

# CS583A: Course Project

Name 1, Name 2, and Name 3

May 19, 2019

## 1 Summary

[Problem descriptions:] We participate an active (or inactive with late submission) competition of classifying cats and dogs based on photos. [Methodology:] The final model we choose is ResNet50, a deep convolutional neural network architecture, which takes  $256 \times 256$  images as input and outputs the class labels. [Implementation:] We implement the convolutional neural network using Keras (or we directly use the ResNet50 provided by Keras) and run the code on a MacBook Pro with one Intel i7 CPU and 32 GB memory (or a workstation with 4 NVIDIA GeForce XXX GPUs.) [Evaluation metric:] Performance is evaluated on the classification accuracy. [Score and ranking:] In the public leaderboard, our score is 0.95123; we rank 79 among the 215 teams. In the private leaderboard, our score is 0.95234; we rank 82 among the 217 teams. (Or, the result on the public leaderboard is not available until Month Day Year.)

[Note: In your report, remove the words in brackets.]

[Note: If you do not have LaTeX editor on your laptop, you can use the free online editor, *OverLeaf*.]

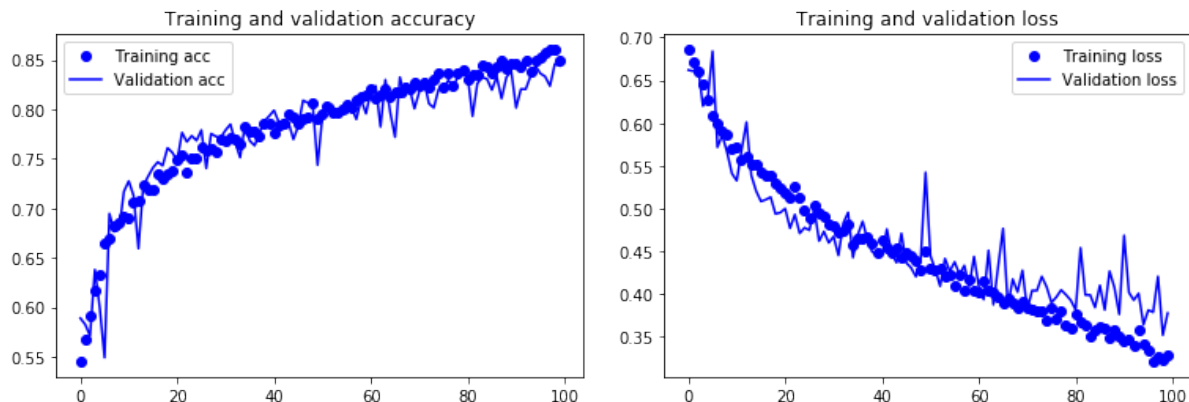
## 2 Problem Description

**Problem.** The problem is to classify cats and dogs based on photos. This is a binary classification and image recognition problem. The competition is at <https://www.kaggle.com/c/dogs-vs-cats>. [Use your own words to give a very short description. Do NOT copy the description on Kaggle.]

**Data.** The data are  $256 \times 256$  JPEG images. The number of training samples is  $n = 25,000$ . The number of classes is 2. The training set is well-balanced:  $n_{\text{dog}} = 12,500$  and  $n_{\text{cat}} = 12,500$ .

**Challenges.** [What makes the problem challenge? E.g., the training set is too small, the data is imbalanced, etc.]

[If your project is different from your proposal, add a section here to justify. Without a convincing justification, you may lose 2 points.]



(a) The classification accuracy on the training set and validation set. (b) The loss on the training set and validation set.

Figure 1: The convergence curves.

### 3 Solution

**Model.** The model we finally choose is the ResNet50 [1], a standard deep convolutional neural network. A description of ResNet is online: [https://en.wikipedia.org/wiki/Residential\\_network](https://en.wikipedia.org/wiki/Residential_network). [Describe your model only if it is non-standard.]

**Implementation.** We implement the ResNet50 model using Keras with TensorFlow as the backend. [Or, we directly use the ResNet model provided by Keras.] Our code is available at <https://github.com/wangshusen/CS583A-2019Spring/>. We run the code on a MacBook Pro with one Intel i7 CPU and 32 GB memory (or a workstation with 4 NVIDIA GeForce XXX GPUs.) It takes 2.2 hours to train the model.

**Settings.** The loss function is categorical cross-entropy. The optimizer is RMSprop. [Specify the other hyperparameters such as learning rate, regularization, epochs, batch size, etc.]

**Advanced tricks.** [If you used advanced tricks, e.g., pretrain the model on ImageNet and fine-tune it on the Dog-VS-Cat dataset, then write down your tricks.]

**Cross-validation.** We tune the parameters using a 5-fold cross-validation. [If you do not have GPU, you can simply partition the training data to 80%-20% or 90%-10% for hyperparameter tuning.] [Just show the results of your final chosen model.] Figure 1 plots the the convergence curves on 80% training data and 20% validation data. [If you use 5-fold cross-validation, just plot one of the five folds.] [Analyze the convergence curves. E.g., if the training accuracy is much better than the validation accuracy, then the model is likely to overfit the data, and that is why you pretrain your model on ImageNet.]

## 4 Compared Methods

[Hint: Try different methods (with brief descriptions) and report their performance. You will lose up to 3 points if you do not compare with baselines. Here are examples.]

**ResNet.** We use the ResNet50 model provided by Keras. Its parameters are pretrained on the ImageNet dataset. The training and validation accuracies are respective 80.0% and 75.0%.

**Fully-connected neural network.** We implemented a 3-layer fully-connected neural network. The width of the layers (from bottom to top) are respectively 256, 512, and 10. We apply dropout/-batch normalization to the second layer. The training and validation accuracies are respective 80.0% and 75.0%.

**Random forest.** We use the random forest model provided by SKlearn. We set xxx to xxx. The training and validation accuracies are respective 80.0% and 75.0%.

**Advanced tricks.** Our finally adopted method is a siamese network with 4 convolutional layers. We applied the following tricks.

- Data augmentation. Without data augmentation, the training and validation accuracies are respective 80.0% and 75.0%. With data augmentation, the training and validation accuracies are respective 82.0% and 77.0%.
- Over-sampling. Because the data is imbalanced (much more negative samples than positive samples), we tried over-sampling and under-sampling and found over-sampling works better. Using both data augmentation and over-sampling, the training and validation accuracies are improved to 85.0% and 79.0%.
- Ensemble. Using data augmentation, over-sampling, and ensemble, the training and validation accuracies are improved to 87.0% and 83.0%.
















## 5 Outcome

[Top 30% is considered good. Top 30 to 60% is considered medium, and you may lose 1 point. Bottom 40% is considered low, and you may lose 2 points. Use screenshots as evidence.]















We participated in an active competition. Our score is 0.89311 in the public leaderboard and 0.90112 in the private leaderboard. We rank 100/312 in the public leaderboard and 101/312 in the private leaderboard. The screenshots are in Figure 2.

## References

- [1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

|     |    |                              |   |         |    |     |
|-----|----|------------------------------|---|---------|----|-----|
| 92  | —  | Breck                        |    | 0.83791 | 6  | 5mo |
| 93  | ▲2 | praveen kumar                |    | 0.83609 | 12 | 5mo |
| 94  | ▼1 | Akila Wajirasena             |    | 0.83600 | 3  | 5mo |
| 95  | ▼1 | Neel Singh                   |    | 0.83593 | 5  | 5mo |
| 96  | —  | yyqing                       |    | 0.83501 | 13 | 5mo |
| 97  | —  | Wojtek Rosinski              |    | 0.83481 | 2  | 7mo |
| 98  | —  | [ods.ai] Artyom Palvelev     |    | 0.83284 | 8  | 5mo |
| 99  | —  | Rob Wishart                  |    | 0.82952 | 33 | 6mo |
| 100 | ▲1 | Paul H                       |   | 0.82930 | 4  | 5mo |
| 101 | ▼1 | AnTICs                       |  | 0.82930 | 6  | 6mo |
| 102 | —  | BayBreeze                    |  | 0.82920 | 5  | 6mo |
| 103 | —  | Waiting for a blender kernel |  | 0.82856 | 6  | 5mo |
| 104 | —  | RashmiNarvekar               |  | 0.82853 | 8  | 5mo |
| 105 | —  | Timeinbetween                |  | 0.82749 | 18 | 5mo |
| 106 | ▲1 | YCKung                       |  | 0.82692 | 1  | 6mo |

(a) Private leaderboard.

|     |      |                              |   |         |    |     |
|-----|------|------------------------------|---|---------|----|-----|
| 90  | ▲215 | Vadim Borisov                |    | 0.83863 | 7  | 5mo |
| 91  | new  | Gaurav Kumar                 |    | 0.83863 | 2  | 5mo |
| 92  | ▲198 | Breck                        |    | 0.83863 | 6  | 5mo |
| 93  | ▲225 | Akila Wajirasena             |    | 0.83691 | 3  | 5mo |
| 94  | ▲52  | Neel Singh                   |    | 0.83674 | 5  | 5mo |
| 95  | ▲68  | praveen kumar                |    | 0.83654 | 12 | 5mo |
| 96  | ▲43  | yyqing                       |    | 0.83574 | 13 | 5mo |
| 97  | ▲43  | Wojtek Rosinski              |   | 0.83560 | 2  | 7mo |
| 98  | ▲106 | [ods.ai] Artyom Palvelev     |  | 0.83358 | 8  | 5mo |
| 99  | ▲53  | Rob Wishart                  |  | 0.83012 | 33 | 6mo |
| 100 | ▲57  | AnTICs                       |  | 0.82996 | 6  | 6mo |
| 101 | ▲77  | Paul H                       |  | 0.82983 | 4  | 5mo |
| 102 | ▲59  | BayBreeze                    |  | 0.82962 | 5  | 6mo |
| 103 | new  | Waiting for a blender kernel |  | 0.82919 | 6  | 5mo |

(b) Public leaderboard.

Figure 2: Our rankings in the leaderboard.