Historical Manuscript Production Date Estimation using Deep Convolutional Neural Networks

Fredrik Wahlberg
Dept. of Information Technology
Uppsala University
fredrik.wahlberg@it.uu.se

Tomas Wilkinson
Dept. of Information Technology
Uppsala University

Anders Brun
Dept. of Information Technology
Uppsala University

***Abstract*—Deep learning has thus far not been used for dating of pre-modern handwritten documents. In this paper, we propose ways of using deep convolutional neural networks (CNNs) to estimate production dates for such manuscripts. In our approach, a CNN can either be used directly for estimating the production date or as a feature learning framework for other regression techniques. We explore the feature learning approach using Gaussian Processes regression and Support Vector Regression.**

The evaluation is performed on a unique large dataset of over 10000 medieval charters from the Swedish collection Svenskt Diplomatariums huvudkartotek (SDHK). We show that deep learning is applicable to the task of dating documents and that the performance is on average comparable to that of a human expert.

***Keywords*-Document analysis; Manuscripts; Document dating; Digital Humanities;**

I. Introduction

In this paper, we will show how deep learning methods can be used for historical analysis in digital humanities. All over the world, pre-modern handwritten manuscripts are being photographed and scanned for cultural heritage purposes on a large scale. Digitization effectively removes the requirement of physical access for doing research on historical writings. However, to make research on the full body of preserved documents possible, computational methods need to be used (it is simply too large for a human research team). We present our successful endeavour of using deep learning techniques, originally developed for computer vision, to support this exciting new type of research in the humanities. We have used deep convolutional neural networks (CNN) for estimating manuscript production dates.

We have also evaluated the advantages of using a CNN for feature learning. In our earlier work [1], [2], we have showed that a Gaussian Process (GP), in combination with handcrafted ink stroke descriptors, can be used for manuscript dating. In [3], Support Vector Regression (SVR) with a two phase estimation refinement have been used in combination with handcrafted features. In this paper, we evaluate these regression techniques with novel features learned from a CNN. This is done by substituting the output layer of the CNN with a GP or SVR based estimator. We show that the features can be learned, eliminating some of the need for the time consuming work of handcrafting features suitable for different periods and scripts.

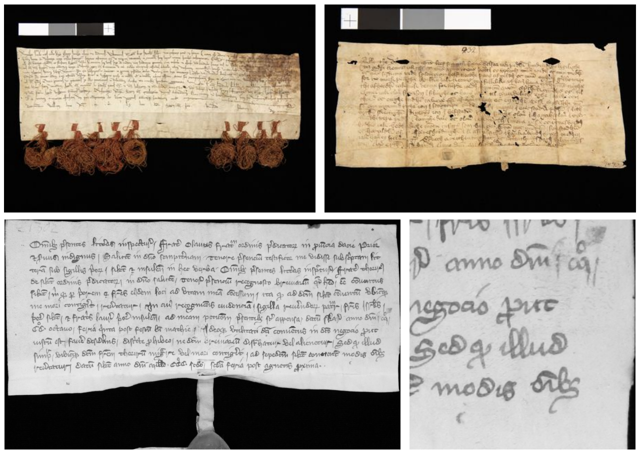
Figure 1. **Top row:** Charters 653 and 21475 from the SDHK database (section I-A), captured as 4 megapixel colour RGB images. In total, the database includes 10992 photographed and dated charters. **Botton row:** The gray scale version in charter 1962 and a cut out example used as input to the CNN.

A concern when using deep neural networks is the amount of training data required. For most applications using historical data, labeled (or even digitized) material is hard to come by. Hence, developing methods with low requirements for training data is necessary. Our networks were pre-trained on other image tasks, in part mitigating this problem. We explore the need for training data by evaluating the networks performance when given training sets of different sizes compared to the full set.

A. Dataset: "Svenskt Diplomatariums huvudkartotek"

The database "Svenskt Diplomatariums huvudkartotek"[1] (SDHK) is a charter collection at the Swedish National Archive containing dated charters from medieval Sweden. We have a used set of almost 11000 charters for training the learning methods described below. Some example images

[1]https://sok.riksarkivet.se/SDHK

from the charter collection are shown in figure 1. All used charters are dated on the day by the original scribe.

The charters contain official communications, explaining the need for dating them on the day. Most of the charters are in Swedish (going from old Swedish to a modern grammar) and a large part in Latin. Most charters are written in Sweden, while a smaller part is written in the important political centers of the time (e.g. Rome).

II. Method

A. Deep Convolutional Neural Nets

In the last few years, deep convolutional neural networks have been shown to work very well on computer vision tasks like image classification [4], [5], object detection [5], [6] and semantic segmentation [7]. It has in, many cases, replaced the traditional computer vision methods when it comes to achieving top performance.

The only previous work using deep learning for document dating that we are aware of is found in [8] where the authors worked with modern (i.e. years 1600-1999) printed text. The date estimation was based on both the image data and text data extracted using OCR software run on the document image. Following the common assumption that deep networks need large amounts of training data, most of their data was used for training (the test set size was about 1% of the training set size).

For our experiments, we use the popular GoogleNet architecture from [9]. A model pre-trained on the Imagenet dataset for 120000 iterations (parameter updates) is used to initialize the model. Despite the pre-training data being far from document images, the network learns general image features that work well for many kinds of images, especially in the lower layers of the network. We also need to adapt the model trained on Imagenet, since it has been trained to classify natural images. We do this by first removing the final layer from the model and replacing it with a fully connected layer with a single output neuron for regression. We also remove the auxiliary classifiers.

The Imagenet dataset consists of RGB images of size $256 \times 256 \times 3$. As a simple method of data augmentation, GoogleNet is typically trained on $224 \times 224 \times 3$ random crops of the full images. To be able to use this network, we need inputs of the same size. We solve this problem with two simple solutions. For the spatial size problem, we generate 20 random crops of $256 \times 256 \times 3$ for each letter, and during training, we randomly crop $224 \times 224 \times 3$ crops of the full images. For the three-channel problem, we opt for the simplest solution we can think of, that is to replicate the input gray scale image in each of the three input channels.

The GoogleNet architecture is based on using a fixed collection of layers as a module that is called the inception module. It is a move away from the more traditional way of thinking about CNNs in terms of individual layers. See Table I for a condensed overview of the GoogleNet architecture.

Table I
Overview of our CNN, based on the Googlenet architecture. For more details see [9], [10].

layer type	kernel size/stride	output size
convolution	$7 \times 7/2$	$112 \times 112 \times 64$
max pool	$3 \times 3/2$	$56 \times 56 \times 64$
convolution	$3 \times 3/1$	$56 \times 56 \times 192$
max pool	$3 \times 3/2$	$28 \times 28 \times 192$
inception(3a)		$28 \times 28 \times 256$
inception(3b)		$28 \times 28 \times 320$
inception(3c)	stride 2	$28 \times 28 \times 576$
inception(4a)		$14 \times 14 \times 576$
inception(4b)		$14 \times 14 \times 576$
inception(4c)		$14 \times 14 \times 576$
inception(4d)		$14 \times 14 \times 576$
inception(4e)	stride 2	$14 \times 14 \times 1024$
inception(5a)		$7 \times 7 \times 1024$
inception(5b)		$7 \times 7 \times 1024$
average pool	$7 \times 7/1$	$1 \times 1 \times 1024$
fully connected		1

We do the fine-tuning in four steps. First we initialize and train the final layer for 7500 iterations, this is done since gradients of the newly initialized output layer is too high compared to the pre-trained network so that fine-tuning all the layers at once would ruin the initial value. Then we fine-tune the last layer of the inception modules (layer 5) and onward for 7500 iterations, then the same with layer 4. Finally, we train inception layers 3 and onward for 50000 iterations. We evaluate our model on the validation set (10% of the data) every 5000 iterations and our final model for each train split is the iteration that performed best on the validation set, on which we evaluated every 5000 iterations during training. For all experiments we use a mini-batch size of 16, a weight decay of 0.0002, and we use ADAM [11] to train the network. We initialize the learning rate of 0.001 and multiply it with 0.1 every 20000 iterations.

During the testing phase, we get one date estimate for each of the 20 random crops. To merge this into a single estimate, we take the median of the 20 date estimates. We find that this works slightly better than taking the mean. We hypothesize that this is because the random crops can sometimes only capture background, causing the network to give a poor date estimate, affecting the mean, but not the median.

B. Support Vector Regression

As described above, a fully connected layer with one output (i.e. a linear combination) was used for projecting the features learned by the CNN onto a time line. In some applications for deep networks, the last layer have successfully been switched out and replaced with a Support Vector Machine (SVM) (e.g. in [12] and [13]). We have replaced the last layer with a Support Vector Regression (SVR) to give us the output data on a time line.

Several applications of SVR for dating can be found in the literature. In [14] and [15], the authors propose a 2 step SVR approach for estimating the age of people. The authors propose a methodology where they first train the SVR on all the training data. The estimations given by this initial model are then used to find a smaller set of training point around every element in the test set. From this smaller set, a local SVR is trained and the estimation of the date in this way refined. The assumption was that in a small volume of the feature space, the parameters of the best SVR estimator might differ from the best SVR on the whole training set. This approach is used in [3] for estimating the production dates of a dutch charter collection, which is similar to SDHK. The authors used hand crafted features, setting their approach apart from ours.

As kernel function for the SVR[2], we chose a radial basis function (RBF). The RBF is shown in equation 1. Note that this kernel does not have automatic relevance detection as the kernel for the Gaussian Process in section II-C does.

$$k(x, x') = \exp(-\gamma \|x - x'\|^2) \tag{1}$$

For estimating the best pair of the model parameters C (penalty of the error term) and γ, we used random sampling in the interval $[-2^{30}, 2^{30}]$. We then trained a SVR on a subset of the training set (10000 randomly picked points) and evaluated the SVR on the full validation set. As described in section II-A, in the validation set, multiple elements belonged to the same charters. Estimations were merged by taking the median for each charter, giving one final estimate per charter. The splits for the sets were the same as for the CNN.

When the best global SVR had been found, a set of all possible local SVRs (i.e. trained on smaller sets) were trained. Finally, the test set was evaluated using first the global SVR and then the set of local SVRs.

C. Gaussian Process Regression

An interesting approach to regression from the Bayesian Non-parametric family of methods, is the Gaussian Process [3] (GP) (see [17]). In [1] and [2], we used a GP to do regression over a bag-of-features type feature space for manuscript dating. There, the feature extraction was handcrafted, while in this paper the CNN did the feature learning. We replaced the output layer of the CNN with a GP after tuning the network.

[2]Our implementation was based on SciKit-Learn http://scikit-learn.org

[3]The Gaussian Process regression was performed using the excellent implementation from [16].

The format for the training set $\mathcal{D}$ (of size n) is that we have pairs of feature vectors and targets $\mathcal{D} = \{(\mathbf{x}_i, y_i)), i = 1...n\}$ for the regression task. The regression is modelled as $y = f(\mathbf{x}) + \epsilon$ where ϵ is Gaussian noise. In the parametric case, our regression method would learn some parameters for f to minimize some loss function (e.g. MSE). Then $f(\mathbf{x})$, with $\mathbf{x}$ from the test set, would be used for making point estimates for y. In the case of a GP, f is drawn from a Gaussian Process prior and the training data is kept and not reduced to some set of parameters. Marginalising over the parameter space gives us a final mapping from the feature space to the target space that maximizes the likelihood of producing each y_i and points $\mathbf{x}_i$. The parameters in f, like σ or l in equation 2, are marginalised giving rise to the name non-parametric. A GP regression estimates a distribution over transfer functions f. Hence, the regression output is a distribution over the target space (in our case, the time line).

The kernel function for the GP , shown in equation 2, was chosen to be a radial basis function (RBF) with automatic relevance determination (ARD). Varying the ARD parameters is equivalent to stretching the feature space in which the regression is performed. In this way, feature weighting can be performed by optimizing over the likelihood of the process giving the target values. We also performed experiments without ARD, letting all l_i have the same value.

$$k(x, x') = \sigma^2 \exp\left(-\frac{1}{2} \sum_{i=1}^{|x|} \frac{(x_i - x_i')^2}{\ell_i^2} \right) \tag{2}$$

A weakness of the standard GP is the computational and memory complexities. The calculations required for inference scale as $O(n^3)$ and the memory requirement as $O(n^2)$. Due to the data augmentation, our training set was larger than could be stored in memory on a machine with 16GB of RAM. Inspired by stochastic gradient descent, we implemented a mini-batch GP regression. In this way, all of the training data was used at some point during the training. This made it possible to use the full training data set for GP regression. However, since only a mini-batch is used at each iteration, the resulting model can only represent a lower bound of a model using the full set for inference at each iteration.

A more sound approach to reducing the memory complexity is the sparse Gaussian process (see [18]). There are several types of sparse GPs, we chose a formulation where a number of "inducing points" picked from the same domain as the training data are used as stand-ins for the actual training data when performing inference. This reduces the computational complexity to $O(nm^2)$, where m is the number of inducing points. However, with a memory complexity of $O(nm)$ and a large training set, our memory limitations only allowed for small values of m. However, having the full training set available at every iteration for inference was a great improvement over our mini-batched GP. Since $n \approx 100000$ we let $m \in \{50, 100, 150, 200\}$.

The inducing points should, as stated above, be sampled from the same domain as the training data. To achieve this we used a mini-batch K-Means to cluster the training data, using the cluster centers as inducing points. This procedure also concentrates the inducing points in areas where there is more data in the training set, hopefully imitating it better than a random sampling.

III. Evaluation

A. Evaluation Dataset: "Svenskt Diplomatariums huvudkartotek" (SDHK)

The images of "Svenskt Diplomatariums Huvudkartotek" (SDHK) was photographed over a period of 10 years. The images were originally taken as a way of reproducing/copying the charters for researchers interested in different parts of the collections. As such, the image quality is lower than if the reproduction was performed today. The high resolution images have a size of 4 M pixels where a large part is the black background. For the web images, there are also jpeg compression. In this paper, we have used the high resolution gray scale images.

B. Evaluation Metrics

Our main evaluation metric was the mean square error (MSE). It is shown in equation 3 where I is the image set, t the target years and f the year estimation function.

$$E_{MSE}(I,t) = \frac{1}{|I|}\sum_{i=0}^{|I|}(f(I_i) - t_i)^2 \tag{3}$$

When presenting the estimated production dates to researchers in for example history, the outliers (i.e. for them the obvious errors) are of higher importance to eliminate then smaller errors. This is consistent with the performance of a human estimate that would almost never produce an extreme outlier (unless intentionally deceived).

We also present percentiles of the absolute errors to give a more intuitive measure of the error distribution.

C. Results

A common assumption for using deep networks is that they require large sets of training data. Hence, we started our evaluation using 60% of the data set for training. To make sure we were right in this, we then reduced the training data in steps of 10 percentage points. These results are shown in the left panel of figure 2. Down to 20% training data the results, in terms of MSE, were comparable to using more data. We think this is because the network was pre-trained on other image tasks before the final tuning for our specific task. Our evaluations of other regression methods, using the CNN only for feature learning, are based on the 10% training set.

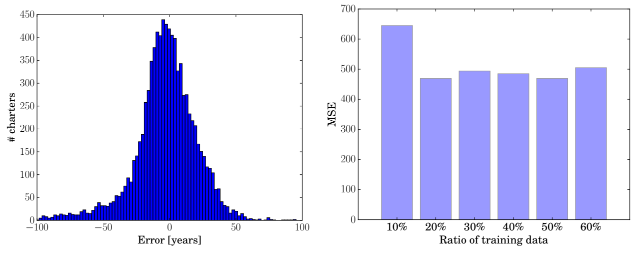

Figure 2. Evaluation results for our modified GoogleNet using 10% training data. **Left:** The distribution of estimation errors, **Right:** Evaluation results in MSE for some training data sizes. Only at 10% training data does the results degrade. Hence, we focused on applying other methods on this training set, using the network only for feature learning.

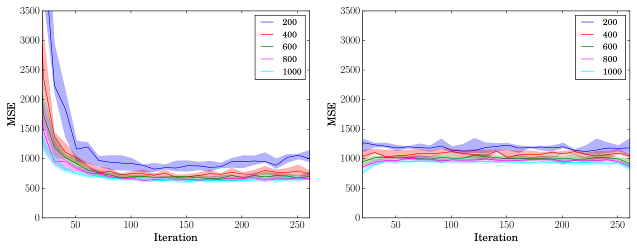

Figure 3. Plot for MSE over training iterations using the mini-bacth Gaussian Process (GP). Evaluations are on the test set, though the training was performed only on the training set. The legend shows the varying mini-batch sizes. A higher batch size gives a better final result. After some time, the randomness of the training batches seem to prevent better tuning. The lines are median performance with a colored region showing the min and max. Note that a GP does not optimize for median MSE (shown here), but likelihood. They are correlated but nonequivalent. **Left:** GP with ARD **Right:** GP without ARD

As shown in table II, the performance of GoogleNet ("CNN" in the table) was a lot better than the performance of AlexNet. Therefore, we stopped our experiments on AlexNet at 60% training data.

To the left in figure 5, the distributions of estimation errors for the SVR as output layer are shown. In both histograms, a large bias can be seen. Bias is expensive in terms of MSE, but this does not completely explain the high MSE score (shown in table II). The distribution of errors are also wider than for other methods.

Refining the estimation using a SVR trained on a smaller local set did not improve the estimation. As seen in figure 6, the global+local SVR is better at lower errors while only using the global SVR is better at higher percentiles.

The evaluation results from using the non-sparse mini-batch GP is shown in figure 3. Using a larger batch when training gave better results, probably due to more information being available at each iteration of the training. However, after some time the performance leveled out with a small variance around some limit. The randomness in what information was represented at each training mini-batch might be the cause of this.

As stated above, a more statistically sound method for

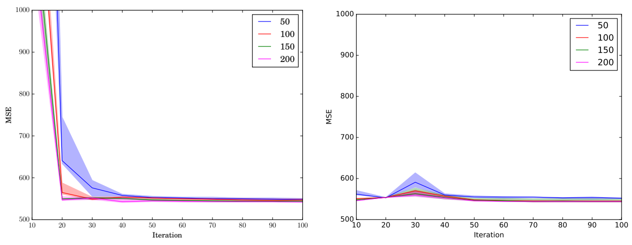

Figure 4. Plot for MSE over training iterations using the Sparse Gaussian Process (GP). Evaluations are on the test set, though the training was performed only on the training set. The legend shows the varying inducing point set size m. Surprisingly, using ARD only gave a slightly better final result, with a much bigger cost in training. The lines are median performance with a colored region showing the min and max. Note that a GP does not optimize for median MSE (shown here) but likelihood, though correlated they are not equivalent. **Left:** Sparse GP with ARD **Right:** Sparse GP without ARD

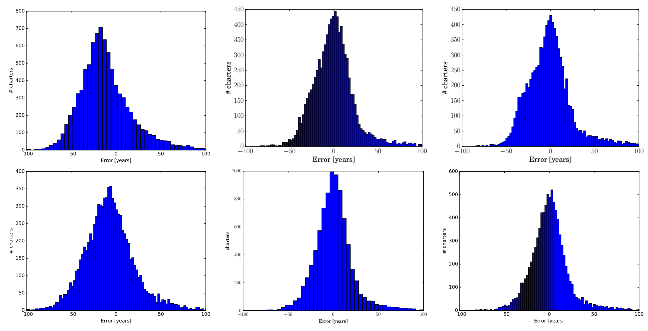

Figure 5. Histograms over the estimation errors for SVR and Gaussian Process (GP) output layers using 10% training data. **Upper Left:** Distribution of estimation errors for the global SVR, **Upper middle:** GP-MiniBatch with Automatic relevance determination ARD, layer. **Upper right:** GP-MiniBatch without ARD, **Lower left:** Distribution of the estimation error for the two phase SVR refinement, **Lower middle:** Sparse GP with ARD, **Lower right:** Sparse GP without ARD

using more training data with restricted resources is a Sparse GP. The evaluation results from using the sparse mini-batch GP is shown in figure 4. Convergence is much faster without ARD but with some penalty in the result. Increasing the number of inducing points m improved the results, but showed diminishing returns with higher values (as computational complexity increased with m^2).

In figure 5, the distributions over estimation errors are shown for all GP variants (ARD, sparseness). Though the mode of the distributions are narrower for Sparse GPs, the main difference are in the tails. The much smaller tails of the Sparse GPs compared to the mini-batch GPs (and also SVRs) has a great impact on MSE. Also, eliminating outliers is of great importance to historians.

Finally, the MSE and error percentiles for some representative chosen runs are shown in table II. In figure 6, the corresponding percentiles of absolute error curves are shown. All methods tried showed very similar results at lower error percentiles. The main difference seem to be how larger error were handled. This is supported by that P25 and P50 in table II are very similar but MSE can double (e.g. in the case of SVR as output layer). Comparing P80 for best and worst estimators is approximately 15 years, which we argue is high. Note that the comparisons from [1] and [2] are produced using a low resolution version of SDHK and hand crafted features (and in the latter case, transcribed text).

To overcome the problem with outliers in the estimations we have used ensembles of estimators in earlier work (see [2]). When the intersection between the sets of outliers from the estimators was small, the higher errors could be lowered (see comparisons in table II) by merging the estimations. For the future it would be very interesting to look at if the worst 5% of the estimates in this paper, are the same set for each estimator.

Table II

EVALUATION METRICS FOR SOME OF THE RESULTS. THE METRICS PRESENTED ARE THE 25th (P25), 50th (P50) AND 75th (P75) PERCENTILES OF THE ABSOLUTE ERROR TOGETHER WITH THE MEAN SQUARE ERROR (MSE). THE SIZE OF THE TRAINING SET IS GIVEN AS PERCENT OF THE FULL SET (I.E. 10992 CHARTERS)

Estimator	Training	P25	P50	P75	MSE
CNN	10%	5.9	12.7	23.3	645
SVR RBF global	10%	8.2	17.9	31.8	905
SVR RBF global+local	10%	6.7	16.4	33.0	940
Mini-batch GP (ARD)	10%	5.9	13.0	23.7	640
Sparse GP (m=200,ARD)	10%	5.1	11.1	20.7	**550**
CNN	20%	4.6	10.0	18.5	**469**
CNN	30%	4.7	10.3	19.0	494
CNN	40%	4.5	10.2	19.0	485
CNN	50%	4.4	10.0	18.5	469
CNN	60%	4.9	10.7	20.1	505
CNN (AlexNet)	60%	-	-	-	840
Comparison from [1]	5%	7.9	18.3	36.8	1389
Comparison from [2]	6%	8	17	30	810
Comparison from [2]	7% (+text)	6	12	22	462

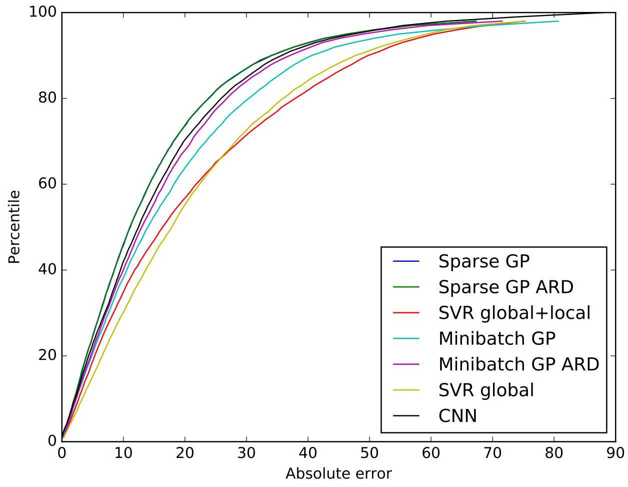

Figure 6. Evaluation results given in varying percentiles of absolute error. "CNN" refers to using the full network and all other types given in the legend are for different output layers, using the CNN for feature learning. Training set was set to 10% of the full collection.

IV. Conclusion

In this paper we show that deep convolutional neural networks can be used for production date estimation for pre-modern handwriting. We show that a network can be trained to have a lower error than ± 20 years for 75% of the manuscripts. This is comparable (and sometimes even better) than a human expert.

When using historical sources, the training data is almost always very limited, we overcome this by using a network that was pre-trained on a image classification task. Hence, we show that as little as 10% of the charter collection "Svenskt Diplomatariums huvudkartotek" (SHDK) is needed for training the network. The evaluation collection consisted of 10992 pre-modern charters.

In previous papers on dating pre-modern material, Support Vector Regression and Gaussian Process Regression have been used (but with hand crafted features). We show that a deep convolutional network can be used for feature learning together with these regression techniques. We are the first to acknowledge that our neural network approach is simplistic (minimal adaptation, random image sampling, pre-training on non-document data), our intention being to show the strength of deep models with minimal domain specific adaptation.

Acknowledgment

We would like to thank Lasse Mårtensson for sharing his insights and expertise in palaeographical manuscript dating. We would also like to thank the Swedish National Archive [4], Sara Risberg and Per-Axel Wiktorsson for their generous help in providing images, metadata and expertise.

This project is a part of q2b, From quill to bytes, an initiative sponsored by the Swedish Research Council (Vetenskapsrådet, Dnr 2012-5743), Riksbankens Jubileumsfond (Regnumber NHS14-2068:1) and Uppsala university.

References

[1] F. Wahlberg, L. Mårtensson, and A. Brun, "Large scale style based dating of medieval manuscripts," in *The 3rd International Workshop on Historical Document Imaging and Processing (HIP15)*, 2015.

[2] F. Wahlberg, L. Mårtensson, and A. Brun, "Large scale continuous dating of medieval scribes using a combined image and language model," in *Proc. of 12th IAPR International Workshop on Document Analysis Systems (DAS)*, April 2016.

[3] S. He, P. Samara, J. Burgers, and L. Schomaker, "Towards style-based dating of historical documents," in *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*. IEEE, 2014, pp. 265–270.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 91–99.

[7] G. Lin, C. Shen, I. Reid *et al.*, "Efficient piecewise training of deep structured models for semantic segmentation," *arXiv preprint arXiv:1504.01013*, 2015.

[8] Y. Li, D. Genzel, Y. Fujii, and A. Popat C., "Publication date estimation for printed historical documents using convolutional neural networks," in *The 3rd International Workshop on Historical Document Imaging and Processing (HIP15)*, 2015.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[11] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[12] F. J. Huang and Y. LeCun, "Large-scale learning with svm and convolutional for generic object categorization," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 284–291.

[13] Y. Tang, "Deep learning using linear support vector machines," *arXiv preprint arXiv:1306.0239*, 2013.

[14] G. Guo, Y. Fu, C. Dyer, and T. Huang, "Image-based human age estimation by manifold learning and locally adjusted robust regression," *Image Processing, IEEE Transactions on*, vol. 17, no. 7, pp. 1178–1188, July 2008.

[15] H. Zhang, A. Berg, M. Maire, and J. Malik, "Svm-knn: Discriminative nearest neighbor classification for visual category recognition," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, 2006, pp. 2126–2136.

[16] GPy, "GPy: A gaussian process framework in python," http://github.com/SheffieldML/GPy, since 2012.

[17] C. E. Rasmussen, "Gaussian processes for machine learning." MIT Press, 2006.

[18] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[4]https://riksarkivet.se/