455 Final Project Report

Problem Description:

Image classification is a traditional challenge in the field of computer vision. In this final project, we worked on a bird image classification competition on kaggle. There are 555 different classes of birds in total, and the task is given the image of a bird, our model should be able to tell the class or species of this bird.

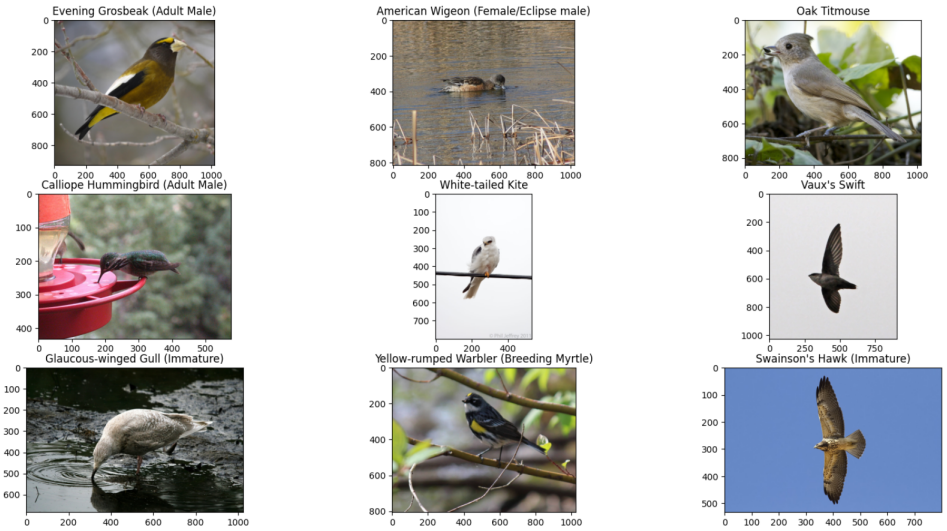

Fig 1. Visualization of some samples in the dataset

Datasets:

Although classification is a traditional problem, this task is not simple. The difficulty not only comes from the huge number of classes, which gives little space for assigning a correct class by chance. Also, the number of images in each class is not even, which means that the model may have a poor performance on the minority class. Lastly, the input images are of various dimensions. Our model should be able to deal with all these difficulties.

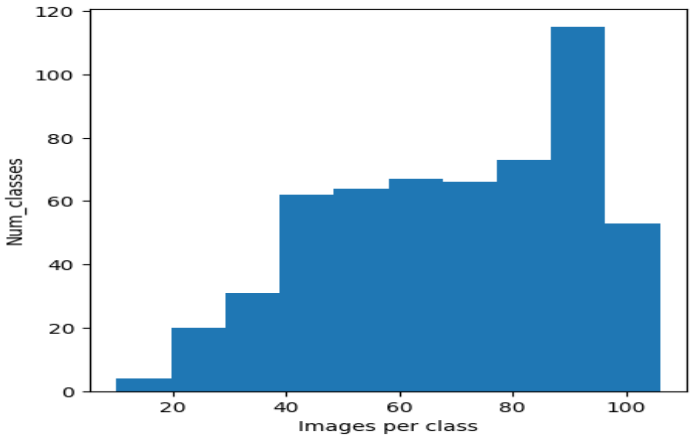

Fig 2. Histogram of images per class

Previous Work:

Since our team members all have certain amounts of background knowledge, we were able to use some of the state of the art classification models. We tried ResNet, ResNeXt, ConvNeXt, EfficientNet, Swin Transformer, and MaxViT. Eventually we adopted Convnext, swSwin Transformer V2, and MaxViT to make an ensemble. There are some code references from previous competitors that we referred to, and they are listed as follows: *Data Preprocessing Template*, *Training and Prediction Template*, *Ensemble Template*.

Our approach:

- Preprocessing

For the data, our first step was to generate augmented images, which were achieved by randomly rotating, flipping, shearing, greyscaling, and gaussian blurring some of the images. These augmented images will prevent overfitting during the training.

Fig 3. Augmented Images

Furthermore, as we already pointed out, the image per class distribution is uneven, so we should do something to balance this. Our approach was to use a weighted random sampler. Each class

with more instances will have a smaller weight, and the minority class will have a higher weight. The weighted sampler will sample images from the dataset based on the weight assigned to each class, so that each image will have a relatively fair chance to be sampled as a training image, and the training data set will have a relatively even distribution of classes.

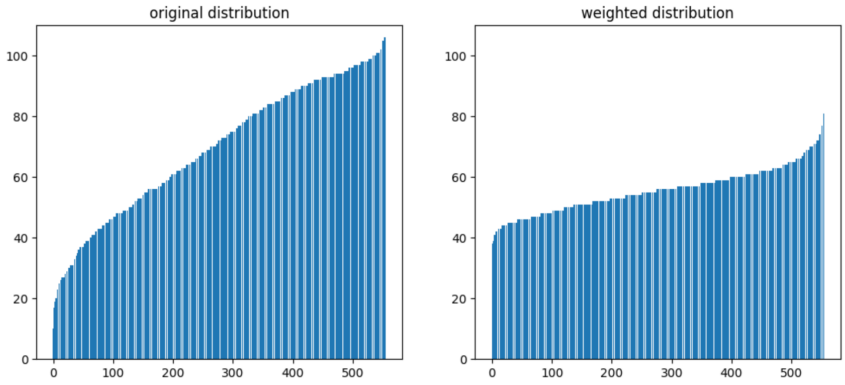

Fig 4. Re-Weighted distribution

Note that using a weight sampler is more reasonable in real applications because it could make our model pay more attention to the skewed groups. However, in our attempts, we found that training with a weight sampler even worsens the test set accuracy. This might be due to the fact that the test set has fewer minority groups and focus more on the majority group. Hence, in our final submission to the competition, we actually didn't use the weight sampler in our training pipeline. But theoretically, it should be better to include that.

- Train/Validation/Test Set Splitting

Before we dive into the training process, we first split our dataset into two categories: training set, validation set, and test set. In this contest, we were given two folders of data. One is the ground truth dataset and one is the dataset with unknown labels. Hence, we split the first dataset into a training set and a validation set with a ratio of 8 : 2, and the second dataset automatically becomes the test set, where the accuracy is measured after we submit our prediction to the competition website. By the way, image augmentation transforms are only applied to the training set.

- Training

During training, we use CrossEntropy as the loss function as the principal performance metric. We employ SGD with custom weight decay (0.0005) and momentum (0.9) as the optimizer. We use the biggest batch size possible as long as we have enough GPU memories, and we use a batch size of 64 for smaller models and 32 for bigger ones. We also make use of the learning rate scheduler.

- Models and Ensemble

We have been experimenting with various CV attention models, including MaxViT, ConvNeXt, and Swin Transformer V2. For the model size, we choose the tiny version for MaxViT, the small version for ConvNeXt, and the tiny version for Swin Transformer V2. Among the three models, MaxViT performs the best, which reaches an 83.65% accuracy in the test set. Then is ConvNeXt, which reaches an 81.85% accuracy in the test set. Finally is Swin Transformer V2, which reaches an 81.55% accuracy in the test set. In our final submission, we also perform model ensemble to integrate all three models together to get a better performance. We do grid search to find the best coefficients between the three models so that they have the highest accuracy on the validation set using the coefficients. We finally choose the coefficient between MaxViT, ConvNeXt, and Swin Transformer V2 to be 0.39 : 0.26 : 0.35.

Results:

The final pipeline achieved a score of 88% on the test set, which is the 1st rank in the public leaderboard.

Discussion:

Our approach is different from the common pipeline by adopting an ensemble instead of just a single model. It seems that a complex model is preferred. Also, we resampled the image for our train dataset.