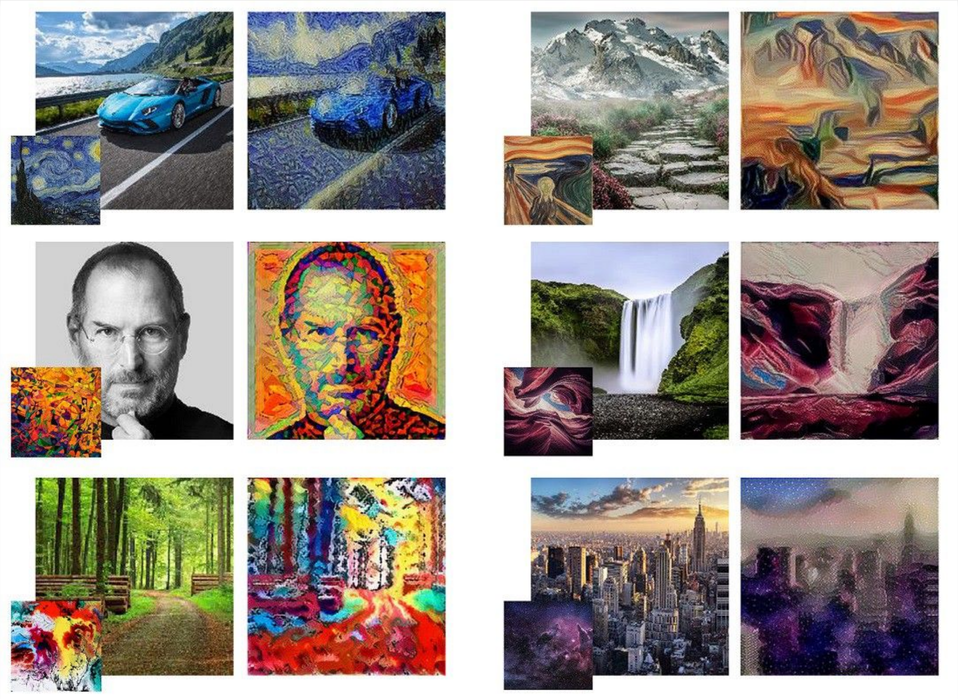
Neural Style Transfer in Audio Domain

Robert Bosch Scholarships
Andrei Bratu, Undergraduate

Purpose of scholarship

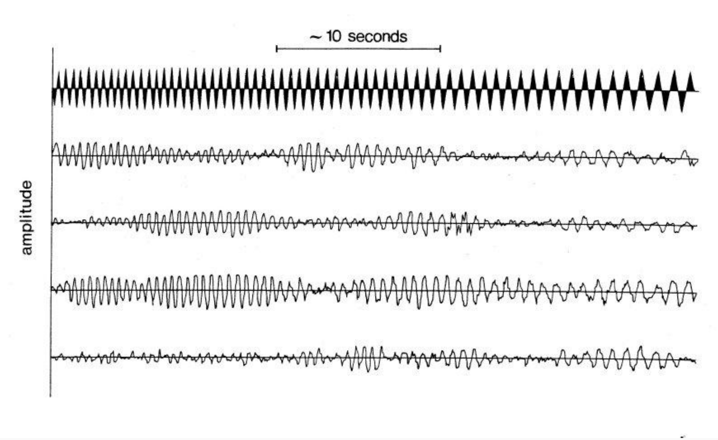


A Neural Algorithm of Artistic Style

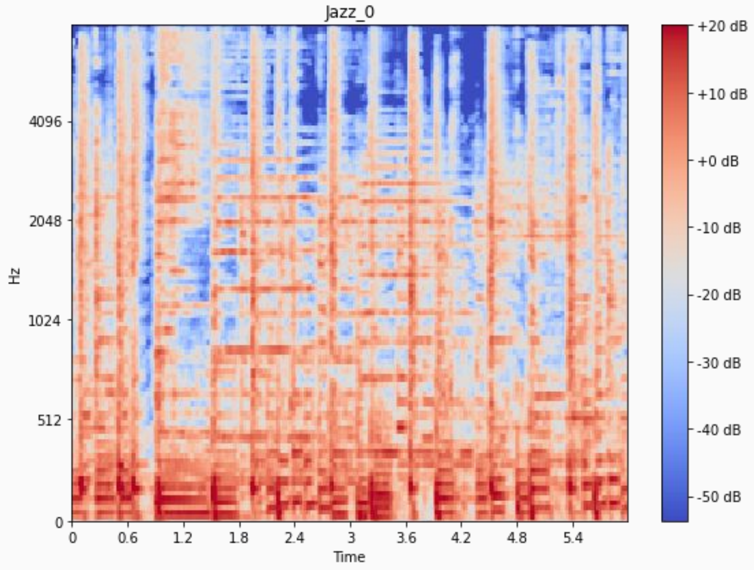
Leon A. Gatys, Alexander S. Ecker,1

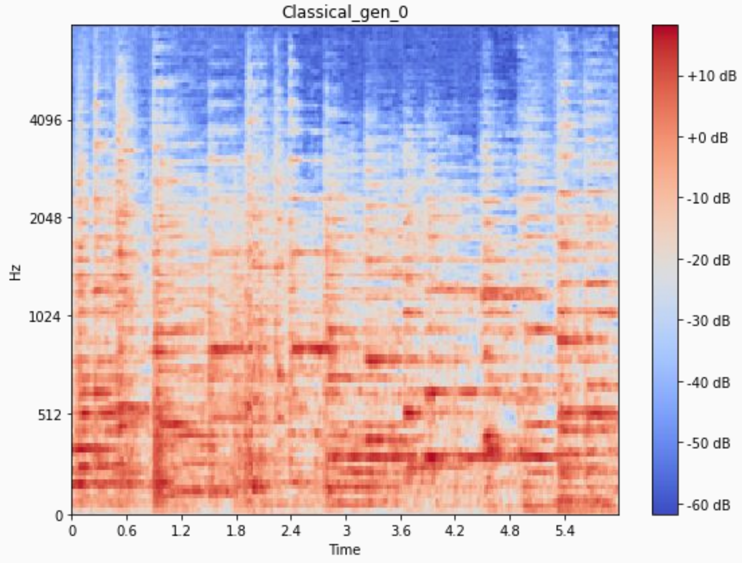
Matthias Bethge

September 2015

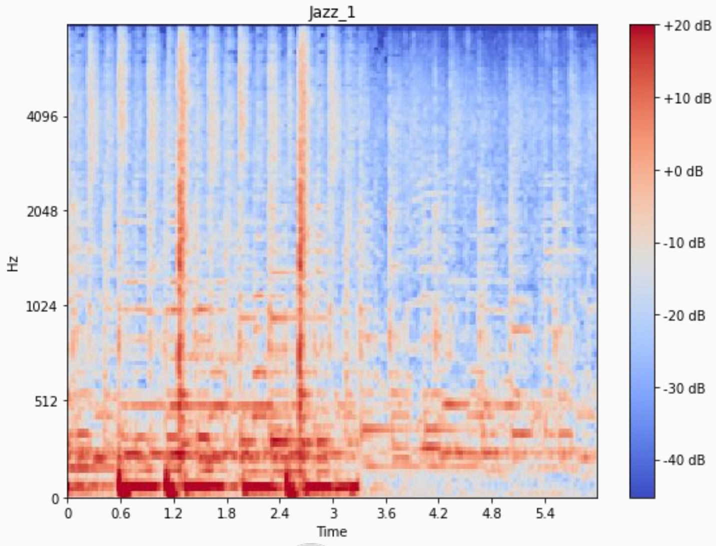


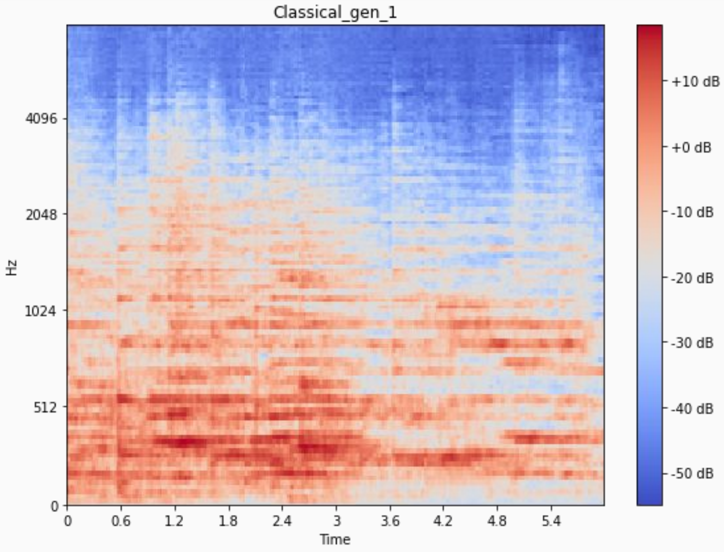
Results


Jazz_0
Hz
4096
2048
1024
512
0
0
0.6
1.2
1.8
2.4
3
3.6
4.2
4.8
5.4
Time
+20 dB
+10 dB
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB

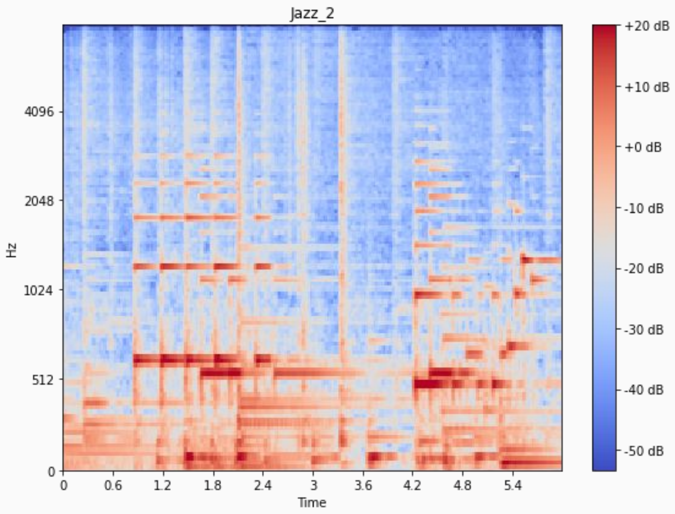

Classical_gen_0
Hz
4096
2048
1024
512
0
0
0.6
1.2
1.8
2.4
3
3.6
4.2
4.8
5.4
Time
+10 dB
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB

Results

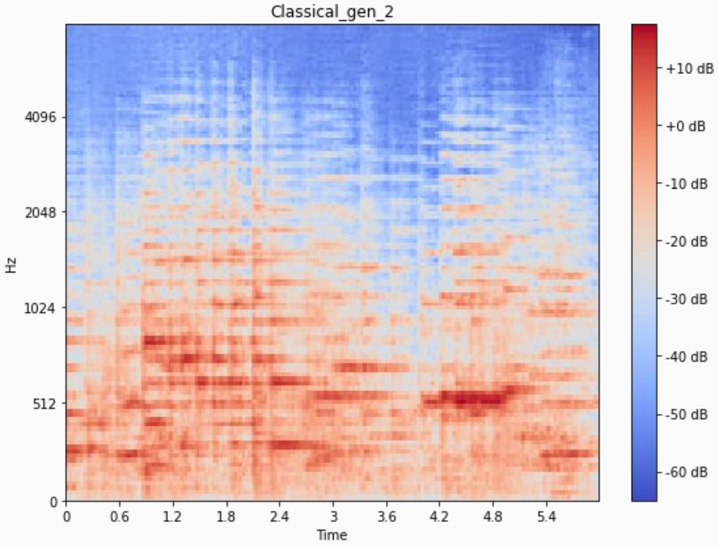

Jazz_1
+20 dB
+10 dB
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
4096
2048
1024
512
0
Hz
0
0.6
1.2
1.8
2.4
3
3.6
4.2
4.8
5.4
Time


Classical_gen_1
+10 dB
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
4096
2048
1024
512
0
Hz
0
0.6
1.2
1.8
2.4
3
3.6
4.2
4.8
5.4
Time

Results



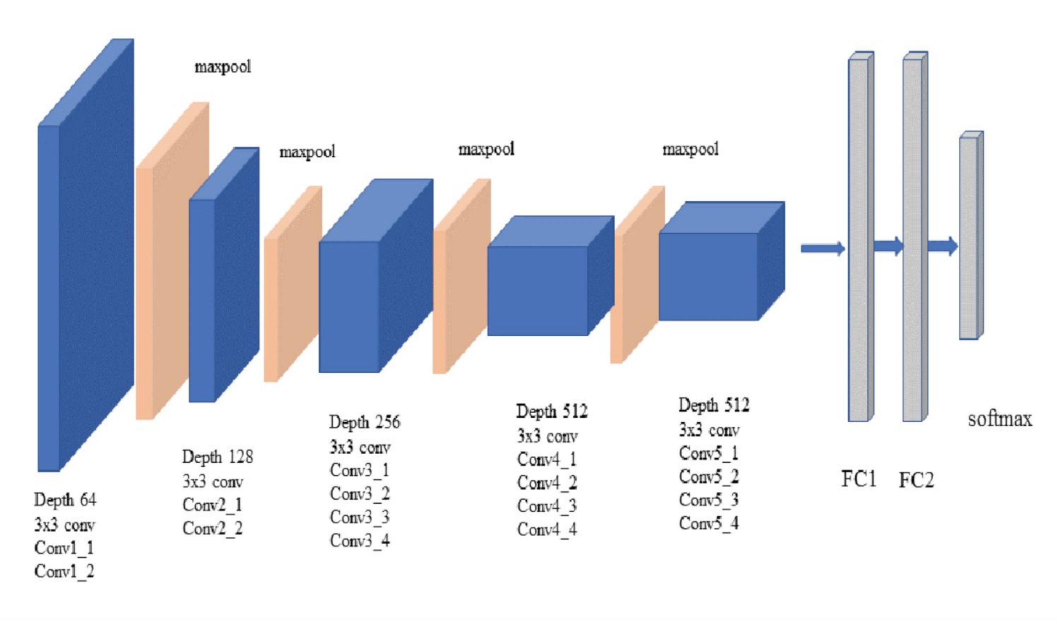
Jazz_2
Hz
4096
2048
1024
512
0
0
0.6
1.2
1.8
2.4
3
3.6
4.2
4.8
5.4
Time
+20 dB
+10 dB
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB



Classical_gen_2
Hz
4096
2048
1024
512
0
0
0.6
1.2
1.8
2.4
3
3.6
4.2
4.8
5.4
Time
+10 dB
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB

Table of Contents

CNN in a nutshell

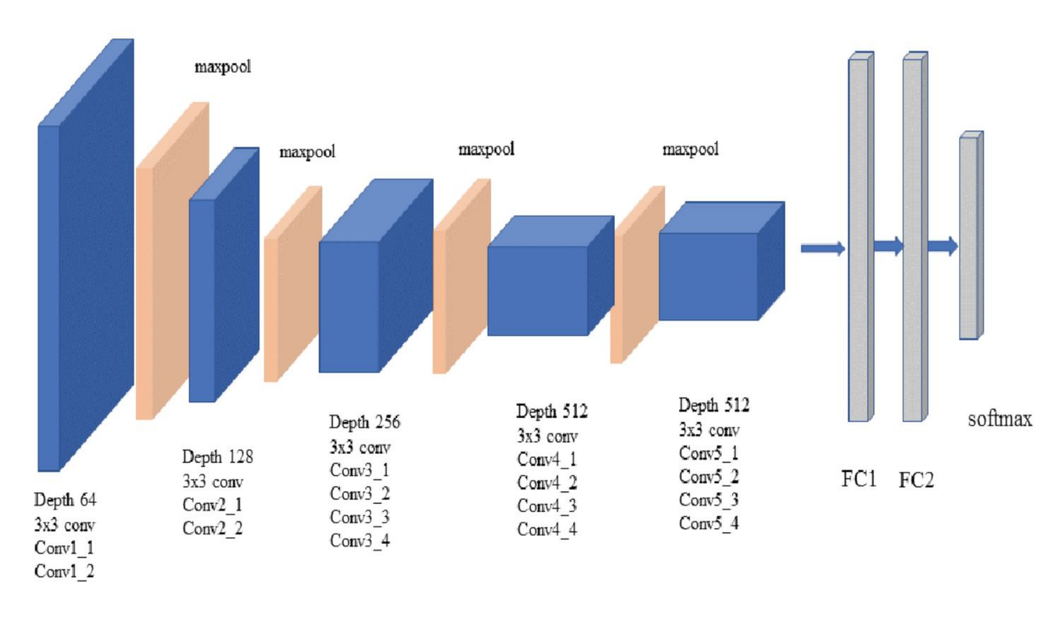


Convolution

Mathematical operation on two functions that produces a third function expressing how the shape of one is modified by the other.

0	1	2
2	2	0
0	1	1

CNN in a nutshell

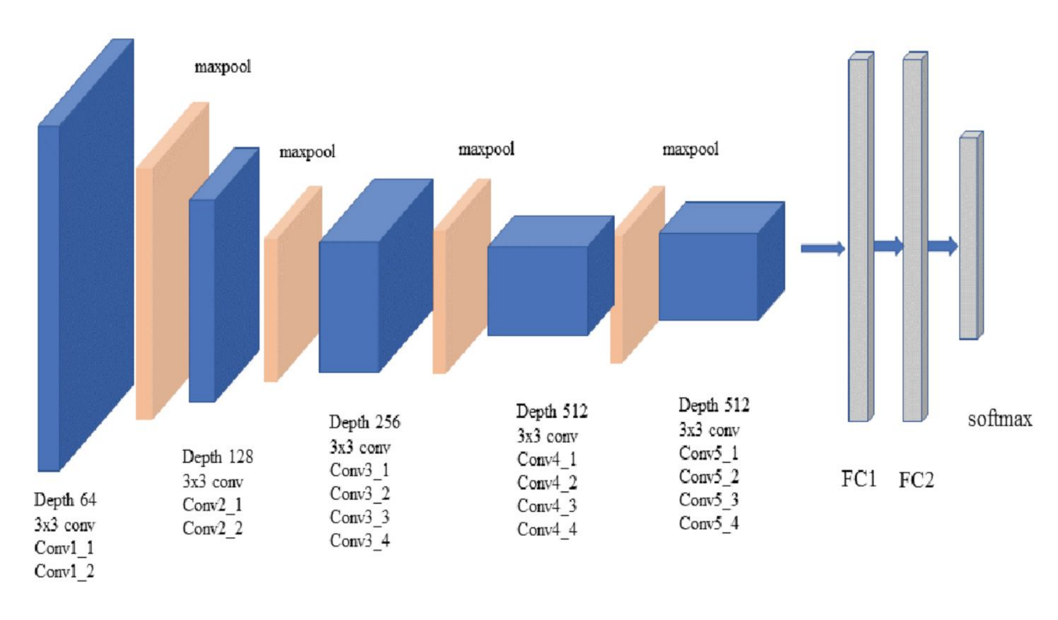


$$h[i,j] = u + \sum_{a,b} V[a,b] \cdot x[i+a, j+b].$$

Translation Invariance

A shift in the inputs x should simply lead to a shift in the activations h . This is only possible if V and u do not actually depend on (i,j).

CNN in a nutshell

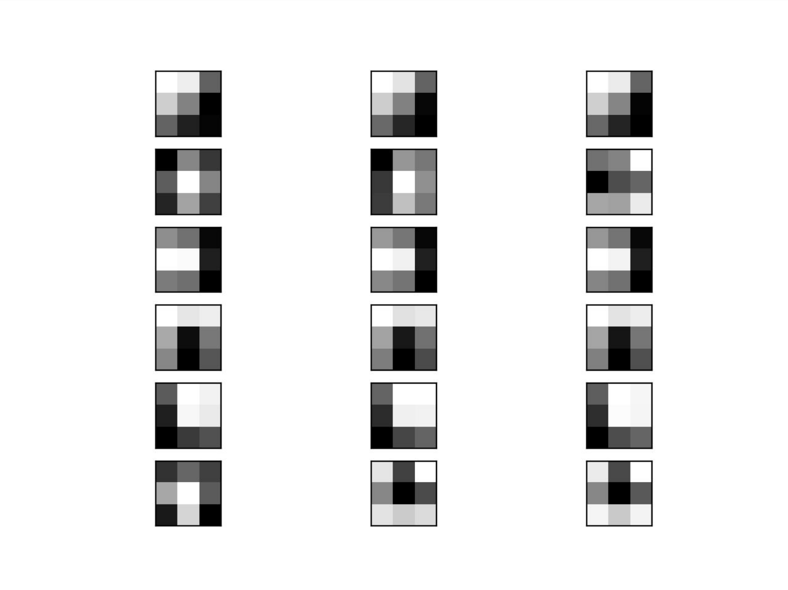


$$h[i,j] = u + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} V[a,b] \cdot x[i+a, j+b].$$

Locality

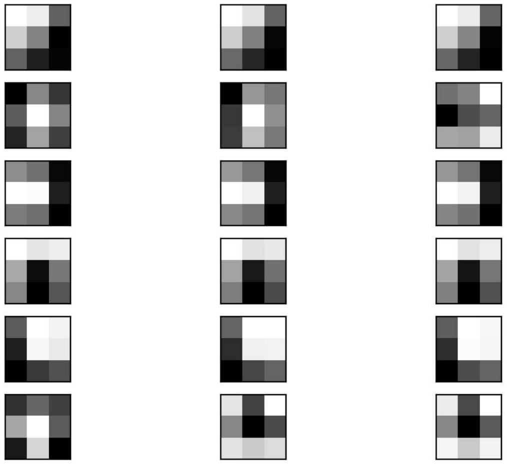
This means that outside some range
$|a|,|b|>\Delta$, $V[a,b]=0$

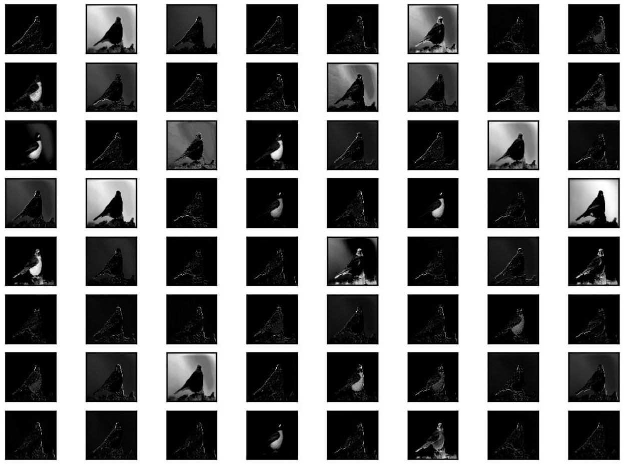
CNN in a nutshell



Courtesy of Machine Learning Mastery

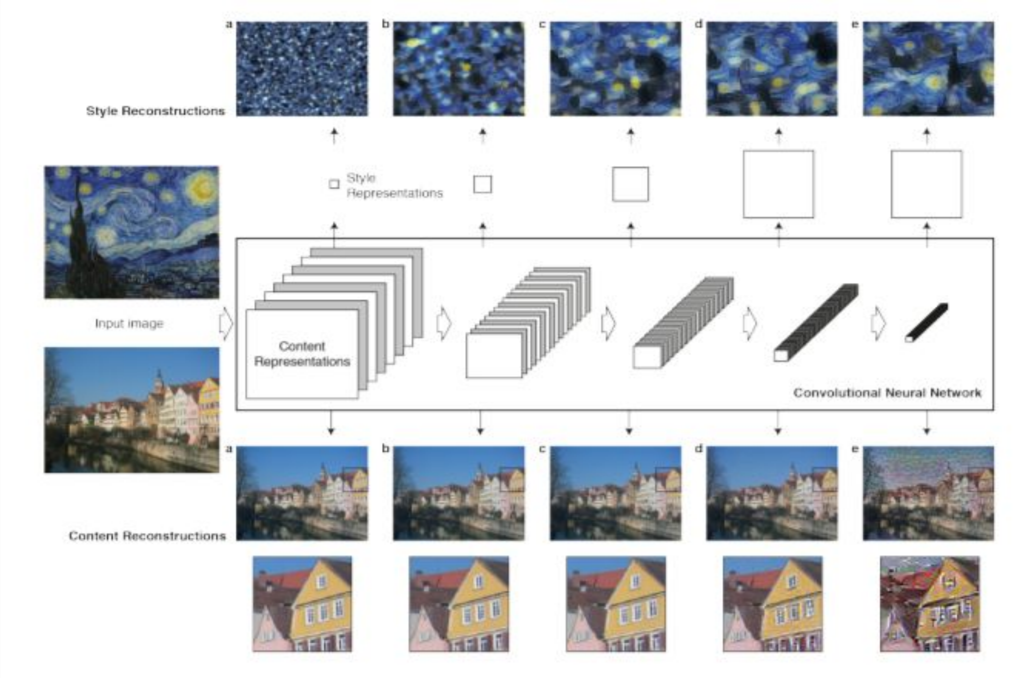
CNN in a nutshell





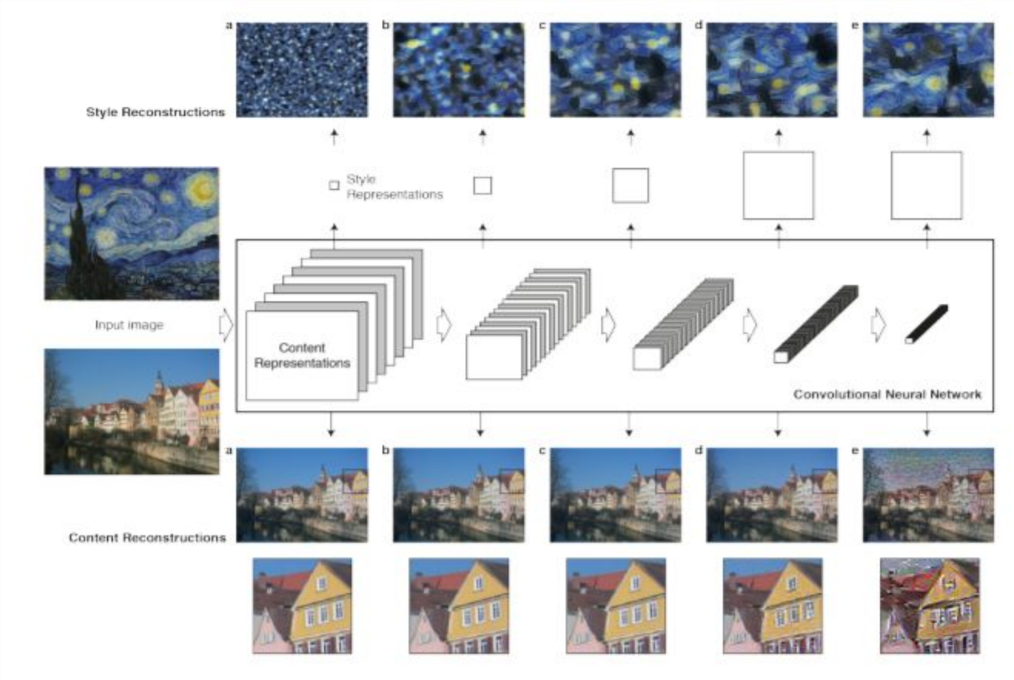
Courtesy of [Machine Learning Mastery]

Gatys Approach



The key finding of this paper is that the representations of content and style in the Convolutional Neural Network are separable. That is, we can manipulate both representations independently to produce new, perceptually meaningful images.

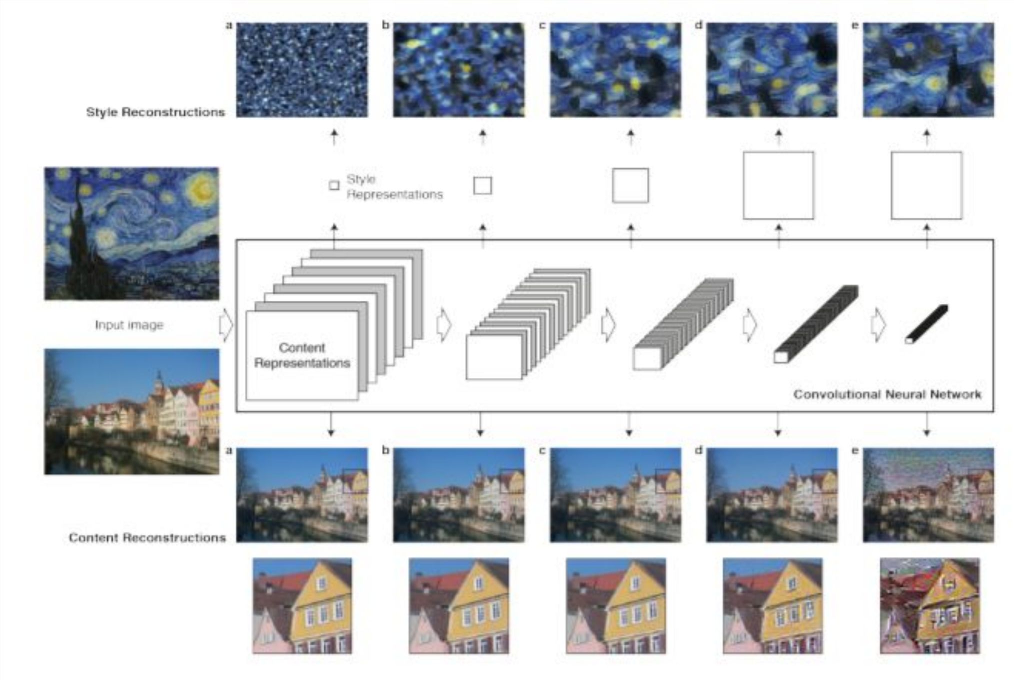
Gatys Approach

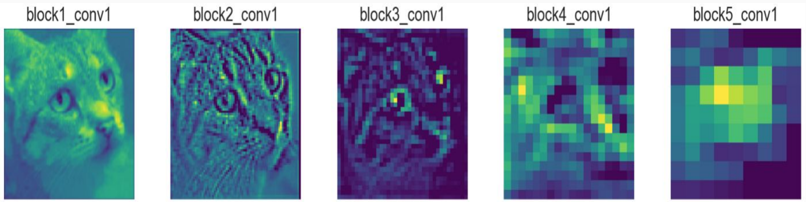


How do we extract the content?

We find that reconstruction from lower layers is almost perfect (a,b,c). In higher layers of the network, detailed pixel information is lost while the high-level content of the image is preserved (d,e)

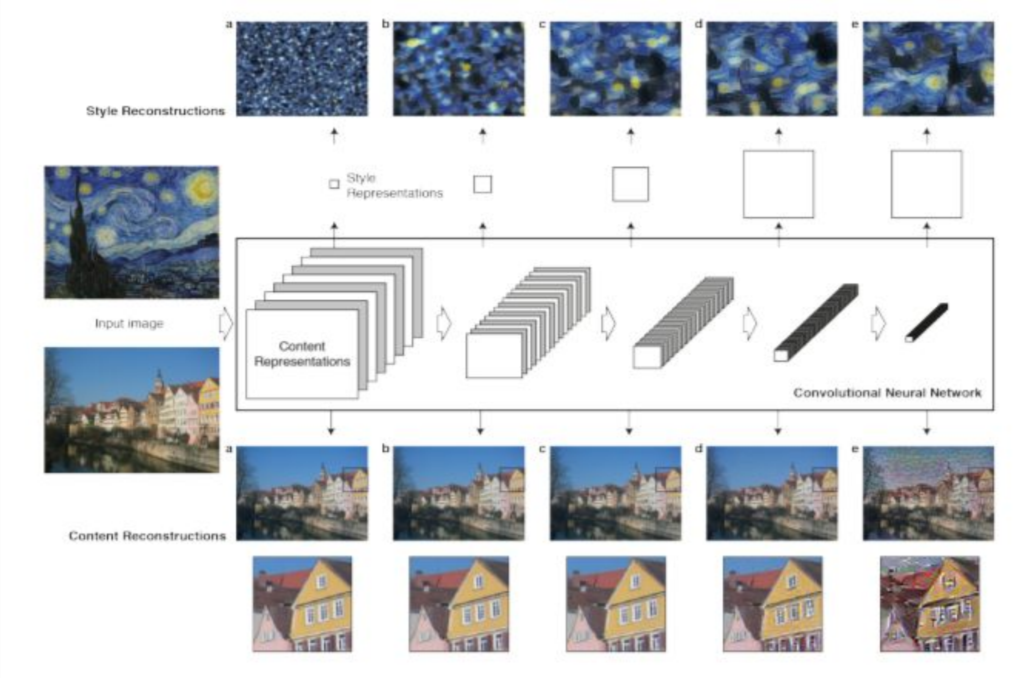
Gatys Approach





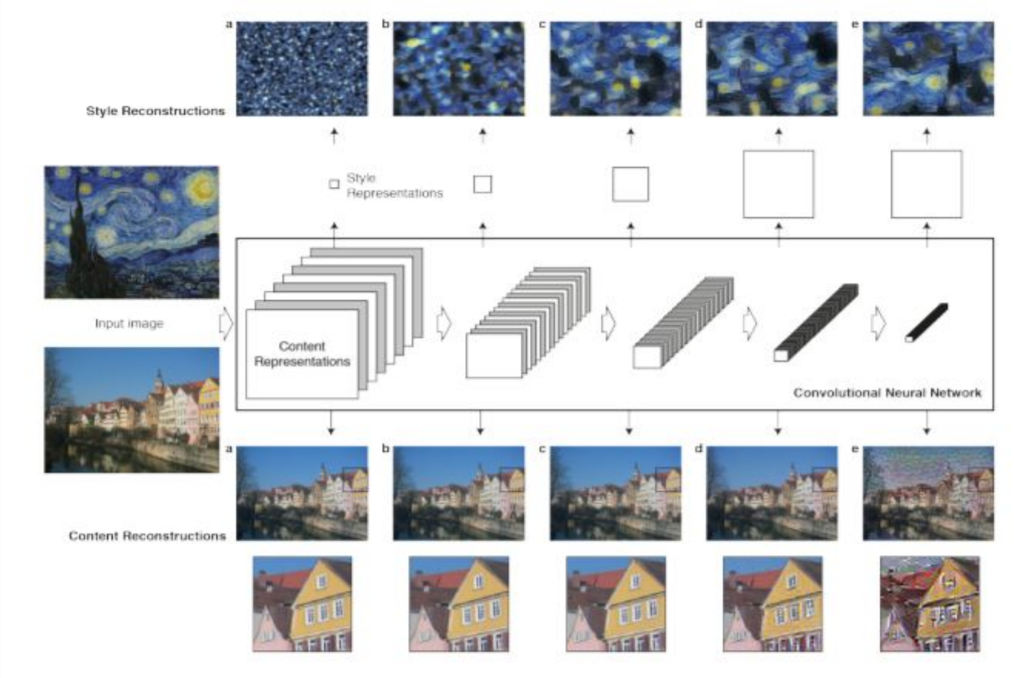
We reconstruct the input image from from layers ‘conv1 1’ (a), ‘conv2 1’ (b), ‘conv3 1’ (c), ‘conv4 1’ (d) and ‘conv5 1’

Gatys Approach



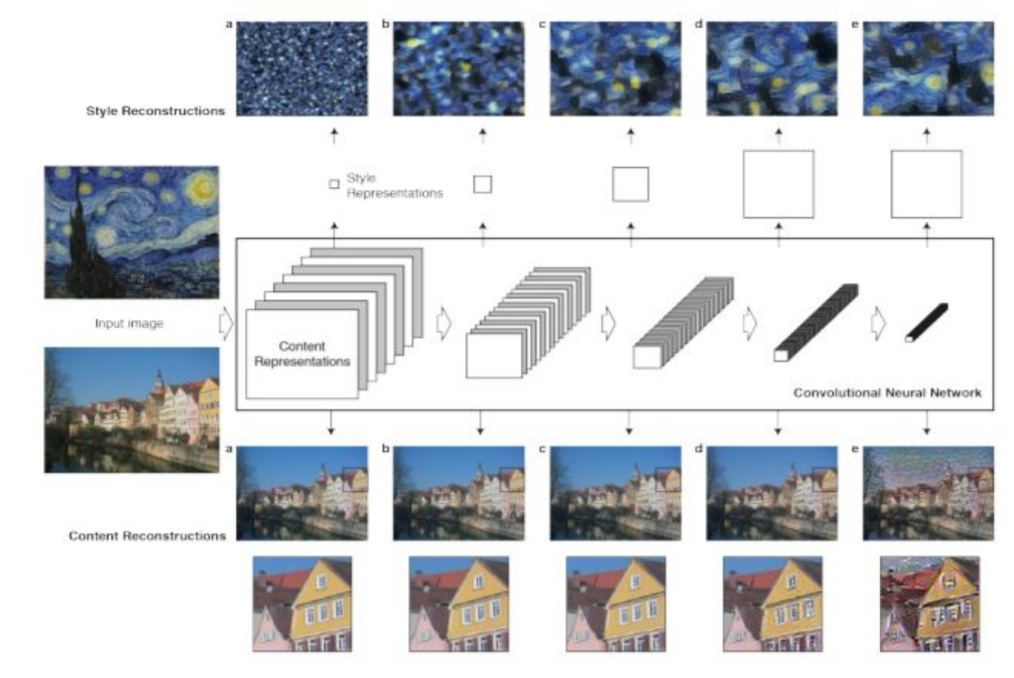
$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

Gatys Approach



How do we extract the style?
*On top of the original CNN representations we built a new feature space that captures the style of an input image. The style representation
computes correlations between the different features in different layers of the CNN*

Gatys Approach



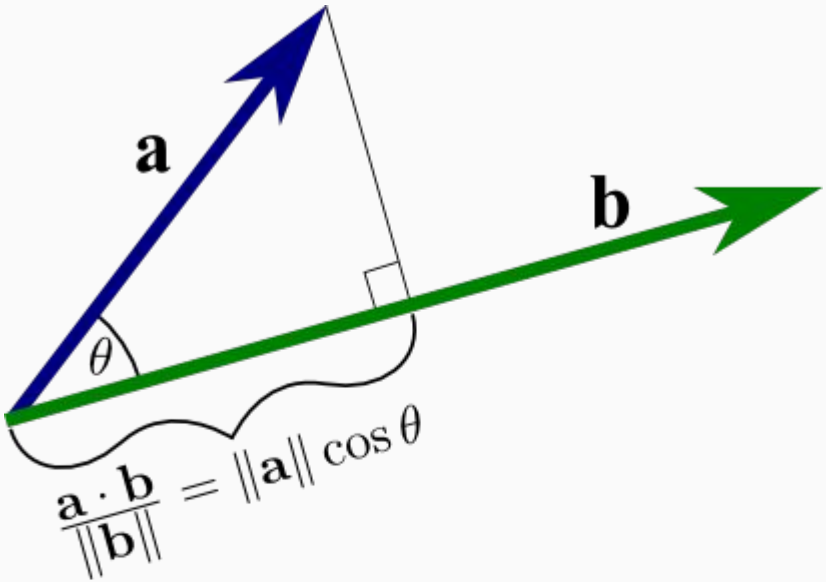
$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l.$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

Math Intuition

Me reading the Style part of Gathys.

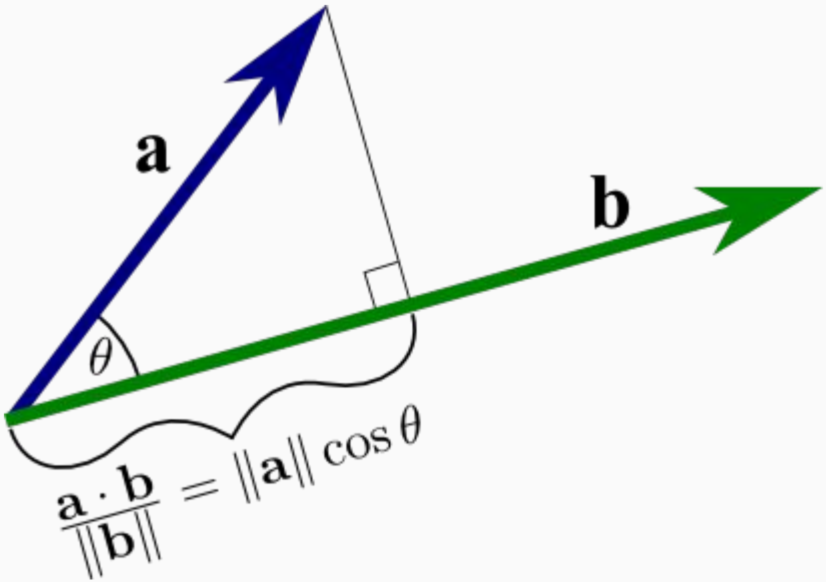
Math Intuition



$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}.$$

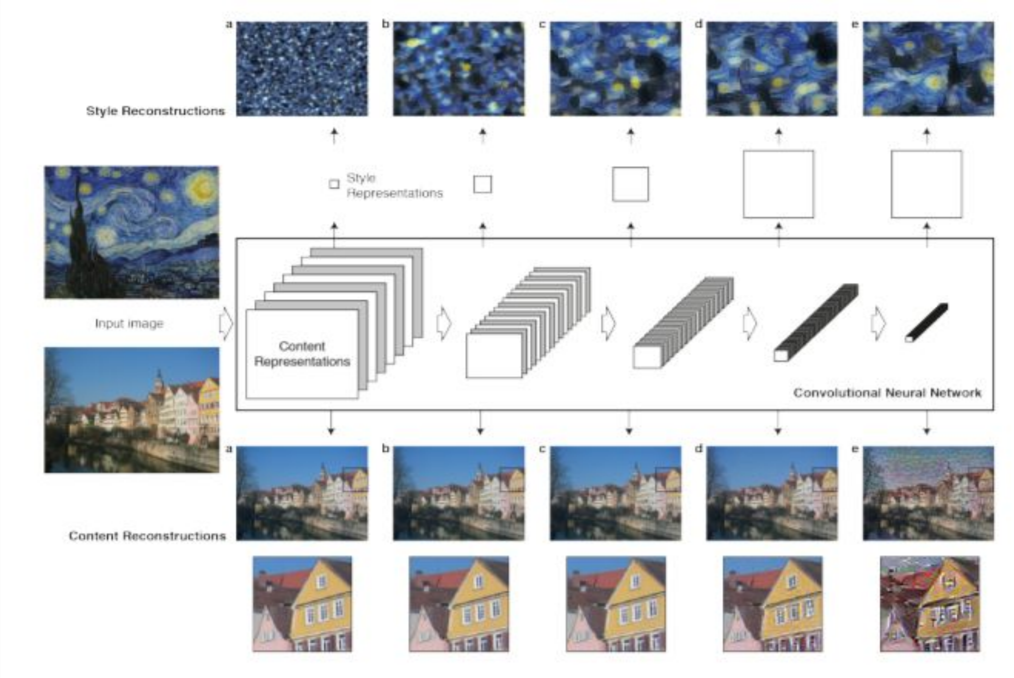
G[i, j] = Dot Product of feature map i and feature map j

Math Intuition



Dot product can be seen as how similar two vectors actually are. The more similar they are, the lesser the angle between them as in fig (a) or more closer the respective coordinates as in fig(b). In both the cases, the result is large. So the more similar they are, the larger the dot product gets.

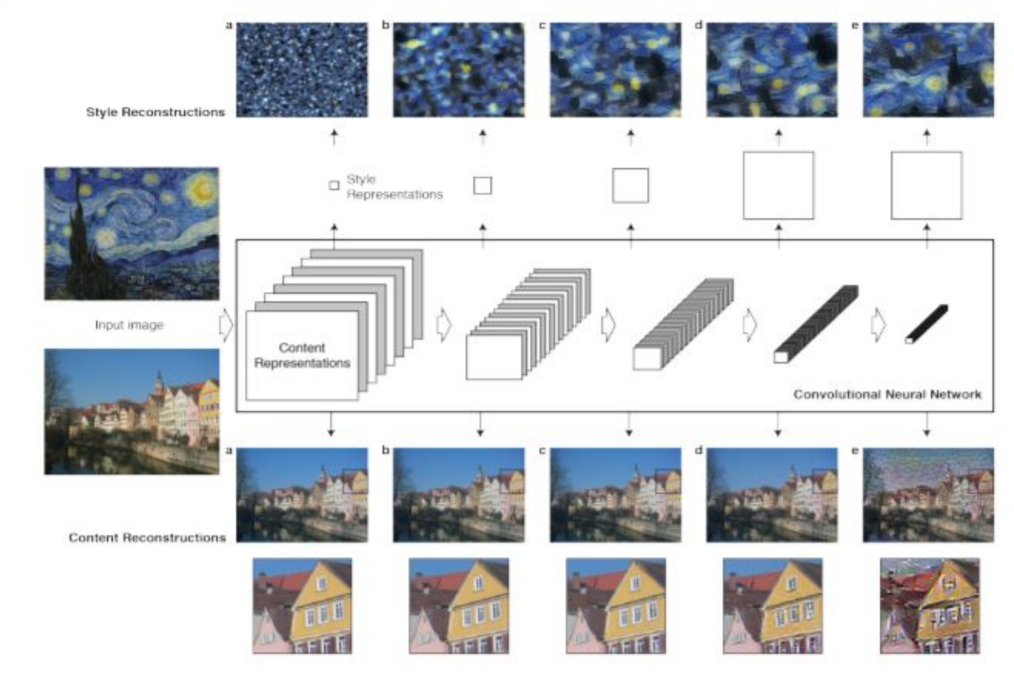
Gatys Approach



$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}.$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - A^l_{ij}\right)^2$$

Gatys Approach



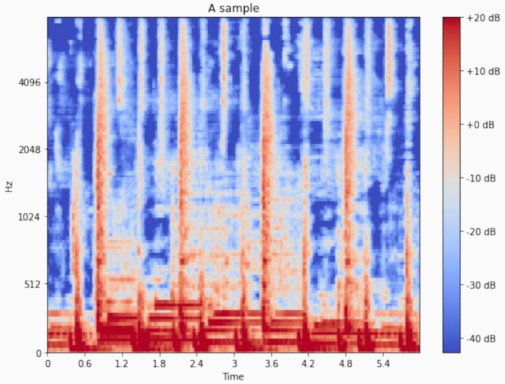
How do we extract the style?

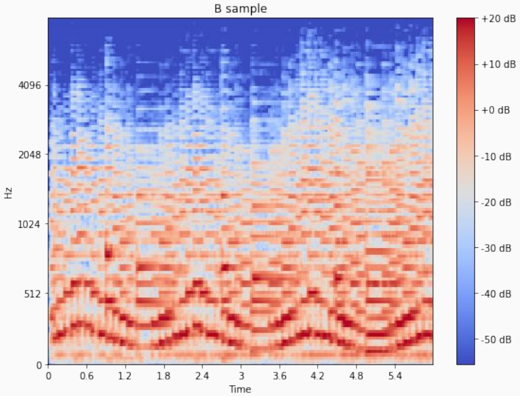
On top of the original CNN representations we built a new feature space that captures the style of an input image. The style representation ***computes correlations between the different features*** *in different layers of the CNN*

Gatys Approach

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

First Attempt

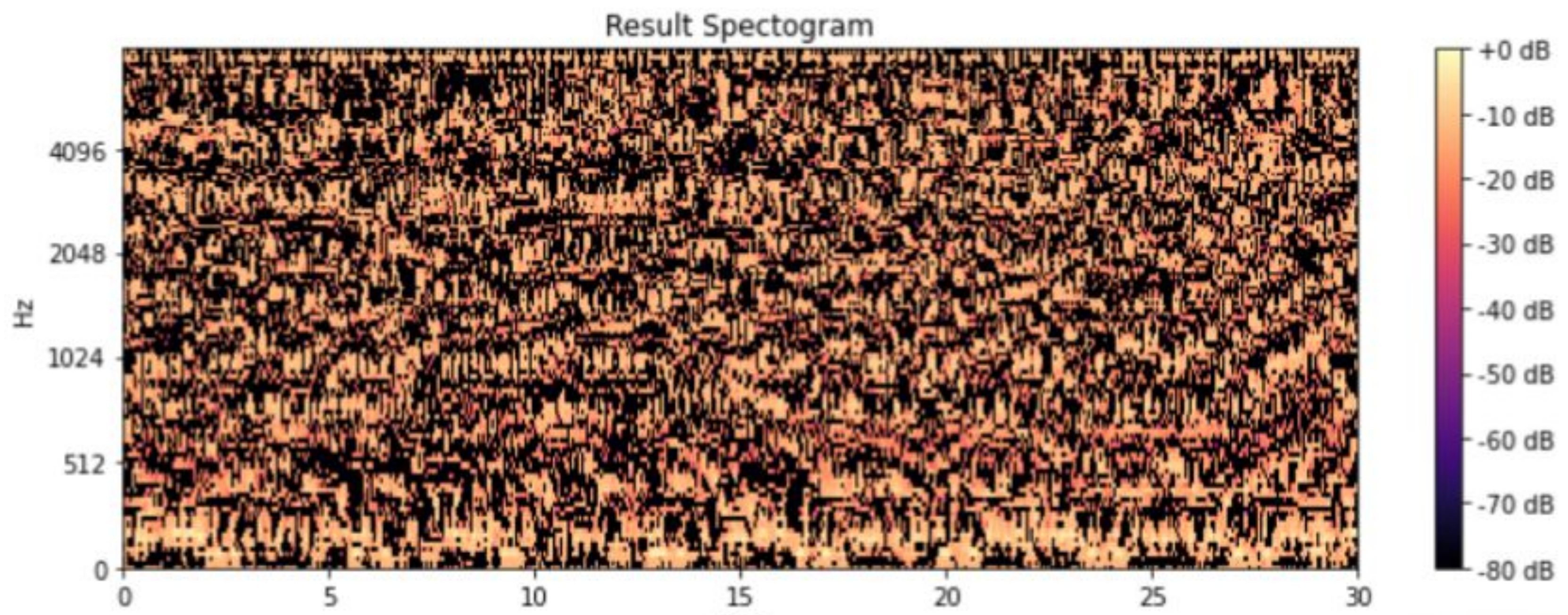




1. Scrape musical clips using Spotify API
2. Transform the audio data in a mel-spectrogram
3. Do a small hack with the VGG19 input
4. VGG19 will just treat the spectrograms as any plain old photo.
5. ???
6. Profit

A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time.

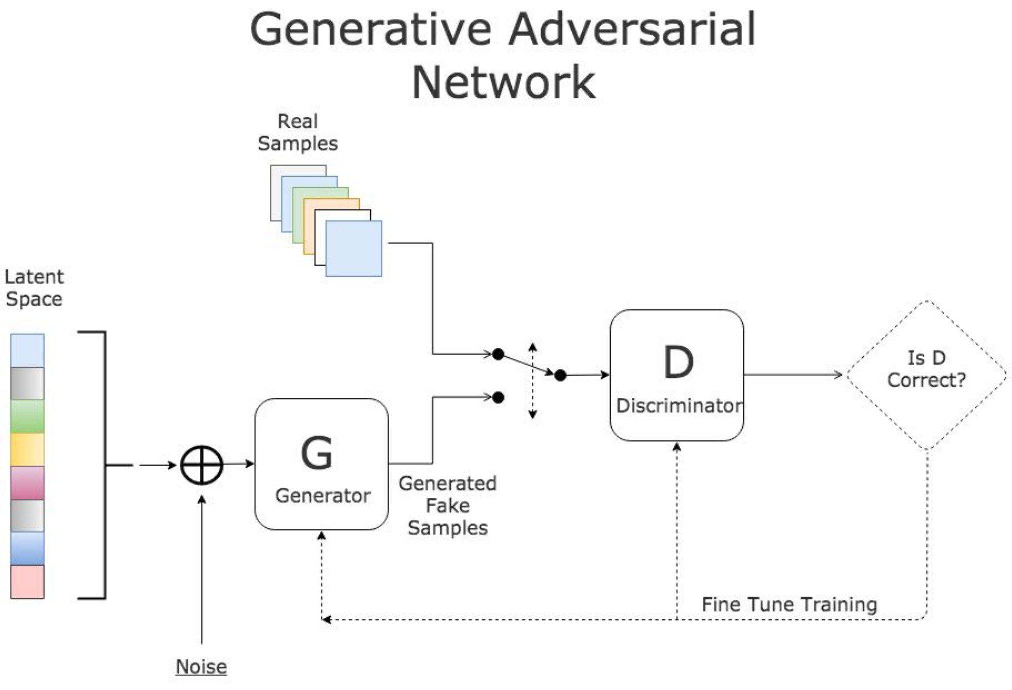
First Attempt



Why did it fail?

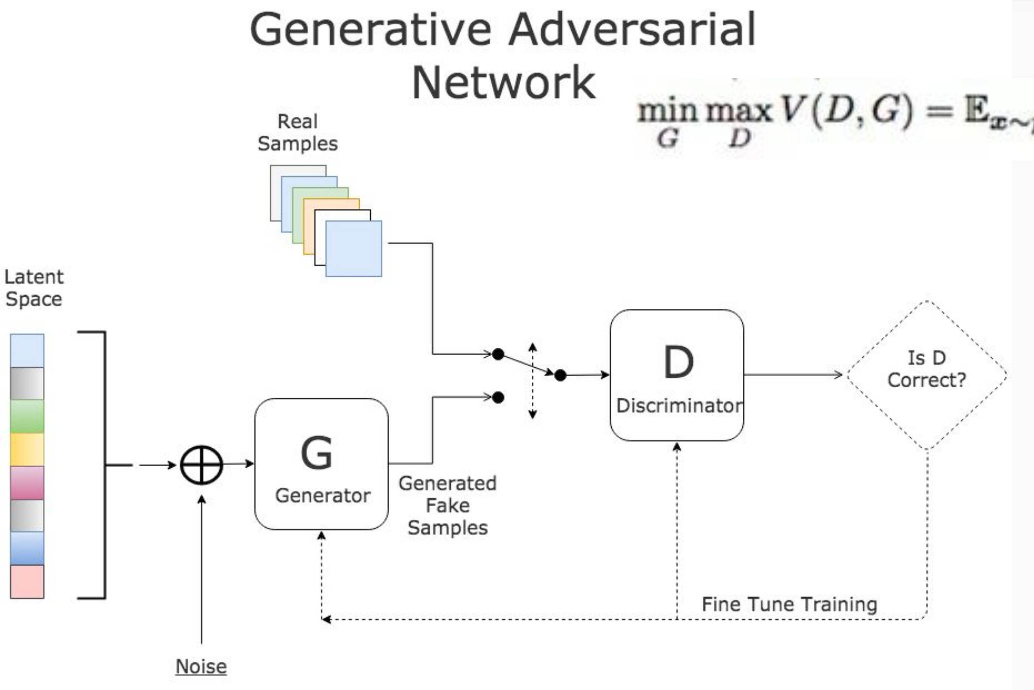
Now What?

GAN in a nutshell



You can imagine the training of a GAN as the attempts of a con artist to fool the curator of a museum with fake paintings.

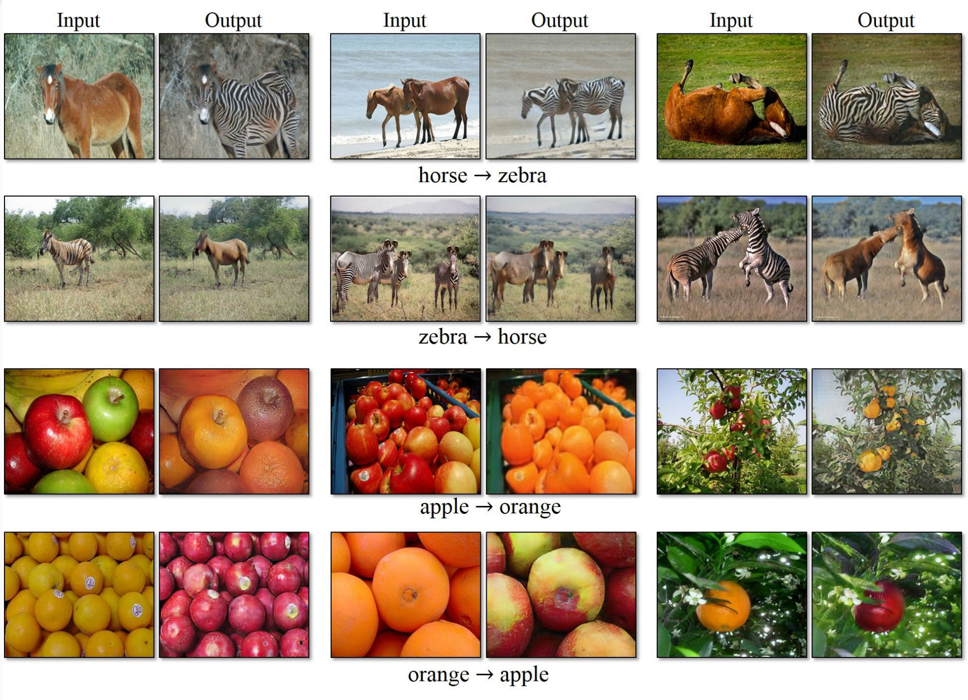
GAN in a nutshell



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

Mathematically, G attempts to transform a vector space A to the "real" vector space B.

GAN in a nutshell



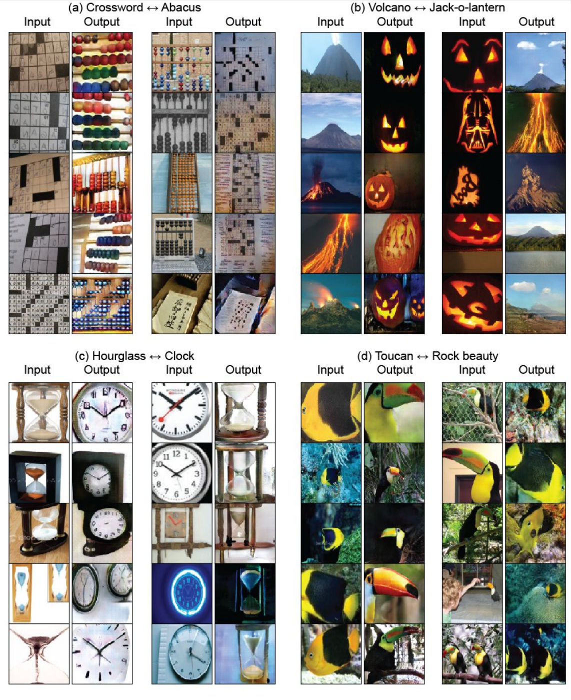
Standard procedures often lead to the well known problem of mode collapse, where all input images map to the same output image and the optimization fails to make progress.

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

The achievements of these models have been limited to a particular subset of domains where this assumption yields good results, namely homogeneous domains that are characterized by style or texture differences.

TraVeLGAN: Image-to-image Translation by Transformation Vector Learning

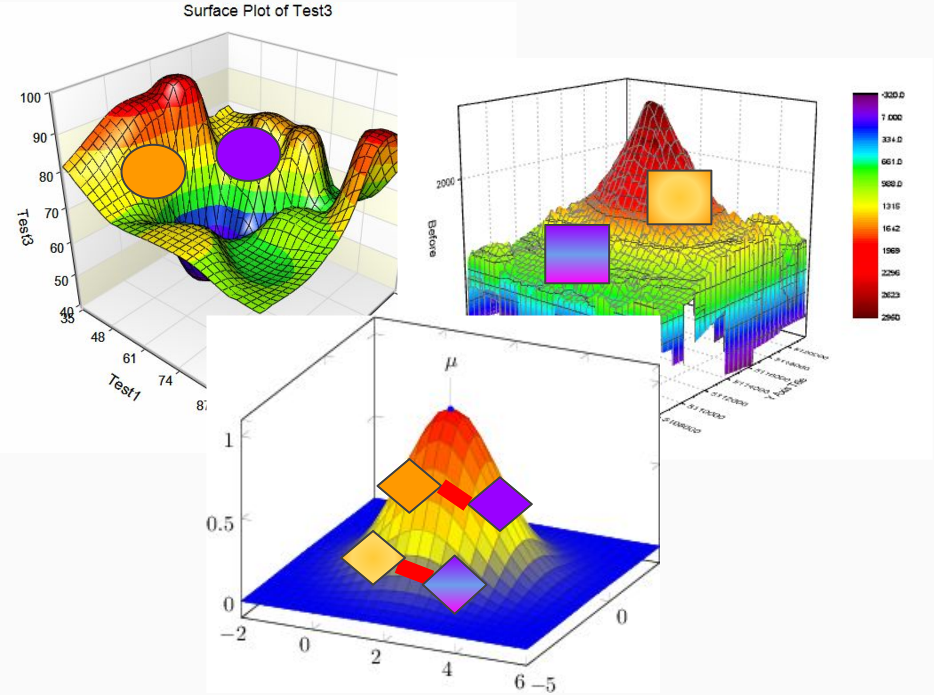
Amodio-Krishnaswamy Approach



Transformation **Ve**ctor **L**earning **GAN**

The TraVeLGAN uses a third network, a siamese network, in addition to the generator and discriminator to produce a latent space of the data to capture high-level semantics characterizing the domains. This space guides the generator during training, by forcing the generator to preserve vector arithmetic between points in this space

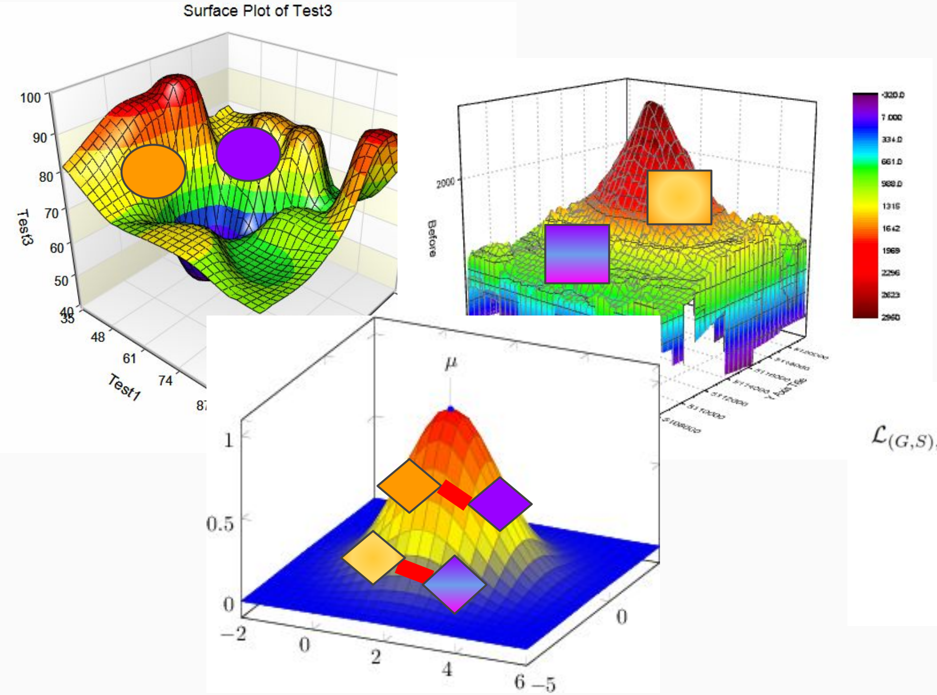
Amodio-Krishnaswamy Approach



Transformation **Ve**ctor **L**earning **GAN**

The TraVeLGAN uses a third network, a siamese network, in addition to the generator and discriminator to produce a latent space of the data to capture high-level semantics characterizing the domains. This space guides the generator during training, by forcing the generator to preserve vector arithmetic between points in this space.

Amodio-Krishnaswamy Approach



Transformation **Ve**ctor **L**earning **GAN**

Transformation vector that connects encodings of a pair of source images must be equal to the transformation vector between the same pair translated by the generator

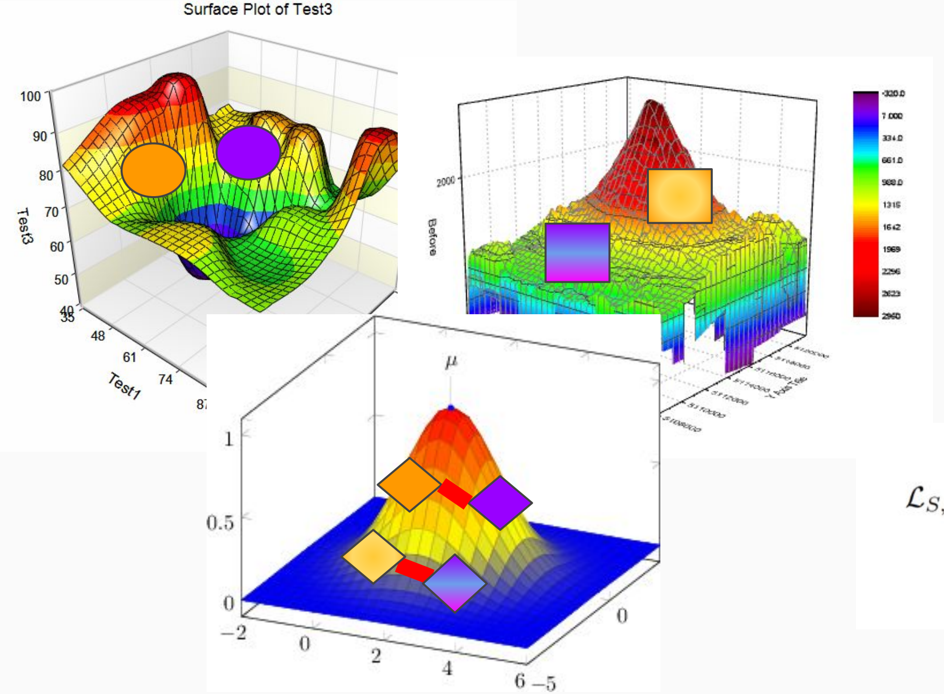
$$\mathcal{L}_{(G,S),TraVeL} = \mathbb{E}_{(a_{\frac{L}{2},1}, a_{\frac{L}{2},2})\sim A}[cosine\_similarity(t_{12}, t'_{12}) + ||t_{12} - t'_{12}||_2^2)] \quad with\ a_{\frac{L}{2},1} \neq a_{\frac{L}{2},2}$$

$$t_{ij} = S(a_{\frac{L}{2},i}) - S(a_{\frac{L}{2},j})$$

$$t'_{ij} = S(G(a_{\frac{L}{2},i})) - S(G(a_{\frac{L}{2},j}))$$

Formal TraVeL loss

Amodio-Krishnaswamy Approach



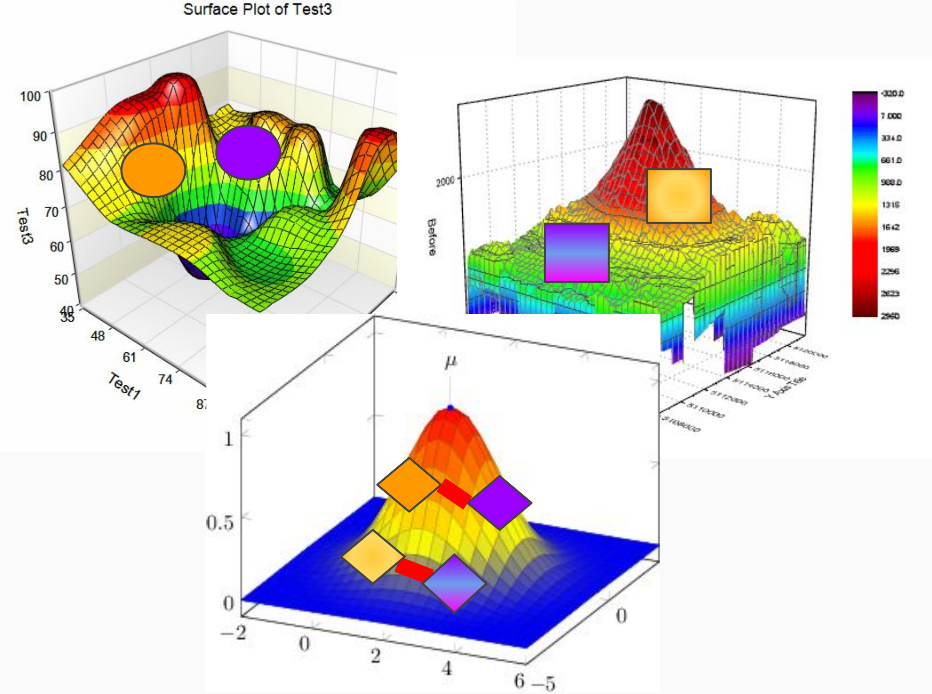
Transformation **Ve**ctor **L**earning **GAN**

The margin loss keeps all the vectors produced by S far from one another, so that the network can't associate the same exact vector to every input and must learn meaningful relationships creating a useful latent space

$$\mathcal{L}_{S,margin} = \mathbb{E}_{(a_{\frac{L}{2},1}, a_{\frac{L}{2},2}) \sim A} max(0, (\delta - ||t_{12}||_2)) \quad with\ a_{\frac{L}{2},1} \neq a_{\frac{L}{2},2}$$

where delta is a fixed value and t is the transformation vector

Amodio-Krishnaswamy Approach



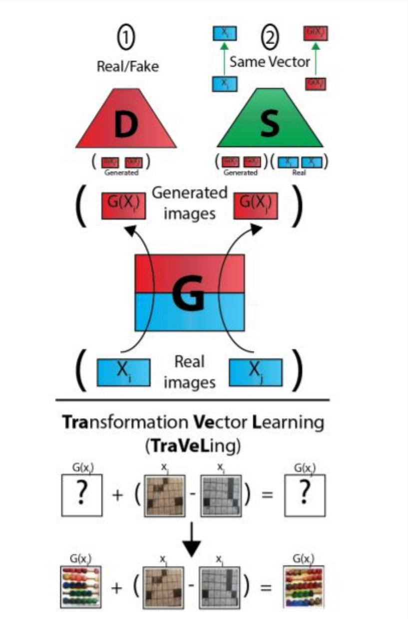
$$\mathcal{L}_D = \mathcal{L}_{D,adv}$$

$$\mathcal{L}_G = \mathcal{L}_{G,adv} + \alpha \mathcal{L}_{G,id} + \beta \mathcal{L}_{(G,S),TraVeL}$$

$$\mathcal{L}_S = \beta \mathcal{L}_{(G,S),TraVeL} + \gamma \mathcal{L}_{S,margin}$$

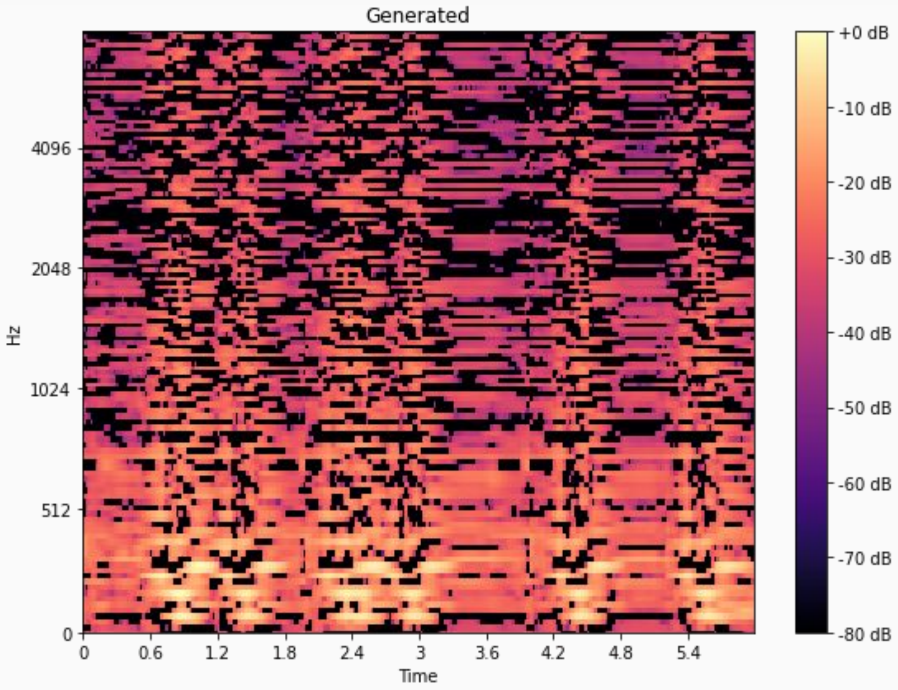
Final losses for generator G, discriminator D, siamese network S

Second Attempt

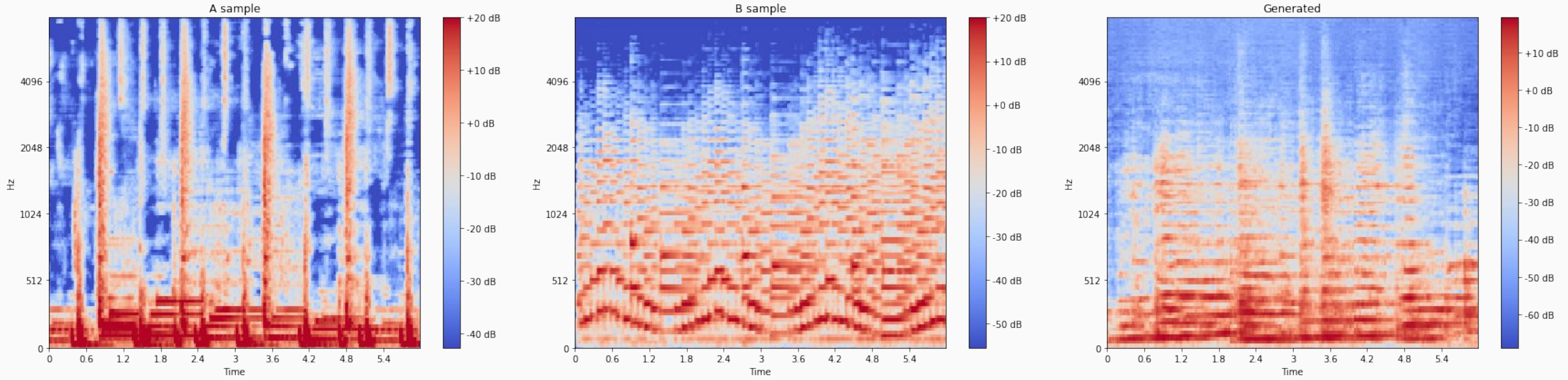


1. Scrape musical clips using Spotify API
2. Transform the audio data in a mel-spectrogram
3. Split the mel-spectrograms into 6 smaller spectrograms for faster training
4. For each spectrogram, split it in halves
5. Obtain the transformations for the two halves
6. Reunite the halves to obtain the entire sample transformation

Second Attempt



Second Attempt



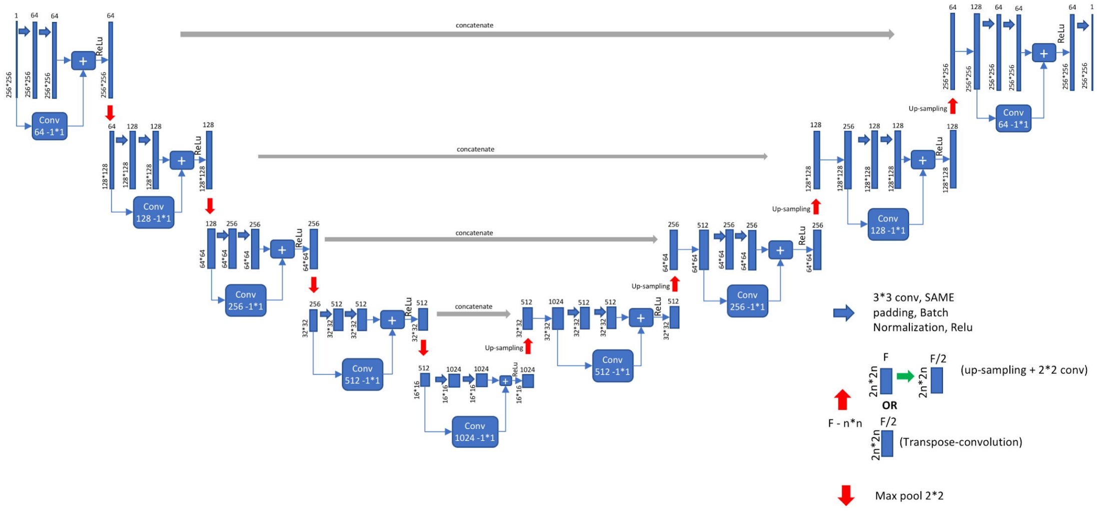
Friendly reminder to normalise your data

Second Attempt

Model: "G"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 128, 86, 1)]	0	
zero_padding2d (ZeroPadding2D)	(None, 128, 88, 1)	0	input_1[0][0]
conv_s_n2d (ConvSN2D)	(None, 1, 86, 256)	98560	zero_padding2d[0][0]
batch_normalization (BatchNorma	(None, 1, 86, 256)	1024	conv_s_n2d[0][0]
leaky_re_lu (LeakyReLU)	(None, 1, 86, 256)	0	batch_normalization[0][0]
zero_padding2d_1 (ZeroPadding2D	(None, 1, 88, 256)	0	leaky_re_lu[0][0]
conv_s_n2d_1 (ConvSN2D)	(None, 1, 44, 256)	590080	zero_padding2d_1[0][0]
batch_normalization_1 (BatchNor	(None, 1, 44, 256)	1024	conv_s_n2d_1[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 1, 44, 256)	0	batch_normalization_1[0][0]
conv_s_n2d_2 (ConvSN2D)	(None, 1, 22, 256)	459008	leaky_re_lu_1[0][0]
batch_normalization_2 (BatchNor	(None, 1, 22, 256)	1024	conv_s_n2d_2[0][0]
leaky_re_lu_2 (LeakyReLU)	(None, 1, 22, 256)	0	batch_normalization_2[0][0]
up_sampling2d (UpSampling2D)	(None, 1, 44, 256)	0	leaky_re_lu_2[0][0]
conv_s_n2d_3 (ConvSN2D)	(None, 1, 44, 256)	459008	up_sampling2d[0][0]
batch_normalization_3 (BatchNor	(None, 1, 44, 256)	1024	conv_s_n2d_3[0][0]
leaky_re_lu_3 (LeakyReLU)	(None, 1, 44, 256)	0	batch_normalization_3[0][0]
concatenate (Concatenate)	(None, 1, 44, 512)	0	leaky_re_lu_3[0][0] leaky_re_lu_1[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 1, 88, 512)	0	concatenate[0][0]
conv_s_n2d_4 (ConvSN2D)	(None, 1, 88, 256)	1179904	up_sampling2d_1[0][0]
leaky_re_lu_4 (LeakyReLU)	(None, 1, 88, 256)	0	conv_s_n2d_4[0][0]
zero_padding2d_2 (ZeroPadding2D	(None, 1, 88, 256)	0	leaky_re_lu[0][0]
concatenate_1 (Concatenate)	(None, 1, 88, 512)	0	leaky_re_lu_4[0][0] zero_padding2d_2[0][0]
conv_s_n2d_transpose (ConvSN2DT	(None, 128, 88, 1)	66049	concatenate_1[0][0]
cropping2d (Cropping2D)	(None, 128, 86, 1)	0	conv_s_n2d_transpose[0][0]

Total params: 2,856,705
Trainable params: 2,852,865
Non-trainable params: 3,840

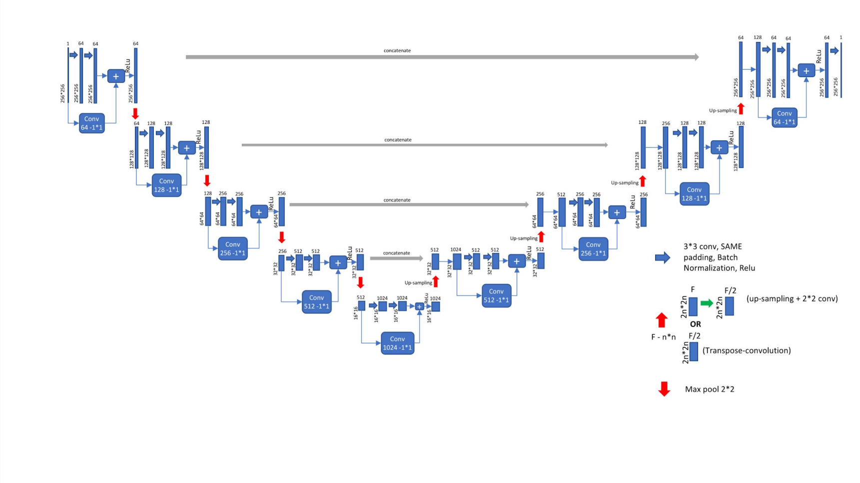


Second Attempt

```
Model: "S"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 128, 86, 1)]      0
_________________________________________________________________
conv_s_n2d_5 (ConvSN2D)      (None, 1, 84, 256)        98560
_________________________________________________________________
batch_normalization_4 (Batch (None, 1, 84, 256)        1024
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 1, 84, 256)        0
_________________________________________________________________
conv_s_n2d_6 (ConvSN2D)      (None, 1, 42, 256)        590080
_________________________________________________________________
batch_normalization_5 (Batch (None, 1, 42, 256)        1024
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 1, 42, 256)        0
_________________________________________________________________
conv_s_n2d_7 (ConvSN2D)      (None, 1, 21, 256)        459008
_________________________________________________________________
batch_normalization_6 (Batch (None, 1, 21, 256)        1024
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 1, 21, 256)        0
_________________________________________________________________
flatten (Flatten)            (None, 5376)              0
_________________________________________________________________
dense (Dense)                (None, 128)               688256
=================================================================
Total params: 1,838,976
Trainable params: 1,836,672
Non-trainable params: 2,304
_________________________________________________________________
```

```
Model: "C"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 128, 258, 1)]     0
_________________________________________________________________
conv_s_n2d_8 (ConvSN2D)      (None, 1, 256, 512)       197120
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 1, 256, 512)       0
_________________________________________________________________
conv_s_n2d_9 (ConvSN2D)      (None, 1, 128, 512)       2359808
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 1, 128, 512)       0
_________________________________________________________________
conv_s_n2d_10 (ConvSN2D)     (None, 1, 64, 512)        1835520
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 1, 64, 512)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 32768)             0
_________________________________________________________________
dense_sn (DenseSN)           (None, 1)                 32770
=================================================================
Total params: 4,425,218
Trainable params: 4,423,681
Non-trainable params: 1,537
_________________________________________________________________
```

Plans for Future



- Generator U-NET architecture resembles an Autoencoder; I'll attempt to build a neural network able to extract the instrumental part from an audio sample
- Post 2 Medium articles: 1 for NST problem, 1 for the problem presented above
- Gain more hands-on knowledge with machine learning on Kaggle

References

1. TraVeLGAN: Image-to-image Translation by Transformation Vector Learning
2. Voice Conversion and Audio Style Transfer on arbitrarily long samples using Spectrograms
3. Spectral Normalization for Generative Adversarial Networks
4. Unsupervised Cross-Domain Image Generation
5. A Neural Algorithm of Artistic Style

Q & A