Eye Spy:
A Convolutional Neural Network Image Classifier for Headshot Photographers

Introduction

I have a side business as a photographer, focusing on headshot photography for film and theatre actors. In a typical session, headshot photographers will take around 300 to 600 photographs. Invariably, the subject is blinking or has partially closed eyes in some of these images.

These photos aren't usually the most flattering images, so after every session, I manually edit out these photographs before sending them to the client. This can take 30 minutes to an hour for each session.

The author

Hourly shooting rates for headshot photographers typically start at $100/hour, so maximizing shooting time and minimizing editing time is always a priority.

I was considering hiring an assistant to do this tedious editing work, but first, I wanted to see if I could create a convolutional neural network to filter through the photographs and classify them, saving me time and expense.

Objective

To build a convolutional neural network that correctly classifies images where the subject is blinking with 80% accuracy, and deploy it as a web app.

Data Collection & Curation

I've been offering headshot sessions for several years, so I have a large archive of images. I manually labelled photographs from my own collection of archived sessions, splitting them into BLINKING and NON_BLINKING groups. Each group included 500 images.

I aimed to create a dataset that was diverse in age, gender, and racial groups. The approximate breakdown was:

- **Gender** Female: 54%, Male 43%, Non-Binary: 3%
- **Race** White: 60%, Black: 27%, Asian: 8%, Other: 3%
- **Age** < 25: 8%, 25-50: 49%, 50+: 38%

For the privacy of my clients I am not including the images on github, but this is an example of 'blinking' and 'non-blinking' photos from the data set:

Examples from the data set

Initial Model

I built the initial model using the Gradient Machine Learning Platform on Paperspace. I used Keras with TensorFlow 2.0 to build a three layer sequential model.

INITIAL MODEL METRICS:

- Train Accuracy: 0.817, Test Accuracy: 0.777
- Precision: 0.778523
- Recall: 0.773333
- F1 score: 0.775920

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 176, 98, 256)      2560
_________________________________________________________________
activation_18 (Activation)   (None, 176, 98, 256)      0
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 88, 49, 256)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 86, 47, 256)       590080
_________________________________________________________________
activation_19 (Activation)   (None, 86, 47, 256)       0
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 43, 23, 256)       0
_________________________________________________________________
flatten_6 (Flatten)          (None, 253184)            0
_________________________________________________________________
dense_12 (Dense)             (None, 64)                16203840
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 65
_________________________________________________________________
activation_20 (Activation)   (None, 1)                 0
=================================================================
Total params: 16,796,545
Trainable params: 16,796,545
Non-trainable params: 0
```

Initial model summary

Model Tuning

I tuned the model using the Keras Hyperband tuner, resulting in an optimal learning rate of .0001 over 10 epochs.

Final Model Metrics

- Train Accuracy: 0.86, Test Accuracy: 0.79
- Precision: 0.77
- Recall: 0.82
- F1 score: 0.775

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 17800)             0
_________________________________________________________________
dense (Dense)                (None, 256)               4557056
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2570
=================================================================
Total params: 4,559,626
Trainable params: 4,559,626
Non-trainable params: 0
_________________________________________________________________
```

Final Model Summary

Conclusion

The convolutional neural network was able to classify blinks versus non-blinks with 79% accuracy on the testing data set.

Next Steps

I plan to go back to the initial model and see which images are being misclassified. I can then provide more training examples of similar images, which may improve model performance.

I would also like to add more data to the data set. I will add an additional 250 images to each set and retrain.

After improving model performance, I'd like to build a more robust app that will import a large number of images, and filter them into two groups for final review. This could either be a web app, an MacOS app, or a plugin to an existing photo editing application.

I view ‘blinks vs non-blinks’ as the first feature of a more comprehensive application that can classify images that are out-of-focus, poorly framed, or have motion blur. Ideally, over time it can become an indispensable virtual assistant for photographers.

Credits

Thank you to my Springboard mentor Blake Arensdorf for helping me to hone my skills, and to the entire Springboard team for their guidance and support.