SimSiam: simple siamese networks for pretraining on unlabeled data

Till Ariel Aczel - s203216

Introduction

It has not been well understood why Siamese networks without negative pairs don't collapse to a degenerate solution, but their simplicity and results make them compelling. This work implements SimSiam [1]: a siamese networks architecture, and investigate the following:

- model performance resistance against **smaller labeled training data-set** size
- predictor optimization strategy, especially the **relative predictor learning rate** compared to the embedding model
- effect of optimizer **weight decay**

Model

Model training is done in two steps. First the model learns general representations via **unsupervised pretraining** on the unlabeled training-set, followed by task-specific **supervised training** on the labeled training-set. In this report for evaluation I am using kNN classification and the commonly used linear evaluating strategy.

$$
\begin{aligned}
\mathbf{t}_1, \mathbf{t}_2 &\sim \mathcal{T}(\mathbf{x}) \\
\mathbf{g}_i &= g\big(f(\mathbf{t}_i)\big), i \in \{1,2\} \\
\mathbf{h}_i &= h(\mathbf{g}_i), i \in \{1,2\} \\
\mathcal{L} &= \frac{1}{2}\Big[\mathcal{D}\big(\mathbf{h}_1, \mathrm{sg}(\mathbf{g}_2)\big) + \mathcal{D}\big(\mathbf{h}_2, \mathrm{sg}(\mathbf{g}_1)\big)\Big] \\
\mathcal{D}\left(\mathbf{u}, \mathbf{v}\right) &= -\frac{\mathbf{u}}{\|\mathbf{u}\|_2} \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|_2}
\end{aligned}
$$

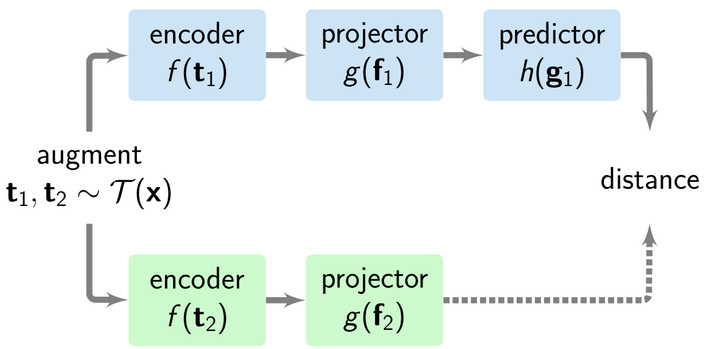

Figure 1: SimSiam model architecture, blue: online network, green: target network. The dotted line represents the stop-gradient operation. Two different augmentations are applied to the same image. The augmented images are fed trough two identical encoders and projectors creating two embeddings. One of the embeddings is processed by the predictor and it's distance is computed to the other embedding with a stop gradient applied.

Augmentation

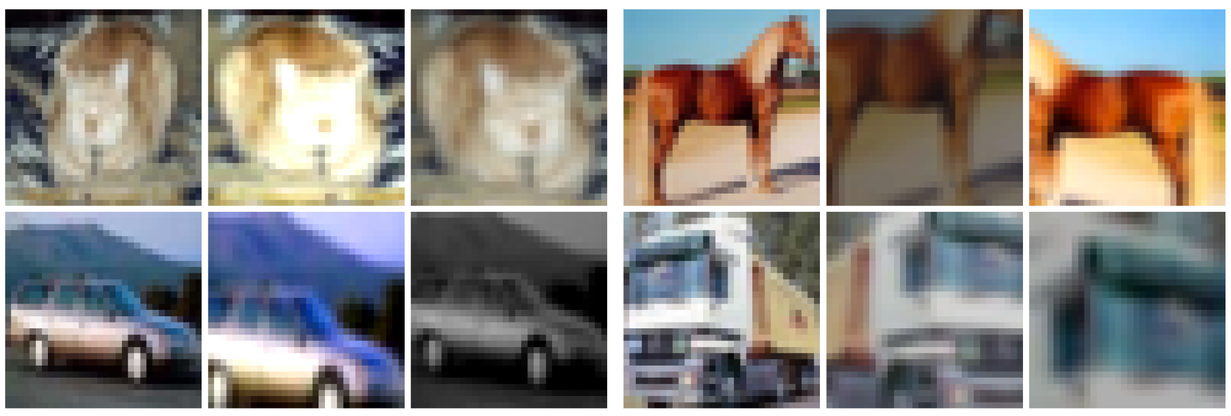

Figure 2: Augmentation samples with the original and two samples for each image. Augmetations include: random resize crop, horizontal flip, color jitter and gray scale.

Reproduction

	kNN acc.	linear acc.
SimSiam [1]	∼90	91.8
reproduction	89.42±0.19	89.94±0.14
80%-20% data split	88.02±0.35	88.34±0.21

Table 1: Test accuracy in percentage. Note that the SimSiam [1] linear accuracy result is with the LARS optimizer. I achieved similar results as presented in the paper.

Predictor learning rate

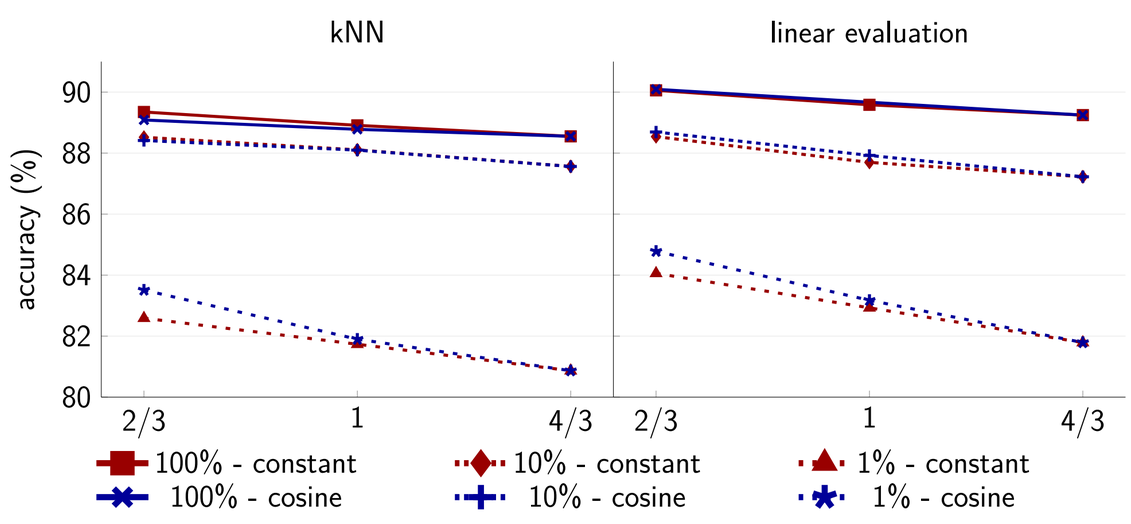

Figure 3: kNN and linear evaluation validation accuracy over different predictor learning rate factor α_h, where $lr_h = \alpha_h \cdot lr$.

Qualitative assessment

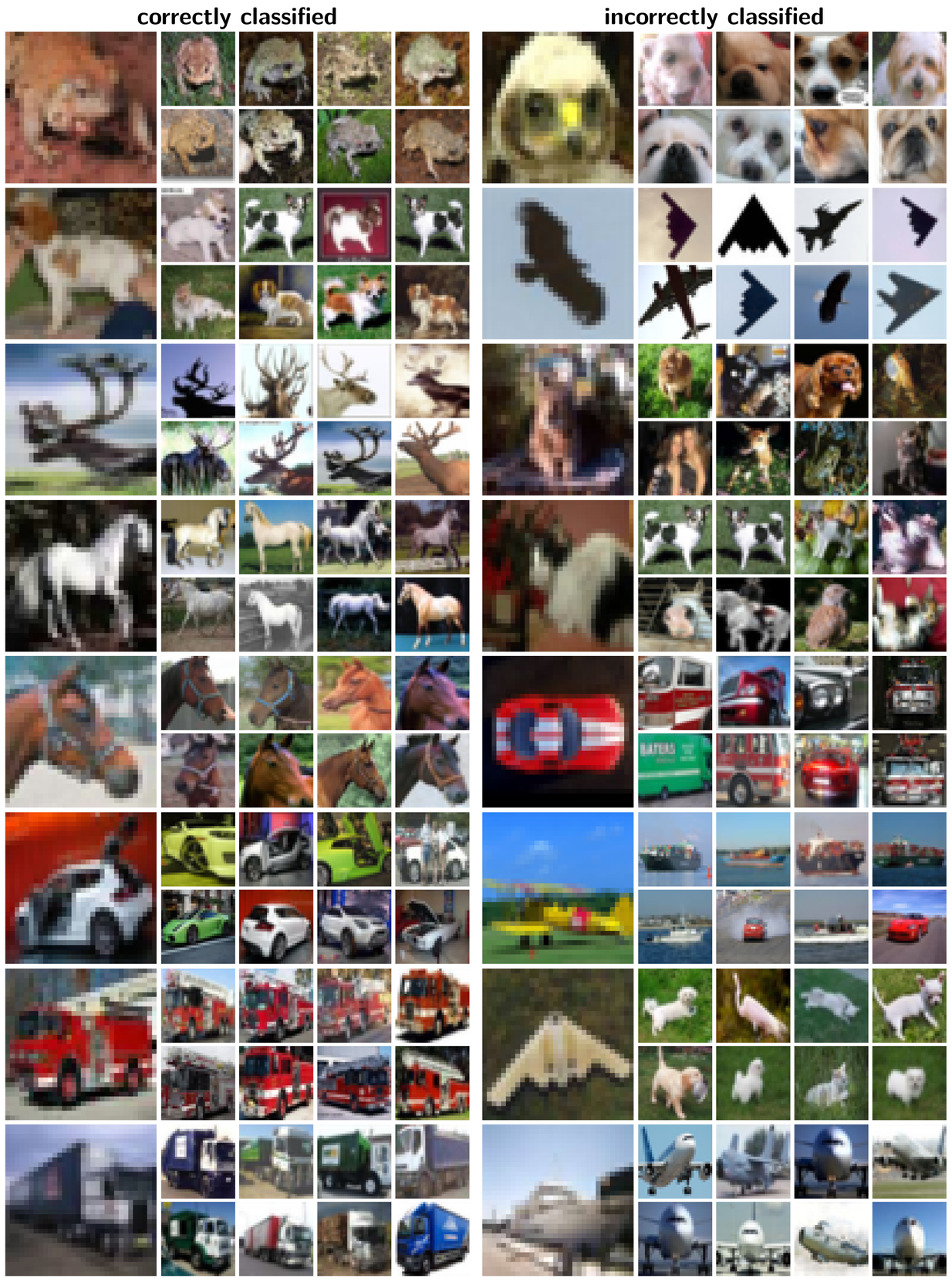

Figure 4: Test images and their closest training-set neighbours. In general the model has problems with images which are rare for the given category. Note that the learned representations are not task specific. For example even though the labels do not have *white horse* or *horse head*, just *horse* it learned the difference between them.

Weight decay

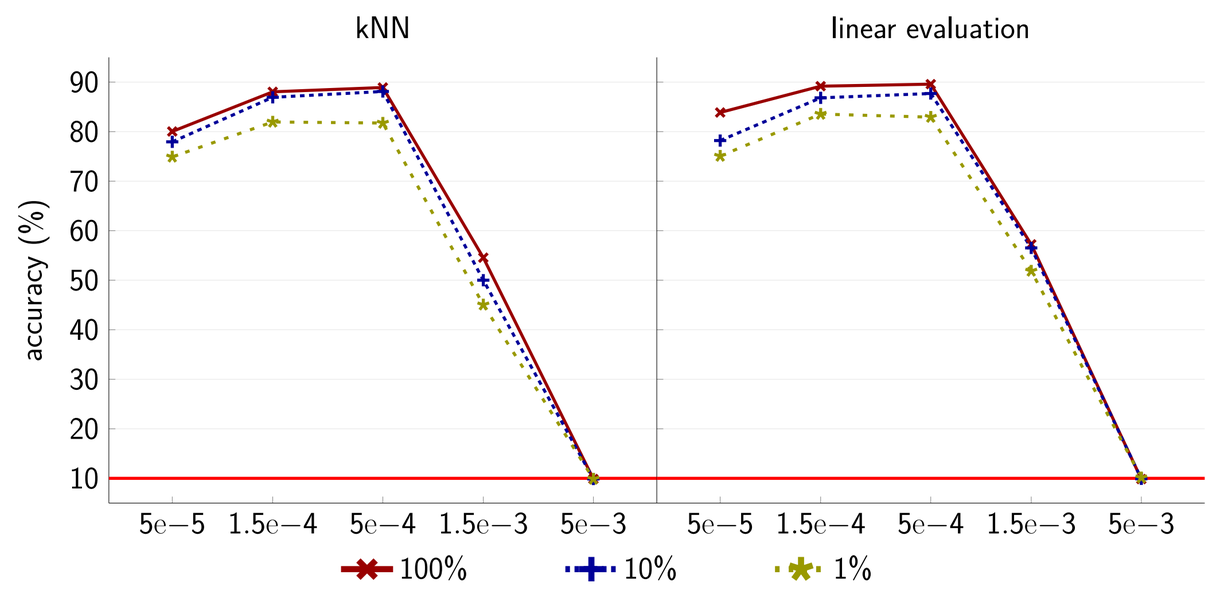

Figure 5: kNN and linear evaluation validation accuracy over different optimizer weight decay. Red line is the random baseline.

Discussion

I have reproduced the results of the SimSiam [1] paper and thus confirmed that a siamese network can be trained on just positive pairs, mode collapse can be avoided. Main takeaways are:

- **Labeling** the training-set **yields diminishing returns**. Only having 10% of the labels achieves similar results as having 100%.
- **Smaller predictor learning rate** might give better results.
- **Weight decay** plays an important role in **convergence**, setting it too small hurts performance, but a high weight decay causes the model to collapse.

References

[1] X. Chen and K. He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

s203216@dtu.dk

https://github.com/tillaczel/simsiam