# Assignment 3 - Image classification

Tong ZHAO
Ecole des Ponts Paristech
tong.zhao@eleves.enpc.fr

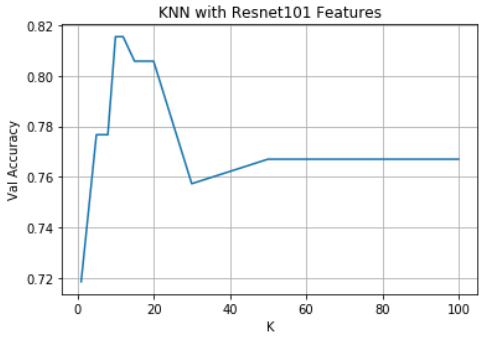
## 1. Baseline: Pretrained CNN as Features

Beginning with a simple thus elegant idea, I choose a method without specific learning step. The principal idea is to extract CNN features by a pretrained model based on ImageNet. Then a K-nearest neighbor classifier or a SVM is used to assign labels for the test dataset.

### 1.1. Models without Data Augmentation

I use Resnet101 model [1] as the pretrained model in this task since there is no overfitting issue. The dimension of the extracted layer, i.e. the last second fully-connected layer, is 2048.

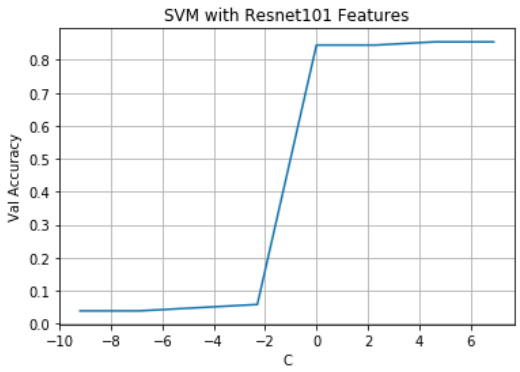
In order to predict label for a test example, I calculate the pairwise cosine distance between the test feature and all training features. Then the top $K$ nearest neighbors vote to obtain the final prediction. The following graph shows the influence of $K$ on val dataset. This model achieves an accuracy of $64.516\%$ on the public part of the test dataset.




```
Best K       :  10
Best Accuracy:  0.8155339805825242
```

Figure 1. The evolution of val accuracy with respect to the number of used neighbors $K$

I choose the support vector machine with gaussian kernel as the second model. It is a very efficient model when we don't have many training examples ($< 10000$). The most important parameter is $C$, which is the penalty parameter of the error term. The following graph shows the val accuracy with respect to the choice of $C$. This model achieves an accuracy of $65.161\%$




```
Best C       :  100
Best Accuracy:  0.8543689320388349
```

Figure 2. The evolution of val accuracy with respect to the penalty parameter $C$

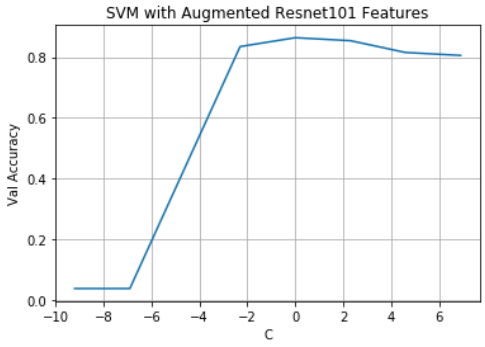
### 1.2. Models with Data Augmentation

One feasible way to improve the result is to add data augmentation step in the pipeline. Following the common operations in deep neural network, I augment the data by random cropping, random rotating, random flipping and random noising the image. Each training examples have 10 copies hence the training set is 10 times larger than the original one.

I re-run the algorithm with KNN and SVM. The result shows that the test accuracy of the new SVM is slightly better ($67.096\%$), whereas the one of the KNN remains the same.

## 2. Deep Learning: Finetune ResNet18

Since our task is a fine-grained classification problem for birds, my second idea is to train a specific model focusing on the bird features. In order to benifit the large feature sets of ImageNet, I initialize at first a pretrained ResNet18 model and finetune it with our training dataset. A randomly




```
Best C       :  1
Best Accuracy:  0.8640776699029126
```

Figure 3. The evolution of val accuracy with respect to the penalty parameter $C$ on logarithmic scale

initialized fully connected layer is added at the end of the model to classify features into classes. The most important point of this method is to control overfitting problem.

### 2.1. Data Augmentation

Even though I use ResNet18 as the model, the number of parameters is still too large compared to the number of training examples. In order to deal with this problem, I implement a dataloader which random resizes each training example at each epoch.

### 2.2. Add Dropout

Then I add a dropout layer after the last fully connected layer, namely the added classifier. The dropout ratio is considered as a hyper-parameter to tune in the experiments. A high dropout ratio means that the classifier should not relay on a large number of neurons, which results in a more stable model. Here I show in figure 6 the evolutions of the training loss, the training accuracy, the val loss and the val accuracy during the training process using the dropout ratio $0.5$ and $0.9$ respectively.

### 2.3. Learning Rate

The learning rate and the optimizer are important to train a good deep neural network. In the experiments I use always Adam [2], which depends less on the choice of the initial learning rate and its update policy during the whole training process. Since I use a pretrained model, the learning rate should not be high, otherwise it destories the learnt features. After several trials, $10^{-5}$ seems to be a good choice. The model achieves an accuracy of $78.064\%$ on the public test dataset.

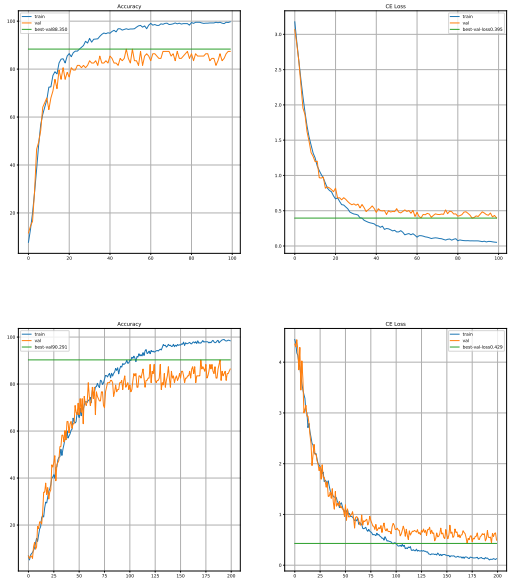


Figure 4. The training process of two models. The above figure corresponds to the one with dropout ratio $0.5$ and the below figure $0.9$.

### 2.4. Number of Layers

The depth of the network decides the model capacity. I replace the ResNet18 by ResNet152 while keeping the same parameters and strategies. The new model is huge thus it is difficult to train and to tune, but the advantage is that the model benefits a better initialization. The result shows, that although the training accuracy and the val accuracy remains the same as the previous one, I get an improvement of $4\%$ on the public test dataset.

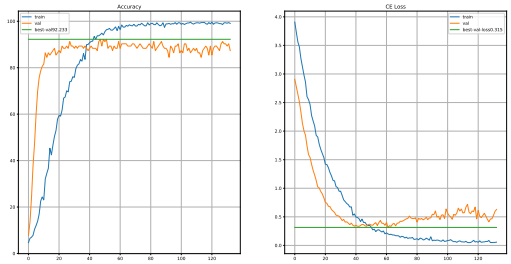


Figure 5. The training process of ResNet152 model

## 3. Joint Learning with Noisy Labels

Our task can also be considered as a noisy learning problem. To formulate the problem, I add test examples to the

training dataset by annotating them by a well-trained model, hence the training set contains noisy labels. The main idea comes from the paper [3]. Instead of updating net parameters in each epoch, I learn jointly the parameters and the training labels.

The whole training process contains two steps: in the first step I fix the target as the given labels. The model is optimized by ADAM using a small learning rate ($10^{-5}$). In the next step, I update after each epoch the target by taking the average of the softmax probabilities of the predictions from last 10 epoches. Two extra regularization loss are then added to cross entropy loss in order to prevent the model from getting stuck at a trival saddle point. The author claims that a high learning rate suppresses the memorization ability of a DNN and prevents it from completely fitting to labels. Hence I use a SGD optimizer with learning rate $0.01$ in this step.

The implementation is finished but I don't have enough time to do experiments on this model before the dealine but the experiment results show that it works. I will spend more time on this method in the following weeks to get a better understanding of the kind of problems.

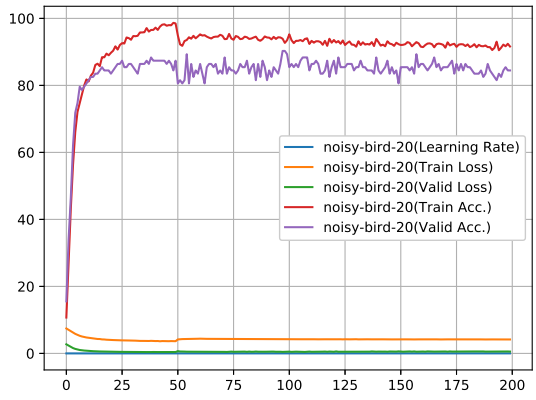



Figure 6. The training process of the joint learning model

## References


[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[3] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. Joint optimization framework for learning with noisy labels. *arXiv preprint arXiv:1803.11364*, 2018.