TinyML Final Project

June 10, 2024

- Bu projede arasınav için yapılan modeli gömülü cihazlarda çalıştırabilmek için Pruning, Fine-tuning, Post-quantization, Quantization Aware training yöntemleri kullanılmıştır ve arasınav-daki kodun üzerine eklenmiştir. Arasınavdan farklı olarak kodda düzenlemeler yapılmıştır.
- Oluşturulan modelin gömülü cihazlar üzerindeki performansını test edebilmek için Edge Impulse Studio üzerinden "Espressif ESP-EYE (ESP32 240MHz)" cihazı için kontrol edilmiştir. Metrikler ilgili yöntemin uygulandığı kısımda yer almaktadır.

1 Önveri işleme(Preprocessing)

- Cifar10 veri seti içeriğinde toplam **60.000 32x32** pixel'lik resimler bulunduran bir veri setidir. Veri seti **10** tane kategoriye ayrılmaktadır: Uçak, Otomobil, Kuş, Kedi, Geyik, Köpek, Kurbağa, At, Gemi, Kamyon.
- Aşağıda yer alan X değişkenindeki her bir satırdaki değerler, 32x32 pixel'lik resim için sırasıyla *kırmızı, yeşil ve mavi(RGB)* değerleri göstermektedir.
- Modeli daha da küçültebilmek için label ve ilgili resimleri verisetinin içinden kaldıran kod eklendi ve işleme süresini kısaltmak için kullanıldı. Label filtering uygulandıktan sonraki metrikler quantization işleminde gösterilmiştir.

[50]:
```
from keras.datasets import cifar10
from keras.utils import to_categorical

import numpy as np

def filter_labels(X, y, labels_to_remove):
    # Create mask
    mask = ~np.isin(y, labels_to_remove).flatten()

    # Apply mask
    X_filtered = X[mask]
    y_filtered = y[mask]

    return X_filtered, y_filtered

def make_preprocessing(X_train, y_train, X_test, y_test):
    # one hot encode uygula labellar uzerinde
    y_train = to_categorical(y_train)
    y_test = to_categorical(y_test)
```

```python
    # Resim pixellerini 0-255 arasindan 0-1 arasina float olarak cek
    X_train = X_train.astype('float32')
    X_test = X_test.astype('float32')
    X_train = X_train / 255.0
    X_test = X_test / 255.0

    return (X_train, y_train), (X_test, y_test)

# Usage example:
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

labels_to_remove = []
#labels_to_remove = [2, 3, 4, 5, 6, 7, 8, 9]

# Filter the datasets
X_train, y_train = filter_labels(X_train, y_train, labels_to_remove)
X_test, y_test = filter_labels(X_test, y_test, labels_to_remove)


print("Veriseti preprocessing yapmadan once:")

print("Veriseti sekil ozeti")
print('Train: X=%s, y=%s' % (X_train.shape, y_train.shape))
print('Test: X=%s, y=%s\n' % (X_test.shape, y_test.shape))

print("X_test: ", X_test[:1])
print("y_test: ", y_test[:1])
print("\n")

(X_train, y_train), (X_test, y_test) = make_preprocessing(X_train, y_train,␣
 ↪X_test, y_test)

print("Veriseti preprocessing yaptiktan sonra:")

print("Veriseti sekil ozeti")
print('Train: X=%s, y=%s' % (X_train.shape, y_train.shape))
print('Test: X=%s, y=%s\n' % (X_test.shape, y_test.shape))

print("X_test: ", X_test[:1])
print("y_test: ", y_test[:1])
print("\n")
```

```
Veriseti preprocessing yapmadan once:
Veriseti sekil ozeti
Train: X=(50000, 32, 32, 3), y=(50000, 1)
Test: X=(10000, 32, 32, 3), y=(10000, 1)
```

```
X_test:  [[[[158 112  49]
   [159 111  47]
   [165 116  51]
   …
   [137  95  36]
   [126  91  36]
   [116  85  33]]

  [[152 112  51]
   [151 110  40]
   [159 114  45]
   …
   [136  95  31]
   [125  91  32]
   [119  88  34]]

  [[151 110  47]
   [151 109  33]
   [158 111  36]
   …
   [139  98  34]
   [130  95  34]
   [120  89  33]]

  …

  [[ 68 124 177]
   [ 42 100 148]
   [ 31  88 137]
   …
   [ 38  97 146]
   [ 13  64 108]
   [ 40  85 127]]

  [[ 61 116 168]
   [ 49 102 148]
   [ 35  85 132]
   …
   [ 26  82 130]
   [ 29  82 126]
   [ 20  64 107]]

  [[ 54 107 160]
   [ 56 105 149]
   [ 45  89 132]
   …
   [ 24  77 124]
```

```
   [ 34  84 129]
   [ 21  67 110]]]]
y_test:  [[3]]


Veriseti preprocessing yaptiktan sonra:
Veriseti sekil ozeti
Train: X=(50000, 32, 32, 3), y=(50000, 10)
Test: X=(10000, 32, 32, 3), y=(10000, 10)

X_test:  [[[[0.61960787 0.4392157  0.19215687]
   [0.62352943 0.43529412 0.18431373]
   [0.64705884 0.45490196 0.2       ]
   …
   [0.5372549  0.37254903 0.14117648]
   [0.49411765 0.35686275 0.14117648]
   [0.45490196 0.33333334 0.12941177]]

  [[0.59607846 0.4392157  0.2       ]
   [0.5921569  0.43137255 0.15686275]
   [0.62352943 0.44705883 0.1764706 ]
   …
   [0.53333336 0.37254903 0.12156863]
   [0.49019608 0.35686275 0.1254902 ]
   [0.46666667 0.34509805 0.13333334]]

  [[0.5921569  0.43137255 0.18431373]
   [0.5921569  0.42745098 0.12941177]
   [0.61960787 0.43529412 0.14117648]
   …
   [0.54509807 0.38431373 0.13333334]
   [0.50980395 0.37254903 0.13333334]
   [0.47058824 0.34901962 0.12941177]]

  …

  [[0.26666668 0.4862745  0.69411767]
   [0.16470589 0.39215687 0.5803922 ]
   [0.12156863 0.34509805 0.5372549 ]
   …
   [0.14901961 0.38039216 0.57254905]
   [0.05098039 0.2509804  0.42352942]
   [0.15686275 0.33333334 0.49803922]]

  [[0.23921569 0.45490196 0.65882355]
   [0.19215687 0.4        0.5803922 ]
   [0.13725491 0.33333334 0.5176471 ]
   …
```

```
  [0.10196079 0.32156864 0.50980395]
  [0.11372549 0.32156864 0.49411765]
  [0.07843138 0.2509804  0.41960785]]

 [[0.21176471 0.41960785 0.627451  ]
  [0.21960784 0.4117647  0.58431375]
  [0.1764706  0.34901962 0.5176471 ]
  …
  [0.09411765 0.3019608  0.4862745 ]
  [0.13333334 0.32941177 0.5058824 ]
  [0.08235294 0.2627451  0.43137255]]]]
y_test:  [[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]]
```

2 Model Eğitimi ve Modeli Kaydetme

2.1 Katman Açıklamaları

- Used link for this model: https://www.kaggle.com/code/ektasharma/simple-cifar10-cnn-keras-code-with-88-accuracy

1. Conv2D
   - Görüntü verileri üzerinde konvolüsyon işlemi gerçekleştirir. Aktivasyon fonksiyonu (burada ReLU) ile çıktıları sıkıştırır, böylece modelin öğrenme yeteneğini artırır.
2. BatchNormalization
   - Ağdaki her katmandan gelen çıktıları normalleştirir, yani ortalamayı sıfıra ve standart sapmayı bir birimlik varyansa ayarlar. Bu, ağın daha hızlı öğrenmesine yardımcı olurken, overfitting'i azaltabilir.
3. MaxPooling2D
   - Her bir bölgenin maksimum değerini alarak bir örüntüyü küçültür ve özellikleri korur. Bu, ağın daha derin ve karmaşık özellikleri öğrenmesine yardımcı olurken, hesaplama maliyetini düşürür.
4. Dropout
   - Belirli bir olasılıkla (aşağıdaki modelde 0.3 veya 0.5) rastgele nöronları devre dışı bırakarak, modelin öğrenme sürecinde nöronların aşırı özelleşmesini önler. Bu, ağın daha genelleştirilmiş ve daha iyi performans gösteren bir model oluşturmasına yardımcı olur.
5. Flatten
   - CNN'de kullanılan konvolüsyon ve havuzlama katmanlarından gelen çıktılar, genellikle 2 Boyutlu veya 3 Boyutlu tensörlerdir(bir görüntünün yükseklik, genişlik ve kanal sayısı). Flatten katmanı, bu 2 Boyutlu veya 3 Boyutlu tensörleri tek boyutlu vektörlere dönüştürerek, bir sonraki katman olan Dense katmanına giriş olarak kullanılacak veri yapısını sağlar.
6. Dense
   - Bu katman, girişten gelen verilerle ağırlıklar arasında nokta çarpımı yapar, ardından bir aktivasyon fonksiyonu uygular ve bir çıkış üretir.

[45]:
```python
from tensorflow_model_optimization.python.core.keras.compat import keras
from keras.losses import categorical_crossentropy

# Used link for this model: https://www.kaggle.com/code/ektasharma/
↪simple-cifar10-cnn-keras-code-with-88-accuracy
model = keras.Sequential([
    keras.layers.InputLayer(input_shape=(32, 32, 3)),

    keras.layers.Conv2D(32, (3, 3), padding='same', activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(32, (3, 3), padding='same', activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Dropout(0.3),

    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
    keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Dropout(0.5),

    keras.layers.Conv2D(128, (3, 3), padding='same', activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(128, (3, 3), padding='same', activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Dropout(0.5),

    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(len(y_train[0]), activation='softmax')
])


model.compile(optimizer='adam', loss=categorical_crossentropy,␣
↪metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, batch_size=64, validation_data=(X_test,␣
↪y_test), verbose=1)

#model.save('models/cifar_model_midterm.keras')

model.save('models/cifar_model_final.keras')
```

```
Epoch 1/10
782/782 [==============================] - 69s 86ms/step - loss: 1.7741 -
accuracy: 0.3914 - val_loss: 1.6847 - val_accuracy: 0.4353
Epoch 2/10
782/782 [==============================] - 70s 89ms/step - loss: 1.2386 -
accuracy: 0.5567 - val_loss: 1.0823 - val_accuracy: 0.6169
Epoch 3/10
782/782 [==============================] - 73s 93ms/step - loss: 1.0267 -
accuracy: 0.6388 - val_loss: 0.9314 - val_accuracy: 0.6742
Epoch 4/10
782/782 [==============================] - 74s 94ms/step - loss: 0.9000 -
accuracy: 0.6877 - val_loss: 0.9383 - val_accuracy: 0.6731
Epoch 5/10
782/782 [==============================] - 77s 98ms/step - loss: 0.8251 -
accuracy: 0.7118 - val_loss: 0.7290 - val_accuracy: 0.7454
Epoch 6/10
782/782 [==============================] - 78s 100ms/step - loss: 0.7695 -
accuracy: 0.7331 - val_loss: 0.7063 - val_accuracy: 0.7555
Epoch 7/10
782/782 [==============================] - 75s 96ms/step - loss: 0.7311 -
accuracy: 0.7489 - val_loss: 0.9182 - val_accuracy: 0.6890
Epoch 8/10
782/782 [==============================] - 75s 96ms/step - loss: 0.6900 -
accuracy: 0.7613 - val_loss: 0.6134 - val_accuracy: 0.7910
Epoch 9/10
782/782 [==============================] - 80s 102ms/step - loss: 0.6616 -
accuracy: 0.7732 - val_loss: 0.5648 - val_accuracy: 0.8046
Epoch 10/10
782/782 [==============================] - 77s 98ms/step - loss: 0.6291 -
accuracy: 0.7835 - val_loss: 0.5614 - val_accuracy: 0.8097
```

3 Model Dosyadan Yükleme ve Doğruluk Değeri Hesaplama

[46]:

```
print("Model Dosyadan Yukleniyor...\n")
#loaded_model = load_model('models/cifar_model_midterm.keras')
loaded_model = keras.models.load_model('models/cifar_model_final.keras')

print("Model Degerlendiriliyor...")
_, acc = loaded_model.evaluate(X_test, y_test, verbose=1)
print('Dogruluk(Accuracy) Yuzdelik Oran: %.3f' % (acc * 100.0))
```

```
Model Dosyadan Yukleniyor…

Model Degerlendiriliyor…
313/313 [==============================] - 6s 17ms/step - loss: 0.5614 -
accuracy: 0.8097
Dogruluk(Accuracy) Yuzdelik Oran: 80.970
```

4 Model Parametreleri Analizi

[47]:
```
loaded_model.summary()
```

```
Model: "sequential_2"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 32, 32, 32)        896

 batch_normalization_14 (Ba  (None, 32, 32, 32)        128
 tchNormalization)

 conv2d_13 (Conv2D)          (None, 32, 32, 32)        9248

 batch_normalization_15 (Ba  (None, 32, 32, 32)        128
 tchNormalization)

 max_pooling2d_6 (MaxPoolin  (None, 16, 16, 32)        0
 g2D)

 dropout_8 (Dropout)         (None, 16, 16, 32)        0

 batch_normalization_16 (Ba  (None, 16, 16, 32)        128
 tchNormalization)

 conv2d_14 (Conv2D)          (None, 16, 16, 64)        18496

 conv2d_15 (Conv2D)          (None, 16, 16, 64)        36928

 batch_normalization_17 (Ba  (None, 16, 16, 64)        256
 tchNormalization)

 max_pooling2d_7 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 dropout_9 (Dropout)         (None, 8, 8, 64)          0

 conv2d_16 (Conv2D)          (None, 8, 8, 128)         73856

 batch_normalization_18 (Ba  (None, 8, 8, 128)         512
 tchNormalization)

 conv2d_17 (Conv2D)          (None, 8, 8, 128)         147584

 batch_normalization_19 (Ba  (None, 8, 8, 128)         512
 tchNormalization)
```

```
 max_pooling2d_8 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)

 dropout_10 (Dropout)        (None, 4, 4, 128)         0

 flatten_2 (Flatten)         (None, 2048)              0

 dense_4 (Dense)             (None, 128)               262272

 batch_normalization_20 (Ba  (None, 128)               512
 tchNormalization)

 dropout_11 (Dropout)        (None, 128)               0

 dense_5 (Dense)             (None, 10)                1290

=================================================================
Total params: 552746 (2.11 MB)
Trainable params: 551658 (2.10 MB)
Non-trainable params: 1088 (4.25 KB)
_________________________________________________________________
```

- Model parametrelerinin toplamı kadar hafızaya ihtiyaç duyulur.
- Model parametreleri aşağıdaki şekilde toplanırsa, aşağıdaki kodun sonucu kadar hafızaya ihtiyaç olduğu gözlenir:

[5]:
```
print("Modeli barındırmak için KB cinsinden hafıza ihtiyacı = ",␣
 ↪(896+128+9248+128+0+0+128+18496+36928+256+0+0+
                                                                    ␣
 ↪73856+512+147584+512+0+0+0+262272+512+0+1290) / 1024)
```

```
Modeli barındırmak için KB cinsinden hafıza ihtiyacı =  539.791015625
```

- Ancak model parametrelerinin özet kısmındaki değeri alırsak: *Total params: 1,656,064*, **6.32MB** hafızaya ihtiyaç olduğu gözlenir

- 8MB(**4MB program hafızası** ve 4MB spiffs hafıza) hafızaya sahip ESP32S3 modeline göre değerlendirme yapılırsa, modelin ancak hafıza düzenlemesi(spiffs hafızadan, program hafızasına alan aktarma) yapıldıktan sonra sığabileceği gözlenir. Yada pruning(budama) işlemi ile model, >=2.32MB budandıktan sonra program hafızasına sığabilir.

[48]:
```
import numpy as np
from sklearn.metrics import precision_score, recall_score, f1_score,␣
 ↪confusion_matrix

print("X_test uzerinden tahmin yapiliyor...")
y_pred = loaded_model.predict(X_test)

y_test = np.argmax(y_test, axis=1)
```

```
y_pred = np.argmax(y_pred, axis=1)
print(y_test)
print(y_pred)

precision = precision_score(y_test, y_pred, average='micro')
print("Precision:", precision)

recall = recall_score(y_test, y_pred, average='micro')
print("Recall:", recall)

f1 = f1_score(y_test, y_pred, average='micro')
print("F1 Score:", f1)

conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
X_test uzerinden tahmin yapiliyor…
313/313 [==============================] - 6s 18ms/step
[3 8 8 … 5 1 7]
[3 8 8 … 5 1 7]
Precision: 0.8097
Recall: 0.8097
F1 Score: 0.8097
Confusion Matrix:
[[811  12  19  11  18   1   4   4 106  14]
 [  9 906   0   1   2   0   3   0  43  36]
 [ 89   1 649  35 125  25  38  14  18   6]
 [ 21   2  49 662  67  98  33  26  32  10]
 [ 11   0  29  27 872  11  17  26   7   0]
 [  9   0  39 157  48 677  12  40  15   3]
 [  8   4  25  36  55   6 842   7  17   0]
 [ 15   1  29  31  43  17   2 855   4   3]
 [ 21   6   1   1   6   2   2   1 955   5]
 [ 33  49   1   5   3   0   5   4  32 868]]
```

5 Pruning ve Fine Tuning İşlemi

- Used link for this method: https://www.tensorflow.org/model_optimization/guide/pruning/pruning_with_

[51]:
```
import tensorflow_model_optimization as tfmot

# Eger kod dogru calismaz ise preprocessing islemini yeniden yapin

prune_low_magnitude = tfmot.sparsity.keras.prune_low_magnitude

batch_size = 128
```

```
epochs = 1
validation_split = 0.1 # 10% of training set will be used for validation set.

num_images = X_train.shape[0] * (1 - validation_split)
end_step = np.ceil(num_images / batch_size).astype(np.int32) * epochs

# Define model for pruning.
pruning_params = {
      'pruning_schedule': tfmot.sparsity.keras.
 ↪PolynomialDecay(initial_sparsity=0.50,
                                                                final_sparsity=0.
 ↪80,
                                                                begin_step=0,␣
 ↪end_step=end_step)
}


loaded_model = keras.models.load_model('models/cifar_model_final.keras')

model_for_pruning = prune_low_magnitude(loaded_model, **pruning_params)

# `prune_low_magnitude` requires a recompile.
model_for_pruning.compile(optimizer='adam',
              loss=keras.losses.CategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])

import tempfile

logdir = tempfile.mkdtemp()

callbacks = [
  tfmot.sparsity.keras.UpdatePruningStep(),
  tfmot.sparsity.keras.PruningSummaries(log_dir=logdir),
]

model_for_pruning.fit(X_train, y_train,
                  batch_size=batch_size, epochs=epochs,␣
 ↪validation_split=validation_split,
                  callbacks=callbacks)

model_for_pruning.save('models/cifar_pruned_finetuned.keras')

print("Model Degerlendiriliyor...")
_, acc_pruned = model_for_pruning.evaluate(X_test, y_test, verbose=1)
print('Dogruluk(Accuracy) Yuzdelik Oran: %.3f' % (acc_pruned * 100.0))
```

```
model_for_pruning.summary()
```

```
352/352 [==============================] - 62s 163ms/step - loss: 0.7515 -
accuracy: 0.7446 - val_loss: 0.7043 - val_accuracy: 0.7638
Model Degerlendiriliyor…
313/313 [==============================] - 6s 19ms/step - loss: 0.7588 -
accuracy: 0.7435
Dogruluk(Accuracy) Yuzdelik Oran: 74.350
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 prune_low_magnitude_conv2d  (None, 32, 32, 32)        1762
 _12 (PruneLowMagnitude)

 prune_low_magnitude_batch_  (None, 32, 32, 32)        129
 normalization_14 (PruneLow
 Magnitude)

 prune_low_magnitude_conv2d  (None, 32, 32, 32)        18466
 _13 (PruneLowMagnitude)

 prune_low_magnitude_batch_  (None, 32, 32, 32)        129
 normalization_15 (PruneLow
 Magnitude)

 prune_low_magnitude_max_po  (None, 16, 16, 32)        1
 oling2d_6 (PruneLowMagnitu
 de)

 prune_low_magnitude_dropou  (None, 16, 16, 32)        1
 t_8 (PruneLowMagnitude)

 prune_low_magnitude_batch_  (None, 16, 16, 32)        129
 normalization_16 (PruneLow
 Magnitude)

 prune_low_magnitude_conv2d  (None, 16, 16, 64)        36930
 _14 (PruneLowMagnitude)

 prune_low_magnitude_conv2d  (None, 16, 16, 64)        73794
 _15 (PruneLowMagnitude)

 prune_low_magnitude_batch_  (None, 16, 16, 64)        257
 normalization_17 (PruneLow
 Magnitude)
```

```
 prune_low_magnitude_max_po  (None, 8, 8, 64)          1
 oling2d_7 (PruneLowMagnitu
 de)

 prune_low_magnitude_dropou  (None, 8, 8, 64)          1
 t_9 (PruneLowMagnitude)

 prune_low_magnitude_conv2d  (None, 8, 8, 128)         147586
 _16 (PruneLowMagnitude)

 prune_low_magnitude_batch_  (None, 8, 8, 128)         513
 normalization_18 (PruneLow
 Magnitude)

 prune_low_magnitude_conv2d  (None, 8, 8, 128)         295042
 _17 (PruneLowMagnitude)

 prune_low_magnitude_batch_  (None, 8, 8, 128)         513
 normalization_19 (PruneLow
 Magnitude)

 prune_low_magnitude_max_po  (None, 4, 4, 128)         1
 oling2d_8 (PruneLowMagnitu
 de)

 prune_low_magnitude_dropou  (None, 4, 4, 128)         1
 t_10 (PruneLowMagnitude)

 prune_low_magnitude_flatte  (None, 2048)              1
 n_2 (PruneLowMagnitude)

 prune_low_magnitude_dense_  (None, 128)               524418
 4 (PruneLowMagnitude)

 prune_low_magnitude_batch_  (None, 128)               513
 normalization_20 (PruneLow
 Magnitude)

 prune_low_magnitude_dropou  (None, 128)               1
 t_11 (PruneLowMagnitude)

 prune_low_magnitude_dense_  (None, 10)                2572
 5 (PruneLowMagnitude)

=================================================================
Total params: 1102761 (4.21 MB)
Trainable params: 551658 (2.10 MB)
Non-trainable params: 551103 (2.10 MB)
```

---

6 Modeli Dahada Küçültmek için Quantization İşlemi ve Tflite Dosyasına Dönüştürme

- Ağırlıklar ve Aktivasyon fonksiyonları int8'e quantized edildiği zaman doğruluk oranında yüksek bir düşüş(10% gibi) yaşandığı için aktivasyon fonksiyonları float32, ağırlıklar int8 olarak tutuldu.
- Edge Impulse Studio üzerinden Espressif ESP-EYE (ESP32 240MHz) cihazı için kontrol edildiği zaman tflite modelin aşağıdaki performansa sahip olduğu görülebilir: PROCESSING TIME: 1323212 ms(1323.212 second). RAM USAGE: 83.7K FLASH USAGE: 592.2K
- Ram ve flash hafıza değerleri uygun olsada işleme süresi çok uzun olmaktadır.
- Label filtering(Sadece Uçak ve Otomobil) uyguladıktan sonra modelin performansı aşağıdaki gibi olmaktadır: PROCESSING TIME: 1268225 ms. RAM USAGE: 83.7K FLASH USAGE: 591.2K
- Yani label filtering işlemi, işleme süresinin azalmasına yardımcı olmadı. 2 tane label bırakıldıktan sonra doğruluk oranı %96 civarına çıktı.

[58]:
```
import tensorflow
from tensorflow_model_optimization.python.core.keras.compat import keras
import numpy as np
import tensorflow_model_optimization as tfmot

model_for_export = tfmot.sparsity.keras.strip_pruning(model_for_pruning)


def representative_dataset_gen():
    for _ in range(100):
        # Get a random batch of images from your dataset
        data = X_train[np.random.choice(X_train.shape[0], 1, replace=False)]
        yield [data.astype(np.float32)]

converter = tensorflow.lite.TFLiteConverter.from_keras_model(model_for_export)
converter.optimizations = [tensorflow.lite.Optimize.DEFAULT]


converter.representative_dataset = representative_dataset_gen


# Use mixed precision quantization (weights in int8, activations in float32)
converter.target_spec.supported_ops = [
    tf.lite.OpsSet.TFLITE_BUILTINS,  # allow float32 activations
    tf.lite.OpsSet.TFLITE_BUILTINS_INT8  # allow int8 weights
]
```

```python
# When applying below code the accuracy drop significantly(10%) - Full Integer␣
↪Quantization
'''
# Specify the target_spec to ensure full integer quantization
converter.target_spec.supported_ops = [tensorflow.lite.OpsSet.
↪TFLITE_BUILTINS_INT8]

# Ensure the input and output tensors are int8
converter.inference_input_type = tensorflow.int8
converter.inference_output_type = tensorflow.int8
'''

tflite_model = converter.convert()
with open('quantized_model.tflite', 'wb') as f:
    f.write(tflite_model)
```

```
INFO:tensorflow:Assets written to: /tmp/tmplgiw50cy/assets
```

```
INFO:tensorflow:Assets written to: /tmp/tmplgiw50cy/assets
/home/d3v3lop3r/.local/lib/python3.10/site-
packages/tensorflow/lite/python/convert.py:964: UserWarning: Statistics for
quantized inputs were expected, but not specified; continuing anyway.
  warnings.warn(
W0000 00:00:1717927310.686009   13266 tf_tfl_flatbuffer_helpers.cc:390] Ignored
output_format.
W0000 00:00:1717927310.686019   13266 tf_tfl_flatbuffer_helpers.cc:393] Ignored
drop_control_dependency.
2024-06-09 13:01:50.686128: I tensorflow/cc/saved_model/reader.cc:83] Reading
SavedModel from: /tmp/tmplgiw50cy
2024-06-09 13:01:50.687971: I tensorflow/cc/saved_model/reader.cc:51] Reading
meta graph with tags { serve }
2024-06-09 13:01:50.687984: I tensorflow/cc/saved_model/reader.cc:146] Reading
SavedModel debug info (if present) from: /tmp/tmplgiw50cy
2024-06-09 13:01:50.704112: I tensorflow/cc/saved_model/loader.cc:234] Restoring
SavedModel bundle.
2024-06-09 13:01:50.752197: I tensorflow/cc/saved_model/loader.cc:218] Running
initialization op on SavedModel bundle at path: /tmp/tmplgiw50cy
2024-06-09 13:01:50.768342: I tensorflow/cc/saved_model/loader.cc:317]
SavedModel load for tags { serve }; Status: success: OK. Took 82217
microseconds.
fully_quantize: 0, inference_type: 6, input_inference_type: FLOAT32,
output_inference_type: FLOAT32
```

7 Tflite Modelin Doğruluk Oranının Hesaplanması

[59]:
```python
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Eger label filtering yapildiysa preprocessing isleminde burada tekrar cifar10␣
↪veri setini "Prepare the test dataset"
# kisminda oldugu gibi yuklemek accuracy'nin dogru bulunamamasına yol acabilir.

# Load the quantized TFLite model
interpreter = tf.lite.Interpreter(model_path="quantized_model.tflite")
interpreter.allocate_tensors()

# Get input and output tensors
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Prepare the test dataset
(_, _), (X_test, y_test) = keras.datasets.cifar10.load_data()
X_test = X_test.astype('float32') / 255.0
y_test = keras.utils.to_categorical(y_test, 10)

# Function to run inference on a single input
def run_inference(input_data):
    input_data = np.expand_dims(input_data, axis=0).astype(np.float32)
    #input_data = np.expand_dims(input_data, axis=0).astype(np.int8)
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()
    output_data = interpreter.get_tensor(output_details[0]['index'])
    return output_data

# Calculate accuracy
correct_predictions = 0
total_predictions = len(X_test)

for i in range(total_predictions):
    input_data = X_test[i]
    true_label = np.argmax(y_test[i])

    # Get model prediction
    output_data = run_inference(input_data)
    predicted_label = np.argmax(output_data)

    if predicted_label == true_label:
        correct_predictions += 1
```

```
accuracy = correct_predictions / total_predictions
print(f"Model accuracy after quantization: {accuracy * 100:.2f}%")
```

```
Model accuracy after quantization: 71.26%
```

8 Doğruluk Oranını Arttırmak için Quantization Aware Training İşleminin Denenmesi

- Bu yöntem post-quantization yaparken full-integer quantization yaptıktan sonra doğruluk oranının ciddi bir şekilde düşmesinden ötürü denedi.
- Quantization Aware Training yaparken BatchNormalization katmanından dolayı hata verdiğinden ötürü yeni model katmanları kullanılarak model oluşturuldu.
- Used link for this model: https://github.com/Ermlab/cifar10keras
- Used link for this quantization method: https://www.tensorflow.org/model_optimization/guide/quantizatio

[4]:
```
import tensorflow_model_optimization as tfmot
from keras.datasets import cifar10
from keras.utils import to_categorical
from tensorflow_model_optimization.python.core.keras.compat import keras
import tensorflow
import tensorflow as tf
import numpy as np

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# one hot encode uygula labellar uzerinde
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Resim pixellerini 0-255 arasindan 0-1 arasina float olarak cek
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0

# Batch Normalization is must be removed for Quantization Aware
# Training so below model is used without batch normalization

# Used link for this model: https://github.com/Ermlab/cifar10keras
model = keras.Sequential([
    keras.layers.Conv2D(32, (3, 3), padding='same', activation='relu',␣
 ↪input_shape=(32, 32, 3)),
    keras.layers.Dropout(0.2),
    keras.layers.Conv2D(32, (3, 3), padding='same', activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
    keras.layers.Dropout(0.2),
```

```
    keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Conv2D(128, (3, 3), padding='same', activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Conv2D(128, (3, 3), padding='same', activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(1024, activation='relu', kernel_constraint=keras.
 ↪constraints.max_norm(3)),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(len(y_train[0]), activation='softmax')
])

sgd = keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=False)
model.compile(optimizer=sgd, loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, batch_size=64, validation_data=(X_test,␣
 ↪y_test), verbose=1)

quantize_model = tfmot.quantization.keras.quantize_model

# q_aware stands for for quantization aware.
q_aware_model = quantize_model(model)

# `quantize_model` requires a recompile.
q_aware_model.compile(optimizer='adam',
              loss=keras.losses.CategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])

q_aware_model.summary()

train_images_subset = X_train[0:1000] # out of 60000
train_labels_subset = y_train[0:1000]

q_aware_model.fit(train_images_subset, train_labels_subset,
                  batch_size=500, epochs=1, validation_split=0.1)

_, baseline_model_accuracy = model.evaluate(
    X_test, y_test, verbose=0)

_, q_aware_model_accuracy = q_aware_model.evaluate(
   X_test, y_test, verbose=0)

print('Baseline test accuracy:', baseline_model_accuracy)
print('Quant test accuracy:', q_aware_model_accuracy)
```

```python
def representative_dataset_gen():
    for _ in range(100):
        # Get a random batch of images from your dataset
        data = X_train[np.random.choice(X_train.shape[0], 1, replace=False)]
        yield [data.astype(np.float32)]

converter = tensorflow.lite.TFLiteConverter.from_keras_model(q_aware_model)
converter.optimizations = [tensorflow.lite.Optimize.DEFAULT]

converter.representative_dataset = representative_dataset_gen

# Specify the target_spec to ensure full integer quantization
converter.target_spec.supported_ops = [tensorflow.lite.OpsSet.
 ↪TFLITE_BUILTINS_INT8]

# Ensure the input and output tensors are int8
converter.inference_input_type = tensorflow.int8
converter.inference_output_type = tensorflow.int8

tflite_model = converter.convert()
with open('quantized_model.tflite', 'wb') as f:
    f.write(tflite_model)
```

```
Epoch 1/10
782/782 [==============================] - 64s 81ms/step - loss: 1.9559 -
accuracy: 0.2809 - val_loss: 1.6377 - val_accuracy: 0.4383
Epoch 2/10
782/782 [==============================] - 68s 87ms/step - loss: 1.4916 -
accuracy: 0.4581 - val_loss: 1.3321 - val_accuracy: 0.5136
Epoch 3/10
782/782 [==============================] - 77s 98ms/step - loss: 1.2905 -
accuracy: 0.5362 - val_loss: 1.1999 - val_accuracy: 0.5645
Epoch 4/10
782/782 [==============================] - 76s 98ms/step - loss: 1.1463 -
accuracy: 0.5892 - val_loss: 1.0674 - val_accuracy: 0.6177
Epoch 5/10
782/782 [==============================] - 74s 95ms/step - loss: 1.0100 -
accuracy: 0.6418 - val_loss: 0.9233 - val_accuracy: 0.6703
Epoch 6/10
782/782 [==============================] - 72s 92ms/step - loss: 0.8990 -
accuracy: 0.6821 - val_loss: 0.8311 - val_accuracy: 0.7076
Epoch 7/10
782/782 [==============================] - 76s 97ms/step - loss: 0.8028 -
accuracy: 0.7177 - val_loss: 0.7942 - val_accuracy: 0.7232
Epoch 8/10
782/782 [==============================] - 74s 94ms/step - loss: 0.7320 -
```

```
accuracy: 0.7422 - val_loss: 0.8557 - val_accuracy: 0.7069
Epoch 9/10
782/782 [==============================] - 77s 99ms/step - loss: 0.6665 -
accuracy: 0.7661 - val_loss: 0.7101 - val_accuracy: 0.7569
Epoch 10/10
782/782 [==============================] - 72s 92ms/step - loss: 0.6077 -
accuracy: 0.7858 - val_loss: 0.6914 - val_accuracy: 0.7606
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 quantize_layer_2 (Quantize  (None, 32, 32, 3)         3
 Layer)

 quant_conv2d_12 (QuantizeW  (None, 32, 32, 32)        963
 rapperV2)

 quant_dropout_10 (Quantize  (None, 32, 32, 32)        1
 WrapperV2)

 quant_conv2d_13 (QuantizeW  (None, 32, 32, 32)        9315
 rapperV2)

 quant_max_pooling2d_6 (Qua  (None, 16, 16, 32)        1
 ntizeWrapperV2)

 quant_conv2d_14 (QuantizeW  (None, 16, 16, 64)        18627
 rapperV2)

 quant_dropout_11 (Quantize  (None, 16, 16, 64)        1
 WrapperV2)

 quant_conv2d_15 (QuantizeW  (None, 16, 16, 64)        37059
 rapperV2)

 quant_max_pooling2d_7 (Qua  (None, 8, 8, 64)          1
 ntizeWrapperV2)

 quant_conv2d_16 (QuantizeW  (None, 8, 8, 128)         74115
 rapperV2)

 quant_dropout_12 (Quantize  (None, 8, 8, 128)         1
 WrapperV2)

 quant_conv2d_17 (QuantizeW  (None, 8, 8, 128)         147843
 rapperV2)

 quant_max_pooling2d_8 (Qua  (None, 4, 4, 128)         1
```

```
 ntizeWrapperV2)

 quant_flatten_2 (QuantizeW  (None, 2048)            1
 rapperV2)

 quant_dropout_13 (Quantize  (None, 2048)            1
 WrapperV2)

 quant_dense_4 (QuantizeWra  (None, 1024)            2098181
 pperV2)

 quant_dropout_14 (Quantize  (None, 1024)            1
 WrapperV2)

 quant_dense_5 (QuantizeWra  (None, 10)              10255
 pperV2)

=================================================================
Total params: 2396370 (9.14 MB)
Trainable params: 2395434 (9.14 MB)
Non-trainable params: 936 (3.66 KB)
_________________________________________________________________
2/2 [==============================] - 4s 1s/step - loss: 0.7571 - accuracy:
0.7156 - val_loss: 0.4268 - val_accuracy: 0.8900
Baseline test accuracy: 0.7605999708175659
Quant test accuracy: 0.7415000200271606
INFO:tensorflow:Assets written to: /tmp/tmpyo93diu3/assets

INFO:tensorflow:Assets written to: /tmp/tmpyo93diu3/assets
/home/d3v3lop3r/.local/lib/python3.10/site-
packages/tensorflow/lite/python/convert.py:964: UserWarning: Statistics for
quantized inputs were expected, but not specified; continuing anyway.
  warnings.warn(
W0000 00:00:1717853204.584972  134275 tf_tfl_flatbuffer_helpers.cc:390] Ignored
output_format.
W0000 00:00:1717853204.584994  134275 tf_tfl_flatbuffer_helpers.cc:393] Ignored
drop_control_dependency.
2024-06-08 16:26:44.585141: I tensorflow/cc/saved_model/reader.cc:83] Reading
SavedModel from: /tmp/tmpyo93diu3
2024-06-08 16:26:44.589804: I tensorflow/cc/saved_model/reader.cc:51] Reading
meta graph with tags { serve }
2024-06-08 16:26:44.589821: I tensorflow/cc/saved_model/reader.cc:146] Reading
SavedModel debug info (if present) from: /tmp/tmpyo93diu3
2024-06-08 16:26:44.627249: I tensorflow/cc/saved_model/loader.cc:234] Restoring
SavedModel bundle.
2024-06-08 16:26:44.748744: I tensorflow/cc/saved_model/loader.cc:218] Running
initialization op on SavedModel bundle at path: /tmp/tmpyo93diu3
2024-06-08 16:26:44.782131: I tensorflow/cc/saved_model/loader.cc:317]
```

```
SavedModel load for tags { serve }; Status: success: OK. Took 196991
microseconds.
fully_quantize: 0, inference_type: 6, input_inference_type: INT8,
output_inference_type: INT8
```

8.1 Quantization Aware Training ile Oluşturulan Tflite Dosyasının Doğruluk Oranını Bulma İşlemi

[6]:

```python
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Load the quantized TFLite model
interpreter = tf.lite.Interpreter(model_path="quantized_model.tflite")
interpreter.allocate_tensors()

# Get input and output tensors
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Prepare the test dataset
(_, _), (X_test, y_test) = keras.datasets.cifar10.load_data()
X_test = X_test.astype('float32') / 255.0
y_test = keras.utils.to_categorical(y_test, 10)

# Function to run inference on a single input
def run_inference(input_data):
    input_data = np.expand_dims(input_data, axis=0).astype(np.int8)
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()
    output_data = interpreter.get_tensor(output_details[0]['index'])
    return output_data

# Calculate accuracy
correct_predictions = 0
total_predictions = len(X_test)

for i in range(total_predictions):
    input_data = X_test[i]
    true_label = np.argmax(y_test[i])

    # Get model prediction
    output_data = run_inference(input_data)
    predicted_label = np.argmax(output_data)

    if predicted_label == true_label:
        correct_predictions += 1
```

```
accuracy = correct_predictions / total_predictions
print(f"Model accuracy after quantization: {accuracy * 100:.2f}%")
```

```
Model accuracy after quantization: 10.00%
```

- Doğruluk oranı %10 olarak belirlendi. Post-quantization yöntemi ile aynı doğruluk değeri döndürdüğünden ötürü quantization aware training işlemi de full-integer quantization işlemi için doğruluk oranını arttırmadı.

9 Modelin İşleme Hızını Azaltabilmek için Yeni Parametreler ile Modelin Eğitilmesi ve Pruning, Fine-tuning ve Post-quantization İşlemlerinin Yeniden Uygulanması

- Yeni modelin parametreleri için kullanılan link: https://studio.edgeimpulse.com/public/51070/latest/acquisit
- Edge Impulse üzerinde eğitilmiş modelin performans değerli şu şekilde verilmektedir: ACCURACY: 74.4% INFERENCING TIME: 1251 ms. PEAK RAM USAGE: 44.7K FLASH USAGE: 308.2K
- Aynı parametreler üzerinde denenerek, benzer doğruluk oranı ve işlem hızına ulaşılmaya çalışıldı.

[61]:
```
### Preprocessing islemi ###

from keras.datasets import cifar10
from keras.utils import to_categorical

import numpy as np

def filter_labels(X, y, labels_to_remove):
    # Create mask
    mask = ~np.isin(y, labels_to_remove).flatten()

    # Apply mask
    X_filtered = X[mask]
    y_filtered = y[mask]

    return X_filtered, y_filtered

def make_preprocessing(X_train, y_train, X_test, y_test):
    # one hot encode uygula labellar uzerinde
    y_train = to_categorical(y_train)
    y_test = to_categorical(y_test)

    # Resim pixellerini 0-255 arasindan 0-1 arasina float olarak cek
    X_train = X_train.astype('float32')
    X_test = X_test.astype('float32')
    X_train = X_train / 255.0
```

```
    X_test = X_test / 255.0

    return (X_train, y_train), (X_test, y_test)

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

labels_to_remove = []
labels_to_remove = [2, 3, 4, 5, 6, 7, 8, 9]

# Filter the datasets
X_train, y_train = filter_labels(X_train, y_train, labels_to_remove)
X_test, y_test = filter_labels(X_test, y_test, labels_to_remove)

(X_train, y_train), (X_test, y_test) = make_preprocessing(X_train, y_train,␣
↪X_test, y_test)





### Model egitme islemi ###
from tensorflow_model_optimization.python.core.keras.compat import keras
from keras.losses import categorical_crossentropy

# Used link for this model: https://studio.edgeimpulse.com/public/51070/latest/
↪acquisition/training?page=1
model = keras.Sequential([
    keras.layers.InputLayer(input_shape=(32, 32, 3)),

    keras.layers.Conv2D(32, (3, 1), padding='same', activation='relu'),
    keras.layers.Conv2D(64, (3, 1), padding='same', activation='relu'),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(len(y_train[0]), activation='softmax')
])


model.compile(optimizer='adam', loss=categorical_crossentropy,␣
↪metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, batch_size=64, validation_data=(X_test,␣
↪y_test), verbose=1)

model.save('models/cifar_model_final_2.keras')

print("Model Dosyadan Yukleniyor...\n")
loaded_model = keras.models.load_model('models/cifar_model_final_2.keras')
```

```python
print("Model Degerlendiriliyor...")
_, acc = loaded_model.evaluate(X_test, y_test, verbose=1)
print('Dogruluk(Accuracy) Yuzdelik Oran: %.3f' % (acc * 100.0))

model.summary()




### Pruning ve Fine-tuning islemi ###
import tensorflow_model_optimization as tfmot

prune_low_magnitude = tfmot.sparsity.keras.prune_low_magnitude

batch_size = 128
epochs = 1
validation_split = 0.1 # 10% of training set will be used for validation set.

num_images = X_train.shape[0] * (1 - validation_split)
end_step = np.ceil(num_images / batch_size).astype(np.int32) * epochs

# Define model for pruning.
pruning_params = {
      'pruning_schedule': tfmot.sparsity.keras.
 ↪PolynomialDecay(initial_sparsity=0.50,
                                                               final_sparsity=0.
 ↪80,
                                                               begin_step=0,
                                                              ␣
 ↪end_step=end_step)
}


loaded_model = keras.models.load_model('models/cifar_model_final_2.keras')

model_for_pruning = prune_low_magnitude(loaded_model, **pruning_params)

# `prune_low_magnitude` requires a recompile.
model_for_pruning.compile(optimizer='adam',
              loss=keras.losses.CategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])

import tempfile

logdir = tempfile.mkdtemp()
```

```
callbacks = [
  tfmot.sparsity.keras.UpdatePruningStep(),
  tfmot.sparsity.keras.PruningSummaries(log_dir=logdir),
]

model_for_pruning.fit(X_train, y_train,
                  batch_size=batch_size, epochs=epochs,␣
 ↪validation_split=validation_split,
                  callbacks=callbacks)

model_for_pruning.save('models/cifar_pruned_finetuned_2.keras')

print("Model Degerlendiriliyor...")
_, acc_pruned = model_for_pruning.evaluate(X_test, y_test, verbose=1)
print('Dogruluk(Accuracy) Yuzdelik Oran: %.3f' % (acc_pruned * 100.0))

model_for_pruning.summary()






### Quantization islemi ###
import tensorflow
from tensorflow_model_optimization.python.core.keras.compat import keras
import numpy as np
import tensorflow_model_optimization as tfmot

model_for_export = tfmot.sparsity.keras.strip_pruning(model_for_pruning)


def representative_dataset_gen():
    for _ in range(100):
        # Get a random batch of images from your dataset
        data = X_train[np.random.choice(X_train.shape[0], 1, replace=False)]
        yield [data.astype(np.float32)]

converter = tensorflow.lite.TFLiteConverter.from_keras_model(model_for_export)
converter.optimizations = [tensorflow.lite.Optimize.DEFAULT]


converter.representative_dataset = representative_dataset_gen


# Use mixed precision quantization (weights in int8, activations in float32)
```

```
converter.target_spec.supported_ops = [
    tf.lite.OpsSet.TFLITE_BUILTINS,  # allow float32 activations
    tf.lite.OpsSet.TFLITE_BUILTINS_INT8  # allow int8 weights
]


tflite_model = converter.convert()
with open('quantized_model_2.tflite', 'wb') as f:
    f.write(tflite_model)








### Tflite modelinin dogruluk degerini kontrol etme ###
import numpy as np
import tensorflow as tf
from tensorflow import keras

# Load the quantized TFLite model
interpreter = tf.lite.Interpreter(model_path="quantized_model_2.tflite")
interpreter.allocate_tensors()

# Get input and output tensors
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Prepare the test dataset
(_, _), (X_test, y_test) = keras.datasets.cifar10.load_data()
X_test = X_test.astype('float32') / 255.0
y_test = keras.utils.to_categorical(y_test, 10)

# Function to run inference on a single input
def run_inference(input_data):
    input_data = np.expand_dims(input_data, axis=0).astype(np.float32)
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()
    output_data = interpreter.get_tensor(output_details[0]['index'])
    return output_data

# Calculate accuracy
correct_predictions = 0
total_predictions = len(X_test)

for i in range(total_predictions):
```

```python
    input_data = X_test[i]
    true_label = np.argmax(y_test[i])

    # Get model prediction
    output_data = run_inference(input_data)
    predicted_label = np.argmax(output_data)

    if predicted_label == true_label:
        correct_predictions += 1

accuracy = correct_predictions / total_predictions
print(f"Model accuracy after quantization: {accuracy * 100:.2f}%")
```

```
Epoch 1/10
157/157 [==============================] - 12s 74ms/step - loss: 0.4421 -
accuracy: 0.8123 - val_loss: 0.2960 - val_accuracy: 0.8720
Epoch 2/10
157/157 [==============================] - 13s 80ms/step - loss: 0.2615 -
accuracy: 0.8896 - val_loss: 0.2428 - val_accuracy: 0.8955
Epoch 3/10
157/157 [==============================] - 13s 80ms/step - loss: 0.2149 -
accuracy: 0.9128 - val_loss: 0.2104 - val_accuracy: 0.9075
Epoch 4/10
157/157 [==============================] - 13s 80ms/step - loss: 0.1871 -
accuracy: 0.9266 - val_loss: 0.2042 - val_accuracy: 0.9150
Epoch 5/10
157/157 [==============================] - 12s 78ms/step - loss: 0.1407 -
accuracy: 0.9449 - val_loss: 0.2026 - val_accuracy: 0.9210
Epoch 6/10
157/157 [==============================] - 12s 79ms/step - loss: 0.1088 -
accuracy: 0.9613 - val_loss: 0.2375 - val_accuracy: 0.9080
Epoch 7/10
157/157 [==============================] - 12s 77ms/step - loss: 0.0847 -
accuracy: 0.9698 - val_loss: 0.2249 - val_accuracy: 0.9170
Epoch 8/10
157/157 [==============================] - 12s 76ms/step - loss: 0.0603 -
accuracy: 0.9809 - val_loss: 0.3056 - val_accuracy: 0.9025
Epoch 9/10
157/157 [==============================] - 12s 78ms/step - loss: 0.0401 -
accuracy: 0.9884 - val_loss: 0.2677 - val_accuracy: 0.9200
Epoch 10/10
157/157 [==============================] - 13s 80ms/step - loss: 0.0230 -
accuracy: 0.9947 - val_loss: 0.2824 - val_accuracy: 0.9235
Model Dosyadan Yukleniyor…

Model Degerlendiriliyor…
63/63 [==============================] - 1s 6ms/step - loss: 0.2824 - accuracy:
```

```
0.9235
Dogruluk(Accuracy) Yuzdelik Oran: 92.350
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_20 (Conv2D)          (None, 32, 32, 32)        320

 conv2d_21 (Conv2D)          (None, 32, 32, 64)        6208

 flatten_4 (Flatten)         (None, 65536)             0

 dense_8 (Dense)             (None, 64)                4194368

 dense_9 (Dense)             (None, 2)                 130

=================================================================
Total params: 4201026 (16.03 MB)
Trainable params: 4201026 (16.03 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
71/71 [==============================] - 9s 113ms/step - loss: 0.0184 -
accuracy: 0.9954 - val_loss: 0.0081 - val_accuracy: 1.0000
Model Degerlendiriliyor…
63/63 [==============================] - 1s 12ms/step - loss: 0.3120 - accuracy:
0.9240
Dogruluk(Accuracy) Yuzdelik Oran: 92.400
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 prune_low_magnitude_conv2d  (None, 32, 32, 32)        610
 _20 (PruneLowMagnitude)

 prune_low_magnitude_conv2d  (None, 32, 32, 64)        12354
 _21 (PruneLowMagnitude)

 prune_low_magnitude_flatte  (None, 65536)             1
 n_4 (PruneLowMagnitude)

 prune_low_magnitude_dense_  (None, 64)                8388674
 8 (PruneLowMagnitude)

 prune_low_magnitude_dense_  (None, 2)                 260
 9 (PruneLowMagnitude)

=================================================================
Total params: 8401899 (32.05 MB)
```

```
Trainable params: 4201026 (16.03 MB)
Non-trainable params: 4200873 (16.03 MB)
_________________________________________________________________
INFO:tensorflow:Assets written to: /tmp/tmpbgldadg1/assets

INFO:tensorflow:Assets written to: /tmp/tmpbgldadg1/assets
/home/d3v3lop3r/.local/lib/python3.10/site-
packages/tensorflow/lite/python/convert.py:964: UserWarning: Statistics for
quantized inputs were expected, but not specified; continuing anyway.
  warnings.warn(
W0000 00:00:1717931610.824218   13266 tf_tfl_flatbuffer_helpers.cc:390] Ignored
output_format.
W0000 00:00:1717931610.824230   13266 tf_tfl_flatbuffer_helpers.cc:393] Ignored
drop_control_dependency.
2024-06-09 14:13:30.824340: I tensorflow/cc/saved_model/reader.cc:83] Reading
SavedModel from: /tmp/tmpbgldadg1
2024-06-09 14:13:30.824766: I tensorflow/cc/saved_model/reader.cc:51] Reading
meta graph with tags { serve }
2024-06-09 14:13:30.824775: I tensorflow/cc/saved_model/reader.cc:146] Reading
SavedModel debug info (if present) from: /tmp/tmpbgldadg1
2024-06-09 14:13:30.828173: I tensorflow/cc/saved_model/loader.cc:234] Restoring
SavedModel bundle.
2024-06-09 14:13:30.849789: I tensorflow/cc/saved_model/loader.cc:218] Running
initialization op on SavedModel bundle at path: /tmp/tmpbgldadg1
2024-06-09 14:13:30.856323: I tensorflow/cc/saved_model/loader.cc:317]
SavedModel load for tags { serve }; Status: success: OK. Took 31981
microseconds.
fully_quantize: 0, inference_type: 6, input_inference_type: FLOAT32,
output_inference_type: FLOAT32

Model accuracy after quantization: 18.47%
```

- Full-integer quantization yaptıktan sonra modelin doğruluk değeri gene %10 seviyesine düştü. Bundan dolayı yukarıdaki kodda yapıldığı şekilde sadece ağırlıkları quantize edecek şekilde yapıldı ancak böylelikle doğruluk değeri yukarıda olduğu gibi 18.47% olarak belirlendi.
- Bu model ile Edge Impulse Studio ile eğitilen modelde olduğu gibi bir doğruluk oranına ulaşılamadı.