A review on feedback alignment and its underlying mechanism in backpropagation through random synaptic feedback weights

Armin Panjehpour[a] **and Negin EsmaielZadeh**[b]

[a] Electrical Engineering Department, Sharif University of Technology

June 30, 2022

The brain is responsible for every thought, feeling and action. But how do the billion neurons that reside in the brain manage all of these? On average, the human brain contains about 100 billion neurons, each of which are connected to up to 10,000 other neurons, passing signals to each other via as many as 1,000 trillion synapses. So, there is this fundamental question in neuroscience that, "what is the mechanism of plasticity in the synapses?". Concerning this, learning algorithms of artificial neural networks (ANNs) have become a key, helping us to learn how the brain works. However, one of the biggest struggles of these ANNs is being able to come up with models which are biologically plausible. Backpropagation (BP) is one of the algorithms for updating these networks, but it requires precise, symmetric backward connectivity pattern, which is impossible given the connectivity pattern of the brain (known as "Weight Transport Problem"). Previously, it has been shown that we can ignore this constraint of BP under some certain circumstances and use constant, random backward weights which can perform as good as BP. Error BackPropagation Through Random Backward Weights (BP-TRW) leads to alignment between forward and backward weights (Feedback Alignment or "FA"). Here, we briefly investigate the mathematical background underlying the FA and show that it is governed by neural activity and its statistical properties like cross-correlation and autocorrelation of the output signals of neurons and error. We show that there is alignment between the update directions of BP and FA method and also, we show that normalizing weight matrices will improve FA significantly.

Feedback Alignment | Backpropagation | Bio-Inspired Artificial Neural Networks

1 | Introduction

Backpropagation is probably, the most fundamental concept in neural networks and it has been used for many years (1). As we know, in BP, backward weights need to be precisely matched with the forward weights in order to propagate the accurate values of error signals to the earlier layers (Fig.1a,b). This structure brings up a major issue called "weight transport problem" which makes the BP to be implausible in the nervous system (2). Moreover, in the brain, axons transmit information unidirectionally which this means an axon can't be both a feed forward and a backward weight together. In this regard, it has been reported that not only learning can occurs without exact weight transport, but also with constant and random backward weights (3). In this method which is called "BP-TRW" (Fig.1c), as the learning proceeds, the angle between backward weights and transpose of forward weights decreases which shows alignment between backward weights and transpose of forward weights. In addition to propagating the error to earlier layers using backward weights between layers, error can be passed directly through random backward weights from output layer to hidden layers which is known as direct feedback alignment (DFA) (4–6). In this work, our focus is on FA and further investigation on DFA is left to future works.

In some previous works, proofs have been reported for alignment of forward and backward weights in some special cases. For example, it has been shown that if forward weight matrices are initialized with a value very close to zero, and the input and desired output of network are kept constant during iterations, backward weights matrices converge to scalar multiple of the Moore-Penrose pseudo-inverse of forward weight matrices (3). Some other special cases are reported too, but non cannot explain the occurrence of FA for arbitrary weight matrix initialization and non-linear networks.

Also, (3) has provided a mechanism of how FA works by freezing forward weights in different layers in an BP-TRW ANN. They demonstrate that the information in backward weights (B_l) accumulate in forward weight of the earlier layer (W_{l-1}) and then is passed to the next forward weight (W_l). It has been shown that continuous growth of Forbenius norm of forward weight matrices can lead to alignment (3). We will show that the main underlying mechanism of FA is not this and in contrast, limiting the Forbenius norm of the forward weight matrices leads to higher alignment.

In this work, at first, we will explore mathematical background of FA which it has not been reported in the previous works. Then, we will show that occurrence of FA is governed by neural activities' statistical properties like cross-correlation and autocorrelation. Next, We will implement a BP-TRW deep ANN and investigate the alignment as the learning process proceeds and compare BP with BP-TRW. We show that FA is robust in various conditions such as data shuffling. Afterwards, we show that there is alignment between update directions of BP and BP-TRW, although they move along completely different trajectories (Investigated using principal component analysis on learnable parameters). In addition, We show that weight normalization will increase the performance and amount of alignment in our network. In the end, we will evaluate the performance of a ANN network which update through a combination of both methods in learning process.

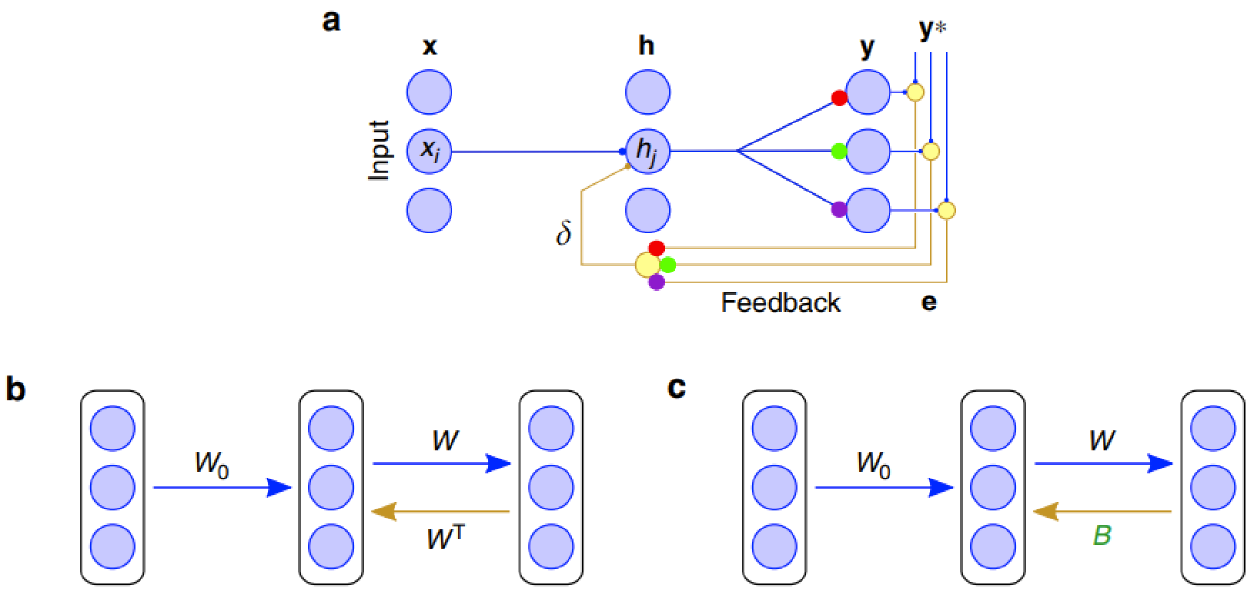

Fig. 1. **(a)** In the backpropagation learning algorithm, neurons need to know each others' synaptic weights. For example, the three colored feedback synapses must have precisely equal values to those colored in forward path. **(b)** For propagating the error to the earlier layers, error is multiplied to the transpose of forward weight matrix W, $\delta_{BP} = W^T e$. **(c)** In BP-TRW, W^T is replaced with constant random weights, B. So $\delta_{FA} = Be$. (3)

2 | Results

Consider a d layer ANN. We denote forward weight matrices by $W_l \in R^{n_l \times n_{l+1}}$, internal value of neurons by $Z_l \in R^{n_b \times n_l}$ and output value of neurons by $L_l = f(Z_l)$, where n_b is batch size, n_l is the number of neurons in layer l, and $f(.)$ is the activation function. For $0 < l \leq d$, internal value of neurons of layer l, are calculated using this linear combination, $Z_l = L_{l-1}W_{l-1} + b_l$ where b_l is the bias vector of layer l. Bias vector addition, adds a row vector to each row of the matrix. Input, output and desired output of network are denoted by $X = L_0 \in \mathbb{R}^{n_b \times n_0}$, $Y = L_d \in \mathbb{R}^{n_b \times n_d}$, and $Y^* \in \mathbb{R}^{n_b \times n_d}$, respectively.

2.1 | Underlying mechanism of feedback alignment In BP-TRW (3), the error signal is propagated to the earlier layers using constant random weights denoted by $B_l \in R^{n_{l+1} \times n_l}$. Except the backward weights, which are held constant during iterations, update directions of forward weights are calculated as below:

$$\Delta W_{l,FA}[k] = \eta L_l[k]^T \delta_{l+1,FA}[k], \quad 0 \leq l < d \qquad [1]$$

$$\delta_{l,FA}[k] = \begin{cases} \delta_{l+1,FA}[k] B_l \odot f'(Z_l[k]), & 0 < l < d. \\ -\eta \dfrac{\partial \mathcal{L}}{\partial Z_d}, & l = d. \end{cases} \qquad [2]$$

η is the learning rate, $\mathcal{L}(Y, Y^*)$ is the loss function and the element-wise multiplication is denoted by $\odot$.

To demonstrate why alignment happens in this update rule, $\Delta W_{l,FA}[k]$ should be expanded to sequential summation along iterations using $W_l[k-o]$ for $0 \leq o < k$ and $0 < l < d$. We can apply first order Taylor approximation on update steps ($\Delta W_l[k]$) by assuming that they are small to find the update directions, as below:

$$\begin{aligned} \Delta W_{l,FA}[k] &= \eta L_l[k]^T \delta_{l+1,FA}[k] \\ &= \eta f(W_{l-1}[k]^T L_{l-1}[k]^T + b_l[k]^T)\delta_{l+1,FA}[k] \\ &= \eta f(\{W_{l-1}[k-1]^T + \eta\delta_{l,FA}[k-1]^T L_l[k-1]\} \\ &\quad L_{l-1}[k]^T + b_l[k]^T)\delta_{l+1,FA}[k] \\ &\approx \eta\{f(W_{l-1}[k-1]^T L_{l-1}[k]^T + b_l[k]^T) + \\ &\quad f'(W_{l-1}[k-1]^T L_{l-1}[k]^T + b_l[k]^T) \odot \\ &\quad \eta\delta_{l,FA}[k-1]^T L_l[k-1] L_{l-1}[k]^T\}\delta_{l+1,FA}[k] \\ &\approx T^1_{l,aln}[k] + T^2_{l,aln}[k] + ... + T^k_{l,aln}[k] + \\ &\quad \eta f(W_{l-1}[0]^T L_{l-1}[k]^T + b_l[k]^T)\delta_{l+1,FA}[k] \end{aligned} \qquad [3]$$

$T^o_{l,aln}[k]$ is alignment term with order of o corresponding to layer l, which are pivots of FA. Alignment terms are defined for $1 \leq o \leq k$ and $0 < l < d$ as below:

$$\begin{aligned} T^o_{l,aln}[k] &= \eta\{f'(\zeta^o_l[k])^T \odot \\ &\quad \eta\delta_{l,FA}[k-o]^T L_l[k-o] L_{l-1}[k]^T\}\delta_{l+1,FA}[k] \\ &= \eta\{f'(\zeta^o_l[k])^T \odot \\ &\quad \eta\{f'(Z_l[k-o])^T \odot B_l^T \delta^{l+1,FAT}[k-o]\} \\ &\quad L_l[k-o] L_{l-1}[k]^T\}\delta_{l+1,FA}[k] \end{aligned} \qquad [4]$$

Where $\zeta^o_l[k]$ denotes $L_{l-1}[k] W_{l-1}[k-o] + b_l[k]$.

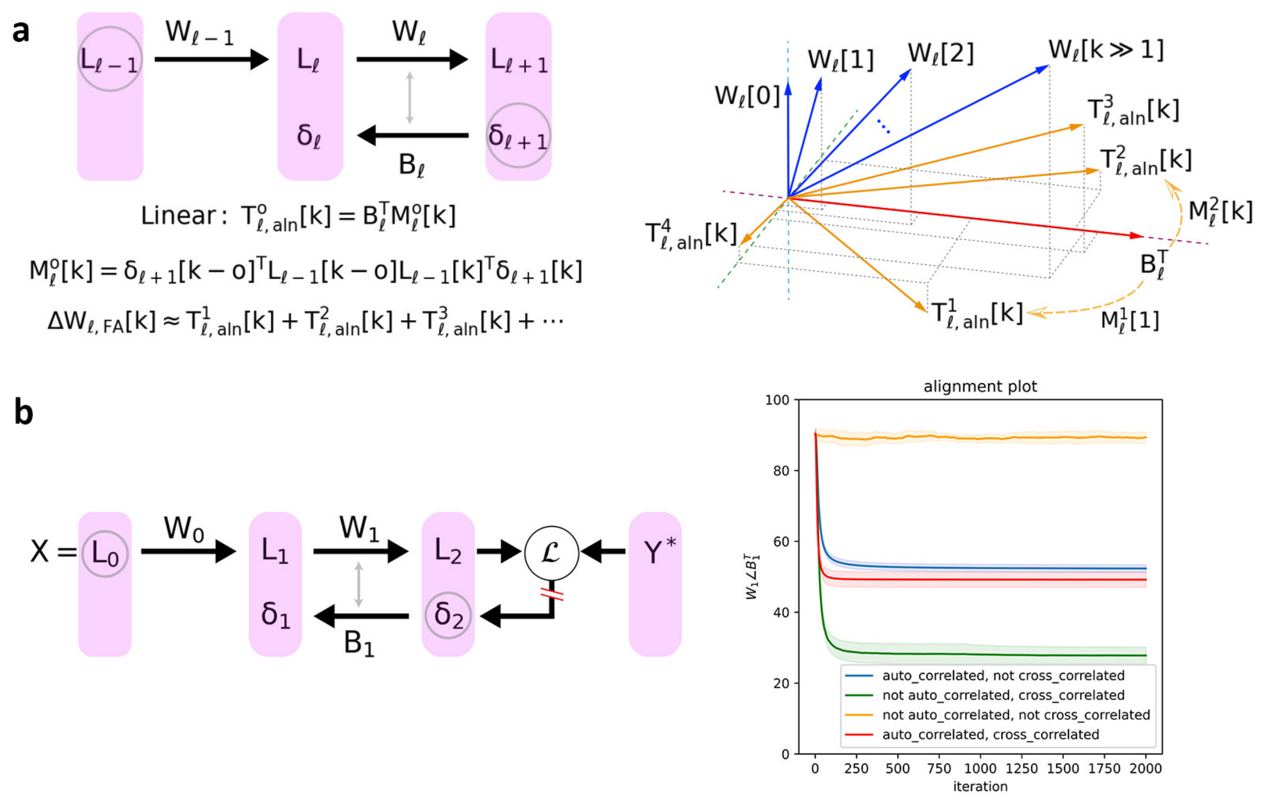
a
$W_{\ell-1}$
$L_{\ell-1}$
L_ℓ
W_ℓ
$L_{\ell+1}$
δ_ℓ
$\delta_{\ell+1}$
B_ℓ
Linear: $T^o_{\ell,\text{aln}}[k] = B^T_\ell M^o_\ell[k]$
$M^o_\ell[k] = \delta_{\ell+1}[k-o]^T L_{\ell-1}[k-o] L_{\ell-1}[k]^T \delta_{\ell+1}[k]$
$\Delta W_{\ell,\text{FA}}[k] \approx T^1_{\ell,\text{aln}}[k] + T^2_{\ell,\text{aln}}[k] + T^3_{\ell,\text{aln}}[k] + \cdots$
$W_\ell[0]$
$W_\ell[1]$
$W_\ell[2]$
$W_\ell[k \gg 1]$
$T^3_{\ell,\text{aln}}[k]$
$T^2_{\ell,\text{aln}}[k]$
$M^2_\ell[k]$
$T^4_{\ell,\text{aln}}[k]$
B^T_ℓ
$T^1_{\ell,\text{aln}}[k]$
$M^1_\ell[1]$
b
$X =$
L_0
W_0
L_1
W_1
L_2
$\mathcal{L}$
Y^*
δ_1
δ_2
B_1
alignment plot
$W_1 \angle B_1^T$
iteration
auto_correlated, not cross_correlated
not auto_correlated, cross_correlated
not auto_correlated, not cross_correlated
auto_correlated, cross_correlated

Fig. 2. (a) By expanding $\Delta W_{l,FA}[k]$, alignment terms ($T^o_{l,aln}[k]$, (Eq.4)) which robustly move forward weight matrices towards transpose of backward weight matrices under some conditions, will be calculated. $L_l[k]$ and $\delta_l[k]$ denote the output and error signals of the network and each row of them consists of data points corresponding to each mini batch at iteration k and each column corresponds to layer *l* **(b)**. Using this open-loop 2 layer ANN, hypothetical signals are created as the inputs and errors using normal distribution with a mean of 0 and a std of 1. Amount of alignment between forward and backward weights is plotted for 4 conditions as explained. The alignment occurs when we have autocorrelation or cross-correlation or both, but do not when there are none. Each trace is average over 10 runs and the shaded areas are one s.d. around the mean.

For simplicity, in this work, we consider linear ANNs, even if our network is non-linear as in the same work, it has been shown that this approximation does not affect the results if the non-linearity is a mild distortion on linear case (7):

$$T^o_l[k] = \eta^2 B^T_l \delta_{l+1}[k-o]^T L_{l-1}[k-o] L_{l-1}[k]^T \delta_{l+1}[k] \qquad [5]$$

Occurrence of FA depends on the transformation matrix $M^o_l[k] = \delta_{l+1}[k-o]^T L_{l-1}[k-o] L_{l-1}[k]^T \delta_{l+1}[k]$ which is applied to B^T_l and creates the alignment terms (Fig.2a). If $M^o_l[k]$ preserves the direction of B^T_l, alignment of $T^o_{l,aln}[k]$ with B^T_1 happens. In general, $M^o_l[k]$ can be decomposed to symmetric and skew-symmetric parts $M^o_l[k] = M^o_{l,skew}[k] + M^o_{l,sym}[k]$. A complete skew-symmetric matrix can totally deviates the direction:

$$\begin{aligned}\langle B^T,\ B^T M^o_{skew}[k]\rangle_F &= tr(B^T M^o_{skew}[k] B) \\ &= -tr(B M^o_{skew}[k] B^T) = 0\end{aligned} \qquad [6]$$

So $B \angle M^o_{skew}[k] B^T = 90°$. Therefore, the amount of alignment is determined by $B^T_l \angle B^T_1 M^o_{l,sym}[k]$ and the ratio of $||B^T_l M^o_{l,skew}[k]||_F / ||B^T_l M^o_{l,sym}[k]||_F$. In other words, the more the transformation matrix resembles a symmetric matrix, the less this ratio will give us an independent random B^T_l. In general, eigen values of symmetric transformation matrix plays an important role in alignment. According to (7), There are many possible conditions that can lead to alignment. Among all of these conditions, one is autocorrelation of output signal of neurons and errors at lag o. This happens when we have a symmetric positive semidefinite transformation matrix. Another condition which leads to alignment is when the error and output signal of neurons are cross-correlated. Also, If error and output signals are cross-correlated and autocorrelated, there would be better amount of alignment comparing to when there is just autocorrelation. Conversely, when there is neither autocorrelation, nor cross correlation between error and output signals, alignment does not happen (Fig.2b). To try out this conditions, we implemented an open-loop 2 layer ANN with ReLU non-linearity (Fig.2b). In autocorrelation condition, we initialized L_0 and δ_2 i.i.d from $\mathcal{N}(0,1)$ and left them constant during the iterations. For cross-correlation condition, we independently initialized L_0 from $\mathcal{N}(0,1)$ and let $\delta_2[k] = L_0[k]$. For the condition consisting both of the previous two conditions, L_0 and δ_2 initialized equally from $\mathcal{N}(0,1)$. For the last condition, at each iteration, both L_0 and δ_2 were initialized using $\mathcal{N}(0,1)$.

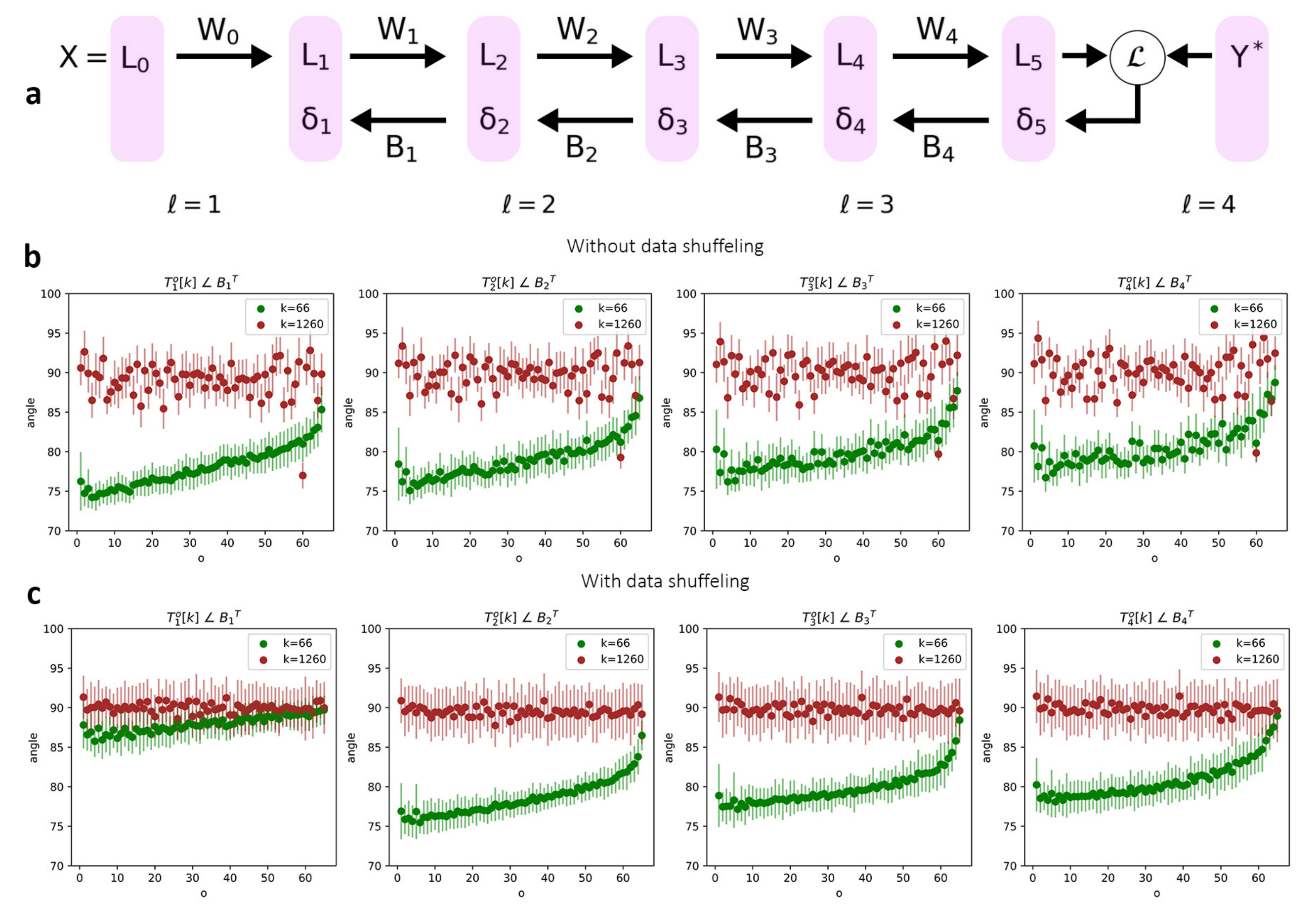

Fig. 3. (a) Layout of the network trained on MNIST. Number of neurons in all hidden layers and output layer is 50 and in input layer is 225. The activiation function is: $f(.) = tanh(ReLU(.))$. **(b,c)** The training set which consists of 60 mini batches is given to the network as input. At the initial phase of learning, when not shuffling, alignment terms are aligned with B_l^T, and those whose orders are multiple of 60 are more aligned. As the learning process proceeds, the amount of alignment of those whose orders are a multiple of 60 will increase but for the other decreases. When we shuffle the data at the beginning of each epoch, we will still have alignment, but all terms behave the same and alignment terms with an order of a multiple of 60 are just like others. This demonstrates the robustness of FA. Each dot is averaged over 30 runs and the error bars are one s.d. around the mean.

2.2 | Potential decline of alignment in the earlier layers of deep ANN

Besides the alignment of forward and backward weights, we can investigate the alignment between $\Delta W_{l,FA}$ and $\Delta W_{l,BP}$ to approximate the update direction of FA according to the update direction in BP. If statistical properties of the layers are similar to each other through the layers, we can expect $W_l \angle B_l^T$ to be consistent through the layers (which is not). However, even if this consistency is seen for $W_l \angle B_l^T$, $\Delta W_{l,FA} \angle \Delta W_{l,BP}$ tends to increase in earlier layers. So, the amount of alignment decreases in earlier layers in BP-TRW. This can be seen in a d-layer ANN theoretically as well as experimentally(for $0 < l < d$):

$$\begin{aligned} \delta_{l,FA} &= \delta_d B_{d-1} B_{d-2} ... B_{l+1} B_l \\ \delta_{l,BP} &= \delta_d W_{d-1}^T W_{d-2}^T ... W_{l+1}^T W_l^T \end{aligned} \qquad [7]$$

for $l = d$, we know that the delta is equal for both methods. As δ propagates to the earlier layers, deviation of $\delta_{l,FA}$ from $\delta_{l,BP}$ tends to increase which leads to the increase of $\Delta W_{l,FA} \angle \Delta W_{l,BP}$. This behavior has also been reported before (8) and we also observed it in the learning process of a deep ANN. (Fig.6c).

2.3 | Training a deep ANN to investigate FA in the learning process

As we have demonstrated, FA could happen under a variety of different conditions. This alignment can be influenced by the neural activity in the network, architecture of the network and properites of dataset. All of these can change the behaviour of neurons and affect the FA. Here we will implement a deep ANN and use BP-TRW to investigate FA during iterations.

2.3.1 | Network configuration Here, we trained a 5-layer non-linear fully connected ANN (Fig.3a) on the famous MNIST dataset which consists handwritten digits images. We chose the activation function of the neurons to be $tanh(ReLU(x))$. So, it is roughly similar to the frequency-current curve of biological neurons. For better performance on the classification task, the output layer is confined between 0 to 1. In order to compare the alignment between different layers, we matched the number of neurons in all hidden layers and output layer to 50 neurons. Since, we have 10 classes in the used dataset, we coded the labels using 5-hot coding (See

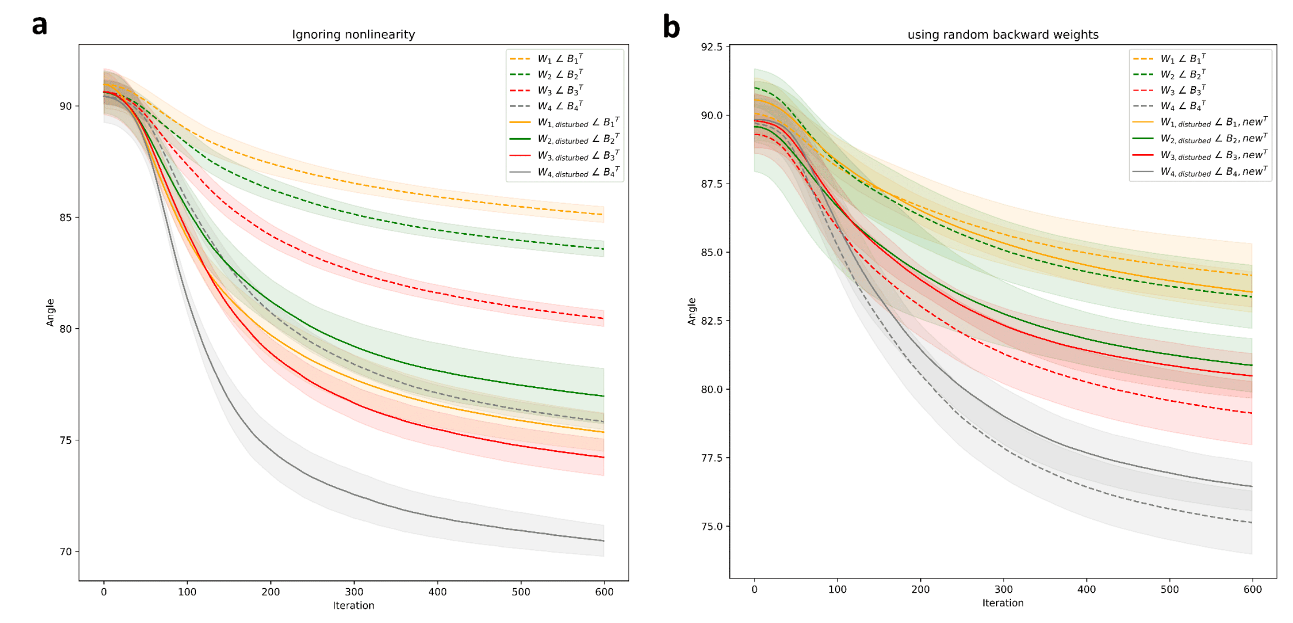

Fig. 4. **(a)**The alignment of the actual forward weights and the disturbed version of forward weights are plotted through iteration for comparison. To simulate this process, in addition to the actual forward weights, a disturbed version of forward weights is calculated during the learning process using the disturbed version of weight updating formula (Eq.8). Where the non-linearity is ignored for calculating the alignment terms. It can be seen that the non-linearity reduces the alignment through iteration, however cannot prevent the alignment occurrence. **(b)** The alignment of the actual forward weights with the actual bacward weights and the disturbed version of forward weights with the corresponding newly generated backward weights are plotted through iteration for comparison. To simulate this process, in adition to the actual forward weights, a disturbed version of forward weights is calculated during the learning process, using the disturbed version of weight updating formula (Eq.8), where new random backward weights are used in calculation of the alignment terms. It can be seen that this distortion affects the amount of alignment through iteration, however cannot prevent the alignment occurence. (a,b) are averaged over 5 runs and shaded areas are one s.d. around the mean.

Methods). To reduce the computational cost, the images are resized to 15×15. So, the number of neurons of input layer is 225. We also normalized the input images pixels values to lie between 0 to 1 by dividing all the values to 255. The batch size was chosen to be 1000, So we have a total of 60 mini batches given the training dataset of MNIST which includes 60000 images. All network parameters were initialized using N(0,0.1). The used loss function is $\mathcal{L} = \frac{1}{2}\sum_{i,j} E_{i,j}^2[k]$, Where $E_{i,j}[k]$ is the difference between the output and the desired output.

2.3.2 | Behaviour of alignment terms Using the implemented network trained by BP-TRW, without data shuffling, we can see that at the initial phase of the learning, all orders of alignments show considerable amount of alignment and the terms which their orders are a multiple of 60, slightly show more alignment compared to their adjacent orders. As the learning process goes on, alignment terms whose orders are a multiple of 60 become more aligned and those whose orders are not a multiple of 60 become less aligned (Fig.3b).
With data shuffling (shuffle mini-batches at the beginning of each epoch), at the initial phase of the learning, all orders of alignments show considerable amount of alignment and as the learning process goes on, the alignment of orders decreases. Here, there is no difference between the orders whose orders are a multiple of 60 and the other alignment terms since the input data of the network is not the same in epochs because of shuffling (Fig.3c). An interesting result that we can see, is the difference of alignment in the first layer in the data shuffled plot, which is different with the result reported by (7).

2.3.3 | Dynamic of alignment terms with respect to non-linearity and backward weights As mentioned before, we can assume non-linearity in alignment terms formulas as a distortion on the amount of alignment but not something that determines its occurrence. So for simplicity, we can ignore the non-linearities and use the linear version. To evaluate the effect of this assumption used for simplification on alignment terms, we updated a disturbed version of the alignment terms using disturbed alignment terms:

$$\Delta W_{disturbed} = \sum_{o} T_{l,aln}^{o}[k] + \eta f(W_{l-1}[0]^T L_{l-1}[k]^T + b_l[k]^T)\delta_{l+1,FA}[k] \quad [8]$$

Comparing the alignment of the main forward weights and the disturbed weights' dynamics, we can see that the non-linearity makes a distortion on weights in the way that causes alignment reduction. However, it still cannot prevent the alignment occurrence at all (Fig.4a).

In the next step, we examine the effect of the backward weights and the dependencies between the backward weights and the transformation matrix ($M_l^o[k]$) on the amount of alignment. In the learning process, the outputs and also the errors of the neurons for each layer are a function of the backward

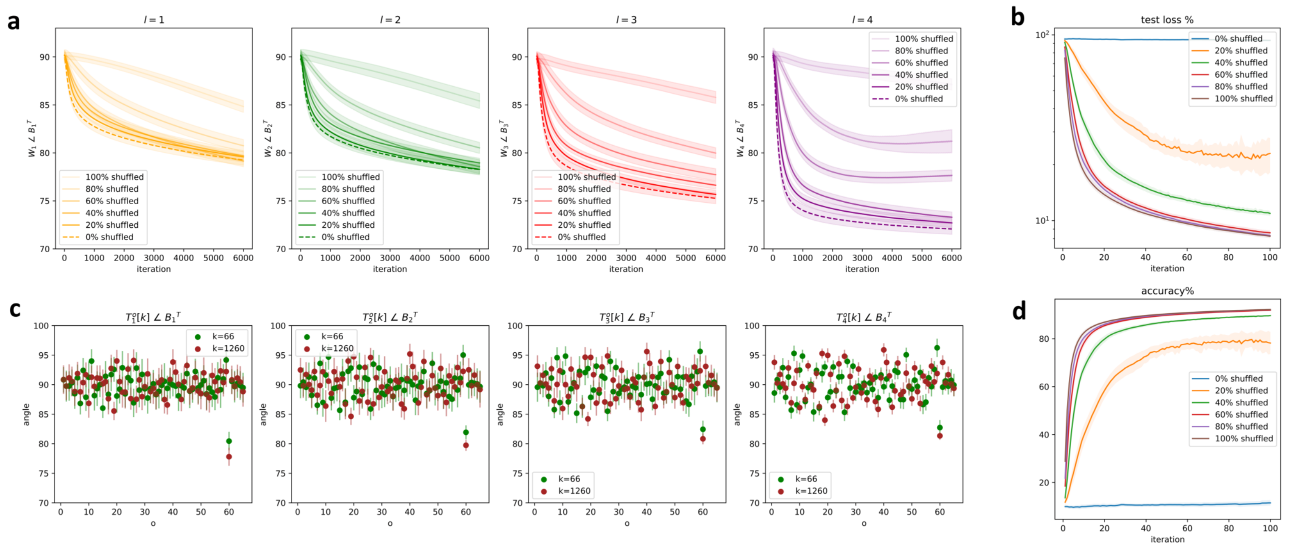

Fig. 5. (a) By shuffling the labels of dataset, the amount of alignment decreases. The more the labels shuffle, the the less the amount of alignment will be. And also, by moving towards earlier layers, the amount of alignment decreases too. **(b,d)** Loss function value in semi-log scale and test accuracy on test data. The more the shuffling, the worse the performance gets.**(c)** At the beginning of the learning, the labels are shuffled and shuffling didn't done again during learning. This disrupts the alignment of terms whose orders are not a multiple of 60. However, terms with an order which is a multiple of 60, still align because of the autocorrelation. All of the plots are averaged over 30 runs and error bars and the shaded areas are one s.d. around the mean.

weight matrices. If the backward weights do not play an effective role in the alignment occurrence, and if the dependencies between the backward weights and the transformation matrix ($M_l^o[k]$) do not either, feedback alignment could be almost assumed to be only the result of the transformation matrix properties. To study the effect of backward weights, we updated a disturbed version of the forward weights using the disturbed version of weight-updating formula (Eq.8) and used the main non-linear function for calculating the alignment terms. However, we used new backward weights ($B_{l,new}$) in calculation of alignment terms. Other parameters, such as the signal outputs and the error are derived from the main non-linear network. Comparing the alignment of the main forward weights and the disturbed weights' dynamics, we can see that the backward weights, as a distortion, affect the amount of alignment (Fig.4b). However, they still cannot prevent the alignment occurrence at all. This means that we can considerably assume the alignment occurrence as a function of the transformation matrix's ($M_l^o[k]$) properties, rather than the backward weights.

2.3.4 | Autocorrelation of the error and output signals of neurons contributes to alignment As we mentioned before (Fig.1b), autocorrelation between the error and output signals can lead to FA. Here, in this ANN training, we divide the training dataset into 60 mini batches which each of them will be repeated after 60 iterations. So, neural activities show a considerable amount of autocorrelation at lags that are a multiple of 60 which causes the alignment terms whose orders are multiple of 60, behave differently (Fig.3b). When we shuffled the data at beginning of each epoch, this alignment can't be seen (Fig.3c) and the autocorrelation does not exist. However, we can see that even with data shuffling we have considerable amount of alignment. We can deduce that the occurrence of alignment and the final behaviour of update directions is influenced by the resultant of all orders of alignments. This tells us that BP-TRW is a robust method for updating a network under many conditions such as data shuffling.

2.3.5 | Behaviour of forward weights and update directions As the learning process goes on in BP-TRW, forward weight matrices align more and more with backward weight matrices (Fig.5a). Also, update directions of BP-TRW aligns with those of BP, and as we move to the earlier layers, the amount of alignment decreases (Fig.5a). To compare the trajectories of all learnable parameters of the network (including forward weights and biases) in BP and BP-TRW, we performed principle component analysis (PCA) on the parameters of two networks each of them training with one of the methods. Initialization of the networks were done equally. We can see that the two methods, converge to completely different local minimums (Fig.6e).

2.3.6 | Weight normalization for improvement of alignment In one previous works (3), it has been reported that under a restricted condition, where input of a 2 layer linear network is white noise, and the network is trained to learn a linear function, continuous growth of the Forbenius norm of the first layer weight matrix leads to alignment. Talking generally, we can say that there is correlation between the amount of alignment and the increase of Forbenius norm of weight matrices. However, there is this one biologically limitation which is unbounded growth of Forbenius norm of weight matrices (Fig.6a). Hence, we limited and fixed the Forbenius norms of input weights of each neuron at each

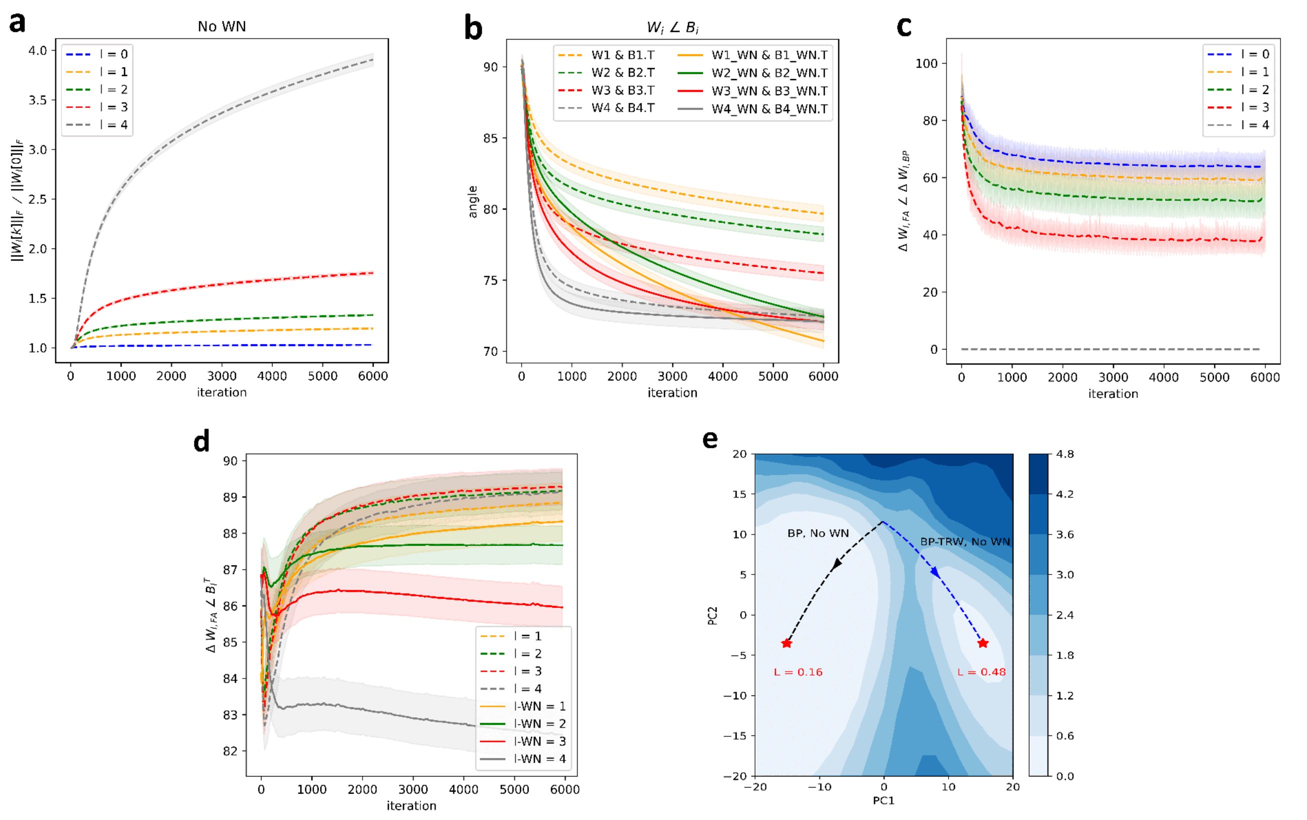

Fig. 6. (a) Continuous growth of the Forbenius norm of $W_l[k]$, which decreases as we move to earlier layers. **(b)** In BP-TRW, forward and backward weights align with each other and WN can improve the amount of alignment. **(c)** Update directions of BP-TRW and BP are aligned, and as we move to the earlier layers, the amount of alignment decreases. In layer 4, update directions of BP-TRW and BP are the same. Each trace is passed through a moving average with a window length of 60. **(d)** There is alignment between backward weights and update directions in FA, which increases using WN. **(e)** PCA is applied on all learnable parameters of both networks. The two most principle components are kept and a trajectory of the two methods is plotted. As it is demonstrating, the two methods converge to two completely different local minimums. The loss value of BP-TRW is more than BP, which shows us that BP performs better than BP-TRW without WN. Contour plot is the loss values of reconstructed points to the high dimension area. (A,B,C,D) are averaged over 30 runs and the shaded areas are one s.d. around the mean.

iteration as below:

$$W_l[k]_{*,i} = \gamma \frac{W_l[k]_{*,i}}{||W_l[k]_{*,i}||_F} \quad [9]$$

Where $W_l[k]_{*,i}$ denotes as ith column of $W_l[k]$ and γ is a positive scalar. We also applied weight normalization (WN) on backward weights at the beginning of the learning. By applying WN, the amount of alignment weights increases significantly (Fig.5b, e). Also, it has been reported that BP-TRW performs better with WN than BP (7). These results are sensitive to the hyperparameters although they are robust for a fair range of hyperparameters (γ and η).

2.3.7 | Combinatory update of the network with BP and BP-TRW When using FA algorithm under proper conditions, there is a normal expectation that the update directions, lead the network to a local or global minima and the back propagation algorithm with appropriate initial parameters also, does so (Fig.6e). Therefore, we guess that after switching the algorithm to BP in between as the BP-TRW algorithm is going on, there should be almost no drastic changes in the update direction reaching the minima. However, observations do not confirm this. In order to study this effect, we trained the previously studied network with the BP-TRW algorithm up to different steps (epoch thresholds). However, switching the algorithm after only few epochs passed from the BP-TRW updating, causes the BP algorithm to get stuck at the point which has a zero gradient and thus, prevents the net to get closer to the minima. We leave further analysis like analysing the learning trajectory of such a network for future works (Fig.7).

4 | Discussion

A. Mathematical background of FA. Artificial neural networks have differences and similarities to the natural ones. For example, BP algorithm is reported not to be biologically plausible since the forward and backward weights structure cannot be precisely matched. BP-TRW showed that not only matched backward weights are not needed but also constant random backward weights will propagate the error to the earlier layers and the network can perform as good as in BP method. In BP-TRW, something called FA happens which is alignment

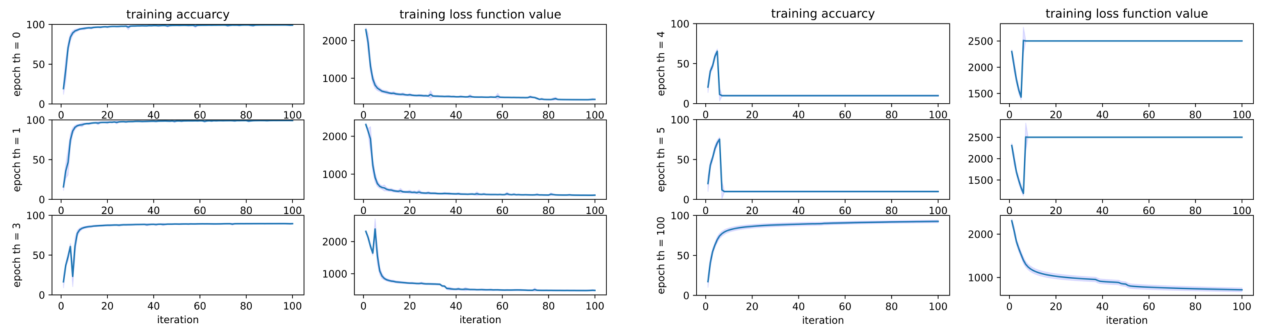

Fig. 7. Combinational method of learning for different epoch thresholds. By switching to BP after training with BP-TRW, the network won't be able to continue learning since BP can't continue the update directions of BP-TRW and it will get stuck in that local minima.

of forward and backward weights which leads to alignment of update directions of BP and BP-TRW.
In this work, we investigated the mathematical background of BP-TRW and showed that happening of FA is not depended on learning process or loss function reduction, but it is correlated with statistics of neural activities. Specifically, we show that autocorrelation and cross-correlation between the output signals of neurons and errors are important features which lead to occurrence of FA (Fig.2b).
As we mentioned, a lot of parameters are involved in the occurrence and amount of alignment such as network structure, dataset, etc. Here we trained a non-linear deep ANN on MNIST dataset. Our approaches can be applied to other configurations, too.
Some weaknesses of BP-TRW are reduction of alignment in earlier layers (Fig.6b), or its sensitivity to parameters and structures which can be overcome by some special considerations in future works.

B. Weight normalization can improve alignment. There are numerous reports that normalization can improve performance of artificial and biological neural networks (8–11). Here, we provide a method for weight normalization in which, the Forbenius norm of weight matrices are fixated by a limited amount at each iteration. We showed that this WN method can improve alignment in the network (Fig.6).

C. Further biological implausibilities of BP and BP-TRW. FA can be biologically plausible for different aspects of neural activity, which we showed here. For example, there has been reported that biological neurons have autocorrelated spike count (12–15). However, weight transport problem is only one of the biologically limitations that BP has and can be avoided by using BP-TRW. But, both BP and BP-TRW have more biologically limitations. For example, the outputs of biological neurons are non-negative, but the error which is backpropagated is signed. Despite the limitations, there has been reported that an approximation of FA may occur in the biological networks based on synaptic plasticity (16–18). So, we can imagine two different scenarios of occurrence of FA in the brain. In the first scenario, for propagating the error, there are some other neurons distinct from forward neurons (a parallel network), which act as feedback weights. In this scenario, there should be one to one cross-projection between neurons in forward and backward paths which seems to be biologically implausible. In the second scenario, the error is backpropagated using the same neurons of forward path through some other axons which act as feedback axons and are distinct from forward axons which can be plausible in the brain.
In BP-TRW, the backward weights are constant, but in the brain, synaptic plasticity surely apply on backward weights, too. So, we can say that, both forward and backward weights are propelled through each other.

D. FA may be just a piece of the brain puzzle. In summary, here we have made a clearer picture of how FA works. However, a lot of questions are remained unanswered. For example, what are the effects of including lateral connections, sparsity, segregation of excitatory and inhibitory neurons. These questions are remained to be answered in future works.

5 | Methods

5.1 | BP and BP-TRW learning methods In BP, forward weights and biases are updated with the equations below (Eq.10). Backward weights of BP are updated by setting them equal to the transpose of updated forward weight matrix at each iteration.

$$\begin{aligned} W_l[k+1] &= W_l[k] + \Delta W_l[k] \\ b_l[k+1] &= b_l[k] + \Delta b_l[k] \\ B_{l,BP}[k+1] &= W_{l,BP}[k+1]^T \end{aligned} \qquad [10]$$

Where the gradient directions computed with BP and BP-TRW, are:

$$\Delta W_l[k] = -\eta \frac{\partial \mathcal{L}}{\partial W_l}|_k = \eta L_l[k]^T \delta_{l+1}[k], 0 \leq l < d \qquad [11]$$

$$\Delta b_l[k] = -\eta \frac{\partial \mathcal{L}}{\partial b_l}|_k = \eta J_{1\times n_b} \delta_l[k], 0 < l \leq d \qquad [12]$$

Where $J_{1\times n_b}$ is a $1 \times n_b$ all one matrix. Error matrices are calculated as below for BP and BP-TRW:

$$\delta_{d,BP}[k] = E[k] \odot f'(Z_d[k]) \qquad [13]$$

$$\delta_{l,BP}[k] =_{l+1,BP} [k] W_l[k]^T \odot f'(Z_d[k]), 0 < l < d \qquad [14]$$

$$\delta_{d,FA}[k] = E[k] \odot f'(Z_d[k]) \quad [15]$$

$$\delta_{l,FA}[k] =_{l+1,FA} [k]B_l[k] \odot f'(Z_d[k]), 0 < l < d \quad [16]$$

5.2 | Angle and cosine similarity between two angles The angle between two matrices B and W with the same dimension is calculated as below:

$$W\angle B = cos^{-1}(\frac{\langle W,\ B[k]\rangle_F}{||W||_F||B||_F} \quad [17]$$

Where $\langle W,\ B[k]\rangle_F$ is the Forbenius inner product of W and B, and $||.||_F$ is the Forbenius norm. Cosine similarity is also calculated by the equation below:

$$cosine_{similarity}(W,B) = \frac{\langle W,\ B[k]\rangle_F}{||W||_F||B||_F} \quad [18]$$

5.3 | N-hot coding the labels In this work, we used 5-hot coding since the number of neurons in the output layer was 50. To generate N-hot coded labels for a dataset which consists of C categories and m output neurons ($n \cdot C \leq m$), we start from the first category to the last one and successively for each category, we initialize its label vector with all zero elements and then randomly select n elements out of $m - c \cdot n$ that were not equal to 1 for any of the previous categories and set them equal to 1.

6 | Data and Code Availability Statement

The MNIST dataset can be found HERE, but we download it from Tenserflow.Keras library.
The code for simulating all of the analysis is available at HERE, under Apache License 2.0. Pytorch library is used in order to run the codes on GPU for faster computations (Pytorch libraries are not used).

References

1. Geoffrey E. Hinton Ronald J. Williams David E. Rumelhart. Learning representations by back-propagating errors. nature 323, 533–536 (1986). https://doi.org/10.1038/323533a0. *Nature*, 1986.
2. D. G. Stork. Is backpropagation biologically plausible. volume 2, pages 241–246. ieee washington, dc. *International Joint Conference on Neural Networks*, 1989.
3. Daniel Cownden3 Douglas B. Tweed4 5 Colin J. Akerman1 Timothy P. Lillicrap1, 2. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communication*, 2016.
4. A. Nøkland. Direct feedback alignment provides learning in deep neural networks. *arXiv preprint arXiv:1609.01596*, 2016.
5. d'Ascoli S. Ohana R. Refinetti, M. and S. Goldt. The dynamics of learning with feedback alignment. *arXivpreprint arXiv:2011.12428*, 2020.
6. Poli I. Launay, J. and F. Krzakala. Principled training of neural networks with direct feedback alignment. *arXiv preprint arXiv:1906.04554.*, 2019.
7. Farokh Marvasti Alireza RahmanSetayesh, Ali Ghazizadeh. Feedback alignment with weight normalization can provide a biologically plausible mechanism for learning. *bioRxiv 2021.06.12.447639; doi: https://doi.org/10.1101/2021.06.12.447639*, 2021.
8. Litwin-Kumar A. Moskovitz, T. H. and L. Abbott. Feedback alignment in deep convolutional networks. *arXiv preprint arXiv:1812.06488.*, 2018.
9. Wang J. Shen, Y. and S. Navlakha. A correspondence between normalization strategies in artificial and biological neural networks. *bioRxiv.*, 2020.
10. Leibo J. Liao, Q. and T. Poggio. How important is weight symmetry in backpropagation? *In Proceedings of the AAAI Conference on Artificial Intelligence, volume 30*, 2016.
11. Bannon N. M. Chen J.-Y. Bazhenov M. Chistiakova, M. and M. Volgushev. Homeostatic role of heterosynaptic plasticity: models and experiments. *Frontiers in computational neuroscience, 9:89.*, 2015.
12. Fascianelli V. Ferrucci L. Cirillo, R. and A. Genovesio. Neural intrinsic timescales in the macaque dorsal premotor cortex predict the strength of spatial response coding. *Iscience, 10:203–210*, 2018.
13. Bernacchia A. Freedman D. J. Romo R. Wallis J. D. Cai X. Padoa-Schioppa C. Pasternak T. Seo H. Lee D. et al. Murray, J. D. A hierarchy of intrinsic timescales across primate cortex. *Nature neuroscience, 17(12):1661–1663.*, 2014.
14. T. Ogawa and H. Komatsu. Differential temporal storage capacity in the baseline activity of neurons in macaque frontal eye field and area v4. *Journal of neurophysiology, 103(5):2433–2445.*, 2010.
15. Tsujimoto S. Marcos E. Fascianelli, V. and A. Genovesio. Autocorrelation structure in the macaque dorsolateral, but not orbital or polar, prefrontal cortex predicts response-coding strength in a visually cued strategy task. *Cerebral cortex, 29(1):230–241.*, 2019.
16. J. C. Whittington and R. Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation, 29(5):1229–1262*, 2017.
17. J. C. Whittington and R. Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences, 23(3):235–250.*, 2019.
18. Santoro A. Marris L. Akerman C. J. Lillicrap, T. P. and G. Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience, pages 1–12.*, 2020.