CLIP: CONNECTING TEXT AND IMAGES (DEEP LEARNING 2021)

PROJECT REPORT

Denis Kuznedelev
Denis.Kuznedelev@skoltech.ru

Denis Rollov
Denis.Rollov@skoltech.ru

Ilya Dubovitskii
Ilya.Dubovitskii@skoltech.ru

Mark Zakharov
Mark.Zakharov@skoltech.ru

Nikolay Goncharov
Nikolay.Goncharov@skoltech.ru

May 28, 2021

ABSTRACT

The common approach for transfer learning on a particular dataset, mostly relies on the use of the network, pretrained on some large dataset, which variety of different classes and diverse data. These networks are believed to be universal feature extractors. However, for this to work one needs a large amount of labeled data (ImageNet, ImageNet-21k, JFT-300). Construction of this dataset is rather an expensive task, since it requires human supervision. Another disadvantage is, that within this approach, one is restricted to the classes, present in the training dataset. Recently, there has be proposed another approach, called CLIP, which is based on collection of the images with the vague word description of this image. Doing so, one has an access to almost unlimited amount of data, and it was shown, that features, trained in this way, are transferable to other domains as well, and in addition, can be use in zero-shot classification. In this work, we would like to compare the performance of the feature extractors on some specific and low-variance dataset between ImageNet and CLIP features. Moreover, we would like to estimate the performance of the CLIP in the zero-shot and few-shot mode.

1 Introduction

1.1 Disadvantages of traditional transfer learning

In the case of rather specific task with relatively small amount of data transfer learning approach has shown to be successful in plethora of applications. Networks, pretrained on large and generic task, like the pronounced ImageNet (1), are universal feature extractors, since this dataset involves a large variety of images, taken in very different settings and with rich diversity of the content. On the other hand, training from scratch on the small dataset is prone to overfitting, since architectures a lot of weights can easily adapt to some spurious patterns in the training data, which are not relevant for the actual problem and demonstrate significant drop in accuracy on the unseen data.

The conventional line of work uses large scale human-labeled datasets. However, there are two major drawbacks. Firstly, collection and labelling of the data is rather expensive. Secondly, one is restricted to the classes, present in the training dataset, since during the training, network had no access to the other classes.

1.2 CLIP approach

Recently, a novel approach for constructing a universal feature extraction has been proposed (2). Motivated by the recent advances in the field of NLP (Natural Language processing), researchers have developed an approach, that relates images and the corresponding captions - CLIP (Contrastive Language–Image Pre-training).

The dataset in constructed as follows - images with the corresponding captions or some kind of text description are collected from the web. Significant advantage of this setting, is that one has access to almost unlimited amount of data, collected from the web, and larger variety of classes and diversity of content, than any human-labeled dataset can potentially have.

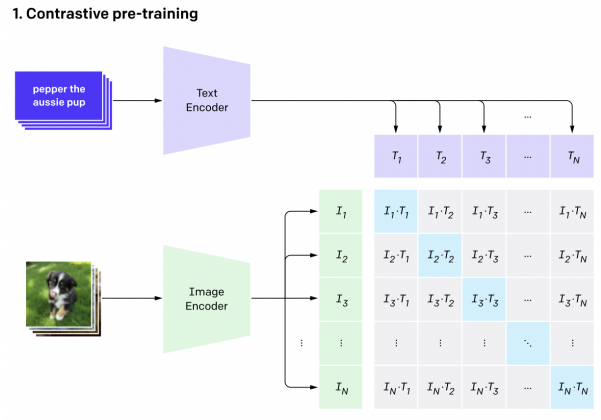


Figure 1: The graphical representation of the CLIP trainning procedure. From (2)

Contrastive Language-Image Pretraining means the following - given a batch of images of size N and the corresponding captions, one passes the image through computer vision architecture - CNN (Convolutional Neural Network) or recently proposed ViT, and the caption embedding is fed through the text transformer. In this way, one obtains N pairs of (image, text) embeddings. The goal of the training procedure is too make the image embedding and the corresponding text description as close, as possible, and make the unrelated images unrelated. Then the training task is transfromed to the classification problem with N classes, where for each image the correct label is the index of caption, corresponding to the given image. Dot products between the normalized image and text features, multiplied by the trainable scaling factor $\exp(T)$ are the logits, which are the measure the confidence of the classifier in the prediction, and the criterion is the standard CE (Cross-Entropy) loss for multiclass classification problem. The whole procedure, can be formulated as follows.

1. Extract image I_f and text T_f features via a forward pass through the visual backbone (text encoder)
$$I_f = V_{\theta_1}(I) \qquad T_f = T_{\theta_2}(T) \tag{1}$$
2. Normalize the features by division by the Euclidean l_2 norm:
$$I_f \to \frac{I_f}{\|I_f\|} \qquad T_f \to \frac{T_f}{\|T_f\|} \tag{2}$$
3. Compute the logits:
$$\text{logits} = \exp(T) I_f T_f^T \tag{3}$$
4. Compute the symmetrized CE loss:
$$\text{labels} = (0\,1\,\ldots\,N-1) \quad \text{loss} = \frac{1}{2}(\text{CE}(\text{labels}, \text{logits}) + \text{CE}(\text{labels}, \text{logits}^T)) \tag{4}$$

Here θ_1 and θ_2 are the visual backbone and the text encoder, which are trained jointly. As in the common classification problem one computes the loss and weights are updated in the backward pass.

There are several potential issues of this method. First of all, batch size needs to be large enough and diverse. Otherwise, since network regards different images as different classes, despite there can be very similar, one is likely to get a poor performance. In addition, labels can be a bit noisy and misleading. However, due to the very large size of the dataset, potential issues with learning on the wrong labels are neglible, most likely.

1.3 CLIP architecture

In the original paper authors have trained several models of different kind and size - 5 configurations of ResNet's and 3 configuration of ViT's (Vision Transformers). The text encoder is the Transformer architecture (3) (4) with masked-self attention.

1.4 The problem setup

In this work we intend to investigate the following directions:

- Compare the quality of the features of the ImageNet pretrained models and the CLIP dataset on a specific problem.
- Perform zero-shot and few-shot classification.
- Estimate the importance of the different upsampling strategies in case the given dataset has a smaller resolution, than the pretrained models take as input.
- Determine hard cases for zero-shot classification.

2 The construction

2.1 Training on the top of CLIP and ImageNet features

In this work we study and compare the efficiency of the conventional approach to finetuning and the proposed by us CLIP inspired, that takes as logits the cosine similarities between the image and text features.

The standard fine-tuning approach.

On the top of the backbone of the visual part of the network (2D CNN - ResNet-101 (5) or ViT (6)) one places the linear layer `nn.Linear(embedding dim, num classes)`, where `embedding dim` is the dimension of the image encoding (512 for ResNet-101 and ViT) and `num classes` is the number of classes in the given dataset.

In case, the task, on which the backbone was generic enough - the weights are rather meaningful before the start of the finetuning procedure, and the model can converge to a good solution rather in a short time.

The question we are going to answer for the particular dataset is - *whether the feature extractor pretrained via the CLIP approach performs better, than trained on the top of ImageNet*. The comprehensive study, that will reliably tell, that one backbone is superior, than the other, requires evaluation on wide range of tasks. Due to the limited resources at out disposal we have checked this statement for only one dataset.

There exist different approaches to fine-tuning

1. Freezing of the whole backbone, and only the top linear layer subject to the training procedure
2. Part of the backbone being trainable or the whole network.

The first approach is cheaper from the computational side and the amount of consumed memory, but the second is more flexible. We have performed experiments for the both cases on the ImageNet and CLIP features.

Cosine similarity inspired training procedure.

In the CLIP paper, when evaluating the pretrained models on the specific datasets, it was figured out, that framing the class labels with the image caption and feeding the resulting sentence through the transformer model gives better quality, than passing solely the name of the class. So, we've decided to follow this approach and chose the three captions, that gave the best-quality in the zero-shot setting on the training dataset.

In order to save computational time, since the caption embeddings are reused each time, one creates a tensor $T_{n,k,c}$ of image caption embeddings with the size:

$$(\text{num classes}, \text{captions per class}, \text{embedding dim})$$

Here the first dimension labels the class, the second the chosen framing of the caption (in our case second index takes values from 0 to 2).

One can account equally for each of the given caption by taking the average of all caption encodings in calculation of the cosine similarity, but we've decided to make the weights w_k in the weighted sum trainable, in order to allow for some captions to be more important, that the other. Then the logits for the class predictions are obtained as follows:

1. Pass the image through the visual part of the network in order to obtain the image embedding I_n, vector of size `embedding dim`.

2. Calculate the weighted sum $\tilde{T}_{n,c}$ of the caption encodings by taking the weighted average of $T_{n,k,c}$ with weights w_k.
3. Normalize the image embeddings I_n and the weighted caption embeddings $\tilde{T}_{n,c}$:

$$I_n \to \frac{I_n}{\|I_n\|} \quad \tilde{T}_{n,c} \to \frac{\tilde{T}_{n,c}}{\sqrt{\sum_c \tilde{T}^2_{n,c}}}$$

   In the latter case each rows of the matrix $\tilde{T}_{n,c}$ is normalized.
4. Calculate the cosine cosine similarity between the input image and the text embeddings for each class and multiply by the scaling factor $\exp(T)$

$$\text{logits}_c = \sum_n I_n \tilde{T}_{n,c} \tag{5}$$

The difference with the CLIP approach is that we use the actual class labels in the `nn.CrossEntropyLoss()` and not the correspondences between the images and the captions.

Advantage of this approach is, that it allows for zero-shot classification with quality significantly better, than the random guess, since during the training of CLIP image and text encoder, text embeddings of the images and captions were forced to be aligned in the same direction. In the fine-tuning procedure image features are trained to be aligned with the feature vector of the given class, obtained by weighted averaging along the embeddings of given captions. This procedure has no such nice interpretation as likelihood maximization, but sounds sensible. And it is interesting to compare the quality between this approach and the standard classifier training.

2.2 Dataset

We have chosen the `Fruits 360` dataset `https://www.kaggle.com/moltean/fruits` as a specific narrow-domain dataset. This dataset contains images of 131 types of fruits. Some of them rather common and are likely to appear rather frequently in the ImageNet and CLIP dataset, such as various apples, oranges and tomatoes. However, there are rare fruits (mangostan, rambutan and others) as well, which are unfamiliar to non-specialists and we would expect, that they would be harder to classify, especially in the zero-shot setting. Another complication would come, that the dataset involves several subspecies of the same fruit, which are difficult to distinguish even for humans.

The training set consists of 67,692 images, and the test set involves 22,688 images. The resolution of all images is 100x100.

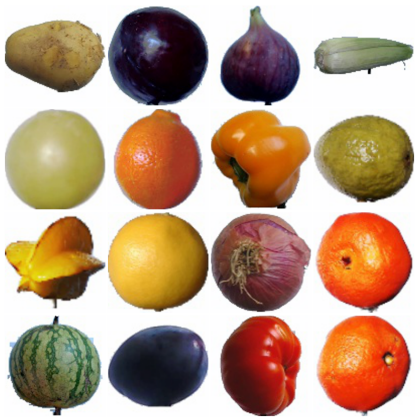


Figure 2: Several samples from the `Fruits 360` dataset

3 Experiments

3.1 Data preparation

First of all, we split the training images into training and validation sets with a 9-to-1 ratio. We fix this division for all the experiments.

Since the resolution of images in our dataset is 100x100, while it is 224x224 for images the CLIP image encoder was trained on, we had to upsample the images to the required size. We decided to do it using resize with a bicubic interpolation, but we also checked whether training on upsampled images produced by SRGAN (7) could result in an improvement. We will report the results in one of the following subsections.

Augmentations applied to training images in all of the models are:

- random horizontal flip with probability 0.5
- random perspective transformation with distortion scale 0.5 and probability 0.5
- random rotation by maximum of 20 degrees
- gaussian blur with kernel size 3 and sigma sampled uniformly from an interval (0.1, 2)

The result of applying these augmentations can be observed here:

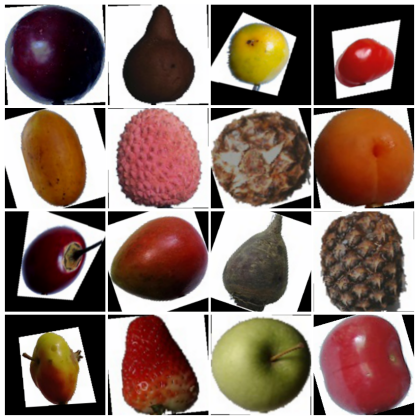


Figure 3: Results of the augmentations application to the `Fruits 360` dataset

Since the dataset is rather balanced (268 samples of the least frequent class and 865 instances of the most frequent with the majority having $400 - 500$ samples) there is no need to apply any technique for handling imbalanced data.

3.2 Optimizer settings

We used batch size of 32 for experiments where backbones were trained, and 48 for linear probing experiments. For heavier backbones (i.e. EfficientNet-B7) we used batch size of 2 and gradient accumulation to keep the number of steps constant for a more fair comparison.

We performed optimization using Adam optimizer (8) with default parameters. The learning rates we used were found by learning rate test (LRT) built into PyTorch-Lightning. The only exceptions were made for linear classifiers trained on EfficientNet's and ResNet's (the one from the CLIP model) features: the learning rates found by the LRT for these models were very high and the models did not learn well. We used learning rate of 8e-3 for them, which is the learning rate found by the LRT for linear probing on top of features obtained from ResNet pretrained on ImageNet.

We were limited in time, so we had to limit the number of epochs: 5 for full-model training and 10 for linear probing models, since they seemed to converge slower. Continuing with the "lack of time" theme, we did one run per model, hence, we do not have estimates of standard deviations for obtained accuracies.

We performed validation two times during each training epoch to have more checkpoints to choose models from. We were validating by accuracy.

We made the experiments reproducible using all the guidelines provided in PyTorch (9) and PyTorch-Lightning documentation.

3.3 SRGAN vs bicubic upsampling

Since the Fruits-360 dataset is constructed from the images of size 100×100 and the ImageNet and CLIP backbones take input of the size 224×224, one needs to transform the given images to the given shape.

In this work we investigate two options:

- Bicubic upsampling.

  The RGB color of the image is determined from evaluation of the polynomials of the 3^{rd} order in coordinates x, y. This approach gives a smoother picture, than simpler bilinear and nearest neighbor upsamplings.

- SRGAN upsampling.

  The more sophisticated and adjustable strategy it to use the GAN (Generative Adversarial Network) for upsampling.

We have used the SRGAN architecture from the paper (7) for the superesolution problem.

At the core of Generator network there are residual blocks with identical layout, two convolutional layers with 3×3 kernels and 64 feature maps followed by batch-normalization layers and ReLU as an activation function. Subsequent upsampling was based on sub-pixel convolution layers.

The Discriminator network contains 8 convolutional layers with and increasing number of 3×3 filter kernels, enlarging by a factor of 2 from 64 to 512 kernels. The resulting 512 feature maps are fed to 2 dense layers and a final sigmoid activation function to obtain a probability for sample classification.

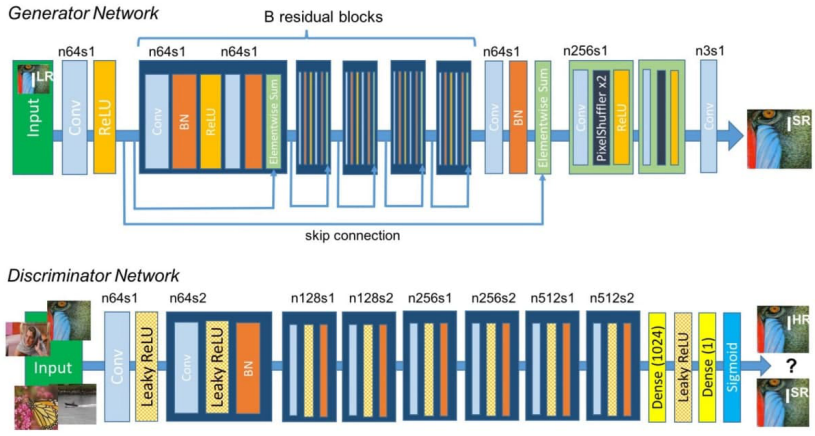
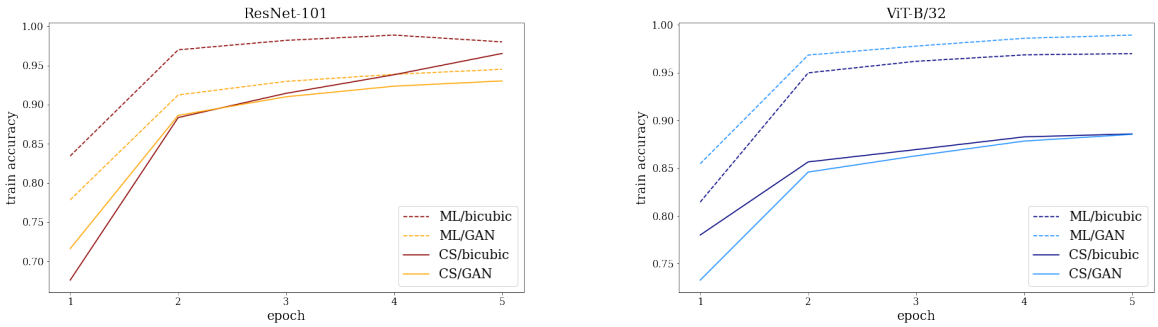


Figure 4: Architecture of Generator and Discriminator Network of SRGAN. Taken from the original paper (7)

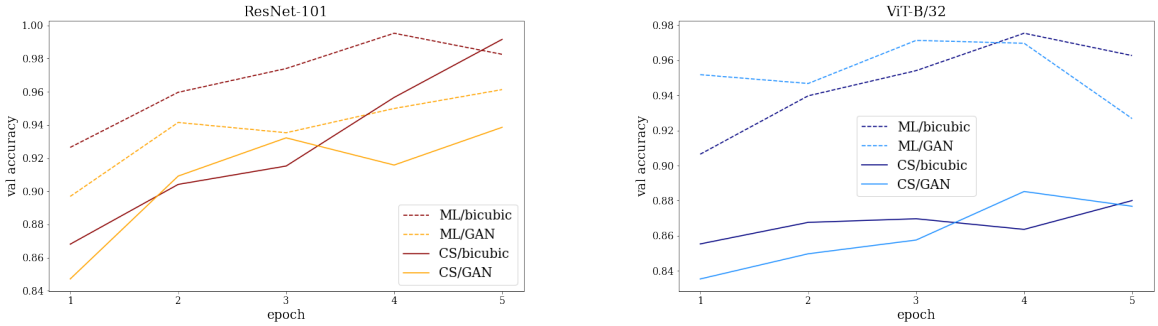
In what follows we demonstrate the resulting quality on the training and validation data for ResNet-101 and ViT-B/32 backbones trained via the standard approach for fine-tuning and cosine similarity inspired. Since the first approach is based on the maximal-likelihood maximization we will denote it in the legend by ML and the second one - CS (cosine similarity). All models are trained for 5 epochs in both cases.



(a) (b)

Figure 5: Training accuracy for the ResNet-101 (a) and ViT-B/32 (b). ML stands for the classical maximal likelihood based classifier training, and CS for the cosine similarity. Red lines are for bicubic upsampling, whereas orange for the GAN upsampling.

Interesting thing to note is that for the ResNet-101 CS performs only a bit worse, than the ML based optimization of the classifier, whereas for the ViT-B/32 drop in the accuracy is rather significant. Second model is more modern and involves the dot products in the self-attention, therefore, one could expect, that CS would work better, than for ResNet-101. The result obtained is rather unexpected and we cannot account for it.



(a) (b)

Figure 6: Validation accuracy for the ResNet-101 (a) and ViT-B/32 (b). ML stands for the classical maximal likelihood based classifier training, and CS for the cosine similarity. Red lines are for bicubic upsampling, whereas orange for the GAN upsampling.

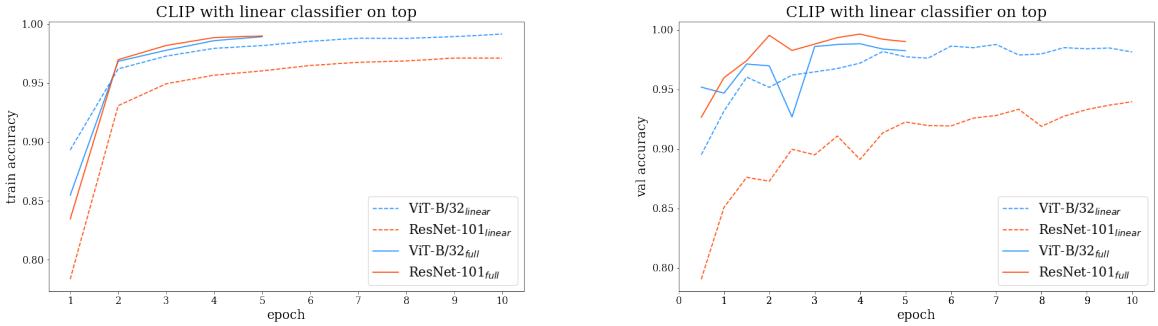
Concerning the comparison of SRGAN and bicubic upsampling approach from the results of the experiments one is unable to detect whether any of the approaches if preferable over the another. In the most cases, bicubic upsampling was slightly better, however, the statistical uncertainties of the results seem to be rather high and change of the random seed may affect the results drastically. In order to make fair and exhaustive comparison one needs to perform runs on multiple seeds and the obtain the reliable estimates of the mean and uncertainty of the classification accuracy. Our estimates from a single run are presented on the Fig 5, Fig 6 and in the Table 3.3.

Model	SRGAN	bicubic upsampling
ViT-B/32 (linear classifier)	0.926	0.973
ViT-B/32 (contrastive)	0.866	0.863
RN-101 (linear classifier)	0.969	0.941
RN-101 (contrastive)	0.931	0.978

Table 1: Test accuracy of the models trained on the CLIP backbone.

3.4 Fine-tuning with the frozen and trainable backbone

The next question, we wanted to compare was the efficiency of fine-tuning on the given dataset with the frozen backbone and only the top layer subject to training and with the trainable backbone. In what follows we would call the first approach *linear* for short and the second - *full*. We have trained the model for 10 epochs in the first case, and 5 epochs for the latter in order for the model to be close to convergence. In the second case convergence was faster in terms of the number of training epochs because on each step more weights are updated at once.



(a) (b)

Figure 7: Training (a) and validation (b) accuracy for fine-tuning with linear classifier. The dashed lines denote only the top layer trainable, and the solid mean the whole backbone trainable.

As one can deduce from the plots, the fully trainable backbone achieves better quality on the training and validation data. For all cases safe ResNet-101 with the trainable backbone there is almost no overfitting. For the *linear* case, there are not a lot of parameters to be adjusted, and the augmentation procedure, presumably, was rather beneficial as well.

Very likely is that optimal decision would be to perform something in between - allow few last layers of the backbone to be modifiable. However, due to the lack of time and computational resources we have not investigated this direction.

3.5 Zero-shot and K-shot classification

Probably, one of the most interesting parts to check is the performance of the CLIP pretrained architectures in the zero-shot and few shot regimes. During the training procedure model was trained to create similar embedding vectors for the image and corresponding captions. CLIP data involves huge variety of the classes and without doubt, a certain amount of fruits was present in the training data. However, some fruits are rather rare and some differ only by subtle details and in order to distinguish them more specialized training procedure is required. Hence, we would expect, that it is unlikely, that model would identify them correctly from the beginning.

In order to estimate efficiency of the zero-shot classification we would need to compare it with the most obvious baseline solution - random guess or constant classifier. The dataset is balanced, therefore these classifiers would give accuracy of order $\frac{1}{num_{classes}} \simeq 0.01$. Therefore, if the zero-shot classifier performs significantly better, than this threshold, CLIP features are beneficial for the problem.

Model	Upsampling Type	Accuracy
ResNet-101	Bicubic	0.2
ResNet-101	GAN	0.181
ViT	Bicubic	0.238
ViT	GAN	0.214

Table 2: Comparison of zero-shot accuracy of models trained on different types of upsampling.

The results of the zero-shot classification are summarized in the Table 3.5. Although the quality of ~ 0.2 may sound poor, it is significantly better, than the accuracy of the random or constant classifier. Moreover, as one can see from the examples of most common mistakes, presented in the Appendix A, many examples are not so easy to distinguish for the non-expert human.

We have performed evaluation for the bicubic upsampling and GAN upsampling in order to check the conjecture - whether the upsampled images would store some latent knowledge of the class depicted and give a hint to the zero-shot classifier. However, as the 3.5 shows, SRGAN's *do not improve the classification accuracy for zero-shot mode*.

The next thing to be studied is the performance in the few-shot mode. The question to be answered is how fast does the network adjust itself to the given training dataset. In order to make the training balanced we have checked the performance of the model on classification task after feeding $2^k, k = 0 \dots 7$ instances of each class and plotted the dependence of the accuracy on the test data in the logarithmic scale.

The salient feature is that the performance deteriorates for the several first samples and then starts to improve after feeding 32 instances of each class to the network. We have performed several runs and the behavior was the same each time. Therefore, this result is a property of the training procedure, not merely a case of accident.

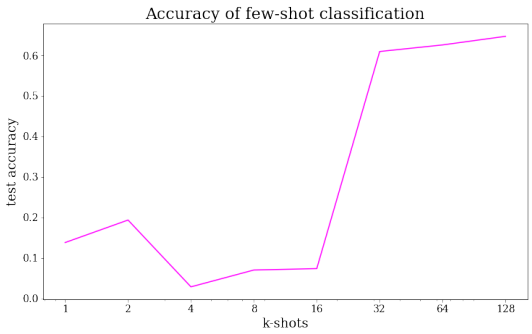


Figure 8: Accuracy of k-shot classification on the pretrained ResNet-101 backbone

Probably, the explanation of this phenomenon is that in the first iteration, the initial relations between images and classes are broken, whereas the weights are not yet fit for the given problem. And then, eventually, the model readjusts to the investigated dataset.

3.6 Comparison of ImageNet and CLIP objective features

One of the main things we wanted to do during the project is to compare the quality of features extracted by the same architectures trained on different objectives, i.e. ImageNet classification and CLIP's objective in our case (ImageNet Resnet and CLIP ResNet hereafter). We also wanted to see how the pretrained models transfer to downstream tasks, i.e. fruit classification in our case.

Furthermore, we wanted to see how a more optimal (in terms of accuracy/Flops ratio) architecture performs against a less optimal one. We hypothesized that the CLIP ResNet would perform better than ImageNet ResNet, so we wanted to see how better architecture would impact the difference in accuracies. For that we chose the largest model from the EfficientNet family, EfficientNet-B7, which has 5.8 GFlops against 7.8 GFlops of ResNet-101. If we could, we would choose a bigger EfficientNet for a fairer comparison, but we had to go with the B7 one. Unfortunately, we did not manage to obtain a useful comparison: EfficientNet scored less than ResNet in both settings. This is likely due to the mentioned difference in Flops.

Model	With Linear Classifier	Fine-Tuned
ResNet-101 (ImageNet)	0.958	0.987
ResNet-101 (CLIP)	0.894	0.941
EffNet-B7 (ImageNet)	0.946	0.982

Table 3: Comparison of models trained on different objectives.

The main result says that features obtained from CLIP ResNet are worse than features from ImageNet ResNet for the task of classifying fruits. We conjecture that this has to do with the following: the ImageNet ResNet's features are specialized for the classification task, i.e. the one we are solving here. This might give it an edge over the CLIP ResNet, since features obtained from it are good for solving a more general task of measuring image-text correspondence.

4 Conclusions

A Common mistakes

One is interested to know not only the accuracy of the predictions - percentage of the correctly classified instances, but where and how is the model under consideration fooled. `Fruits 360` dataset involves subspecies of the same fruit and rather similarly looking fruits like Oranges, Grapefruits, Mandarines. They can be difficult to distinguish even for the human, and one can expect that it would be a struggle for the neural network, especially in zero-shot mode, as well.

We have constructed the confusion matrix on the test dataset and extracted the Top-10 most frequent confusion cases. Then for each such pair we've calculated the IoU (Intersection over Union) for each such pair of classes, namely:

$$\text{IoU}(c_1, c_2) = \frac{m_{c_1 c_2} + m_{c_2 c_1}}{n_{c_1} + n_{c_2}}$$

Here $m_{c_i c_j}$ is the number of times the predicted class was c_i, whereas the true class is c_j, and n_{c_i} is the total number of instances of class c_i. The results for the zero-shot classification with ViT-B/32 backbone are presented in the Table A.

True class	Pred class	Class IoU
Grape Blue	Cherry Wax Black	0.67
Tomato 1	Tomato Cherry Red	0.48
Pepper Yellow	Pepper Orange	0.42
Pear Forelle	Pear Abate	0.46
Grapefruit White	Lemon Meyer	0.50
Lemon	Lemon Meyer	0.50
Grapefruit Pink	Orange	0.50
Apple Red 2	Apple Braeburn	0.49
Mandarine	Orange	0.50
Cherry Wax Yellow	Tomato Yellow	0.51

Table 4: Top-10 most frequently confused classes and corresponding IoU.

In addition, we have visualized the prediction of zero shot model on the Figure 9. One can note, that the common source of mistakes are the subspecies of the same fruits, and the fruits of the similar shape and color.

Figure 9: The visualization of zero-shot predictions

B Performance of the fine-tuned CLIP model

After the fine-tuning the model with pretrained backbone performs rather well on the given dataset and can distinguish even rather similar fruits and identify the subspecies of the given fruit.

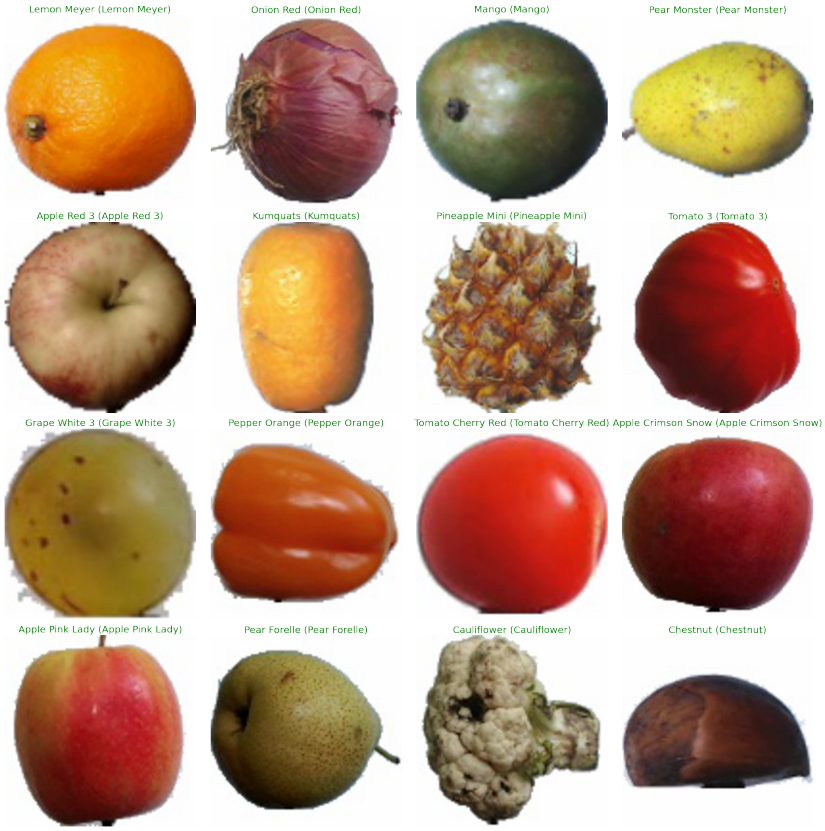


Figure 10: The visualization of the performance of the model with fine-tuned backbone.

C Bilinear vs bicubic upsampling

In addition, we have studied the impact of the choice of non-trainable upsampling procedure for the models, trained on ImageNet. More common choice in this kind of problems is the bilinear or nearest neighbor upsampling. We have taken bilinear and bicubic upsampling and, to our surprise, bicubic upsampling performed noticeably better. The results of the comparison are presented in the Table C.

Model	Bilinear	Bicubic
RN-101 (fine-tuned)	0.978	0.987
RN-101 (with linear classifier)	0.948	0.958
EN-B7 (fine-tuned)	0.970	0.982

Table 5: Comparison of models trained on different objectives.

The results indicate that the bicubic interpolation is noticeably superior to the bilinear one, which was surprising to us.

References

[1] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. `arXiv:1409.0575`.

[2] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. `arXiv:2103.00020`.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. `arXiv:1706.03762`.

[4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019. URL: https://openai.com/blog/better-language-models/.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. arXiv:1512.03385.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. arXiv:2010.11929.

[7] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017. arXiv:1609.04802.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. arXiv:1412.6980.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.