Reclassifying Eagles: Enhancing Image Accuracy with Convolutional Neural Networks

[Parikh Samuolis Reclassification Github]

Riya Parikh, Tomas Samuolis

Faculty of Computing and Data Sciences

DS 340: Intro to Machine Learning/AI

Kevin Gold

December 9, 2024

Introduction

In this project, we aimed to address the challenge of building a convolutional neural network (CNN) architecture capable of identifying and being robust against mislabeled training data. Specifically, our objective was to evaluate how varying levels of image augmentation impact a model's performance in the presence of mislabeled data. Furthermore, we sought to associate a confidence level with each label, enabling the reclassification of the labels with the lowest confidence.

This problem was inspired by a real-world scenario encountered by one of our team members, Tomas, during his internship. In that setting, the task involved classifying insurance claims to determine whether they should receive a particular service. While the positive class correctly included claims that received the service, the negative class contained claims that had not received the service—some of which were mislabeled because they should have received it. This led to a modeling challenge, as the extent of mislabeling was unknown, and mislabeled claims closely resembled the positive class. Although our project focuses on images rather than tabular data, it simulates a similar problem: dealing with mislabeled data and exploring methods to mitigate its impact. By examining the interplay between image augmentation and mislabeling, we aim to contribute to the broader understanding of how CNNs can handle imperfect data and improve classification reliability.

Methodology

As we were primarily focused on creating a model to identify and reclassify incorrect labels, the actual data category itself was not very important. Our final intention was that the model could be used with any image data. We looked through various datasets, and after looking through images used in the Imagenet competition, we decided to take on a bird theme, picking four categories of birds: American Eagles, Chickadee, Black Grouse, and Goldfinch, each of which there were 1300 images for training and 50 images for validation. To better mimic the issue experienced by Tomas while keeping our image set diverse, instead of predicting four distinct categories we chose eagles to be the target class, and relabeled the remaining categories as "non-eagles" leaving us with a base split of 25% target category, and 75% another category.

Furthermore, we chose the VGG16 architecture as a base to classify our images. VGG16 is a convolutional neural network (CNN) model used for image classification with 13 convolutional layers, 3 fully connected layers, 5 pooling layers, and 3×3 convolution kernels. It uses a ReLU activation function. We knew that VGG16 was already a strong, high-performing model when the labels were correct, so we knew that reductions in performance would stem from the mislabeled data rather than flaws in the model itself. To prepare our data to fit within the VGG framework, we scaled it to (224, 224, 3), which included ensuring proper color channels.

After loading and preparing our data, we first established a baseline. We experimented with using slightly different variations of VGG16 excluding the top (fully connected) layers—including presetting with Imagenet base weights and freezing/unfreezing the trainability of the base layers— on data with no labels misclassified. Our best result came from the VGG16

model with pre-trained ImageNet weights that excluded the top layers and froze internal layers. The best result came on epoch 6/15 which resulted in the best validation accuracy of 0.9371. By doing this, we managed to understand how a model with perfect labels would score, giving us context for the model built with misclassified data.

First, to begin our analysis with misclassified eagle images, we randomly misclassified 20% of all the training eagle images in memory. The rate of 20% was picked because there was enough misclassification to trick the model into predicting naively: it outputted 75% accuracy on validation data, with all eagles classified as non-eagles. Picking anything higher wouldn't lead to any worse results, so it didn't make sense to focus on that.

Second, we explored how different levels of data augmentation affected the model, and whether data augmentation could make the model more immune to the mislabeled data without compromising the performance of the model without misclassification. Three levels of data augmentation were tested: simple, moderate, and advanced. Simple consisted of randomly flipping images horizontally, rotating images up to 15 degrees, and zooming by a factor of 0.1. Moderate consisted of flipping images horizontally, rotating images up to 20 degrees, shifting images horizontally and vertically by 0.2, adjusting brightness between 80% and 120%, and zooming by a factor of 0.1. Lastly, advanced consisted of flipping images horizontally, rotating images up to 30 degrees, shifting images horizontally and vertically by 0.3, adjusting brightness between 50% and 150%, zooming by a factor of 0.3, applying shearing transformations of 0.2, and randomly shifting color channels. Ultimately, we found that moderate augmentation provided the best improvements in score, leading to the model surpassing naive classification with misclassified data.

Third, we wanted to explore whether we could use the CNN model to identify its own misclassified images by analyzing the decimal outputs of the model's predictions on its training data. Essentially, we wanted to interpret the outputs of the model as confidence scores for the labels—the closer the model's output is to 1, the more "confident" the model is that the image it sees is an eagle. We wanted to use this fact to try to identify the photos labeled as "non-eagles" with the assumption that the model would tend to have a lower output for photos that are not eagles (output closer to 0) and higher output for images that are eagles but had the "non-eagle" label. However, since the baseline model with 20% misclassified labels just predicted all images as non-eagles (0 class, prediction output <0.5), the model's direct prediction output didn't provide much information. The images with the higher model outputs were just images we already knew were eagles from the beginning.

To address this issue, we added a second channel to our CNN: a column of label-confidences as secondary input beyond just using images. This column was iteratively updated by retraining the model with updated confidence. We initialized our confidences as 1 for all images labeled as eagles, and 0.35 for all images labeled as non-eagles. Eagles were assigned a confidence of 1 because we were fully confident in the eagles with the eagle label. We did not change any of the non-eagle labels to eagles, thus all eagles in the set must be set to true[1]. For

[1] For generalizability, this assumption that all the labels within the target class are correct must hold.

non-eagle images, we chose a threshold of 0.35 because it provided a balance between minimizing false positives and ensuring the model remains cautious in its predictions. This value reflects the observation that most non-eagle images had confidence scores significantly below 0.5, and lowering the threshold helped capture borderline cases more effectively while maintaining a consistent level of skepticism for all labels. As a reminder, while in our code we kept track of which indices got flipped, in the "real world" all one would know is that some of the images within the non-target class were incorrectly labeled, but be equally unsure of them.

Initially, we were hoping that after introducing the confidence input, the predictions would be above 0.5 for the eagles labeled as eagles, very close to 0 for the non-eagles, and something in-between for eagles labeled as non-eagles. The plan was then to update the confidence with the outputs of that model and run until convergence. Nonetheless, we were unable to go down this route as the confidence input was too powerful and resulted in all predictions at a number very close to 0 or very close to 1 (ie. 0.9997 or 0.003). Due to this, updating the confidence to those outputs wouldn't represent our confidence very accurately.

However, when we performed a t-test on the output for true non-eagles compared to mislabeled eagles, we found that there was still a remarkably statistically significant difference between the predictions. The true non-eagles had a lower average prediction than the mislabeled eagles with a p-value in the realm of 1E-60, indicating that our initial hypothesis that the model would be less confident about the mislabeled eagles had merit.

Thus, our alternative approach was to iteratively update our confidence by changing the 5% of images whose predictions were closest to zero from 0.35 to 0. The rationalization for this was that we expected the images having the predictions closest to 0 to disproportionately be true non-eagles. Updating the confidence would then teach the model to better differentiate between true non-eagles and false non-eagles.

To ensure our flipping model used the best possible hyperparameters, we ran a grid search that experimented with various hyperparameters such as the percentage of least confident eagle predictions to adjust (n_percentage with testing parameters of [5, 10, 15, 20]), the number of epochs and number of iterations (max_iterations with testing parameters of [5, 10, 15]), and what percentage of predictions to ultimately flip based off of our earlier steps (n_percent with testing parameters of [5, 10, 15, 20]). At the end of testing, we found that 10 iterations of training a single epoch and changing 5% of the predictions closest to zero (from the images with confidence 0.35) provided the best results at separating the eagles from the non-eagles.

After this, we hoped to find a threshold prediction above which we could change all the labels from 0 to 1, but that wasn't feasible. Even flipping the top 5% of predictions below 0.5 would end up reclassifying a significant number of non-eagles as eagles—which would defeat the purpose and reduce the precision. Instead, we decided to take a more practical approach. We saw that within the highest 5% of predictions below 0.5, there were around 200 total images of which around 60 were eagles. Within the highest 10%, around 110 out of 400 were eagles. Meaning, that we were able to filter out almost half of the mislabeled eagles out of a sample size that was 10 times smaller than the original. A human being could now look through only 400

images rather than 4000, and identify a significant proportion of the mislabeled images. While this is not as convenient as just being able to automatically flip the label within the model, we think it is a significant improvement over the baseline.

Therefore, we took the top 10% of predictions and changed the label of the eagles within them to 1. In an out-of-lab scenario, this would be done by a human looking at the images. But, in our case, since we tracked which indices were wrong, we flipped those automatically. Then, we took our data with our updated labels, and ran it through the original best single channel model (VGG16,pre-trained ImageNet weights, excluding top layers, and freezing internal layers) with moderate augmentation (previously found to be the best), as our final model.

Results

Our final model ended up with an average accuracy of 94.8% (validation loss: 0.1513). This was 1.5 percentage points better than the model with no image augmentation and 1.3 percentage points better than the moderate augmentation with 20% misclassification. In addition to achieving an average accuracy of 94.8%, the model demonstrated strong class-specific and overall performance across precision, recall, and F1 scores. Precision, which measures the proportion of correct positive predictions out of all positive predictions, was 97% for the "Not Eagle" class and 85% for the "Eagle" class. This indicates the model was highly reliable in its "Not Eagle" predictions, though slightly less so for "Eagle." Recall, which measures the proportion of actual positives correctly identified, was 95% for "Not Eagle" and 92% for "Eagle," reflecting the model's effectiveness in identifying relevant instances across both classes. The F1-score, the harmonic mean of precision and recall that provides a single measure of balance between the two, was 96% for "Not Eagle" and 88% for "Eagle." These metrics were chosen because they offer a more nuanced understanding of performance beyond accuracy, particularly for imbalanced datasets or cases with mislabeled data. Macro averages (precision: 91%, recall: 93%, F1-score: 92%) and weighted averages (precision, recall, and F1-score: 94%) further demonstrate the model's robustness and overall balance, even in the presence of class imbalance and mislabeled data. These results, which can be seen in the figure to the right, validate our use of moderate image augmentation and highlight the model's capability to handle noisy datasets effectively.

```
Confusion Matrix:
 [[142   8]
 [  4  46]]

Classification Report:
              precision    recall  f1-score   support

   Not Eagle       0.97      0.95      0.96       150
       Eagle       0.85      0.92      0.88        50

    accuracy                           0.94       200
   macro avg       0.91      0.93      0.92       200
weighted avg       0.94      0.94      0.94       200
```

Conclusions

One thing that was interesting to learn over the course of this project was how to implement multi-input neural networks. Figuring it out was a little challenging though.

Resources online were a little confusing to interpret, and we had some formatting issues that led to bugs. Once we figured it out though, it was satisfying to get the model to work.

Given the inherent difficulty of this problem, we think that this was quite a successful first attempt to understand and tackle it. If we had successfully found a way to correctly change all of the labels, that would probably be pretty groundbreaking. While we were not able to fully automate the process of improving the model, if this was a real-world scenario where one wanted to improve one's labels without looking through the entirety of a dataset, this model would be effective in assisting one with that. Meaning, we were able to effectively provide practical business applications by producing software that can filter data labels. While data labeling is quite expensive, we can reduce the costs.

That being said, the use case of the model in its current form is slightly limited for a few reasons. Firstly, it doesn't provide a huge marginal improvement from the baseline 20% misclassification. A good use case for the model, still, would be if one was looking to squeeze out a few extra percentage points from a labeling model's accuracy. But, if absolute best performance is the goal, assuming that one has all the resources and the dataset is not extremely large, it may be simpler to manually reclassify images to ensure their accuracy.

Moreover, if a given dataset differs significantly from the one used in our analysis, different tuning may have to occur to get similar or better results. For example, 25% of our data is our target class, and we've misclassified 20%. In the real world, without an estimate for how much of a dataset is misclassified within each class subset, much more extensive testing and hyperparameter tuning would be required.

Overall, we believe that there is more value in looking at this project and model as an initial exploration and experimentation with misclassified data, rather than the production of a fully functional model. This experiment showed promise in that the model was able to filter and separate incorrectly labeled data to an extent, but more advanced methods are likely needed for more improvement. One possible improvement to our current system may be a more advanced way to update the confidence scores between model training. Given additional time and resources, we would love to further explore the nuances of this model to improve its quality.