A New Approach to Neural Network Design for Fast Convergence via Feed-forward Loop

Nam Guk Kim

***Abstract*—A feed-forward loop has been widely used in control systems to boost the performance without hurting the overall stability of the system. We propose a new neural network, *FfcNet* by adopting the idea from the control theory. The proposed network adds a pre-designed feed-forward loop in parallel with the existing regular blocks, which is similar to the identity mapping in *ResNet* network. The feed-forward loop helps the overall network to converge faster while keeping overall system stability and accuracy. It is also shown that the feed-forward loop is equivalent to setting a proper initial condition of the parameters in the network. A special dataset of highly distorted 7-segment LED images is prepared to evaluate the performance of the pattern recognition algorithm. We demonstrated the performance of the proposed design through simulations and found the new design improved the convergence rate by 52% from the original *ResNet* network while keeping the same test accuracy.**

***Index Terms*—Fast convergence, feed-forward loop, *FfcNet*, pattern recognition, *ResNet*.**

I. Introduction

Pattern recognition or image classification problem is one of the most popular topics in the recent machine learning area since it is applicable to a wide range of applications such as autonomous driving, fault detection, disease diagnosis and etc. Among the algorithms, *ResNet* network is regarded as one of the most popular and effective algorithms. A series of *ResNet* algorithms [1]-[6] have been proposed since the concept was first proposed in [1], [2]. At the heart of their proposed residual network is the idea that the layers have an identity function in parallel with the conventional convolution neural network as shown in Fig. 1. It has shown the best performance with some variants such as changing the identity function to 1×1 convolution or even its logical extension to so-called *DenseNet* in [3]. With the help of the increased computational power by GPU, the trends in *ResNet* design were adding more layers to improve the accuracy rather than having a structural improvement except for the approaches in [4]-[6], where some structural variations were proposed to improve the speed/accuracy for specific applications. However, the requirement of a big computational power or a slow adaptation rate to train millions of nodes is still a big challenge to the current machine learning area.

A feed-forward loop has been widely and effectively used in many industries. We can directly compensate the external disturbances using accelerometers or gyroscopes during the motion control. We can also apply very sophisticated learning schemes for repetitive tasks as in [7]-[9]. A typical block diagram of a feed-forward controller is shown in Fig. 2. The main advantage of using a feed-forward loop is that it can improve the tracking performance or convergence time without sacrificing the overall system stability. We can extend the well-known Internal Model Principle (IMP) of control theory [10] to the feed-forward control system in Fig. 2 to state that we can achieve a perfect tracking of the reference signal $\mathrm{r(t)} = \sin \omega t$ if the feed-forward controller $\mathrm{C}_f(\mathrm{s})$ is chosen to have $1/(s^2+\omega^2)$ or model of $\mathrm{r(t)}$.

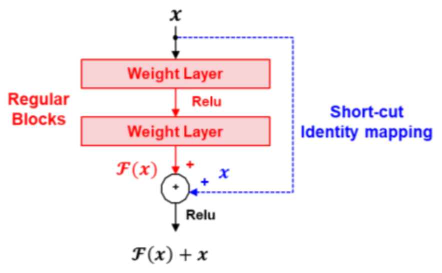


Fig. 1. *ResNet* network with an identity mapping as a residual block.

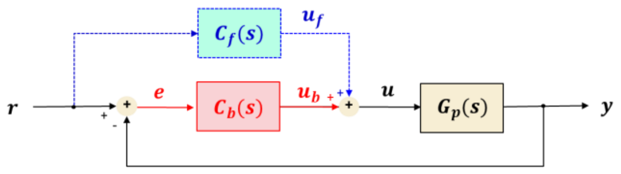


Fig. 2. Block diagram of a typical feed-forward controller.

We often get some ideas or breakthroughs from other domains when we are stuck in difficult problems. It is well-known to control engineers that the design of the state observer is equivalent to the design of the state feedback controller [10]. On the other hand, a preliminary study on the relationship between ILC (Iterative Learning Control) and RL (Reinforced Learning) was given in [11] as well. It is shown that their performance of the error convergence is exactly same when the update gains were chosen in the same way. This kind of studies on equivalence shed some light on the consecutive studies through applying the traditional control theory to the state-of-art machine learning domain.

Inspired by the above 3 prior studies, we propose a new neural network, *FfcNet*, which adds a pre-designed feed-forward loop in parallel with the conventional CNN (Convolution Neural Network). Kindly note that the paths shown as the dotted lines in Fig. 1,2 are feed-forward paths. We will show that the feed-forward loop in the *FfcNet* can make the overall system faster as in the feed-forward control and give similar benefits as the identity mapping in *ResNet* network. We further insist that adding a feed-forward loop is

Manuscript received January 6, 2022; revised January 27, 2022.
Nam Guk Kim is with Cybernetics Imaging Systems, Changwon, Gyeongsangnam-do, Korea, 51391 (e-mail: ngkim823@gmail.com).

equivalent to setting an appropriate initial condition to the estimator. A rigorous derivation of the equivalence is not given here but we will illustrate it with a simple parameter estimation problem. We hope this to motivate the control engineers to dive into the modern machine learning area and continue to contribute the deployment of theory and application from a different viewpoint.

II. Motivation and Preliminary Results

In this chapter, we present motivations of the proposed algorithm along with some preliminary studies for the problem definition.

A. Motivation

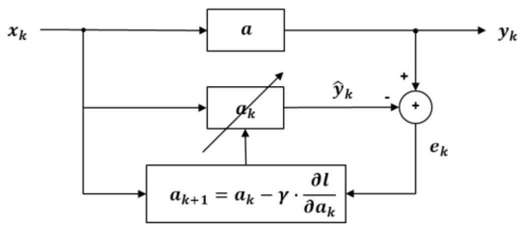
Consider the below SISO system $\sum$ with a single unknown parameter a:

$$\Sigma:\ y = a \cdot x \tag{1}$$

where $x, y, a \in \mathbb{R}$ are the input, output, and gain of the system, respectively. Let us consider a simple parameter estimator Ω such that

$$\Omega: \begin{cases} y_k = a \cdot x_k \\ \hat{y}_k = a_k \cdot x_k \\ e_k = y_k - \hat{y}_k \\ z_k = a - a_k \\ a_{k+1} = a_k + \gamma \cdot e_k \cdot x_k \\ l(e_k) = \frac{1}{2}\|e_k\|^2 \end{cases} \tag{2}$$

where for k^{th} iteration, x_k denotes the input x, y_k denotes the output of y, $\hat{y}_k$ denotes estimated value of output y_k, e_k denotes estimation error of output, $y_k - \hat{y}_k$, z_k denotes estimation error of the parameter, $a - a_k$, and $l(e_k)$ denotes the loss function, respectively. Here, we used the so-called gradient decent method to update the parameter a_k to the direction of minimizing the loss function, that is, $a_{k+1} = a_k - \gamma \cdot \frac{\partial l}{\partial a_k}$.



x_k
a
y_k
$\hat{y}_k$
a_k
e_k
$a_{k+1} = a_k - \gamma \cdot \frac{\partial l}{\partial a_k}$

Fig. 3(a). Block diagram of the simple estimator Ω.

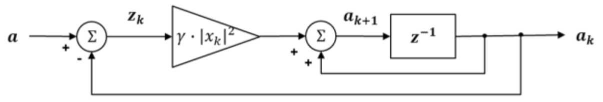
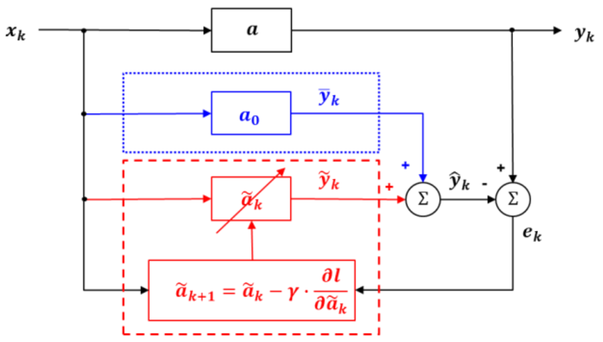


Fig. 3(b). Equivalent block diagram of Fig. 3(a): a regulation problem.

The simple parameter estimator defined in (1) ~ (2) can be depicted as the block diagram in Fig. 3(a), which can be easily transformed into another block diagram in Fig. 3(b). Kindly note that the original estimation problem in Fig. 3(a) was transformed into a set point regulation problem (sort of control problem) with a time-varying gain $\gamma \cdot |x_k|^2$ in Fig. 3(b). The estimation error converges to zero, or equivalently, a_k converges to a if the update gain γ is chosen small enough so that $0 < \gamma \cdot |x_k|^2 < 1$.

Now, let's choose an initial condition of the estimated parameter a_k as a_0. First, we change variables as $\tilde{a}_k = a_k - a_0$, $\hat{y}_k = \bar{y}_k + \tilde{y}_k$, where $\bar{y}_k = a_0 \cdot x_k$ and $\tilde{y}_k = \tilde{a}_k \cdot x_k$. Then, the block diagrams shown in Fig. 3 can be changed into Fig. 4. It is quite obvious that $\tilde{a}_k$ will converge to $a - a_0$ faster as we choose a_0 closer to the true value of a. As shown in Fig. 4, this simple initial condition setting can be interpreted as adding a feed-forward term a_0 to the original system as in the dotted boxes. The faster convergence of the simple estimator with a proper initial condition a_0 matches quite well with the benefit of reducing the tracking error faster by adding a feed-forward loop into the original control loop.



x_k
a
y_k
a_0
$\bar{y}_k$
$\tilde{y}_k$
$\tilde{a}_k$
$\hat{y}_k$
e_k
$\tilde{a}_{k+1} = \tilde{a}_k - \gamma \cdot \frac{\partial l}{\partial \tilde{a}_k}$

Fig. 4(a). Block diagram of the simple parameter estimator Ω with an initial condition a_0.

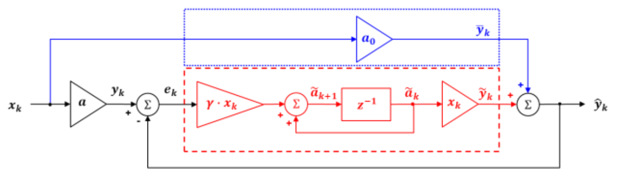


Fig. 4(b). Equivalent block diagram of Fig. 4(a): a feed-forward loop.

B. Problem Definition and Preliminary Results

Now, let's move into a pattern recognition problem. MNIST datasets has been widely used in Kaggle or in-class competition for the performance evaluation of machine learning algorithms. However, we prepared a special set of images to provide a better quantitative analysis on the effect of various noise factors. As shown in Fig. 5, images of digits from 0 to 9, which represent the 7-segment LED images, are specially designed. The image has 31(H)x19(W) resolution or 589 pixels and it is highly distorted by adding measurement noises, turning off some pixels, and shifting the whole images to some directions.

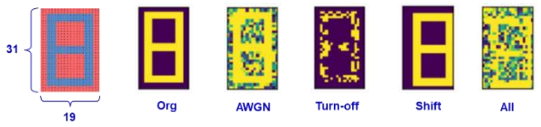


Fig. 5. Illustration of the datasets used in the paper to evaluate the performance of various networks.

First, we model the image recognition system for the distorted 7-segment LED images as

$$\mathbf{y}^i = \mathbf{h}(\boldsymbol{x}^i, \mathbf{d}) + \mathbf{n}, \text{ for } i = 0, 1, 2, \ldots, 9. \tag{3}$$

where $\boldsymbol{x}^i \in \mathbb{R}^{589}$ is an input vector obtained by serializing the bit image of the figure 'i' from left to right and top to bottom and $\boldsymbol{y}^i \in \mathbb{R}^{10}$ is an output vector representing the figure 'i' with $y_{i,j} = \delta_{i,j}$, where $y_{i,j}$ is the j^{th} element of the vector $\boldsymbol{y}^i$ and $\delta_{i,j}$ is the Kronecker delta function such that $\delta_{i,j} = 1$ if $i = j$ and 0 if $i \neq j$. $\boldsymbol{n} \in \mathbb{R}^N$ represents a measurement noise and $\mathbf{d} \in \mathbb{R}^D$ represents structural disturbances like bad cells and translative shifts in the 7-segment LED images stated in the previous paragraph. Now, our problem is to find the best estimate of $\boldsymbol{y}^i$ (or the functions $\mathbf{h}$ in (3)) from the measurement $\boldsymbol{x}^i$ under the measurement noise $\mathbf{n}$ and the structural disturbance $\mathbf{d}$. In other word, to find $\hat{\mathbf{g}} : \mathbb{R}^{10\times589} \rightarrow \mathbb{R}^{10\times10}$ such that

$$\min_{\hat{\mathbf{Y}}} l(\mathbf{Y}, \hat{\mathbf{Y}}) = \min_{\hat{\mathbf{g}}} \|\mathbf{Y} - \hat{\mathbf{g}}(\mathbf{X})\|^2 \qquad (4)$$

Here, $\mathbf{Y} \in \mathbb{R}^{10\times10}$ is the augmented output vector and $\mathbf{X} \in \mathbb{R}^{10\times589}$ is the augmented input vector such that $\mathbf{Y}^T = [(\boldsymbol{y}^0)^T, (\boldsymbol{y}^1)^T, \cdots, (\boldsymbol{y}^9)^T]$ and $\mathbf{X}^T = [(\boldsymbol{x}^0)^T, (\boldsymbol{x}^1)^T, \cdots, (\boldsymbol{x}^9)^T]$. Here, we limit our problem in finding the unknown parameters of the known function $\mathbf{f}$ rather than searching the optimal function in the whole functional space. That is, the problem in (4) can be changed into

$$\min_{\mathbf{a}} \|\mathbf{Y} - \mathbf{f}(\mathbf{a}, \mathbf{X})\|^2 \qquad (5)$$

One of the simplest methods to get a good estimate is to model the system in (3) as a linear model and use the framework of the linear regression, that is, to find $\widehat{\mathbf{W}}$ and $\hat{\mathbf{b}}$ in

$$\mathbf{f}(\hat{\mathbf{a}}, \mathbf{X}) = \mathbf{X} \cdot \widehat{\mathbf{W}} + \hat{\mathbf{b}} \qquad (6)$$

satisfying (5). Then, we can just put the input/output datasets into the regression model or neural network to get the best estimate of the parameters under the given assumptions.

A simple FC (fully connected) linear network and a simple *ResNet* network are shown in Fig. 6 as an example of the neural network to get the solution of the above problem. The FC linear network directly connects 589 inputs to 10 outputs through weights and biases. The *ResNet* network is composed of a series of 2D convolution and 2D average pool followed by a FC linear network. Note that the design of these simple networks is not the main topic of this paper but we introduced it here for the later use when we design our FfcNet network. Detailed design and parameters of the *ResNet* network will be explained in the later chapters.

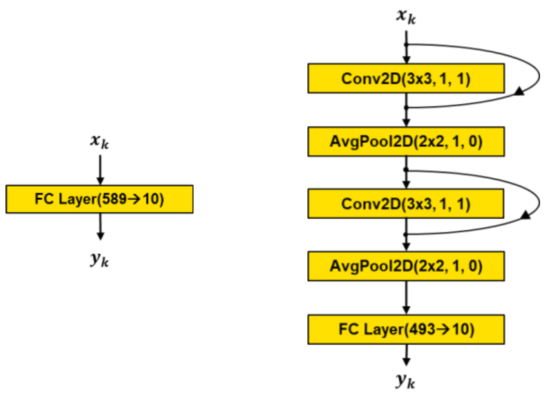


Fig. 6. Typical FC linear network(left) and *ResNet* network(right).

Fig. 7 shows the performance of each network after 100 iterations (200 iterations for the FC linear networks since it converges too slow). As expected, both networks are running fairly well for the above simple pattern recognition problem. Note that the FC linear network converges relatively slowly and failed to get high train/test accuracy due to the limited capability. The *ResNet* network even with a small number of layers improved the train/test accuracy significantly up to 96%/94% but at the cost of longer computation time. Carefully note that we do not mean the number of iterations but the physical time to complete the task. Actually, the runtime for the 100 iterations of the *ResNet* network took 190sec while 200 iterations of the FC linear network only took 180sec due to its simpler network architecture or less computational work.

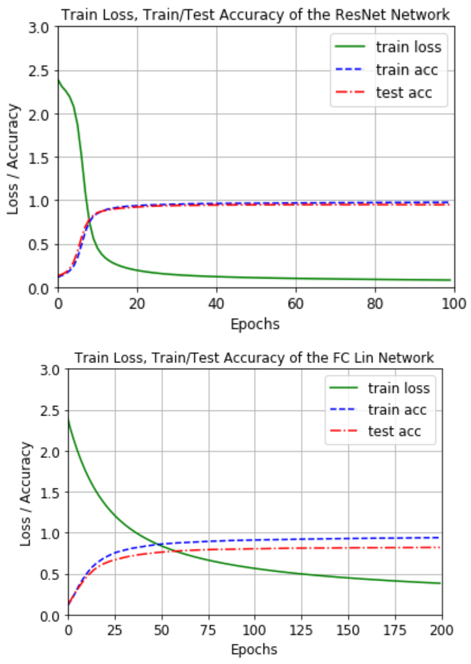


Fig. 7. The performance of a FC linear network (left) and a typical *ResNet* network (right) shown in Fig. 6.

Now, it seems our mission is clear. How can we make the network faster while keeping or further improving the test accuracy? A simple solution to increase the rate of convergence is to increase the update gain γ. However, it might disturb the overall stability of the system or make the system too sensitive to the noise. Instead, we'd like to propose another way of improving the convergence rate and the test accuracy by adding a feed-forward loop in parallel with the conventional CNN.

III. Main Result

In the previous chapters, we've found that adding a feed-forward loop in a control system can reduce the tracking error and setting a proper initial condition in an estimation system can improve the convergence rate of the estimation error. In this chapter, we will propose a new convolution neural network, *FfcNet* which can achieve both faster convergence and higher accuracy while keeping the overall system stable.

A. Framework

Let us look into the *ResNet* network shown in Fig. 1 and the block diagrams of a simple estimator with an initial condition shown in Fig. 4. We can find that both designs have a parallel loop (shown in the dotted box) which boosts the performance of the system. Now, we'd like to extend the idea from setting a proper initial condition a_0 in the simple parameter estimator in (1) ~ (2) to setting an initial *guess* for the parameter $\mathbf{a}$ in (5). The block diagram shown in Fig. 4(a) can be generalized into Fig. 8(a) through replacing the initial value a_0 for the parameter a with the initial guess

$$\bar{\boldsymbol{y}}_{\boldsymbol{k}} = \mathbf{f}_0(\mathbf{a}_0, \boldsymbol{x}_k) \tag{7}$$

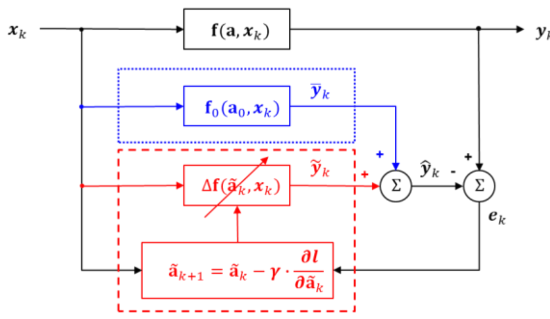
for the target system $\mathbf{f}(\mathbf{a}, \boldsymbol{x}_k)$. Then again, the block diagram in Fig. 8(a) can be transformed into the network diagram shown in Fig. 8(b), which is similar to the *ResNet* network when the identity mapping is replaced by the feed-forward function.

However, the proposed *FfcNet* is different from the original *ResNet* network. The feed-forward loop in *FfcNet* directly connects the input and the final output while the identity mapping in *ResNet* connects the output of the current layer to the input of the next layer with keeping the same dimension. Also, the main purpose of this feed-forward term $\bar{\boldsymbol{y}}_{\boldsymbol{k}}$ in (7) is to give an initial guess (of course, hope to be the best one) of the original function $\mathbf{f}(\mathbf{a}, \boldsymbol{x}_k)$. It does not need to be of the same form as the original function nor being updated during the adaptation. However, the remaining term

$$\tilde{\boldsymbol{y}}_{\boldsymbol{k}} = \Delta\mathbf{f}(\tilde{\mathbf{a}}_k, \boldsymbol{x}_k) = \mathbf{f}(\boldsymbol{a}, \boldsymbol{x}_k) - \mathbf{f}_0(\mathbf{a}_0, \boldsymbol{x}_k) \tag{8}$$

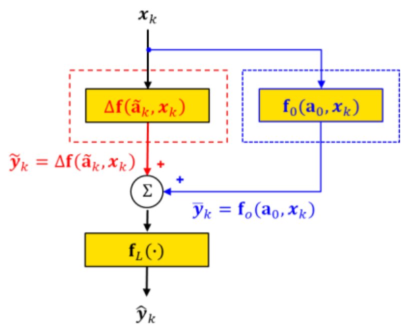
will be continuously trained to fill up the *gap* between the original function and the initial guess. We expect $\Delta\mathbf{f}(\tilde{\mathbf{a}}_k, \boldsymbol{x}_k)$ to be small and thus easy to be trained when our initial choice of $\mathbf{f}_0(\mathbf{a}_0, \boldsymbol{x}_k)$ is close enough to the original function $\mathbf{f}(\boldsymbol{a}, \boldsymbol{x}_k)$. Finally, the proposed *FfcNet* network is composed of two terms:

$$\hat{\boldsymbol{y}}_{\boldsymbol{k}} = \bar{\boldsymbol{y}}_{\boldsymbol{k}} + \tilde{\boldsymbol{y}}_{\boldsymbol{k}} = \mathbf{f}_0(\mathbf{a}_0, \boldsymbol{x}_k) + \Delta\mathbf{f}(\tilde{\mathbf{a}}_k, \boldsymbol{x}_k) \tag{9}$$



x_k
$\mathbf{f}(\mathbf{a}, x_k)$
y_k
$\mathbf{f}_0(\mathbf{a}_0, x_k)$
$\bar{y}_k$
$\Delta\mathbf{f}(\tilde{\mathbf{a}}_k, x_k)$
$\tilde{y}_k$
$\hat{y}_k$
e_k
$\tilde{\mathbf{a}}_{k+1} = \tilde{\mathbf{a}}_k - \gamma \cdot \frac{\partial l}{\partial \tilde{\mathbf{a}}_k}$

Fig. 8(a). Block diagram of the proposed *FfcNet* network: Generalize setting an initial condition into adding a feed-forward loop.



x_k
$\Delta\mathbf{f}(\tilde{\mathbf{a}}_k, x_k)$
$\mathbf{f}_0(\mathbf{a}_0, x_k)$
$\tilde{y}_k = \Delta\mathbf{f}(\tilde{\mathbf{a}}_k, x_k)$
$\bar{y}_k = \mathbf{f}_o(\mathbf{a}_0, x_k)$
$\mathbf{f}_L(\cdot)$
$\hat{y}_k$

Fig. 8(b). Network architecture of the *FfcNet* network: Replace the identity mapping in the *ResNet* with a feed-forward loop.

B. How to Select the Feed-forward Loop?

Now, it seems that choosing a proper initial guess is quite important to the success of the proposed *FfcNet* network design. As said before, we don't have to choose $\mathbf{f}_0(\mathbf{a}_0, \boldsymbol{x}_k)$ as $\mathbf{f}(\mathbf{a}_0, \boldsymbol{x}_k)$ but just close enough to $\mathbf{f}(\boldsymbol{a}, \boldsymbol{x}_k)$. In this section, we'd like to propose two simple and practical methods to choose the feed-forward loop.

Firstly, we can model the original block as a simple linear network as shown in (6). In this case, the model which we trained using a FC linear network might be a good candidate since it already demonstrated 94%/82% of train/test accuracy for the datasets. Definitely, it is not a perfect model - it might be underfitted due to too simple model - but it can represent the original block well enough to be used as the feed-forward loop $\mathbf{f}_0(\mathbf{a}_0, \boldsymbol{x}_k)$. The first candidate of the feed-forward loop can be

$$\mathbf{f}_0(\mathbf{a}_0, \boldsymbol{x}_k) = \boldsymbol{x}_k \cdot \bar{\mathbf{W}} + \bar{\mathbf{b}} \tag{10}$$

where $\bar{\mathbf{W}}$ and $\bar{\mathbf{b}}$ are the weight and bias of the FC linear network which we trained separately.

Secondly, if we've already got some measurement data, we can calculate the best estimate of the linear model using MLE (Maximum Likelihood Estimation), which can be easily found in the textbook of linear systems [10]. That is, we can calculate $\bar{\mathbf{W}}$ and $\bar{\mathbf{b}}$ from pseudo-inverse matrix of the augmented systems as follows

$$\mathbf{W}^* = (\boldsymbol{X}^{\boldsymbol{T}} \cdot \boldsymbol{X})^{-1} \cdot \boldsymbol{X}^{\boldsymbol{T}} \cdot \boldsymbol{Y} \tag{11}$$

where $\mathbf{W}^*$ is the augmented matrix including both weights $\bar{\mathbf{W}}$ and biases $\bar{\mathbf{b}}$, and $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the augmented vectors by combining $\boldsymbol{x}$ and $\boldsymbol{y}$.

Of course, we have many other options to choose as the feed-forward loop but it seems to be beyond the scopes of this paper. In this paper, we are more focusing on the structure of the network and how the sub-networks work together for a better performance.

C. Design of FfcNet Network

The final network architecture of the proposed *FfcNet* network is shown in Fig. 9. We used a similar network architecture for the regular block as the *ResNet* network shown in Fig. 6, which is a series of 2D convolution and 2D average pooling. In the *Conv2D* block, we used (3,3) for kernel, (1,1) for stride and padding. In *AvgPool2D* block, we used (2,2) for size, (1,1) for stride, and (0,0) for padding. Although it is not explicitly shown here, there is a full pre-actuation block with *BN (Batch Normalization)* and *Relu* before the 2D convolution block in the regular block. Most importantly, comparing Fig. 6 and Fig. 9, we find that the short-cut unity mapping in the *ResNet* is replaced by a feed-forward loop in *FfcNet*, which directly connects the input and the final output. We used a FC linear network of the form in (10) but all the weight and bias parameters are frozen after being specially initialized by the pre-designed simple network. Thus, this feed-forward loop is not an adaptation layer but works as a fixed mapping. As mentioned before, the FC linear network and the regular block shown in Fig. 9 are just an example and they can be chosen based on the datasets and target system. Finally, we add 2 outputs using a simple network which is also specially initialized with the identity matrix. Note that this trick is to maximize the benefit of the feed-forward loop when the regular block starts training.

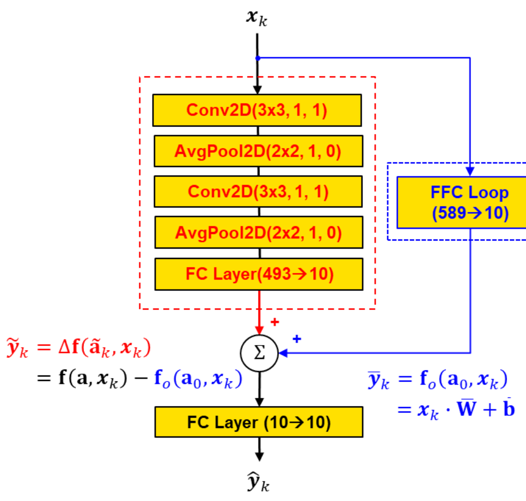


Fig. 9. Network architecture of the *FfcNet* network.

D. Further Understanding of the Feed-forward Loop

Now, we want to further understand how the feed-forward loop works, contributes to shortening the convergence time, and improves the training/test accuracy. From (4) ~ (6) along with (9), the gradient of the loss function $l(\boldsymbol{y}_k,\ \boldsymbol{x}_k)$ in (4), (5) with respect to the input value $\boldsymbol{x}_k$ is given by

$$\begin{aligned}\frac{\partial l}{\partial x_k} &= \frac{\partial l}{\partial \hat{y}_k}\cdot\frac{\partial \hat{y}_k}{\partial x_k}\\ &= -(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k)\cdot\left(\frac{\partial \mathbf{f}_0}{\partial x_k} + \frac{\partial \Delta \mathbf{f}}{\partial x_k}\right) \\ &= -\boldsymbol{e}_k\cdot\frac{\partial \mathbf{f}_0(\mathbf{a}_0, x_k)}{\partial x_k} - \boldsymbol{e}_k\cdot\frac{\partial \Delta \mathbf{f}(\tilde{\mathbf{a}}_k, x_k)}{\partial x_k}\end{aligned} \qquad (12)$$

As is analyzed in [2] along with (12), we can find that the feed-forward loop inherits all the nice features of the unity mapping in *ResNet* network. The gradient $\frac{\partial l}{\partial x_k}$ can be decomposed into two additive terms. The first term of $-\boldsymbol{e}_k \cdot \frac{\partial \mathbf{f}_0(\mathbf{a}_0, x_k)}{\partial x_k}$ propagates information directly from the error to the input layer without being affected by the layers being updated by the parameter $\tilde{\mathbf{a}}_k$. The second term of $-\boldsymbol{e}_k \cdot \frac{\partial \Delta \mathbf{f}(\tilde{\mathbf{a}}_k, x_k)}{\partial x_k}$ propagates throughout the layers which are being updated by the parameters. It is more obvious in the case that the original system is a linear system and we add a feed-forward loop with a linear function. In that case, (12) will be of the form

$$\frac{\partial l}{\partial x_k} = -\boldsymbol{e}_k\cdot\left(\bar{\mathbf{W}}^T + \tilde{\mathbf{W}}_k{}^T\right) \qquad (13)$$

where $\tilde{\mathbf{W}}_k$ is trained to estimate $\mathbf{W} - \bar{\mathbf{W}}$. It clearly shows that the first term (constant matrix) is independent of any weight layers and only the second term (with a subscript k) propagates through the weight layers.

IV. Simulation Results

In this chapter, we demonstrate the performance of the proposed *FfcNet* network through simulation results.

A. Simulation Environment

The *Gluon* library in *Apache mxnet* is used as the base framework of the machine learning throughout the simulations. More precisely, our simulation environments are modified based on the machine learning framework given in [12]. We used the update gain γ of 0.005, *SoftmaxCrossEntropyLoss*() as a loss function, the stochastic gradient descent algorithm to update the parameters, and *Xavier* initialization to initialize the weight parameters except for the special layers in the *FfcNet* network.

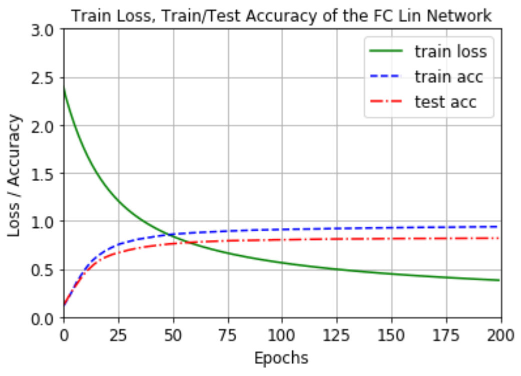
For the datasets, we used 5,890 images with 100 mini-batches for both training and test. As noise factors, we added AWGN of $\boldsymbol{N}(0, 1)$ and turned off 30% pixels randomly. Finally, the training and test images are shifted to a random direction with a PDF of $\boldsymbol{N}(\pm 0.3, 1)$ but to the opposite directions to reflect the environmental shift in datasets.

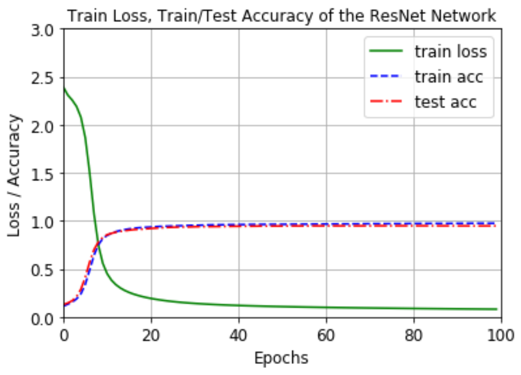
Here, we want to add some comments on the choice of the datasets. The reason why we introduced a new dataset rather than using the popular MNIST datasets or similar ones is that the proposed datasets could give us further opportunities for the quantitative analysis on the network design. Kindly note that the number of train/test datasets, the resolution of the datasets, the amount of measurement noise, structural disturbances, and even the environmental shift can be controlled. We can connect these parameters with the various performance indices in a quantitative manner and even correlate them with other design parameters such as update gain, depth of layer, etc. The results are not included in this paper due to the limited space and we will leave it as the future research topics.

TABLE I: Performance Comparison for 4 Networks

Channels	95% Rise Time	Train Loss	Train Accuracy	Test Accuracy
(1) FC Lin*	84	0.38	0.94	0.82
(2) *ResNet*	46	0.10	0.96	0.94
(3) *FfcNet*#1	22	0.02	0.99	0.96
(4) *FfcNet*#2	4	0.02	0.99	0.97

(*) The simulation results of FC linear network are based on 200 iterations due to the too slow con-vergence while others are based on 100 iterations.





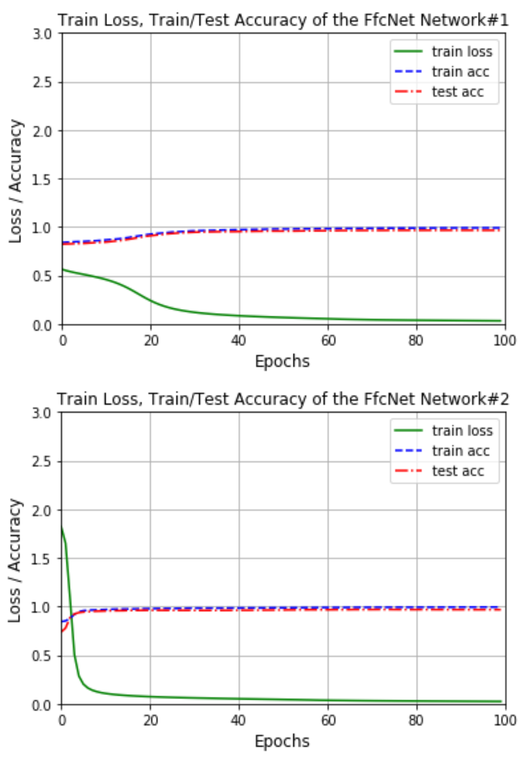


Fig. 10. Simulation results for each network designs. Figures are from top to bottom for networks in (1)~(4).

B. Performance Evaluation

Now, let's compare the performance of each network. We chose the prediction (test) accuracy after 100 iterations and the rate of the convergence (95% rise time of the train accuracy) as the key performance indices. We compared total 4 different networks including 2 results from the preliminary results; (1) FC Lin – a simple FC linear network in Fig. 6, (2) *ResNet* – a typical *ResNet* network in Fig. 6, (3) *FfcNet*#1 – a new *FfcNet* network in Fig. 9 with setting initial conditions from FC Lin, (4) *FfcNet*#2 – a new *FfcNet* network in Fig. 9 with setting initial condition from MLE in (11).

The simulation results are shown in Fig. 10 for each network design and key performance indices are summarized in Table I. As expected, all the other networks except for the simple FC linear network, showed quite good performance of around 95% test accuracy. It is demonstrated that the proposed *FfcNet* network can reduce the convergence time significantly and improve the test accuracy as well.

V. Conclusion

In this paper, we proposed a new neural network design scheme, *FfcNet* which has a faster rate of convergence with improved test accuracy. We proposed a new test platform of identifying 7-segment LED images to evaluate the performance of neural networks. The performance of the proposed network was demonstrated by the improved convergence rate and prediction accuracy for the distorted 7-segment LED images. Although a rigorous mathematical derivation is not given here, an analogy between the feed-forward loop in conventional control schemes and the residual network in *ResNet* network was illustrated conceptually for the better understanding of *FfcNet* network.

The purpose of this paper is to propose a new conceptual design scheme and we hope this preliminary study can seed the future studies on the neural network design such as extending this concept to other more complex systems or other areas beyond image recognition. Another interesting research topic would be the connection with the fine-tuning since it also uses similar schemes of using the pre-trained data to train a new set of data [13]-[15]. We'd like to leave this along with more rigorous mathematical derivation of the convergence for the future study topics.

Appendix A

In Appendix A, we give symbols and notation used throughout the paper.

Symbol	Description
$\mathbf{x}^i$	Input vector in $\mathbb{R}^{589}$ obtained by serializing the bit image of '*i*' in 7 segment LED.
$\mathbf{y}^i$	Output vector in $\mathbb{R}^{10}$ representing the figure '*i*', e.g., $\mathbf{y}^1 = [0,1,0,0,0,0,0,0,0,0]$.
$\mathbf{h}$	A function mapping $\mathbf{x}^i \in \mathbb{R}^{589}$ to $\mathbf{y}^i \in \mathbb{R}^{10}$, i=0,1,2…,9 with a parameter $\mathbf{d} \in \mathbb{R}^{D}$.
$\mathbf{X}$	Augmented input vector in $\mathbb{R}^{10\times589}$ obtained by stacking $\mathbf{x}^i$ in sequence vertically.
$\mathbf{Y}$	Augmented output vector in $\mathbb{R}^{10\times10}$ obtained by stacking $\mathbf{y}^i$ in sequence vertically.
$\mathbf{f}$	A function mapping $\mathbf{X} \in \mathbb{R}^{10\times589}$ to $\mathbf{Y} \in \mathbb{R}^{10\times10}$ with a parameter $\boldsymbol{a} \in \mathbb{R}^{A}$.
$\mathbf{a}$	A vector in $\mathbb{R}^{A}$ which parameterizes the function $\mathbf{f}$ in terms of the augmented input vector $\mathbf{X}$, that is, $\mathbf{Y} = \mathbf{f}(\mathbf{a},\mathbf{X})$.
$\hat{\mathbf{a}}$	Estimate of the parameter vector $\mathbf{a}$ in $\mathbb{R}^{599\times10}$
$\widehat{\mathbf{W}}$	Weight matrix for the linear network with a dimension of $\mathbb{R}^{589\times10}$
$\hat{\mathbf{b}}$	Bias matrix for the linear network with a dimension of $\mathbb{R}^{10\times10}$
$\boldsymbol{x}_k$	k^{th} mini-batch of the augmented input vector $\mathbf{X}$ in $\mathbb{R}^{M\times589}$ with a mini-batch size of M.
$\boldsymbol{y}_k$	k^{th} mini-batch of the augmented output vector $\mathbf{Y}$ in $\mathbb{R}^{M\times10}$ with a mini-batch size of M.
$\mathbf{a}_0$	An initial estimate of the parameter vector $\mathbf{a} \in \mathbb{R}^{A}$ for the function $\mathbf{f}$.
$\mathbf{f}_0$	An estimate of the function $\mathbf{f}$: $\mathbb{R}^{M\times589} \to \mathbb{R}^{M\times10}$.
$\bar{\boldsymbol{y}}_k$	k^{th} mini-batch of the augmented output vector $\mathbf{Y}$ in $\mathbb{R}^{M\times10}$ for the input vector $\boldsymbol{x}_k$, which is obtained from the known function $\mathbf{f}_0(\mathbf{a}_0,\boldsymbol{x}_k)$.
$\Delta\mathbf{f}$	The difference between the function $\mathbf{f}$ and $\mathbf{f}_0$.
$\tilde{\mathbf{a}}_k$	Parameter vector in $\mathbb{R}^{A}$ of the function $\Delta\mathbf{f}$ for the k^{th} mini-batch.
$\tilde{\boldsymbol{y}}_k$	k^{th} mini-batch of the augmented output vector $\mathbf{Y}$ in $\mathbb{R}^{M\times10}$ for the input vector $\boldsymbol{x}_k$, which is obtained from the function $\Delta\mathbf{f}(\tilde{\mathbf{a}}_k,\boldsymbol{x}_k)$.
$\hat{\boldsymbol{y}}_k$	The final estimate of $\boldsymbol{y}_k$ for the input vector $\boldsymbol{x}_k$.
$\boldsymbol{e}_k$	Estimation error $\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k$ for the k^{th} mini-batch.

Conflict of Interest

The author declares no conflict of interest.

Acknowledgment

A special thanks to the professors in the Department of Mechanical Convergence Eng., Gyeongsang Nat'l Univ., Korea during the collaboration for the research activities.

References

[1] K. He, X. Zhang, R. Shaoqing, and J. Sun, "Deep residual learning for image recognition," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[2] K. He, X. Zhang, R. Shaoqing, and J. Sun, "Identity mappings in deep residual networks," in *Proc. European Conference on Computer Vision*, 2016, pp. 630–645.

[3] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q., "Densely connected convolutional networks," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[4] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," in *Proc. the British Machine Vision Conference*, pp. 87.1-87.12, 2016.

[5] D. Y. Yeo, M. S. Kim, and J. H. Bae,"Adversarial optimization-based knowledge transfer of layer-wise dense flow for image classification," *Applied Science*, vol. 11, pp. 3720, 2021.

[6] K. W. Jin and E. C. Lee, " Improving the performance of frequently used Korean handwritten character verification based on artificial intelligence through multimodal fusion," *Applied Science*, vol. 11, pp. 8413, 2021.

[7] H. Kim, S. Kim, J. Back, H. Shim and J. Seo, "Design of stable parallel feed forward compensator and its application to synchronization problem," *Automatica*, vol. 64, no. 2, pp. 208-216, 2016.

[8] N. G. Kim, "A learning approach to asymptotic output tracking in non-minimum phase non-linear system," Ph. D dissertation, Dept. EE, Seoul Nat'l Univ., Korea, 2000.

[9] Y. H. Kim and I. J. Ha, "Asymptotic State Tracking in a Class of Nonlinear Systems via Learning-Based Inversion," *IEEE Transactions on Automatic Control. vol. 45*, pp. 2011–2017, 2000.

[10] C. Chen, *Linear System Theory and Design*, 3rd ed., Oxford University Press Inc., 2012.

[11] Y. Zhang, B. Chu, and Z Shu, "A preliminary study on the relationship between iterative learning control and reinforcement learning," in *Proc. 13th IFAC Workshop on Adaptive and Learning Control Systems ALCOS,* vol. 52. UK, 2019, pp. 314–319.

[12] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into Deep Learning*. [Online]. Available: www.http://d2l.ai/

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. the IEEE Conference on Computer Vision And Pattern Recognition (CVPR),* June 2014, pp. 580-587.

[14] H. C. Jung, S. H. Lee, J. H. Yim, S. J. Park, and J.M. Kim. "Joint fine-tuning in deep neural networks for facial expression recognition," in *Proc. the IEEE International Conference on Computer Vision*, 2015, pp. 2983-2991.

[15] Z. Li and D. Hoiem, "Learning without forgetting," in *Proc. European Conference on Computer Vision*, Springer, pp. 614–629, 2016.

Nam Guk Kim was born in South Korea in 1970. He received the BS, MS and Ph. D degrees in EE from Seoul Nat'l Univ., Seoul, Korea, in 1993, 1995 and 2000, respectively. His major research interests in the univ. were iterative learning and other control algorithm development.

Since 2003, he had worked for Samsung Electronics as a team leader of the storage product. After moving to Seagate Technology in 2011, he continued his career in storage industry as a Sr. Eng. director based in Korea and Singapore. Now, he is working as a research fellow in Cybernetics Imaging Systems, Korea.

His recent interest is the ML/AI, especially for the image recognition and RL. He won 3 CEO awards from all the above 3 companies for his outstanding contributions to the technology development and biz.