Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2020.3026423

Improved Highway Network Block for Training Very Deep Neural Networks

OYEBADE K. OYEDOTUN[1], (Student, IEEE), ABD EL RAHMAN SHABAYEK[1], (Member, IEEE), DJAMILA AOUADA[1], (Senior, IEEE), and BJÖRN OTTERSTEN[1], (Fellow, IEEE)
[1]Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855 Luxembourg

Corresponding author: Oyebade K. Oyedotun (e-mail: oyebade.oyedotun@uni.lu).

This work was funded by the National Research Fund (FNR), Luxembourg, under the project references R-AGR-0424-05-D/Björn Ottersten and CPPP17/IS/11643091/IDform/Aouada.

ABSTRACT Very deep networks are successful in various tasks with reported results surpassing human performance. However, training such very deep networks is not trivial. Typically, the problems of learning the identity function and feature reuse can work together to plague optimization of very deep networks. In this paper, we propose a highway network with gate constraints that addresses the aforementioned problems, and thus alleviates the difficulty of training. Namely, we propose two variants of highway network, *HWGC* and *HWCC*, employing feature summation and concatenation respectively. The proposed highway networks, besides being more computationally efficient, are shown to have more interesting learning characteristics such as natural learning of hierarchical and robust representations due to a more effective usage of model depth, fewer gates for successful learning, better generalization capacity and faster convergence than the original highway network. Experimental results show that our models outperform the original highway network and many state-of-the-art models. Importantly, we observe that our second model with feature concatenation and compression consistently outperforms our model with feature summation of similar depth, the original highway network, many state-of-the-art models and even ResNets on four benchmarking datasets which are CIFAR-10, CIFAR-100, Fashion-MNIST, SVHN and imagenet-2012 (ILSVRC) datasets. Furthermore, the second proposed model is more computationally efficient than the state-of-the-art in view of training, inference time and GPU memory resource, which strongly supports real-time applications. Using a similar number of model parameters for the CIFAR-10, CIFAR-100, Fashion-MNIST and SVHN datasets, the significantly shallower proposed model can surpass the performance of ResNet-110 and ResNet-164 that are roughly 6 and 8 times deeper, respectively. Similarly, for the imagenet dataset, the proposed models surpass the performance of ResNet-101 and ResNet-152 that are roughly three times deeper.

INDEX TERMS Very deep networks, neural network, highway blocks, classification and optimization

I. INTRODUCTION

Deep neural networks have found applications in many real-life tasks; their successes for learning different difficult problems are well documented. In particular, the field of computer vision has hugely benefited from deep neural networks for various applications ranging from pose estimation [1], segmentation [2], action recognition [3], face recognition [4], etc. In recent times, there is a growing trend of using deep networks for learning directly from raw data (i.e. without feature decorrelation, feature selection, image restoration and denoising, etc.) or with very minimal data processing. In essence, deep networks are burdened to cope with various data irregularities, rather than carefully hand-crafting the features that are thought to be important for the task at hand. A motivation for this line of application is that lesser human involvement is required; this translates to shorter time for building models and a reduction in data processing costs. Moreover, the difficulty of the problems that we intend to solve over time has consistently increased. For example, the MNIST dataset[1] that was considered very difficult many years ago, and thus used for benchmarking, is no longer considered as a hard dataset; many works [5], [6] have reported error rates in the range [0.5%, 0.21%]. However, tackling harder learning tasks has motivated a new direction for deep neural networks with many layers

[1]http://yann.lecun.com/exdb/mnist/

of feature representations. Such models are referred to as very deep networks[2] [7]. Interestingly, empirical [7], [8] and theoretical [9]–[11] evidences show that increasing the depth of deep models increases the complexity of the functions that they can capture. The hope is that with the availability of massive amount of data, we can learn a robust classifier that can generalize well on unseen data. Consequently, deep networks with over 15 layers have been reported to produce state-of-the-art results in many works [8], [12]; models with hundreds of layers can be found in [8], [13].

Although very deep networks give very promising results, they are hard to train [7], [8]. In fact, using very simple and small datasets, [14] showed that different initialization schemes [15], [16] and batch normalization [17] do not alleviate the problem of optimizing very deep networks beyond a certain depth. A major problem associated with training very deep networks is that learning the identity function is hard; this problem compounds as depth increases [8]. Consequently, many works [7], [8], [12] that successfully trained very deep networks relied on some means of routing untransformed features from the earlier layers through the model; that is, learning near identity functions for the different layers. One of the earliest models with over 20 layers that was successfully trained is referred to as a *highway network* [7]. The highway network is essentially a stack of highway blocks that is trained end-to-end using the gradient descent optimization method. Particularly, the highway block relies on a gating mechanism for routing untransformed lower layer features through the model to alleviate the problem of model optimization. In another work [8], Residual Network (ResNet) with skip connections of identity mappings was proposed for training very deep networks; the ResNet is essentially a stack of residual blocks with *skip connections*, also known as *shortcut connections*. Skip connections allow the direct routing of lower layer transformations to higher layers, and therefore ResNet can learn near identity functions while alleviating the difficulty of model optimization.

Although very impressive results were reported for the original highway network [7], there are learning shortcomings that are discussed as follows. Indeed, the highway blocks used for constructing the model may make it difficult to learn new feature transformations (or latent representations) which are important for generalization as training progresses. This is a result of how highway blocks are learned. First, the gating units use the sigmoid activation function that limits output range to 0-1. At the start of training, the biases of the gating units of all highway blocks are initialized to negative values. As a result, this situates the gating units close to saturation. While this setup alleviates the difficultly of model optimization by mostly routing untransformed features via the highway block, the gating units further go into saturation as training progresses. This learning setup favours learning essentially untransformed features, and therefore may impact model generalization that typically relies on learning new feature representations in the different hidden layers. Thus, very deep networks learned using highway blocks of the described form may suffer a reduction in model effective depth due to lack of new latent representations for training data, since emphasis is put on learning untransformed features. Alternatively, the same problem can be considered as learning too close to the identity function regime at the expense of learning new transformations; as such, this can hurt generalization performance since the model becomes *inherently* much shallower than the topology implies. Furthermore, employing a gate for every layer translates to roughly doubling the number of parameters for the model. Subsequently, model overfitting can become a concern.

In this paper, the problem of training very deep networks using gating mechanisms as in the highway network [7] is revisited. We address the aforementioned problems of the highway network [7] by employing gate constraints and reformulating the fashion in which transformed and untransformed features are gated. As such, two highway networks, highway network with gate constraints (HWGC) and highway network with concatenated and compressed features (HWCC) are proposed. HWGC uses features summation, while HWCC relies on the concatenation of gated features and compression to achieve better results. Particularly, we note that HWCC is computationally more efficient than HWGC, and allows us to outperform or achieve competitive results with ResNet [8], [18] using similar number of model parameters, but with significantly lesser depth. The model *HWGC* was first presented in [19], but will be elaborated on with additional experimental results in this study.

The two highway network models, HWGC and HWCC, presented in this paper allow a natural extraction of hierarchical feature representations as training progresses. At the start of training, we learn very close to the identity function regime to aid model optimization. However, as training progresses, the model can learn further away from the identity function regime such that new transformations can be extracted in favour of generalization performance. Importantly, the aforementioned merits are achieved without sacrificing the ease of model optimization. In fact, the models can be trained using less epochs than for the original highway network [7]. Specifically, our contributions based on the proposed highway blocks are as follows:

1) More effective use of model depth, since there is a natural learning of new latent representations for data as training progresses; this is critical for the generalization of deep networks.
2) Constructing a very deep highway network that relies on a far smaller number of gates for training than in the original highway network [7]; this can alleviate the problem of model overfitting and improve computational efficiency. Specifically, one gate and three gates can be used for training models having up to 19 layers and 32 layers, respectively.
3) Using the similar (or a smaller) number of model param-

[2]Deep models with 15 or more layers

eters, the proposed models outperform the original highway network [7] and many state-of-the-art models [8], [20]–[23] on five benchmarking datasets. Namely, on CIFAR-10, CIFAR-100, Fashion-MNIST and SVHN datasets, our 19-layers HWGC and HWCC outperform (or at least match) ResNet with 110 [8] and 164 layers [18] that are roughly 6 and 8 times deeper, respectively. On the imagenet dataset, our 34-layers HWGC and HWCC outperform (or at least match) ResNet with 101 layers [8] that is roughly 3 times deeper; and again our 50-layers HWGC and HWCC outperform (or at least match) ResNet with 152 layers [8]. Importantly, HWCC performs better than HWGC on the datasets.

4) For obtaining similar level of model generalization capacity, the proposed highway network models (HWGC, and especially HWCC) in comparison to the original highway network [7], ResNets [8], [18] and DenseNet [12] result in significant reduction of inference time and Graphics Processing Unit (GPU) memory requirement for supporting real-time applications, since they require less depth.

For validating the proposed highway network blocks, CIFAR-10, CIFAR-100, Fashion-MNIST, SVHN and imagenet datasets are used. The remaining sections of this paper are organized as follows. In Section II, works discussing the theoretical motivation for very deep networks, the challenges of training them, the impact of depth on model generalization, along with the different approaches for training are discussed as related works. Section III contains the background on highway networks, including the problem statement. In Section IV, we give the formulation and construction of HWGC and HWCC for alleviating the difficulty of training. Section V presents the settings for model training and the experimental results. The paper is concluded in Section VI.

II. RELATED WORK

A. THEORETICAL MOTIVATION FOR DEEPER MODELS

Along side empirical evidence [7], [8], there are theoretical works [9]–[11] that study the importance of model depth for learning highly varying functions. In [9], sum-product networks are used for analyzing the family of functions that can be learned using models of different depth. The conclusion in the work is that there are families of functions that can be efficiently learned with a deeper model, but using a much shallower model would require an exponential increase in the number of model parameters for learning the same family of functions. Another work [10] arrived at a similar conclusion as in [9], that deeper models are more efficient in approximating very complex target functions; especially those that are compositional. In addition, the same work, Mhaskar et al. [10] noted that deeper networks are capable of working at lower Vapnik-Chervonenkis (VC) dimensions compared to shallower models. The power of deeper models to tackle more difficult problems was also studied in [11]. This work derived upper and lower bounds on model complexity considering the number of hidden units and different activation functions. The observation was that using a similar number of computing units, depth increase resulted in an improved capacity of deep networks for function approximation.

B. CHALLENGES OF TRAINING DEEPER MODELS

Training very deep networks is not trivial. The first challenge is in the fundamental problem of optimization; that is, fitting the training set to the model is difficult. High error rates are typically obtained on the training set. Subsequently, this nullifies any chance of good model generalization to unseen examples, since the training set cannot be learned properly in the first place.

Recently, there has been active research in trying to identify the problems that plague the training of very deep models. In [8], it is observed that learning the identity function is hard. In the work, 20-layer and 56-layer models were constructed for exemplifying the problem of learning the identity function. It was expected that the 56-layer model would be capable of reaching the same training error as the 20-layer model, supposing that the extra 36 layers in the 56-layer model would simply learn the identity function; in this operation regime, the 56-layer model should behave similarly to the 20-layer model, at least on the training set. Instead, it was observed that the 56-layer model converged to a much higher training error than the 20-layer model; this empirically suggests that learning the identity function is hard for deeper networks. In another work [24], the *problem of feature reuse* was considered for the difficulty of training very deep networks. The work described feature reuse as a scenario where there is the dilution of feature transformations (or latent representations) learned from one layer to the following in the apparently long chain of layers in a very deep model. As a result, at the start of training, the features that reach the output layer are considerably unreflective of the input data, especially since model weights are typically initialized randomly. Subsequently, the error gradients computed in the output layer of the model are quite ineffective for driving the model parameters towards convergence. Hence, the model may fail or take 'forever' to converge.

Another well studied problem for the training of very deep networks is *exploding and vanishing gradients* [25]–[27]. Since for training, computed error gradients are back-propagated through several layers with multiplicative effect, there is a considerable chance that error gradients diminish or explode. In [28], it was showed that shattering gradients in which gradients resemble white noise (i.e. are uncorrelated) plague training of very deep networks. Moreover, very deep networks seem to have badly conditioned Jacobians and Hessians of loss functions that hinder optimization [29]–[31].

C. MODEL PERFORMANCE WITH DEPTH

Different tasks have been successfully learned with interesting results using deep networks of few layers (i.e. 3-7 layers) [32], [33]. Nevertheless, addressing more challenging tasks necessitates that the learning characteristics of current models is reassessed with a view to improving their capacity

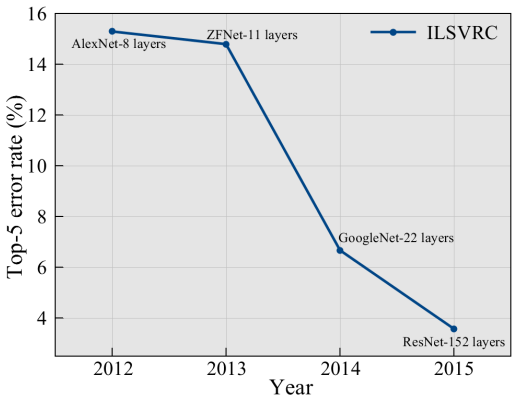

FIGURE 1: Top-5 error rate with depth on the ILSVRC [19]

for generalization. For instance, the best top-5 error rate reported on the ILSVRC2012 dataset in 2012 was 15.3%, and it was obtained using AlexNet [34] which has about 60 million parameters. Considering the current trends in deep networks, AlexNet with 5 convolution layers, 3 max pooling layers and 3 fully connected layers can be considered as a modest model. The ILSVRC2013 object classification challenge was won using a modified AlexNet model [35]. The model was named ZFNet [35] and was constructed by increasing the number of convolution layers to 8; it achieved a top-5 error rate of 14.8%. The ILSVRC2014 object classification challenge winner employed GoogleLeNet [36] which features a signficantly increased model depth in comparison to ZFNet. GoogleLeNet has 22 layers and uses smaller convolution filters of different sizes which produce aggregated features for what was referred to as an inception module. A drastic improvement in generalization performance was observed; a top-5 error rate of 6.67% was achieved. Interestingly, the runner up to ILSVRC2014 object classification challenge relied on a 16-layer model named 'VGG16' [37] that achieved a top-5 error rate of 7.3%. An 'unconventional' and exceptionally deep model with 152 layers achieved the smallest top-5 error rate in ILSVRC2015 object classification challenge; the model was referred to as ResNet [8] achieved an impressive top-5 error rate of 3.57%. Figure 1 summarizes the impact of model depth on reported results on the ILSVRC dataset.

From the evolution of best results reported on ILSVRC dataset, the impact of depth for improving model generalization on hard datasets becomes apparent. It is noted that similar observations have been reported on other hard benchmarking datasets such as CIFAR10, CIFAR-100 and SVHN. For example, most of the models [8], [12], [22] that have approached an error rate of 5% and 23% on CIFAR-10 and CIFAR-100 datasets respectively employed many layers of feature representations.

D. APPROACHES FOR TRAINING VERY DEEP NETWORKS

Motivated by empirical results on different hard tasks and insights from theoretical works, there are works that have successfully trained very deep networks with good results. Interestingly, one feature that all the works have in common is the use of untransformed features (i.e. skip connections) in the different layers of the proposed models. This has also been shown to alleviate the problem of vanishing and exploding gradient, and badly conditioned loss function Hessians [25], [28], [31].

In [8], the ResNet with skip connections was proposed. The different layers of the ResNet are grouped into residual blocks; typically, each block is composed of 2 or 3 layers. The output of a block is usually the addition of the learned transformation and the output of the preceding residual block via a skip connection. It is straightforward to see that the introduction of skip connections addresses the problem of learning the identity function and feature reuse. Consequently, this alleviates the difficulty of model optimization. In the same work [8], models having up to 1202 layers were successfully trained. In a following work [24], very deep networks with stochastic depth were proposed. Essentially, the learning scheme is such that a random subset of the model layers are removed during training for every iteration. The remaining layers are bridged using the identity connections. The training scheme was reported to allow a successful training and even improve model regularization as compared to ResNet. Again, it can be seen that this training setup address the problem of learning identity functions and feature reuse. In a more recent work [12], the proposed model is such that each layer is connected to every other layer; the model was referred to as DenseNet. Many state-of-the-art results on different benchmarking datasets were reported using the DenseNet.

In a different model referred to as *highway network* [7], gating mechanisms were used for addressing the problem of learning identity function and feature reuse. It is one of the pioneering works that first successfully trained deep models having up to 50 layers proposed highway network [7]. Highway network was reported to have been motivated by the construction of the Long Short Term Memory (LSTM) recurrent network [38] for addressing the problem of learning long term dependencies. The highway network relies on gating mechanisms for routing untransformed features from lower layers to higher layers. Specifically, rather than feeding features from the previous layer (or block) all together to the succeeding one as in ResNet, gating mechanisms are used for learning what part of the features in the previous layer (or block) to be routed to the succeeding layer (or block). It was posited that learning what part of the previous feature transformation to carry over to the next layer may be more optimal than 'blindly' carrying over all the features for every layer (or block); gating mechanisms may help reduce features redundancy.

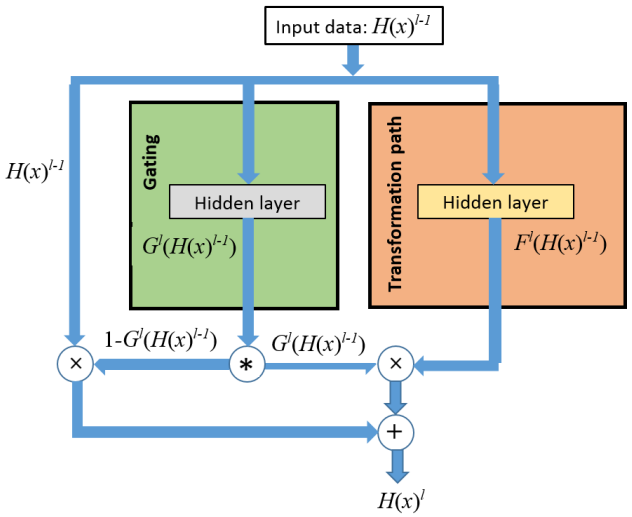

FIGURE 2: Highway network block [7]

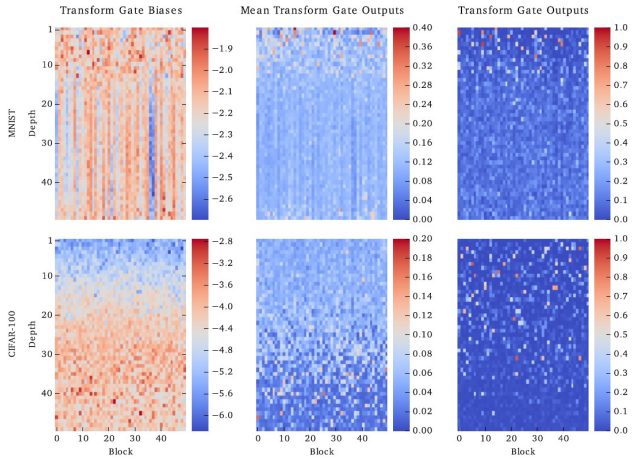

FIGURE 3: Original highway block learning attributes. Top row: 50-layer model trained on MNIST dataset. Bottom row: 50-layer model trained on CIFAR-100 dataset. First column: transform gate biases. Second column: mean of the transform gate outputs over the whole training set. Third column: transform gate output using a random training sample [7]

III. BACKGROUND AND PROBLEM STATEMENT

The background on the model that we build on, highway network [10], is presented in this section. In addition, a high level discussion of the problem statement is given.

A. BACKGROUND: HIGHWAY NETWORK

The highway network is usually a stack of highway blocks that are trained end-to-end. Each highway block employs a gating mechanism for controlling information flow through the model. Given that $H(x)^{l-1}$ is the information on the highway at layer $l-1$, the gating module outputs a signal $G^l(H(x)^{l-1})$ for controlling what information is routed to the succeeding highway network block. The final output of the highway network block at layer l, $H(x)^l$, can be written as

$$H(x)^l = F^l(H(x)^{l-1})G^l(H(x)^{l-1}) + H(x)^{l-1}(1 - G^l(H(x)^{l-1})), \quad (1)$$

where $F^l(H(x)^{l-1})$ is the transformation learned at hidden layer l for input $H(x)^{l-1}$. The form of feature learning in (1) was shown to allow the successful training of very deep networks [7]. Based on the given formulation, the gate G^l at layer l can either permit or hinder the flow of incoming signals $H(x)^{l-1}$ and $F^l(H(x)^{l-1})$, depending on the current state of the gating units; the gates can be either *open* or *closed*. The highway block is shown in Figure 2. For alleviating the difficultly of model optimization due to signal 'degradation' with depth increase (as earlier discussed in Section I), it follows that the gating units are initialized in manner that most of $H(x)^{l-1}$ (untransformed incoming features) are routed via the block early at the start of training to favour early model optimization. The training scheme for achieving this is described as follows:

i The units in the transformation path use the rectified linear function or similar activation function. However, the gating units use the Log-Sigmoid function as the activation function so that the range of units' outputs is from 0 to 1; where, states 0 and 1 denote closed and open gates, respectively.

ii The biases of the gating units, g^l, in G^l, at layer l are initialized to negative values such as -1 or -3 [10] to ensure that most of the features learned at layer l, $H(x)^l$, are untransformed features, $H(x)^{l-1}$. This is due to the gating units' activations, $G^l(H(x)^{l-1})$, being close to 0; see (1).

B. PROBLEM STATEMENT

To address the problem of training very deep networks and fast convergence, [8] proposed the ResNet for which the transformation of the form given below is learned

$$H(x)^l = F^l(H(x)^{l-1}) + H(x)^{l-1}, \quad (2)$$

where notations are the same as in (1).

We observe that [8] posited that ResNet blocks *always* allow the routing of untransformed features from one ResNet block to the succeeding one, and therefore good convergence is achieved. However, the output of a highway network block is the combination of some transformed and untransformed features. The main operation of the highway network block is to learn using a gating mechanism that controls what part of the transformed features and untransformed features to route to the next layer (or block). Generally, the highway block routes mostly untransformed features at the start of training to alleviate the difficulty of training. As the training progresses, the hope is that the highway block starts to route more transformed features. The smooth transitioning from routing untransformed features to routing transformed features is a nice attribute of the highway network. It is noted

in [7] that the gating units of the highway blocks respond selectively to different incoming features. This means that the gates perform some sort of feature filtering. In contrast, ResNet does not possess this interesting learning characteristic. Nevertheless, we discuss in the remaining part of this section some concerns on the learning scheme of the original highway network block that may negatively impact model generalization.

Mainly, we argue that the form of transformation learning for the highway block proposed in [7] and given in (1) does not facilitate the natural extraction of new latent representations. At the start of training, the biases of the gating units for the highway block are initialized to negative values, and therefore positions the gating units close to the lower end of the saturation regime of the sigmoid activation function; that is, close to zero. While this training setup suffices for routing untransformed features from the lower layers to higher layers, this may not be the optimal way to use the gating units since it is well known that Log-Sigmoid units saturate to very small values during training for deep networks [39]. This was also noted in [7]. A direct result of this is that learning new latent representations (or transformations) is hard, since the gating units go further into the lower end of the saturation regime (i.e. become smaller than the initialized values) and therefore places emphasis on routing untransformed features at the expense of routing transformed features which are critical for model generalization; see (1). Specifically, $G^l(H(x)^{l-1})$ converges to zero as training progresses and therefore we can rewrite (1) as

$$H(x)^l \approx H(x)^{l-1} \quad \text{for} \quad i \gg 1 : 1 \leq i \leq t, \tag{3}$$

where i is the epoch index and t is the number of maximum epochs. It can be seen that the result given in (3) does not readily favour model generalization, since the highway network block routes mostly untransformed features to the succeeding block, and new features are not being effectively learned. The fact that gating units go into the lower end of the saturation spectrum (i.e. very close to zero) during training was also acknowledged in the original highway network block [7]. At the start of training, the biases of the gating units for 50-layers models trained on MNIST and CIFAR-100 datasets were initialized to -2 and -4, respectively. However, it was noted that the biases of the gating units became smaller after training. Correspondingly, the mean transform gating units' outputs over the whole MNIST and CIFAR-100 datasets are close to zero; the same is observed for a single random sample as shown in Figure 3.

In addition, another concern is that for tackling the problem of optimization, the original highway network [7] uses gates for every layer of features transformation. This leads to a significant increase in the number of model parameters. A gate can double the overall number of parameters of any layer where it is used. Consequently, there is increased concern of model over-fitting the training data. Moreover, relying on gates for every layer could result in the transformation path becoming somewhat redundant (i.e. not constrained to capture important features), since the gate parameters are enormously large and sufficient to fit the training data [19]. Again, it is observed in our experiments that this may impact model generalization performance. In the following section, we focus on the formulation and learning characteristics of the proposed highway network block for training very deep networks.

IV. PROPOSED APPROACH

The details of the proposed highway network block that addresses the aforementioned problems (i.e. discussed in Section III.B) for the original highway network block are presented in this section. Namely, the two proposed models, HWGC and HWCC, are discussed.

A. HIGHWAY NETWORK WITH GATE CONSTRAINTS

The proposed highway network in this paper, HWGC, is reformulated to achieve the following important learning characteristics:

(i) Prioritize model optimization at the start of training. That is, the gates mostly route untransformed features at the start of training.
(ii) Prioritize learning new feature transformations further into training. That is, the gates route mostly transformed features.
(iii) Require significantly fewer gates for learning several layers of feature transformations, since gate transitioning is more effective for model optimization and generalization.

HWGC block is so constructed to realize the aforementioned learning characteristics by learning a transformation of the form

$$\begin{aligned} H(x)^l = F^l(H(x)^{l-1})(1 - (G^l(H(x)^{l-1}))^m) + \\ H(x)^{l-1}(G^l(H(x)^{l-1}))^n, \end{aligned} \tag{4}$$

where m and n are model hyper-parameters which are employed for remapping the gating function and other denotations remain as earlier stated. HWGC block is shown in Figure 4. The nice characteristic transitioning of learning regime mentioned in (i) & (ii) permits the efficient use of model parameters and necessitates fewer gates for successful training as mentioned in (iii). Thus, there is a lower risk of model over-fitting the training data. In the remaining parts of this section, we discuss the three components of the proposed highway network block given in (4) that are required for learning. Namely, the components are parameters initialization, feature remapping and gate constraints.

1) Parameters initialization

The initialization of gating units in highway blocks is very critical for the successful training of highway networks. Similar to the original highway network block [7], the biases of the gating units are leveraged for controlling the states of the gates. Specifically, we desire gates' states that favour the

routing of mostly untransformed features at the start of training to ease model optimization. Following the formulation given in (4), the biases of the gating units are initialized to relatively small values such as 3 so that the gates are open at the start of training and therefore favour model optimization. Generally, higher bias values are preferred for deeper models [7]. However, Log-Sigmoid units in the proposed highway block can quickly saturate to small values during training, and slow down convergence. In order to tackle this problem, a simple remapping of gating units that is discussed in the following section is employed.

2) Feature gating and re-mapping

For extending the regime of training where the highway block gate routes mostly untransformed features (i.e. in open state and therefore prioritize optimization), model hyperparameters m and n given in (4) are introduced. Since the gating units are Log-Sigmoid units, their outputs are thus bounded as follows: $0 < G^l(H(x)^{l-1}) < 1$. A simple way to emphasize early optimization in training is to put the constraint $n < 1 \leq m$ on the gate as shown in (4); this impedes gate closure on the information highway (i.e. $H(x)^{l-1}$) early in training, and subsequently reduces the impact of gate saturation on optimization. Also, m can be chosen to control the impact of the gate on the highway block transformation path, $F^l(H(x)^{l-1})$. However, as optimization is a major problem for very deep networks, we set $m = 1$ and thus focus on n that directly affects how fast gating units go into saturation. The remapping (or scaling) of gating outputs from $G^l(H(x)^{l-1})$ to $(G^l(H(x)^{l-1}))^n$ for the information highway path $H(x)^{l-1}$ for different values of n is shown in Figure 5. It can be observed in Figure 5 that the output values of the gating units would be maximum (i.e. roughly 1) at the start of training; this is due to the manner in which the gates are initialized; see Section IV.A.1. The arrows show that the output of gating units would decrease as training progresses; however, the remapped output of gating units would decrease much slower. As such, n is a hyperparameter that is set during training.

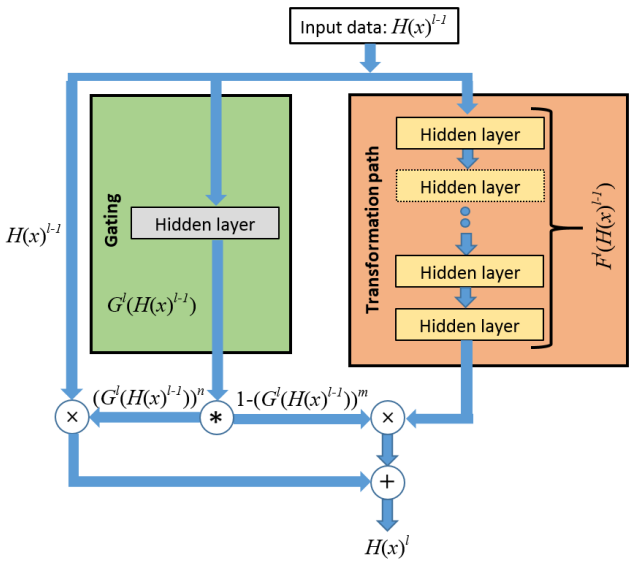

FIGURE 4: Highway network block with gate constraints (HWGC block) [19]

3) Gate constraint

Highway network blocks rely on gates that are parameterized, and therefore can aggravate the problem of model over-fitting. Moreover, in large models with enormous gate parameters, the gates could become flexible enough to hold reasonable amount of latent representations, and as such perform more than alleviating the difficulty of model training; consequently, the transformation path may become somewhat redundant. In order to tackle this problem, the gate parameters are constrained with the hope that their resulting degree of freedom allows for mainly filtering (gating) operation on the information highway, $H(x)^{l-1}$. To achieve this, the max-norm constraint is put on the weights of the gating units; that is, $\| \overrightarrow{w} \| < c$, where w is the weight vector of an arbitrary gating unit. Generally, the appropriate value of the hyper-parameter, c, would be chosen based on model size. This training setup thus enforces the transformation path to learn important features (or representations).

When the different components of the new highway network block are put together, it can be seen that when the gating units go into saturation as it is typical for Log-sigmoid units in deep networks during training, we can write (4) as

$$H(x)^l \approx F^l(H(x)^{l-1}) \quad \text{for} \quad i \gg 1 : 1 \leq i \leq t, \qquad (5)$$

since $G^l(H(x)^{l-1}) \longrightarrow 0$, where notations remain the same as previously stated. From the result given in (5), important implications of the proposed highway block can be drawn as follows:

(i) The proposed highway block essentially learns feature transformations in (5) further in training; this is important for model generalization. This contrasts with the original highway network block that learns mostly untransformed features and prioritize this learning regime

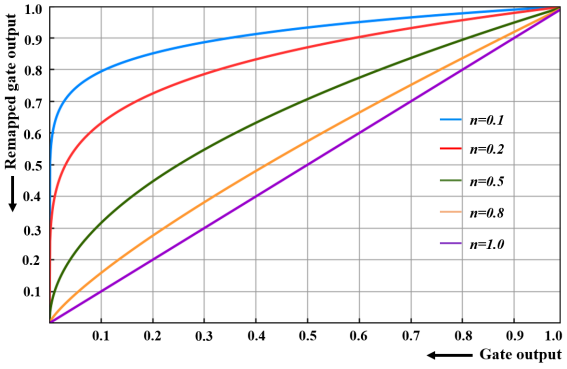

FIGURE 5: Gate remapping

much further in training at the expense of learning new latent representations; see (3) and Figure 3.

(ii) The extreme scenario of the proposed highway block suggests that the proposed model starts out as a relatively 'shallow' model as in (3) and mostly routing untransformed features. However, the model depth grows towards the effective depth as training progresses, since gating units saturate such that we can realize (5). That is, there is nice transitioning from optimization-based to generalization-based regime.

B. HIGHWAY NETWORK WITH GATE CONSTRAINTS, CONCATENATED AND COMPRESSED FEATURES

Computational efficiency is very important in deep network as it is generally desirable to achieve good performance using a model of reasonable size. As such, we take inspiration from recent works that employed fusion of latent representations [12], [40] to propose features concatenation (rather than features summation in HWGC) such that we can leverage features that are already learned at different scales in the earlier layers. The subsequent compression of the output of the proposed highway block allows the extraction of concise representations and maintaining the same number of convolution maps for the succeeding layer. As a result, HWCC generalizes better than HWGC and it can be trained in reasonable time. Thus, it is suitable for applications that require reasonably small inference time. To achieve this, we reformulate the proposed HWGC block as HWCC block to realize the following learning characteristics:

(i) Preserve features that are already learned at different scales in the earlier layers via features concatenation, as opposed to feature summation in [7], [19].

(ii) Reduce features redundancy and proliferation of convolution feature maps for computation via feature compression. This keeps required computation time comparable to HWCC that uses features summation.

For obtaining the aforementioned learning characteristics, HWCC is so formulated that it learns a transformation of the form

$$H_p(x)^l = [F^l(H(x)^{l-1})(1-(G^l(H(x)^{l-1}))^m) \oplus H(x)^{l-1}(G^l(H(x)^{l-1}))^n], \quad (6)$$

where $\oplus$ denotes the *concatenation* operation along convolution maps (or channel) axis. Figure 6 shows the proposed highway network block. The output of HWCC given in (6) is obtained by concatenating the outcome of the gated-transformed features, $F^l(H(x)^{l-1})(1-(G^l(H(x)^{l-1}))^m)$, and gated-untransformed features $H(x)^{l-1}(G^l(H(x)^{l-1}))^n$. However, this can result in features redundancy and doubling the number of convolution maps feeding into the succeeding layer. In order to tackle these problems, an appropriate number of convolutions filters of suitable size can be employed to achieve features compression as shown in Figure 6. Subsequently, using (6), we can write the final block output

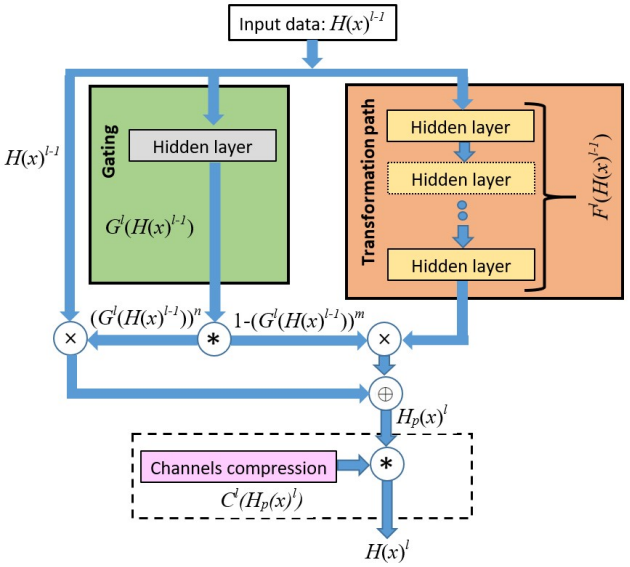

FIGURE 6: Proposed highway network block with gate constraints and concatenated features (HWCC block)

$$H(x)^l = C_k^l(H_p(x)^l), \quad (7)$$

where C_k^l uses r×r convolution filters at layer l, and k is the number of convolution filters. It can be observed that besides the function of reducing the number of output convolution maps for the highway block, compression (bottleneck) operation of C^l can further aid the disentangling of factors of variations in the training data by capturing more concise features representation from both new transformations and transformations that are carried over.

V. EXPERIMENTS

In this section, we describe experiments that are used for validating the proposed highway network block. Specifically, experiments are conducted using five benchmarking datasets which include CIFAR-10[3] and CIFAR-100[3], Fashion-MNIST[4], SVHN[5] and Imagenet[6] datasets. First, experimental settings for the different datasets and models, including hyper-parameters are discussed. Subsequently, results comparisons with the original highway network [7] and state-of-the-art models [8], [12], [18], [22], [41] are given. Lastly, analysis of learned features for the proposed highway blocks gates are presented, along with a high-level discussion that further unravels its manner of operation for successful learning.

The architecture of the most computationally efficient model in this paper on the CIFAR, Fashion-MNIST and SVHN

[3]https://www.cs.toronto.edu/ kriz/cifar.html
[4]https://github.com/zalandoresearch/fashion-mnist
[5]http://ufldl.stanford.edu/housenumbers/
[6]http://image-net.org/challenges/LSVRC/2012/index

TABLE 1: Proposed 19-layer HWCC architecture

19-layer HWCC architecture	
Output size	Input: 32×32×3
32×32	Conv1-64(3×3)
32×32	Conv2-64(1×1)
32×32	Conv3-64(3×3)
32×32	Conv4-96(1×1)
32×32	Conv5-96(1×1)
16×16	MP__1(2×2); stride=2
16×16	Conv6-96(3×3)
16×16	Conv7-96(1×1)
16×16	Conv8-128(1×1)
16×16	Conv9-192(3×3)
7×7	MP__2(3×3); stride=2
32×32	conv10-128(1×1)
32×32	conv11-128(3×3)
32×32	Conv12-192(3×3)
7×7	Conv13-128(1×1)
7×7	Conv14-128(3×3)
7×7	Conv15-192(3×3)
7×7	Highway__conv16-192(1×1)
7×7	Comp.__conv17-128(2×2)
7×7	Conv18-256(1×1)
2×2	MP__3(3×3); stride=3
1×1	2×2 global average pool
C-way softmax	

datasets, 19-layers HWCC, is shown in Table 1; where for convL-M(s×s), L denotes the convolution layer, M is the number of output feature maps and s is the size of convolution filter; Highway__convL-D(s×s) is a highway layer at layer L with D output feature maps; Comp.__convL-D is a compression layer at layer L with D output feature maps. For MP__N(k×k), N denotes the index of the max pooling layer and k is the size of the pooling window. For C-way softmax, C is the number of classes. For reporting experimental results, very deep networks learned using the proposed highway blocks with r×r convolution filters for compression is referred to as 'HWCC'. For the imagenet dataset, we train the HWGC and HWCC models with 34 layers and roughly 43M parameters. In addition, deeper HWGC and HWCC models with 50 layers and roughly 59M parameters are trained.

A. MODEL TRAINING SETTINGS AND EVALUATION CRITERIA

1) Model training settings

For HWGC and HWCC, 19 and 32-layers models are constructed. A similar number of model parameters is used for models with similar depth. Note that compared with the original highway network [7] that employ gates for every layer to alleviate training difficulty, the proposed 19-layers HWGC and HWCC use one gate, while HWGC and HWCC with 32-layers use three gates for successful training. Specifically, gates are employed every 8-13 layers for routing gated untransformed features to alleviate the difficulty of optimization. It is observed from parallel experiments that using a gate for every layer significantly increases the overall number of model parameters, and without any notable improvement in generalization results. The gating units' biases of all models are initialized to a value of 3 at the start of training. The gates re-mapping hyperparameter, n, is set to a value of 0.1; this value is found to give good results on all datasets in this paper. Ablation studies for the hyperparameter, n, in relation to training convergence rates are presented later on in Section V.I. For the compression of the output of the proposed highway blocks, 2×2 convolutional filters are used; the filter size is chosen solely from the perspective of computational efficiency, and thus other filter sizes can be used. A max-norm value of 3 was found suitable for constraining the parameters of gating units. For improving model generalization, units' dropout in the range 5% to 50% are applied in the different hidden layers of the constructed models; dropout is not applied to the gating units. All models are trained using mini-batch gradient decent optimization algorithm with a batch size of 128. A starting learning rate of 0.1 is used; the learning rate is annealed over training to a final value of 10^{-5}. For pre-processing all datasets used in this paper, training and testing data are standardized using training data mean and standard deviation.

2) Evaluation criteria

- Test error rate: note that since our main aim is to demonstrate the effectiveness of the proposed highway blocks for learning very deep networks, we train models of moderate depth and parameters: one model of 19 layers with 1.7M parameters, and the other with 32 layers with 2.6M parameters. It is well known that the performance of deep models can be improved by significantly increasing model depth [8] or width [42]. Consequently, the results reported for the proposed models can be improved by learning much deeper or wider models. Particularly, the original highway network [7] is taken as the baseline model for performance comparison with HWGC and HWCC proposed in this paper.
- Training time: fast training of deep models is important for timely development and testing. Generally, it is desirable that the constructed models require reasonable training time. For comparison, we consider the proposed highway networks against the original highway network [7], ResNet [8] and DenseNet [12].
- Inference time per frame: in many real-life applications, very low latency for inference is of utmost importance to support real-time applications. Therefore, inference time for the proposed highway network in this paper is evaluated against original highway network [7], ResNet [8] and DenseNet [12] which give competitive results on the different datasets.
- GPU memory requirement: typically, minimal GPU memory resource is favoured for training and testing deep neural models. Especially, considering that these models are often deployed for various applications on portable electronic devices with tight GPU memory budgets. As such, the required GPU memory for the proposed highway network in this paper is evaluated against original highway network [7], ResNet [8] and

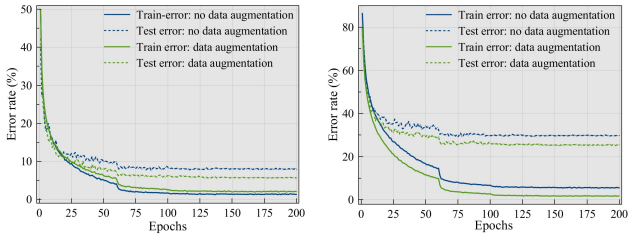

FIGURE 7: Error rate for HWCC-19. Left: CIFAR-10 dataset. Right: CIFAR-100 dataset

DenseNet [12] which give competitive results on the different datasets.

B. CIFAR-10 AND CIFAR-100 DATASETS

CIFAR-10 dataset contains natural RGB images of size 32×32 pixels belonging to 10 different object classes. There are 50,000 and 10,000 samples for training and testing respectively. Experimental results for the dataset without and with data augmentation are given in Table 2 as C10 and C10+, respectively. For data augmentation, we follow the common protocol [7], [8], [42] of only random horizontal flipping, translation by 4 pixels and reflection. The proposed models, HWGC and HWCC, outperform based on achieved test error rates the baseline model, original highway network [7]; see Table 2.

The learning curves showing training and testing error rates without and with data augmentation for HWCC with 19-layers (i.e. HWCC-19) is shown in Figure 7. HWCC-19 achieves error rates of 8.06% and 5.22% without and with data augmentation, respectively. In addition, HWGC-19 achieves error rates of 8.44% and 5.30% without and with data augmentation, respectively. We outperform the baseline model, highway network, ResNet and many state-of-the-art results as given in Table 2. It is noted that though DenseNet [12] achieved lower error rates, without a very careful implementation [45], it has extremely high GPU (Graphics Processing Unit) memory requirements [46]. Also important is that the original highway network [7] requires 400 epochs for training, while HWGC and HWCC require a maximum of 250 and 200 epochs for training, respectively to converge to better test errors. This translates to 37.5% and 50% reduction in the number of training epochs for the original highway network [7], respectively.

CIFAR-100 dataset contains natural RGB images of size 32×32 pixels belonging to 100 different object classes. There are 50,000 and 10,000 samples for training and testing respectively. Experimental results for the dataset without and with data augmentation are given in Table 2 as C10 and C10+, respectively. For data augmentation, we follow the common protocol [7], [8], [42] of only random horizontal flipping, translation by 4 pixels and reflection. Again, HWCC and HWGC outperform based on achieved test error rates the baseline model, original highway network [7]; see Table 2. The learning curves for HWCC-19 showing error rates on the training and testing data without and with data augmentation are shown in Figure 7. HWCC-19 achieves error rates of 29.41% and 24.26% without and with data augmentation, respectively. In addition, HWGC-19 achieves error rates of 29.81% and 24.31% without and with data augmentation, respectively. Furthermore, ResNet and many state-of-the-art results are outperformed as given in Table 2. DenseNet [12] reported lower error rates, but without a very careful implementation, the problem of enormous GPU memory requirement is again a huge concern [46]. Again, it is important to note that the original highway network [7] requires 400 epochs for training, while HWGC and HWCC require a maximum of 300 and 200 epochs for training, respectively. This translates to 25% and 50% reduction in the number of training epochs for the original highway network [7], respectively.

C. FASHION-MNIST DATASET

Fashion-MNIST dataset is a recently collected dataset for benchmarking learning algorithms. The motivation for the collection of the dataset follows from the fact that the popular MNIST handwritten digits dataset is now considered trivial to learn as several works have reported test error rates in the range 0.21-0.50% in recent times. Fashion-MNIST dataset is similar to MNIST dataset. The dataset contains 50,000 and 10,000 training and testing samples, respectively. However, instead of handwritten digits in MNIST dataset, Fashion-MNIST dataset contains images of fashion outfits belonging to 10 different classes that include T-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag and ankle boot. The images are grayscale and of size 28×28 pixels. Fashion-MNIST dataset is considered to be simpler to learn as compared to CIFAR-10 and CIFAR-100 datasets, therefore we train models of smaller depth; that is, only 19-layers HWGC and HWCC. Both HWGC-19 and HWCC-19 are trained for 150 epochs and without data augmentation; obtained test error rates are given in Table 3, along with state-of-the-art results that rely on much more model parameters or depth. It can be seen that HWCC-19 achieves an error rate of 4.87% which, to our knowledge, is the best result reported so far on Fashion-MNIST dataset without data augmentation. The learning curves for HWCC-19 showing error rates on the training and testing data without data augmentation are shown in Figure 8.

D. SVHN DATASET

SVHN (Street View House Number) dataset contains real-life Google street view images with digits in the range 0-9; that is, the images belong to 10 different classes. There are 73,257 and 26,032 digits for training and testing, respectively. In addition, there are 531,131 somewhat easier digits that can be used as extra training data. As with standard training and testing protocol [43], the training and extra training data altogether are used for model training and hyper-parameter selection. For model performance evaluation, the testing data

TABLE 2: Test error rate (%) on CIFAR-10 and CIFAR-100 dataset

Models	Param.	Depth	C10	C10+	C100	C100+
Network in network [23]	1.3M	10	10.41	8.81	35.68	–
Maxout [43]	>6M	5	11.68	9.38	38.57	–
Recurrent CNN [20]	1.86M	–	8.69	7.09	31.75	–
Deeply supervised [21]	1.3M	10	9.69	7.97	–	34.57
All-CNN [41]	1.3M	8	9.08	7.25	–	33.71
ResNet [8]	1.7M	110	13.63	6.41	44.76	27.22
ResNet [18]	1.7M	164	11.26	5.46	35.58	24.33
ResNet [18]	10.2M	1202	10.56*	4.92	33.47	22.71
Wide ResNet [42]	36.5M	28	–	4.17	–	20.50
Wide ResNet [42]	2.2M	40	–	5.33	–	26.04
Weighted ResNet [13]	19.1M	1192	–	5.10	–	–
FractalNet [22]	38.6M	21	7.33	4.60	29.05	23.73
DiracNet [44]	59M	34	–	4.54	–	20.96
DiracNet [44]	3.7M	34	–	5.60	–	26.72
DenseNet (k=24) [12]	1.0M	40	7.00	5.24	27.55	24.42
DenseNet (k=24) [12]	27.2M	100	5.83	3.74	23.42	19.25
Baseline: Highway net [7]	∼2.3M	19	–	7.54	–	32.24
Proposed: HWGC-19*	1.7M	19	**8.44**	**5.30**	**29.81**	**24.31**
Proposed: HWGC-32*	2.6M	32	**8.21**	**5.26**	**29.26**	**24.18**
Proposed: HWCC-19*	1.7M	19	**8.06**	**5.22**	**29.41**	**24.26**
Proposed: HWCC-32*	2.6M	32	**7.96**	**5.14**	**28.77**	**24.02**

*Fair comparison is with models that employ similar number of parameters

TABLE 3: Test error rate (%) on Fashion-MNIST dataset

Models	Param.	Depth	Fashion-MNIST
2C1P2F+Drouout [45]	3.27M	–	8.40
3C1P2F+Dropout [45]	7.14M	–	7.40
GoogleNet [45]	101M	22	6.30
AlexNet [45]	60M	8	10.10
SqueezeNet-200 [45]	0.5M	18	10.00
VGG16 [45]	26M	16	6.50
EvoCNN [45]	6.68M	–	5.47
ResNet [8]	1.7M	110	7.16*
Baseline: highway network [7]	1.7M	19	6.17*
Proposed: HWGC-19	1.7M	19	**5.26**
Proposed: HWCC-19	1.7M	19	**4.87**

*Experiment ran by us

TABLE 4: Test error rate (%) on SVHN dataset

Models	Param.	Depth	SVHN
Network in network [23]	1.3M	10	2.35
Maxout [43]	–	5	2.47
Recurrent CNN [20]	1.86M	–	1.80
Recurrent CNN [20]	2.67M	–	1.77
Deeply supervised [21]	1.3M	10	1.92
FractalNet [22]	38.6M	21	1.87
ResNet [8] (in [12])	1.7M	110	2.01
DenseNet (k=12) [12]	1.0M	40	1.79
DenseNet (k=24) [12]	27.2M	100	1.59
Baseline: highway network [7]	1.7M	19	2.22*
Proposed: HWGC-19	1.7M	19	**1.75**
Proposed: HWCC-19	1.7M	19	**1.71**

*Experiment ran by us

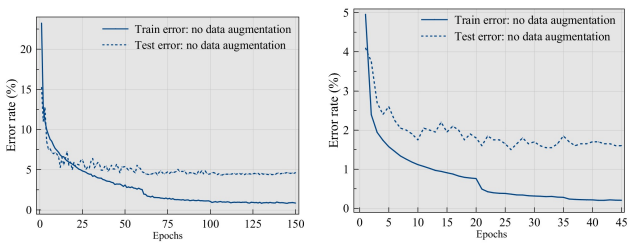

FIGURE 8: Error rate for HWCC-19. Left: Fashion-MNIST dataset. Right: SVHN dataset

is used. Again, the dataset is considered easier to learn, therefore only 19-layers HWGC and HWCC models are trained for the classification problem. Error rates on the testing data are shown in Table 4. The learning curves for HWCC-19 showing the error rates on the training and testing data without data augmentation is shown in Figure 8. The proposed 19-layers HWGC and HWCC with an error rates of 1.75% and 1.71% respectively outperform the original highway network [7], ResNet with 110 layers [8] and other state-of-the-art models based on similar number of model parameters.

E. IMAGENET-2012 (ILSVRC) DATASET

The imagenet dataset contains natural images of 1.2M samples belonging to 1000 different classes. The dataset contains 50,000 natural images for testing. All images are resized to 224×224 pixels for training. The images are augmented during training using random horizontal flips. For testing, the

TABLE 5: Test error rate (%) on imagenet-2012 (ILSVRC) dataset

Models	Param.	Depth	ILSVRC
ResNet [8]	44.5M	101	22.56
ResNet [8]	60.2M	152	21.57
DenseNet (k=48) [12]	30.6M	161	22.33
Baseline: highway network [7]	43.6M	34	23.45*
Baseline: highway network [7]	59.3M	50	22.72*
Proposed: HWGC-34	43.6M	34	**22.46**
Proposed: HWGC-50	58.8M	50	**21.31**
Proposed: HWCC-34	43.1M	34	**22.17**
Proposed: HWCC-50	58.3M	50	**20.78**

*Experiment ran by us

TABLE 6: Training time per epoch on CIFAR-10 dataset

Models	Param.	Depth	Time (s)*
ResNet [18]	1.7M	164	151
ResNet [8]	1.7M	110	106
DenseNet [12]	1.0M	40	175
Baseline: highway network [7]	1.7M	19	55
Proposed: HWGC-19	1.7M	19	**52**
Proposed: HWCC-19	1.7M	19	**48**

*Result with a batch size of 128 on a single Nvidia GTX Titan X GPU

images are first resized to 256×256, and then center crops of 224×224 pixels are fed to the trained models. We train the HWCC and HWGC models both with 34 and 50 layers on the imagenet dataset for 120 epochs. The initial learning rate, which is set to 0.1 is annealed during training to the final value of 10^{-3}. The momentum rate is set to 0.9, while the weight decay penalty is set to 10^{-4}; a training batch size of 64 is used. The training and testing errors per epoch over the training and testing data are not computed during training, since it considerably increases the training time by several hours; the models are simply tested at the end of training.

The top-1 error rates obtained from the trained models are reported in Table 5, where the HWGC and HWCC models outperform the baseline model, the original highway network. Considering models with similar number of parameters, the proposed models with 34 layers slightly outperforms the ResNet-101, which is roughly three times deeper. Similarly, the proposed models with 50 layers surpass the ResNet with 152 layers that is roughly three times deeper. We note that DenseNet-161 with fewer parameters slightly outperforms the proposed models with 34 layers. However, we show later on in Section V.F and Section V.G that the DenseNet requires more training/testing time and GPU memory, respectively; these are crucial limitations on the practical applicability of the DenseNet. Overall, these observations altogether strengthen the proposed reformulations of the original highway network for improved learning.

TABLE 7: Inference time per frame for CIFAR-10 dataset

Models	Param.	Depth	Time (ms)*
ResNet [18]	1.7M	164	1.56
ResNet [8]	1.7M	110	1.21
DenseNet [12]	1.0M	40	0.69
Baseline: highway network [7]	1.7M	19	0.41
Proposed: HWGC-19	1.7M	19	**0.39**
Proposed: HWCC-19	1.7M	19	**0.35**

*Result with a single Nvidia GTX Titan X GPU

TABLE 8: Inference time per frame for imagenet dataset

Models	Param.	Depth	Time (ms)*
ResNet [18]	60.2M	152	49.53
DenseNet (k=48) [12]	30.6M	161	55.63
Baseline: highway network [7]	59.3M	50	34.61
Proposed: HWGC-50	58.8M	50	**34.55**
Proposed: HWCC-50	58.3M	50	**34.79**

*Result with two Nvidia Titan V GPUs

F. TRAINING AND INFERENCE TIME

Training time: As earlier discussed in Section V.A.2, aside obtained error rates, other evaluation criteria that are of interest in this paper such as training and inference time are observed. Table 6 shows the required time per epoch on CIFAR-10 dataset for models with similar number of model parameters; a training batch-size of 128 samples is used for all models. For comparison, we select the original highway network [7] (considering number of parameters and depth), ResNet [8], [18] and DenseNet [12] models that give similar results with the proposed highway networks on CIFAR-10 and CIFAR-100 datasets; see Table 6. DenseNet has the highest training time of 175 secs per epoch, which is expected considering its complicated architecture as every layer's output is connected to all other layers. The ResNet-164 and ResNet-110 require lesser training times of 151 secs and 106 secs per epoch, respectively. HWGC-19 and HWCC-19 require training times of 52 and 48 secs per epoch respectively, while the original highway network [7] requires 55 secs. The original highway network [7] is slightly slower to train than HWGC and HWCC; this may be attributed to the fact it uses highway blocks (i.e. two parallel paths) for every layer, while HWGC-19 and HWCC-19 uses only one highway block. However, HWGC-19 and HWCC-19 require 250 and 200 training epochs respectively, and as such have overall smaller training times than the original highway network [7] which requires 400 training epochs.

Inference time: For inference time evaluation, the time to perform inference on the entire CIFAR-10 dataset is reported in Table 7. It can be seen that HWCC-19 offers a significant reduction in inference time compared to ResNet [8], [18] and DenseNet [12], while at the same time matching or even outperforming (on achieved test error rates) much deeper ResNet

models. HWGC-19 and HWCC-19 have inference times that are comparable to the original highway network [7]. However, note that the original highway network, ResNet-110 and ResNet-164 achieved higher test errors than the proposed highway networks; see Tables 2, 3, & 4. The interesting results of the proposed highway model can be attributed to the fact that it makes more efficient use of model depth (smaller depth results in improved results) following the analysis and construction given in section IV. From Table 7, HWGC-19 and HWCC-19 require inference times of 0.39 and 0.35 secs respectively, while the original highway network requires 0.41 secs (i.e. comparable to the proposed highway networks), DenseNet requires 0.69 secs (i.e. 97.1% increase in inference time from HWCC-19), ResNet-164 requires 1.56 secs (i.e. 345.7% increase in inference time from HGCC-19) and ResNet-110 requires 1.21 secs (i.e. 245.7% increase in inference time). Furthermore, the inference times per frame for the best models trained on the imagenet dataset in Table 5 are reported in Table 8, where it is seen that the proposed models, HWGC and HWCC, require considerably smaller inference time than the significantly deeper ResNet and DenseNet.

It can be observed that the much deeper ResNets have limited use for supporting applications that require fast inference. The proposed model requires an inference time that is similar to those for the original highway network [7] and highway network with gates constraints [19], however with better generalization performance. We acknowledge that DenseNets require fewer parameters but increased depth to achieve similar results for the proposed model. Unfortunately, DenseNets incur much higher inference time and without a careful implementation require enormous GPU memory resources. Moreover, a careful implementation of DenseNet to reduce GPU memory footprint increases overhead computation cost, and as such further increases training and inference time [46]. This can limit its usefulness for many practical applications that require low latency for operation.

G. GPU MEMORY REQUIREMENT

The GPU memory requirement for training deep neural networks is a very critical attribute of a *truly* successful model. The goal is to build deep neural networks that require modest GPU memory. Consequently, using the CIFAR-10 dataset, the evaluation of the proposed models against the original highway network, ResNet-110, ResNet-164 and DenseNet are given in Table 9. It will be seen that the proposed models, HWGC-19 and HWCC-19, have similar requirements to the original highway network, while ResNet and DenseNet clearly have much higher requirements. Particularly, observe that DenseNet with 40 layers and only 1M parameters has a higher GPU memory requirement than even ResNet with 110 layers and 1.7M parameters. Furthermore, Table 10 shows the GPU memory requirements for the models trained on the imagenet dataset. Keeping in mind from Table 5 that the proposed models (i.e. HWGC and HWCC) can match or outperform the ResNet and DenseNet, Table 10 shows that the proposed models require smaller GPU memory than the ResNet and DenseNet. Ultimately, Table 9 and Table 10 results motivate the computational efficiency of the proposed models in comparison to others.

TABLE 9: GPU memory resource for CIFAR-10 dataset

Models	Param.	Depth	GPU (GB)*
ResNet [18]	1.7M	164	3.94
ResNet [8]	1.7M	110	1.76
DenseNet [12]	1.0M	40	2.36
Baseline: highway network [7]	1.7M	19	0.98
Proposed: HWGC-19	1.7M	19	**1.00**
Proposed: HWCC-19	1.7M	19	**1.06**

*Result with a batch size of 128

TABLE 10: GPU memory resource for Imagenet dataset

Models	Param.	Depth	GPU (GB)*
ResNet [18]	44.5M	101	14.57
ResNet [8]	60.2M	152	29.92
DenseNet (k=48) [12]	30.6M	161	16.29
Baseline: highway network [7]	43.6M	34	11.65
Baseline: highway network [7]	59.3M	50	25.23
Proposed: HWGC-34	43.6M	34	**11.33**
Proposed: HWCC-50	58.8M	50	**25.41**
Proposed: HWGC-34	43.1M	34	**11.79**
Proposed: HWCC-50	58.3M	50	**25.96**

*Result with a batch size of 64

H. QUALITATIVE EVALUATION OF GATING UNITS ACTIVATIONS

For qualitative evaluation and analysis, the activations (response) of gating units in HWCC to the different image classes are observed. Specifically, gating units in the first highway network block of HWCC-19 model are observed; similar results are obtained for HWCC with 32 layers, and HWGC with 19 and 32-layers. The major motivation for this is to verify that the gating units indeed perform some sort of feature filtering; this is a nice learning characteristic as earlier stated in Section III.B. Some of the gates' convolution maps in the first HWCC-19 block using randomly chosen images for the 10 different classes in CIFAR-10 dataset are shown as grayscale images in Figure 9. It is straightforward to observe from the convolution maps which appear as binary *discriminative masks* for important parts in the different images that the gating units actually perform some sort of feature filtering that may make learning more interesting than for ResNet which lacks this attribute.

I. ABLATION STUDIES

For further validation of the reformulated gating blocks proposed in this paper, we perform ablation studies to study how the gate remapping hyperparameter, n, presented in

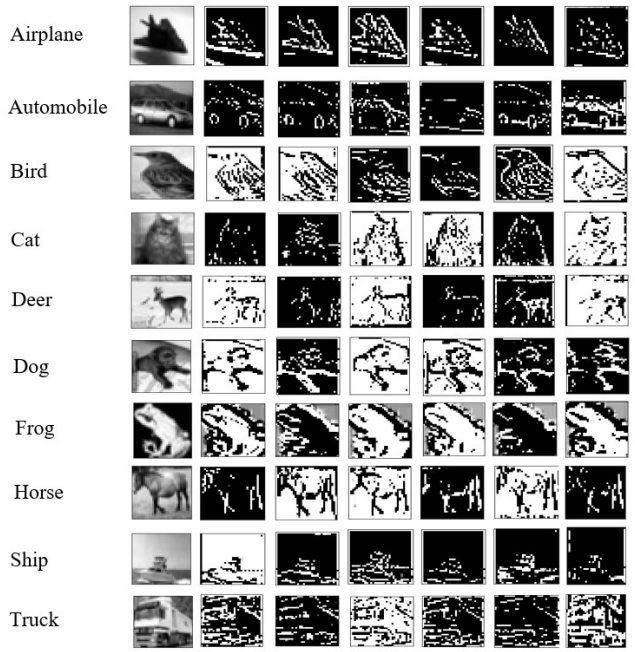

FIGURE 9: Gating units' activations (responses) for the first highway block in HWCC-19 trained on CIFAR-10 dataset. The first column shows the input images for the 10 different classes; other columns show units' activations for different convolution maps

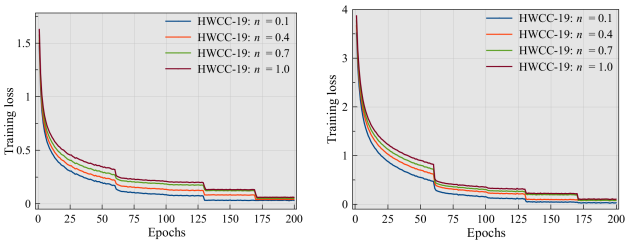

FIGURE 10: Convergence rate for HWCC-19. Left: CIFAR-10 dataset. Right: CIFAR-100 dataset

Section IV.A.2 impacts the rate of training convergence for the proposed models. Figure 10 shows the training loss curve for different values of n using HWCC-19 on CIFAR-10 and CIFAR-100 datasets, respectively; similar observations are made for the HWGC models. It is seen for both datasets that training converges faster for smaller values of n as expected and discussed in Section IV.A.2.

VI. CONCLUSION

Very deep neural networks have been shown empirically and theoretically to have better function approximating capacity as compared to shallow networks. However, training very deep networks is hard especially from an optimization perspective. This paper proposes two highway networks with gate constraints, HWGC and HWCC, for improving the training of very deep networks. The capability to control learning nearly exact identity functions by a clever construction and formulation of the proposed highway blocks is shown analytically and experimentally to allow a more effective usage of model depth than the original highway network. As such, the proposed models generalize better than original highway network and other state-of-the-art models.

Furthermore, experimental results using CIFAR-10, CIFAR-100, Fashion-MNIST, SVHN and imagenet datasets show that using the same number of model parameters, the proposed HWGC and HWCC models with 19-layers mostly outperform ResNets with 110 and 164 layers on all the benchmarking datasets used in this paper; many state-of-the-art models are also outperformed using the proposed model. In addition, experimental results using the imagenet dataset show that the proposed HWGC and HWCC, with 50 layers can match or outperform the ResNet with 101 and 152 layers. From this observation, we conclude that HWGC and HWCC models are more computationally and memory efficient than the original highway network and ResNet, since similar or better test errors can be achieved with a significantly smaller depth that requires much smaller training time, and especially inference time and GPU memory for supporting real-time applications.

REFERENCES

[1] G. G. Demisse, K. Papadopoulos, D. Aouada, and B. Ottersten, "Pose encoding for robust skeleton-based action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 188–194.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[3] K. Papadopoulos, M. Antunes, D. Aouada, and B. Ottersten, "Enhanced trajectory-based action recognition using human pose," in *Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1807–1811.

[4] O. K. Oyedotun and A. Khashman, "Prototype-incorporated emotional neural network," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 8, pp. 3560–3572, 2018.

[5] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *Advances in neural information processing systems*, 2014, pp. 2627–2635.

[6] O. K. Oyedotun, A. E. R. Shabayek, D. Aouada, and B. Ottersten, "Improving the capacity of very deep networks with maxout units," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.

[7] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in neural information processing systems*, 2015, pp. 2377–2385.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] O. Delalleau and Y. Bengio, "Shallow vs. deep sum-product networks," in *Advances in Neural Information Processing Systems*, 2011, pp. 666–674.

[10] H. Mhaskar, Q. Liao, and T. Poggio, "Learning functions: when is deep better than shallow," *arXiv preprint arXiv:1603.00988*, 2016.

[11] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 8, pp. 1553–1565, 2014.

[12] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 2261–2269.

[13] F. Shen, R. Gan, and G. Zeng, "Weighted residuals for very deep networks," in *Systems and Informatics (ICSAI), 2016 3rd International Conference on*. IEEE, 2016, pp. 936–941.

[14] O. K. Oyedotun, A. E. R. Shabayek, D. Aouada, and B. Ottersten, "Training very deep networks via residual learning with stochastic input shortcut connections," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 23–33.

[15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[19] O. K. Oyedotun, A. El Rahman Shabayek, D. Aouada, and B. Ottersten, "Highway network block with gates constraints for training very deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1658–1667.

[20] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3367–3375.

[21] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.

[22] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," in *International Conference on Learning Representations (ICLR)*, 2017.

[23] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[24] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.

[25] G. Philipp, D. Song, and J. G. Carbonell, "Gradients explode-deep networks are shallow-resnet explained," *arXiv preprint arXiv:1712.05577*, 2017.

[26] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013.

[27] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 971–980.

[28] D. Balduzzi, M. Frean, L. Leary, J. Lewis, K. W.-D. Ma, and B. McWilliams, "The shattered gradients problem: If resnets are the answer, then what is the question?" in *International Conference on Machine Learning*, 2017, pp. 342–350.

[29] A. Zaeemzadeh, N. Rahnavard, and M. Shah, "Norm-preservation: Why residual networks can become extremely deep?" *arXiv preprint arXiv:1805.07477*, 2018.

[30] S. Li, J. Jiao, Y. Han, and T. Weissman, "Demystifying resnet," *arXiv preprint arXiv:1611.01186*, 2016.

[31] H. Li, Z. Xu, G. Taylor, and T. Goldstein, "Visualizing the loss landscape of neural nets," *arXiv preprint arXiv:1712.09913*, 2017.

[32] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[33] A. de Brebisson and G. Montana, "Deep neural networks for anatomical brain segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 20–28.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[35] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] B. Xu, R. Huang, and M. Li, "Revise saturated activation functions," in *International Conference on Learning Representations (ICLR) Workshop*, 2016.

[40] O. K. Oyedotun, G. G. Demisse, A. E. R. Shabayek, D. Aouada, and B. E. Ottersten, "Facial expression recognition via joint deep learning of rgb-depth map latent representations." in *ICCV Workshops*, 2017, pp. 3161–3168.

[41] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[42] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference (BMVC)*, E. R. H. Richard C. Wilson and W. A. P. Smith, Eds. BMVA Press, 2016, pp. 87.1–87.12.

[43] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning-Volume 28*. JMLR. org, 2013, pp. III–1319.

[44] S. Zagoruyko and N. Komodakis, "Diracnets: training very deep neural networks without skip-connections," *arXiv preprint arXiv:1706.00388*, 2017.

[45] Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks for image classification," *arXiv preprint arXiv:1710.10741*, 2017.

[46] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger, "Memory-efficient implementation of densenets," *arXiv preprint arXiv:1707.06990*, 2017.

OYEBADE OYEDOTUN (S'14) received the M.Sc. degree in electrical and electronic engineering from Near East University, Lefkosa, Turkey, in 2015. He is currently pursuing the Ph.D. degree with the Computer Vision, Imaging and Machine Intelligence (CVI^2) Research Group at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855 Luxembourg with a focus on deep learning, machine learning and vision applications. He has authored and co-authored many scientific articles in leading IEEE conferences and journals, including Transactions on Neural Networks and Learning Systems. He reviews for several journals, including IEEE TNNLS, IEEE TKDE, IEEE GRSL, IEEE Access, IEEE TAFFC, Neural Computing and Applications. His current research interests include machine learning and vision applications, neural networks, cognition modeling and neuroscience.

ABD EL RAHMAN SHABAYEK is a Research Associate at the SnT since June 2016 as a member of the CV Lab in SIGCOM and currently CVI^2 research group, and a visiting scholar in the VIBOT Erasmus Mundus program on computer vision and Robotics since 2014. He obtained his PhD (2012, France) and Erasmus Mundus MSc (2009, UK, Spain, France) in computer vision and robotics. He was an assistant professor in the Faculty of Computers and Informatics, Suez Canal University and an adjunct assistant professor in both Information Technology Institute and Sinai University as well as a part-time project manager in EULA-Soft in Egypt from 2014 to 2016 and a research fellow in LITIS, University of Rouen, and Le2i-CNRS, University of Bourgogne, France in 2013. His research interests include 3D and non-conventional computer vision, 3D and 2D conventional and geometric deep learning, medical imaging and remote sensing. He actively participated in attracting both academic and industrial funds for different research projects in CVI^2. He is the author and co-author of over 40 publications including two best paper awards. Dr. Shabayek is a Member of the IEEE, the IEEE Signal Processing Society, and INSTICC.

DJAMILA AOUADA is a Senior Research Scientist and an Assistant Professor at the Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg. She is Head of the Computer Vision, Imaging and Machine Intelligence (CVI^2) Research Group at SnT, and Head of the SnT Computer Vision Laboratory. Dr. Aouada received the State Engineering degree in electronics in 2005, from the ÃL'cole Nationale Polytechnique (ENP), Algiers, Algeria, and the Ph.D. degree in electrical engineering in 2009 from North Carolina State University (NCSU), Raleigh, NC, USA. Dr. Aouada has worked as a consultant for multiple renowned laboratories (Los Alamos National Laboratory, Alcatel Lucent Bell Labs., and Mitsubishi Electric Research Labs.). She has been leading the computer vision activities at SnT since 2009. Her research interests span the areas of image processing, computer vision, pattern recognition and data modelling. Dr. Aouada is Senior Member of the IEEE, member of the IEEE Signal Processing Society, IEEE WIE, INSTICC and the Eta Kappa Nu honor society (HKN). She has served as the Chair of the IEEE Benelux Women in Engineering Affinity Group from 2014 to 2016. She is the co-author of four IEEE best paper awards.

BJÖRN OTTERSTEN was born in Stockholm, Sweden, 1961. He received the M.S. degree in electrical engineering and applied physics from LinkÃűping University, LinkÃűping, Sweden, in 1986. In 1989 he received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA. Dr. Ottersten has held research positions at the Department of Electrical Engineering, LinkÃűping University, the Information Systems Laboratory, Stanford University, the Katholieke Universiteit Leuven, Leuven, and the University of Luxembourg. During 96/97 Dr. Ottersten was Director of Research at ArrayComm Inc, a start-up in San Jose, California based on OtterstenâĂŹs patented technology. He has co-authored journal papers that received the IEEE Signal Processing Society Best Paper Award in 1993, 2001, 2006, and 2013 and 7 IEEE conference papers receiving Best Paper Awards. In 1991 he was appointed Professor of Signal Processing at the Royal Institute of Technology (KTH), Stockholm. From 1992 to 2004 he was head of the department for Signals, Sensors, and Systems at KTH and from 2004 to 2008 he was dean of the School of Electrical Engineering at KTH. Currently, Dr. Ottersten is Director for the Interdisciplinary Centre for Security, Reliability and Trust at the University of Luxembourg and is a board member of the Swedish Research Council. From 2012 to 2018 Dr. Ottersten was Digital Champion of Luxembourg, acting as an adviser to the European Commission. Dr. Ottersten has served as Editor in Chief of EURASIP Signal Processing, Associate Editor for the IEEE Transactions on Signal Processing and on the editorial board of IEEE Signal Processing Magazine. He is currently a member of the editorial boards of EURASIP Signal Processing, EURASIP Journal of Applied Signal Processing and Foundations and Trends in Signal Processing. Dr. Ottersten is a Fellow of the IEEE and EURASIP. In 2011 he received the IEEE Signal Processing Society Technical Achievement Award. He has received the European Research Council advanced research grant twice, 2009-2013 and 2017-2022. His research interests include security and trust as well as signal processing in wireless communications, radar and computer vision.