DIGITAL SIGNAL AND IMAGE MANAGEMENT

Leanza Alfio

Franchina Francesco

INTRODUCTION

The main object of this project is to solve three tasks related to digital signal and image management. The three tasks are done on different dataset and using different models and frameworks.

1. In particular the first task is about analyzing a set of signal audio and making binary classification on different audio.
2. The second task is made on a set of images and the purpose is to find a model that classificate the images.
3. The third is about applying the GAN to generate fake images starting from a set of real ones.

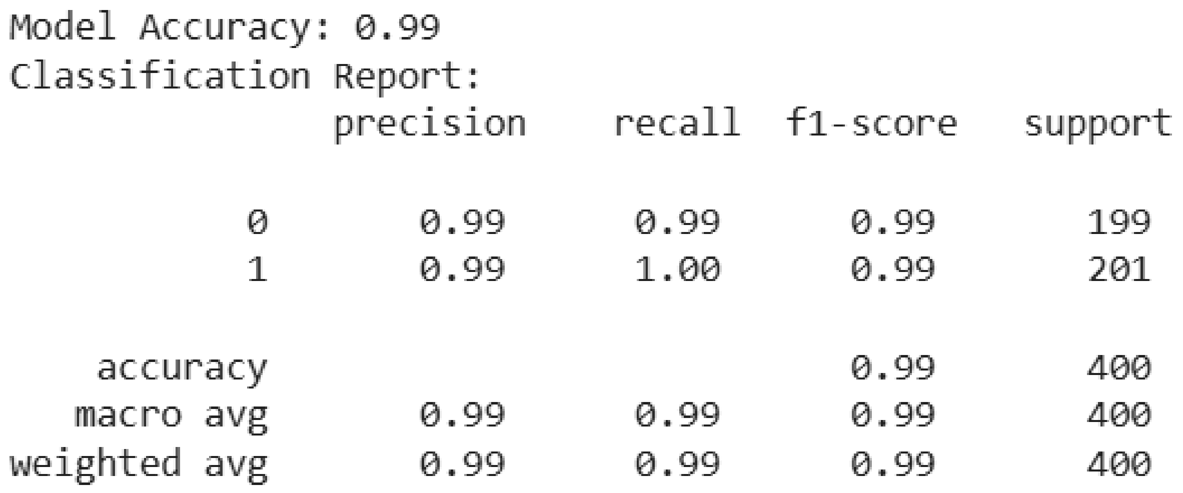
AUDIO GENDER CLASSIFICATION

For the first task we choose a dataset from Kaggle, containing about 15000 signal audio that represent the voice recording of different people both male and female. Our main goal is to define a model that correctly predicts on unseen data if the voice is of a man or of a woman.

- The first step is to prepare the dataset for using: we extract the MFCC features and we assigned the labels to the classes, 0 for male and 1 for female.
- Then we took the first 1000 signal audios for each gender and we splitted in training data (80%) and test data (20%)
- we initialized the Support Vector Machine and we trained the model.

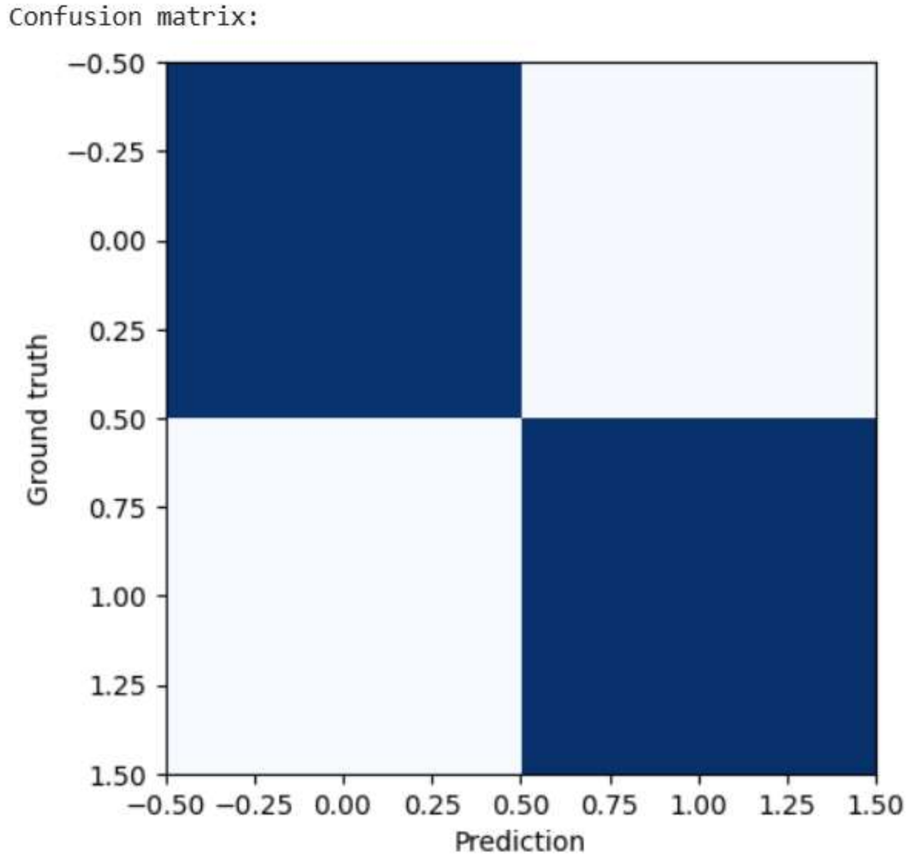
AUDIO GENDER CLASSIFICATION

After we trained the model we tested it and print some metrics to evaluate it, and this are the results:



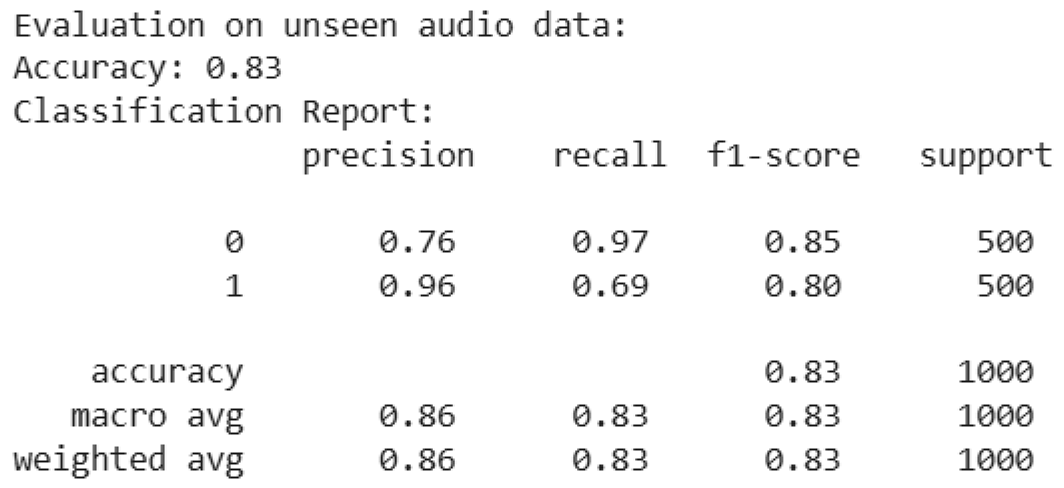
```
Model Accuracy: 0.99
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       199
           1       0.99      1.00      0.99       201

    accuracy                           0.99       400
   macro avg       0.99      0.99      0.99       400
weighted avg       0.99      0.99      0.99       400
```

This is the graphical representation of our results:



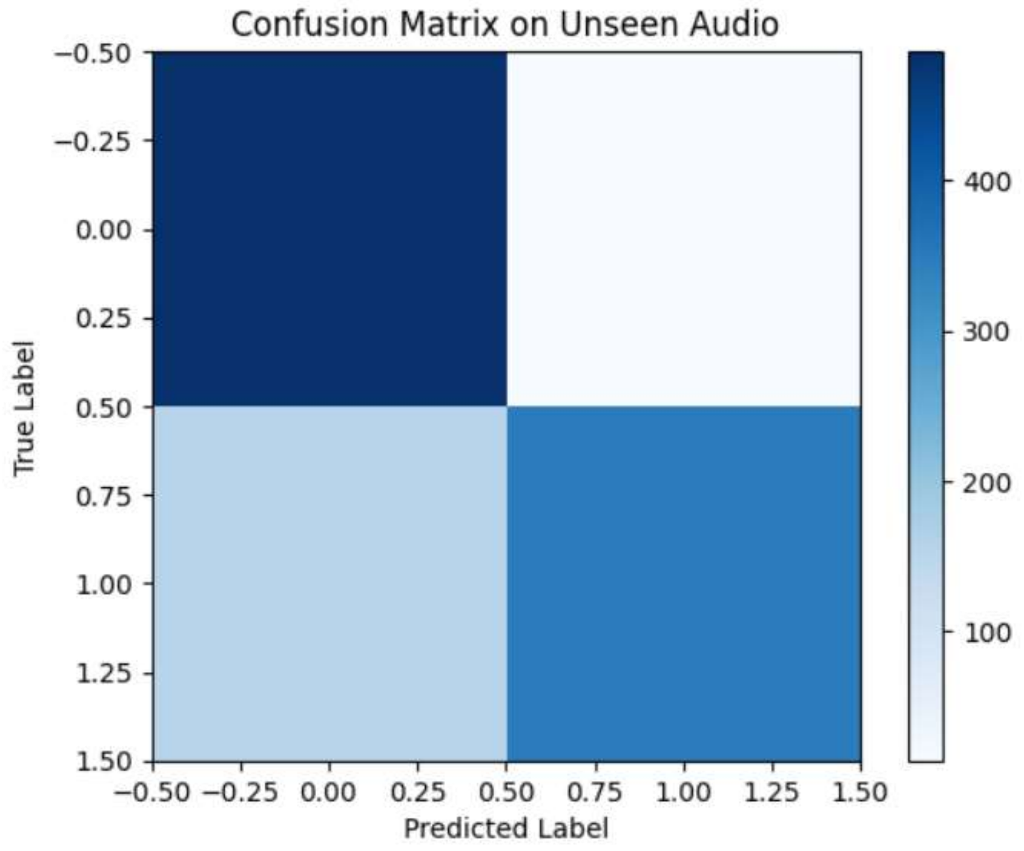
AUDIO GENDER CLASSIFICATION - validation

As we said before we did not use all the signal audio but only 1000 for gender. In order to have a more complete view we used the 500 audios for each gender as validation set to apply our model on new unseen data and see how it performs. We repeated the process and then we run our model and this are the results:



```
Evaluation on unseen audio data:
Accuracy: 0.83
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.97      0.85       500
           1       0.96      0.69      0.80       500

    accuracy                           0.83      1000
   macro avg       0.86      0.83      0.83      1000
weighted avg       0.86      0.83      0.83      1000
```

Finally this is the confusion matrix on the unseen data:

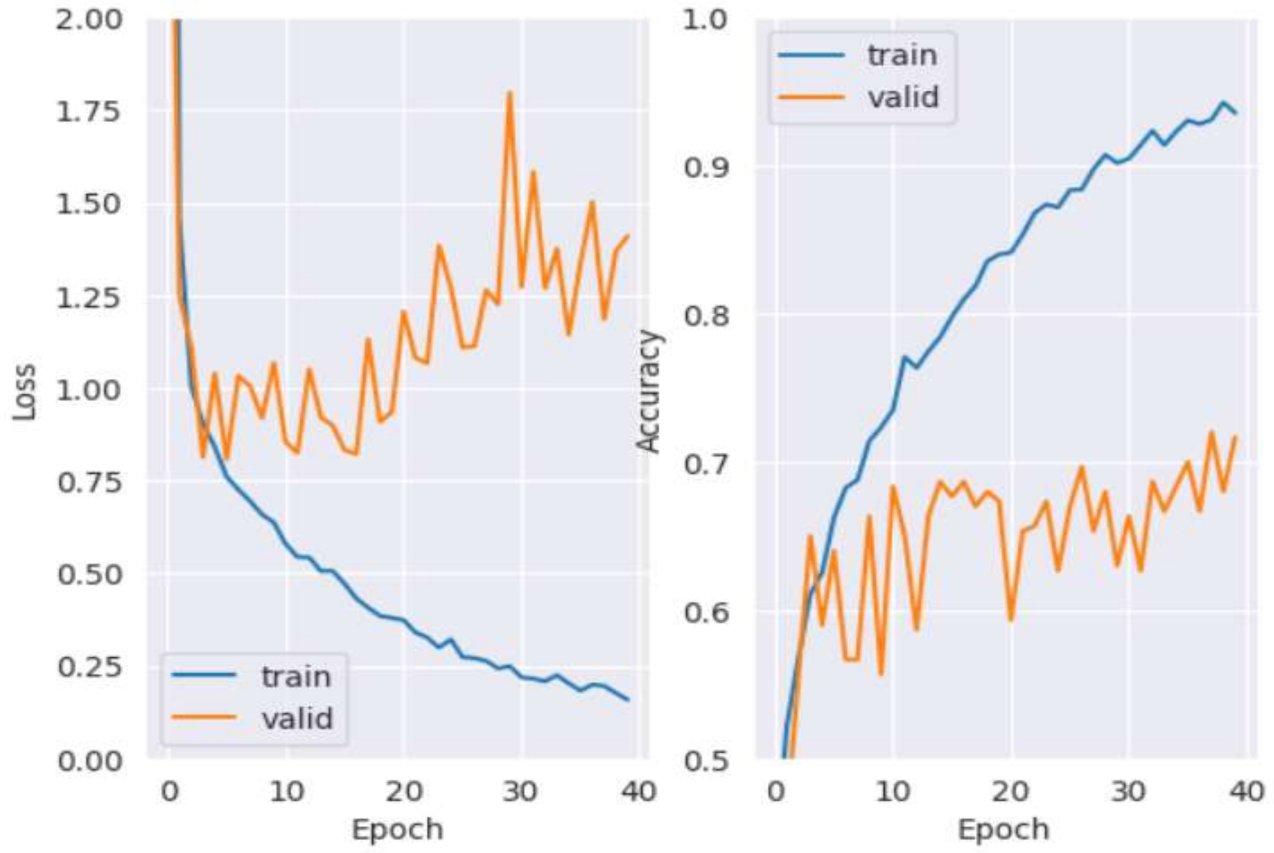


CNN - IMAGE CLASSIFICATION

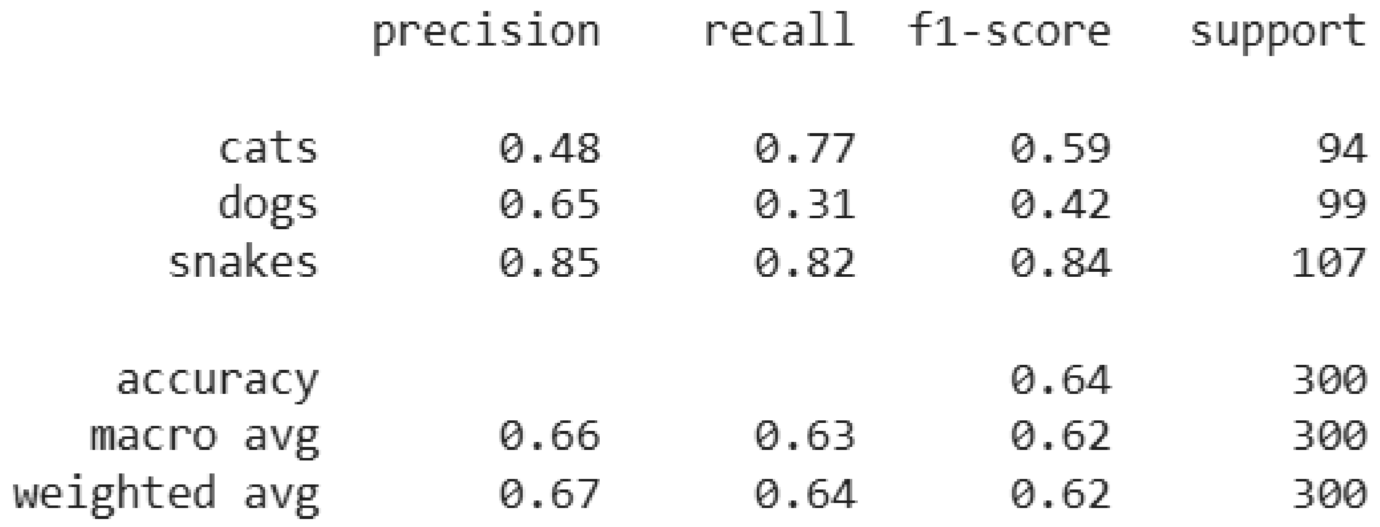
For the second task we used a dataset taken from Kaggle, composed by 3000 images divided into 3 classes: dogs, cats and snakes. The goal of this task was to define a model that correctly classifies new images into the right class.

- we firstly divide our dataset into training (2400), test (300) and validation (300).
- we extract the features from the images, we reduce the spatial dimension and as outputs we obtained the probability for each class.
- we specified the loss function and the optimizer for our model and we chase the accuracy to check the performance of our model
- we then started the training of our neural network with 40 epochs.

This is what we obtained:

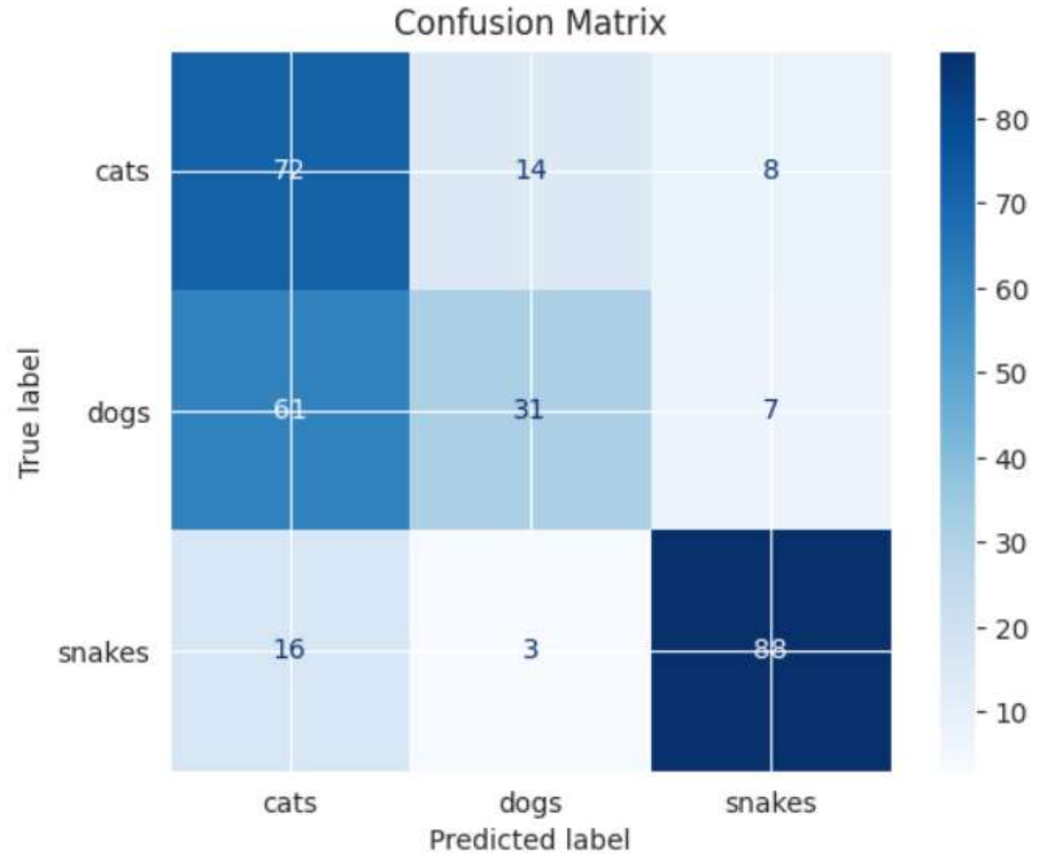


We generate our predictions and run our model on the test set and these are the results:

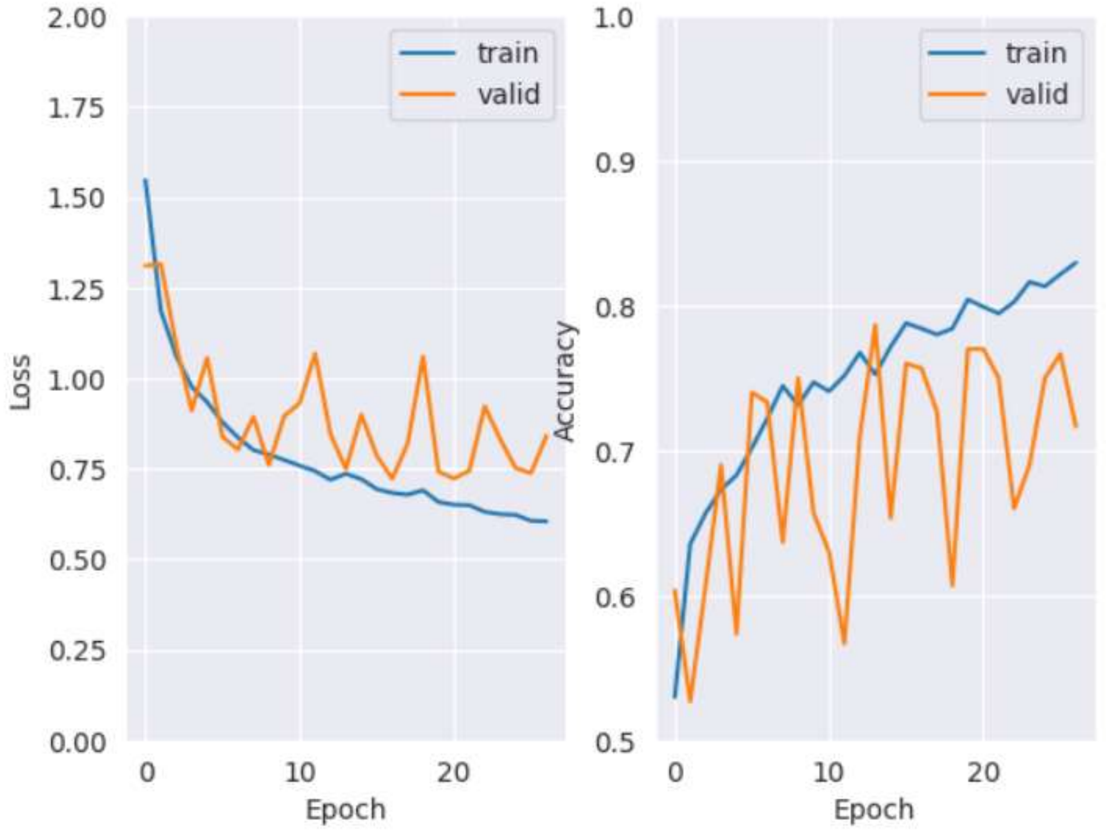


```
              precision    recall  f1-score   support

        cats       0.48      0.77      0.59        94
        dogs       0.65      0.31      0.42        99
      snakes       0.85      0.82      0.84       107

    accuracy                           0.64       300
   macro avg       0.66      0.63      0.62       300
weighted avg       0.67      0.64      0.62       300
```

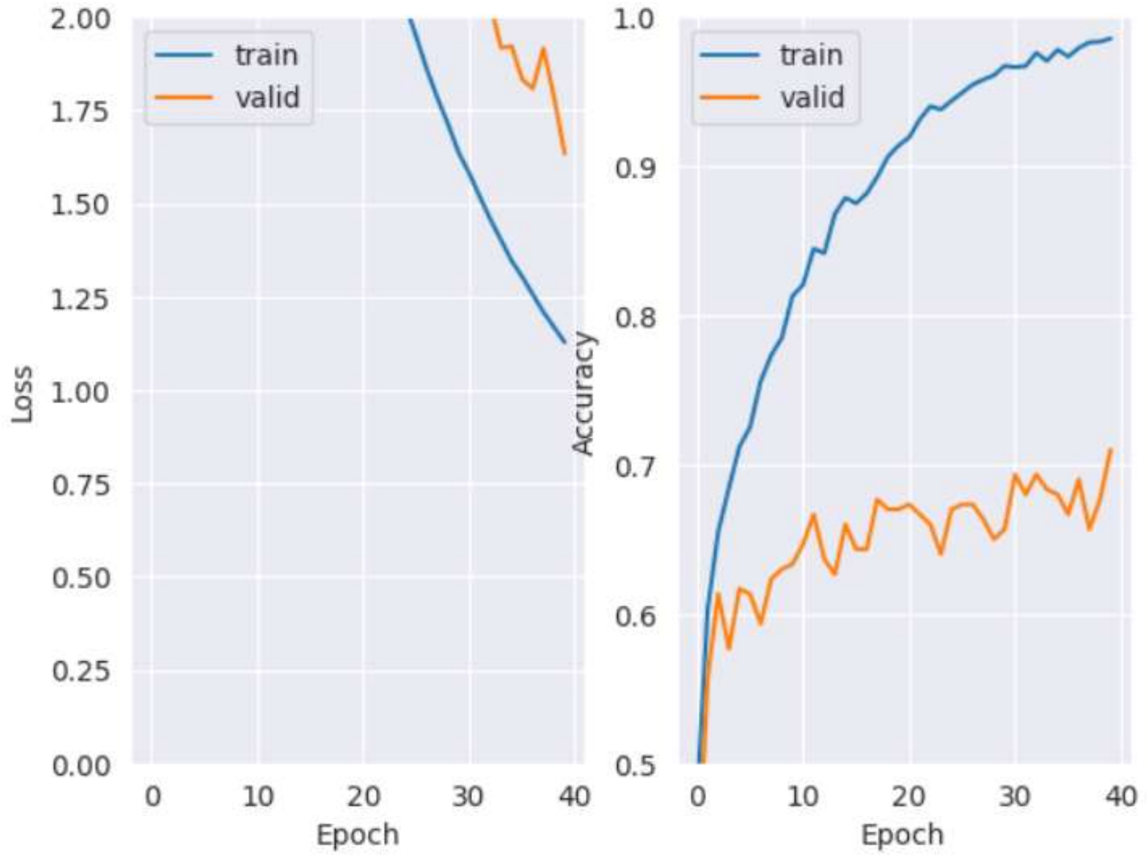
And this is the confusion matrix:



We tried other solutions in order to find the best model. For example in this case we add a callback function to check the trend of the loss function and avoid the model to overfit and stop earlier.



We gave another try by constructing a deeper model in order to try to gain the maximum that our observations can give, but the results were not as we expected:



CNN - data augmentation

- we want to create a CNN to classificate images
- we want to add Data Augmentation to improve generalization
- we want to use regularization L2 and Batch Normalization in order to stabilize the training

CNN - model structure

- the model accept in input RGB images
- we apply data augmentation e.g. horizontal flip, contrast variation...
- we add convolutional blocks: 4 Conv2D levels with progressive filters, batch normalization and max pooling to reduce dimensions.

The idea is to augment the data variety used to train the model and reduce the overfitting, improving the generalization.

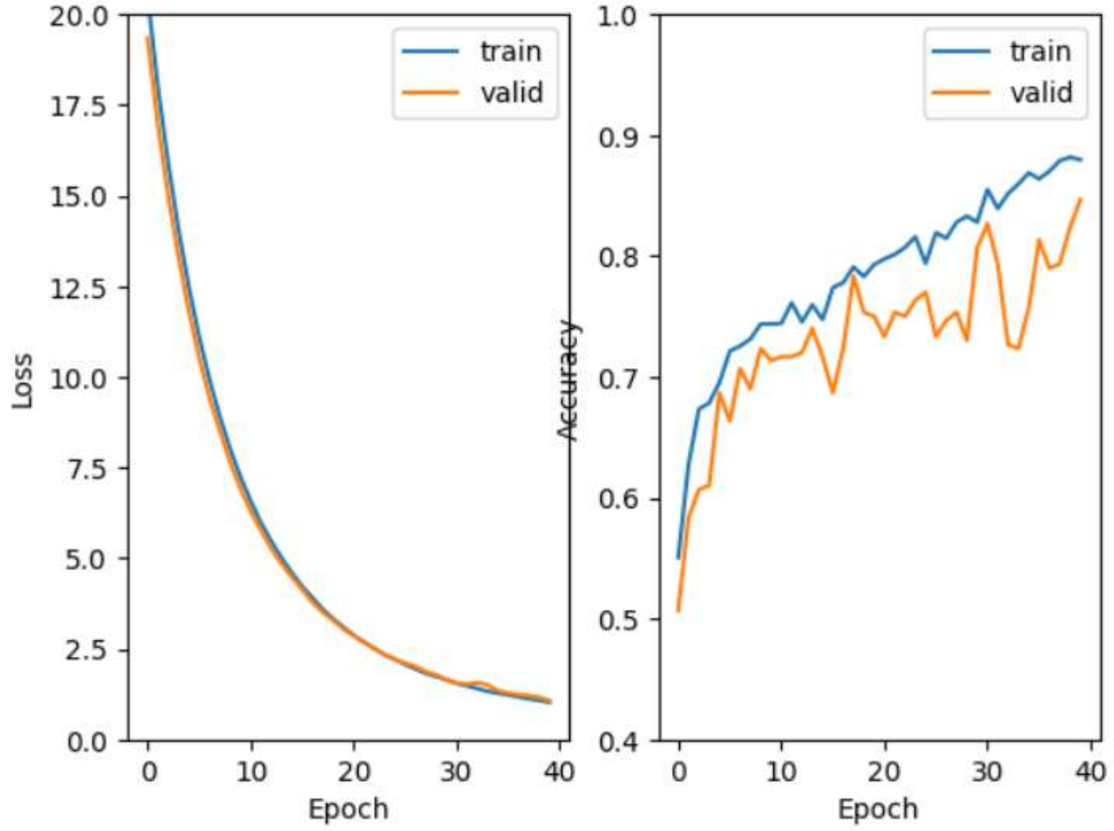
CNN - advantages

The main advantages on apply data augmentation on our model can be resumed in 3 points:

1. Robustness - since data augmentation improves generalization
2. Stability - since batch normalization and SELU avoid training problems
3. Simplicity - the CNN is solid and efficient

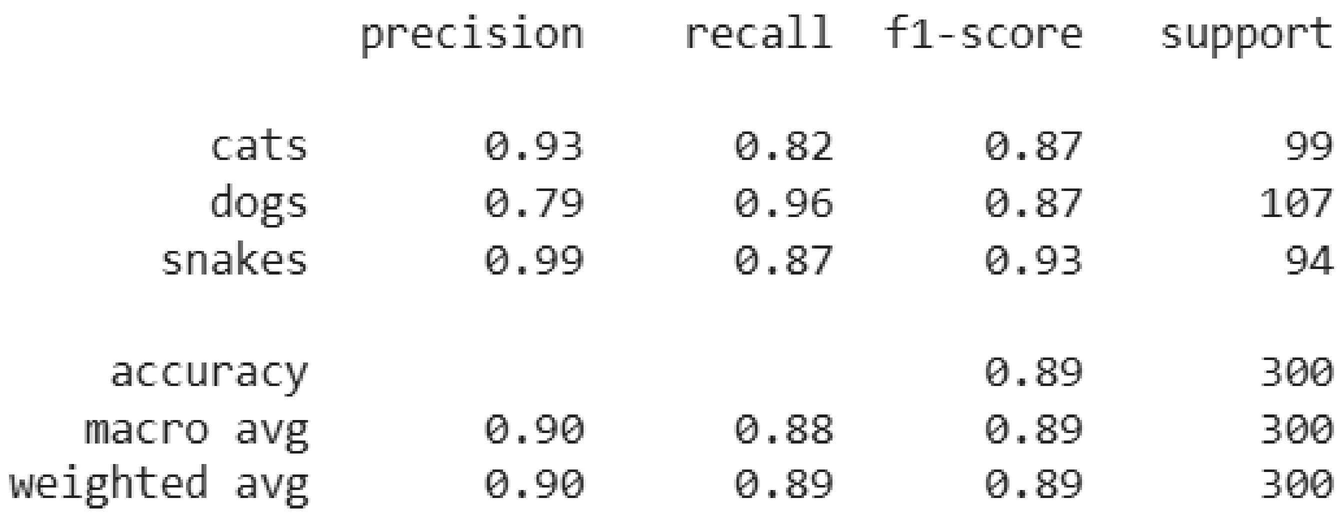
CNN - results

After that, we started training the model and these are the results:



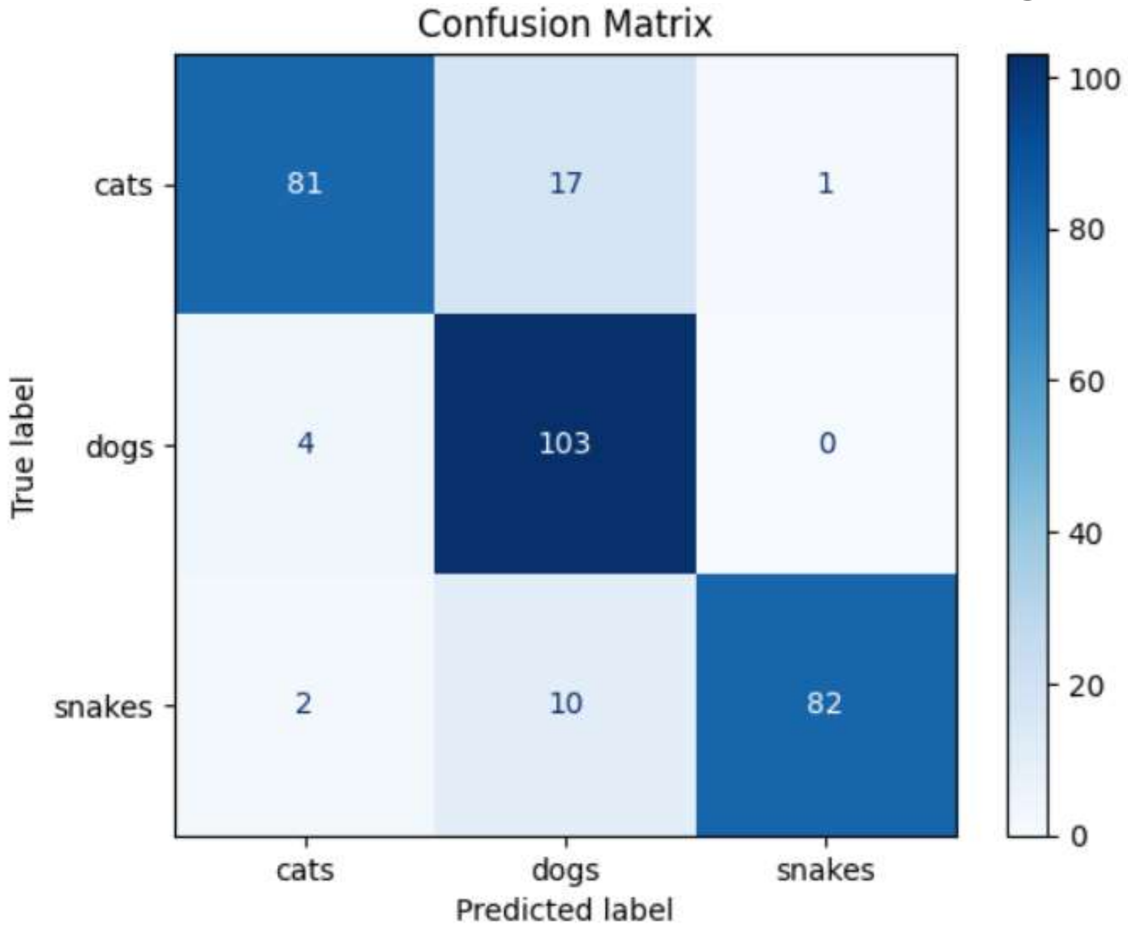
CNN - metrics results

These are our model performance in terms of metrics:



```
              precision    recall  f1-score   support

        cats       0.93      0.82      0.87        99
        dogs       0.79      0.96      0.87       107
      snakes       0.99      0.87      0.93        94

    accuracy                           0.89       300
   macro avg       0.90      0.88      0.89       300
weighted avg       0.90      0.89      0.89       300
```

Finally, this is the confusion matrix of our model with data augmentation:



GAN - FAKE IMAGES GENERATION

For the last task we built a GAN (Generative Adversarial Network), in order to create fake images starting from a dataset taken from Kaggle, composed by 63565 anime face images.

- we defined a generator and a discriminator
- we defined a custom training loop
- we defined a visualization callback (every 5 epochs we generate a grid of 5x10 images)

GAN - generator

Input: Noise vector (latent dimension = 100)

Layers:

- dense layer to project the noise into a 4x4x256 volume
- reshape to prepare for deconvolution
- multiple Conv2DTranspose layers (with Batch Normalization and ReLU activation) to upsample
- final Conv2DTranspose with Sigmoid activation to output a 64x64x3 image

Key Point: The generator learns to create images from random noise

GAN - discriminator

Input: Image of shape 64x64x3

Layers:

- several Conv2D layers with LeakyReLU activation
- Batch Normalization and Dropout for regularization
- flattening layer to convert features into a vector
- dense layer with Sigmoid activation to output a probability (real/fake)

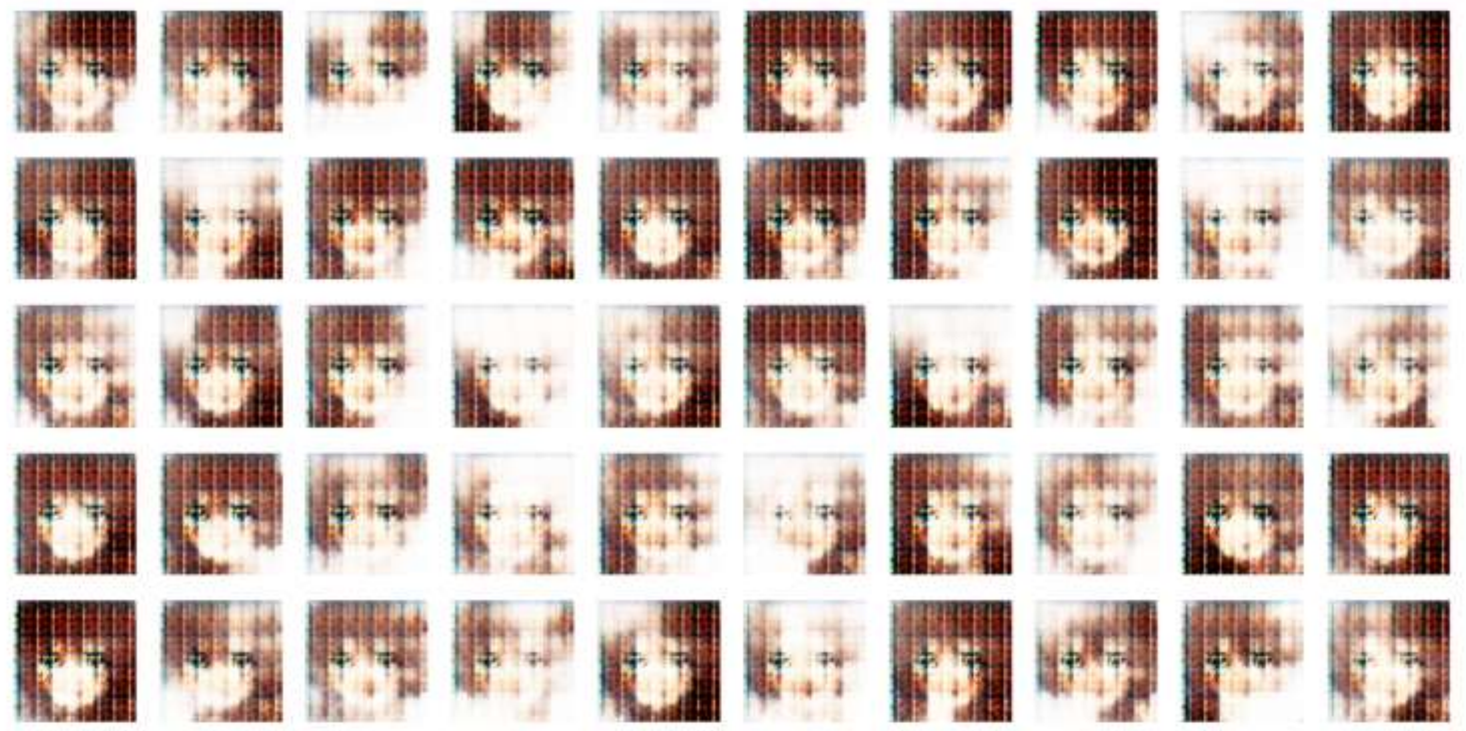
Key Point: The discriminator is trained to distinguish between real and generated images

GAN - output and results

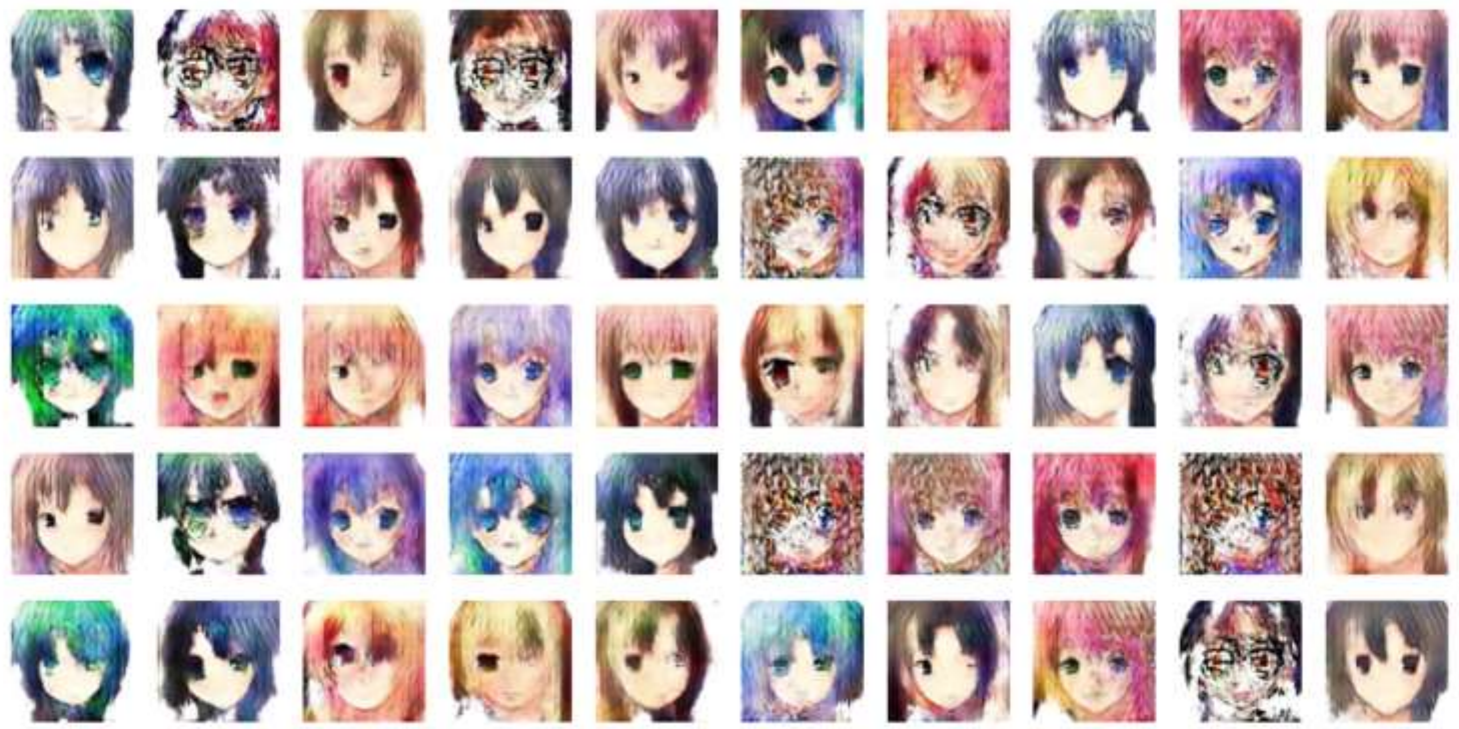
After Training:

- generate a final grid of images from random noise
- visualize the quality and diversity of the generated images

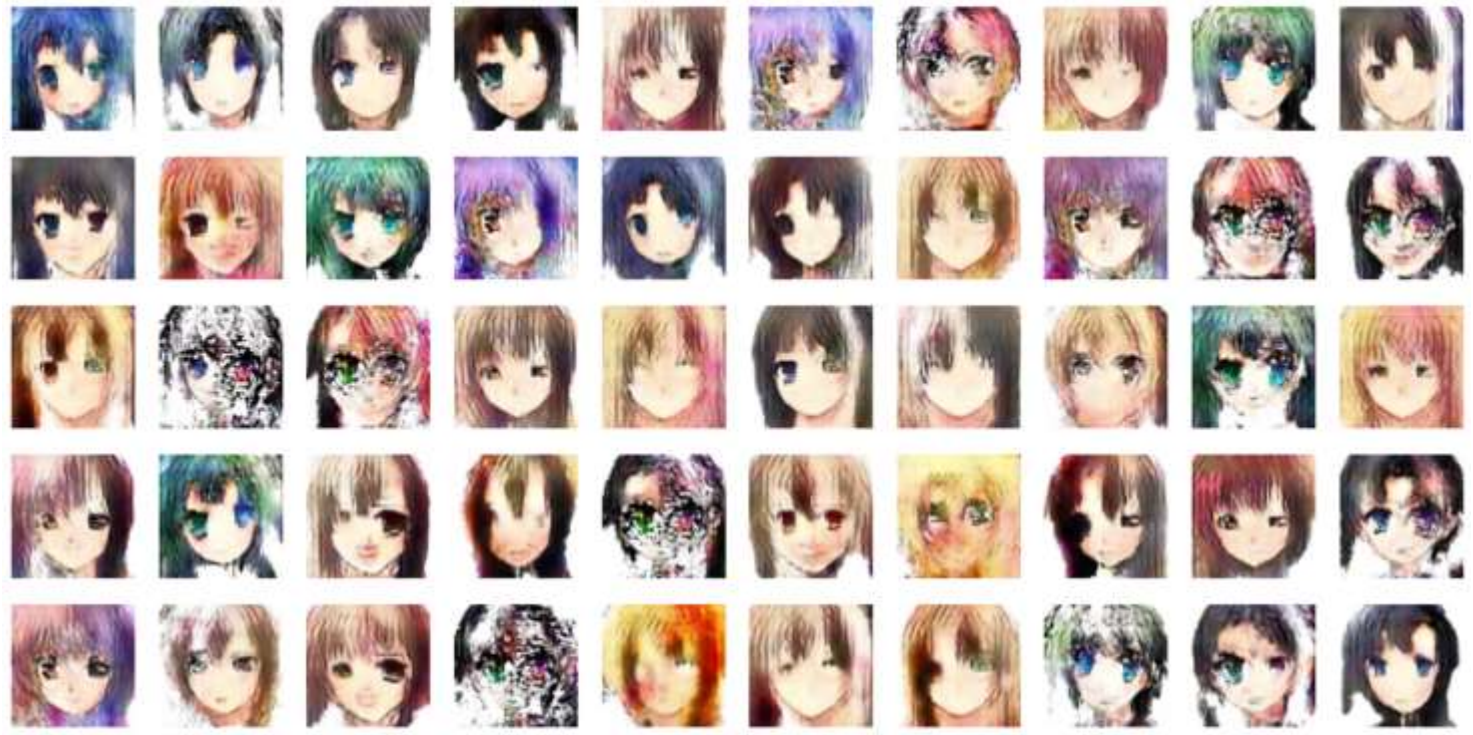
GAN - epoch 0



GAN - epoch 25



GAN - epoch 50



THANKS FOR YOUR ATTENTION!