IMAGE CLASSIFICATION CHALLENGE 2023

TEAM: CARS

Sophia Barbera 10661106 Caterina Giardi 10917727
Anna De Simone 10872083 Riccardo Storchi 10915631

1 DATASET SPECIFICATION

The initial dataset consisted of 5200 96x96 pixel images of plants, addressing a binary classification problem distinguishing between healthy and diseased plants. Dataset cleaning involved removing anomalous "intrusive" images, resulting in 5004 refined images. Further optimization eliminated duplicates, yielding a final dataset of 4850 unique instances.
For experimental analysis, the dataset was divided into 4365 training images and 485 validation images, aiming to enhance model performance. Rigorous evaluation metrics, including accuracy, specificity, recall, and F1 score, provided a comprehensive assessment of the model's predictive capabilities on both training and validation datasets. This systematic evaluation establishes the model's overall performance and potential for accurate real-world predictions.

2 DATASET AUGMENTATION

To address the challenge of a relatively small dataset, we applied a technique that eventually turned out to be crucial in terms of model performance: data augmentation. In the data augmentation process we incorporate a set of transformations like Flip, GridMask, CutMix, Translation, Rotation and Zoom right after the input layer.
This allowed us to apply random changes to each training batch, reducing overfitting without needing to increase the dataset size because instead of creating duplicates, we dynamically augmented images during training, avoiding the need to store physical copies. Furthermore, this not only prevents overfitting and model loss but also strengthens the model's overall robustness.

3 EXPERIMENTS

In this section, we want to list some experiments that we did but ended up not using in the final model.

3.1 CutMix & MixUp

In our model, we chose to primarily apply CutMix as the data augmentation technique, excluding MixUp from the augmentation pipeline. This decision was made upon observing that the simultaneous use of both techniques, CutMix and MixUp, led to a significant alteration of the images, resulting in a notable decrease in model performance. The drastic changes induced by the combination of CutMix and MixUp seemed to negatively impact the model's ability to learn meaningful features from the data. Consequently, we prioritized the exclusive use of CutMix, which provided a sufficient level of diversity in the training data without compromising the model's performance. This strategic choice was made after careful experimentation and analysis of the model's behavior with different augmentation strategies.

3.2 VGG 16 & NASNet

We initially experimented with VGG16, as covered in our lectures, but encountered suboptimal performance on our dataset. The validation accuracy plateaued at 70%, falling short of our expectations. Subsequently, we explored NASNet, which unfortunately yielded even poorer results, with a validation accuracy not surpassing 60%. Our efforts to improve model performance are ongoing, and we are actively exploring alternative architectures to enhance the accuracy of our classification model.

3.3 MobileNetV2

In our experimentation, we also decided to build a model that exploits MobileNet [], which was first introduced to us during lab lectures, and with which we had our first approach to Transfer Learning and Fine Tuning. The net, indeed, turned out to be nicely performant with the classification task given during the lecture. In particular we decided to use MobileNetV2; It outperformed VGG16 and NASNet, but its accuracy consistently plateaued at around 75%, regardless of pooling techniques. This led to a decision to omit MobileNetV2 from our final model selection.

3.4 Test Time Augmentation

Lastly we tried to improve the performance with the technique of Test Time Augmentation [2] generating augmented versions of the input image by flipping and rotating it and then making predictions on the various augmented images that we got. Despite that this turned out to be pretty difficult not knowing the structure of the data that will be tested on our model. Therefore we decided to look for better solutions.

4 FINAL MODEL

4.1 Hyperparameters

- [VAL_DATASET, TRAIN_DATASET]: we split the dataset in order to have 4365 training images and 485 validation images.
- BATCH: we used mini-batch gradient descent with a batch size of 32 samples.
- EPOCHS: we used a maximum of 200 epoch for the trainings; in the EfficientNetV2M we started with 200 epochs for the first one, then decreasing to 100 epochs for the second one, while in the ConvNeXtLarge and in the Ensemble we used 100 epochs in every training.
- PATIENCE: we used a patience of 20 epochs in the various trainings.
- OPTIMIZER: we used AdamW optimization in the EfficientNetV2M instead of Adam because we noticed that it was the ones working better for it, while we used Adam optimization for the ConvNeXtLarge and the ensemble.
- LEARING_RATE: we started in the first training from the default value of AdamW and Adam optimizers which is 1e-e (0.001) and then we used a lower learning rate, that is 1e-5, because we want our model to adapt itself to the new dataset in small steps to avoid losing the features it has already learned.
- LOSS: we used Categorical Cross Entropy since it performs better for our problem, given that we tried with the Binary Cross Entropy combined with the Sigmoid activation function and it resulted in worse performance.
- ACTIVATION FUNCTION: in conjunction with the Categorical Cross Entropy we used Softmax as activation function in the output layer.

4.2 Ensemble of EfficientNetV2-M & ConvNeXtLarge

In this project, the final choice of our image classification model architecture is based on the ensemble of two distinct network: EfficientNetV2-M and ConvNeXtLarge. Both models were implemented and trained.
Our goal was to create a robust and accurate model in which we incorporated data augmentation during training to enhance its generalization capabilities, furthermore, we employed strategies such as layer freezing, fine-tuning, and learning rate adjustments to raise the model's overall performance.

4.3 EfficientNetV2-M

One of the first networks that stood out in performance by performing training on our dataset was EfficientnetV2-M [3]. On the images given to the algorithm, data augmentation was applied to increase the samples on which the

algorithm learns. This model following CNN reports a GlobalAveragePooling(GAP) and a 2-unit dense layer, common in binary classification problems.

4.4 ConvNeXtLarge

By experimenting with the application of various convolutional neural networks (CNNs) in model transfer, we found that ConvexNetLarge [4] proves to be a pre-trained network that yields high accuracy, val_accuracy, and low loss values. Throughout the experimentation, we explored different model architectures. Some excluded dense layers but incorporated only a Global Average Pooling (GAP) layer, with parameters linking the GAP to the final output dense layer. Other models utilized a flatten layer, incorporating multiple dense layers with a decreasing power-of-two units, alongside batch normalization and dropout before the final output dense layer.

Ultimately, based on performance evaluations, we decided to adopt the approach with Global Average Pooling (GAP) for the input. This choice demonstrated superior results, striking a balance between precision and model generalization.

4.5 Fine Tuning

We applied to both models mentioned above the fine-tuning technique, which consists of updating the model weights while retaining the information learned during training on the original model. Specifically, in our case we unlock all layers but freeze the BatchNormalization and LayerNormalization layers (to maintain stability) and then recompile the model for training with the new settings. Before arriving at this design choice, we tried freezing only some layers while keeping others active, but this did not result in a significant increase in performance.

4.6 Ensemble

The excellent performance of these two models led us to think about how to get the best out of both. The answer to this question was to build an ensemble: a model that relies on the output of two model instead of one. The models are combined via the average operator in order to reduce the effect of errors or limitations of each model, producing a more robust prediction to noise.

REFERENCES

[1] MobileNetV2 https://keras.io/api/applications/mobilenet/#mobilenetv2-function

[2] How to use Test Time Augmentation https://machinelearningmastery.com/how-to-use-test-time-augmentation-to-improve-model-performance-for-image-classification/

[3] EfficientNetV2M https://keras.io/api/applications/efficientnet_v2/#efficientnetv2m-function

[4] ConvNeXtLarge https://keras.io/api/applications/convnext/#convnextlarge-function

[5] Ensemble https://machinelearningmastery.com/model-averaging-ensemble-for-deep-learning-neural-networks/

CONTRIBUTIONS

All group members worked consistently with what was agreed upon in the days leading up to the homework. We worked together facing the challenges that a new and complex project can bring while adapting as best we could in each situation.

Cleaning of the dataset: Anna, Caterina, Sophia, Riccardo

Augmentation: Caterina, Sophia

EfficientNet: Riccardo, Anna, Caterina

VGG: Anna, Caterina, Riccardo, Sophia

NASNet: Anna, Sophia

ConvNeXt: Riccardo, Anna, Caterina

Fine Tuning: Caterina, Anna

Ensemble: Riccardo

Report : Sophia, Anna, Caterina, Riccardo