

OSTBAYERISCHE TECHNISCHE HOCHSCHULE  
AMBERG-WEIDEN

DEEP VISION

# **Comparison of Deep-Image-Embedding Methods**

*Lukas Kreussel*

June 6, 2022

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Materials</b>	<b>3</b>
2.1	Datasets . . . . .	3
<b>3</b>	<b>Methods</b>	<b>5</b>
3.1	Siamese networks . . . . .	5
3.2	Backbones . . . . .	6
3.3	Losses . . . . .	6
3.4	Augmentation techniques . . . . .	7
3.5	t-SNE . . . . .	7
<b>4</b>	<b>Results and Discussion</b>	<b>8</b>
4.1	Backbone performance . . . . .	8
4.2	Significance of Loss Functions . . . . .	9
4.3	Differences in Embedding Sizes . . . . .	10
<b>5</b>	<b>Conclusion</b>	<b>11</b>

# **Chapter 1**

## **Introduction**

Deep image embeddings are a type of feature representation of images that can be used for a variety of tasks. The main advantage of using deep image embeddings over traditional hand-crafted features is that they can be learned directly from data. This means that they can be automatically generated from a large dataset, which can be beneficial when training data is limited.

There are many different ways to generate deep image embeddings, but one of the most popular methods is to use a convolutional neural network (CNN). Another popular method, especially in recent years, is to use vision transformer networks (ViT). These networks are trained to embed an image into a vector representation. Most loss-metrics try to minimize the distance between the embedding vectors of similar images and maximize the distance of different images.

Deep image embeddings can be used for image retrieval, object detection, and image classification. In image retrieval, deep image embeddings can be used to represent images in a low-dimensional space. This makes it possible to search for similar images by using a similarity metric such as Euclidean distance. In object detection, deep image embeddings can be used to represent images at different resolutions. This is helpful because the object of interest may be located at different positions in the image depending on the resolution. In image classification, deep image embeddings can be used to represent images in a high-dimensional space. This makes it possible to use a classifier such as a support vector machine (SVM) to learn a decision boundary that can be used to classify images.

# Chapter 2

## Materials

### 2.1 Datasets

#### Tiny Imagenet

The *Tiny ImageNet* dataset is a subset of the ImageNet dataset. It contains only colored images with 64x64 pixels per image and was created as an alternative to the CIFAR [14] datasets. The dataset contains 200 different classes with a total of 500 images per class.[4] A subset of 50 classes is used in the experiments to test different backbones, loss functions and embedding sizes.

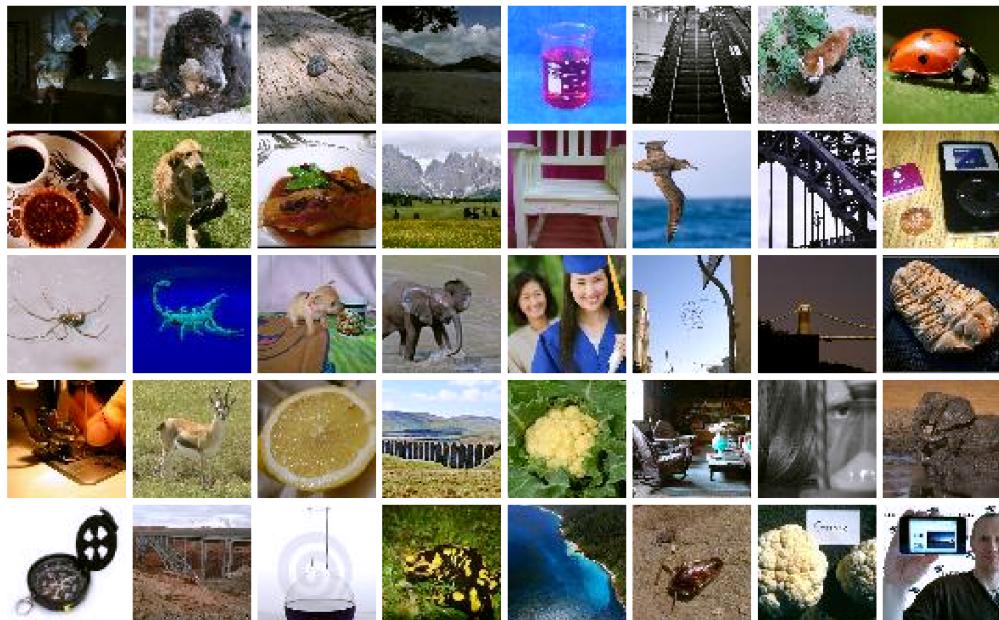


Figure 2.1: Tiny Imagenet

### Internal and External Parts of Cars

The *Internal and External Parts of Cars* dataset [1] is a subset of the CompCars [20] dataset. It contains images of different internal and external car parts. In total there are 27.618 images from 8 different classes. The dataset is used in the experiments to test zero-shot classification performance and a subset of 20 images per class is used to measure the performance on very small datasets.



Figure 2.2: Internal and External Parts of Cars

# Chapter 3

## Methods

### 3.1 Siamese networks

A Siamese neural network is a neural network that consists of two or more identical networks.[2] To generate deep image embeddings from images, these multiple networks are only used in the training step, where the embeddings are calculated for each image and a loss function is used to minimize or maximize the distance between the embeddings, depending on the task. For later inference only one of the networks is used.

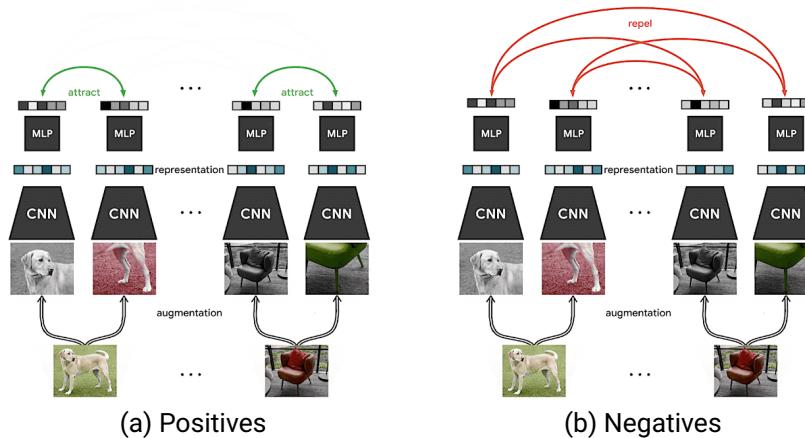


Figure 3.1: Similarity Training in Siamese Networks [19]

Frameworks like SimCLR[3] don't use multiple networks for training, but instead use a single network that processes every image in a mini batch. The resulting embeddings are then sorted by labels and grouped into positives and negatives. Then a loss function is used to minimize distance between the positives and

maximize the distance between the negatives. This approach is a lot faster and uses less memory than using multiple networks.

There is also the *PyTorch Metric Learning*[16] library available for this task, that implements the same approach for generic pytorch models.

## 3.2 Backbones

The following Backbones were used to perform the feature extraction:

- ResNet 50 [9]
- EfficientNetV2 [18]
- MobilNetV3 [11]
- DenseNet [12]
- Vit [7]
- Swin [15]

## 3.3 Losses

The following contrastive loss functions were used to optimize the embedding vectors:

- ContrastiveLoss [8]
- TripletLoss [10]
- SupConLoss [13]
- SNRLoss [21]
- NTXentLoss [3]

## 3.4 Augmentation techniques

The following augmentation techniques were used to augment the training data:

- AutoAugment [5]
- RandAugment [6]
- TrivialAugmentWide [17]

## 3.5 t-SNE

T-distributed stochastic neighbor embedding (t-SNE) is a statistic method for embedding visualization. It is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional data. The algorithm is based on the idea of finding a low-dimensional representation of the data that preserves the local structure of the data.

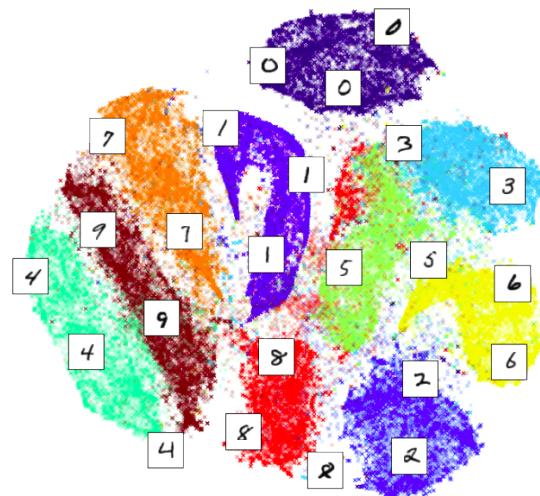


Figure 3.2: t-SNE visualization of MNIST

# Chapter 4

## Results and Discussion

### 4.1 Backbone performance

In this experiment different backbones were used to train a siamese network for five epochs on a subset of 50 classes of the Tiny ImageNet dataset. No augmentations were used. Each network had a target embedding size of 256, used the TripletLoss metric and was initialized with pretrained ImageNet weights.

#### Results

Although ViT and Swin are a lot slower and need more memory than the other backbones they perform a lot better. This property can also be observed in NLP tasks, where transformers originated from. Small models like MobilNetV3 and EfficientNet seem to struggle to generate good embeddings for all classes. They would probably perform a lot better if the embedding tasks contains less classes. The ResNet50 baseline generates good embeddings and even outperforms the larger DenseNet169 backbone.

Backbone	F1-Score
ResNet50	0.664
EfficientNetV2 L	0.540
MobilNetV3	0.367
DenseNet169	0.612
ViT	0.893
Swin	0.934

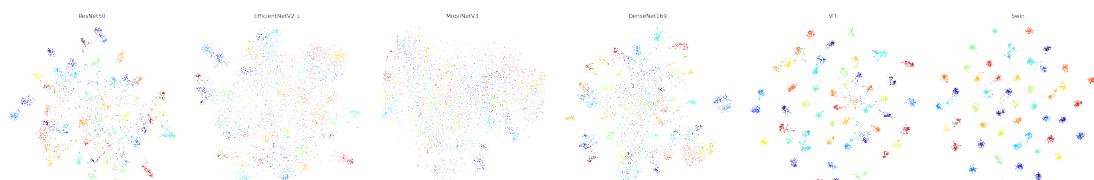


Figure 4.1: t-SNE visualization of backbones

## 4.2 Significance of Loss Functions

In this experiment different loss functions were used to train a siamese network for five epochs on a subset of 50 classes of the Tiny ImageNet dataset. No augmentations were used. As a backbone ResNet50 was used. The network had a target embedding size of 256 and was initialized with pretrained ImageNet weights.

### Results

The SupConLoss function is the best performing loss function. It is also the only loss function that expects the data to be labeled and is optimized for supervised learning. The other loss functions are more general and can also be used for unsupervised learning. We trained the network with labeled data, so its expected that the SupConLoss function performs better than the other loss functions.

Loss	F1-Score
ContrastiveLoss	0.650
TripletLoss	0.660
SupConLoss	0.709
SNRLoss	0.685
NTXentLoss	0.618

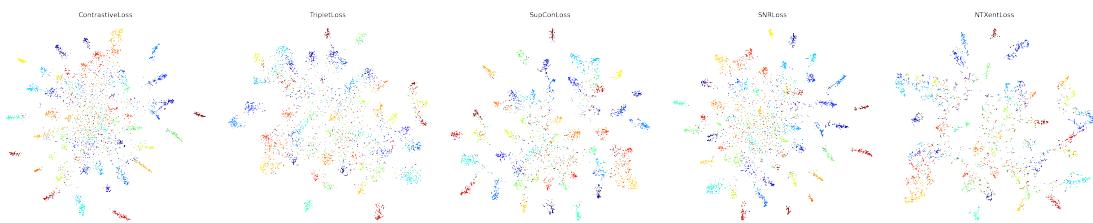


Figure 4.2: t-SNE visualization of loss functions

## 4.3 Differences in Embedding Sizes

In this experiment different embedding sizes were used to train a siamese network for five epochs on a subset of 50 classes of the Tiny ImageNet dataset. No augmentations were used. As a backbone ResNet50 was used. The network used the SupConLoss metric and was initialized with pretrained ImageNet weights.

### Results

The performance of the network get better with higher embedding sizes. This effect seams to plateau at an embedding sizes of 1024. With bigger embedding sizes the parameters of the model increase exponentially and we need a lot more calculations to perform KNN-Searches on the generated embeddings. So its not possible to scale the embedding size indefinitely.

Embedding Size	F1-Score
64	0.654
128	0.683
256	0.712
512	0.719
1024	0.724
2048	0.731

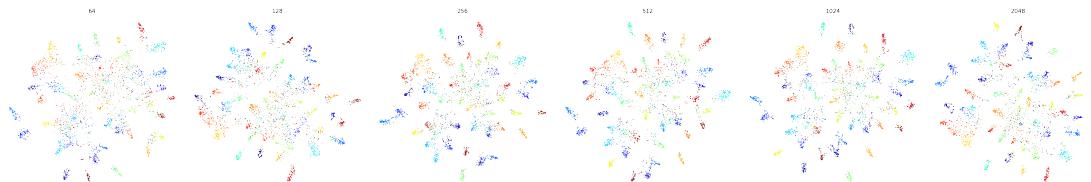


Figure 4.3: t-SNE visualization of embedding sizes

# **Chapter 5**

## **Conclusion**

# Bibliography

- [1] Hamed Ahangari. Internal and external parts of cars, 2021.
- [2] Jane Bromley, James Bentz, Leon Bottou, Isabelle Guyon, Yann Lecun, Cliff Moore, Eduard Sackinger, and Rookpak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:25, 08 1993.
- [3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.
- [4] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets, 2017.
- [5] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data, 2018.
- [6] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.
- [7] 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.
- [8] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006.
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [10] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network, 2014.

- [11] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.
- [12] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016.
- [13] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2020.
- [14] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
- [15] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.
- [16] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. Pytorch metric learning, 2020.
- [17] Samuel G. Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation, 2021.
- [18] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. 2021.
- [19] Geoffrey Hinton Ting Chen. Advancing self-supervised and semi-supervised learning with simclr, 2020.
- [20] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification, 2015.
- [21] Tongtong Yuan, Weihong Deng, Jian Tang, Yinan Tang, and Binghui Chen. Signal-to-noise ratio: A robust distance metric for deep metric learning, 2019.