

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

Prior Art Reference: Artificial Neural Network Computation on Graphics Process Unit (“ANN”)

ANN was published on July 31, 2005 and therefore qualifies as prior art to U.S. Patent No. RE49,461 (the “’461 patent” under, at least, (pre-AIA) 35 U.S.C. § 102(a) and/or § 102(b)). ANN anticipates and/or renders obvious claims 21–31 of the ’461 patent (the “Asserted Claims”).

The chart below provides representative examples of where each element of each claim is found within ANN. Citations are meant to be exemplary, not exhaustive, and NVIDIA reserves the right to identify and discuss additional portions of the reference in support of its contentions and/or to rebut arguments made by Plaintiff. Where NVIDIA states that ANN “discloses” a limitation, that disclosure may be express, implicit, and/or inherent. Citations to figures, drawings, tables, and the like include reference to any accompanying or related text. All internal cross-references are meant to incorporate the cross-referenced material as if fully set forth therein.

It is NVIDIA’s position that Plaintiff’s Infringement Contentions have not established that any accused product or service infringes any valid claim. Thus, NVIDIA’s statements below should not be treated as an admission, implication, or suggestion that NVIDIA agrees with Plaintiff regarding either the scope, construction, or interpretation of any of the Asserted Claims or the infringement theories advanced by Plaintiff in its Infringement Contentions, including whether any Asserted Claim satisfies 35 U.S.C. §§ 101 or 112. The exemplary disclosures in these invalidity contentions are informed by the apparent interpretation of the asserted claims by Plaintiff. These statements are not intended to suggest that NVIDIA agrees with Plaintiff’s application of any claim term, suggest a proposed construction at this stage of the case, or suggest that construction is needed.

To the extent Plaintiff argues that any element below is not disclosed by ANN, a person of ordinary skill in the art would have found it obvious to combine the teachings of ANN with the background knowledge of a person of ordinary skill in the art and/or the additional references, and exemplary teachings, set forth in NVIDIA’s Invalidity Contentions, including all exhibits and appendices thereto. Additional exemplary combinations and reasons for obviousness are described in the cover pleading, and are incorporated by reference herein.

Plaintiff has yet to identify any limitation of the Asserted Claims that it contends is not anticipated and/or rendered obvious by ANN. NVIDIA therefore expressly reserves the right to respond to any such contention, including by identifying additional obviousness combinations, if Plaintiff makes any such contention. NVIDIA reserves the right to rely on additional citations or sources of evidence that also may be applicable, or that may become applicable in light of claim construction, changes in Plaintiff’s infringement contentions, and/or information obtained during discovery as the case progresses.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																
21[pre] A method of executing computations representing an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition, the method comprising:	NVIDIA takes no position in these Invalidity Contentions on whether the entirety of the preamble of this claim is a limitation. To the extent it is a limitation, ANN discloses, expressly and/or inherently a method of executing computations representing an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit, a first memory partition, and a second memory partition. <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses a method of executing computations presenting an artificial neural network on a computer system comprising at least one central processing unit (CPU), a processing unit.</i> The environment for CPU computing is INTEL P4 2.4G. Based on this PC, ATI 9550 and Nvidia GF5700 GPU are used for SOM computing, Nvidia GF6200 GPU is used for MLP computing. <i>A. CPU and GPU Train Time for SOM and Discussion</i> Usually the train process for self organize map is time consuming when the map is large enough. So there is a need to speed up the training procedure. For our test problem, 80 data are chosen for training the SOM and average time for the computation on CPU and GPU can be reached. The result is shown in Fig 4. <table><tr><th></th><th>128*128</th><th>256*256</th><th>512*512</th></tr><tr><td>CPU /ms</td><td>122</td><td>500</td><td>2000</td></tr><tr><td>GPU /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Fig. 4: SOM train computing time on GPU and CPU The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that		128*128	256*256	512*512	CPU /ms	122	500	2000	GPU /ms	211	244	511	GPUN /ms	104	256	900
	128*128	256*256	512*512														
CPU /ms	122	500	2000														
GPU /ms	211	244	511														
GPUN /ms	104	256	900														

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN												
	on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing. The result shows that GPU based MLP computation is about 200 times faster than that of CPU. And the result also shows that GPU computation is fast enough for the locating of the ball in real time. TABLE I: MLP COMPUTATION ON CPU AND GPU. <table><tr><td>CPU /ms</td><td>11328</td></tr><tr><td>GPUN /ms</td><td>46</td></tr></table> V. SOME IMPLEMENTATION DETAILS AND LESSONS To increase the efficiency, some basic rule should obey. The main rules are given below. First, create the GPU hardware program only once and enable it when it is needed. The reason is that when a new program is created, it will be compiled by Cg, which is time consuming. Second, if possible, do one's best to decrease the calculation passes. For computation scheme described in section 2.2, the main bottleneck is at finding minimum procedure. The test shows that the bottleneck comes from multi-pass used in finding minimum procedure. Table 2 shows result for different scheme of finding the minimum. To decrease the pass, we make two changes, the first is to combine the value calculation with one pass of find minimum, and the second is to combine two pass of find minimum computation into one pass. Instead of calculate 4 units, 16 units are calculated. GPU A represent calculation on ATI graphic card and GPUN represent the calculation on NVIDIA graphic card. The performance increase is clear, especially for Nvidia's graphic card. TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPUA more pass /ms</td><td>366</td><td>400</td><td>533</td></tr></table>	CPU /ms	11328	GPUN /ms	46	KFM size	128*128	256*256	512*512	GPUA more pass /ms	366	400	533
CPU /ms	11328												
GPUN /ms	46												
KFM size	128*128	256*256	512*512										
GPUA more pass /ms	366	400	533										

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN			
	GPUA few pass /ms	211	244	511
	GPUN more pass /ms	190	640	2889
	GPUN few pass /ms	104	256	900
	Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. Fourth, hardware from different venders usually has different property. So if one implementation is not work at one kind of hardware, try another implementation. For example, in the calculation of minimum value, firstly we use the following code: <pre>c=tex2D(texture,coords).x<tex2D(texture,coords+half2(0,offset)).x? tex2D(texture,coords) : tex2D(texture,coords+half2(0,offset));</pre> Which can give the correct result in ATI card, but can not get correct result in an Nvidia's card, we think that the inner parallel schemes makes the difference. To work around it, the above sentence is changed to the following logical equivalent one: <pre>half4 c=tex2D(tex,coords); if (c.x>tex2D(tex,coords+half2(0,half side)).x) c=tex2D(tex,coords+half2(0,half side));</pre> Then the result is correct for both graphic cards. VI. CONCLUSION In this paper, implementation for two ANN models on graphic hardware is given. Inherent parallelism of commodity graphic hardware is used to accelerate the computation of ANN. The result shows that GPU is capable for some of ANN calculation, the graphic hardware make it possible for an increasing performance/cost ratio on the area of large size ANN computation.			

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	Compared to Bohn's initial computation on SGI workstation, our implementation has two benefits. One is our calculation is more precise, for we had use the float point computing. The other is that we only use a commodity available graphic card, which is much more easily available than SGI workstation so can be widely used. The implementation on graphic hardware introduce in this paper has other implicit benefit too. For the SOM computing, a multi-texture or 3-D texture can be used to store the map and make more general SOM computing without the restriction of the vector length of 4. For the MLP used in robot soccer, some graphic hardware have "video in" function, using this kind of graphic hardware; image information can be retrieved directly from camera and store on graphic memory, and don't need to transfer data between CPU and GPU, which will speed the process. We can also do other general ANN computation on GPU, because GPU provide almost all arithmetic operation, logic operation and some important mathematic function. And for the application of neural network on images, it is more naive to make such computing on a graphic hardware. For future works, one is to make other kinds of ANN computation on GPU. The other is to further deploy the MLP on more real situation of robot soccer, which include select better parameter and use faster algorithm on GPU. 4-5. III. MLP FOR REAL TIME BALL RECOGNIZING IN FIRA SOCCER ROBOT <i>A. Background</i> Currently, there are two main world soccer-robot competition, one is RoboCupI⁹¹, the other is FIR/ki^{wi}. In both case, Object are identified by their unique color. But in the near future, these color cues may be removed, so new vision algorithms based on model are needed to cope with the situation. The model based algorithm for finding ball and non-ball location will be discussed below 2-3.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	Fig 2. An image from FIRA SimuroSot 3. Fig2 shows an image taken from FIRA robot soccer 5vs5 simulator. Our task is to develop a model to recognize the ball, and then using this model to recognize the ball in real time. For each location, the characters are calculated considering the color value at a small area around the location. The area radius takes as 7, which is shown in Fig 3, 3. Artificial Neural Network (ANN) is widely used in pattern recognition related area. In some case, the computational load is very heavy, in other case, real time process is required. So there is a need to apply a parallel algorithm on it, and usually the computation for ANN is inherently parallel. In this paper, graphic hardware is used to speed up the computation of ANN. In recent years, graphic processing unit (GPU) grows faster than CPU. Graphic hardware venders provide programmability on GPU. In this paper, application of commodity

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	available GPU for two kinds of ANN models was explored. One is the self-organizing maps (SOM); the other is multi layer perceptron (MLP). The computation result shows that ANN computing on GPU is much faster than on standard CPU when the neural network is large. And some design rules for improve the efficiency on GPU are given. 1. Artificial neural network is widely used in classification and pattern recognition. In this paper, two kinds of ANN, self-organizing maps (SOM) I" and multi layer perceptron (MLP), are implemented on graphic hardware for speed up the computation. 1. MLP is a very simple neural network, the process is linear with one input layer, several hidden layer, and an output layer. It is usually trained by back propagation (BP) algorithm. SOM consists of one layer of n-dimensional units (neurons). It is fully connected with the network input. Additionally, there exist lateral connections through which a topological structure is imposed. For the standard model, the topology is a regular two-dimensional map instantiated by connections between each unit and its direct neighbors. 1. Rolfee gives an artificial neural network implementation using a GPU-based BLAS level 3 style single-precisions general matrix-matrix product. Bohni41 describes an SOM calculation method based on OpenGL hardware speed-up on SGI workstation, which inspired our work to further deploy the possibility to implement ANN calculation based on PC commodity graphic hardware. 1.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	In recent years, the graphic hardware performance is doubled every 12 months which is much faster than CPU's performance increase speed which is doubled every 18 months. And GPU vendors had make programmability on GPU, which make it possible for implement general-purpose computation. 1. In this paper, two kinds of ANN computation on GPU are given. In section 2, SOM computation model and implementation on graphic hardware is discussed. In section 3, MLP computation model and implementation on GPU is discussed. In section 4, the computation result and comparison are given for both CPU and GPU. In section 5, some of the design details and lessons we learned during implementation are given. In section 6, conclusion and some future works are given. 1. II. COMPUTATIONAL MODEL AND IMPLEMENT METHOD FOR SOM A. The SOM Computational Model The SOM takes a two-step computation: search for the best matching unit (BMU) and modify the map according to a distance function of the lateral connections. Usually, the Euclid distance is chosen as similarity measure method. The calculation formula takes as follow: $\ W_b - \xi\ < \ W_i - \xi\ \text{ for any } i; \quad [1]$ Modify the unit value regarding a distance function of the lateral connections as follow.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	$W_i^{new} = W_i^{old} - \epsilon \Omega(r_b, f_i) * (W_i^{old} - \xi) \quad [2]$ 1. B. Computational Model of MLP Three layer MLP neural network is selected to recognize the ball. The input layer consists of seven nodes for seven characters. The hidden layer consists of three nodes, and the output layer consists of just one node. Back propagation method was chosen to train the network. The train set takes as follow. 16 locations are selected from each of 10 robot cars, 4 locations are selected from center of ball, and one location is selected from background field. Totally, 165 locations are selected as the train set. For each selected location, seven characters are calculated and take as an element of the train set. The trained MLP is used to recognize and trace the ball in real time. There are two main computation steps for MLP used as the classification machine. The first one is matrix multiplication: $net = w \bullet x + b \quad [3]$ The second one is sigmoid function calculation: $\sigma(net) = \frac{1}{1 + e^{-net}} \quad [4]$ As for the determination of ball on robot soccer, the MLP calculation was applied on each point in the play ground. And MLP was used to distinguish between ball and non-ball position. C. MLP Implementation on Graphic Hardware

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	Current GPU has a limited instruction length and a limited number of temporary variables for calculation at each location. So multipass is needed for complex problem. For MLP discussed in this section, three passes are performed. In the first pass, average values of three color and luminance are calculated. In the second pass, the standard deviations of the three colors are calculated. In the third pass, classification result is got from MLP calculation on the characteristic. 3. The implementation on graphic hardware introduce in this paper has other implicit benefit too. For the SOM computing, a multi-texture or 3-D texture can be used to store the map and make more general SOM computing without the restriction of the vector length of 4. For the MLP used in robot soccer, some graphic hardware have "video in" function, using this kind of graphic hardware; image information can be retrieved directly from camera and store on graphic memory, and don't need to transfer data between CPU and GPU, which will speed the process. 5. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
21[a]: executing, by the at least one CPU, a user interaction stream, the user interaction stream	ANN discloses, expressly and/or inherently, executing, by the at least one CPU, a user interaction stream, the user interaction stream controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition. <i>See e.g.:</i>

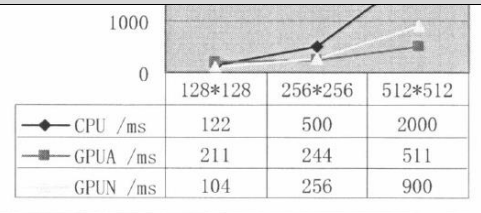
Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
controlling transfer of inputs to the artificial neural network to the first memory partition and the second memory partition;	<i>As a non-limiting example, ANN discloses at least one CPU (e.g., an INTEL P4) and a user interaction stream for controlling transfer of inputs to the neural network being executed on a GPU.</i> The implementation on graphic hardware introduce in this paper has other implicit benefit too. For the SOM computing, a multi-texture or 3-D texture can be used to store the map and make more general SOM computing without the restriction of the vector length of 4. For the MLP used in robot soccer, some graphic hardware have "video in" function, using this kind of graphic hardware; image information can be retrieved directly from camera and store on graphic memory, and don't need to transfer data between CPU and GPU, which will speed the process. 5. IV. RESULTS AND DISCUSSION The environment for CPU computing is INTEL P4 2.4G. Based on this PC, ATI 9550 and Nvidia GF5700 GPU are used for SOM computing, Nvidia GF6200 GPU is used for MLP computing. <i>A. CPU and CPU Train Time for SOM and Discussion</i> Usually the train process for self organize map is time consuming when the map is large enough. So there is a need to speed up the training procedure. For our test problem, 80 data are chosen for training the SOM and average time for the computation on CPU and GPU can be reached. The result is shown in Fig 4.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461

ANN



	128*128	256*256	512*512
CPU /ms	122	500	2000
GPUN /ms	211	244	511
GPUN /ms	104	256	900

Fig. 4: SOM train computing time on GPU and CPU

The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing.

B. MLP Computation Time for CPU and GPU

The application of MLP in this paper is to trace the ball in robot soccer in real time. The result is shown in table I.

4.

The result shows that GPU based MLP computation is about 200 times faster than that of CPU. And the result also shows that GPU computation is fast enough for the locating of the ball in real time.

TABLE I: MLP COMPUTATION ON CPU AND GPU.

CPU /ms	11328
GPUN /ms	46

V. SOME IMPLEMENTATION DETAILS AND LESSONS

To increase the efficiency, some basic rule should obey. The main rules are given below.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																				
	First, create the GPU hardware program only once and enable it when it is needed. The reason is that when a new program is created, it will be compiled by Cg, which is time consuming. Second, if possible, do one's best to decrease the calculation passes. For computation scheme described in section 2.2, the main bottleneck is at finding minimum procedure. The test shows that the bottleneck comes from multi-pass used in finding minimum procedure. Table 2 shows result for different scheme of finding the minimum. To decrease the pass, we make two changes, the first is to combine the value calculation with one pass of find minimum, and the second is to combine two pass of find minimum computation into one pass. Instead of calculate 4 units, 16 units are calculated. GPUa represent calculation on ATI graphic card and GPUN represent the calculation on NVIDIA graphic card. The performance increase is clear, especially for Nvidia's graphic card. TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPUA more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPUA few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. Fourth, hardware from different venders usually has different property. So if one implementation is not work at one kind of hardware, try another implementation. For example, in the calculation of minimum value, firstly we use the following code: <pre>c=tex2D(texture,coords).x<tex2D(texture,coords+half 2(0,offset)).x? tex2D(texture,coords) : tex2D(texture,coords+half2(0,offset));</pre>	KFM size	128*128	256*256	512*512	GPUA more pass /ms	366	400	533	GPUA few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
KFM size	128*128	256*256	512*512																		
GPUA more pass /ms	366	400	533																		
GPUA few pass /ms	211	244	511																		
GPUN more pass /ms	190	640	2889																		
GPUN few pass /ms	104	256	900																		

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	Which can give the correct result in ATI card, but can not get correct result in an Nvidia's card, we think that the inner parallel schemes makes the difference. To work around it, the above sentence is changed to the following logical equivalent one: <pre> half4 c=tex2D(tex,coords); if (c.x>tex2D(tex,coords+half2(0,half side)).x) c=tex2D(tex,coords+half2(0,half side)); </pre> Then the result is correct for both graphic cards. VI. CONCLUSION In this paper, implementation for two ANN models on graphic hardware is given. Inherent parallelism of commodity graphic hardware is used to accelerate the computation of ANN. The result shows that GPU is capable for some of ANN calculation, the graphic hardware make it possible for an increasing performance/cost ratio on the area of large size ANN computation. Compared to Bohn's initial computation on SGI workstation, our implementation has two benefits. One is our calculation is more precise, for we had use the float point computing. The other is that we only use a commodity available graphic card, which is much more easily available than SGI workstation so can be widely used. The implementation on graphic hardware introduce in this paper has other implicit benefit too. For the SOM computing, a multi-texture or 3-D texture can be used to store the map and make more general SOM computing without the restriction of the vector length of 4. For the MLP used in robot soccer, some graphic hardware have "video in" function, using this kind of graphic hardware; image information can be retrieved directly from camera and store on graphic memory, and don't need to transfer data between CPU and GPU, which will speed the process. We can also do other general ANN computation on GPU, because GPU provide almost all arithmetic operation, logic operation and some important mathematic function. And for the application of neural network on images, it is more naive to make such computing on a graphic hardware.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN				
	For future works, one is to make other kinds of ANN computation on GPU. The other is to further deploy the MLP on more real situation of robot soccer, which include select better parameter and use faster algorithm on GPU. 4-5. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA’s Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the ’461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.				
21[b]: executing, by the processing unit, a computational stream, the computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network;	ANN discloses, expressly and/or inherently, executing, by the processing unit, a computational stream, the computational stream controlling data exchange between the user interaction stream and the computational stream during execution of the computations representing the artificial neural network. <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses a computation stream controlling data exchange between a user interaction stream and the computational stream during execution of the computations representing the artificial neural network in the form of data exchange between the CPU and the GPU and the tracing of the soccer ball in real time.</i> The result shows that GPU based MLP computation is about 200 times faster than that of CPU. And the result also shows that GPU computation is fast enough for the locating of the ball in real time. TABLE I: MLP COMPUTATION ON CPU AND GPU. <table border="1" data-bbox="787 1328 1224 1422"> <tr> <td>CPU /ms</td><td>11328</td></tr> <tr> <td>GPUN /ms</td><td>46</td></tr> </table>	CPU /ms	11328	GPUN /ms	46
CPU /ms	11328				
GPUN /ms	46				

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																				
	V. SOME IMPLEMENTATION DETAILS AND LESSONS To increase the efficiency, some basic rule should obey. The main rules are given below. First, create the GPU hardware program only once and enable it when it is needed. The reason is that when a new program is created, it will be compiled by Cg, which is time consuming. Second, if possible, do one's best to decrease the calculation passes. For computation scheme described in section 2.2, the main bottleneck is at finding minimum procedure. The test shows that the bottleneck comes from multi-pass used in finding minimum procedure. Table 2 shows result for different scheme of finding the minimum. To decrease the pass, we make two changes, the first is to combine the value calculation with one pass of find minimum, and the second is to combine two pass of find minimum computation into one pass. Instead of calculate 4 units, 16 units are calculated. GPUa represent calculation on ATI graphic card and GPUN represent the calculation on NVIDIA graphic card. The performance increase is clear, especially for Nvidia's graphic card. TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPUA more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPUA few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. Fourth, hardware from different vendors usually has different property. So if one implementation is not work at one kind of hardware, try another implementation. For example, in the calculation of minimum value, firstly we use the following code:	KFM size	128*128	256*256	512*512	GPUA more pass /ms	366	400	533	GPUA few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
KFM size	128*128	256*256	512*512																		
GPUA more pass /ms	366	400	533																		
GPUA few pass /ms	211	244	511																		
GPUN more pass /ms	190	640	2889																		
GPUN few pass /ms	104	256	900																		

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	c=tex2D(texture,coords).x<tex2D(texture,coords+half 2(0,offset)).x? tex2D(texture,coords) : tex2D(texture,coords+half2(0,offset)); Which can give the correct result in ATI card, but can not get correct result in an Nvidia's card, we think that the inner parallel schemes makes the difference. To work around it, the above sentence is changed to the following logical equivalent one: half4 c=tex2D(tex,coords); if (c.x>tex2D(tex,coords+half2(0,half side)).x) c=tex2D(tex,coords+half2(0,half side)); Then the result is correct for both graphic cards. 4-5. <i>B. Computational Model of MLP</i> Three layer MLP neural network is selected to recognize the ball. The input layer consists of seven nodes for seven characters. The hidden layer consists of three nodes, and the output layer consists of just one node. Back propagation method was chosen to train the network. The train set takes as follow. 16 locations are selected from each of 10 robot cars, 4 locations are selected from center of ball, and one location is selected from background field. Totally, 165 locations are selected as the train set. For each selected location, seven characters are calculated and take as an element of the train set. The trained MLP is used to recognize and trace the ball in real time. There are two main computation steps for MLP used as the classification machine. The first one is matrix multiplication: $net=w \bullet x + b$ [³] The second one is sigmoid function calculation:

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	$\sigma(net) = \frac{1}{1 + e^{-net}}$ As for the determination of ball on robot soccer, the MLP calculation was applied on each point in the play ground. And MLP was used to distinguish between ball and non-ball position. 3. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA’s Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the ’461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
21[c]: shifting control of a data exchange between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network;	ANN discloses, expressly and/or inherently, shifting control of a data exchange between the user interaction stream and the computational stream to the computational stream in response to starting execution of the computations representing the artificial neural network. <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses shifting control of a data exchange between the user interaction stream and the computational stream in response to starting execution of computations in the form of the data exchange between the CPU and the GPU as part of tracking a soccer ball in real time.</i> The result shows that GPU based MLP computation is about 200 times faster than that of CPU. And the result also shows that GPU computation is fast enough for the locating of the ball in real time. TABLE I: MLP COMPUTATION ON CPU AND GPU.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																								
	<table><tr><td>CPU /ms</td><td>11328</td></tr><tr><td>GPUN /ms</td><td>46</td></tr></table> V. SOME IMPLEMENTATION DETAILS AND LESSONS To increase the efficiency, some basic rule should obey. The main rules are given below. First, create the GPU hardware program only once and enable it when it is needed. The reason is that when a new program is created, it will be compiled by Cg, which is time consuming. Second, if possible, do one's best to decrease the calculation passes. For computation scheme described in section 2.2, the main bottleneck is at finding minimum procedure. The test shows that the bottleneck comes from multi-pass used in finding minimum procedure. Table 2 shows result for different scheme of finding the minimum. To decrease the pass, we make two changes, the first is to combine the value calculation with one pass of find minimum, and the second is to combine two pass of find minimum computation into one pass. Instead of calculate 4 units, 16 units are calculated. GPU A represent calculation on ATI graphic card and GPUN represent the calculation on NVIDIA graphic card. The performance increase is clear, especially for Nvidia's graphic card. TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPU A more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPU A few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program.	CPU /ms	11328	GPUN /ms	46	KFM size	128*128	256*256	512*512	GPU A more pass /ms	366	400	533	GPU A few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
CPU /ms	11328																								
GPUN /ms	46																								
KFM size	128*128	256*256	512*512																						
GPU A more pass /ms	366	400	533																						
GPU A few pass /ms	211	244	511																						
GPUN more pass /ms	190	640	2889																						
GPUN few pass /ms	104	256	900																						

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	Fourth, hardware from different venders usually has different property. So if one implementation is not work at one kind of hardware, try another implementation. For example, in the calculation of minimum value, firstly we use the following code: <pre>c=tex2D(texture,coords).x<tex2D(texture,coords+half 2(0,offset)).x? tex2D(texture,coords) : tex2D(texture,coords+half2(0,offset));</pre> Which can give the correct result in ATI card, but can not get correct result in an Nvidia's card, we think that the inner parallel schemes makes the difference. To work around it, the above sentence is changed to the following logical equivalent one: <pre>half4 c=tex2D(tex,coords); if (c.x>tex2D(tex,coords+half2(0,half side)).x) c=tex2D(tex,coords+half2(0,half side));</pre> Then the result is correct for both graphic cards. 4-5. <i>B. Computational Model of MLP</i> Three layer MLP neural network is selected to recognize the ball. The input layer consists of seven nodes for seven characters. The hidden layer consists of three nodes, and the output layer consists of just one node. Back propagation method was chosen to train the network. The train set takes as follow. 16 locations are selected from each of 10 robot cars, 4 locations are selected from center of ball, and one location is selected from background field. Totally, 165 locations are selected as the train set. For each selected location, seven characters are calculated and take as an element of the train set. The trained MLP is used to recognize and trace the ball in real time. There are two main computation steps for MLP used as the classification machine. The first one is matrix multiplication:

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	$net=w \bullet x+b$ [³] The second one is sigmoid function calculation: $\sigma(net) = \frac{1}{1 + e^{-net}}$ As for the determination of ball on robot soccer, the MLP calculation was applied on each point in the play ground. And MLP was used to distinguish between ball and non-ball position. 3. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
21[d]: shifting control of the data exchange between the user interaction stream and the computational stream to the user interaction stream in response to completion or interruption of the	ANN discloses, expressly and/or inherently, shifting control of the data exchange between the user interaction stream and the computational stream to the user interaction stream in response to completion or interruption of the computations representing the artificial neural network. <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses a shifting of control of the data exchange between a user interaction stream and the computational stream to the user interaction stream in</i>

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																				
computations representing the artificial neural network;	<i>response to completion or interruption of calculations in the form of decreasing the data exchange between the CPU and GPU and the use of the “Render to Texture” class and by use of the PBuffer.</i> TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPUA more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPUA few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. 4. <i>B. Computational Model of MLP</i> Three layer MLP neural network is selected to recognize the ball. The input layer consists of seven nodes for seven characters. The hidden layer consists of three nodes, and the output layer consists of just one node. Back propagation method was chosen to train the network. The train set takes as follow. 16 locations are selected from each of 10 robot cars, 4 locations are selected from center of ball, and one location is selected from background field. Totally, 165 locations are selected as the train set. For each selected location, seven characters are calculated and take as an element of the train set. The trained MLP is used to recognize and trace the ball in real time.	KFM size	128*128	256*256	512*512	GPUA more pass /ms	366	400	533	GPUA few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
KFM size	128*128	256*256	512*512																		
GPUA more pass /ms	366	400	533																		
GPUA few pass /ms	211	244	511																		
GPUN more pass /ms	190	640	2889																		
GPUN few pass /ms	104	256	900																		

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	There are two main computation steps for MLP used as the classification machine. The first one is matrix multiplication: $net = w \bullet x + b$ [³] The second one is sigmoid function calculation: $\sigma(net) = \frac{1}{1 + e^{-net}}$ As for the determination of ball on robot soccer, the MLP calculation was applied on each point in the play ground. And MLP was used to distinguish between ball and non-ball position. 3. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
21[e]: queueing a user command received by the user interaction stream during execution of the	ANN discloses, expressly and/or inherently, queueing a user command received by the user interaction stream during execution of the computations representing the artificial neural network. <i>See e.g.:</i>

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
computations representing the artificial neural network; and	<i>As a non-limiting example, ANN discloses queueing a user command (C for graphic).</i> B. SOM Implementation on Programmable GPU As discussed in the previous part, not all OpenGL functions supported by SGI workstation are supported by PC commodity graphic hardware. But recently, graphic hardware vendors provide programmability and some high-level program language. This kind of programmability is at a lower level than OpenGL, and is more powerful than fixed OpenGL function. In this implementation, Cg (C for graphic) is chosen as developing environment. As described in previous part, the calculation contains two steps. The first is finding the best matching unit, and the second is adjusting the value according to the distance from BMU. For finding the BMU, two steps are needed. In the first step, similarity measurement is calculated; in the second step, the minimum values which represent the BMU are found and located. The similarity computation code for GPU takes as follow: Half 4 temp= tex2D(texture,coords) -intrain; c.x = dot(temp,temp); 2. The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	references in NVIDIA’s Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the ’461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
21[f]: executing the user command during execution of the computations representing the artificial neural network at times determined by the computational stream.	ANN discloses, expressly and/or inherently, executing the user command during execution of the computations representing the artificial neural network at times determined by the computational stream. <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses executing user commands during execution of computations (multipass method).</i> The first two sentences calculate the square of Euclid distance of two vectors, the third sentence save the unit coordinate, which will be used late for locating the best match unit. The main difficulty in Cg computation comes from finding the minimum value and determination its location, for there is no global variable in Cg environment. We use a multipass method to calculate it. Our scheme shows as Fig 1. In each pass we find the minimum value of four units show as colored and save the result in a small size texture. After some step, the size decrease to 1, and we can get the minimum value and its location. After get BMU, we can adjust the self-organize map according to equation [2]. 2. <i>B. Computational Model of MLP</i> Three layer MLP neural network is selected to recognize the ball. The input layer consists of seven nodes for seven characters. The hidden layer consists of three nodes, and the output layer consists of just one node. Back propagation method was chosen to train the network. The train set takes as follow. 16 locations are selected from each of 10 robot cars, 4 locations are selected from center of ball, and one location is selected from background field. Totally, 165

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	locations are selected as the train set. For each selected location, seven characters are calculated and take as an element of the train set. The trained MLP is used to recognize and trace the ball in real time. There are two main computation steps for MLP used as the classification machine. The first one is matrix multiplication: $net = w \cdot x + b$ [³] The second one is sigmoid function calculation: $\sigma(net) = \frac{1}{1 + e^{-net}}$ As for the determination of ball on robot soccer, the MLP calculation was applied on each point in the play ground. And MLP was used to distinguish between ball and non-ball position. 3. The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing. 4.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																								
	The result shows that GPU based MLP computation is about 200 times faster than that of CPU. And the result also shows that GPU computation is fast enough for the locating of the ball in real time. TABLE I: MLP COMPUTATION ON CPU AND GPU. <table><tr><td>CPU /ms</td><td>11328</td></tr><tr><td>GPUN /ms</td><td>46</td></tr></table> V. SOME IMPLEMENTATION DETAILS AND LESSONS To increase the efficiency, some basic rule should obey. The main rules are given below. First, create the GPU hardware program only once and enable it when it is needed. The reason is that when a new program is created, it will be compiled by Cg, which is time consuming. Second, if possible, do one's best to decrease the calculation passes. For computation scheme described in section 2.2, the main bottleneck is at finding minimum procedure. The test shows that the bottleneck comes from multi-pass used in finding minimum procedure. Table 2 shows result for different scheme of finding the minimum. To decrease the pass, we make two changes, the first is to combine the value calculation with one pass of find minimum, and the second is to combine two pass of find minimum computation into one pass. Instead of calculate 4 units, 16 units are calculated. GPU A represent calculation on ATI graphic card and GPUN represent the calculation on NVIDIA graphic card. The performance increase is clear, especially for Nvidia's graphic card. TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPU A more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPU A few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table>	CPU /ms	11328	GPUN /ms	46	KFM size	128*128	256*256	512*512	GPU A more pass /ms	366	400	533	GPU A few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
CPU /ms	11328																								
GPUN /ms	46																								
KFM size	128*128	256*256	512*512																						
GPU A more pass /ms	366	400	533																						
GPU A few pass /ms	211	244	511																						
GPUN more pass /ms	190	640	2889																						
GPUN few pass /ms	104	256	900																						

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. Fourth, hardware from different venders usually has different property. So if one implementation is not work at one kind of hardware, try another implementation. For example, in the calculation of minimum value, firstly we use the following code: <pre>c=tex2D(texture,coords).x<tex2D(texture,coords+half 2(0,offset)).x? tex2D(texture,coords) : tex2D(texture,coords+half2(0,offset));</pre> Which can give the correct result in ATI card, but can not get correct result in an Nvidia's card, we think that the inner parallel schemes makes the difference. To work around it, the above sentence is changed to the following logical equivalent one: <pre>half4 c=tex2D(tex,coords); if (c.x>tex2D(tex,coords+half2(0,half side)).x) c=tex2D(tex,coords+half2(0,half side));</pre> Then the result is correct for both graphic cards. 4-5. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.

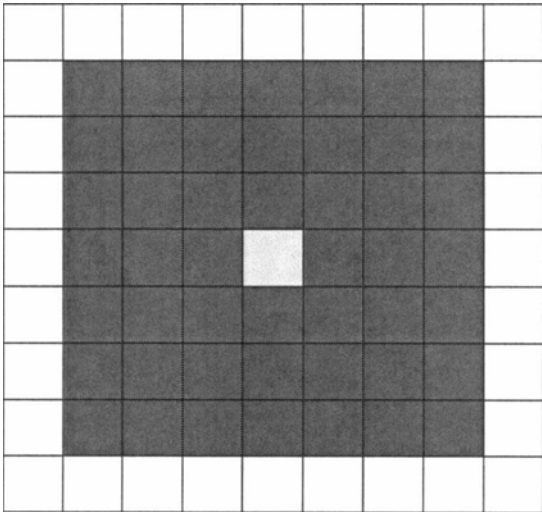
Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																				
22. The method of claim 21, wherein the user interaction stream controls the data exchange between the user interaction stream and the computational stream outside of execution of the computations representing the artificial neural network.	ANN discloses, expressly and/or inherently, the method of claim 21, wherein the user interaction stream controls the data exchange between the user interaction stream and the computational stream outside of execution of the computations representing the artificial neural network. See claim 21. See e.g.: <i>As a non-limiting example, ANN discloses the user interaction stream controlling the data exchange outside of execution of computations in the form of the data exchange between the CPU and GPU and the use the “Render to Texture” class and the PBuffer.</i> TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPUA more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPUA few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. 4. <i>As a non-limiting example, ANN discloses a user interaction stream controlling the data exchange in the form of graphics card code.</i>	KFM size	128*128	256*256	512*512	GPUA more pass /ms	366	400	533	GPUA few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
KFM size	128*128	256*256	512*512																		
GPUA more pass /ms	366	400	533																		
GPUA few pass /ms	211	244	511																		
GPUN more pass /ms	190	640	2889																		
GPUN few pass /ms	104	256	900																		

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
23. The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of computational elements of the computations representing the artificial neural network.	ANN discloses, expressly and/or inherently, the method of claim 21, wherein executing the user interaction stream comprises controlling setting and editing of computational elements of the computations representing the artificial neural network. <i>See claim 21.</i> <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses controlling setting and editing of computational elements by selecting seven parameters as characters of the ball.</i>

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	For each location, the characters are calculated considering the color value at a small area around the location. The area radius takes as 7, which is shown in Fig 3,  Fig 3. region for character calculation Seven parameters are selected as characters of the ball. There are 3 average value of the colors red, green and blue around the concerning position: $\text{mean}_{x,y} = \frac{1}{49} \sum_{i=-3}^3 \sum_{j=-3}^3 c_{i,j}$ 3. The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
24. The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the computations representing the artificial neural network.	ANN discloses, expressly and/or inherently, the method of claim 21, wherein executing the user interaction stream comprises controlling setting and editing of parameters of the computations representing the artificial neural network. <i>See claims 21, 23.</i> <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses controlling setting and editing of parameters of computations by selecting seven parameters as characters of the ball.</i> For each location, the characters are calculated considering the color value at a small area around the location. The area radius takes as 7, which is shown in Fig 3,

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	Fig 3. region for character calculation Seven parameters are selected as characters of the ball. There are 3 average value of the colors red, green and blue around the concerning position: 3. The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
25. The method of claim 21, wherein executing the user interaction stream comprises: controlling setting and editing of parameters of the inputs to the artificial neural network.	ANN discloses, expressly and/or inherently, the method of claim 21, wherein executing the user interaction stream comprises controlling setting and editing of parameters of the inputs to the artificial neural network. <i>See claim 21.</i> <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses execution of the user interaction stream comprising controlling setting and editing of inputs (input layer used for tracking ball in live time).</i> <i>B. Computational Model of MLP</i> Three layer MLP neural network is selected to recognize the ball. The input layer consists of seven nodes for seven characters. The hidden layer consists of three nodes, and the output layer consists of just one node. Back propagation method was chosen to train the network. The train set takes as follow. 16 locations are selected from each of 10 robot cars, 4 locations are selected from center of ball, and one location is selected from background field. Totally, 165 locations are selected as the train set. For each selected location, seven characters are calculated and take as an element of the train set. The trained MLP is used to recognize and trace the ball in real time.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																				
	There are two main computation steps for MLP used as the classification machine. The first one is matrix multiplication: $net=w\bullet x+b$ [³] The second one is sigmoid function calculation: $\sigma(net)=\frac{1}{1+e^{-net}}$ As for the determination of ball on robot soccer, the MLP calculation was applied on each point in the play ground. And MLP was used to distinguish between ball and non-ball position. 3. TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPUA more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPUA few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. 4.	KFM size	128*128	256*256	512*512	GPUA more pass /ms	366	400	533	GPUA few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
KFM size	128*128	256*256	512*512																		
GPUA more pass /ms	366	400	533																		
GPUA few pass /ms	211	244	511																		
GPUN more pass /ms	190	640	2889																		
GPUN few pass /ms	104	256	900																		

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
26. The method of claim 21, wherein executing the user interaction stream comprises: specifying an output to be saved to disk and/or displayed on a screen.	ANN discloses, expressly and/or inherently, the method of claim 21, wherein executing the user interaction stream comprises specifying an output to be saved to disk and/or displayed on a screen. <i>See claim 21.</i> <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses specified output, the “immediate result in a texture” to be saved to disk and or displayed on a screen.</i>

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																				
	TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPUA more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPUA few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.	KFM size	128*128	256*256	512*512	GPUA more pass /ms	366	400	533	GPUA few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
KFM size	128*128	256*256	512*512																		
GPUA more pass /ms	366	400	533																		
GPUA few pass /ms	211	244	511																		
GPUN more pass /ms	190	640	2889																		
GPUN few pass /ms	104	256	900																		
27. The method of claim 21, wherein executing the user interaction stream comprises: parsing elements to be used in the	ANN discloses, expressly and/or inherently, the method of claim 21, wherein executing the user interaction stream comprises parsing elements to be used in the computations representing the artificial neural network. See claim 21.																				

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																				
computations representing the artificial neural network.	See e.g.: As a non-limiting example, ANN discloses parsing elements to be used in the computations in the form of saving intermediate results in a texture to be reused as input data and through use of the PBuffer. TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPUA more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPUA few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples	KFM size	128*128	256*256	512*512	GPUA more pass /ms	366	400	533	GPUA few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
KFM size	128*128	256*256	512*512																		
GPUA more pass /ms	366	400	533																		
GPUA few pass /ms	211	244	511																		
GPUN more pass /ms	190	640	2889																		
GPUN few pass /ms	104	256	900																		

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
28. The method of claim 27, wherein the processing unit comprises a graphics processing unit (GPU) and executing the user interaction stream further comprises: converting the elements into GPU programs.	ANN discloses, expressly and/or inherently, the method of claim 27, wherein the processing unit comprises a graphics processing unit (GPU) and executing the user interaction stream further comprises converting the elements into GPU programs. <i>See claim 27.</i> <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses converting elements into GPU programs in the form of the SOM implementation on Programmable GPU and compilation of graphics card code.</i> B. SOM Implementation on Programmable GPU As discussed in the previous part, not all OpenGL functions supported by SGI workstation are supported by PC commodity graphic hardware. But recently, graphic hardware vendors provide programmability and some high-level program language. This kind of programmability is at a lower level than OpenGL, and is more powerful than fixed OpenGL function. In this implementation, Cg 181 (C for graphic) is chosen as developing environment. As described in previous part, the calculation contains two steps. The first is finding the best matching unit, and the second is adjusting the value according to the distance from BMU. For finding the BMU, two steps are needed. In the first step, similarity measurement is calculated; in the second step, the minimum values which represent the BMU are found and located. The similarity computation code for GPU takes as follow: Half 4 temp= tex2D(texture,coords) -intrain; c.x = dot(temp,temp); 2.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
29. The method of claim 28, wherein executing the user interaction stream comprises: compiling the GPU programs.	ANN discloses, expressly and/or inherently, the method of claim 28, wherein executing the user interaction stream comprises: compiling the GPU programs. <i>See claim 28.</i> <i>See e.g.:</i> <i>ANN discloses compiling GPU programs (C for graphic).</i> B. SOM Implementation on Programmable GPU

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	As discussed in the previous part, not all OpenGL functions supported by SGI workstation are supported by PC commodity graphic hardware. But recently, graphic hardware vendors provide programmability and some high-level program language. This kind of programmability is at a lower level than OpenGL, and is more powerful than fixed OpenGL function. In this implementation, Cg 181 (C for graphic) is chosen as developing environment. As described in previous part, the calculation contains two steps. The first is finding the best matching unit, and the second is adjusting the value according to the distance from BMU. For finding the BMU, two steps are needed. In the first step, similarity measurement is calculated; in the second step, the minimum values which represent the BMU are found and located. The similarity computation code for GPU takes as follow: <pre>Half 4 temp= tex2D(texture,coords) -intrain; c.x = dot(temp,temp);</pre> 2. The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.
30. The method of claim 29, wherein executing the user interaction stream comprises: transferring the GPU programs to the second memory partition.	ANN discloses, expressly and/or inherently, the method of claim 29, wherein executing the user interaction stream comprises transferring the GPU programs to the second memory partition. <i>See claim 29.</i> <i>See e.g.:</i> <i>As a non-limiting example, ANN discloses transferring GPU programs to the second memory partition in the form of the use of C for graphic and the data exchange between the CPU and the GPU and use of the PBuffer.</i> B. SOM Implementation on Programmable GPU As discussed in the previous part, not all OpenGL functions supported by SGI workstation are supported by PC commodity graphic hardware. But recently, graphic hardware vendors provide programmability and some high-level program language. This kind of programmability is at a lower level than OpenGL, and is more powerful than fixed OpenGL function. In this implementation, Cg 181 (C for graphic) is chosen as developing environment. As described in previous part, the calculation contains two steps. The first is finding the best matching unit, and the second is adjusting the value according to the distance from BMU. For finding the BMU, two steps are needed. In the first step, similarity measurement is calculated; in the second step, the minimum values which represent the BMU are found and located. The similarity computation code for GPU takes as follow: Half 4 temp= tex2D(texture,coords) -intrain; c.x = dot(temp,temp); 2.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																				
	TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPUA more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPUA few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.	KFM size	128*128	256*256	512*512	GPUA more pass /ms	366	400	533	GPUA few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
KFM size	128*128	256*256	512*512																		
GPUA more pass /ms	366	400	533																		
GPUA few pass /ms	211	244	511																		
GPUN more pass /ms	190	640	2889																		
GPUN few pass /ms	104	256	900																		
31. The method of claim 21, further comprising: executing, by the at least one CPU, a data output stream, the data output stream controlling transfer of outputs of	ANN discloses, expressly and/or inherently, the method of claim 21, further comprising: executing, by the at least one CPU, a data output stream, the data output stream controlling transfer of outputs of the computations representing the artificial neural network to disk. See claim 21.																				

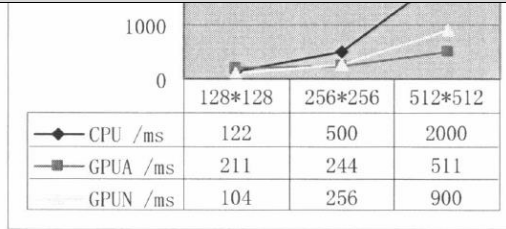
Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
the computations representing the artificial neural network to disk.	<i>See e.g.:</i> <i>As a non-limiting example, ANN discloses at least one CPU (e.g., an INTEL P4) executing a data output stream controlling transfer of outputs, such as results.</i> IV. RESULTS AND DISCUSSION The environment for CPU computing is INTEL P4 2.4G. Based on this PC, ATI 9550 and Nvidia GF5700 GPU are used for SOM computing, Nvidia GF6200 GPU is used for MLP computing. <i>C. CPU and CPU Train Time for SOM and Discussion</i> Usually the train process for self organize map is time consuming when the map is large enough. So there is a need to speed up the training procedure. For our test problem, 80 data are chosen for training the SOM and average time for the computation on CPU and GPU can be reached. The result is shown in Fig 4.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461

ANN



	128*128	256*256	512*512
CPU /ms	122	500	2000
GPUTA /ms	211	244	511
GPUN /ms	104	256	900

Fig. 4: SOM train computing time on GPU and CPU

The result shows that GPU based implementation is faster than CPU, especially for large self organize map. As map size increase, the computation time on GPU increase slowly than that on CPU. And different GPU had different result, for Nvidia's GPU, it takes the least time for small size SOM like 128*128, but for ATI's GPU it takes the least time for larger size SOM like 256*256. The difference may come from vendor's hardware implementation. The result shows that All's graphic card takes more time for the compile of program and the code is better optimized so computation time decrease with more data, but Nvidia's graphic card takes less time for compile and takes more time to computing.

D. MLP Computation Time for CPU and GPU

The application of MLP in this paper is to trace the ball in robot soccer in real time. The result is shown in table I.

4.

The result shows that GPU based MLP computation is about 200 times faster than that of CPU. And the result also shows that GPU computation is fast enough for the locating of the ball in real time.

TABLE I: MLP COMPUTATION ON CPU AND GPU.

CPU /ms	11328
GPUN /ms	46

V. SOME IMPLEMENTATION DETAILS AND LESSONS

To increase the efficiency, some basic rule should obey. The main rules are given below.

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN																				
	First, create the GPU hardware program only once and enable it when it is needed. The reason is that when a new program is created, it will be compiled by Cg, which is time consuming. Second, if possible, do one's best to decrease the calculation passes. For computation scheme described in section 2.2, the main bottleneck is at finding minimum procedure. The test shows that the bottleneck comes from multi-pass used in finding minimum procedure. Table 2 shows result for different scheme of finding the minimum. To decrease the pass, we make two changes, the first is to combine the value calculation with one pass of find minimum, and the second is to combine two pass of find minimum computation into one pass. Instead of calculate 4 units, 16 units are calculated. GPUa represent calculation on ATI graphic card and GPUN represent the calculation on NVIDIA graphic card. The performance increase is clear, especially for Nvidia's graphic card. TABLE H: COMPARISON BETWEEN MORE AND FEW PASS <table><tr><td>KFM size</td><td>128*128</td><td>256*256</td><td>512*512</td></tr><tr><td>GPUa more pass /ms</td><td>366</td><td>400</td><td>533</td></tr><tr><td>GPUa few pass /ms</td><td>211</td><td>244</td><td>511</td></tr><tr><td>GPUN more pass /ms</td><td>190</td><td>640</td><td>2889</td></tr><tr><td>GPUN few pass /ms</td><td>104</td><td>256</td><td>900</td></tr></table> Third, do best to decrease the data exchange between CPU and GPU. Usually OpenGL's PBuffer are used to save the intermediate result in a texture on GPU and reuse it as an input data. Harris¹ had created a class called "Render to Texture" to easy the use of PBuffer. We had used this class in our program. 4. To the extent Plaintiff contends that this claim element is not expressly or inherently disclosed, or is not obvious in view of ANN and the knowledge of a POSITA, it would have been obvious to combine the teachings of ANN with any of the prior art references in NVIDIA's Invalidity Contentions and any supplements thereto and the relevant section of charts for other prior art for the '461 patent in a manner that would result in the subject matter of this	KFM size	128*128	256*256	512*512	GPUa more pass /ms	366	400	533	GPUa few pass /ms	211	244	511	GPUN more pass /ms	190	640	2889	GPUN few pass /ms	104	256	900
KFM size	128*128	256*256	512*512																		
GPUa more pass /ms	366	400	533																		
GPUa few pass /ms	211	244	511																		
GPUN more pass /ms	190	640	2889																		
GPUN few pass /ms	104	256	900																		

Ex. C14 – Invalidity of U.S. Patent No. RE49,461 – ANN

U.S. Patent No. RE49,461	ANN
	limitation given, at the very least, the person's understanding of the state of the art, the problems addressed and solved in the prior art, and the teachings of ANN. Specific examples of such references are cited and explained in these invalidity contentions, including the cover pleading and charts C1–C15, and Appendix C.