Mono/Bi Dimensional Signal Processing and DGAN

Digital Signal & Image Management project

Samir Doghmi - 897358 Luca Iarocci - 894066 Roberto Ferrari - 852220

Contents

01 Mono-Dimensional Signal Classifiction

01 Problem definition

02 metadata selection and exploration

03 Audio processing and segmentation

04 Classification

05 Demo

1. Problem definition

COUGHVID is a crowdsourced dataset of 34,000+ cough recordings. Each recording is described by a set of **self-reported** meatadata:

- record quality (SNR, cough detection score)
- time and place of recording
- subject's demographic features (gender, age)
- subject's health condition:
  - status (COVID-19, symptomatic, or healthy)
  - fever or muscle pain
  - existing respiratory condition

COUGHVID

2,000 expert annotated samples: a subset of recordings is annotated by physicians, provinding for each one further medical details on coughs' nature and severity, and a diagnosis.

Objective: develop a multi-class classification model capable of classifying recordings by subject's **status** with comparable or better results than expert physician.

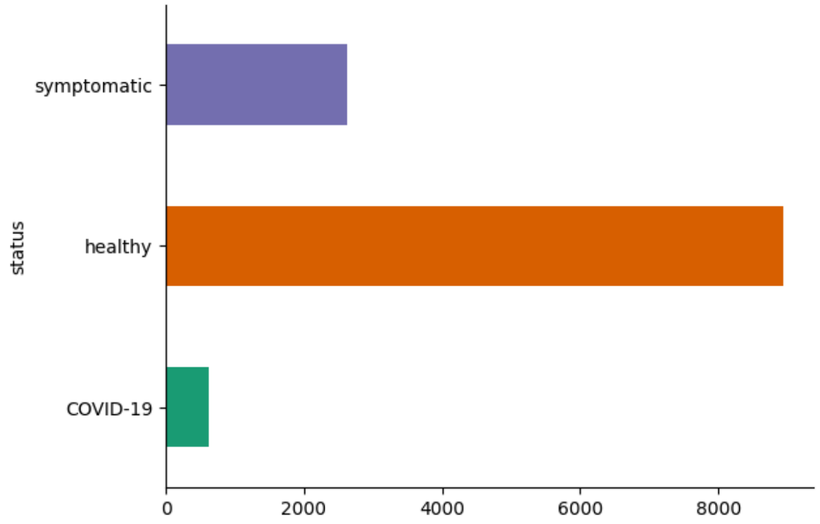
2. Metadata selection and exploration

Matadata based selection

- cough detection score > 0.8 (following dataset creators suggestion → 4.6% FP rate)
- completeness, exclude records with missing demographic info or status
- coherence, exclude records with self-reported healty status and fever at the same time

About **1/3** of the original records met the above requirements.

Exploration



Subjects have a median age of 35 and about 2/3 of them are males.
The target variable status shows an extreme class unmbalaced:

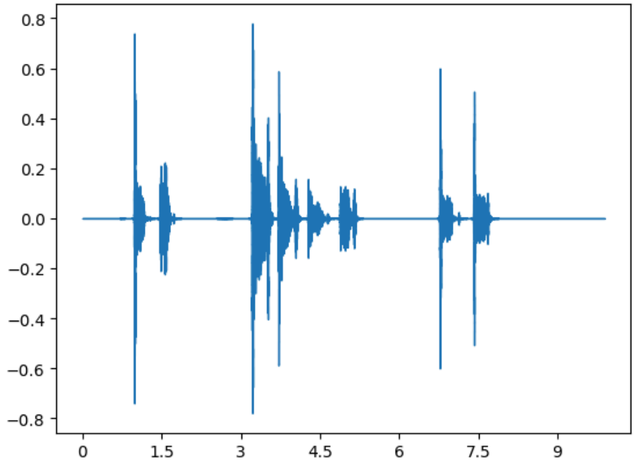
- 5.1% COVID-19
- 73.4% healthy
- 21.5% symptomatic

3. Audio processing and segmentation

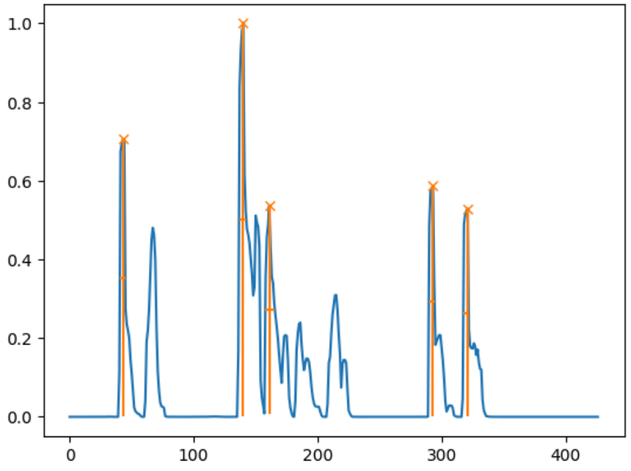
Spectrograms were used to extract audio features from cough recordings. Two approcheas were used for normalizing recording lengths to obtain features of the same dimensions for each recording:

- zero padding or trimming to a target lenght of 10s (roughly the median lenght of recordings)
- extract 2s clip(s) from audios when a coughs is detected (heuristic using energy)

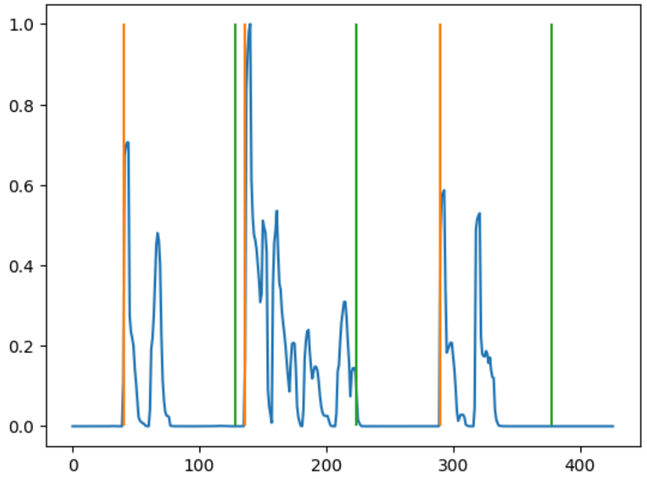
waveform



energy: find peaks



energy: cut clips

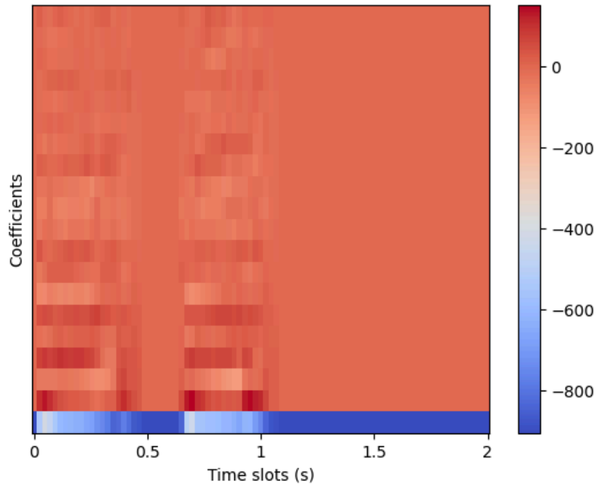


3. Audio processing and segmentation

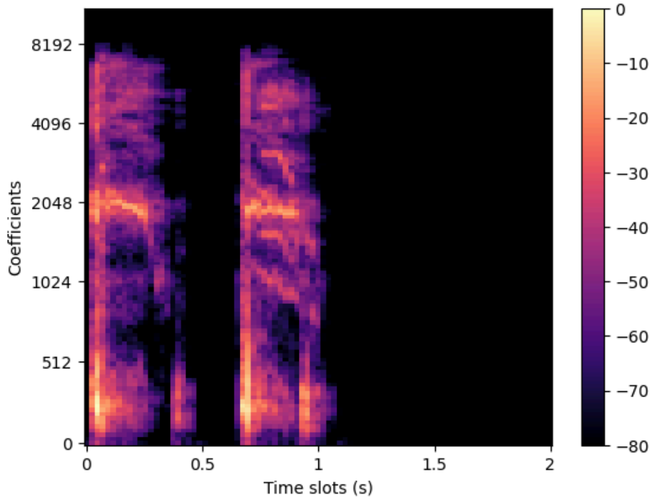
Spectrograms were used to extract audio features from cough recordings. Two approcheas were used for normalizing recording lengths to obtain features of the same dimensions for each recording:

- zero padding or trimming to a target lenght of 10s (roughly the median lenght of recordings)
- extract 2s clip(s) from audios when a coughs is detected (heuristic using energy)

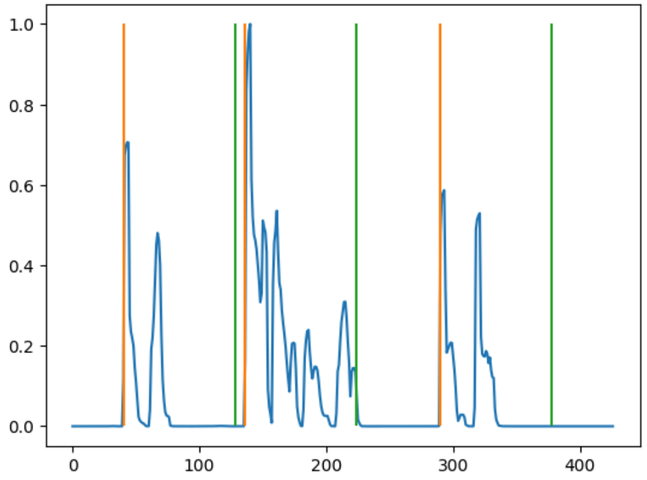
Mel spectrogram



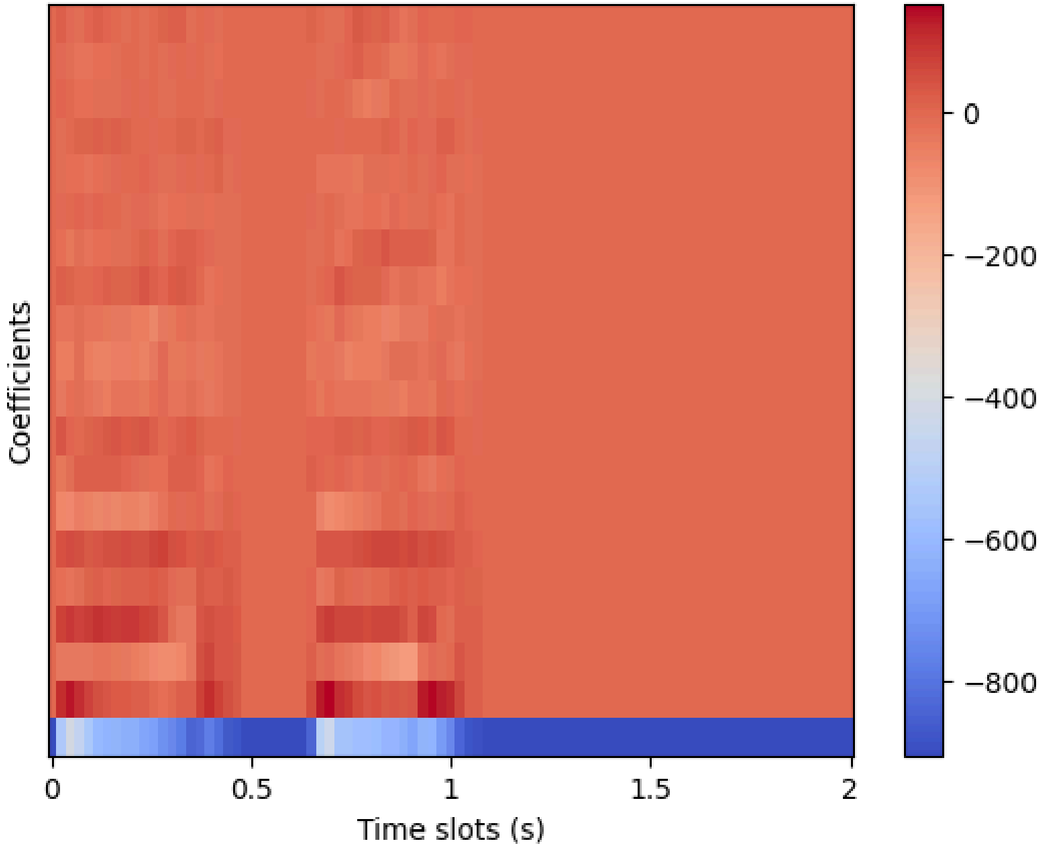
MFCC



energy: cut clips



3. Audio processing and segmentation



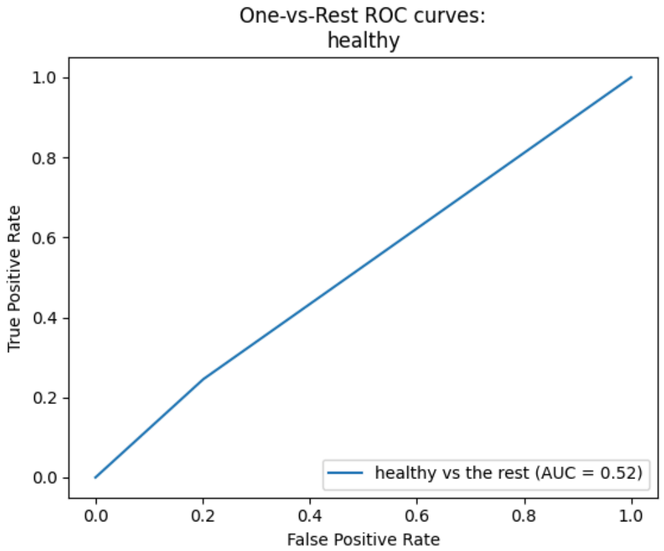
Feature extraction from spectrograms

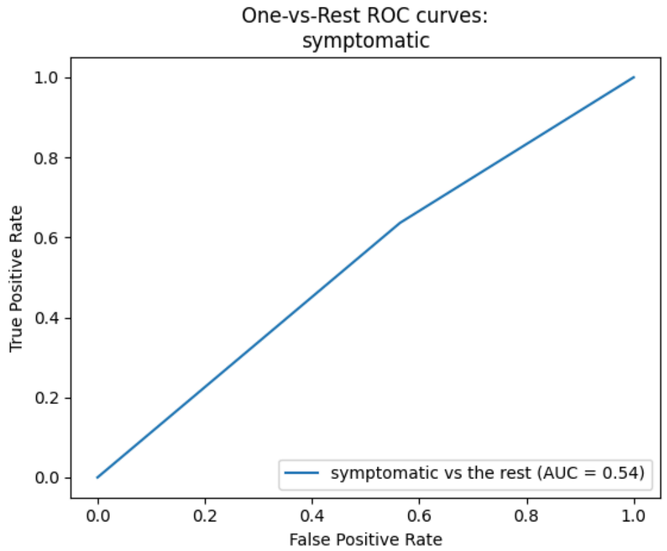
A series of summary statistics were extracted from spectrograms and used as feature for classification, namely:

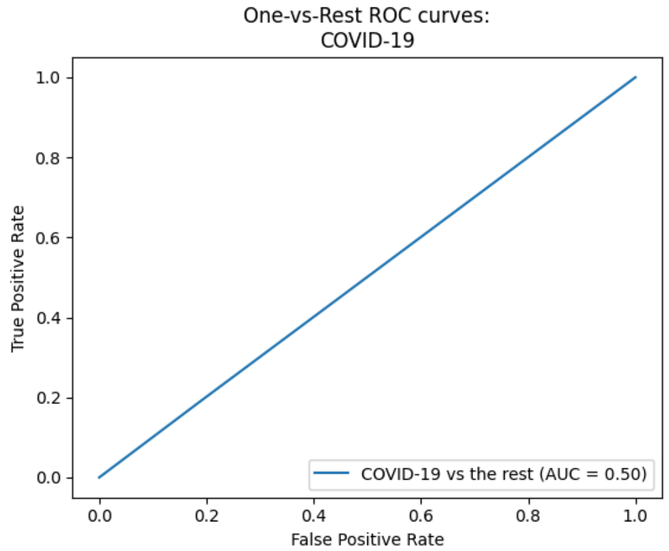
- row and column wise mean
- row and column wise averege
- row and column wise maximum
- row and column wise standard deviation

4. Classification: baseline

I established a benchmark by comparing patient self-reported status and expert diagnosis.







	precision	recall	f1-score	support
COVID-19	0.26	0.19	0.22	509
healthy	0.35	0.25	0.29	611
symptomatic	0.47	0.64	0.54	877
accuracy			0.40	1997
macro avg	0.36	0.36	0.35	1997
weighted avg	0.38	0.40	0.38	1997

It appears that, even for expert physicians, formulating a correct diagnosis based on a cough recording it's a relatively hard task.

4. Classification: ML model results

A histogram-based gradient boosting ensembles classification model was implemented, trained and tested on the same train/test split for all combination of audio segmentation techniques and spectrogram type.
The split was made on a record base to prevent data leakege from same audio clips beign both in train and test.

Overall no specific method outperformed any other, achiving results that could be further improved. Still, the ML classifier performed better than the baseline.

		AUC one-vs-rest		
		COVID-19	healthy	symptomatic
Whole audio	MFCC	0.59	0.60	0.60
	Mel spectrogram	0.58	0.60	0.60
2 seconds clips	MFCC	0.59	0.61	0.62
	Mel spectrogram	0.59	0.59	0.62

5. Demo

02 Bi-Dimensional Signal Classification

01 Data and Preprocessing

02 Image Augmentation

03 Classification: On Scratch

04 Classification: Fine Tuning

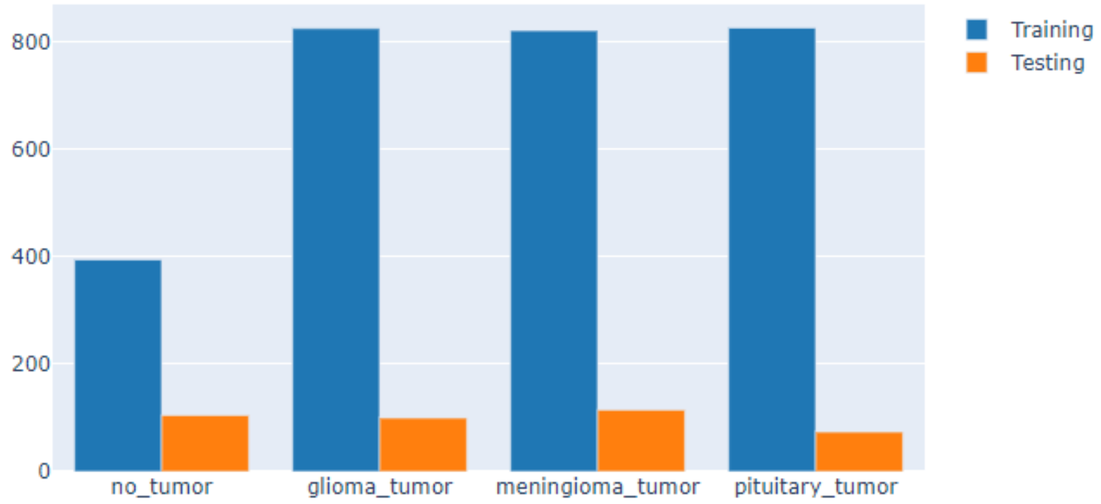
05 VGG-19 Results

06 Demo

2.1 Data and Preprocessing

The brain tumor dataset, provided as a set of slices, from Kaggle, contains 3064 T1-weighted constrast-enhanced MRI images. It's made up by four classes:

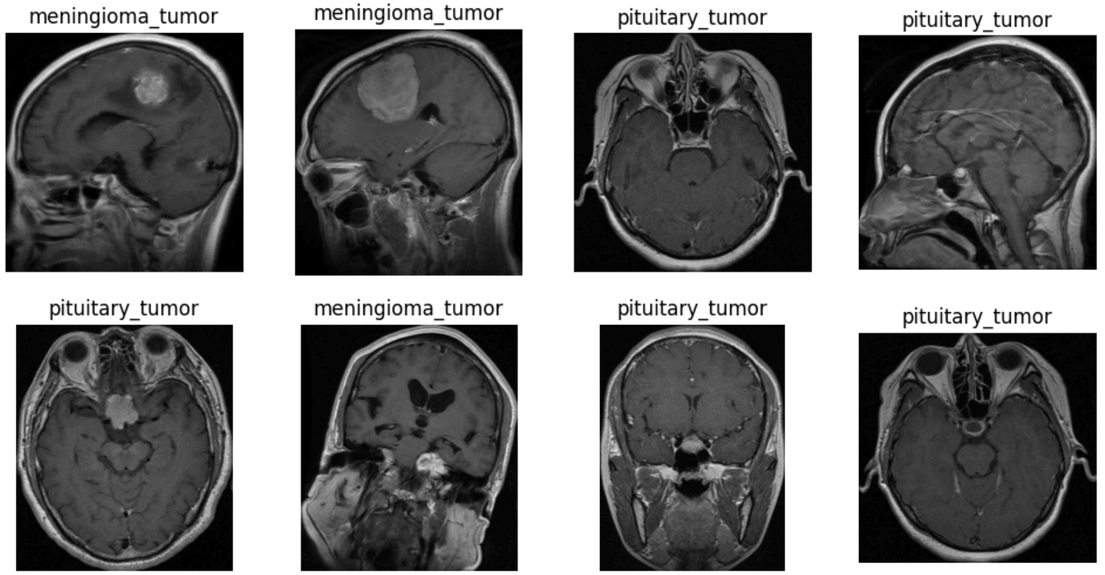
Class	Description
Glioma tumor	It's a type of tumor that starts in the glial cells of the brain or the spine
Maligant tumor	It contrasts with a non-cancerous benign tumor in that a malignancy is not self-limited in its growth. It's capable of invading into adjacent tissues.
Pituary tumor	It's a tumor that occur in the pituitary gland
No tumor	The Brain scan is normal. No Tumor is detected



2.1 Data and Preprocessing

The pre-processing of the images was carried out with the intention of improving the performance of the models by considering the edges of the images:

- **Converting the image to a single channel grayscale image**. It makes things easy for the contour-detection algorithm
- Use the findContours() function to detect the contours in the image
- Draw Contours on a copy of the original RGB Image
- Find bounding rectangle for the **outermost contour**
- Draw the bounding rectangle on the image with contours
- **Crop** the image using the detected contours areas

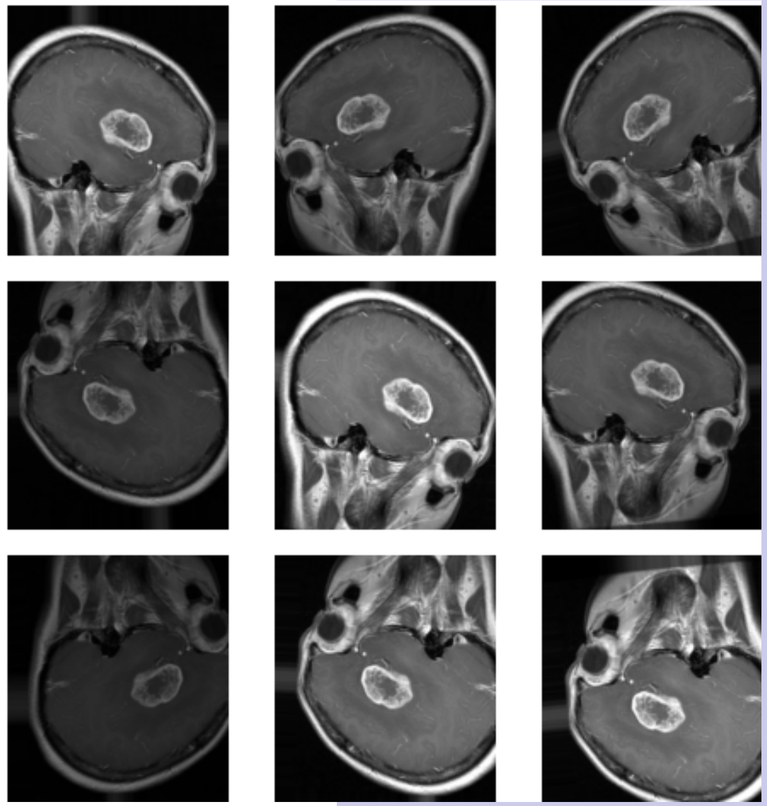


2.2 Image Augmentation

Image augmentation alters images in a way that can help prevent overfitting and increase the robustness of the model.

To avoid overfitting:

- rescale=1./255
- rotation_range=15
- width_shift_range=0.1
- height_shift_range=0.1
- shear_range=0.1
- brightness_range=[0.5, 1.5]
- horizontal_flip=True
- vertical_flip=True



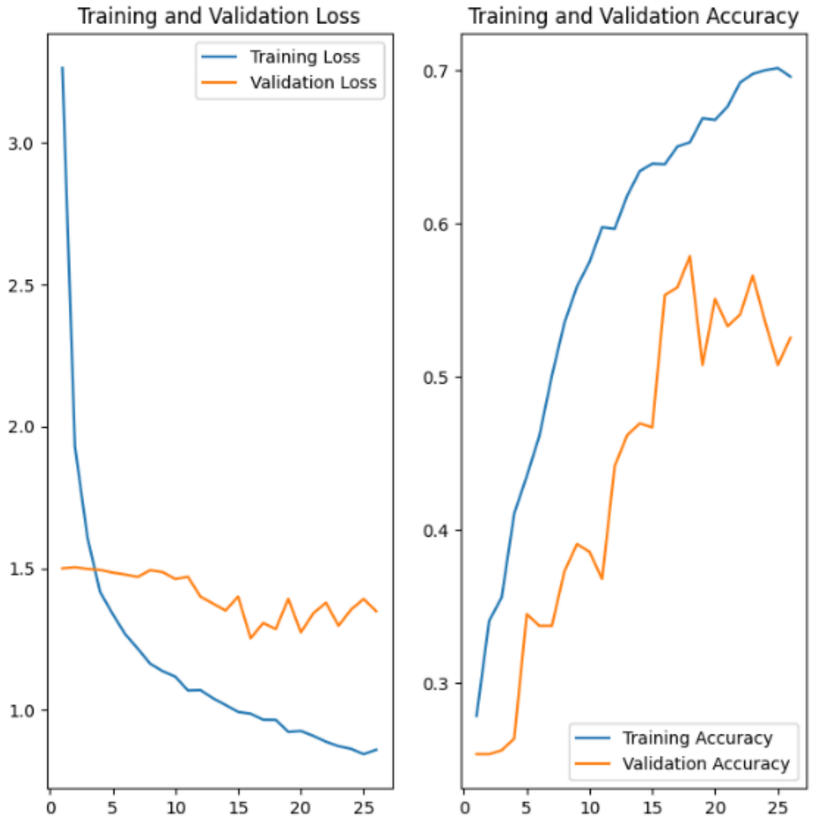
2.3 Classification: On Scratch

The created model built and trained from scratch included 4 convolutional layers, 2 max-pooling layers and 2 dense layer with a total of 2,082,180 parameters.

To mitigate overfitting, we introduce L2. regularization, dropout layers, and the model have been stopped if the val_loss remained stable for 5 epochs

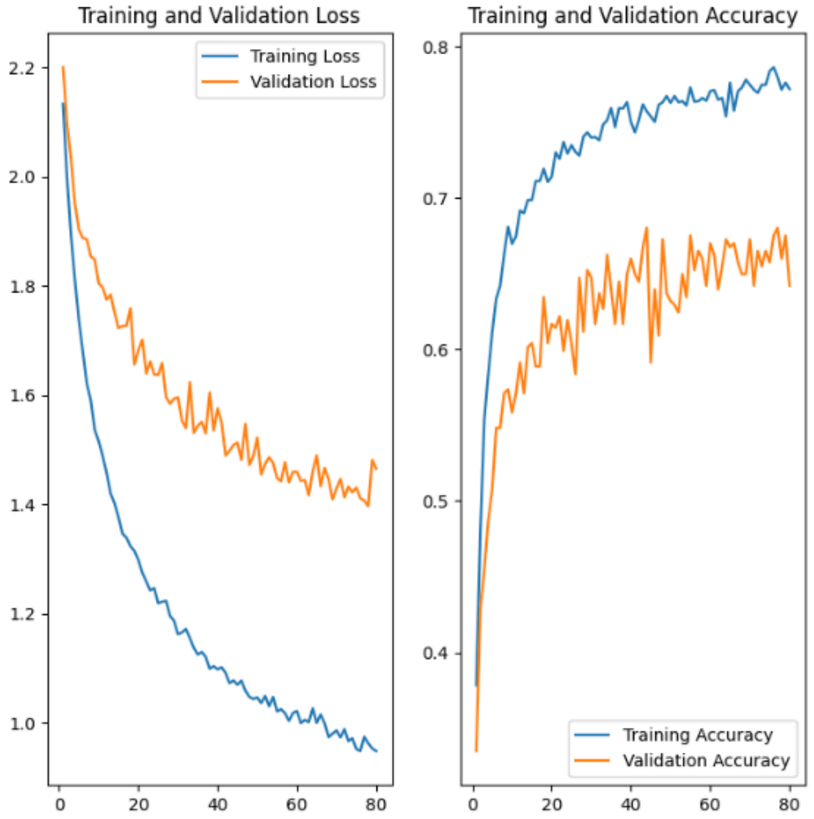
Parameters

- Input size = (224, 224,3)
- Batch size = 64
- Epoch = 26
- Optimizer =Adam
- Loss function = Categorical crossentropy
- Learning rate = 5e-5



2.4 Classification: Fine Tuning

We conducted several fine-tuning experiments with different models, such as VGG16, ResNet50, InceptionResNetV2, and VGG19. We achieved the best performance with VGG19, reaching a validation loss of 1.39 and an accuracy of 0.66.



Parameters

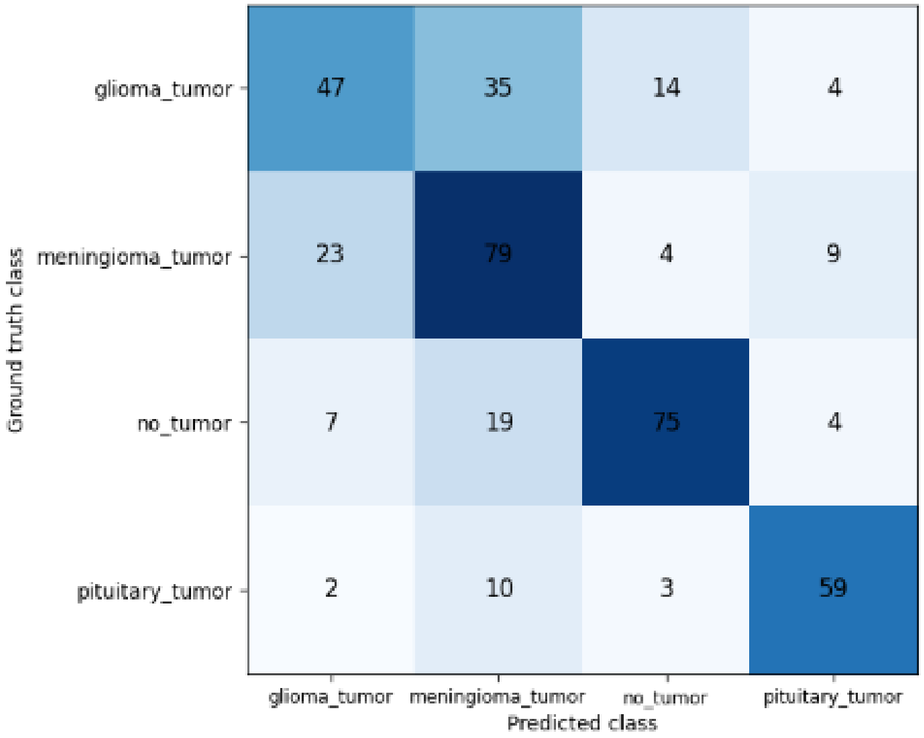
- Input size = (224, 224,3)
- Batch size = 64
- Epoch = 80
- Optimizer =Adam
- Loss function = Categorical crossentropy
- Learning rate = 5e-5
- unfreezing the BatchNormalization layers

2.5 VGG-19 Results

The precision and recall values for the classes meningioma tumour, no tumour and pituitary tumour show some degree of variability.

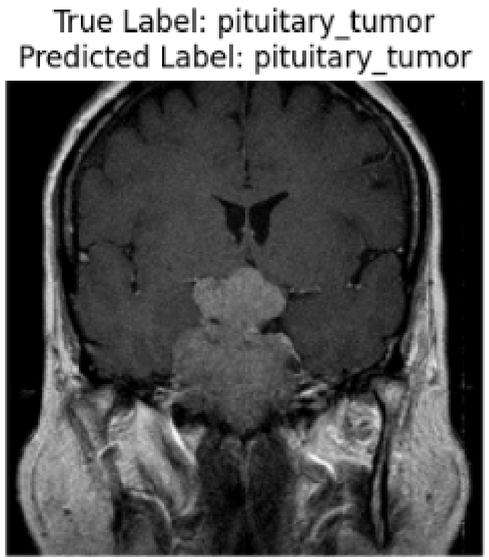
However, the model recall for glioma tumours of 0.47 shows a relatively lower value compared to the other classes, indicating a need for improved detection of glioma tumours.

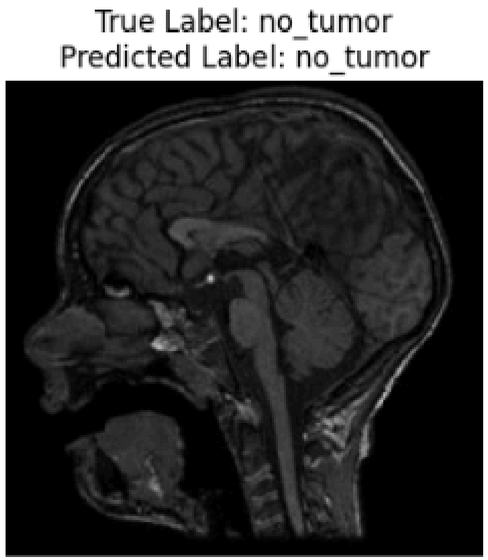
	precision	recall	f1-score	support
glioma_tumor	0.59	0.47	0.53	100
meningioma_tumor	0.55	0.69	0.61	115
no_tumor	0.78	0.71	0.75	105
pituitary_tumor	0.78	0.80	0.79	74
accuracy			0.66	394
macro avg	0.68	0.67	0.67	394
weighted avg	0.67	0.66	0.66	394

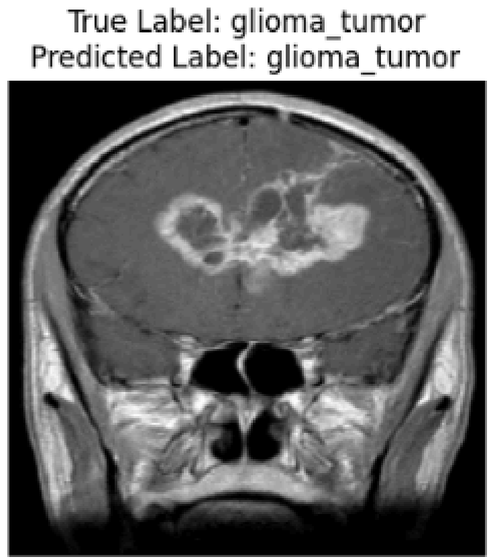


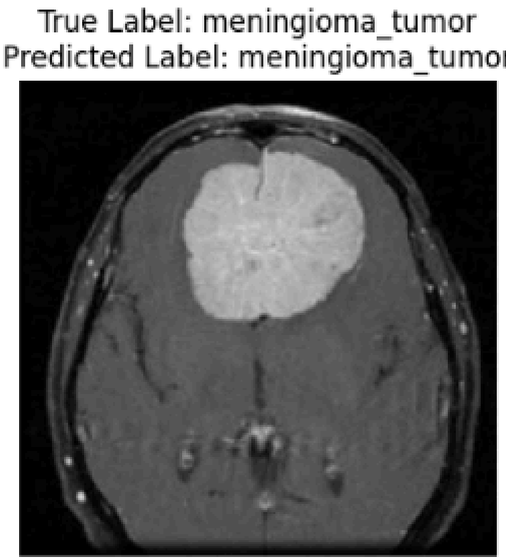
2.6 Demo

We conducted an inference on four images, each representing one of the four classes, by retrieving them from the testing set. The predicted labels for these specific images were surprisingly correct.









03 DCGAN

01 GAN and DCGAN

02 Generator architecture

03 Discriminator architecture

04 Training

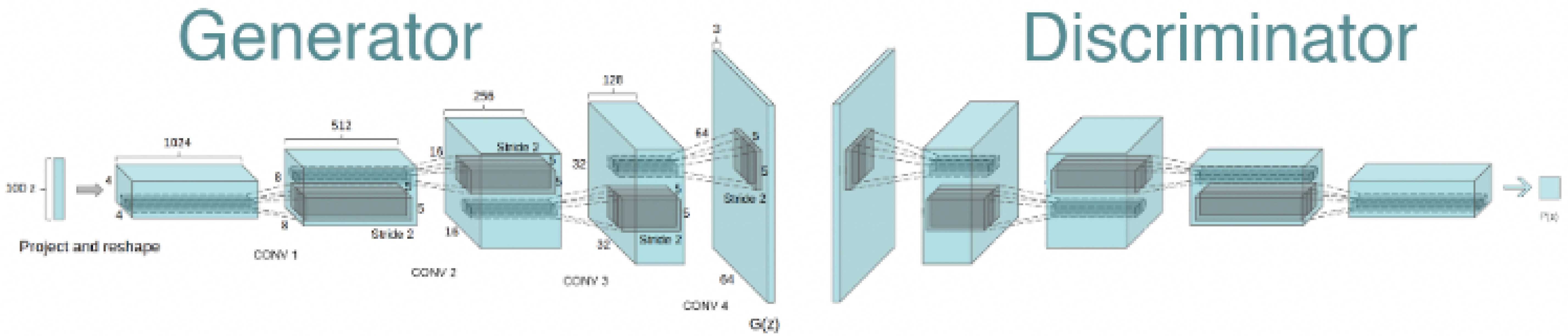
05 G-Loss and D-Loss

06 Final Results

3.1 GAN and DCGAN

DCGAN (Deep Convolutional GAN) is a particular **type of GAN** that implements **deep convolutional** neural networks, instead of fully connected layers, which better capture local structures in images, resulting in improved image quality and stability during training.

For the realization of this project the real world examples on which the model components will be trained are 30'000 images of celebrity faces, already set to a 256x256 format. The **goal** is to generate **new human faces** that, of course, do not exist.

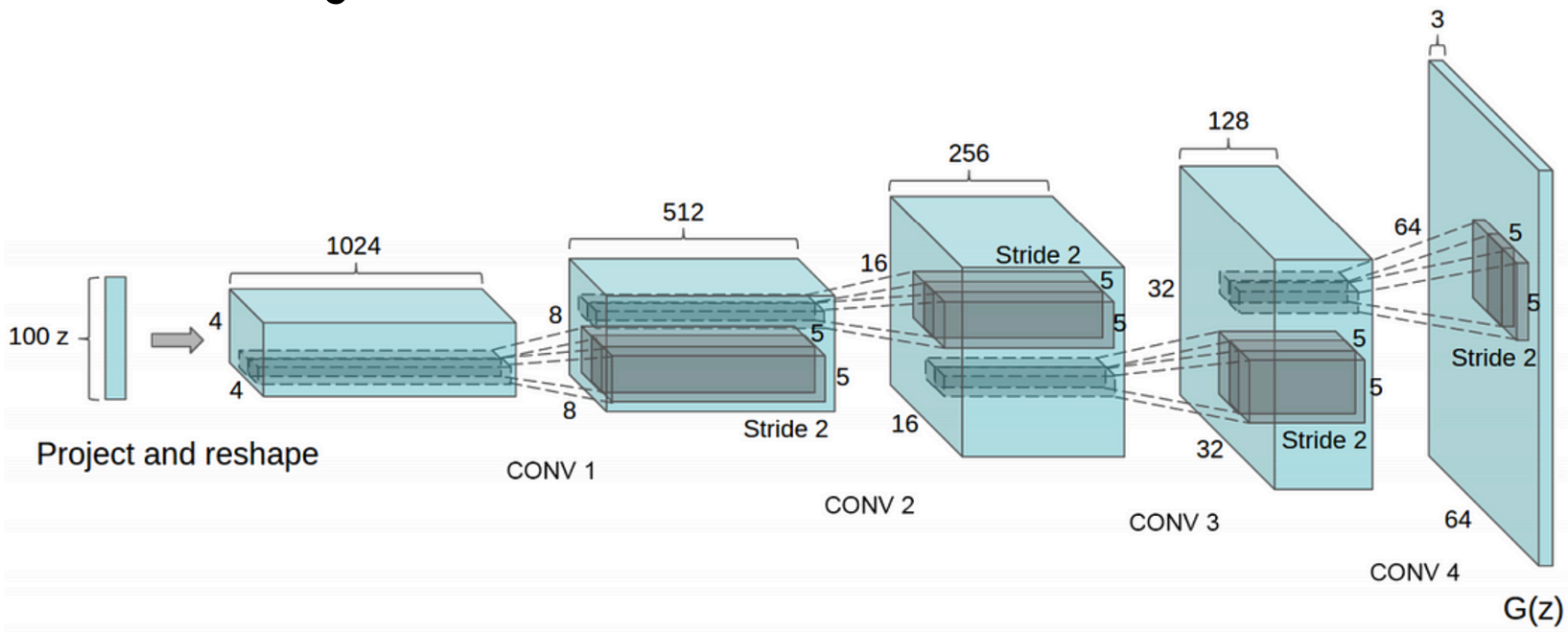


3.2 Generator architecture

In the context of a DCGAN, the **generator** follows a **deep convolutional architecture**, meaning that a de-convolution (or inverse convolution) takes place at each layer, progressively generating bigger feature maps as the spatial dimension increases: by taking as input a latent vector **z**, the spatial dimension is increased starting from a 4x4 feature map and reaching a **final 64x64 image**.

Structure and parameters :

- **layers: 5**
- **latent vector (z): 100**
- **kernel size: 4**
- **stride: 2**
- **padding: 1** (except first layer, that is 0)

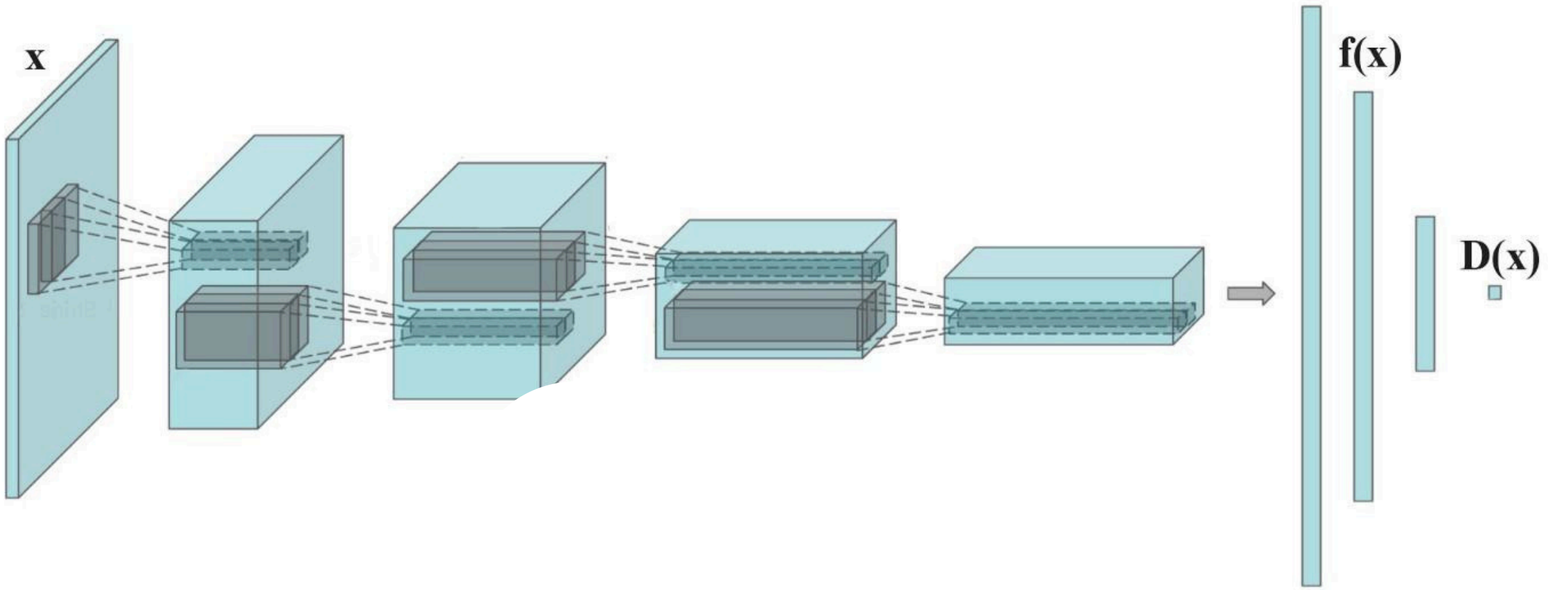


3.3 Discriminator architecture

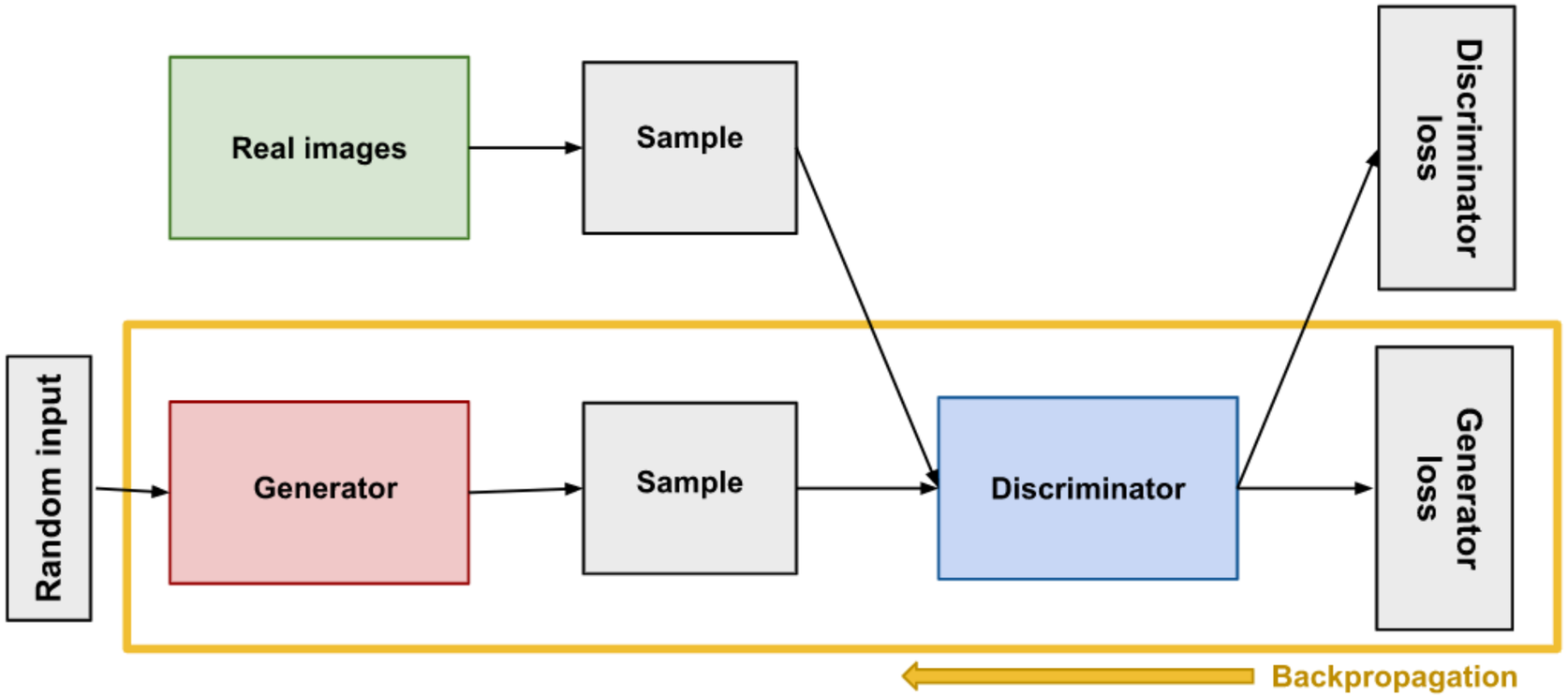
The **discriminator**, opposed to the generator, follows a specular structure: here **de-convolutions** are used to **reduce** the spatial dimensions at each layer, taking as input an image, real or generated, and progressively reshape the spatial dimension until reaching a 4x4 feature map, and finally **outputting a probability** which tells whether the image is real or not.

Structure and parameters :

- **layers: 5**
- **kernel size: 4**
- **stride: 2**
- **padding: 1** (except last layer, that is 0)



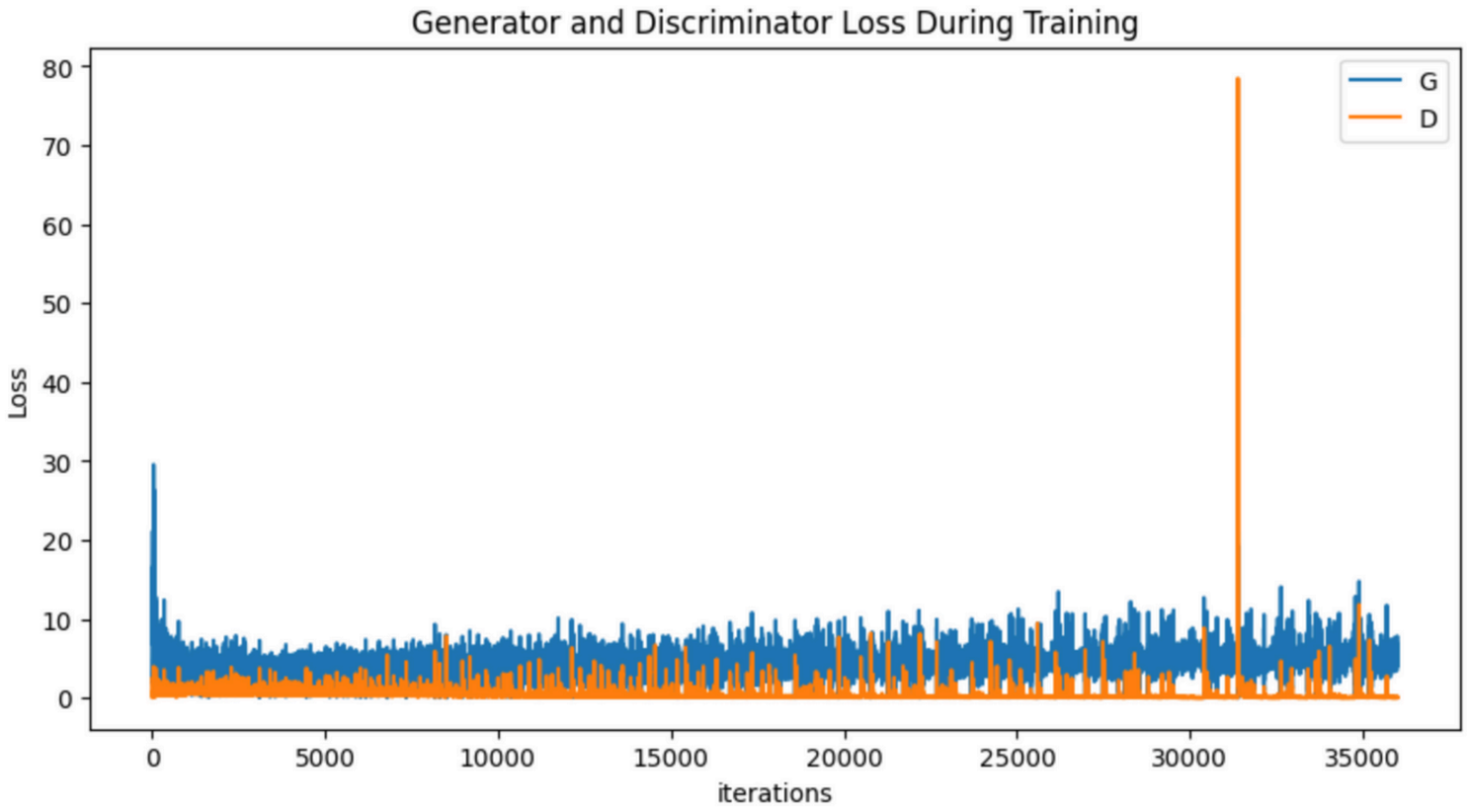
3.4 Training



Parameters:

- **batch size: 125**
- **epochs: 150**
- **learning rate: 0.0002**
- **beta1 for Adam optimizers: 0.5**

3.5 G-Loss and D-loss

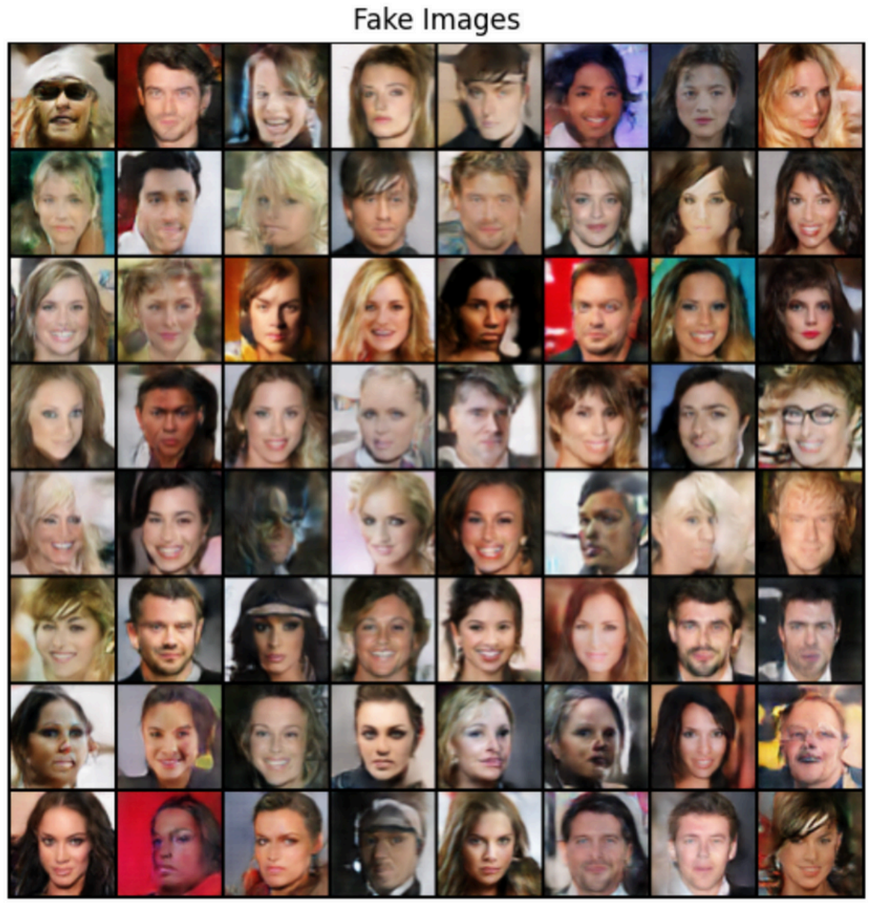


Both **Generator** and **Discriminator losses** seems to better over the iterations - i.e. the difference between the output and the real labels for the former, and the difference between the output and the fake labels for the latter are decreasing - altough the discriminator registered a huge loss around the 32000th iteration: considered the fact that the resulting images are acceptable, the model was not changed.

3.6 Final results

Some observations on the results (random batch from last epoch):

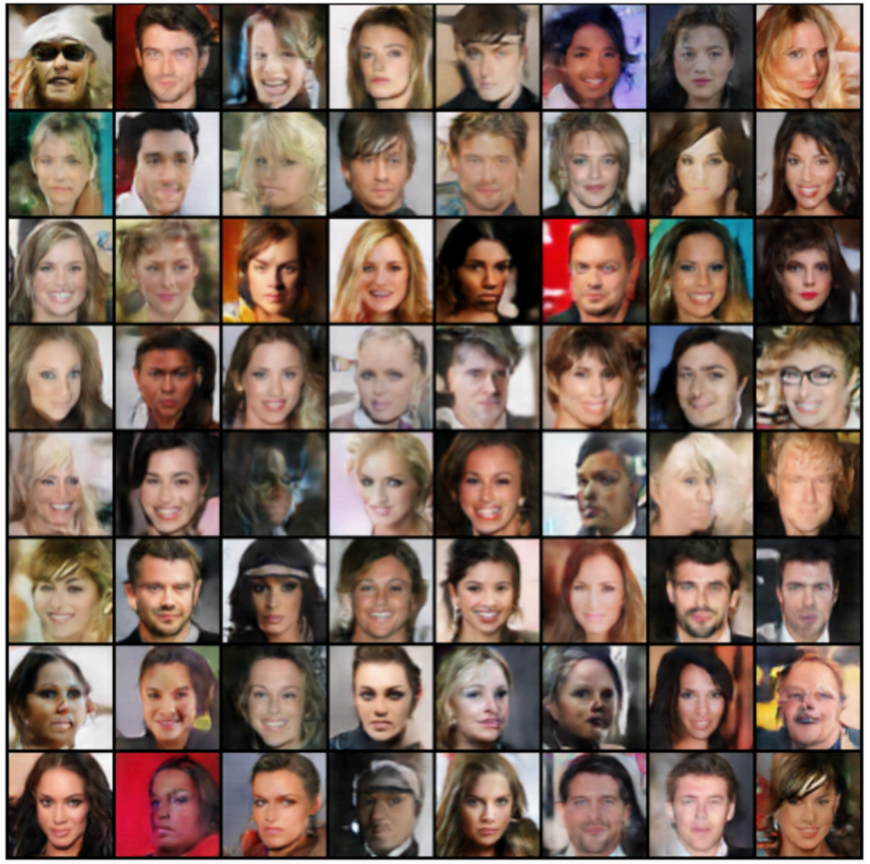
- The general output is acceptable, meaning that most faces are interpretable as human-like.
- The difference between male and female faces is, in most cases, clear.
- Male faces seems to be more plausible than female ones (maybe some anatomic elements makes the generation process easier, i.e. shorter or less elaborated hairstyles, as well as the absence of makeup, or perhaps there are more male faces than female ones in the dataset).



3.6 Final results

Real Images

Fake Images



Thanks for your attention