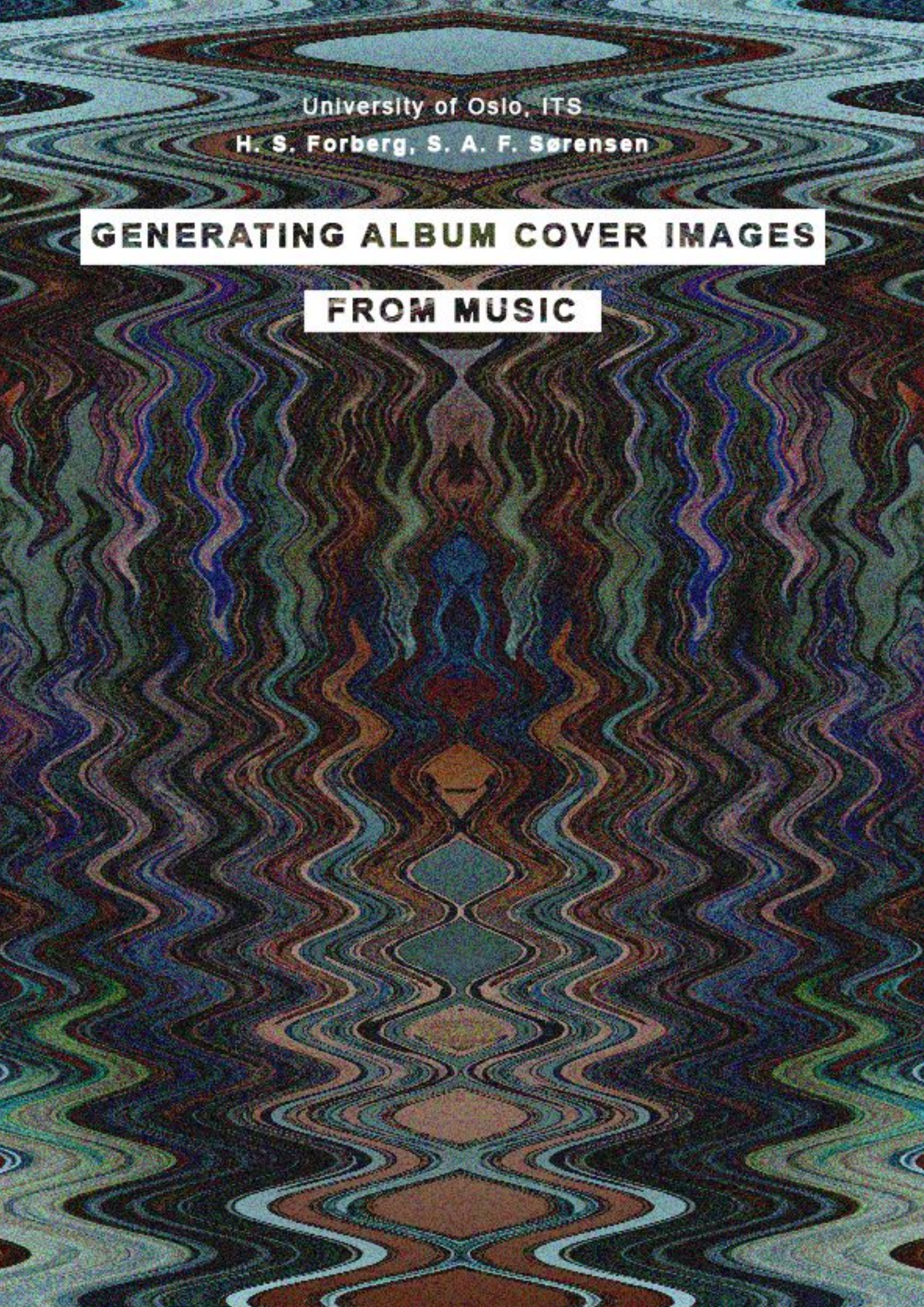
University of Oslo, ITS
H. S. Forberg, S. A. F. Sørensen

GENERATING ALBUM COVER IMAGES FROM MUSIC



University of Oslo, ITS

H. S. Forberg, S. A. F. Sørensen

29. November, 2018

Content list

Description of project

This project is devoted to creating a model that generates a fitting album cover image for any (mp3-type) input song. The model aims to produce album cover images of significant differences between genres.

Introduction

The model in this project consists of two main steps: **music feature classification** and **image generation**.

The **music feature classification** step is performed with a trained convolutional neural network, trained and tested on songs from 19 different genres. The input of the trained network is an mp3-file, and the output is a genre, identified with an integer between 0 and 18. The output-integer is fed into the **image generation** step of the model. In this project, a conditional generative adversarial network (CGAN) is utilized to generate images based on the training set of ~1600 album cover images. The output from the CGAN is a 100x100 RGB image based on album covers of the same genre.

The dataset

The dataset was collected from *Free Music Archive* (FMA) [10]. FMA is a large collection of licence free music that contains web links to corresponding album covers. The dataset contains over 20 000, 30 second snippets of songs from different genres. However, most of the songs were not already classified in genres, or belonged to obscure genres with too few examples to be included in the dataset. The resulting dataset contains ~2000 songs and ~1600 album covers.

Music theory

Music can potentially be divided into thousands of different genres. A music genre identifies which shared musical conventions a piece of music belongs to. However because of the artistic nature of music, these classifications are often subjective. Also, several genres may overlap. Academics have argued that categorizing music is inaccurate and outdated [1]. The content of the music may not always correspond to one distinct genre only, and other factors like cultural context, style and themes also matter [2]. In this project it is attempted to draw correlations between music and images, although it is debatable if this correlation actually exists.

Program flow

1.1 Music feature extraction

To differentiate between different genres, the musical harmonies and rhythm was extracted from each song. The musical harmonies (a combination of frequencies at different time steps) were extracted as *spectrograms* using *Short Time Fourier Transform*, with 0.1 second increments. The result is the "continuous" sound wave discretized to a grayscale image. The rhythm features were extracted to tempograms as grayscale images, using a built in function from the python module *librosa* [11]. Each spectrogram and corresponding tempogram were stacked vertically as one large grayscale image per song. These grayscale images were finally sliced horizontally as shown in figure 1.

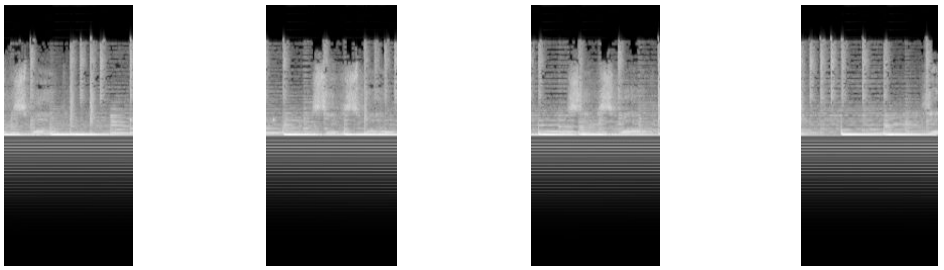


figure 1: stacked spectrogram-tempogram slices

1.2 Music feature classification:

A Convolutional neural network is utilized to classify songs into 19 different genres, each genre with an ID from 0 to 18.

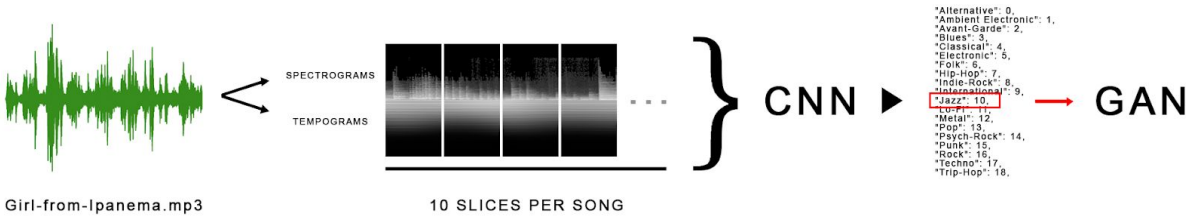


figure 2: music classification flow

2 Image generation

The song's genre ID-number and a noise vector ('z' in figure 3) is fed into a conditional generative adversarial network, to create a unique album cover image based on the given genre.

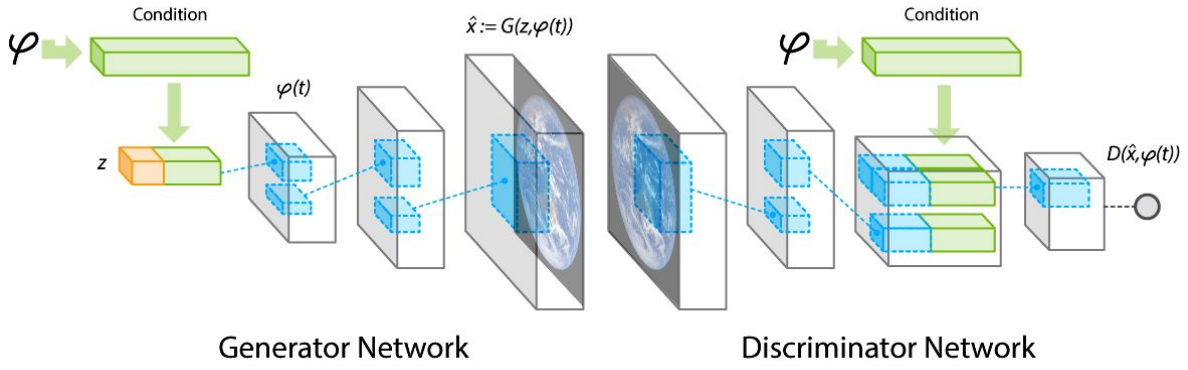


figure 3: C-GAN model flow [12]

Method

A song's spectrogram and tempogram is computed and stacked vertically on top of each other. The stacked image is sliced and fed as input to the **CNN**. Each slice is classified, and the most common genre detection becomes the result from the CNN. The output from the CNN become the input of the **C-GAN**, and the final output from the C-GAN is a 100x100x3 (RGB) image.

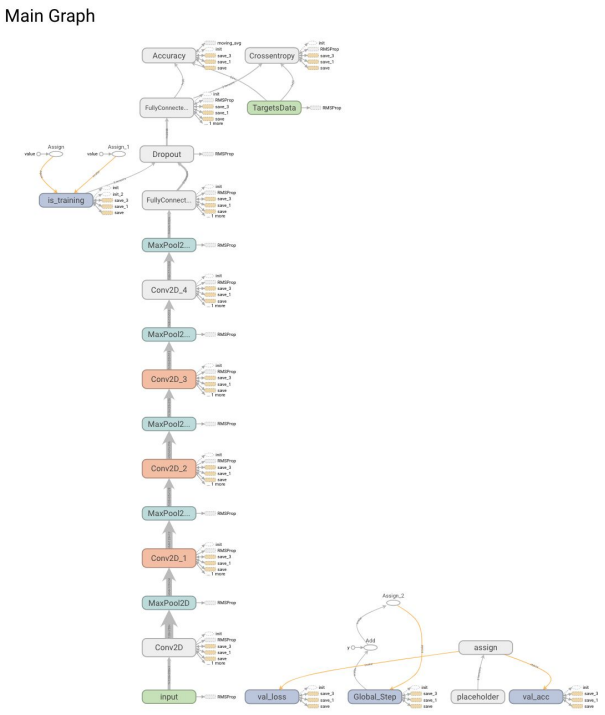
1. CNN

The goal of the CNN is to correctly classify songs into one of the 19 different genres. A random model would guess the correct genre (out of 19 options) in approximately 5% of the times.

The final classification model was first overfit to the training set with a 98% accuracy as proof of concept - classification of the dataset is possible. However, the best validation accuracy achieved was not higher than 43%. As previously mentioned: one could argue that a lot of music can fall in between categories of similarity, such as: punk, rock and metal, which the model had problems distinguishing between.

1.1 Model

The input layer takes a grayscale image of size 128x258 and processes it with a 2D-convolutional layer that is then fed into a max pooling layer. The convolutional-max-pooling-process is repeated three times, before enclosing in a fully connected layer. In the fully connected layer, *L2-regularisation* is applied.



Pooling is added to all of the hidden layers. In this model, the *max pooling* is utilized to gather the maximum value for each neuron-cluster in the previous layer into single input neurons in the next layer. Another technique that might be used for pooling is *average pooling*, taking the average of each neuron-cluster in the previous layer inputting it to single neurons in the next layer. The effect of the average pooling should be less-distinct, and perhaps more "noisy" than the max pooling technique. Considering that the point of the model is to distinguish between 19 distinct genres, the max pooling technique was utilised in the final model.

figure 4: final CNN model

1.2 Improvements

A couple of regularization techniques were tried in order to make the model as general as possible. *L1-regularisation* is a method to shrink the less important feature coefficients to zero, thus removing some features all together. This is a well-performing method in the case of inputs with a large number of features. [3] The initial thought was that the inputs in this project contained a large number of features, each slice contains a significant number of tempo- and frequency features describing a small part of a song. Intuitively it seemed favourable to remove as many features as possible to minimise the false assumptions about the songs' genres. However, when adding the L1-regularization to the second fully connected layer, the training lasted ~200 epochs before the model was overfit, resulting in a validation accuracy of no more than ~31%. Too many of the important features might have been removed, thus the model lost valuable information when trying to classify the songs. Instead, the L2-regularization technique was tried. *L2-regularization* adds a penalty term to the loss function, and more information is kept during the training [3].

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Cost function

L2-regularization

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Cost function

L1-regularization

When applying L2-regularization to the second fully connected layer, the model was overfit in ~30 epochs with a validation accuracy of ~35%. In comparison to L1-regularization, where the model was overfit after ~200 epochs with a validation accuracy of ~30%, the L2-regularization technique was not only a faster, but also a better working regularization technique in this classification task.

To further prevent overfitting, and improve the validation accuracy, dropout was added to the first fully connected layer. Neurons tend to become codependent during training, and thereby creating connections that are not really there.[4] These co-dependencies, or connections, will usually lead to an overfit network. Drop out forces all neurons to learn more robust features. Drop out removes connections between neurons with a probability of a factor between 0 and 1.

Lastly, adding another convolutional layer was tried to improve generality of the classification model. Adding another convolutional layer improved the model some, and the classification model was deemed good enough at this point.

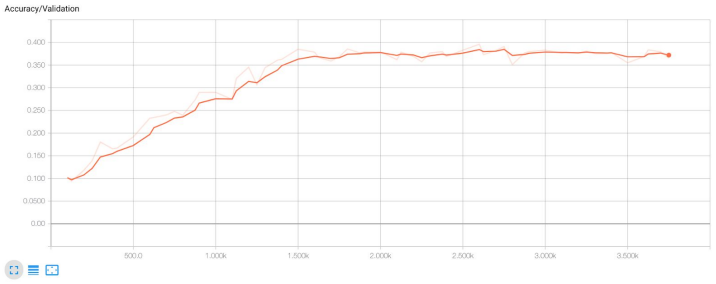


figure 5: 3 convolutional layers with a dropout factor of 0.8

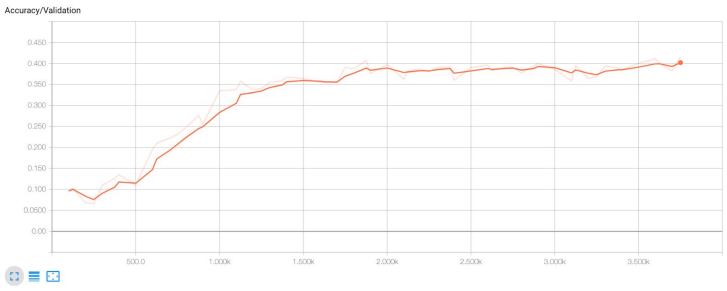


figure 6: 4 convolutional layers with a dropout factor of 0.8

1. 3 Further development

One proposed method to achieve a better validation accuracy is to expand the training set. Initially the training set contained 13000 song slices. The training set was expanded by about 20% or 3000 slices resulting in a total of 16 000 training samples for the final model. The expansion of the training set improved the validation accuracy of about 5%, therefore, a further expansion of the training set is to be suggested.

Another method to improve validation accuracy is to decrease the number of genres/labels. During the testing of the trained model, we noticed that music with distinct drum beats (take Hip-Hop) were easier to classify than music that did not contain drums (take Classical). If specializing a classification model on modern/popular music, gathering rhythm- and tempo features might be a good place to start.

2. C-GAN

In this project a Conditional Generative Adversarial Network (C-GAN) was used to generate album covers based on a given integer input. In this case a genre label is sent into both the generator and discriminator network as the condition. The goal of training the C-GAN is to reach an equilibrium between errors of generator and discriminator, and the learning phase is ended when the generator is able to fool the discriminator in more than 50% of cases[5].

2.1 Model

The discriminator consists of two 2D-convolutional layers with Leaky RELU activation functions applied. The generator consists of two 2D-deconvolutional layers with RELU activation functions applied (see figure 3). The kernel size 5x5 is suggested to generate smoother images than with a smaller kernel size [14]. Batch normalisation is applied to each deconvolution/convolutional layer to normalise the activation functions.

2.2 Improvements

One major problem encountered during training of the C-GAN, was that the discriminator loss quickly converged towards zero, meaning that the generator was not able to fool the discriminator. When the discriminator loss converges to zero,

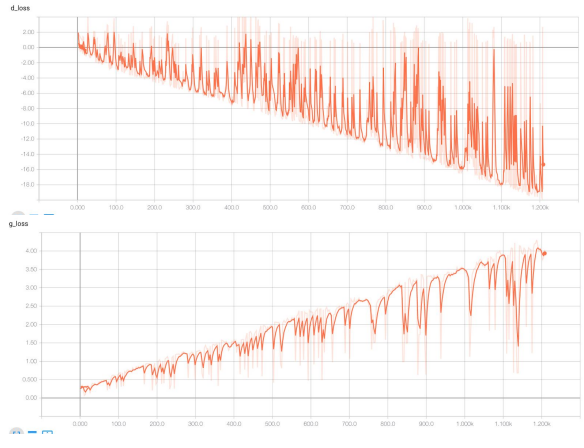
the generator gradient vanishes and no learning happens [7]. The resulting images were noisy, and did not resemble anything from the training set.

To reduce the discriminator's early convergence, the discriminator was trained every other step, to decrease the generator loss and fool the discriminator more often. The generator could train for much longer before the discriminator loss converged to zero. The images were less noisy, and contained uniform shapes, resembling the training set.

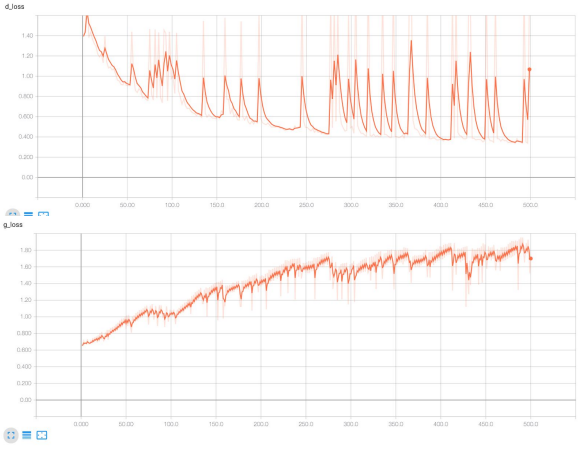
In addition, the initial loss function was a sigmoid cross entropy loss function. To improve the results, *wasserstein loss* with gradient penalty was instead applied. Wasserstein loss provides improved stability, avoids mode collapse and gives meaningful learning curves that makes debugging and hyperparameter adjustments easier [6]. A gradient penalty was added to the discriminator loss, based on the proposition in the article *Improved Training of Wasserstein GANs* [8]. Gradient penalty improves training speed, sample quality and results in more stable gradients that neither vanish nor explode [8].

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}\left[D(\tilde{\boldsymbol{x}})\right] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}\left[D(\boldsymbol{x})\right]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\text{Our gradient penalty}}.$$

Since Wasserstein GAN doesn't suffer from the same type of vanishing gradient problem as other GANs, the discriminator should be as close to optimal as possible[6]. Therefore the discriminator was trained three times more than the generator in the final C-GAN model. When applying wasserstein loss to the model, the sigmoid of the discriminator output is no longer in use, because it is desirable to not constrain the output range [6].



Wasserstein loss function



Sigmoid loss function

The results from the CGAN after applying wasserstein loss started resembling the training set more accurately than with sigmoid cross entropy loss. The generated images contained clear shapes from the training set. It became clear that the training set contained binary images that seemed to pollute the results.

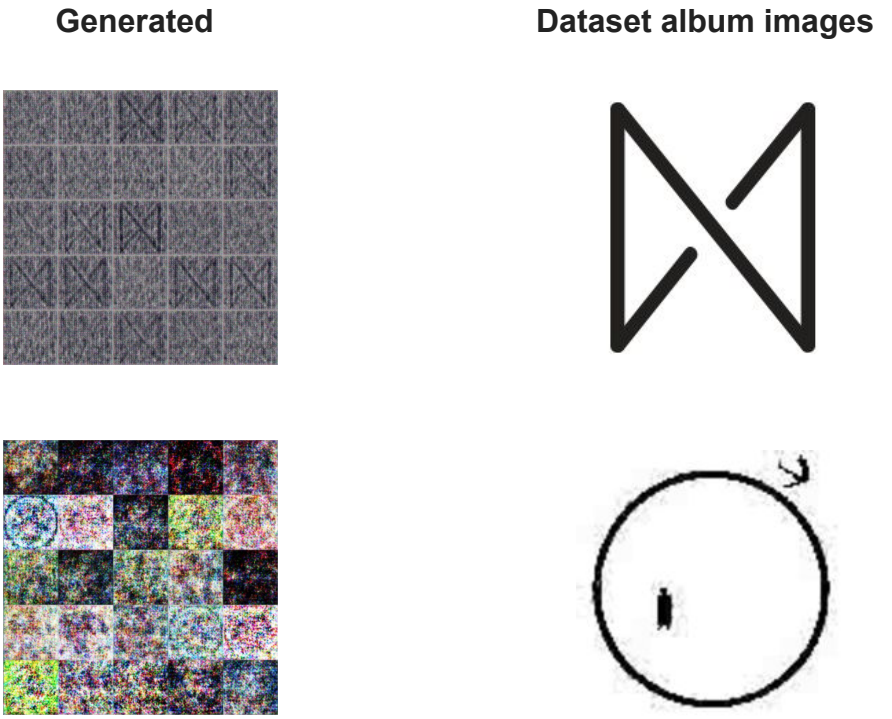


figure 7: temporary image generation results

Although these results show that the generator is able to generate images resembling shapes from the training set, binary images were removed from the training set to get more general results. After removing binary images from the dataset, these shapes discontinued to show up in the generated images.

2.3 Further development

Because the correlation between music and images is a subjective matter, the model can be helped to learn a desired “look” with a few improvements: It is proposed to only include images that contain the kind of information one would want to see in a generated result. Abstract images, and images that resemble noise tended to make the discriminator’s learning stagnate during the course of the project.

Unless abstract images are wanted, it is suggested to exclude abstract/noisy images from the training set.

Another way to get a desirable result might be to add image filters to the model. Filters force the model to produce images that resemble the filters by multiplying the generated images with the filters [9].

A larger kenel size at the top deconvolutional layer in the generator model is suggested, but increasing the kernel size is at the same time computationally costly [14]. One final suggestion is to progressively add layers to the GAN model. The generator will be able to scale images from layer to layer, and this method is especially suggested in cases where one would want larger images [13].

Results

The CNN was able to classify a validation set with an accuracy of ~40%. When training the C-GAN with wasserstein loss and gradient penalty, convergence of the discriminator loss was never reached. The C-GAN was able to generate a range of different album covers for each genre.

Figure 8 illustrates the emerging distinctness of different genres at an early time step.

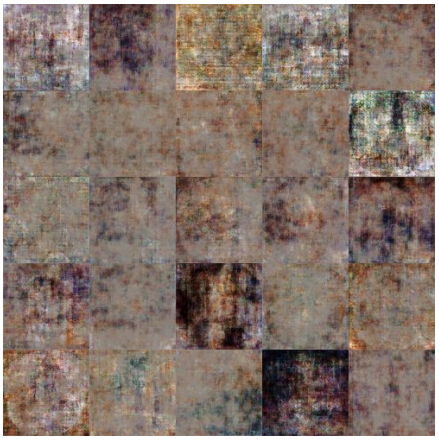
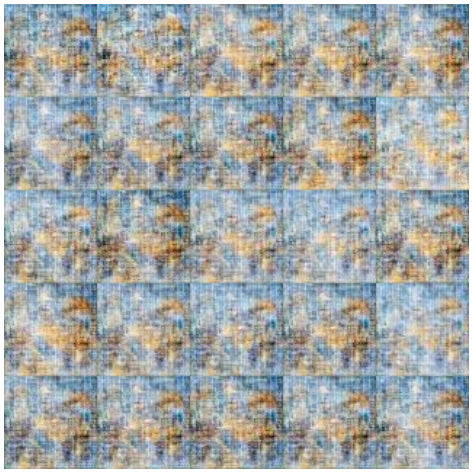
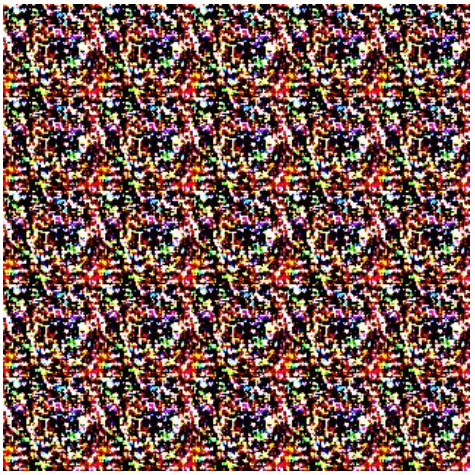
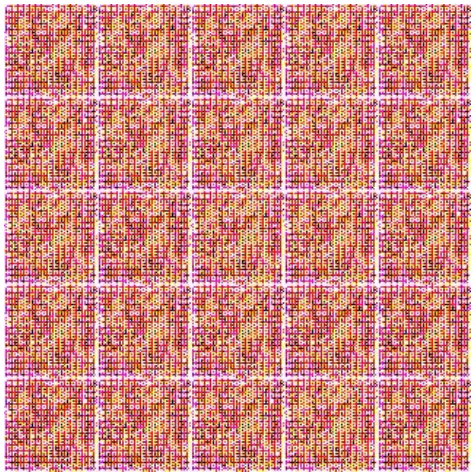


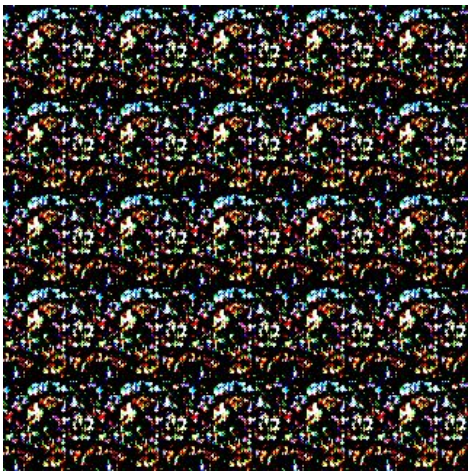
figure 8: a mix of generated images with different labels

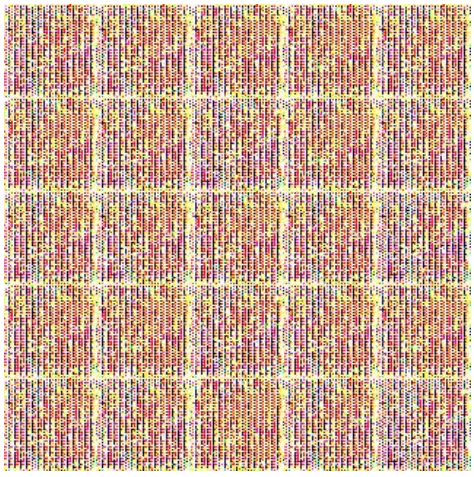
Following, a selection of generated album covers for different time steps and different labels. However abstract, the images contain artifacts like: symbols, shapes and colors, resembling the training set.

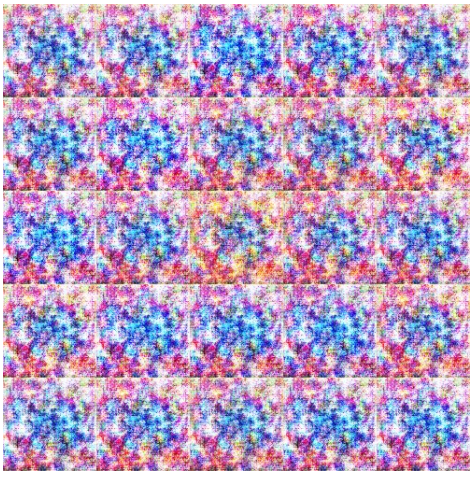


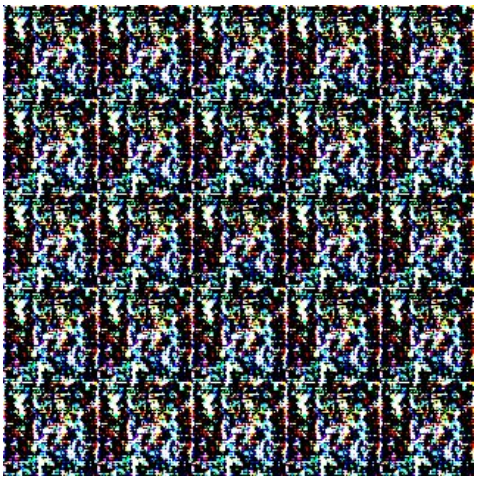


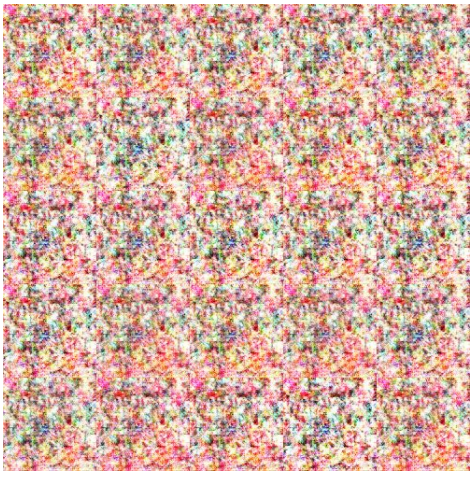


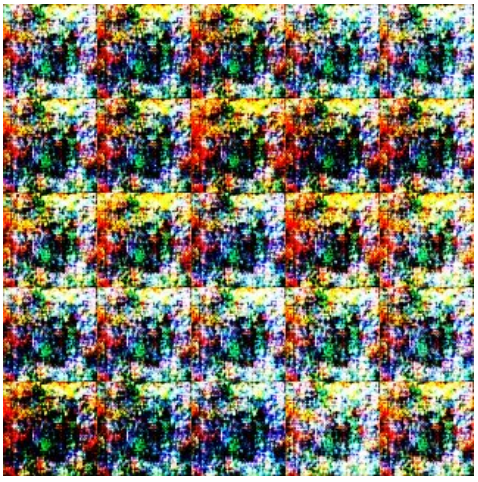


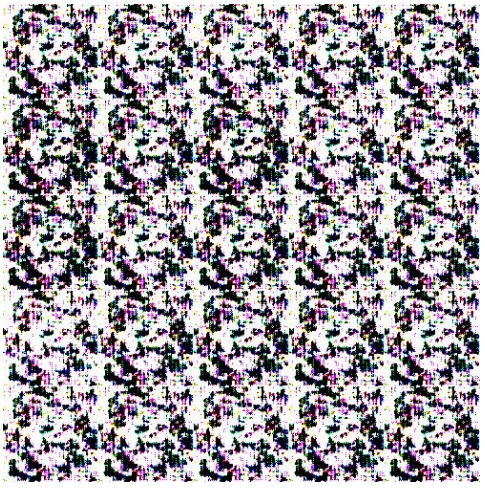


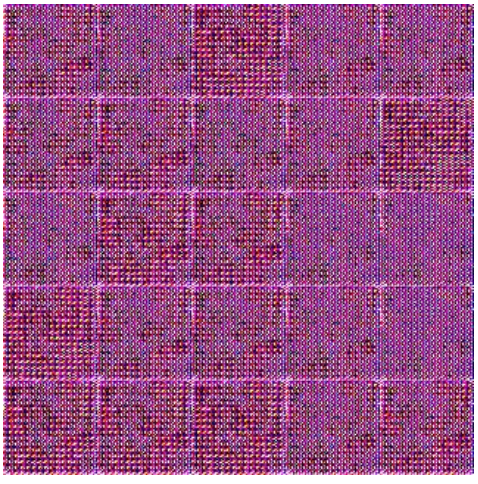


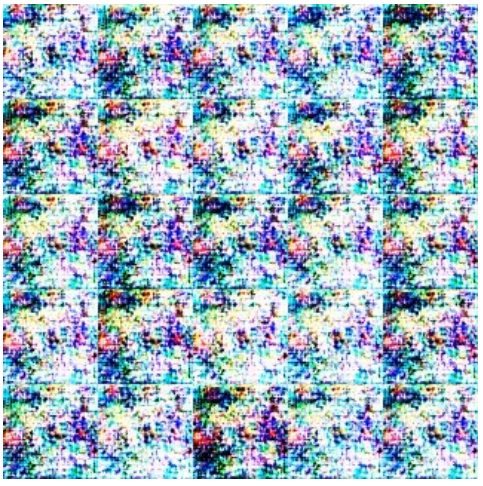


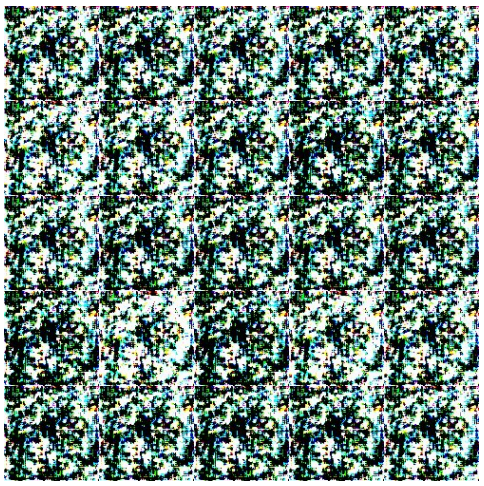


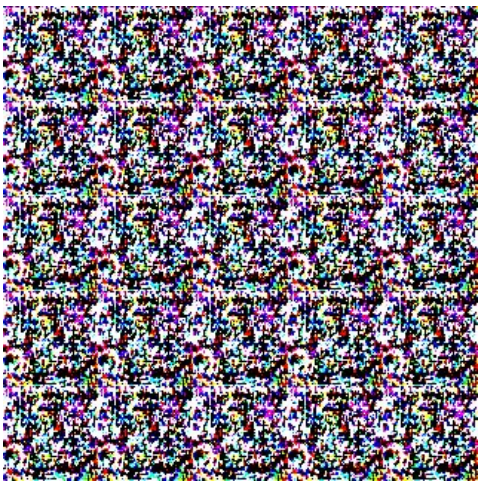


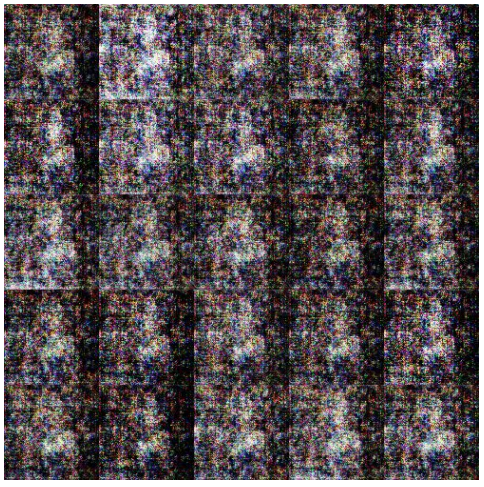


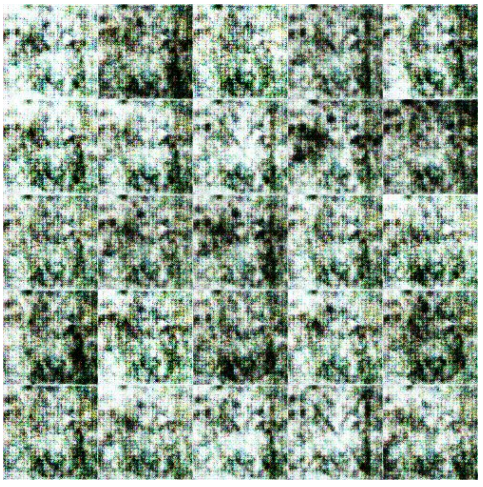












Unfortunately, the generated images were not properly labeled during training. The generated images correspond to different labels, but we can not say for sure which specific genre each image is to represent. Although to properly label each image would require some minor changes in the code, we did not have the time to retrain the C-GAN.

Summary

In this project music features were translated to grayscale images and classified into genres. The final classification model was tested with an accuracy of ~43% for 19 genres. The genre ID was fed into the CGAN and outputs an album cover image based on album cover images of the same genre. The CGAN was able to output distinctly different images for every genre ID.

Bibliography

[1] D. M. Greenberg, M. Kosinski, D. J. Stillwell, B. L. Monteiro, D. J. Levitin, P. J. Rentfrow, *Preferences for Musical Attribute Dimensions Reflect Personality*, 2016. https://journals.sagepub.com/doi/abs/10.1177/1948550616641473

[2] T. Laurie. *Music Genres As Methods*, 2014. https://www.academia.edu/9087511/2014_Music_Genre_As_Method

[3] Anuja Nagpal, *L1 and L2 Regularization Methods*, 2016. https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

[4] Amar Budhiraja, 2016. https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

[5] H. Eraqi. *GANs: Generative Adversarial Networks*, 2018. https://www.linkedin.com/pulse/gans-generative-adversarial-networks-hesham-eraqi

[6] M. Arjovski, S. Chintala, L. Bottou. *Wasserstein GAN*, 2017. https://arxiv.org/pdf/1701.07875.pdf

[7] M. Arjovski, L. Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*, 2017. https://arxiv.org/pdf/1701.04862.pdf

[8] I. Gulrajani, F. Ahmed, M. Arjovski, V. Doumolin, A. Courville. *Improved Training of Wasserstein GANs*, 2017. https://arxiv.org/pdf/1704.00028.pdf

[9] A. Redford, L. Metz, S. Chintala, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, 2016. https://arxiv.org/pdf/1511.06434.pdf

[10] Free Music Archive, 2016. https://github.com/mdeff/fma, http://freemusicarchive.org

[11] Librosa, 2018. https://librosa.github.io/librosa/generated/librosa.feature.tempogram.html

[12] Abto Software, 2017. https://www.abtosoftware.com/blog/image-to-image-translation

[13] T. Karras, T. Aila, S. Laine, J. Lehtinen, *Progressive growing of GANs for improved quality, stability and variation*, 2018. https://arxiv.org/pdf/1710.10196.pdf

[14] Utkarsh Desai, 2018. https://medium.com/@utk.is.here/keep-calm-and-train-a-gan-pitfalls-and-tips-on-training-generative-adversarial-networks-edd529764aa9