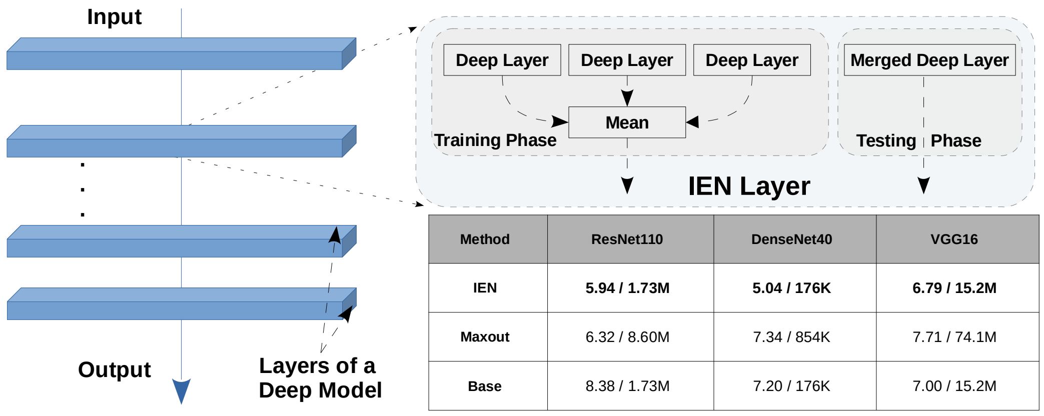
INNER ENSEMBLE NETWORKS: AVERAGE ENSEMBLE AS AN EFFECTIVE REGULARIZER

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce Inner Ensemble Networks (IENs) which reduce the variance within the neural network itself without an increase in the model complexity. IENs utilize ensemble parameters during the training phase to reduce the network variance. While in the testing phase, these parameters are removed without a change in the enhanced performance. IENs reduce the variance of an ordinary deep model by a factor of $1/m^{L-1}$, where m is the number of inner ensembles and L is the depth of the model. Also, we show empirically and theoretically that IENs lead to a greater variance reduction in comparison with other similar approaches such as dropout and maxout. Our results show a decrease of error rates between 1.7% and 17.3% in comparison with an ordinary deep model. We also show that IEN was preferred by Neural Architecture Search (NAS) methods over prior approaches.

1 INTRODUCTION



Method	ResNet110	DenseNet40	VGG16
IEN	**5.94 / 1.73M**	**5.04 / 176K**	**6.79 / 15.2M**
Maxout	6.32 / 8.60M	7.34 / 854K	7.71 / 74.1M
Base	8.38 / 1.73M	7.20 / 176K	7.00 / 15.2M

Figure 1: During the training phase, inner ensemble weights are used to achieve better performance. In the testing phase, these extra weights are no longer necessary. The final model parameters count is the same as that of the original model without additional ensemble parameters. The table is accuracy/ parameters count on CIFAR-10 dataset.

Ensemble learning (Rokach, 2010; Zhou, 2012) is based on a combination of predictions of multiple models trained over the same dataset or random subsets of the dataset to improve model performance. Ensemble methods have been widely used in deep learning (Qiu et al., 2014; Dietterich, 2000; Drucker et al., 1994; Wasay et al., 2020) to improve overall model accuracy. Combining neural networks in ensembles is known to reduce the variance in the prediction. In other words, the ensemble of networks generalizes better to unseen data than a single network (Krogh & Vedelsby, 1995; Geman et al., 1992; Zhou et al., 2002). However, there is a concern regarding the number of parameters of the final model, which involves the parameters of each constituent model. The large number of parameters in the final model decreases inference speed and increases model storage size, which can be problematic for some applications. In this paper, our objective is to find a solution that leads to an overall reduction in variance resulting from the ensembles, yet preserves the number of parameters of the underlying model.

We introduce Inner Ensemble Networks (IENs) to overcome the problem of the increased number of parameters in regular ensembles. IEN uses m instances of the same layer and applies an average operation on the output of these layers as illustrated in Figure 1. IEN can be applied to each layer of a selected base model architecture. We hypothesize that the overall variance of the model can be decomposed into sub-variances within the model layers themselves. By using IENs, we reduce the sub-variances of each layer resulting in an overall variance reduction of the model. Therefore, we primarily explore the averaging method in our setting. Now we refer to regular ensembles as outer ensembles. The performance of IENs is between that of an ordinary deep model and an outer ensemble of multiple instances of the same model.

In this work, we discuss the theoretical aspects of variance reduction resulting from applying IENs. We show how to remove the excess weights used during the training and verify our approach empirically. We also contrast IEN variance reduction with closely related approaches such as dropout (Srivastava et al., 2014) and maxout (Goodfellow et al., 2013) which gives a new perspective to these methods. When using IEN during the training phase, m times the number of parameters in the base model are required to do an inner ensemble. This could be expensive if applied to Fully Connected (FC) layers. We show empirically that using IEN on Convolutional Neural Network (CNN) layers alone results in a performance that exceeds that of a base model. This makes the training cheaper because of the shared parameters property of CNNs. We also show that using IEN on both CNNs and Fully Connected (FC) layers similarly results in a performance which exceeds that of base models, but at the cost of increased training time. Lastly, we extend IEN as a Neural Architecture Search (NAS) (Zoph & Le, 2016) cell. We applied NAS searching algorithms to discover new cells. The discovered cells show a preference for using IEN as a cell operator.

2 RELATED WORK

Ensemble methods are meta-algorithms that combine a set of independently trained networks into one predictive model in order to improve performance. Other methods include bagging and boosting which reduce predictive variance and bias (Opitz & Maclin, 1999). Averaging the results of several models is one of the most common approaches and is extensively adopted in modern deep learning practice (Russakovsky et al., 2015b; Rajpurkar et al., 2016). One variation is to create ensembles of the same model at different training epochs (Qiu et al., 2014). (Abbasian et al., 2013) proposed using inner ensembles to make learning decisions within the learning algorithms of classic machine learning techniques. Other approaches such as (Ba & Caruana, 2014) used a model compression technique, in which they trained a shallow network using an ensemble of networks. Yet, it has a high cost of first training the ensembles and then the actual shallow model unlike IENs which train a single model. (Opitz et al., 2017) developed a method that averages the output layer of a model by encouraging diversity in subsets of the output layer neurons. A main concern is the complexity of the method which is not a simple plug-in method like IENs. In IENs you simply replace the CNN or FC layers with IEN layers without the need to change anything else. Another concern in (Opitz et al., 2017) is that it forces a specific loss function unlike IENs which is agnostic from the choice of a loss function. Though inspired by ensemble methods that average independent networks, the proposed IEN structure is fundamentally different in that a) it only needs the extra ensemble weights during the training phase and b) it trains the ensembles jointly in a single model.

We could state that the idea of IEN existed implicitly in previous methods, starting from maxout (Goodfellow et al., 2013). Maxout replicates deep layers and takes the maximum of the response coming from these replicas. Maxout can be considered as an ensemble that uses max instead of average. Yet, we show both theoretically and empirically that maxout is not a desirable option and IEN is a better alternative. Dropout (Srivastava et al., 2014; Baldi & Sadowski, 2013) can be seen as a geometric mean of several sub-networks within a deep model. We believe that IEN aligns with dropout as they are both only applied during the training phase. However, IEN can be applied on every layer of the deep model unlike dropout. Dropout was originally designed to be applied on FC layers. When it comes to CNNs, dropout is applied carefully (Park & Kwak, 2016) and cannot be utilized to the fullest extent. We also show theoretically IEN variance reduction is greater than that of dropout when $m \geq 2$.

3 Inner Ensemble Nets (IENs)

Let us assume a deep model with L layers. The lth layer has a response $\mathbf{y}_l$ from an input $\mathbf{x}_l$ multiplied by a learnable weight matrix $\mathbf{w}_l$, where $\mathbf{y}_l, \mathbf{x}_l$ and $\mathbf{w}_l$ are all tensors. An ordinary deep model layer can be formulated as:

$$\mathbf{y}_l = \mathbf{w}_l \mathbf{x}_l \tag{1}$$

Thus, an IEN deep layer with m inner ensembles can be defined as:

$$\mathbf{y}_l^{\text{IEN}} = \frac{1}{m} \sum_{i \in m} \mathbf{w}_l^i \mathbf{x}_l \tag{2}$$

Where $\mathbf{w}_l^i = \{w_{kj}^i : k \in d_l, j \in n_l\} \in \mathbb{R}^{d_l \times n_l}$, $\mathbf{x}_l = \{x_i : i \in n_l\} \in \mathbb{R}^{n_l \times 1}$ and $\mathbf{y}_l^{\text{IEN}} = \{y_i : i \in d_l\} \in \mathbb{R}^{d_l \times 1}$. We notice that IEN trains a single model with multiple inner ensembles. Though IEN uses m extra parameters during the training phase, the training time does not scale as a multiplier of m. This makes IEN an economical regularization method (Appendix F).

4 IEN theoretical analysis

In this section, we start with analyzing IEN variance response. We discuss the method of removing the ensemble weights. Then we contrast maxout and dropout variance response with that of IEN.

4.1 IEN variance response

In this section we show that IEN ensembles of size $m \in \mathbb{Z}^{>1}$ decrease the overall variance of a deep net by a factor of:

$$\frac{1}{m^{L-1}}, \text{ where } L \text{ is the number of layers in the network.} \tag{3}$$

We end up with a new initialization that prevents the explosion or degradation of the gradients. To analyze a deep model variance that uses IEN we use the assumptions of (He et al., 2015a) and (Glorot & Bengio, 2010):

- $\mathbf{w}_l^i$ elements are initialized to be mutually independent with zero mean and a symmetric distribution sharing the same distribution variance.
- $\mathbf{x}_l$ elements are initialized to be mutually independent sharing the same distribution with zero mean.
- $\mathbf{w}_l$ and $\mathbf{x}_l$ are independent from each other.

The variance of each $y_l^{\text{IEN}} \in \mathbf{y}_l^{\text{IEN}}$is:

$$\text{Var}\left[y_l^{\text{IEN}}\right] = n_l \text{Var}\left[\frac{1}{m} \sum_{i \in m} w_l^i x_l\right] = n_l \frac{1}{m^2} \text{Var}\left[\sum_{i \in m} w_l^i x_l\right] \tag{4}$$

Because w_l^i and x_l are independent, which implies they are uncorrelated, we can state:

$$\text{Var}\left[y_l^{\text{IEN}}\right] = n_l \frac{1}{m^2} \sum_{i \in m} \text{Var}\left[w_l^i x_l\right] = n_l \frac{1}{m^2} \text{Var}\left[x_l\right] \sum_{i \in m} \text{Var}\left[w_l^i\right] \tag{5}$$

Because all elements of w_l^i share the same distribution parameters, their variances are equal. We can define $\text{Var}\left[w_l^i\right] = \text{Var}\left[w_l\right]$ and by their independence:

$$\text{Var}\left[y_l^{\text{IEN}}\right] = n_l \frac{1}{m} \text{Var}\left[x_l\right] \text{Var}\left[w_l\right] \tag{6}$$

By using the the results and assumptions made at Appendix A.2 given that $x_l = f\left(y_{l-1}\right)$, where f is an activation function and β^2 is a the gain of f, we have:

$$\text{Var}\left[y_l^{\text{IEN}}\right] = n_l \beta^2 \frac{1}{m} \text{Var}\left[w_l\right] \text{Var}\left[y_{l-1}\right] \tag{7}$$

For cascaded l layers and with the fact that $\text{Var}\left[y_1\right]$ represents the variance of the input layer, we arrive at:

$$\text{Var}\left[y_l^{\text{IEN}}\right] = \text{Var}\left[y_1\right]\left(\prod_{l=2}^{L}\beta^2 n_l \frac{1}{m}\text{Var}\left[w_l\right]\right) \tag{8}$$

An ordinary deep model variance response (He et al., 2015a) is:

$$\text{Var}\left[y_L\right] = \text{Var}\left[y_1\right]\left(\prod_{l=2}^{L}\beta^2 n_l \text{Var}\left[w_l\right]\right) \tag{9}$$

From equation 8 and 9, the variance of response using IEN is $\frac{1}{m^{L-1}}$ times less than using an ordinary deep model. This states that going deeper using IEN layers or wider using m will lead to better generalization of the model. We verified this empirically in the experiments section. We also want the multiplication in 8 to have a proper scalar in order to avoid reduction or magnification of the input signal, we want:

$$\frac{\beta^2 n_l}{m}\text{Var}\left[w_l\right] = 1, \forall l \tag{10}$$

So we need to initialize our weights to be:

$$w_l^i \sim \mathcal{N}\left(0, \frac{m}{\beta^2 n_l}\right) \tag{11}$$

For simplicity, we use the same initialization for the first layer.

4.2 Reverting back to the original network size

Though the IEN reduces the variance of the network, we still have the burden of the extra parameters. Because of the training mechanism of IEN, all m ensembles receive the same error signal. This results in weights that are close to each other in value, but not exactly the same because of the initialization. The most straight forward approach is to average these weights. The final weight $\widetilde{\mathbf{w}}_l$ of IEN becomes:

$$\widetilde{\mathbf{w}}_l = \frac{1}{m}\sum_{i\in m}\mathbf{w}_l^i \tag{12}$$

Thus, our IEN layer becomes: $\mathbf{y}_l^{\text{IEN}} = \widetilde{\mathbf{w}}_l\mathbf{x}_l$ which has the exact same number of parameters as before using the IEN, but with the added benefit of variance reduction. Hence, we were able to revert back to the original model size for the inference stage, yet we utilized the extra parameters of IEN during the training phase. Natural questions arise regarding whether the method of averaging the weights is applicable to maxout and whether it is applicable to m separately trained models. The answer to both is no. We verified these findings empirically in the experiments section.

4.3 Connection with dropout

Dropout (Srivastava et al., 2014) is considered as a geometric mean of several small networks (Baldi & Sadowski, 2013). Here we try to connect dropout with IEN and contrast the variance performance. We want to note that we do not introduce IEN as an alternative for dropout. It is simply a tool to be used alongside other deep learning regularization tools. Dropout can be defined as:

$$y_l^{\text{dropout}} = \delta_l w_l x_l \tag{13}$$

Where $\delta_l \sim \text{Bernoulli}\left(p\right)$ of probability p. From B, the variance of a dropout layers is upper bounded as follows:

$$\text{Var}\left[y_L^{\text{dropout}}\right] \leq \text{Var}\left[y_1\right]\left(\prod_{l=2}^{L}\beta^2 n_l \frac{1}{2}\text{Var}\left[w_l\right]\right) \tag{14}$$

Thus, dropout decreases the variance response at most:

$$\leq \frac{1}{2^{L-1}} \text{ times less than using an ordinary deep model.} \tag{15}$$

Comparing equations 15 and 3, we conclude that IEN outperforms dropout if used alone when $m > 2$.

4.4 Connection with maxout

Maxout (Goodfellow et al., 2013) is a universal approximator. It shares a common setting with IEN in terms of having multiple replicas of the weights and it selects the max response out of these replicas. One disadvantage of maxout is that the required number of parameters in both training and inference stays the same, unlike IEN with a finale averaged weight that solves the problem for inference as stated in section 4.2. A maxout layer is defined as:

$$\mathbf{y}_l^{\text{maxout}} = \max\{\mathbf{w}_l^i \mathbf{x}_l\}_i^m \tag{16}$$

Because the usage of $\max$ makes the analysis of variance much harder, we provide three views for the variance of maxout. The first one is a general upper bound (Appendix C) on the variance of maxout, which is not tight, but is helpful in our analysis. The bound is:

$$\text{Var}\left[y_L^{\text{maxout}}\right] \leq \text{Var}\left[y_1\right]\left(\prod_{l=2}^{L} n_l m \beta^2 \text{Var}\left[w_l\right]\right) \tag{17}$$

which suggests that maxout might result in an explosion in the variance by a factor of

$$m^{L-1} \text{ more than using an ordinary deep model.} \tag{18}$$

in comparison with both IEN and even an ordinary deep model. We observe this behavior in Table 1.

The second bound of maxout states the lowest possible variance (Ding et al., 2015) (Appendix D) it can achieve under an assumption that $\left(w_l^i x_l\right) \sim \mathcal{N}\left(0,1\right)$ and a linear activation function is:

$$\text{Var}\left[y_l^{\text{maxout}}\right] \geq n_l \frac{c}{\log m}, c > 0 \tag{19}$$

The IEN variance reduction gain under the later assumptions and by using equation 3 is:

$$\text{Var}\left[y_l^{\text{IEN}}\right] = n_l \frac{1}{m} \tag{20}$$

This maxout bound shows that as the number of ensembles m increases, the variance decreases, controlled by the value of c. The issue is that this bound is only for a single-layer model and it is difficult to extend it to multi-layer models. Yet it gives some insights. Depending on the value of c in equation 19, the IEN could perform better or worse. We highlight that these findings are limited and only valid for the aforementioned assumptions.

The last bound is the asymptotic bound by using extreme value theorem (Leadbetter & Rootzen, 1988) (AppendixE). When m is sufficiently large under the assumptions that $\left(w_l^i x_l\right) \sim \mathcal{N}\left(0,\sigma^2\right)$ and f is a linear activation function, the bound is:

$$\text{Var}\left[y_l^{\text{maxout}}\right] \approx n_l \frac{\pi^2}{6} \frac{\sigma^2}{2\ln m} \tag{21}$$

Suggesting that the asymptotic maxout bound is:

$$\approx \frac{\pi^2}{12\ln\left(m\right)} \text{ times less than ordinary deep model} \tag{22}$$

4.5 Summary of the variance gains

Figure 2 summarizes the variance behaviour in IEN, dropout and maxout. We assume a model with one layer to have common settings that aligns with all derived bounds and f is a linear activation function. We used equation 20 for IEN, and equation 15 for dropout and equations 19, 21 and 18 for maxout. We observe that IEN variance reduction surpasses all previous methods and the different maxout variance bounds except when c is extremely small in maxout lower variance bound. We also observe that the maxout upper bound explodes compared to IEN and dropout. Dropout variance reduction remains the same.

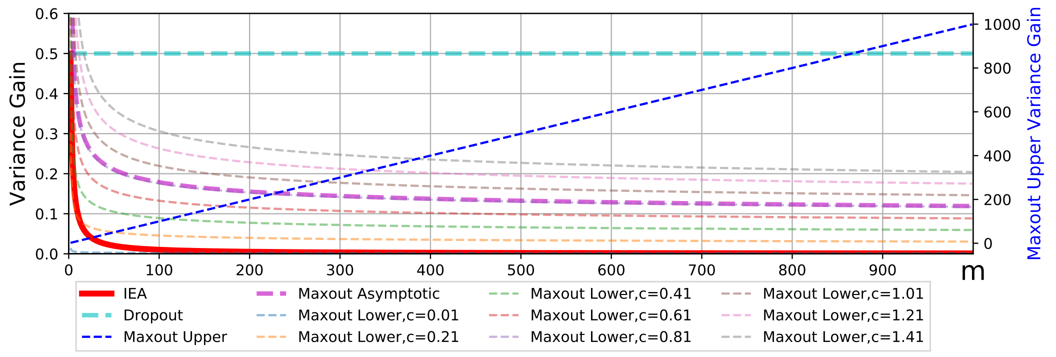


Figure 2: The variance gain of IEN, maxout and dropout versus the number of ensembles in a single linear layer model. Please note that maxout upper variance gain has a different scale.

5 Experiments & discussions

We wanted to verify that IEN generalized the deep model better than other approaches. For this we chose three well-known deep models based on the extent of their residual connections, starting from VGG (Simonyan & Zisserman, 2014) which has no residual connections, then ResNet (He et al., 2015b) which has one residual connection, and finally DenseNet (Huang et al., 2016) which is a fully residual connected net. We believe that benchmarking against these models will give an overview that be can generalized to other architectures. We use both CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009) for all of our experiments as the former shows the performance on a small classification task and the latter on a larger classification task. We also choose to experiment on image classification models as they use both CNN and FC layers. We used the *original hyper-parameters* of the models in the experiments *without any tuning in favor of our method.*

Table 1: Error rate mean and std of IEN, maxout and the original model design on different deep model architectures. The lower, the better. The subscript $\widetilde{\mathbf{w}}$ indicates results using the weight downsizing method from section 4.2. *+FC* stands for IEN applied on both FC and CNN layers. Maxout results are only on CNN layers. The *italic* titled models have the exact same number of parameters as the corresponding base models. The rest have the same number of parameters that scales with m. IEN and maxout have $m = 4$.

Dataset		ResNet56	ResNet110	DenseNet40-12	DenseNet100-12	VGG16	VGG19
CIFAR-10	$IEN_{\widetilde{\mathbf{w}}}$	**6.93±0.26**	**6.27±0.32**	7.50 ± 0.18	**5.22±0.19**	**6.80±0.18**	**6.79±0.30**
	IEN	**6.93±0.26**	**6.28±0.32**	7.5 ± 0.18	**5.22±0.19**	**6.79±0.19**	**6.79±0.31**
	$IEN+FC_{\widetilde{\mathbf{w}}}$	**6.49±0.04**	**5.94±0.26**	**6.89±0.20**	**5.04±0.07**	6.85±0.18	7.15±0.27
	IEN+FC	**6.49±0.04**	**5.94±0.26**	**6.89±0.20**	**5.04±0.07**	6.85±0.18	7.15±0.27
	$Maxout_{\widetilde{\mathbf{w}}}$	89.99±0	90±0	89.70±0.51	90±0	90±0	90±0
	Maxout	7.98±0.32	6.32±0.12	7.34±0.155	6.16±0.54	7.71±0.15	8.16±0.18
	$Base_{\widetilde{\mathbf{w}}}$	89.99	90	90	90	90	90
	Base	8.38±1.2	6.38±0.48	7.20±0.15	5.6±0.12	7.00±0.08	7.02±0.08
CIFAR-100	$IEN_{\widetilde{\mathbf{w}}}$	**29.34±0.49**	28.17±0.11	**29.37±0.30**	**23.76±0.45**	**28.49±0.39**	**29.40±0.22**
	IEN	**29.34±0.49**	28.16±0.10	**29.37±0.30**	**23.76±0.45**	**28.50±0.39**	**29.41±0.23**
	$IEN+FC_{\widetilde{\mathbf{w}}}$	**28.20±0.19**	**27.34±0.28**	**29.76±0.19**	**23.67±0.37**	29.33±0.10	31.81±0.19
	IEN+FC	**28.20±0.19**	**27.34±0.28**	**29.76±0.19**	**23.67±0.37**	29.33±0.10	31.81±0.19
	$Maxout_{\widetilde{\mathbf{w}}}$	99.01±0.01	99±0	99± 0	99 ± 0	99±0	99±0
	Maxout	31.8±1.69	29.47±0.84	30.49±0.75	28.88±5.72	31.68±0.65	34.32±0.30
	$Base_{\widetilde{\mathbf{w}}}$	98.96	99.14	99	90	99	99
	Base	29.97±0.71	27.83±0.64	29.85±0.39	24.01±0.14	29.27±0.27	30.92±0.49

5.1 IEN behavior analysis

Where to apply IEN: IEN can be applied to both CNN and FC layers. The question that arises concerns the cost of training time which is correlated with the number of parameters. Another question is whether it is better to apply IEN to just CNN layers or both CNN and FC layers. From Table 1, applying IEN to CNN layers only, $IEN_{\widetilde{\mathbf{w}}}$, yields better results than the base model as observed for ResNet, VGG an DenseNet. Also, it might not be enough to apply it only to CNN layers, as observed in DenseNet40-12 results. Yet, when it is applied to FC layers, $IEN{+}FC_{\widetilde{\mathbf{w}}}$, too it will yield better results than the base model. So, the model designer has a choice. If IEN is applied to CNN layers only, it will save training time and enhance the performance. If applied to both CNN and FC layers it will increase the training time, but will result in a better performance than the former.

Weight downsizing property: We denote the downsized model using the method from section 4.2 with subscript $\widetilde{\mathbf{w}}$. We observe from Table 1 that downsized IEN models which have the same number of parameters as the base model do have the same performance as the one with the full ensemble parameters. This property was verified across different architectures as shown in the same table. Also, it holds for applying IEN on CNN or FC layers. We also wanted to verify if this property can be extended to the base model or the maxout model. We applied $\widetilde{\mathbf{w}}$ on both of them and as observed from Table 1 their performance was extremely poor and close to random guessing. We observe that the weight downsizing property was only successful in IEN for two reasons. Firstly, the mean is a linear operator allowing the inner ensembles weights to receive the same error signal during the training phase. Secondly, the weights were initialized from the same distribution. This leads the weights to behave very similarly, resulting in closely related weights.

Connection with the variance analysis: From Table 1, we observe that IEN outperforms previous methods such as maxout and the base model. We now recall from the variance reduction gain of IEN equation 3 that going deeper leads to better results. We notice this happens in both ResNet and DenseNet. Yet, it does not hold for VGG-16 and VGG-19 which is the same case for maxout and the base model. This suggests the necessity of residual connections as in (Veit et al., 2016). Aside from VGG results, our variance analysis of IEN aligns with the empirical findings. For maxout, we notice in some cases such as ResNet56 results for CIFAR-10, that it performs better than the base model. This is related to the maxout lower bound that was discussed earlier. However, in some cases such as ResNet56 for CIFAR-100, maxout performs far worse than the base model. This aligns too with the upper and asymptotic variance bounds of maxout. The dropout results are at Appendix G.

Effect of number of ensembles m: We also wanted to study the effect of number of ensembles m on accuracy of the models. Figure 3 illustrates this affect. We notice that IEN with $m = [2, 4, 8]$ outperforms maxout with the same m. Most of the time maxout performs worse than the base model. In a few cases such as CIFAR-10 ResNet56, maxout might perform better than the base model but not better than IEN. Most of the time IEN and IEN+FC outperform the base model. Lastly, using $m = [2, 4]$ is much better than using $m = 8$. This suggests that going higher with the number of inner ensembles is not desirable. This behavior aligns with the typical behavior of an outer ensemble (Rosen, 1996).

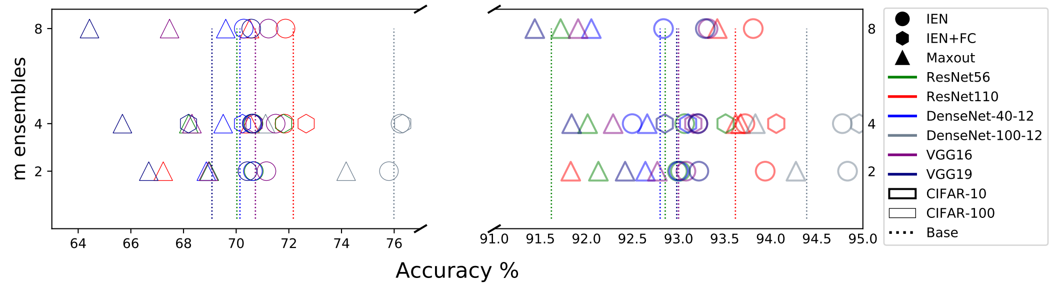


Figure 3: The effect of changing m versus model mean accuracy. IEN, maxout and base model accuracies are shown. CIFAR-10 results are the thin line shapes, while CIFAR-100 results are the thick line shapes. Colors represents different architectures. Dashed line is the base model mean accuracy.

5.2 Comparison with outer-ensemble

We denote regular ensembles as outer ensembles to distinguish them from inner ensembles. From Table 2, an outer ensemble of IENs exceeds in performance an outer ensemble of base models and maxout models using the same number of parameters except in a few cases. From Table 2 and Table 1 we notice that an outer ensemble of the base model outperforms single model IEN, but an IEN model is better than a single base model. This indicates that IEN performance stands in between a single regular model and an ensemble of the same model.

Table 2: Results of outer-ensemble of the models. Each model was trained 3 times and the predictions were averaged. The metric reported is the mean of the error rate. The lower the better. The subscript $\widetilde{\mathbf{w}}$ indicates results with the weight downsizing method from section 4.2. *+FC* stands for IEN applied on Fully Connected layer. All IENs and Maxouts have $m = 4$. The base model ensemble results are from 4 trained models.

Dataset		ResNet56	ResNet101	DenseNet40-12	DenseNet100-12	VGG16	VGG19
CIFAR-10	IEN$_{\widetilde{\mathbf{w}}}$	**5.49**	5.00	5.91	**4.02**	5.68	5.71
	IEN+FC$_{\widetilde{\mathbf{w}}}$	**5.05**	**4.95**	**5.21**	**4.04**	5.70	6.02
	Maxout	6.43	5.20	5.66	4.87	6.24	6.66
	Base	5.68	4.97	5.46	4.59	**5.61**	**5.67**
CIFAR-100	IEN$_{\widetilde{\mathbf{w}}}$	24.38	23.32	24.85	**19.65**	**24.60**	**25.33**
	IEN+FC$_{\widetilde{\mathbf{w}}}$	**23.62**	23.03	**24.44**	**19.11**	25.44	27.44
	Maxout	26.34	24.12	24.72	24.65	26.89	29.29
	Base	24.13	**22.26**	24.85	19.86	24.76	26.02

5.3 Extension to NAS

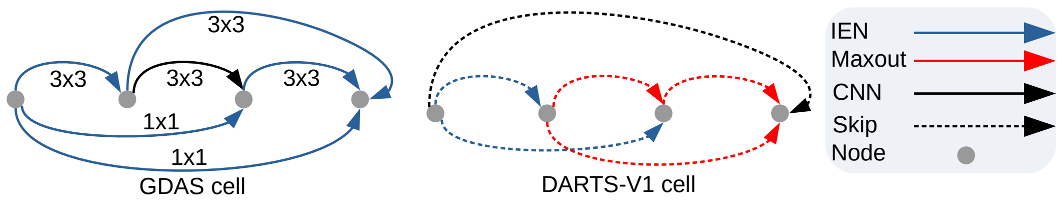


Figure 4: Cells discovered using NAS DARTS-V1 and GDAS search algorithms. 1×1 and 3×3 are the CNN kernel sizes. IEN and maxout have $m = 4$. The cells are acyclic graphs. The dashed arrows are skip connections. Colors represent different approaches.

In order to make sure that IEN is preferred in deep model design, we tested it using NAS search algorithms. We introduced IEN and maxout as operators in both DARTS-V1 (Liu et al., 2018) and GDAS (Dong & Yang, 2019) search algorithms besides regular CNNs. We ran NAS-Bench-201 (Dong & Yang, 2020) on both DARTS and GDAS to discover the cells. We report results on CIFAR-10, CIFAR-100, and ImageNet-16-120. ImageNet-16-120 is a subset of ImageNet (Russakovsky et al., 2015a) that is used by NAS-Bench-201. From Figure 4 GDAS shows a high preference towards using IENs. DARTS-V1 resulted in a mix between maxout and IEN and used IEN in the early stage of the cell. This early usage of IEN might correlate with the importance of reducing the variance in the early layers of a deep model. Also, DARTS resulted in a skip-connect cell which is a known bias in DARTS algorithm (Dong & Yang, 2020). Table 3 shows that our discovered cell which depends on IEN resulted in a better performance than cells that use regular CNNs.

Table 3: Accuracies of four cell models based on NAS-Bench-201 configuration, using our discovered cells and the original NAS-Bench-201 cells.

NAS Search Method	Cell	CIFAR-10	CIFAR-100	ImageNet-16-120
DARTS-V1	**Ours**	**41.28**	**17.36**	**16.77**
	NAS-Bench-201 Cell	39.77	15.03	16.43
GDAS	**Ours**	**91.26±0.11**	**72.68±0.12**	**47.6±0.33**
	NAS-Bench-201 Cell	89.89±0.08	71.34±0.04	41.59± 1.33

6 Conclusion

We presented IENs which enhance the performance of a deep model by utilizing the concept of ensembles while keeping the model parameters free from the excess ensemble weights. We theoretically analyzed the variance reduction gain of IEN and other approaches with respect to ordinary deep model variance. We empirically showed that IENs outperformed other methods when applied on known deep model architectures. We analyzed IEN behavior versus that of outer ensembles and extended it to NAS with the discovery of new cells.

7 Reviews Discussion

Table 4: To *Reviewer#1*: Mean error rate and std reported. Different variations are discussed.

Dataset	Variation	ResNet56	ResNet110	DenseNet40-12	DenseNet100-12
CIFAR10	**IEN (ours)**	**6.49±0.04**	**5.94±0.32**	**6.89±0.20**	**5.04±0.07**
	$\text{lr} = 1/m, w = \sum_j w_j$	89.29±0.03	72.66±1.46	8.52±0.21	6.42±0.08
	$\text{lr} = \text{lr}, w = 1/m \sum_j w_j$	92.3±0.36	90.9±0.06	7.33±0.16	5.29±0.21
	$\text{lr} = \text{lr}, w = \sum_j w_j$	93.08±0.35	84.48±0.74	7.10±0.05	5.07±0.15
CIFAR100	**IEN (ours)**	**28.20±0.19**	**27.34±0.28**	**29.37±0.19**	**23.67±0.37**
	$\text{lr} = 1/m, w = \sum_j w_j$	64.23±0.22	40.27±0.38	32.22±0.44	28.55±0.09
	$\text{lr} = \text{lr}, w = 1/m \sum_j w_j$	69.86±0.31	66.7±0.25	29.48±0.67	23.84±0.25
	$\text{lr} = \text{lr}, w = \sum_j w_j$	70.81±0.15	56.44±0.11	**29.30±0.53**	24.12±0.04

Table 5: To *Reviewer#4*: Mean accuracy on CIFAR-10C and CIFAR-100C datasets (Hendrycks & Dietterich, 2019).

	CIFAR	Arch	Blur				Weather				Digital				Noise		
			Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG	Gauss.	Shot	Impulse
DenseNet100-12	10C	IEN	**82.43**	**56.90**	**78.81**	75.84	**81.95**	**76.73**	**87.31**	**92.61**	**81.06**	**83.12**	**71.30**	76.09	41.11	54.06	51.56
		Maxout	81.74	50.50	77.71	**77.35**	75.77	70.82	84.69	90.32	76.09	82.72	68.19	**78.48**	**52.13**	**62.43**	**54.20**
		Base	79.48	54.92	75.78	71.33	79.95	75.05	86.28	91.85	78.53	81.99	68.68	76.05	38.50	51.42	47.89
	100C	IEN	**56.77**	15.54	48.38	**49.50**	51.64	42.83	61.56	70.72	53.76	55.15	**48.05**	43.34	15.26	23.18	19.10
		Maxout	54.07	16.46	46.75	47.84	46.98	37.54	54.77	66.63	42.18	55.15	44.07	**47.58**	**18.71**	**26.27**	**20.42**
		Base	**56.98**	19.35	50.47	49.06	53.54	45.82	62.81	71.31	**54.10**	56.06	48.12	45.28	14.74	22.27	19.58
ResNet110	10C	IEN	81.30	46.96	75.29	75.22	78.79	75.00	**86.98**	**92.42**	**76.15**	82.21	72.47	77.7	46.95	58.55	56.96
		Maxout	**82.20**	51.96	**76.90**	**77.63**	78.95	**76.97**	85.96	91.91	73.06	**83.54**	**76.57**	**82.11**	**56.57**	**66.35**	**59.66**
		Base	81.36	**56.24**	**76.45**	76.09	**79.6**	76.22	85.45	91.03	73.28	82.6	74.81	80.31	51.50	61.91	58.75
	100C	IEN	**56.21**	22.56	50.27	49.28	49.86	44.94	**58.83**	**67.14**	**49.05**	55.22	48.20	45.90	20.28	27.80	25.55
		Maxout	**56.67**	**32.28**	**51.45**	**51.87**	**52.24**	**50.04**	54.34	65.62	44.61	**57.21**	**53.48**	**55.72**	**31.95**	**39.49**	**31.44**
		Base	**56.21**	24.67	51.18	49.75	50.08	45.73	57.94	66.80	48.56	55.59	48.86	48.21	22.63	30.33	26.26

Table 6: To *Reviewer#4*: Comparison with SWA (Izmailov et al., 2018).

DNN(Budget)	SGD	SGD + IEN	SWA 1.5 Budgets	SWA 1.5 Budgets + IEN
PreResNet110(150)	76.77±0.38	**77.05±0.14**	79.10±0.21	**79.19±0.25**
WideResNet28x10 (200)	80.82±0.23	**81.47±0.14**	82.15±0.27	**83.45±0.13**

Table 7: To *Reviewer#3* Avg pool layer applied after each CNN, same place where IEN was applied.

Dataset		IEN	Base	Avg Pool
CIFAR-10	ResNet56	**6.93±0.26**	8.38±1.2	12.61±0.90
	ResNet110	**6.27±0.32**	6.38±0.48	14.58±0.05
CIFAR-100	ResNet56	**29.34±0.49**	29.97±0.71	39.93±1.80
	ResNet110	**28.17±0.11**	27.83±0.64	41.75±2.21

Table 8: To *Reviewer#3 & #2*. Accuracy on CIFAR-10 dataset.

	Accuracy	Accuracy Increase from Baseline
Baseline of (Opitz et al., 2017)	80.86	-
DivLoss of (Opitz et al., 2017) on their Baseline	82.30	1.70%
Negative Correlation of (Liu & Yao, 1999) on (Opitz et al., 2017) Baseline	80.90	0.03%
Reproduced Baseline of (Opitz et al., 2017) by us	80.50	-
IEN (ours) on the reproduced Baseline	**82.71**	**2.74%**

REFERENCES

Kaiming he initialization. `https://pytorch.org/docs/stable/nn.init.html#torch.nn.init.calculate_gain`. Accessed: 2020-06-10.

Houman Abbasian, Chris Drummond, Nathalie Japkowicz, and Stan Matwin. Inner ensembles: Using ensemble methods inside the learning algorithm. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 33–48. Springer, 2013.

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pp. 2654–2662, 2014.

Pierre Baldi and Peter J Sadowski. Understanding dropout. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2814–2822. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/4878-understanding-dropout.pdf`.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.

Jian Ding, Ronen Eldan, Alex Zhai, et al. On multiple peaks and moderate deviations for the supremum of a gaussian field. *The Annals of Probability*, 43(6):3468–3493, 2015.

X. Dong and Y. Yang. Searching for a robust neural architecture in four gpu hours. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1761–1770, 2019.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020. URL `https://openreview.net/forum?id=HJxyZkBKDr`.

Harris Drucker, Corinna Cortes, Lawrence D Jackel, Yann LeCun, and Vladimir Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6(6):1289–1301, 1994.

emcor (https://stats.stackexchange.com/users/48639/emcor). Extreme value theory - show: Normal to gumbel. Cross Validated. URL `https://stats.stackexchange.com/q/105745`. URL:https://stats.stackexchange.com/q/105745 (version: 2015-05-17).

Cheng-Yang Fu. pytorch-vgg-cifar10. `https://github.com/chengyangfu/pytorch-vgg-cifar10`. Accessed: 2020-05-26.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `http://proceedings.mlr.press/v9/glorot10a.html`.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pp. 1026–1034, USA, 2015a. IEEE Computer Society. ISBN 9781467383912. doi: 10.1109/ICCV.2015.123. URL `https://doi.org/10.1109/ICCV.2015.123`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015b. URL `http://arxiv.org/abs/1512.03385`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL `http://arxiv.org/abs/1608.06993`.

Yerlan Idelbayev. Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch. `https://github.com/akamaster/pytorch_resnet_cifar10`. Accessed: 20xx-xx-xx.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in neural information processing systems*, pp. 231–238, 1995.

MR Leadbetter and Holger Rootzen. Extremal theory for stochastic processes. *The Annals of Probability*, pp. 431–478, 1988.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. *CoRR*, abs/1806.09055, 2018. URL `http://arxiv.org/abs/1806.09055`.

Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural networks*, 12(10): 1399–1404, 1999.

David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

Michael Opitz, Horst Possegger, and Horst Bischof. Efficient model averaging for deep neural networks. In Shang-Hong Lai, Vincent Lepetit, Ko Nishino, and Yoichi Sato (eds.), *Computer Vision – ACCV 2016*, pp. 205–220, Cham, 2017. Springer International Publishing. ISBN 978-3-319-54184-6.

Sungheon Park and Nojun Kwak. Analysis on the dropout effect in convolutional neural networks. In *Asian conference on computer vision*, pp. 189–204. Springer, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Geoff Pleiss, Danlu Chen, Gao Huang, Tongcheng Li, Laurens van der Maaten, and Kilian Q Weinberger. Memory-efficient implementation of densenets. *arXiv preprint arXiv:1707.06990*, 2017.

Xueheng Qiu, Le Zhang, Ye Ren, Ponnuthurai N Suganthan, and Gehan Amaratunga. Ensemble deep learning for regression and time series forecasting. In *Computational Intelligence in Ensemble Learning (CIEL), 2014 IEEE Symposium on*, pp. 1–6. IEEE, 2014.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

Bruce E Rosen. Ensemble learning using decorrelated neural networks. *Connection science*, 8(3-4): 373–384, 1996.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015a. doi: 10.1007/s11263-015-0816-y.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015b.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

user_lambda (https://math.stackexchange.com/users/338905/user_lambda). Distribution of the maximum of a large number of normally distributed random variables. Mathematics Stack Exchange. URL `https://math.stackexchange.com/q/2035079`. URL:https://math.stackexchange.com/q/2035079 (version: 2016-11-29).

Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, pp. 550–558, 2016.

Abdul Wasay, Brian Hentschel, Yuze Liao, Sanyuan Chen, and Stratos Idreos. Mothernets: Rapid deep ensemble learning. In *Proceedings of the 3rd MLSys Conference (MLSys)*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. URL `http://arxiv.org/abs/1611.01578`.

Appendix

Contents

A LEMMAS

A.1 VARIANCE OF THE PRODUCT OF INDEPENDENT RANDOM VARIABLES

if $y_1, y_2, ..., y_n$ are independent random variables, then:

$$\text{Var}\left[y_1 * y_2 ... y_n\right] = \prod_{i=1}^{n}\left(\text{Var}\left[y_i\right] + \mathbb{E}\left[y_i\right]^2\right) - \prod_{i=1}^{n}\mathbb{E}\left[y_i\right]^2 \tag{23}$$

A.2 VARIANCE GAIN OF A NON-LINEAR ACTIVATION FUNCTION

Let y be a random variable, $x = f(y)$, where f is a nonlinear activation function, one can state that:

$$\text{Var}\left[x\right] = \int_{-\infty}^{\infty} f(y)^2 \mathbb{P}(y)\, dy \tag{24}$$

By using first order Taylor series expansion:

$$\text{Var}\left[x\right] \approx \int_{-\infty}^{\infty} \left(f(y_0) + f'(y_0)(y - y_0)\right)^2 \mathbb{P}(y)\, dy \tag{25}$$

Where $y_0 = 0$, then:

$$\text{Var}\left[x\right] \approx \int_{-\infty}^{\infty} f(0)^2 \mathbb{P}(y)\, dy + \int_{-\infty}^{\infty} f'(0)^2 y^2 \mathbb{P}(y)\, dy + 2\int_{-\infty}^{\infty} f(0) f'(0)\, y \mathbb{P}(y)\, dy \tag{26}$$

While Equation 26 is valid only for smoothness C1 functions it also works reasonably on non C1 functions like ReLU= $\max(0, y)$. If we choose $x = \max(0, y)$ aka a ReLU function, we have:

$$x'(y) = \begin{cases} 1 & \text{if } y >= 0 \\ 0 & \text{if } y < 0 \end{cases} \tag{27}$$

By using Equation 26 and by assuming x has zero mean with symmetric distribution:

$$\text{Var}\left[x\right] \approx \int_{-\infty}^{\infty} f'(0)^2 y^2 \mathbb{P}(y)\, dy = \int_{0}^{\infty} y^2 \mathbb{P}(y)\, dy \tag{28}$$

Thus, we can state:

$$\text{Var}\left[x\right] \approx \frac{1}{2}\int_{-\infty}^{\infty} y^2 \mathbb{P}(y)\, dy = \beta^2 \text{Var}\left[y\right] \tag{29}$$

We denote the gain $\frac{1}{2}$as β^2. Same procedure can be applied to different activation functions. A full table of different gains for commonly used deep model activation function can be found at (pyt).

In case x has a non-zero mean, we find the gain β^2 for a ReLU activation:

$$\text{Var}\left[x\right] = \int_{-\infty}^{\infty} x^2 \mathbb{P}(x)\, dx - \left(\int_{-\infty}^{\infty} x \mathbb{P}(x)\, dx\right)^2 \tag{30}$$

$$\text{Var}\left[x\right] = \int_{-\infty}^{\infty} f(y)^2 \mathbb{P}(y)\, dy - \left(\int_{-\infty}^{\infty} f(y) \mathbb{P}(y)\, dy\right)^2 \tag{31}$$

By using first order Taylor series expansion and $y_0 = 0$:

$$\text{Var}\left[x\right] \approx \int_{-\infty}^{\infty} f(0)^2 \mathbb{P}(y)\, dy + \int_{-\infty}^{\infty} f'(0)^2 y^2 \mathbb{P}(y)\, dy + 2\int_{-\infty}^{\infty} f(0) f'(0)\, y \mathbb{P}(y)\, dy - \left(\int_{-\infty}^{\infty} f(y) \mathbb{P}(y)\, dy\right)^2$$

By using ReLU equation 27:

$$\text{Var}\left[x\right] \approx \int_{-\infty}^{\infty} f'(0)^2 y^2 \mathbb{P}(y)\, dy - \left(\int_{-\infty}^{\infty} f(y) \mathbb{P}(y)\, dy\right)^2$$

Suppose

$$\int_{-\infty}^{\infty} f'(0)^2 y^2 \mathbb{P}(y)\, dy = \beta^2 \int_{-\infty}^{\infty} y^2 \mathbb{P}(y)\, dy,\ \beta^2 \leq 1$$

By using the characteristic of ReLU that:

$$\int_{-\infty}^{\infty} f(y)\, \mathbb{P}(y)\, dy \geq \int_{-\infty}^{\infty} y \mathbb{P}(y)\, dy$$

Then:

$$\text{Var}[x] \leq \beta^2 \left(\int_{-\infty}^{\infty} y^2 \mathbb{P}(y)\, dy - \left(\int_{-\infty}^{\infty} y \mathbb{P}(y)\, dy \right)^2 \right) = \beta^2 \text{Var}[y]$$

B Dropout variance response

$$\mathbf{y}_l^{\text{dropout}} = \Delta_l \circ \mathbf{w}_l \mathbf{x}_l$$

Assumptions:

- $\mathbf{w}_l = \{w_{ji} : j \in d_l, i \in n_l\} \in \mathbb{R}^{d_l \times n_l}$ elements are initialized to be mutually independent with zero mean and a symmetric distribution
- $\mathbf{x}_l = \{x_i : i \in n_l\} \in \mathbb{R}^{n_l \times 1}$ elements are initialized to be mutually independent sharing the same distribution with zero mean
- $\Delta_l = \{\delta_i : i \in d_l\} \in \mathbb{R}^{d_l \times 1}$ elements are Bernoulli δ_l mutually independent random variables which are multiples element wise by $\mathbf{w}_l$
- $\mathbf{w}_l$, Δ_l and $\mathbf{x}_l$ are independent from each other
- $\mathbf{y}_l^{\text{dropout}} = \{y_i : i \in n\} \in \mathbb{R}^{d_l \times 1}$

The variance of each $y_l^{\text{dropout}} \in \mathbf{y}_l^{\text{dropout}}$, where l is the layer number, is:

$$\text{Var}\left[y_l^{\text{dropout}}\right] = n_l \text{Var}[\delta_l w_l x_l] \tag{32}$$

By using the lemma in A.1:

$$\begin{aligned}\text{Var}\left[y_l^{\text{dropout}}\right] &= n_l \left[\left(\text{Var}[\delta_l] + \mathbb{E}[\delta_l]^2\right)\left(\text{Var}[w_l] + \mathbb{E}[w_l]^2\right)\left(\text{Var}[x_l] + \mathbb{E}[x_l]^2\right) - \left(\mathbb{E}[\delta_l]^2\right)\left(\mathbb{E}[w_l]^2\right)\left(\mathbb{E}[x_l]^2\right) \right] \\ &= n_l \left[\left(\text{Var}[\delta_l] + \mathbb{E}[\delta_l]^2\right)\left(\text{Var}[w_l] + \cancelto{0}{\mathbb{E}[w_l]^2}\right)\left(\text{Var}[x_l] + \cancelto{0}{\mathbb{E}[x_l]^2}\right) - \left(\mathbb{E}[\delta_l]^2\right)\left(\cancelto{0}{\mathbb{E}[w_l]^2}\right)\left(\cancelto{0}{\mathbb{E}[x_l]^2}\right) \right]\end{aligned}$$

Thus,

$$\text{Var}\left[y_l^{\text{dropout}}\right] = n_l \left(\text{Var}[\delta_l] + \mathbb{E}[\delta_l]^2\right) \text{Var}[w_l]\, \text{Var}[x_l] \tag{33}$$

By using the the results and assumption made at A.2 given that $x_l = f(y_{l-1})$, we have:

$$\text{Var}\left[y_l^{\text{dropout}}\right] = n_l \beta^2 \left(\text{Var}[\delta_l] + \mathbb{E}[\delta_l]^2\right) \text{Var}[w_l]\, \text{Var}[y_{l-1}] \tag{34}$$

For cascaded l layers and with the fact that $\text{Var}[y_1]$ represents the variance of the input layer, we arrive at:

$$\text{Var}\left[y_L^{\text{dropout}}\right] = \text{Var}[y_1] \left(\prod_{l=2}^{L} \beta^2 n_l \left(\text{Var}[\delta_l] + \mathbb{E}[\delta_l]^2\right) \text{Var}[w_l] \right) \tag{35}$$

For a Bernoulli random variable the maximum $\mathbb{E}[\delta_l] = \frac{1}{2}$ with $\text{Var}[\delta_l] = \frac{1}{4}$, then we can upper bound the $\text{Var}\left[y_L^{\text{dropout}}\right]$ to be:

$$\text{Var}\left[y_L^{\text{dropout}}\right] \leq \text{Var}[y_1] \left(\prod_{l=2}^{L} \beta^2 n_l \left(\frac{1}{4} + \left(\frac{1}{2}\right)^2\right) \text{Var}[w_l] \right) \tag{36}$$

$$\text{Var}\left[y_L^{\text{dropout}}\right] \leq \text{Var}\left[y_1\right]\left(\prod_{l=2}^{L}\beta^2 n_l \frac{1}{2}\text{Var}\left[w_l\right]\right) \tag{37}$$

C Maxout variance upper bound

$$\mathbf{y}_l^{\text{maxout}} = \max\{\mathbf{w}_l^i \mathbf{x}_l\}_i^m$$

Assumptions:

- $\mathbf{w}_l^i = \{w_{kj}^i : k \in d_l, j \in n_l\} \in \mathbb{R}^{d_l \times n_l}$ elements are initialized to be mutually independent with zero mean and a symmetric distribution
- $\mathbf{x}_l = \{x_i : i \in n_l\} \in \mathbb{R}^{n_l \times 1}$ elements are initialized to be mutually independent sharing the same distribution with zero mean
- $\mathbf{w}_l$ and $\mathbf{x}_l$ are independent from each other
- $\mathbf{y}_l^{\text{maxout}} = \{y_i : i \in d_l\} \in \mathbb{R}^{d_l \times 1}$

The variance of each $y_l^{\text{maxout}} \in \mathbf{y}_l^{\text{maxout}}$, where l is the layer number, is:

$$\text{Var}\left[y_l^{\text{maxout}}\right] = n_l \text{Var}\left[\max\{w_l^i x_l\}_i^m\right] \tag{38}$$

This can be upper bounded using the fact the $\text{Var}\left[\max\left(z_i\right)_i^m\right] \leq \text{Var}\left[\sum_{i \in m} z_i\right]$, applying this bound:

$$\text{Var}\left[y_l^{\text{maxout}}\right] \leq n_l \text{Var}\left[\sum_{m \in i}\{w_l^i x_l\}\right] = n_l \sum_{m \in i}\text{Var}\left[w_l^i x_l\right] \tag{39}$$

Because w_l^i and x_l are both independent and by using lemma A.1:

$$\begin{aligned}
\text{Var}\left[y_l^{\text{maxout}}\right] &\leq n_l \sum_{m \in i}\left[\left(\text{Var}\left[w_l^i\right] + \mathbb{E}\left[w_l^i\right]^2\right)\left(\text{Var}\left[x_l\right] + \mathbb{E}\left[x_l\right]^2\right) - \left(\mathbb{E}\left[w_l^i\right]^2\right)\left(\mathbb{E}\left[x_l\right]^2\right)\right] \\
&= n_l \sum_{m \in i}\left[\left(\text{Var}\left[w_l^i\right] + \cancelto{0}{\mathbb{E}\left[w_l^i\right]^2}\right)\left(\text{Var}\left[x_l\right] + \cancelto{0}{\mathbb{E}\left[x_l\right]^2}\right) - \left(\cancelto{0}{\mathbb{E}\left[w_l^i\right]^2}\right)\left(\cancelto{0}{\mathbb{E}\left[x_l\right]^2}\right)\right] \\
&= n_l \sum_{m \in i}\text{Var}\left[w_l^i\right]\text{Var}\left[x_l\right]
\end{aligned}$$

By using the the results and assumption made at A.2 given that $x_l = f\left(y_{l-1}\right)$, we have:

$$\text{Var}\left[y_l^{\text{maxout}}\right] \leq n_l \beta^2 \text{Var}\left[y_{l-1}\right]\sum_{m \in i}\text{Var}\left[w_l^i\right] \tag{40}$$

Because all elements of w_l^i share the same distribution, their variance is equal, we can donate $\text{Var}\left[w_l^i\right] = \text{Var}\left[w_l\right]$ and by their independence:

$$\text{Var}\left[y_l^{\text{maxout}}\right] \leq n_l m \beta^2 \text{Var}\left[y_{l-1}\right]\text{Var}\left[w_l\right] \tag{41}$$

For cascaded l layers and with the fact that $\text{Var}\left[y_1\right]$ represents the variance of the input layer, we arrive at:

$$\text{Var}\left[y_L^{\text{maxout}}\right] \leq \text{Var}\left[y_1\right]\left(\prod_{l=2}^{L} n_l m \beta^2 \text{Var}\left[w_l\right]\right) \tag{42}$$

D MAXOUT VARIANCE LOWER BOUND

$$\mathbf{y}_l^{\text{maxout}} = \max\{\mathbf{w}_l^i \mathbf{x}_l\}_i^m$$

Assumptions:

- $\mathbf{w}_l^i = \{w_{kj}^i : k \in d_l, j \in n_l\} \in \mathbb{R}^{d_l \times n_l}$ elements are initialized to be mutually independent with zero mean and a symmetric distribution
- $\mathbf{x}_l = \{x_i : i \in n_l\} \in \mathbb{R}^{n_l \times 1}$ elements are initialized to be mutually independent sharing the same distribution with zero mean
- $\mathbf{w}_l$ and $\mathbf{x}_l$ are independent from each other
- $\left(w_l^i x_l\right) \sim$ iid $\mathcal{N}\left(0, 1\right)$
- $\mathbf{y}_l^{\text{maxout}} = \{y_i : i \in d_l\} \in \mathbb{R}^{d_l \times 1}$

From (Ding et al., 2015) for $i \in m$ the Var $\left[\max_i z_i\right] \geq \frac{c}{\log m}$, where $z_i \sim \mathcal{N}\left(0, 1\right)$ and $c > 0$. Then:

$$\text{Var}\left[y_l^{\text{maxout}}\right] \geq n_l \frac{c}{\log m} \tag{43}$$

E MAXOUT VARIANCE ASYMPTOTIC BOUND

$$\mathbf{y}_l^{\text{maxout}} = \max\{\mathbf{w}_l^i \mathbf{x}_l\}_i^m$$

Assumptions:

- $\mathbf{w}_l^i = \{w_{kj}^i : k \in d_l, j \in n_l\} \in \mathbb{R}^{d_l \times n_l}$ elements are initialized to be mutually independent with zero mean and a symmetric distribution
- $\mathbf{x}_l = \{x_i : i \in n_l\} \in \mathbb{R}^{n_l \times 1}$ elements are initialized to be mutually independent sharing the same distribution with zero mean
- $\mathbf{w}_l$ and $\mathbf{x}_l$ are independent from each other
- $\left(w_l^i x_l\right) \sim$ iid $\mathcal{N}\left(0, \sigma^2\right)$
- $\mathbf{y}_l^{\text{maxout}} = \{y_i : i \in d_l\} \in \mathbb{R}^{d_l \times 1}$

The variance of each $y_l^{\text{maxout}} \in \mathbf{y}_l^{\text{maxout}}$, where l is the layer number, is:

$$\text{Var}\left[y_l^{\text{maxout}}\right] = n_l \text{Var}\left[\max\{w_l^i x_l\}_i^m\right] \tag{44}$$

For more details about this proof please check (user_lambda , https://math.stackexchange.com/users/338905/user_lambda; emcor , https://stats.stackexchange.com/users/48639/emcor; Leadbetter & Rootzen, 1988).

Recall that the standard normal CDF Φ is such that:

$$1 - \Phi\left(y\right) \sim \frac{e^{-y^2/2}}{y\sqrt{2\pi}} \tag{45}$$

when $y \to \infty$ and that, for every y:

$$P\left(Y_n \leq \mu + y\sigma\right) = \Phi\left(y\right)^m \tag{46}$$

Hence, if y_m is chosen such that:

$$m\frac{e^{-y_n^2/2}}{y_n\sqrt{2\pi}} = t \tag{47}$$

for some given t, then:

$$P\left(Y_m \leq \mu + y_m\sigma\right) \to e^{-t} \tag{48}$$

Solving this for y_m yields:

$$y_m = \sqrt{2\ln m} - \frac{\ln\ln m + 2\ln t + \ln\left(4\pi\right)}{2\sqrt{2\ln m}} \tag{49}$$

Then, we can define:

$$Z_m = \sqrt{2 \ln m} \frac{Y_m - \mu}{\sigma} - 2 \ln m + \frac{1}{2} \ln \ln m + \frac{1}{2} \ln (4\pi) \tag{50}$$

where, for every real z:

$$P(Z \leq z) = \exp\left(-e^{-z}\right) = G(\mu = 0, \beta = 1) \text{ a standard Gumbel distribution.} \tag{51}$$

Which aligns with the convergence from extreme value theorem. From Equation 50, for $\mu = 0$, we have and for sufficiently large m:

$$Y_m \approx \frac{\sigma}{\sqrt{2 \ln m}} \left[Z_m + 2 \ln m - \frac{1}{2} \ln \ln m - \frac{1}{2} \ln (4\pi) \right] \tag{52}$$

Hence:

$$\text{Var}[Y_m] \approx \text{Var}\left[\frac{\sigma}{\sqrt{2 \ln m}} \left[Z_m + 2 \ln m - \frac{1}{2} \ln \ln m - \frac{1}{2} \ln (4\pi) \right] \right] \tag{53}$$

$$\text{Var}[Y_m] \approx \frac{\sigma^2}{2 \ln m} \text{Var}[Z_m] \tag{54}$$

A standard Gumbel distribution have a variance of $\frac{\pi^2}{6}$, then:

$$\text{Var}[Y_m] \approx \frac{\pi^2}{6} \frac{\sigma^2}{2 \ln m} \tag{55}$$

In our case $Y_m = y_l^{\text{maxout}}$ and $\text{Var}[w_l x_l] = \sigma^2$, yielding:

$$\text{Var}\left[y_l^{\text{maxout}}\right] \approx n_l \frac{\pi^2}{6} \frac{\text{Var}[w_l x_l]}{2 \ln m} = n_l \frac{\pi^2}{6} \frac{\sigma^2}{2 \ln m} \tag{56}$$

F Models training time and parameters count

Table 9: Model parameters count / Average training time per epoch in seconds. M stands for Million, K stands for Thousand. All models were trained on Nvidia V100 GPU. Models in *italic* title have the same size at testing time. *+FC* stands for IEN applied on both FC and CNN layers. Maxout results are only on CNN layers. All models have $m = 4$

Dataset		ResNet56	ResNet110	DenseNet40-12	DenseNet100-12	VGG16	VGG19
CIFAR-10	*IEN*	4.25M / 55.16	8.60M / 97.17	854.94K / 102.79	3.74M / 239.49	74.11M / 26.62	100.66M / 32.34
	IEN+FC	4.25M / 55.9	8.61M / 105.38	860.26K / 101.01	3.75M / 244.74	76.23M / 27.46	102.79M / 32.93
	Maxout	4.25M / 57.45	8.60M / 110.53	854.94K / 106.33	3.74M / 244.74	74.11M / 28.20	100.66M / 33.98
	Base	853.02K / 17.16	1.73M / 42.01	176.12K / 51.96	769.16K / 115.16	15.25M / 7.61	20.57M / 9.21
CIFAR-100	*IEN*	4.25M / 58.56	8.61M / 94.75	866.91K / 96.13	3.77M / 239.63	74.16M / 26.68	100.71M / 32.29
	IEN+FC	4.28M / 58.30	8.64M / 96.38	920.11K / 98.71	3.75M / 244.74	76.46M / 27.32	103.02M / 33.10
	Maxout	4.25M / 62.64	8.61M / 111.1	866.91K / 104.53	3.77M / 253.85	74.16M / 28.18	100.71M / 34.02
	Base	858.87k / 20.01	1.73M / 36.28	188.09K / 47.73	800.3K / 114.13	15.30M / 7.62	20.61M / 9.20

From Table 9 we notice that Maxout training time is always more than IEN. There is also an increase in both training time and parameters size of IEN+FC against IEN only. Training time of IEN with $m = 4$ is much cheaper than training 4 ordinary models, but it comes at the cost of the memory only during the training time.

G Empirical analysis of dropout

We wanted to analyze the effect of using dropout everywhere except the input and output layers. Dropout performance is the best when its probability is set to 0.5. We tested two combinations, dropout applied as before and IEN followed by dropout. Table 10 summarizes these experiments. We notice that dropout only leads to very poor performance in comparison with IEN and base models. This shows that even though theoretically dropout can lower the variance, it is not applicable everywhere unlike IEN. Also, from the same Table 10 we notice that when IEN is combined with dropout the results get slightly better. This suggests the power of IEN in reducing the variance even when the model is being driven by another component. This dropout behaviour was noticed by (He et al., 2016), yet (Zagoruyko & Komodakis, 2016) suggested the correct placement of dropout is to be inserted into residual block between convolutions with probability set to 0.3. In Table 10 the right column show the results of correctly placed dropout. We still notice that IEN performs better than it.

Table 10: Results of applying dropout everywhere except the input and output layer. Mean error rate and std for multiple runs are reported. The lower the better. IEN uses $m = 4$.

Dataset		IEN	Base	Dropout	Dropout+IEN	Correctly Placed Dropout
CIFAR-10	ResNet56	6.93±0.26	8.38±1.2	90.48±0.04	90.12±0.01	7.38±0.10
	ResNet110	6.27±0.32	6.38±0.48	91.29±0.07	90.13±0.01	9.33±0.18
CIFAR-100	ResNet56	29.34±0.49	29.97±0.71	99.09±0.015	98.98±0.01	29.10±0.32
	ResNet110	28.17±0.11	27.83±0.64	98.91±0.01	98.98±0.01	35.13±0.71

H Guide to the code

Most of the models needed 1x Nvidia V100 with 32 GB of GPU memory for training. Some DenseNet models needed 2x Nvidia V100. Our code is attached for results reproduction with a quick README.MD.

H.1 ResNet Details

We used an open source repository (Idelbayev) that produces the same results like ResNet original paper using Pytorch (Paszke et al., 2019). The models were trained for 200 epochs using a learning rate of 0.1 and batch size of 128.

H.2 DenseNet Details

We used an open source repository (Pleiss et al., 2017) that produces the same results like original DenseNet paper but in more memory efficient way using Pytorch. The models were trained for 300 epochs using a learning rate of 0.1 and batch size of 64. The is code located at:

H.3 VGG Details

We used an open source repository (Fu). The models were trained for 300 epochs using a learning rate of 0.9 and batch size of 128.

H.4 NAS Details

We used the default settings in NAS-Bench-201 (Dong & Yang, 2020). We only added our IEN and Maxout operators and searched, trained the models.