DRAFT

18th ICCRTS

"C2 in Underdeveloped, Degraded and Denied Operational Environments"

Optimizing Content Delivery through Machine Learning

Topics:

Topic 3 – Data, Information, and Knowledge

James Schneider
Securboration Inc.
1050 W. NASA Blvd.
Melbourne FL, 32901

Anton DeFrancesco
Securboration Inc.
1050 W. NASA Blvd.
Melbourne FL, 32901

Bruce McQueary
Securboration Inc.
1050 W. NASA Blvd.
Melbourne FL, 32901

Point of Contact:
Anton DeFrancesco
Securboration Inc.
1050 W. NASA Blvd.
Suite 157
Melbourne FL, 32901
(321) 409-5252 x21
adefrancesco@securboration.com

DRAFT

1. Abstract

Today's military is facing significant content distribution challenges. The volume of web-based content distributed to enable complex synchronized joint operations spanning air, space, and cyber operations is increasing at an astonishing rate. Web-based content coupled with increasing amounts of supporting intelligence currently threaten to overwhelm the military network infrastructure. This challenge is exacerbated by the centralized nature of Operations; examples such as the Air Operations Command (AOC) and its difficulty to operate under disconnected, intermittent, and limited (DIL) connectivity inhibit the efficiency of C2 systems. Inaccessibility to mission critical content is an issue that jeopardizes operations on a daily basis.

This paper proposes using online learning optimization to identify, qualify, and prioritize the distribution of content to the edge – before it is requested. This paper details a process to aggregate features into higher level entities in order to properly predict message criticalness. Basic features will be extracted from the requests such as request time, content size, author, etc.; a deep belief network approach will be investigated to find higher level abstractions of these features. The papers' primary focus will be processing metadata and the application of machine learning algorithms in an effort to schedule data transmission through bandwidth restricted networks.

2. Introduction

With an estimated thirty-five percent of the world's population online [1], global network usage has skyrocketed in the past several years. From 2000 to 2012, the global growth of internet usage has jumped 566%, going from an estimated 361 million users in 2000 to an estimated 2.4 billion in 2012 [2]. Due to the increasing distribution of the internet and the slow proliferation of data (most data travels at 10-20% of the speed of light) a latency problem has developed. To combat the latency the concept of a Content Delivery Network (CDN) gained traction and became more prevalent. A CDN is a system of servers that provide a cache of high usage content for users in a relative area. This cache provides faster, more efficient, and more reliable delivery of the information to the end user by bypassing longer delivery paths and removing a single point of failure in the main site. Current CDN's evolved primarily from two forms of content networks. The first was a basic caching proxy configuration, such as shown in **Error! Reference source not found. Error! Reference source not found.**.

Such a network might typically be employed by an ISP for the benefit of users accessing the internet; such as through a Digital Subscriber Line (DSL) or cable modem. In the interest of improving performance and reducing bandwidth utilization, caching proxies are deployed close to the users. These user requests are directed through the caches, rather than directly to origin servers, by the ISP. When this configuration is properly implemented, the user's entire browsing session goes through a specific caching proxy. That caching proxy will therefore contain the "hot set" of all content being viewed by all of the users of that caching proxy. When a request is being handled at a caching proxy on behalf of a user, other decisions may be made, such as:

- A provider that deploys caches in many geographically diverse locations may also deploy regional parent caches to further aggregate user requests and responses. This may provide additional performance improvement and bandwidth savings. When parents are included, this is known as hierarchical caching.

- Using rich parenting protocols, redundant parents may be deployed such that a failure in a primary parent is detected and a backup is used.

- Using similar parenting protocols, requests may be partitioned such that requests for certain content domains are sent to a specific primary parent. This can help to maximize the efficient use of caching proxy resources.

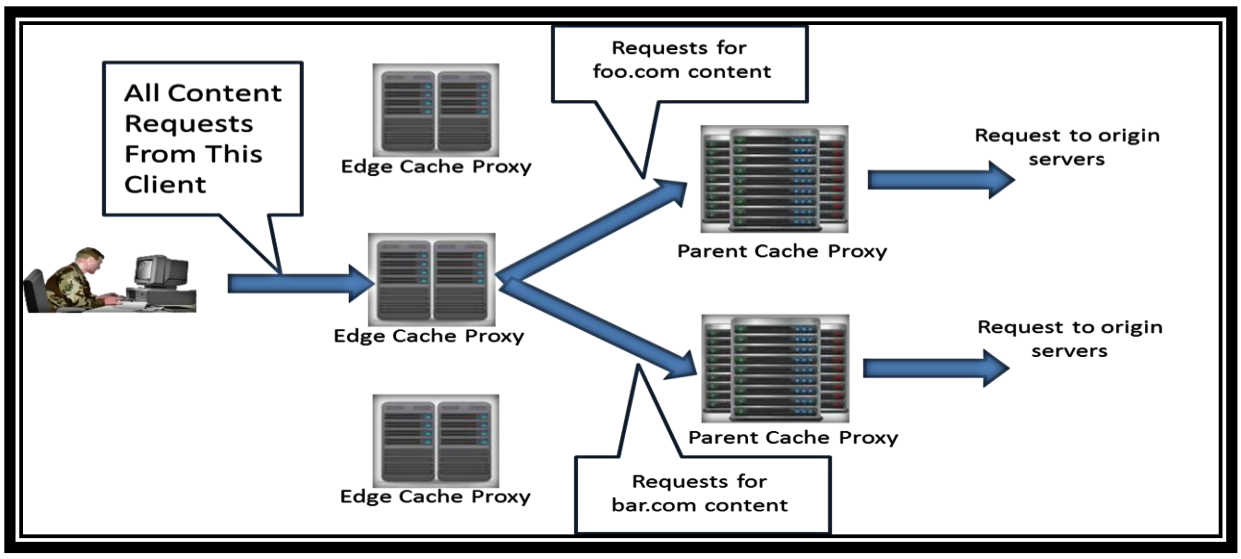


Figure 1: Hierarchical Caching Content Network

Another type of content network that has been in widespread use for several years is a server farm. (**Error! Reference source not found.**) A typical server farm makes use of a so-called "intelligent" or "content" switch (i.e., one that uses information in OSI layers 4-7). The switch examines content requests and dispatches them among a (potentially large) group of servers. Some of the goals of a server farm include:

- Creating the impression that the group of servers is actually a single origin site.
- Load-balancing of requests across all servers in the group.
- Automatic routing of requests away from servers that fail.
- Routing all requests for a particular user agent's session to the same server, in order to preserve session state.

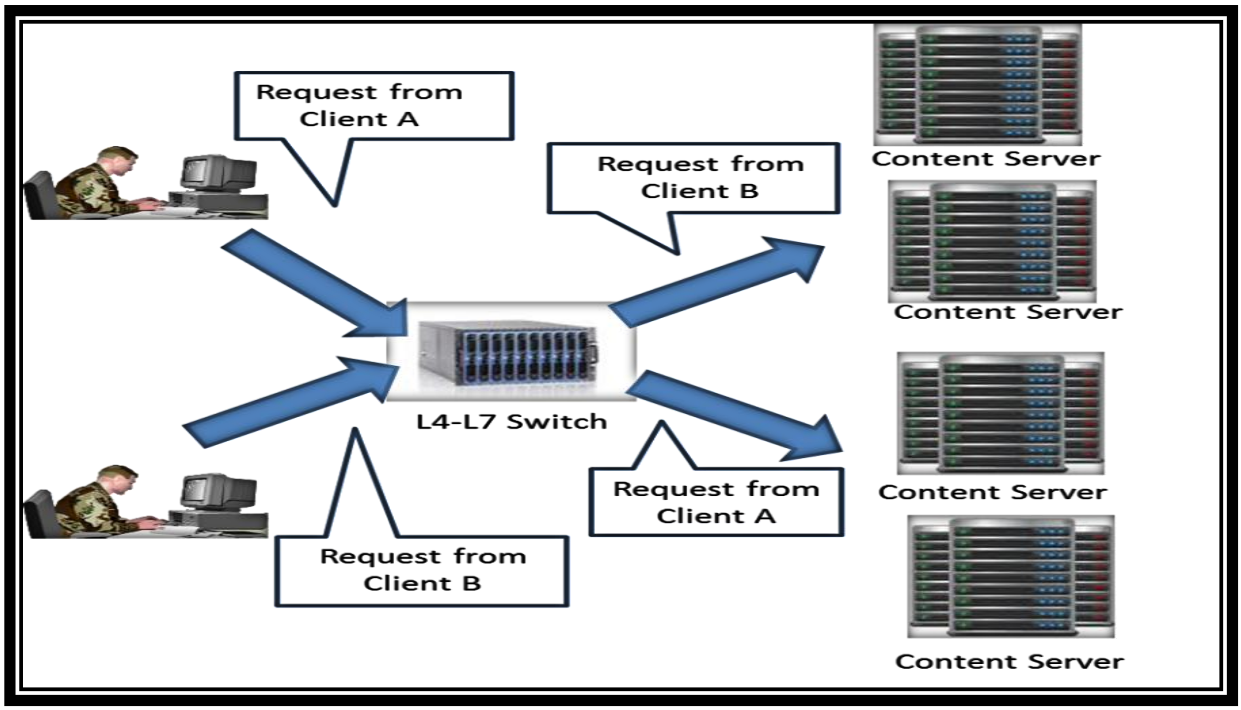


Figure 2: Server Farm Content Network

While hierarchical caching and server farms are useful content delivery techniques, both have limits.

- Server farms can improve the scalability of the origin server. However, since the multiple servers and other elements are typically deployed near the origin server, they do little to improve performance problems due to network congestion.
- Caching proxies can improve performance problems due to network congestion (since they are situated near the clients) but they cache objects based on client demand. Caching based on client demand performs poorly if the requests for a given object, while numerous in aggregate, are spread thinly among many different caching proxies. (In the worst case, an object could be requested n times via n distinct caching proxies, causing n distinct requests to the origin server -- or exactly the same behavior that would occur without any caching proxies in place.) Thus, a content provider with a popular content source can find that it has to invest in large server farms, load balancing, and high-bandwidth connections to keep up with demand. Even with those investments, the user experience may still be relatively poor due to congestion in the network as a whole.

CDN's address these limitations by essentially moving server-farm-like configurations out into network locations more typically occupied by caching proxies. A CDN has multiple replicas of each content item being hosted. A request from a browser for a single content item is directed to a "good" replica, where "good" usually means that the item is served to the client quickly compared to the time it would take fetch it from the origin server, with appropriate integrity and consistency. Static information about geographic locations and network connectivity is usually not sufficient to do a good job of choosing a replica. Instead, a CDN typically incorporates dynamic information about network conditions and load on the replicas, using a variety of optimization techniques to direct request so as to balance the load while providing content in the fastest (or the shortest) route between the server hosting the site and the end-user. **Error! Reference source not found. Error! Reference source not found.** illustrates the Akamai "edge" CDN, which serves content within one network hop of 90% of Internet users.

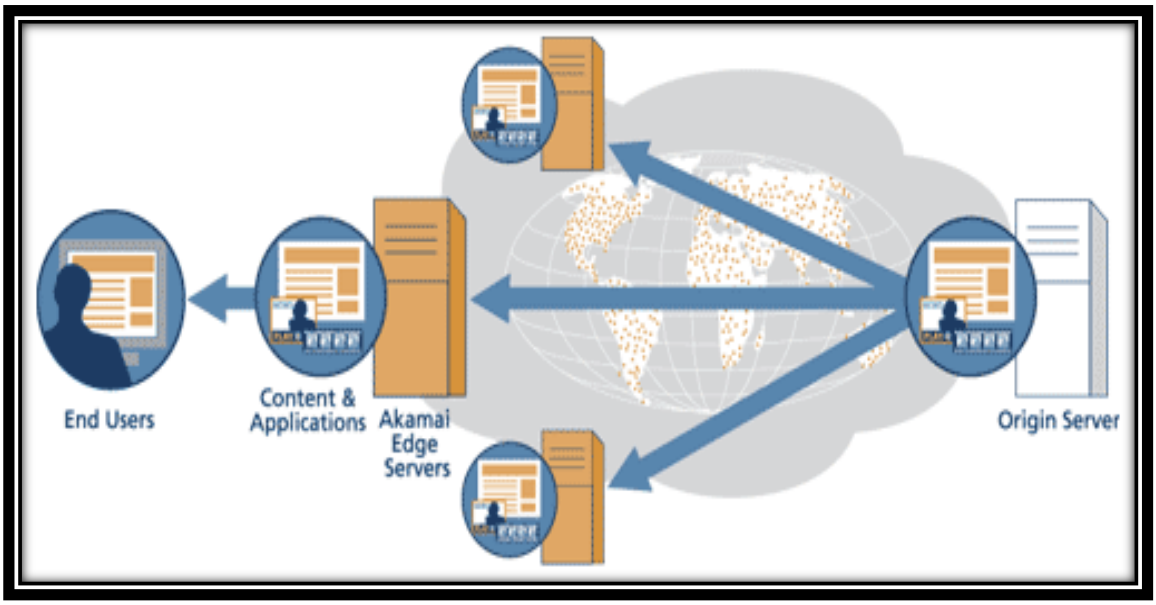


Figure 3: Akamai Edge CDN

Compared to using servers and surrogates in a single data center, a CDN is a relatively complex system encompassing multiple points of presence in locations that may be geographically far apart. Operating a CDN is not easy for a content provider (e.g. Apple, CNN, ESPN, etc.), since a content provider wants to focus its resources on developing high-value content, not on managing network infrastructure. Instead, a more typical arrangement is that a network service provider builds and operates a CDN, offering a content distribution service to a number of content

providers. A CDN enables a service provider to act on behalf of the content provider to deliver copies of origin server content to clients from multiple diverse locations. The increase in number and diversity of location is intended to improve download times and improve the user experience. A CDN has some combination of a content-delivery infrastructure, a request-routing infrastructure, a distribution infrastructure, and an accounting infrastructure. The content- delivery infrastructure consists of a set of "surrogate" servers that deliver copies of content to sets of users. The request-routing infrastructure consists of mechanisms that move a client toward a rendezvous with a surrogate. The distribution infrastructure consists of mechanisms that move content from the origin server to the surrogates. Finally, the accounting infrastructure tracks and collects data on request-routing, distribution, and delivery functions within the CDN.

3. Military Environment Problems

The number of sensors and the amount of data needed to support Command and Control (C2) for synchronized joint air, space, and cyber operations far surpasses, by orders of magnitude, previous content requirements. As an example, *one Global Hawk UAV requires about 500 Megabits per second of bandwidth (five times the entire bandwidth required by all of the US military during Desert Storm)* [3]

Whereas the course of action of doctrinally-based adversaries could be extrapolated with minimal amounts of information, today's asymmetric adversary and operational environment(s) requires persistent surveillance across the intelligence spectrum. The result of today's intelligence, surveillance and reconnaissance (ISR) platforms is an exponential growth of data that places an extreme burden on the military's network infrastructure that includes DIL wireless communication networks, mobile ad hoc networks and airborne networks. This is especially true in the Air Force, given its reliance on the centralized Air Operations Center (AOC) for C2 and ISR needs. AOC sites are tasked with providing critical processing, analysis and dissemination to combat forces at all levels across the command hierarchy, from the Joint Forces Commander to troops on the ground, anywhere in the world.

For example, consider the AOC at Al Udeid Air Base, Qatar which serves both operations in Iraq and Afghanistan. It is responsible for processing, exploiting, and disseminating increasingly large amounts of mission data. Supporting units which require data from the AOC are distributed throughout the Middle East (Qatar, Kuwait, Iraq, Afghanistan, etc.), more distant bases (Diego Garcia, Guam, Ramstein, etc.), and CONUS (bombers & strategic airlift). Additional supported headquarters and coordinating agencies are in country Joint Task Force Headquarters and other organizations (e.g. ISAF) as well as CONUS locations (e.g. CENTCOM HQ at MacDill AFB, FL).

This centralized approach localizes mission critical content in the AOC data servers making dissemination of this content challenging at best. Unlike the robust, ubiquitous network connectivity enjoyed by the civilian community, the military, especially tactical units are at the mercy of disconnected, intermittent, and limited (DIL) network connectivity. As a result, mission success is jeopardized if content is not adequately pushed to the edge, pulled from it, or distributed amongst tactical edge nodes.

Understanding this, Defense Information System Agency (DISA) implemented a managed service version of the commercial Akamai Content Delivery network onto the Defense Information Systems Network (DISN) known as the Global Information Grid (GIG) Content Delivery Service (GCDS). GCDS provides a scalable, flexible content delivery and application deployment platform to support the warfighter. It is a distributed computing platform comprised of globally deployed servers across both the NIPRNET, SIPRNET (with JWICS in the process), optimizing the delivery of DoD web content and web-based applications. Customers on the GCDS platform experience improved availability, improved end-user performance, reduced infrastructure costs, and improved traffic visibility.

The DISA implementation of the CDN provides enhanced service for those located on the GCDS, but due to the nature of the day to day operations, forward installations or those in DIL networks may still be at a disadvantage.

An example of this would be the timeframe prior to a series of missions. Operations and intelligence specialists, preparing for the missions, would start acquiring material for oncoming operations. Due to the nature of mission schedules, multiple specialists will be gathering their intelligence simultaneously. This can pose a problem on DIL networks and can leave mission specialists without some of the necessary information they require for successful mission execution.

This proses a delay problem as military planners interact with one another in some instances to create joint strike missions. In these missions each party respects their chain of command and usually gets orders disseminated from different sources and of varying size. These mission briefings are usually stored on higher security infrastructure rather than local computers and must be temporarily downloaded in order for the soldiers to undergo mission briefing. Due to these quick bursts of activity the CDN quickly becomes overburdened and the military must either provide more CDNs or taper bandwidth and with the nature of fast moving operations the latter is usually employed. This bandwidth tapering leads to many mission critical information feeds such as videos, audio and large text documents being delayed while smaller documents are acquired quickly.

The issues involved in this lowering of bandwidth currently restrict only the broadest of operations, but in the future these scheduling issues will become more prevalent due to increased usage of unmanned vehicles to assist soldiers. In this environment instead of a single transient spike of activity before a mission the activity remains at a higher load level for longer virtually preventing any other simultaneous missions. The solution to this problem is to prioritize information based on metrics these metrics will be known as features henceforth in this paper.

3.1 Scheduling Algorithm features

In a common scenario when all the data is pooled before a mission, the documents, as they are requested, contain some form of metadata. This metadata tells the CDN where the data is heading, where in the queue it was placed, how big the file size is and sometimes contains snippets of the file's contents. The CDN queue's these messages for delivery based on time of request, such that a message that was requested first is delivered first, but this might not be the optimal solution. One can imagine a situation where-in a request for supplemental information is blocking critical mission briefings and slowing down the network.

The solution to this problem is to provide some sort of intelligence to the queuing of information on the CDN for delivery. The biggest issue with this is what sort of features would one use to prioritize information into a queue. The simplest solution is to ask the requestor to assign an importance value to the document being requested; this would make the job of the scheduler easy as the most important documents are quickly delivered while the least important are delivered at a slower rate. Importance weighting though has issues as humans tend to introduce bias into the importance weighting, for example an Air force mission commander might see that the need for maintenance reports on the strike aircraft as vitally important, but in a triage like situation where another unit is requesting intelligence reports on enemy ground units it becomes less important. This is called contextualization of importance and is a very difficult problem as it's not only the information but the situation that influences the true importance of a document.

In- lieu of introducing human importance one can design a machine algorithm to decide importance based on contextual and inherent information in each of the pieces of data. This can be achieved via looking at some features of the data being requested including size, bandwidth used, data type, requestor rank, requestor urgency (correlation between time to mission or TTM), and many other types of information, but machine algorithms can dig deeper. Feature extraction is the process of looking at the contents of the data and accessing their usefulness. Although this sounds impossible it is quiet easy to accomplish, as many pieces of important information contain subtle clues to the mission type, mission parameters, and urgency. Introspecting these pieces of information along with the obvious features gives the machine algorithms its full feature set, but determination of what is important must still be completed as going through all this information in a timely fashion is impossible. This is where machine learning is useful as it allows introspection and deciding of what features are important and which are useless.

4. Machine Learning (ML)

Machine learning is a field associated with finding generalizable mathematical functions that associate some input pattern $x(p)$ to some unknown output pattern $d(p)$ with an error of $E^P(d(p))$ being less than a user specified maximum error. Machine learning grew out of the artificial intelligence field where the problem of recreating a human intelligence was paramount but proved insurmountable. This lead to collapse of artificial intelligence in the late 1980's commonly known as the AI winter [4]. Machine learning sprang up from the remains of the great AI winter and started with a simple premise in lieu of building a generalized learning algorithm, build specialized algorithms and eventually conglomerate these applications into a generalized intelligence. The key aspect of machine learning was to create engineering applications of artificial intelligence, but without the axiomatic reasoning that plagued the original designs. Instead machine learning proceeded to famously let the data speak, that is to create axiomatic notions from probabilistic interpretation of the data.

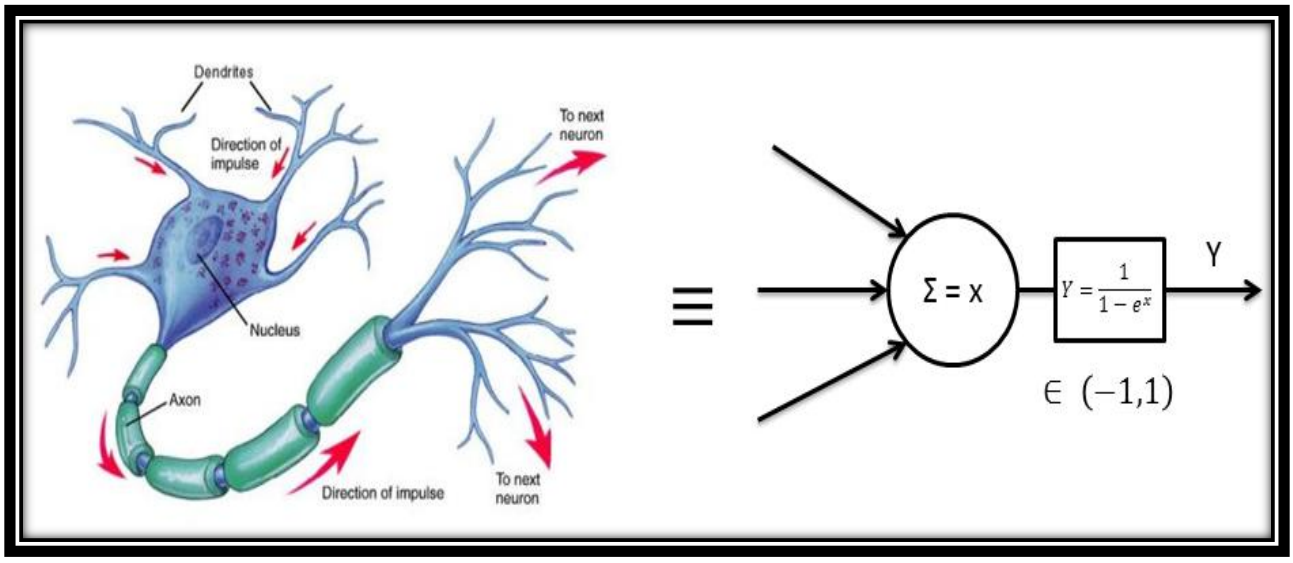


Figure 4: Neuron and its mathematical equivalent [5]

The introduction of Neural Networks was early in the development of artificial intelligence beginning in the late 1940's with Hebbian learning [6]. Hebbian learning was the first example of taking in some unknown pattern $x(p)$ and producing an output pattern $d(p)$. Hebbian learning took advantage of the mathematical foundation of a human neuron that described a neuron a simple computational unit with the power to summarize inputs, if these inputs were above some set threshold then the neuron would fire and produce a signal on its own [7]. Increasingly complex models became apparent when researchers began to realize that many of these small computational units arranged in a specific topological structure allows the structure to discover higher dimensional functions [8]. These multilayered networks though had no ability to learn these functions and the strength of the connections had to be programmed by hand by the user.

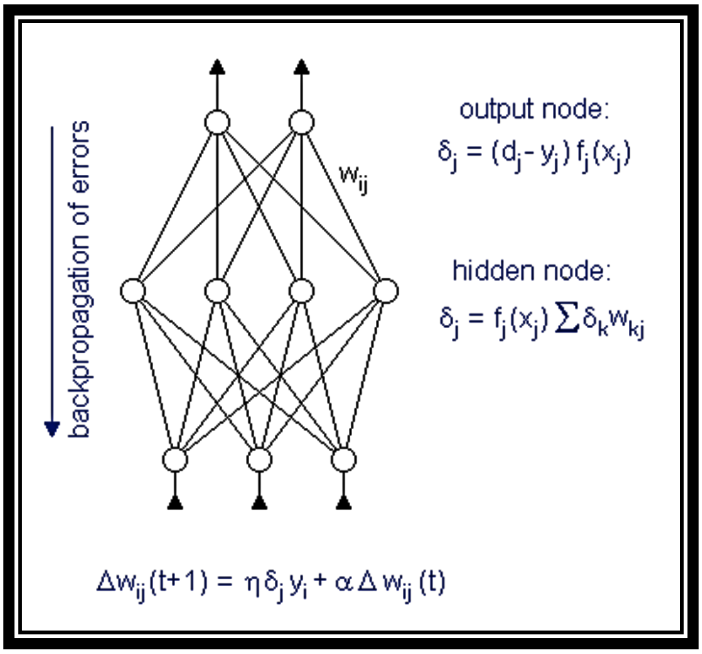


Figure 5: Backpropagation learning algorithm [9]

A revolution came about when the backpropagation learning algorithm was discovered wherein a multilayered neural network can on its own through supervised learning discover its connection strengths [10]. This supervised learning process was constructed similar to how one trains for a test, a dataset is given and in it contains the input pattern but also the desired output pattern. Initially the algorithm is given random connection strengths and is presented with a value from the input pattern. The output of the multilayered neural network is then compared to the correct output pattern, if it matches the next example in the input pattern is presented. If the program mismatches the pattern a cascade of changes moves through the multilayered neural network changing all the connection strengths by a small amount in proportion to the difference between the original output and the desired output. This back-propagation system was a revolution in the field as it proved that learnability was achievable, but there were many issues that persisted. One of these issues was the time complexity of the algorithm as it took weeks to run on even the smallest datasets. A second more pressing issue was that the complexity of the network was limited and thus, some higher dimensional functions could not be discovered by the algorithm.

4.1 Deep Learning

The major issue of pervious neural network learning algorithms was both limitations in efficiency and limitations in the dimensionality achievable by a network. There were other issues that provoked these problems those of feature selection, and data collection. In practice achieving large amounts of data with answers (supervised learning) is difficult and because of this insufficient learning takes place leading to decreased generalizability of the network. In this many researchers turned to simplifying models of data creation but this gave rise of the problem of leakage in data mining [11]. As the neural networks proceeded to learn from the simple model generated by the user instead of finding a general function for the data presented the neural networks usually found how the user created the model. The biggest issue for most of the community was feature selection prior to problem execution of the algorithm. Features are points that the user thinks are important to the overall function i.e. to create a function to determine speed of a vehicle the two most important features would be distance traveled and time, but one can think of other features that may or may not provide a more accurate estimate. These features represented prior information introduced to the algorithm usually causing bias in the calculations and further hampering generalizability of the output function.

Deep Learning is a new paradigm in machine learning that solves not only the major issues pertaining to back-propagation learning but the issues pertaining to learning in general. The key idea behind deep learning is to extend the number of layers of the neural network to a large amount and allow the algorithm to discover the features through an undirected learning process. Deep learning is modeled after the human neo-cortex allowing each sub-network to become tuned to specific features in the data and ignore the rest allowing for an ensemble learning process to take place. Having all these features is advantages in a single algorithm as co-occurrences can be discovered through generative processes; this increases the probability that a set of features will form a smaller set of super features that simplify the function generation process leading to a lower dimensional representation. There are currently two major deep learning paradigms the first being convolutional neural networks and the second are deep belief networks. An explanation of both will be provided to give a foundation on the work completed in relation to this project's goals.

4.2 Convolutional Neural Networks

Convolutional neural networks are based on dimensionality reduction techniques that allow combination of various spatial temporal features into singular values [12]. These networks are usually used on images and best trained on videos, as the network can aggregate sequential information in time to form a better representation of the overall pattern. Convolutional neural networks utilize trainable filters to remove noise. These filters remove scaling and rotational effects from images in order to achieve more accurate results, but filters can be generalized to remove any type of noise in a dataset. Typically the type of noise present in military applications are removed via digital signal processing units on the communication apparatus, but noise that isn't recognized is things such as irrelevant videos, images, or messages. These irrelevant data streams can cause the network to experience higher load at critical times i.e. during a mission briefing.

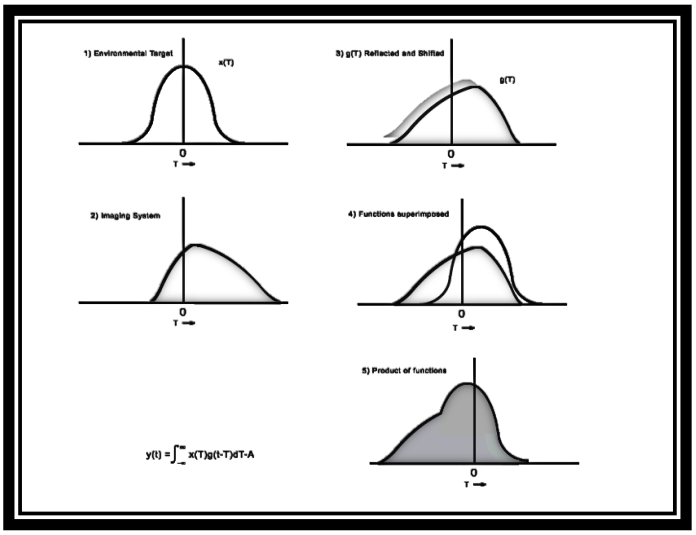


Figure 6: Example of convolution of two functions into a common function [13]

Convolutional neural networks take advantage of a two-step cascading process of convolution and Gibbs sampling of the convoluted space in order to achieve a dimensional reduction [14]. Convolution is the mathematical process to determine how much overlap two independent functions have with respect to one another. An example of the process of convolution is in message importance where y(t) is the time you would send the message and in this example it depends on message priority p(t) and available bandwidth b(t). The convolution of these features would result in the equation $y(t) = p(t) * b(t)$ or another way this can be written as $y(t) = \int_0^t p(\tau) * x(t-\tau)d\tau$. If we assign bandwidth to an inverse bell function and priority and an exponentially rising function with respect to time we come up with the result$y(t) = 2 * (t^2 + 2)$. Convolution allowed two features to be compressed into one; this

process is calculated for every permutation of the feature set until a group of convolution functions has been discovered that is equal to $\frac{n!}{(n-1)!} = n$. The convolution process produces a halving of all the features, furthermore we only pick representative functions with some random probability this is our Gibbs sampling method and allows for further dimension reduction. When this process is completed it is then fed into a simple feed forward neural network which produces some error and the weights are changed via backpropagation learning.

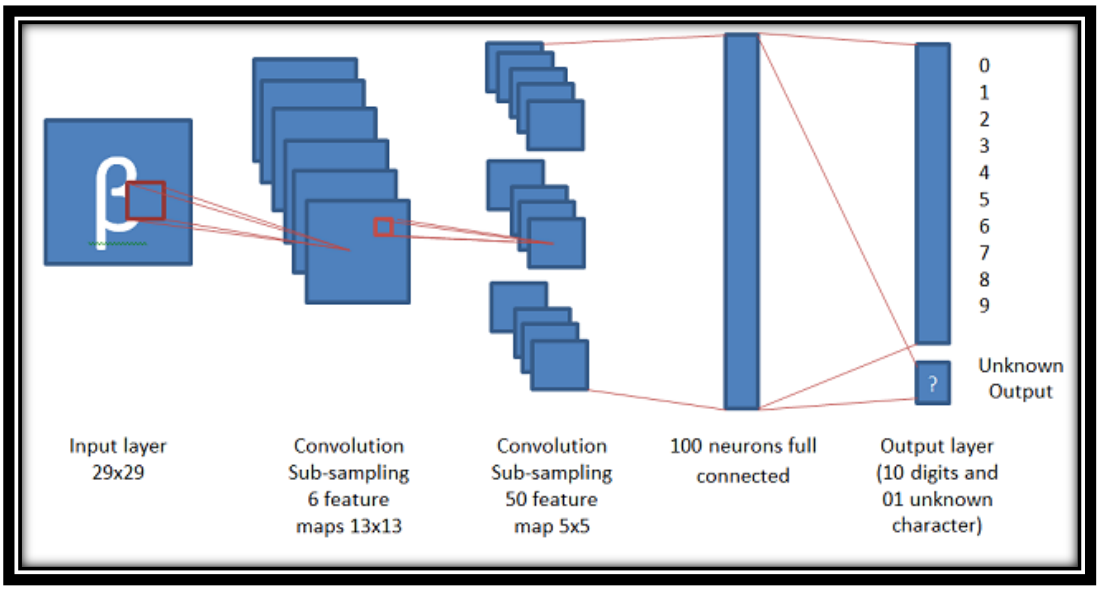


Figure 7: Convolutional Neural Network, example with image recognition [15]

Using convolution allows us to create a tunable filter for noisy information and is advantageous to fixed function filters because the filters must react to the situation at hand. Convolutional neural networks allow this transformation to occur by cascading these filters into unsupervised regions of restricted time-series features these restricted regions are then further restricted until a representative output is produced. For example two messages are waiting to be sent by the network one is a message and the other is a video, the priority of each is extremely high and they are both time critical. In a fixed function scheduling algorithm this would cause a conflict, but in the deep learning paradigm one of these two messages would have already been sent due to predictions gleaned from past experiences. Eventually patterns would be produced that can determine when military maneuvers will take place before they are requested and thus eliminating the need for CDN networks and relying more on a just in time system. There are issues with the convolutional neural network though, foremost training time is extremely lengthy and secondly it has unknown performance when applied to future instances of novel data.

4.3 Deep Belief Networks

While convolutional neural networks allow for discriminative understanding of the current situation the understanding of why is the function correctly works is missing. Deep belief networks allow additional understanding of what inputs produce specific priority requests, enabling a fine-tuning of the convolutional approach. This fine-tuning is needed as the convolutional neural networks will be thrown off by novel data or get stuck inside of a local minimum and always produce a singular output such as all text messages go first. Utilizing deep belief networks can enhance the convolutional neural networks by determining which features where important in the calculation allowing for faster and more general calculations.

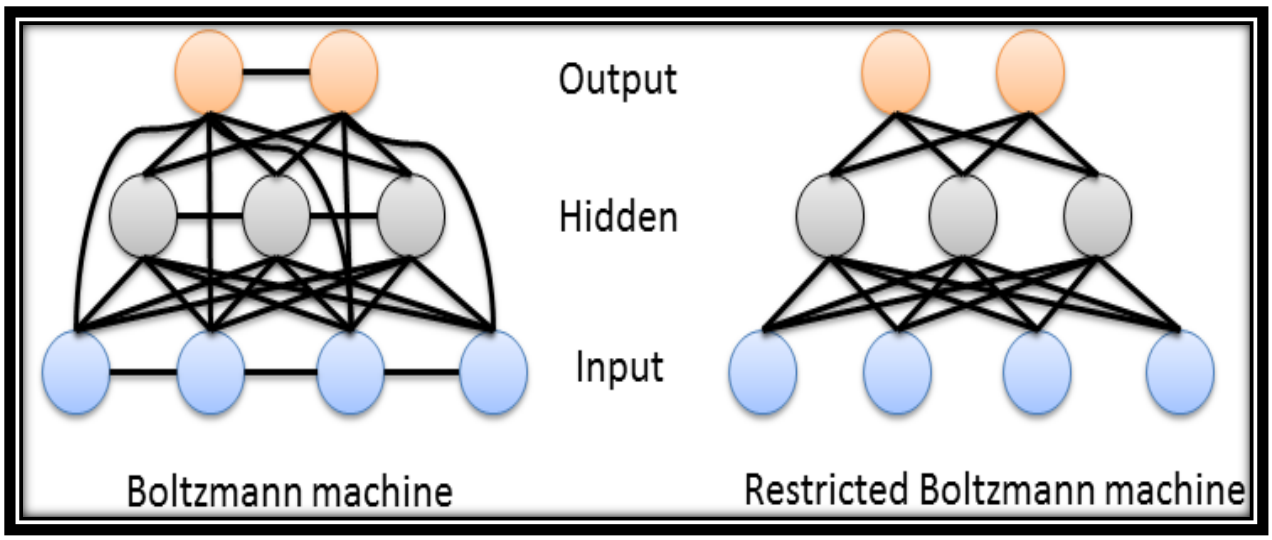


Figure 8: Boltzmann machine vs. Restricted Boltzmann machine

Deep belief networks were proposed in 2006 and are based on the deep learning concept [16]. Deep belief networks enable numerous advantages over convolutional neural networks but the biggest are faster learning and the ability to process both labeled and unlabeled data. The process of training a deep belief network is similar in concept to a convolutional neural network but the basic constituent part is a type of neural network called a Restricted Boltzmann Machine. Restricted Boltzmann Machines or RBMs are fully connected neural networks that have no connections between individual neurons in a single layer and no recurrent connections in the network. With these restrictions in place a RBM can be properly trained using a contrastive divergence algorithm which performs a stochastic gradient descent with Gibbs sampling [17]. Stochastic gradient descent is a procedure of finding the minimum of a function by following the derivative or gradient of the function in a probabilistic manner. This allows a minimization of any derivable function to be found at which point the system is considered balanced and learning complete. The Gibbs sampling procedure allows picking randomly of neurons from each layer in-lieu of calculating each neuron and extrapolating the functions of all from these few picked. In this way a RBM can be trained quickly and arrive at a minimization of the error function thus, predicting the output pattern from an input pattern.

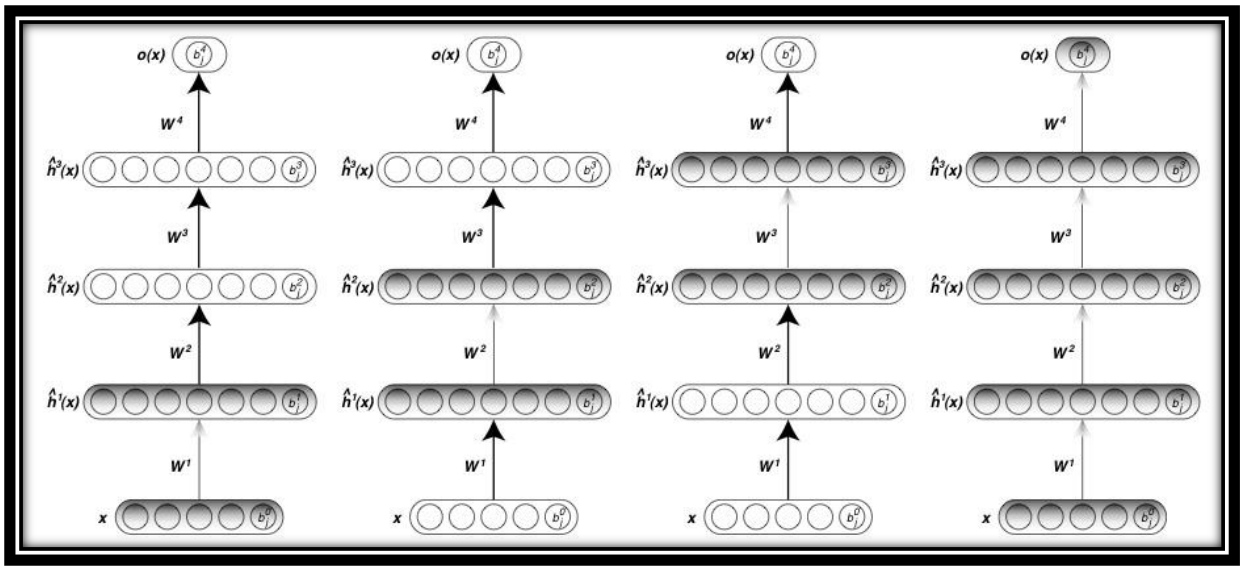


Figure 9: Deep Belief Network, learning via stacked restricted Boltzmann machine [18]

Deep belief networks work by stacking these Restricted Boltzmann Machines on top of one another to from a long chain of networks that utilize the output of the former to discover new correlations between the different features. This performs an unsupervised learning process where-in all the data goes into the algorithm with no human decision as to what is relevant or irrelevant. The deep belief network decides which features are relevant and combines them into higher features which are then used as a basis to predict the proper time to send information across the network. Training of these stacked RBMs is done layer by layer, where each neuron in each layer is calculated by minimizing entropy $E(V^0, h^0) = -[\log p\,(h^0) + \log p\,(V^0|h^0)]$ with the boundaries of this equation being determined by the Bernoulli distribution of entropy for that layer [19]. The first neuron weight which falls within this range is taken as correct and the weight is propagated forward as the correct weight for all neurons in the network. The weights are then eventually propagated backwards until a steady state is reached at which point the outputs of the top layer are used as the high level input features of the convolutional neural network.

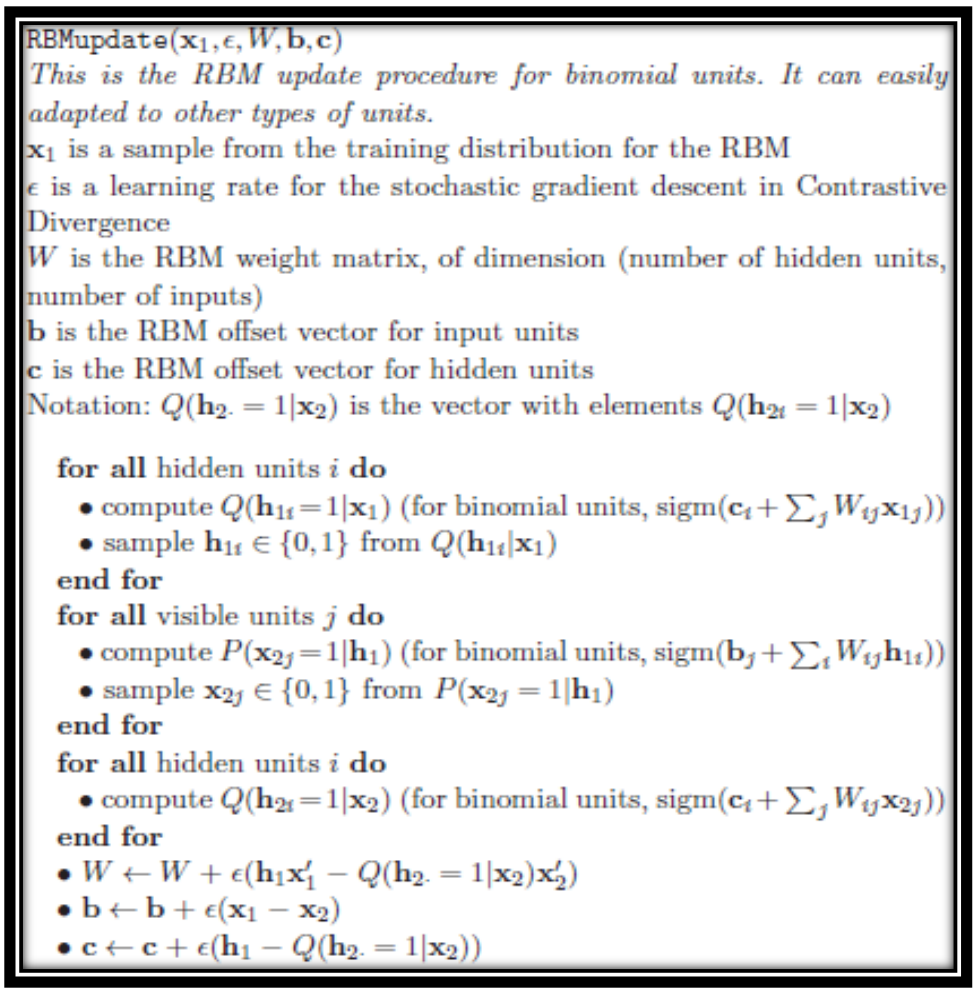
4.4 Deep Learning Applied to Load Balancing

Usage of deep learning in the field of load balancing requires a novel combination of deep learning architectures in order achieve a better understanding of the underlying patterns involved. The biggest issues with scheduling algorithms in a load balancing environment is speed and consistency and through pre-training of the deep learning algorithms both of these can be achieved with relative ease. The deep learning approach must achieve both understanding of the temporal dynamics of load balancing and produce predictions which can be introspected to understand how the algorithm came to its conclusion.

TrainUnsupervisedDBN($\widehat{P}, \epsilon, \ell, W, \mathbf{b}, \mathbf{c}$, mean_field_computation)
Train a DBN in a purely unsupervised way, with the greedy layer-wise procedure in which each added layer is trained as an RBM (e.g., by Contrastive Divergence).
$\widehat{P}$ is the input training distribution for the network
ϵ is a learning rate for the RBM training
ℓ is the number of layers to train
W^k is the weight matrix for level k, for k from 1 to ℓ
$\mathbf{b}^k$ is the visible units offset vector for RBM at level k, for k from 1 to ℓ
$\mathbf{c}^k$ is the hidden units offset vector for RBM at level k, for k from 1 to ℓ
mean_field_computation is a Boolean that is true iff training data at each additional level is obtained by a mean-field approximation instead of stochastic sampling

for $k = 1$ to ℓ **do**
  • initialize $W^k = 0$, $\mathbf{b}^k = 0$, $\mathbf{c}^k = 0$
  while not stopping criterion **do**
    • sample $\mathbf{h}^0 = \mathbf{x}$ from $\widehat{P}$
    for $i = 1$ to $k-1$ **do**
      if mean_field_computation **then**
        • assign $\mathbf{h}^i_j$ to $Q(\mathbf{h}^i_j = 1|\mathbf{h}^{i-1})$, for all elements j of $\mathbf{h}^i$
      else
        • sample $\mathbf{h}^i_j$ from $Q(\mathbf{h}^i_j|\mathbf{h}^{i-1})$, for all elements j of $\mathbf{h}^i$
      end if
    end for
    • RBMupdate($\mathbf{h}^{k-1}, \epsilon, W^k, \mathbf{b}^k, \mathbf{c}^k$) {thus providing $Q(\mathbf{h}^k|\mathbf{h}^{k-1})$ for future use}
  end while
end for

Figure 10: Unsupervised Deep learning network training (note do not initialize weights to 0, initialize randomly (-1.0, 1.0)) [20]



RBMupdate($\mathbf{x}_1, \epsilon, W, \mathbf{b}, \mathbf{c}$)

This is the RBM update procedure for binomial units. It can easily adapted to other types of units.

$\mathbf{x}_1$ is a sample from the training distribution for the RBM

ϵ is a learning rate for the stochastic gradient descent in Contrastive Divergence

W is the RBM weight matrix, of dimension (number of hidden units, number of inputs)

$\mathbf{b}$ is the RBM offset vector for input units

$\mathbf{c}$ is the RBM offset vector for hidden units

Notation: $Q(\mathbf{h}_{2\cdot} = 1|\mathbf{x}_2)$ is the vector with elements $Q(\mathbf{h}_{2i} = 1|\mathbf{x}_2)$

for all hidden units i **do**
- compute $Q(\mathbf{h}_{1i} = 1|\mathbf{x}_1)$ (for binomial units, $\text{sigm}(\mathbf{c}_i + \sum_j W_{ij}\mathbf{x}_{1j})$)
- sample $\mathbf{h}_{1i} \in \{0, 1\}$ from $Q(\mathbf{h}_{1i}|\mathbf{x}_1)$

end for

for all visible units j **do**
- compute $P(\mathbf{x}_{2j} = 1|\mathbf{h}_1)$ (for binomial units, $\text{sigm}(\mathbf{b}_j + \sum_i W_{ij}\mathbf{h}_{1i})$)
- sample $\mathbf{x}_{2j} \in \{0, 1\}$ from $P(\mathbf{x}_{2j} = 1|\mathbf{h}_1)$

end for

for all hidden units i **do**
- compute $Q(\mathbf{h}_{2i} = 1|\mathbf{x}_2)$ (for binomial units, $\text{sigm}(\mathbf{c}_i + \sum_j W_{ij}\mathbf{x}_{2j})$)

end for

- $W \leftarrow W + \epsilon(\mathbf{h}_1\mathbf{x}_1' - Q(\mathbf{h}_{2\cdot} = 1|\mathbf{x}_2)\mathbf{x}_2')$
- $\mathbf{b} \leftarrow \mathbf{b} + \epsilon(\mathbf{x}_1 - \mathbf{x}_2)$
- $\mathbf{c} \leftarrow \mathbf{c} + \epsilon(\mathbf{h}_1 - Q(\mathbf{h}_{2\cdot} = 1|\mathbf{x}_2))$

Figure 11: Restricted Boltzmann Machine weight update function [20]

The first step in creation of a generalized pattern recognizer is to understand the feature space and understand how the features represent the underlying pattern. The feature input space is massive and is a factor in the computational complexity of the algorithm. The deep learning architectures present a method to both compress and express higher order features. The first step in the algorithm is the inclusion of a deep belief network 4 layers deep, this network allows observation of the features and the ability to infer unobserved features. To achieve this each hidden layer must sample the feature space and create an unbiased understanding of the posterior distribution of the data, that is to say it must decide what features are correctly inferring the result state of the input dataset this result state is when to send the message. Using contrastive divergence algorithm this network is trained and the output which are higher dimensional features discovered in an unsupervised fashion from the inputs are directed towards the second portion of the deep learning scheduling algorithm.

After the extensive number of features has been combined to form higher level features, time must be added as an extra dimension. As each feature seen is a kind of snapshot in time and the higher level features represent some functional representation of these features it stands to reason that these functional representations would themselves varying with respect to time. Deep belief networks are not well suited for spatiotemporal networks [16], and as such are missing a critical component of creating a generalized prediction framework for load balancing. Introducing convolutional neural networks into the second layer allows extension of all the higher level features by including a temporal dimension. This will be presented to the convolutional layers which intersplice the temporal dimensions with the higher level features discovered by the deep belief networks into features containing both simultaneously.

This result of this process creates a function that when the unaltered features are added outputs an input for a traditional fully connected neural network.

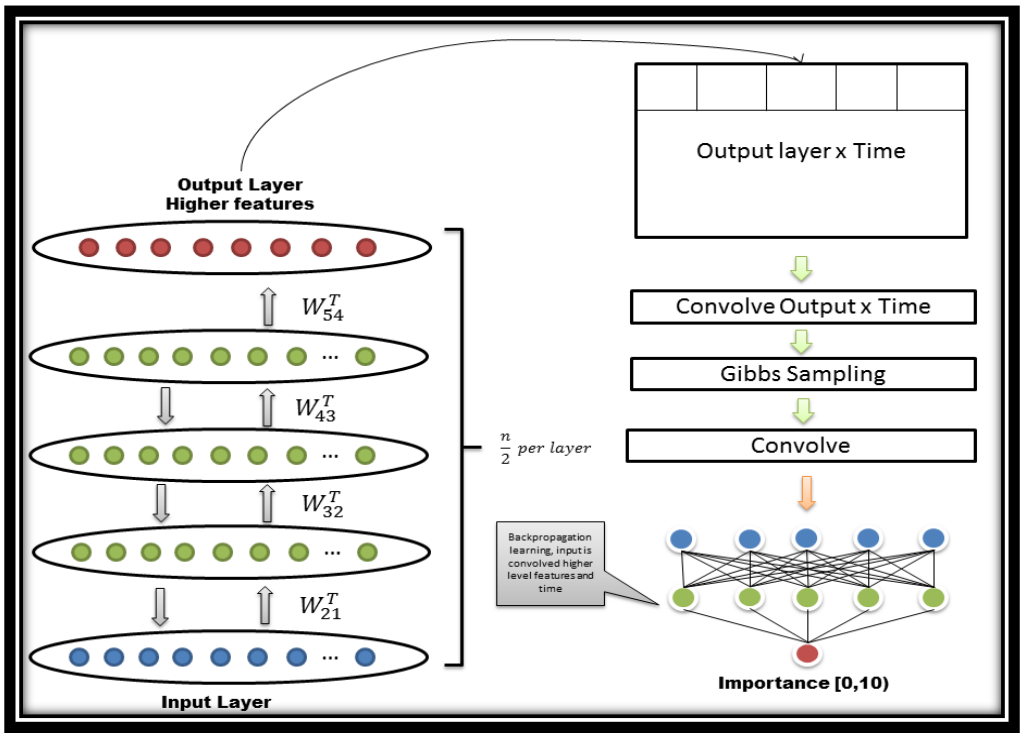


Figure 12: Scheduling Algorithm Architecture

The fully connected neural network serves as the top layer of the scheduling algorithm. This network is presented with data in any format which is transformed by the functions discovered by the previous two layers. It is then trained on the basis of back-propagation, which would be otherwise inefficient given the original feature set. This allows for maximum reusability and understanding of how each of the features interacts forming a decision function in order to derive an importance value. The importance value output of the function represents the decision by the neural network and is adjusted when a user decides that the importance assigned was incorrect. In this manner an online learning process can take place with the algorithm adjusting to varying conditions on the battlefield and thus learning how to schedule and send data needed in limited bandwidth environments.

5. Experiments & Discussion of Results

At present time, the algorithms presented in the paper have only undergone testing using a corpus of close to 6000 documents. The goal is to build a test suite for testing similar to what is done with the MNIST [21] dataset, which is a large image library that is used for rigorous testing of image identification algorithms. The primary dataset used was atmospheric documents provided by the Army Core of Engineers; the dataset contained 5512 documents where each file is a report about civilian structures, civilian environment, and military installations. The dataset was extended with erroneous information obtained from Defense Technical Information Center (DTIC) and GlobalSecurity.org, this information was to seed our dataset with misinformation and make sure the algorithm properly associated this information with a lower priority. Priority had to be added in order for the algorithms to be properly evaluated. To do this we constructed a criteria list and included specific keywords which were totaled up in each document and normalized to determine priority ranking. The priority was casted to a value between [0, 10] inclusive where the higher the value indicated greater priority. The keywords criteria list included specific locations in Afghanistan and specific names of important insurgents and GPS locations.

After the data had been generated it had to be split up into a testing and training set, this process was done with a simple 70/30 split, and each result was verified with 10 folded cross validation. The Deep belief network was compared against two alternative classification algorithms; a support vector machine using radial basis kernel, and a naïve Bayesian classifier. These two classifiers were picked because they are usually the two most commonly used; random forests classification was also identified but not included due to time constraints.

Classifier	Accuracy
Naïve Bayes	69.7149%
SVM (RBF)	72.4973%
RBM (10,000 hidden)	73.2558%
RBM (100 hidden)	74.0311%
DBN (10,000 x 10,000 hidden)	73.5874%
DBN (100 x 100 hidden)	74.5381%

Table 1: Classifier Accuracies

The results shown in table 1 show that overall the deep learning networks classified the data better than other leading classifiers. The most interesting finding was that increasing the number of hidden units on the Restricted Boltzmann Machine decreased the accuracy of the classifier. This was most likely due to the "curse of dimensionality" [5] as the number of hidden units increase the potential dimensionality separation between different data points increase and makes it more difficult for the classifier to properly construct decision planes. The deep belief network performed overall better than the Restricted Boltzmann Machine alone, but this accuracy increase is very small to justify the massive increase in computational processing required in stacking Restricted Boltzmann Machines.

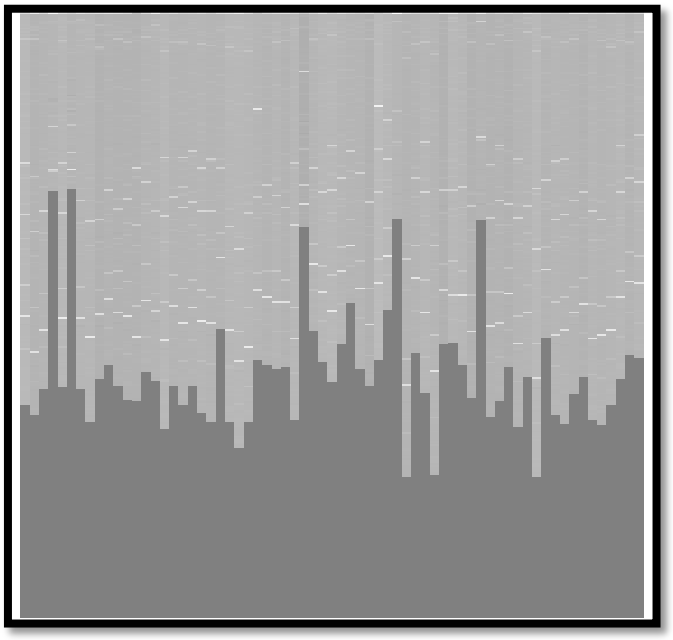


Figure 13: DBN Classification Error

The classification errors of the deep belief network are shown in figure 13 above. What is interesting about this graph is that, as the number of features increases, the accuracy decreases significantly. This is due to limited amounts of these big feature vectors, and represents unique files that the classifier has not seen enough of to properly place into any of the previous patterns it has seen. In figure 14 below, the weights of this network are visualized with the picture on the left being the first layer RBM and the second being the higher level feature

weights of the second layer. The darker the pixels, the higher the weight value and the more important the feature is towards the overall prioritization of the message, similar goes for the second layer, as you can see the DBN has decided that some features are less important at a higher level than others. There is a pattern at level two of the deep belief network on column 4; this column was found to represent the higher level feature combination of file size, tf-idf terms below frequency 5, and date signature of the file. This is interesting as the DBN is now looking for these rarer words and has learned to look for the dates and subject them to higher priority as the date approaches.

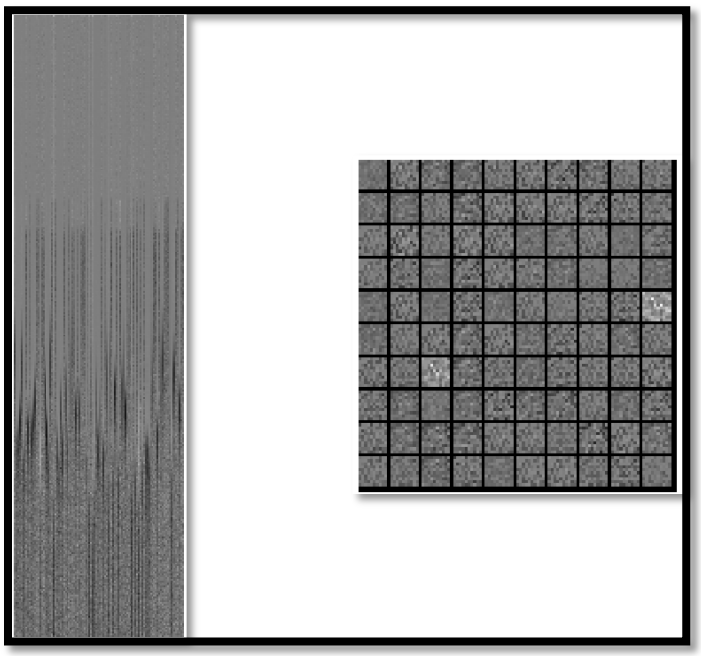


Figure 14: DBN Level 1 and 2 weights

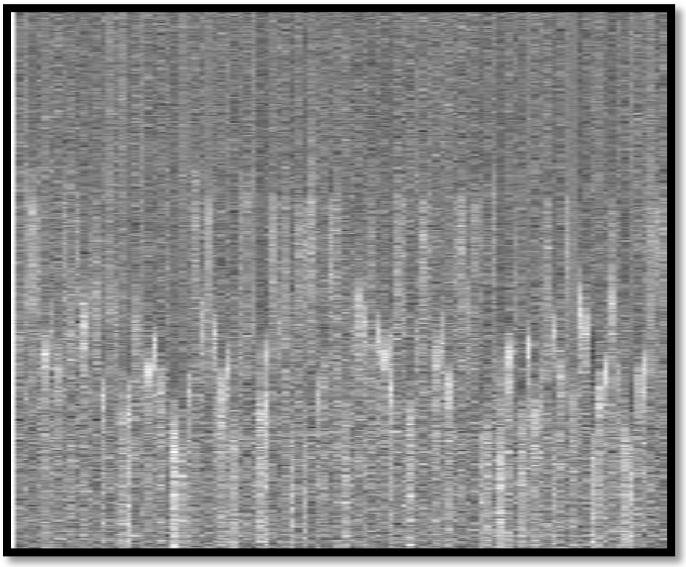


Figure 15: Difference between weights in RBM

Figure 15 demonstrates image subtraction of the learned weights of the different Restricted Boltzmann machines for all the deep belief networks. This image shows that indeed that the larger networks are having issues with finding patterns in the data due to the higher dimensionality. This higher dimensionality works against the user as the accuracy will continue to diverge with increased training. With this in mind combined with the other results obtained, our recommendations are as follows: if the problem is one of binary classification, task utilization of a deep belief network would be considered computational extravagant. One would be better suited to use a simpler Restricted Boltzmann Machine in order to solve this problem. In the case where the problem is a multi-class classification problem broken up into discrete ranks, a deep belief network works better than competing algorithms and it would be in the user's best interest to minimize the number of layers and hidden units in order to achieve the highest accuracy possible. It is in this that the deep learning paradigm can be utilized effectively to correctly and quickly classify network transmission messages.

6. Conclusion

The use of CDNs has greatly enhanced the capabilities of large networks; enabling higher degrees of availability and performance. But on many military networks, much of the network is hampered by DIL connectivity which hampers the transmission of vital resources required for mission critical operations. To help alleviate this constraint, this paper delved into the concept of anticipating content demand before it is required, effectively broadening the window of opportunity that data can be transmitted. Conceptually, this is done by using machine learning to build up a knowledge base of what is important and what is not based on content features. The machine learning algorithms accomplish this internally by using advanced neural networks to evaluate content feature sets against each other, identifying the features that make the largest impact on the overall priority of the content. When the overall priority of the content is established, the information may then be queued up for transmission to the CDN based on its new priority.

We have proven that the deep belief network architecture is optimal when given enough data and when given a discrete ranked problem, but suboptimal when given a binary constrained dataset. We discovered it to be in the best interest of the users of deep learning networks to minimize the number of hidden nodes in their networks in order to increase accuracy and decrease runtime of the algorithm. With these modifications our deep belief network paradigm outperformed similar classification schemes and proved to be more accurate and without the needless unsupervised feature detection needed by the other classification algorithms. In this way a faster, more accurate machine learning method can be constructed for use in military applications that requires less fine-tuning from the user and has the ability to scale to any problem space.

7. References

[1] ICT, "ICT Facts and Figures," 2011. [Online]. Available: http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf. [Accessed 6 2 2013].

[2] Internet World Stats, "Internet Users in the World," 30 June 2012. [Online]. Available: http://www.internetworldstats.com/stats.htm. [Accessed 6 February 2006].

[3] D. Webb, "On the Definition of a Space Weapon," 2005. [Online]. Available: http://praxis.leedsmet.ac.uk/praxis/documents/space_weapons.pdf. [Accessed 6 2 2013].

[4] S. J. Russel and P. Norvig, Artificial Intelligence: A Modern Approach, Upper Saddle River: Prentice Hall, 2003.

[5] C. Rhode, "Intro Neural Networks," 1 Janurary 2010. [Online]. Available: http://lowercolumbia.edu/students/academics/facultypages/rhode-cary/intro-neural-net.htm. [Accessed 13 Feburary 2013].

[6] D. Hebb, The Organization of Behavior, New York, 1949.

[7] W. McCulloch and P. Walter, "A Logical Calculus of Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophyscis,* vol. 5, no. 4, pp. 115-133, 1943.

[8] K. Fukushima, "Cognitron: A self organizing multilayered Neural Network," *Biological Cybernetics,* vol. 20, no. 3-4, pp. 121-136, 1975.

[9] Statistics 4 u, "www.statistics4u.com," Statistics 4 u, 1 January 2008. [Online]. Available: http://www.statistics4u.com/fundstat_eng/img/hl_backprop.png. [Accessed 4 Feburary 2013].

[10] P. Webos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Harvard University, 1974.

[11] K. Shachar, S. Rosset and C. Perlich, "Leakage in data mining: formulation, detection, and avoidance," *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining,* vol. 1, no. 1, pp. 556-563, 2011.

[12] A. Waibel, T. Hanazawa, G. Hinton, K. Shinkano and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transcactions of acoustic, speech and signal processing,* vol. 37, no. 1, pp. 328-339, 1989.

[13] USDA, "ars.usda.gov," United States Department of Agriculture, 18 August 2010. [Online]. Available: http://www.ars.usda.gov/Research/docs.htm?docid=9124&page=4. [Accessed 4 Feburary 2013].

[14] H. O. Simon, Neural Networks and Learning Machines, Pearson Education, 2008.

[15] Vietdungiitb, "www.codeproject.com," Code Project, 31 May 2012. [Online]. Available: http://www.codeproject.com/Articles/376798/Large-pattern-recognition-system-using-multi-neura. [Accessed 4 Feburary 2013].

[16] G. Hinton, S. Osindero and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation,* vol. 18, no. 1, pp. 1527-1554, 2006.

[17] M. A. Carreira-Perpinan and G. Hinton, "On contrastive divergence learning," in *Artifical Intelligence and Statistics*, 2005.

[18] H. Larochelle, "http://www.dmi.usherb.ca/~larocheh/index_en.html," Hugo Larochelle, 12 March 2012. [Online]. Available: http://www.dmi.usherb.ca/~larocheh/images/deep_learning.jpg.

[Accessed 2013 6 Feburary].

[19] G. Hinton, S. Osindero and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation,* vol. 18, no. 7, pp. 1527-1554, 2006.

[20] Y. Bengio, Learning Deep Architectures for AI, 2009.