

VSN - Koalarization in PyTorch

Casper Smet - 1740426

2022/04/10

1 Introduction

The colourisation of grey-scale images is useful in many different domains, states Federico Baldassarre, 2017. With perhaps the most notable example the re-mastering of historical photographs.

In the paper "Deep Koalarization: Image Colorization using CNNs and Inception-Resnet-v2", the authors claim that extracting high level features¹ can improve the colourisation results. Their method uses CIELAB colour space (Luo, 2014). CIELAB contains three channels, with channel L(uminescence) being a kind of grey-scale of the image, and A*B* being colour channels. Federico Baldassarre, 2017 attempts to predict the latter in their experiment using the prior.

Federico Baldassarre, 2017's results were impressive. However, they made one critical error. They use the L-channel from the original, in-colour, image. Ergo, their model uses some apriori that it should not have.

In this paper, we replicate the experiment with the Luminescence channel taken from a grey-scale image. We hypothesise that the results using grey-scaled images will be similar in quality to the original method's results, as the differences in the luminescence channels of the two methods are, to human vision, small.

¹e.g. recognising plants or water

Contents

1	Introduction	1
2	Background	3
2.1	Original paper	3
2.1.1	Network architecture	3
2.1.2	Experiment	3
2.2	Inception-ResNet-v2	4
2.3	Pytorch and the Torch-Image-Models library (TIMM)	4
2.4	ImageNet	4
3	Methodology	5
3.1	Data collection	5
3.2	Replicating Koalarization	5
3.3	Luminescence channel	5
4	Experiments	5
4.1	Training	6
4.2	Results	7
4.2.1	User study	9
5	Conclusion	11
5.1	Future work	11
6	Discussion	12
6.1	Automation	12
6.2	Population sample	12
6.3	Different metrics	12
6.4	Colourising historic photographs	12
6.5	Realness perception on original images	13
A	Appendix: Grading	15
A.1	Implementing vision-algorithms (row 7)	15
A.2	Use of literature (row 8)	16
A.3	Planning (row 9)	16
A.4	GIT (row 10)	16
A.5	Experiment (row 11)	17
A.6	Report (row 12)	17
A.7	Experiment reproducibility (row 13)	17

2 Background

In this section we give some background on this replication study. Namely, the structure of Federico Baldassarre, 2017 network and their experiment, including a description of Inception-ResNet-v2. Furthermore, we describe the modules and frameworks used in this experiment. Finally, we describe the ImageNet project, who's data was used in both this and Federico Baldassarre, 2017's experiment.

2.1 Original paper

2.1.1 Network architecture

Federico Baldassarre, 2017's Colorization network is built up from a combination of Convolutional Neural Networks² and Inception-ResNet-v2. Its input being the L* channel, and its output being the A*B* colour channels from the CIELAB standard.

The network is divided into four components:

1. Feature extractor³
2. Encoder
3. Fusion
4. Decoder

With the latter three subsisting only of CNNs and upsampling layers. The Encoder and Feature extractor components both take the L* channel as input. The Fusion component then combines the extracted high-level features from the Feature Extractor with the mid-level features from the Encoder. It then convolves over the embedding, and passes the result onto the Decoder component. The Decoder component convolves over this and then returns the A*B* colour channels.

2.1.2 Experiment

They trained this network using mean squared error as the loss criterion. This is, however, not how Federico Baldassarre, 2017 judged their model's results. Instead, the original paper tested their model's performance through a user study.

They present the user with a set of twelve images, nine recoloured images and three original images. Next, they asked to choose which images were fake or real.

According to their study, 45.87% of users miss-classified recoloured images as real. It is important to mention that the recoloured images used were selected from their best results.

²Hereafter referred to as CNNs

³Inception-ResNet-v2

2.2 Inception-ResNet-v2

Inception-ResNet-v2 is a neural network architecture for multi-label image classification. It is one of three architectures introduced in Szegedy et al., 2017. Inception-ResNet-v2 is a combination of the Inception architecture and "Residual Connections"⁴(He et al., 2015).

Inception-ResNet-v2 is trained on the ImageNet database (Deng et al., 2009). Therefore, it classifies 1000 different classes.

At the time of its release, it had the highest Top-5 accuracy on Papers with Code's⁵ Image Classification on ImageNet⁶ challenge with a score of 95.1%. Shortly after it was supplanted by ResNet-200, with a score of 95.2%. Today, on 2022/03/30, Inception-ResNet-v2 is the 102nd best performing model on the Image Classification on ImageNet challenge.

In this project, a pretrained implementation of Inception-ResNet-v2 was used. See 2.3 for more information about this implementation.

2.3 Pytorch and the Torch-Image-Models library (TIMM)

The original paper's implementation was made with the deep learning framework Keras-Tensorflow (Chollet et al., 2015).

However, for the sake of variety, this implementation was done in PyTorch (Paszke et al., 2019) instead. Like Keras-Tensorflow, PyTorch is a high performance deep learning framework.

Unlike Keras-Tensorflow, PyTorch does not come with an implementation of Inception-ResNet-v2. Torch-Image-Models⁷ by Wightman, 2019's implementation is used instead.

TIMM offers authentic implementations of various architectures in PyTorch. For some of the architectures, including Inception-ResNet-v2, pretrained models are also available.

2.4 ImageNet

The ImageNet project is a ongoing research effort for collecting and labelling image data for object recognition models. ImageNet (the database) uses labelling based on the WordNet lexical dataset (Fellbaum, 1998).

The ImageNet database is updated every few years. With the years 2012-2018 being combined into one larger database. Federico Baldassarre, 2017 uses the 2011 iteration of ImageNet. As this iteration is no longer available, the next iteration is used instead. That being the 2012-2018 iteration.

⁴i.e. ResNet

⁵A aggregator of scientific papers with source code. Also offers leaderboards for different challenges

⁶paperswithcode.com/sota/image-classification-on-imagenet

⁷Hereafter referred to as TIMM

3 Methodology

3.1 Data collection

As mentioned in 2.4, the data is collected from the ImageNet project. A different iteration of the database from the original authors is used for this project due to availability problems.

This experiment uses the subset of the ImageNet database sourced from Kaggle⁸

The validation set is ensured to be balanced by ImageNet. Ergo, it is assumed that a randomised subset of this data will be approximately as balanced.

Due to hardware constraints, the subset of images used for this experiment is smaller than that of the original authors. The original authors used 60 thousand images, this experiment only uses 45 thousand. The ratio of train-validation images is kept at 8:1, only slightly differing from the original’s ratio of 9:1.

The labels supplied by ImageNet are left unused, as the images are both input and target.

3.2 Replicating Koalarization

In order to replicate the Koalarization paper, a Python Module was developed. This module can be found at github.com/Casper-Smet/koalarization_torch. The module can be installed, and binaries are provided for every release.

As described in 2.3, this implementation was written using PyTorch. The behaviour of the original of the original implementation is replicated as closely as possible. In order to accomplish this, some modules must be implemented independently.

For example, the Conv2D layer in PyTorch does not support both ‘same’ type padding and the use of stride. Nor is there a function for resizing an image with padding, or converting an image CIELAB to SRGB and vice versa. These features were either implemented independently, or imported from TIMM.

3.3 Luminescence channel

Like mentioned in the introduction, unlike the original implementation this implementation takes the L^* channel from a grey-scaled version of the original image. The extraction of the L^* channel takes place in the dataloader module. Ergo, altering the code to reflect the original paper better would be trivial.

4 Experiments

In order to validate Federico Baldassarre, 2017’s experiment, we intend to repeat it.

⁸kaggle.com/c/imagenet-object-localization-challenge

Therefore, we will train a model on our dataset copying the original experiment’s parameters. Next, the trained model is used to recolour all images in the validation set. From these recoloured images, the most convincing ones are selected.

Using these recoloured images, a user study is created. The user is presented with nine recoloured images and three original images. Next, the user is asked which images are real and which are fake. The rate at which users miss-classify the fake images as real is called the realness perception.

The original paper collected about 41 responses. This paper collected 60 responses.

We define the following hypothesis:

$$H_0 : r_1 = r_2$$

Where H_0 is the null hypothesis, r_1 is the realness perception of the original experiment, and r_2 is the realness perception of this experiment.

This hypothesis will be tested using a two-sided t-test⁹.

If the p-value is found to be lower than the alpha value of 0.05, the H_0 is to be rejected. Otherwise, this test confirms Federico Baldassarre, 2017’s experiment.

4.1 Training

As mentioned in 3.1, the model is trained on 40 thousand images, with five thousand images being held out for use as validation data. All images colourised in this report are from the validation set unless specified otherwise.

All images are padded to be square and resized to 244x244. The resultant images then have their A*B* colour channels extracted and simultaneously are converted to grey-scale and have their luminescence channel extracted. The former serving as the model’s target, and the latter as it’s input. These values are also normalised to [-0.5, 0.5].

Just like the original experiment, the network is trained using the ADAM optimiser (Kingma and Ba, 2014) with a learning rate of 0.001. The loss function is Mean Squared Error, or MSE.

After every 100 batches, the running loss and some colourised images from the train set are uploaded to Tensorboard to keep track of progress. At the end of each epoch, the running loss is calculated over the validation set. This, along with some colourised images from the validation set are uploaded to Tensorboard.

By default, the Tensorboard logs can be found in `./log/koala/*DATE*`.

The model was trained for 40 epochs on a laptop¹⁰ with a NVIDIA GeForce GTX 1070, 16 GB of RAM. Using 8 workers for loading in the data, and a batch size of 50, this took approximately 13 hours.

Based on the angle of the loss curve in figures 1a and, more importantly, 1b, the model has not started over-fitting. Ergo, it might be worthwhile to train the model for a greater number of epochs

⁹See docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind_from_stats.html

¹⁰Asus ROG Strix GL702VS-GC198T

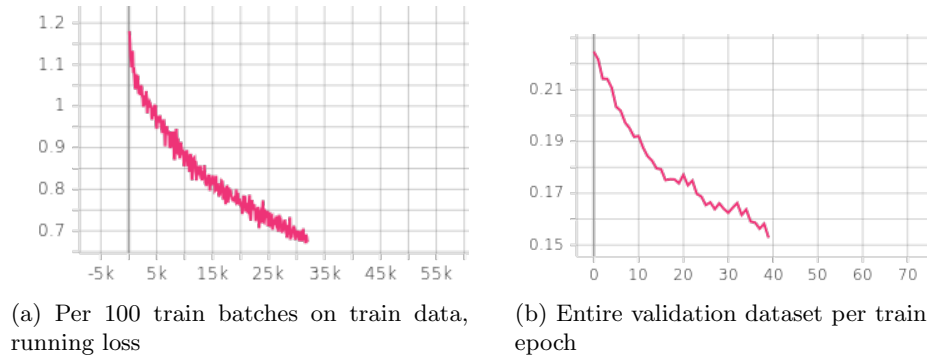


Figure 1: MSE-loss graphs during training. y -axis equals loss

The majority of these hyper-parameters can be changed using command line arguments. For user instructions, see github.com/Casper-Smet/koalarization_torch.

4.2 Results

After training the model on the train subset, the trained model is used to recolour the validation subset. This was done using the evaluation script¹¹

The resulting images appear highly similar to those of the original experiment. This includes the flaws described by Federico Baldassarre, 2017. Some images have been recoloured near-photo-realistically. However, the majority of the images are recoloured with some glaring artefacts, which are usually spots or slashes of bronze-grey.

¹¹See github.com/Casper-Smet/koalarization_torch/blob/main/koala/evaluate.py and the instructions in the README.md.

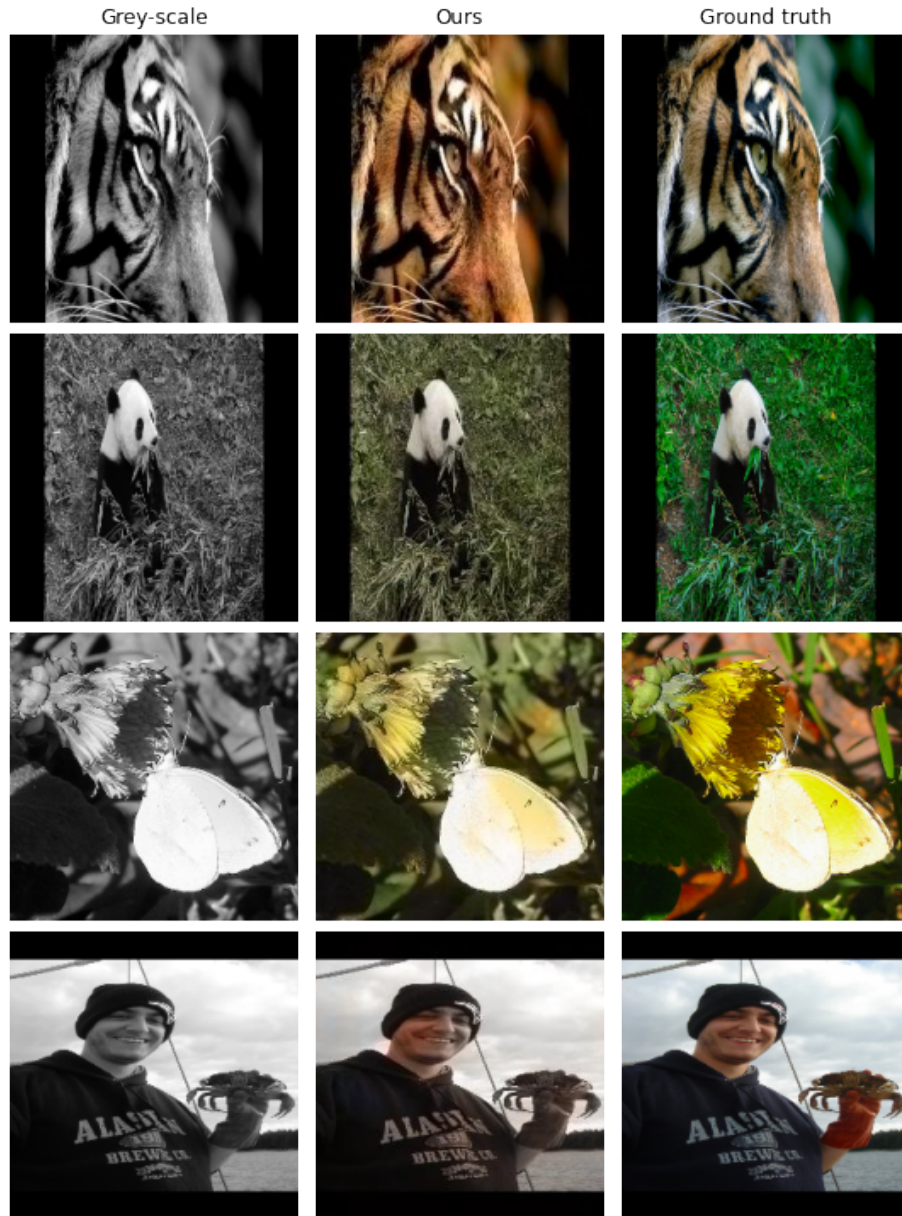


Figure 2: Comparisons between original (ground truth), grey-scaled and re-coloured images.

In figure 2, some of the most near-photo-realistic images are placed next to their grey-scale and ground truth versions. The first image is very realistic looking, yet clearly oversaturated in comparison to its ground truth. However,

this is a rare phenomenon. Furthermore, the background is coloured using orange, just as the tiger. The generated image is clearly different from the original, yet still believable.

The latter images show clear undersaturation, which is by far the more common problem found in the generated images. Specifically, the foliage on rows 2 and 3 are clearly undersaturated in comparison to their ground truth. This falls inline with the original paper’s results: the model seems conservative with its colour-use.

Finally, the model does not seem to perform well on human subjects. At most, parts of the subject skin are coloured correctly. Mostly, it leaves the subjects skin grey. The image in the last row is, in the author’s opinion, the best recoloured image with a human in it.

While this problem is not mentioned explicitly in the original paper, it is apparent when looking at their generated images. See figure 3.



Figure 3: Left coloured by Federico Baldassarre, 2017, right ground truth. Adapted from Federico Baldassarre, 2017.

4.2.1 User study

The user study was done using the Google Forms platform¹². The answers were analysed using Pandas in a Jupyter Notebook¹³.

See figure 4 for the nine recoloured images shown to the user, and how often the user was fooled by each individual image.

¹²Though the form is now closed, it could be found at forms.gle/mm8cPUM3ggne8E6dA.

¹³See github.com/Casper-Smet/koalarization_torch/blob/main/Userstudy.ipynb

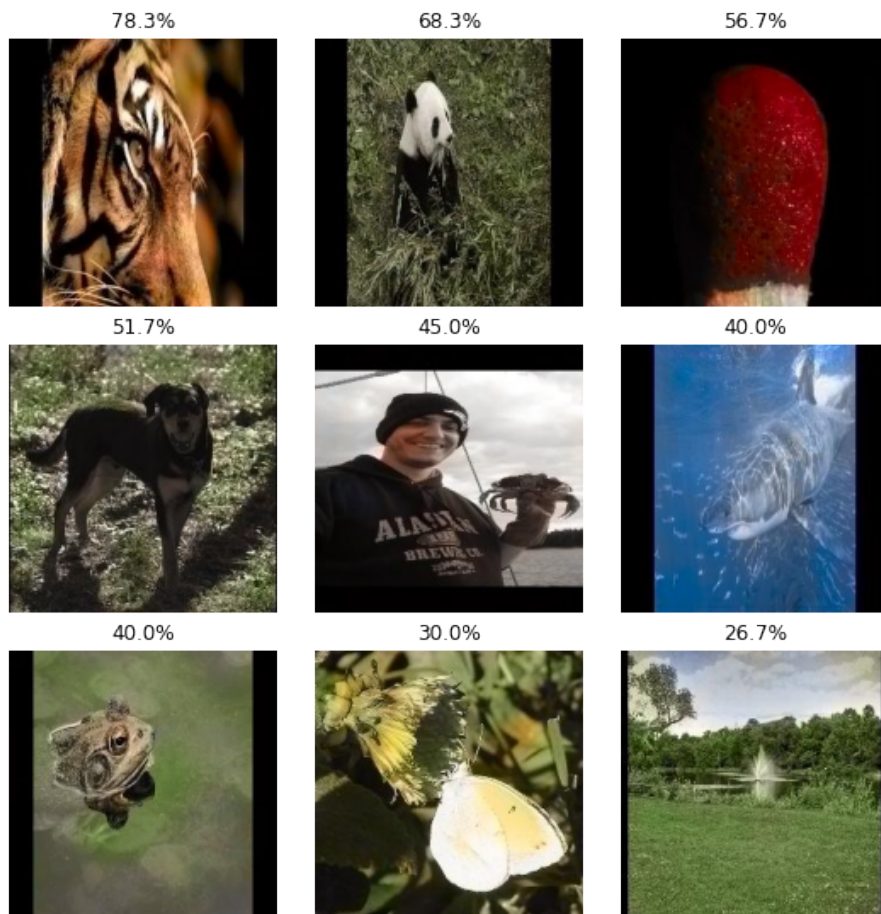


Figure 4: For each recoloured image we give the percentage of users that answered “Real” to the question Real or Fake? The images are sorted according to how often they fooled the user.

The results were similar, yet slightly better, than the original experiment. Four out of the nine images fooled more than half of the users, just like the original experiment. Three other images fooled nearly half of the users, ranging between 40% and 45%. Lastly the last two images fooled slightly more than a quarter of the users.

The lower scoring end of the recoloured images score better than that of the original experiment’s counterparts.

The original paper fooled the user 45.87% of the time, with a standard deviation¹⁴ of 22.36%. This experiment outperformed in both, fooling the users 48.5% of the time, with a standard deviation of 17.1%.

¹⁴Standard deviation of the scores [79.0, 69.1, 64.1, 54.3, 42.4, 35.9, 35.1, 17.9, 15.0]

These values are further processed in the aforementioned Jupyter Notebook. The t-test yielded a p-value of 0.493. $0.493 > 0.05$, ergo, the null-hypothesis cannot be rejected. The slightly greater realness perception and smaller standard deviation are not statistically significant.

5 Conclusion

Despite this experiment’s changes to how the luminescence channel was procured, we hypothesised that the results of this experiment would not differ significantly from Federico Baldassarre, 2017. We believed this to be the case based on the fact that the L^* channel is not visually distinguishable when extracted from grey-scale than when its from the coloured image.

As expected, the results from this experiment are approximately equal to the original experiment. This result validates Federico Baldassarre, 2017’s experiment.

This experiment also reconfirms the weaknesses of the Koalarization architecture. Namely, that this network is not performant in colourising low-level image components like human subjects or dogs. It also confirms that most resulting images are undersaturated.

This experiment also adds value outside of its validation purposes. As this implementation was written in PyTorch, as opposed to the original’s Tensorflow-Keras. Thus, diversifying the Koalarization architecture’s user base. Furthermore, this implementation is well-documented and modular. Ergo, it is highly adaptable.

5.1 Future work

Firstly, we believe it to be prudent to train this model on the greater ImageNet database. 45 thousand images may seem like a lot, but consider that the ImageNet database contains one thousand classes. In a perfectly balanced mono-label¹⁵ dataset, that amount to only 45 images per class.

Training on a larger dataset within a realistic time-frame would require greater computational ability. Future research should make use of compute clusters instead of home hardware.

Secondly, the effects of the feature extractor should be examined, possibly through an ablation study. It would be interesting to see how it affects the colourisation of images.

Thirdly, as Inception-ResNet-V2 is quite old, it would be interesting to see how a newer more performant feature extractor would impact this architecture’s performance. For example, Floirence-CoSwin-H (Yuan et al., 2021), which is, at this time, the highest scoring model in Top-5 accuracy on the Image Classification on ImageNet challenge.

¹⁵As the ImageNet database is multi-label, it is hard to say how the classes are balanced. See 2.4

Finally, it would be worth expanding the user study with a greater and more varied number of images, both recoloured and original. In its current state, all users were presented with the same 12 images, though in a random order. Increasing the number of images presented to the users, and randomising the images shown, may give a more objective view on realness perception.

6 Discussion

6.1 Automation

The user study may give a false impression of this model’s performance. This is because the cherry picked colourised images do not fully represent the whole of this model’s performance.

While the results of this study are good, it is worth noting that full automation is a use case for which this model is simply not good enough. The model’s colouring is simply too inconsistent for this use case.

The model may, however, still be useful in manual colourisation efforts, as it could give a starting point from which people may continue recolouring an image. This might help speed up the labour intensive process of manual colourisation.

6.2 Population sample

The population under which this form was spread is largely Technical Informatics and Artificial Intelligence students at Hogeschool Utrecht. These users, especially the latter, could be better equipped to find mistakes in colourisation efforts versus the average consumer. This can be attributed to their profession.

6.3 Different metrics

While the user study is a clear way to judge the models performance in the eyes of its users, it is difficult to use as an objective metric. Ergo, it is also difficult to compare different colourisation models in performance with this metric.

Comparing this implementation with the Federico Baldassarre, 2017’s implementation can be done by comparing the MSEloss on the validation set. To compare Deep Koalarization with other models, a standardised metric is in order.

The metric currently used on the ”Colorization on ImageNet val” leaderboard on PapersWithCode is Fréchet inception distance (Heusel et al., 2017), or FID in short. Future research might include this metric.

6.4 Colourising historic photographs

Real historic photographs are not always in perfect grey-scale. Often, these images are presented as sepia toned¹⁶. Using grey-scaled versions of these images instead of their original sepia versions may cause data loss.

¹⁶See en.wikipedia.org/wiki/Photographic_print_toning#Sepia_toning

6.5 Realness perception on original images

On interesting metric the original paper discarded, was the realness perception of the original images. See 5 for the scores of each individual image. The realness perception of the original images is 66.67%. The standard deviation in realness perception is 24.7%. In about one third of the cases, an original image is mistaken for a recoloured image.

However, with a sample size of original images this small, it is difficult to make any conclusions based on this information. This adds further need to expanding the user study as mentioned in 5.1.

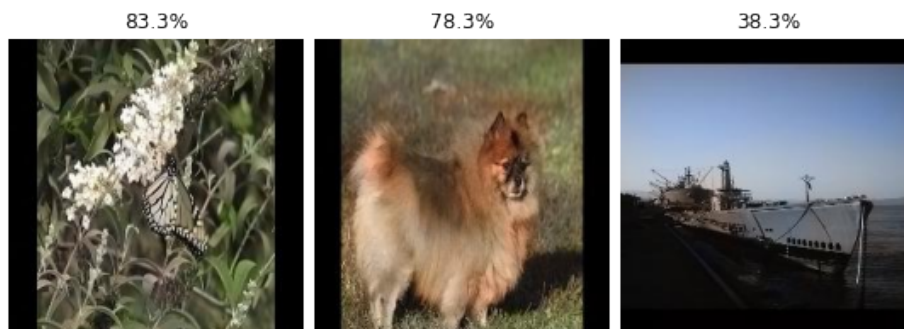


Figure 5: For each recoloured image we give the percentage of users that answered “Real” to the question Real or Fake? The images are sorted according to realness perception.

References

- Chollet, F. et al. (2015). *Keras*. <https://github.com/fchollet/keras>
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, 248–255.
- Federico Baldassarre, L. R.-G., Diego Gonzalez-Morin. (2017). Deep-koalarization: Image colorization using cnns and inception-resnet-v2. *ArXiv:1712.03400*. <https://arxiv.org/abs/1712.03400>
- Fellbaum, C. (Ed.). (1998). *Wordnet: An electronic lexical database*. MIT Press.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. <https://doi.org/10.48550/ARXIV.1512.03385>
- Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. <https://doi.org/10.48550/ARXIV.1412.6980>
- Luo, M. R. (2014). Cielab. In R. Luo (Ed.), *Encyclopedia of color science and technology* (pp. 1–7). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-27851-8_11-1
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. *Thirty-first AAAI conference on artificial intelligence*.
- Wightman, R. (2019). Pytorch image models. <https://doi.org/10.5281/zenodo.4414861>
- Yuan, L., Chen, D., Chen, Y.-L., Codella, N., Dai, X., Gao, J., Hu, H., Huang, X., Li, B., Li, C., Liu, C., Liu, M., Liu, Z., Lu, Y., Shi, Y., Wang, L., Wang, J., Xiao, B., Xiao, Z., ... Zhang, P. (2021). Florence: A new foundation model for computer vision. <https://doi.org/10.48550/ARXIV.2111.11432>

A Appendix: Grading

A.1 Implementing vision-algorithms (row 7)

For the purposes of this assignment, I implemented one algorithm. Namely: the Deep Koalarization image colourisation Neural network.

The implementation is truthful to the original implementation, with the exception of one issue in the original implementation which I have remedied. This issue is technically not part of the architecture itself, but how data is prepared. See 3.3.

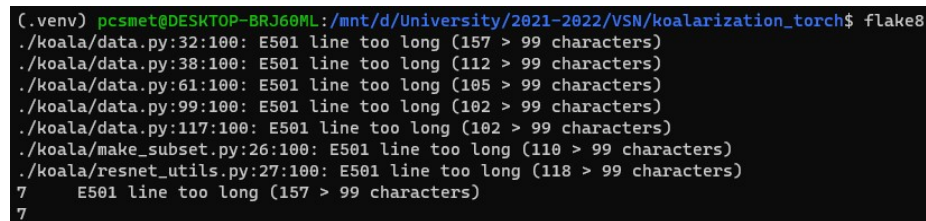
Furthermore, implementing this algorithm required a number of external libraries, namely:

- PyTorch, for basic machine learning functions
- Torchvision, for image manipulations
- PyTorch Image Models, for their implementation of Inception-ResNet-v2
- pytorch_colors, for colour space manipulations, namely CIELAB
- Scikit Image, for colour space manipulations, namely CIELAB

These dependencies can be found in github.com/Casper-Smet/koalarization_torch/blob/main/setup.cfg along with other utility dependencies used during data preparation, training, and validation.

As PyTorch was not covered in class, using it required reading up on its literature. In addition to PyTorch, extra literature was required for implementing the CIELAB colour space, Inception-ResNet-v2 model, and the ImageNet database.

Additionally, all code is documented. This includes class and function level docstrings. All docstrings follow the Google docstring standard. All code also fulfils the Flake8¹⁷ standard, which is an expanded version of the Pep8 standard. The only exceptions to this rule are the Path variables, as they are often too long. Flake8 was configured locally in the `.flake8` configuration file.



```
(.venv) pcsmet@DESKTOP-BRJ60ML:/mnt/d/University/2021-2022/VSN/koalarization_torch$ flake8
./koala/data.py:32:100: E501 line too long (157 > 99 characters)
./koala/data.py:38:100: E501 line too long (112 > 99 characters)
./koala/data.py:61:100: E501 line too long (105 > 99 characters)
./koala/data.py:99:100: E501 line too long (102 > 99 characters)
./koala/data.py:117:100: E501 line too long (102 > 99 characters)
./koala/make_subset.py:26:100: E501 line too long (110 > 99 characters)
./koala/resnet_utils.py:27:100: E501 line too long (118 > 99 characters)
7      E501 line too long (157 > 99 characters)
7
```

Figure 6: flake8 linter ran against the code base

¹⁷See flake8.pycqa.org/

A.2 Use of literature (row 8)

In order to implement the Deep-Koalarization architecture, I had to read and process a number of different papers, articles and books. The most prominent of which I used as sources for this document¹⁸. Moreover, I documented some of these personally in chapter 2.

Sources are also referenced in the code, both explicitly and implicitly. For example, see the `SquarePad` class in `koala/data.py` for an explicit reference.

A.3 Planning (row 9)

My original planning was reasonably accurate, not including the period of illness. Furthermore, in case I underestimated the assignment I added optional tasks.

Through week updates, changes in the planning on GitHub¹⁹, and in person progress reports I kept the professor up to date.

After a period of illness, I constructed a new planning via email that accounted for my two weeks of absence. This new planning, including deadline, was accepted.

A.4 GIT (row 10)

Both the professor and the Teaching Assistant were added to the GitHub repository. My commits were regular, despite the period of illness. See figure 7.

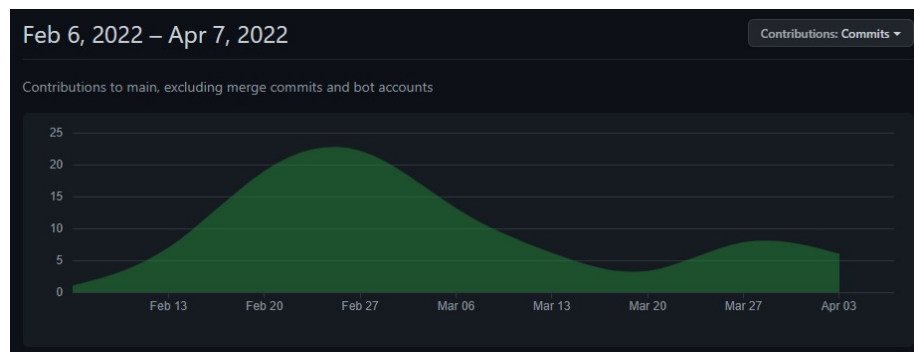


Figure 7: Git commit history for the git repository `koalarization_torch`. Taken on 2022/04/07: 14:30

Furthermore, at the end of most workweeks, I created a release. These releases contain a binary in the form of a Python wheel²⁰. The 1.0.0²¹ release also contained a trained version of the Colorization network.

¹⁸See bibliography

¹⁹See github.com/Casper-Smet/koalarization_torch/blob/main/report/PLANNING.md

²⁰See peps.python.org/pep-0427/

²¹See github.com/Casper-Smet/koalarization_torch/releases/tag/v1.0.0

A.5 Experiment (row 11)

The experiment, its parameters, and its results are well defined in 4. The experiments metrics, namely the MSE for training and the realness perception for validation are defined. These metrics are also interpreted using figures such as 1b and 4.

The uncertainty of realness perception is judged using a two-sided t-test. This ensures that the statistical significance of this experiment's result is taken into consideration when interpreting it.

Finally, the author is critical of the metrics chosen to judge the Koalarization architecture. Most of this critique can be found in chapter 6.

A.6 Report (row 12)

This report follows the structure found on Canvas. Furthermore, it is written for students who have followed the *VSN* course at Hogeschool Utrecht. This includes the expectation of them knowing the basics of machine learning and CNNs. Materials that students are not likely to have encountered at this point were either explained or supplied with a source.

All references are available in APA format.

Moreover, the hypothesis made in the introduction, and its numerical derivative in 4 are well formulated to be proven or disproved. The Conclusion and Discussion chapters also describe possible future work.

A.7 Experiment reproducibility (row 13)

All the scripts necessary for replicating this experiment can be found in the GitHub repository. Furthermore, it is clearly described where the data used in this experiment can be found. Finally, clear instructions are given on how to use these scripts, including (hyper)parameter selection. See the README.md for more information.

Furthermore, Inception-ResNet-V2 can easily be replaced by another feature extractor without altering the code, as it is simply an argument the user can give when instantiating the Colorization network.

Moreover, due to the nature of PyTorch as a machine learning framework, it would be easy to extend this code to ones own wishes. Be that subclassing the Colorization module or its submodules, writing your own dataset instance, or using a different optimiser.