Deep Neural Network for Deep Sea Plankton Classification

Yuming Kuang
Stanford University
ymkuang@stanford.edu

Abstract

Deep sea plankton classification is a competition on Kaggle, named as 'National Data Science Bowl', aiming at automatically identifying plankton. In the project, we explored how to use deep convolutional neural network, together with data augmentation and regularization, to achieve the classification task. We also investigated using the 'code vector' (output of second last fully-connected layer) of CNN models to do model assembly. We introduced PCA, Ridge and Randomized Ridge model assembly methods and showed the Randomized Ridge model can outperform the simple probability averaging method. With single VGG-like [6 Conv + 3 FC] CNN model, we can get 73.90% accuracy and reduce log loss to 0.938 on test set. Using model assembly methods, we can further push to 75.80% accuracy and 0.857 log loss. Finally assembling 11 independent CNN models we achieved log loss 0.772707 on Kaggle.com for the full unlabeled test set and ranked 107th (1049 teams in total, top 1 log loss 0.560994).

1. Introduction

The project focuses on the National Data Science Bowl challenge on Kaggle.com, which aims at building an algorithm to automatedly identifying plankton using microscopic, high-resolution images captured by underwater imagery sensor system. (Figure 1) The algorithm can be then used by scientists at the Hatfield Marine Science Center to study marine food webs, fisheries, ocean conservation and other areas. The automated system has broad applications for assessment of ocean and ecosystem health, and can contributes to protecting the world's oceans. The competition has a reward of $\$175,000$ and 1049 teams compete for the prize [3].

Convolutional Neural Networks (CNN) are a good fit for the image classification problem. In this project, I explored using the CNNs with architectures similar to VGG [6], together with data augmentation, dropout regularization, leaky and parametric ReLU [4] activations and various model assembly methods, to achieve this classification task.

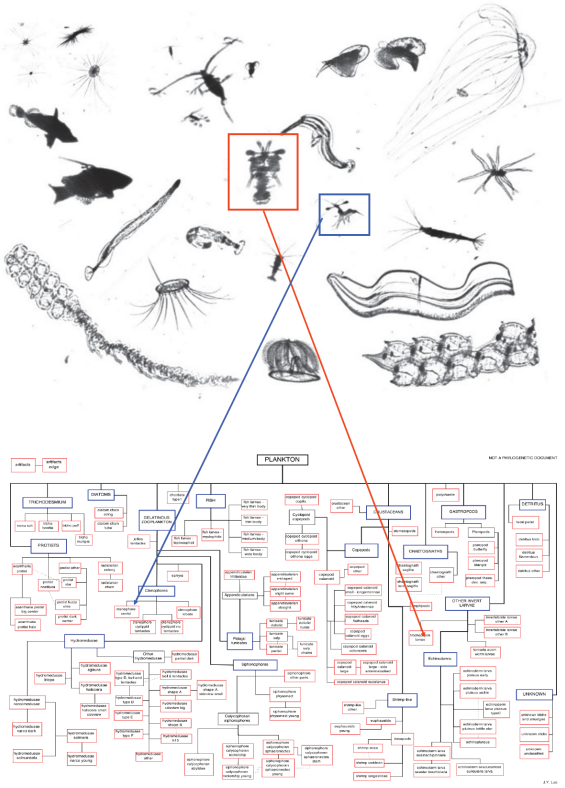

Figure 1. Plankton classification task[3]

The paper is divided into 3 parts. Section 2 focuses on the methods of CNN modeling and training, as well as introducing PCA, Ridge and Randomized Ridge model assembly. Section 3 introduces the data and evaluation criterion, the results using a single CNN model and the model assembly results. Section 4 gives a brief error analysis and discussions.

2. Methods

2.1. CNN Architecture and Training

The bricks of building the VGG architecture are convolutional layers, ReLU layers, max-pooling layers, fully-

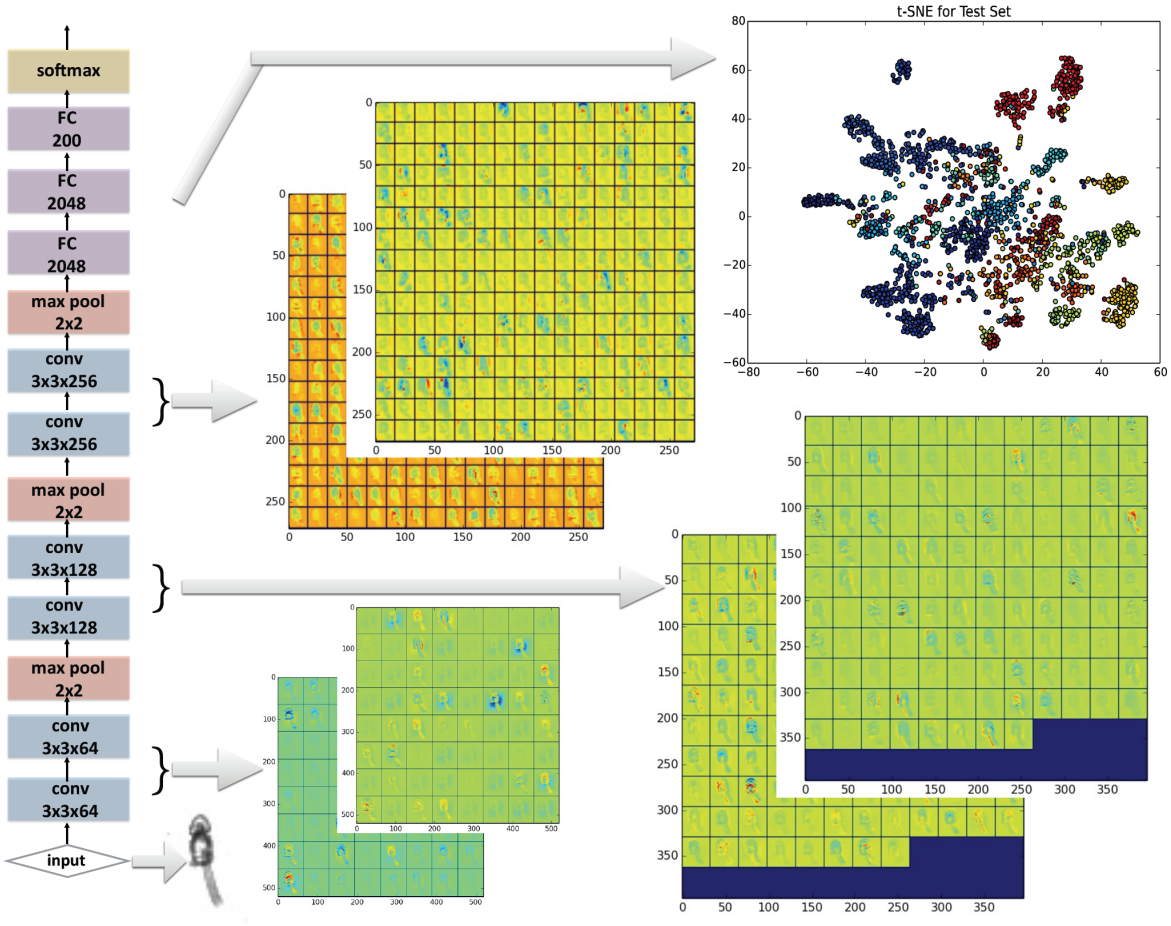

Figure 2. Visualization of CNN: The architecture of [6 Conv + 3 FC] CNN (left); The output of 6 convolutional layers for an example input (right); The t-SNE graph of code vector (the output of the second last FC layer) of the test set (top right).

connected layers and dropout layers. I used these bricks to build VGG-like CNNs from shallow to deep. The architecture can be separated into two parts:

Convolutional Part The convolutional part has structure of '[[[conv - relu] x m] - maxpool] x n'. For the convolutional layers, I found that kernel size 3x3 with pad 1 works the best. The ReLu layer is placed after each convolutional layer to introduce non-linearity. Max-pooling layers reduce the size of the filter result and aggregate the local information. I used 2x2 max-pooling in the models. The number of [conv - relu] before each max-pooling layer (m) is set to be 2 or 1.

Fully-connected Part The fully-connected part has structure of '[fc - relu - dropout] x k - fc - softmax'. The dropout layer is placed after each fully-connected to provide regularization and prevent over-fitting.

Since the key components of the CNN architectures are convolutional layers and fully-connected layers, we use the number of these 2 layers to specify the architectures. For example, '6 Conv + 3 FC' represents an architecture that has 6 convolutional layers and 3 fully connected layers, '[[[conv - relu] x 2] - maxpool] x 3] - [fc - relu - dropout] x 2 - fc - softmax' in particular (Figure 2 left).

The models are trained with Caffe [7] using the Nesterov optimization process [5] for Stochastic Gradient Descent. We also mention a few other setup details:

Data Augmentation The input images are resized to 80x80 and then a random crop of 64x64 and random flip are performed in the training process. I also tried random rotation of a multiple of 90 degree.

Dropout The dropout ratio is set to be 0.5. To see the influence of dropout layer on convolutional layers, I tried a model with adding a dropout layer after the last convolutional layer.

Leaky and Parametric ReLU The leaky ReLU layer activation function is $aI_{x<0} + I_{x\geq 0}$. The negative slope

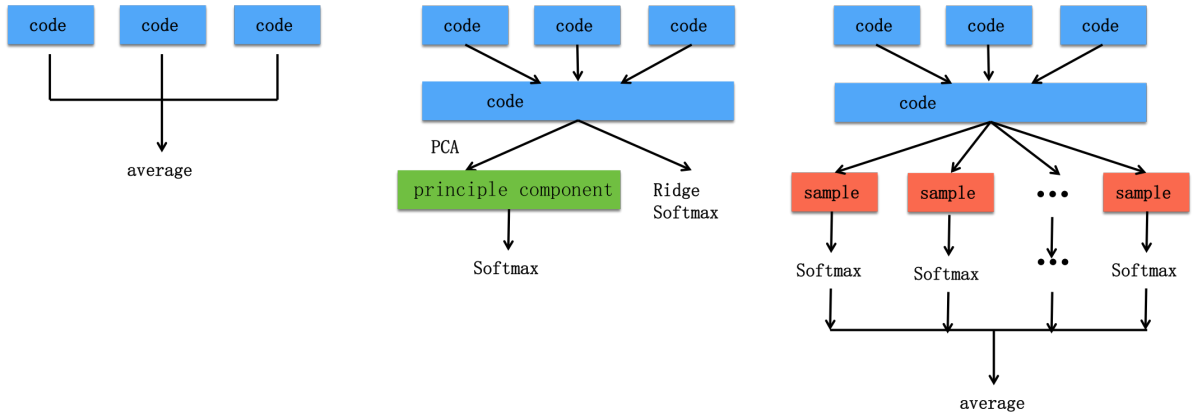

Figure 3. Model assembly methods. Probability averaging (left); PCA and Ridge model assembly (middle); Randomized Ridge model assembly (right).

a is set 0.1. As recently introduced by MSR [4], we can learn a during the back-propagation. In this case, I initialized $a = 0.25$.

2.2. Model Assembly

As introduced in the lectures, model assembly can reduce the variance of the result and thus improve prediction accuracy. In this project, I investigated several way to perform model assembly for CNN models:

Probability Average The simplest model assembly method is averaging the predicted probability of all the models (Figure 3 left). This simple idea actually works quite well since averaging greatly reduce variance.

PCA Denote the output of the second last '[fc - relu]' layer as 'code vector', then code vector provides summary of the features learned by the CNNs. The PCA model assembly process first concatenates code vectors from all the CNN models as extracted features, then computes the first K principal components of the extracted features, and trains a softmax classfier using the principal components finally (Figure 3 middle). Since the code vectors can be strongly correlated, the PCA step provides dimension-reduced, uncorrelated summary of the original code vectors, thus reduces the variance of predictions.

Ridge The L2-penalty can be also use to cure correlated feature, so the ridge model assembly process works similarly to PCA model assembly, except for training a softmax classifier with L2-penalty with all the code vectors, instead of the principal components (Figure 3 middle).

Randomized Ridge The problem of PCA and Ridge model assembly is that they end up with only one classifier for prediction and the classifier might still suffer from high variance. To solve this, randomized ridge model assembly process trains many ridge softmax classifiers on subset of sampled features from the code vectors and then the results of the classifiers are aggregated to get more stable predictions (Figure 3 right). Subsampling the features makes each classifier weaker but less correlated with each other, thus gives better result when aggregating. Compared to Probability Average, the randomized ridge process can aggregate many more models to further reduce variance. This idea of aggregating many weak classifier is similar to the Random Forest algorithm [1], while replacing decision trees with softmax classifiers.

3. Result

3.1. Data and Evaluation

The competition provides ∼30k labeled training images from 121 classes and ∼130k unlabeled testing images. To compare and evaluate different models and methods, I split the labeled data into 3 parts, training set (∼26k), validation set (2k) and test set (2k). As mentioned in the data augmentation process, images are resized to 80x80 and then random crop and flip are performed. The models are trained with the training set and validation set, and accuracy and log loss are evaluated using the test set. I will also report the loss result of the entry in Kaggle competition using models training from the full labeled training set.

We use both accuracy and log loss as evaluation criterion. The log loss is

$$\textbf{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N} log(p_{iy_i})$$

where N is the size of validation or test set, and y_i is the

Method	Val Loss/Acc	Test Loss/Acc
Softmax	/45.63%	/44.43%
2 Conv + 2 FC	1.180/65.15%	1.160/66.90%
5 Conv + 3 FC	0.922/72.10%	0.943/73.30%
6 Conv + 3 FC	0.936/72.65%	**0.938/73.90%**
6 Conv + 3 FC (w/ Leaky ReLU)	**0.902/73.70%**	0.959/73.55%
8 Conv + 3 FC	0.944/72.50%	0.988/73.10%
8 Conv + 3 FC (w/ dropout in last conv)	0.935/72.70%	0.9607/72.05%
8 Conv + 3 FC (w/ rand rotate & param ReLU)	1.708/53.05%	1.809/51.35%

Table 1. Log Loss and Accuracy Result for Single CNN models

true class label for sample i.

3.2. Result of Single Model

3.2.1 Result of Log Loss and Accuracy

The log loss and accuracy result of CNNs models with various architectures and the softmax linear classifier baseline is summarized in Table 1. The baseline is trained with HOG and color histogram features. We make the following analysis about the result:

- The CNN models perform much better then the linear baseline. This shows that CNN extract and aggregate the features of images in a better way than linear baseline.
- Among different architectures, [6 Conv + 3 FC] performs the best on the test set. The models with only 2 and 5 convolutional layers don't provide enough representation power, while the model with 8 convolutional layers suffers from over-fitting.
- To solve the over-fitting problem of [8 Conv] model, I tried adding dropout layer after the last convolutional layer. The dropout layer helps improve the performance of log loss a little, but doesn't improve accuracy.
- Another way to solve the over-fitting problem is to add random rotation in the data augmentation process. I tried this with the [8 Conv] model with parametric ReLU hoping the new ReLU layer can provide more power and faster learning speed. However it leads to a much worse performance. This might come from that random rotation to a multiple of 90 degree adds too much noise to the training set. Random rotation with much smaller degree might work better in this case.
- To improve the performance of [6 Conv + 3 FC] model, I tried using leaky ReLU layer to gain more representation power and faster learning process. It turns out to get a better result in validation set, but doesn't generalize to the test set.

3.2.2 Visualization of CNN Output

To get a better idea of how the [6 Conv + 3 FC] model works, we visualize convolution layers and fully-connected layers in Figure 2. To visualize convolution layers, I collected the output of each convolutional layer for an example and plotted them out. It clearly illustrates how convolutional layers capture image features from local to global scale. For the fully-connected layers, I plotted the t-SNE [2] graph for the code vector, i.e the output of the second last fully-connected layer, to project and visualize the high-dimensional features in 2d space. From the t-SNE graph, we can see that the code vector provides good characteristics to separate and represent each class.

3.3. Result of Model Assembly

To evaluate the model assembly methods, we take 4 models of structure [6 Conv + 3 FC] independently trained on the training set. Then for training, validation and test set, the code vectors of all the 4 models for each sample are collected and concatenated as the extracted features. Then we use the code vectors as input for the following model assembly methods. The total dimension of the code vectors is $2048 \times 4 = 8192$. For tuning hyper-parameter for PCA and Ridge model assembly method, the log loss is used as criterion, because we want to improve the loss which Kaggle uses to rank teams.

3.3.1 PCA

For PCA model assembly method, the number of principal components K for training the classifier is tried with 128, 256, 512, 768, 1024, 1536, 2048, 3072, 4096. The log loss of validation and test test of the softmax classifier trained with each K is shown in Figure 4 (left). We can see that the loss achieves its minimum at around 768, which illustrates the intuition that too small K doesn't extract enough information from the code vector and leads to under-fitting, while too large K doesn't reduce the noise enough and thus over-fitting. The result of K=768 is used in the final comparison.

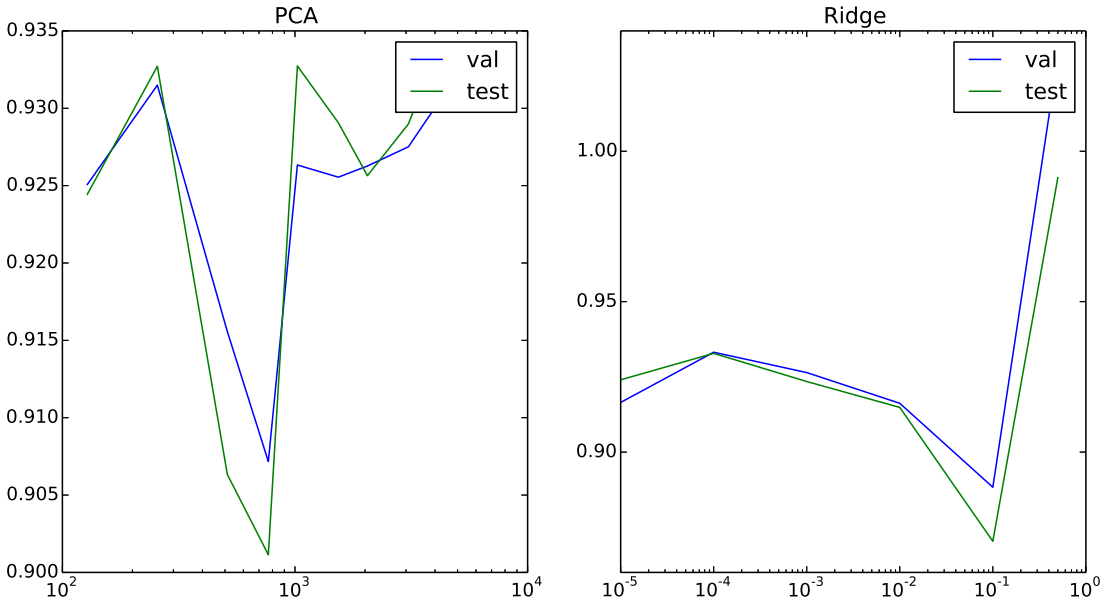

Figure 4. The log loss curve of validation and test set. PCA (left); Ridge (right).

3.3.2 Ridge

Similar to PCA, in ridge model assembly method, we need to tune the L2-penalty regularization term λ for the softmax classifier to achieve minimal loss. If λ is too small, the classifier suffers from correlated feature, while if λ is too large, the regularization is too strong and results in under-fitting. I tried λ with 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 5e-1 and plotted the log loss for validation and test set shown in Figure 4 (right). The loss achieves its minimum at λ=1e-1, and we use the model for λ=1e-1 for final evaluation.

3.3.3 Randomized Ridge

For randomized ridge model assembly method, each time about $\frac{1}{8}$ of the features in the code vector are subsampled and feeded into a softmax classifier with regularization term $\lambda = 1e{-}4$. We trained 100 softmax classifiers in total. Each time a new classifier is trained, we updated the accuracy and log loss for test set. Figure 5 shows the curve of accuracy and log loss for test set compared to the result of probability averaging of the 4 CNN models. We can see the clear improving trend when aggregating more classifiers, and the final result of aggregating 100 models outperforms the simple probability averaging.

3.3.4 Summary and Comparison

The result of the model assembly methods is summarized in Table 2. We make the following comments:

- PCA and Ridge greatly improve the result over the single model, although they are still predicting with one softmax classifier. This implies that PCA and Ridge method are helpful in reducing the model variance.
- The probability averaging baseline outperform PCA and Ridge especially in log loss, which implies that the single classifier of PCA and Ridge still suffer from model variance.
- Randomized Ridge outperforms the simple probability averaging baseline in both log loss and accuracy. The reason behind is that the classifier with subsampled features already has lower variance, and we can aggregate with many of these weaker but more stable classifiers, thus improve predicting power as well as further reduce variance.

3.4. Result in Unlabeled Test data on Kaggle

We trained 11 different CNN models with the number of convolutional layers 5, 6 or 8, using smaller training data set or full labeled training data set. Then the models are assembled using the simple probability averaging method on the full unlabeled test data set ($\sim$130k). The result was uploaded to Kaggle.com as an entry for the competition.

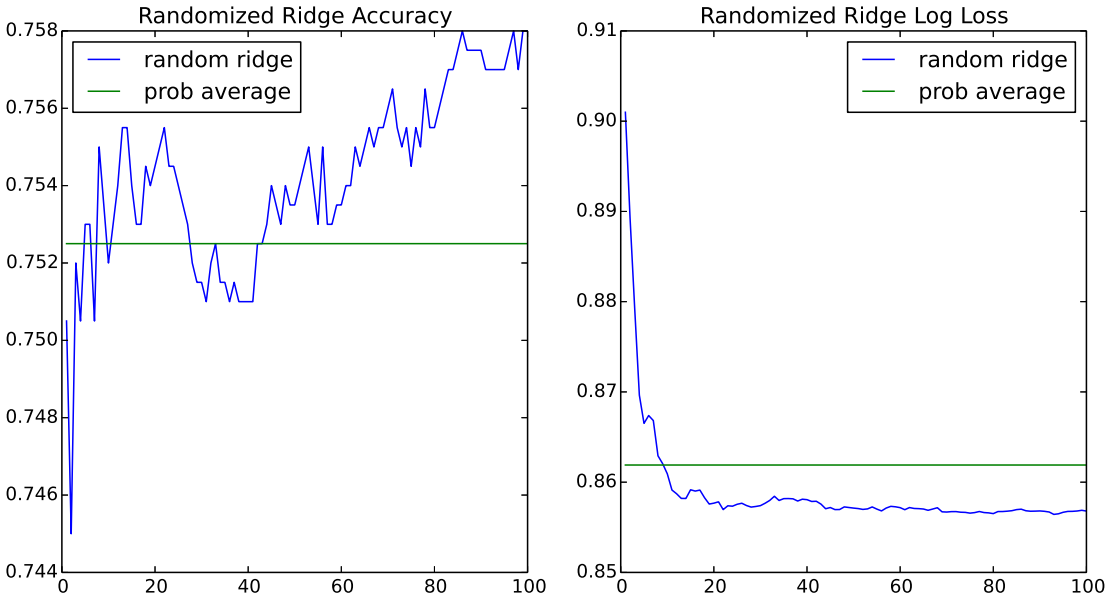

Figure 5. The accuracy (left) and log loss (right) curve on test set of Randomized Ridge model assembly methods.

Method	Test Loss	Test Acc
Single 6 Conv + 3 FC	0.938	73.90%
Probability Averaging	0.862	75.25%
PCA	0.901	75.2%
Ridge	0.870	75.00%
Randomized Ridge	**0.857**	**75.80%**

Table 2. Log Loss and Accuracy Result for Model Assembly Methods

The log loss I got is **0.772707** and I ranked 107th place (top 1 team got log loss 0.560994).

4. Error Analysis

We illustrate the following error sources with examples in Figure 6:

- The image of plankton is just hard to identify. For example, (b) is from 'unknown_blobs_and_smudges', however it just a dark spot and hard to identify, so the classifier can't predict it with certainty.

- The image is labeled as 'unknown' but the image looks like some plankton. For example (a) is labeled as 'unknown_sticks', while it looks like a living plankton with head in its front from the image.

- The image is obscure. For example (d) is from 'appendicularian_s_shape', but the camera seems to take the picture in a strange direction or the plankton is moving fast that the image is obscure to determine.

- The image is taken from a different perspective that is rare in the training set. For example (c) is from 'chaetognath_other', while the image is different and darker from other images in the same class, which implies that the image might be taken from a rare perspective.

5. Conclusion

In this project, we demonstrate that convolutional neural network works well for image classification task from its strong representing power and well-organized structure to extract image features from local and aggregate to global scale. Data augmentation and regularizations like dropout are also important for CNN model to prevent over-fitting and generalize better. We also investigate different model assembly methods. PCA and Ridge softmax classifier can

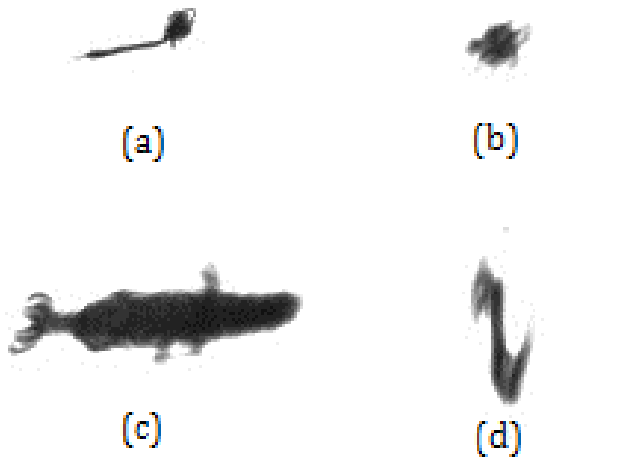

Figure 6. Example images for error analysis

be applied to the code vector and reduce model variance. Simple probability averaging provides extremely good improvement of model assembly, while Randomized Ridge model assembly performs better with training many more reliable weaker softmax classifiers using the code vector of CNN.

References

[1] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] L. V. der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[3] Kaggle. National Data Science Bowl. `http://www.kaggle.com/c/datasciencebowl`, 2014. [Online; accessed 15-March-2015].

[4] H. Kaiming and et al. Delving deep into rectifiers:surpassing human-level performance on imagenet classification. `http://arxiv.org/pdf/1502.01852v1.pdf`, 2015.

[5] Y. Nesterov. A method of solving a convex programming problem with convergence rate o(1/k2). *In Soviet Mathematics Doklady*, 27:372–376, 1983.

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition[j]. *arXiv preprint*, page arXiv:1409.1556, 2014.

[7] J. Yangqing. Caffe:an open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org`.