Brigham Young University
BYU ScholarsArchive

All Theses and Dissertations

2010-05-25

Transformation Learning: Modeling Transferable Transformations In High-Dimensional Data

Christopher R. Wilson
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Computer Sciences Commons

BYU ScholarsArchive Citation

Wilson, Christopher R., "Transformation Learning: Modeling Transferable Transformations In High-Dimensional Data" (2010). *All Theses and Dissertations*. 2334.
https://scholarsarchive.byu.edu/etd/2334

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

المنارة للاستشارات

www.manaraa.com

Transformation Learning: Modeling Transferable Transformations In High-Dimensional Data

Christopher R. Wilson

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Dan Ventura, Chair
Christophe Giraud-Carrier
Robert Burton

Department of Computer Science

Brigham Young University

August 2010

Copyright © 2010 Christopher R. Wilson

All Rights Reserved

www.manaraa.com

ABSTRACT

Transformation Learning: Modeling Transferable Transformations In High-Dimensional Data

Christopher R. Wilson

Department of Computer Science

Master of Science

The goal of learning transfer is to apply knowledge gained from one problem to a separate related problem. *Transformation learning* is a proposed approach to computational learning transfer that focuses on modeling high-level transformations that are well suited for transfer. By using a high-level representation of transferable data, transformation learning facilitates both shallow transfer (intra-domain) and deep transfer (inter-domain) scenarios. Transformations can be discovered in data using manifold learning to order data instances according to the transformations they represent. For high-dimensional data representable with coordinate systems, such as images and sounds, data instances can be decomposed into small sub-instances based on coordinates. Coordinate-based transformation models trained using these sub-instances can effectively approximate transformations from very small amounts of input data compared to the naive transformation modeling approach. In addition, these models are well suited for deep transfer scenarios.

www.manaraa.com

Contents

المنارة للاستشارات

www.manaraa.com

المنارة للاستشارات

www.manaraa.com

List of Figures

المنارة للاستشارات

www.manaraa.com

List of Tables

المنارة للاستشارات

www.manaraa.com

1 INTRODUCTION

Machine learning algorithms, in general, are very effective at discovering models for specific problems, but much less successful in generalizing that knowledge to other problem domains. This is often true even when the separate problems share many similar or analogous elements. Humans, on the other hand, are exceptionally good at this. People frequently apply knowledge from one domain to other related domains [16]. For example, a trained pianist would be able to apply some of his knowledge (such as reading sheet music) to the task of learning to play the saxophone. The goal of learning transfer is to effectively take knowledge about one problem and apply it to another related problem.

For many applications, gathering sufficient data and then training a computational model to represent it is an expensive and time consuming task. By using learning transfer to leverage knowledge from an already existing model instead of building a new model from scratch, the associated costs can be greatly reduced. Even if there is sufficient data to build a reasonable model, learning transfer can sometimes be used to further increase the performance and/or accuracy of that model. For example, a robot learning to navigate a maze can often do so more effectively if it can leverage knowledge gained from navigating other similar mazes.

In addition, because learning transfer is relatively easy for humans and relatively difficult for computers, it is also interesting in terms of understanding human cognition. What is it about the way humans think that makes it so “easy” to generalize knowledge? If computers can be made to transfer knowledge effectively, the solution might provide at least one plausible cognitive model.

المنارة للاستشارات
www.manaraa.com

Learning transfer is broad enough that it encompasses a wide variety of learning tasks and scenarios with the same basic goal of using learning from one task to assist in learning another. Survey papers on the subject usually attempt to divide the learning transfer "landscape" into several subfields [14, 15, 22]. This section discusses the areas most relevant to transformation learning.

One way to categorize learning transfer is in terms of sequential and functional transfer. Sequential transfer involves transferring knowledge through time, using knowledge from previous tasks to aid in learning new tasks. When this is done by creating a shared representation between tasks, this is also known as representational transfer. Thrun's "Lifelong Learning" is an example of sequential transfer [21, 20]. Functional Transfer (also known as parallel transfer) refers to transferring knowledge between tasks that are being learned at the same time. Multi-task Learning (MTL) is an example of this kind of transfer [2]. Intrator & Edelman's Low-Dimensional Representation approach is also a form of functional transfer [10]. Face Recognition is a problem that can potentially benefit from both types of transfer. If the face recognition model is trained on one face at a time, using knowledge from previously learned faces to aid in the training, then it is an example of sequential transfer. If the model is trained on all faces simultaneously, so that each one acts as a bias for all of the others, then it is an example of functional transfer. Because transformation learning learns transformations from one dataset before applying those transformations to a new dataset, it is primarily a form of sequential transfer.

Transfer can also be characterized in terms of the similarity between the domains of the source and target tasks. Shallow transfer occurs in scenarios where the source and target domains are very similar (but not identical). An example of shallow transfer would be applying knowledge about Maze A to navigating Maze B, or using knowledge about recognizing faces for Bob, Ted, and George to recognize faces for Alice and Mary. Baxter provides a formal proof that shows that shallow transfer is always possible given certain conditions, including a sufficient number of related tasks [1]. Gorski & Laird explore several

www.manaraa.com

ways of transferring knowledge in a navigation problem, all of which result in improved performance [8]. Hertzmann *et al.* show that it is possible to learn (and apply) "image analogies," successfully imitating art styles and image filters [9].

Deep transfer occurs when the source and target domains are fundamentally different, but have something in common at a higher level. An example of this would be using knowledge about birds to assist in designing aircraft, or applying knowledge about mushrooms to learning about yeast. Human analogies can also be considered a form of deep transfer: when we say that an umbrella is like a tree, the idea being transferred is the concept of shelter. Gentner argues that meaningful analogies occur when the objects being compared are structurally dissimilar but share important relationships [6]. Davis & Domingos use second-order Markov logic networks to learn and transfer high-level "rules" between tasks [4]. Both of these results point to learning high-level transformations (which can be thought of in terms of both relationships and rules) as being a promising avenue for transferring knowledge between tasks.

One useful application of learning transfer is invariance learning. In this approach, existing knowledge is used to generate a list of possible "invariances" that shouldn't affect the learning model. For example, for face recognition, the angle of the camera and the lighting should not affect the face recognition model. Lando & Edelman solve this problem by training a model that understands variation in camera angle and lighting and using that data to constrain the face recognition model [11]. Thrun generalizes the invariance approach further, using it to facilitate sequential transfer in "lifelong learning" [20, 21]. The transformation learning method described in this thesis has potential as an invariance learning method, since many invariances in a dataset can be represented as transformations (such as the aforementioned camera angles).

One way to transfer learning is to re-represent the information at a higher level, one that is more compatible with both the source and destination data. One higher-level way to look at data is in terms of transformations, as proposed by Ventura [23]. For example,

المنارة للاستشارات
www.manaraa.com

if we have time-lapse pictures of a moving car, that data can be expressed in terms of a transformation (motion) on an object (the car). If we can discover and learn the transformation (motion) in the data, we can apply that knowledge to other objects (cars or otherwise). *Transformation learning* seeks to do exactly this.

In many cases, our desired transfer tasks involve high-dimensional data such as images and sounds. However, many high-dimensional data classes (including both images and sounds) can also be represented in terms of a continuous function or a discrete approximation thereof. For example, instead of representing image data using one attribute per pixel, we can have an attribute for the X coordinate, an attribute for the Y coordinate, and an attribute for the image value at that coordinate. If we normalize these coordinates, then our representation is also invariant in terms of the actual size of the image. This "coordinate-based" way of representing high-dimensional data allows us to learn many transformations with a high degree of accuracy, even from a relatively small number of input data instances. Furthermore, this approach lends itself to models with a high degree of inter-domain transferability.

المنارة للاستشارات
www.manaraa.com

2 TRANSFORMATION LEARNING

The transformation learning process consists of the following steps: *Transformation Discovery*, *Transformation Modeling*, and *Transformation Application.*

2.1 Transformation Discovery

We define $A = \{A_i\}_{0 \leq i < m}$ to represent a dataset consisting of m unique data instances sharing a common domain D. A transformation can be represented as a function T which operates on a data instance $A_i \in A$ so that $A'_i = T(A_i)$, where A'_i is also in D. In order to control the degree of the transformation, an "amount" parameter a can be added so that $A'_i = T(A_i, a)$. This assumes that the transformation being learned can be parameterized by a single variable (the amount parameter), and is therefore considered a one-dimensional transformation. Although the transformation learning approach can be extended to any number of dimensions, doing so in a rigorous and complete way is outside the scope of this thesis. We will focus on the case where there is one transformation to be learned in the data, where the transformation is parameterizable by at most one variable. This assumption allows us to show that the proposed method works for the simplest case first, while still admitting many nontrivial transfer scenarios.

Transformation learning assumes that the data can be ordered according to the transformation being learned. In other words, the series of data instances can be ordered to represent a transformation applied to various degrees. We define a dataset A to be ordered

www.manaraa.com

according to a transformation T if these conditions hold:

1. $A_i, A_j \in A$
2. $A_i = T(A_0, a_1)$
3. $A_j = T(A_0, a_2)$
4. $i > j \Leftrightarrow a_1 > a_2$

Notationally, an ordered dataset that meets these conditions will be delimited using () instead of {}, so that the ordered dataset $A = (A_0, A_1, A_2, ..., A_{m-1})$.

The transformation discovery step is not necessary if we can guarantee that the data is already ordered according to the transformation. This step can also be skipped for the special case of single-step transformations, where the source data contains two instances representing "before" and "after" states: even if it isn't known which is which, there are only two possibilities which can be explored exhaustively.

If we cannot guarantee that the data is ordered according to the transformation being learned, we can discover an appropriate ordering using manifold learning. A manifold is a lower-dimensional surface embedded in a higher-dimensional space, and manifold learning is a class of techniques and algorithms for finding and representing that manifold [3]. Manifold learning has been shown to be effective at finding underlying patterns inherent in high-dimensional data such as images. For example, given several images of a camera panning across a room from left to right, we can discover a one-dimensional manifold that represents that panning action. Or, given a set of images representing an object in various states of rotation, we can find a manifold that represents that rotation (Figure 2.1). Figure 2.2 illustrates a manifold containing dimensions for vertical and horizontal viewing angles, as well as a dimension for lighting.

Our assertion is that this property of discovering underlying patterns makes manifold learning an effective mechanism for discovering transformations. This has been shown

www.manaraa.com

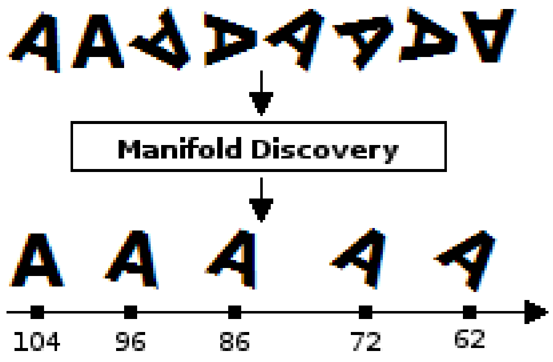


Figure 2.1: Transformation discovery finds the underlying manifold representing the transformation, and maps the original data along that manifold.

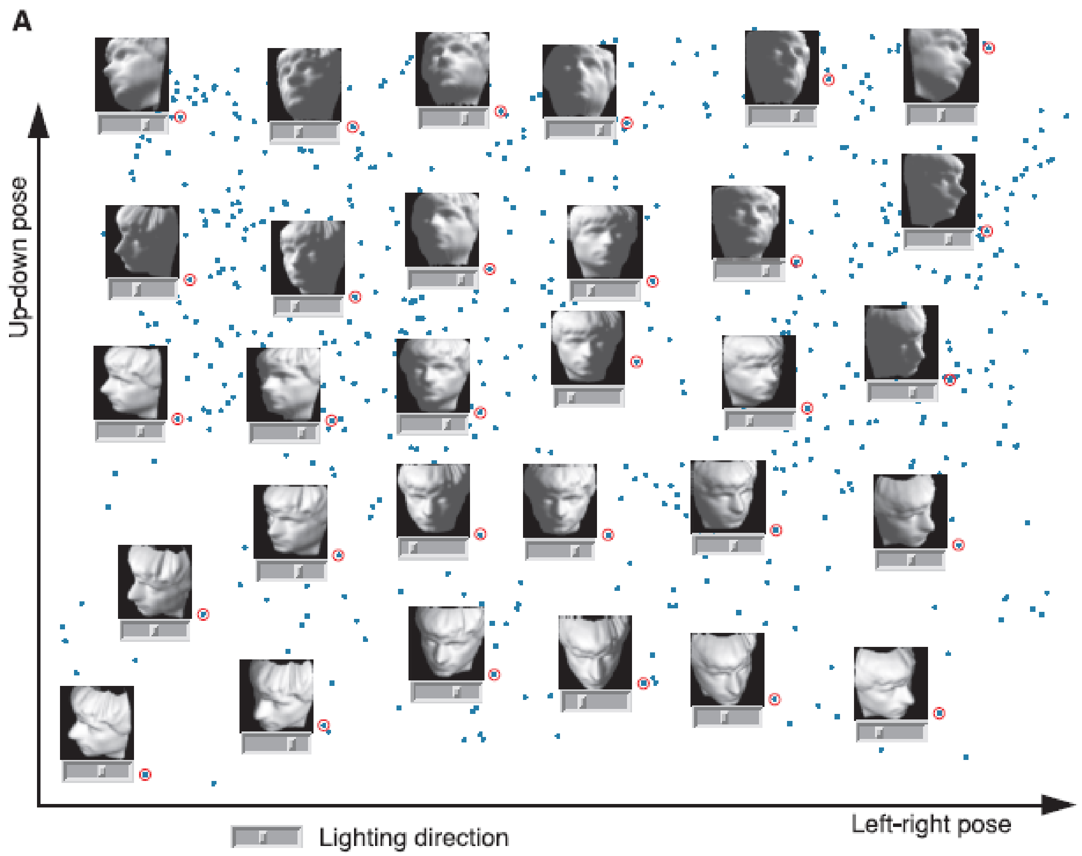


Figure 2.2: Mapping intrinsic dimensions with manifold learning (from Tenembaum *et al.*, Fig. 1A) [19]. Intrinsic dimensions for viewing and lighting angles are extracted from high-dimensional pixel data using the Isomap manifold learning algorithm.

www.manaraa.com

in other research [19], and has been consistently confirmed in our own results for one-dimensional transformations.

The direction of the ordering discovered is arbitrary. For example, given a set of images in various states of blur, a manifold learner may order them from least blurred to most blurred or vice-versa. In general, this is not a problem; prior knowledge could be used to select the desired ordering direction, or both ordering directions could be used to discover a transformation for each.

The usefulness of manifold learning in this step is that it provides conditions under which transformations in the data can be discovered automatically. Given a dataset that has been ordered using manifold learning, and assuming that the data was (or could have been) generated by applying various amounts of a transformation to a starting data instance, the definition of an ordered dataset can be used to reconstruct the transformation.

This is particularly useful if the underlying transformation isn't obvious or doesn't have a closed-form representation. It is also useful if it is necessary to construct a training dataset from real-world data where the underlying transformation is known to be present but there doesn't exist any *a priori* ordering or labeling of that transformation.

In general, a dataset may contain more than one transformation. Although this thesis focuses on the case where there is a single transformation to be discovered, it should be possible to use manifold learning techniques to discover multiple transformations reflected in a dataset. If the data contain n underlying transformations, it may be possible to extract an n-dimensional manifold representing those transformations as demonstrated by Figure 2.2. Manifold learning algorithms generally require the target dimensionality of the embedded manifold to be specified as a parameter. In cases where the intrinsic dimensionality of the embedded manifold is not known, there are ways (such as Levina & Bickel's Maximum Likelihood Method) for estimating the intrinsic dimensionality [12].

المنارة للاستشارات
www.manaraa.com

There are many manifold learning algorithms available, including Isomap [19], LLE (Locally Linear Embedding) [17], and Manifold Sculpting [5]. Our results were obtained primarily using Manifold Sculpting.

2.2 Transformation Modeling

Once an ordering of the data corresponding to a transformation has been provided or discovered, we can train a model to represent that transformation. The general approach is to construct a training set from the data and its ordering suitable for a standard machine learning algorithm. The experiments and the models discussed below are all based on a multilayer perceptron with one hidden layer trained using backpropagation.

2.2.1 Naive Modeling Approach

For a dataset with n attributes, one simple way to model the transformation is to have training set instances that each have an input and output each of size n, where the input and the output both contain each attribute of D. For single-step transformations, where the data consists of a simple before/after pair so that $A = (A_0, A_1)$, the training set is constructed as $W = A_0 \rightarrow A_1$.

In the case where the transformation is ordered along an axis (and therefore parameterizable by some “amount” factor), we can construct a training set that includes an amount parameter. This amount parameter is based on the ordering discovered or provided for the transformation. Given m data instances $(A_0, A_1, ..., A_{m-1})$ in this ordering, we can construct the training set W as follows:

المنارة للاستشارات
www.manaraa.com

$$W = \left\{ \begin{array}{c} (A_0, 0) \rightarrow A_0 \\ (A_0, 1) \rightarrow A_1 \\ (A_0, 2) \rightarrow A_2 \\ \vdots \\ (A_0, m-2) \rightarrow A_{m-1} \end{array} \right\}$$

Each A_i represents a complete data instance, including all of the attributes. The model now includes an additional input attribute representing the transformation amount.

Some transformations have the additional property of being additive. Given amounts a_1 and a_2, T is additive if:

$$T(T(A_i, a_1), a_2) = T(A_i, a_1 + a_2)$$

Rotation is an example of an additive transformation: rotating an image by 3°, then by 7°, is equivalent to rotating an image by 10° (ignoring any artifacts). A brightness threshold on an image, on the other hand, is not additive. Thresholding an image at 20% brightness, then at 30% brightness, is not the same as thresholding it at 50% brightness. If we are willing to assume that the transformation being learned is additive (or approximately so), we can further extend this training set as follows:

$$W = \left\{ \begin{array}{c} (A_0, 0) \rightarrow A_0, (A_0, 1) \rightarrow A_1, (A_0, 2) \rightarrow A_2, \ldots, (A_0, m-2) \rightarrow A_{m-1}, \\ (A_1, 0) \rightarrow A_1, (A_1, 1) \rightarrow A_2, (A_1, 2) \rightarrow A_3, \ldots, (A_1, m-3) \rightarrow A_{m-1}, \\ (A_2, 0) \rightarrow A_2, (A_2, 1) \rightarrow A_3, (A_2, 2) \rightarrow A_4, \ldots, (A_2, m-4) \rightarrow A_{m-1}, \\ \vdots \end{array} \right\}$$

More compactly, the training set can be expressed as

$$W = \{(A_i, a) \rightarrow A_{i+a}\}_{a,i}$$

www.manaraa.com

for all possible a and i. In addition, the amount parameter can be normalized based on m, which represents the cardinality of A and the maximum possible amount. This normalization can be represented as $\frac{a}{m-1}$, so that the training set can be expressed as

$$W = \left\{ (A_i, \frac{a}{m-1}) \rightarrow A_{i+a} \right\}_{a,i}$$

In some situations, it is also desirable to limit the maximum amount of a transformation that can be included in the training set. This is done by adding a constraint of $a < g$ when constructing the training set, so that

$$W = \left\{ (A_i, \frac{a}{g-1}) \rightarrow A_{i+a} \right\}_{a<g,i}$$

This constraint also affects the normalization of the amount, so that the amount parameter in the training set is normalized to the maximum possible amount. For example, when constructing a training set from a set of images rotated in increments of 1°, setting $g = 5$ trains the model so that an input amount of 1.0 corresponds to 5° of rotation. Although in many cases it is effective to use all available amount values (so that $g = m$), there are situations in which limiting the amount is valuable. For example, the spatial complexity of the training set that uses all possible amounts is $O(m^2)$ for a dataset with m instances. Training sets constructed without limiting the maximum amount can be very large, making them difficult to manage and slow to train.

If we do not want to assume the amount parameter is additive, i is restricted to being the first element in the ordering. A naive model with an amount parameter, constructed as a multilayer perceptron, is illustrated in Figure 2.3.

This approach to modeling transformations presents some difficulties. For high-dimensional data types, including images and sounds, very large amounts of data are needed before a reasonable model can be trained. As a result, models trained using this approach are likely to generalize very poorly. So many dimensions make for a very large problem

المنارة للاستشارات

www.manaraa.com

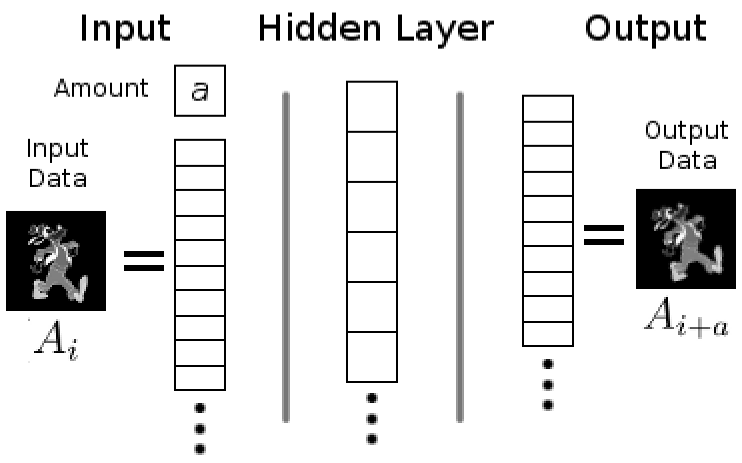


Figure 2.3: A neural network representation of the naive transformation learning approach, including an amount parameter. Each attribute in the original data is represented as a unique node for both the input and the output layers.

space, and the learner will not be able to learn a function that applies to that entire space without adequate sampling.

2.2.2 Coordinate-based Modeling Approach

Many types of high-dimensional data can also be represented as a simpler, low-dimensional function. Instead of representing a 100x100 image as a single *data instance* with one attribute per pixel, we can represent it as a set of 10,000 data *sub-instances* consisting of an x-coordinate, a y-coordinate, and a value. In order to make our models more scale-invariant, the coordinate space is normalized between 0 and 1. This "coordinate-based" representation allows us to create more effective models. By taking advantage of the original data's coordinate system, we can greatly reduce the dimensionality of the transformation function we are trying to learn. Instead of training a neural network model with n input and output nodes, we can use the coordinate system to decompose the data into more, smaller data sub-instances.

Given that D is the domain of A, we assume that D can be represented in terms of a subdomain E and a set of possible coordinates X. In images, for example, D is the domain

www.manaraa.com

representing images and E is the domain representing colors, while X is the set of possible x, y coordinates for the image. Any instance in A can be represented either in terms of D (the naive approach) or in terms of (E, X) (the coordinate-based approach). Using these coordinates, each data instance A_i can be decomposed into a new set of sub-instances where each sub-instance $A_{i_{\vec{x}}}$ is in the sub-domain E. This decomposed set contains one instance per available coordinate value, so that it becomes

$$\{A_{i_{\vec{x}}}\}_{\vec{x} \in X}$$

for a set of possible coordinates X. Using images as an example, this gives us a data sub-instance for each pixel in each image in A. This coordinate-based expansion can be performed in conjunction with the amount-based data expansion discussed earlier, so our resulting training set consists of many data sub-instances of the form

$$(A_{i_{\vec{x}}}, \frac{a}{g-1}) \rightarrow A_{(i+a)_{\vec{x}}}$$

In addition, it is useful to include the coordinate as an function parameter as well, so that our training set can be represented as

$$W = \left\{(A_{i_{\vec{x}}}, \frac{a}{g-1}, \vec{x}) \rightarrow A_{(i+a)_{\vec{x}}}\right\}_{a<g, i, \vec{x}}$$

This allows the transformation function to represent relationships dependent on the coordinate, such as the low-pass filter discussed in Section 3.3.3. Figure 2.4 illustrates this approach.

The coordinate-based approach can be further improved by allowing the training input to also contain a neighborhood window of the original data. This is possible because the existence of a coordinate system allows us to specify neighbors for any given coordinate

المنارة للاستشارات
www.manaraa.com

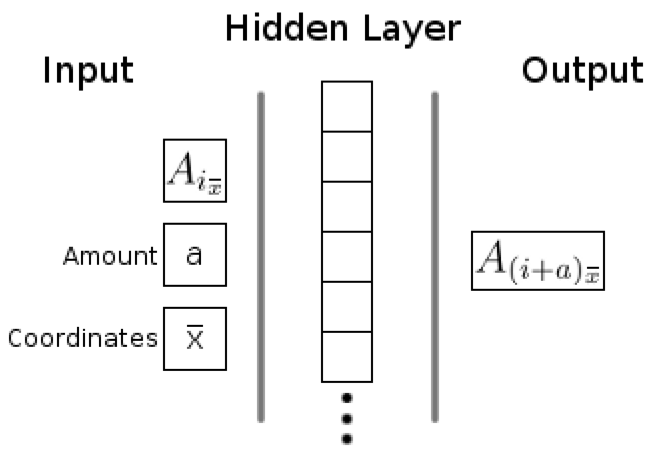


Figure 2.4: A neural network representation of the coordinate-based transformation learning model, including an amount parameter. The input is decomposed by coordinate, so that each sub-instance is representative of a specific coordinate value.

point. We assume the existence of a neighborhood function:

$$k : X \to X^s,$$

where s is the size of the neighborhood. Image data, for example, could use a square 5x5 neighborhood window such as the one illustrated in Figure 2.5, while a simple audio signal could use a similar one-dimensional window. The neighborhood function may or may not include the initial coordinate. For example, a 3x3 neighborhood window on an image may or may not include the center pixel.

We define the window function

$$K : D \times X^s \to E^s,$$

to return the set of values corresponding to a set of coordinates for a data instance. We replace the original sub-instance value parameter for the transformation function with K, so

www.manaraa.com

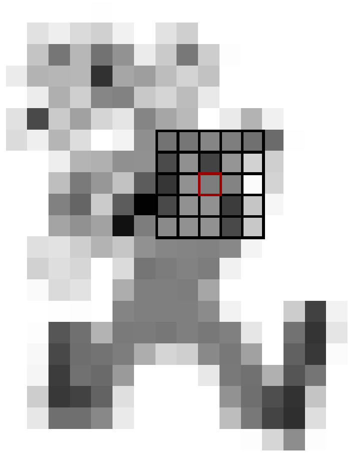


Figure 2.5: An example of a 5x5 neighborhood window on image data.

the training set can now be constructed as:

$$W = \left\{ (K(A_i, k(\vec{x})), \frac{a}{g-1}, \vec{x}) \rightarrow A_{(i+a)\vec{x}} \right\}_{a<g,i,\vec{x}}$$

Figure 2.6 illustrates a neural network model that incorporates the neighborhood window.

This neighborhood window allows our transformation model to discover local effects. For example, if our transformation involves blurring an image, the desired output value for a given pixel is a function of the values of the neighboring pixels. Without including this neighborhood window, the coordinate-based network learning the transformation model would not be able to discover and apply this relationship as easily.

The size and shape of the neighborhood window becomes an adjustable parameter for our transformation model. Larger windows allow the network to discover more far-reaching relationships: an image blur that is 10 pixels wide would be inadequately described using a window that is only 5 pixels wide. However, increasing the size of the window erodes the benefit of using the coordinate-based approach. The larger the window, the higher dimensionality the training input, and the more complex of a function the neural network needs to learn. This, in turn, increases the amount of input data needed before the model

المنارة للاستشارات
www.manaraa.com

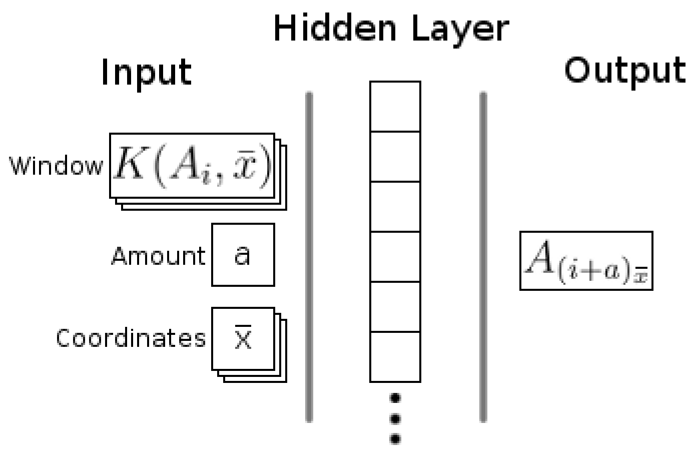


Figure 2.6: A neural network representation of the coordinate-based approach, including a neighborhood window.

can effectively generalize. In addition, if the function being learned does not need the entire neighborhood window, then the extra attributes are not actually useful and are likely to lead only to overfit on irrelevant attributes. In some cases, where neighborhood data is completely irrelevant, it may be desirable to omit it entirely. The best neighborhood window to use depends on the specific transformation to be learned.

So far, we have assumed that the output for the training set is the value of the post-transform (output) data at the same set of coordinates as the pre-transform (input) data, so that the training set is constructed as:

$$W = \left\{ (K(A_i, k(\vec{x})), \frac{a}{g-1}, \vec{x}) \to A_{(i+a)_{\vec{x}}} \right\}_{a<g,i,\vec{x}}$$

This output represents the *direct* value. Since we are dealing with transformations that involve change, however, it may also make sense to let the output represent the *change* in value, rather than the target value itself:

$$W = \left\{ (K(A_i, k(\vec{x})), \frac{a}{g-1}, \vec{x}) \to (A_{(i+a)_{\vec{x}}} - A_{i_{\vec{x}}}) \right\}_{a<g,i,\vec{x}}$$

المنارة للاستشارات

www.manaraa.com

We may also wish to use the *percent change* in value as the output, so that the training set is

$$W = \left\{ (K(A_i, k(\vec{x})), \frac{a}{g-1}, \vec{x}) \rightarrow \frac{A_{(i+a)\vec{x}} - A_{i\vec{x}}}{A_{i\vec{x}}} \right\}_{a<g,i,\vec{x}}$$

(with the special case that the output is 0 when $A_{i\vec{x}} = 0$).

Changing the representation of the training set output allows us to better fit different kinds of transformations: the best representation for a specific task can be determined experimentally. For example, when learning a low-pass filter in Fourier-space filter, it is advantageous to use the percent change because the low-pass filter can easily be represented as a change in magnitude that varies by coordinate. On the other hand, when learning something like image blur, using the change in value is useful because it more easily allows a pixel to be "added" to. A quick rule of thumb is to use percent change for transformations that are multiplicative in nature, change in value for transformations that are additive in nature, and direct value for transformations that are a direction function of the input.

Other parameters that can be set for coordinate-based models depend on the machine learning algorithm being used. The results described in Section 3 were obtained using a multilayer perceptron with one hidden layer trained using backpropagation. Although there are many possible parameters for such algorithms, the most significant one for these experiments was the activation function for the output layer. Consistently, the best results were obtained using either a *sigmoid symmetric* function:

$$y = \frac{2}{1 + e^{-2\gamma x}} - 1$$

or a *linear piecewise symmetric* function:

$$y = \begin{cases} 1 & if \;\; x \geq \frac{1}{\gamma} \\ \gamma x & if \;\; \frac{-1}{\gamma} < x < \frac{1}{\gamma} \\ -1 & if \;\; x \leq \frac{-1}{\gamma} \end{cases}$$

www.manaraa.com

Table 2.1: Summary of parameters for coordinate-based transformation modeling.

Parameter	**Possible Values**
Neighborhood Window	Any neighborhood function k, or none
Training Output	Direct, Change, Percent Change
Amount	Yes/No
Max Amount	g s.t. $0 < g <= m$
Additive	Yes/No
Activation Function	Sigmoid Symmetric, Linear Piecewise Symmetric

Coordinate-based models have a simpler structure and fewer dimensions to explore when trying to learn the transformation. This simpler structure is less prone to overfit and is capable of superior generalization from the same starting data as a naive model. As demonstrated in the results, it is often possible to generalize a good model from as little as a single example. A naive model fails completely in the same scenario. A summary of parameters to consider when constructing a coordinate-based model is presented in Table 2.1.

When applying a coordinate-based model, it is necessary to apply it many times in order to generate a final result. For example, with images, we would need to execute the trained model once for each pixel in the image. This makes coordinate-based models less time-efficient during execution. Another tradeoff is that a coordinate-based model makes an implicit assumption of function smoothness: we assume that the model behaves similarly at coordinates x and x' when x is close to x'. Although many transformations have this property, the coordinate-based approach presented here will not be effective on transformations that do not. Despite these drawbacks, coordinate-based models are the method of choice for transformation learning.

In many cases, the learned transformation model may not be capable of applying as much "amount" as is desired. If, for example, a model is trained to rotate an image up to 10°, and 20° of rotation is required, then the model can be applied more than once to achieve the desired results. This can be repeated an arbitrary number of times, the only difficulty

www.manaraa.com

being that imperfections in the model output are likely to be compounded over multiple applications of the model. Dealing with these errors is not a focus of this thesis, but results are presented in Section 3.5 to demonstrate that when these errors can be compensated for, repeatedly applying a transformation model is a viable option.

2.3 Transformation Application

The next step is to simply apply the transformation model discovered in the original (source) data to new (target) data. For this to work, however, there must be syntactic compatibility between the source and target data. The transformation model was learned in terms of a specific feature space such as a grid of pixels, a sound wave, or some other specific set of attributes. As a result, the trained model will not work properly on target data that has completely different features. For example, given a rotation transformation learned on grayscale images of a certain size, the optimal transfer target would be another grayscale image of the same size. If we tried to apply the rotation transformation to a sound wave, the result would likely be poorly defined. *Syntactic compatibility* refers to the ability to meaningfully interpret the feature space of the target data in terms of the feature space of the source data.

Syntactic compatibility exists if there exists a mapping between the source feature space and the target feature space–in other words, between the source and target domains. When the source and target domains can be represented in terms of coordinates, it is convenient to decompose the domain into a coordinate space and a subdomain, as described in Section 2.2.2. We define the forward feature space mapping function as $F : X \times E \rightarrow X' \times E'$. E and E' represent the domains of the target and source data, respectively, and X and X' represent their coordinate spaces. For each coordinate $\vec{x} \in X$ in the target domain, F is applied to that coordinate and the value $e \in E$ at that coordinate. This results in a new coordinate $\vec{x'}$ and value e' in the source domain. A transformation model T in the source domain is applied to $(\vec{x'}, e')$ giving an output $o' \in E'$. Because o' is in the source subdomain,

المنارة للاستشارات
www.manaraa.com

Table 2.2: Steps for learning transfer using transformation learning.

1.	A transformation model T is learned for the source task.
2.	For each target coordinate $\vec{x} \in X$:
2a.	The target coordinate $\vec{x} \in X$ and the value $e \in E$ at that coordinate are mapped into the source coordinate space X' and domain E' using F, yielding $\vec{x'}$ and e'.
2b.	T is applied to $(\vec{x'}, e')$, yielding $o' \in E'$.
2c.	o' is mapped back into the target task domain E using F', yielding $o \in E$.
2d.	o is applied at $\vec{x}$ in the target domain.

it needs to be mapped back into the target subdomain using a reverse feature space mapping function $F' : E' \rightarrow E$. The mapped output o is then applied for the target task at $\vec{x}$ in the target domain. Table 2.2 describes this procedure.

If the source and target exist in the same domain, then these mappings are not necessary. Otherwise, the mapping functions F and F' need to be either discovered or specified. The task of automatic feature space mapping is an open problem in learning transfer, and solving it is outside the scope of this thesis. A survey of work in this direction is discussed in Torrey & Shavlik [22]. For the purposes of the transformation learning approach, we make the assumption that if the datasets are, in fact, syntactically compatible then either there exists an automatic method for mapping between them or the mapping can be performed manually. All necessary mapping for this thesis is done manually.

While syntactic compatibility measures how similar the structure of the source and target are, *semantic similarity* measures task relatedness once structural and syntactic differences have been accounted for. In general, the deeper the transfer, the less semantic similarity between the tasks. Given a task distance metric C, where low values indicate a higher degree of relatedness, we can measure the semantic similarity between two tasks as being inversely proportionate to C. Although semantic similarity is not a requirement for transformation learning, it represents a tradeoff; if the tasks are sufficiently similar (the value of C is low), then there is little use for transformation learning in the first place. If the

www.manaraa.com

only significant difference between the tasks is their feature space, then all that is needed is feature space mapping. On the other hand, if the source and target tasks are sufficiently dissimilar (the value of C is high), it may be impossible to meaningfully transfer knowledge between them. Discovering a useful metric for C is outside the scope of this thesis, but assuming one exists, we can choose useful transfer targets by looking for tasks where the value of C is between an upper and a lower threshold.

Transformation learning will be useful for sets of tasks which are syntactically compatible (provided appropriate mappings F and F'), have a task distance C between a minimum and maximum threshold, and are defined under the same high-level transformations. By learning these transformations, we can transfer knowledge between such tasks using the steps in Table 2.2.

Determining on which tasks to use the learned transformations (or alternately, which transformations to use on novel target tasks) is a difficult and subjective problem outside the scope of this thesis. Eventually it may be possible to develop a model that can automatically determine which learned transformations are likely to apply meaningfully to potential target tasks. For this thesis, however, transfer sources and targets are determined using human expertise.

Because of their high generalization accuracy, coordinate-based models are well-suited for learning transfer applications with high-dimensional data. Coordinate-based models have fewer inputs and outputs, and this reduced dimensionality can make feature space mapping between source and target tasks easier. Feature space mapping can also be made easier by taking advantage of the structure inherent in coordinate-based systems.

المنارة للاستشارات
www.manaraa.com

3 RESULTS

This section presents experimental results for transformation learning. Although the coordinate-based approach is primarily used, the naive approach is included for some experiments as a point of comparison. A table is provided for each experiment listing parameters for the coordinate-based model. No table is provided for the naive model because it does not have any parameters that cannot be represented in the training set notation. These results show that transformation learning with coordinate-based models allows for transformation models to be effectively learned and generalized, frequently from very little data.

3.1 Manifold Discovery

As discussed in Section 2.1, manifold learning has been shown to be effective at finding underlying patterns in high-dimensional data, including transformations. For all of the amount-dependent transformations discussed in the results, a manifold was discovered using the Manifold Sculpting algorithm that represented the true transformation ordering.

Because the transformation being learned is parameterizable by a single variable (amount), the target dimensionality for the Manifold Sculpting algorithm is set to one. The other main parameter for the algorithm, neighborhood size, is set to five for all of the experiments discussed in this section.

Figure 3.1 illustrates manifolds for image rotation and blur. The image rotation data consists of 36 25x25 images of the letter 'A', rotated in successive increments of 5°. The image blur data consists of a 100x100 image of the character Goofy with the Gaussian blur described in Section 3.4.2 applied, varying the σ parameter from 0 to 5.5 in increments

www.manaraa.com

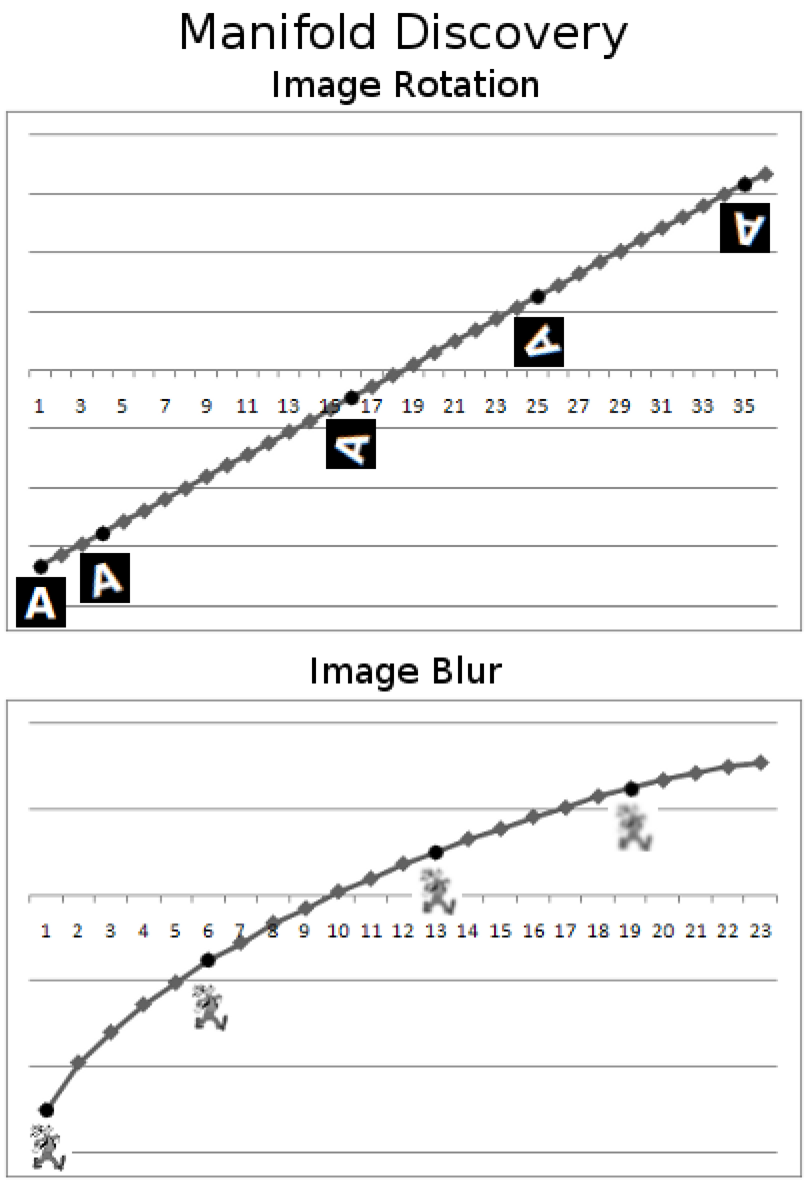


Figure 3.1: Manifold Sculpting, a manifold learning algorithm, effectively discovers the true transformation ordering. The X-axis represents the actual transformation ordering, while the Y-axis represents the ordering discovered using Manifold Sculpting with a target dimensionality of one and a neighborhood size of five. The monotonic relationship between the two axes shows that the discovered ordering is consistent with the true ordering.

www.manaraa.com

of 0.25. The X-axis of the graph represents the actual transformation ordering, while the Y-axis represents the ordering discovered using manifold learning. As demonstrated by the monotonic relationship between the X and Y axes, the discovered ordering is consistent with the true ordering.

As demonstrated by the results for image blur in Figure 3.1, the relationship between the discovered manifold coordinates and the actual transformation is not always linear. Although the amount of Gaussian blur is increased linearly for each image, the manifold learning found the images with a higher amount of blur to be more similar to their neighbors than the images with a lower amount of blur and placed them accordingly.

3.2 Manifold Noise Impact

When evaluating the usefulness of transformation learning based on the ordering discovered using manifold learning, it is worthwhile to analyze the effect that errors in this process can have on the resulting models. Figure 3.2 examines the effect of manifold noise on the resulting training data. The purpose of this is to evaluate the effect of errors that occur in the transformation discovery step, so that the discovered ordering does not match the true ordering.

The "baseline" referred to in the figure is a dataset that is already ordered according to the transformation. For the image rotation experiment, this dataset consists of 36 25x25 images of the letter 'A' rotated successively in increments of 5°, so that $A = (A_0, A_1, A_2, ..., A_{m-1})$. For the sound volume experiment, the baseline dataset consists of 11 audio examples of the spoken word "banana" with the volume successively lowered.

Based on experimenting with manifold learning as a technique for finding the transformation ordering in a dataset, we concluded that the most common error was for the order of adjacent points to be reversed. The data for this experiment is generated by randomly swapping adjacent data points in the baseline n times, simulating this kind of error. The swap operation consists of the following:

للاستشارات المنارة
www.manaraa.com

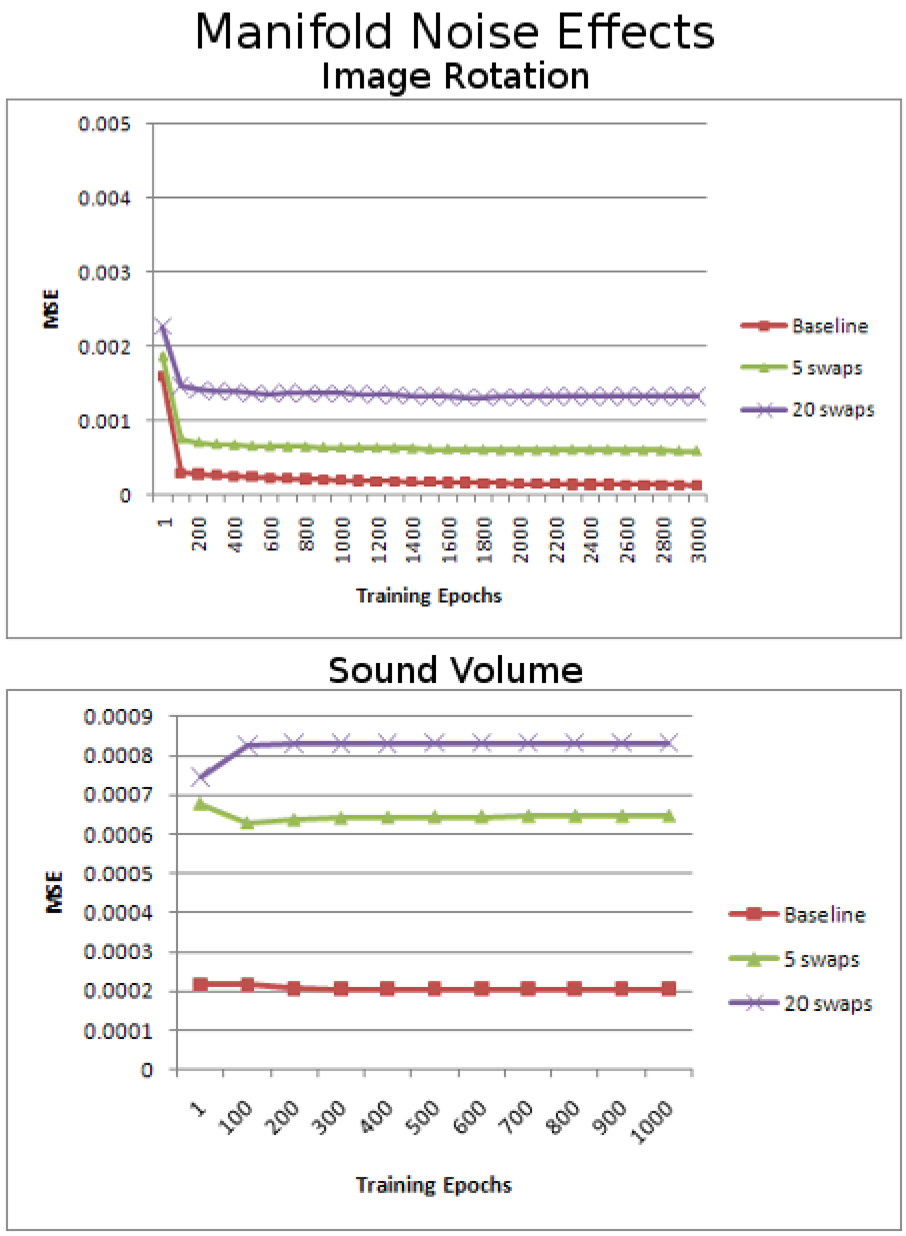


Figure 3.2: The effect of manifold noise on the ability to train transformation models. The baseline data used to generate the image rotation graph consists of 36 25x25 images of the letter 'A' in successive states of rotation. For the sound volume graph, the baseline data consists of 11 audio examples of the spoken word "banana" with the volume successively lowered. For both image rotation and sound volume, adjacent data points are randomly swapped to simulate errors in the transformation discovery step.

www.manaraa.com

Table 3.1: Model parameters for image rotation used to evaluate the impact of manifold noise

Parameter	**Value**
Neighborhood Window	5x5 square window
Training Output	Change
Amount	Yes
Max Amount	3
Additive	Yes
Activation Function	Linear Piecewise Symmetric

1. Choose a random number r so that $0 <= r < m - 1$.
2. Swap the data at A_r and A_{r+1}.

New datasets swap5, swap10, swap15, and swap20 are generated from each baseline dataset using 5, 10, 15, and 20 swaps. Coordinate-based training sets are generated for each reordered dataset as well as its cooresponding baseline.

For each image rotation dataset A the corresponding training set W is defined as

$$W = \left\{ (K(A_i, k(\vec{x})), \frac{a}{2}, \vec{x}) \rightarrow (A_{(i+a)_{\vec{x}}} - A_{i_{\vec{x}}}) \right\}_{a<3,i,\vec{x}}$$

Table 3.1 lists the parameters used for the image rotation transformation model. The maximum amount value g is set to 3 as an arbitrary way to keep the model simple for this experiment.

For each sound volume dataset A the corresponding training set W is defined as

$$W = \left\{ (\frac{a}{10}, \vec{x}) \rightarrow \frac{A_{(i+a)_{\vec{x}}} - A_{i_{\vec{x}}}}{A_{i_{\vec{x}}}} \right\}_{a<11,i,\vec{x}}$$

The parameters for the sound volume transformation model are listed in Table 3.2.

The graphs in Figure 3.2 illustrate how training accuracy over time is affected by errors in the ordering, and shows that the decrease in performance is reasonably small.

www.manaraa.com

Table 3.2: Model parameters for sound volume used to evaluate the impact of manifold noise

Parameter	Value
Neighborhood Window	None
Training Output	Percent Change
Amount	Yes
Max Amount	m
Additive	No
Activation Function	Linear Piecewise Symmetric

We can conclude that although errors in the manifold discovery process can hamper the subsequent modeling of the transformation, these errors are not catastrophic.

3.3 Single-step Transformations

This section discusses results for modeling transformations involving only a single transformation step. For all experiments discussed here, the amount parameter is unused.

3.3.1 Edge Detection

Edge detection in an image is one example of a simple single-step transformation. One common approach to edge detection in images is to use the Laplacian image convolution kernel [7]. An image convolution kernel works by sweeping a square window centered on a target pixel across the image, and at each pixel, taking a weighted sum of the values of the pixels in the window, and storing the resulting value in the target pixel. This can be expressed as

$$P_{target} = \sum_{i \in window} L_i P_i$$

for a set of pixels P and kernel L. The weights for the Laplacian kernel are described in Figure 3.3.

The training data is produced by filtering the starting 100x100 image A_0 of Abraham Lincoln through a Laplacian kernel, giving A_1. Since we have a single before/after pair and

www.manaraa.com

Figure 3.3: The Laplacian image convolution kernel [7], used for edge detection.

-1	-1	-1
-1	8	-1
-1	-1	-1

no amount, the coordinate-based training set W_c is defined as

$$W_c = \{(K(A_0, k(\vec{x})), \vec{x}) \to A_{1_{\vec{x}}}\}_{\vec{x}}$$

We choose X to be the pixel coordinates of the image, normalized to the dimensions of the image so that the top-left pixel of the image is at $(0, 0)$ and the bottom-right pixel of the image is at $(1, 1)$. The fully expanded training set follows this pattern for the 100x100 pixel image:

$$W_c = \left\{ \begin{array}{c} ((K(A_0, k(0,0)), (0,0)) \to A_{1_{(0,0)}}), ((K(A_0, k(0,.01)), (0,.01)) \to A_{1_{(0,.01)}}), \ldots \\ ((K(A_0, k(.01,0)), (.01,0)) \to A_{1_{(.01,0)}}), ((K(A_0, k(.01,.01)), (.01,.01)) \to A_{1_{(.01,.01)}}), \ldots \\ ((K(A_0, k(.02,0)), (.02,0)) \to A_{1_{(.02,0)}}), ((K(A_0, k(.02,.01)), (.02,.01)) \to A_{1_{(.02,.01)}}), \ldots \\ \vdots \end{array} \right\}$$

The training set for the naive model, designated W_n, is constructed simply as

$$W_n = \{A_0 \to A_1\}$$

Table 3.3 lists the parameters used to learn this transformation using a coordinate-based model, and Figure 3.4 illustrates the results and compares them to a naive model.

The similarity between the coordinate-based results and the "expected" results (obtained using the Laplacian image kernel) shows that a very accurate model can be learned, even when trained on only a single example of the transformation. This is likely because the model learns to approximate the Laplacian kernel by using the neighborhood window.

www.manaraa.com

Table 3.3: Model parameters for edge detection experiment

Parameter	Value
Neighborhood Window	3x3 square window
Training Output	Direct
Amount	No
Max Amount	N/A
Additive	N/A
Activation Function	Linear Piecewise Symmetric

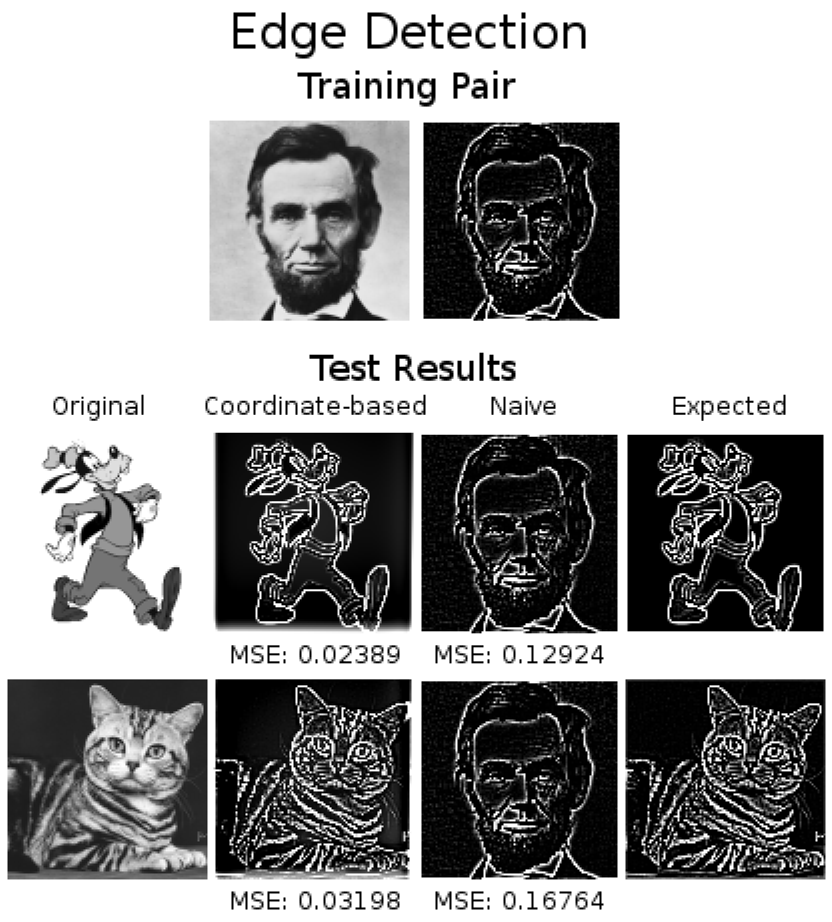


Figure 3.4: A transformation model for edge detection trained using a single image pair consisting of a 100x100 image and its edges (found using a Laplacian kernel). The coordinate-based model closely approximates the Laplacian image kernel for arbitrary test images. The naive model is only capable of producing edges for the training example.

المنارة للاستشارات
www.manaraa.com

The naive model, on the other hand, is incapable of generalizing from a single training example and produces only the output that it was trained on. Because the coordinate-based model is able to decompose the original training pair into many sub-instances, where each sub-instance is capable of accurately representing a Laplacian kernel example, it is able to successfully model edge detection.

3.3.2 Corner detection

Corner detection represents a slightly more difficult task. Although algorithms exist for corner detection in images, this experiment involved using a single manually-labeled training example to train a generalizable corner detection algorithm. The training data is generated by taking the starting 100x100 image A_0, then creating a corresponding image marking where human-perceived corners are (giving A_1). The training set is then constructed in the same way as was done for edge detection in Section 3.3.1, so that X maps to normalized pixel coordinates and the coordinate-based training set is

$$W_c = \{(K(A_0, k(\vec{x})), \vec{x}) \rightarrow A_{1_{\vec{x}}}\}_{\vec{x}}$$

and the naive training set is

$$W_n = \{A_0 \rightarrow A_1\}$$

The only difference in the coordinate-based training set construction between this experiment and the edge detection experiment is the slightly larger neighborhood window. Table 3.4 gives the parameters used for the coordinate-based model, and Figure 3.5 illustrates the results and compares them to results from the naive model. In addition, these results are compared to the output of a common corner detection algorithm, SUSAN [18]. The SUSAN algorithm is used with a difference threshold of 10 and a geometric threshold of 20.

The naive model, presented with only a single training instance, simply learns the output image and provides it for all inputs. The coordinate-based model, however, is able to

المنارة للاستشارات
www.manaraa.com

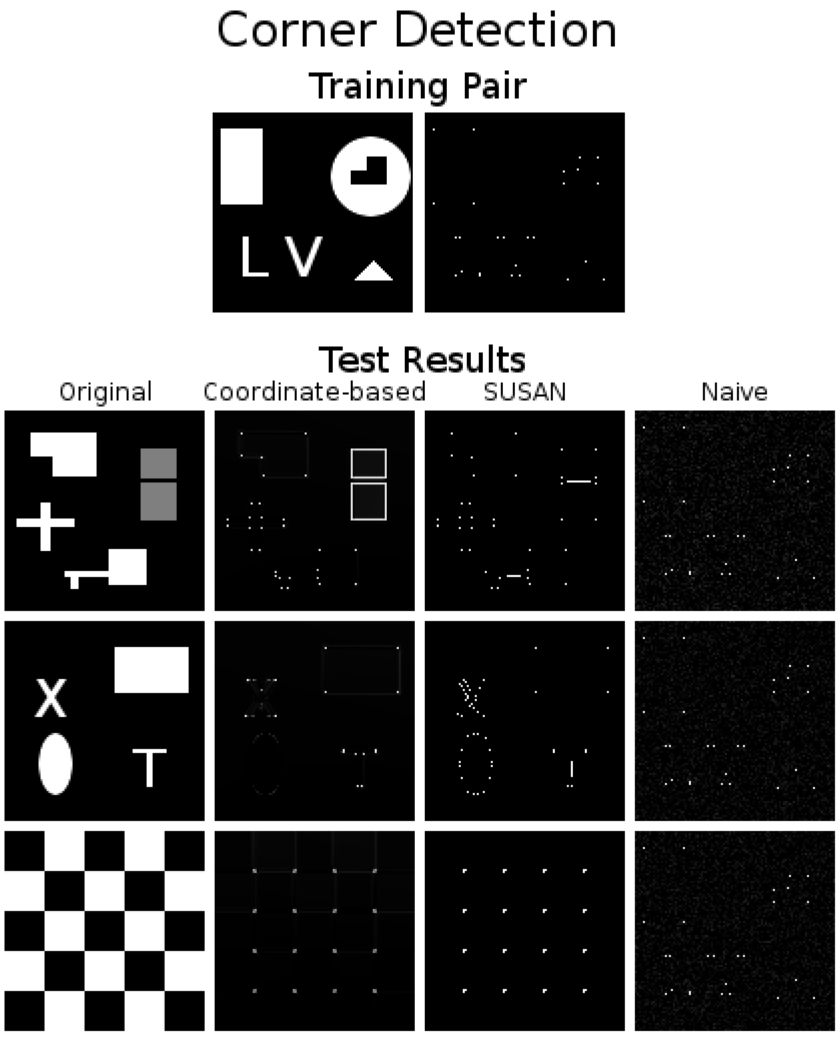


Figure 3.5: A transformation model for corner detection trained using a single 100x100 image pair consisting of an image and its corners (labeled manually). The coordinate-based model accurately finds corners in other images, with only a few errors. The naive model is only capable of finding corners in the training example. The coordinate-based model produces results competetive in quality to a SUSAN [18] corner detector with a difference threshold of 10 and a geometric threshold of 20.

المنارة للاستشارات
www.manaraa.com

Table 3.4: Model parameters for corner detection experiment

Parameter	**Value**
Neighborhood Window	5x5 square window
Training Output	Direct
Amount	No
Max Amount	N/A
Additive	N/A
Activation Function	Sigmoid Symmetric

learn a reasonable approximation of corner detection from the same data. The approximation isn't perfect: there are many places where faint lines appear, but these can be ignored with an appropriate threshold function. The most serious error is that the gray boxes in the first test example are highlighted as being all corners, which is clearly incorrect. The only grayscale values in the training image are due to the smoothing applied when rendering the circle and the letter V. Since the training example has so few examples of grayness, even when considered as a set of sub-instances, it is not surprising that the resulting model has trouble with gray values. Nevertheless, as is the case for edge detection, the result is a fairly good approximation of a corner detector. In addition this corner detection model, trained using a single manually-labeled example, performs competitively with the SUSAN [18] and Moravec [13] corner detection algorithms on the same test data.

3.3.3 Low-pass Filter

A low-pass filter is a signal processing technique that can be applied to both sounds and images. It consists of removing the high-frequency components of a signal, leaving only the lower frequencies. The result is a signal which sounds "foggier" (for sounds) or appears blurrier (for images). Because it is a transformation that is well defined for both sounds and images, it makes a good candidate for cross-domain transfer (in this case, from the sound domain to the image domain).

المنارة للاستشارات
www.manaraa.com

Table 3.5: Model parameters for low-pass filter experiment

Parameter	Value
Neighborhood Window	None
Training Output	Percent Change
Amount	No
Max Amount	N/A
Additive	N/A
Activation Function	Sigmoid Symmetric

The low-pass filter used for this experiment is constructed as a sigmoid multiplier in Fourier space, using the following formula: $f(x) = \frac{1}{1+e^{x-\mu} * \gamma}$ where x is a frequency index in Fourier space, μ is the cutoff, and γ is the gain. Given a sound in Fourier space where x is the frequency index and y is the complex value at that index, the filter is applied so that $y' = y * f(x)$ for all x.

This low-pass filter is applied to a single sound file consisting of the spoken word "cow," with $\mu = 500$ and $\gamma = 0.01$. The original sound, as well as the filtered version, are converted to Fourier space using the Fast Fourier Transform, resulting in A_0 and A_1, respectively. The training set is constructed using the coordinate-based approach with percent change output and an empty neighborhood window, so that

$$W_c = \left\{ (\vec{x}) \to \frac{A_{1_{\vec{x}}} - A_{0_{\vec{x}}}}{A_{0_{\vec{x}}}} \right\}_{\vec{x}}$$

with the special case that the training output is 0 where $A_{0_{\vec{x}}}$ is 0. This training set is used to train the transformation model for a one-dimensional low-pass filter. Table 3.5 contains the parameters used for the transformation model.

Once the one-dimensional model is trained, it is ready to be applied cross-domain to an image. The learned transformation model is in one-dimensional Fourier space (the source task), but we need to apply it to two-dimensional Fourier space (the target task). This is accomplished by following the procedure described in Table 2.2, and requires feature space

المنارة للاستشارات
www.manaraa.com

mapping functions F and F' to be defined. The subdomains E and E' for the target and source tasks are identical, but the coordinate spaces X and X' are not. The feature space translation functions, therefore, can leave the subdomains unchanged and only modify the coordinate space. Radial symmetry is assumed in the target space, so that the coordinate input to the one-dimensional model is the Euclidean distance to the origin in the target space. If the coordinates for the target (2D Fourier space) are in terms of u, v the coordinates for the source (1D Fourier space) are in terms of w, and the value at (u, v) is $e \in E$, the feature space mapping functions F and F' are defined as:

$$\begin{aligned} &F((u,v),e) = (\sqrt{u^2+v^2}, e) \\ &F'(e) = e \end{aligned}$$

The image data is mapped into sound space using F, the transformation is applied, and the result is mapped back into image space using F'. Finally, the inverse Fourier transform is performed to obtain the final image. Figure 3.6 illustrates the results, showing that the low-pass filter model trained on sound files is able to transfer successfully to images.

3.4 Amount-dependent Transformations

This section discusses results for modeling transformations that are parameterized by amount. For all of these experiments, transformation discovery correctly orders the data instances according to the transformation to be learned. Although the true ordering is used when describing these experiments, the fact that the discovered ordering matched the true ordering in every case indicates that the same results would have been obtained.

3.4.1 Image Rotation

The training data for this experiment consists of images of letters rotated successively. The goal is to learn a transformation model for rotation. The learned model is then applied to other images.

المنارة للاستشارات
www.manaraa.com

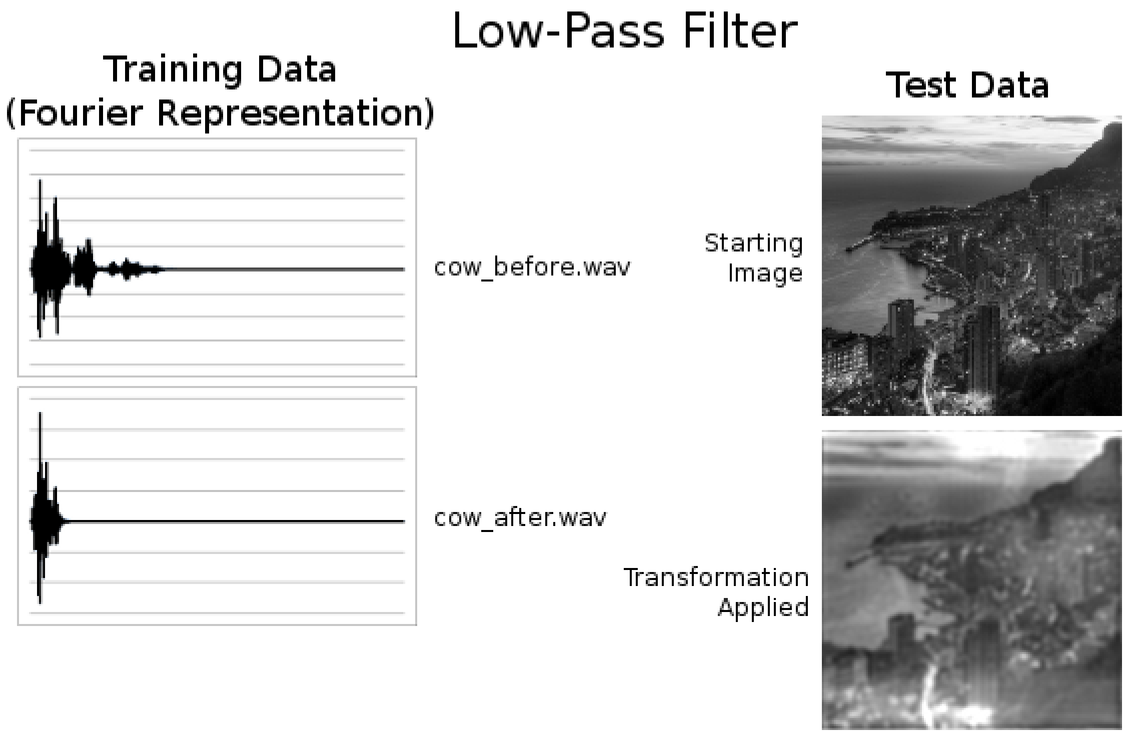


Figure 3.6: An example demonstrating learning transfer between sound and image domains. A low-pass filter is trained from a single sound file and it's filtered version, and the resulting model is applied to blur images.

www.manaraa.com

The rotation is done in increments of 5°, up to 180° of rotation. One model is trained using only the letter 'A', and another is trained using the letters A-L (for a total of 12 letters). These images all have a size of 25x25 pixels. In both cases, the training set is limited to a maximum amount of 5, and the training output is based on change, resulting in the following:

$$W_c = \left\{ (K(A_i, k(\vec{x})), \frac{a}{4}, \vec{x}) \rightarrow (A_{(i+a)_{\vec{x}}} - A_{i_{\vec{x}}}) \right\}_{a<5,i,\vec{x}}$$

The maximum amount constraint is necessary for two reasons. First, the size of the neighborhood window effectively limits the maximum amount of rotation that can be learned. Trying to train on rotation amounts beyond this produces poor results. Second, training sets for image data produced without constraining the maximum amount become too large to effectively manage and train.

The naive training set is also constructed using a constrained amount parameter, so that

$$W_n = \left\{ (A_i, \frac{a}{4}) \rightarrow A_{i+a} \right\}_{a<5,i}$$

Table 3.6 lists coordinate-based model parameters used for training using data generated from both a single letter and several letters. Figure 3.7 demonstrates the results for a single letter and compares them to a naive model. These results show that although some crude approximation of rotation is learned for the coordinate-based transformation model, the naive transformation model is unable to generalize. Figure 3.8 shows that if we add more data, in the form of additional example letters, the coordinate-based model is able to produce a reasonable approximation of image rotation. The naive model performs better than it did: the letter Z is rotated with some degree of accuracy. However, when applied to an image that isn't a letter, the result does not appear to approximate rotation.

One interesting error in the coordinate-based model is that the amount rotated by the model is less than the expected amount. This is most visible in Figure 3.8 for test examples using an amount of 0.8. One possible reason for this is that the neighborhood window isn't

المنارة للاستشارات
www.manaraa.com

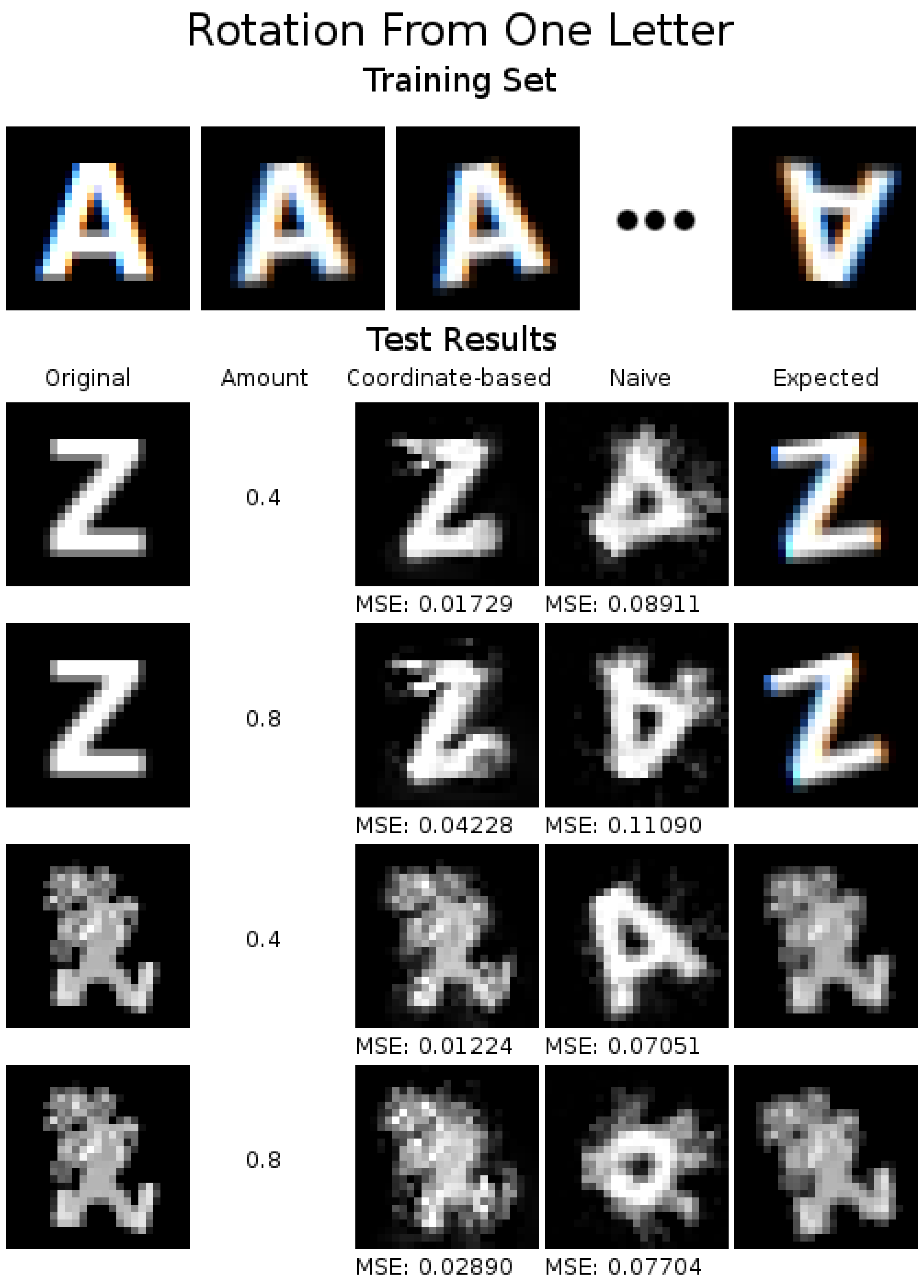


Figure 3.7: A transformation model for rotation trained using a single series of successively rotated 25x25 images of the letter 'A'. The normalized amount parameter controls the degree of rotation applied. The coordinate-based model crudely approximates rotation in this scenario, while the naive model can only produce variations of the letter 'A'.

www.manaraa.com

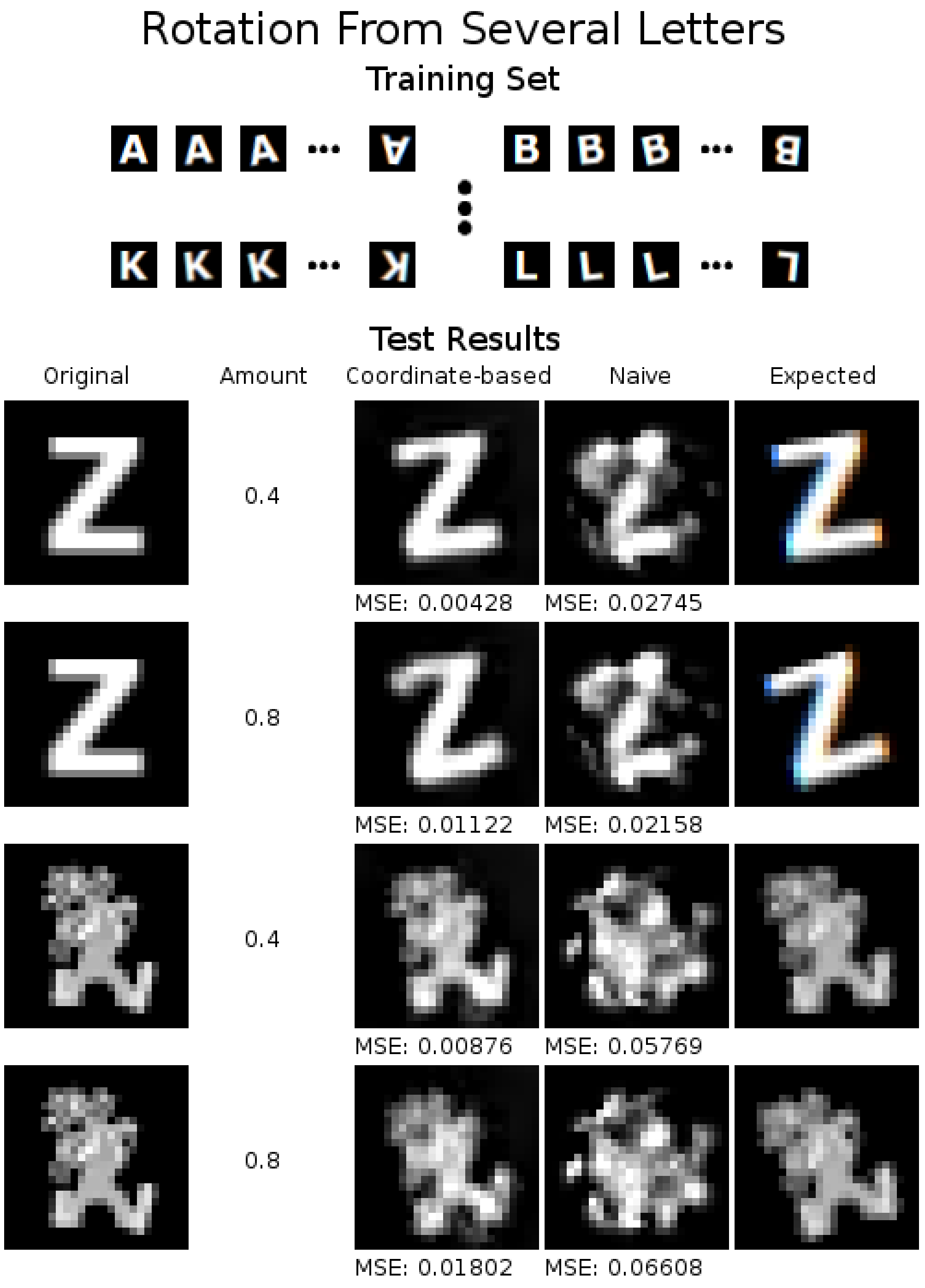


Figure 3.8: A transformation model for rotation trained using twelve series of successively rotated 25x25 images based on the letters A-L. The normalized amount parameter controls the degree of rotation applied. The additional training data allows the coordinate-based transformation model to successfully approximate image rotation more accurately than when using only one letter. The naive model is also able to take advantage of the additional training data, but still fails to produce cohesive images.

المنارة للاستشارات
www.manaraa.com

Table 3.6: Model parameters for image rotation experiments

Parameter	**Value**
Neighborhood Window	7x7 square window
Training Output	Change
Amount	Yes
Max Amount	5
Additive	Yes
Activation Function	Linear Piecewise Symmetric

quite big enough to represent the largest amounts of rotation. The model, when training, may be allowing the amount parameter to be underrepresented because doing so minimizes error. Instead of trying to rotate too far and causing large errors, the model learns to rotate a bit less than it should in order to minimize training error. If the neighborhood window size is increased to compensate, however, the model does not generalize as well on the same training data, exhibiting the tradeoff discussed in Section 2.2.2. Another possible explanation for this "compression" of the amount factor is that the larger amounts simply represent an edge case that the model does not fully explore. This theory is supported by the observation that the degree of compression observed decreases as the model is trained for more epochs.

3.4.2 Image blur

The data for this experiment consists of an image with increasing degrees of Gaussian blur applied. The Gaussian blur function on the image is

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ is the parameter that determines the width of the blur. This function is used to build an image convolution kernel of size $2\lceil 5\sigma \rceil$, where the value for each point in the kernel is calculated in terms of the x and y distance from the center of the kernel using $G(x,y)$.

www.manaraa.com

Table 3.7: Model parameters for image blur experiment

Parameter	**Value**
Neighborhood Window	7x7 square window
Training Output	Change
Amount	Yes
Max Amount	m
Additive	Yes
Activation Function	Linear Piecewise Symmetric

The resulting kernel is then applied to the image in the same manner as the Laplacian kernel in Section 3.3.1.

The training data $A = (A_0, A_1, A_2, A_3, A_4)$ consists of a 100x100 image of Abraham Lincoln with Gaussian blur applied using $\sigma = 0$,1,2,3, and 4 respectively. This blur operation is assumed to be (approximately) additive, although this is actually not quite the case for Gaussian blur. The coordinate-based training set is

$$W_c = \left\{(K(A_i, k(\vec{x})), \frac{a}{4}, \vec{x}) \to (A_{(i+a)_{\vec{x}}} - A_{i_{\vec{x}}})\right\}_{a<5,i,\vec{x}}$$

and the naive training set is

$$W_n = \left\{(A_i, \frac{a}{4}) \to A_{i+a}\right\}_{a<5,i}$$

The output of the trained model on test data is compared with the Gaussian blur algorithm that it approximates by calculating the MSE between the model's output image and the Gaussian blur algorithm's output image. Table 3.7 contains the coordinate-based model parameters used for this experiment. The results in Figure 3.9 demonstrate that the coordinate-based transformation model is able to successfully approximate the blur operator, including the ability to adjust the amount. The naive transformation model, in contrast, is unable to generalize beyond the training data. The result of applying it on the test data is a very dark copy of the Abraham Lincoln image.

www.manaraa.com

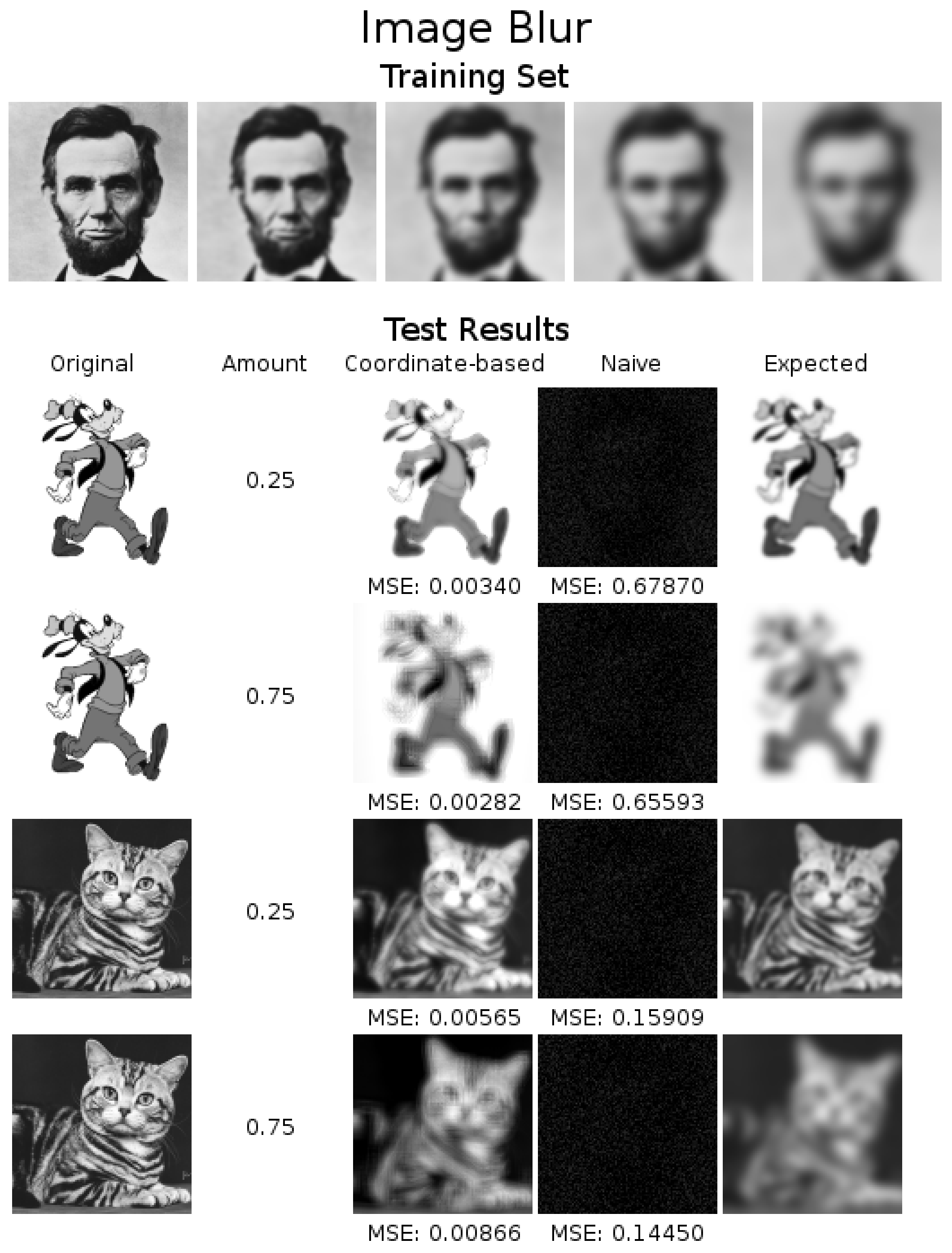


Figure 3.9: A transformation model for blur trained using a single series of successively blurred 100x100 images. The normalized amount parameter controls the degree of blur applied. The coordinate-based model closely approximates the blur it was trained on, and successfully generalizes to other images.

للاستشارات المنارة
www.manaraa.com

The degree of blur that can be successfully learned by the model is limited primarily by the neighborhood window size. A larger neighborhood window allows for a greater degree of blur, but does not generalize as well without additional training data. It should be noted that the coordinate-based model results are not quite as heavily blurred as their corresponding "Expected" images. The image blur model exhibits the same amount compression effect that is observed for image rotation in Section 3.4.1, and is most likely due to the same causes.

3.4.3 Sound Volume

In this experiment, the goal is to learn a volume transformation on sounds and transfer it cross-domain to images. The training data consists of the starting sound A_0 (the spoken word "banana") and 10 copies with the volume successively decreased using the Audacity audio tool $(A_1, A_2, ..., A_{10})$ so that $A = (A_0, A_1, A_2, ..., A_{10})$. As was the case for the low-pass filter, A is converted into Fourier space using the Fast Fourier Transform before training. Percent change is used as the training output, and the coordinate-based training set is constructed as

$$W = \left\{ (\frac{a}{10}, \vec{x}) \rightarrow \frac{A_{(i+a)_{\bar{x}}} - A_{i_{\bar{x}}}}{A_{i_{\bar{x}}}} \right\}_{a<11, i, \bar{x}}$$

with the special case that the training output is 0 where $A_{i_{\bar{x}}}$ is 0. Transfer from sound to image is accomplished using the same procedure and translation functions F and F' as were used for the low-pass filter in Section 3.3.3.

Table 3.8 describes the model parameters used, and Figure 3.10 shows that the decreasing sound volume transformation, when transferred, results in decreasing image brightness (which is an intuitively satisfying result). The degree of transformation is appropriately scalable by amount. The resulting image becomes too dark to see before the full amount is used, which can be explained by the difference in range between human hearing and vision. Although the training set has audible differences in volume for all of the example sounds, these differences fall below the range of human vision (when transferred to images) long before the sounds become inaudible.

المنارة للاستشارات
www.manaraa.com

Table 3.8: Model parameters for sound volume experiment

Parameter	**Value**
Neighborhood Window	None
Training Output	Percent Change
Amount	Yes
Max Amount	m
Additive	No
Activation Function	Linear Piecewise Symmetric

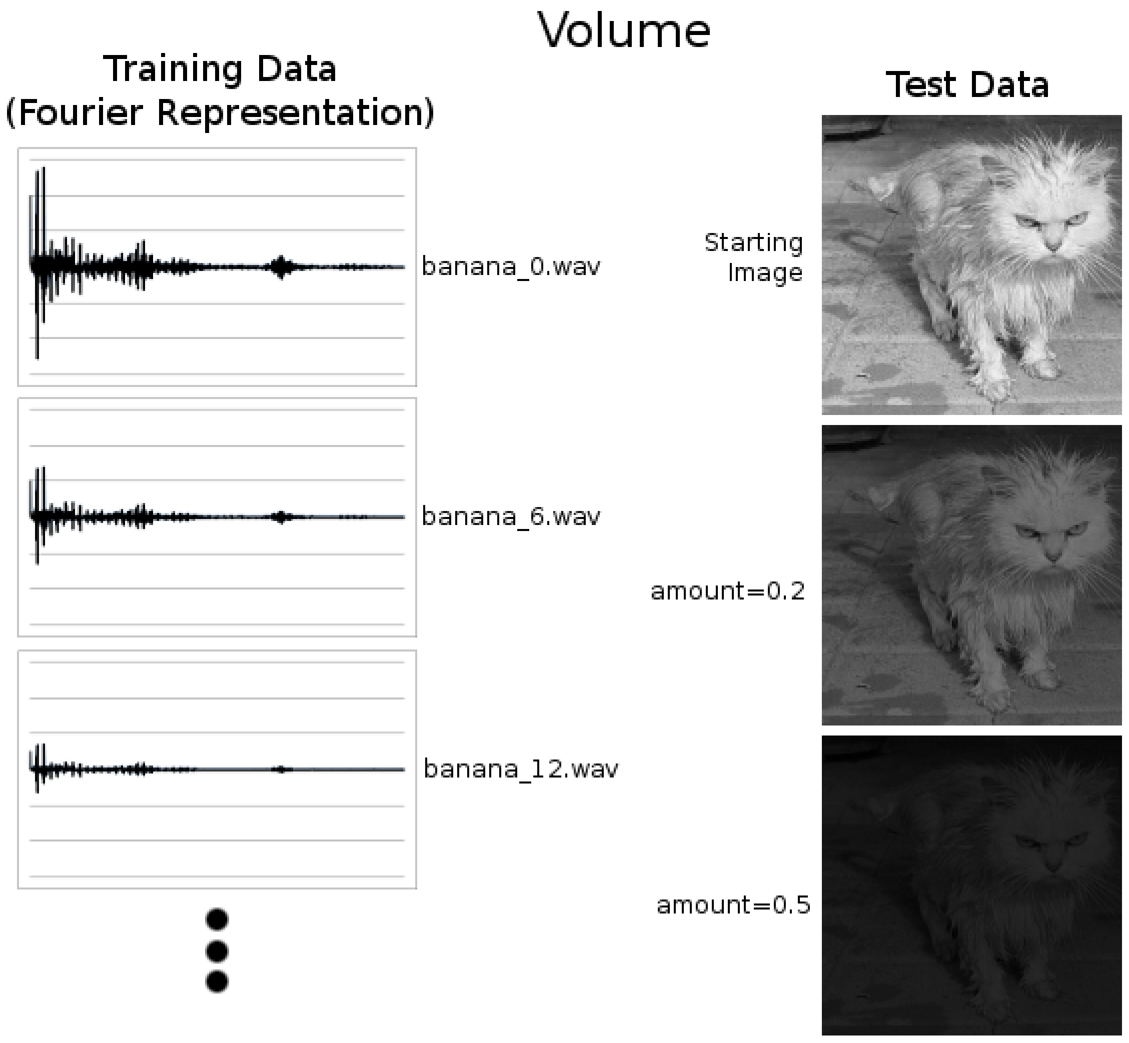


Figure 3.10: A transformation model for decreasing sound volume is trained on a set of audio signals, then transferred to the image domain, resulting in decreasing brightness on images. The normalized amount parameter controls the degree of the transformation that is applied.

www.manaraa.com

Table 3.9: Model parameters for amount-based low-pass filter experiment

Parameter	**Value**
Neighborhood Window	None
Training Output	Percent Change
Amount	Yes
Max Amount	m
Additive	No
Activation Function	Linear Piecewise Symmetric

3.4.4 Low-pass Filter

This experiment repeats the previous low-pass filter transfer experiment in Section 3.3.3, but with the addition of an amount parameter to control the degree. The training data consists of the starting sound A_0 (the spoken word "cow") and 10 copies with the low-pass filter applied with μ ranging from 1000 (A_1) to 100 (A_{10}) in steps of 100 so that $A = (A_0, A_1, A_2, ..., A_{10})$. As was the case previously, A is converted into Fourier space using the Fast Fourier Transform. The training set is then constructed as

$$W = \left\{ (\frac{a}{10}, \bar{x}) \rightarrow \frac{A_{(i+a)_{\bar{x}}} - A_{i_{\bar{x}}}}{A_{i_{\bar{x}}}} \right\}_{a<11,i,\bar{x}}$$

with the special case that the training output is 0 where $A_{i_{\bar{x}}}$ is 0. Transfer is accomplished in exactly the same manner as is discussed in Section 3.3.3. Table 3.9 lists the model parameters, and Figure 3.11 demonstrates that the model trained on audio is successfully transferred to the image domain. The model is able to control the degree of resulting image blur with the amount parameter.

3.5 Successive Application

Although the transformation model includes an amount parameter, it is likely that at some point it will be useful to apply the transformation model for larger amounts than the range it was trained on. As discussed earlier, this can be done by successively applying the model to

المنارة للاستشارات

www.manaraa.com

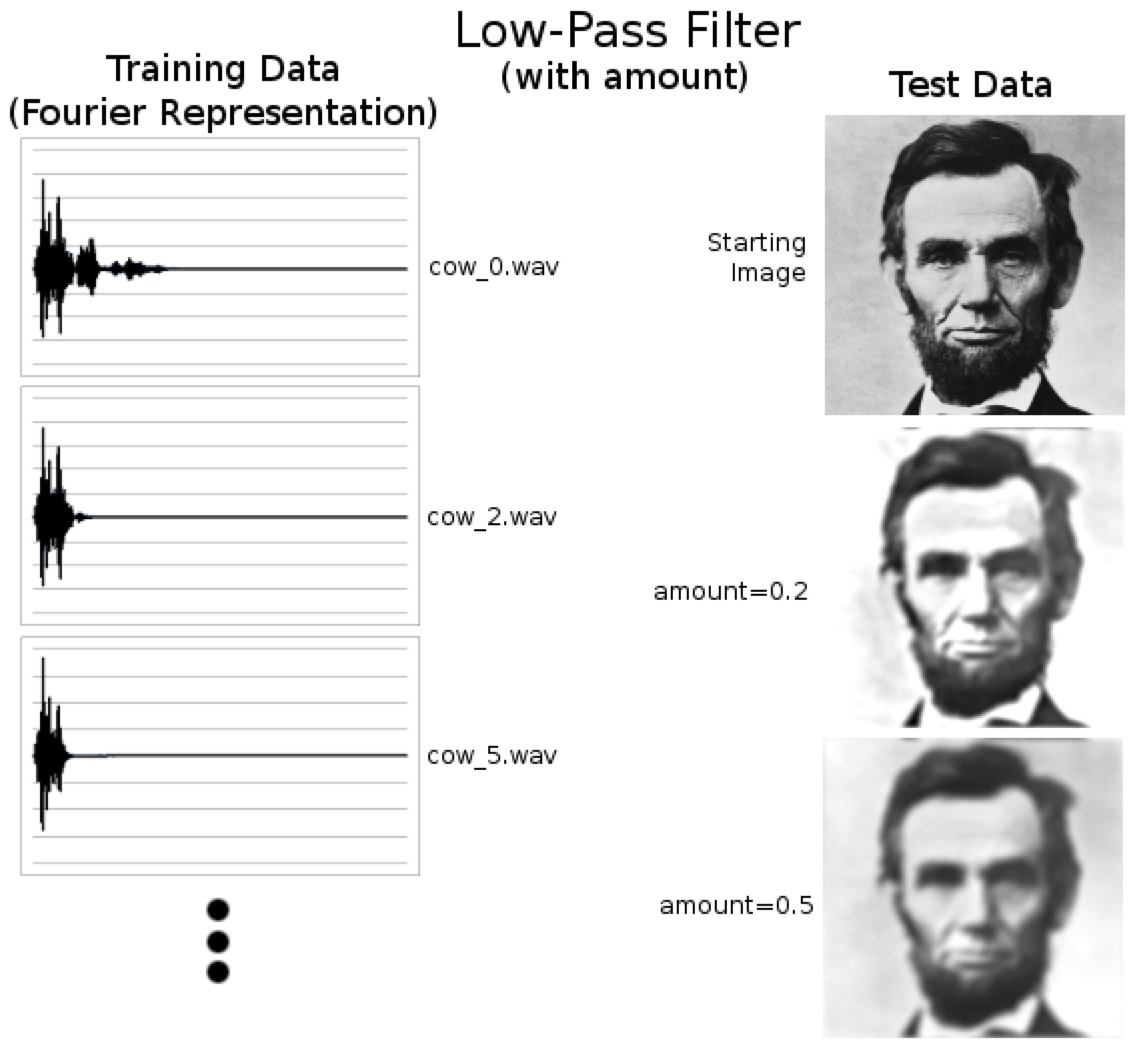


Figure 3.11: An example demonstrating learning transfer between sound and image domains, including an amount parameter. A low-pass filter is trained on a sound file with varying degrees of low-pass filter applied to it, and the resulting model is used to blur images to varying degrees.

www.manaraa.com

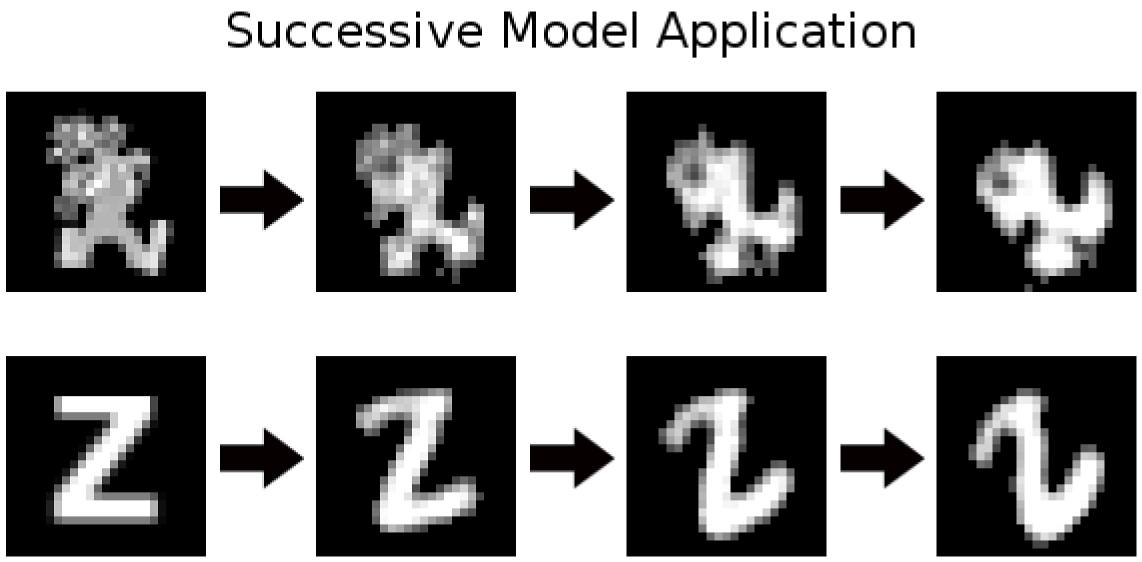


Figure 3.12: A transformation model can be repeatedly applied in order to apply a transformation for amounts larger than the model was trained on. This example uses a simple thresholding technique to reduce noise between applications.

its own output until the desired amount of transformation is achieved. Figure 3.12 demonstrates some results for image rotation using this technique and the image rotation model from Section 3.4.1 trained on multiple letters.

Between each successive application of the model, a simple thresholding technique is used that sets near-black pixels to pure black. For this experiment, this threshold was set to 0.25 on a normalized scale where 0 is black and 1 is white. This helps prevent minor noise in the output from growing into larger errors during successive model applications. There is still some degree of degeneration in the output: the shape of the original image is distorted, and in the case of the grayscale image, gray levels are lost. The latter problem could be resolved by using more advanced noise correction techniques between applications or training on grayscale images as well as black-and-white ones. Although there is some degree of degeneration in the output, the results show that if noise errors can be dealt with at each step, successively applying the model is a viable option for reaching transformation amounts

المنارة للاستشارات
www.manaraa.com

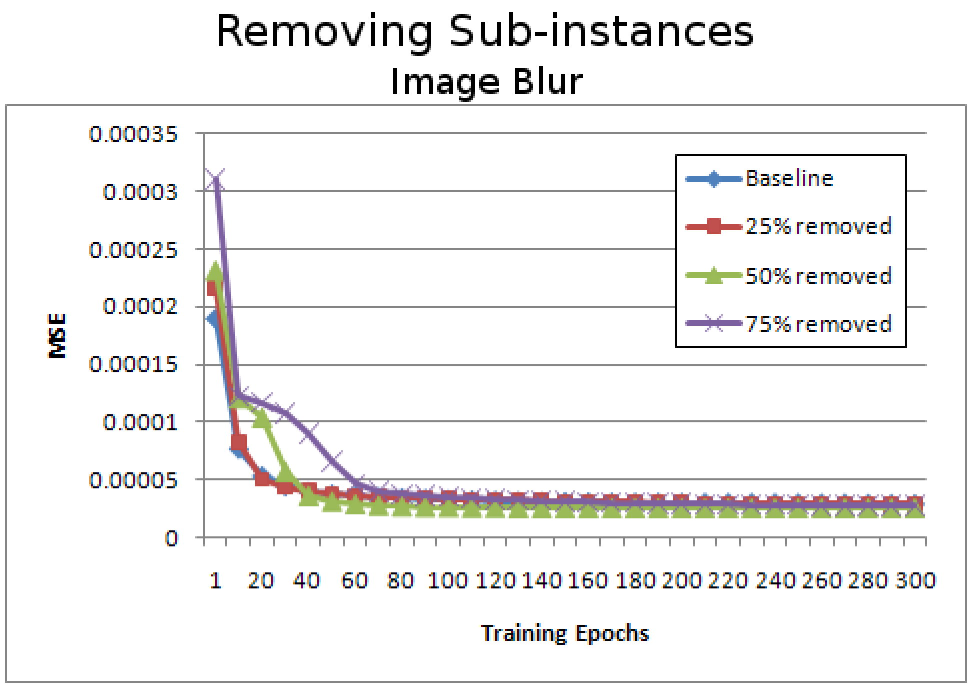
beyond the model's direct range. More refined noise reduction techniques are outside the scope of this thesis.

3.6 Removing Sub-instances

Coordinate-based transformation learning models are often resilient to missing data at the sub-instance level. Although this property has not yet been fully explored, some preliminary results are demonstrated in Figure 3.13. The image blur experiment from Section 3.4.2 is repeated, with elements randomly removed from the training set in various amounts to simulate missing data at the sub-instance level. The full training set from the previous image blur experiment is used as a baseline for comparison.

The results show that image blur, at least, is robust to missing data at the sub-instance level. This indicates that transformation learning is likely to succeed even on noisy or incomplete data.

المنارة للاستشارات
www.manaraa.com



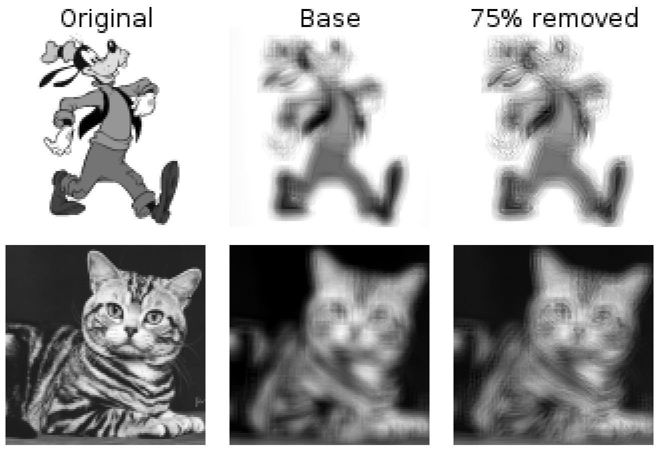


Figure 3.13: An example demonstrating the resilience of coordinate-based models to random removal of training data. A baseline model is trained as in Section 3.4.2, and other models are trained with 25%, 50%, and 75% of the baseline training set removed. The graph illustrates the comparative model accuracies over time during training, and the images compare the baseline and the "75% removed" models. Although training requires more epochs to converge when data is removed, the effect on the learned transformation model is relatively small.

www.manaraa.com

4 CONCLUSION

Transformation learning is an approach to modeling transformations in high-dimensional data so that they are well suited for learning transfer. Manifold learning allows transformations to be discovered in data even when they are not explicitly specified. Once a transformation is discovered, it can be modeled with a machine learning algorithm such as a multilayer perceptron.

By taking advantage of the coordinate systems that are often present, explicitly or implicitly, in high-dimensional data, it is possible to train transformation models with a high degree of generalizability. These coordinate systems allow the data to be broken down into sub-instances that are ordered in terms of the coordinate system. This ordering also provides for the ability to define local neighborhoods, which can be used to help model local effects in transformations.

Because these models are trained on focused sub-instances of the original data, there are fewer input and output parameters. Consequently, such models are less prone to overfit than models generated using the naive approach. When using the coordinate-based approach, transformation models can often be adequately approximated using very little data; sometimes, a single good example is enough for a reasonable model. This is much more difficult to do with the naive approach, which is subject to all of the problems associated with high-dimensional data.

Although certain parameters need to be set, transformation learning is largely capable of discovering and modeling transformations automatically. This ability makes transformation learning likely to be useful for the model of insight proposed by Ventura [23].

المنارة للاستشارات
www.manaraa.com

The coordinate system allows models to be specified both in terms of the coordinates themselves as well as local neighborhood windows based on those coordinates. Some transformations, such as a low-pass filter, are dependent only on the coordinate values. Others, such as image blur or edge detection, are only dependent on the neighborhood window. Some, such as image rotation, use information from both. Between these options, a large number of transformations can be effectively modeled using transformation learning.

An important direction for future work will be to reduce the number of assumptions and the amount of human input necessary for transformation learning to be successful. Transformation learning assumes that the data actually represents a clearly defined transformation, and it would be desirable to have an objective way to determine if this is true for a given dataset. When creating a transformation model, there is a fair amount of experimentation and tweaking that needs to be done in order to discover the best model parameters. If optimal parameters could be automatically discovered, the amount of experimentation needed would be greatly reduced. Also, when attempting to actually transfer between domains, it is necessary to manually provide a feature space translation mechanism. Being able to do this automatically would be an important step towards making transformation learning a more automatic process.

Transformation learning assumes that the data has a coordinate system, and that its nature and intrinsic dimensionality is known. This thesis focuses on data with clearly defined coordinate systems, such as sound and image data. It may be possible to find other kinds of data which contain an implicit coordinate system that can be discovered, perhaps using manifold learning, even if it is not known beforehand. In other words, there may be categories of data for which a coordinate system could be created. If this is the case, then the ability to create coordinate systems, in addition to using existing ones, would allow transformation learning to be useful for a wider variety of data.

Another important point is extending transformation learning to multiple dimensions. Data may contain a transformation that works in multiple dimensions (such as image trans-

المنارة للاستشارات
www.manaraa.com

lation, which exists in X and Y) or multiple unrelated one-dimensional transformations (for example, a dataset containing the letter 'A' in various states of rotation and scaling). The ability to handle both cases would make transformation learning much more powerful.

The current method for determining amount assumes that all of the data instances, once ordered according to the transformation, are fairly evenly spaced on the manifold. This does not account for the possibility of data that is highly skewed or contains large holes, and it would be worthwhile to extend transformation learning to handle these cases. One possible approach is to take advantage of the discovered manifold. Some manifold learning techniques are capable of accurately representing such skew and gaps, and the coordinates discovered using manifold learning could be used to inform the values of the amount parameter when constructing the training dataset. In this situation, when there is a gap in the data, there would be a corresponding gap in the amount parameter used for training. This might help the model train more accurately when presented with data containing skew or large holes.

Another potential application of transformation learning is in invariance learning. Instead of learning transformations for transfer directly *between* datasets, we could use this method to discover and model transformations *within* a dataset. These transformations represent the invariances in the dataset. This allows for a generalization and expansion of the work done by Lando & Edelman and Thrun described in Section 1.

www.manaraa.com

References

[1] J. Baxter, "Theoretical models of learning to learn," in *Learning to learn.* Norwell, MA, USA: Kluwer Academic Publishers, 1998, pp. 71–94.

[2] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

[3] L. Cayton, "Algorithms for manifold learning," University of California, San Diego, Tech. Rep. CS2008-0923, 2005.

[4] J. Davis and P. Domingos, "Deep transfer via second-order markov logic," in *Proceedings of the 26th Annual International Conference on Machine Learning.* New York, NY, USA: ACM, 2009, pp. 217–224.

[5] M. S. Gashler, D. Ventura, and T. Martinez, "Iterative non-linear dimensionality reduction with manifold sculpting," in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008, pp. 513–520.

[6] D. Gentner, "Structure-mapping: A theoretical framework for analogy," *Cognitive Science*, vol. 7, no. 2, pp. 155–170, 1983.

[7] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition).* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[8] N. Gorski and J. Laird, "Experiments in transfer across multiple learning mechanisms," in *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

[9] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques.* New York, NY, USA: ACM Press, 2001, pp. 327–340.

[10] N. Intrator and S. Edelman, "Making a low-dimensional representation suitable for diverse tasks," in *Learning to learn.* Norwell, MA, USA: Kluwer Academic Publishers, 1998, pp. 135–157.

للاستشارات المنارة

www.manaraa.com

[11] M. Lando and S. Edelman, “Receptive field spaces and class-based generalization from a single view in face recognition,” *Network: Computation in Neural Systems*, vol. 6, pp. 551–576, 1995.

[12] E. Levina and P. J. Bickel, “Maximum likelihood estimation of intrinsic dimension,” in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 777–784.

[13] H. P. Morevec, “Towards automatic visual obstacle avoidance,” in *Proceedings of the 5th International Joint Conference on Artificial Intelligence.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1977, pp. 584–584.

[14] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 99, no. PrePrints, 2009.

[15] L. Y. Pratt, “A survey of transfer between connectionist networks,” *Connection Science*, vol. 8, no. 2, pp. 163–184, 1996.

[16] L. Reder and R. L. Klatzky, “The effect of context on training: Is learning situated?” Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep., 1994.

[17] L. K. Saul and S. T. Roweis, “Think globally, fit locally: unsupervised learning of low dimensional manifolds,” *Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2003.

[18] S. M. Smith and J. M. Brady, “Susan—a new approach to low level image processing,” *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[19] J. B. Tenenbaum, V. de Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, no. 5500, pp. 2319–2323, December 2000.

[20] S. Thrun, “Lifelong learning algorithms,” in *Learning to learn.* Norwell, MA, USA: Kluwer Academic Publishers, 1998, pp. 181–209.

[21] S. Thrun and T. M. Mitchell, “Learning one more thing,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence.* Morgan Kaufmann Publishers Inc., 1995, pp. 1217–1223.

[22] L. Torrey and J. Shavlik, “Transfer learning,” in *Handbook of Research on Machine Learning Applications*, E. Soria, J. Martin, R. Magdalena, M. Martinez, and A. Serrano, Eds. IGI Global, 2009.

المنارة للاستشارات
www.manaraa.com

[23] D. Ventura, "Sub-symbolic re-representation to facilitate learning transfer," in *Creative Intelligent Systems, AAAI 2008 Spring Symposium Technical Report SS-08-03*, March 2008, pp. 128–134.

المنارة للاستشارات
www.manaraa.com