Image Based Biometry Assignment #2

Tilen Kavčič
Faculty of Computer and Information Science
Večna pot 113, 1000 Ljubljana

December 1, 2021

1 Introduction

We look at face recognition on WIDER FACE database using convolutional neural networks (CNNs).

2 Data acquisition

We selected the WIDER FACE dataset and extracted the images and annotations. For easier training and validation we picked out a subset of the images. More specifically we selected 996 training images from the demonstration category and 248 validation images from the same category.

3 Preprocessing

For speed we tried to minimize the resolution of the images without hampering training. Image size, for this reason, was set to minimum 224x224. We use 'ImageDataGenerator' to rescale the images [1]. We separate the images into training and validation sets. The code is shown in appendix A.

4 The model

We created a base model from the pre-trained CNN model MobileNet V2 [3]. This model was created from the training data compiled in previous steps.

The model was developed at Google and was pre-trained on the ImageNet dataset, a large dataset of 1.4M images and 1000 classes of web images.

Out base model is a 2D Convolution network (32 nodes, 3 Kernel size, Activation Function). We set the probability of each non-contributing node being dropped to 20%. We use the Softmax activation function. We set Epochs to 10 (model trained in 10 iterations). The code is shown in appendix B.

5 Evaluation

We visualize the learning curves of the model. They are shown in figure 1. The accuracy of the model after 10 epochs is 0.802. The code is shown in appendix C.

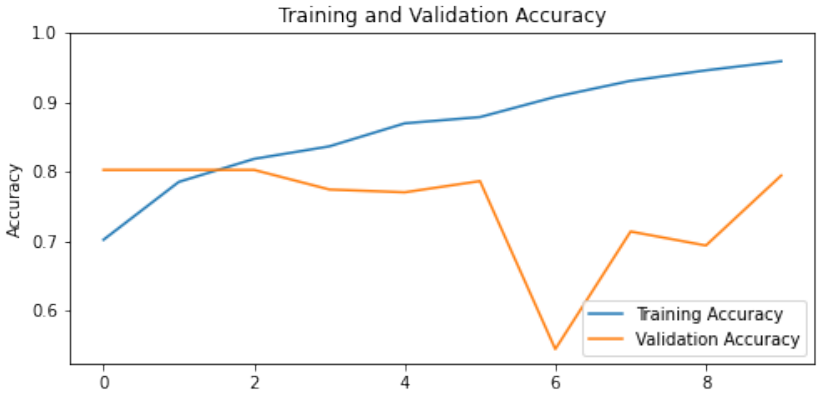

Figure 1: Training and valuation accuracy.

References

[1] devDeejay, "Building A Facial Recognition Machine Learning Model Using TensorFlow," Medium, 17-Mar-2021. [Online]. Available: https://medium.com/softway-blog/building-a-facial-recognition-machine-learning-model-using-tensorflow-6e62fb349794. [Accessed: 13-Dec-2021].

[2] Google Codelabs, "Recognize Flowers With TensorFlow Lite On Android — Google Codelabs," Google Codelabs. [Online]. Available: https://codelabs.developers.google.com/codelabs/recognize-flowers-with-tensorflow-on-android/#2. [Accessed: 13-Dec-2021].

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Mar. 2019, doi: 10.1109/cvpr.2018.00474.

Appendices

A Preprocessing

```
IMAGE_SIZE = 224
BATCH_SIZE = 5

# rescales the images
data_generator = tf.keras.preprocessing.image.
    ImageDataGenerator(
        rescale=1. / 255,
        validation_split=0.2)

train_generator = data_generator.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='training')

val_generator = data_generator.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='validation')

for image_batch, label_batch in train_generator:
    break
```

```
print(train_generator.class_indices)

labels = '\n'.join(sorted(
    train_generator.class_indices.keys())
)

with open('labels.txt', 'w') as f:
    f.write(labels)
```

B The model

```
IMG_SHAPE = (IMAGE_SIZE, IMAGE_SIZE, 3)
base_model = tf.keras.applications.
                MobileNetV2(input_shape=IMG_SHAPE,
                            include_top=False,
                            weights='imagenet')
base_model.trainable = False
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.Conv2D(32, 3, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(3, activation='softmax')
])
model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
print('Number of trainable variables = {}'.
        format(len(model.trainable_variables)))
epochs = 10
history = model.fit(train_generator,
                    epochs=epochs,
                    validation_data=val_generator)
```

C Visualisation

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()), 1])
plt.title('Training and Validation Accuracy')
print(val_acc)
```