Karma Capsule Network 1.0 (KCN)

a distributed trust-less Peer-Peer AI Network of transferable Values
created by dynamically reinforcing capsNet with State & Rewards from the environment.

USPTO Copyrighted & Patent Pending to: Maya Suresh Kannan Balabisegan
(aka Suresh Kannan)

Background: This is the second version of the karmaCapsuleNetwork white paper based on the paper originally published & copyrighted on Nov 17, 2018, PTP Framework published & copyrighted in 2007, Enterprise Integration Dashboard May 29 2008 and EAstir Published & Copyrighted on October 5, 2014.

ABSTRACT:

karma Capsule Network is a trust-less, distributed, Peer-Peer AI network of transferable values. It is created by a technique of dynamically reinforcing a CapsNet by the state and intuitions(rewards) from the environment on which the network is spread across. The outcome of the KCN is transferable and tradable tensor values of entities("Things"). It solves the problems of dynamically tracking and generating values from Spatio-Temporal elements making those values transferable to other peers in a trust-less network. It is a distributed Artificial Neural Network(ANN) network based on the Philosophy of a "Thing"(an entity) and its "karma" a State tensor that represents two or more of the nine Characteristics as evidenced from Aristotle's nominalism(Quantity, Quality, Place, Time, Action, Passion, Situation, Habit, and Relation). The foundational ANN principles of the KCN is derived from capsNet , with modifications for reinforced learning and density derived from Seven Chakra energy center principles. The outcomes from each node(a computer, an electronic device, a machine or an edge device, a Gateway to another network, etc) of the KCN are karmaCapsules. Each karma Capsule is an entity of tensor "Values". The value could represent any "Thing" of some value, so the KCN shall be called an Internet of Values.

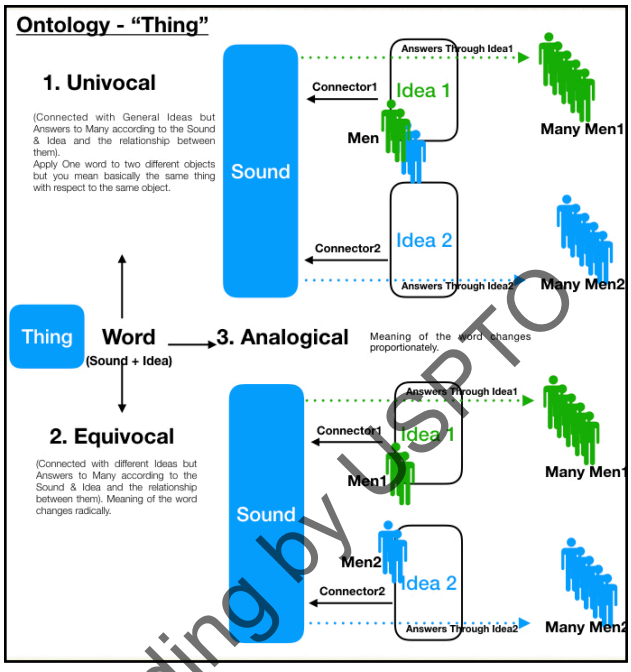
PREFACE:

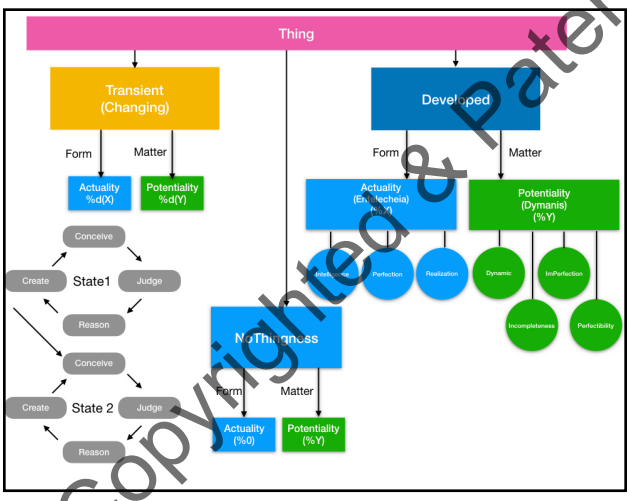
In the current Internet of Things ecosystems, most actors are not interested in the one time Naked-Data from the edge, but the “Outcomes" in the form of “Intelligence”, “Values”, and “Services" they can provide on a continuous basis. Karma Capsule Network is a distributed Artificial Intelligence Neural Network model of outcomes whose source of inputs shall be a human, a plant, an animal, a robot, an edge device, a machine, a mobile vehicle or a computer system. These outcomes are an encapsulation of Data, Actions, characteristics, and policies and are called “KarmaCapsules”. Using KCN, organizations, communities, government, and consortiums can build value-oriented networks to Monetize their cyber-physical assets. These networks of Capsules shall be used in real-time by the user applications and computing platforms to produce useful services & products. Each one of the nodes in the network shall possess one or many karmaCapsules that encapsulating the secured “data” with the immutable “Code”(Artificial intelligence) for various past & future “state” scenarios..

Karma Capsule as a “value” vehicle:

Since existence, humanity always was longing for Acquiring or possessing Wealth. Wealth is a relative and perceived value of something with respect to something else. The thing can be a physical tangible item like gold, silver, diamond, vehicle or intangible items like shares, bonds, etc., or a utility service or a solution or a product that can solve some a problem for a business or an individual. Not everything is an asset but every asset is derived as a perceived value from one or more “things”. It may sound like the “Thing” be real having an ideal state. But in reality, Things are what they are because we named them that way(Aristotle’s nominalism). So, the classification is based on Names, not Ideals.

Look at individual things, observer them in relation to each other and find a bunch of things that resemble each other and put them all in the same class give them the Name. This is the foundational philosophy of KCN. KCN is a hybrid system of symbolic, sub-symbolic and Logic. The LifeCycle of transient things from its initial to the creation of the first capsule is 1. Conceive(Idea), 2. Judge, 3. Reason and 4. Create. In the first step, we create a simple view of things that we see or feel(perceive) based on what is and keep that as some kind of vector memory. This is the first neural action. At this time, there exist no additional qualifying factors that inherently build a clear image in our minds. The second step is the judgment which is essentially joining different ideas(conceptions) together where it affirms or denies each other. for example Each, Square and Circle. It might judge earth is square or earth is Circle or Each is not square or earth is not a circle. The brain only makes judgments against each other; not on its own. The next is the Reason. The reasoning is about making a new judgment from the previous set of judgments of the same thing from different ideas. This is the Level-3 Capsule.



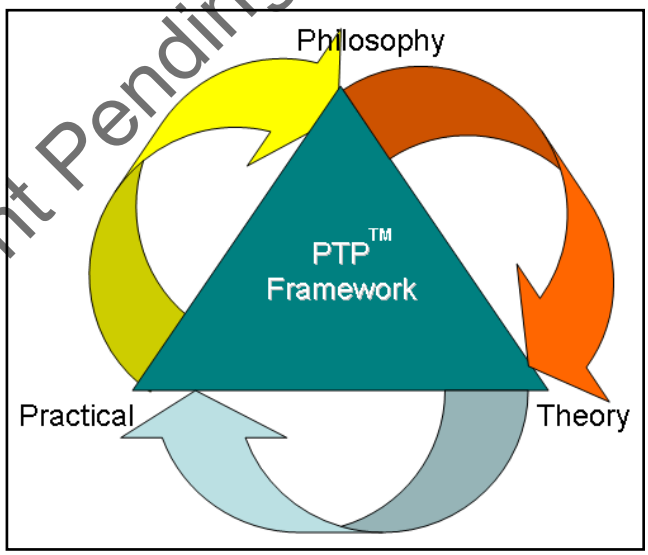


And finally, the Creation of a thing extended(final thing) is done through various operations ("Methods") to make it the final things of some value. These methods are agent algorithms that make use of the vectorial relationship between the various different things that they learn from the environment and fed back as Intuitions. This makes the thing capsuled as transient in nature

not only with respect to the local environment but also the cosmic environment where other “final Things”, “Things Extended” ie fully developed karma Capsules exists. So, essentially, even a fully developed karma Capsule(a thing of value), is still in the transient state because it is part of the cosmic system(a network) where the relationships of things denoted by various tensors in space & time keep changing. The potentiality is the future state within a limit(a—>p) that the fully developed karmaCapsule still will continue as a transient element to acquire its new Value. This is can be easily correlated to the Stock Market where the price of a specific share(a fully developed karmaCapsule) changes depending on other ecosystem parameters and the other shares. Methods (Operations) and its corresponding algorithms should adhere to the higher rules as suggested by Arnold Antoine.

A. Definitions:

1. Do not leave any “Thing” without defining them, whether they are Obscure or equivocal.
2. Define only he “Things” that are perfectly well known or already defined.



B. Axioms:

3. Demand Axioms for “Things” that are perfectly evident.
4. Accept evidence only if it required slight attention to the recognition of its truth. ie you cannot accept something as evidence for the existence of a thing, it requires a lot of attention to its truth.

C. Demonstrations:

5. Accept a thing only if they are clearly Defined or axioms granted or have practically demonstrated already.

In 2007, I published a framework for the human life cycle with respect to the things in the world called PTP Framework2. Anything that can be Philosophically well understood and internalized

can be theoretically well expressed in one or many ways. Anything that can be well expressed theoretically can be implemented successfully in one or many ways.

Anything that is practically implemented successfully gives birth to new philosophies. As introduced by Aristotle's scientific taxonomy, they correspond to Philosophy, genus, species, and Individual. In mathematical language, the genus is a theory and species are models and the individual instances are elements. The theory is a set of true propositions called theorems which can be generated by a bunch of axioms and the corresponding rules. A Model is something that is defined and exists between the Theory and Practical in the PTP framework. In a sense, a model is a particular implementation of a theory, in Aristotle's terminology, it can be the "Thing extended". ie when a set of practical elements where all the operations are defined and all the propositions are true.

6. Always avoid equivocation of things and substitute them with the definitions that restrict and explain their meaning.

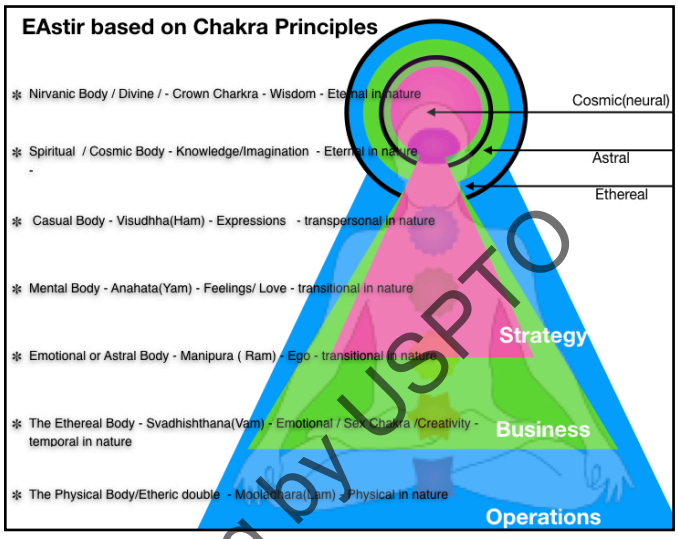
D. Rules of Methods

7. Treat things in their natural order as much possible. Ordering can be done by starting from the most general and simple thing and explain everything which belongs to the nature of the genus before passing to the particular species.

8. Divide every genus into possible species, every whole into its parts, evert difficulty into its cases. Such division shall be attempted as far as possible.

EAstir, Chakra Principle and KCN: In 2014, I published a paper on EAstir, A Business Anatomy, Physiology & Neurology Analytics Engine based on the Seven Chakra Principles and various human bodies, applied to business informatics. In KCN, I took the EAstir into the Cosmic Level applying it universally beyond business to define any Thing under this principle. There are Seven Bodies of Human:

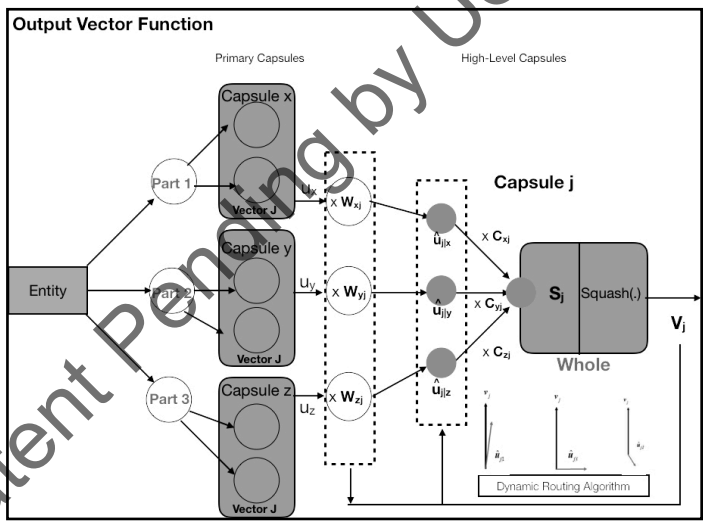
1. The Physical Body/Etheric double - Mooladhara(Lam), is Temporal in nature and the Base Chakra. 2. The Ethereal Body - Svadhishthana(Vam) - Emotional / Sex Chakra /Creativity - temporal in nature. 3. Emotional or Astral Body - Manipura (Ram) - Ego - transitional in nature. 4.Mental Body - Anahata(Yam) - Feelings/ Love - transitional in nature, 5. Casual Body - Visudhha(Ham) - Expressions - transpersonal in nature, 6. Spiritual / Cosmic Body — Ajna(Om) Knowledge - eternal in nature - Imagination, 7. Nirvanic Body / Divine / - Sahasrara - Crown Chakra - Wisdom - eternal in natures



Thoughts are tensors of activities. Most of the sources in the world are subjective in nature. The idea is to create ideal objects out of these subjects by learning & building nine characteristics around these subjects. So, KCN is a Symbolic AI using Categorical grammar. As presented by Emmanuel Kant's transcendental idealism, we shall aim to give a basis for objectivity in terms of subjectivity. It is a kind of 'realism' because it assumes objects exist independently of our perceptions. KCN is designed to create objects of these tensors through encapsulation of all the activities passed through a neural replica of all the layers of chakra in the human body. We are dangerously accustomed to satisfy ourselves with Words to imagine a thing but all they know is arbitrary names in the mind with no clear idea of the thing. Every Idea takes its origin from sense. but all the ideas in our sense are not in the same form in the mind. That's the ideal difference depicted in the seven chakras. The sequential reasoning is very good at giving signals that do light to give better raw direct Intuition. KCN tends to solve the problem of subjectivity, reality, and objectivity by reinforcing the intuitions from the cosmic (the external world) to the layers of the system to give a clear image of what the thing that exists in the external world, its states, actions, and variables.

CapsNet

Humans develop a normal pattern of neural vector activities in their brain when they look or feel certain things in the external world. For example, If I taste a wine, I have developed a brain pattern that will be applied back from memory every time i see a white wine. That alone is the reality. There is no such thing as a special quailer. Mental State is not an internal state. It is a hypothetical external state that's being used to refer to an internal brain state via normal "Causation". We refer to activity vectors in our brains by describing their normal causes and normal effects. That's why the words we use to describe sensations and feelings are the words that we refer to things or activities in the external world. We see the world in the form of entities and these entities are going to have properties.



CapsNet is a vector-based convolutional network that takes into account not just the views of an entity from one viewpoint but from many viewpoints. So, essentially the same thing can be captured differently based on the viewpoints. The use of parameter vectors, or pose matrices, allows Capsule Networks to recognize objects regardless of the viewpoint from which they are viewed.

Furthermore, Capsule Networks are moderately robust to small affine transformations of the data. This helps in clearly capturing, building and tracking a "Thing" and its value with respect to all the Nine characteristics from Aristotle. Convolutional capsules extend the sharing of knowledge across locations to include knowledge about the part-whole relationships that characterize a familiar shape.

Each capsule represents the presence and the instantiation parameters of a multi-dimensional entity that the capsule detects. In the routing process, the capsule detects a particular type of object or object-part. So, essentially a capsule outputs two things, 1. The probability that an

object of that type is present and 2. The generalization of the object(value) including many characteristics that fall under the broader nine characters.

Neural nets typically use simple non-linearities in which a non-linear function is applied to the scalar output of a linear filter. They may also use softmax non-linearities that convert a whole vector of logits into a vector of probabilities. Capsules use a much more complicated non-linearity that converts the whole set of activation probabilities and poses of the capsules in one layer into the activation probabilities and poses of capsules in the next layer.

For simplicity, we have taken a vector(1st-degree tensor). A capsule network consists of several layers of capsules. The set of capsules in layer L is denoted as L. Each capsule has a n x n pose matrix, M, and an activation probability, a. These are like the activities in a standard neural net: they depend on the current input and are not stored. In between each capsule i in layer L and each capsule j in layer L + 1 is a n x n trainable transformation matrix, Wij .

$$V_j = \frac{\|S_j\|^2}{1 + \|S_j\|^2} \times \frac{S_j}{\|S_j\|} \quad \text{.....eq(1)}$$

where v_j is the vector output of capsule j and s_j is its total input.

$$C_{ij} = \frac{exp\,(b_{ij})}{\sum_k exp\,(b_{ik})} \quad \text{.....eq(3)}$$

where the c_{ij} are coupling coefficients that are determined by the iterative dynamic routing process.

$$S_j = \sum_i C_{ij} \hat{u}_{j|i} \qquad \hat{u}_{ij} = W_{ij} \times u_i \quad \text{.....eq(2)}$$

S_j is a weighted sum over all "prediction vectors"

These Wijs (and two learned biases per capsule for a 4x4 matrix) are the only stored parameters and they are learned discriminatively. The pose matrix of capsule i is transformed by Wij to cast a vote Vij = MiWij for the pose matrix of capsule j. The poses and activations of all the capsules in layer L + 1 are calculated by using a non-linear routing procedure which gets as input Vij and ai for all i 2 L; j 2 L+1.

The non-linear procedure is a version of the Expectation-Maximization procedure. It iteratively adjusts the means, variances, and activation probabilities of the capsules in layer L + 1 and the assignment probabilities between all i 2 L; j 2 L+1. In appendix 1, we give a gentle intuitive introduction to routing-by-agreement and describe in detail how it relates to the EM algorithm for fitting a mixture of Gaussians.

Let's consider Vj, the length of the output vector of a capsule is the probability that the entity represented by a specific capsule in layer L. So, a non-linear “squashing" function is used to ensure that short vectors get shrunk to almost zero length and long vectors get shrunk to a length slightly below 1. Discriminative learning can make use of this non-linearity.

For all but the first layer of capsules, the total input to a capsule Sj is a weighted sum over all “prediction vectors” ^uilj from the capsules in the layer below and is produced by multiplying the output ui of a capsule in the layer below by a weight matrix Wij . Coupling coefficient C is learnt through routing function. The coupling coefficients between capsule i and all the capsules in the layer above sum to 1 and are determined by a “routing softmax” whose initial logits bij are the log prior probabilities that capsule i.

Procedure 1 Routing algorithm.

1: **procedure** ROUTING($\hat{u}_{j|i}, r, l$)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` computes Eq. 3
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` computes Eq. 1
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
return $\mathbf{v}_j$

The coupling coefficients between capsule i and all the capsules in the layer above sum to 1 and are determined by a “routing softmax” whose initial logits bij are the log prior probabilities that capsule i should be coupled to capsule j. The log priors can be learned discriminatively at the same time as all the other weights. They depend on the location and type of the two capsules but not on the current input source. The initial coupling coefficients are then iteratively refined by measuring the agreement between the current output vj of each capsule, j, in the layer above and the prediction ^uilj made by capsule i. The agreement is simply the scalar product aij = vj x ^uilj. This agreement is treated as if it was a log-likelihood and is added to the initial logit, bij before computing the new values for all the coupling coefficients linking capsule i to higher-level capsules.

Margin Loss: Since the network produces vector or matrix outputs, existing loss functions cannot be simply reused. However, they can often be adapted and sometimes leverage the additional data, as can be seen in the reconstruction loss. Still, using the CapsNet on a new dataset will often require a new loss function as well.

Margin Loss

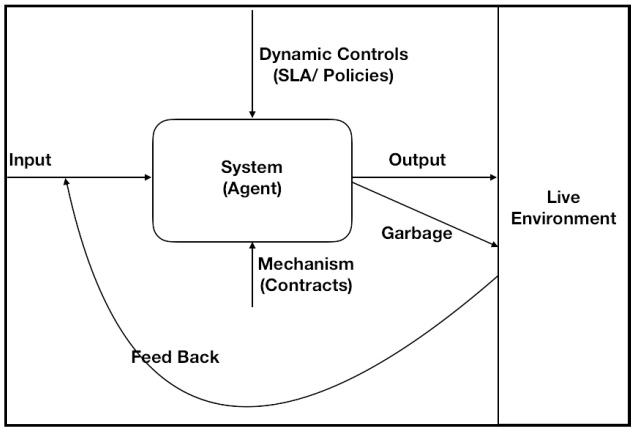
$$L_k = T_k \max(0 ; m^+ - || v_k ||)^2 + \lambda(1 - T_k) \max(0; || v_k || - m^-)^2$$

..... **eq(4)**

Tk = 1 iff a digit of class k is present3 and m+ = 0:9 and m- = 0:1. The down-weighting of the loss for absent digit classes stops the initial learning from shrinking the lengths of the activity vectors of all the digit capsules. here λ = 0:5. The total loss is simply the sum of the losses of all digit capsules.

Reinforcement Learning

Things Model is made of four components, 1. Thing, 2. Manner (Mode) and 3. Thing Modified and 4. Relativity. As a human, all we conceive is represented in our mind, either as a thing, or a manner of a thing or as a thing modified. Thing is subsisting by itself and as the subject of all which we conceive of it. It can otherwise be called “Substance”. 2. Mode (or manner of a thing), or attribute, or quality, that which, being conceived in the thing, and as not able to subsist without it, determines it to be of a certain fashion, and to be so denominated. 3. The Thing Modified when the substance, as determined in a certain manner or mode. 4. Relativity is the Relationship between the Mode and the Thing. With the tensor representation of the entity(thing), the caps net could easily create thing modifies and establishes a various relationship between the characteristics and the thing and with dynamic routing, we can easily trace back in the history to fetch the result of such relationships at specific time-space within the Co-Ordinate Frame and Existence Probability.



A thing in a certain state may represent itself in another state; The unique design of the ANN part of karmaCapsule Network is its ability to combine two ANN, the CapsNet and Reinforced Learning(RL). Reinforced learning is a type of machine learning model that an Agent(an

algorithm) that takes action within an environment. The environment returns two types of information to the agent, Reward(a vector value carrying the quantitative information within the specific time(t) and State, a change in the environment with respect to the action and its potential impact at t+1, as two feedback elements. Thoughts are vectors of activities. Such thoughts happen in the environment and in order to remember to feedback to the karma system, these thoughts should be quantified and fed back to the system as Intuition.

WE LIVE IN DERIVATIVE WORLD. A derivative is an instantaneous rate of change of Something. During the Industrial Age with Static environment (Market Place) the general philosophy of the business society was, If you keep giving what you are giving, you will keep getting what you are getting" – So, don't change any damn thing. Most of the decisions were made based on this assumption and unfortunately, the practice goes extended well beyond the industrial age into the information age. But as the industrial age matured, the environment started to change quite rapidly, breaking the dynamics of such an assumption. So, the market started understanding that If you keep giving what you are giving, you will NOT keep getting what you are getting because your system is not standalone, it is an integral part of a larger system (cosmic environment) – ie your enterprise is an extension of the world enterprise" – So, be ready to deal with "CHANGE"

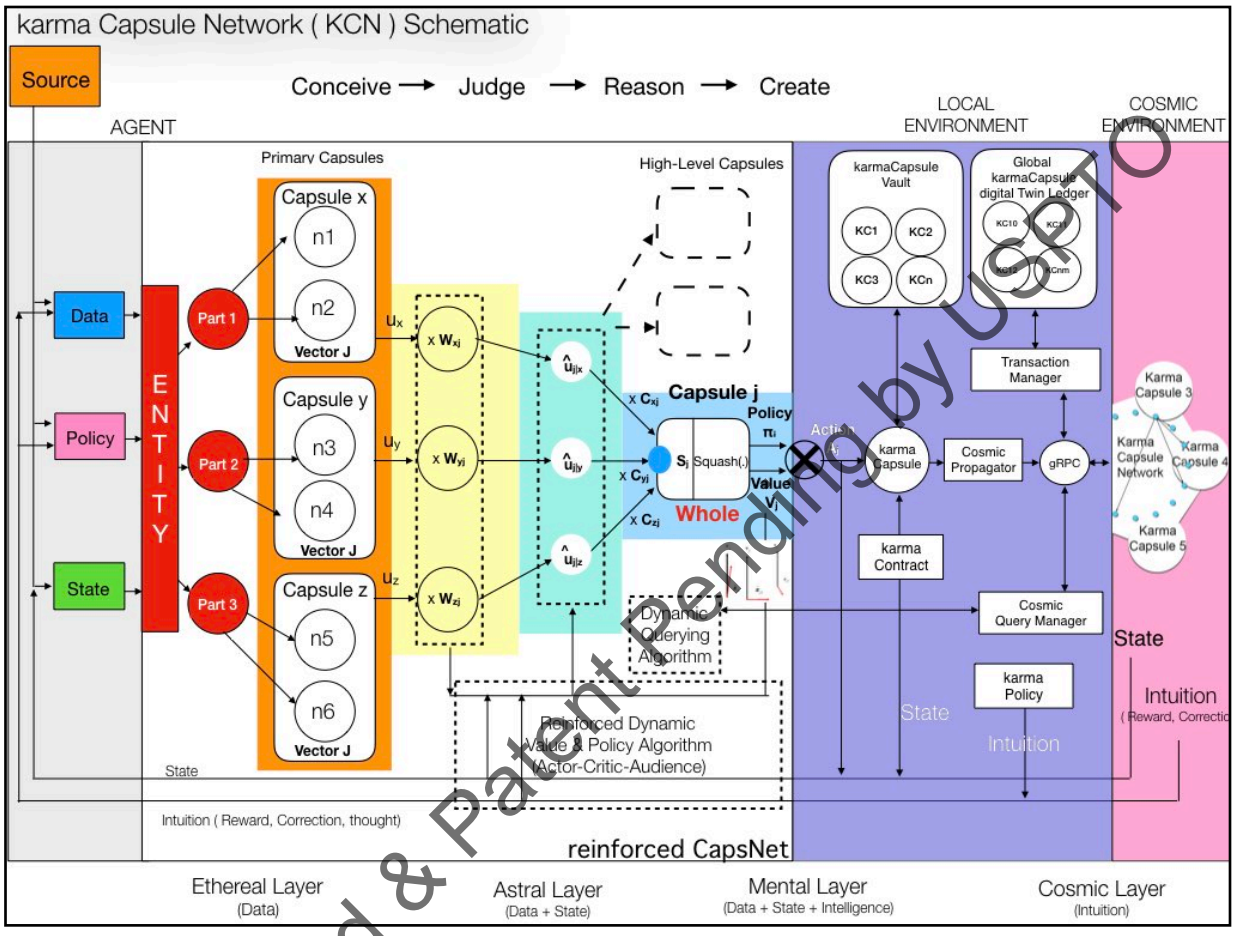
So, we are forced to design systems for the Information (and Service) Age with Dynamic Market Place(Cosmic Environment). Change Means Transform. So, we have to deal with Transformation for solving current generation business-technology problems. But how to Handle/Measure Transformation(Change)? We get an answer from Calculus. Ie using Derivative If x is the item, then derivative f'x = d(y)/d(x).While a single Business or Technology component(Source) Say "x" represents "Order" and solves the problem of a function say "Order Management" represented by F(x) = x2 + 2x + c, Then the "rate of change of order" is represented by (f'x)=2x+2. So, we are very clear from the above example, we should be dealing with "x" to deal with "Order" and f'x to deal with "Order transformation". But wait…Isn't the Enterprise a one big State Machine having many Business state machines working together through the transition? How are we going to handle it? The answer again can be found from "Integral Calculus". Integration is the necessary thing to get business transformation done correctly. So, we should be able to Integrate all differential components and its Derivatives to get

the integration within a limit. Ie from the aggregation of the rate of change of “Orders” in various time frames Situations, controls, we should be able to model the “Order Management” Process. Such model can only be implemented through a Feedback mechanism over the core Artificial Intelligence system that keeps learning every differential steps of The Thing in the environment and back-propagate through the neural to learn/update the previous states of the neurons in accordance with the changes and then integrate back to the new version of the Thing. While the built-in recursive propagation mechanism of Capsnet using the “Chain Rule” of calculus provides the mechanism of traversing backward to capsules and activate their previous value neurons from the associative memory (where it was stored) learn their previous states and quickly traverse back to the current layer and learn to update the current state of the final karma capsule. The specific final karmaCapsule’s state and value change with respect to the environment (say marketplace) cannot be learned simply using the CapsNet recursive model alone. It needs the help of a feedback mechanism to feed the new Data + Tensors(Characteristics) of the entity. That's where the Reinforcement Learning comes as a savior for karma Capsule Network.

KCN is the great tool in applying the Past memories applied to the current Scenario to influence, without having to go through crazy backpropagation as done with the RNN with many many back steps off Recurrent Net. It takes “having what happened some time ago in an associative memory using the current state to access the associative memory to get what happened long term ago, reactivate it. So, you do the learning not by back-propagating to time, but reactivating this back memory. KCN keeps these past memories on separate neuro Modules. Every Synopsis of the caps net has multiple time scales and so, you have a synopsis that learns slowly it; it also has the compound that learns fast. In AI-based on Symbolism, you cannot try to explain the reasoning without the Intuition. The karma Capsule network brings that desperately needed intuition through the reinforcement from the environment both as Thought vectors of activities as well as from overall cumulative actions that happen in the environment.

How Karma Capsules would work(an abstract description for simplicity):

karmaCapsule is a group of neurons whose activity vector represents the instantiation parameters



of a specific type of "Thing". So, the output of a karmaCapsule always presents various specific characteristics, under Nine characteristics under Aristotle's nominalism. So, within every capsule, there shall be various parameters like position, orientation, velocity, hue, time, volume, price, demand factor, etc.

- Design a karmaCapsule Asset Policy: The KCN network understands the neural pattern of activity in the entire karmaCapsuleNetwork and learns the causes and effects of certain Transactions on Certain "Things" within its network and then creates a "karma Intuition". The asset is defined by normal causes and effects. One would describe how an Asset would look like. Basically, a description of the normal consequences of any activity(transaction) Vector. When an abnormal consequence happens on any Specific, or bunch of things in the form of high-value transactions or retention of certain things(with no or very limited transaction for a quite some time) happens these are sensed as Asset activities and creates "karma Intuition" for those things and related activities. These

karma Intuitions essential becomes the Input for the Inner KarmaNetwork. Within the Capsule, there are a bunch of neurons, activities of these neurons represent different properties of the same “Thing” and better it would be “One Thing”.

- Create Initial policy for a set of data from a source towards an entity(Thing) into the CapsNet.
- Create Capsules(a bunch of intermediate neurons) with algorithms.
- Feedback the entire karmaCapsule into the network.
- This creates 1st output karmaCapsule with the value and optimal policy vectors(V and π).
- Feedback the state and the intuition 1st Output Capsule into the network using the Dynamic value & policy algorithm. a Routing Algorithm to route the Input “thing” to the best possible neuron that can make sense of it and knows how to deal with it.
- The capsule is activated only if the transformed parts of the object coming from the layer below, match each other. A capsule with no Child is turned OFF.
- The output from the last layer will be used to create a fully developed karma capsule as a binary object with policies and actions embedded in itself. This karma Capsule is sent to the Vault in the local environment (local virtual machine environment).
- The propagation mechanism will spread the word about the newly created capsule to the cosmic(global) network to all the nodes through gRPC protocol. Each node shall update its local inventory(ledger) with this newly created capsule. Every karma Capsule Digital Twin will be sent as a lightweight object across the network for the nodes to collect and store in the local ledger(Global karmaCapsule digital Twin Ledger. These copies of the capsules will not have the private or any sensitive information available to be retrieved.
- In order to retrieve the transaction information and any private detail, the node should use the cosmic query manager which will talk to the specific karma capsules owner node through the cosmic layer(gRPC communication).

karmaCapsule Content:

Every final capsule from the ANN will create a binary object, consists of Private Area("Tensors of Values", "Policy", tensor of States) and Public Area(smart Contract, public information).

Types of Nodes:

1. Shallow Node - Makes a very shallow network. Ie first layer only(with simple data and state). This node not necessarily available all the time. So, any edge device with weak compute / memory or energy can participate as a shallow node.
2. karmaCapsule Synthesizer node: The synthesizer node is full, a deep node with many layers and capable of creating and hosting many karmaCapsules.
3. Trader Node: Who would only involve in the transactions of trading the values.
4. Broker Node: Broker nodes would only be involved in brokering actives of the transactions/trading of the values.
5. Governance / Monitor Nodes: For governance / legal enforcement/ SEC/ Commodities commissions etc.

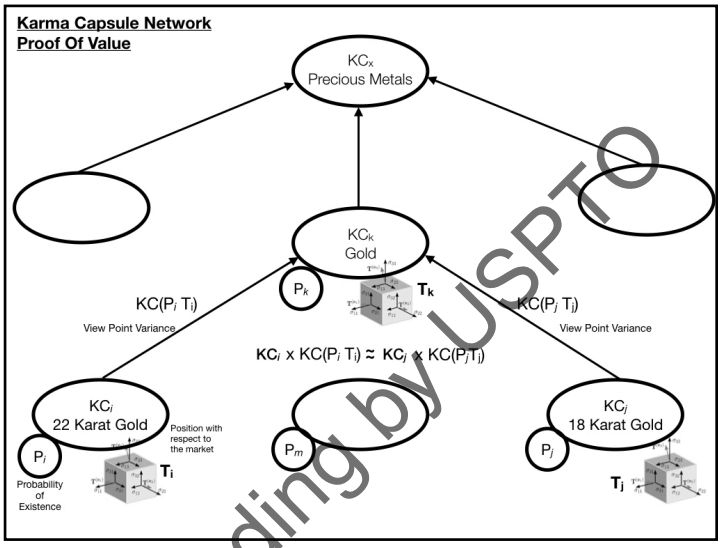
6. Custodian Nodes: These nodes would only participate in taking custody of the Karma Capsules on behalf of the other nodes. Typically a bank / financial institution.

Consensus:

Karma Capsule Network does not compete for transactions. It works not on the principle of money, rather on the principle of Assets. "Systems that think fast not necessarily is Intelligent". KCN is not a money transaction network, it is a value identification and traction network. So, it essentially uses two consensus mechanisms: Proof of Capsule Value and Proof of Capsule Ownership(&authority). The capsule is activated only if the transformed poses of the object coming from the layer below, match each other. Which higher level capsule gets the vote? A capsule with no Child is turned OFF.

Proof of Capsule Value:

The value of a capsule keeps changing due to the Local and Cosmic Environments. The local environment applies the Cross-Entropy cost(or loss) function and uses a combination of the stochastic (random) gradient descent and backpropagation. A typical capsule receives multi-dimensional prediction tensors from capsules in the layer below and looks for a tighter cluster of predictions. If it finds a tighter cluster, it outputs a) a high probability that an entity ("Thing") of this type exists in its domain. and b) the general relativity of the entity with respect to the cluster.



This is the best proof of value mechanism that filters our the noises and fraudulent claims of the value because high-dimensional coincidences do not happen by chance. The whole bunch of low-level entities is voting for the presence and value. At the lower level, "Thing" moves to a different viewpoint or its tensors change, it will be possibly represented by a different capsule. This type of equivariance is called place-coded equivariance. The higher Level Capsule Represent the Whole and all the lower capsules voting for it represents parts. If the part only moves(varies) to a small distance(small change in the tensor values), it will still be represented by the same karmaCapsule but the tensor outputs of the karmaCapsule will change, For example: If the Thing is "Gold", a smaller variances in the characteristics like karat(for purity), it is still Gold but with different mix of gold with copper. So, in this case, the newly identified thing is a Gold but will be a Gold of different quality. This type of equivariance is Rate-Coded. Higher-Level capsules have bigger visibility(and acceptability in the cosmic(the karma Capsule Network), so the low-level place-coded equivariance gets converted to rate-coded.

Proof of Capsule Ownership:

Proof of ownership is one of the two consensus methods of KCN. Every karma Capsule is by default owned by the local node that had created it until it is transferred to a different node's vault. The proof of ownership is verified by the transacting buyer node(s) before any transfer is complete.

Trust-less-ness & Immutability :

Data collected by humans has a problem of "perceived value". Most of these data are empirical data that can change with new empirical evidence. But when data is captured by machines and mathematically stored, it can have a foundational status that is lacking in any other form of knowledge. It is because mathematics consists of imminent truth that cannot be modified by the empirical data. The key to trustlessness is to distribute and disappear the trust among all participants. The custodian of the trust should be the immutable network itself. Most distributed trust-less networks currently used in the world have one way or another, built by programmability either through smart contracts or through consensus algorithms to dictate the logic. KCN, on the other hand, is built on the principle of creating an Autonomous system of "Value" identification, generation and transfer from the sources of information. KCN would let the ANN learn from both sources and cosmic environments and program the LOGIC by itself. So, the applications that are implemented and use KCN would get an Industrial Strength immutable, trust-less distributed ledger, and distributed computing artificial neural network that can learn from any type of environment making it applicable to a wide range of use-cases, industries, and marketplaces.

KCN remembers every capsule the local node had ever created and its transformational phases in the transient memory. So, all transactions, policies, data, rewards, states associated with that specific capsule lives in neurons. KCN can be viewed as a transformer network that looks into all previous memories. These previous memory tensors can be stored in the supplementary memory. These memories of the past can be reached through the recursive algorithm as a query. In CapsNet, recursive calls uses same neurons in the same weight that means when the recursive calls finish, it needs to pop back to the same level, so all that information going on at

the high level when you made that recursive call cannot be stored in the neural activities because we used them in the recursive calls. So, they have to be stored somewhere else. That's why they need tone stored in the supplemental, associative memory. Integral calculus chain rule comes handy to achieve such recursiveness by avoiding backpropagation mechanisms.

A “Key-Value” pair is stored in associative memory. Each synopsis shall be provided with several timescales to have Long-Term Knowledge. And also temporary Stores in the weights. Instead of queries matching the keys of the previous words, the keys of previous words can be used to store things in an associative memory in synopsis and the query could be the input to that associative memory; and the query that is close to the key would produce the value that is close to that the key would have produced. The result of such a query would be the sum of all the weights and by how well that key match.

karmaContracts:

karma Contracts are vehicles of limited programming that an external application, user can use to conduct transactions within the KCN. There can be multiple karma contracts working together to achieve common business goals. For example, an asset transfer agreement - individual trade logics(terms) from the same node or bunch of nodes can be a separate karma contract and they can be combined together as a large contract to execute full trade agreement).

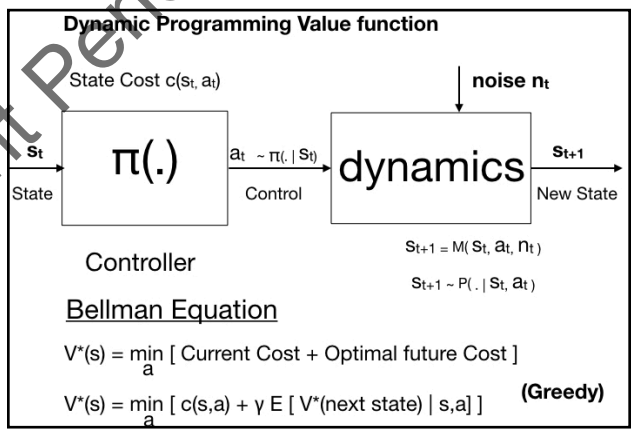
Scalability & Transaction Management:

It is made of homogenous units such as ReLU neurons. It is highly optimizable and scalable supporting complex computations. KCN provides constant running time with the same amount of computing and memory use regardless of input. The local environment shall provide an option to manage the existing transaction by monitoring and modifying (under some situation for this program need to be written with rules). Manage existing transactions - monitor and modify (under extraordinary circumstances) and manage the outcomes. The transactions are happening within the system, we should have a provision to see how many transactions are in the system by hitting the blockchain. a request to modify the blockchain has to be confined to the consensus. if you want to roll back a transaction, the nodes should sign off of that rollback.

The manager should have enough components to monitor the performance and time taken for the transaction. You may want to promote a transaction to execute faster than other transactions.

Karma Policy:

Traditionally, there are three approached, Policy-Space Optimization, Dynamic programming (value function approximation/Iteration) and Actor-Critic (a hybrid of first two). Quite different from it, the karma Capsule network is based on a unique method called Actor-Critic-Audience. (ACA). Principle of Optimality from Bellman's dynamic programming and Markov decision process lays the basic foundation for the policy dynamics of KCN. The Markov property states that the current state completely characterizes the stat of the cosmic environment which is defined by (S - Set of the Possible States, A - Set of Possible Actions, R - distribution of Rewards for a given state-action paid), P - transition probability over next give state -action paid, γ - discount factor.)



The objective is to S —π—>A, identify the right policy that maps the state with the actions. The optimal policy π* shall be achieved by max(Σrtγt) by running the policy for a while. work with π(s,θ) ; θ could be linear or nonlinear regression function in general; in this case, replaced by the CapsNet combination of weights(w) and biases(b). The policy network is usually separated from the Value network in other forms of Agents(ANNs as well). But in the case of capsNet, the architecture accommodates the combination of layers until the final layer to separate the value and policy networks. So as obvious, the Actor represents the policy and critic represents value function. The third component, the audience is a unique component of KCN which represents the bunch of participating nodes in a specific transaction done on the specific karma Capsule. The audience part shall be injected through a programmable interface for karma Policy. Like karma Contracts, policies are immutable elements that can have mathematical algorithms(codes).

Conclusion:

karma capsule Network is a new breed of a network that is a necessary innovation for the current global scenario with the development of Industry 4.0, expansion of electronic machines and gadgets with more businesses moving to cloud and people using more autonomous vehicles, etc. Some of the areas where karma Capsule Network shall be used effectively are: digital Assets creation and Monetization from cyber-physical elements, IoT Edge Devices tracking and Monitoring the Data, Track & Trace of Spatio-Temporal elements in supply chain, predicting the future values and the best policies of trading in exchanges, smart Cities real-time monitoring and control, digital money with intrinsic value, trust-less real estate trading platforms, energy certificates trading, healthcare product provenance, clinical trials, digital assets vault and custody for banks, settlement system for banking and financial transactions and much more. But these are not the only reason I was motivated to create the karma Capsule network. The major motivation for me to create KCN is to reduce the negative effects of AI and Robotics on human social and civic life. I would like to take the human away from being conditioned; let them condition & teach Robots and Robots do good or Dirty Job, thereby humans can focus on becoming human again doing social activities, involved more with nature, family and being self. Robots communicate with each other and with computers so that humans can communicate effectively with fellow humans. We humans should focus on creating a good Social Engineering environment for us to progress as a Civilized society. God created humans; human-created many things that had created complexities around him. Now time to reduce the complexities by disciplining the system around us. We are at crossroads with indisciplined robots getting built around us. Let's get into action disciplining them now itself before it is too late. And I truly believe that karma Casule Network will be the constructive force that I gift to this world.

Sincerely,

By: Maya Suresh Kannan Balabisegan (suresh.balabisegan@gmail.com)

References:

1. Logic Or Art Of Thinking, Arnauld Antoine.
2. PTP Framework - Maya Suresh Kannan Balabisegan, 2007.
3. karmaCapsuleNetwork(First Edition) Nov 17,2018, Maya Suresh Kannan Balabisegan
4. Enterprise Integration Dashboard, An Architecture Of Dynamic Integration of Distributed systems May 29 2008; Maya Suresh Kannan Balabisegan
5. EAstir A Business Anatomy, Physiology & Neurology Analytics Engine, October 5, 2014; Maya Suresh Kannan Balabisegan
6. D.H. Ackley, G.E. Hinton, and T.J. Sejnowski, "A learning algorithm for Boltzmann machines", Cognitive Sci., vol. 9, pp. 147-169, 1985.

6. MATRIX CAPSULES WITH EM ROUTING(https://openreview.net/pdf?id=HJWLfGWRb)Geoffrey Hinton, Sara Sabour, Nicholas Frosst Google Brain Toronto, Canada

7. Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking by Guanghan Ning; Zhi Zhang; Chen Huang, Zhihai He; Xiaobo Ren; Haohong Wang https://archive.org/details/arxiv-1607.05781

8. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding; https://arxiv.org/abs/1510.00149

9. Parsing Natural Scenes and Natural Language with Recursive Neural Networks ; Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, Christopher D. Manning http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.221.4910&rep=rep1&type=pdf

10. Progress Report on Artificial Intelligence, Marvin Minsky and Seymour Papert, Dec 11, 1971 - https://web.media.mit.edu/~minsky/papers/PR1971.html

11. Microcognition: Philosophy, Cognitive Science, and Parallel Distributed Processing, Andy Clark