Article:
Rudd-Orthner, R.N.M. and Mihaylova, L. (2020) Repeatable determinism using non-random weight initialisations in smart city applications of deep learning. Journal of Reliable Intelligent Environments, 6 (1). pp. 31-49. ISSN: 2199-4668

https://doi.org/10.1007/s40860-019-00097-8

Repeatable Determinism using Non-Random Weight Initialisations in Smart City Applications of Deep Learning

Richard N M Rudd-Orthner[1,2], Lyudmila Mihaylova[1]
[1] University of Sheffield, Sheffield, UK, {RNMRudd-Orthner1, L.S.Mihaylova}@sheffield.ac.uk
[2] MASS KSA (a Cohort plc company), Riyadh, KSA, {rruddorthner@mass.co.uk}

Abstract—**Modern Smart City applications draw on the need for requirements that are safe, reliable and sustainable, as such these applications have a need to utilise machine learning mechanisms such that they are consistent with public liability. Machine and Deep Learning networks therefore are required to be in a form that are safe and deterministic in their development and also in their deployment. The viability of non-random weight initialisation schemes in neural networks make the network more deterministic in learning sessions which is a desirable property in safety critical systems where deep learning is applied to smart city applications and where public liability is a concern. The paper uses a variety of schemes over number ranges and gradients and achieved a 98.09% accuracy figure, +0.126% higher than the original random number scheme at 97.964%. The paper highlights that in this case it is the number range and not the gradient that is affecting the achieved accuracy most dominantly, although there can be a coupling of number range with activation functions used. Unexpectedly in this paper, an effect of numerical instability was discovered from run to run when run on a multi-core CPU. The paper also has shown the enforcement of consistent deterministic results on an multi-core CPU by defining atomic critical code regions, and that aids repeatable Information Assurance (IA) in model fitting (or learning sessions). That enforcement of consistent repeatable determinism has also a benefit to accuracy even for the random schemes, and a highest score of 98.29%, +0.326% higher than the baseline was achieved. However, also the non-random initialisation scheme causes weight arrangements after learning to be more structured which has benefits for validation in safety critical applications.**

Keywords— Repeatable Deep Learning Networks, Non-Random Weight Initialization, Security and Information Assurance, Smart Cities Safety-Critical AI, Learning Session Determinism.

I. INTRODUCTION

Smart City applications of Artificial Intelligence (AI) has the potential for growth in many areas in human life as an assistive technology [1], particularly the use of Deep Learning Networks and frameworks; however, applications for Safety Critical software with public liability concerns [2] has additional challenges in security in terms of Information Assurance (IA) in the context of hazard avoidance and safety criticality [3]. It may be argued that the application of AI and Deep Learning Networks have goals for replicating or challenging human abilities against a human performance baseline [4]. Although Safety Critical software has goals of completeness, correctness and repeatability making it rigorous in the development and in the deployed application performance [5], arguably to reach a performance that is *'more than human'*, in that it reduces human error [6]. With this consideration the application of Deep Learning Networks has challenges when applied to Safety Critical software in terms of gaining understanding and confidence for verification and validation of the machine learnt generalisation model [7], and that is a challenge for IA for AI both in the formed neural network generalisation model but also in the processes that formed that model [8].

Smart City applications of deep learning have seen increasing popularity particularly as a decision making assistance tool [9, 10, 11, 12, 13]. A paper by De Souza et al [14] in 2019 surveys Machine Learning and Deep Learning uses in the Smart City applications, and applies data mining techniques to publications. Disregarding keywords that were highlighted in the paper findings but were part of their search criteria, it finds that the publications where it is most prevalent are from China, and with keywords of Models, Consumption, Prediction and Energy Efficiency and related to journals for energy consumption and transportation in the subjects of energy, health, and urban transport. Also a paper by Nosratabadi et al [15] identifies that Smart City applications are using the following techniques: artificial neural networks; support vector machines; decision trees; ensembles, Bayesians, hybrids, neuro-fuzzy; and deep learning. Of particular interest in this paper is the artificial neural network, and within this paper applications of the use of artificial neural networks are cited for: lightning detection, transport, wind turbine stability and water leak detection. Some of these applications provide smart assistance for transport network prediction and parking availability, and others provide efficiencies like water leak detection and wind turbine utilisation. But it is lightning detection that

gets close to the area of hazard prediction and avoidance, and is an example of artificial neural networks being used in risk applications. In that work [16], building and lightning conductor shape datasets are used with a number of artificial neural network architectures to make predictions of the lightning hazards using a process involving detection and classification applied to building-type categories.

As a motivation, in this research paper there are multiple experiments that are conducted with retraining sessions, and the focus of this paper is repeatable determinism between training sessions. Moreover, as a foundation layer for Deep Learning Networks to have determinism in learning sessions, this paper is part of a research thread that is examining the change in weight values of Deep Learning Networks after learning. It builds upon a previous journal paper [17] that was early work to understand weights and biases for formula extraction as a complement to rule extraction. As part of that research an unexpected problem occurred with repeatable determinism; a variation in results occurred, and also as that work was seeking to extract meaning from the weights and biases, the notion of weight initialisation became important. The premise of this is that if the update was to be consistent it might be affected by the initial state, and the paper on the early work [17] found that unused vectors may retain a residue of the initial state. These research thread experiments were required to make measurements and comparisons from a stable set of known weights and biases before and after learning, such that comparisons after learning are repeatable and the experiment is controlled. As such, currently accepted schemes of random number initialisations of the weight values [18] may need to be repeatable to preclude run to run variation. In this way the weight value initialisation is not a varying contributor, and initial condition residues [17] do not influence unused or less used vectors. In consideration of Smart City applications, this research thread is looking to apply artificial neural networks in safety critical fields that may be important where public liability and confidence in a decision assistance is present. As current applications are confined to decision assistance and algorithm selection.

When considering an alternative policy in initialisation [19], the seeding of random numbers or the saving and retrieving of a previously generated random initial state, may provide a repeatable determinism in learning sessions in terms of a start condition that is finite. However, using a non-random initialisation state might provide greater understanding of a network when it is ordered, where as a random set is both asymmetric in values and unordered at the outset. Also, as the Dense Layers of neural networks are fully connected, a disordered arrangement in the weight initialisation state should not be important or significant, and a random number set that is numerically ordered should give the same performance.

Another approach that is motivated to enhance the subsequent update is the Xavier / Glorot and the He et al initialisation approaches in the article [20]. These approaches also use a random number set that is zero centred and can be normally or uniformly distributed, but sets the distribution variance to be dependent on the "number of neurons" in other layers of architecture. However, these two approaches differ by the gain value that they use to avoid saturations or dropouts. He et al builds upon Xavier / Glorot's approach to provide a signal gain normalisation that attains a distinct minima quicker in updates.

Moreover, it may be considered that Xavier / Glorot and He et al initialisation approaches have application to a non-random initialisation scheme. An article [21] states that the use of random numbers allows asymmetry in the start condition whereas using a fixed value provides equal symmetry. Also in that article [21] is a view that, in the case of random initialisation the network is being taught to unlearn an initial condition in favour of a new learning condition. This means that for best effect, the initial condition and the learnt state must have a compatible symmetry transition. This paper looks at the viability of non-random weight initialisation schemes in dense layers, to be used in place of the random number weight initialisations of an established well understood test case, where those symmetry transitions maybe more controlled. Xavier / Glorot and He et al initialisation gain approaches for the update efficiency can also be applied to that non-random scheme.

Another challenge is the time evolution of a solution, or the perfection of a solution with change i.e. the adaptation of a prediction in response to new behaviours. Melen at al provide an alternative approach [22] using a rule based approach that is more acceptable to safety critical applications rather than a neural network approach has been thus far. The Melen et al approach uses a Bayesian Network to control rule probability for selection of Expert System rules. However, in the neural network context, this approach might express the need for repeatable determinism if neural networks are to be updated or retrained when adapting to change, and learning session variations of accuracy are a disruption to this.

Applications to a mission critical problem may be considered. There has been research in this area for several years, with a number of papers relevant to safety critical applications in unmanned air vehicles [23], the automation of space missions [24] and unattended telecoms fault tolerance [25]. However, in the context of Smart City applications that have public liability or hazard concerns, the application of neural networks may be limited to advisory, decision assistance or algorithm selection. Repeatable determinism is a foundation of safety critical applications. In this context, the paper examines the nature of learning session variations, with an assumption that random states used as a stochastic

coverage could have alternatives that allow coverage without learning session variation. As this research is dealing with a fundamental level, the use of a familiar test case, although not a mission critical problem, it is well understood and is a simple architecture with dense layers that is mature to provide a demonstration benchmark.

An advantage of the deep learning network approach in the overall research thread, is that it has the ability to form generalisation models that can perform tasks that may be considered intractable by traditional approaches. However, without controls, it can also form solutions that are not compliant to known understanding or real-world physics. For Safety Critical systems where public liability is a concern, repeatability and determinism are desirable features for verification and validation, both for the processing to form the generalisation model, and when making a prediction with the model when deployed. As both repeatable and deterministic aspects are desirable attributes and also form an experimental control, one aspect that may make a disruption to this is the use of a random number initialisation state of weights before learning. Particularly if a residual of the initial state values is retained and varying, or if the asymmetry of the initial state is uncontrolled. It also may be that in dense layers that the use of random number initialisations is not of positional significance, although the non uniform step between values may be. The familiar test case to be used also uses two dense layers and therefore provides a benchmark for these initialisation schemes.

In Smart city applications neural networks and deep learning can be applied to municipal power grid management, traffic management, fault prediction and avoidance and applications within the home and car. As such applications may have public liability concerns, and the deployment and adoption of such systems may have a safety critical aspect that is required to be dependable but also safety controlled. Arguably the deployment of deep learning neural networks has been restricted to advisory or assistance roles, thus avoiding the responsibility of rigorous verification and validation for safety critical applications. However, clearly there is an appetite for deep learning neural networks in these applications, and this research looks at one aspect of this which is learning session Repeatable Determinism but more particularly the viability of an alternative initialisation scheme that is not random in dense layers.

A. The Paper Structure

The paper's approach is defined in section two, and will use a well known deliberately simplistic example that is mature, familiar and well understood which is using dense layers. In section three, the baseline example is described and the initial baseline performance measured. These initial results will use ten reset runs of learning because there is a run to run variation in learning session results. Also the baseline example uses five epochs, but the first epoch will be focused on as this is the initial learning after the initialisation state. Later in the paper there is a comparative use of the five epochs with highest scoring schemes that will compare back to the example baseline. Although in this example case the learning session variations have a relatively small variance. However, the objective is to gain no variance in learning sessions, as that is the Repeatable Determinism that is desired. In section four the random number generators are seeded and the first epoch focused on to provide a single epoch baseline. In section five, four fixed value schemes are experimented with and result in low scores. In section six, four linear ramp schemes are experimented with that test number ranges with a constant gradient, and better scores are achieved. In section seven the sinusoidal slope initialisation schemes are experimented to introduce a change in slope and with comparable scores to the linear ramp. Although the sinusoid and linear ramps both scored well the learning session variation persists, and section eight resolves the run to run learning session variation issue for discrimination purposes. Section nine experiments with a custom SoftMax function with numerical representation scaling, and section ten draws the conclusions. Section eleven makes recommendations and section twelve identifies areas for further research. The paper's contribution is the analysis of non random initialisation schemes and shows that a linear ramp using the Xavier / Glorot can be just as effective, although the paper also shows the enforcement of repeatable determinism

II. The Approach

This paper looks at different non-random schemes with number ranges and gradients for a weight initialisation scheme to be used in place of a random number initialisation state. In Figure 1 the experiment design is illustrated showing variant and invariant components for the baseline control and experiment cases.

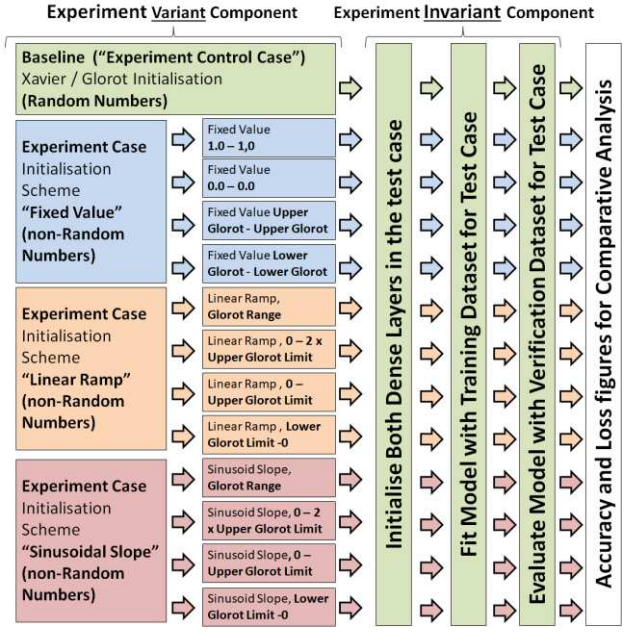

Fig 1 Architecture of the Experiment's Design.

In Figure 1, the original scheme case of the baseline and invariant components is illustrated in Green to show the control case. In Blue, Orange and Red are the three experiment variant classifications, each with four permutations, each of those four experiments use different number ranges around the Xavier / Glorot limits. Although, it is possible that Xavier / Glorot and He et al initialisations can still be applied, but in place of their respective adapted random number variances, a non-random number range and gradient is used that is set between the Xavier / Glorot limits instead.

Three main non-random initialisation weight schemes are experimented with, these are: fixed value, uniform linear ramp and sinusoid slope. In each of these non-random schemes the number range and gradient are changed as experiment controls with Xavier / Glorot limits. Those three schemes that are employed provide discrimination between number ranges and gradient slopes used in those schemes. The fixed value scheme has no gradient and no number range, the linear ramp scheme has a constant gradient and controlled number range, and lastly the sinusoidal slope schemes have a variation in the gradient and a controlled number range.

The research contribution that this paper is seeking to provide is to answer a research question, that is: *"Are random number initialisation weight values required as the initial state before learning to have high accuracy in predictions, or can a non-random scheme also have comparable performance in those predictions".*

This paper is a foundation environment for subsequent research experimental work that will examine changes and adaption to weights and biases after learning model fitting sessions in rule extraction. The foundation environment is using Anaconda Python, NumPy and Keras with TensorFlow deep machine learning framework accessed through the Jupyter Notebook web services environment. This work was required to establish a repeatable result with a defined known initial state that has comparable performance to the existing random initialisation schemes used currently. The experiment's initial state before learning is to be defined, known and predictable such that it may be controlled and accounted for in results as a deterministic initial state that forms an experiment control.

In the experiments a well understood problem that is published will be used. The MNIST dataset in TensorFlow with Keras and NumPy. This is an application of recognising hand written text characters and although in itself may not be a safety critical problem with public liability concerns, text character recognition in number plates still could have legal liabilities, and it is a mature reviewed solution and provides a fixed baseline to demonstrate a performance comparison with dense layers.

III. The Baseline Control Case

This example is familiar to researchers and is being used to demonstrate weight initialisations that are not using random number sequences, see the architecture in Figure 2.

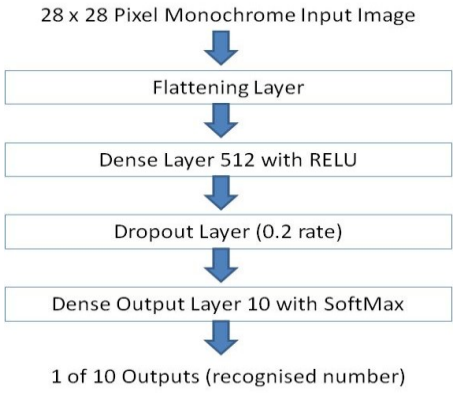

Fig 2 Architecture of the Baseline Example.

This example is known as the *"hello world"* of Neural Networks. When it is run the output from the evaluate command provides both Loss and Accuracy figures, from ten consecutive learning session runs it is noted that the losses and accuracies vary from run to run in each learning session (see Table 1).

Run	Loss	Accuracy
1	**0.06106613632314256**	**98.18**
2	0.06175447308695293	98.16
3	0.07186600035531446	97.72
4	0.06600431568695349	98.18
5	0.06500834331280785	98.13
6	0.06586962914280885	98.01
7	0.07172874092692509	97.96
8	0.08020385432066396	97.65
9	0.0815817079940578	97.71
10	0.07415228985190625	97.94
Mean	**0.069923549**	**97.964**
Var	**5.19614E-05**	**0.042737778**
StdDev	**0.007208428**	**0.206731173**

Table 1: Hello World Example Results.

There is a variation in both the loss and the accuracy figures, which means that there is some variation in the prediction performance depending on the model fitting using random Shuffles and the random initialisation of weights. Random initialisation is a stochastic approach and is an approach that embraces incompleteness in a dataset. However, from the ten runs in Table 1 the minimum loss and the maximum accuracy is in run 1, but it took ten runs to find that that was the initial state that provided the highest score. The reasoning for this variation run to run may be considered to be due to the random number initialisation values present in the weight values and the shuffle ordering of the training dataset. Therefore setting the random number seed values before each learning session should make the random number sequence repeatable, and therefore the accuracy and loss values results the same from each learning session so that it is repeatable and deterministic, as a desirable attribute for safety critical systems in smart city applications.

IV. Seeding the Random Number Generator

Running the same model with the no shuffle added to

the fit command, and again using the evaluate command's loss and accuracy figures the results are gathered. At this point the number of epochs is reduced to one because the early learning in the first epoch after the initialisation is the focus. Also the Tensor Flow and NumPy random seeds have been set to form a baseline value from one epoch that should make the random number sequence defined to be identical in each learning session. The comparative ten run results are in Table 2:

Run	Loss	Accuracy
1	0.1266106634631753	95.91
2	**0.1216393306143582**	**96.21**
3	0.13143637651763856	95.62
4	0.1323663795016706	95.74
5	0.12944038207307457	95.83
6	0.13047181819714607	95.69
7	0.13344295675437898	95.7
8	0.13349654669184238	95.58
9	0.12230887789316476	96.14
10	0.12589706211015583	95.93
Mean	**0.128711039**	**95.835**
Var	**1.92267E-05**	**0.045316667**
StdDev	**0.004384824**	**0.212877116**

Table 2: Single Epoch Random Number Seeded Baseline Results

From these ten results (in Table 2) run 2 has the highest score but it can be seen that the variation is still present at the same variance and standard deviation, although arguably the mean accuracy may have lowered by ~2%. These values without the shuffle in a single epoch form the comparison baseline for further measurement experiments, as it is the epoch that occurs after the initialisation state. This might be an indicator that this variation in learning sessions is not attributable to the initialisation of the weights alone. Adaption of the example allows the random initialize values to be substituted with non-random schemes, and this will allow the three discrimination initialisation cases to be experimented with. The three initialisation schemes will be experimented with and these are: fixed values, linear ramps and sinusoid slopes, these will test different numerical aspects from values, gradients to number ranges. These three schemes provide a isolation of numerical aspects where Fixed Values have no gradient and number range, Linear Ramps have a number range and a fixed gradient and the Sinusoid Slopes have a number range and a variation in the gradient.

V. Fixed Value Scheme

Starting with a weight initialized array of 28×28×512 in the second layer and 512×10 weight values in the fourth layer that are all containing a fixed value one (defined in Figure 3). The weight initialisation layer dimensions in the second layer *(or 1st Dense Layer)* reflects the 512 neurons and the image dimensions in the dataset which is 28 by 28 monochrome pixels. The weight initialisation layer dimensions in the fourth layer *(or 2nd Dense Layer)* reflect the 512 neurons and the 10 categories of numerical single digit values that an image could have in that dataset. The ten learning session runs provided the results in Table 3 with a fixed value one.

1st Dense Layer = {+1 … +1}
2nd Dense Layer = {+1 ... +1}
Fig 3: Fixed Value (1.0) Weight Initialisation Tensor

The ten run results are as follows in Table 3:

Run	Loss	Accuracy
1	14.490169499206543	10.1
2	14.490169499206543	10.1
3	14.490169499206543	10.1
4	14.490169499206543	10.1
5	14.490169499206543	10.1
6	14.490169499206543	10.1
7	14.490169499206543	10.1
8	14.490169499206543	10.1
9	14.490169499206543	10.1
10	14.490169499206543	10.1
Mean	14.490169499206543	10.1
Var	0	0
StdDev	0	0

Table 3: Fixed Value (1.0) Results

When fixed values are used the network initial state is symmetric and many nodes may be performing the same or similar calculation as the initial condition is the same in every weight, also the value one would appear to cause a saturation that Xavier / Glorot and He et al initialisation approaches were meant to overcome. The results have no variance run to run due to the saturation and loss in learning as the scheme provides a symmetric output as the initial state that is not providing variations in every nodes calculations towards the categorisation layer. The next experiment is to use a fixed value of zero instead (in Figure 4), and gains the following results from the ten runs (in Table 4).

1st Dense Layer = {+0 ... +0}
2nd Dense Layer = {+0 ... +0}
Fig 4: Fixed Value (0.0) Weight Initialisation Tensor

The ten run results are as follows in Table 4:

Run	Loss	Accuracy
1	2.3011607303619384	11.35
2	2.301160646820068	11.35
3	2.3011608764648437	11.35
4	2.3011607875823974	11.35
5	2.3011607551574706	11.35
6	2.301160848617554	11.35
7	2.3011607162475585	11.35
8	2.3011607135772705	11.35
9	2.3011607372283938	11.35
10	2.3011608749389647	11.35
Mean	**2.301160769**	**11.35**
Var	**5.88128E-15**	**0**
StdDev	**7.66895E-08**	**0**

Table 4: Fixed Value (0.0) Results

Again the accuracy values have no variance although the loss has a small variance run to run and the Xavier /

Glorot and He et al initialisation approaches would optimise the number range towards a useful range to avoid dropouts and saturations. In the Keras code [26] it is noted that the random initialisation value scheme used by default is "Glorot uniform" which is also known as Xavier. This scheme would have set the random weight initialisation value limits to be between +/-0.0680414 in the first dense layer (as per *sqrt(6/(28×28+512))* and set the weight initialisation values between +/-0.1072113 in the second dense layer as per *sqrt(6/(512+10))*. So using the upper range of those values (Figure 5) the benefit of those values is demonstrated in Table 5:

1st Dense Layer = {+0.0680414 …+0.0680414}
2nd Dense Layer = {+0.1072113 ... +0.1072113}
Fig 5: Fixed Value (Upper Glorot range) Weight Initialisation Tensor

The ten run results are as follows in Table 5:

Run	Loss	Accuracy
1	1.7905318803787231	28.45
2	1.7931770057678222	28
3	1.7885990364074706	28.22
4	1.7926008644104003	27.79
5	1.792194675064087	28.13
6	**1.7864114040374757**	**28.93**
7	1.7913340614318847	27.81
8	1.7892346405029298	28.36
9	1.7960710954666137	27.85
10	1.7946386306762696	28.02
Mean	**1.791479329**	**28.156**
Var	**8.40606E-06**	**0.124671111**
StdDev	**0.00289932**	**0.353087965**

Table 5: Fixed Value (Upper Glorot range) Results

At the upper Glorot limit value the accuracy in the ten runs has increased three fold and endorses the use of the value, however the network is not performing well with fixed values as it is not utilizing and combining different node influences as all values are the same at the outset, and may cause a same or similar calculation in many nodes. Using just the lower limit (Figure 6) may conflict with the RELU in the 1st dense layer but for completeness the lower Glorot limit value is tested (Table 6).

1st Dense Layer = {-0.0680414 …-0.0680414}
2nd Dense Layer = {-0.1072113 ... -0.1072113}
Fig 6: Fixed Value (Lower Glorot range) Weight Initialisation Tensor

The ten run results are as follows in Table 6:

Run	Loss	Accuracy
1	2.301160766983032	11.35
2	2.3011607803344725	11.35
3	2.3011608039855957	11.35
4	2.301160773086548	11.35
5	2.3011607959747313	11.35
6	2.3011607624053956	11.35
7	2.301160643005371	11.35
8	2.301160859680176	11.35
9	2.301160783004761	11.35
10	2.301160697555542	11.35
Mean	**2.301160767**	**11.35**
Var	**3.49831E-15**	**0**
StdDev	**5.91465E-08**	**0**

Table 6: Fixed Value (Lower Glorot range) Results

These results compare with the fixed zero value experiment. However, with fixed values schemes this implies there is a sensitivity to the initialisation value and that the upper Glorot value as a potential to allow better learning. Also the network populated with a single repeated value in every weight as the initialisation scheme provides a symmetry that is difficult for the network learning to overcome, and may cause the network to be under-utilised. But there still remains an application for a fixed values scheme, and this application is to inhibit learning for areas of weights that map to unused input vectors or when a network is deliberately under populated for growth, transferred learning or anticipated retraining in response to changes in the operating environment.

A summary table in Table 7 follows of the experiments with the single epoch baseline and the fixed value weight initialisation schemes:

Experiment	Loss and Accuracy	Comment
Fixed value 1.0	**Accuracy** Mean **10.1%** Var 0 StdDev 0 **Loss** Mean 14.49016949 Var 0 StdDev 0	This scheme is the lowest score, however, it may have some applications to reserve a network area for later use like when an input vector is unused.
Fixed value 0.0	**Accuracy** Mean **11.52%** Var 0 StdDev 0 **Loss** Mean 2.301160769 Var 5.88128E-15 StdDev 7.66895E-08	Low performing and compares with the negative number experiment. However, it may have some applications for an network area to be disregarded.
Fixed value Upper Glorot limit	**Accuracy** Mean **28.156%** Var 0.124671111 StdDev 0.353087965 **Loss** Mean 1.791479329 Var 8.40606E-06 StdDev 0.00289932	Highest score although low performing but does show that the Glorot value has a benefit, although using only that value may under-utilise the network.
Fixed value Lower Glorot limit	**Accuracy** Mean **11.35%** Var 0 StdDev 0 **Loss** Mean 2.301160767 Var 3.49831E-15 StdDev 5.91465E-08	Compares with the zero number experiment and may conflict with the use of RELU in the first dense layer.

Table 7: Fixed Value Summary Table

It appears that using a fixed number as an initialised value has a large impact on the resultant accuracy, and

that some of that accuracy is connected to the value used. This may be unsurprising as a fixed value is a large departure from a random number scheme in terms of number range and the step of influence in a node's initial value. It may be that the value one may have caused the lowest score and was far outside the Glorot values, and the upper Glorot value has a threefold benefit. But using fixed repeated number initialisation schemes may have under-utilised the network, with matching initial influences throughout the network that are performing same or similar updates. However, it could conceivably have a lower impact on learning on a remaining network if used to reserve a network area for growth or transferred learning. The value zero could conceivably be used to disregard an area of a network by switching it off, and indeed zero values are used in pruning. These results may of course only pertain to this model in particular in terms of values used, but we might of expected that fixed number schemes would have an large impact as statistically they are deterministic. However the role of determinism in mission critical applications might have use for forcing deterministic outcomes in areas of a vector input that are to be disregarded or are unused and could be subjects for transferred learning.

Moreover, there is a concern for repeatable determinism with learning sessions, as there is accuracy variations even with the random number generator seeded, the shuffle switched off and also when the weight initialisation vector is finite with the same dataset and architecture. An observation is that a 32bit number floating point bit representation looses resolution when calculations are performed with 5 significant places difference. Also noticing that the input data is positive image values in the scale 0 - 255 rescaled to 0 - 1 and the weight initialisation value from the Glorot limit is 0.06... which is two significant places different. Where a five significant place difference is experienced in an update computation this will begin to affect a 32bit calculation representation accuracy. A possible source of concern is the use of the SoftMax activation function, as the SoftMax function divides a number by the sum of an exponential number set. Alternatively this may be solved by using a larger floating point bit representation for the accumulator in the sum or the use of pre and post scaling of the number scales before and after the accumulator calculations. But this representation accuracy should be a repeatable effect. However, PC processors have a internal 80bit extended floating point precision register, that if interrupted by a task scheduler could cause rounding when a task is stored and retrieved during the learning sessions in asynchronous task scheduling events. This would provide truncation of precision at different times in each learning session and result in variations.

In summary it should also be noted that the results that show a lower mean accuracy are with fixed weight initialisation values. Perhaps the learning is more uniformly affecting other neurons by a similar amount and it could be expected by initial weight values that have no number range variations at the outset of learning. The next set of experiments have a number range in a linear ramp so as to have a numerical difference in influence in each node that can combine, although with a standard interval in the values used.

VI. Linear Ramp Scheme

Using a linear ramp in the Glorot range as the initial values of the weights to provide areas of the neural network that will have different dominance towards an output from the outset of learning, and a fixed gradient of values and number range in those initial weights may be higher performing. The initialisation weight values are defined in Figure 7:

1st Dense Layer = {-0.0680414 …+0.0680414}
2nd Dense Layer = {-0.1072113 ... +0.1072113}
Fig 7: Linear Ramp (Glorot range) Weight Initialisation Tensor

The ten run results are as follows in Table 8:

Run	Loss	Accuracy
1	0.1851132948242128	93.97
2	**0.16467118371911346**	**95.01**
3	0.16555062152668834	94.97
4	0.1738155936975032	94.69
5	0.18077268141284586	94.46
6	0.2119653475858271	93.66
7	0.19896690204292536	93.75
8	0.19868872426487505	93.8
9	0.19239787173271178	93.82
10	0.17692170148193836	94.56
Mean	**0.184886392**	**94.269**
Var	**0.00024044**	**0.276676667**
StdDev	**0.015506117**	**0.526000634**

Table 8: Linear Ramp (Glorot range) Results

It seems that a number range in weight values may be helpful and the accuracy is just ~1.5% less than the baseline result. The number range provides an initial influence that is varied in each node and can be combined providing a better utilisation of the network. The comparable performance with the random scheme is explained by: in dense layers every node is connected so the placement order of values in a non uniform order is not important or significant. But the random number scheme has a non uniform step and is statistically unlikely to arrive on the value zero exactly or the two Glorot limit values, where as in this scheme that is a closer guarantee. To check the number range the same slope is used but nudged upward to positive values as zero to twice the upper Glorot limit and is defined in Figure 8:

1st Dense Layer = {0 …+2×0.0680414}
2nd Dense Layer = {0 ... +2×0.1072113}
Fig 8: Linear Ramp (0 - twice Glorot limit) Weight Initialisation Tensor

The ten run results are as follows (in Table 9):

Run	Loss	Accuracy
1	0.2720342619046569	91.28

2	0.2702090907022357	91.38
3	0.2686337884970009	91.42
4	0.2648709679841995	91.57
5	0.26604176666885615	91.58
6	0.26589927548691633	91.53
7	**0.26427238701581957**	**91.74**
8	0.26519556519016624	91.62
9	0.2715027303129435	91.29
10	0.2719804396606982	91.34
Mean	**0.268064027**	**91.475**
Var	**9.9364E-06**	**0.024094444**
StdDev	**0.003152205**	**0.155223853**

Table 9: Linear Ramp (0 - twice Glorot limit) Results

It seems that the results are similar but a little reduced then the ramp over zero and is ~4% lower than the baseline in accuracy. The gradient was unchanged but the number range was slid up to positive numbers, but that number range reached a number range greater than the Glorot limit and reduced accuracy. In the next experiment the gradient is changed but is now within the Glorot limits, as zero to the upper Glorot limit (in Figure 9):

1st Dense Layer = {0 …+0.0680414}
2nd Dense Layer = {0 ... +0.1072113}
Fig 9: Linear Ramp (0 to Glorot limit) Weight Initialisation Tensor

The ten run results are as follows (in Table 10):

Run	Loss	Accuracy
1	0.239874689412117	92.44
2	0.24703469477891923	92.19
3	0.2418467572107911	92.39
4	**0.22567530573680997**	**92.82**
5	0.25892428045272825	91.72
6	0.24337126431316136	92.52
7	0.22973494304940104	92.79
8	0.2376753763526678	92.35
9	0.23152151829451323	92.71
10	0.23743405939936638	92.55
Mean	**0.239309289**	**92.448**
Var	**9.02308E-05**	**0.105462222**
StdDev	**0.009498988**	**0.324749476**

Table 10: Linear Ramp (0 to Glorot limit) Results

These results are very similar at ~3% less than the baseline, and for completeness trying the negative value of the same gradient as -Glorot to zero (Figure 10). Although it is expected that the ReLU used in the first dense layer's activation function will affect the performance with negative numbers.

1st Dense Layer = {-0.0680414 … 0}
2nd Dense Layer = {-0.1072113 ... 0}
Fig 10: Linear Ramp (-Glorot limit to 0) Weight Initialisation Tensor

The ten run results are as follows (in Table 11):

Run	Loss	Accuracy
1	2.301160806274414	11.35
2	2.301160684585571	11.35
3	2.301160723876953	11.35
4	2.301160803604126	11.35
5	2.3011607578277586	11.35
6	2.301160758972168	11.35
7	2.301160799407959	11.35
8	2.3011608020782472	11.35
9	2.301160831451416	11.35
10	2.301160835647583	11.35
Mean	**2.30116078**	**11.35**
Var	**2.33796E-15**	**0**
StdDev	**4.83524E-08**	**0**

Table 11: Linear Ramp (-Glorot limit to 0) Results

The results are very much lower almost like the negative values that were seen with the fixed values suggesting that layer 2 ReLU activations may have dropouts suppressing values from the outset that are less than zero. A summary table of the results is in Table 12:

Experiment	Loss and Accuracy	Comment
Ramp through Glorot range	**Accuracy** Mean **94.269%** Var 0.276676667 StdDev 0.526000634 **Loss** Mean 0.184886392 Var 0.00024044 StdDev 0.015506117	Learning Session variance in results, but only 1.5% lower accuracy from the baseline in this case.
Same Slope as the Glorot range but slid up to positive numbers	**Accuracy** Mean **91.475%** Var 0.024094444 StdDev 0.155223853 **Loss** Mean 0.268064027 Var 9.9364E-06 StdDev 0.003152205	Variance in numbers, 4% lower accuracy from the baseline in this case.
Change in slope but in positive and Glorot limited	**Accuracy** Mean **92.448%** Var 0.105462222 StdDev 0.324749476 **Loss** Mean 0.239309289 Var 9.02308E-05 StdDev 0.009498988	Variance in numbers, 3% lower accuracy from the baseline in this case.
Same slope as the above experiment but negative values and -Glorot limited	**Accuracy** Mean **11.35%** Var 0 StdDev 0 **Loss** Mean 2.30116078 Var 2.33796E-15 StdDev 4.83524E-08	No variance in the accuracy and variances in Loss. Accuracy is low performing and the ReLU activation function may have affected learning from the outset.

Table 12: Linear Ramp Summary Table

From these results negative number ranges seem to be low performing and positive values higher performing, but the range between Glorot limits was the highest performing in terms of accuracy at just 1.5% less than the baseline. The gradient changed between the experiments with 0.0 to Glorot upper limit and 0.0 to two times the upper Glorot limit but had little difference in results, but

the number range and gradient were changed together. In the next set of experiments the gradient and number range are changed more independently using a sinusoidal slope.

VII. Sinusoid Slope Schemes

A moving gradient is used starting with the Glorot range limits in a sinusoidal form such that the number range is the same but the gradient is changing with respect to the linear ramp experiment of the same range (the sinusoidal form is in Figure 11).

1st Dense Layer = cos({0 ... π}) × 0.0680414
2nd Dense Layer = cos({0 ... π}) × 0.1072113
Fig 11: Sinusoid slope (Glorot range) Weight Initialisation Tensor

The ten run results are as follows (in Table 13):

Run	Loss	Accuracy
1	0.17446787632405758	94.66
2	**0.16790239577889443**	**95.09**
3	0.17015618828460574	94.79
4	0.17184093101769685	94.74
5	0.16502467265054582	95.03
6	0.1692778503138572	94.86
7	0.16809104131534697	94.84
8	0.17081736022941768	94.78
9	0.16529247453436255	95.08
10	0.16685232015512882	94.99
Mean	**0.168972311**	**94.886**
Var	**8.76335E-06**	**0.022937778**
StdDev	**0.002960295**	**0.151452229**

Table 13: Sinusoid slope (Glorot range) Results

The results are similar to the linear ramp over the same number range which was also only ~2% lower than the baseline, Using the positive figure experiment with the same sinusoidal pattern in the range 0 to two times the Glorot upper limit in Figure 12.

1st Dense Layer = cos({0...π})×0.0680414+0.0680414
2nd Dense Layer = cos({0...π})×0.1072113+0.1072113
Fig 12: Sinusoid slope (twice Glorot limit-0) Weight Initialisation Tensor

The ten run results are as follows (in Table 14):

Run	Loss	Accuracy
1	0.2793025099083781	91.4
2	0.2805209428802133	91.43
3	0.28077282523810865	91.37
4	0.2799623735308647	91.44
5	0.27916926593780517	91.38
6	0.27978013244271277	91.38
7	0.28000284353345634	91.39
8	**0.2676098602056503**	**91.68**
9	0.2806942581638694	91.42
10	0.28791632864177225	91.24
Mean	**0.279573134**	**91.413**
Var	**2.41043E-05**	**0.01189**
StdDev	**0.004909613**	**0.109041277**

Table 14: Sinusoid slope (twice Glorot limit-0) Results

The accuracy is 4% lower than the baseline. The next experiment also uses positive values, but only in the range 0 to the upper Glorot limit in the same sinusoidal form (shown in Figure 13).

1st Dense Layer = cos({0...π})×0.0680414/2+0.0680414/2
2nd Dense Layer = cos({0...π})×0.1072113/2+0.1072113/2
Fig 13: Sinusoid (upper Glorot limit to 0) Weight Initialisation Tensor

The ten run results are as follows (in Table 15):

Run	Loss	Accuracy
1	0.2200166605822742	93.41
2	0.2440426151908934	92.68
3	0.248060436925292	92.48
4	0.2443391766808927	92.53
5	0.2500976152040064	92.38
6	0.24831699962988496	92.46
7	0.25095710896626117	92.4
8	0.2400451942332089	92.66
9	0.2460106762483716	92.55
10	**0.24416129550859333**	**92.73**
Mean	**0.243604778**	**92.628**
Var	**7.93523E-05**	**0.08944**
StdDev	**0.008907991**	**0.29906521**

Table 15: Sinusoid Slope (upper Glorot limit to 0) Results

Also slightly lower results at almost 3% less than the baseline, but for completeness the sinusoidal range of lower Glorot Limit to 0 is provided in Figure 14.

1st Dense Layer = cos({0...π})×0.0680414/2 -0.0680414/2
2nd Dense Layer = cos({0...π})×0.1072113/2 - 0.1072113/2
Fig 14: Sinusoid Slope (0-Lower Glorot) Weight Initialisation Tensor

The ten run results are as follows (in Table 16):

Run	Loss	Accuracy
1	2.301160799407959	11.35
2	2.3011607650756836	11.35
3	2.3011606666564943	11.35
4	2.30116068611145	11.35
5	2.3011605926513674	11.35
6	2.301160647583008	11.35
7	2.3011606704711913	11.35
8	2.3011606185913087	11.35
9	2.3011607677459716	11.35
10	2.30116082611084	11.35
Mean	**2.301160704**	**11.35**
Var	**6.39136E-15**	**0**
StdDev	**7.9946E-08**	**0**

Table 16: Sinusoid Slope (0-Lower Glorot) Results

As expected the negative values are low performing, but a summary of these experiments using sinusoidal slope patterns are shown in Table 17:

Experiment	Loss and Accuracy	Comment
Sinusoid slope in Glorot Range	**Accuracy** Mean **94.886%** Var 0.022937778 StdDev 0.151452229 **Loss** Mean 0.168972311 Var 8.76335E-06 StdDev 0.002960295	Almost the same score as the same number range with the linear ramp experiment.

Sinusoid slope from twice the Glorot upper limit to 0	**Accuracy** Mean **91.413%** Var 0.01189 StdDev 0.109041277 **Loss** Mean 0.279573134 Var 2.41043E-05 StdDev 0.004909613	Almost the same score as the same number range with the linear ramp experiment.
Sinusoid slope from Glorot upper limit to 0	**Accuracy** Mean **92.628%** Var 0.08944 StdDev 0.29906521 **Loss** Mean 0.243604778 Var 7.93523E-05 StdDev 0.008907991	Almost the same score as the same number range with the linear ramp experiment.
Sinusoid slope from 0 to lower Glorot limit.	**Accuracy** Mean **11.35%** Var 0 StdDev 0 **Loss** Mean 2.301160704 Var 6.39136E-15 StdDev 7.9946E-08	This result also coincides with the linear ramp of the same number range.

Table 17: Sinusoid Slope Summary Table

The sinusoidal slopes have comparable results to the linear ramps, although when compared with the Glorot range the sinusoidal slopes have a small benefit. Perhaps that might be thought to be due to the non uniform interval step in values, and its' statistical probability of having less values nearer zero as per its bath tub distribution. However, later the solution to the learning session variation will show linear ramps are higher performing when a numerical stability is solved.

Taking the highest score of the sinusoid slopes in the range of the Glorot limits and re-running with the five epochs and enabling the shuffle, provides the following results (in Table 18) as a comparison to the original baseline performance:

Run	Loss	Accuracy
1	0.06873708092225715	97.99
2	0.07566913830568082	97.75
3	0.06941359058758244	97.81
4	0.07690233801202849	97.75
5	**0.07229105311079184**	**98**
6	0.07870250816526823	97.79
7	0.06857179706634488	97.98
8	0.07224223068275024	97.86
9	0.07307772935463581	97.75
10	0.07484171458326745	97.87
Mean	**0.073044918**	**97.855**
Var	**1.22188E-05**	**0.010494444**
StdDev	**0.003495545**	**0.102442396**

Table 18: Five Epoch and Shuffle with High Score Sinusoid Slope

In comparison with the high Score Linear Ramp, the results are shown in Table 19 and the accuracy of learning sessions run to run figures are similar, and the accuracy is about the same as the baseline but is not using random weight initialisations:

Run	Loss	Accuracy
1	0.06948850467810408	97.93
2	0.0810321348624304	97.66
3	0.07320999270802131	97.81
4	0.0818435779891268	97.49
5	**0.06348531504337443**	**97.95**
6	0.07764026021502214	97.67
7	0.08206962382048369	97.53
8	0.07047365378377726	97.86
9	0.07122972569263075	97.9
10	0.06634688437929144	97.93
Mean	**0.073681967**	**97.773**
Var	**4.42831E-05**	**0.029801111**
StdDev	**0.006654552**	**0.172629983**

Table 19: Five Epoch and Shuffle with High Score Linear Ramp

The following graph in Figure 15 is a bar graph of the learning session accuracy results, from each of the consecutively reset learning session runs. It is sorted into numerical order to illustrate the relative variations and curve in variations.

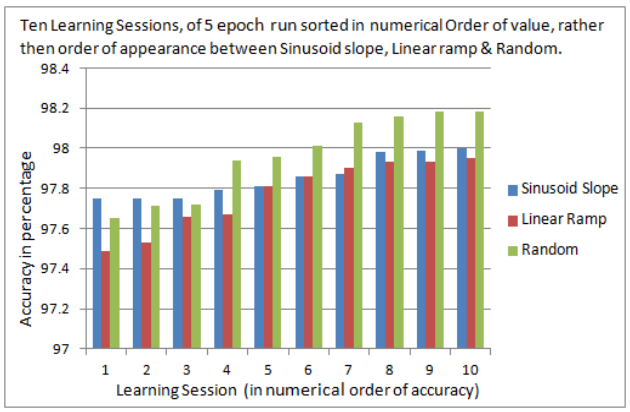

Fig 15: Five Epoch runs between Random, Sinusoid and Linear Ramp

In a comparison of the accuracy values, the accuracy value range between the Original Random, Sinusoid Slope and Linear Ramp schemes is shown in Figure 15. The graph in Figure 15 shows that the original random scheme using Glorot limits still has the higher score potential for the highest accuracy values (98.18%). But the Original Random scheme has the largest variation in accuracy values (0.53%) so that requires more throwaway reset learning session iterations to achieve that score and to know it is the highest value. The next best is Sinusoid slope achieving a maximum of (98%) but has the smallest variation (0.25%) meaning that it achieves a more deterministic measurement quicker and also its' minimum is higher than the minimum of the random schemes. The Linear Ramp achieves a maximum of 97.49% accuracy and variations of 0.46% making it have the lowest accuracy scoring potential, but it still has a smaller

variation and has more determinism then the original random scheme.

Comparing those results with the single Epoch which is the first epoch after the initialisation state. Figure 16 shows the single epoch results for the same schemes.

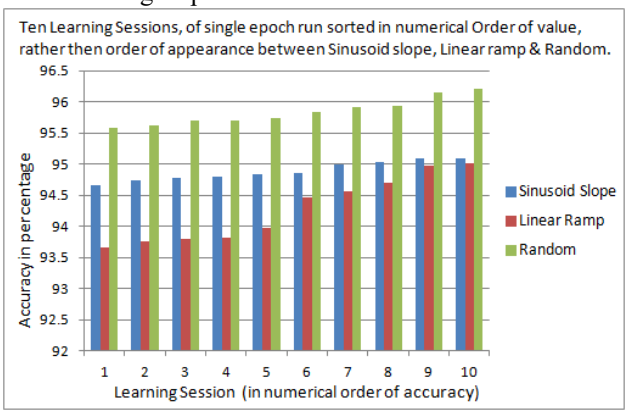

Fig 16: 5 1st Epoch run between Random, Sinusoid and Linear Ramp

Using just the first epoch, the accuracy values are naturally reduced and the variances are increased. The Original Random Scheme's accuracy is 1.97% less (96.21%), the sinusoid slope's accuracy is 2.91% less (95.09%) and the linear ramp's accuracy is 2.94% less (95.01%). However in repeatable determinism order, i.e. variation order in the learning sessions, the lowest variation (i.e. greatest repeatable determinism) is still the sinusoidal slope. Then the original random scheme, has the second lowest variation at 0.63%, and Linear ramp with the highest variation at 1.35%.

In the comparison of those results (Figure 15 and Figure 16) it should be appreciated that the initial condition is a transitory state and the learning will update the weights subsequently as a state change. It may be noted that the linear ramp variance benefited the most from the epochs in terms of repeatable determinism closing its variations by 0.98%, then the sinusoid slopes at 0.18% and followed by the original random number scheme 0.1%. Also the maximum achieved accuracy benefited the linear ramp the most (2.94% increase), then sinusoidal slope (2.91%) followed by 1.97%, Thus meaning that the learning may have been marginally slower in the non random schemes thus requiring more epochs and benefiting more from them.

However, it may be that early learning in the model has been affected, and in some model datasets in learning that the colour of the noise in the image may have been further coloured by the learning session variations and perhaps less fit the selected regularisation scheme as intended, or in this example as there is no regularisation selected, it might provide a unintended regularisation scheme and a third source of noise for colourisation.

Although these results seam to show that the non-random initialisation schemes provide a slightly lower accuracy to random numbers, it is providing higher repeatable determinism then the random number initialisation scheme. The learning session to learning session variation in results is masking the actual repeatable determinism accuracy measurements and causes a number of throwaway learning session runs to be conducted to gain the best accuracy. This learning session variation needs to be tackled to improve the measurement accuracy made, for both the experiment and the baseline values, as the ten learning session runs may be affected by the variation and be adding to a regularisation effect of which those schemes may benefit by different amounts.

VIII. Tackling the Repeatability Run to Run

There is still a variance run to run in the results even using the seeded random numbers with non-random weight value initialisations but taking into account the possibility of the scheduling causing variations in number representations. An experiment to invoke the real-time priority of the scheduler [27] with an affinity to one processor [28] as an attempt to deny or reduce interruption of the task thread. Take note that in some operating systems you may need to run in admin privileged modes.

With the real-time priority selected on a single processor affinity then the learning session becomes completely repeatable in each of the ten runs (see Table 20). This supports the theory that task scheduling is interrupting and truncating calculations in the CPU's internal 80bit extended precision floating point register [29], as now the task is running on one processor uninterrupted in a critical region of code. This provides an accurate repeatable figure for the highest scoring sinusoid scheme (in Table 20).

Run	Loss	Accuracy
1	0.06988054851347116	97.93
2	0.06988054851347116	97.93
3	0.06988054851347116	97.93
4	0.06988054851347116	97.93
5	0.06988054851347116	97.93
6	0.06988054851347116	97.93
7	0.06988054851347116	97.93
8	0.06988054851347116	97.93
9	0.06988054851347116	97.93
10	0.06988054851347116	97.93
Mean	**0.069880549**	**97.93**
Var	**0**	**0**
StdDev	**0**	**0**

Table 20: Highest Score Sinusoid Slope, in a Critical Region of code

It should be noted that this score is the 4th highest value for this scheme encountered for sinusoidal schemes and is achieved in a single learning session run.

Now that the runs are consistent the highest score number ramp scheme with the Glorot range is re-run with no variance in the results and they are similar suggesting that the initialisation and task scheduler denial is providing repeatability run to run (see Table 21):

Run	Loss	Accuracy
1	0.06633297475341242	98.05
2	0.06633297475341242	98.05
3	0.06633297475341242	98.05
4	0.06633297475341242	98.05

5	0.06633297475341242	98.05
6	0.06633297475341242	98.05
7	0.06633297475341242	98.05
8	0.06633297475341242	98.05
9	0.06633297475341242	98.05
10	0.06633297475341242	98.05
Mean	**0.066332975**	**98.05**
Var	**0**	**0**
StdDev	**0**	**0**

Table 21: Highest Score Linear Ramp, in a Critical Region of code

It appears that the ramp is a very slightly better initialisation scheme then the sinusoid of the same number range dismissing the affect of initial varying gradients being of a benefit to the resultant accuracy and loss, at least in this case. This may be because the dense layer is fully connected so the order of numbers is not significant although the distribution values step is, and the ramp values provide a uniform step interval. However, before the processor critical region was used to gain learning session determinism, the task scheduler may have been providing variations in the calculation unintentionally. But now that the 80bit extended precision register's integrity is preserved avoiding random rounding the linear ramp has overtaken the sinusoid slope performance, as the number range and values are assured and the variation in calculations is removed as an unintentional noise source.

Although repeatable results that a comparable score to the baseline is achieved the earlier concern of numerical stability of the SoftMax activation function is investigated.

IX. Custom Number Scaled SoftMax Function

An experiment of the SoftMax activation function used in the final layer, Figure 17 defines a SoftMax function with a rescaling for bit representation numerical stability [30] as was suggested as a possible concern earlier.

Original SoftMax

$$SoftMax(d(1..n)) = \frac{exp(d(1..n))}{sum(exp(d(1..n)))}$$

Modified SoftMax

$$SoftMax(d(1..n)) = \frac{exp(d(1..n) - max(d(1..n)))}{sum(exp(d(1..n) - max(d(1..n))))}$$

Fig 17: Modified SoftMax Function Definition

In Table 22 is the results and there similar suggesting that the numerical stability is having a minor effect although this is the highest accuracy score yet at a 0.04% benefit:

Run	Loss	Accuracy
1	0.06171222376991063	98.09
2	0.06171222376991063	98.09
3	0.06171222376991063	98.09
4	0.06171222376991063	98.09
5	0.06171222376991063	98.09
6	0.06171222376991063	98.09
7	0.06171222376991063	98.09
8	0.06171222376991063	98.09
9	0.06171222376991063	98.09
10	0.06171222376991063	98.09
Mean	**0.061712224**	**98.09**
Var	**0**	**0**
StdDev	**0**	**0**

Table 22: Highest Score Linear Ramp, in a Critical Region of code with modified SoftMax

Although a very marginal increase in accuracy. However, now that the model can be run with repeatable results, the original random scheme is run in a critical region and with the random number initialisation of the weights, but seeded (see Table 23).

Run	Loss	Accuracy
1	0.061059941675240405	98.05
2	0.061059941675240405	98.05
3	0.061059941675240405	98.05
4	0.061059941675240405	98.05
5	0.061059941675240405	98.05
6	0.061059941675240405	98.05
7	0.061059941675240405	98.05
8	0.061059941675240405	98.05
9	0.061059941675240405	98.05
10	0.061059941675240405	98.05
Mean	**0.061059942**	**98.05**
Var	**0**	**0**
StdDev	**0**	**0**

Table 23: Baseline, Random Scheme , in a Critical Region of code

The baseline perfected value is 98.05% which is equal to using the best linear ramp using Glorot limits, although the linear ramp is 0.04% above the baseline when the modified SoftMax is used. The sinusoidal slope using Glorot limits is 0.12% lower than the baseline.

However rerunning the baseline with the modified SoftMax provides 98.29% accuracy which is the highest accuracy achieved and is with random weight initialisation and the modified SoftMax (see Table 24).

Run	Loss	Accuracy
1	0.05775070842197165	98.29
2	0.05775070842197165	98.29
3	0.05775070842197165	98.29
4	0.05775070842197165	98.29
5	0.05775070842197165	98.29
6	0.05775070842197165	98.29
7	0.05775070842197165	98.29
8	0.05775070842197165	98.29
9	0.05775070842197165	98.29
10	0.05775070842197165	98.29
Mean	**0.057750708**	**98.29**
Var	**0**	**0**
StdDev	**0**	**0**

Table 24: Baseline, Random Scheme , in a Critical Region of code & modified SoftMax

The results show that also the random weight initialisation scheme suffers in accuracy from the scheduler truncation and also the SoftMax numerical representation instability. In Figure 18 is the final results and have been referenced to the baseline performance. The fixed value scheme has been disregarded as it was so low performing.

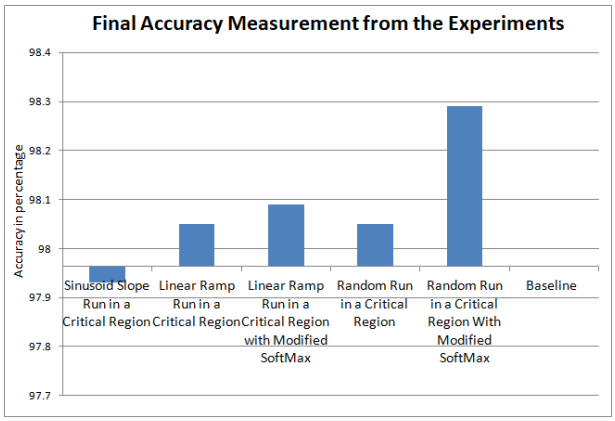

Fig 18: Final Perfected Results Graph

In Figure 18 the *"Random Run in a Critical Region With Modified SoftMax"* is the highest score, and is also higher than the original *"Baseline"*. The next highest score achieved is the *"Linear Ramp Run in a Critical Region with Modified SoftMax"* and is also above the *"Baseline"* performance. The Third highest score is jointly held by the *"Linear Ramp Run in a Critical Region"* and *"Random Run in Critical Region"*. While only the *"Sinusoid Slope Run in a Critical Region"* scored lower than the original *"Baseline"*.

X. Conclusion

In summary the initial original code has a mean accuracy of 97.964% and using random numbers, conventional thoughts might be that the weight values and random numbers were responsible alone for the variations in successive learning session results run to run. However, when the random seeds are set to a defined seed value the variation in the results continues in learning sessions. The paper was able to establish a stable set of results making the processing repeatable and deterministic in every learning session, and the baseline performance would increase to 98.05% and would be equally matched by a non-random scheme using a linear ramp that used the same Gloror ranges as the random initialisation in both dense layers of the architecture. A minor numerical floating point representations stability may be present in the SoftMax function, and replacing this function with an alternative that had numerical scaling, had the effect of increased accuracy in all experiment cases. A thirds source of random noise variation was discovered from scheduling truncating the stored values between schedules affecting the integrity of the internal 80bit extended floating-point precision register. The solution to task scheduler learning session variation was to define critical regions of code that are uninterruptable preserving the 80bit extended floating-point precision register integrity. All operating systems that are using Intel processors and an internal extended precision register may have this problem. This source of learning session variation is seen with task scheduling on CPUs with extended floating-point precision registers that are internal like Intel PC processors, although it should be noted that GPU results also may have a different result again as the GPU implementations are different, and have special Floating point Fused Multiply Add (FFMA) features for 32 and 64 bit floating point calculations to preserve precision resolution in consecutive calculations. Although GPUs may not be available in all Smart City hardware deployments and developments.

The paper also tested a variety of initialisation schemes and when the solution to the learning session variation was applied an equality to the random number accuracy was achieved at 98.05%, between a random number initialisation and non-random number weight initialization scheme that used a linear ramp in the same Glorot limit range. With the addition of a modified SoftMax function the non random linear ramp scheme achieved a further 0.04% accuracy. But however, when the same modified SoftMax function was applied to the random initialisation a further 0.324% increase was seen. But it should be noted that more optimal non-random schemes may exist, but the paper has shown that random number initialisation is not an imperative requirement. It may be that the number step interval sequence could be optimised, as the difference between the random scheme and the non random scheme is the interval step between the values. Although the placement arrangement of the value steps should not be significant numerically to a fully connected layer in a dense layer. However the placement of the values may be more intuitive to understanding the learning that has occurred if they are numerically ordered.

Furthermore, it is also possible that the learning session variations may be contributing to a regularisation effect to help to not over fit a model by reducing significant bit resolution, and a comparison with the original code with critical regions and no critical regions shows that the effect can also be a loss in accuracy as even random schemes benefit from the critical regions. This paper has looked at dense layers and it may follow that initialisation schemes could be set depending on the layer type, the activation function and the regularisation scheme used. The 80bit floating point representation does have benefits to achieved accuracy with the Glorot limit range, but there are also benefits to determinism in successive run results and that may have benefits to make a Deep Learning network capability accessible to safety critical systems with public liability concerns in smart city applications.

Moreover, the paper has demonstrated deterministic repeatable results in successive runs without random initialisations meaning that safety critical smart city applications with public liability concerns may have the test and qualification determinism required by those applications and the test environment is viable for further experiments.

But it was the research question of *"Are random number initialisation weight values required as the initial state before learning to have high accuracy in predictions,*

or can a non-random scheme also have comparable performance in those predictions" and the answer is that it is possible to use non-random sequences with comparable performance. Also random numbers are not an imperative requirement for accuracy performance. But where learning session variations exist as a result of task scheduling defining critical regions will benefit both random and non-random schemes. The critical regions will also allow a model to arrive at the optimised answer in a single learning session, thus reducing development time, and simplifying further relearning sessions that are in response to environmental changes.

Although, it is also possible that coupling in those schemes may connect with: the deep learning architecture, the layer type and the activation function used when not using dense layers. Also the numerical ordering of the weights in dense layers at the outset may provide more understandable learning transitions of the weights after learning to provide repeatable deterministic results and a better organisation for analysing the receptive field from the categorisation back through the dense layers after a learning session. That might be a support to safety critical systems that have public liability concerns with verification and validation obligations going forward in hazard avoidance smart city applications.

An important benefit to using the non-random initialisation scheme in dense layers is that it structures the weights after learning to have an order of influence arrangement in the neuron order. This is because the initialisation scheme provided a coupling of the node position to the initial influence that each node has. This is viable in dense layers because they are fully connected so the sequencing order of weight values is not significant to the result only the number range and interval. But however from a safety critical validation perspective the ordering of influence in the result promotes the understanding of weights and the influence values that combine for each pixel in learning. In appendix A an image representation of the weights is given for the highest score random scheme and linear ramp in both dense layers, and the weight structure along the neuron axis is striking in the non random scheme in the 1st dense layer, of which its' structure is inherited in the tensor length axis in the 2nd dense layer.

XI. Recommendations

It may follow that in training a neural network with a smart city application the definition of critical regions within the code for model fitting and evaluation can provide repeatable deterministic loss and accuracy scores which is particularly valuable when setting hyper parameters. It may also follow that the 80 bit extended floating-point precision register would provide truncation resulting in higher losses and therefore lower accuracy. The use of the critical regions may also be considered in prediction, as in public liability applications the prediction performance also needs to be assured.

When GPUs are not available, in either development or deployment the variation in results between learning session and prediction should not be ignored and indeed can be resolved.

The SoftMax numerical bit representation can be enhanced although the amount of enhancement achieved may be relative to the decimal position difference experienced in the datasets and also the model configuration, but using the scaling may make a model more accurate when configuring it.

XII. Further Research

A limitation of this research is that it has used a single model using dense layers. Couplings of the initialisation scheme to the layer architecture, activation function, optimisation and regularisation used is an area of further research.

It also may be that the initialisation scheme of the biases could also have a benefit, and the matching of weight and bias initialisation schemes to activation functions and architectures is also a subject of further research.

Using a non-random initialisation scheme appears to show that the number range seems to be more important than gradient changes. However with dense layers it may be the numerical steps between values that is important and a more optimal non-random initialisation value interval step scheme may be found.

XIII. References

[1] Pramathi J Navarathna, Vindhya P Malagi, "Artificial Intelligence in Smart City Analysis", Smart Systems and Inventive Technology (ICSSIT) 2018 International Conference on, pp. 44-47, 2018

[2] Srivastava, S., Bisht, A. and Narayan, N. (2017). Safety and security in smart cities using artificial intelligence — A review - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/7943136/citations#citations [Accessed 7 Dec. 2019].

[3] Knight, John. (2002). Safety Critical Systems: Challenges and Directions. Proceedings - International Conference on Software Engineering. 547 - 550. 10.1145/581339.581406.

[4] Serban, A. (2019). Designing Safety Critical Software Systems to Manage Inherent Uncertainty - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/8712362 [Accessed 7 Dec. 2019].

[5] Carpenter, P. (2019). Verification of Requirements for Safety-Critical Software. [online] Sigada.org. Available at: http://www.sigada.org/conf/sigada2001/private/SIGAda2001-CDROM/SIGAda1999-Proceedings/p23-carpenter.pdf [Accessed 7 Dec. 2019].

[6] Pratiwi, M., Harefa, J. and Nanda, S. (2015). Mammograms Classification Using Gray-level Co-occurrence Matrix and Radial Basis Function Neural Network. [online] https://www.sciencedirect.com. Available at: https://reader.elsevier.com/reader/sd/pii/S1877050915018

694?token=36245E12CFA8B44446DEC48A67D0E6E4AC6BF4776E9C4C39C4E5708EE59181F68874F21E4F1C6A47B7D8F3ED0B088D65 [Accessed 7 Dec. 2019].

[7] Borg, M., Englund, C., Wnuk, K., Duran, B., Levandowski, C., Gao, S., Tan, Y., Kaijser, H., Lönn, H. and Törnqvist, J. (2018). Safely Entering the Deep: A Review of Verification and Validation for Machine Learning and a Challenge Elicitation in the Automotive Industry. [online] arXiv.org. Available at: https://arxiv.org/abs/1812.05389 [Accessed 7 Dec. 2019].

[8] Ding, J., Hu, X. and Gudivada, V. (2017). A Machine Learning Based Framework for Verification and Validation of Massive Scale Image Data - IEEE Journals & Magazine. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/7875094 [Accessed 7 Dec. 2019].

[9] Rudas, I. and Horvath, L. (2002). Modeling man-machine processes in CAD/CAM and flexible manufacturing systems - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/570602 [Accessed 7 Dec. 2019].

[10] Chan, H., Rice, E., Vayanos, P., Tambe, M. and Morton, M. (2017). Evidence From the Past: AI Decision Aids to Improve Housing Systems for Homeless Youth. [online] Aaai.org. Available at: https://www.aaai.org/ocs/index.php/FSS/FSS17/paper/viewPaper/15995 [Accessed 7 Dec. 2019].

[11] Zetumer, S. and Harris, H. (2017). Identifying strangulated small bowel obstruction with machine learning. Journal of Clinical and Translational Science, [online] 1(S1), p.19. Available at: https://www.cambridge.org/core/journals/journal-of-clinical-and-translational-science/article/2476/5ABFA40F9D23FBA20FDB046363E8971E [Accessed 7 Dec. 2019].

[12] Sarif, S. and Zaibon, S. (2012). Decisional Guidance for Computerised Personal Decision Aid (ComPDA). [online] Kmice.cms.net.my. Available at: http://www.kmice.cms.net.my/ProcKMICe/KMICe2012/PDF/CR206.pdf [Accessed 7 Dec. 2019].

[13] Singh, M., Geyer, P. and KU, L. (2019). Statistical decision assistance for determining energy-efficient options in building design under uncertainty. [online] https://lirias.kuleuven.be. Available at: https://lirias.kuleuven.be/retrieve/542316 [Accessed 7 Dec. 2019].

[14] De Souza, J., de Francisco, A., Piekarski, C. and do Prado, G. (2019). Data Mining and Machine Learning to Promote Smart Cities: A Systematic Review from 2000 to 2018. [online] Mdpi.com. Available at: https://www.mdpi.com/2071-1050/11/4/1077/pdf [Accessed 28 Sep. 2019].

[15] Nosratabadi, S., Mosavi, A., Keivani, R., Ardabil, S. and Aram, F. (2019). State of the art survey of deep learning and machine learning models for smart cities and urban sustainability. [online] Preprints.org. Available at: https://www.preprints.org/manuscript/201908.0154/v1/download [Accessed 28 Sep. 2019].

[16] Ullah, I., et al., Smart Lightning Detection System for Smart-City Infrastructure Using Artificial Neural Network. Wireless Personal Communications, 2019. 106(4): p. 1743-1766.

[17] Rudd-Orthner, R. and Milhaylova, L. (2019). Numerical Discrimination of the Generalisation Model from Learnt Weights in Neural Networks. Annals of Emerging Technologies in Computing, [online] 3(4), pp.1-14. Available at: https://www.researchgate.net/publication/335023149_Numerical_discrimination_of_the_generalisation_model_from_learnt_weights_in_neural_networks.

[18] FAN, F. and WANG, G. (2018). IEEE Xplore Full-Text PDF:. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8350369 [Accessed 7 Dec. 2019].

[19] Duch, W., Adamczak, R. and Jankowski, N. (1997). Initialization and optimization of multilayered perceptrons. [online] Wwwold.fizyka.umk.pl. Available at: http://wwwold.fizyka.umk.pl/publications/kmk/initmlp.pdf [Accessed 14 Dec. 2019].

[20] Kakaraparthi, V. (2018). Xavier and He Normal (He-et-al) Initialization. [online] Medium. Available at: https://medium.com/@prateekvishnu/xavier-and-he-normal-he-et-al-initialization-8e3d7a087528 [Accessed 5 Oct. 2019].

[21] Ananthram, A. (2018). Random Initialization For Neural Networks : A Thing Of The Past. [online] Medium. Available at: https://towardsdatascience.com/random-initialization-for-neural-networks-a-thing-of-the-past-bfcdd806bf9e [Accessed 5 Oct. 2019].

[22] Melen, R., Sartori, F., & Grazioli, L. (2015). Modeling and understanding time-evolving scenarios. JOURNAL OF SYSTEMICS, CYBERNETICS AND INFORMATICS, 13(5), 62-67.

[23] Ernest et al, N. (2019). Genetic Fuzzy based Artificial Intelligence for Unmanned Combat Aerial Vehicle Control in Simulated Air Combat Missions. [online] Research Gate. Available at: https://www.researchgate.net/profile/Nicholas_Ernest/publication/301944635_Genetic_Fuzzy_based_Artificial_Intelligence_for_Unmanned_Combat_Aerial_Vehicle_Control_in_Simulated_Air_Combat_Missions/links/576c4e5408ae193ef3a9a384/Genetic-Fuzzy-based-Artificial-Intelligence-for-Unmanned-Combat-Aerial-Vehicle-Control-in-Simulated-Air-Combat-Missions.pdf [Accessed 29 Jan. 2019].

[24] Freitas Jr. et al, R. (2019). Advanced Automation for Space Missions. [online] Rfreitas.com. Available at: http://www.rfreitas.com/Astro/AASMJAS1982.htm [Accessed 29 Jan. 2019].

[25] Lawson, D. and James, M. (2019). SHARP: A multi-mission artificial intelligence system for spacecraft telemetry monitoring and diagnosis. [online] Ntrs.nasa.gov. Available at: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19900009128.pdf [Accessed 29 Jan. 2019].

[26] Allaire, J. (2019). rstudio/keras. [online] GitHub. Available at: https://github.com/rstudio/keras/blob/master/R/initializers.R [Accessed 2 Jan. 2019].

[27] Niederberger, B. (2019). Set Process Priority In Windows « Python recipes « ActiveState Code. [online] Code.activestate.com. Available at: https://code.activestate.com/recipes/496767-set-process-priority-in-windows/ [Accessed 2 Jan. 2019].

[28] Shimao (2019). What process controls the CPU affinity of new python processes. [online] Super User. Available at: https://superuser.com/questions/1273705/what-process-

controls-the-cpu-affinity-of-new-python-processes [Accessed 2 Jan. 2019].
[29] Herf, M. (2000). stereopsis: know your FPU. [online] Stereopsis.com. Available at: http://stereopsis.com/FPU.html [Accessed 29 Jan. 2019].
[30] Alvas (2019). How to implement the Softmax function in Python. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/34968722/how-to-implement-the-softmax-function-in-python [Accessed 2 Jan. 2019].

XIV. Appendix A

In Figure 19 is an image of the 1st dense layer using the random initialisation scheme and it can be notice that it has a uniform random structure.

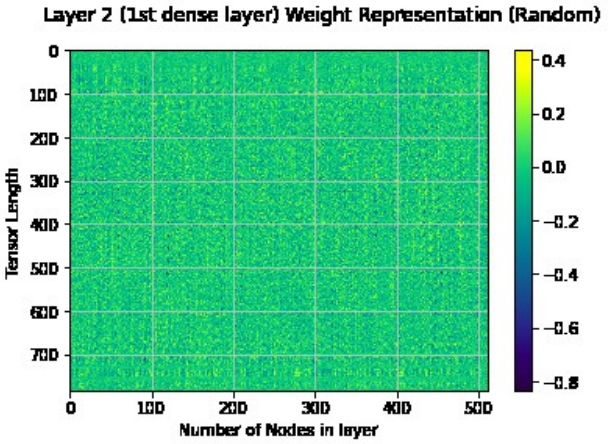

Fig 19: 1st Dense Layer as an Image using Random Scheme

In Figure 20 is an image of the 1st dense layer but this time using the non-random linear ramp initialisation scheme and it can be notice that it has a structure along the *"Number of nodes"* axis as the initialisation scheme provided a ordering in the nodes that developed in learning. This is as a result of the linear ramp that has grown the weights in adjacent addresses, these weights have been effected by the initial condition and are ordered such that influence in pixels are aligned.

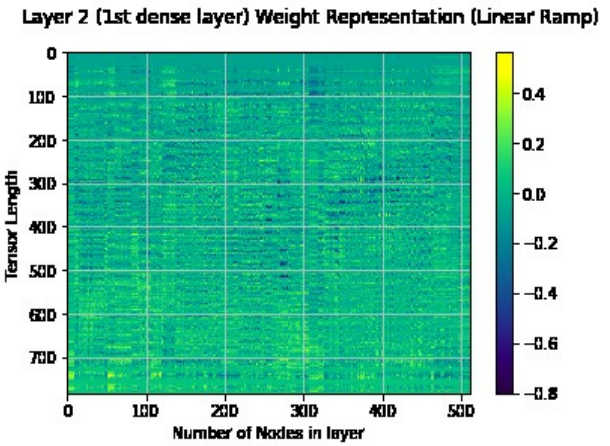

Fig 20: 1st Dense Layer as an Image using Linear Ramp

In Figure 21 is an image of the 2nd dense layer at the classifier output stage, using the random initialisation scheme, and it can be notice that it has a uniform random structure. i.e. is more random then Figure 22 as the random ordering has re-arranged the weights.

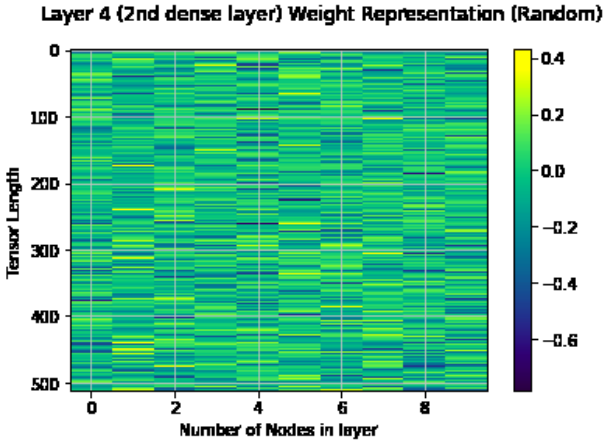

Fig 21: 2nd Dense Layer as an Image using Random Scheme

In Figure 22 is an image of the 2nd dense layer at the classifier output stage, but using the linear ramp initialisation scheme instead, and it can be notice that it has a structure that was inherited from the node order of the first dense layer in the tensor length axis.

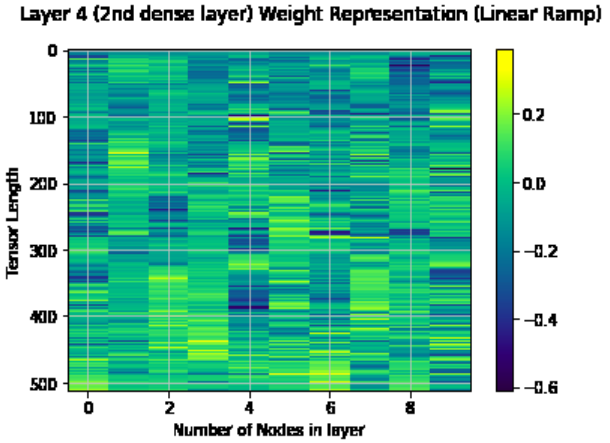

Fig 22: 2nd Dense Layer as an Image using Linear Ramp