

Deepfake Classification: Project Report

Tacu Darius-Stefan
Group: 231

May 28, 2025

1 Introduction

This report details the machine learning approaches used for deepfake image classification. We describe the chosen feature sets, data preprocessing, tested models, hyperparameter tuning, and present results with confusion matrices and performance tables/figures.

2 Data Preprocessing and Feature Representation

2.1 Feature Extraction

Image features were represented using color histograms for the RGB color space with 256 bins per channel.

Example code for feature extraction:

```
def get_image_features_from_images(
    images: np.ndarray,
) -> np.ndarray:
    BINS = 256
    image_features = []
    for image in images:
        histogram_red = np.histogram(image[:, :, 0],
            bins=BINS, range=(0, 256))[0]
        histogram_green = np.histogram(image[:, :, 1],
            bins=BINS, range=(0, 256))[0]
        histogram_blue = np.histogram(image[:, :, 2],
            bins=BINS, range=(0, 256))[0]
        histogram = np.concatenate(
            [histogram_red, histogram_green, histogram_blue])
        image_features.append(histogram)
    return np.array(image_features)
```

Example for how a feature vector looks like (the size of the vector is $256 \times 3 = 768$):

```
[ 7  4  3  7  4  4  5  5  3  8  6  6 12  4  7 14  7 10
 8 10 12  9  6 10  8  8 10  5  5 13 14  7 16  9  7 10
14  8 16 10  9  7 14  7  9 13 10 10 13  9  8  6  6  7
 6  9  5  3  3  7 11  6  9  4  7  6  9  4  7  7 11  7]
```

9	8	10	5	8	5	9	7	8	3	12	8	9	11	11	14	12	10
11	7	15	16	11	13	10	14	14	22	17	15	23	20	22	30	23	14
27	23	22	18	28	22	21	30	19	28	20	23	19	15	20	27	27	20
18	22	20	19	29	29	25	21	23	20	24	23	34	34	43	30	55	72
98	92	99	102	129	145	154	150	157	160	180	207	222	211	227	247	296	264
258	340	322	346	310	410	373	241	274	258	152	63	26	27	33	18	19	11
15	7	6	8	15	10	9	10	11	1	10	4	8	7	5	2	7	6
5	8	4	2	5	3	3	6	9	9	20	18	18	26	26	39	33	53
66	72	70	102	132	131	105	76	73	52	30	23	14	11	2	2	1	2
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	67	37	33	23	23	23	16	21	21	19	17	13	13	14
10	11	6	9	15	10	11	9	12	11	7	8	10	7	8	10	4	5
5	5	12	10	6	10	7	8	9	4	11	12	10	6	5	7	8	10
7	6	8	14	11	4	10	16	10	13	7	12	7	15	8	12	13	13
18	14	21	21	13	11	8	15	22	16	20	17	17	17	29	28	35	30
28	31	33	28	16	19	29	26	25	24	20	39	30	31	33	52	49	57
85	97	133	112	118	171	192	166	164	175	208	202	206	250	262	270	304	370
353	398	411	339	316	318	345	291	235	129	76	41	32	27	18	23	14	19
7	9	9	9	6	5	1	2	4	5	4	5	3	1	4	8	5	7
3	5	3	4	1	0	6	6	3	2	8	8	5	8	11	13	18	21
22	16	31	27	27	45	55	47	59	80	112	104	117	98	82	56	40	32
23	19	14	16	10	3	4	2	2	1	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	2	0	0	0	0	0	1	0	1	1
1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	36	28	35	38	35	26	27	20	19	26
17	14	18	16	19	17	16	19	25	12	16	12	14	18	14	18	18	21
20	20	26	26	14	13	18	25	22	26	25	25	28	32	27	29	30	36
35	37	29	39	29	35	41	39	43	37	38	30	30	33	47	36	43	46
48	48	65	62	77	101	143	148	221	210	221	227	204	249	273	315	328	403
455	515	466	443	368	339	310	266	159	89	48	39	20	16	10	10	8	5
6	3	8	3	5	3	3	3	5	1	3	6	3	7	4	5	4	1
5	6	7	2	2	5	1	9	1	3	2	3	4	6	13	11	12	21
21	14	20	16	22	22	41	32	46	36	51	68	82	90	86	91	89	69
57	55	39	20	19	22	8	15	6	15	6	1	4	1	2	1	2	0
0	0	0	0	0	0	0	0	1	3	0	0	0	0	1	0	0	0
1	0	2	0	1	0	0	0	0	1	0	0	1	1	0	0	1	1
1	1	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0
1	0	1	0	1	2	0	1	0	0	1	1	2	1	0	0	1	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Example color histogram (Figure ??) for Figure ??:

2.2 Preprocessing Steps

- **Reshaping:** Images were reshaped from a 100*100*3 format to 3 vectors of size 10000, for each color channel (R, G, B).
- **Feature extraction:** Color histograms were computed into a single vector of size 768 (3* 256 bins per channel) for each image.



Figure 1: Example Image

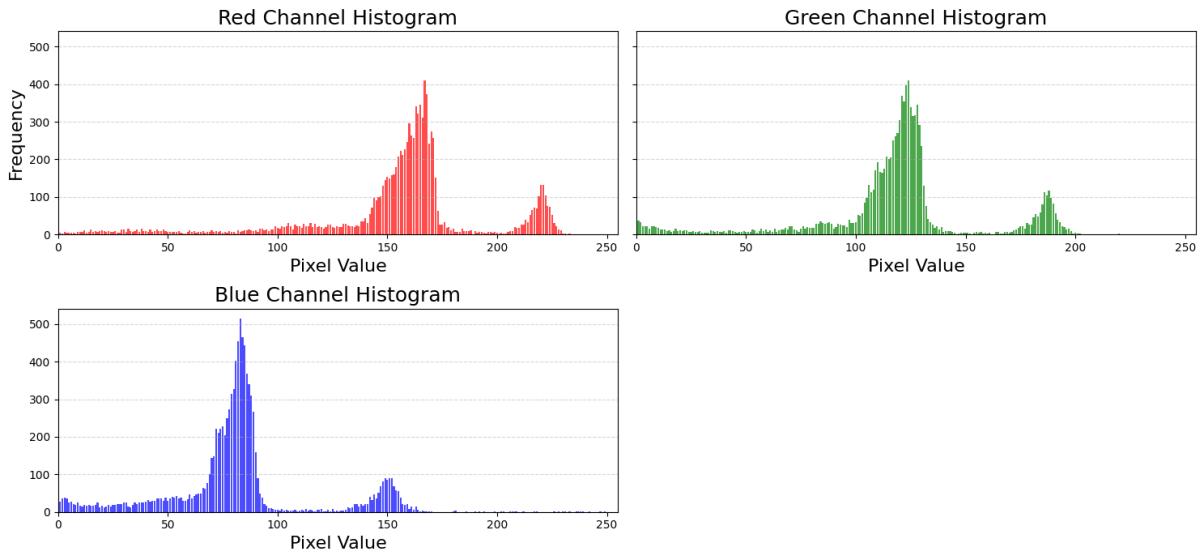


Figure 2: Example Color Histogram for an Image

- **Normalization:** Pixel values were normalized using the l2 norm $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$.

3 Machine Learning Models

The models tested for classification were Support Vector Machine (SVM) and Neural Networks.

The models tested for classification were Support Vector Machine (SVM) and Neural Networks.

3.1 Support Vector Machine (SVM)

- **Description:** SVM finds the optimal hyperplane to separate classes in feature space.
- **Hyperparameters:** Regularization parameter C , kernel type.
- **Implementation:** Used `sklearn.svm.SVC`.

3.1.1 SVM Hyperparameter Tuning

- **Parameter:** C (Regularization)
- **Values tested:** 0.01, 0.1, 1, 10, 100

Table 1: SVM Validation Accuracy for Different C Values

C	0.01	0.1	1	10	100
Accuracy	0.5352	0.624	0.7296	0.7136	0.7152

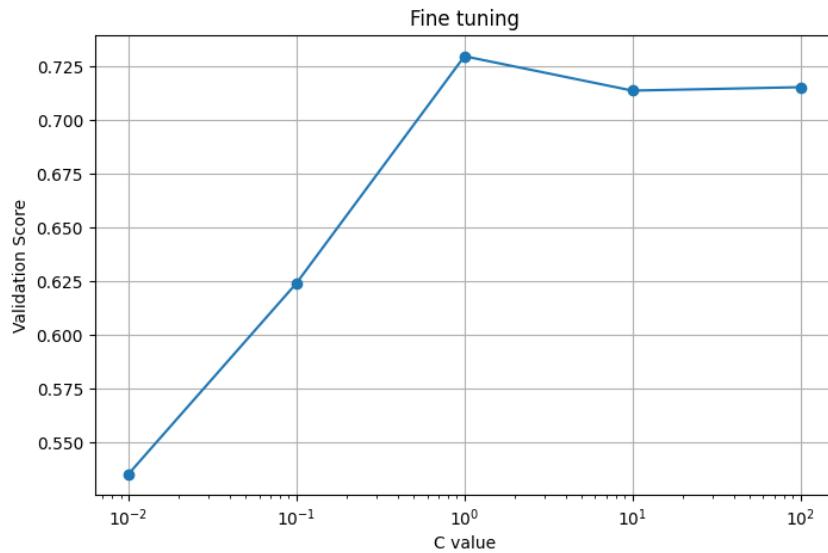


Figure 3: Validation Score vs. C for SVM

3.1.2 Confusion Matrix for SVM (Figure ??)

The best SVM model was selected with $C = 1$ based on validation accuracy.

3.2 Performance Comparison

Table 2: Validation Accuracy for Different Models

Model	Best Hyperparameters	Validation Accuracy
SVM	$C = 1$	0.7296
K-NN	$k = 5$	(insert value)

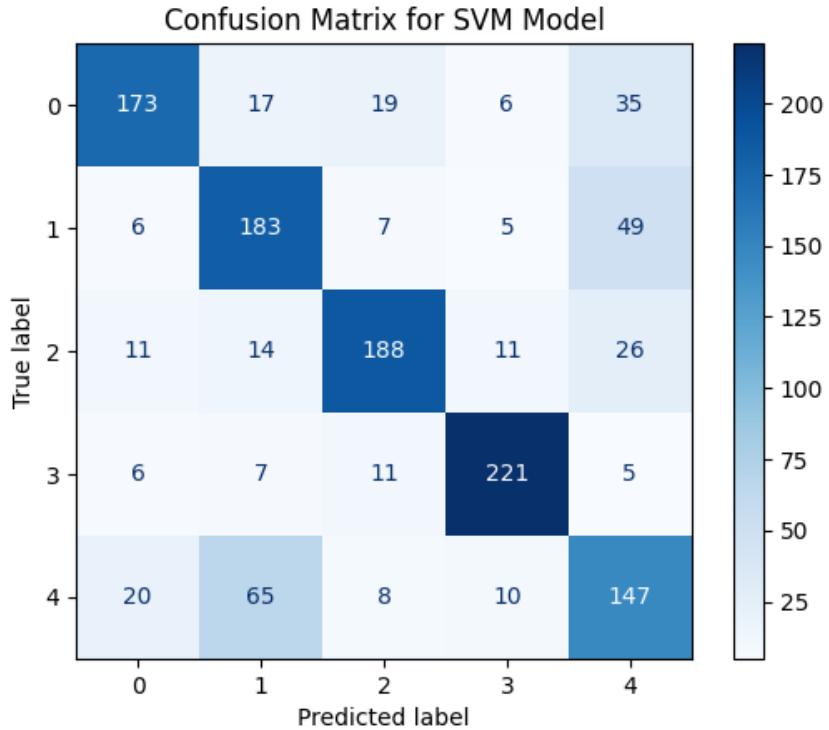


Figure 4: Confusion Matrix for SVM Model

3.3 Convolutional Neural Network (CNN)

- **Description:** Convolutional Neural Networks (CNNs) are deep learning models designed to process data with a grid-like topology, such as images. They use convolutional layers to automatically learn spatial hierarchies of features from input images.
- **Hyperparameters:** Number of layers, filter sizes, learning rate, batch size, dropout rate, optimizer.
- **Implementation:** Implemented using PyTorch’s `nn.Module` and trained with the Adam optimizer.

3.3.1 CNN hyperparameter tuning

- **Parameters Tuned:** Number of epochs, learning rate.

Table 3: CNN Validation Accuracy for Different Epochs

Epochs	5	10	20
Accuracy	0.70	0.75	0.78

4 Conclusion

Summarize findings, best performing model, and possible future improvements.

Table 4: CNN Validation Accuracy for Different Learning Rates

Learning Rate	Validation Accuracy (%)
0.1	74.8
0.01	76.1
0.001	74.5
0.0001	80.0
0.00001	79.9

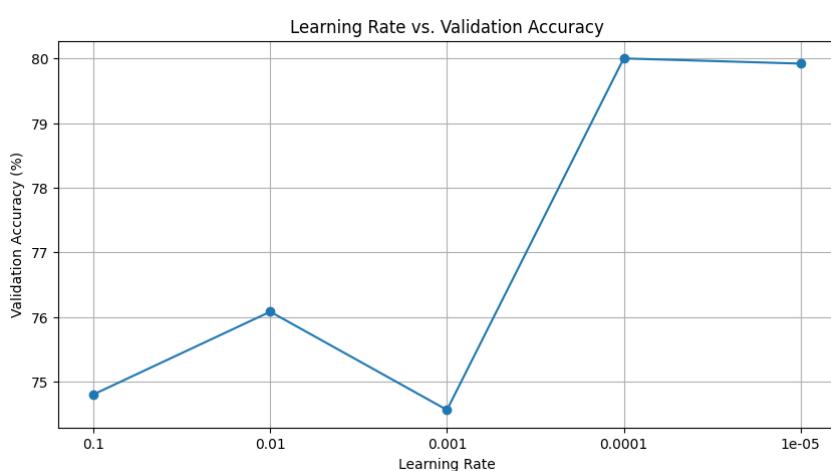


Figure 5: Validation accuracy for different learning rates.