

Model Development Phase Template

Date	20 July 2024
Team ID	SWTID1720082030
Project Title	Hydration Essentials: Classifying Water Bottle
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code demonstrates the code for the CNN model that we have used .

Initial Model Training Code (5 marks):

```
model = Sequential([
    resize_and_rescale,
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation=None),
    BatchNormalization(),
    Activation('relu'),

    Dense(64, activation=None),
    BatchNormalization(),
    Activation('relu'),

    Dense(num_classes, activation='softmax'),
])

model.build(input_shape=input_shape)
```

Model Training

```
[ ] model.compile(optimizer='SGD', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
[ ] history = model.fit(train_ds, epochs=75, validation_data=val_ds)
```

```
Epoch 1/75
12/12 [=====] - 56s 4s/step - loss: 0.9289 - accuracy: 0.5866 - val_loss: 0.9941 - val_accuracy: 0.6094
Epoch 2/75
12/12 [=====] - 53s 4s/step - loss: 0.6754 - accuracy: 0.7151 - val_loss: 0.8896 - val_accuracy: 0.7031
Epoch 3/75
12/12 [=====] - 52s 4s/step - loss: 0.6126 - accuracy: 0.7542 - val_loss: 0.9081 - val_accuracy: 0.6250
Epoch 4/75
12/12 [=====] - 52s 4s/step - loss: 0.5209 - accuracy: 0.7905 - val_loss: 0.9017 - val_accuracy: 0.5781
Epoch 5/75
12/12 [=====] - 65s 5s/step - loss: 0.5197 - accuracy: 0.8156 - val_loss: 0.8909 - val_accuracy: 0.6579
Epoch 6/75
12/12 [=====] - 52s 4s/step - loss: 0.5746 - accuracy: 0.7682 - val_loss: 0.9422 - val_accuracy: 0.5781
Epoch 7/75
12/12 [=====] - 55s 4s/step - loss: 0.4760 - accuracy: 0.8359 - val_loss: 0.8388 - val_accuracy: 0.5938
Epoch 8/75
12/12 [=====] - 54s 4s/step - loss: 0.4946 - accuracy: 0.8045 - val_loss: 0.8811 - val_accuracy: 0.5469
```

```
Epoch 73/75
12/12 [=====] - 55s 4s/step - loss: 0.0211 - accuracy: 1.0000 - val_loss: 0.0546 - val_accuracy: 0.9844
Epoch 74/75
12/12 [=====] - 62s 5s/step - loss: 0.0193 - accuracy: 0.9972 - val_loss: 0.0371 - val_accuracy: 0.7344
Epoch 75/75
12/12 [=====] - 55s 4s/step - loss: 0.0273 - accuracy: 0.9974 - val_loss: 0.0823 - val_accuracy: 0.9844

[ ] scores = model.evaluate(test_ds)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])

2/2 [=====] - 5s 1s/step - loss: 0.0427 - accuracy: 1.0000
Test loss: 0.0427468940615654
Test accuracy: 1.0
```

Model Validation and Evaluation Report (5 marks)

Model	Summary	Training and Validation Performance Metrics
CNN	<p>Used batch normalisation and dropout(optimizer:adam)</p> <pre> # Model definition model = Sequential([resize_and_rescale, Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), flatten, Dense(128, activation=None), BatchNormalization(), Activation('relu'), Dropout(0.3), # Dropout layer with a dropout rate of 0.5 Dense(64, activation=None), BatchNormalization(), Activation('relu'), Dropout(0.3), # Dropout layer with a dropout rate of 0.5 Dense(num_classes, activation='softmax'),]) model.build(input_shape=input_shape) [] model.summary() Show hidden output Model Training [10] model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy']) </pre>	<pre> Epoch 10/20 12/12 [=====] - 51s 4s/step - loss: 0.4702 - accuracy: 0.8156 - val_loss: 1.0931 - val_accuracy: 0.7056 Epoch 11/20 12/12 [=====] - 52s 4s/step - loss: 0.5226 - accuracy: 0.8240 - val_loss: 1.0705 - val_accuracy: 0.5156 Epoch 12/20 12/12 [=====] - 55s 4s/step - loss: 0.5152 - accuracy: 0.8073 - val_loss: 1.2011 - val_accuracy: 0.5938 Epoch 13/20 12/12 [=====] - 52s 4s/step - loss: 0.3686 - accuracy: 0.8603 - val_loss: 0.6035 - val_accuracy: 0.7031 Epoch 14/20 12/12 [=====] - 51s 4s/step - loss: 0.4980 - accuracy: 0.8268 - val_loss: 2.0623 - val_accuracy: 0.6719 Epoch 15/20 12/12 [=====] - 52s 4s/step - loss: 0.3808 - accuracy: 0.8436 - val_loss: 0.6620 - val_accuracy: 0.6719 Epoch 16/20 12/12 [=====] - 45s 5s/step - loss: 0.3322 - accuracy: 0.8939 - val_loss: 0.8423 - val_accuracy: 0.6094 Epoch 17/20 12/12 [=====] - 51s 4s/step - loss: 0.2081 - accuracy: 0.9022 - val_loss: 0.6745 - val_accuracy: 0.7031 Epoch 18/20 12/12 [=====] - 61s 5s/step - loss: 0.2812 - accuracy: 0.8880 - val_loss: 0.6107 - val_accuracy: 0.7012 Epoch 19/20 12/12 [=====] - 55s 4s/step - loss: 0.2775 - accuracy: 0.9010 - val_loss: 0.3635 - val_accuracy: 0.8125 Epoch 20/20 12/12 [=====] - 52s 4s/step - loss: 0.2363 - accuracy: 0.9190 - val_loss: 0.6742 - val_accuracy: 0.8125 score = model.evaluate(test_ds) print('Test loss:', score[0]) print('Test accuracy:', score[1]) 2/2 [=====] - 7s 2s/step - loss: 0.6075 - accuracy: 0.7809 Test loss: 0.697538128547668 Test accuracy: 0.700875 </pre>
CNN	<p>(optimizer:adam)</p> <p>Only 2 fully connected layers</p>	<pre> Epoch 20/25 12/12 [=====] - 59s 5s/step - loss: 0.1861 - accuracy: 0.9297 - val_loss: 0.2400 - val_accuracy: 0.8750 Epoch 21/25 12/12 [=====] - 61s 5s/step - loss: 0.1480 - accuracy: 0.9469 - val_loss: 0.1238 - val_accuracy: 0.9511 Epoch 22/25 12/12 [=====] - 51s 4s/step - loss: 0.1114 - accuracy: 0.9581 - val_loss: 0.0780 - val_accuracy: 0.9844 Epoch 23/25 12/12 [=====] - 59s 5s/step - loss: 0.0993 - accuracy: 0.9635 - val_loss: 0.0915 - val_accuracy: 0.9531 Epoch 24/25 12/12 [=====] - 48s 6s/step - loss: 0.1429 - accuracy: 0.9469 - val_loss: 0.1708 - val_accuracy: 0.9375 Epoch 25/25 12/12 [=====] - 57s 5s/step - loss: 0.1569 - accuracy: 0.9413 - val_loss: 0.1356 - val_accuracy: 0.9219 Epoch 26/25 12/12 [=====] - 56s 4s/step - loss: 0.2065 - accuracy: 0.9274 - val_loss: 0.2045 - val_accuracy: 0.9219 Epoch 27/25 12/12 [=====] - 56s 4s/step - loss: 0.2333 - accuracy: 0.9106 - val_loss: 0.2664 - val_accuracy: 0.9062 Epoch 28/25 12/12 [=====] - 58s 5s/step - loss: 0.2030 - accuracy: 0.9089 - val_loss: 0.1109 - val_accuracy: 0.9608 Epoch 29/25 12/12 [=====] - 50s 4s/step - loss: 0.1758 - accuracy: 0.9305 - val_loss: 0.1407 - val_accuracy: 0.9375 score = model.evaluate(test_ds) print('Test loss:', score[0]) print('Test accuracy:', score[1]) 2/2 [=====] - 0s 2s/step - loss: 3.7529e-04 - accuracy: 1.0000 Test loss: 0.0003352901880160287 Test accuracy: 1.0 </pre>

	<pre> input_shape = (32, 32, 3, 1) num_classes = 3 model = Sequential([resize_and_rescale, Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Flatten(), Dense(64, activation=None), Activation('relu'), Dense(num_classes, activation='softmax'),]) model.build(input_shape=input_shape) Model Training model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy']) history = model.fit(train_ds, epochs=35, validation_data=val_ds) </pre>	
CNN	<p>(optimizer:adam) 3 fully connected layers</p> <pre> model = Sequential([resize_and_rescale, Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Flatten(), Dense(128, activation=None), BatchNormalization(), Activation('relu'), Dense(64, activation=None), BatchNormalization(), Activation('relu'), Dense(num_classes, activation='softmax'),]) model.build(input_shape=input_shape) Model Training model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy']) history = model.fit(train_ds, epochs=35, validation_data=val_ds) </pre>	<pre> 12/12 [=====] 575 5s/step - loss: 0.8351 - accuracy: 1.0000 - val_loss: 0.1360 - val_accuracy: 0.9219 Epoch 32/35 12/12 [=====] 576 5s/step - loss: 0.8305 - accuracy: 0.9972 - val_loss: 0.1268 - val_accuracy: 0.9844 Epoch 33/35 12/12 [=====] 576 5s/step - loss: 0.8648 - accuracy: 0.9948 - val_loss: 0.4969 - val_accuracy: 1.0000 Epoch 34/35 12/12 [=====] 515 4s/step - loss: 0.8627 - accuracy: 0.9916 - val_loss: 0.1485 - val_accuracy: 0.9375 Epoch 35/35 12/12 [=====] 515 4s/step - loss: 0.8834 - accuracy: 0.9832 - val_loss: 0.2818 - val_accuracy: 0.9375 Epoch 36/35 12/12 [=====] 686 5s/step - loss: 0.8414 - accuracy: 1.0000 - val_loss: 0.1987 - val_accuracy: 0.9211 Epoch 37/35 12/12 [=====] 515 4s/step - loss: 0.8726 - accuracy: 1.0000 - val_loss: 0.8818 - val_accuracy: 0.9844 Epoch 38/35 12/12 [=====] 565 5s/step - loss: 0.8479 - accuracy: 0.9944 - val_loss: 0.3236 - val_accuracy: 0.9219 Epoch 39/35 12/12 [=====] 548 4s/step - loss: 0.8538 - accuracy: 0.9916 - val_loss: 0.4663 - val_accuracy: 0.8750 Epoch 40/35 12/12 [=====] 515 4s/step - loss: 0.8780 - accuracy: 1.0000 - val_loss: 0.2938 - val_accuracy: 0.9062 Epoch 41/35 12/12 [=====] 558 4s/step - loss: 0.8694 - accuracy: 0.9832 - val_loss: 0.2346 - val_accuracy: 0.8706 Epoch 42/35 12/12 [=====] 495 4s/step - loss: 0.8382 - accuracy: 0.9972 - val_loss: 0.6772 - val_accuracy: 0.9737 Epoch 43/35 12/12 [=====] 576 4s/step - loss: 0.8208 - accuracy: 1.0000 - val_loss: 0.8754 - val_accuracy: 0.9737 Epoch 44/35 12/12 [=====] 686 5s/step - loss: 0.8242 - accuracy: 1.0000 - val_loss: 0.2858 - val_accuracy: 0.9219 Epoch 45/35 12/12 [=====] 548 4s/step - loss: 0.8208 - accuracy: 1.0000 - val_loss: 0.3171 - val_accuracy: 0.9062 scores = model.evaluate(test_ds) print('Test loss:', scores[0]) print('Test accuracy:', scores[1]) 2/2 [=====] 66 2s/step - loss: 0.1713 - accuracy: 0.9375 Test loss: 0.17190800881256 Test accuracy: 0.9375 </pre>
CNN	<p>Optimiser: SGD 3 layers</p> <pre> input_shape = (32, 32, 3, 1) num_classes = 3 model = Sequential([resize_and_rescale, Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Conv2D(64, kernel_size=(3, 3), activation='relu'), MaxPooling2D(pool_size=(2, 2)), Flatten(), Dense(128, activation=None), BatchNormalization(), Activation('relu'), Dense(64, activation=None), BatchNormalization(), Activation('relu'), Dense(num_classes, activation='softmax'),]) model.build(input_shape=input_shape) model.summary() Show hidden output Model Training model.compile(optimizer='SGD', loss='sparse_categorical_crossentropy', metrics=['accuracy']) history = model.fit(train_ds, epochs=75, validation_data=val_ds) </pre>	<pre> Epoch 69/75 12/12 [=====] 575 4s/step - loss: 0.8789 - accuracy: 0.9974 - val_loss: 0.1329 - val_accuracy: 0.9737 Epoch 70/75 12/12 [=====] 555 4s/step - loss: 0.8342 - accuracy: 1.0000 - val_loss: 0.2916 - val_accuracy: 0.8986 Epoch 71/75 12/12 [=====] 525 4s/step - loss: 0.8711 - accuracy: 0.9916 - val_loss: 0.4238 - val_accuracy: 0.8594 Epoch 72/75 12/12 [=====] 495 4s/step - loss: 0.8547 - accuracy: 0.9868 - val_loss: 0.1330 - val_accuracy: 0.9737 Epoch 73/75 12/12 [=====] 555 4s/step - loss: 0.8345 - accuracy: 1.0000 - val_loss: 0.1894 - val_accuracy: 0.9688 Epoch 74/75 12/12 [=====] 525 4s/step - loss: 0.8554 - accuracy: 0.9944 - val_loss: 0.1575 - val_accuracy: 0.9375 Epoch 75/75 12/12 [=====] 585 4s/step - loss: 0.8351 - accuracy: 1.0000 - val_loss: 0.8809 - val_accuracy: 0.9844 scores = model.evaluate(test_ds) print('Test loss:', scores[0]) print('Test accuracy:', scores[1]) 2/2 [=====] 56 3s/step - loss: 0.8427 - accuracy: 1.0000 Test loss: 0.842768940615054 Test accuracy: 1.0 import matplotlib.pyplot as plt plt.title('75 epochs with sgd') plt.plot(history.history['accuracy'], color='red', label='train') plt.plot(history.history['val_accuracy'], color='blue', label='validation') plt.legend() plt.show() </pre> 