Ensemble strategies II – Awareness

Based on *"Why M heads are better than 1: Training a Diverse Ensemble of Deep Networks"* [1]

https://arxiv.org/abs/1511.06314

This section explores the second decision to make when building ensembles of deep models.

It is the continuation of the strategies defined in this previous work. However, this work understands ensembles as entire models that could be trained together to outperform the previous strategies, which simply cared about how to better aggregate the outputs of individually trained networks.

In essence, this works answers the question:

What is the best strategy to build and ensemble?

Table of Contents

Statements

Regardless the approach used to explain the success of ensembles, the concept of diversity is at the heart of every explanation. Training multiple learners with decorrelated errors will improve the prediction when averaging their predictions.

Motivation

Authors of [1] claims that ensembling is typically treated as a post-hoc procedure implemented by averaging independently trained models with model variation induced by bagging or random initialization.

Objective

Firstly, understand, from the current strategies for ensembling, which ones are not suitable for deep models.

Then, come up with strategies to improve the state-of-art approaches by explicitly encouraging diversity within the ensemble.

Metrics

"There is no way to make progress if you cannot measure it" – François Chollet

It is necessary to define metrics to make comparison between the different strategies to build an ensemble. Metrics used in [1] are:

Ensemble-Mean Accuracy

This is the accuracy of the naïve averaging strategy at test time – averaging the beliefs of all members. As discussed, a strong performance of this strategy indicates that the ensemble members generally agree.

Oracle Accuracy

This is the accuracy if an oracle could select the prediction of the most accurate member for each example. This is a measure of the optimal performance that an ensemble could reach.

1. Standard approaches to train ensembles

There are 2 strategies mostly used in ensembles of deep models.

Random Initialization

Random Initialization consists on using the same model with a different random initialization and train each of them in the entire dataset.

The loss surfaces are highly non-convex and neural network training relies on the ability to find minimizers of those functions [2]. Because of the nature of these surfaces, whether or not to get stuck in a local minimum highly depends (of course among many other factors) on where in the surface the network start. Furthermore, [3] exposes the problematic of the appearance of saddle points in these high dimensional spaces. Such saddle points are surrounded by high error plateaus that can dramatically slow down learning and give the illusory impression of the existence of a local minimum.

Therefore, random initialization **encourage diversity by forcing the individual learners to explore different parts of the surface of the loss** function, to take advantage of these exploration when aggregating the predictions to provide a final ensemble output.

Bagging

Bagging consists also on using the same model. However, this time the initialization of the models is the same, but each one is trained with a different subsample of the dataset.

Therefore, diversity is created in this case by **forcing each individual learner to specialize in some part** of the data.

There is also possible to make a combination of the two approaches.

Experiments

Authors observed that:

Bagging reduced the Ensemble-Mean accuracy compared to random initialization, and the Oracle Accuracy remains nearly constant.

The bagged networks have been exposed to less data, since **bagging reduces the unique data exemplars by 37% for each member**. Therefore,

---

Random Initialization may not only be sufficient but preferred over bagging for deep networks, given their large parameter space and the necessity of large training data.

---

2. Tree Nets – Parameter Sharing

Authors in this paper propose the TreeNets. They are **ensembles of CNNs that share some part of the network**, always from the input up to the called **branching point**. This shape gives the name of tree nets.

Motivation

The motivation behind TreeNets is that current ensemble approaches introduce a lot of duplication of parameters and operations. In particular, convolutional neural networks (CNNs), widely used in computer vision applications, are deep models that perform a great number of operations and also take a considerable time to train. This phenomenon becomes a bottleneck that could be faces using TreeNets.

TreeNets differ from each other in where the branching point set. Therefore, it allows an entire spectrum across the length of the network of partially shared structures, shown in Figure 2. The margins of this range correspond to the fully shared and the no shared structures shown in Figure 1.

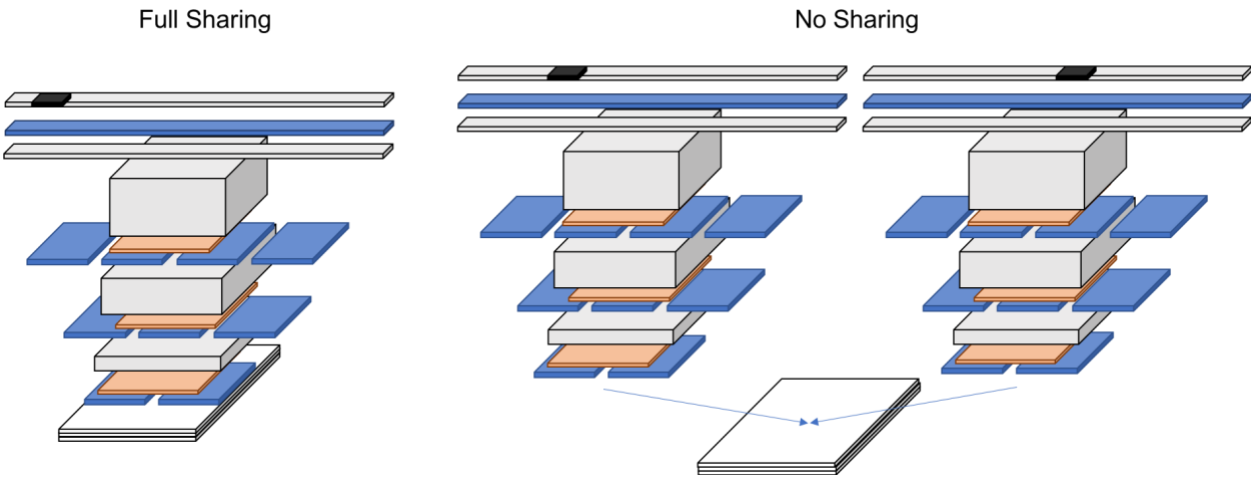

Figure 1. Full shared and Zero shared TreeNets

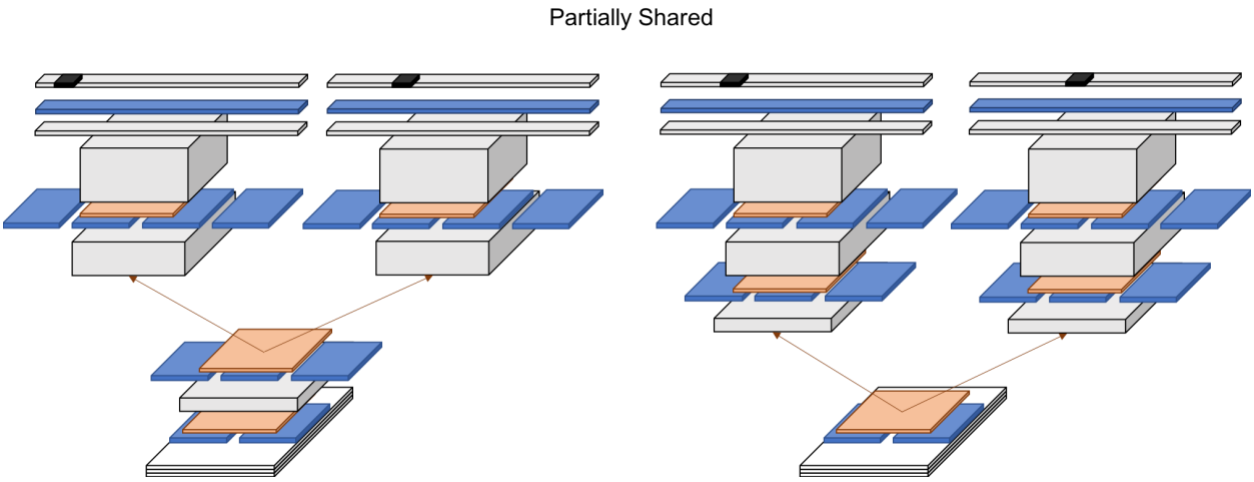

Figure 2. Partially Shared TreeNets

Observations

Authors evaluated TreeNets on classic benchmark datasets for image classification (ImageNet, ILSVC-Alex, ILSCVRC-NiN), object detection (PASCAL VOC 2007 using Fast R-CNN).

Shared parameter networks not only retain the performance of full ensembles, but can outperform them

---

Trying to give sense to this behavior, we could accept that:

By sharing the low-level weights, each weight is being updated by multiple sources of supervision.

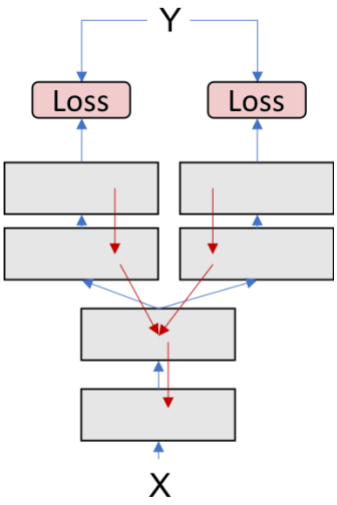

Figure 3. Regularization

In the scheme of **Error! Reference source not found.**, it is better shown how two different outputs, what we could say about 1 attempt of each individual member of the ensemble to predict the value of the response is computed separately. However, in the backpropagation of the error computed by the loss function, they converge after the branching points.

This indicates that the feedback of having 2 experiences is updated in a single weight update by the shared layers.

TreeNets can therefore improve generalization thanks to this sharing, which favors slightly better low-level representation.

Furthermore, in classic ensemble techniques, each individual learner performs about as well as the base architecture. However, in TreeNets, the performance of the individual networks is also improved, apart from the ensemble performance.

Conclusion

- TreeNets with a small number of initial shared layers (typically 1-2) outperform classic ensembling in terms of accuracy.
- Because of sharing layers, the total number of parameters the ensemble consist on is smaller.
- Less parameters yields to a reduction in the computation time, as well as in the memory requirements.

3. Ensemble-Aware Loss Functions

Authors motivation for having ensemble-aware loss function comes from the counter-intuitive observation of averaging on the probabilities being worse than averaging in the scores of the individual learners to provide an ensemble output, as mentioned in previous sections.

Bringing Figure 4 from that section where it was defined the different output aggregation methods:

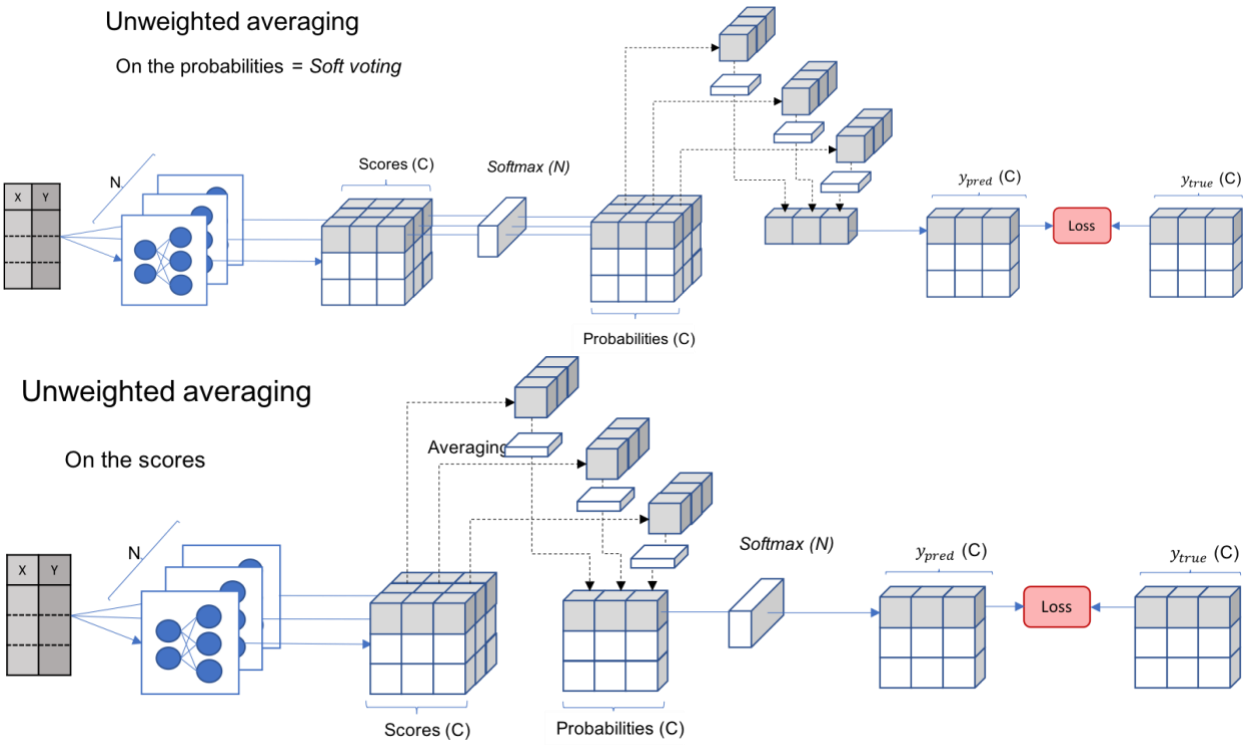

Figure 4. Output averaging on ensembles

The natural idea is to think that if the accuracy is measured as the correspondence between the averaged beliefs and the real values, it should be optimal to average the probabilities during training.

How is it possible that explicitly optimizing the performance of Ensemble-Mean does worse than averaging independently trained models? – ***Lack of diversity***

In both averaging scenarios, the gradients are mixed when backpropagating them through the networks.

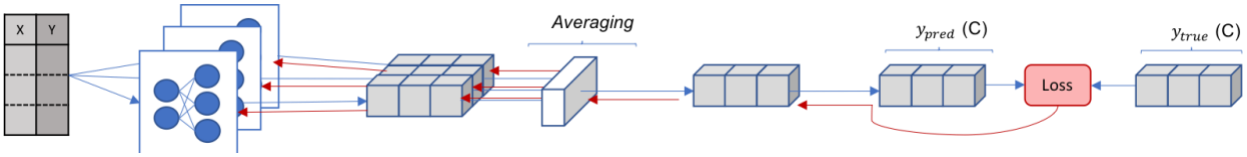

Figure 5. Loss of gradients diversity

As shown in Figure 5, the gradient is calculated right after the computation of the loss function. This same gradient is backpropagated equally to each network of the ensemble.

The responsibility for mistakes is shared, which eliminates gradients diversity.

This can also be understood mathematically, if we calculate the gradients in a generic averaging layer:

$$\mu(x_1, \dots, x_N) = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \text{(Eq 1)}$$

It results on:

$$\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial \mu}\frac{\partial \mu}{\partial x_i} = \frac{\partial l}{\partial \mu}\frac{1}{N} \qquad \text{(Eq 2)}$$

Which does not depend on i !

3.2. Encouraging diversity (via Multiple Choice Learning)

So far, the ensemble model was giving a single answer because of the averaging of the individual network beliefs. However, there are many settings in which **giving more than one solution could be more interesting than producing a single answer**. Multiple Choice Learning [4] introduced a way of getting several solutions.

From the point of view of an oracle which always selects the best output out of the m models that the ensemble consists on, the loss, called **oracle set-loss**, is expressed as:

$$\mathcal{L}_{set}(x, y) = \min_{m \in [1,M]} l(\theta_m(x), y) \qquad \text{(Eq 3)}$$

Given that for classification tasks the most loss function used is the cross-entropy:

$$l(x, y) = -\log\left(p_y^{\theta_m}\right) \qquad \text{(Eq 4)}$$

Being $p_y^{\theta_m}$ the predicted probability (after softmax).

If we use a binary variable α_{mi} to indicate whether the network θ_m was the one with the lowest error, it is possible to define a **Cross-Entropy Oracle Set-Loss** over a dataset D.

$$\mathcal{L}_{set}(D) = \frac{1}{|D|}\sum_{(x_i, y_i) \in D}\sum_{m=1}^{M} -\alpha_{mi}\log\left(p_y^{\theta_m}\right) \qquad \text{(Eq 5)}$$

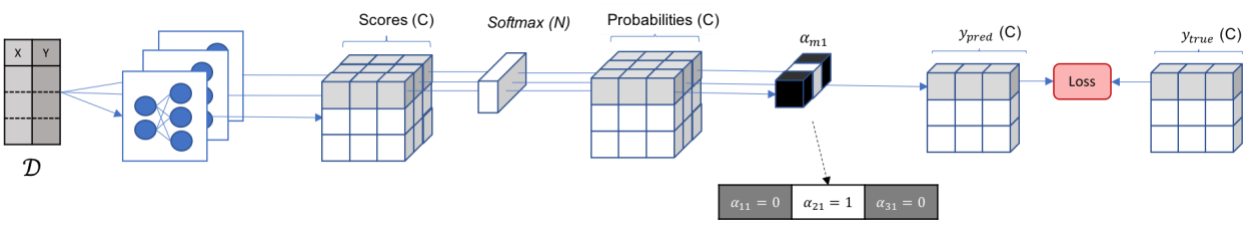

Figure 6. Oracle Set-Loss Architecture

Figure 6 shows how Eq 5 is defined. The α_{mi} represents which individual network is the one used for the particular example i. Only 1 value of the α vector will be 1, being 0 the rest of them. Note how this cross-entropy oracle set-loss expression is an upper bound on how good your model could possible do.

This can now be extended allowing α to admit k predictors.

Varying k from 1 to M trades off between diversity and the number of examples each predictor sees, which affects generalization and convergence. Therefore, as k increase, the oracle accuracy decreases in exchange of individual network accuracies.

Observation

Each individual model performs really poor, 19-27% accuracy. However, as an ensemble, the oracle accuracy is over 93%.

This shows that the networks have specialized in different aspects of the dataset, and that there is in fact diversity among the networks. We can also track this behavior by comparing which test examples are assigned to each ensemble member: Figure 7 shows the labels distribution depending on the value of k.

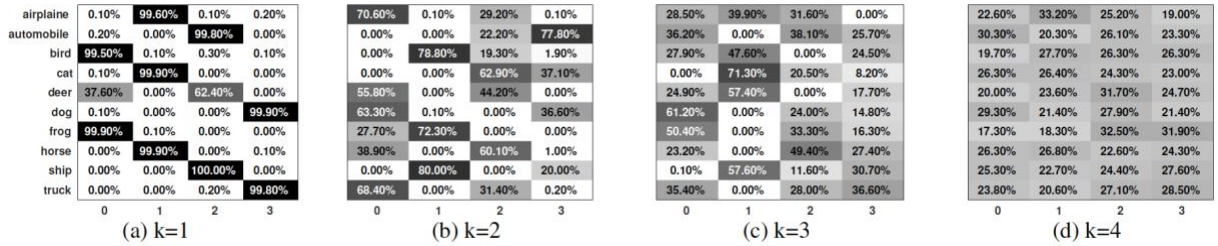

Figure 7. Assigned classes distribution

For k=1, we could observe almost complete division of the label space. Each network has fully specialized in only some classes and are 'ignorant' for the rest of them ~ ***Label Clustering***

As the value of k increases, uniformity is being introduced and every network converges to the same knowledge, losing all the diversity within the ensemble.

Conclusions

The diversity introduced by random initialization of parameters is desirable over the one from bagging.

Averaging the beliefs has the unintended effect of removing diversity from ensembles.

Diversity is important in high level layers. However, it might be better no diversity in low-level filters.

Bibliography

[1] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall and D. Batra, "Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks," 2015.

[2] A. "Visualizing the loss landscape of neural nets".

[3] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization".

[4] A. Guzman-Rivera, D. Natra and P. Kohli, "Multiple Choice Learning: Learning to Pproduce Multiple Stuctured Ouputs," *In Advances in Neural Information Processing Systems,* 2012.