Compatibility analysis of fine-tuned models for fine-grained text-based zero-shot image retrieval

András Schmelczer[1]

[1]Leiden Institute for Advanced Computer Science, Leiden University, the Netherlands

1 Introduction

Retrieving images by describing them with a single English sentence has immense utility for professional and hobby searchers alike. It is even more useful when capable of generalising to novel and/or highly domain-specific classes of images. This fundamental task at the intersection of image and natural language processing has seen substantial progress over the last decades [1–4].

With recent breakthroughs in transfer learning [5], utilising large pretrained models for novel, domain-specific tasks has become commonplace. Hence, the aim of this report is to apply and analyse the quality of pairs of pretrained and subsequently fine-tuned text and image classification models for text-based zero-short image retrieval. Including DistilBERT [6] and classical TF-IDF vectors for encoding texts; and VGG [7], ResNet [8], and Inception V3 [9] for embedding the images.

To make the problem more exciting and challenging, the six derived neural architectures are applied to two datasets both of which has a large number of fine-grained classes. The goal is not only to see the resulting accuracy of different pairings but also to analyse any similarity, alignment and/or structure in the resulting embedding spaces.

2 Data

The same datasets are used as in Reed et al. [3], namely, Caltech-UCSD *birds* 200 [10] and Oxford-102 *flowers* [11]. The former contains nearly 12 thousand photos scraped from Flickr of 200 bird species. As can be seen in Figure 1, each class contains at most 60 images, and is quite balanced. The Oxford flowers dataset is similar in nature but focuses on common species of flowers from around the United Kingdom. It is less balanced (see Figure 2) and also contains slightly fewer, about 8 thousand images. Nonetheless, in practice, *birds* is reported to be more challenging for learning [3, 12].

Next to their class label, each image also comes with 10

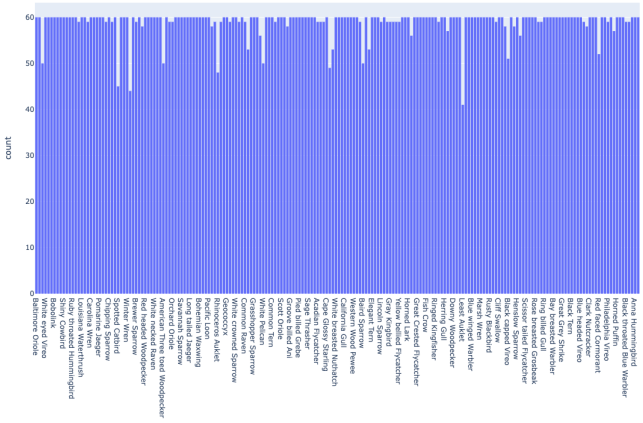

Fig. 1. Distribution of classes (species) in the Caltech-UCSD birds 200 dataset [10].

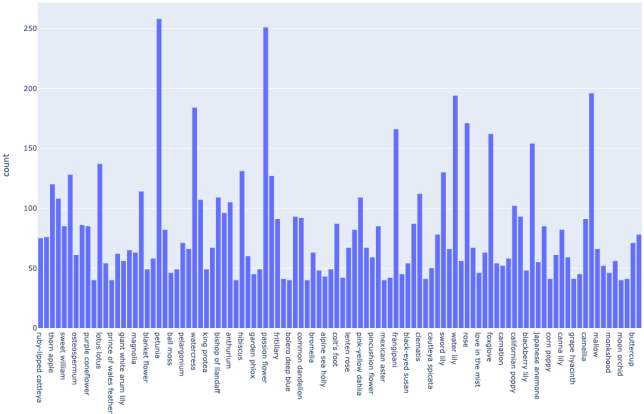

Fig. 2. Distribution of classes (species) in the Oxford-102 flowers dataset [11].

short descriptions crowdsourced by Reed et al. [3]. These only focus on the subject of the photos and are rich in feature descriptions. Thus, overall there are about 100k `(image, caption, species)` datapoints for training and evaluation.

The datasets were selected for their fine-grained nature. For example, discriminating automobiles from trees is less

challenging than telling apart trees of different species. Especially in the case of *birds*, this is further complicated by the dissimilar visual appearance of males and females of the same species. Additionally, both *birds* and *flowers* are a product of nature, resulting in large relative variance even between the instances of the same class. Hence, these can serve as an ideal testbed for investigating how models pretrained on more general classes can be fine-tuned for subtle ones.

3 Background

Six image retrieval systems are developed and compared in this report. They are all slight variations of the same architecture which relies on a text encoder and an image encoder component connected by a linear projection.

3.1 Text-encoder

There are two approaches for representing text in learning applications: sparse or dense vectors. TF-IDF uses the former. In my implementation, the documents (captions) are cleaned, mapped to equivalence-classes, and then packed into a count vector which gets a logarithmic function applied to it component-wise. Inverse document frequency is not taken into account stemming from the clutter-free and highly descriptive nature of the captions. The SMART notation [13] describes this version with the `TXC` acronym.

When it comes to dense vectors, the state-of-the-art are transformers. This architecture was introduced in [14], and a transfer-learning-compatible implementation was created called BERT [15]. There are numerous derivations of this model which either adjust its architecture [6], prune its weights [16], pretrain it on a different dataset [17], or fine-tune it on a domain-specific one. For this comparison, DistilBERT was chosen to represent this family of models because it achieves compelling performance on a wide range of tasks [6].

3.2 Image-encoder

For embedding images, three well-known and historically significant architectures were selected. These are: VGG [7], which showed that dramatically increasing the number of layers may result in superior classification performance. ResNet [8], which introduced another leap in network depth with the help of residual connections, resulting in a more accurate model with up to 152 layers (8 times as many as VGG had). And finally, Inception V3 [9], which was used to demonstrate that better performance can be reached not only with larger models but also with smaller and more sophisticated designs.

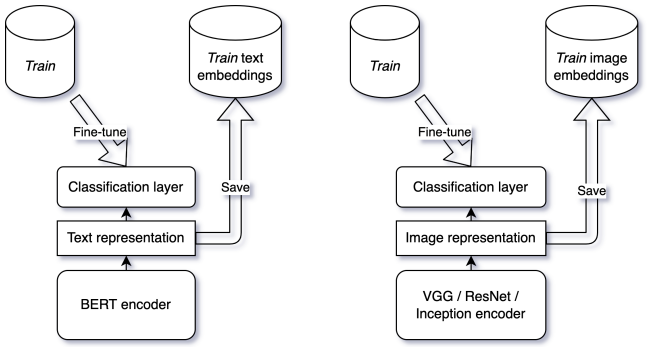

Fig. 3. The first training step is to separately fine-tune the networks via text-classification and image-classification. *Train* denotes a class-disjoint split of either *birds* or *flowers*.

All three of these models had been pretrained on the ImageNet dataset [18].

4 Methods

First, both datasets were split in a class-disjoint way into two sets with a ratio of 7 to 3. The latter is the *test* split while the former was further divided into *train* (80%) and *evaluation* (20%) splits.

For implementing the fine-tuning, training, and evaluation of the models, the following libraries were used: gensim [19], Hugging Face's transformers [20], PyTorch [21], and scikit-learn [22].

4.1 Steps

The main idea of the compared methods is to fine-tune two distinct models on different (but semantically related) tasks and then analyse and utilise the hypothesised similarity in their embedding spaces. Figure 3 shows the fine-tuning stage. Each model (except TF-IDF[1]) has a fully-connected top layer with as many outputs as there are classes in the *train* split. After fine-tuning, the embeddings of the *train* dataset are also calculated and stored for later use.

For the image classification, the learning rate was 1×10^{-3} and constant for all networks, crops and resizing were used for data augmentation. DistilBERT had a learning rate of 2×10^{-5} and a weight decay of 0.01 for regularisation. All networks used minibatches of size 32.

Next, the classification layers can be removed. Their only purpose was to serve as a proxy for learning the discriminative features of the given domain. The saved embedding of the *train* split are fed into a linear regression model (without regularisation) which finds a projection between them. This

[1] In the case of TF-IDF, no fine-tuning or classification task is required. It only needs the captions to create a dictionary and use it for mapping any textual input into a vector.

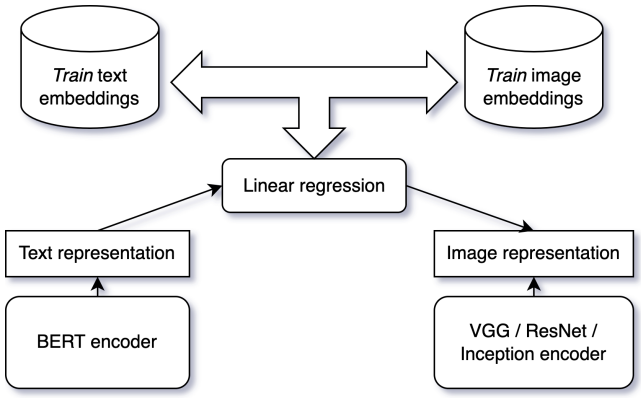

Fig. 4. For the second-stage of training, the last, classification layers are removed from both models. The exposed, raw text embeddings are mapped to the image-embeddings using a linear projection learned from the actual embeddings of the *train* split.

is shown in Figure 4. After this, the image representations (embeddings) can be reached from two directions. Either by using the fine-tuned image encoder, or — more interestingly — by taking the description of the image, getting its embedding using the text encoder and then putting that value through the linear regression. This is the core concept we can use for retrieving images using textual descriptions.

Figure 5 shows the final step, the system's online operation. The vector representations of the *test* images are calculated and stored in a k-d tree for convenient access. When a new query arrives, it is encoded using the DistilBERT model, the encoding is projected, and the *test* image with the most similar (smallest Euclidean distance) actual image embedding is returned.

5 Results & discussion

There are separate scripts and Jupyter notebooks for each step of the experimentation available on GitHub[2]. The experiments were run on Leiden University's DS Lab machines. Considering the number of models involved, it took a relatively[3] short amount of running time to complete: barely over 20 hours on a single GPU.

5.1 Fine-tuning

Fine-tuning worked out as expected, all networks were able to find adequately discriminative features. Figure 6 and 7 shows the *train* and *validation*[4] accuracies of the image-classification task on the *birds* and *flowers* datasets respectively. Classifying bird species turned out to be

[2] github.com/schmelczer/mir-final
[3] Compared with the amount of time required for pre-training.
[4] Which was used to check for overfitting.

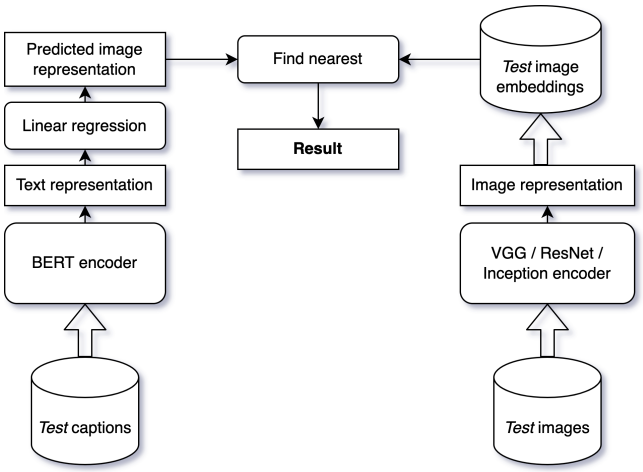

Fig. 5. First, each image from the *test* split is encoded using the fine-tuned and subsequently beheaded image-classifier. For each caption, an image embedding is predicted using the fine-tuned text encoder and a linear projection. Finally, the images with the most similar embedding to the predicted image embedding are returned.

significantly more difficult than flowers. In the case of the latter, nearly perfect accuracy is reached, while the best validation accuracy on *birds* is barely over 80%. In both cases, Inception had the best results, while VGG and ResNet did very similarly.

The evaluation accuracy seems to be trending upwards even after 20 epochs, it is likely that better performance could be achieved with longer training, especially, when combined with tuning the learning rate.

5.2 Results tables

We have seen that the networks perform reasonably well on their own. When combining them, we get the results of Table 1 and 2. The difficulty of *birds* is still clearly visible: the highest Top-1 accuracy on it is achieved by the DistilBERT-VGG pair and is only 19%, while the Top-5 is 43%. This duo is strong on *flowers* as well but is very slightly outperformed by DistilBERT-Inception with 40% and 58% Top-1 and 5 accuracies respectively. Even though DistilBERT-ResNet never took the first place, its performance is stably the 2nd.

Motivated by the unbalance of *flowers*, an F1 score is included as a metric, however, it does not change the ranking of the models. Also, whichever metric we look at, the pairs with TF-IDF nearly always come last, with the low discriminative power of individual tokens and a tiny dictionary of around 7000 combined with the loss of ordering information are hard to compensate for with another network.

Although the results are 10% worse than the best one achieved by Reed et al. [3] in their similar setup, it still re-

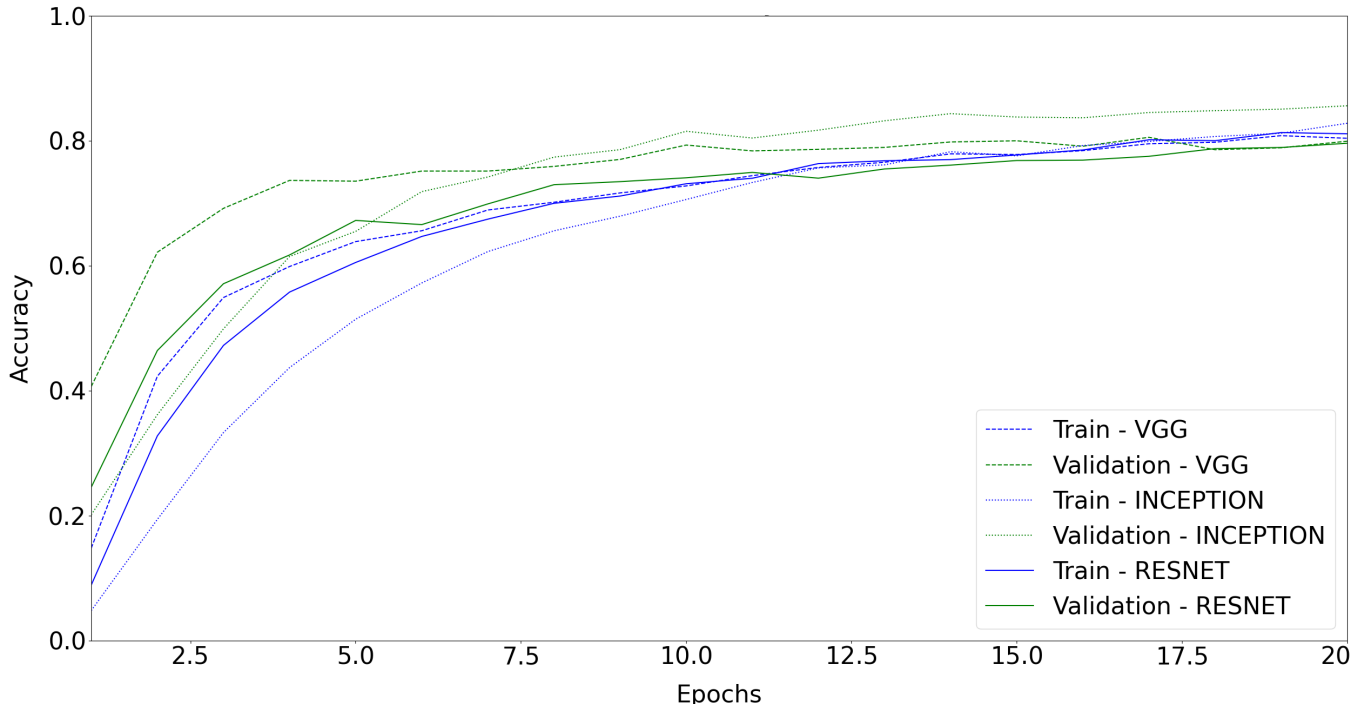

Fig. 6. The accuracy metrics over time of the image classifiers on the *birds* train and validation datasets.

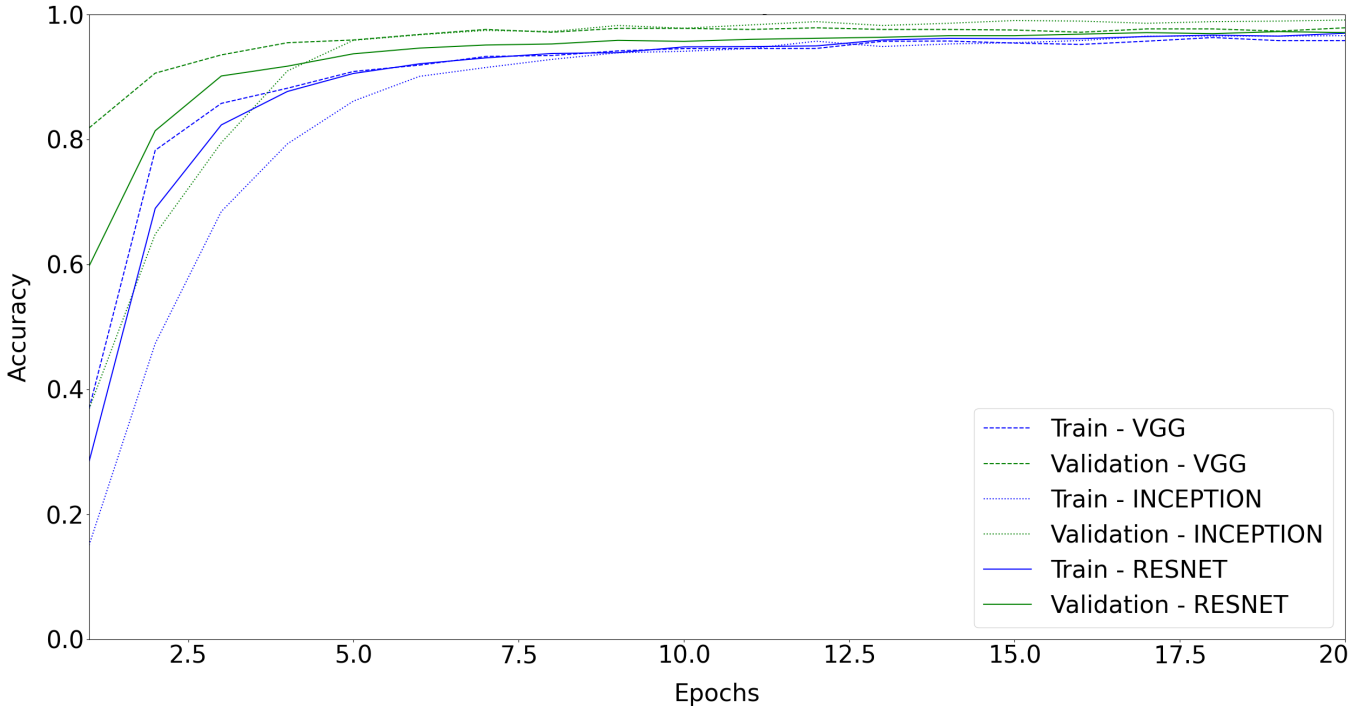

Fig. 7. The accuracy metrics over time of the image classifiers on the *flowers* train and validation datasets.

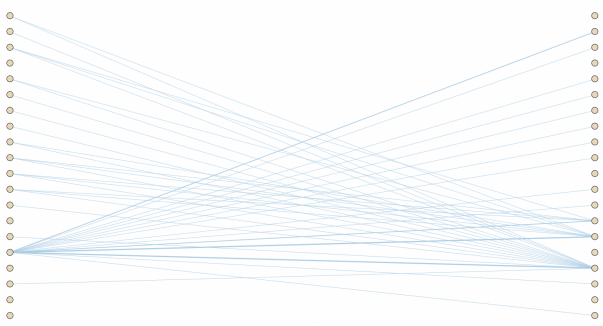

Fig. 8. Weights between DistilBERT and ResNet. The nodes denote the dimensions of text- (left) and image-embeddings (right) which are spatially grouped into a fixed number of bins. The edges correspond to the weights with the highest absolute values between them.

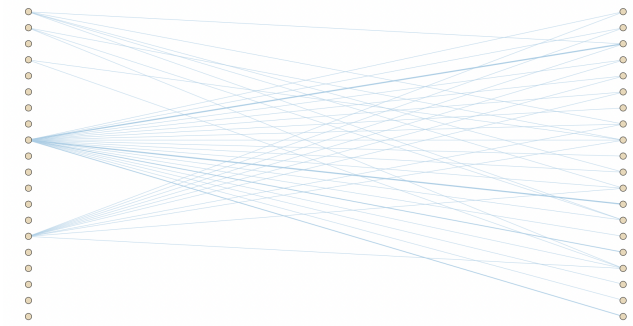

Fig. 9. Weights between TF-IDF and ResNet. The nodes denote the dimensions of text- (left) and image-embeddings (right) which are spatially grouped into a fixed number of bins. The edges correspond to the weights with the highest absolute values between them.

veals an interesting potential of this late-fusion architecture: using off-the-shelf networks for highly domain-specific problems is a viable option. It might not result in the best achievable performance — for that, a custom architecture is likely necessary — but it is a nearly trivial first approach. For example, it could serve as an ideal baseline model.

5.3 Embedding projection

The aforementioned results are considerably better than random. That means there must be some — albeit noisy — transformation between text- and image-embeddings trained using merely semantically similar tasks. Let us investigate them.

Fortunately, with a text embedding size of l_{text} and image embedding size of l_{image} there are only $l_{text} \cdot l_{image}$ weights in the linear projection between them. These are shown in Figure 10. The y-axes show the components of the encoded text while the x-axes correspond to the dimensions of the image embedding space. Note, that the range of the axes vary considerably.

This visualisation provides us with many insights. For instance, a clear divide can be seen between the dense and sparse embeddings: the former show vertical while the latter, horizontal banding. This means that many individual dimensions of the encoded image vector have high correlation with most BERT embedding components, while others do not correlate with any of them. On the other hand, dimensions of the image embedding only correlate with a small number of tokens of TF-IDF. Overall, the TF-IDF models have a sparser mapping to image embeddings. It is also interesting to note, that although BERT-ResNET has the sharpest weight matrix with the highest peaks, this property is not enough to make it the most accurate.

In order to further understand the difference that the choice of text encoder makes, let us look at Figure 8 and 9. The higher quality of the pairs with DistilBERT is not at all surprising when looking at Figure 8: most dimensions highly-correlate with at least one dimension of the other network. However, the hotspots on both sides leads us to believe that some dimensions have noticeably higher discriminative power than others. On the other hand, the components of TF-IDF require a multitude of image vector components to express, hence, the branching out nature of the connections going from left to right. Also, many tokens just do not end up with large weights, which means that these setups are less democratic and that may lead to less generalisability [23].

6 Conclusion

A simple, general architecture with six variations was investigated in the report with the purpose of fine-tuning models for fine-grained highly domain-specific text-based zero-shot image retrieval. The experiments show that there is some linearly expressible similarity between the embedding spaces of text and image classifiers fine-tuned on different but semantically similar tasks. These similarities provide us with insights about the compatibility of different architectures. DistilBERT was found to be of higher utility than TF-IDF in this context. Surprisingly, it would be difficult to name a single image classifier network as the winner. Even the eldest, VGG, have managed to outperform the rest in certain cases.

Even though all six networks are outperformed by the setup of Reed et al. [3], they still perform surprisingly well given their generic late-fusion architecture. The experiments reveal an interesting potential: combining off-the-shelf networks for highly domain-specific problems is a viable solution. It might not result in state-of-the-art performance — for that, a custom architecture is likely necessary — but they could serve as a simple baseline for new problems.

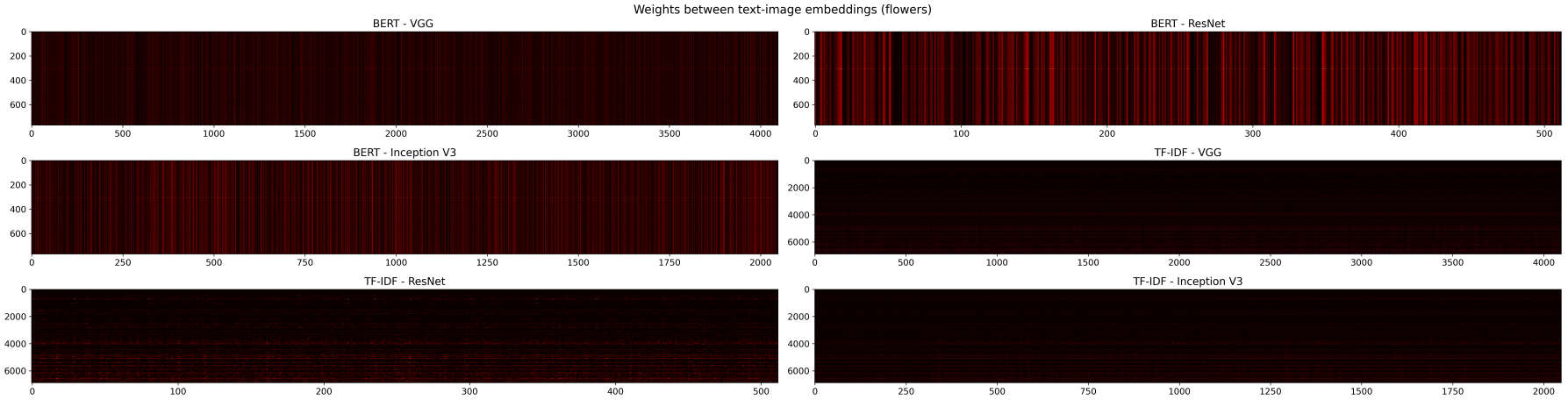

Fig. 10. The absolute value of the weights of the linear layer mapping the text embeddings into predicted image embeddings for each of the six presented models. The y-axis shows the components of the encoded text while the x-axis corresponds to the dimensions of the image embedding space.

Table 1. Comparison of the 6 presented models on the task of zero-shot text-based image retrieval using the Caltech-UCSD *birds* 200 [10] dataset.

	Top-1 Accuracy	Precision	Recall	F1-score	Top-5 Accuracy
DistilBERT - VGG	0.19	0.25	0.19	0.19	0.43
DistilBERT - ResNet	0.11	0.17	0.11	0.10	0.32
DistilBERT - Inception v3	0.12	0.33	0.11	0.11	0.29
TF-IDF - VGG	0.11	0.28	0.11	0.11	0.33
TF-IDF - ResNet	0.10	0.33	0.10	0.09	0.24
TF-IDF - Inception v3	0.10	0.38	0.10	0.09	0.21

Table 2. Comparison of the 6 presented models on the task of zero-shot text-based image retrieval using the Oxford-102 *flowers* [11] dataset.

	Top-1 Accuracy	Precision	Recall	F1-score	Top-5 Accuracy
DistilBERT - VGG	0.37	0.30	0.27	0.23	0.57
DistilBERT - ResNet	0.36	0.31	0.26	0.24	0.53
DistilBERT - Inception v3	0.40	0.29	0.29	0.27	0.58
TF-IDF - VGG	0.20	0.28	0.16	0.13	0.46
TF-IDF - ResNet	0.18	0.28	0.13	0.12	0.33
TF-IDF - Inception v3	0.24	0.24	0.17	0.15	0.43

References

1. V. N. Gudivada and V. V. Raghavan, "Content based image retrieval systems," *Computer*, vol. 28, no. 9, pp. 18–22, 1995.
2. W. Li, L. Duan, D. Xu, and I. W.-H. Tsang, "Text-based image retrieval using progressive multi-instance learning," in *2011 International Conference on Computer Vision*, pp. 2049–2055, IEEE, 2011.
3. S. Reed, Z. Akata, H. Lee, and B. Schiele, "Learning deep representations of fine-grained visual descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 49–58, 2016.
4. A. Latif, A. Rasheed, U. Sajid, J. Ahmed, N. Ali, N. I. Ratyal, B. Zafar, S. H. Dar, M. Sajid, and T. Khalil, "Content-based image retrieval and feature extraction: a comprehensive review," *Mathematical Problems in Engineering*, vol. 2019, 2019.
5. K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.
6. V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.
7. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
8. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
9. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.
10. P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-ucsd birds 200," 2010.

11. M.-E. Nilsback and A. Zisserman, “Automated flower classification over a large number of classes,” in *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.
12. S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, “Generative adversarial text to image synthesis,” in *International conference on machine learning*, pp. 1060–1069, PMLR, 2016.
13. C. Buckley, “Implementation of the smart information retrieval system,” tech. rep., Cornell University, 1985.
14. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
15. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
16. A. de Wynter and D. J. Perry, “Optimal subarchitecture extraction for bert,” *arXiv preprint arXiv:2010.10499*, 2020.
17. I. Beltagy, K. Lo, and A. Cohan, “Scibert: A pretrained language model for scientific text,” *arXiv preprint arXiv:1903.10676*, 2019.
18. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.
19. R. Řehřek, P. Sojka, *et al.*, “Gensim—statistical semantics in python,” *Retrieved from genism. org*, 2011.
20. T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, “Huggingface’s transformers: State-of-the-art natural language processing,” *arXiv preprint arXiv:1910.03771*, 2019.
21. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in neural information processing systems*, vol. 32, 2019.
22. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, “Scikit-learn: Machine learning in python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
23. H. Jégou and A. Zisserman, “Triangulation embedding and democratic aggregation for image search,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3310–3317, 2014.