Deepfake Detection - Project Report

Author: El Abassi Tamer Alexandru

---

1. Introduction

In this report, I will present all the models I tried for detecting deepfake images. Specifically, a K-Nearest Neighbors (KNN) model and a Convolutional Neural Network (CNN) model. I will also describe how I processed the data and tuned the hyperparameters, including failed attempts. It's worth noting that I initially built a simple KNN model with low accuracy (~65%), after which I implemented a CNN model.

1.1 Dataset

→ training set: 12.500 images (.png, size 100 x 100)

→ validation set: 1.250 images (.png, size 100 x 100)

→ test set: 6.500 images (.png, size 100 x 100)

---

2. Modelul CNN - modelul ales

2.1 Procesarea datelor & Augmentarea datelor

→ Resize:

- **120 x 120 → first version**, performed well since the original images were 100 x 100
- **160 x 160 → final version**, slightly better than 120 x 120
- **224 x 224** → performed significantly worse, unless a 5th CNN layer was added

→ Pixel Normalization:

- normalized pixel values to the [0, 1] range to maintain small, consistent input scales

→ Data Augmentation:

Designed to diversify the image data and prevent overfitting:

- **horizontal flip** (on 50% of images), to avoid overfitting on orientation
- **brightness** variation (± 10%) → to simulate lighting changes
- **contrast variation** (0.9% - 1.1%)
- **rotation** (± 15%)
- **zoom** (0.9% - 1.1%)
- **translation** (± 10%) → to make the model invariant to object position

---

2.2 CNN Arhitecture

→ Input

- RGB Image tensor (H, W, 3)

→ 4 Convolational Blocks

- **Conv2D**: kernel 3x3, padding = “same”, filters = [32, 64, 128, 256], activation = ReLU
- **BatchNormalization**
- **MaxPooling2D**: downsampling 2x2
- **Dropout** (0.25): randomly disables 25% of feature maps to prevent overfitting
- **Final layers: GlobalAveragePooling, Dense(512) → BatchNorm → Dropout, Dense (256) → BatchNorm → Dropout, Dense (5) with Softmax** (for 5-class classification)

→ Dense Block 1:

- **Dense(512**), activation = ReLU
- **BatchNormalization**
- **Dropout** (0.6) → high dropout, tried 0.3 but accuracy decreased

→ Dense Block 2:

- **Dense(256)**, activation = ReLU
- **BatchNormalization**
- **Dropout** (0.4) → tried 0.2, but accuracy decreased

→ Final Layer

- **Dense** (5) + **Softmax** for final classification (with L2 regularization)

2.3 Hyperparameters Tested

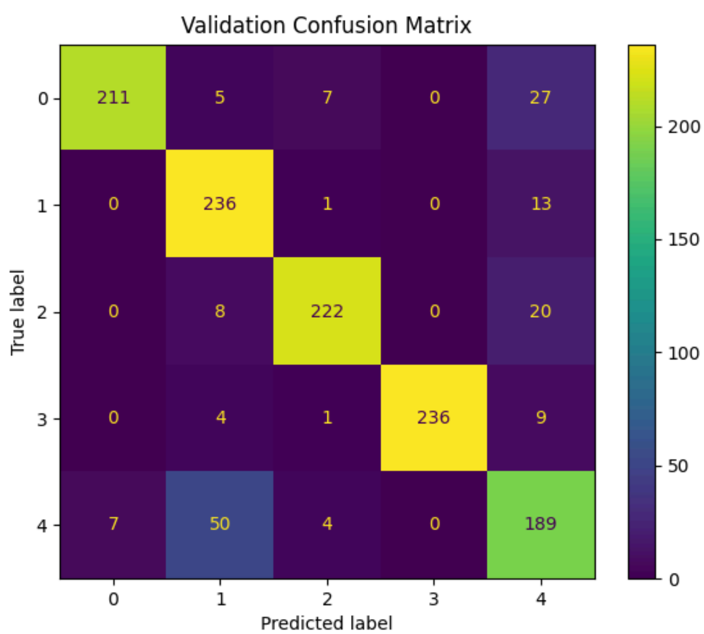
- **Resolution:** 120, 160 si 224
- **Batch Size:** 32, 64, 128
- **Learning Rate:** 1e-3, 2e-3, 5e-4
- **Optimizer:** Adam, AdamW
- **Number of Convolational Blocks:** 4, 5

Experiment ID	Resolution	Batch	LR	Optimizer	Depth	Val Acc	Concluzii
E1	120	64	1e-3	Adam	4	0.86	Solid baseline.
E2	160	32	1e-3	Adam	4	0.89	Rezolutia 1 better.
E3	224	64	1e-3	Adam	4	0.84	Resolution 224 is too large, originals a 100 x 100.
E4	224	64	1e-3	Adam	5	0.85	Not even or more convolatior block is enough to justify 224 resolution.

E5	160	128	1e-3	Adam	5	0.87	Batch 128 likely too large.
E6	160	64	2e-3	Adam	4	0.90	Changing learning ra leads to slight improvement
E7	160	64	5e-4	Adam	4	0.89	Not optimal LR.
E8	160	64	1e-3	AdamW	4	0.88	AdamW doesr help.
E9	**160**	**64**	**1e-3**	**Adam**	**4**	**0.91**	**Best combinatior**
E10	224	64	1e-3	AdamW	5	0.88	Not better than E9.

Final chosen model: E9.

2.4 Confusion Matrix



3. KNN Model - First Model Approach

3.1 Data Prepocessing

→ Bins: (8,8,8)

- vector of 512 dimensions

→ Image loading

- image loaded via tf.io.read_file + tf.image.decode_image, converted to N x 3 vector and L1-normalized

---

3.2 Structura KNN

→ BINS = (8,8,8)

- compact 3D histogram (512 dims), enough to capture color without high complexity

→ NEIGHBORS = 5

- tested with 1, 3, 5, 7, 9 → 5 was most balanced against noise, underfitting and overfitting

→ SEED = 42

- for reproducibility

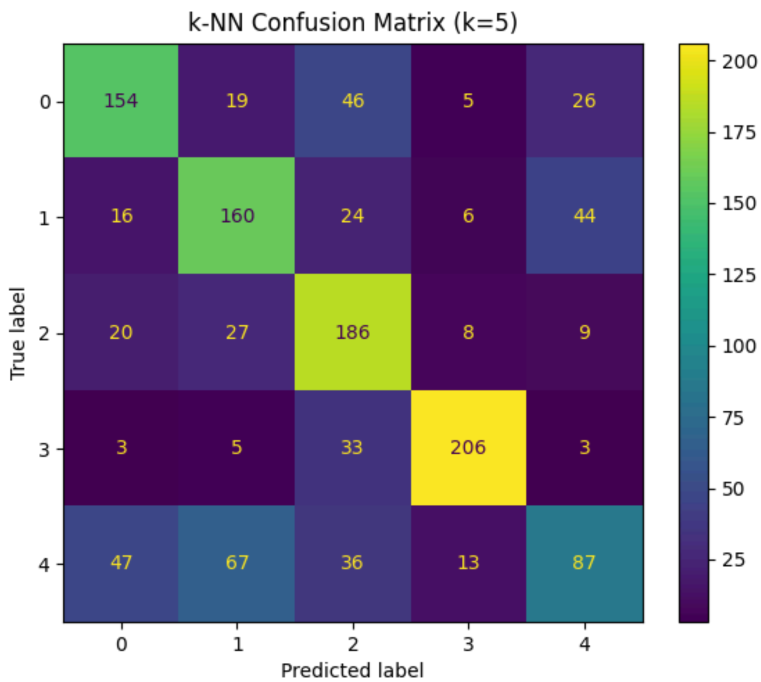
→ Metric = Euclidian

- standard for continous spaces

3.3 Hyperparameters Results

k	Validation Accuracy
1	0.62
3	0.65
5	**0.67**
7	0.65
9	0.64

3.4 Confusion Matrix


k-NN Confusion Matrix (k=5)
True label
Predicted label
0
1
2
3
4
154
19
46
5
26
16
160
24
6
44
20
27
186
8
9
3
5
33
206
3
47
67
36
13
87
200
175
150
125
100
75
50
25