TinyML Midterm Project

April 23, 2024

1 Kütüphaneleri Ekleme(Imports)

[ ]:
```
import sys
from matplotlib import pyplot
import numpy as np
from keras.datasets import cifar10
from keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten,␣
↪BatchNormalization, Dropout, Input
from keras.losses import categorical_crossentropy
from keras.callbacks import ModelCheckpoint
from sklearn.metrics import precision_score, recall_score, f1_score,␣
↪confusion_matrix
```

2 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.

[12]:
```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

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("Veriseti preprocessing yapmadan once:")
print("X_test: ", X_test[:1])
print("y_test: ", y_test[:1])
print("\n")

# 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

print("Veriseti preprocessing yaptiktan sonra:")
print("X_test: ", X_test[:1])
print("y_test: ", y_test[:1])
print("\n")
```

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

Veriseti preprocessing yapmadan once:
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:
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.]]
```

3 Model Eğitimi ve Modeli Kaydetme

3.1 Katman Açıklamaları

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.

[ ]:
```
input_shape = (32, 32, 3)
input_layer = Input(shape=input_shape)

model = Sequential([input_layer])

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

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

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

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

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

history = model.fit(X_train, y_train, epochs=100, batch_size=64,␣
 ↪validation_data=(X_test, y_test), verbose=1)
```

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

4 Model Dosyadan Yükleme ve Doğruluk Değeri Hesaplama

[6]:
```
print("Model Dosyadan Yukleniyor...\n")
loaded_model = load_model('models/cifar_model_midterm.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             10s 31ms/step -
accuracy: 0.8795 - loss: 0.3862
Dogruluk(Accuracy) Yuzdelik Oran: 88.040
```

5 Model Parametreleri Analizi

[20]:
```
loaded_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
batch_normalization (BatchNormalization)	(None, 32, 32, 32)	128
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9,248
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 32)	128
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 32)	128

conv2d_2 (Conv2D)	(None, 16, 16, 64)	18,496
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36,928
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_1 (Dropout)	(None, 8, 8, 64)	0
conv2d_4 (Conv2D)	(None, 8, 8, 128)	73,856
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 128)	512
conv2d_5 (Conv2D)	(None, 8, 8, 128)	147,584
batch_normalization_5 (BatchNormalization)	(None, 8, 8, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_2 (Dropout)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262,272
batch_normalization_6 (BatchNormalization)	(None, 128)	512
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

Total params: 1,656,064 (6.32 MB)

Trainable params: 551,658 (2.10 MB)

Non-trainable params: 1,088 (4.25 KB)

Optimizer params: 1,103,318 (4.21 MB)

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

[22]:
```
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.

[23]:
```
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             9s 29ms/step
[3 8 8 … 5 1 7]
[3 8 8 … 5 1 7]
Precision: 0.8804
Recall: 0.8804
F1 Score: 0.8804
Confusion Matrix:
[[870   8  26  12  16   1   6   4  48   9]
```

```
[  4 941   1   3   2   2   5   0  10  32]
[ 24   0 794  37  47  37  42  15   4   0]
[  7   1  23 783  22 114  32  10   4   4]
[  2   1  17  35 890  17  19  15   4   0]
[  3   0  16  88  29 835  10  18   0   1]
[  2   1  12  43   8  12 917   1   3   1]
[  5   0   5  25  23  30   5 906   1   0]
[ 19   4   2   5   3   1   5   1 947  13]
[ 10  35   3  10   0   1   4   1  15 921]]
```