D:\UG\_studies\sem 8\Image and video processing\Final project in correct format>python interface.py

2025-04-24 21:14:50.077552: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF\_ENABLE\_ONEDNN\_OPTS=0`.

2025-04-24 21:14:51.136892: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF\_ENABLE\_ONEDNN\_OPTS=0`.

2025-04-24 21:14:55.340047: I tensorflow/core/platform/cpu\_feature\_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.

To enable the following instructions: AVX2 AVX512F AVX512\_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.

Epoch 1/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 97s 3s/step - accuracy: 0.3497 - loss: 1.6850 - val\_accuracy: 0.5377 - val\_loss: 1.2676

Epoch 2/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 94s 3s/step - accuracy: 0.5534 - loss: 1.3049 - val\_accuracy: 0.6083 - val\_loss: 1.0629

Epoch 3/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 96s 3s/step - accuracy: 0.6119 - loss: 1.0594 - val\_accuracy: 0.6900 - val\_loss: 0.8828

Epoch 4/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 96s 3s/step - accuracy: 0.6619 - loss: 0.9427 - val\_accuracy: 0.7557 - val\_loss: 0.7005

Epoch 5/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 97s 3s/step - accuracy: 0.7312 - loss: 0.7627 - val\_accuracy: 0.8108 - val\_loss: 0.6162

Epoch 6/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 98s 3s/step - accuracy: 0.8115 - loss: 0.6207 - val\_accuracy: 0.8528 - val\_loss: 0.4609

Epoch 7/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 101s 3s/step - accuracy: 0.8344 - loss: 0.5406 - val\_accuracy: 0.8740 - val\_loss: 0.4015

Epoch 8/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 103s 3s/step - accuracy: 0.8553 - loss: 0.4601 - val\_accuracy: 0.8745 - val\_loss: 0.4119

Epoch 9/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 108s 3s/step - accuracy: 0.8644 - loss: 0.4438 - val\_accuracy: 0.8807 - val\_loss: 0.3865

Epoch 10/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 105s 3s/step - accuracy: 0.8693 - loss: 0.4257 - val\_accuracy: 0.8842 - val\_loss: 0.3762

Epoch 11/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 101s 3s/step - accuracy: 0.8765 - loss: 0.4020 - val\_accuracy: 0.8877 - val\_loss: 0.3506

Epoch 12/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 112s 3s/step - accuracy: 0.8827 - loss: 0.3846 - val\_accuracy: 0.8950 - val\_loss: 0.3358

Epoch 13/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 99s 3s/step - accuracy: 0.8855 - loss: 0.3680 - val\_accuracy: 0.8946 - val\_loss: 0.3407

Epoch 14/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 103s 3s/step - accuracy: 0.8853 - loss: 0.3654 - val\_accuracy: 0.8934 - val\_loss: 0.3351

Epoch 15/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 98s 3s/step - accuracy: 0.8913 - loss: 0.3530 - val\_accuracy: 0.9013 - val\_loss: 0.3260

Epoch 16/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 97s 3s/step - accuracy: 0.8915 - loss: 0.3510 - val\_accuracy: 0.8964 - val\_loss: 0.3258

Epoch 17/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 96s 3s/step - accuracy: 0.8947 - loss: 0.3398 - val\_accuracy: 0.8967 - val\_loss: 0.3130

Epoch 18/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 98s 3s/step - accuracy: 0.8963 - loss: 0.3261 - val\_accuracy: 0.8971 - val\_loss: 0.3240

Epoch 19/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 103s 3s/step - accuracy: 0.9008 - loss: 0.3172 - val\_accuracy: 0.9008 - val\_loss: 0.3084

Epoch 20/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 107s 3s/step - accuracy: 0.8942 - loss: 0.3088 - val\_accuracy: 0.8980 - val\_loss: 0.3179

Epoch 21/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 107s 3s/step - accuracy: 0.8994 - loss: 0.2987 - val\_accuracy: 0.8962 - val\_loss: 0.3235

Epoch 22/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 100s 3s/step - accuracy: 0.9030 - loss: 0.3023 - val\_accuracy: 0.9031 - val\_loss: 0.3077

Epoch 23/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 99s 3s/step - accuracy: 0.9023 - loss: 0.2876 - val\_accuracy: 0.9013 - val\_loss: 0.3160

Epoch 24/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 97s 3s/step - accuracy: 0.9009 - loss: 0.2918 - val\_accuracy: 0.8997 - val\_loss: 0.3248

Epoch 25/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 97s 3s/step - accuracy: 0.9077 - loss: 0.2837 - val\_accuracy: 0.8987 - val\_loss: 0.3119

Epoch 26/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 97s 3s/step - accuracy: 0.9091 - loss: 0.2774 - val\_accuracy: 0.9024 - val\_loss: 0.3097

Epoch 27/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 103s 3s/step - accuracy: 0.9089 - loss: 0.2735 - val\_accuracy: 0.9017 - val\_loss: 0.3246

Epoch 28/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 102s 3s/step - accuracy: 0.9060 - loss: 0.2745 - val\_accuracy: 0.9013 - val\_loss: 0.3342

Epoch 29/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 102s 3s/step - accuracy: 0.9093 - loss: 0.2660 - val\_accuracy: 0.9031 - val\_loss: 0.3146

Epoch 30/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 131s 4s/step - accuracy: 0.9135 - loss: 0.2650 - val\_accuracy: 0.8999 - val\_loss: 0.3265

Epoch 31/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 188s 6s/step - accuracy: 0.9143 - loss: 0.2527 - val\_accuracy: 0.9010 - val\_loss: 0.3132

Epoch 32/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 162s 5s/step - accuracy: 0.9116 - loss: 0.2632 - val\_accuracy: 0.8943 - val\_loss: 0.3355

Epoch 33/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 174s 5s/step - accuracy: 0.9146 - loss: 0.2563 - val\_accuracy: 0.8932 - val\_loss: 0.3510

Epoch 34/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 204s 5s/step - accuracy: 0.9172 - loss: 0.2469 - val\_accuracy: 0.9008 - val\_loss: 0.3253

Epoch 35/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 172s 5s/step - accuracy: 0.9186 - loss: 0.2450 - val\_accuracy: 0.8999 - val\_loss: 0.3550

Epoch 36/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 182s 5s/step - accuracy: 0.9197 - loss: 0.2405 - val\_accuracy: 0.8999 - val\_loss: 0.3399

Epoch 37/200

34/34 ━━━━━━━━━━━━━━━━━━━━ 199s 5s/step - accuracy: 0.9174 - loss: 0.2388 - val\_accuracy: 0.9006 - val\_loss: 0.3175

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my\_model.keras')` or `keras.saving.save\_model(model, 'my\_model.keras')`.

✅ Model saved to checkpoints/final\_weights.h5

Loading test data from: data

Loaded 70 valid test samples

Loading test data from: data

Loaded 70 valid test samples

✅ Loaded 70 test samples

3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 145ms/step

2025-04-24 22:32:55.463528: W tensorflow/core/framework/local\_rendezvous.cc:404] Local rendezvous is aborting with status: OUT\_OF\_RANGE: End of sequence

💾 Saved predictions to data\predictions\_report.csv

📊 Classification Report:

precision recall f1-score support

a 1.00 0.90 0.95 10

b 0.77 1.00 0.87 10

c 1.00 0.80 0.89 10

d 0.91 1.00 0.95 10

e 1.00 0.90 0.95 10

f 0.91 1.00 0.95 10

m 1.00 0.90 0.95 10

accuracy 0.93 70

macro avg 0.94 0.93 0.93 70

weighted avg 0.94 0.93 0.93 70

🎯 Confusion Matrix:

a b c d e f m

-- --- --- --- --- --- --- ---

a 9 1 0 0 0 0 0

b 0 10 0 0 0 0 0

c 0 2 8 0 0 0 0

d 0 0 0 10 0 0 0

e 0 0 0 0 9 1 0

f 0 0 0 0 0 10 0

m 0 0 0 1 0 0 9