Application of Metric Learning to Large-scale Image Classification Task

Mykola Baranov[1], Yuriy Shcherbyna[1]

[1] *Ivan Franko National University of Lviv, Universytetska St, 1, Lviv, L'vivs'ka oblast, 79000, Ukraine*

Abstract

Deep learning has introduced a lot of successful approaches in a lot of supervised learning areas including computer vision. Modern neural network-based models have proved a human-level performance accuracy. The one limitation that has come along with deep learning is - the requirement of large-scale datasets to train such models. Despite the large-scale datasets like ImageNet or OpenImages, a huge amount of classes are left uncovered. In order to extend the existing model to one more class a lot of data collection and annotation is required. Few-shot learning approaches tackle the issue of large-scale dataset requirements. Most of the few-shot learning approaches tackle the problem of identity recognition (like face recognition etc). Novel object classification remains a challenging task. In this work, we built metric learning based deep learning model based on triplet loss. We explore how the triplet loss-driven model may be applicable to image recognition in a case of a lack of data. Our experiments show that such kind of model leads up to 83% accuracy using only a few samples per class.

Keywords

Few-shot learning, metric learning, distance learning, deep learning, computer vision

1. Introduction

Deep learning models recently have achieved great success in various computer vision tasks like image classification, segmentation, object detection, etc. A lot of progress has been achieved due to increasing model capacity - researchers came up with models like ResNeXt[1] or Inception[2]. Some steps have been performed towards tradeoff between models depth and parameters count - the family of EfficientNet[3] is the best choice when processing speed and accuracy balance is required. It has been proven that using a reach large-scale datasets leads to good results in terms of model key point indicators. It is natural since deep learning models have a trend to generalize data. So, providing a large amount of data leads to better generalization while training the same model on a few samples definitely will lead to overfitting. There are numerous techniques for preventing overfitting (random erase[4], CutOut[5], grid mask[6], drop block[7], and others) but such techniques make it harder to overfit specific features of images by augmentation it, but it is almost useless where a number of training samples are tiny.

The approach of synthetic data generating may be used to generate synthetic data using generative adversarial networks[8]. It is proven to generate realistic images in a controlled environment[9] but it fails to produce completely new images of the given object (for example, side view instead of front view).

Transfer learning is a widely used approach to fine-tune existing pretrained models on a small dataset. In computer vision, it was usually done by cutting off head leyers and replacing them with

[1]COLINS-2022: 6th International Conference on Computational Linguistics and Intelligent Systems, May 12–13, 2022, Gliwice, Poland.
EMAIL: mykola.baranov@lnu.edu.ua (M. Baranov); yuriy.shcherbyna@lnu.edu.ua (Y. Shcherbyna)
ORCID: 0000-0003-1509-2924 (M. Baranov); 0000-0002-4942-2787 (Y. Shcherbyna)

newly created ones. It is proven that fine-tuning only the last layers gives a significant improvement by transferring knowledge of the base model to the tuned model[10].

Described research trend is extensive development rather than intensive in terms of model flexibility. It is impossible to extend the classification model class without

- Model retraining
- Extending a large-scale dataset

Moreover, even satisfaction of the requirements described above does not guarantee a successful model retraining. It is worth mentioning that a lot of business cases of deep learning model applications do not accept manual work like data annotation. Others do not support handling computation resources on their own. A typical example of such a business case - is intelligent cashier-less checkout in the supermarket.

Few-shot learning is a novel approach to deep learning. The main idea of this concept - is to replace classical supervised learning tasks. Instead of forcing the model to generalize the training dataset (or memorize in case of overfitting), we would train the model's ability to learn. It is natural that a baby is able to recognize a car by seeing only several cars in its life. It is done by memorizing several samples of cars and then just performing an intelligent comparison of new objects with existing ones. This is exactly what metric learning does.

A lot of few-shot research works are focused on individual recognition rather than different object recognition (e.g. face recognition is the main focus of few-shot learning since all faces share the same shape but differ in details). In this paper, we move the focus of few-shot object classification from individual recognition to different object recognition. In this work, we explore the capacity of few-shot classification approaches in terms of the replacement of traditional deep learning classification approaches. In summary the contribution of this paper:

1. Build a few-shot 1000-class classification deep learning pipeline
2. Explore strengths and weaknesses of our model on routine object classification
3. Compare results with traditional approaches of image classification

2. Related works

Traditional classification approaches use softmax activation along with cross-entropy loss. That means that each model output of i-th neuron is responsible for the probability of i-th class. Obviously, such models have a constant number of classes that they deal with.

Metric learning suggests as to take another approach - instead of directly predicting the class of input pictures we calculate the distance between picture pairs. In other words, we build a novel metric function that is able to calculate the distance in picture space. Such a function should satisfy the following requirements:

1. Distance between pictures that belong to the same class should be minimal
2. Distance between different pictures should be as high as possible
3. Distance between the same pictures should be zero

Obviously, comparing pictures pixel-wise leads to poor results. Usually, such metrics are constructed by the deep learning model backbone (feature extractor) and some predefined metric (such as L2-distance or cosine distance). In that setup backbone is a trainable part, which learns the embeddings of pictures.

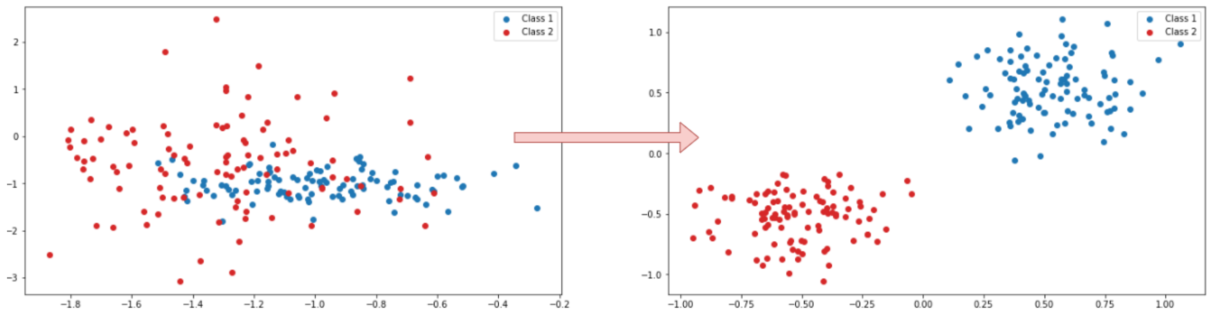

Figure 1: Example of embedding learning

The simple loss that forces the model to learn such embeddings is a contrast loss[11]:

$$(1-Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2$$

where Y corresponds to pair labels. It is equal to 1 if pair of images contains images of different objects. Otherwise, it is equal to 0. In contrast loss formula Y acts as a switcher by tightening two different formulas without an explicit switcher while keeping the formula differentiable. It allows us to use contrast loss in backpropagation. D_w here represents a distance between two images in some arbitrary metric space. In a nutshell, optimizing of contrast loss force model to increase the distance between images of different objects and keep the small distance within the images of the same object. Margin parameters m control a maximum allowed distance between pairs of images. It allows paying attention to pairs that fail to satisfy loss rather than continue optimizing successful cases.

Further study showed that pairwise loss is not the best option. Triplet loss performs much better in a lot of cases[12,13]. The idea of triplet loss is the same as in contrast loss but it operates with three samples

1. Anchor - random sample from the dataset
2. Positive - random sample of the same class as an anchor
3. Negative - random sample of other class

$$\mathcal{L} = max(d(a,p) - d(a,n) + margin, 0)$$

where *d(a, p)* stands for the distance between anchor sample to positive sample, and *d(a, n)* stands for the distance between anchor sample and negative sample. The target of triplet loss - make the distance between different samples higher than the distance between similar samples. It implicitly defines our main target. Margins are used in order to prevent the overfitting of easier samples (actually they play a similar role to the margin of contrast loss).

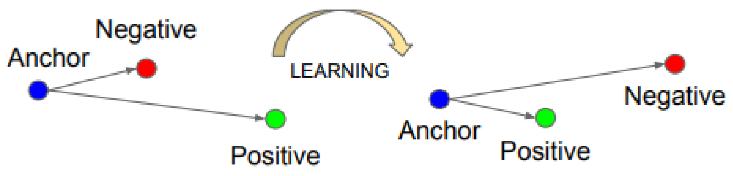

Figure 2: The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between the anchor and a negative of a different identity[6]

However, selection of random triples is not the best option. We will show that it leads to optimisation the model to a local minimum in the next sections. In work[14] several strategies of triplet mining were proposed. Triplet may be classified into three groups:

1. Easy triplet: triplets which have a loss of 0
2. Hard triplets: triplets where the negative is closer to the anchor than the positive
3. Semi-hard triplets: triplets where the negative is not closer to the anchor than the positive, but which still have positive loss

Impact of such strategies will be exposed in the next sections.

Centre loss[12] penalizes the distance between embeddings and their corresponding class centers in the Euclidean space to achieve intra-class compactness. However, it has limited applications since a lot of real-life scene images do not form dense clusters. So centroid of the cluster may light ut of the cluster which will lead to the broken training process. The formula of center loss is provided on figure 3. L_{ct-c} stands for the contrastive-center loss; x_i denotes the training sample embeddings with dimension d. y_i stands for the label of x_i. c_{yi} represents the y-ith class center of deep features with dimension the same dimension. Hyperparameter δ is a constant used for numeric stability.

$$L_{ct-c} = \frac{1}{2}\sum_{i=1}^{m} \frac{\|x_i - c_{y_i}\|_2^2}{(\sum_{j=1,j\neq y_i}^{k} \|x_i - c_j\|_2^2) + \delta}$$

3. Methods and Materials

3.1 Dataset and task definition

There are a lot of techniques of few-shot learning. One of the most popular - using support set. Let's define an N-way-K-Shot classification problem. In that setup, the support set contains samples of N-classes. So, during forward pass such models could classify only 1 class of N. There are a lot of papers and benchmarks tackling that problem.

It is natural that an increasing number of classes *N* will decrease the overall accuracy of our model. So, that approach leads to pure results while working with a huge amount of classes (like MiniImage[13]).

In this work, we are going to build a model that will classify images along with 1000 classes where only 10 images per class are available. Since we have a limited amount of data per class, traditional classification will lead to poor results.

Our work is mainly based on an FSS-1000 dataset[15]. This dataset is specially collected for few-shot learning purposes. A lot of popular datasets (like ImageNet[13], PASCAL VOC[14], ILSVRC[16]) introduce a high bias. It may be a classes balance, semantic balance, items per image balance, etc. The main purpose of the FSS-1000 dataset is to illuminate any bias from data. Thus it contains 1000 classes. Each class represents 10 pictures. All pictures were collected across different search engines (Google, Yahoo, Bing, etc). Moreover, each picture has a constant resolution of 224×224 and only one class instance is presented at once. These properties ensure that our model will have no bias to any specific properties of images. In contrast to that, model trained on ImagenNt may be perfect in dog classification (because that dataset contains more than 200 dog breeds) but fail on car recognition.

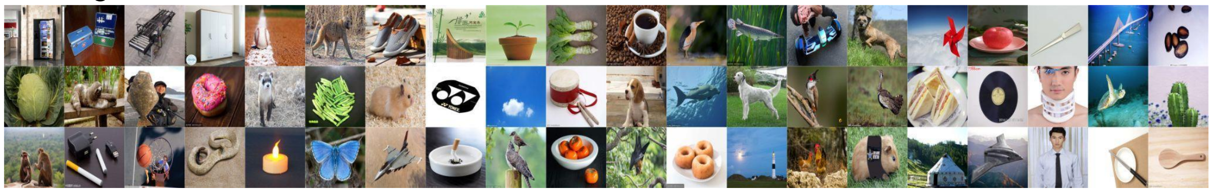

Figure 3. Example of the FSS-1000 dataset samples.

FSS-1000 provides us with instance segmentation masks. Since this work is focused on a classification task we crop each image by its bounding box mask. We calculate the bounding box mask as a minimum and maximum coordinate of the mask both for X and Y axes. Each cropped sample was resized to the standard resolution of $224x224$ in order to match the expected resolution of pretrained backbone models.

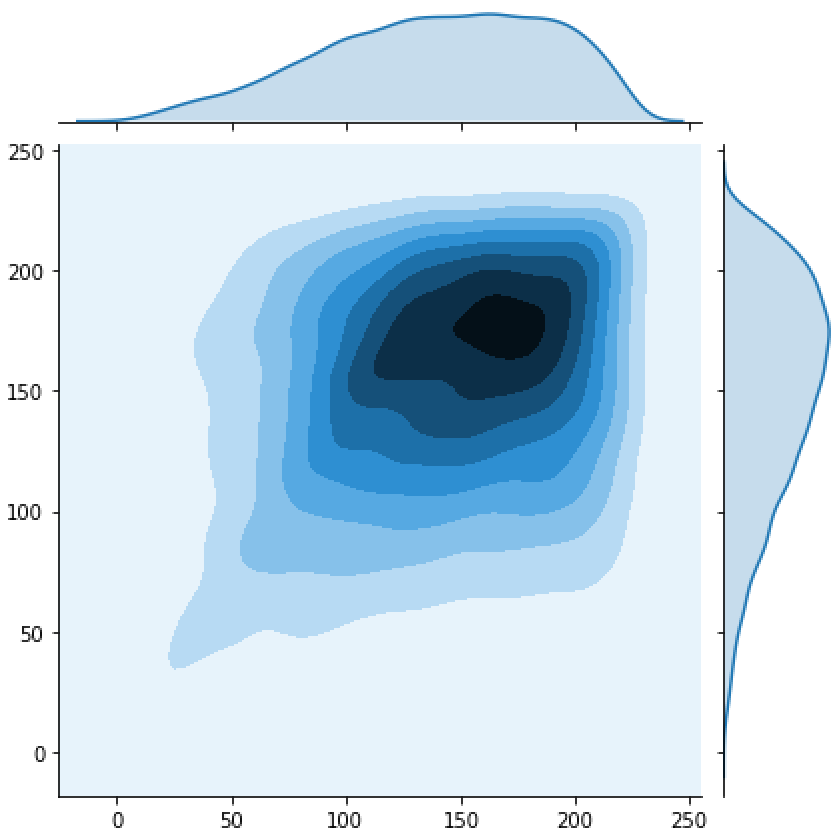

Figure 4. Visualization of cropped objects resolution objects in the FSS-1000 dataset. In that figure, we can observe that the best tradeoff between information loss and compression ratio is 175*x*175. However original size is required in order to prevent information loss.

Resizing of cropped samples provides us with additional distortion. In figure 5 we can observe the level of such distortions. However, according to the original aspect ratio exploration, most of the samples have a 1:1 aspect ratio which illuminates any distortion while resizing to 224*x*224.

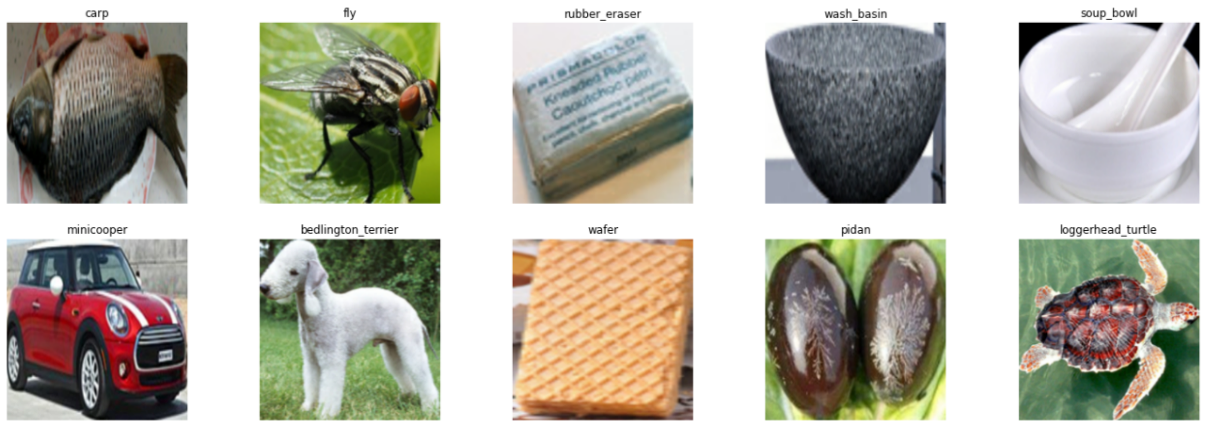

Figure 5. Cropped and resized samples. Here we can observe a few aspect ratio distortions.

3.2 Methodology and setup

In our experiments we propose two deep learning pipelines. One is suitable for binary classification (e.g. decide if two given images are similar or not). The second one - traditional image classification on 1000 classes. Actually, both pipelines are quite similar. The main difference lies in embedding the processor module.

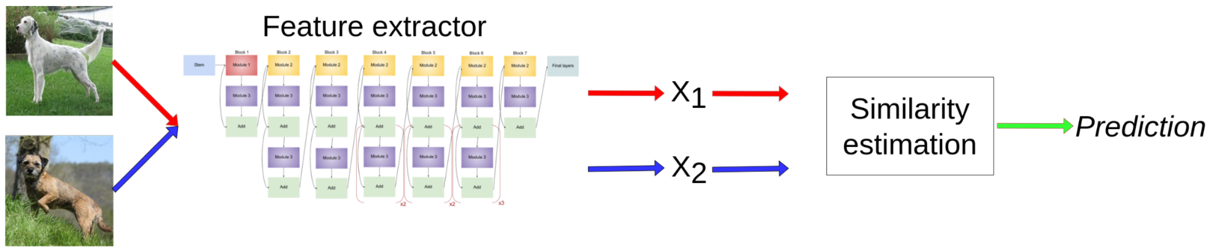

Figure 6. Diagram of proposed pipeline suitable for binary classification. The similarity estimation module compares the distance between embedding with a predefined threshold which has been extracted from the training dataset.

Since the dataset we are working with is quite limited we decide to use a model from the EfficientNet family. In our experiments, we used EfficientNet B2[3] as the main backbone of our model. We took pretrained weights on ImageNet[7] as an initial weight.

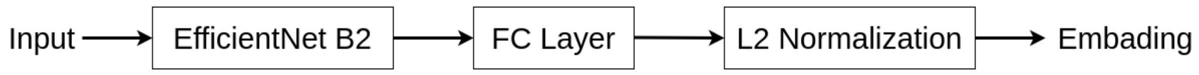

Figure 7. Diagram of embedding extraction module.

A fully connected layer is initialized randomly. L2 normalized is used in order to keep a meaningful scale of extracted embeddings.

For a 1000-class classification task we start from the pipeline defined above. First of all, we cut off a similarity estimation model. It was replaced by a traditional K-Nearest neighbors model. Since the KNN model cannot be trained using gradient descent methods we introduce a 2-stage training pipeline. At the first stage, we train a backbone model using triplet loss. Moreover, we may reuse the backbone extracted trained for the binary classification task. In the second stage, we calculate embedding for each sample in the training dataset. Afterward, extracted embeddings are feed to the KNN model along with class labels. So, the backbone model extracts embedding from the input image. The second KNN predicts class labels based on nearest neighbors.

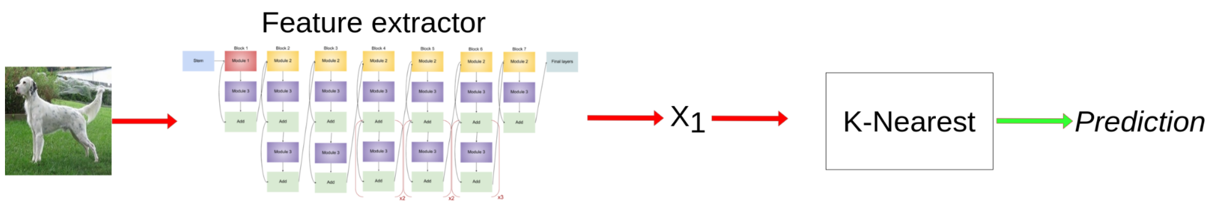

Figure 7. Diagram of the model suitable for a 1000-class classification task.

4. Experiment

4.1 Dataset setup

All training and evaluations have been performed on an FSS-1000 dataset. We split the dataset randomly per class. We take 7 samples of each class for training. The rest part is used for validation purposes. We follow a traditional approach of 80% to 20% dataset split but shuffling has been performed in respect of class balance. As a result, we came up with a perfectly balanced dataset with exactly the same number of samples per class both in training and validation splits.

4.2 Embaing model training

We define use embedding size of 256 (i.e. fully connected layer size is set to 256). In order to keep backbone weights healthy, we start training with frozen backbone weights. After reaching the loss plateau we unfreeze backbone weights and continue training. Such setup helps us to increase final accuracy by ~2%.

Triplet loss is used as the main loss to optimize during training. We use a semi-hard triplet mining policy[16]. We set the margin to 1 for triplet loss and L2-distance is used as a distance between embeddings.

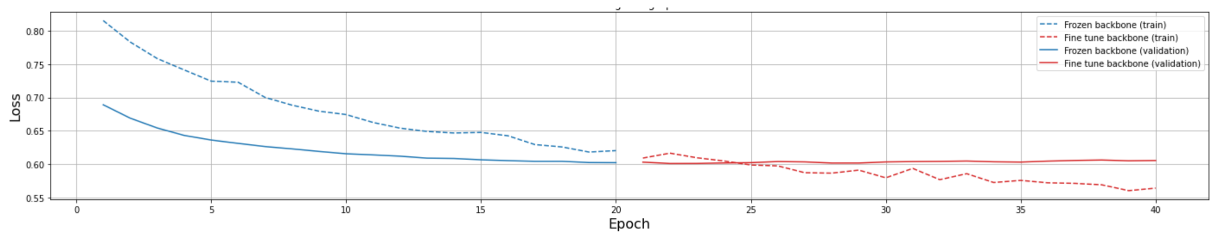

Figure 8. Training loss graph of backbone model

We calculate distances between each pair of samples. The distribution of such distances is presented in Figure 9. We calculate the best suitable threshold for the training set using the Otsu thresholding algorithm. Applying an obtained threshold (0.0048) to the validation test gives us **0.921 true positive rate** and **0.998 true negative rate.**

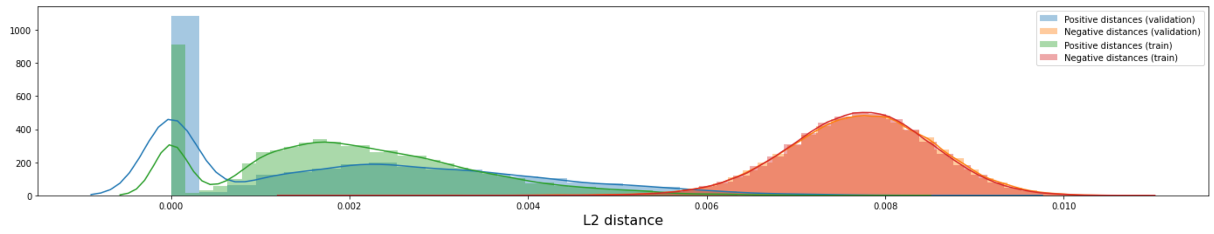

Figure 9. Embedding distance distributions of training and validation sets.

5. Results

We use the KNN model with K=7. That number of the nearest neighbors is chosen according to the number of train samples per class. We fit the KNN model to the training embeddings.

Table 1
Classification scores

Model	Accuracy train	Accuracy validation
Our (top 1)	0.944	**0.836**
Our (top 5)	0.999	**0.957**
EfficientNet B2 + softmax (top 1)	1.0	0.723
EfficientNet B2 + softmax (top 5)	1.0	0.914

It is very important to select a triplets mining policy if using triplet loss. Deep models tend to overfit training data (especially if training on small datasets). Using triplet loss allows us to increase the number of samples exponentially (by incorporating three samples there are a lot of different combinations). Despite a large number of different triplets, there is still a chance to overfit the model. This is caused by the fact that model optimizes all distances between samples despite having very good distances between easy samples. In order to prevent such behavior, a triplet mining policy is implemented.

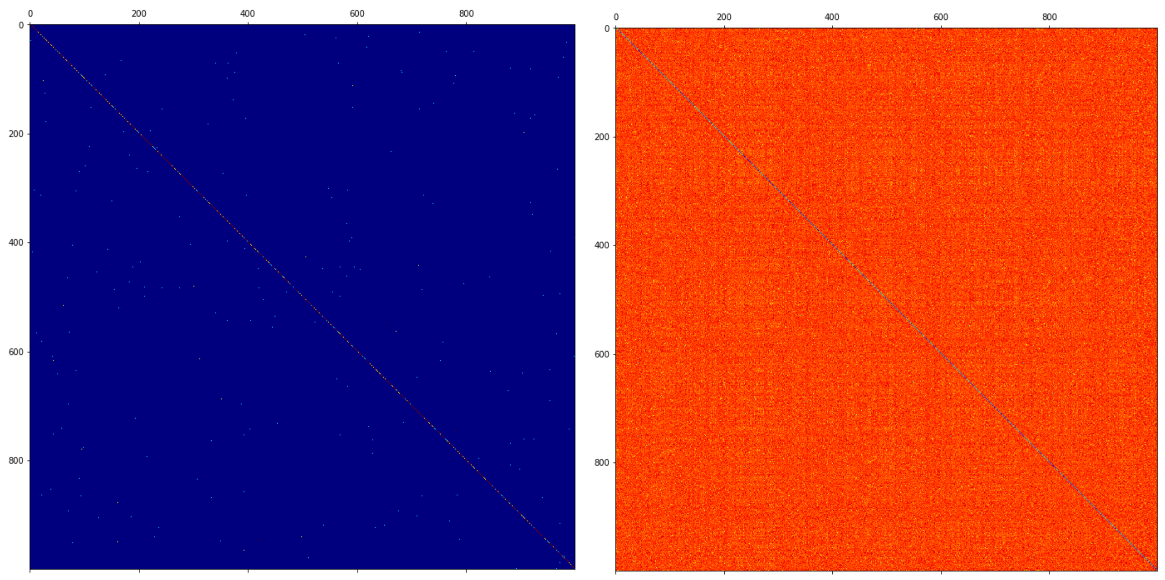
Figure 9. Class confusion matrix (left) and class distance heatmap (right)

Figure 10 presents differences between naive triplet meaning (random) and semi-hard triplet mining.

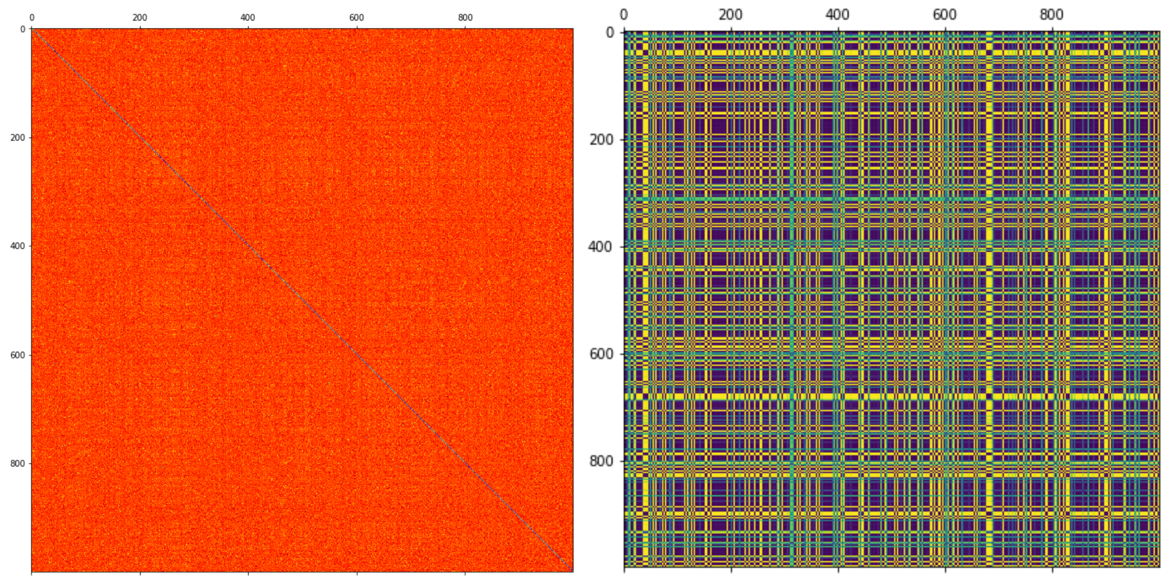
Figure 10. Difference between obtained distances using semi-hard triplet mining policy (left) and random triplet mining (right).

As we can see, random triplet mining tends to generate very good embeddings between some classes, but embeddings of the rest are awful. Such setup may produce nice scores for binary classification tasks, but for 1000-class classification, only 0.122 accuracy was obtained.

6. Discussions

Traditional approaches of deep learning model cames up with a model that is able to classify images end-to-end. Actually, such models implicitly consist of two parts - feature extraction and feature classification. The goal of classification layers is to attribute input features to one of the classes. Part of the feature extraction module - is to produce such features that classification layers will be able to classify. Thus, combining these two modules in one model along with gradient descent optimization methods gives us an end-to-end trainable model. Since those modules are training simultaneously feature extractor tries to produce features that are easy to classify. So, there are no constraints for such embeddings. Thus, the model pays attention to a unique part of the object despite the real importance of such part (for example, the glass may be treated as the most important part of the object to be classified as a car, despite numerous other objects with glasses).

In contrast to that approaches, the same networks explicitly find separable embeddings. By separable here we mean that model is forced to calculate features that are distinctive from other class embeddings but not suitable. In other words, the model doesn't operate with class labels at all.

In Figure 9 we show how well our embeddings are separable. There is a tiny overlap between positive and negative distances which stands for hard negative and hard positive pairs. In this overlap, we are likely to give a wrong prediction on classification but having a distance value we can evaluate a confidence level and refuse to give any prediction or make a bias toward one of the classes (which class actually depends on the task). In Figure 9 we also plot a distribution for training and validation test sets, so there is a good fit for validation. It indicates that there was no overfit. The same conclusion is proven by observing a validation loss following the training loss during the training process.

Having such embeddings we can apply a distance-based classification model to it. We find the K-Nearest neighbor classifier the best candidate for classification. K-Fold on the training set (with K=5) gives us the best parameters for KNN - 7 neighbors and weighted distance comparison. Evaluation of such a model on the validation sets proves a significant performance in comparison with the traditional model. Especially, it gives us more than a 10% of accuracy boost in comparison with EfficientNet B2 followed by classification layers. We also evaluate a top 5 accuracy and got relatively the same boost performance.

We also want to emphasize the gap between training and validation accuracies. The traditional model overfit training data completely while our ending extractor model still avoids overfitting and produces better performance. Note, that we do not consider the K-Nearest neighbor model overfit since it should overfit training data by the definition.

The key difference between the few-shot model and the traditional end-to-end model is e semantic of extracted embeddings. In a triplet loss based model, we can only find a similar embedding in the database while features don't provide any class-specific representation. That's why such a relatively small amount of data is enough to fit the model while the traditional approach leads to overfitting.

7. Conclusions

Classical approaches in deep learning image classifications usually consist of two modules implicitly tightened together - the feature extraction module and embedding classification layers. Training of such models leads to making feature extraction models find class-specific embeddings. Since there are no specific contains for extracted embeddings they are not separable in general.

In this work, we have tackled the issue of the large-scale classification task while dealing with a small amount of data. The proposed two-stage classification model is very promising in terms of class capacity, resistance to overfitting, and ability to fit unseen classes. Especially, it gives us a significant accuracy boost (more than 10% of top 1 accuracy) while keeping off overfitting. Distances between produced embeddings satisfy normal distribution without any outliers except easy positive pairs (very similar images in simple words). It indicates that such a model may be easily extended to predict novel classes just as is. There is enough to precalculate embeddings of unseen classes and such a model is likely to classify unseen images correctly.

The few-shot model seems to be a promising approach for large-scale image classification in terms of a number of internal parameters as well. Our model contains less than 9M of parameters. The smallest model with a similar performance on ImageNet 1000-class classification contains up to 66M parameters[17] which is more than 7 times bigger). So, our approach assures to be much faster than traditional approaches with the same performance but still remains dramatically fewer data per class.

We plan some additional steps for the further research. In particular, we are going to experiment with embedding size, triplet loss margin, etc. We believe that better embeddings may be obtained by ensembling embeddings from different models.

8. References

[1] Xie, Saining, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks." In Proceedings of the IEEE conference on computer vision and pattern recognition (2017): 1492-1500.

[2] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognitio (2015): 1-9.

[3] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." In International conference on machine learning (2019): 6105-6114.

[4] Zhong, Zhun, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. "Random erasing data augmentation." In Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 07 (2020): 13001-13008

[5] DeVries, Terrance, and Graham W. Taylor. "Improved regularization of convolutional neural networks with cutout." arXiv preprint arXiv:1708.04552 (2017).

[6] Chen, Pengguang, Shu Liu, Hengshuang Zhao, and Jiaya Jia. "Gridmask data augmentation." arXiv preprint arXiv:2001.04086 (2020).

[7] Ghiasi, Golnaz, Tsung-Yi Lin, and Quoc V. Le. "Dropblock: A regularization method for convolutional networks." Advances in neural information processing systems 31 (2018).

[8] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. “ImageNet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness” In International Conference on Learning Representations (ICLR), 2019.

[9] Tonioni, Alessio, Eugenio Serra, and Luigi Di Stefano. "A deep learning pipeline for product recognition on store shelves." In 2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS) (2018); 25-31.

[10] Zhuang, Fuzhen, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. "A comprehensive survey on transfer learning." Proceedings of the IEEE 109, no. 1 (2020): 43-76.

[11] Khosla, Prannay, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. "Supervised contrastive learning." Advances in Neural Information Processing Systems 33 (2020): 18661-18673.

[12] Qi, Ce, and Fei Su. "Contrastive-center loss for deep neural networks." In 2017 IEEE international conference on image processing (ICIP) (2017):. 2851-2855.

[13] Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang et al. "Imagenet large scale visual recognition challenge." International journal of computer vision 115, no. 3 (2015): 211-252.

[14] S. Vicente, J. Carreira, L. Agapito and J. Batista, "Reconstructing PASCAL VOC," 2014 IEEE Conference on Computer Vision and Pattern Recognition, (2014): 41-48, doi: 10.1109/CVPR.2014.13.

[15] Li, Xiang, Tianhan Wei, Yau Pun Chen, Yu-Wing Tai, and Chi-Keung Tang. "Fss-1000: A 1000-class dataset for few-shot segmentation." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2869-2878. 2020.

[16] Xuan, Hong, Abby Stylianou, and Robert Pless. "Improved embeddings with easy positive triplet mining." In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (2020): 2474-2482

[17] Xie, Qizhe, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. "Self-training with noisy student improves imagenet classification." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (2020): 10687-10698.