Deep Learning Assignment - Bird Species Classification

Overview - Main objective

We are working for a company that is developing a nature observation application where the user is able to upload a photo and our deep learning models will produce details and information about the main object of the photo. Our application is already able to identify whether an uploaded picture contains a bird. The Chief Data Officer (CDO) has requested our team to spend time enhancing the application by developing an image classification deep learning model that is finetuned on data of different bird species. Once the model identifies the bird species, our application will pull information from different sources and provide them to the user.

For this project, we are only concerned with developing a bird species classifier from bird images. A different team is taking care of the part that provides information to the user. We have also clarified with the CDO that the existing classifier that identifies birds vs other objects will stay in place. Once the first model classifies something as a bird, our model will be used for more specific classification. The CDO and upper management expect that this enhancement will increase the appeal of the app, drawing in more users and additional traffic.

The CDO has said that high accuracy is important, but we should aim to develop small models that can be used even in edge devices.

The main objective of this report is to provide an overview of the data that was used to finetune this model, the steps to analyze and clean the data, the models that were trained and how to compare to each other as well as larger variants (to observe whether there is significant drop in accuracy when using small models) and what is our final recommendation.

Data

We will be using a dataset of 525 different bird species that has been gathered from a variety of internet sources. The dataset contains 84,365 images (on average 160 images per species, before cleaning) that can be used to train the model, an additional 2,625 images in a validation set (5 per species) and a final 2,625 images in a hidden test set. The first two sets of images will be used to train and validate our model, while the test set will only be used at the end of the process to confirm model performance. Most of the images are sized 224x224 using RGB channels and have been stored in JPG format. Our investigation into the data revealed there were cases with odd sizes and ratios. More on that in the next section.

The data is available to download from Hugging Face. It was originally uploaded to Kaggle but is no longer available there (details and links on the Hugging Face page for the dataset).

With this analysis and model development exercise we will attempt to create an image classifier for these 525 bird species. Of course, even with 525 species, our model might be limited in identifying bird species due to the vast ecological diversity and potential lack of representation in the training dataset. However, we believe that this approach is an important first step and can hopefully be expanded in future releases once we are able to get our hands on more data and images for additional species.

Data Exploration and Data Cleaning

Before we begin developing models, we spend time exploring and cleaning the data, to ensure we remove duplicates and bad quality images to avoid causing performance issues for our models. We run this analysis across all three sets of data, namely train/validation/test. The train portion is cleaned to improve model performance while validation and test are cleaned to ensure our metrics represent the actual capabilities of the model.

To identify exact or near duplicates, low information and odd photos, we leverage the cleanvision library. Cleanvision managed to identify potential issues with 220 images across the three splits of the data. There were 211 images which were tagged as having odd size, 7 that were blurry, 2 that were near duplicates (one duplicate) and one image with an odd aspect ratio (same as one of the images flagged with odd size, therefore 221 issues affecting 220 images).

The size issue mostly lies with images for a particular bird species, plush crested jays (only in train and validation splits, while test is normal), with only one image coming from the loggerhead shrike. We will resize all images to 224x224 as part of our image transformations anyways so this is not expected to be an issue. We will investigate performance of the final model on the test set for that species. On the single image of the loggerhead shrike, we observe that this is actually smaller than 224x224 but we do not expect performance issues considering the dataset size.

We reviewed the images classified as blurry and observed no noticeable blur that would affect model performance post finetuning. Finally, there is a picture of the same bird identified as both a band tailed guan (train/055) and a blue throated piping guan (train/152). Inspection of the data revealed the correct label is blue throated piping guan, therefore the first instance was removed.

In an initial attempt to perform data augmentation, we will also define transforms of the data that would flip or rotate the images every time a batch is fed into the model during training. We note that further enhancements in terms of data augmentations will be investigated in future versions of the model, where we will attempt to emulate weather conditions, such as foggy images. For this instance of the model we will try to make it robust to the various angles of the pictures taken by the app users.

Deep Learning Models - Image Classification Algorithms

In this section, we explore different vision model architectures as well as different size variants for each architecture to compare and benchmark accuracy, prediction time and model size, all of which are important factors to consider when we intend to put that model to production. We will explore variants of ResNet, EfficientNet and ConvNeXt models.

All above model architectures have already been trained extensively on the ImageNet dataset, which already contains different types of birds. Therefore, instead of starting from scratch, we will aim to finetune these models from available checkpoints, with the assumption that the model will have already come up with bird-related features in its initial training phase. In the ResNet section, we will test that assumption on the ResNet34 variant by training a version from zero as well as finetuning.

For each attempt at finetuning, we apply the image transforms that were used in the initial training of the model. These steps are available when loading the model weights through torchvision. For the ResNet34 that was trained from the beginning on this dataset, we mimic the transforms that were used in the original ResNet training. Additionally, to improve model performance, augmentations to the original images were considered and applied during training on each batch. This means that each batch was randomly modified at training time (different augmentations potentially applied each epoch) to make the model more robust and reduce overfitting. Augmentations applied were random horizontal flipping, perspective, rotation and affine transformations. Finally, different learning rate regiments were tested and it was chosen to apply one-cycle policy to adjust the learning rate over the training procedure, while overall the AdamW optimizer was used in every training scenario. These settings seemed to materially improve model performance in terms of accuracy on the validation dataset by a couple percentage points.

Pretrained models are finetuned for two epochs where only the final/classifier layer is trained and then for a further ten epochs. Model weights are saved for each epoch so that we can use the weights that achieve best performance on the validation dataset, in case after a few epochs the model overfits the training data and performance on validation data degrades.

In the following subsections, we will present the performance of these models. We report on the best accuracy on the validation dataset for each epoch (pretrained models were trained for 2 warm-up epochs and 10 additional epochs where all parameters were unfrozen), the size of the weights in Mbs and the average inference time for a single image in seconds. The average inference time was calculated by getting the average of the inference time for the validation dataloader and dividing that time by the size of the validation dataset (2625 images).

ResNet

You can check below table for the different variations of ResNet that were trained and best performance achieved, in terms of accuracy. Average prediction time and model size are also presented.

Model	Accuracy (Best)	Weight Size (Mb)	Avg Inference Time (s)
ResNet18	94.12%	45.78	0.012
ResNet34 (no-pretraining)	91.2%	86.22	0.023
ResNet34	94.27%	86.22	0.023
ResNet50	94.61%	98.34	0.036

First of all, we can compare accuracy between the two versions of ResNet34 that we trained. We can see that the one that was trained from scratch does not achieve the same level of performance and actually underperforms all variants. Therefore, going forward only pretrained models for all architectures will be considered. In addition, we can see from the table that performance peaks around 94% for all ResNet models and does not materially improve with larger variants. Our proposal would be to use the smallest variant, ResNet18, as a candidate from this family of models, considering it performs almost the same as ResNet34 and is only less accurate by 0.5% when compared against ResNet50. Achieving small model size and faster inference time with a small sacrifice of accuracy is desirable and we can see that inference for a single image doubles and triples as we go larger in size.

EfficientNet

There are many variants of the original EfficientNet model ranging from B0 to B07. For this exercise the first three (B0, B1, B2) were considered. The table below summarizes the results:

Model	Accuracy (Best)	Weight Size (Mb)	Avg Inference Time (s)
EfficientNet B0	96.09%	18.72	0.015
EfficientNet B1	96.1%	28.74	0.025
EfficientNet B2	97.54%	33.76	0.039

We can immediately see that EfficientNet seems to perform slightly better than ResNet, as even the smallest model, B0, is able to outperform ResNet50 (the largest we tested). Inference time are comparably increasing as a larger architecture is used and it is only slightly higher than the time achieved by ResNet. Finally, parameter size is quite smaller, the largest EfficientNet (B2) that we tried is smaller in size than the smallest ResNet. Our initial conclusion is that for a very immaterial increase in inference time, EfficientNet is a more appropriate architecture compared to ResNets. We can achieve impressive accuracy even with the smallest variants.

ConvNeXt

We only examined the smallest version of this architecture, ConvNeXt Tiny, as training time was significantly longer for this architecture compared to ResNet and EfficientNet. The model took twice as long to train for the same number of epochs as EfficientNetB2, and the results were not significantly better. See below for an overview:

Model	**Accuracy (Best)**	**Weight Size (Mb)**	**Avg Inference Time (s)**
ConvNeXt Tiny	95.82%	112.87	0.034

It is clear that performance is slightly better compared to ResNet50, in terms of accuracy and inference time, while its size is slightly larger. It does not seem to perform better compared to EfficientNet.

Findings

Our research has yielded several interesting findings. First of all, it is clear that we see a benefit from using pretrained models to adapt to our problem domain. All models tested perform quite well in terms of accuracy but there are additional considerations in terms of model size, as we want the model to be available in devices, and inference time, as we do not want to reduce user satisfaction of our application. Users expect the application to be fast. The reason model size is a consideration is that the business respects user privacy and offers them the potential to run the model locally on their devices.

Comparing the different architectures, EfficientNet seems to outperform both ResNet and ConvNeXt models in terms of accuracy and model size. B0 and B1 achieve similar levels of accuracy for the same amount of training, while B2 provides an additional 1.5% of accuracy by almost doubling the size and inference time compared to B0. However, in all cases, inference time for a single image is negligible in comparison to human perception. Here we need to note that inference depends on hardware, so we cannot necessarily expect similarly low times in lower-spec devices. The inference time can only be viewed as a comparison across models.

Our suggestion is to leverage the EfficientNet B0 model, as it offers the best balance between accuracy, model size and inference time, in the latter only slightly being outpeformed by ResNet18. In case the business ever decides to offer dedicated inference over API (currently does not to allow users to keep all their images local), we could consider using a larger variant like B2, but the benefits at this point are going to be marginal.

Flaws and Future Improvements

So far, we have briefly touched upon some of the potential flaws in our model development strategy. In this section we will reiterate and propose venues of future remediation.

Firstly, our model was trained on a set of images for a small number of species (525). While we expect the majority of our userbase lives in areas where the species can be found in nature, it is clear that we do not cover the entirety of potential species that can be encountered in the wild. Our models could benefit from training on larger datasets, potentially leveraging the LASBIRD data (Large Scale Bird Recognition Dataset).

Additionally, based on the current training and validation data we have at our disposal, the classifier was only trained on clear images of birds, typically under ideal conditions. We do not expect the average user will be trying to predict the species on such images. The model will need to be re-trained on either an expanded dataset that contains images under different lighting and weather conditions, or we will need to simulate light, weather and blur effects by randomly modifying hue, contrast, brightness and other aspects of the image during batch training. However, to be able to properly evaluate our classifier, we will need appropriate enhancement of the validation dataset. Otherwise, we will potentially observe model performance degradation on our validation data.

Finally, one further venue of future exploration would be to try training even larger models and using distillation strategies (teacher-student) to see if we can improve the performance of our smaller models.

Conclusion

Our project investigated the feasibility of designing a bird species classifier that can be used to enhance user experience of our app by providing additional functionality. We provided an overview of the data used and demonstrated the method used to review and clean it before training our models. We tested different model architectures and variants and compared our models on accuracy, model size and inference time. The EfficientNet architecture was the clear winner in terms of accuracy and size, outperforming both ResNet and ConvNeXt models, while having comparable but slightly worse inference time compared to ResNet. In general, we did not observe material reduction in accuracy when using smaller variants compared to their larger counterparts and concluded that we can use smallest available model for our production pipeline. We tested the EfficientNet B0 model on our unseen test data that was not used for either training or validation and observed an impressive 98.12% accuracy score, which further assures us that our training process was successful. Our overall recommendation at this stage is to use EfficientNet B0 for our application and to continue using this architecture for future iterations and improvements. Next steps for our project, should it be greenlit by the CDO, would include acquiring and cleaning a larger dataset of images, designing a strategy to maintain model accuracy under less ideal conditions of input image, and investigate whether large model variants can be leveraged for a distillation strategy.