CSC411 Assignment 3 Report

Name: Zhaoyue Cheng, Chundi Liu
Student Number:999563510, 1003655528
Kaggle Teamname: Racoon
Kaggle Username: SuperSuperSuperDeep, VeryVeryVeryDeep

(a) An introduction describing at a high level some approaches that you considered, and why you considered them

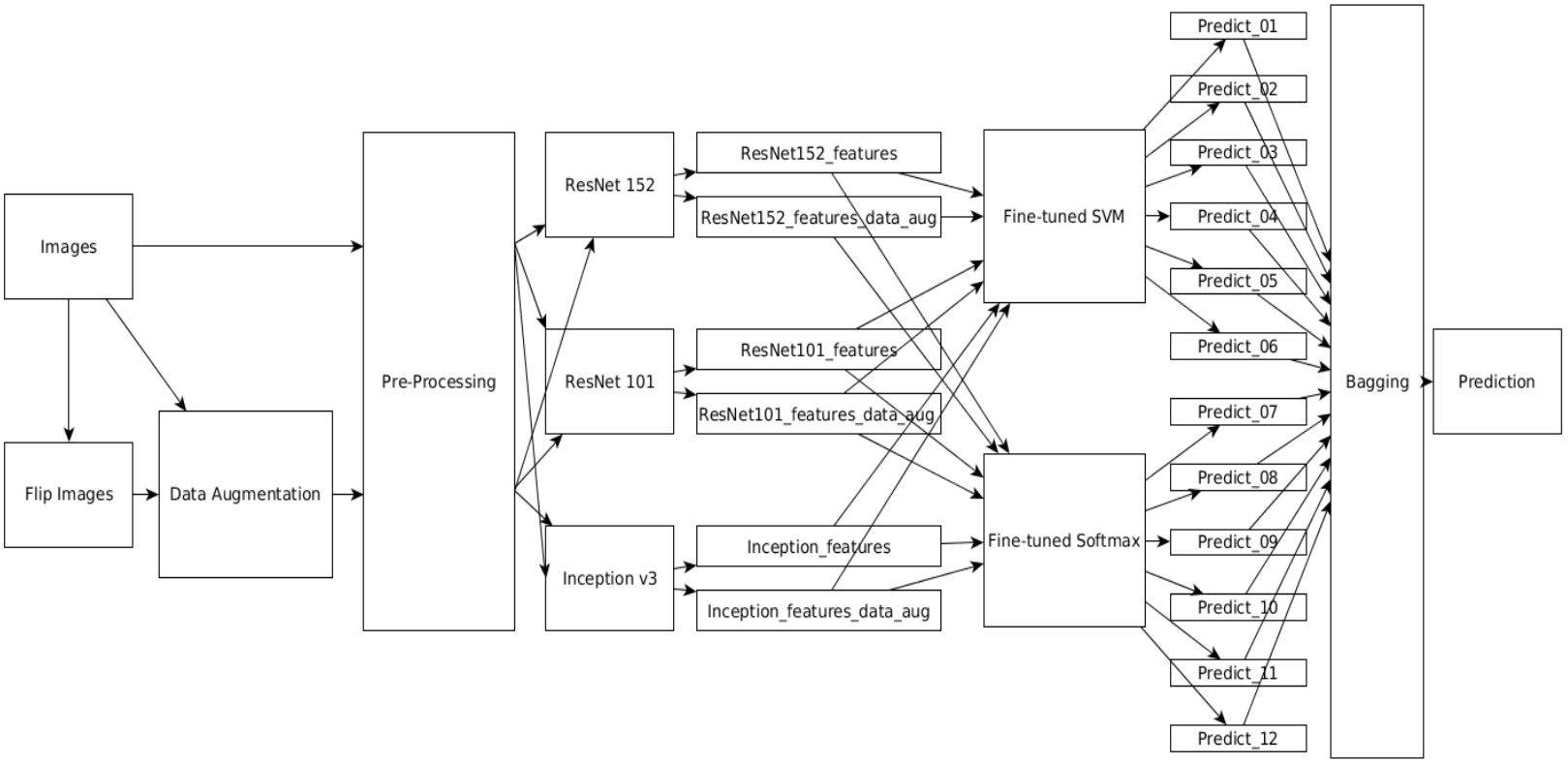
At the very beginning, we considered **Convolutional Neural Network (CNN)** since we have experience with it in the second assignment. It always produces good performance on image recognition problems and competitions. Since 2006, many methodologies have been developed to overcome the difficulties encountered in training deep CNNs. Most notably, Alex Krizhevsky proposed a classic CNN architecture (Alex-Net) and outperformed many previous methodologies on the image classification task. With the success of AlexNet, several works are proposed to improve its performance. In this project, we first built a CNN with a very simple structure. We used cross-validation to test the performance of our model. 75% of the data was used as the training data and the remaining 25% was test data. The whole training process took a long time and we got 42.061% accuracy on average.

We believe that the low accuracy is due to the relatively small size of our dataset, then we tried to use the ***pre-trained CaffeNet and ResNet-152*** to extract features (called CNN codes) in Caffe Framework and different classifiers to predict the labels. After several experiments, we finally decided to use a fine-tuned Softmax classifier. Then we got 64.845% and 84.124% accuracy respectively on public test dataset in Kaggle system.

In practice, very few people train an entire Convolutional Network from random initialization, because it is relatively rare to have a dataset of sufficient size. Also, the training process usually takes a long time even with help of multiple GPUs. Instead of training the model from scratch, people tend to use pre-trained models to extract features and then use different classifiers to classify the features. This method is known as Transfer Learning which is widely used nowadays, as it takes less time to train the models and at the same time produces higher accuracy in many applications.

In the final approach, we ***combine the idea of transfer learning with the idea of ensemble method*** discussed in class. We combined some of the best models including ResNet-101, ResNet-152 and Google Inception v3 as the feature extractors. Besides, considering the size of the dataset is relatively small and the skewed distribution of different classes, we flip all the images horizontally to get more training data. We use 6 groups of features in total, then we used SMV and Softmax classifiers, the two classifiers with the best performance, to predict image labels. Then, by utilizing bagging method, we choose the majority as the final prediction. Eventually ,we got a 85.670% accuracy on the public test set by using the ensemble model.

(b) A description of your submitted solution



Submitted Model Structure

My submitted solution uses a pretrained Resnet and some other ConvNets together without the fully connected layer. We treat it as a feature extractor, and then we train a SVM classifier and Softmax Classifier on top of it. We flip images to create a larger training set. Also, we used different Resnet (e.g, Resnet - 101, Resnet - 152) and Inception net V3, SVM classifier and Softmax Classifier with different parameters and then use the bagging method to get the final prediction. In the following paragraph, I will introduce different parts of the structure of my submitted solution.

We use the idea of Transfer Learning in our model. Basically, we use the ConvNet without the last fully-connected layer as the fixed feature extractor. Then we treat the rest of the ConvNet as a fixed feature extractor for the new dataset. For example, one of the feature extractors we used is Resnet-152, after passing the input image, we get a 2048-dimension feature vector which is the CNN codes. After extracting the features, we train a SVM classifier and Softmax classifier using the CNN codes for every image as the input data and the original output category as the output to get the final model.

I will use the Resnet-152 and Softmax Classifier in the following paragraph as an example to show how one of the models in the ensemble method works.

1) Image Preprocessing
   Since we do not have a lot of training data, we flip our images and resize our images to get more training data. Every time we flip our image or resize our image, we get a new set of training images so that we can have more training data for our model.

2) Pre-trained ConvNet as fixed feature extractor
   We used the pre-trained ConvNets (e.g. Resnet-152, Resnet-101, Inception v3) as our feature extractor by removing the last fully-connected layer. For every input image, we pass it into the pre-trained Resnet and compute the forward pass. I will use Resnet-152 as an example. We get a 2048-dimension feature vector each time after the forward pass which are the features extracted (i.e. the CNN codes).

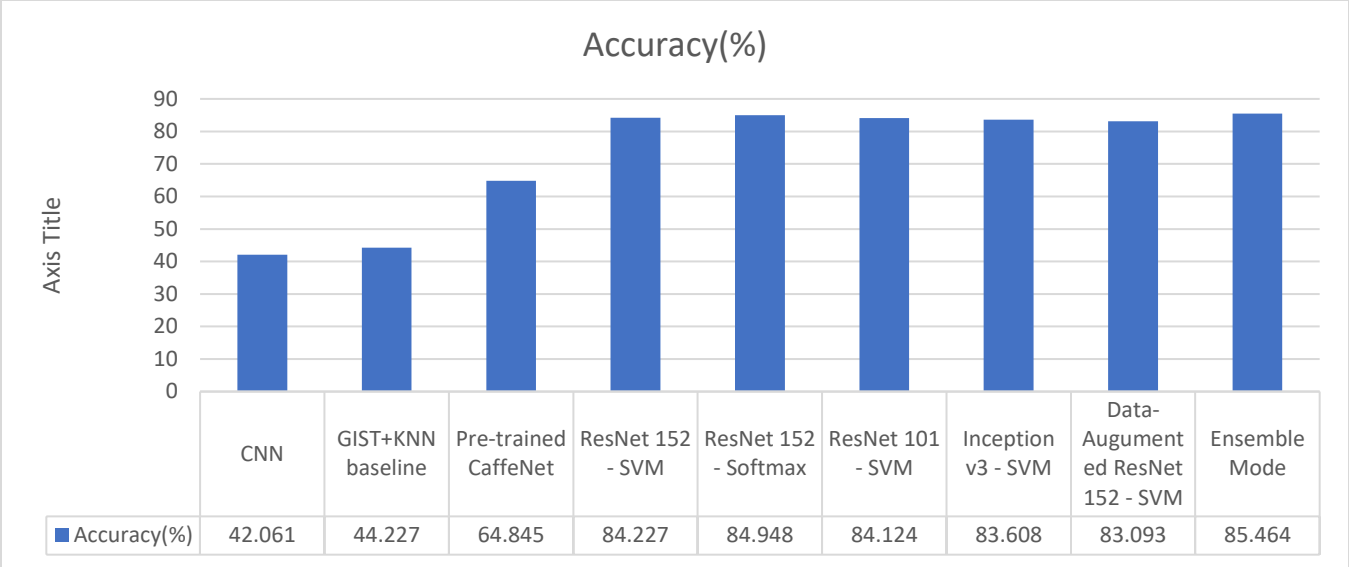
3) Softmax classifier
   After we get the 2048-dimension feature vector extracted using the Resnet-152. We train a Softmax classifier on top of it by using the feature vectors for every image as the input data and the original class label as the target data. To get the prediction, we first compute the forward pass on the test data to get the CNN codes (feature vectors) for the test data. Then we feed it into the SoftMax classifier trained to get the final prediction of class label.

4) Ensemble methods - bagging
   We used different Pre-trained methods on the same overlapping training sets, and then we average the predictions. We use pre-trained Resnet - 152, Resnet – 101 and Inception V3 on the same overlapping training data set. Then we train different SVM and SoftMax classifier for each of them with respect to the different CNN codes and the target labels. Finally, we use the bagging method to select the majority prediction among different models as the final prediction.

(c) Experiment Environment:

[EC2 instance: p2.xlarge] [CPU: Intel Xeon E5] [GPU: NVIDIA K80]

Model	GPUs	CPUs	Mem(GiB)	GPU Memory(GiB)
p2.xlarge	1	4	41	12

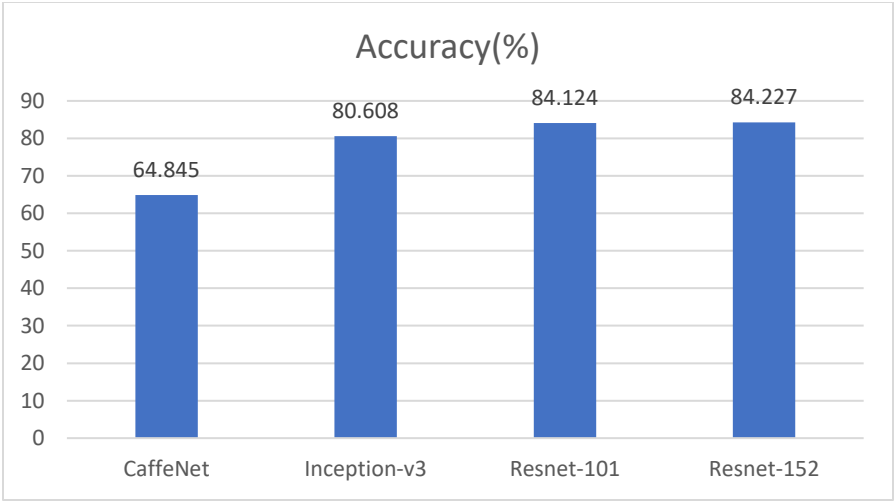


Take ResNet-152 as an example, we used different classifiers to predict the labels and fine-tune the parameter of these classifiers.

Design and analysis of experiments:

Experiment 1:
We tried different Pre-Trained Networks to see which one is the best.
Networks tried:
CaffeNet(AlexNet), Resnet-101, Resnet-152, Inception-v3
We used the above pre-trained networks without the fully connected layer as feature extractors as the CNN codes, and then we train a SVM with the same parameter(not tuned) on the CNN codes, and we use cross validation to compare the scores, and we use the mean of the scores as the final score.

Result:

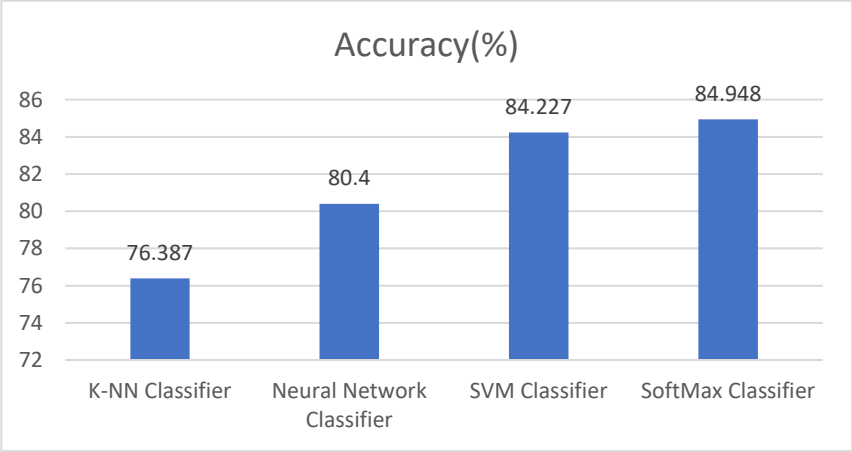


Experiment 2:
We tried different models on the CNN codes extracted by Resnet-152 to see which one is the best.
Networks tried:
K-NN, Neural Network, Linear SVM, Softmax

Parameters	150	Hidden(1000,)	Kernel: rbf Max-Iteration: 600	Class='multinomial' C=0.001 Solver='lbfgs' Max_iter=3000

We used the above networks to train on the CNN codes extracted by Resnet-152 on the test set, and we use cross validation to compare the scores, and we use the mean of the scores as the final score.

Result:



(d) Conclusion:
For image classification problems with relatively small training data set, utilize the idea of transfer learning is a more practical solution as the dataset is usually not large enough to train a well-designed deep convolutional neural network.

Also, using the idea of bagging can further improve model accuracy. However, the approaches are not independent so the ensemble method only improves the model a little bit.

We tried different approaches (e.g. train CNN, transfer learning, transfer learning with bagging). We get our best model by combining transfer learning and ensemble methods.

Bonus:

Overview:
We consider the classification problem in multi-labels case, in which case labels may have relationships with each other. In this part, we first implement a simple algorithm: training 24 different one-vs-rest classifiers. We get an accuracy of 94.286% in this case.

Formulation of the Problem:

The problem is that we have multiple labels for one image, then we want to train a model such that we get high a score using the given metric on the validation/test set.

$$a = \sum_i \sum_j \frac{I[pred_{i,j} = label_{i,j}]}{N_{sample} N_{classes}}$$

Model the label relationship in math:

The label relationship can be written into a conditional probability matrix,

$$\begin{pmatrix} p(C_1|C_1)\ p(C_1|C_2) \cdots p(C_1|C_j) \\ p(C_2|C_1)\ p(C_2|C_2) \cdots p(C_2|C_j) \\ \vdots \\ p(C_i|C_1)\ p(C_i|C_2) \cdots p(C_i|C_j) \end{pmatrix}$$

Within the matrix, if C_i, C_j have a co-occurrence relationship, we should have

$$p(C_i|C_j) \approx p(C_j|C_i) \approx 1$$

If C_i, C_j have a hierarchical relationship for example, if C_i is a subset of C_j,we should have

$$p(C_j|C_i) \approx 1$$

Experiment:

First, we use a 0-1 vector of length 24 with each one representing each label. For the 7000 training examples, we read in the data and form a matrix with shape [7000, 24], for each row, the value of the column corresponding to the label is 1 if the training example has the label, otherwise, the value of the column corresponding to the label is 0.

Then we train 24 binary SVCs with linear kernels to learn a discriminative model for each possible label based on the training input(the matrix specified above).

We split the training into training and valuation data sets and use the above method to train on the training data set. Then use the metric specified in the assignment:

$$a = \sum_i \sum_j \frac{I[pred_{i,j} = label_{i,j}]}{N_{sample} N_{classes}}$$

We get a final validation accuracy of 0.94286

Other thoughts:

Aside from the model described above, we have another idea of implementing the multi-label algorithm. Inspired by the idea in image captioning [reference 6], we want to use a recursive neural network combining with a CNN to learn the label sequences from CNN-code. The multi-label sequences would be learned as a simple version of image description. The RNN would learn the label relationship from ordered label sequences. However, we haven't finished this implementation when writing the report.

(e) References:

[1]"CS231n Convolutional Neural Networks for Visual Recognition", Cs231n.github.io, 2016. [Online]. Available: http://cs231n.github.io/transfer-learning/. [Accessed: 02- Dec- 2016].

[2]"Caffe | Deep Learning Framework", Caffe.berkeleyvision.org, 2016. [Online]. Available: http://caffe.berkeleyvision.org/. [Accessed: 02- Dec- 2016].

[3]"Caffe | Feature extraction with Caffe C++ code.", Caffe.berkeleyvision.org, 2016. [Online]. Available: http://caffe.berkeleyvision.org/gathered/examples/feature_extraction.html. [Accessed: 02- Dec- 2016].

[4] Tensorflow.org, 2016. [Online]. Available: https://www.tensorflow.org/versions/r0.12/tutorials/image_recognition/index.html. [Accessed: 02- Dec- 2016].

[5]"Multilabel classification — scikit-learn 0.18.1 documentation", Scikit-learn.org, 2016. [Online]. Available: http://scikit-learn.org/stable/auto_examples/plot_multilabel.html#sphx-glr-auto-examples-plot-multilabel-py. [Accessed: 02- Dec- 2016].

[6]A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1-1, 2016.

[7]"BVLC/caffe", GitHub, 2016. [Online]. Available: https://github.com/BVLC/caffe/wiki/Model-Zoo. [Accessed: 02- Dec- 2016].

[8]"CS231n Convolutional Neural Networks for Visual Recognition", Cs231n.github.io, 2016. [Online]. Available: http://cs231n.github.io/transfer-learning/. [Accessed: 02- Dec- 2016].

[9]"Caffe | Deep Learning Framework", Caffe.berkeleyvision.org, 2016. [Online]. Available: http://caffe.berkeleyvision.org/. [Accessed: 02- Dec- 2016].

[10]"Caffe | Feature extraction with Caffe C++ code.", Caffe.berkeleyvision.org, 2016. [Online]. Available: http://caffe.berkeleyvision.org/gathered/examples/feature_extraction.html. [Accessed: 02- Dec- 2016].

[11] Tensorflow.org, 2016. [Online]. Available: https://www.tensorflow.org/versions/r0.12/tutorials/image_recognition/index.html. [Accessed: 02- Dec- 2016].

[12]"Multilabel classification — scikit-learn 0.18.1 documentation", Scikit-learn.org, 2016. [Online]. Available: http://scikit-learn.org/stable/auto_examples/plot_multilabel.html#sphx-glr-auto-examples-plot-multilabel-py. [Accessed: 02- Dec- 2016].

[13]A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1-1, 2016.

[14]"BVLC/caffe", GitHub, 2016. [Online]. Available: https://github.com/BVLC/caffe/wiki/Model-Zoo. [Accessed: 02- Dec- 2016].