Latent Log-Linear Models for Handwritten Digit Classification

Thomas Deselaers, *Member, IEEE*, Tobias Gass, Georg Heigold, and Hermann Ney, *Member, IEEE*

Abstract—We present latent log-linear models, an extension of log-linear models incorporating latent variables and we propose two applications thereof: log-linear mixture models and image deformation-aware log-linear models. The resulting models are fully discriminative, can be trained efficiently, and the model complexity can be controlled. Log-linear mixture models offer additional flexibility within the log-linear modeling framework. Unlike previous approaches, the image deformation-aware model directly considers image deformations and allows for a discriminative training of the deformation parameters. Both are trained using alternating optimization. For certain variants convergence to a stationary point is guaranteed and in practice even variants without this guarantee converge and find models that perform well. We tune the methods on the USPS dataset and evaluate on the MNIST dataset demonstrating the generalization capabilities of our proposed models. Our models, although using significantly fewer parameters, are able to obtain competitive results with models proposed in the literature.

Index Terms—Log-linear models, latent variables, conditional random fields, OCR, image classification

1 Introduction

INCORPORATING *latent*, or *hidden*, variables into a model is a well-known means of increasing its expressiveness. Latent variables are not directly observed from the data but inferred from other variables. In machine learning and pattern recognition, latent variables are for example used in speech recognition to account for temporal variabilities [32], in information retrieval and natural language processing to analyze the relationships between terms and concepts [23], and in object recognition to model the positions of object parts [8].

We develop two log-linear models incorporating latent variables: log-linear mixture models and deformation-aware log-linear models. In general the training of models with latent variables is hard. Therefore many approaches restrict their choice of models to those for which efficient and optimal algorithms exist. In order to improve the expressiveness of such models often the *kernel trick* is applied, e.g. in support vector machines (SVMs) [35]. To train an SVM, a convex optimization problem is solved, which can be done optimally and efficiently. The resulting classifiers are linear hyperplanes and the kernel trick allows for complex models by optimizing the decision hyperplane implicitly in a very high (possibly infinite) dimensional space. For many applications, kernel methods have been able to obtain very good results in the last years.

In this work we present extensions to log-linear models. Starting from a conventional log-linear model, we develop

- log-linear mixture models which increase the flexibility of the model in general;
- deformation-aware log-linear models which increase the flexibility of the model by incorporating prior knowledge about image deformations.

Although the function to be optimized during training of our proposed models is not convex, we present an efficient training algorithm, which is guaranteed to converge to a stationary point and finds models that perform well in practice. To the best of our knowledge, the deformation-aware log-linear model is the first model which jointly (and discriminatively) trains the deformation parameters with the remaining model parameters. We carefully evaluate the proposed models on two public OCR datasets.

1.1 Related Work

Discriminative Modeling

The aim of this work is to add flexibility and to learn parameters that reflect domain specific prior knowledge into a *discriminative* classification framework. In kernel methods, additional flexibility is obtained through the kernel trick [35], e.g. deformation invariance [13, 14]. However, it is not easy to learn kernel parameters and thus these are often tuned using cross validation [5]. Gehler and Nowozin [10] propose a method to implicitly learn the kernel parameters while training the classification model by selecting kernels from a potentially infinite set of base kernels.

Instead of using a kernel, we start from a conventional discriminative log-linear model and incorporate latent

- *Thomas Deselaers and Tobias Gass are with the Computer Vision Laboratory, Department of Information Technology and Electrical Engineering, ETH Zurich, Zurich, Switzerland. E-mail:* `thomas@deselaers.de`, `gass@vision.ee.ethz.ch`
Georg Heigold and Hermann Ney are with the Human Language Technology and Pattern Recognition group, Computer Science Department of RWTH Aachen University in Aachen, Germany. E-mail: {`heigold`, `ney`}`@cs.rwth-aachen.de`
The research presented in this work has been performed while all authors were with the Human Language Technology and Pattern Recognition group, Computer Science Department of RWTH Aachen University in Aachen, Germany.

variables to extend its flexibility. The resulting models can be seen as conditional random fields (CRFs) [22] with latent variables [12, 31], where the latent variables account for the assignment of observations to the mixture components in the first case and for the deformations in the second case. Heigold et al. [16] and Gunawardana et al. [12] use log-linear models within hidden Markov models (HMMs) for speech recognition. An approach similar to the log-linear mixtures presented here was used for local-feature-based object classification in [38].

Deformation Modeling

To model image deformations, conventional approaches can be split into two groups:

Approaches that directly incorporate invariance: Haasdonk and Keysers [15] incorporate the tangent distance into support vector machines. DeCoste and Schölkopf [7] use kernel jittering to obtain translated support vectors in a two-step training approach. Keysers et al. [19, 20] use transformation invariant distance measures in a nearest neighbor framework. They also propose a deformation-aware Gaussian model but do not train the deformation parameters.

Approaches that implicitly incorporate invariance: Another approach is not to incorporate the deformation invariance into the model but to use a huge amount of synthetically deformed data during training. LeCun et al. [25] and Simard [36] train multi-layer convolutional neural networks that implicitly learn the occurring deformations from training data.

The first approach has the disadvantage that during testing a large amount of potentially computationally expensive image comparisons have to be performed whereas in the second approach the training procedure may become very expensive. None of these approaches explicitly learns the parameters of the allowed deformations but the deformation model was hand-coded by the system developers, either in designing the distance function or in generating the deformed training samples. In contrast to these approaches to transformation invariant classification, Memisevic and Hinton [27] proposed an approach to learn image transformations from corresponding image pairs using conditional restricted Boltzmann machines. This approach can also be used for classification, but the deformation and classification parameters are decoupled.

In our deformation-aware log-linear model, we aim at training

- a small (in the number of parameters) model that
- directly models deformations,
- automatically learns which deformations are allowed (and desired), and
- is efficient to train and apply.

We build our approach around the image distortion model (IDM) [20], a zero-order, non-linear deformation model, which we briefly describe in Section 5. A preliminary version of the part on deformation-aware models was published in [9].

Structure

The remainder of this paper is structured as follows: In Section 3 we present how log-linear models are extended to incorporate latent variables and how this is applied to create log-linear mixture models (Section 4) and deformation-aware log-linear models (Section 5). In Section 6 we experimentally evaluate the proposed approaches on two standard datasets. We tune and evaluate both models on the USPS dataset for optical character recognition (OCR). Then, we use the settings that worked best on the USPS dataset to train and to evaluate a model on the MNIST dataset. Finally the paper is summarized and concluded.

2 Log-linear Models

Log-linear models [6, 21] are discriminative classification models that have been used successfully in many applications, such as natural language processing [3, 30]. Log-linear models are closely related to other machine learning techniques such as perceptrons and SVMs. In a log-linear model the posterior probability for class c is given directly as

$$p_\theta(c|X) = \frac{\exp(g_\theta(c, X))}{\sum_{c'} \exp(g_\theta(c', X))}, \qquad (1)$$

where $g_\theta(c, X)$ is a linear function of the input vector X, i.e. $g_\theta(c, X) = \alpha_c + \lambda_c^T X$ with parameters $\theta = \{\alpha_c, \lambda_c\}$ for $c = 1, \dots, C$. X is a feature vector to be classified. The parameters θ are estimated in training. As such, log-linear models do not incorporate invariance with respect to any variabilities in the input data explicitly but are able to learn which variations occur in their training data implicitly.

The input vectors X can be represented by (possibly non-linear) functions $f(c, X)$ of c and X. This allows for great flexibility in this type of model and the incorporation of higher order features. Analogously, a discriminant function g_θ which is non-linear in the input vectors X can be used. The resulting models are called generalized log-linear models. A special case are models with a quadratic (in X) discriminant function $g_\theta(c, X) = \alpha_c + \lambda_c^T X + X^T \Lambda_c X$. These are called second-order log-linear models and can be trained efficiently analogously to the linear case. We also refer to the experimental evaluation (Section 6.2) where we use a kernelized log-linear model for comparison.

To train a log-linear model given a set of training observations $\{X_1, \dots, X_N\}$ with labels $\{c_1, \dots, c_N\}$, we maximize the (regularized) maximum mutual information (MMI) criterion over the parameters θ[1] [18]

$$F_{\text{MMI}}(\theta) = \frac{1}{N} \sum_n \log p_\theta(c_n|X_n) - \gamma ||\theta||^2 \qquad (2)$$

where $\gamma > 0$ is the regularization factor, and $||\theta||^2$ is the L_2 norm over all model parameters θ. The training of log-linear models according to this criterion is a

[1] also referred to as maximum-likelihood over the posteriors

convex optimization problem leading to a linear decision boundary and several algorithms exist that allow for effectively finding the globally optimal model [6, 26, 28].

The class posterior of a single Gaussian classifier can be expressed in log-linear form [1]. Heigold et al. [16] showed that training Gaussian models and log-linear models according to the same criterion leads to the same classifier. Experimentally, it was observed that training a log-linear model may be numerically more stable, since the inversion of the covariance matrix is not required. This is particularly interesting for Gaussian models with full covariance which correspond to second-order log-linear models.

3 Log-Linear Models with Latent Variables

In this section, we describe a general approach of incorporating latent variables into log-linear models to better model the variability of the data to be recognized. In Sections 4 and 5 we present applications that use latent variables to extend the capabilities of log-linear models.

To integrate a discrete latent variable A into a log-linear model, we sum over the joint probability of the newly introduced latent variable:

$$p_\theta(c|X) = \sum_A p_\theta(c, A|X) = \frac{\sum_A \exp(g_\theta(c, A, X))}{\sum_{c'} \sum_{A'} \exp(g_\theta(c', A', X))} \tag{3}$$

Training such a model according to the MMI criterion (eq. (2)) is not a convex problem anymore due to the sum in the numerator and is NP-hard. In Section 3.2 we present a training algorithm which in practice finds good models and for which under certain circumstances convergence can be guaranteed.

3.1 Maximum Approximation

The joint probability for a given configuration A of the latent variable and a class is in the form of a regular log-linear model over pseudo-classes (c, A):

$$p_\theta(c, A|X) = \frac{\exp(g_\theta(c, A, X))}{\sum_{(c',A')} \exp(g_\theta(c', A', X))} \tag{4}$$

This means that given a configuration A for each training sample such models can be trained efficiently and optimally. However, the *correct* configuration of A is not known and thus we approximate A using the configuration $A_c(X)$ which maximizes the discriminant function for an observation X:

$$A_c(X) = \arg\max_A \{g_\theta(c, A, X)\} = \arg\max_A \{p_\theta(c, A|X)\} \tag{5}$$

We expect the maximum approximation, $p_\theta(c|X) \approx p_\theta(c, A_c|X)$ to be a good approximation because the approximated function exponentially decreases from its maximum [2].

Combining these observations, we derive a training algorithm that is guaranteed to improve the training criterion F_{MMI} in every iteration; when the criterion cannot be improved anymore, the algorithm converges (cf. Section 3.2). Analogously to applying the maximum approximation in the numerator, it can also be applied in the denominator. Therefore, a configuration of the latent variables has to be determined for each class for each observation independently: $A_1^C(X) = (A_1(X), \ldots, A_C(X))$ is determined according to Eq. (5) and used in the posterior:

$$p_\theta(c, A_1^C|X) = \frac{\exp(g_\theta(c, A_c, X))}{\sum_{c'} \exp(g_\theta(c', A_{c'}, X))} \tag{6}$$

Each A_c maximizes its respective class-specific discriminant function: $A_c = \arg\max_A \{g_\theta(c, A, X)\}$ (for all classes $c = 1, \ldots, C$).

In the following, we refer to the model without the maximum approximation as SumSum (eq. (3)), the model with the maximum approximation in the numerator only is denoted as MaxSum (eq. (4)), and the model with the maximum approximation in numerator and denominator is denoted MaxMax (eq. (6)). An overview of the different model variants is given in Table 1 along with the latent variables that have to be pre-determined, and the derivatives required for training.

3.2 Training Method

To train the models we apply gradient descent methods. For the SumSum model, the gradients can be calculated directly. For the MaxSum method, the configuration A_{c_n} of the correct class c_n is fixed for each observation X_n. For the MaxMax method, the configurations A_c for all classes $c = 1, \ldots, C$ are fixed for each observation X_n. The models are then iteratively trained, alternating between reestimating the model parameters and updating the configuration of the latent variables.

Table 1 shows the derivatives used for the training. The alternating training procedure works as follows:

1) For each training sample X_n estimate the configuration A of the latent variable according to Eq. (5) for
   - class $c = c_n$, when training a MaxSum-model.
   - each class $c = 1, \ldots, C$, when training a MaxMax-model.
2) Optimize $F_{\text{MMI}}(\theta)$ with fixed configuration A using a gradient descent method. We use the limited-memory BFGS (LBFGS) Newton method [26].
3) if not converged, go back to 1.

When training a MaxSum-model, step 2 is identical to training a conventional log-linear model over pseudo-classes (c, A) (cf. eq. (4)). This is a convex optimization problem where LBFGS [26] will converge to the global optimum for the parameters θ given the configuration of the latent variables A. Given this set of parameters θ, the algorithm goes back to step 1 and chooses the configuration of latent variables A that maximizes the training criterion (2). A will only be changed if it can be

TABLE 1: Overview on the variants of the maximum approximation in comparison to conventional log-linear models along with the derivatives of the training criterion (Eq. (2)). The column "latent" specifies which latent variables need to be pre-determined to calculate $p(c|X)$. These variables also have to be pre-determined for each training observation to calculate the derivative (note $p(A|c_n, X_n) = \frac{p(c_n, A|X_n)}{\sum_{A'} p(c_n, A'|X_n)}$).

Model	$p(c\mid X)$	latent	$\frac{\partial}{\partial\theta} \sum\limits_n \log p_\theta(c_n\mid X_n)$
log-lin. mod.	$\frac{\exp(g_\theta(c,X))}{\sum\limits_{c'} \exp(g_\theta(c,X))}$	–	$\sum\limits_n \sum\limits_c [\delta(c, c_n) - p_\theta(c\mid X_n)] \frac{\partial}{\partial\theta} g_\theta(c, X_n)$
log-linear model with latent variables			
SUMSUM	$\frac{\sum\limits_A \exp(g_\theta(c,A,X))}{\sum\limits_{c'} \sum\limits_{A'} \exp(g_\theta(c',A',X))}$	all free	$\sum\limits_n \sum\limits_c \sum\limits_A [\delta(c, c_n) p_\theta(A\mid c_n, X_n) - p_\theta(c, A\mid X_n)] \frac{\partial}{\partial\theta} g_\theta(c, A, X_n)$
MAXSUM	$\frac{\exp(g_\theta(c,A_c(X),X))}{\sum\limits_{c'} \sum\limits_{A'} \exp(g_\theta(c',A',X))}$	$A_c(X)$	$\sum\limits_n \sum\limits_c \sum\limits_A [\delta(c, c_n) \delta(A_{c_n}(X_n), A) - p_\theta(c, A\mid X_n)] \frac{\partial}{\partial\theta} g_\theta(c, A, X_n)$
MAXMAX	$\frac{\exp(g_\theta(c,A_c(X),X))}{\sum\limits_{c'} \exp\big(g_\theta(c',A_{c'}(X),X)\big)}$	$A_1(X) \dots A_C(X)$	$\sum\limits_n \sum\limits_c [\delta(c, c_n) - p_\theta(c, A_c(X_n)\mid X_n)] \frac{\partial}{\partial\theta} g_\theta(c, A_c(X_n), X_n)$

improved, since otherwise it would remain constant. If it is changed, in step 2, the parameters are re-learned. Otherwise, in step 2 no parameter update is performed and training is converged to a stationary point(neither changing A nor θ can improve the criterion (2)). That is at this point we either have a local optimum or a stationary point with two configurations A of the latent variable that have the same criterion. Furthermore, the MMI criterion (2) is bounded from above [4, 11].

Unfortunately, such a guarantee cannot be shown for training MAXMAX or SUMSUM models. For MAXMAX models, changing A may lead to a deterioration of the training criterion. However, in practice training also converges for these models (cf. Section 6.1.1).

4 LOG-LINEAR MIXTURE MODELS

MIXTURE models, such as Gaussian mixture distributions (GMDs), are a standard technique to allow for modeling complex data in Gaussian approaches. Analogously to the way Gaussian mixtures extend single Gaussians, we extend log-linear models toward log-linear mixture models (LLMMs). In this case, the configuration of the latent variable models the alignment of observations to the model components (densities). The posterior is given as

$$p_\theta(c|X) = \sum_i p_\theta(c, i|X) \tag{7}$$

$$= \frac{\sum_i p_\theta(c, i, X)}{\sum_{c'} \sum_{i'} p_\theta(c', i', X)} \tag{8}$$

$$= \frac{\sum_i \exp(g_\theta(c, i, X))}{\sum_{c'} \sum_{i'} \exp(g_\theta(c', i', X))} \tag{9}$$

where g_θ is chosen as $g_\theta(c, i, X) = \alpha_{ci} + \lambda_{ci}^T X$ and $\theta = \{\alpha_{ci}, \lambda_{ci}\}$.

LLMMs are the discriminative counterpart of Gaussian mixture models analogously to the relationship between log-linear models and single Gaussian classifiers [33]. This relationship allows for transforming one into the other and e.g. to use a GMD model to initialize an LLMM or vice versa.

4.1 Initialization

Since training of these models is not convex, the result may depend on the initialization of the model. We investigate three different initializations:

4.1.1 Initialization from a Gaussian Mixture Model

We train a Gaussian mixture for each of the classes c using the EM algorithm. Therefore we start from a single Gaussian, which is then incrementally split and re-estimated, until the desired number of densities is obtained. This algorithm is known to lead to stable results, and unlike the k-means algorithms does not depend on a random initialization. The Gaussian mixture densities are then converted into an LLMM following [16, 33].

4.1.2 Initialization by Incremental Splits

Analogously to our GMD training algorithm, we iteratively split the LLMM until the desired number of components is obtained: We start from a simple log-linear model. Then, we iteratively split the model components by duplicating and disturbing using a small ε. After each split, we perform the normal training procedure until convergence. Once the desired amount of densities is obtained and converged, training terminates.

4.1.3 Random Initialization

We start with the desired number of densities where all model parameters are initialized with random numbers between 0 and 1.

5 DEFORMATION-AWARE LOG-LINEAR MODELS

Here we use the configuration of the latent variable to model the deformation of an image as an alignment of its individual pixels to the pixels of the model.

Deformation-invariance for handwritten character recognition has been thoroughly investigated for various distance functions in the context of nearest neighbor classification [20, 37]. Our deformation model is inspired by the IDM which has been proposed by several authors independently under different names. For example, it has been described as "local perturbations" [37] and as "shift similarity" [29].

Here, we follow the formulation of [20]. The IDM aligns an image pixel-wise to a prototype image. To allow for efficient computation, no dependencies between alignments of neighboring pixels are considered. An image alignment $(xy)_{11}^{IJ}$ maps each pixel ij of image A of size $I \times J$ to a pixel $(xy)_{ij}$ in the prototype (image) B:

$$(xy)_{11}^{IJ} : (ij) \mapsto (xy)_{ij}, \forall i = 1, \ldots, I, j = 1, \ldots, J. \quad (10)$$

To restrict the number of possible alignments, a maximal warp-range W, i.e. the maximal displacement between ij and $(xy)_{ij}$, is defined. An example alignment is shown in Figure 1. For nearest neighbor classification, a distance d_{idm} between two images A and B is defined as:

$$d_{\text{idm}}(A, B) = \min_{(xy)_{ij}} \left\{ \sum_{ij} d_{\text{local}}(A_{ij}, B_{(xy)_{ij}}) \right\}. \quad (11)$$

To compute $d_{\text{idm}}(A, B)$, a minimizing alignment $\widehat{(xy)_{11}^{IJ}} = \arg\min_{(xy)_{11}^{IJ}} \left\{ \sum_{ij} d_{\text{local}}(A_{ij}, B_{(xy)_{ij}}) \right\}$ is computed. The alignment of a pixel ij to a pixel $(xy)_{ij}$ is restricted by warp-range W to a local image region:

$$|i - W| \leq W \text{ and } |j - W| \leq W \quad (12)$$

d_{local} is a local distance function comparing pixel (ij) in image A and pixel $(xy)_{ij}$ in image B by comparing pixels values or features thereof from local image regions around ij and $(xy)_{ij}$. A small example alignment is shown in Figure 1.

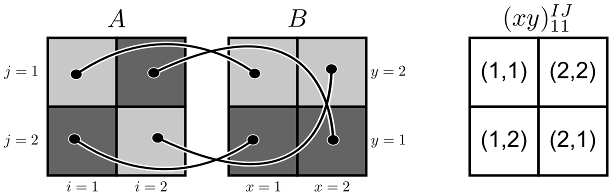


Fig. 1: **Image alignment.** Two images A and B of size 2×2 pixels and an alignment $(xy)_{11}^{IJ}$ between them. $(xy)_{11}^{IJ}$ specifies for every pixel (ij) in image A a location $(xy)_{ij}$ in image B; e.g. the dark gray pixel in image A at position $(i, j) = (2, 1)$ is mapped to the dark-gray pixel $(x, y) = (2, 2)$ in image B.

The deformation-aware log-linear models consider the full image alignment $(xy)_{11}^{IJ}$ as a latent variable which is marginalized out:

$$\begin{aligned} p_\theta(c|X) &= \sum_{(xy)_{11}^{IJ}} p_\theta(c, (xy)_{11}^{IJ}|X) && (13) \\ &= \frac{\sum_{(xy)_{11}^{IJ}} p_\theta(c, (xy)_{11}^{IJ}, X)}{\sum_{c'} \sum_{(x'y')_{11}^{IJ}} p_\theta(c', (x'y')_{11}^{IJ}, X)} && (14) \\ &= \frac{\sum_{(xy)_{11}^{IJ}} \exp\left(g_\theta(c, (xy)_{11}^{IJ}, X)\right)}{\sum_{c'} \sum_{(x'y')_{11}^{IJ}} \exp\left(g_\theta(c', (x'y')_{11}^{IJ}, X)\right)} && (15) \end{aligned}$$

We define the discriminant function $g_\theta(c, (xy)_{11}^{IJ}, X)$ for class c, a given image alignment $(xy)_{11}^{IJ}$, and image X as

$$g_\theta(c, (xy)_{11}^{IJ}, X) = \alpha_c + \sum_{ij} \left(\alpha_{cij(xy)_{ij}} + \lambda_{c(xy)_{ij}}^T X_{ij} \right), \quad (16)$$

where $\theta = \{\alpha_c, \alpha_{cij(xy)_{ij}}, \lambda_{c(xy)_{ij}}\}$ and α_c is a class bias. The $\alpha_{cij(xy)_{ij}}$ correspond to class-, position, and alignment depending deformation priors. A high value of $\alpha_{cij(xy)_{ij}}$ means that a pixel at position (i, j) is likely to be aligned to position (x, y) in class c. The alignment $(xy)_{11}^{IJ}$ aligns the observation X to the class-dependent weight vector $\lambda_{c(xy)_{ij}}$ which can be considered the normal of the decision hyperplane in a two class problem. In this model, the full alignment between the image X and the model parameters α and λ is considered as a latent variable (e.g. the entire alignment $(xy)_{11}^{IJ}$ at the right of Figure 1 is a latent variable). In the experiments, we show a visualization of the α and the λ parameters in Figure 8 and 9.

Note, that each pixel (i, j) in image X is represented by a D-dimensional feature vector which can e.g. consist of Sobel features extracted from a square local neighborhood around position (ij) (cf. Figure 2). The corresponding weight vectors $\lambda_{c(xy)_{ij}}$ are of the same dimensionality D. For each class c, we train $D \times IJ$ many λ-parameters.

This formulation involves a huge number of potential configurations of the latent variable (image alignments) but can be evaluated efficiently in the IDM since the alignments of the individual pixels are modeled independently. Starting from a sum over all possible deformations (denoted by $[(xy)_{11}^{IJ}]$), we rewrite the posterior such that the contributions of the individual pixels are

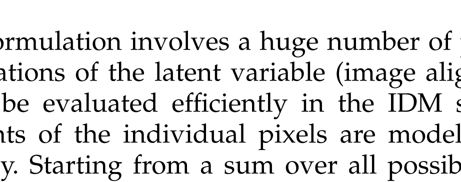


Fig. 2: Each pixel can be represented by a descriptor of Sobel features extracted from a local neighborhood (Figure from [20]).

evaluated independently:

$$\sum_{[(xy)_{11}^{IJ}]} p\left(c,(xy)_{11}^{IJ}|X\right) \tag{17}$$

$$= \frac{\sum\limits_{[(xy)_{11}^{IJ}]} \exp(\alpha_c)\exp(\sum_{ij}\alpha_{cij(xy)_{ij}} + \lambda^T_{c(xy)_{ij}}X_{ij})}{\sum\limits_{\tilde{c}}\sum\limits_{[\widetilde{(xy)}_{11}^{IJ}]} \exp(\alpha_{\tilde{c}})\exp(\sum_{ij}\alpha_{\tilde{c}ij\widetilde{(xy)}_{ij}} + \lambda^T_{\tilde{c}\widetilde{(xy)}_{ij}}X_{ij})} \tag{18}$$

$$= \frac{\exp(\alpha_c)\sum\limits_{[(xy)_{11}^{IJ}]}\prod_{ij}\exp(\alpha_{cij(xy)_{ij}} + \lambda^T_{c(xy)_{ij}}X_{ij})}{\sum\limits_{\tilde{c}}\exp(\alpha_{\tilde{c}})\sum\limits_{[\widetilde{(xy)}_{11}^{IJ}]}\prod_{ij}\exp(\alpha_{\tilde{c}ij\widetilde{(xy)}_{ij}} + \lambda^T_{\tilde{c}\widetilde{(xy)}_{ij}}X_{ij})} \tag{19}$$

$$= \frac{\exp(\alpha_c)\prod\limits_{ij}\sum\limits_{(xy)_{ij}\in\mathcal{W}(ij)}\exp(\alpha_{cij(xy)_{ij}} + \lambda^T_{c(xy)_{ij}}X_{ij})}{\sum\limits_{\tilde{c}}\exp(\alpha_{\tilde{c}})\prod\limits_{ij}\sum\limits_{\widetilde{(xy)}_{ij}\in\mathcal{W}(ij)}\exp(\alpha_{\tilde{c}ij\widetilde{(xy)}_{ij}} + \lambda^T_{\tilde{c}\widetilde{(xy)}_{ij}}X_{ij})} \tag{20}$$

This transformation reduces the complexity to evaluate this from summing over $|\mathcal{W}(i,j)|^{W\cdot H}$ products of $H\cdot W$ terms to $H \cdot W$ sums of $|\mathcal{W}(i,j)|$ terms, where $\mathcal{W}(i,j)$ is the area which has to be considered for potential alignments of the pixel at position (i,j). Using the default warp-range $W = 2$ of the IDM [20] this relates to evaluating a product over 256 sums of 25 terms instead of evaluating a sum over 25^{256} products of 256 terms for 16×16 pixel images.

To the best of our knowledge this is the first model in which all model parameters, including deformation priors $\alpha_{cij(xy)_{ij}}$ can be trained jointly with the other model parameters. This allows the model to learn which deformations are valid for which of the classes to account for intra-class variability and which deformations should not be allowed because they would allow for bridging the inter-class variability. In Figure 9 we visualize the learned $\alpha_{cij(xy)_{ij}}$ and it can e.g. be seen that it is possible to make a "0" wider and narrower but that it must not become too narrow (because otherwise it might turn into a "1").

To avoid evaluating a sum over the latent variable, we use the maximum approximation: to train a model with the maximum approximation, we apply the algorithm described in section 3.2. For the MAXSUM case, in step 1 for each training sample X_n the alignment $\widehat{(xy)}_{11}^{IJ}{}_{c_n}$ is determined that maximizes the discriminant function for the correct class:

$$\widehat{(xy)}_{11}^{IJ}{}_{c_n} := \arg\max_{(xy)_{11}^{IJ}} \left\{g_\theta(c_n,(xy)_{11}^{IJ},X_n)\right\} \tag{21}$$

This is efficient since the alignment can be computed for each pixel independently.

Analogously, in the MAXMAX-case, the alignment $\widehat{(xy)}_{11}^{IJ}{}_{c}$ maximizing the discriminant function has to be determined for each class independently. After these alignments have been determined for each training sample X_n, the training algorithm proceeds to step 2 (Section 3.2).

5.1 Deformation Prior Sharing

In our initial formulation, the $\alpha_{cij(xy)_{ij}}$ model the deformation priors separately for each class and pixel position. This leads to a large number of parameters modeling similar properties. To reduce the number of parameters and allow for sharing deformation information we propose five parameter sharing strategies over classes, positions, and deformations, respectively:

Full alpha. Deformation parameters are trained for each class, position, and possible deformation independently, i.e. $\alpha(c,i,j,i-x,j-y)$ is a function of the class c, the position (i,j) in the test image, and the deformation $(i-x,j-y)$. In this setup, we have a total of $C(IJ)(2W+1)^2$ deformation priors α to be trained.

Position independent. Deformation parameters are shared among the positions in the test image. In this case, $\alpha(c,i-x,j-y)$ is a function of the class c and the deformation $(i-x,j-y)$. In this setup, we have a total of $C(2W+1)^2$ deformation priors α.

Deformation independent. Deformation parameters are shared among the deformations, i.e. $\alpha(c,i,j,\delta(x=i\wedge y=j))$ is a function of the class c and the position (i,j) in the test image, furthermore it is only distinguished between no-deformation $(i-x=0\wedge j-y=0)$ and deformation $(i-x\neq 0 \vee j-y\neq 0)$. In this setup, we have $2C(IJ)$ deformation priors α.

Position and deformation independent. Deformation parameters are shared among positions in the test image and deformations. This is a combination of *"position independent"* and *"deformation independent"*. Here, $\alpha(c,\delta(x=i\wedge y=j))$ is a function only of the class c and the question whether it there is a deformation or not. In this case, we only have $2C$ deformation priors α, one for deformation and one for no-deformation per class.

Class independent. Deformation parameters are shared among the classes. This can be combined with any of the previous schemes. For the first case of full, but class-independent $\alpha(i,j,i-x,j-y)$ is a function of image position and every possible deformation, but independet of the class resulting in $(IJ)(2W+1)^2$ deformation priors α.

6 EXPERIMENTAL ANALYSIS

WE evaluated our methods on two well-known databases: The USPS dataset [34] and the MNIST dataset [24]. We used the smaller USPS dataset to tune our methods and thoroughly investigated the effects of different settings. Then we performed experiments on the MNIST dataset with the settings that turned out to work best on the USPS dataset. This allowed us (a) to reduce the required computations and (b) to avoid overfitting to a particular dataset.

The USPS Dataset.

The US Postal Service task is still one of the most widely used reference data sets for handwritten character recognition and allows for fast experiments due to its small size. The test set contains a large amount of image variability and is considered to be a 'hard' recognition task. The training and test sets consist of 7,291 and 2,007 observations respectively. All images are of size 16×16 pixels and scaled between 0.0 and 1.0.

The MNIST Dataset.

The modified NIST (MNIST) database can be considered the standard benchmark for handwritten character recognition. A large number of reference results are available. The MNIST data set is larger in size than the USPS data set. The training and test sets consist of 60,000 and 10,000 images of size 28×28 pixels, respectively. Also, all pixel values are scaled between 0.0 and 1.0.

Example images for both the USPS and the MNIST dataset are shown in Figure 3.

In the following we first present experimental results on the USPS dataset using LLMMs (Section 6.1) and using the deformation-aware log-linear models (Section 6.2). We tuned all model settings on the USPS dataset and then transfered these to the MNIST dataset (Section 6.3) to show the generalization capabilities and avoid tuning on the MNIST test set.

These datasets are well suited for this transfer since both are 10-class handwritten digit classification tasks on gray value images. The images in the MNIST dataset are slightly larger (20×20 pixels centered in a 28×28 pixel) and in MNIST there are more training and test images. While the larger image size might justify a larger warp-range, we did not observe big changes in informal experiments (which was also observed by Keysers et al. [20]). For significantly larger images, we would expect a larger warp-range to be appropriate. Note that we consider our experiments on the USPS dataset as preparatory experiments for the experiments on the MNIST data.

6.1 Log-linear Mixture Models

First we present experiments directly comparing the performance of Gaussian and LLMMs. The Gaussian mixtures were used to initialize the LLMMs following [16, 33]. We first trained a Gaussian mixture model, then transformed this into log-linear form, and then trained this model until convergence using the alternating optimization techniques. The results of these experiments are given in Figure 4, where the "0 split/1 density" per class results correspond to single Gaussians and plain log-linear models, accordingly. The experiments were performed with regularization factor $\gamma = 10^{-6}$ on untransformed pixel values as features. It can be observed that for both models the error rate decreases with increasing numbers of densities. This effect is much stronger for the generative GMD model. For the LLMM a slight overfitting effect can be observed for high numbers of densities. However, the error rates of the LLMM clearly outperform the GMD even with far smaller models due to the discriminative training criterion. The GMD-models were trained generatively following the maximum likelihood approach and the LLMMs were trained according to the MMI criterion. Since these experiments suggest that more than "5 splits/32 densities" per class are not helpful, we restricted most of the upcoming experiments to 32 densities.

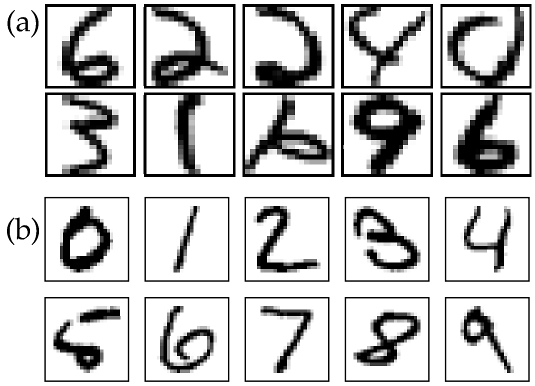


Fig. 3: Example images for (a) the USPS and (b) the MNIST dataset.

6.1.1 Maximum Approximation

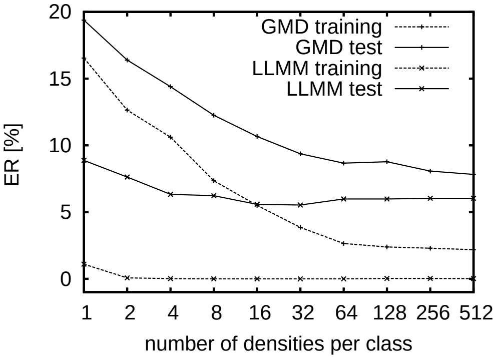
In the experiments described above, we used the MaxMax variant. During training we alternated between updating the latent density alignments and reestimating the model parameters θ using 20 iterations of LBFGS.

We investigated the impact of the maximum approximation. The results of the experiments using the three different approaches (MaxMax, MaxSum, and SumSum) are shown in Figure 5. Interestingly the use of the maximum approximation has a positive impact on the results. The results of the MaxMax experiments are best. The results from the MaxSum experiments are also better than those from using SumSum. This is an interesting result since for the MaxMax case no theoretical convergence guarantee can be given.

To obtain more insight into the convergence of the training using the three different approaches to the maximum approximation, we measured the error rate and the score of the optimized training criterion as a function the first 20 iterations. Figure 6 shows the plot for the three variants of the maximum approximation using 16 model components per class. It can be observed that for the SumSum and the MaxSum case the training criterion improves monotonously while it goes up and down for the MaxMax case. However, even for the MaxMax case, training converges because the improvements when updating the parameters θ are consistently larger than the deteriorations of the criterion when updating the assignment of training observations to model components. Over all the training using the maximum approximation in numerator and denominator converges faster and leads to better results. Therefore, in all succeeding experiments we use the MaxMax approach.

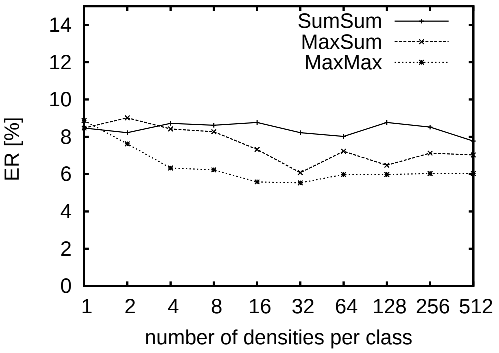
6.1.2 The Effect of Initialization

So far, we initialized the LLMMs using a mixture of Gaussians with the appropriate number of densities.



split	dens. / class	GMD ER [%] training	GMD ER [%] test	LLMM ER [%] training	LLMM ER [%] test
0	1	16.5	19.4	1.1	8.9
1	2	12.6	16.4	0.1	7.6
2	4	10.6	14.4	0.0	6.3
3	8	7.4	12.3	0.0	6.2
4	16	5.5	10.7	0.0	5.6
5	32	3.9	9.4	0.0	5.5
6	64	2.6	8.7	0.0	6.0
7	128	2.4	8.8	0.0	6.0
8	256	2.3	8.1	0.0	6.0
9	512	2.2	7.8	0.0	6.0

Fig. 4: Training and test error rates [%] using Gaussian mixtures and LLMMs on the USPS dataset with different numbers of densities.



split	dens. / class	ER [%] SumSum	ER [%] MaxSum	ER [%] MaxMax
0	1	8.5	8.5	8.9
1	2	8.2	9.0	7.6
2	4	8.7	8.4	6.3
3	8	8.6	8.3	6.2
4	16	8.8	7.3	5.6
5	32	8.2	6.1	**5.5**
6	64	8.0	7.2	6.0
7	128	8.8	6.5	6.0
8	256	8.5	7.1	6.0
9	512	7.8	7.0	6.0

Fig. 5: Test error rates [%] on the USPS dataset using the different approaches to the maximum approximation to train LLMMs. SumSum = no maximum approximation, MaxSum = maximum approximation in the numerator, MaxMax = maximum approximation in numerator and denominator.

In the following, we evaluate other initialization methods (Section 4.1). Table 2 shows the results of these experiments. Using only a few (i.e. 8 or less) mixture components, the models obtained from splitting discriminative models outperform those initialized from Gaussian mixtures. For the models with many (16 or more) components, those initialized from the Gaussian mixtures perform best. In general random initialization performs worse than the other methods. We assume that the number of local optima of the training criterion grows with the number of densities and that the initialization using a Gaussian mixture often leads to finding better local optima than using random initialization or discriminatively split models.

TABLE 2: Test error rates [%] on the USPS dataset obtained using LLMMs with different initializations of the training. (experiments with random initialization were repeated three times and results are averaged. The variance of these experiments was $< 10^{-5}$).

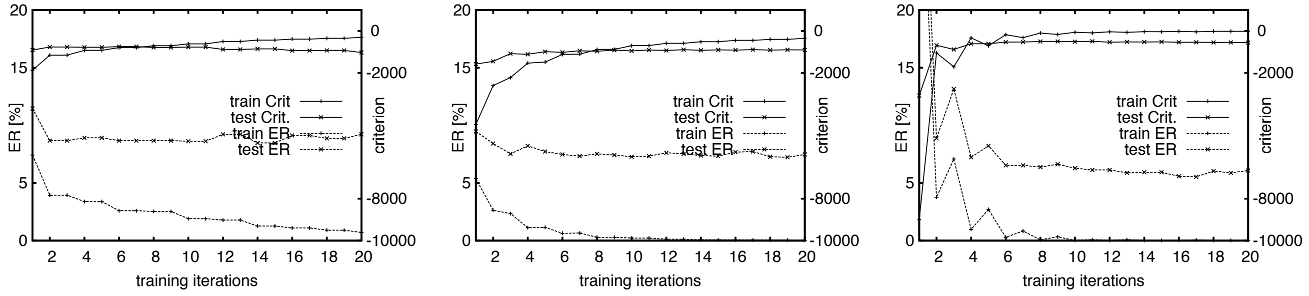
splits	dens. / class	ER [%] using initialization GMD	split	random
0	1	8.9	8.9	8.9
1	2	7.6	6.8	7.9
2	4	6.3	6.1	6.8
3	8	6.2	6.1	6.3
4	16	5.6	6.1	6.4
5	32	5.5	6.1	5.9

6.1.3 Different Features

Since we used Sobel gradient features in the deformation-aware log-linear models (following [20]), we also evaluated them here. Here we investigated plain gray values, Sobel features, absolute values of Sobel features, squared Sobel features, second order features over the entire image, and second order features over local (5x5 pixels) image regions (cf. Section 2). That is, we use the product of the gray values of every pair of pixels within a 5x5 neighborhood as features.

These experiments are performed using 0-5 splits corresponding to 1-32 densities per class. The models were initialized from a Gaussian mixture model. The results of these experiments are given in Table 3. Note that using plain Sobel features does not affect the result since linear transformations of the features cancel out and thus have no impact.

From the results it can be seen that better features lead to a significant improvement for models with few components. In models with many components, the effect is very small, and in some cases, even overfitting can be observed. Overall, the best result is obtained using absolute values of Sobel features.



(a) SumSum (b) MaxSum (c) MaxMax

Fig. 6: Convergence of the training of LLMMs on the USPS dataset using the different variants of the maximum approximation. SumSum = no maximum approximation, MaxSum = maximum approximation in the numerator, MaxMax = maximum approximation in the numerator and denominator.

TABLE 3: Test error rates [%] on the USPS dataset using LLMMs with different image features. Note that using untransformed Sobel features leads to the same results as pure gray values since they are only a linear transformation of the gray values. Row "Dim." is the dimensionality of the feature space in which the parameters are trained.

splits	$\frac{\text{dens.}}{\text{class}}$	gray val.	|Sobel|	Sobel^2	2$^{\text{nd}}$	(5x5)-2$^{\text{nd}}$
Dim.		256	512	512	32896	6400
0	1	8.9	5.9	6.2	5.7	5.8
1	2	7.6	5.4	5.7	5.1	6.0
2	4	6.3	5.0	5.7	5.0	5.7
3	8	6.2	**4.4**	5.2	5.1	5.4
4	16	5.6	4.6	5.2	5.3	5.4
5	32	5.5	4.7	5.6	5.4	5.6

From these experiments, we observed that LLMMs are robust models which lead to good results. The results are improved from 8.6% error rate for simple log-linear models to 5.5% error rate for LLMMs. By choosing the right number of components, the maximum approximation in nominator and denominator, a good initialization, and proper features it is possible to obtain an error rate of 4.4%. In Section 6.3, we transfer the found settings to the MNIST task and show that these settings generalize well to this dataset.

6.2 Deformation-aware Log-linear Models

In the following we experimentally investigated and tuned the deformation-aware log-linear models on the USPS dataset. First we investigated the warp-range and different image features for alignment and classification. Then we investigated the effects of the different parameter sharing schemes presented. We also investigate different training methods and initializations and compare the obtained results to an SVM and a kernelized log-linear model with IDM distance kernel. While the SVM can determine a relatively sparse set of support vectors (30-50% of the training vectors), the kernelized log-linear model will always give a weight to all training observations and thus is computationally expensive.

6.2.1 Warp-range

One crucial setting of the IDM is the warp-range W, which controls the maximal horizontal and vertical displacement for each pixel. In Figure 7 the effect of different warp-ranges on the error rate on the USPS dataset is shown. In these experiments we used simple Sobel features. It can be seen that beyond a warp-range of $W = 2$ hardly any improvement is possible and since smaller warp-ranges W have faster runtimes for the experiments in the following, we kept $W = 2$, which is also concordant with the results Keysers et al. [20] report.

6.2.2 Features

Keysers et al. [20] observed that local context is essential to determine good alignments and they found that subwindows of Sobel features performed best. Here, we investigated the impact of different local descriptors and sub-windows on the classification performance. The results of these experiments are shown in Table 4. Note that here, in contrast to [20], this changes the entire model and not just the distance function.

We compared eight different setups: simple gray values, Sobel features, absolute values of Sobel features, and a combination of Sobel and absolute Sobel. Each feature setup has been evaluated with and without 3×3 subwindows. It can be observed that using Sobel features, scaled from -1 to 1, leads to a significant improvement over using just gray values and there is hardly a difference in the test error rate whether local context is used or not. Absolute Sobel values do not reach the performance of plain Sobel features as they lose the direction of the edge information. Nonetheless, combining the two improves the result because the combination contains both improved features for alignment as well as nonlinear combinations of the original features. The observation that Sobel features are important to determine good alignments is consistent with the observations by

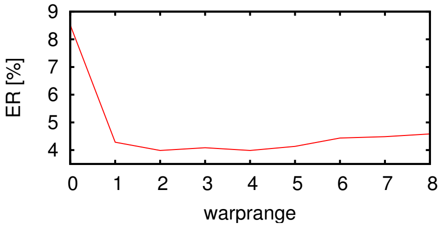


Fig. 7: The effect of different warp-ranges on the test error rate [%] on the USPS test data [9].

Keysers et al. [20]. It can be observed that the use of sub-windows leads to a minor improvement when using the combined Sobel descriptors. Due to the minor improvements using the feature combination but nonetheless greatly increased training effort, we used simple Sobel features for further investigations and re-combined the best approaches in Section 6.3 for the MNIST dataset.

Figure 8 shows a visualization of the λ_{cxy} parameters from an experiment with Sobel features and without local-context. λ_{cxy} are averaged over the horizontal and vertical Sobel features. A bright pixel in the λ_{cxy} in Figure 8 denotes that our model expectes a bright-to-dark (left-to-right or top-to-bottom) gradient in an image of class c at position (x, y), and a dark pixel denotes an dark-to-bright gradient (for characters written in dark ink on white background). For most classes, the structure is well recognizable with an exception for the classes 4 and 7 which have stronger variations than the other classes which makes it more difficult to interpret the prototypes.

6.2.3 The Deformation Parameters

In this section, we first analyzed the $\alpha_{cij(xy)_{ij}}$ parameters learned and then we evaluated the different deformation parameter sharing strategies described in Section 5.1.

We visualized the learned $\alpha_{cij(xy)_{ij}}$ from the previous experiments in Figure 9. For most classes the structure of the class is clearly visible. For example, for class "0", it is good to move away from the center (and not inwards). Movements on the outline of the zero are all equally likely. For class "1", moving up and down in the middle, but not to the left and right is allowed. On the left side of the image, moving down is not desired and on the right side, moving up is not desired. For class "9", there are many high values on the bottom left of the top circle which is a very common variability in writing style.

TABLE 4: The impact of different local features and local context on the test error rate [%] when using deformation-aware log-linear models on the USPS dataset [9]. D is the dimensionality of the feature vectors representing the individual pixels (cf. Section 5).

local context used:	no			yes		
Features	train	test	D	train	test	D
gray values	2.6	7.6	1	0.5	7.6	9
Sobel	0.0	4.0	2	0.0	4.0	18
abs(Sobel)	0.8	4.9	2	0.2	4.8	18
Sobel + abs(Sobel)	0.0	3.8	4	0.1	3.6	36

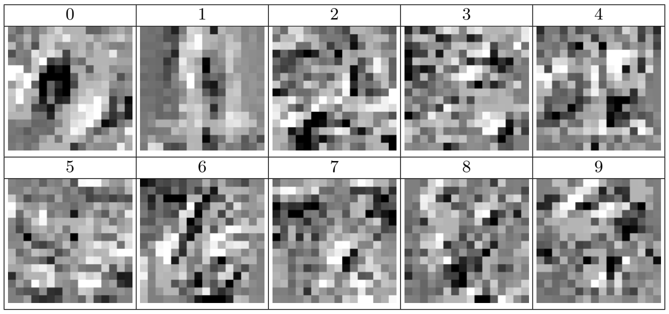


Fig. 8: Visualizations of the λ_{cxy} from an experiment on the USPS dataset.

Table 5 shows the results obtained using the different strategies for deformation parameter sharing described in Section 5.1. It can be observed that, although the number of parameters is significantly reduced, the error rates on the test data are only slightly affected. This shows that it is not necessary to have position- and deformation-specific deformation priors but that most of the relevant deformation information can be stored in the λ-parameters. Thus we assume that the models with shared deformation parameters generalize better. The observation that the number of deformation papers does not have a big influence on the results confirms the observation from Keysers et al. [20] that the nearly parameter-free IDM, while one of the simplest deformation models, achieves very competitive resuls.

6.2.4 Initialization and Alternating Optimization

The deformation-aware log-linear model can be rewritten as a Gaussian model analogously to the mixture model [33]. Thus it is possible to initialize the model from a deformation-aware Gaussian model [20]. Since we cannot guarantee convergence to the *global* optimum of the parameters, we considered three different ways to initialize the model in this section: initialization from a non-deformation invariant log-linear model, initialization from a deformation-aware generative Gaussian model, and initialization of all parameters with zeros.

For these alternatives, we compared the results using different training schemes. In the scheme "*fixed alignment*", we initialized the model, determined an alignment of the training data to the init model, and kept this alignment fixed until convergence. In the scheme "*alternating optimization*", we used the alternating optimization procedure described before. We initialized the model and alternated between re-aligning and parameter updates until convergence. The results of these experiments are given in Table 6.

Interestingly, the final result is nearly independent of the initialization, which indicates that the alternating optimization is able to find a good set of parameters independent of the starting point. Only for the model initialized from the deformation aware Gaussian model, the alternating optimization has no effect. We believe

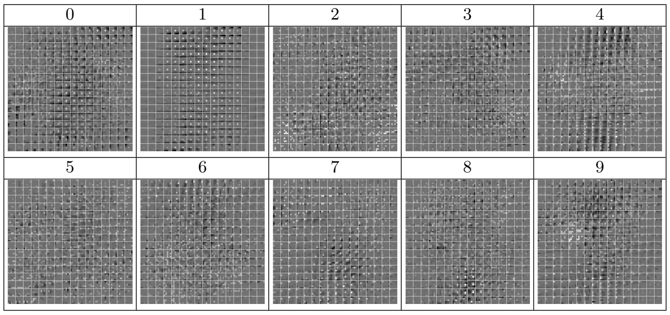


Fig. 9: Visualizations of the $\alpha_{cij(xy)_{ij}}$ from an experiment on the USPS dataset.

TABLE 5: Training and test error rates [%] on the on the USPS dataset using the different deformation prior sharing methods along with the number of deformation parameters and the number of parameters of the entire model [9].

Sharing scheme	def. param	total param	test ER
full alphas	64000	69130	4.0
class indep.	6400	11530	3.8
def. indep.	5120	10250	3.9
pos. indep.	250	5380	4.1
pos. &def. indep.	20	5150	3.9
+ class indep.	2	5132	3.9

that this model is stuck in a strong local optimum. However, if alternating optimization is not used, the other two models are clearly worse, which again highlights the importance of the alternating optimization. The training time for the different initializations is similar where generally the model initialized with a log-linear model needs fewer iterations than the other two.

6.3 Experiments on the MNIST Dataset & Comparison to the State-of-the-Art

In this section we transfer the results obtained on the USPS dataset to the MNIST dataset. For that purpose, we chose the settings which performed best on the USPS dataset, trained the corresponding models on the MNIST training set and evaluated on the MNIST test set. Only for very few settings (such as the number of splits of the mixture models), we performed multiple experiments on the MNIST dataset to demonstrate the stepwise improvement of the results.

The parameters that were transfered for the LLMMs are the number of densities (2,4,32) and the image features used (squared Sobel features). For the deformation-aware log-linear models we transfered the warp-range ($W = 2$), the image features (Sobel horizontal and vertical), the deformation prior sharing (position and deformation independent $\alpha(c, \delta(x = i \wedge y = j))$), and the size of the local context (=1). For both LLMMs and deformation-aware log-linear models we transfered which variant of the maximum approximation we use (MAXMAX) and the regularization factor $\gamma = 10^{-6}$.

TABLE 6: Training and test error rates [%] on the USPS dataset obtained using the different initializations with and without alternating optimization [9].

		fixed align.		altern. opt.	
Init.	initial	train	test	train	test
Gaussian	6.5	0.7	4.6	0.7	4.6
log-linear	8.3	0.1	5.9	0.0	4.0
zero init	-	1.9	8.3	0.0	4.0

The results for the experiments on both datasets using LLMMs and deformation aware log-linear models along the number of parameters of the respective models are given in Table 7. The first block of results shows the results obtained using the LLMM approach, the second block contains the results obtained using the deformation-aware log-linear model. Additionally to the classification error rate of the individual approaches on both the USPS and the MNIST dataset, we give the total number of parameters and an estimate of the runtime of the respective model relative to the performance of the simplest model: a single log-linear model.

For the LLMMs, we evaluated models with 2, 4, and 32 densities per class using gray values only and using squared horizontal and vertical Sobel features. Each of these models was initialized using a mixture of Gaussians and the training was performed using the MAXMAX-setup. Analogously to the experiments on the USPS dataset, it can be observed that for the models with only few densities, the use of the Sobel features leads to significant improvement whereas the improvement is much smaller for the models with sufficient densities.

For these LLMMs, the number of parameters as well as the runtime grows linear in the number of model densities. It can be observed that the improvement with a higher number of densities is slightly higher on the MNIST dataset than on the USPS dataset which is probably due to less overfitting problems since the amount of training data is much bigger. In general for the mixtures, it can be observed that the models which lead to improvements on the USPS dataset also lead to improvements on the MNIST dataset.

For the deformation-aware log-linear models, a combination of Sobel and absolute Sobel with *position and deformation independent α sharing* improves the results. Additionally using local context does not lead to an improvement but rather to overfitting. All improvements using settings optimized on the USPS dataset consistently transfer to improvements on the MNIST database, showing the good generalization capabilities of our model.

Table 8 shows comparison results from the literature, some of them very simple, some of them state-of-the-art. The first model is a simple single Gaussian classifier with diagonal covariance matrix, which can also be seen as a naïve Bayes classifier. This one is, along with its discriminative counterpart, the simple log-linear model, the fastest approach. In these two approaches, a test observation only needs to be compared to one prototype per class.

The single Gaussian model with IDM, already per-

TABLE 7: Comparison of test error rates [%], number of parameters and run-time of the LLMMs and the deformation-aware log-linear model to the state-of-the-art for the USPS and the MNIST dataset.

	USPS		MNIST		
Model	# param.	ER	# param.	ER	run-time factor
log-linear model	2570	8.2	7850	7.4	1
+ abs(SobelHV)	5130	5.5	15690	3.0	2
LLMMs on gray values					
1 split/2 densities	5140	7.6	15700	6.7	2
2 split/4 densities	10280	6.3	31400	5.4	4
5 split/32 densities	82240	5.5	251200	**3.5**	32
LLMMs on squared Sobel features					
1 split/2 densities	10260	5.7	31380	3.0	4
2 split/4 densities	20520	5.7	62760	2.5	8
5 split/32 densities	164160	5.6	502080	2.1	64
deform. aware log-lin. model (with Sobel)	69130	4.0	211690	1.6	50
+ abs(SobelHV)	74250	3.8	227370	1.3	100
+ deformation parameter sharing	10340	3.6	31390	1.4	100
+ local context	92190	3.7	282270	1.5	900

TABLE 8: Comparison to the state-of-the-art for the USPS and the MNIST dataset. When two run-time factors are given, the first is for USPS, the second for MNIST.

		USPS		MNIST		
Model	ref.	# param.	ER	# param.	ER	run-time factor
single Gaussians		2560	18.5	7840	18.0	1
single Gaussians + IDM	[20]	2560	6.5	7840	5.8	50
nearest neighbor	[20]	1866496	5.6	47040000	3.1	729/6000
nearest neighbor + IDM	[20]	1866496	2.4	47040000	0.6	36455/300000
SVM		658177	4.4	15411905	1.5	256/1963
SVM with IDM kernel	[13]	530705	2.8	-	0.7	10300/100000
kernel log-linear model (with IDM kernel)		1873797	5.3	47100010	-	36455/300000
DBN	[17]	640610	-	1665010	1.3	210/ 220
convolutional neural network	[36]	-	-	180580	0.4	-/25

forms much better but an IDM comparison is about 50 times as expensive as a simple component-wise comparison (due to the use of Sobel features and a deformation window of 5×5 pixels ($W = 2$)).

The next comparison result is a nearest neighbor classifier which is frequently used as a simple baseline and already performs reasonably well. However, a problem with this approach is that at test time, the runtime depends on the number of training samples. If the nearest neighbor uses the IDM to compare the images, it obtains one of the best published results on both datasets. Note, that the use of computationally more complex image alignment models can lead to a small additional improvement [20].

We also compare the performance to SVMs with and without explicit deformation modeling. The standard SVM uses a Gaussian radial basis function kernel and SVM+IDM is the method from [13]. For SVMs, the number of operations to classify an observation depends on the number of support vectors. In these two datasets, the number of support vectors is typically about 30% of all training samples and thus these methods need about one third of the runtime of the corresponding nearest neighbor classifier. For the SVM incorporating the IDM, we used a radial basis kernel of the symmetric variant of the IDM distance defined as $K_{\text{IDM}}(X, V) = \exp\left(-\frac{\gamma}{2}(d_{\text{idm}}(X, V) + d_{\text{idm}}(V, X))\right)$, since non-symmetric kernels cause problems in training SVMs. Although this kernel is not positive definite, it was observed by Haasdonk [13] that in practice the training terminates with a good result. We note that the symmetric IDM distance is known to perform worse than the asymmetric one in nearest-neighbor experiments (3.4% instead of 2.4%). Nonetheless, the support vector machines obtain excellent results on both datasets, where the results on the MNIST database have been reported by [13] and the results on the USPS database have been obtained using our own implementation.

Furthermore, we used the same IDM kernel as we used in the SVM also in a kernelized log-linear model, which then learns a weight for each training observation. Apart from a prohibitive runtime at test time, at training time, the full kernel matrix between all pairs of training samples has to be computed and stored which requires a large amount of memory. The performance of this approach is neither competitive with our deformation-aware log-linear model nor with the respective SVMs.

For further comparison we give two state-of-the-art results on the MNIST database using deep belief networks and convolutional neural networks. Both are based on neural networks where the deep belief network is proposed as a general learning technique [17] which does not incorporate prior knowledge about the data. It does not even assume that the observations are images. The convolutional neural networks were designed with digit recognition in mind and are trained from a huge amount of automatically deformed training data [36]. The convolutional neural network obtains one of the best published results on the MNIST dataset despite its small size and efficient classification stage. However, the training phase for this network is computationally very expensive because during training the training data is automatically deformed several thousand times.

As an overview, it can be seen that our method compares favorably well to other methods. In particular in comparison with the other fast methods, only the convolutional neural networks, which are difficult to create and optimize, outperform our methods with a comparable computation time. Furthermore, the small number of parameters in our model is a good indicator for its generalization performance which is underlined by the successful transfer of the settings from the USPS dataset to the MNIST dataset.

7 Summary

We presented how latent variables can be incorporated into log-linear models and two direct applications of this approach: log-linear mixture models and deformation-aware log-linear models. We presented an efficient and effective training algorithm for these approaches and showed that they work well in practice. Both approaches were demonstrated to perform well on two widely-used image classification tasks and obtain competitive results with other approaches albeit only very few parameters have to be trained. The deformation-aware log-linear model is the first approach to train the deformation parameters jointly with the remaining model parameters. By sharing deformation parameters among pixels, the number of parameters can be further reduced resulting in improved generalization.

References

[1] J. Anderson. Logistic discrimination. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of Statistics 2*, pages 169–191. North-Holland, 1982.

[2] O. Barndorff-Nielsen and P. Jupp. Approximating exponential models. *Annals of the Institute of Statistical Mathematics*, 41(2):247–267, 1988.

[3] O. Bender, F. Och, and H. Ney. Maximum entropy models for named entity recognition. In *7th Conference on Computational Natural Language Learning*, pages 148–152, Edmonton, Canada, May 2003.

[4] J. C. Bezdek and R. J. Hathaway. Convergence of alternating optimization. *Neural Parallel Scientific Computation*, 11(4):351–368, 2003.

[5] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46 (1):131–159, 2002.

[6] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, Oct. 1972.

[7] D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1-3): 161–190, 2002.

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2009.

[9] T. Gass, T. Deselaers, and H. Ney. Deformation-aware log-linear models. In *Symposium of the German Association for Pattern Recognition (DAGM)*, LNCS, Jena, Germany, Sept. 2009.

[10] P. V. Gehler and S. Nowozin. Let the kernel figure it out: Principled learning of pre-processing for kernel classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[11] A. Gunawardana and W. Byrne. Convergence theorems for generalized alternating minimization procedures. *Journal of Machine Learning Research*, 6: 2049–2073, 2005.

[12] A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt. Hidden conditional random fields for phone classification. In *International Conference on Spoken Language Processing*, pages 117 – 120, Lisbon, Portugal, Sept. 2005.

[13] B. Haasdonk. *Transformation Knowledge in Pattern Analysis with Kernel Methods.* PhD thesis, Albert-Ludwigs-Universität Freiburg, 2005.

[14] B. Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–492, 2005.

[15] B. Haasdonk and D. Keysers. Tangent distance kernels for support vector machines. In *International Conference on Pattern Recognition*, pages 864–868, Quebec City, Canada, Sept. 2002.

[16] G. Heigold, P. Lehnen, R. Schlueter, and H. Ney. On the equivalence of Gaussian and log-linear HMMs. In *Interspeech*, Brisbane, Australia, Sept. 2008.

[17] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[18] T. Jebara. *Machine Learning: Discriminative and Generative*. Kluwer, 2003.

[19] D. Keysers, W. Macherey, H. Ney, and J. Dahmen. Adaptation in statistical pattern recognition using tangent vectors. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 26(2):269–274, Feb. 2004.

[20] D. Keysers, T. Deselaers, C. Gollan, and H. Ney. Deformation models for image recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(8):1422–1435, Aug. 2007.

[21] S. Kullback. Estimating and testing interaction parameters in the log-linear model. unpublished manuscript, 1971.

[22] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting

and labeling sequence data. In *ICML*, 2001.

[23] T. K. Landauer, D. McNamara, S. Dennis, and W. Kintsch, editors. *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum Associates, 2007.

[24] Y. LeCun and C. Cortes. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998.

[26] D. Liu and J. Nocedal. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–528, 1989.

[27] R. Memisevic and G. Hinton. Unsupervised learning of image transformations. In *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, USA, June 2007.

[28] T. P. Minka. A comparison of numerical optimizers for logistic regression. Technical report, Microsoft Research, Oct. 2003. revised Nov 2004.

[29] S. Mori, K. Yamamoto, and M. Yasuda. Research on machine recognition of handprinted characters. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 6(4):386–405, 1984.

[30] F. Och and H. Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, PA, USA, July 2002.

[31] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(10):1848–1852, 2007.

[32] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb. 1989.

[33] L. K. Saul and D. D. Dee. Multiplicative updates for classification by mixture models. In *Neural Information Processing Systems Conference*, volume 14, pages 897–904, Cambridge, MA, USA, 2002.

[34] B. Schölkopf. The USPS dataset. ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data/.

[35] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, USA, 2002.

[36] P. Simard. Best practices for convolutional neural networks applied to visual document analysis. In *7th International Conference on Document Analysis and Recognition*, pages 958–962, Edinburgh, Scotland, Aug. 2003.

[37] S. Uchida and H. Sakoe. A survey of elastic matching techniques for handwritten character recognition. *IEICE Transactions on Information and Systems*, E88-D(8):1781–1790, 2005.

[38] T. Weyand, T. Deselaers, and H. Ney. Log-linear mixtures for object class recognition. In *British Machine Vision Conference*, 2009.

Thomas Deselaers Thomas Deselaers is a researcher at the the Computer Vision Laboratory of ETH Zurich. He received his diploma and his PhD degree from RWTH Aachen University in Aachen, Germany in 2004 and 2008, respectively. From March 2004 to December 2008 he was a full time researcher at the Human Language Technology and Pattern Recognition Group of the Computer Science Department of RWTH Aachen University, where he has been the head of the image processing and understanding group since 2005.

Tobias Gass Tobias Gass is a PhD student at the Computer Vision Laboratory of ETH Zurich. Before that he was a researcher at the Human Language Processing Group of the Computer Science Department of the RWTH Aachen University in Aachen, Germany. He received his diploma in computer science from RWTH Aachen University in 2009. In 2006 he was a visiting student researcher at the Medical Imaging Group of the University Hospitals Geneva in Geneva, Switzerland. His research interests are image and object recognition, discriminative modeling and data mining.

Georg Heigold Georg Heigold is a researcher at the Human Language Technology and Pattern Recognition Group of the Computer Science Department of RWTH Aachen University where he received his PhD degree in 2010. He received the Diploma degree in physics from ETH Zurich, Switzerland in 2000. He worked as a software engineer at De La Rue, Bern, Switzerland from 2000 to 2003. Since 2004, he has been with the Computer Science Department of RWTH Aachen University in Aachen, Germany. In 2008, he was a research intern at Microsoft research in Redmond, USA. His research interests include automatic speech recognition, discriminative training, and log-linear models.

Hermann Ney received the Dipl. degree in physics from the University of Goettingen, Germany, in 1977 and the Dr.-Ing. degree in electrical engineering from the TU Braunschweig (University of Technology), Germany, in 1982.

In 1977, he joined Philips Research Laboratories (Hamburg and Aachen, Germany) where he worked on various aspects of speaker verification, isolated and connected word recognition and large vocabulary continuous-speech recognition. In 1985, he was appointed head of the Speech and Pattern Recognition group. In 1988-1989 he was a visiting scientist at AT&T Bell Laboratories, Murray Hill, NJ. In July 1993, he joined RWTH Aachen (University of Technology), Germany, as a professor for computer science.

His work is concerned with the application of statistical techniques and dynamic programming for decision-making in context. His current interests cover pattern recognition and the processing of spoken and written language, in particular signal processing, search strategies for speech recognition, language modeling, automatic learning and language translation.