transfer learning as the target language has sufficiently large datasets. Therefore, smaller languages such as the studied south Asian languages with less established public datasets are more applicable. Of these smaller languages, at least Gujarati [13], Bengali [1] and Urdu [21] all contain modifiers, which can be seen as analogous to the fossil catalogue under- or overline denoting upper and lower jaw, increasing the problem similarity. Many characters are cursive in nature [21], which makes especially character segmentation solutions more applicable; both the south Asian and fossil catalogue characters are closer to equal in segmentation difficulty. The last trait making these languages applicable is that character sizes vary in many, at least in Bangla [26], which is also the case in fossil catalogues with the smaller upper and lower script characters. For the historical scripts, a study on machine reading historical manuscripts written on palm leaves was included in this review as the scanning of these scripts is more challenging due to the old leaves, and thus the study contained valuable insight on image quality enhancement techniques [31]. Additionally, historical scripts contain far more characters than modern languages, making the problem more challenging and thus more interesting. Next, these studies are presented in more detail.

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 [30] 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.

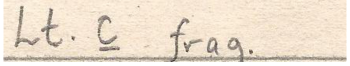
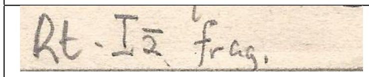
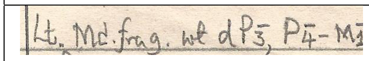
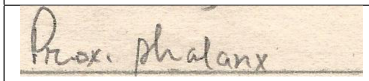
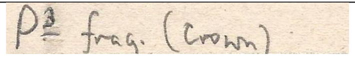
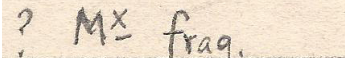
Input Image	**Label**
	Lt. C̲ frag.
	Rt. I2̌̄ frag.
	Lt. Md. frag. wt dP3̄̌, P4̄̌-M1̄̌
	Prox. phalanx
	P3̲̂ frag. (Crown)
	? Mx̲̂ frag.

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

While this work managed to accurately clean a part of dental markings present in the catalogues, there are several potential directions to significantly improve the digitization of such catalogues in general, the main direction being improving the quality of word-location defining bounding boxes. The main problem in the bounding boxes provided by Azure Vision [16] was that the dental marking words were often cut across many words or were present as a part of a longer word. This is likely due to the space width varying greatly between different pieces of handwriting. To improve the correctness of handwriting segmentation to words, developing a fine-tuned word detector could improve the result: a specialist model encoded with knowledge on what kinds of words are more likely to be present in the data would likely be much better at finding correct bounding boxes. An example of such case is seen in Figure 8: a segmenter with the knowledge of tooth marking notation could easily see that the dm_{3-4} section is one word, which is difficult to a generalist model due to the spacing between the characters.

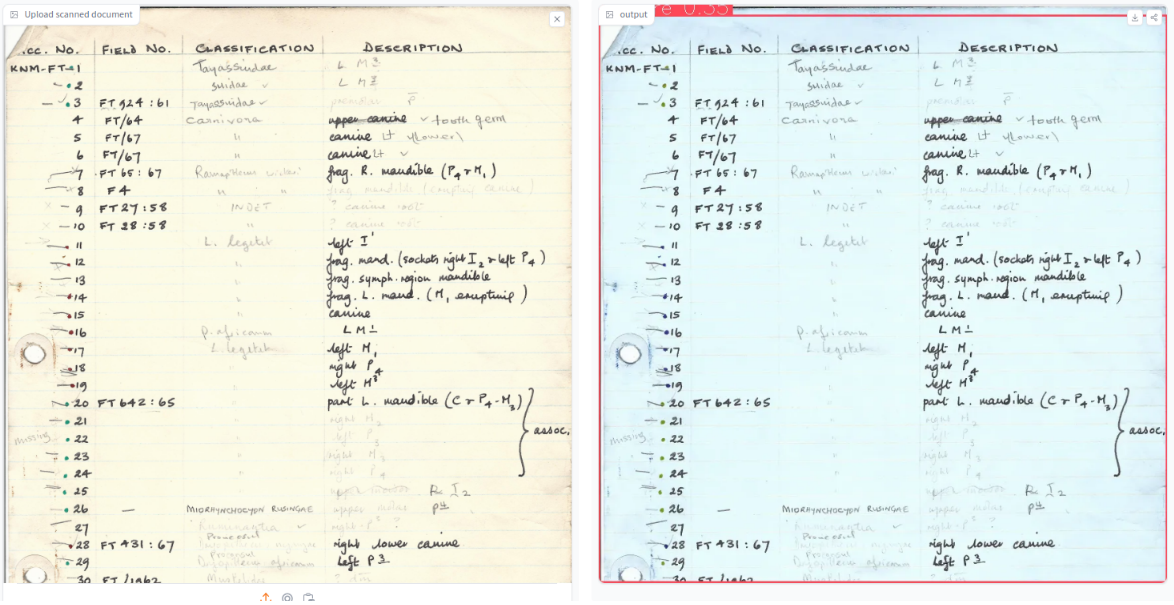
The problem of finding objects in images is much harder than image classification and has attracted attention from the computer vision research community very recently. Main reasons for the difficulty are the labour required to obtain ground truth bounding boxes and the fact that object detection requires better image resolution to work well, therefore requiring more computational resources [23]. One way to circumvent manual annotation for fossil catalogues could be to automatically detect correct bounding boxes from commercial OCR engine output files for instance by regular expression matching the word readings with a vocabulary of words likely to be present in fossil catalogues, much like the approach implemented in this work. For the detection model, several recent advances seem promising, but might not be directly applicable without fine-tuning due to the fact that the standard problem of object detection is to detect three-dimensional objects in 3D scenes, which is quite different from word detection on a flat, scanned document. This is likely the reason why an initial experiment ran on the Hugging Face API version [17] of YOLO-10 [33], a variant of the highly influential object detector YOLO [22], resulted in a very poor segmentation found in Figure 7: the entire image was segmented as one object. Some possibly more applicable models, however not as easily testable due to lesser popularity, are the open-vocabulary version of YOLO [34], the hierachical text spotter [14], and the Detectron by Meta [2]. However, due to the catalogue-specific problem illustrated in Figure 8, it is likely that these models would require well-optimized fine-tuning to work well. Experimenting with such implementations is a substantial effort and was omitted from this work to keep the scope reasonable.

Another possible line of extending the work would be improving the accuracy of the image classification to tooth type. Some ideas are to use a support vector classifier instead of fully connected layers combined with the softmax activation on the last layer, and to experiment with more recent ImageNet classifiers with better top-5 and top-1 error rates. However, using the newer models would require more effort since good pretrained weights are more difficult to find for the less established models, and there is less evidence of their applicability to downstream tasks. Improving the model would still be highly useful, since a more sophisticated word object recognizer would result in a more variable set of tooth marking images to be given to the tooth type classifier, making its classifying problem much more difficult. Still, the object detection problem should be given more attention since even the potential harder variants the dental marking classification task are relatively simple image classification problems when comparing them to challenges such as basic ImageNet classification.

References

[1] S. Chatterjee, R. Dutta, D. Ganguly, K. Chatterjee, and S. Roy. *Bengali Handwritten Character Classification Using Transfer Learning on Deep Convolutional Neural Network.* Apr. 12, 2020. ISBN: 978-3-030-44688-8. DOI: 10.1007/978-3-030-44689-5_13.

[2] *Detectron.* URL: https://ai.meta.com/tools/detectron (visited on 10/11/2024).



.cc. No.	Field No.	Classification	Description
KNM-FT-1		Tayassuidae	L M^3
2		Suidae ✓	L M^3
3	FT 924 : 61	Tayassuidae ✓	premolar $\bar{P}$
4	FT/64	Carnivora	~~upper canine~~ ✓ tooth germ
5	FT/67	"	canine Lt (lower)
6	FT/67	"	canine Lt ✓
7	FT 65 : 67	Ramapithecus wickeri	frag. R. mandible (P_4 & M_1)
8	F4	" "	frag. mandible (erupting canine)
9	FT 27 : 58	INDET	? canine root
10	FT 28 : 58	"	? canine root
11		L. legetet	left I^1
12		"	frag. mand. (sockets right I_2 & left P_4)
13		"	frag. symph. region mandible
14		"	frag. L. mand. (M_1 erupting)
15		"	canine
16		P. africanus	L M^1
17		L. legetet	left M_1
18		"	right P^4
19		"	left M^3
20	FT 642 : 65	"	part L. mandible (C & P_4-M_3) } assoc.
21		"	right M_2
22		"	left P_3
23		"	right M_3
24		"	right P_4
25			upper incisor R. I_2
26	—	Miorhynchocyon rusingae	upper molar P^4
27		Ruminantia ✓	right P^3 ?
28	FT 431 : 67	Proconsul Dryopithecus nyanzae	right lower canine
29		Proconsul Dryopithecus africanus	left P_3
30	FT /1962	Mustelidae	? dm

Figure 7: Preliminary object detection test with the YOLO-10 object detector using the Hugging Face inference API [17].

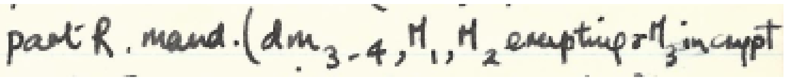

part R. mand. (dm$_{3-4}$, M_1, M_2 erupting & M_3 in crypt

Figure 8: Example of a sentence hard to segment to words without context knowledge

[3] D. Erhan, A. Courville, Y. Bengio, and P. Vincent. “Why Does Unsupervised Pre-training Help Deep Learning?” In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, Mar. 31, 2010, pp. 201–208. URL: https://proceedings.mlr.press/v9/erhan10a.html (visited on 09/26/2024).

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

[5] 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.

[6] Q. Groom, M. Dillen, H. Hardy, S. Phillips, L. Willemse, and Z. Wu. “Improved Standardization of Transcribed Digital Specimen Data”. In: *Database* 2019 (Jan. 1, 2019), baz129. ISSN: 1758-0463. DOI: 10.1093/database/baz129. URL: https://academic.oup.com/database/article/doi/10.1093/database/baz129/5670756 (visited on 09/30/2024).

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

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

[9] Know Your Meme. *We Need To Go Deeper.* Accessed: 2024-10-11. 2024. URL: `https://knowyourmeme.com/memes/we-need-to-go-deeper`.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems.* Vol. 25. Curran Associates, Inc., 2012. URL: `https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html` (visited on 10/02/2024).

[11] F.-F. Li and E. Adeli. *Transfer Learning.* `https://cs231n.github.io/transfer-learning/`. Course materials from CS231n: Convolutional Neural Networks for Visual Recognition, Stanford University. Accessed: 2024-10-16. 2024.

[12] 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]`.

[13] K. Limbachiya, A. Sharma, P. Thakkar, and D. Adhyaru. "Identification of Handwritten Gujarati Alphanumeric Script by Integrating Transfer Learning and Convolutional Neural Networks". In: *Sādhanā* 47.2 (May 19, 2022), p. 102. ISSN: 0973-7677. DOI: `10.1007/s12046-022-01864-9`. URL: `https://doi.org/10.1007/s12046-022-01864-9` (visited on 09/24/2024).

[14] S. Long, S. Qin, Y. Fujii, A. Bissacco, and M. Raptis. "Hierarchical Text Spotter for Joint Text Spotting and Layout Analysis". In: *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV).* 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). Waikoloa, HI, USA: IEEE, Jan. 3, 2024, pp. 892–902. ISBN: 9798350318920. DOI: `10.1109/WACV57701.2024.00095`. URL: `https://ieeexplore.ieee.org/document/10483893/` (visited on 10/11/2024).

[15] H. Mallison. "Digitizing Methods for Paleontology: Applications, Benefits and Limitations". In: *Computational Paleontology.* Ed. by A. M. Elewa. Berlin, Heidelberg: Springer, 2011, pp. 7–43. ISBN: 978-3-642-16271-8. DOI: `10.1007/978-3-642-16271-8_2`. URL: `https://doi.org/10.1007/978-3-642-16271-8_2` (visited on 09/30/2024).

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

[17] *Omoured/YOLOv10-Document-Layout-Analysis · Hugging Face.* Jan. 25, 2023. URL: `https://huggingface.co/omoured/YOLOv10-Document-Layout-Analysis` (visited on 10/11/2024).

[18] S. J. Pan and Q. Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[19] M. M. Phillips, K.-C. Cheng, and A. Hotz. *A Museum Overflowing With Prehistoric Treasures Races to Save Itself.* The Wall Street Journal. Oct. 12, 2024. URL: `https://www.wsj.com/world/africa/nairobi-national-museum-natural-history-leakey-832f0262` (visited on 10/15/2024).

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

[21] A. Rasheed, N. Ali, B. Zafar, A. Shabbir, M. Sajid, and M. T. Mahmood. "Handwritten Urdu Characters and Digits Recognition Using Transfer Learning and Augmentation With AlexNet". In: *IEEE Access* 10 (2022), pp. 102629–102645. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2022.3208959`. URL: `https://ieeexplore.ieee.org/abstract/document/9900315` (visited on 09/24/2024).

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE, June 2016, pp. 779–788. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.91. URL: http://ieeexplore.ieee.org/document/7780460/ (visited on 10/11/2024).

[23] L. Ruotsalainen. *Computer Vision Applications with CNNs.* Lecture for the course ”Computer Vision” presented at the University of Helsinki, Faculty of Science. Oct. 2024.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[25] S. Shanmugavel, J. Kannan, A. Vaithilingam Sudhakar, and . . “Handwritten Optical Character Extraction and Recognition from Catalogue Sheets”. In: *International Journal of Engineering & Technology* 7.4.5 (Sept. 22, 2018), p. 36. ISSN: 2227-524X. DOI: 10.14419/ijet.v7i4.5.20005. URL: https://www.sciencepubco.com/index.php/ijet/article/view/20005 (visited on 09/30/2024).

[26] M. Shopon, N. Mohammed, and M. A. Abedin. “Bangla Handwritten Digit Recognition Using Autoencoder and Deep Convolutional Neural Network”. In: *2016 International Workshop on Computational Intelligence (IWCI)*. 2016 International Workshop on Computational Intelligence (IWCI). Dec. 2016, pp. 64–68. DOI: 10.1109/IWCI.2016.7860340. URL: https://ieeexplore.ieee.org/abstract/document/7860340 (visited on 09/24/2024).

[27] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* arXiv.org. Sept. 4, 2014. URL: https://arxiv.org/abs/1409.1556v6 (visited on 10/04/2024).

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

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going Deeper With Convolutions”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, pp. 1–9. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html (visited on 10/07/2024).

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

[31] N. Thuon, J. Du, and J. Zhang. “Improving Isolated Glyph Classification Task for Palm Leaf Manuscripts”. In: *Frontiers in Handwriting Recognition.* Ed. by U. Porwal, A. Fornés, and F. Shafait. Cham: Springer International Publishing, 2022, pp. 65–79. ISBN: 978-3-031-21648-0. DOI: 10.1007/978-3-031-21648-0_5.

[32] M. D. Uhen, A. D. Barnosky, B. Bills, J. Blois, M. T. Carrano, M. A. Carrasco, G. M. Erickson, J. T. Eronen, M. Fortelius, R. W. Graham, E. C. Grimm, M. A. O’Leary, A. Mast, W. H. Piel, P. D. Polly, and L. K. Säilä. “From Card Catalogs to Computers: Databases in Vertebrate Paleontology”. In: *Journal of Vertebrate Paleontology* 33.1 (Jan. 1, 2013), pp. 13–28. ISSN: 0272-4634. DOI: 10.1080/02724634.2012.716114. URL: https://doi.org/10.1080/02724634.2012.716114 (visited on 09/24/2024).

[33] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding. "Yolov10: Real-time end-to-end object detection". In: *arXiv preprint arXiv:2405.14458* (2024).

[34] *YOLO-World: Real-Time Open-Vocabulary Object Detection.* ar5iv. URL: https://ar5iv.labs.arxiv.org/html/2401.17270 (visited on 10/11/2024).

[35] 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.

[36] S. Zhang, V. Callaghan, and Y. Che. "Image-Based Methods for Dietary Assessment: A Survey". In: *Journal of Food Measurement and Characterization* 18 (Oct. 28, 2023). DOI: 10.1007/s11694-023-02247-2.

[37] 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.