Using the Representation in a Neural Network's Hidden Layer for Task-Specific Focus of Attention

Shumeet Baluja
tani@csl.sony.co.jp
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Dean A. Pomerleau
pomerleau @ cs .emu .edu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In many real-world tasks, the ability to focus attention on the important features of the input is crucial for good performance. In this paper a mechanism for achieving task-specific focus of attention is presented. A saliency map, which is based upon a computed expectation of the contents of the inputs at the next time step, indicates which regions of the input retina are important for performing the task. The saliency map can be used to accentuate the features which are important, and de-emphasize those which are not. The performance of this method is demonstrated on a real-world robotics task: autonomous road following. The applicability of this method is also demonstrated in a non-visual domain. Architectural and algorithmic details are provided, as well as empirical results.

1. Introduction

Many real world tasks have the property that only a small fraction of the available input is important at any particular time. On some tasks this extra input can easily be ignored. But often the similarity between the important input features and the irrelevant features is great enough to interfere with task performance. Two examples of this phenomena are the famous "cocktail party effect", otherwise known as speech recognition in a noisy environment, and image processing of a cluttered scene. In both cases, the extraneous information in the input can be easily confused with the important features, making the task much more difficult.

In this paper, we will use the representation of a neural network's hidden layer, trained to perform a time sequential task, to make predictions of what the next inputs will be. These predictions can be used as a pre-processor for the next inputs; they can provide a mechanism to focus the network's attention on the important features. In the next section, focus of attention is described in greater detail, and the notion of a saliency map is introduced. The cognitive foundations of focus of attention are also briefly explored. In section 3, the task of interest, autonomous road following, is presented, as well as results of using selective attention to improve performance on this task. In section 4, a synthetic problem is described; it contains many of the same difficulties as the road following task, yet lends itself to easier interpretation and analysis. The problem also serves to exhibit the benefits of a saliency map in a non-vision oriented task. Finally, in sections 5 & 6, conclusions and suggestions for future research are presented.

A previous paper [Baluja & Pomerleau, 1994] presented preliminary results of using a task-specific saliency map on two simulated vision-based tasks. The findings of that paper are summarized in section 2.1; the results of that paper are expanded upon in section 3, to make them applicable to real-world tasks. In this paper, a method used to create an adequate number of training examples for training the saliency map is presented. These issues were not present in the tasks explored in [Baluja & Pomerleau, 1994] as the examples were created artificially and were inexpensive to generate.

2. Focus of Attention: Background and Implementation

Focus of attention has been studied in a variety of contexts. One of the largest branches of study has examined attention in static images. For example, [Triesman & Gelade, 1980] [Hulbert & Poggio, 1985] describe a study in which a subject recognizes the letter "S" mixed in a field "X"'s and "T"'s of various colors. The "S" *pops out* to the subject, suggesting a preattentive and parallel processing of the image. If an object must be distinguished by "conjunctive" features such as color and shape, the search seems to be performed in a more serial fashion [Hulbert & Poggio, 1985]. The most commonly accepted analogy for this is termed the "spotlight" hypothesis. The spotlight moves from one location to another, the area in which it focuses are operatively defined to be the areas in which an improved performance is found in the tasks of stimulus detection, identification, localization, or simple and choice response times [Umilta, 1988].

Computational models of the spotlight mechanism have been proposed in the context of artificial neural networks [Mozer, 1988][Koch & Ullman, 1985]. Mozer makes the distinction between "data-driven" and "conceptually-driven" guidance of the spotlight. A simple case of "Data-Driven" guidance is that the spotlight should be drawn to objects, but not to empty spaces in the visual field. A "Conceptually-Driven" spotlight is controlled directly by a "higher level of cognition" (goal driven). This is necessary when reading, in which text must be scanned from left to right.

When longer time intervals are introduced, the process of

1. It should be noted that a moving the "spotlight of attention" does not necessarily entail eye movements. [Umilta, 1988] discusses the relationship between eye movements and focus of attention.

focusing attention becomes more challenging; objects can move and change. Nonetheless, people can routinely solve the problem of focusing, and maintaining, attention on moving and changing objects. This ability to do this with "odd-man-out"[2] features has been has been called *indexing* [Ullman, 1985]. The ability to index is a prerequisite for visual motor coordination and object description [Trick & Pylyshyn, 1991]. Nonetheless, not all items can be indexed in this way. [Allport, 1989] suggests a more complicated procedure, termed "selection-for-action" which introduces the task-specific nature of focusing. Information about irrelevant features or objects must be filter out, to avoid crosstalk and confusion with respect to the feature or objects of interest. This model relates to the model presented in this paper.

In our attempts to design a mechanism to focus attention, the goal was to create a conceptually driven *expectation* which can maintain attention on moving objects which may change shape, orientation and position. Further, as different tasks will require analyzing different portions of the scene, the focus of attention must be task-specific (selection-for-action). The focus of attention must designate as important only the portions of the scene which are necessary for completing the task. The next section describes the mechanism to implement this type of attention focusing.

2.1. Creating a Conceptually Driven Saliency Map

Saliency maps have been used in the field of computer vision to direct processing to only the relevant portions of the scene. However, in many studies, saliency maps have been constructed in a bottom-up manner [Clark and Ferrier, 1992]. A very cursory summary of the bottom-up approach is that multiple different feature detectors are placed around the input image. Each type of feature detector may contain a weight associated with it, to signify the relative importance of the particular feature. The region of the image which contains the highest weighted sum of the detected features is the portion of the scene which is focused upon. The approach taken in this paper is very different. The features and their weightings are developed simultaneously with the saliency map to solve the particular task. In this proposed method, the expectation of where the features will be in the *next* frame plays a key role in determining which portions of the visual scene are *going to be* focused upon. This will be explained in this section.

The first step in creating a conceptually based saliency map is determining which portions of the image are important. The creation of a saliency map is based upon a very basic analysis of the neural network. The underlying premise is that if a strictly layered (connections are only between adjacent layers) feed-forward neural network can solve a given task, the activations of the hidden layer contain, in some form, the important information for this task from the input layer. One method of finding out what information is contained within the hidden layer is to attempt to reconstruct the original input image, based solely upon the representation developed in the hidden layer. This method of reconstruction is closely related to *Input Reconstruction Reliability Estimation (IRRE)* [Pomerleau, 1993]. The reliability estimation in IRRE is made by reconstructing the input image by using linear transformations of the activations in the hidden layer, and comparing the resulting reconstruction with the actual image. The greater the similarity between the actual input image and the reconstructed input image, the more the internal representation has captured the important input features, and therefore the more reliable the network's response. Figure 1 provides a schematic of IRRE.

2. These are features which are distinguished by attributes such as color, or by motion in an otherwise stationary background.

This method is related to auto-encoding networks. In these networks, the output is trained to reproduce the input layer [Cottrell & Munro, 1988]. The hidden layer, which is usually used as a bottleneck, captures important features for reconstructing the input. The difference between these networks and the ones employed in this study is that the networks used here were not trained to reproduce the input layer accurately; they were trained to perform well on a specific task. All of the representational power in the hidden units is devoted to solving the task only. The portions of the input which can be reconstructed accurately are the portions of the input which the hidden unit activations have encoded to solve the task. If the requirement of task-specificity did not exist, auto-encoder networks, or methods such as principal components analysis could capture many features of the input. However, the features found by these methods are important for reconstructing the image, not for solving the particular task. As only a fraction of the features found may be important for the task, it is difficult to focus attention on task-specific features based upon these methods.

Although IRRE provides a method to determine which portions of the input the network finds important, a notion of time is necessary to focus attention in *future* frames. Instead of attempting to reconstruct the current input, the network is trained to predict the *next* input (in Figure 1, this corresponds to changing the subscript *t* to t+l, in the reconstructed inputs). The next input is predicted based upon the important task-specific features in the current image.

The prediction can be trained in a supervised manner, by training the network to predict the next set of inputs in the time sequence. The training example (the next inputs) may contain noise or extraneous features. However, since the hidden units only encode information to solve the task, the network will be unable to construct the noise in its prediction. More details on this idea, and methods to use the expectation of the next inputs, are described in the next sections.

2.2. Differences in Expectation and Realization

To this point, a method to create an expectation of what the next inputs will be has been described. There are two fundamentally different ways in which to interpret the difference between the expected next input and the actual next

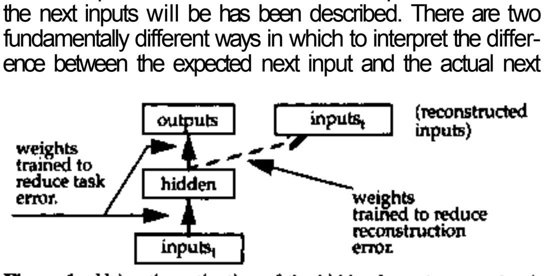

outputs
inputs$_t$
(reconstructed inputs)
weights trained to reduce task error.
hidden
weights trained to reduce reconstruction error.
inputs$_t$

Figure 1: Using the activation of the hidden layer to reconstruct the input. The weights between the input and hidden layers are trained to only reduce task error, *not* reconstruction error. Extra hidden layers for reconstruction can be used.

input. The first interpretation is that the difference between the expected and the actual input is the point of interest because it is a region which was *not* expected. This has applications in anomaly detection, or in the analysis of visual scenes in which the object of interest is moving across a stationary background.

In the second interpretation, the difference between the expected and actual inputs is considered noise. This interpretation is used throughout the rest of the paper. Processing should be de-emphasized from the regions in which the difference is large. This makes the assumption that there is enough information in the previous inputs to specify what the important portions of the next image will be. As will be shown in the tasks described in sections 3 and 4, this method has the ability to remove spurious features and noise. It is interesting to note that in this interpretation, it is important that the prediction of the future state *not* be too accurate. If the prediction matched the next image exactly, the noise in the next image would also be reconstructed. Although the network is trained to predict future inputs with example training images which may contain noise, the network is *not* able to reproduce the noise due to the hidden layer's limited capacity, the task-specific hidden units, and the task-specific nature of training. The implementation of the saliency map is described in the next section.

2.3. Using Expectation to Filter Noise

In this study, the saliency map was used as follows: the difference between the expectation of input image$_{t+j}$ (derived from input image$_t$) and the actual input image$_{t+1}$ was calculated. This difference image was scaled to the range of 0.0 to 1.0. The smaller the difference, the closer the value to 1.0. Each pixel of the difference image was then passed through a sigmoid; alternatively, a hard-step function could have been used. This results in the saliency map. This map indicates the portions of the input to which the network should be paying attention. In order to emphasize these regions for the input, the saliency map is multiplied, pixel by pixel, with input image$_{t+]}$. The result after multiplication was used as the input into the network. This has the effect of lowering the activation of the portions of the input which do not match the expectation. The portions of the input which match the expectation are left unaltered. Examples of this filtering, for the task of autonomous road following, are shown in section 3.

Training the neural network with a saliency map may require more pattern presentations than training a network which does not employ one. With feedback to the inputs, the system becomes dynamic. As the training for the task improves, the saliency map becomes more refined, and more of the correct information is filtered out of the images. The images input to the network later in the training process possess different qualities than the those input earlier in training. Because the network is trained to reduce the task error, the hidden representation changes to adapt to the new images. This causes changes in the prediction of the next inputs, and the cycle continues. The cycles ends when either the system reaches a stable state or training is stopped. In practice, the system can be trained by using the standard backpropagation algorithm, with small learning rates, albeit with longer training times.

One of the problems encountered in focusing attention using this method, is the necessity to determine the features which are important for solving the task before the task is solved. The difficulty is that since the important features are task-specific, and are developed from the network's hidden layer, the task must first be solved. This "chicken-and-egg" problem is avoided in many situations because some of the images used for training may not contain noise. Therefore, a small amount of learning is able to proceed without explicit focus of attention. Once a few of the important features are determined, the system can, in many cases, bootstrap itself. In the implementations described here, the system is trained to build the saliency map and solve the task simultaneously.

2.4. Relations to Other Recurrent Neural Networks

The use of the feedback connections proposed are related to other recurrent neural networks [Jordan, 1989][Stornetta, 1988]. At time t, the Jordan network uses feedback from the output units of time *t-l,* combined with feedback from the context units at time *t-l,* to create new context units. These units are used as additional inputs in the current time-step. The Stornetta architecture uses the context units as a preprocessor of the input units [Hertz *et. al,* 1993]. The context units are arranged with one-to-one connections with the inputs, and have feedback connections from themselves, which carry activation from the previous time step.

There are several important distinctions between the Jordan and Stornetta architectures and the one used in this paper. The first is that the Stornetta architecture uses context units which are able to form arbitrary representations. In the architecture presented here, the feedback is from units which have a very defined task. The feedback is the prediction of the next inputs. Also, the feedback is multiplicative, and acts like a filter, unlike the architecture in the Jordan networks.

Another large difference in these networks is that the feedback units are explicitly trained in a supervised manner. In architectures like the Stornetta, the context units are trained in a manner similar to the training of the hidden units. The representations which are formed are created to reduce the error in a subsequent output layer.

Finally, the last difference is in the problems which these architectures are designed to address. Although most of the recurrent networks which have been explored in the literature have attempted to address the problem of sequence recognition and reproduction, this architecture is not suited to these tasks as the feedback is restricted to be the prediction of the next inputs.

3. Image Based Autonomous Road Following

One of the principle motivations for creating an algorithm which can focus attention is to perform visual processing in cluttered scenes. A real-world application which requires such attention focusing is autonomous road following.

In the domain of autonomous road following, the goal is to control a robot vehicle by analyzing the scene of the road ahead, and choosing a direction to travel based on the location of important features like lane markings and road edges. This is a difficult task since the scene ahead is often cluttered with extraneous features such as other vehicle, pedestrians, trees, crosswalks, road signs and other objects that can appear on or around a roadway. For the general task of autonomous navigation, these extra features are extremely important; however, for the restricted task of road following, these features are distractions. While we have had significant success on the road following task using simple feed-forward neural networks to map images of the road into steering commands [Pomerleau, 1993], these simple methods fail when presented with cluttered environments like those encountered when driving in heavy traffic, or on city streets.

3.1. The ALVINN Road Following System

ALVINN is an artificial neural network based perception system which learns to control CMU's NAVLAB vehicles by watching a person drive. ALVINN's architecture consists of a single hidden layer backpropagation network. The input layer of the network is a 30x32 unit two dimensional "retina" which receives input from the vehicle's video camera. The correct steering direction is determined from the activation of 30 output units. The output units attempt to create a gaussian centered around the correct steering direction. If the Gaussian is centered around unit 1, this indicates the vehicle should make a sharp left, if the center is around unit 30, the vehicle should make a sharp right, etc. To teach the network to steer, ALVINN is shown video images from an onboard camera as a person drives. For each image, it is trained to output the steering direction in which the person is steering.

Recently, there has been an emphasis on using the ALVINN system as a driver warning device. In one of the proposed uses of this system, the model will warn drivers if they begin to drift over lane markings (indicating that they may be entering a lane with on-coming traffic, or leaving the road, etc.). The system must be robust with respect to other road features, such as road and off-road boundaries, cars, or other lane markings. Experiments for this task are described in the next section.

3.2. Eliminating Noise Using a Saliency Map

The purpose of using a saliency map within this domain is to eliminate features of the road which the neural network may mistake as lane markers, and therefore output an incorrect steering direction. Training the network to solve the task by focusing on the important regions of the scene is described below.

Approximately 1200 images were gathered from a camera mounted on the left side of a car, pointed downwards and slightly ahead of the vehicle. The car was driven through city and residential neighborhoods around Pittsburgh, PA. The images were subsampled to a 30x32 pixel representation. In each of these images, a single X location of the lane marker was hand marked around the 20th row. (The total interactive time was -25 minutes). The task is to produce a gaussian of activation in the outputs, centered around the X location of the lane marker in the 20th row of the image. Sample images and target outputs are shown in Figure 2.

In training the network, there are several problems which must be addressed. The first is that there is only a limited amount of training data. Further, assuming that the driver has directed the car well, the center line has probably stayed within a small region of the input image. Therefore, the network has not been trained to recognize lane-markers outside the middle regions of the image. Additionally, because the prediction task attempts to forecast future inputs, it is important not to bias the network to memorize the image transitions in the training set. For example, had the driver chosen a slightly different action, the location of the important features in the next set of inputs may have been different.

In order to alleviate these problems with the training set, the following modifications were made: In training, extra images were created by translating the original images to the left or right by up to 5 columns. The portions of the image which were not specified after the translation were filled in with the last previous real pixel value in the current row. The output was also translated either to the left or right by the same amount as the image. This translation yields usable images because the camera is pointed downwards. If the camera had been pointed more ahead of the vehicle, more sophisticated rotations would have been required to maintain the correct perspective, as were used in [Pomerleau, 1991].

The sequential nature of focusing attention dictated that these images could not simply be added to the training set. For example, an image at time r, which is translated 3 steps to the left, should not be followed by an image at time *t+J* which is translated 3 steps to the right. If it were, the important features would jump a large distance, and this is unlikely to happen in practice. To avoid this problem, the expanded training set is used in the following manner: the image at time step *t+J* is chosen at a random translation which differs by, at most, ±1 from the image at time step *t*. This ensures that large jumps of the important features are not present between consecutive time steps. As the network is trained through many passes through the training set, images are seen with different translations.

In addition to using the translated images as described above, to ensure that the network learns many of the possible transitions from any image, the errors from predicting 25 potential next input images are used. These 25 "next input" images are created as follows: the images at time steps *t-2, t-1, t, t+J & t+2,* are translated by 0,1 & 2 columns to the left and right. The error between the predicted next state and these 25 images are used for the backpropagation algorithm. This training is done because any of the 25 images are reasonable expectations for the next input, based upon the current inputs. In many tasks, using previous time steps, or time steps beyond *t+J* may not work (see section 4). Nonetheless, for this domain, the important features, such as the lane markings, will remain relatively consistent for short periods of time, and can be used in training future predictions.

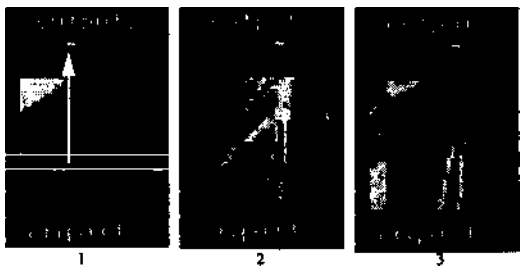
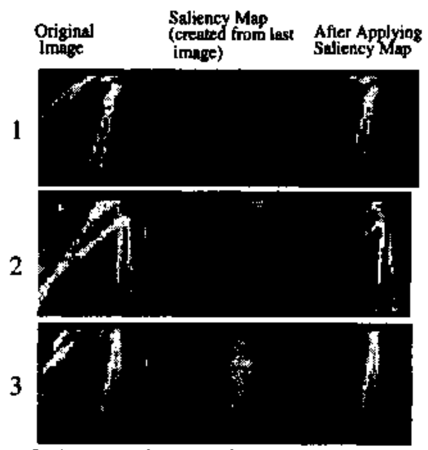


Figure 2: Three sample input images and *target* outputs. Image 1 shows the region from which the lane marker was hand selected. In image 2 there is an extra lane marking. In image 3 the lanemarker is not completely visible.



Original Image
Saliency Map (created from last image)
After Applying Saliency Map
1
2
3

Figure 3: Images before and after applying the saliency map. In images 1 the edge of the road is bright enough to cause distractions. In 2, the two lane markings may confuse the lane tracker, and cause it oscillate between the lane markings. In 3, a passing car is removed from the image

After training in the manner described above, the results of this experiment were very promising. The lane tracker was able to remove various types of distracting noise from the images (See Figure 3). The performance of the lane tracker with the saliency map revealed a 15-20% improvement over the lane-tracker without the saliency map. The improvement was not greater because many of the image in the test set do not contain noise; with these images, a standard ANN can be used to accurately estimate the lanemarker position. Nonetheless, in order to maintain a user's trust in a driver warning system, it is crucial that false alarms are minimized. Further, since the system is designed to take control of the vehicle in hazardous situations, under no circumstances may the system be distracted by spurious lanemarking or similar appearing features. The saliency map has provided an effective mechanism to focus attention on the important portions of the scene.

This task is made easier because the relevant features are in approximately the same location in many images, and are very similar in shape. This is not, however, required for the algorithm to work well. Example tasks in which this was not the case can be found in [Baluja & Pomerleau, 1994] and in the next section.

4. Determining Markov Transitions in Noisy Environments

In this section, we describe a synthetic problem which contains many of the difficulties of the road following task, yet lends itself to easier interpretation and analysis. This problem also serves to demonstrate the saliency map's ability to work in non-visual domains.

In this experiment, there are 20 inputs and 2 outputs. Only 1 input is turned on (activation of+1) in each example. Output 1 (O_1) should be turned on if the input which is turned on is an input between 1 and 10 (I_1-I_{10}); output 2 should be turned off (activation of -1). If the input which is on is I_{11}-I_{20} the role of the output units should be reversed. The inputs turn on in a random order, which is determined at the beginning of the run; the order remains the same throughout the run. These Markov chains proceed 20 steps, in which each input is turned on exactly once. After the 20 steps, the cycle repeats. The transitions can span any size gap in the input layer. This is unlike the previous task, of road following, in which features did not make large spatial jumps between successive frames. Also, the previous task was not strictly Markovian, as many next images were equally possible from each image.

The task, as described above, can be solved easily by a standard neural network. In fact, the order of activation is irrelevant. The task becomes more difficult when random inputs can turn on in addition to the input which should be turned on. With the addition of noise, the order of activation becomes important. When there is more than one input turned on, O_1 should only be turned on if the underlying Markov process dictates that an input I_1-I_{10} should be on. It should not turn on when the Markov process dictates, for example, that I_{16} should currently be on, and random noise has turned on I_5. In the cases in which noise exists in the input units, it is necessary to be able to determine what the underlying Markov transitions are, in order to determine which activated input is noise, and which is not. It was found that using a network with two hidden layers yielded good performance on this task. As in the previous task, although the hidden layers are connected to the prediction layer, the errors on the prediction do not influence the training of connections from the hidden layer to the input layer. The outputs specifying the next set of inputs is directly analogous to the expectation outputs used in the previous application.

Several forms of noise were tested in this experiment. The first introduced randomly occurring noise into the input layer. The state of a randomly chosen input is flipped (i.e. if it was activated to +1, it is changed to -1, and vice-versa). With large amounts of training, networks both with and without a saliency map are able learn the appropriate Markov transition rules. In fact, a neural network is not necessary for learning the transition rules. A simple method to learn these rules is to count, for each activated input at time t, which inputs turn on in time step t+/. Even with a large amount of noise, the transition rules can quickly be found.

A much harder task is to use a second, independent, Markov process to turn on inputs in a different order. In this task, only one of the processes is of interest, and the other process is noise. For this problem, the simple counting method described above will not work. The motivation for using two processes, instead of simple randomly occurring noise, is that this method more closely relates to real-world tasks, in which distractions may be coherent, structured features or objects which persist through multiple time steps. Examples of these include multiple voices or conversations in the context of speech recognition or multiple lane markers in the context of road following.

4.1. Experiments - Results

The task described above with two Markov processes was conducted as follows: In each input presentation, two inputs were turned on. One corresponded to the actual process of interest, the other to the noise process. Approximately every 100 pattern presentations, each process was restarted randomly from a randomly chosen position in each sequence.

See Figure 4 for an example.

There are two measures of performance for the task described above, each are measured every 4000 pattern presentations. The first is to measure how well the transition rules of the real process were discovered. This is determined as follows: each input is individually turned on in separate presentations (total of 20). For each presentation, the unit with the highest activation in the expectation portion of the outputs is determined. If this corresponds to the next input which would be turned on if the sequence was being presented, the output is correct.

The second measure of performance is how well the networks perform on the main task (turning on either O1 or 02). This is determined as follows: the error on 500 pattern presentations is measured. The inputs include noise from the distracting process. To ensure that the network has not memorized how to de-emphasize the noise process when started in a particular position with respect to the real process, the noise and real process are restarted 10 times in random locations. The sum square errors, over all presentations, on the two output units are summed.

There are three training methods examined in this paper, these correspond to training the hidden units using the training signal from either the main task, or the expectation outputs, or both. The training method which corresponds to the method used with the lane-marking task, described in the last section, is "Main-Task Only". In the "Prediction Task Only", only the errors from the expectation are used to train the hidden units. The features developed for solving the prediction task must be used to reduce the main task error. In the "Both Tasks" training procedure, errors from the expectation outputs and the main task outputs are used.

All three of these methods are attempted with and without the saliency map. A typical run is shown in Figure 5. The large oscillations in performance are due to the significant noise in the training and testing. If the network recognizes the wrong features, and mistakes the noise process for the real process, errors increase dramatically. The results for all six of the training sessions are shown below, in Tables I & II. Runs are continued for 3xl06 pattern presentations. In order to judge whether the differences in the average performances are relevant, the significance for the differences in sum squared error are measured here. Each network was trained 12 times, with random initial starting weights. For these tests, a two sided Mann-Whitney test is used at the 95% significance level. This test is a non-parametric (the underlying distribution is not assumed to be normal) equivalent of the standard two-sample pooled /-test. The differences in the means, measured by both criteria, are significant between using a saliency map and not, in training with the "main task", and with "both tasks". The differences, measured by both criteria, are not significant with the networks trained with only the "prediction task". Networks trained only for prediction do not perform well on the main task. Networks trained to perform the main task perform significantly better. The networks trained to reduce the task error, and which use a saliency map, perform the best.

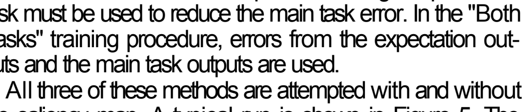


Figure 4: An input/output sequence for the task described. The real and noise process transitions are shown on the left. After 8 time steps, the processes are restarted in random positions, the real process at step 2, and the noise process at step 3. O_1 should be on if the real process has turned on I_1 or I_2, O_2 should be on if the real process has turned on I_3 or I_4.

Table I: Lowest Task Error (Sum Squared Error of Both Outputs, 500 patterns).

Saliency Map	Table Interpretation	How the Hidden Units were Trained		
		Main Task	Prediction Task	Both Tasks
With	Avg, Std. Dev	101.3, 82.2	651.0, 89.8	117.2, 35.3
	Min, Max	30.5, 298.9	504.2, 830.1	81.0, 216.4
Without	Avg. Std. Dev	380.1,34.3	603.1, 84.8	389.8, 25.5
	Min, Max	294.5, 424.8	464.1, 716.7	335.4, 424.1

Table II: Lowest Number of Incorrect Transition Rules (20 maximum)

Saliency Map	Table Interpretation	How the Hidden Units were Trained		
		Main Task	Prediction Task	Both Tasks
With	Avg, Std. Dev	0.6, 1.3	2.4, 2.7	0.0, 0.0
	Min, Max	0.0, 4.0	0.0, 7.0	0.0, 0.0
Without	Avg. Std. Dev	6.2, 0.24	4.4, 0.28	3.0, 0.25
	Min, Max	5.0, 7.0	2.0, 6.0	2.0, 4.0

The difference between training with "both tasks" and training only with the "main task", when using a saliency map, is significant when measured in task error, but not when measured in transition rule error. These results suggest that training the hidden units just on the main task provides enough information to do well on both the prediction and main tasks. Using some of the representational power in the hidden units for prediction (as with "Both Tasks" & "Prediction Task Only") hurts performance when measured on the task error. When the hidden units are trained to only do the prediction task, the results are significantly worse, measured

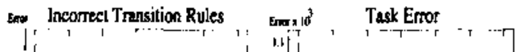


Figure 5: Typical run using only the main task to train hidden units. Extended run shown for $6x10^6$ pattern presentations. Runs shown with and without saliency map.

by both error metrics, than when the main task is used. This indicates that in this task, the errors from the main task must be used to improve performance on the main task *and on the prediction task.*

5. Conclusions

In this paper, a method for focusing attention on the portions of the input which are important for completing a particular task was presented. This method of using attention has been demonstrated in both visual and non-visual domains with promising results. The artificial neural network used in this study was able to avoid distractions by focusing attention on only the relevant portions of a scene. The feedback mechanism which is presented in this algorithm is more restricted in comparison to other recurrent neural network architectures. Nonetheless, it is enough to focus processing to relevant portions of the input. The resulting network can be trained by using the standard backpropagation learning algorithm. Extensions to the presented architecture can use methods of encoding task context (such as [Jordan, 1989]) in addition to the attention mechanisms.

Unlike many of the previous studies which use neural networks to predict future states, this work has presented an algorithm which relies on the *limited* accuracy of the neural network's future state prediction. The premise of this algorithm is that future event prediction cannot be perfect. In particular, the network cannot accurately predict the future states when the future states contain noise or spurious features. By analyzing the difference between what is expected in the next time step and what is actually present in the next time step, it is possible to determine which of the features in the input retina are noise, and which are important to completing the particular task.

In this paper, we have demonstrated the applicability of this algorithm on a real-world application, that of autonomous road following. The algorithm is able to avoid being misled by extra lane markings, and other features which have very similar appearances, which could cause the algorithm to steer the vehicle incorrectly. One of the future directions is to use this network in a driver run off-road warning and control system. Other future directions for study are presented below.

6. Future Directions

Relations to Kalman filtering and PCA analysis are currently being analyzed. In addition, alternative implementations of the saliency map are also being investigated. These include alternative methods to apply the information in the saliency map, and the use of additional previous inputs for more time-context.

An open question is from which hidden layer(s) should the expectation be constructed? Different hidden layers will contain information at different levels of transformation from the original inputs. Another direction for future research is using the saliency map as a tool for interacting with other knowledge-sources. The information in the saliency map can be useful for high-level attribute selection and weighting in other algorithms. Another form of interaction between the saliency map and external knowledge sources is using the knowledge sources to create or augment the saliency map directly. This interaction can provide "suggestions" to where the network needs to devote attention.

Acknowledgments

Thanks are extended to the anonymous referees for their very helpful and constructive comments.

Shumeet Baluja is supported by a National Science Foundation Graduate Fellowship. Support has come from "Perception for Outdoor Navigation" (contract number DACA76-89-C-0014, monitored by the US Army Topographic Engineering Center) and "Unmanned Ground Vehicle System" (contract number DAAE07-90-C-R059, monitored by TACOM). Support has also come from the National Highway Traffic Safety Administration under contract number DTNH22-93-C-07023. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the National Science Foundation, ARPA, or the U.S. Government.

References

Allport, A. (1989) "Visual Attention" *Foundations of Cognitive Science.* Posner, M. (ed.) MIT Press. Cambridge, Ma. 631-683

Baluja, S. & Pomerleau D.A.(1994), "Using a Saliency Map for Active Spatial Selective Attention:". *Advances in Neural Information Processing Systems* 7. G. Tesauro, D. Touretzky, J.Alspector.

Clark, J. & Ferrier, N. (1992) "Attentive Visual Servoing" In *Active Vision,* (ed) A. Blake & A.Yuille. MIT Press.

Cottrell, G.W. & Munro, P. (1988) "Principal Component Analysis of Images via back-propagation." *Proc Soc. of Photo-Optical Instr. Eng.,* Cambridge, MA.

Hertz, J., Krogh, A., Palmer, R. (1989) *Introduction to the Theory of Neural Computation.* Addison-Wesley.

Hulbert, A. & Poggio, T. (1985) "Spotlight on Attention". MIT AI Laboratory Memo. AI- 817.

Jordan, M.I. (1989). "Serial Order: A Parallel, Distributed Processing Approach." In *Advances in Connectionist Theory: Speech,* eds. J.L. Elman and D.E. Rumelhart. Hillsdale: Erlbaum.

Koch,C. & Ullman, S.(1985)"Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry'7/u/nan *Neurobiology* 4.

Mozer, M.C. (1988) A Connectionist Model of Selective Attention in Visual Perception. TR, University of Toronto, CRG-TR-88-4.

Pomerleau, D.A. (1991) "Efficient Training of Artificial Neural Networks for Autonomous Navigation". *Neural Computation* 3:1.

Pomerleau, D.A. (1993) *Neural Network Perception for Mobile Robot Guidance,* Kluwer Academic Publishing.

Stornetta, W.S., T. Hogg, and B.A. Huberman (1988). "A Dynamical Approach to Temporal Pattern Processing." In *Neural Information Processing Systems.* Ed. D.Z. Anderson, 750-759. NY:

Trick L.M. & Pylyshyn, Z.W. (1991), "Preattentive Indexing and Visual Counting: FINSTs and the Enumeration of Concentric Items". University of Western Ontario, Cogmem #58, 2/91.

Triesman, A. & Gelade, G. "A Feature Integration Theory of Attention" *Cognitive Psychology.*: 12,97-136, 1980.

Ullman, S. (1984) "Visual Routines" in *Visual Cognition.* Pinker, S. (ed.) MIT Press. Cambridge, Massachusetts, pp. 97-160.

Umilta, C. (1988) "Orienting of Attention" in *Handbook of Neuropsychology,* V 1. (Eds) F. Boiler, J. Grafman. Elsevier, Amsterdam.