Binary Detection of Avian Attributes: Leveraging Convolutional Neural Networks to Identify Hundreds of Attributes from Images

Cole Robert Stokes

Department of Computer Science & Department of Statistics and Data Science

The University of Texas at Austin

AI 395T: Case Studies in Machine Learning

Dr. Junfeng Jiao

December 2, 2024

Abstract

This study explores large-scale, multi-label, binary attribute classification using the CUB-200-2011 dataset. This dataset is widely used in computer vision with over 11,000 images of birds from 200 species. The main challenges of this study include limited access to processing hardware resulting in constrained model complexity and non-spatial attribute data. A custom-built convolution neural network is used to detect which attributes are present in each photo. The model is optimized using a weighted binary cross-entropy loss. Two different weights are applied to the positive class, one favors recall and the other precision. The analysis breaks down the performance of the model using each weight and dives deeper into attribute class imbalances and attribute category performance. Using the F1 score as the main performance statistic, this simple model can achieve between 0.48 and 0.49 for all attributes and over 0.8 for the top attribute subclass. The project is concluded with suggestions for improving the large-scale, multi-label binary attribute classification.

Table of Contents

1 Introduction

In this case study, large-scale binary classification of attributes will be explored by using the CUB-200-2011 Dataset. This dataset is comprised of thousands of images containing birds. For each image, there is a list of attributes that are annotated as present or not present. A couple examples of these attributes are “red-colored nape” and “spatulate-shaped bill” (Wah et al., 2011). Each image may only contain a small subset of the hundreds of possible attributes. This leads to imbalanced classes and complexities during classification. This class imbalance in classification is a well-known challenge in machine learning. Some proposed techniques to address class imbalance issues are re-sampling and class-weighting methods (He & Garcia, 2009; Cui et al., 2019). We will explore class-weighting in this study.

One of the main challenges faced in this study is the limited access to graphical processing units (GPUs). This limits how deep the model’s construct can be, which limits the ability to capture complex fine-grained details in the images. Another challenge is the nature of the attribute data. CUB-200-2011 only provides binary information on whether a specific attribute is present in the image, with no spatial aspect. Without spatial context, the detection of attributes becomes a much more difficult task because the model must rely on the entire image. Given these challenges and limitations, we will aim to optimize large-scale attribute detection without complex models and enhanced hardware capabilities.

1.1 The CUB-200-2011 Dataset

The Cub-200-2011 dataset was organized by researchers from the California Institute of Technology (Caltech) and the University of California, San Diego (UCSD). It contains 11,788 images of 200 bird species. Images were collected from Flickr and have a wide range of dimensions. Each image is annotated with a species label, a bounding box, the locations of 15 anatomical parts, and the presence or absence of 312 attributes. The part and attribute annotations were crowdsourced using Amazon Mechanical Turk (MTurk), a platform where workers can make money by completing tasks like labeling data for machine learning projects (Amazon Mechanical Turk, 2024).

There are multiple attribute categories within the attribute list. For example, the category “has belly color” has 15 sub-attributes within it. A species of bird can have more than one

attribute within each category. For example, the common yellowthroat can be annotated to have crown colors of grey, white, and buff. Because more than one attribute can be present in each category, we will stick to binary attribute classification and not multi-classification by category.

For this study, only the images and the annotated binary attributes will be used. In the initial paper introducing the CUB-200-2011 dataset, Wah et al. (2011) explored species classification, species detection, and part localization; however, attribute classification was not touched on.

1.2 Significance of Binary Attribute Classification

Attribute detection is an important aspect of computer vision that enables the fine-grained detection of images. In the case of this study, instead of trying to classify a bird, we are trying to pick out features that describe it, such as the coloration of the wings and the shape of the wings. This small-scale detection of attributes has been useful in many domains beyond this such as facial recognition and medical diagnostics (Huang et al., 2020; Zhou et al., 2021). In the case of avian classification, small-scale detection can significantly impact the ability of a model to differentiate between very similar species and help us understand avian biological features better.

Many additional challenges are faced when detecting attributes in images. The quality of timing of images significantly impacts this as attributes become hard to detect with poor image quality and the broad range of poses the bird may be captured in. A model must be able to generalize the training data enough to perform well. In addition to the binary class imbalance, there are imbalances in the data of how often certain attributes appear. For example, a bird having a yellow beak is much more likely than one having an iridescent body color. This may be accounted for through advanced weighting techniques and data augmentation (Lin et al., 2017). A multi-head deep learning model that predicts correlated features can improve accuracy (Zhang et al., 2016). This may be a useful strategy in classifying both bird species and their attributes.

Though there are many challenges to overcome with binary attribute classification, it is worthwhile to find solutions to greatly enhance computer vision capabilities.

2 Previous Work

The CUB-200-2011 dataset has been widely used for fine-grained image classification such as species classification, species detection, parts localization, and binary attribute classification.

Since multi-label, binary, attribute classification is the main objective of this study, we will explore previous works in the areas of attribute classification and multi-label classification.

2.1 Attribute Classification

The focus of attribute classification is to detect specific features in an image by learning fine-grained patterns. In CUB-200-2011, there are 312 attributes such as “red wing color”. Since there are a large number of attributes that vary in likelihood and visibility, this can become very challenging. Other areas of computer vision share similar challenges with binary attribute detection such as object detection and facial recognition (Cui et al., 2018; Zhang et al., 2014).

Zhang et al. (2014) explored part-based models that can localize anatomical features. This improves the detection of attributes during classification tasks. Cui et al. (2018) focused on fine-grain classification on a large scale, similar to the problem we encounter in this study. Ghaffarian et al. (2021) and Li et al. (2023) explored how attention mechanisms in remote sensing and medical imaging respectively can improve classification within neural networks. This was not largely explored while testing the model, but it seemed to decrease performance. Perhaps it would work better with spatial attribute data.

Chen et al. (2022) explored correlation-based loss functions to improve the detection of attributes, especially ones that often appear together. They focused on techniques such as Pearson Linear Correlation (PLC) and Spearman Rank Correlation (SRC). This technique was also touched on during the model construction, but it should be explored more in future studies.

2.2 Multi-label Classification

Multi-label classification is the process of predicting multiple outputs from a single input. In the case of this study, we are classifying whether each of the 312 attributes is present from a single image. A large challenge of this is the class imbalance where there are many more 0s (not presents) than 1s (presents). He and Garcia (2009) explore strategies to deal with this imbalance issue. Some of the techniques they used were resampling and cost-sensitive learning. These methods made a big impact on improving accuracy.

More recently multi-task models have been deployed to successfully improve multi-label classification problems. Yiguan et al. (2022) experimented with neural networks to classify

animal species and detect their attributes simultaneously. This leverages shared representation because species and the species' attributes can be highly correlated in some cases. Cui et al. (2019) proposed a loss function that can account for class-imbalanced data. Custom loss functions such as these have been able to improve the performance of tasks like binary attribute classification of the CUB-200-2011 dataset. The Python library YOLO (You Only Look Once) is currently a leading framework for computer vision applications and may be better to use for this project than a custom CNN, however, it is much more computationally expensive. You can also create a custom loss function using the YOLO framework (Redmon et al., 2016).

The utilization of these techniques in attribute and multi-label classification has proven useful. However, due to many challenges such as class imbalance and imperfect solutions, there is still much to be improved upon. For this project, there was limited time and computing power to test or fully explore many of these techniques. These techniques will be good next steps.

3 Methodology

The task of binary attribute classification for the CUB-200-2011 dataset was approached using a Convolutional Neural Network (CNN). We will explore how the dataset was preprocessed, what image augmentations were applied, and what model architecture was used. We will then describe the loss function, optimization steps, and what performance metrics will be used.

3.1 Data Engineering

The CUB-200-2011 dataset provides annotations for bird species, bounding boxes, parts, and attributes in text files. The dataset is organized into an image directory containing 200 subdirectories, each corresponding to a bird species. Each species directory has about 59 images, resulting in 11,788 images in total.

To preprocess this data, a script was developed using the Pandas library in Python which can be used to store information in DataFrames (McKinney, 2010). The script constructed a DataFrame with three columns: image ID, image path, and a list of associated attribute annotations. This format structure made it easier to access the information needed.

The dataset was split into training and validation sets using the Scikit-learn library which provides a variety of useful machine learning tools (Pedregosa et al., 2011). The data split command was given a seed of 200 for reproducibility. After splitting the data, it was loaded into PyTorch tensors using PyTorch's DataLoader (Paszke et al., 2019). PyTorch was also extensively used later during model construction. A batch size of 32 was used and the training data was shuffled before training to improve generalization.

3.2 Image Processing

Image preprocessing can improve CNN performance by making the images consistent. Images were resized to 64 × 64 pixels using OpenCV (OpenCV Team, 2024). This resolution was chosen mainly for computational efficiency. Higher resolutions, such as 128 × 128 or 256 × 256, were tested but yielded only small increases in performance metrics while significantly increasing training time.

Each of the image's color channels was normalized to a [0, 1] range by dividing the pixel values by 255. The image was also normalized. Given the difference in color channel orders between OpenCV (BGR) and PyTorch (RGB), images were converted from BGR to RGB and transposed to match PyTorch's requirements. These steps are important so that the data behaves correctly during the CNN training.

Data augmentation techniques, such as noise addition, rotation, and cropping, were initially tested but led to reduced accuracy. This augmentation was not used in the final image processing pipeline.

3.3 Model Architecture

This classification model is a custom-designed CNN trying to balance efficiency and performance. The architecture consisted of four convolutional blocks, each comprising of:

- Main block: A convolutional layer with kernel size 3, stride 1, and padding 1, followed by batch normalizations and ReLU (Rectified Linear Unit) activations. This was done twice. Convolution layers can single out small parts of images at a time (kernels) to pick out small details making it good for computer vision tasks (Lecun et al., 1998). ReLU activation function incorporates non-linearities into the model

letting it learn complex patterns (Goodfellow et al., 2016). Batch normalization can help stabilize the model by normalizing the layer inputs (Ioffe & Szegedy, 2015).

- Skip connection: A single convolutional layer with kernel size 1 and stride 1. It mapped the input dimensions to the output dimensions. This helps with the stabilization of gradients (He et al., 2016).
- Max pooling: A pooling layer with kernel size 2 and stride 2 down sampled feature maps at the end of each block. Max pooling is important for model efficiency while filtering out unnecessary data that can be considered noise (Zeiler and Fergus, 2014).

The blocks progressively mapped feature dimensions higher. First 3 (for RGB) to 32 and then the blocks doubled the dimensions until reaching 256. After the convolutional layers, an adaptive average pooling layer generalizes features in the data. Finally, the tensor was flattened and passed through a fully connected network with two hidden layers (512 nodes each) and a final output layer producing logits for 312 attributes. The fully connected layer essentially lets the model make a classification decision (LeCun et al., 1998; Goodfellow et al., 2016). Dropout layers (p=0.5) were added to generalize and prevent overfitting (Srivastava et al., 2014).

This architecture, while simpler than pre-trained models like ResNet (He et al., 2016), was chosen for its computational efficiency, given limited hardware resources.

3.4 Loss Function

For multi-label binary classification, Binary Cross Entropy with Logits Loss was selected. This loss function integrates sigmoid activation internally. This makes it more stable than using standard Binary Cross Entropy Loss with a separate sigmoid activation. To address the class imbalance in the dataset, a weight was applied to the positive class, which tended to greatly increase the F1 score. Two weight values were used, 3.5 and 5.0.

3.5 Optimization

The optimizer used was AdamW (Loshchilov & Hutter, 2019). This is used to generalize gradients and improve optimization by using weight decay. A learning rate scheduler with a step size of 5 and a decay factor (gamma) of 0.5 was also utilized to prevent exploding gradients. The initial learning rate was set to 0.0001 and it steadily decreased. Gradient clipping was also used

to fight against exploding gradients, The model was trained for 25 epochs. Each hyperparameter was experimented with and the best set was chosen.

3.6 Performance Metrics

When training models it is important to define what metrics should be optimized and monitored. For this study, we will use training loss, training accuracy, validation accuracy, precision, recall, and F1 score. F1 score will be used a the primary metric since we have highly imbalanced classes and we want to minimize both recall and precision.

Training loss measures the error between the model's predictions and true labels on the training dataset this is calculated by passing the prediction and targets through our loss function. If the training loss is decreasing, that means the model is learning how to better fit the training data. If the loss gets too low then the model may be overfitting (Goodfellow et al., 2016).

Training accuracy goes hand-in-hand with training loss but instead gives a proportion of correctly classified items in the training data. High training accuracy is generally good as long as the model is not overfitting. If this is the case, we need to adjust the model (Géron, 2019).

Validation accuracy evaluates model performance similar to training accuracy but on an unseen dataset. Monitoring validation accuracy helps us see if the model is overfitting, and early stopping is often used with this metric (Chollet, 2018). If the gap between training and validation accuracy does not change, then hyperparameter tuning or a change in the model architecture may be needed. (Yosinski et al., 2014).

Precision is the ratio between the number of predicted positive labels that are correct and the total number of predicted positive labels.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

High precision is important in situations where you do not want to raise false alarms such as fraud detection (Saito & Rehmsmeier, 2015).

Recall (or sensitivity) is the ratio between the number of predicted positive labels that are correct and the total number of predictions that should be correct:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

High recall is necessary when you do not want to miss something that may be hazardous like in cancer detection (Davis & Goadrich, 2006).

The F1 score balances precision and recall by taking their harmonic mean:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

This is very valuable for instances with imbalanced datasets, where optimizing for one metric alone may lead to predicting all values one way or another. It is possible with these scenarios to predict everything as not present and get over a 90% validation accuracy. A high F1 score means that both false positives and false negatives are relatively low compared to true positives (Lipton & Tripathi, 2017).

Looking at all these metrics together can help us to understand what is really going on with the model and can provide insights into how to improve it.

4 Results and Analysis

The six metrics were recorded at every epoch to measure the model's performance, training loss, training accuracy, validation accuracy, precision, recall, and F1 score. We will compare two identical models with differing loss function weights of 3.5 and 5.0 over the 25 epochs. An exploration of the occurrence frequency of each attribute will be conducted. The model at the epoch that achieved the highest F1 score will be evaluated on individual attribute subcategories. We will dive deeper and explore which attributes the model could predict well and where it fell short.

4.1 Model Performance

The 3.5-weight model gradually decreased loss and increased training accuracy indicating it learned from the data. Training accuracy increased from just above 87% to almost 91% by epoch 25. Validation accuracy jumped around between 89.5% and 90% while the precision and recall tried to balance each other out. It ended up decreasing slightly. Precision didn’t change much and jumped around between 0.48 and 0.50. However, the model made much bigger strides with recall

increasing it from about 0.28 until it stabilized around 0.45 around epoch 10. The model's F1 score also rose from 0.35 and stabilized at 0.47 around epoch 8. Overall, it started with weighting its predictions towards the not-present class and worked on improving recall while holding precision steady to achieve its F1 score.

The 5.0-weight model also gradually decreased loss and increased training accuracy meaning the model kept training. Training accuracy is lower than the other model starting at 85% and increasing to 89% by epoch 25. Validation accuracy increased slightly initially before jumping around 88%. Precision started at 0.4 and increased to and stayed around 0.44 for most of the following epochs. This model also saw a large increase in recall starting at 0.38 and stabilizing at 0.55. The model's F1 score rose from 0.39 and stabilized at just above 0.48 just after epoch 5.

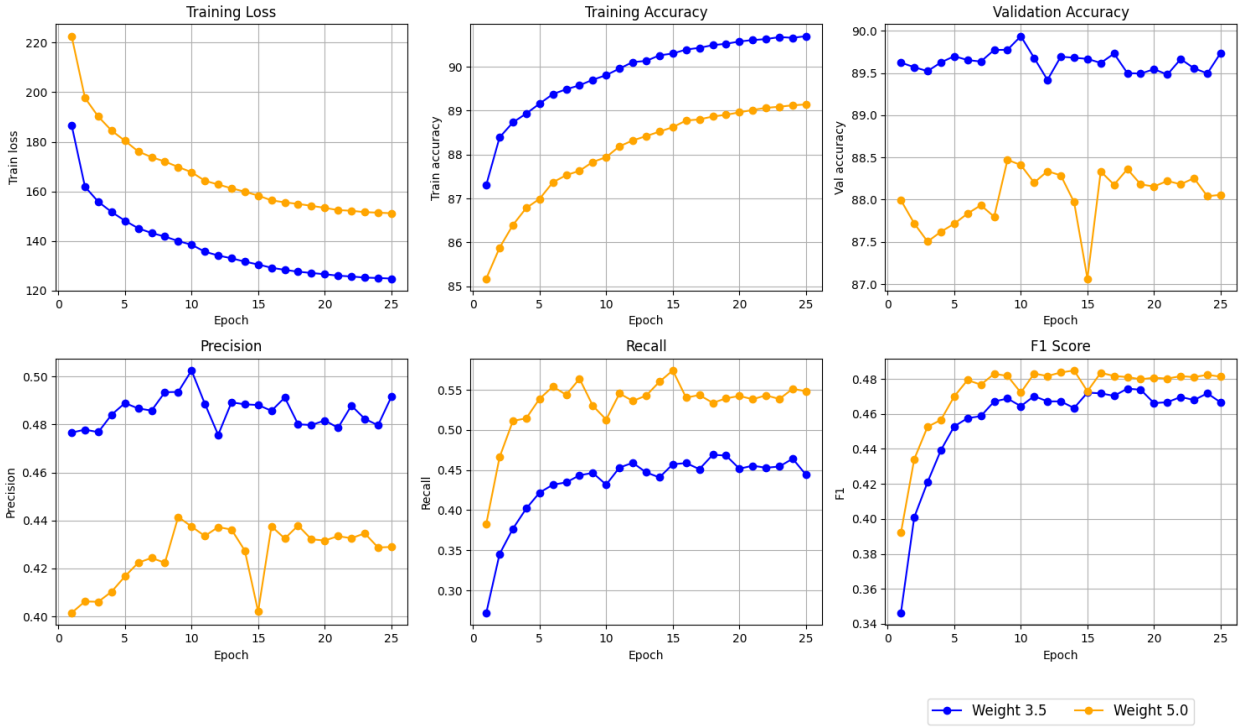

Figure 1. Compares the performance metrics between the model with a 3.5 and a 5.0 positive class weight. The metrics included are training loss, training accuracy, validation accuracy, precision, recall, and F1 score.

The 3.5-weight model did a bit better in validation accuracy and favored precision over recall. The 5.0-weight model prioritized recall and achieved a higher F1 score. In regards to the F1 score, the 5.0 weight model at epoch 14 achieved very close to 0.485. We will use this model

for further analysis. Overall, this performance is not terrible, but it can be greatly improved upon. Since its recall was 0.56 that epoch, of the attributes that should've been detected, it detected a little over half of them. Since its precision was only around 0.43, it indicates that a bit more attributes were detected that weren't there than attributes that were.

4.2 Attribute Category Imbalance

Before measuring the performance of each attribute category, we will look at a stacked bar chart taken from the entire dataset. Each bar represents a category, and each stack on the bar represents an attribute in that category. The number of stacks per bar ranges from 4 to 15. Attribute 1 would be the first attribute listed in the category as shown in the CUB-200-2011 dataset. Attribute 15 would be the last. The combined proportion of some bars is greater than 1 because birds can have multiple attributes from some categories.

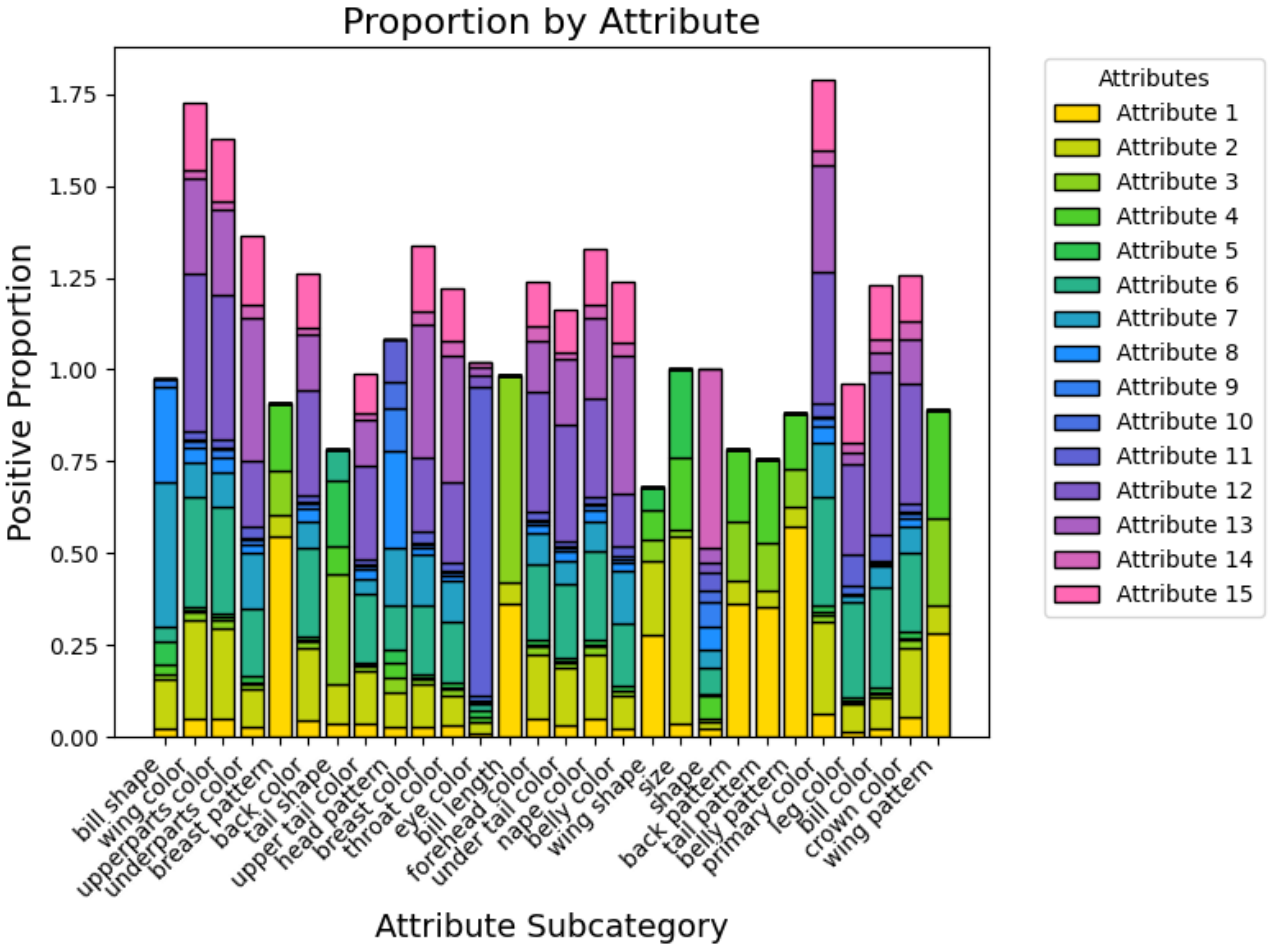

Figure 2. Displays a segmented bar chart where each bar is an attribute category, and each segment of a bar is an attribute. The height of each segment is the proportion of images in the dataset that contain that attribute.

We can observe from the attribute proportions that the prevalence of any given attribute varies widely. This ranges from less than 1% to about 75%. We also see that the distribution within attribute categories widely varies. For example, the head patterns across species are fairly distributed while the eye color is mainly dominated by black. The proportion of an attribute from a category present in the image also varies. Wing shape is the least annotated category with an average of around 0.7 per image. Primary color is the most annotated category with an average of 1.8 attributes per image.

4.3 Attribute Category Performance

Since the 5.0-weight model at epoch 14 performed the best with regard to the F1 score we use this model to analyze the attribute category class imbalances. We will measure the validation accuracy, precision, recall, and F1 score of each category to determine how well the model performs in each category.

The validation accuracy varies across categories ranging from 65% to almost 100%, with shape and eye color categories achieving near-perfect accuracy. Other shape and color-related categories also have high accuracies. Pattern and size-related categories didn't perform as well with bill length and wing pattern doing the worst. This indicates that the model can much more easily pick up colors and shapes rather than how big it is and fine-grained patterns.

Precision values fluctuate widely across the categories, with some achieving precision close to 0.9, while others show significantly lower values, around 0.3. Eye color by far has the highest precision, with the next highest around 0.6. Primary color and size do fairly well compared to other groups while leg color and wing shape do poorly.

The recall values range from low to moderate, with some attributes achieving values around 0.8 such as size, eye color, and bill length, indicating that the model can identify most of the positive cases in these categories. On the other hand, categories like head pattern and tail shape are not identified correctly.

The F1 scores, showing a harmonic mean between precision and recall very widely. With most categories between 0.3 and 0.6. Again, eye color does the best achieving the best balance between precision and recall, reaching over 0.8. Head pattern performs the worst, about 0.2, indicating it dies very bad with these positive predictions.

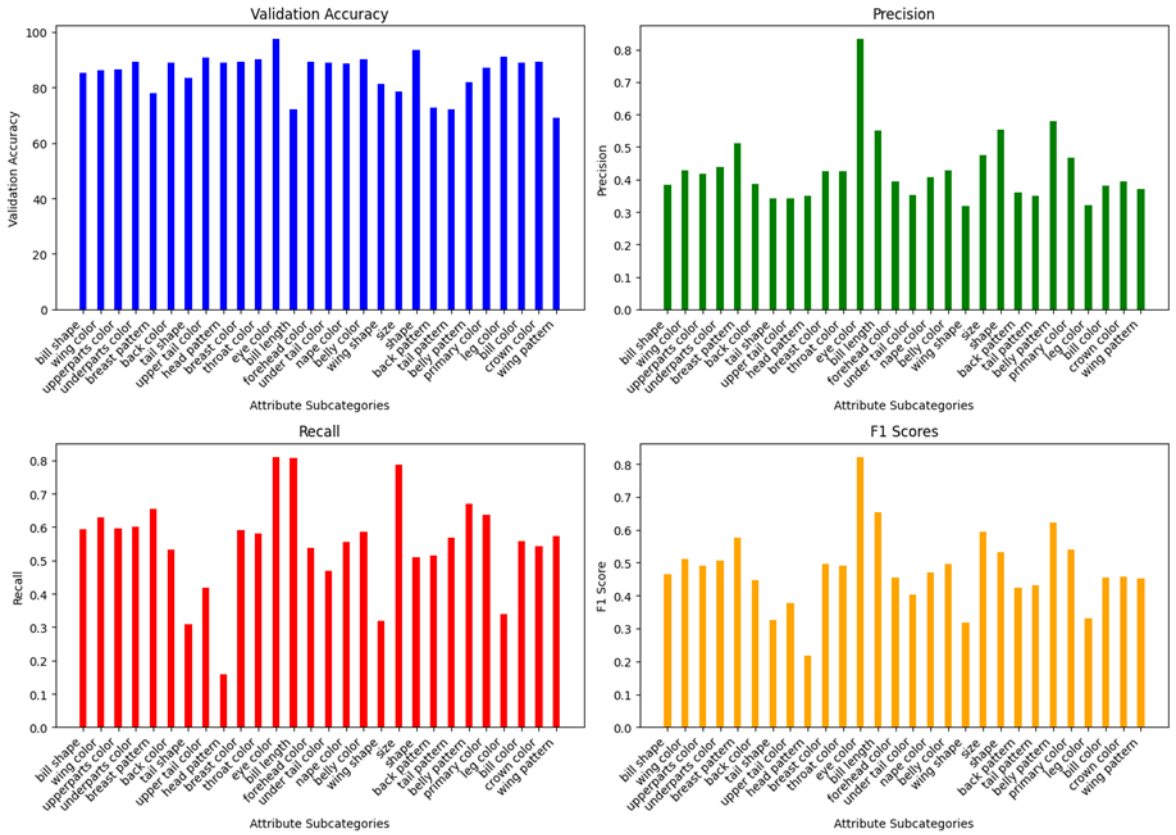

Figure 3. Displays the validation accuracies, precision, recall, and F1 scores for each attribute category.

We can see that even though pattern and size-related categories did not do well with validation accuracy some actually did quite well with F1 while others still didn't. The same pattern is evident with the categories with higher validation accuracies. The only category that seems to dominate in each of these four statistics is eye color. This achieves over 95% validation accuracy and has an F1 score of over 0.8. Recall and precision are both around 0.8 as well. This is surprising because the eyes are usually a small attribute and most of this category is dominated by the color black.

5 Conclusion and Future Work

In this study, a convolutional neural network was applied to binary attribute classification on the CUB-200-2011 dataset. Even with the class imbalances and hardware limitations, the model demonstrated improvements and was able to classify some attribute categories quite well, particularly eye color. It was shown that by weighting the model more towards to positive class

but not too much (around 5.0) we could optimize F1 without changing anything else. The model achieved mid-range F1 overall and struggled to surpass 0.5.

Many techniques can be explored to improve the large-scale binary attribute classification of birds. A deeper or pre-trained model with correlation-based loss functions and attention mechanisms may be able to greatly increase F1 while maintaining high accuracy. GPU acceleration will make this much more possible. Adding an additional head to jointly predict species and attributes may also improve this task. Adding parts segmentation and connecting them to the attributes or giving the spatial location of each attribute will enable the mode to learn more efficiently, especially on the small and very fine-grained patterns. Lastly, increased image resolution and additional data augmentation may be able to generalize the data for cases beyond the CUB-200-2011 dataset.

6 References

Amazon Mechanical Turk. (2024). Crowdsourcing platform for dataset annotations. Retrieved from https://www.mturk.com

Chen, C., Yang, X., Huang, R., Hu, X., Huang, Y., Lu, X., Zhou, X., Luo, M., Ye, Y., Xue, S., Miao, J., Xiong, Y., & Ni, D. (2022). Fine-grained Correlation Loss for Regression. International Conference on Medical Image Computing and Computer-Assisted Intervention. https://doi.org/10.48550/arXiv.2207.00347

Chollet, F. (2018). Deep Learning with Python. Manning Publications.

Cui, Y., Song, Y., Sun, C., Howard, A.G., & Belongie, S.J. (2018). Large Scale Fine-Grained Categorization and Domain-Specific Transfer Learning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4109-4118. https://doi.org/10.48550/arXiv.1806.06193

Cui, Y., et al. (2019). Class-balanced loss based on effective number of samples. CVPR. https://doi.org/10.48550/arXiv.1901.05555

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In Proceedings of the 23rd International Conference on Machine Learning (ICML '06) (pp. 233–240). Association for Computing Machinery. https://doi.org/10.1145/1143844.1143874

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.

Ghaffarian, S., Valente, J., van der Voort, M., & Tekinerdogan, B. (2021). Effect of Attention Mechanism in Deep Learning-Based Remote Sensing Image Processing: A Systematic Literature Review. *Remote Sensing*, *13*(15), 2965. https://doi.org/10.3390/rs13152965

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

Huang, C., Li, Y., Loy, C. C., & Tang, X. (2020). Deep imbalanced learning for face recognition and attribute prediction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(11), 2781-2794. https://doi.org/10.1109/TPAMI.2019.2914680

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. ArXiv, abs/1502.03167. https://doi.org/10.48550/arXiv.1502.03167

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. https://doi.org/10.1109/5.726791

Li, X., Li, M., Yan, P., Li, G., Jiang, Y., Luo, H., & Yin, S. Deep Learning Attention Mechanism in Medical Image Analysis: Basics and Beyonds. *International Journal of Network Dynamics and Intelligence*. 2023, *2*(1), 93–116. doi: https://doi.org/10.53941/ijndi0201006

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017 (pp. 2999-3007). https://doi.org/10.1109/ICCV.2017.324

Lipton, Z.C., & Tripathi, S. (2017). Precise Recovery of Latent Vectors from Generative Adversarial Networks. ArXiv, abs/1702.04782. https://doi.org/10.48550/arXiv.1702.04782

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. In Proceedings of the International Conference on Learning Representations (ICLR). Retrieved from https://arxiv.org/abs/1711.05101

McKinney, W. (2010). Data structures for statistical computing in Python. Proceedings of the 9th Python in Science Conference, 51–56.

OpenCV Team. (2024). OpenCV: Open source computer vision library. Retrieved from https://opencv.org

Paszke, A., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 32.

Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788. https://doi.org/10.1109/CVPR.2016.91

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLOS ONE. https://doi.org/10.1371/journal.pone.0118432

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958. DOI: 10.5555/2627435.2670313

Stokes, C. (2024). *Avian Multi-label Binary Attribute Classification* [GitHub repository]. https://github.com/colestokes5/Avian-Multi-label-Binary-Attribute-Classification.git

Wah, C., Branson, S., Welinder, P., Perona, P., & Belongie, S. (2011). The Caltech-UCSD Birds-200-2011 Dataset (Published). California Institute of Technology.

Yiguan Liao, Changzhen Qiu, Zhiyong Zhang, Jiejun Chen, Jiajun Zheng, Keyuan Su, Haoran Li, & Liang Wang. (2022). *Animal attribute recognition via multi-task learning based on YOLOX: A multi-task learning network based on YOLOX to realize target detection and attribute recognition at the same time*. In *Proceedings of the 2021 5th International Conference on Video and Image Processing (ICVIP '21)* (pp. 7–12). Association for Computing Machinery. https://doi.org/10.1145/3511176.3511178

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? ArXiv, abs/1411.1792. https://doi.org/10.48550/arXiv.1411.1792

Zeiler, M.D., & Fergus, R. (2013). Visualizing and Understanding Convolutional Networks. ArXiv, abs/1311.2901. https://doi.org/10.48550/arXiv.1311.2901

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters, 23*, 1499-1503. https://doi.org/10.48550/arXiv.1604.02878

Zhang, N., Donahue, J., Girshick, R.B., & Darrell, T. (2014). Part-Based R-CNNs for Fine-Grained Category Detection. *European Conference on Computer Vision*. https://doi.org/10.48550/arXiv.1407.3867

Zhou, G., et al. (2021). Learn fine-grained adaptive loss for multiple anatomical landmark detection in medical images. *IEEE Journal of Biomedical and Health Informatics, 25*(10), 3854-3864. https://doi.org/10.1109/JBHI.2021.3080703