Hierarchical image classification in CNNs

Prashant Kumar
(Dated: June 12, 2019)

Can hierarchical classification improve the classification accuracy of neural networks when the number of categories is large? To explore this question, I construct a neural network that classifies the CIFAR100 database into 10 coarse categories. Correlations between the coarse classification probabilities and the filters of another neural network are then used to perform the fine classification into 100 categories. I try three different approaches for coarse category construction. No significant change in accuracies compared to the baseline was observed.

I. INTRODUCTION

The image classification problem is one of the prime applications of convolutional neural networks (CNNs). In its simplest form, a single CNN model is used for this problem. However, when the number of possible categories is large, It might be useful to provide partial classification of images at an intermediate stage as opposed to end-to-end learning. One might find it attractive to let the neural networks learn what information to provide on its own. However, it may be useful to provide input from human intuition of learning. In this project, I want to see if partial classification into coarse categories might be useful. This is similar to the idea of hierarchical classification where an image is first classified into coarse categories and then into fine categories.

Hierarchical classification via a CNN architecture HD-CNN has been introduced in Ref. 2. They divided the K fine labels into C coarse categories with certain fine classes present in multiple coarse categories. An image is first classified into C categories, then the output of this classification is fed into C number of independent fine-classification layers to further classify these into the fine classes. They train the coarse classifier first based on a pre-constructed category hierarchy. Then they train fine-classifier layers.

On a different note, an approach has been used in Ref. 1 to perform classifcation of images where different image classes are visually very similar. For example, in identifying a particular species of bird from a set of pictures of birds. The filters of the convolutional layers of the neural network are combined in a bilinear way. This is done based on the intuition that different filters learn different features and thus correlations between them should be useful for classification. I use a method inspired by this approach.

In this project, I investigate whether hierarchical classification can be utilized to help improve the accuracy of a neural network. The fine classification layers are all independent in the HD-CNN architecture. However, it might be more useful to have a single network that does all the fine classifications. Therefore, inspired by the bilinear-CNN, I employ the outer product between the coarse category probabilities and the filters of the convolutional layers of a neural network. It is expected that each coarse category will choose a different combination of filters to do classification into fine categories, possibly improving accuracy. Further, since fine classification layers are not independent of each other, one can use disjoint coarse categories. In this project, I have tried three different approaches for coarse category construction.
(*The code for this project can be found in 3.*)

II. CNN ARCHITECTURE AND BROAD APPROACH

All experiments are done on CIFAR100 image database. It has 100 classes with 500 training images per class. The test set has additional 100 images per class.

Architecture details: The proposed architecture is shown in Fig. 1. For the coarse neural network, I choose VGG13 architecture that classifies the image into 10-coarse categories. For the main neural network, I choose VGG16. The output of the *Conv Layers* corresponds to the filter activations of VGG16 after the 13 convolutional layers. The neural network then evaluates the outer product of these filters with the coarse classification output. Thus, the input given to the classification layers is $512 * 10$-dimensional and the overall output is 100-dimensional. The classification layers are the usual linear, fully connected layers of VGG16. Batch normalzation has been used after every convolutional layer. Additionally, two dropout layers are added for regularization before the linear layers in the classification layers. The base code for VGG on CIFAR100 has been taken from 4. In addition, weight initialization has been used as in Ref. 5.

Outline of training methodology: The coarse neural network in Fig. 1 is first trained to do the coarse classification on the training set. Then this trained coarse NN is used while training the main NN.

Outline of coarse category construction: I compare results from three different methods for coarse category construction. In the first method, a category hierarchy is constructed by creating 10 coarse categories. Each coarse category corresponds to a subset of the 100 fine labels. Unlike Ref. 2, I classify a label into only one coarse category. In the second method, instead of grouping various labels together, the training images are

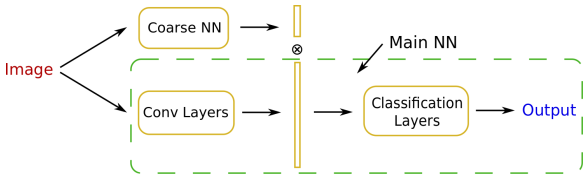

FIG. 1. Coarse neural network is trained to perform classification into 10 coarse categories. Its output is combined with the activations of convolutional layers of the main neural network through an outer product to perform the final classification into 100 categories.

grouped together to find 10 categories of images regardless of their actual label. In the third method, the full composite neural network model is trained end-to-end without any knowledge of a category hierarchy. The last method tells us whether the coarse classification is actually useful. The first method is named "Category Hierarchy"(CH), the second method "Coarse visual category" (CVC) and the third method "End-to-End"(ETE).

III. COARSE CATEGORY CONSTRUCTION

In this section, I describe in detail how coarse categories are constructed in each of the three methods.

A. Method 1: Constructing category hierarchy (CH)

To construct a category hierarchy, Ref. 2 utilize the spectral clustering on the confusion matrix of a trained neural network. A second method would be to use the coarse classes already defined in CIFAR100 databse. I have tried a different approach.

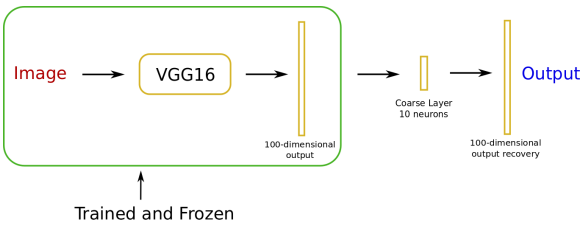

FIG. 2. A fully connected coarse classification layer is added to a trained VGG16 network to project the 100-dimensional output to 10-dimensions. A 100-dimensional fully connected recovery layer is further added to recover the original classification.

First, I train a VGG16 neural network on the training set that classifies the training set into 100 categories. Then the 100-dimensional output of the VGG16 neural network is classified into 10 coarse categories using the structure shown in Fig. 2. I modify the trained VGG16 as follows. A batch normalization layer and a fully-connected layer of 10 neurons with *relu* activation is added. This is called the coarse layer. It serves the purpose of projecting the 100-classes to 10-dimensions. After this, a fully connected recovery layer of 100-neurons is added. Thus, this layer classifies the 10-dimensional data back into 100-classes. The two new layers added to this neural network are trained with cross entropy loss. Thus, this neural network maximizes the recoverability of the output after the 100-classes are projected onto 10-dimensions. The intuition behind this is that visually dissimilar categories are easier to distinguish and thus easy to recover by the recovery layer.

After training these last two layers, I calculate the activations of the 10-dimensional coarse layer on every image in the training set. For each of the 100 labels, I find the neuron that gets activated the most by the 500 images of this label. This neuron is considered to be the coarse class of this label. The exact category hierarchy is highly dependent on the random initialization and pretraining. However, I select the one that has a roughly balanced fine label distribution.

I found that using a strong L1 regularization with coefficient 0.005 for the weights of the recovery layer helped in producing more balanced coarse categories. This was done based on the understanding that L1 regularization is a feature selector, thus it makes the weight matrix of the recovery layer sparse. So, a given fine label uses a relatively small number of neurons from the coarse layer. The work gets spread out more uniformly over the neurons of the coarse layer.

B. Method 2: Constructing coarse visual categories (CVC)

One disadvantage of the approach outlined in the previous subsection is that all images corresponding to a given fine label might not correspond to the same coarse category. This was observed during experimentation too. Thus, I construct coarse categories image by image in this case. For a given image, it corresponds to the neuron that gets activated the most in the coarse layer in Fig. 2. In this way, the images under the same coarse class are more visually similar than the CH method.

C. Method 3: End-to-End (ETE) trained neural network

This doesn't require any explicit coarse category construction and is trained end-to-end. It is hoped that this network will construct coarse categories by itself.

IV. TRAINING AND RESULTS

Data preprocessing: For training and testing, all images are first normalized so that the mean of each color channel is zero and the standard deviation is 1.

Further, the training set is flipped randomly in the horizontal direction and a random crop to 32×32 pixel size is employed with a padding of 4 pixels on each edge.

Train-Dev split: I split the 500 training images of each label into 400-100 train-validation set. Validation set is not fixed and is chosen randomly at every run of the code.

Optimation: For the training of all neural networks, an initial leaning rate of 0.01 is used. It is multiplied by a factor of 0.3 if the validation set loss doesn't decrease for 5 consecutive steps. L2 regularization is set to 0.001. I use stochastic gradient descent with momentum 0.9 for the optimization algorithm.

A. VGG16 Baseline

I choose VGG16 for baseline. The obtained accuracies are shown in Table. I.

VGG16 baseline on CIFAR100	
Dataset	**Accuracy**
Trainset	94.37%
Validation set	69.03%
Test set	68.74%

TABLE I. Baseline on VGG16 network after 81 epochs.

The training set accuracy is quite big compared to the validation and test set accuracies. This indicates a high degree of overfitting.

Note: I have avoided using a pretrained VGG16. In Fig. 1, the classification layers take input of size $512 * 10$ *which are not available as pre-trained. So, in order to avoid getting information from a state of the art pre-trained neural networks, I decided to use self-trained network so that the results are unbiased. One major trouble with this is the presence of overfitting during training. It can spoil the coarse neural network classification.*

B. Method 1: Category Hierarchy (CH)

A category hierarchy constructed by the neural network is shown in Table II. For this step, I don't use the baseline VGG16 as it suffers from severe overfitting. I employ early stopping to train another VGG16 that achieves 67.77% training and 62.50% validation set accuracies.

The coarse categories seem very sensible. For example, multiple types of trees and forest are classified in category 1. Vehicles are classified into category 2. Similarly, big carnivores in category 3, humans in category 4, sea animals and fish in category 6, furniture in category 7 and insects in category 9. Other categories don't seem to have a single identifiable pattern. One can only speculate why apple is classified alongside big cats.

Category Hierarchy	
Trees	chair, forest, maple tree, motorcycle, oak tree, orange, pine tree, snake, willow tree, worm
Vehicles	bus, cattle, fox, lawn mower, mountain, palm tree, pickup truck, streetcar, tank, tractor, train
Big Carnivores	apple, leopard, lion, tiger, wolf
Category 4	bowl, can, clock, keyboard, lamp, plate, raccoon, rocket, skunk
Humans	baby, bicycle, boy, cloud, flatfish, girl, man, plain, possum, sea, woman
Sea Animals	crab, dinosaur, dolphin, lobster, otter, ray, seal, shark, trout, turtle, whale
Furniture/ straight lines?	bed, road, table, television, wardrobe
Category 8	aquarium fish, bear, beaver, camel, chimpanzee, elephant, hamster, rose
Insects	bee, beetle, bottle, butterfly, caterpillar, cockroach, couch, cup, orchid, pear, poppy, spider, sunflower, sweet pepper, telephone, tulip
Category 10	bridge, castle, crocodile, house, kangaroo, lizard, mouse, mushroom, porcupine, rabbit, shrew, skyscraper, snail, squirrel

TABLE II.

A neural network with VGG13 architecture is trained to classify the training images into these 10 coarse categories. The accuracy achieved by the coarse classification neural network is shown in Table III. Quite surprisingly, the accuracies were generally good for the coarse categories that have a human identifiable pattern. They are shown in Fig. 3.

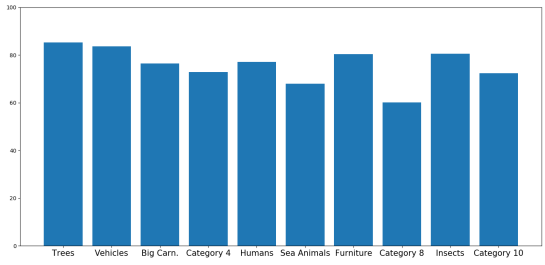

FIG. 3. Individual coarse class accuracy for coarse NN. Roughly, human recognizable categories have better accuracies.

One can clearly notice severe overfitting in the coarse classification. Since it is used in the training phase of the main neural network, it is bound to worsen the generalizability of the trained model.

The main neural network is now trained end-to-end with the trained coarse neural network. The coarse neural network is frozen during this phase. The classification accuracies are mentioned in Table IV.

Coarse NN accuracies	
Dataset	**Accuracy**
Trainset	82.09%
Validation set	75.90%
Test set	75.88%

TABLE III. Coarse NN accuracy after 31 epochs.

Overall accuracies	
Dataset	**Accuracy**
Trainset	84.46%
Validation set	69.90%
Test set	65.45%

TABLE IV. Main NN accuracy after 66 epochs

The neural network achieves 0.87% higher accuracy compared to the baseline on the validation set. However, it can be noticed that it doesn't generalize very well on the test set. As mentioned earlier, the reason for this is overfitting during coarse NN training phase. Another noticeable thing is that it doesn't overfit the training set as much as the baseline.

The minute improvement in the validation set accuracy leads us to speculate whether a better trained coarse classifier can actually improve the accuracy on the test set. Better training methods have to be used to achieve this. For example, following the approach of Ref. 2, one can do this by training the coarse NN partially followed by training the main NN. Then, both the coarse NN and main NN can be fine tuned.

C. Method 2: Coarse Visual Categories (CVC)

The training-validation set is used to train the coarse neural network. The accuracy obtained is shown in Table V.

Coarse NN accuracies	
Dataset	**Accuracy**
Trainset	81.92%
Validation set	77.72%
Test set	77.61%

TABLE V. Coarse NN accuracy after 32 epochs

The main neural-network accuracies are shown in Table VI

Overall accuracies	
Dataset	**Accuracy**
Trainset	85.01%
Validation set	67.22%
Test set	65.71%

TABLE VI. Main NN accuracy after 76 epochs.

It is surprising that this performs worse than CH model. I had expected the opposite because the images in this coarse category construction are supposed to be more visually similar. Also, the degree of overfitting and validation error were smaller for coarse NN in this case. One reason could be that the recovery layer in Fig. 2 overfitting.

D. Method 3: End-to-End (ETE)

For end-to-end training, I use a slightly modified neural network architecture as shown in Fig. 4. The coarse classification shouldn't require independent convolutional layers. Therefore, the activations of the convolutional layers are passed through the 3 fully connected linear layers of VGG13 to classify the image into 10 coarse categories. These are denoted by coarse classification layers in the figure. This is then used in a similar manner as earlier to compute correlations between convolutional layer activations and the coarse classification. The network is trained end-to-end.

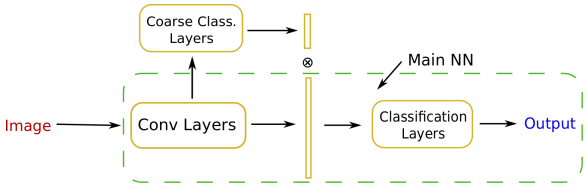

FIG. 4. End-to-End architecture

Activation for the last coarse classification layer has been chosen to be *softmax*. The classification accuracies are presented in Table VII.

Overall accuracies	
Dataset	**Accuracy**
Trainset	92.56%
Validation set	68.94%
Test set	68.42%

TABLE VII. Main NN accuracy after 67 epochs.

Upon analyzing the coarse classification layers, I found that virtually all images were classified into the same category. So, it is obvious that this method doesn't build a category hierarchy by itself. To achieve this, we can add a loss term to the model that incentivizes classification of images into different categories. After trying multiple simpler loss functions, I tried using the following for a

single mini-batch of images of size B:

$$c^{(i)} = \arg\max_j o_j^{(i)} \tag{1}$$

$$\sigma_c = \sum_{i, c^{(i)}=c} o^{(i)} \tag{2}$$

$$\mathcal{L} = \lambda_1 \sum_c \left(-||\sigma_c||^2 + \lambda_2 ||\sigma_c||^4\right) + \lambda_3 \sum_{c \neq c'} \sigma_c^T \sigma_c' \tag{3}$$

where $c^{(i)}$ represents the coarse class of the image "i" in a mini-batch, $o^{(i)}$ represents the 10-dimensional output of the coarse layers. Intuitively, this loss function incentivizes clustering between images that are visually similar. The first term is the Mexican hat potential that promotes bigger alignment of the coarse classification along the maximally activated neuron. The second term promotes smaller overlap between images corresponding different coarse classes. I haven't achieved much success in this for now. In retrospect, I spent a lot of time using the above loss function to do a different project on unsupervised image classification on CIFAR-10. I think a simpler loss function like the following could have worked for the purpose of this course project:

$$\mathcal{L} = \lambda \sum_{i \neq j} {o^{(i)}}^T o^{(j)} \tag{4}$$

V. CONCLUSION AND FUTURE WORK

In general, I didn't observe much improvement in accuracy in these neural network architectures. However, for CH and CVC methods, it seems that avoiding overfitting in the coarse neural network might lead to an increase in accuracy. Better training methods need to be used in this regard. One way of doing this is training coarse NN and main NN in cycles. One could train just the coarse NN for a small number of epochs, then train main NN for a small number of epochs. This would constitute one training cycle. After repeating multiple cycles, one could train both of them together to achieve the fully trained network.

A different approach than employed in this project would be to try to implement filter-filter correlations within a coarse category as suggested by Ref. 1. One way of doing it would be to consider the filter-filter outer product followed by a projection onto a smaller number of dimensions. And then this can be followed by an outer product with the coarse classification probabilities.

VI. ACKNOWLEDGMENTS

In addition to the great teaching team for the course, I benefited from discussions with my friends Shubham Toshiniwal and Palak Bhushan.

---

[1] Tsung-Yu Lin and Aruni Roy Chowdhury and Subhransu Maji. Bilinear CNN Models for Fine-grained Visual Recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

[2] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu. HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2740– 2748, 2015.

[3] `https://github.com/prashantkumar16/hicCNN`

[4] `https://github.com/kuangliu/pytorch-cifar`

[5] `https://gist.github.com/jeasinema/ed9236ce743c8efaf30fa2ff732749f5`