Homework 2 - Machine Learning

Andrea Sanchietti 1883210

January 2022

Abstract

In the following document I will illustrate the work done and the results of my models. Firstly, I tried to solve the classification problem through a custom architecture, and compared it with ResNet, achieving good results. I then trained three different autoencoders in order to learn a latent space and used the best one to classify images.

1 Dataset

I chose two different dataset to work with. The first one is EuroSat, a collection that addresses the challenge of land use and land cover classification using Sentinel-2 satellite images. It consists of 10 classes with in total 27,000 labeled and geo-referenced images. I divided this dataset in 80% training set and 20% validation set. The second dataset is Birds 450 species, and offers 70,626 training images, 22500 test images(5 images per species) and 2250 validation images.

2 Related Work

In order to deliver a good result, I started looking for papers that could help me for the classification and, in general, that concern about the datasets I chose. These works helped me understanding the state of the art on the datasets and have a clear idea of what are the methods used today.

For the EuroSat problem, I found some interesting works:

- The first papers were the oen by the datasets: *EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification* [3] and *BIRDS 450 SPECIES- IMAGE CLASSIFICATION* (on kaggle).
- A paper that I found about *EuroSat* was [6] *In-domain representation learning for remote sensing*, which is an investigation on the in-domain representation learning to find out which are the important characteristics for a dataset to be a good source for remote sensing representation learning.
- Another interesting work that I found was *Self-supervised Learning in Remote Sensing: A Review* [11]. In this paper, the authors show different self-supervised methods tested on some geo-related datasets. They divided methods based on the taxonomy of self supervised learning: Generative (Autoencoders and GANs), Predictive(Spatial, Spectral and Temporal context) and Contrastive (Negative sampling, Clustering, Knowledge distillation and Redundancy reduction). This led me to try an approach based on an autoencoder and an svm.
- Led by the idea of the previously mentioned approach, I started looking at papers on autoencoders that could help me and I found two interesting works: *Walking the Tightrope: An Investigation of the Convolutional Autoencoder Bottleneck* [4] and *Push it to the Limit: Discover Edge-Cases in Image Data with Autoencoders* [5].

  The former investigates the effect of the width, height and depth on the correct reconstruction of the image in an Autoencoder, showing that the those who influence more performances are width and height of the latent space.

  The latter showed instead an analysis of the latent space based on PCA. They applied the decomposition to the encodings and then sorted the dataset according to the values of individual principal components. Finally, they found that samples at the high and low ends of the distribution often share specific semantic characteristics.
- I also found a work called *Learning to Generate Images with Perceptual Similarity Metrics* [8] that showed a metric for autoencoders called MS-SSIM. This metric considers the multi-scale structural-similarity of an image, yielding a better loss function if compared to MSE and MAE.
- I then looked for some state of the art models for classification and read about *CoAtNet* [1] (a convolutional transformer that established a new state-of-the-art result for ImageNet in 2021), *MobileNetV2* [7] (a network that used Inverted ResidualS and motivated me to investigate Residual blocks), *ResNet* [2] (the network that introduced Residual blocks)
- Two more papers worthy of a citation that I read but didn't apply for the classification task were. The first one is the one that introduced the self-attention mechanism in 2017 and introduced transformers :*Attention Is All You Need* [10]. The second one is a paper that shows how pooling layers in convolutional networks can be replaced by strides: *Striving for Simplicity: The All Convolutional Net* [9]

3 Tested Methods for EuroSat

For this dataset, the first thing that I did was to train ResNet50 in order to have a model to compare my experiments with. The train had 5e-4 as learning rate, SGD_M, momentum at 0.9 as [6] suggested and 40 epochs. The result of the training are shown together with the plots of the other models in order to have a direct comparation. The best result on the validation set was of 0.9191, with an epoch training loss of about 0.314.

I then started experimenting with convolutional and residual blocks. My first model (Model_0) was a simple CNN with some convolutional kernels, relu as activation function for hidden layers and softmax for the output layer. I used both the Adam and the SGD-M optimizers and the one that worked better was the second one. The loss function was the categorical accuracy (with one hot encoding), the number of epoch was 20 and the learning rate was set to 1e-3 (I also tried 1e-4 but it was too slow). The model managed to learn the training set well, with about 0.99 of categorical accuracy. However, the validation started converging after only six epochs and the maximum value reached was 0.77 (fig. 1)

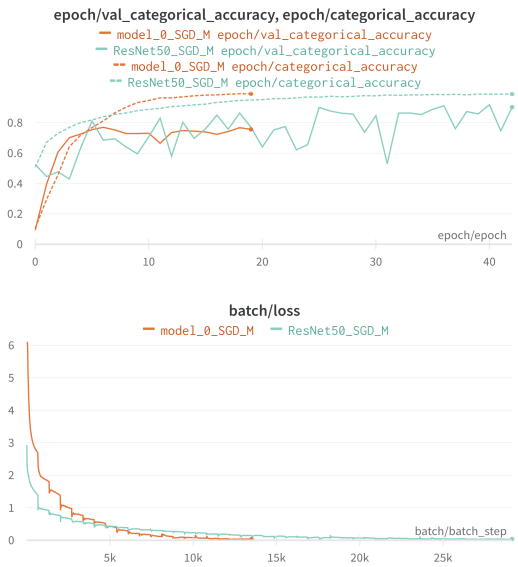


Figure 1: Accuracy and loss for model_0.

Based on this result, I modified the network by adding some Residual blocks 2.

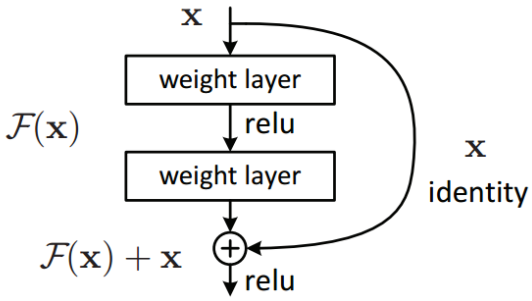


Figure 2: Residual block presented in the ResNet paper [2].

A residual block helps because, as the paper that introduced them explains [2], is easier to optimize the residual mapping than to optimize the original one. The new model (Model_1) was trained with 40 epochs and 1e-6 as learning rate. I tested both SGD and SGD_M as optimizer and the best result was yielded by the SGD optimizer. For the SGD_M version I also halved the number of parameters in order to overcome possible overfittig given by the high amount of parameters. Figure 3 shows results of the training phase.

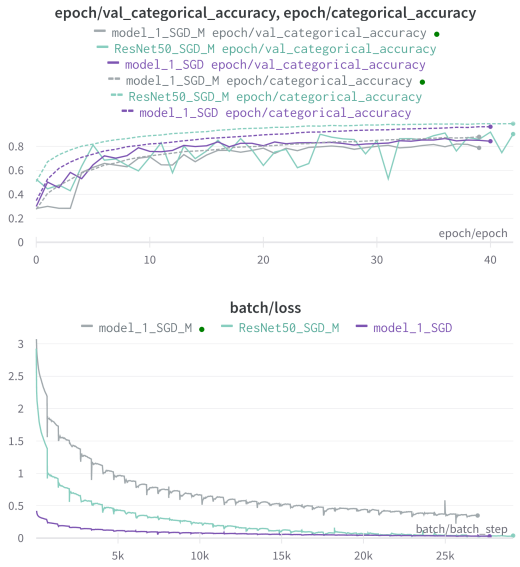


Figure 3: Accuracy and loss for model_0.

Since the SGD_M version didn't diverge with only 40 epochs, I tried to fit more epochs with a lower learning rate (1e-7) and saw what could happen. From the plot 4 we can see that the model started underfitting and was not able to learn more than 0.90 for the training accuracy and 0.83 for the validation:

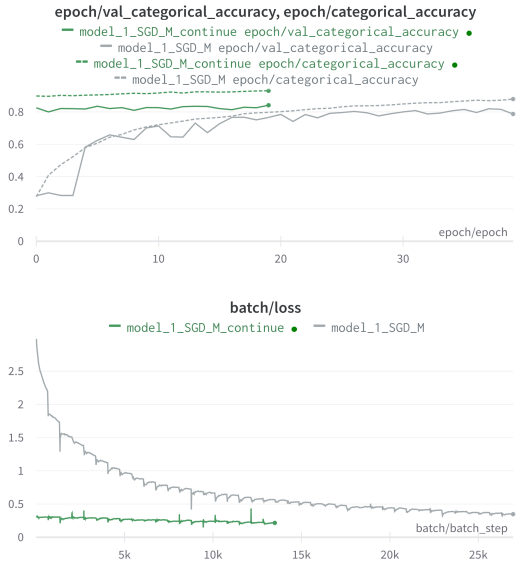


Figure 4: Continue of training for Model_0 with SGD_M.

My last model consisted in removing some of the convolutional layers from the second model in order to decrease the number of parameters (Model_2). I used the Adam optimizer for this experiment, and the model outperformed the previous one, with 0.8839 of validation accuracy (+1.89% improvement). It was also able to fit faster the dataset, surpassing the previous model after the 20^{th} epoch. However it didn't overcome ResNet by about 3.5%:

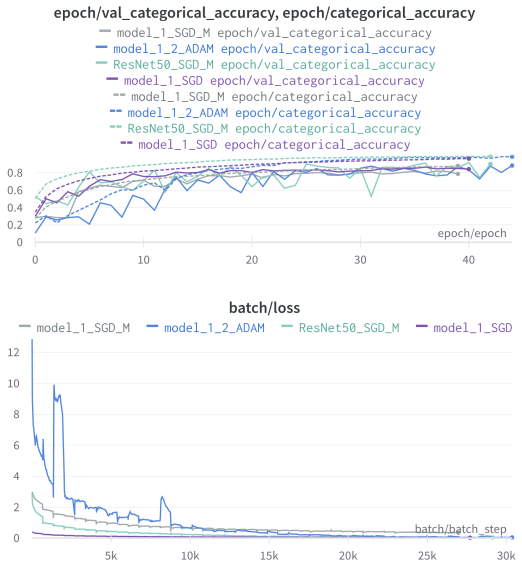


Figure 5: Accuracy and loss for model_2.

Given the acceptable results from this second model (0.8839 of validation accuracy, compared to 0.9191 of ResNet), I decided to plot some confusion matrices:

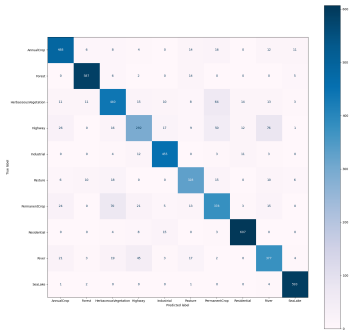


Figure 6: Confusion matrix for My best model

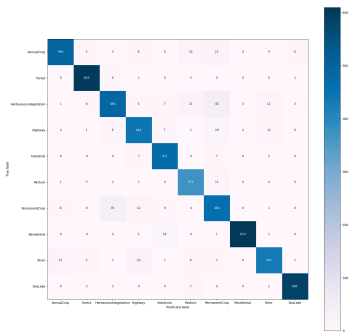


Figure 7: Confusion matrix for ResNet

The figures show that my network struggles at differing highways from rivers, permanent crops from herbaceous vegetation and highways. On the other hand, ResNet only struggles in highways and rivers.

At the end of training I calculated Precision and Recall for ResNet and Model_1 (tab. 3). I accidentally lost weights for the other models because of disk space, so I could not load and test them too.

Model	Precision	Accuracy
ResNet	0,9185	0,9987
Model_1	0,84	0,81

4 Tested Methods for BIRDS-450

Something that really surprised me is how my models poorly scaled when used for the BIRDS-450 dataset. They indeed had a problem related to overfitting because, as it can be seen by the plot 8, my best model started converging sooner if compared to ResNet. This problem was related to the number of parameters of my network as it was over one hundred million.

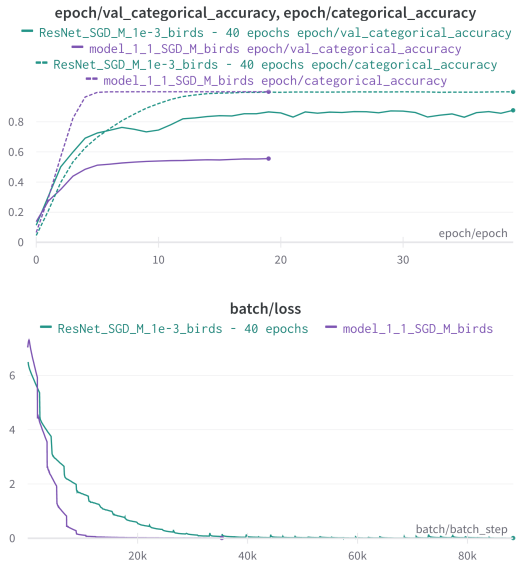


Figure 8: Accuracy and loss for birds. The overfitting problem is clear.

In order to fight back overfitting I tried applying three different techniques:

- **Argumentation**: I added data argumentation in order to have a better flexibility of the network. This argumentation was applied to every model.
- **Dropout**: I added dropout layers for every convolutional layer with dropout factor of 0.2 in both Model_4.
- **L2 kernel regularization**: I used L2 regularization for every convolutional layer both in Model_4 and Model_5.

I also used the model with halved convolutional filters (the same used for Model_1 SGD_M). This made the network converge faster than before and also improved performances (the number of parameters was then 15 millions for the model).

But let's go in order. The first thing that I did was training Model_1, ResNet and compare results 9:

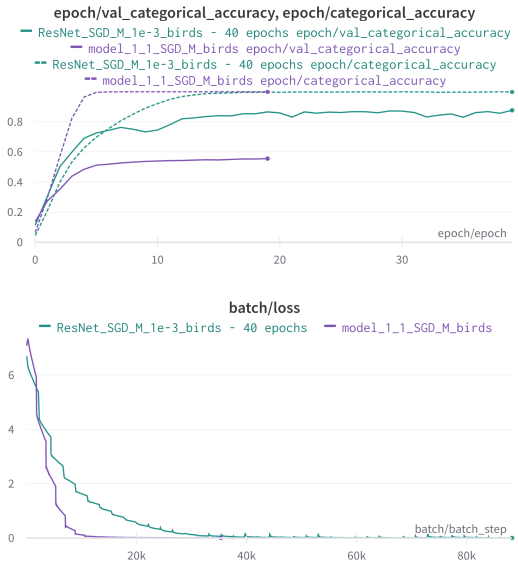


Figure 9: Accuracy and loss for Model_0 and ResNet on Birds-450

The highest validation accuracy score was reached by ResNet with 0.8754. At this point I tested Model_3 on the problem as it has less convolutional layers (-1 for each convolutional block) and filters (1/4 for each convolutional layer), and could led to less overfitting. In figure 10 we can see how actually it didn't perform as expected:

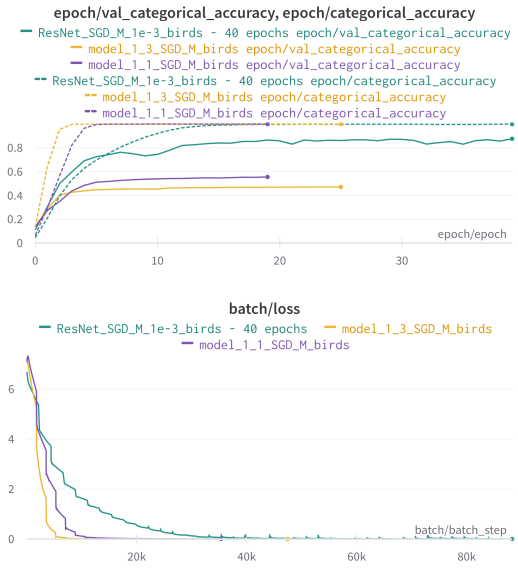


Figure 10: Accuracy and loss for Model_3 compared to previous ones on Birds-450

Model_3 indeed converged faster than Model_1 and ResNet, however it also overfitted more than them. So I decided combining the methods described in 4 and test a different method. Model_4 had both dropout and kernel L2 regularization, with the same number of filters as Model_1, but didn't brought improvements:

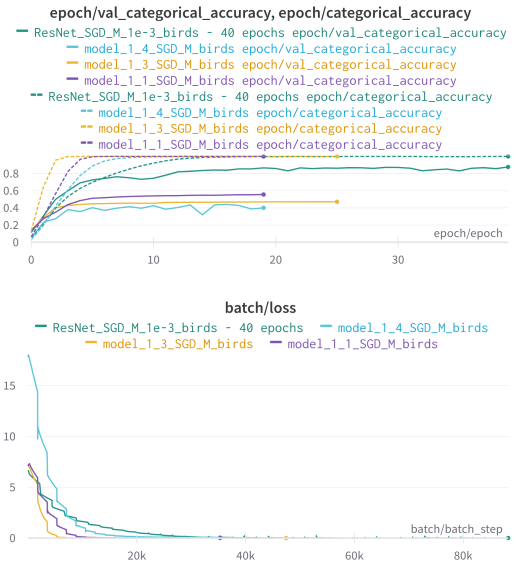


Figure 11: Accuracy and loss for Model_4 compared to previous ones on Birds-450

At this point I designed Model_5 with less filters (half for each layer if compared to Model_4), with batch normalization instead of dropout and without residual blocks. I trained this model with three different optimizers (SGD_M and Adadelta) and different learning rates §(fig. 12). The one that fitted better than the others was SGD_M with learning rate 1e-4 and 0.7271 of validation accuracy . On the other hand, Adadelta did not perform well. As we can notice by figure 13, this new model was able to overcome overfitting better that the previous models. However we are still far form ResNet.

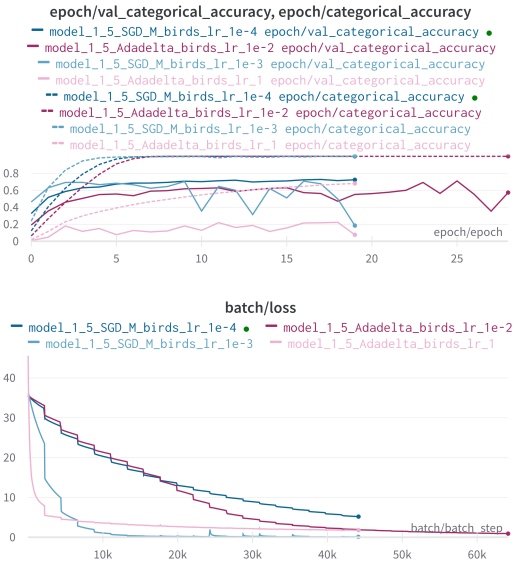


Figure 12: Accuracy and loss for different learning rates and optimizers for Model_5 on Birds-450

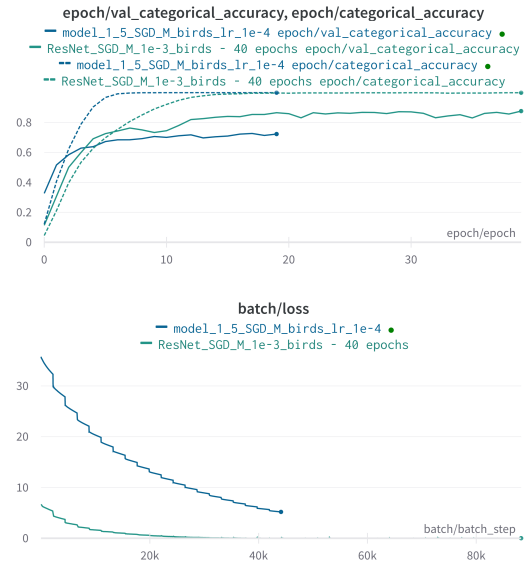


Figure 13: ResNet vs my best model (Model_5) Birds-450

My final fully custom test was based on reducing the number of filters from Model_5 in order to check if the validation accuracy could be increased by generalizing more. Model_6 was then designed and tested with the same optimizer and learning rate that gave best results for Model_5. I also increased the size of kernels for the first three convolutional blocks. This helped the network to fit without overfitting too much. Results are shown in the next figure (14)

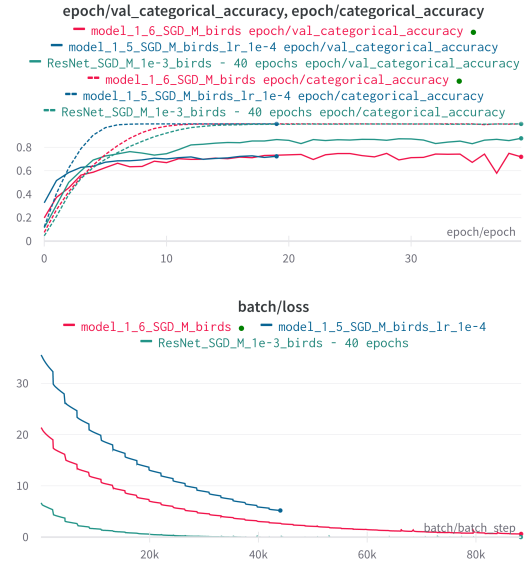


Figure 14: Model_6

we can notice that Model_6 was able to improve only by a little percentage the validation accuracy (best at 0.748).

My final test was based on transfer learning: I used Inception v3 pretrained on imagenet with a custom output. This last model outperformed every other model, ResNet included:

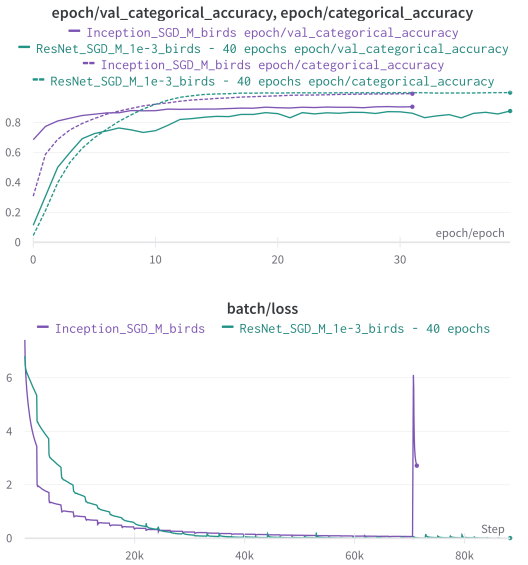


Figure 15: Inception v3 vs ResNet

I then plotted two confusion matrices, one from my best model and the other from ResNet. As you can imagine they are huge, so I will not display them in this elaborate. I will put them in the folder.

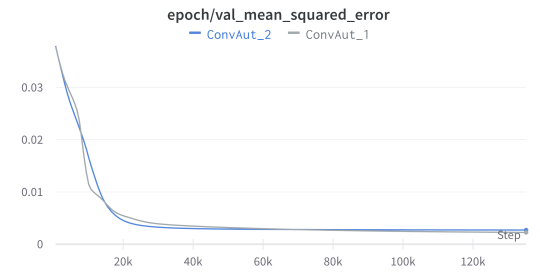
I finally measured Precision and accuracy for Model_5 and Model_6 (tab. 4). Just like for the previous chapter, I accidentally lost weights for the other models because of disk space, so I could not load and test them too.

Model	Precision	Accuracy
Model_5	0,727	0,629
Model_6	0,748	0,67

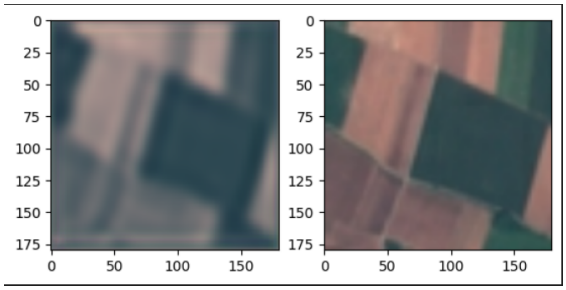
5 Autoencoders for EuroSat

As said during the introduction, I wanted to train an autoencoder in order to learn the latent space for the EuroSat dataset and then use an SVM for classification. During this work, I could not figure out how to make the sklearn implementation of the SVM with the encoder (i reached about 10% accuracy, which is pretty low), and since there is a little time left and i already trained a lot of models, i will only report loss and results from the autoencoders.

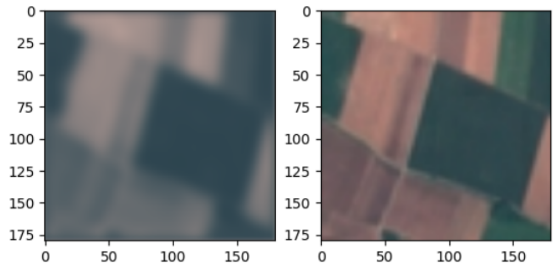
So the first thing that I did was to train two different autoencoders. They both reduced the width and height of layers to 45×45, but the thickness for the first one was of 32 while the thickness for the second was just 8. The plot of the validation loss function (I used MSE) and some reconstructions follows 16:



(a) Validation loss function



(b) Reconstruction at iteration 200 for the first autoencoder



(c) Reconstruction at iteration 200 for the second autoencoder

Figure 16: Loss and Results for Autoencoders

As we can see, the reconstruction from the second autoencoder is sharper.

6 Conclusions and Future Work

In conclusion, an investigation on two different dataset was done during this work and some of the tested models behaved very well.

I tested with my hands how to deal with overfit and underfit by experimenting with different methods that we studied during class.

I also looked for state-of-the-art models and found out about self-attention and transformers (that I didn't use because we didn't study them during the course) which are able to yield astonishing result even for classification problems (see [1]). I also experimented with residual blocks and saw how they are able to avoid overfitting and help the network to converge well.

References

[1] Zihang Dai et al. *CoAtNet: Marrying Convolution and Attention for All Data Sizes*. 2021. DOI: 10.48550/ARXIV.2106.04803. URL: https://arxiv.org/abs/2106.04803.

[2] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[3] Patrick Helber et al. *EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification*. 2017. DOI: 10.48550/ARXIV.1709.00029. URL: https://arxiv.org/abs/1709.00029.

[4] Ilja Manakov, Markus Rohm, and Volker Tresp. *Walking the Tightrope: An Investigation of the Convolutional Autoencoder Bottleneck*. 2019. DOI: 10.48550/ARXIV.1911.07460. URL: https://arxiv.org/abs/1911.07460.

[5] Ilja Manakov and Volker Tresp. *Push it to the Limit: Discover Edge-Cases in Image Data with Autoencoders*. 2019. DOI: 10.48550/ARXIV.1910.02713. URL: https://arxiv.org/abs/1910.02713.

[6] Maxim Neumann et al. *In-domain representation learning for remote sensing*. 2019. DOI: 10.48550/ARXIV.1911.06721. URL: https://arxiv.org/abs/1911.06721.

[7] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: (2018). DOI: 10.48550/ARXIV.1801.04381. URL: https://arxiv.org/abs/1801.04381.

[8] Jake Snell et al. *Learning to Generate Images with Perceptual Similarity Metrics*. 2015. DOI: 10.48550/ARXIV.1511.06409. URL: https://arxiv.org/abs/1511.06409.

[9] Jost Tobias Springenberg et al. *Striving for Simplicity: The All Convolutional Net*. 2014. DOI: 10.48550/ARXIV.1412.6806. URL: https://arxiv.org/abs/1412.6806.

[10] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

[11] Yi Wang et al. *Self-supervised Learning in Remote Sensing: A Review*. 2022. DOI: 10.48550/ARXIV.2206.13188. URL: https://arxiv.org/abs/2206.13188.

7 Models Architecture

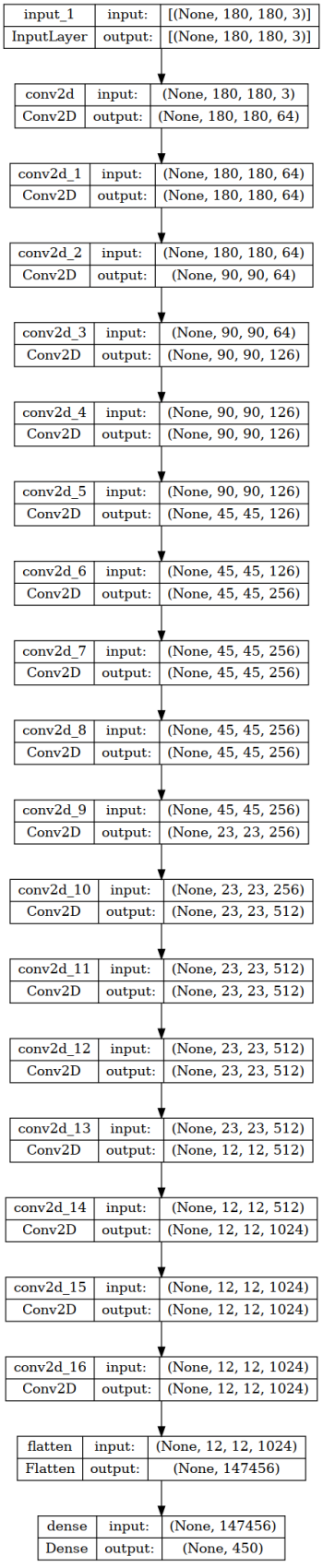


Figure 17: Model 0

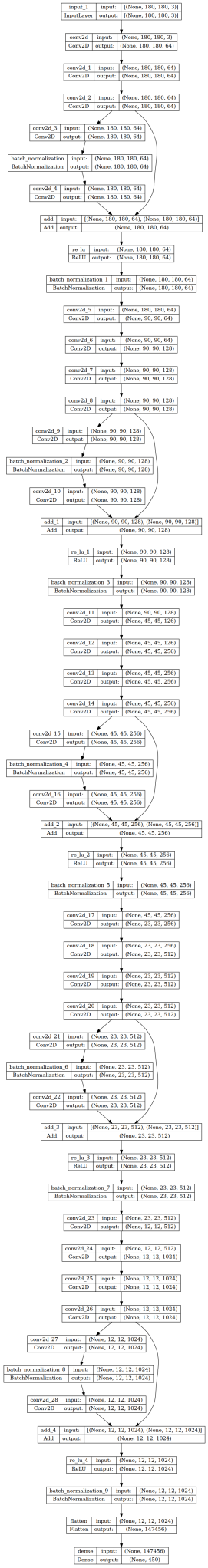


Figure 18: Model 1

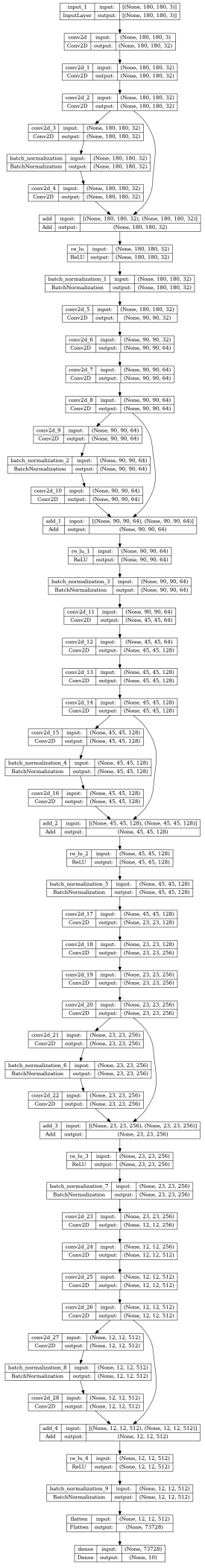


Figure 19: Model 1 rescaled

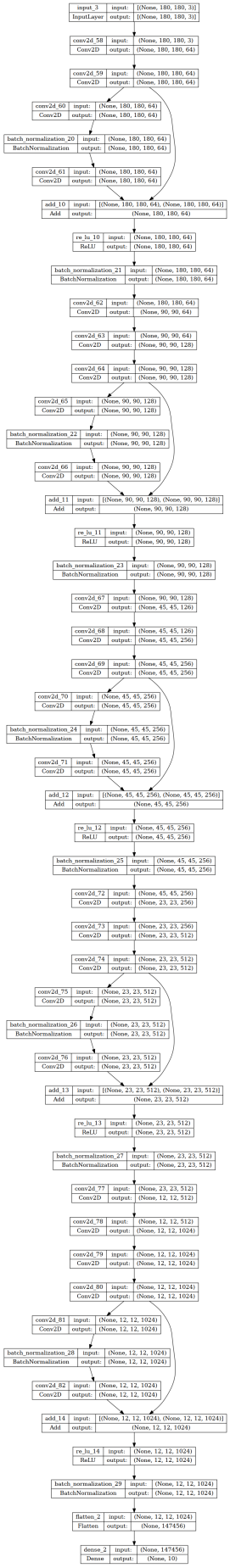


Figure 20: Model 2.

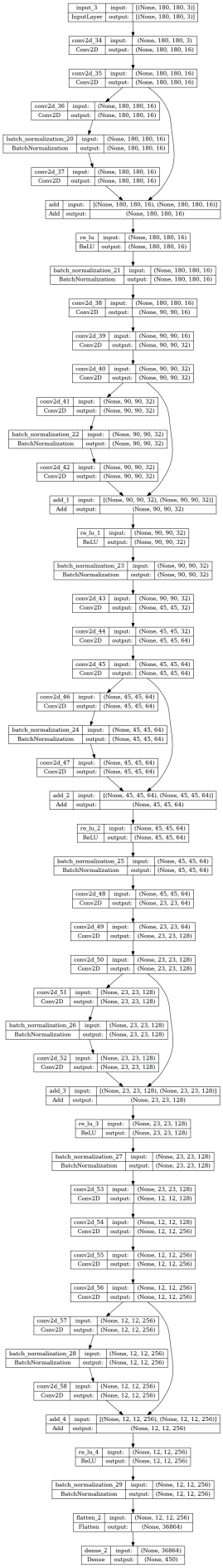


Figure 21: Model 3.

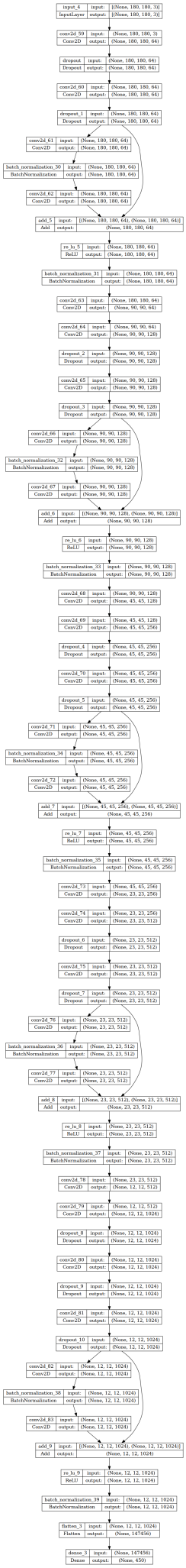


Figure 22: Model 4.

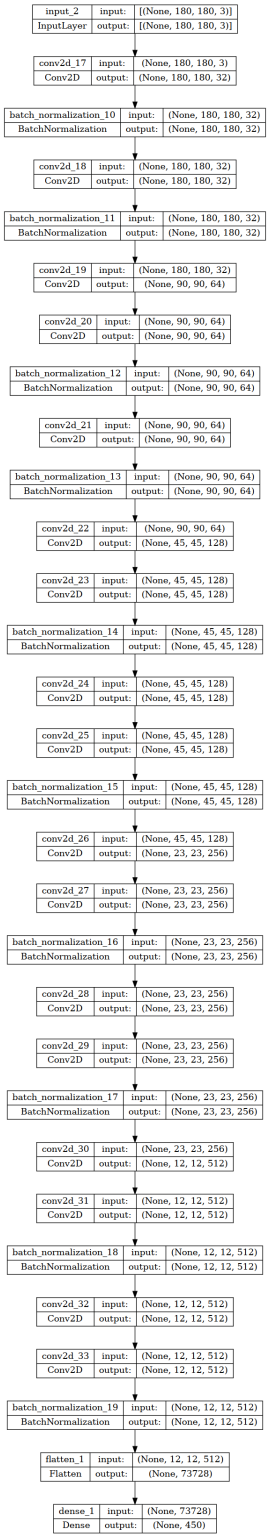


Figure 23: Model 5.

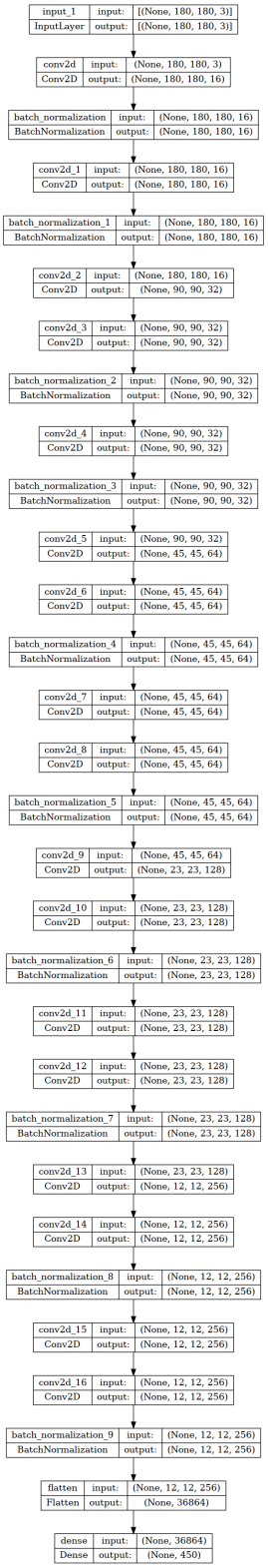


Figure 24: Model 6.