Normalizing Flows to detect OOD Data

Normalizing Flows

- Class of deep generative models composed of a series of simple invertible functions.
- They model a target distribution:

$$p_X(x) = p_Z\left(f^{-1}(x)\right)\left|det\frac{\partial f^{-1}}{\partial x}\right|, \qquad p(D) = \prod_{x\in D} p_X(x);$$

- Training uses the maximum log-likelihood principle of training data with respect to the parameters of the function f

Affine Coupling Layers

- Each layer applies a transformation:

$$f_{aff}^{-1}\left(x_{id}, x_{change}\right) = \left(y_{id}, y_{change}\right), \begin{cases} y_{id} = x_{id} \\ y_{change} = \left(x_{change} + t(x_{id})\right) \odot \exp(s(x_{id})) \end{cases}$$

- x_{id}, x_{change} and y_{id}, y_{change} are, respectively, disjoint parts of input and output
- $s(), t()$ are scale and shift operators, usually implemented by a Neural Network
- Efficient computation of the Jacobian: $\log\left|det\frac{\partial f^{-1}}{\partial x}\right| = \sum_{i=1}^{\dim(x_{change})} s(x_{id})_i$

Advantages of Normalizing Flows (Over GANs and VAEs)

- No need to put noise on the output → more powerful local variance models
- More stable training process, fewer problems in tuning hyperparameters
- Much easier to converge

Disadvantages of Normalizing Flows

- Not as expressive as other approaches
- Volume preservation over transformation (to be injective) leads to a very high dimensional latent space, harder to interpret
- Generated samples are not as good as the other approaches

OOD Data Detection with Normalizing Flows

- Due to exact likelihood as output, they can be used to find Out of Distribution Data (OOD), by assigning a threshold on the outputs.
- Problems arises due to the inductive bias of the flow. It determines which specific solution is found by the training and how high is the likelihood to be assigned.
- Inductive bias needs must be aligned with the semantic structure of data while Normalizing Flows tends to learn local graphical features.
- This is due also to the Coupling Layers Co-Adaptation, due to the parameter learning which uses local pixel correlation.

Solutions for OOD data detection

- Reduction of the st-network capacity adding a bottleneck to decrease the latent space dimension
- Training on high-level semantic representations instead of raw data, introducing an embedding as pre-processing.

Case Study: FaceForensic Dataset

5000 videos:

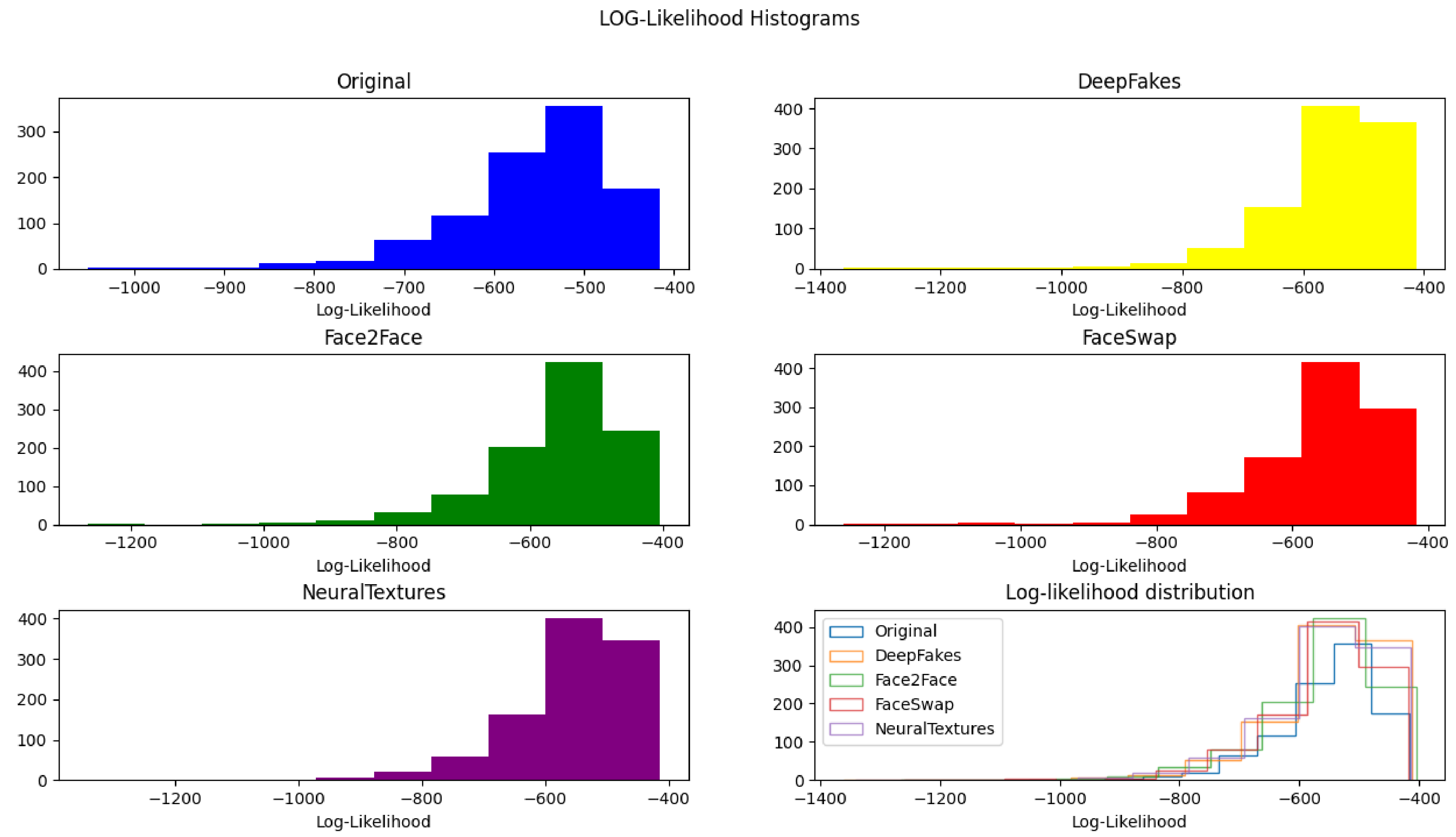
- 1000 original YouTube videos of people mostly speaking frontally without occlusion
- 4x1000 manipulated videos applying 4 different Face Swapping techniques:
  - ***FaceSwap***: Graphic based approach to transfer the facial region from a source to a destination video, using sparse facial reference points
  - ***DeepFake***: Replacement of a target face by an observed one. It relies on two autoencoders with a shared encoder trained to reconstruct training images
  - ***Face2Face***: Transfer of the expressions of a source video to a target one while maintaining the identity of the target person. Keyframes are used to generate a dense facial reconstruction which can be used to re-synthesize the face under different illumination and expressions
  - ***NeuralTextures***: Use of original data to learn the neural texture of the target person based on geometry. It modifies only the mouth region.

Objective

Training of the Normalizing Flow on the original videos and then verification of the likelihood of each method of Fake Swapping. The main point is to determine how well these methods create fake faces.

Process

- Application of MediaPipe for face detection to crop all 5000 videos
- Data Embedding via a 3D ConvNet, pretrained for action recognition, to obtain 400-dimension data, for all 5000 videos.
- Use of RealNVP model that implements Normalizing Flow to learn the distribution of the original data.
  - Adaptation of the model to the new data dimensionality
  - Hyperparameters tuning
- Application of the trained model to manipulated embedded videos.


LOG-Likelihood Histograms
Original
DeepFakes
Face2Face
FaceSwap
NeuralTextures
Log-likelihood distribution
Log-Likelihood
Original
DeepFakes
Face2Face
FaceSwap
NeuralTextures

Results

Inability to find a threshold between different distributions due to:

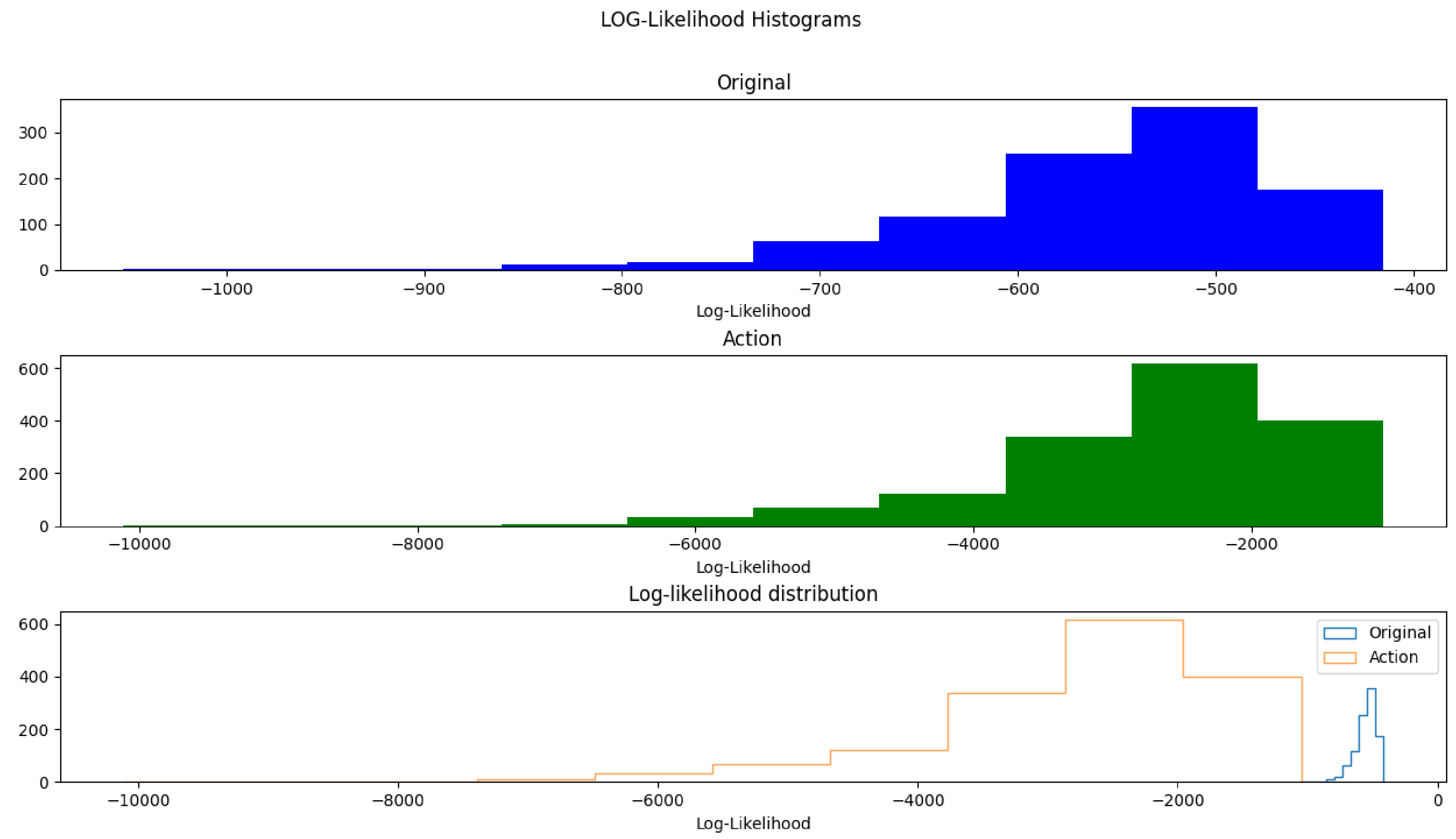
- Goodness of the Swap methods [unlikely]
- Incapacity of the model to grasp the true features, perhaps also conditioned by the compression rate of the videos [40]
- Wrong characterization during the embedding, because an action prediction Convnet is not specialized in faces textures.
- Because we already have a cropped face, low-level features are needed to distinguish different details, so a high-semantic representation of data could be counterintuitive to the objective.

Possible improvements

- Use a better video resolution
- Change the embedding architecture, more appropriate with respect to the specific task
- Try with original data instead of embedding ones to catch low-level features.

Case Study: UCF11 Dataset

- YouTube videos with 11 categories (like basket shooting or cycling etc.):
  - 25 subgroups for all similar poses, apparency etc.
    - At least 4 videos for each folder.
- To proof the effectiveness of the theoretical basis of the model.
- Clear distinction between the two distributions and a higher likelihood to in-distribution data


LOG-Likelihood Histograms
Original
300
200
100
0
−1000
−900
−800
−700
−600
−500
−400
Log-Likelihood
Action
600
400
200
0
−10000
−8000
−6000
−4000
−2000
Log-Likelihood
Log-likelihood distribution
600
400
200
0
−10000
−8000
−6000
−4000
−2000
0
Log-Likelihood
Original
Action