Image Classification

By Erik Paulson

The Data

For this project, I used this fish image dataset from O. Ulucan, D. Karakaya, M. Turkan posted on Kaggle

https://www.kaggle.com/datasets/crowww/a-large-scale-fish-dataset

- 9 different classes of fish
- 1000 images per class

The purpose

The purpose of this classification model is to clearly identify the type of fish for any purpose, such as:

1. For identification purposes while fishing (to show to game wardens, to know what you have caught if new to the area, etc)

2. For personal diet usage (correctly identify the fish so the nutritional information can be looked up for constructing a meal)

Methodology

Tested a few different methods of classification:

1. Logistic Regression

2. Deep Learning Sequential Model

3. Deep Learning Transfer Learning

My Process

I wanted the model to be able to take in as much data as possible:

Split 1: 9010 training images and 430 test images

But I also wanted to see if a more balanced ratio would help accuracy:

Split 2: 7600 training images and 1830 test images

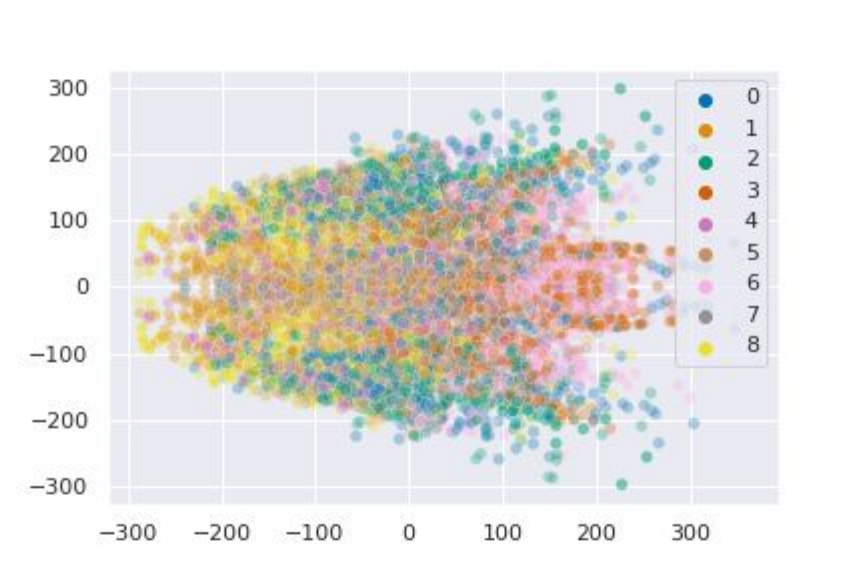
The main metric I wanted to focus on was accuracy, since I mostly care about how many images the model gets right.

Logistic Regression

Cons:

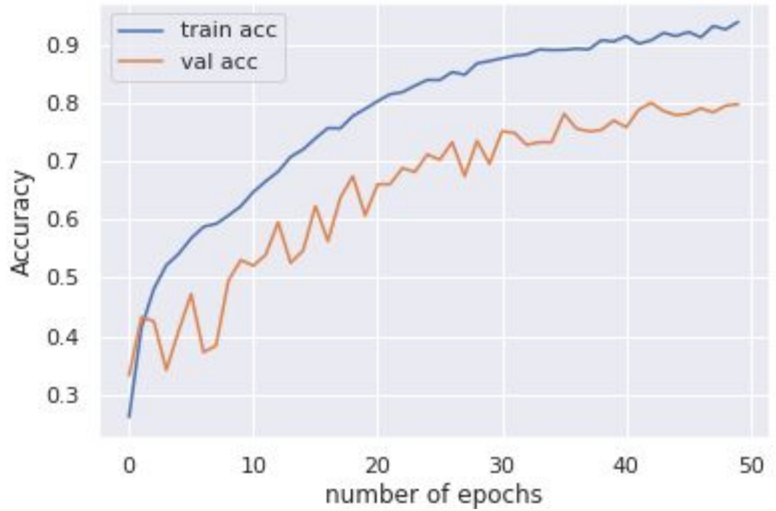
- Low accuracy
- High time to process images (convert to array, flatten, label)
- Test accuracy of ~.25

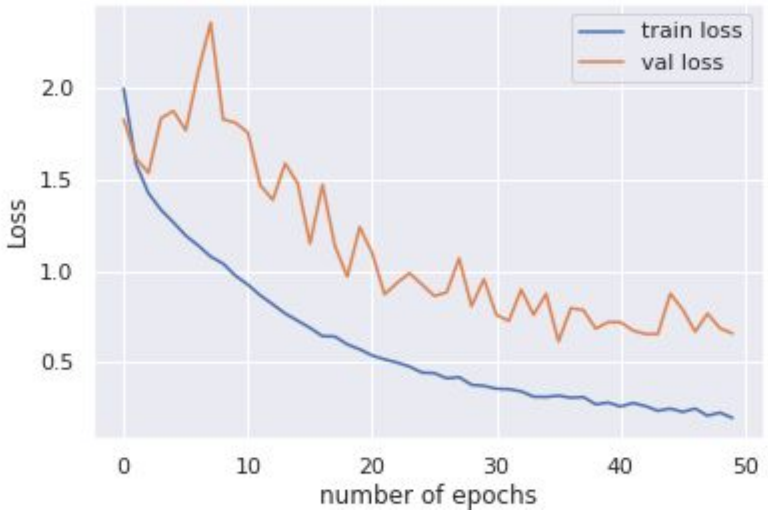
Plot of fish images transformed using PCA, very difficult to discern the categories



Split One Sequential Model

Maxed out at about .75 test accuracy

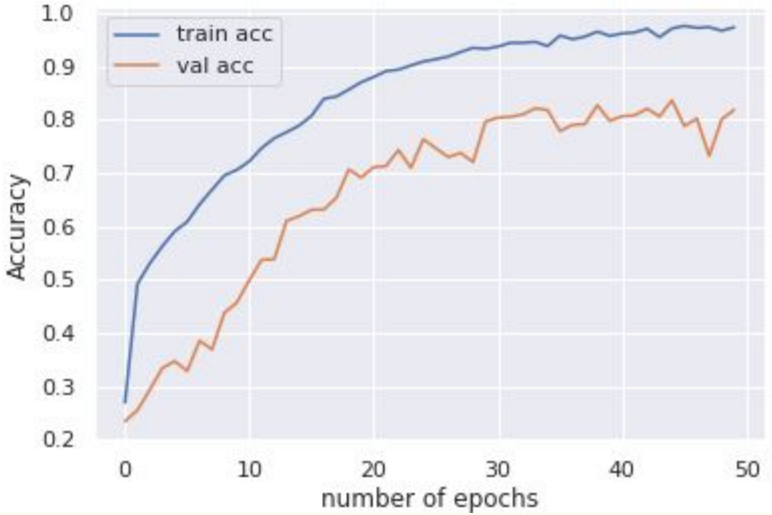


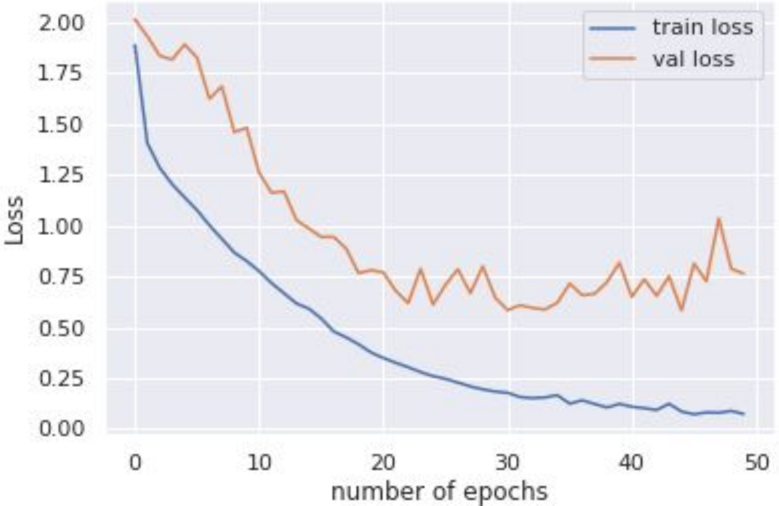


Split Two Sequential Model

Notice the number of epochs required for

Decent accuracy and loss

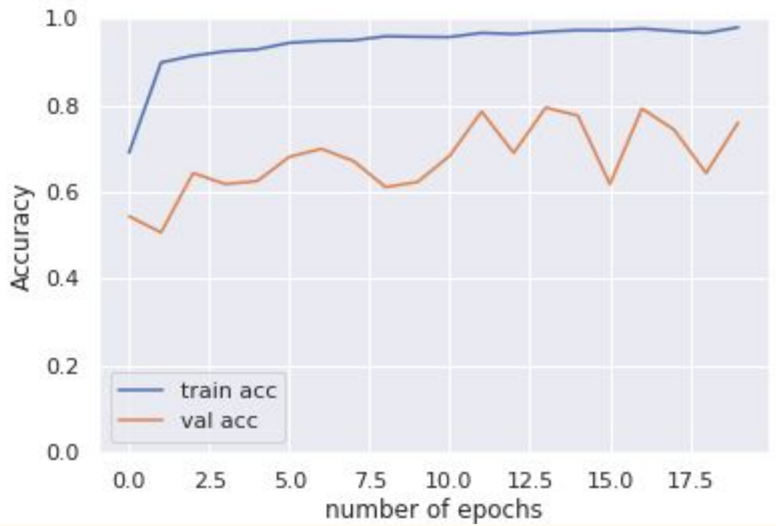


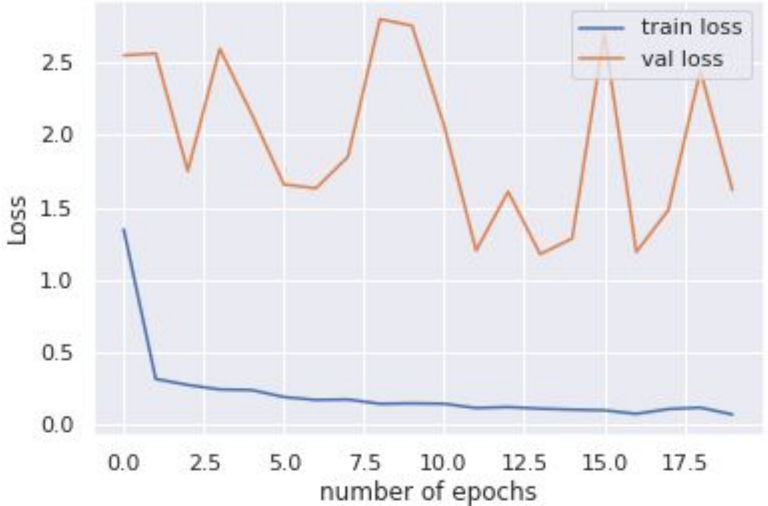


Maxed out at about .80 test accuracy

Split One Transfer Learning

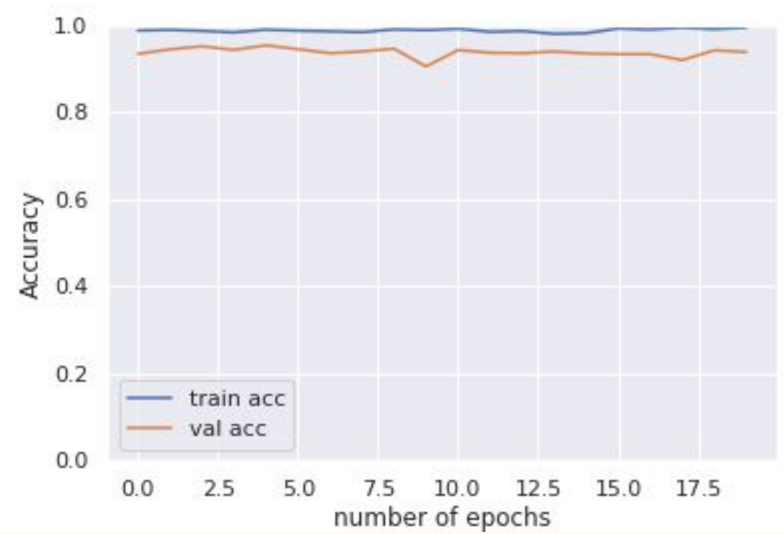
- Higher initial accuracy, fewer epochs needed
- Test accuracy maxed at about .80

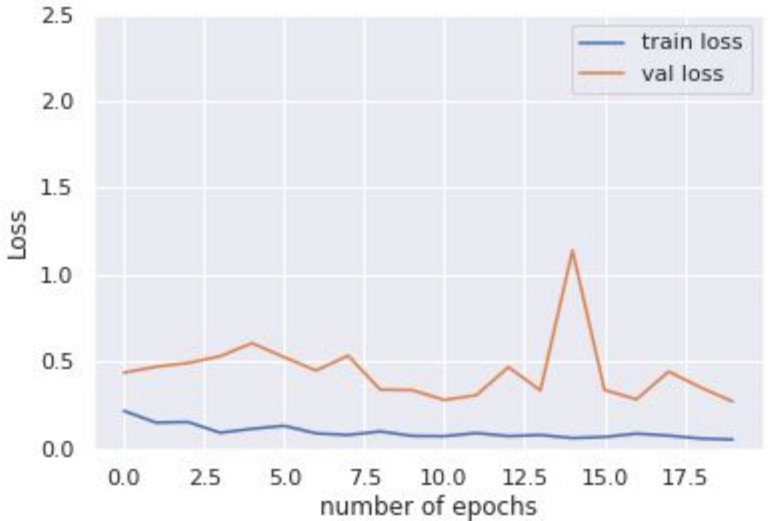




Split Two Transfer Learning

- Fewer epochs needed to train, very high initial test accuracy (over .90)





Future Use

I think the next steps of using this model would be:

1. Further refining to try and achieve a maximum accuracy of .95+ so the identification can be trusted by anyone using the model

2. Adding more data points to try widen the models identification power (there are thousands of fish species, how will it perform with a much higher number of classes?)

Conclusions

- Transfer learning is the best approach to solve this particular problem.

- The additional speed of utilizing transfer learning to train a model to high accuracy is beneficial.

- While neural networks like many data points, make sure to have balanced splits. Having a balanced train/test split improves accuracy and reduced overfitting.

Appendix

```
[ ] model2.summary()

 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten_5 (Flatten)         (None, 25088)             0

 dense_11 (Dense)            (None, 4096)              102764544

 dropout_1 (Dropout)         (None, 4096)              0

 dense_12 (Dense)            (None, 4096)              16781312

 batch_normalization_189 (Ba  (None, 4096)             16384
 tchNormalization)

 dense_13 (Dense)            (None, 9)                 36873

=================================================================
Total params: 134,313,801
Trainable params: 119,590,921
Non-trainable params: 14,722,880
_________________________________________________________________
```

Appendix

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 10)      280

 max_pooling2d (MaxPooling2D  (None, 112, 112, 10)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 20)      1820

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 20)       0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 30)        5430

 global_average_pooling2d (G  (None, 30)               0
 lobalAveragePooling2D)

 dense (Dense)               (None, 20)                620

 dense_1 (Dense)             (None, 9)                 189

=================================================================
Total params: 8,339
Trainable params: 8,339
Non-trainable params: 0
_________________________________________________________________
```