Multi-Class University Food Classification using Transfer Learning (MobileNetV2) with CNN

Parsa Yousefi Nejad

TABLE OF CONTENTS

TABLE OF CONTENTS

OBJECTIVES OF THE PROJECT

MobileNetV2: Inverted Residuals and Linear Bottlenecks

Mark Sandler   Andrew Howard   Menglong Zhu   Andrey Zhmoginov   Liang-Chieh Chen
Google Inc.
{sandler, howarda, menglong, azhmogin, lcchen}@google.com

Abstract

In this paper we describe a new mobile architecture, MobileNetV2, that improves the state of the art performance of mobile models on multiple tasks and benchmarks as well as across a spectrum of different model sizes. We also describe efficient ways of applying these mobile models to object detection in a novel framework we call SSDLite. Additionally, we demonstrate how to build mobile semantic segmentation models through a reduced form of DeepLabv3 which we call Mobile DeepLabv3.

applications.

This paper introduces a new neural network architecture that is specifically tailored for mobile and resource constrained environments. Our network pushes the state of the art for mobile tailored computer vision models, by significantly decreasing the number of operations and memory needed while retaining the same accuracy.

Our main contribution is a novel layer module: the inverted residual with linear bottleneck. This module takes as an input a low-dimensional compressed representation which is first expanded to high dimension and filtered with a lightweight depthwise convo-

Technical
Model Summery

```
Model: "sequential"

 Layer (type)                  Output Shape              Param #

 mobilenetv2_1.00_224 (Func    (None, 7, 7, 1280)        2257984
 tional)

 global_average_pooling2d (    (None, 1280)              0
 GlobalAveragePooling2D)

 dense (Dense)                 (None, 256)               327936

 dropout (Dropout)             (None, 256)               0

 dense_1 (Dense)               (None, 32)                8224

 dense_2 (Dense)               (None, 9)                 297
                                              Add Markdown Cell

Total params: 2594441 (9.90 MB)
Trainable params: 336457 (1.28 MB)
Non-trainable params: 2257984 (8.61 MB)
```

Dataset and train/val Generators

```python
train_generator = datagen_train.flow_from_directory(
    '/Users/parsa/Desktop/Food',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

val_generator = datagen_val.flow_from_directory(
    '/Users/parsa/Desktop/Food',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation',
)
```

Data Augmentation


```
datagen_train = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    brightness_range=[0.5, 1.5],
    channel_shift_range=0.2,
    vertical_flip=True,
    fill_mode='nearest',
)
datagen_val = ImageDataGenerator(rescale=1./255, validation_split=0.15)
```
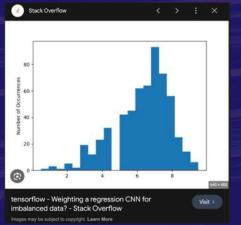
Weight Balancing

tensorflow - Weighting a regression CNN for imbalanced data? - Stack Overflow

Images may be subject to copyright. Learn More

Dealing with Imbalanced Data in TensorFlow: Class Weights | by Angel Igareta | Towards Data Science

Balancing Class Weights (*inorder to improve perrformace on minority classes*)

+ Code   + Markdown
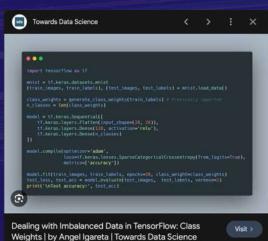
```python
labels = train_generator.classes

class_counts = np.bincount(labels)
total_samples = np.sum(class_counts)
class_weights = total_samples / (len(class_counts) * class_counts)

class_weight_dict = dict(zip(range(len(class_counts)), class_weights))
```

Start Training the Model

- Fine-tuing the model on my small Dataset

```python
his=model.fit(train_generator, epochs=30, validation_data=val_generator, class_weight=class_weight_dict)
```

Training Steps

Start Training the Model

- Fine-tuing the model on my small Dataset

```python
his=model.fit(train_generator, epochs=30, validation_data=val_generator, class_weight=class_weight_dict)
```

```
Epoch 1/30
6/6 [==============================] - 29s 5s/step - loss: 10.8897 - accuracy: 0.1304 - val_loss: 9.7474 - val_accuracy: 0.3000
Epoch 2/30
6/6 [==============================] - 25s 4s/step - loss: 9.6405 - accuracy: 0.2174 - val_loss: 9.0709 - val_accuracy: 0.1500
Epoch 3/30
6/6 [==============================] - 25s 4s/step - loss: 8.5484 - accuracy: 0.2298 - val_loss: 8.2256 - val_accuracy: 0.0500
Epoch 4/30
6/6 [==============================] - 25s 5s/step - loss: 8.0764 - accuracy: 0.1988 - val_loss: 7.5761 - val_accuracy: 0.2500
Epoch 5/30
6/6 [==============================] - 25s 4s/step - loss: 7.3981 - accuracy: 0.2174 - val_loss: 6.9731 - val_accuracy: 0.3500
Epoch 6/30
6/6 [==============================] - 25s 5s/step - loss: 6.7613 - accuracy: 0.2236 - val_loss: 6.4106 - val_accuracy: 0.2500
```

```python
his=model.fit(train_generator, epochs=30, validation_data=val_generator, class_weight=class_weight_dict, callbacks = [callback])
```

```
Epoch 1/30
3/3 [==============================] - 24s 10s/step - loss: 1.3040 - accuracy: 0.7888 - val_loss: 0.8583 - val_accuracy: 1.0000
Epoch 2/30
3/3 [==============================] - 21s 7s/step - loss: 1.1557 - accuracy: 0.8447 - val_loss: 0.8613 - val_accuracy: 1.0000
Epoch 3/30
3/3 [==============================] - 21s 7s/step - loss: 1.2200 - accuracy: 0.8075 - val_loss: 0.8619 - val_accuracy: 0.9655
Epoch 4/30
3/3 [==============================] - 21s 9s/step - loss: 1.2884 - accuracy: 0.7391 - val_loss: 0.8473 - val_accuracy: 0.9655
```

```python
datagen_train = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    brightness_range=[0.5, 1.5],
    channel_shift_range=0.2,
    vertical_flip=True,
    fill_mode='nearest',
)
datagen_val = ImageDataGenerator(rescale=1./255, validation_split=0.15)

train_generator = datagen_train.flow_from_directory(
    '/Users/parsa/Desktop/Food',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

val_generator = datagen_val.flow_from_directory(
    '/Users/parsa/Desktop/Food',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation',
)
```

(just Original Images)

```python
datagen_train2 = ImageDataGenerator(rescale=1./255)
datagen_val2 = ImageDataGenerator(rescale=1./255, validation_split=0.20)

train_generator2 = datagen_train2.flow_from_directory(
    '/Users/parsa/Desktop/Food',
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

val_generator2 = datagen_val2.flow_from_directory(
    '/Users/parsa/Desktop/Food',
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

Final Results

Model Evaluation


```
2/2 [==============================] - 1s 234ms/step
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       0.83      1.00      0.91        10
           2       1.00      0.89      0.94         9
           3       1.00      1.00      1.00         2
           4       0.92      0.85      0.88        13
           5       0.75      1.00      0.86         3
           6       1.00      1.00      1.00         4
           7       0.93      0.88      0.90        16
           8       1.00      1.00      1.00         1

    accuracy                           0.92        64
   macro avg       0.94      0.96      0.94        64
weighted avg       0.93      0.92      0.92        64

[[ 6  0  0  0  0  0  0  0  0]
 [ 0 10  0  0  0  0  0  0  0]
 [ 0  0  8  0  0  0  0  1  0]
 [ 0  0  0  2  0  0  0  0  0]
 [ 0  2  0  0 11  0  0  0  0]
 [ 0  0  0  0  0  3  0  0  0]
 [ 0  0  0  0  0  0  4  0  0]
 [ 0  0  0  0  1  1  0 14  0]
 [ 0  0  0  0  0  0  0  0  1]]
```

```
1/1 [==============================] - 0s 250ms/step
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         7
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         1
           4       1.00      1.00      1.00        12
           7       1.00      1.00      1.00         7

    accuracy                           1.00        32
   macro avg       1.00      1.00      1.00        32
weighted avg       1.00      1.00      1.00        32
```

Model in Action

THANKS!

References:
Andrew Deep
Jose Deep
StackOverflow
Tensorflow
ChatGPT 4O

Parsa Yousefi Nejad
140053611048