Coarse & Fine grained Image Classification

Shubham Kumar Bharti (15807702)
Sunny Kant(15817740)

5th February 2019

Abstract

In this work we aim to learn various classification models to tackle the task of coarse and fine grained image classification. The given dataset consists of some coarse classes. Each coarse class has some fine classes, each of which in turn consists of multiple images of the fine class. The images in a coarse class is union of all images in its fine subclasses. The task can stated as follows: given some test images, you have to correctly classify it into its correct coarse and fine grained class.

1 Introduction

Image classification is one of important problem in Computer Vision where the task is to assign the correct class to a input image. Early methods to solve this problem consisted mainly of learning good feature vector of an image and then learning a classifier model on this feature vector. Despite being intuitive to understand, these models were not powerful enough to do classification anywhere closer to human performance. The first success of Deep Learning was demonstrated on ImageNet[DDS+09] Classification task in 2012 and with that there was a splurge in newer and better Deep Neural Networks that kept on improving the task of Image Classification finally surpassing human level performance in very short period of time. But Deep Learning methods required a lot of expensive computational resources and were also found to be data hungry. Training these models on small dataset lead to overfitting and the models are not able to generalize well to new unseen input instances.

2 Classification Models

We experimented with several Deep Learning models for the given coarse/fine grained classification task. These methods we employed are mentioned as follows.

2.1 Deep Neural Networks

Deep Learning methods using Convolutional Neural Networks(CNN) have been shown to perform very good in Image Classification tasks. CNN takes the advantages of spatial coherence of images and its parameter sharing mechanism using Convolutional filters dramatically reduces the number of models parameters and also enables it to be spatial/translation invariant. Morevoer, they also employ pooling layer which serves to downscale the images(although recent works has suggested that strided-filters work better to downscale images). Transfer learning methods using Deep Neural

Networks perform well with the amount/type of data given for this assignment and various deep models have been shown to achieve much better accuracy in Image Classification task. So, we have used some of the well known pre-trained models like ResNet[HZRS15], DenseNet[HLvW16], SqueezeNet[IHM$^+$16] etc. for feature extraction and further classification in an end to end manner. Keeping the number of model parameters constrains in mind, we have used following pre trained Deep Learning models.

2.1.1 ResNet

Ideally, a deeper neural networks should perform atleast as good as the shallow one because just by learning identity mapping for extra layers deeper neural networks can learn same function as its shallower counterpart. But in reality it was observed that the deeper neural networks start performing worse than the shallow ones because of vanishing gradient problem that increases with networks depth.

To solve this problem, the author introduced "identity shortcut connections" in between layers that expects the extra layers to learn just the identity mapping and hence perform at least as good as shallow network. It also alleviate vanishing gradient problem of very deep networks as the output of the residual part also has an additive input component. In our experiments we restrict to the use of ResNet18 and ResNet34 architectures.

2.1.2 DenseNet

DenseNet connects each layer to every other layer in a dense block(transition layer) in a feed-forward fashion. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. It alleviate vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters as comapred to ResNets. In our experiments, we have used DenseNet121 and DenseNet201 variants.

2.1.3 SqueezeNet

The building brick of SqueezeNet is called fire module, which contains two layers: a squeeze layer and an expand layer. A SqueezeNet stackes a bunch of fire modules and a few pooling layers. The squeeze layer and expand layer keep the same feature map size, while the former reduce the depth to a smaller number, the later increase it.

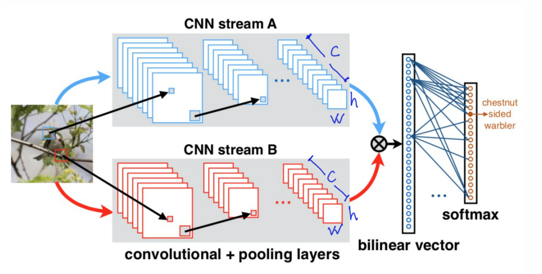
2.1.4 Bilinear CNN

In Bilinear CNN, two parallel CNNs are used for extracting features and their feature space are multiplied using outer product at each location and pooled to obtain the bilinear vector. This bilinear vector is further passed through a dense classification layer to obtain predictions. We have used ResNet18/34 and DenseNet121/201 variants as a feature extracter in the bilinear-CNN pipeline.

2.2 Variational Auto-Encoded Networks (VAE-Net)

This is a semi-supervised method which basically consists of two main steps namely encoding and classification. The idea here is that, we first encode the input data to a latent space and train a classifier on the latent space. The hypothesis here is that, in a latent space it will be easier for the network to learn a discriminator for different classes.

Figure 1: Bilinear CNN



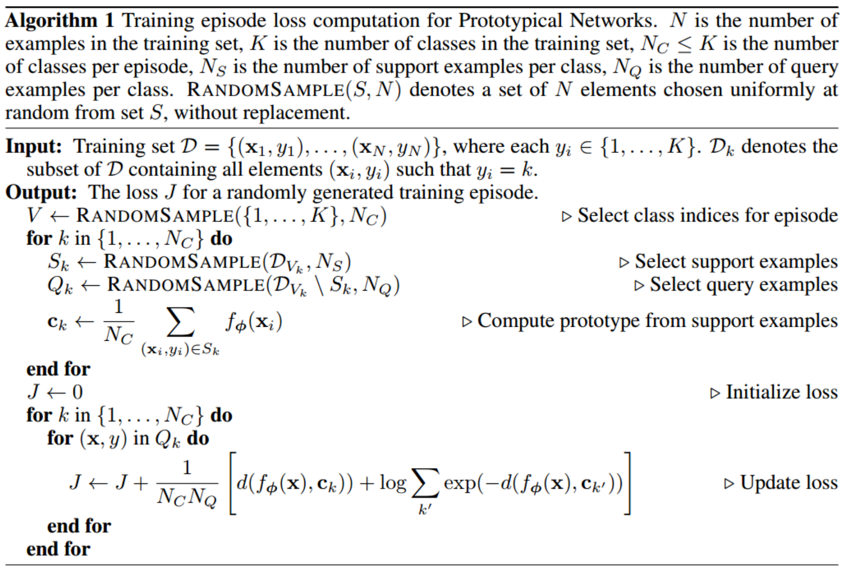
In the encoding step we use a Variational Auto-Encoding network to encode the image into a low dimensional input space. Next we learn a classifier on the outputs of latent space. The advantage of this model is that since the classifier is trained on a Variationally encoded input space, even a smaller classifier networks works well for classification task. Moreover, due to the variational properties of the encoded space, the classifier learns to generalize well in input examples. In our experiments we have used simple Resnet18 architechture for VAE encoder part and a 4 layer deconvolutional network for VAE decoder part. Further, we have used a dense feed forward network to learn a classifier using VAE encoded latent vectors.

2.3 Prototypical Networks (ProtoNet)

Prototypical Network[SSZ17] provide a very neat and intuitive framework for Few-shot classification, the task to learn to accommodate new classes not seen during training time, given only a few sample of images from the new class at test time. The method is based on the idea that there exists an good embedding in which points belonging to same class cluster around a single prototype representing the class. Once the prototype of all classes are obtained, the classification follows by assigning the input image to its closest prototype. Learning is achieved by minimizing the negative log likelihood of the true class k defined by $J(\phi) = -\log p_\phi(y = k|x)$.

The embedding feature mapping is obtained by using a Deep CNN architecture. In our experiments we have used pretrained resnet-34 architecture with finetuned learnable parameters to map an input image to dense embedding space. We further use euclidean distance metric to measure the embedding space similarity between prototype and embedded input points. We modified the training method of this algorithm to accommodate fine grained classification of images as few-shot learning problem. For each of the coarse classes we considered some of their fine grained classes to be unknown during the training and rest of their fine grained classes to be known during training. Essentially, the images of the classes available during training serve as support set and images of rest of the classes serve as query points. In our fine grained classification task, we do not expect new classes at test time(which was the case in few-shot learning), hence we choose to use the entire fine class dataset for training and validation.

Figure 2: Training Algorithm for Prototypical Networks



Algorithm 1 Training episode loss computation for Prototypical Networks. N is the number of examples in the training set, K is the number of classes in the training set, $N_C \leq K$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. RANDOMSAMPLE(S, N) denotes a set of N elements chosen uniformly at random from set S, without replacement.

```
Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the
  subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.
Output: The loss $J$ for a randomly generated training episode.
  $V \leftarrow$ RANDOMSAMPLE$(\{1, \ldots, K\}, N_C)$                ▷ Select class indices for episode
  for $k$ in $\{1, \ldots, N_C\}$ do
    $S_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k}, N_S)$                ▷ Select support examples
    $Q_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k} \setminus S_k, N_Q)$          ▷ Select query examples
    $\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$     ▷ Compute prototype from support examples
  end for
  $J \leftarrow 0$                                                  ▷ Initialize loss
  for $k$ in $\{1, \ldots, N_C\}$ do
    for $(\mathbf{x}, y)$ in $Q_k$ do
      $J \leftarrow J + \frac{1}{N_C N_Q} \left[ d(f_\phi(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'})) \right]$     ▷ Update loss
    end for
  end for
```

3 Experiments and Results

We split the dataset into train-val-test set in proportion 7:2:1. All the models are trained and validated using train and val set and final best model is selected using test set.

3.1 Coarse Classification Model:

3.1.1 Vanilla Methods

We have used DenseNet201 pre-trained model as feature extractor and fed the extracted features in the trainable fully connected layer for classification in an end to end manner. To adapt the pre-trained models to our task, we have removed last fully connected layer(classifier layer) of DenseNet201 and have connected feed-forward dense layer of output dimension 36(number of classes) in an end to end manner. We also fine-tuned other architectures like ResNet18/34, SqueezeNet and found them to work almost same on coarse gained task. The training accuracy obtained with Vanilla methods itself were found to be very good on testing data.

3.1.2 Adaptation/Experiments

To experiment further on coarse grained classification task, we implemented a simpler Variational Encoded classification method as described in the above section2.2. We observed that simple end-to-end ResNet18 architecture performed better than our two step classification method based on VAEs.

3.1.3 Accuracy

Test accuracy on Coarse Classification obtained on the above mentioned split of data are as follows.

Model	Test Accuracy
ResNet18	99.70%
ResNet34	99.80%
DenseNet201	99.80 %
VAE-Net	95.20 %

3.2 Fine Classification Models

3.2.1 Vanilla Methods

In the first part, we have fine-tuned pretrained vanilla ResNet34, DenseNet201 for fine grained classification of input instances to one of 36 classes. We have done tuning in two phases for DenseNet201 model, firstly we learned weights connecting features to fully connected layer only and then we fine tuned the whole model. We have implemented several extended versions of ResNet and DenseNet and found single layer dense network to perform equivalent to other dense deeper networks. In fine-grained task, the extended deeper networks started over-fitting after some training.

3.2.2 Adaptations/Experiments

Later we have trained Bilinear CNN models for fine-grained classification task. We have considered following two CNNs for ex-tracting features in the bilinear model.

1. ResNet18 and ResNet18 We extracted output of layer before classification for both parts.
2. DenseNet201 and ResNet34 We extracted features from denseblock2 layer in DenseNet201 and from layer 3 in ResNet34. Parameters upto denseblock2 layer is 1,429,632 and upto layer 3 in ResNet is 8,170,304.
3. ResNet18 and SqueezeNet121 We took features of layer8 of SqueezeNet1-1 after eliminating first row and column and output after layer 3 of ResNet18. Number of parameters upto layer8 in SqueezeNet1-1 is 120,416 and in ResNet18 is 2,782,784.
4. DenseNet201 and DenseNet201 We took features obtained from denseblock2 layer for both cases. Number of patameters are 1,429,632 each.

In each case, after bilinear convolution, fully connected bilinear of size 256*256 is mapped to fully connected layer of 32 labels. We observed that some of the fine grained classes in our dataset(such as those of cars) have very few smaple images and this is leading the Deep models like ResNet and DenseNet to perform poorly on them. To deal with scarcity of data, we tried a few-shot learning method called "Prototypical Networks"[SSZ17]. We modified the training procedure of Prototypical Networks to suit to our need as mentioned in above section2.3.

3.2.3 Accuracy

Test accuracy on Fine Classification obtained for the above mentioned split of data are as follows.

Model	Test Accuracy
ResNet34	90.30%
DenseNet201	96.50%
Bilinear(R18+R18)	89.40%
ProtoNet	86.50 %

We have obtained best accuracy in case of DenseNet201 model and used that for predicting labels of given test dataset. We have tried data annotations and fine tuning of all parameters of DenseNet201 and other hyperparameters to get better accuracy.

References

[DDS+09] Deng, J. ; Dong, W. ; Socher, R. ; Li, L.-J. ; Li, K. ; Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: *CVPR09*, 2009

[HLvW16] Huang, Gao ; Liu, Zhuang ; van der Maaten, Laurens ; Weinberger, Kilian Q.: Densely Connected Convolutional Networks. In: *arXiv e-prints* (2016), Aug, S. arXiv:1608.06993

[HZRS15] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep Residual Learning for Image Recognition. In: *arXiv e-prints* (2015), Dec, S. arXiv:1512.03385

[IHM+16] Iandola, Forrest N. ; Han, Song ; Moskewicz, Matthew W. ; Ashraf, Khalid ; Dally, William J. ; Keutzer, Kurt: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and &lt;0.5MB model size. In: *arXiv e-prints* (2016), Feb, S. arXiv:1602.07360

[SSZ17] Snell, Jake ; Swersky, Kevin ; Zemel, Richard S.: Prototypical Networks for Few-shot Learning. In: *arXiv e-prints* (2017), Mar, S. arXiv:1703.05175