Colorization of grey-scale images

Alexander Olsson
aleol711@student.liu.se

Erik Berglund
eribe627@student.liu.se

Erik Kronberg
erikr271@student.liu.se

Joar Måhlén
joama595@student.liu.se

Lovisa Stålebrink
lovst419@student.liu.se

Ludvig Fors
ludfo119@student.liu.se

1. Introduction

This project takes on the challenge of colorizing black-and-white images. Film and image colorization is an old concept that has existed since the early 20th century. Historically this practice has been done by hand. Modern technology has today expanded the possibilities around what is achievable dramatically.

Image colorization is a difficult area often studied within the field of Computer Vision. Modern approaches to this problem include deep learning models trained for this specific task, to colorize black-and-white images. Deep learning models are typically trained to colorize images by utilizing typical semantic meanings often found in images. For example, generally the sky is blue, a strawberry is red and grass is green. However the semantics does not apply in all situations, for instance a rose might typically be red but in reality its color can vary a lot. Also there are many colors that share the same grey scale value, meaning that a grey scale image could therefore be colored in various ways. For these reasons image colorization is considered to be an undetermined problem.

To handle this problem the goal of image colorization is not necessarily to recover the ground truth color of the image. Often the goal is instead considered to be to colorize an image with possible colors that could possibly fool a human. In order to evaluate the result of a colorization model a Turing test is a common practice. In the Turing test human participants are shown two images, one having ground truth colors and the other has been colorized. The participants task is then to guess which the fake image is.

There are several different techniques that can be utilized to solve the image colorization problem. According to the paper *Review on Different Methods of Image Colorization* all the solutions to the colorization problem can be generally categorized as one of the following, Manual approach, Scribble based approach, Example-based approach and Learning-based approach [7]. Furthermore, the paper states that most of the solutions today are in a fully automatic manner using deep learning methods. This paper will dive deeper and examine two different methods to colorize the images, where both methods have a learning-based approach. The first one uses a Conditional Generative Adversarial Network, cGAN, and the other method using a Convolutional Neural Network, CNN. Both of the previously mentioned solutions to this problem are well researched and many good examples of both CNN and cGAN models exist that generate very realistic colorized images [4][9].
With the evidence from the previously mentioned papers, it is expected that both models will have the possibility of generating realistic colorized images from black and white images.

2. Methods

This section will give a more in-depth description of two methods used in solving the specified task of colorizing images. Both methods utilize neural networks to solve the task and the project uses TensorFlow [1] for building the models.

2.1. Conditional Generative Adversarial Network

One of the methods that were implemented in order to colorize the images was *Conditional Generative Adversarial Network*, or cGAN for short.

2.1.1 cGAN - Theory

Generative Adversarial Networks was first introduced in 2014 by Goodfellow in the paper *Generative Adversarial Networks* [5]. The architecture describes two neural networks training together to generate new images, the *Generator* and the *Discriminator*. Given random noise z the Generator G generates an image $G(z)$ and the Discriminator D classifies images x as real from the data set or fake, generated by the Generator, $D(x)$ see figure 1.

In order to train the model a value function that utilizes both networks and can be optimized is needed. Since D outputs a single scalar representing the probability that the input image is from the real data set, the model is trained to maximize $D(x)$ when x comes from the data set. Simulta-

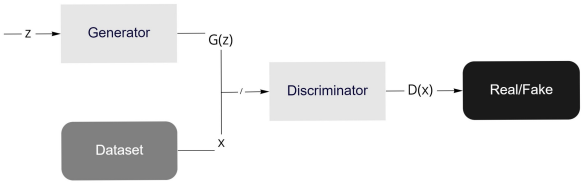


Figure 1: Illustration of the GAN architecture.

neously G will be trained to maximize $D(G(z))$. That way generated images will be trained to look like real images. Which is described in Goodfellow value function [5].

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]. \quad (1)$$

The value function in equation 1 is used as the loss function in the training.

The GAN structure is not enough. This projects ambition is to colorize a given image. Therefore the Generator G needs more than random noise z as input, it also needs to use a grey-scaled image to generate colors on. Which is were a conditional GAN (cGAN) can be used.

Conditional Generative Adversarial Networks was introduced in 2014 by M. Mizra in the paper *Conditional Generative Adversarial Nets* [10]. Mizra expanded Goodfellows GAN idea to enable both the Generator G and Discriminator D to be conditioned by some type of extra information y. The conditioning is done by feeding the information y to both G and D resulting in the following modified value function.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|\mathbf{y})] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|\mathbf{y})))]. \quad (2)$$

In this project the color space used is the *YUV*-format which consists of three channels (filters) where the Y-channel is the gray-scale image and the U and V channel describes the colours [11], and equation 2 is used as the loss function with the following representation:

- y grey-scaled image, light channel (Y).
- x the two color channels (UV) in a YUV color representation.
- z random noise with same dimensions as y.

Clarified in figure 2.

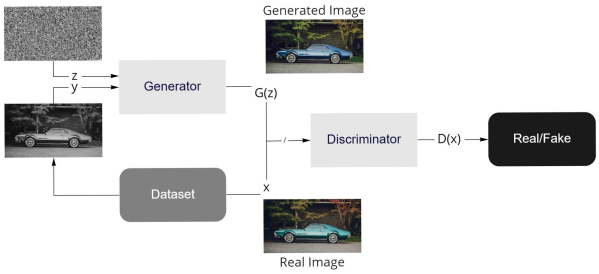


Figure 2: Illustration of this project cGAN model.

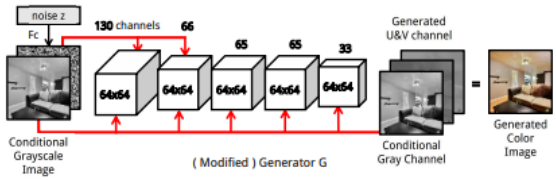


Figure 3: Illustration of the generator architecture. Credit to [4].

2.1.2 cGAN - Architecture

The architectures of the generator G and the discriminator D are mainly inspired by the paper [4].

The generator G has the architecture as in Figure 3. The network inputs a gray-scale image y of size $64x64$ and a noise vector z of dimension 100. Both y and z contain normalized values between -1 and 1. The noise vector is first mapped to a $64x64$ image Z via a fully connected layer which combined with the y forms the input to the next layers. The network consists of multiple $blocks$. Each block (except the last one) consists of a Convolution [3], Batch-Normalization [6] and ReLU [2] layer with the output shape $64x64$. During execution, the blocks output a predefined number of filters that also includes y and/or Z which are added as extra filters at the end of the block. For example: The first block has 130 filters including both y and Z so the number of channels that should be predicted from the Conv-BatchNorm-ReLU layers in the block is 128. The last block only consists of two layers. A convolutional layer and an $tanh$ activation function that scales the input to values between -1 and 1. After denormalization, the output of the last block is a colorized image in *YUV*-format.

The discriminator D has the architecture as in Figure 4. The input is represented in *YUV*-format, the same as the output of the generator. The blocks of the discriminator have a similar structure as the blocks in the generator. The only difference is that the strides of the convolution layers are set to 2 which down-scales the input in each block and that the activation function is changed from ReLU to LeakyReLU [12] with α set to 0.2. The output of the last

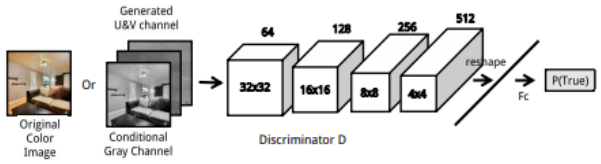


Figure 4: Illustration of the discriminator architecture. Credit to [4].

block, with 512 filters of size $4x4$, is reshaped into a single column vector. This vector is then fed into a fully connected dense layer with a $sigmoid$ activation function to get a value between 0 and 1.

2.1.3 cGAN - Algorithm

Algorithm 1 cGAN algorithm

1: **function** TRAIN_DISCRIMINATOR
2: Generate random minibatch of m sampled noise $\{z^1, ..., z^m\}$ each of size s_z.
3: Sample minibatch of m gray-scale images $\{y^1, ..., y^m\}$ each of shape $[s, s, 1]$.
4: Get coresponding minibatch of m real color images $\{x^1, ..., x^m\}$ from data distribution $p_{data}(x)$.
5: Optimize discriminator by minimizing:

$$\nabla_{\theta_d} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(y^{(i)}, z^{(i)})))] \tag{3}$$

6: **function** TRAIN_GENERATOR
7: Generate random minibatch of m sampled noise $\{z^1, ..., z^m\}$ each of size s_z.
8: Sample minibatch of m gray-scale images $\{y^1, ..., y^m\}$
9: Optimize generator by minimizing:

$$\nabla_{\theta_g} \sum_{i=1}^{m} \log(1 - D(G(y^{(i)}, z^{(i)}))) \tag{4}$$

10: **function** MAIN
11: Initialize the generator and discriminator networks
12: **for** number of epochs **do**
13: **goto** *train_discriminator*.
14: **goto** *train_generator*.

The cGAN algorithm used is described by Algorithm 1.

The optimizer used during training is the Adam optimizer [8] with learning rates α_G and α_D. Hyperparameters for the algorithm can be found in Table 1.

Info	Parameter	Value
Generator learning rate	α_G	0.00005
Discriminator learning rate	α_D	0.00005
Batch size	m	64
Noise vector dim	s_z	100

Table 1: Hyperparameters for cGAN.

2.2. Modified VGG16

The second method that was implemented is a modified VGG16 network and was introduced by Zhang, et al. in *Colorful Image Colorization* [14].

2.2.1 Modified VGG16 - Theory

The model requires pictures in the CIE Lab color space which consists of three channels *L*, *a* and *b*. The *L* channel reflects the intensity of a pixel where 0 is equal to black and 100 is equal to white. The *a* channel represents the red and green colors in the image, while the *b* channel represents the blue and yellow. Both channels have a value between $-128 \leq a, b \leq 127$. Figure 6 gives a good understanding of the color space.

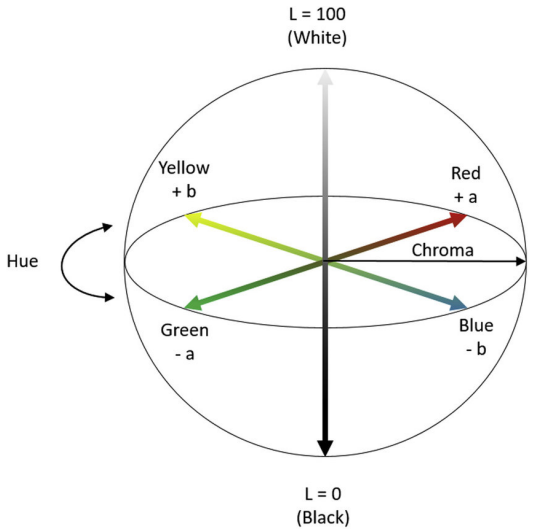


Figure 5: Lab color space. Credit to [9].

Given the *L* channel, $\mathbf{X}$, the objective is to learn a mapping, $\hat{\mathbf{Y}} = F(\mathbf{X})$, to the associated color channels *a* and *b*. This can be seen as either a regression or a multinomial

classification problem. Both problems can be solved by using a CNN, $\hat{\mathbf{Z}} = G(\mathbf{X})$. The regression approach is a fully end-to-end trainable solution, which predicts a pixel value for each pixel in the image. The mapping to be learnt here $F(\mathbf{X})$ is therefore equal to $G(\mathbf{X})$. The L channel is then added to the predicted *ab* channels, to construct the final output image. This model is used as a baseline for the second approach, where a probability distribution is predicted instead. A function $H(\hat{\mathbf{Z}})$ is then used to map the predicted distribution $\hat{\mathbf{Z}}$ to a point estimate $\hat{\mathbf{Y}}$ in the associated color space. This mapping is done outside of the CNN, which makes this solution not end-to-end trainable. The mapping $F(\mathbf{X})$ can then be said to be a composition of the CNN G, which produces a predicted distribution over all pixels, and the operation H, which generates a final prediction.

The regression approach CNN is evaluated with a L2-norm loss defined as:

$$L_{l2}(\hat{\mathbf{Y}}, \mathbf{Y}) = \sum_{h,w} (\mathbf{Y}_{h,w} - \hat{\mathbf{Y}}_{h,w})^2$$

The loss function like the Euclidean loss is in this case inadequate. It is not robust to the inherent ambiguity and multimodal nature. In color prediction, this averaging effect of the set of distinct *ab* values will generate desaturated results, which can result in brownish output images. Therefore, it is better to treat the problem as a multinomial classification. The second approach which is a combination of a CNN and a multinomial classification is evaluated with a multinomial cross-entropy loss. This loss is defined accordingly:

$$L_{cl}(\hat{\mathbf{Z}}, \mathbf{Z}) = -\sum_{h,w} \sum_{q} \mathbf{Z}_{h,w,q} \log(\hat{\mathbf{Z}}_{h,w,q}).$$

In figure 6 the *ab* color space can be seen divided into 313 squares. These squares are called bins and this representation of the colour space is needed to be able to treat the problem as a classification problem. The amount of bins affects the result and 313 was chosen since Zhang, et al.[14] used it in their experiments. The authors described their choice of 313 bins as it gives a quite large amount of different colours while keeping the limits of each bins quite simple. No experiments with other values was presented, however. As stated before, the model will favour colours in the middle of the spectrum if an euclidean loss is used since those colours are quite close to all other colours. This can be seen in figure 7 which shows the probability distribution of colours in the data set used by Zhang, et al [14].

In the final stage of the classification approach, the bin with the highest probability for a specific index is used to colorize the corresponding pixel. This can be represented by the following equation

$$H(\mathbf{Z}_{h,w}) = \arg\max_{q \in Q} \mathbf{Z}_{h,w},$$

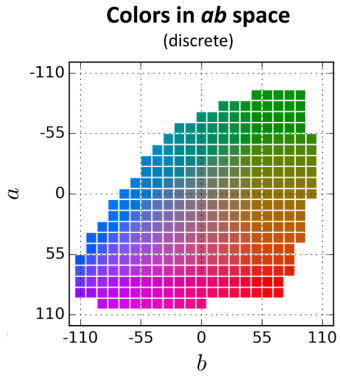


Figure 6: Colors in ab space discretized. Credit to [14].

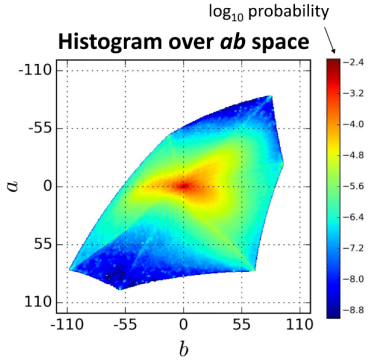


Figure 7: Probabilities of ab color space in log scale. Credit to [14].

where $\mathbf{Z}_{h,w}$ is a vector containing the probabilities for each bin. The output is thus an index which in the next step is transformed to the corresponding a and b values. As for the regression approach, the L channel is then added to the predicted *ab* channels.

2.2.2 Modiefied VGG16 - Architecture

The CNN that is used in the regression model is illustrated in figure 8. It takes a black and white image (the L channel of a CEI Lab image) as its input and then sends it through multiple convolutional layers, illustrated with a block in the figure. These blocks contains 2 or 3 convolutional layers with a Rectified Linear Unit (ReLU) as its activation function, followed by a batch normalization. The network does not contain any pooling so all spatial resampling is done between the different blocks.

In the article by Zhang, et al.[14], only the classification architecture was presented. For the regression model to be able to work properly, some modifications had to be made to the network. The output layer was changed to output two values representing the *ab* channels. The activation function

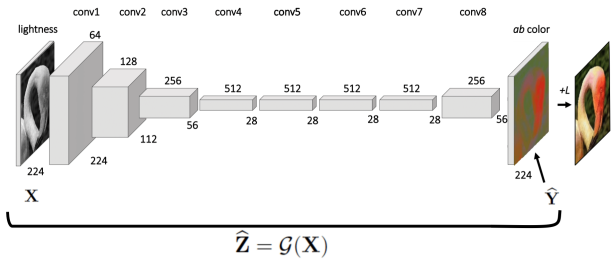


Figure 8: Illustration of the modified VGG16 baseline architecture.

in the output layer was changed from *ReLU* to *tanh*, since the *ab* channels can be both negative and positive. The *L* channel was normalized by dividing it by 100 to get values between 0 and 1. The *ab* channel was normalized by dividing it with 128 in order for all values to be between -1 and 1. The presented architecture is also used as a baseline for the multinomial classification model.

In figure 9 the architecture of the multinomial classification approach is illustrated. Compared to the baseline only one block is added, which has a size of 56x56x313. This block is the output of the network and this is where the probability distributions are predicted. As described earlier in the theory, the bin with the highest probability is chosen for colorizing the corresponding pixel. Since the features in this block only have a size of 56x56, the output of the mapping needs to be up-scaled with a factor 4 to get the initial size of the image.

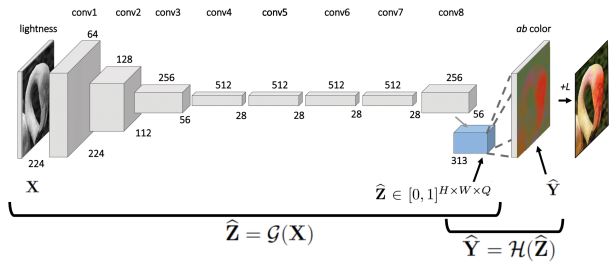


Figure 9: Illustration of the final modified VGG16 architecture [14].

To be able to decrease the computational time in the training stage, pre-trained weights are used in the layers up to and including block $conv4$ in the network. The pre-trained weights are trained for image classification on a subset of the ImageNet data set containing 1000 categories and over 1.2 million images. Pre-trained weights are only added up to the point were the architecture of the network is the same as the one the weights have been trained on. To avoid removing any useful information the pre-trained weights are frozen during training. The pre-trained weights was generated from a VGG16 network which takes three channels, R-G-B, as its input. The modified VGG16 network only takes one input channel, so it needs to be extended to be able to take three. Since a black and white image only is represented by one channel, three identical lightness channels will be sent to the network. To get the most out of the pre-trained weights the network will only be fed images of size 224x224 since that is the size the weights initially were trained on.

Table 2 illustrates the different hyperparameters used for the architecture. These where combined with the Adam optimizer in the training stage.

Parameter	Value
Regression learning rate lr	10^{-4}
Classification learning rate lr	3×10^{-5}
Batch size	64
Input size	224x224

Table 2: Hyperparameters.

2.3. Description of data set

Several data sets were used during development and evaluation of the models.

2.3.1 LSUN: Dinning Room

The LSUN: dinning room [13] data set consists of 657,571 coloured images of different sizes. The data set contains images of dinning rooms of different styles. Because of memory limitations, only 5% of the images where used during training. Preprocessing was done before training by scaling each image to $64x64$ images. This data set will be used to train and evaluate the cGAN model.

2.3.2 Linnaeus 5

The Linnaeus 5[1] data set consists of 1600 256x256 coloured images for each of its five classes. There are downsampled versions of the data set with images of sizes 128x128, 64x64 and 32x32. The Linnaeus data set was used mainly with 256x256 and 128x128. The modified VGG16 models will both be trained and evaluated on Linnaeus 5 data set.

2.4. Description of evaluation protocol

To evaluate the performance of the two models produced by the different methods a Turing test will be conducted as described in the introduction. The test will consist of 15 randomly sampled colorized images from each model together with the ground truth image. The test will be held through

[1] http://chaladze.com/l5/

an online survey and the number of participants will therefore depend on the amount of responses the survey receives.

3. Results

In this chapter the result from both models are presented.

3.1. cGAN

Below the results from the Conditional generative adversarial network is presented.

3.1.1 Loss

The idea behind Generative adversarial networks is that both models should counteract each other, meaning that the two models should balance each others loss. When one model is good with a low loss the other should have a higher loss and therefore make a larger change to move its loss back down (and the first mentioned model up). The model produced in this project have not succeeded in accomplishing this phenomena. As after 1000 iterations the discriminators loss proceeds constantly moving downwards, while the generator start moving upwards, but the longest training session for the model was only about 3000 iterations.

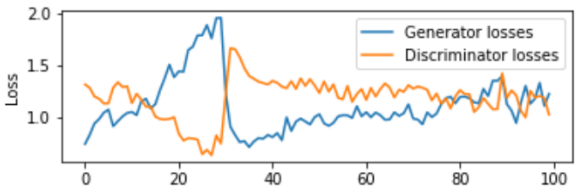


Figure 10: cGAN loss graph. Orange line displays the discriminators loss, the blue the generators loss. Each x represents 10 iterations.

In figure 10 the phenomena of that the two models battling each other is present. For instance at iteration 300 the generator has a high loss, the generator drastically changes yielding a large improvement and the loss of the generator drastically decreases while the discriminators loss increase. However in this graph a form of early stopping has been applied based on previous training.

The early stopping was set at 1000 iterations, after that the generator loss is constantly increasing, yielding worse and worse colorizations. Given that the batch size was set at 64, the model has been trained on 64 000 images which is around two epochs of the data set.

3.1.2 Accuracy

The accuracy of the discriminator is clearly correlated with both the generator and the discriminator loss, as expected.

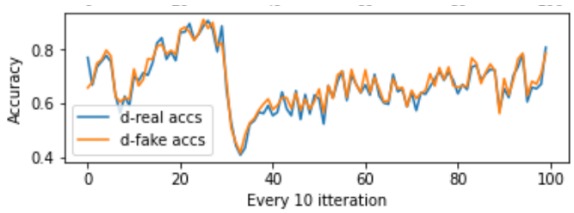
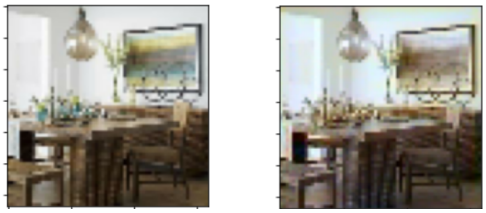


Figure 11: cGAN discriminator accuracy graph. Orange line displays the accuracy of classifying fake images as fake, the blue real images as real. Each x represents 10 iterations.

In figure 11 it is clear that the discriminator is equally good at classifying real images as real and fake as fake. The correlation between the loss can be seen through the movement of both graphs. For instance when the accuracy has a clear decrease is at the same position where the generator loss decreases and the discriminator loss increases. Before the early stopping the accuracy of the discriminator oscillates between 60 to 70%.

3.1.3 Images

Below images produced by the generator after early stopping is presented.



(a) Real image (b) Colorized image

Figure 12: Successful colorization with cGAN.

From figure 12 there is clear that the model is able to understand where outlines of objects (e.g. furniture) and colorize them into a specific realistic color. The painting on the wall of the images manifest the fact that the generator does not try to duplicate the real image. Instead the generator attempts to colorize the image to something ”realistic” in order to fool the discriminator. Leading to that both paintings have different colors. But it is this fact that can cause issues for the generator as well.

On the left side of figure 13 there is a window as seen on the left/real image. Outside there is a blue sky and some of the blue light shines through the window. However, the generator can not understand this and will therefore not colorize the sky blue and colorize the light from the window. This results in the fact that a human easily can see which of the images is real and which is fake. Similar effects in

(a) Real image (b) Colorized image

Figure 13: Unsuccessful colorization with cGAN.

images creates an issue for the generator in order to make an image that can trick humans. The images generated from the model was however never evaluated by other people outside of the group so it is unclear how well the model would perform in an Turing Test for an example.

3.2. Modified VGG16

In this section the results from the modified VGG16 model will be presented. The results from baseline regression model and the multinomial classification model are presented separately.

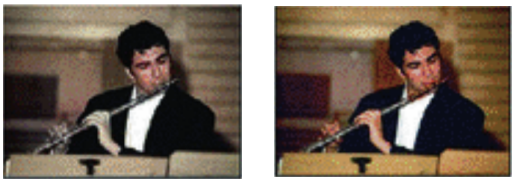
Common for both the baseline and the multinomial model is that they are only trained on the berries category of the Linnaeus 5 data set. This was due to the training time of the VGG16 model being very time consuming.

3.2.1 Regression

Figure 19 presents a few examples of colorized images obtained by the baseline model. Figure 18 presents the corresponding ground-truth images. When looking at the colorized images it is clear that the model is struggling with the colorization. When the images contain only berries and there are not much distracting colors in the background the model works best. The overall color of the image is plausible for the those images although minor inaccuracies are easily spotted when comparing to the ground truth images.

The colorization of the model is worse on images were the object is surrounded by a more lively and colorful background, noticeable on the first two images. The model seem to have difficulties understanding shapes and outlines of certain objects causing the color of the object to spread out from the outlines of the objects.

The overall result from the baseline colorization is poor and it is easy to determine which image is the ground-truth one when comparing two images. For this reason and for the lack of time a Turing test to evaluate the model performance was deemed unnecessary. Figure 14 and 15 displays the corresponding results using regression by Zhang et al [14]. It is clear that their model are doing a better job following the outlines of the objects. As mentioned previously the L2-norm loss function tend to lead to desaturated colors which is evident in their results while not as noticeable in our results.



(a) Colorized image (b) True image

Figure 14: Successful colorization using regression by Zhang et al. [14].

(a) Colorized image (b) True image

Figure 15: Unsuccessful colorization using regression by Zhang et al. [14].

3.2.2 Regression loss

Figure 16 displays the loss graph for the baseline VGG16 model. To avoid overfitting early stopping was used by ending the training when the test error started to increase. Judging by the graph it is evident that model is learning and that it performs better for every epoch until the loss converges for the test data set. The training is stopped after approximately 35 epochs, after that the model starts overfitting leading to test error increasing and the model producing worse colorizations.

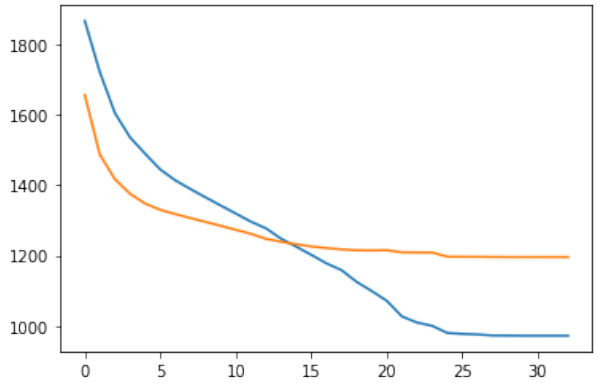


Figure 16: Regression loss graph. Orange line displays the test error, the blue the training error.

3.2.3 Multinomial classification

In figure 20 predicted images from the multinomial classification approach is shown. It can be seen that the model struggles even more to colorize the images than the regression approach. Large parts of the different images are colorized as grey even though the true image is very colorful. Another behavior that seems to be recurring is that the colors red and green are predicted very frequently. By looking at the images it seems like the model has a hard time to find contours as well. This behavior results in large regions with the same colors. As for the regression model, the images predicted from the multinomial classification model can not trick a human to believe that it is the ground-truth image. Therefore, no Turing test where carried out for this model neither.

3.2.4 Multinomial classification loss

The multinomial classification model was trained for 250 epochs and the loss for each of those is shown in figure 17. In the first couple of epochs the loss decreases significantly. After this it drops with a value between 3-7 for each epoch, which shows that the training require a lot of time to reach a model with a low error score. By looking at the graph we can see that the loss tends to continue to decrease if more training could have been carried out. The test loss is also a little lower than the train loss, which shows that the model is not overfit to the training data set.

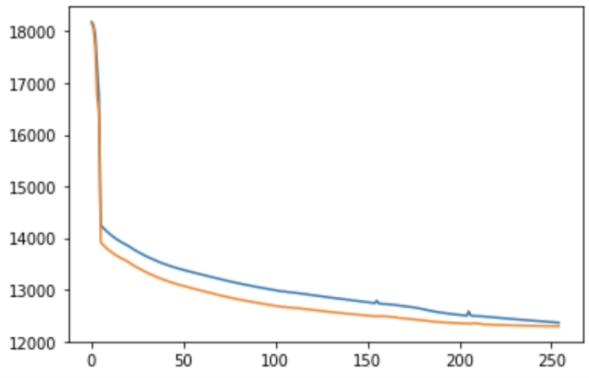


Figure 17: Multinomial classification loss graph. Orange line displays the test error, the blue the training error.

4. Discussion

In this chapter the result for the different models are discussed. The performance of the models are compared to the result obtained by the scientific papers together with reasoning about why they might differ.

4.1. cGAN

Even though the cGAN-model was only trained for two epochs the results generated were mostly all realistic. Some of the generated images were still quite poor in quality and it performed worse than the model from the paper of which it was based on, though this was expected as the time frame and access to hardware was limited for this project.

As the cGAN-model only aims to generate an image that is realistic and does not try to replicate the original image this works well for the training set selected for the model. This is because of the fact that for an example a chair or a table, which often can be found in the LSUN: Dining Room data set, can be many different colors but if the image instead was on something more color-specific, problems would quickly occur.

4.1.1 Selection of data set

It is always important to be studious when choosing a data set for when training a neural network. For an example if the selected data set has many images of an outside setting it is likely that the neural network quickly will learn that the upper part of an image is blue, because of the sky, and the lower part of the image is green, because of the ground. So for this project the focus was on learning how to colorize images of an inside setting where images are more diverse. In the beginning of the project we first used the LSUN: Classroom data set but was later changed due to the fact that the images in LSUN: classroom could vary quite a bit from image to image, thus making it hard for the generator to get a hold of what kind of image to create to due to vast diversity. One other reason is that the LSUN: Classroom data set often has humans in the images which seemed to be harder for the neural network to colorize. With this said, it is not known how well the cGAN-model in this project would perform on images with an outside setting, a more diversified data set or images with humans in it. However some evidence of that it will perform poorly can be seen in Figure 13 where the generator fails to colorize the sky blue. This makes the cGAN in this report quite niche and this should be kept in consideration when looking at the generated images.

4.1.2 Limitations in training

As mentioned in section 3.1.1 an early stopping was implemented at 1000 iterations to avoid the generators loss to increase to much. Due to limitations of Google Colab the longest we trained our cGAN model was around 3000 iterations and it could be that the poor quality of the generated images after 1000 iterations was due to the generators loss function being stuck in a local minima. Perhaps if the training would have kept going for more epochs it is possible

Figure 18: Real images.

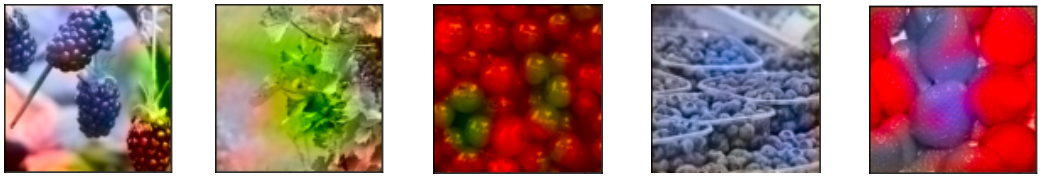


Figure 19: Predicted images for baseline.

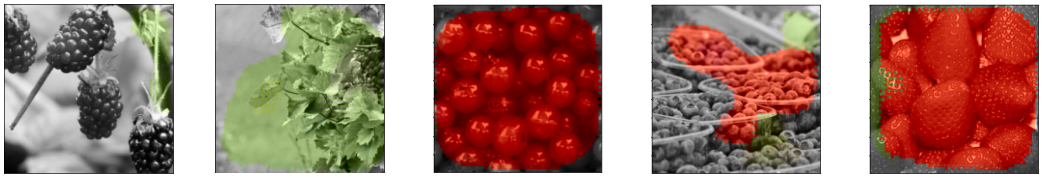


Figure 20: Predicted images for multinomial classification.

that the generator would have left the local minima and the gradiant descent could have found a better local minima or even the global minima for our loss function. If this is the case it is also possible that the loss functions of the discriminator and the generator would enter the phenomenon where they are battling each other once again.

4.2. Modified VGG16

It was known from the beginning that a lot of training time would be needed and that it would be a defining factor for the result. For this reason we used pre-trained weights to minimize the need for extensive. However the quality of the result was lower than expected and the amount of training required was still underestimated. Even when training on a subset of the Linnaeus data set the training time was very time consuming. In the study of Zhang et al. they fully trained their network for the colorization task on over 1.3 million images [14]. It is believed that the size of the training data is the most contributing reason for the big difference between the results. This theory is supported by the fact that when training the current models the loss is decreasing and the colorization becomes better. Still the model do not generalize very well, although the object is mainly colorized with plausibly colors. The regression model produced better result which was also surprising since the article that this work was based on argued that regression was worse than classification.

4.2.1 Regression

The regression model performed decently on certain pictures. It detected objects quite well and certain parts of the images looked very promising. The model produced just a couple of images that looked good and convincing. For those images the structure was similar, with mostly a blue color with different shades in the image. It was evident that the model favoured the color blue for many of the images in the test data set. A reason for this might be that the color distribution of the berries class of the Linnaeus data set was uneven, meaning that it contained a lot of blue images. Another reason might be because there are many types of berries in blue shades, for example black berries, blue berries, elderberries and grapes. However the same argument could be stated for red berries too. To determine this further analyses of the data set is required.

4.2.2 Multinomial classification

The multinomial classification model performed much worse than expected. An improvement over the regression model was expected because of the result in the article and faster training time. However that was not the case since the multinomial classification seemed to not be able to differentiate between objects. Images usually only had one or two different colours in them as well. It is not clear why this was the outcome, but a possible reason is that the multinomial classification method requires more training time. When looking at the loss graph in figure 17 it is clear that the model can be trained longer since no signs of overfitting

is present. Perhaps the multinomial classification is more sensitive to data sets with images that are very similar to each other. It is mentioned in the article that their model was trained on more powerful machines, for a longer period and with more data than our model. All three reasons surely has an impact on the performance of our model, and it is difficult to determine what would improve our model the most. It is sadly not mentioned in the article [14] how many epochs the model completed. Knowing this metric would have made it easier to determine what the problem with our solution is. It is possible that we have fewer epochs completed, which would mean that we are not training the model long enough. It is also possible that more epochs has been completed which would indicate that our data set is too small.

4.3. Model comparison

When comparing the results from the two models it is quite clear that the images generated from the cGAN-model are more credible than the ones generated from the modified VGG16-model. The cGAN-model also generated better images after a short training time when compared to the modified VGG16-model. However, the comparison of the images of the two models is quite unjust due to the selected data sets for each model. The cGAN-models data set solely consists of images of dining rooms thus limiting the models exposure of nature in the images of which it is trained and evaluated on. The data set used for the VGG16-models on the other hand only contains nature themed images and all of them containing berries. This makes the comparison of the two models complicated as if the image, which can be seen in Figure 19, contains a strawberry it will only seem realistic if the model colors it red. But if the image instead contains a table or a chair a multiple of colors can be chosen from and the image still appears to be realistic. So the modified VGG16-model needs to be very precise when estimating which color the berry should be when the cGAN-model can choose from a wider array of colors when estimating and still get a realistic image.

4.4. Future improvements

The following section describes potential improvements that could be made to increase the performance of the models. Further evaluation of the current results from the models are also discussed.

4.4.1 cGAN

As mentioned in section 4.1.2 it would be interesting to see if the model would improve if training would have been kept going for more iterations than what has been done in this paper. Another improvement that could have been contributing to a better model would be if a larger part of the data set would have been used, as for the model in this paper only 5 percent of the LSUN: Dining Room was used due to the memory limitations of Google Colab. As mentioned in section 4.1.1 the model is only trained and evaluated on images from an indoor setting, more specifically only for dining rooms, due to time and hardware limitations. For further evaluation it would be interesting to train the model on other data sets that have more variations than the LSUN: Dining room data set to see how it would perform on a more generalized plane. Finally, the evaluation of the cGAN-model is only done within the project group and no real test are done to see how well the generated images are perceived, this is something that would make it easier to benchmark the models.

4.4.2 Modified VGG16

Common for both training time VGG16 models was that the training time was very time consuming. Because of this a future improvement would be to train both the VGG16 models on a larger data set with better hardware to speed up the training time. To improve the results further it would be interesting to experiment with different hyperparameters, for example through the use of grid search.

To further evaluate the practical use of the model it would be interesting to evaluate whether the model can be used to improve the performance of a image classifier on greyscale images as suggested by Zhang et al. [14].

4.5. Conclusions

For this project it can be concluded that the cGAN-model was a better fit for the project scope and for the available hardware. The cGAN-model was noticeably faster to train than the VGG16 models and it could therefore be trained on a much larger data set. The colorized images yielded by the cGAN-model was more believable than the results from the VGG16 models. It is believed that the current result from the cGAN-model can fool a human, however, further evaluation is needed to quantify its performance. At the current stage it is assessed that the VGG16 models can not produce believable results.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] A. F. Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[3] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee, 2017.

[4] Y. Cao, Z. Zhou, W. Zhang, and Y. Yu. Unsupervised diverse colorization via generative adversarial networks. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 151–166. Springer, 2017.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[7] T. Joji, S. Abraham, R. Venugopal, and S. K R. Ijsrset195901 — review on different methods of image colorization. 05 2019.

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] B. Ly, E. Dyer, J. Feig, A. Chien, and S. Bino. Research techniques made simple: Cutaneous colorimetry: A reliable technique for objective skin color measurement. *The Journal of investigative dermatology*, 140:3–12.e1, 01 2020.

[10] M. Mirza and S. Osindero. Conditional generative adversarial nets, 2014.

[11] M. Podpora, G. P. Korbas, and A. Kawala-Janik. Yuv vs rgb-choosing a color space for human-machine interaction. In *FedCSIS (Position Papers)*, pages 29–34, 2014.

[12] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[13] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[14] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.