Deep Convolutional Network for Bird Classification

Zihao Kong
zikong@ucsd.edu

Baichuan Wu
bwu@ucsd.edu

Abstract

We propose a *convolutional neural network* (CNN) approach to the bird classification task for this programming assignment. Commonly applied in analyzing visual imagery, convolutional neural networks indulges uncontested superiority comparing to their traditional deep neural network counterparts, which, unfortunately, aren't scalable to various input sizes. In this programming assignment, besides exploring the mathematical and structural backbones of the convolutional neural network, we handcrafted a convolutional network that has the capability of distinguishing various categories of bird images from an exceptionally small size of training dataset, using the PyTorch framework. Our custom model achieved 42.55% accuracy on the testing dataset.

1 Introduction

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.[3] They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

At the outset, we obtained a well-organized dataset of real-world images from the Caltech-UCSD birds dataset. [4] This dataset has images of 200 categories of North American birds. The dataset has around 12,000 images out of which 6000 are training images and the rest are test images. We then applied a mean cancellation and z-scoring covariance equalization to the dataset. We firstly practiced the training procedure on a baseline CNN model, which employs a 4 convolutional and 2 pooling layer structure, with cross entropy loss function. We report an accuracy of 38.52% on the baseline model. We then switched to a custom model, leveraging valuable ideas from the writeup, for performance improvements. In this modified model, we achieved 42.55% accuracy. Lastly, we performed a transfer learning, using both vgg-16 and resnet 18 models, we achieved 79.39% accuracy using this method.

2 Related Work

Work by Lin et. al. [1] proposed a bilinear model, which consists of two feature extractors (CNNs) whose outputs are multiplied using outer product to obtain a vectorized image descriptor. Thier

architechture tackles the pairwise feature interaction in a translationally invariant manner, which is particularly useful for fine-grained categorizetion. Their implementation shed some light to our approach on revising the baseline model. Besides, another work by Luo et. al. proposed a filter level pruning method for deep convolutional network compression. [2]

3 Models

3.1 Baseline Model Architecture

The baseline model is simply constructed out of three convolutional layers, trailed by a maximum pooling layer, then fed into another convolutional layer and average pooling layer before heading to the fully connected layer.

3.2 Custom Model Architecture

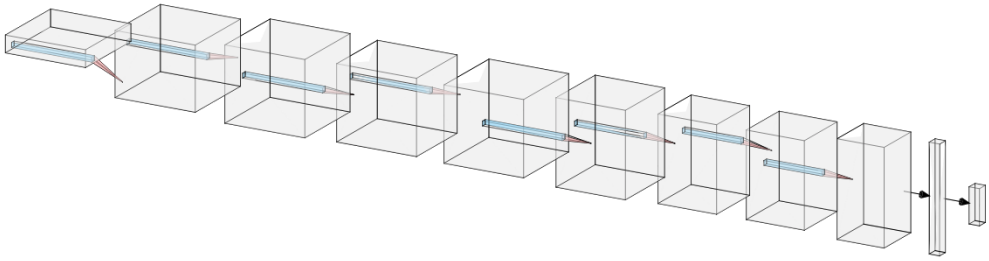


Figure 1: Visualization of the Custom Model

For the custom model, we proposes a 6-layers architecture, with six convolutional layers and two maximum pooling layers in between, activated by ReLU activation function. The model is fine tuned with the best hyper-parameters before evaluating test accuracy, using the holdout set. We propose this architecture in lieu of the baseline model to tackle its unavoidably annoying overfitting issue, due to limited availability of training data. Using this approach, we achieved 42.55% test accuracy - a gain around 4.03% testing accuracy. Before arriving at this model, we trapped ourselves with another proposal that aimed to imitate AlexNet. Nevertheless this attempt failed with a staggering 35.83% accuracy. We speculate this architecture that we imitated didn't effectively overshot the overfitting problem, since it also demonstrates a huge disagreement in testing and validation accuracy.

Table 1: Layer Architecture of the Custom Model

Layer	C_{in}	C_{out}	Kernel	Stride
conv1	3	64	3×3	1
conv2	64	128	3×3	1
conv3	128	128	3×3	1
maxpool	128	128	3×3	1
conv4	128	256	3×3	1
conv5	256	256	3×3	1
maxpool	256	256	3×3	1
conv6	256	512	3×3	1

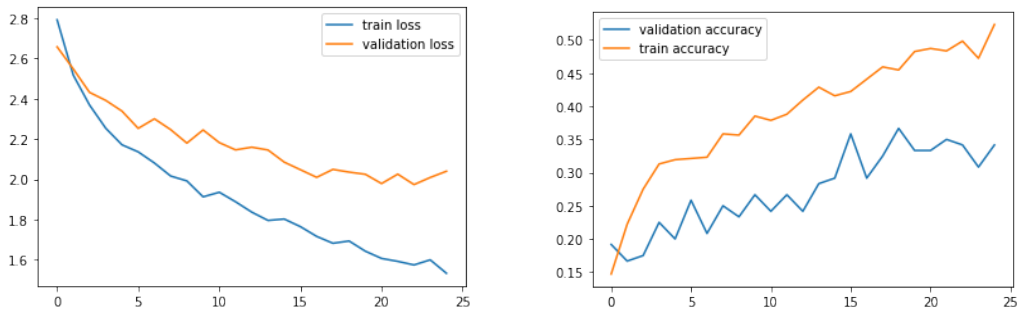
3.3 VGG-16 and ResNet18

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet. It makes the improvement over AlexNet by replacing large kernel-sized filters with multiple 3×3 kernel sized filters one after another.

ResNet18 is a residual learning framework that eases the training of networks that are substantially deeper than those used previously.

4 Experiments

4.1 Baseline Model



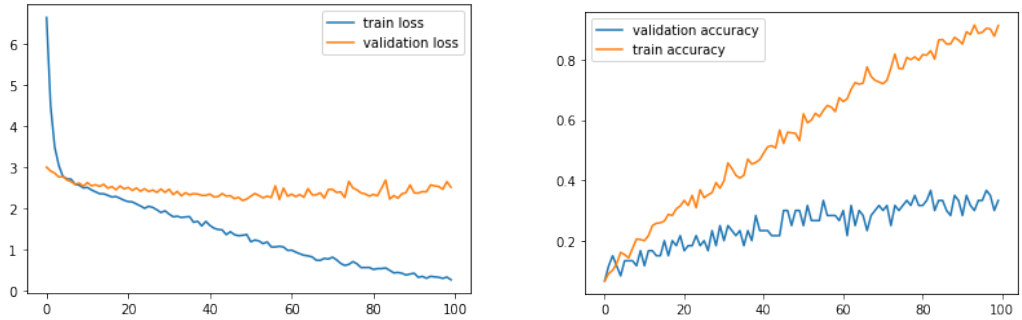
(a) Loss (b) Accuracy

Figure 2: Baseline Model Training and Validation Performance; $\alpha = 10^{-4}$, $b = 20$

The baseline model is trained using cross-entropy loss function and Xavier Initialization for 25 epochs. We have also used an Adam optimizer with learning rate 10^{-4}. Two validation sets are used to capture best weights over the 25 epochs.

4.2 Custom Model

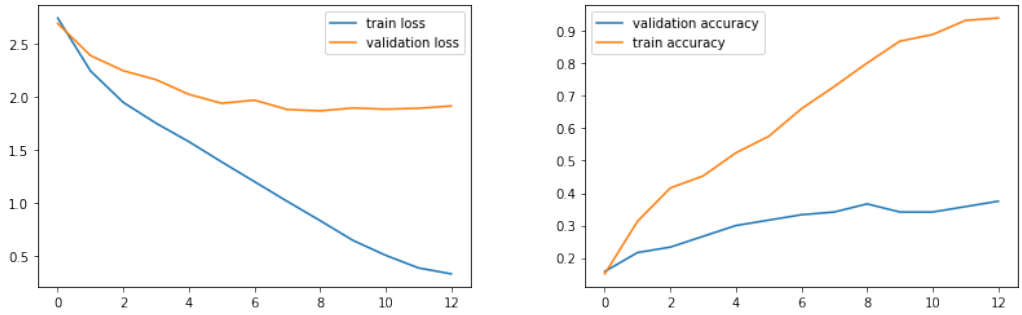
We have taken two approaches on the custom model, we begin showing the results from our failed attempt:



(a) Loss (b) Accuracy

Figure 3: Failed Custom Model Training and Validation Performance; $\alpha = 10^{-4}$, $b = 30$

These figures demonstrated the heavy overfitting nature of this ill-fated custom model, despite our attempts to alleviate the overfitting by using strong L2 regularization. We eventually decided to abandon this approach and move on with another approach, with the following results:



(a) Loss (b) Accuracy

Figure 4: Resultant Custom Model Training and Validation Performance; $\alpha = 10^{-4}$, $b = 15$

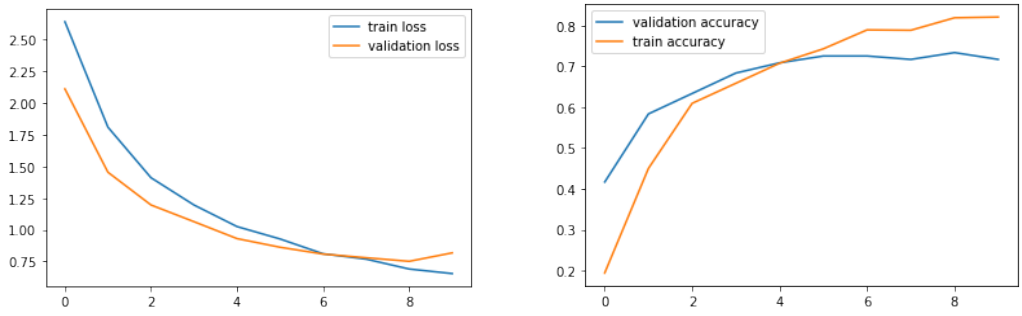
Using this approach, we carried out a modest 42.55% test accuracy.

Table 2: Performance Comparison between Two Distinct Custom Model Approaches

	Accuracy	
Architecture	Training	Testing
Failed Custom	91.30%	35.83%
Resultant Custom	95.27%	42.55%

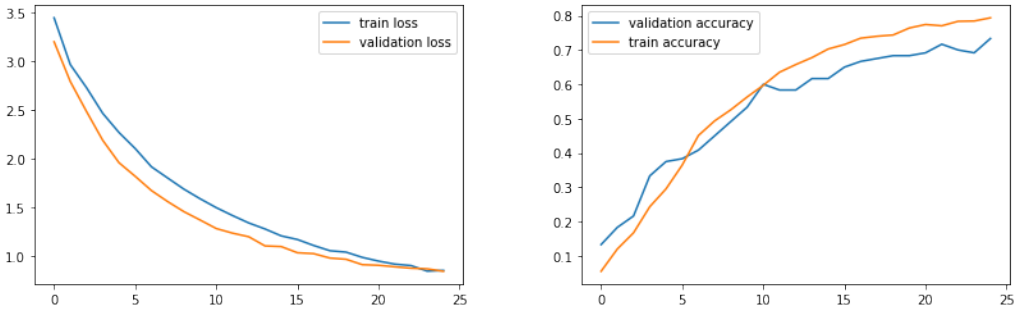
Due to the staggering difference between training and testing accuracy, we conclude that our resultant model still suffers from overfitting, which is an outstanding problem yet to be solved.

4.3 VGG-16 and ResNet Transfer Learning



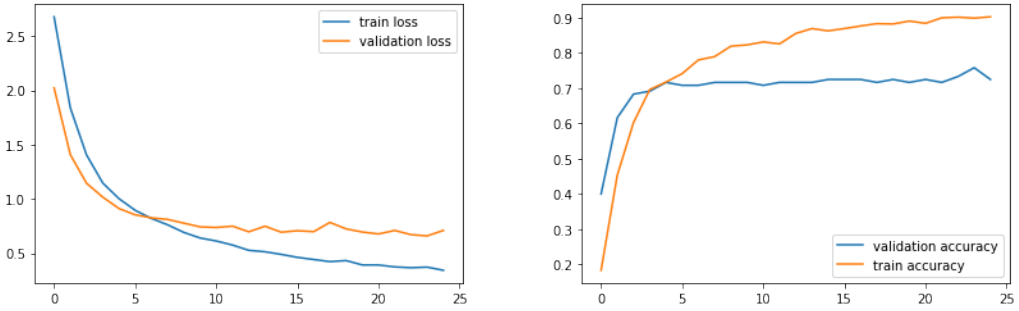
(a) Loss (b) Accuracy

Figure 5: VGG-16 Training and Validation Loss, Frozen



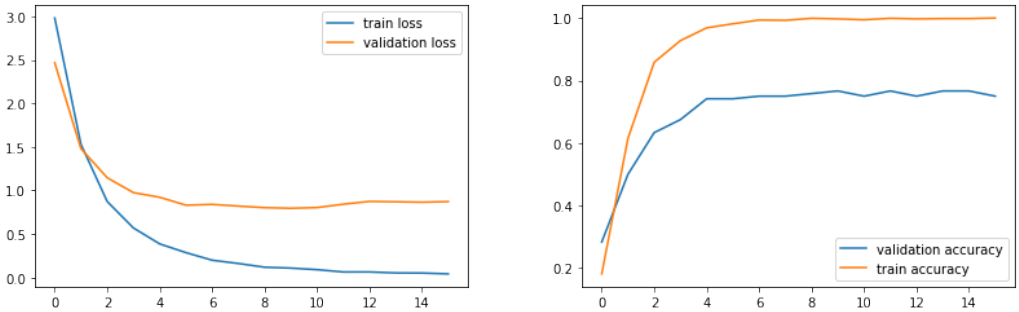
(a) Loss (b) Accuracy

Figure 6: ResNet18 Training and Validation Loss, Frozen



(a) Loss (b) Accuracy

Figure 7: VGG16 Training and Validation Loss



(a) Loss (b) Accuracy

Figure 8: ResNet18 Training and Validation Loss

5 Feature Map and Weights Analysis

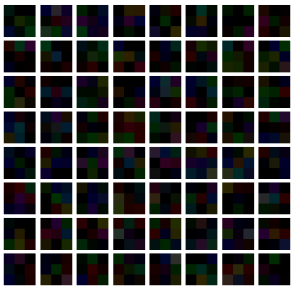


Figure 9: Weight Map of the First Convolutional Layer, Best Custom Model

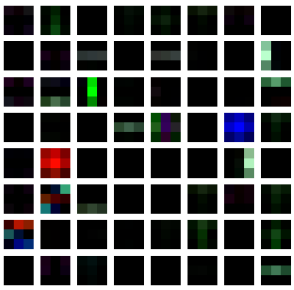


Figure 10: Weight Map of the First Convolutional Layer, VGG-16

We can observe distinctive R, G, and B patches visually, resembling particular learned pattern.

6 Discussion

We summarize all aforementioned discussions with a comparison between all methods and models used:

Table 3: Performance Comparison across All Models

	Accuracy	
Architecture	Training	Testing
Baseline	55.65%	38.52%
Failed Custom	91.30%	35.83%
Resultant Custom	95.27%	42.55%
VGG-16 (Frozen)	83.45%	72.17%
ResNet18 (Frozen)	79.81%	75.54%
VGG-16	99.49%	79.39%
ResNet18	99.85%	77.71%

As shown in the table, VGG-16 and ResNet18 undoubtedly outperforms the baseline and our custom model. Not only in terms of final testing accuracy, but also a smaller deviation between training and testing accuracy, suggesting a significant mitigation of the persistent overfitting issue. We believe that we shall improve our custom model in terms of modifying layer architectures to better conform which used by VGG-16 and ResNet, that is - deeper. Over the course of hyper-parameter tuning, we tuned the learning rate and weight decay to smoothen the learning curve, also, we have strengthened L2 regularization to mitigate overfitting in our custom model. Nevertheless, the change of architecture most significantly benefits our custom model in terms of accuracy.

7 Team Contributions

We hereby conclude our report and state contributions from each of our team members (names are listed alphabetically by last name, **equal contributions**):

- Zihao Kong: responsible for the implementation of classes and methods, reviewed and debugged the entire code repository.
- Baichuan Wu: responsible for the implementation of classes and methods, reviewed and refactored code. Wrote the report. Organized code and produced figures for submission.

References

References

[1] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. “Bilinear CNN Models for Fine-Grained Visual Recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[2] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. “ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[3] M.V. Valueva et al. “Application of the residue number system to reduce hardware costs of the convolutional neural network implementation”. In: *Mathematics and Computers in Simulation* 177 (2020), pp. 232–243. ISSN: 0378-4754. DOI: https://doi.org/10.1016/j.matcom.2020.04.031. URL: http://www.sciencedirect.com/science/article/pii/S0378475420301580.

[4] C. Wah et al. *The Caltech-UCSD Birds-200-2011 Dataset*. Tech. rep. CNS-TR-2011-001. California Institute of Technology, 2011.