CS783-Assignment2

Paramansh Singh, Subham Kumar

160474, 160707

1 Problem Statement

In this assignment we are basically trying to tackle two problems:

- Given an image classify it in some coarse grained classes.
- Given an image classify it in fine grained classes.

Basically when we say fine grained classes these are actually sub-classes of coarse grained classes.

2 Dataset

Our dataset for coarse grained classification contains 5 classes viz. **aircrafts**, **birds**, **cars**, **dogs**, **flowers**. For fine grained classification each of these classes contain some sub-classes.So we have a total of 2423 images to train for coarse grained classification. These classes then have variable number of samples for fine grained classification.

3 Our Approach and Experiments

3.1 Coarse Grained Classification

For this part of our problem statement we used **Resnet-18** as our feature extractor.Rather than training it from scratch we used the pretrained model. But we didn't use the complete **Resnet-18** architecture rather we extracted the features from the average pooling layer which actually returned a 512 dimensional vector for each image.Then we added 2 FC layers,first taking these extracted feature as input and outputting a 128 dimensional vector and the second one taking this as an input and outputting scores for each class i.e. a 5 dimensional vector.For this part of model we used Cross-entropy Loss function along with Stochastic Gradient Descent with learning rate being 0.001 and momentum=0.9 with nesterov acceleration.
With the above mentioned approach we got very high accuracy and even got good results from some of the sample images not in the provided data.

3.2 Fine Grained Classification

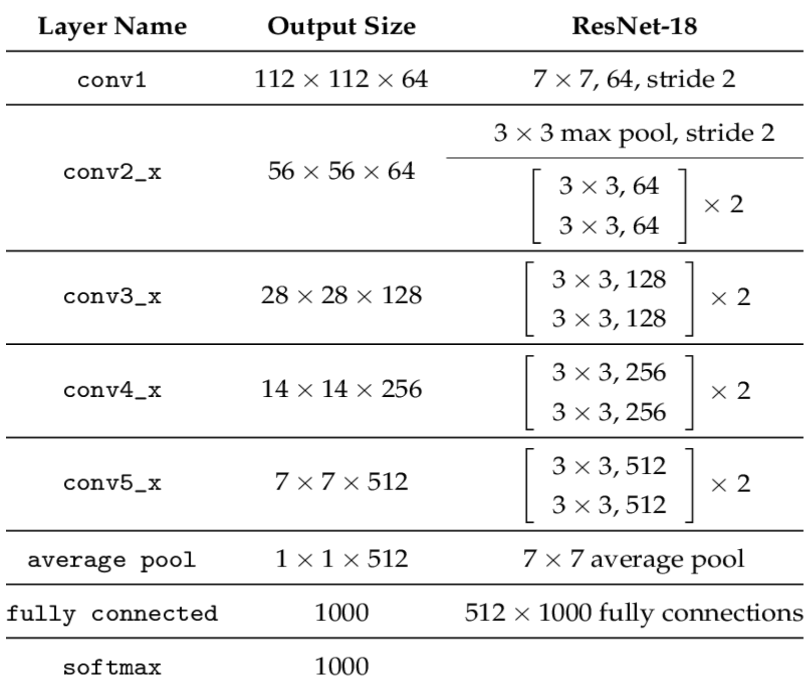
3.2.1 Resnet-18

Approach 1

This approach is mostly same as used for coarse grained classification. Here also we used `Resnet-18` as our feature extractor. We got 512 dimensional vector feature from the average pooling layer and fed it to our fully connected layer to classify in one of the 36 fine classes. We used 92-8 train-val split and got accuracy of around 89%.But validation loss doesn't seem to stabilize too much.

Approach 2

This approach first uses coarse grained classification and then the output of it is concatenated as one hot encoding with the 512 dimensional feature vector extracted from resnet-18 and then the model is trained for 36 classes.It actually improved the accuracy to 91.3% with the same train-val split as above.



Layer Name	Output Size	ResNet-18
`conv1`	$112 \times 112 \times 64$	7×7, 64, stride 2
`conv2_x`	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
`conv3_x`	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
`conv4_x`	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
`conv5_x`	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
`average pool`	$1 \times 1 \times 512$	7×7 average pool
`fully connected`	1000	512×1000 fully connections
`softmax`	1000	

Figure 1: This is an image of typical `Resnet18`. We extracted features from the average pool layer.

3.2.2 NTS-net

This approach is based mainly on Ze Yang et. al's "Learning to Navigate for Fine-grained Classification". It uses `NTS-net` or Navigator-Teacher-Scrutinizer Network. It is based on the fact that informative regions help us to better characterize the object and thus using these features can help to imorove prediction. Broadly, the navigator detects the informative regions in the image as directed by the Teacher (the teacher provides feedback for proposed regions). The Scrutinizer then makes predictions of the proposed regions. Below, we have described the method in a more detailed manner.

- **Navigator and Teacher:** The method used in this is inspired by Regional Proposal Network (RPN) proposed by Ren et. al. Broadly, the Navigator network takes an input an image and produces a bunch of rectangular regions at different scales and of different ratios. The network finally produces a list denoting the informativeness of the regions which are then sorted and top M regions are forwarded to the teacher network which returns the confidence associated with each region
- **Scrutinizer:** The scrutinizer takes in feature vectors of the image and concatenates it with the features of the proposed regions. Number of proposed regions is a hyper-parameter of the model. Note that the **same feature extractor model** (with same parameters) is used for extracting features for the whole image as well as the rectangular regions. The original paper used `Resnet-50` to extract features, but we used pretrained model of `Resent-18`

Approach 1

We first tried directly modelling all the 36 classes (fine-classes of all coarse classes modeled as separate classes) and pass the data through the NTS Network. We tried this because in both the coarse grained and fine-grained classification, the initial step is extracting features via the `Resnet 18` network and we felt that NTS-Net could learn about the coarse class structure as well.

Approach 2

Although the above method achieved great accuracy ($> 95\%$) on training and validation data, on some sample test cases (not from the data), it failed to predict even the correct coarse class. Hence we thougt of a more natural approach, in which we will first model the coarse classes and then have separate models for each of the fine-classes once we have the coarse class. It is important to note that in effect the feature extraction step is same for all the phases and we use the same set of `Resnet-18` features for all.

The separate model for each class better captures the *intra-class variation* in each class and *inter-class variation* is captured by the coarse grained network. Thus we expect better results from this hierarchical model than directly modelling each fine class.

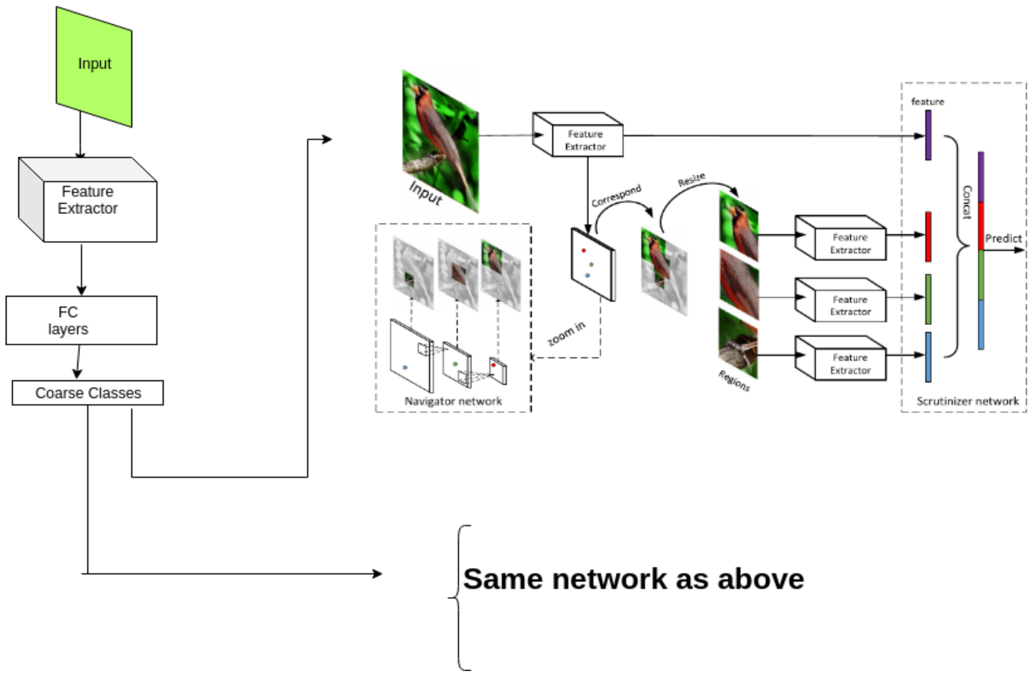


Figure 2: Approach 2

Approach 3

Another approach that we tried was using coarse grained classifier to predict the class and get a **one hot vector**. This one hot vector was fed to the network to model all subclasses via a single model. The vector was concatenated with the image features in the layer where concatenation of original-image-features and part-image-features take place. `Resnet-18` was used to extract the image features.

We tried this approach as this would help to provide a feature corresponding to the coarse class and we expected lesser coarse class mis-classification errors. We got approx. 93 % accuracy on 90-10 train test split with this data which was slightly less than 95 obtained directly without using this vector.

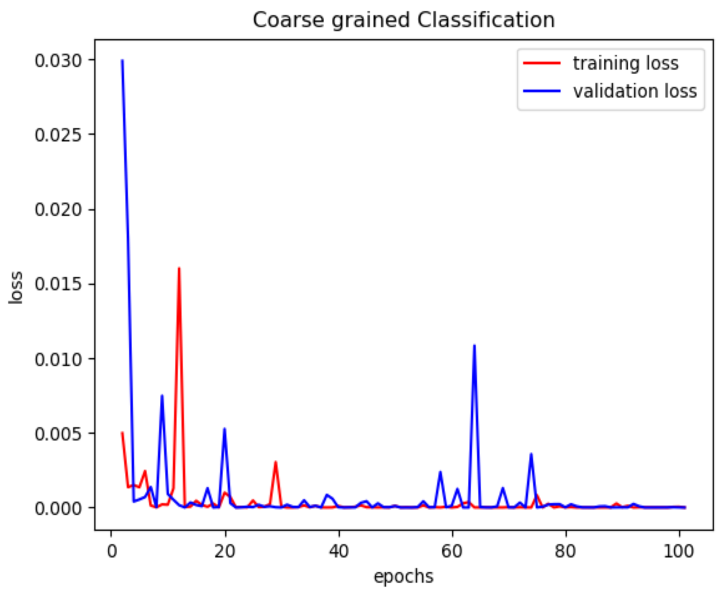
3.3 Final Model

Our final model is based on **Approach 3**. We did coarse grained classification using the method described initially and used modified NTS-net to model the 36 fine sub-classes. The network layers not described above are same as described in the model in the original paper. **The network's image is contained in the folder submitted.**

4 Results

4.1 Coarse Grained Classification

For coarse grained case we used 95-5 train/val split with batch size of 4 for training for 100 epochs.Both training and validation accuracy were in the order of 10^{-5}.The losses also seem to be stable.



4.2 Fine Grained Classification

Below are the results of training with 75:25 train test split. As it can be seen almost everytime, there is some overfitting, but in many cases test accuracy is also good (some mdels with less iterations had even better test accuracy)

Dataset	**Aircrafts**	**Birds**	**Cars**	**Dogs**	**Flowers**
Train Accuracy	0.99	0.99	1.0	0.99	0.99
Test Accuracy	0.97	0.85	0.90	0.90	0.99

5 Major Challenges and Improvisations

Overfitting: The main challenge in training the data was that of overfitting. With many parameters in the deep network model and very less data for each, especially in the *birds* class, fine-grained models almost always overfit on the data. In *aircrafts*, *dogs*, and *flowers*, we still got validation accuracy close to training accuracy, but the data always overfitted. For this we used dropout with probability 0.5 in the final layer of `Resnet-18` architecture which resulted in a slight improvement in the accuracy.
Modifying Network Architecture We had to change the model to reduce number of parameters as the default network modified the feature extraction parameters and used `Resnet-50` which had more parameters and resulted in more overfitting.
Concatenation of Coarse Classifier Output: This is the Approach 3 that we finally implemented.

References

[1] Deep Residual Learning for Image Recognition
[2] A survey on deep learning-based fine-grained object classification and semantic segmentation
[3] Learning to Navigate for Fine-grained Classification