Binary Operator Design by *k*-Nearest Neighbor Learning with Application to Image Resolution Increasing

Hae Yong Kim
Dept. Eng. Sistemas Eletrônicos, Universidade de São Paulo
Av. Prof. Luciano Gualberto, trav. 3, 158
CEP 05508-900, São Paulo, SP, Brazil
Email: hae@lps.usp.br

Abstract

In a typical office environment, heterogeneous devices and software, each working in a different spatial resolution, must interact together. Thus, there frequently arises the resolution conversion problem. The present paper addresses the spatial resolution increasing of binary images and documents (for example, conversion of a 300 dpi image into 600 dpi). A new, accurate and efficient solution to this problem is proposed. It makes use of the *k*-nearest neighbor learning to automatically design a windowed zoom operator starting from pairs of in-out sample images. The resulting operator is stored in a look-up-table, which is extremely fast computationally and therefore fit for real-time applications. Usually, it is useful to know a priori the sample complexity (the quantity of training samples needed to get, with probability 1-δ, an operator with accuracy ε). We use the Probably Approximately Correct (PAC) learning theory to compute it, for both noise-free and noisy cases. Since the PAC theory usually yields an overestimated sample complexity, the statistical estimation is used to estimate, a posteriori, a tight error bound. The statistical estimation is also used to show that the *k*-nearest neighbor learning has a good inductive bias that allows reducing the quantity of training sample images needed.

I. Introduction

In a typical office environment, digital images and documents are manipulated by a set of non-homogeneous interacting devices and software that form a system able to scan, edit, display, print, transmit, perform OCR, and execute several forms of image processing tasks. As each sys-

tem component may operate at a different spatial resolution, there frequently arises the necessity of resolution conversion, in order to allow digital images and documents to migrate from one system component to another.

Spatial resolution decreasing is a relatively easy task. In contrast, spatial resolution increasing (or zooming) is difficult, since the input-image usually does not contain all the necessary information to generate a perfectly zoomed output-image. Moreover, the ideal zooming depends on the "context" of the application. For example, the optimal operator designed to zoom twice the "Times, 12 pt., 300 dpi" characters may not be optimal for another font or a handwritten document.

Many different image zooming algorithms have been developed for gray-scale and color images. However, it seems that binary image zooming has so far received much less attention. Loce et al. (1997a,b) present some of a small number of published binary zooming techniques. This is somewhat puzzling, since binary zooming is indeed necessary in practice. For example, consider how often digital documents at 300 dpi have to be printed in a 600 dpi printer.

Many image processing tasks are based on the windowed operator (W-operator). A W-operator is an image transformation where the color of an output-pixel is a function of the colors of its neighboring pixels in the input-image. Some works propose to use the machine learning framework to design automatically a W-operator starting from in-out training sample images (Dougherty, 1992; Kim et al., 1997, 1998, 1999). Especially, Kim and Cipparrone (1998) have proposed to use the *k*-nearest neighbor (*k*-NN) learning to design W-operator. We use the same idea here to design the windowed zoom operator (WZ-operator). Some of the ideas developed here were published in a conference paper (Kim and Barreto, 2000).

Loce et al. (1997a,b) in essence expound two techniques to zoom binary images. They use non-increasing and increasing filters (we are using words "filter" and "operator" as synonyms). The present paper proposes a number of improvements upon them:

First, as the true probability distribution governing the zooming process is usually unknown, in practice the statistics derived from in-out sample images may be used instead. Conse-

quently, the best operator one can get is the operator that is optimal over the training images (disregarding the inductive bias, to be explained later). We call this the empirically optimal (e-optimal) operator. The increasing filter technique (Loce et al., 1997a, chap. 9; 1997b) may yield an empirically sub-optimal solution while the non-increasing filter approach (Loce and Dougherty, 1997a, chap. 6) and ours always design an e-optimal operator. Besides, previous works do not analyze the dissimilarity between the empirically and the truly optimal operators, implicitly assuming that the statistics derived from sample images are a close approximation of the true probability distribution. We propose statistical techniques to estimate the difference between the two error rates. Furthermore, previous techniques do not adopt any explicit inductive bias. The inductive bias is the set of prior assumptions by which the learner generalizes beyond the observed training data, to infer the classification of new instances. A learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances. We have adopted the *k*-NN learning because it has a solid inductive bias, exhaustively tested in many different applications. We show experimentally its effectiveness for the problem addressed.

Second, the previous works seem to demand human intervention in the operator design process. Our technique is, on the contrary, fully automatic.

Third, the increasing filter technique is an attempt to improve the non-increasing one, aiming at hardware implementation: it is focused on designing a "logically efficient" operator, i.e., an operator represented using a small number of logic gates. Our approach focuses on software implementation, where logic reduction loses its attractiveness because simpler logic does not necessarily mean a faster technique. Instead, the use of appropriate data structures and algorithms may yield faster methods, by reducing the computational complexity. In order to speed up W-operator evaluating, Jones and Svalbe (1994) use look-up-table (LUT), Kim et al. (1997; 1998; 1999) use a tree-shaped data structure, and Robert and Malandain (1998) use binary decision diagram. The LUT is extremely fast in the evaluating stage and allows the implementation of the exact *k*-NN learning, but its demand for memory and training time increases exponentially as the window size grows. The tree-structure requires only a moderate quantity of memory and training time, but its evaluating time is far longer than LUT (when it implements the exact *k*-NN learning using a data

structure known as kd-tree, which requires a back-tracking process) or slightly longer than LUT (when it implements the decision tree learning, a machine learning strategy closely related to *k*-NN learning). The binary decision diagram is as fast as tree-structure in evaluation and uses less memory, but its training process is very slow. We have adopted the LUT solution, because experimental results have shown that zooming printed and handwritten documents does not demand large windows.

In this paper, we were compelled to use notations and terminologies derived from two different branches of knowledge: image processing and machine learning. This forced us to use, in many occasions, two different notations or two different terms for a single object.

The remainder of this paper is divided as follows:

In the next section, we present the machine learning theory applied to the W-operator design. This section is subdivided into 6 subsections. In subsection II.A, we formalize the binary W-operator learning problem. In subsection II.B, we explain how to estimate the sample complexity for noise-free problems. Subsection II.C extends this theory to noisy cases. Subsection II.D elucidates how to estimate the difference between true-error rates of learned and optimal W-operators. Subsection II.E expounds the *k*-NN learning. Subsection II.F explains how the two learning algorithms can be compared, to be used later to show the effectiveness of the *k*-NN inductive bias.

In section III, previously developed theory is applied to binary image zooming by *k*-NN learning. This section is subdivided into 4 subsections. Subsection III.A defines the WZ-operator and explains the process of WZ-operator design by *k*-NN learning. Subsection III.B describes experimental results obtained in the printed document resolution increasing and shows that the designed WZ-operator is virtually optimal. Subsection III.C shows the effectiveness of *k*-NN inductive bias by analyzing the empirical error rates of different learning algorithms. Subsection III.D is devoted to resolution increasing of handwritten documents.

Finally, in section IV we present our conclusions.

II. Binary W-Operator Learning

A. The Problem

Let us start our exposition by stating some results of machine learning theory and by applying them to design binary windowed operators (W-operators), because certainly the image processing community is unfamiliar with these results. We will simply state these results, without explaining or demonstrating them. The reader is referred to works (Mitchell, 1997; Anthony and Biggs, 1992; Haussler, 1992) for further informations.

We will make use of the PAC (Probably Approximately Correct) learning theory to bring forth the sample complexity. The sample complexity is the quantity of training samples needed to get, with probability at least 1-δ, a W-operator with the expected error at most ε. Unfortunately, often only an overestimated sample complexity can be obtained using the PAC theory. Even though, it will be useful as an upper bound of the needed amount of samples, and to show the convergence of the learning process. Besides, the PAC-learning theory will permit us to express rigorously the W-operator learning problem, and may sharpen considerably one's insight into the problem.

Let us define a binary image as a function $Q{:}\mathbb{Z}^2 \to \{0,1\}$. The support of a binary image Q is a finite subset of $\mathbb{Z}^2$ where the image is actually defined. The size of support is the number of pixels of the image and an image is considered to be filled with a background color out of its support.

A binary W-operator Ψ is a function that maps a binary image into another, defined via a set of w points called window

$$\vec{W} = \{W_1, \ldots, W_w\},\ W_i \in \mathbb{Z}^2$$

and a concept or characteristic function

$$\psi{:}\ \{0,1\}^w \to \{0,1\}$$

as follows:

$$\Psi(Q)(p) = \psi(Q(W_1 + p), \ldots, Q(W_w + p)),$$

where $p \in \mathbb{Z}^2$. A point W_i of the window is called a peephole.

Let the images $A^{\mathbf{x}}$, $A^{\mathbf{y}}$, $Q^{\mathbf{x}}$ and $Q^{\mathbf{y}}$ be respectively the training input-image, the training output-image, the image to-be-processed and the (supposedly unknown) ideal output-image. We can suppose that there is only one pair of training images ($A^{\mathbf{x}}$ and $A^{\mathbf{y}}$), because if there are many pairs, they can be "glued" together to form a single pair. In order to design the W-operator $\hat{\Psi}$, the user must choose an appropriate window $\vec{W}$ by hand.

Let us denote the content in $A^{\mathbf{x}}$ of the window $\vec{W}$ shifted to $p \in \mathbb{Z}^2$ as $a_p^{\mathbf{x}}$ and call it the training instance or the sample input-pattern at pixel *p*:

$$a_p^{\mathbf{x}} = [A^{\mathbf{x}}(W_1 + p), A^{\mathbf{x}}(W_2 + p), \ldots, A^{\mathbf{x}}(W_w + p)] \quad \in \{0,1\}^w .$$

Each pattern $a_p^{\mathbf{x}}$ is associated with an output-color or classification $A^{\mathbf{y}}(p) \in \{0,1\}$. Let us denote the data obtained when all pixels of $A^{\mathbf{x}}$ and $A^{\mathbf{y}}$ are scanned by

$$\vec{a} = \left((a_{p_1}^{\mathbf{x}}, A^{\mathbf{y}}(p_1)), \ldots, (a_{p_m}^{\mathbf{x}}, A^{\mathbf{y}}(p_m))\right)$$

and call it the sample sequence (*m* is the amount of pixels of images $A^{\mathbf{x}}$ and $A^{\mathbf{y}}$). Each element $(a_{p_i}^{\mathbf{x}}, A^{\mathbf{y}}(p_i)) \in \vec{a}$ is called an example or a sample. Let us similarly construct the sequence

$$\vec{q} = \left((q_{p_1}^{\mathbf{x}}, Q^{\mathbf{y}}(p_1)), \ldots, (q_{p_n}^{\mathbf{x}}, Q^{\mathbf{y}}(p_n))\right)$$

from $Q^{\mathbf{x}}$ and $Q^{\mathbf{y}}$ (*n* is the quantity of pixels of $Q^{\mathbf{x}}$ and $Q^{\mathbf{y}}$). Each $q_{p_i}^{\mathbf{x}}$ is called a query-pattern or an instance to-be-processed and the output $Q^{\mathbf{y}}(p_i) \in \{0,1\}$ is called the ideal output-color or the ideal classification.

The learner or learning algorithm **A** is called upon to construct a W-operator $\hat{\Psi}$ based on $A^{\mathbf{x}}$ and $A^{\mathbf{y}}$ such that, when $\hat{\Psi}$ is applied to $Q^{\mathbf{x}}$, the resulting processed-image $\hat{Q}^{\mathbf{y}} = \hat{\Psi}(Q^{\mathbf{x}})$ is expected to be similar to the ideal output-image $Q^{\mathbf{y}}$. More precisely, the learner **A** should construct a characteristic function or hypothesis $\hat{\psi}$ based on the sample sequence $\vec{a}$ such that, when $\hat{\psi}$ is applied to a query-pattern $q_{p_i}^{\mathbf{x}}$, its classification $\hat{Q}^{\mathbf{y}}(p_i) = \hat{\psi}(q_{p_i}^{\mathbf{x}})$ is expected to be the same as $Q^{\mathbf{y}}(p_i)$ with high probability. The function $\hat{\psi}$ and the window $\vec{W}$ together represent the W-operator $\hat{\Psi}$.

B. Noise-Free Case

Let us distinguish noise-free from noisy cases. Our target problem (binary resolution increasing) is noisy and so are almost all practical problems. However, the study of noise-free problems will help us to understand better the noisy cases.

In a noise-free environment, there exists a clearly defined target concept $\psi: \{0,1\}^w \rightarrow \{0,1\}$ that the learner is called upon to learn. In such an environment, we can suppose that the training instances $a^{\mathbf{x}}_{p_i}$ are drawn randomly and independently from the space $\{0,1\}^w$ by a probability distribution *P*. Furthermore, the output colors $A^{\mathbf{y}}(p_i)$ are generated by applying the target function ψ to $a^{\mathbf{x}}_{p_i}$, that is, $A^{\mathbf{y}}(p_i) = \psi(a^{\mathbf{x}}_{p_i})$ for all pairs $(a^{\mathbf{x}}_{p_i}, A^{\mathbf{y}}(p_i)) \in \vec{a}$.

The learner **A** considers some set $H \subset \left(\{0,1\}^w \rightarrow \{0,1\}\right)$ of possible hypothesis when attempting to learn the target concept ψ. If no information on ψ is available, the learner should assume $H = \left(\{0,1\}^w \rightarrow \{0,1\}\right)$. However, some a priori information can highly simplify the learning process, because it can greatly reduce the cardinality of the hypotheses space *H*. For example, emulating an erosion Ψ with the information that Ψ is an erosion is much easier than emulating Ψ when no prior information is available (examples 2 and 3). In the W-operator training stage, the learner **A** receives a sample sequence $\vec{a}$ and searches for a hypothesis $\hat{\psi} = \mathbf{A}(\vec{a})$ in the space *H*.

Let us define the true-error of the hypothesis $\hat{\psi}$ as the probability that $\hat{\psi}$ will misclassify an instance $q^{\mathbf{x}}_{p_i}$ drawn at random according to *P*:

$$\text{error}_P(\hat{\psi}) = P\left\{q^{\mathbf{x}}_{p_i} \in \{0,1\}^w \;\middle|\; \psi(q^{\mathbf{x}}_{p_i}) \neq \hat{\psi}(q^{\mathbf{x}}_{p_i})\right\}$$

According to the PAC theory (Mitchell, 1997; Anthony and Biggs, 1992), any consistent learner using a finite hypotheses space *H* with target function $\psi \in H$ will, with probability at least (1-δ), output a hypothesis $\hat{\psi}$ with error at most ε, after observing *m* examples drawn randomly by *P*, as long as

$$m \geq \frac{1}{\varepsilon}\left[\ln\left(\frac{1}{\delta}\right) + \ln\left(|H|\right)\right]. \qquad (1)$$

A learner is consistent if it outputs hypotheses that perfectly fit the training data, whenever possible. The bound (1) is often a substantial overestimate, mainly because no additional assumptions were made on the learner but the consistency. Some examples of use of equation (1) follow.

Example 1: In figure 1, a fingerprint image $A^{\mathbf{x}}$ (1a) has been processed by a 3×3 W-operator Ψ, yielding image $A^{\mathbf{y}}$ (1b). This operator consisted of the union of 8 hit-or-miss operators. The hit-or-miss is one of the elementary operators of mathematical morphology and its definition can be found, for example, in (Gonzalez and Woods, 1992). Let us suppose that somehow we were told that Ψ is defined in a 3×3 window. Using this information and images $A^{\mathbf{x}}$ and $A^{\mathbf{y}}$, a W-operator $\hat{\Psi}$ has been constructed by a consistent learner. According to (1), with probability at least 99%, the true-error of $\hat{\Psi}$ will be at most 1%, as long as the training images have an amount of pixels

$$m \geq \frac{1}{0.01}\left[\ln\left(\frac{1}{0.01}\right) + \ln\left(2^{2^9}\right)\right] \cong 35950\,.$$

Since images $A^{\mathbf{x}}$ and $A^{\mathbf{y}}$ have 200×200=40000 pixels, almost certainly $\hat{\Psi}$ will present an error rate at most 1%. Not surprisingly, when $\hat{\Psi}$ was applied to another fingerprint image (figure 1c), an image $\hat{Q}^{\mathbf{y}} = \hat{\Psi}(Q^{\mathbf{x}})$ (figure 1d) exactly equal to the ideal output $Q^{\mathbf{y}} = \Psi(Q^{\mathbf{x}})$ was produced, that is, $\hat{\Psi}$ presented zero error. This test has been repeated a number of times and the error rates have always been zero.

Note that the analysis above holds only if images $A^{\mathbf{x}}$ and $Q^{\mathbf{x}}$ are supposed to have been generated by the same probability distribution. That is, $A^{\mathbf{x}}$ and $Q^{\mathbf{x}}$ must be of the same type: fingerprint images, handwritten images, printed documents, and so on.

Example 2: Let us solve again the example 1, this time supposing that the operator is defined within a 7×7 window. In this case:

$$m \geq \frac{1}{0.01}\left[\ln\left(\frac{1}{0.01}\right) + \ln\left(2^{2^{49}}\right)\right] \cong 3.9 \times 10^{16}\,.$$

That is, sample images must be larger than $(2\times10^8)\times(2\times10^8)$! Clearly, an image this large cannot be obtained in practice.

Example 3: Let us solve again the example 2, now supposing that we were told that the target operator is an erosion whose structuring element fits within a 7×7 window. An erosion is an elementary operator of mathematical morphology and its definition can be found, for example, in (Gonzalez and Woods, 1992). As each one of 49 peepholes may either belong or not to the structuring element, the target operator must be one of 2^{49} erosions. Thus, $|H| = 2^{49}$ and:

$$m \geq \frac{1}{0.01}\left[\ln\left(\frac{1}{0.01}\right) + \ln\left(2^{49}\right)\right] \cong 3857 .$$

That is, any pair of training images larger than 63×63 will be enough, to be compared with $(2\times10^8)\times(2\times10^8)$ images needed in example 2.

The simplification above holds only using a learner specifically designed for erosions. Similar results can be derived for other elementary operators such as dilation, hit-or-miss, union of *k* erosions, and so on.

C. Noisy Case

In order to model the noisy case, let us suppose that each example $(a_p^{\mathbf{x}}, A^{\mathbf{y}}(p)) \in \vec{a}$ has been generated independently by an unknown joint probability distribution *P* in the space $\{0,1\}^w \times \{0,1\}$. Let us also suppose that each element $(q_{p_i}^{\mathbf{x}}, Q^{\mathbf{y}}(p_i)) \in \vec{q}$ has been generated independently by the same distribution *P*.

The true-error of a hypothesis ψ now should be defined as the probability that ψ will misclassify an example $(q_{p_i}^{\mathbf{x}}, Q^{\mathbf{y}}(p_i))$ drawn at random according to *P*:

$$\text{error}_P(\psi) = P\left\{(q_{p_i}^{\mathbf{x}}, Q^{\mathbf{y}}(p_i)) \in \{0,1\}^w \times \{0,1\} \;\middle|\; \psi(q_{p_i}^{\mathbf{x}}) \neq Q^{\mathbf{y}}(p_i)\right\}$$

In noisy situation, there is no clearly defined target function. In its stead, there is an optimal function ψ* with minimal true-error. Let the empirical error (e-error) of a hypothesis ψ over a sequence $\vec{a}$ be defined as the proportion of errors committed when ψ classifies instances in $\vec{a}$:

$$\text{error}_{\vec{a}}(\psi) = \frac{1}{m}\left|\left\{(a_{p_i}^{\mathbf{x}}, A^{\mathbf{y}}(p_i)) \in \vec{a} \;\middle|\; \psi(a_{p_i}^{\mathbf{x}}) \neq A^{\mathbf{y}}(p_i)\right\}\right|,$$

where *m* is the length of $\vec{a}$.

Let $\hat{\psi}$ be the hypothesis with minimal e-error over $\vec{a}$ and let ψ^* be the hypothesis with minimal true-error. Then (Haussler, 1992)

$$\Pr\left[\text{error}_P(\hat{\psi}) - \text{error}_P(\psi^*) > \varepsilon\right] < \delta,$$

as long as H is finite and the length m of $\vec{a}$ satisfies:

$$m \geq \frac{1}{2\varepsilon^2}\left[\ln\left(\frac{1}{\delta}\right) + \ln\left(2|H|\right)\right]. \qquad (2)$$

Unfortunately, the sample complexity yielded by this equation is even more overestimated than that of (1). Given a sample sequence $\vec{a}$, the empirically optimal (e-optimal) hypothesis $\hat{\psi}$ can be easily constructed. Let us define that a learner **A** is e-optimal if it always generates hypothesis e-optimal over the training sequence. If **A** were e-optimal, given a query-pattern $q^{\mathbf{x}}_{p_i}$, what should be its classification $\hat{\psi}(q^{\mathbf{x}}_{p_i}) = \mathbf{A}(\vec{a})(q^{\mathbf{x}}_{p_i})$? Let $(a^{\mathbf{x}}_{r_1}, A^{\mathbf{y}}(r_1))$, ..., $(a^{\mathbf{x}}_{r_N}, A^{\mathbf{y}}(r_N))$ be N training examples of $q^{\mathbf{x}}_{p_i}$, that is, $a^{\mathbf{x}}_{r_j} = q^{\mathbf{x}}_{p_i}$, $1 \leq j \leq N$ (suppose that these are the only examples of $q^{\mathbf{x}}_{p_i}$ in $\vec{a}$). As there are noises, the N examples above may disagree on the classification of $q^{\mathbf{x}}_{p_i}$. To minimize e-error, the classification should be decided by the majority of votes of these training samples:

$$\hat{\psi}(q^{\mathbf{x}}_{p_i}) \leftarrow \text{mode}(A^{\mathbf{y}}(r_1), \ldots, A^{\mathbf{y}}(r_N)).$$

Note that every e-optimal learner is consistent in noise-free environment.

A good practical noisy problem is the binary resolution increasing, to be analyzed in section III. We present below a simpler example.

Example 4: The fingerprint images 1a and 1c have been corrupted by "salt-and-pepper" noise, resulting in images depicted in 2a and 2b. On average, 1 in every 40 pixels have changed its color. We want to design a 3×3 W-operator $\hat{\Psi}$ such that an image similar to the noiseless ideal-output $A^{\mathbf{y}}$ (figure 1b) results when the noisy image 2a is processed by $\hat{\Psi}$. In order to achieve this goal, a W-operator $\hat{\Psi}$ has been designed by an e-optimal learner using images 2a and 1b as the in-out training samples. Since images 2a and 1b have 200×200 pixels, with probability at least

99%, the difference between the true-error of optimal 3×3 operator Ψ^* and the true-error of $\hat{\Psi}$ will not exceed 6.71%, that is, $\text{error}_P(\hat{\psi}) - \text{error}_P(\psi^*) \leq 0.0671$, because:

$$\frac{1}{2 \times 0.0671^2}\left[\ln\left(\frac{1}{0.01}\right) + \ln\left(2 \times 2^{2^9}\right)\right] \cong 40000\,.$$

Clearly, the theory above only holds if the learner is e-optimal. The *k*-NN learning, though not originally e-optimal, can become e-optimal by modifying slightly its original rule.

D. Statistical Estimation of Error Rate

This subsection will expound techniques to compute a tighter bound of the error rate. These techniques will be very useful, since equations (1) and (2) often yield an overestimated error rate and sample complexity. Contrarily to prior formulae, the techniques of this subsection can only be applied after designing the W-operator, with the additional condition that the ideal output-image $Q^{\mathbf{y}}$ is available. It is licit to suppose that the ideal output will be available to perform tests, because we are supposing that a pair of in-out sample images is available to design the W-operator. And, if sample images are available, they can be broken in two pieces: training images ($A^{\mathbf{x}}$, $A^{\mathbf{y}}$) and test images ($Q^{\mathbf{x}}$, $Q^{\mathbf{y}}$).

So, supposing that the ideal output $Q^{\mathbf{y}}$ is available, a simple reckoning of non-matching pixels between $Q^{\mathbf{y}}$ and $\hat{Q}^{\mathbf{y}}$ provides the e-error rate. And, given the observed accuracy of a hypothesis over a limited sample of data, it is possible to know how well this estimates the accuracy over additional examples. To this purpose, let us construct one or two-sided confidence intervals. Further explanations on confidence intervals of mean of binomial random variables can be found in (Mitchell, 1997, chap. 5) or in many elementary books on Statistics. With *N*% confidence:

$$\text{error}_P(\hat{\psi}) \in \text{error}_{\vec{q}}(\hat{\psi}) \pm z_N \sqrt{\frac{\text{error}_{\vec{q}}(\hat{\psi})(1 - \text{error}_{\vec{q}}(\hat{\psi}))}{n}}\,, \tag{3}$$

$$\text{error}_P(\hat{\psi}) \leq \text{error}_{\vec{q}}(\hat{\psi}) + z'_N \sqrt{\frac{\text{error}_{\vec{q}}(\hat{\psi})(1 - \text{error}_{\vec{q}}(\hat{\psi}))}{n}}\,, \tag{4}$$

where *n* is the length of $\vec{q}$; z_N defines the half width of the smallest interval about the mean that includes *N*% of the total probability mass under the normal distribution with standard deviation 1;

and $z'_N \equiv z_{2N-1}$. For example, $z'_{84\%} = z_{68\%} = 1.00$, $z'_{95\%} = z_{90\%} = 1.64$, $z'_{99\%} = z_{98\%} = 2.33$ and $z_{99\%} = 2.58$. Formulae (3) and (4) ordinarily yield a much more accurate estimate of the error rate than (1) and (2).

In noise-free cases, it suffices to know a tight upper bound of the designed operator's true-error rate ($\text{error}_P(\hat{\psi})$). But, in noisy cases, the minimal error ($\text{error}_P(\psi^*)$) also must be estimated since, as the designed operator will never get a true-error smaller than the minimum, an operator can be considered a good solution if its true-error is close to the minimum. Unfortunately, we have no means to estimate $\text{error}_P(\psi^*)$ directly, because the optimal operator is unknown. We describe below an artifice that has managed to establish nice lower bounds of $\text{error}_P(\psi^*)$. Though very simple, we have never seen it written elsewhere.

Let us first construct the hypothesis $\hat{\psi}^*$, e-optimal over $\vec{q}$. If the learner **A** is e-optimal, $\hat{\psi}^* = \mathbf{A}(\vec{q})$. Note that we are training the operator with the very images ($Q^{\mathbf{x}}$, $Q^{\mathbf{y}}$) that will be used in the test. Clearly, $\text{error}_{\vec{q}}(\hat{\psi}^*) \le \text{error}_{\vec{q}}(\psi^*)$ and $\text{error}_{\vec{q}}(\hat{\psi}^*)$ can be measured experimentally. Then, we can use the following inequality to estimate the lower bound of $\text{error}_P(\psi^*)$:

$$\begin{aligned} \text{error}_P(\psi^*) &\ge \text{error}_{\vec{q}}(\psi^*) - z'_N \sqrt{\frac{\text{error}_{\vec{q}}(\psi^*)(1-\text{error}_{\vec{q}}(\psi^*))}{n}} \\ &\ge \text{error}_{\vec{q}}(\hat{\psi}^*) - z'_N \sqrt{\frac{\text{error}_{\vec{q}}(\hat{\psi}^*)(1-\text{error}_{\vec{q}}(\hat{\psi}^*))}{n}} \end{aligned} \tag{5}$$

This inequality holds, with confidence level N%, whenever:

$$\frac{b+1-\sqrt{b(b+1)}}{2(b+1)} \le \text{error}_{\vec{q}}(\hat{\psi}^*) \le \text{error}_{\vec{q}}(\psi^*) \le \frac{b+1+\sqrt{b(b+1)}}{2(b+1)} \tag{6}$$

where $b = n/(z'_N)^2$. Note that the inequality (6) is true for almost any practical problems and consequently so is (5).

Example 5: In example 4, we have concluded with 99% of confidence that the designed operator $\hat{\Psi}$ committes at most 6.71% more errors than the optimal 3×3 operator. On the other hand, in order to establish a tighter error bound, the e-error of $\hat{\Psi}$ (i.e., the difference between images 1d and 2c) was measured and found to be 4.992%. Using (3), we conclude with 99% of confidence

that the true-error of $\hat{\Psi}$ is in the interval (4.992±0.281)%. The e-optimal operator $\hat{\Psi}^*$ over the test images (figures 2b and 1d) has yielded e-error 4.723% when processing image 2b. Using equation (5), we conclude with 99% of confidence that the true-error of the optimal 3×3 operator is at least (4.723-0.247)%. Consequently, with confidence higher than 99%, the true-error of $\hat{\Psi}$ is at most 0.797% higher than the optimal operator's true-error, that is:

$$\text{error}_P(\hat{\psi}) - \text{error}_P(\psi^*) \leq 0.00797 .$$

This result confirms that equation (2) yields an overestimated error rate, for 0.797% is much smaller than 6.71%.

E. k-Nearest Neighbor Learning

In subsections II.B and II.C, we supposed that the learner is e-optimal (or consistent) in order to compute the sample complexity. But the e-optimality alone does not fully specify a learning algorithm, because there are many different e-optimal learners. To completely specify a learner, an inductive bias must be chosen as well. We chose the *k*-nearest neighbor (*k*-NN) learning because its inductive bias is one of most widely used in practice and experimental tests have confirmed its effectiveness for the problem addressed.

The *k*-NN learning is a conceptually straightforward approach to approximating target functions. Its inductive bias corresponds to the assumption that the classification of an instance will be most similar to the classification of other instances that are nearby in distance. This is especially true for W-operator and WZ-operator design, because it is natural and intuitive assigning visually similar patterns to the same class.

In the naive *k*-NN learning algorithm, the training consists of simply storing the presented training data. According to the *k*-NN rule, for each query-pattern $q_{p_i}^{\mathbf{x}}$, the *k* most similar training input-patterns must be sought in $\vec{a}$. As we are dealing with binary images, the distance between $q_{p_i}^{\mathbf{x}}$ and the training input-patterns may be measured using the Hamming distance (i.e., the number of non-matching bits) or the weighted Hamming distance. In the later case, some peepholes (e.g., central peepholes) can be given more importance than others (e.g., peripheral peepholes). Figure 4 depicts one weightless and one weighted windows. The output is defined as the most

common clasification among the *k* nearest training examples. Clearly, this original *k*-NN rule is not e-optimal. But changing it slightly as follows, it becomes e-optimal:

1) If the query-pattern $q^{\mathbf{x}}_{p_i}$ appears once or more times in $\vec{a}$, its classification will be the majority of votes of only these training instances. That is, let $a^{\mathbf{x}}_{r_1}$, ..., $a^{\mathbf{x}}_{r_N}$ be training instances such that $a^{\mathbf{x}}_{r_j} = q^{\mathbf{x}}_{p_i}, 1 \le j \le N$. Then, let $\hat{q}^{\mathbf{y}}_{p_i} \leftarrow \underset{1 \le j \le N}{\text{mode}}\left[A^{\mathbf{y}}(r_j)\right]$. In this case, *N* may be greater, equal or smaller than *k*.

2) On the other hand, if the query-pattern $q^{\mathbf{x}}_{p_i}$ was never seen before, search for its *k* most similar instances in $\vec{a}$ and take the majority of votes of them. That is, let $a^{\mathbf{x}}_{r_1}$, ..., $a^{\mathbf{x}}_{r_N}$ be the *N* most similar instances of $q^{\mathbf{x}}_{p_i}$, according to some distance measure. Then, again let $\hat{q}^{\mathbf{y}}_{p_i} \leftarrow \underset{1 \le j \le N}{\text{mode}}\left[A^{\mathbf{y}}(r_j)\right]$. In this case, *N* may be equal or greater than *k* (if there are ties), but it can never be smaller than *k*.

We call this modified rule the empirically optimal *k*-NN (ek-NN) learning.

F. Comparing Different Inductive Biases

Often we are interested in comparing the performance of two learning algorithms $\mathbf{A}_1$ and $\mathbf{A}_2$ rather than two specific hypotheses. For example, we may want to determine whether ek-NN inductive bias is more effective than others. In other words, we wish to estimate the expected difference of the true-error rates:

$$E\left[error_P(\mathbf{A}_1(\vec{a})) - error_P(\mathbf{A}_2(\vec{a}))\right] = \sum_{\vec{a} \in (\{0,1\}^w \times \{0,1\})^m} \left[error_P(\mathbf{A}_1(\vec{a})) - error_P(\mathbf{A}_2(\vec{a}))\right] P^m(\vec{a}).$$

To establish a confidence interval for the quantity above, the two learners $\mathbf{A}_1$ and $\mathbf{A}_2$ must be trained using *K* independent training sequences $\vec{a}_i, 1 \le i \le K$, and applied to *K* different test sequences $\vec{q}_i, 1 \le i \le K$. This process will yield *K* differences between e-errors of $\mathbf{A}_1$ and $\mathbf{A}_2$:

$$\delta_i = error_{\vec{q}_i}(\mathbf{A}_1(\vec{a}_i)) - error_{\vec{q}_i}(\mathbf{A}_2(\vec{a}_i)),\ 1 \le i \le K.$$

One or two-sided confidence intervals can be built from δ_1, ..., δ_K using the *t*-distribution. With confidence *N*%:

$$\underset{\vec{a}\in P^m}{E}\left[error_P(\mathbf{A}_1(\vec{a}))-error_P(\mathbf{A}_2(\vec{a}))\right]\in \bar{\delta}\pm t_{N,k-1}s_{\bar{\delta}} \tag{7}$$

$$\underset{\vec{a}\in P^m}{E}\left[error_P(\mathbf{A}_1(\vec{a}))-error_P(\mathbf{A}_2(\vec{a}))\right]> \bar{\delta}- t'_{N,k-1}s_{\bar{\delta}} \tag{8}$$

where:

- $s_{\bar{\delta}}\equiv\sqrt{\frac{1}{K(K-1)}\sum_{i=1}^{K}(\delta_i-\bar{\delta})^2}$;
- $\bar{\delta}\equiv\left(\delta_1+\cdots+\delta_K\right)/K$;
- $t_{N,K-1}$ defines the half width of the smallest interval about the mean that includes *N*% of the total probability mass under the normalized *t*-distribution with (*K*-1) degrees of freedom; and $t'_{N,K-1}\equiv t_{2N-1,K-1}$.

For example, $t'_{95\%,2}=t_{90\%,2}=2.92$, $t'_{97.5\%,2}=t_{95\%,2}=4.30$ and $t'_{99\%,2}=t_{98\%,2}=6.96$.

III. Resolution Increasing by *k*-NN Learning

A. Windowed Zoom Operator Design by k-NN Learning

In this subsection, we will use the theory developed so far to increase the resolution of printed or handwritten documents.

Let us first define the windowed zoom operator (WZ-operator). A WZ-operator Ψ is defined via a window $\vec{W}$ and f^2 characteristic functions ψ_0, ..., ψ_{f^2-1}, where f is the zoom factor. This paper will be concerned only with binary zooming by an integer factor *f*. Moreover, to simplify notation, we will assume that the column and row zoom factors are equal. For example, *f*=2 increases the spatial resolution twice in each coordinate. Each characteristic function is a Boolean function $\psi_i:\{0,1\}^w\rightarrow\{0,1\}$ and sometimes we will refer to the set of f^2 functions ψ_i as ψ ($\psi:\{0,1\}^w\rightarrow\{0,1\}^{(f^2)}$.) The functions ψ_i convert an input pixel *p* into f^2 output pixels y_i based on the content of the window $\vec{W}$ shifted to *p*, i.e., for $0\le i<f^2$ (figure 3):

$$y_i=\Psi(Q)(f\,p+d_i)=\psi_i(Q(W_1+p),\ldots,Q(W_w+p)),$$

where $p \in \mathbb{Z}^2$ and d_i is the displacement vector associated with i-th characteristic function. For example, in figure 3, the characteristic functions ψ_0, ..., ψ_3 convert pixel p into pixels y_0, ..., y_3 based on the content of 3×3 neighboring window.

In order to apply a WZ-operator to an image $Q^{\mathbf{x}}$, the ek-NN rule must be applied to each pattern to-be-zoomed $q^{\mathbf{x}}_{p_i}$ of $Q^{\mathbf{x}}$. Unfortunately, this process is excessively slow: to zoom one single pixel, the whole sample image $A^{\mathbf{x}}$ must be scanned. Our experiences show that this naive algorithm may take months or even years to zoom an image, using a conventional computer.

We use look-up-table (LUT) to accelerate this process. LUT allows implementing the exact ek-NN learning and is extremely fast in the evaluating time, what makes it suitable for real-time applications. But its demand for memory and training time increases exponentially as the window size grows. This makes it impossible to use this method for large windows. Fortunately, experiments have shown that windows as small as 3×3, 4×4 or a window with 17 peepholes (figure 4) can generate good WZ-operators for zooming printed and handwritten documents.

Clearly, a Boolean function $\psi:\{0,1\}^w \rightarrow \{0,1\}$ can be represented as a LUT with 2^w rows, numbered from 0 to 2^w-1, where each cell is either 0 or 1. Thus, a table to represent f^2 functions must have 2^w rows and f^2 columns, occupying $2^w f^2$ bits. For example, using a 4×4 window and zoom factor f=3, the table will occupy 589824 bits or 73728 bytes. Each column will represent a characteristic function or hypothesis $\hat{\psi}_i$.

The k-NN learning process must fill out the LUT. The index l of each row represents a binary pattern $q^{\mathbf{x}}_l$, w bits long. For each $q^{\mathbf{x}}_l$, the ek-NN rule must be applied. This process can be substantially accelerated by creating an array where each input pattern of $A^{\mathbf{x}}$ appears only once, together with the number of votes for white and black outputs. Then, the searching is performed in this array instead of in $A^{\mathbf{x}}$. Note that the array of non-repeating patterns can be created quickly using any $O(m \log m)$ sorting algorithm, like quicksort or heapsort (Cormen et al., 1990), followed by an $O(m)$ algorithm to eliminate repeated patterns.

Once LUT is completely filled, given a pattern to-be-zoomed, the zoomed output-pixels y_i can be computed effortlessly by simply indexing the corresponding row in LUT.

B. Printed Character Resolution Increasing

In order to test the ideas above expounded, we have designed a 3×3 WZ-operator to increase the resolution of documents containing characters Times 12 pt. (both normal and italic) from 300 dpi to 600 dpi. The sample images have been obtained by printing electronic documents to computer files using a PostScript printer driver, and then converting PostScript files into binary images. Although the sample images are noiseless, the problem must be considered noisy, because one 3×3 pattern in 300 dpi may correspond to two or more different patterns in 600 dpi.

Let us use (2) to estimate the size of the training images needed to obtain the WZ-operator $\hat{\Psi}$ with error rate at most 2% higher than the optimal 3×3 operator. Using confidence level 99%:

$$m \geq \frac{1}{2\varepsilon^2}\left[\ln\left(\frac{1}{\delta}\right)+\ln\left(2|H|\right)\right]=\frac{1}{2\times 0.02^2}\left[\ln\left(\frac{1}{0.01}\right)+\ln(2)+2^9\times\ln(2)\right]\cong 450238$$

We have two pairs of independent sample images ($A^{\mathbf{x}}$, $A^{\mathbf{y}}$) and ($Q^{\mathbf{x}}$, $Q^{\mathbf{y}}$) with characters Times 12 pt. (figure 5). The sizes of these images are (554×813, 1108×1626) and (558×740, 1116×1480), respectively. Image $A^{\mathbf{x}}$ is large enough to yield the requested accuracy, since 554×813=450402. A LUT was constructed using the e1-NN rule. The training took 5s and the application less than 1s in a Pentium 300.

The processed image $\hat{Q}^{\mathbf{y}}$ (figure 5c) and the ideal image $Q^{\mathbf{y}}$ (figure 5b) differed in 1.058% of pixels and they are very similar visually. Note that actually 4 independent characteristic functions were designed and their individual e-errors are depicted in the first row of table 1. Once the e-errors have been measured, the following question may arise: “Is it possible to increase substantially the accuracy of the designed operator?” We will use (4) and (5) to show that it is impossible to get any substantial gain in the quality of the WZ-operator, as long as a 3×3 window is used. We will show that:

1) The measured e-error of $\hat{\Psi}$ is a good estimate of its true-error.

2) The optimal 3×3 operator’s true-error is very close to that of $\hat{\Psi}$.

Using equation (4), with confidence 99%:

$$\text{error}_P(\hat{\psi}) \le 0.01058 + 2.33\sqrt{\frac{0.01058(1-0.01058)}{1116\times 1480}} = (1.058 + 0.019)\%,$$

what demonstrates the first statement.

To show the second statement, we have designed the WZ-operator $\hat{\Psi}^*$ e-optimal over ($Q^{\mathbf{x}}$, $Q^{\mathbf{y}}$) and applied it to image $Q^{\mathbf{x}}$. The obtained e-errors are shown in the second row of table 1. Using the obtained data and equation (5), we conclude with 99% of confidence that:

$$\text{error}_P(\psi^*) \ge 0.01045 - 2.33\sqrt{\frac{0.01045(1-0.01045)}{1116\times 1480}} = (1.045 - 0.018)\%.$$

This clearly shows that there cannot exist any 3×3 WZ-operator substantially better than $\hat{\Psi}$, because, with probability 99%, the true-error of $\hat{\Psi}$ is at most 1.077% while with the same probability the optimal 3×3 operator's true-error is at least 1.027%.

Once we have demonstrated that the designed operator is virtually the optimal 3×3 operator, another question can arise: "Can the quality of the operator be improved if a larger window were used?" We repeated the tests using a weightless 17-window (figure 4). The third row of table 1 shows the obtained e-error. The quality of WZ-operator improved only slightly. Besides, row 4 shows that, even using a 17-window, the minimal error cannot be substantially smaller than 0.92%. This time, the training took 148s but the application still took less than 1s.

C. Evaluating ek-NN Inductive Biases

In this subsection, we will test whether the inductive bias of ek-NN learning is effective for the resolution increasing. To this purpose, different ek-NN learnings have been compared with the e-optimal learner with random inductive bias. A learner with random inductive bias classifies any unseen pattern at random. In order to make the differences of inductive biases evident, rather small training images (116×516, 232×1032) were used. On the other hand, the test images ($Q^{\mathbf{x}}$, $Q^{\mathbf{y}}$) were fairly large (740×558, 1480×1116) to get an accurate estimate of true-error rate. The tests have been repeated 3 times, each time using a completely independent set of images.

The results are depicted in table 2. The first row presents error rates of e-optimal learner with random bias. Rows 2-6 present e-errors of ek-NN learning for different values of *k* using a

weightless 17-window, and rows 7-11 using a weighted 17-window. Row 12 is the e-error of the operator e-optimal over the test images. Finally, as a mere curiosity, row 13 shows the e-errors obtained by simply replicating each pixel four times.

To show the effectiveness of ek-NN inductive bias, let us compare random bias (row 1) with weightless e1-NN (row 2). Note that weightless e1-NN presents the highest error rate among ek-NN's. The average difference between two learners was $\overline{\delta} = 0.437\%$. Is this difference statistically significant? To answer this question let us construct a confidence interval. Using equation (8), with confidence 95%:

$$\underset{\vec{a}\in P^m}{E}\left[error_P(\mathbf{A}_1(\vec{a})) - error_P(\mathbf{A}_2(\vec{a}))\right] > (0.437 - 0.025)\% .$$

This clearly shows that ek-NN inductive bias helps to lessen the error rate.

According to table 2, it seems that weighted windows yield less errors than weightless ones and that the error becomes minimal for $k \cong 10$. But, as these differences are tiny, more tests are needed to validate these suppositions.

D. Handwritten Documents

The technique above has also been applied to handwritten documents (figure 6). The sizes of both the training images ($A^{\mathbf{x}}$, $A^{\mathbf{y}}$) and the test images ($Q^{\mathbf{x}}$, $Q^{\mathbf{y}}$) were (672×848, 1344×1696). The training took 9s using a weightless 3×3 window while the application took less than 1s. Image 6a is the original document $Q^{\mathbf{x}}$, 6b is the ideal output $Q^{\mathbf{y}}$ and 6c is the processed image $\hat{Q}^{\mathbf{y}}$. The difference between images $Q^{\mathbf{y}}$ and $\hat{Q}^{\mathbf{y}}$ is 1.14%. The 3×3 operator $\hat{\Psi}^*$, e-optimal over test images, presented e-error rate 1.13%. This clearly shows that the designed WZ-operator is virtually the optimal one.

As a curiosity, the 3×3 WZ-operator designed to increase the resolution of printed documents was applied to the handwriting 6a, generating figure 6d. The e-error was 1.56%. This shows that the WZ-operator designed to zoom printed characters is not fit to zoom handwritten documents. Generally, the fitness of a WZ-operator depends on the context of application.

IV. Conclusions

In this paper, a new technique to increase the spatial resolution of binary images has been presented. It is based on the *k*-nearest neighbor learning, which has a good inductive bias that allows reducing the quantity of training sample images needed to get some given accuracy (sample complexity). Some machine learning and statistical estimation techniques have been presented to estimate the sample complexity, the accuracy of the designed operator and the accuracy of the optimal operator. The statistical estimation has also been used to show the effectiveness of the *k*-nearest neighbor inductive bias for the problem addressed. The operator generated by learning is stored in a look-up-table, in order to speed up the application. The construction of the look-up-table was greatly accelerated by using some temporary data structures. The proposed technique has been implemented and proved to be both extremely fast and accurate.

Acknowledgments

The author would like to thank Drs. M. G. S. Bruno and R. J. S Carmo for their valuable suggestions and comments.

References

M. Anthony and N. Biggs, Computational Learning Theory - An Introduction, Cambridge University Press, 1992.

T.H. Cormen, C.E. Leiserson and R.L. Rivest, Introduction to Algorithms, The MIT Press, 1990.

E.R. Dougherty, Optimal Mean-Square N-Observation Digital Morphological Filters - I. Optimal Binary Filters, CVGIP: Image Understanding 1 (vol. 55, 1992), pp. 36-54.

R.C. Gonzalez and R.E. Woods, Digital Image Processing, Addison-Wesley, 1992.

D. Haussler, Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications, Information and Computation 100 (1992), pp. 78-150.

R. Jones and I. Svalbe, Morphological Filtering as Template Matching, IEEE T Pattern Analysis Machine Intelligence 4 (vol. 16, 1994), pp. 438-443.

R.P. Loce and E.R. Dougherty, Enhancement and Restoration of Digital Documents: Statistical Design of Nonlinear Algorithms, SPIE Press, 1997a.

R.P. Loce, E.R. Dougherty, R.E. Jodoin and M.S. Cianciosi, Logically Efficient Spatial Resolution Conversion Using Paired Increasing Operators, Real-Time Imaging 1 (vol. 3, 1997b), pp. 7-16.

H.Y. Kim, Quick Construction of Efficient Morphological Operators by Computational Learning, Electronics Letters 4 (vol. 33, 1997), pp. 286-287.

H.Y. Kim and F.A.M. Cipparrone, Automatic Design of Nonlinear Filters by Nearest Neighbor Learning, Proc IEEE Int Conf Image Processing, 1998, vol. 2, pp. 737-741 (TP7.05).

H.Y. Kim, Segmentation-Free Printed Character Recognition by Relaxed Nearest Neighbor Learning of Windowed Operator, Proc Brazilian Symp on Computer Graphics and Image Processing 1999, pp. 195-204.

H.Y. Kim and P.S.L.M. Barreto, Fast Binary Image Resolution Increasing by k-Nearest Neighbor Learning, Proc IEEE Int Conf on Image Processing, 2000, vol. 2, pp. 327-330 (TA9.06).

T.M. Mitchell, Machine Learning, WCB/McGraw-Hill, 1997.

L. Robert and G. Malandain, Fast Binary Image Processing Using Binary Decision Diagrams, Computer Vision and Image Understanding 1 (vol. 72, 1998), pp. 1-9.

	window	ψ_1	ψ_2	ψ_3	ψ_4	average
$\text{error}_{\vec{q}}(\hat{\psi}_i)$	3×3	1.10%	1.03%	1.05%	1.05%	1.058%
$\text{error}_{\vec{q}}(\hat{\psi}_i^*)$ e-optimal over $\vec{q}$	3×3	1.09%	1.02%	1.04%	1.03%	1.045%
$\text{error}_{\vec{q}}(\hat{\psi}_i)$	17	1.01%	0.95%	1.00%	1.02%	0.995%
$\text{error}_{\vec{q}}(\hat{\psi}_i^*)$ e-optimal over $\vec{q}$	17	0.95%	0.89%	0.92%	0.92%	0.920%

Table 1: Empirical errors using e1-NN rule.

	window	test 1	test 2	test 3	average
1. Random bias	17	1.638%	1.680%	1.622%	1.647%
2. e1-NN	17 weightless	1.218%	1.238%	1.174%	1.210%
3. e5-NN	17 weightless	1.208%	1.234%	1.166%	1.203%
4. e10-NN	17 weightless	1.206%	1.236%	1.168%	1.203%
5. e20-NN	17 weightless	1.212%	1.241%	1.166%	1.206%
6. e40-NN	17 weightless	1.218%	1.244%	1.168%	1.210%
7. e1-NN	17 weighted	1.191%	1.206%	1.143%	1.180%
8. e5-NN	17 weighted	1.180%	1.202%	1.130%	1.171%
9. e10-NN	17 weighted	1.178%	1.199%	1.129%	1.169%
10. e20-NN	17 weighted	1.180%	1.200%	1.124%	1.168%
11. e40-NN	17 weighted	1.184%	1.201%	1.130%	1.172%
12. $\text{error}_{\vec{q}}(\hat{\psi}_i^*)$	17	0.920%	1.012%	0.922%	0.952%
13. Pixel replicating	-	1.540%	1.670%	1.580%	1.597%

Table 2: Empirical errors obtained to compare different inductive biases.

(1a) Noise-free input-sample $A^{\mathbf{x}}$

(1b) Noise-free output-sample $A^{\mathbf{y}}$

(1c) Image to-be-processed $Q^{\mathbf{x}}$

(1d) Ideal output-image $Q^{\mathbf{y}} = \hat{Q}^{\mathbf{y}}$

Figure 1: Emulation of an unknown windowed operator by machine learning using noiseless sample images.

(2a) Noisy input-sample

(2b) Noisy image to-be-processed

(2c) Processed-image

Figure 2: Emulation of an unknown windowed operator by machine learning using noise-corrupted input (2a) and noiseless output (1b) as sample images.

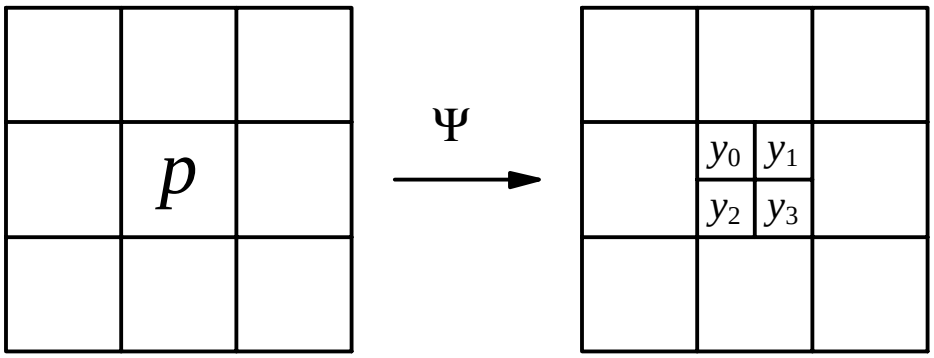


Figure 3: 3×3 windowed zoom operator with zoom factor *f*=2.

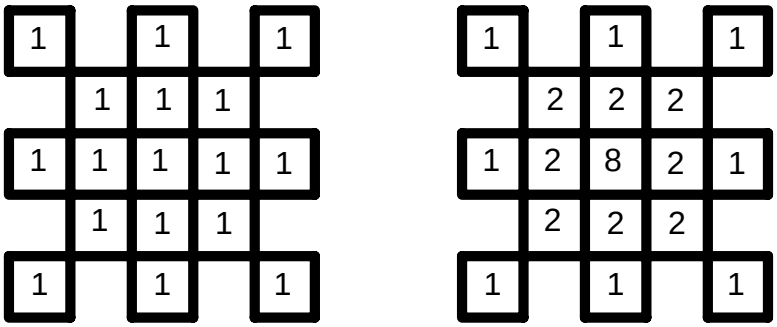
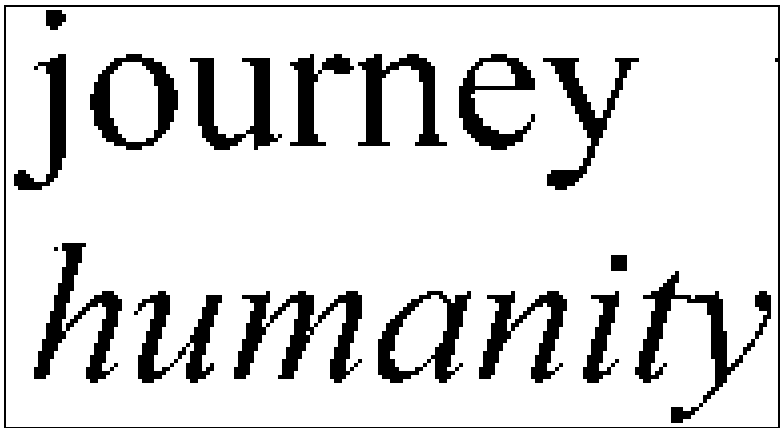
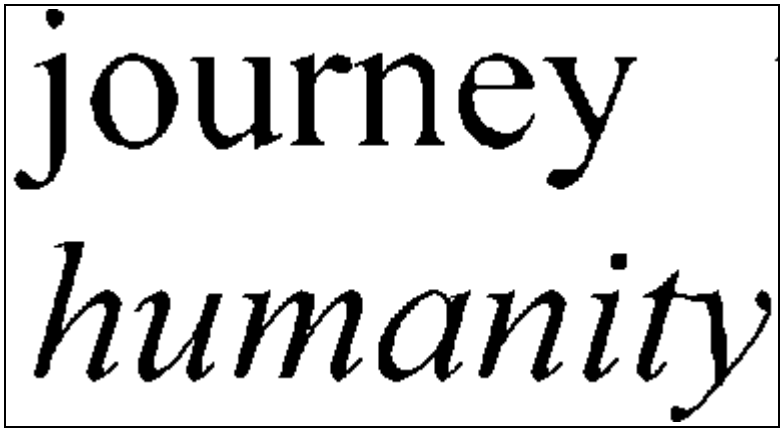


Figure 4: Weightless and weighted windows with 17 peepholes.



(5a) original image Q^{x} at 300 dpi

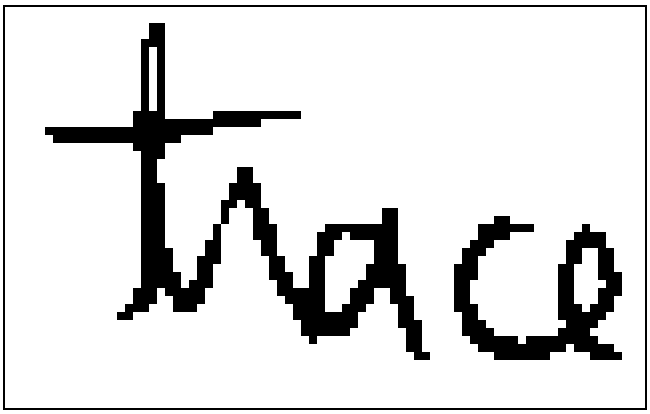
(5b) ideal output-image Q^{y} at 600 dpi



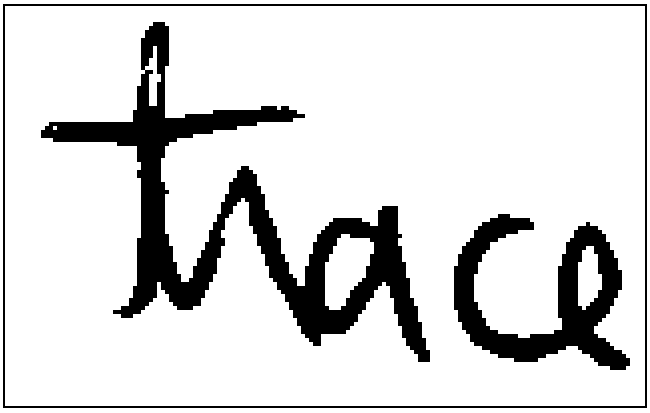
(5c) resolution-increased image $\hat{Q}^{y}$ (3×3 window)

(5d) resolution-increased image $\hat{Q}^{y}$ (17 window)

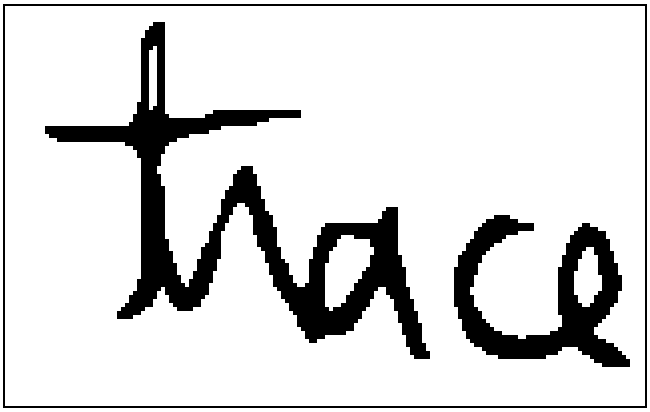
Figure 5: Resolution increasing of printed characters (Times, 12 pt.) using windowed zoom operators designed by empirically optimal 1-nearest neighbor learning.



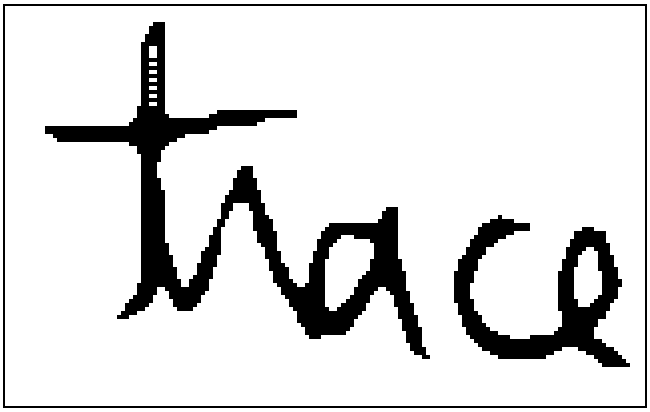
(6a) original image $Q^{\mathbf{x}}$



(6b) ideal output image $Q^{\mathbf{y}}$ (supposedly unknown)



(6c) resolution-increased image $\hat{Q}^{\mathbf{y}}$.



(6d) image obtained using operator designed to zoom printed characters.

Figure 6: Resolution increasing of handwritten document using windowed zoom operators designed by empirically optimal 1-nearest neighbor learning.