
The Effect of Image Interpolation on the Generalization of Generative and Discriminative Models

William Watson
The Johns Hopkins University
billwatson@jhu.edu

Abstract

In statistical classification, we can either take the *generative* or *discriminative* approach to modeling. In computer vision, rescaling techniques known as *image interpolation* are analogous to dimensionality reduction. We explore the generalization and tractability of *generative-discriminative pairs* when interpolating images to smaller sizes.

1 Introduction

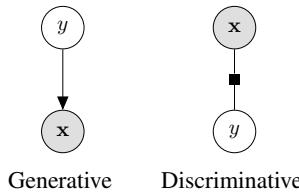
Probabilistic graphical models have many applications in computer science, ranging from image segmentation to named entity recognition. However, these models can become unwieldy when our feature space \mathbf{x} grows. In the case of images, as we increase the number of pixels, the total number of raw features grows exponentially. For square rasterized images, the total number of raw features is $3 \times \text{size}^2$.

This leads to issues of tractability for both parameter estimation and inference. This paper concerns itself with using image interpolation as a dimensionality reduction technique to make learning tractable. We vary the dimensionality to examine the generalization ability of our models and the efficiency of parameter estimation. We describe our data in Section 4, our experimental procedure in Section 7, and image interpolation (Section 3).

We experiment with two modeling approaches: *generative* and *discriminative* learning. Both approaches are outlined in Section 2. Our models are formally described in Sections 5 and 6. Our results are presented in Sections 8 and 9).

2 Statistical Classification

Classification is the task of assigning a label y to a set of observed features \mathbf{x} . Yet the way we approach modeling these relationships can be broken down into either the *generative* or the *discriminative* approach.



The generative approach models the joint distribution $p(y, \mathbf{x})$, and can assign labels through Bayes rule [9].

$$\hat{y} = \operatorname{argmax}_y p(y, \mathbf{x}) = \operatorname{argmax}_y p(y) \cdot p(\mathbf{x}|y) \quad (1)$$

Generative models are attractive in that distributions are learned over the feature space for individual classes. However, an important limitation mentioned by McCallum et al. is that modeling a distribution *per feature* quickly becomes intractable as the dimensionality of our observed variables \mathbf{x} grows [8]. While simple models can mitigate these issues by assuming independence among the features, allowing complex dependencies between inputs may result in increased performance.

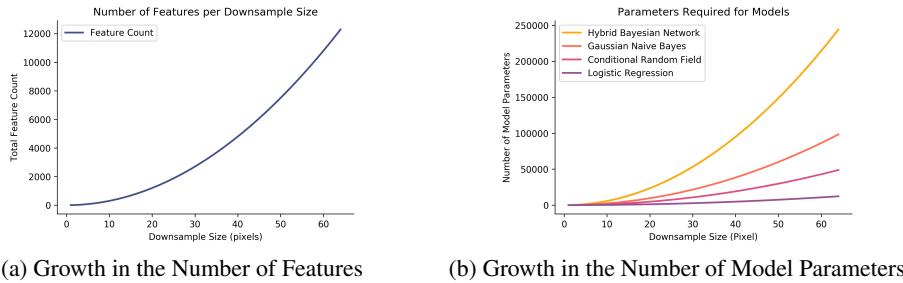
An alternative approach is to model the conditional probabilities directly, ignoring the feature distributions. This is the discriminative approach, and is sometimes referred to as a *distribution-free classifier*. By ignoring the feature distributions, parameters are learned only on the conditional likelihood $p(y|\mathbf{x})$, resulting in a compact model that can handle large feature spaces, as dependencies are ignored [8].

To compare generative and discriminative modeling, one can experiment with pairs of classifiers that can be considered analogous to each other. More formally, a *generative-discriminative pair* is a parametric family of probabilistic models that can either be fit to maximize the joint probability $p(y, \mathbf{x})$ or the conditional likelihood $p(y|\mathbf{x})$ [9]. The simplest pairing is the Naive Bayes classifier (generative) and Logistic Regression (discriminative). This paradigm can be extended to sequential and general models to form more pairs. We discuss our experimental pairs in Sections 5 and 6.

3 Image Interpolation

The most common problem encountered in machine learning is the *curse of dimensionality*. Essentially, as our feature space \mathbf{x} grows in size, we require a larger number of parameters and samples to effectively learn. This can quickly become intractable in complex models, and reasonably difficult in simple ones. We can see the growth in features as we increase the image size in Figure 2. One method to reduce the number of features for a data set is Principle Component Analysis, which projects data to a smaller dimension with components that maximize the variance. However, interpretability is lost, and the structure of the painting is unintelligible.

Figure 2: As the image size increases, the problem quickly becomes intractable



(a) Growth in the Number of Features

(b) Growth in the Number of Model Parameters

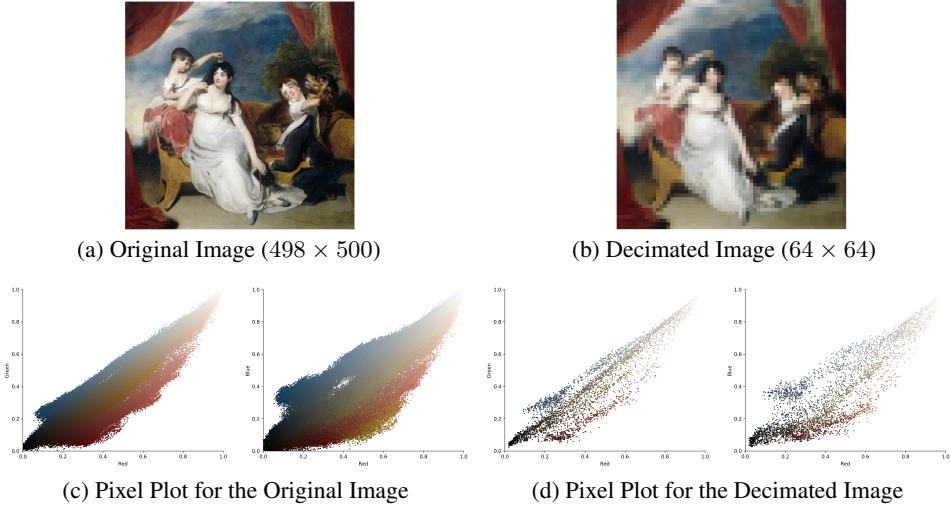
Image interpolation offers a convenient way to downsample images according to their local neighborhood, aggregating otherwise noisy estimates into a single value. This may be beneficial for classification tasks, as we combine a method analogous to dimensionality reduction while preserving the structure of the original content. Interpolation can be used to both decimate and expand the image. Several methods exists for this task, including linear, bicubic, nearest neighbor, and *area relation* [1].

The preferred method for image decimation, or downsampling, is resampling using pixel area relation as it gives moire'-free results. We can see in Figure 3 that the feature space of the pixels in relation to their original values is comparable, albeit with information loss.

4 Data

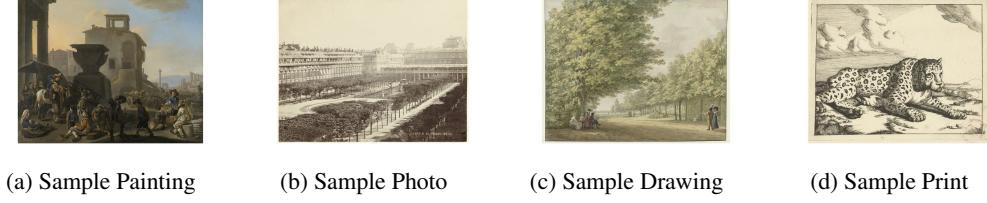
Our data is drawn from a subset of the Rijkmuseum Dataset [4]. In order to have enough samples for a challenging classification task, we elected to perform type categorization. We used the top 4 art

Figure 3: Results of Interpolating a 249,000 pixel image to a 4,096 pixel image.



types: paintings, photos, drawings, and prints. Each category is limited to the size of the smallest set, which is 2,269 photos. Hence we have a dataset of 9,076 images split evenly between the 4 classes. For training purposes, we use 10% of the data as our test set (8,168 training, 908 test).

Figure 4: Class Examples



From Figure 4, we can see that paintings are vastly different from the other classes, while photos and prints may be difficult to distinguish between. Photos and prints share a similar hue and tone, and drawings do not always have the vibrant colors found in paintings. Hence, our 4-way classification task should be difficult to learn, allowing ample room to experiment with different models.

5 Baseline Models

5.1 Gaussian Naive Bayes

Naive Bayes is a simple generative classifier based on applying Bayes' theorem to extract the conditional likelihood $p(y | \mathbf{x})$. The fundamental assumption for the features is "naive", assuming that every feature is conditionally independent given then class label y for K features. We can exploit the conditional independence assumption and that $p(\mathbf{x})$ is constant, hence our formulation for the joint distribution is:

$$p(y, \mathbf{x}) = p(y) \cdot \prod_{k=1}^K p(x_k | y) \quad (2)$$

Our task is to find the class label y that maximizes the posterior conditional likelihood:

$$\begin{aligned} \hat{y} &= \operatorname{argmax}_y p(y|\mathbf{x}) = \operatorname{argmax}_y \frac{p(y) \cdot \prod_{k=1}^K p(x_k | y)}{p(\mathbf{x})} \\ &= \operatorname{argmax}_y p(y) \cdot \prod_{k=1}^K p(x_k | y) \end{aligned} \quad (3)$$

Here, $p(y)$ is the frequency of class y in the training set. For our purposes we assume the features are normally distributed per class. This is beneficial for learning, as we learn class-wise means and variances, in contrast to the discrete case where each variable can take on 256 states. Hence, our conditional feature likelihood is:

$$p(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \quad (4)$$

To learn a Gaussian Naive Bayes model, we use Maximum Likelihood Estimation (MLE) for the parameters μ_{kc} and σ_{kc} per feature k and class c , and class weights θ_c . The MLE equations for N samples are:

$$\begin{aligned} \hat{\theta}_c &= \frac{1}{N} \sum_{i=1}^N \mathbb{1}\{y_i = y_c\} = \frac{N_c}{N} \\ \hat{\mu}_{kc} &= \frac{1}{N_c} \sum_{i=1}^N \mathbb{1}\{y_i = y_c\} \cdot x_{ik} \\ \hat{\sigma}_{kc} &= \frac{1}{N_c} \sum_{i=1}^N \mathbb{1}\{y_i = y_c\} \cdot (x_{ik} - \mu_{kc})^2 \end{aligned} \quad (5)$$

Here, $N_c = \sum_{i=1}^N \mathbb{1}\{y_i = y_c\}$, or the number of data points belonging to class c [5].

5.2 Multinomial Logistic Regression

Logistic Regression is the discriminative analog to the Naive Bayes classifier, sometimes referred to as the *maximum-entropy classifier* or a normalized *log-linear model*. As a discriminative model, we can optimize directly for the conditional likelihood $p(y | \mathbf{x})$, with $Z(\mathbf{x})$ as the normalizing partition function.

$$p(y | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{j=1}^K \theta_j f_j(y, \mathbf{x})\right\} \quad (6)$$

McCallum et al. formulates the Logistic Regression model, parameterized by weights $\theta \in \mathbb{R}^K$, in which a single set of weights is shared across all the classes [8]. This is achieved through a set of feature functions that are nonzero for a single class:

$$\begin{aligned} f_{y',j}(y, \mathbf{x}) &= \mathbb{1}\{y' = y\} \cdot x_j \\ f_{y'}(y, \mathbf{x}) &= \mathbb{1}\{y' = y\} \end{aligned} \quad (7)$$

This notation, while not standard, mirrors our later formulation of Conditional Random Fields in Section 6.2.

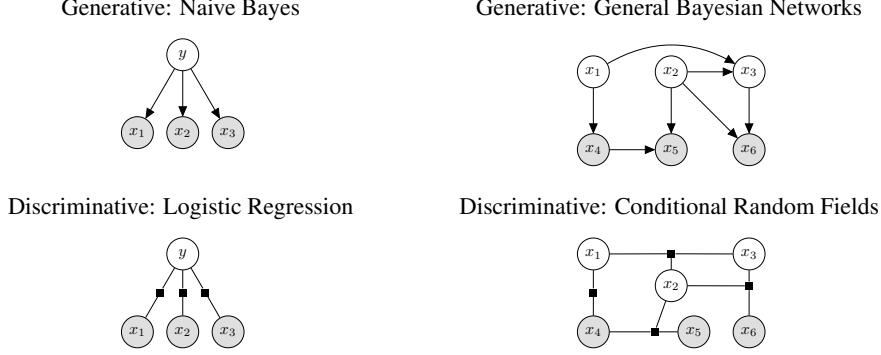
For parameter estimation, we opted to use scikit-learn's implementation of the L-BFGS algorithm, which approximates the Broyden-Fletcher-Goldfarb-Shanno algorithm [7]. This is a quasi-Newton optimization method that approximates the Hessian matrix to reduce memory usage. Scikit-learn optimizes the L2 penalized multinomial loss [7] [5]. Here, y_i is the true label and y_c could be any label:

$$\min_{\theta} \frac{1}{2} \cdot \theta^T \theta - \sum_{i=1}^N \left[\sum_{c=1}^C \left[\mathbb{1}\{y_i = y_c\} \cdot \sum_{k=1}^K \theta_k f_k(y_i, \mathbf{x}_i) \right] - \log Z(\mathbf{x}_i) \right] \quad (8)$$

6 General Models

6.1 Hybrid Bayesian Networks

General Bayesian Networks (GBNs) are directed graphical models that are comprised of a set of nodes and edges. The nodes are our random variables, and they interrelate through directed edges. This allows us to encode complex relations between our features \mathbf{x} that are not captured by the Naive



Bayes classifier. We can exploit these relationships to decompose the joint distribution into factors conditioned on the node's parents [5].

$$p(\mathbf{x}) = \prod_{k=1}^K p(x_k | pa(x_k)) \quad (9)$$

Since our labels are discrete and our features are continuous, we use Hybrid Bayesian Networks, otherwise known as *Conditional Linear Gaussian (CLG)* models if every discrete variable has only discrete parents and every continuous variable is represented as a CLG Conditional Probability Distribution (CPD) [2]. Continuous nodes make use of the multivariate gaussian distribution, parameterized by a mean vector μ and a symmetric square covariance matrix Σ .

$$p(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right\} \quad (10)$$

The conditional linear Gaussian CPD for a continuous variable x_j , discrete parents $\mathbf{U} = \{U_1, \dots, U_M\}$, continuous parents $\mathbf{C} = \{C_1, \dots, C_K\}$ is, for every value $\mathbf{u} \in Val(\mathbf{U})$ with $k+1$ coefficients $a_{\mathbf{u},0}, \dots, a_{\mathbf{u},k}$ and variance $\sigma_{\mathbf{u}}^2$:

$$p(x_j | \mathbf{u}, \mathbf{c}) = \mathcal{N} \left(a_{\mathbf{u},0} + \sum_{k=1}^K a_{\mathbf{u},k} \cdot c_k ; \sigma_{\mathbf{u}}^2 \right) \quad (11)$$

The log likelihood function ℓ for a continuous node X_i is:

$$\ell = \log \mathcal{L} = \sum_{i=1}^N \left\{ -\frac{1}{2} \log(2\pi\sigma_{\mathbf{u}}^2) - \frac{1}{2} \cdot \frac{1}{\sigma_{\mathbf{u}}^2} \left[a_{\mathbf{u},0} + \sum_{k=1}^K a_{\mathbf{u},k} \cdot c_k^i - x_j^i \right]^2 \right\} \quad (12)$$

The MLE parameters are obtained by ordinary linear regression of each node x_j on its continuous parents $c \in \mathbf{C}$, for each class value $\mathbf{u} \in \{0, 1, 2, 3\}$ of its discrete parent [2].

In contrast, our discrete node is the class label y , which is just the frequency. The maximum likelihood estimate is:

$$\hat{\theta}_{y_c} = \frac{\sum_{i=1}^N \mathbb{1}\{y^i = y_c\}}{\sum_{i=1}^N \sum_{y_k} \mathbb{1}\{y^i = y_k\}} = \frac{\sum_{i=1}^N \mathbb{1}\{y^i = y_c\}}{N} \quad (13)$$

Here, y_c is the class being estimated, y^i is the label for the i -th sample, and y_k are the other classes. N is the number of samples.

We elected to use a hybrid model as the discrete case led to *data fragmentation*, and not every state configuration can be seen in the training set.

Inference is performed by averaging likelihood weighting simulations using $n = 500$ random samples. For discrete values, the prediction is the highest conditional probability. The predicted value of a continuous value is the expected value of the conditional distribution.

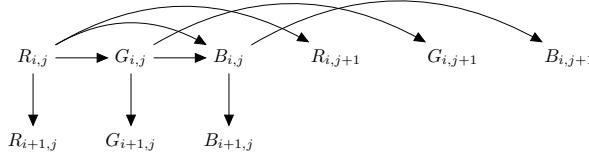


Figure 6: Hybrid Bayesian Network Model Architecture, for Pixel ij with color channels RGB

6.2 Conditional Random Fields (CRFs)

Conditional Random Fields (CRFs) are discriminative graphical models that can model output variables \mathbf{y} conditioned on features \mathbf{x} compactly, in contrast to generative models. CRFs are natural extensions to Logistic Regression for general graphs. Our formulation, parameter estimation, and inference approach comes from McCallum et al. [8] and Lafferty et al. [3].

CRFs model a conditional likelihood $p(\mathbf{y} \mid \mathbf{x})$ on a general graph \mathcal{G} , composed of a set of factors $F = \{\psi_a\}_{a=1}^A$. We reiterate McCallum's description of a CRF with parameter tying, where a set of factors can share a set of feature functions $\{f_{pk}(\mathbf{x}_c, \mathbf{y}_c)\}_{k=1}^{K(p)}$ and parameters $\theta_p \in \mathbb{R}^{K(p)}$ for a given *clique template* C_p . These clique templates come from partitioning the factors of graph \mathcal{G} into $\mathcal{C} = \{C_1, C_2, \dots, C_p\}$. Hence, our model is composed of the following:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \prod_{C_p \in \mathcal{C}} \prod_{\psi_c \in C_p} \psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p) \\ \psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p) &= \exp \left\{ \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(\mathbf{x}_c, \mathbf{y}_c) \right\} \\ Z(\mathbf{x}) &= \sum_{\mathbf{y}} \prod_{C_p \in \mathcal{C}} \prod_{\psi_c \in C_p} \psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p) \end{aligned} \quad (14)$$

Here, p is a factor index, f_{pk} is a feature function, and θ_{pk} is the weight associated with it. K is the number of features and $K(p)$ are the features found in factor p . Our labels are discrete and can be written as feature functions f_{pk} composed of observation functions q_{pk} to make *label-observation features*.

$$f_{pk}(\mathbf{y}_c, \mathbf{x}_c) = \mathbf{1}\{\mathbf{y}_c = \tilde{\mathbf{y}}_c\} \cdot q_{pk}(\mathbf{x}_c) \quad (15)$$

Parameter estimation for CRFs is done through MLE, where we can minimize the conditional negative log likelihood:

$$\ell(\theta) = - \sum_{C_p \in \mathcal{C}} \sum_{\psi_c \in C_p} \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(\mathbf{x}_c, \mathbf{y}_c) + \log Z(\mathbf{x}) \quad (16)$$

The partial derivative with respect to our parameter θ_{pk} associated with the clique template C_p is:

$$\frac{\partial \ell}{\partial \theta_{pk}} = - \sum_{\psi_c \in C_p} f_{pk}(\mathbf{x}_c, \mathbf{y}_c) + \sum_{\psi_c \in C_p} \sum_{\mathbf{y}'_c} f_{pk}(\mathbf{x}_c, \mathbf{y}'_c) \cdot p(\mathbf{y}'_c \mid \mathbf{x}) \quad (17)$$

Minimization of the cost function $\ell(\theta)$ is done through the L-BFGS optimization algorithm [6].

7 Experimental Task

To effectively study these objective measures, we conduct this experiment per model per downsample size. Our choices of sizes included: 1, 2, 4, 8, 16, 32, 64, and 128. Larger image sizes, such as 256×256 , quickly became impossible due to memory constraints. For the HBN, any test larger than 16×16 was intractable due to high prediction costs.

For a single downsample size, we performed a 3-fold cross validation on the full 9,076 image set, using 908 images randomly as test. This gives us three point estimates of accuracy along with confidence intervals. Variables measured for the cross-validation include: training accuracy, testing

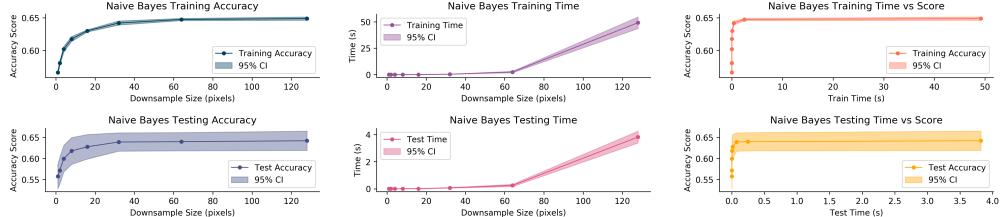


Figure 7: Generalization and Tractability Results for Gaussian Naive Bayes

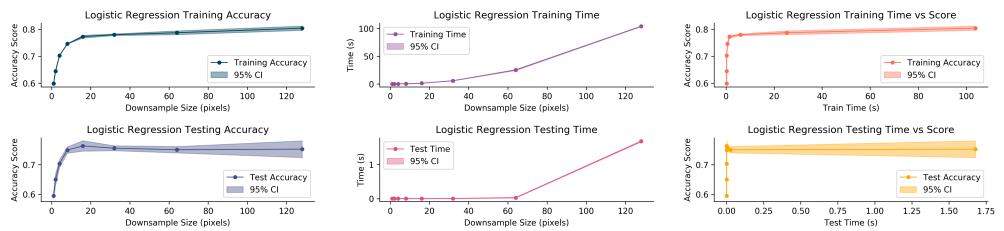


Figure 8: Generalization and Tractability Results for Multinomial Logistic Regression

accuracy, fit time, and score time. These measures provide two fronts of analysis: generalization and tractability.

All images were rescaled such that its pixel color channels ranged from $[0, 1]$, and we assume our data is continuous to avoid data fragmentation.

8 Results: Baseline Models

The baseline models, while naive in its assumptions, did reasonably well in terms of generalization ability and tractability. The Gaussian Naive Bayes model (Figure 7), performed similarly in the training and testing phase. However, compared to Logistic Regression (Figure 8), the Naive Bayes model takes longer to converge to a stationary accuracy as the number of features grows.

The time to fit the Naive Bayes was significantly smaller than the multinomial Logistic Regression. This effect stems from the lack of a quasi second-order optimization approach that Logistic Regression uses through L-BFGS. However, Logistic Regression was faster in inference, since Naive Bayes computes the gaussian likelihood for every feature x_{ijk} to make a prediction.

From Figures 7 and 8, no noticeable gains are made for either model for downsampled images larger than 16×16 , and in the case of Logistic Regression, can even overfit to noise. The best explanation is that image interpolation creates robust, strong features from raw data, reducing noise and helping generalization. However, it is important to note the significant loss in accuracy when the image is decimated to smaller sizes than 16×16 as information is lost. While the models do surprisingly well with a single color, these results show that image interpolation can effectively improve generalization and tractability of raw image data given a reasonable interpolation size.

9 Results: General Models

For the Hybrid Bayesian Network (Figure 9), initial generalization ability was improved from inter-feature connections. However, this formulation proved to be intractable after 32×32 . Hence, while we improve accuracy scores through complex interactions (Figure 6), this is offset by increased inference times. Parameter estimation proved to be effective, as each node learned a small subset of coefficients through linear regression on continuous parents and induced sparsity from our architecture. However, inference time is significantly larger than either the Conditional Random Field or the baselines.

The CRF (Figure 10) took longer to train than any other model, and yet had a comparable generalization curve to Logistic Regression, while outperforming the generative models. Logistic Regression performed better in training, yet had similar test scores. The increased train times result from a more

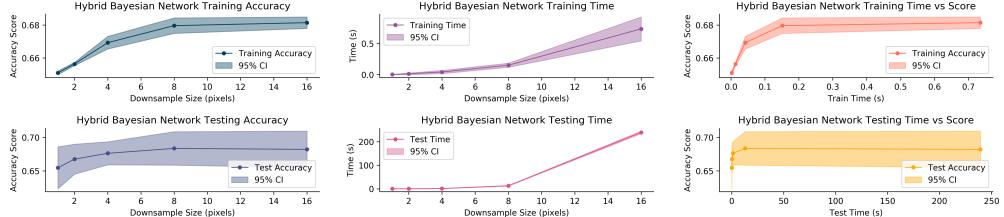


Figure 9: Generalization and Tractability Results for a Hybrid Bayesian Network

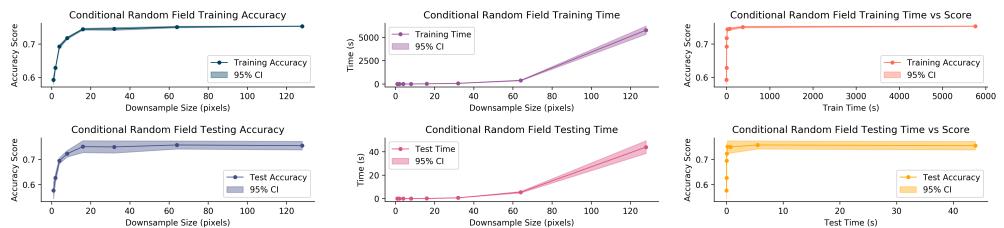


Figure 10: Generalization and Tractability Results for a Conditional Random Field

complicated optimization problem, using L-BFGS. In contrast, inference was faster than the Hybrid Bayesian Network.

10 Results: General Findings

Discriminative models learned better than their generative counterparts, as they model the conditional probability $p(y | x)$ directly. Hence, they are less concerned with modeling the features, and results in a lower parameter count. However, they are susceptible to over-fitting.

Image interpolation can effectively reduce the size of the data and models by creating robust features invariant to noise. Caution must be exerted if too much information is lost. Results stagnated or degraded as the feature count increased.

Adding more complex interactions did not help our learning task, and in some cases hindered our performance in generalization and tractability. This suggests that image interpolation encoded all relevant interactions between pixels in the original image that we do not need to account for with general graphs.

11 Conclusion

Our results show that image interpolation can be a useful technique for dimensionality reduction when using graphical models for classification. Our original images of 500×500 contained approximately 750,000 features, and interpolation can yield comparable results when reduced to a size of 16×16 (768 features). Image decimation is still prone to information loss and provides sub-optimal performance in heavily downsampled images. However, our results indicate that interpolation can be an attractive dimensionality reduction technique that preserves structure while improving generalization and tractability of graphical models.

All code for this project can be found here: <https://github.com/nextBillyonair/PGMProject>

12 Future Work

An extension of these experiments would be to examine the effect of color quantization, a technique used to reduce the number of colors in an image. This would compress the feature space for an image. Another extension would be to use structure learning algorithms on decimated images to explore what interactions are supported by the underlying distributions. Finally, examining how inter-class confusion in inference is influenced by interpolation techniques could yield interesting insights in what our models learn.

References

- [1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
- [2] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.
- [3] J. Lafferty. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289. Morgan Kaufmann, 2001.
- [4] T. E. J. Mensink and J. C. van Gemert. The rijksmuseum challenge: Museum-centered visual recognition. In *ACM International Conference on Multimedia Retrieval*, 2014.
- [5] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
- [6] S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Found. Trends. Comput. Graph. Vis.*, 6(3–4):185–365, Mar. 2011.
- [7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [8] C. Sutton and A. McCallum. An Introduction to Conditional Random Fields. *arXiv e-prints*, page arXiv:1011.4088, Nov 2010.
- [9] A. Y. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Adv. Neural Inf. Process. Sys*, 2, 04 2002.