Improving Hierarchical Image Classification with Merged CNN Architectures

Anuvabh Dutt
Univ. Grenoble Alpes, CNRS,
Grenoble-INP, LIG,
F-38000 Grenoble France
anuvabh.dutt@univ-grenoble-alpes.fr

Denis Pellerin
Univ. Grenoble Alpes, CNRS,
Grenoble-INP, GIPSA-Lab,
F-38000 Grenoble France
denis.pellerin@gipsa-lab.grenoble-inp.fr

Georges Quénot
Univ. Grenoble Alpes, CNRS,
Grenoble-INP, LIG,
F-38000 Grenoble France
georges.quenot@imag.fr

ABSTRACT

We consider the problem of image classification using deep convolutional networks, with respect to hierarchical relationships among classes. We investigate if the semantic hierarchy is captured by CNN models or not. For this we analyze the confidence of the model for a category and its sub-categories. Based on the results, we propose an algorithm for improving the model performance at test time by adapting the classifier to each test sample and without any re-training. Secondly, we propose a strategy for merging models for jointly learning two levels of hierarchy. This reduces the total training time as compared to training models separately, and also gives improved classification performance.

KEYWORDS

Hierarchical classification, Ensemble learning, Image classification

ACM Reference format:
Anuvabh Dutt, Denis Pellerin, and Georges Quénot. 2017. Improving Hierarchical Image Classification with Merged CNN Architectures. In *Proceedings of CBMI '17, Florence, Italy, June 19-21, 2017,* 7 pages.
DOI: 10.1145/3095713.3095745

1 INTRODUCTION

Recent research has shown that convolutional neural networks (CNNs) are a powerful and generic class of models for various tasks. Particularly, for image analysis, a single model architecture is able to perform several tasks like classification, object detection and semantic segmentation. Models such as ResNet [5] and DenseNet [6] are very efficient in the use of their parameters and are able to learn highly discriminative and reusable features.

When target categories are organized in a hierarchy, DCNN models are in principle able to automatically learn and exploit the implicit hierarchical structure. However, we have shown in a previous work [4] that they actually do it only partially as it is possible to further improve the overall classification performance by explicitly using the subsumption relations between target categories for enforcing consistency between their predicted probabilities. We consider here the simple case of two hierarchical levels, corresponding to "coarse" and "fine" categories, where all categories at a same hierarchical level are mutually exclusive. The method is based on an interpolation between the probabilities directly predicted at the coarse level and those *inferred* at the coarse level, from predictions at the fine level. The gain comes without any additional training, it is adapted for each test sample, and it is obtained using no other information besides the subsumption relations. As a first contribution, we confirmed that this gain is still effective, though reduced, when ensemble learning [3] is used.

We observed that the probability adjustment procedure works significantly better when applied to separately trained models for coarse and fine categories, as compared to jointly training a model [4]. However, the inconvenience of a separate training is that the training time and the number of model parameters are roughly linear in the number of levels in the hierarchy. To overcome this limitation, as a second and main contribution, we propose partially merged network architectures that have the same or better classification performance, with only a small increase in training time and number of parameters.

CBMI '17, Florence, Italy

978-1-4503-5333-5/17/06…$15.00
DOI: 10.1145/3095713.3095745

2 RELATED WORK

Multi task learning [2] is a general framework for training a model on several tasks. If we consider classification at different levels of hierarchy as different tasks, our work can be thought of as using tasks to complement each other, in the sense that improvement in performance of one task can be obtained from the other.

Royer et al. [10] proposes a probabilistic framework to adapt the predictions of a classifier, based on the sequence of images that are presented to the classifier at test time. Their work is concentrated on a flat distribution of categories. In our case, we are concerned with adjusting and correcting our classifier by taking into account hierarchical relationships among object categories. This adjustment is done by taking into consideration just one test sample and not the previously seen test samples. The correction they do also takes into account the feedback from the prediction, that is whether the classifier was correct or not.

More closely related is the work of Jia et al. [8]. Given query images, they aim to identify the semantic subspace out of all categories, to which the test images belong. However, they work in a setting involving a single classifier with high capacity, trained on large amounts of data. Their aim is to select a relevant 'portion' of this trained classifier in order to discriminate over a subset of categories. In our setting, we aim to 'transfer knowledge' between two distinct classifiers which were trained separately or as partially merged. The only information we use during inference is the given hierarchy of categories and a single test sample.

Ensemble learning has been proven to be very efficient when used with DCNNs [3]. It consists in training a number of instances of exactly the same model with different random initializations and then averaging their predictions. Experiments have shown that

this approach is much more efficient than using the same model architecture with a correspondingly higher number of parameters.

3 METHOD

In this section, we describe our problem setting and our proposed approach. First, we recall the motivation and probability adjustment scheme as shown in [4]. We then present the notion of merging architectures. In the next section, we experimentally validate our approach.

3.1 Problem Setting

We consider a standard multi-class classification setting. The goal is to assign to each input sample $x \in \mathcal{X}$, a class label $y \in \mathcal{Y} = \{l_1, ..., l_N\}$. The assignment of labels for each input depends on the output of a classifier, $f : \mathcal{X} \rightarrow \mathcal{Y}$, as $f(x) = argmax_y f_y(x)$. We consider a probabilistic classifier whose output is $P(y|x)$ and denotes the probability of the class label y for a test sample x. The $P(y|x)$ values are often obtained by applying a soft-max function to the $f_y(x)$ raw output of a classifier.

We consider a setting with two sets of class (or category) labels corresponding to a two-level hierarchy with N_c *coarse* labels $\mathcal{Y}_c = \{c_1, ..., c_{N_c}\}$ and N_f *fine* labels $\mathcal{Y}_f = \{f_1, ..., f_{N_f}\}$. The hierarchical relation is defined by a function s from $\{1, ..., N_f\}$ to $\{1, ..., N_c\}$ specifying that the category with label $c_{s(j)}$ is the super-category of the category with label f_j.

Considering these two sets of labels, it is possible to train either separate classifiers, producing respectively N_c and N_f probability values, or a single joint classifier producing directly $N_c + N_f$ probability values. In the latter case, a single network is used but two separate soft-max functions are used at the end. In both cases, any test sample will receive a predicted probability for each of the N_f *fine* sub-categories for each of the N_c *coarse* categories. Obtaining models is quite straightforward and there is a vast collection of available network architectures and pre-trained models [1, 11].

Our goal here is two-fold: to ensure consistent probability predictions across the two levels and to improve the predictions of the coarse and fine classifiers using the information from the hierarchy of labels. For each coarse category label y_c, we would expect its predicted probability to be equal to the sum of the predicted probability for all the fine category label y_f which are subcategories of y_c, i.e. for all i in $\{1, ..., N_c\}$ and any test sample x, $P(c_i|x) = \sum_{s(j)=i} P(f_j|x)$. Ensuring such consistency may be useful when the probabilities are later combined for computing the probabilities of more complex events. We hypothesize that it can also lead to a better classification performance at both the coarse and the fine levels.

The coarse / fine probabilities consistency is far from being satisfied in real applications, even for the joint training that does better than the separate one. Figure 1a shows the probability obtained from two classifiers trained at the coarse and fine levels. The horizontal axis shows the probability assigned examples for being in a particular coarse category (in this case, category 1 from CIFAR-20. See section 4). The vertical axis shows the total probability assigned to the corresponding fine categories, for the same example. Green dots denote data points which actually belong to the coarse category.

3.2 Classifier Adaptation

We want to enforce the consistency constraint that the confidence that a classifier has for a coarse category and the corresponding fine categories should be equal. Formally, this is denoted as:

$$\forall i \in \{1, ..., N_c\}\ P(c_i|x) = \sum_{s(j)=i} P(f_j|x) \quad (1)$$

This state is depicted in figure 1c. For all data samples, equation 1 will be satisfied if we map them to the diagonal. This procedure is outlined in algorithm 1. The basic principle is to orthogonally project the point $(P(c_i|x), \sum_{s(j)=i} P(f_j|x))$ on the diagonal line making both coordinates equal to $(P(c_i|x) + \sum_{s(j)=i} P(f_j|x))/2$. This is done independently for each x test sample, this is why x does not appear in algorithm 1. The adjusted value for $P(c_i|x)$ is directly obtained from the projection. The adjusted values for $P(f_j|x)$ are scaled with the same factor for all j with $s(j) = i$ so that the adjusted value for $\sum_{s(j)=i} P(f_j|x)$ matches the projection. Note that this method guarantees that the sum of the coarse (resp. fine) class probabilities remains equal to 1.

In order to control the degree to which we enforce equation 1, we introduce an interpolation factor, α. In algorithm 1, the term α, is present inside the `for` loop. This term allows us to linearly interpolate between the original probabilities (Figure 1a) and the condition of equation 1 (figure 1c). For $\alpha = 0$, no adjustment is made. For $\alpha = 1$, a full adjustment is made. Note that for $\alpha = 2$, the adjusted values for $P(c_i|x)$ correspond to a direct computation as $\sum_{s(j)=i} P(f_j|x)$, i.e. the predictions for the coarse categories are computed only from the predictions from their fine sub-categories. Since, we modify the output probabilities based on the test sample, we term this procedure the *classifier adaptation* step.

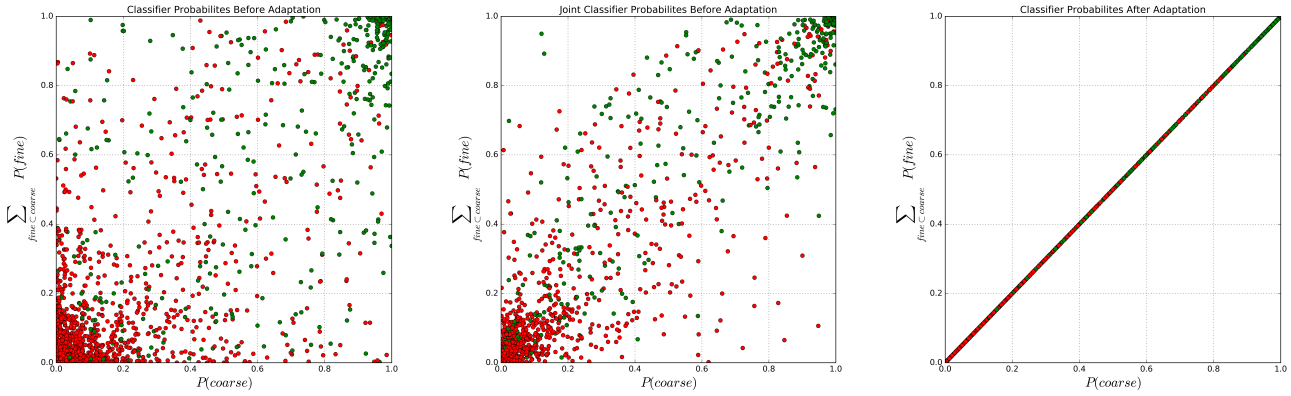
ALGORITHM 1: Classifier Adjustment

Input: Probabilities from coarse and fine classifiers
Output: Adapted classification probabilities, adjusted_coarse[i], adjusted_fine[i]

```
coarse[i], fine[i] //Lists of coarse and fine classes;
fine_coarse[j] // Coarse class of fine classes j;
P_coarse[i], P_fine[j] // Probability of coarse and fine classes;
Q_coarse[i] = ∑_{fine_coarse[j]=i} P_fine[j] // sum of probability of coarse
  classes of class i;
for i = 1, #coarse do
    target_prob = (P_coarse[i] + Q_coarse[i]) / 2;
    adjusted_coarse[i] =
    (1 − α)× P_coarse[i] + α× target_prob ;
    if Q_coarse[i] != 0 then
        for fine_coarse[j] = i do
            fine_target_prob =
            (P_fine[j] × target_prob[i]) / Q_coarse[i] ;
            adjusted_fine[j] =
            (1 − α)× P_fine[j] + α× fine_target_prob;
        end
    else
        adjusted_fine[j] =
        adjusted_coarse[i] / (∑_{fine_coarse[j]=i} 1)
    end
end
```

3.3 Inference with Classifier Adaptation

Given the trained models, inference is done by using algorithm 1. For each test sample, first we obtain the probabilities from the coarse and fine classifiers. These probabilities are input to algorithm 1



(a) Seperate Training **(b) Joint Training** **(c) Adjusted Probabilities**

Figure 1: Distribution of probabilities for a coarse class obtained from coarse grain classification (horizontal axis) and from fused fine grain classifications (vertical axis). Left and middle: direct soft-max values from classifiers, respectively for separate coarse and fine classifications and for joint classification. Right: probabilities adjusted to enforce that $P(c_i|x) = \sum_{s(j)=i} P(f_j|x)$.

which outputs the adjusted probabilities. The final prediction corresponds to the class having the maximum probability after the adjustment. The cost involved in doing this step is linear in the number of the super-classes. The interpolation factor, α, can be tuned using a validation set for obtaining the best results.

Note that the adjustment of probabilities is done according to the hierarchy of all the labels. The algorithm does not have access to the true label.

3.4 Merged Architectures

A ResNet model is composed of several *residual blocks*. Each residual block contains a certain number of convolution filters, followed by a `batchnorm` [7] layer and a non-linearity. A residual block takes as input, feature maps of dimension $d \times d$ and outputs feature maps of dimension $d' \times d'$.

Based on their feature map dimensions, convolutional layers of residual (and many other) networks can be organized into a small number of stages (typically three). At the output of each stage (or block), the feature maps are down sampled using a pooling layer, and the number of feature maps is (typically) doubled in the following stage. We show this structure in figure 2a. Figure 2b shows a simplified view while including the soft-max operator and log-likelihood loss function used during the training phase. The architecture of DenseNets [6] follows a similar pattern, with the only difference being that each of these stages is called a "dense block", within which a layer L takes as input the output feature maps of all previous layers $\{1...L-1\}$.

We propose to merge layers among the coarse and fine models. The intuition is that several shared features will be useful for both the coarse and fine categories. This may lead to faster and more robust feature learning as the training phase will benefit from gradient signals from the two loss functions.

The granularity of the merge is at the level of the residual blocks. Merging (or splitting) layers inside a residual block is non-trivial (especially in the case of DenseNet). We explore merging strategies after each of the 3 computation stages. After each of the stages, a `maxpool` layer is used to aggregate all the features that have been computed so far. Hence this serves as a good point in the computation pipeline to determine until which point in the computation pipeline, the layers should be merged.

We denote the usual training situation by `separate` (Figure 2c). In this case, two models are trained, one for the coarse and one for the fine categories. For the case of merged models, training is done for both category levels simultaneously. We explore variants of merged models:

- `merge-0` Fine and coarse models are trained independently but both receive the same input and the loss is summed (Figure 2c). Though the networks in Figures 2c and 2d are different, they should theoretically lead to exactly the same training and predictions.
- `merge-1`, `merge-2`, `merge-3` are denoted by figures 2e, 2f, 2g. The common blocks in the layers are closer to the input. If the fully connected block contains only a single linear layer (which is generally the case in very deep networks), it does not make any difference whether it is merged or not. `merge-3` corresponds to a joint training where all layers are shared, and only the loss function separates the coarse and fine categories.

4 EXPERIMENTS

We conducted experiments for testing the hypothesis that enforcing consistency constraints between the probabilities for the coarse and fine categories can improve the classification performance, and for determining which of the merged network architectures provide the best performance. Additionally, we evaluated how these results hold when the ensembles (averaging the prediction of several instances of the same network trained with different random initialization) are used. Initially, we trained multiple network instances for figuring out how significant were the observed differences between the five considered model configurations and the different error rates obtained with different values of the adjustment parameter, α. We then used these different training instances for evaluating the ensemble learning approach (only for ResNet models).

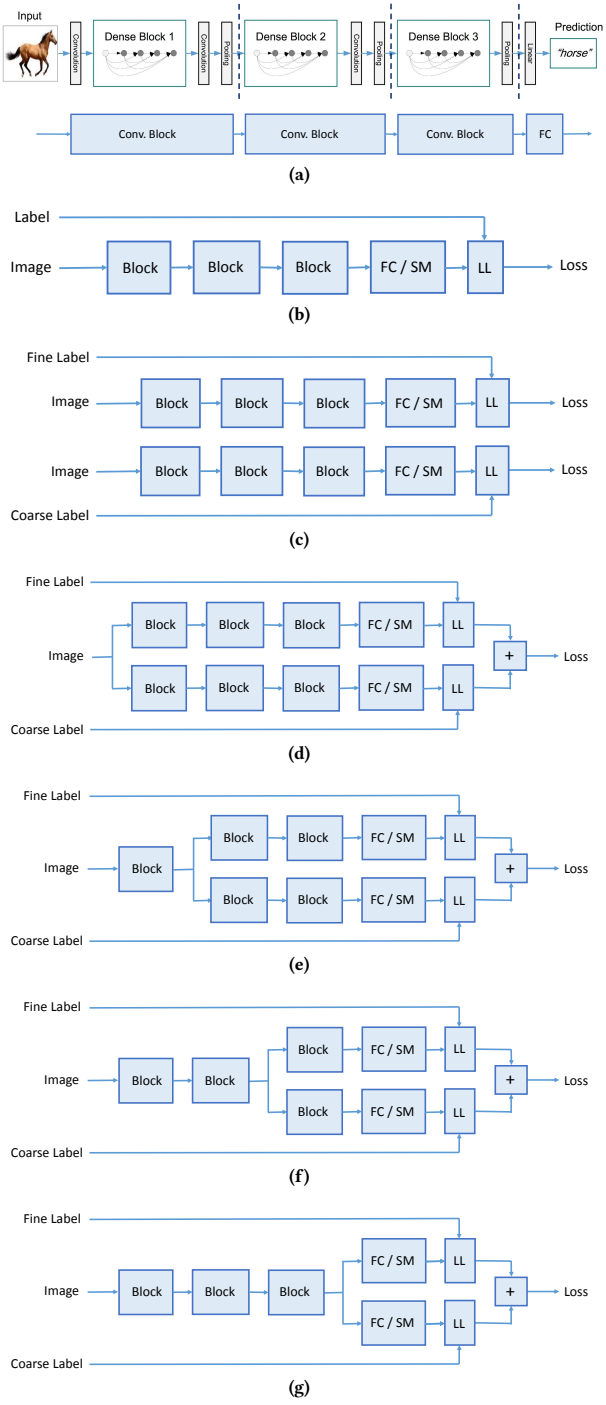


Figure 2: `Merge` architectures: (a) classical block decomposition of very deep or ultra-deep networks with boundaries at resolution changes with an example from [6]; (b) simplified diagram including the soft-max and loss functions for the training phase; (c) coarse and fine classification with separate networks; (d) merge of the loss function only; (e)-(g) progressive merge of the convolution blocks.

Data Set. We used the CIFAR-100 data set [9] for all the experiments. This data set contains 50,000 training images and 10,000 test images. These images are belong to 100 categories that we shall consider as the fine categories. The authors of the CIFAR dataset also provide a list of 20 coarse categories, which we refer to as CIFAR-20. Each coarse category groups exactly 5 fine sub-categories. All of the the fine (resp. coarse) categories are exclusive of each other.

Performance metric. The error rate on the top-1 prediction is used as the performance metric for both the CIFAR-20 and the CIFAR-100 classifications.

Classifier Training. Two state of the art deep convolutional neural network (DCNN) architectures were used: ResNet-164 [5] and DenseNet-100-BC [6]. For the case of training with the merged models, the soft-max output layer (SM) is split into two parts corresponding to the 100 fine and 20 coarse classes. This is depicted in figure 2c-2g. The LL block (loss layer) is removed during prediction.

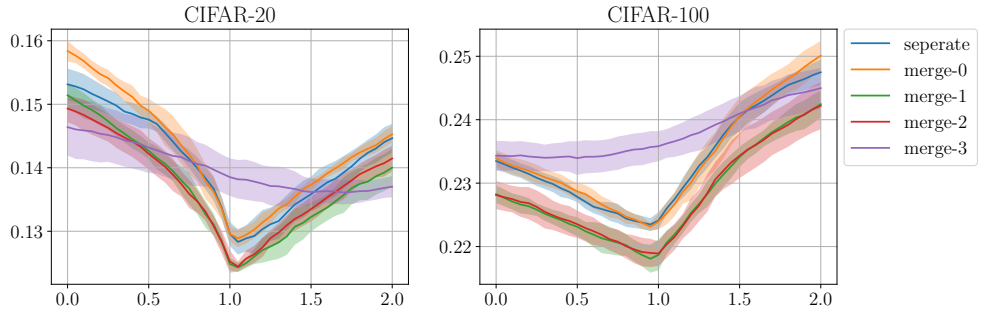
Standard data augmentation is used during training, which is horizontally flipping the images with 0.5 probability and taking random crops of 32×32, after padding images with 4 pixels on each side. For each model, the hyper-parameters and learning rate schedules were set according to the reported values in their corresponding papers.

Results. We trained the five different merged architectures described in figure 2c-2g five times for ResNet-164 and for DenseNet-100-BC. We then applied classifier adaptation while varying the α parameter from 0 to 2 and we plotted the top-1 error rate for both coarse and fine categories as a function of α for all combinations.

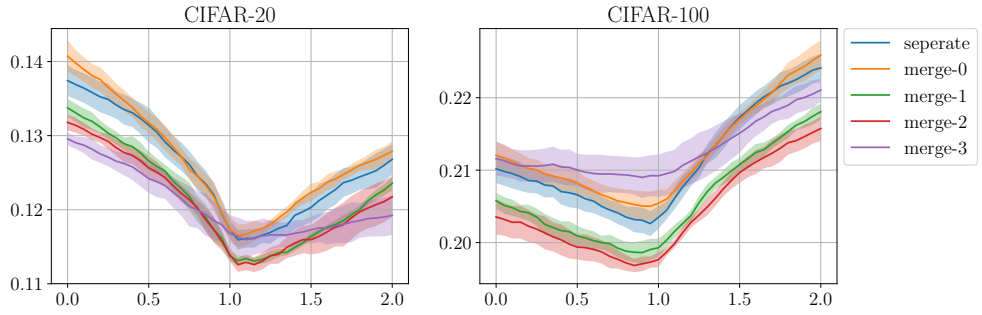
Results are presented in figure 3 for ResNet-164 and in figure 4 for DenseNet-100-BC. In figure 3, plots are given for ensemble learning with ensembles of different sizes: ensemble-n corresponds to the averaging of the predictions from n different instances of the same architecture (ensemble-1 corresponds to prediction from a single instance). Using the five different trainings, it is possible to get a rough estimate of the standard deviations around the mean value. In figures 3 and 4, the solid lines denote the mean and the shaded regions denote one standard deviation around the mean.

Analysis of ResNet results. From figure 3, we can make the following observations:

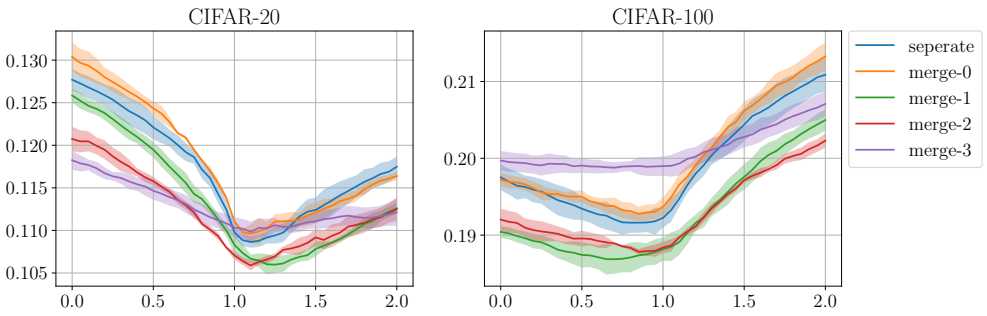
- The statistical differences between the various merged architectures is sometimes low since there is a significant overlap between the shaded areas and these are only at one standard deviation. Still, the difference becomes sharper when considering the ensemble-2 and ensemble-4 versions and the evolution of the performance according the α parameter is likely to be significant.
- The `separate` and `merge-0` are sometimes slightly different though they should theoretically be identical. This might be due to some normalization steps in the implementation of the stochastic gradient descent that do not work exactly in the same way in both cases. The difference is significant only in the CIFAR-20 case and for values of α close to 0.
- For both the CIFAR-20 and CIFAR-100, the optimal value for the α parameter is quite close to 1 as expected and classifier adaptation leads to a significant performance improvement, confirming our hypothesis. The optimal value is slightly higher (resp. lower) than 1 for CIFAR-20 (resp. CIFAR-100) and this effect seems to be more



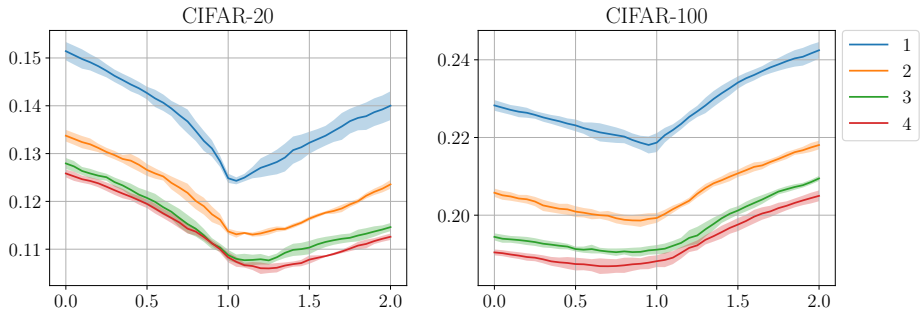
(a) Ensemble-1 for the five merged architectures



(b) Ensemble-2 for the five merged architectures



(c) Ensemble-4 for the five merged architectures



(d) Ensemble-1 to ensemble-4 for the `merge-1` architecture

Figure 3: Top-1 error rates (Y axis) for the different merged ResNet-164-based architectures with respect to the interpolation factor α (X axis). Ensemble-n represents the top-1 error rate after averaging the predictions from n models.

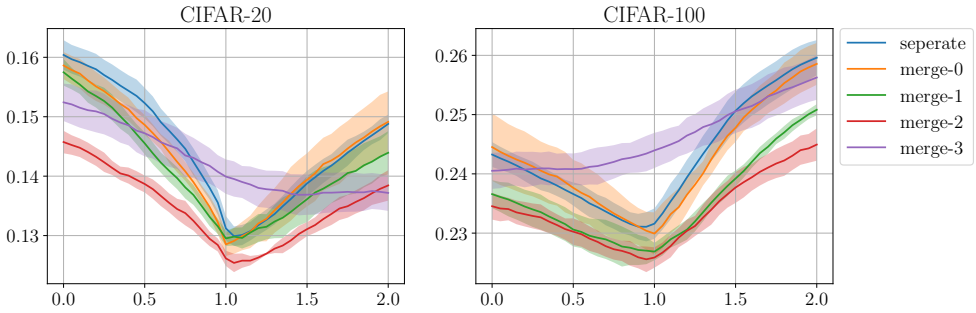


Figure 4: Top-1 error rates (Y axis) for the different merged DenseNet-100-based architectures with respect to the interpolation factor α (X axis). The displayed results are for ensemble-1 (single instance).

important in the case of ensemble with higher number of instances.

- For CIFAR-20, the performance obtained *only* from the probability predictions for the fine categories (at $\alpha = 2$) are significantly better than the performance obtained from the direct probability prediction for the coarse categories ($\alpha = 0$). However the performance obtained by combining both ($\alpha = 1$) are even better.
- For CIFAR-100, the performances obtained by combining probability predictions for both coarse and fine categories ($\alpha = 1$) are significantly better than the performances obtained only from the probability prediction for the fine categories ($\alpha = 0$).
- For different merged architectures, the intermediate ones, `merge-1` and `merge-2`, appear to have equivalent performance, and are better than the fully split or fully merged ones. We hypothesize that this is because it is better to fuse the first layers which are more general but not the last layers which tend to be more specific to the coarse or fine target categories.
- Ensemble learning provides a very significant gain as this has already been observed in simple training (without considering label hierarchy) but classifier adaptation provides an additional gain for all numbers of instances, especially for the coarse categories, even though the gain tends to decrease with the number of instances.

Analysis of DenseNet results. From figure 4, we can make the following observations:

- Results obtained using DenseNet-100 are consistent with those obtained using ResNet-164.
- As expected, the DenseNet-100 error rates are lower than ResNet-164's.

Statistical significance. The statistical significance of the different combinations have been studied using a paired Student's t-test.

- Differences at $\alpha = 1$ between `separate` and `merge-0` (resp. between `merge-1` and `merge-2`) are not statistically significant.
- Differences at $\alpha = 1$ between `merge-1` and `merge-2` on one side and between `separate` and `merge-0` on the other side are statistically significant at a p-value below 0.05.
- Differences between α values around the optimal value are statistically significant for variation of ± 0.1 for CIFAR-100 and of ± 0.2 for CIFAR-20.

5 DISCUSSION AND CONCLUSION

We presented a simple scheme based on the test sample for improving the accuracy of CNN models doing both coarse and fine classification. This adaptation is based on the semantic relationship among the training data of the model and does not depend on the any other information. We have proven experimentally that this scheme is quite effective and improves on the performance of state of the art models on CIFAR-100. The algorithm was demonstrated for CNN models but the formulation of our approach is such that it can be applied to any classifier that predicts probabilities over categories, for example Platt normalized SVM.

We used the classifier adaptation scheme from [4] to show that adjusting probabilities according to their semantics leads to increase in performance even in the case of ensembles.

We presented the method of merging models to train on coarse and fine categories together. We showed that this scheme is effective not only at reducing the training time but also at improving the performance over separate training of models, which is the most common paradigm. The merged architectures improve over the state of the art performance. We also showed that it is not restricted to any particular architecture. It holds true for different architectures and even when prediction is done over model ensembles.

For future work, we plan to investigate over multiple levels of hierarchy. This can be obtained from the ILSVRC-2012 data set. By itself, the labels do not have any hierarchy but as they are arranged according to WordNet, we can obtain the structure. Also, we currently considered the case in which categories are not always exclusive or in which some images do not contain any category. The main difference is that we would have to deal with inequalities rather than equalities related to the probability values and that adjustments would have to be done only when inequalities are not satisfied. In such cases, adjustments could be done with the same method.

ACKNOWLEDGMENTS

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01)

REFERENCES

[1] BVLC. *Caffe Model Zoo.* https://github.com/BVLC/caffe/wiki/Model-Zoo.

[2] Rich Caruana. 1998. Multitask learning. In *Learning to learn*. Springer, 95–133.

[3] Liran Chen and Greg Shakhnarovich. 2014. Learning Ensembles of Convolutional Neural Networks.

[4] Anuvabh Dutttt, Denis Pellerin, and Georges QuQuénot. 2017. Improving Image Classification using Coarse and Fine Labels. In *Proceedings of ACM International Conference on Multimedia Retrieval*.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision*. Springer, 630–645.

[6] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2016. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993* (2016).

[7] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 448–456.

[8] Yangqing Jia and Trevor Darrell. 2013. Latent task adaptation with large-scale hierarchies. In *Proceedings of the IEEE International Conference on Computer Vision*. 2080–2087.

[9] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. (2009).

[10] Amelie Royer and Christoph H Lampert. 2015. Classifier adaptation at prediction time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1401–1409.

[11] TensorFlow. *Model Zoo.* https://github.com/tensorflow/models.