International Journal of Machine Learning and Cybernetics (2021) 12:763–781
https://doi.org/10.1007/s13042-020-01201-8

ORIGINAL ARTICLE

MS-NET: modular selective network

Round robin based modular neural network architecture with limited redundancy

Intisar Md Chowdhury[1] · Kai Su[1] · Qiangfu Zhao[1]

Received: 29 March 2020 / Accepted: 17 September 2020 / Published online: 30 September 2020

Abstract
We propose a modular architecture of Deep Neural Network (DNN) for multi-class classification task. The architecture consists of two parts, a router network and a set of expert networks. In this architecture, for a C-class classification problem, we have exactly C experts. The backbone network for these experts and the router are built with simple and identical DNN architecture. For each class, the modular network has a certain number ρ of expert networks specializing in that particular class, where ρ is called the redundancy rate in this study. We demonstrate that ρ plays a vital role in the performance of the network. Although these experts are light weight and weak learners alone, together they match the performance of more complex DNNs. We train the network in two phase wherein, first the router is trained on the whole set of training data followed by training each expert network enforced by a new stochastic objective function that facilitates alternative training on a small subset of expert data and the whole set of data. This alternative training provides an additional form of regularization and avoids over-fitting the expert network on subset data. During the testing phase, the router dynamically selects a fixed number of experts for further evaluation of the input datum. The modular nature and low parameter requirement of the network makes it very suitable in distributed and low computational environments. Extensive empirical study and theoretical analysis on CIFAR-10, CIFAR-100 and F-MNIST substantiate the effectiveness and efficiency of our proposed modular network.

Keywords Modular neural networks · Deep learning · Knowledge-distillation · Multi-class classification · Image classification

1 Introduction

Deep Neural Networks (DNNs) in the last two decades have shown it's superiority in the field of visual object recognition [26, 62, 64]; image segmentation [5, 9, 63, 76]; speech recognition and translation [3, 29]; natural language processing [11, 68]; reinforcement learning [43, 56, 57]; bio informatics [63]; educations [38, 73]; and so on. Despite their simple layered structures of neurons and connections, they have outperformed other machine learning models [74]. This superiority has been achieved due to its ability of complex non-linear mapping from input to output, automated rich and discriminate features learning as opposed to hand-engineered low-level features such as GABOR features [41], local binary patterns [2], SIFT [54] and so on. With the passage of time, we can notice that not only the performance is levitating dramatically, also networks are getting deeper [14, 31, 69] and wider [77]. As a result, these finer networks are lacking few important and desirable properties such as interpretability or comprehensibility, practical applicability in low computational devices and so on. In addition, problems such as catastrophic forgetting with the arrival of new data [28], lack of memory efficiency, have also started to arise. Fortunately, various novel approaches have been proposed to mitigate a few of these shortcomings. Recent notable approaches include knowledge distillation from the cumbersome models to smaller models [33]; compression of knowledge from ensemble to a single model [8]; pruning of neural

✉ Intisar Md Chowdhury
d8211106@u-aizu.ac.jp

Kai Su
m5232109@u-aizu.ac.jp

Qiangfu Zhao
qf-zhao@u-aizu.ac.jp

1 System Intelligence Laboratory, The University of Aizu, Aizu-Wakamatsu 965-8580, Japan

networks [48, 51, 58, 81, 82]; efficient Neural Architecture Search (NAS) [61, 82, 83]; modular neural network design [4, 30, 40, 42, 74]; and so on. There have been also significant advances in efficient hardware design architectures for DNNs. Intel Corporation has developed a neural computation stick powered by the Vision Processing Unit (VPU) which can accelerate the inference phase of complex DNN on a low computational device. Google has also recently developed small edge Tensor Processing Unit (TPU) for high-performance machine learning inference. These small ASIC devices for DNN can easily execute deep Convolutional Neural Networks (CNN), which make it one of the best alternatives for cloud-based service. Unfortunately, when it comes to state-of-the-art networks, these ASIC devices still face performance bottle-neck when executed in real-time scenarios. Thus, it is necessary to devote time and research on mitigating the above shortcomings of DNN.

In this paper, we propose a novel modular neural network framework for multi-class classification, which is inherently simple and easy to implement. The key idea is to leverage a fixed number of experts, each with parameters as few as possible during the inference phase, while maintaining accuracy comparable to relatively complex and monolithic state-of-the-art DNNs. The proposed framework has a close resemblance to the model of the human brain depicted by Minsky in [55], where he described the human brain as a collection of specialist agents interconnected by nerve-bundles. Quoting from [55] *We're born with proto-specialists involved with hunger, laughter, fear and anger, sleep and sexual activity- and surely many other functions no one has discovered yet- each based upon a somewhat different architecture and mode of operation*. Analogous to this brain model, our framework consists of a countable set of expert agents and a router agent. In this literature, we term the expert agents as the expert networks and router agent as the router network. Each of these expert networks is expert on a specific subtask and their computation take place independently. Although they are not superior individually for a whole set of tasks, they outperform each individual network when they execute collectively. The router network moderates the execution of these expert networks. The concept of the modular neural network itself is not new. The key concept of modular connectionist goes back to the mid-1980s in [40]. A number of contributions such as [30, 40, 50, 72] have approached the task of speech recognition using the modular connectionist theory. A majority of the proposed modular architectures are equipped with a gating network (analogous to our router network) and a set of expert networks. Despite the popularity of modular connectionist models during the 80s, the modular approach in recent DNNs (such as CNN, Recurrent Neural Network(RNN) and so on) era is relatively sparse, until recently Hinton and Vinyals have introduced the novel concept of knowledge distillation in neural network [33]. Our proposed modular neural network framework which we termed as the *MS-Net* has a close resemblance to [4, 24, 33, 60] literature in the following key points: i) We divide the dataset into a number of subsets/subtasks/concepts. Afterward, we train a fixed number of expert neural networks on each of these subsets ii) The router module navigates us to those expert networks for further re-evaluation.

However, in addition to the above points and our previous work [39][1], the novelty of our contributions to this research can be summarized as follows:

1. We propose a simple data partitioning technique for the modular neural network based on the Round Robin method. This technique enables decomposition of the dataset into C subsets of class indices, where C is the total number of classes available in the dataset.
2. We provide a detailed theoretical and empirical study on effect of redundancy variable ρ on the complexity and performance of MS-Net.
3. We theoretically demonstrate that the proposed MS-Net requires no more than C expert networks to effectively specialize on the corresponding C subsets of classes.
4. We propose a new stochastic objective function to train the expert networks. The proposed objective function is composed of two terms, wherein the first one facilitates optimizing the expert networks on data from its corresponding subset classes, and the second term optimizes networks on data from the whole set of classes.

2 Outline

We arrange the paper in the following order.

1. Sect. 3: Related works on modular neural networks and machine learning models, ensemble learning and so on.
2. Sect. 4: An overview of the MS-Net architecture.
3. Sect. 5: Detailed discussion on Round-Robin based dataset partition.
4. Sect. 6: Training procedure of expert networks, including algorithm.
5. Sect. 7: Inference phase on MS-Net, including algorithm.
6. Sect. 8: Detailed discussion about the datasets and experiment settings.
7. Sect. 9: Empirical analysis.

[1] Part of contributions in this paper have also appeared in previous literature [39] titled *Selective Modular Neural Network*, where (i) ρ and n were limited to only 2 (ii) simple cross-entropy loss function was used to train the experts (iii) preliminary results of few public datasets.

8. Sect. 10: Guidance for optimal hyper-parameters selection for the network.
9. Sect.11 Effects of knowledge-distillation on MS-Net.
10. Sect.12 Discussion on results and comparison to state-of-the-art DNNs.
11. Sect. 13: Conclusion and possible future works.

3 Related work

Modular architectures have been famous in neural networks or connectionist models for a long time. In addition to that, modularity has also been widely implemented in other traditional machine learning models. This approach has not only boosted the performance of these learning models, but also introduced virtues such as interpret-ability, training efficiency, distributed computation, reduction of parameters and so on [4]. In this section, we provide an overview on the neural networks and other machine learning models which exhibit modular behaviour.

Class binarization (CB) is one of the most well-known method in the modular framework. It can be considered as a special case of ensemble learning, where each binary module is assigned to learn or distinguish a single concept or class from the rest. Among different CB techniques, *ONE VS ALL* (un-ordered binarization) is the most commonly practiced technique in neural network [4], support vector machine [12], due to its computational efficiency and performance boost. The technique first appeared in the literature [10]. The method constructs C binary classifiers in total, where C is the total number of classes. Despite its simplicity, the method suffers from class imbalance, since the number of positive instances is smaller compared to the negative instances for each binary classifier. In addition, an ordered variant of the mentioned CB technique requires only $C-1$ classifiers. However, the class imbalance short-coming was later resolved by the method *ONE VS ONE* which appeared in the literature *Separate-and-Conquer Rule Learning* [21]. A more systematic method for generating binary classifiers which is known as the *Round Robin learning* was introduced by the same author in the literature [22–24]. Due to its systematic method of creating binary classifier, it carries more interpret-ability. The method has demonstrated that a total of $C\,(C-1)/2$ classifiers are constructed using the Round-Robin method. Each of these classifiers is a pair-wise-classifier, expert on two specific classes or concepts. Thus, the issue of class imbalance no longer prevails. In addition to that, authors have shown that this approach requires relatively fewer amount of data during training as oppose to *ONE VS ALL* method. However, during the inference phase all $C\,(C-1)/2$ classifiers require evaluation. With a view to resolving this computational issue, a relatively recent literature [60] proposed an efficient prediction algorithm for these ensembles, where pair-wise classifiers can be dynamically chosen without any drop in accuracy.

Knowledge distillation (KD) is a recent and very popular method for compression of complex and cumbersome DNNs. The method was first proposed by Hinton et al. [33]. This method is now widely implemented in deep learning research and industrial applications. Studies such as, [20, 78] have shown that, KD not only allows compression but also enables a relatively smaller student model to outperform its teacher model. The key idea is to train a student network to mimic the output features or the class probability distributions of the teacher network. The literature's [33] contribution was not only limited to KD, the authors have also proposed a modular network framework that has a very close resemblance to our proposed framework. The model consists of two main parts, a generalist network and a set of independent expert networks. Each of these expert networks is a simple CNN, which is trained on data that are often confused and misclassified by the generalist network. Thus, each individual expert is classifier of type *CONFUSABLE SUBSET VS ALL*, where one part is the CONFUSABLE set of task and the rest ends up with single DUSTBIN class. This notion implies that the generalist model needs to be evaluated first to obtain those CONFUSABLE set of classes. In-order to (i) retain knowledge about the non-expert classes (ii) avoid over-fitting and (iii) solve the class imbalance problem the author initialized the expert networks with the weights of generalist network. The literature has shown that, as the number of expert networks increases, the accuracy increases proportionally. However, there have been no concrete indication and estimation on the number of experts covering those CONFUSABLE set of classes. In addition, the literature states that there can be situation where there are no expert networks covering a certain set of classes (since the generalist network is already confident on its prediction for those certain set of classes).

Recent research [80] titled *Deep Mutual Learning (DML)* which consists of cohort of student models resembles modular behavior. The DML enables a number of student models to mutually learn from one another by minimizing the Kullback Leibler (KL) Divergence between their predictions, which is a special case of KD [33]. The experiments have shown that the number of student networks in cohort during training can be extended to more than two. Moreover, empirical results show that, multiple student neural networks trained by the mutual learning out-perform single model network trained independently. This learning process has also shown to outperform the KD method.

Other notable recent research contribution on modular neural network includes the famous Generative Adversarial Network (GAN)[27], where two networks, discriminator and the generator network co-operate and compete against each other. There are also different variants of GAN which comprise of more than two networks [45]. Research [16] proposed modular like architecture that is build upon the existing state-of-the-art neural networks. In the literature, they re-configure the model parameters into several parallel branches where each branch is a stand-alone neural network. They have demonstrated that, the average of the log-probabilities of multiple parallel branches give better representation as opposed to the single independent branch.

In this paper, our modular neural network framework has a very close similarity to the literature [4, 24, 33], such as presence of gating network and expert networks. But in contrast to *ONE VS ONE* and *ONE VS ALL*, our expert networks are not limited to binary classifiers. We introduce a simple Round-Robin based systematic data partition technique which enables us to train each expert on subset of multiple classes. A contrast to note that, unlike ensemble learning method such as well known AdaBoost [18],Bagging [6], Random Forest [7], Gradient boosting [19] and so on which requires the collective wisdom of all available classifiers, our network does not require to run all the neural network models during inference. The novelty in our proposed framework is that, the router of the MS-Net extensively reduces the number of expert network evaluation during the inference phase. Since the partition of dataset is systematic, i) it gives us prior knowledge on which experts are specialist on which subsets, which also facilitates us to dynamically chose specific number of expert networks during inference. ii) it guarantees presence of multiple expert networks for a single concept or class, thus we have a certain degree of fault tolerance in case other experts or the router network fail to correctly classify the data.

4 Proposed network architecture

The network has two main modules, a router network, and a pool of expert networks. In the expert network pool there are C expert networks, where C is the total number of classes available in a given dataset. A simplified image of our modular framework is shown in Fig. 1. The expert networks and router network have the same network architecture. A very important issue is the size of the network. In our experiment, a cumbersome and computationally expensive network is not desirable. On the contrary, we also do not want the network to face performance bottle-neck due to simple architecture. There are many remarkable literature relating to the compact, efficient and accurate Neural Architecture Search (NAS) [17, 83] in recent times, but this topic is out of scope for this paper. However, the choice of architectures of any network are dependent on the complexity of the dataset. Considering the computational issue and memory efficiency, we chose *ResNet-20* [31] as the initial backbone network, which is of one the most minimalist and light weight network to our knowledge. We leverage the *Resnet-20* as the backbone of MS-Net to find the optimal hyper-parameters

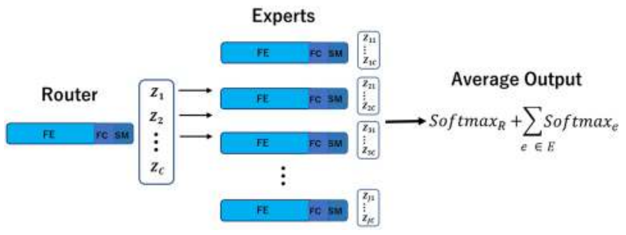


Fig. 1 Test Phase version of MS-Net. FE and FC depict Feature Extractor and Fully Connected layer of neural network respectively. E is the set of all experts dynamically selected by the router network R. The first block represents the router network which dynamically selects the expert networks based on its softmax (SM) confidence. The second part is the pool of expert networks further re-evaluating the router's *top-n* most likely predictions. Finally, the network aggregates the soft-max scores of router and selected experts

such as, the value for β, *top-n* and so on. After we obtain the optimal hyper-parameters through extensive empirical study with *ResNet-20* we train other complex DNNs which are, *GoogleNet* [69] and *MobileNet* [35] as the backbone network of MS-Net.

5 Round robin based data-set partition with sliding window

In this research, the redundancy rate plays a vital role in the performance of the framework. We denote the redundancy rate as ρ. The variable ρ has two main interpretations. First, ρ is the size of each subset of class indices. Second, each class index appears exactly in ρ subsets of class indices. In any sense, when ρ is larger more expert networks will get the chance to see the training data from any particular class. This is the reason why we called ρ redundancy rate.

In order to prove the above two points let us introduce several notations. First, we use $\mathcal{D} = \{d_i | i = 1, .., N\}$ to denote the whole training data set, where N is the total number of training data; and $\mathcal{T} = \{t_i | i = 1, .., N\}$ to denote the set of teacher signals, where t_i is associated with d_i for $i = 1, 2, ..., N$. To partition the subsets for training the expert networks, we leverage a sliding window of length k and stride s. Refer to Fig. 2 for a graphical overview of dataset partition. In this figure, we arrange the indices of all classes in a ring-shaped manner. The sliding window length k is a positive integer less than C, which is the total number of classes Table 1. The redundancy rate ρ depends directly on k. Each time when we shift the sliding window with a stride s over the ring in a Round-Robin fashion (clockwise), we obtain a subset sub_i which contains k class indices. We use $S = \{sub_1, sub_2, ...\}$ to denote the set of all class index sets so far obtained. We can prove that, for any value of k and with stride $s = 1$, the cardinality of S is always equal to the total number of classes C. Since

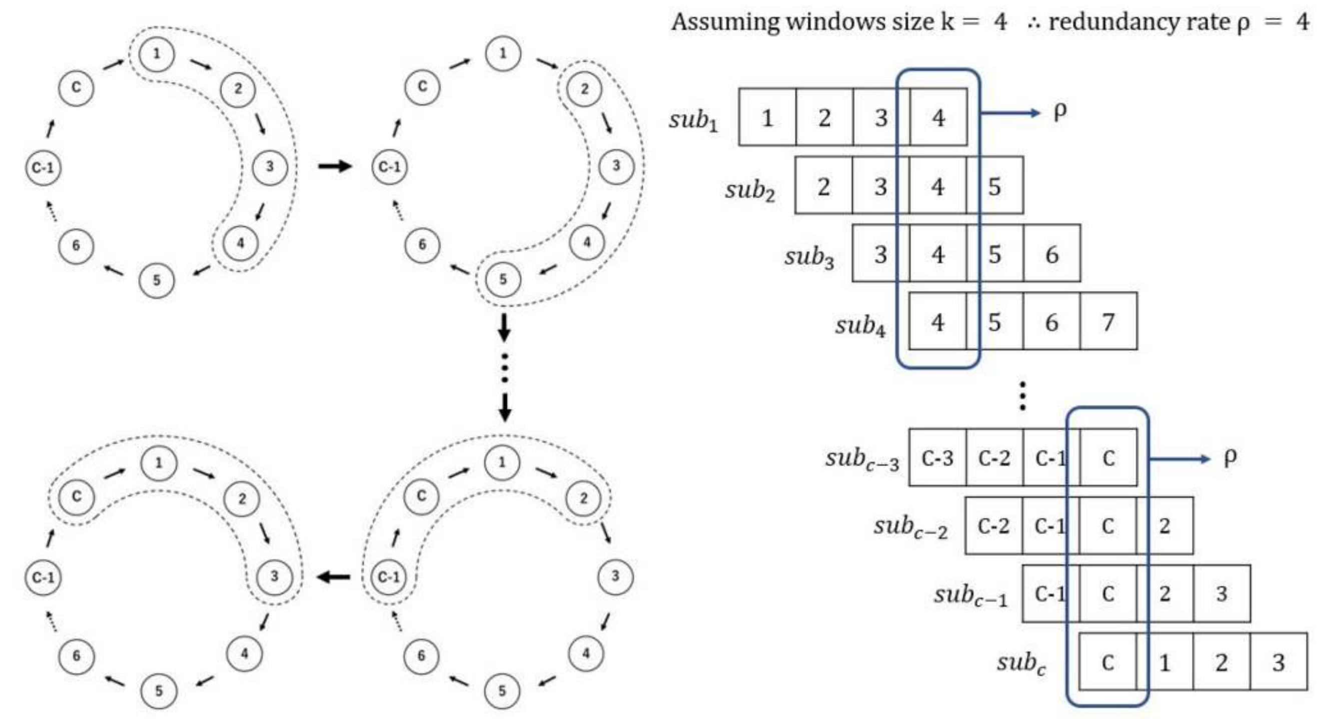


Fig. 2 Round Robin partition of the dataset. The left image depicts the sliding of the window over the classes. In each sliding operation, we have a subset. The sliding operation continues for C times. The right image illustrates the effect of size of the sliding window on the redundancy variable ρ. In the image we fix sliding window size to 4, hence we have each class occurring in exactly four subsets

Table 1 Backbone networks

Network	# of parameters (M)	MACs (G)
ResNet-20 [32]	0.269	0.041
MobileNet [35]	2.36	0.33
GoogleNet [69]	6.20	16.04

we have C target classes, and if we can prove $|S| = C$, we can conclude that our framework requires no more than C experts.

Lemma 1 *With stride $s = 1$ and for any value of k, the cardinality of S is always equivalent to the total number of class C available in the data-set.*

Proof If k is the length of sliding window of any length, by the convolution arithmetic [15] we can state that the number of class index sets in S as:

$$|S| = \frac{(C-k)}{s} + k \tag{1}$$

Since we are using the Round Robin rotation, the later term k is added instead of 1. As, $s = 1$, Eq. (1) can be re-written as:

$$\begin{aligned} |S| &= C - k + k \\ &= C \end{aligned} \tag{2}$$

Thus, with stride 1, the total number of class index sets or the number of expert networks is always equal to the number of classes. □

Lemma 2 *If the length of sliding window is k and stride $s = 1$, the index for each class occurs exactly in k class index sets or in other words, we have exactly k experts related to each class.*

This implies that the redundancy rate ρ is determined by the sliding window size k. This phenomenon also suggests that, k determines the fault tolerance of the proposed MS-Net. As the value of k increases, we have more experts for each particular class (Note that, the total number of experts remains constant i.e. C). On the contrary, as we decrease k, the redundancy rate or the number of experts specializing on that particular class decreases.

Proof Let us assume that the sliding window length is k, where $k < C$. After the $n-th$ $(n = 0, 1, ..., C-1)$ sliding operation, we obtain the following class index sets.

$$sub_{n+1} = \{n \bmod C + 1, ..., (n + k - 1) \bmod C + 1\}.$$

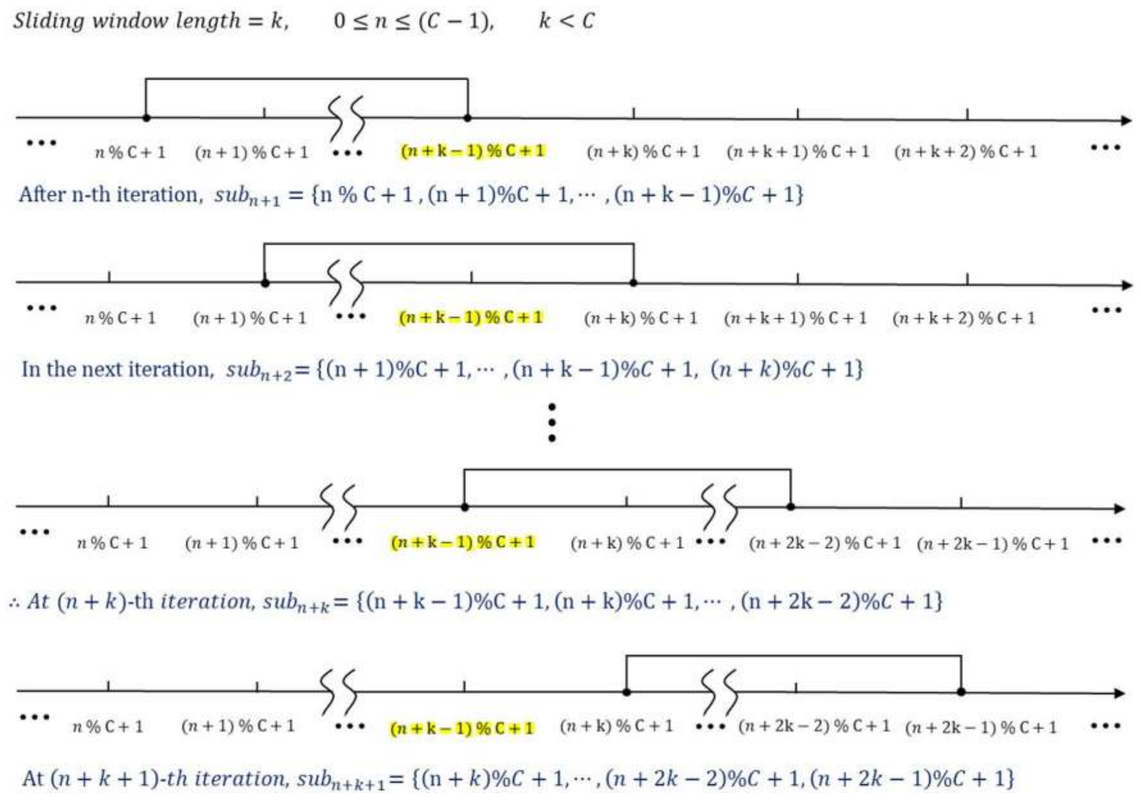

Sliding window length = k, 0 ≤ n ≤ (C − 1), k < C
n % C + 1
(n + 1) % C + 1
(n + k − 1) % C + 1
(n + k) % C + 1
(n + k + 1) % C + 1
(n + k + 2) % C + 1
After n-th iteration, sub_{n+1} = {n % C + 1 , (n + 1)%C + 1, ⋯ , (n + k − 1)%C + 1}
In the next iteration, sub_{n+2} = {(n + 1)%C + 1, ⋯ , (n + k − 1)%C + 1, (n + k)%C + 1}
(n + 2k − 2) % C + 1
(n + 2k − 1) % C + 1
∴ At (n + k)-th iteration, sub_{n+k} = {(n + k − 1)%C + 1, (n + k)%C + 1, ⋯ , (n + 2k − 2)%C + 1}
At (n + k + 1)-th iteration, sub_{n+k+1} = {(n + k)%C + 1, ⋯ , (n + 2k − 2)%C + 1, (n + 2k − 1)%C + 1}

Fig. 3 Illustration of Lemma 2. The figure depicts that, with a sliding window length of k, each (in this figure, the highlighted class index $(n+k-1) \bmod C+1$ is shown to occur k times.) class index occurs in exactly k subsets. This also suggests that for each class in the dataset MS-Net has k expert networks

According to the definitions of the sliding window and the class index sets, $|sub_{n+1}| = k$. Since we are performing Round Robin rotation, we use the modulus operator for indices of each class. □

Without loss of generality, we show that the index $(n+k-1) \bmod C+1$ exists in exactly k class index sets. During the Round-Robin partition, we shift each element of sub_{n+1} to the left of the sliding window with stride $s = 1$ as depicted in Figs. 3 and 4. Thus, in each sliding operation we introduce a new class index to the right of the sliding window, which in the case of sub_{n+1} is $(n+k-1) \bmod C+1$. In the same way, for the next sliding operation, we have,

$$sub_{n+2} = \{(n+1) \bmod C+1, ..., \\ (n+k) \bmod C+1\}.$$

As we can observe in sub_{n+2}, the class index $n \bmod C+1$ ceases to exist and a new index $(n+k) \bmod C+1$ arrives in the right most position. In addition, the index $(n+k-1) \bmod C+1$ shifts one position to the left. After the $n+k-1$-th sliding operation, we have,

$$sub_{n+k} = \{(n+k-1) \bmod C+1, ..., \\ (n+2k-2) \bmod C+1\}.$$

The index $(n+k-1) \bmod C+1$ is now at the left most position. After the $n+k-th$ sliding operation, $(n+k-1) \bmod C+1$ will no longer exist in the subset sub_{n+k+1} because

$$sub_{n+k+1} = \{(n+k) \bmod C+1, ..., \\ (n+2k-1) \bmod C+1\}.$$

It is clear from above equation, $(n+k-1) \bmod C+1 \notin sub_{n+k+1}$ since after the $n+k-th$

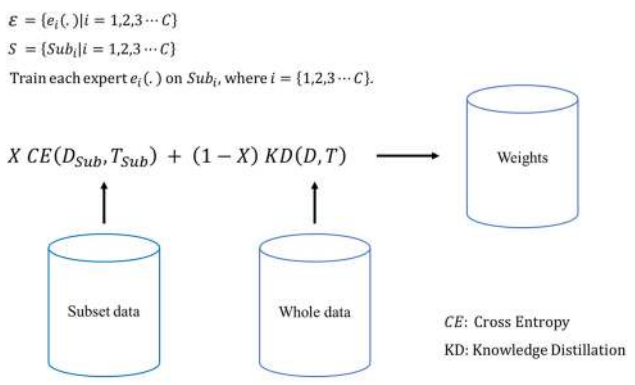

ℰ = {e_i(.)|i = 1,2,3 ⋯ C}
S = {Sub_i|i = 1,2,3 ⋯ C}
Train each expert e_i(.) on Sub_i, where i = {1,2,3 ⋯ C}.
X CE(D_Sub, T_Sub) + (1 − X) KD(D, T)
Weights
Subset data
Whole data
CE: Cross Entropy
KD: Knowledge Distillation

Fig. 4 Illustration of the training phase. This figure is the pictorial version of the Training phase section

sliding operation, the class index $(n+k-1) \bmod C+1$ slides out of the window. Thus, $(n+k-1) \bmod C+1$ occurs in $\{sub_{n+1}, sub_{n+2}, ..., sub_{n+k}\}$ or exactly in $(n+k)-(n+1)+1 = k-1+1 = k$ class index sets, which also concludes we have k experts for the class $(n+k-1) \bmod C+1$.

6 Training phase

We perform the training procedure in two steps. First, we train the router network on whole dataset. Second, we train C experts on the subsets which can be constructed based on the class index sets obtained in Round-Robin fashion depicted in Sect. 5. We denote the router network as $y = R(.) : \mathcal{D} \to \mathcal{T}$, where $\mathcal{D}$ and $\mathcal{T}$ are the dataset and the corresponding label set, respectively. The output of the router network is the softmax defined in Eq. (3), where we obtain the probabilities $q_1, ..., q_C$ for all C classes. Here, $z_1, ..., z_C$ are the logit scores for the corresponding classes.

$$q_i = \frac{exp(z_i)}{\sum_j exp(z_j)}. \tag{3}$$

For our modular network framework, the *top*-1 score does not require to be strictly accurate. Since it is obvious that, the likeliness of the correct answer to be in *top-n* (as n increases) is higher than *top*-1, we take into consideration the *top-n* most probable answers. The role of the expert networks comes into play in this situation, where a set of experts further re-evaluate the router's *top-n* predictions. Thus, the accuracy of the experts have a significant effect on the MS-Net performance. Let us assume, we have a set of expert neural networks $\mathcal{E} = \{e(.)_1, ..., e(.)_C\}$. In order to ensure these experts effectively specialize on the subsets, we formulate a stochastic objective function which we depict in the Eq. (4). The objective function optimizes each of the expert network on its corresponding subset data $\{\mathcal{D}_{sub_i}, \mathcal{T}_{sub_i}\}$ using cross entropy loss function, where $\mathcal{D}_{sub_i} = \{d_j \in \mathcal{D} | t_j \in sub_i \land 1 \le j \le N\}$ and $\mathcal{T}_{sub_i} = \{t_j \in \mathcal{T} | t_j \in sub_i \land 1 \le j \le N\}$ and on the whole set of data $\{\mathcal{D}, \mathcal{T}\}$ using KD function, alternatively.

The knowledge is distilled from the router network. Thus the router is the teacher model. The alteration between two the loss terms in Eq. 4 is controlled by the Bernoulli random variable $\mathcal{X}$ with the probability

$$Prob(\mathcal{X} = 1) = \beta.$$

The stochastic nature of the objective function for a certain range of β provides (i) balanced training of networks and (ii) better regularization. Again, the cardinality of each class index set sub_i is determined by the redundancy variable ρ. In our experiment we demonstrate the effectiveness and performance of the framework for $\rho = 2, 3$ and 4. We stress that, during the inference phase, as we increase ρ the number of expert network evaluation increases linearly. Due to the stochastic training of expert networks on whole dataset using KD (the second part of Eq. (4)), these networks are no longer limited to its corresponding subset data. Rather, each of the network is an expert on their own subset classes, in the meantime has certain generalization ability on the data of other classes.

In Eq. (4), the first term optimizes the expert network $e_i()$ on the classes defined by sub_i, weighted by Bernoulli random variable $\mathcal{X}$ which takes a value of 1 based on the probability β. The later term of Eq. (4) optimizes the network on the whole dataset weighted by $1-X$ based on probability $1-\beta$. Thus, β controls the trade-off between two loss terms in Eq. (4).

$$Loss_{e_i}^{kd} = \mathcal{X}\, l_{sub_i} + (1-\mathcal{X})\, KD_{all} \tag{4}$$

where,

$$l_{sub_i} = -\sum_{l,t \in (\mathcal{D}_{sub_i}, \mathcal{T}_{sub_i})}^{N} \sum_{m \in sub_i} \delta(t,m) \log(P_{e_i}^m(l)) \tag{5}$$

and

$$\begin{aligned} KD_{all} = -\alpha \sum_{j=1}^{N} \sum_{k=1}^{C} P_{e_i}(d_j)\ \log \frac{P_{e_i}(d_j)}{P_R(d_j)} \\ -(1-\alpha) \sum_{j=1}^{N} \sum_{k=1}^{C} \delta(t_j,k) \log(P_{e_i}^k(d_j)) \end{aligned} \tag{6}$$

In the above equations, $\delta(t,k)$ is the Kronecker delta function defined by

$$\delta(t,k) = \begin{cases} 1, & t = k, \\ 0, & t \ne k \end{cases}$$

The hyper-parameter α controls the trade-off between the KD and cross-entropy loss, where $0 < \alpha < 1$. The value of α during training depends on the performance of the teacher network. A high α value puts more weight on the distilled knowledge of teacher network and vice-versa. In our experiment, we aim to retain as much knowledge as possible from the router network (here the router network is the teacher network for experts) to the expert networks. In this way, we ensure that, the expert networks are at-least as good as the router network and if not, better. Thus in this literature, we fix the α value to 0.8. However, to learn more about the fine tuning of KD parameters we suggest to refer to the literature [33]. The purpose of leveraging KD in the loss function $Loss_{e_i}^{kd}$ is to simply retain all the knowledge of the router network in the experts. To illustrate the contrast, we construct another objective function depicted in Eq. (7) which is a variant of objective function in Eq. (4), but without knowledge distillation (wokd) term. We retrain all the experts using the loss function $Loss^{wokd}_{e_i}$ and illustrate performance gain by KD in the result discussion section.

$$Loss^{wokd}_{e_i} = \mathcal{X}\, l_{sub_i} + (1 - \mathcal{X})\, l_{all} \tag{7}$$

where,

$$l_{all} = -\sum_{j=1}^{N} \sum_{k=1}^{C} \delta(t_j, k) \log(P^k(d_j)) \tag{8}$$

Algorithm 1 illustrates the step by step training procedure of the MS-Net. In the Algorithm 1, line 1 through 4 performs the initialization of variable containers. In line 4 we obtain the subset class indices using the method discussed in Sect. 4. Line 6 and 7 load the subset of training data corresponding to the class index sets. In the Line 9 we randomly sample training data which consist of all classes. Thus we have two set of training data available, one with classes exclusively from the class index sets and the other with all available classes. Line 10 and 11 perform the forward pass of the expert network $e_i(.)$ for the data from *all classes* and class index set respectively. However, the objective function defined in line 12 optimizes either of the term based on the state of the random variable X. Finally we perform the back-propagation of the loss term followed by parameter update for expert network. We carry out this procedure for rest of class index sets and expert networks.

Algorithm 1: Training phase of MS-Net depicted in Figure 4

```
Input: Class index sets
Output: $C$ trained expert networks
1  Dataset with all classes:
     $\mathcal{D} = \{d_i | i = 1, .., N\}$
2  Teacher signal: $\mathcal{T} = \{t_i | i = 1, .., N\}$
3  Expert networks: $\mathcal{E} = \{e()_i | i = 1, .., C\}$
4  Class index sets: $\mathcal{S} = \{sub_i | i = 1, .., C\}$
5  for $i$, $sub$ in $enumerate(\mathcal{S})$ do
6      $\mathcal{D}_{sub} = \{d_i \in \mathcal{D} | t_i \in sub_i \wedge 1 \leq i \leq N\}$
7      $\mathcal{T}_{sub} = \{t_i \in \mathcal{T} | t_i \in sub_i \wedge 1 \leq i \leq N\}$
8      for $d_{sub}, t_{sub}$ in $enumerate(\mathcal{D}_{sub}, \mathcal{T}_{sub})$ do
9          $d_{all}, t_{all} = RandomSampler(\mathcal{D}, \mathcal{T})$
10         $o = e_i(d_{all})$
11         $o' = e_i(d_{sub})$
12         $L_{e_i} = \mathcal{X}\, l_{sub_i}(o', t_{sub}) + (1 - \mathcal{X})\, l_{all}(o, t_{all})$
             $\triangleright Pr(\mathcal{X} = 1) = \beta$ and
             $Pr(\mathcal{X} = 0) = 1 - \beta$
13         $BackPropagation(L_{e_i})$
14         $UpdateParameter(e_i)$
15     end
16 end
```

7 Inference phase

During the inference phase of MS-Net, the cost or the model complexity is dependent on two key parameters, namely, n for *top*-n evaluation; and the redundancy rate ρ. In the testing phase, the input is first fed to the router. From the router, we obtain the probability scores for each class. Since the router is relatively small it is less likely that most of time the *top*-1 will be correct. But needless to say, the probability of obtaining a correct answer increases as the value of n increases. Thus, we select the *top*-n most likely classes or predictions $\mathcal{P} = \{p_1, .., p_n\}$ from the sorted softmax scores $q_1, .., q_n$ of the router. Next, for each predicted class p_i the router chooses ρ experts from the expert pool, where $i = \{1, .., n\}$. Thus, as ρ increases the number of expert evaluation for a particular class increases proportionally. For each element or prediction in $\mathcal{P}$, we select a set of experts using the following equation:

$$\bar{\mathcal{E}} = \bigcup_{p \in \mathcal{P}} \{e(.)_p \in \mathcal{E} | \exists sub \in \mathcal{S} \wedge p \in sub\} \tag{9}$$

where, $\mathcal{E}$ is the set of all experts whose cardinality $|\mathcal{E}| = C$ (refer to Lemma 1), and $\bar{\mathcal{E}}$ is a subset of experts available for a certain set of predictions $\mathcal{P}$ for a single input datum. In the proposed MS-NET we will always have C expert neural networks. This is shown by Lemma 1 and Lemma 2. However, during inference we do not leverage all C expert neural networks. Rather, the expert neural networks are selected based on ρ and n. For each input datum, the router selects n most likely classes for re-checking. For each class, we use ρ expert neural networks to provide information for making the final decision. Thus, MS-Net leverages at-most $(\rho * n)$ and at-least $(\rho + (n - 1))$ expert networks during the inference phase. The value of $(\rho * n)$ and $(\rho + (n - 1))$ are always smaller than C. In this paper the maximum value for ρ and n are only 4 and 3 respectively. The prediction we obtain from the aggregated softmax of the set of selected experts $\bar{\mathcal{E}}$ for input x is presented in Eq. (10). For single input x, the softmax returns $\{q_1, .., q_C\}$, where each of the element q_i is the probability of x belonging to the class i.

$$\mathcal{O} = sm_r + \sum_{e \in \bar{\mathcal{E}}} sm_e(x) \tag{10}$$

where, sm_r and sm_e are the softmax scores by the router and experts, respectively. Finally, we take the most likely output label or the predicted class using Eq. (11)

$$prediction = \text{arg-max}_j(\mathcal{O} | q_j \in \mathcal{O}, 1 \leq j \leq C) \tag{11}$$

Algorithm 2: Testing phase of MS-Net

```
Input: Dataset $\mathcal{D}$
Output: accuracy: Accuracy of MS-Net
1  Empty list for top-n: $preds = \{\}$
2  Router network: $\mathcal{R}(.)$
3  Expert networks: $\mathcal{E} = \{e_i(.)|i = 1, .., C\}$
4  Flag dictionary: $visited[sub_1 : sub_C] = False$
5  Class index sets: $\mathcal{S} = \{sub_i|i = 1, .., C\}$
6  Counter: $c = 0$
7  for $d, t$ in $enumerate(\mathcal{D}, \mathcal{T})$ do
8      $preds = R(d)$
9      for $p$ in $preds$ do
10         for $i, sub$ in $enumerate(\mathcal{S})$ do
11             if $p \in$ in sub & visited[sub] == False then
12                 $loadExpertModel(e_i(.))$
13                 $asm = asm + softmax(e_i(d))$
14                 ▷ asm is the Average Soft-Max
                   $count = count + 1$
15                 $visited[sub] = True$
16             else
17                 continue
18             end
19         end
20         $prediction = \text{arg-max}_i(asm[0 : C])$
21         if $prediction == t$ then
22             correct = correct + 1
23         end
24     end
25     $accuracy = 100 * \frac{correct}{|D|}$
26 end
```

Algorithm 2 represents the testing phase of the MS-Net. Line 1 through 6 initialize the variables and all the networks (router and expert networks). Initially, we pass the input to the router network in line 8. We select the *top-n* most probable predictions from the router whose further re-evaluation start from line 9. Based on the prediction of router we select a fixed number of expert networks. As discussed in the earlier section, the number of expert networks for inference is governed by the variable ρ and *top-n*. In the worst case scenario we will have to evaluate at-most $(\rho * n)$ expert networks and in best case $(\rho + (n - 1)$ expert networks. We aggregate the softmax of all the expert networks in line 13 and increment the count (so far evaluated expert networks). After all the expert networks are evaluated we take the corrected or re-evaluated output based on the highest softmax value in line 20. The final output is the accuracy of MS-Net. In Line 4 and 15 of the Algorithm 2 the Boolean dictionary list $visited[sub_1 : sub_C]$ ensures that we are not evaluating an expert for particular subset more than once. This optimization comes into play during situation where the index of two or more predictions of router are consecutive numbers.

8 Experiments

8.1 Datasets

To evaluate and validate the effectiveness of the network we leverage three public datasets, which are CIFAR-10 or C-10 (Canadian Institute For Advanced Research)[1], CIFAR-100 or C-100[1], and F-MNIST (Fashion-Modified National Institute of Standards and Technology database). The CIFAR-10 dataset consists of 60,000 32X32 color images with 10 classes. Each class has 6,000 images. The dataset is divided into two parts with 50,000 images for training purposes and 10,000 images for testing[1]. The CIFAR-100 is just like CIFAR-10 but with 100 classes containing 600 images for each class. Among these 600 images for each class, 500 are for training and the rest 100 for testing. Moreover, the 100 classes are grouped into 20 super-classes. The F-MNIST database is a large database of fashion accessories. The database contains 60,000 training images and 10,000 testing images with 10 classes, where each image is 28X28 gray-scale image.

8.2 Experiment settings

We implement MS-Net in the PyTorch framework [44], and perform all the experiments on single NVIDIA RTX 2080 GPU. The setting of hyper-parameters during training slightly vary across different datasets. However, for all datasets, we use Stochastic Gradient Descent(SGD) with momentum. We set the initial learning rate for all routers and experts to $lr = 0.1$ and momentum to 0.9. Hyper-parameters such as batch size, iterations and learning rate decay scheduler (γ) differ across routers, experts and datasets which are shown in the Table 2.

9 Result discussion

For the CIFAR-10 and CIFAR-100 dataset, we perform a detailed empirical study on the effect of variable β (of objective function Eq. (4)) on expert networks during the training phase. We also perform analysis on effect of ρ and n during the test phase. In addition, beside ResNet-20 we also provide performance of MS-Net with two well-known DNNs as backbone. However, in this paper we perform all the empirical analysis and hyper-parameters search with

Table 2 Training hyper-parameters for router and experts

Network	Dataset	Batch size	Epochs	Steps
Router	C-10	32	300	50
	C-100	128	300	50
	F-MNIST	64	200	60
Experts	C-10	32	30	8
	C-100	16	25	8
	F-MNIST	64	30	10

Table 3 Performance on CIFAR-10 with variable probability distribution β

β	ρ	n	acc. (%)	δ
0.3	2	2	93.70	+102
		3	93.65	+97
	3	2	94.74	+206
		3	94.60	+191
	4	2	94.80	+212
		3	94.85	+217
0.5	2	2	93.65	+97
		3	93.66	+98
	3	2	94.75	+207
		3	94.64	+196
	4	2	95.00	+228
		3	95.00	+228
0.7	2	2	93.60	+92
		3	93.58	+90
	3	2	94.83	+215
		3	94.75	+207
	4	2	95.03	+235
		3	95.10	+242
0.9	2	2	93.54	+86
		3	93.34	+66
	3	2	94.15	+147
		3	94.06	+138
	4	2	95.38	+270
		3	95.30	+261
1.0	2	2	93.30	+52
		3	93.09	+41
	3	2	93.85	+117
		3	93.83	+115
	4	2	94.01	+133
		3	93.90	+123

The backbone (ResNet-20) score is 92.68%, and the δ score depicts the number of samples correctly re-classified by MS-Net expert networks (relative to the backbone)

the backbone ResNet-20. Tables 3 and 4 represent the performance of MS-Net (with ResNet-20 backbone) for CIFAR-10 and CIFAR-100, respectively Table 5.

In Table 6 we demonstrate the performance of individual expert network on subset class indices for dataset CIFAR-10 and F-MNIST. CIFAR-100 has 100 classes which make it difficult to interpret the performance of all 100 expert networks in a table. The table illustrates several key points about the MS-Net. Firstly, we observe that each of the expert network performs with remarkable score on its corresponding subset. That is, the performance of e_i on its corresponding subset sub_i, where $i = \{1, ..., C\}$, is very good (highlighted on Table 6). The performance of any expert networks on the whole set or on subsets assigned to other expert networks is relatively lower. Secondly, the performance of the Router R on each individual subset is significantly lower than that of the expert networks. However, when we execute the router and the expert networks together, they perform very well.

The empirical results for CIFAR-10 and CIFAR-100 suggest that, during training phase, fixing β to value 0.9 in the objective function tends to give relatively higher scores. To avoid redundant experiments, we perform rest of the training with β fixed to 0.9. Table 5 presents the performance of MS-Net for F-MNIST.

It is clear that with $\beta = 1$ in Eq. (4) we simply optimize the expert networks on training data sampled from subset class indices. On the contrary, with $\beta = 0$ we optimize the expert networks on the dataset comprising of all the available classes, which is analogous to the *naive Ensemble Learning (EL)* of DNNs. The optimal value for β has no theoretical bindings , rather it is dependent on the dataset. Expert networks trained with β in the range $0.3 \sim 0.9$ give near optimal classification scores. However, fixing β to either 0 or 1 during training degrades the performance scores, which implies that we should maintain a certain range for β while optimizing the proposed loss function. The variable n tells the experts up to how many *top-n* most probable prediction of router to further re-evaluate. For all the experiments, we re-evaluate up-to *top*-3 of router's prediction. The δ depicts the total number of samples correctly re-classified by the experts. *A positive δ value depicts the number of samples expert networks have correctly re-classified and a negative value for δ indicates the number of mis-classifications by the experts, or in other words, δ is the measurement of improvement in accuracy by our framework relative to the router network.* All the scores that we report in this paper (figures and tables) are relative to the backbone network, which in this case is the router network. It is worth noting that, we use the online inference method during the testing. Thus for MS-Net, we make the prediction for a single observation at each iteration as oppose to batch processing. Due to modular nature of the framework, the online inference is the simplest implementation.

Table 4 Performance on CIFAR-100 with variable probability distribution of β

β	ρ	n	acc. (%)	δ
0.3	2	2	71.00	+132
		3	70.80	+127
	3	2	71.27	+170
		3	71.09	+152
	4	2	71.06	+150
		3	71.06	+150
0.5	2	2	71.07	+148
		3	71.05	+142
	3	2	71.10	+152
		3	71.28	+170
	4	2	71.05	+142
		3	71.25	+167
0.7	2	2	71.00	+136
		3	71.01	+142
	3	2	71.03	+144
		3	71.11	+152
	4	2	71.52	+193
		3	71.25	+167
0.9	2	2	70.68	+110
		3	71.00	+141
	3	2	70.85	+127
		3	71.09	+151
	4	2	71.61	+203
		3	71.28	+170
1.0	2	2	69.73	+15
		3	69.72	+14
	3	2	69.74	+16
		3	69.52	− 5
	4	2	70.16	+58
		3	69.75	+17

The backbone (ResNet-20) score is 69.58%, and the δ score depicts the number of samples correctly re-classified by MS-Net expert networks (relative to the backbone)

Table 5 Performance on F-MNIST with $\beta = 0.9$

Dataset	ρ	n	acc. (%)	δ
FMNIST	2	2	95.80	+60
		3	95.96	+74
	3	2	95.80	+60
		3	96.77	+156
	4	2	96.02	+80
		3	96.77	+156

The backbone (ResNet-20) score for F-MNIST is 95.22%, and the δ score depicts the number of samples correctly re-classified by MS-Net's expert networks (relative to the backbone)

9.1 Performance on CIFAR-10

Table 3 represents the performance of MS-Net for CIFAR-10. From our experimental results, we can deduce the following key observations.

1. As we increase the value of ρ the accuracy increases. Refer to Fig. 8 (d, e) for graphical illustration of this phenomenon. However, for the case of CIFAR-10 increasing *top-n* beyond the value 2 does not improve the performance further (Fig. 8 (a, b and c)).
2. We observe gradual improvement in performance for the expert networks trained with increasing β which can be confirmed by Figs. 5 (a) and 6. The score gets lowest when we train the expert networks with $\beta = 1$. This phenomenon suggests that training the expert networks solely on its subset classes ($\beta = 1$ i.e. clamping $\mathcal{X} = 1$ in the objective function during the whole training process) does not improve performance, rather degrades. This degradation of result occurs due to imbalanced logit value in the last layer since the expert networks do not encounter any training data from rest of the classes (classes apart from the subset classes) during the training phase. Training these experts on the whole set of data alternatively within the optimal range of probability distribution substantially improve the performance. This method acts as a very effective regularization, as it prevents the experts from over-fitting on the dataset from subset classes. A graphical overview of the effect of the probability distribution β is presented in the bar chart Fig. 6.
3. In our experiment, we obtain the best score (with ResNet-20 backbone) for CIFAR-10 (95.38%) with $\rho = 4, n = 2$ and $\beta = 0.9$. The δ score with the mentioned parameters is +270, which means, integration of the expert networks with router further improves the performance by +2.70%. In other words, the router with a backbone network ResNet-20 has a *top*-1 accuracy of 92.68% and by integrating the experts for further re-evaluation, we levitate the *top*-1 score by +2.70 i.e. 95.38%.

9.2 Performance on CIFAR-100

Table 4 represents the result for CIFAR-100. For CIFAR-100, the same hyper-parameters $\rho = 4, n = 2$ and $\beta = 0.9$ give relatively high score of 71.68%. We can observe from the Table 7 that router's *top*-1 performance (ResNet-20) for CIFAR-100 is only 69.58%, and with the integration of the experts the performance increases by 2.48%. This phenomenon suggests that as we increase ρ and n we are more likely to get higher accuracy. The scores in Table 7

Table 6 Performance of individual expert network on all the subsets. The highlighted parts depict the score of each expert on its corresponding subset class index obtained through the Round Robin partition

Dataset	net	sub										S
		sub_0	sub_1	sub_2	sub_3	sub_4	sub_5	sub_6	sub_7	sub_8	sub_9	
C-10	e_0	**97.55**	95.125	89.6	63.325	58.07	60.15	69.15	75.87	83.27	91.17	75.13
	e_1	87.52	**98.22**	80.22	71.6	62.27	57.57	66	62.32	72.23	76.38	72.65
	e_2	80.05	92	**99.17**	83.52	65.55	55.32	45.97	47.85	58.92	70.2	70.54
	e_3	76.8	82.2	83.77	**97.55**	90.2	87	84.47	79.07	81	74.7	83.05
	e_4	70.87	74.25	74.75	83	**97.05**	95.42	92.82	89.5	88	81	84.5
	e_5	62.52	62.65	68.7	76.62	92.57	**98.7**	92	86.2	79.2	71.47	78.88
	e_6	61.27	62.42	62.37	71.27	84.7	91.25	**98.9**	91.25	82.35	73.5	77.77
	e_7	73.57	71.27	71.87	77.27	87.65	92.62	96.52	**98.27**	91.72	82.7	83.43
	e_8	81	77.92	73.25	78.17	84.45	87.4	92.47	96.27	**98.47**	91.23	85.33
	e_9	87.97	78.62	72.47	65.57	73.12	75.05	81.6	91.23	92.47	**98.8**	81.73
	R	90.07	90.02	87.87	89.97	92.85	93.1	94.87	94.15	94.62	93.15	92.68
FMNIST	e_0	**98.67**	85.05	80.30	60.57	60.77	77.35	77.85	97.87	97.27	96.17	82.32
	e_1	90.52	**97.10**	95.35	75.92	76.40	76.45	77.27	86.40	86.62	87.52	84.48
	e_2	91.52	96.90	**98.50**	80.80	80.25	80.27	81.82	92.15	91.40	92.20	87.90
	e_3	60.90	80.52	80.70	**98.72**	97.35	93.97	93.05	74.80	74.55	60.52	80.11
	e_4	40.12	62.60	62.55	82.25	**97.37**	94.60	90.90	70.05	70.10	51.55	72.61
	e_5	65.67	70.70	71.60	83.22	88.52	**98.32**	96.65	82.55	81.05	68.70	80.50
	e_6	74.95	70.95	70.50	76.90	80.47	96.65	**98.75**	86.42	86.87	78.15	81.81
	e_7	81.97	77.47	74.30	67.45	77.97	81.55	84.77	**98.55**	98.12	92.12	83.23
	e_8	75.22	71.95	60.27	51.45	45.60	45.01	57.67	74.74	**99.67**	91.35	84.66
	e_9	90.10	85.35	83.95	80.20	87.87	91.20	92.60	96.22	98.32	**99.87**	90.57
	R	90.07	90.02	87.87	89.97	92.85	93.1	94.87	94.15	94.62	93.15	95.22

The last column S depicts the performance of experts on whole set of data. The row with R represents the performance of router network on individual subset. In this table, all the experts and the router network have *ResNet-20* backbone. The ρ is 4 which also depicts the cardinality of each subset. We train all experts with $\beta = 0.9$

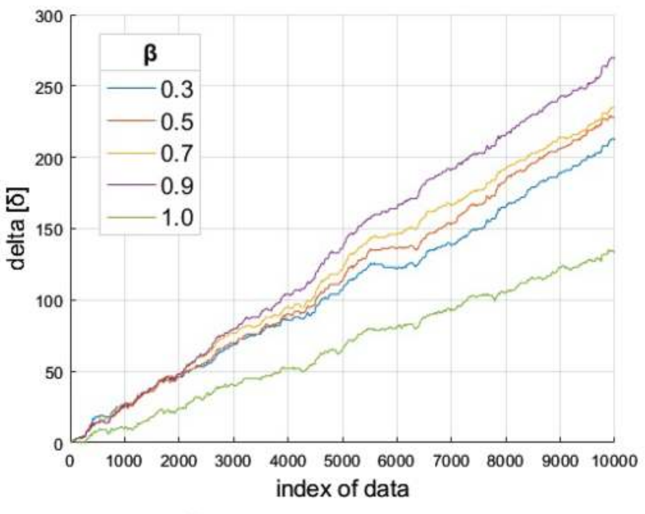
depict that MS-Net has relatively lower score on CIFAR-100 compared to CIFAR-10 and F-MNIST. This phenomenon is also observable for other state-of-the-art DNN (refer to Table 8). The probable reason for such low performance is mostly due to fewer amount of data per class in CIFAR-100. While CIFAR-10 has 6000 samples per class, CIFAR-100 has only 600 samples. This problem has been mitigated to a certain extent recently by leveraging large scale Transfer Learning (ImageNet pre-trained)[46], learning data augmentation policy or Auto-Augment (AA) [13], task-specific NAS with Transfer Learning (TL) [59, 71], Neural Architecture through hybrid online TL with multi-objective evolutionary search procedure [75] and so on. The MS-Net proposed in this study also has a significant improvement in performance compared to the backbone networks. We may expect further improvement if we introduce TL and other techniques described above.

9.3 Performance on F-MNIST

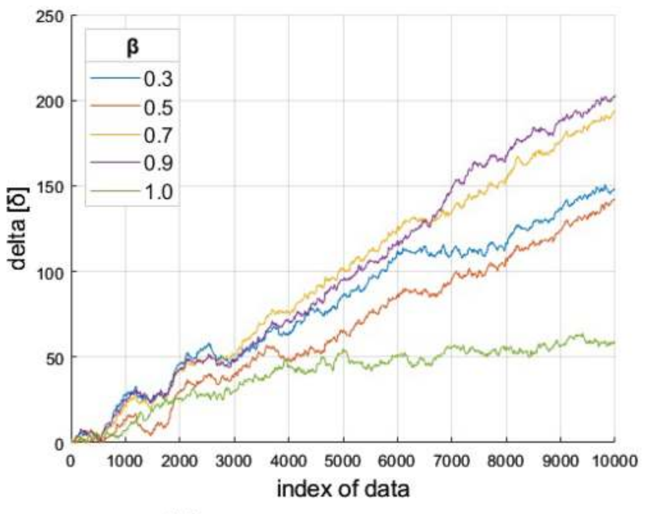
Table 5 represents the result of MS-Net on F-MNIST. In order to avoid redundant experiment the same hyper-parameters that give optimal score for CIFAR-10 and CIFAR-100 are set during the training. The network achieves score of 96.77% on F-MNIST test set. The delta score is +156, which is relatively higher. To our knowledge, the best score for F-MNIST was 96.30%, reported by Wide-ResNet-28-10 with Random Erasing data-augmentation[79]. Thus, MS-Net achieves a state-of-the-art score for this particular dataset.

10 Optimal value for ρ, top-n evaluation and β

A very common intuition is that as we increase the n of the router we can score (at best) as good as the router's *top-n* prediction score. In practice, increasing n beyond the value 2 does not substantially improve the performance, however, increasing the value of ρ gradually improves the accuracy of the network. For numerical comparison please refer to the Tables 3 and 4. In addition, Figs. 6 and 7 represent the effect of ρ and n for CIFAR-10 and CIFAR-100 respectively. An interesting observation from



(a) Delta score for CIFAR-10



(b) Delta score for CIFAR-100.

Fig. 5 The objective function of MS-Net optimized with different probability distribution β. The y-axis depicts the δ scores (no. of samples correctly re-classified by experts). The x-axis represents the index of each data-points. Each point in the graph depicts the number of samples correctly re-classified (of the ResNet-20 router) by the experts till that particular data-index

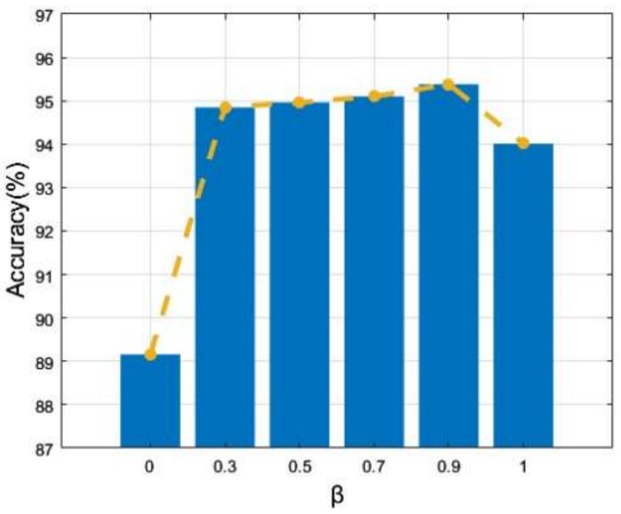


Fig. 6 Performance (%) of MS-Net (with Resnet-20 backbone) on CIFAR-10 with variable distribution for β. It is evident that optimizing the objective function with two extreme values $\beta = 0$ or 1 does not provide with an optimal performance. Probability distribution ranging from 0.3 to 0.9 tends to give the near optimal performance

the Fig. 8 is that, as we increase ρ the performance levitates dramatically, on the contrary, increasing n does not increase accuracy with a big margin. This is also the case for the CIFAR-100 dataset. The graphs in Fig. 9 indicate that, for a fixed value of ρ, increasing n further increases the accuracy, but not with a substantial margin. Although in this literature, our experiment is limited to $\rho \leq 4$, we can anticipate that for CIFAR-100 further increasing ρ will increase the accuracy. The reason is that CIFAR-100 is relatively difficult dataset with a large number of classes. Thus, from the above observations we can conclude with following guidelines for optimal parameters selection.

1. Evaluating till *top*-2 probable predictions of the router will suffice. This statement is true at-least for all the dataset we have explored so far.
2. Setting the redundancy rate variable ρ to 3 provides with a comparable classification score for all cases. We know that increasing ρ implies that we have more expert networks for each class. Thus, in situation where we have enough resource budgets, we can increase the variable ρ beyond 3 for more redundant expert networks and accuracy.
3. During the training phase the variable β of objective function (Eq. (4)) plays a crucial role in performance. Although there are no fixed value or theoretical bindings for β, we recommend to avoid fixing β to two extreme values i.e. 0 and 1. Optimizing the expert networks keeping β in range of 0.3–0.9 tend to give optimal score.

Thus, to keep the experiments simple, the training of MS-Net (implementation with different backbone networks) in the rest of the paper will confine to the above mentioned hyper-parameters.

Table 7 Performance of MS-Net for CIFAR-10 (C-10), CIFAR-100 (C-100) and F-MNIST

Type	Methods	C-10	C-100	F-MMNIST	# Param. (M)
Backbone	ResNet-20 [32]	92.68	69.58	95.22	0.269
	GoogleNet [69]	92.93	78.03	93.70	6.2
	MobileNet [35]	94.43	68.08	95.00	2.36
MS-Net framework	**MS-Net (ResNet-20)**	**95.38**	**71.61**	**96.77**	**2.95**
	MS-Net (GoogleNet)	**97.01**	**85.05**	**96.80**	**55.80**
	MS-Net (MobileNet)	**96.01**	**78.03**	**96.80**	**21.24**

The first section depicts the score of backbone networks itself, which also indicates the performance of routers. The second section represents the performance of our proposed framework (MS-Net) equipped with different backbone networks. We train MS-Net with different backbone networks with exact same hyper-parameters

Table 8 Performance of state-of-the-art networks for CIFAR-10 (C-10), CIFAR-100 (C-100) and F-MNIST

Type	Methods	C-10	C-100	F-MMNIST	# Param. (M)
Type-I	GPIPE + TL [37]	99.00	91.30	–	556
	Shared WRN [65]	97.47	82.57	–	118
	SGDR WRN-28-10 3 runs × 3 snapshots [53]	96.75	83.36	–	329
	SGDR WRN-28-10 16 runs × 3 snapshots [53]	96.75	83.36	–	1752
Type-II	Res2net-29 [25]	-	83.44	–	36.9
	PyramidNet+ShakeDrop+ Fast AA [52]	98.3	88.3	–	-
	Wide GatedResNet [66] (4,10) + Dropout	96.35	81.73	–	36.5
	DenseNet [36]	96.54	82.62	95.40	20
	Inception [70]	-	77.19	–	22.3
	VGG16-BN [67]	92.64	72.93	93.50	34.0
	ResNet-101 [32]	93.75	77.78	94.9	42.7
	ResNet-152 [32]	95.38	77.61	–	58.3
	FractalNet [47]	95.4	76.27	–	38.6
	NAS V3 [83]	96.35	-	–	37.4
	NASNet-A (7 @ 2304) [84]	97.03	-	–	27.6
	EfficientNet-B7 + NAS + TL [71]	98.9	-	–	64

The first section reports the score for relatively larger DNN, which we term as Type-I. The second section i.e. the Type-II are the networks that share relatively same computational capacity and parameters as MS-Net. Type-II section also includes automated learned architectures (without human intervention) through evolutionary search, reinforcement learning and so on. The table is divided for ease of comparison and contrast

11 Training expert networks without knowledge distillation

KD plays a vital role in training the expert networks. Earlier in the section 6 we proposed a slight variant of objective function (Eq. (4)) where we replace the KD term with simple cross entropy loss term (Eq. (7)). We train MS-Net on CIFAR-10, CIFAR-100 and F-MNIST leveraging the loss $Loss^{wokd}$ depicted in Eq. (7). The hyper-parameters are exactly identical to the experiments done with KD loss. The results show that, expert networks of MS-Net optimized without KD loss drops in accuracy with a substantial margin. In Fig. 10 we show the contrast for the classification accuracy. Figure 11 depict the contrast between the δ of MS-Net trained with and without the KD loss for CIFAR-10, CIFAR-100 and F-MNIST respectively. Using distillation in the loss term assists the expert networks in retaining the existing knowledge of the router. This also ensures the experts are at-least as good as the router network in the worst case scenario. In other words, the expert networks are less prone to mis-classify samples that are already correctly classified by the router networks.

12 Comparison to state-of-the-art results

In this section we provide a brief comparison of MS-Net to the performance of existing state-of-the-art DNNs on CIFAR-10, CIFAR-100 and F-MNIST dataset. For comparison we provide two tables, Tables 7 and 8, wherein,

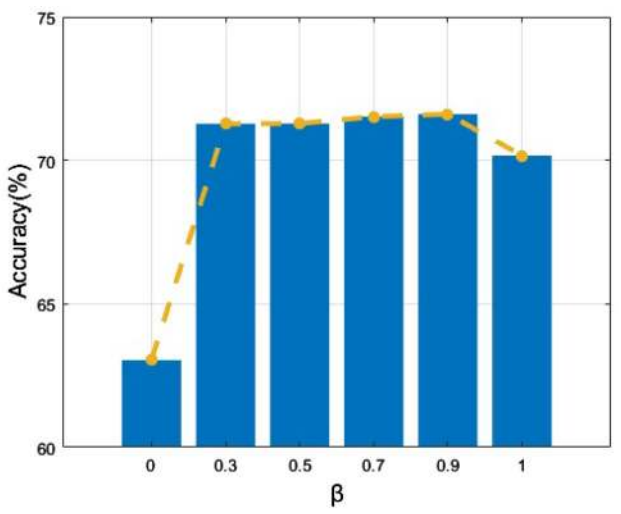


Fig. 7 Performance (%) of MS-Net on CIFAR-100 (ResNet-20 backbone) with different distribution for β. The optimal score for distribution 0.3 to 0.9 holds for CIFAR-100 too

first table depicts the performance of backbone networks (routers) and MS-Net frameworks (with different backbone networks), and second table represents the performance of state-of-the-art DNNs. For the ease of comparison and illustration, we divide the benchmark Table 8 into two types, where Type-I represents the network with large number of parameters, Type-II with parameters and computational resource almost similar to our proposed framework. The Type-II networks also include several architectures learned by using computationally expensive methods (e.g. evolutionary search and reinforcement learning) equipped with transfer learning (TL). For bench-marking, we refer to the site [79].

We can observe from the Table 7 that, MS-Net framework elevates the classification accuracy with a significant margin relative to the backbone router. Comparing MS-Net framework with Type-I networks from Table 8, the network actually performs with a neck-and-neck scores. However, compared to Type-II networks (approximately similar parameter counts) MS-Net performs with high score relative to most of the networks. The highest score that we obtain so far is with the backbone network *GoogLeNet*, leveraging at most 55.80M parameters (Table 7). This high score and setup undeniably come with a trade-off of more computational resources and parameter budget.

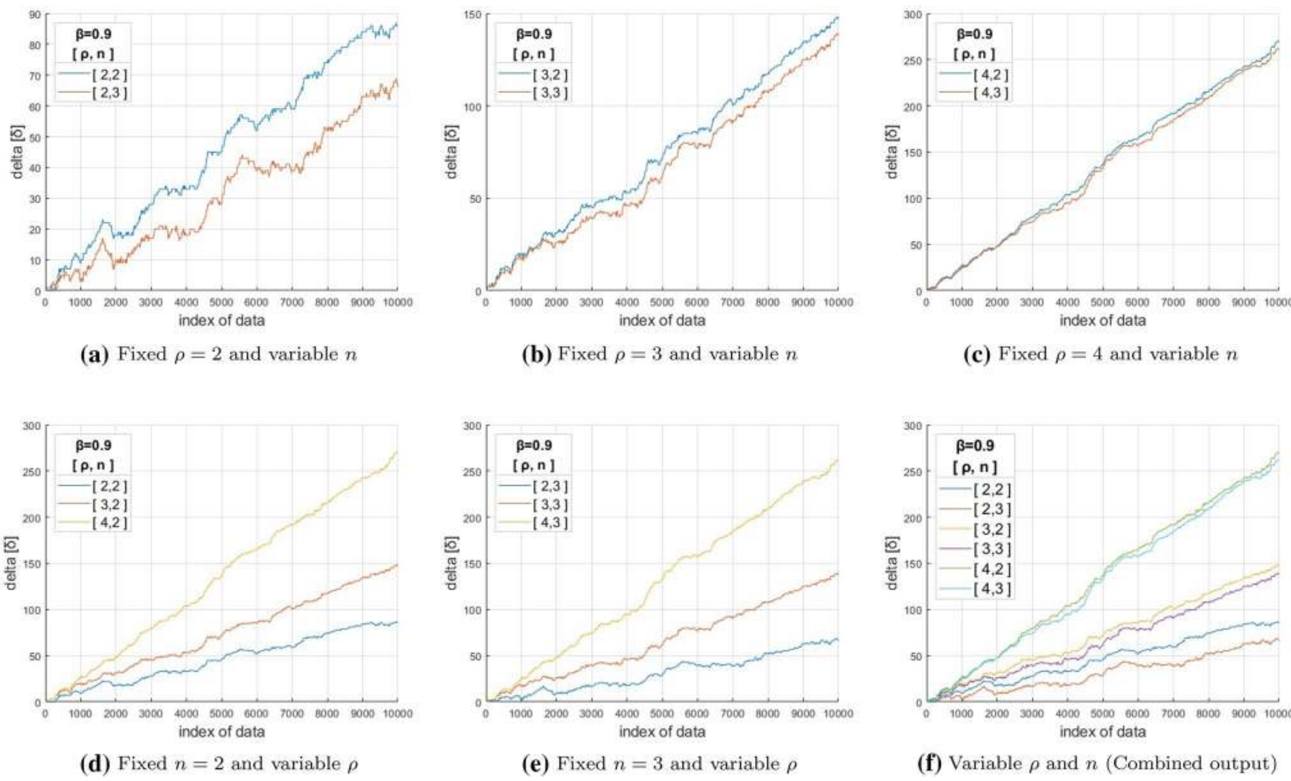
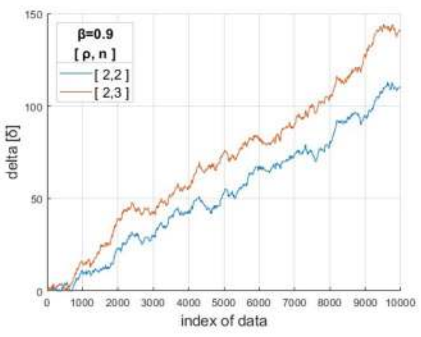
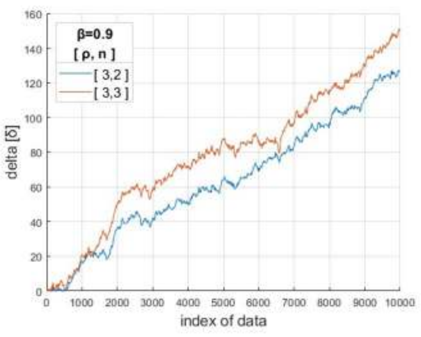


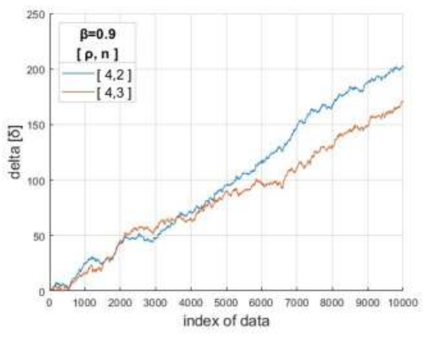
Fig. 8 The effect of variable n and ρ on CIFAR-10 during test phase: The first row (a,b and c) depicts the variation of δ score by keeping ρ fixed and changing variable n, i.e. it demonstrates how the network performs when we tweak the variable n. The second row (d and e) depicts performance of network keeping the n fixed while nudging variable ρ. The last Figure (f) summarizes all the δ score Figures (a, b, c, d and e) in a single graph for comparison



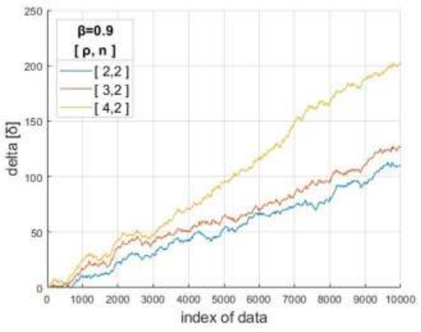
(a) Fixed $\rho = 2$ and variable n



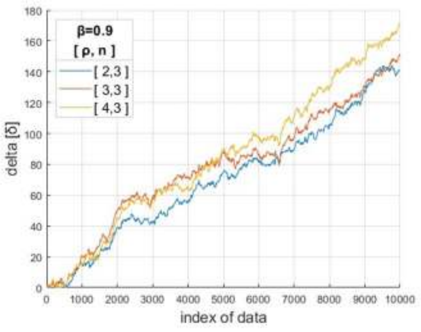
(b) Fixed $\rho = 3$ and variable n



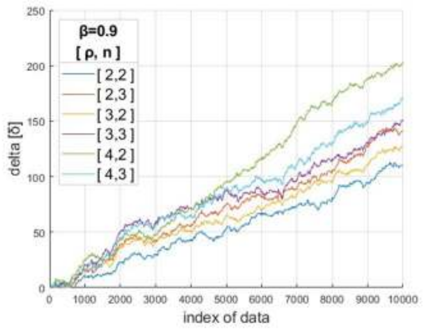
(c) Fixed $\rho = 4$ and variable n



(d) Fixed $n = 2$ and variable ρ

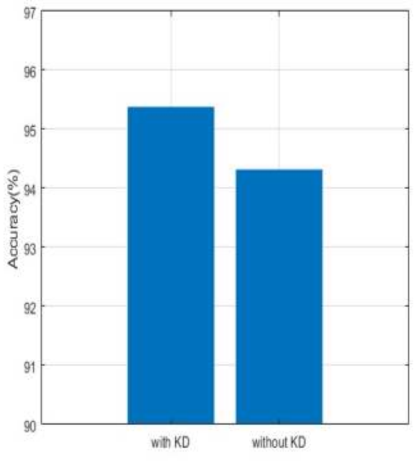


(e) Fixed $n = 3$ and variable ρ

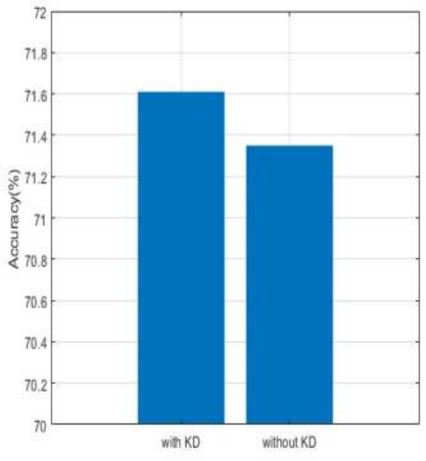


(f) Variable ρ and n (Combined output)

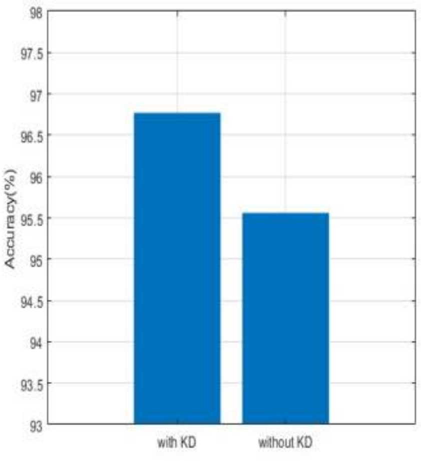
Fig. 9 The effect of variable n and ρ on CIFAR-100 during test phase: The first row (a,b and c) depicts the variation of δ score while keeping ρ fixed and changing variable n, i.e. it demonstrates how the network performs when we tweak the variable n. The first two figures of second row depict the performance of network keeping n fixed while nudging variable ρ. The last Fig. (f) combine all the Fig. (a, b, c, d and e) for depicting the contrast clearly



(a) CIFAR-10



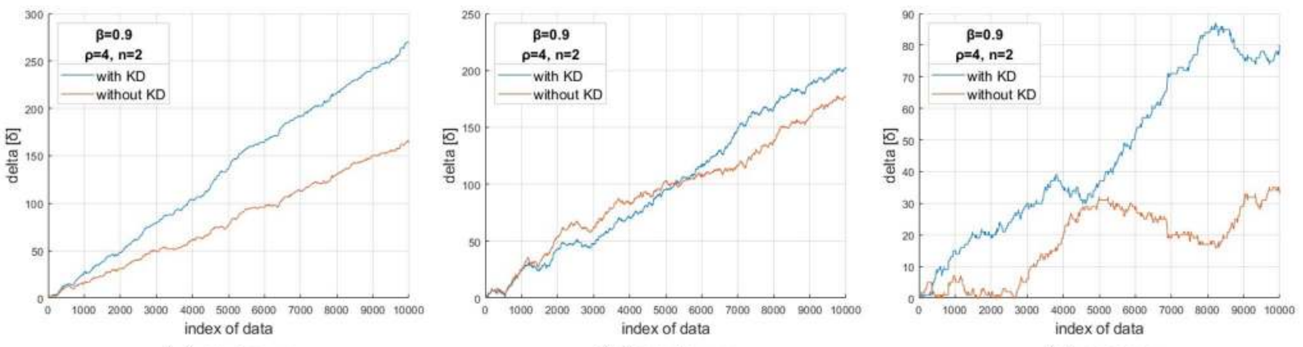
(b) CIFAR-100



(c) F-MNIST

Fig. 10 Performance of the MS-Net trained with and without KD loss

Most recently, researchers have been trying to find the best structure using evolutionary algorithms, reinforcement learning algorithms, and so on, and some very interesting results have been obtained [75, 82–84]. For example, for the database CIFAR-10, the best performance obtained so far is 98.9% (refer to Type-II section



(a) CIFAR-10 **(b)** CIFAR-100 **(c)** F-MNIST

Fig. 11 Contrast of scores: The figure represents the δ score difference for MS-Net trained with and without the KD loss

of Table 8) and the model's training parameter number is 64M [71]. However, based on the *'no free lunch theorem'* [34], an optimal model is usually fine-tuned for some specific database, and the model may not be useful for solving other problems. Even for the same problem with more observed data, to preserve the best performance, we have to use a very expensive process to re-design the model. On the other hand, the MS-NET structure proposed in this study is very simple, and can leverage the performance of any existing state-of-the-art models by increasing the inference cost slightly. In this sense, MS-NET can be a good starting point for solving various problems.

13 Conclusion

In this paper, we have proposed a modular neural network architecture termed as the MS-Net (Modular Redundant Network). For a C-class classification problem, the network consisted of a router network and C expert networks. In summary, the key idea of the research have been to further re-evaluate the *top-n* most probable predictions of the router by leveraging these expert networks. To effectively train these expert networks we have proposed a stochastic objective function equipped with the knowledge distillation technique that facilitates alternative training on a subset of expert data and whole set of data. This alternative training have been regulated by clamping a Bernoulli random variable to each of loss function term. We have constructed the subsets of data systematically by Round-Robin fashion. As a result, it has provided us with a mean to control the redundancy of each class in the set of subsets, which have also allowed us to know which expert network is a specialist on which subset (thus we have more interpret-ability). We have shown that, with a very limited parameter budget and simple DNN as backbone, our network has achieved performance comparable or sometimes equivalent to more complex DNNs.

An interesting research direction would be to apply Neural Architecture Search (NAS) strategy in MS-Net. We can anticipate that implementing such approach can further cut down redundant parameters of expert networks. In this way, each expert network can be reshaped and designed based on its assigned expert data. Fortunately, the modular nature of MS-Net have allowed each of the individual expert network to be independent and local, i.e. they are not dependent on each other during the training and inference phase. This suggests that, the experts can be trained and tested in parallel, which give us an opportunity to utilize powerful parallel computing systems. Further optimization of this network can be obtained by reducing the number of expert networks evaluation. As we have known from our experiments, during the inference phase the router network chooses fixed number of experts for further evaluation. The final prediction can been obtained only after all the selected experts have been completely evaluated. This can be time-consuming and redundant for easy data or patterns. To mitigate the unnecessary evaluation of experts the concept of *Progressive inference* introduced in the literature[49] can be leveraged in our modular network. The main idea is to stop evaluation of expert networks once we reach a certain softmax confidence or threshold (the threshold can be obtained through trial and error). In this way the parameter usage can be further reduced without comprising the network accuracy. So far, in this paper we have leveraged the router network as the teacher model for knowledge distillation. We can anticipate that using more accurate and powerful DNN as the teacher model can assist the expert networks in generalizing better. In our future work, we anticipate to deploy the proposed modular neural network in real world scenario. In order to test the network, we will build a test bed equipped with multiple neural computational sticks (powered by vision processing units) and run several experts in parallel for faster and efficient inference.

Compliance with ethical standards

Conflict of interest There is no conflict of interest.

References

1. URL https://www.cs.toronto.edu/~kriz/cifar.html
2. Ahonen T, Hadid A, Pietikäinen M (2004) Face recognition with local binary patterns. European conference on computer vision. Springer, Berlin, pp 469–481
3. Amodei D, Ananthanarayanan S, Anubhai R, Bai J, Battenberg E, Case C, Casper J, Catanzaro B, Cheng Q, Chen G, et al. (2016) Deep speech 2: End-to-end speech recognition in english and mandarin. In: International conference on machine learning, pp 173–182
4. Anand R, Mehrotra K, Mohan CK, Ranka S (1995) Efficient classification for multiclass problems using modular neural networks. IEEE Trans Neural Netw 6(1):117–124
5. Badrinarayanan V, Kendall A, Cipolla R (2017) Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell 39(12):2481–2495
6. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
7. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
8. Buciluă C, Caruana R, Niculescu-Mizil A (2006) Model compression. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 535–541. ACM
9. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2017) Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans Pattern Anal Mach Intell 40(4):834–848
10. Clark P, Boswell R (1991) Rule induction with cn2: Some recent improvements. European Working Session on Learning. Springer, Berlin, pp 151–163
11. Collobert R, Weston J (2008) A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning, pp 160–167. ACM
12. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
13. Cubuk ED, Zoph B, Mane D, Vasudevan V, Le QV (2019) Autoaugment: Learning augmentation strategies from data. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 113–123
14. Cubuk ED, Zoph B, Shlens J, Le QV (2019) Randaugment: Practical data augmentation with no separate search. arXiv preprint arXiv:1909.13719
15. Dumoulin V, Visin F (2016) A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285
16. Dutt A, Pellerin D, Quénot G (2019) Coupled ensembles of neural networks. Neurocomputing
17. Elsken T, Metzen JH, Hutter F (2018) Neural architecture search: A survey. arXiv preprint arXiv:1808.05377
18. Freund Y, Schapire RE (1995) A desicion-theoretic generalization of on-line learning and an application to boosting. European conference on computational learning theory. Springer, Berlin, pp 23–37
19. Friedman JH (2002) Stochastic gradient boosting. Comput Statist Data Analy 38(4):367–378
20. Furlanello T, Lipton ZC, Tschannen M, Itti L, Anandkumar A (2018) Born again neural networks. arXiv preprint arXiv:1805.04770
21. Fürnkranz J (1999) Separate-and-conquer rule learning. Artif Intell Rev 13(1):3–54
22. Fürnkranz J (2002) Pairwise classification as an ensemble technique. European Conference on Machine Learning. Springer, Berlin, pp 97–110
23. Fürnkranz J (2002) Round robin classification. J Mach Learn Res 2:721–747 (Mar)
24. Fürnkranz J (2003) Round robin ensembles. Intell Data Anal 7(5):385–403
25. Gao S, Cheng MM, Zhao K, Zhang XY, Yang MH, Torr PH (2019) Res2net: A new multi-scale backbone architecture. IEEE transactions on pattern analysis and machine intelligence
26. Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
27. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680
28. Goodfellow IJ, Mirza M, Xiao D, Courville A, Bengio Y (2013) An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211
29. Graves A, Mohamed Ar, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing, pp 6645–6649. IEEE
30. Hampshire JB II, Waibel A (1992) The meta-pi network: Building distributed knowledge representations for robust multisource pattern recognition. IEEE Trans Pattern Anal Mach Intell 7:751–769
31. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
32. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. European conference on computer vision. Springer, Berlin, pp 630–645
33. Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531
34. Ho YC, Pepyne DL (2002) Simple explanation of the no-free-lunch theorem and its implications. J Optimiz Theory Appl 115(3):549–570
35. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861
36. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
37. Huang Y, Cheng Y, Bapna A, Firat O, Chen D, Chen M, Lee H, Ngiam J, Le QV, Wu Y, et al. (2019) Gpipe: Efficient training of giant neural networks using pipeline parallelism. In: Advances in Neural Information Processing Systems, pp 103–112
38. Intisar CM, Watanobe Y (2018) Classification of online judge programmers based on rule extraction from self organizing

feature map. In: 2018 9th International Conference on Awareness Science and Technology (iCAST), pp 313–318. IEEE
39. Intisar CM, Zhao Q (2019) A selective modular neural network framework. In: 2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST), pp 1–6. IEEE
40. Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE et al (1991) Adaptive mixtures of local experts. Neural Comput 3(1):79–87
41. Jain AK, Farrokhnia F (1991) Unsupervised texture segmentation using gabor filters. Pattern Recogn 24(12):1167–1186
42. Jordan MI, Jacobs RA (1994) Hierarchical mixtures of experts and the em algorithm. Neural Comput 6(2):181–214
43. Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: A survey. J Artif Intell Res 4:237–285
44. Ketkar N (2017) Introduction to pytorch. Deep learning with python. Springer, Berlin, pp 195–208
45. Kim T, Cha M, Kim H, Lee JK, Kim J (2017) Learning to discover cross-domain relations with generative adversarial networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp 1857–1865. JMLR. org
46. Kolesnikov A, Beyer L, Zhai X, Puigcerver J, Yung J, Gelly S, Houlsby N (2019) Big transfer (bit): General visual representation learning. arXiv preprint arXiv:1912.11370
47. Larsson G, Maire M, Shakhnarovich G (2016) Fractalnet: Ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648
48. LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: Advances in neural information processing systems, pp 598–605
49. Lee H, Lee JS (2018) Local critic training for model-parallel learning of deep neural networks. arXiv preprint arXiv:1805.01128
50. Leung HC, Zue VW (1989) Applications of error back-propagation to phonetic classification. In: Advances in neural information processing systems, pp 206–214
51. Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2016) Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710
52. Lim S, Kim I, Kim T, Kim C, Kim S (2019) Fast autoaugment. arXiv preprint arXiv:1905.00397
53. Loshchilov I, Hutter F (2016) Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983
54. Lowe DG, et al. (1999) Object recognition from local scale-invariant features. In: iccv, vol. 99, pp 1150–1157
55. Minsky M (1988) Society of mind. Simon and Schuster, New York
56. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602
57. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529
58. Molchanov P, Tyree S, Karras T, Aila T, Kautz J (2016) Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440
59. Nayman N, Noy A, Ridnik T, Friedman I, Jin R, Zelnik L (2019) Xnas: Neural architecture search with expert advice. In: Advances in Neural Information Processing Systems, pp 1977–1987
60. Park SH, Fürnkranz J (2012) Efficient prediction algorithms for binary decomposition techniques. Data Min Knowl Discovery 24(1):40–77
61. Pham H, Guan MY, Zoph B, Le QV, Dean J (2018) Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268
62. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
63. Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. International Conference on Medical image computing and computer-assisted intervention. Springer, Berlin, pp 234–241
64. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al (2015) Imagenet large scale visual recognition challenge. Int J Computer Vision 115(3):211–252
65. Savarese P, Maire M (2019) Learning implicitly recurrent cnns through parameter sharing. arXiv preprint arXiv:1902.09701
66. Savarese PH, Mazza LO, Figueiredo DR (2016) Learning identity mappings with residual gates. arXiv preprint arXiv:1611.01260
67. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
68. Socher R, Bengio Y, Manning C (2012) Deep learning for nlp. Tutorial at Association of Computational Logistics (ACL)
69. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
70. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826
71. Tan M, Le QV (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946
72. Waibel A, Sawai H, Shikano K (1989) Modularity and scaling in large phonemic neural networks. IEEE Trans Acoust Speech Signal Process 37(12):1888–1898
73. Warburton K (2003) Deep learning and education for sustainability. Int J Sustainab Higher Educ 4(1):44–56
74. Watanabe C, Hiramatsu K, Kashino K (2018) Modular representation of layered neural networks. Neural Netw 97:62–73
75. Wong C, Houlsby N, Lu Y, Gesmundo A (2018) Transfer learning with neural automl. In: Advances in Neural Information Processing Systems, pp 8356–8365
76. Wu H, Zhang J, Huang K, Liang K, Yu Y (2019) Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation. arXiv preprint arXiv:1903.11816
77. Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1492–1500
78. Yang C, Xie L, Qiao S, Yuille A (2018) Knowledge distillation in generations: More tolerant teachers educate better students. arXiv preprint arXiv:1805.05551
79. Zalandoresearch: zalandoresearch/fashion-mnist (2019). URL https://github.com/zalandoresearch/fashion-mnist
80. Zhang Y, Xiang T, Hospedales TM, Lu H (2018) Deep mutual learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4320–4328
81. Zhao Q (1997) Stable online evolutionary learning of nn-mlp. IEEE Trans Neural Netw 8(6):1371–1378
82. Zhao Q, Higuchi T (1996) Evolutionary learning of nearest-neighbor mlp. IEEE Trans Neural Netw 7(3):762–767
83. Zoph B, Le QV (2016) Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578
84. Zoph B, Vasudevan V, Shlens J, Le QV (2018) Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8697–8710