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 [6]. 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 [48]. 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 [26]. 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 [23]. 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 Paleoenvironmental reconstruction

3.2 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 [6]. 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 [13], 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.

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

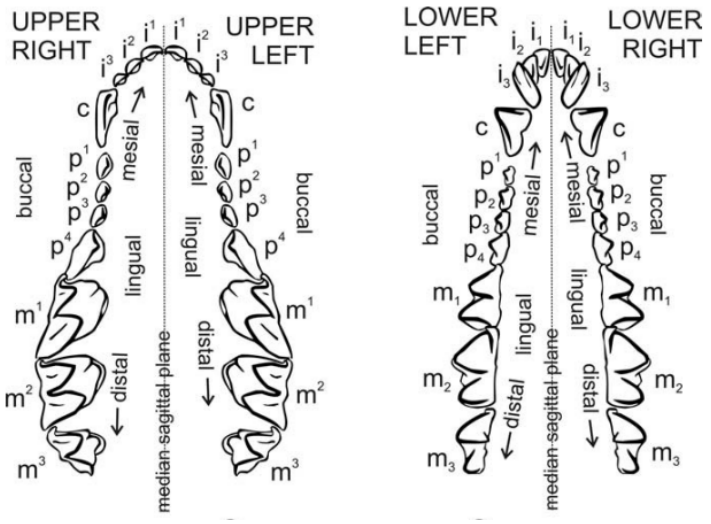


Figure 1: Mammal teeth composition, from Hillson [13].

output layer. Weights of adjacent layers are combined by activation functions, that are constrained to nonlinear functions with scalar inputs and outputs [27]. 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.

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

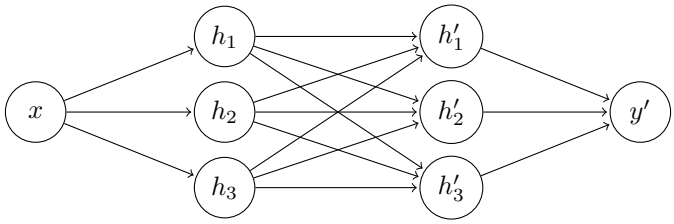


Figure 2: Visualization of a neural network with scalar input and output, and two fully connected layers, redrawn after [27].

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:

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	$\mathrm{M}^{\underline{1}}$	m^{1}	M1
Lower	$\mathrm{M}_{\overline{1}}$	m_{1}	m1

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

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

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 [27]. 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 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 3.2. 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.

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

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 [45]: 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 [38]. 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 [19].

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 [27].

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 [27]. 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 **??**. 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' [14].

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**
        $output \leftarrow \text{ForwardPropagation}(input, W, b)$
        $loss \leftarrow \text{CalculateLoss}(output, target)$
        $(gradients\_W, gradients\_b) \leftarrow \text{BackwardPropagation}(input, output, target, W, b)$
        $W \leftarrow W - learning\_rate \times gradients\_W$
        $b \leftarrow b - learning\_rate \times gradients\_b$
    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 [27]. 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 character classification for optical character recognition is the cross-entropy loss.

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 [27] for cross-entropy and the original paper [9] 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.

4.3 Architectures

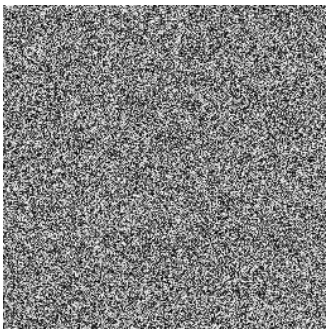
Neural network architectures are alternative ways of constructing the network layer computations for cases where the standard fully connected layer computation presented in Equation 2 is not ideal. Alternative layer types are used to make the model pay more attention to desired aspects [17], such as 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.

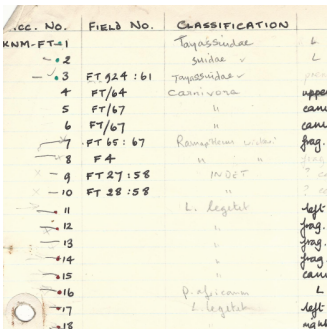
4.3.1 Convolutional layers

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 [17].

The two main pieces of prior knowledge encoded to the convolutional layer computation are invariance to transformations and local relatedness of pixels [27]. 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.



(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.

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

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 [27]. 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 6. For a colored image with three color channels, the kernel becomes three-dimensional. An illustration of such case is found in Figure 4.

$$h_{ij} = a\left[\beta + \sum_{m=1}^{3}\sum_{n=1}^{3} \omega_{mn} x_{i+m-2,j+n-2}\right], \tag{6}$$

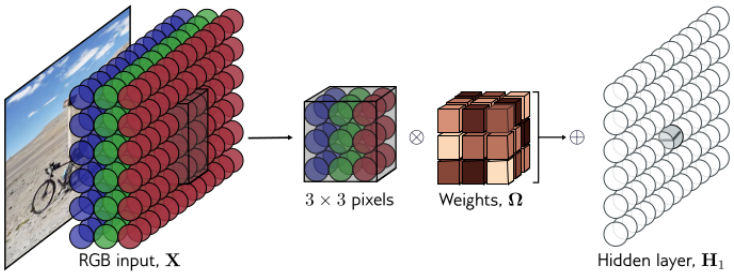


Figure 4: An illustration of a convolution computation in the case of a color image, from [27].

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 [27]. For the bordering pixels, where some kernel pixels fall outside of the input matrix, the options are to skip 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 6. 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 [27].

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 [27].

Many advances in convolutional networks have been motivated by the ImageNet image classification competition [32], 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 [17], the VGG model family [36], and the Inception architecture used in GoogLeNet [39].

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) [17]. 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, a regularization heuristic where connections are randomly dropped to avoid over-reliance on certain connections [37].

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 [39]. 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 [36]. 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 Residual connections

After the success of the very deep networks (VGG) [36], the question arised of whether achieving better approximation capacity with a neural network simply was about stacking more layers [12]. The previous major problem of vanishing and exploding gradients had largely been solved by input and between-layer normalization operations. The vanising gradient problem refers to when

Figure 5: The meme that inspired the Inception architecture in GoogLeNet [39], from [16].

the gradient of the loss with respect to some parameters becomes zero, these parameters can no longer be updated with the weight update rule presented in Algorithm 1 on line 9 for the duration of the entire training process. Exploding gradients, on the other hand, lead to numerical instability as adding a very large number to the weight is generally undesirable; the optimum is best approached gradually. Normalization solves these problems by updating the activations to a consistent distribution, thus avoiding gradients getting stuck to very large or very small values [15]. However, after stacking even more layers with the help of normalization operations, another issue, the degradation problem occurred. This refers to the phenomenon where when iteratively training deeper networks, the accuracy reaches a peak and then starts degrading quickly [12]. This suggested that there was something dysfunctional in the conventional deep network computations that started showing symptoms when a sufficiently deep network was trained.

As a solution to the degradation problem, He et al. [12] presented the residual connections. The premise was that if you add a skip connection that can pass any layer computation in the network, the deeper network contains in itself all the shallower variants, and the resulting accuracy can never be worse than in shallower networks. The skip connection was implemented by adding the input to the conventional layer computation, thus the layer computation becomes, expressing the layer without skip connection as $F(x)$, $F(x) + x$. Empirical results showed that this solved the degradation issue, and lead to a vast improvement to overall accuracy. It was hypothesized that

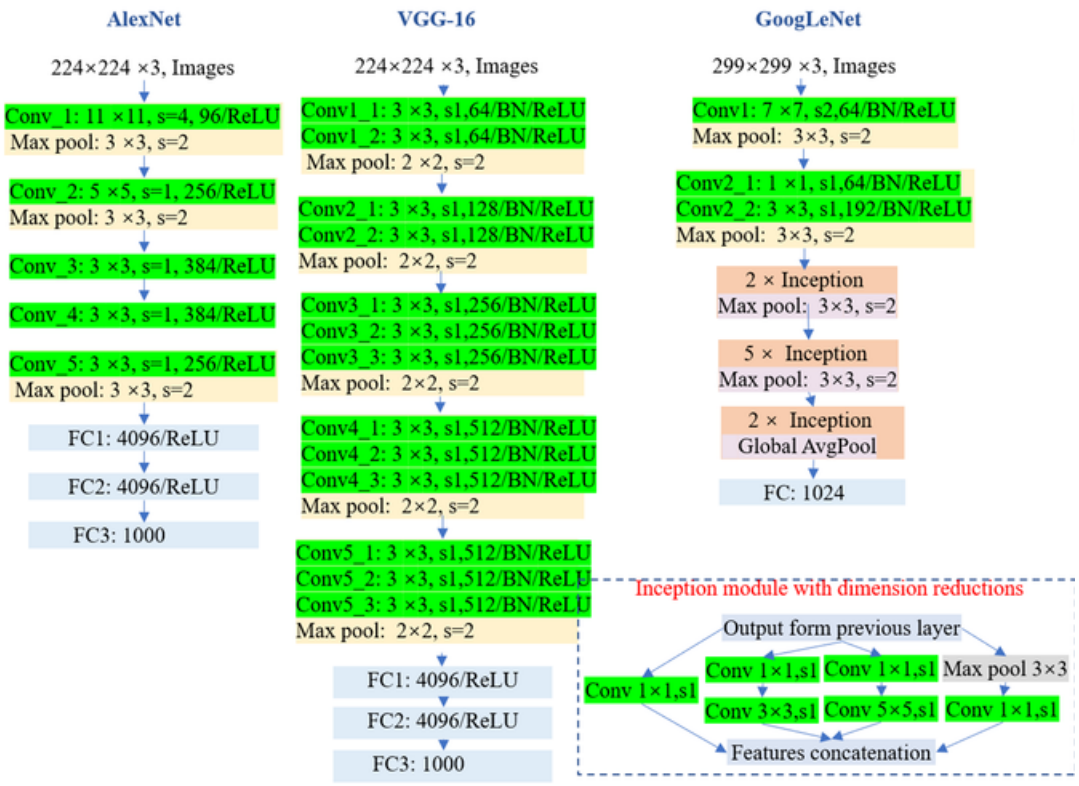


Figure 6: Architectures of AlexNet [17], GoogLeNet [39], and VGG-16 [36], the VGG model variant with 16 layers, from [46].

the reason for the degradation problem was that the deeper the network is, the harder it is for the network to approximate identity mappings, and when the depth passes a certain threshold, this drawback outweighs the added benefit of more layers. As the skip connection makes approximating the identity mapping very easy, all weights in the nonlinear computation can be set to zero, the identity mapping problem is completely eliminated. The end results were impressive: the ResNet architecture won multiple image detection and segmentation competitions, including the ImageNet challenge of 2015, where the new best top-5 error of 3,57% was reached with a network eight times deeper than the deepest VGG architecture [12].

4.3.3 Autoencoders

4.3.4 Transformers and the multi-head self-attention

The multi-head self-attention layer present in transformer architectures was originally motivated by natural language processing tasks, as other architectures encode prior knowledge on language poorly. Words in sentences sometimes refer to other words very far from each other in the sentence, which is impossible to model with a convolutional layer that only uses the immediate neighborhood as the input for activation computations. As sequences are processed as splitting the sequence to

tokens, for instance words, and generating longer embedding descriptors for describing the tokens and their position in the sequence, input sizes become insurmountably large to be processed with fully connected layers [27].

To construct the self-attention operation, the nature of language was studied as an information retrieval problem. The activation value for each input token should be computed using relevant parts of the rest of the input, thus one needs a method for ranking the other tokens in the sequence by relevance from the perspective of the input token. Using a search engine as an analogue, the token can be seen as a query, the input given to a search engine prompt. The alternatives to rank, web pages in the search engine case, are the other tokens. The end result, called the value, are the available web pages in the search engine case or tokens in the transformer case, ranked according to relevance.

To find the relevance ranking, the keys (all tokens or all web pages) and the query (a search prompt or an input token) need to be compared. A rudimentary search engine implements this by computing a similarity metric: a single value describing how much of the words on the query appear on each web page. This is computed for token embeddings between the query and key vectors with the cosine similarity, given in Equation 7. This similarity metric computes the cosine of the angle between two vectors, and scales it by vector magnitudes, giving a scalar metric with one denoting perfect similarity and -1 perfect dissimilarity. In the self-attention operation, as all activations are computed at once, the queries and keys form matrices Q and K. To compute the dot product with matrix multiplication, the other matrix is transposed, and the scaling is sligtly varied from cosine similarity by dividing by the square root of the number of tokens, denoted d_k. The result is passed through the softmax function (see Equation 3), to construct an attention matrix where each element describes how much attention should be paid between each token pair. Finally, to get the end result of tokens ranked by relevance, the token embeddings, forming the value matrix V, are multiplied with the attention matrix. To include learnable parameters to the layer, the key, query and value matrices are computed by multiplying the token embedding matrices with parameter matrices W_k, W_q and W_v. This forms the dot-product self-attention computation, summarized in Equation 8.

$$\text{Cosine similarity}(\mathbf{k}, \mathbf{q}) = \frac{\mathbf{k} \cdot \mathbf{q}}{\|\mathbf{k}\|\|\mathbf{q}\|} \tag{7}$$

$$\begin{gathered}\text{Self-attention}(\mathbf{X}) = \text{softmax}\left[\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right]\mathbf{V}, \\ \text{where} \quad \mathbf{V} = \mathbf{X}\mathbf{W}_v,\ \mathbf{K} = \mathbf{X}\mathbf{W}_k \text{ and } \mathbf{Q} = \mathbf{X}\mathbf{W}_q.\end{gathered} \tag{8}$$

With the self-attention operation, one limitation remains: one attention layer is only able to focus on one aspect of the input sequence at a time. To allow multiple interpretations, many self-attention operations are run in parallel, which is called multi-head self-attention. To not increase computation, the token embeddings are divided across the heads, and the results from the value matrix multiplied with the attention matrix are concatenated back together, creating a result of equal dimension as the one-head version. The self-attention operation with multiple heads is summarized in Figure 7.

The main premise of the transformer architecture [42] is that the multi-head self-attention is the only architecture required to successfully learn nature language processing tasks. The first

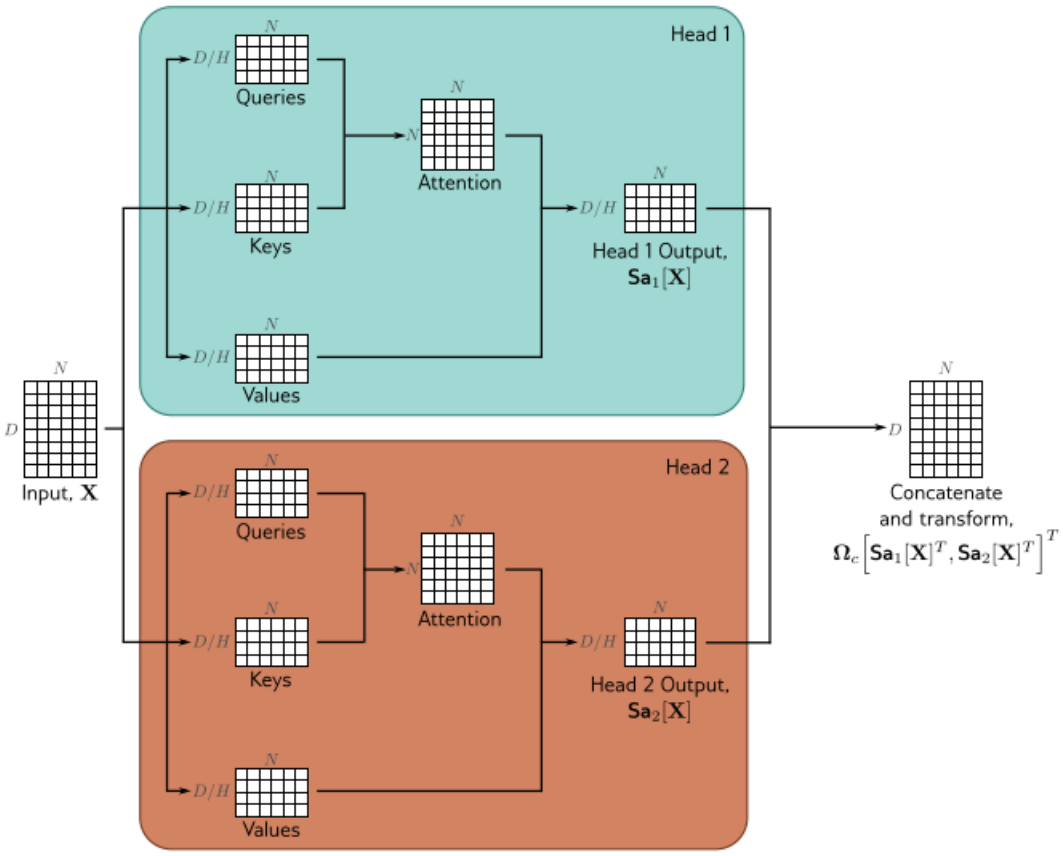


Figure 7: Multi-head self-attention with two heads, from [27].

transformer model, an autoencoder with both the encoder and decoder parts consisting of transformer blocks with a multi-head self-attention and a fully connected layer, both followed by batch normalization and with a residual connection, achieved a new best result in machine translation from English to German by a significant margin [42]. The most famous application of attention-based models is, however, a model taught to continue a sentence fragment with the next most likely character. The chat-style variants of these models, most famous being ChatGPT by OpenAI, has become very well known among the general public and has even been speculated to show signs of general artificial intelligence.

In computer vision, transformer architectures have been successfully implemented. The most notable of these is the Vision Transformer (ViT) [4], that achieved a top-1 error rate of 11.45% on the ImageNet challenge, an impressive result yet less accurate than the best convolutional networks of the time. Falling behind convolutional networks likely was the result of the transformer architecture lacking prior knowledge on images, translation and pointwise transformation invariance, leading to larger requirements of data and computational resources to reach high accuracies. Still, transformer

architectures are actively experimented with in many computer vision tasks, including handwritten character classification [40].

The layer computation variants presented in this section; convolution, residual connection, encoder-decoder chain and the multi-head self attention form the set of alternatives commonly used for image classification. Models built with these mechanisms have achieved impressive accuracies on classifying the ImageNet samples according to what can be seen in each image. To utilize these models in handwritten character recognition, a mechanism for applying this knowledge on a new task is required. This is called transfer learning and is presented next.

4.4 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 [25]. 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 ([5], [25]). The experimental section presented by Erhan et al. [5] 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 [25]. The practical implementation of transfer learning is better understood when one views a neural network as a hierachical feature 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 [14], 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 [5].

While the principle of transfer learning was known with many variants and names before, the first survey on the topic [25] 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 $X \rightarrow y$ 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 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 [32]. 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 [18]. 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.

4.4.1 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 [25], 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 $X \to X$ 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 [35]. 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. [41] 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 [22], 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. [10] 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. [33]. 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: there are similarities in 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 south Asian languages studied next with less

established public datasets are more applicable. Of these smaller languages, at least Gujarati [20], Bangla [2] and Urdu [28] 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 [28], 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 [35], 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 [40]. Additionally, historical scripts contain far more characters than modern languages, making the problem more challenging and thus more interesting.

Before each of the reviewed articles is presented in more detail, a few points on the overall quality of the work studied must be noted. Unfortunately, the analytical basis of the work reviewed was quite lacking; for the most part, model training setup decisions were made without any reasoning for why the decisions were made. Especially lacking are analyses on how similar the source task is to the target task, which would be critically important in getting transfer learning to work properly. Also unusual choices would have been useful to have a reason, but these were mostly missing. Where a reason was given, it often only resorted to listing general strengths of the technique chosen, but still lacked the most important part; why this technique was better than other available alternatives for this specific task.

The reasons and considerations regarding setup decisions are not nice-to-haves, but have quite some consequences for the quality of the research conclusions. In later work, one cannot evaluate the reasoning behind the choices. For some choices, it can be noticed that they do not make much sense at all, an example of this being using image horizontal or vertical flipping as a data augmentation technique for character images [40] (in latin letters for instance, a flipped L could become a J). Also, some test data setups become questionable from a generalization perspective, such as hyperparameter optimizing for the best angle to rotate every image as a data augmentation technique [30] (the idea of data augmentation is to augment the data set with valid images that just happen to not occur in the test set, and it is hard to imagine a case where only one rotation angle would produce valid training images), or applying data augmentation for the test set [49] (when all test images are not real-world samples, test accuracy is not a good representation of real-world performance). Irrelevant specifications are occasionally given, such as hardware when runs are not timed [40], and relevant details, such as if parameters were frozen, are sometimes omitted [8]. Less relevant metrics are measured, such as inference times even when the model is not part of a real-time system [8], or metrics where the exact definition is unclear, such as positive and negative related rates ([7], [28]) when the definition or implications of non-class-specific positive/negative related metrics is unclear in OCR in the usual case where correctness is equally important among all classes. Finally, analytical comparisons become frayed, such as the statement that a worse accuracy is compensated by the benefit of having to train for less epochs [2], when training in an OCR transfer learning setting is generally completed in order of minutes even on a CPU.

Due to these limitations of the reviewed work, the unexplored alternatives cannot be crossed out as probably unlikely to work well. Therefore, this work will experiment with some other than the most popular approaches that are implemented in the works presented next.

Zhao et al. [47] achieved new state of the art accuracy in multi-style Tibetan glyph classification with a model chain approach that first classifies glyphs by style and then recognizes the glyph with

a style-specific model. As these downstream residual learning models are trained with transfer learning with a Tibetan recognition model as the base model, the source and target data domains are very similar, thus the transfer setting is simpler than in the fossil catalogue case. However, the approach of chaining simple classifiers as a solution to a more complex problem served as an inspiration for the pipeline approach implemented in this thesis.

As another pipeline approach for detecting characters out of a longer sequence, Akhlaghi et al. [1] implemented a reading system for Farsi phone catalogues by chaining a gradient histogram based segmentation algorithm with a digit classifier. While the segmentation task is due to uniform character sizes much simpler than the catalogue case, the work has a possibly useful literature review on character segmentation. For the classification, a very small convolutional network was trained from scratch to a satisfactory accuracy of 94,6%, proving that with careful architecture design, even small models can learn to accurately classify digits.

The rest of the literature reviewed in this section is directly analoguous to the fossil case: a ImageNet or autoencoder base model is trained with transfer learning to detect a new set of characters.

Rasheed et al. [28] created an AlexNet-based Urdu recognition model, achieving 98,12% accuracy in digit recognition. The common last layer replacement with a fully connected layer with softmax activation was benchmarked against using AlexNet as a feature extractor before support vector machine classification, the former giving the best result.

For Bangla character classification, Chatterjee et al. reported 96,12% accuracy by fine-tuning a ResNet model. The work employed highly sophisticated transfer learning and learning rate scheduling algorithms, aiming to converge training in as little epochs as possible, an aim already discussed to not likely be worth the pursuit. Another concern was the concluding statement "if those misclassified points were removed from the dataset, the accuracy will improve further", which was not done to keep the benchmarking accurate.

In Bangla digit classification, Shopon et al. [35] implemented unsupervised pretraining and found a new best accuracy in Bangla digits and best accuracy among all articles reviewed in this section of 99,5% by chaining an autoencoder with a deep convolutional network. This proves that while less popular, unsupervised pretraining based approaches are worth experimenting with.

Lastly for the Bangla language, Zunair et al. [49] experimented with very unconventional transfer learning approaches to classify digits. The authors experiment with freezing layers below not frozen layers, an intuitively odd technique as later layer computation is very dependent on the output of the previous ones. Another even more preculiar approach is to immediately freeze the last fully connected layer, hindering any updates to the randomly initialized weights. The end result of 97,09% is stated to prove that these unusual approaches are worth experimenting with further, but due to this result being only found with this one data set and one base model, VGG16, and without any theoretical basis, the evidence for the sensibility of this approach is not quite sufficient.

Several articles detected characters form the Gujarati language. Goel et al. [7] were the first to attempt deep learning for Gujarati digits, achieving 96,5% accuracy with the EfficientNet architecture. However, the authors utilized validation error on the test set as a training stopping condition and then reported accuracies on the same set, leading to data leakage and a result not necessarily representative of true generalization capability. In a later article [8], the authors expanded their experimentation to other architectures, created several transfer learning settings informed by analysis on source and target task similarity, fixed the data leakage issue, and reported an improved accuracy of 97,92%, again with the EfficientNet architecture. Additionally, adding a fully connected layer is again found to outperform using a support vector machine as the downstream classifier.

Limbachiya et al. [20] implemented digits and letters detection by comparing various convolutional architectures, achieving the best accuracy of 97,03% with the MobileNet architecture. Transfer learning was implemented with a less usual top layer architecture: dropout was added between two new fully connected layers.

As the only work reading characters from a banchmark dataset of handwriting, Rizky et al. [30] first compared various architectures and then used the best model, VGG-16, to compare multiple transfer learning approaches, reaching a final accuracy of 98,16% by fine-tuning the whole network. However, to achieve this accuracy, the exact best data augmentation parameters of rotation angle, image scale and blur style are searched for, raising concerns of real-world generalizability of this result. While the dataset is likely to be easier than the fossil case, the relative accuracy of various approaches can give valuable insight to inform choices in designing the dental marking recognition model.

Lastly, a very interesting application case of reading characters used in ancient historical scripts written on palm leaves, Thuon et al. [40] achieved a very impressive accuracy of 93,55% on more than 100 classes from challenging palm leaf scan images. The work includes many valuable parts: image enhancement, data augmentation and dataset expansion insight, and comparison of multiple convolutional and attention based model architectures. As the main result it is concluded that convolutional networks tend to outperform transformer architectures.

Details on all the works presented above are summarized on Tables 3 and 4. Where multiple settings were experimeted with, the class count producing the highest accuracy score and the three best base model variants are listed starting from the best for brevity. The Tibetan detection article [47] is omitted due to transfer setting being very different to the other works. Replacing the last layer with a softmax layer with ouput size equal to target class count is not stated in the table as it was done in every case. The shorthand 'FC' is used to denote a fully connected layer.

Article	**Language**	**Number of Classes**	**Samples per Class**
[1]	Farsi	10	8,800
[20]	Gujarati	54	1,400
[2]	Bangla	84	1,977
[28]	Urdu	10	840
[35]	Bangla	10	2,922
[30]	English	36	2,055
[8]	Gujarati	10	800
[40]	Ancient Balinese (1), Sundanese (2), Khmer (3)	100 (1), 111 (2), 60 (3)	193 (1), 1,836 (2), 122 (3)
[7]	Gujarati	10	250
[49]	Bangla	10	8,500

Table 3: Literature Summary: Datasets

This chapter presented work related to the problem of recognizing handwritten dental fossil markings. Since little previous work on fossil catalogues existed, the attention was turned to a related problem on an abstract level; recognizing handwritten letters and digits in South Asian languages. The insight gained from the literature review will be used in creating the model training setup for the required deep learning models.

Article	Best Accuracy	Base Model(s)	Transfer Method
[1]	99.37%	New Architecture	-
[20]	97.03%	MobileNet, DenseNet, VGG16	Train new FC layer + dropout + output
[2]	96.12%	ResNet50	Train all layers, increase learning rate for later layers
[28]	98.21%	AlexNet	Train all layers but last layer most
[35]	99.5%	Autoencoder (trained on Bangla)	Append small deep CNN to autoencoder, no freezing
[30]	98.16%	VGG16, DenseNet121, ResNet18	Train all layers
[8]	97.92%	EfficientNetV2S, Xception, ResNet101	Train all fully connected layers
[40]	91.85%	EfficientNetB0, EfficientNetB1, ResNet101	Freeze first layers
[7]	96.5%	EfficientNetV2S, InceptionV3, ResNet101	Add 2 FC layers + output layer, no freezing
[49]	97.09%	VGG16	Freeze layers 16-20 (3 convolutional, 1 pooling layer)

Table 4: Literature Summary: Accuracy, Base Model and Transfer Method

6 Experimental setup

The goal of this thesis is to recognize special characters used to denote tooth type, jaw and index number to increase the accuracy of the element description in the digitized fossil catalogue dataset of the National Museum of Kenya. This chapter presents an end-to-end system that achieves this goal and gives reasoning for the design decisions made for the training setups of the required deep learning models.

The main hypothesis is that accurately extracting tooth notation from catalogue images is best completed with a divide-and-conquer approach: by decomposing the main problem into small subproblems of simple univariate classification, and chaining these to a tooth detection pipeline, the subproblems become easy enough to be solved with perfect accuracy on limited data. The aim of the experiments is to verify the plausibility of this approach and find the most accurate deep learning models to solve the subproblems involved.

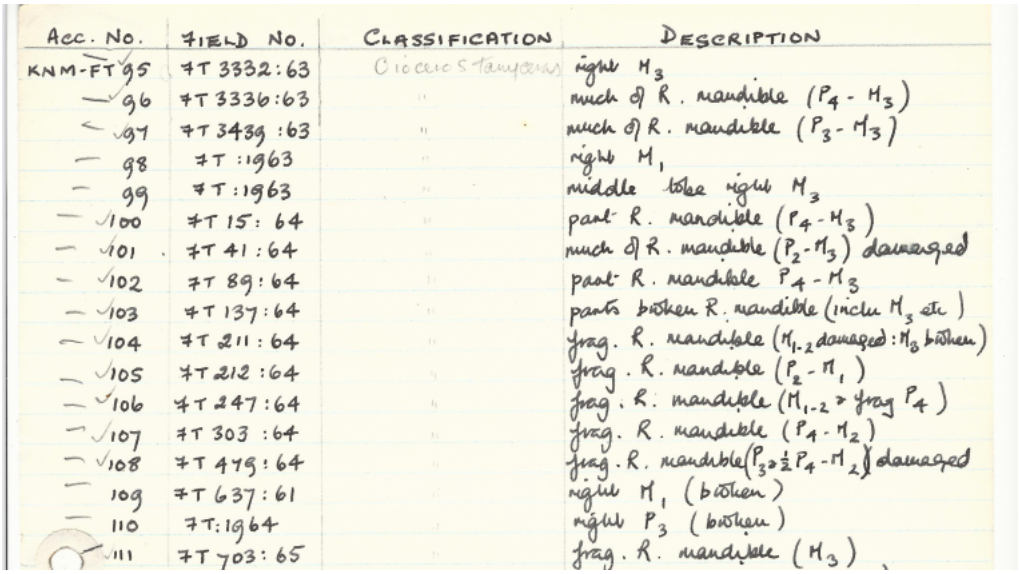
This chapter is organized as follows. In Section 6.1, the main problem is formulated as a set of $X \rightarrow y$ mappings that jointly achieve the ambiguous goal of cleaning the element description column in the fossil dataset. Then, the training settings attempted to build the required models are chosen, informed by studied literature, in Section 6.2.

6.1 Problem formulation

To give an exact task to a neural network, one needs to explicitly define the inputs and outputs to the network, and determine how they will be obtained and used. As this work is a continuation of a previous digitzation project, this section first presents the starting point from the project output and characteristics of the scan images.

The fossil catalogue images are hand-drawn tables with information such as accession identifiers, localities and element descriptions. The element description column, under the "description" or "element" header in the catalogues, contains information on what part of the skeleton the fossil is from and therefore is the only column with tooth notation. These tables were created before standardized databases were available, thus the catalogues occasionally contain special symbols recognizable by humans but that might only occur once in the whole set of catalogues. Additionally, no standard exists for how the elements should be notated, perhaps because only a few people created these markings, leading to a less standardized style. A sample of the catalogues is presented in Figure 8. The previous project managed to digitize most of the contents of the table, but since the untuned Azure AI OCR model [23] was used, the tooth notation recognition loses information

of upper or lower jaw tooth due to special characters, and sometimes erroneously recognizes tooth type letters and index numbers as characters one can know to be incorrect with prior knowledge on characters present in tooth notation. The aim of the experiments in this work is to find an approach for fixing these errors.



Acc. No.	Field No.	Classification	Description
KNM-FT 95	FT 3332:63	Cricetidae Stenomys	right M_3
96	FT 3336:63		much of R. mandible (P_4 - M_3)
97	FT 3439 :63		much of R. mandible (P_3 - M_3)
98	FT :1963		right M_1
99	FT :1963		middle lobe right M_3
100	FT 15: 64		part R. mandible (P_4 - M_3)
101	FT 41 :64		much of R. mandible (P_2 - M_3) damaged
102	FT 89: 64		part R. mandible P_4 - M_3
103	FT 137: 64		parts broken R. mandible (incl M_3 etc)
104	FT 211 : 64		frag. R. mandible (M_{1-2} damaged : M_3 broken)
105	FT 212 :64		frag. R. mandible (P_2 - M_1)
106	FT 247 :64		frag. R. mandible (M_{1-2} > frag P_4)
107	FT 303 :64		frag. R. mandible (P_4 - M_2)
108	FT 479 : 64		frag. R. mandible (P_3 > ½ P_4 - M_2) damaged
109	FT 637 : 61		right M_1 (broken)
110	FT: 1964		right P_3 (broken)
111	FT 703: 65		frag. R. mandible (M_3)

Figure 8: A sample from the fossil catalogue sheets of the National Museum of Kenya

The aim is for the end result dataset to contain a column with a tuple of all teeth present in the element description, and to replace the erroneously read teeth in the element column with the correct marking. The notation used in the end result will be the digital notation presented in Table 2. The recognition will be built assuming that bounding boxes for cropping out individual words in the catalogues are available; for this purpose, JSON files with bounding boxes obtained in the previous project output obtained with Azure Vision are used. An example of the desired output, obtained by hand-annotating as no perfect solution is found in this work, is presented in Figure 9. As can be seen from the dataset sample, the end result in terms of the other data is far from perfect, and therefore much remains to be done after this work.

The guiding trade-off to discuss when deciding which neural networks can be of use to achieve the goal of corrected tooth notation is between accuracy and generality. The more prior information one gives to the model, or the more constrained the task is for the model, the better the results are. However, a very constrained model is poor at handling the inevitable variability in human-written text. The approach implemented aims to start from the most constrained model for multiple reasons. As this domain is relatively unexplored for OCR research, it is a reasonable approach to start out from simple solutions. As one can always first filter out the easiest cases, for instance teeth where bounding box segmentation result is very good, and recognize those, cleaning data instances that are easiest is a low-hanging fruit worth starting out from. As more generalist models are usually less accurate even on the easier samples, they are more likely to make mistakes even on these easier cases. Additionally, as the aim of automated digitization is to avoid the tedium of manually checking each resulting sample, no model with accuracy less than the human perception

ACC_NO	FIELD_NO	CLASSIFICATION	DESCRIPTION	TOOTH_RECORDS
KNM-FT 95	7T 3332:63	Ció ceros Tamyoua	m3	(m3)
96	7T 3336:63	much	of R. Mandible (p4- m3)	(p4, m3)
	7T 3439 : 63	much	of R. Mandible (p3-m3)	(p3, m3)
98	7T : 1963	right	m1	(m1)
99	#T : 1963		middle tobe right m3	(m3)
100	++ 15:64	pant	R. Mandible (p4-m3)	(p4, m3)
101	7T 41:64	much	of R. Mandible (p2-m3) damaged	(p2, m3)
102	7T 89:64	paul-	R. Mandible p4-m3	(p4, m3)
−103	7T137:64	parts	broken R. Mandible (inclu m3 etc)	(m3)
−104	7T 211: 64	frag.	R. Mandible (m1-2, damaged: m3 bothe	(m1, m2, m3)
105	7T 211: :64		frag. R. Mandible (p4. M2.)	(p2, m1)
~106	7T247:64	Jag.	R. Mandible (m1-2 > frag p4)	(m1, m2, p4)
/107	#T 303 :64		Jurag. R. Mandible (p4-m2)	(p4,m2)

Figure 9: The goal: teeth are listed in tooth records column and the description column contains correct markings.

level of approximately 99,7% are usable, since even in a case of minor insecurity the sample needs to be checked: sample sizes in paleontological studies are relatively small, thus the cost of a mistake in a dataset is severe as it propagates to every study utilizing the data point.

With the end goal of correcting the tooth notation and guiding design principle of starting out simple in mind, the next sections will discuss two major questions in how to set up the tooth recognition system: whether to give the whole content of the cell under element header or only a word to the model, and how to set up the target classes for the classification tasks.

6.1.1 By-sequence or by-word recognition

To choose the inputs for the neural networks, one needs to choose whether to give the element cell contents as one image, split to words or to characters. As the segmentation data is only available in sentence and word level, the character option is not available in this work.

Sequence learning is a machine learning problem where both input and output lengths vary [38]. The clear benefit is its flexibility; more difficult cases such as many teeth noted together, eg. M_{1-3}, which in many cases end up segmented to different words, could be recognized. A fine-tuned sequence learning model could also be given prior information on that the domain the text is from is about fossils, which would make a large part of the available vocabulary impossible to occur, likely increasing the result quality. As for downsides, the clear practical hindrance is that the flexibility requires huge parameter counts: an attempt was done to fine-tune the Microsoft TrOCR model [19], but the training set sizes and available computational resources proved insufficient to train the model with 500 million parameters even when updating only one layer, and results of smaller models tested were not satisfactory. The end result of the fine-tuning attempts had severely decreased accuracy on handwriting without any special characters. A theoretical description of this case is that because both the domain, from general handwriting to fossil catalogues, and the task, the characters to recognize, changed, the transfer distance becomes so large that significant training data and computational resources are required to adjust the model to the new problem.

Recognizing the markings word by word essentially becomes character classification in the tooth marking case if one constrains the input images to be samples with one letter and one number, thus leaving out the multi-tooth markings with a number interval (eg. p_{2-4}). This approach has the benefit that after the rather easy binary classification of a word to tooth marking to other word one

can encode prior information on the dental row to the classification task by having, for instance, one class for each tooth type, creating classification settings with relatively small number of classes. These can be then solved even with small amouts of data and limited computing resources. The drawback is a lack of flexibility, propagation of bounding box detection errors and that left and right jaw information has to be left out from the tooth records column: the jaw side information is not always given in the same word, in format such as 'LM_1', but in other parts of the descriptions, in formats like 'left mandibular frag. with M_1'.

Given the problem of classifying a tooth notation image with a type specifying letter and an index number, the classification setting is formulated next.

6.1.2 Tooth marking classification: multi-label classification or classifier chaining

Given the chosen approach that the processing is completed word by word, and more variable inputs like notation marking multiple teeth with one word are left for future work, one still needs to formalize the output format. The other than tooth case is straightforward; as the Azure AI results are very good, one can re-use the already obtained reading result. Less straightforward is which classes to set up for the tooth marking classification: the image has many points of information, tooth type, jaw and index number with interdependencies like the index number range depending on the tooth type.

The simplest formalization would be to have one class per one tooth, the classes becoming for lower jaw m1-m3, p1-p4, c and i1-i3, totaling 22 classes when including the upper jaw teeth. This, however, encodes prior information falsely: univariate classification would mean that all classes are equally dissimilar to one another, which clearly is not the case as some inputs share same letters and some the same numbers. Thus, there are three separate problems; which jaw, which tooth type and which index number is found in the image, and the univariate formalization does not encode this fact.

The natural variation would be multivariate classification with the classes m, p, i, c, up, low, 1, 2, 3 and 4. However, the most general multivariate setting does not constrain these classes to any groups or that one class would be needed to be chosen from each [45], thus other works would present approaches where for example the correct classes m, p, 3 and 4 all at once are possible, which is clearly wrong in the tooth case. Additionally, even if the target classes would be grouped and one class being correct would be enforced, the tooth could be classified with a nonexistent index number, such as the third canine. Thus, the possible classes would need to be constrained to encode this fact, leading to a very complex and unusual classification problem for which it could be difficult to find existing solutions. Added to this, the work on Tibetan glyph recognition identified the complex multivariate formalization of previous Tibetan recognition, where one recognized the character and writing style both at once, the main reason why the resulting accuracies were not as good as in univariate classification [47].

These shortcomings, along with the successful pipeline approach implemented for Tibetan recognition [47] lead to the univariate classifier chaining approach used in this work: first, one gives the image to a type recognition model, that ouputs one probability vector of four classes, molar, premolar, canine or incisor, and a upper or lower jaw binary classifier. Then, given the tooth type, the image is given to the index number classifier, which has the correct number of classes given the type: three for molars, four for premolars, no model for canine since one is the only correct index, and three for incisors. The three-class cases were built with separate models for the network to not learn connections between the letter and index number, which would be incorrect. The upper

and lower classifier was not chained in this way with the tooth type classifier (which would result in eg. a upper or lower molar decision model), because incisor and canine samples were too few to construct datasets of sufficient size, and all tooth types have both upper and lower jaw variants. This approach was not directly found in any other previous work, but was decided for since it seemed like the only one that correctly encoded all similarities and impossible class combinations.

Next, the end-to-end pipeline for getting the tooth markings out of the catalogues using these models is presented.

6.1.3 The proposed pipeline

To summarize the problem formulation discussion above, the end-to-end dental fossil marking system, illustrated as a flowchart in Figure 10, has the following components.

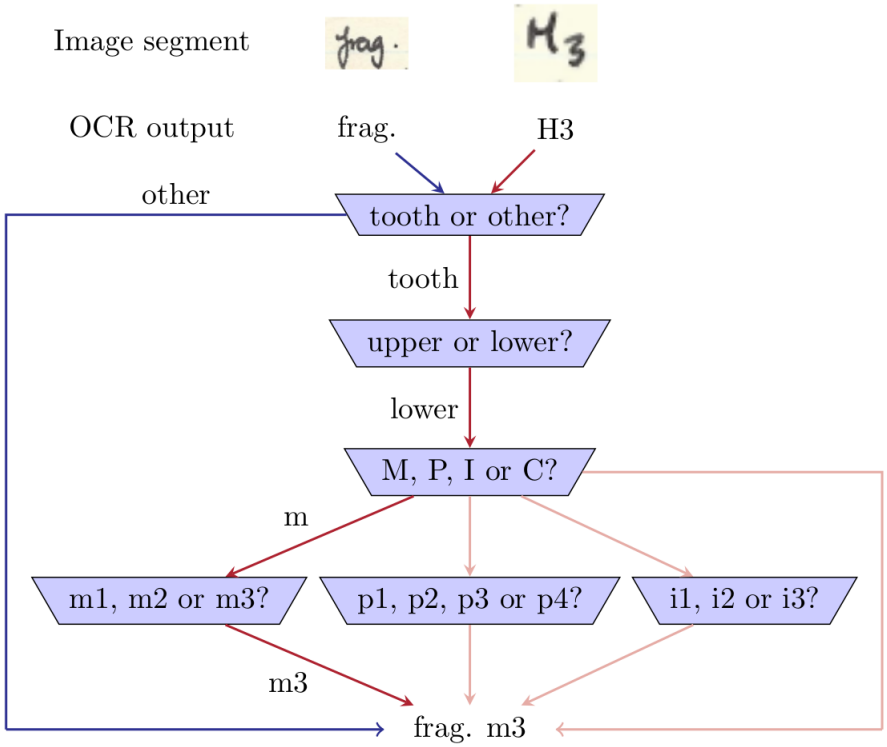


Figure 10: Flowchart of the proposed tooth recognition pipeline

For all words, bounding boxes and reading results, detected on the element column by Azure AI, the word is first classified to tooth marking or other word. To sufficiently constrain the input data to the downstream model, this classification is completed with a simple regular expression checking whether the Azure reading result of the word consists of a letter and number, or only the letter C. Once more robust classifiers are built, this part can then be replaced with another deciding heuristic that would then allow a larger variety in tooth notation images to be cleaned; the regular expression only recognizes a small fraction due to frequent errors in bounding box detection.

The images of the words, extracted from the catalogue by the bounding box, that are classified as tooth markings are given first to a binary image classifier deciding whether the tooth is upper or lower jaw and then to a model recognizing if the tooth type is molar, premolar or incisor: as canines lack the number and are recognized as teeth only when Azure claims the word to contain a C, there is no need to re-classify the tooth marking as a canine. It should be noted that as soon as the tooth marking decision heuristic is made more complicated, it is likely that the fourth class has to be added to the tooth type classifier. Once the type is decided, the tooth marking image is given to the appropriate index number model. Finally, the three pieces of information obtained; jaw, type and number; are combined as the digital tooth marking notation. As the final step, all words in the element column are concatenated, and all found teeth are collected as a tuple to the tooth records column.

To avoid as much manual verifying of the result as possible, additional confidence information is added to each tooth marking entry. As each classifier produces the probability of the image containing the chosen class as a side result, this probability can be saved as a confidence score. The confidence is taken per element description phrase (such as 'much of R. mandible (P_4 - M_3)') by taking the minimum over all model confidence scores obtained during inference for the tooth records. For example, if in the example description every other decision would be of 100% confidence but the second tooth being a molar would only have 70% confidence, the confidence of the entire row would be 70%. This conservative approach is due to the costliness of errors: one needs to be very certain before data can be left without manual verification.

Next, details related to building the models presented in this section and presented.

6.2 Building the models

Given the chosen system and using a regular expression as the classifier of words to tooth markings and other words, a total of five classifiers are needed for the system: the upper/lower classifier, tooth type classifier, and the three downstream tooth index number classifiers. The experimental section considers how to build these models with maximal accuracy, for which several design decisions need to be made for setting up the experiments: data preprocessing and augmentation along with base model, transfer learning method and hyperparameter selection. After a presentation of how the training dataset was created, these aspects are considered.

6.2.1 Creating the training dataset

The training dataset with xx (TODO insert how many) samples was extracted from the fossil catalogues using the first two steps on the processing pipeline in Figure 10: for each word under the element describing header, the Azure Vision output was classified to tooth or other with the regular expression, and images with tooth notation were cropped out from the catalogue. Along with the image, the OCR reading result was saved. To have as versatile notation as possible, images were extracted from several catalogues.

After extraction, data balancing and splitting to training, testing and validation sets was completed. Because incisors and canines were much more rare than molars and premolars in the catalogues, some of the former were manually cropped out of catalogues to achieve more variety among these classes. After this, the differences in sample count between the most frequent class and others was computed, and augmented images were saved for the other classes to make class sample counts exactly equal. The augmentation method used will be presented in section TODO

aug section. The dataset sizes and samples counts per class for each model are collected in Table TODO (train val test and samples per class + split ratios for all 5 model training sets!).

For data labeling, the previous outputs from Azure Vision were heavily utilized. For tooth types and index numbers, if the reading result was a valid class, it was assumed to be correct and used as a label, which saved a significant amount of labeling effort. As the images were saved in class-specific directories according to the PyTorch ImageFolder interface, the correctness of the labels and that all images really were tooth markings could easily be verified by visual inspection. Upper and lower jaw labeling was completed manually with a ternary label, zero noting lower, one upper, and -1 an undecipherable jaw. The undecipherable class was included as there were images where the jaw could not be decided by a human annotator, and therefore the correct answer for such an instance would be 'unknown'. Forcing these to be either upper or lower jaw would distort the classification task, as perfect accuracy would be an undesirable result as the model should make mistakes with the undecipherable samples, and confuse the model as it is required to find signals on why the notation is either jaw when none are to be found.

As the work was completed under a non-disclosure agreement to protect the data, the training datasets were not made public. As the problem ended up being an interesting task from the perspective of optical character recognition, bringing even parts of the catalogue scans available for researchers and machine learning hobbyists could prove to be very valuable both for improving the digitized data quality and advancing character recognition techniques.

A small sample of the training dataset of premolars is presented in Figure 11.

Figure 11: Samples from the training dataset for tooth type classification of premolar markings.

6.2.2 Data preprocessing

The preprocessing of the training, validation and test images was completed by grayscaling, denoising and resizing. As the color channels in the images have no meaning for the text content on the image, these were removed by grayscaling the input images and then stacking three identical grayscale images to match the three-channel input size required by the base models. Denoising was implemented due to the images containing noise from small shadows from paper texture and

eraser smudges with the most basic routine, a Gaussian filter. The kernel size 3 was chosen by experimentation and visual inspection. Finally, the images were resized to ImageNet image size of 224x224 using bilinear interpolation. A catalogue sample before and after grayscaling and denoising can be found in Figure 12.

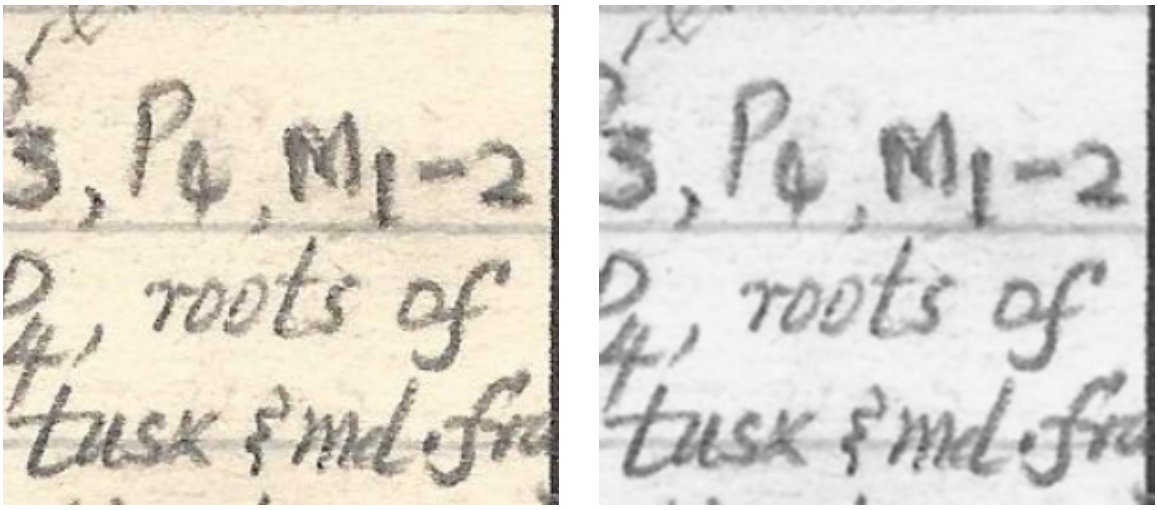


Figure 12: A catalogue sample with visible paper texture shadow and eraser smudge noise before and after grayscaling and denoising.

No more preprocessing was implemented on the input images to keep the transfer distance between ImageNet images, that are real-world noisy images, and the target domain images small. For this reason, for instance, flipping the background color to black and text to white was not implemented, even though this is a common practice in other character recognition works. The practice is likely habitual; when using MNIST base models this increased source and target domain similarity, while this is not the case when using ImageNet as the source domain.

More heavy preprocessing steps of binarization and extracting only the letter or number from the image were attempted but not implemented in the final system. Binarizing the input images was considered as evidence was found for that to increase result accuracy [40], but was not implemented since the noise, in many cases parts of neighboring characters due to bounding box errors, are occasionally of darker color than the correct text, making thresholding difficult. Additionally, initial experiments with a MNIST base model suggested the resulting accuracies were decreasing when using binary input images. However, binarization scheme was not designed carefully and the experimentation was not exhaustive, thus binarization could be experimented with further. Another aspect considered after binarizing was to try to segment out the letter for the tooth type recognizer and only the number to the index number recognizer, but it was found that there was so much variability among the images that implementing a deterministic algorithm that worked in every case was very hard. As the premise of deep learning is that such challenging feature extraction becomes unnecessary as the neural network is better at extracting relevant information than a human designing preprocessing routines [27], these heavier preprocessing operations were not used.

6.2.3 Data augmentation

The data augmentation operations were chosen according to the variation that can occur among scanned documents with handwriting. Due to paper and pen color differences, functions on pixel intensity values were used to adjust brightness and contrast by using the torchvision ColorJitter transformation with brightness factor range from 0.95 to 1.05 and contrast factor range from 0.8 to 1.2. To mimic bounding box detection errors, a random crop was implemented with the torchvision RandomResizedCrop with scale from 0.9 to 1.0 to not crop out significant parts of the character but still introduce cropping variation. Rotation from -5 to 5 degrees was implemented as some tables were sligtly tilted in the scan images. Lastly, handwriting variation was mimiced with horizontal and vertical shear by setting the shear ranges for both x and y directions from 1 to 10 pixels with the RandomAffine operation. The background was filled in with white after rotation when necessary to match the other background as closely as possible. A batch of training images after augmentation is presented in Figure 13.

Figure 13: Samples of the training dataset after data augmentation.

As class imbalance is known to reduce classification performance [11], data balancing was implemented using a part of the aforementioned augmentations. The balancing was implemented by computing the sample size differences between the most frequent class, and saving a sufficient number of augmented images to the data directories to make the counts of samples in each class exactly equal. Only the random crop and rotation was implemented for class balancing to not run the same augmentation procedure for the new images twice as this double transformation could lead to very distorted training samples.

6.2.4 Base model selection

To select appropriate base models, one first needs to select the source task and domain. Since most reviewed work uses image classification as the source task, this is assumed to work best. Another potential source task would have been the identity mapping task given to the autoencoder, but it was not implemented due to small target datasets and lesser popularity. As the source domain, there are two datasets used in previous work: the classification of digits in the MNIST dataset, and the ILSVRC classification challenge with the ImageNet dataset. The popular assumption that ImageNet is a better source domain was verified with a short experiment by training a MNIST classifier [34] to classify tooth types with best accuracy of 89%, that was clearly outperformed by the ImageNet-based classifiers. This verifies the commonly assumed hypothesis that real-world images of any objects is a more similar domain than the MNIST dataset of handwritten digits, which is known to contain ideal-case images with little noise.

The ImageNet classifier base models used in this the experiments are selected according to results from related work, reviewed in Section 5.2. The models were chosen according to the simple heuristic of checking which base models occurred multiple times when listing the top three models in terms of accuracy. Analyzing Table 4, EfficientNet models consistently outperform other architectures, most often with the V2S variant, closely followed by the 16-layer variant of the VGG architecture [36], DenseNet121, and ResNet101 (todo: cite the model original papers). Therefore, these four architectures were chosen as base models to experiment with. The shortcoming in this base model selection heuristic is that the previous work did not give much reasoning for their base model selection, it is possible that some high-performing architectures were not included because previous works did not choose to experiment with them. As one has to limit the search space of architectures in some way, previous success was still kept as the deciding heuristic even when acknowledging these limitations.

6.2.5 Transfer method selection

For the transfer method, the reviewed work in Table 4 has a big variety of approaches, the only common trend being that freezing less seems to outperform freezing many layers. However, how much to freeze is very dependent on the source and target problem differences; in very similar cases the features should be more useful and thus freezing more should work better, and target dataset size; if target data is small one should freeze more to avoid overfitting to the training set. For these reasons it is argued that transfer method selection should be based on experimentation rather than previous results.

For the choice of which layers to freeze, layers will be only frozen so that a frozen layer never occurs after a not frozen layer. This is because of the assumption that high-level layers encode generic and lower layers more specific features, and the generic features are assumed to be most transferable among tasks. Additionally, as specific features build on the generic ones, freezing layers below ones modified is assumed to confuse the model. Thus, all experiments consist of some layers at the start of the network being frozen and others at the end being open for updates. A single number noting the number of trained layers at the end of the network is therefore a sufficient description of the transfer approach.

As another experimentation principle, the trials will start out from settings where more layers are frozen, proceeding to cases where more parameters can be updated. This is because the setting with more freezing saves computational effort, and is less likely to overfit to the training set as there is a lesser degree of freedom available for updating the parameters. Therefore, it is sensible to look for the model that re-uses as much parameters as possible but still has enough parameters to train to find a good optimum for the target problem: this is assumed to result in the best generalization capability.

6.2.6 Hyperparameter selection

As for hyperparameters, values are selected according to popular or well-performing options in reviewed literature to scope down the search space in the experiments. Hyperparameter tuning was not implemented to keep experimentation between approaches fair; if the optimization is better for some case than other, one can no longer deduce whether the difference in result accuracy was due to hyperparameter or base model and transfer approach optimality. Therefore, hyperparameters are fixed between experiments to common values used previously in similar tasks. The selected

hyperparameters, most chosen as the most popular and number of epochs chosen as an average value among the reviewed works, are summarized in Table 5.

Table 5: Hyperparameters used in the experiments.

Hyperparameter	**Value**
Batch size	64
Number of epochs	50
Optimizer	Stochastic Gradient Descent
Learning rate	0.0001
Loss function	Categorical cross-entropy

In this chapter, the experimental setup along with reasoning for design decisions were given. In the problem formulation, classifying words with a chain of univariate classifiers was opted for to maximize the resulting accuracy by simplifying the problem as far as possible, because this is the first attempt to automate data cleaning for digitized fossil catalogues. Preprocessing was kept minimal with denoising and grayscaling, and augmentation was set up to variations that might occur in the scanning process. Base models tried were chosen according to previous success, resulting in selecting the EfficientNet, VGG16, DenseNet121 and ResNet101 architectures. The transfer methods are compared by experimentation: the model with most layers frozen while still being able to find a good optimum for the target task is searched by iteratively unfreezing layers from the end of the network. In the next chapter, results from these experiments are presented and evaluated.

7 Results and discussion

To evaluate the experiment results, the accuracy score is used as the only metric. As false positives and false negatives are not of major concern in character recognition, these were not utilized. For class-specific error rates, confusion matrices are generated, but otherwise confusion rates between classes are not studied since class count is small in all tasks and keeping a single evaluation metric makes comparison between experiments increases clarity.

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 [23] 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 15: 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 [31]. 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 [24] of YOLO-10 [43], a variant of the highly influential object detector YOLO [29], resulted in a very poor segmentation found in Figure 14: 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 [44], the hierachical text spotter [21], and the Detectron by Meta [3]. However, due to the catalogue-specific problem illustrated in Figure 15, 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] M. Akhlaghi and V. Ghods. "Farsi Handwritten Phone Number Recognition Using Deep Learning". In: *SN Applied Sciences* 2.3 (Feb. 14, 2020), p. 408. ISSN: 2523-3971. DOI: 10.1007/s42452-020-2222-5. URL: https://doi.org/10.1007/s42452-020-2222-5 (visited on 09/25/2024).

[2] 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.

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

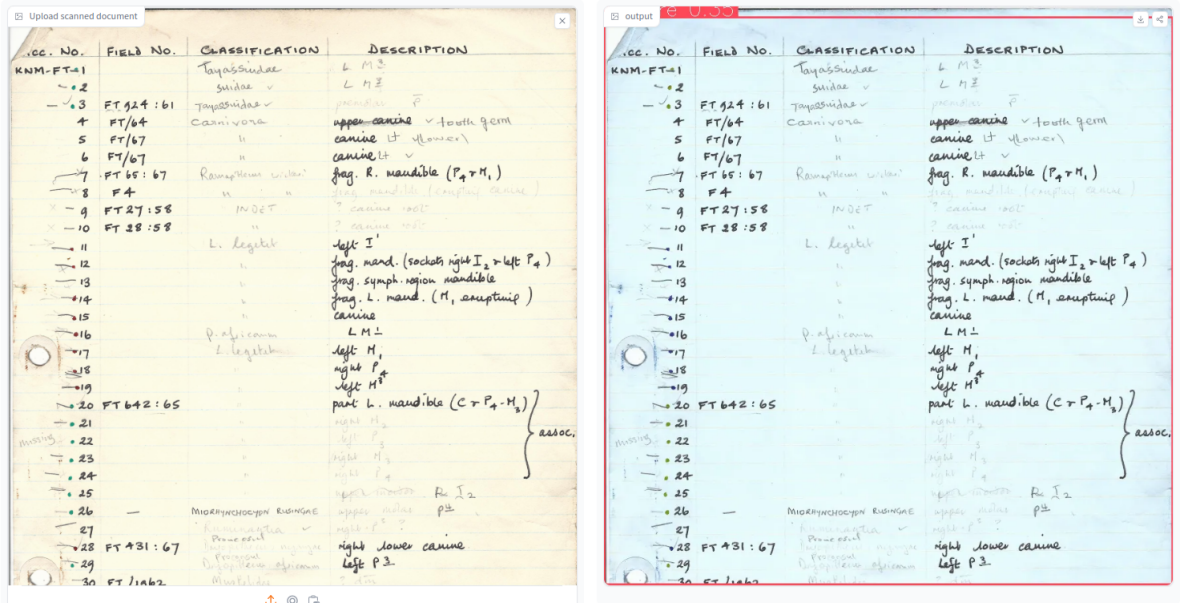


Figure 14: Preliminary object detection test with the YOLO-10 object detector using the Hugging Face inference API [24].

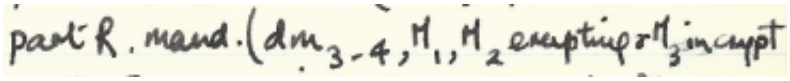


Figure 15: Example of a sentence hard to segment to words without context knowledge

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. *An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. June 3, 2021. arXiv: 2010.11929 [cs]. URL: http://arxiv.org/abs/2010.11929 (visited on 09/26/2024). Pre-published.

[5] 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).

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

[7] P. Goel and A. Ganatra. "A Pre-Trained CNN Based Framework for Handwritten Gujarati Digit Classification Using Transfer Learning Approach". In: *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT). Jan. 2022, pp. 1655–1658. DOI: 10.1109/

ICSSIT53264.2022.9716483. URL: https://ieeexplore.ieee.org/abstract/document/9716483 (visited on 09/24/2024).

[8] P. Goel and A. Ganatra. "Handwritten Gujarati Numerals Classification Based on Deep Convolution Neural Networks Using Transfer Learning Scenarios". In: *IEEE Access* 11 (2023), pp. 20202–20215. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3249787. URL: https://ieeexplore.ieee.org/abstract/document/10054369 (visited on 09/24/2024).

[9] 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.

[10] 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).

[11] H. He and E. A. Garcia. "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (Sept. 2009), pp. 1263–1284. ISSN: 1558-2191. DOI: 10.1109/TKDE.2008.239. URL: https://ieeexplore.ieee.org/document/5128907/?arnumber=5128907 (visited on 09/30/2024).

[12] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

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

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

[15] S. Ioffe. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

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

[17] 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).

[18] 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.

[19] 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].

[20] 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).

[21] 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).

[22] 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).

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

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

[25] 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.

[26] 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).

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

[28] 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).

[29] 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).

[30] A. F. Rizky, N. Yudistira, and E. Santoso. *Text Recognition on Images Using Pre-Trained CNN*. Feb. 10, 2023. DOI: 10.48550/arXiv.2302.05105. arXiv: 2302.05105 [cs]. URL: http://arxiv.org/abs/2302.05105 (visited on 09/24/2024). Pre-published.

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

[32] 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.

[33] 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).

[34] A. Shawon, M. Jamil-Ur Rahman, F. Mahmud, and M. Arefin Zaman. "Bangla Handwritten Digit Recognition Using Deep CNN for Large and Unbiased Dataset". In: *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*. 2018, pp. 1–6. DOI: 10.1109/ICBSLP.2018.8554900.

[35] 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).

[36] 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).

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

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

[39] 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).

[40] 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.

[41] 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).

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention Is All You Need". In: ().

[43] 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).

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

[45] 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.

[46] 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.

[47] G. Zhao, W. Wang, X. Wang, X. Bao, H. Li, and M. Liu. "Incremental Recognition of Multi-Style Tibetan Character Based on Transfer Learning". In: *IEEE Access* 12 (2024), pp. 44190–44206. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3381039. URL: https://ieeexplore.ieee.org/document/10478003/?arnumber=10478003 (visited on 09/24/2024).

[48] 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.

[49] H. Zunair, N. Mohammed, and S. Momen. "Unconventional Wisdom: A New Transfer Learning Approach Applied to Bengali Numeral Classification". In: *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*. 2018 International Conference on Bangla Speech and Language Processing (ICBSLP). Sept. 2018, pp. 1–6. DOI: 10.1109/ICBSLP.2018.8554435. URL: https://ieeexplore.ieee.org/abstract/document/8554435 (visited on 09/24/2024).