**Tiny CNN Model Training and Deployment Report**

**Project:** Tiny CNN for Binary Classification  
**Programming Language:** Python  
**IDE:** PyCharm  
**Deployment Target:** ESP32-CAM (TFLite INT8 Quantized Model)

**1. Environment Setup**

* **IDE:** PyCharm
* **Python Interpreter:** Virtual environment (.venv)
* **Required Libraries:**
* pip install ultralytics tensorflow matplotlib numpy

**2. Dataset**

* **Training Dataset:** 28 images, 2 classes
* **Validation Dataset:** 8 images, 2 classes
* **Image Size:** 96x96 (grayscale)
* **Batch Size:** 16

**3. Tiny CNN Architecture**

| **Layer Type** | **Output Shape** | **Parameters** | **Activation** | **Notes** |
| --- | --- | --- | --- | --- |
| Input | (96, 96, 1) | 0 | - | Grayscale input image |
| Conv2D | (94, 94, 8) | 80 | ReLU | Kernel size: 3x3 |
| MaxPooling2D | (47, 47, 8) | 0 | - | Pool size: 2x2 |
| Conv2D | (45, 45, 16) | 1168 | ReLU | Kernel size: 3x3 |
| MaxPooling2D | (22, 22, 16) | 0 | - | Pool size: 2x2 |
| Conv2D | (20, 20, 32) | 4640 | ReLU | Kernel size: 3x3 |
| Flatten | 12800 | 0 | - | Flatten 3D feature maps |
| Dense | 32 | 409632 | ReLU | Fully connected layer |
| Dropout | 32 | 0 | - | Dropout rate: 0.2 |
| Dense (Output) | 1 | 33 | Sigmoid | Binary classification output |

* **Total Parameters:** 415,553
* **Trainable Parameters:** 415,553
* **Non-trainable Parameters:** 0

**4. Model Compilation**

* **Optimizer:** Adam
* **Learning Rate:** 0.0001
* **Loss Function:** Binary Crossentropy
* **Metrics:** Accuracy

model.compile(

optimizer=keras.optimizers.Adam(learning\_rate=1e-4),

loss="binary\_crossentropy",

metrics=["accuracy"]

)

**5. Training Performance**

| **Epoch** | **Training Loss** | **Training Accuracy** | **Validation Loss** | **Validation Accuracy** |
| --- | --- | --- | --- | --- |
| 1 | 0.7499 | 25% | 0.4709 | 100% |
| 2 | 0.4728 | 100% | 0.2773 | 100% |
| 3 | 0.3114 | 100% | 0.1695 | 100% |
| 4 | 0.1950 | 100% | 0.1000 | 100% |
| 5 | 0.1192 | 100% | 0.0552 | 100% |
| 6 | 0.0709 | 100% | 0.0294 | 100% |
| 7 | 0.0363 | 100% | 0.0156 | 100% |
| 8 | 0.0278 | 100% | 0.0084 | 100% |
| 9 | 0.0189 | 100% | 0.0047 | 100% |
| 10 | 0.0120 | 100% | 0.0027 | 100% |

**Observation:**

* The model converged very quickly, achieving perfect validation accuracy (100%) by the second epoch.
* Training and validation losses decreased smoothly, indicating stable learning.

**6. Model Saving**

model.save("person\_tinycnn.h5")

* Model saved in HDF5 format.
* Recommended alternative: model.save('person\_tinycnn.keras') (native Keras format).

**7. INT8 Quantization and TFLite Conversion**

# Convert to INT8 TFLite

tflite\_model = converter.convert()

with open("person\_tinycnn\_int8.tflite", "wb") as f:

f.write(tflite\_model)

* **Quantization Type:** INT8 (integer-only inference)
* **Purpose:** Reduce model size and enable deployment on microcontrollers.
* **Output:** person\_tinycnn\_int8.tflite

**8. Deployment: Convert TFLite to C++ Source for ESP32-CAM**

with open(tflite\_model\_path, "rb") as f:

data = f.read()

with open(cc\_file\_path, "w") as f:

f.write("#include <cstdint>\n\n")

f.write("const unsigned char waste\_classifier\_int8\_tflite[] = {\n")

for i, byte in enumerate(data):

f.write(f"0x{byte:02x},")

if (i + 1) % 12 == 0:

f.write("\n")

f.write("\n};\n")

f.write(f"const unsigned int waste\_classifier\_int8\_tflite\_len = {len(data)};\n")

* Generated .cc file can be directly included in Arduino or ESP32-CAM projects.

**9. Summary**

| **Aspect** | **Details** |
| --- | --- |
| Model Type | Tiny CNN (Convolutional Neural Network) |
| Input Size | 96x96 grayscale images |
| Output | Binary classification (sigmoid activation) |
| Total Parameters | 415,553 |
| Loss Function | Binary Crossentropy |
| Optimizer | Adam (learning rate 0.0001) |
| Epochs | 10 |
| Batch Size | 16 |
| Data Augmentation | Random Flip, Random Rotation |
| Evaluation Metrics | Accuracy |
| Training Accuracy | Up to 100% |
| Validation Accuracy | 100% |
| Deployment Format | TFLite INT8 + C++ source file (.cc) |

**Deployment Target:** ESP32-CAM or compatible microcontroller for real-time image classification.

✅ **Conclusion:**  
The Tiny CNN model successfully achieves perfect validation accuracy on the provided dataset. The model is optimized and quantized for embedded deployment, making it suitable for edge AI applications on ESP32-CAM.