Proceedings Paper:
Rudd-Orthner, R. orcid.org/0000-0002-2534-0920 and Mihaylova, L. orcid.org/0000-0001-5856-2223 (2021) Non-random weight initialisation in deep convolutional networks applied to safety critical artificial intelligence. In: 2020 13th International Conference on Developments in eSystems Engineering (DeSE). International Conference on Developments in eSystems Engineering (DESE), 14-17 Dec 2020, Liverpool, UK. IEEE , Liverpool, United Kingdom , pp. 1-8. ISBN 9781665422383

https://doi.org/10.1109/DeSE51703.2020.9450242

Non-Random Weight Initialisation in Deep Convolutional Networks Applied to Safety Critical Artificial Intelligence

Richard N M Rudd-Orthner[1], Lyudmila Mihaylova[1]
[1] University of Sheffield, Sheffield, UK, {RNMRudd-Orthner1, L.S.Mihaylova}@sheffield.ac.uk

Abstract—**This paper presents a non-random weight initialisation scheme for convolutional neural network layers. It builds upon previous work that was limited to perceptron layers, but in that work repeatable determinism was achieved with equality in categorisation accuracy between the established random scheme and a linear ramp non-random scheme. This work however, is in Convolutional layers and are the layers that have been responsible for better than human performance in image recognition. The previous perceptron work found that number range was more important rather than the gradient. However, that was due to the fully connected nature of dense layers. Although, in convolutional layers by contrast, there is an order direction implied, and the weights relate to filters rather than image pixel positions, so the weight initialisation is more complex. However, the paper demonstrates a better performance, over the currently established random schemes with convolutional layers. The proposed method also induces earlier learning through the use of striped forms, and as such has less unlearning of the traditionally speckled random forms. That proposed scheme also provides a higher performing accuracy in a single learning session, with improvements of: 3.35% un-shuffled, 2.813% shuffled in the first epoch and 0.521% over the 5 epochs of the model. Of which the first epoch is more relevant as it is the epoch after initialisation. Also the proposed method is repeatable and deterministic, which is also a desirable quality for safety critical applications within image classification. The proposed method is also robust to He initialisation values too, and scored 97.55% accuracy compared to 96.929% accuracy with the Glorot/ Xavier in the traditional random forms, of which the benchmark model was originally optimised with.**

Keywords— Repeatable, Weight Initialization, Information Assurance, Convolutional Layers.

I. Introduction and Background Work

Convolutional layers in neural networks have been used in Artificial Intelligence (AI) applications, and led to the use of multiple layers separated by non-linearity functions. This layering of hidden layers are said to be deep, and the successes of that architecture led to the Deep Learning research thread. The convolutional layers may have translation to brain anatomy with respect to Hubel and Wiesel [1]. Whom examined spider monkey's brain activity when under a light anaesthetic, while stimulating the retina with images of spots, stripes and patterns. Convolutional layers have had biological inspirations, and are generally accepted as providing hierarchical feature extraction in a Deep Convolutional network [2, 3]. Later in 2017 Alex Krizhevsky's paper [4] would prove to become an influential paper, and demonstrated better than human performance with image categorisation in the image net challenge 2012 using Deep Convolutional networks. Convolutional Neural Networks have played an influential role ever since. Although, applications of convolutional layers provide important human level capabilities, but they have not been embraced into mission critical applications [5 - 8], owing in part to learning session variations, and certification of the network content as complete and correct. Current random initialisation schemes provide a cross-validation variation in accuracy that is visible over regularisation. To this end, the previous published background work to this paper, also examined repeatable determinism, but in perceptron layers, and proposed a method, although was tightly coupled to the perceptron layers only [9]. However, that paper, resolved a numerical stability issue for repeatability, and also proposed a non-random scheme for determinism, and achieved an almost equal performance in accuracy proving viability. That work was furthered in a journal published version [10], which used the Glorot/Xavier [11], limit values with those non-random schemes and this time achieved an equal accuracy performance to the random scheme. That work also had the benefit of ordering the weights after learning, into a structured form along the number of neurons axis and highlighted the correlation in structure at pixel indexes. See Figure 1 left for an image of the learnt weights with the random scheme, and right for the non-random scheme.

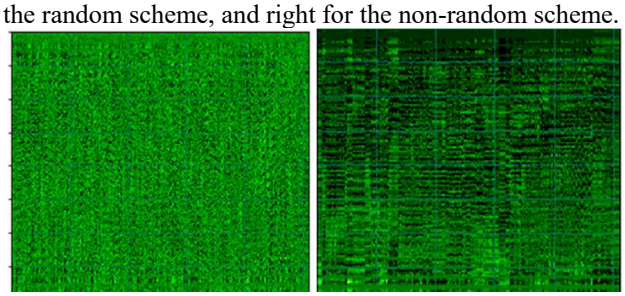

Fig. 1 Weights after learning results from the previous work [10].

It was noted in that previous work [10], that both weight sequences have an equivalence in performance, but the non-random scheme has a structure that may have a benefit for rule extraction. As the weights have been grown in an ordered sequence along the number of neurons (in the x plane), and shows activation correlations at pixel positions (in the y plane), and that helps to generalise in a rule extraction approach as the pixel activations have been clustered to neighbours. However, both those previous papers [9, 10] were confined to perceptron layers and this paper furthers that work into convolutional layers. That earlier work in perceptron layers did prove that an equal performance of a non-random weight initialisations scheme was viable, and that random number initialisation is not necessary. Which is the same assertion of Blumenfeld et al. too [11], in an experiment of zeroing of some of the weights in a convolutional layer, however zeroing of weights is not this paper's approach. Furthermore, the order of weights in the background perceptron layers work was not significant, due to the fully connected links of nodes, and in convolutional layers the weights relate to filters that slide in a direction, and so the order is significant. So that previous form is not applicable to convolutional layers.

A. *Contemporary Related Work*

Ding et al. [12], propose a shuffle leap frog algorithm approach, for the update and initialisation with random Gaussian forms in the area of Fundus Images lesions. The approach presented in that paper contains random numbers, initially in a Gaussian distribution optimised with the shuffle leap frog algorithm, where as the approach presented in this paper, does not contain random numbers. Wang et al. [13], propose a 2D Principle Component Analysis (2DPCA) approach to the initialisation of convolutional networks to adjust the weight difference values to promote back propagation. This approach avoids the use of random numbers, and uses samples of the dataset instead making it convergent to sample data seen. However, in our work the approach is not connected to the sample data, only the architecture in terms of filter geometries and layer types used. Ferreira et al. [14], examines weight initialisation using a Denoising Autoencoder (DAE) in the field of classifying tumour samples through dataset sampling, this is also a convergent data sample approach.

B. *Contribution and Novelty of this Paper*

This paper's contribution is a proposed alternative method, for generating a non-random sequence for the initialisation of convolutional layers rather than perceptron layers. Where the proposed non-random sequence has formations of stripes and dots in that initialisation state, and as such is predisposed to the application, It also allows earlier learning, to lower the loss quicker, without using data sampling. The proposed method also arrives at a higher performing accuracy in the first learning session, that is repeatable and deterministic. Where as in the established random number schemes several learning sessions are required to establish which learning session conducted has provided the best accuracy from a variation of random initialisation states.

C. *Structure of the paper*

This papers structured as follows: Section one is the introduction to the area and both the background and other contemporary related works. Section two is the benchmark baseline model, and is comparable to the background work in perceptron layers. It also presents the baseline results of that model, reusing the critical region code from the background work to re-produce a 'repeatable' learning session. Section three compares the established random initialisation method as a benchmark to the proposed method. Section four explains the number of weights and image size in each layer of the benchmark model, so the proposed method can be explained and the Glorot/Xavier limit values calculated. Section five shows the weight differences of the established random scheme and the proposed non-random scheme, before and after learning with an illustration of what has been learnt. Section 6 presents the proposed non-random scheme method that achieved a higher accuracy score and explains the design, and also verifies robustness to He et al. initialisation limits. Section seven is the discussion and conclusions.

II. Benchmark Baseline Model and Method

In the foundation work, perceptron layers and the MNIST dataset [15], were used as an example that is familiar to researchers, and to demonstrate non-random weight initialisations the same application and dataset is used, but in a convolutional form. This is the benchmark that has been optimised by researchers, and so comparisons can be made between the proposed scheme and the previous benchmark too. In Figure 2 is the architecture of the convolutional layer form of the benchmark model.

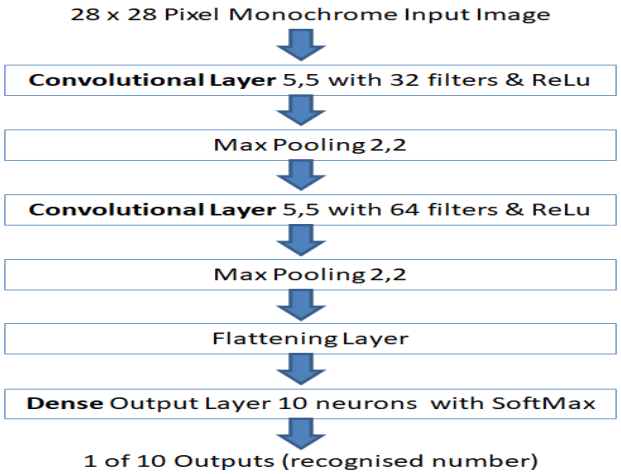

Fig. 2 Architecture of the Benchmark Model.

The model architecture is the equivalent of the perceptron layer foundation work's benchmark, but in a

convolutional layer form, and as such forms a comparison bridge to the background work in perceptron layers.

Using the repeatable critical-region code from the background work, that removed a source of numerical instability in learning session variation, that was identified in the original perceptron repeatable determinism paper [9]. The benchmark results are in Table 1, and that benchmark model is using the Glorot/Xavier random number scheme initialisation as per its' definition within Keras by Torres [16] and has a stated accuracy of about ~97%. Although it should be noted, that there are higher scoring models using the MNIST dataset in a convolutional form, but a high accuracy score of 99.8% [17], provides little-head room to show an improvement. That model also requires 50 epochs, and that is along learning duration beyond the initial condition in this context, where the random shuffle may be a more dominant random effect. This table forms the experiment control from the Torres baseline as a benchmark.

TABLE 1
BENCHMARK RANDOM INITIALISATION RESULTS.

Seeded	Epochs	Accuracy	Loss
Yes	5 Shuffled	**97.1%**	**0.09996312594935299**
No	5 Shuffled	96.9%	0.10337305160537362
No	5 Shuffled	96.97%	0.10660055329352618
No	5 Shuffled	96.84%	0.10624442052207887
No	5 Shuffled	96.96%	0.10476687839999795
No	5 Shuffled	96.84%	0.1067980943121016
No	5 Shuffled	97.08%	0.10235729512907564
No	5 Shuffled	96.87%	0.10271317201182247
No	5 Shuffled	96.73%	0.10741757678128779
No	5 Shuffled	97.0%	0.10189960528686642
Averages:		96.929%	0.104213377329148
Yes	1 Shuffled	**91.68%**	**0.2875490588128567**
No	1 Shuffled	91.68%	0.28503509197235105
No	1 Shuffled	91.5%	0.2884808440446854
No	1 Shuffled	91.33%	0.29792458724975585
No	1 Shuffled	91.25%	0.30420758697986605
No	1 Shuffled	91.64%	0.29925555539131166
No	1 Shuffled	90.9%	0.31180655114650724
No	1 Shuffled	91.2%	0.3049829860568047
No	1 Shuffled	90.84%	0.3010126953959465
No	1 Shuffled	90.65%	0.30433687148690225
Averages:		91.267%	0.298459182853698
Yes	1 No Shuffle	**89.87%**	**0.32091661343574523**
No	1 No Shuffle	89.69%	0.31784898174405096
No	1 No Shuffle	89.29%	0.33380672977566717
No	1 No Shuffle	90.46%	0.30266491156816483
No	1 No Shuffle	90.11%	0.31882508975863455
No	1 No Shuffle	90.25%	0.313027676063776
No	1 No Shuffle	90.76%	0.3063296886742115
No	1 No Shuffle	89.84%	0.3295604445934296
No	1 No Shuffle	89.55%	0.3244086824059486
No	1 No Shuffle	89.48%	0.32875925261974337
Averages:		89.93%	0.319614807063937

The results in Table 1 show the benchmark results with the full 5 epochs shuffled, and reaching the approximate stated accuracy of that model, but also shows just 1 epoch, as the 1st epoch after initialisation is of interest. Those 1st epoch runs are also in two forms which are: with or without the shuffle, as there are two random effects (weight initialisation and shuffle order), and this allows those effects to be distinguished. So that when no shuffle is used there is only the effect of weight initialisation, and the shuffled version shows the equivalence to the 5 epoch results where only shuffles forms are valid to have reordering in each of the 5 epochs. When the random number generator is seeded (shown in bold) the results are completely repeatable from learning session to learning session. But also more results have been added, not using the seeding of the random number generator, to show the accuracy variation that different random number initialisation sequences have, that are visible over regularisation. As the random number generator seeding has two affects: the weights initialisation values and the shuffle reorganisation of the dataset. For that reason, table 1 shows results from three configurations: 5 epochs with shuffles, a single epoch with the shuffle and a single epoch with no shuffle as that is the effect of the random number initialisation in the weights alone. Those results show an average accuracy of 89.93% in a single epoch with no shuffle, an average of 91.267% (+1.337%) when a shuffle is used, and an average accuracy of 96.929% (+5.6625% greater) with the use of 5 shuffled epochs.

III. COMPARISON OF THE BENCHMARK WITH THE PROPOSED METHOD

The presented non-random initialisation scheme will achieve 93.28% in a single epoch with no shuffle, +3.35% better then the random scheme. 94.08% accuracy will be achieved when a shuffle is used in that single epoch and again using the non-random scheme which is a 2.813% gain over the benchmark with the random scheme. Then 97.45% (+0.521% over the benchmark) when 5 epochs are used. Those results are summarised in Table 2.

TABLE 2
BENCHMARK GAINS OF THE NON-RANDOM INITIALISATION SCHEME.

Epochs	Accuracy	Loss	Benchmark Gain
5 Shuffled	**97.45%**	0.0859082410749048	**+0.521%**
4 Shuffled	**97.07%**	0.09643121291957796	N/A
3 Shuffled	**96.54%**	0.11417882204577327	N/A
2 Shuffled	**96.01%**	0.14150242246389388	N/A
1 Shuffled	**94.08%**	0.2201271739989519	**+2.813%**
1 No Shuffle	**93.28%**	0.23073574766814708	**+3.35%**

It is worth noting, that the best gain is achieved in the first epoch, which is the epoch that occurs after the weight initialisation. Less relative gain is achieved in further epochs, as the learning is occurring longer after the initialisation in the subsequent epochs, diminishing its influence. An interpretation is the subsequent learning is more equivalent but earlier learning has a higher benefit.

IV. UNDERSTANDING THE WEIGHTS AND IMAGE SIZES

However, to understand the presented non-random weight scheme, the structure of the weights and the image size, in the benchmark model need to be understood clearly. To understand how the weights are used is critical, to understanding how convolutional layers use the weights and effect the image size. This is quite different

from perceptron layers, and as in the findings of the journal version of the perceptron repeatable determinism paper [10]. When using the Glorot/Xavier limits [18], the results were enhanced with the non-random scheme over the results initially presented in the conference paper [9]. This was because of the tolerance and matching to the model architecture in terms of propagation value limits. Thus, also here the Gloror/Xavier value limits need to be calculated, Figure 3 demonstrates the adjustment of image and weights sizes in the benchmark model.

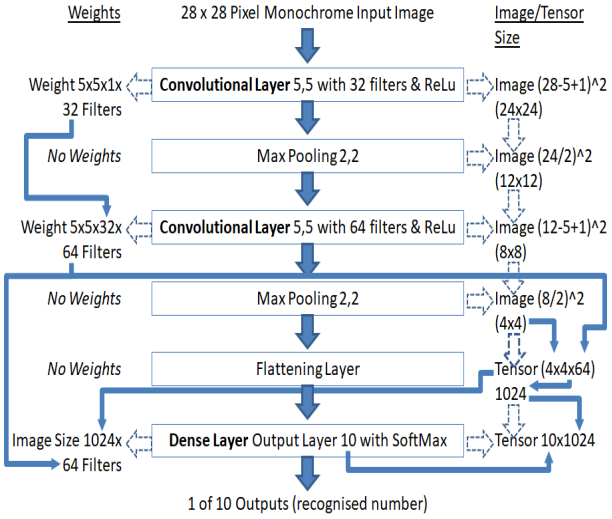

Fig. 3 Weight & image size adjustments.

Convolutional layers use the weights for the filters, and not the pixels, As such their dimensions are: column by row of the filter, then by depth (channels), where that depth may be inherited from the previous convolutional layer's number of filters. Perceptron layers use the image size by previous layer's filters, as the previous layers filter would have translated to depth, then that is multiplied by the number of neurons. Table 3 shows the weight and image size at each layer in the benchmark model.

TABLE 3
WEIGHT AND IMAGE DATA SIZES.

Layer	Filter/Pool /Neurons	Depth	Image/ Tensor Size	Weights
Input 28x28x1	N/A	1 (B/W image)	28x28 (748)	N/A
Conv Layer 1	5 by 5 by 32 filers	1	24x24 (576)	800
Max Pooling	2 by 2	32	12x12 (144)	N/A
Conv Layer 2	5 by 5 by 64 filters	32	8x8 (64)	51200
Max Pooling	2 by 2	64	4x4 (16)	N/A
Flatten Layer	N/A	1	1x(4x4x64) 1024	N/A
Dense Layer	10	1	10x1024 (10240)	10240

To calculate the Glorot/Xavier limits, See Equations (1-3), and note that the calculated values have been rounded to 8 decimal places, and used as such and are shown as such in the Equations (1-3):

$$ConvLayer1 = \sqrt{\frac{6}{(5*5*1x32)}} = 0.08660254, \quad (1)$$

$$ConvLayer2 = \sqrt{\frac{6}{(5*5*64+5*5*1*32)}} = 0.05 \quad \text{and} \quad (2)$$

$$DenseLayer = \sqrt{\frac{6}{(4*4*64+10)}} = 0.07617551. \quad (3)$$

Alongside the Glorot/Xavier value limits, the structure of the weight initialization sequence also requires to have positional variations in each filter. Those positional variations, are also to be aligned for feature extraction, in a Hubel and Wiesel [1] stripes intuition. That structure, is ideally to be more allied to edge detection than a random value placement as the start condition. This is to predisposed the initial condition to the application of image classification, and outperform the random scheme, and thus reduce dataset sizes by inducing earlier learning, by less unlearning of the initial state.

V. COMPARISON OF THE WEIGHTS BEFORE AND AFTER LEARNING

In Figure 4, is the random scheme benchmarks weights before learning of the first convolutional layer, and in Figure 5, are the same filter weights but after learning.

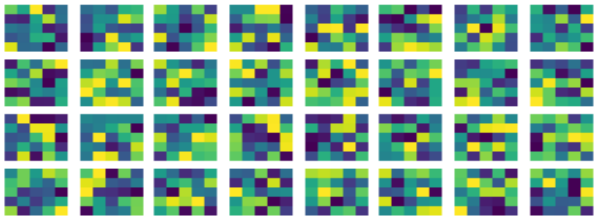

Fig. 4 Initial Weights in first Conv Layer with the Random Scheme.

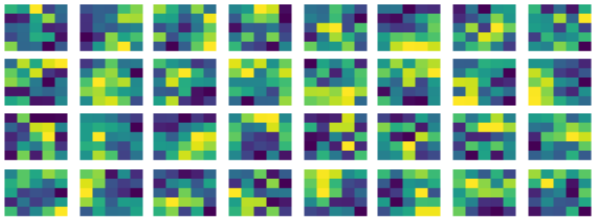

Fig. 5 Learnt Filter Weights in first Conv Layer with the Random Scheme.

It may be noted that there are similarities in some of the filters, between the before and after learning. Suggesting that the initial condition has a dominant effect in the subsequent learning. However, looking carefully, adaption can be notice between them. If the Initial weights are subtracted from the learnt weights, the adaption can be seen more clearly in Figure 6.

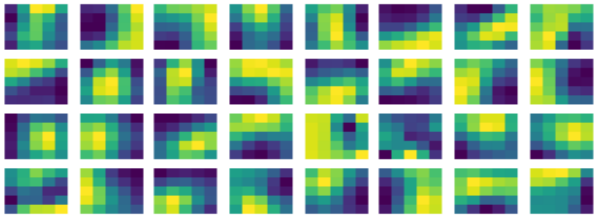

Fig. 6 Learnt Filter Weight Updates of the with Random scheme.

Consistent with Figure 6, convolutional filters examined after learning may be expected to have stripes, spots and perhaps curves, that may be used in edge detection of feature extraction, that will be hierarchically connected to form shapes in later layers. Also convolutional layers, offer the ability to change the image resolution, at which a filter and the subsequent layers operate. Owing to this, striped and spotted patterns may be more conducive in shape detection, that vary from filter to filter. Figure 7 shows the non-random weight filter initialisation of the proposed scheme, produced for the first convolutional layer, and Figure 8 shows the same weights of the filters but after learning.

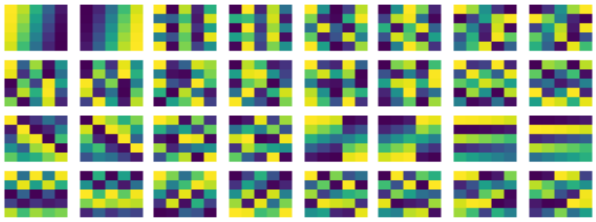

Fig. 7 Filter Initialisation in first Conv Layer with the proposed scheme.

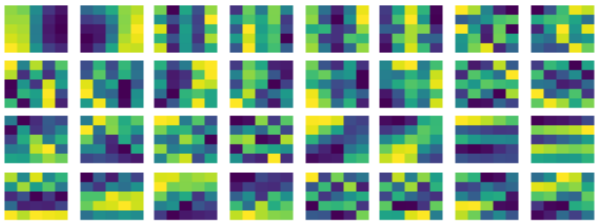

Fig. 8 Learnt Filter Weights in first Conv Layer with the proposed scheme.

It may be noted that there are also striking similarities between the before and after learning of the proposed scheme, but this scheme is higher performing then the random scheme. There are also similarities between the after learning of the random and non-random scheme. However, again if you subtract the original initialisation from the learnt weights, what has been learnt can be shown, or rather what has been added to the initial condition in learning, as shown in Figure 9.

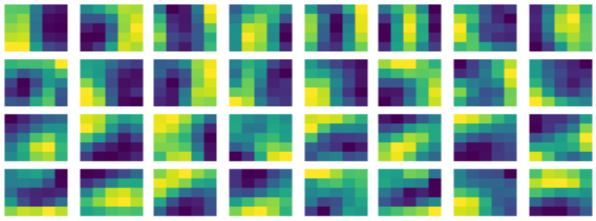

Fig. 9 Learnt Filter Weight Updates of with the proposed scheme.

The initial condition therefore has an effect of their arrangement from the outset of learning, and different arrangements will affect the after learning result. So the proposed initialisation method makes arrangements that have stripes, spots and curves that are different in each filter, such that the subsequent learning adapts to the dataset quicker. These arrangements are also different in each filter providing a filter diversity of edge detection, in different orientations.

VI. The Proposed Scheme

So as to explain the design of the scheme, and how its derived. Firstly, the positional variation of values is important as it relates to a filter sweeping across the pixels. If the modulation of arrangement position is based on two as a vector, then it may relate to stripes as a 2D matrix when different column values are used, and that is also connected to the resolution in that filter. That striping can then be controlled by the convolutional layer's hyper-parameters. As also the stripe orientation in different filter arrangements is important. An algorithm published in the papers [19, 20], that produced a least adjacent arrangement based on a modulation of two for dataset shuffling, has some attractive properties in this application. It was originally an alternative to the established random shuffle approach. This non-random shuffle approach, rearrange the dataset to produce a sequence with the first half of the input, that was output at a stride of two and then in filled those gaps with the remaining vector, also at a stride of two but in reverse order. This resulted in a placement with smallest and largest numbers neighbouring each other at the start of the vector. Figure 10 shows, the sequence with a vector length of 10, and with the unordered in row 1 and the reordered in row 2.

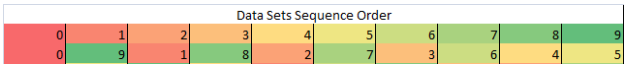

Fig. 10 Least Neighbour of vector length 10 [19-20].

This process can be repeated iteratively as an in place operation, and when done so, the original sequence order will repeat at iteration: vector length minus 1. See Figure 11, where row 1 and row 10 (the unordered, and 9th iteration of reordering), are the same due to that repeating nature.

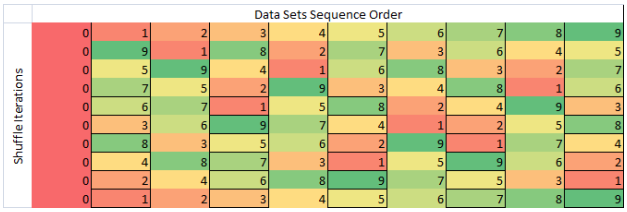

Fig. 11 Least Neighbour shuffle over 1 unordered and 9 reordering iterations [19-20].

Referring to Figure 11, it can be noted that there is a sliding shift in value placement with each iteration, and that those shifts occur in both diagonal slants, which is useful to provide tolerance with row and column bounding differences in different row, column defined filters. Another attractive feature of this reordering algorithm, is illustrated with a linear ramp of values. Where a re-orientation through 90 degrees occurs after the first iteration, and the iteration before the sequence is

repeated, at vector length minus 2 number of iterations. Figure 12 shows, the re-orientation from the 1st iteration sequence (in Black) and the penultimate iteration (length minus 2) (in Red), which is the iteration before the repeat in the sequence.

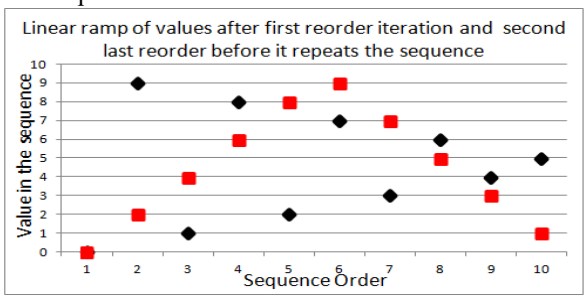

Fig. 12 Least Neighbour iterations: 1 and 8.

As the sequence repeats, that algorithm also has a fixed number of combinations so the number of unique filters can be estimated. Although, the algorithm from the paper [19, 20], is further enhanced to deal with odd number length vectors. The structured English for the algorithm is in Figure 13 as amended:

```
data_out = Function shuffle_data (vector: data)

    len = length(data)

    If ( mod(len, 2)  !=  0)
        data_out [len-1]  = data [ int( len/2 ) ]
    End If

    Loop For n = 0 to int( len / 2 ) - 1
        data_out[n*2]     = data[n]
        data_out[n*2+1] =data[len-1-n]
    End Loop

Return data_out
```

Fig. 13 Least Neighbour Shuffle Algorithm.

It may be noted, that this algorithm will always have the same value in the first location, and this was not significant in the shuffle application, but however, it is significant in this application. So experiments were conducted with pre-placement shift offsets in the data, and also with a data direction alternation of this algorithm. These experiments proved to not be as high performing, although did provide a higher number of unique filters. As with convolutional layers the order of filter values is significant, so an alternation of the data is conducted instead. So that every second filter is reversed or flipped and the memory is addressed through column, row and depth for the odd filter numbers, and vector address reversed as then depth, row column for the even filters. This provides two different filters of alternating direction placement of each shuffle iteration, doubling the number of unique filters on offer, with also a crucial disruption variation to the first position value.

As well as the order placement, the value distribution of the values in the initialisation sequence may also be significant, as images an less likely to be uniformly distributed. So experiments were conducted with linear ramps, as these had been the highest performing in perceptron layers. However, in this case the application of a sinusoidal slope was higher performing on convolutional layers, and the linier ramp in dense perceptron layers both still reordered by this same scheme. This may be because of the cos(x) content is a partial distribution of a sine function, or at least it's distribution has a match to convolutional layers and images. The structured English for the algorithm that calls the shuffle is in Figure 14.

```
Loop For d = 0 to LayerFilter-1
    n = 0
    m = FilterColumn * FilterRow * LayerDepth-1
    Loop For c = 0 to  LayerDepth-1
        Loop For a = 0 to FilterColum-1
            Loop For b  =  0 to FilterRow-1
                initval[a][b][c][d] = FunctionVal (n,m, Glorot, Type)
              n=n+1
            End Loop
        End Loop
    End Loop

    initvalTrans = transpose(initval)
    If mod(d+1,2) == 0
        initvalTransvect =reshape(initvalTrans[d], (1,5*5*1))
        initvalTransvect =flip(initvalTransvect)
        initvalTrans[d]  = reshape(initvalTransvect, (1,5,5))
    End IF
    Loop For n = 0 to int(d/2)-1
        initvalTransvect =.reshape(initvalTrans[d], (5*5*1, 1))
        initvalTransvect = shuffle_data(initvalTransvect)
        initvalTrans[d]  = reshape(initvalTransvect, (1,5,5))
    End Loop
    initva1 = transpose(initvalTrans)
End Loop
```

Fig. 14 Filter Re-order Organisation Algorithm.

The slope alternatives are as in the structured English in Figure 15.

```
value = function FunctionVal (n,m, GlorotLimit, Type)
    If Type = Convolutional
        value =  cos (n/m*pi())*GlorotLimit
    Else
        value =  n/m*(2* GlorotLimit)- GlorotLimit
    End If
Return Value
```

Fig. 15 Filter Slope Algorithm.

So in Summary the resulting initialisation weights, that were higher performing provided a sinusoidal bathtub distribution in convolutional layers, and a uniform distribution in the dense perceptron layer. Where the reordering provides uniquely reordered filters in each filter, with alternating vector directions and that reordering provides a two position base shift least neighbour arrangement. That least neighbour arrangement has a pattern reorientation from row column to column row that is progressive in each filter, with a matching filter direction alternative providing the numerical compliment weight arrangement. The total number of unique filters will be (filter_row × filter_column × depth - 1) × 2. If the losses are compared during learning, of the first epoch, then the loss does reduce quicker with the non-random scheme.

See Figure 16 for the random scheme and Figure 17 for the non-random scheme.

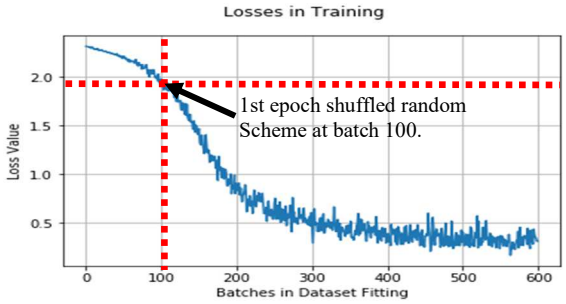

Fig. 16 Losses over Batches fitting Random Scheme Shuffled.

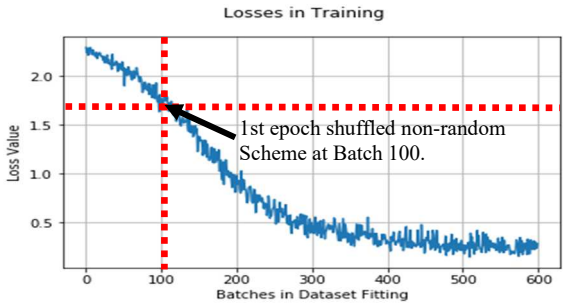

Fig. 17 Losses over Batches fitting proposed Scheme Shuffled.

It may be noted that in comparison at batch 100 the non-random scheme has achieved a lower loss. These results are also consistent if the shuffle is disabled making the comparison with only the difference in initialisation, and again the loss is less in the non random scheme at the batch 100 point (see Figure 18 and 19). Thus the earlier learning has benefited relatively.

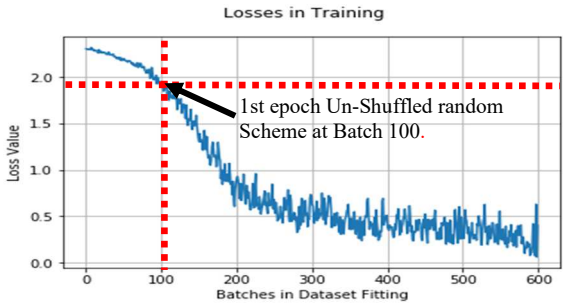

Fig. 17 Losses over Batches fitting Random Scheme not Shuffled.

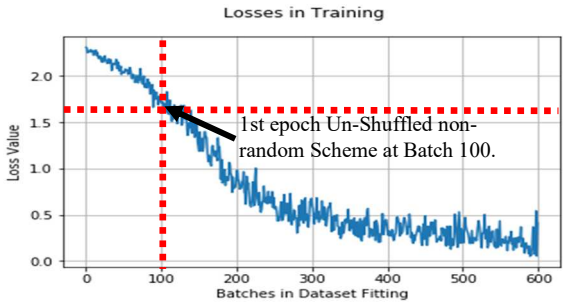

Fig. 18 Losses over Batches fitting proposed Scheme not Shuffled.

There is an enquiring question raised, and that is: although it is a departure from the benchmark model, would the proposed method be robust to He et al. initialisation limits instead. The He et al. initialisation limit values are calculated as in Equations (4-6):

$$ConvLayer1 = \sqrt{\frac{6}{(5*5*1x32)}} = 0.08660254, \quad (4)$$

$$ConvLayer2 = \sqrt{\frac{6}{(5*5*64)}} = 0.06123724 \text{ and} \quad (5)$$

$$DenseLayer = \sqrt{\frac{6}{(4*4*64)}} = 0.07654655. \quad (6)$$

As before the limit values are rounded to 8 places so as to be compatible in comparison. The results using the He et al. initialisation are presented in Table 4.

TABLE 4
HE ET AL. LIMITS WITH THE PROPOSED INITIALISATION SCHEME.

Epochs	He et al. Loss	He et al. Accuracy	Gain over Glorot
5 Shuffled	0.08277090748995543	97.55%	**+0.10%**
4 Shuffled	0.0928040012665093	97.22%	**+0.15%**
3 Shuffled	0.10946995529122651	96.64%	**+0.10%**
2 Shuffled	0.13462381604388357	96.17%	**+0.16%**
1 Shuffled	0.2057170445650816	94.53%	**+0.45%**
1 No Shuffle	0.21759726359546183	93.56%	**+0.28%**

The He et al. limit values are being used with the non-random initialisation proposed method, and as such is robust to the He et al. initialisation limits as well, and with further gains. In all epoch cases the He et al. initialisation with the proposed method offers a further positive accuracy gain, compared to the Glorot/Xavier limits that were also using the proposed method. However, the greatest increase in accuracy are in both the first epoch cases. Suggesting that the He et al. initialisation limits have benefited again in extra earlier learning, over and above the earlier learning gains of the Glorot/Xavier limits previously.

VII. DISCUSSION AND CONCLUSION

This was follow up work from the perceptron layer non-random initialisation research towards repeatable determinism, but for convolutional layers in neural networks instead of perceptron layers. As the previous perceptron layer work would not be applicable to convolutional layers, nor either in models that use a perceptron layer with convolutional layers in the architecture. This proposed method, is applicable to convolutional layer architectures, and can achieve a higher accuracy in a single learning session, with a computational initialisation state number sequence that has been designed to be more conducive to edge detection in images.

This method has a computational known maximum number of unique filter arrangements, and those filter arrangements are deterministic and repeatable, and as such have a known *number sequence set* for safety critical applications in image classification applications. Those filter arrangements also act with the hyper-parameter control values, most notably the filter row and column values.

The distribution used in the non-random rearrangements are selected based on layer type, with sinusoid slope with the rearrangement for convolutional layer, and linier ramp with rearrangements subsequent in dense layers. The losses during learning show a quicker reduction in losses using this proposed form, and also when used with shuffles and epochs provide a higher performing accuracy result in the first learning session, that is inherited over the complete set of epochs. The repeatable deterministic property provides no variation in learning sessions, aiding the speed of development of a model. The proposed method is also robust to He et al. initialisation value limits, and when used with the proposed method offer a further accuracy gain. That proposed scheme provides a higher accuracy in a single learning session +3.35% un-shuffled, and +2.813% shuffled in the first epoch. When the He initialisation is used, instead 97.55% accuracy compared to 96.929% accuracy with the Glorot/ Xavier random form benchmark. Further work, may make more focused consideration to assessing the He et al. initialisation with a comparison baseline, and a further extension of the work could embrace different datasets, particularly those datasets with multi-channel colour images.

VIII. REFERENCES

[1] D. Hubel and T. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," in *The Journal of Physiology*, vol. 195, no. 1, pp. 215-243, Mar 1958.

[2] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *IEEE Conference Publication*. Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/5537907 [Accessed 7 Feb. 2020], 2010.

[3] J. Masci, U. Meier, D. Cireşan and J Schmidhuber, "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," in *Honkela T., Duch W., Girolami M., Kaski S. (eds) Artificial Neural Networks and Machine Learning – ICANN 2011. ICANN 2011. Lecture Notes in Computer Science*, vol. 6791. Springer, Berlin, Heidelberg, 2011.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Papers.nips.cc*. 2017. Available at: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf [Accessed 7 Feb. 2020].

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

[6] J. Knight. "Safety critical systems: challenges and directions," in *Proceedings - International Conference on Software Engineering*. pp. 547 - 550, 2002.

[7] A. Serban, "Designing safety critical software systems to manage inherent uncertainty" in *IEEE Conference Publication,* Ieeexplore.ieee.org. 2019. Available at: https://ieeexplore.ieee.org/abstract/document/8712362 [Accessed 7 Dec. 2019].

[8] P. Carpenter, "Verification of requirements for safety-critical software," Sigada.org. 2019. Available at: http://www.sigada.org/conf/sigada2001/private/SIGAda2001-CDROM/SIGAda1999-Proceedings/p23-carpenter.pdf [Accessed 7 Dec. 2019].

[9] R. Rudd-Orthner and L. Mihaylova, "Non-random weight initialisation in deep learning networks for repeatable determinism", in *IEEE Conference Publication*. Ieeexplore.ieee.org, 2019. Available at: https://ieeexplore.ieee.org/document/8770007 [Accessed 7 Feb. 2020].

[10] R. Rudd-Orthner and L. Mihaylova, "Repeatable determinism using non-random weight initialisations in smart city applications of deep learning". in *Journal of Reliable Intelligent Environments*, 2020.

[11] Y. Blumenfeld, D. Gilboa and D. Soudry, "Beyond signal propagation: is feature diversity necessary in deep neural network initialization?," in *ArXiv preprints arXiv:2007.01038*, 2020.

[12] W. Ding, Y. Sun, L. Ren, H. Ju, Z. Feng and M. Li, "Multiple lesions detection of fundus images based on convolution neural network algorithm with improved SFLA," in *Special Section on Deep Learning Algorithms for Internet of Medical Things*, 2020.

[13] Y. Wang, Y. Rong, H. Pan, K. Liu, Y. Hu, F. Wu, W. Peng, X. Xue and J. Chen, "PCA based kernel initialization for convolutional neural networks," in: *Tan Y., Shi Y., Tuba M. (eds) Data Mining and Big Data. DMBD 2020. Communications in Computer and Information Science*, vol. 1234. Springer, Singapore, 2020.

[14] M.F. Ferreira, R. Camacho and L.F. Teixeira, "Autoencoders as weight initialization of deep classification networks for cancer versus cancer studies," in *Proceedings of CIBB*, 2019.

[15] Y. LeCun, C. Cortes and C. Burges, "MNIST handwritten digit database," Yann.lecun.com. Available at: http://yann.lecun.com/exdb/mnist/. [Accessed: 26- Sep- 2020].

[16] J. Torres, "Convolutional neural networks for beginners", Medium, 2018.. Available at: https://towardsdatascience.com/convolutional-neural-networks-for-beginners-practical-guide-with-python-and-keras-dc688ea90dca. [Accessed: 26- Sep- 2020].

[17] Kassem, "MNIST: Simple CNN keras (Accuracy : 0.99)=>Top 1%", Kaggle.com, 2019. [Online]. Available: https://www.kaggle.com/elcaiseri/mnist-simple-cnn-keras-accuracy-0-99-top-1. [Accessed: 26- Sep- 2020].

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

[19] R. Rudd-Orthner and L. Mihaylova,. "Numerical discrimination of the generalisation model from learnt weights in neural networks," in *IEEE Conference Publication*. Ieeexplore.ieee.org, 2019. Available at: https://ieeexplore.ieee.org/document/8941988 [Accessed 8 Feb. 2020].

[20] R. Rudd-Orthner and L. Milhaylova, "Numerical discrimination of the generalisation model from learnt weights in neural networks," http://aetic.theiaer.org. 2020. Available at: http://aetic.theiaer.org/archive/v3/v3n4/v3n4.html [Accessed 8 Feb. 2020].