Team 13: Wild Life Classification with Deep Learning Methods

Li, Zhexu zhl411@ucsd.edu

Ye, Yiheng yiy291@ucsd.edu

Liu, Andy anl043@ucsd.edu

Abstract

The effectiveness of wildlife preservation relies heavily on in-depth understanding of the diversity and behavior of animals, which requires intensive manual labor of experts to correctly label the images collected. In this project, we will try to automate this process by constructing a deep learning model for wildlife classification. Source codes of our project are uploaded to: `https://github.com/YihengYe/ECE228-project`

1 Introduction

Wildlife conservation is an essential part of environment protection, a diverse wildlife provides balance and stability to the ecosystem. Modern wildlife management requires in-depth understanding about the behavior and diversity of different animals in the habitat, usually through sensors and cameras. Traditionally, such insights were produced by human experts who manually label and study those images. But with more cameras installed, manually labeling that large amount of images generated everyday becomes extremely expensive and sometimes infeasible. Therefore, it's important to automate the classification process.

In this project, we aim to build an accurate wildlife classification model to solve that problem and free the experts from this tedious task. We plan to construct a deep neural network for the task, and train it on an animal image data set from kaggle which contains 90 animal species. The resulting model will be helpful for automatic wildlife classification, and it could provide valuable information about characteristics of certain species. The contributions of our project includes following important points:

1. Utilized various data augmentation techniques to deal with certain drawbacks of the dataset, including irregular image sizes, and insufficient training images.
2. Experimented with many current state-of-the-art deep learning models and investigated underfitting or overfitting issues in those models. We discovered the ResNet50 to be the best performing architecture for our animal image data set, and chose it to be the baseline model to improve upon.
3. Significantly improved the vanilla ResNet50 model's architecture by adding a regulation block, while maintaining computationally efficient. The improved model achieved an 7% improvement in validation accuracy, compared to the vanilla ResNet50 architecture.

2 Related Works

In *Insights and Approaches using Deep Learning to Classify Wildlife*[1], Miao et al. tried several state-of-the-art convolution neural networks to classify wildlife from a camera-trap dataset which contains 111,467 images of 20 species. They employed multiple novel modifications in their convolution layer and their best accuracy was 87.5%. One important reason behind their high accuracy is that their

Preprint. Under review.

dataset was very clean and uniform, and every specie has over 5000 training images available, while our dataset has 5400 images for 90 species.

In *Assessing Convolutional Neural Network Animal Classification Models for Practical Applications in Wildlife Conservation*[2], Larson applied ten different CNNs models on seven datasets, simulating 13 scenarios. Her experiments revealed some important aspects people should consider in order to achieve optimal performance when selecting or training a CNN for use in a wildlife camera-trap project.

In *Automatically Identifying, Counting, and Describing Wild Animals in Camera-Trap Images with Deep Learning*[3], Norouzzade et al. utilized CNN to identify and count different species of animals from camera-trap data. They worked with a 3.2 million-image Snapshot Serengeti dataset and achieves an accuracy of 93.8%, but instead of focusing on classifying wild animals, they focused more on simplifying the background environment of the images, and describing the behavior of the wildlife.

Overall, the above works are inspirational for our project. Despite their focus and dataset are often quite different compared to ours, they still provided some valuable ideas which might be helpful for us to develop our own model.

3 Data Set and Data Augmentation

Our data set is an animal image collection which contains 5400 images of 90 different animal species [5]. The data set turned out to be tricky to work with. Each category only has 60 images available, with irregular dimensions, some of them are only 100 * 100 while the others could be 1280 * 720. Originally, we took a naive approach which transforms the data to a regular size, and preprocess the data beforehand to pick the most suitable images. The preprocessed data set turned out to hurt the performance of the model a lot. To achieve a better performance, we utilized with various augmentation techniques, including flipping, shifting, rotation, binarization, and blurring. We didn't use noising because many images in the data set are already quite noisy. The augmented data set contains 10378 images of 88 species (we found two redundant species in the dataset). The augmentation process expanded originally small data set by a lot, and it will be helpful for us to build a more robust classifier.

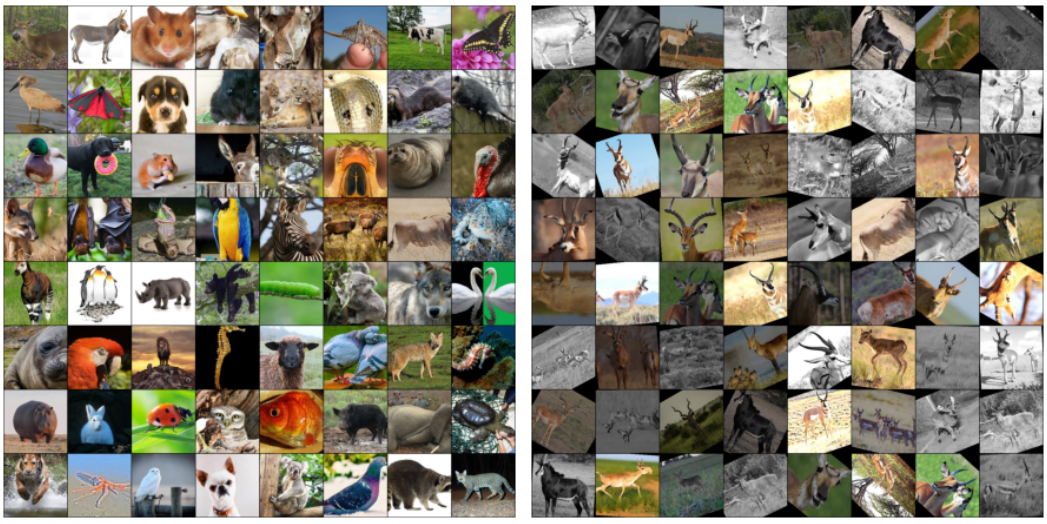

Figure 1: Example grid images for animal image data set before augmentation (left) and after augmentation (right)

4 Problem Formulation

Like mentioned above, the goal of our project is to create a deep learning model for accurate wildlife classification. To be more specific, after we split our dataset into training set (70%) and validation set, the objective is to create a model which can achieve a high classification accuracy on the validation set. Since our project is essentially a classification task, it's natural to use the Cross-Entropy loss on the training process, which is defined by:

$$Loss = -\Sigma_{c=1}^{M} y_{o,c} log(p_{o,c}) \tag{1}$$

Where M is number of classes we need to classify, p is the predicted probability that our prediction o is of class c, and y is the binary indicator for the classification which is defined as:

$$y = \begin{cases} 0, & o \neq c \\ 1, & o = c \end{cases} \tag{2}$$

Furthermore, to evaluate the model's performance, we calculate its' classification accuracy on the validation set, which is defined by:

$$Accuracy = \frac{Number \quad of \quad Correctly \quad Classified \quad Images}{Number \quad of \quad Validation \quad image} \tag{3}$$

5 Method Applied

5.1 Optimizer

To minimize the Cross-Entropy loss mentioned above, we chose to use the Adam optimizer, which for every step it calculates:

$$m_t = \beta_1 m_t + (1 - \beta_1)(\frac{\partial L}{\partial w_t}) \tag{4}$$

$$V_t = \beta_2 V_t + (1 - \beta_2)(\frac{\partial L}{\partial w_t})^2 \tag{5}$$

Which it basically combines momentum and RMSProp (Root Mean Square Propagation) together, where $\beta 1$ and $\beta 2$ are the parameters for momentum and RMSProp, respectively. After that, we have:

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{6}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{7}$$

And finally we can update our weight w at step t as:

$$w_t = w_{t-1} - \gamma \frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \epsilon} \tag{8}$$

Where γ is the learning rate defined for this algorithm.

5.2 Baseline Selection

In order to select an appropriate baseline model that has the potential for later improvement, we have experimented with many state-of-the-art deep learning architectures, including a 9-layer MLP, VGG 16, ResNet 50, and DenseNet 121. The detailed setting and results of these experiments will be discussed in the Experiment Results section later. We also visualized the training error curve and validation accuracy curve of the models to better understand their behavior. And we chose the ResNet50 to be our baseline model to improve upon, because it performed the best among all the architectures we tried. We did notice overfitting in the ResNet50 architecture, though it was not as serious as other complex models. Our objective now is to implement modifications to the ResNet50 architecture in order to achieve better performance.

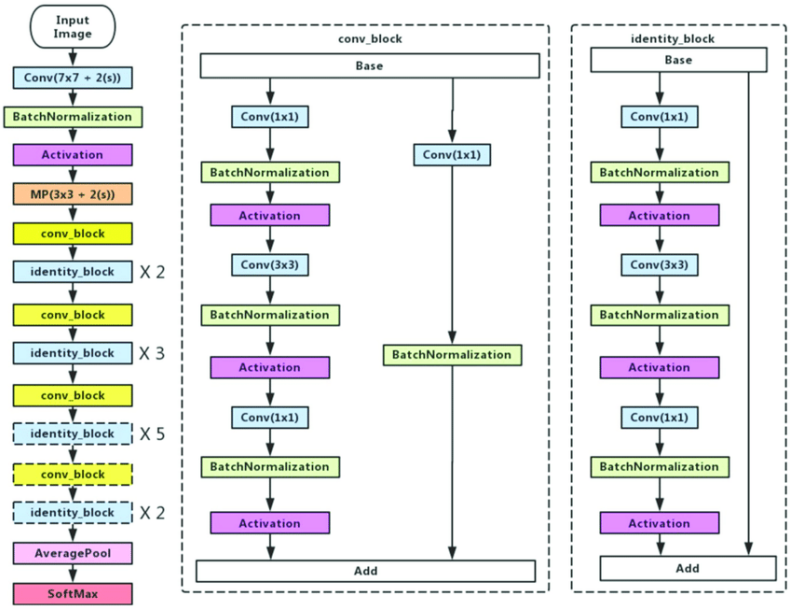

Figure 2: ResNet50 Architecture [6]

5.3 Architecture Improvement

Resnet was introduced by He et al. [7], it introduced a number of improvements over the the VGG architecture. The most important introduction was the Residual Block, which is defined by:

$$y = F(x, \{W_i\}) + x \tag{9}$$

Where F(x) is approximated by the weight (Wi) layers we have in each block. In our model, one such block includes 3 convolutional layers and thus we will use this structure to perform shortcut connection within our neural network. The overview of the ResNet50 architecture [6] is shown in figure 2.

We experimented with various techniques in efforts to optimize the ResNet 50 architecture. Since we observed overfitting during ResNet 50 training, we mainly focused on adding extra regularization terms. The performance of each regularization term will be discussed in the Improvement Analysis section later. In the end, the optimal modification we made was to replace the fully connected layer with our "Regularization Block" which includes two linear layers, a batch normalization layer, and a dropout layer (20%). The structure of our modification is shown in figure 3.

The batch normalization is defined by:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{VAR[x^{(k)}]}} \tag{10}$$

Which acts like a regularization term. The dropout layer regularize the model by randomly drop 20% of the layer outputs. The design decision behind every term is that the first linear layer will wrap up the pretrained weights, the batch normalization and dropout together will provide better regularization, and the linear layer in the end generates classifications. And based on our experiments, our modification improved our model performance significantly with a little extra computational cost, which can be observed in the Experiment Results section.

6 Experiment Results, Analysis, and Discussion

6.1 Comparison with Baselines

To setup the experiments, we used the augmented dataset, and preprocessed the dataset by upsampling all images to 224 x 224, centercrop and normalize. We use 70% images for training, and the remaining

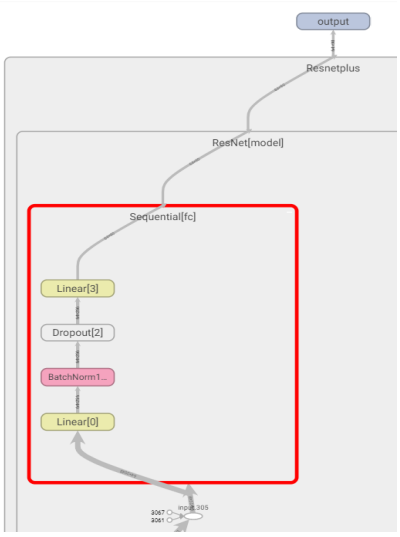

Figure 3: Our Regulation Block

images for validation (which is testing set in our case). And for all models we trained we used Cross-Entropy Loss and Adam Optimizer mentioned above, with the same batch size = 64 (because of GPU memory limit we couldn't try larger batch), learning rate = 0.0001, and all with weights pretrained on ImageNet except the naive 9 layers MLP.

Our experiments shown that the naive MLP model was underfitting a lot, achieved only 29% validation accuracy. While for the other more complex models, the VGG 16 was heavily overfitting with decreasing validation accuracy towards the end of training, achieved 67% validation accuracy. The more recent and more complex DenseNet 121 also struggled a lot with overfitting, but in the end it achieved a validation accuracy of 80% which is higher than the VGG 16. The ResNet 50 on the other hand, while also overfitted, achieved the highest validation accuracy among all the baseline models: 85%. Since it's already overfitted, we decided not to try any other more complex network, and sticked with the ResNet50.

After we implemented our own architecture improved on the ResNet50 with our custom Regularization block added (mentioned in section 5.3), we trained our model using the same setting, and achieved a validation accuracy of 92% which is about 7% higher than the vanilla ResNet50 model, 12% higher than the DenseNet 121, and 25% higher than the VGG 16. The training error curves and validation accuracy curves are shown in figure 4, the naive MLP was not included because it was not pretrained so not meaningful for comparison. In the Training Error vs. Epoch plot on the left we can see our improved architecture converged significantly faster than other models, and the Validation Accuracy vs. Epoch on the right shows our model constantly outperformed all other models by a considerable amount.

Overall, our improved model has brought an impressive 8% improvement over the vanilla ResNet 50 architecture, with little extra computational cost (1 extra linear layer in additional the one replaced the original fully connected layer, 1 batchnorm layer and 1 dropout layer). And we believe our improved model has striked a good balance between complexity and simplicity, while the MLP and VGG was a bit too simple, requiring way more data, and the vanilla ResNet50 and DenseNet was too complex, resulting in serious overfitting.

6.2 Improvement Analysis

In this section, based on the feedback provided by the professor, we are going to investigate which part of our Regularization Block contributed the most improvement. The creation of this specific Regularization Block involves a lot of trials and errors, we've spent a large amount of training time in efforts to find the optimal modification to the network. Among the regularization terms we tried, BatchNorm layer and Dropout layer performed the best. Since both two methods brought improvements to the network, we decided to combining them together, which resulted in the

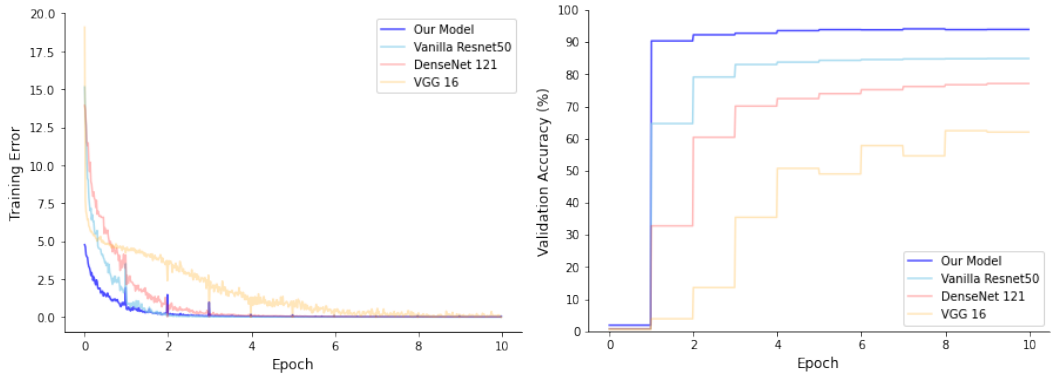

Figure 4: Plots of Training Error (Left) and Validation Accuracy (right) for Model Comparison.

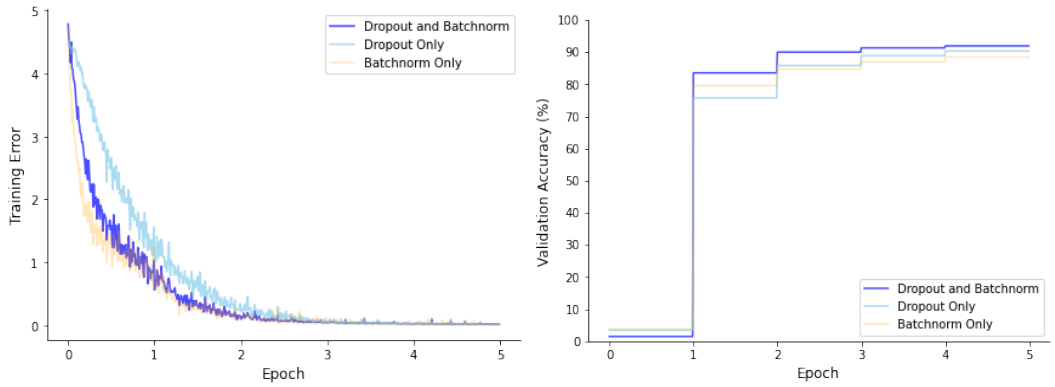

Figure 5: Plots of Training Error (Left) and Validation Accuracy (Right) for Improvement Analysis.

Regularization Block we introduced above. Again, the intuition behind our custom Regularization Block is simple: it's basically a fully connected layer to wrap up the pretrained weights, a BatchNorm layer and a Dropout layer for better regularization, and a fully connected layer to generate the classifications. We compared the performance of those 3 methods (Dropout only, BatchNorm only, Dropout and Batchnorm). The Dropout only model achieved a validation accuracy of 90%, the BatchNorm only model also had a validation accuracy of 89%. The Dropout and BatchNorm combined model achieved an validation accuracy of 92% which is higher than the other two. The resulting training and validation curves are shown in figure 5, and we can see the Dropout and BatchNorm combined model converged the fastest and was constantly outperform the other two. Therefore, we believed our custom Regularization Block performed well for our tasks.

6.3 Results Discussion

Based on the results and analysis discussed above, our improved model outperformed all the models we have experimented by a considerable amount. And our improvement analysis also shown that, while only adding either Dropout or Batch Normalization to the ResNet50 could improve the performance, combining both methods together gives faster convergence and way better improvements. On the other hand, the reasons why the other models (MLP, VGG 16, Resnet 50, DenseNet 121) were not working as well can be explained in two aspects: they are either too simple or too complex. Simple structures like VGG16 and MLP were struggling to get a good presentation of the dataset, and they need way more data in order to capture important features, and even our augmented data set cannot meet their demand. The DenseNet121 and Vanilla ResNet50, however, were too complex, and thus resulted in serious overfitting, making them hard to generalize well on the validation data set. Therefore, we believed adding our custom "regulation block" to the complex model could help a lot for improving generalizability and overall performance. And based on our experiments, our model indeed met our expectation, reaching an accuracy of 93%, 7% higher than

the vanilla Resnet50. And we believe our model has striked a good balance between complexity and simplicity.

7 Conclusion and Future Works

In this project, we experimented with various data augmentation techniques to improve the quality of our animal dataset which is small and contains images of irregular dimensions. We also tried a number of state-of-the-art deep learning architectures, including MLP, VGG, ResNet, and DenseNet, and experimented with different regulation terms to reduce overfitting. After a lot of trials and errors, we discovered the optimal architecture for our task, which was built upon the ResNet50 structure with additional regulation blocks. Our improved model achieved a validation accuracy of 93% after being trained for 10 epochs, which brought considerable improvements over other models, with a little extra computational costs. In the future, we can experiment with more augmentation techniques and regulation terms, and further expand the dataset, to improve the capability of our model.

8 Individual Contribution

Yiheng Ye wrote parts of the proposal, milestone, poster, and final report. He generated model structures diagrams for this report. He built the project repo and created the MLP model. He also set up basic hyperparameter settings,training-validation splits, and notebook structures. Furthermore, he also wrote parts of Readme write up. He participated in the poster session and introduced the experiment results at the session.

Zhexu Li wrote codes for running the different baseline models and conducted experiment to find the best way to improve the validation accuracy, which resulted in the regulation block introduced. He helped with constructing the augmented dataset, as well. He also wrote parts of the proposal, milestone, poster, and final report. He participated in poster session and delivered the majority of the presentation. He has been involving discussions of the project.

Andy Liu wrote parts of the proposal, milestone, poster, and final report. He helped build the augmented data set. He participated in poster session and answered questions from teachers and guests. He also printed and set up the poster.

Reference

[1] Miao, Z., Gaynor, K.M., Wang, J. et al. Insights and approaches using deep learning to classify wildlife. Sci Rep 9, 8137 (2019). `https://doi.org/10.1038/s41598-019-44565-w`

[2] Larson, Julia, "Assessing Convolutional Neural Network Animal Classification Models for Practical Applications in Wildlife Conservation" (2021). Master's Theses. 5184. DOI: https://doi.org/10.31979/etd.ysr5-th9v `https://scholarworks.sjsu.edu/etd_theses/5184`

[3] Norouzzadeh, M.,Nguyen A., Kosmala, M. et al. "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning". DOI: `https://doi.org/10.1073/pnas.1719367115`

[4] Karen Simonyan, Andrew Zisserman "Very Deep Convolutional Networks for Large-Scale Image Recognition" *arXiv preprint arXiv:1409.1556*

[5] Animal Image Dataset (90 Different Animals) `https://www.kaggle.com/datasets/iamsouravbanerjee/animal-image-dataset-90-different-animals`

[6] Ji, Qingge & Huang, Jie & He, Wenjie & Sun, Yankui. (2019). Optimized Deep Convolutional Neural Networks for Identification of Macular Diseases from Optical Coherence Tomography Images. Algorithms. $12.51.10.3390/a12030051$

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition arXiv:1512.03385