Contents

1 Abstract **2**

2 Introduction **2**

3 Fundamentals on paleoecology **3**
- 3.1 Dental ecometrics . . . 3
- 3.2 Composition of mammal teeth . . . 3

4 Deep Neural Networks for Optical Character Recognition **5**
- 4.1 Deep Neural Networks . . . 5
- 4.2 Training neural networks . . . 5
  - 4.2.1 Loss functions . . . 6
  - 4.2.2 Evaluating model performance . . . 6
- 4.3 Architectures . . . 7
  - 4.3.1 Convolutional layers . . . 7
- 4.4 Residual connections . . . 8
  - 4.4.1 Transformers . . . 8
  - 4.4.2 Autoencoders . . . 9
- 4.5 Techniques and heuristics for improving performance . . . 10
- 4.6 Transfer learning . . . 10
  - 4.6.1 Foundation models . . . 10

5 Related work **11**
- 5.1 Approaches to digitization of handwritten fossil data . . . 11
- 5.2 Approaches to character recognition with small target domain datasets . . . 11

6 Experimental setup **13**
- 6.1 Problem formulation . . . 13
  - 6.1.1 Sequence learning or character classification . . . 14
  - 6.1.2 Tooth marking classification: multi-label classification or classifier chaining . 14
  - 6.1.3 The proposed pipeline . . . 15
- 6.2 Building the models . . . 15
  - 6.2.1 Creating the training dataset . . . 15
  - 6.2.2 Data preprocessing . . . 16
  - 6.2.3 Data augmentation . . . 16
  - 6.2.4 Base model selection . . . 17
  - 6.2.5 Transfer method selection . . . 17
  - 6.2.6 Evaluation metric selection . . . 17
  - 6.2.7 Hyperparameter selection . . . 17

7 Results and discussion **18**

1 Abstract

Keywords: Optical character recognition, Few-shot transfer learning, Vision transformers, Paleontological databases

2 Introduction

broad research area: paleontology: data analysis on fossil finds dig fossil from the ground. identify:which bone,species,time. write this on a field slip, a slice of baking sheet like paper analysis: take set of fossils, use methods for deducing eg climate, habitat, vegetation why this is interesting reactions of ecosystems to climate change what ancient worlds were like how ancient humans lived mass extinction events

now we have a stack of baking sheets in a room in kenya. paleontologists form all over the world want to solve climate change, among other problems big data methods would explode what paleoecology can do so we need to put the baking sheets on the computer to do analysis on big data what is the topic of my work? the baking sheets contain weird characters that a normal reader cannot read. my topic is to read them

given scanned images and data with bounding boxes of sentences and words where tooth denoting words are badly read, how can tooth element recognition results be improved? Goal is to have both this is what the element / nature of specimen column says (eg example here) and what/which teeth are found in this specimen in standardized format (eg example here)

new for intro

motivating why this is relevant, wsj article [24]

limitations: compute (no HPC utilized), small amount of labeled data, scope is masters thesis so very advanced techniques are not feasible.

relevance of this work: KNM is able to have way more precise dental element markings to other catalogues: previous project + this a complete solution to digitizing the handwritten data relevance of this work: any field that does: - ocr on unconventional characters - ocr where each character has a multivariate output (eg. this is an a. it is underlined could be letter and underlined /not underlined)

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 archive 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

paleoecology is data analysis on fossil data most common application: paleoenvironmental reconstruction definition of the problem: whan ancient habitats were like and what changes they underwent at which times (ch2) basic process: learn fossil data to environment statistical relationship in present, apply to past

we need correlations but even those are hard: something correlates in present might not correlate in past Nature is highly complicated -¿ models and assumptions enable drawing conclusions from animal communities

analysis results: how animal communities react to environmental change -¿ get information of what is to come with climate change (faith ch2) dental data is especially useful, whole area of dental ecometrics presented below after that, mammal teeth row is presented to introduce terminology present in the data.

3.1 Dental ecometrics

tolerance = range of an environmental variable that is hospitable for the species [7] a mapping from taxa to environmental variables -¿ know fossil assemblage, deduce environment

modern alternative to this: transfer functions: mappings from taxa data to enviroment learned using machine learning / statistical models (ch9, birks 1995) benefit instead of tolerances/niches: they have subjective interpretation problems (book ch 9) esp. teeth: dental ecometrics = inference of transfer functions given dental data (ch9 liu et al 2012), [22]

taxon free: dental ecometrics relation of traits of animals and livelihood = ecomorphology [22]: teeth -¿ diet -¿ what plants grew and habitat -¿ climate etc example: dental lophs and hypsodonty to temperature and precipitation, from [22] two kinds of plants, browse (dry), grass (moist). dominant molar, usually 2nd, primary chewer tooth grass -¿ high-crowned ie hypsodont molars browse -¿ dry plants need more blade-like molars ie more pronounced lophs then you take the teeth, check hypsodontry and lophs of the species present, check occurences of species in regions statistical model (linear regression relationship) between teeth and present environment use mapping to fossil teeth in the past -¿ get past temperature variables. [22] for instance got temperature

-¿ sample sizes and data precision are important

3.2 Composition of mammal teeth

Fossils occur when animal / plant remains are deposited in a sediment in a way that preserves some part of its original form. Since teeth are the hardest material in animals, large fraction of found parts are teeth. Fossil finding is followed by identification to most specific taxon possible largely a technical skill (ch5), teeth are identified down to type and number, how manyeth the teeth are, counting from center to edge or other way round?? specimen can be either one tooth or fragments of the jaw bone where there are multiple teeth (markings like M1-3) from [13] what teeth are composed of

the jaw bones lower jaw bones: mandibles permanent and deciduous (D), nonpermanent "milk" teeth (laita vaan jos löytyy d-hampaita)

right and left sides are always symmetrical, denoted simply L or R or Lt or Rt or left or right. left is left looking from the animal, not the observers perspective Identicality also causes that sometimes tooth fossils are misidentified to the wrong side and corrected (ei lähteestä vaan nähty datasta koska l ja r on sutattu aika monta kertaa ja vaihettu

four classes, front to back: three incisors (I), one canine (C), four premolars (P), three molars (M). top bottom left right. top/bottom noting upper jaw as superscript lower jaw as lower script, purpose: incisor -¿ catching, canine -¿ stabbing / killing prey, molars are for chewing. premolars are bit like canines bit like molars, function varies lot between taxa including holding, cutting and chewing. also form and number of each present changes between taxa. sometimes lower jaw as line on top and upper jaw as line on bottom, sometimes both are used: upper script number with line on bottom. Line is "the other jaw" if there are less of a type of teeth eg two premolars, they might be no 1 and 2 or no 3 and 4

this chapter presented mammal teeth terminology and a quick overview of paleo analysis done on dental data

As the models in paleoecological analysis are mostly fairly simple, the hardest practical part of paleoecology is to construct a sufficiently large representation of the functional traits from the past environment. Therefore, paleoecology is, in my novice opinion, most severely limited by aspects of data management. Imagine having a database, with all fossils globally. All relevant traits could be saved as columns, and the specimen could be 3D imaged along these traits. Running a paleoecological experiment would only require designing the experiment, a database query, and fitting the model. In this way, experimentation could be iterated much faster, and most work time of experts in paleontology would be spent considering analytical, rather than practical problems. From a technical perspective, constructing an all-encompassing database is not a particularly challenging problem. note here: given current data science methods it is not easy but methods for this definitely already exist Where the challenge lies more is in intercultural collaboration, standardization of fossil data representation, and if the problem receives attention.

The goal of this thesis is not to construct such an ideal database, as it could be seen as the ultimate goal of paleontological databases. Neither is the primary aim to uniformize the tooth notation in the catalogues of the National Museum of Kenya specifically, although it is the main output of this work. The goal of this work is to speculate and test computational methods that would efficiently uniformize dental fossil data in general. Thus, how this work relates to dental ecometrics is that I aim to come up with methods that would allow a tiny step toward solving the ultimate problem of collecting all data in easily accessible format. My humble wish is that this would accelerate ecometric experimentation, saving the cognitive capacity of experts to solving more sophisticated, analytical problems.

This thesis is not aiming to build

This aim, that could be seen as the ultimate goal of paleontological databases, is not the aim of this thesis, but the point work aims to take a tiny step towards.

(also why aim for any less like as a humanity though, wasting time battling with datasets is wasted lifetime) (not the aim of this thesis but a nice big grand goal of paleontological databases) like what we should imo aim for as a humanity my goal is to speculate and test computational methods that would uniformize fossil data my goal is also not just clean kenyan teeth tiny step toward the ultimate goal

4 Deep Neural Networks for Optical Character Recognition

separate problems: character classification (easy, kNN, SVM), reading variable-length text (harder) [21] introduce the problem of ocr, example: fossil catalogue

4.1 Deep Neural Networks

- neurons and activation functions. maybe examples of activation functions: relu, sigmoid, softmax

what is a neural net weights in layers: floating point numbers, grouped in groups activations: connections between weights, nonlinear scalar to scalar functions

- feed forward network computes output from input with the feed forward. you have an input, bunch of numbers then, you compute a linear combination and pass that through an activation function

$$h_d = a\left[\theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} h_{d-1}\right] \tag{1}$$

hd, hd-1 are hidden unit values. d is layer index. thetas are weights D is width of previous layer [25]

a is any nonlinear funtion, simplest is the rectified linear unit relu, x=x if x¿0, x=0 if x¡0. hidden units are taken as inputs to next layer and next and next. thetas = weights largest, deepest: hundreds of layers, hundreds of millions of weights then last hidden unit output it the output of the network. tadaa.

- universal function approximator the theory of the universal approximation capacity: this algebraic construct, given correct weights, activation functions and structure, could approximate any mapping from input to output. note: input/output dimension can be anything

examples from ocr relevant in this case input is always image ie 2d matrix if grayscale or 3d tensor if rgb image.

classification problem, softmax activation image of tooth sample (letter+number) to four classes, M P I or C. input: image, output: four probabilities that sum to 1, probability this is a m p i or c. sum to 1 is achieved with softmax. output is largest probability.

multilabel classification problem: image of tooth sample. output: first MIPC, second if it is upper or lower jaw output can be two arrays, one like before, other a 0-1 probability for upper jaw. upper if this ¿ 0.5

present sequence to sequence learning: more complex case: image of sentence output: text on image, variable length. here output layer is a more complex structure of probabilities for each position in the result sequence.

insert image example: inputs and outputs of different learnable functions with neural networks

NEW: universal approximation theorem is tighter! it states one layer fc network should be able to approximate any function, so no need even to have the optimal architecture. architectures are for easing practical computation matters

4.2 Training neural networks

u have encoded the structure, starting weights. data, input/output pairs. training = process that adjusts weights so that network becomes a good approximator

1. take part of data as training data 3 divide to batches, common batch sizes are exponents of 2, 2-32 usually 4. pass a batch through the network. get output 5. pass output and correct to loss function: map from output, correct to scalar, 0 is good, large value bad. 6. Compute loss function gradient with respect to network weights using automatic differentiation. algorithm to get this in code is called backpropagation because it moves backward in the network 7. gradient informs how to adjust weights so that loss should decrease. adjust weights in this direction by preset amount called learning rate. Optimizers are algorithms that determine the specifics of "moving in the direction of decreasing loss" most common stochastic gradient descent (inserts randomness to movement) and adam (uses previous iteration movements known as momentum in process to make movements more smooth) 8. run batches until out of data = epoch. run many epochs, stop according to stopping condition that is known to be a state when the network weights are good. goal: reach global minimum of loss function, difficult! would be perfect approximation 8. test on unseen test data to see generalization performance

lots of details on this process, rest is about that

4.2.1 Loss functions

more specific: how do you map output and label to scalar value describing how good the output was? ocr point of view

Loss function is a function from model predictions and ground truth labels that describes with a single scalar value how good the match was, low number describing a good match [25]. These functions are constructed to be equivalent with maximum likelihood solution, think the model would output a conditional distribution of outputs, p(y—x). each ground truth label in the training set should have a high probability in this distribution. Product of all these probabilities is called likelihood. Find parameters that maximize the likelihood of the training data set. Loss functions are derived so that parameters bringing loss to zero is equivalent to the parameters with maximum likelihood. Derivations are out of scope.

- cross-entropy loss kullback-leibler divergence of correct conditional probability and conditional probability parametrized by current model parameters. (show formula, 5.27), correct is not dependent on parameters so is omitted. show 5.29, what is left from that (until here from [25])

then: how cross-entropy loss is computed used for classification problems. eg. is this letter in this image an 'a' or a 'b'. correct probabilities eg .1 and .9 for a or b. model says .2 and .8. discretisized cross entropy computes it as .2*log.1+.8*log.9, log in base 2.

word detection models: have a predefined vocabulary, layer for probability of each word. loss is cross entropy for these probabilities compared to target probability distribution, where correct word has probability 1 and all other have probability 0

- CTC loss

maybe

4.2.2 Evaluating model performance

common in character recognition: accuracy, f1 sometimes also used explain these and differences

accuracy: correct out of those classified

why one uses precision and recall instead of accuracy sometimes simple accuracy is not a good measure, eg cancer screenings: 1 in 500 has cancer. then a dummy saying everyone is healthy would be correct 499/500 of the time, accuracy 99,8%. measurements anyhow relevant would be 99.8-100, inconvenient range.

precision: fraction of positives identified. dummy would have precision 0 recall: out of those noted as positives, fraction of correct ones. dummy would have 0 too f1 is an average to lump these together in one number

going forward: only consider simple accuracy scores as none of the classes are very rare in the fossil case.

4.3 Architectures

different ways of constructing layers, makes model pay more attention to desired things and reduces parameter count from fully connected layers, ie. encode priors [16]

relevant here: present the ImageNet competition that has initiated many new architectures.

4.3.1 Convolutional layers

encode a prior reduces the need to learn parameters. limit is cpu resources and data so given data and cpu, get as good model as you can requires constraining the problem by making assumptions [16]

prior encoded: move image a bit and it is still an image of the same object (translation invariance) and nearby pixels are usually like each other (have a statistical relationship) fully connected nets do not consider input value positions in input vector, how near or far from each other they are, in any way. [25]

practical use: less parameters required -¿ less computational complexity

convolution (cross-correlation) you have input and kernel. input: image matrix (2d) kernel: 2n+1 sized matrix (uneven to make it center around the input pixel processed) kernel slides like a sliding window through the input. for each middle position of the kernel: result is the dot product of the pixels in equal positions, eq 10.6 if the image is rgb, kernel is 3d with depth 3. (bike fig 10.10 here)

aspects: borders, channels, convolution size, dilation and stride default variant reduces size because border pixels cannot be used to combat that: zero or reflective, padding to keep output and input sizes equal, or valid convolution: dont compute positions where there are not enough previous inputs to use the whole kernel channels: run many convolutions in parallel -¿ more convolution kernel weights to learn size stride and dilation, figure 10.3 size = kernel size stride = step between kernel computation dilation = kernel is interleaved with zeros, large region but less weights stride dilation size fig here

usually conv layer followed by pooling layer pooling usually networks are constructed so that layer by layer layer width decreases and has more depth (more channels, like red green and blue but not with this meaning) changing size more than by the few pixels present at image edges: max pooling. take maximum of 2x2 area, collect these values as the output activations. other, less popular variants: mean and average pooling

historical scetch: alexnet [16] brought this to mainstream in 2012, imagenet 2014 saw GoogleLeNet [32] and VGG [31]. these highlighted because the architectures are utilized in handwritten character classification. alexnet, top5 error: 17% best so far whats new regularization with dropout, large model, large conv kernels (11x11-3x3 kernels), relu instead of previously popular tanhs sped up training, multiple GPUs used when training, normalization of convolution results, overlapping pooling (areas of pooling for each pixel overlap) impact: brought deep learning to center of ml research, deep nets seem to outperform humans in feature engineering.

googlelenet [32] won 2014 with top 5 error rate 6,75%. method: fixed number of allowed multiple-adds in forward pass, used inception (named after we need to go deeper meme) layer: 1x1 3x3 and 5x5 sparse convolutional kernels, also sparse fully connected layers ie not all weights are connected to save compute. Also inspired by biological visual systems. allowed for deeper and wider network, turned out to be great in imagenet and object detection.

vgg [31]. achieved 6,8% with imagenet 2014 postsubmission, submitted 7,3% thus second in competition. method: simple 3x3 convolutions, tested different depths. best depth 19. new that previous best models did big kernels and shallower nets. also good model because of the simplicity + the paper included in appendix that demonstrated vgg as a feature extractor to be used in transfer learning.

4.4 Residual connections

problem? when nn is deeper training becomes harder how/why? vanishing exploding gradients is solved by normalization in initialization and between layers. oma: the weight values are initialized / updated to values that are not super small or super large so gradients wont vanish another problem: degradation of accuracy. accuracy reaces a peak point and then starts to decrease so by adding residual connection (give formula updated for layer computation) the deep network has a shallower subnetwork so the result can never have worse training error than its shallower versions layer computation noted activation=F(x), computation becomes F(x)+x. +x is a connection that skips a layer. (add fig 2 from he et al) how the problems relate: degradation problem hypotethized to be a consequence of stacs of nonlinearities being bad at learning identity mappings

solution/benefit? aids training deeper networks 152 layers, 8x more than was previously considered 'deep' depth allows for better accuracy learning the identity mapping is easier as additional clue why this is good, not primary motivation as many state

why solution works? won essentially every competition it participated in

flow of thought there was the inspo that it might be that being better at approximating arbitrary functions ie building accurate nns for anything would just be about stacking layers the vanishing exploding gradients problem update a weight parameter w lr*gradient, if gradient is huge you shift parameter a lot, numerically unstable training. if gradient goes to zero, the weight can no longer change this was already solved with input and batch normalization layers: gradients dont get stuck to some specific values another problem: degradation of accuracy deepen deepen the network -¿ peak accuracy then it reduces so there is something dysfunctional about the standard computation that makes training deep hard

solution, he et al if you add identity connection (kinda any layer can be used as the output), there are shallower subnetworks in the deeper network -¿ accuracy can never be worse than in the shallower networks. quick formula and figure here turned out to be much better than in the shallower networks hypothesized: the deeper the network the harder it is to learn the identity mapping this is the cause of the degradation problem and bc residual connection solves it, degradation problem vanishes and indeed the resulting network won many competitions including imagenet challenge with a net 8x deeper than previous deepst, the VGG

4.4.1 Transformers

words reference to one another that are sometimes far far away in the sequence, conv pays attention only close up with the filter, what is relevant depends on a this word b the other word large embedding vectors -¿ fc layer won't do really, you have to share parameters search engine analogue for attention embed again a prior by embedding how words relating to each other works. search engine analogue. query is what we want to find out in the sequence (search query) key is a descriptor of the thing (page content for a simple engine) value is the search result (page contents ranked)

similarity can be computed with cosine similarity that measures vector direction difference, vec1 dot prod vec2 divided by magnitude (length) vec1 times magnitude (length)vec 2. (give equation for cosine similarity) attention with the text sequence between words so self attention transformer the query key and value is all the same because we (a sequence and its parts) are paying attention to ourselves (he same sequence), so same embedding multiplied with key query value weight vectors to get key matrix k query matrix q value matrix v so then you have the query the key and the value query and key is computed cosine similarity qkt/sqrt(dk), scaling factor not quite cosine similarity but similar (lol). you get attention: how much each word pays attention to other words softmax to put all attention values between 0 and 1 second vec needs transpose to avoid dimension errors (matmul takes col from one row from other and we want col from both so you have the original sequence and how much attention each word pays to each word (attention filter). multiply original by attention you get the original weighted by how much attention it is sensible to pay between parts so this is one head

to be able to pay attention to multiple things at a time you construct multiple attention filters, so do that many times in parallel -¿ multi-head self attention multihead: divide query and key embeddings by splitting embedding matrix, do many self-attention oeprations on those in parallel, concateneate result

this is the multihead self attention block. here you go a summarizing figure

transformer architecture uses this and fc layers like so transformer is the first architecture to only utilize the attention mechanism (no other layers but multihead self attention and fc layers) transformer has an encoder and decoder part (so is a variant of an autoencoder) transformer block: multihead self attention and fc layer, both have batch norm after and both have residual connection teach the network to continue sentence with the next word, selfsupervised eg hide last word

tadaa you get a model that can continue the prompt with valid text, add some features and you have a LLM

vision application: divide the image to 16x16 patches, interpret those as ”words”. a transformer architecture that has as output the class probabilities for imagenet. good results but lots of training because transformer does not encode the vision priors of translation and pointwise function invariance toward transformers

4.4.2 Autoencoders

most of this from [9]

labeling costs -¿ unsupervised learning 1. labeling cost eli unsupervised or selfsupervised, which is learning without any labels. x is input and output is latent variable of x, z. if input is an image z is obtained from x somehow eg add noise to image, add 1 to all intensity values, etc. no matter if it makes sense just that there is an algo for getting the latent variable.

what is an autoencoder 2. autoencoder=input x latent is also x. normal net: just copy. eg shallow network no hidden units: output at poistion i is input at position j, no learning trick: constrain variants: less hidden units than input (undercomplete autoencoders), regularizing encourages smaller parameters or sparsity most basic: input to hidden (encoder) hidden to input (decoder)

3. why this is useful / interesting? encoder and decoder nets are mini nets in a net that you take

out and use for interesting things. - variational autoencoders: learn data prob distro, generate new samples - dimensionality reduction/feature engineering by getting the hidden unit - use features also in character recognition: eg shoponbangla

4.5 Techniques and heuristics for improving performance

data augmentation

dropout regularization (alexnet uses so need to intruduce)

4.6 Transfer learning

relation to the coarse level training loop: how to initialize model parameters this sect from [23]. they initially formalized the problem and uniformized the terminology

basics: what it is, basic premise: if you start out from a parameter configuration that solves a related task well, there is lesser need to train to solve the new problem [23]. why: save compute (imagenet models vgg lenet alexnet training times 5 days to 3 weeks even when using GPUs [31]) and data labeling (usual constraints in ml model building [15]) also why: training many parameters on a dataset overfits model to the dataset, so freezing layers prevents overfitting [32], optimization idea: using pretraining that makes sense, you start from a parameter configuration in a 'basin of attraction' of a good minimum [6], ie a place where gradients point toward a good local minimum training is strongly influenced by early examples so pretraining prevents overfitting to the supervised task get sgd trapped in a parameter space that is better so resulst is better even when target data is abundant [6]

inductive transfer, transductive transfer, instance transfer, relational knowledge transfer task (inductive) transfer: tasks differ, data same or different task transfer relatedness + the more related the better results negative transfer ie things made worse if things differ too much. super important to consider the distance of transfer ie how similar the tasks are broad methods for inductive transfer: feature representation transfer, parameter transfer; priors or hyperparameters of model are assumed to be shared, instance transfer: reuse source data domain (transductive) transfer (only data differs) central idea in all of these: model first n-1 layers is a feature extractor seeing model as model doing the feature engineering: all but last layer are like feature extraction that allows a linear model to be fit from features to output. first layers learn lower level features, higher up more high level [6], so how much to freeze is about how high level features are usable relational-knowledge-transfer: when data is not i.i.d (not considered here)

my task. source task is image classification with different data than mine inductive transfer where data and task differ. i will test both only feature representation transfer (re-search for optimal hyperparameters) and parameter transfer (use hyperparameters used in source model training) different countries mark tooth fossil in different notation, so fine-tune to adjust

basically nowadays it is usually insensible to ever train from scratch [17] therefore this work definitely transfer learns. also previous work that does not transfer learn is dismissed, as it does not inform this case where target data is scarce

4.6.1 Foundation models

Idea: since transfer learning source task performance is not important, target task is the main point [23], you can come up with some nonsensical task such as map image to the same image to get unsupervised task, train massively on massive compute, and use the model for varieties of tasks.

idea: do as little as possible in the target domain because labeling is the expensive thing [15], this helps because unsupervised or selfsupervised allows however many labeled samples one wants.

come up with task with no value in itself but learns useful features. eg identity mapping in autoencoders, mask a part of an image and fill it in, get in a pair of images and say whether they are a transformation of one another. save the pretrained model, can be used with transfer learning in manymany tasks [6]: why unsupervised pretraining works so well analytical analysis: seems to act as a regularizer generalization is better: intuition, model has seen a much larger variety of data

well this is used in ocr as well. so you would take some model trained on a large mass of data and then fine tune it on the characters

5 Related work

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

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

5.1 Approaches to digitization of handwritten fossil data

[35] reviews various digital paleo data portals. encourages further digitization but does not really mention how one should go about doing that

[20] review of digitizing fossil data. only considers digitizing physical bone pieces to various 3d images. does not take any stand on digitizing handwritten notes on the samples

[10] does identify my problem here: most fossil data is in handwritten paper format with many things already known such as taxon. gives suggestions on data format and management but mostly assuming that there is a human transcriber. presents ocr reading as a possibly one day possible option. closest to automation known in this work was making imaging of physical samples more efficient with a conveyor belt system, which has nothing to do with handwriting ocr. Notes that especially cleaning of automated ocr read data is a hard problem.

only work doing the exact same thing: [28], but quality of the work is very poor, only presents rudimentary very basic general aspects of image processing such as canny edge detection or contour detection from characters, which is very very far away from working ocr. the paper lists absolutely no results but is an evidence that someone has somewhere at least attempted this.

So: to the best of my knowledge there is no other work successfully reading handwritten fossil scans or cleaning already automatically read systems outside of 2024 spring data science project for KNM, for which this is the continuation. therefore also here i will do the simplest things first, since it is a good idea to start with simple and once successful, proceed to harder problems that would digitize and clean more data.

5.2 Approaches to character recognition with small target domain datasets

- since there is no old work on fossil i consider other cases and cases that are on abstract level similar

why sanskrit based Indian languages are a good inspo field for KNM

sanskrit based languages are: gujarati devangari, bangla, kannada, urdu [19] small languages lots of transfer there because languages are small -¿ small datasets -¿ transfer needed [19] major languages have sufficient data to train from scratch (and haven't realized transfer can help even then? [26] [37] most relevant is how many samples are available per class but for knm, asian characters like chinese / korean / kanji could be more applicable because eg kanji contain subcharacter sections (radicals) that have distinct meaning. knm 'radicals': letter number underline.

modifiers modifiers like knm has the top and low line in: gujarati [19] bengali [3] urdu [26]

cursive [26] (where it is hard to do connected components based image processing)

character sizes vary in bangla [30] also knm with small numbers

general applicability: bounding box error setting also did first bounding boxes (with way easier case tho)-¿ similar part of character cut off / too much background problems maybe [1] [27]

why ancient things could be from [34] more than one character per writing unit -¿ knm also has letter and number images + scans are more difficult to read because of old 'paper' -¿ focus on preprocessing -¿ similar to knm (but harder than knm probs) also pages where you extract characters -¿ maybe similar ish bounding box extraction mistakes huge amount of classes so hardest problem

(be vague here: not 'no xxx done', but 'little xxx done'

papers over all: little whys or careful critical analysis for decisions done is given. especially transfer distance mentioned in 2 papers of 11, and is most important in getting transfer to work -¿ no/little reasoning on base model or transfer method selection. also, unusual choices should have a why but mostly don't

what counts to me as a proper why: why this decision was the best option among all available alternatives. many had a 'this is a good approach' or generic strengths of some approach. that is not enough since it does not take a stand on what about the alternatives, why not them

The reasons and considerations regarding setup decisions are not nice-to-haves, but have quite some consequences for the quality of the research conclusions. -¿ i cannot evaluate the reasoning behind the choice -¿ sometimes choices that clearly don't make sense are done such as flipping character images for augmentation or probably don't make sense such as color flipping when imagenet is source task (increases transfer distance, real world image backgrounds are usually light not dark, would decrease in mnist thats why the color flipping is a common approach but imagenet != mnist) -¿ train test setups that dont make sense from generalization perspective. optimizing for best rotation angle (real world images only rotated by specific angle???), augmenting the test set (then test set is not representative of the real world) -¿ irrelevant specifications are given (eg hardware when no timing is done) and relevant specifications are missing (how many layers you froze) when analytical basis is fragile -¿ irrelevant/less relevant metrics are measured (inference time when it's not a real time application, false positive true positive rates when ocr really does not have a clear definition of a positive vs negative as its not binary classification) -¿ reasoning is frayed (such as we compensate our worse accuracy with that we train with less epochs, when training in transfer learning completes even on a cpu in order of minutes so epochs does not really matter that much)

final consequence —¿ if we don't have a clear analytical reason why unexperimented with alternatives would work worse, one cannot assume they would be worse -¿ one cannot fully trust that the approaches presented are the best.

always imagenet so i will do the same. what is the most common solutions imagenet cnn + transfer what are unusual solutions given my prior knowledge solutions that don't make sense (freezing a layer below an unfrozen layer) at least this solution lacks a feature representation related or analytical explanation why it works less common ml approaches unsupervised pretraining autoencoder skipped to scope down as it is not that popular of an approach transformer based models but I know transformers outperform cnns on imagenet tasks, therefore I should test transformers as well layer freezing variants in papers: so I test:

6 Experimental setup

now we consider the fossil case again. my goal in this work is xxx.

the hypothesis: doing the pipeline approach and classifying teeth to types with a series of models will result in highly accurate cleaned tooth data verify that this is possible experiment with base models and layer freezing to find best accuracy score overview of chapter sect 6.1 formulate the problem ie how to interpret the ambiguous goal of ”clean element description tooth markings” set of x to y mappings that can be taught to a deep neural network and how to build a system from the mappings that achieves the goal sect 6.2 the training settings attempted informed by literature for creating the required models is chosen

6.1 Problem formulation

point of this section: what data we are talking about, what has been done, what is my goal on high level subsections formalize the high level problem to a problem that can be solved with a nn

Goal of this work is ....

data and what has been done until now what the catalogue is exactly: tabular handwritten log of fossils element description column (which part of the animal) handwritten notation from time before standardized computers, so very variable writing with no notation standard -¿ rare surprise characters occur (eg 1/2 as a one half of a tooth noter) old note on this: has been done by different but few annotators, no logs on who logged what, everyone had a bit different style of notating. also no clearly defined standard for notating specimens. so might be that actual data used will have characters or words not present in training set photo-to-tabular has been done but: teeth have errors in ocr output (eg tooth types that are not m p i or c) upper lower jaw is unknown since ocr used in table (azure) cannot read up/low jaw notation

what i am doing now with the catalogue is to take an element description (give sample image). of this description, i should find what teeth are mentioned, and list them as a tuple in another column. (give what teeth there are) and put the cleaned teeth to the description instead of the not clean word to aid in this i have also the generalist OCR ouputs from azure why standardize you can eg query in a database for all molars in the data if you for example want to study dental traits present in molars only, such as tooth surface wear.

so: we have the vague goal of catalogue to cleaned element description with teeth tuple extracted.

general on goal to nn: the generality vs accuracy tradeoff and what i should aim for different types of defining inputs and outputs, discussion on what to choose generality ease trade off: constrain to simple input output relation causes model to be more accurate (it can be given more prior information on what is present), but less applicable to variability in data. example of very constrained setting input: perfect images of teeth letters output: letter example of a very unconstrained setting input: the whole catalogue image output: a tabular dataset of all text there perfectly cleaned no matter the anomalies A balance between the extremes, closer to the easy side why we went for more simple approaches first because the part of data that can be digitized with the tightest constraints and the simplest models are most accurate. Do that first, deal with the remaining later better to clean small subset of data well than large amount of data badly. difference to other domains of big data: each fossil sample is expensive and human laborous to obtain, thus "big" sample is small compared to other ml applications, 1000 is a lot 10k huge. [7] so get a clean sample that is small is still valuable because paleo samples are relatively small also: not much done before so it is better to start out simple and build from there

first question: is the input the whole phrase, words or characters? sect sequence learning. since words are better, section tooth thinks about how to solve image of word to tooth type problem. once we know how i show u how to use the models, how the system will work

6.1.1 Sequence learning or character classification

sequence definition: image to variable-length phrase (text in the image) word per word approach, inspired by [**tibetan˙ocr**]: input is image of one word one tooth = one word output is the text on the image, when the word is a tooth output is valid tooth notation so m1-3 p1-2 etc left and right are not there like lm3 because it is sometimes in other formats like "left mandibular frag with m1-2" character by character approach: input character image, output character we cannot do this because there is no readily made segmentation of image to characters would also not make much sense because tooth notation letter and number should be kept together -¿ choose between sequence and word

sequence benefits: adaptable to many kinds and lengths of input also clean the nontooth word outputs at the same time

sequence bad sides: finetuning just one layer on 80 training images for two epochs took about 15 minutes of [18] -¿ all hyperparameter optimization etc is out of question with this heavy training. finetuning on images with tooth data caused errors on words that (give a sample of confused reading attempts: leftleftleftleft) inductive transfer learning is the case here, target task differs from source task the target set of characters has changed Encoding this to the large encoder decoder transformers would require rewriting parts of the preprocessor and model which is too complex given the level of this work. only adequately accurate models are large collaboration efforts to create so code is complex eg trocr is by Microsoft employees

word by word benefits feasible given available data and computing resources possible to encode the tooth type classes, eg have a class "third molar" classifying characters has been essentially solved, easy problem also classifying to tooth or not tooth should be easy

word by word bad sides not as flexible. but we started with the easier anyway to have something thats working

so, word by word. next: formulate image of word to correct output more precisely

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

approach: tooth or not tooth. then classify teeth. not tooth straightforward: use azure output tooth less straightforward, topic here: x is the image of the marking, y is tooth type, but what is the softmaxed probability vector the model should output? so what are the target classes?

formulation of tooth image to tooth type univariate: m1,m2,m3,p1,p2,p3,p4,c,i1,i2,i3 up and low (22 classes) -: does not encode that all m's share letter m, all ups share traits we have kinda three separate problems: 123, updown, mpic. this does not encode that but states we have one problem with 22 possible, separate solutions with no relations between solutions (all classes are equally dissimilar to each other) −¿ incorrect encoding of prior knowledge alternative multivariate: up/down, MPIC, 1234 input image, output three vectors: updown, mpic, 1234 -: multiclass classification general case does not work like this general case is like image can have a dog and a cat, but here an image cannot be m and i [36] so this is special grouped classification -¿ less prior work as problem is more unusual -: no such thing as a 3rd canine so not all variable combinations are possible. no easy well established way to encode that to the model what will we do if the model thinks its a 4th canine? -: even when it was easy [**tibetan˙ocr**] kinda already argued the multilabel is a bad approach

model ensemble/ chained classifiers to the rescue separate models. mpic model, updown model, m1m2m3 model, i1i2i3 model, p1p2p3p4 model simplify the problem of each model encode the fact that equal likelihood for M to be lower or P to be lower problem is more alike the literature reviewed separate index model to encode the no 4th molar prior model does not learn index number based on what the letter is by keeping letter not changing

not inspired from any paper but my own idea, felt like a sensible way to formulate the problem, so we have for all models a well-solved well-defined basic classification problem. partly inspired by [**tibetan˙ocr**], though

reason for this choice: it encodes most prior knowledge in the output structure, use as much from azure as possible act according to premise of transfer learning: utilize all prior knowledge, fine tune your approach to problem as little as possible

6.1.3 The proposed pipeline

chain of models, inspired by [**tibetan˙ocr**]: take in image, tooth or not (tibetan: classify which script style, then recognize so classifier chaining) if tooth give to tooth classifier if not tooth give to basic word reader

summary: the pipeline split catalogue to word segments by azure bounding boxes classify each word: tooth notation or not? if not tooth output azure output if tooth give to m p i c model give to upper lower model if m give to m1 m2 m3 model if p give to p1 p2 p3 p4 model if i give to i1 i2 i3 model return index type upperlower combined note: left right ignored for now as it is more complex (l lt left, not always right before word) take from all models min confidence of the top class pick and use that as confidence notation for tooth classification. save it then collect all teeth to tuple confidence of teeth tuple is the minimum confidence among all tooth. threshold: if below x, mark the row for manual inspection, otherwise dont inspect (esim human level recognition like 99,7% or smthn) then concatenate description back

NEW how should i note unknown jaw in inference? mention false positives in tooth/other (other that are recognized as teeth) confidence system should output low class probabilities -¿ confidence threshold also avoids false positives

6.2 Building the models

so now we know we need these models: restate models. now we choose what to try when building them list subsubsections, these are the aspects

6.2.1 Creating the training dataset

how i got the data

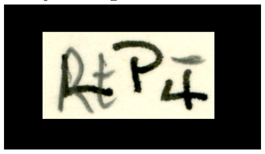
Data was extracted from scans by getting bounding boxes from Azure Vision API, finding the correct column (nature of specimen or element), and cropping the image according to bounding boxes. regular expressioning, mention () , L R

multiple catalogues used to create training set to get versatile notation train and val from same catalogues, test set from new one to hopefully introduce concept drift train on train, model select on val, report accuracy on test set -¿ true not best but more so worse case generalization accuracy, get that high u real good

labeling: mpi, index number: assume that azure reading is correct where reading is a valid tooth notation if incorrect, leave out label and handlabel (dont throw out since these are the hard cases and are valuable!) up low: hand label with binary label 0 for low 1 for up, -1 for undeciberable why an undeciberable class: - undecipherable upper/lower: the correct answer is not any of the two, so false or correct classification is equally bad, so accuracy does not fully reflect if the model works perfectly or not. - forcing model to think something a human cannot classify should contain some traits where it could figure which one it is is false and will confuse the model -¿ worse results

hard cases Some have been corrected by writing on top and thus are very hard even for humans to read, this is also an example of correction, line on top with different intensity of the line:

NEW: sample amount taken was limited by how labeling, about 1000 felt sensible extraction by getting all words from all over catalogue w regex, inspect dir and delete those that are not teeth (usually 1-5 pcs in 100 samples, many had w k or r as the letter idk why)



if it was completely impossible for mpic or index number, was omitted. these were veryvery rare, a few instances out of hundreds of em

6.2.2 Data preprocessing

data preprocessing

grayscaling (rgb to gray, stack 3 on top to match input size. remove rgb so model does not try to deduce something based on color, it should not) smoothing? w gaussian filter?

5.11 screenshot: see image has this noise where the scanner distinguishes on the paper some kind of noise, it should definitely be removed

but not too much preprocessing: it makes the image look less like a real world image so it could increase transfer distance, but then again there was the some paper that said binarizing would be great apparently also sometimes pencil has been erased so there is pencil eraser noise

segmenting out the letter: does not make sense since the premise of deep learning is to to feature extraction better than a human would do. to segment out the letter you would need to know which area is the letter and to know that you need to know what letter there is which is the problem in the first place, so unsolvable egg hen problem. show some example from preprocessing notebook

no binarizing: some letters are super weak text so theyd get lost and some background clutter might be dark. popular was wan thresholding but im not confident in this case where intensities change so much you could do any deterministic binarization

6.2.3 Data augmentation

how i will augment: those that might happen in scanning process intensity value change (idk scanner illuminates differently) change background and letter contrast (some paper is lighter than other) random cropping (to mimic bounding box detection errors, dont translate bc that creates the white background that never occur in test data) minor rotation (up to 5 degrees, scanner does not turn that much and handwriting is generally pretty straight) blurring filter (to mimic pen vs pencil) data balancing by saving some augmented images, otherwise random aug in experiments: accuracy up by 10% when balancing classes in input data, as predicted in [11]

6.2.4 Base model selection

Chosen public dataset was EMNIST [4] because it is the closest to my problem: a large dataset of labeled letters and numbers.

According to [2], this model was the best: [29], was chosen as the base model. they trained on emnist-balanced, so no difference between upper -and lower case letters. not a problem for us since in the handwritten catalogues, upper or lowercase was not used to distinguish upper and lower jaw

mnist and real world fundamental difference: mnist is perfect world, real images vary much more [16] therefore: not e(mnist) as source problem, but imagenet classification.

return to tables 3,4 which base models seemed to work in literature those variants that occur many times in top3 efficientnet vgg16 densenet121 resnet101 obvious flaw: no whys in prev work why they chose what they chose but one cannot test everything so we choose these.

6.2.5 Transfer method selection

why i only always keep unfrozen layers after frozen layers some papers did some first layer unfreezing, or middle layer unfreezing ie something else but not freezing last layers. dismiss that because they did not state why they did that. from the learn higher and higher level features point of view having an not frozen layer and then after that a frozen one does not make any sense; the later layer is based on the previous. so never in experiments have a frozen layer below a layer that is frozen

why i start from more frozen to less frozen more frozen is less training effort, more generalizable (less is fit to specific data set) this is overall better so therefore first this if the search space of parameters is too small (model converges to low accuracy), freeze one layer by one until overfits: training error is much larger than test error. sign: we trained too many parameters / for too long

6.2.6 Evaluation metric selection

accuracy.

it is noticed that up low is the hardest ml task of these. so select based on uplow performance and use that on the rest of models if not 100% accuracy reconsider your choices

not tp fp things because they dont make sense

not any which classes get confused outside of confusion matrix: there are so few classes anyways

6.2.7 Hyperparameter selection

fix hyperparameters: do not experiment with too much at a time. Take hyperparameters that were most common among reviewed literature.

the hyperparameters

batch size num epochs optimizer learning rate learning rate scheduling loss function train test split check test error granularity many runs and final accuracy averaging; how many runs

For each: list what papers used generally, pick a nice average value then i tested with the best model these to check sensitivity to hyperparameter choice, so robustness of optimization. esp change the random seed to see results dont depend on that but so the hyperparam sensitivity thing: if the experiment results hold across a population -¿ which method works best should work best in all hyperparametrizations and not only in this one. vary hyperparameters and check if comparison (which was better than which) still holds differences in accuracy are due to the hyperparameters being more optimal in some cases than some you can never really eliminate this one should further verify how robust the experiments are to hyperparameter changes I can test around a little bit to check that my some better than some holds at least in most cases

this chapter gave an overview of how the experiments were set up and why they were set up in this way. a sentence from all: sequence not done bc its complex. multiclass not done bc it is worse according to [**tibetan˙ocr**]. training setting: preprocessing by denoising only because nn premise is that the net is better at feature extraction than human. augmentation: intensity change background contrast change random crop letter blur minor rotation because these can happen in catalogue scans base models are: efficientnet vgg16 densenet121 resnet101 because they were popular and or won comparing benchmarks in previous work freezing first more then less bc if more works its better, find sweet spot of most freezing that is still accurate.

7 Results and discussion

start of section: reiterate what was experimented with

see if train/test accuracies are very similar or very different. conclusions from that? overfitting? + you need to list train accs for this overfitting? should be hard when only one/a few layers trained and rest is from a different dataset. occurred more when more layers were trained

train the last layer (inspo from [**tibetan˙ocr**], test hypothesis of reusing feature extractor) what else? find from literature!

cleaning data from original format with gaussian blur and otsu thresholding surprisingly did not change results much

augmentation to balance classes resulted in a 10% increase in tooth type classification accuracy using a simple translation, with mnist base model. translation also makes sense since that is the main variable changing between images, zoom or rotation changing is more rare as these are scans.

initial phase experiments: mnist transfer to mpi since it was thought to be a similar problem. however like noted by [16], real world image classification is much more complex than digit classification as the benchmark images are ideal cases with eg perfectly even background. Therefore later phases experimented with transfer from imagenet classifiers since that is what most of papers in related work also did. also [8] had the same result (that imagenet is better than mnist even though mnist data looks more like character images)

start with most common ie cnn, most common transfer method ie freeze all but last layer

test both mnist and imagenet transfer best acc with mnist base: 89.

alexnet without any hyperparameter tuning with imagenet weights: immediately 100 percent on mpi, upperlower surprisingly harder, 95 percent

training vgg is slow

is hyperparameter setting fair? are hyperparameters equally optimal for all training settings? comparing is not ok if some models find a better optimum than others because of hyperparams

Training Accuracy (%)	**Validation Accuracy (%)**	**Layers Trained**	**Base Model**
92.71	**92.21**	all	AlexNet [16]
90.83	88.96	10	AlexNet [16]
84.30	80.52	5	AlexNet [16]
64.09	61.69	1	AlexNet [16]
95.25	**91.56**	all	ViT [5]
68.84	66.88	10	ViT [5]
68.73	64.94	5	ViT [5]
50.06	42.86	1	ViT [5]
90.72	**87.01**	all	VGG16 [31]
84.31	80.52	10	VGG16 [31]
72.27	67.53	5	VGG16 [31]
43.65	35.06	1	VGG16 [31]
88.73	81.17	all	DenseNet121 [14]
66.08	65.58	10	DenseNet121 [14]
65.75	65.58	5	DenseNet121 [14]
55.80	62.34	1	DenseNet121 [14]
64.75	65.58	all	ResNet101 [12]
64.75	65.58	10	ResNet101 [12]
64.75	65.58	5	ResNet101 [12]
60.77	63.64	1	ResNet101 [12]
64.53	64.29	all	EfficientNetV2S [33]
64.09	64.29	10	EfficientNetV2S [33]
64.09	64.29	5	EfficientNetV2S [33]
54.59	54.55	1	EfficientNetV2S [33]

Table 1: Training and Validation Accuracy for Different Base Models and Layer Freezing setups for Upper / Lower jaw classification

Accuracy, first seed (%)	**Accuracy, second seed (%)**	**Difference (%)**	**Layers Trained**
88.31	92.21	3.9	all
85.71	88.96	3.25	10
75.32	80.52	5.2	5
52.60	61.69	9.09	1

Table 2: Random seed sensitivity: validation accuracies with two different seeds, AlexNet upper/lower