Class Means as an Early Exit Decision Mechanism

Alperen Gormez and Erdem Koyuncu
Department of Electrical and Computer Engineering, University of Illinois at Chicago
{agorme2,ekoyuncu}@uic.edu

***Abstract*—State-of-the-art neural networks with early exit mechanisms often need considerable amount of training and fine-tuning to achieve good performance with low computational cost. We propose a novel early exit technique based on the class means of samples. Unlike most existing schemes, our method does not require gradient-based training of internal classifiers. This makes our method particularly useful for neural network training in low-power devices, as in wireless edge networks. In particular, given a fixed training time budget, our scheme achieves higher accuracy as compared to existing early exit mechanisms. Moreover, if there are no limitations on the training time budget, our method can be combined with an existing early exit scheme to boost its performance, achieving a better trade-off between computational cost and network accuracy.**

***Index Terms*—Neural networks, early exit, class means.**

I. Introduction

Modern deep learning models require a vast amount of computational resources to effectively perform various tasks such as object detection [1], image classification [2], machine translation, [3] and text generation [4]. Deploying deep learning models to the edge, such as to mobile phones or the Internet of Things (IoT), thus becomes particularly challenging due to device computation and energy limitations [5]. Moreover, the law of diminishing returns applies to the trade-off computation-performance trade-off [6]: The increase in a deep learning model's performance is often marginal as compared to the increase in the amount of computation.

One of the primary reasons behind traditional deep learning models' high computation demand is their tunnel-like design. In fact, traditional models apply the same sequence of operations to any given input. However, in many real world datasets, certain inputs may consist of much simpler features as compared to other inputs [6]. In such a scenario, it becomes desirable to design more efficient architectures that can exploit the heterogeneous complexity of dataset members. This can be achieved by introducing additional exit points to the models [7]–[9]. These exit points prevent simple inputs to traverse the entire network, reducing the computational cost of inference.

Despite reduced inference time, existing early exit neural network architectures require additional training and fine-tuning for the early exit points, which increases the training time [6], [8], [9]. This side-effect is undesirable for scenarios in which the training has to be done in a low-power device. An ideal solution is a plug-and-play approach that does not require gradient-based training and performs well. In this work, we propose such an early-exit mechanism based on the *class means* of input samples for the image classification task. By taking the mean of layer outputs for each class at every layer of the model, *class means* are obtained. During inference, output of a layer is compared with the corresponding class means using Euclidean distance as the metric. If the output of the layer is close enough to a class mean, the execution is stopped and the sample exits the network. In fact, as seen in Fig. 1, some samples can be classified easily at early stages of the network by just considering a "nearest class mean" decision rule, suggesting the potential effectiveness of our method for reducing computational cost.

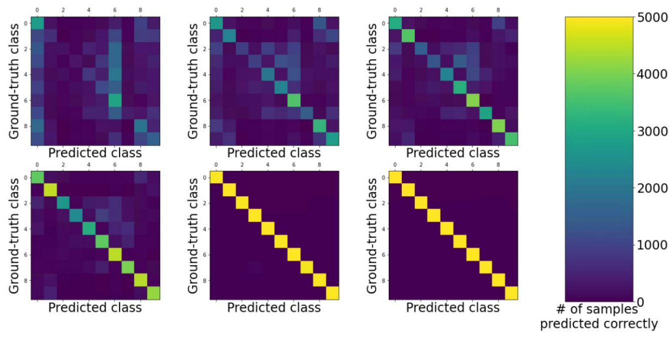

Fig. 1. Confusion matrix of the classifications done according to the nearest class mean on CIFAR-10 training set. From left to right, top to bottom; the results belong to the first convolutional layer; 1^{st}, 3^{rd}, 10^{th}, 15^{th} and 30^{th} residual block of ResNet-152.

To the best of our knowledge, our method is the first early exit mechanism that does not require a gradient-based training and does not modify the original network by any means. Furthermore, our method does not have a hyper-parameter for the early exit locations unlike existing schemes. These features make our method simpler to use and easier to deploy on low-power devices. While using class means as the only early exit mechanism requires just a single feed forward pass, existing early exit methods reaches the same performance after training for multiple epochs, which suggests our method is more agile and powerful yet simpler. Moreover, combining class means with the existing mechanisms that require gradient-based training achieves a better trade-off in terms of computation cost and network accuracy.

We show the effectiveness of our method on CIFAR-10 [10] dataset using ResNet-152 [2] and WideResNet-101 models [11]. Using class means as the sole decision mechanism for early exiting results in 35% better accuracy or 50% faster inference time compared to the current state-of-the-art. When combined with the state-of-the-art, we increase the accuracy by 9% or decrease the inference time by 25%.

II. Related Work

Our work is related to the area of *conditional computation* [12], where several small networks are trained to control the computation flow of one deep neural network. For this purpose, adding gates between the blocks of residual networks have been proposed [13]. During inference, these gates allow the input to skip unnecessary blocks, thus saving computation time. However, unlike our proposed method, the gated network has to be trained from scratch. Conditional computation policies can also be learned through reinforcement learning [14]. However, this latter approach forces the inputs to go through the entire network as it does not incorporate early exit points.

One of the earliest works that explicitly propose the idea of early exiting is [7], where the authors consider adding a cascade of linear layers after convolutional layers as control blocks. Rather than just linear layers, adding *branches* consisting of convolutional layers to the original model has also been studied [8]. A significant drawback of this method is that the branches may increase the computational cost due to convolutional layers. In addition, this idea requires branches to be trained with the original model jointly, from scratch. In a more recent study, *internal classifiers (ICs)* consisting of a feature reduction layer and a single linear layer are added after certain layers in the network [9]. Hence, the methods presented in these studies modify the original network by adding linear or convolutional layers. Moreover, they require gradient updates to train those layers. Also, an implicit hyper-parameter is the locations of the early exit points. Our method is better suited for low-power applications compared to existing studies since we do not modify the original model, do not require gradient based training and additional hyper-parameters.

In addition to layer level early exits, a network level early exit mechanism has been introduced in [6]. Both the layer level exit and the network level exit require decision functions to be inserted between the layers and networks. This type of architecture freezes the weights of the original model, and then trains the decision functions one by one using weighted binary classification. The drawback of this approach is such an alternative optimization, which may consume a lot of time and energy when there are many decision functions to optimize.

The idea of early exit neural networks shows promising results in natural language processing domain too. Adding early exit points between the transformer layers reduce the computation time with marginal loss in performance [15], [16]. Applications for neural network based early exit mechanisms to computer fault management have also been considered [17].

Intermediate layer outputs have been used for classification in few-shot and one-shot learning settings in the past [18]–[21]. These studies are closely related with the area of *metric learning*. The closest work to ours is *prototypical networks*, in which *prototypes* for each class are computed [20]. However, none of those approaches aim to reduce the computation cost.

III. Using Class Means for Early Exit

We study the problem of image classification. Let $(x_0^{(i)}, y^{(i)}) \in D$ be an image-label pair from the dataset D consisting of N samples, where $y^{(i)} \in \{1, 2, \dots, K\}$ and $i \in \{1, 2, \dots, N\}$. We denote the network F with M layers as a sequence $l_1, l_2, \dots, l_M$. Let $\hat{y}^{(i)}$ denote the prediction of the network, $x_j^{(i)}$ denote the output of layer j, and $\hat{y}_j^{(i)}$ denote the prediction in case the input exits the network after layer j, for $j = 1, 2, \dots, M$. We can write the full equation for the network F as

$$x_j^{(i)} = l_j(x_{j-1}^{(i)}),\ j = 1, 2, \dots, M. \tag{1}$$

A. Class Means

The input to our early exit method is the network F trained on D. The network F is not modified by any means. Therefore, we can obtain the class means for each class at each layer easily by just a forward pass. This is especially useful when the training time budget is fixed. Let S_k denote the set of samples whose ground-truth label is k, and c_j^k denote the mean of the output of layer j for class k. In other words, let

$$c_j^k = \frac{1}{|S_k|} \sum_{n \in S_k} x_j^{(n)}. \tag{2}$$

Then, the Euclidean distance between a layer output $x_j^{(i)}$ and K class means c_j^k is computed via

$$d_j^{(i)^k} = ||x_j^{(i)} - c_j^k||_2,\ k \in \{1, 2, \dots, K\}. \tag{3}$$

After calculating $d_j^{(i)^k}$ at each layer for every sample in the dataset, we normalize the distances for each class as

$$d_j^{(i)^k} := \frac{d_j^{(i)^k}}{\frac{1}{N}\sum_{i=1}^{N} d_j^{(i)^k}},\ k \in \{1, 2, \dots, K\}. \tag{4}$$

Finally, the normalized distances are converted to probabilities of input belonging to a class in order to perform inference. This is done using the softmax function as

$$P(\hat{y}_j^{(i)} = k) = \text{softmax}(-d_j^{k^{(i)}}). \tag{5}$$

During inference, the decision of exiting after l_j or moving forward to l_{j+1} is made according to a threshold value T_j. If the largest softmax probability is greater than the specificed threshold T_j, execution is stopped and the class with the largest softmax probability is predicted. In other words, if

$$\max(\text{softmax}(-d_j^{k^{(i)}})) > T_j, \tag{6}$$

then the network predicts

$$\hat{y}_j^{(i)} = \arg\max_k(\text{softmax}(-d_j^{k^{(i)}})). \tag{7}$$

Otherwise, the input moves forward to the next layer. In the worst case, execution ends at the last layer of the network.

Class means performs differently according to different set of threshold values. Following the existing works, we exhaustively try different sets of threshold values on the training set and choose the set of thresholds that gives the desired computational cost and network accuracy. We then evaluate these thresholds on a test set during the inference phase. The full procedure of our method is shown in Algorithm 1.

Algorithm 1 Class Means for Early Exiting

```
**Input:** Network layers $l_j$, dataset $D$, thresholds $T_j$
if training then
    for $j = 1$ to $M$ do
        $x_j^{(i)} = l_j(x_{j-1}^{(i)})$
        Calculate class means $c_j^k$, $k \in \{1, 2, \ldots, K\}$
    end for
end if
if inference then
    for $j = 1$ to $M$ do
        $x_j^{(i)} = l_j(x_{j-1}^{(i)})$
        Compute $d_j^{k^{(i)}} = ||x_j^{(i)} - c_j^k||_2$
        Normalize $d_j^{k^{(i)}}$ as in (4).
        if $\max(\text{softmax}(-d_j^{k^{(i)}})) > T_j$ then
            Early Exit with $\arg\max_k(\text{softmax}(-d_j^{k^{(i)}}))$.
        end if
    end for
end if
```

B. Combination of Class Means with Existing Schemes

Performance of an existing exit scheme can be boosted by combining it with our class means mechanism. In this context, existing methods decide either according to the entropy of the early exit prediction [8], or the largest probability value in the prediction [9]. In the former strategy, the input to the neural network exits early if the entropy is smaller than a threshold T_j for Layer j of the network. In the latter, the sample exits early if the largest probability value is greater than T_j.

Existing methods use only $x_j^{(i)}$ as the input to the next layer. We propose feeding the class means c_j^k as additional inputs to the layers, which improves the performance. During inference, if the j^{th} internal classifier decides to skip to the next layer, class means are consulted. If class means do not approve moving on, the prediction of the j^{th} internal classifier is returned and the input exits early. Hence, an input can skip to the next layer if and only if it receives approval from both the internal classifier (which can be based on any existing scheme) and our simple class means classifier.

IV. Results

We validate the effectiveness of our method on CIFAR-10 dataset, which consists of 50000 training and 10000 test images [10]. There are 10 classes which have equal amount of samples. We use ResNet-152 [2] and WideResNet-101 [11] models. For both models, we use the same data augmentation scheme and the hyper-parameter values stated in [13] for training. We use cross-entropy loss for all trainings.

We run experiments in two settings. First, we fix the training time budget and compare class means with the existing methods in terms of network accuracy and floating point operations (FLOPs) performed during inference. In the second setting, we allow training of the internal classifiers, and we compare the combination of class means and internal classifiers against existing methods.

A. Class Means Under a Fixed Training Time Budget

In the fixed training time budget setting, we compare class means with existing methods in two ways. First, class means is compared with Shallow-Deep Networks [9] and BranchyNet [8], which are trained for only one epoch since class means require only a single forward pass. We do not include here the Bolukbasi-Wang-Dekel-Saligrama (BWDS) method [6], because in this method, each decision function requires a separate training. Hence, we cannot train an entire BWDS network with many decision functions using only one epoch.

Shallow-Deep Networks add internal classifiers after certain layers in the network. We use the procedure described in [9] and add 6 ICs after the layers which correspond to the 15%, 30%, 45%, 60%, 75%, 90% of the entire network in terms of FLOPs. For BranchyNet, we add 2 branches to the original network. The first branch is after the first convolutional layer, and the second branch is after the layer that corresponds to 1/3 of the whole network in terms of FLOPs.

In the second setting, we still consider a fixed training time budget, but this time we allow the separate training of decision functions in the BWDS method. As suggested by the authors, there are 6 early exit points, hence 6 decision functions. We train each decision function for 1 epoch, resulting in 6 separate epochs. To make a fair comparison, we train Shallow-Deep Networks and BranchyNet for 6 epochs as well. We combine class means with Shallow-Deep Networks.

To select the best thresholds, we uniformly draw 10000 random vectors of length M from $[0, 1]$. The j^{th} element of the vector is T_j and it corresponds to l_j. Following Algorithm 1, we obtain 10000 different points on the FLOPs-Accuracy plane. We compute the convex hull of this point cloud, and take the upper half of the boundary of the convex hull. This gives us the best performing thresholds on the training set. Then, we use these thresholds on test set and form the FLOPs-Accuracy curve using time sharing.

As shown in Fig. 2, class means outperform all existing schemes under a fixed training time budget. Specifically, using class means as the only decision mechanism achieves 35% better accuracy or %50 faster inference time for certain cases. This is because class means use the pretrained features of the original model, which generalizes well to the dataset and can be used for classification. On the other hand, internal classifiers require further training. Therefore, they require more time to be ready for the task of classification.

It should also be noted that class means achieve the same performance as the full network but using only 25%-30% of the available computational resources. This suggests that training internal classifiers is not an absolute necessity.

B. Class Means with Unlimited Training

In this setting, we lift the training time budget. We combine class means with Shallow-Deep Networks and compare this combination against BranchyNet, the BWDS method, and Shallow-Deep Networks only.

We train the internal classifiers of our merger of class means and Shallow-Deep for 100 epochs. The corresponding high

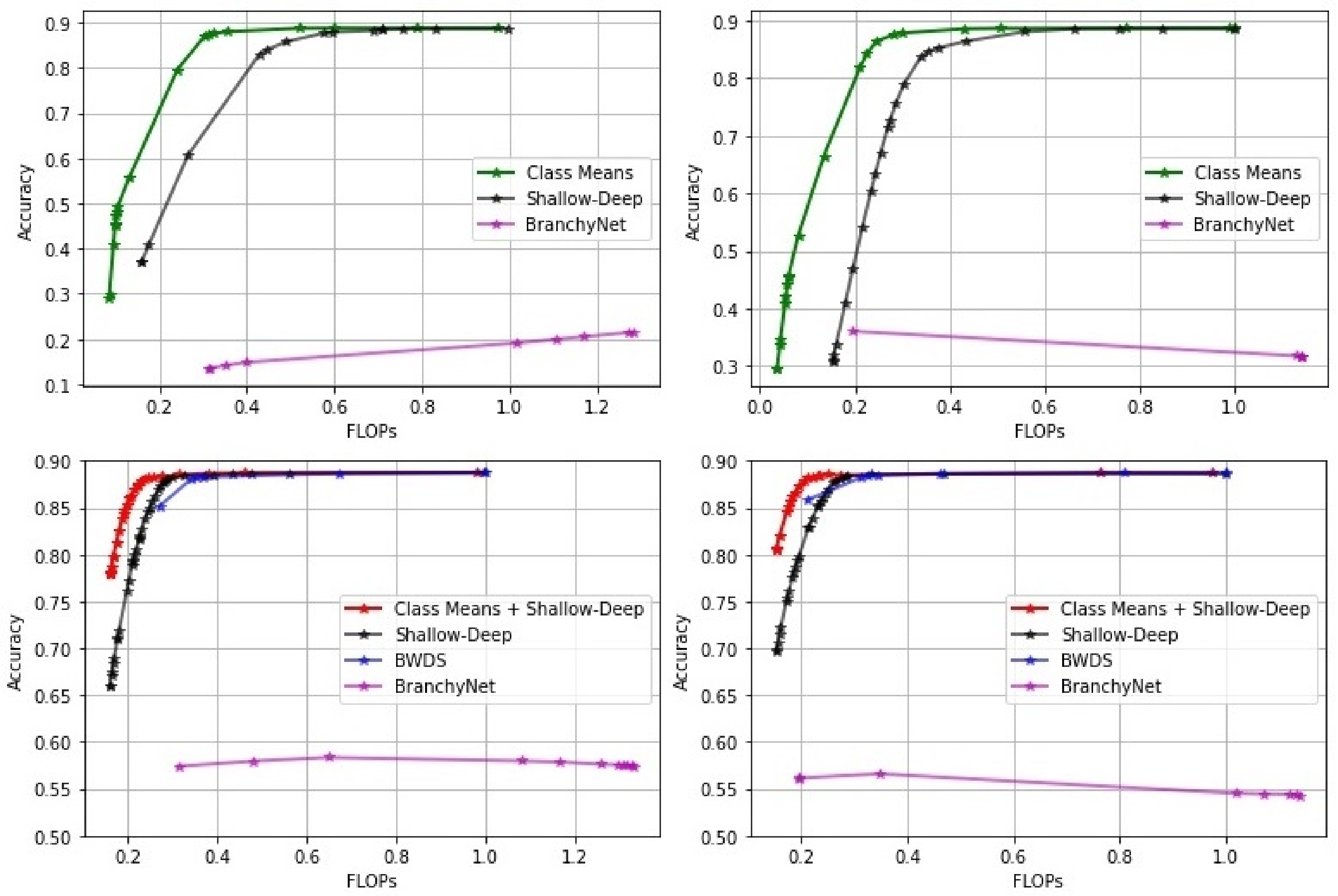

Fig. 2. Comparison of Class Means with existing methods under fixed training time budget (top: one epoch, bottom: six epochs) for ResNet-152 (left) and WideResNet-101 (right).

computational complexity may not be desirable for low-power devices. We follow the same threshold selection procedure described above. During inference, the decision of early exit is made according to both the class means and the ICs.

We train BranchyResNet-152 and BranchyWideResNet-101 for 300 and 250 epochs respectively. We use stochastic gradient descent with batch size of 256. The loss of the branches are added up to the loss of the final layer and the weighted average is taken as prescribed in [8]. We have trained multiple combination of weights, and found that $[1/6, 1/4, 1]$ achieves the best performance. For thresholds, we follow the same procedure as class means, but this time the range of values is $[0, \log K]$ where $K = 10$, because we consider the entropy of the predictions to make a decision.

Since the task of training decision functions in the BWDS method is a binary classification task (i.e., early exit or not) they converge rather quickly. We separately train the classifiers that come after the decision functions as well. We use pooling and single linear layer for these, as suggested by the authors.

As shown in Fig. 3, combining class means with internal classifiers achieves a better trade-off between the computational cost and network accuracy. For low computational budget, class means improve the accuracy by more than 2%, with less computation. We also observe that BranchyNet suffers from training the network with the branches jointly. These branches hurt the overall performance. Moreover, convolutional layers in the branches add a significant computational cost without a considerable gain in accuracy.

Although early exit mechanisms with decision functions like the BWDS method provide decent performance, experimental results show that threshold based early exit mechanisms perform better. In other words, making a decision about early exit and then performing classification fares worse than the threshold-based strategies where classification and exiting decisions are melted into the same pot.

According to Fig. 3, we can conclude that the consulting mechanism between the class means and the internal classifiers is generally beneficial. The common decision that is reached by the two classifiers can rectify possible misclassifications and avoid unnecessary computation. This can be seen as an example of ensembles, in which multiple classifiers are used to make a decision. Interestingly, our ensemble reduces the total computational cost unlike ordinary ensemble methods.

One improvement to our method may be to consider the class means exits only at specific points as in other existing early exit schemes [9], rather than after each layer. This can

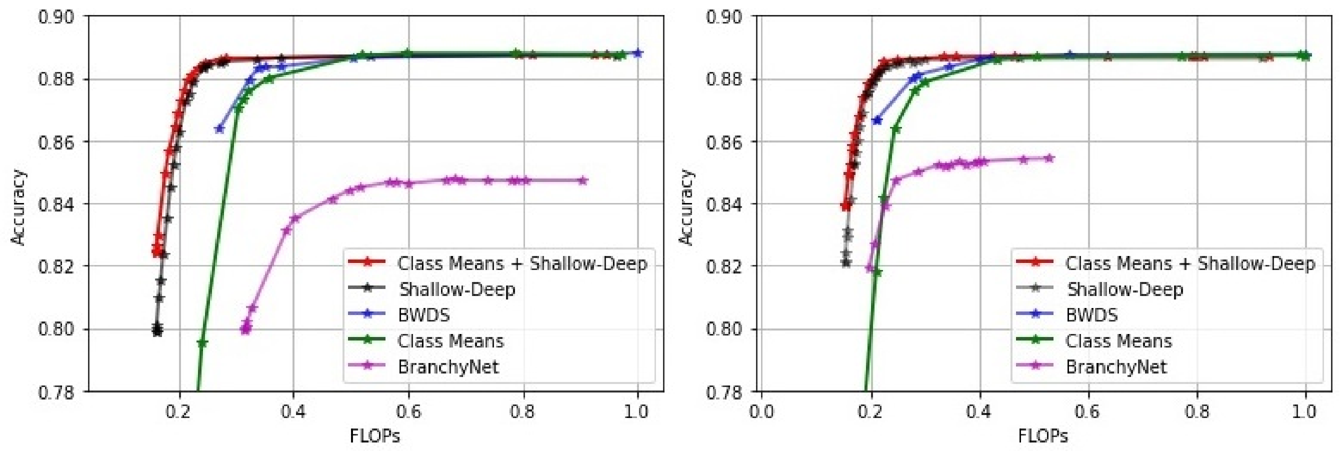

Fig. 3. Comparison of early exit methods for ResNet-152 (left) and WideResNet-101 (right).

reduce the computational overhead. However, introducing such modifications give rise to additional hyperparameters, which may be undesirable. In addition, rather than considering every channel, channels with the most distinctive properties can be used while calculating the class means. Another improvement can be made in the distance metric. Small metric learning networks can be used for better discrimination.

V. Conclusion

We propose a novel early exit mechanism based on the class means. Unlike existing early exit mechanisms, our method does not require gradient-based training, which makes it useful for network training on low-power devices. Under fixed training time budget, our method outperforms existing early exit schemes. In addition, combining class means with existing methods achieve better trade-off between the computational cost and the network accuracy.

Acknowledgment

This work was supported in part by the NSF Award CCF-1814717, and in part by an award from the University of Illinois at Chicago Discovery Partners Institute Seed Funding Program.

References

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

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

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[5] P. Li, E. Koyuncu, and H. Seferoglu, "ResPipe: Resilient model-distributed DNN training at edge networks," in *IEEE Intl. Conf. Acoustics Speech Signal Process. (ICASSP)*, Jun. 2021.

[6] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for efficient inference," in *International Conference on Machine Learning*. PMLR, 2017, pp. 527–536.

[7] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 475–480.

[8] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[9] Y. Kaya, S. Hong, and T. Dumitras, "Shallow-deep networks: Understanding and mitigating network overthinking," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3301–3310.

[10] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[11] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[12] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[13] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–18.

[14] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, "Conditional computation in neural networks for faster models," *arXiv preprint arXiv:1511.06297*, 2015.

[15] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei, "Bert loses patience: Fast and robust inference with early exit," *arXiv preprint arXiv:2006.04152*, 2020.

[16] J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "Deebert: Dynamic early exiting for accelerating bert inference," *arXiv preprint arXiv:2004.12993*, 2020.

[17] M. Biasielli, C. Bolchini, L. Cassano, E. Koyuncu, and A. Miele, "A neural network based fault management scheme for reliable image processing," *IEEE Transactions on Computers*, vol. 69, no. 5, pp. 764–776, 2020.

[18] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[19] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," *arXiv preprint arXiv:1606.04080*, 2016.

[20] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *arXiv preprint arXiv:1703.05175*, 2017.

[21] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208.