Ann project classification

Stein Vandenbroeke s0205627

May 29, 2025

1 Training the model

First, we try to find good parameters to train the model. The specific case determines what parameters to choose.

1.1 Vanilla supervised learning

We can set different values for certain parameters: batch size, learning optimizer, image size, scheduler, and image augmentation. To determine the optimal values for these parameters, we conducted various tests. As shown in Figure 1, the impact of different-sized images is illustrated, and we can conclude that a larger image yields the best accuracy but also requires the longest training time. The effect of data augmentation (Random horizontal flip) is shown in Figure 2. This figure shows almost no impact of augmenting or not augmenting. This can be because our training dataset is large enough that there is no overfitting and thus no need for augmenting the training data. I used the ReduceLROnPlateau scheduler to get better results based on the validation loss. When the validation loss does not increase for three consecutive epochs, the learning rate will be decreased by 0.1. This gave more stable results than the usage of MultiStepLR with a rate change at 60% and 80% of 0.1. The difference can be found in image 3. To determine the optimal batch size and learning rate, I experimented with various combinations. I concluded that a batch size of 60, combined with a starting learning rate of 0.0005 (see Figure 4), gave good results with a test accuracy of 85%. When examining the t-SNE plot of this model, we can see that most of the items are well-separated, with only a few overlapping. When looking at the confusion matrix in Figure 6. We can see that categories 0 and 4 are often confused, with 18% of category 4's predictions going to category 0 (This is no surprise because categories 0 are bedrooms and categories four are living rooms they look very similar in comparison to, for example, living rooms and the sea). If we compare this with the t-SNE values, we see that many purple and blue dots are clustered together on the right side. This is just an example. If you look further, you can see a pattern between items that are clustered together and that are confused in the confusion matrix.

1.2 SimCLR

First, we train a model using unlabeled data (self-supervised contrastive pre-training). When this is trained, we add an extra linear classification layer. To train the model using unlabeled data, we transform our image into two images using RandomResizedCrop, RandomHorizontalFlip, Random ColorJitter, RandomGrayscale, and resize. And the model output for two augmentations of the same image must be as similar as possible. To do this, we use the info_nce_loss loss function. After training this, we freeze the model and add a linear classification layer that will be trained using a labeled dataset (in our case, the same dataset). The advantage of self-supervised contrastive pre-training is that overfitting is almost impossible because we always use two different augmentations of the same image. And our dataset is not labeled, allowing for easier data collection (in our case, we don't use this advantage because we only got a labeled dataset). To pretrain the model, we again try to find the optimal parameters after some testing. We found that the parameter: a batch size of 256, input image size of 128, learning rate of 0.001 and no scheduler gave the best results. The longer we trained the model the better our fine_tuning results where (as expected). And the only reason to stop the training processes was through computational and time limits. After 800 epochs (40 min training) our top5 accuracy calculated with the cos similarity was around 75%. When fine-tuning the model with a linear classification layer, it almost immediately achieves its best validation accuracy of 83%, as shown in

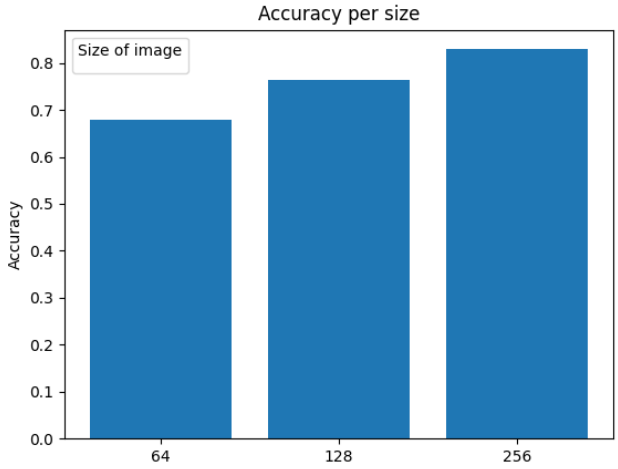

Figure 1: Impact of input size of the images

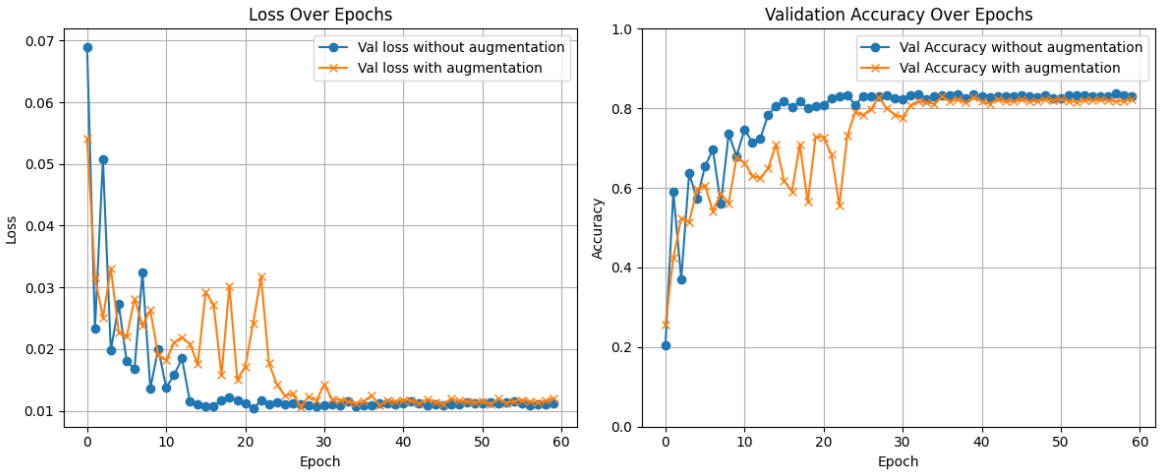

Figure 2: Impact of training set augmentation

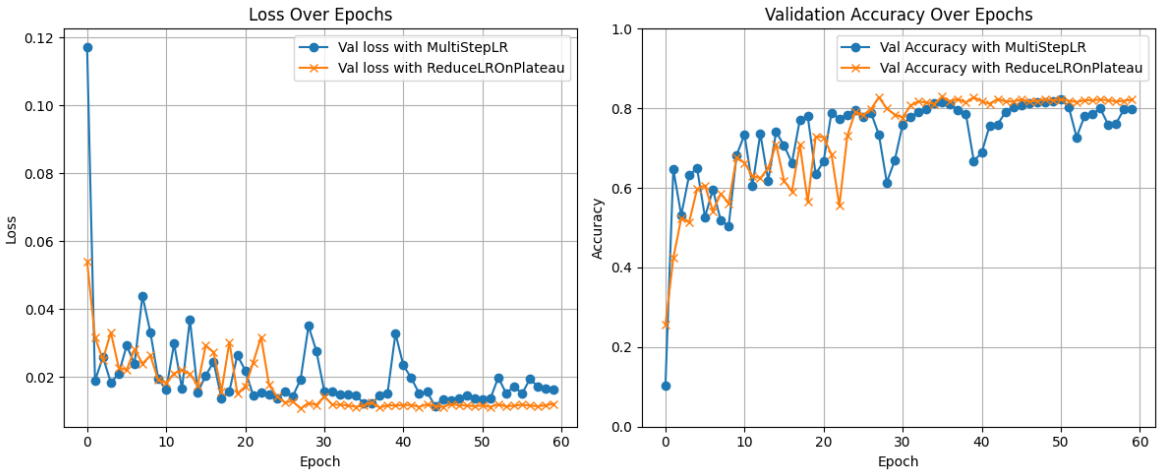

Figure 3: Impact of different schedulers

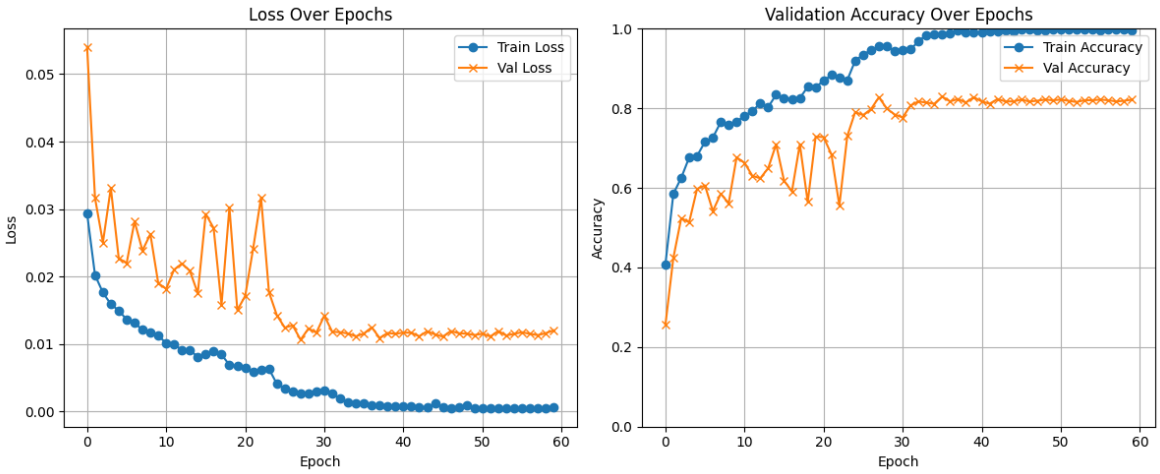

Figure 4: Best vanilla supervised learning model

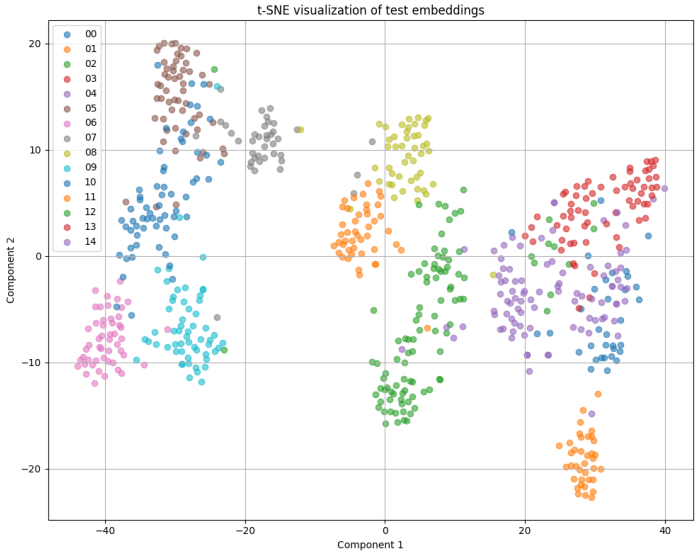

Figure 5: t-SNE visualization vanilla supervised learning model

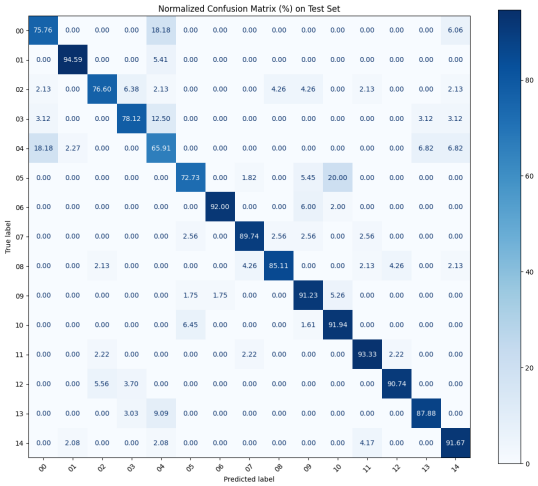

Figure 6: Confusion matrix vanilla supervised learning model

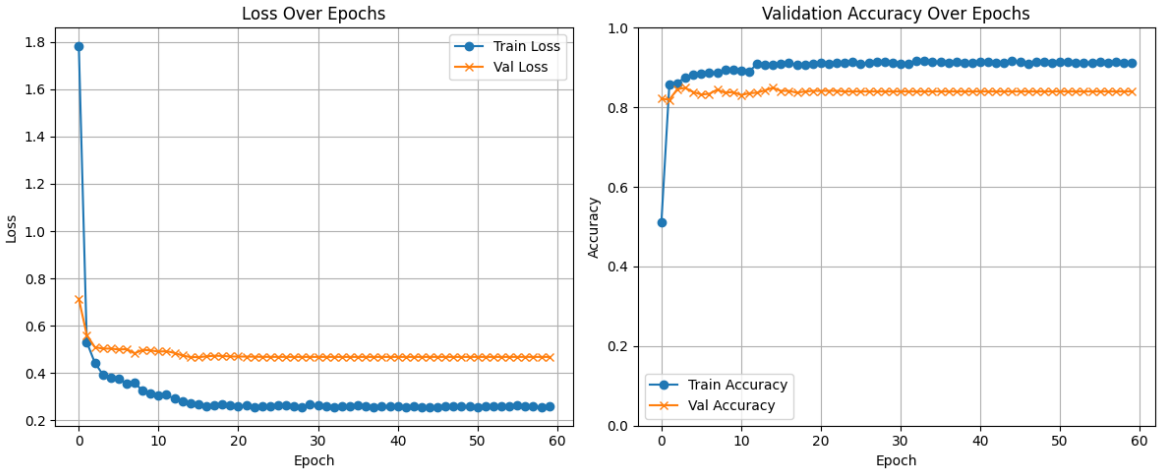

Figure 7: Loss and validation accuracy of SimCLR

Figure 7, where it achieves 79% in just one epoch. This is because the pre-trained network has already learned all the important features, and the fine-tuning only has to use this to determine what category it is. When we use this model on our test dataset, we got an accuracy of 86%. When we analyze the t-SNE plot **??** from before the linear layer, it is clear that SimCLR successfully clusters the same image groups together, except for category four, which is spread out among the other categories. This translates in the final results where this category had a low accuracy (See Fig

1.3 Supervised Contrastive Learning (SupCon)

SupCon works in principle the same as SimCLR, it also uses the pre-training and then the fine-tuning linear classifier layer. The difference is that now, instead of training on the difference between the two images, we also use the labels. To make this change, we only had to change our transform function (we don't need 2 augmentations of the same image) and our loss function. Again, we tried different hyperparameters, and the following values gave us the best results: a batch size of 256, input image size of 128, learning rate of 0.001, 400 epochs and no scheduler gave the best results. When we fine-tuned this model, we got a validation accuracy of 80% after one epoch and after three epochs around 85% (see Figure 10). The test dataset got an accuracy of 87%. If we look at this t-SNE plot see Figure 11, we can see all categories are clustered well, but again with an overlap at categories zero, three, and four (all in-house images). This again translates to the final linear layer training, where these categories perform worse (see Figure 12)

2 Comparing the models

Now that we have the best hyperparameters selected for our models, we can compare them against each other. Table TABLE reference shows the selected parameters and accuracies of each model.

Model	Optimizer	Scheduler	Start LN	Image Size	Batch Size	Epochs	Criterion	Test Accuracy
Supervised Learning	AdamW	ReduceLROnPlateau	0.0005	256x256	60	60	CrossEntropyLoss	85.50%
SimCLR Pretraining	AdamW	None	0.001	128x128	256	800	InfoNCE	–
SimCLR Finetuning	AdamW	ReduceLROnPlateau	0.001	128x128	1000	40	CrossEntropyLoss	86.09%
SupCon Pretraining	AdamW	None	0.001	128x128	256	400	InfoNCE	–
SupCon Finetuning	AdamW	ReduceLROnPlateau	0.001	128x128	1000	40	CrossEntropyLoss	87.20%

When comparing the results of our models, we can see that SimCLR performs better than the normal model and SupCon outperforms SimCLR. This was expected because, with SimCLR, we don't use the labels to pre-train the model, although we have access to them. SimCLR and SupCon also have the advantage that there are way more training examples because we augment the dataset every time differently, so during training, we never have multiple times the same sample. Another advantage

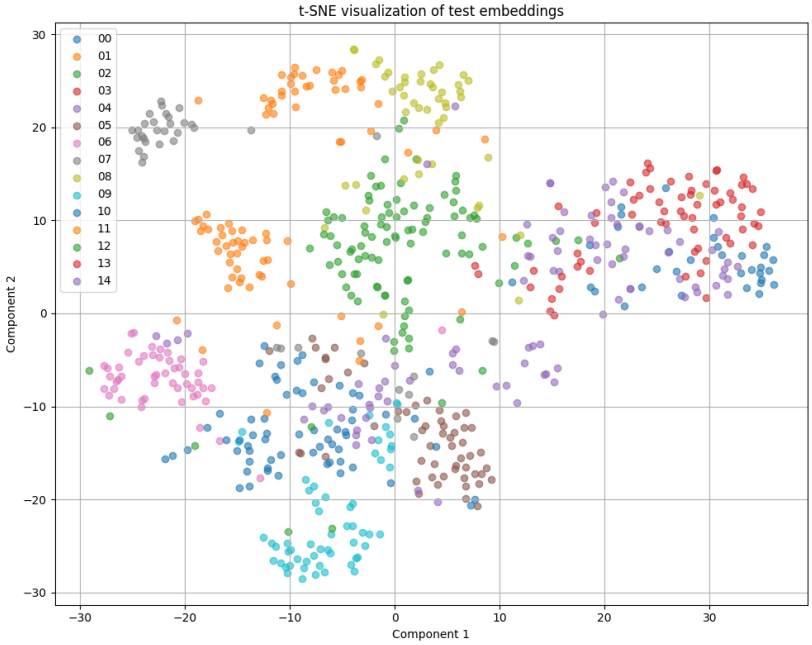

Figure 8: t-SNE visualization SimCLR model before linair layer

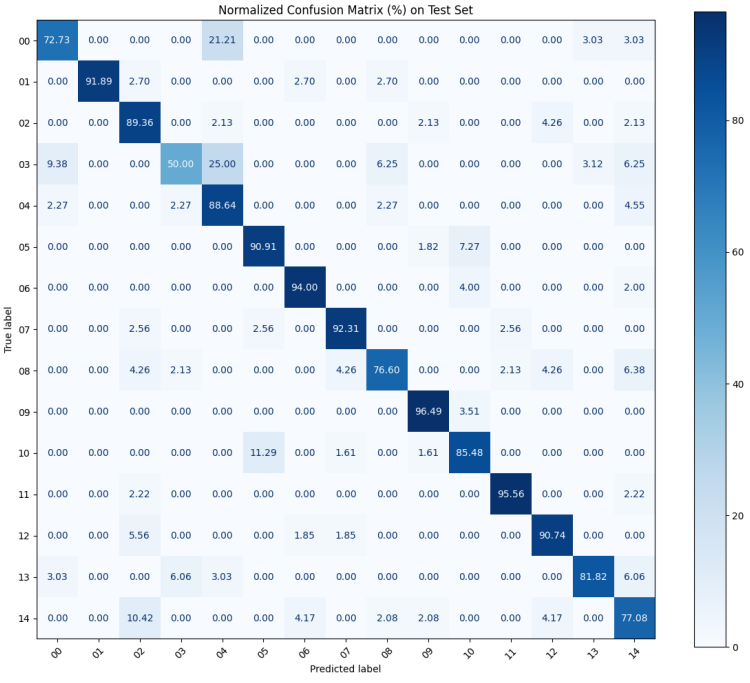

Figure 9: Confusion matrix SimCLR model after linear layer

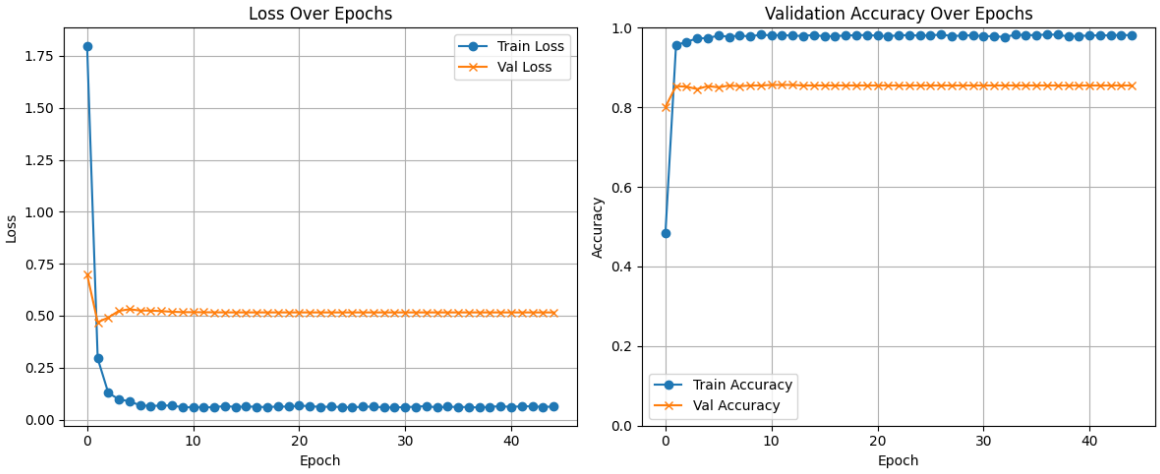

Figure 10: Loss and validation accuracy of SupCon

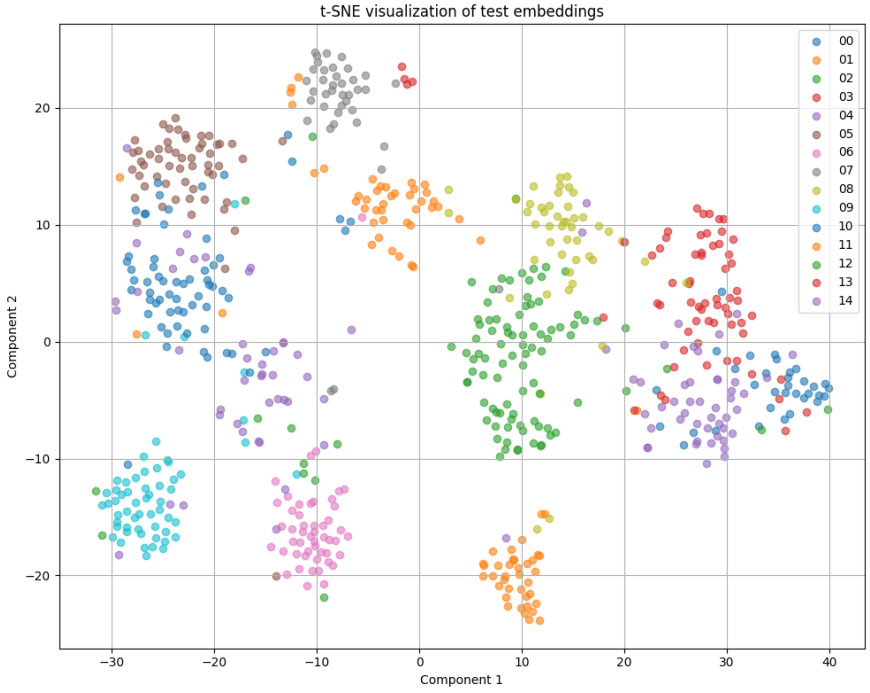

Figure 11: t-SNE visualization of SupCon model before linair layer

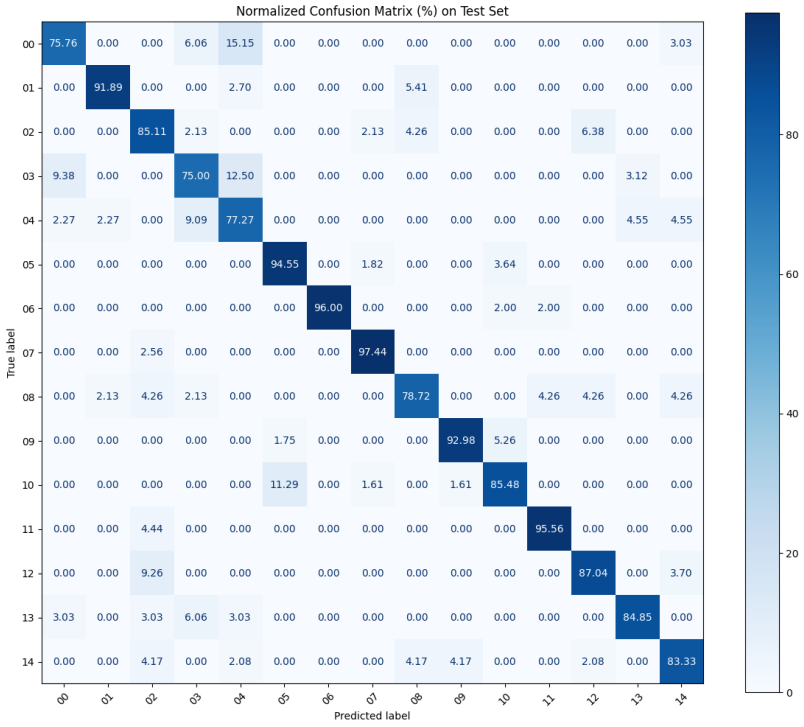

Figure 12: Confusion matrix SupCon model after linear layer

is that these pre-trained models can be used for other kinds of classifications, such as those that distinguish between inside and outside without the need to change them. The only thing you should do is replace the linear layer with another classifier.