To tackle task 4 we first tried the following approach. The image triplet was converted to a single image, which was split into quadrants. The left upper corner represented the anchor image, whereas the bottom right corner was filled with straight zeros. We divided the dataset into two parts of equal size. For one half we placed the similar image in the right upper corner and the dissimilar one in the bottom left corner. This approach was reversed for the other dataset half. In doing so we hoped to learn to extract the features according to the position in the total image. However, scores did not exceed ~60 %. Therefore, we explored siamese nets that showed promising results according to Keras tutorials. The network was designed with a pre-trained encoder module and a customized classifier using FC layers. As encoders we initially tried Resnet18/34/50. Performance for these nets was equally good, but did not pass the BL. As a second encoder variant we tried efficientnet-b0 as it has few parameters and did really well on the top 5 % and 1 % errors on the ImageNet dataset. However, scores were barely better than the expectation value of random guessing so that we turned our attention to vgg-11, which turned out to be unsuitable. Specifically, the huge network architecture caused problems training it on our own machines and on Colab. Finally, densnet121 was used as encoder module and yielded the most promising scores on our validation set right away. This network was thus further experimented with. Different data augmentation techniques such as random flips, resizing with center cropping, and image perspective changes were implemented. Combinations of these transformations were tried to avoid overfitting on the validation set and provide us with a validation set score that is comparable to the achieved test score. In the end, the computed validation set accuracies matched the test accuracies reasonably well (val acc. was usually 3 % above test acc.). By using several such transformations the score climbed to ~68.0 %. Due to scores remaining at this level of accuracy we decided to use fewer transforms leading to scores of ~70 %. As far as loss functions are concerned, we first tried contrastive loss as two papers proposed it as state-of-the-art and the loss function is quite intuitive for this task. After a few iterations we implemented triplet loss as proposed on the keras webpage. The model that managed to pass the BL was trained with triplet loss. Throughout the project the discussed encoder modules were trained both with frozen parameters as well as by updating pre-trained weights. No general advice can be proposed here. On the one hand, ResnetXX performed ~10 % poorer with frozen weights compared to a “full” backward pass. densenet121 yielded a better performance by freezing the pre-trained weights on the other hand. The final breakthrough was achieved with densnet121 as encoder (frozen weights) and a simple FC layer image similarity classifier.