Università Commerciale Luigi Bocconi

Lazzat Kengessova 3171487
Ivan Golovko 3195714
Artemii Neff 3182098
Krzysztof Budzik 3171967

Analysis of Deep Learning Models on Fruits360 Dataset

Deep Learning for Computer Vision 20600, Fall 2022
Final Group Project Report
Link to GitHub Repository

Milan, Italy, December 2022

Table of Contents

Abstract

The purpose of this report is to present the outcomes of the final project for the "Deep Learning for Computer Vision 20600" course at Bocconi University. The goal of the project was to assess the performance of various deep learning algorithms on the Fruits360 Kaggle dataset. We start from simple CNNs built from scratch and compare them to models, obtained through transfer learning from more complex pre-trained models. We conclude that ViT, which is state of the art computer vision model, indeed provides a boost in quality of predictions compared to less recent CNNs.

Link to GitHub Repository: https://github.com/igolovko3/bocconi_cv_final_project

Dataset analysis

Fruits360 dataset contains 90380 100x100 RGB images of fruits and vegetables distributed across 131 classes. The background is removed, and the position of the fruit is always the same -- it's centered and occupies most of the image. However, the “orientation” of the fruit may change, meaning that the same fruit is observed from different angles.

First, we make sure that the dataset is balanced. Overall, as Figure 1 and Figure 2 show, it is: the distribution of class sizes both in train and test data is fairly close to uniform (this justifies using accuracy as the main performance measure for our models). Although there are a few outliers among the categories, none of them is too far from the mean (at most twice larger or twice smaller than the average category). The shares of classes in train and in test data are also consistent: none of the t-tests for differences in means shows significance.

Potential data augmentations

In this section we discuss the potential data augmentations that are applicable to this dataset.

Mirror flips are unlikely to produce completely new samples (but at the same time also will give valid images with the same properties as the original ones). Rotations, on the other hand, are – for example, there are seemingly no "vertical" bananas, and no "diagonal" cactus fruits. However, with rotations we need to be careful. Many of the images in the sample are almost entirely occupied by the fruits, and the borders are very tight. Thus, to avoid undesired cropping of the fruits, further on we only consider rotations, proportional to 90°.

Affine transforms (changing the shape), shifts/crops (spoiling centering) and adding noise (changing the texture) are, probably, undesirable. They may produce the images that are too far not only from the originals, but also from the ones that the model will see in test.

To sum up, data augmentation is unlikely to produce serious improvements – the dataset is rather large (about 500 train samples per class), and the

samples within one class are very homogenous, so overfitting doesn't look like a major threat.

Moreover, according to the creators of the dataset, each class contains only images of one exact exemplar. Thus, in some sense overfitting of the model to certain imperfections and individual traits of the fruits, while being bad for generalizability, will likely be a positive thing for test performance. Still, rotations (especially proportional to 90°) may turn out useful, since they will produce images similar in overall characteristics yet different from the originals.

Baseline model

Setup

To have a baseline for more complex models and to test the impact of data augmentation, we start with a simple CNN with the following architecture (similar to ones that the authors use in the original paper [1]):

- 4 layers of Convolution-3x3 + ReLU + MaxPooling-2x2 layers with increasing numbers of output channels (16-32-64-128)
- FC-1024 + ReLU
- FC-256 + ReLU
- Fully-connected output layer with softmax activation (since we want to have probability distribution over multiple classes)

The model is trained on data rescaled to unit interval. We use cross-entropy loss and train model over 30 epochs. No early stopping is used, since these are “diagnostic” runs, and we might want to know the full learning dynamics.

First, we train a plain model without data augmentations or dropouts. Then, we train same exact architecture, but with the following regularizations and data augmentations:

- Dropout layer before dense layers
- Random rotations, proportional to 90° degrees
- Mirror flips (both horizontal and vertical)

Results

First of all, both models show strong performance both in and out of sample, reaching > 96.5% accuracy on test set (see Figure 3). There are no major differences in the final results (loss seems to be a little bit lower in baseline, but the difference is minor). Both models in some sense fail to escape overfitting – on train accuracy is considerably better than on test. However, the learning curve of the model with dropout and data augmentations seems more smooth and stable. Thus, for more complex models it might be useful to use some form of regularization.

Though we have > 100 classes, the majority of the errors seems to come from a small subset of them (see Figure 4). At this point it's not quite clear how to address it, because there might be several problems. For example:

- “Pear 2” seems to be the most difficult class to predict. In the random sample of errors we see (see Figure 5) two of those being confused with apples – they are indeed difficult to distinguish, so maybe we just need stronger and more complex models
- “Pepper orange”, also a problematic class, is being twice confused with other orange fruits (mandarin and orange cherry). Obviously, color seems to play a role, so another potential regularization could be changing brightness/contrast (or even converting to grayscale) to make models pay more attention to shapes and textures.

Colors augmentation

Based on what we’ve observed in the previous paragraph, we try an even stronger augmentation strategy: we additionally randomly shift the brightness of images and with probability 0.2 convert them to grayscale.

The results turn out to be quite poor. It seems that such strong image distortions are too much to handle for such a simple model. It converges to a worse state both in terms of train and test performance, and learning seems much more unstable.

Moreover, color augmentations seem to be useless in increasing the generality of the model and even make it worse – the gap between train and test accuracy increases (see Figure 6).

Transfer learning

Next, we try using more complex pre-trained models (or parts of them) as feature extractors and train classifiers on top of them. Hereafter we use only dropout as the means of regularization – effect of data augmentations is unclear after what we've seen previously.

VGG-16

We start with the VGG-16 model, which is a deep convolutional network with (originally) 16 trainable layers, of which the first 13 are 3x3 convolutions. We take the output of the last convolutional block (so the same inputs that go into the first fully-connected layer), which produces 4096 hidden features. We add on top our own classifier with the following architecture:

- Dropout
- FC-1024 + ReLU
- FC-256 + ReLU
- Output layer with softmax

The model is trained on rescaled to unit interval data, using ADAM optimized and cross-entropy loss.

The training strategy is as follows: first, we train just the classifier for 15 epochs – the weights of pre-trained convolutional stay the same; then we fine-tune the whole network for 10 more epochs with a smaller learning rate.

This strategy seems quite reasonable: before fine-tuning the model reaches 95.7% on the test (worse than a baseline – could be explained by the smaller number of trainable parameters so far). After fine tuning the model converges to a better state, and test accuracy significantly increases up to 97.2% (see Figure 7).

ViT

Finally, we try utilizing ViT model (a visual transformer trained also on ImageNet), this time just as a feature extractor (transformer itself is too costly to fine-tune). We take 768 features that it produces and train a very simple classifier with the following architecture:

- FC-512 + ReLU + Dropout hidden layer
- FC + Softmax output layer

The model is trained on normalized and resized to 224x224 images (according to what ViT expects to receive as an input) across 7 epochs using cross-entropy loss and ADAM optimizer.

The results are impressive: even with such a small number of trainable parameters (the smallest across all models), in just 7 epochs the model is able to reach > 99% test accuracy. Moreover, the learning curve seems really smooth (even on batch level), which signals that the results are not just random fluctuations (see Figure 8).

Summary

Overall, we try several approaches: simple CNNs trained from scratch with different data augmentations, fine-tuned deep convolutional network and the Visual Transformer model as a feature extractor.

Model	Epochs	Train Accuracy	Test Accuracy
Baseline CNN	30	99.96%	96.6%
Baseline CNN + Dropout + Rotations	30	99.53%	96.56%
Baseline CNN + Dropout + Rotations + Colors	30	97.57%	85.55%
VGG-16 Frozen	15	99.66%	95.69%
VGG-16 Fine-tuned	15 + 10	99.82%	97.16%
ViT + Classifier	7	∼100%	99.06%

Recapping once more the accuracy scores of the tested models, the value proposition of transfer learning with state-of-the art pre-trained models becomes absolutely obvious: training just a very small classifier on top of ViT, while not even the most computationally expensive thing, yields the best results. Moreover, training also the last dense layer in ViT itself, though being a much longer task, yields > 98 % accuracy just after 2 epochs (can be seen in one of the non-final notebooks) – so there's even more space for potential improvement.

References

[1] H. Mures, M. Oltean (2018). Fruit recognition from images using deep learning. *Acta Univ. Sapientiae, Informatica, 10,* 1 (2018) 26–42

[2] Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., & Dosovitskiy, A. (2021). Do vision transformers see like convolutional neural networks?. *Advances in Neural Information Processing Systems*, *34*, 12116-12128.

Kaggle Notebooks:

https://www.kaggle.com/code/shivanir23/fruits-good-bad-vision-transformer-acc-99

https://www.kaggle.com/code/pankul/image-classification-w-vgg16-weights

Appendix

Figure 1. *Distribution of classes in Train data*

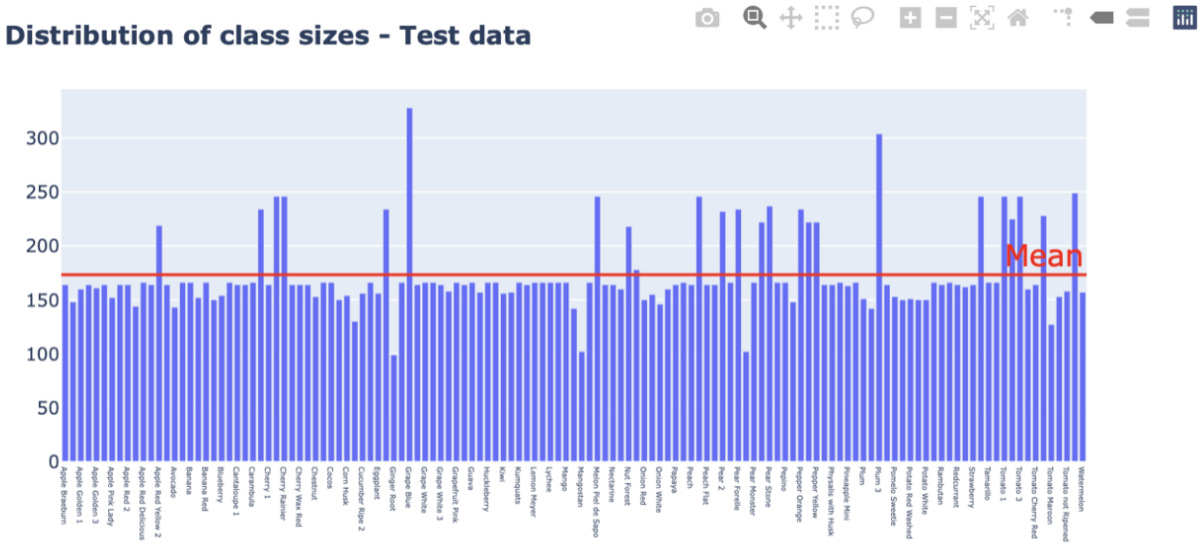

Figure 2. *Distribution of classes in Test data*

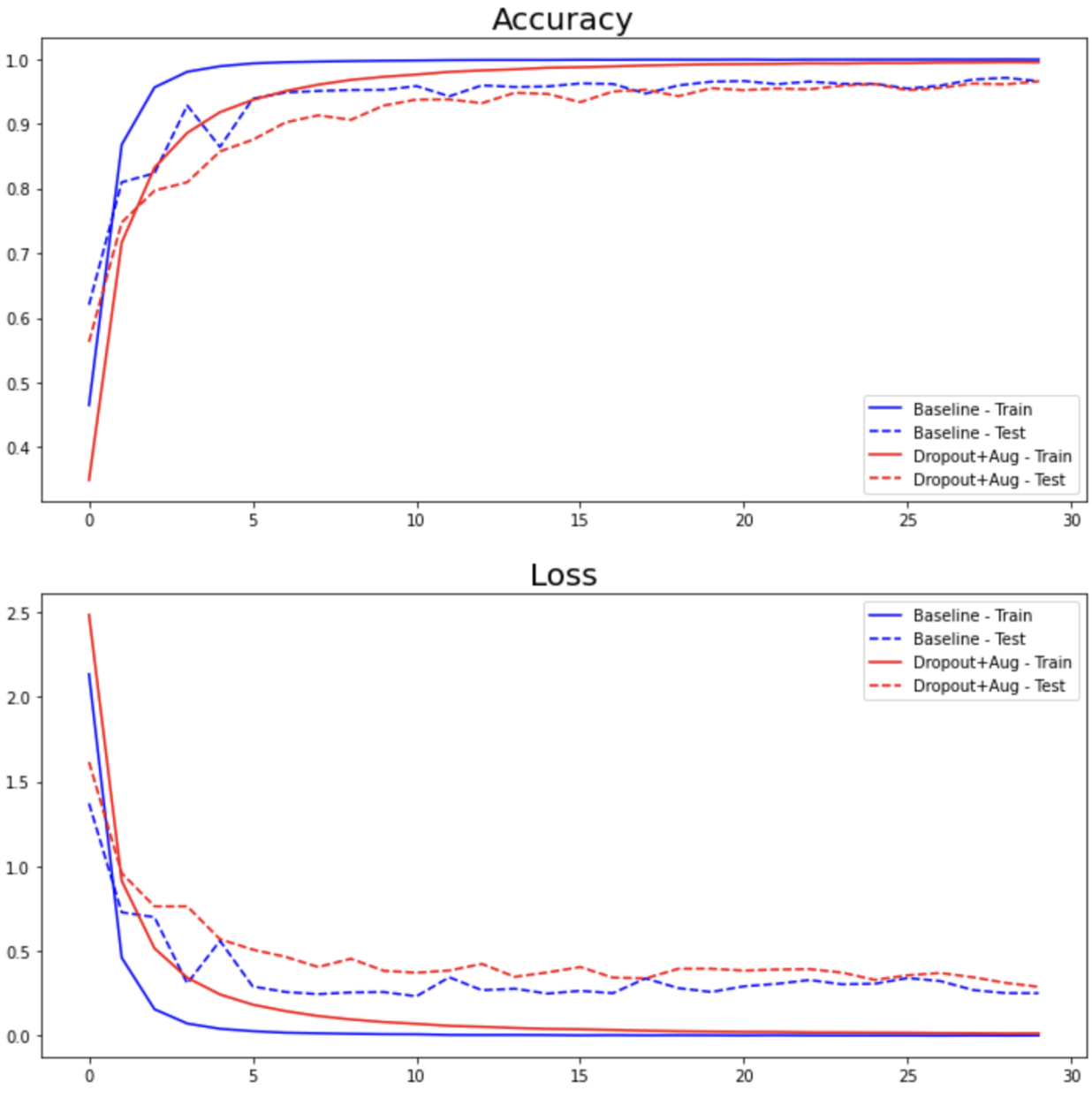

Figure 3. *Accuracy and loss, Baseline VS Dropout + Data Augmentation*

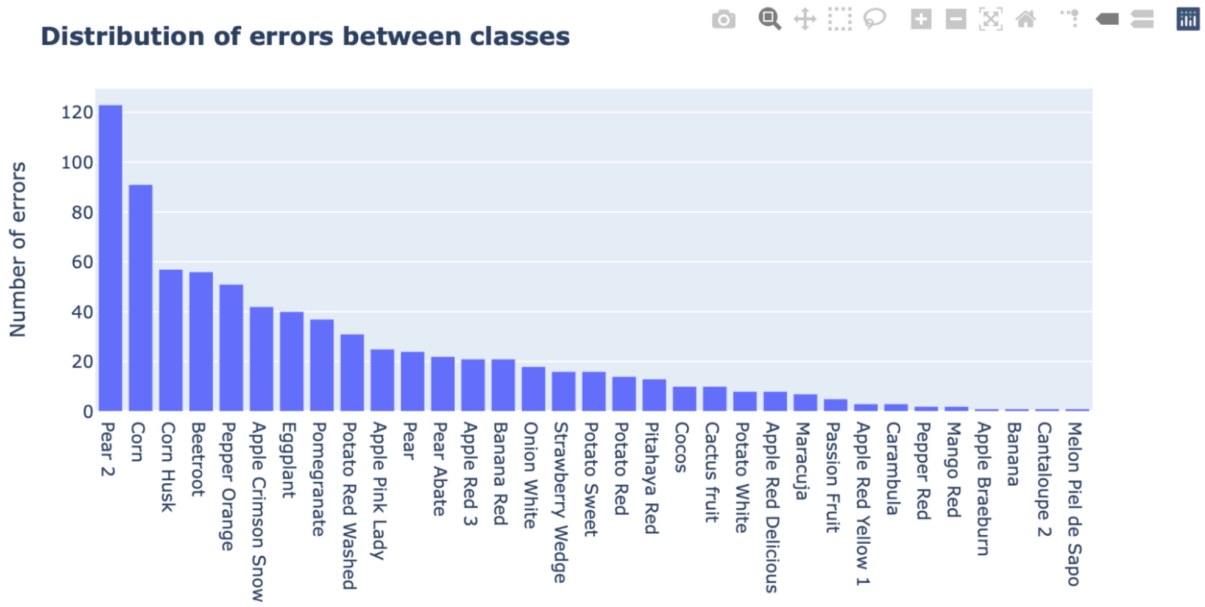

Figure 4. *Distribution of errors, Dropout + Aug model*

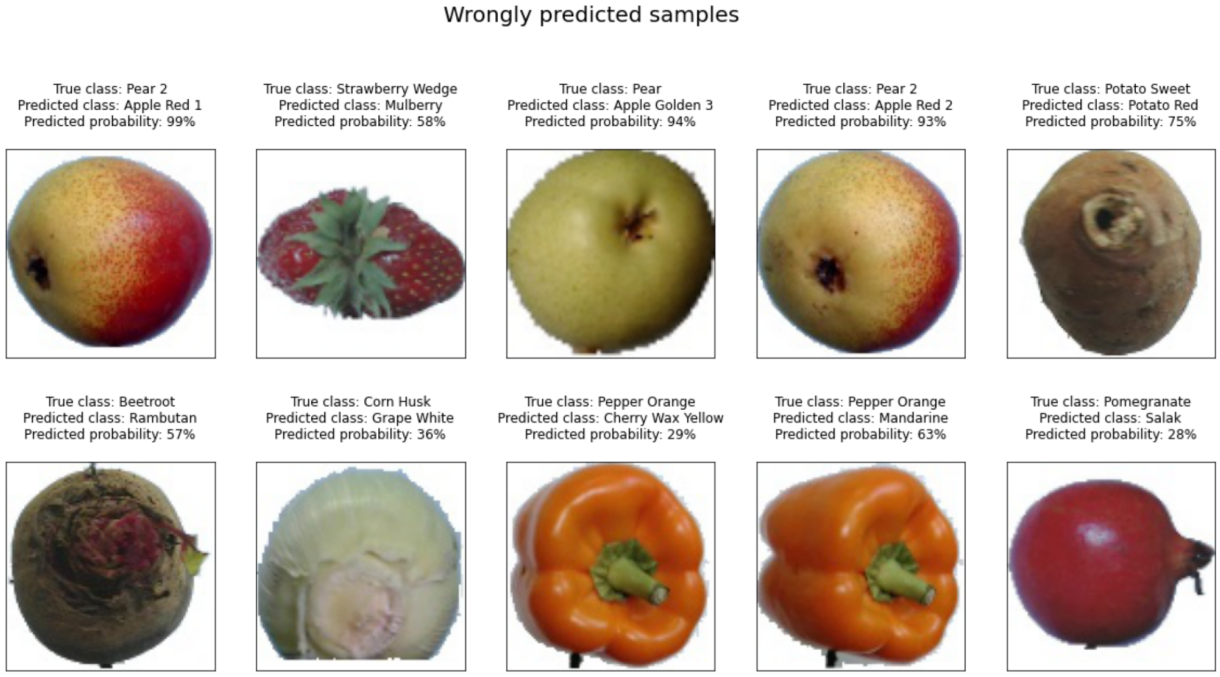

Figure 5. *Some of the wrongly predicted samples*

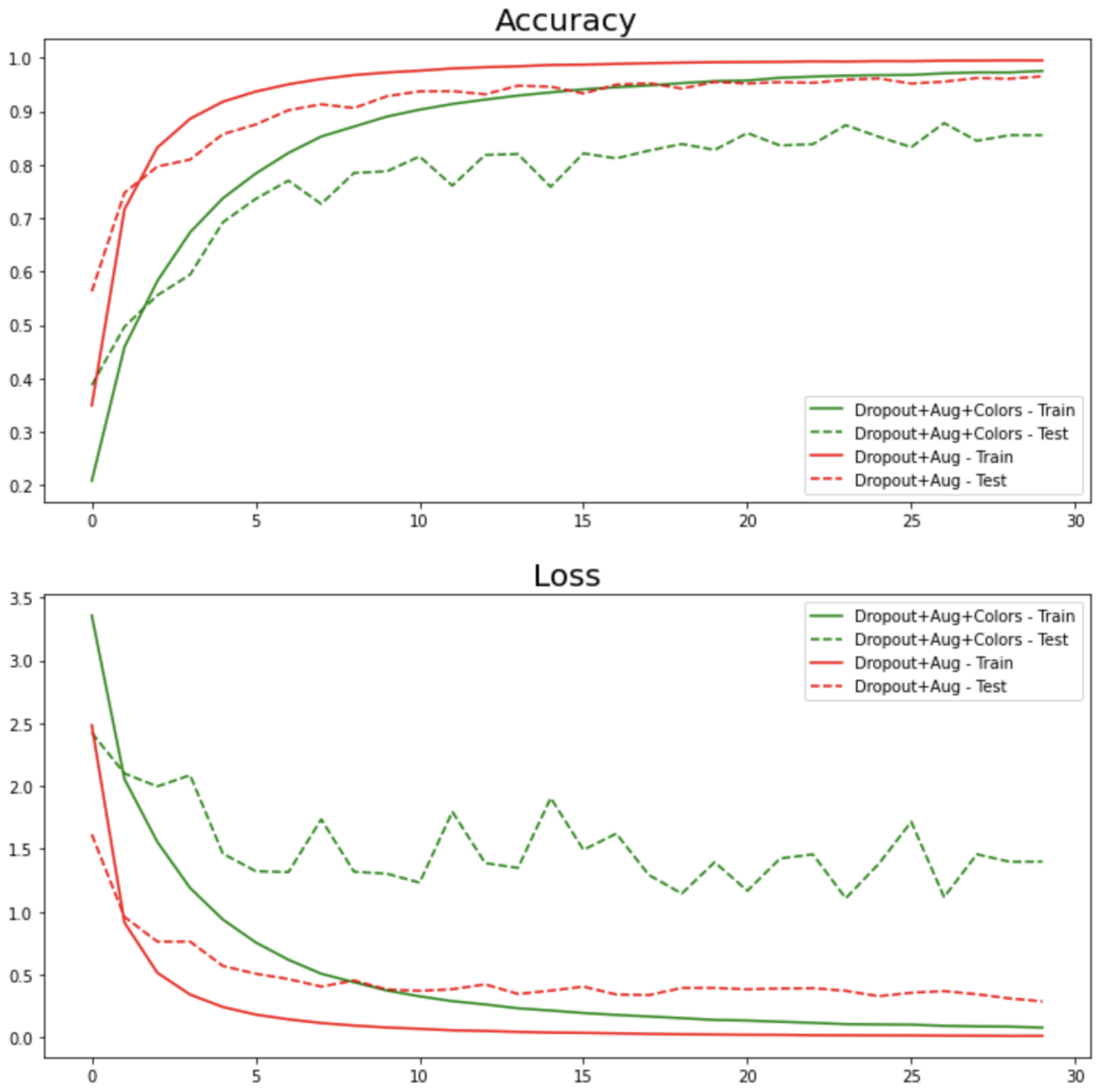

Figure 6. *Baseline with colors augmentation VS only Dropout + Rotations & Flips*

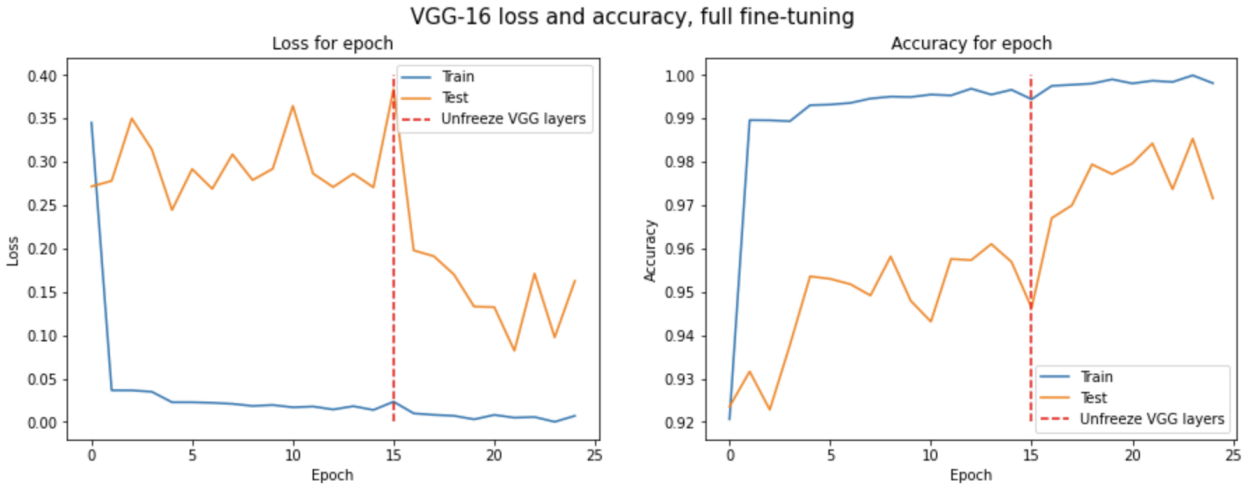

Figure 7. *VGG-16 fine-tuning, loss and accuracy*

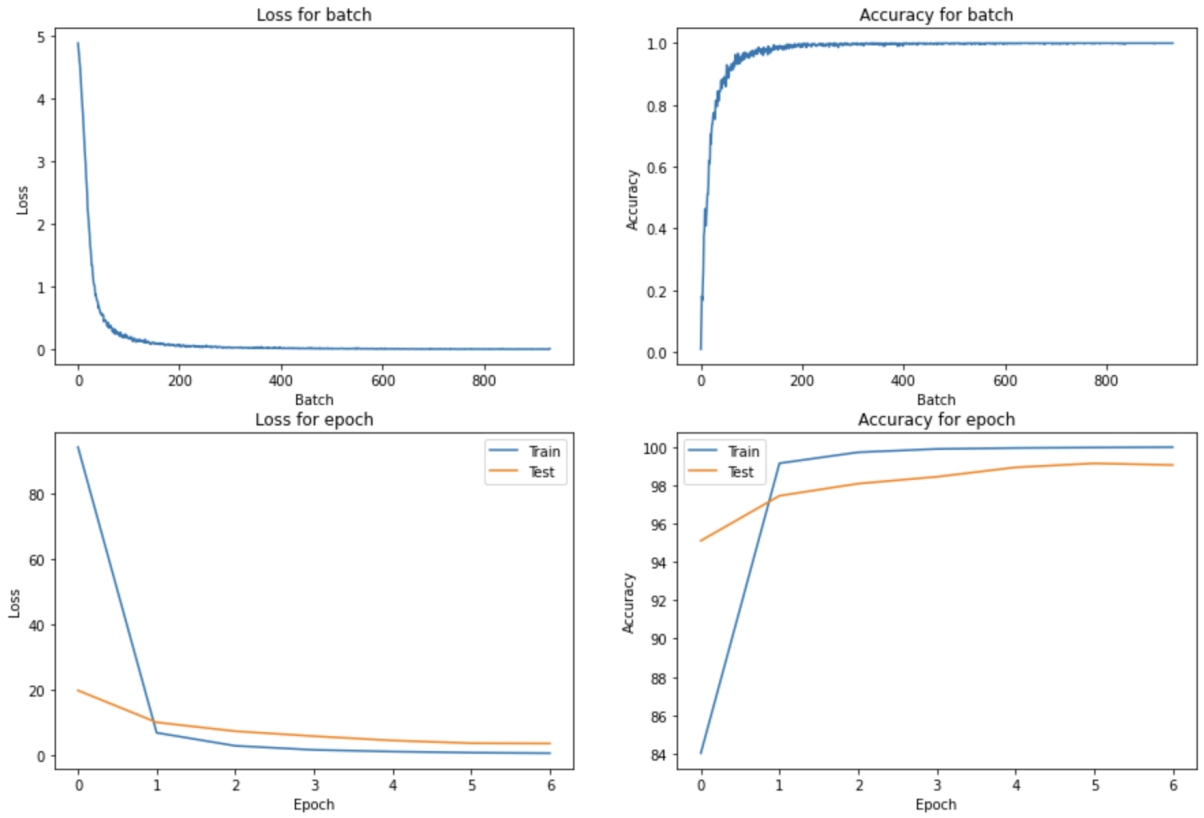

Figure 8. *ViT feature extraction, batch and epoch loss and accuracy*