POLITECNICO DI MILANO

Artificial Neural Networks & Deep Learning

Homework1

Team:

Persepolis

Students:

Ali Noshad - 101108

Kiarash Rezaei - 991495

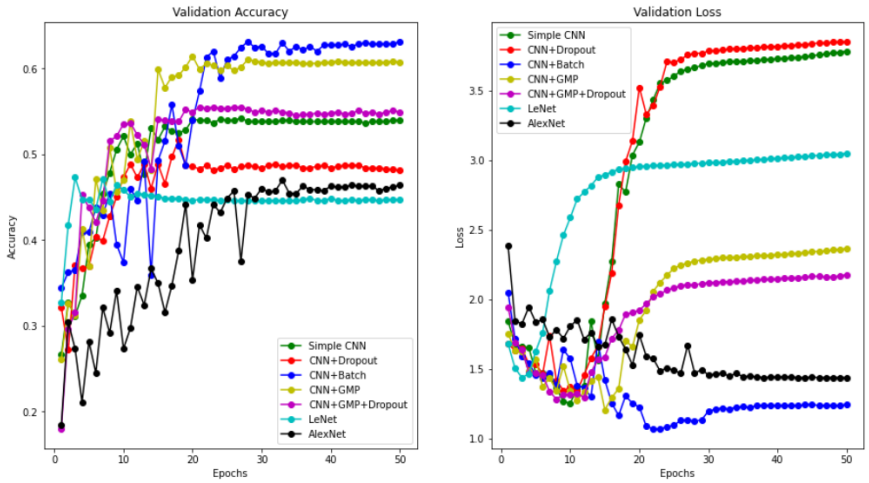
Ashkan Ramezani - 101150

ACADEMIC YEAR 2022-2023

1. Data exploration and Visualization: First, in this work, we analyzed data provided for the challenge based on the two following aspects: RGB statistical distribution and class distribution. Regarding the first analysis, we understood all channel distributions are approximately the same. Thus, we could not extract any specific information from each channel. Performing the second analysis, we found out two main issues, one was that we had an imbalance dataset due to the sizes of class "species 1" and "species 6" (with 186 and 222 images, respectively). Another problem was that in total the amount data was quite low, since deep learning approaches always involves with a large number of parameters, that need to be tuned (trained) so having appropriate amount of data is crucial, otherwise our model will be overfitted on the training data and it cannot be generalized well during the test time that affects the final prediction. Our approach for coping with these challenges was to benefit from Data Augmentation techniques (to increase the size of our dataset) and Transfer Learning (to improve the ability of generalization of our model).

2. Data Splitting: During experiments, in the first phase, for selecting appropriate models and hyperparameter tuning, since the number of submissions were limited, we were required test and fine tune our model locally and then submit the best obtained results, in this regard and to have more unbiased estimate of the challenge test set, we divided the dataset into 3 parts, 20% of data were considered as the test set, and the rest of data were divided into 80% and 20% for training and validation, respectively. The training data used for training the model. The hyperparameter tuning procedure was done using the validation data, and finally the best obtained results were considered for testing on the local test set. However, after selecting the best model and tuning the hyperparameters, in the second phase, we divided the whole data into two parts: training and validation in which we considered 90% images as the training and the rest (10%) as the validation set. Because in this way we were training on more samples and the model can generalize better on the test set of the challenge. For credibility of our experiments and accurate tuning of the models, in the first phase of our experiments we defined a seed for the random function which allows reproducibility of the results. This is how we could fully observe the effect of hyperparameters on the models and compare them after we tuned them.

3. CNN design from scratch: In the first step, we only used dense layers for classification that dealt with underfitting, and the accuracy (ACC) for validation data was 29.11%. Following this, we abandoned using only dense layers for prediction and we focused on convolutional neural networks (CNNs) which are more suited for image processing. CNNs have increased the performance substantially, first we started with basic models such as pure multi-layer CNN based on the one described in the lab and did some modification in the number of filters and layers. This CNN is composed of 6 "Convolutional" layers, 4 "MaxPooling" layers followed by two "Fully connected" layers with "ReLu" and "Softmax" activations, respectively. Using this approach, we managed to reach the accuracy of 98.99% on the training set, and obtained the accuracy of the 63.88% on the validation, by analyzing the results obtained on the validation and training phase, we could clearly notice the sign of overfitting of our model on the training set, since the gap between the obtained results were very large which meant that our model could not generalize well on the new data. Regarding this, we tried to improve the results by using "Dropout", "Batch normalization", and "GlobalMaxPooling" layers. Parallel to these experiments we also considered and implemented from scratch some proven basic models such as "LeNet " and "AlexNet", which yield good results in many tasks based on the research conducted in this field. Figure.1 illustrates the obtained results during the validation phase based on these modifications compared to these pre-defined architectures. According to the obtained results we found out all of these models are overfitted on training dataset, as we mentioned earlier, however doing some experiments with aforementioned layers we managed to moderately reduce the effect of overfitting and improve the performance with "Dropout", "Global MaxPooling" and "Batch Normalization". During the test time in both local test and challenge framework, we reached approximately %59 accuracy. However, we still suffered from the server overfitting issue, so we decided to test approaches which help to tackle this problem such as data augmentation and transfer learning, and also use more recent, advanced, and state-of-art architectures in next stages.



4. Data Augmentation: As previously mentioned, we decided to increase the size of the dataset to avoid overfitting. At this step, we tried to choose the most effective techniques based on the previously performed analysis on the dataset and visually checking images as well as trying techniques and comparing their corresponding effects on results. At last, we realized some techniques are better not to use (either not effective or results in worse prediction accuracy) due to the nature of our dataset. For instance, shifting, rotation, random crop, image lightening and adding noise were not effective. On the other hand, vertical and horizontal flip caused better prediction results. Zooming was also effective. We would like to discuss the rationale behind not choosing some techniques: for example, using the shifts in the data augmentation introduce null space in images which can create an issue, however, fill mode can be used to tackle this but we did not observe much improvements even by using fill mode. Regarding the rotation, we tried different degrees of rotation and based on the obtained results in the submission, with 90 degrees rotation we obtained the same results as before and other test degrees did not cause any improvement. Data augmentation helped in reducing the problem of overfitting and we managed to obtain more reliable results with aforementioned methods. For example, with models such as CNN+GMP and CNN+Batch we managed to reach approximately around 62% on the test set.

5. Transfer Learning: After making multiple modifications in our CNN architectures, we realized that the accuracy could not experience further improvements. So, we decided to use some state-of-art pre-trained architectures together with aforementioned augmentation techniques and make it more compatible with our task by putting a network on top of them. Regarding this, we started using some basic architectures and analyzed more advanced methods as we moved forward. We started with VGG16, VGG19, ReNet50, ResNet152, InceptionV3, MobileNetV2, EfficientNetB0, EfficientNetB4 and EfficientNetB6. We used these architectures pre-trained on the imagenet, and we just modified the top of the network in all the pre-trained architectures and optimized them to the task we are trying to solve. In the top we experimented with several layers such as "GlobalAveragePooling", "GlobalMaxPooling", different number of dense layers with different number of nodes, "Batch Normalization" and "Dropout". Furthermore, we analyzed the behavior of these architectures by manipulating the number of trainable layers and tuned these hyperparameters and model configurations on the validation set. Adding some layers were really useful, for example adding "GlobalMaxPooling", "GlobalAveragePooling" layers greatly reduced the number of parameters which could tackle the overfitting, however, some layers were not useful in all the models, for example "Batch Normalization" layer caused an improvement in EfficientNetB0 architecture both in validation and test, but in all other model caused a good results on the validation but it did not improved much on the test set and the submission part. Training with different batch sizes was also tested and the best ones were 16 and 8. Additionally, we analyzed the confusion matrix in the evaluation step and as it was anticipated, the number of true positives (TP) for class0 was not as good as the other classes (due to the unbalance characteristic of our dataset). Hence, we decided to weight classes based on the number of images they have -as we were fitting the model- to improve TP of class0. Although it

improved the predictions of class0, the overall performance worsened, and we did not use it anymore. Finally, we discovered that using a "GlobalAveragePooling" layer followed by only one "Dense" layer with 256 neurons and "Dropout" before the "Softmax" layer was considered the best approach in approximately all models. With regards to normalizing images in this part, we imported and used the specific preprocessing functions from tensorflow library and placed it in the "Imagedatagenerator" function, and we preprocessed the test images in the challenge framework using these functions as well. During training in all the experiments, we used the learning rate reduction with patience 3 monitoring the validation accuracy, and also, we used the model checkpoint to save the best model for submission based on the validation loss. Table 1 describes the obtained results regarding the submission of these fine-tuned architectures.

Model	Local Accuracy	Submitted Accuracy
VGG16	0.7408	~0.64
VGG19	0.7335	~0.59
EfficientNetB4	0.7003	—--
EfficientNetB0	0.7989	~0.77
EfficientNetB6	**0.8357**	**~0.80**
ResNet50	**0.9145**	**~0.83**
ResNet152	**0.9145**	**~0.81**
MobileNetV2	0.8215	—--
InceptionV3	0.5870	—--

6. Ensemble Model: Finally, to further improve our results we considered using an ensemble approach, based on well-known competitions, research and the professor's comment about ensemble techniques which can enhance the accuracy of the prediction up to 10%. Therefore, we used the three best models (highlighted above) based on their results on the submissions.We created the following 3 different ensemble models. First one comprised of EfficientNetB6 networks (submitted accuracy ~0.84), second one was a combination of EfficientNetB6, ResNet50 and ResNet152(was not submitted due to poor performance) and the last one included ResNet50 and ResNet152(submitted accuracy ~0.88). For the prediction we used the most voted technique, in which the most voted class is considered as the final prediction of the ensemble model. All the models retrained several times using different distributions of data on up to 90% of it. However, the third model could obtain the best performance. To be more specific about the last model, we trained and used 11 ResNet50 and 3 ResNet152 for the ensemble approach. At last, we could obtain an approximate accuracy of **0.86** on the final competition phase.

NOTE: Since models are retrained for several times, with the same build function and different data distributions, the exact submitted models (submission zip file) can be downloaded through this link.